



資料庫開發人員指南

Amazon Redshift



Amazon Redshift: 資料庫開發人員指南

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能附屬於 Amazon，或與 Amazon 有合作關係，亦或受到 Amazon 贊助。

Table of Contents

簡介	1
使用 Amazon Redshift 的先決條件	1
Amazon Redshift 架構	1
資料倉儲架構	2
效能	4
單欄式儲存	7
工作負載管理	9
搭配其他服務使用 Amazon Redshift	10
範本資料庫	11
最佳實務	18
進行概念證明	18
步驟 1：設定 POC 的範圍	19
步驟 2：啟動 Amazon Redshift	20
步驟 3：載入資料	21
步驟 4：分析您的資料	22
步驟 5：最佳化	24
設計資料表的最佳實務	25
選擇最佳的排序索引鍵	25
選擇最佳的分佈方式	26
使用自動壓縮	27
定義限制條件	27
使用可能的最小欄位大小	27
在日期欄位中使用日期/時間資料類型	28
載入資料最佳實務	28
瞭解如何載入資料	28
使用 COPY 命令載入資料	29
使用單一 COPY 指令	29
載入資料檔案	29
壓縮您的資料檔案	30
在載入前後驗證資料檔案	30
使用多資料列插入	30
使用大量插入	31
以排序索引鍵順序載入資料	31
將資料載入循序區塊	31

使用時間序列資料表	32
排程維護時段	32
設計查詢的最佳實務	32
遵循顧問建議	34
支援建議程式的 Amazon Redshift 區域	35
檢視建議員建議	36
Advisor 建議	37
教學課程	50
自動資料表最佳化	51
啟用、停用和監控自動資料表最佳化	51
啟用自動資料表最佳化	51
從資料表中移除自動資料表最佳化	52
監控自動資料表最佳化	52
資料欄壓縮	53
壓縮編碼	54
測試壓縮編碼	63
資料分佈	68
資料分佈概念	69
分佈樣式	70
檢視分佈樣式	71
評估查詢模式	73
指定分佈樣式	73
評估查詢計畫	74
查詢計劃範例	76
分佈範例	80
排序金鑰	83
多維資料配置排序 (預覽)	84
複合排序索引鍵	85
交錯排序索引鍵	85
資料表限制	86
載入資料	88
使用 載入資料表 COPY	91
登入資料和存取許可	92
準備您的輸入資料	93
從 Amazon S3 載入資料	94
從 Amazon 載入資料 EMR	105

從遠端主機載入資料	111
從 Amazon DynamoDB 中載入	119
驗證資料已正確載入	121
驗證輸入資料	122
利用自動壓縮載入資料表	122
針對縮小資料表進行優化	125
載入預設的欄位值	125
針對資料載入進行故障診斷	126
載入具有連續檔案擷取的資料表 (預覽)	132
使用 載入資料表 DML	134
更新和插入	135
執行深層複製	142
分析資料表	146
自動分析	146
新資料表資料的分析	146
ANALYZE 命令歷史記錄	151
清空資料表	152
自動資料表排序	152
自動清空刪除	153
VACUUM 頻率	154
排序階段和合併階段	154
清空閾值	155
清空類型	155
將真空時間降到最低	155
管理並行寫入操作	163
可序列化隔離	164
寫入和讀/寫操作	168
並行寫入範例	169
教學課程：從 Amazon S3 載入資料	170
必要條件	171
概觀	171
步驟 1：建立叢集	172
步驟 2：下載資料檔案	173
步驟 3：上傳檔案至 Amazon S3 儲存貯體	174
步驟 4：建立範例資料表	175
步驟 5：執行COPY命令	178

步驟 6：清空及分析資料庫	194
步驟 7：清理您的資源	195
Summary	195
卸載資料	197
將資料卸載到 Amazon S3	197
卸載加密的資料檔案	200
以分隔或固定寬度的格式卸載資料	202
重新載入已卸載資料	203
使用者定義的函數	205
UDF 安全性和許可	205
防止UDF命名衝突	206
多載函數名稱	206
避免與內建 Amazon Redshift 函數衝突	206
純量 SQL UDFs	207
範例	207
Scalar Python UDFs	208
範例	209
Python UDF 資料類型	209
Python 語言支援	211
限制	215
記錄錯誤和警告	215
Scalar Lambda UDFs	217
管理 Lambda UDF 安全和許可	218
設定 Lambda 的授權參數 UDFs	218
使用 Amazon Redshift 和 Lambda 之間的JSON介面	220
的使用案例範例 UDFs	222
建立預存程序	224
預存程序概觀	224
命名預存程序	227
安全和權限	228
傳回結果集	229
管理交易	231
捕捉錯誤	243
記錄預存程序日誌	251
限制	251
PL/pgSQL 語言參考	252

PL/pgSQL 參考慣例	252
PL/pgSQL 的結構	253
支援的 PL/pgSQL 陳述式	258
具體化視觀表	273
具體化檢視查詢	275
自動查詢重寫以使用具體化視檢視	276
使用須知	276
限制	277
重新整理具體化視觀表	278
自動重新整理具體化視觀表	281
自動具體化視觀表	282
SQL 自動化具體化視觀表的範圍和考量	283
自動具體化視觀表限制	283
自動具體化視觀表的計費	284
其他資源	284
在具體化視觀表中使用使用者定義的函數 (UDF)	284
在具體化視觀表UDF中參考	284
將擷取串流至具體化視觀表	286
資料如何從串流服務流向 Redshift	286
改善效能的資料剖析最佳實務	287
串流擷取行為和資料類型	287
開始使用 Amazon Kinesis Data Streams 中的串流擷取	290
從 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 開始串流擷取	294
電動汽車站的資料串流擷取教學 (使用 Kinesis)	305
Data Catalog 檢視	309
必要條件	309
End-to-end 範例	311
安全記錄	311
考量事項	317
查詢空間資料	318
教學課程：使用空間SQL函數	321
必要條件	321
步驟 1：建立資料表並載入測試資料	322
步驟 2：查詢空間資料	324
步驟 3：清除您的資源	328
載入 Shapefile	328

術語	329
邊界框	329
幾何有效性	330
幾何簡單性	332
H3	334
考量事項	334
使用聯合查詢來查詢資料	336
開始使用聯合查詢進行 Postgre SQL	337
開始使用聯合查詢來發布 SQL CloudFormation	338
為 Redshift 聯合查詢啟動 CloudFormation 堆疊	338
從外部結構描述查詢資料	339
開始使用聯合查詢至 [我的] SQL	340
建立秘密和IAM角色	341
必要條件	341
使用聯合查詢的範例	343
使用聯合查詢與 Postgre 的範例 SQL	343
使用混合大小寫名稱的範例	346
將聯合查詢與 My 搭配使用的範例 SQL	347
資料類型差異	348
考量事項	352
支援的聯合資料庫版本	353
Amazon Redshift Spectrum	355
Amazon Redshift Spectrum 概觀	355
Amazon Redshift Spectrum 區域	356
Amazon Redshift Spectrum 限制	356
開始使用 Amazon Redshift Spectrum	357
必要條件	358
CloudFormation	358
逐步開始使用 Redshift Spectrum	358
步驟 1. 建立 IAM 角色	359
步驟 2：將IAM角色與叢集建立關聯	362
步驟 3：建立外部結構描述與外部資料表	363
步驟 4：在 Amazon S3 中查詢您的資料	364
啟動 CloudFormation 堆疊，然後查詢資料	367
IAM Amazon Redshift Spectrum 的政策	371
Amazon S3 許可	372

跨帳戶 Amazon S3 許可	372
授予或限制使用 Redshift Spectrum 存取	373
最低許可	374
鏈結IAM角色	375
存取 AWS Glue 資料	376
Redshift Spectrum 和 Lake Formation	384
使用資料篩選條件來實現列層級和儲存格層級安全	385
Amazon Redshift Spectrum 中查詢的資料檔案	386
Redshift Spectrum 的資料格式	386
Redshift Spectrum 的壓縮類型	388
Redshift Spectrum 的加密	388
外部結構描述	389
使用外部目錄	391
外部資料表	395
虛擬資料欄	397
分割 Redshift Spectrum 外部資料表	398
對應至ORC資料欄	403
為 Hudi 管理的資料建立外部資料表	406
建立 Delta Lake 資料的外部資料表	407
使用 Apache Iceberg 資料表	410
使用 Apache Iceberg 資料表時的注意事項	410
支援的資料類型	411
Amazon Redshift Spectrum 查詢效能	413
資料處理選項	416
執行相互關聯子查詢	417
指標	418
查詢疑難排解	418
超過重試次數	419
存取已限流	419
超過資源限制	420
分割的資料表沒有傳回資料列	421
未授權的錯誤	421
不相容的資料格式	421
在 Amazon Redshift DDL中使用 Hive 時的語法錯誤	422
建立暫存資料表的許可	422
無效的範圍	422

無效的 Parquet 版本編號	423
教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料	423
概觀	423
步驟 1：建立包含巢狀資料的外部資料表	424
步驟 2：使用 SQL 擴充功能查詢 Amazon S3 中的巢狀資料	425
巢狀資料使用案例	429
巢狀資料限制 (預覽)	431
序列化複雜的嵌套 JSON	433
HyperLogLog 草圖	436
考量事項	437
限制	437
範例	438
範例：在子查詢中傳回基數	438
範例：從子查詢中的合併草圖傳回 HLLSKETCH 類型	439
範例：從合併多個 HyperLogLog 草圖傳回草圖	439
範例：使用外部資料表透過 S3 資料產生 HyperLogLog 草圖	440
跨資料庫查詢	443
考量事項	445
限制	445
範例	446
搭配查詢編輯器使用跨資料庫查詢	450
資料共用	452
資料共用概述	452
資料共用使用案例	453
不同層級的資料共用	453
資料共用一致性管理	454
在 Amazon Redshift 中使用資料共用時的考量事項	454
叢集加密管理	454
限制	455
提供資料共用的區域	455
資料共用	458
標準資料共用	459
AWS Data Exchange 資料共用	460
AWS Lake Formation 管理的資料共用	463
資料共用生產者和取用者	465
Datashare 詳細資訊	465

連線至取用者資料庫	474
存取共用資料	475
存取共用物件的中繼資料	475
將 Amazon Redshift 資料共用與商業智慧工具整合	476
監視和稽核資料共用	476
將 Amazon Redshift 資料共用與 整合 AWS CloudTrail	477
管理資料共用工作	478
使用 SQL 介面管理資料共用	478
使用主控台管理資料共用	518
使用 管理資料共用 CloudFormation	534
使用主控台透過寫入功能管理資料共用 (預覽)	540
Amazon Redshift 中的半結構化資料	551
SUPER 資料類型的使用案例	551
SUPER 資料類型使用的概念	552
SUPER 資料考量事項	553
SUPER 範例資料集	554
將半結構化資料載入 Amazon Redshift	556
剖析JSON文件至SUPER資料欄	556
使用 COPY 在 Amazon Redshift 中載入JSON資料	557
卸載半結構化資料	562
查詢半結構化資料	563
Navigation (導覽)	564
解除巢狀化查詢	565
物件取消樞紐	566
動態類型	567
寬鬆的語義	570
自我檢查的種類	570
排序依據	572
運算子和函數	572
算術運算子	573
算術函數	573
陣列函數	574
SUPER 組態	575
的 Lax 和嚴格模式 SUPER	576
使用大寫和混合大小寫字母存取JSON欄位	576
剖析選項	578

限制	578
SUPER 資料類型和具體化檢視	581
加速 PartiQL 查詢	581
搭配具體化檢視使用SUPER資料類型的限制	585
機器學習	586
機器學習概述	588
機器學習如何解決問題	588
Amazon Redshift ML 的術語和概念	589
適用於新手和專家的機器學習	590
使用 Amazon Redshift ML 的成本	592
搭配 使用 Amazon Redshift ML 的成本 SageMaker	592
搭配 Amazon Bedrock 使用 Amazon Redshift ML 的成本	592
入門：Amazon Redshift ML	594
管理設定	594
使用模型可解釋性與 Amazon Redshift ML	600
Amazon Redshift ML 概率指標	600
Amazon Redshift ML 的教學課程	603
教學課程：建置客戶流失模型	604
教學課程：建置遠端推論模型	612
教學課程：建置 K 平均值叢集模型	617
教學課程：建置多類別分類模型	626
教學課程：建置XGBoost模型	636
教學課程：建置迴歸模型	641
教學課程：使用線性學習程式建置迴歸模型	654
教學課程：使用線性學習程式建置多類別分類模型	662
Amazon Redshift ML 與 Amazon Bedrock 整合	683
建立IAM角色	684
建立外部模型	684
使用外部模型	685
提示詞工程	687
查詢效能調校	689
查詢處理	689
查詢計劃和執行工作流程	689
建立和解譯查詢計劃	691
檢閱查詢計劃步驟	698
影響查詢效能的因素	701

查詢分析和改善	702
查詢分析工作流程	702
檢閱查詢提醒	703
分析查詢計劃	705
分析查詢摘要	706
查詢改善	711
診斷查詢以進行查詢調校	715
查詢疑難排解	719
連線失敗	720
查詢懸置	720
查詢耗費太長時間	721
載入失敗	722
載入耗費太長時間	723
載入資料不正確	723
設定JDBC擷取大小參數	723
工作負載管理	725
切換WLM模式	726
修改WLM組態	727
從手動遷移WLM至自動 WLM	727
自動 WLM	728
優先順序	729
並行擴展模式	729
使用者群組	729
查詢群組	730
萬用字元	730
查詢監控規則	730
檢查是否有自動 WLM	730
查詢優先順序	731
手動 WLM	735
並行擴展模式	737
並行層級	737
使用者群組	738
查詢群組	739
萬用字元	739
WLM 要使用的記憶體百分比	739
WLM 逾時	739

查詢監控規則	740
WLM 查詢佇列跳轉	740
教學課程：設定手動WLM佇列	743
並行擴展	760
並行擴展功能	761
並行擴展的限制	761
並行擴展的區域	762
並行擴展候選項目	763
設定並行擴展佇列	732
監控並行擴展	764
並行擴展系統檢視	765
短期查詢加速	766
最長SQA執行時間	766
監控 SQA	767
WLM 佇列指派規則	767
佇列指派範例	769
將查詢指派給佇列	772
根據使用者角色將查詢指派給佇列	772
根據使用者群組將查詢指派給佇列	773
將查詢指派給查詢群組	773
將查詢指派給超級使用者佇列	774
動態和靜態屬性	774
WLM 動態記憶體配置	776
動態WLM範例	776
查詢監控規則	778
定義查詢監控規則	779
已佈建 Amazon Redshift 的查詢監控指標	780
Amazon Redshift Serverless 的查詢監控指標	783
查詢監控規則範本	785
查詢監控規則的系統資料表和檢視	786
WLM 系統資料表和檢視	786
WLM 服務類別 IDs	788
資料庫安全	789
Amazon Redshift 安全性概觀	790
預設的資料庫使用者許可	791
超級使用者	791

使用者	792
建立、更改和刪除使用者	792
群組	793
建立、更改和刪除群組	793
控制使用者和群組存取的範例	794
結構描述	795
搜尋路徑	796
建立、更改和刪除結構描述	796
許可	797
角色類型存取控制	797
角色階層	797
角色指派	798
Amazon Redshift 系統定義角色	798
系統許可	801
資料庫物件許可	810
ALTER DEFAULT PRIVILEGES 適用於 RBAC	810
RBAC 中角色使用的考量	810
管理角色	811
教學課程：使用 建立角色和查詢 RBAC	811
資料列層級安全性	830
在SQL陳述式中使用RLS政策	830
每個使用者結合多個政策	830
RLS 政策所有權和管理	832
政策相依物件和原則	834
考量與限制	835
最佳實務	838
End-to-end 範例	840
中繼資料安全性	844
動態資料遮罩	845
SQL DDM政策的 命令	846
DDM 政策階層	847
DDM 搭配SUPER類型路徑使用	848
條件式動態資料遮罩	853
DDM 系統檢視	854
考量事項	856
End-to-end 範例	860

限定範圍權限	863
考量事項	864
SQL 參考	865
Amazon Redshift SQL	865
SQL 領導節點上支援的 函數	865
Amazon Redshift 和 PostgreSQL	868
使用 SQL	875
SQL 參考慣例	876
基本元素	876
表達式	927
條件	932
SQL 命令	959
ABORT	963
ALTER DATABASE	964
ALTER DATASHARE	968
ALTER DEFAULT PRIVILEGES	971
ALTER EXTERNAL VIEW (預覽)	975
ALTER FUNCTION	977
ALTER GROUP	978
ALTER IDENTITY PROVIDER	979
ALTER MASKING POLICY	981
ALTER MATERIALIZED VIEW	982
ALTER RLS POLICY	983
ALTER ROLE	984
ALTER PROCEDURE	986
ALTER SCHEMA	987
ALTER SYSTEM	989
ALTER TABLE	990
ALTER TABLE APPEND	1013
ALTER USER	1018
ANALYZE	1024
ANALYZE COMPRESSION	1026
ATTACH MASKING POLICY	1029
ATTACH RLS POLICY	1030
BEGIN	1031
CALL	1033

CANCEL	1036
CLOSE	1039
COMMENT	1039
COMMIT	1042
COPY	1043
CREATE DATABASE	1136
CREATE DATASHARE	1152
CREATE EXTERNAL FUNCTION	1153
CREATE EXTERNAL MODEL	1163
CREATE EXTERNAL SCHEMA	1167
CREATE EXTERNAL TABLE	1176
CREATE EXTERNAL VIEW	1203
CREATE FUNCTION	1205
CREATE GROUP	1210
CREATE IDENTITY PROVIDER	1211
CREATE LIBRARY	1213
CREATE MASKING POLICY	1216
CREATE MATERIALIZED VIEW	1217
CREATE MODEL	1222
CREATE PROCEDURE	1248
CREATE RLS POLICY	1253
CREATE ROLE	1255
CREATE SCHEMA	1256
CREATE TABLE	1259
CREATE TABLE AS	1280
CREATE USER	1291
CREATE VIEW	1298
DEALLOCATE	1302
DECLARE	1303
DELETE	1307
DESC DATASHARE	1310
DESC IDENTITY PROVIDER	1312
DETACH MASKING POLICY	1313
DETACH RLS POLICY	1314
DROP DATABASE	1314
DROP DATASHARE	1316

DROP EXTERNAL VIEW (預覽)	1317
DROP FUNCTION	1319
DROP GROUP	1321
DROP IDENTITY PROVIDER	1322
DROP LIBRARY	1322
DROP MASKING POLICY	1323
DROP MODEL	1324
DROP MATERIALIZED VIEW	1325
DROP PROCEDURE	1326
DROP RLS POLICY	1327
DROP ROLE	1328
DROP SCHEMA	1329
DROP TABLE	1331
DROP USER	1335
DROP VIEW	1336
END	1339
EXECUTE	1339
EXPLAIN	1340
FETCH	1347
GRANT	1349
INSERT	1372
INSERT (外部資料表)	1379
LOCK	1381
MERGE	1382
PREPARE	1389
REFRESH MATERIALIZED VIEW	1391
RESET	1393
REVOKE	1394
ROLLBACK	1411
SELECT	1413
SELECT INTO	1480
SET	1481
SET SESSION AUTHORIZATION	1486
SET SESSION CHARACTERISTICS	1487
SHOW	1487
SHOW COLUMNS	1488

SHOW EXTERNAL TABLE	1490
SHOW DATABASES	1493
SHOW MODEL	1496
SHOW DATASHARES	1499
SHOW PROCEDURE	1500
SHOW SCHEMAS	1501
SHOW TABLE	1503
SHOW TABLES	1505
SHOW VIEW	1507
START TRANSACTION	1508
TRUNCATE	1508
UNLOAD	1510
UPDATE	1541
VACUUM	1548
SQL 函數參考	1555
僅限領導節點函數	1556
彙總函數	1557
陣列函數	1584
位元彙整函數	1589
條件式運算式	1597
資料類型格式化函數	1611
日期和時間函數	1642
雜湊函數	1710
HyperLogLog 函數	1720
JSON 函數	1725
機器學習函數	1742
數學函數	1745
物件函數	1782
空間函數	1792
字串函數	1926
SUPER 類型資訊函數	2000
VARBYTE 函數	2015
範圍函數	2024
系統管理函數	2086
系統資訊函數	2096
保留字	2128

系統資料表和檢視參考	2132
系統資料表和檢視的類型	2133
系統資料表和檢視中資料的可見性	2133
篩選系統產生的查詢	2134
遷移僅供佈建查詢以SYS監控檢視查詢	2134
從已佈建的叢集遷移至 Amazon Redshift Serverless	2134
停留在已佈建叢集的同時更新查詢	2135
使用SYS監控檢視改善查詢識別符追蹤	2135
範例	2136
系統資料表查詢、程序和工作階段 ID	2143
SVV 中繼資料檢視	2143
SVV_ACTIVE_CURSORS	2145
SVV_ALL_COLUMNS	2146
SVV_ALL_SCHEMAS	2148
SVV_ALL_TABLES	2150
SVV_ALTER_TABLE_RECOMMENDATIONS	2151
SVV_ATTACHED_MASKING_POLICY	2152
SVV_COLUMNS	2155
SVV_COLUMN_PRIVILEGES	2157
SVV_DATABASE_PRIVILEGES	2158
SVV_DATASHARE_PRIVILEGES	2160
SVV_DATASHARES	2161
SVV_DATASHARE_CONSUMERS	2163
SVV_DATASHARE_OBJECTS	2164
SVV_DEFAULT_PRIVILEGES	2166
SVV_DISKUSAGE	2168
SVV_EXTERNAL_COLUMNS	2171
SVV_EXTERNAL_DATABASES	2171
SVV_EXTERNAL_PARTITIONS	2172
SVV_EXTERNAL_SCHEMAS	2173
SVV_EXTERNAL_TABLES	2174
SVV_FUNCTION_PRIVILEGES	2176
SVV_GEOGRAPHY_COLUMNS	2177
SVV_GEOMETRY_COLUMNS	2179
SVV_IAM_PRIVILEGES	2180
SVV_IDENTITY_PROVIDERS	2181

SVV_INTEGRATION	2183
SVV_INTEGRATION_TABLE_STATE	2184
SVV_INTERLEAVED_COLUMNS	2186
SVV_LANGUAGE_PRIVILEGES	2187
SVV_MASKING_POLICY	2188
SVV_ML_MODEL_INFO	2188
SVV_ML_MODEL_PRIVILEGES	2189
SVV_MV_DEPENDENCY	2191
SVV_MV_INFO	2192
SVV_QUERY_INFLIGHT	2194
SVV_QUERY_STATE	2195
SVV_REDSHIFT_COLUMNS	2198
SVV_REDSHIFT_DATABASES	2200
SVV_REDSHIFT_FUNCTIONS	2201
SVV_REDSHIFT_SCHEMA_QUOTA	2203
SVV_REDSHIFT_SCHEMAS	2204
SVV_REDSHIFT_TABLES	2205
SVV_RELATION_PRIVILEGES	2206
SVV_RLS_APPLIED_POLICY	2208
SVV_RLS_ATTACHED_POLICY	2209
SVV_RLS_POLICY	2210
SVV_RLS_RELATION	2212
SVV_ROLE_GRANTS	2213
SVV_ROLES	2214
SVV_SCHEMA_PRIVILEGES	2214
SVV_SCHEMA_QUOTA_STATE	2216
SVV_SYSTEM_PRIVILEGES	2217
SVV_TABLE_INFO	2218
SVV_TABLES	2222
SVV_TRANSACTIONS	2223
SVV_USER_GRANTS	2225
SVV_USER_INFO	2226
SVV_VACUUM_PROGRESS	2227
SVV_VACUUM_SUMMARY	2229
SYS 監控檢視	2231
SYS_ANALYZE_COMPRESSION_HISTORY	2232

SYS_ANALYZE_HISTORY	2235
SYS_APPLIED_MASKING_POLICY_LOG	2236
SYS_AUTO_TABLE_OPTIMIZATION	2238
SYS_CONNECTION_LOG	2240
SYS_COPY_JOB (預覽)	2244
SYS_COPY_REPLACEMENTS	2245
SYS_DATASHARE_CHANGE_LOG	2246
SYS_DATASHARE_CROSS_REGION_USAGE	2249
SYS_DATASHARE_USAGE_CONSUMER	2250
SYS_DATASHARE_USAGE_PRODUCER	2251
SYS_EXTERNAL_QUERY_DETAIL	2253
SYS_EXTERNAL_QUERY_ERROR	2256
SYS_INTEGRATION_ACTIVITY	2258
SYS_INTEGRATION_TABLE_STATE_CHANGE	2260
SYS_LOAD_DETAIL	2261
SYS_LOAD_ERROR_DETAIL	2264
SYS_LOAD_HISTORY	2266
SYS_MV_REFRESH_HISTORY	2270
SYS_MV_STATE	2272
SYS_PROCEDURE_CALL	2275
SYS_PROCEDURE_MESSAGES	2276
SYS_QUERY_DETAIL	2278
SYS_QUERY_HISTORY	2283
SYS_QUERY_TEXT	2293
SYS_RESTORE_LOG	2295
SYS_RESTORE_STATE	2297
SYS_SCHEMA_QUOTA_VIOLATIONS	2299
SYS_SERVERLESS_USAGE	2300
SYS_SESSION_HISTORY	2302
SYS_SPATIAL_SIMPLIFY	2304
SYS_STREAM_SCAN_ERRORS	2305
SYS_STREAM_SCAN_STATES	2307
SYS_TRANSACTION_HISTORY	2309
SYS_UDF_LOG	2311
SYS_UNLOAD_DETAIL	2313
SYS_UNLOAD_HISTORY	2314

SYS_USERLOG	2316
SYS_VACUUM_HISTORY	2318
用於移轉至 SYS 監視檢視的系統檢視對映	2321
SYS_QUERY_HISTORY	2322
SYS_QUERY_DETAIL	2323
SYS_RESTORE_LOG	2324
SYS_RESTORE_STATE	2324
SYS_TRANSACTION_HISTORY	2324
SYS_QUERY_TEXT	2325
SYS_CONNECTION_LOG	2325
SYS_SESSION_HISTORY	2325
SYS_LOAD_DETAIL	2325
SYS_LOAD_HISTORY	2325
SYS_LOAD_ERROR_DETAIL	2325
SYS_UNLOAD_HISTORY	2326
SYS_UNLOAD_DETAIL	2326
SYS_COPY_REPLACEMENTS	2326
SYS_DATASHARE_USAGE_CONSUMER	2326
SYS_DATASHARE_USAGE_PRODUCER	2326
SYS_DATASHARE_CROSS_REGION_USAGE	2326
SYS_DATASHARE_CHANGE_LOG	2327
SYS_EXTERNAL_QUERY_DETAIL	2327
SYS_EXTERNAL_QUERY_ERROR	2327
SYS_VACUUM_HISTORY	2327
SYS_ANALYZE_HISTORY	2327
SYS_ANALYZE_COMPRESSION_HISTORY	2328
SYS_MV_REFRESH_HISTORY	2328
SYS_MV_STATE	2328
SYS_PROCEDURE_CALL	2328
SYS_PROCEDURE_MESSAGES	2328
SYS_UDF_LOG	2328
SYS_USERLOG	2329
SYS_SCHEMA_QUOTA_VIOLATIONS	2329
SYS_SPATIAL_SIMPLIFY	2329
系統監控 (僅限已佈建)	2329
STL 記錄的檢視	2329

STV用於快照資料的表格	2453
SVCS主要和並行擴展叢集的檢視	2502
主要叢集的 SVL 檢視	2528
系統目錄資料表	2593
PG_ATTRIBUTE_INFO	2594
PG_CLASS_INFO	2594
PG_DATABASE_INFO	2596
PG_DEFAULT_ACL	2597
PG_EXTERNAL_SCHEMA	2599
PG_LIBRARY	2600
PG_PROC_INFO	2601
PG_STATISTIC_INDICATOR	2601
PG_TABLE_DEF	2602
PG_USER_INFO	2605
查詢目錄資料表	2606
組態參考	2612
修改伺服器組態	2613
analyze_threshold_percent	2614
值 (粗體為預設值)	2614
描述	2614
範例	2614
cast_super_null_on_error	2615
值 (粗體為預設值)	2615
描述	2615
datashare_break_glass_session_var	2615
值 (粗體為預設值)	2615
描述	2615
範例	2615
datestyle	2616
值 (粗體為預設值)	2616
描述	2615
範例	2616
default_geometry_encoding	2616
值 (粗體為預設值)	2616
描述	2615
describe_field_name_in_uppercase	2617

值 (粗體為預設值)	2617
描述	2615
範例	2616
downcase_delimited_identifier	2617
值 (粗體為預設值)	2617
描述	2615
使用須知	2618
enable_case_sensitive_identifier	2619
值 (粗體為預設值)	2619
描述	2619
範例	2619
使用須知	2620
enable_case_sensitive_super_attribute	2621
值 (粗體為預設值)	2621
描述	2622
範例	2622
使用須知	2623
enable_numeric_rounding	2623
值 (粗體為預設值)	2623
描述	2624
範例	2624
enable_result_cache_for_session	2625
值 (粗體為預設值)	2625
描述	2625
範例	2625
enable_vacuum_boost	2626
值 (粗體為預設值)	2626
描述	2615
error_on_nondeterministic_update	2626
值 (粗體為預設值)	2626
描述	2615
範例	2616
extra_float_digits	2627
值 (粗體為預設值)	2627
描述	2627
範例	2627

interval_forbid_composite_literals	2628
值 (粗體為預設值)	2628
描述	2615
json_serialization_enable	2629
值 (粗體為預設值)	2629
描述	2615
json_serialization_parse_nested_strings	2629
值 (粗體為預設值)	2629
描述	2615
max_concurrency_scaling_clusters	2629
值 (粗體為預設值)	2629
描述	2630
max_cursor_result_set_size	2630
值 (粗體為預設值)	2630
描述	2630
mv_enable_aqmv_for_session	2630
值 (粗體為預設值)	2630
描述	2630
navigate_super_null_on_error	2630
值 (粗體為預設值)	2630
描述	2615
parse_super_null_on_error	2631
值 (粗體為預設值)	2631
描述	2615
pg_federation_repeatable_read	2631
值 (粗體為預設值)	2631
描述	2615
範例	2631
query_group	2632
值 (粗體為預設值)	2632
描述	2632
search_path	2633
值 (粗體為預設值)	2633
描述	2633
範例	2633
spectrum_enable_pseudo_columns	2634

值 (粗體為預設值)	2634
描述	2635
範例	2635
enable_spectrum_oid	2635
值 (粗體為預設值)	2635
描述	2635
範例	2635
spectrum_query_maxerror	2635
值 (粗體為預設值)	2635
描述	2635
範例	2636
statement_timeout	2636
值 (粗體為預設值)	2636
描述	2636
範例	2636
stored_proc_log_min_messages	2637
值 (粗體為預設值)	2637
描述	2615
timezone	2637
值 (粗體為預設值)	2637
語法	2637
描述	2638
時區格式	2638
範例	2640
use_fips_ssl	2641
值 (粗體為預設值)	2641
描述	2615
wlm_query_slot_count	2641
值 (粗體為預設值)	2641
描述	2641
範例	2642
文件歷史紀錄	2643
舊版更新	2651
.....	mmdclxxiii

Amazon Redshift 簡介

歡迎使用《Amazon Redshift 資料庫開發人員指南》。本指南著重於協助您了解如何使用 Amazon Redshift 來建立和管理資料倉儲。如果您以設計師、軟體開發人員或管理員的身分使用資料庫，本指南會為您提供設計、建置、查詢和維護資料倉儲所需的資訊。

Amazon Redshift 是一種在雲端中完全受管的 PB 級資料倉儲服務。Amazon Redshift Serverless 可讓您無需佈建資料倉儲的一般組態，即可存取和分析資料。系統會自動佈建資源，並有智慧地擴展資料倉儲容量，即使是最嚴苛且無法預測的工作負載，也能為其提供快速的效能。資料倉儲閒置時不會產生費用，因此只需按實際用量支付費用。無論資料集的大小為何，您都可以立即在 Amazon Redshift 查詢編輯器 v2 或偏好的商業智慧 (BI) 工具中載入資料並開始查詢。在零管理環境中 easy-to-use 享受最佳價格效能和熟悉 SQL 的功能。

主題

- [使用 Amazon Redshift 的先決條件](#)
- [Amazon Redshift 架構](#)
- [範本資料庫](#)

使用 Amazon Redshift 的先決條件

本主題說明使用 Amazon Redshift 所需的先決條件。

使用本指南之前，您應該閱讀 [Redshift Serverless 資料倉儲入門](#)，其中會說明如何完成下列任務。

- 使用 Amazon Redshift Serverless 建立資料倉儲。
- 使用 Amazon Redshift 查詢編輯器第 2 版載入範例資料
- 從 Amazon S3 載入資料。

您也應該知道如何使用 SQL 用戶端，並對 SQL 語言有基本的了解。

Amazon Redshift 架構

本主題可協助您了解組成 Amazon Redshift 的元件。

Amazon Redshift 資料倉儲是企業級的關聯式資料庫查詢與管理系統。

Amazon Redshift 透過多種類型的應用程式來支援用戶端連線，包括商業智慧 (BI)、報告、資料與分析工具。

執行分析查詢時，會以多階段操作的方式，擷取、比較和評估大量的資料，來產生最終的結果。

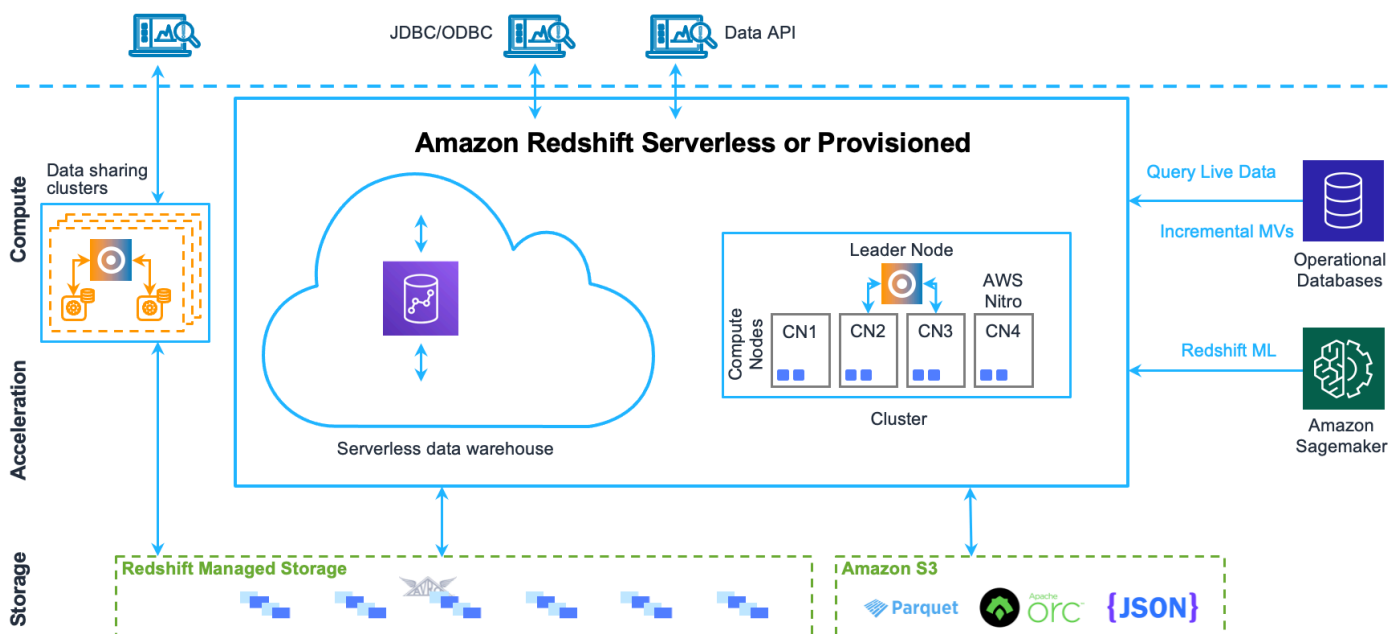
藉由結合大規模的平行處理作業、單欄式資料儲存體，和超高效率的針對式資料壓縮編碼機制，Amazon Redshift 可實現高效率的儲存與最佳化的查詢效能。本節提供 Amazon Redshift 系統架構的介紹。

主題

- [資料倉儲系統架構](#)
- [Amazon Redshift 效能](#)
- [單欄式儲存](#)
- [工作負載管理](#)
- [搭配其他服務使用 Amazon Redshift](#)

資料倉儲系統架構

本節說明組成 Amazon Redshift 資料倉儲架構的元件，如下圖所示。



用戶端應用程式

Amazon Redshift 與各種資料載入和 ETL (擷取、轉換和載入) 工具和商業智慧 (BI) 報告、資料探勘和分析工具整合。Amazon Redshift 是以開放標準 Postgre 為基礎 SQL , 因此大多數現有的 SQL 用戶端應用程式只能使用最少的變更。如需 Amazon Redshift SQL 和 Postgre 之間重要差異的資訊 SQL , 請參閱 [Amazon Redshift 和 PostgreSQL](#)。

叢集

Amazon Redshift 資料倉儲的核心基礎設施元件是叢集。

叢集是由一或多個運算節點所組成。如果為叢集佈建了兩個或多個運算節點，則會有另外的領導節點負責統籌運算節點和處理外部通訊。您的用戶端應用程式只會直接和領導節點互動，外部應用程式不會知道運算節點的存在。

領導節點

領導節點會管理與用戶端程式的通訊，以及和運算節點的所有通訊。此節點會剖析和制定執行計畫，來執行資料庫的操作，尤其是取得複雜查詢結果所需的一系列步驟。領導節點會根據執行計畫，來編譯程式碼、將編譯過的程式碼分發到運算節點，並指派資料的各部分給每個運算節點。

領導者節點只有在查詢參考儲存在運算節點上的資料表時，才會將 SQL 陳述式分發給運算節點。所有其他查詢僅在領導節點上執行。Amazon Redshift 旨在僅在領導節點上實作某些 SQL 函數。如果查詢使用這些函式的任一個，而且參考位於運算節點上的資料表，則將會傳回錯誤。如需詳細資訊，請參閱 [SQL 領導節點上支援的 函數](#)。

運算節點

領導節點會針對執行計畫的個別元素，來編譯程式碼，並將程式碼指派給個別的運算節點。運算節點會執行編譯過的程式碼，並將中間的結果傳回領導節點，以進行最終的彙總。

每個運算節點都有自己的專用 CPU 記憶體，由節點類型決定。隨著工作負載的增長，您可以藉由增加節點的數量、將節點的類型升級，或是同時實行這兩種做法，來增加叢集的運算容量。

Amazon Redshift 會針對您的運算需求，提供數種節點類型。如需每個節點類型的詳細資訊，請參閱《Amazon Redshift 管理指南》中的 < [Amazon Redshift 叢集](#) >。

Redshift 受管儲存

資料倉儲資料會存放在單獨的儲存層 Redshift Managed Storage () 中 RMS。RMS 可讓您使用 Amazon S3 儲存體將儲存體擴展到 PB。RMS 可讓您獨立擴展和支付運算和儲存費用，以便您可以僅根據運算需求調整叢集的大小。它會自動使用高效能 SSD 型本機儲存體做為 Tier-1 快取。它還會利用

資料區塊溫度、資料區塊存留時間和工作負載模式等最佳化功能提供高效能，同時在需要時自動將儲存擴展到 Amazon S3，而無需採取任何動作。

節點配量

運算節點可分割為分割。每個分割會分配到節點的一部分記憶體和磁碟空間，並且用以處理指派給節點的部分工作負載。領導節點會負責將資料分發給分割，並將任何查詢或其他資料庫操作的工作負載，分派給這些分割。這些分割接著會平行運作，以完成操作。

每個節點的分割數目，取決於叢集的節點大小。如需每個節點大小有多少配量的相關資訊，請移至《Amazon Redshift 管理指南》中的 < [關於叢集和節點](#) >。

建立資料表時，您可以選擇性地指定一個欄做為分佈索引鍵。當資料載入資料表時，會根據為資料表定義的分佈索引鍵，將資料列分發給節點分割。選擇理想的分佈索引鍵，可讓 Amazon Redshift 利用平行處理作業，以高效率的方式載入資料和執行查詢。如需關於選擇分佈索引鍵的詳細資訊，請參閱[選擇最佳的分佈方式](#)。

內部網路

Amazon Redshift 善用了高頻寬連線、鄰近位置與自訂通訊協定，在領導節點和運算節點之間，提供私密的超高速網路通訊。運算節點會在獨立的隔離網路中執行，用戶端應用程式絕對不會直接存取此網路。

資料庫

叢集包含一個或多個資料庫。使用者資料儲存於運算節點上。您的 SQL 用戶端會與領導者節點通訊，進而協調與運算節點執行的查詢。

Amazon Redshift 是關聯式資料庫管理系統（RDBMS），因此與其他 RDBMS 應用程式相容。雖然它提供與一般相同的 RDBMS 功能，包括線上交易處理（OLTP）函數，例如插入和刪除資料，但 Amazon Redshift 已針對非常大型資料集的高效能分析和報告進行最佳化。

Amazon Redshift 是以 Postgre 為基礎 SQL。Amazon Redshift 和 PostgreSQL 在設計和開發資料倉儲應用程式時，需要考量許多非常重要的差異。如需 Amazon Redshift 與 Postgre 有何 SQL 不同的相關資訊 SQL，請參閱 [Amazon Redshift 和 PostgreSQL](#)。

Amazon Redshift 效能

本主題說明驅動效能的 Amazon Redshift 元件。了解這些元件將協助您調整效能，並使用 Amazon Redshift 疑難排解效能不佳的問題。

Amazon Redshift 運用這些效能功能，實現超快速的查詢執行。

主題

- [大規模平行處理](#)
- [單欄式資料儲存體](#)
- [資料壓縮](#)
- [查詢最佳化工具](#)
- [結果快取](#)
- [經過編譯的程式碼](#)

大規模平行處理

大規模平行處理（MPP）可讓您快速執行在大量資料上執行的最複雜查詢。多重運算節點進行所有查詢處理作業，最後再將結果彙總（每個節點的核心會針對整個資料的一部分，執行相同的編譯後查詢片段）。

Amazon Redshift 會將資料表的列分發給運算節點，以平行處理資料。透過為每個資料表選擇適合的分佈索引鍵，您可以將資料的分發最佳化，進而平衡工作負載，並且將資料在節點之間的移動次數減到最少。如需詳細資訊，請參閱[選擇最佳的分佈方式](#)。

從一般檔案載入資料，可透過將工作負載分散到多個節點，一邊同時從多個檔案讀取，來發揮平行處理的效益。如需如何將資料載入資料表的相關資訊，請參閱[載入資料的 Amazon Redshift 最佳實務](#)。

單欄式資料儲存體

資料庫資料表的單欄式儲存方式，大幅地降低了整體磁碟輸入/輸出需求，成為實現最佳化分析查詢效能的重要因素。如需詳細資訊，請參閱[單欄式儲存](#)。

以適當地方式將欄排序時，查詢處理器就能夠快速地篩選掉龐大的資料區塊子集。如需詳細資訊，請參閱[選擇最佳的排序索引鍵](#)。

資料壓縮

資料壓縮可降低對儲存空間的需求，進而減少磁碟輸入/輸出，提升查詢作業的效能。執行查詢作業時，會將壓縮的資料讀入記憶體，然後在進行查詢時將資料解壓縮。載入記憶體的資料量減少，即可讓 Amazon Redshift 配置更多的記憶體來分析資料。由於單欄式儲存會依順序儲存類似的資料，Amazon Redshift 就能夠採用與單欄式資料類型特別相關的適應性壓縮編碼機制。對資料表欄進行資料壓縮的最

佳方式，就是在載入資料表和資料時，讓 Amazon Redshift 套用最佳化的壓縮編碼機制。如要進一步了解自動資料壓縮，請參閱[利用自動壓縮載入資料表](#)。

查詢最佳化工具

Amazon Redshift 查詢執行引擎整合了查詢最佳化工具，該最佳化工具具有 MPP 意識，並利用資料欄導向的資料儲存。Amazon Redshift 查詢最佳化工具建置了重大的增強與擴充功能，可用來處理複雜的分析查詢作業，這類查詢經常包含多重資料表的合併、子查詢和彙總操作。若要進一步了解如何將查詢作業最佳化，請參閱[查詢效能調校](#)。

結果快取

為了縮短查詢執行期並改善系統效能，Amazon Redshift 會將特定查詢類型的結果快取在領導者節點的記憶體中。當使用者提交查詢時，Amazon Redshift 會針對結果快取，檢查是否建立了查詢結果的有效快取副本。如果在結果快取中找到符合者，Amazon Redshift 會使用快取的結果，不會執行查詢。使用者並不會察覺到結果快取作業的進行。

結果快取預設為開啟。若要關閉目前工作階段的結果快取作業，請將 [enable_result_cache_for_session](#) 參數設定為 off。

當下列所有情況都符合時，Amazon Redshift 會使用新查詢的快取結果：

- 提交查詢的使用者，對查詢中所使用的物件具有存取許可。
- 查詢中的資料表或檢視尚未經過修改。
- 查詢不會使用每次執行時都必須評估的函數，例如 GETDATE。
- 查詢並未參考 Amazon Redshift Spectrum 外部資料表。
- 可能影響查詢結果的組態參數不變。
- 查詢在語法上符合快取的查詢。

為了最大限度地提高快取效率並有效利用資源，Amazon Redshift 不會快取某些大型查詢結果集。Amazon Redshift 會根據許多因素判斷是否要快取查詢結果。這些因素包括快取中的項目數量，以及 Amazon Redshift 叢集的執行個體類型。

若要判斷查詢作業是否使用了結果快取，請查詢 [SVL_QLOG](#) 系統畫面。如果查詢使用了結果快取，source_query 欄會傳回來源查詢的查詢 ID。如果未使用結果快取，則 source_query 資料欄值為 NULL。

下列範例顯示，由 userid 104 和 userid 102 所提交的查詢，使用了 userid 100 所執行查詢的結果快取。

```
select userid, query, elapsed, source_query from svl_qlog
where userid > 1
order by query desc;
```

userid	query	elapsed	source_query
104	629035	27	628919
104	629034	60	628900
104	629033	23	628891
102	629017	1229393	
102	628942	28	628919
102	628941	57	628900
102	628940	26	628891
100	628919	84295686	
100	628900	87015637	
100	628891	58808694	

經過編譯的程式碼

領導節點會將經過完整最佳化編譯的程式碼，分發給叢集的所有節點。編譯查詢可省去與直譯器相關的資源耗用，進而提升執行期速度，尤其是複雜的查詢作業。編譯後的程式碼會經過快取，並在同一個叢集上的工作階段之間共用。因此，未來執行相同查詢的速度會更快，即使使用不同的參數也應是如此。

查詢執行引擎會為 JDBC 和 ODBC 連線通訊協定編譯不同的程式碼，因此使用不同通訊協定的兩個用戶端會分別產生編譯程式碼的第一次成本。使用相同通訊協定的用戶端，可從共用快取的程式碼而受惠。

單欄式儲存

本節說明資料欄儲存，這是 Amazon Redshift 用來有效儲存表格式資料的方法。

資料庫資料表的單欄式儲存方式是實現最佳化分析查詢效能的重要因素，因為其大幅地降低了整體磁碟輸入/輸出需求。這麼做可降低您需要從磁碟載入的資料量。

下列的一系列圖解，說明單欄式資料儲存如何實現效率，進而提升將資料擷取到記憶體時的效率。

第一張圖顯示資料庫資料表的記錄，通常是如何以列的形式儲存到磁碟區塊中。

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL



在典型的關聯式資料庫資料表中，每列會包含單一筆記錄的欄位值。在列式資料庫儲存法中，資料區塊會依順序儲存每個連續資料欄的值，這構成了一個整列。如果區塊的容量小於記錄的大小，則整筆記錄所佔的儲存空間可能會超過一個區塊。如果區塊的容量大於記錄的大小，則整筆記錄所佔的儲存空間可能會少於一個區塊，造成磁碟空間使用上的無效率。在線上交易處理（OLTP）應用程式中，大多數交易涉及經常讀取和寫入整個記錄的所有值，通常是一次一個記錄或少量記錄。因此，資料列儲存最適合OLTP資料庫。

下一張圖解顯示，使用單欄式儲存法時，每欄的值如何按順序儲存在磁碟區塊中。

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL



使用單欄式儲存法時，每個資料區塊會儲存多個資料列單欄的值。當記錄進到系統時，Amazon Redshift 會在不被察覺的狀況下，將各欄的資料轉換為單欄式儲存模式。

在這個簡化的範例中，相較於列式儲存法，使用單欄式儲存法時，每個資料區塊可儲存多達三倍的記錄欄位值。這代表相較於列式儲存法，針對相同數目的記錄讀取相同數量的欄位值時，只需要進行三分之一的輸入/輸出操作。實際上，所使用的資料表如果包含龐大數量的欄數和列數時，單欄式儲存法的儲存效率甚至會更高。

另一個額外的優點是，由於每個區塊都儲存相同類型的資料，因此區塊資料可以採用專為欄資料類型而選擇的壓縮方法，以進一步減少磁碟空間和 I/O。如需根據資料類型決定壓縮編碼方法的詳細資訊，請參閱 [壓縮編碼](#)。

在磁碟上儲存資料所省下的空間，也可沿用到擷取資料然後將資料儲存到記憶體。由於許多的資料庫操作一次只需存取或操作一個或少量的欄，因此藉由只擷取查詢實際所需的資料欄區塊，可以節省記憶體的空間。當OLTP交易通常涉及少量記錄的大部分或所有資料欄時，資料倉儲查詢通常只會讀取大量資料欄的幾欄。這代表針對相同數目的資料列讀取相同數量的欄位值時，只需要進行部分的輸入/輸出操作。其使用處理資料列區塊時所需的一小部分記憶體。實際上，所使用的資料表如果包含龐大數量的欄數和列數時，欄式儲存法的儲存效率會按比例提高。例如，假設資料表包含 100 個資料欄。使用 5 個資料欄的查詢作業，將只需要讀取資料表中所包含資料的 5%。大型資料庫可能會有數十億或甚至數兆的記錄，都能重複實現這種節省效率。相較之下，列式資料庫也會讀取包含其他 95 個非必要資料欄的資料區塊。

典型的資料庫區塊大小範圍介於 2 KB 到 32 KB 之間。Amazon Redshift 使用效率更高的 1 MB 區塊大小，這在執行任何的資料庫載入作業，或查詢作業執行所包含的其他操作時，可進一步減少所需的 I/O 請求數量。

工作負載管理

本節說明工作負載管理 (WLM)，協助您了解 Amazon Redshift 如何準備和執行查詢。

Amazon Redshift 工作負載管理 (WLM) 可在工作負載中啟用彈性的管理優先順序，以便快速執行的短查詢不會卡在長時間執行查詢後的佇列中。Amazon Redshift 會根據服務類別，在執行期建立查詢佇列，定義各種佇列類型的組態參數，包括內部系統佇列和使用者可存取的佇列。從使用者的角度來看，使用者可存取的服務類別和佇列在功能上是相同的。為了保持一致，本文件使用佇列一詞，來同時表示使用者可存取的服務類別，以及執行時間佇列。

Redshift 提供自動工作負載管理，稱為自動 WLM，經過調校以處理不同的工作負載，是建議的預設值。透過自動 WLM，Redshift 會在查詢到達時判斷資源使用率，並動態判斷要在主叢集、貨幣擴展叢集上執行，還是將每個叢集傳送至佇列。(將查詢排入佇列時，自動會WLM優先處理較短時間的查詢。) Automatic 可將總輸送量WLM最大化，並可讓您維持高效率的資料庫資源。您執行工作負載時不必擔心其大小或排程方式。已佈建叢集的預設值WLM為自動。如需詳細資訊，請參閱[實作自動 WLM](#)。

Note

Amazon Redshift Serverless 工作群組一律使用自動 WLM。

在執行大量查詢或資源密集型查詢時，工作負載管理可以在本機資源上佇列時擴展到其他運算資源。並行擴展與自動WLM支援幾乎無限制並行使用者和查詢的一致效能。

如果您需要精細的手動WLM最佳化，Redshift 佈建叢集可提供手動。在此，客戶會管理資源配置、查詢並行和佇列。查詢執行時，會根據使用者的使用者群組，或透過比對佇列組態中列出的查詢群組，將查詢WLM指派給佇列。這是使用使用者設定的查詢群組標籤來設定。如需詳細資訊，請參閱[實作手動 WLM](#)。

雖然隨著時間的推移，手冊WLM可以進行微調以符合您的工作負載模式，但在大多數情況下，我們並不建議使用它，因為其靜態性質可能會讓您更難適應一天內或長時間內不斷變化的工作負載。它需要更多監控和持續調校。此外，WLM在許多情況下，手動使用運算資源的效率不如自動 WLM，例如，如果手動設定佇列來限制分配給他們的記憶體。

衡量工作負載管理組態成功的重要指標是系統輸送量，換句話說，已成功完成多少查詢。系統輸送量以每秒查詢為單位來測量。如需系統指標的詳細資訊，請參閱[監控 Amazon Redshift 叢集效能](#)。

管理WLM組態的最簡單方法是使用 Amazon Redshift Management 主控台。您也可以使用 [Amazon Redshift 命令列介面](#) (CLI) 或 [Amazon Redshift API](#)。如需實作和使用工作負載管理的詳細資訊，請參閱[實作工作負載管理](#)。

搭配其他服務使用 Amazon Redshift

本節說明如何使用其他服務作為 Amazon Redshift 資料的來源和目的地。

Amazon Redshift 與其他 AWS 服務整合，可讓您使用資料安全功能，快速可靠地移動、轉換和載入資料。

S3

Amazon Simple Storage Service (Amazon S3) 是將資料儲存在雲端中的 Web 服務。Amazon Redshift 利用平行處理功能，從儲存在 Amazon S3 儲存貯體中的多個資料檔案讀取和載入資料。如需詳細資訊，請參閱[從 Amazon S3 載入資料](#)。

您也可以利用平行處理模式，從 Amazon Redshift 資料倉儲將資料匯出至 Amazon S3 上的多個資料檔案。如需詳細資訊，請參閱[在 Amazon Redshift 卸載數據](#)。

DynamoDB

Amazon DynamoDB 是完全受管的無SQL資料庫服務。您可以使用 COPY命令，從單一 Amazon DynamoDB 資料表載入具有資料的 Amazon Redshift 資料表。如需詳細資訊，請參閱[從 Amazon DynamoDB 資料表載入資料](#)。

SSH

您可以在 Amazon Redshift 中使用 COPY命令從一或多個遠端主機載入資料，例如 Amazon EMR 叢集、Amazon EC2執行個體或其他電腦。COPY 使用 連線至遠端主機，SSH並在遠端主機上執行

命令以產生資料。Amazon Redshift 支援多個同時連線。COPY 命令會平行讀取和載入來自多個主機來源的輸出。如需詳細資訊，請參閱[從遠端主機載入資料](#)。

AWS Data Pipeline

您可以使用 AWS Data Pipeline 來自動化進出 Amazon Redshift 的資料移動和轉換。透過使用的內建排程功能 AWS Data Pipeline，您可以排程和執行重複性任務，而不必撰寫自己的複雜資料傳輸或轉換邏輯。例如，您可以設定重複性的任務，自動將資料從 Amazon DynamoDB 複製到 Amazon Redshift。如需教學課程，引導您逐步完成建立管道的程序，以定期將資料從 Amazon S3 移至 Amazon Redshift，請參閱 AWS Data Pipeline 開發人員指南中的[使用 將資料複製到 Amazon Redshift AWS Data Pipeline](#)。

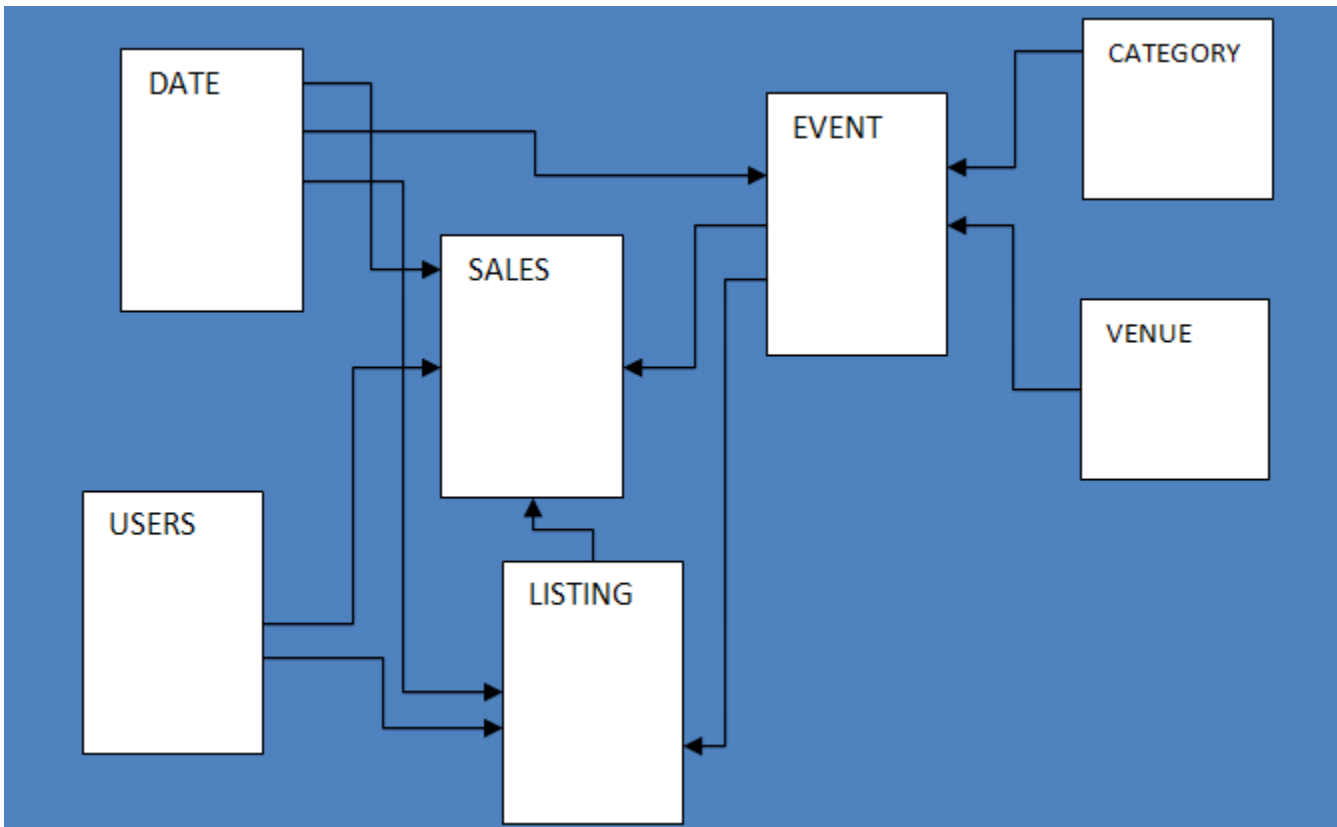
AWS DMS

您可以使用 將資料遷移至 Amazon Redshift AWS Database Migration Service。AWS DMS 可以將資料遷移至最常使用的商業和開放原始碼資料庫，例如 Oracle、PostgreSQL、Microsoft SQL Server、Amazon Redshift、Aurora 資料庫叢集、DynamoDB、Amazon S3、MariaDB 和 My SQL。如需詳細資訊，請參閱[使用 Amazon Redshift 資料庫做為 AWS Database Migration Service 的目標](#)。

範本資料庫

本節說明 TICKIT，這是 Amazon Redshift 文件範例使用的範例資料庫。

此小型資料庫包含七個資料表：兩個事實資料表和五個維度。您可以依照 [Amazon Redshift 入門指南中的步驟 4：將資料從 Amazon S3 載入](#) Amazon Redshift，以載入 TICKIT 資料集。



此資料庫應用程式範例可協助分析人員追蹤虛構TICKIT網站的銷售活動，使用者可在線上購買和銷售體育賽事、節目和音樂會的門票。分析可特別辨識門票隨著時間的動向、賣方的成功比率以及熱賣活動、會場和季節。分析可使用此資訊來為經常造訪網站的買方和賣方提供誘因，以吸引新使用者和促使廣告和促銷。

例如，下列查詢會根據在 2008 所售的門票數，找到在聖地牙哥中前五大賣方：

```

select sellerid, username, (firstname || ' ' || lastname) as name,
city, sum(qtysold)
from sales, date, users
where sales.sellerid = users.userid
and sales.dateid = date.dateid
and year = 2008
and city = 'San Diego'
group by sellerid, username, name, city
order by 5 desc
limit 5;

```

sellerid	username	name	city	sum
49977	JJK84WTE	Julie Hanson	San Diego	22

```

19750 | AAS23BDR | Charity Zimmerman | San Diego | 21
29069 | SVL81MEQ | Axel Grant          | San Diego | 17
43632 | VAG08HKW | Griffin Dodson     | San Diego | 16
36712 | RXT40MKU | Hiram Turner       | San Diego | 14
(5 rows)

```

本指南中用於範例的資料庫包含一個小型資料集；兩個事實資料表各包含少於 200,000 列，且維度範圍從CATEGORY資料表中的 11 列到USERS資料表中的約 50,000 列。

此指南中的資料庫範例特別示範 Amazon Redshift 資料表設計的主要功能：

- 資料分佈
- 資料排序
- 直欄式壓縮

如需TICKIT資料庫中資料表結構描述的相關資訊，請選擇下列索引標籤：

CATEGORY table

欄名稱	資料類型	描述
CATID	SMALLINT	主要索引鍵，每個列的唯一 ID 值。每個列代表購買或賣出之特定類型的活動門票。
CATGROUP	VARCHAR (10)	活動群組的描述性名稱，例如 Shows 和 Sports 。
CATNAME	VARCHAR (10)	群組中活動類型的簡短描述性名稱，例如 Opera 和 Musicals 。
CATDESC	VARCHAR (50)	活動類型的較長描述性名稱，例如 Musical theatre 。

DATE table

欄名稱	資料類型	描述
DATEID	SMALLINT	主要索引鍵，每個列的唯一 ID 值。每個列代表日曆年中的一天。
CALDATE	DATE	日曆日期，例如 2008-06-24 。
DAY	CHAR (3)	一週中的一天 (短格式)，例如 SA 。
WEEK	SMALLINT	週編號，例如 26 。
MONTH	CHAR (5)	月名稱 (短格式)，例如 JUN 。
QTR	CHAR (5)	季編號 (1 到 4)。
YEAR	SMALLINT	四位數年份 (2008)。
HOLIDAY	BOOLEAN	標記表示該日是否為國定假日 (美國)。

EVENT table

欄名稱	資料類型	描述
EVENTID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每個列代表個別的项目，這些活動會在特定時間在特定的會場舉辦。
VENUEID	SMALLINT	VENUE 資料表的外鍵參考。
CATID	SMALLINT	CATEGORY 資料表的外鍵參考。
DATEID	SMALLINT	DATE 資料表的外鍵參考。
EVENTNAME	VARCHAR (200)	活動名稱，例如 Hamlet 或 La Traviata 。
STARTTIME	TIMESTAMP	活動的完整日期和開始時間，例如 2008-10-10 19:30:00 。

VENUE table

欄名稱	資料類型	描述
VENUEID	SMALLINT	主要索引鍵，每個列的唯一 ID 值。每個列代表活動舉辦的特定會場。
VENUENAME	VARCHAR (100)	場所的確切名稱，例如 Cleveland Browns Stadium 。
VENUECITY	VARCHAR (30)	城市名稱，例如 Cleveland 。
VENUESTATE	CHAR (2)	兩個字母的州或省縮寫 (美國和加拿大)，例如 OH 。
VENUESEATS	INTEGER	會場可用座位數上限 (如果知道)，例如 73200 。為達示範目的，此欄位包含一些空值和零。

USERS table

欄名稱	資料類型	描述
USERID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每列代表註冊的使用者 (買方或賣方或兩者)，這些使用者已列出或購買至少一個活動的門票。
USERNAME	CHAR (8)	8 字元英數使用者名稱，例如 PGL08LJI 。
FIRSTNAME	VARCHAR (30)	使用者的名字，例如 Victor 。
LASTNAME	VARCHAR (30)	使用者的姓氏，例如 Hernandez 。
CITY	VARCHAR (30)	使用者的住家所在城市，例如 Naperville 。
STATE	CHAR (2)	使用者的住家所在州，例如 GA 。
EMAIL	VARCHAR (100)	使用者的電子郵件地址，此欄位包含隨機 Latin 值，例如 turpis@accumsanlaoreet.org 。

欄名稱	資料類型	描述
PHONE	CHAR (14)	使用者的 14 個字元的電話號碼，例如 (818) 765-4255 。
LIKESPORTS, ...	BOOLEAN	一系列 10 個不同欄位，您可透過 true 和 false 值識別使用者的喜好。

LISTING table

欄名稱	資料類型	描述
LISTID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每個列代表一組特定活動之門票的清單。
SELLERID	INTEGER	USERS 資料表的外鍵參考，識別銷售票證的使用者。
EVENTID	INTEGER	EVENT 資料表的外鍵參考。
DATEID	SMALLINT	DATE 資料表的外鍵參考。
NUMTICKETS	SMALLINT	可供銷售的門票數，例如 2 或 20 。
PRICEPERTICKET	DECIMAL (8 , 2)	個別門票的定價，例如 27.00 或 206.00 。
TOTALPRICE	DECIMAL (8 , 2)	此清單 (NUMTICKETS*) 的總價格PRICE PERTICKET。
LISTTIME	TIMESTAMP	張貼此傳單的完整日期和時間，例如 2008-03-18 07:19:35 。

SALES table

欄名稱	資料類型	描述
SALESID	INTEGER	主要索引鍵，每個列的唯一 ID 值。每個列代表特定活動一或多個門票的銷售 (如特定清單中所提供)。
LISTID	INTEGER	LISTING 資料表的外鍵參考。
SELLERID	INTEGER	USERS 資料表的外鍵參考 (售出票證的使用者)。
BUYERID	INTEGER	USERS 資料表的外鍵參考 (購買票證的使用者)。
EVENTID	INTEGER	EVENT 資料表的外鍵參考。
DATEID	SMALLINT	DATE 資料表的外鍵參考。
QTYSOLD	SMALLINT	門票銷售數，從 1 到 8 。(您最多可以在單一交易中銷售 8 張門票。)
PRICEPAID	DECIMAL (8 , 2)	為門票支付的總價，例如 75.00 或 488.00 。門票的個別價格為 PRICEPAID/QTYSOLD。
COMMISSION	DECIMAL (8 , 2)	公司會從銷售中收取 15% 佣金，例如 11.25 或 73.20 。賣方會收到 85% PRICEPAID 的值。
SALETIME	TIMESTAMP	銷售結束的完整日期和時間，例如 2008-05-24 06:21:47 。

Amazon Redshift 最佳實務

接下來，您可以找到有關規劃概念驗證、設計資料表、載入資料至資料表，以及為 Amazon Redshift 撰寫查詢的最佳實務，以及有關使用 Amazon Redshift Advisor 的討論。

Amazon Redshift 是不一樣的其他 SQL 數據庫系統。為了完全實現 Amazon Redshift 架構的優點，您必須特別設計、建置及載入您的資料表，以使用大規模平行處理、單欄式資料儲存體，以及單欄式資料壓縮。如果您的資料載入與查詢執行時間比預期長，或比您希望的長，表示您可能忽略了關鍵資訊。

如果您是經驗豐富的資料 SQL 庫開發人員，強烈建議您在開始開發 Amazon Redshift 資料倉儲之前先檢閱本主題。

如果您不熟悉開發 SQL 資料庫，這個主題不是最好的起點。我們建議您先閱讀[執行命令，在 Amazon Redshift 入門指南中定義和使用資料倉儲中的資料庫](#)，並親自嘗試範例。

在本主題中，您可以找到最重要的開發原則概觀，以及實作這些原則的具體技巧、範例及最佳實務。沒有任何單一實務可以套用至每個應用程式。在完成資料庫設計之前，請評估所有的選項。如需詳細資訊，請參閱[自動資料表最佳化](#)、[在 Amazon Redshift 中載入資料](#)、[查詢效能調校](#)，以及參考章節。

主題

- [為 Amazon Redshift 進行概念證明 \(POC \)](#)
- [設計資料表的 Amazon Redshift 最佳實務](#)
- [載入資料的 Amazon Redshift 最佳實務](#)
- [設計查詢的 Amazon Redshift 最佳實務](#)
- [遵循 Amazon Redshift 顧問的建議](#)

為 Amazon Redshift 進行概念證明 (POC)

Amazon Redshift 是一種熱門的雲端資料倉儲，提供全受管雲端服務，可與組織的 Amazon 簡易儲存服務資料湖、即時串流、機器學習 (ML) 工作流程、交易工作流程等整合。以下各節將引導您完成在 Amazon Redshift 上進行概念驗證 (POC) 的過程。此處的資訊可協助您為 POC 設定目標，並利用可自動佈建和設定 POC 服務的工具。

Note

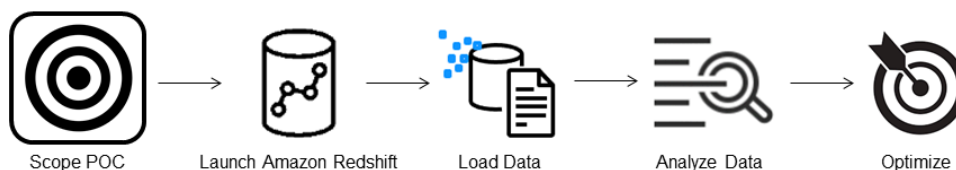
如需以 PDF 格式提供此資訊的副本，請在[亞馬遜紅移資源頁面上選擇執行您自己的 Redshift POC 連結](#)。

執行 Amazon Redshift 的 POC 時，您可以測試、證明和採用各種功能，包括 best-in-class 安全功能、彈性擴展、輕鬆整合和擷取，以及靈活的分散式資料架構選項。



請按照以下步驟進行成功的 POC。

步驟 1：設定 POC 的範圍



執行 POC 時，您可以選擇使用自己的資料，也可以選擇使用基準測試資料集。當您選擇自己的資料時，您可以針對資料執行自己的查詢。使用基準測試資料，範例查詢會隨基準提供。如果您還沒準備好[使用自己的資料進行 POC](#)，請參閱[使用範例資料集](#)以取得更多詳細資訊。

一般而言，我們建議您將兩週的資料用於 Amazon Redshift POC。

通過執行以下操作開始：

1. 確定您的業務和功能需求，然後向後工作。常見的範例包括：更快的效能、更低的成本、測試新的工作負載或功能，或 Amazon Redshift 與其他資料倉儲之間的比較。
2. 設定成為 POC 成功標準的特定目標。例如，通過更快的性能，列出您希望加速的前五個進程的列表，並包括當前的運行時間以及所需的運行時間。這些可以是報告、查詢、ETL 程序、資料擷取或您目前的痛點。

3. 確定運行測試所需的特定範圍和工件。您需要哪些資料集才能遷移或持續導入 Amazon Redshift，以及執行測試以根據成功標準進行測量需要哪些查詢和程序？有兩種方式可以進行：

攜帶您自己的資料

- 要測試您自己的數據，請提出測試成功標準所需的最小可行數據成品列表。例如，如果您目前的資料倉儲有 200 個資料表，但您要測試的報表只需要 20 個資料表，則只需使用較小的資料表子集，就可以更快地執行 POC。

使用範例資料集

- 如果您沒有準備好自己的資料集，您仍然可以使用業界標準的基準資料集 (例如 [TPC-DS](#) 或 [TPC-H](#)) 開始在 Amazon Redshift 上執行 POC，並執行範例基準測試查詢來利用 Amazon Redshift 的強大功能。您可以在建立 Amazon Redshift 資料倉儲之後，從您的資料倉儲中存取這些資料集。如需如何存取這些資料集和範例查詢的詳細指示，請參閱 [步驟 2：啟動 Amazon Redshift](#)。

步驟 2：啟動 Amazon Redshift



Amazon Redshift 透過快速、簡單且安全的大規模雲端資料倉儲，加快您獲得洞見的時間。您可以在 [Redshift 無伺服器主控台](#) 上啟動倉儲，並在幾秒鐘內從資料獲得深入解析，以快速開始。使用 Redshift 無伺服器，您可以專注於交付業務成果，而不必擔心管理資料倉儲的問題。

設定 Amazon Redshift 無伺服器

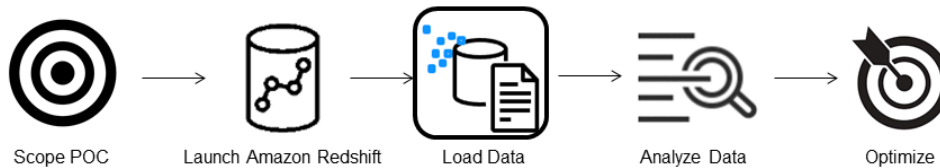
第一次使用 Redshift 無伺服器時，主控台會引導您完成啟動倉儲所需的步驟。您也可能有資格獲得帳戶中 Redshift 無伺服器用量的抵免額度。如需有關選擇免費試用的詳細資訊，請參閱 [Amazon Redshift 免費試用版](#)。遵循 Amazon Redshift Redshift 入門指南中 [使用 Redshift 無伺服器建立資料倉儲](#) 中的步驟，以使用 Redshift 無伺服器建立資料倉儲。如果您沒有想要載入的資料集，本指南也會提供如何載入範例資料集的步驟。

如果您之前已在帳戶中啟動 Redshift 無伺服器，請按照 Amazon Redshift 管理指南中的 [命名空間建立工作群組](#) 中的步驟進行操作。倉儲可用之後，您可以選擇載入 Amazon Redshift 中提供的範例資料。

如需使用 Amazon Redshift 查詢編輯器 v2 載入資料的相關資訊，請參閱 Amazon Redshift 管理指南中的 [載入範例資料](#)。

如果您要使用自己的資料而不是載入範例資料集，請參閱 [步驟 3：載入資料](#)。

步驟 3：載入資料



啟動 Redshift 無伺服器之後，下一步就是載入 POC 的資料。無論您要上傳簡單的 CSV 檔案、從 S3 擷取半結構化資料，還是直接串流資料，Amazon Redshift 都能提供彈性，讓您能夠快速輕鬆地將資料從來源移至 Amazon Redshift 表格。

選擇下列其中一種方法來載入資料。

上傳本機檔案

若要快速擷取和分析，您可以使用 [Amazon Redshift 查詢編輯器 v2](#) 輕鬆地從本機桌面載入資料檔案。它具有處理各種格式的文件，例如 CSV，JSON，AVRO，實木複合地板，ORC 等的功能。若要讓身為管理員的使用者能夠使用查詢編輯器 v2 從本機桌面載入資料，您必須指定通用的 Amazon S3 儲存貯體，而且使用者帳戶必須 [設定適當的許可](#)。您可以 [使用查詢編輯器 V2 在 Amazon Redshift 中輕鬆安全地追蹤資料載入以取得 step-by-step 指導](#)。

加載 Amazon S3 文件

若要將資料從 Amazon S3 儲存貯體載入 Amazon Redshift，請先使用 [COPY 命令](#)，指定來源 Amazon S3 位置，並將 Amazon Redshift 表作為目標。確保已正確設定 IAM 角色和許可，以允許 Amazon Redshift 存取指定的 Amazon S3 儲存貯體。遵循 [教學課程：從 Amazon S3 載入資料](#) 以取得 step-by-step 指導。您也可以查詢編輯器 v2 中選擇「載入資料」選項，直接從 S3 儲存貯體載入資料。

持續資料擷取

[自動複製 \(處於預覽狀態\)](#) 是 [COPY 命令](#) 的延伸，可自動從 Amazon S3 儲存貯體連續載入資料。當您建立複製任務時，Amazon Redshift 會偵測在指定路徑中建立新的 Amazon S3 檔案時，然後自動載入這些檔案，而無需您介入。Amazon Redshift 會追蹤載入的檔案，以確認檔案只載入一次。如需如何建立複製工作的指示，請參閱 [COPYJOB\(預覽\)](#)

Note

自動複製目前處於預覽狀態，僅在特定 AWS 區域的佈建叢集中受支援。若要建立自動複製的預覽叢集，請參閱[從 Amazon S3 載入具有連續檔案擷取的資料表（預覽）](#)。

載入串流資料

串流擷取提供低延遲、高速的串流資料擷取功能，從 [Amazon Kinesis Data Streams](#) 和 [Amazon Managed Streaming for Apache Kafka](#) 匯入 Amazon Redshift。Amazon Redshift 串流擷取使用具體化視觀表，該視觀表會使用 [auto](#) 重新整理直接從串流更新。具體化視觀表會對應至串流資料來源。您可以在串流資料上執行篩選和彙總，做為具體化視觀表定義的一部分。如需從串流載入資料的 step-by-step 指引，請參閱 [Amazon Kinesis Data Streams 入門](#) 或適用於 [Apache Kafka 的 Amazon 受管串流入門](#)。

步驟 4：分析您的資料



在建立 Redshift 無伺服器工作群組和命名空間並載入資料之後，您可以從 [Redshift](#) 無伺服器主控台的導覽面板開啟查詢編輯器 v2，立即執行查詢。您可以使用查詢編輯器 v2，針對自己的資料集測試查詢功能或查詢效能。

使用 Amazon Redshift 查詢編輯器 v2 進行查詢

您可以從 Amazon Redshift 主控台存取查詢編輯器 v2。請參閱[使用 Amazon Redshift 查詢編輯器 v2 簡化資料分析](#)，以取得如何使用查詢編輯器 v2 設定、連接和執行查詢的完整指南。

或者，如果你想運行一個負載測試作為你的 POC 的一部分，你可以通過以下步驟來安裝和運行 Apache JMeter 的做到這一點。

運行使用阿帕奇 JMeter 的負載測試

若要執行負載測試以模擬「N」使用者同時向 Amazon Redshift 提交查詢，您可以使用 [Apache JMeter](#)，這是一種以 Java 為基礎的開放原始碼工具。

若要安裝和設定 Apache JMeter 以針對 Redshift 無伺服器工作群組執行，請遵循[使用分析自動化工具組自動化 Amazon Redshift 負載測試](#)中的說明進行操作。AWS 它使用[AWS 分析自動化工具組 \(AAA\)](#)，這是一個用於動態部署 Redshift 解決方案的開放原始碼公用程式，自動啟動這些資源。如果您已將自己的資料載入 Amazon Redshift，請務必執行步驟 #5 — 自訂 SQL 選項，以確保您提供想要針對表格測試的適當 SQL 陳述式。使用查詢編輯器 v2 一次測試這些 SQL 陳述式，以確保它們執行時沒有錯誤。

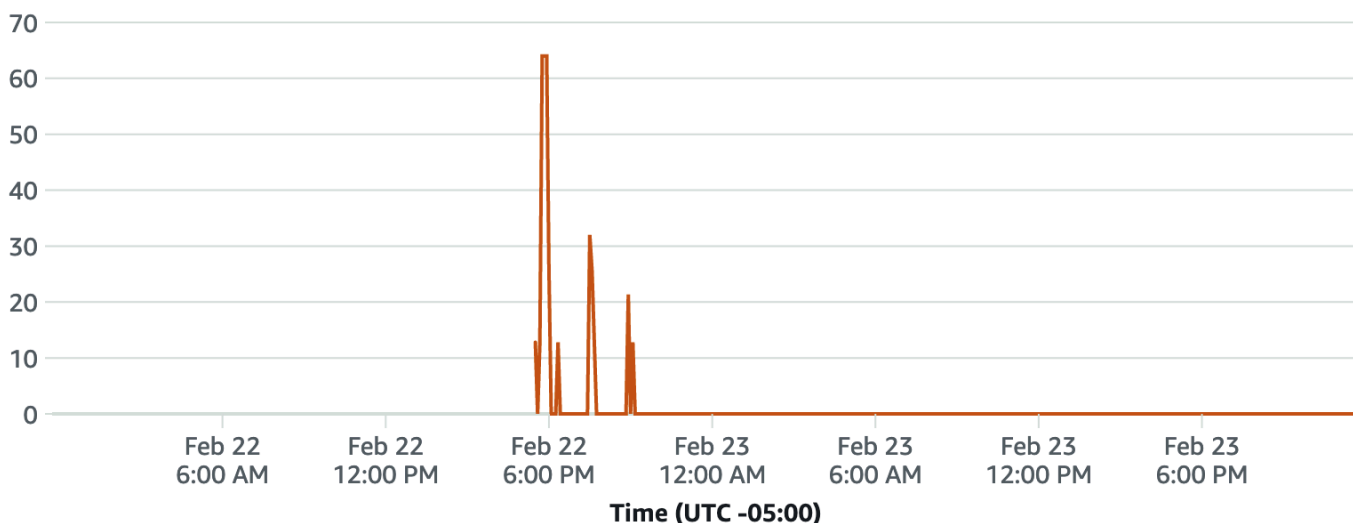
完成自訂 SQL 陳述式並完成測試計劃之後，請針對 Redshift 無伺服器工作群組儲存並執行測試計劃。若要監視測試進度，請開啟 [Redshift 無伺服器主控台](#)，瀏覽至 [查詢和資料庫監視]，選擇 [查詢歷程記錄] 索引標籤，然後檢視查詢的相關資訊。

對於效能測量結果，請選擇 Redshift 無伺服器主控台上的「資料庫效能」索引標籤，以監視「資料庫連線」和「CPU 使用率」等指標。您可以在此檢視圖表以監控使用的 RPU 容量，並觀察 Redshift Serverless 如何在工作群組上執行負載測試時自動擴展以滿足並行工作負載需求。

RPU capacity used

Overall capacity in Redshift processing units (RPUs).

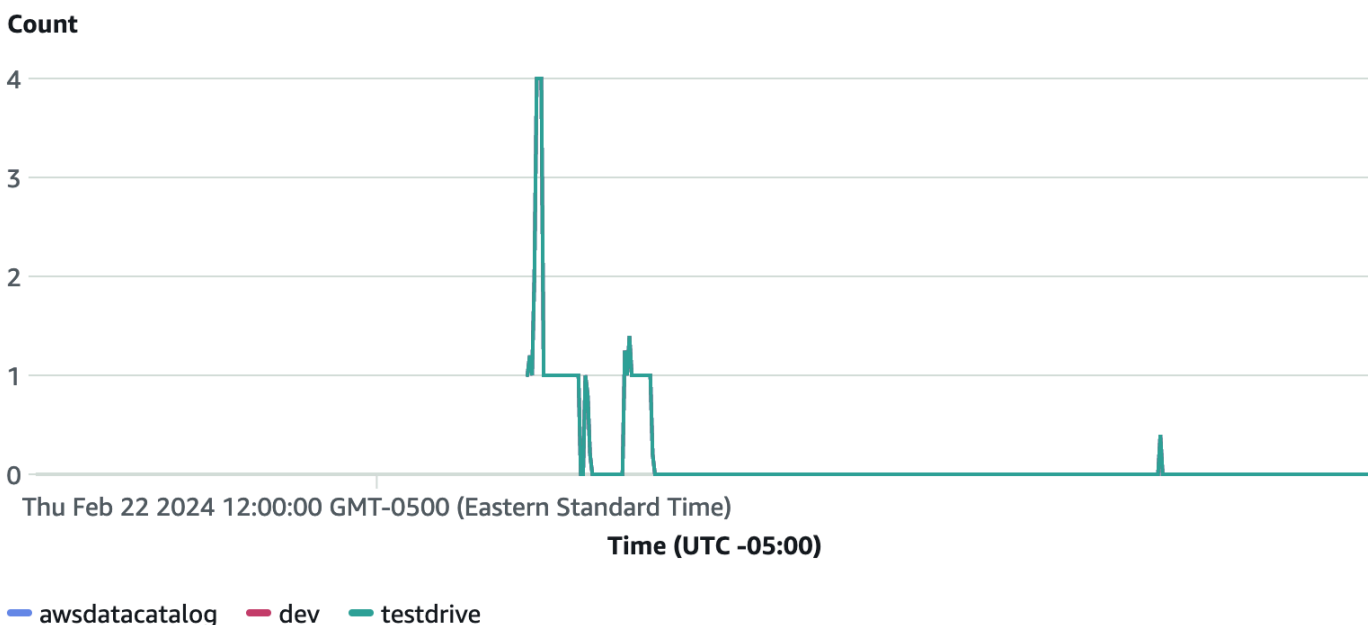
Average capacity used



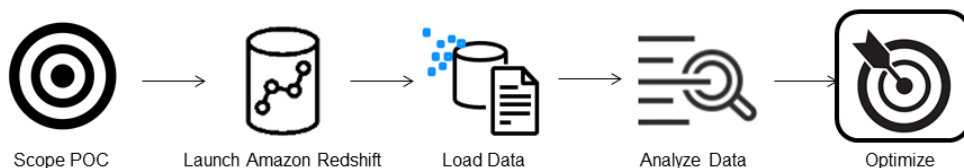
資料庫連線是另一個有用的測量結果，可在執行負載測試時監視，以瞭解工作群組在指定時間如何處理多個並行連線，以滿足日益增加的工作負載需求。

Database connections

The number of active database connections.



步驟 5：最佳化



Amazon Redshift 透過提供各種組態和功能來支援個別使用案例，讓成千上萬的使用者每天處理 EB 的資料，並強化其分析工作負載。在這些選項之間進行選擇時，客戶正在尋找可協助他們判斷最佳資料倉儲組態以支援 Amazon Redshift 工作負載的工具。

試駕

您可以使用「[試用方案](#)」，根據潛在的組態自動重新執行現有工作負載，並分析對應的輸出，以評估要將工作負載移轉至的最佳目標。如需使用試用方案評估不同 [Amazon Redshift 組態的相關資訊](#)，請參閱 [使用 Redshift 試用方案尋找工作負載的最佳 Amazon Redshift 組態](#)。

設計資料表的 Amazon Redshift 最佳實務

在您規劃資料庫時，有些關鍵的資料表設計決策會嚴重影響整體查詢效能。這些設計選擇也會嚴重影響儲存需求，進而因減少 I/O 操作數及處理查詢所需的記憶體而影響查詢效能。

在本節中，您可以找到最重要的設計決策摘要及最佳化查詢效能的最佳實務。[自動資料表最佳化](#) 提供資料表設計選項的詳細說明與範例。

主題

- [選擇最佳的排序索引鍵](#)
- [選擇最佳的分佈方式](#)
- [讓我們COPY選擇壓縮編碼](#)
- [定義主索引鍵與外部索引鍵限制條件](#)
- [使用可能的最小欄位大小](#)
- [在日期欄位中使用日期/時間資料類型](#)

選擇最佳的排序索引鍵

Amazon Redshift 依據排序索引鍵，將您的資料以排序順序儲存於磁碟。Amazon Redshift 查詢最佳化工具使用排序順序來決定最佳查詢計畫。

Note

當您使用自動資料表最佳化時，您不需要選擇資料表的排序索引鍵。如需詳細資訊，請參閱[自動資料表最佳化](#)。

部分最佳做法建議如下：

- 若要讓 Amazon Redshift 選擇適當的排序順序，請為排序索引鍵指定 AUTO。
- 如果會經常查詢較新的資料，請指定時間戳記欄位做為排序索引鍵的前導欄。

查詢將更有效率，因為它們會跳過位於時間範圍之外的整個區塊。

- 如果您經常對一個欄位進行範圍篩選或相等篩選，請指定該欄位做為排序索引鍵。

Amazon Redshift 可跳過讀取該欄位的整個資料區塊。它可以這麼做是因為它會追蹤儲存於每個區塊的最小與最大欄位值，並跳過不適用於述詞範圍的區塊。

- 如果您經常聯結資料表，請指定聯結欄位做為排序索引鍵與分佈索引鍵。

這麼做可以讓查詢最佳化工具選擇排序合併聯結，而非較慢的雜湊聯結。由於資料已經以聯結索引鍵進行排序，查詢最佳化工具可跳過排序合併聯結的排序階段。

選擇最佳的分佈方式

當您執行查詢時，查詢最佳化工具會視需要將資料列重新配送至運算節點，以執行任何聯結與彙總。選擇資料表配送樣式的目的是，藉由在執行查詢之前，將資料配置至需要的位置，以降低重新配送步驟所帶來的影響。

Note

當您使用自動資料表最佳化時，您不需要選擇資料表的分佈樣式。如需詳細資訊，請參閱[自動資料表最佳化](#)。

部分最佳做法建議如下：

1. 將事實資料表與一維度資料表配送至它們的共通欄位。

您的事實資料表只能有一個分佈索引鍵。聯結其他索引鍵的任何資料表都不會與事實資料表共置。依據其聯結的頻率與聯結資料列的大小，選擇一個維度進行共置。將維度表格的主索引鍵和事實資料表的對應外來索引鍵指定為DISTKEY。

2. 依據篩選過的資料集大小，選擇最大的維度。

只有用於聯結的資料列必須配送，因此應考量篩選之後的資料集大小，而非資料表的大小。

3. 在經過篩選的結果集中選擇高基數的欄位。

例如，如果您將銷售資料表配送至日期欄位，您可能會獲得相當平均的資料配送，除非大部分的銷售是季節性的。但是，如果您通常使用限制範圍的述詞來篩選以縮小日期期間，則大多數篩選過的資料列會發生在一組有限的分割上，而且查詢工作負載會發生偏斜。

4. 變更某些維度表以使用ALL分佈。

如果維度資料表無法與事實資料表或其他重要的聯結資料表共置，您可以藉由將整個資料表發佈至所有節點，以大幅提升查詢效能。使用ALL分配會使儲存空間需求倍增，並增加載入時間和維護作業，因此您應該在選擇ALL分配之前權衡所有因素。

若要讓 Amazon Redshift 選擇適當的分佈樣式，請指定分佈樣式的 AUTO。

如需選擇分佈樣式的相關資訊，請參閱 [查詢最佳化的資料分佈](#)。

讓我們 COPY 選擇壓縮編碼

當您建立資料表時，可指定壓縮編碼，但在大多數情況下，自動壓縮可產生最佳結果。

ENCODEAUTO 是表格的預設值。當資料表設定為 ENCODEAUTO，Amazon Redshift 會自動管理表格中所有欄的壓縮編碼。如需詳細資訊，請參閱 [CREATE TABLE](#) 和 [ALTER TABLE](#)。

此指 COPY 令會分析您的資料，並將壓縮編碼自動套用至空白資料表，做為載入作業的一部分。

選擇壓縮編碼時，自動壓縮會平衡整體效能。如果排序索引鍵欄位的壓縮比相同查詢中的其他欄位高出許多，限制範圍的掃描執行效果可能較差。因此，自動壓縮會選擇效率較低的壓縮編碼，以維持排序索引鍵欄位與其他欄位之間的平衡。

假設您資料表的排序索引鍵為日期或時間戳記，而且該資料表使用許多大型 varchar 欄位。在此情況下，如果不壓縮排序索引鍵欄位，您可能會獲得較高的效能。在資料表上執行 [ANALYZE COMPRESSION](#) 命令，然後使用編碼以建立新資料表，但省略排序索引鍵的壓縮編碼。

自動壓縮編碼有效能上的成本，但僅限於資料表是空的，而且尚未有壓縮編碼的情況。對於經常建立的短期資料表和資料表 (例如臨時資料表)，請使用自動壓縮載入資料表一次，或執行命 ANALYZE COMPRESSION 令。然後，使用這些編碼建立新資料表。您可以將編碼加入 CREATE TABLE 陳述式中，或使用 CREATE TABLE LIKE 以相同編碼建立新資料表。

如需詳細資訊，請參閱 [利用自動壓縮載入資料表](#)。

定義主索引鍵與外部索引鍵限制條件

在適當的位置定義資料表之間的主索引鍵與外部索引鍵限制條件。儘管它們只是資訊性的，但查詢最佳化工具會使用這些限制條件來產生更有效的查詢計畫。

除非應用程式強制執行限制，否則請勿定義主索引鍵和外部索引鍵限制條件。Amazon Redshift 不會強制執行唯一的主索引鍵和外部索引鍵限制條件。

如需有關 Amazon Redshift 如何使用限制條件的其他資訊，請參閱 [資料表限制](#)。

使用可能的最小欄位大小

為方便起見，請勿使用最大欄位大小。

反之，舉例而言，請考量您可能在資料欄位中儲存的最大值，並相應地調整其大小。例如，用於存儲郵局使用的美國州和地區縮寫的CHAR列只需要CHAR (2)。

在日期欄位中使用日期/時間資料類型

Amazon Redshift 比CHAR或更有效率地存放DATE和TIMESTAMP資料VARCHAR，因此可提高查詢效能。根TIMESTAMP據您需要的解析度使用DATE或資料類型，而不是儲存日期/時間資訊時的字元類型。如需詳細資訊，請參閱[日期時間 \(Datetime\) 類型](#)。

載入資料的 Amazon Redshift 最佳實務

載入極大的資料集將非常耗時，並消耗大量運算資源。資料的載入方式也會影響查詢效能。本節介紹使用COPY命令、大量插入和臨時資料表有效載入資料的最佳作法。

主題

- [透過教學課程瞭解如何載入資料](#)
- [使用COPY命令載入資料](#)
- [使用單個COPY命令從多個文件加載](#)
- [載入資料檔案](#)
- [壓縮您的資料檔案](#)
- [在載入前後驗證資料檔案](#)
- [使用多資料列插入](#)
- [使用大量插入](#)
- [以排序索引鍵順序載入資料](#)
- [將資料載入循序區塊](#)
- [使用時間序列資料表](#)
- [排程維護時段](#)

透過教學課程瞭解如何載入資料

[教學課程：從 Amazon S3 載入資料](#)引導您逐步完成將資料上傳到 Amazon S3 儲存貯體的步驟，然後使用COPY命令將資料載入到表格中。此教學課程包括協助排解載入錯誤的問題，並比較從單一檔案與從多個檔案載入之間的效能差異。

使用COPY命令載入資料

該COPY命令從 Amazon S3、Amazon、亞馬 Amazon DynamoDB 或遠端主機上的多個資料來源 parallel 載入資料。EMRCOPY相較於使用INSERT陳述式，載入大量資料的效率更高，而且也能更有效地儲存資料。

若要取得有關使用COPY指令的更多資訊，請參閱[從 Amazon S3 載入資料](#)和[從 Amazon DynamoDB 資料表載入資料](#)。

使用單個COPY命令從多個文件加載

Amazon Redshift 可以從多個已壓縮的資料檔案平行載入資料。您可以透過使用 Amazon S3 物件字首或使用資訊清單檔案來指定要載入的檔案。

不過，如果您使用多個並行COPY命令從多個檔案載入一個資料表，Amazon Redshift 就會強制執行序列化載入。如果表格定義了排序欄，這種類型的負載速度會慢得多，並且需要在結束時VACUUM進行處理。若要取得有關使用 parallel 載COPY入資料的更多資訊，請參閱 [〈〉 從 Amazon S3 載入資料](#)。

載入資料檔案

來源資料檔案有不同的格式，並會使用不同的壓縮演算法。使用COPY命令載入資料時，Amazon Redshift 會載入 Amazon S3 儲存貯體前綴參照的所有檔案。(字首是物件索引鍵名稱開頭的字元字串。) 如果前置詞指的是MPP多個可以分割的檔案或檔案，Amazon Redshift 會利用 Amazon Redshift 的架構 parallel 載入資料。這會在叢集中的節點上劃分工作負載。相反地，當您從無法分割的檔案載入資料時，Amazon Redshift 會強制執行序列化載入，而此速度會慢很多。以下各節說明將不同檔案類型載入 Amazon Redshift 的建議方式 (視其格式和壓縮而定)。

從可分割的檔案載入資料

下列檔案可在載入資料時自動分割檔案：

- 一個未壓縮CSV的文件
- 一個CSV文件壓縮 BZIP
- 一個柱狀文件 (包/) ORC

Amazon Redshift 會自動將 128MB 或更大的檔案分割成多個區塊。列文件，特別是實木複合地板和 ORC，如果它們小於 128MB，則不會分割。Redshift 會利用平行運作的分割部分來載入資料。這可提供快速的載入效能。

從部可分割的檔案載入資料

使用其他壓縮演算法 (例如 JSONCSV, 或) 壓縮的檔案類型不會自動分割。GZIP對於這些檔案, 我們建議將資料手動分割為多個大小接近的較小文件 (壓縮後的 1 MB 到 1 GB)。此外, 檔案數量應為您叢集中分割的倍數。如需如何將資料分割成多個檔案的詳細資訊, 以及使用載入資料的範例COPY, 請參閱[從 Amazon S3 載入資料](#)。

壓縮您的資料檔案

當您想要壓縮大型載入檔案時, 我們建議您使用 gzip、lzop、bzip2 或 ZStandard 來壓縮檔案, 並將資料分割成多個較小的檔案。

使用GZIP指COPY令指定LZOPBZIP2、或ZSTD選項。此範例會從以管道分隔的 lzop 檔案載入TIME資料表。

```
copy time
from 's3://amzn-s3-demo-bucket/data/timerows.lzo'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
lzop
delimiter '|';
```

在某些情況下, 您不必分割未壓縮的資料檔案。如需分割資料的詳細資訊和使用COPY載入資料的範例, 請參閱 [〈〉 從 Amazon S3 載入資料](#)。

在載入前後驗證資料檔案

從 Amazon S3 載入資料之前, 請先確認 Amazon S3 儲存貯體包含所有正確的檔案, 而且只包含這些檔案。如需詳細資訊, 請參閱[驗證您的儲存貯體中存在正確的檔案](#)。

在載入操作完成後, 查詢 [STL_LOAD_COMMITS](#) 系統資料表, 確認預期的檔案已經載入。如需詳細資訊, 請參閱[驗證資料已正確載入](#)。

使用多資料列插入

如果指COPY令不是選項, 而您需要SQL插入, 請盡可能使用多列插入。當您的資料一次只有一列或幾列時, 資料壓縮是沒有效率的。

多資料列插入可批次執行一系列的插入, 藉此提升效能。下列範例會使用單一INSERT陳述式, 將三個資料列插入四欄資料表中。這仍是小型插入, 僅用於說明多資料列插入的語法。

```
insert into category_stage values
```

```
(default, default, default, default),  
(20, default, 'Country', default),  
(21, 'Concerts', 'Rock', default);
```

如需詳細資訊和範例，請參閱 [INSERT](#)。

使用大量插入

使用具有SELECT子句的大量插入作業，以進行高效能的資料插入。

當您必須從一個資料表將資料或資料的子集移至另一個資料表時，請使用 [INSERT](#) 與 [CREATE TABLE AS](#) 命令。

例如，下面的INSERT語句選擇從CATEGORY表中的所有行，並將它們插入到 CATEGORY _ STAGE 表。

```
insert into category_stage  
(select * from category);
```

下列範例會建立 CATEGORY _ STAGE 做為的副本，CATEGORY並CATEGORY將中的所有列插入 CATEGORY _ STAGE。

```
create table category_stage as  
select * from category;
```

以排序索引鍵順序載入資料

以排序索引鍵的順序載入資料，以避免必須清空。

如果每個批次第新資料皆遵循您資料表中的現有資料列，您的資料將按照排序順序正確儲存，而且無需執行清空。您不需要在每次載入時預先COPY排序資料列，因為它會在載入時對每批傳入資料進行排序。

例如，您每天根據當天的活動來載入資料。如果您的排序索引鍵為時間戳記欄位，您的資料將按照排序順序儲存。會發生這樣的順序，是因為當天的資料總是會附加至前一天資料的結尾。如需詳細資訊，請參閱[以排序索引鍵順序載入您的資料](#)。如需清除操作的相關資訊，請參閱[清除資料表](#)。

將資料載入循序區塊

如果您需要新增大量資料，將資料依據排序順序載入循序區塊，即可無需執行清除。

例如，假設您需要載入 2017 年 1 月至 2017 年 12 月的活動資料表。假設每個月都在單一檔案中，請載入一月、二月等的資料列。當您載入完成時，資料表已完全排序，無需執行清空。如需詳細資訊，請參閱[使用時間序列資料表](#)。

載入非常大的資料集時，需要排序的空間可能會超過可用的總空間。藉由將資料載入至較小的區塊，可在每次載入時，使用較少的中間排序空間。此外，載入較小的區塊可讓您在 COPY 失敗且復原時更容易重新啟動。

使用時間序列資料表

如果您的資料有固定的保留期間，您可以將資料整理為時間序列資料表的順序。在這樣的順序中，每個資料表都是相同的，但包含不同時間範圍的資料。

您只需在相應的表上運行 DROPTABLE 命令即可輕鬆刪除舊數據。這種方法比執行大規模 DELETE 處理程序快得多，而且不必執行後續 VACUUM 處理程序來回收空間。要隱藏數據存儲在不同的表中的事實，您可以創建一個 UNIONALL 視圖。刪除舊資料時，請細化 UNIONALL 檢視以移除放置的資料表。同樣的，當您將新的期間載入至新資料表時，請將新資料表新增至該檢視。若要傳送訊號至最佳化器以跳過掃描與查詢篩選條件不匹配的資料表，您的檢視定義將會篩選對應至每個資料表的日期範圍。

避免在檢視中有太多資料 UNIONALL 表。每個額外的資料表都會增加一些查詢的處理時間。資料表不需要使用相同的時間範圍。例如，您可能有不同期間的資料表，例如每日、每月及每年。

如果您使用時間序列資料表，並以時間戳記欄位做為排序索引鍵，將可有效地按照排序索引鍵的順序載入資料。如此便不必清除以重新排序資料。如需詳細資訊，請參閱[以排序索引鍵順序載入您的資料](#)。

排程維護時段

如果排定的維護發生在查詢執行時，查詢將會終止並轉返，您必須予以重新啟動。排定長時間執行的作業 (例如大型資料載入或 VACUUM 作業)，以避免維護時間。您還可以通過以較小的增量執行數據載入並管理操作大小，將風險降到最低，並在需要時更輕鬆地重新啟 VACUUM 動。如需詳細資訊，請參閱[將資料載入循序區塊](#) 和 [清空資料表](#)。

設計查詢的 Amazon Redshift 最佳實務

為達到最大的查詢效能，在建立查詢時，請遵循這些建議：

- 依據最佳實務設計資料表，為查詢效能提供穩固的基礎。如需詳細資訊，請參閱[設計資料表的 Amazon Redshift 最佳實務](#)。
- 避免使用 `select *`。僅包含您實際需要的欄位。
- 使用 [CASE 條件式表達式](#) 執行複雜的彙總，而非從相同的資料表多次選取。

- 除非絕對必要，否則請勿使用交叉聯結。沒有聯結條件的這些聯結，會導致兩個資料表的笛卡兒乘積。交叉聯結通常會執行為巢狀迴圈聯結，這是可能的聯結類型中最慢的。
- 如果查詢中的一個資料表僅用於述詞條件，而且子查詢返回少量資料列 (小於約 200)，則使用子查詢。下列範例會使用子查詢來避免連結LISTING資料表。

```
select sum(sales.qtysold)
from sales
where salesid in (select listid from listing where listtime > '2008-12-26');
```

- 使用述詞以盡可能限制資料集。
- 在述詞中，使用最便宜的運算子。[比較條件](#) 運算子比 [LIKE](#) 運算子好。LIKE 運算子仍然優於 [SIMILAR 至](#) 或 [POSIX 運算子](#)。
- 避免在查詢述詞中使用函式。使用函式需要大量資料列以解析查詢的中間步驟，因此會提高查詢成本。
- 如果可能的話，請使用WHERE子句來限制資料集。接著，查詢規劃器可使用資料列順序，協助判斷哪些記錄符合條件，以跳過掃描大量的磁碟區塊。如果不使用它，查詢執行引擎必須完整掃描參與的欄位。
- 新增述詞以篩選參與聯結的資料表，即使這些述詞套用相同的篩選條件。查詢會傳回相同的結果集，但 Amazon Redshift 可在掃描步驟之前篩選聯結資料表，並有效地跳過掃描這些資料表的區塊。如果您在用於聯結條件的欄位上進行篩選，就不需要備援篩選條件。

例如，假設您要聯結 SALES 與 LISTING 以搜尋 12 月之後列出之門票的門票銷售，依照銷售者分組。這兩個資料表皆以日期排序。以下查詢聯結資料表上的共同索引鍵，並篩選大於 12 月 1 日的 listing.listtime 值。

```
select listing.sellerid, sum(sales.qtysold)
from sales, listing
where sales.salesid = listing.listid
and listing.listtime > '2008-12-01'
group by 1 order by 1;
```

子WHERE句不包含的述詞sales.saletime，因此會強制執行引擎掃描整個SALES資料表。如果您知道篩選條件會導致較少的資料列參與聯結，那麼也請加入該篩選條件。下列範例大幅縮短執行時間。

```
select listing.sellerid, sum(sales.qtysold)
from sales, listing
where sales.salesid = listing.listid
```

```
and listing.listtime > '2008-12-01'  
and sales.saletime > '2008-12-01'  
group by 1 order by 1;
```

- 在 GROUP BY 子句中使用排序索引鍵，以便查詢規劃工具可以使用更有效率的彙總。當查詢的 GROUP BY 清單只包含排序索引鍵資料行 (其中之一也是分佈索引鍵) 時，查詢可能符合單階段彙總的資格。GROUPBY 清單中的排序索引鍵欄必須包含第一個排序索引鍵，然後包含您要按排序索引鍵順序使用的其他排序索引鍵。例如，使用第一個排序索引鍵、第一和第二個排序索引鍵，第一、二和第三個排序索引鍵等等，都是有效的。使用第一和第三個排序索引鍵是無效的。

您可以執行 [EXPLAIN](#) 命令並在查詢的彙總步驟尋找 XN GroupAggregate，以確認使用單一階段彙總。

- 如果您同時使用 GROUP BY 和 ORDER BY 子句，請確保在兩個子句中以相同的順序放置列。亦即，使用以下方法。

```
group by a, b, c  
order by a, b, c
```

請勿使用以下方法。

```
group by b, c, a  
order by a, b, c
```

遵循 Amazon Redshift 顧問的建議

為協助您提高 Amazon Redshift 叢集的效能並降低營運成本，Amazon Redshift Advisor 提供您有關進行變更的具體建議。Advisor 透過分析叢集的效能與用量指標來開發其自訂的建議。這些量身訂製的建議與操作及叢集設定有關。為協助您排定最佳化的優先順序，Advisor 會依據影響的高低順序將建議項目進行排名。

Advisor 的建議以觀察相關效能統計與操作資料為基礎。Advisor 藉由在叢集上執行測試來判斷是否有測試值在指定範圍內，進而開發出觀察結果。如果測試結果在範圍之外，Advisor 將為您的叢集產生觀察結果。同時，Advisor 會建立建議，告訴您如何將觀察到的值恢復至最佳實務的範圍。Advisor 只會顯示將為效能與操作帶來大幅影響的建議。當 Advisor 判斷建議已獲得處理，便會從建議清單中移除該建議。

例如，假設您的資料倉儲包含大量未壓縮的資料表欄位。在此情況下，您可以使用 ENCODE 參數指定欄壓縮以重建資料表，以節省叢集儲存空間成本。在另一個範例中，假設 Advisor 觀察到您的叢集在未

壓縮的資料表資料中包含大量資料。在這種情況下，它會提供您SQL程式碼區塊來尋找適用於壓縮的資料表資料行，以及說明如何壓縮這些資料行的資源。

支援建議程式的 Amazon Redshift 區域

Amazon Redshift 顧問功能僅在以下情況下可用 AWS 地區：

- 美國東部 (維吉尼亞北部) 區域 (us-east-1)
- 美國東部 (俄亥俄) 區域 (us-east-2)
- 美國西部 (加利佛尼亞北部) 區域 (us-west-1)
- 美國西部 (奧勒岡) 區域 (us-west-2)
- 非洲 (開普敦) 區域 (af-south-1)
- 亞太區域 (香港) 區域 (ap-east-1)
- 亞太區域 (海德拉巴) 區域 (ap-south-2)
- 亞太區域 (雅加達) 區域 (ap-southeast-3)
- 亞太區域 (墨爾本) 區域 (ap-southeast-4)
- 亞太區域 (孟買) 區域 (ap-south-1)
- 亞太區域 (大阪) (ap-northeast-3)
- 亞太區域 (首爾) 區域 (ap-northeast-2)
- 亞太區域 (新加坡) 區域 (ap-southeast-1)
- 亞太區域 (雪梨) 區域 (ap-southeast-2)
- 亞太區域 (東京) 區域 (ap-northeast-1)
- 加拿大 (中部) 區域 (ca-central-1)
- 加拿大西部 (卡加利) 區域 (ca-west-1)
- 中國 (北京) 區域 (cn-north-1)
- 中國 (寧夏) 區域 (cn-northwest-1)
- 歐洲 (法蘭克福) 區域 (eu-central-1)
- 歐洲 (愛爾蘭) 區域 (eu-west-1)
- 歐洲 (倫敦) 區域 (eu-west-2)
- 歐洲 (米蘭) 區域 (eu-south-1)
- 歐洲 (巴黎) 區域 (eu-west-3)

- 歐洲 (西班牙) 區域 (eu-south-2)
- 歐洲 (斯德哥爾摩) 區域 (eu-north-1)
- 歐洲 (蘇黎世) 區域 (eu-central-2)
- 以色列 (特拉維夫) 區域 (il-central-1)
- 中東 (巴林) 區域 (me-south-1)
- 中東 (UAE) 區域 (me-central-1)
- 南美洲 (聖保羅) 區域 (sa-east-1)

主題

- [檢視 Amazon Redshift 顧問建議](#)
- [Amazon Redshift Advisor 建議](#)

檢視 Amazon Redshift 顧問建議

您可以使用 Amazon Redshift 馬 Amazon Redshift 主控台、Amazon Red API shift 或 AWS CLI。若要存取建議，您必須擁有IAM角色或身分redshift:ListRecommendations附加的權限。

在 Amazon Redshift 佈建的主控台上檢視 Amazon Redshift 顧問建議

您可以在以下位置檢視 Amazon Redshift 顧問建議 AWS Management Console.

在主控台上檢視針對 Amazon Redshift 馬遜紅移叢集的亞馬遜紅移顧問建議

1. 登入 AWS Management Console 並在<https://console.aws.amazon.com/redshiftv2/>打開 Amazon Redshift 控制台。
2. 在導覽選單中，選擇 Advisor。
3. 展開建議事項以檢視更多詳細資訊。在此頁面上，您可以排序和分組建議。

使用 Amazon Redshift 操作查看 Amazon Red API shift 顧問建議

您可以使用 Amazon Redshift 列出 Amazon Redshift 集群的 Amazon Redshift 顧問建議。API通常，您可以使用您選擇的程式設計語言來開發和應用程式，以呼叫redshift:ListRecommendationsAPI使用 AWS SDK。如需詳細資訊，請參閱 Amazon Redshift API 參考資料[ListRecommendations](#)中的。

使用以下方式檢視 Amazon Redshift 建議 AWS Command Line Interface 操作

您可以使用列出 Amazon Redshift 集群的 Amazon Redshift 顧問建議 AWS Command Line Interface。如需詳細資訊，請參閱中[的清單建議 AWS CLI 指令參考](#)。

Amazon Redshift Advisor 建議

Amazon Redshift Advisor 提供有關如何最佳化 Amazon Redshift 叢集以提高效能並節省營運成本的建議。如前所述，您可以在主控台找到各項建議的解釋。您可以在以下章節中找到這些建議的進一步詳細資訊。

主題

- [壓縮由載入的 Amazon S3 檔案物件 COPY](#)
- [隔離多個作用中資料庫](#)
- [重新分配工作負載管理 \(WLM\) 記憶體](#)
- [在期間跳過壓縮分析 COPY](#)
- [拆分 Amazon S3 對象加載 COPY](#)
- [更新資料表統計資訊](#)
- [啟用短期查詢加速](#)
- [更改資料表上的分佈索引鍵](#)
- [變更資料表上的排序索引鍵](#)
- [修改資料欄上的壓縮編碼](#)
- [資料類型建議](#)

壓縮由載入的 Amazon S3 檔案物件 COPY

該COPY命令利用 Amazon Redshift 中的大規模 parallel 處理 (MPP) 架構，以 parallel 方式讀取和載入資料。它可從 Amazon S3、DynamoDB 資料表讀取檔案，以及從一或多個遠端主機讀取文字輸出。

載入大量資料時，強烈建議您使用指COPY令從 S3 載入壓縮的資料檔案。壓縮大型資料集可節省上傳檔案至 Amazon S3 的時間。COPY還可以通過在讀取文件時解壓縮來加快加載過程。

分析

載入大型未壓縮資料集的長時間執行COPY命令，通常有機會大幅改善效能。Advisor 分析會識別載入大型未壓縮資料集的COPY命令。在此情況下，Advisor 會產生在 Amazon S3 中的來源檔案實作壓縮的建議。

建議

確保每COPY個載入大量資料或大量執行時間的資料，從 Amazon S3 擷取壓縮的資料物件。您可以以超級使用者身分執行下列命COPY令，識別從 Amazon S3 載入大型未壓縮資料集的SQL命令。

```
SELECT
    wq.userid, query, exec_start_time AS starttime, COUNT(*) num_files,
    ROUND(MAX(wq.total_exec_time/1000000.0),2) execution_secs,
    ROUND(SUM(transfer_size)/(1024.0*1024.0),2) total_mb,
    SUBSTRING(querytxt,1,60) copy_sql
FROM stl_s3client s
JOIN stl_query q USING (query)
JOIN stl_wlm_query wq USING (query)
WHERE s.userid>1 AND http_method = 'GET'
      AND POSITION('COPY ANALYZE' IN querytxt) = 0
      AND aborted = 0 AND final_state='Completed'
GROUP BY 1, 2, 3, 7
HAVING SUM(transfer_size) = SUM(data_size)
AND SUM(transfer_size)/(1024*1024) >= 5
ORDER BY 6 DESC, 5 DESC;
```

如果在您載入資料之後，暫存資料仍留存在 Amazon S3 (資料湖經常出現此情況)，以壓縮形式儲存此資料可降低您的儲存空間成本。

實作提示

- 壓縮之後的理想物件大小介於 1–128 MB 之間。
- 您可以用 gzip、lzop 或 bzip2 格式壓縮檔案。

隔離多個作用中資料庫

根據最佳實務，建議您隔離 Amazon Redshift 中的各個資料庫。查詢執行於特定資料庫，而且無法存取叢集上其他資料庫的資料。但是，您在叢集上所有資料庫執行的查詢，將共用相同的底層叢集儲存空間與運算資源。當單一叢集包含多個使用中資料庫時，其工作負載通常是不相關的。

分析

Advisor 分析會檢閱叢集上的所有資料庫，以找出同時執行的使用中工作負載。如果有同時執行的使用中工作負載，Advisor 分析會產生建議，建議您考慮將資料庫遷移至個別的 Amazon Redshift 叢集。

建議

考慮將經常被查詢的資料庫移至個別的專屬叢集。使用個別的叢集可減少資源爭奪並提升查詢效能。這是因為它能讓您設定各個叢集的儲存空間大小、成本，以及各個工作負載所需的效能。同時，不相關的工作負載通常會受益於不同的工作負載管理組態。

若要識別哪些資料庫正在使用中，您可以以超級使用者的身分執行此SQL命令。

```
SELECT database,
       COUNT(*) as num_queries,
       AVG(DATEDIFF(sec,starttime,endtime)) avg_duration,
       MIN(starttime) as oldest_ts,
       MAX(endtime) as latest_ts
FROM stl_query
WHERE userid > 1
GROUP BY database;
```

實作提示

- 因為使用者必須特別連接至各個資料庫，而查詢只能存取單一資料庫，因此將資料庫移至個別叢集對使用者的影響很小。
- 移動資料庫的選項之一是執行以下步驟：
 1. 暫時將目前叢集的快照還原至相同大小的叢集。
 2. 從新的叢集刪除所有資料庫，要移動的目標資料庫除外。
 3. 調整叢集大小以符合適當的節點類型，並計算資料庫的工作負載。

重新分配工作負載管理 (WLM) 記憶體

Amazon Redshift 將使用者查詢路由到 [實作手冊 WLM](#) 以進行處理。工作負載管理 (WLM) 定義這些查詢路由至佇列的方式。Amazon Redshift 會為每個佇列都配置叢集的一部分可用記憶體。佇列的記憶體再分配給佇列的查詢槽。

當佇列設定的插槽多於工作負載的需要時，配置到這些未使用插槽的記憶體將無法充分利用。減少設定的插槽以符合尖峰工作負載需求，可將未使用的記憶體重新分配給使用中的插槽，並藉此提升查詢效能。

分析

Advisor 分析將檢閱工作負載並行需求，以識別具有未使用插槽的查詢佇列。Advisor 發現以下情形時將會產生建議，建議您降低佇列中的插槽數量：

- 在整個分析過程中有完全未使用之插槽的佇列。
- 擁有四個以上插槽，而且在整個分析過程中至少有兩個未使用插槽的佇列。

建議

減少設定的插槽以符合尖峰工作負載需求，可將未使用的記憶體重新分配給使用中的插槽。針對其插槽不會完全充分利用的佇列，考慮減少其設定的插槽數量。若要識別這些佇列，您可以以超級使用者身分執行下列SQL命令，比較每個佇列的每小時尖峰時段需求。

```
WITH
generate_dt_series AS (select sysdate - (n * interval '5 second') as dt from (select
row_number() over () as n from stl_scan limit 17280)),
apex AS (
SELECT iq.dt, iq.service_class, iq.num_query_tasks, count(iq.slot_count) as
service_class_queries, sum(iq.slot_count) as service_class_slots
FROM
(select gds.dt, wq.service_class, wsc.num_query_tasks, wq.slot_count
FROM stl_wlm_query wq
JOIN stv_wlm_service_class_config wsc ON (wsc.service_class =
wq.service_class AND wsc.service_class > 5)
JOIN generate_dt_series gds ON (wq.service_class_start_time <= gds.dt AND
wq.service_class_end_time > gds.dt)
WHERE wq.userid > 1 AND wq.service_class > 5) iq
GROUP BY iq.dt, iq.service_class, iq.num_query_tasks),
maxes as (SELECT apex.service_class, trunc(apex.dt) as d, date_part(h,apex.dt) as
dt_h, max(service_class_slots) max_service_class_slots
from apex group by apex.service_class, apex.dt,
date_part(h,apex.dt))
SELECT apex.service_class - 5 AS queue, apex.service_class, apex.num_query_tasks AS
max_wlm_concurrency, maxes.d AS day, maxes.dt_h || ':00 - ' || maxes.dt_h || ':59' as
hour, MAX(apex.service_class_slots) as max_service_class_slots
FROM apex
JOIN maxes ON (apex.service_class = maxes.service_class AND apex.service_class_slots =
maxes.max_service_class_slots)
GROUP BY apex.service_class, apex.num_query_tasks, maxes.d, maxes.dt_h
ORDER BY apex.service_class, maxes.d, maxes.dt_h;
```

此資max_service_class_slots料欄代表該小時WLM查詢佇列中查詢插槽的數目上限。如有未利用的佇列，可[修改參數群組](#)以實作插槽減量最佳化，如 Amazon Redshift 管理指南所述。

實作提示

- 如果您的工作負載量變化很大，請確定分析有擷取到尖峰使用期間。如果沒有，請SQL重複運行前面的內容以監視峰值並發需求。
- 如需有關解譯上述SQL程式碼之查詢結果的詳細資訊，請參閱中的 [wlm_apex_hourly.sql 指令碼](#)。
GitHub

在期間跳過壓縮分析 COPY

當您使用COPY命令宣告的壓縮編碼將資料載入空白資料表時，Amazon Redshift 會套用儲存壓縮。此最佳化可確保叢集中的資料有效地儲存，即使是由最終使用者載入。需要套用壓縮的分析可能需要很長的時間。

分析

Advisor 分析會檢查自動壓縮分析所延遲的COPY作業。此分析會在資料載入時取樣資料以判斷壓縮編碼。此取樣類似 [ANALYZE COMPRESSION](#) 命令所執行的取樣。

當您將資料載入為結構化程序的一部分時，例如在隔夜擷取、轉換、load (ETL) 批次中，您可以事先定義壓縮。您也可以最佳化您的資料表定義，以永久跳過此階段，而不產生任何負面影響。

建議

若要透過跳過壓縮分析階段來改善COPY回應速度，請實作下列兩個選項之一：

- 建立使用COPY指令載入的任何表格時，請使用 column ENCODE 參數。
- 透過在COPY指令中提供COMPUPDATE OFF參數來完全關閉壓縮。

最佳解決方案通常是在資料表建立期間使用欄位編碼，因為此方法也同時保有將壓縮資料儲存於磁碟的好處。您可以使用指ANALYZECOMPRESSION令來建議壓縮編碼，但您必須重新建立資料表才能套用這些編碼。若要自動執行此程序，您可以使用 AWS [ColumnEncodingUtility](#)，在「GitHub」上找到。

若要識別最近觸發自動壓縮分析的COPY作業，請執行下列SQL命令。

```
WITH xids AS (  
    SELECT xid FROM stl_query WHERE userid>1 AND aborted=0
```

```

AND querytxt = 'analyze compression phase 1' GROUP BY xid
INTERSECT SELECT xid FROM stl_commit_stats WHERE node=-1)
SELECT a.userid, a.query, a.xid, a.starttime, b.complyze_sec,
a.copy_sec, a.copy_sql
FROM (SELECT q.userid, q.query, q.xid, date_trunc('s',q.starttime)
starttime, substring(querytxt,1,100) as copy_sql,
ROUND(datediff(ms,starttime,endtime)::numeric / 1000.0, 2) copy_sec
FROM stl_query q JOIN xids USING (xid)
WHERE (querytxt ilike 'copy %from%' OR querytxt ilike '% copy %from%')
AND querytxt not like 'COPY ANALYZE %') a
LEFT JOIN (SELECT xid,
ROUND(sum(datediff(ms,starttime,endtime))::numeric / 1000.0,2) complyze_sec
FROM stl_query q JOIN xids USING (xid)
WHERE (querytxt like 'COPY ANALYZE %'
OR querytxt like 'analyze compression phase %')
GROUP BY xid ) b ON a.xid = b.xid
WHERE b.complyze_sec IS NOT NULL ORDER BY a.copy_sql, a.starttime;

```

實作提示

- 請確定在ETL程序期間建立的所有大小相當大小的資料表 (例如臨時資料表和暫存資料表) 都宣告所有資料行的壓縮編碼 (第一個排序索引鍵除外)。
- 針對前面的命令所識別的每個SQL命令，估計載入之COPY資料表的預期存留期大小。如果您有信心資料表將維持極小，請以 COMPUPDATE OFF 參數一併關閉壓縮。否則，請先使用明確壓縮建立資料表，然後再使用COPY指令載入資料表。

拆分 Amazon S3 對象加載 COPY

該COPY命令利用 Amazon Redshift 中的大規模 parallel 處理 (MPP) 架構，從 Amazon S3 上的檔案讀取和載入資料。此命COPY令會從多個檔案 parallel 載入資料，將工作負載劃分叢集中的各個節點。若要達到最佳傳輸量，強烈建議您將資料分割為多個檔案來善加利用平行處理。

分析

Advisor 分析可識別載入 Amazon S3 中暫存之少量檔案中包含的大型資料集的COPY命令。從幾個檔案載入大型資料集的長時間執行COPY命令，通常有機會大幅改善效能。當 Advisor 識別出這些COPY命令需要大量時間時，會建立建議，藉由將資料分割成 Amazon S3 中的其他檔案，以提高平行處理原則。

建議

在此情況下，建議採取以下依照優先順序排序的動作：

1. 最佳化載入檔案少於叢集節點數目的COPY指令。
2. 最佳化載入檔案少於叢集磁碟片段數目的COPY命令。
3. 最佳化COPY指令，其中檔案數目不是叢集磁碟片段數目的倍數。

某些COPY命令會載入大量資料或執行相當長的持續時間。對於這些命令，建議您從 Amazon S3 載入資料物件的數量等於叢集中分割數量的倍數。若要識別每個COPY命令已載入多少 S3 物件，請以超級使用者身分執行下列SQL程式碼。

```
SELECT
    query, COUNT(*) num_files,
    ROUND(MAX(wq.total_exec_time/1000000.0),2) execution_secs,
    ROUND(SUM(transfer_size)/(1024.0*1024.0),2) total_mb,
    SUBSTRING(querytxt,1,60) copy_sql
FROM stl_s3client s
JOIN stl_query q USING (query)
JOIN stl_wlm_query wq USING (query)
WHERE s.userid>1 AND http_method = 'GET'
      AND POSITION('COPY ANALYZE' IN querytxt) = 0
      AND aborted = 0 AND final_state='Completed'
GROUP BY query, querytxt
HAVING (SUM(transfer_size)/(1024*1024))/COUNT(*) >= 2
ORDER BY CASE
    WHEN COUNT(*) < (SELECT max(node)+1 FROM stv_slices) THEN 1
    WHEN COUNT(*) < (SELECT COUNT(*) FROM stv_slices WHERE node=0) THEN 2
    ELSE 2+((COUNT(*) % (SELECT COUNT(*) FROM stv_slices))/(SELECT COUNT(*)::DECIMAL FROM
    stv_slices))
END, (SUM(transfer_size)/(1024.0*1024.0))/COUNT(*) DESC;
```

實作提示

- 節點中的分割數目取決於叢集的節點大小。如需各個節點類型有多少分割的相關資訊，請參閱《Amazon Redshift 管理指南》中的 < [Amazon Redshift 中的叢集和節點](#) >。
- 您可以透過為集合指定通用字首或字首索引鍵，或在資訊清單檔案中明確列出檔案來載入多個檔案。如需載入檔案的相關資訊，請參閱 [從已壓縮和未壓縮的檔案載入資料](#)。

- Amazon Redshift 在分割工作負載時，不會考量檔案大小。分割您的載入資料，使檔案皆有相同的大小，在壓縮之後介於 1 MB 至 1 GB。

更新資料表統計資訊

Amazon Redshift 使用成本型查詢最佳化工具來為查詢選擇最佳執行計畫。成本估算是根據使用 ANALYZE 指令收集的資料表統計資料。當統計資訊已過時或遺失時，資料庫可能會選擇效率較低的查詢執行計畫，特別是複雜的查詢。維護最新統計資訊有助於複雜的查詢以可能的最短時間執行。

分析

「建議程式」分析會追蹤統計資料為 out-of-date 或遺失的表格。它會檢閱與複雜查詢相關聯的資料表存取中繼資料。如果經常使用複雜模式存取的表格遺漏統計資料，「建議程式」會建立要執行的重要建議 ANALYZE。如果經常使用複雜模式存取的表格具有 out-of-date 統計資料，「建議程式」會建立建議來執行 ANALYZE。

建議

每當表格內容發生重大變更時，請使用 ANALYZE。我們建議您在使用 COPY 或 INSERT 指令將大量新資料列載入至既有資料表 ANALYZE 時執行。我們也建議您在使用 UPDATE 或 DELETE 指令修改大量列 ANALYZE 時執行。若要識別遺失的資料表或 out-of-date 統計資料，請以超級使用者身分執行下列 SQL 命令。其結果將從最大到最小的資料表進行排序。

若要識別遺失的資料表或 out-of-date 統計資料，請以超級使用者身分執行下列 SQL 命令。其結果將從最大到最小的資料表進行排序。

```
SELECT
    ti.schema||'.'||ti."table" tablename,
    ti.size table_size_mb,
    ti.stats_off statistics_accuracy
FROM svv_table_info ti
WHERE ti.stats_off > 5.00
ORDER BY ti.size DESC;
```

實作提示

預設 ANALYZE 臨界值為 10%。這個預設值表示，如果資料表資料列的資料列自上次變更後 ANALYZE 少於 10%，則該 ANALYZE 命令會略過指定的資料表。因此，您可能會選擇在每個 ETL 程序結束時發出 ANALYZE 指令。採用這種方法意味著通常 ANALYZE 被跳過，但也可以確保在需要時 ANALYZE 運行。

ANALYZE統計資料對於聯結中使用的資料行 (例如, JOIN tbl_a ON col_b) 或作為述詞 (例如) 的資料行的影響最大WHERE col_b = 'xyz'。依預設, ANALYZE會收集指定表格中所有資料行的統計資料。如果需要, 您可以ANALYZE僅針對影響最大ANALYZE的資料行執行, 以減少執行所需的時間。您可以執行下列SQL命令來識別用作述詞的資料行。您也可以指定 ANALYZE PREDICATE COLUMNS, 讓 Amazon Redshift 選擇要分析哪些欄。

```
WITH predicate_column_info as (
SELECT ns.nspname AS schema_name, c.relname AS table_name, a.attnum as col_num,
a.attname as col_name,
CASE
WHEN 10002 = s.stakind1 THEN array_to_string(stavalues1, '||')
WHEN 10002 = s.stakind2 THEN array_to_string(stavalues2, '||')
WHEN 10002 = s.stakind3 THEN array_to_string(stavalues3, '||')
WHEN 10002 = s.stakind4 THEN array_to_string(stavalues4, '||')
ELSE NULL::varchar
END AS pred_ts
FROM pg_statistic s
JOIN pg_class c ON c.oid = s.starelid
JOIN pg_namespace ns ON c.relnamespace = ns.oid
JOIN pg_attribute a ON c.oid = a.attrelid AND a.attnum = s.staattnum)
SELECT schema_name, table_name, col_num, col_name,
pred_ts NOT LIKE '2000-01-01%' AS is_predicate,
CASE WHEN pred_ts NOT LIKE '2000-01-01%' THEN (split_part(pred_ts,
'||',1))::timestamp ELSE NULL::timestamp END as first_predicate_use,
CASE WHEN pred_ts NOT LIKE '%||2000-01-01%' THEN (split_part(pred_ts,
'||',2))::timestamp ELSE NULL::timestamp END as last_analyze
FROM predicate_column_info;
```

如需詳細資訊, 請參閱[分析資料表](#)。

啟用短期查詢加速

短查詢加速 (SQA) 會在較長時間執行的查詢之前, 優先考慮選取的短時間執行查詢。SQA在專用空間中執行短時執行的查詢, 因此不會強制SQA查詢在較長查詢後面的佇列中等候。SQA僅優先處理短時執行且位於使用者定義佇列中的查詢。使用時SQA, 短時間執行的查詢會更快地開始執行, 使用者可以更快地看到

如果您開啟SQA, 您可以減少或排除專用於執行簡短查詢的工作負載管理 (WLM) 佇列。此外, 長時間執行的查詢不需要處理佇列中插槽的簡短查詢, 因此您可以將WLM佇列設定為使用較少的查詢位置。使用較低的並行性時, 就大部分工作負載而言, 查詢傳輸量會增加, 且整體系統效能會改善。如需詳細資訊, 請參閱[短期查詢加速](#)。

分析

Advisor SQA 會检查工作負載模式，並報告最近查詢的數目，以減少延遲和SQA符合資格查詢的每日佇列時間。

建議

修改要開啟的WLM模型組態SQA。Amazon Redshift 採用機器學習演算法來分析每一個合格查詢。從您的查詢模式中SQA學習時，預測也會有所改善。如需詳細資訊，請參閱[設定工作負載管理](#)。

當您開啟時SQA，依預WLM設會將短查詢的最大執行階段設定為動態。我們建議保留動態設定以達到SQA最大執行時間。

實作提示

若要檢查SQA是否已開啟，請執行下列查詢。如果查詢傳回資料列，SQA則會開啟。

```
select * from stv_wlm_service_class_config
where service_class = 14;
```

如需詳細資訊，請參閱[監控 SQA](#)。

更改資料表上的分佈索引鍵

Amazon Redshift 會根據資料表分佈樣式將資料表列分佈至整個叢集。具有KEY分佈的資料表需要欄作為分配索引鍵 (DISTKEY)。資料表資料列會根據叢集的資料行DISTKEY值指派給叢集的節點片段。

適當DISTKEY地在每個節點片段上放置類似數目的列，並且在接合條件中經常被參考。當資料表連結至其DISTKEY資料行時，就會發生最佳化的聯結，以加速查詢效能。

分析

Advisor 會分析叢集的工作負載，以識別最適當的發佈索引鍵，以便從發KEY佈樣式中顯著受益的資料表。

建議

Advisor 提供的[ALTER TABLE](#)陳述式會根據資料表DISTKEY的分析來改變DISTSTYLE和。若要實現顯著的效能優勢，請務必在建議群組中實作所有SQL陳述式。

重新分配ALTERTABLE耗用叢集資源的大型資料表，並在不同時間需要暫時資料表鎖定。因此，請在其他叢集工作負載較輕時實作每個建議群組。如需最佳化表格分佈屬性的詳細資訊，請參閱 [Amazon Redshift 工程的進階資料表設計操作手冊：分佈樣式與分佈索引鍵](#)。

若要取得有關ALTERDISTSTYLE和的更多資訊DISTKEY，請參閱[ALTER TABLE](#)。

Note

如果沒有看到建議，並不一定表示目前的發佈樣式是最合適的。如果沒有足夠的資料或重新分配的預期益處很小，Advisor 不會提供建議。

Advisor 建議適用於特定資料表，且不一定適用於含相同資料欄名稱的資料表。欄位名稱相同的資料表，其各欄特性仍可能不同，除非資料表中的資料也相同。

如果您看到工作所建立或卸除之臨時資料表的建議，請修改處理ETL作業以使用建議ETL程式建議的分配金鑰。

變更資料表上的排序索引鍵

Amazon Redshift 會根據資料表的[排序索引鍵](#)對表格列進行排序。表格列會根據排序鍵欄值進行排序。

透過適當的排序索引鍵排序表格可以加速查詢效能，特別是有範圍限制述詞的查詢，因為需要從磁碟讀取的表格區塊較少。

分析

Advisor 會分析您叢集在數天的工作負載，為您的表格找出能提供助益的排序索引鍵。

建議

Advisor 提供兩組ALTERTABLE陳述式，可根據資料表的分析來變更資料表的排序索引鍵：

- 變更目前沒有排序索引鍵可新增排序索引鍵的資料表的COMPOUND陳述式。
- 將排序索引鍵從排序索引鍵改INTERLEAVED為COMPOUND或不改變排序索引鍵的陳述式

使用複合排序索引鍵可大幅降低維護額外負荷。具有複合排序索引鍵的表格不需要交錯排序所需的昂貴VACUUMREINDEX作業。實際上，就大多數 Amazon Redshift 工作負載而言，複合排序索引鍵比交錯排序索引鍵更有效果。但是，如果資料表很小，則不使用排序索引鍵以避免排序索引鍵儲存空間負荷會更有效率。

使用排序大型資料表時 ALTERTABLE，會耗用叢集資源，並在不同時間需要資料表鎖定。當叢集的工作負載不高時，請實作每個建議。如需最佳化表格排序索引鍵組態的詳細資訊，請參閱 [Amazon Redshift 工程的進階資料表設計操作手冊：複合和交錯排序索引鍵](#)。

如需有關的更多資訊 ALTERSORTKEY，請參閱[ALTER TABLE](#)。

Note

如果您沒有看到表格的建議，這並不一定表示當前配置是最好的。當沒有足夠的資料或排序的預期益處不大時，Advisor 不會提供建議。

Advisor 建議適用於特定表格，且不一定適用於含有相同名稱和資料類型的表格。根據表格中的資料和工作負載，共用欄名稱的表格可能會有不同的建議。

修改資料欄上的壓縮編碼

壓縮是欄層級操作，可降低資料儲存時的大小。Amazon Redshift 中會使用壓縮來節省儲存空間並透過減少磁碟 I/O 量來改善查詢效能。我們建議您根據資料類型和查詢模式為每個資料欄採用最佳壓縮編碼。透過最佳壓縮，查詢可以更有效率地執行，而且資料庫會佔用最少的儲存空間。

分析

Advisor 會持續執行叢集工作負載和資料庫結構描述的分析，以識別每個資料表資料欄的最佳壓縮編碼。

建議

Advisor 會根據特定資料行的分析，提供變更特定資料行壓縮編碼的 ALTER TABLE 陳述式。

以 [ALTER TABLE](#) 變更資料欄壓縮編碼會耗用叢集資源，且需要在不同時間鎖定資料表。建議您在叢集工作負載很輕時實作建議內容。

作為參考，[ALTER TABLE 範例](#) 會顯示數個變更資料欄編碼的陳述式。

Note

當沒有足夠的資料或變更編碼的預期益處不大時，Advisor 不會提供建議。

資料類型建議

Amazon Redshift 具有適用於各種使用案例的 SQL 資料類型庫。這些包括整數類型 (例如 INT) 和儲存字元的類型 (例如 VARCHAR)。Redshift 會以最佳化的方式儲存類型，以提供快速存取和良好的查詢效能。此外，Redshift 還提供用於特定類型的函數，您可以使用這些函數對查詢結果進行格式化或執行計算。

分析

Advisor 會持續執行叢集工作負載和資料庫結構描述的分析，以識別可從資料類型變更中顯著受益的資料欄。

建議

Advisor 提供的 ALTER TABLE 陳述式會新增包含建議資料類型的新資料欄。隨附的 UPDATE 陳述式會將資料從現有資料欄複製到新資料欄。建立新資料欄並載入資料後，請變更查詢和擷取指令碼以存取新資料欄。然後利用專門用於新資料類型的功能和函數 (可在 [SQL 函數參考](#) 中找到)。

將現有資料複製到新資料欄可能需要一些時間。建議您在叢集的工作負載較輕時，實作每個建議程式建議。在 [資料類型](#) 中參考可用資料類型的清單。

請注意，當沒有足夠的資料或變更資料類型的預期益處不大時，Advisor 不會提供建議。

Amazon Redshift 的教學課程

請遵循這些教學課程中的步驟以瞭解 Amazon Redshift 功能：

[教學課程：從 Amazon S3 載入資料](#)

在本教學中，您會逐步完成將資料從 S3 儲存貯體中的資料檔案從頭到尾載入 Amazon Redshift 資料庫表格的程序。

[教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料](#)

在本自學課程中，您會使用 Redshift 光譜來查詢巢狀資料。Redshift 頻譜可讓您查詢實木複合地板、ORCJSON、或 Ion 檔案格式的資料。

[教學課程：設定手動工作負載管理 \(WLM\) 佇列](#)

在本教學中，您將 Amazon Redshift 設定為使用手動工作負載管理 (WLM) 佇列。Amazon Redshift 使用 WLM 佇列來管理資源分割方式以執行並行查詢。如果您需要使用多個 WLM 佇列，WLM 則必須將 Amazon Redshift 設定為使用手動。

[教學課程：搭配 Amazon Redshift 使用空間 SQL 函數](#)

在本自學課程中，您將使用空間函數查詢資料。您可以使用空間函數來查詢幾何圖形和地理資料。

[Amazon Redshift ML 的教學課程](#)

在這些自學課程中，您會建立和使用機器學習模型。

[教學課程：使用 建立角色和查詢 RBAC](#)

在本教學課程中，您會使用以角色為基礎的存取控制 (RBAC) 在您建立的資料庫中建立和使用權限。使用 RBAC，您可以建立具有特定權限的角色，例如唯讀或讀寫權限。然後，您可以將這些角色指派給使用者，以授與他們指定的權限。

自動資料表最佳化

自動資料表最佳化是一項自我調整功能，可透過套用排序和分佈索引鍵來自動最佳化資料表的設計，而無須管理員介入。透過使用自動化來調整資料表的設計，您可以快速開始並獲得最快的效能，而無需花費時間來手動調整和實作資料表最佳化。

自動資料表最佳化會持續觀察查詢與資料表的互動方式。其使用進階人工智慧方法來選擇排序和分佈索引鍵，以最佳化叢集工作負載的效能。如果 Amazon Redshift 判定套用索引鍵可改善叢集效能，則資料表會在叢集建立後的數小時內自動進行修改，而且會盡可能降低對查詢造成的影響。

為了充分利用此自動化功能，Amazon Redshift 管理員會建立新資料表，或修改現有資料表以使其能夠使用自動最佳化。分佈樣式或排序索引鍵為 AUTO 的現有資料表已啟用自動化功能。當您對這些資料表執行查詢時，Amazon Redshift 會判斷排序索引鍵或分佈索引鍵是否能改善效能。如果是的話，Amazon Redshift 就會自動修改資料表，而不需要管理員介入。如果執行最低限度的查詢數目，則會在啟動叢集後的數小時內套用最佳化。

如果 Amazon Redshift 判斷分佈索引鍵可改善查詢的效能，則分佈樣式為 AUTO 的資料表可以將其分佈樣式變更為 KEY。

主題

- [啟用、停用和監控自動資料表最佳化](#)
- [資料欄壓縮以減少儲存資料的大小](#)
- [查詢最佳化的資料分佈](#)
- [排序金鑰](#)
- [資料表限制](#)

啟用、停用和監控自動資料表最佳化

根據預設，未明確定義排序索引鍵或分佈索引鍵而建立的資料表會設定為 AUTO。在建立資料表時，您也可以手動明確設定排序或分佈索引鍵。如果您設定排序或分佈索引鍵，則資料表不會自動受到管理。

啟用自動資料表最佳化

若要自動最佳化現有資料表，請使用 ALTER 陳述式選項將資料表變更為 AUTO。您可以選擇為排序索引鍵定義自動化，但不能定義分佈索引鍵的自動化 (反之亦然)。如果您執行 ALTER 陳述式將資料表轉換為自動資料表，則會保留現有的排序索引鍵和分佈樣式。

```
ALTER TABLE table_name ALTER SORTKEY AUTO;
```

```
ALTER TABLE table_name ALTER DISTSTYLE AUTO;
```

如需詳細資訊，請參閱[ALTER TABLE](#)。

一開始，資料表沒有分佈索引鍵或排序索引鍵。分佈樣式會設定為 EVEN 或 ALL，取決於資料表大小。隨著資料表的大小不斷增加，Amazon Redshift 會套用最佳分佈索引鍵和排序索引鍵。最佳化會在執行最少查詢數目後的小時內套用。在決定排序索引鍵最佳化時，Amazon Redshift 會嘗試最佳化資料表掃描期間從磁碟讀取的資料區塊。在決定分佈樣式最佳化時，Amazon Redshift 會嘗試最佳化叢集節點之間傳輸的位元組數。

從資料表中移除自動資料表最佳化

您可以從自動最佳化中移除資料表。從自動化中移除資料表需要選取排序索引鍵或分佈樣式。若要變更分佈樣式，請指定特定的分佈樣式。

```
ALTER TABLE table_name ALTER DISTSTYLE EVEN;
```

```
ALTER TABLE table_name ALTER DISTSTYLE ALL;
```

```
ALTER TABLE table_name ALTER DISTSTYLE KEY DISTKEY c1;
```

若要變更排序索引鍵，您可以定義排序索引鍵或選擇「無」。

```
ALTER TABLE table_name ALTER SORTKEY(c1, c2);
```

```
ALTER TABLE table_name ALTER SORTKEY NONE;
```

監控自動資料表最佳化

系統檢視 SVV_ALTER_TABLE_RECOMMENDATIONS 會記錄資料表目前的 Amazon Redshift Advisor 建議。此檢視會顯示所有資料表的建議，包括已針對或未針對自動最佳化定義的資料表。

若要檢視資料表是否已定義為自動最佳化，請查詢系統檢視 SVV_TABLE_INFO。項目只會針對目前工作階段資料庫中可見的資料表而顯示。建議會在叢集建立後的幾小時內開始插入檢視 (每天兩次)。建議可用之後，就會在一小時內開始執行。在套用建議之後 (無論是由 Amazon Redshift 或您執行)，該建議將不再出現在檢視中。

系統檢視 `SVL_AUTO_WORKER_ACTION` 會顯示 Amazon Redshift 採取的所有動作的稽核日誌，以及資料表的先前狀態。

系統檢視 `SVV_TABLE_INFO` 會列出系統中的所有資料表，以及用來指出資料表的排序索引鍵和分佈樣式是否設定為 `AUTO` 的資料欄。

如需這些系統檢視的相關資訊，請參閱 [系統監控 \(僅限已佈建\)](#)。

資料欄壓縮以減少儲存資料的大小

壓縮是欄層級操作，可降低資料儲存時的大小。壓縮可節省儲存空間，並降低從儲存體讀取的資料大小，這樣會減少磁碟 I/O 數量，因而改善查詢效能。

`ENCODE AUTO` 是資料表的預設值。當資料表設定為 `ENCODE` 時 `AUTO`，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如需詳細資訊，請參閱 [CREATE TABLE](#) 和 [ALTER TABLE](#)。

不過，如果您為資料表中的任何資料欄指定壓縮編碼，則資料表將不再設定為 `ENCODE AUTO`。Amazon Redshift 不再自動管理資料表中所有資料欄的壓縮編碼。

建立資料表時，您可以手動將壓縮類型或編碼套用至資料表中的資料欄。或者，您可以使用 `COPY` 命令自動分析和套用壓縮。如需詳細資訊，請參閱 [讓我們 COPY 選擇壓縮編碼](#)。如要套用自動壓縮的詳細資訊，請參閱 [利用自動壓縮載入資料表](#)。

Note

我們強烈建議使用 `COPY` 命令來套用自動壓縮。

如果新資料表與另一個資料表共用相同的資料特性，您可以選擇手動套用壓縮編碼。或者，如果您在測試過程中發現自動壓縮過程中套用的壓縮編碼不適合您的資料，則可能會這樣做。如果您選擇手動套用壓縮編碼，則可以針對已填入的資料表執行 [ANALYZE COMPRESSION](#) 命令，並使用結果來選擇壓縮編碼。

若要手動套用壓縮，您可以為個別資料欄指定壓縮編碼，作為 `CREATE TABLE` 陳述式的一部分。語法如下。

```
CREATE TABLE table_name (column_name  
data_type ENCODE encoding-type)[, ...]
```

其中 `encoding-type` 是取自於下節中的關鍵字資料表。

例如，下列陳述式會建立雙欄資料表 PRODUCT。將資料載入資料表時，PRODUCT_ID 資料欄不會壓縮，但會壓縮 PRODUCT_NAME 資料欄，並使用位元組字典編碼（BYTEDICT）。

```
create table product(  
product_id int encode raw,  
product_name char(20) encode bytedict);
```

您可以使用 ALTERTABLE 命令將資料欄新增至資料表時，指定資料欄的編碼。

```
ALTER TABLE table-name ADD [ COLUMN ] column_name column_type ENCODE encoding-type
```

主題

- [壓縮編碼](#)
- [測試壓縮編碼](#)

壓縮編碼

壓縮編碼會指定在資料列新增至資料表時，套用至一欄資料值的壓縮類型。

ENCODE AUTO 是資料表的預設值。當資料表設定為 ENCODE 時 AUTO，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如需詳細資訊，請參閱 [CREATE TABLE](#) 和 [ALTER TABLE](#)。

不過，如果您為資料表中的任何資料欄指定壓縮編碼，則資料表將不再設定為 ENCODE AUTO。Amazon Redshift 不再自動管理資料表中所有資料欄的壓縮編碼。

當您使用 CREATE 時 TABLE，當您為資料表中的任何資料欄指定壓縮編碼時，ENCODE AUTO 會停用。如果 ENCODE AUTO 已停用，Amazon Redshift 會自動將壓縮編碼指派給您未指定 ENCODE 類型的資料欄，如下所示：

- 定義為排序索引鍵的資料欄會被指派 RAW 壓縮。
- 定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的資料欄會被指派 RAW 壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMALDATE、TIMESTAMP 或 TIMESTAMPTZ 資料類型的資料欄會被指派 AZ64 壓縮。
- 定義為 CHAR 或 VARCHAR 資料類型的資料欄會被指派 LZ0 壓縮。

您可以使用 ALTERTABLE 來變更資料表的編碼。如果您 ENCODE AUTO 使用 ALTER 停用 TABLE，Amazon Redshift 將不再自動管理資料欄的壓縮編碼。在您變更壓縮編碼類型或

ENCODEAUTO再次啟用ENCODEAUTO之前，所有資料欄都會保留其在停用時擁有的壓縮編碼類型。

Amazon Redshift 支援下列壓縮編碼：

Raw

原始編碼是指定為排序索引鍵的資料欄的預設編碼REAL，以及定義為 BOOLEAN、或 DOUBLEPRECISION資料類型的資料欄。系統使用 Raw 編碼，以原始、未壓縮格式來儲存資料。

AZ64

AZ64 是由 Amazon 設計的專有壓縮編碼演算法，可實現高壓縮率並改善查詢處理。該AZ64演算法的核心會壓縮較小的資料值群組，並使用單一指令、多個資料（SIMD）指令進行平行處理。使用 AZ64 可大幅節省儲存成本，並針對數值、日期和時間資料類型提供高效能。

您可以使用 AZ64作為壓縮編碼，使用 CREATETABLE和 ALTERTABLE陳述式定義具有下列資料類型的資料欄：

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- DATE
- TIMESTAMP
- TIMESTAMPTZ

Byte-dictionary


在 byte dictionary 編碼中，唯一值的個別字典是針對磁碟上資料欄值的每個區塊而建立的。(Amazon Redshift 磁碟會佔用 1 MB。)字典最多可包含 256 個一元位元組值，其會以索引形式儲存至原始資料值。如果在單一區塊中儲存超過 256 個值，則額外的值會以原始、未壓縮格式寫入至區塊。對於每個磁碟區塊都會重複此程序。

這種編碼對低基數字串資料欄非常有效。當資料欄的資料網域少於 256 個唯一值時，這是最佳的編碼。

對於使用編碼的字串資料類型（CHAR 和 VARCHAR）的資料欄BYTEDICT，Amazon Redshift 會執行直接透過壓縮資料操作的向量化掃描和述詞評估。這些掃描使用硬體特定的單一指令和多個

資料 (SIMD) 指令進行平行處理。這會顯著加快字串資料欄的掃描速度。如果 CHAR/VARCHAR 資料欄包含長字元字串，則位元組字典編碼會特別節省空間。

假設資料表具有具有 CHAR (30) 資料類型的資料 COUNTRY 欄。載入資料時，Amazon Redshift 會建立字典，並將索引值填入資料 COUNTRY 欄。字典包含已建立索引的唯一值，而且資料表本身只包含對應值的一位元組下標。

 Note

會儲存固定長度字元欄的多餘空格。因此，在 CHAR (30) 欄中，當您使用位元組字典編碼時，每個壓縮值都會節省 29 個位元組的儲存。

下表代表 COUNTRY 資料欄的字典。

唯一資料值	字典索引	大小 (固定長度，每個值 30 個位元組)
England	0	30
United States of America	1	30
Venezuela	2	30
Sri Lanka	3	30
Argentina	4	30
Japan	5	30
合計		180

下表代表 COUNTRY 欄中的值。

原始資料值	原始大小 (固定長度，每個值 30 個位元組)	壓縮值 (索引)	新大小 (位元組)
England	30	0	1

原始資料值	原始大小 (固定長度，每個值 30 個位元組)	壓縮值 (索引)	新大小 (位元組)
England	30	0	1
United States of America	30	1	1
United States of America	30	1	1
Venezuela	30	2	1
Sri Lanka	30	3	1
Argentina	30	4	1
Japan	30	5	1
Sri Lanka	30	3	1
Argentina	30	4	1
合計	300		10

此範例中的合計壓縮大小計算如下：6 個不同項目儲存在字典 ($6 * 30 = 180$)，而且資料表包含 10 個 1 位元組壓縮值，合計 190 個位元組。

Delta

Delta 編碼對日期時間欄很有用。

Delta 藉由記錄資料欄中彼此跟隨的値之間的差異來壓縮資料。此差異會記錄在磁碟上資料欄値之每個區塊的個別字典中。(Amazon Redshift 磁碟會佔用 1 MB。) 例如，假設資料欄包含 10 個整數，順序從 1 到 10。第一個會儲存為 4 位元組整數 (加上 1 位元組旗標)。接下來的九個會分別儲存為値為 1 的位元組，表示比前一個値多 1。

Delta 編碼附有兩個變異：

- DELTA 會將差異記錄為 1 位元組值 (8 位元整數)

- DELTA32K 將差異記錄為 2 位元組值 (16 位元整數)

如果資料欄中的大多數值可以透過使用單一位元組進行壓縮，則 1 位元組的變化非常有效。但是，如果差異較大，則此編碼 (在最壞的情況下) 會比儲存未壓縮資料的效果差。類似邏輯適用於 16 位元版本。

如果兩個值之間的差異超過 1 位元組範圍 (DELTA) 或 2 位元組範圍 (DELTA32K)，則會儲存完整的原始值，並帶有 1 位元組開頭的旗標。1 位元組範圍從 -127 到 127，而 2 位元組範圍從 -32K 到 32K。

下表顯示數值欄的 Delta 編碼如何運作。

原始資料值	原始大小 (位元組)	差異 (delta)	壓縮值	壓縮大小 (位元組)
1	4		1	1+4 (旗標 + 實際值)
5	4	4	4	1
50	4	45	45	1
200	4	150	150	1+4 (旗標 + 實際值)
185	4	-15	-15	1
220	4	35	35	1
221	4	1	1	1
合計	28			15

LZO

LZO 編碼提供非常高的壓縮比率，並具有良好的效能。LZO 編碼對存放非常長字元字串的 CHAR 和 VARCHAR 資料欄尤其有效。它們特別適用於自由格式文字，例如產品描述、使用者評論或 JSON 字串。

Mostly

當資料欄的資料類型大於大部份儲存值所需的資料類型時，Mostly 編碼很有用。您可以對此類型的資料欄指定 mostly 編碼，將資料欄中的大部分值壓縮為更小的標準儲存空間大小。剩下無法壓縮的值會以其原始格式儲存。例如，您可以將 16 位元資料欄，例如資料INT2欄，壓縮為 8 位元儲存。

通常，mostly 編碼會使用下列資料類型：

- SMALLINT/INT2 (16 位元)
- INTEGER/INT (32 位元)
- BIGINT/INT8 (64 位元)
- DECIMAL/NUMERIC (64 位元)

選擇 mostly 編碼的適當變異來符合資料欄之資料類型的大小。例如，MOSTLY8套用至定義為 16 位元整數資料欄的資料欄。不允許套用至 16 位元資料類型MOSTLY16的資料欄，或MOSTLY32套用至 32 位元資料類型的資料欄。

當資料欄中有相當多的值無法壓縮時，Mostly 編碼可能比無壓縮更沒效率。在將其中一種編碼套用到資料欄之前，請執行檢查。您現在將載入的大部分值 (也可能在未來載入) 應納入下表中顯示的範圍。

編碼	壓縮儲存空間大小	可以壓縮的值範圍 (範圍外的值會以原始形式儲存)
MOSTLY8	1 位元組 (8 位元)	-128 到 127
MOSTLY16	2 位元組 (16 位元)	-32768 到 32767
MOSTLY32	4 位元組 (32 位元)	-2147483648 到 +2147483647

Note

對於小數，忽略小數點以判斷值是否納入範圍。例如，1, 234.56 會被視為 123, 456，並且可以在資料MOSTLY32欄中壓縮。

例如，VENUE資料表中的VENUEID資料欄定義為原始整數資料欄，這表示其值會耗用 4 個位元組的儲存空間。不過，此欄的值目前範圍是 0 到 309。因此，使用 MOSTLY16 編碼重新建立和重新載入此資料表VENUEID會將該資料欄中每個值的儲存體減少為 2 個位元組。

如果另一個資料表中參考VENUEID的值大多介於 0 到 127 的範圍內，則將該外鍵資料欄編碼為可能合理MOSTLY8。在做出選擇之前，請針對參考資料表資料執行數個查詢，以了解值是否大部分都落入 8 位元、16 位元或 32 位元範圍。

下表顯示使用 MOSTLY8、MOSTLY16和 MOSTLY32編碼時，特定數值的壓縮大小：

原始值	原始INT或BIG INT大小 (位元組)	MOSTLY8 壓縮大小 (位元組)	MOSTLY16 壓縮大小 (位元組)	MOSTLY32 壓縮大小 (位元組)
1	4	1	2	4
10	4	1	2	4
100	4	1	2	4
1000	4	與原始資料大小相同	2	4
10000	4		2	4
20000	4		2	4
40000	8	與原始資料大小相同	與原始資料大小相同	4
100000	8			4
2000000000	8			4

Run length

執行長度編碼會將連續重複的值取代為由值和連續出現次數 (執行長度) 組成的符記。唯一值的個別字典是針對磁碟上資料欄值的每個區塊而建立的。(Amazon Redshift 磁碟會佔用 1 MB。) 此編碼最適合於資料值通常連續重複的資料表，例如，依那些值排序資料表時。

例如，假設大維度資料表中的資料欄具有可預測的小網域，例如可能值少於 10 的資料COLOR欄。這些值可能會落在整個資料表中的長序列中，即使資料未排序也是如此。

不建議在指定為排序索引鍵的任何資料欄上套用執行長度編碼。當區塊包含類似的列數時，限制範圍的掃描執行效果會更好。如果排序索引鍵欄位的壓縮比相同查詢中的其他欄位高出許多，限制範圍的掃描執行效果可能較差。

下表使用COLOR資料欄範例來顯示執行長度編碼的運作方式。

原始資料值	原始大小 (位元組)	壓縮值 (符記)	壓縮大小 (位元組)
Blue	4	{2,藍色}	5
Blue	4		0
Green	5	{3,綠色}	6
Green	5		0
Green	5		0
Blue	4	{1,Blue}	5
Yellow	6	{4,Yellow}	7
Yellow	6		0
Yellow	6		0
Yellow	6		0
合計	51		23

Text255 and Text32k

Text255 和 text32k 編碼有助於壓縮經常重複出現相同字詞的資料VARCHAR欄。唯一單字的個別字典是針對磁碟上資料欄值的每個區塊而建立的。(Amazon Redshift 磁碟會佔用 1 MB。)字典包含資料欄中前 245 個唯一單字。那些單字會在磁碟上被代表 245 個值之一的一位元組索引值所取代，而且未在字典中表示的任何單字會以未壓縮形式儲存。對於每個 1 MB 磁碟區塊都會重複此程序。如果建立索引的單字經常出現在資料欄中，則資料欄將產生高壓縮率。

對於 text32k 編碼，原則相同，但每個區塊的字典不會擷取特定數目的單字。反之，字典會為其找到的每個唯一單字建立索引，直到結合的項目達到 32K 長度，減去一些額外負荷。索引值是以兩個位元組儲存。

例如，請考慮VENUE資料表中的資料VENUENAME欄。**Arena**、**Center** 和 **Theatre** 等單字反覆出現在這個欄中，而且若套用 text255 壓縮，則很可能出現在每個區塊中的前 245 個單字之中。如果是這樣，則此資料欄將受益於壓縮。因為每次那些單字出現，它們將只佔用 1 個位元組的儲存空間(而不是分別佔用 5、6 或 7 個位元組)。

ZSTD

Zstandard (ZSTD) 編碼提供高壓縮率，在不同資料集之間具有非常良好的效能。ZSTD 特別適用於存放各種長短字串的 CHAR 和 VARCHAR資料欄，例如產品描述、使用者備註、日誌和JSON字串。其中 Delta 編碼或大部分編碼等某些演算法可能使用比沒有壓縮更多的儲存空間，ZSTD不太可能增加磁碟用量。

ZSTD 支援

SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLEPRECISION、BOOLEAN、CHAR、VAR和TIMESTAMPTZ 資料類型。

下表識別支援的壓縮編碼和支援編碼的資料類型。

編碼類型	CREATE TABLE 和 中的關鍵字 ALTER TABLE	資料類型
Raw (無壓縮)	RAW	全部
AZ64	AZ64	SMALLINT, INTEGER, BIGINT, DECIMAL, DATE, TIMESTAMP, TIMESTAMPTZ
Byte dictionary	BYTEDICT	SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE PRECISION, CHAR, VARCHAR, DATE, TIMESTAMP, TIMESTAMPTZ
Delta	DELTA	SMALLINT, INT, BIGINT, DATE, TIMESTAMP, DECIMAL

編碼類型	CREATE TABLE 和 中的關鍵字 ALTER TABLE	資料類型
	DELTA32K	INT, BIGINT, DATE, TIMESTAMP, DECIMAL
LZO	LZO	SMALLINT, INTEGER, BIGINT, DECIMAL, CHAR, VARCHAR, DATE, TIMESTAMP, TIMESTAMPTZ, SUPER
Mostlyn	MOSTLY8	SMALLINT, INT, BIGINT, DECIMAL
	MOSTLY16	INT, BIGINT, DECIMAL
	MOSTLY32	BIGINT, DECIMAL
Run-length	RUNLENGTH	SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, TIMESTAMP, TIMESTAMPTZ
文字	TEXT255	VARCHAR 僅限
	TEXT32K	VARCHAR 僅限
Zstandard	ZSTD	SMALLINT, INTEGER, BIGINT, DECIMAL, REAL, DOUBLE PRECISION, BOOLEAN, CHAR, VARCHAR, DATE, TIMESTAMP, TIMESTAMPTZ, SUPER

測試壓縮編碼

如果您決定手動指定資料欄編碼，可能想要使用您的資料測試不同編碼。

Note

我們建議您盡可能使用 COPY 命令載入資料，並允許 COPY 命令根據您的資料選擇最佳編碼。或者，您可以使用 [ANALYZE COMPRESSION](#) 命令，來檢視對現有資料建議的編碼。如要套用自動壓縮的詳細資訊，請參閱[利用自動壓縮載入資料表](#)。

若要執行有意義的資料壓縮測試，您必須有大量資料列。在此範例中，我們使用從兩個資料表中選取的陳述式來建立資料表並插入資料列；VENUE 以及 LISTING。我們會捨棄通常會聯結兩個資料表的 WHERE 子句。結果是 VENUE 資料表中的每個資料列都會聯結到 LISTING 資料表中的所有資料列，總計超過 3,200 萬資料列。這稱為笛卡爾連接，通常不建議使用。但是，為此，這是建立許多資料列的便捷方法。如果您的現有資料表具有您要測試的資料，則可略過此步驟。

在我們有一個包含範例資料的資料表後，我們會建立一個包含七個資料欄的資料表。每個資料欄都有不同的壓縮編碼：raw、bytedict、lzo、run length、text255、text32k 和 zstd。我們會執行從第一個資料表中選取資料的 INSERT 命令，將完全相同的資料填入每個資料欄。

若要測試壓縮編碼，請執行以下動作：

1. (選用) 首先，使用笛卡爾連接來建立一個具有大量資料列的資料表。如果您想要測試現有資料表，請略過此步驟。

```
create table cartesian_venue(  
venueid smallint not null distkey sortkey,  
venueid varchar(100),  
venuecity varchar(30),  
venuestate char(2),  
venuestate integer);  
  
insert into cartesian_venue  
select venueid, venueid, venuecity, venuestate, venuestate  
from venue, listing;
```

2. 接著，以您想要比較的編碼建立資料表。

```
create table encodingvenue (  
venueraw varchar(100) encode raw,  
venuebytedict varchar(100) encode bytedict,  
venuelzo varchar(100) encode lzo,  
venuerunlength varchar(100) encode runlength,  
venuetext255 varchar(100) encode text255,
```

```
venuetext32k varchar(100) encode text32k,
venuezstd varchar(100) encode zstd);
```

3. 使用具有SELECT子句的INSERT陳述式，將相同的資料插入所有資料欄。

```
insert into encodingvenue
select venuename as venueraw, venuename as venuebytedict, venuename as venuelzo,
venuename as venuerunlength, venuename as venuetext32k, venuename as venuetext255,
venuename as venuezstd
from cartesian_venue;
```

4. 驗證新資料表中的資料列數量。

```
select count(*) from encodingvenue

count
-----
38884394
(1 row)
```

5. 查詢 [STV_BLOCKLIST](#) 系統資料表，以比較每個資料欄使用的 1 MB 磁碟區塊數。

MAX 彙總函數會傳回每欄的最高區塊數。STV_BLOCKLIST 資料表包含三個系統產生的資料欄的詳細資訊。此範例在 WHERE子句 `col < 6` 中使用 排除系統產生的資料欄。

```
select col, max(blocknum)
from stv_blocklist b, stv_tbl_perm p
where (b.tbl=p.id) and name = 'encodingvenue'
and col < 7
group by name, col
order by col;
```

此查詢會傳回下列結果。資料欄是從零開始編號。根據您設定叢集的方式，您的結果可能具有不同號碼，但相對大小應該類似。您可以看到，第二欄的BYTEDICT編碼產生了此資料集的最佳結果。這種方法具有比 20:1 更好的壓縮比。LZO 和 ZSTD 編碼也會產生卓越的結果。當然，不同資料集會產生不同結果。當資料欄包含較長的文字字串時，LZO通常會產生最佳壓縮結果。

```
col | max
-----+-----
0 | 203
1 | 10
2 | 22
```

```

3 | 204
4 | 56
5 | 72
6 | 20
(7 rows)

```

如果您有現有資料表中的資料，則您可以使用 [ANALYZE COMPRESSION](#) 命令，來檢視對資料表建議的編碼。例如，下列範例顯示VENUE資料表 CARTESIAN_ 的建議編碼VENUE，其中包含 3,800 萬列。ANALYZE COMPRESSION 建議為資料VENUENAME欄LZO編碼的通知。ANALYZE COMPRESSION 根據多個因素選擇最佳壓縮，其中包括減少百分比。在此特定情況下，BYTEDICT可提供更好的壓縮，但LZO也會產生超過 90% 的壓縮。

```
analyze compression cartesian_venue;
```

Table	Column	Encoding	Est_reduction_pct
reallybigvenue	venueid	lzo	97.54
reallybigvenue	venuename	lzo	91.71
reallybigvenue	venuecity	lzo	96.01
reallybigvenue	venuestate	lzo	97.68
reallybigvenue	venueseats	lzo	98.21

範例

下列範例會建立資料表，CUSTOMER其中包含具有各種資料類型的資料欄。此CREATETABLE陳述式顯示這些資料欄的許多可能的壓縮編碼組合之一。

```

create table customer(
  custkey int encode delta,
  custname varchar(30) encode raw,
  gender varchar(7) encode text255,
  address varchar(200) encode text255,
  city varchar(30) encode text255,
  state char(2) encode raw,
  zipcode char(5) encode bytedict,
  start_date date encode delta32k);

```

下表顯示為CUSTOMER資料表選取的資料欄編碼，並提供選項的說明：

資料行	資料類型	編碼	說明
CUSTKEY	int	delta	CUSTKEY 由唯一的連續整數值組成。由於差異是 1 位元組，DELTA 因此是不錯的選擇。
CUSTNAME	varchar(30)	raw	CUSTNAME 有一個具有少數重複值的大型網域。任何壓縮編碼通常可能成效不佳。
GENDER	varchar(7)	text255	GENDER 是具有許多重複值的極小網域。Text255 與重複出現相同字詞的資料 VARCHAR 欄運作良好。
ADDRESS	varchar(200)	text255	ADDRESS 是一個大型網域，但包含許多重複的單字，例如街道、大道、北部、南部等。文字 255 和文字 32k 有助於壓縮重複出現相同字詞的資料 VARCHAR 欄。資料欄長度很短，因此 text255 是個好選擇。
CITY	varchar(30)	text255	CITY 是一個大型網域，具有一些重複值。特定城市名稱比其他城市名稱更常使用。Text255 是不錯

資料行	資料類型	編碼	說明
			的選擇，原因與相同 ADDRESS。
STATE	char(2)	raw	在美國，STATE 是 50 個雙字元值的精確網域。Bytedict 編碼將產生一些壓縮，但是因為資料欄大小只是兩個字元，所以壓縮可能不值得解除壓縮資料的額外負荷。
ZIPCODE	char(5)	bytedict	ZIPCODE 是少於 50,000 個唯一值的已知網域。特定郵遞區號比其他郵遞區號更常出現。當資料欄包含有限數目的唯一值時，Bytedict 編碼很有效。
START_DATE	date	delta32k	Delta 編碼對日期時間資料欄很有用，尤其是在資料列依日期順序載入時。

查詢最佳化的資料分佈

當您將資料載入資料表時，Amazon Redshift 會根據資料表的分佈樣式，將資料表的資料列分佈至每一個運算節點。當您執行查詢時，查詢最佳化工具會視需要將資料列重新配送至運算節點，以執行任何聯結與彙總。選擇資料表配送樣式的目的是，藉由在執行查詢之前，將資料配置至必要的位置，以降低重新配送步驟所帶來的影響。

Note

本節將向您介紹 Amazon Redshift 資料庫中資料分佈的原則。建議您使用 `DISTSTYLE AUTO` 建立資料表。如果您這樣做，Amazon Redshift 會使用自動資料表最佳化來選擇資料分發樣式。如需詳細資訊，請參閱[自動資料表最佳化](#)。本節的其餘部分提供有關分佈樣式的詳細資料。

主題

- [資料分佈概念](#)
- [分佈樣式](#)
- [檢視分佈樣式](#)
- [評估查詢模式](#)
- [指定分佈樣式](#)
- [評估查詢計畫](#)
- [查詢計劃範例](#)
- [分佈範例](#)

資料分佈概念

Amazon Redshift 的一些資料分佈概念如下。

節點和配量

Amazon Redshift 叢集是一組節點。叢集中的每個節點都有自己的作業系統、專用的記憶體，以及專用的磁碟儲存空間。一個節點是領導者節點，其會管理運算節點的資料分佈和查詢處理任務。運算節點提供執行這些工作的資源。

運算節點的磁碟儲存空間會分成一些配量。每一節點的配量數目取決於叢集的節點大小。節點全部會參與在執行的平行查詢中，以處理盡可能平均地分佈於配量的資料。如需每個節點大小有多少配量的相關資訊，請參閱《Amazon Redshift 管理指南》中的[關於叢集和節點](#)。

資料重新分佈

當您將資料載入資料表時，Amazon Redshift 會根據資料表的分佈樣式，將資料表的資料列分佈至每一個節點配量。做為查詢計畫的一部分，最佳化器會決定資料區塊需要位於何處，才能最佳地執行查詢。

然後，系統會在查詢執行時實際移動或重新分配資料。重新分佈可能需要將特定資料列傳送至節點，以將整個資料表聯結或廣播至所有節點。

資料重新分佈可以佔查詢計劃成本的很大一部分，並且它產生的網絡流量可能會影響其他資料庫操作，並降低整體系統效能。在某種程度上，您可以預期最初定位資料的最佳位置，將資料重新分佈的影響降至最低。

資料分佈目標

當您將資料載入資料表時，Amazon Redshift 會根據您建立資料表時選擇的分佈樣式，將資料表的資料列分佈至運算節點及配量。資料分佈具有兩個主要目標：

- 在叢集的節點之間平均分配工作負載。分佈不平均 (或資料分佈扭曲) 會迫使某些節點比其他節點更忙碌工作，導致查詢效能下降。
- 為了盡量減少查詢執行時的資料移動。如果參與聯結或彙總的資料列已在其聯結資料列位於其他資料表的叢集上共置，則最佳化器不需要在查詢執行期間重新分佈儘可能多的資料。

您為資料庫選擇的分佈策略對查詢效能、儲存空間需求、資料載入和維護具有重要影響。藉由為每個資料表選擇最好的分佈樣式，您可以平衡資料分佈，並大幅改善整體系統效能。

分佈樣式

建立資料表時，您可以指定下列其中一種分佈樣式：AUTO、KEY、EVEN或 ALL。

如果您未指定分佈樣式，Amazon Redshift 會使用AUTO分佈。

AUTO 分佈

透過AUTO分佈，Amazon Redshift 會根據資料表資料的大小來指派最佳分佈樣式。例如，如果指定AUTO分佈樣式，Amazon Redshift 最初會將ALL分佈樣式指派給小型資料表。當資料表變大時，Amazon Redshift 可能會將分佈樣式變更為 KEY，選擇主索引鍵 (或複合主索引鍵的一欄) 作為分佈索引鍵。如果資料表變大，而且任何資料欄都不適合成為分發金鑰，Amazon Redshift 會將分發樣式變更為 EVEN。分佈樣式的變更會發生在背景中，對使用者查詢的影響最小。

若要檢視 Amazon Redshift 自動執行的動作來修改資料表分佈索引鍵，請參閱 [SVL_AUTO_WORKER_ACTION](#)。若要檢視有關修改資料表分佈索引鍵的目前建議，請參閱 [SVV_ALTER_TABLE_RECOMMENDATIONS](#)。

若要檢視套用至資料表的分佈樣式，請查詢 PG_CLASS_INFO系統目錄檢視。如需詳細資訊，請參閱[檢視分佈樣式](#)。如果您沒有使用 CREATETABLE陳述式指定分佈樣式，Amazon Redshift 會套用 AUTO分佈。

EVEN 分佈

領導者節點會以循環模式在配量之間分佈資料列，而不考慮任何特定資料欄的值。EVEN 當資料表不參與聯結時，分發是適當的。當KEY分佈和ALL分佈之間沒有明確的選擇時，也是適當的。

KEY 分佈

資料列根據一個欄的值來分佈。領導節點會將相符的值放在相同的節點配量。如果您根據聯結索引鍵來分佈一對資料表，領導者節點會根據聯結欄的值將資料列共置在配量。這樣，來自共同資料欄的比對值會實際儲存在一起。

ALL 分佈

整個資料表的副本分佈至每個節點。當EVEN分佈或KEY分佈在每個節點上僅放置資料表資料列的一部分時，ALL分佈可確保每個資料列都與資料表參與的每個聯結集中。

ALL 分佈會乘以叢集中節點數量所需的儲存體，因此載入、更新或插入資料到多個資料表需要更長的時間。ALL 分佈僅適用於相對較慢的移動資料表；也就是說，未經常或廣泛更新的資料表。由於查詢期間重新分發小資料表的成本較低，因此將小維度資料表定義為 DISTSTYLE 並沒有顯著的好處ALL。

Note

指定資料欄的分佈樣式之後，Amazon Redshift 就會在叢集層級上處理資料分佈。Amazon Redshift 不需要或不支援在資料庫物件內分割資料的概念。您不需要建立資料表空間或定義資料表的分割方案。

在某些情況下，您可以在建立資料表的分佈樣式之後，變更該分佈樣式。如需詳細資訊，請參閱 [ALTER TABLE](#)。對於建立資料表分佈樣式之後無法加以變更的情況，您可以重新建立資料表，並以深層複製填入新資料表。如需詳細資訊，請參閱 [執行深層複製](#)

檢視分佈樣式

若要檢視資料表的分佈樣式，請查詢 PG_CLASS_INFO 檢視或 SVV_TABLE_INFO 檢視。

PG_CLASS_INFO 中的資料 RELEFFECTIVEDISTSTYLE 欄表示資料表目前的分佈樣式。如果資料表使用自動分佈，RELEFFECTIVEDISTSTYLE 則為 10、11 或 12，這表示有效分佈樣式為 AUTO (ALL)、AUTO (EVEN) 或 AUTO () KEY。如果資料表使用自動分佈，分佈樣式一開始可能會顯示 AUTO (ALL)，然後在資料表成長時變更為 AUTO (EVEN) 或 AUTO (KEY)。

下表提供資料 RELEFFECTIVEDISTSTYLE 欄中每個值的分佈樣式：

RELEFFECTIVEDISTSTYLE	目前分佈樣式
0	EVEN
1	KEY
8	ALL
10	AUTO (ALL)
11	AUTO (EVEN)
12	AUTO (KEY)

SVV_TABLE_INFO 中的 DISTSTYLE 資料欄表示資料表目前的分佈樣式。如果資料表使用自動分佈，則 DISTSTYLE 為 AUTO (ALL)、AUTO (EVEN) 或 AUTO (KEY)。

下列範例使用三種分佈樣式和自動分佈建立四個資料表，然後查詢 SVV_TABLE_INFO 以檢視分佈樣式。

```
create table public.dist_key (col1 int)
diststyle key distkey (col1);

insert into public.dist_key values (1);

create table public.dist_even (col1 int)
diststyle even;

insert into public.dist_even values (1);

create table public.dist_all (col1 int)
diststyle all;

insert into public.dist_all values (1);

create table public.dist_auto (col1 int);

insert into public.dist_auto values (1);

select "schema", "table", diststyle from SVV_TABLE_INFO
```

```
where "table" like 'dist%';
```

schema	table	diststyle
public	dist_key	KEY(col1)
public	dist_even	EVEN
public	dist_all	ALL
public	dist_auto	AUTO(ALL)

評估查詢模式

選擇分佈樣式只是資料庫設計的一個環節。請在整個系統的內容中考慮分佈樣式，同時使用其他重要因素來平均分佈，例如叢集大小、壓縮編碼方法、排序索引鍵，以及資料表限制。

利用儘可能接近真實資料的資料來測試您的系統。

若要為分佈樣式做出好選擇，您必須了解 Amazon Redshift 應用程式的查詢模式。識別系統中成本最高的查詢，並根據那些查詢的需求來設計初始的資料庫。決定查詢總成本的因素包括查詢執行所需的時間長度，以及耗用多少運算資源。決定查詢成本的其他因素是執行的頻率，以及對其他查詢和資料庫作業的干擾程度。

識別成本最高查詢使用的資料表，並評估其在查詢執行期中的角色。請考慮聯結和彙總資料表的方式。

使用本節中的準則來選擇每個資料表的分佈樣式。當您這樣做時，請建立資料表，並將儘可能接近真實資料的資料載入其中。然後測試資料表是否有您希望使用的查詢類型。您可以評估查詢解釋計畫來識別調校機會。比較載入時間、儲存空間和查詢執行期，以便平衡系統的整體需求。

指定分佈樣式

本節中指定分佈樣式的考量和建議會使用星狀結構描述做為範例。您的資料庫設計可能根據星狀結構描述、某個星狀結構描述變體，或完全不同的結構描述。Amazon Redshift 會設計為可搭配您選擇的任何結構描述設計而有效運作。本節中的原理適用於任何設計結構描述。

1. 指定您的所有資料表的主索引鍵和外部索引鍵。

Amazon Redshift 不會強制執行主索引鍵和外部索引鍵限制，但查詢最佳化器會在產生查詢計劃時使用這些限制。如果您設定主索引鍵和外部索引鍵，則您的應用程式必須維護索引鍵的有效性。

2. 將事實資料表與其最大維度資料表配送至它們的通用欄位。

根據參與最常用聯結的資料集大小，而不只是資料表的大小來選擇最大維度。如果通常會篩選資料表，請使用WHERE子句，則只有一部分的資料列會參與聯結。這類資料表對重新分佈比提供

更多資料的更小資料表具有更少的影響。將維度表的主索引鍵和事實表的對應外部索引鍵指定為 DISTKEY。如果多個資料表使用相同的分佈索引鍵，它們也將與事實資料表共置。您的事實資料表只能有一個分佈索引鍵。聯結其他索引鍵的任何資料表都不會與事實資料表共置。

3. 指定其他維度資料表的分佈索引鍵。

根據其主索引鍵或外部索引鍵來分佈資料表，取決於它們最常與其他資料表聯結的方式。

4. 評估是否要變更部分維度資料表以使用ALL分佈。

如果維度資料表無法與事實資料表或其他重要的聯結資料表共置，您可以藉由將整個資料表發佈至所有節點，以大幅提升查詢效能。使用ALL分佈會乘以儲存空間需求，並增加載入時間和維護操作，因此在選擇ALL分佈之前，您應該權衡所有因素。下一節說明如何透過評估EXPLAIN計劃來識別要ALL分發的候選者。

5. 使用其餘資料表的AUTO分佈。

如果資料表大幅非規範化且未參與聯結，或者您沒有另一個分佈樣式的明確選擇，請使用AUTO分佈。

若要讓 Amazon Redshift 選擇適當的分佈樣式，請不要明確指定分佈樣式。

評估查詢計畫

您可以使用查詢計畫來識別可將分佈樣式最佳化的候選者。

在做出初始設計決策之後，請建立資料表、將資料載入其中，然後測試它們。使用儘可能接近真實資料的測試資料集。測量載入時間做為比較的基準。

評估哪些查詢代表您預期執行成本最高的查詢，尤指使用聯結和彙總的查詢。比較各種設計選項的執行期。比較執行期時，請不要將第一次執行查詢的時間列入計算，因為第一次執行期包含編譯時間。

DS_DIST_NONE

不需要重新分佈，因為對應配量是在運算節點上共置。您通常只有一個 DS_DIST_NONE 步驟，也就是事實資料表和一個維度資料表之間的聯結。

DS_DIST_ALL_NONE

不需要重新分發，因為內部聯結表使用的是 DISTSTYLE ALL。整個資料表位於每個節點上。

DS_DIST_INNER

重新配送內部資料表。

DS_DIST_OUTER

重新配送外部資料表。

DS_BCAST_INNER

將整個內部資料表的副本播送至所有運算節點。

DS_DIST_ALL_INNER

整個內部資料表會重新分發至單一切片，因為外部資料表使用 DISTSTYLE ALL。

DS_DIST_BOTH

重新配送這兩個資料表。

DS_DIST_NONE 和 DSDIST__ALL__NONE 很好。它們指出該步驟不需要重新分佈，因為所有聯結已共置。

DS_DIST_INNER 表示因為內部資料表正在重新分發給節點，所以步驟的成本可能相對較高。DS_DIST_INNER 表示外部資料表已正確分佈在聯結金鑰上。將內部資料表的分發金鑰設定為聯結金鑰，以將其轉換為 DS_DIST_NONE。在某些情況下，無法在聯結索引鍵上分佈內部資料表，因為外部資料表不會在聯結索引鍵上分佈。如果是這種情況，請評估是否使用內部資料表的 ALL 分佈。如果資料表未頻繁或大量更新，且其大小足以負擔高重新分佈成本，請將分佈樣式變更為 ALL，然後再次測試。ALL 分佈會導致載入時間增加，因此當您重新測試時，請在評估因素中包含載入時間。

DS_DIST_ALL_INNER 不好。這表示整個內部資料表會重新分發給單一切片，因為外部資料表使用 DISTSTYLE ALL，因此整個外部資料表的副本會位於每個節點上。這會在單一節點上造成一連串無效的聯結執行期，而不是使用所有節點來善用平行執行期。DISTSTYLE ALL 僅適用於內部聯結表。反之，指定分佈索引鍵或甚至對外部資料表使用分佈。

DS_BCAST_INNER 和 DS_DIST_BOTH 不好。通常，這些重新分佈之所以發生，原因是未在資料表的分佈索引鍵上聯結資料表。如果事實資料表尚未有分佈索引鍵，請將聯結資料欄指定為這兩個資料表的分佈索引鍵。如果事實資料表已在另一個資料欄上具有分佈索引鍵，則請評估變更分佈索引鍵來共置此聯結是否將改善整體效能。如果變更外部資料表的分佈索引鍵不是最佳選擇，您可以透過 DISTSTYLE ALL 為內部資料表指定來實現錯位。

下列範例顯示具有 DS_BCAST_INNER 和 DS_DIST_NONE 標籤的查詢計劃的一部分。

```
-> XN Hash Join DS_BCAST_INNER (cost=112.50..3272334142.59 rows=170771 width=84)
    Hash Cond: ("outer".venueid = "inner".venueid)
```

```

-> XN Hash Join DS_BCAST_INNER (cost=109.98..3167290276.71 rows=172456
width=47)
    Hash Cond: ("outer".eventid = "inner".eventid)
    -> XN Merge Join DS_DIST_NONE (cost=0.00..6286.47 rows=172456 width=30)
        Merge Cond: ("outer".listid = "inner".listid)
        -> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497
width=14)
            -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=24)

```

將維度資料表變更為使用 DISTSTYLE 後 ALL，相同查詢的查詢計畫會顯示 DS_DISTALL_NONE 取代 DS_BCAST_INNER。另外，聯結步驟的相對成本會發生大幅變更。與之前查詢中的 3272334142.59 相比，總成本是 14142.59。

```

-> XN Hash Join DS_DIST_ALL_NONE (cost=112.50..14142.59 rows=170771 width=84)
    Hash Cond: ("outer".venueid = "inner".venueid)
    -> XN Hash Join DS_DIST_ALL_NONE (cost=109.98..10276.71 rows=172456 width=47)
        Hash Cond: ("outer".eventid = "inner".eventid)
        -> XN Merge Join DS_DIST_NONE (cost=0.00..6286.47 rows=172456 width=30)
            Merge Cond: ("outer".listid = "inner".listid)
            -> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497
width=14)
                -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=24)

```

查詢計畫範例

此範例顯示如何評估查詢計畫來找出最佳化分佈的機會。

使用 EXPLAIN 命令執行下列查詢，以產生查詢計畫。

```

explain
select lastname, catname, venueid, venuecity, venuestate, eventname,
month, sum(pricepaid) as buyercost, max(totalprice) as maxtotalprice
from category join event on category.catid = event.catid
join venue on venue.venueid = event.venueid
join sales on sales.eventid = event.eventid
join listing on sales.listid = listing.listid
join date on sales.dateid = date.dateid
join users on users.userid = sales.buyerid
group by lastname, catname, venueid, venuecity, venuestate, eventname, month
having sum(pricepaid)>9999
order by catname, buyercost desc;

```

在 TICKIT 資料庫中，SALES 是事實資料表，LISTING 也是其最大維度。為了整理資料表，SALES 會分佈在上LISTID，這是 的外部金鑰LISTING，而 LISTING 則分佈在其主金鑰 上LISTID。下列範例顯示 SALES和 的CREATETABLE命令LISTING。

```
create table sales(  
  salesid integer not null,  
  listid integer not null distkey,  
  sellerid integer not null,  
  buyerid integer not null,  
  eventid integer not null encode mostly16,  
  dateid smallint not null,  
  qtysold smallint not null encode mostly8,  
  pricepaid decimal(8,2) encode delta32k,  
  commission decimal(8,2) encode delta32k,  
  saletime timestamp,  
  primary key(salesid),  
  foreign key(listid) references listing(listid),  
  foreign key(sellerid) references users(userid),  
  foreign key(buyerid) references users(userid),  
  foreign key(dateid) references date(dateid))  
  sortkey(listid,sellerid);  
  
create table listing(  
  listid integer not null distkey sortkey,  
  sellerid integer not null,  
  eventid integer not null encode mostly16,  
  dateid smallint not null,  
  numtickets smallint not null encode mostly8,  
  priceperticket decimal(8,2) encode bytedict,  
  totalprice decimal(8,2) encode mostly32,  
  listtime timestamp,  
  primary key(listid),  
  foreign key(sellerid) references users(userid),  
  foreign key(eventid) references event(eventid),  
  foreign key(dateid) references date(dateid));
```

在下列查詢計劃中，聯結的合併聯結步驟SALES LISTING會顯示 DS_DIST_NONE，這表示該步驟不需要重新分發。但是，向上移動查詢計畫，另一個內部聯結會顯示 DS_BCAST_INNER，這表示內部資料表在查詢執行中廣播。因為只有一對資料表可以使用索引鍵分佈來進行共置，所以五個資料表必須重新播送。

QUERY PLAN


```

XN Merge (cost=1015345167117.54..1015345167544.46 rows=1000 width=103)
  Merge Key: category.catname, sum(sales.pricepaid)
  -> XN Network (cost=1015345167117.54..1015345167544.46 rows=170771 width=103)
    Send to leader
    -> XN Sort (cost=1015345167117.54..1015345167544.46 rows=170771 width=103)
      Sort Key: category.catname, sum(sales.pricepaid)
      -> XN HashAggregate (cost=15345150568.37..15345152276.08 rows=170771
width=103)
        Filter: (sum(pricepaid) > 9999.00)
        -> XN Hash Join DS_BCAST_INNER (cost=742.08..15345146299.10
rows=170771 width=103)
          Hash Cond: ("outer".catid = "inner".catid)
          -> XN Hash Join DS_BCAST_INNER
(cost=741.94..15342942456.61 rows=170771 width=97)
            Hash Cond: ("outer".dateid = "inner".dateid)
            -> XN Hash Join DS_BCAST_INNER
(cost=737.38..15269938609.81 rows=170766 width=90)
              Hash Cond: ("outer".buyerid = "inner".userid)
              -> XN Hash Join DS_BCAST_INNER
(cost=112.50..3272334142.59 rows=170771 width=84)
                Hash Cond: ("outer".venueid =
"inner".venueid)
                -> XN Hash Join DS_BCAST_INNER
(cost=109.98..3167290276.71 rows=172456 width=47)
                  Hash Cond: ("outer".eventid =
"inner".eventid)
                  -> XN Merge Join DS_DIST_NONE
(cost=0.00..6286.47 rows=172456 width=30)
                    Merge Cond: ("outer".listid =
"inner".listid)
                    -> XN Seq Scan on listing
(cost=0.00..1924.97 rows=192497 width=14)
                      -> XN Seq Scan on sales
(cost=0.00..1724.56 rows=172456 width=24)
                        -> XN Hash (cost=87.98..87.98
rows=8798 width=25)
                          -> XN Seq Scan on event
(cost=0.00..87.98 rows=8798 width=25)
                            -> XN Hash (cost=2.02..2.02 rows=202
width=41)
                              -> XN Seq Scan on venue
(cost=0.00..2.02 rows=202 width=41)
                                -> XN Hash (cost=499.90..499.90 rows=49990
width=14)

```

```

                                -> XN Seq Scan on users
(cost=0.00..499.90 rows=49990 width=14)
                                -> XN Hash (cost=3.65..3.65 rows=365 width=11)
                                    -> XN Seq Scan on date (cost=0.00..3.65
rows=365 width=11)
                                -> XN Hash (cost=0.11..0.11 rows=11 width=10)
                                    -> XN Seq Scan on category (cost=0.00..0.11 rows=11
width=10)

```

其中一個解決方案是將資料表變更為具有 DISTSTYLE ALL。

```

ALTER TABLE users ALTER DISTSTYLE ALL;
ALTER TABLE venue ALTER DISTSTYLE ALL;
ALTER TABLE category ALTER DISTSTYLE ALL;
ALTER TABLE date ALTER DISTSTYLE ALL;
ALTER TABLE event ALTER DISTSTYLE ALL;

```

EXPLAIN 再次使用 執行相同的查詢，並檢查新的查詢計劃。聯結現在會顯示 DSDIST__ALL_NONE，表示不需要重新分發，因為資料是使用 DISTSTYLE 分發到每個節點ALL。

```

QUERY PLAN
XN Merge (cost=1000000047117.54..1000000047544.46 rows=1000 width=103)
  Merge Key: category.catname, sum(sales.pricepaid)
    -> XN Network (cost=1000000047117.54..1000000047544.46 rows=170771 width=103)
      Send to leader
        -> XN Sort (cost=1000000047117.54..1000000047544.46 rows=170771 width=103)
          Sort Key: category.catname, sum(sales.pricepaid)
            -> XN HashAggregate (cost=30568.37..32276.08 rows=170771 width=103)
              Filter: (sum(pricepaid) > 9999.00)
                -> XN Hash Join DS_DIST_ALL_NONE (cost=742.08..26299.10
rows=170771 width=103)
                  Hash Cond: ("outer".buyerid = "inner".userid)
                    -> XN Hash Join DS_DIST_ALL_NONE (cost=117.20..21831.99
rows=170766 width=97)
                      Hash Cond: ("outer".dateid = "inner".dateid)
                        -> XN Hash Join DS_DIST_ALL_NONE
(cost=112.64..17985.08 rows=170771 width=90)
                          Hash Cond: ("outer".catid = "inner".catid)
                            -> XN Hash Join DS_DIST_ALL_NONE
(cost=112.50..14142.59 rows=170771 width=84)
                              Hash Cond: ("outer".venueid =
"inner".venueid)

```

```

-> XN Hash Join DS_DIST_ALL_NONE
(cost=109.98..10276.71 rows=172456 width=47)
      Hash Cond: ("outer".eventid =
"inner".eventid)
-> XN Merge Join DS_DIST_NONE
(cost=0.00..6286.47 rows=172456 width=30)
      Merge Cond: ("outer".listid =
"inner".listid)
-> XN Seq Scan on listing
(cost=0.00..1924.97 rows=192497 width=14)
-> XN Seq Scan on sales
(cost=0.00..1724.56 rows=172456 width=24)
-> XN Hash (cost=87.98..87.98
rows=8798 width=25)
      -> XN Seq Scan on event
(cost=0.00..87.98 rows=8798 width=25)
-> XN Hash (cost=2.02..2.02 rows=202
width=41)
      -> XN Seq Scan on venue
(cost=0.00..2.02 rows=202 width=41)
-> XN Hash (cost=0.11..0.11 rows=11 width=10)
      -> XN Seq Scan on category
(cost=0.00..0.11 rows=11 width=10)
      -> XN Hash (cost=3.65..3.65 rows=365 width=11)
      -> XN Seq Scan on date (cost=0.00..3.65
rows=365 width=11)
-> XN Hash (cost=499.90..499.90 rows=49990 width=14)
      -> XN Seq Scan on users (cost=0.00..499.90 rows=49990
width=14)

```

分佈範例

下列範例顯示資料如何根據您在 `CREATETABLE` 陳述式中定義的選項分佈。

DISTKEY 範例

查看 `TICKIT` 資料庫中 `USERS` 資料表的結構描述。 `USERID` 定義為資料 `SORTKEY` 欄和資料 `DISTKEY` 欄：

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'users';
```

column	type	encoding	distkey	sortkey
--------	------	----------	---------	---------

```

-----+-----+-----+-----+-----
userid      | integer          | none      | t      |      1
username    | character(8)     | none      | f      |      0
firstname   | character varying(30) | text32k   | f      |      0
...

```

USERID 是此表格上分佈欄的好選擇。如果您查詢 SVV_DISKUSAGE 系統檢視，您可以看到資料表分佈非常均勻。資料欄編號為零，資料欄 0 USERID 也是。

```

select slice, col, num_values as rows, minvalue, maxvalue
from svv_diskusage
where name='users' and col=0 and rows>0
order by slice, col;

```

```

slice| col | rows | minvalue | maxvalue
-----+-----+-----+-----+-----
0    | 0  | 12496 | 4         | 49987
1    | 0  | 12498 | 1         | 49988
2    | 0  | 12497 | 2         | 49989
3    | 0  | 12499 | 3         | 49990
(4 rows)

```

資料表包含 49,990 個資料列。資料列 (num_values) 資料欄顯示每個配量包含大約相同數目的資料列。minvalue 和 maxvalue 資料欄顯示每一個配量上的值範圍。每個層幾乎都包含整個範圍的值，因此每個層都有很大機會參與執行查詢，該查詢會篩選使用者的範圍IDs。

此範例示範小型測試系統上的分佈。配量總數通常高得多。

如果您經常使用 STATE 欄加入或分組，您可以選擇在 STATE 欄上分發。下列範例顯示您建立具有與資料表相同資料但 DISTKEY 將設定為 STATE 欄之新 USERS 資料表的案例。在此情況下，分佈不是平均的。配量 0 (13,587 個資料列) 比配量 3 (10,150 個資料列) 多保留大約 30% 的資料列。在規模很大的資料表中，此分佈扭曲數量可能對查詢處理具有負面影響。

```

create table userskey distkey(state) as select * from users;

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'userskey' and col=0 and rows>0
order by slice, col;

slice | col | rows | minvalue | maxvalue
-----+-----+-----+-----+-----

```

```

 0 | 0 | 13587 | 5 | 49989
 1 | 0 | 11245 | 2 | 49990
 2 | 0 | 15008 | 1 | 49976
 3 | 0 | 10150 | 4 | 49986
(4 rows)

```

DISTSTYLE EVEN 範例

如果您使用與資料表相同的資料建立新USERS資料表，但將 DISTSTYLE設定為 EVEN，則資料列一律會平均分佈於切片。

```

create table userseven diststyle even as
select * from users;

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'userseven' and col=0 and rows>0
order by slice, col;

```

```

slice | col | rows | minvalue | maxvalue
-----+-----+-----+-----+-----
 0 | 0 | 12497 | 4 | 49990
 1 | 0 | 12498 | 8 | 49984
 2 | 0 | 12498 | 2 | 49988
 3 | 0 | 12497 | 1 | 49989
(4 rows)

```

不過，因為分佈不是根據特定資料欄，所以查詢處理能力可能降低，尤其在資料表聯結至其他資料表時更是如此。聯結資料欄若少了分佈，通常會影響可以有效地執行聯結操作的類型。當這兩個資料表在各自聯結資料欄上進行分佈和排序時，會最佳化聯結、彙總和分組操作。

DISTSTYLE ALL 範例

如果您使用與資料表相同的資料建立新USERS資料表，但將 DISTSTYLE設定為 ALL，則所有資料列都會分佈到每個節點的第一個層。

```

select slice, col, num_values as rows, minvalue, maxvalue from svv_diskusage
where name = 'usersall' and col=0 and rows > 0
order by slice, col;

```

```

slice | col | rows | minvalue | maxvalue
-----+-----+-----+-----+-----

```

```

0 | 0 | 49990 |          4 |          49990
2 | 0 | 49990 |          2 |          49990

```

(4 rows)

排序金鑰

Note

建議您使用 SORTKEY AUTO 建立資料表。如果您這麼做，Amazon Redshift 會使用自動資料表最佳化來選擇排序索引鍵。如需詳細資訊，請參閱[自動資料表最佳化](#)。本節的其餘部分會提供有關排序順序的詳細資訊。

或者，建立資料表時，您可以定義一或多個資料欄做為排序索引鍵。當資料最初載入至空的資料表時，資料列會依排序儲存在磁碟上。排序索引鍵資料欄的相關資訊會傳遞至查詢規劃器，而且規劃器會使用此資訊，來建構利用資料排序方式的計劃。如需詳細資訊，請參閱[CREATE TABLE](#)。如需建立排序索引鍵時最佳作法的資訊，請參閱[選擇最佳的排序索引鍵](#)。

排序可以有效率地處理限制範圍的述詞。Amazon Redshift 將單欄式資料儲存在 1 MB 磁碟區塊中。每個區塊的 min 和 max 值都會儲存為中繼資料的一部分。如果查詢使用範圍受限的述詞，則查詢處理器可以使用 min 和 max 值，在資料表掃描期間快速地略過大量區塊。例如，假設資料表儲存了依日期排序的五年資料，而查詢指定了一個月的日期範圍。在這種情況下，您最多可以從掃描中移除 98% 的磁碟區塊。如果未排序資料，則必須掃描更多的磁碟區塊 (可能全部)。

您可以指定複合或交錯排序索引鍵。當查詢述詞使用字首，即依序的排序索引鍵子集時，複合排序索引鍵更有效。交錯排序索引鍵對排序索引鍵中的每個資料欄都提供相等的權重，所以查詢述詞可以使用構成排序索引鍵之資料欄的任何子集，順序不拘。

若要了解選擇的排序索引鍵對查詢效能的影響，請使用 [EXPLAIN](#) 命令。如需詳細資訊，請參閱[查詢計劃和執行工作流程](#)。

若要定義排序類型，請將 INTERLEAVED 或 COMPOUND 關鍵字與 CREATETABLE 或 CREATE TABLE AS 陳述式搭配使用。預設值為 COMPOUND。COMPOUND 當您定期使用 INSERT、或 DELETE 操作更新資料表時 UPDATE，建議使用。INTERLEAVED 排序索引鍵最多可使用八個資料欄。視您的資料和叢集大小而定，VACUUM REINDEX 會花費比更長的時間，VACUUMFULL 因為它會產生額外的傳遞來分析交錯的排序金鑰。交錯資料表的排序和合併操作可能需要更長的時間，因為交錯排序可能需要重新排列比複合排序更多的資料列。

若要檢視資料表的排序索引鍵，請查詢 [SVV_TABLE_INFO](#) 系統檢視。

主題

- [多維資料配置排序 \(預覽\)](#)
- [複合排序索引鍵](#)
- [交錯排序索引鍵](#)

多維資料配置排序 (預覽)

以下是預覽版中資料表的多維資料配置排序的發行前版本文件。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

Note

此功能僅可透過預覽叢集或預覽工作群組使用。若要建立預覽叢集，請參閱 Amazon Redshift 管理指南中的 [建立預覽叢集](#)。若要建立預覽工作群組，請參閱 Amazon Redshift 管理指南中的 [建立預覽工作群組](#)。

多維度資料配置排序索引鍵是一種 AUTO 排序索引鍵，以工作負載中找到的重複述詞為基礎。如果您的工作負載具有重複述詞，Amazon Redshift 可以透過共置滿足重複述詞的資料列來改善資料表掃描效能。多維資料配置排序索引鍵會藉由分析工作負載中出現的重複述詞來存放資料，而不是以嚴格的資料欄順序存放資料表的資料。在工作負載中可以找到多個重複述詞。根據您的工作負載，這種排序索引鍵可以改善許多述詞的效能。Amazon Redshift 會自動判斷使用 AUTO 排序索引鍵定義的資料表是否應該使用排序索引鍵方法。

例如，假設您的資料表的資料是按照資料欄排序。可能需要檢查許多資料區塊，藉以判斷該資料是否符合工作負載中的述詞。但是，如果資料是按照述詞順序儲存在磁碟上，則需要掃描較少區塊以滿足查詢。在這種情況下，使用多維資料配置排序索引鍵是有好處的。

若要檢視查詢是否使用多維資料配置索引鍵，請參閱 [SYS_QUERY_DETAIL](#) 視觀表的 `step_attribute` 資料欄。當該值為 `multi-dimensional`，則將多維資料配置用於查詢。若要檢視以 AUTO 排序索引鍵定義的資料表是否使用多維資料配置，請參閱 [SVV_TABLE_INFO](#) 檢視的 `sortkey1` 欄。當該值為 `padb_internal_mddl_key_col`，則將多維資料配置用於資料表排序索引鍵。

若要防止 Amazon Redshift 使用多維資料配置排序索引鍵，請選擇不同的資料表排序索引鍵選項。SORTKEY AUTO如需SORTKEY選項的詳細資訊，請參閱 [CREATE TABLE](#)。

複合排序索引鍵

複合索引鍵是由排序索引鍵定義中列出的所有資料欄組成，並依它們的列出順序排列。當查詢的篩選條件套用條件 (例如篩選條件和聯結) 來使用排序索引鍵的字首時，複合排序索引鍵最有用。當查詢僅依賴次要排序資料欄，而未參考主要資料欄時，複合排序的效能優勢會降低。COMPOUND 是預設排序類型。

複合排序索引鍵可能會加快聯結、GROUPBY 和 ORDER BY 操作，以及使用 PARTITION BY 和 ORDER BY 的視窗函數。例如，在聯結資料欄上分佈和預先排序資料時，通常比雜湊聯結還要快的合併聯結更為可行。複合排序索引鍵也可協助改善壓縮。

當您將資料列新增至已包含資料的已排序資料表時，未排序的區域會擴大，因而對效能產生重大影響。當資料表使用交錯排序時，效果更明顯，尤其是排序資料欄包含單調增加的資料 (例如日期或時間戳記資料欄) 時更明顯。定期執行VACUUM操作，特別是在大量資料載入之後，以重新排序和重新分析資料。如需詳細資訊，請參閱[縮小未排序區域的大小](#)。使用吸塵器清理資料後，執行ANALYZE命令來更新查詢規劃器的統計中繼資料是很好的做法。如需詳細資訊，請參閱[分析資料表](#)。

交錯排序索引鍵

交錯排序可對排序索引鍵中的每個資料欄或資料欄子集提供相等的權重。如果多個查詢使用不同資料欄進行篩選，則您通常可以使用交錯排序樣式來改善那些查詢的效能。當查詢在次要排序資料欄上使用限制性述詞時，相較於複合排序，交錯排序可大幅改善查詢效能。

Important

不要在具有依序增加屬性 (如身分資料欄、日期或時間戳記) 的資料欄上使用交錯排序索引鍵。

透過實作交錯排序索引鍵所獲得的效能改善，應該與增加的載入和清空時間進行權衡。

交錯排序對於篩選子WHERE句中一或多個排序索引鍵資料欄的高度選擇性查詢最有效，例如 `select c_name from customer where c_region = 'ASIA'`。隨著受限之排序資料欄的數目增加，交錯排序的優勢更明顯。

使用大型資料表時，交錯排序更有效。排序會套用至每個配量。因此，當資料表足夠大到每個配量需要多個 1 MB 區塊時，交錯排序最有效。在這裡，查詢處理器可以使用限制性述詞跳過區塊的顯著比例。若要檢視資料表使用的區塊數目，請查詢 [STV_BLOCKLIST](#) 系統檢視。

在單一資料欄上進行排序時，如果資料欄值具有很長的常用字首，則交錯排序可能提供比複合排序更好的效能。例如，URLs通常以 "http://www". 開頭 複合排序索引鍵使用字首中數目受限的字元，因而導致許多索引鍵重複。交錯排序會對區域映射值使用內部壓縮方法，讓它們更能區分常用字首很長的資料欄值。

將 Amazon Redshift 佈建叢集遷移至 Amazon Redshift Serverless 時，Redshift 會使用交錯排序索引鍵DISTSTYLEKEY將資料表轉換為複合排序索引鍵。DISTSTYLE 不會變更。如需分佈樣式的相關資訊，請參閱[使用資料分佈樣式](#)。

VACUUM REINDEX

當您將資料列新增至已包含資料的已排序資料表時，效能可能會在一段時間後惡化。複合排序和交錯排序都會發生這種惡化，但是對交錯資料表的影響更大。會VACUUM還原排序順序，但交錯資料表的操作可能需要更長的時間，因為合併新的交錯資料可能涉及修改每個資料區塊。

最初載入資料表時，Amazon Redshift 會分析排序索引鍵資料欄中值的分佈，並使用該資訊以取得排序索引鍵資料欄的最佳交錯。當資料表越來越大時，排序索引鍵資料欄中值的分佈可能會變更或扭曲，尤其是日期或時間戳記資料欄。如果扭曲變得太大，效能可能會受到影響。若要重新分析排序金鑰並還原效能，請使用REINDEX關鍵字執行 VACUUM 命令。因為必須額外分析傳遞資料，VACUUM REINDEX 可能需要比VACUUM交錯資料表的標準更長的時間。若要檢視索引鍵分佈扭曲及上次重建索引時間的相關資訊，請查詢 [SVV_INTERLEAVED_COLUMNS](#) 系統檢視。

如需如何判斷執行頻率VACUUM以及執行時間的詳細資訊VACUUMREINDEX，請參閱 [決定是否重新索引](#)。

資料表限制

唯一性、主索引鍵和外部索引鍵限制僅供參考，Amazon Redshift 不會在您填入資料表時強制執行它們。例如，如果您將資料插入具有相依性的資料表中，即使插入違反限制也可以成功執行。不過，主要金鑰和外部金鑰會用作規劃提示，如果您ETL的程序或應用程式中的其他程序強制執行其完整性，則應宣告這些提示。

例如，查詢規劃工具會在特定統計計算中使用主索引鍵和外部索引鍵。這樣做是為了推斷影響子查詢裝飾關係技術的唯一性和參考關係。透過這樣做，它可以排序大量聯結並刪除多餘的聯結。

規劃器會運用這些索引鍵關係，但其假設 Amazon Redshift 資料表中的所有索引鍵與載入時一樣有效。如果您的應用程式允許無效的外部索引鍵或主索引鍵，則有些查詢可能傳回不正確的結果。例如，如果主索引鍵不是唯一的，SELECTDISTINCT查詢可能會傳回重複的資料列。如果您懷疑其有效性的話，請勿對您的資料表定義索引鍵限制。但是，當您知道主索引鍵和外部索引鍵有效時，您應該一律宣告它們和限制唯一性。

Amazon Redshift 會強制執行NOTNULL資料欄限制。

如需資料表限制條件的相關資訊，請參閱[CREATE TABLE](#)。如需如何捨棄具有相依性之資料表的資訊，請參閱 [DROP TABLE](#)。

在 Amazon Redshift 中載入資料

有數種方式可將資料載入 Amazon Redshift 資料庫。要載入的熱門資料來源之一是 Amazon S3 檔案。下表摘要說明從 Amazon S3 來源開始使用的一些方法。

使用方法	描述	需要方法時
COPY 命令	執行批次檔案擷取，以從 Amazon S3 檔案載入資料。此方法利用 Amazon Redshift 的平行處理功能。如需詳細資訊，請參閱 使用 COPY 命令載入資料表 。	當需要手動啟動批次檔案擷取的基本資料載入需求時，應使用。此方法主要用於自訂和第三方檔案擷取管道，或一次性或臨時檔案擷取工作負載。
COPY... CREATE JOB 命令 (自動複製)	在追蹤的 Amazon S3 路徑上建立新檔案時，會自動執行您的 COPY 命令。如需詳細資訊，請參閱 從 Amazon S3 載入具有連續檔案擷取的資料表 (預覽) 。	在 Amazon S3 上建立新檔案時，當檔案擷取管道需要自動擷取資料時，應使用。Amazon Redshift 會追蹤擷取的檔案，以防止資料重複。此方法需要由 Amazon S3 儲存貯體擁有者進行組態。
從資料湖查詢載入	建立外部資料表以在 Amazon S3 檔案上執行資料湖查詢，然後執行 INSERT INTO 命令，將資料湖查詢的結果載入本機資料表。如需詳細資訊，請參閱 Redshift Spectrum 的外部資料表 。	應用於下列任何案例： <ul style="list-style-type: none"> 從表格格式載入 AWS Glue 和開啟 (例如 Apache Iceberg、Apache Hudi 或 Delta Lake)。 需要部分擷取來源檔案 (例如，執行 WHERE 子句以擷取特定資料列時需要)。 擷取特定資料欄 (例如執行 SELECT 命令) 或隨時隨地進行基本資料轉換 (例如套用基本操作或呼叫來源檔案

使用方法	描述	需要方法時
		UDFs的值) 所需的更多彈性。
您可以考慮的其他方法		
串流擷取	串流擷取提供低延遲、高速的串流資料擷取，從 Amazon Kinesis Data Streams 和 Amazon Managed Streaming for Apache Kafka 擷取到 Amazon Redshift 佈建或 Redshift Serverless 具體化檢視。如需詳細資訊，請參閱 開始使用 Amazon Kinesis Data Streams 中的串流擷取 和 從 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 開始串流擷取 。	當資料先串流到 Amazon S3 上的檔案，然後從 Amazon S3 載入時，應考慮使用案例。如果不需要在 Amazon S3 上保留資料，您通常可以考慮將資料直接串流到 Amazon Redshift。
執行資料湖查詢	直接從資料湖資料表執行查詢，而不是將資料表的內容擷取到本機資料表。如需詳細資訊，請參閱 Amazon Redshift Spectrum 。	當使用案例不需要在 Amazon Redshift 中執行本機資料表查詢時，應使用。
使用 Amazon Redshift 查詢編輯器 v2 進行批次載入	您可以在 Amazon Redshift 查詢編輯器 v2 上以視覺化方式準備和執行批次檔案擷取工作負載。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的 從 S3 載入資料 。	當您想要查詢編輯器 v2 準備 COPY 陳述式，而且想要視化工具簡化 COPY 陳述式準備程序時，應使用。

使用方法	描述	需要方法時
使用 Amazon Redshift 查詢編輯器 v2 從本機檔案載入資料	您可以從桌面直接將檔案上傳至 Amazon Redshift 資料表，而不需要手動將檔案上傳至 Amazon S3。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的 從本機檔案設定和工作流程載入資料 。	當您需要快速從本機電腦載入檔案以進行一次性查詢時，應使用。透過此方法，Amazon Redshift 查詢編輯器 v2 會將檔案暫時存放在客戶擁有的 Amazon S3 儲存貯體上，並使用此 Amazon S3 路徑執行複製命令。

COPY 命令是載入資料表的最有效方法。您也可以使用 INSERT 命令將資料新增至資料表，但其效率遠低於使用 COPY。COPY 命令可以同時從多個資料檔案或多個資料串流讀取。Amazon Redshift 會將工作負載配置到 Amazon Redshift 節點，並平行執行載入操作，包括排序資料列和跨節點層分佈資料。

Note

Amazon Redshift Spectrum 外部資料表處於唯讀狀態。您無法 COPY 或 INSERT 至外部資料表。

若要存取其他 AWS 資源上的資料，Amazon Redshift 必須具有存取這些資源的許可，並執行存取資料的必要動作。您可以使用 AWS Identity and Access Management (IAM) 限制使用者對 Amazon Redshift 資源和資料的存取。

初始資料載入後，如果您新增、修改或刪除大量資料，您應該執行 VACUUM 命令來重新組織資料，並在刪除後回收空間。您也應該執行 ANALYZE 命令來更新資料表統計資料。

主題

- [使用 COPY 命令載入資料表](#)
- [從 Amazon S3 載入具有連續檔案擷取的資料表 \(預覽\)](#)
- [使用 DML 命令載入資料表](#)
- [執行深層複製](#)
- [分析資料表](#)
- [清空資料表](#)

- [管理並行寫入操作](#)
- [教學課程：從 Amazon S3 載入資料](#)

使用 COPY命令載入資料表

此COPY命令會利用 Amazon Redshift 大規模平行處理（MPP）架構，從 Amazon S3 上的檔案、DynamoDB 資料表或從一或多個遠端主機的文字輸出平行讀取和載入資料。

在了解COPY命令的所有選項之前，建議您先了解載入 Amazon S3 資料的基本選項。Amazon Redshift 入門指南示範 COPY命令的簡單使用方式，以使用預設IAM角色載入 Amazon S3 資料。如需詳細資訊，請參閱[步驟 4：將資料從 Amazon S3 載入至 Amazon Redshift](#)。

Note

我們強烈建議使用 COPY命令載入大量資料。使用個別INSERT陳述式填入資料表可能會非常慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 INSERT INTO ... SELECT 或 CREATE TABLE AS 來改善效能。如需詳細資訊，請參閱 [INSERT](#) 或 [CREATE TABLE AS](#)。

若要從其他 AWS 資源載入資料，Amazon Redshift 必須具有存取資源和執行必要動作的許可。

若要授予或撤銷使用COPY命令將資料載入資料表的權限，請授予或撤銷INSERT權限。

您的資料必須採用適當的格式，才能載入至您的 Amazon Redshift 資料表。本節介紹在載入之前準備和驗證資料的準則，以及在執行陳述式之前驗證COPY陳述式的準則。

為了保護您檔案中的資訊，您可以在將資料檔案上傳至 Amazon S3 儲存貯體之前加密資料檔案；COPY會在資料執行載入時解密資料。您也可以對使用者提供暫時的安全性登入資料，來限制對您的載入資料的存取。暫時安全性登入資料提供加強的安全性，因為有效期限較短，且過期之後不能重複使用。

Amazon Redshift 具有內建於的功能，COPY可快速載入未壓縮、分隔的資料。但您可以使用 gzip、lzop 或 bzip2 壓縮檔案來節省上傳檔案的時間。

如果COPY查詢中有下列關鍵字，則不支援自動分割未壓縮的資料：ESCAPE、REMOVEQUOTES和 FIXEDWIDTH。但支援 CSV關鍵字。

為了協助確保資料在 AWS Cloud 內傳輸時的安全，Amazon Redshift 使用硬體加速SSL與 Amazon S3 或 Amazon DynamoDB 通訊，以進行 COPY、UNLOAD備份和還原操作。

直接從 Amazon DynamoDB 資料表載入您的資料表時，您可以選擇控制所使用的 Amazon DynamoDB 佈建輸送量。

您可以選擇讓 COPY分析輸入資料，並在載入程序中自動將最佳壓縮編碼套用至資料表。

主題

- [登入資料和存取許可](#)
- [準備您的輸入資料](#)
- [從 Amazon S3 載入資料](#)
- [從 Amazon 載入資料 EMR](#)
- [從遠端主機載入資料](#)
- [從 Amazon DynamoDB 資料表載入資料](#)
- [驗證資料已正確載入](#)
- [驗證輸入資料](#)
- [利用自動壓縮載入資料表](#)
- [優化用於縮小資料表的儲存](#)
- [載入預設的欄位值](#)
- [針對資料載入進行故障診斷](#)

登入資料和存取許可

若要使用 Amazon S3、Amazon DynamoDB、Amazon EC2 或 Amazon EMR 等其他 AWS 資源載入或卸載資料，Amazon Redshift 必須具有存取資源的許可，並執行存取資料的必要動作。例如，若要從 Amazon S3 載入資料，COPY 必須具有儲存貯體的 LIST 存取權，以及儲存貯體物件的 GET 存取權。

若要取得存取資源的授權，Amazon Redshift 必須經過身分驗證。您可以選擇角色型存取控制或金鑰型存取控制中的一個。本節說明這兩個方法的概觀。如需詳細資訊和範例，請參閱[存取其他 AWS 資源的許可](#)。

角色類型存取控制

透過角色型存取控制，Amazon Redshift 會暫時代表您擔任 AWS Identity and Access Management (IAM) 角色。然後，根據授予角色的授權，Amazon Redshift 可以存取所需的 AWS 資源。

我們建議您使用角色型存取控制，因為它除了保護您的 AWS 憑證之外，還提供更安全、更精細的存取 AWS 資源和敏感使用者資料的控制。

若要使用角色型存取控制，您必須先使用 Amazon Redshift 服務IAM角色類型建立角色，然後將角色連接至資料倉儲。角色至少必須具備 [IAM COPY、UNLOAD和的許可 CREATE LIBRARY](#) 中列出的許可。如需建立IAM角色並將其連接至叢集的步驟，請參閱 [Amazon Redshift 管理指南 中的建立IAM角色以允許 Amazon Redshift 叢集存取 AWS 服務](#)。

您可以使用 Amazon Redshift 管理主控台、或將角色新增至叢集CLI，或檢視與叢集相關聯的角色API。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的[使用IAM角色授權COPY和UNLOAD操作](#)。

當您建立IAM角色時，會IAM傳回角色的 Amazon Resource Name (ARN)。若要使用IAM角色執行 COPY命令，ARN請使用 IAM_ROLE 參數或 CREDENTIALS 參數提供角色。

下列COPY命令範例使用 IAM_ROLE 參數搭配 角色MyRedshiftRole進行身分驗證。

```
COPY customer FROM 's3://amzn-s3-demo-bucket/mydata'  
IAM_ROLE 'arn:aws:iam::12345678901:role/MyRedshiftRole';
```

AWS 使用者至少必須擁有 中列出的許可[IAM COPY、UNLOAD和的許可 CREATE LIBRARY](#)。

金鑰型存取控制

透過金鑰型存取控制，您可以為有權存取包含資料之 AWS 資源的使用者提供存取金鑰 ID 和秘密存取金鑰。

Note

我們強烈建議使用IAM角色進行身分驗證，而不是提供純文字存取金鑰 ID 和秘密存取金鑰。如果您選擇以金鑰為基礎的存取控制，切勿使用 AWS 您的帳戶（根）憑證。一律建立IAM使用者，並提供該使用者的存取金鑰 ID 和秘密存取金鑰。如需建立IAM使用者的步驟，請參閱在[AWS 您的帳戶中建立IAM使用者](#)。

準備您的輸入資料

如果您的輸入資料與將接收資料的資料表資料欄不相容，則COPY命令將會失敗。

使用下列準則來幫助確保您的輸入資料有效：

- 您的資料只能包含 UTF-8 個字元，最長四個位元組。

- 確認 CHAR 和 VARCHAR 字串的長度不超過對應資料欄的長度。VARCHAR 字串以位元組為單位，而不是字元，因此，例如，佔用四個位元組的四個字元字串，每個都需要一個 VARCHAR (16) 欄。
- 多位元組字元只能與 VARCHAR 資料欄搭配使用。驗證多位元組字元長度不超過四個位元組。
- 確認 CHAR 資料欄的資料僅包含單位元組字元。
- 請勿包含任何特殊字元或語法來指出記錄中的最後一個欄位。此欄位可以是分隔符號。
- 如果您的資料包含 null 終止子，也稱為 NUL (UTF-8 0000) 或二進位零 (0x000) ，您可以使用 COPY 命令中的 NULL AS 選項將這些字元載入 NULLS CHAR 或 null as '\0' null as '\000' VARCHAR 資料欄。如果您不使用 NULL AS ， null 終止器會導致您的 COPY 失敗。
- 如果您的字串包含特殊字元，例如分隔符號和內嵌的新行，請使用 ESCAPE 選項搭配 [COPY](#) 命令。
- 驗證所有單引號和雙引號均正確成對。
- 驗證浮點字串採用標準浮點格式，例如 12.123 ，或指數格式，例如 1.0E4 。
- 驗證所有時間戳記和日期字串遵循 [DATEFORMAT](#) 和 [TIMEFORMAT](#) 字串的規格。預設時間戳記格式為 YYYY-MM-DD hh : mm : ss ，預設日期格式為 YYYY-MM-DD 。
- 如需個別資料類型的界限和限制的相關資訊，請參閱[資料類型](#)。如需多位元組字元錯誤的詳細資訊，請參閱[多位元組字元載入錯誤](#)

從 Amazon S3 載入資料

此 COPY 命令會利用 Amazon Redshift 大規模平行處理 (MPP) 架構，從 Amazon S3 儲存貯體中的檔案或多個檔案平行讀取和載入資料。如果檔案經過壓縮，您可以將資料分割為多個檔案，以充分利用平行處理的優勢。(此規則有例外情況。如需詳細資訊，請參閱[載入資料檔案](#)。) 您也可以藉由在資料表上設定分佈索引鍵來善加利用平行處理。如需分佈索引鍵的相關資訊，請參閱[查詢最佳化的資料分佈](#)。

資料會載入到目標資料表，每個資料列一行。資料檔案中的欄位會依次與資料表資料欄對應，由左至右。資料檔案中的欄位可以是固定寬度或以字元分隔；預設的分隔符號為管線 (|)。依預設，所有資料表資料欄已載入，但您可以選擇性地定義以逗點分隔的資料欄清單。如果 COPY 命令中指定的資料欄清單中未包含資料表資料欄，則會以預設值載入。如需詳細資訊，請參閱[載入預設的欄位值](#)。

主題

- [從已壓縮和未壓縮的檔案載入資料](#)
- [將檔案上傳至 Amazon S3 以搭配使用 COPY](#)
- [使用 COPY 命令從 Amazon S3 載入](#)

從已壓縮和未壓縮的檔案載入資料

載入壓縮的資料時，建議您將每個資料表的資料分割為多個檔案。當您載入未壓縮的分隔資料時，COPY命令會使用大規模平行處理（MPP）和掃描範圍，從 Amazon S3 儲存貯體中的大型檔案載入資料。

從多個壓縮的檔案載入資料

如果您有壓縮的資料，建議您將每個資料表的資料分割為多個檔案。COPY 命令可以平行載入多個檔案的資料。您可以透過為集合指定通用字首或字首索引鍵，或在資訊清單檔案中明確列出檔案來載入多個檔案。

將您的資料分割為檔案，使得檔案的數量為您的叢集中配量數量的倍數。以該方式，Amazon Redshift 可以在配量間平均分割資料。每一節點的配量數目取決於叢集的節點大小。例如，每個 dc2.large 運算節點有兩個層，每個 dc2.8xlarge 運算節點有 16 個層。如需每個節點大小有多少配量的相關資訊，請參閱《Amazon Redshift 管理指南》中的[關於叢集和節點](#)。

節點全部會參與在執行的平行查詢中，以處理盡可能平均地分佈於配量的資料。如果您有具有兩個 dc2.large 節點的叢集，您可以將資料分割為四個檔案或四個檔案的倍數。Amazon Redshift 在分割工作負載時，不會考量檔案大小。因此，您需要確定壓縮後的檔案大小大致相同（從 1 MB 到 1 GB）。

若要使用物件字首來識別載入檔案，將每個檔案以通用字首命名。例如，venue.txt 檔案可以分割為四個檔案，如下所示。

```
venue.txt.1  
venue.txt.2  
venue.txt.3  
venue.txt.4
```

如果您將多個檔案放入儲存貯體的資料夾中，並將資料夾名稱指定為字首，會COPY載入資料夾中的所有檔案。如果您使用資訊清單檔案明確列出要載入的檔案，檔案可以位在不同的儲存貯體或資料夾中。

如需資訊清單檔案的相關資訊，請參閱[Example: COPY from Amazon S3 using a manifest](#)。

從未壓縮的已分隔檔案載入資料

當您載入未壓縮的分隔資料時，COPY命令會使用 Amazon Redshift 中的大規模平行處理（MPP）架構。Amazon Redshift 會自動使用平行運作的配量，以從 Amazon S3 儲存貯體中的大型檔案中載入資料範圍。檔案必須已分隔，才能進行平行載入。例如，以縱線字元分隔。使用 COPY命令的自動平行

資料載入也可用於 CSV 檔案。您可以藉由在資料表上設定分佈索引鍵來利用平行處理。如需分佈索引鍵的相關資訊，請參閱[查詢最佳化的資料分佈](#)。

當COPY查詢包含下列任何關鍵字時，不支援自動REMOVEQUOTES平行資料載入：ESCAPE、和FIXEDWIDTH。

來自檔案的資料會載入到目標資料表，每個資料列一行。資料檔案中的欄位會依次與資料表資料欄對應，由左至右。資料檔案中的欄位可以是固定寬度或以字元分隔；預設的分隔符號為管線 (|)。依預設，所有資料表資料欄已載入，但您可以選擇性地定義以逗點分隔的資料欄清單。如果 COPY命令中指定的資料欄清單中不包含資料表資料欄，則會載入預設值。如需詳細資訊，請參閱[載入預設的欄位值](#)。

如果資料未壓縮也未分隔時，請遵循以下一般程序從 Amazon S3 載入資料：

1. 將檔案上傳至 Amazon S3。
2. 執行COPY命令以載入資料表。
3. 驗證資料已正確載入。

如需COPY命令的範例，請參閱[COPY 範例](#)。如需載入 Amazon Redshift 的資料相關資訊，請檢查[STL_LOAD_COMMITS](#) 和 [STL_LOAD_ERRORS](#) 系統資料表。

如需有關節點及每個節點所含配量的相關資訊，請參閱《Amazon Redshift 管理指南》中的[關於叢集和節點](#)。

將檔案上傳至 Amazon S3 以搭配 使用 COPY

將文字檔案上傳到 Amazon S3 時，有幾個可採取的方法：

- 如果您有壓縮的檔案，建議您分割大型檔案以利用 Amazon Redshift 中的平行處理。
- 另一方面，COPY會自動分割大型、未壓縮、以文字分隔的檔案資料，以促進平行處理，並有效地從大型檔案分發資料。

建立 Amazon S3 儲存貯體來保存您的資料檔案，然後將資料檔案上傳至儲存貯體。如需有關建立儲存貯體及上傳檔案的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的[使用 Amazon S3 儲存貯體](#)。

Important

必須在與您叢集所在的相同 AWS 區域中建立存放資料檔案的 Amazon S3 儲存貯體，除非您使用 [REGION](#) 選項來指定 Amazon S3 儲存貯體所在的區域。

確定 S3 IP 範圍已新增至您的允許清單。若要進一步了解所需的 S3 IP 範圍，請參閱[網路隔離](#)。

您可以在使用 Amazon S3 Amazon S3主控台建立儲存貯體時選取區域，或使用 Amazon S3 API或 建立儲存貯體時指定端點，藉此在特定區域中建立 Amazon S3 儲存貯體CLI。

在資料載入之後，驗證 Amazon S3 上有正確的檔案。

主題

- [管理資料一致性](#)
- [將加密資料上傳到 Amazon S3](#)
- [驗證您的儲存貯體中存在正確的檔案](#)

管理資料一致性

Amazon S3 為所有 AWS 區域 Amazon S3 儲存貯體上的 COPY、UNLOADINSERT (外部資料表)、CREATEEXTERNALTABLEAS 和 Amazon Redshift Spectrum 操作提供強大的 read-after-write 一致性。此外，Amazon S3 Select、Amazon S3 存取控制清單、Amazon S3 物件標籤和物件中繼資料 (例如HEAD物件) 上的讀取操作非常一致。如需資料一致性的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 [Amazon S3 資料一致性模式](#)。

將加密資料上傳到 Amazon S3

Amazon S3 同時支援伺服器端加密和用戶端加密。此主題討論伺服器端和用戶端加密之間的差異，並說明將用戶端加密與 Amazon Redshift 搭配使用的步驟。伺服器端加密對 Amazon Redshift 來說是透明的。

伺服器端加密

伺服器端加密是靜態資料加密，也就是說，Amazon S3 會在上傳資料時加密您的資料，並在您存取資料時將資料解密。當您使用 COPY 命令載入資料表時，從 Amazon S3 上的伺服器端加密或未加密物件載入的方式沒有差異。如需伺服器端加密的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[使用伺服器端加密](#)。

用戶端加密

在用戶端加密中，您的用戶端應用程式會管理您的資料的加密、加密金鑰和相關工具。您可以使用用戶端加密將資料上傳至 Amazon S3 儲存貯體，然後使用 COPY 命令搭配 ENCRYPTED 選項和私有加密金鑰載入資料，以提供更高的安全性。

您可以使用封套加密來加密您的資料。利用封套加密，您的應用程式會以獨佔方式處理所有加密。您的私有加密金鑰和未加密的資料永遠不會傳送至 AWS，因此安全管理加密金鑰非常重要。如果您遺失加密金鑰，您將無法取消加密資料，也無法從 復原加密金鑰 AWS。封套加密會結合快速對稱加密的效能，同時保有非對稱金鑰所提供金鑰管理的更高安全性。Amazon S3 one-time-use 加密用戶端會產生對稱金鑰（信封對稱金鑰）來加密您的資料，然後該金鑰會由根金鑰加密，並與資料一起存放在 Amazon S3 中。Amazon Redshift 在載入期間存取您的資料時，系統會使用您的實際金鑰來擷取並解密加密的對稱金鑰，然後將資料解密。

若要在 Amazon Redshift 中使用 Amazon S3 用戶端加密資料，請參閱《Amazon Simple Storage Service 使用者指南》的 [使用用戶端加密保護資料](#) 來遵循所述步驟，並搭配您使用的其他要求：

- 對稱加密 – SDK 適用於 Java AWS AmazonS3EncryptionClient 類別的 使用前面所述的信封加密，這是以對稱金鑰加密為基礎。使用此類別來建立 Amazon S3 用戶端，以上傳用戶端加密的資料。
- 256 位元 AES 根對稱金鑰 – 根金鑰會加密信封金鑰。您可以將根金鑰傳遞至您的 AmazonS3EncryptionClient 類別執行個體。儲存此金鑰，因為您將需要用它將資料複製到 Amazon Redshift。
- 用來儲存加密信封金鑰的物件中繼資料 – 依預設，Amazon S3 會將信封金鑰儲存為 AmazonS3EncryptionClient 類別的物件中繼資料。儲存為物件中繼資料的加密信封金鑰會在解密程序期間使用。

Note

如果您 API 第一次使用加密時收到密碼加密錯誤訊息，則 版本的 Java 密碼編譯延伸模組 (JCE) 司法管轄區政策檔案 JDK 可能會將加密和解密轉換的金鑰長度上限限制為 128 位元。如需解決此問題的相關資訊，請參閱 Amazon Simple Storage Service 使用者指南 中的 [使用適用於 Java 的 AWS SDK 指定用戶端加密](#)。

如需使用 COPY 命令將用戶端加密檔案載入 Amazon Redshift 資料表的相關資訊，請參閱 [從 Amazon S3 載入加密的資料檔案](#)。

範例：上傳用戶端加密資料

如需如何使用適用於 Java 的 AWS SDK 上傳用戶端加密資料的範例，請參閱 Amazon Simple Storage Service 使用者指南 中的 [使用用戶端加密保護資料](#)。

第二個選項會顯示在用戶端加密期間要讓資料可以在 Amazon Redshift 中載入，您必須進行的選擇。具體來說，範例顯示使用物件中繼資料來存放加密的信封金鑰，以及使用 256 位元 AES 根對稱金鑰。

此範例提供範例程式碼，AWS SDK 使用 for Java 建立 256 位元 AES 對稱根金鑰，並將其儲存至檔案。然後該範例會使用可先在用戶端上先加密樣本資料的 S3 加密用戶端，將物件上傳至 Amazon S3。此範例也會下載物件，並驗證資料一致。

驗證您的儲存貯體中存在正確的檔案

將您的檔案上傳至您的 Amazon S3 儲存貯體之後，我們建議列出儲存貯體的內容，以驗證已包含所有正確的檔案，並且沒有不需要的檔案。例如，如果儲存貯體 `amzn-s3-demo-bucket` 保存名為 `venue.txt.back` 的檔案，下列命令會載入該檔案 (可能非特意)：

```
COPY venue FROM 's3://amzn-s3-demo-bucket/venue' ... ;
```

如果您想要具體控制載入的檔案，您可以使用資訊清單檔案來明確列出資料檔案。如需使用資訊清單檔案的詳細資訊，請參閱 COPY 命令 [copy_from_s3_manifest_file](#) 的選項和 [Example: COPY from Amazon S3 using a manifest](#) COPY 範例中的選項。

如需列出儲存貯體內容的相關資訊，請參閱《Amazon S3 開發人員指南》中的 [列出物件金鑰](#)。

使用 COPY 命令從 Amazon S3 載入

使用 [COPY](#) 命令從 Amazon S3 上的資料檔案平行載入資料表。您可以透過使用 Amazon S3 物件字首或使用資訊清單檔案來指定要載入的檔案。

使用字首指定要載入之檔案的語法如下所示：

```
COPY <table_name> FROM 's3://<bucket_name>/<object_prefix>'
authorization;
```

資訊清單檔案是 JSON 格式化的檔案，列出要載入的資料檔案。使用資訊清單檔案指定要載入之檔案的語法如下所示：

```
COPY <table_name> FROM 's3://<bucket_name>/<manifest_file>'
authorization
```

```
MANIFEST;
```

要載入的資料表必須已存在於資料庫中。如需建立資料表的相關資訊，請參閱 SQL 參考 [CREATE TABLE](#) 中的。

授權值提供 Amazon Redshift 存取 Amazon S3 物件所需的 AWS 授權。如需所需許可的詳細資訊，請參閱 [IAM COPY、UNLOAD 和的許可 CREATE LIBRARY](#)。偏好的身分驗證方法是指定 IAM_ROLE 參數，並為具有必要許可 IAM 的角色提供 Amazon Resource Name (ARN)。如需詳細資訊，請參閱 [角色類型存取控制](#)。

若要使用 IAM_ROLE 參數進行身分驗證，請取代 `<aws-account-id>` 以及 `<role-name>` 如下列語法所示。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

下列範例顯示使用 IAM 角色的身分驗證。

```
COPY customer
FROM 's3://amzn-s3-demo-bucket/mydata'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如需其他授權選項的相關資訊，請參閱 [授權參數](#)

如果您想要在不實際載入資料表的情況下驗證資料，請使用 NOLOAD 選項搭配 [COPY](#) 命令。

下列範例顯示名為 venue.txt 的檔案中以管線分隔資料的前幾個資料列。

```
1|Toyota Park|Bridgeview|IL|0
2|Columbus Crew Stadium|Columbus|OH|0
3|RFK Stadium|Washington|DC|0
```

將檔案上傳至 Amazon S3 之前，請將檔案分割為多個檔案，讓 COPY 命令可以使用平行處理載入檔案。檔案數量應為您叢集中的分割的倍數。分割您的載入資料，使檔案皆有相同的大小，在壓縮之後介於 1 MB 至 1 GB。如需詳細資訊，請參閱 [從已壓縮和未壓縮的檔案載入資料](#)。

例如，venue.txt 檔案可以分割為四個檔案，如下所示：

```
venue.txt.1
venue.txt.2
venue.txt.3
```

```
venue.txt.4
```

下列COPY命令會使用管道分隔資料，在 Amazon S3 儲存貯體 中具有字首「venue」的資料檔案中載入VENUE資料表amzn-s3-demo-bucket。

Note

下列範例中的 Amazon S3 儲存貯體 amzn-s3-demo-bucket 不存在。如需在現有 Amazon S3 儲存貯體中使用實際資料的範例COPY命令，請參閱[載入範例資料](#)。

```
COPY venue FROM 's3://amzn-s3-demo-bucket/venue'  
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
DELIMITER '|';
```

如果不存在具有 'venue' 金鑰前綴的任何 Amazon S3 物件，則載入會失敗。

主題

- [使用資訊清單指定資料檔案](#)
- [從 Amazon S3 載入壓縮的資料檔案](#)
- [從 Amazon S3 載入固定寬度資料](#)
- [從 Amazon S3 載入多位元組資料](#)
- [從 Amazon S3 載入加密的資料檔案](#)

使用資訊清單指定資料檔案

您可以使用資訊清單來確保COPY命令載入資料載入所需的所有檔案，以及僅載入所需的檔案。您可以使用資訊清單從不同儲存貯體載入檔案，或載入不共用相同字首的檔案。除了提供COPY命令的物件路徑之外，您還會提供 JSON格式化文字檔案的名稱，以明確列出要載入的檔案。資訊清單中URL的 必須指定檔案的儲存貯體名稱和完整物件路徑，而不只是字首。

如需資訊清單檔案的詳細資訊，請參閱COPY範例 [使用資訊清單來指定資料檔案](#)。

下列範例顯示從不同儲存貯體JSON載入檔案，以及檔案名稱開頭為日期戳記的。

```
{  
  "entries": [  
    {"url":"s3://amzn-s3-demo-bucket1/2013-10-04-custdata", "mandatory":true},
```



```

    {"url":"s3://amzn-s3-demo-bucket1/2013-10-05-custdata", "mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket2/2013-10-04-custdata", "mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket2/2013-10-05-custdata", "mandatory":true}
  ]
}

```

選用的 `mandatory` 旗標會指定如果找不到檔案，是否 COPY 應該傳回錯誤。 `mandatory` 的預設值為 `false`。無論任何強制性設定為何，如果找不到檔案， COPY 都會終止。

下列範例使用上一個名為 `customer` 的範例中的資訊清單執行 COPY 命令 `cust.manifest`。

```

COPY customer
FROM 's3://amzn-s3-demo-bucket/cust.manifest'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
MANIFEST;

```

使用 `UNLOAD` 建立的清單清單 UNLOAD

UNLOAD 操作使用 `MANIFEST` 參數建立的清單清單可能具有 COPY 操作不需要的金鑰。例如，以下 `UNLOAD` 清單檔案包含 `meta` 索引鍵，針對 Amazon Redshift Spectrum 外部資料表，還有載入 ORC 或 Parquet 檔案格式的資料檔案時，都需要此索引鍵。 `meta` 索引鍵包含的 `content_length` 索引鍵具有的值為檔案的實際大小 (以位元組為單位)。 COPY 操作只需要 `url` 金鑰和選用的 `mandatory` 金鑰。

```

{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/unload/manifest_0000_part_00", "meta":
    { "content_length": 5956875 }},
    {"url":"s3://amzn-s3-demo-bucket/unload/unload/manifest_0001_part_00", "meta":
    { "content_length": 5997091 } }
  ]
}

```

如需資訊清單檔案的相關資訊，請參閱 [Example: COPY from Amazon S3 using a manifest](#)。

從 Amazon S3 載入壓縮的資料檔案

若要載入使用 `gzip`、`lzop` 或 `bzip2` 壓縮的資料檔案，請包含對應的選項：`GZIP`、`LZOP` 或 `BZIP2`。

例如，下列命令會從使用 `lzop` 壓縮的檔案載入。

```

COPY customer FROM 's3://amzn-s3-demo-bucket/customer.lzo'

```

```
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
DELIMITER '|' LZOP;
```

Note

如果您使用 lzop 壓縮資料檔案並使用 --filter 選項，則 COPY 命令不支援該檔案。

從 Amazon S3 載入固定寬度資料

固定寬度資料檔案其資料的每個資料欄長度一致。固定寬度資料檔案中的每個欄位有完全相同的長度和位置。對於固定寬度資料檔案中的字元資料（CHAR 和 VARCHAR），您必須將前置或尾端空格作為預留位置，才能保持寬度一致。針對整數，您必須使用前方零做為預留位置。固定寬度資料檔案沒有分隔符號可分隔資料欄。

若要將固定寬度資料檔案載入現有資料表，USE 請載入 COPY 命令中的 FIXEDWIDTH 參數。您的資料表規格必須符合 fixedwidth_spec 的值，資料才能正確載入。

若要從檔案載入固定寬度資料至資料表，請發出下列命令：

```
COPY table_name FROM 's3://amzn-s3-demo-bucket/prefix'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
FIXEDWIDTH 'fixedwidth_spec';
```

fixedwidth_spec 參數是字串，包含每個資料欄的識別碼以及每個資料欄的寬度，以冒號分隔。**column:width** 對組是使用逗號分隔。識別碼可以是您選擇的任何項目：數字、字母或兩者的結合。識別碼對資料表本身沒有關聯，所有規格必須以與資料表相同的順序包含資料欄。

下列兩個範例會顯示相同的規格，第一個使用數值識別碼，而第二個使用字串識別碼：

```
'0:3,1:25,2:12,3:2,4:6'
```

```
'venueid:3,venueid:25,venueid:12,venueid:2,venueid:6'
```

下列範例顯示可使用上述規格載入 VENUE 資料表的固定寬度範例資料：

```
1 Toyota Park           Bridgeview  IL0
2 Columbus Crew Stadium Columbus    OH0
3 RFK Stadium           Washington  DC0
```

```
4 CommunityAmerica Ballpark Kansas City KS0
5 Gillette Stadium Foxborough MA68756
```

下列COPY命令會將此資料集載入VENUE資料表：

```
COPY venue
FROM 's3://amzn-s3-demo-bucket/data/venue_fw.txt'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
FIXEDWIDTH 'venueid:3,venueid:25,venueid:12,venueid:2,venueid:6';
```

從 Amazon S3 載入多位元組資料

如果您的資料包含非ASCII多位元組字元（例如中文或西里爾文字元），您必須將資料載入VARCHAR資料欄。VARCHAR資料類型支援4位元組UTF-8個字元，但CHAR資料類型只接受單位元組ASCII字元。您無法將五位元組或更長的字元載入Amazon Redshift資料表。如需CHAR和的詳細資訊VARCHAR，請參閱[資料類型](#)。

若要檢查輸入檔案使用的編碼，請使用Linux *file* 命令：

```
$ file ordersdata.txt
ordersdata.txt: ASCII English text
$ file uni_ordersdata.dat
uni_ordersdata.dat: UTF-8 Unicode text
```

從 Amazon S3 載入加密的資料檔案

您可以使用COPY命令，透過伺服器端加密、用戶端加密或兩者，載入上傳至Amazon S3的資料檔案。

此COPY命令支援下列類型的Amazon S3加密：

- 使用Amazon S3-managed伺服器端加密（SSE-S3）
- 使用AWS KMS keys（SSE-KMS）的伺服器端加密
- 使用用戶端對稱根金鑰的用戶端加密

此COPY命令不支援下列類型的Amazon S3加密：

- 使用客戶提供的金鑰進行伺服器端加密（SSE-C）
- 使用的用戶端加密AWS KMS key
- 使用客戶提供非對稱根金鑰的用戶端加密

如需 Amazon S3 加密的相關資訊，請參閱《Amazon Simple Storage Service 開發人員指南》中的[使用伺服器端加密保護資料](#)和[使用用戶端加密保護資料](#)。

UNLOAD 命令會使用 SSE-S3 自動加密檔案。您也可以使用 SSE 或 KMS 用戶端加密搭配客戶受管對稱金鑰來卸載。如需詳細資訊，請參閱 [卸載加密的資料檔案](#)

COPY 命令會自動識別並使用 SSE-S3 和 SSE- 載入加密的檔案 KMS。您可以指定 ENCRYPTED 選項並提供金鑰值，以使用用戶端對稱根金鑰載入加密的檔案。如需詳細資訊，請參閱 [將加密資料上傳到 Amazon S3](#)。

若要載入用戶端加密資料檔案，請使用 MASTER_SYMMETRIC_KEY 參數提供根金鑰值，並包含 ENCRYPTED 選項。

```
COPY customer FROM 's3://amzn-s3-demo-bucket/encrypted/customer'  
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
MASTER_SYMMETRIC_KEY '<root_key>'  
ENCRYPTED  
DELIMITER '|';
```

若要載入壓縮 gzip、lzop 或 bzip2 的加密資料檔案，請包含 GZIP、LZOP 或 BZIP2 選項，以及根金鑰值和 ENCRYPTED 選項。

```
COPY customer FROM 's3://amzn-s3-demo-bucket/encrypted/customer'  
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
MASTER_SYMMETRIC_KEY '<root_key>'  
ENCRYPTED  
DELIMITER '|'  
GZIP;
```

從 Amazon 載入資料 EMR

您可以使用 COPY 命令，從設定為將文字檔案寫入叢集的 Hadoop 分散式檔案系統（HDFS）的 Amazon EMR 叢集平行載入資料，作為固定寬度檔案、字元分隔檔案、CSV 檔案或 JSON 格式化檔案。

從 Amazon 載入資料的程序 EMR

本節將逐步引導您從 Amazon EMR 叢集載入資料的程序。下列小節提供必須完成每個步驟的詳細資訊。

- [步驟 1：設定IAM許可](#)

建立 Amazon EMR叢集並執行 Amazon Redshift COPY命令的使用者必須具有必要的許可。

- [步驟 2：建立 Amazon EMR叢集](#)

設定叢集將文字檔案輸出至 Hadoop 分散式檔案系統 (HDFS)。您需要 Amazon EMR叢集 ID 和叢集的主要公有 DNS (託管叢集的 Amazon EC2執行個體的端點)。

- [步驟 3：擷取 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址](#)

公有金鑰可讓 Amazon Redshift 叢集節點建立與主機的SSH連線。您將使用每個叢集節點的 IP 地址來設定主機安全群組，以允許使用這些 IP 地址從您的 Amazon Redshift 叢集存取。

- [步驟 4：將 Amazon Redshift 叢集公有金鑰新增至每個 Amazon EC2 主機的授權金鑰檔案](#)

您可以將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案，讓主機能夠識別 Amazon Redshift 叢集並接受SSH連線。

- [步驟 5：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址](#)

修改 Amazon EMR執行個體的安全群組，以新增輸入規則以接受 Amazon Redshift IP 地址。

- [步驟 6：執行 COPY 命令以載入資料](#)

從 Amazon Redshift 資料庫執行 COPY命令，將資料載入 Amazon Redshift 資料表。

步驟 1：設定IAM許可

建立 Amazon EMR叢集並執行 Amazon Redshift COPY命令的使用者必須具有必要的許可。

設定IAM許可

1. 為將建立 Amazon EMR叢集的使用者新增下列許可。

```
ec2:DescribeSecurityGroups
ec2:RevokeSecurityGroupIngress
ec2:AuthorizeSecurityGroupIngress
redshift:DescribeClusters
```

2. 為將執行 COPY命令IAM的角色或使用者新增下列許可。

```
elasticmapreduce:ListInstances
```

3. 將下列許可新增至 Amazon EMR叢集IAM的角色。

```
redshift:DescribeClusters
```

步驟 2：建立 Amazon EMR 叢集

COPY 命令會從 Amazon EMR Hadoop 分散式檔案系統 () 上的檔案載入資料HDFS。建立 Amazon EMR叢集時，請設定叢集將資料檔案輸出至叢集的 HDFS。

若要建立 Amazon EMR叢集

1. 在與 Amazon Redshift EMR叢集相同的 AWS 區域中建立 Amazon 叢集。

如果 Amazon Redshift 叢集位於 中VPC，Amazon EMR叢集必須位於相同的VPC群組中。如果 Amazon Redshift 叢集使用 EC2-Classic 模式 (亦即，它不在 中VPC)，Amazon EMR叢集也必須使用 EC2-Classic 模式。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的[在虛擬私有雲端 \(VPC \) 中管理叢集](#)。

2. 設定叢集將資料檔案輸出至叢集的 HDFS。HDFS 檔案名稱不得包含星號 (*) 或問號 (?)。

Important

檔案名稱不能包括星號 (*) 或問號 (?)。

3. 在 Amazon EMR叢集組態中為自動終止選項指定否，讓叢集在COPY命令執行期間保持可用狀態。

Important

如果在 COPY完成之前變更或刪除任何資料檔案，您可能會有非預期的結果，或者COPY操作可能會失敗。

4. 請注意叢集 ID 和主要公有 DNS (託管叢集的 Amazon EC2執行個體端點)。您將在稍後的步驟中使用該資訊。

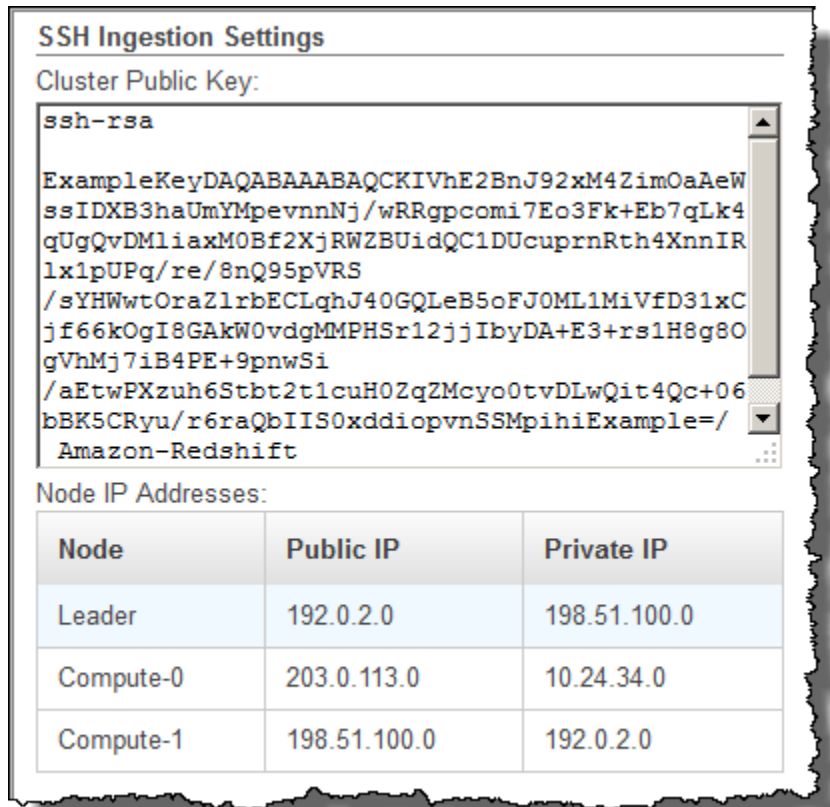
步驟 3：擷取 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址

您將使用每個叢集節點的 IP 地址來設定主機安全群組，以允許使用這些 IP 地址從您的 Amazon Redshift 叢集存取。

使用主控台擷取您叢集的 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址

1. 存取 Amazon Redshift 管理主控台。
2. 在導覽窗格中，選擇叢集連結。
3. 從清單選取您的叢集。
4. 找到SSH擷取設定群組。

記下 Cluster Public Key (叢集公有金鑰) 和 Node IP addresses (節點 IP 地址)。您將在稍後的步驟中用到它們。



SSH Ingestion Settings

Cluster Public Key:

```
ssh-rsa
ExampleKeyDAQABAAABAQCKIVhE2BnJ92xM4ZimOaAeW
ssIDXB3haUmYMpevnnNj/wRRgpcomi7Eo3Fk+Eb7qLk4
qUgQvDMliaxM0Bf2XjRWZBUidQC1DUCuprnRth4XnnIR
lx1pUPq/re/8nQ95pVRS
/sYHWwtOraZ1rbECLqhJ40GQLeB5oFJ0ML1MiVfD31xC
jf66kOgI8GAkW0vdgMMPHSr12jjIbyDA+E3+rs1H8g8O
gVhMj7iB4PE+9pnwSi
/aEtwPXzuh6Stbt2t1cuH0ZqZMcyo0tvDLwQit4Qc+06
bBK5CRyu/r6raQbIIS0xddiopvnSSMpihiExample=/
Amazon-Redshift
```

Node IP Addresses:

Node	Public IP	Private IP
Leader	192.0.2.0	198.51.100.0
Compute-0	203.0.113.0	10.24.34.0
Compute-1	198.51.100.0	192.0.2.0

您將使用步驟 3 中的私有 IP 地址，將 Amazon EC2 主機設定為接受來自 Amazon Redshift 的連線。

若要使用 Amazon Redshift 擷取叢集的叢集公有金鑰和叢集節點 IP 地址 CLI，請執行 `describe-clusters` 命令。例如：

```
aws redshift describe-clusters --cluster-identifier <cluster-identifier>
```

回應將包含 ClusterPublicKey 值和私有和公有 IP 地址清單，如下所示：

```
{
  "Clusters": [
    {
      "VpcSecurityGroups": [],
      "ClusterStatus": "available",
      "ClusterNodes": [
        {
          "PrivateIPAddress": "10.nnn.nnn.nnn",
          "NodeRole": "LEADER",
          "PublicIPAddress": "10.nnn.nnn.nnn"
        },
        {
          "PrivateIPAddress": "10.nnn.nnn.nnn",
          "NodeRole": "COMPUTE-0",
          "PublicIPAddress": "10.nnn.nnn.nnn"
        },
        {
          "PrivateIPAddress": "10.nnn.nnn.nnn",
          "NodeRole": "COMPUTE-1",
          "PublicIPAddress": "10.nnn.nnn.nnn"
        }
      ],
      "AutomatedSnapshotRetentionPeriod": 1,
      "PreferredMaintenanceWindow": "wed:05:30-wed:06:00",
      "AvailabilityZone": "us-east-1a",
      "NodeType": "dc2.large",
      "ClusterPublicKey": "ssh-rsa AAAABExamplepublickey...Y3TA1 Amazon-
Redshift",
      ...
    }
  ]
}
```

若要使用 Amazon Redshift 擷取叢集的叢集公有金鑰和叢集節點 IP 地址 API，請使用 `DescribeClusters` 動作。如需詳細資訊，請參閱 Amazon Redshift CLI 指南或 Amazon Redshift API 指南 [DescribeClusters](#) 中的 [describe-cluster](#)。

步驟 4：將 Amazon Redshift 叢集公有金鑰新增至每個 Amazon EC2 主機的授權金鑰檔案

您可以將叢集公有金鑰新增至所有 Amazon EMR 叢集節點的每個主機的授權金鑰檔案，以便主機識別 Amazon Redshift 並接受 SSH 連線。

將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

1. 使用SSH連線存取主機。

如需使用 連線至執行個體的相關資訊SSH，請參閱 Amazon EC2使用者指南 中的[連線至執行個體](#)。

2. 從主控台或CLI回應文字複製 Amazon Redshift 公有金鑰。

3. 將公有金鑰的內容複製和貼上至主機上的 /home/<ssh_username>/.ssh/authorized_keys 檔案。包括完整字串，包括字首 "ssh-rsa" 和字尾 "Amazon-Redshift"。例如：

```
ssh-rsa AAAACTP3isxgGzVWoIWpbVvRC0zYdVifMrh... uA70BnMHCaMiRdmvsD0edZD0edZ Amazon-Redshift
```

步驟 5：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址

若要允許對主機執行個體的輸入流量，請編輯安全群組，並為每個 Amazon Redshift 叢集節點新增一個輸出規則。對於類型，使用連接埠 22 上的SSHTCP通訊協定選取。針對來源，輸入您在 [步驟 3：擷取 Amazon Redshift 叢集公有金鑰和叢集節點 IP 地址](#) 中擷取的 Amazon Redshift 叢集節點私有 IP 地址。如需將規則新增至 Amazon EC2安全群組的詳細資訊，請參閱 Amazon EC2使用者指南 中的[為您的執行個體授權傳入流量](#)。

步驟 6：執行 COPY 命令以載入資料

執行 [COPY](#)命令以連線至 Amazon EMR叢集，並將資料載入 Amazon Redshift 資料表。Amazon EMR 叢集必須繼續執行，直到COPY命令完成為止。例如，請勿將叢集設定為自動終止。

Important

如果在 COPY完成之前變更或刪除任何資料檔案，您可能會有非預期的結果，或者COPY操作可能會失敗。

在 COPY命令中，指定 Amazon EMR叢集 ID 以及HDFS檔案路徑和檔案名稱。

```
COPY sales
FROM 'emr://myemrclusterid/myoutput/part*' CREDENTIALS
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

您可以在檔案名稱引數中使用萬用字元星號 (*) 和問號 (?)。例如，`part*` 會載入檔案 `part-0000`、`part-0001`，以此類推。如果您只指定資料夾名稱，會 COPY 嘗試載入資料夾中的所有檔案。

Important

如果您使用萬用字元或僅使用資料夾名稱，請確認不會載入不需要的檔案，否則 COPY 命令會失敗。例如，某些程序可能將日誌檔案寫入至輸出資料夾。

從遠端主機載入資料

您可以使用 COPY 命令從一或多個遠端主機平行載入資料，例如 Amazon EC2 執行個體或其他電腦。COPY 使用 連線至遠端主機，SSH 並在遠端主機上執行命令以產生文字輸出。

遠端主機可以是 Amazon EC2 Linux 執行個體，或是設定為接受 SSH 連線的其他 Unix 或 Linux 電腦。本指南假設您的遠端主機是 Amazon EC2 執行個體。當其他電腦上的程序不同時，指南將指出差異。

Amazon Redshift 可以連線至多個主機，也可以開啟每個主機的多個 SSH 連線。Amazon Redshift 會透過每個連線傳送一個唯一的命令，將文字輸出產生到主機的標準輸出，然後 Amazon Redshift 會像讀取文字檔一樣讀取該輸出。

開始之前

開始之前，您應該備妥下列各項：

- 您可以使用 連線到的一或多個主機機器，例如 Amazon EC2 執行個體 SSH。
- 主機上的資料來源。

您將提供 Amazon Redshift 叢集將在主機上執行以產生文字輸出的命令。叢集連線至主機後，COPY 命令會執行命令、從主機的標準輸出讀取文字，並將資料平行載入 Amazon Redshift 資料表。文字輸出必須採用 COPY 命令可以擷取的形式。如需詳細資訊，請參閱 [準備您的輸入資料](#)

- 從您的電腦存取主機。

對於 Amazon EC2 執行個體，您將使用 SSH 連線來存取主機。您將需要存取主機以將 Amazon Redshift 叢集的公有金鑰新增至主機的授權金鑰檔案。

- 執行中的 Amazon Redshift 叢集。

如需如何啟動叢集的相關資訊，請參閱《Amazon Redshift 入門指南》<https://docs.aws.amazon.com/redshift/latest/gsg/>。

載入資料程序

本節會逐步演練如何從遠端主機載入資料的程序。下列小節提供每個步驟中必須完成的詳細資訊。

- [步驟 1：擷取叢集公有金鑰和叢集節點 IP 地址](#)

公有金鑰可讓 Amazon Redshift 叢集節點建立與遠端主機的SSH連線。您將使用每個叢集節點的 IP 地址來設定主機安全群組或防火牆，以允許使用這些 IP 地址從您的 Amazon Redshift 叢集存取。

- [步驟 2:將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案](#)

您可以將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案，讓主機能夠識別 Amazon Redshift 叢集並接受SSH連線。

- [步驟 3：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址](#)

對於 Amazon EC2，修改執行個體的安全群組以新增輸入規則，以接受 Amazon Redshift IP 地址。對於其他主機，請修改防火牆，以便您的 Amazon Redshift 節點能夠建立與遠端主機的SSH連線。

- [步驟 4：取得主機的公有金鑰](#)

您可以選擇性地指定 Amazon Redshift 應該使用公有金鑰來識別主機。您必須找到公有金鑰，並將文字複製到您的資訊清單檔案。

- [步驟 5：建立資訊清單檔案](#)

資訊清單是 JSON格式化的文字檔案，其中包含 Amazon Redshift 連線到主機和擷取資料所需的詳細資訊。

- [步驟 6：上傳資訊清單檔案至 Amazon S3 儲存貯體](#)

Amazon Redshift 會讀取清單檔案，並使用該資訊來連線至遠端主機。如果 Amazon S3 儲存貯體不是位於與 Amazon Redshift 叢集所在的同一區域，您必須使用 [REGION](#) 選項來指定資料所在的區域。

- [步驟 7：執行 COPY命令以載入資料](#)

從 Amazon Redshift 資料庫執行 COPY命令，將資料載入 Amazon Redshift 資料表。

步驟 1：擷取叢集公有金鑰和叢集節點 IP 地址

您將使用每個叢集節點的 IP 地址來設定主機安全群組，以允許使用這些 IP 地址從您的 Amazon Redshift 叢集存取。

使用主控台擷取您的叢集的叢集公有金鑰和叢集節點 IP 地址

1. 存取 Amazon Redshift 管理主控台。
2. 在導覽窗格中，選擇叢集連結。
3. 從清單選取您的叢集。
4. 找到SSH擷取設定群組。

記下 Cluster Public Key (叢集公有金鑰) 和 Node IP addresses (節點 IP 地址)。您將在稍後的步驟中用到它們。

SSH Ingestion Settings

Cluster Public Key:

```
ssh-rsa
ExampleKeyDAQABAAAABAQCKIVhE2BnJ92xM4ZimOaAeW
ssIDXB3haUmYMpevnnNj/wRRgpcomi7Eo3Fk+Eb7qLk4
qUgQvDMLiaxM0Bf2XjRWZBUidQC1DUcuprnRth4XnnIR
lx1pUPq/re/8nQ95pVRS
/sYHWwtOraZ1rbECLqhJ40GQLeB5oFJ0ML1MiVfD31xC
jf66kOgI8GakW0vdgMMPHSr12jjIbyDA+E3+rs1H8g8O
gVhMj7iB4PE+9pnwSi
/aEtwPXzuh6Stbt2t1cuH0ZqZMcyo0tvDLwQit4Qc+06
bBK5CRyu/r6raQbIIS0xddiopvnSSMpihiExample=/
Amazon-Redshift
```

Node IP Addresses:

Node	Public IP	Private IP
Leader	192.0.2.0	198.51.100.0
Compute-0	203.0.113.0	10.24.34.0
Compute-1	198.51.100.0	192.0.2.0

您將使用步驟 3 的 IP 地址來設定主機以接受來自 Amazon Redshift 的連線。取決於您連線的主機類型，以及其是否位於中VPC，您將使用公有 IP 地址或私有 IP 地址。

若要使用 Amazon Redshift 擷取叢集的叢集公有金鑰和叢集節點 IP 地址CLI，請執行 `describe-clusters` 命令。

例如：

```
aws redshift describe-clusters --cluster-identifier <cluster-identifier>
```

回應將包含 ClusterPublicKey 和私有和公有 IP 地址清單，如下所示：

```
{
  "Clusters": [
    {
      "VpcSecurityGroups": [],
      "ClusterStatus": "available",
      "ClusterNodes": [
        {
          "PrivateIPAddress": "10.nnn.nnn.nnn",
          "NodeRole": "LEADER",
          "PublicIPAddress": "10.nnn.nnn.nnn"
        },
        {
          "PrivateIPAddress": "10.nnn.nnn.nnn",
          "NodeRole": "COMPUTE-0",
          "PublicIPAddress": "10.nnn.nnn.nnn"
        },
        {
          "PrivateIPAddress": "10.nnn.nnn.nnn",
          "NodeRole": "COMPUTE-1",
          "PublicIPAddress": "10.nnn.nnn.nnn"
        }
      ],
      "AutomatedSnapshotRetentionPeriod": 1,
      "PreferredMaintenanceWindow": "wed:05:30-wed:06:00",
      "AvailabilityZone": "us-east-1a",
      "NodeType": "dc2.large",
      "ClusterPublicKey": "ssh-rsa AAAABExamplepublickey...Y3TA1 Amazon-
Redshift",
      ...
    }
  ]
}
```

若要使用 Amazon Redshift 擷取叢集的叢集公有金鑰和叢集節點 IP 地址 API，請使用 DescribeClusters 動作。如需詳細資訊，請參閱 Amazon Redshift CLI 指南或 Amazon Redshift API 指南 [DescribeClusters](#) 中的 [describe-cluster](#)。

步驟 2: 將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

您可以將叢集公有金鑰新增至每個主機的授權金鑰檔案，讓主機識別 Amazon Redshift 並接受 SSH 連線。

將 Amazon Redshift 叢集公有金鑰新增至主機的授權金鑰檔案

1. 使用SSH連線存取主機。

如需使用 連線至執行個體的相關資訊SSH，請參閱 Amazon EC2使用者指南 中的[連線至執行個體](#)。

2. 從主控台或CLI回應文字複製 Amazon Redshift 公有金鑰。

3. 將公有金鑰的內容複製和貼上至遠端主機上的 /home/<ssh_username>/.ssh/authorized_keys 檔案。<ssh_username> 必須符合資訊清單檔案中 "username" 欄位的值。包括完整字串，包括字首 "ssh-rsa" 和字尾 "Amazon-Redshift"。例如：

```
ssh-rsa AAAACTP3isxgGzVWoIWpbVvRC0zYdVifMrh... uA70BnMHCaMiRdmvsD0edZD0edZ Amazon-Redshift
```

步驟 3：設定主機以接受所有 Amazon Redshift 叢集的 IP 地址

如果您使用 Amazon EC2執行個體或 Amazon EMR叢集，請將傳入規則新增至主機的安全群組，以允許來自每個 Amazon Redshift 叢集節點的流量。對於類型，使用連接埠 22 上的SSHTCP通訊協定選取。針對來源，輸入您在 [步驟 1：擷取叢集公有金鑰和叢集節點 IP 地址](#) 中擷取的 Amazon Redshift 叢集節點 IP 地址。如需將規則新增至 Amazon EC2安全群組的資訊，請參閱 Amazon EC2使用者指南中的[授權執行個體的傳入流量](#)。

使用私有 IP 地址的時機：

- 您的 Amazon Redshift 叢集不在虛擬私有雲端（VPC）中，而 Amazon EC2 -Classic 執行個體，兩者都在同一 AWS 區域。
- 您有位於的 Amazon Redshift 叢集VPC，以及位於相同 AWS 區域的 Amazon EC2 VPC執行個體 VPC。

否則，請使用公有 IP 地址。

如需在 中使用 Amazon Redshift 的詳細資訊VPC，請參閱 Amazon Redshift 管理指南 中的在[虛擬私有雲端（VPC）中管理叢集](#)。

步驟 4：取得主機的公有金鑰

您可以選擇性地在資訊清單檔案中提供主機的公有金鑰，使得 Amazon Redshift 可以識別主機。COPY 命令不需要主機公有金鑰，但基於安全考量，強烈建議使用公有金鑰來協助防止 'man-in-the-middle' 攻擊。

您可以在下列位置中尋找主機的公有金鑰，其中的 `<ssh_host_rsa_key_name>` 為主機公有金鑰的唯一名稱：

```
: /etc/ssh/<ssh_host_rsa_key_name>.pub
```

Note

Amazon Redshift 僅支援RSA金鑰。我們不支援DSA金鑰。

在步驟 5 建立您的資訊清單檔案時，您會將公有金鑰的文字貼上至資訊清單檔案項目中的「公有金鑰」資料欄。

步驟 5：建立資訊清單檔案

COPY 命令可以使用連線至多個主機SSH，並可以建立與每個主機的多個SSH連線。COPY 會透過每個主機連線執行命令，然後將命令的輸出平行載入資料表。資訊清單檔案是 Amazon Redshift 用來連線至主機JSON的文字檔案。資訊清單檔案會指定SSH主機端點和在主機上執行的命令，以將資料傳回 Amazon Redshift。您可以選擇在每個項目中包含主機公有金鑰、登入使用者名稱及 mandatory 旗標。

在本機電腦上建立資訊清單檔案。在稍後的步驟中，您會將檔案上傳到 Amazon S3。

資訊清單檔案的格式如下所示：

```
{
  "entries": [
    {
      "endpoint": "<ssh_endpoint_or_IP>",
      "command": "<remote_command>",
      "mandatory": true,
      "publickey": "<public_key>",
      "username": "<host_user_name>"
    },
    {
      "endpoint": "<ssh_endpoint_or_IP>",
      "command": "<remote_command>",
```

```
    "mandatory": true,  
    "publickey": "<public_key>",  
    "username": "host_user_name"}  
  ]  
}
```

資訊清單檔案包含每個SSH連線的一個「輸入」建構。每個項目代表單一SSH連線。您可以對單一主機建立多個連線，也可以對多台主機建立多個連線。如上所示，欄位名稱和值都需要雙引號。不需要雙引號的唯一值為強制欄位的布林值 **true** 或 **false**。

下列描述資訊清單檔案中的欄位。

端點

主機URL的地址或 IP 地址。例如，"ec2-111-222-333.compute-1.amazonaws.com" 或 "22.33.44.56"

command

主機將執行的命令，以產生文字或二進位 (gzip、lzop 或 bzip2) 輸出。命令可以是使用者 "host_user_name" 有許可執行的任何命令。命令可能只是列印檔案這麼簡單，也可能查詢資料庫或啟動指令碼。輸出（文字檔案、gzip 二進位檔案、lzop 二進位檔案或 bzip2 二進位檔案）必須採用 Amazon Redshift COPY命令可以擷取的形式。如需詳細資訊，請參閱[準備您的輸入資料](#)。

publickey

(選用) 主機的公有金鑰。如果提供公有金鑰，Amazon Redshift 會使用此金鑰來識別主機。如果未提供公有金鑰，Amazon Redshift 不會嘗試識別主機。例如，如果遠端主機的公有金鑰是：ssh-rsa AbcCbaxxx...xxxDHKJ root@amazon.com，請在公有金鑰欄位中輸入下列文字：AbcCbaxxx...xxxDHKJ。

mandatory

(選用) 指示如果連線失敗，COPY命令是否應該失敗。預設值為 false。如果 Amazon Redshift 未成功建立至少一個連線，則COPY命令會失敗。

使用者名稱

(選用) 用來登入主機系統並執行遠端命令的使用者名稱。使用者登入名稱與步驟 2 中用來將公有金鑰新增至主機授權金鑰檔案的登入必須相同。預設的使用者名稱為 "redshift"。

下列範例顯示完整的清單檔案，可開啟對相同主機的四個連線，並透過每個連線執行不同的命令：


```
{
  "entries": [
    {"endpoint": "ec2-184-72-204-112.compute-1.amazonaws.com",
      "command": "cat loaddata1.txt",
      "mandatory": true,
      "publickey": "ec2publickeyportionoftheec2keypair",
      "username": "ec2-user"},
    {"endpoint": "ec2-184-72-204-112.compute-1.amazonaws.com",
      "command": "cat loaddata2.txt",
      "mandatory": true,
      "publickey": "ec2publickeyportionoftheec2keypair",
      "username": "ec2-user"},
    {"endpoint": "ec2-184-72-204-112.compute-1.amazonaws.com",
      "command": "cat loaddata3.txt",
      "mandatory": true,
      "publickey": "ec2publickeyportionoftheec2keypair",
      "username": "ec2-user"},
    {"endpoint": "ec2-184-72-204-112.compute-1.amazonaws.com",
      "command": "cat loaddata4.txt",
      "mandatory": true,
      "publickey": "ec2publickeyportionoftheec2keypair",
      "username": "ec2-user"}
  ]
}
```

步驟 6：上傳資訊清單檔案至 Amazon S3 儲存貯體

上傳資訊清單檔案至 Amazon S3 儲存貯體。如果 Amazon S3 儲存貯體與 Amazon Redshift 叢集不位於同一個 AWS 區域，您必須使用 [REGION](#) 選項來指定資訊清單所在的 AWS 區域。如需有關建立 Amazon S3 儲存貯體及上傳檔案的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》<https://docs.aws.amazon.com/AWSAmazonS3/latest/gsg/>。

步驟 7：執行 COPY 命令以載入資料

執行 [COPY](#) 命令以連線至主機，並將資料載入至 Amazon Redshift 資料表。在 COPY 命令中，指定資訊清單檔案的明確 Amazon S3 物件路徑，並包含 SSH 選項。例如

```
COPY sales
FROM 's3://amzn-s3-demo-bucket/ssh_manifest'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
DELIMITER '|'
SSH;
```

Note

如果您使用自動壓縮，COPY命令會執行兩次資料讀取，這表示它會執行遠端命令兩次。第一次讀取會提供樣本進行壓縮分析，第二次讀取就實際載入資料。如果執行遠端命令兩次可能因為潛在的副作用造成問題，您應該關閉自動壓縮。若要關閉自動壓縮，請使用COMPUPDATE選項設定為來執行COPY命令OFF。如需詳細資訊，請參閱[利用自動壓縮載入資料表](#)。

從 Amazon DynamoDB 資料表載入資料

您可以使用COPY命令，從單一 Amazon DynamoDB 資料表載入具有資料的資料表。

Important

除非您使用 [REGION](#) 選項指定 Amazon DynamoDB 資料表所在的 AWS 區域，否則必須在與叢集相同的 AWS 區域中建立提供資料的 Amazon DynamoDB 資料表。

此COPY命令使用 Amazon Redshift 大規模平行處理（MPP）架構，從 Amazon DynamoDB 資料表平行讀取和載入資料。您可以藉由在 Amazon Redshift 資料表上設定配送樣式來善加利用平行處理。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

Important

當COPY命令從 Amazon DynamoDB 資料表讀取資料時，產生的資料傳輸是該資料表佈建輸送量的一部分。

若要避免耗用過量的佈建的讀取輸送量，建議您不要從位在生產環境中的 Amazon DynamoDB 資料表載入資料。如果您確實從生產資料表載入資料，建議您將 READRATIO 選項設定為遠低於未使用佈建輸送量的平均百分比。低READRATIO設定有助於將限流問題降到最低。若要使用 Amazon DynamoDB 資料表的完整佈建輸送量，請將 READRATIO 設為 100。

COPY 命令使用下列規則，將 DynamoDB 資料表擷取的項目中的屬性名稱與現有 Amazon Redshift 資料表中的資料欄名稱相符：

- Amazon Redshift 資料表資料欄與 Amazon DynamoDB 項目屬性為不區分大小寫的比對。如果 DynamoDB 資料表中的項目包含多個屬性，這些屬性只有在價格和等情況下才會不同PRICE，則 COPY 命令將會失敗。

- 不符合 Amazon DynamoDB 資料表中屬性的 Amazon Redshift 資料表資料欄會載入為 NULL 或空白，具體取決於 [COPY](#) 命令中以 EMPTYASNULL 選項指定的值。
- 不符合 Amazon Redshift 資料表中資料欄的 Amazon DynamoDB 屬性則會遭捨棄。屬性是在比對之前讀取，因此即使是捨棄的屬性也會使用資料表佈建的輸送量的一部分。
- 僅支援具有純量 STRING 和 NUMBER 資料類型的 Amazon DynamoDB 屬性。不支援 Amazon DynamoDB BINARY 和 SET 資料類型。如果 COPY 命令嘗試載入具有不支援資料類型的屬性，則該命令將會失敗。如果屬性與 Amazon Redshift 資料表資料欄不相符，COPY 不會嘗試載入它，也不會引發錯誤。

COPY 命令使用下列語法從 Amazon DynamoDB 資料表載入資料：

```
COPY <redshift_tablename> FROM 'dynamodb://<dynamodb_table_name>'
authorization
readratio '<integer>';
```

授權值是存取 Amazon DynamoDB 資料表所需的 AWS 憑證。如果這些憑證對應至使用者，則該使用者必須具有 SCAN 和正在載入 DESCRIBE 之 Amazon DynamoDB 資料表的許可。

授權值提供叢集存取 Amazon DynamoDB 資料表所需的 AWS 授權。對於要載入 DESCRIBE 的 Amazon DynamoDB 資料表，許可必須包含 SCAN 和 [IAM COPY、UNLOAD 和的許可 CREATE LIBRARY](#)。偏好的身分驗證方法是指定 IAM_ROLE 參數，並為具有必要許可 IAM 的角色提供 Amazon Resource Name (ARN)。如需詳細資訊，請參閱 [角色類型存取控制](#)。

若要使用 IAM_ROLE 參數進行身分驗證，`<aws-account-id>` 以及 `<role-name>` 如下列語法所示。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

下列範例顯示使用 IAM 角色的身分驗證。

```
COPY favoritemovies
FROM 'dynamodb://ProductCatalog'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如需其他授權選項的相關資訊，請參閱 [授權參數](#)

如果您想要在不實際載入資料表的情況下驗證資料，請使用 NOLOAD 選項搭配 [COPY](#) 命令。

下列範例會將來自 DynamoDB FAVORITEMOVIES 資料表的資料載入資料表 my-favorite-movies-table。讀取活動可能耗用最多 50% 的佈建輸送量。

```
COPY favoritemovies FROM 'dynamodb://my-favorite-movies-table'  
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
READRATIO 50;
```

為了最大限度地提高輸送量，COPY命令會在叢集中的運算節點之間平行載入來自 Amazon DynamoDB 資料表的資料。

具備自動壓縮的佈建輸送量

根據預設，每當您指定沒有壓縮編碼的空白目標資料表時，COPY命令都會套用自動壓縮。自動壓縮分析最初會從 Amazon DynamoDB 資料表取樣大量資料列。樣本數是以 COMPROWS 參數的值為基礎。預設值為每個配量 100,000 個資料列。

取樣之後，會捨棄樣本資料列並載入整個資料表。因此，許多資料列會讀取兩次。如需自動壓縮運作方式的相關資訊，請參閱[利用自動壓縮載入資料表](#)。

Important

當COPY命令從 Amazon DynamoDB 資料表讀取資料，包括用於取樣的資料列時，產生的資料傳輸是該資料表佈建輸送量的一部分。

從 Amazon DynamoDB 載入多位元組資料

如果您的資料包含非ASCII多位元組字元（例如中文或西里爾文字元），您必須將資料載入VARCHAR資料欄。VARCHAR 資料類型支援 4 位元組 UTF-8 個字元，但CHAR資料類型只接受單位元組ASCII字元。您無法將五位元組或更長的字元載入 Amazon Redshift 資料表。如需 CHAR和 的詳細資訊 VARCHAR，請參閱[資料類型](#)。

驗證資料已正確載入

在載入操作完成後，查詢 [STL_LOAD_COMMITS](#) 系統資料表，確認預期的檔案已經載入。在同一交易中執行COPY命令和負載驗證，以便在負載發生問題時復原整個交易。

下列查詢會傳回項目，以載入TICKET資料庫中的資料表：

```
SELECT query, trim(filename) AS filename, curtime, status
```

```
FROM stl_load_commits
WHERE filename like '%tickit%' order by query;
```

query	filename	curtime	status
22475	tickit/allusers_pipe.txt	2013-02-08 20:58:23.274186	1
22478	tickit/venue_pipe.txt	2013-02-08 20:58:25.070604	1
22480	tickit/category_pipe.txt	2013-02-08 20:58:27.333472	1
22482	tickit/date2008_pipe.txt	2013-02-08 20:58:28.608305	1
22485	tickit/allevvents_pipe.txt	2013-02-08 20:58:29.99489	1
22487	tickit/listings_pipe.txt	2013-02-08 20:58:37.632939	1
22489	tickit/sales_tab.txt	2013-02-08 20:58:37.632939	1

(6 rows)

驗證輸入資料

若要在實際載入資料之前驗證 Amazon S3 輸入檔案或 Amazon DynamoDB 資料表中的資料，請使用 NOLOAD 選項搭配 [COPY](#) 命令。NOLOAD 使用 搭配您用來載入資料的相同 COPY 命令和選項。NOLOAD 會檢查所有資料的完整性，而不會將其載入資料庫。如果您嘗試載入資料，NOLOAD 此選項會顯示發生的任何錯誤。

例如，如果您指定不正確的 Amazon S3 路徑作為輸入檔案，Amazon Redshift 會顯示下列錯誤。

```
ERROR: No such file or directory
DETAIL:
-----
Amazon Redshift error: The specified key does not exist
code:          2
context:       S3 key being read :
location:      step_scan.cpp:1883
process:       xenmaster [pid=22199]
-----
```

若要對錯誤訊息進行故障診斷，請參閱 [載入錯誤參考](#)。

如需使用 NOLOAD 選項的範例，請參閱 [COPY 具有 NOLOAD 選項的命令](#)。

利用自動壓縮載入資料表

您可以根據自己對資料的評估，將壓縮編碼手動套用至資料表中的資料欄。或者，您可以使用 COPY 命令搭配 COMPUPDATE 設定為 ON，根據範例資料自動分析和套用壓縮。

您可以在建立和載入全新資料表時使用自動壓縮。COPY 命令會執行壓縮分析。藉由在已填入的資料表上執行 [ANALYZE COMPRESSION](#) 命令，您也可以執行壓縮分析，而不載入資料或變更資料表上的壓縮。例如，當您想要分析資料表上的壓縮以備日後使用ANALYZECOMPRESSION時，您可以執行 `ANALYZE COMPRESSION`，同時保留現有的資料定義語言（DDL）陳述式。

選擇壓縮編碼時，自動壓縮會平衡整體效能。如果排序索引鍵欄位的壓縮比相同查詢中的其他欄位高出許多，限制範圍的掃描執行效果可能較差。因此，自動壓縮將跳過排序索引鍵資料欄上的資料分析階段，並保留使用者定義的編碼類型。

如果您尚未明確定義RAW編碼類型，則自動壓縮會選擇編碼。ANALYZE COMPRESSION 行為相同。為了獲得最佳查詢效能，請考慮使用 RAW 進行排序索引鍵。

自動壓縮的運作方式

當 COMPUPDATE 參數開啟時，每當您使用空白目標資料表執行COPY命令，且所有資料表資料欄都具有RAW編碼或沒有編碼時，COPY命令都會套用自動壓縮。

若要將自動壓縮套用至空資料表，無論其目前的壓縮編碼為何，請在將 COMPUPDATE 選項設定為 ON 的情況下執行 COPY 命令。若要關閉自動壓縮，請使用 COMPUPDATE選項設定為 OFF 來執行 COPY 命令OFF。

您不可以套用自動壓縮至已包含資料的資料表。

Note

自動壓縮分析需要在載入資料中有足夠的資料列 (每個配量至少 100,000 資料列)，才能產生有意義的樣本。

自動壓縮會隨著載入交易在背景中執行這些操作：

1. 從輸入檔案載入初始的資料列樣本。樣本數是以 COMPROWS 參數的值為基礎。預設為 100,000。
2. 為每個資料欄選擇壓縮選項。
3. 樣本資料列從資料表移除。
4. 以所選壓縮編碼重新建立資料表。
5. 使用新編碼載入和壓縮整個輸入檔案。

執行 COPY 命令後，資料表會完全載入、壓縮，並可供使用。如果您稍後載入更多資料，附加的資料列會根據現有編碼壓縮。

如果您只想要執行壓縮分析，請執行 ANALYZE COMPRESSION，這比執行完整 COPY 更有效率。然後您可以評估結果來決定是否使用自動壓縮或手動重新建立資料表。

只有 COPY 命令支援自動壓縮。另外，您可以在建立資料表時手動套用壓縮編碼。如需手動壓縮編碼的詳細資訊，請參閱 [資料欄壓縮以減少儲存資料的大小](#)。

自動壓縮範例

在此範例中，假設 TICKIT 資料庫包含名為 LISTING 資料表複本 BIGLIST，而且您希望在載入大約 300 萬列時，自動壓縮此資料表。

載入和自動壓縮資料表

1. 確保資料表是空白的。您僅可以套用自動壓縮至空白資料表：

```
TRUNCATE biglist;
```

2. 使用單一 COPY 命令載入資料表。雖然資料表是空白的，可能已指定部分較舊的編碼。若要協助 Amazon Redshift 執行壓縮分析，請將 COMPUPDATE 參數設定為 ON。

```
COPY biglist FROM 's3://amzn-s3-demo-bucket/biglist.txt'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
DELIMITER '|' COMPUPDATE ON;
```

由於未指定 COMPROWS 任何選項，因此會使用每個切片 100,000 列的預設和建議樣本數。

3. 查看 BIGLIST 資料表的新結構描述，以檢閱自動選擇的編碼結構描述。

```
SELECT "column", type, encoding
from pg_table_def where tablename = 'biglist';
```

Column	Type	Encoding
listid	integer	az64
sellerid	integer	az64
eventid	integer	az64
dateid	smallint	none
numtickets	smallint	az64
priceperticket	numeric(8,2)	az64
totalprice	numeric(8,2)	az64
listtime	timestamp without time zone	az64

4. 驗證已載入預期的資料列數量：

```
select count(*) from biglist;
```

```
count
```

```
-----
```

```
3079952
```

```
(1 row)
```

當資料列稍後使用 COPY 或 INSERT 陳述式附加到此表格時，會套用相同的壓縮編碼。

優化用於縮小資料表的儲存

如果您有資料欄非常少但資料列數量非常大的資料表，則三個隱藏的中繼資料身分資料欄（INSERT_XID、DELETE_XID、ROW_ID）將耗用資料表不成比例的磁碟空間。

為了最佳化隱藏資料欄的壓縮，請盡可能在單一 COPY 交易中載入資料表。如果您載入具有多個不同 COPY 命令的資料表，INSERT_XID 欄將無法良好壓縮。如果您使用多個 COPY 命令，則必須執行真空操作，但不會改善 INSERT_ 的壓縮 XID。

載入預設的欄位值

您可以選擇性地在 COPY 命令中定義資料欄清單。如果資料表中的資料欄從資料欄清單中省略，COPY 會使用 CREATE TABLE 命令中指定的 DEFAULT 選項提供的值載入資料欄，NULL 或者如果未指定 DEFAULT 選項，則使用 載入資料欄。

如果 COPY 嘗試 NULL 將指派給定義為 NOT 的欄 NULL，則 COPY 命令會失敗。如需指派 DEFAULT 選項的相關資訊，請參閱 [CREATE TABLE](#)。

從 Amazon S3 上的資料檔案載入時，資料欄清單中的資料欄必須採用與資料檔案中的欄位相同的順序。如果資料檔案中的欄位在資料欄清單中沒有對應的資料欄，則 COPY 命令會失敗。

從 Amazon DynamoDB 資料表載入時，順序並不重要。Amazon DynamoDB 屬性中不符合 Amazon Redshift 資料表中資料欄的所有欄位則會遭捨棄。

使用 COPY 命令將 DEFAULT 值載入資料表時，適用下列限制：

- 如果 [IDENTITY](#) 資料欄包含在資料欄清單中，也必須在 [COPY](#) 命令中指定 EXPLICIT_IDS 選項，否則 COPY 命令將會失敗。同樣地，如果從 IDENTITY 資料欄清單中省略資料欄，並指定 EXPLICIT_IDS 選項，則 COPY 操作將會失敗。

- 由於所有載入的資料列評估的DEFAULT表達式相同，因此使用 RANDOM () 函數的DEFAULT表達式會將相同的值指派給所有的資料列。
- DEFAULT 包含 CURRENT_DATE 或 的表達式SYSDATE會設定為目前交易的時間戳記。

如需範例，請參閱[COPY 範例](#)中的「從具有預設值的檔案載入資料」。

針對資料載入進行故障診斷

當您將資料載入 Amazon Redshift 資料表時，您可能會遇到來自 Amazon S3 的錯誤、無效的輸入資料和COPY命令錯誤。下列各節提供有關識別和解決資料載入錯誤的資訊。

主題

- [S3ServiceException 錯誤](#)
- [用於對資料載入進行故障診斷的系統資料表](#)
- [多位元組字元載入錯誤](#)
- [載入錯誤參考](#)

S3ServiceException 錯誤

最常見的 s3ServiceException errors 是由於憑證字串格式不正確或不正確、叢集和儲存貯體位於不同 AWS 區域，以及 Amazon S3 許可不足所導致。

本節提供每個類型的錯誤的故障診斷資訊。

無效的登入資料字串

如果您的登入資料字串的格式不正確，您會收到下列錯誤訊息：

```
ERROR: Invalid credentials. Must be of the format: credentials
'aws_access_key_id=<access-key-id>;aws_secret_access_key=<secret-access-key>
[;token=<temporary-session-token>]'
```

驗證登入資料字串不包含任何空格或換行符號，並且以單引號括住。

無效的存取金鑰 ID

如果您的存取金鑰 ID 不存在，您會收到下列錯誤訊息：

```
[Amazon](500310) Invalid operation: S3ServiceException:The AWS Access Key Id you provided does not exist in our records.
```

這通常是複製和貼上錯誤。驗證已正確輸入存取金鑰 ID。另外，如果您使用的是暫時工作階段金鑰，請檢查 token 的值是否已經設定。

無效的私密存取金鑰

如果您的私密存取金鑰不正確，您會收到下列錯誤訊息：

```
[Amazon](500310) Invalid operation: S3ServiceException:The request signature we calculated does not match the signature you provided.  
Check your key and signing method.,Status 403,Error SignatureDoesNotMatch
```

這通常是複製和貼上錯誤。驗證已正確輸入私密存取金鑰，並且它是存取金鑰 ID 的正確金鑰。

儲存貯體位於不同區域

COPY 命令中指定的 Amazon S3 儲存貯體必須與叢集位於相同的 AWS 區域。如果您的 Amazon S3 儲存貯體和您的叢集位於不同的區域，您會收到類似以下的錯誤：

```
ERROR: S3ServiceException:The bucket you are attempting to access must be addressed using the specified endpoint.
```

您可以使用 Amazon S3 管理主控台在建立儲存貯體時選取區域，或使用 Amazon S3 API 或建立儲存貯體時指定端點，來在特定區域中建立 Amazon S3 儲存貯體 CLI。如需詳細資訊，請參閱[將檔案上傳至 Amazon S3 以搭配使用 COPY](#)。

如需 Amazon S3 區域的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[存取儲存貯體](#)。

或者，您可以使用 [REGION](#) 選項搭配 COPY 命令來指定區域。

存取遭拒

如果使用者沒有足夠的許可，您會收到下列錯誤訊息：

```
ERROR: S3ServiceException:Access Denied,Status 403,Error AccessDenied
```

一個可能的原因，是憑證識別的使用者沒有 LIST 和 Amazon S3 儲存貯體的 GET 存取權。如需其他原因的資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[針對 Amazon S3 中的拒絕存取 \(403 禁止\) 錯誤進行疑難排解](#)。

若要了解如何管理儲存貯體的使用者存取權，請參閱《Amazon Simple Storage Service 使用者指南》中的[Amazon S3 中的 Identity and Access Management](#)。

用於對資料載入進行故障診斷的系統資料表

下列 Amazon Redshift 系統資料表在對資料載入問題進行故障診斷時很實用：

- 查詢 [STL_LOAD_ERRORS](#) 以探索特定載入期間發生的錯誤。
- 查詢 [STL_FILE_SCAN](#) 以檢視特定檔案的載入時間或查看是否甚至讀取特定檔案。
- 查詢 [STL_S3CLIENT_ERROR](#) 來尋找從 Amazon S3 傳輸資料時所遇到錯誤的詳細資訊。

尋找和診斷載入錯誤

1. 建立檢視或定義可傳回載入錯誤詳細資訊的查詢。下列範例會將 STL_LOAD_ERRORS 資料表連結至 STV_TBL_PERM 資料表，以比對資料表 IDs 與實際資料表名稱。

```
create view loadview as
(select distinct tbl, trim(name) as table_name, query, starttime,
trim(filename) as input, line_number, colname, err_code,
trim(err_reason) as reason
from stl_load_errors sl, stv_tbl_perm sp
where sl.tbl = sp.id);
```

2. 將 COPY 命令中的 MAXERRORS 選項設定為足夠大的值 COPY，以便傳回有關資料的有用資訊。如果 COPY 遇到錯誤，錯誤訊息會指示您參閱 STL_LOAD_ERRORS 資料表以取得詳細資訊。
3. 查詢 LOADVIEW 檢視以查看錯誤詳細資訊。例如：

```
select * from loadview where table_name='venue';
```

```
tbl | table_name | query | starttime
-----+-----+-----+-----
100551 | venue | 20974 | 2013-01-29 19:05:58.365391
```

input	line_number	colname	err_code	reason
venue_pipe.txt	1	0	1214	Delimiter not found

4. 在輸入檔案或載入指令碼中修正問題，根據檢視傳回的資訊。要監看的一些一般載入錯誤包括：

- 資料表中的資料類型和輸入資料欄位中的值不符。
- 資料表中資料欄的數量與輸入資料中的欄位數量不符。
- 引號不符。Amazon Redshift 同時支援單引號和雙引號；不過，這些引號必須正確成對。
- 輸入檔案中日期/時間資料的格式不正確。
- Out-of-range 輸入檔案中的值（數值欄）。
- 超出其壓縮編碼限制的資料欄的獨特值數量。

多位元組字元載入錯誤

具有CHAR資料類型的欄只接受 UTF-8 個字元的單位元組，最多可接受 127 個位元組的值，或 7F 十六進位，這也是ASCII字元集。VARCHAR 資料欄接受 UTF-8 個字元，最多四個位元組。如需詳細資訊，請參閱[字元類型](#)。

如果載入資料中的資料行包含對資料行資料類型無效的字元，會COPY傳回錯誤，並在錯誤編號為 1220 的 STL_LOAD_ERRORS 系統日誌資料表中記錄資料列。ERR_REASON 欄位包含無效字元的位元組序列，以十六進位表示。

修正您的載入資料中無效字元的一個替代方式是在載入程序期間取代無效的字元。若要取代無效的 UTF-8 個字元，請使用 COPY命令指定 ACCEPTINVCHARS選項。如果設定 ACCEPTINVCHARS選項，您指定的字元會取代程式碼點。如果未設定 ACCEPTINVCHARS選項，Amazon Redshift 接受有效字元 UTF-8。如需詳細資訊，請參閱[ACCEPTINVCHARS](#)。

下列程式碼點清單有效 UTF-8，如果未設定 ACCEPTINVCHARS選項，COPY操作不會傳回錯誤。但是，這些字碼指標不是有效的字元。您可以使用 [ACCEPTINVCHARS](#) 選項，以您指定的字元取代字碼指標。這些字碼指標包括範圍從 0xFDD0 到 0xFDEF 的值和最多到 0x10FFFF 的值，並且以 FFFE 或 FFFF 結尾：

- 0xFFFFE, 0x1FFFFE, 0x2FFFFE, ..., 0xFFFFFE, 0x10FFFFE
- 0xFFFFF, 0x1FFFFF, 0x2FFFFF, ..., 0xFFFFFF, 0x10FFFFF

下列範例顯示COPY嘗試將 UTF-8 字元載入e0 a1 c7a4資料CHAR欄時的錯誤原因。

```
Multibyte character not supported for CHAR
(Hint: Try using VARCHAR). Invalid char: e0 a1 c7a4
```

如果錯誤與VARCHAR資料類型相關，則錯誤原因會包含錯誤代碼，以及無效的 UTF-8 十六進位序列。下列範例顯示COPY嘗試將 UTF-8 載入 a4 VARCHAR 欄位時的錯誤原因。

```
String contains invalid or unsupported UTF-8 codepoints.
Bad UTF-8 hex sequence: a4 (error 3)
```

下表列出VARCHAR載入錯誤的描述和建議因應措施。如果發生其中一個錯誤，請將字元取代為有效的 UTF-8 碼序列，或移除字元。

錯誤代碼	描述
1	UTF-8 位元組序列超過 支援的四位元組上限VARCHAR。
2	UTF-8 位元組序列不完整。COPY 在字串結束前找不到多位元組字元的預期連續位元組數。
3	UTF-8 個單位元組字元超出範圍。開始位元組不得為 254、255 或介於 128 和 191 (含) 之間的任何字元。
4	位元組序列中結尾位元組的值超出範圍。持續位元組必須介於 128 和 191 (含) 之間。
5	UTF-8 字元保留為替代字元。替代程式碼點 (U+D800 到 U+DFFF) 無效。
8	位元組序列超過最大 UTF-8 碼點。
9	UTF-8 位元組序列沒有相符的程式碼點。

載入錯誤參考

如果從檔案載入資料時發生任何錯誤，請查詢 [STL_LOAD_ERRORS](#) 資料表來識別錯誤，並判斷可能的說明。下表列出資料載入期間可能發生的所有錯誤代碼：

載入錯誤代碼

錯誤代碼	描述
1200	不明的剖析錯誤。聯絡支援。
1201	在輸入檔案中找不到欄位分隔符號。
1202	輸入資料的欄數超過 中定義的欄數DDL。
1203	輸入資料的欄數少於 中定義的欄數DDL。
1204	輸入資料超出資料類型可接受的範圍。
1205	日期格式無效。請參閱 DATEFORMAT 和 TIMEFORMAT字串 以取得有效的格式。
1206	時間戳記格式無效。請參閱 DATEFORMAT 和 TIMEFORMAT字串 以取得有效的格式。
1207	資料包含的值超出預期的範圍 0-9。
1208	FLOAT 資料類型格式錯誤。
1209	DECIMAL 資料類型格式錯誤。
1210	BOOLEAN 資料類型格式錯誤。
1211	輸入資料行未包含資料。
1212	找不到載入檔案。
1213	指定為 的欄位NOTNULL不包含資料。
1214	找不到分隔符號。
1215	CHAR 欄位錯誤。
1216	輸入行無效。
1217	識別欄值無效。

錯誤代碼	描述
1218	使用 NULL AS '\0' 時，包含 null 終止器（ NUL或 UTF-8 0000 ）的欄位包含一個以上的位元組。
1219	UTF-8 十六進位包含無效的數字。
1220	字串包含無效或不支援的 UTF-8 程式碼點。
1221	檔案的編碼與COPY命令中指定的不同。
1222	整數值溢出錯誤。
1223	資料類型無效。
1224	輸入資料格式不正確JSON或超級資料類型不正確。
8001	COPY 具有 MANIFEST 參數 需要 Amazon S3 物件的完整路徑。
9005	指定的結束鍵無效。

從 Amazon S3 載入具有連續檔案擷取的資料表（預覽）

這是自動複製的預先發行文件 COPY（SQL JOB），其位於預覽版本中。文件和功能會隨時變更。我們建議僅在測試環境中使用此功能，不要在生產環境中使用。公開預覽將於 2024 年 10 月 31 日結束。預覽叢集將在預覽版結束的兩週後自動移除。如需預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

Note

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。

2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。AWS 區域 列出您目前 中帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的[建立叢集](#)。
5. 選擇建立叢集按鈕以建立預覽叢集。
6. 預覽叢集可用時，請使用SQL用戶端載入和查詢資料。

您的叢集必須使用名為 preview_2023 的預覽軌道建立。使用新叢集進行測試時，不支援將叢集還原至此軌道。自動複製功能不適用於 Amazon Redshift Serverless 工作群組。

此預覽可在下列 中使用 AWS 區域：

- 美國東部 (俄亥俄) 區域 (us-east-2)
- 美國東部 (維吉尼亞北部) 區域 (us-east-1)
- 美國西部 (奧勒岡) 區域 (us-west-2)
- 亞太區域 (東京) 區域 (ap-northeast-1)
- 歐洲 (斯德哥爾摩) 區域 (eu-north-1)
- 歐洲 (愛爾蘭) 區域 (eu-west-1)

您可以使用 COPYJOB，從存放在 Amazon S3 中的檔案將資料載入 Amazon Redshift 資料表。Amazon Redshift 會偵測何時將新的 Amazon S3 檔案新增至COPY命令中指定的路徑。然後，COPY命令會自動執行，而無需建立外部資料擷取管道。Amazon Redshift 會追蹤哪些檔案已載入完成。Amazon Redshift 會決定每個COPY命令一起批次的檔案數目。您可以在系統檢視中看到產生的COPY命令。

您可以定義COPYJOB一次。之後的執行會使用相同參數。

您可以使用 CREATE、LIST、SHOWDROP、ALTER和 RUN任務的選項來管理負載操作。如需詳細資訊，請參閱[COPYJOB\(預覽\)](#)。

您可以查詢系統檢視以查看COPYJOB狀態和進度。提供的視圖如下：

- [SYS_COPY_JOB \(預覽\)](#) – 包含每個目前定義的 COPY 的資料列JOB。
- [STL_LOAD_ERRORS](#) – 包含來自COPY命令的錯誤。
- [STL_LOAD_COMMITS](#) – 包含用於對COPY命令資料載入進行故障診斷的資訊。
- [SYS_LOAD_HISTORY](#) – 包含COPY命令的詳細資訊。
- [SYS_LOAD_ERROR_DETAIL](#) – 包含COPY命令錯誤的詳細資訊。

若要取得 COPY 載入的檔案清單JOB，請執行下列取代範例 `<job_id>`:

```
SELECT job_id, job_name, data_source, copy_query,filename,status, curtime
FROM sys_copy_job copyjob
JOIN stl_load_commits loadcommit
ON copyjob.job_id = loadcommit.copy_job_id
WHERE job_id = <job_id>;
```

使用DML命令載入資料表

Amazon Redshift 支援標準資料處理語言 (DML) 命令 (UPDATE、 和 DELETE) INSERT，您可以使用這些命令來修改資料表中的資料列。您也可以使用 TRUNCATE命令來執行快速大量刪除。

Note

我們強烈鼓勵您使用 [COPY](#) 命令來載入大量資料。使用個別INSERT陳述式填入資料表可能會非常慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 INSERT INTO ... SELECT FROM 或 CREATE TABLE AS 來改善效能。如需詳細資訊，請參閱 [INSERT](#) 或 [CREATE TABLE AS](#)。

如果您插入、更新或刪除資料表中大量的資料列，相對於變更前的資料列數，請在完成時針對資料表執行 ANALYZE和 VACUUM命令。如果應用程式隨時間累積了許多小變更，建議您將 ANALYZE和 VACUUM命令排程為定期執行。如需詳細資訊，請參閱 [分析資料表](#) 和 [清空資料表](#)。

主題

- [更新和插入新資料](#)

更新和插入新資料

您可以使用 MERGE 命令，有效率地將新資料新增至現有資料表。建立臨時資料表，然後使用本節所述的其中一個方法從臨時資料表中更新目標資料表，即可執行合併操作。如需 MERGE 命令的詳細資訊，請參閱 [MERGE](#)。

[合併範例](#) 使用 Amazon Redshift 的範例資料集，稱為 TICKIT 資料集。作為先決條件，您可以遵循 [常見資料庫任務入門](#) 中的指示來設定 TICKIT 資料表和資料。有關範例資料集的更多詳細資訊，請參閱 [範例資料庫](#)。

合併方法 1：取代現有資料列

如果您要覆寫目標資料表中的所有資料欄，執行合併最快的方式是取代現有的資料列。這只會掃描目標資料表一次，透過使用內部連結來刪除將要更新的資料列。刪除資料列之後，這些資料列會透過臨時資料表中的單一插入操作被新資料列取代。

如果下列各項皆成立，請使用此方法：

- 您的目標資料表和您的臨時資料表包含相同的資料欄。
- 您想要將目標資料表資料欄中的所有資料取代為所有臨時資料表資料欄。
- 您將在合併中使用臨時資料表的所有資料列。

如果任何這些條件不適用，請使用合併方法 2：不使用 指定資料欄清單 MERGE，如下節所述。

如果您不會使用預備資料表中的所有資料列，請使用子 WHERE 句篩選 DELETE 和 INSERT 陳述式，以排除未變更的資料列。不過，如果預備資料表中的大部分資料列都不會參與合併，建議您 INSERT 以個別步驟執行 UPDATE 和 ，如本節稍後所述。

合併方法 2：指定資料欄清單而不使用 MERGE

使用此方法來更新目標資料表中的特定資料欄，而不覆寫整個資料列。此方法需要比上一個方法更長的時間，因為它需要額外的更新步驟，而且不使用 MERGE 命令。如果下列任一項成立，請使用此方法：

- 並非目標資料表中的所有資料欄都是要更新的。
- 臨時資料表中的多數資料列將不會用於更新。

主題

- [建立暫時的預備資料表](#)
- [取代現有資料列來執行合併操作](#)
- [不使用 MERGE 命令指定資料欄清單，以執行合併操作](#)
- [合併範例](#)

建立暫時的預備資料表

臨時資料表為暫時的資料表，其中保存了將用來對目標資料表進行變更的所有資料，包括更新和插入。

合併操作需要臨時資料表和目標資料表之間的聯結。若要共置聯結資料列，請將臨時資料表的分佈索引鍵設定為與目標資料表的分佈索引鍵相同的資料欄。例如，如果目標資料表使用外部索引鍵資料欄做為其分佈索引鍵，請對臨時資料表的分佈索引鍵使用相同的資料欄。如果您使用 [CREATE TABLE LIKE](#) 陳述式建立臨時資料表，臨時資料表將從父資料表繼承分佈索引鍵。如果您使用 CREATE TABLE AS 陳述式，新資料表不會繼承分佈金鑰。如需詳細資訊，請參閱 [查詢最佳化的資料分佈](#)

如果分佈索引鍵與主索引鍵不同，並且分佈索引鍵不會隨著合併操作更新，請在分佈索引鍵資料欄上新增備援聯結述詞以啟用共置聯結。例如：

```
where target.primarykey = stage.primarykey
and target.distkey = stage.distkey
```

若要驗證查詢將使用已合併聯結，請使用 執行查詢，[EXPLAIN](#) 並在所有聯結上檢查 DS_DIST_NONE。如需詳細資訊，請參閱 [評估查詢計畫](#)

取代現有資料列來執行合併操作

當您執行程序中詳述的合併操作時，請將建立和卸除臨時資料表以外的所有步驟置於單一交易中。如果有任何步驟失敗，您可以復原交易。使用單一交易也會減少認可的數目，如此可節省時間和資源。

取代現有資料列來執行合併操作

1. 建立臨時資料表，然後將它填入要合併的資料，如下列虛擬程式碼所示。

```
CREATE temp table stage (like target);

INSERT INTO stage
SELECT * FROM source
WHERE source.filter = 'filter_expression';
```

2. 使用與預備階段資料表MERGE執行內部聯結，以更新目標資料表中與預備階段資料表相符的列，然後將所有剩餘列插入與預備階段資料表不相符的目標資料表。

建議您在單一MERGE命令中執行更新和插入操作。

```
MERGE INTO target
USING stage [optional alias] on (target.primary_key = stage.primary_key)
WHEN MATCHED THEN
UPDATE SET col_name1 = stage.col_name1 , col_name2= stage.col_name2, col_name3 =
  {expr}
WHEN NOT MATCHED THEN
INSERT (col_name1 , col_name2, col_name3) VALUES (stage.col_name1, stage.col_name2,
  {expr});
```

3. 捨棄臨時資料表。

```
DROP TABLE stage;
```

不使用 MERGE 命令指定資料欄清單，以執行合併操作

當您執行情序中詳述的合併操作時，請將所有步驟置於單一交易中。如果有任何步驟失敗，您可以復原交易。使用單一交易也會減少認可的數目，如此可節省時間和資源。

指定資料欄清單來執行合併操作

1. 將整個操作放在單一交易區塊中。

```
BEGIN transaction;
...
END transaction;
```

2. 建立臨時資料表，然後將它填入要合併的資料，如下列虛擬程式碼所示。

```
create temp table stage (like target);
insert into stage
select * from source
where source.filter = 'filter_expression';
```

3. 使用內部聯結搭配臨時資料表來更新目標資料表。

- 在 UPDATE 子句中，明確列出要更新的欄。

- 使用臨時資料表來執行內部聯結。
- 如果分佈索引鍵與主索引鍵不同，並且分佈索引鍵將不會更新，請在分佈索引鍵上新增備援聯結。若要驗證查詢將使用已合併聯結，請使用 執行查詢，[EXPLAIN](#)並在所有聯結上檢查 DS_DIST_NONE。如需詳細資訊，請參閱 [評估查詢計畫](#)
- 如果您的目標資料表是依時間戳記排序，請在目標資料表上新增述詞來利用受範圍限制的掃描。如需詳細資訊，請參閱[設計查詢的 Amazon Redshift 最佳實務](#)。
- 如果您將不會在合併中使用所有資料列，請新增子句來篩選要變更的資料列。例如，在一或多個資料欄上新增不相等篩選條件，以排除未變更的資料列。
- 將更新、刪除和插入操作放置在單一交易區塊，如此一來若發生問題，各個項目都將能復原。

例如：

```
begin transaction;

update target
set col1 = stage.col1,
col2 = stage.col2,
col3 = 'expression'
from stage
where target.primarykey = stage.primarykey
and target.distkey = stage.distkey
and target.col3 > 'last_update_time'
and (target.col1 != stage.col1
or target.col2 != stage.col2
or target.col3 = 'filter_expression');
```

4. 使用內部聯結搭配目標資料表從臨時資料表刪除不需要的資料列。目標資料表中的部分資料列已符合臨時資料表中的對應資料列，而其他資料列在先前的步驟中已更新。在任一情況下，插入不需要這些資料。

```
delete from stage
using target
where stage.primarykey = target.primarykey;
```

5. 從臨時資料表插入其餘的資料列。在子VALUES句中使用您在步驟 2 的UPDATE陳述式中使用的相同資料欄清單。

```
insert into target
(select col1, col2, 'expression'
```

```
from stage);  
  
end transaction;
```

6. 捨棄臨時資料表。

```
drop table stage;
```

合併範例

下列範例會執行合併以更新SALES資料表。第一個範例使用較簡易的方法，就是從目標資料表刪除，然後從臨時資料表插入所有資料列。第二個範例需要更新目標資料表中的特定資料欄，因此包括額外的更新步驟。

[合併範例](#) 使用 Amazon Redshift 的範例資料集，稱為TICKIT資料集。作為先決條件，您可以遵循指南 [常見資料庫任務入門](#) 中的指示來設定TICKIT資料表和資料。有關範例資料集的更多詳細資訊，請參閱[範例資料庫](#)。

範例合併資料來源

本節中的範例需要包括更新和插入的樣本資料來源。針對這些範例，我們將建立名為SALES_UPDATE的資料表範例，該SALES資料表使用資料表中的資料。我們將以代表12月新銷售活動的隨機資料填入新的資料表。我們將使用SALES_UPDATE範例資料表，在下列範例中建立暫存資料表。

```
-- Create a sample table as a copy of the SALES table.  
  
create table tickit.sales_update as  
select * from tickit.sales;  
  
-- Change every fifth row to have updates.  
  
update tickit.sales_update  
set qtysold = qtysold*2,  
pricepaid = pricepaid*0.8,  
commission = commission*1.1  
where saletime > '2008-11-30'  
and mod(sellerid, 5) = 0;  
  
-- Add some new rows to have inserts.  
-- This example creates a duplicate of every fourth row.
```

```
insert into tickit.sales_update
select (salesid + 172456) as salesid, listid, sellerid, buyerid, eventid, dateid,
qtysold, pricepaid, commission, getdate() as saletime
from tickit.sales_update
where saletime > '2008-11-30'
and mod(sellerid, 4) = 0;
```

根據相符索引鍵取代現有資料列的合併範例

下列指令碼使用 SALES_UPDATE 資料表對 SALES 資料表執行合併操作，其中包含 12 月銷售活動的新資料。此範例會取代 SALES 資料表中具有更新的資料列。針對此範例，我們將會更新 qtysold 和 pricepaid 資料欄，但將 commission 和 saletime 維持不變。

```
MERGE into tickit.sales
USING tickit.sales_update sales_update
on ( sales.salesid = sales_update.salesid
and sales.listid = sales_update.listid
and sales_update.saletime > '2008-11-30'
and (sales.qtysold != sales_update.qtysold
or sales.pricepaid != sales_update.pricepaid))
WHEN MATCHED THEN
update SET qtysold = sales_update.qtysold,
pricepaid = sales_update.pricepaid
WHEN NOT MATCHED THEN
INSERT (salesid, listid, sellerid, buyerid, eventid, dateid, qtysold , pricepaid,
commission, saletime)
values (sales_update.salesid, sales_update.listid, sales_update.sellerid,
sales_update.buyerid, sales_update.eventid,
sales_update.dateid, sales_update.qtysold , sales_update.pricepaid,
sales_update.commission, sales_update.saletime);

-- Drop the staging table.
drop table tickit.sales_update;

-- Test to see that commission and salestime were not impacted.
SELECT sales.salesid, sales.commission, sales.salestime, sales_update.commission,
sales_update.salestime
FROM tickit.sales
INNER JOIN tickit.sales_update sales_update
ON
sales.salesid = sales_update.salesid
AND sales.listid = sales_update.listid
```

```
AND sales_update.saletime > '2008-11-30'  
AND (sales.commission != sales_update.commission  
OR sales.salestime != sales_update.salestime);
```

不使用 指定資料欄清單的合併範例 MERGE

下列範例會執行合併操作，以更新 SALES 12 月銷售活動的新資料。我們需要包括更新和插入的樣本資料，以及沒有變更的資料列。在此範例中，我們希望更新 QTYSOLD 和 PRICEPAID 資料欄，但保留 COMMISSION 和 SALETIME 不變。下列指令碼使用 SALES_UPDATE 資料表在 SALES 資料表上執行合併操作。

```
-- Create a staging table and populate it with rows from SALES_UPDATE for Dec  
create temp table stagesales as select * from sales_update  
where saletime > '2008-11-30';  
  
-- Start a new transaction  
begin transaction;  
  
-- Update the target table using an inner join with the staging table  
-- The join includes a redundant predicate to collocate on the distribution key -- A  
  filter on saletime enables a range-restricted scan on SALES  
  
update sales  
set qtysold = stagesales.qtysold,  
  pricepaid = stagesales.pricepaid  
from stagesales  
where sales.salesid = stagesales.salesid  
and sales.listid = stagesales.listid  
and stagesales.saletime > '2008-11-30'  
and (sales.qtysold != stagesales.qtysold  
or sales.pricepaid != stagesales.pricepaid);  
  
-- Delete matching rows from the staging table  
-- using an inner join with the target table  
  
delete from stagesales  
using sales  
where sales.salesid = stagesales.salesid  
and sales.listid = stagesales.listid;  
  
-- Insert the remaining rows from the staging table into the target table  
insert into sales  
select * from stagesales;
```



```
-- End transaction and commit
end transaction;

-- Drop the staging table
drop table stagesales;
```

執行深層複製

深層複製會使用會自動排序資料表的大量插入來重新建立和重新填入資料表。如果資料表有大型未排序的區域，深層複製較清空的速度快得多。建議您僅在可以追蹤更新時，再於深層複製操作期間進行並行更新。該程序完成後，將差異更新移到新資料表中。VACUUM 操作支援自動並行更新。

您可以選擇以下其中一個方法來建立原始資料表的複本：

- 使用原始資料表 DDL。

如果 CREATE TABLE DDL 可用，這是最快且偏好的方法。如果建立新的資料表，您可以指定所有資料表和資料欄屬性，包括主索引鍵和外部索引鍵。您可以使用 SHOWTABLE 函數尋找原始 DDL。

- 使用 CREATE TABLE LIKE。

如果原始 DDL 無法使用，您可以使用 CREATETABLELIKE 重新建立原始資料表。新的資料表會繼承父資料表的編碼、分佈索引鍵、排序索引鍵和非 null 屬性。新資料表不會繼承父資料表的主索引鍵和外部索引鍵屬性，但您可以使用 [ALTER TABLE](#) 新增它們。

- 建立暫時資料表並截斷原始資料表。

如果您必須保留父資料表的主索引鍵和外部索引鍵屬性。如果父資料表具有相依性，您可以使用 CREATE TABLE ... AS (CTAS) 來建立暫存資料表。然後截斷原始資料表並從暫時資料表填入它。

相較於使用永久資料表，使用暫時資料表可大幅改善效能，但有遺失資料的風險。暫時資料表在結束建立它所在的工作階段中時自動捨棄。TRUNCATE 即使位於交易區塊內，也會立即遞交。如果 TRUNCATE 成功，但工作階段在下列操作 INSERT 完成之前關閉，則會遺失資料。如果無法接受資料遺失，請使用永久資料表。

建立資料表副本之後，您可能必須授與新資料表的存取權。您可以使用 [GRANT](#) 來定義存取權限。若要檢視並授與資料表的所有存取權限，您必須是下列其中一個角色：

- 超級使用者。
- 您想要複製之資料表的擁有者。
- 具有查看資料表權限ACCESSSYSTEMTABLE之權限，以及所有相關許可之授予權限的使用者。

此外，您可能必須針對深層複製所在的結構描述授與使用許可。如果深層複製的結構描述與原始資料表的結構描述不同，且也不是 `public` 結構描述，則必須授與使用許可。若要檢視並授與使用權限，您必須是下列其中一個角色：

- 超級使用者。
- 可授予深層複本結構描述USAGE許可的使用者。

使用原始資料表執行深層複製 DDL

1. (選用) DDL 透過執行名為 `v_generate_tbl_ddl` 的指令碼來重新建立資料表。
2. 使用原始 `CREATE TABLE` 建立資料表的副本DDL。
3. 使用 `INSERT INTO ... SELECT` 陳述式將原始資料表中的資料填入複本。
4. 檢查舊資料表上授與的權限。您可以在 `SVV_RELATION_PRIVILEGES` 系統檢視中查看這些許可。
5. 如有必要，請對新資料表授與舊資料表的許可。
6. 針對在原始資料表中具有權限的每個群組和使用者授與使用許可。如果您的深層複製資料表位於 `public` 結構描述中，或與原始資料表位於相同的結構描述中，則不需要執行此步驟。
7. 捨棄原始資料表。
8. 使用 `ALTERTABLE` 陳述式將複本重新命名為原始資料表名稱。

下列範例會使用具SAMPLE名 `sample_copy` 的複本在SAMPLE資料表上執行深層複製。

```
--Create a copy of the original table in the sample_namespace namespace using the
original CREATE TABLE DDL.
create table sample_namespace.sample_copy ( ... );

--Populate the copy with data from the original table in the public namespace.
insert into sample_namespace.sample_copy (select * from public.sample);

--Check SVV_RELATION_PRIVILEGES for the original table's privileges.
select * from svv_relation_privileges where namespace_name = 'public' and relation_name
= 'sample' order by identity_type, identity_id, privilege_type;
```

```
--Grant the original table's privileges to the copy table.
grant DELETE on table sample_namespace.sample_copy to group group1;
grant INSERT, UPDATE on table sample_namespace.sample_copy to group group2;
grant SELECT on table sample_namespace.sample_copy to user1;
grant INSERT, SELECT, UPDATE on table sample_namespace.sample_copy to user2;

--Grant usage permission to every group and user that has privileges in the original
table.
grant USAGE on schema sample_namespace to group group1, group group2, user1, user2;

--Drop the original table.
drop table public.sample;

--Rename the copy table to match the original table's name.
alter table sample_namespace.sample_copy rename to sample;
```

若要使用 執行深層複製 CREATE TABLE LIKE

1. 使用 建立新資料表CREATETABLELIKE。
2. 使用 INSERT INTO ... SELECT陳述式，將目前資料表中的資料列複製到新資料表。
3. 檢查舊資料表上授與的權限。您可以在 SVV_RELATION_PRIVILEGES 系統檢視中查看這些許可。
4. 如有必要，請對新資料表授與舊資料表的許可。
5. 針對在原始資料表中具有權限的每個群組和使用者的使用者授與使用許可。如果您的深層複製資料表位於 public 結構描述中，或與原始資料表位於相同的結構描述中，則不需要執行此步驟。
6. 捨棄目前的資料表。
7. 使用 ALTABLE陳述式將新資料表重新命名為原始資料表名稱。

下列範例會使用 CREATE TABLE 對SAMPLE資料表執行深層複製LIKE。

```
--Create a copy of the original table in the sample_namespace namespace using CREATE
TABLE LIKE.
create table sameple_namespace.sample_copy (like public.sample);

--Populate the copy with data from the original table.
insert into sample_namespace.sample_copy (select * from public.sample);

--Check SVV_RELATION_PRIVILEGES for the original table's privileges.
```

```
select * from svv_relation_privileges where namespace_name = 'public' and relation_name
= 'sample' order by identity_type, identity_id, privilege_type;

--Grant the original table's privileges to the copy table.
grant DELETE on table sample_namespace.sample_copy to group group1;
grant INSERT, UPDATE on table sample_namespace.sample_copy to group group2;
grant SELECT on table sample_namespace.sample_copy to user1;
grant INSERT, SELECT, UPDATE on table sample_namespace.sample_copy to user2;

--Grant usage permission to every group and user that has privileges in the original
table.
grant USAGE on schema sample_namespace to group group1, group group2, user1, user2;

--Drop the original table.
drop table public.sample;

--Rename the copy table to match the original table's name.
alter table sample_namespace.sample_copy rename to sample;
```

建立暫時資料表並截斷原始資料表來執行深層複製

1. 使用 CREATE TABLE AS 建立具有原始資料表資料列的暫存資料表。
2. 截斷目前的資料表。
3. 使用 INSERT INTO ... SELECT 陳述式，將暫存資料表中的資料列複製到原始資料表。
4. 捨棄暫時資料表。

下列範例會透過建立暫存SALES資料表和截斷原始資料表，在資料表上執行深層複製。由於原始資料表仍會保留，因此您不需要對複製資料表授與許可。

```
--Create a temp table copy using CREATE TABLE AS.
create temp table salestemp as select * from sales;

--Truncate the original table.
truncate sales;

--Copy the rows from the temporary table to the original table.
insert into sales (select * from salestemp);

--Drop the temporary table.
drop table salestemp;
```

分析資料表

ANALYZE 操作會更新查詢規劃器用來選擇最佳計劃的統計中繼資料。

在大多數情況下，您不需要明確執行 ANALYZE 命令。Amazon Redshift 會監控工作負載的變更，並在背景自動更新統計資料。此外，COPY 命令會在將資料載入空白資料表時自動執行分析。

若要明確分析資料表或整個資料庫，請執行 [ANALYZE](#) 命令。

自動分析

Amazon Redshift 會持續監控資料庫，並自動在背景中執行分析操作。為了盡量減少對系統效能的影響，自動分析會在工作負載較低期間執行。

預設會啟用自動分析。若要關閉自動分析，請修改叢集的參數群組，將 `auto_analyze` 參數設為 **false**。

為了減少處理時間並提高整體系統效能，對於修改程度較小的任何資料表，Amazon Redshift 會略過自動分析。

分析操作會略過具有 up-to-date 統計資料的資料表。如果您在擷取、轉換和載入（ETL）工作流程中 ANALYZE 執行，自動分析會略過具有目前統計資料的資料表。同樣地，當自動分析更新了資料表的統計資料時，明確會 ANALYZE 略過資料表。

新資料表資料的分析

根據預設，COPY 命令會在將資料載入空白資料表 ANALYZE 後執行。無論資料表 ANALYZE 是否空白，您都可以透過設定 STATUPDATE ON 強制。如果您指定 STATUPDATE OFF，ANALYZE 則不會執行。只有資料表擁有者或超級使用者可以執行 ANALYZE 命令，或使用 STATUPDATE 設定為 ON 的 COPY 命令執行命令。

Amazon Redshift 也會分析您使用下列命令建立的新資料表：

- CREATE TABLE AS (CTAS)
- CREATE TEMP TABLE AS
- SELECT INTO

對最初載入資料後未分析的新資料表執行查詢時，Amazon Redshift 會傳回警告訊息。在後續更新或載入之後查詢資料表時不會發生警告。當您在參考尚未分析之資料表的查詢上執行 EXPLAIN 命令時，會傳回相同的警告訊息。

每當將資料新增至非空白資料表就會大幅改變資料表大小時，您可以明確更新統計資訊。您可以執行 `ANALYZE` 命令，或將 `STATUPDATE ON` 選項與 `COPY` 命令搭配使用，藉此執行此操作。若要檢視自上次以來已插入或刪除的資料列數目的詳細資訊 `ANALYZE`，請查詢 [PG_STATISTIC_INDICATOR](#) 系統目錄表。

您可以將 [ANALYZE](#) 命令的範圍指定為下列其中一項：

- 整個目前資料庫
- 單一資料表
- 單一資料表中的一或多個特定資料欄
- 可能用作查詢中述詞的資料欄

`ANALYZE` 命令會從資料表取得資料列範例、執行一些計算，並儲存產生的資料欄統計資料。依預設，Amazon Redshift 會為 `DISTKEY` 資料欄執行範例傳遞，而另一個範例傳遞則會為資料表中的所有其他資料欄執行。如果您想要為資料欄的子集產生統計資料，您可以指定以逗點分隔的資料欄清單。您可以使用 `ANALYZE PREDICATECOLUMNS` 子句執行，略過不用作述詞的資料欄。

`ANALYZE` 操作是資源密集的，因此僅在實際上需要統計資料更新的資料表和資料欄上執行它們。您不需要經常或按相同的排程分析所有資料表中的所有資料欄。如果資料本質上變更，請分析在下列項目中經常使用的資料欄：

- 排序和群組操作
- 聯結
- 查詢述詞

為了縮短處理時間並改善整體系統效能，Amazon Redshift `ANALYZE` 會略過任何變更資料列百分比比較低的資料表，如 [analyze_threshold_percent](#) 參數所決定。依預設，分析閾值是設為 10%。您可以執行 `SET` 命令來變更目前工作階段的分析閾值。

較不需要頻繁分析的欄是代表事實和指標的欄，以及從未實際查詢過的任何相關屬性，例如大型 `VARCHAR` 欄。例如，請考慮 `TICKIT` 資料庫中的 `LISTING` 資料表。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'listing';
```

```
column          |          type          | encoding | distkey | sortkey
-----+-----+-----+-----+-----
```

listid	integer	none	t	1
sellerid	integer	none	f	0
eventid	integer	mostly16	f	0
dateid	smallint	none	f	0
numtickets	smallint	mostly8	f	0
priceperticket	numeric(8,2)	bytedict	f	0
totalprice	numeric(8,2)	mostly32	f	0
listtime	timestamp with...	none	f	0

如果每天載入此資料表時有大量新記錄，則經常在查詢中使用的資料LISTID欄必須定期分析為聯結金鑰。如果 TOTALPRICE和 LISTTIME是查詢中常用的限制，您可以在每個工作日分析這些資料欄和分發金鑰。

```
analyze listing(listid, totalprice, listtime);
```

假設應用程式中的賣家和事件更為靜態，且日期IDs是指僅涵蓋兩到三年的固定天數。在此情況下，這些欄的唯一值不會大幅變更。不過，每個唯一值的執行個體數目將會穩定增加。

此外，請考慮與資料TOTALPRICE欄相比，不常查詢 NUMTICKETS和 PRICEPERTICKET量值的情況。在此情況下，您可以每週末在整資料表上執行一次 ANALYZE 命令，以更新未每天分析的五個資料欄的統計資料：

述詞欄位

做為指定資料欄清單的便利替代方案，您可以選擇僅分析可能用來作為述詞的資料列。執行查詢時，在聯結中使用的任何資料欄、篩選條件或群組依據子句會在系統目錄中標示為述詞資料欄。當您 ANALYZE使用 PREDICATECOLUMNS子句執行時，分析操作只會包含符合下列條件的資料欄：

- 此資料欄已標示為述詞資料欄。
- 資料欄是分佈索引鍵。
- 資料欄是排序索引鍵的一部分。

如果資料表的任何資料欄都未標示為述詞，即使PREDICATECOLUMNS已指定，也會ANALYZE包含所有資料欄。如果沒有資料欄已標示為述詞資料欄，可能是因為尚未查詢該資料表。

當您工作負載的查詢模式相對穩定PREDICATECOLUMNS時，您可以選擇使用。當查詢模式為可變模式時，使用時，使用經常使用不同的資料欄作為述詞，PREDICATECOLUMNS可能會暫時導致過時的統計資料。過時的統計資料可能導致次佳的查詢執行期計劃和較長的執行期。不過，下次ANALYZE使用 PREDICATE 執行時COLUMNS，會包含新的述詞欄。

若要檢視述詞資料欄的詳細資訊，請使用下列項目SQL建立名為 PREDICATE_ 的檢視COLUMNS。

```
CREATE VIEW predicate_columns AS
WITH predicate_column_info as (
SELECT ns.nspname AS schema_name, c.relname AS table_name, a.attnum as col_num,
a.attname as col_name,
CASE
WHEN 10002 = s.stakind1 THEN array_to_string(stavalues1, '|||')
WHEN 10002 = s.stakind2 THEN array_to_string(stavalues2, '|||')
WHEN 10002 = s.stakind3 THEN array_to_string(stavalues3, '|||')
WHEN 10002 = s.stakind4 THEN array_to_string(stavalues4, '|||')
ELSE NULL::varchar
END AS pred_ts
FROM pg_statistic s
JOIN pg_class c ON c.oid = s.starelid
JOIN pg_namespace ns ON c.relnamespace = ns.oid
JOIN pg_attribute a ON c.oid = a.attrelid AND a.attnum = s.staattnum)
SELECT schema_name, table_name, col_num, col_name,
pred_ts NOT LIKE '2000-01-01%' AS is_predicate,
CASE WHEN pred_ts NOT LIKE '2000-01-01%' THEN (split_part(pred_ts,
'|||',1))::timestamp ELSE NULL::timestamp END as first_predicate_use,
CASE WHEN pred_ts NOT LIKE '%|||2000-01-01%' THEN (split_part(pred_ts,
'|||',2))::timestamp ELSE NULL::timestamp END as last_analyze
FROM predicate_column_info;
```

假設您對LISTING資料表執行下列查詢。請注意，LISTID、LISTTIME和EVENTID用於聯結、篩選和依子句分組。

```
select s.buyerid,l.eventid, sum(l.totalprice)
from listing l
join sales s on l.listid = s.listid
where l.listtime > '2008-12-01'
group by l.eventid, s.buyerid;
```

當您查詢 PREDICATE_COLUMNS 檢視時，如下列範例所示，您會看到 LISTID、EVENTID和 LISTTIME標記為述詞欄。

```
select * from predicate_columns
where table_name = 'listing';
```

```
schema_name | table_name | col_num | col_name          | is_predicate |
first_predicate_use | last_analyze
```



```

-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
public      | listing    |      1 | listid      | true      | 2017-05-05
19:27:59 | 2017-05-03 18:27:41
public      | listing    |      2 | sellerid    | false     |
| 2017-05-03 18:27:41
public      | listing    |      3 | eventid     | true      | 2017-05-16
20:54:32 | 2017-05-03 18:27:41
public      | listing    |      4 | dateid      | false     |
| 2017-05-03 18:27:41
public      | listing    |      5 | numtickets  | false     |
| 2017-05-03 18:27:41
public      | listing    |      6 | priceperticket | false     |
| 2017-05-03 18:27:41
public      | listing    |      7 | totalprice  | false     |
| 2017-05-03 18:27:41
public      | listing    |      8 | listtime    | true      | 2017-05-16
20:54:32 | 2017-05-03 18:27:41

```

讓統計資料保持在最新狀態，可讓查詢規劃工具選擇最佳計劃，進而改善查詢效能。Amazon Redshift 會在背景自動重新整理統計資料，您也可以明確執行 ANALYZE 命令。如果您選擇明確執行 ANALYZE，請執行下列動作：

- 在執行查詢之前執行 ANALYZE 命令。
- 在每個常規載入或更新週期結束時，定期在資料庫上執行 ANALYZE 命令。
- 在您建立的任何新資料表以及經歷重大變更的任何現有資料表或資料欄上執行 ANALYZE 命令。
- 考慮在不同類型的資料表和資料欄的排程上執行 ANALYZE 操作，取決於它們在查詢中的使用及其變更的傾向。
- 若要節省時間和叢集資源，請在執行時使用 PREDICATECOLUMNS 子句 ANALYZE。

在將快照還原至佈建的叢集或無伺服器命名空間之後，或繼續暫停的佈建叢集之後，您不需要明確執行 ANALYZE 命令。Amazon Redshift 會在這些情況下保留系統資料表資訊，因此不需要手動 ANALYZE 命令。Amazon Redshift 將根據需求繼續執行自動分析操作。

分析操作會略過具有 up-to-date 統計資料的資料表。如果您在擷取、轉換和載入 (ETL) 工作流程中 ANALYZE 執行，自動分析會略過具有目前統計資料的資料表。同樣地，當自動分析更新了資料表的統計資料時，明確會 ANALYZE 略過資料表。

ANALYZE 命令歷史記錄

知道資料表或資料庫上上次執行ANALYZE命令的時間很有用。執行ANALYZE命令時，Amazon Redshift 會執行多個如下所示的查詢：

```
padb_fetch_sample: select * from table_name
```

查詢 STL_ANALYZE 以檢視分析操作的歷史記錄。如果 Amazon Redshift 使用自動分析來分析資料表，`is_background` 欄會設為 `t` (true)。否則會設為 `f` (false)。下列範例會加入 `STV_TBL_PERM` 以顯示資料表名稱和執行期詳細資訊。

```
select distinct a.xid, trim(t.name) as name, a.status, a.rows, a.modified_rows,
  a.starttime, a.endtime
from stl_analyze a
join stv_tbl_perm t on t.id=a.table_id
where name = 'users'
order by starttime;
```

xid	name	status	rows	modified_rows	starttime	endtime
1582	users	Full	49990	49990	2016-09-22 22:02:23	2016-09-22 22:02:28
244287	users	Full	24992	74988	2016-10-04 22:50:58	2016-10-04 22:51:01
244712	users	Full	49984	24992	2016-10-04 22:56:07	2016-10-04 22:56:07
245071	users	Skipped	49984	0	2016-10-04 22:58:17	2016-10-04 22:58:17
245439	users	Skipped	49984	1982	2016-10-04 23:00:13	2016-10-04 23:00:13

(5 rows)

或者，您可以執行更複雜的查詢，以傳回在每個包含 ANALYZE 命令的已完成交易中執行的所有陳述式：

```
select xid, to_char(starttime, 'HH24:MM:SS.MS') as starttime,
datediff(sec,starttime,endtime ) as secs, substring(text, 1, 40)
```

```

from svl_statementtext
where sequence = 0
and xid in (select xid from svl_statementtext s where s.text like 'padb_fetch_sample
%' )
order by xid desc, starttime;

```

xid	starttime	secs	substring
1338	12:04:28.511	4	Analyze date
1338	12:04:28.511	1	padb_fetch_sample: select count(*) from
1338	12:04:29.443	2	padb_fetch_sample: select * from date
1338	12:04:31.456	1	padb_fetch_sample: select * from date
1337	12:04:24.388	1	padb_fetch_sample: select count(*) from
1337	12:04:24.388	4	Analyze sales
1337	12:04:25.322	2	padb_fetch_sample: select * from sales
1337	12:04:27.363	1	padb_fetch_sample: select * from sales
...			

清空資料表

Amazon Redshift 可以自動排序和對背景中的資料表執行VACUUMDELETE操作。如要在載入或一系列的累加式更新之後清理資料表，您也可以對整個資料庫或個別資料表執行 [VACUUM](#) 命令。

Note

只有具有必要資料表權限的使用者才能有效地清除資料表。如果 VACUUM 在沒有必要的資料表許可的情況下執行，則操作已成功完成，但沒有效果。如需有效執行的有效資料表許可清單 VACUUM，請參閱 [VACUUM](#)。

因此，我們建議視需要清空個別資料表。我們也建議使用此方法，因為清空整個資料庫可能會是相當耗費資源的操作。

自動資料表排序

Amazon Redshift 會在背景中自動排序資料，以依據其排序索引鍵的順序維護資料表資料。Amazon Redshift 會追蹤您的掃描查詢，以判斷資料表中的哪些區段適合排序。

取決於系統上的載入，Amazon Redshift 會自動啟動排序。此自動排序可減少執行VACUUM命令以保持資料排序金鑰順序的需求。如果您需要資料完全依排序索引鍵順序排序，例如在大量資料載入之後，

您仍然可以手動執行VACUUM命令。若要判斷您的資料表是否會因執行 VACUUM 而受益SORT，請監控 中的資料vacuum_sort_benefit欄[SVV_TABLE_INFO](#)。

Amazon Redshift 追蹤會掃描每個資料表上使用排序索引鍵的查詢。Amazon Redshift 會估計掃描及篩選每個資料表資料獲得改善的最大百分比 (如果資料表已完全排序的話)。您可以在 [SVV_TABLE_INFO](#) 中的 vacuum_sort_benefit 資料行中看見此估計。您可以使用此欄和 unsorted欄，來判斷何時查詢可以從資料表VACUUMSORT上手動執行中受益。unsorted 資料行反映了資料表的實體排序順序。vacuum_sort_benefit 資料欄會指定手動執行 排序資料表的影響VACUUMSORT。

例如，考量以下查詢：

```
select "table", unsorted,vacuum_sort_benefit from svv_table_info order by 1;
```

table	unsorted	vacuum_sort_benefit
sales	85.71	5.00
event	45.24	67.00

針對“sales”資料表，即使資料表約 86% 是實體未排序的，因 86% 未排序而造成的查詢效能影響也只有 5%。這可能是因為查詢只存取了資料表的一小部分，或是存取資料表的查詢非常少。針對“event”資料表，該資料表約 45% 是實體未排序的。但 67% 的查詢效能影響指出查詢可能存取了資料表的較大部分，或是存取資料表的查詢數相當龐大。資料表「事件」可能受益於執行 VACUUM SORT。

自動清空刪除

當您執行刪除時，資料列會標示為要刪除，但不會移除。Amazon Redshift 會根據資料庫資料表中已刪除的資料列數目，自動在背景執行VACUUMDELETE操作。Amazon Redshift 會將 排程 VACUUMDELETE在負載減少期間執行，並在負載過高期間暫停操作。

主題

- [VACUUM 頻率](#)
- [排序階段和合併階段](#)
- [清空閾值](#)
- [清空類型](#)
- [將真空時間降到最低](#)

VACUUM 頻率

您應該根據需求時常進行清空，才能保有一致的查詢效能。在決定執行VACUUM命令的頻率時，請考慮以下因素：

- 在叢集上預期活動最少VACUUM的期間執行，例如晚上或指定的資料庫管理時段。
- 在維護時段之外執行VACUUM命令。如需詳細資訊，請參閱[排程維護時段](#)。
- 大量未排序的區域會造成較長的清空時間。如果您延遲清空，清空會耗費更長時間，因為必須重新整理更多資料。
- VACUUM 是一項 I/O 密集型操作，因此完成真空所需的時間越長，它對叢集上執行的並行查詢和其他資料庫操作的影響就越大。
- VACUUM 使用交錯排序的資料表需要更長的時間。如要評估是否必須重新排序交錯的資料表，請查詢 [SVV_INTERLEAVED_COLUMNS](#) 檢視。

排序階段和合併階段

Amazon Redshift 會在兩個階段中執行清空操作：首先，排序未排序區域中的資料列，然後在必要時，將資料表尾端新排序的資料列與現有資料列合併。清空大型資料表時，清空操作會以一系列的步驟進行，其中包含遞增排序，接著是合併。如果操作失敗或如果 Amazon Redshift 在清空期間離線，部分清空的資料表或資料庫將處於一致的狀態，但您將必須手動重新開始清空操作。遞增排序會遺失，但不需要再次清空失敗之前認可的合併資料列。如果未排序的區域很大，損失的時間可能會很可觀。如需排序和合併階段的相關資訊，請參閱[減少合併資料列的磁碟區](#)。

使用者可以在清空資料表時加以存取。您可以在將資料表清空時執行查詢和寫入操作，但同時執行 DML 和 真空執行時，兩者可能需要更長的時間。如果您在真空期間執行 UPDATE 和 DELETE 陳述式，系統效能可能會降低。增量合併會暫時封鎖並行 UPDATE 和 DELETE 操作，而 UPDATE 和 DELETE 操作則會依序暫時封鎖受影響資料表上的增量合併步驟。DDL 操作，例如 ALTER 會遭到封鎖 TABLE，直到真空操作完成資料表為止。

Note

控制其運作VACUUM方式的各種修改程式。您可以根據當前需求，使用這些修飾詞來制定清空操作。例如，使用 `VACUUM RECLUSTER` 縮短真空操作，方法是不執行完全合併操作。如需詳細資訊，請參閱[VACUUM](#)。

清空閾值

根據預設，會VACUUM略過任何資料表的排序階段，其中已排序資料表列的 95% 以上。跳過排序階段可以大幅改善VACUUM效能。若要變更單一資料表的預設排序閾值，請在執行VACUUM命令時包含資料表名稱和 TO 閾值PERCENT參數。

清空類型

如需不同清空類型的詳細資訊，請參閱 [VACUUM](#)。

將真空時間降到最低

Amazon Redshift 會自動排序資料並在背景VACUUMDELETE執行。這可減少執行 VACUUM 命令的需要。吸塵可能是一個耗時的程序。根據您的資料性質，我們建議您採取下列做法，以將真空時間降至最低。

主題

- [決定是否重新索引](#)
- [縮小未排序區域的大小](#)
- [減少合併資料列的磁碟區](#)
- [以排序索引鍵順序載入您的資料](#)
- [使用時間序列表來減少儲存的資料](#)

決定是否重新索引

您通常可以使用交錯的排序樣式大幅改善查詢效能，但如果排序索引鍵資料欄中值的配送變更，效能可能隨著時間降級。

當您一開始使用 COPY或 CREATE TABLE AS 載入空的交錯資料表時，Amazon Redshift 會自動建置交錯索引。如果您一開始使用 載入交錯資料表INSERT，則需要在VACUUMREINDEX之後執行，以初始化交錯索引。

隨著時間，在您新增具有新排序索引鍵值的資料列時，如果排序索引鍵資料欄中值的配送變更，效能可能降級。如果您的新資料列主要在現有排序索引鍵值的範圍內，則不需要重建索引。執行 VACUUMSORTONLY或 VACUUMFULL以還原排序順序。

查詢引擎可以使用排序順序來有效地選取處理查詢需要掃描的資料區塊。針對交錯的排序，Amazon Redshift 會分析排序索引鍵資料欄值，以判斷最佳排序順序。如果隨著資料列新增，索引鍵值的配送變

更或偏移，排序策略將不再最佳，而排序的效能優勢將會降級。若要重新分析排序金鑰分佈，您可以執行 VACUUM REINDEX。重建索引操作相當耗時，因此，若要決定資料表是否將可從重建索引獲益，請查詢 [SVV_INTERLEAVED_COLUMNS](#) 檢視。

例如，下列查詢顯示使用交錯排序索引鍵的資料表的詳細資訊。

```
select tbl as tbl_id, stv_tbl_perm.name as table_name,
col, interleaved_skew, last_reindex
from svv_interleaved_columns, stv_tbl_perm
where svv_interleaved_columns.tbl = stv_tbl_perm.id
and interleaved_skew is not null;
```

tbl_id	table_name	col	interleaved_skew	last_reindex
100048	customer	0	3.65	2015-04-22 22:05:45
100068	lineorder	1	2.65	2015-04-22 22:05:45
100072	part	0	1.65	2015-04-22 22:05:45
100077	supplier	1	1.00	2015-04-22 22:05:45

(4 rows)

interleaved_skew 的值為比率，指出偏移量。值為 1 表示沒有偏移。如果偏移大於 1.4，除非底層集合中固有偏移，否則 VACUUMREINDEX 通常會改善效能。

您可以使用 last_reindex 中的日期值來判斷自上次重建索引後經過的時間。

縮小未排序區域的大小

載入大量新資料至已包含資料的資料表，或當您不隨著例行維護操作清空資料表時，未排序的區域會成長。若要避免長時間執行清空操作，請使用下列做法：

- 根據定期排程執行清空操作。

如果您以小幅度載入資料表（例如代表資料表中資料列總數小百分比的每日更新），則 VACUUM 定期執行將有助於確保個別的真​​空操作快速進行。

- 先執行最大型的載入。

如果您需要載入具有多個 COPY 操作的新資料表，請先執行最大載入。執行初始載入至新的或截斷的資料表時，所有資料會直接載入至排序的區域，因此不需要清空。

- 截斷資料表，而非刪除所有資料列。

從資料表刪除資料列不會回收該資料列佔用的空間，直到您執行清空操作為止；不過，截斷資料表會清空資料表和回收磁碟空間，因此不需要清空。或者，捨棄資料表和重新建立它。

- 截斷或捨棄測試資料表。

如果您因測試目的載入少量的資料列至資料表，在完成之前請勿刪除資料列。而是截斷資料表，並在後續生產載入操作中重新載入這些資料列。

- 執行深層複製。

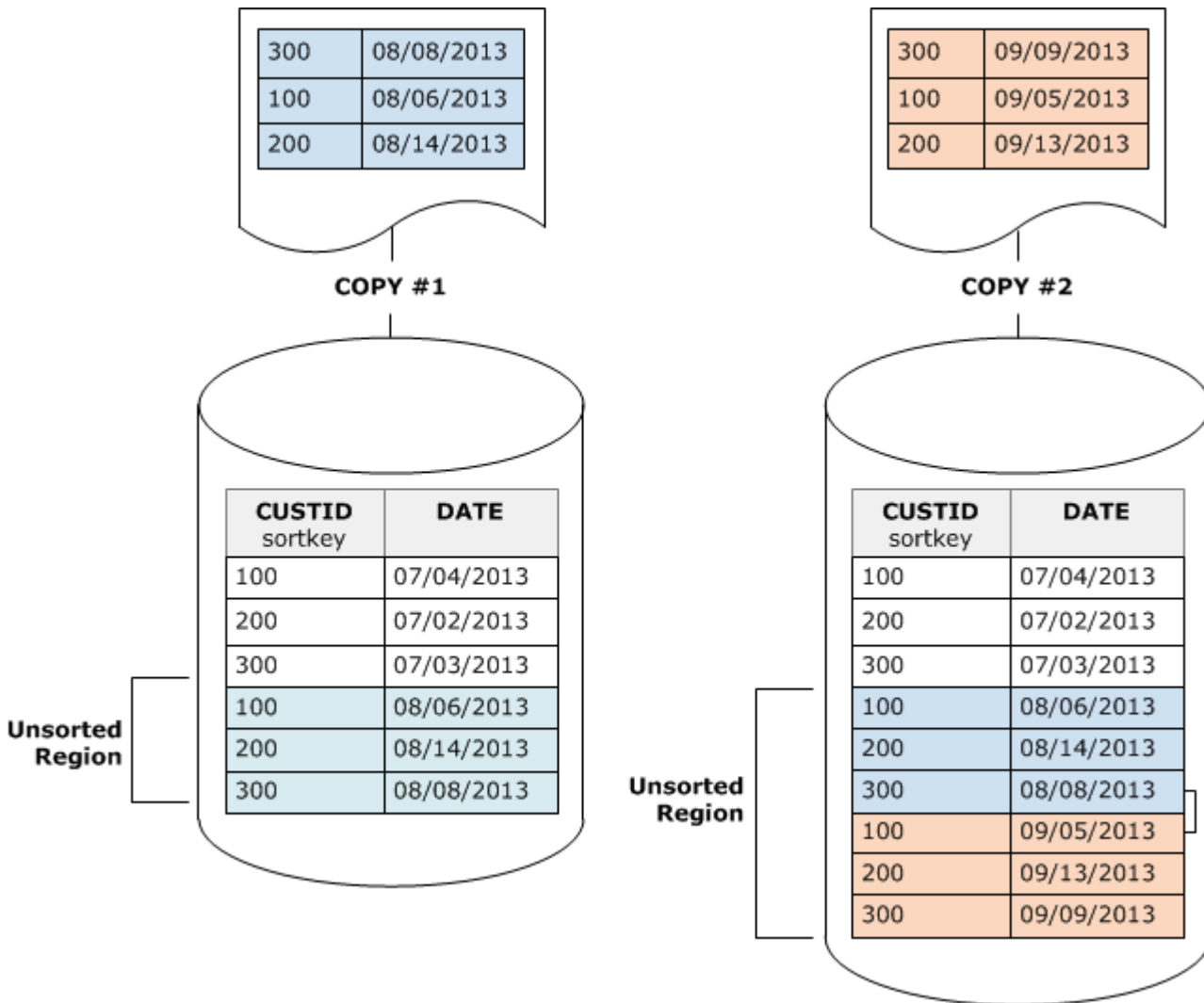
如果使用複合排序索引鍵資料表的資料表有大型未排序的區域，深層複製較清空的速度快得多。深層複製會使用會自動重新排序資料表的大量插入來重新建立和重新填入資料表。如果資料表有大型未排序的區域，深層複製較清空的速度快得多。取捨是您不可以在深層複製操作期間進行並行更新，但清空期間卻可以。如需詳細資訊，請參閱[設計查詢的 Amazon Redshift 最佳實務](#)。

減少合併資料列的磁碟區

如果清空操作必須將新資料列合併至資料表的排序區域，清空所需的時間將隨著資料表變大而增加。您可以透過減少必須合併的資料列數量來改善清空效能。

在清空之前，資料表在資料表的標頭中包含排序的區域，接著是未排序的區域，它會在資料列新增或更新時成長。COPY 操作新增一組資料列時，新資料列會在新增至資料表結尾的未排序區域時，在排序索引鍵上進行排序。新資料列會在其自己的集合內排序，但不會在未排序的區域內排序。

下圖說明兩次連續COPY操作後的未排序區域，其中排序索引鍵為 CUSTID。為求簡化，此範例顯示複合排序索引鍵，但相同的原則適用於交錯的排序索引鍵，除了未排序區域的影響大於交錯的資料表。



清空會以兩個階段還原資料表的排序順序：

1. 將未排序的區域排序為新排序的區域。

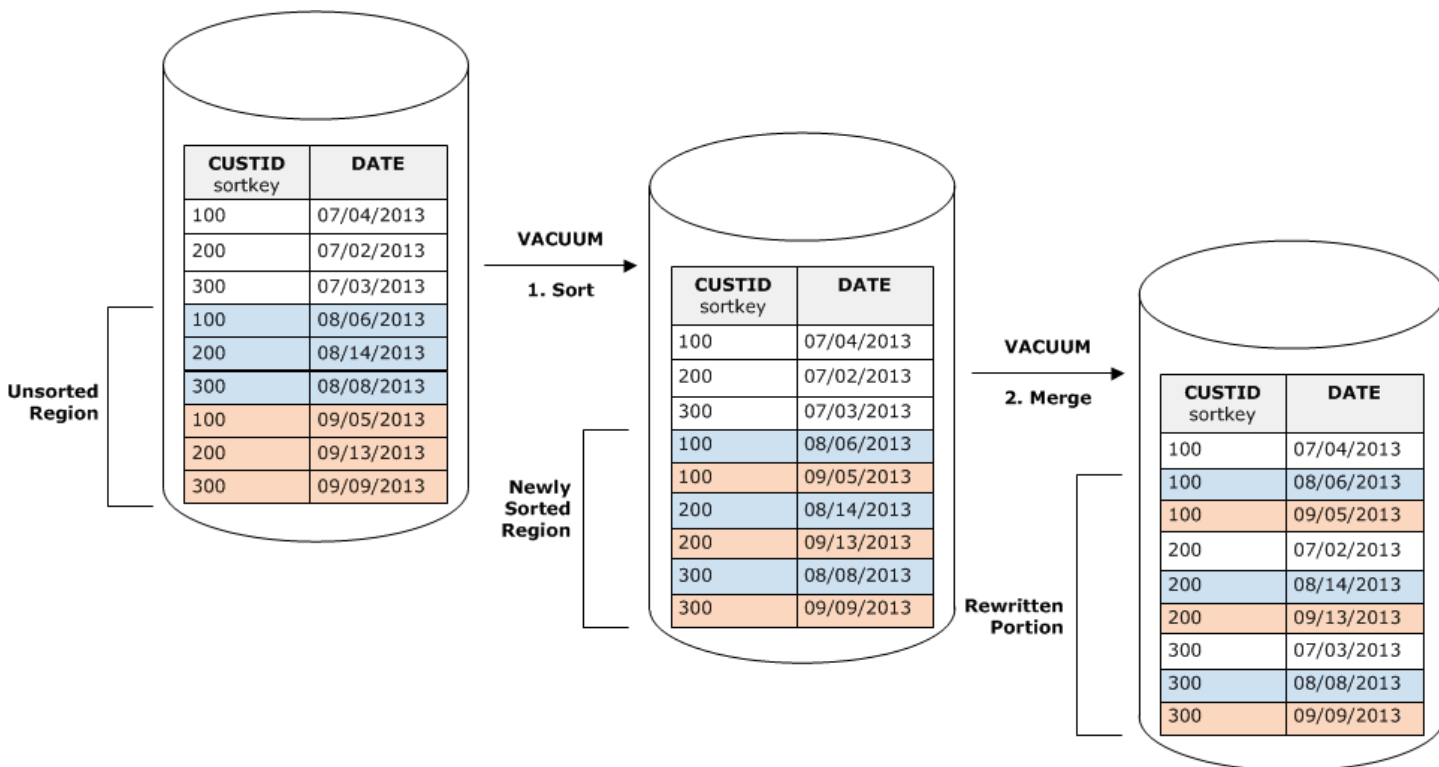
第一個階段相當經濟實惠，因為只會重新寫入未排序的區域。如果新排序區域的排序索引鍵值的範圍高於現有範圍，則只需要重新寫入新資料列，清空即完成。例如，如果排序的區域包含 ID 值 1 到 500，而和後續的複製操作會新增大於 500 的索引鍵值，那麼只必須重新寫入未排序的區域。

2. 將新排序的區域與先前排序的區域合併。

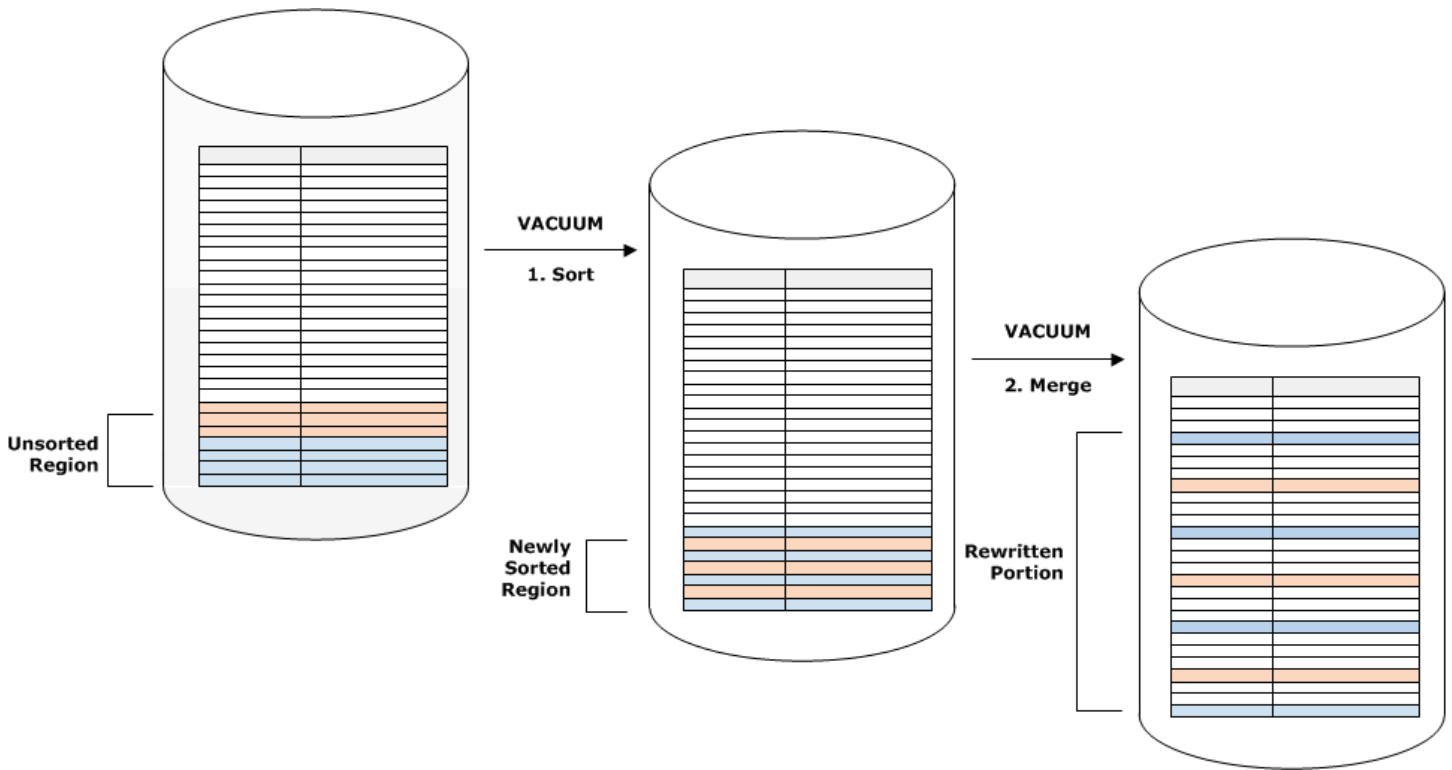
如果新排序區域中的索引鍵與排序區域中的索引鍵重疊，則 VACUUM 需要合併資料列。從新排序區域 (最下方的排序索引鍵) 的開頭開始，清空會從將合併的資料列從先前排序的區域和新排序的區域寫入至新的一組區塊。

新排序索引鍵範圍與現有排序索引鍵重疊的程度，決定需要重新寫入先前排序區域的程度。如果未排序的索引鍵分散在現有排序範圍中，清空可能需要重新寫入資料表的現有部分。

下圖顯示真空如何排序和合併新增至資料表的資料列，其中 CUSTID 是排序索引鍵。因為每個複製操作會新增一組新資料列，其具有的索引鍵值與現有索引鍵重疊，幾乎必須重新寫入整個資料表。此表顯示單一的排序和合併，但在實務上，大型的清空包含一系列的遞增排序和合併步驟。



如果一組新資料列中排序索引鍵的範圍與現有索引鍵的範圍重疊，合併階段的成本會隨著資料表成長，繼續按比例成長至資料表大小，同時排序階段的成本會維持與未排序區域的大小成比例。在這類情況下，合併階段的成本會使得排序階段的成本相形見绌，如下表所示。



若要判斷資料表重新合併的比例，請在真空操作完成後查詢 `SVV_VACUUM_SUMMARY`。下列查詢顯示隨著時間的推移，`CUSTSALES` 表而增長的六個連續吸塵器的影響。

```
select * from svv_vacuum_summary
where table_name = 'custsales';
```

table_name	xid	sort_	merge_	elapsed_	row_	sortedrow_	block_
		partitions	increments	time	delta	delta	delta
		partitions					
custsales	7072	3	2	143918314	0	88297472	1524
	47						
custsales	7122	3	3	164157882	0	88297472	772
	47						
custsales	7212	3	4	187433171	0	88297472	767
	47						
custsales	7289	3	4	255482945	0	88297472	770
	47						
custsales	7420	3	5	316583833	0	88297472	769
	47						

```

custsales | 9007 |          3 |          6 | 306685472 | 0 | 88297472 | 772
|         47
(6 rows)

```

`merge_increments` 資料欄提供針對每個清空操作所合併資料量的指示。如果合併增量的數目超過資料表大小成長比例後續清空的增加，即表示因為現有和新排序的區域重疊，每個清空操作會重新合併資料表中增加數量的資料列。

以排序索引鍵順序載入您的資料

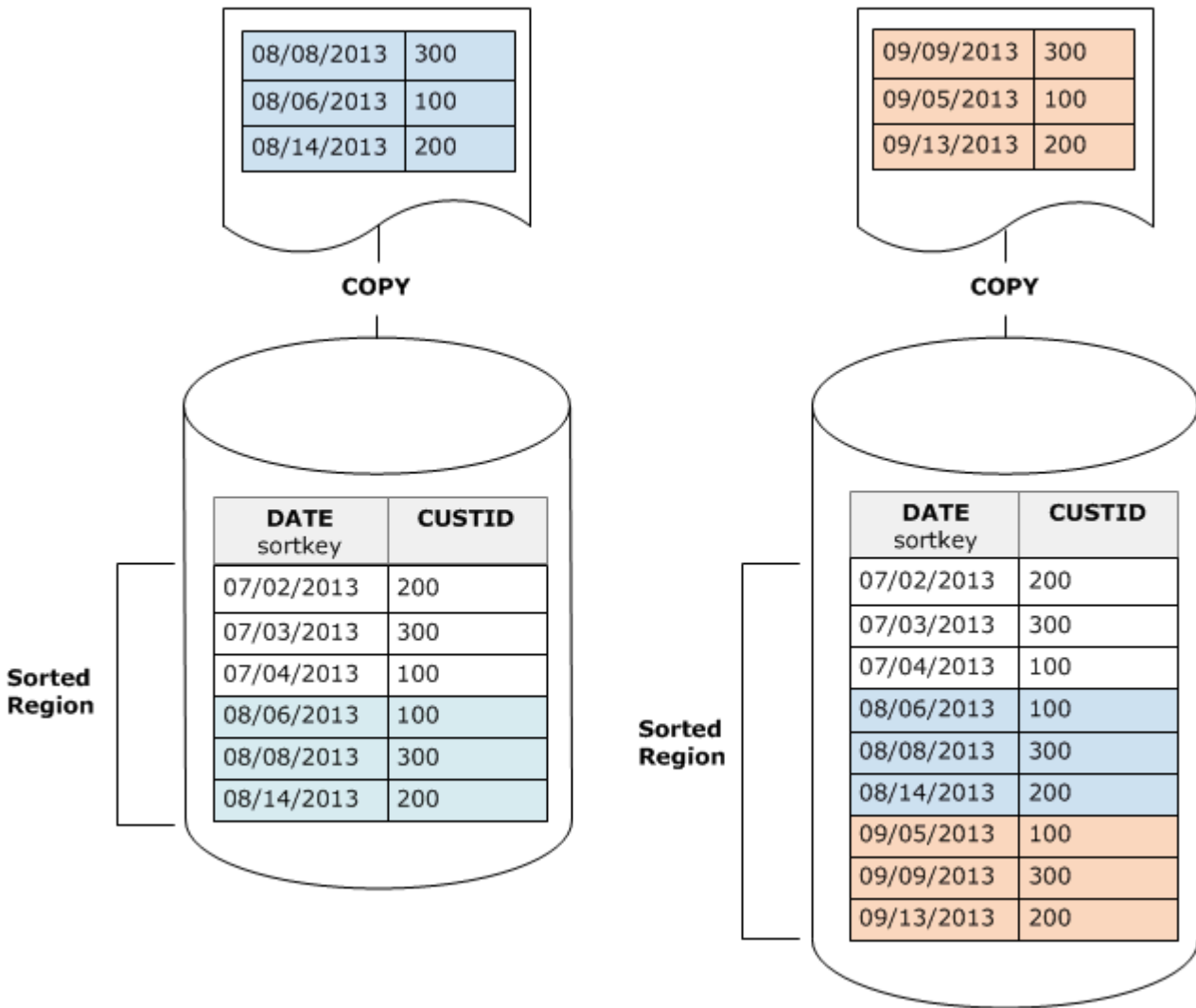
如果您使用 `COPY` 命令以排序索引鍵順序載入資料，則可能會減少或甚至移除清空的必要性。

`COPY` 當下列所有項目都為 `true` 時，會自動將新資料列新增至資料表的排序區域：

- 資料表使用複合排序索引鍵搭配僅一個排序資料欄。
- 排序欄為 `NOT NULL`。
- 資料表完全經過排序或為空白。
- 所有新資料列的排序順序高於現有資料列，包括標記進行刪除的資料列。在此執行個體中，Amazon Redshift 會使用排序索引鍵的前八個位元組來判斷排序順序。

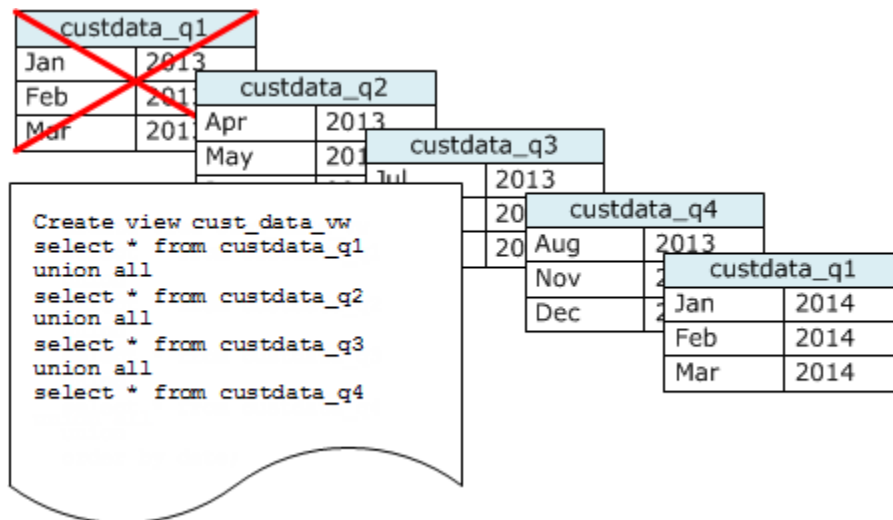
例如，假設您有一個使用客戶 ID 和時間記錄客戶事件的資料表。如果您依客戶 ID 排序，遞增載入新增的新資料列的排序索引鍵範圍很可能會與現有範圍重疊，如先前的範例所示，導致代價高昂的清空操作。

如果您將您的排序索引鍵設為時間戳記資料欄，您的新資料列將以排序順序附加在資料表結尾，如下表所示，降低或甚至消除清空的需求。



使用時間序列表來減少儲存的資料

如果您維護資料一段時間，使用一系列資料表，如下圖所述。



每次您新增一組資料即建立新的資料表，然後刪除系列中最舊的資料表。您會獲得雙重優勢：

- 您可以避免刪除資料列的額外成本，因為DROPTABLE操作比大量更有效率DELETE。
- 如果資料表是依時間戳記排序，則不需要清空。如果每個資料表包含一個月的資料，清空將至少必須重新寫入一個月的資料量，即使資料表未依時間戳記排序。

您可以透過報告隱藏資料存放在多個資料表中的事實的查詢來建立使用UNIONALL檢視。如果查詢會依排序索引鍵篩選，查詢規劃器可以有效地略過所有未使用的資料表。對於其他類型的查詢，效率UNIONALL可能較低，因此您應該在使用資料表的所有查詢內容中評估查詢效能。

管理並行寫入操作

Amazon Redshift 允許在遞增載入或修改資料表時加以讀取。

在一些傳統資料資料倉儲和商業智慧應用程式中，只有在每夜載入完成時，資料庫才可供使用者使用。在這類情況下，當分析查詢執行和報告產生時的一般工作時段期間不允許進行更新；不過，應用程式數目的增加會長時間或甚至整天保持在線上，使得載入時段的概念過時。

Amazon Redshift 藉由允許在遞增載入或修改資料表時加以讀取，來支援這些類型的應用程式。查詢只會看到最新認可的資料版本或快照，而非等候認可的下一個版本。如果您想要特定查詢等候來自另一個寫入操作的認可，您必須據以排程。

下列主題說明一些牽涉交易、資料庫快照、更新和並行行為的重要概念和使用案例。

主題

- [可序列化隔離](#)

- [寫入和讀/寫操作](#)
- [並行寫入範例](#)

可序列化隔離

有些應用程式需要的不僅僅是並行查詢和載入，還有並行寫入多個資料表或相同的資料表的能力。在此情況下，並行表示重疊，而不是排程在完全相同的時間執行。如果第二個交易在第一個認可之前開始，則會把兩個交易會視為並行。並行操作可能源自受相同使用者或不同使用者控制的不同工作階段。

Note

Amazon Redshift 支援預設的自動遞交行為，其中每個單獨執行SQL命令都會個別遞交。如果您在交易區塊中含括一組命令 (由 [BEGIN](#) 和 [END](#) 陳述式定義)，該區塊會以一個交易的形式認可，因此您可以在必要時加以復原。此行為的例外是 TRUNCATE和 VACUUM命令，其會自動遞交目前交易中所有未完成的變更。

有些SQL用戶端會自動發出 BEGIN 和 COMMIT 命令，因此用戶端會控制一組陳述式是作為交易執行，還是每個個別陳述式都會作為自己的交易執行。請查看您正在使用的界面文件。例如，使用 Amazon Redshift JDBC驅動程式時，JDBCPreparedStatement具有查詢字串的包含多個（以分號分隔）SQL命令，會以單一交易方式執行所有陳述式。相反地，如果您使用 SQL Workbench/J 並設定 AUTO COMMIT ON，則如果您執行多個陳述式，則每個陳述式都會作為自己的交易執行。

Amazon Redshift 中是使用資料表上的寫入鎖定及可序列化隔離的原則，以保護的方式支援並行寫入操作。可序列化隔離可維持對資料表執行的交易對該資料表執行的唯一交易的錯覺。例如，兩個並行執行的交易，T1 和 T2，必須產生與下列至少一項相同的結果：

- T1 和 T2 會依照該順序循序執行。
- T2 和 T1 會依照該順序循序執行。

並行交易彼此看不見對方；它們無法偵測彼此的變更。每個並行交易將在交易開始時建立資料庫的快照。在交易中建立資料庫快照時，會針對大多數SELECT陳述式、COPY、UPDATE、DELETE INSERT和 等DML命令TRUNCATE，以及下列DDL命令的第一次出現：

- ALTER TABLE（新增或捨棄資料欄）
- CREATE TABLE

- DROP TABLE
- TRUNCATE TABLE

如果並行交易的任何序列執行會產生與其並行執行相同的結果，這些交易會被視為「可序列化」並且可安全地執行。如果這些交易中沒有任何序列執行會產生相同的結果，系統會停止和還原所執行陳述式可能破壞序列化可行性的交易。

系統目錄表（PG）和其他 Amazon Redshift 系統資料表（STL 和 STV）不會鎖定在交易中。因此，在提交任何並行交易時，即可看到從 DDL 和 TRUNCATE 操作產生的資料庫物件變更。

例如，假設當兩個並行交易 T1 和 T2 開始時，資料表 A 存在於資料庫。假設 T2 透過從 PG_TABLES 目錄資料表中選取傳回資料表清單。然後 T1 捨棄資料表 A 並遞交，然後 T2 再次列出資料表。資料表 A 現在不會再出現在清單中。如果 T2 嘗試查詢已捨棄的資料表，Amazon Redshift 會傳回「關聯不存在」錯誤。傳回資料表的清單至 T2 或檢查資料表 A 存在的類別查詢，不會受限於與對使用者資料表執行的相同隔離規則。

對這些資料表更新的交易會在讀取已認可隔離模式中執行。PG 字首類別資料表不支援快照隔離。

系統資料表和類別資料表的可序列化隔離

資料庫快照也會在交易中建立，用於參考使用者建立資料表或 Amazon Redshift 系統資料表（STL 或 STV）的任何 SELECT 查詢。SELECT 未參考任何資料表的查詢不會建立新的交易資料庫快照。INSERT、DELETE 和 UPDATE 陳述式僅在系統目錄資料表（PG）上運作，也不會建立新的交易資料庫快照。

如何修正可序列化隔離錯誤

ERROR : 1023 DETAIL : Redshift 中資料表上的序列化隔離違規

當 Amazon Redshift 偵測到可序列化隔離錯誤時，您會看到錯誤訊息，如下所示。

```
ERROR:1023 DETAIL: Serializable isolation violation on table in Redshift
```

若要解決可序列化隔離錯誤，您可以嘗試以下方法：

- 重試已取消的交易。

Amazon Redshift 偵測到並行工作負載不可序列化。其表明應用程式邏輯中有差距，此問題通常可以透過重試遇到錯誤的交易來解決。如果問題仍然存在，請嘗試使用其他方法。

- 將任何無須位於同一不可分割交易中的操作，移到交易之外。

當兩個交易中的個別操作以可能影響另一個交易結果的方式交叉參考時，適用此方法。例如，以下兩個工作階段各自啟動一個交易。

```
Session1_Redshift=# begin;
```

```
Session2_Redshift=# begin;
```

每個交易中的SELECT陳述式結果可能會受到另一個交易中的INSERT陳述式影響。換句話說，假設您以任何順序，依序執行以下陳述式。在所有情況下，結果都是傳回多一系列的SELECT其中一個陳述式，而不是同時執行交易。依序執行的操作中，沒有任何執行順序可以產生與並行執行相同的結果。因此，最後一個執行的操作會導致可序列化隔離錯誤。

```
Session1_Redshift=# select * from tab1;  
Session1_Redshift=# insert into tab2 values (1);
```

```
Session2_Redshift=# insert into tab1 values (1);  
Session2_Redshift=# select * from tab2;
```

在許多情況下，SELECT陳述式的結果並不重要。換句話說，交易中操作的不可分割性並不重要。在這些情況下，將SELECT陳述式移至其交易之外，如下列範例所示。

```
Session1_Redshift=# begin;  
Session1_Redshift=# insert into tab1 values (1)  
Session1_Redshift=# end;  
Session1_Redshift=# select * from tab2;
```

```
Session2_Redshift # select * from tab1;  
Session2_Redshift=# begin;  
Session2_Redshift=# insert into tab2 values (1)  
Session2_Redshift=# end;
```

在這些範例中，交易中沒有交叉參考。這兩個INSERT陳述式彼此之間沒有影響。在這些範例中，至少有一個順序，可讓交易依序執行並產生與並行執行相同的結果。這表示交易是可序列化的。

- 請透過鎖定每個工作階段中的所有資料表來強制序列化。

[LOCK](#) 命令能封鎖可能導致可序列化隔離錯誤的操作。使用 LOCK 命令時，請務必執行下列動作：

- 鎖定受交易影響的所有資料表，包括交易內受唯讀SELECT陳述式影響的資料表。
- 無論操作的執行順序為何，均以相同的順序鎖定資料表。
- 在交易開頭，執行任何操作之前，先鎖定所有資料表。
- 對並行交易使用快照隔離

使用具有快照隔離的 ALTER DATABASE 命令。如需 ALTER 的 SNAPSHOT 參數的詳細資訊 DATABASE，請參閱 [參數](#)。

ERROR : 1018DETAIL : 關聯不存在

當您在不同的工作階段中執行並行 Amazon Redshift 操作時，您會看到類似如下所示的錯誤訊息。

```
ERROR: 1018 DETAIL: Relation does not exist.
```

Amazon Redshift 中的交易會遵循快照隔離。交易開始後，Amazon Redshift 會擷取資料庫的快照。對於交易的整個生命週期，交易會根據快照中反映的資料庫狀態操作。如果交易從快照集中不存在的資料表進行讀取，則會擲回先前顯示的 1018 錯誤訊息。即使另一個並行交易在交易擷取快照集後建立資料表，交易也無法從新建立的資料表中進行讀取。

為了解決此序列化隔離錯誤，您可以嘗試將交易的開始移至您知道該資料表存在的位置。

如果該資料表是由另一個交易建立，則這一個位置點應至少在已遞交該交易之後。此外，請確定沒有遞交任何可能捨棄資料表的並行交易。

```
session1 = # BEGIN;  
session1 = # DROP TABLE A;  
session1 = # COMMIT;
```

```
session2 = # BEGIN;
```

```
session3 = # BEGIN;  
session3 = # CREATE TABLE A (id INT);  
session3 = # COMMIT;
```

```
session2 = # SELECT * FROM A;
```

session2 執行為讀取操作的最後一個操作會導致可序列化隔離錯誤。當 session2 擷取快照集，而資料表已被遞交的 session1 捨棄時，就會發生這個錯誤。換句話說，即使並行的 session3 已經建立資料表，但 session2 仍看不到該資料表，因為其不在快照中。

若要解決此錯誤，您可以按照下方式重新排序工作階段。

```
session1 = # BEGIN;  
session1 = # DROP TABLE A;  
session1 = # COMMIT;
```

```
session3 = # BEGIN;  
session3 = # CREATE TABLE A (id INT);  
session3 = # COMMIT;
```

```
session2 = # BEGIN;  
session2 = # SELECT * FROM A;
```

現在，當 session2 擷取其快照時，session3 已經遞交，且該資料表位於資料庫中。Session2 可以從資料表中讀取而不會出現任何錯誤。

寫入和讀/寫操作

透過決定何時與如何執行不同的類型的命令，您可以管理並行寫入操作的特定行為。以下是與此討論相關的命令：

- COPY 命令，執行載入（初始或增量）
- INSERT 一次附加一或多個列的命令
- UPDATE 命令，可修改現有的資料列
- DELETE 命令，其會移除資料列

COPY 和 INSERT 操作是純寫入操作，但 DELETE 和 UPDATE 操作是讀取/寫入操作。(若要讓資料列可供刪除或更新，必須先加以讀取。) 並行寫入操作的結果取決於並行執行的特定命令。COPY 與相同資料表的 INSERT 操作會保留在等待狀態，直到鎖定釋放為止，然後它們會正常進行。

UPDATE 和 DELETE 操作的行為不同，因為它們在進行任何寫入之前依賴初始資料表讀取。由於並行交易彼此看不到，因此 UPDATEs 和 DELETEs 必須讀取上次遞交的資料快照。當第一個 UPDATE 或 DELETE 釋出其鎖定時，第二個 UPDATE 或 DELETE 需要判斷要使用的資料是否可能過時。它將不會過時，因為第二個交易不會取得其資料的快照，直到第一個交易已釋出其鎖定為止。

並行寫入交易可能的死鎖情況

每當交易牽涉到一個以上資料表的更新，當它們都嘗試寫入相同的資料表集時，並行執行的交易有可能變得死鎖。交易會在認可或復原時一次釋出其所有資料表鎖定；它不會一次撤回一個鎖定。

例如，假設交易 T1 和 T2 在大約相同的時間開始。如果 T1 開始寫入資料表 A 和 T2 開始寫入資料表 B，這兩個交易都可以繼續而不衝突；不過，如果 T1 完成寫入資料表 A 並且必須開始寫入資料表 B，它將無法繼續，因為 T2 仍在 B 上保有鎖定。相反地，如果 T2 完成寫入資料表 B 並且必須開始寫入資料表 A，它將無法繼續，因為 T1 仍在 A 上保有鎖定。因為這兩個交易都可釋出其鎖定，直到其所有寫入操作都認可，因此沒有任一交易可以繼續。

為了避免這類的死鎖，您必須謹慎排程並行寫入操作。例如，在交易中，您應該一律以相同的順序更新資料表，如果指定鎖定，則在執行任何DML操作之前，以相同的順序鎖定資料表。

並行寫入範例

下列虛擬程式碼範例示範交易如何繼續或在並行執行時等候。

將COPY操作並行至相同的資料表

交易 1 將資料列複製到LISTING資料表中：

```
begin;
copy listing from ...;
end;
```

交易 2 會在單獨的工作階段中同時開始，並嘗試將更多資料列複製到LISTING資料表中。交易 2 必須等到交易 1 釋放LISTING資料表上的寫入鎖定，然後才能繼續。

```
begin;
[waits]
copy listing from ;
end;
```

如果一個或兩個交易包含 INSERT 命令而非 COPY 命令，則會發生相同的行為。

來自相同資料表的並行DELETE操作

交易 1 會從資料表刪除資料列：

```
begin;
```

```
delete from listing where ...;
end;
```

交易 2 會並行開始，並嘗試從相同的資料表刪除資料列。它會成功，因為它會在嘗試刪除資料列之前等候交易 1 完成。

```
begin
[waits]
delete from listing where ;
end;
```

如果一個或兩個交易包含對相同資料表的UPDATE命令，而不是DELETE命令，則會發生相同的行為。

使用讀取和寫入操作混合的並行交易

在此範例中，交易 1 會從USERS資料表中刪除資料列、重新載入資料表、執行 COUNT (*) 查詢，然後刪除 ANALYZE，然後再提交：

```
begin;
delete one row from USERS table;
copy ;
select count(*) from users;
analyze ;
end;
```

同時，交易 2 會開始。此交易會嘗試將其他資料列複製到USERS資料表中、分析資料表，然後執行與第一個交易相同的 COUNT (*) 查詢：

```
begin;
[waits]
copy users from ...;
select count(*) from users;
analyze;
end;
```

第二個交易將會成功，因為它必須等候第一個完成。其COUNT查詢會根據其已完成的載入傳回計數。

教學課程：從 Amazon S3 載入資料

本教學將從頭到尾引導您完成從 Amazon S3 儲存貯體中的資料檔案將資料載入 Amazon Redshift 資料庫資料表的過程。

在此教學課程中，您將執行下列操作：

- 下載使用逗號分隔值（CSV）、字元分隔和固定寬度格式的資料檔案。
- 建立 Amazon S3 儲存貯體，然後上傳資料檔案至該儲存貯體。
- 啟動 Amazon Redshift 叢集並建立資料庫資料表。
- 使用 COPY 命令從 Amazon S3 上的資料檔案載入資料表。
- 對載入錯誤進行故障診斷，並修改 COPY 命令以修正錯誤。

預估時間：60 分鐘

估計費用：叢集每小時 \$1.00

必要條件

您需要以下的事前準備：

- 用來啟動 Amazon Redshift 叢集並在 Amazon S3 中建立儲存貯體 AWS 的帳戶。
- 您的 AWS 登入資料（IAM 角色）可從 Amazon S3 載入測試資料。如果您需要新的 IAM 角色，請前往 [建立 IAM 角色](#)。
- SQL 用戶端，例如 Amazon Redshift 主控台查詢編輯器。

本教學課程設計為可獨立進行。除了本教學課程之外，我們也建議您完成下列教學課程，以便更全面地了解如何設計和使用 Amazon Redshift 資料庫：

- [Amazon Redshift 入門指南](#) 會逐步引導您完成建立 Amazon Redshift 叢集和載入範例資料的程序。

概觀

您可以使用 INSERT 命令或使用 COPY 命令將資料新增至 Amazon Redshift 資料表。在 Amazon Redshift 資料倉儲的規模和速度下，該 COPY 命令比 INSERT 命令快上許多倍且更有效率。

此 COPY 命令使用 Amazon Redshift 大規模平行處理（MPP）架構，從多個資料來源平行讀取和載入資料。您可以從 Amazon S3、Amazon EMR 或可透過 Secure Shell（SSH）連線存取的任何遠端主機上的資料檔案載入。或者，您可以直接從 Amazon DynamoDB 資料表載入。

在本教學課程中，您可以使用 COPY 命令從 Amazon S3 載入資料。在此運用的許多原則也適合用於從其他資料來源載入。

若要進一步了解如何使用 COPY 命令，請參閱下列資源：

- [載入資料的 Amazon Redshift 最佳實務](#)
- [從 Amazon 載入資料 EMR](#)
- [從遠端主機載入資料](#)
- [從 Amazon DynamoDB 資料表載入資料](#)

步驟 1：建立叢集

如果您已有想要使用的叢集，則可略過此步驟。

針對本教學中的練習，請使用四個節點的叢集。

建立叢集

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。

在導覽功能表上，選擇已佈建的叢集儀表板。

Important

請確定您具備執行叢集操作的必要許可。如需授予必要許可的資訊，請參閱[授權 Amazon Redshift 存取 AWS 服務](#)。

2. 在右上角，選擇要 AWS 建立叢集的區域。基於本教學的用途，請選擇美國西部 (奧勒岡)。
3. 在導覽選單上，選擇叢集，然後選擇建立叢集。建立叢集頁面隨即出現。
4. 在建立叢集頁面上輸入叢集的參數。為參數選擇您自己的值，但變更下列值時除外：
 - 選擇 **dc2.large** 作為節點類型。
 - 選擇 **4** 作為節點數目。
 - 在叢集許可區段中，從可用 IAM 角色 中選擇角色。IAM 此角色應為您先前建立的角色，且具備 Amazon S3 的存取權限。然後選擇關聯 IAM 角色，將其新增至叢集的關聯 IAM 角色清單。
5. 選擇建立叢集。

請遵循 [Amazon Redshift 入門指南](#) 步驟，從 SQL 用戶端連線至叢集並測試連線。您不需要完成「入門」其餘的建立資料表、上傳資料、嘗試範例查詢步驟。

步驟 2：下載資料檔案

在此步驟，您會下載一組範例資料檔案到您的電腦。在下一個步驟，您會將檔案上傳到 Amazon S3 儲存貯體。

下載資料檔案

1. 下載壓縮檔案：[LoadingDataSampleFiles.zip](#)。
2. 將檔案解壓縮至您電腦中的資料夾。
3. 確認資料夾中包含下列檔案。

```
customer-fw-manifest
customer-fw.tbl-000
customer-fw.tbl-000.bak
customer-fw.tbl-001
customer-fw.tbl-002
customer-fw.tbl-003
customer-fw.tbl-004
customer-fw.tbl-005
customer-fw.tbl-006
customer-fw.tbl-007
customer-fw.tbl.log
dwdate-tab.tbl-000
dwdate-tab.tbl-001
dwdate-tab.tbl-002
dwdate-tab.tbl-003
dwdate-tab.tbl-004
dwdate-tab.tbl-005
dwdate-tab.tbl-006
dwdate-tab.tbl-007
part-csv.tbl-000
part-csv.tbl-001
part-csv.tbl-002
part-csv.tbl-003
part-csv.tbl-004
part-csv.tbl-005
part-csv.tbl-006
part-csv.tbl-007
```


步驟 3：上傳檔案至 Amazon S3 儲存貯體

在此步驟，您要建立 Amazon S3 儲存貯體，並將資料檔案上傳至該儲存貯體。

將檔案上傳至 Amazon S3 儲存貯體

1. 在 Amazon S3 中建立儲存貯體。

如需有關建立儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[建立儲存貯體](#)。

- a. 登入 AWS Management Console 並在 開啟 Amazon S3 主控台<https://console.aws.amazon.com/s3/>。
- b. 選擇建立儲存貯體。
- c. 選擇 AWS 區域。

在和叢集相同的區域中建立儲存貯體。如果叢集位於美國西部 (奧勒岡) 區域，請選擇美國西部 (奧勒岡) 區域 (us-west-2)。

- d. 在建立儲存貯體對話方塊的儲存貯體名稱中，輸入儲存貯體名稱。

在 Amazon S3 現有的所有儲存貯體名稱中，您選擇的儲存貯體名稱必須是唯一的。其中一種協助確保唯一性的方法是在儲存貯體名稱的前面加上組織名稱。儲存貯體名稱必須符合一些規則。如需詳細資訊，請參閱 [Amazon Simple Storage Service 使用者指南](#) 中的儲存貯體限制與局限。

- e. 為其餘選項選擇建議的預設值。
- f. 選擇建立儲存貯體。

當 Amazon S3 成功建立您的儲存貯體，主控台會在儲存貯體面板中顯示您的空儲存貯體。

2. 建立資料夾。

- a. 選擇新儲存貯體的名稱。
- b. 選擇建立資料夾按鈕。
- c. 將新資料夾命名為 **load**。

Note

您建立的儲存貯體不在沙盒中。在此練習中，您會將物件新增至真正的儲存貯體。您需要為您在儲存貯體中儲存物件的期間支付一筆名目費用。如需 Amazon S3 定價的相關資訊，請移至 [Amazon S3 定價](#) 頁面。

3. 將資料檔案上傳至新的 Amazon S3 儲存貯體。

- a. 選擇資料夾的名稱。
- b. 在上傳精靈中，選擇新增檔案。

遵循 Amazon S3 主控台的指示，上傳您下載並擷取的所有檔案。

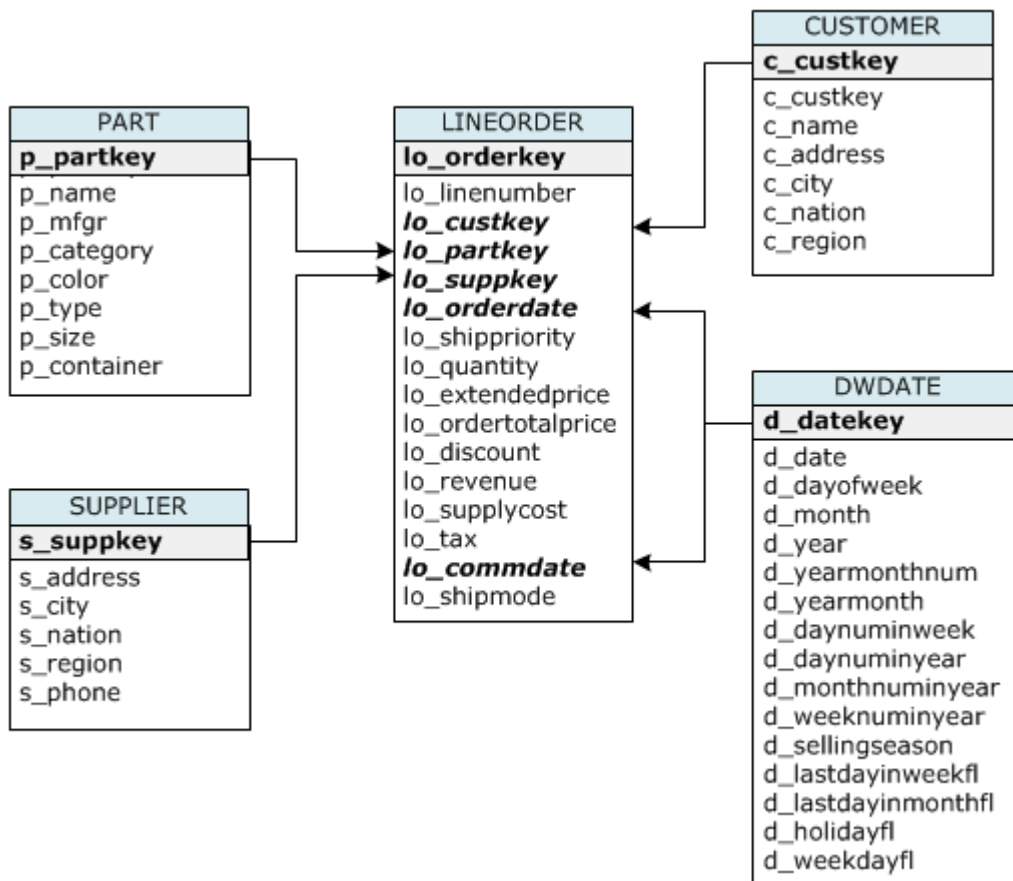
- c. 選擇上傳。

使用者登入資料

Amazon Redshift COPY命令必須具有讀取 Amazon S3 儲存貯體中檔案物件的存取權。如果您使用相同的使用者憑證來建立 Amazon S3 儲存貯體並執行 Amazon Redshift COPY命令，則該COPY命令具有所有必要的許可。如果您要使用不同的使用者登入資料，可以使用 Amazon S3 存取控制來授予存取權。Amazon Redshift COPY命令至少需要 ListBucket 和 GetObject 許可才能存取 Amazon S3 儲存貯體中的檔案物件。如需對 Amazon S3 資源的存取控制相關資訊，請參閱[管理對 Amazon S3 資源的存取許可](#)。

步驟 4：建立範例資料表

在本教學課程中，您可以使用一組以 Star Schema Benchmark (SSB) 結構描述為基礎的五個資料表。下圖顯示SSB資料模型。



SSB 資料表可能已存在於目前的資料庫中。如果是這樣，請在使用下一個步驟中的CREATETABLE命令建立資料表之前，先將資料表從資料庫移除。本教學課程中使用的資料表可能會有和現有資料表不同的屬性。

建立範例資料表

- 若要捨棄SSB資料表，請在SQL用戶端中執行下列命令。

```
drop table part cascade;
drop table supplier;
drop table customer;
drop table dwdate;
drop table lineorder;
```

- 在SQL用戶端中執行下列CREATETABLE命令。

```
CREATE TABLE part
(
  p_partkey    INTEGER NOT NULL,
  p_name       VARCHAR(22) NOT NULL,
```

```
p_mfgr      VARCHAR(6),
p_category  VARCHAR(7) NOT NULL,
p_brand1    VARCHAR(9) NOT NULL,
p_color     VARCHAR(11) NOT NULL,
p_type      VARCHAR(25) NOT NULL,
p_size      INTEGER NOT NULL,
p_container VARCHAR(10) NOT NULL
);

CREATE TABLE supplier
(
  s_suppkey  INTEGER NOT NULL,
  s_name     VARCHAR(25) NOT NULL,
  s_address  VARCHAR(25) NOT NULL,
  s_city     VARCHAR(10) NOT NULL,
  s_nation   VARCHAR(15) NOT NULL,
  s_region   VARCHAR(12) NOT NULL,
  s_phone    VARCHAR(15) NOT NULL
);

CREATE TABLE customer
(
  c_custkey  INTEGER NOT NULL,
  c_name     VARCHAR(25) NOT NULL,
  c_address  VARCHAR(25) NOT NULL,
  c_city     VARCHAR(10) NOT NULL,
  c_nation   VARCHAR(15) NOT NULL,
  c_region   VARCHAR(12) NOT NULL,
  c_phone    VARCHAR(15) NOT NULL,
  c_mktsegment VARCHAR(10) NOT NULL
);

CREATE TABLE dwdate
(
  d_datekey      INTEGER NOT NULL,
  d_date         VARCHAR(19) NOT NULL,
  d_dayofweek    VARCHAR(10) NOT NULL,
  d_month        VARCHAR(10) NOT NULL,
  d_year         INTEGER NOT NULL,
  d_yearmonthnum INTEGER NOT NULL,
  d_yearmonth    VARCHAR(8) NOT NULL,
  d_daynuminweek INTEGER NOT NULL,
  d_daynuminmonth INTEGER NOT NULL,
  d_daynuminyear INTEGER NOT NULL,
```

```
d_monthnuminyear    INTEGER NOT NULL,
d_weeknuminyear     INTEGER NOT NULL,
d_sellingseason     VARCHAR(13) NOT NULL,
d_lastdayinweekfl   VARCHAR(1) NOT NULL,
d_lastdayinmonthfl  VARCHAR(1) NOT NULL,
d_holidayfl        VARCHAR(1) NOT NULL,
d_weekdayfl        VARCHAR(1) NOT NULL
);
CREATE TABLE lineorder
(
  lo_orderkey        INTEGER NOT NULL,
  lo_linenumbers     INTEGER NOT NULL,
  lo_custkey         INTEGER NOT NULL,
  lo_partkey         INTEGER NOT NULL,
  lo_suppkey         INTEGER NOT NULL,
  lo_orderdate       INTEGER NOT NULL,
  lo_orderpriority   VARCHAR(15) NOT NULL,
  lo_shippriority    VARCHAR(1) NOT NULL,
  lo_quantity        INTEGER NOT NULL,
  lo_extendedprice   INTEGER NOT NULL,
  lo_ordertotalprice INTEGER NOT NULL,
  lo_discount        INTEGER NOT NULL,
  lo_revenue         INTEGER NOT NULL,
  lo_supplycost      INTEGER NOT NULL,
  lo_tax             INTEGER NOT NULL,
  lo_commitdate      INTEGER NOT NULL,
  lo_shipmode        VARCHAR(10) NOT NULL
);
```

步驟 5：執行COPY命令

您可以執行COPY命令來載入SSB結構描述中的每個資料表。COPY 命令範例示範使用數個COPY命令選項從不同檔案格式載入，以及疑難排解載入錯誤。

COPY 命令語法

[COPY](#) 命令基本語法如下。

```
COPY table_name [ column_list ] FROM data_source CREDENTIALS access_credentials
[options]
```

若要執行COPY命令，請提供下列值。

資料表名稱

COPY 命令的目標資料表。此資料表必須已存在於資料庫中。此資料表可以是暫時性或持久性。COPY 命令會將新的輸入資料附加到資料表中的任何現有資料列。

資料欄清單

根據預設，會依序將來源資料中的欄位COPY載入資料表資料欄。您可以選擇指定資料欄清單 (以逗號分隔資料欄名稱的清單)，以便將資料欄位映射到特定的資料欄。您在本教學中不會使用資料行清單。如需詳細資訊，請參閱 COPY 命令參考 [Column List](#) 中的。

資料來源

您可以使用 COPY 命令從 Amazon S3 儲存貯體、Amazon EMR 叢集、使用 SSH 連線的遠端主機或 Amazon DynamoDB 資料表載入資料。在本教學中，您會從 Amazon S3 儲存貯體中的資料檔案載入資料。從 Amazon S3 進行載入時，您必須提供儲存貯體的名稱和資料檔案的位置。若要執行此作業，請提供資料檔案的物件路徑，或是明確列出每個資料檔案及其位置的資訊清單檔案位置。

- 索引鍵字首

使用物件索引鍵可以唯一識別儲存在 Amazon S3 中的物件。物件索引鍵包含儲存貯體名稱、資料夾名稱、物件名稱 (如果有)。物件索引鍵字首是指具有相同字首的多個物件。物件路徑是 COPY 命令用來載入共用金鑰字首的所有物件的金鑰字首。例如，索引鍵字首 `custdata.txt` 可以是指單一檔案或數個檔案，包括 `custdata.txt.001`、`custdata.txt.002` 等。

- 清單檔案

在某些案例中，您可能需要載入具有不同字首的檔案，例如從多個儲存貯體或資料夾進行載入。在其他案例中，您可能需要排除具有特定字首的檔案。在這些案例中，您可以使用資訊清單檔案。資訊清單檔案會明確列出每個載入檔案及其唯一物件索引鍵。您可以在本教學課程稍後使用資訊清單檔案載入 PART 資料表。

登入資料

若要存取包含要載入資料 AWS 的資源，您必須為具有足夠權限的使用者提供 AWS 存取憑證。這些憑證包含 IAM 角色 Amazon Resource Name (ARN)。若要從 Amazon S3 載入資料，憑證必須包含 ListBucket 和 GetObject 許可。如果您的資料已加密，則需要其他登入資料。如需詳細資訊，請參閱 COPY 命令參考 [授權參數](#) 中的。如需管理存取權的相關資訊，請前往 [管理對 Amazon S3 資源的存取許可](#)。

選項

您可以使用 COPY 命令指定許多參數，以指定檔案格式、管理資料格式、管理錯誤，以及控制其他功能。在本教學課程中，您可以使用下列 COPY 命令選項和功能：

- 索引鍵字首

如需如何透過指定金鑰前綴從多個檔案載入的相關資訊，請參閱 [使用 NULL AS 載入 PART 資料表](#)。

- CSV 格式

如需有關如何載入 CSV 格式資料的資訊，請參閱 [使用 NULL AS 載入 PART 資料表](#)。

- NULL AS

如需如何使用 NULL AS PART 選項載入的資訊，請參閱 [使用 NULL AS 載入 PART 資料表](#)。

- 字元分隔的格式

如需有關如何使用 DELIMITER 選項的資訊，請參閱 [使用 載入 SUPPLIER 資料表 REGION](#)。

- REGION

如需有關如何使用 REGION 選項的資訊，請參閱 [使用 載入 SUPPLIER 資料表 REGION](#)。

- 固定格式寬度

如需如何從固定寬度資料載入 CUSTOMER 資料表的資訊，請參閱 [使用 載入 CUSTOMER 資料表 MANIFEST](#)。

- MAXERROR

如需有關如何使用 MAXERROR 選項的資訊，請參閱 [使用 載入 CUSTOMER 資料表 MANIFEST](#)。

- ACCEPTINVCHARS

如需有關如何使用 ACCEPTINVCHARS 選項的資訊，請參閱 [使用 載入 CUSTOMER 資料表 MANIFEST](#)。

- MANIFEST

如需有關如何使用 MANIFEST 選項的資訊，請參閱 [使用 載入 CUSTOMER 資料表 MANIFEST](#)。

- DATEFORMAT

如需有關如何使用 DATEFORMAT 選項的資訊，請參閱 [使用 載入 DWD 資料表 DATEFORMAT](#)。

- GZIP、LZOP 和 BZIP2

如需有關如何壓縮檔案的相關資訊，請參閱[使用多個檔案載入LINEORDER資料表](#)。

- COMPUPDATE

如需有關如何使用 COMPUPDATE 選項的資訊，請參閱 [使用多個檔案載入LINEORDER資料表](#)。

- 多個檔案

如需如何載入多個檔案的相關資訊，請參閱[使用多個檔案載入LINEORDER資料表](#)。

載入SSB資料表

您可以使用下列COPY命令來載入SSB結構描述中的每個資料表。每個資料表的命令會示範不同的COPY選項和疑難排解技術。

若要載入SSB資料表，請遵循下列步驟：

1. [取代儲存貯體名稱和 AWS 憑證](#)
2. [使用 NULL AS 載入PART資料表](#)
3. [使用 載入SUPPLIER資料表 REGION](#)
4. [使用 載入CUSTOMER資料表 MANIFEST](#)
5. [使用 載入DWDATE資料表 DATEFORMAT](#)
6. [使用多個檔案載入LINEORDER資料表](#)

取代儲存貯體名稱和 AWS 憑證

本教學課程中的COPY命令會以下列格式顯示。

```
copy table from 's3://<your-bucket-name>/load/key_prefix'  
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'  
options;
```

針對每個COPY命令，執行下列動作：

1. Replace (取代) *<your-bucket-name>* 與叢集位於相同區域中的儲存貯體名稱。

本步驟假設儲存貯體與叢集位於同一區域。或者，您可以使用 [REGION](#) 選項搭配 COPY 命令來指定區域。

2. Replace (取代) `<aws-account-id>` 以及 `<role-name>` 您自己的 AWS 帳戶 和 IAM 角色。以單引號括住的登入資料字串區段不可包含任何空格或分行符號。請注意，的格式ARN可能與範例略有不同。當您執行COPY命令時，最好從IAM主控台複製角色ARN的，以確保其準確。

使用 NULL AS 載入PART資料表

在此步驟中，您可以使用 CSV和 NULL AS 選項載入PART資料表。

COPY 命令可以平行載入多個檔案的資料，這比從單一檔案載入更快。為了示範這個原則，本教學課程將每個資料表中的資料分成八個檔案，即使檔案很小。在稍後的步驟中，您會比較從單一檔案與從多個檔案載入之間的時間差異。如需詳細資訊，請參閱[載入資料檔案](#)。

索引鍵字首

您可以透過為數個檔案指定索引鍵字首，或在資訊清單檔案中明確列出檔案，來從多個檔案載入資料。在此步驟中，您會使用金鑰前綴。在稍後的步驟中，您會使用資訊清單檔案。's3://amzn-s3-demo-bucket/load/part-csv.tbl' 索引鍵字首會載入 load 資料夾中的下列這些檔案。

```
part-csv.tbl-000
part-csv.tbl-001
part-csv.tbl-002
part-csv.tbl-003
part-csv.tbl-004
part-csv.tbl-005
part-csv.tbl-006
part-csv.tbl-007
```

CSV 格式

CSV代表逗號分隔值的 是用於匯入和匯出試算表資料的常見格式。CSV 比逗號分隔格式更具彈性，因為它可讓您在欄位中包含引號字串。COPY 來自 CSV 格式的預設引號字元是雙引號 (")，但您可以使用 QUOTE AS 選項指定另一個引號字元。在欄位內使用引號字元時，請多加一個引號字元來逸出此字元。

PART 下表的 CSV格式化資料檔案中摘錄顯示以雙引號 () 括住的字串 "LARGE ANODIZED BRASS"。其也顯示了引用字串中以兩個雙引號括起來的字串 ("MEDIUM ""BURNISHED"" TIN")。

```
15,dark sky,MFGR#3,MFGR#47,MFGR#3438,indigo,"LARGE ANODIZED BRASS",45,LG CASE
22,floral beige,MFGR#4,MFGR#44,MFGR#4421,medium,"PROMO, POLISHED BRASS",19,LG DRUM
```

```
23,bisque slate,MFGR#4,MFGR#41,MFGR#4137,firebrick,"MEDIUM ""BURNISHED"" TIN",42,JUMBO
JAR
```

PART 資料表的資料包含導致COPY失敗的字元。在此練習中，您會對錯誤進行故障診斷，並修正錯誤。

若要載入 CSV 格式的資料，請csv新增至您的COPY命令。執行下列命令以載入PART資料表。

```
copy part from 's3://<your-bucket-name>/load/part-csv.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
csv;
```

您可能會看到類似下列的錯誤訊息。

```
An error occurred when executing the SQL command:
copy part from 's3://amzn-s3-demo-bucket/load/part-csv.tbl'
credentials' ...

ERROR: Load into table 'part' failed. Check 'stl_load_errors' system table for
details. [SQL State=XX000]

Execution time: 1.46s

1 statement(s) failed.
1 statement(s) failed.
```

若要取得有關錯誤的詳細資訊，請查詢 STL_LOAD_ERRORS 資料表。下列查詢使用 SUBSTRING 函數來縮短資料欄的可讀性，並使用 LIMIT 10 來減少傳回的資料列數。您可以調整 substring(filename,22,25) 中的值，以符合您的儲存貯體名稱長度。

```
select query, substring(filename,22,25) as filename,line_number as line,
substring(colname,0,12) as column, type, position as pos, substring(raw_line,0,30) as
line_text,
substring(raw_field_value,0,15) as field_text,
substring(err_reason,0,45) as reason
from stl_load_errors
order by query desc
limit 10;
```

query	filename	line	column	type	pos
-----+	-----+	-----+	-----+	-----+	-----+

```

333765 | part-csv.tbl-000 | 1 | | 0 |
line_text | field_text | reason
-----+-----+-----
15,NUL next, | | Missing newline: Unexpected character 0x2c f

```

NULL AS

part-csv.tbl 資料檔案使用NUL終止字元（\x000 或 \x0）來指示NULL值。

Note

儘管拼寫非常相似，NUL但 NULL 並不相同。NUL 是 UTF-8 字元，具有通常用於指示記錄結束（）x000的編碼點EOR。NULL 是表示資料不存在SQL的值。

依預設，COPY會將NUL終止器字元視為EOR字元並終止記錄，這通常會導致意外的結果或錯誤。在文字資料NULL中沒有單一的標準指示方法。因此，NULLAS COPY命令選項可讓您指定載入資料表NULL時要取代的字元。在此範例中，您要COPY將NUL終止器字元視為NULL值。

Note

接收NULL值的資料表欄必須設定為 nullable。也就是說，它不得在CREATETABLE規格中包含NOTNULL限制條件。

若要PART使用 NULL AS 選項載入，請執行下列COPY命令。

```

copy part from 's3://<your-bucket-name>/load/part-csv.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
csv
null as '\000';

```

若要驗證COPY載入NULL的值，請執行下列命令，以僅選取包含的資料列NULL。

```

select p_partkey, p_name, p_mfgr, p_category from part where p_mfgr is null;

```

```

p_partkey | p_name | p_mfgr | p_category
-----+-----+-----+-----

```

```

15 | NUL next |      | MFGR#47
81 | NUL next |      | MFGR#23
133 | NUL next |      | MFGR#44
(2 rows)

```

使用 載入SUPPLIER資料表 REGION

在此步驟中，您可以使用 DELIMITER和 REGION選項載入SUPPLIER資料表。

Note

用於載入SUPPLIER資料表的檔案提供於 AWS 範例儲存貯體中。您不需要在這個步驟上傳檔案。

字元分隔的格式

字元分隔檔案中的欄位是以特殊字元隔開，例如縱線字元 (|)、逗號 (,) 或 Tab 字元 (\t)。字元分隔檔案可以使用任何單一ASCII字元作為分隔符號，包括其中一個非列印ASCII字元。您可以使用 DELIMITER選項指定分隔符號字元。預設分隔符號是縱線字元 (|)。

下表的資料摘錄SUPPLIER使用管道分隔格式。

```

1|1|257368|465569|41365|19950218|2-HIGH|0|17|2608718|9783671|4|2504369|92072|2|
19950331|TRUCK
1|2|257368|201928|8146|19950218|2-HIGH|0|36|6587676|9783671|9|5994785|109794|6|
19950416|MAIL

```

REGION

您應該盡可能在與 Amazon Redshift 叢集相同的 AWS 區域中找到負載資料。如果您的資料和叢集位於相同區域中，可以降低延遲，並且避免跨區域傳輸資料的成本。如需詳細資訊，請參閱 [載入資料的 Amazon Redshift 最佳實務](#)

如果您必須從不同 AWS 區域載入資料，請使用 REGION選項指定載入資料所在的 AWS 區域。如果指定區域，則所有載入資料 (包括資訊清單檔案) 都必須位於指定的區域。如需詳細資訊，請參閱 [REGION](#)。

如果您的叢集位於美國東部 (維吉尼亞北部) 區域，請執行下列命令，將SUPPLIER資料表從位於美國西部 (奧勒岡) 區域的 Amazon S3 儲存貯體中以管道分隔的資料載入。在這個範例中，請勿變更儲存貯體名稱。

```
copy supplier from 's3://awssampledwest2/ssbgz/supplier.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '|'
gzip
region 'us-west-2';
```

如果您的叢集不在美國東部（維吉尼亞北部）區域，請執行下列命令，將SUPPLIER資料表從管道分隔資料載入位於美國東部（維吉尼亞北部）區域的 Amazon S3 儲存貯體。在這個範例中，請勿變更儲存貯體名稱。

```
copy supplier from 's3://awssampledwest/ssbgz/supplier.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
delimiter '|'
gzip
region 'us-east-1';
```

使用 載入CUSTOMER資料表 MANIFEST

在此步驟中，您可以使用 FIXEDWIDTH、ACCEPTINVCHARS、MAXERROR和 MANIFEST選項來載入CUSTOMER資料表。

此練習的範例資料包含當COPY嘗試載入時會導致錯誤的字元。您可以使用 MAXERRORS選項和 STL_LOAD_ERRORS 系統資料表來疑難排解載入錯誤，然後使用 ACCEPTINVCHARS和 MANIFEST 選項來消除錯誤。

固定寬度格式

固定寬度格式將每個欄位定義為固定數量的字元，而不是使用分隔符號分隔欄位。下表的資料摘錄 CUSTOMER使用固定寬度格式。

1	Customer#000000001	IVhzIApeRb	MOROCCO	0MOROCCO	AFRICA	25-705
2	Customer#000000002	XSTf4,NCwDVaWNe6tE	JORDAN	6JORDAN	MIDDLE EAST	23-453
3	Customer#000000003	MG9kdTD	ARGENTINA5	ARGENTINA	AAMERICA	11-783

標籤/寬度配對的順序必須完全符合資料表欄的順序。如需詳細資訊，請參閱[FIXEDWIDTH](#)。

CUSTOMER 資料表資料的固定寬度規格字串如下所示。

```
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
```

```
c_region :12, c_phone:15,c_mktsegment:10'
```

若要從固定寬度資料載入CUSTOMER資料表，請執行下列命令。

```
copy customer
from 's3://<your-bucket-name>/load/customer-fw.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
c_region :12, c_phone:15,c_mktsegment:10';
```

您應該會看到類似下列的錯誤訊息。

```
An error occurred when executing the SQL command:
copy customer
from 's3://amzn-s3-demo-bucket/load/customer-fw.tbl'
credentials'...

ERROR: Load into table 'customer' failed. Check 'stl_load_errors' system table for
details. [SQL State=XX000]

Execution time: 2.95s

1 statement(s) failed.
```

MAXERROR

根據預設，第一次COPY遇到錯誤，命令會失敗並傳回錯誤訊息。為了節省測試時間，您可以使用 MAXERROR選項COPY指示在失敗之前略過指定數量的錯誤。由於我們預期第一次測試載入CUSTOMER資料表資料時會出現錯誤，因此請maxerror 10新增至 COPY 命令。

若要使用 FIXEDWIDTH和 MAXERROR選項進行測試，請執行下列命令。

```
copy customer
from 's3://<your-bucket-name>/load/customer-fw.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
c_region :12, c_phone:15,c_mktsegment:10'
maxerror 10;
```

這次，您不會看到錯誤訊息，而是得到類似下列的警示訊息。

Warnings:

Load into table 'customer' completed, 112497 record(s) loaded successfully.
 Load into table 'customer' completed, 7 record(s) could not be loaded. Check
 'stl_load_errors' system table for details.

警告表示 COPY遇到七個錯誤。若要檢查錯誤，請查詢 STL_LOAD_ERRORS 資料表，如下列範例所示。

```
select query, substring(filename,22,25) as filename,line_number as line,
substring(colname,0,12) as column, type, position as pos, substring(raw_line,0,30) as
line_text,
substring(raw_field_value,0,15) as field_text,
substring(err_reason,0,45) as error_reason
from stl_load_errors
order by query desc, filename
limit 7;
```

STL_LOAD_ERRORS 查詢的結果應類似於以下內容。

query	filename	line	column	type	pos	line_text	field_text	error_reason
334489	customer-fw.tbl.log	2	c_custkey	int4	-1	customer-fw.tbl	customer-f	Invalid digit, Value 'c', Pos 0, Type: Integ
334489	customer-fw.tbl.log	6	c_custkey	int4	-1	Complete	Complete	Invalid digit, Value 'C', Pos 0, Type: Integ
334489	customer-fw.tbl.log	3	c_custkey	int4	-1	#Total rows	#Total row	Invalid digit, Value '#', Pos 0, Type: Integ
334489	customer-fw.tbl.log	5	c_custkey	int4	-1	#Status	#Status	Invalid digit, Value '#', Pos 0, Type: Integ
334489	customer-fw.tbl.log	1	c_custkey	int4	-1	#Load file	#Load file	Invalid digit, Value '#', Pos 0, Type: Integ
334489	customer-fw.tbl000	1	c_address	varchar	34	1	Customer#000000001	.Mayag.ezR String contains invalid or unsupported UTF8
334489	customer-fw.tbl000	1	c_address	varchar	34	1	Customer#000000001	.Mayag.ezR String contains invalid or unsupported UTF8

(7 rows)

檢查結果時，您可以看到 error_reasons 資料欄中有兩則訊息：

- Invalid digit, Value '#', Pos 0, Type: Integ

這些錯誤是由 `customer-fw.tbl.log` 檔案引起。問題在於，它是日誌檔案不是資料檔案，不應載入它。您可以使用資訊清單檔案來避免載入錯誤的檔案。

- String contains invalid or unsupported UTF8

VARCHAR 資料類型支援最多三個位元組的多位元組 UTF-8 個字元。如果載入資料包含不支援或無效的字元，您可以使用 `ACCEPTINVCHARS` 選項，將每個無效的字元取代為指定的替代字元。

另一個載入的問題比較難偵測 — 載入產生非預期的結果。若要調查此問題，請執行下列命令來查詢 `CUSTOMER` 資料表。

```
select c_custkey, c_name, c_address
from customer
order by c_custkey
limit 10;
```

c_custkey	c_name	c_address
2	Customer#000000002	XSTf4,NCwDVaWNe6tE
2	Customer#000000002	XSTf4,NCwDVaWNe6tE
3	Customer#000000003	MG9kdTD
3	Customer#000000003	MG9kdTD
4	Customer#000000004	XxVSJsL
4	Customer#000000004	XxVSJsL
5	Customer#000000005	KvpyuHCplrB84WgAi
5	Customer#000000005	KvpyuHCplrB84WgAi
6	Customer#000000006	sKZz0CsnMD7mp4Xd0YrBvx
6	Customer#000000006	sKZz0CsnMD7mp4Xd0YrBvx

(10 rows)

這些資料列應該是獨一無二的，但其中有重複。

另一個檢查非預期結果的方法，是去確有已載入的資料列數量。在我們的案例中，應該載入 100000 列資料，但載入訊息指出載入 112497 個記錄。已載入額外的資料列，因為 `COPY` 已載入外部檔案 `customer-fw.tbl0000.bak`。

在這個練習中，您會使用資訊清單檔案來避免載入錯誤的檔案。

ACCEPTINVCHARS

根據預設，當COPY遇到資料欄資料類型不支援的字元時，它會略過資料列並傳回錯誤。如需有關無效 UTF-8 個字元的資訊，請參閱 [多位元組字元載入錯誤](#)。

您可以使用 MAXERRORS 選項忽略錯誤並繼續載入，然後查詢 STL_LOAD_ERRORS 以尋找無效字元，然後修正資料檔案。不過，MAXERRORS 最適合用於疑難排解負載問題，通常不應用於生產環境。

ACCEPTINVCHARS 此選項通常是管理無效字元的更好選擇。ACCEPTINVCHARS 指示COPY使用指定的有效字元取代每個無效字元，並繼續載入操作。您可以指定除 以外的任何有效ASCII字元NULL作為取代字元。預設的替代字元是問號 (?)。會以相同長度的取代字串COPY取代多位元組字元。例如，4 個位元組的字元會被取代為 '????'。

COPY 傳回包含無效 UTF-8 個字元的列數。它也會為每個受影響的資料列將項目新增至 STL_REPLACEMENTS 系統資料表，每個節點層最多 100 個資料列。其他無效 UTF-8 個字元也會被取代，但這些取代事件不會記錄下來。

ACCEPTINVCHARS 僅適用於 VARCHAR 欄。

對於此步驟，您可以使用替換字元 ACCEPTINVCHARS 來新增 '^'。

MANIFEST

當您COPY從 Amazon S3 使用金鑰字首時，您可能會載入不需要的資料表。例如，'s3://amzn-s3-demo-bucket/load/ 資料夾包含 8 個檔案，它們有相同的索引鍵字首 customer-fw.tbl : customer-fw.tbl0000、customer-fw.tbl0001 等。然而，同一資料夾中也包含無關的檔案 customer-fw.tbl.log 和 customer-fw.tbl-0001.bak。

為了確保載入所有正確的檔案，且只有正確的檔案，請使用資訊清單檔案。資訊清單是JSON格式的文字檔案，明確列出每個要載入來源檔案的唯一物件金鑰。檔案物件可以位於不同的資料夾或儲存貯體，但必須在同一個區域中。如需詳細資訊，請參閱 [MANIFEST](#)。

以下顯示 customer-fw-manifest 的文字。

```
{
  "entries": [
    {"url": "s3://<your-bucket-name>/load/customer-fw.tbl-000"},
    {"url": "s3://<your-bucket-name>/load/customer-fw.tbl-001"},
    {"url": "s3://<your-bucket-name>/load/customer-fw.tbl-002"},
```

```

{"url":"s3://<your-bucket-name>/load/customer-fw.tbl-003"},
{"url":"s3://<your-bucket-name>/load/customer-fw.tbl-004"},
{"url":"s3://<your-bucket-name>/load/customer-fw.tbl-005"},
{"url":"s3://<your-bucket-name>/load/customer-fw.tbl-006"},
{"url":"s3://<your-bucket-name>/load/customer-fw.tbl-007"}
]
}

```

使用資訊清單檔案載入CUSTOMER資料表的資料

1. 在文字編輯器中開啟 customer-fw-manifest 檔案。
2. Replace (取代) *<your-bucket-name>* 儲存貯體的名稱。
3. 儲存檔案。
4. 將檔案上傳到儲存貯體上的 load 資料夾。
5. 執行下列 COPY 命令。

```

copy customer from 's3://<your-bucket-name>/load/customer-fw-manifest'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
fixedwidth 'c_custkey:10, c_name:25, c_address:25, c_city:10, c_nation:15,
  c_region :12, c_phone:15,c_mktsegment:10'
maxerror 10
acceptinvchars as '^'
manifest;

```

使用 載入DWDATE資料表 DATEFORMAT

在此步驟中，您可以使用 DELIMITER和 DATEFORMAT選項載入DWDATE資料表。

載入 DATE 和 TIMESTAMP 欄時，COPY預期預設格式，即 YYYY-MM-DD日期的格式，以及 YYYY-MM-DD時間戳記的 HH:MI:SS。如果載入資料不使用預設格式，您可以使用 DATEFORMAT和 TIMEFORMAT 來指定格式。

下列摘錄顯示DWDATE資料表中的日期格式。注意兩個資料欄的日期格式不一致。

```

19920104 1992-01-04          Sunday January 1992 199201 Jan1992 1 4 4 1...
19920112 January 12, 1992 Monday January 1992 199201 Jan1992 2 12 12 1...
19920120 January 20, 1992 Tuesday January 1992 199201 Jan1992 3 20 20 1...

```

DATEFORMAT

您只能指定一個日期格式。如果載入資料包含不一致的格式，可能位於不同的資料欄中，或者如果載入時不知道該格式，則可以DATEFORMAT搭配'auto'引數使用。指定'auto'時，會COPY識別任何有效的日期或時間格式，並將其轉換為預設格式。'auto'選項可識別使用DATEFORMAT和TIMEFORMAT字串時不支援的多種格式。如需詳細資訊，請參閱[搭配 DATEFORMAT和 使用自動辨識 TIMEFORMAT](#)。

若要載入DWDATE資料表，請執行下列COPY命令。

```
copy dwwdate from 's3://<your-bucket-name>/load/dwwdate-tab.tbl'  
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'  
delimiter '\t'  
dateformat 'auto';
```

使用多個檔案載入LINEORDER資料表

此步驟使用 GZIP和 COMPUPDATE選項載入LINEORDER資料表。

在本練習中，您可以從單一資料檔案載入LINEORDER資料表，然後再次從多個檔案載入資料表。執行此作業可讓您比較兩個方法的載入時間。

Note

用於載入LINEORDER資料表的檔案提供於 AWS 範例儲存貯體中。您不需要在這個步驟上傳檔案。

GZIP、LZOP和 BZIP2

您可以使用 gzip、lzop 或 bzip2 壓縮格式來壓縮您的檔案。從壓縮檔案載入時，會在載入過程中COPY取消壓縮檔案。將檔案壓縮可以節省儲存空間及縮短載入時間。

COMPUPDATE

當在沒有壓縮編碼的情況下COPY載入空資料表時，它會分析載入資料以判斷最佳編碼。接著它會將資料表修改為使用這些編碼，再開始載入。這個分析程序需要一點時間，但幾乎在每個資料表都會發生一次。若要節省時間，您可以透過COMPUPDATE關閉來略過此步驟。若要啟用COPY時間的準確評估，請COMPUPDATE關閉此步驟。

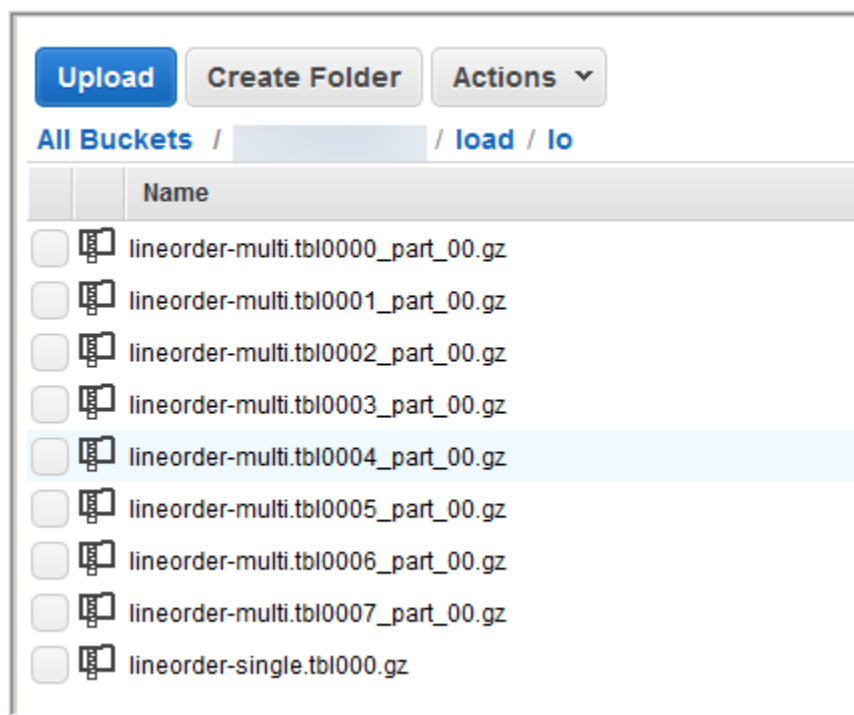
多個檔案

COPY 命令從多個檔案平行載入時，可以非常有效地載入資料，而不是從單一檔案載入。您可以將您的資料分割為檔案，使得檔案的數量為您叢集中配量數量的倍數。若您執行此作業，Amazon Redshift 會分割工作負載，並在配量中平均分配資料。每一節點的配量數目取決於叢集的節點大小。如需每個節點大小有多少配量的相關資訊，請移至《Amazon Redshift 管理指南》中的[關於叢集和節點](#)。

例如，在本教學中使用的 dc2.large 運算節點有兩個配量，所以四個叢集總共有八個配量。先前的步驟將載入的資料分成八個檔案，即使檔案很小。在這個步驟中，您會比較從單一大檔案與從多個檔案載入之間的時間差異。

您用於本教學的檔案約有 1500 萬筆記錄，大小約 1.2 GB。以 Amazon Redshift 的規模來說這些檔案很小，但足以示範從多個檔案載入的效能優點。由於在本教學中這些檔案已夠大，載入這些檔案再將其上傳到 Amazon S3 所需的時間很長。因此，您可以直接從 AWS 範例儲存貯體載入檔案。

下列螢幕擷取畫面顯示的資料檔案LINEORDER。



COPY 使用多個檔案評估的效能

1. COPY 從單一檔案對執行下列命令。請勿變更儲存貯體名稱。

```
copy lineorder from 's3://awssampledload/lo/lineorder-single.tbl'  
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'  
gzip  
compupdate off  
region 'us-east-1';
```

2. 結果類似以下這樣。請注意執行時間。

```
Warnings:
Load into table 'lineorder' completed, 14996734 record(s) loaded successfully.

0 row(s) affected.
copy executed successfully

Execution time: 51.56s
```

3. COPY 從多個檔案對 執行下列命令。請勿變更儲存貯體名稱。

```
copy lineorder from 's3://awssampledload/lo/lineorder-multi.tbl'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
gzip
compupdate off
region 'us-east-1';
```

4. 結果類似以下這樣。請注意執行時間。

```
Warnings:
Load into table 'lineorder' completed, 14996734 record(s) loaded successfully.

0 row(s) affected.
copy executed successfully

Execution time: 17.7s
```

5. 比較執行時間。

在我們的範例中，載入 1500 萬個記錄的時間從 51.56 秒降到 17.7 秒，減少了百分之 65.7。

這些結果來自於使用具有四個節點的叢集。如果您的叢集有更多節點，節省的時間會加倍。典型的 Amazon Redshift 叢集有數十到數百個節點，差別更巨大。如果您的叢集只有單一節點，執行時間的差異很小。

步驟 6：清空及分析資料庫

每當您新增、刪除或修改大量資料列時，您應該執行 VACUUM 命令，然後執行 ANALYZE 命令。vacuum (清空) 會恢復已刪除資料列的空間，並還原排序。ANALYZE 命令會更新統計資料中繼資料，讓查詢最佳化工具能夠產生更準確的查詢計劃。如需詳細資訊，請參閱[清空資料表](#)。

如果您以排序索引鍵的順序載入資料，清空很快。在本教學課程中，您已新增大量資料列，但是將它們新增到空資料表。在這種情況下，不需要重新排序，您也沒有刪除任何資料列。COPY 會在載入空資料表後自動更新統計資料，因此您的統計資料應為 up-to-date。但基於良好的內務處理，您會清空並分析資料庫以完成本教學。

若要清空並分析資料庫，執行下列命令。

```
vacuum;  
analyze;
```

步驟 7：清理您的資源

叢集只要執行就會繼續產生費用。完成本教學課程後，您應按照 Amazon Redshift 入門指南中的 [步驟 5：撤銷存取權並刪除範例叢集](#) 中的步驟，將環境恢復到先前的狀態。

如果您想要保留叢集，但復原SSB資料表使用的儲存體，請執行下列命令。

```
drop table part;  
drop table supplier;  
drop table customer;  
drop table dwdate;  
drop table lineorder;
```

下一頁

[Summary](#)

Summary

在本教學課程中，您將資料檔案上傳到 Amazon S3，然後使用COPY命令將檔案中的資料載入 Amazon Redshift 資料表。

您已使用以下格式載入資料：

- 字元分隔
- CSV
- 固定寬度

您可以使用 STL_LOAD_ERRORS 系統資料表來疑難排解載入錯誤，然後使用 REGION、MANIFEST、ACCEPTINVCHARS、DATEFORMAT 和 NULL AS MAXERROR 選項來解決錯誤。

您已運用下列最佳實務來載入資料：

- [使用 COPY 命令載入資料](#)
- [載入資料檔案](#)
- [使用單個 COPY 命令從多個文件加載](#)
- [壓縮您的資料檔案](#)
- [在載入前後驗證資料檔案](#)

如需 Amazon Redshift 最佳實務的相關資訊，請參閱下列連結：

- [載入資料的 Amazon Redshift 最佳實務](#)
- [設計資料表的 Amazon Redshift 最佳實務](#)
- [設計查詢的 Amazon Redshift 最佳實務](#)

在 Amazon Redshift 卸載數據

若要將資料從資料庫表格卸載到 Amazon S3 儲存貯體中的一組檔案，您可以將 [UNLOAD](#) 命令與 SELECT 陳述式搭配使用。您可以透過分隔的格式或是固定寬度的格式來卸載資料，無論資料載入所使用的格式為何。您也可以指定是否要建立壓縮 GZIP 檔。

您也可以使用臨時的安全性登入資料，來限制使用者對 Amazon S3 儲存貯體的存取。

主題

- [將資料卸載到 Amazon S3](#)
- [卸載加密的資料檔案](#)
- [以分隔或固定寬度的格式卸載資料](#)
- [重新載入已卸載資料](#)

將資料卸載到 Amazon S3

Amazon Redshift 會將 select 陳述式的結果分散在一組檔案中，每個節點分割中一或多個檔案，來簡化資料的平行重新載入。或者，您可以透過新增 PARALLELOFF 選項，指定 [UNLOAD](#) 應將結果依序寫入一或多個檔案。您可以透過指定 MAXFILESIZE 參數來限制 Amazon S3 中的檔案大小。UNLOAD 使用 Amazon S3 伺服器端加密 (SSE-S3) 自動加密資料檔案。

您可以在 Amazon Redshift 支援的 UNLOAD 命令中使用任何選取陳述式，但在外部選取中使用 LIMIT 子句的選取項除外。例如，您可以使用其中包含特定欄位的 select 陳述式，或使用 where 子句來聯結多個表格。如果您的查詢包含引號 (例如用來括住常值)，您需要在查詢常值中逸出它們 (\)。如需詳細資訊，請參閱 [SELECT](#) 命令參考。如需有關使用 LIMIT 子句的詳細資訊，請參閱指 UNLOAD 令的 [使用須知](#)

例如，下列 UNLOAD 命令會將 VENUE 表格的內容傳送到 Amazon S3 儲存貯體 s3://amzn-s3-demo-bucket/ticket/unload/。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/ticket/unload/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

先前範例建立的檔名包含字首 'venue_'。

```
venue_0000_part_00
```



```
venue_0001_part_00  
venue_0002_part_00  
venue_0003_part_00
```

依預設，UNLOAD會根據叢集中的磁碟片段數目，parallel 寫入資料到多個檔案。若要將資料寫入單一檔案，請指定PARALLELOFF。UNLOAD依序寫入數據，完全根據 ORDER BY 子句進行排序，如果使用的話。資料檔案大小上限為 6.2 GB。如果資料大小超過上限，則UNLOAD會建立其他檔案，每個檔案最多 6.2 GB。

下列範例會將內容VENUE寫入單一檔案。僅需要一個檔案，因為檔案大小少於 6.2 GB。

```
unload ('select * from venue')  
to 's3://amzn-s3-demo-bucket/ticket/unload/venue_'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
parallel off;
```

Note

該UNLOAD命令被設計為使用 parallel 處理。我們建議在大多數情況下保持PARALLEL啟用狀態，尤其是在使用COPY命令將使用檔案載入表格時。

假設總資料大小VENUE為 5 GB，下列範例會將內容寫入 50 個VENUE檔案，每個檔案大小為 100 MB。

```
unload ('select * from venue')  
to 's3://amzn-s3-demo-bucket/ticket/unload/venue_'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
parallel off  
maxfilesize 100 mb;
```

如果您在 Amazon S3 路徑字串中包含前置詞，檔案名稱UNLOAD會使用該前置詞。

```
unload ('select * from venue')  
to 's3://amzn-s3-demo-bucket/ticket/unload/venue_'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

您可以透過在UNLOAD命令中指定MANIFEST選項，建立列出卸載檔案的資訊清單檔案。資訊清單是JSON格式的文字檔案，可明確列出寫入 Amazon S3 URL 的每個檔案。

下列範例包含資訊清單選項。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/ticket/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

下列範例顯示四個卸載檔案的資訊清單。

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket/ticket/venue_0000_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/ticket/venue_0001_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/ticket/venue_0002_part_00"},
    {"url":"s3://amzn-s3-demo-bucket/ticket/venue_0003_part_00"}
  ]
}
```

資訊清單檔案可用來載入相同的檔案，方法是使用COPY與選MANIFEST項。如需詳細資訊，請參閱[使用資訊清單指定資料檔案](#)。

完成UNLOAD操作後，請導覽至UNLOAD寫入檔案的 Amazon S3 儲存貯體，以確認資料已正確卸載。您將會在每個分割看到一或多個含編號的檔案，從編號零開始。如果您指定了該MANIFEST選項，您還將看到一個以 'manifest' 結尾的文件。例如：

```
amzn-s3-demo-bucket/ticket/venue_0000_part_00
amzn-s3-demo-bucket/ticket/venue_0001_part_00
amzn-s3-demo-bucket/ticket/venue_0002_part_00
amzn-s3-demo-bucket/ticket/venue_0003_part_00
amzn-s3-demo-bucket/ticket/venue_manifest
```

您可以在UNLOAD完成後呼叫 Amazon S3 清單操作，以程式設計方式取得寫入 Amazon S3 的檔案清單。您也可以查詢 STL _ UNLOAD _ LOG。

下列查詢會傳回由建立的檔案的路徑名稱。UNLOAD [PG_LAST_QUERY_ID](#) 函數會傳回最近的查詢。

```
select query, substring(path,0,40) as path
from stl_unload_log
where query=2320
order by path;
```

```

query |          path
-----+-----
 2320 | s3://amzn-s3-demo-bucket/venue0000_part_00
 2320 | s3://amzn-s3-demo-bucket/venue0001_part_00
 2320 | s3://amzn-s3-demo-bucket/venue0002_part_00
 2320 | s3://amzn-s3-demo-bucket/venue0003_part_00
(4 rows)

```

如果資料量很大，Amazon Redshift 可能會根據分割將檔案分為多個部分。例如：

```

venue_0000_part_00
venue_0000_part_01
venue_0000_part_02
venue_0001_part_00
venue_0001_part_01
venue_0001_part_02
...

```

下列UNLOAD命令在 select 陳述式中包含引號字串，因此引號會逸出(=\ 'OH\ ')。

```

unload ('select venuename, venuecity from venue where venuestate=\ 'OH\ ' ')
to 's3://amzn-s3-demo-bucket/ticket/venue/ '
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';

```

依預設，UNLOAD將會失敗，而不是覆寫目標值區中的現有檔案。若要覆寫現有檔案 (包括資訊清單檔案)，請指定ALLOWOVERWRITE選項。

```

unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
allowoverwrite;

```

卸載加密的資料檔案

UNLOAD使用 Amazon S3 伺服器端加密功能自動建立檔案 AWS 受管加密金鑰 (SSE-S3)。您也可以指定伺服器端加密 AWS Key Management Service 使用客戶管理的金鑰進行 key (SSE-KMS) 或用戶端加密。UNLOAD不支援使用客戶受管金鑰的 Amazon S3 伺服器端加密。如需詳細資訊，請參閱[使用伺服器端加密保護資料](#)。

若要使用 Amazon S3 器端加密與 AWS KMS key 時，請使用 KMS_KEY_ID 參數來提供索引鍵 ID，如下列範例所示。

```
unload ('select venueName, venueCity from venue')
to 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
KMS_KEY_ID '1234abcd-12ab-34cd-56ef-1234567890ab'
encrypted;
```

如果您想要提供自己的加密金鑰，可以使用 UNLOAD 指令搭配 ENCRYPTED 選項，在 Amazon S3 中建立用戶端加密資料檔案。UNLOAD 使用 Amazon S3 用戶端加密使用的相同信封加密程序。然後，您可以使用該 COPY 命令和 ENCRYPTED 選項來加載加密文件。

運作程序如下：

1. 您建立了一個 base64 編碼的 256 位元 AES 金鑰，該金鑰將用作私密加密金鑰或根對稱金鑰。
2. 您發出包含根對稱金鑰和 ENCRYPTED 選項的 UNLOAD 命令。
3. UNLOAD 生成一個 one-time-use 對稱密鑰（稱為信封對稱密鑰）和初始化向量（IV），它用來加密數據。
4. UNLOAD 使用根對稱金鑰加密信封對稱金鑰。
5. UNLOAD 然後將加密的資料檔案存放在 Amazon S3，並將加密的信封金鑰和 IV 做為物件中繼資料存放在每個檔案中。加密信封金鑰會儲存做為物件中繼資料 x-amz-meta-x-amz-key 且 IV 會儲存做為物件中繼資料 x-amz-meta-x-amz-iv。

如需有關信封加密程序的詳細資訊，請參閱 [用戶端資料加密 AWS SDK 對於 Java 和 Amazon S3](#) 文章。

若要卸載加密的資料檔案，請將根金鑰值新增至認證字串，並包含選 ENCRYPTED 項。如果您使用此選 MANIFEST 項，資訊清單檔案也會加密。

```
unload ('select venueName, venueCity from venue')
to 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
manifest
encrypted;
```

若要卸載 GZIP 壓縮的加密資料檔案，請包括 GZIP 選項以及根金鑰值和選 ENCRYPTED 項。

```
unload ('select venueName, venueCity from venue')
to 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted gzip;
```

若要載入加密的資料檔案，請新增具有相同根金鑰值的 MASTER SYMMETRIC __ KEY 參數，並包含選 ENCRYPTED 項。

```
copy venue from 's3://amzn-s3-demo-bucket/encrypted/venue_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key '<root_key>'
encrypted;
```

以分隔或固定寬度的格式卸載資料

您可以分隔或固定寬度的格式來卸載資料。預設輸出是以縱線分隔 (使用 '|' 字元)。

以下範例將逗點指定為分隔符號：

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/ticket/venue/comma'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter ',';
```

產生的輸出看起來像這樣：

```
20,Air Canada Centre,Toronto,ON,0
60,Rexall Place,Edmonton,AB,0
100,U.S. Cellular Field,Chicago,IL,40615
200,Al Hirschfeld Theatre,New York City,NY,0
240,San Jose Repertory Theatre,San Jose,CA,0
300,Kennedy Center Opera House,Washington,DC,0
...
```

若要卸載設為以 Tab 鍵分隔之檔案的相同結果，請發出下列命令：

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/ticket/venue/tab'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
```

```
delimiter as '\t';
```

或者，您可以使用FIXEDWIDTH規格。此規格包含每個表格欄位和欄位寬度的識別碼 (字元數)。該UNLOAD命令將失敗，而不是截斷數據，因此請指定至少與該列最長條目相同的寬度。卸載固定寬度的資料的運作方式與卸載分隔資料的方式類似，差別在於產生的輸出未包含分隔字元。例如：

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/ticket/venue/fw'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth '0:3,1:100,2:30,3:2,4:6';
```

固定寬度的輸出看起來如下：

```
20 Air Canada Centre      Toronto      ON0
60 Rexall Place           Edmonton    AB0
100U.S. Cellular Field   Chicago     IL40615
200Al Hirschfeld Theatre New York CityNY0
240San Jose Repertory TheatreSan Jose     CA0
300Kennedy Center Opera HouseWashington    DC0
```

如需有關FIXEDWIDTH規格的詳細資訊，請參閱[UNLOAD](#)指令。

重新載入已卸載資料

若要重新載入卸載作業的結果，您可以使用指COPY令。

下列範例顯示使用資訊清單檔案卸載資VENUE料表、截斷並重新載入的簡單案例。

```
unload ('select * from venue order by venueid')
to 's3://amzn-s3-demo-bucket/ticket/venue/reload_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
delimiter '|';

truncate venue;

copy venue
from 's3://amzn-s3-demo-bucket/ticket/venue/reload_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest
```

```
delimiter '|';
```

重新載入之後，VENUE表格如下所示：

```
select * from venue order by venueid limit 5;
```

venueid	venue name	venue city	venue state	venue seats
1	Toyota Park	Bridgeview	IL	0
2	Columbus Crew Stadium	Columbus	OH	0
3	RFK Stadium	Washington	DC	0
4	CommunityAmerica Ballpark	Kansas City	KS	0
5	Gillette Stadium	Foxborough	MA	68756

(5 rows)

Amazon Redshift 中的使用者定義函數

您可以使用子SQLSELECT句或 Python 程式建立自訂純量使用者定義函數 (UDF)。新函數儲存於資料庫，可供具有足夠權限的任何使用者執行。您執行自訂純量的方式與執行現有 Amazon Redshift 函數UDF的方式大致相同。

對於 Python UDFs，除了使用標準 Python 功能之外，您還可以匯入自己的自訂 Python 模組。如需詳細資訊，請參閱的 [Python 語言支援 UDFs](#)。請注意，Python 3 不適用於 Python UDFs。若要取得 Amazon Redshift 的 Python 3 支援UDFs，請[Scalar Lambda UDFs](#)改用。

您也可以建立 AWS Lambda UDFs使用 Lambda 中定義的自訂函數作為SQL查詢的一部分。Lambda UDFs可讓您撰寫複雜內容，UDFs並與第三方元件整合。他們也可以協助您克服目前 Python 和 SQL 的一些限制UDFs。例如，它們可協助您存取網路和儲存資源，並撰寫更多完整的SQL 陳述式。您可以使用 Lambda 支援UDFs的任何程式設計語言來建立 Lambda，例如 Java、Go PowerShell、Node.js、C#、Python 和 Ruby。或者，您也可以使用自訂執行期。

根據預設，所有使用者都可以執行 UDFs。如需權限的相關資訊，請參閱 [UDF 安全性和許可](#)。

主題

- [UDF 安全性和許可](#)
- [防止UDF命名衝突](#)
- [純量 SQL UDFs](#)
- [Scalar Python UDFs](#)
- [Scalar Lambda UDFs](#)
- [使用者定義函數 \(UDFs \) 的使用案例範例](#)

UDF 安全性和許可

若要建立 UDF，您必須具有 SQL或 ppythonu (Python) 使用語言的許可。根據預設，USAGEON LANGUAGESQL會授予 PUBLIC，但您必須明確授予 USAGE ON LANGUAGEPLPYTHONU給特定使用者或群組。

若要撤銷的使用SQL，請先撤銷的使用PUBLIC。然後SQL，僅將上的用量授予允許建立 SQL 的特定使用者或群組UDFs。下列範例SQL會從 撤銷對的使用PUBLIC。然後，它會將使用權授予使用者群組 udf_devs。

```
revoke usage on language sql from PUBLIC;
```



```
grant usage on language sql to group udf_devs;
```

若要執行 UDF，您必須具有為每個函數執行此操作的許可。根據預設，UDFs 會授予執行新的許可 PUBLIC。若要限制用量，請從撤銷函數 PUBLIC 的此許可。然後將許可授予特定個人或群組。

下列範例 `f_py_greater` 會從撤銷對函數的執行 PUBLIC。然後，它會將使用權授予使用者群組 `udf_devs`。

```
revoke execute on function f_py_greater(a float, b float) from PUBLIC;  
grant execute on function f_py_greater(a float, b float) to group udf_devs;
```

根據預設，超級使用者具備所有權限。

如需詳細資訊，請參閱 [GRANT](#) 和 [REVOKE](#)。

防止UDF命名衝突

在實作之前，您可以考慮您的UDF命名慣例，以避免潛在的衝突和非預期的結果。因為函數名稱可以過載，所以它們可能與現有和未來的 Amazon Redshift 函數名稱發生衝突。本主題討論過載，並說明避免衝突的策略。

多載函數名稱

函數是以其名稱和簽章來識別，而簽章是輸入引數的數目和引數的資料類型。如果相同結構描述中的兩個函數具有不同的簽章，則它們可以具有相同的名稱。換言之，函數名稱可以過載。

當您執行查詢時，查詢引擎會根據您提供的引數數目和引數的資料類型來決定要呼叫哪個函數。您可以使用過載來模擬具有可變引數數目的函數，而此數目最多可為 [CREATE FUNCTION](#) 命令允許的限制。

避免與內建 Amazon Redshift 函數衝突

建議您UDFs使用字首命名所有 `f_`。Amazon Redshift 會保留專屬於UDFs的字首 `f_`，並以為UDF名稱加上字首 `f_`，可確保UDF您的姓名不會與任何現有或未來的 Amazon Redshift 內建SQL函數名稱衝突。例如，透過命名新的UDF `f_sum`，您可以避免與 Amazon Redshift SUM函數衝突。同樣地，如果您命名新的函數 `f_fibonacci`，則如果 Amazon Redshift 新增名為 `FIBONACCI` 的函數於未來的版本 `FIBONACCI` 中，則避免衝突。

如果和內建SQL函數存在於不同的結構描述中，您可以建立UDF與現有 Amazon Redshift 內建函數具有相同名稱UDF和簽章的，而不會使函數名稱過載。由於系統目錄結構描述 `pg_catalog` 中存在內建函

數，因此您可以在另一個結構描述中建立UDF具有相同名稱的，例如公有或使用者定義的結構描述。在某些情況下，您可能會呼叫未使用結構描述名稱明確限定的函數。如果是這樣，Amazon Redshift 預設會先搜尋 `pg_catalog` 結構描述。因此，內建函數會在UDF具有相同名稱的新之前執行。

您可以透過將搜尋路徑設定為將 `pg_catalog` 放在末尾來變更此行為。如果您這樣做，您的UDFs優先於內建函數，但此做法可能會導致非預期的結果。採用唯一命名策略 (例如使用保留字首 `f_`) 是更為可靠的實務。如需詳細資訊，請參閱 [SET](#) 和 [search_path](#)。

純量 SQL UDFs

純量SQLUDF包含子句，該SQLSELECT子句會在呼叫函數時執行，並傳回單一值。[CREATE FUNCTION](#) 命令定義下列參數：

- (選用) 輸入引數。每個引數必須具有一個資料類型。
- 一個傳回資料類型。
- 一個SQLSELECT子句。在SELECT子句中，根據函數定義中的引數順序，使用 `$1`、`$2` 等參考輸入引數。

輸入和傳回資料類型可以是任何標準 Amazon Redshift 資料類型。

請勿在FROM子句中包含SELECT子句。相反地，請在呼叫 SQL 的 SQL陳述式中包含 FROM子句UDF。

SELECT 子句不能包含下列任何類型的子句：

- FROM
- INTO
- WHERE
- GROUP 依
- ORDER 依
- LIMIT

純量SQL函數範例

下列範例會建立一個函數，比較兩個數字並傳回較大的值。如需詳細資訊，請參閱[CREATE FUNCTION](#)。

```
create function f_sql_greater (float, float)
  returns float
  stable
as $$
  select case when $1 > $2 then $1
    else $2
  end
$$ language sql;
```

下列查詢會呼叫新的 `f_sql_greater` 函數來查詢SALES資料表，並傳回 COMMISSION或 20%PRICEPAID，以較大者為準。

```
select f_sql_greater(commission, pricepaid*0.20) from sales;
```

Scalar Python UDFs

純量 Python UDF包含 Python 程式，會在呼叫函數時執行，並傳回單一值。[CREATE FUNCTION](#) 命令定義下列參數：

- (選用) 輸入引數。每個引數必須具有一個名稱和一個資料類型。
- 一個傳回資料類型。
- 一個可執行的 Python 程式。

輸入和傳回資料類型可以是

SMALLINT、INTEGER、BIGINT、DECIMALREAL、DOUBLEPRECISION、BOOLEAN、CHAR、VARCHAR DATE或 TIMESTAMP。此外，Python UDFs可以使用資料類型 ANYELEMENT，Amazon Redshift 會根據執行階段提供的引數自動轉換為標準資料類型。如需詳細資訊，請參閱 [ANYELEMENT 資料類型](#)

當 Amazon Redshift 查詢呼叫純量 時UDF，執行階段會發生下列步驟：

1. 函數會將輸入引數轉換為 Python 資料類型。

如需將 Amazon Redshift 資料類型映射至 Python 資料類型，請參閱 [Python UDF 資料類型](#)。

2. 函數會執行 Python 程式，剖析已轉換的輸入引數。
3. Python 程式碼會傳回單一值。傳回值的資料類型必須與函數定義指定的RETURNS資料類型對應。
4. 函數會將 Python 傳回值轉換為指定的 Amazon Redshift 資料類型，然後將該值傳回至佇列。

Note

Python 3 不適用於 Python UDFs。若要取得 Amazon Redshift 的 Python 3 支援UDFs，請[Scalar Lambda UDFs](#)改用。

Scalar Python UDF範例

下列範例會建立一個函數，比較兩個數字並傳回較大的值。請注意，雙貨幣符號 (\$\$) 之間程式碼的縮排是 Python 需求。如需詳細資訊，請參閱[CREATE FUNCTION](#)。

```
create function f_py_greater (a float, b float)
  returns float
stable
as $$
  if a > b:
    return a
  return b
$$ language plpythonu;
```

下列查詢會呼叫新f_greater函數來查詢SALES資料表，並傳回 COMMISSION或20%PRICEPAID，以較大者為準。

```
select f_py_greater (commission, pricepaid*0.20) from sales;
```

Python UDF 資料類型

Python UDFs可以使用任何標準 Amazon Redshift 資料類型作為輸入引數和函數的傳回值。除了標準資料類型之外，UDFs還支援資料類型 ANYELEMENT，Amazon Redshift 會根據執行時提供的引數自動轉換為標準資料類型。Scalar UDFs可以傳回資料類型 ANYELEMENT。如需詳細資訊，請參閱[ANYELEMENT 資料類型](#)。

在執行期間，Amazon Redshift 會將引數從 Amazon Redshift 資料類型轉換為 Python 資料類型以進行處理。然後，它將傳回值從 Python 資料類型轉換為對應的 Amazon Redshift 資料類型。如需 Amazon Redshift 資料類型的相關資訊，請參閱[資料類型](#)。

下表將 Amazon Redshift 資料類型映射至 Python 資料類型。

Amazon Redshift 資料類型	Python 資料類型
smallint	int
integer	
bigint	
short	
長整數	
decimal 或 numeric	decimal
double	float
real	
布林值	bool
char	string
varchar	
timestamp	datetime

ANYELEMENT 資料類型

ANYELEMENT 是多態資料類型。這表示如果使用 宣告引數資料類型ANYELEMENT的函數，則當呼叫函數時，該函數可接受任何標準 Amazon Redshift 資料類型作為該引數的輸入。ANYELEMENT 引數會設定為呼叫函數時實際傳遞的資料類型。

如果函數使用多種ANYELEMENT資料類型，則在呼叫函數時，它們都必須解析為相同的實際資料類型。所有ANYELEMENT引數資料類型都會設定為傳遞給 的第一個引數的實際資料類型 ANYELEMENT。例如，宣告為 `f_equal(anelement, anelement)` 的函數將採用任何兩個輸入值，只要它們是相同的資料類型。

如果函數的傳回值宣告為 ANYELEMENT，則至少有一個輸入引數必須為 ANYELEMENT。傳回值的實際資料類型與ANYELEMENT輸入引數提供的實際資料類型相同。

的 Python 語言支援 UDFs

您可以UDF根據 Python 程式設計語言建立自訂。[Python 2.7 標準程式庫](#)可在 中使用UDFs，但下列模組除外：

- ScrolledText
- Tix
- Tkinter
- tk
- turtle
- smtpd

除了 Python 標準程式庫外，下列模組是 Amazon Redshift 實作的一部分：

- [numpy 1.8.2](#)
- [pandas 0.14.1](#)
- [python-dateutil 2.2](#)
- [pytz 2014.7](#)
- [scipy 0.12.1](#)
- [six 1.3.0](#)
- [wsgiref 0.1.2](#)

您也可以匯入自己的自訂 Python 模組，並透過UDFs執行[CREATE LIBRARY](#)命令使其可用於。如需詳細資訊，請參閱[範例：匯入自訂 Python 程式庫模組](#)。

Important

Amazon Redshift 會封鎖透過 對檔案系統的所有網路存取和寫入存取UDFs。

Note

Python 3 不適用於 Python UDFs。若要取得 Amazon Redshift 的 Python 3 支援UDFs，請[Scalar Lambda UDFs](#)改用。

範例：匯入自訂 Python 程式庫模組

您可以使用 Python 語言語法來定義純量函數。您可以使用 Python 標準程式庫模組和 Amazon Redshift 預先安裝模組。您也可以建立自己的自訂 Python 程式庫模組，並將程式庫匯入叢集，或使用 Python 或第三方的現有程式庫。

您建立的程式庫不可包含與 Python 標準程式庫模組或 Amazon Redshift 預先安裝 Python 模組同名的模組。如果現有使用者安裝的程式庫使用相同的 Python 套件做為您建立的程式庫，則您必須捨棄現有的程式庫，然後才能安裝新的程式庫。

您必須是超級使用者或具有 USAGE ON LANGUAGE plpythonu 權限，才能安裝自訂程式庫；不過，任何具有足夠權限來建立函數的使用者都可以使用已安裝的程式庫。您可以查詢 [PG_LIBRARY](#) 系統目錄，來檢視叢集上已安裝之程式庫的相關資訊。

將自訂 Python 模組匯入叢集

本節提供將自訂 Python 模組匯入至您的叢集的範例。若要執行本節中的步驟，您必須具有 Amazon S3 儲存貯體，而您會在其中上傳程式庫套件。然後，在您的叢集中安裝套件。如需建立儲存貯體的相關資訊，請前往《Amazon Simple Storage Service 使用者指南》中的[建立儲存貯體](#)。

在此範例中，假設您建立 UDFs 來使用資料中的位置和距離。從SQL用戶端工具連線至 Amazon Redshift 叢集，然後執行下列命令來建立函數。

```
CREATE FUNCTION f_distance (x1 float, y1 float, x2 float, y2 float) RETURNS float
IMMUTABLE as $$
    def distance(x1, y1, x2, y2):
        import math
        return math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2)

    return distance(x1, y1, x2, y2)
$$ LANGUAGE plpythonu;

CREATE FUNCTION f_within_range (x1 float, y1 float, x2 float, y2 float) RETURNS bool
IMMUTABLE as $$
    def distance(x1, y1, x2, y2):
        import math
        return math.sqrt((y2 - y1) ** 2 + (x2 - x1) ** 2)

    return distance(x1, y1, x2, y2) < 20
$$ LANGUAGE plpythonu;
```

請注意，會複製先前函數中的幾行程式碼。此重複是必要的，因為 UDF 無法參考另一個的內容 UDF，而且兩個函數都需要相同的功能。然而，不是複製多個函數中的程式碼，而是您可以建立自訂程式庫，並設定您的函數來使用它。

若要這樣做，首先遵循下列步驟來建立程式庫套件：

1. 建立名為 `geometry` 的資料夾。此資料夾是程式庫的最上層套件。
2. 在 `geometry` 資料夾中，建立名為 `__init__.py` 的檔案。請注意，檔案名稱包含兩個雙底線字元。此檔案向 Python 表示可以初始化套件。
3. 另外，在 `geometry` 資料夾中，建立名為 `trig` 的資料夾。此資料夾是程式庫的子套件。
4. 在 `trig` 資料夾中，建立另一個名為 `__init__.py` 的檔案，以及一個名為 `line.py` 的檔案。在此資料夾中，`__init__.py` 向 Python 表示可以初始化子套件，而且 `line.py` 是包含程式庫程式碼的檔案。

您的資料夾和檔案結構應該與下列資料夾和檔案結構相同：

```
geometry/  
  __init__.py  
  trig/  
    __init__.py  
    line.py
```

如需套件結構的相關資訊，請前往 Python 網站上 Python 教學課程中的[模組](#)。

5. 下列程式碼包含程式庫的類別和成員函數。複製它並貼至 `line.py`。

```
class LineSegment:  
    def __init__(self, x1, y1, x2, y2):  
        self.x1 = x1  
        self.y1 = y1  
        self.x2 = x2  
        self.y2 = y2  
    def angle(self):  
        import math  
        return math.atan2(self.y2 - self.y1, self.x2 - self.x1)  
    def distance(self):  
        import math  
        return math.sqrt((self.y2 - self.y1) ** 2 + (self.x2 - self.x1) ** 2)
```


在您建立了套件之後，請執行下列動作，來準備套件並將其上傳至 Amazon S3。

1. 將 geometry 資料夾的內容壓縮成名為 geometry.zip 的 .zip 檔案。不要包括 geometry 資料夾本身；只包括資料夾的內容，如下所示：

```
geometry.zip
  __init__.py
  trig/
    __init__.py
    line.py
```

2. 將 geometry.zip 上傳至您的 Amazon S3 儲存貯體。

Important

如果 Amazon S3 儲存貯體與 Amazon Redshift 叢集不位於同一個區域，您必須使用 REGION 選項來指定資料所在的區域。如需詳細資訊，請參閱 [CREATE LIBRARY](#)。

3. 從 SQL 用戶端工具中，執行下列命令來安裝程式庫。Replace (取代) `<bucket_name>` 使用儲存貯體的名稱，並取代 `<access key id>` 以及 `<secret key>` 存取金鑰和來自 AWS Identity and Access Management (IAM) 使用者憑證的秘密存取金鑰。

```
CREATE LIBRARY geometry LANGUAGE plpythonu FROM 's3://<bucket_name>/geometry.zip'
  CREDENTIALS 'aws_access_key_id=<access key id>;aws_secret_access_key=<secret key>;';
```

在您的叢集中安裝程式庫之後，您需要設定函數來使用程式庫。若要這樣做，請執行下列命令。

```
CREATE OR REPLACE FUNCTION f_distance (x1 float, y1 float, x2 float, y2 float) RETURNS
float IMMUTABLE as $$
  from trig.line import LineSegment

  return LineSegment(x1, y1, x2, y2).distance()
$$ LANGUAGE plpythonu;

CREATE OR REPLACE FUNCTION f_within_range (x1 float, y1 float, x2 float, y2 float)
RETURNS bool IMMUTABLE as $$
  from trig.line import LineSegment

  return LineSegment(x1, y1, x2, y2).distance() < 20
$$ LANGUAGE plpythonu;
```

在上述命令中，`import trig/line` 會從本節中的原始函數中刪除重複的程式碼。您可以在多個中重複使用此程式庫提供的功能UDFs。請注意，若要匯入模組，您只需要指定子套件和模組名稱的路徑 (`trig/line`)。

Python UDF限制條件

在此主題中列出的限制中，您可以在任何使用 Amazon Redshift 內建純量函數UDFs的地方使用。如需詳細資訊，請參閱[SQL 函數參考](#)。

Amazon Redshift Python UDFs具有下列限制條件：

- Python UDFs無法存取網路或讀取或寫入檔案系統。
- 使用者安裝的 Python 程式庫的總和大小不得超過 100 MB。
- 對於使用自動工作負載管理 (WLM) 的佈建叢集，以及無伺服器工作群組，Amazon Redshift UDF 一次只能執行一個 Python。如果您嘗試UDF同時執行多個 Python，Amazon Redshift 會將剩餘的 Python 排入佇列，UDFs以便在工作負載管理佇列中執行。SQL UDFs 使用自動時，沒有並行限制 WLM。
- 使用手動WLM佈建叢集時，每個叢集UDFs可同時執行的 Python 數量限制為叢集並行總層級的四分之一。例如，並行為 15 的佈建叢集最多可以執行三個並行 Python UDFs。
- 使用 Python 時UDFs，Amazon Redshift 不支援 SUPER和 HLLSKETCH 資料類型。

在 Python 中記錄錯誤和警告 UDFs

您可以使用 Python 記錄模組，在中建立使用者定義的錯誤和警告訊息UDFs。您可以在查詢執行之後，查詢 [SVL_UDF_LOG](#) 系統檢視來擷取日誌記錄的訊息。

Note

UDF 記錄會耗用叢集資源，並可能影響系統效能。我們建議只針對開發和故障診斷實作記錄。

在查詢執行期間，日誌處理常式會將訊息寫入 `SVL_UDF_LOG` 系統檢視，以及對應的函數名稱、節點和層。日誌處理常式會為每個切片將一列寫入每則訊息的 `SVL_UDF_LOG`。訊息會截斷至 4096 個位元組。UDF 日誌限制為每個切片 500 列。當日誌已滿時，日誌處理常式會捨棄舊訊息，並將警告訊息新增至 `SVL_UDF_LOG`。

Note

Amazon Redshift UDF 日誌處理常式會逸出具有反斜線 \n () 字元的新行 ()、管道 | () 字元和反斜線 \ (\) 字元。

根據預設，UDF 日誌層級會設定為 WARNING。CRITICAL 會記錄日誌層級為 WARNING、ERROR 和的訊息。嚴重性較低的訊息 DEBUGINFO、和 NOTSET 會被忽略。若要設定 UDF 日誌層級，請使用 Python Logger 方法。例如，下列 會將日誌層級設定為 INFO。

```
logger.setLevel(logging.INFO)
```

如需使用 Python 記錄模組的相關資訊，請參閱 Python 文件中的 [Python 的記錄機能](#)。

下列範例會建立名為 f_pyerror 的函數，用來匯入 Python 記錄模組、將記錄器執行個體化，以及記錄錯誤。

```
CREATE OR REPLACE FUNCTION f_pyerror()
RETURNS INTEGER
VOLATILE AS
$$
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)
logger.info('Your info message here')
return 0
$$ language plpythonu;
```

下列範例查詢 SVL_UDF_LOG 以檢視在上一個範例中記錄的訊息。

```
select funcname, node, slice, trim(message) as message
from svl_udf_log;
```

funcname	query	node	slice	message
f_pyerror	12345	1	1	Your info message here

Scalar Lambda UDFs

Amazon Redshift 可以使用中定義的自訂函數 AWS Lambda 作為 SQL 查詢的一部分。您可以使用 Lambda 支援 UDFs 的任何程式設計語言撰寫純量 Lambda，例如 Java、Go、PowerShell、Node.js、C#、Python 和 Ruby。或者，您也可以使用自訂執行期。

[CREATE EXTERNAL FUNCTION](#) 命令會建立下列參數：

- (選用) 具有資料類型的引數清單。
- 一個傳回資料類型。
- 由 Amazon Redshift 呼叫之外部函數的一個函數名稱。
- Amazon Redshift 叢集有權擔任並呼叫 Lambda 的一個 IAM 角色。
- Lambda UDF 調用的一個 Lambda 函數名稱。

如需 CREATE EXTERNAL 的相關資訊 FUNCTION，請參閱 [CREATE EXTERNAL FUNCTION](#)。

此函數的輸入和傳回資料類型可以是任何標準 Amazon Redshift 資料類型。

Amazon Redshift 可確保外部函數可以傳送和接收批次引數和結果。

Lambda UDFs 是在 Lambda 中定義和管理的，您可以控制在 Amazon Redshift UDFs 中調用這些權限的存取權限。您可以在同一個查詢中調用多個 Lambda 函數，或多次調用相同的函數。

在支援純量函數的 SQL 陳述式的任何子句 UDFs 中使用 Lambda。您也可以在任何 SQL 陳述 UDFs 式中使用 Lambda UPDATE，例如 SELECT、INSERT、或 DELETE。

Note

使用 Lambda UDFs 可能會從 Lambda 服務產生額外費用。它是否這樣做取決於 Lambda 請求數（UDF 調用）和 Lambda 程式執行的總持續時間等因素。不過，在 Amazon Redshift UDFs 中使用 Lambda 無需額外費用。如需有關 AWS Lambda 定價的資訊，請參閱 [AWS Lambda 定價](#)。

Lambda 請求的數量取決於 UDF 使用 Lambda 的特定 SQL 陳述式子句。例如，假設函數用於如下所示的 WHERE 子句中。

```
SELECT a, b FROM t1 WHERE lambda_multiply(a, b) = 64; SELECT a, b  
FROM t1 WHERE a*b = lambda_multiply(2, 32)
```

在此情況下，Amazon Redshift 會呼叫每個的第一個 SELECT 陳述式，並只呼叫第二個 SELECT 陳述式一次。

不過，在查詢的投影部分UDF中使用 `lambda`，可能只會針對結果集中的每個合格或彙總資料列調用 `lambda` 函數一次。

UDF 安全性和許可

若要建立 `lambda` UDF，請確定您有在 `lambda` 上使用 `LANGUAGE EXFUNC` 的許可。您必須明確將 `USAGE ON LANGUAGE EXFUNC` 或撤銷 `USAGE ON LANGUAGE EXFUNC` 授予特定使用者、群組或公有使用者。

下列範例 `EXFUNC` 會將 `lambda` 的使用量授予 `PUBLIC`。

```
grant usage on language exfunc to PUBLIC;
```

下列範例會從 `exfunc` 撤銷用量，`PUBLIC` 然後將用量授與使用者群組 `lambda_udf_devs`。

```
revoke usage on language exfunc from PUBLIC;  
grant usage on language exfunc to group lambda_udf_devs;
```

若要執行 `lambda` UDF，請確定您擁有呼叫的每個函數的許可。根據預設，UDFs 會授予執行新 `lambda` 的許可。`PUBLIC` 若要限制用量，請從 `lambda` 撤銷函數 `PUBLIC` 的此許可。然後將權限授予特定使用者或群組。

下列範例會從 `lambda` 撤銷對函數 `exfunc_sum` 的執行 `PUBLIC`。然後，它會將使用權授予使用者群組 `lambda_udf_devs`。

```
revoke execute on function exfunc_sum(int, int) from PUBLIC;  
grant execute on function exfunc_sum(int, int) to group lambda_udf_devs;
```

根據預設，超級使用者具備所有權限。

如需授予和撤銷權限的相關資訊，請參閱 [GRANT](#) 和 [REVOKE](#)。

設定 `lambda` 的授權參數 UDFs

`CREATE EXTERNAL FUNCTION` 命令需要授權才能在中叫用 `lambda` 函數 `AWS Lambda`。若要啟動授權，請在執行 `CREATE EXTERNAL FUNCTION` 命令時指定 `AWS Identity and Access Management (IAM)` 角色。如需 `IAM` 角色的詳細資訊，請參閱 `IAM 使用者指南` 中的 [IAM 角色](#)。

如果現有IAM角色具有呼叫連接至叢集之 Lambda 函數的許可，您可以在 IAM_ROLE 參數中將角色 Amazon Resource Name (ARN) 取代為 命令。下列各節說明在 CREATE EXTERNAL FUNCTION 命令中使用IAM角色的步驟。

為 Lambda 建立IAM角色

此IAM角色需要許可才能叫用 Lambda 函數。建立IAM角色時，會以下列其中一種方式提供許可：

- 在建立IAM角色時，在連接許可AWSLambdaRole政策頁面上連接政策。AWSLambdaRole 政策授予調用 Lambda 函數的許可，這是最低要求。如需詳細資訊和其他政策，請參閱 AWS Lambda 開發人員指南 中的[適用於的身分型IAM政策 AWS Lambda](#)。
- 建立您自己的自訂政策，透過所有資源或特定 Lambda 函數的 `lambda:InvokeFunction` 許可連接至您的IAM角色ARN。如需如何建立政策的詳細資訊，請參閱 IAM 使用者指南 中的[建立IAM政策](#)。

下列範例政策可讓您在特定 Lambda 函數上調用 Lambda。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Invoke",
      "Effect": "Allow",
      "Action": [
        "lambda:InvokeFunction"
      ],
      "Resource": "arn:aws:lambda:us-west-2:123456789012:function:my-function"
    }
  ]
}
```

如需 Lambda 函數資源的詳細資訊，請參閱 IAM API 參考 中的 [Lambda 動作的資源和條件](#)。

建立具有必要許可的自訂政策後，您可以在建立IAM角色時，將政策連接至連接許可政策頁面上IAM的角色。

如需建立IAM角色的步驟，請參閱 [Amazon Redshift 管理指南 中的授權 Amazon Redshift 代表您存取其他 AWS 服務](#)。

如果您不想建立新IAM角色，您可以將先前提到的許可新增至現有IAM角色。

將IAM角色與叢集建立關聯

將IAM角色連接至叢集。您可以使用 Amazon Redshift 管理主控台CLI、或將角色新增至叢集，或檢視與叢集相關聯的角色API。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的[將IAM角色與叢集建立關聯](#)。

在命令中包含IAM角色

在CREATEEXTERNALFUNCTION命令ARN中包含IAM角色。當您建立IAM角色時，會IAM傳回角色的 Amazon Resource Name (ARN)。若要指定IAM角色，ARN請將 IAM_ROLE 參數提供給角色。以下顯示 IAM_ROLE 參數的語法。

```
IAM_ROLE 'arn:aws:iam::aws-account-id:role/role-name'
```

若要叫用位於相同區域內其他帳戶的 Lambda 函數，請參閱 [Amazon Redshift 中的鏈結IAM角色](#)。

使用 Amazon Redshift 和 之間的JSON介面 AWS Lambda

Amazon Redshift 對 Amazon Redshift 進行通訊的所有 Lambda 函數使用通用介面。

下表顯示您可以預期JSON承載的指定 Lambda 函數的輸入欄位清單。

欄位名稱	描述	值範圍
request_id	通用唯一識別碼 (UUID)，可唯一識別每個調用請求。	有效的 UUID。
叢集	叢集的完整 Amazon Resource Name (ARN)。	有效的叢集 ARN。
使用者	進行呼叫的使用者名稱。	有效的使用者名稱。
database	執行查詢之資料庫的名稱。	有效的資料庫名稱。
external_function	進行呼叫之外部函數的完整名稱。	有效的完整函數名稱。

欄位名稱	描述	值範圍
query_id	進行呼叫之查詢的查詢 ID。	有效的查詢 ID。
num_records	承載中的引數數目。	值為 1 - 2 ⁶⁴ 。
引數	指定格式的資料承載。	陣列格式的資料必須是JSON陣列。如果引數數目大於 1，則每個元素都是一個記錄，該記錄是一個陣列。透過使用陣列，Amazon Redshift 會保留承載中記錄的順序。

JSON 陣列的順序決定批次處理的順序。Lambda 函數必須反覆處理引數，並產生確切的記錄數。以下是承載的範例。

```
{
  "request_id" : "23FF1F97-F28A-44AA-AB67-266ED976BF40",
  "cluster" : "arn:aws:redshift:xxxx",
  "user" : "adminuser",
  "database" : "db1",
  "external_function": "public.foo",
  "query_id" : 5678234,
  "num_records" : 4,
  "arguments" : [
    [ 1, 2 ],
    [ 3, null],
    null,
    [ 4, 6]
  ]
}
```

Lambda 函數的傳回輸出包含下列欄位。

欄位名稱	描述	值範圍
success	函數成功或失敗的指示。	值為 "true" 或 "false"。

欄位名稱	描述	值範圍
error_msg	如果成功值為 "false" (如果函數失敗), 則會顯示錯誤訊息; 否則, 會忽略此欄位。	有效的訊息。
num_records	承載中的記錄數。	值為 1 - 2 ⁶⁴ 。
results	以指定格式呼叫的結果。	N/A

以下是 Lambda 函數輸出的範例。

```
{
  "success": true,    // true indicates the call succeeded
  "error_msg" : "my function isn't working", // shall only exist when success != true
  "num_records": 4,   // number of records in this payload
  "results" : [
    1,
    4,
    null,
    7
  ]
}
```

當您從SQL查詢呼叫 Lambda 函數時, Amazon Redshift 會根據下列考量確保連線的安全性:

- GRANT 和REVOKE許可。如需UDF安全和許可的詳細資訊, 請參閱 [UDF 安全性和許可](#)。
- Amazon Redshift 只會向指定的 Lambda 函數提交最小資料集。
- Amazon Redshift 只會呼叫具有指定IAM角色的指定 Lambda 函數。

使用者定義函數 (UDFs) 的使用案例範例

您可以將 Amazon Redshift 與其他元件整合, 以使用使用者定義的函數來解決業務問題。以下是其他人如何使用UDFs其使用案例的一些範例:

- [使用 Amazon Redshift Lambda 存取外部元件 UDFs](#) – 說明 Amazon Redshift Lambda 如何UDFs運作並逐步完成建立 LambdaUDF。
- [使用 Amazon Redshift、Amazon Translate 和 Amazon Comprehend 的 SQL 函數來翻譯和分析文字](#) – 提供預先建置的 Amazon Redshift LambdaUDFs，只需按幾下滑鼠即可安裝文字欄位，以進行翻譯、編輯和分析。
- [從 Amazon Redshift 存取 Amazon Location Service](#) – 說明如何使用 Amazon Redshift Lambda 與 Amazon Location Service UDFs整合。
- [使用 Amazon Redshift 和 Protegrity 進行資料權杖化](#) – 說明如何將 Amazon Redshift Lambda UDFs 與 Protegrity Serverless 產品整合。
- [Amazon Redshift UDFs](#) – Amazon Redshift SQL、Lambda 和 Python 的集合UDFs。

在 Amazon Redshift 中建立預存程序

本主題說明如何在 Amazon Redshift 中建立和使用預存程序。預存程序是多個程式可以使用的SQL陳述式集合。

您可以使用 Postgre 程SQL序語言 PL/Pg SQL 來定義 Amazon Redshift 存儲程序，以執行一組SQL查詢和邏輯操作。程序儲存於資料庫，可供具有足夠資料庫權限的任何使用者執行。

與使用者定義函數 (UDF) 不同，預存程序除了SELECT查詢之外，還可以併入資料定義語言 (DDL/DML) 和資料操作語言 ()。預存程序不需要傳回任何值。您可以使用程序性語言 (包括迴圈和條件式表達式) 來控制邏輯流量。

如需建立和管理預存程序之命SQL令的詳細資訊，請參閱下列命令主題：

- [CREATE PROCEDURE](#)
- [ALTER PROCEDURE](#)
- [DROP PROCEDURE](#)
- [SHOW PROCEDURE](#)
- [CALL](#)
- [GRANT](#)
- [REVOKE](#)
- [ALTER DEFAULT PRIVILEGES](#)

主題

- [Amazon Redshift 中的預存程序概觀](#)
- [PL/pgSQL 語言參考](#)

Amazon Redshift 中的預存程序概觀

本主題說明預存程序的用途和使用方式的詳細資訊。

預存程序通常用於封裝資料轉換、資料驗證的邏輯和特定商業邏輯。透過將多個SQL步驟合併到預存程序中，您可以減少應用程式和資料庫之間的往返次數。

如需精細的存取控制，您可以建立預存程序來執行功能，而不需要讓使用者存取基礎資料表。例如，只有擁有者或超級使用者才能截斷資料表，使用者需要寫入權限，才能將資料插入資料表。您可以建立預

存程序來執行任務，而不是授予使用者對基礎資料表的權限。然後，您會給予使用者權限來執行預存程序。

具有DEFINER安全性屬性的預存程序會以預存程序擁有者的權限執行。根據預設，預存程序具有INVOKER安全性，這表示程序會使用呼叫程序之使用者的權限。

若要建立預存程序，請使用 [CREATE PROCEDURE](#) 命令。若要執行程序，請使用 [CALL](#) 命令。本節稍後提供範例。

Note

在建立 Amazon Redshift 預存程序時，有些用戶端可能會顯示下列錯誤。

```
ERROR: 42601: [Amazon](500310) unterminated dollar-quoted string at or near "$$
```

發生此錯誤是因為用戶端無法正確剖析使用分號分隔陳述式和美元符號 (\$) 引用的陳述式。CREATE PROCEDURE這會導致只有一部分陳述式傳送至 Amazon Redshift 伺服器。您通常可以透過用戶端的 Run as batch 或 Execute selected 選項來解決此錯誤。舉例而言，使用 Aginity 用戶端時，您可以利用 Run entire script as batch 選項。當您使用 SQL「工作台/J」時，我們建議使用版本 124。當您使用 SQL 工作台階 /J 版本 125 時，請考慮指定替代分隔符號作為因應措施。CREATEPROCEDURE包含以分號 (;) 分隔的 SQL 陳述式。定義替代分隔符號 (例如斜線 (/) 並將其放在陳述式的結尾，會將CREATEPROCEDURE陳述式傳送至 Amazon Redshift 伺服器進行處理。以下是範例。

```
CREATE OR REPLACE PROCEDURE test()  
AS $$  
BEGIN  
    SELECT 1 a;  
END;  
$$  
LANGUAGE plpgsql  
;  
/
```

如需詳細資訊，請參閱 SQL WorkBench/J 文件中的 [替代分隔符號](#)。或者使用具有更好支援剖析CREATEPROCEDURE陳述式的用戶端，例如 [Amazon Redshift 主控台](#) 中的 [查詢編輯器](#) 或 TablePlus。

主題

- [命名預存程序](#)
- [預存程序的安全和權限](#)
- [從存儲過程返回結果集](#)
- [管理交易](#)
- [捕捉錯誤](#)
- [記錄預存程序日誌](#)
- [預存程序限制](#)

下列範例顯示不具有輸出引數的程序。根據預設，引數是輸入 (IN) 引數。

```
CREATE OR REPLACE PROCEDURE test_sp1(f1 int, f2 varchar)
AS $$
BEGIN
    RAISE INFO 'f1 = %, f2 = %', f1, f2;
END;
$$ LANGUAGE plpgsql;

call test_sp1(5, 'abc');
INFO: f1 = 5, f2 = abc
CALL
```

Note

當您撰寫預存程序時，我們建議您採用保護敏感值的最佳作法：

不要在預存程序邏輯中對任何敏感資訊進行硬式編碼。例如，請勿在預存程序主體的 CREATEUSER 陳述式中指派使用者密碼。這會造成安全性風險，因為硬式編碼值可以記錄為目錄資料表中的結構描述資料。應改為透過參數將敏感值 (例如密碼) 當做引數傳遞給預存程序。

如需預存程序的相關資訊，請參閱 [CREATE PROCEDURE](#) 和 [在 Amazon Redshift 中建立預存程序](#)。如需目錄資料表的相關資訊，請參閱 [系統目錄資料表](#)。

下列範例顯示具有輸出引數的程序。引數是輸入 (IN)，輸入和輸出 (INOUT) 和輸出 (OUT)。

```
CREATE OR REPLACE PROCEDURE test_sp2(f1 IN int, f2 INOUT varchar(256), out_var OUT
varchar(256))
```

```

AS $$
DECLARE
    loop_var int;
BEGIN
    IF f1 is null OR f2 is null THEN
        RAISE EXCEPTION 'input cannot be null';
    END IF;
    DROP TABLE if exists my_etl;
    CREATE TEMP TABLE my_etl(a int, b varchar);
    FOR loop_var IN 1..f1 LOOP
        insert into my_etl values (loop_var, f2);
        f2 := f2 || '+' || f2;
    END LOOP;
    SELECT INTO out_var count(*) from my_etl;
END;
$$ LANGUAGE plpgsql;

```

```
call test_sp2(2,'2019');
```

```

          f2          | column2
-----+-----
2019+2019+2019+2019 | 2
(1 row)

```

命名預存程序

本主題說明有關預存程序名稱的詳細資料。

如果您定義的程序有相同名稱和不同的引數資料類型 (或簽章)，則會建立新程序。因此，程序名稱會多載。如需詳細資訊，請參閱[多載程序名稱](#)。Amazon Redshift 不會啟用根據輸出引數的程序多載。您不能有兩個程序是具有相同的名稱和輸入引數資料類型，但卻有不同的輸出引數類型。

擁有者或超級使用者可以將預存程序的主體換成具有相同簽章的預存程序。若要變更預存程序的簽章或傳回類型，請捨棄預存程序再重新建立。如需詳細資訊，請參閱 [DROP PROCEDURE](#) 和 [CREATE PROCEDURE](#)。

在實作預存程序之前，您可以考量命名慣例，以避免潛在的衝突和非預期的結果。因為您可以多載程序名稱，它們可能會與現有和未來的 Amazon Redshift 程序名稱相衝突。

多載程序名稱

程序是以其名稱和簽章來識別，而簽章是輸入引數的數目和引數的資料類型。如果相同結構描述中的兩個程序具有不同的簽章，則它們可以具有相同的名稱。換言之，您可以過載程序名稱。

當您執行程序時，查詢引擎會根據您提供的引數數目和引數的資料類型，以決定要呼叫哪個程序。您可以使用多載來模擬具有可變數目引數的程序，直到CREATEPROCEDURE命令允許的限制為止。如需詳細資訊，請參閱[CREATE PROCEDURE](#)。

防止命名衝突

我們建議您在所有程序名稱前加上 sp_。Amazon Redshift 專門為預存程序保留 sp_ 字首。您可以在程序名稱前方加上 sp_，以確保程序名稱不會與任何現有或未來的 Amazon Redshift 程序名稱相衝突。

預存程序的安全和權限

本主題說明建立和執行預存程序所需的資料庫認證。

根據預設，所有使用者具有建立程序的權限。若要建立程序，您必須擁有預設授與語言 PL/PG SQL 的 USAGE 權限。PUBLIC 根據預設，只有超級使用者和擁有者才有權限呼叫程序。如果超級使用者想要防止使用者建立預存程序，則可以從使用者 REVOKE USAGE 在 PL/PG SQL 上執行。

若要呼叫程序，您必須獲得程序的 EXECUTE 權限。依預設，程序擁有者和超級使用者會授與新程序的 EXECUTE 權限。如需詳細資訊，請參閱[GRANT](#)。

根據預設，建立程序的使用者就是擁有者。依預設 CREATEDROP，擁有者對程序具有、和 EXECUTE 權限。超級使用者具備所有權限。

SECURITY 屬性控制程序存取資料庫物件的權限。建立預存程序時，您可以將 SECURITY 屬性設定為 DEFINER 或 INVOKER。如果您指定 SECURITY INVOKER，程序會使用叫用程序之使用者的權限。如果您指定 SECURITY DEFINER，程序會使用程序擁有者的權限。INVOKER 為預設值。

由於 SECURITY DEFINER 程序是以擁有該程序之使用者的權限執行，因此您必須確定程序不會遭到濫用。為了確保 SECURITY DEFINER 程序不會被濫用，請執行以下操作：

- 授予 EXECUTE 權限給 SECURITY DEFINER 程序給特定使用者，而不是授予 PUBLIC。
- 以結構描述名稱來限定程序需要存取的所有資料庫物件。例如，使用 myschema.mytable，而不只是 mytable。
- 如果您無法依結構描述限定物件名稱，請在建立程序 search_path 時使用 SET 選項進行設定。設定 search_path 來排除不受信任使用者所能撰寫的任何結構描述。此方法防止此程序的任何發起人

建立物件 (例如, 資料表或檢視), 而遮罩應該由程序使用的物件。如需SET選項的更多資訊, 請參閱[CREATE PROCEDURE](#)。

下列範例將 `search_path` 設為 `admin`, 以確保從 `admin` 結構描述存取 `user_creds` 資料表, 而不是從 `public` 或發起人的 `search_path` 中任何其他結構描述來存取。

```
CREATE OR REPLACE PROCEDURE sp_get_credentials(userid int, o_creds OUT varchar)
AS $$
BEGIN
    SELECT creds INTO o_creds
    FROM user_creds
    WHERE user_id = $1;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER
-- Set a secure search_path
SET search_path = admin;
```

從存儲過程返回結果集

本主題說明預存程序傳回資料的方式。

您可以使用游標或暫存資料表來傳回結果集。

傳回游標

若要傳回游標, 請使用`refcursor`資料類型定義的INOUT引數建立程序。當您呼叫程序時, 請為游標命名。然後, 您可以依照名稱從游標中獲取結果。

下列範例會建立以使`get_result_set`用`refcursor`資料類型命名的INOUT引`rs_out`數命名的程序。此程序會使用SELECT陳述式開啟游標。

```
CREATE OR REPLACE PROCEDURE get_result_set (param IN integer, rs_out INOUT refcursor)
AS $$
BEGIN
    OPEN rs_out FOR SELECT * FROM fact_tbl where id >= param;
END;
$$ LANGUAGE plpgsql;
```

以下CALL命令打開帶有名稱的光標`mycursor`。請只在交易內使用游標。


```
BEGIN;  
CALL get_result_set(1, 'mycursor');
```

開啟游標後，您就可以從游標擷取，如下列範例所示。

```
FETCH ALL FROM mycursor;
```

id	secondary_id	name
1	1	Joe
1	2	Ed
2	1	Mary
1	3	Mike

(4 rows)

最後，交易會遞交或轉返。

```
COMMIT;
```

預存程序傳回的游標受到相同的限制和效能考量，如中所述 `DECLARECURSOR`。如需詳細資訊，請參閱 [游標限制條件](#)。

下列範例會示範使用來自的 `refcursor` 資料類型呼叫 `get_result_set` 預存程序 JDBC。常值 `'mycursor'` (游標名稱) 會傳給 `prepareStatement`。然後，從 `ResultSet` 擷取結果。

```
static void refcursor_example(Connection conn) throws SQLException {  
    conn.setAutoCommit(false);  
    PreparedStatement proc = conn.prepareStatement("CALL get_result_set(1,  
'mycursor')");  
    proc.execute();  
    ResultSet rs = statement.executeQuery("fetch all from mycursor");  
    while (rs.next()) {  
        int n = rs.getInt(1);  
        System.out.println("n " + n);  
    }  
}
```

使用暫存資料表

若要傳回結果，您可以傳回含有結果列之暫存資料表的控制代碼。用戶端可以將名稱當作參數提供給預存程序。在存儲過程中，SQL 可以使用 `dynamic` 來在臨時表上進行操作。下列顯示一個範例。

```

CREATE PROCEDURE get_result_set(param IN integer, tmp_name INOUT varchar(256)) as $$
DECLARE
    row record;
BEGIN
    EXECUTE 'drop table if exists ' || tmp_name;
    EXECUTE 'create temp table ' || tmp_name || ' as select * from fact_tbl where id <= '
    || param;
END;
$$ LANGUAGE plpgsql;

CALL get_result_set(2, 'myresult');
tmp_name
-----
myresult
(1 row)

SELECT * from myresult;
 id | secondary_id | name
----+-----+-----
  1 |             1 | Joe
  2 |             1 | Mary
  1 |             2 | Ed
  1 |             3 | Mike
(4 rows)

```

管理交易

您可以建立具有預設交易管理行為或非原子行為的預存程序。

預設模式預存程序交易管理

預設的交易模式自動認可行為會導致個別執行的每個SQL命令單獨認可。對預存程序的呼叫會被視為單一SQL命令。程序內的SQL陳述式的行為就好像它們位於交易區塊中，該交易區塊會在呼叫開始和結束時在呼叫結束時隱含地開始。對另一個程序的巢狀呼叫會與其他任何SQL陳述式一樣處理，並在與呼叫者相同交易的內容中運作。如需自動遞交行為的相關資訊，請參閱[可序列化隔離](#)。

但是，假設您從使用者指定的交易區塊內呼叫預存程序 (由BEGIN... 定義) COMMIT)。在此情況下，預存程序中的所有陳述式都會在使用者指定交易的內容中執行。程序不會隱含地在結束時遞交。發起人控制程序遞交或轉返。

如果執行預存程序時發生任何錯誤，您在目前交易中做的所有變更都會復原。

您可以使用下列交易來控制預存程序中的陳述式：

- COMMIT— 提交在當前事務中完成的所有工作，並隱式開始一個新的事務。如需詳細資訊，請參閱[COMMIT](#)。
- ROLLBACK— 回復目前交易中完成的工作，並隱含地開始新的交易。如需詳細資訊，請參閱[ROLLBACK](#)。

TRUNCATE是您可以從預存程序中發出並影響交易管理的另一個陳述式。在 Amazon Redshift 中，隱式TRUNCATE發出提交。在預存程序的範圍內，此行為不變。當TRUNCATE陳述式從預存程序中發出時，它會認可目前的交易，並開始新的交易。如需詳細資訊，請參閱[TRUNCATE](#)。

在COMMIT、ROLLBACK或TRUNCATE陳述式之後的所有陳述式都會在新交易的內容中執行。它們會這樣做COMMIT，直到遇到ROLLBACK、或TRUNCATE陳述式或預存程序結束為止。

當您在預存程序中使用COMMITROLLBACK、或TRUNCATE陳述式時，會套用下列條件約束：

- 如果從交易區塊內呼叫預存程序，就無法發出COMMITROLLBACK、或TRUNCATE陳述式。這項限制適用於預存程序本身和任何巢狀程序呼叫內部。
- 如果預存程序是使用SET config選項建立的，就無法發出COMMITROLLBACK、或TRUNCATE陳述式。這項限制適用於預存程序本身和任何巢狀程序呼叫內部。
- 處理COMMIT、ROLLBACK或TRUNCATE陳述式時，任何開啟 (明確或隱含) 的游標都會自動關閉。如需了解明確和隱含游標的限制，請參閱[預存程序限制](#)。

此外，您無法執行COMMIT或ROLLBACK使用動態SQL。但是，您可以TRUNCATE使用動態運行SQL。如需詳細資訊，請參閱[動態 SQL](#)。

使用預存程序時，請考慮 PL/Pg SQL 中的 BEGIN AND END 陳述式僅用於分組。而無法用來啟動或結束交易。如需詳細資訊，請參閱[區塊](#)。

下列範例示範從明確交易區塊內呼叫預存程序時的交易行為。從預存程序外發出的兩個 insert 陳述式和從它之內發出的一個 insert 陳述式，都是相同交易 (3382) 的一部分。當使用者發出明確遞交時，交易就會遞交。

```
CREATE OR REPLACE PROCEDURE sp_insert_table_a(a int) LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO test_table_a values (a);
END;
```

```

$$;

Begin;
  insert into test_table_a values (1);
  Call sp_insert_table_a(2);
  insert into test_table_a values (3);
Commit;

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;

```

userid	xid	pid	type	stmt_text
103	3382	599	UTILITY	Begin;
103	3382	599	QUERY	insert into test_table_a values (1);
103	3382	599	UTILITY	Call sp_insert_table_a(2);
103	3382	599	QUERY	INSERT INTO test_table_a values (\$1)
103	3382	599	QUERY	insert into test_table_a values (3);
103	3382	599	UTILITY	COMMIT

相對的，假設有同樣的陳述式從明確交易區塊外發出，而且該工作階段的自動遞交設定為 ON。在這種情況下，每個陳述式都會在自己的交易內運作。

```

insert into test_table_a values (1);
Call sp_insert_table_a(2);
insert into test_table_a values (3);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;

```

userid	xid	pid	type	stmt_text
103	3388	599	QUERY	insert into test_table_a values (1);
103	3388	599	UTILITY	COMMIT
103	3389	599	UTILITY	Call sp_insert_table_a(2);
103	3389	599	QUERY	INSERT INTO test_table_a values (\$1)
103	3389	599	UTILITY	COMMIT
103	3390	599	QUERY	insert into test_table_a values (3);
103	3390	599	UTILITY	COMMIT

下列範例會在插入到之後發出TRUNCATE陳述式test_table_a。該TRUNCATE語句發出一個隱式提交，提交當前事務 (3335) 並啟動一個新事務 (3336)。新的交易會在程序結束時遞交。

```
CREATE OR REPLACE PROCEDURE sp_truncate_proc(a int, b int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table_a values (a);
  TRUNCATE test_table_b;
  INSERT INTO test_table_b values (b);
END;
$$;
```

```
Call sp_truncate_proc(1,2);
```

```
select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;
```

userid	xid	pid	type	stmt_text
103	3335	23636	UTILITY	Call sp_truncate_proc(1,2);
103	3335	23636	QUERY	INSERT INTO test_table_a values (\$1)
103	3335	23636	UTILITY	TRUNCATE test_table_b
103	3335	23636	UTILITY	COMMIT
103	3336	23636	QUERY	INSERT INTO test_table_b values (\$1)
103	3336	23636	UTILITY	COMMIT

下列範例會TRUNCATE從巢狀呼叫發出 a。TRUNCATE提交到目前為止在事務中的外部和內部過程中完成的所有工作 (3344)。它會啟動新的交易 (3345)。新的交易會在外層程序結束時遞交。

```
CREATE OR REPLACE PROCEDURE sp_inner(c int, d int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO inner_table values (c);
  TRUNCATE outer_table;
  INSERT INTO inner_table values (d);
END;
$$;
```

```
CREATE OR REPLACE PROCEDURE sp_outer(a int, b int, c int, d int) LANGUAGE plpgsql
AS $$
```

```

BEGIN
  INSERT INTO outer_table values (a);
  Call sp_inner(c, d);
  INSERT INTO outer_table values (b);
END;
$$;

Call sp_outer(1, 2, 3, 4);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;

userid | xid  | pid  | type  |
-----+-----+-----+-----+
                                stmt_text
-----+-----+-----+-----+
103 | 3344 | 23636 | UTILITY | Call sp_outer(1, 2, 3, 4);
103 | 3344 | 23636 | QUERY   | INSERT INTO outer_table values ( $1 )
103 | 3344 | 23636 | UTILITY | CALL sp_inner( $1 , $2 )
103 | 3344 | 23636 | QUERY   | INSERT INTO inner_table values ( $1 )
103 | 3344 | 23636 | UTILITY | TRUNCATE outer_table
103 | 3344 | 23636 | UTILITY | COMMIT
103 | 3345 | 23636 | QUERY   | INSERT INTO inner_table values ( $1 )
103 | 3345 | 23636 | QUERY   | INSERT INTO outer_table values ( $1 )
103 | 3345 | 23636 | UTILITY | COMMIT

```

下列範例顯示在提交TRUNCATE陳述式時，游標cur1已關閉。

```

CREATE OR REPLACE PROCEDURE sp_open_cursor_truncate()
LANGUAGE plpgsql
AS $$
DECLARE
  rec RECORD;
  cur1 cursor for select * from test_table_a order by 1;
BEGIN
  open cur1;
  TRUNCATE table test_table_b;
  Loop
    fetch cur1 into rec;
    raise info '%', rec.c1;
    exit when not found;
  End Loop;

```

```

END
$$;

call sp_open_cursor_truncate();
ERROR: cursor "cur1" does not exist
CONTEXT: PL/pgSQL function "sp_open_cursor_truncate" line 8 at fetch

```

下列範例會發出TRUNCATE陳述式，而且無法從明確交易區塊中呼叫。

```

CREATE OR REPLACE PROCEDURE sp_truncate_atomic() LANGUAGE plpgsql
AS $$
BEGIN
    TRUNCATE test_table_b;
END;
$$;

Begin;
    Call sp_truncate_atomic();
ERROR: TRUNCATE cannot be invoked from a procedure that is executing in an atomic
context.
HINT: Try calling the procedure as a top-level call i.e. not from within an explicit
transaction block.
Or, if this procedure (or one of its ancestors in the call chain) was created with SET
config options, recreate the procedure without them.
CONTEXT: SQL statement "TRUNCATE test_table_b"
PL/pgSQL function "sp_truncate_atomic" line 2 at SQL statement

```

下列範例顯示不是超級使用者或資料表擁有者的使用者，可以在資料表上發出TRUNCATE陳述式。使用者會使用 Security Definer 預存程序來執行此動作。該範例會出現下列操作：

- user1 建立表格 test_tbl。
- user1 建立預存程序 sp_truncate_test_tbl。
- user1 授予預存程序的 EXECUTE 權限給 user2。
- user2 執行預存程序以截斷表格 test_tbl。此範例顯示 TRUNCATE 命令前後的資料列計數。

```

set session_authorization to user1;
create table test_tbl(id int, name varchar(20));
insert into test_tbl values (1,'john'), (2, 'mary');
CREATE OR REPLACE PROCEDURE sp_truncate_test_tbl() LANGUAGE plpgsql
AS $$

```

```

DECLARE
  tbl_rows int;
BEGIN
  select count(*) into tbl_rows from test_tbl;
  RAISE INFO 'RowCount before Truncate: %', tbl_rows;
  TRUNCATE test_tbl;
  select count(*) into tbl_rows from test_tbl;
  RAISE INFO 'RowCount after Truncate: %', tbl_rows;
END;
$$ SECURITY DEFINER;
grant execute on procedure sp_truncate_test_tbl() to user2;
reset session_authorization;

set session_authorization to user2;
call sp_truncate_test_tbl();
INFO: RowCount before Truncate: 2
INFO: RowCount after Truncate: 0
CALL
reset session_authorization;

```

以下示例發出COMMIT兩次。第一次COMMIT提交在事務 10363 中完成的所有工作，並隱式啟動事務 10364。交易 10364 是由第二個COMMIT語句承諾。

```

CREATE OR REPLACE PROCEDURE sp_commit(a int, b int) LANGUAGE plpgsql
AS $$
BEGIN
  INSERT INTO test_table values (a);
  COMMIT;
  INSERT INTO test_table values (b);
  COMMIT;
END;
$$;

call sp_commit(1,2);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;
userid |  xid  | pid  | type  |
          stmt_text
-----+-----+-----+-----
+-----+-----+-----+-----

```



```

100 | 10363 | 3089 | UTILITY | call sp_commit(1,2);
100 | 10363 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
100 | 10363 | 3089 | UTILITY | COMMIT
100 | 10364 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
100 | 10364 | 3089 | UTILITY | COMMIT

```

如sum_vals果大於 2，下列範例會發出ROLLBACK陳述式。第一條ROLLBACK語句回滾在事務 10377 完成的所有工作，並開始一個新的事務 10378。10378 交易會在程序結束時遞交。

```

CREATE OR REPLACE PROCEDURE sp_rollback(a int, b int) LANGUAGE plpgsql
AS $$
DECLARE
    sum_vals int;
BEGIN
    INSERT INTO test_table values (a);
    SELECT sum(c1) into sum_vals from test_table;
    IF sum_vals > 2 THEN
        ROLLBACK;
    END IF;

    INSERT INTO test_table values (b);
END;
$$;

call sp_rollback(1, 2);

select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;

userid | xid | pid | type |
          stmt_text
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
100 | 10377 | 3089 | UTILITY | call sp_rollback(1, 2);
100 | 10377 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
100 | 10377 | 3089 | QUERY   | SELECT sum(c1) from test_table
100 | 10377 | 3089 | QUERY   | Undoing 1 transactions on table 133646 with current
xid 10377 : 10377
100 | 10378 | 3089 | QUERY   | INSERT INTO test_table values ( $1 )
100 | 10378 | 3089 | UTILITY | COMMIT

```

非原子模式預存程序交易管理

在NONATOMIC模式中建立的預存程序與以預設模式建立的程序具有不同的交易控制行為。與預存程序之外SQL命令的自動認可行為類似，程序內的每個SQL陳述式都會在自己的交易中執行，並自動認可。NONATOMIC如果使用者在NONATOMIC預存程序中開始明確交易區塊，則區塊內的SQL陳述式不會自動認可。交易區塊控制項遞交或復原其中的陳述式。

在NONATOMIC預存程序中，您可以使用STARTTRANSACTION陳述式在程序內開啟明確的交易區塊。不過，如果已經有開啟的交易區塊，這個陳述式就不會執行任何動作，因為 Amazon Redshift 不支援子交易。上一筆交易會繼續進行。

當您在NONATOMIC程序內使用游標FOR迴圈時，請確定在迭代查詢結果之前開啟明確的交易區塊。否則，當循環內的SQL語句自動提交時，游標會關閉。

使用NONATOMIC模式行為時的一些注意事項如下：

- 如果沒有打開的事務塊，存儲過程中的每個SQL語句都會自動提交，並且會話自動提交設置為 ON。
- 如果從交易區塊內呼叫預存程序，您可以發出 COMMITROLLBACK/TRUNCATE陳述式來結束交易。這在預設模式下是不可能的。
- 您可以發出STARTTRANSACTION陳述式，在預存程序中開始交易區塊。

下列範例會示範使用NONATOMIC預存程序時的交易行為。下列所有範例的工作階段都將自動遞交設定為 ON。

在下列範例中，NONATOMIC預存程序有兩個INSERT陳述式。在交易區塊外呼叫程序時，程序中的每個INSERT陳述式都會自動認可。

```
CREATE TABLE test_table_a(v int);
CREATE TABLE test_table_b(v int);

CREATE OR REPLACE PROCEDURE sp_nonatomic_insert_table_a(a int, b int) NONATOMIC AS
$$
BEGIN
    INSERT INTO test_table_a values (a);
    INSERT INTO test_table_b values (b);
END;
$$
LANGUAGE plpgsql;

Call sp_nonatomic_insert_table_a(1,2);
```

```
Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;
```

userid	xid	pid	type	stmt_text
1	1792	1073807554	UTILITY	Call sp_nonatomic_insert_table_a(1,2);
1	1792	1073807554	QUERY	INSERT INTO test_table_a values (\$1)
1	1792	1073807554	UTILITY	COMMIT
1	1793	1073807554	QUERY	INSERT INTO test_table_b values (\$1)
1	1793	1073807554	UTILITY	COMMIT

(5 rows)

但是，當從一個BEGIN.. COMMIT塊，所有的語句都是同一個事務的一部分 (xid = 1799)。

```
Begin;
  INSERT INTO test_table_a values (10);
  Call sp_nonatomic_insert_table_a(20,30);
  INSERT INTO test_table_b values (40);
Commit;
```

```
Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;
```

userid	xid	pid	type	stmt_text
1	1799	1073914035	UTILITY	Begin;
1	1799	1073914035	QUERY	INSERT INTO test_table_a values (10);
1	1799	1073914035	UTILITY	Call sp_nonatomic_insert_table_a(20,30);
1	1799	1073914035	QUERY	INSERT INTO test_table_a values (\$1)
1	1799	1073914035	QUERY	INSERT INTO test_table_b values (\$1)
1	1799	1073914035	QUERY	INSERT INTO test_table_b values (40);
1	1799	1073914035	UTILITY	COMMIT

(7 rows)

在此範例中，兩個INSERT陳述式介於 STARTTRANSACTION... COMMIT。當程序在交易區塊之外呼叫時，這兩個INSERT陳述式位於相同的交易中 (xid=1866)。

```
CREATE OR REPLACE PROCEDURE sp_nonatomic_txn_block(a int, b int) NONATOMIC AS
$$
BEGIN
```

```

START TRANSACTION;
INSERT INTO test_table_a values (a);
INSERT INTO test_table_b values (b);
COMMIT;
END;
$$
LANGUAGE plpgsql;

Call sp_nonatomic_txn_block(1,2);

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;

```

userid	xid	pid	type	stmt_text
1	1865	1073823998	UTILITY	Call sp_nonatomic_txn_block(1,2);
1	1866	1073823998	QUERY	INSERT INTO test_table_a values (\$1)
1	1866	1073823998	QUERY	INSERT INTO test_table_b values (\$1)
1	1866	1073823998	UTILITY	COMMIT

(4 rows)

當從BEGIN... 內調用該過程時 COMMIT塊，程序STARTTRANSACTION內部什麼都不做，因為已經有一個開放的事務。程序COMMIT內部提交當前事務 (xid = 1876) 並啟動一個新事務。

```

Begin;
INSERT INTO test_table_a values (10);
Call sp_nonatomic_txn_block(20,30);
INSERT INTO test_table_b values (40);
Commit;

Select userid, xid, pid, type, trim(text) as stmt_text
from svl_statementtext where pid = pg_backend_pid() order by xid , starttime ,
sequence;

```

userid	xid	pid	type	stmt_text
1	1876	1073832133	UTILITY	Begin;
1	1876	1073832133	QUERY	INSERT INTO test_table_a values (10);
1	1876	1073832133	UTILITY	Call sp_nonatomic_txn_block(20,30);
1	1876	1073832133	QUERY	INSERT INTO test_table_a values (\$1)
1	1876	1073832133	QUERY	INSERT INTO test_table_b values (\$1)
1	1876	1073832133	UTILITY	COMMIT

```

1 | 1878 | 1073832133 | QUERY | INSERT INTO test_table_b values (40);
1 | 1878 | 1073832133 | UTILITY | COMMIT
(8 rows)

```

此範例示範如何使用游標迴圈。test_table_a 資料表具有三個值。我們的目標是迭代這三個值，並將它們插入到資料表 test_table_b。如果存NONATOMIC儲過程以下面的方式創建，它會拋出錯誤光標 "cur1" 在第一個循環中執行INSERT語句之後不存在。這是因為 auto 提交INSERT關閉了打開的光標。

```

insert into test_table_a values (1), (2), (3);

CREATE OR REPLACE PROCEDURE sp_nonatomic_cursor() NONATOMIC
LANGUAGE plpgsql
AS $$
DECLARE
    rec RECORD;
    cur1 cursor for select * from test_table_a order by 1;
BEGIN
    open cur1;
    Loop
        fetch cur1 into rec;
        exit when not found;
        raise info '%', rec.v;
        insert into test_table_b values (rec.v);
    End Loop;
END
$$;

CALL sp_nonatomic_cursor();

INFO: 1
ERROR: cursor "cur1" does not exist
CONTEXT: PL/pgSQL function "sp_nonatomic_cursor" line 7 at fetch

```

為了使光標循環工作，請將其放在 STARTTRANSACTION... COMMIT。

```

insert into test_table_a values (1), (2), (3);

CREATE OR REPLACE PROCEDURE sp_nonatomic_cursor() NONATOMIC
LANGUAGE plpgsql
AS $$
DECLARE
    rec RECORD;

```

```
    cur1 cursor for select * from test_table_a order by 1;
BEGIN
  START TRANSACTION;
  open cur1;
  Loop
    fetch cur1 into rec;
    exit when not found;
    raise info '%', rec.v;
    insert into test_table_b values (rec.v);
  End Loop;
  COMMIT;
END
$$;

CALL sp_nonatomic_cursor();

INFO:  1
INFO:  2
INFO:  3
CALL
```

捕捉錯誤

本主題說明 Amazon Redshift 如何處理錯誤。

當預存程序中的查詢或命令造成錯誤時，後續查詢不會執行，而且會復原交易。但是您可以使用 EXCEPTION 塊處理錯誤。

Note

預設行為是，即使預存程序中沒有其他產生錯誤的條件，錯誤也會導致後續查詢無法執行。

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  WHEN OTHERS THEN
    statements
END;
```

當發生例外狀況時，您新增例外狀況處理區塊時，您可以撰寫RAISE陳述式和大多數其他 PL/ SQL PG 陳述式。例如，您可以使用自訂訊息引發例外狀況，或將記錄插入記錄資料表。

進入例外狀況處理區塊時，會復原目前的交易，並建立新的交易以執行區塊中的陳述式。如果區塊中的陳述式沒有錯誤地執行，就會遞交交易並重新擲回例外狀況。最後，預存程序會結束。

例外區塊中唯一支援的條件是OTHERS，符合除查詢取消以外的每個錯誤類型。此外，如果例外狀況處理塊中發生錯誤，則外部例外狀況處理塊可以捕獲該錯誤。

當NONATOMIC程序內部發生錯誤時，如果錯誤是由例外狀況區塊處理，則不會重新擲回錯誤。請參閱 PL/PG SQL 語句拋RAISE出異常處理塊捕獲的異常。此陳述式僅在例外狀況處理區塊中有效。如需詳細資訊，請參閱 [RAISE](#)。

使用CONTINUE處理常式控制預存程序中發生錯誤之後的情況

CONTINUE處理常式是一種例外狀況處理常式，可控制NONATOMIC預存程序內的執行流程。透過使用的過程，您可以揪出問題點並處理例外狀況，而不需結束現有陳述式區塊。通常當預存程序中發生錯誤時，流程會中斷，並將錯誤傳回給呼叫者。但是，在某些使用案例中，錯誤情況還沒有嚴重到足以中斷流程。您可能想要按既定程序處理錯誤，使用您在不同的交易中選擇的錯誤處理邏輯，然後繼續執行錯誤後續的陳述式。語法如下列所示。

```
[ DECLARE
  declarations ]
BEGIN
  statements
EXCEPTION
  [ CONTINUE_HANDLER | EXIT_HANDLER ] WHEN OTHERS THEN
    handler_statements
END;
```

有數個系統資料表可協助您收集有關各種錯誤類型的資訊。如需詳細資訊，請參閱[STL_LOAD_ERRORS](#)、[STL_ERROR](#)及[SYS_STREAM_SCAN_ERRORS](#)。您也可以使用其他系統資料表來疑難排解錯誤。有關這些的更多資訊，請參閱[系統資料表和檢視參考](#)。

範例

下列範例說明如何在例外狀況處理區塊中撰寫陳述式。預存程序正在使用預設的交易管理行為。

```
CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);
```

```

CREATE OR REPLACE PROCEDURE update_employee_sp() AS
$$
BEGIN
    UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
    EXECUTE 'select invalid';
EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'An exception occurred.';
    INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp();

INFO:  An exception occurred.
ERROR:  column "invalid" does not exist
CONTEXT:  SQL statement "select invalid"
PL/pgSQL function "update_employee_sp" line 3 at execute statement

```

在這個範例中，如果我們呼叫 `update_employee_sp`，會引發資訊訊息「發生例外狀況。」並將錯誤訊息插入記錄資料表的 `employee_error_log` 日誌中。在預存程序結束之前，會再次擲回原始例外狀況。下列查詢顯示執行範例所產生的記錄。

```

SELECT * from employee;

firstname | lastname
-----+-----
Tomas     | Smith

SELECT * from employee_error_log;

          message
-----
Error message: column "invalid" does not exist

```

如需有關的更多資訊RAISE，包括格式化說明和其他圖層清單，請參閱[支援的 PL/pgSQL 陳述式](#)。

下列範例說明如何在例外狀況處理區塊中撰寫陳述式。預存程序正在使用NONATOMIC交易管理行為。在此範例中，程序呼叫完成後，不會擲回呼叫者的錯誤。UPDATE陳述式不會因為下一個陳述式中的錯誤而復原。會引發資訊訊息，並在記錄資料表中插入錯誤訊息。

```

CREATE TABLE employee (firstname varchar, lastname varchar);

```



```

INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

-- Create the SP in NONATOMIC mode
CREATE OR REPLACE PROCEDURE update_employee_sp_2() NONATOMIC AS
$$
BEGIN
    UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
    EXECUTE 'select invalid';
EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'An exception occurred.';
    INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp_2();
INFO: An exception occurred.
CALL

SELECT * from employee;

  firstname | lastname
-----+-----
   Adam    | Smith
(1 row)

SELECT * from employee_error_log;

                message
-----
Error message: column "invalid" does not exist
(1 row)

```

此範例示範如何使用兩個子區塊建立程序。呼叫預存程序時，第一個子區塊的錯誤會由其例外狀況處理區塊處理。在第一個子區塊完成之後，程序會繼續執行第二個子區塊。從結果可以看到，程序呼叫完成時沒有擲回任何錯誤。在表員工UPDATE和INSERT操作承諾。來自兩個例外狀況區塊的錯誤訊息都會插入記錄資料表中。

```

CREATE TABLE employee (firstname varchar, lastname varchar);
INSERT INTO employee VALUES ('Tomas','Smith');
CREATE TABLE employee_error_log (message varchar);

```

```

CREATE OR REPLACE PROCEDURE update_employee_sp_3() NONATOMIC AS
$$
BEGIN
    BEGIN
        UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
        EXECUTE 'select invalid1';
    EXCEPTION WHEN OTHERS THEN
        RAISE INFO 'An exception occurred in the first block.';
        INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
    END;
    BEGIN
        INSERT INTO employee VALUES ('Edie','Robertson');
        EXECUTE 'select invalid2';
    EXCEPTION WHEN OTHERS THEN
        RAISE INFO 'An exception occurred in the second block.';
        INSERT INTO employee_error_log VALUES ('Error message: ' || SQLERRM);
    END;
END;
$$
LANGUAGE plpgsql;

CALL update_employee_sp_3();
INFO:  An exception occurred in the first block.
INFO:  An exception occurred in the second block.
CALL

SELECT * from employee;

  firstname | lastname
-----+-----
  Adam      | Smith
  Edie      | Robertson
(2 rows)

SELECT * from employee_error_log;

                message
-----+-----
  Error message: column "invalid1" does not exist
  Error message: column "invalid2" does not exist
(2 rows)

```

下面的例子演示了如何使用CONTINUE異常處理程序。此範例會建立兩份資料表，並在預存程序中使用資料表。CONTINUE處理常式會在具有NONATOMIC交易管理行為的預存程序中控制執行流程。

```
CREATE TABLE tbl_1 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_1() NONATOMIC AS
$$
BEGIN
    INSERT INTO tbl_1 VALUES (1);
    -- Expect an error for the insert statement following, because of the invalid value
    INSERT INTO tbl_1 VALUES ("val");
    INSERT INTO tbl_1 VALUES (2);
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

呼叫預存程序：

```
CALL sp_exc_handling_1();
```

流程如下所示：

1. 因為嘗試在資料欄插入不相容的資料類型而發生錯誤。控制傳遞給EXCEPTION塊。當進入例外狀況處理區塊時，會復原目前的交易，並建立新的隱含交易以執行區塊中的陳述式。
2. 如果 CONTINUE _ 中的語句HANDLER運行沒有錯誤，控制權傳遞給聲明立即導致異常後面的語句。（如果 CONTINUE _ 中的語句HANDLER引發了新的異常，則可以使用EXCEPTION塊內的異常處理程序來處理它。）

呼叫範例預存程序之後，資料表包含下列記錄：

- 如果執行 `SELECT * FROM tbl_1;`，它會傳回兩個記錄。這些包含值 1 和 2。
- 如果執行 `SELECT * FROM tbl_error_logging;`，它會傳回一個包含下列值的記錄：發生的錯誤、42703 和 `tbl_1` 中不存在資料欄 "val"。

下列其他錯誤處理範例會同時使用處理EXIT常式和處理常式CONTINUE。此範例建立兩個表：一個資料表和一個記錄表。另外還建立一個展示如何處理錯誤的預存程序：

```
CREATE TABLE tbl_1 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_2() NONATOMIC AS
$$
BEGIN
    INSERT INTO tbl_1 VALUES (1);
    BEGIN
        INSERT INTO tbl_1 VALUES (100);
        -- Expect an error for the insert statement following, because of the invalid
value
        INSERT INTO tbl_1 VALUES ("val");
        INSERT INTO tbl_1 VALUES (101);
    EXCEPTION EXIT_HANDLER WHEN OTHERS THEN
        INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
    END;
    INSERT INTO tbl_1 VALUES (2);
    -- Expect an error for the insert statement following, because of the invalid value
    INSERT INTO tbl_1 VALUES ("val");
    INSERT INTO tbl_1 VALUES (3);
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
    INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

建立預存程序後，使用下列命令呼叫：

```
CALL sp_exc_handling_2();
```

當內部異常塊中發生錯誤時，該異常塊由內部集合括住END，BEGIN並且由處理程序處理。EXIT外部區塊中發生的任何錯誤都會由處理CONTINUE常式處理。

呼叫範例預存程序之後，資料表包含下列記錄：

- 如果執行 `SELECT * FROM tbl_1;`，它會傳回四個記錄，其值為 1、2、3 和 100。
- 如果執行 `SELECT * FROM tbl_error_logging;`，它會傳回兩個記錄。它們具有以下值：發生的錯誤、42703 和 `tbl_1` 中不存在資料欄 "val"。

如果表 `tbl_error_logging` 不存在，則會引發例外狀況。

下面的例子演示了如何使用CONTINUE異常處理程序與FOR循環。此範例會建立三個資料表，並在預存程序內的FOR迴圈中使用這些資料表。FOR循環是結果集變體，這意味著它迭代查詢的結果：

```
CREATE TABLE tbl_1 (a int);
INSERT INTO tbl_1 VALUES (1), (2), (3);
CREATE TABLE tbl_2 (a int);
CREATE TABLE tbl_error_logging(info varchar, err_state varchar, err_msg varchar);

CREATE OR REPLACE PROCEDURE sp_exc_handling_loop() NONATOMIC AS
$$
DECLARE
  rec RECORD;
BEGIN
  FOR rec IN SELECT a FROM tbl_1
  LOOP
    IF rec.a = 2 THEN
      -- Expect an error for the insert statement following, because of the
      invalid value
      INSERT INTO tbl_2 VALUES("val");
    ELSE
      INSERT INTO tbl_2 VALUES (rec.a);
    END IF;
  END LOOP;
EXCEPTION CONTINUE_HANDLER WHEN OTHERS THEN
  INSERT INTO tbl_error_logging VALUES ('Encountered error', SQLSTATE, SQLERRM);
END;
$$ LANGUAGE plpgsql;
```

呼叫預存程序：

```
CALL sp_exc_handling_loop();
```

呼叫範例預存程序之後，資料表包含下列記錄：

- 如果執行 `SELECT * FROM tbl_2;`，它會傳回兩個記錄。這些包含值 1 和 3。
- 如果執行 `SELECT * FROM tbl_error_logging;`，它會傳回一個包含下列值的記錄：發生的錯誤、42703 和 `tbl_2` 中不存在資料欄 "val"。

有關CONTINUE處理程序的使用說明：

- CONTINUE_HANDLER 和 EXIT_HANDLER 關鍵字只能在NONATOMIC存儲過程中使用。

- CONTINUE_HANDLER 和 EXIT_HANDLER 關鍵字是可選的。EXIT_HANDLER 是預設值。

記錄預存程序日誌

本主題說明 Amazon Redshift 用於預存程序記錄的預存程序和檢視。

預存程序的詳細資訊都記錄在下列系統資料表和檢視中：

- SVL_ STORED PROC_CALL — 記錄預存程序呼叫的開始時間和結束時間，以及呼叫是否在完成前結束的詳細資料。如需詳細資訊，請參閱[SVL_STORED_PROC_CALL](#)。
- SVL_ STORED PROC_MESSAGES — RAISE 查詢所發出之預存程序中的訊息會以對應的記錄層級進行記錄。如需詳細資訊，請參閱[SVL_STORED_PROC_MESSAGES](#)。
- SVL_QLOG — 會針對從預存程序呼叫的每個查詢記錄程序呼叫的查詢 ID。如需詳細資訊，請參閱[SVL_QLOG](#)。
- STL_ UTILITYTEXT — 預存程序呼叫完成之後會記錄下來。如需詳細資訊，請參閱[STL_UTILITYTEXT](#)。
- PG PROC_INFO — 此系統目錄視圖顯示有關存儲過程的信息。如需詳細資訊，請參閱[PG_PROC_INFO](#)。

預存程序限制

本主題說明 Amazon Redshift 儲存程序的限制。

使用 Amazon Redshift 預存程序時有下列限制。

Amazon Redshift 和 Postgre 存儲過程支持之間SQL的區別

以下是 Amazon Redshift 和 Post SQL gre 中存儲過程支持之間的差異：

- Amazon Redshift 不支援子交易，因此對例外狀況處理區塊的支援有限。

考量與限制

以下是 Amazon Redshift 中預存程序的考量事項：

- 資料庫的預存程序數目上限為 10,000。
- 程序的原始程式碼大小上限為 2 MB。

- 您在使用者工作階段中可同時開啟的明確和隱含游標數目上限是一個。FOR 循環遍歷一個語句的結果集打開 SQL 隱式游標。不支援巢狀游標。
- 明確和隱含游標的結果集大小限制，與標準 Amazon Redshift 游標相同。如需詳細資訊，請參閱[游標限制條件](#)。
- 巢狀呼叫層數上限為 16。
- 輸入引數的程序參數數目上限為 32，輸出引數的上限也是 32。
- 預存程序中的變數數目上限為 1,024。
- 預存程序內不支援任何需要其交易內容的 SQL 命令。範例包括：
 - PREPARE
 - CREATE/DROP DATABASE
 - CREATE EXTERNAL TABLE
 - VACUUM
 - SET LOCAL
 - ALTER TABLE APPEND
- 資料類型不支援透過 Java refcursor 資料庫連線 (JDBC) 驅動程式呼叫的 registerOutParameter 方法。關於使用 refcursor 資料類型的範例，請參閱[從存儲過程返回結果集](#)。

PL/pgSQL 語言參考

Amazon Redshift 中的預存程序以 PostgreSQL PL/pgSQL 程序性語言為基礎，但有一些重要差異。在此參考中，您可以找到 Amazon Redshift 所實作的 PL/pgSQL 語法的詳細資訊。如需 PL/pgSQL 的相關資訊，請參閱 PostgreSQL 文件中的 [PL/pgSQL - SQL 程序性語言](#)。

主題

- [PL/pgSQL 參考慣例](#)
- [PL/pgSQL 的結構](#)
- [支援的 PL/pgSQL 陳述式](#)

PL/pgSQL 參考慣例

在本節中，您可以找到 PL/pgSQL 預存程序語言的語法編寫慣例。

字元	描述
CAPS (大寫字母)	大寫字詞為關鍵字。
[]	方括號是用來表示選用的引數。方括號中的多個引數，代表您可以選擇任何數目的引數。此外，方括號中不同行的引數，代表 Amazon Redshift 剖析器期望引數的排序，等於在語法中所列出的順序。
{ }	大括號表示您需要選擇大括號內的其中一個引數。
	直立線符號會將您可選擇的引數隔開。
<i>####</i>	紅色斜體的字表示預留位置。插入適當的值來取代紅色斜體字詞。
...	省略符號表示您可以重複前一個元素。
'	單引號中的字詞表示您必須輸入引用內容。

PL/pgSQL 的結構

PL/pgSQL 是一種程序性語言，具有很多建構與其他程序性語言相同。

主題

- [區塊](#)
- [變數宣告](#)
- [別名宣告](#)
- [內建變數](#)
- [記錄類型](#)

區塊

PL/pgSQL 是區塊結構化語言。程序的整個主體都定義在區塊內，其中包含變數宣告和 PL/pgSQL 陳述式。陳述式也可以是巢狀區塊或子區塊。

以分號作為宣告和陳述式的結尾。在區塊或子區塊中的 END 關鍵字後面加上分號。請勿在關鍵字 DECLARE 和 BEGIN 後面使用分號。

您可以混合大小寫來撰寫所有關鍵字和識別碼。識別碼除非以雙引號括住，否則會隱含轉換為小寫。

雙連字號 (--) 表示註解開始，一直延續到行尾。/* 表示區塊註解開始，一直延續到出現下一個 */。您不能將區塊註解巢狀化。不過，您可以用區塊註解圍住雙連字號註解，雙連字號可以隱藏區塊註解分隔符號 /* 和 */。

在區塊的陳述式區段中，任何陳述式可以是子區塊。您可以使用子區塊進行邏輯分組，或將變數區域化給一小組陳述式。

```
[ <<label>> ]
[ DECLARE
  declarations ]
BEGIN
  statements
END [ label ];
```

在區塊前面的宣告區段中所宣告的變數，在每次進入區塊時會初始化為其預設值。換言之，在每次函數呼叫時不只初始化一次。

下列顯示一個範例。

```
CREATE PROCEDURE update_value() AS $$
DECLARE
  value integer := 20;
BEGIN
  RAISE NOTICE 'Value here is %', value; -- Value here is 20
  value := 50;
  --
  -- Create a subblock
  --
  DECLARE
    value integer := 80;
  BEGIN
    RAISE NOTICE 'Value here is %', value; -- Value here is 80
  END;

  RAISE NOTICE 'Value here is %', value; -- Value here is 50
END;
$$ LANGUAGE plpgsql;
```

使用標籤來識別要在 EXIT 陳述式中使用的區塊，或限定區塊中宣告的變數名稱。

請勿混淆 PL/pgSQL 中用於陳述式分組的 BEGIN/END 與用於交易控制的資料庫命令。PL/pgSQL 中的 BEGIN 和 END 只用於分組。而無法用來啟動或結束交易。

變數宣告

在區塊的 DECLARE 區段中宣告區塊的所有變數，但迴圈變數除外。變數可以使用任何有效的 Amazon Redshift 資料類型。關於支援的資料類型，請參閱[資料類型](#)。

PL/pgSQL 變數可以是 Amazon Redshift 支援的任何資料類型，還有 RECORD 和 refcursor。如需 RECORD 的相關資訊，請參閱[記錄類型](#)。如需 refcursor 的相關資訊，請參閱[游標](#)。

```
DECLARE
name [ CONSTANT ] type [ NOT NULL ] [ { DEFAULT | := } expression ];
```

以下，您可以找到變數宣告範例。

```
customerID integer;
numberofitems numeric(6);
link varchar;
onerow RECORD;
```

對整數範圍反覆運算的 FOR 迴圈的迴圈變數，自動宣告為整數變數。

DEFAULT 子句 (如有提供) 指定進入區塊時指派給變數的初始值。如果未提供 DEFAULT 子句，變數會初始化為 SQL NULL 值。CONSTANT 選項防止指派變數，因此在整個區塊內，值維持固定。如果指定 NOT NULL，則指派空值會導致執行時間錯誤。所有宣告為 NOT NULL 的變數必須指定非空值的預設值。

每次進入區塊時會評估預設值。例如，假設指派 now() 給 timestamp 類型的變數，則變數會具有目前函數呼叫的時間，而非預先編譯函數時的時間。

```
quantity INTEGER DEFAULT 32;
url VARCHAR := 'http://mysite.com';
user_id CONSTANT INTEGER := 10;
```

refcursor 資料類型是預存程序內的游標變數的資料類型。預存程序可以傳回 refcursor 值。如需詳細資訊，請參閱[從存儲過程返回結果集](#)。

別名宣告

如果預存程序的簽章省略引數名稱，您可以宣告該引數的別名。

```
name ALIAS FOR $n;
```

內建變數

支援下列內建變數：

- FOUND
- SQLSTATE
- SQLERRM
- GET DIAGNOSTICS integer_var := ROW_COUNT;

FOUND 是布林值類型的特殊變數。在每個程序呼叫內，FOUND 剛開始為 false。FOUND 由下列類型的陳述式來設定：

- SELECT INTO

將 FOUND 設為 true 表示傳回一列，設為 false 表示未傳回任何列。

- UPDATE、INSERT 和 DELETE

將 FOUND 設為 true 表示至少一列受影響，設為 false 表示不影響任何列。

- FETCH

將 FOUND 設為 true 表示傳回一列，設為 false 表示未傳回任何列。

- FOR 陳述式

將 FOUND 設為 true 表示 FOR 陳述式反覆運算一或多次，否則為設為 false。這適用於 FOR 陳述式的所有三種變體：整數 FOR 迴圈、記錄集 FOR 迴圈，以及動態記錄集 FOR 迴圈。

在 FOR 迴圈結束時設定 FOUND。在迴圈執行期內，FOR 陳述式不會修改 FOUND。不過，迴圈主體內的其他陳述式執行可能變更它。

下列顯示一個範例。

```
CREATE TABLE employee(empname varchar);
CREATE OR REPLACE PROCEDURE show_found()
AS $$
DECLARE
```

```
myrec record;
BEGIN
  SELECT INTO myrec * FROM employee WHERE empname = 'John';
  IF NOT FOUND THEN
    RAISE EXCEPTION 'employee John not found';
  END IF;
END;
$$ LANGUAGE plpgsql;
```

在例外處理常式內，特殊變數 SQLSTATE 包含錯誤碼，對應於已引發的例外狀況。特殊變數 SQLERRM 包含與例外狀況相關聯的錯誤訊息。這些變數在例外處理常式外未定義，如果使用，則會顯示錯誤。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE sqlstate_sqlerrm() AS
$$
BEGIN
  UPDATE employee SET firstname = 'Adam' WHERE lastname = 'Smith';
  EXECUTE 'select invalid';
  EXCEPTION WHEN OTHERS THEN
    RAISE INFO 'error message SQLERRM %', SQLERRM;
    RAISE INFO 'error message SQLSTATE %', SQLSTATE;
END;
$$ LANGUAGE plpgsql;
```

ROW_COUNT 與 GET DIAGNOSTICS 命令搭配使用。它顯示傳送到 SQL 引擎的最後一個 SQL 命令所處理的列數。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE sp_row_count() AS
$$
DECLARE
  integer_var int;
BEGIN
  INSERT INTO tbl_row_count VALUES(1);
  GET DIAGNOSTICS integer_var := ROW_COUNT;
  RAISE INFO 'rows inserted = %', integer_var;
END;
$$ LANGUAGE plpgsql;
```

記錄類型

RECORD 類型不是真的資料類型，只是預留位置。記錄類型變數採用 SELECT 或 FOR 命令期間指派給它們的列的實際列結構。記錄變數在每次被指派一個值時可能變更子結構。記錄變數在第一次被指派之前沒有子結構。嘗試存取其中的欄位會擲出執行時間錯誤。

```
name RECORD;
```

下列顯示一個範例。

```
CREATE TABLE tbl_record(a int, b int);
INSERT INTO tbl_record VALUES(1, 2);
CREATE OR REPLACE PROCEDURE record_example()
LANGUAGE plpgsql
AS $$
DECLARE
    rec RECORD;
BEGIN
    FOR rec IN SELECT a FROM tbl_record
    LOOP
        RAISE INFO 'a = %', rec.a;
    END LOOP;
END;
$$;
```

支援的 PL/pgSQL 陳述式

PL/pgSQL 陳述式以程序性建構 (包括迴圈和條件式表達式) 來擴增 SQL 命令，以控制邏輯流程。可使用大部分 SQL 命令，包括資料處理語言 (DML) (例如 COPY、UNLOAD 和 INSERT) 和資料定義語言 (DDL) (例如 CREATE TABLE)。如需完整 SQL 命令的清單，請參閱 [SQL 命令](#)。此外，Amazon Redshift 還支援下列 PL/pgSQL 陳述式。

主題

- [指派](#)
- [SELECT INTO](#)
- [無操作](#)
- [動態 SQL](#)
- [傳回](#)
- [條件式 : IF](#)

- [條件式：CASE](#)
- [迴圈](#)
- [游標](#)
- [RAISE](#)
- [交易控制](#)

指派

指派陳述式可指派值給變數。表達式必須傳回單一值。

```
identifier := expression;
```

也接受使用非標準的 = 來指派 (而不是 :=)。

如果表達式的資料類型不符合變數的資料類型，或變數具有大小或精確度，則會隱含轉換結果值。

如下列範例所示。

```
customer_number := 20;  
tip := subtotal * 0.15;
```

SELECT INTO

SELECT INTO 陳述式將多欄 (但只有一列) 的結果指派給一個記錄變數或一個純量變數清單。

```
SELECT INTO target select_expressions FROM ...;
```

在上述語法中，*target* 可以是記錄變數，或簡單變數和記錄欄位的逗號分隔清單。*select_expressions* 清單和命令的剩餘部分與一般 SQL 中相同。

如果以變數清單作為 *target*，選取的值必須完全符合目標的結構，否則會發生執行時間錯誤。當記錄變數為目標時，它本身會自動設定為查詢結果欄的列類型。

INTO 子句幾乎可出現在 SELECT 陳述式中的任意處。通常就出現在 SELECT 子句之後，或就在 FROM 子句之前。亦即，就出現在 *select_expressions* 清單之前或之後。

如果查詢未傳回任何列，NULL 值會指派給 *target*。如果查詢傳回多列，第一列會指派給 *target*，其餘捨棄。除非陳述式包含 ORDER BY，否則無法確定第一列。

若要判斷指派是否傳回至少一行，請使用特殊的 FOUND 變數。

```
SELECT INTO customer_rec * FROM cust WHERE custname = lname;
IF NOT FOUND THEN
  RAISE EXCEPTION 'employee % not found', lname;
END IF;
```

若要測試記錄結果是否為空值，您可以使用 IS NULL 條件。無法判斷是否已捨棄任何其他列。下列範例處理未傳回任何列的情況。

```
CREATE OR REPLACE PROCEDURE select_into_null(return_webpage OUT varchar(256))
AS $$
DECLARE
  customer_rec RECORD;
BEGIN
  SELECT INTO customer_rec * FROM users WHERE user_id=3;
  IF customer_rec.webpage IS NULL THEN
    -- user entered no webpage, return "http://"
    return_webpage = 'http://';
  END IF;
END;
$$ LANGUAGE plpgsql;
```

無操作

無操作陳述式 (NULL;) 是不執行任何動作的預留位置陳述式。無操作陳述式可以表示 IF-THEN-ELSE 鏈的一個分支是空的。

```
NULL;
```

動態 SQL

若要產生動態命令，以便每次從 PL/pgSQL 預存程序執行時，都可涉及不同的資料表或不同的資料類型，請使用 EXECUTE 陳述式。

```
EXECUTE command-string [ INTO target ];
```

在上述語法中，*command-string* 是產生字串 (文字類型) 的表達式，而此字串包含要執行的命令。此 *command-string* 值會傳送到 SQL 引擎。在命令字串上不會替換 PL/pgSQL 變數。您必須在建構命令字串時插入變數的值。

Note

您無法從動態 SQL 內使用 COMMIT 和 ROLLBACK 陳述式。如需在預存程序內使用 COMMIT 和 ROLLBACK 陳述式的相關資訊，請參閱[管理交易](#)。

使用動態命令時，您通常需要處理單引號逸出。建議使用 \$ 符號引用來圍住函數主體中放在引號內的固定字串。在建構的查詢中要插入的動態值需要特別處理，因為動態值本身可能包含引號。下列範例對整個函數採用 \$ 符號引用，因此引號不需要加倍。

```
EXECUTE 'UPDATE tbl SET '  
  || quote_ident(colname)  
  || ' = '  
  || quote_literal(newvalue)  
  || ' WHERE key = '  
  || quote_literal(keyvalue);
```

上述範例顯示函數 `quote_ident(text)` 和 `quote_literal(text)`。此範例將包含欄和資料表識別碼的變數傳給 `quote_ident` 函數。也將包含所建構命令中常值字串的變數傳給 `quote_literal` 函數。這兩個函數都採取適當步驟，傳回分別以雙引號或單引號括住的輸入文字，並適當地逸出任何內嵌的特殊字元。

\$ 符號引用僅適用於括住固定文字。請勿將上述範例寫成下列格式。

```
EXECUTE 'UPDATE tbl SET '  
  || quote_ident(colname)  
  || ' = $$'  
  || newvalue  
  || '$$ WHERE key = '  
  || quote_literal(keyvalue);
```

不能這樣做，因為如果 `newvalue` 的內容碰巧包含 \$\$，則範例會失敗。您可能選擇的任何其他 \$ 符號引用分隔符號也有相同問題。若要安心地括住無法事先得知的文字，請使用 `quote_literal` 函數。

傳回

RETURN 陳述式從預存程序傳回到發起人。

```
RETURN;
```


下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE return_example(a int)
AS $$
BEGIN
  FOR b in 1..10 LOOP
    IF b < a THEN
      RAISE INFO 'b = %', b;
    ELSE
      RETURN;
    END IF;
  END LOOP;
END;
$$ LANGUAGE plpgsql;
```

條件式：IF

在 Amazon Redshift 所使用的 PL/pgSQL 語言中，IF 條件式陳述式有下列形式：

- IF ... THEN

```
IF boolean-expression THEN
  statements
END IF;
```

下列顯示一個範例。

```
IF v_user_id <> 0 THEN
  UPDATE users SET email = v_email WHERE user_id = v_user_id;
END IF;
```

- IF ... THEN ... ELSE

```
IF boolean-expression THEN
  statements
ELSE
  statements
END IF;
```

下列顯示一個範例。

```
IF parentid IS NULL OR parentid = ''
THEN
    return_name = fullname;
    RETURN;
ELSE
    return_name = hp_true_filename(parentid) || '/' || fullname;
    RETURN;
END IF;
```

- IF ... THEN ... ELSIF ... THEN ... ELSE

關鍵字 ELSIF 也可以拼寫為 ELSEIF。

```
IF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
[ ELSIF boolean-expression THEN
    statements
    ...] ]
[ ELSE
    statements ]
END IF;
```

下列顯示一個範例。

```
IF number = 0 THEN
    result := 'zero';
ELSIF number > 0 THEN
    result := 'positive';
ELSIF number < 0 THEN
    result := 'negative';
ELSE
    -- the only other possibility is that number is null
    result := 'NULL';
END IF;
```

條件式：CASE

在 Amazon Redshift 所使用的 PL/pgSQL 語言中，CASE 條件式陳述式有下列形式：

- 簡單 CASE

```
CASE search-expression
WHEN expression [, expression [ ... ]] THEN
    statements
[ WHEN expression [, expression [ ... ]] THEN
    statements
... ]
[ ELSE
    statements ]
END CASE;
```

簡單 CASE 陳述式可根據運算元等式而有條件地執行。

search-expression 值會評估一次，然後連續地與 WHEN 子句中的每個 *expression* 相比較。如果發現相符，則對應的 *statements* 會執行，接著控制權就移轉到 END CASE 之後的下一個陳述式。不會評估後續的 WHEN expressions。如果找不到相符，則 ELSE *statements* 會執行。不過，如果 ELSE 不存在，則會引發 CASE_NOT_FOUND 例外狀況。

下列顯示一個範例。

```
CASE x
WHEN 1, 2 THEN
    msg := 'one or two';
ELSE
    msg := 'other value than one or two';
END CASE;
```

- 搜尋的 CASE

```
CASE
WHEN boolean-expression THEN
    statements
[ WHEN boolean-expression THEN
    statements
... ]
[ ELSE
    statements ]
END CASE;
```

搜尋形式的 CASE 可根據布林值表達式的真實性而有條件地執行。

每個 WHEN 子句的 *boolean-expression* 會依次評估，直到發現產生 true 為止。然後對應的陳述式會執行，接著控制權就移轉到 END CASE 之後的下一個陳述式。不會評估後續的 WHEN *expressions*。如果找不到 true 結果，則 ELSE *statements* 會執行。不過，如果 ELSE 不存在，則會引發 CASE_NOT_FOUND 例外狀況。

下列顯示一個範例。

```
CASE
WHEN x BETWEEN 0 AND 10 THEN
  msg := 'value is between zero and ten';
WHEN x BETWEEN 11 AND 20 THEN
  msg := 'value is between eleven and twenty';
END CASE;
```

迴圈

在 Amazon Redshift 所使用的 PL/pgSQL 語言中，迴圈陳述式有下列形式：

- 簡單迴圈

```
[<<label>>]
LOOP
  statements
END LOOP [ label ];
```

簡單迴圈定義無條件的迴圈，將會無限期重複，直到由 EXIT 或 RETURN 陳述式終止為止。巢狀迴圈內的 EXIT 和 CONTINUE 陳述式可使用選用標籤，以指定 EXIT 和 CONTINUE 陳述式所指的迴圈。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE simple_loop()
LANGUAGE plpgsql
AS $$
BEGIN
  <<simple_while>>
  LOOP
    RAISE INFO 'I am raised once';
    EXIT simple_while;
```

```
    RAISE INFO 'I am not raised';
END LOOP;
    RAISE INFO 'I am raised once as well';
END;
$$;
```

- 結束迴圈

```
EXIT [ label ] [ WHEN expression ];
```

如果 *label* 不存在，最內層迴圈會終止，END LOOP 之後的陳述式會接下去執行。如果 *label* 存在，它必須是巢狀迴圈或區塊的目前或某個外層的標籤。然後，具名迴圈或區塊會終止，控制權會延續到迴圈或區塊相對應 END 之後的陳述式。

如果指定 WHEN，只有在 *expression* 為 true 時，迴圈才會結束。否則，控制權會移轉到 EXIT 之後的陳述式。

您可以對所有類型的迴圈使用 EXIT；不限於用於無條件的迴圈。

與 BEGIN 區塊一起使用時，EXIT 會將控制權移轉到區塊結束之後的下一個陳述式。為此，必須使用標籤。無標籤的 EXIT 絕不可能有對稱的 BEGIN 區塊。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE simple_loop_when(x int)
LANGUAGE plpgsql
AS $$
DECLARE i INTEGER := 0;
BEGIN
    <<simple_loop_when>>
    LOOP
        RAISE INFO 'i %', i;
        i := i + 1;
        EXIT simple_loop_when WHEN (i >= x);
    END LOOP;
END;
$$;
```

- 繼續迴圈

```
CONTINUE [ label ] [ WHEN expression ];
```

如果未提供 *label*，執行會跳到最內層迴圈的下一次反覆運算。亦即會略過迴圈主體中剩餘的所有陳述式。然後，控制權會返回到迴圈控制表達式 (如果有的話)，以決定是否需要再一次迴圈反覆運算。如果 *label* 存在，它指定繼續執行的迴圈的標籤。

如果指定 WHEN，只有在 *expression* 為 true 時，迴圈的下一次反覆運算才會開始。否則，控制權會移轉到 CONTINUE 之後的陳述式。

您可以對所有類型的迴圈使用 CONTINUE；不限於用於無條件的迴圈。

```
CONTINUE mylabel;
```

- WHILE 迴圈

```
[<<label>>]
WHILE expression LOOP
  statements
END LOOP [ label ];
```

只要 *boolean-expression* 評估為 true，WHILE 陳述式會重複一連串陳述式。就在進入迴圈主體之前會檢查表達式。

下列顯示一個範例。

```
WHILE amount_owed > 0 AND gift_certificate_balance > 0 LOOP
  -- some computations here
END LOOP;

WHILE NOT done LOOP
  -- some computations here
END LOOP;
```

- FOR 迴圈 (整數變體)

```
[<<label>>]
FOR name IN [ REVERSE ] expression .. expression LOOP
  statements
END LOOP [ label ];
```

FOR 迴圈 (整數變數) 建立迴圈對整數值範圍反覆運算。變數名稱會自動定義為整數類型，並且只在迴圈內結束。迴圈內會忽略變數名稱的任何現有定義。定義範圍下限和上限的兩個表達式會在進入迴圈時評估一次。如果您指定 REVERSE，則每一次反覆運算後會減去間距值，而不是相加。

如果下限大於上限 (或在 REVERSE 情況下是小於)，迴圈主體不會執行。不會引發錯誤。

如果標籤附加到 FOR 迴圈，則您可以使用該標籤，以限定名稱來參考整數迴圈變數。

下列顯示一個範例。

```
FOR i IN 1..10 LOOP
  -- i will take on the values 1,2,3,4,5,6,7,8,9,10 within the loop
END LOOP;

FOR i IN REVERSE 10..1 LOOP
  -- i will take on the values 10,9,8,7,6,5,4,3,2,1 within the loop
END LOOP;
```

- FOR 迴圈 (結果集變體)

```
[<<label>>]
FOR target IN query LOOP
  statements
END LOOP [ label ];
```

target 是記錄變數，或純量變數的逗號分隔清單。從查詢產生的每一列會連續指派給目標，而每一列會執行一次迴圈主體。

FOR 迴圈 (結果集變體) 可讓預存程序逐一查看查詢的結果，並相應地操作該資料。

下列顯示一個範例。

```
CREATE PROCEDURE cs_refresh_reports() AS $$
DECLARE
  reports RECORD;
BEGIN
  FOR reports IN SELECT * FROM cs_reports ORDER BY sort_key LOOP
    -- Now "reports" has one record from cs_reports
    EXECUTE 'INSERT INTO ' || quote_ident(reports.report_name) || ' ' ||
reports.report_query;
  END LOOP;
```

```
RETURN;  
END;  
$$ LANGUAGE plpgsql;
```

- FOR 迴圈搭配動態 SQL

```
[<<label>>]  
FOR record_or_row IN EXECUTE text_expression LOOP  
    statements  
END LOOP;
```

FOR 迴圈搭配動態 SQL 可讓預存程序逐一查看動態查詢的結果，並相應地操作該資料。

下列顯示一個範例。

```
CREATE OR REPLACE PROCEDURE for_loop_dynamic_sql(x int)  
LANGUAGE plpgsql  
AS $$  
DECLARE  
    rec RECORD;  
    query text;  
BEGIN  
    query := 'SELECT * FROM tbl_dynamic_sql LIMIT ' || x;  
    FOR rec IN EXECUTE query  
    LOOP  
        RAISE INFO 'a %', rec.a;  
    END LOOP;  
END;  
$$;
```

游標

您可以設定游標，而不要一次執行整個查詢。cursor 封裝查詢，每次在查詢結果中讀取幾列。這麼做的一個理由是當結果包含大量的列時，避免記憶體溢位。另一個理由是為了傳回預存程序已建立之游標的參考，此游標可讓發起人讀取列。此方法可以很有效率地從預存程序傳回很大的列集。

若要在 NONATOMIC 預存程序中使用游標，請將游標迴圈置於 START TRANSACTION...COMMIT 之間。

若要設定游標，首先要宣告游標變數。在 PL/pgSQL 中存取游標完全是透過游標變數，該變數一定是特殊資料類型 refcursor。refcursor 資料類型只是保留游標的參考。

您可以將變數宣告為 `refcursor` 類型，以建立游標變數。或者，您可以使用如下的游標宣告語法。

```
name CURSOR [ ( arguments ) ] FOR query ;
```

在上述語法中，*arguments* (如有指定) 是 *name datatype* 組的逗號分隔清單，每一組定義要由 *query* 中的參數值所取代的名稱。稍後開啟游標時指定用來替換這些名稱的實際值。

如下列範例所示。

```
DECLARE
  curs1 refcursor;
  curs2 CURSOR FOR SELECT * FROM tenk1;
  curs3 CURSOR (key integer) IS SELECT * FROM tenk1 WHERE unique1 = key;
```

這三個變數全部都是資料類型 `refcursor`，但第一個可用於任何查詢。相反地，第二個已繫結完整指定的查詢，而最後一個已繫結參數化查詢。開啟游標時，`key` 值會換成整數參數值。變數 `curs1` 可說成未繫結，因為沒有繫結至任何特定查詢。

游標必須先開始，才能用來擷取列。PL/pgSQL 有三種形式的 `OPEN` 陳述式，其中兩種使用未繫結的游標變數，第三種使用繫結的游標變數：

- 開啟以選取：開啟游標變數，並給予要執行的指定查詢。游標不能已開啟。另外，還必須已宣告為未繫結的游標 (亦即，宣告為簡單的 `refcursor` 變數)。SELECT 查詢的處理方式與 PL/pgSQL 中的其他 SELECT 陳述式相同。

```
OPEN cursor_name FOR SELECT ...;
```

下列顯示一個範例。

```
OPEN curs1 FOR SELECT * FROM foo WHERE key = mykey;
```

- 開啟以執行：開啟游標變數，並給予要執行的指定查詢。游標不能已開啟。另外，還必須已宣告為未繫結的游標 (亦即，宣告為簡單的 `refcursor` 變數)。將查詢指定為字串表達式的方式與 `EXECUTE` 命令中相同。此方法很有彈性，可讓查詢隨著每一次執行而變化。

```
OPEN cursor_name FOR EXECUTE query_string;
```

下列顯示一個範例。

```
OPEN curs1 FOR EXECUTE 'SELECT * FROM ' || quote_ident($1);
```

- 開啟繫結的游標：這種 OPEN 用於開啟已在宣告時繫結查詢的游標變數。游標不能已開啟。僅當游標宣告為接受引數時，實際引數值表達式的清單才必須出現。查詢中會替換這些值。

```
OPEN bound_cursor_name [ ( argument_values ) ];
```

下列顯示一個範例。

```
OPEN curs2;  
OPEN curs3(42);
```

開啟游標後，您可以利用下述的陳述式來使用它。這些陳述式不一定要出現在開啟游標的同一預存程序中。您可以從預存程序傳回 `refcursor` 值，並讓發起人繼續操作游標。交易結束時，所有入口會隱含關閉。因此，只有在交易結束後，您才可以使用 `refcursor` 值來參考開啟的游標。

- `FETCH` 從游標中將下一列擷取到目標。此目標可以是列變數、記錄變數，或簡單變數的逗號分隔清單，如同 `SELECT INTO` 一樣。如同 `SELECT INTO` 一樣，您可以檢查特殊變數 `FOUND`，查明是否已取得一行。

```
FETCH cursor INTO target;
```

下列顯示一個範例。

```
FETCH curs1 INTO rowvar;
```

- `CLOSE` 會將已開啟的游標的基礎入口關閉。在交易結束之前，您可以使用此陳述式來提早釋放資源。您也可以使用此陳述式來釋放游標變數，供再次開啟。

```
CLOSE cursor;
```

下列顯示一個範例。

```
CLOSE curs1;
```

RAISE

使用 RAISE level 陳述式來報告訊息和引發錯誤。

```
RAISE level 'format' [, variable [, ...]];
```

可能的等級包括 NOTICE、INFO、LOG、WARNING 和 EXCEPTION。EXCEPTION 會引發錯誤，通常會取消目前的交易。其他等級只產生不同優先等級的訊息。

在格式字串內，% 會換成下一個選用引數的字串表示法。撰寫 %% 以發出常值 %。目前，選用引數必須是簡單變數，不是表達式，而格式必須是簡單字串常值。

在下列範例中，v_job_id 的值會取代字串中的 %。

```
RAISE NOTICE 'Calling cs_create_job(%)', v_job_id;
```

使用 RAISE 陳述式重新擲出例外狀況處理區塊捕獲的例外狀況。這個陳述式僅在 NONATOMIC 模式預存程序的例外狀況處理區塊中有效。

```
RAISE;
```

交易控制

您可以在 Amazon Redshift 所使用的 PL/pgSQL 語言中使用交易控制陳述式。如需在預存程序內使用 COMMIT、ROLLBACK 和 TRUNCATE 陳述式的相關資訊，請參閱[管理交易](#)。

在 NONATOMIC 模式預存程序中，使用 START TRANSACTION 啟動交易區塊。

```
START TRANSACTION;
```

Note

PL/pgSQL 陳述式 START TRANSACTION 與 SQL 命令 START TRANSACTION 有以下不同：

- 在預存程序中，START TRANSACTION 與 BEGIN 不同義。
- PL/pgSQL 陳述式不支援選擇性的隔離層級和存取許可關鍵字。

Amazon Redshift 中的具體化視觀表

本節說明如何在 Amazon Redshift 中建立和使用具體化檢視。具體化檢視是存放查詢結果的資料庫物件，可用於改善效能和效率。

在資料倉儲環境中，應用程式通常必須在大型資料表上執行複雜的查詢。範例是對包含數十億列的資料表執行多資料表聯結和彙總的SELECT陳述式。處理這些查詢相當耗費系統資源，且運算結果也會耗費時間。

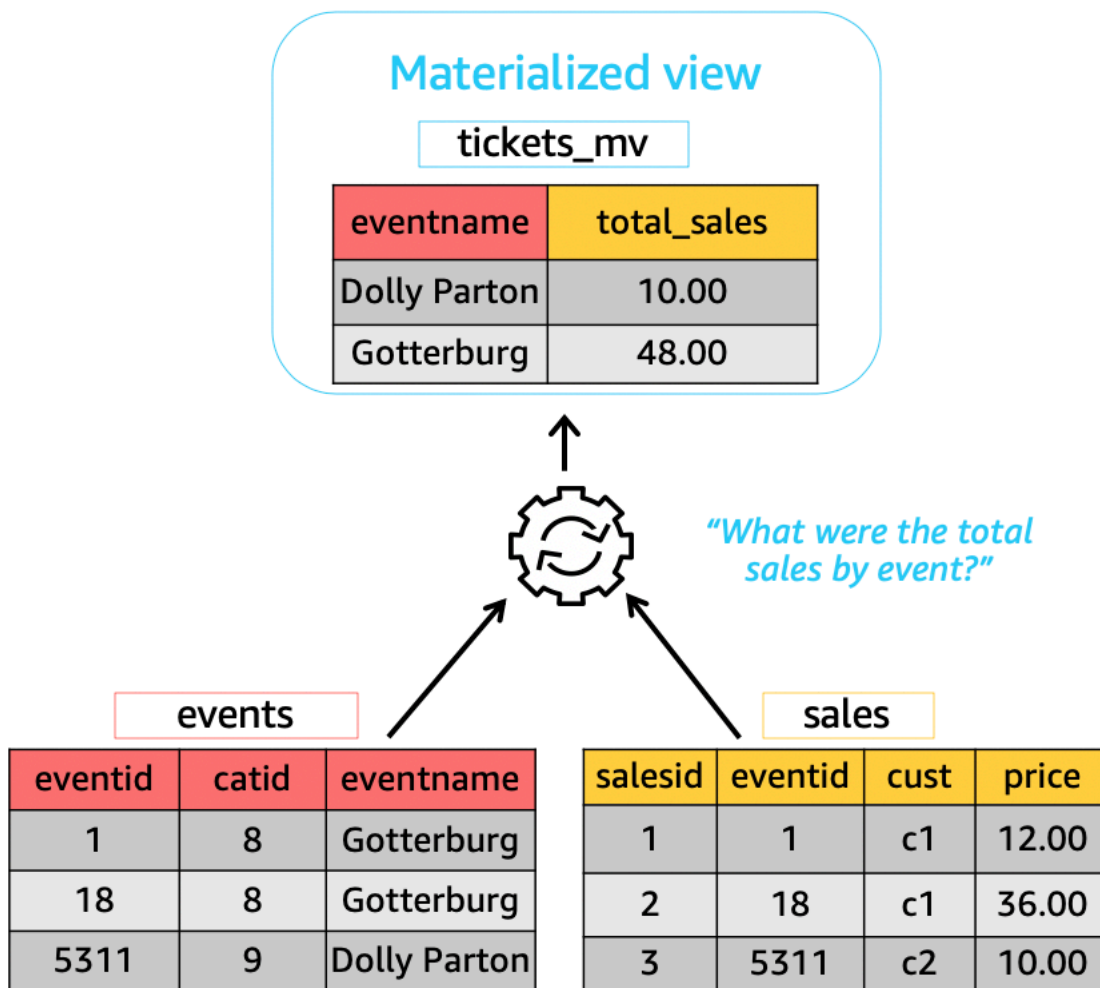
Amazon Redshift 中的具體化視觀表提供處理這些問題的方式。具體化視觀表包含預先計算的結果集，以一個或多個基礎資料表上的SQL查詢為基礎。您可以發出SELECT陳述式來查詢具體化檢視，方法與查詢資料庫中其他資料表或檢視的方式相同。Amazon Redshift 會從具體化視觀表傳回預先計算的結果，完全不需要存取基底資料表。從使用者的觀點而言，傳回查詢結果的速度會比從基底資料表擷取相同的資料時要快上許多。

具體化視觀表在加速可預測和重複的查詢時特別有用。相較於對大型資料表 (例如彙整或多個聯結) 執行耗費資源的查詢，應用程式可以改為查詢具體化視觀表，並擷取預先運算的結果集。例如，請考慮使用一組查詢來填入儀表板的案例，例如 Amazon QuickSight。這種使用案例便相當適合使用具體化視觀表，因為查詢是可預期的，且會一再地重複進行。

您可以根據其他具體化視觀表來定義具體化視觀表。使用具體化視觀表上的具體化視觀表來擴充具體化視觀表的功能。在這種方法中，現有具體化視觀表的角色會與查詢用來擷取資料的基底資料表相同。

此方法對於針對不同的彙總或 GROUP BY 選項重複使用預先計算聯結特別有用。例如，結合客戶資訊 (包含數百萬個資料列) 與項目訂單詳細資訊 (包含數十億個資料列) 的具體化視觀表。這是重複按需計算的高成本查詢。您可以針對在此具體化視觀表上建立的具體化視觀表使用不同的 GROUP BY 選項，並與其他資料表聯結。這樣做可以節省計算時間，否則每次都要執行高成本的基礎聯結。[STV_MV_DEPS](#) 資料表會顯示具體化視觀表在其他具體化視觀表上的相依性。

當您建立具體化檢視時，Amazon Redshift 會執行使用者指定的SQL陳述式，以從基礎資料表或資料表收集資料，並儲存結果集。下圖提供SQL查詢使用兩個基礎資料表 tickets_mv events和定義的具體化檢視概觀sales。



然後，您可以在查詢中使用這些具體化視觀表來加速查詢。此外，即使查詢未明確參照具體化視觀表，Amazon Redshift 也可以自動重寫這些查詢以使用具體化視觀表。當您無法將查詢變更為使用具體化視觀表時，自動重寫查詢在提升效能方面尤其強大。

若要更新具體化檢視中的資料，您可以隨時使用REFRESHMATERIALIZEDVIEW陳述式手動重新整理具體化檢視。Amazon Redshift 會識別在基底資料表或資料表中發生的變更，然後將這些變更套用到具體化視觀表。由於自動重寫查詢需要具體化視觀表維持在最新狀態，所以身為具體化視觀表擁有者，請務必在基礎資料表變更時重新整理具體化視觀表。

Amazon Redshift 提供了幾種讓具體化視觀圖保持在最新狀態的方法，以便自動重寫。您可以使用自動重新整理選項設定具體化視觀表，以在更新具體化視觀表的基礎資料表時重新整理具體化視觀表。此自動重新整理操作會在叢集資源可用時執行，以盡量減少對其他工作負載的干擾。由於自動重新整理排程取決於工作負載，因此您可以更好地控制 Amazon Redshift 重新整理具體化視觀表的時機。您可以使用 Amazon Redshift 排程器API和主控台整合來排程具體化檢視重新整理任務。如需查詢排程的相關資訊，請參閱在 [Amazon Redshift 主控台上排程查詢](#)。

當具體化視觀表的資料有服務層級協議 up-to-date (SLA) 要求時，這樣做特別有用。您也可以手動重新整理任何可以自動重新整理的具體化視觀表。如需如何建立具體化視觀表的相關資訊，請參閱[CREATE MATERIALIZED VIEW](#)。

您可以發出SELECT陳述式來查詢具體化檢視。如需如何查詢具體化視觀表的相關資訊，請參閱[具體化檢視查詢](#)。結果集最終會隨著在基礎資料表中插入、更新和刪除資料而過時。您可以隨時重新整理具體化視觀表，以使用基礎資料表中的最新變更來更新它。如需如何重新整理具體化視觀表的相關資訊，請參閱[REFRESH MATERIALIZED VIEW](#)。

如需用於建立和管理具體化檢視的SQL命令詳細資訊，請參閱下列命令主題：

- [CREATE MATERIALIZED VIEW](#)
- [ALTER MATERIALIZED VIEW](#)
- [REFRESH MATERIALIZED VIEW](#)
- [DROP MATERIALIZED VIEW](#)

如需用於監控具體化視觀表的系統資料表和檢視相關資訊，請參閱下列主題：

- [STV_MV_INFO](#)
- [STL_MV_STATE](#)
- [SVL_MV_REFRESH_STATUS](#)
- [STV_MV_DEPS](#)

主題

- [具體化檢視查詢](#)
- [自動查詢重寫以使用具體化視檢視](#)
- [重新整理具體化視觀表](#)
- [自動具體化視觀表](#)
- [在具體化視觀表中使用使用者定義的函數 \(UDF \)](#)
- [將擷取串流至具體化視觀表](#)

具體化檢視查詢

您可以在任何SQL查詢中使用具體化檢視，方法是參考具體化檢視名稱作為資料來源，例如資料表或標準檢視。

當查詢存取具體化視觀表時，其只會看見最近一次重新整理後儲存在具體化視觀表中的資料。因此，查詢可能無法看見具體化視觀表所對應基底資料表中所有最新的變更。

如果其他使用者想要查詢具體化檢視，具體化檢視的擁有者會將SELECT許可授予這些使用者。其他使用者不需要具備基礎資料表的SELECT許可。具體化檢視的擁有者也可以撤銷其他使用者的SELECT許可，以防止他們查詢具體化檢視。

如果具體化檢視的擁有者不再擁有基礎資料表的SELECT許可：

- 擁有者無法再查詢具體化視觀表。
- 具有具體化檢視SELECT許可的其他使用者無法再查詢具體化檢視。

以下範例會查詢 tickets_mv 具體化視觀表。如需用於建立具體化檢視的SQL命令的詳細資訊，請參閱 [CREATE MATERIALIZED VIEW](#)。

```
SELECT sold
FROM tickets_mv
WHERE catgroup = 'Concerts';
```

因為查詢結果是預先運算的，因此無須存取基礎資料表 (category、event 和 sales)。Amazon Redshift 可以直接從 tickets_mv 傳回結果。

自動查詢重寫以使用具體化視檢視

您可以使用 Amazon Redshift 中具體化視觀表的自動查詢重寫功能，讓 Amazon Redshift 重寫查詢以使用具體化視觀表。這樣做可以加速查詢工作負載，即使是未明確參考具體化視觀表的查詢也一樣。當 Amazon Redshift 重寫查詢時，其只會使用最新的具體化視觀表。

使用須知

若要檢查查詢是否使用自動重寫，您可以檢查查詢計畫或 STL_EXPLAIN。以下顯示陳述SELECT式和原始查詢計畫的EXPLAIN輸出。

```
SELECT catgroup, SUM(qtysold) AS sold
FROM category c, event e, sales s
WHERE c.catid = e.catid AND e.eventid = s.eventid
GROUP BY 1;

EXPLAIN
  XN HashAggregate (cost=920021.24..920021.24 rows=1 width=35)
```

```

-> XN Hash Join DS_BCAST_INNER (cost=440004.53..920021.22 rows=4 width=35)
    Hash Cond: ("outer".eventid = "inner".eventid)
-> XN Seq Scan on sales s (cost=0.00..7.40 rows=740 width=6)
-> XN Hash (cost=440004.52..440004.52 rows=1 width=37)
    -> XN Hash Join DS_BCAST_INNER (cost=0.01..440004.52 rows=1 width=37)
        Hash Cond: ("outer".catid = "inner".catid)
        -> XN Seq Scan on event e (cost=0.00..2.00 rows=200 width=6)
        -> XN Hash (cost=0.01..0.01 rows=1 width=35)
            -> XN Seq Scan on category c (cost=0.00..0.01 rows=1
width=35)

```

以下顯示成功自動重寫後的EXPLAIN輸出。此輸出包括查詢計劃中具體化視觀表的掃描，其取代了原始查詢計劃的一部分。

```

* EXPLAIN
  XN HashAggregate (cost=11.85..12.35 rows=200 width=41)
    -> XN Seq Scan on mv_tbl__tickets_mv__0 derived_table1 (cost=0.00..7.90
rows=790 width=41)

```

無論重新整理策略為何，例如自動、排程或手動，都 up-to-date 只會考慮自動重寫查詢的（全新）具體化檢視。因此，原始查詢會傳回 up-to-date 結果。在查詢中明確參照具體化視觀表時，Amazon Redshift 會存取具體化視觀表中目前儲存的資料。此資料可能不會反映具體化視觀表之基礎資料表的最新變更。

您可以使用在叢集 1.0.20949 版或更新版本上建立的具體化視觀表自動查詢重寫功能。

您可以使用 SET mv_enable_aqmv_for_session 來停止工作階段層級的自動查詢重寫 FALSE。

限制

以下是針對具體化視觀表使用自動查詢重寫的限制。

- 自動查詢重寫適用於不參照或包含下列任何項目的具體化視觀表：
 - 子查詢
 - 左側、右側或完整外部聯結
 - 設定操作
 - 任何彙總函數，除了 SUM、COUNT、MAX、MIN 和 AVG。(這些是唯一可與自動查詢重寫搭配使用的彙總函數。)
 - 具有的任何彙總函數 DISTINCT
 - 任何範圍函數

- SELECT DISTINCT 或 HAVING子句
- 外部資料表
- 其他具體化視觀表
- 自動查詢重寫會重寫參考使用者定義 Amazon Redshift 資料表的SELECT查詢。Amazon Redshift 不會重寫以下查詢：
 - CREATE TABLE AS 陳述式
 - SELECT INTO 陳述式
 - 目錄或系統資料表的查詢
 - 具有外部聯結或SELECTDISTINCT子句的查詢
- 如果查詢未自動重寫，請檢查您是否具有指定具體化檢視的SELECT許可，且[mv_enable_aqmv_for_session](#)選項設定為 TRUE。

您也可以檢查 STV_MV_，檢查具體化檢視是否符合自動重寫查詢的資格INFO。如需詳細資訊，請參閱[STV_MV_INFO](#)。

重新整理具體化視觀表

本主題說明如何從基礎資料表重新整理材料檢視中的資料。

當您建立具體化視觀表時，其內容會反映當時基礎資料庫資料表或資料表的狀態。即使應用程式變更基底資料表中的資料，具體化視觀表中的資料仍會維持不變。若更新具體化視觀表中的資料，您可以隨時使用 REFRESH MATERIALIZED VIEW 陳述式來手動重新整理具體化視觀表。當您執行此陳述式時，Amazon Redshift 會識別在基底資料表或資料表中發生的變更，並將這些變更套用到具體化視觀表。

Amazon Redshift 有兩種策略可重新整理具體化視觀表：

- 在許多案例中，Amazon Redshift 可以執行累加式重新整理。在累加式重新整理中，Amazon Redshift 會快速識別自上次重新整理之後，基礎資料表中資料的變更，並更新具體化視觀表中的資料。定義具體化視觀表時，查詢中使用的下列SQL建構支援增量重新整理：
 - 包含子句 SELECT、FROM、【INNER】 JOIN、WHERE、GROUP BY 或 的建構HAVING。
 - 包含彙總的，例如 SUM、MIN、MAX、AVG和 COUNT。
 - 大多數內建SQL函數，特別是不可變的函數，因為這些函數具有相同的輸入引數，並且始終產生相同的輸出。

以資料共用資料表為基礎的具體化視觀表也支援累加式重新整理。

- 如果無法進行累加式更新，則 Amazon Redshift 會執行完全的重新整理。完整重新整理會重新執行基礎SQL陳述式，取代具體化檢視中的所有資料。
- Amazon Redshift 會根據用於定義具體化視觀表的SELECT查詢，自動為具體化視觀表選擇重新整理方法。

重新整理具體化視觀表上的具體化視觀表並不是串聯處理。換句話說，假設您有一個具體化視觀表 A 取決於具體化視觀表 B。在此情況下，調用 REFRESH MATERIALIZED VIEW A 時，即使 B 是，也會使用目前版本的 B 重新整理 A out-of-date。要使 A 完全保持最新狀態，請在重新整理 A 之前，先在單獨的交易中重新整理 B。

以下範例顯示如何以程式設計方式，為具體化視觀表建立完整重新整理計劃。若要重新整理具體化視觀表 v，請先重新整理具體化視觀表 u 若要重新整理具體化視觀表 v，請先重新整理具體化視觀表 u，然後再重新整理具體化視觀表 v。

```
CREATE TABLE t(a INT);
CREATE MATERIALIZED VIEW u AS SELECT * FROM t;
CREATE MATERIALIZED VIEW v AS SELECT * FROM u;
CREATE MATERIALIZED VIEW w AS SELECT * FROM v;

WITH RECURSIVE recursive_deps (mv_tgt, lvl, mv_dep) AS
( SELECT trim(name) as mv_tgt, 0 as lvl, trim(ref_name) as mv_dep
  FROM stv_mv_deps
  UNION ALL
  SELECT R.mv_tgt, R.lvl+1 as lvl, trim(S.ref_name) as mv_dep
  FROM stv_mv_deps S, recursive_deps R
  WHERE R.mv_dep = S.name
)

SELECT mv_tgt, mv_dep from recursive_deps
ORDER BY mv_tgt, lvl DESC;

mv_tgt | mv_dep
-----+-----
v      | u
w      | u
w      | v
(3 rows)
```

當您REFRESHMATERIALIZEDVIEW在視具體化視觀表而定 out-of-date的具體化視觀表上執行時，下列範例會顯示資訊性訊息。

```
create table a(a int);
```

```
create materialized view b as select * from a;
```

```
create materialized view c as select * from b;
```

```
insert into a values (1);
```

```
refresh materialized view c;
```

INFO: Materialized view c is already up to date. However, it depends on another materialized view that is not up to date.

```
REFRESH MATERIALIZED VIEW b;
```

INFO: Materialized view b was incrementally updated successfully.


```
REFRESH MATERIALIZED VIEW c;
```

INFO: Materialized view c was incrementally updated successfully.

Amazon Redshift 目前對於具體化視觀表的累加式重新整理有下列限制。

Amazon Redshift 不支援使用下列SQL元素透過查詢定義的具體化檢視增量重新整理：

- OUTER JOIN (RIGHT、 LEFT或 FULL)。
- 設定操作 UNION、 INTERSECT、 EXCEPT和 MINUS。
- 彙總函數
MEDIAN、 PERCENTILE_CONT、 LISTAGG、 STDDEV_SAMP、 STDDEV_POP、 APPROXIMATECOUNT、 APPROXIMATE 和 位元彙總函數。

 Note

支援 COUNT、 SUM和 AVG彙總函數。

- DISTINCT 彙總函數，例如 DISTINCT SUM、 COUNT DISTINCT 等。
- 視窗函數。
- 使用暫存資料表進行查詢最佳化的查詢，例如最佳化通用子運算式。

- 子查詢。
- 在定義具體化視觀表的查詢中，參照下列查詢中的格式的外部資料表。
 - Delta Lake
 - Hudi

對於使用上述格式以外的格式定義的具體化視觀表，預覽軌道上支援累加式重新整理。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的[建立預覽叢集](#)。如需設定預覽工作群組的詳細資訊，請參閱 Amazon Redshift 管理指南中的[建立預覽工作群組](#)。

自動重新整理具體化視觀表

使用 up-to-date 或 修改為具有自動重新整理選項時，Amazon Redshift 可以使用其基礎資料表中的資料自動重新整理具體化檢視。Amazon Redshift 會在基本資料表變更後盡快自動重新整理具體化視觀表。

為了在盡量不影響叢集內作用中工作負載的情況下完成重新整理最重要的具體化視觀表，Amazon Redshift 會考慮多個因素。這些因素包括目前的系統負載、重新整理所需的資源、可用的叢集資源，以及使用具體化視觀表的頻率。

Amazon Redshift 會優先考慮您的工作負載而非自動重新整理，並且可以停止自動重新整理以保留使用者工作負載的效能。此方法可能會延遲重新整理部分具體化視觀表。在某些情況下，您的具體化視觀表可能需要更具決定性的重新整理行為。若是如此，請考慮如 所述使用手動重新整理[REFRESH MATERIALIZED VIEW](#)，或使用 Amazon Redshift 排程器API操作或主控台進行排程重新整理。

您可以使用 CREATE MATERIALIZED 設定具體化檢視的自動重新整理VIEW。您也可以使用 AUTOREFRESH子句自動重新整理具體化檢視。如需建立具體化視觀表的相關資訊，請參閱[CREATE MATERIALIZED VIEW](#)。您可以使用 [ALTER MATERIALIZED VIEW](#) 來開啟目前具體化視觀表的自動重新整理。

重新整理具體化視觀表時應考慮以下項目：

- 即使您尚未為具體化檢視啟用自動重新整理，您仍然可以使用REFRESHMATERIALIZEDVIEW命令明確重新整理具體化檢視。
- Amazon Redshift 不會自動重新整理外部資料表上定義的具體化視觀表。
- 對於重新整理狀態，您可以檢查 SVL_MV_REFRESH_STATUS，其中會記錄使用者啟動或自動重新整理的查詢。
- 若要在重新運算限定具體化視觀表REFRESH上執行，請確定您有結構描述的CREATE許可。如需詳細資訊，請參閱[GRANT](#)。

自動具體化視觀表

本主題說明 Amazon Redshift 如何使用自動化具體化視觀表來改善效能。Amazon Redshift 會根據資料庫活動和效能自動建立具體化檢視。根據預設，Amazon Redshift 會使用自動化具體化檢視。

具體化視觀表是改善 Amazon Redshift 中查詢效能的強大工具。其透過儲存預先計算的結果集來做到這一點。類似的查詢不必每次都重新執行相同的邏輯，因為它們可以從現有的結果集擷取記錄。開發人員和分析師會在分析工作負載之後建立具體化視觀表，以判斷哪些查詢可以受益，以及每個具體化視觀表的維護成本是否值得。隨著工作負載的增長或變化，這些具體化視觀表必須經過審查，以確保它們繼續提供實際的效能優勢。

Redshift 中的自動具體化視觀表 (AutoMV) 功能提供與使用者建立的具體化視觀表相同的效能優勢。Amazon Redshift 會使用機器學習持續監控工作負載，並在有益的情況下建立新的具體化視觀表。AutoMV 會平衡建立和維持具體化視觀表最新狀態的成本，以及查詢延遲的預期效益。系統也會監控先前建立的 A，utoMVs 並在它們不再受益時將其捨棄。

AutoMV 行為和功能與使用者建立的具體化視觀表相同。其使用相同的準則和限制以自動且累加的方式進行重新整理。就像使用者建立的具體化檢視一樣，[自動查詢重寫以使用具體化視檢視](#)識別可以從系統建立的 A 中受益的查詢utoMVs。它會自動重寫這些查詢，以使用 A utoMVs，改善查詢效能。開發人員不需要修改查詢即可利用 AutoMV。

Note

自動具體化視觀表會間歇性重新整理。重寫為使用 AutoMV 的查詢永遠會傳回最新的結果。當 Redshift 偵測到資料不是最新的時候，就不會將查詢重寫為從自動具體化視觀表讀取。相反地，查詢會從基底資料表中選取最新資料。

任何具有重複使用之查詢的工作負載都可以受益於 AutoMV。常用案例包括：

- 儀表板 - 儀表板廣泛用於提供關鍵業務指標 (KPIs)、事件、趨勢和其他指標的快速檢視。它們通常具有包含圖表和資料表的通用版面配置，但會顯示不同的檢視來用於篩選或維度選取操作 (例如向下鑽研)。儀表板通常有一組共同的查詢，以不同的參數重複使用。儀表板查詢可以從自動具體化視觀表中獲益匪淺。
- 報告 - 報告查詢可以安排在不同的頻率下進行，根據業務需求和報告的類型。此外，它們可以是自動或隨需形式。報告查詢的一個共同特徵是可以長時間執行且屬於資源密集型。使用 AutoMV 時，這些查詢不需要在每次執行時重新計算，這會減少 Redshift 中每個查詢的執行期和資源使用率。

若要關閉自動具體化視觀表，請將 `auto_mv` 參數群組更新為 `false`。如需詳細資訊，請參閱《Amazon Redshift 叢集管理指南》中的 [Amazon Redshift 參數群組](#)。

SQL 自動化具體化視觀表的範圍和考量

- 自動具體化檢視可由查詢或子查詢啟動和建立，只要它包含子GROUP BY句或下列彙總函數之一：SUM、MIN、COUNTMAX或AVG。但不能包含以下任何項目：
 - 左側、右側或完整外部聯結
 - 彙總SUM、COUNT、MINMAX和以外的函數AVG。(這些特定函式可與自動查詢重寫搭配使用。)
 - 任何彙總函數，包括DISTINCT
 - 任何範圍函數
 - SELECT DISTINCT 或 HAVING子句
 - 其他具體化視觀表

不保證符合準則的查詢會啟動自動具體化視觀表的建立。系統會根據其對工作負載的預期效益和要維護的資源成本(包括系統要重新整理的成本)，決定要從哪些候選項目建立檢視。每個產生的具體化視觀表都可透過自動查詢重寫來使用。

- 即使 AutoMV 可能是由子查詢或集合運算子的個別支段啟動，產生的具體化視觀表也不會包含子查詢或集合運算子。
- 若要判斷 AutoMV 是否用於查詢，請檢視EXPLAIN計劃並在輸出%_auto_mv_%中尋找。如需詳細資訊，請參閱 [EXPLAIN](#)。
- 外部資料表(例如資料共用和聯合資料表)不支援自動具體化視觀表。

自動具體化視觀表限制

下列是使用自動具體化視觀表的限制：

- A 的最大數目utoMVs - 叢集中每個資料庫的自動化具體化視觀表限制為 200。
- 儲存空間和容量 - AutoMV 的一個重要特點是，它是使用備用背景週期來執行，以協助實現使用者工作負載不受影響的目的。如果叢集忙碌或儲存空間不足，AutoMV 會停止其活動。具體而言，在叢集總容量的 80% 時，不會建立新的自動具體化視觀表。在總容量的 90% 時，它們可能會被捨棄，以在不降低效能的情況下繼續使用者工作負載。如需決定叢集容量的相關資訊，請參閱 [STV_NODE_STORAGE_CAPACITY](#)。

自動具體化視觀表的計費

Amazon Redshift 的自動最佳化功能可建立和重新整理自動具體化視觀表。此程序的運算資源無須付費。自動具體化視觀表的儲存費用是以一般儲存費率計費。如需詳細資訊，請參閱 [Amazon Redshift 定價](#)。

其他資源

下列部落格文章提供有關自動具體化視觀表的進一步說明 其中詳細介紹了其建立、維護和捨棄方式。還說明了推動這些決策的基礎演算法：[使用自動具體化視觀表最佳化 Amazon Redshift 查詢效能](#)。

此影片從具體化視觀表的說明開始，並示範其如何改善效能及節省資源。然後，透過程序流程動畫和現場示範，提供自動具體化視觀表的深入說明。

在具體化視觀表中使用使用者定義的函數（UDF）

您可以在 Amazon Redshift 具體化檢視UDF中使用純量。在 python 中定義這些項目，或在具體化檢視定義中SQL參考它們。

在具體化視觀表UDF中參考

下列程序顯示在具體化視觀定義中如何使用 UDFs 來執行簡單的算術比較。

1. 建立要在具體化視觀表定義中使用的資料表。

```
CREATE TABLE base_table (a int, b int);
```

2. 在 python 中建立一個純量使用者定義函數，該函數會傳回布林值，指出整數是否大於比較整數。

```
CREATE OR REPLACE FUNCTION udf_python_bool(x1 int, x2 int) RETURNS bool IMMUTABLE
AS $$
    return x1 > x2
$$ LANGUAGE plpythonu;
```

或者，UDF使用 建立功能相似的 SQL，您可以使用它來比較結果與第一個結果。

```
CREATE OR REPLACE FUNCTION udf_sql_bool(int, int) RETURNS bool IMMUTABLE
AS $$
    select $1 > $2;
```

```
$$ LANGUAGE SQL;
```

3. 建立具體化檢視，從您建立的資料表中選取 並參考 UDF。

```
CREATE MATERIALIZED VIEW mv_python_udf AS SELECT udf_python_bool(a, b) AS a FROM  
base_table;
```

或者，您可以建立參考 SQL 的具體化檢視UDF。

```
CREATE MATERIALIZED VIEW mv_sql_udf AS SELECT udf_sql_bool(a, b) AS a FROM  
base_table;
```

4. 新增資料至資料表，然後重新整理具體化視觀表。

```
INSERT INTO base_table VALUES (1,2), (1,3), (4,2);
```

```
REFRESH MATERIALIZED VIEW mv_python_udf;
```

或者，您可以重新整理參考 SQL 的具體化檢視UDF。

```
REFRESH MATERIALIZED VIEW mv_sql_udf;
```

5. 查詢具體化視觀表的資料。

```
SELECT * FROM mv_python_udf ORDER BY a;
```

查詢結果如下：

```
a  
----  
false  
false  
true
```

這會針對最後一組值傳回 true，因為資料欄 a (4) 的值大於資料欄 b (2) 的值。

6. 或者，您可以查詢參考 SQL 的具體化檢視UDF。SQL 函數的結果與 Python 版本的結果相符。

```
SELECT * FROM mv_sql_udf ORDER BY a;
```


查詢結果如下：

```
a
-----
false
false
true
```

這將針對最後一組值傳回 true 以進行比較。

7. 使用 DROP 陳述式 CASCADE 來捨棄使用者定義的函數和參考該函數的具體化檢視。

```
DROP FUNCTION udf_python_bool(int, int) CASCADE;
```

```
DROP FUNCTION udf_sql_bool(int, int) CASCADE;
```

將擷取串流至具體化視觀表

本主題說明如何使用具體化檢視來快速存取串流資料。

串流擷取提供從 [Amazon Kinesis Data Streams](#) 或 [Amazon Managed Streaming for Apache Kafka](#) 到 Amazon Redshift 佈建資料庫或 Amazon Redshift Serverless 資料庫的低延遲、高速資料擷取。資料會落在為目的設定的 Redshift 具體化檢視中。這會導致快速存取外部資料。串流擷取可縮短資料存取時間並降低儲存成本。您可以使用少量的 SQL 命令集，為 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組設定它。設定後，每個具體化視觀表重新整理都可以每秒擷取數百 MB 的資料。

資料如何從串流服務流向 Redshift

它有助於了解串流擷取的運作方式，以及流程中使用的資料庫物件。資料直接從資料串流提供者流向 Amazon Redshift 佈建叢集或 Amazon Redshift Serverless 工作群組。沒有臨時登陸區域，例如 Amazon S3 儲存貯體。佈建的叢集或工作群組是串流取用者。在 Redshift 資料庫中，從串流讀取的資料會落在具體化視觀表中。資料會在到達時進行處理。例如，可以使用取用 JSON 值，並映射到具體化視觀表的資料欄 SQL。重新整理具體化檢視時，Redshift 會耗用來自己配置 Kinesis 資料碎片或 Kafka 分割區的資料，直到檢視與串流保持最新狀態為止。

Amazon Redshift 串流擷取的使用案例涉及持續產生且必須在短時間內處理的資料，或從原始伺服器延遲。這通常稱為近乎即時的分析。來源可以包括 IT 裝置、系統遙測裝置，以及來自忙碌網站或應用程式的點擊串流資料。

改善效能的資料剖析最佳實務

當您設定串流擷取時，剖析傳入資料的方式有一些選項。做法可能包括在資料到達時執行業務邏輯或格式設定。我們建議採用下列最佳實務，以避免錯誤或資料遺失。這些衍生自內部測試，並協助客戶進行故障診斷組態和剖析問題。

- 從串流資料中擷取值 – 如果您在具體化檢視定義中使用 [JSON_EXTRACT_PATH_TEXT](#) 函數來剖析或分割串流 JSON，可能會顯著影響效能和延遲。若要解釋，使用 `JSON_EXTRACT_PATH` 擷取的每個資料欄TEXT都會JSON重新剖析傳入的資料欄。之後，會進行資料類型轉換、篩選和商業邏輯計算。例如，這表示如果您從JSON資料中擷取 10 個資料欄，則每個JSON記錄會剖析 10 次，其中包括其他邏輯。這會導致更高的擷取延遲。我們建議的替代方法是使用 [JSON_PARSE](#) 函數將JSON記錄轉換為 Redshift 的SUPER資料類型。串流資料登陸具體化視觀表後，請使用 PartiQL 從JSON資料SUPER表示法中擷取個別字串。如需詳細資訊，請參閱[查詢半結構化資料](#)。

此外，請注意，`JSON_EXTRACT_PATH_TEXT` 具有 64KB 的資料大小上限。因此，如果任何JSON記錄大於 64KB，使用 `JSON_EXTRACT_PATH_TEXT` 處理它會導致錯誤。


- 將 Amazon Kinesis Data Streams 串流或 Amazon MSK主題映射至多個具體化檢視 – 我們不建議建立多個具體化檢視，以從單一串流或主題擷取資料。這是因為每個具體化檢視都會為 Kafka 主題中的 Kinesis Data Streams 串流或分割區中的每個碎片建立取用者。這可能會導致限流或超過串流或主題的輸送量。它也可能導致成本提高，因為您多次擷取相同的資料。當您設定串流擷取時，建議您為每個串流或主題建立一個具體化檢視。

如果您的使用案例需要您將資料從一個KDS串流或MSK主題擷取到多個具體化視觀表，請先參閱[AWS 大數據部落格](#)，特別是[使用 Amazon Redshift Streaming Ingestion 與 Amazon 實作 near-real-time分析的最佳實務MSK](#)，然後再執行此操作。

串流擷取行為和資料類型

下表說明各種資料類型的技術行為詳細資訊和大小限制。建議您先熟悉這些內容，再設定具體化檢視以進行串流擷取。

功能或行為	描述
Kafka 主題長度限制	Kafka 主題的名稱不得超過 128 個字元 *不包括引號)。如需詳細資訊，請參閱 名稱和識別碼 。
增量重新整理和具體化視觀表 JOINS上的	具體化視觀表必須是可以增量維護的。根據預設，Kinesis 或 Amazon 無法進行完整重新計算，MSK因為它們不會保留超過

功能或行為	描述
	<p>24 小時或 7 天的串流或主題歷史記錄。您可以在 Kinesis 或 Amazon MSK 中設定更長的資料保留期。但是，這可能會導致更多的維護和成本。此外，目前 JOINS 不支援在 Kinesis 串流或 Amazon MSK 主題上建立的具體化檢視。在串流或主題上建立具體化視觀表之後，您可以建立另一個具體化視觀表，將串流具體化視觀表結合至其他具體化視觀表、資料表或檢視。</p> <p>如需詳細資訊，請參閱 REFRESH MATERIALIZED VIEW。</p>
記錄剖析	<p>Amazon Redshift 串流擷取不支援由 Kinesis Producer Library 彙總的剖析記錄 (KPL 金鑰概念 - 彙總)。彙總記錄會被擷取，但會儲存為二進制協定緩衝區資料。(如需詳細資訊，請參閱 協定緩衝區)。視您將資料推送至 Kinesis 的方式而定，您可能需要關閉此功能。</p>
解壓縮	<p>VARBYTE 不支援解壓縮。因此，包含壓縮資料的記錄無法在 Redshift 中查詢。在將資料新增至 Kinesis 串流或 Amazon MSK 主題之前，請先解壓縮您的資料。</p>
記錄大小上限	<p>Amazon Redshift 可以從 Kinesis 或 Amazon MSK 擷取的任何記錄大小上限為 16,777,216 位元組 (16 MiB)，這是 Amazon Redshift 中 VARBYTE 資料類型支援的大小上限。根據預設，在 Kinesis 資料串流或 Amazon MSK 主題上建立的 Amazon Redshift 串流具體化檢視，會將資料欄的大小分別設定為 1,048,576 位元組 (1 MiB) 和 16,777,216 位元組 (16 MiB)。</p> <div data-bbox="594 1419 1507 1738" style="border: 1px solid #add8e6; border-radius: 15px; padding: 10px; margin-top: 10px;"> <p> Note</p> <p>1MiB 是可放入 Kinesis 資料串流的所有記錄的目前大小上限。如需 Kinesis 大小限制的詳細資訊，請前往 Amazon Kinesis Data Streams 開發人員指南中的 配額和限制。</p> </div>

功能或行為	描述
錯誤記錄	<p>在無法擷取記錄給 Redshift 的每個情況下，因為資料的大小超過上限，則會略過該記錄。在這種情況下，具體化視觀表重新整理仍然會成功，並且每個錯誤記錄的區段都會寫入 SYS_STREAM_SCAN_ERRORS 系統表。但不會略過商務邏輯所產生的錯誤，例如計算中的錯誤或類型轉換所產生的錯誤。在將邏輯新增至具體化檢視定義之前，請仔細測試邏輯。</p>
Amazon MSK VPC多私有連線	<p>Redshift 串流擷取目前不支援 Amazon MSK 多VPC私有連線。或者，您可以使用 VPC對等連線 透過中央集線器連線 VPCs 或 AWS Transit Gateway 連線 VPCs 和內部部署網路。這兩種都可以讓 Redshift 與 Amazon MSK 叢集或另一個中的 Amazon MSK Serverless 通訊 VPC。</p>
自動重新整理用量和啟用	<p>具體化視觀表或視觀表的自動重新整理查詢會被視為任何其他使用者工作負載。自動重新整理會在資料到達時從串流載入資料。</p> <p>您可以針對為串流擷取建立的具體化視觀表明確開啟自動重新整理。若要這樣做，請在具體化視觀表定義中指定 AUTO REFRESH。預設為手動重新整理。若要為現有具體化視觀表指定自動重新整理以進行串流擷取，您可以執行 ALTER MATERIALIZED VIEW 以將其開啟。如需詳細資訊，請參閱 CREATE MATERIALIZED VIEW 或 ALTER MATERIALIZED VIEW。</p>
串流擷取和 Amazon Redshift Serverless	<p>適用於佈建叢集上 Amazon Redshift 串流擷取的設定和組態指示，也適用於 Amazon Redshift Serverless 上的串流擷取。請務必指定必要的層級 RPU，以支援具有自動重新整理和其他工作負載的串流擷取。如需詳細資訊，請參閱 Amazon Redshift Serverless 的計費。</p>
Amazon Redshift 節點位於與 Amazon MSK 叢集不同的可用區域中	<p>當您設定串流擷取時，如果啟用 Amazon 的機架感知，Amazon Redshift 會嘗試在相同的可用性 Amazon MSK 叢集中連線至 Amazon 叢集 MSK。如果所有節點都位於與 Amazon Redshift 叢集不同的可用區域，則可能會產生跨可用區域資料傳輸成本。若要避免這種情況，請在與 Redshift 佈建叢集或工作群組相同的 AZ 中至少保留一個 Amazon MSK 代理程式叢集節點。</p>

功能或行為	描述
重新整理開始位置	建立具體化檢視後，其初始重新整理會從 Kinesis 串流 TRIM_HORIZON 的開始，或從 Amazon MSK 主題的偏移 0 開始。
資料格式	支援的資料格式僅限於可以從轉換的資料格式 VARBYTE。如需詳細資訊，請參閱 VARBYTE 類型 和 VARBYTE 運算子 。
將記錄附加至資料表	您可以執行 ALTER TABLE APPEND，從現有的來源具體化檢視將資料列附加至目標資料表。這只有在具體化視觀表設定為串流擷取時才有作用。如需詳細資訊，請參閱 ALTER TABLE APPEND 。
執行中 TRUNCATE 或 DELETE	<p>您可以使用下列方式，從用於串流擷取的具體化檢視中移除記錄：</p> <ul style="list-style-type: none"> • TRUNCATE – 這會從已設定為串流擷取的具體化檢視中刪除所有資料列。其不會執行資料表掃描。如需詳細資訊，請參閱 TRUNCATE。 • DELETE – 這會從設定串流擷取的具體化檢視中刪除所有資料列。如需詳細資訊，請參閱 DELETE。

開始使用 Amazon Kinesis Data Streams 中的串流擷取

本主題說明如何使用具體化視觀表從 Kinesis Data Streams 取用串流資料。

設定 Amazon Redshift 串流擷取涉及建立對應至串流資料來源的外部結構描述，以及建立參照外部結構描述的具體化視觀表。Amazon Redshift 串流擷取支援以 Kinesis Data Streams 作為來源。因此，在設定串流擷取之前，您必須擁有可用的 Kinesis Data Streams 來源。如果您沒有來源，請按照 [Amazon Kinesis Data Streams 入門](#) 中的 Kinesis 文件指示進行操作，或使用 [透過 AWS 管理主控台建立串流](#) 中的指示，在主控台上建立一個來源。

Amazon Redshift 串流擷取會使用具體化視觀表，執行 REFRESH 時會直接從串流更新該具體化視觀表。具體化視觀表會對應至串流資料來源。您可以對串流資料執行篩選和彙總，以做為具體化視觀表定義的一部分。您的串流擷取具體化視觀表 (基礎具體化視觀表) 只能參照一個串流，但是您可以建立與基礎具體化視觀表及其他具體化視觀表或資料表結合的額外具體化視觀表。

Note

串流擷取和 Amazon Redshift Serverless - 本主題中的組態步驟同時適用於佈建的 Amazon Redshift 叢集和 Amazon Redshift Serverless。如需詳細資訊，請參閱[串流擷取行為和資料類型](#)。

假設您有可用的 Kinesis Data Streams 串流，第一步是使用 CREATE EXTERNAL SCHEMA 在 Amazon Redshift 中定義結構描述，並參照 Kinesis Data Streams 資源。之後，若要存取串流中的資料，請在具體化視觀表中定義 STREAM。您可以使用半結構化 SUPER 格式儲存串流記錄，或定義結構描述，將資料轉換為 Redshift 資料類型。當您查詢具體化檢視時，傳回的記錄是 point-in-time 串流的檢視。

1. 使用信任政策建立 IAM 角色，讓您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組擔任該角色。如需如何設定 IAM 角色信任政策的相關資訊，請參閱[授權 Amazon Redshift 代表您存取其他 AWS 服務](#)。建立後，角色應具有下列 IAM 政策，提供與 Amazon Kinesis 資料串流通訊的許可。

IAM Kinesis Data Streams 中未加密串流的政策

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadStream",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamSummary",
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:ListShards",
        "kinesis:DescribeStream"
      ],
      "Resource": "arn:aws:kinesis:*:0123456789:stream/*"
    },
    {
      "Sid": "ListStream",
      "Effect": "Allow",
      "Action": "kinesis:ListStreams",
      "Resource": "*"
    }
  ]
}
```

```
}

```

IAM Kinesis Data Streams 中加密串流的政策

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "ReadStream",
    "Effect": "Allow",
    "Action": [
      "kinesis:DescribeStreamSummary",
      "kinesis:GetShardIterator",
      "kinesis:GetRecords",
      "kinesis:ListShards",
      "kinesis:DescribeStream"
    ],
    "Resource": "arn:aws:kinesis:*:0123456789:stream/*"
  },
  {
    "Sid": "DecryptStream",
    "Effect": "Allow",
    "Action": [
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:0123456789:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  },
  {
    "Sid": "ListStream",
    "Effect": "Allow",
    "Action": "kinesis:ListStreams",
    "Resource": "*"
  }
  ]
}
```

2. 檢查您的 VPC ，並驗證您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 是否有使用 NAT 閘道或網際網路閘道透過網際網路存取 Kinesis Data Streams 端點的路由。如果您希望 Redshift 和 Kinesis Data Streams 之間的流量保留在 AWS 網路中，請考慮使用 Kinesis Interface VPC Endpoint。如需詳細資訊，請參閱[使用 Amazon Kinesis Data Streams Kinesis Data Streams with Interface VPC Endpoints](#)。
3. 在 Amazon Redshift 中，建立外部結構描述以將 Kinesis 中的資料對應至結構描述。

```
CREATE EXTERNAL SCHEMA kds
FROM KINESIS
IAM_ROLE { default | 'iam-role-arn' };
```

Kinesis Data Streams 的串流擷取不需要驗證類型。它使用 CREATE EXTERNAL SCHEMA 陳述式中定義的 IAM 角色來發出 Kinesis Data Streams 請求。

選用：使用 REGION 關鍵字來指定 Amazon Kinesis Data Streams 或 Amazon MSK 串流所在的區域。

```
CREATE EXTERNAL SCHEMA kds
FROM KINESIS
REGION 'us-west-2'
IAM_ROLE { default | 'iam-role-arn' };
```

在此範例中，該區域會指定來源串流的位置。IAM_ROLE 是範例。

4. 建立具體化視觀表以取用串流資料。使用如下的陳述式，如果無法剖析記錄，則會導致錯誤。如果您不希望略過錯誤記錄，請使用像這樣的命令。

```
CREATE MATERIALIZED VIEW my_view AUTO REFRESH YES AS
SELECT *
FROM kds.my_stream_name;
```

Kinesis 串流名稱會區分大小寫，且可同時包含大寫和小寫字母。若要從具有大寫名稱的串流擷取，您可以在 `enable_case_sensitive_identifier` 資料庫層級將組態設定為 `TRUE`。如需詳細資訊，請參閱 [名稱和識別碼](#) 與 [enable_case_sensitive_identifier](#)。

若要開啟自動重新整理，請使用 `AUTO REFRESH YES`。預設行為是手動重新整理。請注意，當您使用 `CAN_JSON_PARSE` 時，可能會略過無法剖析的記錄。

中繼資料資料欄包括下列項目：

中繼資料資料欄	資料類型	描述
<code>approximate_arrival_timestamp</code>	沒有時區的時間戳記	將記錄插入 Kinesis 串流的大約時間

中繼資料資料欄	資料類型	描述
partition_key	varchar(256)	Kinesis 用來將記錄指派給碎片的索引鍵
shard_id	char(20)	從中擷取記錄的串流內的碎片唯一識別碼
sequence_number	varchar(128)	Kinesis 碎片中記錄的唯一識別碼
refresh_time	沒有時區的時間戳記	重新整理的開始時間
kinesis_data	varbyte	Kinesis 串流的記錄

請務必注意，如果您的具體化視觀定義中有業務邏輯，則在某些情況下，業務邏輯錯誤可能會導致串流擷取遭到封鎖。這可能會導致您必須捨棄並重新建立具體化視觀表。為了避免這種情況，我們建議您盡可能簡單地使用邏輯，並在擷取資料後對資料執行大部分的商業邏輯檢查。

- 重新整理檢視，這會調用 Redshift 從串流讀取，並將資料載入具體化視觀表。

```
REFRESH MATERIALIZED VIEW my_view;
```

- 查詢具體化視觀表中的資料。

```
select * from my_view;
```

從 Amazon Managed Streaming for Apache Kafka (Amazon MSK) 開始串流擷取

本主題說明如何MSK使用具體化檢視從 Amazon 取用串流資料。

Amazon Redshift 串流擷取的目的是簡化直接從串流服務擷取串流資料至 Amazon Redshift 或 Amazon Redshift Serverless 的程序。這適用於 Amazon MSK Provisioned 和 Amazon MSK Serverless，以及 Kinesis Data Streams。Amazon Redshift 串流擷取不需要在 Amazon S3 中暫存 Kinesis Data Streams 串流或 Amazon MSK主題，然後再將串流資料擷取至 Redshift。

在技術層面上，從 Amazon Kinesis Data Streams 和 Amazon 串流擷取MSK，提供低延遲、高速的串流或主題資料擷取到 Amazon Redshift 具體化視觀表。在設定之後，您可以使用具體化視觀表重新整理採用大量資料。

MSK 執行下列步驟，為 Amazon 設定 Amazon Redshift 串流擷取：

1. 建立對應至串流資料來源的外部結構描述。
2. 建立參照外部結構描述的具體化視觀表。

在設定 Amazon Redshift 串流擷取之前，您必須有可用的 Amazon MSK來源。如果您沒有來源，請遵循[開始使用 Amazon MSK](#)中的指示。

Note

串流擷取和 Amazon Redshift Serverless - 本主題中的組態步驟同時適用於佈建的 Amazon Redshift 叢集和 Amazon Redshift Serverless。如需詳細資訊，請參閱[串流擷取行為和資料類型](#)。

從 Kafka 設定IAM許可和執行串流擷取

假設您有可用的 Amazon MSK叢集，第一步是使用 定義 Redshift 中的結構描述，CREATE EXTERNAL SCHEMA並將 Amazon MSK叢集作為資料來源參考。之後，若要存取主題中的資料，請在具體化視觀表中定義 STREAM。您可以使用預設的 Amazon Redshift VARBYTE資料類型來儲存主題的記錄，或定義將資料轉換為半結構化SUPER格式的結構描述。當您查詢具體化檢視時，傳回的記錄是 point-in-time主題的檢視。

1. 如果您使用 AUTHENTICATION NONE 連線到 MSK，則不需要IAM任何角色。不過，如果您使用 AUTHENTICATIONIAM或 MTLS 來驗證 Amazon MSK叢集，您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 命名空間必須具有具有適當許可的連接IAM角色。使用信任政策建立IAM角色，讓您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 命名空間擔任該角色。建立角色後，請新增下列其中一個許可以支援 IAM或 MTLS。對於 mTLS 身分驗證，Amazon Redshift 使用的憑證可以存放在 或 中 AWS Certificate Manager AWS Secrets Manager，因此您必須選擇符合憑證存放位置的政策。將角色連接至您的 Amazon Redshift 佈建叢集或 Redshift Serverless 命名空間。如需有關如何設定IAM角色信任政策的資訊，請參閱[授權 Amazon Redshift 代表您存取其他 AWS 服務](#)。

AUTHENTICATION IAM:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MSKIAMpolicy",
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:ReadData",
        "kafka-cluster:DescribeTopic",
        "kafka-cluster:Connect"
      ],
      "Resource": [
        "arn:aws:kafka:*:0123456789:cluster/MyTestCluster/*",
        "arn:aws:kafka:*:0123456789:topic/MyTestCluster/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "kafka-cluster:AlterGroup",
        "kafka-cluster:DescribeGroup"
      ],
      "Resource": [
        "arn:aws:kafka:*:0123456789:group/MyTestCluster/*"
      ]
    }
  ]
}
```

AUTHENTICATION MTLS : 使用存放在 中的憑證 AWS Certificate Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MSKmTLSACMpolicy",
      "Effect": "Allow",
      "Action": [
        "acm:ExportCertificate"
      ],
      "Resource": [
        "arn:aws:acm:us-east-1:444455556666:certificate/certificate_ID"
      ]
    }
  ]
}
```

```

    ]
  }
]
}

```

AUTHENTICATION MTLS：使用存放在 中的憑證 AWS Secrets Manager

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "MSKmtlsSecretsManagerpolicy",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Resource": [
        "arn:aws:secretsmanager:us-east-1:444455556666:secret:secret_ID"
      ]
    }
  ]
}


```

2. 檢查您的 VPC，並驗證您的 Amazon Redshift 叢集或 Amazon Redshift Serverless 是否有前往 Amazon MSK 叢集的路由。Amazon MSK 叢集的傳入安全群組規則應允許 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組的安全群組。您指定的連接埠取決於 Amazon MSK 叢集上設定的身分驗證方法。如需詳細資訊，請參閱 [連接埠資訊和從內 AWS 外存取 VPC](#)

下表顯示要設定用於從 Amazon 串流擷取的免費組態選項 MSK：

Amazon Redshift 組態	Amazon MSK 組態	在 Redshift 和 Amazon 之間開啟的連接埠 MSK
AUTHENTICATION NONE	TLS 傳輸已停用	9092
AUTHENTICATION NONE	TLS 傳輸已啟用	9094
AUTHENTICATION IAM	IAM	9098/9198
AUTHENTICATION MTLS	TLS 傳輸已啟用	9094

Amazon Redshift 身分驗證是在 CREATEEXTERNALSCHEMA 陳述式中設定。

 Note

在 Amazon MSK 叢集已啟用互通傳輸層安全 (mTLS) 身分驗證的情況下，將 Amazon Redshift 設定為使用 AUTHENTICATIONNONE 指示其使用連接埠 9094 進行未驗證的存取。不過，這會失敗，因為連接埠正由 mTLS 身分驗證使用中。因此，我們建議您在使用 AUTHENTICATION m 時切換至 mTLS。

3. 在 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組上啟用增強型 VPC 路由。如需詳細資訊，請參閱 [啟用增強型 VPC 路由](#)。
4. 在 Amazon Redshift 中，建立外部結構描述以映射至 Amazon MSK 叢集。語法如下：

```
CREATE EXTERNAL SCHEMA MySchema
FROM MSK
[ IAM_ROLE [ default | 'iam-role-arn' ] ]
AUTHENTICATION [ none | iam | mTLS ]
[AUTHENTICATION_ARN 'acm-certificate-arn' | SECRET_ARN 'ssm-secret-arn' ];
```

在 FROM 子句中，MSK 表示結構描述會從 Managed Kafka Services 映射資料。

AUTHENTICATION 表示使用 Amazon 進行串流擷取的身分驗證類型 MSK。有三種類型可用：

- no – 指定不需要身分驗證。這相當於 上的未驗證存取 MSK。
- iam — 指定 IAM 身分驗證。當您選擇此選項時，請確定 IAM 角色具有 IAM 身分驗證的許可。如需設定所需 IAM 政策的詳細資訊，請參閱 [從 Kafka 設定 IAM 許可和執行串流擷取](#)。
- mTLS – 指定相互傳輸層安全透過促進用戶端和伺服器之間的身分驗證來提供安全通訊。在此情況下，用戶端為 Redshift，伺服器為 Amazon MSK。如需使用 m 設定串流擷取的詳細資訊 TLS，請參閱 [使用 mTLS 進行身分驗證，以便從 Amazon 擷取 Redshift 串流 MSK](#)。

請注意，不支援使用使用者名稱和密碼進行串流擷取的 Amazon MSK 身分驗證。

AUTHENTICATION_ARN 指定您用來建立加密連線 ARN 的 ACM 相互傳輸層安全 (mTLS) 憑證的。

SECRET_ARN 指定包含 Amazon Redshift 要用於 m 的憑證之 AWS Secrets Manager 秘密的英文 TLS。

下列範例示範如何在建立外部結構描述時設定 URI Amazon MSK叢集的代理程式：

```
CREATE EXTERNAL SCHEMA my_schema
FROM MSK
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION IAM
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9098'
```

不使用身分驗證：

```
CREATE EXTERNAL SCHEMA my_schema
FROM MSK
AUTHENTICATION none
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9092,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9092'
```

使用 mTLS

```
CREATE EXTERNAL SCHEMA my_schema
FROM MSK
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION MTLS
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-west-1.amazonaws.com:9094,b-
2.myTestCluster.123z8u.c2.kafka.us-west-1.amazonaws.com:9094'
AUTHENTICATION_ARN 'acm-certificate-arn' | [ SECRET_ARN 'ssm-secret-arn' ];
```

如需建立外部結構描述的詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#)。

5. 建立具體化視觀表以取用主題的資料。使用 SQL 命令，例如下列範例。

```
CREATE MATERIALIZED VIEW MyView AUTO REFRESH YES AS
SELECT *
FROM MySchema."mytopic";
```

Kafka 主題名稱會區分大小寫，且可同時包含大寫和小寫字母。若要從具有大寫名稱的主題擷取，您可以在 `enable_case_sensitive_identifier` 工作階段或資料庫層級將組態設定為。如需詳細資訊，請參閱 [名稱和識別碼](#) 與 [enable_case_sensitive_identifier](#)。

若要開啟自動重新整理，請使用 `AUTO REFRESH YES`。預設行為是手動重新整理。

中繼資料資料欄包括下列項目：

中繼資料資料欄	資料類型	描述
<code>kafka_partition</code>	<code>bigint</code>	從 Kafka 主題記錄的分割區 ID
<code>kafka_offset</code>	<code>bigint</code>	Kafka 主題中給定分割區記錄的位移
<code>kafka_timestamp_type</code>	<code>char(1)</code>	在 Kafka 記錄中使用的時間戳記類型： <ul style="list-style-type: none"> • C – 在用戶端記錄建立時間 (<code>CREATE_TIME</code>) • L – 在 Kafka 伺服器端記錄附加時間 (<code>LOG_APPEND_TIME</code>) • U – 記錄建立時間無法使用 (<code>NO_TIMESTAMP_TYPE</code>)
<code>kafka_timestamp</code>	沒有時區的時間戳記	記錄的時間戳記值
<code>kafka_key</code>	<code>varbyte</code>	Kafka 記錄的索引鍵
<code>kafka_value</code>	<code>varbyte</code>	從 Kafka 收到的記錄
<code>kafka_headers</code>	<code>super</code>	從 Kafka 收到的記錄標頭
<code>refresh_time</code>	沒有時區的時間戳記	重新整理的開始時間

請務必注意，如果您的具體化視觀定義中有業務邏輯，這會導致業務邏輯錯誤，這在某些情況下可能會導致串流擷取失敗。這可能會導致您必須捨棄並重新建立具體化視觀表。為了避免這種情況，我們建議您讓業務邏輯保持簡單，並在擷取資料後對資料執行其他邏輯。

- 重新整理檢視，這會調用 Amazon Redshift 從主題讀取，並將資料載入具體化視觀表。

```
REFRESH MATERIALIZED VIEW MyView;
```

- 查詢具體化視觀表中的資料。

```
select * from MyView;
```

執行 REFRESH 時，具體化視觀表會直接從主題更新。您可以建立對應至 Kafka 主題資料來源的具體化視觀表。您可以對資料執行篩選和彙總，以做為具體化視觀表定義的一部分。您的串流擷取具體化視觀表 (基礎具體化視觀表) 只能參照一個 Kafka 主題，但是您可以建立與基礎具體化視觀表及其他具體化視觀表或資料表結合的額外具體化視觀表。

如需串流擷取限制的相關資訊，請參閱 [串流擷取行為和資料類型](#)。

使用 mTLS 進行身分驗證，以便從 Amazon 擷取 Redshift 串流 MSK

相互傳輸層安全性 (mTLS) 為伺服器提供驗證其傳送資訊之用戶端的方法，以及讓用戶端驗證伺服器的方法。使用 mTLS 的好處是為多個產業垂直應用程式中的各種使用案例提供信任的身分驗證。這些包括金融、零售、政府和醫療保健產業的使用案例。在串流擷取至 Redshift 的情況下，身分驗證會在伺服器之間進行，在此情況下是 Amazon MSK、Amazon Redshift 佈建叢集或 Amazon Redshift Serverless 工作群組。

本主題提供程序和 SQL 命令範例，示範建立外部結構描述的方法，該結構描述使用 mTLS 在 Redshift 用戶端和 Amazon MSK 伺服器之間進行身分驗證。本主題中的步驟補充了設定來自 Amazon 的串流擷取的完整步驟。MSK 這些詳細資訊請參閱 [從 Amazon Managed Streaming for Apache Kafka \(Amazon MSK\) 開始串流擷取](#)。

使用 mTLS 進行串流擷取的先決條件

本節提供使用 mTLS 與 AWS Certificate Manager 或 Amazon 進行串流擷取的先決條件步驟 SageMaker。

- 作為初步步驟，您必須擁有或建立私有憑證授權機構 (PCA)，您可以使用它來發行憑證，以透過安全通訊管道啟用安全通訊。AWS Private Certificate Authority (Private CA) 是執行此函數的可用服務。如需詳細資訊，請參閱 AWS Private Certificate Authority 使用者指南 中的 [建立私有 CA](#)。在稍後的步驟中，您會發出憑證並將其連接至您的 Amazon MSK 叢集，以啟用與 Redshift 的加密通訊。

2. 建立支援 mTLS 用戶端身分驗證的 Amazon MSK 叢集。如需設定 Amazon MSK 叢集的詳細資訊，請參閱 Amazon Managed Streaming for Apache Kafka 開發人員指南 中的 [如何建立支援用戶端身分驗證的叢集](#)。
3. 編輯 Amazon MSK 叢集的安全設定，使用 AWS Certificate Manager (ACM) 開啟 TLS 用戶端身分驗證，然後選擇您先前建立的 AWS Private CA (PCA)。如需詳細資訊，請參閱 Amazon Managed Streaming for Apache Kafka 開發人員指南 中的 [更新叢集的安全性設定](#)。

使用 mTLS 與 進行串流擷取 AWS Certificate Manager

下列程序說明如何利用 AWS Certificate Manager (ACM) 進行憑證儲存和管理，為 Redshift 串流擷取設定 mTLS：

1. 透過 請求私有憑證 ACM。執行此操作時，請選取 PCA 您在先決條件區段中建立的 做為憑證授權機構。ACM 會儲存已簽署的憑證和連接的私有金鑰，以進行安全通訊。如需使用 管理憑證的詳細資訊 ACM，請參閱 AWS Certificate Manager 使用者指南 中的 [發行和管理憑證](#)。
2. 對於您用來管理 Redshift 叢集或 Amazon Redshift Serverless 工作群組 IAM 的角色，請連接匯出憑證的許可，即 `acm:ExportCertificate`。如需使用 Amazon 設定串流擷取必要 IAM 資源的詳細資訊 MSK，請參閱 [從 Kafka 設定 IAM 許可和執行串流擷取](#)。在下一個步驟中指定相同的 IAM 角色，以建立外部結構描述。
3. 取得 Amazon MSK 叢集 URI 的引導代理程式。取得引導代理程式 URI 詳述於 [Amazon Managed Streaming for Apache Kafka 開發人員指南 中的取得 Amazon MSK 叢集的引導代理程式](#)。
4. 執行類似下列範例的 SQL 命令，以使用 建立外部結構描述，將 Amazon MSK 叢集中的串流映射至 Redshift 外部結構描述 mTLS。

```
CREATE EXTERNAL SCHEMA my_schema
FROM MSK
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION mTLS
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9094,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9094'
AUTHENTICATION_ARN 'arn:aws:acm:Region:444455556666:certificate/certificate_ID';
```

重要參數：

- IAM_ROLE – 與叢集相關聯的 IAM 角色，用於串流擷取。

- URI – Amazon MSK叢集URI的引導代理程式。請注意，連接埠 9094 是指定用於與代理程式通訊以進行TLS加密。
- AUTHENTICATION_ARN – ACM憑證ARN的。當您選擇發行的憑證時，ACM主控台中ARN會提供。

執行這些組態步驟後，您可以建立 Redshift 具體化檢視，參考範例中定義的結構描述，然後使用 REFRESH MATERIALIZED VIEW 串流資料。這在從 Amazon 開始串流的步驟中詳細說明MSK，網址為：[從 Amazon Managed Streaming for Apache Kafka 開始串流擷取 \(MSK \)](#)。

使用 mTLS 與 進行串流擷取 AWS Secrets Manager

如果您不想在 中參考憑證，則可以利用 AWS Secrets Manager 進行憑證管理，為 Redshift 串流擷取設定 mTLSACM。以下步驟說明如何設定它。

1. 使用您選擇的工具建立憑證簽署請求和私有金鑰。然後，您可以使用簽署請求來產生簽署的憑證，使用與用來產生 Amazon MSK叢集憑證的相同 AWS 私有 CA (PCA)。如需發出憑證的詳細資訊，請參閱 參考 [IssueCertificate](#)中的。AWS Private Certificate Authority API
2. 使用 擷取憑證 AWS Private Certificate Authority。如需詳細資訊，請參閱 AWS Private Certificate Authority 使用者指南 中的[擷取私有憑證 >](#)。
3. 將上一個步驟中產生的憑證和私有金鑰儲存在 中 AWS Secrets Manager。選擇Other type of secret並使用純文字格式。索引鍵值對的格式應該為 {"certificate": "<cert value>", "privateKey": "<pkey value>"}, 如下列範例所示。如需在 中建立和管理秘密的詳細資訊 AWS Secrets Manager，請參閱 AWS Secrets Manager 使用者指南 中的[使用 建立和管理秘密 AWS Secrets Manager](#)。

```
{
  "certificate": "-----BEGIN CERTIFICATE-----
klhds1kfjahksgdfkgioeuyihbflahabhbdslv6akybeoiwv1hoaiusdhbahsbd
H4hAX8/eE96qCcjkpfT84EdvHzp6fC+/WwM0oXlwUEW1vfMCXNaG5D8SqRq3qA==
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
klhds1kfjahksgdfkgioeuyihbflahabhbdslv6akybeoiwv1hoaiusdhbahsbd
wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
-----END CERTIFICATE-----",
  "privateKey": "-----BEGIN PRIVATE KEY-----
klhds1kfjahksgdfkgioeuyihbflahabhbdslv6akybeoiwv1hoaiusdhbahsbd
70D4m1dBES3Fj++hDMH9rYRp99RqtC0f0EW0Ue139K0i10sW+cyhAoc9Ci2+Jo/k
17u2N1iGILMQEZuCRtnJ0kFYkw==
-----END PRIVATE KEY-----"}
}
```

4. 在 Redshift 中，執行 SQL 命令以建立外部結構描述。您可以使用 AUTHENTICATION 類型 `mtls`。您也可以指定 Amazon MSK 叢集 URI 和秘密 AWS Secrets Manager。

```
CREATE EXTERNAL SCHEMA my_schema
FROM MSK
IAM_ROLE 'arn:aws:iam::012345678901:role/my_role'
AUTHENTICATION mtls
URI 'b-1.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9094,b-2.myTestCluster.123z8u.c2.kafka.us-
west-1.amazonaws.com:9094'
SECRET_ARN 'arn:aws:secretsmanager:us-east-1:012345678910:secret:myMTLSSecret';
```

重要參數：

- IAM_ROLE – 與叢集相關聯的 IAM 角色，用於串流擷取。
- URI – Amazon MSK 叢集 URI 的引導代理程式。請注意，連接埠 9094 是指定用於與代理程式通訊以進行 TLS 加密。
- SECRET_ARN – Secrets Manager ARN 的秘密，其中包含用於 m 的憑證 TLS。

啟用現有外部結構描述的 mTLS 身分驗證

如果您有用於串流擷取的現有外部結構描述，而且想要實作相互 TLS 身分驗證，您可以執行下列命令，指定 ARN 中的 mTLS 身分驗證和 ACM 憑證 ACM。

```
ALTER EXTERNAL SCHEMA schema_name
AUTHENTICATION mtls
AUTHENTICATION_ARN 'arn:aws:acm:Region:444455556666:certificate/certificate_ID';
```

或者，您可以參考 [中的秘密](#) 來指定 mTLS ARN 身分驗證 AWS Secrets Manager。

```
ALTER EXTERNAL SCHEMA schema_name
AUTHENTICATION mtls
SECRET_ARN 'arn:aws:secretsmanager:us-east-1:012345678910:secret:myMTLSSecret';
```

使用 Kinesis 擷取串流資料

此程序示範如何從名為 `ev_station_data` 的 Kinesis 串流擷取資料，其中包含來自不同 EV 充電站的耗用資料，JSON 格式為 `{}` 。該結構描述是明確定義的。此範例說明如何將資料儲存為原始資料，JSON 以及如何在擷取資料時將 JSON 資料轉換為 Amazon Redshift 資料類型。

生產者設定

1. 使用 Amazon Kinesis Data Streams，按照以下步驟建立名為 `ev_station_data` 的串流。選擇隨需做為容量模式。如需詳細資訊，請參閱[透過 AWS Management Console 建立串流](#)。
2. [Amazon Kinesis 資料產生器](#) 可協助您產生用於串流的測試資料。請依照工具中的詳細步驟來開始使用，並使用下列資料範本來產生資料：

```
{
  "_id" : "{{random.uuid}}",
  "clusterID": "{{random.number(
    {
      "min":1,
      "max":50
    }
  )}}",
  "connectionTime": "{{date.now("YYYY-MM-DD HH:mm:ss")}}",
  "kWhDelivered": "{{commerce.price}}",
  "stationID": "{{random.number(
    {
      "min":1,
      "max":467
    }
  )}}",
  "spaceID": "{{random.word}}-{{random.number(
    {
      "min":1,
      "max":20
    }
  )}}",
  "timezone": "America/Los_Angeles",
  "userID": "{{random.number(
    {
      "min":1000,
      "max":500000
    }
  )}}"
}
```

串流資料中的每個JSON物件具有下列屬性：

```
{
  "_id": "12084f2f-fc41-41fb-a218-8cc1ac6146eb",
  "clusterID": "49",
  "connectionTime": "2022-01-31 13:17:15",
  "kWhDelivered": "74.00",
  "stationID": "421",
  "spaceID": "technologies-2",
  "timezone": "America/Los_Angeles",
  "userID": "482329"
}
```

Amazon Redshift 設定

這些步驟會說明如何設定具體化視觀表以擷取資料。

1. 建立外部結構描述以將 Kinesis 中的資料對應至 Redshift 物件。

```
CREATE EXTERNAL SCHEMA evdata FROM KINESIS
IAM_ROLE 'arn:aws:iam::0123456789:role/redshift-streaming-role';
```

如需如何設定IAM角色的資訊，請參閱 [開始使用 Amazon Kinesis Data Streams 中的串流擷取](#)。

2. 建立具體化視觀表以取用串流資料。下列範例顯示定義具體化視觀表以擷取JSON來源資料的兩種方法。

首先，以半結構化SUPER格式儲存串流記錄。在此範例中，JSON來源存放在 Redshift 中，而不轉換為 Redshift 類型。

```
CREATE MATERIALIZED VIEW ev_station_data AS
  SELECT approximate_arrival_timestamp,
         partition_key,
         shard_id,
         sequence_number,
         case when can_json_parse(kinesis_data) then json_parse(kinesis_data) else null
         end as payload,
         case when not can_json_parse(kinesis_data) then kinesis_data else null end as
         failed_payload
  FROM evdata."ev_station_data" ;
```

相反地，在下列具體化視觀表定義中，具體化視觀表在 Redshift 中有定義的結構描述。具體化檢視會分佈在串流UUID的值上，並依`approximatearrivaltimestamp`值排序。

```
CREATE MATERIALIZED VIEW ev_station_data_extract DISTKEY(6) sortkey(1) AUTO REFRESH
YES AS
  SELECT refresh_time,
         approximate_arrival_timestamp,
         partition_key,
         shard_id,
         sequence_number,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), '_id', true)::character(36)
         as ID,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), 'clusterID', true)::varchar(30)
         as clusterID,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), 'connectionTime', true)::varchar(30)
         as connectionTime,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), 'kWhDelivered', true)::DECIMAL(10,2)
         as kWhDelivered,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), 'stationID', true)::DECIMAL(10,2)
         as stationID,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), 'spaceID', true)::varchar(100)
         as spaceID,
         json_extract_path_text(from_varbyte(kinesis_data,
         'utf-8'), 'timezone', true)::varchar(30)as timezone,

         json_extract_path_text(from_varbyte(kinesis_data, 'utf-8'), 'userID', true)::varchar(30)
         as userID
  FROM evdata."ev_station_data"
  WHERE LENGTH(kinesis_data) < 65355;
```

查詢串流

1. 查詢重新整理的具體化視觀表以取得使用狀況統計資料。

```
SELECT to_timestamp(connectionTime, 'YYYY-MM-DD HH24:MI:SS') as connectiontime
```

```
,SUM(kWhDelivered) AS Energy_Consumed
,count(distinct userID) AS #Users
from ev_station_data_extract
group by to_timestamp(connectionTime, 'YYYY-MM-DD HH24:MI:SS')
order by 1 desc;
```

2. 檢視結果。

connectiontime	energy_consumed	#users
2022-02-08 16:07:21+00	4139	10
2022-02-08 16:07:20+00	5571	10
2022-02-08 16:07:19+00	8697	20
2022-02-08 16:07:18+00	4408	10
2022-02-08 16:07:17+00	4257	10
2022-02-08 16:07:16+00	6861	10
2022-02-08 16:07:15+00	5643	10
2022-02-08 16:07:14+00	3677	10
2022-02-08 16:07:13+00	4673	10
2022-02-08 16:07:11+00	9689	20

AWS Glue Data Catalog 檢視

本主題說明如何在 [中](#) 建立檢視 AWS Glue Data Catalog。您可以使用 Data Catalog 中的檢視，使用相同的結構描述來存取不同資料來源中的資料。

透過在 Data Catalog 中建立檢視，您可以建立單一通用檢視結構描述和中繼資料物件，以便在 Amazon Athena 和 Amazon EMR Spark 等引擎中使用。這樣做可讓您跨資料湖和資料倉儲使用相同的視觀表，以符合您的使用案例。Data Catalog 中的視觀表的特殊之處在於它們被歸類為定義者視觀表，其中存取權限由建立視觀表的使用者定義，而不是由查詢視觀表的使用者定義。以下是在 Data Catalog 中建立視觀表的一些使用案例和好處：

- 建立根據使用者需要的權限來限制資料存取的視觀表。例如，您可以使用 Data Catalog 中的檢視，以防止不在人力資源部門工作的員工查看個人識別資訊（PII）。
- 請確定使用者無法存取不完整的記錄。透過將某些篩選條件套用至 Data Catalog 視觀表，您可確保 Data Catalog 視觀表中的資料記錄始終是完整的。
- Data Catalog 視觀表具有包含的安全優勢，可確保用於建立視觀表的查詢定義必須完成才能建立視觀表。此安全利益表示資料目錄中的檢視不容易受到惡意播放器的 SQL 命令影響。
- Data Catalog 中的視觀表支援與一般視觀表相同的優點，例如允許使用者存取視觀表，而無需將基礎資料表提供給使用者。

若要在 Data Catalog 中建立視觀表，您必須具有 [Spectrum 外部資料表](#)、[Lake Formation 受管資料共用](#) 中包含的物件，或 [Apache Iceberg 資料表](#)。

Data Catalog 視觀表的定義存放在 AWS Glue Data Catalog 中。AWS Lake Formation 用於透過資源授予、資料欄授予或標籤型存取控制授予存取權。如需有關在 Lake Formation 中授予和撤銷存取權的詳細資訊，請參閱 [授予和撤銷 Data Catalog 資源的權限](#)。

必要條件

在 Data Catalog 中建立視觀表之前，請確定您已完成以下先決條件：

- 請確定您的 IAM 角色具有下列信任政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
```



```

    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "glue.amazonaws.com",
          "lakeformation.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- 您也需要下列傳遞角色政策。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1",
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "glue.amazonaws.com",
            "lakeformation.amazonaws.com"
          ]
        }
      }
    }
  ]
}

```

- 最後，您也需要下列權限。
 - Glue:GetDatabase
 - Glue:GetDatabases
 - Glue:CreateTable
 - Glue:GetTable

- Glue:UpdateTable
- Glue>DeleteTable
- Glue:GetTables
- Glue:SearchTables
- Glue:BatchGetPartition
- Glue:GetPartitions
- Glue:GetPartition
- Glue:GetTableVersion
- Glue:GetTableVersions

End-to-end 範例

首先，根據您的 Data Catalog 資料庫建立外部結構描述。

```
CREATE EXTERNAL SCHEMA IF NOT EXISTS external_schema FROM DATA CATALOG DATABASE
'external_data_catalog_db'
IAM_ROLE 'arn:aws:iam::123456789012:role/sample-role';
```

您現在可以建立 Data Catalog 視觀表。

```
CREATE EXTERNAL PROTECTED VIEW external_schema.remote_view
AS SELECT * FROM external_schema.remote_table;
```

然後，您可以開始查詢您的視觀表。

```
SELECT * FROM external_schema.remote_view;
```

如需有關 Data Catalog 中檢視相關SQL命令的詳細資訊，請參閱 [CREATE EXTERNAL VIEW](#)、[ALTER EXTERNAL VIEW](#)和 [DROP EXTERNAL VIEW](#)。

安全記錄

當查詢參考多方視線檢視時，Redshift 會遮罩在 Redshift 系統日誌中記錄的中繼資料。多方位表示檢視支援不同查詢引擎的SQL方言，例如 Redshift 和 Amazon EMR。下表中的資料會針對具有相同查詢 ID 的所有查詢進行遮罩。下表列出套用安全記錄的系統檢視和資料欄。

系統資料表	敏感資料欄
SYS_EXTERNAL_QUERY_DETAIL	資料欄：source_type、total_partitions、qualified_partitions、scan_files、returned_rows、returned_bytes、file_format、file_location、external_query_text、warning_message。如需詳細資訊，請參閱 SYS_EXTERNAL_QUERY_DETAIL 。
SYS_EXTERNAL_QUERY_ERROR	資料欄：file_location、rowid、Column_name、original_value、modified_value、trigger、action、action_value、error_code。如需詳細資訊，請參閱 SYS_EXTERNAL_QUERY_ERROR 。
SYS_QUERY_DETAIL	資料欄：Step_name、Table_id、Table_name、put_bytes、input_rows、output_bytes、output_rows、block_read、block_write、local_read_IO、remote_read_IO、served_block_local_disk、served_block_remote_disk。如需詳細資訊，請參閱 SYS_QUERY_DETAIL 。
SYS_QUERY_HISTORY	欄位：Return_rows、Return_bytes。如需詳細資訊，請參閱 SYS_QUERY_HISTORY 。
STL_AGGR	資料欄：資料列、位元組、tbl、類型。如需詳細資訊，請參閱 STL_AGGR 。
STL_BCAST	資料欄：資料列、位元組、封包。如需詳細資訊，請參閱 STL_BCAST 。
STL_DDLTEXT	欄位：文字。如需詳細資訊，請參閱 STL_DDLTEXT 。
STL_DELETE	資料欄：資料列、tbl。如需詳細資訊，請參閱 STL_DELETE 。
STL_DIST	資料欄：資料列、位元組、封包。如需詳細資訊，請參閱 STL_DIST 。
STL_EXPLAIN	欄位：Plannode、資訊。如需詳細資訊，請參閱 STL_EXPLAIN 。
STL_HASH	資料欄：資料列、位元組、tbl、est_rows。如需詳細資訊，請參閱 STL_HASH 。
STL_HASHJOIN	資料欄：資料列、tbl、num_parts、join_type。如需詳細資訊，請參閱 STL_HASHJOIN 。

系統資料表	敏感資料欄
STL_INSERT	資料欄：資料列、tbl。如需詳細資訊，請參閱 STL_INSERT 。
STL_LIMIT	資料欄：資料列。如需詳細資訊，請參閱 STL_LIMIT 。
STL_MERGE	資料欄：資料列。如需詳細資訊，請參閱 STL_MERGE 。
STL_MERGEJOIN	資料欄：資料列、tbl。如需詳細資訊，請參閱 STL_MERGEJOIN 。
STL_NESTLOOP	資料欄：資料列、tbl。如需詳細資訊，請參閱 STL_NESTLOOP 。
STL_PARSE	資料欄：資料列。如需詳細資訊，請參閱 STL_PARSE 。
STL_PLAN_INFO	資料欄：資料列、位元組。如需詳細資訊，請參閱 STL_PLAN_INFO 。
STL_PROJECT	資料欄：資料列、tbl。如需詳細資訊，請參閱 STL_PROJECT 。
STL_QUERY	資料欄：querytxt。如需詳細資訊，請參閱 STL_QUERY 。
STL_QUERY_METRICS	資料欄：max_rows、資料列、max_blocks_read、block_read、max_blocks_to_disk、block_to_disk、max_query_scan_size、query_scan_size。如需詳細資訊，請參閱 STL_QUERY_METRICS 。
STL_QUERYTEXT	欄位：文字。如需詳細資訊，請參閱 STL_QUERYTEXT 。
STL_RETURN	資料欄：資料列、位元組。如需詳細資訊，請參閱 STL_RETURN 。
STL_SAVE	資料欄：資料列、位元組、tbl。如需詳細資訊，請參閱 STL_SAVE 。
STL_SCAN	資料欄：資料列、位元組、擷取、類型、tbl、Rows_pre_filter、perm_table_name、Scanned_mega_value。如需詳細資訊，請參閱 STL_SCAN 。
STL_SORT	資料欄：資料列、位元組、tbl。如需詳細資訊，請參閱 STL_SORT 。
STL_TR_CONFLICT	資料欄：table_id。如需詳細資訊，請參閱 STL_TR_CONFLICT 。
STL_UNDONE	資料欄：table_id。如需詳細資訊，請參閱 STL_UNDONE 。

系統資料表	敏感資料欄
STL_UNIQUE	資料欄：資料列、類型、位元組。如需詳細資訊，請參閱 STL_UNIQUE 。
STL_UTILITYTEXT	欄位：文字。如需詳細資訊，請參閱 STL_UTILITYTEXT 。
STL_WINDOW	資料欄：資料列。如需詳細資訊，請參閱 STL_WINDOW 。
STV_BLOCKLIST	欄位：col、tbl、num_values、minvalue、maxvalue。如需詳細資訊，請參閱 STV_BLOCKLIST 。
STV_EXEC_STATE	資料欄：資料列、位元組、標籤。如需詳細資訊，請參閱 STV_EXEC_STATE 。
STV_LOCKS	資料欄：table_id。如需詳細資訊，請參閱 STV_LOCKS 。
STV_QUERY_METRICS	資料欄：資料列、max_rows、block_read、max_blocks_read、max_blocks_to_disk、block_to_disk、max_query_scan_size、query_scan_size。如需詳細資訊，請參閱 STV_QUERY_METRICS 。
STV_STARTUP_RECOVERY_STATE	資料欄：table_id、table_name。如需詳細資訊，請參閱 STV_STARTUP_RECOVERY_STATE 。
STV_TBL_PERM	資料欄：id、資料列、red_rows、temp、block_count、query_scan_size。如需詳細資訊，請參閱 STV_TBL_PERM 。
STV_TBL_TRANS	資料欄：id、資料列、大小。如需詳細資訊，請參閱 STV_TBL_TRANS 。
SVCS_EXPLAIN	欄位：Plannode、資訊。如需詳細資訊，請參閱 SVCS_EXPLAIN 。
SVCS_PLAN_INFO	資料欄：資料列、位元組。如需詳細資訊，請參閱 SVCS_PLAN_INFO 。
SVCS_QUERY_SUMMARY	資料欄：步驟、資料列、位元組、rate_row、rate_byte、label、rows_pre_filter。如需詳細資訊，請參閱 SVCS_QUERY_SUMMARY 。

系統資料表	敏感資料欄
SVCS_S3LIST	欄位：儲存貯體、字首、max_file_size、avg_file_size。如需詳細資訊，請參閱 SVCS_QUERY_SUMMARY 。
SVCS_S3LOG	欄位：訊息。如需詳細資訊，請參閱 SVCS_QUERY_SUMMARY 。
SVCS_S3PARTITION_SUMMARY	欄位：total_partitions、qualified_partitions、min_assigned_partitions、max_assigned_partitions、avg_assigned_partitions。如需詳細資訊，請參閱 SVCSSUMMARYPARTITION 。
SVCS_S3QUERY_SUMMARY	欄位：external_table_name、file_format、s3_scanned_rows、s3_scanned_bytes、s3query_returned_rows、s3query_returned_bytes。如需詳細資訊，請參閱 SVCSSUMMARYQUERY 。
SVL_QUERY_METRICS	資料欄：Step_label、Scan_row_count、join_row_count、nested_loop_join_row_count、return_row_count、spectrum_scan_row_count、spectrum_scan_size_mb。如需詳細資訊，請參閱 SVL_QUERY_METRICS 。
SVL_QUERY_METRICS_SUMMARY	資料欄：Step_label、Scan_row_count、join_row_count、nested_loop_join_row_count、return_row_count、spectrum_scan_row_count、spectrum_scan_size_mb。如需詳細資訊，請參閱 SVL_QUERY_METRICS_SUMMARY 。
SVL_QUERY_REPORT	資料欄：資料列、位元組、標籤、資料列_pre_filter。如需詳細資訊，請參閱 SVL_QUERY_REPORT 。
SVL_QUERY_SUMMARY	資料欄：資料列、位元組、資料列_pre_filter。如需詳細資訊，請參閱 SVL_QUERY_SUMMARY 。
SVL_S3LIST	欄位：儲存貯體、字首、max_file_size、avg_file_size。如需詳細資訊，請參閱 SVL_S3LIST 。
SVL_S3LOG	欄位：訊息。如需詳細資訊，請參閱 SVL_S3LOG 。
SVL_S3PARTITION	資料欄：資料列、位元組、標籤、資料列_pre_filter。如需詳細資訊，請參閱 SVL_S3PARTITION 。

系統資料表	敏感資料欄
SVL_S3PARTITION_SUMMARY	欄位：total_partitions、qualified_partitions、min_assigned_partitions、max_assigned_partitions、avg_assigned_partitions。如需詳細資訊，請參閱 SVL_S3PARTITION_SUMMARY 。
SVL_S3QUERY	欄位：external_table_name、file_format、s3_scanned_rows、s3_scanned_bytes、s3query_returned_rows、s3query_returned_bytes。如需詳細資訊，請參閱 SVL_S3QUERY 。
SVL_S3QUERY_SUMMARY	欄位：external_table_name、file_format、s3_scanned_rows、s3_scanned_bytes、s3query_returned_rows、s3query_returned_bytes。如需詳細資訊，請參閱 SVL_S3QUERY_SUMMARY 。
SVL_S3RETRIES	欄位：file_size、位置、訊息。如需詳細資訊，請參閱 SVL_S3RETRIES 。
SVL_SPECTRUM_SCAN_ERROR	資料欄：位置、Rowid、colname、Original_value、Remoded_value。如需詳細資訊，請參閱 SVL_SPECTRUM_SCAN_ERROR 。
SVL_STATEMENTTEXT	欄位：類型、文字。如需詳細資訊，請參閱 SVL_STATEMENTTEXT 。
SVL_STORED_PROC_CALL	資料欄：querytxt。如需詳細資訊，請參閱 SVL_STORED_PROC_CALL 。
SVL_STORED_PROC_MESSAGES	欄位：查詢文字。如需詳細資訊，請參閱 SVL_STORED_PROC_MESSAGES 。
SVL_UDF_LOG	資料欄：funcname。如需詳細資訊，請參閱 SVL_UDF_LOG 。
SVV_DISKUSAGE	欄位：名稱、col、tbl、區塊、num_values、minvalue、maxvalue。如需詳細資訊，請參閱 SVV_DISKUSAGE 。
SVV_QUERY_STATE	資料欄：資料列、位元組、標籤。如需詳細資訊，請參閱 SVV_QUERY_STATE 。
SVV_TABLE_INFO	資料欄：table_id、table。如需詳細資訊，請參閱 SVV_TABLE_INFO 。
SVV_TRANSACTIONS	欄位：關係。如需詳細資訊，請參閱 SVV_TRANSACTIONS 。

考量與限制

以下是適用於在 Data Catalog 中建立的視觀表的考量事項和限制。

- 您無法建立以其他視觀表為基礎的 Data Catalog 視觀表。
- 在 Data Catalog 視觀表中，您只能有 10 個基本資料表。
- 視觀表的定義者必須擁有基本資料表的完整 SELECT GRANTABLE 權限。
- 視觀表只能包含 Lake Formation 物件和內建項目。視觀表內不允許使用以下物件。
 - 系統表
 - 使用者定義的函數 (UDFs)
 - 不在 Lake Formation 受管資料共用中的 Redshift 資料表、視觀表、具體化視觀表和後期繫結視觀表。
- 視觀表不能包含巢狀 Redshift Spectrum 表。
- AWS Glue 檢視基本物件的表示法必須與檢視位於相同 AWS 帳戶 和 區域中。

在 Amazon Redshift 中查詢空間資料

空間資料描述已定義空間 (空間參考系統) 中幾何圖形的位置和形狀。Amazon Redshift 支援具有 GEOMETRY 和 GEOGRAPHY 資料類型的空間資料，其中包含空間資料和可選的資料空間參考系統識別符 (SRID)。

空間資料包含能用來表示地理特徵的幾何資料。這類資料的例子包括天氣報告、地圖方向、含地理位置的推文、商店位置，以及飛機航線。空間資料在商業分析、報告和預測方面扮演重要角色。

您可以使用 Amazon Redshift SQL 函數查詢空間資料。空間資料包含物件的幾何值。

GEOMETRY 資料類型操作會在笛卡爾平面上運作。雖然空間參考系統識別符 (SRID) 存放在物件內，但這 SRID 只是座標系統的識別符，而且在用於處理 GEOMETRY 物件的演算法中不會扮演任何角色。相反地，GEOGRAPHY 資料類型上的操作會將物件內部的座標視為球體上的球形座標。此球狀體由定義 SRID，其參考地理空間參考系統。根據預設，GEOGRAPHY 資料類型是使用空間參考 (SRID) 4326 建立，參考世界地理系統 (WGS) 84。如需的詳細資訊 SRIDs，請參閱 Wikipedia 中的 [空間參考系統](#)。

您可以使用 ST_Transform 函數來轉換來自各種空間參考系統的座標。完成座標轉換後，您也可以兩者之間使用簡單的轉換，只要輸入 GEOMETRY 使用地理編碼即可 SRID。這種轉換只是複製坐標而無需進一步的轉換。例如：

```
SELECT ST_AsEWKT(ST_GeomFromEWKT('SRID=4326;POINT(10 20)')::geography);
```

```
st_asewkt
```

```
-----  
SRID=4326;POINT(10 20)
```

若要進一步了解 GEOMETRY 和 GEOGRAPHY 資料類型之間的差異，請考慮 SFO 使用世界地理系統 (BER) 84 計算柏林機場 () 和舊金山機場 (WGS) 之間的距離。使用 GEOGRAPHY 資料類型時，結果會以公尺為單位。將 GEOMETRY 資料類型與 SRID 4326 搭配使用時，結果為度，無法轉換為公尺，因為一度的距離取決於地球幾何的位置。

對 GEOGRAPHY 資料類型的計算主要用於現實的圓形地球計算，如一個國家/地區精確而不失真的面積。但是其計算成本要高得多。因此，ST_Transform 可以將您的座標轉換為適當的局部投影座標系統，並更快地對 GEOMETRY 資料類型進行計算。

使用空間資料，您可以執行查詢來執行以下作業：

- 尋找兩點之間的距離。
- 檢查是否一個區域 (多邊形) 包含其他區域。
- 檢查一個線串 (linestring) 是否與另一個線串或多邊形交集。

您可以使用 GEOMETRY 資料類型來保存空間資料的值。Amazon Redshift 中的 GEOMETRY 值可以定義二維 (2D)、三維 (3DZ)、具量值二維 (3DM) 及四維 (4D) 幾何基本資料類型：

- 二維 (2D) 幾何圖形由平面中的兩個直角座標 (x、y) 指定。
- 三維 (3DZ) 幾何圖形由空間中的三個直角座標 (x、y、z) 指定。
- 具量值二維 (3DM) 幾何圖形由三個坐標 (x、y、m) 指定，前兩個座標是平面中的直角坐標，第三個是測量值。
- 四維 (4D) 幾何圖形由四個坐標 (x、y、z、m) 指定，前三個座標是空間中的直角坐標，第四個是測量值。

如需基本幾何資料類型的相關資訊，請參閱 Wikipedia 中的 [Well-known text representation of geometry \(幾何圖形的熟知文字表示法\)](#)。

您可以使用 GEOGRAPHY 資料類型來保存空間資料的值。Amazon Redshift 中的 GEOGRAPHY 值可以定義二維 (2D)、三維 (3DZ)、具量值二維 (3DM) 及四維 (4D) 幾何基本資料類型：

- 二維 (2D) 幾何圖形由球體上的經度和緯度座標指定。
- 三維 (3DZ) 幾何圖形由球體上的經度、緯度和高度座標指定。
- 具量值二維 (3DM) 幾何圖形由三個坐標 (經度、緯度和量值) 指定，前兩個座標是球體上的角度坐標，第三個是測量值。
- 四維 (4D) 幾何圖形由四個座標 (經度、緯度、高度、量值) 指定，其中前三個是經度、緯度和高度，第四個是測量值。

如需地理坐標系統的相關資訊，請參閱 Wikipedia 中的 [地理坐標系統](#) 和 [球形坐標系統](#)。

GEOMETRY 和 GEOGRAPHY 資料類型具有下列子類型：

- POINT
- LINESTRING
- POLYGON
- MULTIPOINT

- MULTILINESTRING
- MULTIPOLYGON
- GEOMETRYCOLLECTION

Amazon Redshift SQL函數支援下列幾何資料表示法：

- 地理JSON
- 眾所周知的文字（WKT）
- 延伸已知文字（EWKT）
- 眾所周知的二進位（WKB）表示法
- 擴充已知的二進位（EWKB）

您可以在 GEOMETRY 和 GEOGRAPHY 資料類型之間進行轉換。

下列 會將 linestring 從 SQL轉換為 GEOMETRY GEOGRAPHY。

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)')::geography);
```

```
st_asewkt
```

```
-----  
SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

下列 會將 linestring 從 SQL轉換為 GEOGRAPHY GEOMETRY。

```
SELECT ST_AsEWKT(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)')::geometry);
```

```
st_asewkt
```

```
-----  
SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

Amazon Redshift 提供許多查詢空間資料的SQL函數。除了 ST_IsValid 函數之外，接受 GEOMETRY 物件作為引數的空間函數會預期此 GEOMETRY 物件是有效的幾何圖形。如果 GEOMETRY 或 GEOGRAPHY 物件無效，則不會定義空間函數的行為。如需有效性的相關資訊，請參閱 [幾何有效性](#)。

如需查詢空間資料SQL之函數的詳細資訊，請參閱 [空間函數](#)。

如需載入空間資料的詳細資訊，請參閱 [載入 GEOMETRY 或 GEOGRAPHY 資料類型的資料欄](#)。

主題

- [教學課程：搭配 Amazon Redshift 使用空間SQL函數](#)
- [將 Shapefile 載入 Amazon Redshift](#)
- [Amazon Redshift 空間資料的術語](#)
- [將空間資料與 Amazon Redshift 搭配使用時的考量事項](#)

教學課程：搭配 Amazon Redshift 使用空間SQL函數

本教學課程示範如何將部分空間SQL函數與 Amazon Redshift 搭配使用。

若要這麼做，您可以使用空間SQL函數查詢兩個資料表。該教學課程會使用公共資料集中的資料，該資料將德國柏林的出租住所位置資料與郵遞區號建立關聯。

主題

- [必要條件](#)
- [步驟 1：建立資料表並載入測試資料](#)
- [步驟 2：查詢空間資料](#)
- [步驟 3：清除您的資源](#)

必要條件

在此教學課程中，您需執行下列資源：

- 您可以存取和更新的現有 Amazon Redshift 叢集和資料庫。在現有叢集中，您可以建立資料表、載入範例資料和執行SQL查詢，以示範空間函數。您的叢集至少要有兩個節點。若要了解如何建立叢集，請按照 [Amazon Redshift 入門指南](#) 中的步驟進行操作。
- 若要使用 Amazon Redshift 查詢編輯器，請確定您的叢集位於支援查詢編輯器的 AWS 區域中。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [使用查詢編輯器來查詢資料庫](#)。
- AWS Amazon Redshift 叢集的憑證，允許它從 Amazon S3 載入測試資料。如需如何存取 Amazon S3 AWS 等其他服務的詳細資訊，請參閱 [授權 Amazon Redshift 存取 AWS 服務](#)。
- 名為的 AWS Identity and Access Management (IAM) 角色mySpatialDemoRole，其具有AmazonS3ReadOnlyAccess連接至讀取 Amazon S3 資料的受管政策。若要建立具有從 Amazon

S3 儲存貯體載入資料許可的角色，請參閱[Amazon Redshift COPY管理指南中的使用IAM角色授權 UNLOAD、和 CREATEEXTERNALSCHEMA操作](#)。

- 建立IAM角色 之後mySpatialDemoRole，該角色需要與您的 Amazon Redshift 叢集建立關聯。如需如何建立該關聯的詳細資訊，請參閱 Amazon Redshift [COPY管理指南 中的使用IAM角色授權 UNLOAD、和 CREATEEXTERNALSCHEMA操作](#)。

步驟 1：建立資料表並載入測試資料

本教學課程使用的來源資料來自名為 accommodations.csv 和 zipcodes.csv 的檔案。

accommodations.csv 檔案是來自 insideairbnb.com 的開放原始碼資料。zipcodes.csv 檔案提供的郵遞區號是來自德國柏林布蘭登堡國家統計機構 (Amt für Statistik Berlin-Brandenburg) 的開放原始碼資料。兩個資料來源皆經過創用 CC (Creative Commons) 授權。該資料僅限於德國柏林地區。這些檔案位於 Amazon S3 公用儲存貯體中，可與本教學課程搭配使用。

您可以選擇性地從下列 Amazon S3 連結下載來源資料：

- [accommodations 資料表的來源資料](#)。
- [zipcode 資料表的來源資料](#)。

使用下列程序來建立資料集並載入測試資料。

建立資料表並載入測試資料

1. 開啟 Amazon Redshift 查詢編輯器。如需使用查詢編輯器的相關資訊，請參閱《Amazon Redshift 管理指南》中的[使用查詢編輯器來查詢資料庫](#)。
2. 刪除本教學課程使用的任何資料表 (如果這些資料表已經存在於您的資料庫中)。如需詳細資訊，請參閱[步驟 3：清除您的資源](#)。
3. 建立 accommodations 資料表以儲存每個住所的地理位置 (經度和緯度)、清單名稱以及其他業務資料。

本教學課程會探討德國柏林的房間租賃。shape 資料欄儲存的是住所位置的地理點。其他資料欄包含有關租賃的資訊。

若要建立 accommodations 資料表，請在 Amazon Redshift 查詢編輯器中執行下列 SQL 陳述式。

```
CREATE TABLE public.accommodations (
```

```

id INTEGER PRIMARY KEY,
shape GEOMETRY,
name VARCHAR(100),
host_name VARCHAR(100),
neighbourhood_group VARCHAR(100),
neighbourhood VARCHAR(100),
room_type VARCHAR(100),
price SMALLINT,
minimum_nights SMALLINT,
number_of_reviews SMALLINT,
last_review DATE,
reviews_per_month NUMERIC(8,2),
calculated_host_listings_count SMALLINT,
availability_365 SMALLINT
);

```

4. 在查詢編輯器中建立 `zipcode` 資料表以儲存柏林郵遞區號。

郵遞區號在 `wkb_geometry` 資料欄中定義為多邊形。其餘資料欄會描述有關郵遞區號的其他空間中繼資料。

若要建立 `zipcode` 資料表，請在 Amazon Redshift 查詢編輯器中執行下列 SQL 陳述式。

```

CREATE TABLE public.zipcode (
  ogc_field INTEGER PRIMARY KEY NOT NULL,
  wkb_geometry GEOMETRY,
  gml_id VARCHAR(256),
  spatial_name VARCHAR(256),
  spatial_alias VARCHAR(256),
  spatial_type VARCHAR(256)
);

```

5. 使用範例資料來載入資料表。

本教學課程的範例資料提供於 Amazon S3 儲存貯體中，允許所有已驗證 AWS 使用者的讀取存取權。請確定您已提供允許存取 Amazon S3 的有效 AWS 登入資料。

若要將測試資料載入資料表，請執行下列 COPY 命令。Replace (取代) `account-number` 使用您自己的 AWS 帳戶號碼。以單引號括住的登入資料字串區段不可包含任何空格或分行符號。

```

COPY public.accommodations
FROM 's3://redshift-downloads/spatial-data/accommodations.csv'
DELIMITER ','

```

```
IGNOREHEADER 1 REGION 'us-east-1'
CREDENTIALS 'aws_iam_role=arn:aws:iam::account-number:role/mySpatialDemoRole';
```

```
COPY public.zipcode
FROM 's3://redshift-downloads/spatial-data/zipcode.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
CREDENTIALS 'aws_iam_role=arn:aws:iam::account-number:role/mySpatialDemoRole';
```

6. 執行下列命令，可驗證每一個資料表是否正確載入。

```
select count(*) from accommodations;
```

```
select count(*) from zipcode;
```

下列結果會顯示每個測試資料資料表的列數。

資料表名稱	資料列
住所	22,248
zipcode	190

步驟 2：查詢空間資料

建立和載入資料表後，您可以使用SQLSELECT陳述式來查詢它們。下列查詢示範的是一些您可以擷取的資訊。您可以編寫許多其他使用空間函數來滿足您需求的查詢。

查詢空間資料

- 查詢以取得 accommodations 資料表中所儲存清單項目的總數，如下所示。空間參考系統為 World Geodetic System (WGS) 84，其具有唯一的空間參考識別碼 4326。

```
SELECT count(*) FROM public.accommodations WHERE ST_SRID(shape) = 4326;
```

```
count
-----
```

22248

2. 使用一些其他屬性，以眾所周知的文字（WKT）格式擷取幾何物件。此外，您可以驗證此郵遞區號資料是否也儲存在 World Geodetic System（WGS）84 中，該系統使用空間參考 ID（SRID）4326。空間資料必須儲存在相同的空間參考系統中才能互通。

```
SELECT ogc_field, spatial_name, spatial_type, ST_SRID(wkb_geometry),
       ST_AsText(wkb_geometry)
FROM public.zipcode
ORDER BY spatial_name;
```

```
ogc_field  spatial_name  spatial_type  st_srid  st_astext
-----
0          10115        Polygon      4326     POLYGON((...))
4          10117        Polygon      4326     POLYGON((...))
8          10119        Polygon      4326     POLYGON((...))
...
(190 rows returned)
```

3. 選取柏林 Mitte（10117）的多邊形，以 GeoJSON 格式、其維度和此多邊形中的點數量表示。

```
SELECT ogc_field, spatial_name, ST_AsGeoJSON(wkb_geometry),
       ST_Dimension(wkb_geometry), ST_NPoints(wkb_geometry)
FROM public.zipcode
WHERE spatial_name='10117';
```

```
ogc_field  spatial_name  spatial_type
st_dimension  st_npoint
-----
4            10117        {"type":"Polygon", "coordinates":[[[...]]]}      2
331
```

4. 執行下列 SQL 命令，以檢視距布蘭登堡門 500 公尺內的住宿數量。

```
SELECT count(*)
FROM public.accommodations
WHERE ST_DistanceSphere(shape, ST_GeomFromText('POINT(13.377704 52.516431)', 4326))
< 500;
```



```
count
-----
29
```

5. 透過執行以下查詢，從附近所列出住所中儲存的資料取得布蘭登堡門的粗略位置。

此查詢需要子選取 (subselect)。這會產生不同的計數，因為要求的位置與以前的查詢不一樣，這更接近住所。

```
WITH poi(loc) as (
  SELECT st_astext(shape) FROM accommodations WHERE name LIKE '%brandenburg gate%'
)
SELECT count(*)
FROM accommodations a, poi p
WHERE ST_DistanceSphere(a.shape, ST_GeomFromText(p.loc, 4326)) < 500;
```

```
count
-----
60
```

6. 執行以下查詢以顯示布蘭登堡門周圍所有住所的詳細資訊 (按價格遞減排序)。

```
SELECT name, price, ST_AsText(shape)
FROM public.accommodations
WHERE ST_DistanceSphere(shape, ST_GeomFromText('POINT(13.377704 52.516431)', 4326))
< 500
ORDER BY price DESC;
```

```
name                                                                 price  st_astext
-----
DUPLEX APARTMENT/PENTHOUSE in 5* LOCATION! 7583                    300
  POINT(13.3826510209548 52.5159819722552)
DUPLEX-PENTHOUSE IN FIRST LOCATION! 7582                          300
  POINT(13.3799997083855 52.5135918444834)
...
(29 rows returned)
```

7. 執行以下查詢，透過郵遞區號擷取最昂貴的住所。

```
SELECT
  a.price, a.name, ST_AsText(a.shape),
  z.spatial_name, ST_AsText(z.wkb_geometry)
FROM accommodations a, zipcode z
WHERE price = 9000 AND ST_Within(a.shape, z.wkb_geometry);
```

```
price  name                               st_astext
      spatial_name      st_astext
-----
9000   Ueber den Dächern Berlins Zentrum  POINT(13.334436985013
52.4979779501538)  10777          POLYGON((13.3318284987227
52.4956021172799,...
```

8. 透過使用子查詢來計算出最高，最低或中等的住所價格。

下列查詢依郵遞區號列出了中等的住所價格。

```
SELECT
  a.price, a.name, ST_AsText(a.shape),
  z.spatial_name, ST_AsText(z.wkb_geometry)
FROM accommodations a, zipcode z
WHERE
  ST_Within(a.shape, z.wkb_geometry) AND
  price = (SELECT median(price) FROM accommodations)
ORDER BY a.price;
```

```
price name                               st_astext
      spatial_name      st_astext
-----
45    "Cozy room Berlin-Mitte"          POINT(13.3864349535358 52.5292016386514)
10115          POLYGON((13.3658598465795 52.535659581048, ...
...
(723 rows returned)
```

9. 執行以下查詢以擷取柏林中列出的住所數量。為了找到熱點，這些住所會按郵遞區號進行分組，並按供應量排序。

```
SELECT z.spatial_name as zip, count(*) as numAccommodations
FROM public.accommodations a, public.zipcode z
WHERE ST_Within(a.shape, z.wkb_geometry)
GROUP BY zip
ORDER BY numAccommodations DESC;
```

```
zip    numaccommodations
-----
10245  872
10247  832
10437  733
10115  664
...
(187 rows returned)
```

步驟 3：清除您的資源

叢集只要執行就會繼續產生費用。完成本教學課程後，您可以刪除範例叢集。

如果您要保留叢集，但又想復原測試資料表所使用的儲存體，請執行下列命令來刪除資料表。

```
drop table public.accommodations cascade;
```

```
drop table public.zipcode cascade;
```

將 Shapefile 載入 Amazon Redshift

您可以使用 COPY 命令，將存放在 Amazon S3 中的 Esri 形狀檔擷取至 Amazon Redshift 資料表。Shapefile 會以向量格式儲存地理特徵的幾何位置和屬性資訊。Shapefile 格式可以在空間上描述空間物件，例如點、線和多邊形。如需 Shapefile 的相關資訊，請參閱 Wikipedia 中的 [Shapefile](#)。

COPY 命令支援資料格式參數 SHAPEFILE。依預設，Shapefile 的第一個資料欄是 GEOMETRY 或 IDENTITY 資料欄。所有後續的資料欄都會遵循 Shapefile 中指定的順序。但是，目標資料表不需要處於此確切配置，因為您可以使用 COPY 資料欄映射來定義順序。如需有關 COPY 命令 shapefile 支援的資訊，請參閱 [SHAPEFILE](#)。

在某些情況下，產生的幾何圖形大小可能會大於 Amazon Redshift 中可儲存幾何圖形的最大值。如果是這樣，您可以使用 COPY 選項 SIMPLIFY 或 SIMPLIFY AUTO 簡化擷取期間的幾何，如下所示：

- 指定 SIMPLIFY *tolerance* 以使用 Ramer-Douglas-Peucker 演算法和指定的容差簡化擷取期間的所有幾何。
- 指定 SIMPLIFY AUTO 而不容忍，只使用演算法簡化大於大小上限的 Ramer-Douglas-Peucker 幾何。此方法會計算仍可在大小上限內儲存物件的最小容許值。
- 指定 SIMPLIFY AUTO *max_tolerance* 以僅使用演算法和自動計算的容差來 Ramer-Douglas-Peucker 簡化大於大小上限的幾何。此方法可確保容許值不超過容許值上限。

如需 GEOMETRY 資料值大小上限的詳細資訊，請參閱 [將空間資料與 Amazon Redshift 搭配使用時的考量事項](#)。

在某些情況下，容許值會低到記錄無法縮小到 GEOMETRY 資料值的大小上限。在這些情況下，您可以使用 COPY 命令 MAXERROR 的選項來忽略所有或最多特定數量的擷取錯誤。

此 COPY 命令也支援載入 GZIP shapefiles。若要執行此操作，請指定 COPY GZIP 參數。使用此選項時，必須獨立壓縮所有 Shapefile 元件，並共用相同的壓縮字尾。

如果投影描述檔案 (.prj) 與 shapefile 存在，Redshift 會使用它來判斷空間參考系統 ID (SRID)。如果 SRID 有效，則產生的幾何會 SRID 指派此。如果與輸入幾何相關聯的 SRID 值不存在，則產生的幾何 SRID 值為零。您可以將 SET read_srid_on_shapefile_ingestion 設定為 OFF，在工作階段層級停用空間參考系統 ID 的自動偵測。

查詢 SYS_SPATIAL_SIMPLIFY 或 SVL_SPATIAL_SIMPLIFY 系統檢視，即可檢視已簡化的記錄及已計算的容許值。當您指定 SIMPLIFY *tolerance*，此檢視會包含每個 COPY 操作的記錄。若不指定，則會包含每個已簡化幾何圖形的記錄。如需詳細資訊，請參閱 [SYS_SPATIAL_SIMPLIFY](#) 或 [SVL_SPATIAL_SIMPLIFY](#)。

如需載入 Shapefile 的範例，請參閱 [將 Shapefile 載入 Amazon Redshift](#)。

Amazon Redshift 空間資料的術語

下列術語用來描述 Amazon Redshift 的一些空間函數。

邊界框

幾何或地理的週框方塊會定義為幾何或地理中所有點座標範圍的交叉乘積 (跨維度)。對於二維幾何圖形，週框方塊是完全包含幾何中所有點的矩形。例如，多邊形 POLYGON((0 0, 1 0, 0 2, 0 0)) 的

週框方塊是由點 (0, 0) 和 (1, 2) 定義為其左下角和右上角的矩形。Amazon Redshift 會在幾何圖形中預先計算並儲存週框方塊，以加速形成幾何述詞和空間聯結。例如，如果兩個幾何圖形的週框方塊不相交，則這兩個幾何圖形就不會相交，而且不能存在於使用 `st_Intersect` 述詞的空間聯結結果集中。

您可以使用空間函數來加入 ([AddBBox](#))、捨棄 ([DropBBox](#)) 及決定週框方塊的支援 ([SupportsBBox](#))。Amazon Redshift 支援對所有幾何子類型預先計算週框方塊。

下列範例顯示如何更新資料表中的現有幾何圖形，以將其與週框方塊一起儲存。如果您叢集的叢集版本為 1.0.26809 或更新版本，則依預設會使用預先計算的週框方塊建立所有新的幾何圖形。

```
UPDATE my_table SET geom = AddBBox(geom) WHERE SupportsBBox(geom) = false;
```

更新現有幾何之後，建議您在更新的資料表上執行 `VACUUM` 命令。如需詳細資訊，請參閱 [VACUUM](#)。

若要設定工作階段期間是否使用週框方塊對幾何圖形進行編碼，請參閱 [default_geometry_encoding](#)。

幾何有效性

Amazon Redshift 使用的幾何演算法會假設輸入幾何圖形是有效的幾何圖形。如果演算法的輸入無效，則結果處於未定義狀態。下節說明 Amazon Redshift 針對每個幾何子類型使用的幾何有效性定義。

點 (Point)

若以下其中一個條件成立，則點將視為有效：

- 點是空點。
- 所有點座標均為有限浮點數。

點可以是空點。

線串 (Linestring)

若以下任何一個條件成立，則線串將視為有效：

- 線串是空的，也就是不包含任何點。
- 非空線串中的所有點都具有使用有限浮點數的座標。
- 如果線串不是空的，則必須是一維線條；也就是不能退化為一個點。

線串不能包含空點。

線串可以具有重複的連續點。

線串可以具有自我相交。

多邊形 (Polygon)

若以下任何一個條件成立，則多邊形將視為有效：

- 多邊形是空的，也就是不包含任何環。
- 如果不是空的，則當下列所有條件皆成立時，多邊形為有效：
 - 多邊形的所有環都是有效的。若以下任何所有條件皆成立，則環將視為有效：
 - 環的所有點都具有使用有限浮點數的座標。
 - 環是封閉的；也就是環的第一個點和最後一個點重合。
 - 環沒有任何自我交集。
 - 環是二維的。
 - 多邊形的環具有一致的方向。也就是說，如果您遍歷任何環，多邊形的內部還會在您的右側或左側。這意味著，如果多邊形的外部環是順時針或逆時針方向，則所有多邊形的內部環必須具有相同的逆時鐘或順時針方向。
 - 所有內環必須位於多邊形的外環內。
 - 內環不能是巢狀結構，也就是說，內環不能在另一個內環內。
 - 內環和外環只能在有限數量的點上相交。
 - 多邊形的內部必須單連通。

多邊形不能包含空點。

多點 (Multipoint)

若以下任何一個條件成立，則多點將視為有效：

- 多點是空的，也就是不包含任何點。
- 多點不是空的，且根據點有效性定義，所有點都是有效的。

多點可以包含一個或多個空點。

多點可以有重複的點。

多重線串 (Multiinestring)

若以下任何一個條件成立，則多重線串將視為有效：

- 多重線串是空的，也就是不包含任何線串。
- 根據線串有效性定義，非空多重線串中的所有線串都是有效的。

僅由空線串組成的非空多重線串會視為有效。

多重線串中的空線串不會影響其有效性。

多重線串可以具有包含重複連續點的線串。

多重線串可以具有自我相交。

多重線串不能包含空點。

多重多邊形 (Multipolygon)

若以下任何一個條件成立，則多重多邊形將視為有效：

- 多重多邊形不包含任何多邊形 (也就是空的)。
- 多重多邊形不是空的，且下列所有條件皆成立：
 - 多重多邊形中的所有多邊形都是有效的。
 - 多重多邊形中沒有兩個多邊形可以在無限數量的點上相交。特別是，這意味著任何兩個多邊形的內部不能相交，並且只能在有限數量的點上碰到。

多重多邊形中的空多邊形不會使多重多邊形無效。

多重多邊形不能包含空點。

幾何集合

若以下任何一個條件成立，則幾何集合將視為有效：

- 幾何集合是空的，也就是不包含任何幾何圖形。
- 非空幾何集合中的所有幾何圖形都是有效的。

此定義仍然適用於巢狀幾何圖形集合，但是以遞迴方式套用。

幾何集合可以包含空點和具有空點的多點。

幾何簡單性

Amazon Redshift 使用的幾何演算法會假設輸入幾何圖形是有效的幾何圖形。如果演算法的輸入無效，則簡單性檢查會處於未定義狀態。下節說明 Amazon Redshift 針對每個幾何子類型使用的幾何簡單性定義。

點 (Point)

若以下任何一個條件成立，則有效點將視為簡單：

- 有效點一律視為簡單。
- 空點視為簡單。

線串 (Linestring)

若以下任何一個條件成立，則有效線串將視為簡單：

- 線串是空的。
- 線串不是空的，且下列所有條件皆成立：
 - 線串沒有重複的連續點。
 - 線串沒有自我相交，除了可能是會重合的第一個點和最後一個點。換句話說，除了在邊界點之外，線串不能具有自我相交。

多邊形

如果有效多邊形不包含任何重複的連續點，則視為簡單多邊形。

多點 (Multipoint)

若以下任何一個條件成立，則有效多點將視為簡單：

- 多點是空的，也就是不包含任何點。
- 多點的兩個非空點不重合。

多重線串 (Multilinestring)

若以下任何一個條件成立，則有效多重線串將視為簡單：

- 多重線串是空的。
- 多重線串不是空的，且下列所有條件皆成立：
 - 多重線串的所有線串都是簡單線串。
 - 多重線串的任何兩條線串都不相交，除了作為兩個線串邊界點的點。

僅由空線串組成的非空多重線串會視為空多重線串。

多重線串中的空線串不會影響其簡單性。

多重線串中的封閉線串不能與多重線串中的任何其他線串相交。

多重線串不可具有包含重複連續點的線串。

多重多邊形 (Multipolygon)

如果有效多重多邊形不包含任何重複的連續點，則視為簡單多邊形。

幾何集合

若以下任何一個條件成立，則有效幾何集合將視為簡單：

- 幾何集合是空的，也就是不包含任何幾何圖形。
- 非空幾何集合中的所有幾何圖形都是簡單的。

此定義仍然適用於巢狀幾何圖形集合，但是以遞迴方式套用。

H3

H3 是一種分層式地理空間索引網格系統，此系統會提供了一種將空間座標索引到平方公尺解析度的方法。索引的資料可以跨不同的資料集聯結，並以不同的精確程度彙總。H3 根據格點啟用一系列演算法和最佳化，包括最近鄰點、最短路徑、漸層平滑化等。H3 索引指的是可以是六邊形或五邊形的儲存格。該空間依照解析度被分層細分。H3 支援 0-15 的 16 個解析度，內含。0 是最普遍的，15 是最好。

Amazon Redshift 提供以下 H3 空間函數：

- [H3_FromLongLat](#)
- [H3_FromPoint](#)
- [H3_Polyfill](#)

將空間資料與 Amazon Redshift 搭配使用時的考量事項

將空間資料與 Amazon Redshift 搭配使用時的考量事項如下：

- GEOMETRY 或 GEOGRAPHY 物件的大小上限是 1,048,447 位元組。
- Amazon Redshift Spectrum 本身不支援空間資料。因此，您無法建立或改變包含 GEOMETRY 或 GEOGRAPHY 資料欄的外部資料表。
- Python 使用者定義函數（UDFs）的資料類型不支援 GEOMETRY 或 GEOGRAPHY 資料類型。
- 您無法使用 GEOMETRY 或 GEOGRAPHY 資料欄做為 Amazon Redshift 資料表的排序索引鍵或分配索引鍵。
- 您無法在 SQL ORDER BY、GROUP BY 或 DISTINCT 子句中使用 GEOMETRY 或 GEOGRAPHY 資料欄。
- 您無法在許多 SQL 函數中使用 GEOMETRY 或 GEOGRAPHY 資料欄。

- 您無法在每個格式的 GEOMETRY或 GEOGRAPHY資料欄上執行UNLOAD操作。您可以 UNLOADGEOMETRY或 GEOGRAPHY資料欄到文字或逗號分隔值（CSV）檔案。這樣做GEOGRAPHY會寫入十六進位EWKB格式GEOMETRY的資料。如果EWKB資料的大小超過 4 MB，則會發生警告，因為資料稍後無法載入資料表。
- GEOMETRY 或 GEOGRAPHY資料的支援壓縮編碼為 RAW。
- 使用 JDBC或 ODBC驅動程式時，請使用自訂類型映射。在此案例中，用戶端應用程式必須具備資訊，指出 ResultSet 物件的哪些參數是 GEOMETRY 或 GEOGRAPHY 物件。ResultSetMetadata 操作會傳回 VARCHAR 類型。
- 若要從 SHAPEFILE 複製地理日期，請先擷取至 GEOMETRY 欄中，然後將物件轉換為 GEOGRAPHY 物件。

下列非空間函數可接受 GEOMETRY 或 GEOGRAPHY 類型的輸入，或是 GEOMETRY 或 GEOGRAPHY 類型的資料欄：

- 彙總函數 COUNT
- 條件式表達式COALESCE和 NVL
- CASE 表達式
- GEOMETRY 和 的預設編碼GEOGRAPHY為 RAW。如需詳細資訊，請參閱[壓縮編碼](#)。

使用 Amazon Redshift 中的聯合查詢來查詢資料

透過在 Amazon Redshift 中使用聯合查詢，您可以跨操作資料庫、資料倉儲和資料湖查詢和分析資料。使用聯合查詢功能，您可以在外部資料庫的即時資料上將來自 Amazon Redshift 的查詢與跨 Amazon Redshift 和 Amazon S3 環境的查詢整合在一起。聯合查詢可以與 Amazon 中的外部數據庫一起使 RDS 用 PostgreSQL，Amazon Aurora PostgreSQL 兼容版本，Amazon 我的亞馬遜和 Amazon RDS Aurora 我的 SQL 的兼容版本。SQL

您可以使用聯合查詢，將即時資料整合為商業智慧 (BI) 和報告應用程式的一部分。例如，若要讓資料擷取至 Amazon Redshift 更容易，您可以使用聯合查詢來執行下列動作：

- 直接查詢操作資料庫。
- 快速套用變換。
- 將資料載入目標資料表，而不需要複雜的擷取、轉換、load (ETL) 管線。

為了減少透過網路的資料移動並改善效能，Amazon Redshift 會將聯合查詢的部分計算直接分配到遠端操作資料庫。Amazon Redshift 也會視需要使用其平行處理能力來支援執行這些查詢。

執行聯合查詢時，Amazon Redshift 會先從領導節點與 RDS 或 Aurora 資料庫叢集資料庫執行個體建立用戶端連線，以擷取表格中繼資料。從運算節點中，Amazon Redshift 會發出述詞向下推送的子查詢並擷取結果資料列。然後，Amazon Redshift 會在運算節點之間分配結果資料列，以便進一步處理。

系統檢視會記錄有關傳送至 Amazon Aurora PostgreSQL 資料庫或 Amazon RDS PostgreSQL 資料庫的查詢詳細資訊。[SVL_FEDERATED_QUERY](#)

主題

- [開始使用聯合查詢進行 PostgreSQL](#)
- [開始使用聯合查詢來發布 SQL AWS CloudFormation](#)
- [開始使用聯合查詢至 \[我的\] SQL](#)
- [建立密碼和 IAM 角色以使用聯合查詢](#)
- [使用聯合查詢的範例](#)
- [Amazon Redshift 和支持的 PostgreSQL 和我的數據庫之間的數據類型差異 SQL](#)
- [使用 Amazon Redshift 存取聯合資料時的注意事項](#)

開始使用聯合查詢進行 Postgre SQL

若要建立聯合查詢，您可以遵循以下一般方法：

1. 設定從您的 Amazon Redshift 叢集到 Amazon RDS 或 Aurora Postgre SQL 資料庫執行個體的連線。

若要這麼做，請確定您的 RDS Postgre SQL 或 Aurora Postgre SQL 資料庫執行個體可以接受來自 Amazon Redshift 叢集的連線。我們建議您的 Amazon Redshift 叢集和 Amazon RDS 或 Aurora Postgre SQL 執行個體位於相同的虛擬私有雲端 (VPC) 和子網路群組中。如此一來，您就可以將 Amazon Redshift 叢集的安全群組新增至您 RDS 或 Aurora Postgre SQL 資料庫執行個體的安全群組的輸入規則。

您也可以設定 VPC 對等互連或其他網路，讓 Amazon Redshift 與您的 RDS 或 Aurora Postgre SQL 執行個體建立連線。如需有關 VPC 網路的詳細資訊，請參閱下列內容。

- [什麼是 VPC 對等互連？](#) 在 Amazon 對 VPC 等指南
- RDS 使用 Amazon 使用者指南 VPC [中的資料庫執行個體](#)

Note

在某些情況下，您必須啟用增強型 VPC 路由：例如，如果您的 Amazon Redshift 叢集與您的 RDS 或 Aurora Postgre SQL 執行個體位於 VPC 於不同的執行個體，或者它們位於相同的路由中 VPC 且您的路由需要。否則，當您執行聯合查詢時，您可能會收到逾時錯誤。

2. AWS Secrets Manager 為您的波斯特 RDS 格雷 SQL 和 Aurora · 波斯特格雷數據庫設置秘密。SQL 然後參照 AWS Identity and Access Management (IAM) 存取原則和角色中的密碼。如需詳細資訊，請參閱 [建立密碼和 IAM 角色以使用聯合查詢](#)。

Note

如果您的叢集使用增強型 VPC 路由，您可能需 VPC 要為 AWS Secrets Manager。當 Amazon Redshift 叢集的 VPC 和子網路無法存取公有 AWS Secrets Manager 端點時，這是必要的。使用 VPC 界面端點時，VPC 和中 Amazon Redshift 叢集之間的通訊會以私密方式從您 AWS Secrets Manager 的端點介面路由 VPC 到端點界面。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的 [建立介面端點](#)。

3. 將您先前建立的 IAM 角色套用至 Amazon Redshift 叢集。如需詳細資訊，請參閱 [建立密碼和 IAM 角色以使用聯合查詢](#)。

4. 使用外部結構描述 Connect 至您的 RDS Postgre SQL 和 Aurora SQL 資料庫。如需詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#)。如需如何使用聯合查詢的範例，請參閱 [使用聯合查詢的範例](#)。
5. 執行參照參考 RDS Postgre SQL 和 Aurora Postgre 資料庫的外部結構描述的SQL查詢。SQL

開始使用聯合查詢來發布 SQL AWS CloudFormation

您可以使用聯合查詢在操作資料庫之間進行查詢。在這份入門指南中，您可以使用範例 AWS CloudFormation 堆疊啟用從 Amazon Redshift 叢集到 Aurora Postgre 無伺服器資料庫的聯合查詢來自動設定。SQL您可以快速啟動並執行，而不必執行SQL陳述式來佈建資源。

堆疊會建立外部結構描述，並參照您的 Aurora Postgre SQL 執行個體，其中包含具有範例資料的資料表。您可以從 Redshift 叢集中查詢外部結構描述中的資料表。

如果您想要透過執行SQL陳述式來設定外部結構描述來開始使用聯合查詢，而不使用 CloudFormation，請參閱 [開始使用聯合查詢進行 Postgre SQL](#)。

在執行聯合查詢的 CloudFormation 堆疊之前，請確定您已開啟資料的 Amazon Aurora Postgre SQL 相容版本無伺服器資料庫。API您可以打開數據庫屬性API中的數據。如果找不到設定，請再次確認您執行的是 Aurora Postgre 的無伺服器執行個體。SQL此外，請確定您擁有使用RA3節點的 Amazon Redshift 叢集。建議您將 Redshift 叢集和無伺服器 Aurora Postgre SQL 執行個體都位於相同的虛擬私有雲端 (VPC) 和子網路群組中。如此一來，您就可以將 Amazon Redshift 叢集的安全群組新增至 Aurora Postgre SQL 資料庫執行個體的安全群組輸入規則。

如需開始設定 Amazon Redshift 叢集的詳細資訊，請參閱 [開始使用 Amazon Redshift 佈建的資料倉儲](#)。如需有關使用設定資源的詳細資訊 CloudFormation，請參閱 [什麼是 AWS CloudFormation ?](#)。如需有關設定 Aurora 資料庫叢集資料庫的詳細資訊，請參閱 [建立 Aurora 資料庫叢集無伺服器 v1 資料庫叢集](#)。

為 Redshift 聯合查詢啟動 CloudFormation 堆疊

使用下列程序為 Amazon Redshift 啟動您的 CloudFormation 堆疊，以啟用聯合查詢。在執行此操作之前，請確定您已設定 Amazon Redshift 叢集和無伺服器 Aurora Postgre SQL 執行個體。

若要啟動同盟查詢的 CloudFormation 堆疊

1. 按一下此處 [啟動CFN堆疊](#) 以啟動中的 CloudFormation服務 AWS Management Console。

如果出現系統提示，請登入。

堆疊建立程序會啟動，參考存放在 Amazon S3 中的 CloudFormation 範本檔案。CloudFormation 模板是JSON格式的文本文件，它聲明構成堆棧的 AWS 資源。

2. 選擇下一步以輸入堆疊詳細資料。
3. 在參數底下，為叢集輸入下列內容：
 - Amazon Redshift 叢集名稱，例如 **ra3-consumer-cluster**
 - 特定的資料庫名稱，例如 **dev**
 - 資料庫使用者的名稱，例如 **consumeruser**

同時輸入 Aurora DB 叢集資料庫的參數，包括使用者、資料庫名稱、連接埠和端點。我們建議您使用測試叢集和測試無伺服器資料庫，因為堆疊會建立數個資料庫物件。

選擇下一步。

堆疊選項隨即出現。

4. 選擇下一步以接受預設設定。
5. 在 [功能] 下，選擇 [我確認 AWS CloudFormation 可能會建立IAM資源]。
6. 選擇建立堆疊。

選擇 [建立堆疊]。CloudFormation 佈建模板資源，這需要大約 10 分鐘，並創建一個外部模式。

如果建立堆疊時發生錯誤，請執行下列動作：

- 檢視 CloudFormation 事件索引標籤，瞭解可協助您解決錯誤的資訊。
- 請確定您輸入正確的 Amazon Redshift 叢集名稱、資料庫名稱和資料庫使用者名稱。同時檢查 Aurora Postgre SQL 實例的參數。
- 請確定您的叢集具有RA3節點。
- 請確定您的資料庫和 Redshift 叢集位於相同的子網路和安全群組中。

從外部結構描述查詢資料

若要使用下列程序，請確定您在所述叢集和資料庫上有執行查詢所需的許可。

使用聯合查詢來查詢外部資料庫

1. 使用用戶端工具 (例如 Redshift 查詢編輯器) 連線至您在建立堆疊時輸入的 Redshift 資料庫。
2. 查詢堆疊所建立的外部結構描述。

```
select * from svv_external_schemas;
```

[SVV_EXTERNAL_SCHEMAS](#) 檢視會傳回可用外部結構描述的相關資訊。在此情況下，會傳回堆疊所建立的外部結構描述 (myfederated_schema)。如果您有任何設定，可能還會傳回其他外部結構描述。檢視也會傳回結構描述的關聯資料庫。資料庫是您在建立堆疊時輸入的 Aurora 資料庫叢集資料庫。堆疊會將資料表新增至 Aurora DB 叢集資料庫 (稱為)category，並將另一個名為的資料表新增sales。

3. 對參考 Aurora Postgre SQL 資料庫的外部結構描述中的表格執行SQL查詢。查詢如下列範例所示。

```
SELECT count(*) FROM myfederated_schema.category;
```

category 資料表會傳回幾個記錄。您也可以從 sales 資料表中傳回記錄。

```
SELECT count(*) FROM myfederated_schema.sales;
```

如需更多範例，請參閱[使用聯合查詢的範例](#)。

開始使用聯合查詢至 [我的] SQL

若要建立「我的SQL資料庫」的聯合查詢，請遵循下列一般方法：

1. 設定從 Amazon Redshift 叢集到 Amazon RDS 或 Aurora 我的SQL資料庫執行個體的連線。

若要這麼做，請確定我的SQL或 Aurora RDS 我的SQL資料庫執行個體可以接受來自 Amazon Redshift 叢集的連線。我們建議您的 Amazon Redshift 叢集和 Amazon RDS 或 Aurora 我的SQL執行個體位於相同的虛擬私有雲端 (VPC) 和子網路群組中。如此一來，您就可以將 Amazon Redshift 叢集的安全群組新增至您RDS或 Aurora My SQL 資料庫執行個體的安全群組的輸入規則。

您也可以設定VPC對等互連或其他網路，讓 Amazon Redshift 連線到您的執行個體RDS或 Aurora 我SQL的執行個體。如需有關VPC網路的詳細資訊，請參閱下列內容。

- [什麼是VPC對等互連？](#) 在 Amazon 對VPC等指南

- RDS使用 Amazon 使用者指南VPC中的資料庫執行個體

Note

如果您的 Amazon Redshift 叢集與您的RDS或 Aurora 我的SQL執行個體不同VPC，請啟用增強型VPC路由功能。否則，當您執行聯合查詢時，您可能會收到逾時錯誤。

2. 為「我的」SQL 和「RDS我 AWS Secrets Manager 的 Aurora」SQL 資料庫設定密碼。然後參照 AWS Identity and Access Management (IAM) 存取原則和角色中的密碼。如需詳細資訊，請參閱[建立密碼和IAM角色以使用聯合查詢](#)。

Note

如果您的叢集使用增強型VPC路由，您可能需VPC要為 AWS Secrets Manager. 當 Amazon Redshift 叢集的VPC和子網路無法存取公有 AWS Secrets Manager 端點時，這是必要的。使用VPC界面端點時，VPC和中 Amazon Redshift 叢集之間的通訊會以私密方式從您 AWS Secrets Manager 的端點介面路由VPC到端點界面。如需詳細資訊，請參閱 Amazon VPC 使用者指南中的[建立介面端點](#)。

3. 將您先前建立的IAM角色套用至 Amazon Redshift 叢集。如需詳細資訊，請參閱[建立密碼和IAM角色以使用聯合查詢](#)。
4. 使用外部結構描述 Connect 到我的RDSSQL和 Aurora 我的SQL資料庫。如需詳細資訊，請參閱[CREATE EXTERNAL SCHEMA](#)。如需如何使用聯合查詢的範例，請參閱[將聯合查詢與 My 搭配使用的範例 SQL](#)。
5. 執行參考參考 RDS My SQL 和 Aurora 我的SQL資料庫的外部結構描述的SQL查詢。

建立密碼和IAM角色以使用聯合查詢

下列步驟說明如何建立密碼和IAM角色以搭配聯合查詢使用。

必要條件

請確定您具有下列必要條件，才能建立密碼和IAM角色以搭配聯合查詢使用：

- 具有使用者名稱和密碼驗證的 Po RDS stgre SQL、Aurora Postgre SQL 資料庫執行個體 SQL、RDS我的或 Aurora 我的SQL資料庫執行個體。
- 具備支援聯合查詢叢集維護版本的 Amazon Redshift 叢集。

若要使用建立密碼 (使用者名稱和密碼) AWS Secrets Manager

1. 使用擁有您RDS或 Aurora 資料庫叢集執行個體的帳戶登入 Secrets Manager 主控台。
2. 選擇儲存新機密。
3. 選擇RDS資料庫磚的「證明資料」。針對 使用者名稱和密碼，輸入您執行個體的值。確認或為加密金鑰選擇值。然後選擇您的秘密將訪問的RDS數據庫。

Note

我們建議您使用預設加密金鑰 (DefaultEncryptionKey)。如果您使用自訂加密金鑰，則用來存取密碼的IAM角色必須新增為金鑰使用者。

4. 輸入秘密的名稱，使用預設的選擇繼續建立步驟，然後選擇儲存。
5. 檢視您的密碼，並記下您為識別密碼而建立的機密ARN值。

使用秘密建立安全政策

1. 登入 AWS Management Console 並開啟IAM主控台，位於<https://console.aws.amazon.com/iam/>。
2. 建立JSON類似下列內容的策略。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessSecret",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-rds-secret-VNenFy"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
```

```
        "secretsmanager:GetRandomPassword",
        "secretsmanager:ListSecrets"
    ],
    "Resource": "*"
}
]
```

若要擷取秘密，您需要列出和讀取動作。我們建議您將資源限制在您建立的特定秘密。若要這麼做，請使用密碼的 Amazon 資源名稱 (ARN) 來限制資源。您也可以使用 IAM 主控台上的視覺化編輯器來指定權限和資源。

3. 給予政策名稱並完成建立過程。
4. 導覽至 IAM 角色。
5. 建立 Redshift-可自訂的 IAM 角色。
6. 將您剛建立的 IAM 原則附加到現有 IAM 角色，或建立新 IAM 角色並附加原則。
7. 在 IAM 角色的 [信任關係] 索引標籤上，確認角色包含信任實體 `redshift.amazonaws.com`。
8. 記下 ARN 您建立的「角色」。這 ARN 可以訪問秘密。

將 IAM 角色附加到您的 Amazon Redshift 叢集

1. 登入 AWS Management Console 並開啟 Amazon Redshift 主控台，<https://console.aws.amazon.com/redshiftv2/> 網址為。
2. 在導覽選單上，選擇叢集。會列出目前 AWS 區域中您帳戶的叢集。
3. 選擇清單中的叢集名稱以檢視該叢集的詳細資訊。
4. 對於動作，選擇管理 IAM 角色。便會顯示「管理 IAM 角色」頁面。
5. 將您的 IAM 角色新增至叢集。

使用聯合查詢的範例

下列範例顯示如何執行聯合查詢。SQL 使用連接到 Amazon Redshift 數據庫的 SQL 客戶端運行。

使用聯合查詢與 Postgre 的範例 SQL

下列範例顯示如何設定參考 Amazon Redshift 資料庫、Aurora Postgre SQL 資料庫和 Amazon S3 的聯合查詢。此範例會示範聯合查詢的運作方式。若要在您自己的環境中執行它，請將其變更為符合您的環境。如需執行此動作的先決條件，請參閱[開始使用聯合查詢進行 Postgre SQL](#)。

建立參考 Aurora Postgre SQL 資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA apg
FROM POSTGRES
DATABASE 'database-1' SCHEMA 'myschema'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-R0'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

建立另一個參考 Amazon S3 的外部結構描述，該結構描述使用 Amazon Redshift Spectrum。同時，將使用結構描述的許可授予 public。

```
CREATE EXTERNAL SCHEMA s3
FROM DATA CATALOG
DATABASE 'default' REGION 'us-west-2'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-S3';

GRANT USAGE ON SCHEMA s3 TO public;
```

顯示 Amazon Redshift 資料表中的資料列數。

```
SELECT count(*) FROM public.lineitem;

count
-----
25075099
```

在 Aurora 郵政SQL表格中顯示行數。

```
SELECT count(*) FROM apg.lineitem;

count
-----
11760
```

顯示 Amazon S3 中的資料列數。

```
SELECT count(*) FROM s3.lineitem_1t_part;

count
```

```
-----
6144008876
```

創建從 Amazon Redshift , Aurora Postgre SQL 和 Amazon S3 表的視圖。這個檢視是用來執行您的聯合查詢。

```
CREATE VIEW lineitem_all AS
  SELECT
    l_orderkey,l_partkey,l_suppkey,l_linenumber,l_quantity,l_extendedprice,l_discount,l_tax,l_returnflag,
      l_shipdate::date,l_commitdate::date,l_receiptdate::date,
    l_shipinstruct ,l_shipmode,l_comment
  FROM s3.lineitem_1t_part
  UNION ALL SELECT * FROM public.lineitem
  UNION ALL SELECT * FROM apg.lineitem
  with no schema binding;
```

搭配限制結果的述詞，顯示 lineitem_all 檢視中的資料列數。

```
SELECT count(*) from lineitem_all WHERE l_quantity = 10;

   count
-----
123373836
```

找出每年 1 月某件項目的銷售數。

```
SELECT extract(year from l_shipdate) as year,
       extract(month from l_shipdate) as month,
       count(*) as orders
FROM lineitem_all
WHERE extract(month from l_shipdate) = 1
AND l_quantity < 2
GROUP BY 1,2
ORDER BY 1,2;
```

```
 year | month | orders
-----+-----+-----
 1992 |     1 |  196019
 1993 |     1 | 1582034
 1994 |     1 | 1583181
 1995 |     1 | 1583919
```

```
1996 | 1 | 1583622
1997 | 1 | 1586541
1998 | 1 | 1583198
2016 | 1 | 15542
2017 | 1 | 15414
2018 | 1 | 15527
2019 | 1 | 151
```

使用混合大小寫名稱的範例

若要查詢具有混合大小寫名稱的資料庫、結構描述、資料表或資料行的支援 Postgre SQL 遠端資料庫，然後將設定為 `enable_case_sensitive_identifier true` 如需此工作階段參數的相關資訊，請參閱 [enable_case_sensitive_identifier](#)。

```
SET enable_case_sensitive_identifier TO TRUE;
```

一般而言，資料庫和結構描述名稱是小寫的。下列範例顯示如何連線至支援的 Postgre SQL 遠端資料庫，該資料庫的資料庫和結構描述名稱為小寫，以及資料表和資料行的大小寫混合名稱。

建立參考具有小寫 SQL 資料庫名稱 (`dblower`) 和小寫結構描述名稱 (`schemalower`) 的 Aurora Postgre 資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA apg_lower
FROM POSTGRES
DATABASE 'dblower' SCHEMA 'schemalower'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-R0'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

在執行查詢的工作階段中，將 `enable_case_sensitive_identifier` 設定為 `true`。

```
SET enable_case_sensitive_identifier TO TRUE;
```

執行聯合查詢以選取 Postgre SQL 資料庫中的所有資料。資料表 (`MixedCaseTab`) 和資料欄 (`MixedCaseName`) 具有混合大小寫的名稱。結果是一個資料列 (`Harry`)。

```
select * from apg_lower."MixedCaseTab";
```

```
MixedCaseName
```

```
-----
```

```
Harry
```

下列範例顯示如何連線至支援的 Postgre SQL 遠端資料庫，該資料庫具有資料庫、結構描述、資料表和資料行的大小寫混合名稱。

將 `enable_case_sensitive_identifier` 設定為 `true`，然後再建立外部結構描述。如果 `enable_case_sensitive_identifier` 在建立外部結構描述之前未設定為 `true`，則會發生資料庫不存在錯誤。

建立參考具有混合大小寫SQL資料庫 () 和綱要 (UpperDB) 名稱的 Aurora Postgre 資料庫的外部結構描述。UpperSchema

```
CREATE EXTERNAL SCHEMA apg_upper
FROM POSTGRES
DATABASE 'UpperDB' SCHEMA 'UpperSchema'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-R0'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

執行聯合查詢以選取 Postgre SQL 資料庫中的所有資料。資料表 (MixedCaseTab) 和資料欄 (MixedCaseName) 具有混合大小寫的名稱。結果是一個資料列 (Harry)。

```
select * from apg_upper."MixedCaseTab";
```

```
MixedCaseName
```

```
-----
```

```
Harry
```

將聯合查詢與 My 搭配使用的範例 SQL

下列範例顯示如何設定參考 Aurora My SQL 資料庫的聯合查詢。此範例會示範聯合查詢的運作方式。若要在您自己的環境中執行它，請將其變更為符合您的環境。如需執行此動作的先決條件，請參閱[開始使用聯合查詢至 \[我的\] SQL](#)。

本範例取決於下列先決條件：

- 在 Aurora 我的SQL資料庫的 Secrets Manager 中設定的秘密。此密碼會在IAM存取原則和角色中參照。如需詳細資訊，請參閱[建立密碼和IAM角色以使用聯合查詢](#)。
- 已設定連結 Amazon Redshift 和 Aurora 我的SQL安全群組。

建立參考 Aurora 我的SQL資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA amysql
FROM MYSQL
DATABASE 'functional'
URI 'endpoint to remote hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-SecretsManager-R0'
SECRET_ARN 'arn:aws:secretsmanager:us-west-2:123456789012:secret:federation/test/
dataplane-apg-creds-YbVKQw';
```

執行 Aurora 我的資料SQL表的範例SQL選取範例，以顯示 Aurora My 中的員工資料表中的一個資料列SQL。

```
SELECT level FROM amysql.employees LIMIT 1;

level
-----
      8
```

Amazon Redshift 和支持的 Postgre SQL 和我的數據庫之間的數據類型差異 SQL

下表顯示了一個 Amazon Redshift 數據類型對應到相應的 Amazon RDS Postgre SQL 或 Aurora Postgre 數據類型的映射。SQL

Amazon Redshift 資料類型	RDS波斯特SQL或 Aurora 波斯特格數SQL據類型	描述
SMALLINT	SMALLINT	帶正負號的 2 位元組整數
INTEGER	INTEGER	帶正負號的 4 位元組整數

Amazon Redshift 資料類型	RDS波斯特SQL或 Aurora 波斯特格數SQL據類型	描述
BIGINT	BIGINT	帶正負號的 8 位元組整數
DECIMAL	DECIMAL	可選擇精確度 (有效位數) 的精確數值
REAL	REAL	單精度浮點數
DOUBLE PRECISION	DOUBLE PRECISION	雙精度浮點數
BOOLEAN	BOOLEAN	邏輯布林值 (true/false)
CHAR	CHAR	固定長度的字元字串
VARCHAR	VARCHAR	可變長度的字元字串 (使用者定義的限制)
DATE	DATE	日曆日期 (年、月、日)
TIMESTAMP	TIMESTAMP	日期和時間 (未使用時區)
TIMESTAMPTZ	TIMESTAMPTZ	日期和時間 (包含時區)
GEOMETRY	郵政 GIS GEOMETRY	空間資料

以下波斯RDS格雷SQL和 Aurora 波斯特格雷SQL數據類型在 Amazon Redshift 轉換為 VARCHAR (64K) :

- JSON, JSONB
- 陣列
- BIT, BIT VARYING
- BYTEA
- 複合類型

- 日期和時間類型 INTERVALTIME , TIMEWITHTIMEZONE
- 列舉類型
- 貨幣類型
- 網路地址類型
- 數值類型 SERIALBIGSERIAL、SMALLSERIAL、和 MONEY
- 物件識別碼類型
- pg_lsn 類型
- 虛擬類型
- 範圍類型
- 文字搜尋類型
- TXID_SNAPSHOT
- UUID
- XML類型

下表顯示 Amazon Redshift 資料類型與對應的 Amazon 我的SQL或 Aurora RDS 我的SQL資料類型的對應。

Amazon Redshift 資料類型	RDS我的SQL或 Aurora 我的SQL資料類型	描述
BOOLEAN	TINYINT(-)	邏輯布林值 (true 或 false)
SMALLINT	TINYINT(UNSIGNED)	帶正負號的 2 位元組整數
SMALLINT	SMALLINT	帶正負號的 2 位元組整數
INTEGER	SMALLINT UNSIGNED	帶正負號的 4 位元組整數
INTEGER	MEDIUMINT (UNSIGNED)	帶正負號的 4 位元組整數

Amazon Redshift 資料類型	RDS我的SQL或 Aurora 我的SQL資料類型	描述
INTEGER	INT	帶正負號的 4 位元組整數
BIGINT	INT UNSIGNED	帶正負號的 8 位元組整數
BIGINT	BIGINT	帶正負號的 8 位元組整數
DECIMAL	BIGINT UNSIGNED	可選擇精確度 (有效位數) 的精確數值
DECIMAL	DECIMAL(米, D)	可選擇精確度 (有效位數) 的精確數值
REAL	FLOAT	單精度浮點數
DOUBLE PRECISION	DOUBLE	雙精度浮點數
CHAR	CHAR	固定長度的字元字串
VARCHAR	VARCHAR	可變長度的字元字串 (使用者定義的限制)
DATE	DATE	日曆日期 (年、月、日)
TIME	TIME	時間 (不含時區)
TIMESTAMP	TIMESTAMP	日期和時間 (未使用時區)
TIMESTAMP	DATETIME	時間 (不含時區)
VARCHAR(四)	YEAR	代表年份的可變長度字元

當TIME資料超出範圍時，就會產生錯誤。

在 Amazon Redshift 中，下列 RDS「我的」SQL 和「我的 Aurora」SQL 資料類型會轉換為 VARCHAR (64K)：

- BIT
- BINARY
- VARBINARY
- TINYBLOB, BLOB, MEDIUMBLOB, LONGBLOB
- TINYTEXT, TEXT, MEDIUMTEXT, LONGTEXT
- ENUM
- SET
- SPATIAL

使用 Amazon Redshift 存取聯合資料時的注意事項

某些 Amazon Redshift 功能不支援存取聯合資料。您可以在下面找到相關的限制和考量。

以下是將聯合查詢與 Amazon Redshift 搭配使用時的限制及考量：

- 聯合查詢支援讀取外部資料來源。您無法在外部資料來源中寫入或建立資料庫物件。
- 在某些情況下，您可能會在與 Amazon Redshift 不同的 AWS 區域中存取亞馬遜RDS或 Aurora 資料庫叢集資料庫。在這些情況下，跨 AWS 區域傳輸資料通常會產生網路延遲和帳單費用。我們建議您將 Aurora 全球資料庫與 Amazon Redshift 叢集位於相同 AWS 區域的本機端點搭配使用。Aurora 全域資料庫使用專用的基礎設施來進行跨越任意兩個 AWS 區域的儲存體類型複寫，其延遲一般低於 1 秒。
- 考慮存取 Amazon RDS 或 Aurora 資料庫叢集的成本。例如，使用此功能存取 Aurora 資料庫叢集時，Aurora 資料庫叢集費用是根據IOPS。
- 聯合查詢不會啟用從RDS或 Aurora 資料庫叢集存取 Amazon Redshift 的功能。
- 聯合查詢僅適用於同時提供 Amazon Redshift 和 Amazon RDS 或 Aurora 資料庫叢集的 AWS 區域。
- 聯合查詢目前不支援 ALTER SCHEMA。若要變更結構描述，請使用 DROP 再使用 CREATE EXTERNAL SCHEMA。
- 聯合查詢不會使用並行擴展。
- 聯合查詢目前不支援透過 Postgre SQL 外部資料包裝函式存取。
- RDS我SQL或 Aurora 的聯合查詢我的SQL支援READCOMMITTED層級的交易隔離。

- 如果沒有指定，Amazon Redshift 會連接到我RDS的SQL或 Aurora 我SQL的端口 3306。在為 My 建立外部結構描述之前，請先確認我的SQL連接埠號碼SQL。
- 如果未指定，Amazon Redshift 會連接到 5432 埠SQL上的波斯RDS特格雷SQL或 Aurora 波斯特格雷。在為 Postgre 建立外部結構描述之前，請先確認SQL連接埠號碼。SQL
- 從 My 獲取TIMESTAMP和DATE數據類型時SQL，零值被視為NULL。
- 如果使用 Aurora DB 叢集資料庫讀取器端點，可能會發生「無效的快照集」錯誤。這可以透過以下其中一個方法來避免：
 - 使用特定的 Aurora 資料庫叢集執行個體端點 (而非使用 Aurora 資料庫叢集端點)。此方法使用 REPEATABLE_READ事務隔離從 Postgre SQL 數據庫的結果。
 - 使用 Aurora 資料庫叢集讀取器端點，並針對工作階段設pg_federation_repeatable_read定為 false。此方法使用READCOMMITTED事務隔離從 Postgre SQL 數據庫的結果。如需 Aurora 資料庫叢集讀取器端點的詳細資訊，請參閱 Amazon [Aurora Aurora 資料庫叢集端點的類型](#) (英文)。如需 pg_federation_repeatable_read 的資訊，請參閱「[pg_federation_repeatable_read](#)」。

以下是使用 Postgre SQL 資料庫的聯合查詢時，交易的考量事項：

- 如果查詢由聯合資料表組成，則領導節點會在遠端資料庫上啟動READONLYREPEATABLE_READ交易。在 Amazon Redshift 交易期間會一直保留此交易。
- 領導者節點透過呼叫 pg_export_snapshot 來建立遠端資料庫的快照，並對受影響的表進行讀取鎖定。
- 運算節點會啟動交易，並使用在領導者節點上建立的快照，向遠端資料庫發出查詢。

支援的聯合資料庫版本

Amazon Redshift 外部結構描述可以參考外部 RDS Postgre SQL 或 Aurora 中的資料庫。SQL當其這樣做時，適用下列限制：

- 建立參考 Aurora 資料庫叢集的外部結構描述時，Aurora Postgre SQL 資料庫必須是 9.6 版或更新版本。
- 建立參考 Amazon 的外部結構描述時RDS，Amazon RDS Postgre SQL 資料庫必須是 9.6 版或更新版本。

Amazon Redshift 外部結構描述可以參考外部RDS我的SQL或 Aurora 我SQL的資料庫。當其這樣做時，適用下列限制：

- 建立參考 Aurora 資料庫叢集的外部結構描述時，Aurora 我的SQL資料庫必須是 5.6 版或更新版本。
- 建立參考 Amazon 的外部結構描述時RDS，RDS我的SQL資料庫必須是 5.6 版或更新版本。

Amazon Redshift Spectrum

本節說明如何使用 Redshift Spectrum 有效讀取 Amazon S3 中的資料。

使用 Amazon Redshift Spectrum，可以有效率地查詢 Amazon S3 裡的檔案並擷取結構化與半結構化的資料，無需將資料載入 Amazon Redshift 資料表。Redshift Spectrum 採用了大量的平行處理，以便非常快速的執行大型資料庫查詢。大部分的處理發生在 Redshift Spectrum 層，大部分資料則保留在 Amazon S3。可以多個叢集同時查詢 Amazon S3 上的相同資料集，無需為每個叢集複製資料副本。

主題

- [Amazon Redshift Spectrum 概觀](#)
- [開始使用 Amazon Redshift Spectrum](#)
- [IAM Amazon Redshift Spectrum 的政策](#)
- [Redshift Spectrum 和 AWS Lake Formation](#)
- [Amazon Redshift Spectrum 中查詢的資料檔案](#)
- [Amazon Redshift Spectrum 中的外部結構描述](#)
- [Redshift Spectrum 的外部資料表](#)
- [搭配 Amazon Redshift 使用 Apache Iceberg 資料表](#)
- [Amazon Redshift Spectrum 查詢效能](#)
- [資料處理選項](#)
- [範例：在 Redshift Spectrum 中執行相互關聯子查詢](#)
- [Amazon Redshift Spectrum 中的指標](#)
- [Amazon Redshift Spectrum 中的查詢疑難排解](#)
- [教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料](#)

Amazon Redshift Spectrum 概觀

本主題說明使用 Redshift Spectrum 有效讀取 Amazon S3 的詳細資訊。

Amazon Redshift Spectrum 位於獨立於您叢集之外的專屬 Amazon Redshift 伺服器上。Amazon Redshift 會推送許多運算密集型任務 (例如述詞篩選和彙整) 到 Redshift Spectrum 層。因此，與其他查詢相比，Redshift Spectrum 查詢使用的叢集處理容量要低得多。Redshift Spectrum 也能聰明的擴展。根據您的查詢需求，Redshift Spectrum 可能會使用數千個執行個體來運用大規模平行處理。

您可以透過定義檔案結構，並將其做為資料表註冊到外部資料目錄中來建立 Redshift Spectrum 資料表。外部資料目錄可以是 AWS Glue、Amazon Athena 隨附的資料目錄，或您自己的 Apache Hive 中繼存放區。您可以使用資料定義語言（DDL）命令，或使用連接至外部資料目錄的任何其他工具，從 Amazon Redshift 建立和管理外部資料表。您的任何 Amazon Redshift 叢集都可以立即變更外部資料目錄。

您也可以選擇在一個或多個欄上對外部資料表進行分割。將分割區定義為外部資料表的一部分可以提高效能。這種改進的出現，是因為 Amazon Redshift 查詢最佳化工具會刪除不包含查詢資料的分割區。

在您定義 Redshift Spectrum 資料表後，您可以像任何其他 Amazon Redshift 資料表一樣查詢和聯結資料表。Redshift Spectrum 不支援外部資料表的更新操作。您可以將 Redshift Spectrum 資料表新增至多個 Amazon Redshift 叢集，並從相同 AWS 區域中的任何叢集查詢 Amazon S3 上的相同資料。當您更新 Amazon S3 資料檔案時，資料可立即從任何 Amazon Redshift 叢集進行查詢。

您存取 AWS Glue 的資料目錄可能會加密，以增加安全性。如果 AWS Glue 目錄已加密，則並非所有 AWS 區域都提供的 AWS Key Management Service（AWS KMS）金鑰 AWS Glue 來存取 AWS Glue catalog。AWS Glue catalog 加密。如需支援 AWS 區域的清單，請參閱 [開發人員指南](#) 中的 [加密和安全存取 AWS Glue](#)。[AWS Glue](#) 如需 AWS Glue Data Catalog 加密的詳細資訊，請參閱 [開發人員指南](#) 中的 [加密 AWS Glue 您的資料目錄](#)。[AWS Glue](#)

Note

您無法使用與標準 Amazon Redshift 資料表相同的資源來檢視 Redshift Spectrum 資料表的詳細資訊，例如 [PG_TABLE_DEF](#)、CLASS、[STV_TBL_PERMPG](#) 或 `information_schema`。如果您的商業智慧或分析工具無法識別 Redshift Spectrum 外部資料表，請將您的應用程式設定為查詢 [SVV_EXTERNAL_TABLES](#) 與 [SVV_EXTERNAL_COLUMNS](#)。

Amazon Redshift Spectrum 區域

除非區域特定文件中另有指定，否則可在 Amazon Redshift 可用的 AWS 區域中使用 Redshift Spectrum。如需商業區域中的 AWS 區域 可用性，請參閱 [Redshift API 服務端點](#) Amazon Web Services 一般參考。

Amazon Redshift Spectrum 限制

本主題說明使用 Redshift Spectrum 的限制。

使用 Redshift Spectrum 時，請注意下列考量事項：

- Amazon Redshift 叢集和 Amazon S3 儲存貯體必須位於相同 AWS 區域。
- Redshift Spectrum 不支援VPC使用佈建叢集增強路由。若要存取您的 Amazon S3 資料，您可能需要執行額外的設定步驟。如需詳細資訊，請參閱 Amazon [Redshift 管理指南 中的 Redshift Spectrum 和增強型VPC路由](#)。
- Redshift Spectrum 支援 Amazon S3 存取點別名。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[為您的存取點使用儲存貯體型別名](#)。不過，Redshift Spectrum 不支援 VPC Amazon S3 存取點別名。如需詳細資訊，請參閱 Amazon [Redshift 管理指南 中的 Redshift Spectrum 和增強型VPC路由](#)。
- 您不能在外部資料表上執行更新或刪除操作。若要在指定的結構描述中建立新的外部資料表，您可以使用 CREATE EXTERNAL TABLE。如需 CREATE EXTERNAL 的詳細資訊TABLE，請參閱 [CREATE EXTERNAL TABLE](#)。若要將SELECT查詢結果插入外部目錄上的現有外部資料表，您可以使用 INSERT (外部資料表)。如需 INSERT (外部資料表) 的詳細資訊，請參閱 [INSERT \(外部資料表 \)](#)。
- 除非您使用已啟用 AWS Lake Formation 的 AWS Glue Data Catalog ，否則您無法控制外部資料表上的使用者許可。但是，您可以在外部結構描述上授予和撤銷許可。如需使用 AWS Lake Formation 的詳細資訊，請參閱 [Redshift Spectrum 和 AWS Lake Formation](#)。
- 要執行 Redshift Spectrum 查詢，資料庫使用者必須具有在資料庫中建立臨時資料表的許可。下列範例可在資料庫 spectrumdb 上授予臨時許可至 spectrumusers 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

如需詳細資訊，請參閱[GRANT](#)。

- 使用 Athena Data Catalog 或 AWS Glue Data Catalog 作為中繼資料存放區時，請參閱 Amazon Redshift 管理指南 中的[配額和限制](#)。
- Redshift Spectrum 不支援 Amazon EMR搭配 Kerberos。

開始使用 Amazon Redshift Spectrum

在本教學課程中，您將了解如何使用 Amazon Redshift Spectrum 直接從 Amazon S3 上的檔案查詢資料。如果您已經有叢集和SQL用戶端，則可以使用最少的設定完成本教學課程。

Note

Redshift Spectrum 查詢會產生額外費用。在本教學中執行範例查詢的成本是名目成本。如需定價的相關資訊，請參閱 [Amazon Redshift Spectrum 定價](#)。

必要條件

若要使用 Redshift Spectrum，您需要 Amazon Redshift 叢集和連線至叢集的 SQL 用戶端，才能執行 SQL 命令。叢集與 Amazon S3 中的資料檔案必須在相同的 AWS 區域。

如需有關如何建立 Amazon Redshift 叢集的資訊，請參閱 [Amazon Redshift 入門指南](#) 中的 [Amazon Redshift 佈建資料倉儲](#) 入門。如需有關如何連線至叢集的資訊，請參閱 [Amazon Redshift 入門指南](#) 中的 [連線至 Amazon Redshift 資料倉儲](#)。

在下面的一些範例中，範例資料位於美國東部 (維吉尼亞北部) 區域 (us-east-1)，因此您需要一個同樣位於 us-east-1 中的叢集。或者，您可以使用 Amazon S3 將資料物件從下列儲存貯體和資料夾複製到 AWS 區域 叢集所在的 儲存貯體：

- s3://redshift-downloads/ticket/spectrum/customers/*
- s3://redshift-downloads/ticket/spectrum/sales_partition/*
- s3://redshift-downloads/ticket/spectrum/sales/*
- s3://redshift-downloads/ticket/spectrum/salesevent/*

執行類似下列的 Amazon S3 命令，將位於美國東部 (維吉尼亞北部) 的範例資料複製到您的 AWS 區域。執行命令之前，請在儲存貯體中建立儲存貯體和資料夾，以符合 Amazon S3 複製命令。Amazon S3 複製命令的輸出會確認檔案已複製到 *bucket-name* 在您想要的 AWS 區域。

```
aws s3 cp s3://redshift-downloads/ticket/spectrum/ s3://bucket-name/ticket/spectrum/ --copy-props none --recursive
```

Redshift Spectrum 入門使用 AWS CloudFormation

除了下列步驟之外，您也可以存取 Redshift Spectrum DataLake AWS CloudFormation 範本，以使用您可以查詢的 Amazon S3 儲存貯體建立堆疊。如需詳細資訊，請參閱 [啟動您的 AWS CloudFormation 堆疊](#)，然後在 [Amazon S3](#) 中查詢您的資料。

逐步開始使用 Redshift Spectrum

若要開始使用 Amazon Redshift Spectrum，請遵循這些步驟：

- [步驟 1. 建立 Amazon Redshift IAM 的角色](#)
- [步驟 2：將 IAM 角色與叢集建立關聯](#)

- [步驟 3：建立外部結構描述與外部資料表](#)
- [步驟 4：在 Amazon S3 中查詢您的資料](#)

步驟 1. 建立 Amazon Redshift IAM 的角色

您的叢集需要授權才能存取 中的外部 Data Catalog AWS Glue 或 Amazon Athena，以及 Amazon S3 中的資料檔案。若要提供該授權，您可以參考連接至叢集的 AWS Identity and Access Management (IAM) 角色。如需搭配 Amazon Redshift 使用角色的詳細資訊，請參閱[使用IAM角色授權COPY和UNLOAD操作](#)。

Note

在某些情況下，您可以將 Athena Data Catalog 遷移至 AWS Glue Data Catalog。如果您的叢集位於 AWS Glue 支援的 AWS 區域中，且您在 Athena Data Catalog 中有 Redshift Spectrum 外部資料表，則可以執行此操作。若要搭配 Redshift Spectrum 使用 AWS Glue Data Catalog，您可能需要變更IAM政策。如需詳細資訊，請參閱《Athena 使用者指南》中的[升級至 AWS Glue 資料目錄](#)。


當您為 Amazon Redshift 建立角色時，請選擇下列其中一個方法：

- 如果您將 Redshift Spectrum 與 Athena Data Catalog 或 AWS Glue Data Catalog 搭配使用，請遵循中概述的步驟為 [Amazon Redshift 建立IAM角色](#)。
- 如果您將 Redshift Spectrum 與 AWS Glue Data Catalog 啟用的 搭配使用 AWS Lake Formation，請遵循下列程序中所述的步驟：
 - [使用 AWS Glue Data Catalog 啟用的 為 Amazon Redshift 建立IAM角色 AWS Lake Formation](#)
 - [在 Lake Formation 資料庫中授予資料表的查詢SELECT許可](#)

為 Amazon Redshift 建立IAM角色

1. 開啟[IAM主控台](#)。
2. 在導覽窗格中，選擇 Roles (角色)。
3. 選擇 Create Role (建立角色)。
4. 選擇AWS 服務做為信任的實體，然後選擇Redshift做為使用案例。
5. 在其他的 使用案例 AWS 服務下，選擇 Redshift - 可自訂，然後選擇下一步。

6. 連接許可政策頁面隨即出現。如果您使用的是 AWS Glue Data Catalog `AWSGlueConsoleFullAccess`，請選擇 `AmazonS3ReadOnlyAccess` 和 `AmazonAthenaFullAccess`。或者，如果您使用的是 Athena 資料目錄，則選擇 `AmazonAthenaFullAccess`。選擇 Next (下一步)。

 Note

`AmazonS3ReadOnlyAccess` 政策會授予您的叢集對所有 Amazon S3 儲存貯體的唯讀存取權限。若要僅授予 AWS 範例資料儲存貯體的存取權，請建立新的政策並新增下列許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Resource": "arn:aws:s3:::redshift-downloads/*"
    }
  ]
}
```

7. 在 Role name (角色名稱) 中輸入角色名稱，例如 `myspectrum_role`。
8. 檢閱資訊，然後選擇 Create role (建立角色)。
9. 在導覽窗格中，選擇角色。選擇新角色的名稱以檢視摘要，然後將角色ARN複製到剪貼簿。此值是您剛建立之角色的 Amazon Resource Name (ARN)。您可以在建立外部資料表時使用該值，以便在 Amazon S3 上參考資料檔案。

使用 AWS Glue Data Catalog 啟用的 為 Amazon Redshift 建立IAM角色 AWS Lake Formation

1. 在 開啟IAM主控台 <https://console.aws.amazon.com/iam/>。
2. 在導覽窗格中，選擇政策。

如果這是您第一次選擇 Policies (政策)，將會顯示 Welcome to Managed Policies (歡迎使用受管政策) 頁面。選擇 Get Started (開始使用)。

3. 選擇 Create policy (建立政策)。

4. 選擇在 JSON 索引標籤上建立政策。
5. 貼上下列JSON政策文件，授予對 Data Catalog 的存取權，但拒絕 Lake Formation 的管理員許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RedshiftPolicyForLF",
      "Effect": "Allow",
      "Action": [
        "glue:*",
        "lakeformation:GetDataAccess"
      ],
      "Resource": "*"
    }
  ]
}
```

6. 完成時，選擇 Review (檢閱) 以檢閱該政策。政策驗證程式會回報任何語法錯誤。
7. 在 Review policy (檢閱政策) 頁面的 Name (名稱) 中，輸入 **myspectrum_policy** 來為您所建立的政策命名。輸入 Description (說明) (選用)。檢閱政策 Summary (摘要) 來查看您的政策所授予的許可。然後選擇 Create policy (建立政策) 來儲存您的工作。

建立政策之後，您可以提供存取權給使用者。

若要提供存取權，請新增權限至您的使用者、群組或角色：

- 中的使用者和群組 AWS IAM Identity Center：

建立權限合集。請按照 AWS IAM Identity Center 使用者指南 中的 [建立權限合集](#) 說明進行操作。

- IAM 透過身分提供者在 中管理的使用者：

建立聯合身分的角色。請遵循 IAM 使用者指南 中 [為第三方身分提供者（聯合）建立角色](#) 的指示。

- IAM 使用者：

- 建立您的使用者可擔任的角色。請遵循 IAM 使用者指南 中 [為IAM使用者建立角色](#) 的指示。

- (不建議) 將政策直接附加至使用者，或將使用者新增至使用者群組。請遵循 IAM 使用者指南 中 [將許可新增至使用者（主控台）](#) 的指示。

在 Lake Formation 資料庫中授予資料表的查詢SELECT許可

1. 在開啟 Lake Formation 主控台<https://console.aws.amazon.com/lakeformation/>。
2. 在導覽窗格中，選擇資料湖許可，然後選擇授予。
3. 遵循AWS Lake Formation 開發人員指南的[使用具名資源方法授予資料表權限](#)中的指示。請提供下列資訊：
 - 針對IAM角色，選擇您建立IAM的角色 `myspectrum_role`。當您執行 Amazon Redshift 查詢編輯器時，它會使用此IAM角色來取得資料的許可。

Note

若要在已啟用 Lake Formation 的 Data Catalog 中的資料表上授予查詢的SELECT許可，請執行下列動作：

- 在 Lake Formation 中註冊資料的路徑。
- 在 Lake Formation 中授予使用者該路徑的許可。
- 建立的資料表會位於 Lake Formation 中註冊的路徑。

4. 選擇 Grant (授予)。

Important

根據最佳實務，請僅允許透過 Lake Formation 許可存取基礎 Amazon S3 物件。若要防止未經授權的存取，則可移除授予 Lake Formation 外部 Amazon S3 物件的任何許可。如果您在設定 Lake Formation 之前曾存取 Amazon S3 物件，請移除先前設定的任何IAM政策或儲存貯體許可。如需詳細資訊，請參閱[將 AWS Glue 資料許可升級至 AWS Lake Formation 模型](#)和 [Lake Formation 許可](#)。

步驟 2：將IAM角色與叢集建立關聯

現在，您有一個 IAM角色授權 Amazon Redshift 為您存取外部 Data Catalog 和 Amazon S3。此時，您必須將該角色與您的 Amazon Redshift 叢集建立關聯。

將IAM角色與叢集建立關聯

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您要更新的叢集名稱。
3. 針對動作，選擇管理IAM角色。IAM 角色頁面隨即出現。
4. 選擇輸入ARN，然後輸入 ARN或 IAM角色，或從清單中選擇IAM角色。然後選擇新增IAM角色，將其新增至已連接IAM角色的清單。
5. 選擇完成，將IAM角色與叢集建立關聯。叢集會進行修改以完成變更。

步驟 3：建立外部結構描述與外部資料表

在外部結構描述中建立外部資料表。外部結構描述參考外部資料目錄中的資料庫，並提供 IAM角色 ARN，授權您的叢集代表您存取 Amazon S3。您可以在 Amazon Athena Data Catalog AWS Glue Data Catalog或 Apache Hive 中繼存放區中建立外部資料庫，例如 Amazon EMR。在此範例中，當您建立外部結構描述 Amazon Redshift 時，在 Amazon Athena 資料目錄中建立外部資料庫。如需詳細資訊，請參閱 [Amazon Redshift Spectrum 中的外部結構描述](#)。

建立外部結構描述與外部資料表

1. 若要建立外部結構描述，請將下列命令ARN中IAM的角色取代為您在 [步驟 1](#) 中ARN建立的角色。然後在SQL用戶端中執行 命令。

```
create external schema myspectrum_schema
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
create external database if not exists;
```

2. 若要建立外部資料表，請執行下列CREATEEXTERNALTABLE命令。

Note

您的叢集和 Amazon S3 儲存貯體必須位於相同的 AWS 區域。在此範例 CREATEEXTERNALTABLE命令中，具有範例資料的 Amazon S3 儲存貯體位於美國東部（維吉尼亞北部）AWS 區域。若要查看來源資料，請下載 [sales_ts.000 檔案](#)。

您可以修改此範例，以便在不同的 中執行 AWS 區域。在所需的 中建立 Amazon S3 儲存貯體 AWS 區域。使用 Amazon S3 複製命令複製銷售資料。然後將範例 CREATE EXTERNAL TABLE 命令中的位置選項更新到您的儲存貯體。

```
aws s3 cp s3://redshift-downloads/ticket/spectrum/sales/ s3://bucket-name/
ticket/spectrum/sales/ --copy-props none --recursive
```

Amazon S3 複製命令的輸出會確認檔案已複製到 *bucket-name* 在您想要的 中 AWS 區域。

```
copy: s3://redshift-downloads/ticket/spectrum/sales/sales_ts.000 to
s3://bucket-name/ticket/spectrum/sales/sales_ts.000
```

```
create external table myspectrum_schema.sales(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/ticket/spectrum/sales/'
table properties ('numRows'='172000');
```

步驟 4：在 Amazon S3 中查詢您的資料

建立外部資料表之後，您可以使用您用來查詢其他 Amazon Redshift 資料表的相同 SELECT 陳述式來查詢它們。這些 SELECT 陳述式查詢包括聯結資料表、彙總資料，以及對述詞進行篩選。

在 Amazon S3 中查詢您的資料

1. 取得 MYSPECTRUM_ 資料表中的資料列數目 SCHEMA。SALES

```
select count(*) from myspectrum_schema.sales;
```

```
count
-----
172462
```

- 為符合最佳實務，請將較大的事實資料表存放於 Amazon S3，並將較小的維度資料表存放於 Amazon Redshift。如果您在 [Load data 中載入範例資料](#)，您的資料庫中會有一個名為 EVENT 的資料表。如果沒有，請使用下列命令建立 EVENT 資料表。

```
create table event(
eventid integer not null distkey,
venueid smallint not null,
catid smallint not null,
dateid smallint not null sortkey,
eventname varchar(200),
starttime timestamp);
```

- 將下列 COPY 命令 ARN 中 IAM 的角色取代為您在中 ARN 建立的角色，以載入 EVENT 資料表 [步驟 1. 建立 Amazon Redshift IAM 的角色](#)。您可以選擇性地從中的 [Amazon S3 儲存貯體下載和檢視的來源資料allevvents_pipe.txt](#) AWS 區域 us-east-1。Amazon S3

```
copy event from 's3://redshift-downloads/ticket/allevvents_pipe.txt'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
delimiter '|' timeformat 'YYYY-MM-DD HH:MI:SS' region 'us-east-1';
```

下列範例將外部 Amazon S3 資料表 MYSPECTRUM_SCHEMA.SALES 與本機 Amazon Redshift 資料表連結 EVENT，以尋找前 10 個事件的總銷售額。

```
select top 10 myspectrum_schema.sales.eventid,
sum(myspectrum_schema.sales.pricepaid) from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;
```

```
eventid | sum
-----+-----
      289 | 51846.00
```



```

7895 | 51049.00
1602 | 50301.00
 851 | 49956.00
7315 | 49823.00
6471 | 47997.00
2118 | 47863.00
 984 | 46780.00
7851 | 46661.00
5638 | 46280.00

```

4. 檢視先前查詢的查詢計畫。請注意針對 Amazon S3 上的資料執行的 S3 Seq Scan、S3 HashAggregate 和 S3 Query Scan 步驟。

```

explain
select top 10 myspectrum_schema.sales.eventid,
  sum(myspectrum_schema.sales.pricepaid)
from myspectrum_schema.sales, event
where myspectrum_schema.sales.eventid = event.eventid
and myspectrum_schema.sales.pricepaid > 30
group by myspectrum_schema.sales.eventid
order by 2 desc;

```

QUERY PLAN

```

-----
XN Limit (cost=1001055770628.63..1001055770628.65 rows=10 width=31)

-> XN Merge (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

    Merge Key: sum(sales.derived_col2)

-> XN Network (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

    Send to leader

```

```
    -> XN Sort (cost=1001055770628.63..1001055770629.13 rows=200
width=31)

        Sort Key: sum(sales.derived_col2)

    -> XN HashAggregate (cost=1055770620.49..1055770620.99
rows=200 width=31)

        -> XN Hash Join DS_BCAST_INNER
(cost=3119.97..1055769620.49 rows=200000 width=31)

            Hash Cond: ("outer".derived_col1 = "inner".eventid)

    -> XN S3 Query Scan sales (cost=3010.00..5010.50
rows=200000 width=31)

        -> S3 HashAggregate (cost=3010.00..3010.50
rows=200000 width=16)

            -> S3 Seq Scan myspectrum_schema.sales
location:"s3://redshift-downloads/ticket/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)

                Filter: (pricepaid > 30.00)

    -> XN Hash (cost=87.98..87.98 rows=8798 width=4)

        -> XN Seq Scan on event (cost=0.00..87.98
rows=8798 width=4)
```

啟動您的 AWS CloudFormation 堆疊，然後在 Amazon S3 中查詢您的資料

建立 Amazon Redshift 叢集並連線至叢集後，您可以安裝 Redshift Spectrum DataLake AWS CloudFormation 範本，然後查詢資料。

CloudFormation 安裝 Redshift Spectrum 入門 DataLake 範本並建立包含下列項目的堆疊：

- 名為 `myspectrum_role` 且與您的 Redshift 叢集相關聯的角色
- 名為 `myspectrum_schema` 的外部結構描述

- Amazon S3 儲存貯體中名為 sales 的外部資料表
- 名為 event 且已載入資料的 Redshift 資料表

若要啟動 Redshift Spectrum 入門 DataLake CloudFormation 堆疊

1. 選擇 [啟動CFN堆疊](#)。CloudFormation 主控台會開啟並選取 DataLake.yml 範本。

您也可以下載並自訂 Redshift Spectrum 入門 DataLake CloudFormation [CFN範本](#)，然後開啟 CloudFormation 主控台 [https://console.aws.amazon.com \(/cloudformation\)](https://console.aws.amazon.com(/cloudformation))，然後使用自訂範本建立堆疊。

2. 選擇 Next (下一步)。
3. 在參數下，輸入 Amazon Redshift 叢集名稱、資料庫名稱和您的資料庫使用者名稱。
4. 選擇 Next (下一步)。

堆疊選項隨即出現。

5. 選擇下一步以接受預設設定。
6. 檢閱 功能 下的資訊 和 ，然後選擇我確認 AWS CloudFormation 可能會建立IAM資源。
7. 選擇建立堆疊。

如果在建立堆疊時發生錯誤，請參閱下列資訊：

- 檢視 CloudFormation 事件索引標籤，以取得可協助您解決錯誤的資訊。
- 在再次嘗試操作之前，請先刪除 DataLake CloudFormation 堆疊。
- 請確定您已連線到 Amazon Redshift 資料庫。
- 請確定您為 Amazon Redshift 叢集名稱、資料庫名稱和資料庫使用者名稱輸入正確的資訊。

在 Amazon S3 中查詢您的資料

您可以使用您用來查詢其他 Amazon Redshift 資料表的相同SELECT陳述式來查詢外部資料表。這些SELECT陳述式查詢包括聯結資料表、彙總資料，以及對述詞進行篩選。

下列查詢會傳回 myspectrum_schema.sales 外部資料表中的列數。

```
select count(*) from myspectrum_schema.sales;
```

```
count
```

```
-----  
172462
```

將外部資料表與本機資料表聯結

以下範例會結合使用外部資料表 `myspectrum_schema.sales` 與本機資料表 `event`，以尋找前十名事件的總銷售額。

```
select top 10 myspectrum_schema.sales.eventid, sum(myspectrum_schema.sales.pricepaid)  
  from myspectrum_schema.sales, event  
 where myspectrum_schema.sales.eventid = event.eventid  
 and myspectrum_schema.sales.pricepaid > 30  
 group by myspectrum_schema.sales.eventid  
 order by 2 desc;
```

```
eventid | sum  
-----+-----  
    289 | 51846.00  
   7895 | 51049.00  
   1602 | 50301.00  
    851 | 49956.00  
   7315 | 49823.00  
   6471 | 47997.00  
   2118 | 47863.00  
    984 | 46780.00  
   7851 | 46661.00  
   5638 | 46280.00
```

檢視查詢計畫

檢視先前查詢的查詢計畫。請注意在 Amazon S3 的資料上執行的 S3 Seq Scan、S3 HashAggregate 和 S3 Query Scan 步驟。

```
explain  
select top 10 myspectrum_schema.sales.eventid, sum(myspectrum_schema.sales.pricepaid)  
  from myspectrum_schema.sales, event  
 where myspectrum_schema.sales.eventid = event.eventid  
 and myspectrum_schema.sales.pricepaid > 30  
 group by myspectrum_schema.sales.eventid  
 order by 2 desc;
```

QUERY PLAN

```
-----  
XN Limit (cost=1001055770628.63..1001055770628.65 rows=10 width=31)  
  
-> XN Merge (cost=1001055770628.63..1001055770629.13 rows=200 width=31)  
  
Merge Key: sum(sales.derived_col2)  
  
-> XN Network (cost=1001055770628.63..1001055770629.13 rows=200 width=31)  
  
Send to leader  
  
-> XN Sort (cost=1001055770628.63..1001055770629.13 rows=200 width=31)  
  
Sort Key: sum(sales.derived_col2)  
  
-> XN HashAggregate (cost=1055770620.49..1055770620.99 rows=200  
width=31)  
  
-> XN Hash Join DS_BCAST_INNER (cost=3119.97..1055769620.49  
rows=200000 width=31)  
  
Hash Cond: ("outer".derived_col1 = "inner".eventid)  
  
-> XN S3 Query Scan sales (cost=3010.00..5010.50  
rows=200000 width=31)  
  
-> S3 HashAggregate (cost=3010.00..3010.50  
rows=200000 width=16)
```

```
                                -> S3 Seq Scan spectrum.sales
location:"s3://redshift-downloads/ticket/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)

                                Filter: (pricepaid > 30.00)

                                -> XN Hash (cost=87.98..87.98 rows=8798 width=4)

                                -> XN Seq Scan on event (cost=0.00..87.98
rows=8798 width=4)
```

IAM Amazon Redshift Spectrum 的政策

本主題說明使用 Redshift Spectrum 所需的 IAM 許可。

根據預設，Amazon Redshift Spectrum 在支援的 AWS Glue Data Catalog AWS 區域中使用 AWS Glue。在其他 AWS 區域中，Redshift Spectrum 使用 Athena Data Catalog。您的叢集需要授權才能存取中的外部資料目錄，AWS Glue 或 Athena 和 Amazon S3 中的資料檔案。您可以參考連接至叢集的 AWS Identity and Access Management (IAM) 角色來提供該授權。如果您使用 Apache Hive 中繼存放區來管理資料目錄，則無需提供對 Athena 的存取權限。

您可以鏈結角色，以便您的叢集可以擔任未連接到叢集的其他角色。如需詳細資訊，請參閱 [Amazon Redshift Spectrum 中的鏈結 IAM 角色](#)。

您存取的 AWS Glue 目錄可能會加密，以增加安全性。如果 AWS Glue 目錄已加密，您需要 AWS KMS 金鑰 AWS Glue 才能存取 AWS Glue 資料目錄。如需詳細資訊，請參閱 [開發人員指南](#) 中的 [加密 AWS Glue 您的資料目錄](#)。 [AWS Glue](#)

主題

- [Amazon S3 許可](#)
- [跨帳戶 Amazon S3 許可](#)
- [授予或限制使用 Redshift Spectrum 存取的政策](#)
- [授予最低許可的政策](#)
- [Amazon Redshift Spectrum 中的鏈結 IAM 角色](#)
- [控制對 AWS Glue Data Catalog 的存取](#)

Amazon S3 許可

您的叢集至少需要 GET 和 Amazon S3 儲存貯體的 LIST 存取權。如果您的儲存貯體與叢集不在同一個 AWS 帳戶中，則儲存貯體也必須授權您的叢集存取資料。如需詳細資訊，請參閱[授權 Amazon Redshift 代表您存取 AWS 其他服務](#)。

Note

Amazon S3 儲存貯體無法使用限制只能從特定 VPC 端點存取的儲存貯體政策。

下列政策會授予任何 Amazon S3 儲存貯體的 GET 和 LIST 存取權。此政策允許存取 Redshift Spectrum 的 Amazon S3 儲存貯體以及 COPY 操作。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:Get*", "s3:List*"],
    "Resource": "*"
  }]
}
```

下列政策會授予 GET 並 LIST 存取名為 `amzn-s3-demo-bucket` 的 Amazon S3 儲存貯體。

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:Get*", "s3:List*"],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
  }]
}
```

跨帳戶 Amazon S3 許可

若要授予 Redshift Spectrum 存取屬於另一個 AWS 帳戶的 Amazon S3 儲存貯體中資料的權限，請將下列政策新增至 Amazon S3 儲存貯體。如需詳細資訊，請參閱[授予跨帳戶儲存貯體許可](#)。

```
{
  "Version": "2012-10-17",
```

```

"Statement": [
  {
    "Sid": "Example permissions",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::redshift-account:role/spectrumrole"
    },
    "Action": [
      "s3:GetBucketLocation",
      "s3:GetObject",
      "s3:ListMultipartUploadParts",
      "s3:ListBucket",
      "s3:ListBucketMultipartUploads"
    ],
    "Resource": [
      "arn:aws:s3:::bucketname",
      "arn:aws:s3:::bucketname/*"
    ]
  }
]
}

```

授予或限制使用 Redshift Spectrum 存取的政策

若要授予只使用 Redshift Spectrum 的 Amazon S3 儲存貯體存取權，請加入允許使用者代理程式 AWS Redshift/Spectrum 存取的條件。以下政策僅允許 Redshift Spectrum 存取 Amazon S3 儲存貯體。它不包括其他存取，例如 COPY 操作。

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:Get*", "s3:List*"],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {"StringEquals": {"aws:UserAgent": "AWS Redshift/
Spectrum"}}
  }]
}

```

同樣地，您可能想要建立允許 COPY 操作存取 IAM 的角色，但不包括 Redshift Spectrum 存取。若要這麼做，請加入拒絕使用者代理程式 **AWS Redshift/Spectrum** 存取的條件。以下政策允許存取除 Redshift Spectrum 之外的 Amazon S3 儲存貯體。


```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["s3:Get*", "s3:List*"],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {"StringNotEquals": {"aws:UserAgent": "AWS Redshift/
Spectrum"}}
  ]
}
```

授予最低許可的政策

下列政策授予將 Redshift Spectrum 與 Amazon S3 AWS Glue 和 Athena 搭配使用所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketLocation",
        "s3:GetObject",
        "s3:ListMultipartUploadParts",
        "s3:ListBucket",
        "s3:ListBucketMultipartUploads"
      ],
      "Resource": [
        "arn:aws:s3:::bucketname",
        "arn:aws:s3:::bucketname/folder1/folder2/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateDatabase",
        "glue>DeleteDatabase",
        "glue:GetDatabase",
        "glue:GetDatabases",
        "glue:UpdateDatabase",
        "glue:CreateTable",
        "glue>DeleteTable",
        "glue:BatchDeleteTable",

```

```

        "glue:UpdateTable",
        "glue:GetTable",
        "glue:GetTables",
        "glue:BatchCreatePartition",
        "glue:CreatePartition",
        "glue>DeletePartition",
        "glue:BatchDeletePartition",
        "glue:UpdatePartition",
        "glue:GetPartition",
        "glue:GetPartitions",
        "glue:BatchGetPartition"
    ],
    "Resource": [
        "*"
    ]
}
]
}

```

如果您將 Athena 用於資料目錄而非 AWS Glue，則政策需要完整 Athena 存取。下列政策會授予對 Athena 資源的存取許可。如果外部資料庫位於 Hive 中繼存放區中，則您不需要 Athena 存取權限。

```

{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": ["athena:*"],
    "Resource": ["*"]
  }]
}

```

Amazon Redshift Spectrum 中的鏈結IAM角色

當您將角色連接至叢集時，您的叢集可以擔任該角色，以 AWS Glue 代表您存取 Amazon S3、Athena 和 。如果連接至叢集的角色無法存取必要資源，您可鏈結其他角色，該角色甚至可以來自其他帳戶。您的叢集接著會暫時擔任鏈結的角色，以存取資料。您也可以用鏈結角色的方式來授予跨帳戶存取。您最多可以鏈結 10 個角色。鏈結中的每個角色都會擔任鏈結中的下一個角色，直到叢集擔任鏈結尾端的角色為止。

若要鏈結角色，您應建立角色間的信任關係。擔任其他角色的角色，必須擁有允許其擔任指定角色的許可政策。另一方面，要傳送許可的角色必須擁有信任政策，允許其將許可傳送給其他鏈結的角色。如需詳細資訊，請參閱 [Amazon Redshift 中的鏈結IAM角色](#)。

當您執行 CREATE EXTERNAL SCHEMA 命令時，您可以包含以逗號分隔的角色清單來連結角色 ARNs。

Note

鏈結的角色清單不得包含空格。

在以下範例中，MyRedshiftRole 連接到叢集。MyRedshiftRole 會擔任角色 AcmeData，它屬於帳戶 111122223333。

```
create external schema acme from data catalog
database 'acmedb' region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole,arn:aws:iam::111122223333:role/
AcmeData';
```

控制對 AWS Glue Data Catalog 的存取

如果您將 AWS Glue 用於資料目錄，則可以將精細存取控制與 IAM 政策一起套用至 AWS Glue 資料目錄。例如，您可能只想向特定 IAM 角色公開幾個資料庫和資料表。

下列各節說明各種層級的資料存取 IAM 政策 AWS Glue。

主題

- [資料庫操作的策略](#)
- [資料表操作的策略](#)
- [分割區操作的策略](#)

資料庫操作的策略

如果您想要授予使用者檢視和建立資料庫的許可，他們需要資料庫和 AWS Glue 資料目錄的存取權限。

下列範例查詢會建立資料庫。

```
CREATE EXTERNAL SCHEMA example_db
FROM DATA CATALOG DATABASE 'example_db' region 'us-west-2'
IAM_ROLE 'arn:aws:iam::redshift-account:role/spectrumrole'
```

```
CREATE EXTERNAL DATABASE IF NOT EXISTS
```

下列IAM政策提供建立資料庫所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabase",
        "glue:CreateDatabase"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:catalog"
      ]
    }
  ]
}
```

下列範例查詢會列出目前的資料庫。

```
SELECT * FROM SVV_EXTERNAL_DATABASES WHERE
databasename = 'example_db1' or databasename = 'example_db2';
```

下列IAM政策提供列出目前資料庫所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetDatabases"
      ],
    }
  ]
}
```

```

    "Resource": [
      "arn:aws:glue:us-west-2:redshift-account:database/example_db1",
      "arn:aws:glue:us-west-2:redshift-account:database/example_db2",
      "arn:aws:glue:us-west-2:redshift-account:catalog"
    ]
  }
]
}

```

資料表操作的策略

如果要提供使用者檢視、建立、卸除、變更或對資料表採取其他動作的許可，使用者便需要數種類型的存取。使用者需要自行存取資料表、其所屬的資料庫，以及目錄。

以下範例查詢會建立外部資料表。

```

CREATE EXTERNAL TABLE example_db.example_tbl0(
  col0 INT,
  col1 VARCHAR(255)
) PARTITIONED BY (part INT) STORED AS TEXTFILE
LOCATION 's3://test/s3/location/';

```

下列IAM政策提供建立外部資料表所需的最低許可。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:CreateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:catalog",
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
      ]
    }
  ]
}

```

```
]
}
```

以下每個範例查詢會列出目前的外部資料表。

```
SELECT * FROM svv_external_tables
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

```
SELECT * FROM svv_external_columns
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

```
SELECT parameters FROM svv_external_tables
WHERE tablename = 'example_tbl0' OR
tablename = 'example_tbl1';
```

下列IAM政策提供列出目前外部資料表所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetTables"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:catalog",
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/
example_tbl0",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl1"
      ]
    }
  ]
}
```

```
    ]
  }
]
```

以下範例查詢會修改現有資料表。

```
ALTER TABLE example_db.example_tbl0
SET TABLE PROPERTIES ('numRows' = '100');
```

下列IAM政策提供變更現有資料表所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetTable",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:catalog",
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
      ]
    }
  ]
}
```

以下範例查詢會捨棄現有資料表。

```
DROP TABLE example_db.example_tbl0;
```

下列IAM政策提供捨棄現有資料表所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:DeleteTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:catalog",
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
      ]
    }
  ]
}
```

分割區操作的政策

如果要提供使用者執行分割區層級操作 (檢視、建立、捨棄、變更等等) 的許可，使用者需要分割區所屬資料表的許可。他們也需要相關資料庫和 AWS Glue 資料目錄的許可。

下列範例查詢會建立分割區。

```
ALTER TABLE example_db.example_tbl0
ADD PARTITION (part=0) LOCATION 's3://test/s3/location/part=0/';
ALTER TABLE example_db.example_t
ADD PARTITION (part=1) LOCATION 's3://test/s3/location/part=1/';
```

下列IAM政策提供建立分割區所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```



```

    "Effect": "Allow",
    "Action": [
      "glue:GetTable",
      "glue:BatchCreatePartition"
    ],
    "Resource": [
      "arn:aws:glue:us-west-2:redshift-account:catalog",
      "arn:aws:glue:us-west-2:redshift-account:database/example_db",
      "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
    ]
  }
]
}

```

下列範例查詢會列出目前的分割區。

```

SELECT * FROM svv_external_partitions
WHERE schemaname = 'example_db' AND
tablename = 'example_tbl0'

```

下列IAM政策提供列出目前分割區所需的最低許可。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:GetPartitions",
        "glue:GetTables",
        "glue:GetTable"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:catalog",
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
      ]
    }
  ]
}

```

```
    }  
  ]  
}
```

以下範例查詢會修改現有分割區。

```
ALTER TABLE example_db.example_tbl0 PARTITION(part='0')  
SET LOCATION 's3://test/s3/new/location/part=0/';
```

下列IAM政策提供變更現有分割區所需的最低許可。

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "glue:GetPartition",  
        "glue:UpdatePartition"  
      ],  
      "Resource": [  
        "arn:aws:glue:us-west-2:redshift-account:catalog",  
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",  
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"  
      ]  
    }  
  ]  
}
```

以下範例查詢會捨棄現有分割區。

```
ALTER TABLE example_db.example_tbl0 DROP PARTITION(part='0');
```

下列IAM政策提供捨棄現有分割區所需的最低許可。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "glue:DeletePartition"
      ],
      "Resource": [
        "arn:aws:glue:us-west-2:redshift-account:catalog",
        "arn:aws:glue:us-west-2:redshift-account:database/example_db",
        "arn:aws:glue:us-west-2:redshift-account:table/example_db/example_tbl0"
      ]
    }
  ]
}
```

Redshift Spectrum 和 AWS Lake Formation

本主題說明如何將 Redshift Spectrum 與 Lake Formation 搭配使用。Lake Formation 是一項用於共用分析資料的服務。

您可以使用 AWS Lake Formation 集中定義和強制執行資料庫、資料表和資料欄層級存取政策，以存取存放在 Amazon S3 中的資料。透過啟用 Lake Formation 的 AWS Glue Data Catalog 註冊資料後，您即可使用 Redshift Spectrum 等數種服務來查詢該資料。

Lake Formation 提供資料目錄的安全功能和管理能力。在 Lake Formation 中，您能夠授予和撤銷資料庫、資料表、欄和基礎 Amazon S3 儲存體等資料目錄物件的許可。

Important

您只能在 Lake Formation 可用的 AWS 區域中，將 Redshift Spectrum 與已啟用 Lake Formation 的資料目錄搭配使用。如需可用區域的清單，請參閱AWS 一般參考中的[AWS Lake Formation 端點和配額](#)。

透過使用 Redshift Spectrum 搭配 Lake Formation，您可以執行以下操作：

- 使用 Lake Formation 做為集中位置，您可以在其中授予和撤銷對資料湖中所有資料的許可和存取控制許可。Lake Formation 提供許可階層，用以控制對資料目錄中資料庫和資料表的存取權。如需詳細資訊，請參閱《AWS Lake Formation 開發人員指南》中的 [Lake Formation 許可概觀](#)。
- 建立外部資料表並在資料湖中的資料上執行查詢 資料湖帳戶管理員必須先向 Lake Formation 註冊包含來源資料的現有 Amazon S3 路徑，帳戶中的使用者才可以執行查詢。管理員還需建立資料表，並授予使用者相關許可。您可授予資料庫、資料表或資料欄層級的存取權限。管理員可以使用 Lake Formation 中的資料篩選條件，對儲存在 Amazon S3 中的敏感資料授予精細的存取控制。如需詳細資訊，請參閱[使用資料篩選條件來實現列層級和儲存格層級安全](#)。

在資料目錄中註冊資料之後，每當使用者嘗試執行查詢時，Lake Formation 都會驗證該特定主體對資料表的存取權。Lake Formation 會將暫臨時憑證提供給 Redshift Spectrum，並執行查詢。

- AWS Glue Data Catalog 使用透過 GetCredentials 或 取得的IAM憑證針對自動掛載執行 Redshift Spectrum 查詢GetClusterCredentials，並管理資料庫使用者（IAMR：username 或 IAM：username）的 Lake Formation 許可。

當您將 Redshift Spectrum 搭配為 Lake Formation 啟用的資料目錄一起使用時，必須具備以下其中一項：

- 與具有資料目錄許可的叢集相關聯的IAM角色。
- 設定為管理外部資源存取的聯合IAM身分。如需詳細資訊，請參閱[使用聯合身分來管理 Amazon Redshift 對本機資源的存取和 Amazon Redshift 外部資料表的存取](#)。

Important

在已啟用 Lake Formation 的資料目錄中使用 Redshift Spectrum 時，您無法連結IAM角色。

若要進一步了解設定與 Redshift Spectrum AWS Lake Formation 搭配使用的必要步驟，請參閱 AWS Lake Formation 開發人員指南 中的[教學課程：從 Lake Formation 中的JDBC來源建立資料湖](#)。具體而言，請參閱[使用 Amazon Redshift Spectrum 查詢資料湖中的資料](#)，以了解與 Redshift Spectrum 整合的詳細資訊。本主題中使用的資料 AWS 和資源取決於教學課程中的先前步驟。

使用資料篩選條件來實現列層級和儲存格層級安全

您可以在 中定義資料篩選條件 AWS Lake Formation，以控制 Redshift Spectrum 查詢的資料列層級和儲存格層級對資料目錄中定義資料的存取。若要進行此設定，您需要執行以下任務：

- 使用以下資訊在 Lake Formation 中建立資料篩選條件：
 - 欄規格，內含要包含在查詢結果中或從查詢結果中排除的欄清單。
 - 列篩選運算式，指定要包含在查詢結果中的列。

如需如何建立資料篩選條件的相關資訊，請參閱《AWS Lake Formation 開發人員指南》中的 [Lake Formation 中的資料篩選條件](#)。

- 在 Amazon Redshift 中建立外部資料表，該資料表會參考已啟用 Lake Formation 的資料目錄中的資料表。如需如何使用 Redshift Spectrum 查詢 Lake Formation 資料表的詳細資訊，請參閱《AWS Lake Formation 開發人員指南》中的 [使用 Amazon Redshift Spectrum 查詢資料湖中的資料](#)。

在 Amazon Redshift 中定義資料表之後，您可以查詢 Lake Formation 資料表，以及存取僅限資料篩選條件允許的列和欄。

如需如何在 Lake Formation 中設定列層級和儲存格層級安全，然後使用 Redshift Spectrum 進行查詢的詳細指南，請參閱 [使用 Amazon Redshift Spectrum 搭配 AWS Lake Formation 中定義的列層級和儲存格層級安全政策](#)。

Amazon Redshift Spectrum 中查詢的資料檔案

本節說明如何以 Redshift Spectrum 支援的格式在 Amazon S3 中建立資料檔案。

您在 Amazon Redshift Spectrum 中用於查詢的資料檔案，通常與您用於其他應用程式的檔案為相同類型。例如，相同的檔案類型會與 Amazon Athena、Amazon EMR 和 Amazon 搭配使用 QuickSight。您可以直接從 Amazon S3 查詢原始格式的資料。若要執行此作業，資料檔案的格式必須是 Redshift Spectrum 支援的格式，且位於您叢集可存取的 Amazon S3 儲存貯體中。

具有資料檔案的 Amazon S3 儲存貯體和 Amazon Redshift 叢集必須位於相同 AWS 區域。如需有關支援 AWS 區域的資訊，請參閱 [Amazon Redshift Spectrum 區域](#)。

Redshift Spectrum 的資料格式

Redshift Spectrum 支援下列結構化與半結構化的資料格式。

檔案格式	單欄式	支援平行讀取	分割單位
Parquet	是	是	資料列群組

檔案格式	單欄式	支援平行讀取	分割單位
ORC	是	是	Stripe
RcFile	是	是	資料列群組
TextFile	否	是	Row
SequenceFile	否	是	資料列或區塊
RegexSerde	否	是	Row
開啟CSV	否	是	Row
AVRO	否	是	區塊
Ion	否	否	不適用
JSON	否	否	N/A

在上表中，標題指出下列項目：

- 單欄式 – 檔案格式是否實體上以資料欄導向的結構儲存資料，而不是以資料列導向的結構。
- 支援平行讀取 – 檔案格式是否支援讀取檔案內的個別區塊。讀取個別區塊可以跨多個獨立 Redshift Spectrum 請求分配檔案的處理，而不必在單一請求中讀取完整檔案。
- 分割單位 – 對於可平行讀取的檔案格式，分割單位是單一 Redshift Spectrum 請求可處理的最小資料區塊。

Note

文字檔案中的時間戳記值格式必須為 `yyyy-MM-dd HH:mm:ss.SSSSSS`，如以下的時間戳記值所示：`2017-05-01 11:30:59.000000`。

我們建議使用例如 Apache Parquet 的單欄式儲存檔案格式。使用單欄式儲存檔案格式，您只需選擇所需的資料欄，即可最大程度地減少 Amazon S3 中的資料傳輸。

Redshift Spectrum 的壓縮類型

為減少儲存空間，提高效能並降低成本，我們強烈建議您壓縮您的資料檔案。Redshift Spectrum 會根據副檔名識別檔案壓縮類型。

Redshift Spectrum 支援下列壓縮類型與副檔名。

壓縮演算法	副檔名	支援平行讀取
Gzip	.gz	否
Bzip2	.bz2	是
Snappy	.snappy	否

您可以套用不同層級的壓縮。最常見的是，您可以壓縮整個檔案或壓縮檔案中的個別區塊。在檔案層級壓縮單欄式格式不會產生效能優勢。

為了使 Redshift Spectrum 能夠平行讀取檔案，下列條件必須成立：

- 檔案格式支援平行讀取。
- 檔案層級壓縮 (如果有的話) 支援平行讀取。

檔案內的個別分割單位是否使用可平行讀取的壓縮演算法進行壓縮並不重要，因為每個分割單位都是由單一 Redshift Spectrum 請求處理。此情況的範例是 Snappy 壓縮的 Parquet 檔案。Parquet 檔案中的個別資料列群組會使用 Snappy 進行壓縮，但檔案的頂層結構仍會保持未壓縮狀態。在此情況下，可以平行讀取檔案，因為每個 Redshift Spectrum 請求都可以從 Amazon S3 中讀取和處理個別列群組。

Redshift Spectrum 的加密

Redshift Spectrum 可透明的解密使用以下加密選項加密的資料檔案：

- 伺服器端加密 (SSE-S3)，使用 Amazon S3 管理的 AES-256 加密金鑰。
- 使用 AWS Key Management Service (SSE-) 管理的金鑰進行伺服器端加密 KMS。

Redshift Spectrum 不支援 Amazon S3 用戶端加密。如需伺服器端加密的相關資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[使用伺服器端加密保護資料](#)。

Amazon Redshift 使用大規模平行處理 (MPP) 來快速執行操作於大量資料的複雜查詢。Redshift Spectrum 擴展了相同的原則來查詢外部資料，並依需要使用多個 Redshift Spectrum 執行個體來掃描檔案。將檔案放在每個資料表的單獨資料夾中。

您可以透過執行下列動作來最佳化資料以進行平行處理：

- 如果您的檔案格式或壓縮不支援平行讀取，請將大型檔案分成許多較小的檔案。建議您使用介於 64 MB 與 1 GB 之間的檔案大小。
- 將所有檔案維持相同大小。如果某些檔案比其他檔案大得多，則 Redshift Spectrum 無法平均分配工作負載。

Amazon Redshift Spectrum 中的外部結構描述

本主題說明如何透過 Redshift Spectrum 建立和使用外部結構描述。外部結構描述是您用來作為參考的資料表集合，用於存取 Amazon Redshift 叢集外部的資料。這些資料表包含 Redshift Spectrum 讀取之外部資料的中繼資料。

所有外部資料表必須建立在使用 [CREATE EXTERNAL SCHEMA](#) 陳述式建立的外部結構描述中。

Note

某些應用程式會交互使用 database (資料庫) 和 schema (結構描述) 詞彙。在 Amazon Redshift，我們使用術語「結構描述」。

Amazon Redshift 外部結構描述參考了外部資料目錄中的外部資料庫。您可以在 Amazon Redshift、[Amazon Athena](#)、[AWS Glue Data Catalog](#) 或 Amazon 等 Apache Hive 中繼存放區中建立外部資料庫 [EMR](#)。若您在 Amazon Redshift 中建立了外部資料庫，該資料庫會位於 Athena 資料目錄中。若要在 Hive 中繼存放區中建立資料庫，您需要在 Hive 應用程式中建立資料庫。

Amazon Redshift 需要授權才能代表您存取 Athena 中的資料目錄以及 Amazon S3 中的資料檔案。若要提供該授權，請先建立 AWS Identity and Access Management (IAM) 角色。然後將角色連接至叢集，並在 Amazon Redshift `CREATE EXTERNAL SCHEMA` 陳述式中提供角色的 Amazon Resource Name (ARN)。如需授權的相關資訊，請參閱 [IAM Amazon Redshift Spectrum 的政策](#)。

Note

如果您目前在 Athena Data Catalog 中有 Redshift Spectrum 外部資料表，您可以將 Athena Data Catalog 遷移至 AWS Glue Data Catalog。若要搭配 Redshift Spectrum 使用 AWS Glue

Data Catalog，您可能需要變更IAM政策。如需詳細資訊，請參閱 Amazon Athena 使用者指南中的[升級至 AWS Glue 資料目錄](#)。

若要在建立外部結構描述的同時建立外部資料庫，請指定 FROM DATA CATALOG 並在您的 CREATE EXTERNAL DATABASE 陳述式中包含 CREATE EXTERNAL SCHEMA 子句。

以下範例使用外部資料庫 spectrum_schema 建立名為 spectrum_db 的外部結構描述。

```
create external schema spectrum_schema from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
create external database if not exists;
```

如果使用 Athena 管理資料目錄，請指定 Athena 資料庫名稱和 Athena 資料目錄所在的 AWS 區域。

以下範例使用 Athena 資料目錄中的預設 sampledb 資料庫建立外部結構描述。

```
create external schema athena_schema from data catalog
database 'sampledb'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
region 'us-east-2';
```

Note

region 參數參考 Athena Data Catalog 所在的 AWS 區域，而不是 Amazon S3 中資料檔案的位置。

如果您使用 Hive 中繼存放區管理資料目錄，例如 Amazon EMR，則必須設定安全群組以允許叢集之間的流量。

在 CREATEEXTERNALSCHEMA 陳述式中，指定 FROM HIVE METASTORE 並包含中繼存放區 URI 和連接埠號碼。以下範例使用名為 hive_db 的 Hive 中繼存放區資料庫建立外部結構描述。

```
create external schema hive_schema
from hive metastore
database 'hive_db'
uri '172.10.10.10' port 99
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
```

若要檢視叢集的外部結構描述，請查詢 PG_EXTERNAL_SCHEMA 目錄表或 SVV_EXTERNAL_SCHEMAS 檢視。下列範例查詢 SVV_EXTERNAL_SCHEMAS，其會聯結 PG_EXTERNAL_SCHEMA 和 PG_NAMESPACE。

```
select * from svv_external_schemas
```

如需完整的命令語法與範例，請參閱 [CREATE EXTERNAL SCHEMA](#)。

在 Amazon Redshift Spectrum 中使用外部目錄

Amazon Redshift Spectrum 外部資料庫和外部資料表的中繼資料儲存於外部資料目錄中。Redshift Spectrum 中繼資料預設儲存於 Athena 資料目錄中。您可在 Athena 主控台中檢視並管理 Redshift Spectrum 資料庫和資料表。

您也可以使用 Athena 或 Hive 中繼存放區，例如 Amazon，使用 Hive 資料定義語言（DDL）建立和管理外部資料庫和外部資料表 EMR。

Note

我們建議使用 Amazon Redshift 來建立並管理 Redshift Spectrum 中的外部資料庫和外部資料表。

檢視 Athena 中的 Redshift Spectrum 資料庫和 AWS Glue

您可以透過將 IF CREATE EXTERNAL DATABASE NOT EXISTS 子句作為 CREATE EXTERNAL SCHEMA 陳述式的一部分來建立外部資料庫。在這類案例中，外部資料庫的中繼資料會儲存在您的資料目錄內。您透過外部結構描述所建立的外部資料表中繼資料，也會儲存在資料目錄中。

Athena AWS Glue 和維護每個支援的資料目錄 AWS 區域。若要檢視資料表中繼資料，請登入 Athena 或 AWS Glue 主控台。在 Athena 中，選擇資料來源、您的 AWS Glue，然後檢視資料庫的詳細資訊。在中 AWS Glue，選擇資料庫、您的外部資料庫，然後檢視資料庫的詳細資訊。

如果您使用 Athena 建立和管理外部資料表，請使用 CREATE EXTERNAL 註冊資料庫 SCHEMA。例如，下列命令可註冊名為 sampledb 的 Athena 資料庫。

```
create external schema athena_sample
from data catalog
database 'sampledb'
```

```
iam_role 'arn:aws:iam::123456789012:role/mySpectrumRole'
region 'us-east-1';
```

當您查詢 SVV_EXTERNAL_TABLES 系統檢視時，您會看到 Athena sampledb 資料庫中的資料表，以及您在 Amazon Redshift 中建立的資料表。

```
select * from svv_external_tables;
```

schemaname	tablename	location
athena_sample	elb_logs	s3://athena-examples/elb/plaintext
athena_sample	lineitem_1t_csv	s3://myspectrum/tpch/1000/lineitem_csv
athena_sample	lineitem_1t_part	s3://myspectrum/tpch/1000/lineitem_partition
spectrum	sales	s3://redshift-downloads/ticket/spectrum/sales
spectrum	sales_part	s3://redshift-downloads/ticket/spectrum/sales_part

註冊 Apache Hive 中繼存放區資料庫

如果您在 Apache Hive 中繼存放區中建立外部資料表，您可以使用 CREATE EXTERNAL SCHEMA 在 Redshift Spectrum 中註冊這些資料表。

在 CREATEEXTERNALSCHEMA 陳述式中，指定 FROMHIVEMETASTORE 子句並提供 Hive 中繼存放區 URI 和連接埠號碼。此 IAM 角色必須包含存取 Amazon S3 的許可，但不需要任何 Athena 許可。以下為註冊 Hive 中繼存放區的範例。

```
create external schema if not exists hive_schema
from hive metastore
database 'hive_database'
uri 'ip-10-0-111-111.us-west-2.compute.internal' port 9083
iam_role 'arn:aws:iam::123456789012:role/mySpectrumRole';
```

讓您的 Amazon Redshift 叢集存取您的 Amazon EMR 叢集

如果您的 Hive 中繼存放區位於 Amazon 中 EMR，您必須讓 Amazon Redshift 叢集存取您的 Amazon EMR 叢集。若要這麼做，您可以建立 Amazon EC2 安全群組。然後，您可以允許來自 Amazon

Redshift 叢集EC2的安全群組和 Amazon EMR叢集的安全群組的所有傳入流量進入安全群組。然後將 EC2安全性新增至 Amazon Redshift 叢集和 Amazon EMR叢集。

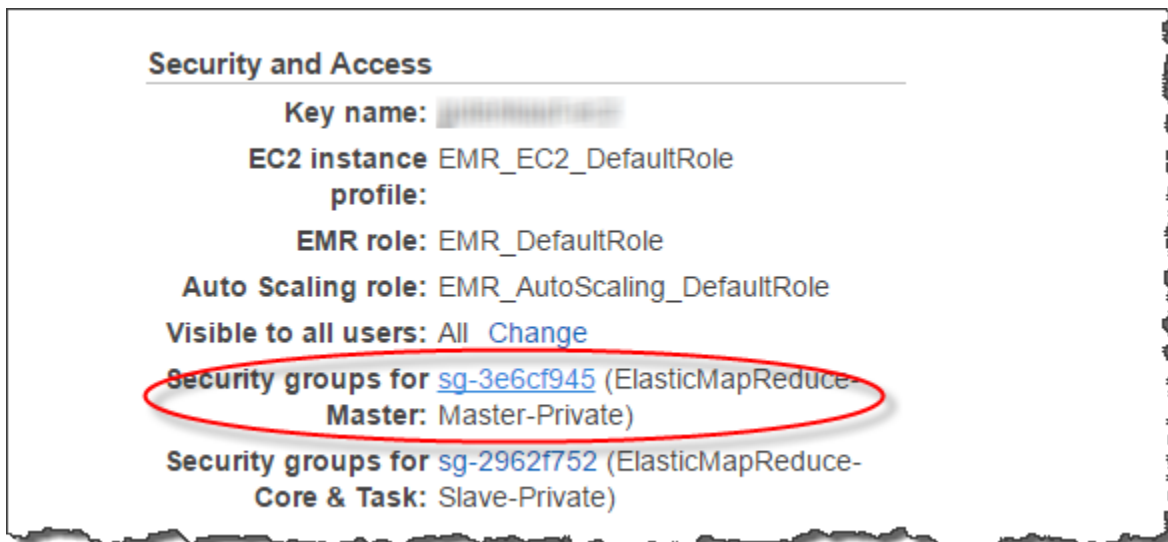
檢視您的 Amazon Redshift 叢集的安全群組名稱

如要顯示安全群組，請執行以下作業：

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後從清單選擇叢集以開啟其詳細資訊。
3. 選擇屬性，然後檢視網路與安全設定區段。
4. 在安全群組中尋找您的VPC安全群組，並將其記下。

檢視 Amazon EMR主節點安全群組名稱


1. 開啟您的 Amazon EMR叢集。如需詳細資訊，請參閱 Amazon EMR管理指南 中的[使用安全組態來設定叢集安全性](#)。
2. 在安全和存取下，記下 Amazon EMR主節點安全群組名稱。



若要建立或修改 Amazon EC2安全群組，以允許 Amazon Redshift 與 Amazon 之間的連線 EMR

1. 在 Amazon EC2儀表板中，選擇安全群組。如需詳細資訊，請參閱 Amazon EC2使用者指南中的[安全群組規則](#)
2. 選擇建立安全群組。

3. 如果您使用的是 VPC，請選擇 VPC Amazon Redshift 和 Amazon EMR 叢集所在的。
4. 新增傳入規則。
 1. 對於類型，選擇自訂 TCP。
 2. 對於 Source (資源)，選擇 Custom (自訂)。
 3. 輸入您 Amazon Redshift 安全群組的名稱。
5. 新增另一個傳入規則。
 1. 針對類型，選擇 TCP。
 2. 針對 Port Range (連接埠範圍)，輸入 9083。

 Note

的預設連接埠 EMRHMS 為 9083。如果您的 HMS 使用不同的連接埠，請在傳入規則和外部結構描述定義中指定該連接埠。

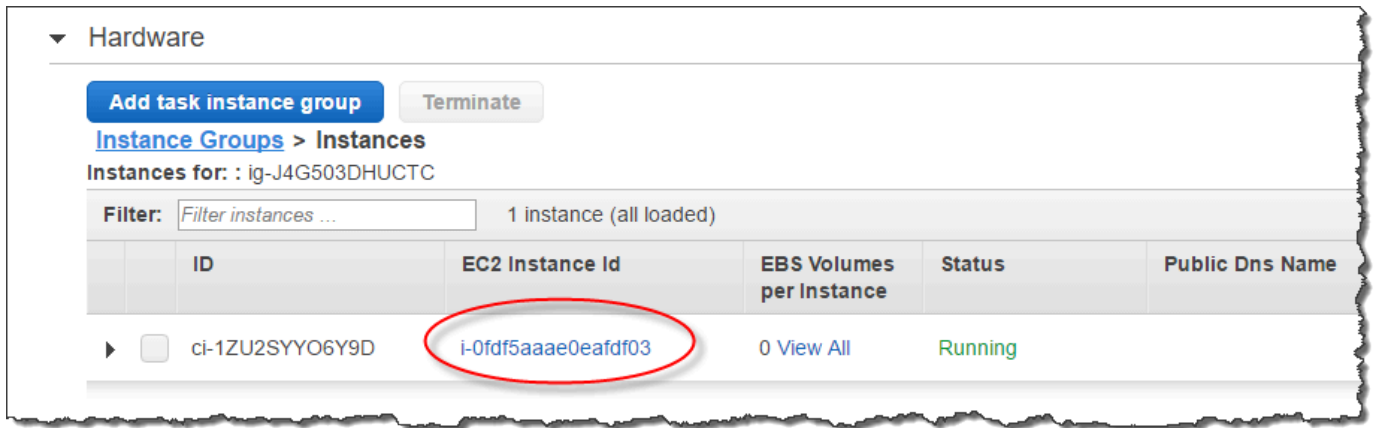
3. 對於 Source (資源)，選擇 Custom (自訂)。
6. 輸入安全群組的名稱和說明。
7. 選擇建立安全群組。

若要將您在上一個程序中建立的 Amazon EC2 安全群組新增至 Amazon Redshift 叢集

1. 在 Amazon Redshift 中，選擇您的叢集。
2. 選擇 Properties (屬性)。
3. 檢視網路與安全設定，然後選擇編輯。
4. 在 VPC 安全群組 中，選擇新的安全群組名稱。
5. 選擇 Save changes (儲存變更)。

將 Amazon EC2 安全群組新增至您的 Amazon EMR 叢集

1. 在 Amazon 中 EMR，選擇叢集。如需詳細資訊，請參閱 Amazon EMR 管理指南 中的 [使用安全組態來設定叢集安全](#)。
2. 在 Hardware (硬體) 下，選擇主節點的連結。
3. 在 EC2 執行個體 ID 欄中選擇連結。



4. 對於動作，選擇安全性、變更安全群組。
5. 在關聯的安全群組中，選擇新的安全群組，然後選擇新增安全群組。
6. 選擇 Save (儲存)。

Redshift Spectrum 的外部資料表

本主題說明如何透過 Redshift Spectrum 建立和使用外部資料表。外部資料表是您用來存取 Amazon Redshift 叢集外部資料的參考資料表。這些資料表包含 Redshift Spectrum 讀取的外部資料的中繼資料。

您會在外部結構描述中建立外部資料表。您必須為外部結構描述的擁有者或超級使用者，始可建立外部資料表。若要轉移外部結構描述的所有權，請使用 [ALTER SCHEMA](#) 來變更擁有者。下列範例會將 spectrum_schema 結構描述的擁有者變更為 newowner。

```
alter schema spectrum_schema owner to newowner;
```

若要執行 Redshift Spectrum 查詢，您需要以下許可：

- 結構描述使用許可
- 在目前資料庫建立暫時資料表的許可

下列範例可在結構描述 spectrum_schema 上授予使用許可至 spectrumusers 使用者群組。

```
grant usage on schema spectrum_schema to group spectrumusers;
```

下列範例可在資料庫 spectrumdb 上授予臨時許可至 spectrumusers 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

您可以在 Amazon Redshift、AWS Glue Amazon Athena 或 Apache Hive 中繼存放區中建立外部資料表。如需詳細資訊，請參閱 AWS Glue 開發人員指南 中的[開始使用 AWS Glue](#)、Amazon Athena 使用者指南 中的[入門](#)，或 Amazon EMR開發人員指南 中的[Apache Hive](#)。

如果您的外部資料表在 AWS Glue、Athena 或 Hive 中繼存放區中定義，則您首先會建立參考外部資料庫的外部結構描述。然後，您可以參考SELECT陳述式中的外部資料表，方法是在資料表名稱前面加上結構描述名稱，而不需要在 Amazon Redshift 中建立資料表。如需詳細資訊，請參閱[Amazon Redshift Spectrum 中的外部結構描述](#)。

若要允許 Amazon Redshift 檢視 中的資料表 AWS Glue Data Catalog，請glue:GetTable新增至 Amazon Redshift IAM角色。否則，您可能會得到如下的錯誤：

```
RedshiftIamRoleSession is not authorized to perform: glue:GetTable on resource: *;
```

例如，假設您在 Athena 外部目錄中定義了名為 lineitem_athena 的外部資料表。在此情況下，您可以定義名為 的外部結構描述athena_schema，然後使用下列SELECT陳述式查詢資料表。

```
select count(*) from athena_schema.lineitem_athena;
```

若要在 Amazon Redshift 中定義外部資料表，請使用 [CREATE EXTERNAL TABLE](#) 命令。外部資料表陳述句定義了資料表欄、資料檔案格式，以及 Amazon S3 中的資料位置。Redshift Spectrum 會掃描指定資料夾以及任何子資料夾裡的檔案。Redshift Spectrum 會忽略以句號、底線或井號 (、_ 或 #) 開頭，或以波狀符號 (~) 結尾的檔案和隱藏檔案。

下列範例會在名為 的 Amazon Redshift 外部結構描述SALES中建立名為 的資料表spectrum。該資料位於 Tab 鍵分隔的文字檔案中。

```
create external table spectrum.sales(  
  salesid integer,  
  listid integer,  
  sellerid integer,  
  buyerid integer,  
  eventid integer,  
  dateid smallint,  
  qtysold smallint,  
  pricepaid decimal(8,2),
```

```
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t'
stored as textfile
location 's3://redshift-downloads/ticket/spectrum/sales/'
table properties ('numRows'='172000');
```

若要檢視外部資料表，請查詢 [SVV_EXTERNAL_TABLES](#) 系統畫面。

虛擬資料欄

Amazon Redshift 預設會以虛擬資料欄 `$path`、`$size` 和 `$spectrum_oid` 建立外部資料表。選擇 `$path` 欄可檢視 Amazon S3 上資料檔案的路徑，選擇 `$size` 欄可檢視查詢傳回的每列資料檔案大小。`$spectrum_oid` 欄提供使用 Redshift Spectrum 執行關聯查詢的能力。如需範例，請參閱 [範例：在 Redshift Spectrum 中執行相互關聯子查詢](#)。您必須以雙引號分隔 `$path`、`$size` 和 `$spectrum_oid` 欄名稱。SELECT * 子句不會傳回虛擬欄。您必須在查詢中明確包含 `$path`、`$size` 和 `$spectrum_oid` 欄名稱，如以下範例所示。

```
select "$path", "$size", "$spectrum_oid"
from spectrum.sales_part where saledate = '2008-12-01';
```

您可以藉由將 `spectrum_enable_pseudo_columns` 組態參數設定為 `false`，以停用工作階段的虛擬資料欄建立。如需詳細資訊，請參閱 [spectrum_enable_pseudo_columns](#)。您也可以將 `enable_spectrum_oid` 設定為 `false`，僅停用 `$spectrum_oid` 虛擬資料欄。如需詳細資訊，請參閱 [enable_spectrum_oid](#)。不過，停用 `$spectrum_oid` 虛擬資料欄也會停用對 Redshift Spectrum 相關查詢的支援。

Important

選擇 `$size`、`$path` 或 `$spectrum_oid` 會產生費用，因為 Redshift Spectrum 會掃描 Amazon S3 上的資料檔案以判斷結果集的大小。如需詳細資訊，請參閱 [Amazon Redshift 定價](#)。

虛擬資料欄範例

下列範例會傳回外部資料表相關的資料檔案大小總和。


```
select distinct "$path", "$size"
from spectrum.sales_part;
```

\$path	\$size
s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01/	1616
s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-02/	1444
s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-03/	1644

分割 Redshift Spectrum 外部資料表

當您分割資料時，可以在分割區索引鍵上進行篩選來限制 Redshift Spectrum 掃描的資料量。您可透過任何索引鍵來分割您的資料。

常見做法是根據時間對資料進行分割。例如，您可以選擇依年、月、日和小時來進行分割。如果您有來自多個來源的資料，則可以按資料來源識別碼和日期進行分割。

下列程序說明如何分割您的資料。

分割您的資料

1. 根據分割區索引鍵將資料儲存在 Amazon S3 的資料夾中。

為每個分割區值建立一個資料夾，並使用分割區索引鍵和值命名該資料夾。例如，若您要依日期分割資料，您可能有名為 saledate=2017-04-01、saledate=2017-04-02 的資料夾，依此類推。Redshift Spectrum 會掃描分割區資料夾以及任何子資料夾裡的檔案。Redshift Spectrum 會忽略以句號、底線或井號 (.、_ 或 #) 開頭，或以波狀符號 (~) 結尾的檔案和隱藏檔案。

2. 建立外部資料表，並在 PARTITIONED BY 子句中指定分割區金鑰。

分割區索引鍵不得為資料表欄位的名稱。資料類型可以是 SMALLINT、INTEGER、BIGINT、DECIMAL、REALDOUBLEPRECISION、BOOLEAN、CHAR、DATE 或 TIMESTAMP 資料類型。

3. 新增分割區。

使用 [ALTER TABLE ... ADD PARTITION](#) 新增每個分割區、指定分割區資料欄和索引鍵值，以及 Amazon S3 中分割區資料夾的位置。您可以在單一 ... ALTER TABLE ADD 陳述式中新增多個分割區。以下範例將為 '2008-01' 與 '2008-03' 新增分割區。

```
alter table spectrum.sales_part add
partition(saledate='2008-01-01')
```

```
location 's3://redshift-downloads/tickit/spectrum/sales_partition/  
saledate=2008-01/'  
partition(saledate='2008-03-01')  
location 's3://redshift-downloads/tickit/spectrum/sales_partition/  
saledate=2008-03/';
```

Note

如果您使用 AWS Glue 目錄，則可以使用單一 ALTER TABLE 陳述式新增最多 100 個分割區。

分割資料範例

在此範例中，您將建立一個由單一分割區索引鍵值分割的外部資料表，以及由兩個索引鍵值所分割的外部資料表。

此範例的範例資料位於 Amazon S3 儲存貯體中，該儲存貯體提供所有已驗證 AWS 使用者的讀取存取權。您的叢集與外部資料檔案必須在相同的 AWS 區域。範例資料儲存貯體位於美國東部 (維吉尼亞北部) 區域 (us-east-1)。若要使用 Redshift Spectrum 存取資料，您的叢集必須也在 us-east-1 中。若要列出 Amazon S3 中的資料夾，請執行下列命令。

```
aws s3 ls s3://redshift-downloads/tickit/spectrum/sales_partition/
```

```
PRE saledate=2008-01/  
PRE saledate=2008-03/  
PRE saledate=2008-04/  
PRE saledate=2008-05/  
PRE saledate=2008-06/  
PRE saledate=2008-12/
```

如果您沒有外部結構描述，請執行下列命令。將 Amazon Resource Name (ARN) 替換為您的 AWS Identity and Access Management (IAM) 角色。

```
create external schema spectrum  
from data catalog  
database 'spectrumdb'  
iam_role 'arn:aws:iam::123456789012:role/myspectrumrole'  
create external database if not exists;
```

範例 1：透過單一分割區索引鍵進行分割

在下列範例中，您會建立以月份進行分割的外部資料表。

若要建立以月份分割的外部資料表，請執行以下命令。

```
create external table spectrum.sales_part(  
  salesid integer,  
  listid integer,  
  sellerid integer,  
  buyerid integer,  
  eventid integer,  
  dateid smallint,  
  qtysold smallint,  
  pricepaid decimal(8,2),  
  commission decimal(8,2),  
  saletime timestamp)  
partitioned by (saledate char(10))  
row format delimited  
fields terminated by '|'   
stored as textfile  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/'  
table properties ('numRows'='172000');
```

若要新增分割區，請執行下列ALTER TABLE 命令。

```
alter table spectrum.sales_part add  
partition(saledate='2008-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01/'  
  
partition(saledate='2008-03')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-03/'  
  
partition(saledate='2008-04')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-04/';
```

若要從分割資料表選取資料，請執行下列查詢。

```
select top 5 spectrum.sales_part.eventid, sum(spectrum.sales_part.pricepaid)  
from spectrum.sales_part, event  
where spectrum.sales_part.eventid = event.eventid  
and spectrum.sales_part.pricepaid > 30
```

```
and saledate = '2008-01'
group by spectrum.sales_part.eventid
order by 2 desc;
```

```
eventid | sum
-----+-----
  4124 | 21179.00
  1924 | 20569.00
  2294 | 18830.00
  2260 | 17669.00
  6032 | 17265.00
```

若要檢視外部資料表分割區，請查詢 [SVV_EXTERNAL_PARTITIONS](#) 系統畫面。

```
select schemaname, tablename, values, location from svv_external_partitions
where tablename = 'sales_part';
```

```
schemaname | tablename | values | location
-----+-----+-----
+-----+-----+-----
spectrum | sales_part | ["2008-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-01
spectrum | sales_part | ["2008-03"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-03
spectrum | sales_part | ["2008-04"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-04
```

範例 2：透過多個分割區索引鍵進行分割

若要建立以 date 與 eventid 分割的外部資料表，請執行以下命令。

```
create external table spectrum.sales_event(
salesid integer,
listid integer,
sellerid integer,
buyerid integer,
eventid integer,
dateid smallint,
qtysold smallint,
pricepaid decimal(8,2),
```

```
commission decimal(8,2),
saletime timestamp)
partitioned by (salesmonth char(10), event integer)
row format delimited
fields terminated by '|'
stored as textfile
location 's3://redshift-downloads/ticket/spectrum/salesevent/'
table properties ('numRows'='172000');
```

若要新增分割區，請執行下列ALTER TABLE命令。

```
alter table spectrum.sales_event add
partition(salesmonth='2008-01', event='101')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-01/
event=101/'

partition(salesmonth='2008-01', event='102')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-01/
event=102/'

partition(salesmonth='2008-01', event='103')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-01/
event=103/'

partition(salesmonth='2008-02', event='101')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-02/
event=101/'

partition(salesmonth='2008-02', event='102')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-02/
event=102/'

partition(salesmonth='2008-02', event='103')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-02/
event=103/'

partition(salesmonth='2008-03', event='101')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-03/
event=101/'

partition(salesmonth='2008-03', event='102')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-03/
event=102/'
```

```
partition(salesmonth='2008-03', event='103')
location 's3://redshift-downloads/ticket/spectrum/salesevent/salesmonth=2008-03/
event=103/';
```

執行下列查詢以從分割的資料表選擇資料。

```
select spectrum.sales_event.salesmonth, event.eventname,
       sum(spectrum.sales_event.pricepaid)
from spectrum.sales_event, event
where spectrum.sales_event.eventid = event.eventid
      and salesmonth = '2008-02'
      and (event = '101'
           or event = '102'
           or event = '103')
group by event.eventname, spectrum.sales_event.salesmonth
order by 3 desc;
```

salesmonth	eventname	sum
2008-02	The Magic Flute	5062.00
2008-02	La Sonnambula	3498.00
2008-02	Die Walkure	534.00

將外部資料表資料欄映射至ORC資料欄

您可以使用 Amazon Redshift Spectrum 外部資料表，以 ORC 格式查詢檔案的資料。最佳化的資料列資料欄（ORC）格式是一種支援巢狀資料結構的資料欄儲存檔案格式。如需關於查詢巢狀資料的相關資訊，請參閱[使用 Amazon Redshift Spectrum 查詢巢狀資料](#)。

當您建立參考ORC檔案中資料的外部資料表時，您可以將外部資料表中的每個資料欄映射到ORC資料中的資料欄。若要這麼做，您會使用下列其中一個方法：

- [依位置映射](#)
- [依資料欄名稱映射](#)

依資料欄名稱映射為預設值。

依位置映射

透過位置映射，外部資料表中定義的第一欄會映射到ORC資料檔案中的第一欄，第二欄到第二欄，依此類推。依位置映射需要外部資料表和ORC檔案中的資料欄順序相符。如果資料欄的順序不相符，則可以依名稱映射資料欄。

⚠ Important

在舊版中，Redshift Spectrum 預設使用位置映射。如果您需要對現有資料表使用位置映射，請將資料表屬性 `orc.schema.resolution` 設為 `position`，如以下範例所示。

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='position');
```

例如，資料表 `SPECTRUM.ORC_EXAMPLE` 的定義如下。

```
create external table spectrum.orc_example(
int_col int,
float_col float,
nested_col struct<
  "int_col" : int,
  "map_col" : map<int, array<float >>
>
) stored as orc
location 's3://example/orc/files/';
```

資料表結構的摘要如下。

- 'int_col' : int
- 'float_col' : float
- 'nested_col' : struct
 - o 'int_col' : int
 - o 'map_col' : map
 - key : int
 - value : array
 - value : float

基礎ORC檔案具有下列檔案結構。

- ORC file root(id = 0)

```

o 'int_col' : int (id = 1)
o 'float_col' : float (id = 2)
o 'nested_col' : struct (id = 3)
  - 'int_col' : int (id = 4)
  - 'map_col' : map (id = 5)
    - key : int (id = 6)
    - value : array (id = 7)
      - value : float (id = 8)

```

在此範例中，您可以將外部資料表中的每個資料欄嚴格對應至ORC檔案中的資料欄。以下顯示映射。

外部資料表資料欄名稱	ORC 資料欄 ID	ORC 資料欄名稱
int_col	1	int_col
float_col	2	float_col
nested_col	3	nested_col
nested_col.int_col	4	int_col
nested_col.map_col	5	map_col
nested_col.map_col.key	6	不適用
nested_col.map_col.value	7	不適用
nested_col.map_col.value.item	8	NA

依資料欄名稱映射

使用名稱映射，您可以將外部資料表中的資料欄映射到相同層級ORC檔案中具名資料欄，名稱相同。

例如，假設您想要將上一個範例的資料表對應SPECTRUM. ORC_EXAMPLE至使用以下ORC檔案結構的檔案。

```

• ORC file root(id = 0)
  o 'nested_col' : struct (id = 1)
    - 'map_col' : map (id = 2)
      - key : int (id = 3)
      - value : array (id = 4)

```



```

    - value : float (id = 5)
  - 'int_col' : int (id = 6)
  o 'int_col' : int (id = 7)
  o 'float_col' : float (id = 8)

```

使用位置映射，Redshift Spectrum 會嘗試進行下列映射。

外部資料表資料欄名稱	ORC 資料欄 ID	ORC 資料欄名稱
int_col	1	struct
float_col	7	int_col
nested_col	8	float_col

當您使用上述位置映射查詢資料表時，由於結構不同，因此 SELECT 命令在類型驗證時失敗。

您可以使用資料欄名稱映射，將相同的外部資料表映射至先前範例中所示的檔案結構。資料表資料欄 int_col、float_col 和 依資料欄名稱 nested_col 映射至 ORC 檔案中名稱相同的資料欄。外部資料表中名為 nested_col 的資料欄為 struct 資料欄，具有的子資料欄名為 map_col 和 int_col。子欄也會依資料欄名稱正確對應至 ORC 檔案中的對應資料欄。

為在 Apache Hudi 中管理的資料建立外部資料表

若要查詢 Apache Hudi Copy On Write (CoW) 格式的資料，您可以使用 Amazon Redshift Spectrum 外部資料表。Hudi Copy On Write 資料表是儲存在 Amazon S3 中的 Apache Parquet 檔案的集合。您可以讀取 Apache Hudi 版本 0.5.2、0.6.0、0.7.0、0.8.0、0.9.0、0.10.0、0.10.1、0.11.0 和 0.11.1 中的 Copy On Write (CoW) 資料表，這些表是透過插入、刪除和更新插入寫入操作來建立和修改的。例如，不支援 Bootstrap 資料表。如需詳細資訊，請參閱開放原始碼 Apache Hudi 文件中的 [Copy On Write 資料表](#)。

當您建立參考 Hudi CoW 格式之資料的外部資料表時，您會將外部資料表中的每個欄映射到 Hudi 資料中的欄。映射是透過欄完成的。

已分割和未分割 Hudi 資料表的資料定義語言 (DDL) 陳述式與其他 Apache Parquet 檔案格式的資料定義語言 () 陳述式類似。對於 Hudi 資料表，您可以將 INPUTFORMAT 定義為 org.apache.hudi.hadoop.HoodieParquetInputFormat。LOCATION 參數必須指向包含 .hoodie 資料夾的 Hudi 資料表基底資料夾，這是建立 Hudi 提交時間表所必需的。在某些情況

下，Hudi 資料表上的SELECT操作可能會失敗，並且找不到有效的 Hudi 遞交時間表。若是如此，請檢查 .hoodie 資料夾是否位於正確的位置並包含有效的 Hudi 提交時間表。

Note

只有在使用 AWS Glue Data Catalog時才支援 Apache Hudi 格式。當您使用 Apache Hive 中繼存放區做為外部目錄時，不支援此功能。

DDL 定義未分割資料表的 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'org.apache.hadoop.hive.q1.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.q1.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

DDL 定義分割資料表的 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
PARTITIONED BY(pcolumn1 pcolumn1-type[,...])
ROW FORMAT SERDE 'org.apache.hadoop.hive.q1.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hudi.hadoop.HoodieParquetInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.q1.io.parquet.MapredParquetOutputFormat'
LOCATION 's3://s3-bucket/prefix'
```

若要將分割區新增至分割的 Hudi 資料表，請執行 ALTERTABLEADDPARTITION命令，其中LOCATION參數指向具有屬於分割區的檔案的 Amazon S3 子資料夾。

DDL 要新增分割區的 具有下列格式。

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION 's3://s3-bucket/prefix/partition-path'
```

為在 Delta Lake 中管理的資料建立外部資料表

若要查詢 Delta Lake 資料表中的資料，您可以使用 Amazon Redshift Spectrum 外部資料表。

若要從 Redshift Spectrum 存取 Delta Lake 資料表，請在查詢前產生清單檔案。Delta Lake 清單檔案包含構成 Delta Lake 資料表一致快照的檔案清單。在分割資料表中，每個分割區都有一個清單檔案。Delta Lake 資料表是儲存在 Amazon S3 中的 Apache Parquet 檔案的集合。如需詳細資訊，請參閱開放原始碼 Delta Lake 文件中的 [Delta Lake](#)。

當您建立參考 Delta Lake 資料表中資料的外部資料表時，您會將外部資料表中的每個欄映射到 Delta Lake 資料表中的欄。映射是透過欄名稱完成的。

已分割和未分割 Delta Lake 資料表DDL的 與其他 Apache Parquet 檔案格式的 相似。對於 Delta Lake 資料表，您可以將 INPUTFORMAT 定義為 `org.apache.hadoop.hive.q1.io.SymlinkTextInputFormat`，以及將 OUTPUTFORMAT 定義為 `org.apache.hadoop.hive.q1.io.HiveIgnoreKeyTextOutputFormat`。LOCATION 參數必須指向資料表基底資料夾中的清單檔案資料夾。如果 Delta Lake 資料表上的 SELECT 操作失敗，可能的原因請參閱 [Delta Lake 資料表的限制和疑難排解](#)。

DDL 定義未分割資料表的 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'org.apache.hadoop.hive.q1.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.q1.io.SymlinkTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.q1.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://s3-bucket/prefix/_symlink_format_manifest'
```

DDL 定義分割資料表的 具有下列格式。

```
CREATE EXTERNAL TABLE tbl_name (columns)
PARTITIONED BY(pcolumn1 pcolumn1-type[,...])
ROW FORMAT SERDE 'org.apache.hadoop.hive.q1.io.parquet.serde.ParquetHiveSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.q1.io.SymlinkTextInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.q1.io.HiveIgnoreKeyTextOutputFormat'
LOCATION 's3://s3-bucket>/prefix/_symlink_format_manifest'
```

若要將分割區新增至分割的 Delta Lake 資料表，請執行 ALTER TABLE ADD PARTITION 命令，其中 LOCATION 參數指向包含分割區清單的 Amazon S3 子資料夾。

DDL 要新增分割區的 具有下列格式。

```
ALTER TABLE tbl_name
```

```
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION
's3://s3-bucket/prefix/_symlink_format_manifest/partition-path'
```

或直接執行DDL指向 Delta Lake 資訊清單檔案的。

```
ALTER TABLE tbl_name
ADD IF NOT EXISTS PARTITION(pcolumn1=pvalue1[,...])
LOCATION
's3://s3-bucket/prefix/_symlink_format_manifest/partition-path/manifest'
```

Delta Lake 資料表的限制和疑難排解

從 Redshift Spectrum 查詢 Delta Lake 資料表時，請考慮下列事項：

- 如果清單檔案指向不再存在的快照集或分割區，則查詢會失敗，直到產生新的有效清單檔案為止。例如，這可能是因為基礎資料表上的 VACUUM 操作，
- Delta Lake 清單檔案僅提供分割區層級一致性。

下表說明查詢 Delta Lake 資料表時出現某些錯誤的一些潛在原因。

錯誤訊息	可能原因
儲存貯體 s3-bucket-1 中的 Delta Lake 清單檔案不能包含儲存貯體 s3-bucket-2 中的項目。	清單檔案項目指向與指定儲存貯體不同的 Amazon S3 儲存貯體中的檔案。
Delta Lake 檔案預計會位於相同的資料夾中。	清單檔案項目指向具有與指定字首不同的 Amazon S3 字首的檔案。
找不到 Delta Lake 清單檔案 manifest-path 中列出的檔案 filename。	在 Amazon S3 中找不到清單檔案中列出的檔案。
擷取 Delta Lake 清單檔案時發生錯誤。	在 Amazon S3 中找不到清單檔案。
無效的 S3 路徑。	清單檔案檔案中的項目不是有效的 Amazon S3 路徑，或清單檔案檔案已損毀。

搭配 Amazon Redshift 使用 Apache Iceberg 資料表

本主題說明如何使用 Apache 冰山格式的表格與 Redshift 頻譜或 Redshift 無伺服器。阿帕奇冰山是一種用於巨大的分析表的高性能格式。

您可以使用 Redshift 頻譜或 Redshift 無服務器來查詢編目的 Apache 冰山表 AWS Glue Data Catalog。Apache 冰山是用於資料湖的開放原始碼資料表格式。如需詳細資訊，請參閱 Apache Iceberg 文件中的 [Apache Iceberg](#)。

Amazon Redshift 為查詢 Apache Iceberg 資料表提供交易一致性。您可以在使用 Amazon Amazon Redshift 執行查詢EMR時，使用 Amazon Athena 和 Amazon 等合規服務 ACID (原子性、一致性、隔離、耐久性) 操控資料表中的資料。Amazon Redshift 可以使用 Apache Iceberg 中繼資料中儲存的表格統計資料來最佳化查詢計劃並減少查詢處理期間的檔案掃描。使用 Amazon RedshiftSQL，您可以將 Redshift 表與資料湖表結合在一起。

若要開始將 Iceberg 資料表與 Amazon Redshift 搭配使用：

1. 創建一個阿帕奇冰山表 AWS Glue Data Catalog 數據庫使用兼容的服務，如 Amazon Athena 或 Amazon EMR。若要使用 Athena 建立 Iceberg 資料表，請參閱《Amazon Athena 使用者指南》中的 [使用 Apache Iceberg 資料表](#)。
2. 建立具有可存取資料湖的關聯IAM角色的 Amazon Redshift 叢集或 Redshift 無伺服器工作群組。如需如何建立叢集或工作群組的詳細資訊，請參閱 [Amazon Redshift 佈建的資料倉儲入門](#) 和 Amazon Redshift [入門指南中的開始使用 Redshift 無伺服器資料倉儲](#)。
3. 使用查詢編輯器 v2 或協力廠商用SQL戶端 Connect 線至叢集或工作群組。如需如何使用查詢編輯器 v2 連線的詳細資訊，請參閱 [Amazon Redshift 管理指南中的使用用SQL戶端工具連線至 Amazon Redshift 資料倉儲](#)。
4. 在 Amazon Redshift 資料庫中為包含 Iceberg 資料表的特定資料目錄資料庫建立外部結構描述。如需建立外部結構描述的相關資訊，請參閱 [Amazon Redshift Spectrum 中的外部結構描述](#)。
5. 執行SQL查詢以存取您建立的外部結構描述中的 Iceberg 資料表。

將 Apache Iceberg 資料表與 Amazon Redshift 搭配使用時的注意事項

將 Amazon Redshift 與 Iceberg 資料表結合使用時，請考慮以下事項：

- Iceberg 版本支援 - Amazon Redshift 支援對以下版本的 Iceberg 資料表執行查詢：
 - 版本 1 定義如何使用不可變資料檔案管理大型分析表。

- 版本 2 增加支援資料列層級更新和刪除的功能，同時保持現有資料檔案不變，並使用刪除檔案來處理資料表資料變更。

有關版本 1 和版本 2 資料表之間的區別，請參閱 Apache Iceberg 文件中的[格式版本變更](#)。

- 僅限查詢 - Amazon Redshift 支援 Apache Iceberg 資料表的唯讀存取。它支援交易一致的選擇查詢。您可以使用像 Amazon Athena 這樣的服務來定義和更新冰山表的模式 AWS Glue Data Catalog。
- 新增分割區 - 您不需要為 Apache Iceberg 資料表手動新增分割區。Amazon Redshift 會自動偵測 Apache Iceberg 資料表中的新分割區，而且不需要手動操作即可更新資料表定義中的分割區。分割區規格中的任何變更也會自動套用至您的查詢，而無需任何使用者介入。
- 將冰山資料導入亞馬遜紅移 — 您可以使用 INSERT INTO 或 CREATE TABLE AS 命令將資料從冰山表格匯入本機 Amazon Redshift 表格中。您目前無法使用該 COPY 命令將 Apache 冰山表的內容導入本地 Amazon Redshift 表中。
- 具體化視觀表 - 您可以在 Apache Iceberg 資料表上建立具體化視觀表，就像 Amazon Redshift 中的任何其他外部資料表一樣。其他資料湖資料表格式的考量相同，也適用於 Apache Iceberg 資料表。目前不支援累加式更新、自動重新整理、自動查詢重新寫入，以及資料湖資料表 MVs 上的自動。
- AWS Lake Formation 精細的存取控 Amazon Redshift 援 AWS Lake Formation 阿帕奇冰山表上的細粒度訪問控制。
- 使用者定義的資料處理參數 - Amazon Redshift 支援 Apache Iceberg 資料表上的使用者定義資料處理參數。您可以在現有檔案上使用使用者定義的資料處理參數來自訂外部資料表中查詢的資料，以避免掃描錯誤。這些參數提供處理資料表結構描述與檔案實際資料之間不相符的功能。您也可以可以在 Apache Iceberg 資料表上使用使用者定義的資料處理參數。
- 資料共用 - Amazon Redshift 資料共用目前不支援資料湖資料表，包括 Apache Iceberg 資料表。
- 時間歷程查詢 - Apache Iceberg 資料表目前不支援時間歷程查詢。
- 定價 - 當您從叢集存取 Iceberg 資料表時，您需要支付 Redshift Spectrum 定價的費用。當您從工作群組存取 Iceberg 資料表時，您需要支付 Redshift Serverless 定價的費用。如需 Redshift Spectrum 和 Redshift Serverless 定價的相關資訊，請參閱 [Amazon Redshift 定價](#)。

Apache Iceberg 資料表支援的資料類型

本主題說明 Redshift 頻譜可以從 Apache 冰山格式的資料表讀取的支援資料類型。

Amazon Redshift 可以查詢包含下列資料類型的 Iceberg 資料表：

```
binary
boolean
date
```

```

decimal
double
float
int
list
long
map
string
struct
timestamp without time zone

```

如需 Iceberg 資料類型的相關資訊，請參閱 Apache Iceberg 文件中的 [Iceberg 結構描述](#)。

下表顯示 Amazon Redshift 資料類型與 Iceberg 資料表資料類型之間的關係。

Iceberg 類型	Amazon Redshift 類型	備註
boolean	boolean	
-	tinyint	Amazon Redshift 中的 Iceberg 資料表不支援。
-	smallint	Amazon Redshift 中的 Iceberg 資料表不支援。
int	int	在 Amazon Redshift SQL 語句中，這種類型是 INTEGER.
long	bigint	
double	double	
float	float	
decimal(P, S)	decimal(P, S)	P 是總位數，S 是小數部分的位數。
-	char	Redshift Spectrum 中的 Iceberg 資料表不支援。
string	string	在 Amazon Redshift SQL 語句中，這種類型是 VARCHAR.
binary	binary	

Iceberg 類型	Amazon Redshift 類型	備註
date	date	
time	-	
timestamp	timestamp	
timestamp tz	-	
list<E>	array	
map<K,V>	map	
struct<.. >	struct	
fixed(L)	-	Redshift Spectrum 目前不支援 fixed(L) 類型。

如需 Amazon Redshift 中資料類型的相關資訊，請參閱[資料類型](#)。

Amazon Redshift Spectrum 查詢效能

本主題說明如何改善 Redshift Spectrum 查詢效能。

請查看查詢計畫以尋找已推送到 Amazon Redshift Spectrum 層的步驟。

下列步驟與 Redshift Spectrum 查詢相關：

- S3 循序掃描
- S3 HashAggregate
- S3 查詢掃描
- 序列掃描 PartitionInfo
- 分割區循環

以下範例顯示了將外部資料表與本地資料表連接的查詢的查詢計畫。請注意針對 Amazon S3 HashAggregate 上的資料執行的 S3 序列掃描和 S3 步驟。 Amazon S3

```
explain
select top 10 spectrum.sales.eventid, sum(spectrum.sales.pricepaid)
from spectrum.sales, event
where spectrum.sales.eventid = event.eventid
and spectrum.sales.pricepaid > 30
group by spectrum.sales.eventid
order by 2 desc;
```

QUERY PLAN

```
-----
XN Limit (cost=1001055770628.63..1001055770628.65 rows=10 width=31)

-> XN Merge (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

    Merge Key: sum(sales.derived_col2)

-> XN Network (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

    Send to leader

-> XN Sort (cost=1001055770628.63..1001055770629.13 rows=200 width=31)

    Sort Key: sum(sales.derived_col2)

-> XN HashAggregate (cost=1055770620.49..1055770620.99 rows=200
width=31)

-> XN Hash Join DS_BCAST_INNER (cost=3119.97..1055769620.49
rows=200000 width=31)
```

```

Hash Cond: ("outer".derived_col1 = "inner".eventid)

-> XN S3 Query Scan sales (cost=3010.00..5010.50
rows=200000 width=31)

-> S3 HashAggregate (cost=3010.00..3010.50
rows=200000 width=16)

-> S3 Seq Scan spectrum.sales
location:"s3://redshift-downloads/ticket/spectrum/sales" format:TEXT
(cost=0.00..2150.00 rows=172000 width=16)
Filter: (pricepaid > 30.00)

-> XN Hash (cost=87.98..87.98 rows=8798 width=4)

-> XN Seq Scan on event (cost=0.00..87.98
rows=8798 width=4)

```

請注意查詢計畫中的以下元素：

- S3 Seq Scan 節點顯示的篩選條件 `pricepaid > 30.00` 已在 Redshift Spectrum 層中進行處理。

XN S3 Query Scan 節點下的篩選條件節點指示了從 Redshift Spectrum 層傳回資料上方 Amazon Redshift 中的述詞處理。

- S3 HashAggregate 節點表示了 Redshift Spectrum 層中的 `group by` 子句 (`group by spectrum.sales.eventid`) 彙整。

以下為改善 Redshift Spectrum 效能的方式：

- 使用 Apache Parquet 格式的資料檔案。Parquet 會以單欄格式儲存資料，所以 Redshift Spectrum 可從掃描中消除不需要的欄位。當資料為文字檔案格式，Redshift Spectrum 將需要掃描整個檔案。
- 使用多個檔案以最佳化您的平行處理。將您的檔案維持大於 64 MB。通過保持相同的檔案大小以避免資料大小扭曲。如需有關 Apache Parquet 檔案和組態建議的資訊，請參閱 Apache Parquet 文件中的 [檔案格式：組態](#)。
- 在查詢中盡可能使用最少欄位。

- 將大型事實資料表放在 Amazon S3 中，並將常用的較小維度資料表保存在您本機 Amazon Redshift 資料庫中。
- 透過設定 TABLE PROPERTIES numRows 參數來更新外部資料表統計資料。使用 [CREATE EXTERNAL TABLE](#) 或 [ALTER TABLE](#) 設定 TABLE PROPERTIES numRows 參數，以反映資料表中的資料列數。Amazon Redshift 不會分析外部資料表來產生查詢最佳化工具用來產生查詢計劃的資料表統計資料。如果未設定外部資料表的資料表統計資料，則 Amazon Redshift 會產生查詢執行計畫。Amazon Redshift 根據以下假設產生此計畫：外部資料表為較大資料表，而本機資料表為較小資料表。
- Amazon Redshift 查詢計畫器會盡可能將述詞和彙總推送到 Redshift Spectrum 查詢層。當從 Amazon S3 傳回大量資料時，該處理將受到叢集資源的限制。Redshift Spectrum 將自動擴展以處理大量請求。因此，只要您可以將處理推送到 Redshift Spectrum 層，您的整體效能便會改善。
- 撰寫您的查詢以使用有資格推送到 Redshift Spectrum 層的篩選條件和彙總。

以下是可以推送到 Redshift Spectrum 層的一些操作範例：

- GROUP BY 子句
- 比較條件和模式比對條件，例如 LIKE。
- 彙總函數，例如 COUNT、SUM、MIN、AVG 和 MAX。
- 字串函式。

無法推送至 Redshift Spectrum 層的操作包括 DISTINCT 和 ORDER BY。

- 使用分割區來限制掃描的資料。根據最常見的查詢述詞對資料進行分割，然後透過在分割區欄位進行篩選以刪除分割區。如需詳細資訊，請參閱 [分割 Redshift Spectrum 外部資料表](#)。

查詢 [SVL_S3PARTITION](#) 以檢視分割區總數與合格的分割區。

- 使用 AWS Glue 的統計資料產生器來計算資料表的資料 AWS Glue Data Catalog 欄層級統計資料。一旦在 Data Catalog 中 AWS Glue 產生資料表的統計資料，Amazon Redshift Spectrum 會自動使用這些統計資料來最佳化查詢計劃。如需使用 計算資料欄層級統計資料的詳細資訊 AWS Glue，請參閱 AWS Glue 開發人員指南 中的 [使用資料欄統計資料](#)。

資料處理選項

本主題說明如何設定 Redshift Spectrum 如何處理非預期格式的資料。

您可以在建立外部資料表時設定資料表參數，以調整外部資料表中查詢的資料。否則，可能會發生掃描錯誤。如需詳細資訊，請參閱 TABLE PROPERTIES 中的 [CREATE EXTERNAL TABLE](#)。如需範例，請參閱 [資料處理範例](#)。如需錯誤清單，請參閱 [SVL_SPECTRUM_SCAN_ERROR](#)。

您可以在建立外部資料表TABLEPROPERTIES時設定下列項目，以指定在外部資料表中查詢資料的輸入處理。

- `column_count_mismatch_handling`，識別檔案包含的列值是否少於或多於外部資料表定義中指定的欄數。
- `invalid_char_handling` 為包含 VARCHAR、CHAR和 字串資料的欄中的無效字元指定輸入處理。當您REPLACE為 指定 時`invalid_char_handling`，您可以指定要使用的取代字元。
- `numeric_overflow_handling`，指定包含整數和小數資料之欄中的強制轉換溢位處理。
- `surplus_bytes_handling` 指定包含VARBYTE資料的欄中剩餘位元組的輸入處理。
- `surplus_char_handling` 在包含 VARCHAR、CHAR和 字串資料的欄中指定多餘字元的輸入處理。

您可以設定組態選項來取消超過錯誤數目上限的查詢。如需詳細資訊，請參閱[spectrum_query_maxerror](#)。

範例：在 Redshift Spectrum 中執行相互關聯子查詢

本主題說明如何在 Redshift Spectrum 中執行關聯的子查詢。關聯的子查詢是使用外部查詢值的查詢。

您可以在 Redshift Spectrum 中執行相互關聯子查詢。`$spectrum_oid` 虛擬資料欄提供使用 Redshift Spectrum 執行相關查詢的能力。若要執行關聯的子查詢，`$spectrum_oid`必須啟用虛擬資料欄，但不會出現在SQL陳述式中。如需詳細資訊，請參閱[虛擬資料欄](#)。

若要為此範例建立外部結構描述和外部資料表，請參閱[開始使用 Amazon Redshift Spectrum](#)。

以下是 Redshift Spectrum 中的相互關聯子查詢範例。

```
select *
from myspectrum_schema.sales s
where exists
( select *
from myspectrum_schema.listing l
where l.listid = s.listid )
order by salesid
limit 5;
```

```
salesid listid sellerid buyerid eventid dateid qtysold pricepaid
commission saletime
```

1	1	36861	21191	7872	1875	4	728	109.2
	2008-02-18 02:36:48							
2	4	8117	11498	4337	1983	2	76	11.4
	2008-06-06 05:00:16							
3	5	1616	17433	8647	1983	2	350	52.5
	2008-06-06 08:26:17							
4	5	1616	19715	8647	1986	1	175	26.25
	2008-06-09 08:38:52							
5	6	47402	14115	8240	2069	2	154	23.1
	2008-08-31 09:17:02							

Amazon Redshift Spectrum 中的指標

本主題說明可用來監控 Redshift Spectrum 查詢的系統檢視。

您可以使用以下的系統檢視來監控 Amazon Redshift Spectrum 查詢：

- [SVL_S3QUERY](#)

使用 SVL_S3QUERY 檢視取得區段和節點層 Redshift Spectrum 查詢（S3 查詢）的詳細資訊。

- [SVL_S3QUERY_SUMMARY](#)

使用 SVL_S3QUERY_SUMMARY 檢視取得已在系統上執行之所有 Amazon Redshift Spectrum 查詢（S3 查詢）的摘要。

以下是在 SVL_S3QUERY_ 中需要尋找的一些事項SUMMARY：

- Redshift Spectrum 查詢已處理的檔案數。
- 從 Amazon S3 掃描的位元組數。Redshift Spectrum 查詢的成本反映在從 Amazon S3 掃描的資料量中。
- 從 Redshift Spectrum 層傳回至叢集的位元組數。傳回的大量資料可能會影響系統效能。
- Redshift Spectrum 請求的最大持續期間和平均持續時間。長時間執行的請求可能表示存在瓶頸。

Amazon Redshift Spectrum 中的查詢疑難排解

本主題是您使用 Amazon Redshift Spectrum 查詢時可能遇到的常見問題的參考。

查詢 [SVL_S3LOG](#) 系統資料表以檢視由 Redshift Spectrum 查詢產生的錯誤。

主題

- [超過重試次數](#)
- [存取已限流](#)
- [超過資源限制](#)
- [分割的資料表沒有傳回資料列](#)
- [未授權的錯誤](#)
- [不相容的資料格式](#)
- [在 Amazon Redshift DDL 中使用 Hive 時的語法錯誤](#)
- [建立暫存資料表的許可](#)
- [無效的範圍](#)
- [無效的 Parquet 版本編號](#)

超過重試次數

如果 Amazon Redshift Spectrum 請求逾時，則該請求會被取消並重新提交。在五次重試失敗後，該查詢即失敗並顯示以下錯誤。

```
error: Spectrum Scan Error: Retries exceeded
```

可能的子句包括：

- 大型檔案 (大於 1 GB)。檢查 Amazon S3 中的檔案大小，尋找大型檔案和檔案大小扭曲。將大型檔案分成許多介於 100 MB 與 1 GB 的較小檔案。試著將檔案維持相同大小。
- 慢速網路傳輸量。請稍後嘗試您的查詢。

存取已限流

Amazon Redshift Spectrum 受其他服務的服務配額限制 AWS。在高使用量下，Redshift Spectrum 請求可能需要減慢速度，因而導致以下錯誤。

```
error: Spectrum Scan Error: Access throttled
```

可能發生兩種類型的限流：

- 存取受到 Amazon S3 限流。

- 由限流的存取 AWS KMS。

錯誤內容提供有關限流類型的詳細資料。您可以在下面找到此限流的原因和可能的解決方案。

存取受到 Amazon S3 限流

如果字首上的讀取請求率太高，Amazon S3 可能會限流 Redshift Spectrum 請求。如需您可以在 Amazon S3 中達到的 GET/HEAD 請求率資訊，請參閱 Amazon Simple Storage Service 使用者指南中的[最佳化 Amazon S3 效能](#)。Amazon S3 GET/HEAD 請求速率會將字首上的所有 GET/HEAD 請求納入考量，讓存取相同字首的不同應用程式共用總請求速率。

如果您的 Redshift Spectrum 請求經常受到 Amazon S3 限流，請減少 Redshift Spectrum 對 Amazon S3 發出的 Amazon S3 GET/HEAD 請求數量。若要執行此動作，請嘗試將小型檔案合併成大型檔案。我們建議使用大於 64 MB 的檔案。

另外，請考慮分割 Redshift Spectrum 資料表，以從早期篩選中獲益，並減少 Amazon S3 中存取的檔案數量。如需詳細資訊，請參閱[分割 Redshift Spectrum 外部資料表](#)。

存取由 AWS KMS 限流

如果您使用伺服器端加密（SSE-S3 或 SSE-KMS）將資料存放在 Amazon S3，Amazon S3 AWS KMS 會針對 Redshift Spectrum 存取的每個檔案呼叫 API 操作。這些請求會計入您的密碼編譯操作配額；如需詳細資訊，請參閱[AWS KMS 請求配額](#)。如需 SSE-S3 和 SSE-KMS 的詳細資訊，請參閱 Amazon Simple Storage Service 使用者指南中的[使用伺服器端加密保護資料](#)，以及[使用存放於 Amazon S3 中的 KMS 金鑰保護資料](#) [AWS KMS](#)。

減少 Redshift Spectrum 向 Amazon S3 發出請求數量的第一步 AWS KMS 是減少存取的檔案數量。若要執行此動作，請嘗試將小型檔案合併成大型檔案。我們建議使用大於 64 MB 的檔案。

如果您的 Redshift Spectrum 請求經常受到限制 AWS KMS，請考慮為您的密碼編譯操作 AWS KMS 請求率請求增加配額。若要申請提高限制，請參閱 Amazon Web Services 一般參考中的[AWS 服務限制](#)。

超過資源限制

Redshift Spectrum 會對請求可以使用的記憶體數量強加上限。需要更多記憶體的 Redshift Spectrum 請求失敗，導致下列錯誤。

```
error: Spectrum Scan Error: Resource limit exceeded
```

有兩個常見的原因可能會造成 Redshift Spectrum 請求超出其記憶體限額：

- Redshift Spectrum 會處理大塊資料，這些資料不能以較小的區塊分割。
- 大型彙總步驟是由 Redshift Spectrum 處理。

建議您使用支援平行讀取的檔案格式，分割大小為 128 MB 或更少。如需支援的檔案格式和建立資料檔的一般指導方針，請參閱[Amazon Redshift Spectrum 中查詢的資料檔案](#)。使用不支援平行讀取的檔案格式或壓縮演算法時，建議您將檔案大小保持在 64 MB 與 128 MB 之間。

分割的資料表沒有傳回資料列

如果您的查詢沒有從分割的外部資料表傳回任何資料列，請檢查是否已為此外部資料表新增分割區。Redshift Spectrum 僅會掃描使用 ALTER TABLE ... ADD PARTITION 明確新增之 Amazon S3 位置中的檔案。請查詢 [SVV_EXTERNAL_PARTITIONS](#) 畫面以尋找現有分割區。針對每個缺少的分割區執行 ALTER TABLE ... ADD PARTITION。

未授權的錯誤

確認叢集IAM的角色允許存取 Amazon S3 檔案物件。如果您的外部資料庫位於 Amazon Athena 上，請確認IAM角色允許存取 Athena 資源。如需詳細資訊，請參閱[IAM Amazon Redshift Spectrum 的政策](#)。

不相容的資料格式

對於單欄檔案格式 (如 Apache Parquet)，該欄位類型已嵌入了資料。CREATE EXTERNAL TABLE 定義中的資料欄類型必須與資料檔案的資料欄類型相符。如果有不相符的部分，您將收到類似以下的錯誤訊息：

```
File 'https://s3bucket/location/file' has an incompatible Parquet schema
for column 's3://s3bucket/location.col1'. Column type: VARCHAR, Par
```

由於訊息長度限制，錯誤訊息可能會被截斷。若要獲取包含欄位名稱與欄位類型的完整錯誤訊息，請查詢 [SVL_S3LOG](#) 系統畫面。

下列範例查詢 SVL_S3LOG 上次完成的查詢。

```
select message
from svl_s3log
where query = pg_last_query_id()
```



```
order by query,segment,slice;
```

以下是顯示完整錯誤訊息的結果範例。

```
message
```

```
-----  
Spectrum Scan Error. File 'https://s3bucket/location/file has an incompatible  
Parquet schema for column ' s3bucket/location.col1'.  
Column type: VARCHAR, Parquet schema:\noptional int64 l_orderkey [i:0 d:1 r:0]\n
```

若要更正此錯誤，請修改外部資料表以符合 Parquet 檔案的欄位類型。

在 Amazon Redshift DDL 中使用 Hive 時的語法錯誤

Amazon Redshift 支援 CREATE EXTERNAL TABLE 類似 Hive 的資料定義語言 (DDL) DDL。不過，這兩種類型的 DDL 不一定完全相同。如果您複製 Hive DDL 來建立或修改 Amazon Redshift 外部資料表，您可能會遇到語法錯誤。以下是 Amazon Redshift 和 Hive 之間的差異範例 DDL：

- Amazon Redshift 需要單一引號 (')，其中 Hive DDL 支援雙引號 (")。
- Amazon Redshift 不支援 STRING 資料類型。請改用 VARCHAR。

建立暫存資料表的許可

要執行 Redshift Spectrum 查詢，資料庫使用者必須具有在資料庫中建立臨時資料表的許可。下列範例可在資料庫 spectrumdb 上授予臨時許可至 spectrumusers 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

如需詳細資訊，請參閱 [GRANT](#)。

無效的範圍

Redshift Spectrum 預期 Amazon S3 中屬於外部資料表的檔案在查詢期間不會被覆寫。如果發生這種情況，可能會導致以下錯誤。

```
Error: HTTP response error code: 416 Message: InvalidRange The requested range is not  
satisfiable
```

為避免發生錯誤，請確定 Amazon S3 檔案在使用 Redshift Spectrum 查詢時不會覆寫這些檔案。

無效的 Parquet 版本編號

Redshift Spectrum 會檢查它存取的每個 Apache Parquet 檔案的中繼資料。若檢查失敗，可能會導致類似下列內容的錯誤：

```
File 'https://s3.region.amazonaws.com/s3bucket/location/file' has an invalid version number
```

導致檢查失敗的常見原因有兩個：

- Parquet 檔案在查詢期間已被覆寫 (請參閱[無效的範圍](#))。
- Parquet 檔案已損毀。

教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料

本教程演示如何使用 Redshift 頻譜查詢嵌套數據。嵌套數據是包含嵌套字段的數據。巢狀欄位是以單一實體形式結合在一起的欄位，例如陣列、結構或物件。

主題

- [概觀](#)
- [步驟 1：建立包含巢狀資料的外部資料表](#)
- [步驟 2：使用 SQL 擴充功能查詢 Amazon S3 中的巢狀資料](#)
- [巢狀資料使用案例](#)
- [巢狀資料限制 \(預覽\)](#)
- [序列化複雜的嵌套 JSON](#)

概觀

Amazon Redshift Spectrum 支援以實木複合地板、ORCJSON、和離子檔案格式查詢巢狀資料。Redshift Spectrum 會存取使用外部資料表的資料。您可以建立使用複雜資料類型 struct、array 和 map 的外部資料表。

例如，假設您的資料檔案在 Amazon S3 名為 customers 的資料夾中包含下列資料。雖然沒有單一元素，但此範例資料中的每個 JSON 物件都代表資料表中的一列。

```
{"id": 1,
 "name": {"given": "John", "family": "Smith"},
```

```
"phones": ["123-457789"],
"orders": [{"shipdate": "2018-03-01T11:59:59.000Z", "price": 100.50},
           {"shipdate": "2018-03-01T09:10:00.000Z", "price": 99.12}]
}
{"id": 2,
 "name": {"given": "Jenny", "family": "Doe"},
 "phones": ["858-8675309", "415-9876543"],
 "orders": []
}
{"id": 3,
 "name": {"given": "Andy", "family": "Jones"},
 "phones": [],
 "orders": [{"shipdate": "2018-03-02T08:02:15.000Z", "price": 13.50}]
}
```

您現在可以使用 Amazon Redshift Spectrum 來查詢檔案中的巢狀資料。下列教學課程說明如何使用 Apache Parquet 資料執行此操作。

必要條件

如果您尚未使用 Redshift Spectrum，請先遵循[開始使用 Amazon Redshift Spectrum](#)中的步驟，再繼續進行。

若要建立外部結構描述，請將下列命令ARN中的IAM角色取代為ARN您在建立角色中[建立的IAM角色](#)。然後在SQL客戶端中運行該命令。

```
create external schema spectrum
from data catalog
database 'myspectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myspectrum_role'
create external database if not exists;
```

步驟 1：建立包含巢狀資料的外部資料表

您可以從 Amazon S3 下載[來源資料](#)以檢視來源資料。

若要建立此教學課程的外部資料表，請執行以下命令。

```
CREATE EXTERNAL TABLE spectrum.customers (
  id      int,
  name    struct<given:varchar(20), family:varchar(20)>,
  phones  array<varchar(20)>,
```

```

orders array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/ticket/spectrum/customers/';

```

在上述範例中，外部資料表 `spectrum.customers` 使用 `struct` 和 `array` 資料類型來定義具有巢狀資料的資料欄。Amazon Redshift Spectrum 支援以實木複合地板、ORCJSON、和離子檔案格式查詢巢狀資料。Apache Parquet 檔案的 `STORED AS` 參數是 `PARQUET`。LOCATION 參數必須參考包含巢狀資料或檔案的 Amazon S3 資料夾。如需詳細資訊，請參閱 [CREATE EXTERNAL TABLE](#)。

您可以在任何層級將 `array` 和 `struct` 類型形成巢狀。例如，您可以如以下範例所示定義名為 `toparray` 的資料欄。

```

toparray array<struct<nestedarray:
    array<struct<morenestedarray:
        array<string>>>>>

```

您也可以如以下範例中的資料欄 `x` 所示，將 `struct` 類型形成巢狀。

```

x struct<a: string,
    b: struct<c: integer,
        d: struct<e: string>
    >
>

```

步驟 2：使用 SQL 擴充功能查詢 Amazon S3 中的巢狀資料

紅移頻譜支援透過 Amazon Redshift SQL 語法的擴充功能來查詢 `array` 和 `struct` 複雜類型。map

延組模組 1：存取 structs 的欄位

您可以使用可將欄位名稱串連為路徑的點表示法，從 `struct` 資料欄擷取資料。例如，以下查詢會傳回客戶的名字和姓氏。名字是透過長路徑 `c.name.given` 存取。姓氏是透過長路徑 `c.name.family` 存取。

```

SELECT c.id, c.name.given, c.name.family
FROM   spectrum.customers c;

```

前述查詢會傳回下列資料。

```

id | given | family

```

```

---|-----|-----
1 | John | Smith
2 | Jenny | Doe
3 | Andy | Jones
(3 rows)

```

`struct` 可以是另一個 `struct` 的資料欄，可以是任何層級的另一個 `struct` 的資料欄。存取這類深入巢狀 `struct` 中資料欄的路徑可以有任意長度。例如，查看以下範例中資料欄 `x` 的定義。

```

x struct<a: string,
      b: struct<c: integer,
              d: struct<e: string>
            >
      >

```

您可以以 `x.b.d.e` 的形式存取 `e` 中的資料。

擴展 2：範圍在一個 FROM 子句中的數組

您可以從 `array` 資料欄擷取資料 (並且透過延伸模組 `map` 資料欄)，方法是在 `FROM` 子句中指定 `array` 資料欄以取代資料表名稱。延伸模組會套用到主查詢的 `FROM` 子句，以及子查詢的 `FROM` 子句。

您可以依位置參考 `array` 元素，例如 `c.orders[0]`。(預覽)

藉由將範圍與 `arrays` 聯結結合，您可以達成各種解巢狀，如下列使用案例所述。

使用內部聯結解巢狀

下列查詢會針對具有訂單的客戶選取客戶 IDs 與訂單出貨日期。`FROM` 子句中的 SQL 副檔名 `c.orders` `o` 取決於別名 `c`。

```

SELECT c.id, o.shipdate
FROM   spectrum.customers c, c.orders o

```

針對具有訂單的每個客戶 `c`，`FROM` 子句會為客戶 `c` 的每個訂單 `o` 傳回一個資料列。該資料列會結合客戶資料列 `c` 和訂單資料列 `o`。那麼，`SELECT` 子句只會保留 `c.id` 和 `o.shipdate`。結果如下所示。

```

id|      shipdate
--|-----
1 |2018-03-01  11:59:59
1 |2018-03-01  09:10:00

```

```
3 | 2018-03-02 08:02:15
(3 rows)
```

別名 `c` 提供客戶欄位的存取，而別名 `o` 則提供訂單欄位的存取。

語義類似於標準SQL。您可以將 FROM 子句想成執行下列巢狀迴圈，它的後面接著 SELECT 選擇要輸出的欄位。

```
for each customer c in spectrum.customers
  for each order o in c.orders
    output c.id and o.shipdate
```

因此，如果客戶沒有訂單，客戶便不會出現在結果中。

您也可以將這想成執行 JOIN 搭配 `customers` 資料表和 `orders` 陣列的 FROM 子句。實際上，您也可以如下列範例所示寫入查詢。

```
SELECT c.id, o.shipdate
FROM   spectrum.customers c INNER JOIN c.orders o ON true
```

Note

如果名為 `c` 的結構描述存在於名為 `orders` 的資料表中，則 `c.orders` 會參考資料表 `orders`，而非 `customers` 的陣列資料欄。

使用左聯結解巢狀

下列查詢會輸出所有客戶名稱和其訂單。如果客戶尚未下訂單，仍會傳回客戶的名稱。但是，在這種情況下，順序列是 NULL，如下面的例子 Jenny Doe 所示。

```
SELECT c.id, c.name.given, c.name.family, o.shipdate, o.price
FROM   spectrum.customers c LEFT JOIN c.orders o ON true
```

前述查詢會傳回下列資料。

id	given	family	shipdate	price
1	John	Smith	2018-03-01 11:59:59	100.5
1	John	Smith	2018-03-01 09:10:00	99.12
2	Jenny	Doe		

```
3 | Andy | Jones | 2018-03-02 08:02:15 | 13.5
(4 rows)
```

延伸模組 3：直接使用別名存取純量的陣列

當 FROM 子句中別名 p 的範圍超過純量的陣列，則查詢會完全以 p 的形式參考 p 的值。例如，以下查詢會產生客戶名稱與電話號碼的配對。

```
SELECT c.name.given, c.name.family, p AS phone
FROM spectrum.customers c LEFT JOIN c.phones p ON true
```

前述查詢會傳回下列資料。

```
given | family | phone
-----|-----|-----
John  | Smith  | 123-4577891
Jenny | Doe    | 858-8675309
Jenny | Doe    | 415-9876543
Andy  | Jones  |
(4 rows)
```

延伸模組 4：存取 map 的元素

Redshift Spectrum 會將 map 資料類型視為 array 類型，其中包含的 struct 類型具有 key 資料欄和 value 資料欄。key 必須是 scalar；值可以是任何資料類型。

例如，以下程式碼會建立外部資料表，其中具有 map 用於儲存電話號碼。

```
CREATE EXTERNAL TABLE spectrum.customers2 (
  id      int,
  name    struct<given:varchar(20), family:varchar(20)>,
  phones  map<varchar(20), varchar(20)>,
  orders  array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/ticket/spectrum/customers/';
```

因為 map 類型的行為類似具有資料欄 key 和 value 的 array 類型，您可以將前述結構描述想像成如下。

```
CREATE EXTERNAL TABLE spectrum.customers3 (
```

```

id      int,
name    struct<given:varchar(20), family:varchar(20)>,
phones  array<struct<key:varchar(20), value:varchar(20)>>,
orders  array<struct<shipdate:timestamp, price:double precision>>
)
STORED AS PARQUET
LOCATION 's3://redshift-downloads/ticket/spectrum/customers/';

```

下列查詢會傳回具有行動電話號碼之客戶的名稱，並傳回每個名稱的號碼。映射查詢被視為與查詢 struct 類型的巢狀 array 的同等項目。下列查詢只會在您如先前所述建立了外部資料表時傳回資料。

```

SELECT c.name.given, c.name.family, p.value
FROM   spectrum.customers c, c.phones p
WHERE  p.key = 'mobile';

```

Note

對key於 a map string 是 lon 和JSON文件類型。

巢狀資料使用案例

本主題描述巢狀資料的使用案例。嵌套數據是包含嵌套字段的數據。巢狀欄位是以單一實體形式結合在一起的欄位，例如陣列、結構或物件。

您可以將先前描述的擴充功能與常用SQL功能結合使用。以下使用案例說明一些常見的組合。這些範例可協助示範如何使用巢狀資料。本教學課程中未提供。

主題

- [擷取巢狀資料](#)
- [使用子查詢彙整巢狀資料](#)
- [聯結 Amazon Redshift 和巢狀資料](#)

擷取巢狀資料

您可以使用 CREATE TABLE AS 陳述式從包含複雜資料類型的外部資料表擷取資料。下列查詢會使用 LEFT JOIN 從外部資料表擷取所有客戶和其電話號碼，並將該資訊儲存在 Amazon Redshift 資料表 CustomerPhones。


```
CREATE TABLE CustomerPhones AS
SELECT  c.name.given, c.name.family, p AS phone
FROM    spectrum.customers c LEFT JOIN c.phones p ON true;
```

使用子查詢彙整巢狀資料

您也可以使用子查詢來彙總巢狀資料。以下範例說明此方法。

```
SELECT c.name.given, c.name.family, (SELECT COUNT(*) FROM c.orders o) AS ordercount
FROM  spectrum.customers c;
```

會傳回下列資料。

given	family	ordercount
Jenny	Doe	0
John	Smith	2
Andy	Jones	1

(3 rows)

Note

依上層資料列群組來彙總巢狀資料時，最有效的方式是先前範例中說明的方式。在該範例中，`c.orders` 的巢狀資料列會依其上層資料列 `c` 群組。或者，如果您知道每個 `customer` 的 `id` 是唯一的，並且 `o.shipdate` 絕不會是 `null`，則可以如以下範例所示彙總。不過，此方法一般不如先前的範例來得有效。

```
SELECT  c.name.given, c.name.family, COUNT(o.shipdate) AS ordercount
FROM    spectrum.customers c LEFT JOIN c.orders o ON true
GROUP BY c.id, c.name.given, c.name.family;
```

您也可以在參照上階查詢別名 (`c`) 的 `FROM` 子句中使用子查詢來編寫查詢，並擷取陣列資料。下列範例示範此方法。

```
SELECT c.name.given, c.name.family, s.count AS ordercount
FROM  spectrum.customers c, (SELECT count(*) AS count FROM c.orders o) s;
```

聯結 Amazon Redshift 和巢狀資料

您也可以在外部資料表中聯結 Amazon Redshift 資料和巢狀資料。例如，假設您在 Amazon S3 中有以下巢狀資料。

```
CREATE EXTERNAL TABLE spectrum.customers2 (  
  id      int,  
  name    struct<given:varchar(20), family:varchar(20)>,  
  phones  array<varchar(20)>,  
  orders  array<struct<shipdate:timestamp, item:int>>  
);
```

同時假設您在 Amazon Redshift 中有以下資料表。

```
CREATE TABLE prices (  
  id int,  
  price double precision  
);
```

下列查詢會基於上述尋找每個客戶購買的總數量和金額。下列範例僅為示範。它只會在您如先前所述建立了資料表時傳回資料。

```
SELECT  c.name.given, c.name.family, COUNT(o.date) AS ordercount, SUM(p.price) AS  
  ordersum  
FROM    spectrum.customers2 c, c.orders o, prices p ON o.item = p.id  
GROUP BY c.id, c.name.given, c.name.family;
```

巢狀資料限制 (預覽)

本主題說明使用 Redshift 頻譜讀取巢狀資料的限制。嵌套數據是包含嵌套字段的數據。巢狀欄位是以單一實體形式結合在一起的欄位，例如陣列、結構或物件。

Note

以下清單中標記 (預覽) 的限制僅適用於預覽叢集，以及在下列區域中建立的預覽工作群組。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)

- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

如需有關設定預覽叢集的詳細資訊，請參閱 Amazon Redshift 管理指南中的[建立預覽叢集](#)。如需設定預覽工作群組的詳細資訊，請參閱 Amazon Redshift 管理指南中的[建立預覽工作群組](#)。

下列限制適用巢狀資料：

- array 或 map 類型可以包含其他 array 或 map 類型，只要嵌套 arrays 上的查詢或 maps 不傳回 scalar 值。(預覽)
- Amazon Redshift Spectrum 僅以外部資料表形式支援複雜資料類型。
- 子查詢結果資料欄必須是最上層。(預覽)
- 如果 OUTER JOIN 表達式參照巢狀資料表，則只能參考該資料表和其巢狀陣列 (和對映)。如果 OUTER JOIN 表達式未參考巢狀資料表，則可以參考任何數量的非巢狀資料表。
- 如果子查詢中的 FROM 子句參考巢狀資料表，則不能參考任何其他資料表。
- 如果子查詢取決於參考上層資料表的巢狀資料表，則子查詢只能在 FROM 子句中使用上層資料表。您無法在任何其他子句中使用上層，例如 SELECT 或 WHERE 子句。例如，下列查詢不會執行，因為子查詢的 SELECT 子句參照上層資料表 c。

```
SELECT c.name.given
FROM   spectrum.customers c
WHERE (SELECT COUNT(c.id) FROM c.phones p WHERE p LIKE '858%') > 1;
```

下列查詢可運作，因為上層 c 僅用於子查詢的 FROM 子句。

```
SELECT c.name.given
FROM   spectrum.customers c
WHERE (SELECT COUNT(*) FROM c.phones p WHERE p LIKE '858%') > 1;
```

- 存取 FROM 子句以外位置巢狀資料的子查詢必須傳回單一值。唯一的例外是 WHERE 子句中的 (NOT) EXISTS 運算子。
- 不支援 (NOT) IN。
- 所有巢狀類型的最大深度為 100。此限制適用於所有檔案格式 (實木複合地板 ORC、離子和 JSON)。
- 存取巢狀資料的彙總子查詢只能參考其 FROM 子句中 (而非外部資料表) 的 arrays 和 maps。

- 不支援查詢 Redshift Spectrum 資料表中巢狀資料的虛擬資料欄。如需詳細資訊，請參閱[虛擬資料欄](#)。
- 當透過在 FROM 子句中指定陣列或映射資料欄來擷取資料時，您只能從這些資料欄中選擇值 (如果值為 scalar)。例如，下列查詢會嘗試從陣列內部的 SELECT 元素。選取 arr.a 的查詢是有效的，因為 arr.a 是一個 scalar 值。第二個查詢不起作用，因為 array 是從 FROM 子句中的 s3.nested_table 中擷取的陣列。(預覽)

```
SELECT array_column FROM s3.nested_table;

array_column
-----
[{"a":1}, {"b":2}]

SELECT arr.a FROM s3.nested_table t, t.array_column arr;

arr.a
-----
1

--This query fails to run.
SELECT array FROM s3.nested_table tab, tab.array_column array;
```

您不能在本身來自另一個陣列或映射的 FROM 子句中使用陣列或映射。若要選取陣列或嵌套在其他陣列內的其他複雜結構，請考慮在 SELECT 陳述式中使用索引。

序列化複雜的嵌套 JSON

本主題示範如何以JSON格式序列化巢狀資料。嵌套數據是包含嵌套字段的數據。巢狀欄位是以單一實體形式結合在一起的欄位，例如陣列、結構或物件。

本教學課程中示範的方法的替代方法是將頂層巢狀集合資料行查詢為序列化JSON。您可以使用序列化來檢查、轉換和擷取巢狀資料，就像使用 Redshift Spectrum JSON 一樣。此方法支援ORC、JSON、離子和鑲木地板格式。使用工作階段組態參數 `json_serialization_enable` 來設定序列化行為。設定時，複雜JSON資料類型會序列化為 VARCHAR (65535)。嵌套JSON可以使用訪問[JSON 函數](#)。如需詳細資訊，請參閱[json_serialization_enable](#)。

例如，如果沒有設定 `json_serialization_enable`，下列直接存取巢狀欄的查詢會失敗。

```
SELECT * FROM spectrum.customers LIMIT 1;
```

```
=> ERROR: Nested tables do not support '*' in the SELECT clause.

SELECT name FROM spectrum.customers LIMIT 1;

=> ERROR: column "name" does not exist in customers
```

設定 `json_serialization_enable` 可讓您直接查詢頂層集合。

```
SET json_serialization_enable TO true;

SELECT * FROM spectrum.customers order by id LIMIT 1;

id | name | phones | orders
---+-----+-----+-----
1 | {"given": "John", "family": "Smith"} | ["123-457789"] | [{"shipdate": "2018-03-01T11:59:59.000Z", "price": 100.50}, {"shipdate": "2018-03-01T09:10:00.000Z", "price": 99.12}]

SELECT name FROM spectrum.customers order by id LIMIT 1;

name
-----
{"given": "John", "family": "Smith"}
```

序列化嵌套JSON時，請考慮以下項目。

- 當集合資料行序列化為 VARCHAR (65535) 時，無法直接存取其巢狀子欄位做為查詢語法的一部分 (例如，在篩選條件子句中)。但是，JSON函數可以用來訪問嵌套JSON。
- 不支援下列特殊表示法：
 - ORC工會
 - ORC具有複雜類型鍵的映射
 - lon 資料包
 - 離子 SEXP
- 時間戳記會傳回為ISO序列化字串。
- 原始映射鍵被提升為字串 (例如，1 到 "1")。
- 頂級空值被序列化為NULLs。
- 如果序列化溢出 65535 的最大大VARCHAR小，則儲存格會設定為。NULL

序列化包含字JSON字符串的複雜類型

默認情況下，嵌套集中包含的字符串值被序列化為轉義JSON字符串。當字符串有效時，轉義可能是不可取的。JSON相反，您可能想要編寫嵌套的子元素或VARCHAR直接為字段。JSON使用 `json_serialization_parse_nested_strings` 工作階段層級組態啟用此行為。同時設定 `json_serialization_enable` 和 `json_serialization_parse_nested_strings` 時，有效JSON值會以內嵌方式序列化，不含逸出字元。當值無效時JSON，會將其逸出，就好像未設定 `json_serialization_parse_nested_strings` 組態值一樣。如需詳細資訊，請參閱 [json_serialization_parse_nested_strings](#)。

例如，假設上一個範例中的資料包含JSON為 `name` VARCHAR (20) 欄位中的structs複雜類型：

```
name
-----
{"given": "{\"first\": \"John\", \"middle\": \"James\"}", "family": "Smith"}
```

設定 `json_serialization_parse_nested_strings` 時，`name` 欄會依照下列方式序列化：

```
SET json_serialization_enable TO true;
SET json_serialization_parse_nested_strings TO true;
SELECT name FROM spectrum.customers order by id LIMIT 1;

name
-----
{"given": {"first": "John", "middle": "James"}, "family": "Smith"}
```

而不是這樣逸出：

```
SET json_serialization_enable TO true;
SELECT name FROM spectrum.customers order by id LIMIT 1;

name
-----
{"given": "{\"first\": \"John\", \"middle\": \"James\"}", "family": "Smith"}
```

HyperLogLog 草圖

本主題說明如何在 Amazon Redshift 中使用 HyperLogLog 草圖。HyperLogLog 是計數分隔問題的演算法，可近似資料集中的不同元素數量。HyperLogLog 草圖是有關資料集的唯一性資料的陣列。

HyperLogLog 是用來估算多組基數的演算法。基數是指多重集中相異值的數目。例如，在 {4,3,6,2,2,6,4,3,6,2,2,3} 的集合中，基數為 4，因為不同的值有 4、3、6 和 2。

HyperLogLog 演算法的精確度（也稱為 m 值）可能會影響預估基數的準確性。在基數估算期間，Amazon Redshift 會使用預設精確度值 15。針對較小的資料集，此值最多可以到 26。因此，平均相對誤差範圍介於 0.01–0.6% 之間。

計算多集的基數時，HyperLogLog 演算法會產生稱為 HLL 草圖的建構。HLL 草圖會封裝多集中不同值的相關資訊。Amazon Redshift 資料類型 HLLSKETCH 代表這類草圖值。此資料類型可用來將草圖儲存在 Amazon Redshift 資料表中。此外，Amazon Redshift 支援可套用至 HLLSKETCH 值作為彙總和純量函數的操作。您可以使用這些函數來擷取的基數，HLLSKETCH 並結合多個 HLLSKETCH 值。

從大型資料集擷取基數時，HLLSKETCH 資料類型可提供顯著的查詢效能優勢。您可以使用 HLLSKETCH 值預先彙總這些資料集，並將其存放在資料表中。Amazon Redshift 可以直接從儲存 HLLSKETCH 的值中擷取基數，而無需存取基礎資料集。

處理 HLL 草圖時，Amazon Redshift 會執行最佳化，將草圖的記憶體佔用空間降到最低，並最大化擷取基數的精確度。Amazon Redshift 在 HLL 草圖、稀疏和密集方面使用兩個表示法。以稀疏格式 HLLSKETCH 啟動。當新值插入到其中時，其大小會增加。當其大小達到密集表示的大小後，Amazon Redshift 會自動將草圖從稀疏轉換為密集。

Amazon Redshift 匯入、匯出和列印 HLLSKETCH，就像草圖是稀疏格式 JSON 一樣。當草圖為密集格式時，Amazon Redshift 會匯入、匯出和列印 HLLSKETCH 做為 Base64 字串。如需的詳細資訊 UNLOAD，請參閱 [卸載數 HLLSKETCH 據類型](#)。若要將文字或逗號分隔值（CSV）資料匯入 Amazon Redshift，請使用 COPY 命令。如需詳細資訊，請參閱 [載入 HLLSKETCH 資料類型](#)。

如需搭配使用函數的資訊 HyperLogLog，請參閱 [HyperLogLog 函數](#)。

主題

- [考量事項](#)
- [限制](#)
- [範例](#)

考量事項

本主題說明 Amazon Redshift HyperLogLog 中的使用詳細資訊。

以下是在 Amazon Redshift HyperLogLog 中使用的考量事項：

- 下列非HyperLogLog 函數可以接受類型 的輸入HLLSKETCH或類型 的欄HLLSKETCH：
 - 彙總函數 COUNT
 - 條件式表達式COALESCE和 NVL
 - CASE 表達式
- 支援的編碼為 RAW。
- 您可以在資料表上執行UNLOAD操作，將HLLSKETCH資料欄輸入文字或 CSV。您可以使用資料 UNLOADHLLSKETCH欄來寫入HLLSKETCH資料。Amazon Redshift 會以稀疏表示的JSON格式顯示資料，或以密集表示的 Base64 格式顯示資料。如需的詳細資訊UNLOAD，請參閱 [卸載數 HLLSKETCH據類型](#)。

以下顯示用於以 格式表示之稀疏 HyperLogLog 草圖的JSON格式。

```
{"version":1,"logm":15,"sparse":{"indices":  
[15099259,33107846,37891580,50065963],"values":[2,3,2,1]}}
```

- 您可以使用 COPY命令將文字或CSV資料匯入 Amazon Redshift。如需詳細資訊，請參閱[載入 HLLSKETCH資料類型](#)。
- 的預設編碼HLLSKETCH為 RAW。如需詳細資訊，請參閱[壓縮編碼](#)。

限制

本主題說明 Amazon Redshift HyperLogLog 中的限制。

以下是在 Amazon Redshift HyperLogLog 中使用的限制：

- Amazon Redshift 資料表不支援資料HLLSKETCH欄作為 Amazon Redshift 資料表的排序索引鍵或分佈索引鍵。
- Amazon Redshift 不支援 ORDER BY、GROUPBY 或 DISTINCT子句中的HLLSKETCH資料欄。
- 您只能將UNLOADHLLSKETCH資料欄轉換為文字或CSV格式。然後，Amazon Redshift 會以JSON格式或 Base64 格式寫入HLLSKETCH資料。如需的詳細資訊UNLOAD，請參閱 [UNLOAD](#)。
- Amazon Redshift 僅支援精確度（logm 值）為 15 的 HyperLogLog 草圖。

- JDBC 和 ODBC 驅動程式不支援 HLLSKETCH 資料類型。因此，結果集會使用 VARCHAR 來表示 HLLSKETCH 值。
- Amazon Redshift Spectrum 原生不支援 HLLSKETCH 資料。因此，您無法建立或修改具有資料 HLLSKETCH 欄的外部資料表。
- Python 使用者定義函數 (UDFs) 的資料類型不支援 HLLSKETCH 該資料類型。如需 Python 的詳細資訊 UDFs，請參閱 [Scalar Python UDFs](#)。

範例

本節包含 HyperLogLog 搭配 Amazon Redshift 使用的範例。

主題

- [範例：在子查詢中傳回基數](#)
- [範例：從子查詢中的合併草圖傳回 HLLSKETCH 類型](#)
- [範例：從合併多個 HyperLogLog 草圖傳回草圖](#)
- [範例：使用外部資料表透過 S3 資料產生 HyperLogLog 草圖](#)

範例：在子查詢中傳回基數

下列範例會針對名為 Sales 的資料表，傳回子查詢中每個草圖的基數。

```
CREATE TABLE Sales (customer VARCHAR, country VARCHAR, amount BIGINT);
INSERT INTO Sales VALUES ('David Joe', 'Greece', 14.5), ('David Joe', 'Greece',
  19.95), ('John Doe', 'USA', 29.95), ('John Doe', 'USA', 19.95), ('George Spanos',
  'Greece', 9.95), ('George Spanos', 'Greece', 2.95);
```

下列查詢會為每個國家/地區的客戶產生 HLL 草圖，並擷取基數。這會顯示了來自每個國家/地區的独特客戶。

```
SELECT hll_cardinality(sketch), country
FROM (SELECT hll_create_sketch(customer) AS sketch, country
      FROM Sales
      GROUP BY country) AS hll_subquery;
```

```
hll_cardinality | country
-----+-----
1              | USA
```

2 | Greece

...

範例：從子查詢中的合併草圖傳回HLLSKETCH類型

下列範例傳回代表子查詢中個別草圖組合的單一HLLSKETCH類型。草圖是使用 HLL_COMBINE 彙總函數來組合。

```
SELECT hll_combine(sketch)
FROM (SELECT hll_create_sketch(customer) AS sketch
      FROM Sales
      GROUP BY country) AS hll_subquery

          hll_combine
-----
{"version":1,"logm":15,"sparse":{"indices":[29808639,35021072,47612452],"values":
[1,1,1]}}
(1 row)
```

範例：從合併多個 HyperLogLog 草圖傳回草圖

在下列範例中，假設 page-users 資料表會針對使用者在指定網站上造訪的每個頁面儲存預先彙總的草圖。此資料表中的每一列都包含一個 HyperLogLog 草圖，代表顯示已造訪頁面IDs的所有使用者。

```
page_users
-- +-----+-----+-----+
-- | _PARTITIONTIME | page          | sketch |
-- +-----+-----+-----+
-- | 2019-07-28     | homepage     | CHAQkAQYA... |
-- | 2019-07-28     | Product A    | CHAQxPnYB... |
-- +-----+-----+-----+
```

下列範例會結合預先彙總的多個草圖並產生單一草圖。此草圖封裝了每個草圖封裝的集合基數。

```
SELECT hll_combine(sketch) as sketch
FROM page_users
```

輸出結果類似如下。

```
-- +-----+
```

```
-- | sketch |
-- +-----+
-- | CHAQ3sGoCxcCIAuCB4iAIBgTIBgqgIAgAwY.... |
-- +-----+
```

建立新的草圖時，您可以使用 HLL_CARDINALITY 函數來取得集體不同的值，如下所示。

```
SELECT hll_cardinality(sketch)
FROM (
  SELECT
    hll_combine(sketch) as sketch
  FROM page_users
) AS hll_subquery
```

輸出結果類似如下。

```
-- +-----+
-- | count |
-- +-----+
-- | 54356 |
-- +-----+
```

範例：使用外部資料表透過 S3 資料產生 HyperLogLog 草圖

下列範例快取 HyperLogLog 草圖，以避免直接存取 Amazon S3 進行基數估算。

您可以在外部資料表中預先彙總和快取 HyperLogLog 草圖，這些資料表已定義用來存放 Amazon S3 資料。透過這樣做，您可以在不存取基礎資料的情況下擷取基數估計值。

例如，假設您已將一組以 Tab 鍵分隔的文字檔卸載至 Amazon S3。您可以執行下列查詢，以定義 Amazon Redshift 外部結構描述 (名為 sales) 中名為 spectrum 的外部資料表。此範例的 Amazon S3 儲存貯體位於美國東部 (維吉尼亞北部) AWS 區域。

```
create external table spectrum.sales(
  salesid integer,
  listid integer,
  sellerid smallint,
  buyerid smallint,
  eventid integer,
  dateid integer,
```

```

qtysold integer,
pricepaid decimal(8,2),
commission decimal(8,2),
saletime timestamp)
row format delimited
fields terminated by '\t' stored as textfile
location 's3://redshift-downloads/ticket/spectrum/sales/';

```

假設您要計算在任意日期購買商品的不同買家。若要這麼做，下列範例會為買方產生一年中每天IDs的草圖，並將結果存放在 Amazon Redshift 資料表中 hll_sales。

```

CREATE TABLE hll_sales AS
SELECT saletime, hll_create_sketch(buyerid) AS sketch
FROM spectrum.sales
GROUP BY saletime;

SELECT TOP 5 * FROM hll_sales;

```

輸出結果類似如下。

```

-- hll_sales

-- | saletime          | sketch
-- |                   |
-- +-----+
+-----+
-- | 7/22/2008 8:30    | {"version":1,"logm":15,"sparse":{"indices":[9281416],"values":
[1]}}
-- | 2/19/2008 0:38    | {"version":1,"logm":15,"sparse":{"indices":[48735497],"values":
[3]}}
-- | 11/5/2008 4:49    | {"version":1,"logm":15,"sparse":{"indices":[27858661],"values":
[1]}}
-- | 10/27/2008 4:08   | {"version":1,"logm":15,"sparse":{"indices":[65295430],"values":
[2]}}
-- | 2/16/2008 9:37    | {"version":1,"logm":15,"sparse":{"indices":[56869618],"values":
[2]}}
-- +-----+
+-----+

```

下列查詢顯示 2008 年感恩節後星期五期間購買項目的不同買家預估數量。

```

SELECT hll_cardinality(hll_combine(sketch)) as distinct_buyers

```

```
FROM hll_sales
WHERE trunc(saletime) = '2008-11-28';
```

輸出結果類似如下。

```
distinct_buyers
-----
386
```

假設您想要取得特定日期範圍內購買商品的不同使用者數量。例如，可能是從感恩節後的星期五到下一個星期一。為了得到此結果，下列查詢會使用 `hll_combine` 彙總函數。此函數可避免重複計算在超過所選範圍一天時購買商品的買家。

```
SELECT hll_cardinality(hll_combine(sketch)) as distinct_buyers
FROM hll_sales
WHERE saletime BETWEEN '2008-11-28' AND '2008-12-01';
```

輸出結果類似如下。

```
distinct_buyers
-----
1166
```

若要保留 `hll_sales` 資料表 up-to-date，請在每天結束時執行下列查詢。這樣做會根據今天購買項目並將其新增至 `hll_sales` 資料表 IDs 的買家的 HyperLogLog 產生草圖。

```
INSERT INTO hll_sales
SELECT saletime, hll_create_sketch(buyerid)
FROM spectrum.sales
WHERE TRUNC(saletime) = to_char(GETDATE(), 'YYYY-MM-DD')
GROUP BY saletime;
```

跨資料庫查詢

本主題描述跨資料庫查詢，這些查詢是在單一 Amazon Redshift 叢集內的多個 Amazon Redshift 資料庫上執行。

透過在 Amazon Redshift 中使用跨資料庫查詢，您可以在 Amazon Redshift 叢集中跨資料庫進行查詢。透過跨資料庫查詢，無論您連線到哪個資料庫，都可以查詢 Amazon Redshift 叢集中任何資料庫的資料。跨資料庫查詢可避免產生資料複本，並簡化您的資料組織，以支援來自相同資料倉儲的多個業務群組。

透過跨資料庫查詢，您可以執行下列操作：

- 查詢 Amazon Redshift 叢集中各個資料庫的資料。

您不僅可以從連線的資料庫中進行查詢，還可以從您有權存取的任何其他資料庫中讀取資料。

當您在任何其他未連線的資料庫上查詢資料庫物件時，您只能讀取這些資料庫物件。您可以使用跨資料庫查詢存取 Amazon Redshift 叢集上任何資料庫的資料，而不必連線到該特定資料庫。這樣做可協助您快速輕鬆地查詢和聯結分散在 Amazon Redshift 叢集中多個資料庫的資料。

您也可以單查詢中結合來自多個資料庫的資料集，並使用商業智慧 (BI) 或分析工具來分析資料。您可以使用標準 Amazon Redshift SQL 命令，繼續為使用者設定精細的資料表層級存取控制。如此一來，您可以協助確保使用者只能看到他們有權存取之資料的相關子集。

- 查詢物件。

您可以使用以三部分標記法表示的完整物件名稱來查詢其他資料庫物件。任何資料庫物件的完整路徑都包含三個元件：資料庫名稱、結構描述和物件名稱。您可以使用完整路徑標記法 (*database_name.schema_name.object_name*)，從任何其他資料庫存取任何物件。若要存取特定資料欄，請使用 *database_name.schema_name.object_name.column_name*。

您也可以使用外部結構描述標記法，為另一個資料庫中的結構描述建立別名。此外部結構描述會參照另一個資料庫和結構描述配對查詢可以使用外部結構描述標記法 (*external_schema_name.object_name*)，來存取其他資料庫物件。

在相同的唯讀查詢中，您可以查詢各種資料庫物件，例如使用者資料表、一般檢視、具體化視觀表，以及來自其他資料庫的近期繫結檢視。

- 管理許可。

有權存取 Amazon Redshift 叢集中任何資料庫中物件的使用者都可以查詢這些物件。您可以使用 [GRANT](#) 命令將權限授予使用者和使用者群組。當使用者不再需要特定資料庫物件的存取權時，您也可以使用 [REVOKE](#) 命令撤銷權限。

- 使用中繼資料和 BI 工具。

您可以建立外部結構描述，參照同一個 Amazon Redshift 叢集內另一個 Amazon Redshift 資料庫中的結構描述。如需詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#) 命令。

建立外部結構描述參照之後，Amazon Redshift 會顯示 [SVV_EXTERNAL_TABLES](#) 和 [SVV_EXTERNAL_COLUMNS](#) 中其他資料庫結構描述下的資料表，以利工具探索中繼資料。

若要整合跨資料庫查詢與 BI 工具，您可以使用下列系統檢視。這些檢視可協助您檢視 Amazon Redshift 叢集上已連線資料庫和其他資料庫中物件中繼資料的相關資訊。

以下是顯示 Amazon Redshift 叢集中所有 Amazon Redshift 物件及所有資料庫的外部物件的系統檢視：

- [SVV_ALL_COLUMNS](#)
- [SVV_ALL_SCHEMAS](#)
- [SVV_ALL_TABLES](#)

以下是顯示 Amazon Redshift 叢集中所有資料庫的所有 Amazon Redshift 物件的系統檢視：

- [SVV_REDSHIFT_COLUMNS](#)
- [SVV_REDSHIFT_DATABASES](#)
- [SVV_REDSHIFT_FUNCTIONS](#)
- [SVV_REDSHIFT_SCHEMAS](#)
- [SVV_REDSHIFT_TABLES](#)

主題

- [考量事項](#)
- [限制](#)
- [跨資料庫查詢範例](#)
- [搭配查詢編輯器使用跨資料庫查詢](#)

考量事項

本主題說明 Amazon Redshift 中跨資料庫查詢的使用詳細資訊。

當您使用 Amazon Redshift 中的跨資料庫查詢功能時，請考慮下列事項：

- Amazon Redshift 支援 ra3.4xlarge、ra3.16xlarge、ra3.xlplus 和 ra3.large 節點類型的跨資料庫查詢。
- Amazon Redshift 支援在同一個 Amazon Redshift 叢集中的一個或多個資料庫上聯結來自資料表或檢視的資料。
- Amazon Redshift Serverless 支援與 Amazon Redshift 叢集相同的跨資料庫功能，因此您可以在無伺服器命名空間中的一或多個資料庫中聯結來自資料表或檢視的資料。
- 連線資料庫上交易中的所有查詢都會讀取其他資料庫中狀態相同的資料，也就是交易開始時的資料狀態。此方法有助於跨資料庫提供查詢的交易一致性。Amazon Redshift 支援跨資料庫查詢的交易一致性。
- 若要跨資料庫取得中繼資料，請使用 SVV_ALL* 和 SVV_REDSHIFT* 中繼資料檢視。您不能使用三部分標記法或外部結構描述來查詢 information_schema 和 pg_catalog 下的跨資料庫中繼資料表或檢視。

限制

本主題說明 Amazon Redshift 中跨資料庫查詢的限制。

當您使用 Amazon Redshift 中的跨資料庫查詢功能時，請留意下列限制：

- 當您在任何其他未連線的資料庫上查詢資料庫物件時，您只能讀取這些資料庫物件。
- 若查詢在參照另一個資料庫物件的其他資料庫上建立，則您無法查詢該檢視。
- 您只能在叢集中其他資料庫的物件上建立近期繫結檢視和具體化視觀表。您無法在叢集中其他資料庫的物件上建立一般檢視。
- Amazon Redshift 不支援具有資料欄層級權限的資料表進行跨資料庫查詢。
- Amazon Redshift 不支援 AWS Glue 或 聯合資料庫上的查詢型錄物件。若要查詢這些物件，請先建立參照每個資料庫中那些外部資料來源的外部結構描述。
- 不支援在具有交錯排序索引鍵的資料表上執行跨資料庫查詢。

跨資料庫查詢範例

本主題包含如何使用跨資料庫查詢的範例。跨資料庫查詢是在單一 Amazon Redshift 叢集內的多個資料庫上執行的查詢。

使用下列範例可協助您了解如何設定參考 Amazon Redshift 資料庫的跨資料庫查詢。

若要開始，請在您的 Amazon Redshift 叢集中建立 db1 和 db2 資料庫及使用者 user1 和 user2。如需詳細資訊，請參閱 [CREATE DATABASE](#) 和 [CREATE USER](#)。

```
--As user1 on db1
CREATE DATABASE db1;

CREATE DATABASE db2;

CREATE USER user1 PASSWORD 'Redshift01';

CREATE USER user2 PASSWORD 'Redshift01';
```

作為 db1 上的 user1，建立資料表、授予 user2 存取權限，以及將值插入 table1。如需詳細資訊，請參閱 [GRANT](#) 和 [INSERT](#)。

```
--As user1 on db1
CREATE TABLE table1 (c1 int, c2 int, c3 int);

GRANT SELECT ON table1 TO user2;

INSERT INTO table1 VALUES (1,2,3),(4,5,6),(7,8,9);
```

作為 db2 上的 user2，使用三部分標記法在 db2 中執行跨資料庫查詢。

```
--As user2 on db2
SELECT * from db1.public.table1 ORDER BY c1;
c1 | c2 | c3
----+-----+----
1  |  2 |  3
4  |  5 |  6
7  |  8 |  9
(3 rows)
```

作為 db2 上的 user2，建立外部結構描述，並使用外部結構描述標記法在 db2 中執行跨資料庫查詢。

```
--As user2 on db2
CREATE EXTERNAL SCHEMA db1_public_sch
FROM REDSHIFT DATABASE 'db1' SCHEMA 'public';

SELECT * FROM db1_public_sch.table1 ORDER BY c1;
```

c1	c2	c3
1	2	3
4	5	6
7	8	9

(3 rows)

若要建立不同的檢視並授予這些檢視的許可，作為 db1 上的 user1，請執行以下操作。

```
--As user1 on db1
CREATE VIEW regular_view AS SELECT c1 FROM table1;

GRANT SELECT ON regular_view TO user2;

CREATE MATERIALIZED VIEW mat_view AS SELECT c2 FROM table1;

GRANT SELECT ON mat_view TO user2;

CREATE VIEW late_bind_view AS SELECT c3 FROM public.table1 WITH NO SCHEMA BINDING;

GRANT SELECT ON late_bind_view TO user2;
```

作為 db2 上的 user2，請使用三部分標記法執行下列跨資料庫查詢，以檢視特定檢視。

```
--As user2 on db2
SELECT * FROM db1.public.regular_view;
c1
----
1
4
7
(3 rows)

SELECT * FROM db1.public.mat_view;
```

```

c2
----
8
5
2
(3 rows)

SELECT * FROM db1.public.late_bind_view;
c3
----
3
6
9
(3 rows)

```

作為 db2 上的 user2，請使用外部結構描述標記法執行下列跨資料庫查詢，以查詢近期繫結檢視。

```

--As user2 on db2
SELECT * FROM db1_public_sch.late_bind_view;
c3
----
3
6
9
(3 rows)

```

作為 db2 上的 user2，請在單一查詢中使用連線的資料表執行以下命令。

```

--As user2 on db2
CREATE TABLE table1 (a int, b int, c int);

INSERT INTO table1 VALUES (1,2,3), (4,5,6), (7,8,9);

SELECT a AS col_1, (db1.public.table1.c2 + b) AS sum_col2, (db1.public.table1.c3 + c)
AS sum_col3 FROM db1.public.table1, table1 WHERE db1.public.table1.c1 = a;
col_1 | sum_col2 | sum_col3
-----+-----+-----
1     | 4        | 6
4     | 10       | 12
7     | 16       | 18
(3 rows)

```

下列範例會列出叢集上的所有資料庫。

```
select database_name, database_owner, database_type
from svv_redshift_databases
where database_name in ('db1', 'db2');
```

database_name	database_owner	database_type
db1	100	local
db2	100	local

(2 rows)

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 結構描述。

```
select database_name, schema_name, schema_owner, schema_type
from svv_redshift_schemas
where database_name in ('db1', 'db2');
```

database_name	schema_name	schema_owner	schema_type
db1	pg_catalog	1	local
db1	public	1	local
db1	information_schema	1	local
db2	pg_catalog	1	local
db2	public	1	local
db2	information_schema	1	local

(6 rows)

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 資料表或檢視。

```
select database_name, schema_name, table_name, table_type
from svv_redshift_tables
where database_name in ('db1', 'db2') and schema_name in ('public');
```

database_name	schema_name	table_name	table_type
db1	public	late_bind_view	VIEW
db1	public	mat_view	VIEW
db1	public	mv_tbl__mat_view__0	TABLE
db1	public	regular_view	VIEW
db1	public	table1	TABLE
db2	public	table2	TABLE

(6 rows)

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 和外部結構描述。

```
select database_name, schema_name, schema_owner, schema_type
from svv_all_schemas where database_name in ('db1', 'db2');
```

database_name	schema_name	schema_owner	schema_type
db1	pg_catalog	1	local
db1	public	1	local
db1	information_schema	1	local
db2	pg_catalog	1	local
db2	public	1	local
db2	information_schema	1	local
db2	db1_public_sch	1	external

(7 rows)

下列範例會列出叢集上所有資料庫的所有 Amazon Redshift 和外部資料表。

```
select database_name, schema_name, table_name, table_type
from svv_all_tables
where database_name in ('db1', 'db2') and schema_name in ('public');
```

database_name	schema_name	table_name	table_type
db1	public	regular_view	VIEW
db1	public	mv_tbl__mat_view__0	TABLE
db1	public	mat_view	VIEW
db1	public	late_bind_view	VIEW
db1	public	table1	TABLE
db2	public	table2	TABLE

(6 rows)

搭配查詢編輯器使用跨資料庫查詢

本主題說明如何搭配查詢編輯器使用跨資料庫查詢。跨資料庫查詢是在單一 Amazon Redshift 叢集內的多個資料庫上執行的查詢。

您可以使用跨資料庫查詢存取 Amazon Redshift 叢集上任何資料庫的資料，而不必連線到該特定資料庫。當您在任何其他未連線的資料庫上執行跨資料庫查詢時，您只能讀取這些資料庫物件。

您可以使用以三部分標記法表示的完整物件名稱來查詢其他資料庫物件。任何資料庫物件的完整路徑都包含三個元件：資料庫名稱、結構描述和物件名稱。一個範例是 *database_name.schema_name.object_name*。

搭配查詢編輯器 v2 使用跨資料庫查詢

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 建立叢集以在 Amazon Redshift 查詢編輯器 v2 中使用跨資料庫查詢。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [建立叢集](#)。
3. 使用適當許可啟用查詢編輯器的存取。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的 [使用查詢編輯器來查詢資料庫](#)。
4. 在導覽選單中，選擇查詢編輯器 v2，然後連線至您叢集的資料庫。

當您第一次連線到查詢編輯器 v2 時，Amazon Redshift 預設會顯示已連線資料庫的資源。

5. 選擇您有權檢視其資料庫物件的其他資料庫。若要檢視物件，請確定您有適當的許可。選擇資料庫之後，Amazon Redshift 會顯示資料庫中的結構描述清單。

選取結構描述以查看該結構描述內的資料庫物件清單。

Note

Amazon Redshift 不能直接支援作為 AWS Glue 或聯合資料庫一部分的查詢目錄物件。若要查詢這些物件，請先建立參照每個資料庫中那些外部資料來源的外部結構描述。具有三部分標記法的 Amazon Redshift 跨資料庫查詢不支援結構描述 `information_schema` 和 `pg_catalog` 下的中繼資料表，因為這些中繼資料檢視專屬於某個資料庫。

6. (選擇性) 篩選您選取之結構描述的資料表或檢視。

Amazon Redshift 中的資料共用

使用 Amazon Redshift，您可以安全地跨 Amazon Redshift 叢集或其他 AWS 服務共用資料。資料共用可讓您共享即時資料，而不必建立複本或移動。資料庫管理員和資料工程師可以使用資料共用來提供安全、唯讀的資料存取，以進行分析，同時維持對資料的控制。資料分析師、商業智慧專業人員和資料科學家可以利用共用資料來獲得洞見，而不會複製或移動資料。常見的使用案例包括與合作夥伴共用資料、啟用跨職能分析，以及促進組織內的資料普及化。下列各節涵蓋在 Amazon Redshift 中設定和管理資料共用的詳細資訊。

使用 Amazon Redshift 資料共用，您可以安全地跨 Amazon Redshift 叢集、工作群組共用即時資料的存取權 AWS 帳戶，AWS 區域而無需手動移動或複製資料。由於資料是即時的，所有使用者都可以在更新後立即在 Amazon Redshift 中查看最 up-to-date 一致且最一致的資訊。

您可以在佈建叢集、無伺服器工作群組 AWS 帳戶、可用區域和之間共用資料 AWS 區域。您可以在叢集類型之間共用，也可以在佈建的叢集和無伺服器之間共用。

您可以共用相同內不同 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組之間讀取和寫入的資料庫物件 AWS 帳戶，或彼此共用 AWS 帳戶。您也可以跨區域寫入資料。您可以授予許可 SELECT，例如 UPDATE 不同資料表的 INSERT、和，以及不同結構描述 CREATE 的 USAGE 和。一旦確認寫入交易，資料即可供所有倉儲使用。

如需在 PREVIEW_2023 追蹤中設定資料共用功能的詳細資訊，請參閱[共用資料寫入存取權（預覽）](#)。

Note

ra3.xlplus 叢集目前無法使用透過資料共用的多倉儲寫入功能。若要使用此功能，請建立 ra3.4xl 叢集、ra3.16xl 叢集或 Amazon Redshift Serverless 工作群組。

資料共用概述

透過 Amazon Redshift，您可以安全地跨 Amazon Redshift 叢集共享即時資料，簡化資料共用工作流程，並減少對複雜擷取、轉換和載入（ETL）程序的需求。資料共用可讓您跨 Amazon Redshift 叢集共用即時資料，無需複製資料即可即時存取最新資料。資料庫管理員、資料分析師和資料工程師可能對此功能感興趣，以簡化團隊和組織內部或跨組織的資料存取和協作。它支援使用案例，例如與分析團隊共用即時生產資料、跨分散式資料來源提供即時報告，以及透過集中控制存取許可來簡化資料治理。下列各節將介紹在 Amazon Redshift 中設定和管理資料共用的詳細資訊。

透過資料共用，您可以安全輕鬆地跨 Amazon Redshift 叢集共用即時資料。

如需如何使用開始使用資料共用和管理資料共用的資訊 AWS Management Console，請參閱 [管理資料共用工作](#)。

Amazon Redshift 的資料共用使用案例

Amazon Redshift 資料共用對於下列使用案例特別有用：

- 支援不同類型的業務關鍵工作負載 – 使用中央擷取、轉換和載入（ETL）叢集，與多個商業智慧（BI）或分析叢集共用資料。此方法可為個別工作負載提供讀取工作負載隔離和計費。您可以根據工作負載的特定價格和效能需求調整個別工作負載運算的大小和規模。
- 實現跨群組協同合作 — 實現跨團隊和業務群組的無縫協作，以實現更廣泛的分析、資料科學和跨產品影響分析。
- 以服務形式提供資料 — 在整個組織中共用資料即服務。
- 在環境之間共用資料 — 在開發、測試和生產環境之間共用資料。您可以透過分享精細層級不同的資料來提高團隊敏捷性。
- 授權存取 Amazon Redshift 中的資料 – 在 AWS Data Exchange 目錄中列出客戶可以在幾分鐘內找到、訂閱和查詢的 Amazon Redshift 資料集。

資料共用寫入存取使用案例（預覽）

寫入的資料共用有幾個重要的使用案例：

- 更新生產者上的業務來源資料 – 您可以將資料作為服務在整個組織中共用，然後消費者也可以對來源資料執行動作。例如，他們可以傳回 up-to-date 值或確認收到資料。這些只是幾個可能的商務使用案例。
- 在生產者上插入其他記錄 – 消費者可以將記錄新增至原始來源資料。如有需要，這些可以標記為來自消費者。

如需如何在資料共用上執行寫入操作的詳細資訊，請參閱 [共用資料寫入存取權（預覽）](#)。

Amazon Redshift 中不同層級的資料共用

您可以使用 Amazon Redshift 在不同層級上共用資料。這些層級包括資料庫、結構描述、資料表、檢視（包括一般、後期繫結和具體化檢視），以及 SQL 使用者定義的函數（UDFs）。您可以為指定資料庫建立多個資料庫。在建立共用的資料庫中，資料共用可以包含多個結構描述的物件。

透過共用資料的這個彈性，您可以獲得精細存取控制。您可以針對需要存取 Amazon Redshift 資料的不同使用者和企業量身打造此控制項。

Amazon Redshift 中資料共用的一致性管理

Amazon Redshift 為所有生產者和消費者叢集提供交易一致性，並與所有消費者共用 up-to-date 資料以及一致的檢視。

您可以持續更新生產者叢集上的資料。交易中取用者叢集上的所有查詢都會讀取共用資料的相同狀態。Amazon Redshift 不會考慮由生產者叢集上的另一個交易所變更的資料，這些資料是在取用者叢集上的交易開始之後遞交的。在生產者叢集上遞交資料變更之後，取用者叢集上的新交易可以立即查詢更新的資料。

強大的一致性消除了共用資料期間低保真報告可能包含無效結果的風險。若是財務分析或結果可能用於準備機器學習模型訓練資料集的情況，此要素尤其重要。

在 Amazon Redshift 中使用資料共用時的考量事項

以下是使用 Amazon Redshift 資料共用的考量事項。如需資料共用限制的資訊，請參閱 [資料共用的限制](#)。

- 跨區域資料共用包含額外的跨區域資料傳輸費用。這些資料傳輸費用不適用於同一區域，但僅適用於不同區域。如需詳細資訊，請參閱[管理跨區域資料共用的成本控制](#)。
- 當您從資料共用讀取資料時，會保持連線至本機叢集資料庫。如需從資料共用建立的資料庫設定和讀取的詳細資訊，請參閱[查詢資料共用物件](#)。
- 取用者需支付查詢生產者資料所需的所有運算和跨區域資料傳輸費用。生產者須支付其佈建叢集或無伺服器命名空間中資料的基礎儲存費用。
- 查詢共用資料的效能取決於取用者叢集的運算容量。

資料共用的叢集加密管理

若要跨 共用資料 AWS 帳戶，生產者和取用者叢集都必須加密。

在 Amazon Redshift 中，您可以為您的叢集開啟資料庫加密，以協助保護靜態資料。開啟叢集的加密時，叢集和其快照的資料區塊和系統中繼資料會加密。您可以在啟動叢集時開啟加密，或修改未加密的叢集來使用 AWS Key Management Service (AWS KMS) 加密。如需 Amazon Redshift 資料庫加密的相關資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 資料庫加密](#)。

為了保護傳輸中的資料，傳輸過程中的所有資料都會透過生產者叢集的加密結構描述加密。取用者叢集會在載入資料時採用此加密結構描述。取用者叢集接著會以一般加密叢集的形式運作。生產者和取用者之間的通訊也會使用共用金鑰結構描述進行加密。如需傳輸中加密的相關資訊，請參閱[傳輸中加密](#)。

資料共用的限制

以下是在 Amazon Redshift 中使用資料共用時的限制：

- 所有佈建RA3叢集類型和 Amazon Redshift Serverless 都支援資料共用。其他叢集類型不支援此功能。
- 如果生產者和取用者叢集和無伺服器命名空間都位於同一個帳戶中，則它們必須具有相同的加密類型（未加密或加密）。在包括 Lake Formation 受管資料共用在內的所有其他情況下，消費者和生產者都必須加密。這是出於安全目的。但是，他們不需要共用相同的加密金鑰。
- 您只能SQLUDFs透過資料共用進行共用。UDFs 不支援 Python 和 Lambda。
- 如果生產者資料庫具有特定的定序，請為取用者資料庫使用相同的定序設定。
- Amazon Redshift 不支援將外部結構描述、資料表或外部資料表上的近期繫結檢視新增至資料共用。
- Amazon Redshift 不支援生產者叢集上的巢狀SQL使用者定義函數。
- Amazon Redshift 不支援使用交錯排序索引鍵的共用資料表，以及參考具有交錯排序索引鍵之資料表的檢視。
- 取用者不能將資料共用物件新增到另一個資料共用。此外，取用者無法將參考資料共用物件的檢視新增到另一個資料共用中。
- Amazon Redshift 不支援存取在準備和執行存取之間同時DDL發生的資料共用物件。
- Amazon Redshift 不支援透過資料共用來共用預存程序。
- Amazon Redshift 不支援共用中繼資料系統檢視和系統資料表。

提供資料共用的區域

下表列出資料共用功能的可用性。

區域	相同區域的資料共用	跨區域資料共用	AWS Lake Formation 受管資料共用
美國東部 (維吉尼亞北部) (us-east-1)	是	是	是

區域	相同區域的資料共用	跨區域資料共用	AWS Lake Formation 受管資料共用
美國東部 (俄亥俄) (us-east-2)	是	是	是
美國西部 (加利佛尼亞北部) (us-west-1)	是	是	是
美國西部 (奧勒岡) (us-west-2)	是	是	是
亞太區域 (香港) (ap-east-1)	是	否	否
亞太區域 (孟買) (ap-south-1)	是	是	是
亞太區域 (海德拉巴) (ap-south-2)	是	否	否
亞太區域 (東京) (ap-northeast-1)	是	是	是
亞太區域 (新加坡) (ap-southeast-1)	是	是	是
亞太區域 (雪梨) (ap-southeast-2)	是	是	是
亞太區域 (雅加達) ; (ap-southeast-3)	是	否	否
亞太區域 (墨爾本) (ap-southeast-4)	是	否	否
亞太區域 (首爾) (ap-northeast-2)	是	是	是

區域	相同區域的資料共用	跨區域資料共用	AWS Lake Formation 受管資料共用
亞太區域 (大阪) (ap-northeast-3)	是	否	否
中國 (北京) (cn-north-1)	是	否	否
非洲 (開普敦) (af-south-1)	是	是	否
加拿大西部 (卡加利) (ca-west-1)	是	否	否
加拿大 (中部) (ca-central-1)	是	是	是
歐洲 (法蘭克福) (eu-central-1)	是	是	是
歐洲 (蘇黎世) (eu-central-2)	是	否	否
歐洲 (愛爾蘭) (eu-west-1)	是	是	是
歐洲 (倫敦) (eu-west-2)	是	是	是
歐洲 (巴黎) (eu-west-3)	是	是	是
歐洲 (米蘭) (eu-south-1)	是	否	否
歐洲 (西班牙) (eu-south-2)	是	否	否

區域	相同區域的資料共用	跨區域資料共用	AWS Lake Formation 受管資料共用
歐洲 (斯德哥爾摩) (eu-north-1)	是	是	是
中東 (UAE) (me-central-1)	是	否	否
中東 (巴林) (me-south-1)	是	否	否
以色列 (特拉維夫) (il-central-1)	是	否	否
南美洲 (聖保羅) (sa-east-1)	是	是	是
AWS GovCloud (美國東部) (us-gov-east-1)	是	否	是
AWS GovCloud (美國西部) (us-gov-west-1)	是	否	是

用於資料共用的多倉儲寫入的區域可用性

在 PREVIEW_2023 追蹤中，資料共用具有寫入操作和更精細共用功能的功能。如需如何設定這些資料的詳細資訊，請參閱[共用資料寫入存取權 \(預覽\)](#)。如需可用預覽功能之區域的相關資訊，請參閱[提供資料共用的區域 \(預覽\)](#)。

Amazon Redshift 資料共用的資料共用

資料共用是在 Amazon Redshift 中共用資料的單位。使用資料共用在相同 AWS 帳戶 或不同的 中共用資料 AWS 帳戶。此外，也可以在不同的 Amazon Redshift 叢集之間共用資料以供讀取用。

每個資料共用都與 Amazon Redshift 叢集中的特定資料庫相關聯。

生產者叢集管理員可以建立資料共用，並新增資料共用物件以與其他叢集共用資料，此動作稱為輸出共用。取用者叢集管理員可以從其他叢集接收資料共用，此動作稱為輸入共用。有關生產者和取用者的詳細資訊，請參閱 [資料共用生產者和取用者](#)。

Datashare 物件是叢集上特定資料庫的物件，生產者叢集管理員可以將其新增至資料共用，以便與資料取用者共用。對於資料取用者而言，資料共用物件是唯讀的。資料共用物件的範例包括資料表、檢視和使用者定義函數。建立資料共用或編輯資料共用時，您可以隨時將資料共用物件新增至資料共用。

調整叢集大小或暫停生產者叢集時，資料共用會繼續運作。

資料共用有不同的類型。

主題

- [標準資料共用](#)
- [AWS Data Exchange 資料共用](#)
- [AWS Lake Formation 管理的資料共用](#)
- [資料共用生產者和取用者](#)
- [Amazon Redshift 詳細資訊的資料共用](#)

標準資料共用

透過標準資料共用，您可以在佈建叢集、無伺服器工作群組 AWS 帳戶、可用區域和 之間共用資料 AWS 區域。您可以在叢集類型之間共用，也可以在佈建的叢集和 Amazon Redshift Serverless 之間共用。

若要共用資料，請注意下列佈建叢集、無伺服器命名空間和 AWS 帳戶 識別符：

- 佈建的叢集命名空間是識別 Amazon Redshift 佈建叢集的識別碼。命名空間全域唯一識別碼 (GUID) 會在佈建叢集建立期間自動建立並連接至叢集。命名空間 Amazon Resource Name (ARN) 採用 `arn : {partition} : redshift : {region} : {account-id} : namespace : {namespace-guid}` 格式。您可以在 Amazon Redshift 主控台的叢集詳細資料頁面上查看已佈建叢集的命名空間。

在資料共用工作流程中，命名空間GUID值和叢集命名空間ARN用於與 中的叢集共用資料 AWS 帳戶。您也可以使用 `current_namespace` 函數尋找目前叢集的命名空間。

- 無伺服器命名空間是識別 Amazon Redshift Serverless 的識別碼。命名空間全域唯一識別碼 (GUID) 會在 Amazon Redshift Serverless 建立期間自動建立並連接至執行個體。無伺服器命名空間ARN採用 `arn : {partition} : redshift-serverless : {region} : {account-id} : namespace / {namespace-guid}` 格式。

- AWS 帳戶 可以是資料共用的取用者，且每個 都以 12 位數 AWS 帳戶 ID 表示。

對於標準共用，請考量到下列內容：

- 刪除生產者叢集時，Amazon Redshift 會刪除生產者叢集所建立的資料共用。備份和還原生產者叢集時，建立的資料共用仍會存在還原的叢集上。但是，授予其他叢集的資料共用權限在還原的叢集上不再有效。警將資料共用的使用許可重新授予所需的取用者叢集。取用者叢集上的取用者資料庫會指向建立快照集之原始叢集的資料共用。若要從還原的叢集查詢共用資料，取用者叢集管理員會建立不同的資料庫。或者，管理員可以捨棄並重新建立現有的使用者資料庫，以使用新還原叢集中的資料共用。
- 刪除取用者叢集並從快照中將其還原時，先前與此叢集共用的存取權將不再有效，也不再可見。如果還原的取用者叢集上仍需要資料共用的存取權，則生產者叢集管理員必須再次將資料共用的使用權授予已還原的取用者叢集。取用者叢集管理員必須捨棄從非作用中資料庫建立的任何過時取用者資料庫。然後，管理員必須在生產者重新授予許可之後，從資料共用中重新建立取用者資料庫。由於已還原叢集的叢集命名空間與原始叢集GUID不同，因此當取用者或生產者叢集從備份還原時，會重新授予資料共用許可。

AWS Data Exchange 資料共用

您可以使用 AWS Data Exchange 資料共用來管理 Amazon Redshift 資料共用的帳單。

AWS Data Exchange Datashare 是授權單位，可透過 共用您的資料 AWS Data Exchange。AWS 管理與 Amazon Redshift 資料共用的訂閱和使用相關聯的所有帳單 AWS Data Exchange 和付款。核准的資料提供者可以將 AWS Data Exchange 資料共用新增至 AWS Data Exchange 產品。當客戶訂閱具有 AWS Data Exchange 資料共用的產品時，他們可以存取產品中的資料共用。

AWS Data Exchange for Amazon Redshift 可讓您方便地透過 授權存取您的 Amazon Redshift 資料 AWS Data Exchange。當客戶訂閱具有 AWS Data Exchange 資料共用的產品時，AWS Data Exchange 會自動將客戶新增為產品隨附所有資料 AWS Data Exchange 共用的資料取用者。發票會自動產生，付款會透過 集中收集並自動支付 AWS Marketplace Entitlement Service。

提供者可以在 Amazon Redshift 中以細微層級授權資料，例如結構描述、資料表、檢視和使用使用者定義的函數。您可以在多個 AWS Data Exchange 產品中使用相同的 AWS Data Exchange 資料共用。新增至 AWS Data Exchange 資料共用的任何物件都可供取用者使用。生產者可以使用 Amazon Redshift API操作、SQL命令和 Amazon Redshift 主控台 AWS Data Exchange，代表他們檢視 管理的所有 AWS Data Exchange 資料共用。訂閱產品 AWS Data Exchange 資料共用的客戶可以唯讀存取資料共用中的物件。

想要使用第三方生產者資料的客戶可以瀏覽 AWS Data Exchange 目錄，以探索和訂閱 Amazon Redshift 中的資料集。AWS Data Exchange 訂閱處於作用中狀態後，他們可以從叢集中的資料共用建立資料庫，並在 Amazon Redshift 中查詢資料。

AWS Data Exchange 資料共用的運作方式

以生產者管理員身分管理 AWS Data Exchange 資料共用

如果您是資料生產者（也稱為上的提供者 AWS Data Exchange），您可以建立連線至 Amazon Redshift 資料庫 AWS Data Exchange 的資料共用。若要將 AWS Data Exchange 資料共用新增至上的產品 AWS Data Exchange，您必須是已註冊的 AWS Data Exchange 提供者。

如需如何開始使用 AWS Data Exchange 資料共用的詳細資訊，請參閱 [在上共用授權的 Amazon Redshift 資料 AWS Data Exchange](#)。

使用 AWS Data Exchange 資料共用作為具有作用中 AWS Data Exchange 訂閱的取用者

如果您是具有作用中 AWS Data Exchange 訂閱的消費者（也稱為上的訂閱者 AWS Data Exchange），您可以在 AWS Data Exchange 主控台上瀏覽 AWS Data Exchange 目錄，以探索包含 AWS Data Exchange 資料共用的產品。

訂閱包含 AWS Data Exchange 資料共用的產品後，請從叢集中的資料共用建立資料庫。然後，您可以直接在 Amazon Redshift 中查詢資料，而無需擷取、轉換和載入資料。

如需如何開始使用 AWS Data Exchange 資料共用的詳細資訊，請參閱 [在上共用授權的 Amazon Redshift 資料 AWS Data Exchange](#)。

對於 AWS Data Exchange 共用，請考量到下列內容：

- 刪除生產者叢集時，Amazon Redshift 會刪除生產者叢集所建立的資料共用。備份和還原生產者叢集時，建立的資料共用仍會存在還原的叢集上。若要讓資料訂閱者能夠繼續存取資料，請再次建立 AWS Data Exchange 資料共用，並將其發佈至產品的資料集。取用者叢集上的取用者資料庫會指向建立快照集之原始叢集的資料共用。若要從還原的叢集查詢共用資料，取用者叢集管理員會建立不同的資料庫，或捨棄並重新建立現有的取用者資料庫，以使用新還原的叢集中新建立 AWS Data Exchange 的資料共用。
- 刪除取用者叢集並從快照中將其還原時，先前與此叢集共用的存取權仍將有效且可見。取用者叢集管理員必須捨棄從非作用中資料共用建立的任何過時取用者資料庫，並在生產者重新授予許可之後，從資料共用重新建立取用者資料庫。由於還原的叢集命名空間與原始叢集GUID不同，因此當生產者叢集從備份還原時，會重新授予資料共用許可。

- 如果您有任何 AWS Data Exchange 資料共用，建議您不要刪除叢集。執行此類修改可能會違反 AWS Data Exchange 中的資料產品條款。

使用 AWS Data Exchange 搭配 Amazon Redshift 時的考量事項

在 AWS Data Exchange Amazon Redshift 中使用時，請考慮下列事項：

- 生產者和消費者都必須使用 RA3 執行個體類型，才能使用 Amazon Redshift 資料共用。生產者必須將 RA3 執行個體類型與最新的 Amazon Redshift 叢集版本搭配使用。
- 生產者和取用者叢集都必須加密。
- 您必須註冊為 AWS Data Exchange 提供者才能列出上的產品 AWS Data Exchange，包括包含 AWS Data Exchange 資料共用的產品。如需詳細資訊，請參閱 [開始成為提供者](#)。
- 您不需要是已註冊的 AWS Data Exchange 提供者，即可透過尋找、訂閱和查詢 Amazon Redshift 資料 AWS Data Exchange。
- 若要控制對資料的存取，請在開啟公開存取設定的情況下建立 AWS Data Exchange 資料共用。若要變更 AWS Data Exchange 資料共用以關閉可公開存取的設定，請將工作階段變數設定為允許 ALTER DATASHARE SET PUBLICACCESSIBLE FALSE。如需詳細資訊，請參閱 [ALTER DATASHARE 用量備註](#)。
- 生產者無法從 AWS Data Exchange 資料共用手動新增或移除取用者，因為資料共用的存取權是根據對包含 AWS Data Exchange 資料共用 AWS Data Exchange 的產品具有作用中訂閱而授予的。
- 生產者無法檢視消費者執行的 SQL 查詢。他們只能透過只有生產者可以存取的 Amazon Redshift 資料表，檢視中繼資料，例如查詢數量或取用者查詢的物件。如需詳細資訊，請參閱 [在 Amazon Redshift 中監視和稽核資料共用](#)。
- 建議您將資料共用設為可公開存取。如果不這樣做，AWS Data Exchange 使用公開存取的消費者叢集的訂閱者將無法使用您的資料共用。
- 建議您不要使用 DROP DATASHARE 陳述式刪除與其他人 AWS 帳戶共用 AWS Data Exchange 的資料共用。如果您這樣做，AWS 帳戶有權存取資料共用的將失去存取權。此動作不可復原。執行此類修改可能會違反 AWS Data Exchange 中的資料產品條款。如果您想要刪除 AWS Data Exchange 資料共用，請參閱 [DROP DATASHARE 用量備註](#)。
- 對於跨區域資料共用，您可以建立 AWS Data Exchange 資料共用以共用授權的資料。
- 使用來自不同區域的資料時，取用者須支付從生產者區域到取用者區域的跨區域資料傳輸費用。

AWS Lake Formation管理的資料共用

使用 Amazon Redshift，您可以透過 AWS Lake Formation 受管資料共用，在 AWS 帳戶和 Amazon Redshift 叢集之間存取和共用即時資料。AWS Lake Formation 資料共用可讓資料提供者安全地與任何取用者共用來自 Amazon S3 資料湖的即時資料，包括其他 AWS 帳戶和 Amazon Redshift 叢集。

使用 AWS Lake Formation，您可以集中定義和強制執行 Amazon Redshift 資料共用的資料庫、資料表、資料欄和資料列層級存取許可，並限制使用者對資料共用內物件的存取。透過 Lake Formation 共用資料，您可以在 Lake Formation 中定義許可，並將這些許可套用至任何資料共用及其物件。例如，如果您的資料表包含員工資訊，您可以使用 Lake Formation 的資料欄層級篩選條件，以防止不在人力資源部門工作的員工查看個人身分資訊（PII），例如社會安全號碼。如需資料篩選的相關資訊，請參閱《AWS Lake Formation 開發人員指南》中的 [Lake Formation 中的資料篩選與儲存格層級安全性](#)。

您也可以使用標籤來設定 Lake Formation 資源的許可。如需詳細資訊，請參閱 [Lake Formation 的標籤式存取控制](#)。

Amazon Redshift 目前在相同帳戶內或跨帳戶共用時，可支援透過 Lake Formation 進行資料共用。目前不支援跨區域共用。

以下是如何使用 Lake Formation 控制資料共用許可的概觀：

1. 在 Amazon Redshift 中，生產者叢集或工作群組管理員會在生產者叢集或工作群組上建立資料共用，並將使用權授予 Lake Formation 帳戶。
2. 生產者叢集或工作群組管理員可授權 Lake Formation 帳戶存取資料共用。
3. Lake Formation 管理員可探索並註冊資料共用。他們還必須探索 AWS Glue ARNs 他們有權存取，並將資料共用與建立關聯 AWS Glue Data Catalog ARN。如果您使用的是 AWS CLI，則可以探索並接受 Redshift CLI 操作 `describe-data-shares` 和 `associate-data-share-consumer` 的資料共用。若要註冊資料共用，請使用 Lake Formation CLI 操作 `register-resource`。
4. Lake Formation 管理員會在 `中` 建立聯合資料庫 AWS Glue Data Catalog，並設定 Lake Formation 許可，以控制使用者對資料共用中物件的存取。如需 `中` 聯合資料庫的詳細資訊 AWS Glue，請參閱 [管理 Amazon Redshift 資料共用 中的資料許可](#)。
5. Lake Formation 管理員探索他們可存取的 AWS Glue 資料庫，並將資料共用與建立關聯 AWS Glue Data Catalog ARN。
6. Redshift 管理員探索 ARNs 他們可存取的 AWS Glue 資料庫，使用資料庫在 Amazon Redshift 取用者叢集中建立外部 AWS Glue 資料庫 ARN，並將用量授予 [使用 IAM 憑證驗證的資料庫使用者](#)，以開始查詢 Amazon Redshift 資料庫。

7. 資料庫使用者可以使用檢視 `SVV_EXTERNAL_TABLES` 和 `SVV_EXTERNAL_COLUMNS` 來尋找他們有權存取的 AWS Glue 資料庫中的所有資料表或資料欄，然後他們可以查詢 AWS Glue 資料庫的資料表。
8. 當生產者叢集或工作群組管理員決定不再與取用者叢集共用資料時，生產者叢集管理員可以撤銷使用權、取消授權或從 Redshift 刪除資料共用。Lake Formation 中的相關許可和物件不會自動刪除。

如需以生產者叢集或工作群組管理員 AWS Lake Formation 身分與共用資料共用的詳細資訊，請參閱 [以生產者身分使用 Lake Formation 管理的資料共用](#)。若要取用生產者叢集或工作群組中的共用資料，請參閱 [以取用者身分使用 Lake Formation 管理的資料共用](#)。

AWS Lake Formation 搭配使用 Amazon Redshift 時的考量和限制

以下是透過 Lake Formation 共用 Amazon Redshift 資料的考量和限制。如需有關資料共用考量和限制的資訊，請參閱 [在 Amazon Redshift 中使用資料共用時的考量事項](#)。如需 Lake Formation 限制的相關資訊，請參閱 [有關在 Lake Formation 中使用 Amazon Redshift 資料庫的注意事項](#)。

- 目前不支援跨區域與 Lake Formation 共用資料共用。
- 如果已針對共用關係上的使用者定義資料欄層級篩選，則執行 `SELECT *` 操作只會傳回使用者有權存取的資料欄。
- 不支援 Lake Formation 的儲存格層級篩選。
- 如果您已建立檢視及其資料表並將其共用至 Lake Formation，則可以設定篩選來管理資料表的存取權，Amazon Redshift 會在取用者叢集使用者存取共用物件時強制執行 Lake Formation 定義的政策。當使用者存取與 Lake Formation 共用的檢視時，Redshift 只會強制執行檢視 (而非檢視中資料表) 上定義的 Lake Formation 政策。但是，當使用者直接存取資料表時，Redshift 會強制執行資料表上已定義的 Lake Formation 政策。
- 如果資料表已設定 Lake Formation 篩選，則您無法根據共用資料表建立取用者的具體化視觀表。
- Lake Formation 管理員必須具有 [資料湖管理員](#) 許可及 [接受資料共用所需的許可](#)。
- 生產者取用者叢集必須是具有最新 Amazon Redshift 叢集版本的 RA3 叢集，或是無伺服器工作群組，才能透過 Lake Formation 共用資料。
- 生產者和取用者叢集都必須加密。
- 將資料共用與 Lake Formation 共用時，在生產者叢集或工作群組中實作的 Redshift 資料列層級和資料欄層級存取控制政策會遭到忽略。Lake Formation 管理員必須在 Lake Formation 中設定這些政策。生產者叢集或工作群組管理員可以使用 [ALTER TABLE](#) 命令 RLS 關閉資料表。
- 透過 Lake Formation 共用資料共用僅適用於可同時存取 Redshift 和 Lake Formation 的使用者。

資料共用生產者和取用者

資料生產者 (也稱為資料共用生產者 (data sharing producer 或 datashare producer)) 是您要從中共用資料的叢集。生產者叢集管理員和資料庫擁有者可以使用 `CREATEDATASHARE` 命令建立資料共用。您可以從您要生產者叢集與取用者叢集共用的資料庫新增物件，例如結構描述、資料表、檢視和 SQL 使用者定義函數 (UDFs)。

資料共用 AWS Data Exchange 的資料生產者 (也稱為上的提供者 AWS Data Exchange) 可以透過授權資料 AWS Data Exchange。核准的提供者可以將 AWS Data Exchange 資料共用 AWS Data Exchange 新增至產品。

當客戶訂閱具有 AWS Data Exchange 資料共用的產品時，AWS Data Exchange 會自動在產品隨附的所有資料 AWS Data Exchange 共用上將客戶新增為資料取用者。AWS Data Exchange 也會在其訂閱結束時從 AWS Data Exchange 資料共用中移除所有客戶。AWS Data Exchange 也會自動管理具有 AWS Data Exchange 資料共用的付費產品的帳單、發票、付款收集和付款分發。如需詳細資訊，請參閱 [AWS Data Exchange 資料共用](#)。若要註冊為 AWS Data Exchange 資料提供者，請參閱 [開始成為提供者](#)。

資料取用者 (也稱為資料共用取用者 (data sharing consumer 或 datashare consumer)) 是從生產者叢集接收資料共用的叢集。

共用資料的 Amazon Redshift 叢集可以位於相同、不同 AWS 帳戶 或不同的 中 AWS 區域，因此您可以跨組織共用資料並與其他方協作。取用者叢集管理員會接收其獲得使用權的資料共用，並檢閱每個資料共用的內容。若要使用共用的資料，取用者叢集管理員會從資料共用中建立 Amazon Redshift 資料庫。然後，管理員會將資料庫的許可指派給取用者叢集中的使用者和角色。授予許可之後，使用者和角色可以將共用物件列為標準中繼資料查詢的一部分，以及取用者叢集上的本機資料。他們可以立即開始查詢。

如果您是擁有作用中 AWS Data Exchange 訂閱的取用者 (也稱為上的訂閱者 AWS Data Exchange)，您可以在 Amazon Redshift 中尋找、訂閱和精細查詢資料，up-to-date 而不需要擷取、轉換和載入資料。如需詳細資訊，請參閱 [AWS Data Exchange 資料共用](#)。

Amazon Redshift 詳細資訊的資料共用

使用 Amazon Redshift，您可以安全地跨 Amazon Redshift 叢集共用即時資料，而不必複製或傳輸資料。Amazon Redshift 的資料共用可讓您與相同或不同 AWS 帳戶和 AWS 區域中的任何 Amazon Redshift 叢集共用即時查詢結果，包括來源資料的更新。

主題

- [Amazon Redshift 中的資料共用狀態值](#)
- [叢集內和叢集之間的資料共用](#)
- [在 Amazon Redshift 資料共用中使用檢視](#)
- [使用 IAM 政策管理對資料共用API操作的存取](#)
- [查詢資料庫](#)

Amazon Redshift 中的資料共用狀態值

本主題說明資料共用在 Amazon Redshift 中可能具有的狀態。

使用跨帳戶資料共用時，會有不同的資料共用狀態，需要您執行動作。您的資料共用的狀態可以是作用中、需要的動作或非作用中。

以下說明每個資料共用狀態及其必要動作：

- 當生產者叢集管理員建立資料共用時，生產者叢集上的資料共用狀態為待授權。生產者叢集管理員可以授權資料取用者存取資料共用。取用者叢集管理員沒有任何動作。
- 當生產者叢集管理員授權資料共用時，生產者叢集上的資料共用狀態會變成已授權。生產者叢集管理員沒有任何動作。當資料共用的資料取用者至少有一個關聯時，資料共用狀態會從已授權變更為作用中。

接著，取用者叢集上資料共用的共用狀態會變成可用 (需要在 Amazon Redshift 主控台上採取動作)。取用者叢集管理員可以將資料共用與資料取用者建立關聯，或拒絕資料共用。取用者叢集管理員也可以使用 AWS CLI 命令 `describeDatashareforConsumer` 來檢視資料共用的狀態。或者，管理員可以使用 CLI 命令 `describeDatashare` 並提供資料共用 Amazon Resource Name (ARN)，以檢視資料共用的狀態。

- 當取用者叢集管理員將資料共用與資料取用者建立關聯時，生產者叢集上的資料共用狀態會變成作用中。當資料共用的資料取用者至少有一個關聯時，資料共用狀態會從已授權變更為作用中。生產者叢集管理員不需要執行任何動作。

取用者叢集上的資料共用狀態會變成作用中。取用者叢集管理員不需要執行任何動作。

- 當取用者叢集管理員從資料共用中移除取用者關聯時，資料共用狀態會變成作用中或已授權。當資料共用與另一個資料取用者之間至少存在一個關聯時，它會變成作用中。當生產者叢集上的資料共用沒有任何取用者關聯時，它會變成已授權。生產者叢集管理員沒有任何動作。

如果移除所有關聯，則資料共用狀態會在取用者叢集上變成需要採取動作。當資料共用可供取用者使用時，取用者叢集管理員可以將資料共用與資料取用者重新建立關聯。

- 當取用者叢集管理員拒絕資料共用時，生產者叢集上的資料共用狀態會在取用者叢集上變成需要採取動作和已拒絕。生產者叢集管理員可以重新授權資料共用。取用者叢集管理員沒有任何動作。
- 當生產者叢集管理員從資料共用中移除授權時，生產者叢集上的資料共用狀態會變成需要採取動作。如有必要，生產者叢集管理員可以選擇重新授權資料共用。取用者叢集管理員不需要執行任何動作。

叢集內和叢集之間的資料共用

只有在不同的 Amazon Redshift 佈建叢集或無伺服器工作群組之間共用資料時，才需要資料共用。在同一個叢集中，只要您擁有其他資料庫中物件的必要權限，就可以使用簡單的三部分標記法 `database.schema.table` 來查詢另一個資料庫。

在 Amazon Redshift 中管理資料共用的許可

身為生產者叢集管理員，您可以保留所共用資料集的控制權。您可以將新物件新增至資料共用，或從資料共用中移除物件。您也可以授予或撤銷取用者叢集、AWS 帳戶或 AWS 區域的整個資料共用存取權。撤銷許可時，取用者叢集會立即失去共用物件的存取權，並停止在 `SVV_` 中的 `INBOUND` 資料共用清單中查看它們 `DATASHARES`。

下列範例會建立資料共用 `salesshare`、新增結構描述 `public`，並將資料表新增至 `public.tickit_sales_redshift salesshare`。它也會 `salesshare` 將的使用許可授予指定的叢集命名空間。

```
CREATE DATASHARE salesshare;

ALTER DATASHARE salesshare ADD SCHEMA public;

ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;

GRANT USAGE ON DATASHARE salesshare TO NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

對於 `CREATE DATASHARE`，超級使用者和資料庫擁有者可以建立資料共用。如需詳細資訊，請參閱 [CREATE DATASHARE](#)。對於 `ALTER DATASHARE`，具有要新增或移除之資料共用物件所需許可的資料共用擁有者可能會變更資料共用。如需相關資訊，請參閱 [ALTER DATASHARE](#)。

身為生產者管理員，當您捨棄資料共用時，資料共用不會再列在取用者叢集上。使用者叢集上從捨棄的資料共用中建立的資料庫和結構描述參考會繼續存在，但其中沒有任何物件。取用者叢集管理員必須手動刪除這些資料庫。

在取用者方面，取用者叢集管理員可以透過從資料共用建立資料庫來決定哪些使用者和角色應該可以存取共用資料。根據您在建立資料庫時所選擇的選項，您可以按照下列方式控制對資料庫的存取。如需從資料共用建立資料庫的詳細資訊，請參閱 [CREATE DATABASE](#)。

如需設定資料共用和從取用者讀取資料的詳細資訊，請參閱 [共用 AWS 帳戶內的資料的讀取存取權](#)。

在沒有子WITHPERMISSIONS句的情況下建立資料庫

管理員可以在資料庫或結構描述層級上控制存取權。若要在結構描述層級上控制存取權，管理員必須從在資料共用中建立的 Amazon Redshift 資料庫建立外部結構描述。

下列範例會授予在資料庫層級和結構描述層級上存取共用資料表的許可。

```
GRANT USAGE ON DATABASE sales_db TO Bob;

CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE sales_db SCHEMA 'public';

GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

若要進一步限制存取權，您可以在共用物件頂端建立檢視，僅公開必要的資料。然後，您可以使用這些視觀表將存取權提供給使用者和角色。

一旦授予使用者資料庫或結構描述的存取權後，他們就可以存取該資料庫或結構描述中的所有共用物件。

使用 WITHPERMISSIONS子句建立資料庫

授予資料庫或結構描述的使用權之後，系統管理員可以使用與本機資料庫或結構描述相同的權限授予程序，進一步控制存取權。如果沒有個別物件許可，即使獲得USAGE許可，使用者也無法存取資料共用資料庫或結構描述中的任何物件。

下列範例會授予在資料庫層級上存取共用資料表的許可。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
GRANT USAGE FOR SCHEMAS IN DATABASE sales_db TO Bob;
GRANT SELECT ON sales_db.public.tickit_sales_redshift TO Bob;
```

授予資料庫或結構描述的存取權之後，使用者仍然需要獲得資料庫或結構描述中任何您要讓他們存取之物件的相關權限。

使用 進行精細共用 WITH PERMISSIONS

讓叢集或 Serverless 工作群組查詢資料共用

此步驟假設資料共用源自您帳戶中的其他叢集或 Amazon Redshift Serverless 命名空間，或者來自另一個帳戶，且已與您正在使用的命名空間相關聯。

1. 取用者資料庫管理員可以從資料共用建立資料庫。

```
CREATE DATABASE my_ds_db [WITH PERMISSIONS] FROM DATASHARE my_datashare OF
  NAMESPACE 'abc123def';
```

如果您建立資料庫WITHPERMISSIONS，您可以將資料共用物件的精細許可授予不同的使用者和角色。如果沒有這樣做，對資料共用資料庫授予USAGE許可的所有使用者和角色都會授予資料共用資料庫中所有物件的所有許可。

2. 以下說明如何將權限授予 Redshift 資料庫使用者或角色。您必須連線到本機資料庫，才能執行這些陳述式。如果您在執行授予陳述式之前在資料共用資料庫上執行USE命令，則無法執行這些陳述式。

```
GRANT USAGE ON DATABASE my_ds_db TO ROLE data_eng;
GRANT CREATE, USAGE ON SCHEMA my_ds_db.my_shared_schema TO ROLE data_eng;
GRANT ALL ON ALL TABLES IN SCHEMA my_ds_db.my_shared_schema TO ROLE data_eng;

GRANT USAGE ON DATABASE my_ds_db TO bi_user;
GRANT USAGE ON SCHEMA my_ds_db.my_shared_schema TO bi_user;
GRANT SELECT ON my_ds_db.my_shared_schema.table1 TO bi_user;
```

在 Amazon Redshift 資料共用中使用檢視

生產者叢集可以共用一般、近期繫結和具體化視觀表。共用一般檢視或近期繫結檢視時，您不需要共用基底資料表。下表顯示資料共用如何支援檢視。

檢視表名稱	可以將此檢視新增到資料共用嗎？	取用者可以在跨叢集的資料共用物件上建立此檢視嗎？
一般檢視	是	否
近期繫結檢視	是	是

檢視表名稱	可以將此檢視新增到資料共用嗎？	取用者可以在跨叢集的資料共用物件上建立此檢視嗎？
具體化視觀表	是	是，但僅限完整的新整理

下面的查詢顯示資料共用支援的一般檢視輸出。如需一般檢視定義的資訊，請參閱 [CREATE VIEW](#)。

```
SELECT * FROM tickit_db.public.myevent_regular_vw
ORDER BY eventid LIMIT 5;
```

```

eventid | eventname
-----+-----
  3835  | LeAnn Rimes
  3967  | LeAnn Rimes
  4856  | LeAnn Rimes
  4948  | LeAnn Rimes
  5131  | LeAnn Rimes
```

下面的查詢顯示資料共用支援的近期繫結檢視輸出。如需近期繫結檢視的詳細資訊，請參閱 [CREATE VIEW](#)。

```
SELECT * FROM tickit_db.public.event_lbv
ORDER BY eventid LIMIT 5;
```

```

eventid | venueid | catid | dateid | eventname | starttime
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
   1    |   305   | 8     | 1851   | Gotterdammerung | 2008-01-25
14:30:00
   2    |   306   | 8     | 2114   | Boris Godunov   | 2008-10-15
20:00:00
   3    |   302   | 8     | 1935   | Salome           | 2008-04-19
14:30:00
   4    |   309   | 8     | 2090   | La Cenerentola (Cinderella) | 2008-09-21
14:30:00
   5    |   302   | 8     | 1982   | Il Trovatore    | 2008-06-05
19:00:00
```

下面的查詢顯示資料共用支援的具體化視觀表輸出。如需具體化視觀表的詳細資訊，請參閱 [CREATE MATERIALIZED VIEW](#)。

```
SELECT * FROM tickit_db.public.tickets_mv;
```

```

catgroup | qty sold
-----+-----
Concerts | 195444
Shows    | 149905

```

您可以維護生產者叢集中所有租用戶的通用資料表。您也可以與取用者叢集共用依維度資料欄 (例如 `tenant_id` (`account_id` 或 `namespace_id`)) 篩選的資料子集。若要這麼做，您可以在基底資料表上定義檢視，並在這些 ID 欄上使用篩選，例如 `current_aws_account = tenant_id`。在取用者方面，當您查詢檢視時，您只會看到符合您帳戶資格的資料列。若要這麼做，您可以使用 Amazon Redshift 內容函數 `current_aws_account` 和 `current_namespace`。

下列查詢會傳回目前 Amazon Redshift 叢集所在的帳戶 ID。如果您已連線到 Amazon Redshift，則可以執行此查詢。

```
select current_user, current_aws_account;
```

```

current_user | current_aws_account
-----+-----
dwuser      | 111111111111
(1 row)

```

下列查詢會傳回目前 Amazon Redshift 叢集的命名空間。如果您已連線到資料庫，則可以執行此查詢。

```
select current_user, current_namespace;
```

```

current_user | current_namespace
-----+-----
dwuser      | 86b5169f-01dc-4a6f-9fbb-e2e24359e9a8
(1 row)

```

資料共用中具體化視觀表的增量重新整理

共用基礎資料表時，Amazon Redshift 支援消費者資料共用中具體化視觀表的增量重新整理。增量重新整理是一種操作，其中 Amazon Redshift 會識別在上一次重新整理後基礎資料表或資料表中的變

更，並僅更新具體化檢視中的對應記錄。如需此行為的詳細資訊，請參閱 [CREATE MATERIALIZED VIEW](#)。

使用 IAM 政策管理對資料共用API操作的存取

若要控制對資料共用API操作的存取，請使用IAM動作型政策。如需有關如何管理IAM政策的資訊，請參閱 IAM 使用者指南 中的 [管理IAM政策](#)。

如需使用資料共用API操作所需的許可資訊，請參閱 Amazon Redshift 管理指南 中的 [使用資料共用API操作所需的許可](#)。

若要讓跨帳戶資料共用更安全，您可以使用 `AuthorizeDataShare` 和 `ConsumerIdentifierDeauthorizeDataShare` API操作的條件式金鑰。透過這樣做，您可以明確控制 AWS 帳戶 可以對兩個API操作進行呼叫的。

對於不是您自己帳戶的取用者，您可以拒絕授權或取消授權資料共用。若要這麼做，請在IAM政策中指定 AWS 帳戶 號碼。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Deny",
      "Action": [
        "redshift:AuthorizeDataShare",
        "redshift:DeauthorizeDataShare"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "redshift:ConsumerIdentifier": "555555555555"
        }
      }
    }
  ]
}
```

您可以在 IAM政策中允許具有的 `DataShareArn testshare2`生產者與具有 `111122223333` AWS 帳戶的消費者明確共用。

```
{
```

```
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "VisualEditor0",
    "Effect": "Allow",
    "Action": [
      "redshift:AuthorizeDataShare",
      "redshift:DeauthorizeDataShare"
    ],
    "Resource": "arn:aws:redshift:us-east-1:666666666666:datashare:af06285e-8a45-4ee9-b598-648c218c8ff1/testshare2",
    "Condition": {
      "StringEquals": {
        "redshift:ConsumerIdentifier": "111122223333"
      }
    }
  }
]
```

查詢資料庫

在 Amazon Redshift 中存取共用資料

您可以使用標準SQL介面或JDBCODBC驅動程式，以及資料來探索共用資料API。您也可以透過熟悉的商業智慧 (BI) 和分析工具，以高效能方式查詢資料。您可以參考其他 Amazon Redshift 資料庫中的物件來執行查詢，只要這些資料庫是您有權存取的叢集的本機和遠端資料庫即可。

只要保持連線至叢集中的本機資料庫，就可以這麼做。然後，您可以從資料庫建立取用者資料庫以使用共用資料。

完成此操作後，您可以連線至取用者資料庫，並使用部分符號查詢共用物件。或者，您可以使用三部分符號 () 連線到本機叢集資料庫，並查詢取用者資料庫中的物件 *consumer_database_name.schema_name.table_name*。您也可以使用取用者資料庫中結構描述的外部結構描述連結進行查詢。您可以同時查詢本機資料和相同查詢內其他叢集共用的資料。這類查詢可以參考來自目前連線資料庫和其他未連線資料庫的物件，包括從資料共用建立的取用者資料庫。

在 Amazon Redshift 中存取資料共用的中繼資料

為了協助叢集管理員探索資料共用，Amazon Redshift 提供了一組中繼資料檢視來列出資料共用。這些檢視會列出叢集中建立的資料共用，以及從相同帳戶內的其他叢集、其他帳戶或其他 AWS 區域接收的資料共用。這些檢視會顯示下列資訊：

- 叢集共用和接收的資料共用
- 資料共用中資料庫物件的內容，包括基本共用中繼資料、物件和取用者

使用 SHOW SCHEMAS 檢視與連線資料庫相關聯的資料共用中共用結構描述的清單。如需詳細資訊，請參閱[SHOW SCHEMAS](#)。

使用 SHOWTABLES 可從與連線資料庫相關聯的資料共用，檢視共用結構描述中的資料表清單。如需詳細資訊，請參閱[SHOW TABLES](#)。

使用 SHOW COLUMNS 檢視與連線資料庫相關聯的資料共用中共用資料表的資料欄清單。如需詳細資訊，請參閱[SHOW COLUMNS](#)。

使用 SVV_DATASHARES 檢視在您的叢集（傳出）中建立並與他人共用（傳入）的所有資料共用清單。如需詳細資訊，請參閱[SVV_DATASHARES](#)。

使用 SVV_DATASHARE_CONSUMERS 檢視資料取用者清單。如需詳細資訊，請參閱[SVV_DATASHARE_CONSUMERS](#)。

使用 SVV_DATASHARE_OBJECTS 檢視叢集（傳出）中建立並與他人共用（傳入）的所有資料共用中的物件清單。如需詳細資訊，請參閱[SVV_DATASHARE_OBJECTS](#)。

將 Amazon Redshift 資料共用與商業智慧工具整合

若要將資料共用與商業智慧（BI）工具整合，建議您使用 Amazon Redshift JDBC 或 ODBC 驅動程式。

Amazon Redshift JDBC 和 ODBC 驅動程式支援驅動程式中的 GetCatalogs API 操作，這會傳回所有資料庫的清單，包括從資料共用建立的資料庫。這些驅動程式也支援下游作業，例如 GetSchemas、GetTables 等，這些作業會從 GetCatalogs 傳回的所有資料庫傳回資料。即使呼叫中未明確指定目錄，驅動程式也會提供此支援。如需 JDBC 或 ODBC 驅動程式的詳細資訊，請參閱[Amazon Redshift 管理指南 中的在 Amazon Redshift 中設定連線](#)。

您可以直接連線至從資料共用建立的取用者資料庫，方式與任何其他資料庫相同。如果您的工具中有連線切換使用者介面，資料庫清單應包含從資料共用建立的資料庫。您可以使用 SVV_REDSHIFT_ 探索這些取用者資料庫 DATABASES。

在 Amazon Redshift 中連線至取用者資料庫

透過直接連線至資料共用資料庫，您可以直接連線至從資料共用建立的資料庫，方法與連線至任何其他類型的 Amazon Redshift 資料庫相同。例如，您可以使用 JDBC 或 ODBC 驅動程式、Amazon Redshift

查詢編輯器 v2 或任何其他可連線至 Amazon Redshift 資料庫的工具，來連線至從資料共用建立的資料庫。如需詳細資訊，請參閱[使用SQL用戶端工具連線至 Amazon Redshift 資料倉儲](#)。

存取共用資料

當您連線到從資料共用建立的資料庫時，您可以使用兩段式表示法 () 來查詢共用物件 `schema_name.table_name`。如果在取用者資料庫搜尋路徑中可以找到資料表，您也可以使用單部分表示法 (`table_name`)。

如果您想要執行跨資料庫查詢，您可以使用三部分表示法 (`consumer_database_name.schema_name.table_name`)。這些查詢可以參考叢集上其他取用者資料庫的共用物件，或本機資料庫的本機物件。他們也可以參考來自相同查詢中其他叢集的本機資料庫和資料。

Note

從資料共用建立的資料庫沒有本機目錄。因此，存取本機目錄資料表的任何查詢，例如 `pg_class`，都會傳回空白結果。

存取共用物件的中繼資料

為了協助叢集管理員探索取用者資料庫中的共用物件，Amazon Redshift 提供一組中繼資料檢視和 SHOW 命令，列出這些物件的中繼資料。當您連線到取用者資料庫時，這些中繼資料檢視和命令不支援跨資料庫中繼資料探索。它們只會傳回與連線資料庫相關聯的資料共用中共用物件的中繼資料。

SHOW SCHEMAS 使用 檢視與連線資料庫相關聯的資料共用中共用結構描述的清單。如需詳細資訊，請參閱[SHOW SCHEMAS](#)。

使用 SHOW TABLES 可從與連線資料庫相關聯的資料共用，檢視共用結構描述中的資料表清單。如需詳細資訊，請參閱[SHOW TABLES](#)。

使用 SHOW COLUMNS 檢視與連線資料庫相關聯的資料共用中共用資料表的資料欄清單。如需詳細資訊，請參閱[SHOW COLUMNS](#)。

使用 SVV_ALL_SCHEMAS 在與連線資料庫相關聯的資料共用中檢視共用結構描述的清單。如需詳細資訊，請參閱[SVV_ALL_SCHEMAS](#)。

使用 SVV_ALL_TABLES 檢視與連線資料庫相關聯的資料共用中的共用資料表清單。如需詳細資訊，請參閱[SVV_ALL_TABLES](#)。

使用 `SVV_ALL_COLUMNS` 檢視與連線資料庫相關聯的資料共用中的共用資料欄清單。如需詳細資訊，請參閱 [SVV_ALL_COLUMNS](#)。

將 Amazon Redshift 資料共用與商業智慧工具整合

若要將資料共用與商業智慧 (BI) 工具整合，建議您使用 Amazon Redshift JDBC 或 ODBC 驅動程式。Amazon Redshift JDBC 和 ODBC 驅動程式支援驅動程式中的 `GetCatalogs` API 操作。此操作會傳回所有資料庫的清單，包括從資料共用建立的資料庫。

這些驅動程式也支援下游操作，例如 `GetSchemas` 和 `GetTables`，這些操作會從 `GetCatalogs` 傳回的所有資料庫傳回資料。即使您未在通話中明確指定目錄，驅動程式也會提供此支援。如需 JDBC 或 ODBC 驅動程式的詳細資訊，請參閱 Amazon Redshift 管理指南 中的 [設定連線](#)。

Amazon Redshift 查詢編輯器 v2 在其連線切換介面中包含取用者資料庫。不過，大多數工具都排除這些資料庫，且僅包含本機叢集資料庫做為可連線的資料庫。

Note

`sys:internal` 已新增名為 `sys:internal` 的新系統資料庫以進行內部維護。有些工具會將此系統資料庫納入為可連線的資料庫。不過，您無法連線到它，也無法針對其物件執行查詢。

在 Amazon Redshift 中監視和稽核資料共用

使用 Amazon Redshift，您可以監控和稽核資料共用活動，以確保合規性和安全性。

透過稽核資料共用，生產者可以追蹤資料共用的演變。例如，稽核有助於追蹤何時建立資料共用、新增或移除物件，以及授予或撤銷 Amazon Redshift 叢集、AWS 帳戶或 AWS 區域的許可。

除了稽核之外，生產者和取用者還可以追蹤各種細微性 (例如帳戶、叢集和物件層級) 的資料共用使用情況。如需追蹤使用情形和稽核檢視的相關資訊，請參閱 [SVL_DATASHARE_CHANGE_LOG](#) 和 [SVL_DATASHARE_USAGE_PRODUCER](#)。

您可以透過查詢系統檢視來監視資料共用。

1. 想要共用資料的生產者叢集管理員會建立 Amazon Redshift 資料共用。然後，生產者叢集管理員會新增所需的資料庫物件。這些可能是資料共用的結構描述、資料表和檢視，並且會指定要與之共用物件的取用者清單。

使用下列系統檢視來查看合併的檢視，以追蹤生產者和/或用戶叢集上資料共用的變更和使用情況：

- [SYS_DATASHARE_CHANGE_LOG](#)
- [SYS_DATASHARE_USAGE_CONSUMER](#)
- [SYS_DATASHARE_USAGE_PRODUCER](#)

使用下列系統檢視來查看輸出資料共用的資料共用物件和資料取用者資訊：

- [SVV_DATASHARES](#)
- [SVV_DATASHARE_CONSUMERS](#)
- [SVV_DATASHARE_OBJECTS](#)

2. 取用者叢集管理員會查看他們被授權使用的資料庫，並使用 [SVV_DATASHARES](#) 檢視輸入資料共用來檢閱每個資料共用的內容。

若要使用共用的資料，每個取用者叢集管理員會從資料共用中建立 Amazon Redshift 資料庫。然後，管理員會將許可指派給取用者叢集中的適當使用者和角色。使用者和角色可以透過檢視下列中繼資料系統檢視，將共用物件列為標準中繼資料查詢的一部分，並且可以立即開始查詢資料。

- [SVV_REDSHIFT_COLUMNS](#)
- [SVV_REDSHIFT_DATABASES](#)
- [SVV_REDSHIFT_FUNCTIONS](#)
- [SVV_REDSHIFT_SCHEMAS](#)
- [SVV_REDSHIFT_TABLES](#)

若要檢視 Amazon Redshift 本機和共用結構描述及外部結構描述的物件，請使用下列中繼資料系統檢視進行查詢。

- [SVV_ALL_COLUMNS](#)
- [SVV_ALL_SCHEMAS](#)
- [SVV_ALL_TABLES](#)

當您連線到取用者資料庫時，會停用跨資料庫探索。中繼資料系統檢視只會傳回與連線資料庫相關聯之資料共用中共用物件的中繼資料。

將 Amazon Redshift 資料共用與 整合 AWS CloudTrail

資料共用與 AWS CloudTrail. CloudTrail 整合，是一種服務，可提供 Amazon Redshift. CloudTrail captures AWS 中使用者、角色或服務所採取動作的記錄。會擷取所有以事件形式進行資料共用的 API 呼叫。擷取的呼叫包括從 AWS CloudTrail 主控台呼叫，以及對資料共用操作的程式碼呼叫。如需 Amazon Redshift 與 整合的詳細資訊 AWS CloudTrail，請參閱 [使用 記錄 CloudTrail](#)。

如需的詳細資訊 CloudTrail，請參閱[如何 CloudTrail 運作](#)。

管理資料共用工作

使用 Amazon Redshift，您可以安全地跨 Amazon Redshift 叢集共用資料，讓資料取用者無需複製或複製即可查詢和存取即時資料。資料共用可讓您建立和設定資料共用，這是參考您要共用之資料庫物件的生產者端物件。

您可以使用 SQL 介面或 Amazon Redshift 主控台來開始資料共用。

主題

- [使用 SQL 介面管理資料共用](#)
- [使用主控台管理資料共用](#)
- [使用 AWS CloudFormation 管理資料共用](#)
- [使用主控台透過寫入功能管理資料共用 \(預覽\)](#)

使用 SQL 介面管理資料共用

使用 Amazon Redshift，您可以使用 SQL 介面控制資料庫和叢集之間的資料存取和共用。管理資料共用可讓您建立資料共用，這是生產者資料庫與一或多個取用者資料庫或叢集之間的生產者-消費者關係。

您可以在 AWS 帳戶內部或 AWS 區域之間的不同 Amazon Redshift 叢集之間共用資料以供讀取。

主題

- [共用 內資料的讀取存取權 AWS 帳戶](#)
- [共用資料的寫入存取權 \(預覽\)](#)
- [跨 共用資料 AWS 帳戶](#)
- [跨 共用資料 AWS 區域](#)
- [在上共用授權的 Amazon Redshift 資料 AWS Data Exchange](#)
- [使用 AWS Lake Formation 受管資料共用](#)

共用 內資料的讀取存取權 AWS 帳戶

使用 Amazon Redshift，您可以在相同 AWS 帳戶中跨不同的資料庫使用者或群組共用資料的讀取存取權。此功能可讓您精細地控制資料存取權限，確保只有授權的使用者或群組才能讀取特定資料集。

以生產者叢集管理員或資料庫擁有者的身分共用資料以供讀取

1. 在叢集中建立資料共用。如需詳細資訊，請參閱[CREATE DATASHARE](#)。

```
CREATE DATASHARE salesshare;
```

叢集超級使用者和資料庫擁有者可以建立資料共用。每個資料共用都會在建立期間與資料庫相關聯。只有來自該資料庫的物件才能在該資料共用中共用。可以在具有相同或不同物件細微程度的相同資料庫上建立多個資料共用。叢集可以建立的資料共用數量不限。

您也可以使用 Amazon Redshift 主控台建立資料共用。如需詳細資訊，請參閱[建立資料共用](#)。

2. 委派在資料共用上操作的許可。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

下列範例政策會授予許給 salesshare 上的 dbuser。

```
GRANT ALTER, SHARE ON DATASHARE salesshare TO dbuser;
```

叢集超級使用者和資料共用的擁有者可以對其他使用者授與或撤銷資料共用的修改許可。

3. 在資料庫共用中新增物件或移除物件。若要將物件新增至資料共用，請在新增物件之前先新增結構描述。當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。務必明確地新增這些物件。如需詳細資訊，請參閱[ALTER DATASHARE](#)。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;  
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;  
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

您也可將檢視新增到資料共用。

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM  
public.tickit_sales_redshift;  
ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;
```

使用 ALTERDATASHARE 可在指定的結構描述中共用結構描述和資料表、檢視和函數。超級使用者、資料共用擁有者，或擁有資料共用 ALTER 或資料共用 ALL 許可的使用者，可以變更資料共用，以新增物件或從中移除物件。使用者應具有在資料共用中新增或移除物件的許可。使用者也應是物件的擁有者 SELECT，或具有物件的 USAGE、或 ALL 許可。

您也可以使用 GRANT 將物件新增至資料共用。此範例顯示如何：

```
GRANT SELECT ON TABLE public.tickit_sales_redshift TO DATASHARE salesshare;
```

此語法在功能上相當於 ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;。

使用 INCLUDENEW子句將指定結構描述中建立的任何新資料表、檢視或SQL使用者定義函數 (UDFs) 新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;  
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除物件。如需詳細資訊，請參閱[將資料共用物件新增到資料共用](#)、[從資料共用中移除資料共用物件](#)及[編輯在您帳戶中建立的資料共用](#)。

4. 在資料共用中新增或移除取用者。下列範例會將取用者叢集命名空間新增至 salesshare。命名空間是帳戶中取用者叢集的命名空間全域唯一識別碼 (GUID)。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

```
GRANT USAGE ON DATASHARE salesshare TO NAMESPACE  
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您只能將許可授予GRANT陳述式中的一個資料共用取用者。

叢集超級使用者和資料共用物件的擁有者，或具有資料共用SHARE許可的使用者，可以將取用者新增至資料共用或從中移除取用者。若要這麼做，他們會使用 GRANTUSAGE或 REVOKE USAGE。

若要尋找您目前看到的叢集命名空間，您可以使用 CURRENT_SELECT NAMESPACE 命令。若要尋找相同 中不同叢集的命名空間 AWS 帳戶，請前往 Amazon Redshift 主控台叢集詳細資訊頁面。在該頁面上，找到新增的命名空間欄位。

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除資料取用者。如需詳細資訊，請參閱 [將資料取用者新增至資料共用](#) 和 [從資料共用中移除資料取用者](#)。

5. (選擇性) 將安全性限制新增至資料共用。下列範例顯示具有公用 IP 存取權的取用者叢集可以讀取資料共用。如需詳細資訊，請參閱[ALTER DATASHARE](#)。

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE = TRUE;
```

您可以在建立資料共用之後修改有關取用者類型的屬性。例如，您可以定義要使用指定資料共用中資料的叢集無法公開存取。若查詢來自不符合資料共用中所指定安全性限制的取用者叢集，則會在查詢執行期遭到拒絕。

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱[編輯在您帳戶中建立的資料共用](#)。

6. 列出在叢集中建立的資料共用，並查看資料共用的內容。

下列範例會顯示名為 salesshare 之資料共用的資訊。如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

```
DESC DATASHARE salesshare;
```

producer_account	producer_namespace	share_type	share_name
object_type	object_name	include_new	
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_users_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_venue_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_category_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_date_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_event_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_listing_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_sales_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
schema	public	t	
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
view	public.sales_data_summary_view		

下列範例顯示生產者叢集上的輸出資料共用。

```
SHOW DATASHARES LIKE 'sales%';
```

輸出結果類似如下。

```
share_name | share_owner | source_database | consumer_database | share_type |
createdate | is_publicaccessible | share_acl | producer_account |
producer_namespace
-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
salesshare | 100 | dev | | OUTBOUND
| 2020-12-09 02:27:08 | True | | 123456789012 |
13b8833d-17c6-4f16-8fe4-1a018f5ed00d
```

如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

您也可以使用 [SVV_DATASHARES](#)、[SVV_DATASHARE_CONSUMERS](#) 和 [SVV_DATASHARE_OBJECTS](#) 來檢視資料共用、資料共用中的物件，以及資料共用取用者。

7. 捨棄資料共用。如需詳細資訊，請參閱[DROP DATASHARE](#)。

您可以在任何時間點使用 [DROP DATASHARE](#) 刪除資料共用物件。叢集超級使用者和資料共用擁有者可以捨棄資料共用。

以下範例會捨棄名為 salesshare 的資料共用。

```
DROP DATASHARE salesshare;
```

您也可以使用 Amazon Redshift 主控台刪除資料共用。如需詳細資訊，請參閱[刪除在您帳戶中建立的資料共用](#)。

8. ALTER DATASHARE 使用 從資料共用隨時移除資料共用中的物件。使用 REVOKE USAGE ON 撤銷對特定取用者的資料共用許可。它撤銷資料共用中物件的USAGE許可，並立即停止對所有取用者叢集的存取。存取權被撤銷後，列出資料共用和中繼資料查詢 (例如列出資料庫和資料表) 不會傳回共用物件。

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱[編輯在您帳戶中建立的資料共用](#)。

9. 如果您不想再與取用者共用資料，請撤銷命名空間中的資料共用存取權。

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱[編輯在您帳戶中建立的資料共用](#)。

以取用者叢集管理員的身分共用資料以供讀取

1. 列出可供您使用的資料共用，並檢視資料共用的內容。如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

下列範例會顯示指定生產者命名空間的輸入資料共用資訊。當您以取用者叢集管理員 DESCDATASHARE 身分執行時，您必須指定檢視傳入資料共用 NAMESPACE 的選項。

```
DESC DATASHARE salesshare OF NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

producer_account	producer_namespace	share_type	share_name
object_type	object_name	include_new	
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_users_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_venue_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_category_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_date_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_event_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_listing_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_sales_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
schema	public		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
view	public.sales_data_summary_view		

只有叢集超級使用者才能執行此動作。您也可以使用 `SVV_DATASHARES` 檢視資料共用，以及使用 `SVV_DATASHARE_OBJECTS` 檢視資料共用中的物件。

下列範例顯示取用者叢集中的輸入資料共用。

```
SHOW DATASHARES LIKE 'sales%';
```

```

share_name | share_owner | source_database | consumer_database | share_type
| createdate | is_publicaccessible | share_acl | producer_account |
producer_namespace
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
salesshare |          |          |          | INBOUND
|          |          |          |          | 123456789012 |
13b8833d-17c6-4f16-8fe4-1a018f5ed00d

```

- 身為資料庫超級使用者，您可以建立參照資料共用的本機資料庫。如需詳細資訊，請參閱 [CREATE DATABASE](#)。

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

如果您想要更精細地控制對本機資料庫中物件的存取，請在建立資料庫時使用 `WITHPERMISSIONS` 子句。這可讓您在步驟 4 中為資料庫中的物件授予物件層級權限。

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您可以透過查詢 [SVV_REDSHIFT_DATABASES](#) 檢視來查看您從資料共用中建立的資料庫。您可以直接連線至這些資料庫，也可以連線至取用者叢集上的本機資料庫，並執行跨資料庫查詢，以查詢來自資料共用資料庫的資料。您不能在現有資料共用中建立的資料庫物件頂端建立資料共用。不過，您可以將資料複製到取用者叢集上的個別資料表中，執行所需的任何處理，然後共用已建立的新物件。

您也可以使用 Amazon Redshift 主控台從資料共用中建立資料共用。如需詳細資訊，請參閱 [從資料共用中建立資料庫](#)。

3. (選擇性) 建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱[CREATE EXTERNAL SCHEMA](#)。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
'public';
```

4. 視需要，將從資料共用建立的資料庫和結構描述參照的許可授予使用者叢集中的使用者和角色。如需詳細資訊，請參閱[GRANT](#) 或 [REVOKE](#)。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

如果您在沒有 WITH 的情況下建立資料庫 PERMISSIONS，您只能將從資料共用建立的整個資料庫的許可指派給您的使用者和角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。如果是這樣，您可以建立指向資料共用中特定結構描述的外部結構描述參考 (如上一個步驟所述)，並在結構描述層級上提供精細的許可。

您也可以用在共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。也可以考慮讓生產者叢集以所需的精細程度為您建立其他資料共用。

如果您在步驟 2 WITH PERMISSIONS 中使用 建立資料庫，則必須為共用資料庫中的物件指派物件層級許可。僅具有 USAGE 許可的使用者在獲得其他物件層級許可 WITH PERMISSIONS 之前，無法存取使用 建立的資料庫中的任何物件。

```
GRANT SELECT ON sales_db.public.ticket_sales_redshift to Bob;
```

5. 查詢資料共用中共用物件的資料。

在取用者資料庫和取用者叢集結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集中的本機物件。若要這麼做，他們會使用 JDBC 或 ODBC 驅動程式或 SVV_ALL 和 SVV_REDSHIFT 檢視。

生產者叢集在每個結構描述中可能會有多個資料庫、資料表和檢視的結構描述。取用者端的使用者只能看到透過資料共用提供的物件子集。這些使用者無法從生產者叢集看到整個中繼資料。此方法有助於透過資料共用提供精細的中繼資料安全控制

您可以繼續連線到本機叢集資料庫。但是現在，您也可以使用三部分 database.schema.table 表示法，從資料共用中建立的資料庫和結構描述中讀取。您可以執行跨越任何和所有可見資料庫的查

詢。這些資料庫可以是叢集上的本機資料庫，也可以是從資料共用建立的資料庫。取用者叢集無法連線至從資料共用建立的資料庫。

您可以使用完整資格存取資料。如需詳細資訊，請參閱[跨資料庫查詢範例](#)。

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;
```

salesid	listid	sellerid	buyerid	eventid	dateid	qtysold	pricepaid	commission	saletime
1	1	36861	21191	7872	1875	4	728.00	109.20	2008-02-18 02:36:48
2	4	8117	11498	4337	1983	2	76.00	11.40	2008-06-06 05:00:16
3	5	1616	17433	8647	1983	2	350.00	52.50	2008-06-06 08:26:17
4	5	1616	19715	8647	1986	1	175.00	26.25	2008-06-09 08:38:52
5	6	47402	14115	8240	2069	2	154.00	23.10	2008-08-31 09:17:02

您只能在共用物件上使用SELECT陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

除了查詢之外，取用者還可以在共用物件上建立檢視。僅支援近期繫結檢視或具體化視觀表。Amazon Redshift 不支援共用資料的一般檢視。取用者建立的檢視可跨越多個本機資料庫或從資料共用建立的資料庫。如需詳細資訊，請參閱[CREATE VIEW](#)。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

共用資料的寫入存取權（預覽）

您可以在相同、帳戶和跨區域內共用不同 Amazon Redshift 叢集或 Amazon Redshift Serverless 工作群組 AWS 帳戶的讀取和寫入資料庫物件。本主題中的程序說明如何設定包含寫入許可的資料共用。您可以授予許可 SELECT，例如 UPDATE 不同資料表的 INSERT、和，以及結構描述 CREATE 的 USAGE 和。一旦確認寫入交易，資料即可供所有倉儲使用。生產者帳戶管理員可以判斷特定命名空間或區域是否取得唯讀 read-and-write，或任何對資料的存取。

以下各節說明如何設定資料共用。這些過程假設您正在佈建的叢集或 Amazon Redshift Serverless 工作群組中的資料庫中工作。

唯讀資料共用與讀寫資料共用

之前資料共用中的物件在所有情況下都是唯讀的。寫入資料共用中的物件是一項新功能。只有當生產者特別將類似 INSERT 或 CREATE 針對物件的寫入權限授予資料共用時，資料共用中的物件才會啟用寫入。此外，對於跨帳戶共用，生產者必須授權資料共用以進行寫入，而取用者必須關聯特定叢集和工作群組以進行寫入。本主題後續章節中的詳細資訊如下。

您可以授予資料共用的許可（預覽）

您可以在資料共用內容中授予不同的物件類型和各種權限。

結構描述：

- USAGE
- CREATE

資料表：

- SELECT
- INSERT
- UPDATE
- DELETE
- TRUNCATE
- DROP
- REFERENCES

函數：

- EXECUTE

資料庫：

- CREATE

預覽版資料共用的要求和限制

- 連線 – 您必須直接連線至資料共用資料庫，或執行 USE 命令以寫入資料共用。不過我們很快將能夠透過三部分記號來實現這項操作。
- 可用性 – 您必須使用無伺服器工作群組、ra3.4xl 叢集或 ra3.16xl 叢集才能使用此功能。
- 中繼資料探索 – 當您透過 Redshift ODBC、JDBC 或 Python 驅動程式直接連線至資料共用資料庫的取用者時，您可以透過下列方式檢視目錄資料：
 - SQL [SHOW](#) 命令。
 - 查詢 information_schema 資料表和視觀表。
 - 查詢 [SVV 中繼資料檢視](#)。
- 資料 API – 您無法透過資料 連線至資料共用資料庫 API。即將提供相關支援。
- 許可可見性 – 取用者看不到授予資料共用的許可。我們很快就會新增這項功能。
- 加密 – 對於跨帳戶資料共用，生產者和取用者叢集都必須加密。
- 隔離層級 – 資料庫的隔離層級必須是快照隔離，才能允許其他無伺服器工作群組和叢集寫入。
- 自動操作 – 寫入資料共用物件的取用者不會觸發自動分析操作。因此，生產者必須在將資料插入資料表後手動執行分析，才能更新資料表統計資料。如果沒這麼做，查詢計劃可能不是最好的。
- 多陳述式查詢和交易 – 目前不支援交易區塊以外的多陳述式查詢。因此，如果您使用 dbeaver 之類的查詢編輯器，並且您有多個寫入查詢，則需要將查詢包裝在明確的 BEGIN...END 交易陳述式中。

SQL 支援的 陳述式

以下陳述式適用於透過寫入進行資料共用的公開預覽版本：

- BEGIN | START TRANSACTION
- END | COMMIT | ROLLBACK
- COPY 沒有 COMPUPDATE
- { CREATE | DROP } SCHEMA

- { CREATE | DROP | SHOW } TABLE
- CREATE TABLE table_name AS
- DELETE
- { GRANT | REVOKE } privilege_name ON OBJECT_TYPE object_name TO consumer_user
- INSERT
- SELECT
- INSERT INTO SELECT
- TRUNCATE
- UPDATE
- 超級資料類型列欄

不支援的陳述式類型 — 不支援下列項目：

- 寫入生產者時，對取用者倉庫進行多陳述式查詢。
- 並行擴縮查詢從取用者寫入生產者。
- 從取用者寫入到生產者的自動複製作業。
- 從取用者到生產者的串流工作寫入。
- 消費者在生產者叢集上建立零ETL整合資料表。如需零ETL整合的詳細資訊，請參閱[使用零ETL整合](#)。
- 使用交錯排序索引鍵寫入資料表。

以生產者帳戶管理員身分，在具有寫入許可的帳戶中共用資料（預覽）

之前資料共用中的物件在所有情況下都是唯讀的。寫入資料共用中的物件是一項新功能。只有當生產者特別將類似INSERT或CREATE針對物件的寫入權限授予資料共用時，資料共用中的物件才會啟用寫入。本主題後續章節中的詳細資訊如下。

如果您正在尋找唯讀資料共用的現有文件，請參閱 [Amazon Redshift 中的跨叢集共用資料](#)。

若要開始資料共用，生產者上的管理員會建立資料共用，並將物件加入至其中：

1. 生產者資料庫擁有者或[超級使用者](#)會建立資料共用。資料共用是資料庫物件、權限和取用者的邏輯容器。(取用者是您的帳戶和其他帳戶中的叢集或 Amazon Redshift Serverless 命名空間。) 每個資料共用都與其建立的資料庫相關聯，只能新增來自該資料庫的物件。以下命令會建立資料共用。

```
CREATE DATASHARE my_datashare [PUBLICACCESSIBLE = TRUE];
```

設定 `PUBLICACCESSIBLE = TRUE` 允許取用者從可公開存取的叢集和佈建的工作群組查詢資料共用。如果您不允許，請將其忽略，或明確地將其設定為 `false`。

資料共用擁有者必須在要新增至資料共用的結構描述 `USAGE` 上授予。 `GRANT` 命令是新的。它用於在結構描述上授予各種動作，包括 `CREATE` 和 `USAGE`。結構描述包含共用物件：

```
CREATE SCHEMA myshared_schema1;
CREATE SCHEMA myshared_schema2;

GRANT USAGE ON SCHEMA myshared_schema1 TO DATASHARE my_datashare;
GRANT CREATE, USAGE ON SCHEMA myshared_schema2 TO DATASHARE my_datashare;
```

或者，管理員可以繼續執行 `ALTER` 命令，將結構描述新增至資料共用。只有以這種方式新增結構描述時才會授予 `USAGE` 許可。

```
ALTER DATASHARE my_datashare ADD SCHEMA myshared_schema1;
```

2. 管理員新增結構描述之後，就可以授予結構描述中物件的資料共用權限。這些是讀取和寫入權限 `GRANT ALL` 範例顯示如何授予所有許可。

```
GRANT SELECT, INSERT ON TABLE myshared_schema1.table1, myshared_schema1.table2,
myshared_schema2.table1
TO DATASHARE my_datashare;

GRANT ALL ON TABLE myshared_schema1.table4 TO DATASHARE my_datashare;
```

您可以繼續執行類似新增資料表 `ALTERDATASHARE` 的命令。執行此操作時，只會對新增的物件授予 `SELECT` 許可。

```
ALTER DATASHARE my_datashare ADD TABLE myshared_schema1.table1,
myshared_schema1.table2, myshared_schema2.table1;
```

3. 系統管理員會將資料共用的使用權授予帳戶中的特定命名空間。您可以在 [ARN 叢集詳細資訊頁面](#)、[Amazon Redshift Serverless 命名空間詳細資訊頁面](#) 或執行命令 `中找到作為一部分的命名空間 IDSELECT current_namespace。如需詳細資訊，請參閱 CURRENT_NAMESPACE。`

```
GRANT USAGE ON DATASHARE my_datashare TO NAMESPACE '86b5169f-012a-234b-9fbb-e2e24359e9a8';
```

跨帳戶共用資料寫入許可 (預覽)

使用 Amazon Redshift，您可以跨 AWS 帳戶共用資料並授予寫入許可，從而在團隊或組織之間實現協作和資料共用。跨帳戶資料共用可讓您建立資料提供者帳戶，以建立和管理資料庫、結構描述和資料表，然後可以安全地與資料消費者帳戶共用。下列各節示範在 Amazon Redshift 中設定跨帳戶資料共用和授予寫入存取權的程序。

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 [PREVIEW_2023](#) 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

如果您尚未在 [PREVIEW_2023](#) 追蹤上建立資料共用，請前往 [共用資料寫入存取權 \(預覽\)](#) 以開始使用。

將共用資料關聯為取用者資料安全管理員 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 [PREVIEW_2023](#) 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

如果您尚未在 [PREVIEW_2023](#) 追蹤上建立資料共用，請前往 [共用資料寫入存取權 \(預覽\)](#) 以開始使用。

先決條件：在生產者管理員授予共用資料庫物件的特定動作之後執行本節所述步驟，如果資料共用正在與其他帳戶共用，則生產者安全管理員會授予存取權。

取用者安全管理員確定以下內容：

- 帳戶中的所有命名空間、帳戶中特定區域的命名空間或特定命名空間，是否都可以存取資料共用。
- 如果命名空間可以存取資料共用，無論這些命名空間是否具有寫入權限。

消費者安全管理員可以透過主控台、CLI或 建立資料共用的關聯API。如果透過 CLI，管理員會使用下列命令：

```
associate-data-share-consumer
--data-share-arn <value>
--consumer-identifier <value>
[--allow-writes | --no-allow-writes]
```

如需 命令的詳細資訊，請參閱 [associate-data-share-consumer](#)。

消費者安全管理員在將資料共用與命名空間建立關聯時，必須明確allow-writes設定為 true，以允許使用 INSERT和 UPDATE命令。如果不這樣做，使用者只能執行讀取操作，例如 SELECT、USAGE或 EXECUTE權限。

您可以透過使用不同的值再次呼叫 associate-data-share-consumer，以變更資料共用命名空間的關聯。舊關聯會被新關聯覆寫，因此，如果您最初有建立關聯並設定 allow-writes，但建立關聯並指定 no-allow-writes，或者只是沒有指定值，則會撤銷取用者的寫入權限。

以生產者安全管理員身分授權資料共用以進行寫入 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 PREVIEW_2023 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

如果您尚未在 PREVIEW_2023 追蹤上建立資料共用，請前往 [共用資料寫入存取權 \(預覽\)](#) 以開始使用。

Note

這僅適用於在帳戶之間共用資料共用時。

生產者安全管理員確定以下內容：

- 是否其他帳戶可以存取資料共用功能。
- 如果某帳戶可以存取資料共用，無論該帳戶是否具有寫入權限。

需要下列IAM許可才能授權資料共用：

redshift : AuthorizeDataShare

您可以使用 CLI 呼叫或 來授權用量和寫入API :

```
authorize-data-share
--data-share-arn <value>
--consumer-identifier <value>
[--allow-writes | --no-allow-writes]
```

如需 命令的詳細資訊，請參閱 [authorize-data-share](#)。

取用者識別碼可以是：

- 十二位數 AWS 帳戶 ID。
- 命名空間識別符 ARN。

請注意，在授權步驟不會授予寫入權限。授權資料共用以進行寫入，這只允許帳戶擁有資料共用管理員授予的寫入權限。如果管理員不允許寫入，則特定取用者可用的唯一許可為 SELECT、USAGE 和 EXECUTE。

您可以透過再次呼叫 `authorize-data-share` 來變更資料共用取用者的授權，但使用的是不同的值。舊授權會被新授權覆寫。因此，如果您最初授權並允許寫入，但重新授權並指定 `no-allow-writes` 或根本不指定值，則取用者將會撤銷其寫入權限。

提供資料共用的區域 (預覽)

使用 Amazon Redshift，您可以在相同或不同 AWS 區域中跨 Amazon Redshift 叢集共用資料，從而啟用跨區域分析、資料駐留和災難復原案例。提供資料共用的區域是指支援從資料共用共用資料的 AWS 區域，該區域是 Amazon Redshift 物件，其中包含可在帳戶或叢集之間共用的一或多個資料庫物件。下列各節提供有關在 Amazon Redshift 中跨區域設定和管理資料共用的詳細資訊。

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 `PREVIEW_2023` 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以在提供資料共用的相同或不同 AWS 區域中，跨 Amazon Redshift 叢集共用即時資料。

如果您尚未在 PREVIEW_2023 追蹤上建立資料共用，請前往[共用資料寫入存取權（預覽）](#)以開始使用。

下列區域提供預覽版的資料共用：

- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國東部 (俄亥俄) (us-east-2)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

跨 共用資料 AWS 帳戶

您可以在 AWS 帳戶之間共用資料以供讀取。跨 共用資料 AWS 帳戶 的運作方式類似於在 帳戶內共用資料。不同之處在於，在 AWS 帳戶之間共用資料時需要雙向交握。生產者帳戶管理員可以授權取用者帳戶存取資料共用，也可以選擇不授權任何存取權。若要使用授權的資料共用，取用者帳戶管理員可以與資料共用建立關聯。管理員可以將資料共用與消費者帳戶中的整個 AWS 帳戶 或特定叢集建立關聯，或拒絕資料共用。如需在帳戶內共用資料的相關資訊，請參閱[共用 內資料的讀取存取權 AWS 帳戶](#)。

資料共用的資料取用者可以是相同帳戶中或不同 AWS 帳戶中的叢集命名空間。您不需要建立單獨的資料共用，即可在帳戶中和帳戶之間共用。

對於跨帳戶資料共用，生產者和取用者叢集都必須加密。

與 共用資料時 AWS 帳戶，生產者叢集管理員會以實體 AWS 帳戶 身分與 共用。取用者叢集管理員可以決定取用者帳戶中的哪些叢集命名空間可以存取資料共用。

主題

- [生產者叢集管理員動作](#)
- [取用者帳戶管理員動作](#)
- [取用者叢集管理員動作](#)

生產者叢集管理員動作

使用 Amazon Redshift，您可以在生產者叢集上執行管理任務，以管理資料擷取和負載處理。

如果您是生產者叢集管理員或資料庫擁有者，請依照下列步驟執行：

1. 在叢集中建立資料共用，並將資料共用物件新增至資料共用。如需如何建立資料共用和將資料共用物件新增至資料共用的詳細步驟，請參閱 [共用內資料的讀取存取權 AWS 帳戶](#)。如需 `CREATEDATASHARE` 和 `ALTER` 的相關資訊 `DATASHARE`，請參閱 [CREATE DATASHARE](#) 和 [ALTER DATASHARE](#)。

下列範例會將不同的資料共用物件加入至資料共用 `salesshare`。

```
-- Add schema to datashare
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;

-- Add table under schema to datashare
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;

-- Add view to datashare
ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;

-- Add all existing tables and views under schema to datashare (does not include
  future table)
ALTER DATASHARE salesshare ADD ALL TABLES in schema public;
```

您也可以使用 Amazon Redshift 主控台建立或編輯資料共用。如需詳細資訊，請參閱 [建立資料共用](#) 和 [編輯在您帳戶中建立的資料共用](#)。

2. 委派在資料共用上操作的許可。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

下列範例政策會授予許給 `salesshare` 上的 `dbuser`。

```
GRANT ALTER, SHARE ON DATASHARE salesshare TO dbuser;
```

叢集超級使用者和資料共用的擁有者可以對其他使用者授與或撤銷資料共用的修改許可。

3. 在資料共用中新增或移除取用者。下列範例會將 AWS 帳戶 ID 新增至 `salesshare`。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012';
```

您只能將許可授予 `GRANT` 陳述式中的一個資料取用者。

叢集超級使用者和資料共用物件的擁有者，或具有資料共用SHARE許可的使用者，可以將取用者新增至資料共用或從中移除取用者。若要這麼做，他們會使用 GRANT USAGE 或 REVOKE USAGE。

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除資料取用者。如需詳細資訊，請參閱 [將資料取用者新增至資料共用](#) 和 [從資料共用中移除資料取用者](#)。

4. (選用) AWS 帳戶 如果您不想再與消費者共用資料，請從 撤銷對資料共用的存取權。

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012';
```

如果您是生產者帳戶管理員，請依照下列步驟執行：

將用量授予 後 AWS 帳戶，資料共用狀態為 pending_authorization。生產者帳戶管理員應該使用 Amazon Redshift 主控台授權資料共用，並選擇資料取用者。

登入 <https://console.aws.amazon.com/redshiftv2/>。然後選擇要授權存取資料共用或從中移除授權的資料取用者。獲授權的資料取用者會收到對資料共用採取動作的通知。如果您要將叢集命名空間新增為資料取用者，則不必執行授權。授權資料取用者之後，他們就可以存取資料共用物件，並建立取用者資料庫來查詢資料。如需詳細資訊，請參閱 [授予或移除資料共用的授權](#)。

取用者帳戶管理員動作

透過 Amazon Redshift，您可以管理消費者帳戶，並控制其對資料倉儲資源的存取。

如果您是取用者帳戶管理員，請依照下列步驟執行：

若要將從其他帳戶共用的一或多個資料共用與您帳戶中的整個 AWS 帳戶 或特定叢集命名空間建立關聯，請使用 Amazon Redshift 主控台。

登入 <https://console.aws.amazon.com/redshiftv2/>。然後，將從其他帳戶共用的一或多個資料共用與您帳戶中的整個 AWS 帳戶 或特定叢集命名空間建立關聯。如需詳細資訊，請參閱 [建立資料共用的關聯](#)。

與 AWS 帳戶 或特定叢集命名空間建立關聯後，資料共用即可供取用。您也可以隨時變更資料共用關聯。將個別叢集命名空間的關聯變更為 時 AWS 帳戶，Amazon Redshift 會使用 AWS 帳戶 資訊覆寫叢集命名空間。將關聯從 AWS 帳戶 變更為特定叢集命名空間時，Amazon Redshift 會使用叢集命名空間 AWS 帳戶 資訊覆寫資訊。帳戶中的所有叢集命名空間都可以存取資料。

取用者叢集管理員動作

使用 Amazon Redshift，您可以在取用者叢集上執行管理任務，以管理資料擷取和負載處理。

如果您是取用者叢集管理員，請依照下列步驟執行：

1. 列出可供您使用的資料共用，並檢視資料共用的內容。只有當生產者叢集管理員已授權資料共用，且取用者叢集管理員已接受資料共用並與之建立關聯時，才能使用資料共用的內容。如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

下列範例會顯示指定生產者命名空間的輸入資料共用資訊。當您以取用者叢集管理員 DESC DATASHARE 身分執行時，您必須指定 NAMESPACE 和 帳戶 ID 以檢視傳入資料共用。對於輸出資料共用，請指定資料共用名稱。

```
SHOW DATASHARES LIKE 'sales%';
```

share_name	share_owner	source_database	consumer_database	share_type	createdate	is_publicaccessible	share_acl	producer_account	producer_namespace
salesshare	t			INBOUND				123456789012	'dd8772e1-d792-4fa4-996b-1870577efc0d'

```
DESC DATASHARE salesshare OF ACCOUNT '123456789012' NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

producer_account	producer_namespace	share_type	share_name	object_type	object_name
123456789012	dd8772e1-d792-4fa4-996b-1870577efc0d	INBOUND	salesshare	table	public.ticket_users_redshift
123456789012	dd8772e1-d792-4fa4-996b-1870577efc0d	INBOUND	salesshare	table	public.ticket_venue_redshift
123456789012	dd8772e1-d792-4fa4-996b-1870577efc0d	INBOUND	salesshare	table	public.ticket_category_redshift
123456789012	dd8772e1-d792-4fa4-996b-1870577efc0d	INBOUND	salesshare	table	public.ticket_date_redshift
123456789012	dd8772e1-d792-4fa4-996b-1870577efc0d	INBOUND	salesshare	table	public.ticket_event_redshift
123456789012	dd8772e1-d792-4fa4-996b-1870577efc0d	INBOUND	salesshare	table	public.ticket_listing_redshift

```

123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND | salesshare |
table | public.ticket_sales_redshift
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d | INBOUND | salesshare |
schema | public
(8 rows)

```

只有叢集超級使用者才能執行此動作。您也可以使用 SVV_DATASHARES 檢視資料共用，並使用 SVV_DATASHARE_OBJECTS 檢視資料共用中的物件。

下列範例顯示取用者叢集中的輸入資料共用。

```
SELECT * FROM SVV_DATASHARES WHERE share_name LIKE 'sales%';
```

```

share_name | share_owner | source_database | consumer_database | share_type
| createdate | is_publicaccessible | share_acl | producer_account |
producer_namespace
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
salesshare | | | | INBOUND |
| t | | | 123456789012 | 'dd8772e1-
d792-4fa4-996b-1870577efc0d'

```

```
SELECT * FROM SVV_DATASHARE_OBJECTS WHERE share_name LIKE 'sales%';
```

```

share_type | share_name | object_type | object_name |
producer_account | producer_namespace
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
INBOUND | salesshare | table | public.ticket_users_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
INBOUND | salesshare | table | public.ticket_venue_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
INBOUND | salesshare | table | public.ticket_category_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
INBOUND | salesshare | table | public.ticket_date_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
INBOUND | salesshare | table | public.ticket_event_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
INBOUND | salesshare | table | public.ticket_listing_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
INBOUND | salesshare | table | public.ticket_sales_redshift |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d

```

```
INBOUND | saleshare | schema | public |
123456789012 | dd8772e1-d792-4fa4-996b-1870577efc0d
(8 rows)
```

2. 建立參照資料共用的本機資料庫。從資料共用建立資料庫時，請指定 NAMESPACE 和 帳戶 ID。如需詳細資訊，請參閱 [CREATE DATABASE](#)。

```
CREATE DATABASE sales_db FROM DATASHARE saleshare OF ACCOUNT '123456789012'
NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

如果您想要更精細地控制對本機資料庫中物件的存取，請在建立資料庫時使用 WITHPERMISSIONS 子句。這可讓您在步驟 4 中為資料庫中的物件授予物件層級權限。

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE saleshare OF ACCOUNT
'123456789012' NAMESPACE 'dd8772e1-d792-4fa4-996b-1870577efc0d';
```

您可以透過查詢 [SVV_REDSHIFT_DATABASES](#) 檢視來查看您從資料共用中建立的資料庫。您可以直接連線至這些資料庫，也可以連線至取用者叢集上的本機資料庫，並執行跨資料庫查詢，以查詢來自資料共用資料庫的資料。您不能在現有資料共用中建立的資料庫物件頂端建立資料共用。不過，您可以將資料複製到取用者叢集上的個別資料表中，執行所需的任何處理，然後共用已建立的新物件。

3. (選擇性) 建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#)。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA
'public';
```

4. 視需要，將從資料共用建立的資料庫和結構描述參照的許可授予使用者叢集中的使用者或角色。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

如果您在沒有 WITH 的情況下建立資料庫 PERMISSIONS，您只能將從資料共用建立的整個資料庫的許可指派給您的使用者或角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。如果是這樣，您可以建立指向資料共用中特定結構描述的外部結構描述參照，如上一個步驟所述。然後，您可以在結構描述層級上提供精細的許可。您也可以在建用物件之上建立近期繫

結檢視，並使用這些檢視來指派精細的許可。也可以考慮讓生產者叢集以所需的精細程度為您建立其他資料共用。您可以依需要，為從資料共用中建立的資料庫建立盡可能多的結構描述參照。

如果您在步驟 2 WITH PERMISSIONS 中使用 建立資料庫，則必須為共用資料庫中的物件指派物件層級許可。僅具有 USAGE 許可的使用者在獲得其他物件層級許可WITHPERMISSIONS之前，無法存取使用 建立的資料庫中的任何物件。

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

5. 查詢資料共用中共用物件的資料。

在取用者資料庫和取用者叢集結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集中的本機物件。若要執行此操作，請使用 JDBC 或 ODBC 驅動程式或 SVV_ALL 和 SVV_REDSHIFT 檢視。

生產者叢集在每個結構描述中可能會有多個資料庫、資料表和檢視的結構描述。取用者端的使用者只能看到透過資料共用提供的物件子集。這些使用者無法從生產者叢集看到所有中繼資料。此方法有助於透過資料共用提供精細的中繼資料安全控制

您可以繼續連線到本機叢集資料庫。但是現在，您也可以使用三部分 database.schema.table 表示法，從資料共用中建立的資料庫和結構描述中讀取。您可以執行跨越任何和所有可見資料庫的查詢。這些資料庫可以是叢集上的本機資料庫，也可以是從資料共用建立的資料庫。取用者叢集無法連線至從資料共用建立的資料庫。

您可以使用完整資格存取資料。如需詳細資訊，請參閱[跨資料庫查詢範例](#)。

```
SELECT * FROM sales_db.public.tickit_sales_redshift;
```

您只能在共用物件上使用SELECT陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

除了執行查詢之外，取用者還可以在共用物件上建立檢視。僅支援近期繫結檢視和具體化視觀表。Amazon Redshift 不支援共用資料的一般檢視。取用者建立的檢視可跨越多個本機資料庫或從資料共用建立的資料庫。如需詳細資訊，請參閱[CREATE VIEW](#)。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
```

```
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

跨 共用資料 AWS 區域

您可以在 AWS 區域中的 Amazon Redshift 叢集之間共用資料以供讀取。透過跨區域資料共用，您可以跨 共用資料，AWS 區域 而無需手動複製資料。您不需要將資料卸載到 Amazon S3，也不需要將資料複製到新的 Amazon Redshift 叢集或執行跨區域快照複製。

透過跨區域資料共用，即使叢集位於不同區域 AWS 帳戶，您也可以在同一或不同中的 AWS 帳戶 叢集之間共用資料。與相同 AWS 帳戶 但不同的 Amazon Redshift 叢集共用資料時 AWS 區域，請遵循與在 中共用資料的相同工作流程 AWS 帳戶。如需詳細資訊，請參閱[共用 內資料的讀取存取權 AWS 帳戶](#)。

如果共用資料的叢集位於不同的 AWS 帳戶 和 中 AWS 區域，您可以遵循與跨消費者叢集共用資料相同的工作流程，AWS 帳戶 並在取用者叢集上包含區域層級關聯。跨區域資料共用支援與整個 AWS 帳戶、整個 AWS 區域或 內特定叢集命名空間的資料共用關聯 AWS 區域。如需跨 共用資料的詳細資訊 AWS 帳戶，請參閱[跨 共用資料 AWS 帳戶](#)。

使用來自不同區域的資料時，取用者須支付從生產者區域到取用者區域的跨區域資料傳輸費用。

若要使用資料共用，取用者帳戶管理員可以透過下列三種方式之一來與資料共用建立關聯。

- 與整個 AWS 帳戶 跨度建立關聯 AWS 區域
- 與 AWS 區域 中的特定 建立關聯 AWS 帳戶
- 與 中特定叢集命名空間的關聯 AWS 區域

當管理員選擇整個 時 AWS 帳戶，AWS 區域 帳戶中不同的所有現有和未來叢集命名空間都可以存取資料共用。消費者帳戶管理員也可以選擇區域中的特定 AWS 區域 或叢集命名空間，以授予他們對資料共用的存取權。

如果您是生產者叢集管理員或資料庫擁有者，請建立資料共用、將資料庫物件和資料取用者新增至資料共用，並將許可授予資料取用者。如需詳細資訊，請參閱[生產者叢集管理員動作](#)。

如果您是生產者帳戶管理員，請使用 AWS Command Line Interface (AWS CLI) 或 Amazon Redshift 主控台授權資料共用，然後選擇資料取用者。

如果您是取用者帳戶管理員，請依照下列步驟執行：

若要將一或多個從其他帳戶共用的資料共用與 內的完整 AWS 帳戶 或特定 AWS 區域 或叢集命名空間 建立關聯 AWS 區域，請使用 Amazon Redshift 主控台。

透過跨區域資料共用，您可以使用 AWS Command Line Interface (AWS CLI) 或 Amazon Redshift 主控台在特定 AWS 區域 中新增叢集。

若要指定一或多個 AWS 區域，您可以使用 `associate-data-share-consumer` CLI 命令搭配選用 `consumer-region` 選項。

使用 CLI，下列範例會將 Salesshare 與整個 AWS 帳戶 與 `associate-entire-account` 選項建立關聯。您一次只能與一個區域建立關聯。

```
aws redshift associate-data-share-consumer
--region {PRODUCER_REGION}
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--associate-entire-account
```

以下範例會將 Salesshare 與美國東部 (俄亥俄) 區域 (`us-east-2`) 產生關聯。

```
aws redshift associate-data-share-consumer
--region {PRODUCER_REGION}
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:0123456789012:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--consumer-region 'us-east-2'
```

下列範例會將 Salesshare 與亞太區域 (雪梨) 區域 () 中另一個 AWS 帳戶 的特定取用者叢集命名空間建立關聯 `ap-southeast-2`。

```
aws redshift associate-data-share-consumer
--data-share-arn arn:aws:redshift:{PRODUCER_REGION}:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/Salesshare
--consumer-arn 'arn:aws:redshift:ap-southeast-2:{CONSUMER_ACCOUNT}:namespace:
{ConsumerImmutableClusterId}'
```

您可以使用 Amazon Redshift 主控台，將資料共用與 中整個 AWS 帳戶 或特定 AWS 區域 或叢集命名空間 建立關聯 AWS 區域。若要執行此操作，請登入 <https://console.aws.amazon.com/redshiftv2/>。然後，將從其他帳戶共用的一或多個資料共用與您的整個 AWS 帳戶 或特定 AWS 區域 或 AWS 區域內的特定叢集命名空間 建立關聯。如需詳細資訊，請參閱 [建立資料共用的關聯](#)。

與 AWS 帳戶 或特定叢集命名空間建立關聯後，資料共用即可供取用。您也可以隨時變更資料共用關聯。將個別叢集命名空間的關聯變更為時 AWS 帳戶，Amazon Redshift 會使用 AWS 帳戶 資訊覆寫叢集命名空間。將關聯從 AWS 帳戶 變更為特定叢集命名空間時，Amazon Redshift 會使用叢集命名空間 AWS 帳戶 資訊覆寫資訊。將整個 關聯變更為 AWS 帳戶 特定 AWS 區域和叢集命名空間時，Amazon Redshift 會使用特定區域和叢集命名空間 AWS 帳戶 資訊覆寫資訊。

如果您是取用者叢集管理員，則可以建立參照資料共用的本機資料庫，並視需要，將從資料共用建立的資料庫許可授予取用者叢集中的使用者或角色。您也可以在建用物件上建立檢視，以及建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱[取用者叢集管理員動作](#)。

管理跨區域資料共用的成本控制

使用 Amazon Redshift，您可以透過設定資料共用來管理跨區域資料共用的成本控制，以限制 AWS 區域之間傳輸的資料量。管理跨區域資料共用的成本控制可讓您設定資料傳輸限制、監控資料傳輸用量，並在接近或超過這些限制時接收通知。

使用來自不同區域的資料時，取用者須支付從生產者區域到取用者區域的跨區域資料傳輸費用。不同區域的資料傳輸價格不同。費用是根據每次成功執行查詢所掃描的資料位元組計算。如需 Amazon Redshift 定價的相關資訊，請參閱[Amazon Redshift 定價](#)。

系統會根據位元組數量向您收費 (四捨五入至下一個 MB)，每個查詢最少 10MB。您可以針對查詢使用情況設定成本控制，並檢視叢集上每個查詢傳輸的資料量。

若要監看和控制跨區域資料共用的使用情況和相關成本，您可以建立每日、每週、每月的用量限制，並定義達到這些限制時 Amazon Redshift 自動採取的動作，以透過可預測性協助維持預算。如需 Amazon Redshift 使用量限制的相關資訊，請參閱[管理 Amazon Redshift 中的用量限制](#)。

根據您設定的用量限制，Amazon Redshift 採取的動作可能是將事件記錄到系統資料表、傳送 CloudWatch 警示，並通知管理員使用 Amazon SNS，或關閉跨區域資料共用以進一步使用。如需這些動作的相關資訊，請參閱[管理 Amazon Redshift 中的用量限制](#)。

若要在 Amazon Redshift 主控台中建立用量限制，請在叢集的動作下選擇設定用量限制。您可以使用叢集效能或監控索引標籤自動產生的 CloudWatch 指標，監控您的用量趨勢，並取得超出定義限制的用量提醒。您也可以使用 AWS CLI 或 Amazon Redshift API 操作，以程式設計方式建立、修改和刪除用量限制。如需詳細資訊，請參閱[管理 Amazon Redshift 中的用量限制](#)。

在上共用授權的 Amazon Redshift 資料 AWS Data Exchange

建立 AWS Data Exchange 資料共用並將其新增至 AWS Data Exchange 產品時，供應商可以在 Amazon Redshift 中授權資料，供消費者在擁有作用中 AWS Data Exchange 訂閱時在 Amazon Redshift 中探索、訂閱和查詢 up-to-date 資料。

將 AWS Data Exchange 資料共用新增至 AWS Data Exchange 產品後，消費者在訂閱啟動時會自動存取產品的資料共用，並在其訂閱處於作用中狀態時保留其存取權。

主題

- [以生產者身分使用 AWS Data Exchange 資料共用](#)
- [以取用者身分使用 AWS Data Exchange 資料共用](#)

以生產者身分使用 AWS Data Exchange 資料共用

使用 Amazon Redshift，您可以透過建立和管理資料共用，以生產者 AWS Data Exchange 身分與共用即時資料產品。

如果您是生產者叢集管理員，請依照下列步驟在 Amazon Redshift 主控台上管理 AWS Data Exchange 資料共用：

1. 在叢集中建立資料共用，以便在上共用資料，AWS Data Exchange 並授予 AWS Data Exchange 對資料共用的存取權。

叢集超級使用者和資料庫擁有者可以建立資料共用。每個資料共用都會在建立期間與資料庫相關聯。只有來自該資料庫的物件才能在該資料共用中共用。可以在具有相同或不同物件細微程度的相同資料庫上建立多個資料共用。可以在叢集中建立的資料共用數量不限。

您也可以使用 Amazon Redshift 主控台建立資料共用。如需詳細資訊，請參閱[建立資料共用](#)。

使用 MANAGEDBYADX 選項，在執行 CREATEDATASHARE 陳述式 AWS Data Exchange 時隱含授予對的資料共用存取權。這表示 AWS Data Exchange 管理此資料共用。您只能在建立新的資料共用時使用 MANAGEDBYADX 選項。您不能使用 ALTERDATASHARE 陳述式來修改現有的資料共用以新增 MANAGEDBYADX 選項。使用 MANAGEDBYADX 選項建立資料共用後，只能 AWS Data Exchange 存取和管理資料共用。

```
CREATE DATASHARE salesshare
[[SET] MANAGEDBY [=] {ADX} ];
```

- 將物件新增到資料共用。生產者管理員會繼續管理資料共用中可用的 AWS Data Exchange 資料共用物件。

若要將物件新增至資料共用，請在新增物件之前先新增結構描述。當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。您必須明確地新增這些物件。如需詳細資訊，請參閱 [ALTER DATASHARE](#)。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;  
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;  
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

您也可將檢視新增到資料共用。

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM  
public.tickit_sales_redshift;  
ALTER DATASHARE salesshare ADD TABLE public.sales_data_summary_view;
```

使用 ALTERDATASHARE來共用指定結構描述中的結構描述、資料表、檢視和函數。超級使用者、資料共用擁有者或擁有資料共用ALTER或ALL許可的使用者，可以變更資料共用，以新增物件或從中移除物件。使用者應具有在資料共用中新增或移除物件的許可。使用者也應是物件的擁有者 SELECT，或擁有物件的 USAGE、或 ALL許可。

使用 INCLUDENEW子句將在指定結構描述中建立的任何新資料表、檢視或SQL使用者定義函數（UDFs）新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;  
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

您也可以使用 Amazon Redshift 主控台在資料共用中新增或移除物件。如需詳細資訊，請參閱 [將資料共用物件新增到資料共用](#)、[從資料共用中移除資料共用物件](#)及[編輯 AWS Data Exchange 資料共用](#)。

- 若要授權存取 的資料共用 AWS Data Exchange，請執行下列其中一項操作：
 - AWS Data Exchange 使用 `aws redshift authorize-data-share` 中的ADX關鍵字明確授權 的資料共用存取權API。這允許 AWS Data Exchange 識別服務帳戶中的資料共用，並管理將取用者與資料共用建立關聯。

```
aws redshift authorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier ADX
```

您可以使用 `AuthorizeDataShare` 和 `ConsumerIdentifier` 的條件式金鑰 `DeauthorizeDataShare` APIs，明確允許或拒絕對 IAM 政策 APIs 中的兩者 AWS Data Exchange 進行呼叫。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Deny",
      "Action": [
        "redshift:AuthorizeDataShare",
        "redshift:DeauthorizeDataShare"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIgnoreCase": {
          "redshift:ConsumerIdentifier": "ADX"
        }
      }
    }
  ]
}
```

- 使用 Amazon Redshift 主控台來授權或移除 AWS Data Exchange 資料共用的授權。如需詳細資訊，請參閱[授予或移除資料共用的授權](#)。
- 或者，您可以在將 AWS Data Exchange 資料共用匯入 AWS Data Exchange 資料集時隱含授權存取資料共用。

若要移除存取 AWS Data Exchange 資料共用的授權，請在 `aws redshift deauthorize-data-share` API 操作中使用 `ADX` 關鍵字。透過這樣做，您可以讓 AWS Data Exchange 識別服務帳戶中的資料共用，並管理從資料共用中刪除關聯的操作。

```
aws redshift deauthorize-data-share
```

```
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier ADX
```

4. 列出在叢集中建立的資料共用，並查看資料共用的內容。

下列範例會顯示名為 salesshare 之資料共用的資訊。如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

```
DESC DATASHARE salesshare;
```

producer_account	producer_namespace	share_type	share_name
object_type	object_name	include_new	
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_users_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_venue_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_category_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_date_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_event_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_listing_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
table	public.tickit_sales_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
schema	public		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	OUTBOUND	salesshare
view	public.sales_data_summary_view		

下列範例顯示生產者叢集上的輸出資料共用。

```
SHOW DATASHARES LIKE 'sales%';
```

輸出結果類似如下。

```

share_name | share_owner | source_database | consumer_database | share_type |
createdate | is_publicaccessible | share_acl | producer_account |
producer_namespace
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----
salesshare | 100 | dev | | OUTBOUND
| 2020-12-09 02:27:08 | True | | 123456789012 |
13b8833d-17c6-4f16-8fe4-1a018f5ed00d

```

如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

您也可以使用 [SVV_DATASHARES](#)、[SVV_DATASHARE_CONSUMERS](#) 和 [SVV_DATASHARE_OBJECTS](#) 來檢視資料共用、資料共用中的物件，以及資料共用取用者。

5. 捨棄資料共用。建議您不要刪除 AWS 帳戶使用 DROPDATASHARE 陳述式與其他共用 AWS Data Exchange 的資料共用。這些帳戶將會失去資料共用的存取權。此動作不可復原。這可能會違反中的資料產品優惠條款 AWS Data Exchange。如果您想要刪除 AWS Data Exchange 資料共用，請參閱 [DROP DATASHARE 用量備註](#)。

以下範例會捨棄名為 salesshare 的資料共用。

```

DROP DATASHARE salesshare;
ERROR: Drop of ADX-managed datashare salesshare requires session variable
datashare_break_glass_session_var to be set to value '620c871f890c49'

```

若要允許捨棄 AWS Data Exchange 資料共用，請設定 datashare_break_glass_session_var 變數，然後再次執行 DROPDATASHARE 陳述式。如果您想要刪除 AWS Data Exchange 資料共用，請參閱 [DROP DATASHARE 用量備註](#)。

您也可以使用 Amazon Redshift 主控台刪除資料共用。如需詳細資訊，請參閱 [刪除在您的帳戶中建立 AWS Data Exchange 的資料共用](#)。

6. ALTER DATASHARE 使用 從資料共用隨時移除資料共用中的物件。使用 REVOKE USAGE ON 撤銷對特定取用者的資料共用許可。它會撤銷資料共用中物件的 USAGE 許可，並立即停止對所有取用者叢集的存取。存取權被撤銷後，列出資料共用和中繼資料查詢 (例如列出資料庫和資料表) 不會傳回共用物件。

```

ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;

```

您也可以使用 Amazon Redshift 主控台編輯資料共用。如需詳細資訊，請參閱[編輯 AWS Data Exchange 資料共用](#)。

7. 授予或撤銷 GRANT USAGE AWS Data Exchange 資料共用。您無法授予或撤銷 GRANT USAGE AWS Data Exchange 資料共用。下列範例顯示為 AWS Data Exchange 管理 AWS 帳戶的資料共用授予 GRANT USAGE 許可時的錯誤。

```
CREATE DATASHARE salesshare MANAGEDBY ADX;
```

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '012345678910';  
ERROR: Permission denied to add/remove consumer to/from datashare salesshare.  
Datashare consumers are managed by ADX.
```

如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

如果您是生產者叢集管理員，請依照下列步驟在 AWS Data Exchange 主控台上建立和發佈資料共用產品：

- 建立 AWS Data Exchange 資料共用後，生產者會建立新的資料集、匯入資產、建立修訂，以及建立和發佈新產品。

使用 Amazon Redshift 主控台建立資料集。如需詳細資訊，請參閱[在上建立資料集 AWS Data Exchange](#)。

如需詳細資訊，請參閱[在上提供資料產品 AWS Data Exchange](#)。

以取用者身分使用 AWS Data Exchange 資料共用

使用 Amazon Redshift，您可以從存取和分析資料集，AWS Data Exchange 而不必儲存或管理資料複本。

如果您是消費者，請依照下列步驟探索包含 AWS Data Exchange 資料共用的資料產品，並查詢 Amazon Redshift 資料：

1. 在 AWS Data Exchange 主控台上，探索並訂閱包含資料共用 AWS Data Exchange 的資料產品。

訂閱啟動後，您可以存取已授權的 Amazon Redshift 資料，這些資料會以資產的形式匯入包含 AWS Data Exchange 資料共用的資料集。

如需如何開始使用包含 AWS Data Exchange 資料共用的資料產品的詳細資訊，請參閱 [上的訂閱資料產品 AWS Data Exchange](#)。

2. 如有需要，在 Amazon Redshift 主控台上，建立 Amazon Redshift 叢集。

如需如何建立叢集的資訊，請參閱[建立叢集](#)。

3. 列出可供您使用的資料共用，並檢視資料共用的內容。如需詳細資訊，請參閱 [DESC DATASHARE](#) 和 [SHOW DATASHARES](#)。

下列範例會顯示指定生產者命名空間的輸入資料共用資訊。當您以取用者叢集管理員 DESCDATASHARE 身分執行時，您必須指定 ACCOUNT 和 NAMESPACE 選項，以檢視傳入資料共用。

```
DESC DATASHARE salesshare of ACCOUNT '123456789012' NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

producer_account	producer_namespace	share_type	share_name
object_type	object_name	include_new	
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_users_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_venue_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_category_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_date_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_event_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_listing_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
table	public.tickit_sales_redshift		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
schema	public		
123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	INBOUND	salesshare
view	public.sales_data_summary_view		

只有叢集超級使用者才能執行此動作。您也可以使用 SVV_DATASHARES 檢視資料共用，並使用 SVV_DATASHARE_OBJECTS 檢視資料共用中的物件。

下列範例顯示取用者叢集中的輸入資料共用。

```
SHOW DATASHARES LIKE 'sales%';
```

```

share_name | share_owner | source_database | consumer_database | share_type
| createdate | is_publicaccessible | share_acl | producer_account |
producer_namespace
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
salesshare |          |          |          | INBOUND
|          |          |          |          | 123456789012 |
13b8833d-17c6-4f16-8fe4-1a018f5ed00d

```

4. 建立參照資料共用的本機資料庫。您必須指定 ACCOUNT 和 NAMESPACE 選項，才能為 AWS Data Exchange 資料共用建立本機資料庫。如需詳細資訊，請參閱 [CREATE DATABASE](#)。

```
CREATE DATABASE sales_db FROM DATASHARE salesshare OF ACCOUNT '123456789012'
NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

如果您想要更精細地控制對本機資料庫中物件的存取，請在建立資料庫時使用 WITHPERMISSIONS 子句。這可讓您在步驟 6 中為資料庫中的物件授予物件層級權限。

```
CREATE DATABASE sales_db WITH PERMISSIONS FROM DATASHARE salesshare OF ACCOUNT
'123456789012' NAMESPACE '13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

您可以透過查詢 [SVV_REDSHIFT_DATABASES](#) 檢視來查看您從資料共用中建立的資料庫。您可以直接連線至這些資料庫，也可以連線至取用者叢集上的本機資料庫，並執行跨資料庫查詢，以查詢來自資料共用資料庫的資料。您不能在現有資料共用中建立的資料庫物件頂端建立資料共用。不過，您可以將資料複製到取用者叢集上的個別資料表中，執行所需的任何處理，然後共用已建立的新物件。

您也可以使用 Amazon Redshift 主控台從資料共用中建立資料共用。如需詳細資訊，請參閱 [從資料共用中建立資料庫](#)。

5. (選擇性) 建立外部結構描述，以參照取用者叢集上匯入之取用者資料庫中的特定結構描述，並對其指派精細的許可。如需詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#)。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA 'public';
```

- 視需要，將從資料共用建立的資料庫和結構描述參照的許可授予使用者叢集中的使用者或角色。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

如果您在沒有 WITH 的情況下建立資料庫 PERMISSIONS，您只能將從資料共用建立的整個資料庫的許可指派給您的使用者和角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。如果是這樣，您可以建立指向資料共用中特定結構描述的外部結構描述參考 (如上一個步驟所述)，並在結構描述層級上提供精細的許可。

您也可以共用在共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。也可以考慮讓生產者叢集以所需的精細程度為您建立其他資料共用。您可以依需要，為從資料共用中建立的資料庫建立盡可能多的結構描述參照。

如果您在步驟 4 WITH PERMISSIONS 中使用 建立資料庫，則必須為共用資料庫中的物件指派物件層級許可。僅具有 USAGE 許可的使用者在獲得其他物件層級許可 WITH PERMISSIONS 之前，無法存取使用 建立的資料庫中的任何物件。

```
GRANT SELECT ON sales_db.public.tickit_sales_redshift to Bob;
```

- 查詢資料共用中共用物件的資料。

在取用者資料庫和取用者叢集結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集中的本機物件。若要這麼做，他們會使用 JDBC 或 ODBC 驅動程式、SHOW 命令或 SVV_ALL 和 SVV_REDSHIFT 檢視。

生產者叢集在每個結構描述中可能會有多個資料庫、資料表和檢視的結構描述。取用者端的使用者只能看到透過資料共用提供的物件子集。這些使用者無法從生產者叢集看到整個中繼資料。此方法有助於透過資料共用提供精細的中繼資料安全控制

您可以繼續連線到本機叢集資料庫。但是現在，您也可以使用三部分 database.schema.table 表示法，從資料共用中建立的資料庫和結構描述中讀取。您可以執行跨越任何和所有可見資料庫的查

詢。這些資料庫可以是叢集上的本機資料庫，也可以是從資料共用建立的資料庫。或者，您可以直接連線至這些取用者資料庫，並針對具有部分記號的共用物件執行查詢。

您可以使用完整資格存取資料。如需詳細資訊，請參閱[跨資料庫查詢範例](#)。

```
SELECT * FROM sales_db.public.tickit_sales_redshift ORDER BY 1,2 LIMIT 5;
```

salesid	listid	sellerid	buyerid	eventid	dateid	qtysold	pricepaid	commission	saletime
1	1	36861	21191	7872	1875	4	728.00	109.20	2008-02-18 02:36:48
2	4	8117	11498	4337	1983	2	76.00	11.40	2008-06-06 05:00:16
3	5	1616	17433	8647	1983	2	350.00	52.50	2008-06-06 08:26:17
4	5	1616	19715	8647	1986	1	175.00	26.25	2008-06-09 08:38:52
5	6	47402	14115	8240	2069	2	154.00	23.10	2008-08-31 09:17:02

您只能在共用物件上使用SELECT陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

除了查詢之外，取用者還可以在共用物件上建立檢視。僅支援近期繫結檢視或具體化視觀表。Amazon Redshift 不支援共用資料的一般檢視。取用者建立的檢視可跨越多個本機資料庫或從資料共用建立的資料庫。如需詳細資訊，請參閱[CREATE VIEW](#)。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.
CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

使用 AWS Lake Formation 受管資料共用

使用 Amazon Redshift，您可以存取和分析透過 AWS Lake Formation 資料共用共享的資料。AWS Lake Formation 資料共用可跨 AWS 帳戶和 Amazon Redshift 叢集進行安全的資料共用，而不必複製或移動基礎資料。

共用資料 AWS Lake Formation 可讓您集中定義 Amazon Redshift 資料共用的 AWS Lake Formation 許可，並限制使用者存取資料共用內的物件。

主題

- [以生產者身分使用 Lake Formation 管理的資料共用](#)
- [以取用者身分使用 Lake Formation 管理的資料共用](#)

以生產者身分使用 Lake Formation 管理的資料共用

透過 Amazon Redshift，您可以使用 AWS Lake Formation 受管資料共用作為生產者，在 AWS 帳戶和 Amazon Redshift 叢集之間安全地共用即時資料。Lake Formation 管理的資料共用是可讓您將來自 Amazon Redshift 叢集的即時資料與其他 AWS 帳戶和服務共用的物件。

身為生產者叢集或工作群組管理員，請依照下列步驟將資料共用至 Lake Formation：

1. 在叢集中建立資料共用，並授權 AWS Lake Formation 存取資料共用。

只有叢集超級使用者和資料庫擁有者可以建立資料共用。每個資料共用都會在建立期間與資料庫相關聯。只有來自該資料庫的物件才能在該資料共用中共用。可以在具有相同或不同物件細微程度的相同資料庫上建立多個資料共用。可以在叢集中建立的資料共用數量不限。

```
CREATE DATASHARE salesshare;
```

2. 將物件新增到資料共用。生產者叢集或工作群組管理員會繼續管理可用的資料共用物件。若要將物件新增至資料共用，請在新增物件之前先新增結構描述。當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。您必須明確地新增這些物件。如需詳細資訊，請參閱 [ALTER DATASHARE](#)。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;  
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;  
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

您也可將檢視新增到資料共用。支援的檢視為標準檢視、近期繫結檢視和具體化視觀表。

```
CREATE VIEW public.sales_data_summary_view AS SELECT * FROM
public.tickit_sales_redshift;
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
```

使用在指定的結構描述中ALTERDATASHARE共用結構描述、資料表和檢視。超級使用者、資料共用擁有者或擁有資料共用ALTER或ALL許可的使用者，可以變更資料共用，以新增物件或從中移除物件。資料庫使用者應該是物件的擁有者SELECT，或對物件具有USAGE、或ALL許可。

使用INCLUDENEW子句將指定結構描述中建立的任何新資料表和檢視新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。

```
ALTER DATASHARE salesshare ADD SCHEMA PUBLIC;
ALTER DATASHARE salesshare SET INCLUDENEW = TRUE FOR SCHEMA PUBLIC;
```

3. 將資料共用的存取權授予 Lake Formation 管理員帳戶。

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '012345678910' VIA DATA CATALOG;
```

若要撤銷使用權，請使用下列命令。

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '012345678910' VIA DATA CATALOG;
```

4. 使用 `aws redshift authorize-data-share` API 操作授權存取 Lake Formation 的資料共用。這麼做可允許 Lake Formation 識別服務帳戶中的資料共用，並管理將取用者與資料共用的關聯。

```
aws redshift authorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
--consumer-identifier {"DataCatalog/<consumer-account-id>"}
```

若要從 Lake Formation 管理的資料共用中移除授權，請使用 `aws redshift deauthorize-data-share` API 操作。如此一來，您 AWS Lake Formation 便允許 辨識服務帳戶中的資料共用，並移除授權。

```
aws redshift deauthorize-data-share
--data-share-arn arn:aws:redshift:us-east-1:{PRODUCER_ACCOUNT}:datashare:
{PRODUCER_CLUSTER_NAMESPACE}/salesshare
```

```
--consumer-identifier {"DataCatalog/<consumer-account-id>"}
```

在任何時候，如果生產者叢集或工作群組管理員決定不再需要與取用者叢集或工作群組共用資料，他們可以使用 `DROP DATASHARE` 刪除資料共用、取消授權資料共用或撤銷資料共用許可。Lake Formation 中的相關許可和物件不會自動刪除。

```
DROP DATASHARE salesshare;
```

授權 Lake Formation 帳戶管理資料共用後，Lake Formation 管理員可以探索共用的資料共用、將日期共用與 Data Catalog 建立關聯 ARN，以及在資料共用 AWS Glue Data Catalog 連結中建立資料庫。若要使用建立資料共用的關聯 AWS CLI，請使用命令 [associate-data-share-consumer](#)。若要跨共用資料共用 AWS 區域，請在 `associate-data-share-consumer` 命令中指定 `--region` 參數，或使用 AWS 主控台選擇您的資料取用者。下列範例示範如何跨區域共用 Lake Formation 管理的資料共用。

```
aws redshift associate-data-share-consumer --region <region-1>
--data-share-arn 'arn:aws:redshift:us-
east-1:12345678912:datashare:035c45ea-61ce-86f0-8b75-19ac6102c3b7/sample_share'
--consumer-arn 'arn:aws:glue:<region-1>:111912345678:catalog'
```

Lake Formation 管理員還必須建立本機資源，以定義資料共用中的物件應如何對應到 Lake Formation 中的物件。如需探索資料共用和建立本機資源的相關資訊，請參閱 [管理 Amazon Redshift 資料共用中的資料許可](#)。

以取用者身分使用 Lake Formation 管理的資料共用

使用 Amazon Redshift，您可以存取和分析透過資料共用與您共用 AWS Lake Formation 的資料。資料共用是一種資料產品，其中包含來自不同資料來源的一系列資料物件，例如資料表或資料庫。

AWS Lake Formation 管理員探索資料共用邀請並在連結至資料共用 AWS Glue Data Catalog 的中建立資料庫後，取用者叢集或工作群組管理員可以將叢集與中的資料共用和資料庫建立關聯 AWS Glue Data Catalog，建立取用者叢集或工作群組的本機資料庫，並授予 Amazon Redshift 取用者叢集或工作群組中使用者和角色的存取權以開始查詢。請遵循以下步驟來設定查詢許可。

1. 如有需要，請在 Amazon Redshift 主控台建立 Redshift 叢集做為取用者叢集或工作群組使用。如需如何建立叢集的資訊，請參閱 [建立叢集](#)。
2. 若要列出 AWS Glue Data Catalog 取用者叢集或工作群組使用者可存取的資料庫，請執行 [SHOWDATABASES](#) 命令。

```
SHOW DATABASES FROM DATA CATALOG [ACCOUNT <account-id>,<account-id2>] [LIKE <expression>]
```

這樣做會列出資料目錄中可用的資源，例如 AWS Glue 資料庫的 ARN、資料庫名稱，以及有關資料共用的資訊。

3. 使用 ARN 來自 SHOW 的 AWS Glue 資料庫 DATABASES，在取用者叢集或工作群組中建立本機資料庫。如需詳細資訊，請參閱 [CREATE DATABASE](#)。

```
CREATE DATABASE lf_db FROM ARN <lake-formation-database-ARN> WITH [NO] DATA CATALOG SCHEMA [<schema>];
```

4. 視需要，將從資料共用建立的資料庫和結構描述參照的存取權授予取用者叢集或工作群組中的使用者和角色。如需詳細資訊，請參閱 [GRANT](#) 或 [REVOKE](#)。請注意，從 [CREATE USER](#) 命令建立的使用者無法存取已與 Lake Formation 共用的資料共用中的物件。只有同時擁有 Redshift 和 Lake Formation 存取權的使用者才能存取已與 Lake Formation 共用的資料共用。

```
GRANT USAGE ON DATABASE sales_db TO IAM:Bob;
```

身為取用者叢集或工作群組管理員，您只能將從資料共用建立的整個資料庫的許可指派給使用者和角色。在某些情況下，您需要更精細地控制從資料共用中建立的資料庫物件子集。

您也可以為共用物件之上建立近期繫結檢視，並使用這些檢視來指派精細的許可。您也可以考慮讓生產者叢集或工作群組以所需的精細程度為您建立其他資料共用。您可以為從資料共用中建立的資料庫建立盡可能多的結構描述參照。

5. 資料庫使用者可以使用檢視 SVV_EXTERNAL_TABLES 和 SVV_EXTERNAL_COLUMNS 尋找 AWS Glue 資料庫中的所有共用資料表或資料欄

```
SELECT * from svv_external_tables WHERE redshift_database_name = 'lf_db';  
  
SELECT * from svv_external_columns WHERE redshift_database_name = 'lf_db';
```

6. 查詢資料共用中共用物件的資料。

在取用者資料庫和取用者叢集或工作群組結構描述上具有許可的使用者和角色，可以探索和瀏覽任何共用物件的中繼資料。他們也可以探索和瀏覽取用者叢集或工作群組中的本機物件。若要這麼做，他們可以使用 JDBC 或 ODBC 驅動程式，或是 SVV_ALL 和 SVV_EXTERNAL 檢視。


```
SELECT * FROM lf_db.schema.table;
```

您只能在共用物件上使用SELECT陳述式。不過，您可以從不同本機資料庫中的共用物件查詢資料，在取用者叢集中建立資料表。

```
// Connect to a local cluster database

// Create a view on shared objects and access it.

CREATE VIEW sales_data
AS SELECT *
FROM sales_db.public.tickit_sales_redshift
WITH NO SCHEMA BINDING;

SELECT * FROM sales_data;
```

使用主控台管理資料共用

使用 Amazon Redshift，您可以使用主控台管理資料共用與寫入，以控制 Amazon Redshift 叢集和 AWS 帳戶之間的存取和管理資料。下列各節提供使用 Amazon Redshift 主控台設定和管理資料共用與寫入 step-by-step 的指示。

使用 Amazon Redshift 主控台管理在您帳戶中建立或從其他帳戶共用的資料共用。

您需要建立、編輯或刪除資料共用的許可。如需相關資訊，請參閱 [在 Amazon Redshift 中管理資料共用的許可](#)。

- 如果您是生產者叢集管理員，您可以建立資料共用、新增資料取用者、新增資料共用物件、從資料共用建立資料庫、編輯資料共用或從CLUSTERS索引標籤刪除資料共用。

從瀏覽功能表中，瀏覽叢集索引標籤，從叢集清單中選擇叢集。然後執行下列其中一項：

- 選擇資料共用索引標籤，從在我的命名空間中建立的資料庫中選擇一個資料共用。然後執行下列其中一項：
 - [建立資料共用](#)

建立資料共用時，您可以新增資料共用物件或資料取用者。如需詳細資訊，請參閱 [將資料共用物件新增到資料共用](#) 和 [將資料取用者新增至資料共用](#)。

- [編輯在您帳戶中建立的資料共用](#)
- [刪除在您帳戶中建立的 資料共用](#)
- 選擇資料共用，然後從來自其他叢集的資料共用區段中選擇資料共用。然後執行下列其中一項：
 - [建立資料共用](#)
 - [從資料共用中建立資料庫](#)
- 選擇資料庫，然後從資料庫區段中選擇資料庫。然後選擇建立資料共用。如需詳細資訊，請參閱[從資料共用中建立資料庫](#)。

Note

若要檢視資料庫和資料庫中的物件，或檢視叢集中的資料共用，請連線至資料庫。如需詳細資訊，請參閱[連線至資料庫](#)。

連線至資料庫

使用 Amazon Redshift，您可以建立與資料倉儲叢集的連線，並執行SQL查詢、載入資料或執行管理任務。連線至資料庫是指在用戶端應用程式或工具與 Amazon Redshift 叢集之間建立安全頻道的程序。下列各節將提供 step-by-step如何連線至 Amazon Redshift 資料庫的指示。

連線至資料庫以檢視此叢集中的資料庫和資料庫中的物件，或檢視資料共用。

用來連線至指定資料庫的使用者憑證必須具備檢視所有資料共用的必要許可。

如果沒有本機連線，請執行下列其中一項動作：

- 在叢集詳細資訊頁面中，從資料庫索引標籤的資料庫或資料共用物件區段中，選擇連線到資料庫以檢視叢集中的資料庫物件。
- 在叢集詳細資訊頁面中，從資料共用索引標籤中執行下列其中一項操作：
 - 在來自其他叢集的資料共用區段中，選擇連線到資料庫以檢視其他叢集的資料共用。
 - 在我的叢集中建立的資料共用區段中，選擇連線到資料庫以檢視您叢集中的資料共用。
- 在連線到資料庫視窗中，執行下列其中一項動作：
 - 如果您選擇建立新連線，請選擇 AWS Secrets Manager 來使用預存密碼來驗證連線的存取權。

或者，選擇暫時憑證以使用資料庫憑證來驗證連線的存取權。指定資料庫名稱和資料庫使用者的值。

選擇連線。

- 選擇使用最近使用的連線來連線到您擁有必要許可的其他資料庫。

Amazon Redshift 會自動進行連線。

建立資料庫連線之後，您可以開始建立資料共用、查詢資料共用或從資料共用建立資料庫。

建立資料共用

使用 Amazon Redshift，您可以使用資料共用在 Amazon Redshift 叢集或 AWS 帳戶之間共用即時資料。資料共用是消費者生產者物件，可讓您將來自 Amazon Redshift 叢集的即時資料與其他叢集或 AWS 帳戶共用。建立資料共用可啟用安全的資料共用，同時維持對存取的控制，並確保資料保持 up-to-date。下列各節提供建立資料共用和新增資料庫物件的詳細資訊，例如結構描述、資料表和檢視，以安全地共用即時資料。

建立資料共用

身為生產者叢集管理員，您可以從叢集詳細資訊頁面的資料庫或資料共用索引標籤中建立資料共用。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 在叢集詳細資訊頁面上，執行下列作業：
 - 從資料庫索引標籤的資料庫區段中，選擇資料庫。資料庫詳細資訊頁面會隨即出現。

選擇建立資料共用。您只能從本機資料庫建立資料共用。如果您尚未連線到資料庫，則會出現連線到資料庫頁面。依照 [連線至資料庫](#) 中的步驟連線到資料庫。如果有最近使用的連線，就會顯示建立資料共用頁面。

- 如果您沒有資料庫連線，請從資料共用索引標籤的資料共用區段中連線到資料庫。

在我的叢集中建立的資料共用區段中，選擇建立資料共用。建立資料共用頁面會隨即出現。

4. 在資料共用資訊區段中，選擇下列其中一項：
 - 選擇資料共用來建立資料共用，以便在不同的 Amazon Redshift 叢集或相同 AWS 帳戶 或不同 AWS 帳戶之間共用資料以供讀取。
 - 選擇 AWS Data Exchange 資料共用以建立資料共用，透過 授權您的資料 AWS Data Exchange。
5. 指定資料共用名稱、資料庫名稱和可公開存取的值。

當您變更資料庫名稱時，請建立新的資料庫連線。

- 在資料共用物件區段中，選擇新增。新增資料共用頁面會隨即出現。若要將物件新增至資料共用，請遵循 [將資料共用物件新增到資料共用](#)。
- 在資料取用者區段中，您可以選擇發佈至 Redshift 帳戶，或發佈至 AWS Glue Data Catalog，這會開始透過 Lake Formation 共用資料的程序。將您的資料共用發佈到 Redshift 帳戶意味著與另一個作為取用者叢集的 Redshift 帳戶共用您的資料。

Note

建立資料共用之後，您就無法編輯要發佈到其他選項的組態。

- 選擇建立資料共用。

Amazon Redshift 會建立資料共用。建立資料共用之後，您就可以從資料共用中建立資料庫。

將資料共用物件新增到資料共用

將一個或多個物件新增到資料共用。對於資料取用者而言，資料共用物件是唯讀的。

您可以在不新增資料共用物件的情況下建立資料共用，稍後再新增物件。

只有當您將至少一個物件新增至資料共用時，資料共用才會變為作用中狀態。

- 從資料共用清單中選擇要新增物件的資料共用。
- 選擇新增。新增資料共用物件頁面會隨即出現。
- 在新增其他資料共用物件之前，請至少向資料共用新增一個結構描述。選擇新增並重複來新增多個結構描述。
- 您可以選擇從指定結構描述新增所選物件類型的所有現有物件，或從指定的結構描述新增特定個別物件。選擇物件類型，例如資料表和檢視或使用者定義的函數。
- 您可以選擇新增並重複來新增指定的結構描述和資料共用物件，並繼續新增另一個結構描述和物件。

將資料取用者新增至資料共用

您可以將一個或多個資料取用者新增至資料共用。資料取用者可以是唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

您必須明確選擇關閉或開啟與具有公用存取權的叢集共用您的資料共用。

- 選擇將叢集命名空間新增至資料共用。命名空間是 Amazon Redshift 叢集的全域唯一識別碼 (GUID)。
- 選擇新增 AWS 帳戶 至資料共用。指定的 AWS 帳戶 必須具有資料共用的存取權限。

將資料湖資料表新增至資料共用

透過資料共用，資料生產者可以安全地與相同 AWS 帳戶或不同帳戶中的取用者共用精細的資料庫物件，例如結構描述和資料表。生產者也可以跨區域共用物件。本主題說明如何將物件從資料湖，特別是從 AWS Glue 資料目錄新增至資料共用。它涵蓋兩個使用案例：

- 將繫結較晚的檢視新增至從資料湖參考資料表的資料共用 – 這對取用者來說很方便，因為初步組態，例如在外部來源資料上定義許可，例如使用 Lake Formation，可能已經完成。另一個好處是新增至資料共用的檢視，可以從資料湖使用 Redshift 原生資料表聯結資料表。
- 將資料表從外部結構描述直接新增至資料共用 – 這可讓沒有額外圖層或邏輯的取用者使用資料湖中的物件。取用者可以查詢資料表，或與取用者上的資料表聯結。

當您使用從 Redshift [CREATE EXTERNAL SCHEMA](#) 中的 AWS 資料目錄中參考資料表後，這些案例就會適用。AWS 資料目錄中的任何資料表都可以是來源。

Note

您新增至資料共用的資料湖資料表可包含向 Lake Formation 註冊的資料表和 AWS Glue 資料目錄資料表。

建立外部結構描述和外部資料表

您可以建立外部結構描述和外部資料表，以便將它們新增至以下各節中的資料共用。這些是初步步驟。如果您已完成此操作，可以略過本節。

1. 在生產者上，建立參考 Amazon S3 中存放之資料湖資料的外部結構描述。外部結構描述參考 AWS Glue Data Catalog。範例中的角色和區域為範例：

```
CREATE EXTERNAL SCHEMA external_schema_name FROM DATA CATALOG
DATABASE 'glue_database_name'
IAM_ROLE 'arn:aws:iam::123456789012:role/sample-role'
REGION 'us-east-1';
```

2. 在外部結構描述中建立資料湖資料表。

```
CREATE EXTERNAL TABLE external_schema_name.sales(  
  salesid INTEGER,  
  sellerid INTEGER,  
  buyerid INTEGER,  
  saledate DATE,  
  pricepaid DECIMAL(8,2))  
ROW FORMAT delimited  
FIELDS TERMINATED BY '\t'  
STORED AS textfile  
LOCATION 's3://redshift-downloads/tickit/spectrum/sales/';
```

範例包含 LOCATION。它必須是指定 s3://{bucket_name}/{folder}/資料夾的格式。資料夾的長度必須至少為一個字元。您可以選擇性地包含子資料夾。若要查看在資料湖中建立資料表的其他範例，請參閱 CREATE EXTERNAL [的範例](#) TABLE。

Note

只有生產者上 IAM 的角色可在資料表上 SELECT 存取的資料表才支援共用。

將參考資料湖資料表的繫結後期檢視新增至資料共用

當您根據 AWS 資料目錄中的外部結構描述建立資料表，並想要將其新增至資料共用時，最常見的方法是新增參考您建立之資料表的 Redshift 後期繫結檢視，其中包含來自資料湖的資料。下列程序顯示步驟：

1. 建立繫結後期檢視，參考您先前建立的外部資料表：

```
CREATE VIEW lbv AS  
select * from external_schema_name.sales, other_schema.t1  
WITH NO SCHEMA BINDING;
```

2. 將檢視結構描述新增至資料共用。這是包含繫結後期檢視的本機結構描述。

```
ALTER DATASHARE dsx_datashare ADD SCHEMA public;
```

3. 將包含後期繫結檢視所參考之資料表的結構描述新增至資料共用。新增至資料共用的檢視中參考的任何基礎資料表都需要新增結構描述，無論結構描述是否包含本機資料庫物件或來自資料湖的物件。請注意，您必須先新增此結構描述，才能新增繫結後期檢視。

```
ALTER DATASHARE dsx_datashare ADD SCHEMA external_schema_name;  
ALTER DATASHARE dsx_datashare ADD SCHEMA other_schema;
```

4. 使用 SQL 命令將檢視新增至資料共用。請注意，資料表名稱包含結構描述字首。

```
ALTER DATASHARE my_datashare ADD TABLE public.lbv;
```

5. 確認檢視和結構描述已成功新增至資料共用：

```
SELECT * FROM svv_datashare_objects WHERE share_name = 'my_datashare';
```

6. 取用者管理員從資料共用建立資料庫，然後將用量授予取用者使用者。

完成這些步驟後，具有資料共用檢視存取權的資料庫取用者使用者可以查詢資料。

將資料湖資料表直接新增至資料共用

將外部結構描述中的資料表新增至資料共用類似於新增檢視。這適用於消費者想要以原始狀態查詢資料湖資料表，或消費者想要將其加入消費者資料倉儲資料表的情況。下列步驟說明如何使用將資料湖資料表新增至資料共用SQL。

1. 建立外部結構描述和外部資料表，如本主題的第一節所述。
2. 探索外部結構描述中的現有資料表，以確認您建立的資料表可用：

```
SELECT * FROM svv_external_tables WHERE schemaname = 'external_schema_name';
```

3. 將外部結構描述新增至資料共用：

```
ALTER DATASHARE my_datashare ADD SCHEMA external_schema_name;
```

4. 將外部資料表新增至資料共用。請注意，資料表名稱包含結構描述字首：

```
ALTER DATASHARE my_datashare ADD TABLE external_schema_name.sales;
```

5. 確認資料表已成功新增至資料共用：

```
SELECT * FROM svv_datashare_objects WHERE share_name = 'my_datashare';
```

如需更詳細的指示，請參閱[共用 AWS 帳戶中資料的讀取存取權](#)。

- 在取用者上，也就是接收共用資料的資料庫，管理員會建立資料共用的關聯，讓使用者可以查詢共用資料表。如需如何執行此步驟的詳細資訊，請參閱[以取用者身分管理來自其他帳戶的資料共用](#)。

管理員完成這些步驟後，取用者上的資料庫使用者可以寫入查詢，從共用資料表擷取資料，並將其與取用者上的其他資料表聯結。

將資料湖物件新增至資料共用的用量備註

當您在資料共用中使用資料表和資料湖的檢視時，需要注意幾個項目：

- 使用 記錄 AWS CloudTrail – 資料生產者帳戶可以在存取透過資料共用共用的資料湖資料表時，使用 AWS CloudTrail 日誌進行稽核：
 - 使用日誌資料來控制資料存取 – CloudTrail 日誌會記錄有關誰存取共用資料表的詳細資訊，包括 Redshift 資料共用生產者和取用者。識別符可在 AssumeRole CloudTrail 日誌下的 ExternalId 欄位中使用。資料擁有者可以透過 動作設定 IAM 政策中資料存取的其他限制。如需透過政策定義資料存取的詳細資訊，請參閱[存取第三方擁有 AWS 的帳戶](#)。
- 安全與取用者許可 – 對於 Lake Formation 註冊資料表，Amazon S3 資源由 Lake Formation 保護，並使用 Lake Formation 提供的憑證提供。

將資料湖物件新增至資料共用的帳單考量

下列詳細說明如何在資料共用中儲存和掃描資料湖物件的成本屬性：

- 當取用者從資料湖查詢共用物件時，掃描成本會向取用者收取。
 - 當取用者是佈建叢集時，Redshift 會使用 Redshift Spectrum 掃描 Amazon S3 資料。因此，Spectrum 成本由消費者帳戶支付。
 - 當取用者是 Amazon Redshift Serverless 工作群組時，Spectrum 無需另外付費。
- Amazon S3 的儲存和操作成本，例如清單儲存貯體，會計入擁有每個 Amazon S3 儲存貯體的帳戶。

如需 Amazon Redshift Serverless 帳單的其他詳細資訊，請參閱 [Amazon Redshift Serverless 帳單](#)。[Amazon Redshift 定價](#) 提供更多帳單和定價資訊。

授予或移除資料共用的授權

身為生產者叢集管理員，請選擇要授權存取資料共用或從中移除授權的資料取用者。獲授權的資料取用者會收到對資料共用採取動作的通知。如果您要將叢集命名空間新增為資料取用者，則不必執行授權。

先決條件：若要授予或移除資料共用的授權，必須至少已將一個資料取用者新增至資料共用。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇在我的帳戶中。
4. 在我帳戶中的資料共用區段中，執行下列其中一項動作：
 - 選擇您要授權的一個或多個取用者叢集。授權資料取用者頁面會隨即出現。然後選擇授權。

如果您在建立資料共用時選擇發佈至 AWS Glue Data Catalog，則只能將資料共用的授權授予 Lake Formation 帳戶。

對於 AWS Data Exchange 資料共用，您一次只能授權一個資料共用。

當您授權 AWS Data Exchange 資料共用時，您將與 AWS Data Exchange 服務共用資料，並允許代表您 AWS Data Exchange 管理對資料共用的存取權。當取用者訂閱產品時，會將取用者帳戶新增為 AWS Data Exchange 資料共用的資料取用者，AWS Data Exchange 以允許取用者存取資料共用。AWS Data Exchange 沒有資料共用的讀取存取權。

- 選擇您要從中移除授權的一個或多個取用者叢集。然後選擇移除授權。

授權資料取用者之後，他們就可以存取資料共用物件，並建立取用者資料庫來查詢資料。

移除授權後，資料取用者會立即失去對資料共用的存取權限。

以取用者身分管理來自其他帳戶的資料共用

使用 Amazon Redshift，您可以從其他 AWS 帳戶使用資料共用，啟用跨帳戶資料共用和協作。資料共用是跨 Amazon Redshift 叢集共用即時資料的安全方法，即使它們位於不同的 AWS 帳戶中也是如此。下列各節提供設定存取、查詢共用資料和以取用者身分監控資料共用活動的詳細步驟。

建立資料共用的關聯

作為取用者叢集管理員，您可以將從其他帳戶共用的一或多個資料共用與整個 AWS 帳戶或帳戶中的特定叢集命名空間建立關聯。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。

3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇您要建立關聯的資料共用，然後選擇建立關聯。當顯示 [與資料共用建立關聯] 頁面時，選擇下列其中一個關聯類型：

- 選擇整個 AWS 帳戶，將 AWS 帳戶中不同 AWS 區域的所有現有和未來叢集命名空間與資料共用建立關聯。然後選擇建立關聯。

如果資料共用發佈至 AWS Glue Data Catalog，您只能將資料共用與整個 AWS 帳戶建立關聯。

- 選擇特定 AWS 區域和叢集命名空間，將一或多個 AWS 區域和特定叢集命名空間與資料共用建立關聯。
 - a. 選擇新增區域，將特定 AWS 區域和叢集命名空間新增至資料共用。隨即顯示新增 AWS 區域頁面。
 - b. 選擇 AWS 區域。
 - c. 執行以下任意一項：
 - 選擇新增所有叢集命名空間，將此區域中所有現有和未來的叢集命名空間新增至資料共用。
 - 選擇新增特定叢集命名空間，將此區域中的一或多個特定叢集命名空間新增至資料共用。
 - 選擇一或多個叢集命名空間，然後選擇新增 AWS 區域。
 - d. 選擇建立關聯。

如果您要將資料共用與 Lake Formation 帳戶建立關聯，請前往 Lake Formation 主控台建立資料庫，然後定義資料庫的許可。如需詳細資訊，請參閱 AWS Lake Formation 開發人員指南中的[設定 Amazon Redshift 資料共用的許可](#)。建立 AWS Glue 資料庫或聯合資料庫後，您可以將查詢編輯器 v2 或任何偏好的 SQL 用戶端與取用者叢集搭配使用來查詢資料。如需詳細資訊，請參閱[以取用者身分使用 Lake Formation 管理的資料共用](#)。

與資料建立關聯之後，資料共用就會變為可用。

您也可以隨時變更資料共用關聯。將特定 AWS 區域和叢集命名空間變更為整個 AWS 帳戶時，Amazon Redshift 會使用 AWS 帳戶資訊覆寫特定區域和叢集命名空間資訊。AWS 帳戶中的所有 AWS 區域和叢集命名空間，即可存取資料共用。

當將關聯從特定叢集命名空間變更為指定 AWS 區域中的所有叢集命名空間時，此區域中的所有叢集命名空間都可以存取資料共用。

從資料取用者移除資料共用的關聯

身為取用者叢集管理員，您可以從資料取用者移除資料共用的關聯。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇要從資料取用者中移除關聯的資料共用。
5. 在資料取用者段落中，選擇一或多個要從中移除關聯的資料取用者。然後選擇移除關聯。
6. 當移除關聯頁面出現時，請選擇移除關聯。

移除關聯之後，資料取用者將失去對資料共用的存取權。您可以隨時變更資料取用者關聯。

拒絕資料共用

身為取用者叢集管理員，您可以拒絕任何狀態為[可用或作用中](#)的資料共用。拒絕資料共用後，取用者叢集使用者將失去資料共用的存取權。如果您呼叫DescribeDataSharesForConsumerAPI操作，Amazon Redshift 不會傳回被拒絕的資料共用。如果生產者叢集管理員執行DescribeDataSharesForProducerAPI操作，他們就會看到資料共用遭到拒絕。一旦資料共用被拒絕，生產者叢集管理員可以再次將資料共用授權給取用者叢集，取用者叢集管理員可以選擇將其AWS 帳戶與資料共用建立關聯或拒絕。

如果 AWS 您的帳戶與資料共用有關聯，且與 Lake Formation 管理的資料共用有待定關聯，拒絕 Lake Formation 管理的資料共用關聯也會拒絕原始資料共用。若要拒絕特定關聯，生產者叢集管理員可以從指定的資料共用中移除授權。此動作不會影響其他資料共用。

若要拒絕資料共用，請使用 AWS 主控台、API操作 RejectDataShare或 reject-datashare中的 AWS CLI。

若要使用 AWS 主控台拒絕資料共用：

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇您要拒絕的資料共用。出現拒絕資料共用頁面時，選擇拒絕。

拒絕資料共用之後，您就無法還原變更。Amazon Redshift 會從清單中移除資料共用。若要再次查看資料共用，生產者管理員必須再次對其授權。

管理現有資料共用

使用 Amazon Redshift，您可以管理現有的資料共用，以控制對 Amazon Redshift 叢集中資料的存取。下列各節提供在 Amazon Redshift 環境中管理資料共用 step-by-step 的指引。

檢視資料共用

從 DATASHARES 或 CLUSTERS 索引標籤檢視資料共用。

- 使用 DATASHARES 索引標籤列出您帳戶中或其他帳戶中的資料共用。
 - 若要檢視您帳戶中建立的資料共用，請選擇在我的帳戶中，然後選擇您要檢視的資料共用。
 - 若要檢視從其他帳戶共用的資料庫，請選擇來自其他帳戶，然後選擇您要檢視的資料共用。
- 使用 CLUSTERS 索引標籤列出叢集或其他叢集中的資料共用。

連線到資料庫 如需詳細資訊，請參閱[連線至資料庫](#)。

然後從來自其他叢集的資料共用或在我叢集中建立的資料共用區段中選擇一個資料共用，以檢視其詳細資訊。

從資料共用中移除資料共用物件

您可以使用下列程序，從資料共用中移除一個或多個物件。

從資料共用中移除一個或多個物件

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱[連線至資料庫](#)。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 若要對資料共用移除一或多個資料共用物件，請執行下列其中一項操作：
 - 若要從資料共用中移除結構描述，請選擇一或多個結構描述。然後選擇移除。Amazon Redshift 會從資料共用中移除指定的結構描述和指定結構描述的所有物件。
 - 若要從資料共用中移除資料表和檢視，請選擇一或多個資料表和檢視。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有資料表和檢視。

- 若要從資料共用中移除使用者定義函數，請選擇一或多個使用者定義函數。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有使用者定義函數。

從資料共用中移除資料取用者

您可以從資料共用中移除一或多個資料取用者。資料取用者可以是可唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

從叢集命名空間或 IDs AWS 帳戶中選擇一或多個資料取用者，然後選擇移除。

Amazon Redshift 會從資料共用中移除指定的資料取用者。您會立即失去資料共用的存取權。

編輯在您帳戶中建立的資料共用

使用主控台編輯在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱[連線至資料庫](#)。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 在資料共用物件或資料取用者區段中進行任何變更。

Note

如果您選擇將資料共用發佈至 AWS Glue Data Catalog，則無法編輯組態，將資料共用發佈至其他 Amazon Redshift 帳戶。

7. 選擇 Save changes (儲存變更)。

Amazon Redshift 會以這些變更來更新您的資料共用。

刪除在您帳戶中建立的資料共用

使用主控台刪除在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱[連線至資料庫](#)。
5. 選擇一或多個您要刪除的資料共用，然後選擇刪除。刪除資料共用頁面會隨即出現。

刪除與 Lake Formation 共用的資料共用不會自動移除 Lake Formation 中的相關許可。若要將其刪除，請前往 Lake Formation 控制台。

6. 輸入刪除以確認刪除指定的資料共用。
7. 選擇刪除。

刪除資料共用之後，資料共用取用者會失去資料共用的存取權。

查詢資料庫

使用 Amazon Redshift，您可以從生產者叢集查詢資料共用之間的資料，以安全地存取即時資料，而無需複製或傳輸。下列各節涵蓋在您的 Amazon Redshift 環境中設定和查詢資料共用的詳細資訊。

從資料共用中建立資料庫

若要開始查詢資料共用中的資料，請從資料共用中建立資料庫。僅可從指定資料共用中建立一個資料庫。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 在來自其他叢集的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱[連線至資料庫](#)。
5. 選擇您要在其中建立資料庫的資料共用，然後選擇從資料共用建立資料庫。從資料共用建立資料庫頁面會隨即出現。
6. 在資料庫名稱中，指定資料庫名稱。資料庫名稱必須是 1–64 個英數字元 (僅限小寫字母)，且不能是保留字。
7. 選擇 Create (建立)。

建立資料庫之後，您就可以查詢資料庫中的資料。

管理 AWS Data Exchange 資料共用

使用 Amazon Redshift，您可以安全地共用和接收來自的即時資料，AWS Data Exchange 而無需建立和管理資料擷取或管道。管理 AWS Data Exchange 資料共用可讓您訂閱第三方資料產品，並將即時資料串流直接整合到您的 Amazon Redshift 資料倉儲中。下列各節示範 AWS Data Exchange 管理 Amazon Redshift 叢集內的資料共用。

在 上建立資料集 AWS Data Exchange

在 上建立資料集 AWS Data Exchange。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 在我的帳戶建立的資料共用區段中，選擇 AWS Data Exchange 資料共用。
5. 選擇在 上建立資料集 AWS Data Exchange。如需詳細資訊，請參閱[發佈新產品](#)。

編輯 AWS Data Exchange 資料共用

使用主控台編輯 AWS Data Exchange 資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

對於 AWS Data Exchange 資料共用，您無法變更資料取用者。

若要編輯 AWS Data Exchange 資料共用的公開存取設定，請使用查詢編輯器 v2。Amazon Redshift 會產生隨機的一次性值來設定工作階段變數，以允許關閉此設定。如需詳細資訊，請參閱[ALTER DATASHARE 用量備註](#)。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 從導覽功能表中選擇編輯器，然後選擇查詢編輯器 v2。
4. 如果這是您第一次使用查詢編輯器 v2，請設定您的 AWS 帳戶。根據預設，AWS 擁有的金鑰會用來加密資源。如需設定的詳細資訊 AWS 帳戶，請參閱 Amazon Redshift 管理指南中的[設定 AWS 帳戶](#)。

- 若要連線至 AWS Data Exchange 資料共用所在的叢集，請在樹狀檢視面板中選擇資料庫和叢集名稱。若出現提示，請輸入連線參數。
- 複製下列SQL陳述式。下列範例會變更 salesshare 資料共用的可公開存取設定。

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
```

- 若要執行複製的SQL陳述式，請選擇查詢並將複製的SQL陳述式貼到查詢區域中。接著選擇執行。

下列錯誤會隨即出現：

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
ERROR: Alter of ADX-managed datashare salesshare requires session variable
datashare_break_glass_session_var to be set to value 'c670ba4db22f4b'
```

值 'c670ba4db22f4b' 是 Amazon Redshift 在非建議操作發生時隨機產生的一次性值。

- 將下列範例陳述式複製並貼上查詢區域。接著執行命令。SET datashare_break_glass_session_var 命令會套用許可，以允許 AWS Data Exchange 資料共用的不建議操作。

```
SET datashare_break_glass_session_var to 'c670ba4db22f4b';
```

- 再次執行ALTERDATASHARE陳述式。

```
ALTER DATASHARE salesshare;
```

Amazon Redshift 會以這些變更來更新您的資料共用。

刪除在您的帳戶中建立 AWS Data Exchange 的資料共用

使用主控台刪除在帳戶中建立 AWS Data Exchange 的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

- 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
- 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
- 從導覽功能表中選擇編輯器，然後選擇查詢編輯器 v2。

4. 如果這是您第一次使用查詢編輯器 v2，請設定您的 AWS 帳戶。根據預設，AWS 擁有的金鑰會用來加密資源。如需設定的詳細資訊 AWS 帳戶，請參閱 Amazon Redshift 管理指南中的[設定 AWS 帳戶](#)。
5. 若要連線至 AWS Data Exchange 資料共用所在的叢集，請在樹狀檢視面板中選擇資料庫和叢集名稱。若出現提示，請輸入連線參數。
6. 複製下列 SQL 陳述式。下列範例會捨棄 salesshare 資料共用。

```
DROP DATASHARE salesshare
```

7. 若要執行複製的 SQL 陳述式，請選擇查詢並將複製的 SQL 陳述式貼到查詢區域中。接著選擇執行。

下列錯誤會隨即出現：

```
ERROR: Drop of ADX-managed datashare salesshare requires session variable datashare_break_glass_session_var to be set to value '620c871f890c49'
```

值 '620c871f890c49' 是 Amazon Redshift 在非建議操作發生時隨機產生的一次性值。

8. 將下列範例陳述式複製並貼上查詢區域。接著執行命令。SET datashare_break_glass_session_var 命令會套用許可，以允許 AWS Data Exchange 資料共用的不建議操作。

```
SET datashare_break_glass_session_var to '620c871f890c49';
```

9. 再次執行 DROP DATASHARE 陳述式。

```
DROP DATASHARE salesshare;
```

刪除資料共用之後，資料共用取用者會失去資料共用的存取權。

刪除共用 AWS Data Exchange 資料共用可能會違反 中的資料產品條款 AWS Data Exchange。

使用 AWS CloudFormation 管理資料共用

您可以使用佈建 AWS 資源的 AWS CloudFormation 堆疊來自動化資料共用設定。CloudFormation 堆疊會設定相同 AWS 帳戶中兩個 Amazon Redshift 叢集之間的資料共用。因此，您可以開始資料共用，而無需執行 SQL 陳述式來佈建資源。

堆疊會在您指定的叢集上建立資料共用。資料共用包括資料表和範例唯讀資料。您的其他 Amazon Redshift 叢集可以讀取此資料。

如果您想要透過執行SQL陳述式來設定資料共用並授予許可，以開始在 AWS 帳戶中共用資料，CloudFormation請參閱 [共用 內資料的讀取存取權 AWS 帳戶](#)。

在執行資料共用 CloudFormation 堆疊之前，您必須使用具有建立IAM角色和 Lambda 函數許可的使用者登入。您還需要在同一個帳戶中使用兩個 Amazon Redshift 叢集。您可以使用其中一個叢集 (生產者)來共用範例資料，使用另一個叢集 (取用者) 來讀取範例資料。這些叢集的主要需求是每個叢集都使用RA3節點。如需其他需求，請參閱 [在 Amazon Redshift 中使用資料共用時的考量事項](#)。

如需開始設定 Amazon Redshift 叢集的詳細資訊，請參閱[開始使用 Amazon Redshift 佈建的資料倉儲](#)。如需使用 自動設定的詳細資訊 CloudFormation，請參閱[什麼是 AWS CloudFormation ?](#)

Important

啟動 CloudFormation 堆疊之前，請確定在同一帳戶中有兩個 Amazon Redshift 叢集，且叢集使用RA3節點。確保每個叢集都有一個資料庫和一個超級使用者。如需詳細資訊，請參閱 [CREATE DATABASE](#) 和 [superuser](#)。

若要啟動 Amazon Redshift 資料共用的 CloudFormation 堆疊：

1. 按一下[啟動CFN堆疊](#)，這將帶您前往 中的 CloudFormation 服務 AWS Management Console。

如果出現系統提示，請登入。

堆疊建立程序開始，參考儲存在 Amazon S3 中的 CloudFormation 範本檔案。CloudFormation 範本是 JSON 格式的文字檔案，可宣告 AWS 構成堆疊的資源。如需 CloudFormation 範本的詳細資訊，請參閱[了解範本基本知識](#)。

2. 選擇下一步以輸入堆疊詳細資料。
3. 在參數底下，為每個叢集輸入下列內容：
 - 您的 Amazon Redshift 叢集名稱，例如 **ra3-consumer-cluster**
 - 您的資料庫名稱，例如 **dev**
 - 資料庫使用者的名稱，例如 **consumeruser**

我們建議您使用測試叢集，因為堆疊會建立數個資料庫物件。

選擇下一步。

4. 堆疊選項隨即出現。

選擇下一步以接受預設設定。

5. 在功能下，選擇我確認 AWS CloudFormation 可能會建立IAM資源。

6. 選擇建立堆疊。

CloudFormation 使用 範本建置 Amazon Redshift 堆疊大約需要 10 分鐘，建立名為 `myproducer_share` 的資料共用。堆疊會在堆疊詳細資訊中指定的資料庫中建立資料共用。只有來自該資料庫的物件才能共用。

如果建立堆疊時發生錯誤，請執行下列動作：

- 請確定您為每個 Redshift 叢集輸入正確的叢集名稱、資料庫名稱和資料庫使用者名稱。
- 確保您的叢集具有RA3節點。
- 請確定您使用具有建立IAM角色和 Lambda 函數許可的使用者登入。如需建立IAM角色的詳細資訊，請參閱[建立IAM角色](#)。如需 Λ 函數建立政策的相關資訊，請參閱[函數開發](#)。

查詢您建立的資料共用

若要使用下列程序，請確定您在每個所述叢集上有執行查詢所需的許可。

查詢您的資料共用：

1. 使用 Amazon Redshift 查詢編輯器 v2 等用戶端工具，連線至建立堆疊時 CloudFormation輸入的資料庫上的生產者叢集。
2. 查詢資料庫。

```
SHOW DATASHARES;
```

```
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
|  share_name    | share_owner | source_database | consumer_database | share_type
| createdate    | is_publicaccessible | share_acl | producer_account |
producer_namespace |
```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+
| myproducer_share | 100          | sample_data_dev | myconsumer_db   | INBOUND
| NULL            | true         | NULL            | producer-acct  | your-
producer-namespace |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+

```

上述命令會傳回由堆疊建立的資料共用名稱，稱為 `myproducer_share`。也會傳回與資料共用相關聯的資料庫名稱 (`myconsumer_db`)。

複製生產者命名空間識別碼，以便在稍後步驟中使用。

3. 描述資料共用中的物件。

```
DESC DATASHARE myproducer_share;
```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+
| producer_account | producer_namespace | share_type |
share_name        | object_type        | object_name | include_new |
+-----+-----+-----+-----+
+-----+
| producer-acct | your-producer-namespace | OUTBOUND |
myproducer_share | schema             | myproducer_schema | true
|
| producer-acct | your-producer-namespace | OUTBOUND |
myproducer_share | table              | myproducer_schema.tickit_sales | NULL
|
| producer-acct | your-producer-namespace | OUTBOUND |
myproducer_share | view               | myproducer_schema.ticket_sales_view | NULL
|
+-----+-----+-----+-----+
+-----+-----+-----+-----+
+-----+

```

當您描述資料共用時，其會傳回資料表和檢視的屬性。堆疊會將含有範例資料的資料表和檢視新增至生產者資料庫，例如 `tickit_sales` 和 `tickit_sales_view`。如需 TICKIT 範例資料庫的詳細資訊，請參閱 [範本資料庫](#)。

您不必委派對資料共用執行查詢的許可。堆疊會授予必要許可。

4. 使用用戶端工具連線至取用者叢集。描述資料共用，指定生產者的命名空間。

```
DESC DATASHARE myproducer_share OF NAMESPACE '<namespace id>'; --specify the unique
  identifier for the producer namespace
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| producer_account |          producer_namespace          | share_type |
share_name      | object_type |          object_name          | include_new |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
|  producer-acct |          your-producer-namespace          | INBOUND    |
myproducer_share | schema      | myproducer_schema              | NULL        |
|
|  producer-acct |          your-producer-namespace          | INBOUND    |
myproducer_share | table       | myproducer_schema.tickit_sales | NULL        |
|
|  producer-acct |          your-producer-namespace          | INBOUND    |
myproducer_share | view        | myproducer_schema.ticket_sales_view | NULL        |
|
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
```

5. 您可以透過指定資料共用的資料庫和結構描述來查詢資料共用中的資料表。如需詳細資訊，請參閱 [跨資料庫查詢範例](#)。下列查詢會從 TICKIT 範例資料庫中的 SALES 資料表傳回銷售和銷售者資料。如需詳細資訊，請參閱 [範本資料庫](#)。

```
SELECT * FROM myconsumer_db.myproducer_schema.tickit_sales_view;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
commission |      saletime      |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```

+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
|      1 |      1 |   36861 |   21191 |    7872 |    1875 |      4 |      728 |
109.2 | 2008-02-18 02:36:48 |
|      2 |      4 |    8117 |   11498 |    4337 |    1983 |      2 |      76 |
11.4 | 2008-06-06 05:00:16 |
|      3 |      5 |    1616 |   17433 |    8647 |    1983 |      2 |     350 |
52.5 | 2008-06-06 08:26:17 |
|      4 |      5 |    1616 |   19715 |    8647 |    1986 |      1 |     175 |
26.25 | 2008-06-09 08:38:52 |
|      5 |      6 |   47402 |   14115 |    8240 |    2069 |      2 |     154 |
23.1 | 2008-08-31 09:17:02 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+

```

Note

查詢會針對共用結構描述中的檢視執行。您不能直接連線到從資料共用中建立的資料庫。它們是唯讀金鑰。

6. 若要執行包含彙總的查詢，請使用下列範例。

```

SELECT * FROM myconsumer_db.myproducer_schema.tickit_sales ORDER BY 1,2 LIMIT 5;

+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| salesid | listid | sellerid | buyerid | eventid | dateid | qtysold | pricepaid |
commission |      saletime      |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
|      1 |      1 |   36861 |   21191 |    7872 |    1875 |      4 |      728 |
109.2 | 2008-02-18 02:36:48 |
|      2 |      4 |    8117 |   11498 |    4337 |    1983 |      2 |      76 |
11.4 | 2008-06-06 05:00:16 |
|      3 |      5 |    1616 |   17433 |    8647 |    1983 |      2 |     350 |
52.5 | 2008-06-06 08:26:17 |
|      4 |      5 |    1616 |   19715 |    8647 |    1986 |      1 |     175 |
26.25 | 2008-06-09 08:38:52 |
|      5 |      6 |   47402 |   14115 |    8240 |    2069 |      2 |     154 |
23.1 | 2008-08-31 09:17:02 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+

```

查詢會從範例資料傳回銷售和銷售者TICKIT資料。

如需更多資料共用查詢範例，請參閱 [共用 內資料的讀取存取權 AWS 帳戶](#)。

使用主控台透過寫入功能管理資料共用 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 PREVIEW_2023 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以使用主控台管理資料共用與寫入，以控制 Amazon Redshift 叢集和 AWS 帳戶之間的存取和管理資料。下列各節提供使用 Amazon Redshift 主控台設定和管理資料共用與寫入 step-by-step 的指示。

如需設定 PREVIEW_2023 追蹤的詳細資訊，請參閱下列其中一項：

- 對於 Amazon Redshift Serverless 預覽：[建立預覽工作群組](#)
- 對於 Amazon Redshift 佈建的叢集預覽：[建立預覽叢集](#)

如需開始使用資料共用的詳細資訊，請前往 [共用資料的寫入存取權 \(預覽\)](#)。

使用 Amazon Redshift 主控台管理在您帳戶中建立或從其他帳戶共用的資料共用。

連線至資料庫 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 PREVIEW_2023 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以使用資料共用跨 AWS 帳戶和 Amazon Redshift 叢集連線至資料庫。

如需開始使用資料共用的詳細資訊，請前往 [共用資料寫入存取權 \(預覽\)](#)。

連線至資料庫以檢視此叢集中的資料庫和資料庫中的物件，或檢視資料共用。

用來連線至指定資料庫的使用者憑證必須具備檢視所有資料共用的必要許可。

如果沒有本機連線，請執行下列其中一項動作：

- 在叢集詳細資訊頁面中，從資料庫索引標籤的資料庫或資料共用物件區段中，選擇連線到資料庫以檢視叢集中的資料庫物件。
- 在叢集詳細資訊頁面中，從資料共用索引標籤中執行下列其中一項操作：
 - 在來自其他叢集的資料共用區段中，選擇連線到資料庫以檢視其他叢集的資料共用。
 - 在我的叢集中建立的資料共用區段中，選擇連線到資料庫以檢視您叢集中的資料共用。
- 在連線到資料庫視窗中，執行下列其中一項動作：
 - 如果您選擇建立新連線，請選擇 AWS Secrets Manager 來使用預存密碼來驗證連線的存取權。

或者，選擇暫時憑證以使用資料庫憑證來驗證連線的存取權。指定資料庫名稱和資料庫使用者的值。

選擇連線。

- 選擇使用最近使用的連線來連線到您擁有必要許可的其他資料庫。

Amazon Redshift 會自動進行連線。

建立資料庫連線之後，您可以開始建立資料共用、查詢資料共用或從資料共用建立資料庫。

建立資料共用和新增物件 (預覽)

建立資料共用

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 `PREVIEW_2023` 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以使用資料共用在 Amazon Redshift 叢集或 AWS 帳戶之間共用即時資料。資料共用是消費者生產者物件，可讓您將來自 Amazon Redshift 叢集的即時資料與其他叢集或 AWS 帳戶共用。建立資料共用可啟用安全的資料共用，同時維持對存取的控制，並確保資料保持 up-to-date。下列各節提供建立資料共用和新增資料庫物件的詳細資訊，例如結構描述、資料表和檢視，以安全地共用即時資料。

如需開始使用資料共用的詳細資訊，請前往 [共用資料寫入存取權 \(預覽\)](#)。

身為生產者叢集管理員，您可以從叢集詳細資訊頁面的資料庫或資料共用索引標籤中建立資料共用。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。

2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。

3. 在叢集詳細資訊頁面上，執行下列作業：

- 從資料庫索引標籤的資料庫區段中，選擇資料庫。資料庫詳細資訊頁面會隨即出現。

選擇建立資料共用。您只能從本機資料庫建立資料共用。如果您尚未連線到資料庫，則會出現連線到資料庫頁面。依照 [連線至資料庫 \(預覽\)](#) 中的步驟連線到資料庫。如果有最近使用的連線，就會顯示建立資料共用頁面。

- 如果您沒有資料庫連線，請從資料共用索引標籤的資料共用區段中連線到資料庫。

在我的叢集中建立的資料共用區段中，選擇建立資料共用。隨即出現建立資料共用頁面。

4. 從這裡，您可以新增各種類型的資料庫物件。選擇新增按鈕以新增物件。隨即會出現對話方塊。執行以下步驟：

1. 選擇一個結構描述或多個結構描述。這樣做可讓結構描述中的物件加入。
2. 從結構描述選取物件類型。

從這裡您可以選擇兩個選項來新增物件：

- 從結構描述新增特定物件 — 如果您選擇此項，它會依名稱列示個別物件。您可以選取物件，並將其新增至資料共用。例如，您可以視需要新增特定的資料表和預存程序。然後，您選取的結構描述中的資料表和預存程序會包含在資料共用中。設定權限會在後續步驟中進一步說明。繼續執行檢視和其他類型，選取要加入的物件。
 - 將所選物件類型中的所有現有物件新增至結構描述 – 這會新增所有物件。
3. 您也可以選擇是否要新增未來的物件。當您選擇包括新增至結構描述的資料共用物件時，表示新增至結構描述的物件會自動新增至資料共用。
 4. 選擇新增以完成區段，並新增物件。這些全列在資料共用物件之下。
 5. 加入物件之後，您可以選取個別物件，並編輯其權限。如果您選取結構描述，會出現對話方塊，詢問您是否要新增限定範圍權限。這使得結構描述中的每個現有或新增的物件都有一組預先選取的權限，適用於該物件類型。例如，管理員可以設定所有新增資料表具有 SELECT 和 UPDATE 許可，例如。
 6. 設定結構描述權限後，您可以逐步瀏覽其他物件類型並選取其權限。例如，您可以將 UPDATE 許可新增至特定資料表。
 7. 在資料取用者區段中，您可以新增命名空間或新增 AWS 帳戶作為資料共用的取用者。

8. 選擇建立資料共用以儲存變更。

建立資料共用之後，它會出現在我的命名空間中建立的資料共用下的清單中。如果您從清單中選擇資料共用，您可以檢視其取用者、物件及其他屬性。

將資料取用者新增至資料共用

您可以將一個或多個資料取用者新增至資料共用。資料取用者可以是唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

您必須明確選擇關閉或開啟與具有公用存取權的叢集共用您的資料共用。

- 選擇將叢集命名空間新增至資料共用。命名空間是 Amazon Redshift 叢集的全域唯一識別碼 (GUID)。
- 選擇新增 AWS 帳戶 至資料共用。指定的 AWS 帳戶 必須具有資料共用的存取權限。

授予或移除資料共用的授權 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 PREVIEW_2 023 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以透過授權或撤銷指定取用者的授權來控制對資料共用的存取。資料共用可讓您在相同或不同的 AWS 帳戶中跨 Amazon Redshift 叢集共用即時資料，提供順暢的分發和使用分析資料的方式。下列各節提供 step-by-step 在 Amazon Redshift 中授權和撤銷取用者存取資料共用的指示。

如需開始使用資料共用的詳細資訊，請前往 [共用資料寫入存取權 \(預覽\)](#)。

身為生產者叢集管理員，請選擇要授權存取資料共用或從中移除授權的資料取用者。獲授權的資料取用者會收到對資料共用採取動作的通知。如果您要將叢集命名空間新增為資料取用者，則不必執行授權。

先決條件：若要授予或移除資料共用的授權，必須至少已將一個資料取用者新增至資料共用。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。從這裡可以看到名為 資料共用取用者的清單。選擇您要授權的一個或多個取用者叢集。然後選擇授權。

3. 系統顯示授權帳戶對話方塊。您可以選擇一兩種授權類型。
 - 在 [叢集名稱或工作群組名稱] 上唯讀 — 這表示取用者沒有可用的寫入權限，即使資料共用建立者授予寫入權限也一樣。
 - 在 [叢集名稱或工作群組名稱] 上讀取和寫入 — 這表示建立者授予的所有權限 (包括寫入權限) 均可供取用者使用。
4. 選擇 Save (儲存)。

您也可以授權 AWS Data Exchange 為取用者。

1. 如果您在建立資料共用時選擇發佈至 AWS Glue Data Catalog，則只能將資料共用的授權授予 Lake Formation 帳戶。

對於 AWS Data Exchange 資料共用，您一次只能授權一個資料共用。

當您授權 AWS Data Exchange 資料共用時，您將與 AWS Data Exchange 服務共用資料共用 AWS Data Exchange，並允許代表您管理對資料共用的存取權。當取用者訂閱產品時，會將取用者帳戶新增為 AWS Data Exchange 資料共用的資料取用者，AWS Data Exchange 以允許取用者存取資料共用。AWS Data Exchange 沒有資料共用的讀取存取權。

2. 選擇 Save (儲存)。

授權資料取用者之後，他們就可以存取資料共用物件，並建立取用者資料庫來查詢資料。

移除授權：

選擇您要從中移除授權的一個或多個取用者叢集。然後選擇移除授權。

移除授權後，資料取用者會立即失去對資料共用的存取權限。

以取用者身分關聯或拒絕資料共用 (預覽)

使用 Amazon Redshift，您可以關聯或拒絕其他 AWS 帳戶共用的資料共用，實現跨組織邊界的無縫且安全的資料共用。Datashares 是可共用的資料庫物件，可從一或多個 Amazon Redshift 資料庫封裝資料。下列各節示範了與資料共用建立關聯或拒絕共用的程序，讓您釋放 Amazon Redshift 環境中共用資料資源的全部潛力。

建立資料共用的關聯

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 PREVIEW_2023 追蹤的公開預覽中取得。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

如需開始使用資料共用的詳細資訊，請前往 [共用資料寫入存取權 \(預覽\)](#)。

身為取用者叢集管理員，您可以將一個或多個從其他帳戶共用的資料共用與整個 AWS 帳戶或帳戶中的特定叢集命名空間建立關聯。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。選擇來自其他帳戶。
3. 在來自其他帳戶的資料共用區段中，選擇您要建立關聯的資料共用，然後選擇建立關聯。當顯示關聯資料共用頁面時，選擇下列其中一個關聯類型：
 - 選擇整個 AWS 帳戶，將 AWS 帳戶中不同 AWS 區域的所有現有和未來叢集命名空間與資料共用建立關聯。

如果資料共用發佈至 AWS Glue Data Catalog，您只能將資料共用與整個 AWS 帳戶建立關聯。
4. 您可以從這裡選擇允許的權限。選擇如下：
 - 唯讀 – 如果您選擇唯讀，即使這些許可已授予並授權給生產者，也要在取用者上寫入類似 UPDATE 或 INSERT 無法使用的許可。
 - 讀取和寫入 — 取用者資料共用使用者將擁有生產者授予和授權的所有權限，包括讀取和寫入。
5. 或選擇特定 AWS 區域和叢集命名空間，將一或多個 AWS 區域和特定叢集命名空間與資料共用建立關聯。選擇新增區域，將特定 AWS 區域和叢集命名空間新增至資料共用。隨即顯示新增 AWS 區域頁面。
6. 選擇 AWS 區域。
7. 執行以下任意一項：
 - 選擇新增所有叢集命名空間，將此區域中所有現有和未來的叢集命名空間新增至資料共用。
 - 選擇新增特定叢集命名空間，將此區域中的一或多個特定叢集命名空間新增至資料共用。
 - 選擇一或多個叢集命名空間，然後選擇新增 AWS 區域。
8. 選擇關聯。

生產者可以傳回並變更授權的設定，這可能會影響取用者的關聯設定。

如果您要將資料共用與 Lake Formation 帳戶建立關聯，請前往 Lake Formation 主控台建立資料庫，然後定義資料庫的許可。如需詳細資訊，請參閱 AWS Lake Formation 開發人員指南中的[設定 Amazon Redshift 資料共用的許可](#)。建立 AWS Glue 資料庫或聯合資料庫後，您可以將查詢編輯器 v2 或任何偏好的 SQL 用戶端與取用者叢集搭配使用來查詢資料。。

與資料建立關聯之後，資料共用就會變為可用。

您也可以隨時變更資料共用關聯。將特定 AWS 區域和叢集命名空間的關聯變更為整個 AWS 帳戶時，Amazon Redshift 會使用 AWS 帳戶資訊覆寫特定區域和叢集命名空間資訊。AWS 帳戶中的所有 AWS 區域和叢集命名空間，即可存取資料共用。

當將關聯從特定叢集命名空間變更為指定 AWS 區域中的所有叢集命名空間時，此區域中的所有叢集命名空間都可以存取資料共用。

從資料取用者移除資料共用的關聯

身為取用者叢集管理員，您可以從資料取用者移除資料共用的關聯。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。資料共用清單頁面會隨即出現。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇要從資料取用者中移除關聯的資料共用。
5. 在資料取用者段落中，選擇一或多個要從中移除關聯的資料取用者。然後選擇移除關聯。
6. 當移除關聯頁面出現時，請選擇移除關聯。

移除關聯之後，資料取用者將失去對資料共用的存取權。您可以隨時變更資料取用者關聯。

拒絕資料共用

身為取用者叢集管理員，您可以拒絕任何狀態為可用或作用中的資料共用。拒絕資料共用後，取用者叢集使用者將失去資料共用的存取權。如果您呼叫 DescribeDataSharesForConsumerAPI 操作，Amazon Redshift 不會傳回被拒絕的資料共用。如果生產者叢集管理員執行 DescribeDataSharesForProducerAPI 操作，他們就會看到資料共用遭到拒絕。一旦資料共用遭到拒絕，生產者叢集管理員可以再次將資料共用授權給取用者叢集，取用者叢集管理員可以選擇將其 AWS 帳戶與資料共用建立關聯或拒絕。

如果 AWS 您的帳戶與資料共用有關聯，且與 Lake Formation 管理的資料共用有待定關聯，拒絕 Lake Formation 管理的資料共用關聯也會拒絕原始資料共用。若要拒絕特定關聯，生產者叢集管理員可以從指定的資料共用中移除授權。此動作不會影響其他資料共用。

若要拒絕資料共用，請使用 AWS 主控台、API 操作 `RejectDataShare` 或 `reject-datashare` 中的 AWS CLI。

若要使用 AWS 主控台拒絕資料共用：

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表中，選擇資料共用。
3. 選擇來自其他帳戶。
4. 在來自其他帳戶的資料共用區段中，選擇您要拒絕的資料共用。出現拒絕資料共用頁面時，選擇拒絕。

拒絕資料共用之後，您就無法還原變更。Amazon Redshift 會從清單中移除資料共用。若要再次查看資料共用，生產者管理員必須再次對其授權。

管理現有資料共用 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 `PREVIEW_2023` 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以管理現有的資料共用，以控制對 Amazon Redshift 叢集中資料的存取。下列各節提供修改資料共用物件、管理資料共用許可和更新資料共用屬性 step-by-step 的指引，以有效控制和稽核 Amazon Redshift 環境中的資料存取。

如需開始使用資料共用的詳細資訊，請前往 [共用資料寫入存取權 \(預覽\)](#)。

檢視資料共用

從 `DATASHARES` 或 `CLUSTERS` 索引標籤檢視資料共用。

- 使用 `DATASHARES` 索引標籤列出您帳戶中或其他帳戶中的資料共用。
 - 若要檢視您帳戶中建立的資料共用，請選擇在我的帳戶中，然後選擇您要檢視的資料共用。

- 若要檢視從其他帳戶共用的資料庫，請選擇來自其他帳戶，然後選擇您要檢視的資料共用。
- 使用 CLUSTERS 索引標籤列出叢集中或其他叢集中的資料共用。

連線到資料庫 如需詳細資訊，請參閱[連線至資料庫 \(預覽\)](#)。

然後從來自其他叢集的資料共用或在我叢集中建立的資料共用區段中選擇一個資料共用，以檢視其詳細資訊。

從資料共用中移除資料共用物件

您可以使用下列程序，從資料共用中移除一個或多個物件。

從資料共用中移除一個或多個物件

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱[連線至資料庫 \(預覽\)](#)。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 若要對資料共用移除一或多個資料共用物件，請執行下列其中一項操作：
 - 若要從資料共用中移除結構描述，請選擇一或多個結構描述。然後選擇移除。Amazon Redshift 會從資料共用中移除指定的結構描述和指定結構描述的所有物件。
 - 若要從資料共用中移除資料表和檢視，請選擇一或多個資料表和檢視。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有資料表和檢視。
 - 若要從資料共用中移除使用者定義函數，請選擇一或多個使用者定義函數。然後選擇移除。或者，您也可以選擇依結構描述移除，移除指定結構描述中的所有使用者定義函數。

從資料共用中移除資料取用者

您可以從資料共用中移除一或多個資料取用者。資料取用者可以是可唯一識別 Amazon Redshift 叢集或 AWS 帳戶的叢集命名空間。

從叢集命名空間或 IDs AWS 帳戶選擇一或多個資料取用者，然後選擇移除。

Amazon Redshift 會從資料共用中移除指定的資料取用者。您會立即失去資料共用的存取權。

編輯在您帳戶中建立的資料共用

使用主控台編輯在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。
4. 於在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。
5. 選擇您要編輯的資料共用，然後選擇編輯。資料共用詳細資訊頁面會隨即出現。
6. 在資料共用物件或資料取用者區段中進行任何變更。

Note

如果您選擇將資料共用發佈至 AWS Glue Data Catalog，則無法編輯組態，將資料共用發佈至其他 Amazon Redshift 帳戶。

7. 選擇 Save changes (儲存變更)。

Amazon Redshift 會以這些變更來更新您的資料共用。

刪除在您帳戶中建立的 資料共用

使用主控台刪除在您帳戶中建立的資料共用。首先連線到資料庫以查看在您帳戶中建立的資料共用清單。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 於在我的帳戶中建立的資料共用區段中，選擇連線到資料庫。
5. 選擇一或多個您要刪除的資料共用，然後選擇刪除。刪除資料共用頁面會隨即出現。

刪除與 Lake Formation 共用的資料共用不會自動移除 Lake Formation 中的相關許可。若要將其刪除，請前往 Lake Formation 控制台。

6. 輸入刪除以確認刪除指定的資料共用。

7. 選擇刪除。

刪除資料共用之後，資料共用取用者會失去資料共用的存取權。

查詢資料共用 (預覽)

這是多資料倉儲透過 Amazon Redshift 的資料共用功能寫入的預先發行文件，可在 PREVIEW_2023 追蹤的公開預覽中找到。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版服務參與。

使用 Amazon Redshift，您可以從生產者叢集查詢資料共用之間的資料，以安全地存取即時資料，而無需複製或傳輸。下列各節涵蓋在您的 Amazon Redshift 環境中設定和查詢資料共用的詳細資訊。

如需開始使用資料共用的詳細資訊，請前往 [共用資料寫入存取權 \(預覽\)](#)。

從資料共用中建立資料庫

若要開始查詢資料共用中的資料，請從資料共用中建立資料庫。僅可從指定資料共用中建立一個資料庫。

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇您的叢集。叢集詳細資訊頁面會隨即出現。
3. 選擇資料共用。資料共用清單會隨即出現。
4. 在來自其他叢集的資料共用區段中，選擇連線到資料庫。如需詳細資訊，請參閱 [連線至資料庫 \(預覽\)](#)。
5. 選擇您要在其中建立資料庫的資料共用，然後選擇從資料共用建立資料庫。從資料共用建立資料庫頁面會隨即出現。
6. 在資料庫名稱中，指定資料庫名稱。資料庫名稱必須是 1–64 個英數字元 (僅限小寫字母)，且不能是保留字。
7. 選擇 Create (建立)。

建立資料庫之後，您可以查詢資料庫中的資料或執行寫入作業 (如果取用者管理員已授予、授權和關聯)。

Amazon Redshift 中的半結構化資料

透過使用 Amazon Redshift 中的半結構化資料支援，您可以在 Amazon Redshift 資料倉儲中擷取和儲存半結構化資料。Amazon Redshift 使用 SUPER 資料類型和 PartiQL 語言，擴展資料倉儲功能，以與 SQL 和無 SQL 資料來源整合。如此一來，Amazon Redshift 就能對關聯式和半結構化儲存資料進行有效的分析，例如 JSON。

Amazon Redshift 提供兩種半結構化資料支援：SUPER 資料類型和 Amazon Redshift Spectrum。

如果您需要插入或更新低延遲的小批次 JSON 資料，請使用 SUPER 資料類型。此外，當您的查詢需要強大的一致性、可預測的查詢效能、複雜的查詢支援，以及與不斷演進的結構描述和無結構描述資料搭配使用的易用性 SUPER 時，請使用。

相反地，如果您的資料查詢需要與其他 AWS 服務整合，以及主要存放在 Amazon S3 中的資料用於封存，請使用 Amazon Redshift Spectrum 搭配開放檔案格式。

SUPER 資料類型的使用案例

使用 Amazon Redshift 中的 SUPER 資料類型的半結構化資料支援可提供卓越的效能、彈性和易用性。下列使用案例有助於示範如何搭配使用半結構化資料支援 SUPER。

快速且彈性地插入 JSON 資料 – Amazon Redshift 支援可剖析 JSON 並儲存為 SUPER 值的快速交易。插入交易的運作速度比在已將屬性分割為 SUPER 傳統資料欄的資料表中執行相同的插入速度快上五倍。例如，假設傳入的格式 JSON 為 {“a” : ...、“b” : ...、“c”...、...}。您可以使用單一資料 SUPER 欄 S 將傳入的存放在資料表 TJ JSON 中，而不是使用資料欄「a」、「b」、「c」等，來加速插入效能。當中有數百個屬性時 JSON，SUPER 資料類型的效能優勢會變得相當龐大。

此外，SUPER 資料類型不需要一般結構描述。您不需要在儲存 JSON 之前先介紹和清除傳入的內容。例如，假設傳入 JSON 具有字串“c”屬性，而其他具有整數“c”屬性，但沒有 SUPER 資料類型。在這種情況下，您必須分隔 c_string 和 c_int 欄或清理資料。相反地，與 SUPER 資料類型相反，JSON 所有資料都會在擷取期間儲存，而不會遺失資訊。稍後，您可以使用的 PartiQL 延伸 SQL 來分析資訊。

靈活的探索查詢 – 將半結構化資料（例如 JSON）儲存到 SUPER 資料值後，您可以查詢該資料，而不需要內嵌結構描述。您可以使用 PartiQL 動態類型和寬鬆的語義來執行查詢並探索您需要的深度巢狀資料，而不需要在查詢之前強加結構描述。

擷取、載入、將（ETL）操作轉換為傳統具體化視觀表的彈性查詢 – 將無結構和半結構化資料儲存到之後 SUPER，您可以使用 PartiQL 具體化視觀表來介紹資料並將其切碎為具體化視觀表。

具有分解資料的具體化視觀表是傳統分析案例的效能和可用性優勢的一個很好的範例。當您對分解資料執行分析時，Amazon Redshift 具體化視觀表的單欄式組織可提供更好的效能。此外，需要擷取資料的傳統結構描述的使用者和商業智慧 (BI) 工具可以使用檢視 (無論是具體化或虛擬化) 做為傳統的資料結構描述呈現方式。

在您的 PartiQL 具體化視觀表將 JSON 或 中找到的資料擷取 SUPER 到傳統的欄式具體化視表之後，您可以查詢具體化視觀表。如需 SUPER 資料類型如何使用具體化檢視的詳細資訊，請參閱 [SUPER 資料類型和具體化檢視](#)。

您可以將動態資料遮罩政策套用至 SUPER 類型資料欄路徑上的 scalar 值。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。如需搭配 SUPER 資料類型使用動態資料遮罩的相關資訊，請參閱 [搭配 SUPER 資料類型路徑使用動態資料遮罩](#)。(預覽)

如需 SUPER 資料類型的相關資訊，請參閱 [SUPER 類型](#)。

如需使用 SUPER 資料類型的範例，請參閱此主題的子節，以開頭 [SUPER 範例資料集](#)。

SUPER 資料類型使用的概念

接下來，您可以找到一些 Amazon Redshift SUPER 資料類型概念。

了解 SUPER 資料類型在 Amazon Redshift 中是什麼 – SUPER 資料類型是一種 Amazon Redshift 資料類型，可儲存包含 Amazon Redshift 純量和可能巢狀的陣列和結構的無結構描述陣列和結構。SUPER 資料類型可以原生儲存不同格式的半結構化資料，例如源自文件導向來源 JSON 的資料。您可以新增資料 SUPER 欄來儲存半結構化資料，並寫入存取資料 SUPER 欄的查詢，以及一般純量資料欄。如需 SUPER 資料類型的詳細資訊，請參閱 [SUPER 類型](#)。

JSON 將無結構描述輸入 SUPER - Amazon Redshift 可以接收並擷取無結構描述 JSON SUPER 的值，並使用彈性半結構化 SUPER 資料類型。例如，Amazon Redshift 可以將 JSON 值 `【10.5, "first"】` 擷取至 SUPER 值 `【10.5, 'first'】`，即包含 Amazon Redshift 十進位 10.5 和 varchar 'first' 的陣列。Amazon Redshift 可以使用 COPY 命令或 JSON 剖析函數，例如 `json_parse ('【10.5, "first"】')`，將擷取 JSON 至 SUPER 值。根據預設，COPY 和 `json_parse` 都會 JSON 使用嚴格剖析語意進行擷取。您也可以使用資料庫資料本身建構包括陣列和結構 SUPER 的值。

擷取結構描述的不規則結構時，資料 SUPER 欄不需要修改結構描述 JSON。例如，在分析點擊串流時，您最初會存放在具有屬性 "IP" 和 "時間" 的「點擊」結構 SUPER 欄中。您可以在不變更結構描述的情況下新增屬性「customer id」，以擷取此類變更。

用於 SUPER 資料類型的原生格式是二進位格式，需要的空間比文字格式中的 JSON 值小。這可讓查詢時更快擷取和執行時間處理 SUPER 值。

使用 PartiQL 查詢 SUPER 資料 – PartiQL 是 SQL-92 的向後相容延伸，許多 AWS 服務目前使用。使用 PartiQL 時，熟悉的 SQL 建構可無縫結合對傳統表格 SQL 資料和半結構化資料的存取 SUPER。您可以執行物件和陣列導覽以及解除巢狀化陣列。PartiQL 將標準 SQL 語言擴展為宣告性表達和處理巢狀和多值資料。

PartiQL 是 的延伸 SQL，其中資料 SUPER 欄的巢狀和無結構描述資料是一流的公民。PartiQL 不要求在查詢編譯時對所有查詢運算式進行類型檢查。此方法可讓包含 SUPER 資料類型的查詢表達式，在存取 SUPER 資料欄內的實際資料類型時，在查詢執行期間動態輸入。此外，PartiQL 在寬鬆模式下運作，其中類型不一致不會導致失敗，但會傳回 null。無結構描述和延遲查詢處理的組合使得 PartiQL 非常適合擷取、載入、傳輸（ELT）應用程式，其中您的 SQL 查詢會評估擷取於 SUPER 資料欄中 JSON 的資料。

與 Redshift Spectrum 整合 – Amazon Redshift 在透過 JSON、Parquet 和具有巢狀資料的其他格式執行 Redshift Spectrum 查詢時，支援 PartiQL 的多個方面。Redshift Spectrum 僅支援具有結構描述的巢狀資料。例如，使用 Redshift Spectrum，您可以宣告 JSON 資料在結構描述 `ARRAY <STRUCT<a : INTEGER, b : DECIMAL (5, 2) >>` 中具有屬性巢狀_結構描述範例。此屬性的結構描述決定了資料永遠包含陣列，其中包含整數 a 和小數 b 的結構。如果資料變更為包含更多屬性，類型也會變更。相反地，SUPER 資料類型不需要結構描述。您可以使用具有不同屬性或類型的結構元素來儲存陣列。此外，一些值可以儲存在陣列之外。

如需支援 SUPER 資料類型之函數的相關資訊，請參閱下列內容：

- [ABS 函數](#)
- [CEILING \(或 CEIL \) 函數](#)
- [FLOOR 函數](#)
- [ROUND 函數](#)
- [SIGN 函數](#)
- [TRUNC 函數](#)

SUPER 資料考量事項

使用 SUPER 資料時，請考慮下列事項：

- 使用 JDBC 驅動程式 1.2.50 版、ODBC 驅動程式 1.4.17 版或更新版本，以及 Amazon Redshift Python 驅動程式 2.0.872 版或更新版本。

如需 JDBC 驅動程式的相關資訊，請參閱 [設定 JDBC 連線](#)。

如需ODBC驅動程式的相關資訊，請參閱[設定ODBC連線](#)。

- 在 [SUPER 範例資料集](#) 中尋找下列主題中使用的結構描述範例。
- 下列主題中使用的所有SQL程式碼範例都包含相同的 S3 字首以供下載。其中包括資料定義語言 (DDL) 和COPY陳述式，以及使用的特定 TPC-H 修改查詢SUPER。

若要檢視或下載SQL檔案，請執行下列其中一項操作：

- 下載[SUPER教學課程SQL檔案](#) 和 [TPC-H 檔案](#)。
- 使用 Amazon S3 CLI執行下列命令。您可以使用自己的目標路徑。

```
aws s3 cp s3://redshift-downloads/semistructured/tutorialscripts/semistructured-tutorial.sql /target/path
aws s3 cp s3://redshift-downloads/semistructured/tutorialscripts/super_tpch_queries.sql /target/path
```

如需SUPER組態的詳細資訊，請參閱 [SUPER 組態](#)。

SUPER 範例資料集

您可以探索和分析SUPER範例資料集，其中包含與跨各種類別、區域和時段的虛構產品銷售相關的資料。下列各節提供存取、查詢和操作 Amazon Redshift 叢集中SUPER範例資料集的詳細資訊。

用於擷取和查詢範例的資料表結構描述和資料模型定義如下。

```
/*customer-orders-lineitem*/
CREATE TABLE customer_orders_lineitem
(c_custkey bigint
,c_name varchar
,c_address varchar
,c_nationkey smallint
,c_phone varchar
,c_acctbal decimal(12,2)
,c_mktsegment varchar
,c_comment varchar
,c_orders super
);

/* Datamodel of documents to be stored in c_orders Super column would be as follows*/
ARRAY < STRUCT < o_orderkey:bigint
, o_orderstatus:string
```

```

        ,o_totalprice:double
        ,o_orderdate:string
        ,o_orderpriority:string
        ,o_clerk:string
        ,o_shippriority:int
        ,o_comment:string
        ,o_lineitems:ARRAY < STRUCT < l_partkey:bigint
                                   ,l_suppkey:bigint
                                   ,l_linenumber:int
                                   ,l_quantity:double
                                   ,l_extendedprice:double
                                   ,l_discount:double
                                   ,l_tax:double
                                   ,l_returnflag:string
                                   ,l_linestatus:string
                                   ,l_shipdate:string
                                   ,l_commitdate:string
                                   ,l_receiptdate:string
                                   ,l_shipinstruct:string
                                   ,l_shipmode:string
                                   ,l_comment:string
                                   > >
        > >

/*part*/
CREATE TABLE part
(
  p_partkey bigint
  ,p_name varchar
  ,p_mfgr varchar
  ,p_brand varchar
  ,p_type varchar
  ,p_size int
  ,p_container varchar
  ,p_retailprice decimal(12,2)
  ,p_comment varchar
);

/*region-nations*/
CREATE TABLE region_nations
(
  r_regionkey smallint
  ,r_name varchar
  ,r_comment varchar

```

```
,r_nations super
);

/* Datamodel of documents to be stored in r_nations Super column would be as follows*/
ARRAY < STRUCT < n_nationkey:int,n_name:string,n_comment:string > >

/*supplier-partssupp*/
CREATE TABLE supplier_partssupp
(
  s_suppkey bigint
  ,s_name varchar
  ,s_address varchar
  ,s_nationkey smallint
  ,s_phone varchar
  ,s_acctbal double precision
  ,s_comment varchar
  ,s_partssups super
);

/* Datamodel of documents to be stored in s_partssups Super column would be as follows*/
ARRAY < STRUCT <
ps_partkey:bigint,ps_availqty:int,ps_supplycost:double,ps_comment:string > >
```

將半結構化資料載入 Amazon Redshift

使用SUPER資料類型在 Amazon Redshift 中保留和查詢階層和一般資料。Amazon Redshift 引入 `json_parse` 函數以 JSON 格式剖析資料，並將其轉換為SUPER表示式。Amazon Redshift 也支援使用 COPY 命令載入SUPER資料欄。支援的檔案格式為 JSON、Avro、文字、逗號分隔值（CSV）格式、Parquet 和 ORC。

如需下列範例中所用資料表的詳細資訊，請參閱[SUPER 範例資料集](#)。

如需 `json_parse` 函數的詳細資訊，請參閱[JSON_PARSE 函數](#)。

SUPER 資料類型的預設編碼為 ZSTD。

剖析JSON文件至SUPER資料欄

您可以使用 `json_parse` 函數將JSON資料插入或更新至資料SUPER欄。函數會剖析 JSON 格式的資料，並將其轉換為SUPER資料類型，您可以在 INSERT 或 UPDATE 陳述式中使用。

下列範例會將JSON資料插入SUPER資料欄。如果查詢中缺少`json_parse`函數，Amazon Redshift 會將值視為單一字串，而不是必須剖析的JSON格式化字串。

如果您更新SUPER資料欄，Amazon Redshift 會要求將完整文件傳遞至資料欄值。Amazon Redshift 不支援部分更新。

```
INSERT INTO region_nations VALUES(0,
  'lar deposits. blithely final packages cajole. regular waters are final requests.
regular accounts are according to',
  'AFRICA',
  JSON_PARSE('{"r_nations":[
    {"n_comment":" haggle. carefully final deposits detect slyly agai",
      "n_nationkey":0,
      "n_name":"ALGERIA"
    },
    {"n_comment":"ven packages wake quickly. regu",
      "n_nationkey":5,
      "n_name":"ETHIOPIA"
    },
    {"n_comment":" pending excuses haggle furiously deposits. pending, express pinto
beans wake fluffily past t",
      "n_nationkey":14,
      "n_name":"KENYA"
    },
    {"n_comment":"rns. blithely bold courts among the closely regular packages use
furiously bold platelets?",
      "n_nationkey":15,
      "n_name":"MOROCCO"
    },
    {"n_comment":"s. ironic, unusual asymptotes wake blithely r",
      "n_nationkey":16,
      "n_name":"MOZAMBIQUE"
    }
  ]
}')));
```

使用 COPY 在 Amazon Redshift 中載入SUPER資料欄

在下列各節中，您可以了解使用 COPY命令將JSON資料載入 Amazon Redshift 的不同方法。

從 JSON 和 Avro 複製資料

透過在 Amazon Redshift 中使用半結構化資料支援，您可以載入 JSON 文件，而無需將其 JSON 結構的屬性分割為多個資料欄。

Amazon Redshift 提供兩種使用擷取 JSON 文件的方法 COPY，即使 JSON 結構完全或部分未知：

1. 使用 `noshred` 選項，將從 JSON 文件衍生的資料儲存到單一 SUPER 資料欄。當結構描述不知道或預期會改變時，此方法非常有用。因此，此方法可讓您更輕鬆地將整個組分存放在單一 SUPER 資料欄中。
2. 使用 `auto` 或 `jsonpaths` 選項，將 JSON 文件分割成多個 Amazon Redshift 資料欄。屬性可以是 Amazon Redshift 純量或 SUPER 值。

您可以將這些選項與 JSON 或 Avro 格式搭配使用。

碎紙前 JSON 物件的大小上限為 4 MB。

將 JSON 文件複製到單一 SUPER 資料欄

若要將 JSON 文件複製到單一 SUPER 資料欄，請使用單一 SUPER 資料欄建立資料表。

```
CREATE TABLE region_nations_noshred (rdata SUPER);
```

將資料從 Amazon S3 複製到單一 SUPER 資料欄。若要將 JSON 來源資料擷取至單一 SUPER 資料欄，請在 `FORMAT JSON` 子句中指定 `noshred` 選項。

```
COPY region_nations_noshred FROM 's3://redshift-downloads/semistructured/tpch-nested/
data/json/region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'noshred';
```

COPY 成功擷取後 JSON，資料表會有一個 `rdata` SUPER 資料欄，其中包含整個 JSON 物件的資料。擷取的資料會維護 JSON 階層的所有屬性。不過，分葉會轉換成 Amazon Redshift 純量類型，以便有效率地處理查詢。

使用下列查詢來擷取原始 JSON 字串。

```
SELECT rdata FROM region_nations_noshred;
```

當 Amazon Redshift 產生 SUPER 資料欄時，它可以透過 JSON 序列化使用 JDBC 作為字串來存取。如需詳細資訊，請參閱 [序列化複雜的嵌套 JSON](#)。

將 JSON 文件複製到多個 SUPER 資料欄

您可以將 JSON 文件分割成多個資料欄，可以是 SUPER 資料欄或 Amazon Redshift 純量類型。Amazon Redshift 會將 JSON 物件的不同部分分散到不同的資料欄。

```
CREATE TABLE region_nations
(
  r_regionkey smallint
  ,r_name varchar
  ,r_comment varchar
  ,r_nations super
);
```

若要將上一個範例的資料複製到資料表中，請在 FORMAT JSON 子句中指定 AUTO 選項，以在多個資料欄之間分割 JSON 值。COPY 會比對具有資料欄名稱的頂層 JSON 屬性，並允許巢狀值擷取為 SUPER 值，例如 JSON 陣列和物件。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'auto';
```

當 JSON 屬性名稱為混合大小寫時，請在 FORMAT JSON 子句中指定 auto ignorecase 選項。如需 COPY 命令的詳細資訊，請參閱 [使用「自動忽略案例」選項從 JSON 資料載入](#)。

在某些情況下，資料欄名稱和 JSON 屬性之間存在不相符，或者要載入的屬性巢狀超過層級深度。如果是這樣，請使用 jsonpaths 檔案手動將 JSON 屬性對應至 Amazon Redshift 資料欄。

```
CREATE TABLE nations
(
  regionkey smallint
  ,name varchar
  ,comment super
  ,nations super
);
```

假設您想要將資料載入至資料欄名稱與JSON屬性不相符的資料表。在下列範例中的 `nations` 表就是這種表格。您可以建立 `jsonpaths` 檔案，依據屬性在 `jsonpaths` 陣列中的位置，將屬性路徑對應至表格欄。

```
{
  "jsonpaths": [
    "$.r_regionkey",
    "$.r_name",
    "$.r_comment",
    "$.r_nations"
  ]
}
```

`jsonpaths` 檔案的位置會用作 `FORMAT` 的引數JSON。

```
COPY nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 's3://redshift-downloads/semistructured/tpch-nested/data/jsonpaths/
nations_jsonpaths.json';
```

使用下列查詢來存取顯示資料分佈到多個欄的資料表。使用 `JSON` 格式列印 `SUPER` 資料欄。

```
SELECT r_regionkey,r_name,r_comment,r_nations[0].n_nationkey FROM region_nations ORDER
BY 1,2,3 LIMIT 1;
```

`Jsonpaths` 檔案會將JSON文件中的欄位映射至資料表資料欄。您可以擷取其他資料欄，例如分佈和排序索引鍵，同時仍然將完整文件載入為資料 `SUPER` 欄。以下查詢將完整文件載入到 `nations` 欄。 `name` 欄是排序索引鍵，而 `regionkey` 欄是分散索引鍵。

```
CREATE TABLE nations_sorted (
  regionkey smallint,
  name varchar,
  nations super
) DISTKEY(regionkey) SORTKEY(name);
```

根 `jsonpath "$` 映射到文件的根目錄，如下所示：

```
{
  "jsonpaths": [
    "$.r_regionkey",
    "$.r_name",
```

```

    "$"
  ]
}

```

jsonpaths 檔案的位置會用作 FORMAT 的引數JSON。

```

COPY nations_sorted FROM 's3://redshift-downloads/semistructured/tpch-nested/data/json/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 's3://redshift-downloads/semistructured/tpch-nested/data/jsonpaths/
nations_sorted_jsonpaths.json';

```

從文字 和 複製資料 CSV

Amazon Redshift 以序列化 表示文字和CSV格式的資料SUPER欄JSON。資料SUPER欄需要有效的JSON格式，才能載入正確的類型資訊。取消物件、陣列、數字、布林值和 null 值的引號。將字串值換成雙引號。SUPER 資料欄使用文字和CSV格式的標準逸出規則。對於 CSV，分隔符號會根據CSV標準逸出。對於文字，如果選擇的分隔符號也可能出現在SUPER欄位中，請在 COPY和 期間使用ESCAPE選項UNLOAD。

```

COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/csv/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT CSV;

```

```

COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/text/
region_nation'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
DELIMITER ','
ESCAPE;

```

從資料欄格式 Parquet 和 複製資料 ORC

如果您的半結構化或巢狀資料已提供 Apache Parquet 或 Apache ORC 格式，您可以使用 COPY命令將資料擷取至 Amazon Redshift。

Amazon Redshift 資料表結構應符合 Parquet 或 ORC 檔案的資料欄數和資料欄資料類型。透過在 COPY命令SERIALIZETOJSON中指定，您可以在檔案中載入與資料表中的資料欄對齊的任何SUPER資料欄類型，作為 SUPER。這包括結構和陣列類型。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/parquet/region_nation'  
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'  
FORMAT PARQUET SERIALIZETOJSON;
```

下列範例使用 ORC 格式。

```
COPY region_nations FROM 's3://redshift-downloads/semistructured/tpch-nested/data/orc/region_nation'  
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'  
FORMAT ORC SERIALIZETOJSON;
```

當日期或時間資料類型的屬性在 ORC 中編碼時，Amazon Redshift 會在編碼時將其轉換為 varcharSUPER。

卸載半結構化資料

使用 Amazon Redshift，您可以將半結構化資料從 Amazon Redshift 叢集匯出至 Amazon S3，格式有多種，包括文字、Apache Parquet、Apache ORC 和 Avro。下列各節將引導您在 Amazon Redshift 中設定和執行半結構化資料的卸載操作。

CSV or text formats

您可以使用逗號分隔值 (CSV) 或文字格式，將包含 SUPER 資料欄的資料表卸載至 Amazon S3。Amazon Redshift 使用導覽和不巢子句的組合，以 SUPER 資料格式將階層式資料卸載至 Amazon S3 CSV 或文字格式。隨後，您可以針對未載入的資料建立外部資料表，並使用 Redshift Spectrum 進行查詢。如需使用 UNLOAD 和所需 IAM 許可的資訊，請參閱 [UNLOAD](#)。

在執行下列範例之前，請先使用 [將半結構化資料載入 Amazon Redshift](#) 中的程序填入 region_nations 表格。如需下列範例中所用資料表的詳細資訊，請參閱 [SUPER 範例資料集](#)。

下列範例將資料卸載到 Amazon S3。

```
UNLOAD ('SELECT * FROM region_nations')  
TO 's3://xxxxxx/'  
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3-Write'  
DELIMITER AS '|'   
GZIP  
ALLOWOVERWRITE;
```

與使用者定義字串代表 null 值的其他資料類型不同，Amazon Redshift 會使用 JSON 格式匯出 SUPER 資料欄，並表示為 Null，如 JSON 格式所決定。因此，SUPER 資料欄會忽略 UNLOAD 命令中使用的 NULL 【AS】 選項。

Parquet format

您可以 Parquet 格式將資料表與 SUPER 資料欄卸載至 Amazon S3。Amazon Redshift 代表 Parquet 中的 SUPER 資料欄作為 JSON 資料類型。這使半結構化資料可以在 Parquet 中表示。您可以使用 Redshift Spectrum 查詢這些資料欄，或使用 COPY 命令將其擷取回 Amazon Redshift。如需使用 UNLOAD 和所需 IAM 許可的資訊，請參閱 [UNLOAD](#)。

下列範例會以 Parquet 格式將資料卸載至 Amazon S3。

```
UNLOAD ('SELECT * FROM region_nations')
TO 's3://xxxxxxx/'
IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxxx:role/Redshift-S3-Write'
FORMAT PARQUET;
```

查詢半結構化資料

透過 Amazon Redshift，您可以查詢和分析半結構化資料，例如 JSON、Avro 或 Ion，以及結構化資料。半結構化資料是指具有彈性結構描述的資料，允許階層式或巢狀結構。下列各節示範使用 Amazon Redshift 對開放資料格式的支援查詢半結構化資料，讓您從複雜的資料結構中解除鎖定有價值的資訊。

Amazon Redshift 使用 PartiQL 語言，提供 SQL 關聯式、半結構化和巢狀資料的相容存取。

PartiQL 使用動態類型進行運作。這種方法可在結構化、半結構化和巢狀資料集的組合上實現直覺式篩選、聯結和彙總。存取巢狀資料時，PartiQL 語法會使用點符號和陣列下標來進行路徑導覽。它還允許 FROM 子句項目反覆運算陣列，並用於不巢狀操作。接下來，您可以找到不同查詢模式的說明，這些模式將 SUPER 資料類型的使用與路徑和陣列導覽、取消巢狀、取消樞紐和聯結相結合相結合。

如需下列範例中所用資料表的詳細資訊，請參閱 [SUPER 範例資料集](#)。

主題

- [Navigation \(導覽\)](#)
- [解除巢狀化查詢](#)
- [物件取消樞紐](#)
- [動態類型](#)
- [寬鬆的語義](#)

- [自我檢查的種類](#)
- [排序依據](#)

Navigation (導覽)

Amazon Redshift 使用 PartiQL，分別使用 [...] 括號和點符號啟用導覽到陣列和結構中。此外，您也可以使用點符號將導覽混合到結構中，以及使用括號符號將導覽混合到陣列中。例如，下列範例假設 `c_ordersSUPER` 資料欄是具有結構的陣列，且屬性名為 `o_orderkey`。

若要擷取 `customer_orders_lineitem` 資料表中的資料，請執行以下命令。將 IAM 角色取代為您自己的憑證。

```
COPY customer_orders_lineitem FROM 's3://redshift-downloads/semistructured/tpch-nested/
data/json/customer_orders_lineitem'
REGION 'us-east-1' IAM_ROLE 'arn:aws:iam::xxxxxxxxxxxx:role/Redshift-S3'
FORMAT JSON 'auto';

SELECT c_orders[0].o_orderkey FROM customer_orders_lineitem;
```

Amazon Redshift 也使用資料表別名做為表示法的字首。以下範例與前面的範例是相同的查詢。

```
SELECT cust.c_orders[0].o_orderkey FROM customer_orders_lineitem AS cust;
```

您可以在所有類型的查詢中使用點和括號符號，例如篩選、聯結和彙總。您可以在查詢中使用這些符號，其中通常有欄參考。下列範例使用可篩選結果的 SELECT 陳述式。

```
SELECT count(*) FROM customer_orders_lineitem WHERE c_orders[0]. o_orderkey IS NOT
NULL;
```

下列範例使用 GROUP BY 和 ORDER BY 子句中的括號和點導覽。

```
SELECT c_orders[0].o_orderdate,
       c_orders[0].o_orderstatus,
       count(*)
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderkey IS NOT NULL
GROUP BY c_orders[0].o_orderstatus,
         c_orders[0].o_orderdate
ORDER BY c_orders[0].o_orderdate;
```

解除巢狀化查詢

為了解決巢狀查詢，Amazon Redshift 使用 PartiQL 語法在 SUPER 陣列上反覆運算。它會使用查詢的 FROM 子句導覽陣列來實現此目的。使用上一個範例，下列範例會迭代 `c_orders` 的屬性值。

```
SELECT c.*, o FROM customer_orders_lineitem c, c.c_orders o;
```

取消巢狀語法是 FROM 子句的延伸。在標準中 SQL，子 FROM 句 `x (AS) y` 表示 `y` 會反覆運算關聯的每個組 `x`。在這種情況下，`x` 指的是關係，而 `y` 指的是關係 `x` 的別名。同樣地，使用 FROM 子句項目取消巢狀的 PartiQL 語法 `x (AS) y` 表示 `y (SUPER)` 陣列表達式 `x` 中的每個 `(SUPER)` 值反覆運算。在此情況下，`x` 是 SUPER 表達式，`y` 也是的別名 `x`。

左運算元也可以使用點和括號符號進行常規導覽。在上一個範例中，`customer_orders_lineitem c` 是 `customer_order_lineitem` 基底資料表上的迭代，而 `c.c_orders o` 是 `c.c_orders` 陣列表上的迭代。若要迭代 `o_lineitems` 屬性 (即陣列中的陣列)，您可以新增多個子句。

```
SELECT c.*, o, l FROM customer_orders_lineitem c, c.c_orders o, o.o_lineitems l;
```

使用 AT 關鍵字迭代陣列時，Amazon Redshift 也支援陣列索引。子句 `x AS y AT z` 會迭代陣列 `x` 並產生做為陣列索引的欄位 `z`。下列範例顯示陣列索引的運作方式。

```
SELECT c_name,
       orders.o_orderkey AS orderkey,
       index AS orderkey_index
FROM customer_orders_lineitem c, c.c_orders AS orders AT index
ORDER BY orderkey_index;
```

c_name	orderkey	orderkey_index
Customer#000008251	3020007	0
Customer#000009452	4043971	0

(2 rows)

下列範例會迭代純量陣列。

```
CREATE TABLE bar AS SELECT json_parse('{"scalar_array": [1, 2.3, 45000000]}') AS data;
SELECT index, element FROM bar AS b, b.data.scalar_array AS element AT index;
```

index	element
0	1
1	2.3
2	45000000


```

0 | 1
1 | 2.3
2 | 45000000
(3 rows)

```

下列範例會迭代多個層級的陣列。這個範例會使用多個解除巢狀化子句來迭代到最內層的陣列。f.multi_level_array AS 陣列會迭代 multi_level_array。陣列 AS 元素是在 multi_level_array 內的陣列的迭代。

```

CREATE TABLE foo AS SELECT json_parse('[[[1.1, 1.2], [2.1, 2.2], [3.1, 3.2]]]') AS
multi_level_array;

SELECT array, element FROM foo AS f, f.multi_level_array AS array, array AS element;

  array   | element
-----+-----
 [1.1,1.2] | 1.1
 [1.1,1.2] | 1.2
 [2.1,2.2] | 2.1
 [2.1,2.2] | 2.2
 [3.1,3.2] | 3.1
 [3.1,3.2] | 3.2
(6 rows)

```

如需 FROM 子句的詳細資訊，請參閱 [FROM 子句](#)。

物件取消樞紐

若要執行物件撤銷，Amazon Redshift 會使用 PartiQL 語法來反覆運算 SUPER 物件。它使用具有 UNPIVOT 關鍵字查詢子句 FROM 句來執行此操作。在這種情況下，表達式是 c.c_orders[0] 物件。範例查詢會反覆運算物件傳回的每個屬性。

```

SELECT attr as attribute_name, json_typeof(val) as value_type
FROM customer_orders_lineitem c, UNPIVOT c.c_orders[0] AS val AT attr
WHERE c_custkey = 9451;

```

```

attribute_name | value_type
-----+-----
o_orderstatus  | string
o_clerk        | string
o_lineitems    | array
o_orderdate    | string

```

```

o_shippriority | number
o_totalprice   | number
o_orderkey     | number
o_comment      | string
o_orderpriority | string
(9 rows)

```

與取消巢狀一樣，撤銷語法也是 FROM 子句的延伸。差別在於轉調語法使用 UNPIVOT 關鍵字來表示其正在對物件而不是陣列進行迭代。它使用 AS value_alias 迭代物件內的所有值，並使用 AT attribute_alias 迭代所有屬性。請考慮下列語法片段：

```
UNPIVOT expression AS value_alias [ AT attribute_alias ]
```

Amazon Redshift 支援在單一 FROM 子句中使用物件調用和陣列取消巢狀，如下所示：

```

SELECT attr as attribute_name, val as object_value
FROM customer_orders_lineitem c, c.c_orders AS o, UNPIVOT o AS val AT attr
WHERE c_custkey = 9451;

```

當您使用物件取消樞紐時，Amazon Redshift 不支援相關的取消樞紐。具體來說，假設您有多個在不同查詢層級中取消樞紐的範例，而內部未樞紐分析會參考外部層級的範例。Amazon Redshift 不支援這種類型的多重取消樞紐。

如需 FROM 子句的詳細資訊，請參閱 [FROM 子句](#)。如需示範如何使用 PIVOT 和查詢結構化資料的範例，UNPIVOT 請參閱 [PIVOT 和 UNPIVOT 範例](#)。

動態類型

動態類型不需要明確轉換從點和括號路徑中擷取的資料。Amazon Redshift 使用動態輸入來處理無結構描述 SUPER 資料，而無需在查詢中使用資料類型之前宣告。動態輸入會使用導覽至 SUPER 資料欄的結果，而不必明確將其轉換為 Amazon Redshift 類型。動態輸入在聯結和 GROUP BY 子句中最有用。下列範例使用的 SELECT 陳述式不需要將點和括號表達式明確轉換為一般 Amazon Redshift 類型。如需類型相容性和轉換的相關資訊，請參閱 [類型相容性與轉換](#)。

```

SELECT c_orders[0].o_orderkey
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderstatus = 'P';

```

當 c_orders[0].o_orderstatus 是字串 'P' 時，此查詢中的等號計算結果為 true。在所有其他情況下，等號評估為 false，包括相等參數是不同類型的情況。

動態和靜態類型

如果不使用動態類型，則無法確定 `c_orders[0].o_orderstatus` 是字串、整數還是結構。您只能判斷 `c_orders【0】.o_orderstatus` 是 SUPER 資料類型，可以是 Amazon Redshift 純量、陣列或結構。`c_orders【0】.o_orderstatus` 的靜態類型是 SUPER 資料類型。傳統上，類型隱含為 `SQL` 中的靜態類型 SQL。

Amazon Redshift 使用動態類型來處理無結構描述資料。當查詢評估資料時，`c_orders[0].o_orderstatus` 結果是特定類型。例如，對 `customer_orders_lineitem` 的第一則記錄評估 `c_orders[0].o_orderstatus` 可能會得到整數。對第二則記錄進行評估可能會得到字串。這些是運算式的動態類型。

當使用具有動態類型的點和括號表達式的 SQL 運算子或函數時，Amazon Redshift 會產生類似於使用標準 SQL 運算子或函數搭配個別靜態類型的結果。在此範例中，當路徑運算式的動態類型是字串時，與字串 'P' 的比較是有意義的。只要 `c_orders[0].o_orderstatus` 的動態類型是字串以外的任何其他資料類型，則等式傳回 `false`。當使用類型錯誤的引數時，其他函數傳回 `null`。

下列範例使用靜態類型撰寫先前的查詢：

```
SELECT c_custkey
FROM customer_orders_lineitem
WHERE CASE WHEN JSON_TYPEOF(c_orders[0].o_orderstatus) = 'string'
          THEN c_orders[0].o_orderstatus::VARCHAR = 'P'
          ELSE FALSE END;
```

請注意相等述詞和比較述詞之間的以下區別。在上一個範例中，如果您使用述詞 `less-than-or-equal` 取代等式述詞，語義會產生 `null`，而不是 `false`。

```
SELECT c_orders[0]. o_orderkey
FROM customer_orders_lineitem
WHERE c_orders[0].o_orderstatus <= 'P';
```

在此範例中，如果 `c_orders[0].o_orderstatus` 是字串，且按字母順序等於或小於 'P'，則 Amazon Redshift 傳回 `true`。如果它的字母順序大於 'P'，則 Amazon Redshift 傳回 `false`。但是，如果 `c_orders[0].o_orderstatus` 不是字串，Amazon Redshift 將傳回 `null`，因為 Amazon Redshift 無法比較不同類型的值，如以下查詢所示：

```
SELECT c_custkey
FROM customer_orders_lineitem
WHERE CASE WHEN JSON_TYPEOF(c_orders[0].o_orderstatus) = 'string'
```

```
THEN c_orders[0].o_orderstatus::VARCHAR <= 'P'
ELSE NULL END;
```

動態類型並不排除具有最低可比性的類型比較。例如，您可以將 CHAR 和 VARCHAR Amazon Redshift 純量類型轉換為 SUPER。它們與字串相當，包括忽略類似於 Amazon Redshift CHAR 和 VARCHAR 類型的尾隨空格字元。同樣地，整數、小數和浮點值與 SUPER 值相當。特別是對於小數欄，每個值也可以有不同的小數位數。Amazon Redshift 仍將它們視為動態類型。

Amazon Redshift 也支援評估為深度相等的物件和陣列的相等性，例如深入評估物件或陣列並比較所有屬性。請謹慎使用深度相等，因為執行深度相等的過程可能非常耗時。

使用動態類型進行聯結

對於聯結，動態輸入會自動比對具有不同動態類型的值，而不執行長時間 CASEWHEN 分析，以了解可能出現的資料類型。例如，假設您的組織隨著時間變更了其用於部分索引鍵的格式。

一開始發行的整數部分索引鍵更換為字串部分索引鍵，例如 'A55'，後來再次更換為陣列部分索引鍵，例如組合字串和數字的 ['X', 10]。Amazon Redshift 不需要對部分索引鍵執行冗長的案例分析，而且可以使用聯結，如下列範例所示。

```
SELECT c.c_name
      ,l.l_extendedprice
      ,l.l_discount
FROM customer_orders_lineitem c
      ,c.c_orders o
      ,o.o_lineitems l
      ,supplier_partsupp s
      ,s.s_partsupps ps
WHERE l.l_partkey = ps.ps_partkey
AND c.c_nationkey = s.s_nationkey
ORDER BY c.c_name;
```

下列範例顯示如果不使用動態類型，相同查詢可能會有多複雜且效率低落：

```
SELECT c.c_name
      ,l.l_extendedprice
      ,l.l_discount
FROM customer_orders_lineitem c
      ,c.c_orders o
      ,o.o_lineitems l
      ,supplier_partsupp s
      ,s.s_partsupps ps
```

```

WHERE CASE WHEN IS_INTEGER(l.l_partkey) AND IS_INTEGER(ps.ps_partkey)
  THEN l.l_partkey::integer = ps.ps_partkey::integer
  WHEN IS_VARCHAR(l.l_partkey) AND IS_VARCHAR(ps.ps_partkey)
  THEN l.l_partkey::varchar = ps.ps_partkey::varchar
  WHEN IS_ARRAY(l.l_partkey) AND IS_ARRAY(ps.ps_partkey)
    AND IS_VARCHAR(l.l_partkey[0]) AND IS_VARCHAR(ps.ps_partkey[0])
    AND IS_INTEGER(l.l_partkey[1]) AND IS_INTEGER(ps.ps_partkey[1])
  THEN l.l_partkey[0]::varchar = ps.ps_partkey[0]::varchar
    AND l.l_partkey[1]::integer = ps.ps_partkey[1]::integer
  ELSE FALSE END
AND c.c_nationkey = s.s_nationkey
ORDER BY c.c_name;

```

寬鬆的語義

根據預設，SUPER值的導覽操作會傳回 null，而不是在導覽無效時傳回錯誤。如果值SUPER不是物件，或者該SUPER值是物件，但不包含查詢中使用的屬性名稱，則物件導覽會無效。例如，下列查詢會存取SUPER資料欄 cdata 中的無效屬性名稱：

```
SELECT c.c_orders.something FROM customer_orders_lineitem c;
```

如果SUPER值不是陣列或陣列索引超出界限，則陣列導覽會傳回 Null。下列查詢會傳回 null，因為 c_orders[1][1] 超出範圍。

```
SELECT c.c_orders[1][1] FROM customer_orders_lineitem c;
```

使用動態輸入來投射SUPER值時，Lax 語意特別有用。如果轉換無效，則將SUPER值轉換為錯誤類型會傳回 null，而不是錯誤。例如，下列查詢傳回 null，因為它無法將物件屬性 o_orderstatus 的字串值 'Good' 轉換為 INTEGER。Amazon Redshift 會傳回VARCHAR要INTEGER投射的錯誤，但不會傳回投SUPER射的錯誤。

```
SELECT c.c_orders.o_orderstatus::integer FROM customer_orders_lineitem c;
```

自我檢查的種類

SUPER 資料欄支援檢查函數，這些函數會傳回 SUPER值的動態類型和其他類型資訊。最常見的範例是 JSON_TYPEOF 純量函數，根據SUPER值的動態類型，傳回VARCHAR具有布林值、數字、字串、物件、陣列或 null 值的。Amazon Redshift 支援SUPER資料欄的下列布林函數：

- DECIMAL_PRECISION

- DECIMAL_SCALE
- IS_ARRAY
- IS_BIGINT
- IS_CHAR
- IS_DECIMAL
- IS_FLOAT
- IS_INTEGER
- IS_OBJECT
- IS_SCALAR
- IS_SMALLINT
- IS_VARCHAR
- JSON_TYPEOF

如果輸入值為 null，所有這些函數傳回 false。IS_SCALAR、IS_ 和 IS_ARRAY 是互斥的 OBJECT，涵蓋除了 null 以外的所有可能值。

若要推斷與資料對應的類型，Amazon Redshift 會使用 JSON_TYPEOF 函數，傳回 SUPER 值的類型（最上層），如下列範例所示：

```
SELECT JSON_TYPEOF(r_nations) FROM region_nations;
 json_typeof
-----
array
(1 row)
```

```
SELECT JSON_TYPEOF(r_nations[0].n_nationkey) FROM region_nations;
 json_typeof
-----
number
```

Amazon Redshift 將此視為單一長字串，類似於將此值插入資料 VARCHAR 欄而非 SUPER。由於資料欄為 SUPER，因此單一字串仍然是有效 SUPER 值，差異會在 JSON_TYPEOF：

```
SELECT IS_VARCHAR(r_nations[0].n_name) FROM region_nations;
 is_varchar
-----
```

```
true
(1 row)
```

```
SELECT r_nations[4].n_name FROM region_nations
WHERE CASE WHEN IS_INTEGER(r_nations[4].n_nationkey)
            THEN r_nations[4].n_nationkey::INTEGER = 15
            ELSE false END;
```

排序依據

Amazon Redshift 不會定義具有不同動態類型的值之間的SUPER比較。字串SUPER的值不會小於或大於數字SUPER的值。若要搭配SUPER資料欄使用 ORDER BY 子句，Amazon Redshift 會定義在 Amazon Redshift 使用 ORDER BY 子句排序SUPER值時，要觀察的不同類型之間的總排序。動態類型之間的順序是布林、數字、字串、陣列、物件。以下範例顯示了不同類型的順序：

```
INSERT INTO region_nations VALUES
(100, 'name1', 'comment1', 'AWS'),
(200, 'name2', 'comment2', 1),
(300, 'name3', 'comment3', ARRAY(1, 'abc', null)),
(400, 'name4', 'comment4', -2.5),
(500, 'name5', 'comment5', 'Amazon');

SELECT r_nations FROM region_nations order by r_nations;

r_nations
-----
-2.5
1
"Amazon"
"AWS"
[1, "abc", null]
(5 rows)
```

如需 ORDER BY 子句的詳細資訊，請參閱 [ORDER BY 子句](#)。

運算子和函數

透過 Amazon Redshift，您可以使用運算子和函數，使用SUPER資料對大型資料集執行進階分析。SUPER 資料的運算子和函數是可對存放在 SQL Amazon Redshift 資料表中的半結構化資料進行複雜分析和操作的建構。

以下各節將介紹在 Amazon Redshift 中使用 SUPER 運算子和函數，以釋放半結構化資料的全部潛力的語法、範例和最佳實務。

算術運算子

SUPER 值支援使用動態輸入的所有基本算術運算子 +、-、*、/、%。操作的結果類型會保留為 SUPER。對於所有運算子，除了二元運算子 + 之外，輸入運算元必須是數字。否則，Amazon Redshift 傳回 null。當 Amazon Redshift 執行這些運算子且動態類型不會變更時，將保留小數和浮點值之間的差異。但是，當您使用乘法和除法時，小數位數會發生變化。算術溢位仍然會導致查詢錯誤，它們不會變更為 null。如果輸入是數字，則二元運算子 + 執行加法；如果輸入是字串，則執行串連。如果一個運算元是一個字串，而另一個運算元是一個數字，則結果為 null。如果 SUPER 值不是數字，則 Unary 字首運算子 + 和 - 傳回 null，如下列範例所示：

```
SELECT (c_orders[0]. o_orderkey + 0.5) * c_orders[0]. o_orderkey / 10 AS math FROM
customer_orders_lineitem;
      math
-----
1757958232200.1500
(1 row)
```

動態輸入可讓 中的十進位值 SUPER 具有不同的規模。Amazon Redshift 會將十進位值視為不同的靜態類型，並允許所有數學運算。Amazon Redshift 會根據運算元的比例，動態計算結果小數位數。如果其中一個運算元是浮點數，則 Amazon Redshift 會將另一個運算元提升為浮點數並產生浮點數結果。

算術函數

Amazon Redshift 支援資料 SUPER 欄的下列算術函數。如果輸入不是數字，以下項目會傳回 null：

- FLOOR。如需詳細資訊，請參閱 [FLOOR 函數](#)。
- CEIL 和 CEILING。如需詳細資訊，請參閱 [CEILING \(或 CEIL\) 函數](#)。
- ROUND。如需詳細資訊，請參閱 [ROUND 函數](#)。
- TRUNC。如需詳細資訊，請參閱 [TRUNC 函數](#)。
- ABS。如需詳細資訊，請參閱 [ABS 函數](#)。

下列範例會使用算術函數來查詢資料：

```
SELECT x, FLOOR(x), CEIL(x), ROUND(x)
FROM (
  SELECT (c_orders[0]. o_orderkey + 0.5) * c_orders[0].o_orderkey / 10 AS x
```



```
FROM customer_orders_lineitem
);
```

x	floor	ceil	round
1389636795898.0500	1389636795898	1389636795899	1389636795898

ABS 函數會保留輸入小數的規模，同時 FLOOR、CEIL。會 ROUND 消除輸入小數的規模。

陣列函數

Amazon Redshift 支援以下陣列組合和實用函數

array、array_concat、subarray、array_flatten、get_array_length 和 split_to_array。

您可以使用 ARRAY 函數從 Amazon Redshift 資料類型中的值建構 SUPER 陣列，包括其他 SUPER 值。

下列範例使用變數函數 ARRAY：

```
SELECT ARRAY(1, c.c_custkey, NULL, c.c_name, 'abc') FROM customer_orders_lineitem c;
          array
-----
[1,8401,null,""Customer#000008401"", ""abc""]
[1,9452,null,""Customer#000009452"", ""abc""]
[1,9451,null,""Customer#000009451"", ""abc""]
[1,8251,null,""Customer#000008251"", ""abc""]
[1,5851,null,""Customer#000005851"", ""abc""]
(5 rows)
```

下列範例使用陣列串連搭配 ARRAY_CONCAT 函數：

```
SELECT ARRAY_CONCAT(JSON_PARSE('[10001,10002]'),JSON_PARSE('[10003,10004]'));
          array_concat
-----
[10001,10002,10003,10004]
(1 row)
```

下列範例使用陣列操作搭配 函數，該 SUBARRAY 函數會傳回輸入陣列的子集。

```
SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);
          subarray
-----
```

```
["c","d","e"]
(1 row)
```

下列範例使用 ARRAY_ 將多個層級的陣列合併為單一陣列 FLATTEN :

```
SELECT x, ARRAY_FLATTEN(x) FROM (SELECT ARRAY(1, ARRAY(2, ARRAY(3, ARRAY())))) AS x);
```

```

      x          | array_flatten
-----+-----
 [1,[2,[3,[]]]] | [1,2,3]
(1 row)
```

陣列函數 ARRAY_CONCAT 和 ARRAY_FLATTEN 使用動態輸入規則。如果輸入不是陣列，它們會傳回 null 而不是錯誤。GET_ARRAY_LENGTH 函數會傳回指定物件或 SUPER 陣列路徑的陣列長度。

```

SELECT c_name
FROM customer_orders_lineitem
WHERE GET_ARRAY_LENGTH(c_orders) = (
    SELECT MAX(GET_ARRAY_LENGTH(c_orders))
    FROM customer_orders_lineitem
);
```

下列範例使用 SPLIT_TO_ 將字串分割為字串陣列 ARRAY。該函數使用分隔符號做為選用參數。如果沒有分隔符號，則預設為逗號。

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');
```

```

      split_to_array
-----
 ["12","345","6789"]
(1 row)
```

SUPER 組態

您可以針對特定案例設定 SUPER 資料。下列各節提供根據您的資料格式需求選擇和套用適當 SUPER 組態的詳細資訊。

主題

- [的 Lax 和嚴格模式 SUPER](#)
- [使用大寫和混合大小寫 JSON 欄位名稱或屬性存取欄位](#)

- [的剖析選項 SUPER](#)

的 Lax 和嚴格模式 SUPER

當您查詢SUPER資料時，路徑表達式可能不符合實際SUPER的資料結構。如果您嘗試存取物件或陣列元素的不存在成員，則如果您的查詢是在預設的通量模式下執行，Amazon Redshift 會傳回NULL值。如果以嚴格模式執行查詢，則 Amazon Redshift 會傳回錯誤訊息。下列工作階段參數可以設定為開啟或關閉寬鬆模式。

下列範例會使用工作階段參數來啟用寬鬆模式。

```
SET navigate_super_null_on_error=ON; --default lax mode for navigation

SET cast_super_null_on_error=ON; --default lax mode for casting

SET parse_super_null_on_error=OFF; --default strict mode for ingestion
```

使用大寫和混合大小寫JSON欄位名稱或屬性存取欄位

當您的JSON屬性名稱為大寫或混合大小寫時，您必須能夠以區分大小寫的方式導覽SUPER類型結構。若要這麼做，您可以設定 `enable_case_sensitive_identifier`，TRUE並以雙引號包裝大寫和混合大小寫屬性名稱。您也可以 `enable_case_sensitive_super_attribute` 將設定為TRUE。在這種情況下，您可以在查詢中使用大寫和混合大小寫的屬性名稱，而無需將它們用雙引號引起來。

下列範例說明如何設定 `enable_case_sensitive_identifier` 來查詢資料。

```
SET enable_case_sensitive_identifier to TRUE;

-- Accessing JSON attribute names with uppercase and mixedcase names
SELECT json_table.data."ITEMS"."Name",
       json_table.data."price"
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;

 Name | price
-----+-----
  "TV" | 345
(1 row)

RESET enable_case_sensitive_identifier;
```

```
-- After resetting the above configuration, the following query accessing JSON
attribute names with uppercase and mixedcase names should return null (if in lax
mode).
SELECT json_table.data."ITEMS"."Name",
       json_table.data."price"
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;
```

name	price
	345

(1 row)

下列範例說明如何設定 `enable_case_sensitive_super_attribute` 來查詢資料。

```
SET enable_case_sensitive_super_attribute to TRUE;
-- Accessing JSON attribute names with uppercase and mixedcase names

SELECT json_table.data.ITEMS.Name,
       json_table.data.price
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;
```

name	price
"TV"	345

(1 row)

```
RESET enable_case_sensitive_super_attribute;

-- After resetting enable_case_sensitive_super_attribute, the query now returns NULL
for ITEMS.Name (if in lax mode).

SELECT json_table.data.ITEMS.Name,
       json_table.data.price
FROM
  (SELECT json_parse('{"ITEMS":{"Name":"TV"}, "price": 345}') AS data) AS json_table;
```

name	price
	345

(1 row)

的剖析選項 SUPER

當您使用 `JSON_PARSE` 函數將JSON字串剖析為SUPER值時，會套用某些限制：

- 相同的屬性名稱不能出現在相同物件中，但可以出現在巢狀物件中。`json_parse_dedup_attributes` 組態選項允許 `JSON_PARSE` 僅保留上次重複屬性的出現，而不是傳回錯誤。
- 字串值不得超過 65535 位元組的系統最大 `varchar` 大小。`json_parse_truncate_strings` 組態選項允許 `JSON_PARSE ()` 自動截斷超過此限制的字串，而不會傳回錯誤。此行為只會影響字串值，而不會影響屬性名稱。

如需 `JSON_PARSE` 函數的詳細資訊，請參閱 [JSON_PARSE 函數](#)。

下列範例顯示如何將 `json_parse_dedup_attributes` 組態選項設定為傳回重複屬性錯誤的預設行為。

```
SET json_parse_dedup_attributes=OFF; --default behavior of returning error instead of de-duplicating attributes
```

下列範例顯示如何設定 `json_parse_truncate_strings` 組態選項，針對長度超過此限制的字串傳回錯誤的預設行為。

```
SET json_parse_truncate_strings=OFF; --default behavior of returning error instead of truncating strings
```

限制

使用 Amazon Redshift，您可以使用SUPER資料類型來儲存和查詢半結構化資料，例如 JSON、Avro 或 Ion。SUPER 資料類型限制是指在 Amazon Redshift 中使用此資料類型時的限制和界限。下列各節提供有關SUPER資料類型特定限制的詳細資訊，例如大小上限、巢狀層級，以及半結構化資料中支援的資料類型。

- 您無法將SUPER資料欄定義為分佈或排序索引鍵。
- 個別SUPER物件最多可存放 16 MB 的資料。
- SUPER 物件中的個別值限制為對應 Amazon Redshift 類型的長度上限。例如，載入至 `VARCHAR` 的單一字串值SUPER長度上限為 65535 位元組。

- 您無法在SUPER資料欄上執行部分更新或轉換操作。
- 您無法在右側聯結或完整外部聯結中使用SUPER資料類型及其別名。
- SUPER 資料類型不支援XML傳入或傳出序列化格式。
- 在參考資料表變數進行取消巢狀的子查詢（無論是否相關）FROM子查詢子句中，查詢只能參考其父資料表，不能參考其他資料表。
- 轉換限制

SUPER 值可以在其他資料類型之間轉換，但以下情況除外：

- Amazon Redshift 不區分 0 等級的整數和小數。
- 如果比例不為零，SUPER則資料類型的行為與其他 Amazon Redshift 資料類型相同，但 Amazon Redshift 會將 SUPER相關的錯誤轉換為 null，如下列範例所示。

```
SELECT 5::bool;
  bool
-----
  True
(1 row)

SELECT 5::decimal::bool;
ERROR:  cannot cast type numeric to boolean

SELECT 5::super::bool;
  bool
-----
  True
(1 row)

SELECT 5.0::bool;
ERROR:  cannot cast type numeric to boolean

SELECT 5.0::super::bool;
  bool
-----
(1 row)
```

- Amazon Redshift 不會將日期和時間類型轉換為SUPER資料類型。Amazon Redshift 只能從資料類型轉換日期和時間SUPER資料類型，如下列範例所示。

```
SELECT o.o_orderdate FROM customer_orders_lineitem c,c.c_orders o;
  order_date
```

```

-----
"2001-09-08"
(1 row)

SELECT JSON_TYPEOF(o.o_orderdate) FROM customer_orders_lineitem c,c.c_orders o;
json_typeof
-----
string
(1 row)

SELECT o.o_orderdate::date FROM customer_orders_lineitem c,c.c_orders o;
order_date
-----
2001-09-08
(1 row)

--date/time cannot be cast to super
SELECT '2019-09-09'::date::super;
ERROR:  cannot cast type date to super

```

- 從非純量值（物件和陣列）轉換為字串會傳回 NULL。若想正確序列化這些非純量值，請不要轉換它們。而是使用 `json_serialize` 轉換非純量值。`json_serialize` 函數傳回 `varchar`。一般而言，您不需要將非純量值轉換為 `varchar`，因為 Amazon Redshift 會隱含序列化，如下列第一個範例所示。

```

SELECT r_nations FROM region_nations WHERE r_regionkey=300;
r_nations
-----
[1,"abc",null]
(1 row)

SELECT r_nations::varchar FROM region_nations WHERE r_regionkey=300;
r_nations
-----
(1 row)

SELECT JSON_SERIALIZE(r_nations) FROM region_nations WHERE r_regionkey=300;
json_serialize
-----
[1,"abc",null]

```

(1 row)

- 對於不區分大小寫的資料庫，Amazon Redshift 不支援SUPER資料類型。對於不區分大小寫的資料欄，Amazon Redshift 不會將其轉換為 SUPER類型。因此，Amazon Redshift 不支援與觸發轉換的不區分大小寫SUPER資料欄互動的資料欄。
- Amazon Redshift 不支援揮發性函數，例如 RANDOM () 或 TIMEOFDAY ()，這些函數在未將外部資料表或 IN 函數左側 (LHS) 與此類子查詢巢狀的子查詢中。

SUPER 資料類型和具體化檢視

透過 Amazon Redshift，您可以使用SUPER資料類型來增強具體化視觀表的效能和靈活性。SUPER 資料類型可讓您從具體化視觀表中存放大量來自基礎資料表的資料欄，讓您直接查詢具體化視觀表，而無需加入基礎資料表。下列各節說明如何在 Amazon Redshift 中建立和使用具有SUPER資料類型的具體化檢視。

Amazon Redshift 擴展具體化視觀表的功能，以在具體化視觀表中使用SUPER資料類型和 PartiQL。SQL 和 PartiQL 查詢可以使用增量具體化檢視預先計算。如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

將無結構和半結構化資料儲存到 後SUPER，您可以使用 PartiQL 具體化檢視來介紹資料並將其切碎為具體化檢視。

加速 PartiQL 查詢

您可以使用具體化視觀表來加速在SUPER資料欄中導覽和/或非巢狀階層資料的 PartiQL 查詢。建立一或多個具體化檢視，將SUPER值分割為多個資料欄，並利用 Amazon Redshift 分析查詢的資料欄組織。因此，查詢會使用具體化視觀表。

具體化視觀表本質上是提取和標準化巢狀資料。標準化的程度取決於您在將SUPER資料轉換為傳統資料欄資料的努力程度。

使用具體化視觀表將資料分割為SUPER資料欄

透過 Amazon Redshift，您可以使用具體化檢視將資料分割為SUPER資料欄，藉此改善查詢效能並減少儲存需求。碎紙是指將複雜資料類型分解為半結構化JSON或XML更小、更扁平的資料欄的程序。SUPER 資料欄是一種特殊形式的資料欄儲存，已針對快速掃描碎片資料進行最佳化。

下列各節說明使用 Amazon Redshift 中的具體化檢視將資料分割為SUPER資料欄的步驟和考量。

下列範例顯示具體化檢視，將巢狀資料切碎，產生的資料欄仍然是SUPER資料類型。


```
SELECT c.c_name, o.o_orderstatus
FROM customer_orders_lineitem c, c.c_orders o;
```

下列範例顯示一個具體化視觀表，該視觀表會根據分解的資料建立傳統的 Amazon Redshift 純量欄。

```
SELECT c.c_name, c.c_orders[0].o_totalprice
FROM customer_orders_lineitem c;
```

您可以建立單一具體化視觀表 `super_mv` 來加速這兩項查詢。

要回答第一個查詢，您必須具體化屬性 `o_orderstatus`。您可以省略 `c_name` 屬性，因為它不涉及巢狀導覽或解除巢狀化。您也必須在具體化視觀表中包含 `customer_orders_lineitem` 的屬性 `c_custkey`，以便能夠將基底資料表聯結至具體化視觀表。

要回答第二個查詢，您還必須具體化屬性 `o_totalprice` 和 `c_orders` 的陣列索引 `o_idx`。因此，您可以存取 `c_orders` 的索引 0。

```
CREATE MATERIALIZED VIEW super_mv distkey(c_custkey) sortkey(c_custkey) AS (
  SELECT c_custkey, o.o_orderstatus, o.o_totalprice, o_idx
  FROM customer_orders_lineitem c, c.c_orders o AT o_idx
);
```

具體化視觀表 `super_mv` 的 `o_orderstatus` 和 `o_totalprice` 屬性為 SUPER。

具體化視觀表 `super_mv` 會在基底資料表 `customer_orders_lineitem` 發生變更時，累加式重新整理。

```
REFRESH MATERIALIZED VIEW super_mv;
INFO: Materialized view super_mv was incrementally updated successfully.
```

若要將第一個 PartiQL 查詢重寫為一般 SQL 查詢，請加入 `customer_orders_lineitem` 與 `super_mv`，如下所示。

```
SELECT c.c_name, v.o_orderstatus
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey;
```

同樣地，您可以重新撰寫第二個 PartiQL 查詢。下列範例會在 `o_idx = 0` 上使用篩選器。

```
SELECT c.c_name, v.o_totalprice
FROM customer_orders_lineitem c
```

```
JOIN super_mv v ON c.c_custkey = v.c_custkey
WHERE v.o_idx = 0;
```

在 CREATE MATERIALIZED VIEW 命令中，將 c_custkey 指定為 super_mv 的分發金鑰和排序金鑰。Amazon Redshift 執行高效率的合併聯結，假設 c_custkey 也是 customer_orders_lineitem 的分散索引鍵和排序索引鍵。如果不是這種情況，您可以指定 c_custkey 做為 customer_orders_lineitem 的排序索引鍵和分散索引鍵，如下所示。

```
ALTER TABLE customer_orders_lineitem
ALTER DISTKEY c_custkey, ALTER SORTKEY (c_custkey);
```

使用 EXPLAIN 陳述式來驗證 Amazon Redshift 是否在重寫查詢上執行合併聯結。

```
EXPLAIN
  SELECT c.c_name, v.o_orderstatus
  FROM customer_orders_lineitem c JOIN super_mv v ON c.c_custkey = v.c_custkey;

QUERY PLAN

-----
  XN Merge Join DS_DIST_NONE (cost=0.00..34701.82 rows=1470776 width=27)
  Merge Cond: ("outer".c_custkey = "inner".c_custkey)
   -> XN Seq Scan on mv_tbl__super_mv__0 derived_table2 (cost=0.00..14999.86
rows=1499986 width=13)
   -> XN Seq Scan on customer_orders_lineitem c (cost=0.00..999.96 rows=99996
width=30)
(4 rows)
```

從分解的資料建立 Amazon Redshift 純量欄

儲存在 中的無結構描述資料 SUPER 可能會影響 Amazon Redshift 的效能。例如，篩選述詞或聯結條件做為範圍限制掃描可能無法有效地使用區域地圖。使用者和 BI 工具可以使用具體化視觀表做為資料的常規表示形式，並提高分析查詢的效能。

下列查詢會掃描具體化視觀表 super_mv 和篩選 o_orderstatus。

```
SELECT c.c_name, v.o_totalprice
FROM customer_orders_lineitem c
JOIN super_mv v ON c.c_custkey = v.c_custkey
WHERE v.o_orderstatus = 'F';
```

檢查 `stl_scan` 以確認 Amazon Redshift 無法在 `o_orderstatus` 範圍限制掃描上有效地使用區域地圖。

```
SELECT slice, is_rrscan FROM stl_scan
WHERE query = pg_last_query_id() AND perm_table_name LIKE '%super_mv%';
```

```
slice | is_rrscan
-----+-----
      0 | f
      1 | f
      5 | f
      4 | f
      2 | f
      3 | f
(6 rows)
```

下列範例會調整具體化視觀表 `super_mv`，從分解的資料建立純量欄。在此情況下，Amazon Redshift 會從轉換為 `o_orderstatus SUPER VARCHAR`。此外，請指定 `o_orderstatus` 做為 `super_mv` 的排序索引鍵。

```
CREATE MATERIALIZED VIEW super_mv distkey(c_custkey) sortkey(c_custkey, o_orderstatus)
AS (
  SELECT c_custkey, o.o_orderstatus::VARCHAR AS o_orderstatus, o.o_totalprice, o_idx
  FROM customer_orders_lineitem c, c.c_orders o AT o_idx
);
```

重新執行查詢之後，請確認 Amazon Redshift 現在可以使用區域地圖。

```
SELECT v.o_totalprice
FROM super_mv v
WHERE v.o_orderstatus = 'F';
```

您可以驗證範圍限制掃描現在使用區域地圖，如下所示。

```
SELECT slice, is_rrscan FROM stl_scan
WHERE query = pg_last_query_id() AND perm_table_name LIKE '%super_mv%';
```

```
slice | is_rrscan
-----+-----
      0 | t
      1 | t
```

```
2 | t
3 | t
4 | t
5 | t
(6 rows)
```

搭配具體化檢視使用SUPER資料類型的限制

使用 Amazon Redshift，您可以使用SUPER資料類型建立具體化檢視，以預先計算複雜的查詢並改善查詢效能。SUPER 資料類型可讓您將查詢的結果集儲存為平面資料結構，以加快存取和處理速度。

下列各節提供有關在 Amazon Redshift 中使用SUPER資料類型與具體化檢視的限制和最佳實務的詳細資訊。

Amazon Redshift 中的具體化視觀表在 PartiQL 或 方面沒有任何特定限制SUPER。

如需建立具體化視觀表時一般SQL限制的相關資訊，請參閱 [限制](#)。

如需具體化檢視增量重新整理的一般SQL限制資訊，請參閱 [增量重新整理的限制](#)。

機器學習

Amazon Redshift 機器學習 (Amazon Redshift ML) 是一種強大的雲端型服務，可讓各種技能等級的分析師和資料科學家更輕鬆地使用機器學習技術。Amazon Redshift ML 使用模型產生結果。您可以透過下列方式使用模型：

- 您可以將要訓練模型的資料，以及與資料輸入相關聯的中繼資料提供給 Amazon Redshift。然後 Amazon Redshift ML 在 Amazon 中建立模型 SageMaker，擷取輸入資料中的模式。透過使用自己的模型資料，您可以使用 Amazon Redshift ML 來識別資料中的趨勢，例如流失預測、客戶生命週期值或營收預測。您可以使用這些模型來產生新輸入資料的預測，而不會產生額外的成本。
- 您可以使用 Amazon Bedrock 提供的其中一個基礎模型 (FM)，例如 Claude 或 Amazon Titan。使用 Amazon Bedrock，您只需幾個步驟，即可將大型語言模型 (LLMs) 的強大功能與 Amazon Redshift 中的分析資料結合。透過使用外部大型語言模型 (LLM)，您可以使用 Amazon Redshift 對資料執行自然語言處理 (NLP)。您可以使用 NLP 來產生文字、情緒分析或翻譯等應用程式。如需將 Amazon Bedrock 與 Amazon Redshift 搭配使用的相關資訊，請參閱 [Amazon Redshift ML 與 Amazon Bedrock 整合](#)。

Note

選擇不使用您的資料進行服務改善

如果您使用的是 Amazon Bedrock 模型，而且不想為了改善服務而 AWS 處理資料，則必須啟用 Amazon Bedrock 的選擇退出政策。

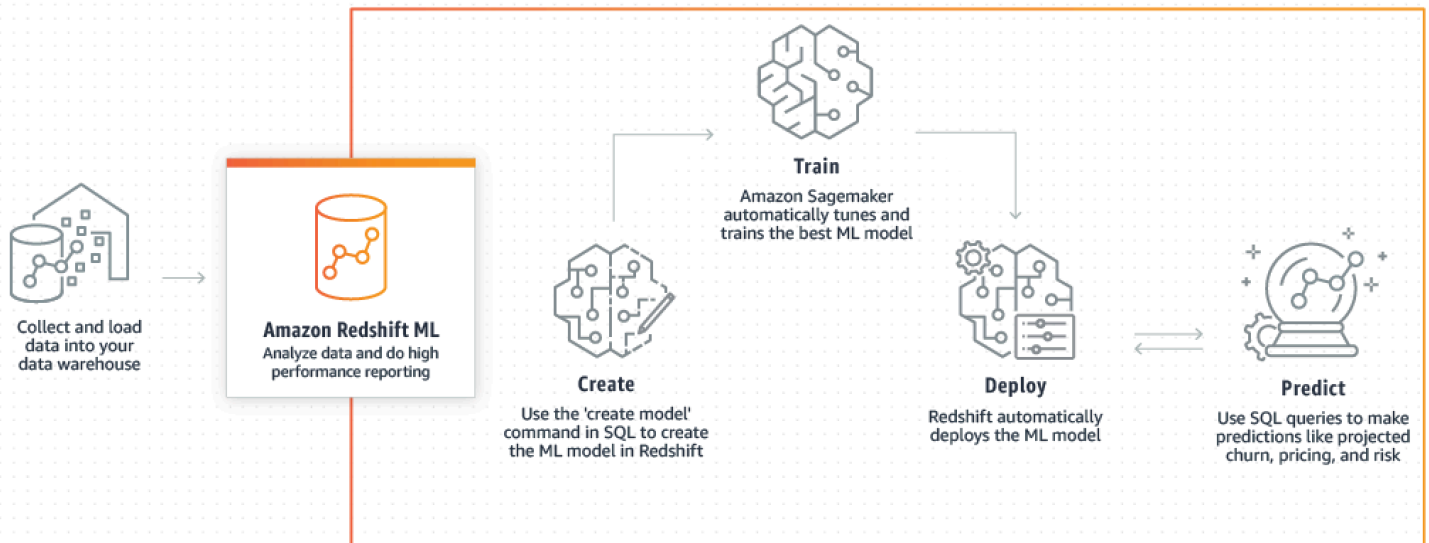
Note

LLMs 會產生不準確或不完整的資訊。我們建議驗證 LLMs 產生的資訊，以確保其準確且完整。

Amazon Redshift ML 如何與 Amazon 搭配使用 SageMaker

Amazon Redshift 與 Amazon SageMaker Autopilot 搭配使用，以自動取得最佳模型，並在 Amazon Redshift 中提供預測函數。

下圖說明 Amazon Redshift ML 的運作方式。



一般工作流程如下：

1. Amazon Redshift會將訓練資料匯出至 Amazon S3。
2. Amazon SageMaker Autopilot 會預先處理訓練資料。預處理會執行重要功能，例如輸入遺漏值。其會識別特定資料欄是可分類的 (例如郵遞區號)，正確格式化這些資料欄以進行訓練，並執行許多其他工作。選擇要套用至訓練資料集的最佳前置處理器本身是問題，而 Amazon SageMaker Autopilot 會自動化其解決方案。
3. Amazon SageMaker Autopilot 會尋找演算法和演算法超參數，以最準確的預測交付模型。
4. Amazon Redshift 會將預測函數註冊為 Amazon Redshift 叢集中的SQL函數。
5. 當您執行CREATEMODEL陳述式時，Amazon Redshift 會使用 Amazon SageMaker 進行訓練。因此，訓練模型會產生相關的成本。這是 AWS 帳單 SageMaker 中 Amazon 的個別明細項目。您也需要支付 Amazon S3 用於存放訓練資料的儲存費用。使用 建立CREATEMODEL的模型進行推論，您可以在 Redshift 叢集上編譯和執行，無需付費。使用 Amazon Redshift ML 不會收取額外的 Amazon Redshift 費用。

主題

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [使用 Amazon Redshift ML 的成本](#)
- [開始使用 Amazon RedShift ML](#)
- [Amazon Redshift ML 的教學課程](#)

- [Amazon Redshift ML 與 Amazon Bedrock 整合](#)

機器學習概述

透過 Amazon Redshift，您可以利用機器學習功能，從資料中取得寶貴的洞見。此 Machine Learning (ML) 概觀會示範如何探索、視覺化和準備資料，以進行 ML 模型訓練和部署。下列各節將引導您完成使用 Amazon Redshift ML 透過機器學習釋放資料潛力的程序。

透過使用 Amazon Redshift ML，您可以使用 SQL 陳述式訓練機器學習模型，並在 SQL 查詢中叫用它們以進行預測。

若要協助您學習如何使用 Amazon Redshift ML，您可以觀看下列影片：[Amazon Redshift ML](#)。

如需設定 Redshift 叢集或無伺服器工作群組的先決條件、使用 Amazon Redshift ML 的許可和擁有權的相關資訊，請參閱下列各節。這些章節也會說明簡單訓練和預測在 Amazon Redshift ML 中的運作方式。

機器學習如何解決問題

機器學習模型會在訓練資料中尋找模式，然後將這些模式套用至新資料，以產生預測。在機器學習中，您可以透過學習最能解釋您資料的模式來訓練這些模型。然後，您可以使用模型對新資料進行預測 (也稱為推論)。機器學習通常是一個反覆的過程，您可以透過變更參數和改進訓練資料來繼續提高預測的準確性。如果資料變更，則會使用新資料集重新訓練新模型。

為了解決各種業務目標，有不同的基本機器學習方法可用。

在 Amazon Redshift ML 中進行監督式學習

Amazon Redshift 支援受監督式學習，這是進階企業分析最常用的方法。當您擁有一組已建立的資料並了解特定輸入資料如何預測各種業務成果時，監督式學習是首選的機器學習方法。這些結果有時稱為標籤。特別是，如果您的資料集是一個包含特徵 (輸入) 和目標 (輸出) 的屬性資料表。例如，假設您有一個提供過去和現在客戶的年齡和郵遞區號的資料表。假設您還有一個「作用中」欄位，而此欄位對於現在的客戶而言是 true，對於已暫停其會員資格的客戶來說是 false。監督式機器學習的目標是發現導致客戶流失的年齡和郵遞區號模式，以其目標為“False”的客戶表示。您可以使用此模型來預測可能流失的客戶，例如暫停其會員資格，並可能提供保留獎勵。

Amazon Redshift 支援監督式學習，其中包括迴歸、二進制分類和多類別分類。迴歸是指預測連續值的問題，例如客戶的總支出。二進制分類指的是預測兩個結果之一的問題，例如預測客戶是否會流失。多類別分類是指預測許多結果之一的問題，例如預測客戶可能感興趣的項目。資料分析師和資料科學家可

以使用它來執行監督式學習，以解決諸如預測、個性化或客戶流失預測等問題。您還可以在諸如預測哪些銷售會結束、收入預測、詐騙偵測和客戶終身價值預測等問題中使用監督式學習。

在 Amazon Redshift ML 中進行非監督式學習

非監督式學習會使用機器學習演算法來分析和分組未標籤的訓練資料。演算法會發現隱藏的模式或分組。目標是為資料中的基礎結構或分佈建立模型，以進一步了解資料。

Amazon Redshift 支援 K 平均值叢集演算法，以解決非監督式學習問題。此演算法可解決您想在資料中探索分組的叢集問題。K 平均值演算法會嘗試尋找資料中的離散群組。未分類的資料會根據其相似性和差異進行分組和分割。透過分組，K 平均值演算法會反覆決定最佳質心，並將每個成員指派給最接近的質心。最接近屬於同一群組的相同質心的成員。群組的成員會盡可能地與相同群組中的其他成員相似，並盡可能地和其他群組的成員不同。例如，K 平均值叢集演算法可用於對受到流行病影響的城市進行分類，或根據消費品產品的受歡迎程度對城市進行分類。

使用 K 平均值演算法時，您可以指定輸入 k，指定要在資料中尋找的叢集數目。該演算法的輸出是 k 個質心的集合。每個資料點都屬於 k 個叢集中最接近它的其中一個叢集。每個叢集都會由其質心描述。質心可以被認為是叢集的多維平均值。K 平均值演算法會比較距離，以了解叢集彼此之間有多不同。較大的距離通常表示叢集之間的差異較大。

預處理資料對 K 平均值來說很重要，因為可確保模型的特徵保持相同的比例並產生可靠的結果。Amazon Redshift 支援 CREATE MODEL 陳述式的某些 K 平均值前置處理器，例如 StandardScaler、MinMax 和 NumericPassthrough。如果您不想套用任何 K 平均值的預處理，請 NumericPassthrough 明確選擇 作為轉換器。如需 K 平均值的相關資訊，請參閱 [CREATE MODEL 使用 K 參數 MEANS](#)。

若要協助您了解如何使用 K 平均值叢集執行非監督式訓練，您可以觀看以下影片：[使用 K 平均值叢集進行非監督式訓練](#)。

Amazon Redshift ML 的術語和概念

下列術語用來描述 Amazon Redshift ML 的一些概念。

- Amazon Redshift 中的機器學習會使用單一 SQL 命令訓練模型。Amazon Redshift ML 和 Amazon SageMaker 管理所有資料轉換、許可、資源使用和探索適當的模型。
- 訓練是 Amazon Redshift 透過在模型中執行指定的資料子集來建立機器學習模型的階段。Amazon Redshift 會自動在 Amazon 中啟動訓練任務，SageMaker 並產生模型。
- 預測（也稱為推論）是在 Amazon Redshift SQL 查詢中使用模型來預測結果。在推論時，Amazon Redshift 會使用以模型為基礎的預測函數做為較大查詢的一部分來產生預測。預測是在 Redshift 叢集本機上計算的，因此可提供高輸送量、低延遲和零額外成本。

- 透過自帶模型 (BYOM) ，您可以使用在 Amazon Redshift 之外與 Amazon 一起訓練的模型 SageMaker ，在 Amazon Redshift 本機進行資料庫內推論。Amazon Redshift ML 支援在本機推論 BYOM 中使用。
- 當模型在 Amazon 中預先訓練 SageMaker、由 Amazon SageMaker Neo 編譯，以及在 Amazon Redshift ML 中本地化時，會使用本機推論。若要將支援本機推論的模型匯入 Amazon Redshift，請使用 CREATEMODEL 命令。Amazon Redshift 會呼叫 Amazon SageMaker Neo 匯入預先訓練的 SageMaker 模型。您可以在那裡編譯模型，然後將編譯後的模型匯入 Amazon Redshift。使用本機推論以加快速度並降低成本。
- 當 Amazon Redshift 調用部署在 中的模型端點時，會使用遠端推論 SageMaker。遠端推論可讓您靈活地叫用所有類型的自訂模型和深度學習模型，例如您在 Amazon 中建置和部署的 TensorFlow 模型 SageMaker。

同樣重要的是以下各項：

- Amazon SageMaker 是完全受管的機器學習服務。透過 Amazon SageMaker，資料科學家和開發人員可以輕鬆建置、訓練模型，並直接將模型部署到生產就緒的託管環境中。如需 Amazon 的相關資訊 SageMaker，請參閱 [Amazon 開發人員指南 中的什麼是 SageMaker](#) Amazon SageMaker。
- Amazon SageMaker Autopilot 是功能集，可根據您的資料自動訓練和調整分類或迴歸的最佳機器學習模型。您可以保持完整的控制權和可見性。Amazon SageMaker Autopilot 支援表格格式的輸入資料。Amazon SageMaker Autopilot 為線性迴歸、二進位分類和多類別分類提供自動資料清理和預處理、自動演算法選擇。它也支援自動超參數最佳化 (HPO)、分散式訓練、自動執行個體和叢集大小選擇。如需 Amazon SageMaker Autopilot 的相關資訊，請參閱 Amazon SageMaker 開發人員指南 中的使用 Amazon [SageMaker Autopilot 自動化模型開發](#)。
- Amazon Bedrock 是一項完全受管的服務，提供從 AI21 Labs、Anthropic、Cohere、Meta、Mistral AI、Stability AI 和 Amazon 等頂尖 AI 公司透過單一 提供的高效能基礎模型 (FMs) API，以及建置生成性 AI 應用程式所需的廣泛功能。

適用於新手和專家的機器學習

使用 Amazon Redshift，無論您是新手還是 ML 專家，都可以利用 Machine Learning (ML) 功能從資料中取得洞見。Machine Learning 是一種 Amazon Redshift 功能，可讓您使用 SQL 命令建立、訓練和部署 ML 模型，而不需要廣泛的 ML 專業知識或複雜的資料工程。

下列各節會引導您完成使用 Machine Learning 的程序，讓您透過 Amazon Redshift 釋放資料的完整潛力。

Amazon Redshift ML 可讓您使用單一SQLCREATEMODEL命令來訓練模型。此CREATEMODEL命令會建立模型，讓 Amazon Redshift SQL 使用熟悉的建構產生模型型預測。

當您在機器學習、工具、語言、演算法和 方面沒有專業知識時，Amazon Redshift ML 特別有用APIs。使用 Amazon Redshift ML 時，您不必執行與外部機器學習服務整合所需的無差別繁重工作。Amazon Redshift 可為您節省格式化和移動資料、管理許可控制或建置自訂整合、工作流程和指令碼的時間。您可以輕鬆使用熱門的機器學習演算法，並簡化從訓練到預測期間頻繁迭代的訓練需求。Amazon Redshift 會自動探索最佳演算法，並針對您的問題調整最佳模型。您可以從 Amazon Redshift 叢集中進行預測，而不需要將資料移出 Amazon Redshift，也不需要與其他服務互動並支付費用。

Amazon Redshift ML 可支援資料分析師和資料科學家使用機器學習。機器學習專家也可以使用其知識來引導CREATEMODEL陳述式只使用其指定的層面。透過這樣做，您可以加快尋找最佳候選者CREATEMODEL所需的時間、提高模型的準確性，或兩者兼有。

此CREATEMODEL陳述式可讓您靈活地指定訓練任務的參數。使用此彈性，機器學習新手或專家就可以選擇自己喜歡的預處理器、演算法、問題類型和超參數。例如，對客戶流失感興趣的使用者可能會針對問題類型為二進位分類的CREATEMODEL陳述式指定，該分類非常適合客戶流失。然後，CREATEMODEL陳述式會縮小其對最佳模型的搜尋範圍，使其成為二進位分類模型。即使使用者選擇問題類型，仍可使用CREATEMODEL陳述式的許多選項。例如，會CREATEMODEL探索並套用最佳的預先處理轉換，並探索最佳的超參數設定。

Amazon Redshift ML 使用 Amazon SageMaker Autopilot 自動尋找最佳模型，讓訓練更輕鬆。在幕後，Amazon SageMaker Autopilot 會根據您提供的資料，自動訓練和調整最佳的機器學習模型。然後，Amazon SageMaker Neo 會編譯訓練模型，並在 Redshift 叢集中提供預測。當您使用訓練的模型執行機器學習推論查詢時，查詢可以使用 Amazon Redshift 的大量平行處理功能。同時，查詢可以使用以機器學習為基礎的預測。

- 作為機器學習入門者，具有機器學習不同層面的一般知識，例如前置處理器、演算法和超參數，請僅針對您指定的層面使用 CREATEMODEL陳述式。然後，您可以縮短尋找最佳候選者或提高模型準確性CREATEMODEL所需的時間。此外，您可以透過引入其他領域知識 (例如問題類型或目標)，來增加預測的商業價值。例如，在客戶流失案例中，如果「客戶不活躍」的結果很少見，則 F1 目標通常高於準確度目標。因為高準確度模型可能會一直預測到「客戶很活躍」，此結果雖然具有高準確度，但商業價值很小。如需 F1 目標的相關資訊，請參閱 Amazon SageMaker API 參考 中的[utoMLJob目標](#)。

如需CREATEMODEL陳述式基本選項的詳細資訊，請參閱 [簡單 CREATE MODEL](#)。

- 若身為機器學習進階從業人員，您可以為某些 (但不是全部) 特徵指定問題類型和預處理器。然後 CREATEMODEL遵循您在指定方面的建議。同時，CREATEMODEL仍會探索其他功能的最佳

前置處理器和最佳超參數。如需如何限制訓練管道的一或多個層面的相關資訊，請參閱 [CREATE MODEL 使用使用者指南](#)。

- 若是身為機器學習專家，您可以完全掌控訓練和超參數調整。然後，CREATEMODEL陳述式不會嘗試探索最佳的前置處理器、演算法和超參數，因為您做出了所有選擇。如需如何CREATEMODEL搭配 AUTO 使用的詳細資訊OFF，請參閱 [CREATE XGBoost 具有的模型 AUTO OFF](#)。
- 身為資料工程師，您可以在 Amazon 中引進預先訓練的XGBoost模型， SageMaker 並將其匯入 Amazon Redshift 以進行本機推論。透過自備模型（BYOM），您可以使用在 Amazon Redshift 之外與 Amazon SageMaker 一起訓練的模型，在 Amazon Redshift 本機進行資料庫內推論。Amazon Redshift ML 支援在本機或遠端推論BYOM中使用。

如需如何使用CREATEMODEL陳述式進行本機或遠端推論的詳細資訊，請參閱 [自備模型 \(BYOM\) - 本機推論](#)。

身為 Amazon Redshift ML 使用者，您可以選擇下列任何選項來訓練和部署模型：

- 問題類型，請參閱 [CREATE MODEL 使用使用者指南](#)。
- 目標，請參閱 [CREATE MODEL 使用使用者指南](#) 或 [CREATE XGBoost 具有的模型 AUTO OFF](#)。
- 模型類型，請參閱 [CREATE XGBoost 具有的模型 AUTO OFF](#)。
- 預處理器，請參閱 [CREATE MODEL 使用使用者指南](#)。
- 超參數，請參閱 [CREATE XGBoost 具有的模型 AUTO OFF](#)。
- 攜帶您自己的模型（BYOM），請參閱 [自備模型 \(BYOM\) - 本機推論](#)。

使用 Amazon Redshift ML 的成本

透過 Amazon Redshift，您可以利用機器學習功能從資料中取得洞見，而不需要大量的資料工程或機器學習專業知識。下列各節說明使用 Amazon Redshift ML 的相關成本，協助您規劃和最佳化費用，同時運用此強大的機器學習整合。

搭配使用 Amazon Redshift ML 的成本 SageMaker

適用於的 Amazon Redshift ML SageMaker 使用您現有的叢集資源進行預測，因此您可以避免額外的 Amazon Redshift 費用。建立或使用模型不會產生額外的 Amazon Redshift 費用。預測發生在 Redshift 叢集本機，因此除非您需要調整叢集大小，否則您無需支付額外費用。Amazon Redshift ML 使用 Amazon SageMaker 來訓練您的模型，這確實會產生額外的相關費用。

Amazon Redshift 叢集中執行的預測函數不需額外費用。CREATE MODEL 陳述式使用 Amazon SageMaker 並產生額外費用。成本會隨著訓練資料中的儲存格數量而增加。儲存格數量是記錄數目 (訓練查詢或資料表中) 乘以欄數的乘積。例如，當 CREATEMODEL陳述式的SELECT查詢建立 10,000 筆記錄和 5 欄時，其建立的儲存格數量為 50,000。

在某些情況下，SELECT查詢產生的訓練資料CREATEMODEL超過您提供的 MAX_CELLS 限制 (如果您未提供限制，則預設為 100 萬)。在這些情況下，會CREATEMODEL隨機選擇大約 MAX_CELLS (也就是訓練資料集中的「資料欄數目」記錄)。CREATE MODEL 然後使用這些隨機選擇的組合來執行訓練。隨機抽樣可確保減少的訓練資料集沒有任何偏差。因此，透過設定 MAX_CELLS，您可以控制訓練成本。

使用CREATEMODEL陳述式時，您可以使用 MAX_CELLS 和 MAX_RUNTIME 選項來控制成本、時間和潛在的模型準確性。

MAX_RUNTIME 指定使用 AUTO ON 或 OFF 選項 SageMaker 時，訓練可花費的時間上限。訓練任務通常比 MAX_ 更早完成RUNTIME，具體取決於資料集的大小。模型訓練完成後，Amazon Redshift 會在背景執行其他工作，以便在叢集中編譯和安裝模型。因此，CREATEMODEL可能需要超過 MAX_RUNTIME 才能完成。不過，MAX_RUNTIME 會限制 SageMaker 中用於訓練模型的運算量和時間。您可以使用 SHOW 隨時檢查模型的狀態MODEL。

當您CREATEMODEL以 AUTO ON 執行時，Amazon Redshift ML 會使用 SageMaker Autopilot 自動且智慧地探索不同的模型 (或候選項目) 來尋找最佳模型。MAX_RUNTIME 會限制花費的時間和運算。如果 MAX_RUNTIME 設定過低，即使只有一個候選者也可能沒有足夠的時間探索。如果您看到「Autopilot candidate has no model」錯誤，CREATEMODEL請以較大的 MAX_RUNTIME 值重新執行。如需此參數的詳細資訊，請參閱 Amazon 參考 [MaxAutoMLJobRuntimeInSeconds](#) 中的 SageMaker API

當您CREATEMODEL使用 AUTO 執行時OFF，MAX_RUNTIME 會對應至訓練任務在中執行的時間限制 SageMaker。訓練任務通常會更快完成，取決於資料集的大小和使用的其他參數，例如 MODEL_TYPE 中的 num_roundsXGBOOST。

您也可以執行時指定較小的 MAX_CELLS CREATE 值來控制成本或縮短訓練時間MODEL。儲存格是資料庫中的項目。每一列會對應至與資料欄數量一樣的儲存格，可以是固定寬度或不同寬度。MAX_CELLS 會限制儲存格數量，因此會限制用於訓練模型的訓練範例數量。根據預設，MAX_CELLS 設定為 100 萬個儲存格。減少 MAX_CELLS 會減少CREATEMODEL該 Amazon Redshift 匯出中SELECT查詢結果的資料列數目，並傳送至 SageMaker 以訓練模型。因此，減少 MAX_CELLS 會減少用於使用 AUTO ON 和 AUTO 訓練模型的資料集大小OFF。此方法有助於減少訓練模型的成本和時間。若要查看特定訓練任務的訓練和計費時間相關資訊，請選擇 Amazon 中的訓練任務 SageMaker。

增加 MAX_RUNTIME 和 MAX_CELLS 通常會透過允許探索更多候選項目 SageMaker 來改善模型品質。如此一來，SageMaker 可能需要更多時間來訓練每個候選者，並使用更多資料來訓練更好的模型。如果您想要更快速的迭代或探索資料集，請使用較低的 MAX_RUNTIME 和 MAX_CELLS。如果您想要提高模型的準確性，請使用更高的 MAX_RUNTIME 和 MAX_CELLS。

如需各種儲存格數目的成本和免費試用的相關資訊，請參閱 [Amazon Redshift 定價](#)。

搭配 Amazon Bedrock 使用 Amazon Redshift ML 的成本

搭配使用 Amazon Redshift ML 與 Amazon Bedrock 會產生額外費用。如需詳細資訊，請參閱 [Amazon Bedrock 定價](#)。

開始使用 Amazon RedShift ML

Amazon Redshift ML 讓使用者使用熟悉的 SQL 命令輕鬆 SQL 建立、訓練和部署機器學習模型。使用 Amazon Redshift ML，您可以使用 Redshift 叢集中的資料，透過 Amazon 訓練模型 SageMaker。之後，這些模型會經過本地化，並且可以在 Amazon Redshift 資料庫中進行預測。Amazon Redshift ML 目前支援機器學習演算法：XGBoost (AUTO ON 和 OFF) 和多層感知器 (AUTO ON)、K 平均值 (AUTO OFF) 和線性學習器。

主題

- [針對 Amazon Redshift ML 管理進行叢集和組態設定](#)
- [使用模型可解釋性與 Amazon Redshift ML](#)
- [Amazon Redshift ML 概率指標](#)

針對 Amazon Redshift ML 管理進行叢集和組態設定

在您使用 Amazon Redshift ML 之前，請先完成叢集設定並設定使用 Amazon Redshift ML 的許可。

使用 Amazon Redshift ML 的叢集設定

使用 Amazon Redshift ML 之前，請先完成以下先決條件。

身為 Amazon Redshift 管理員，請針對使用 Amazon Redshift 佈建叢集執行下列一次性設定。如需將 Amazon Redshift ML 與 Amazon Redshift Serverless 搭配使用，請參閱 [Amazon Redshift Serverless 入門](#)。

若要為 Amazon Redshift ML 執行一次性叢集設定

1. 使用 AWS Management Console 或 AWS Command Line Interface () 建立 Redshift 叢集AWS CLI。請務必在建立叢集時連接 AWS Identity and Access Management (IAM) 政策。如需將 Amazon Redshift ML 與 Amazon 搭配使用所需的許可的詳細資訊 SageMaker，請參閱[使用 Amazon Redshift 機器學習 \(ML \) 所需的許可](#)
2. 以下列其中一種方式建立使用 Amazon Redshift ML 所需的IAM角色：
 - 若要 SageMaker 搭配 Amazon Redshift ML 使用，請使用 AmazonS3FullAccess和 AmazonSageMakerFullAccess政策建立IAM角色。如果您打算同時建立預測模型，也可以將 AmazonForecastFullAccess 政策附加至您的角色。
 - 若要將 Amazon Bedrock 與 Amazon Redshift ML 搭配使用，請使用 AmazonS3FullAccess和 AmazonBedrockFullAccess政策建立IAM角色。
 - 建議您透過 Amazon Redshift 主控台建立IAM角色，該主控台具有執行SQL命令的許可AmazonRedshiftAllCommandsFullAccess政策，例如 CREATE MODEL。Amazon Redshift 使用無縫 API型機制，AWS 帳戶 以程式設計方式代表您在 中建立IAM角色。Amazon Redshift 會自動將現有的 AWS 受管政策連接至IAM角色。此方法表示您可以留在 Amazon Redshift 主控台中，而且不需要切換到IAM主控台來建立角色。如需詳細資訊，請參閱[將IAM角色建立為 Amazon Redshift 的預設角色](#)。

當IAM角色建立為叢集的預設值時，會包含 redshift作為資源名稱的一部分，或使用 Redshift 特定標籤來標記這些資源。

如果您的叢集已開啟增強型 Amazon VPC路由，您可以使用透過 Amazon Redshift 主控台建立 IAM的角色。此IAM角色已連接AmazonRedshiftAllCommandsFullAccess政策，並將下列許可新增至政策。這些額外許可允許 Amazon Redshift 建立和刪除您帳戶中的彈性網路介面 (ENI)，並將其附加至在 Amazon EC2或 Amazon 上執行的編譯任務ECS。這樣做可讓您的 Amazon S3 儲存貯體中的物件只能從虛擬私有雲端 (VPC) 內存取，並封鎖網際網路存取。

```
{
  "Effect": "Allow",
  "Action": [
    "ec2:DescribeVpcEndpoints",
    "ec2:DescribeDhcpOptions",
    "ec2:DescribeVpcs",
    "ec2:DescribeSubnets",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeNetworkInterfaces",
    "ec2>DeleteNetworkInterfacePermission",
    "ec2>DeleteNetworkInterface",
    "ec2>CreateNetworkInterfacePermission",
```

```

    "ec2:CreateNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
  ],
  "Resource": "*"
}

```

若要使用 Amazon Bedrock 基礎模型，請新增下列區段：

```

// Required section if you use Bedrock models.
{
  "Effect": "Allow",
  "Action": "bedrock:InvokeModel",
  "Resource": [
    "arn:aws:bedrock:<region>::foundation-model/*"
  ]
}

```

- 如果您想要使用更嚴格的政策建立IAM角色，您可以使用下列政策。您也可以修改此政策以因應您的需求。

Amazon S3 儲存貯體 `redshift-downloads/redshift-ml/` 是用於其他步驟和範例之範例資料的儲存所在位置。如果您不需要從 Amazon S3 載入資料，則可以將其移除。或者，也可以將其取代為您用來將資料載入 Amazon Redshift 的其他 Amazon S3 儲存貯體。

所以此 `your-account-id`、`####`、和 `amzn-s3-demo-bucket` 值是您在 `CREATEMODEL` 命令中指定的值。

(選用) 如果您在使用 Amazon Redshift ML 時指定 AWS KMS 金鑰，請使用範例政策的 AWS KMS 金鑰區段。所以此 `your-kms-key` value 是您作為 `CREATEMODEL` 命令一部分使用的金鑰。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:PutMetricData",
        "ecr:BatchCheckLayerAvailability",
        "ecr:BatchGetImage",

```

```

        "ecr:GetAuthorizationToken",
        "ecr:GetDownloadUrlForLayer",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:DescribeLogStreams",
        "logs:PutLogEvents",
        "sagemaker:*Job*",
        "sagemaker:AddTags",
        "sagemaker:CreateModel",
        "sagemaker:CreateEndpoint",
        "sagemaker:CreateEndpointConfig",
        "sagemaker>DeleteEndpoint",
        "sagemaker>DeleteEndpointConfig",
        "sagemaker>DeleteModel"
    ],
    "Resource": "*"
},
{
    "Effect": "Allow",
    "Action": [
        "iam:PassRole",
        "s3:AbortMultipartUpload",
        "s3:GetObject",
        "s3>DeleteObject",
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:iam::<your-account-id>:role/<your-role>",
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::redshift-downloads/*"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "s3:GetBucketLocation",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::redshift-downloads"
    ]
}
// Optional section needed if you use AWS KMS keys.

```



```

    ,{
      "Effect": "Allow",
      "Action": [
        "kms:CreateGrant",
        "kms:Decrypt",
        "kms:DescribeKey",
        "kms:Encrypt",
        "kms:GenerateDataKey*"
      ],
      "Resource": [
        "arn:aws:kms:<your-region>:<your-account-id>:key/<your-kms-key>"
      ]
    }
  ]
}

```

- 若要允許 Amazon Redshift 和 SageMaker 擔任角色以與其他服務互動，請將下列信任政策新增至IAM角色。

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "redshift.amazonaws.com",
          "sagemaker.amazonaws.com",
          "forecast.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- (選用) 建立 Amazon S3 儲存貯體和 AWS KMS 金鑰。這些可讓 Amazon Redshift 用來儲存傳送至 Amazon SageMaker 並從 Amazon 接收訓練模型的訓練資料 SageMaker。
- (選用) 建立角色IAM和 Amazon S3 儲存貯體的不同組合，以控制對不同使用者群組的存取。
- (選用) 當您開啟 Redshift 叢集的VPC路由時，請為 VPC Redshift 叢集所在的 建立 Amazon S3 端點和 SageMaker 端點。這樣做可讓流量在 期間在服務VPC之間執行CREATE您的 MODEL。如需VPC路由的詳細資訊，請參閱 [Amazon Redshift 中的增強VPC路由](#)。

如需VPC為超參數調校任務指定私有所需許可的詳細資訊，請參閱[搭配使用 Amazon Redshift ML 與 Amazon 所需的許可 SageMaker](#)。

如需如何使用 CREATEMODEL陳述式開始為不同使用案例建立模型的資訊，請參閱 [CREATE MODEL](#)。

管理許可和擁有權

就像其他資料庫物件 (例如資料表或函數) 一樣，Amazon Redshift 會繫結建立和使用機器學習模型來存取控制機制。建立執行預測函數的模型有不同的許可。

下列範例使用兩個使用者群組 retention_analyst_grp (模型建立者) 和 marketing_analyst_grp (模型使用者) 來說明 Amazon Redshift 如何管理存取控制。保留分析師會建立機器學習模型，讓其他一組使用者可透過取得的許可使用這些模型。

超級使用者可以GRANTUSER或GROUP許可使用以下陳述式建立機器學習模型。

```
GRANT CREATE MODEL TO GROUP retention_analyst_grp;
```

如果使用者在上擁有一般CREATE許可，具有此許可的使用者或群組可以在叢集中的任何結構描述中建立模型SCHEMA。機器學習模型是結構描述階層的一部分，與資料表、檢視表、程序和使用者定義函數類似。

假設結構描述 demo_ml 已存在，請依照下列方式授予兩個使用者群組結構描述上的許可。

```
GRANT CREATE, USAGE ON SCHEMA demo_ml TO GROUP retention_analyst_grp;
```

```
GRANT USAGE ON SCHEMA demo_ml TO GROUP marketing_analyst_grp;
```

若要讓其他使用者使用您的機器學習推論函數，請授予 EXECUTE許可。下列範例使用 EXECUTE許可授予 marketing_analyst_grp 使用該模型的GROUP許可。

```
GRANT EXECUTE ON MODEL demo_ml.customer_churn_auto_model TO GROUP  
marketing_analyst_grp;
```

使用 REVOKE陳述式搭配 CREATEMODEL和 EXECUTE，從使用者或群組撤銷這些許可。如需許可控制命令的相關資訊，請參閱 [GRANT](#) 和 [REVOKE](#)。

使用模型可解釋性與 Amazon Redshift ML

透過 Amazon Redshift ML 中的模型可解釋性，您可以使用特徵重要性值，協助您了解訓練資料中的每個屬性如何影響預測結果。

模型可解釋性可以解釋模型進行的預測，以協助改善您的機器學習 (ML) 模型。模型可解釋性可幫助解釋這些模型如何使用特徵歸因方法進行預測。

Amazon Redshift ML 整合了模型可解釋性，為 Amazon Redshift ML 使用者提供模型解釋功能。如需模型可解釋性的詳細資訊，請參閱 Amazon SageMaker 開發人員指南 中的 [什麼是 Machine Learning 預測的公平性和模型可解釋性？](#)。

模型解釋性還可監控模型在生產環境中為特徵屬性漂移所做的推論。它還提供了一些工具來幫助您生成模型治理報告，您可以使用這些報告告知風險和合規團隊以及外部監管機構。

當您在使用 CREATE MODEL 陳述式時指定 AUTO ON 或 AUTO OFF 選項時，在模型訓練任務完成後，會 SageMaker 建立解釋輸出。您可以使用 EXPLAIN_MODEL 函數，以 JSON 格式查詢可解釋性報告。如需詳細資訊，請參閱 [機器學習函數](#)。

Amazon Redshift ML 概率指標

在監督式學習問題中，類別標籤是使用輸入資料的預測結果。例如，如果您使用模型來預測客戶是否會重新訂閱串流服務，則可能的標籤是很有可能和不太可能。Redshift ML 會提供概率指標的功能，可為每個標籤指派概率以指出其可能性。這可以幫助您根據預測的結果做出更明智的決策。在 Amazon Redshift ML 中，建立問題類型為二進位分類或多類別分類的 AUTO ON 模型時，可以使用機率指標。如果您省略 AUTO ON 參數，Redshift ML 會假設模型應該有 AUTO ON。

建立模型

建立模型時，Amazon Redshift 會自動偵測模型類型和問題類型。如果是分類問題，Redshift 會自動建立第二個推論函數，您可以使用該函數輸出相對於每個標籤的機率。第二個推論函數的名稱是您指定的推論函數名稱接著字串 `_probabilities`。例如，如果您將推論函數命名為 `customer_churn_predict`，則第二個推論函數的名稱為 `customer_churn_predict_probabilities`。然後，您可以查詢此函數以獲取每個標籤的概率。

```
CREATE MODEL customer_churn_model
FROM customer_activity
    PROBLEM_TYPE BINARY_CLASSIFICATION
TARGET churn
```

```

FUNCTION customer_churn_predict
IAM_ROLE {default}
AUTO ON
SETTINGS ( S3_BUCKET 'amzn-s3-demo-bucket'

```

獲取概率

機率函數準備就緒後，執行 命令會傳回包含傳回機率陣列及其相關標籤的 [SUPER 類型](#)。例如，結果 "probabilities" : [0.7, 0.3], "labels" : ["False.", "True."] 表示 False 標籤的概率為 0.7，而 True 標籤的概率為 0.3。

```

SELECT customer_churn_predict_probabilities(Account_length, Area_code,
      VMail_message, Day_mins, Day_calls, Day_charge, Eve_mins, Eve_calls,
      Eve_charge, Night_mins, Night_calls, Night_charge, Intl_mins, Intl_calls,
      Intl_charge, Cust_serv_calls)
FROM customer_activity;

customer_churn_predict_probabilities
-----
{"probabilities" : [0.7, 0.3], "labels" : ["False.", "True."]}
{"probabilities" : [0.8, 0.2], "labels" : ["False.", "True."]}
{"probabilities" : [0.75, 0.25], "labels" : ["True.", "False"]}

```

概率和標籤陣列一律會按其概率降序排序。您可以編寫查詢，透過取消巢狀機率函數傳回 SUPER 的結果，僅傳回具有最高機率的預測標籤。

```

SELECT prediction.labels[0], prediction.probabilities[0]
      FROM (SELECT customer_churn_predict_probabilities(Account_length,
      Area_code,
      VMail_message, Day_mins, Day_calls, Day_charge, Eve_mins, Eve_calls,
      Eve_charge, Night_mins, Night_calls, Night_charge, Intl_mins, Intl_calls,
      Intl_charge, Cust_serv_calls) AS prediction
FROM customer_activity);

  labels  | probabilities
-----+-----
"False." | 0.7
"False." | 0.8
"True."  | 0.75

```

為了使查詢更簡單，您可以將預測函數的結果儲存在資料表中。

```
CREATE TABLE churn_auto_predict_probabilities AS
  (SELECT customer_churn_predict_probabilities(Account_length, Area_code,
    VMail_message, Day_mins, Day_calls, Day_charge, Eve_mins, Eve_calls,
    Eve_charge, Night_mins, Night_calls, Night_charge, Intl_mins,
    Intl_calls, Intl_charge, Cust_serv_calls) AS prediction
  FROM customer_activity);
```

您可以查詢含有結果的資料表，以僅傳回概率高於 0.7 的預測。

```
SELECT prediction.labels[0], prediction.probabilities[0]
FROM churn_auto_predict_probabilities
WHERE prediction.probabilities[0] > 0.7;
```

labels	probabilities
"False."	0.8
"True."	0.75

使用索引符號，你可以得到特定標籤的概率。下列範例會傳回所有 True. 標籤的概率。

```
SELECT label, index, p.prediction.probabilities[index]
FROM churn_auto_predict_probabilities p, p.prediction.labels AS label AT index
WHERE label='True.';
```

label	index	probabilities
"True."	0	0.3
"True."	0	0.2
"True."	0	0.75

下列範例會傳回所有具有 True. 標籤且概率大於 0.7 的資料列，這表示客戶可能流失。

```
SELECT prediction.labels[0], prediction.probabilities[0]
FROM churn_auto_predict_probabilities
WHERE prediction.probabilities[0] > 0.7 AND prediction.labels[0] = "True.";
```

labels	probabilities
"True."	0.75

Amazon Redshift ML 的教學課程

您可以使用 Amazon Redshift ML 使用 SQL 陳述式訓練機器學習模型，然後在 SQL 查詢中叫用模型進行預測。Amazon Redshift 中的機器學習會使用單一 SQL 命令訓練模型。Amazon Redshift 會自動在 Amazon 中啟動訓練任務，SageMaker 並產生模型。建立模型之後，您可以使用模型的預測函數在 Amazon Redshift 中執行預測。

請遵循這些教學課程中的步驟以瞭解 Amazon Redshift ML 特徵：

- [教學課程：建置客戶流失模型](#) – 在本教學課程中，您可以使用 Amazon Redshift ML 建立使用 CREATE MODEL 命令的客戶流失模型，並針對使用者案例執行預測查詢。然後，您可以使用 CREATE MODEL 命令產生的 SQL 函數實作查詢。
- [教學課程：建置遠端推論模型](#) – 下列教學課程會說明如何建立先前在 Amazon Redshift SageMaker 之外在 Amazon 中訓練和部署的 [隨機剪切森林模型](#)。
- [教學課程：建置 K 平均值叢集模型](#) – 在本教學課程中，您可以使用 Amazon Redshift ML 根據 [K 平均演算法](#) 建立、訓練和部署機器學習模型。
- [教學課程：建置多類別分類模型](#) – 在本教學課程中，您可以使用 Amazon Redshift ML 來建立機器學習模型，以解決多類別分類問題。多類別分類演算法將資料點分類為三個或多個類別之一。然後，您可以使用 CREATE MODEL 命令產生的 SQL 函數實作查詢。
- [教學課程：建置 XGBoost 模型](#) – 在本教學課程中，您可以使用 Amazon S3 的資料建立模型，並使用 Amazon Redshift ML 對模型執行預測查詢。XGBoost 演算法是漸層增強型樹狀目錄演算法的最佳化實作。
- [教學課程：建置迴歸模型](#) – 在本教學課程中，您可以使用 Amazon Redshift ML 建立機器學習迴歸模型，並在模型上執行預測查詢。迴歸模型允許您預測數值結果，例如房屋的價格，或有多少人將使用城市的自行車租賃服務。
- [教學課程：使用線性學習程式建置迴歸模型](#) – 在本教學課程中，您會使用 Amazon S3 的資料建立線性學習器模型，並使用 Amazon Redshift ML 對模型執行預測查詢。SageMaker 線性學習器演算法可解決迴歸或多類別分類問題。
- [教學課程：使用線性學習程式建置多類別分類模型](#) – 在本教學課程中，您可以使用 Amazon S3 的資料建立線性學習器模型，然後使用 Amazon Redshift ML 對模型執行預測查詢。SageMaker 線性學習器演算法可解決迴歸或分類問題。

教學課程：建置客戶流失模型

在本教學課程中，您可以使用 Amazon Redshift ML 建立使用 CREATEMODEL 命令的客戶流失模型，並針對使用者案例執行預測查詢。然後，您可以使用 CREATE MODEL 命令產生的 SQL 函數實作查詢。

您可以使用簡單的 CREATEMODEL 命令來匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATEMODEL 陳述式來指定訓練資料作為資料表或 SELECT 陳述式。

此範例會使用歷史資訊來建構行動電信業者的客戶流失機器學習模型。首先，SageMaker 訓練您的機器學習模型，然後使用任意客戶的設定檔資訊來測試模型。驗證模型後，Amazon 會將模型和預測函數 SageMaker 部署到 Amazon Redshift。您可以使用預測函數來預測客戶是否會流失。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他二進制分類問題，例如預測銷售潛在客戶是否會結案。您還可以預測金融交易是否為詐騙。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

若要完成本教學課程，您需要以下先決條件：

- 您必須為 Amazon Redshift ML 設定一個 Amazon Redshift 叢集。若要這麼做，請使用[針對 Amazon Redshift ML 管理進行叢集和組態設定的文件](#)。
- 您用來建立模型的 Amazon Redshift 叢集，以及用來暫存訓練資料和儲存模型成品的 Amazon S3 儲存貯體必須位於相同 AWS 區域。
- 若要下載本文件中使用的 SQL 命令和範例資料集，請執行下列其中一項操作：
 - 下載[SQL 命令](#)、[客戶活動檔案](#) 和 [Abalone 檔案](#)。
 - 使用 AWS CLI for Amazon S3，執行下列命令。您可以使用自己的目標路徑。

```
aws s3 cp s3://redshift-downloads/redshift-ml/tutorial-scripts/redshift-ml-tutorial.sql </target/path>
aws s3 cp s3://redshift-downloads/redshift-ml/customer_activity/customer_activity.csv </target/path>
aws s3 cp s3://redshift-downloads/redshift-ml/abalone_xgb/abalone_xgb.csv </target/path>
```

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 編輯和執行查詢，以及視覺化結果。

執行下列查詢會建立名為 `customer_activity` 的資料表，並從 Amazon S3 擷取範例資料集。

```
DROP TABLE IF EXISTS customer_activity;

CREATE TABLE customer_activity (
  state varchar(2),
  account_length int,
  area_code int,
  phone varchar(8),
  intl_plan varchar(3),
  vMail_plan varchar(3),
  vMail_message int,
  day_mins float,
  day_calls int,
  day_charge float,
  total_charge float,
  eve_mins float,
  eve_calls int,
  eve_charge float,
  night_mins float,
  night_calls int,
  night_charge float,
  intl_mins float,
  intl_calls int,
  intl_charge float,
  cust_serv_calls int,
  churn varchar(6),
  record_date date
);
```



```
COPY customer_activity
FROM 's3://redshift-downloads/redshift-ml/customer_activity/'
REGION 'us-east-1' IAM_ROLE default
FORMAT AS CSV IGNOREHEADER 1;
```

步驟 2：建立機器學習模型

流失率是我們在此模型中的目標輸入。模型的所有其他輸入都是有助於建立函數以預測流失的屬性。

下列範例使用 CREATEMODEL 操作來交付模型，該模型會使用客戶的年齡、郵遞區號、支出和案例等輸入來預測客戶是否處於作用中狀態。在下列範例中，將 amzn-s3-demo-bucket 取代為您自己的 Amazon S3 儲存貯體。

```
CREATE MODEL customer_churn_auto_model
FROM
  (
    SELECT state,
           account_length,
           area_code,
           total_charge/account_length AS average_daily_spend,
           cust_serv_calls/account_length AS average_daily_cases,
           churn
    FROM customer_activity
    WHERE record_date < '2020-01-01'
  )
TARGET churn FUNCTION ml_fn_customer_churn_auto
IAM_ROLE default SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket'
);
```

上述範例中的 SELECT 查詢會建立訓練資料。TARGET 子句指定哪個資料欄是 CREATEMODEL 操作用來學習如何預測的機器學習標籤。目標資料欄「流失率」會指出客戶是否仍然具有作用中的會員資格或已暫停會員資格。S3_BUCKET 欄位是您先前建立的 Amazon S3 儲存貯體的名稱。Amazon S3 儲存貯體用於在 Amazon Redshift 和 Amazon 之間共用訓練資料和成品 SageMaker。其餘的資料欄是用於預測的特徵。

如需 CREATEMODEL 命令基本使用案例的語法和功能摘要，請參閱 [簡單 CREATE MODEL](#)。

新增伺服器端加密的許可 (選用)

根據預設，Amazon Redshift 會使用 Amazon SageMaker Autopilot 進行訓練。特別是，Amazon Redshift 會將訓練資料安全地匯出到客戶指定的 Amazon S3 儲存貯體。如果您未指定 KMS_KEY_ID，則預設會使用伺服器端加密 SSE-S3 加密資料。

當您使用受 AWS KMS 管金鑰 (SSE-MMS) 使用伺服器端加密來加密輸入時，請新增下列許可：

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt"
    "kms:Decrypt"
  ]
}
```

如需 Amazon SageMaker 角色的詳細資訊，請參閱 [Amazon 開發人員指南](#) 中的 [Amazon SageMaker 角色](#)。 SageMaker

檢查模型訓練的狀態 (選擇性)

您可以使用 SHOWMODEL 命令來了解模型何時準備就緒。

使用下列操作來檢查模型的狀態。

```
SHOW MODEL customer_churn_auto_model;
```

下列為上一個操作的輸出範例。

```
+-----+
+-----+
+
|           Key           |
|           Value        |
|           |            |
+-----+
+-----+
+
| Model Name              |
| customer_churn_auto_model |
|                           |
```

```

|      Schema Name      |
|          public      |
|      |               |
|      Owner           |
|          awsuser     |
|      |               |
|      Creation Time   |
|      Tue, 14.06.2022 17:15:52
|      |               |
|      Model State     |
|          TRAINING    |
|      |               |
|      |               |
|      TRAINING DATA: |
|      |               |
|      Query           | SELECT STATE, ACCOUNT_LENGTH, AREA_CODE, TOTAL_CHARGE /
|      ACCOUNT_LENGTH AS AVERAGE_DAILY_SPEND, CUST_SERV_CALLS / ACCOUNT_LENGTH AS
|      AVERAGE_DAILY_CASES, CHURN |
|      |               |
|      FROM CUSTOMER_ACTIVITY
|      |               |
|      WHERE RECORD_DATE < '2020-01-01'
|      |               |
|      Target Column   |
|          CHURN       |
|      |               |
|      |               |
|      PARAMETERS:    |
|      |               |
|      Model Type      |
|          auto        |
|      |               |
|      Problem Type    |
|      |               |
|      |               |

```

```

|      Objective      |
|
|      AutoML Job Name      |
|      redshiftml-20220614171552640901      |
|
|      Function Name      |
|      ml_fn_customer_churn_auto      |
|
|      Function Parameters      |
|      account_length area_code average_daily_spend average_daily_cases      |
|
|      Function Parameter Types      |
|      varchar int4 int4 float8 int4      |
|
|      IAM Role      |
|      default-aws-iam-role      |
|
|      S3 Bucket      |
|      amzn-s3-demo-bucket      |
|
|      Max Runtime      |
|      5400      |
|
+-----+
+-----+
+

```

模型訓練完成後，`model_state` 變數會變成 `Model is Ready`，且預測函數會變為可用。

步驟 3：執行模型的預測

您可以使用 SQL 陳述式來檢視預測模型所做的預測。在此範例中，`CREATEMODEL` 操作建立的預測函數名為 `ml_fn_customer_churn_auto`。預測函數的輸入引數會對應於特徵的類型，例如 `varchar` 會用於 `state`，而整數會用於 `account_length`。預測函數的輸出與 `CREATEMODEL` 陳述式的資料 `TARGET` 欄類型相同。

1. 您用了 2020-01-01 之前的資料訓練模型，因此現在您可以在測試集上使用預測功能。以下查詢顯示 2020-01-01 之後註冊的客戶是否會經歷流失的預測。

```

SELECT
    phone,
    ml_fn_customer_churn_auto(

```

```

        state,
        account_length,
        area_code,
        total_charge / account_length,
        cust_serv_calls / account_length
    ) AS active
FROM
    customer_activity
WHERE
    record_date > '2020-01-01';

```

2. 下列範例針對不同的使用案例使用相同的預測函數。在此案例中，Amazon Redshift 會從不同州的客戶中 (記錄日期大於 2020-01-01)，預測流失者和非流失者的比例。

```

WITH predicted AS (
    SELECT
        state,
        ml_fn_customer_churn_auto(
            state,
            account_length,
            area_code,
            total_charge / account_length,
            cust_serv_calls / account_length
        ) :: varchar(6) AS active
    FROM
        customer_activity
    WHERE
        record_date > '2020-01-01'
)
SELECT
    state,
    SUM(
        CASE
            WHEN active = 'True.' THEN 1
            ELSE 0
        END
    ) AS churners,
    SUM(
        CASE
            WHEN active = 'False.' THEN 1
            ELSE 0
        END
    ) AS nonchurners,

```

```
    COUNT(*) AS total_per_state
FROM
    predicted
GROUP BY
    state
ORDER BY
    state;
```

3. 下列範例會使用預測函數來預測某州中流失的客戶百分比。在此案例中，Amazon Redshift 會預測記錄日期大於 2020-01-01 的流失百分比。

```
WITH predicted AS (
    SELECT
        state,
        ml_fn_customer_churn_auto(
            state,
            account_length,
            area_code,
            total_charge / account_length,
            cust_serv_calls / account_length
        ) :: varchar(6) AS active
    FROM
        customer_activity
    WHERE
        record_date > '2020-01-01'
)
SELECT
    state,
    CAST((CAST((SUM(
        CASE
            WHEN active = 'True.' THEN 1
            ELSE 0
        END
    ))) AS FLOAT) / CAST(COUNT(*) AS FLOAT)) AS DECIMAL (3, 2)) AS pct_churn,
    COUNT(*) AS total_customers_per_state
FROM
    predicted
GROUP BY
    state
ORDER BY
    3 DESC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon RedshiftML 的成本](#)
- [CREATE MODEL 命令](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：建置遠端推論模型

下列教學課程會說明如何建立先前在 Amazon Redshift SageMaker 之外，在 Amazon 中訓練和部署的 [隨機剪切森林模型](#)。隨機分割森林演算法會偵測資料集中的異常資料點。建立具有遠端推論的模型可讓您將隨機剪切森林 SageMaker 模型帶入 Amazon Redshift。然後，在 Amazon Redshift 中，您可以使用 SQL 對遠端 SageMaker 端點執行預測。

您可以使用 CREATEMODEL 命令從 Amazon SageMaker 端點匯入機器學習模型，並準備 Amazon Redshift 預測函數。使用 CREATEMODEL 操作時，您會提供 SageMaker 機器學習模型的端點名稱。

在本教學課程中，您可以使用模型端點建立 Amazon Redshift 機器學習 SageMaker 模型。一旦您的機器學習模型準備就緒，您就可以使用它在 Amazon Redshift 中執行預測。首先，您可以在 Amazon 中訓練並建立端點 SageMaker，然後取得端點名稱。然後，您可以使用 CREATEMODEL 命令來使用 Amazon Redshift ML 建立模型。最後，您可以使用 CREATEMODEL 命令產生的預測函數對模型執行預測。

使用案例範例

您可以使用隨機分割森林模型和遠端推論來偵測其他資料集中的異常狀況，例如預測電子商務交易的快速增加或減少。您還可以預測天氣或地震活動的顯著變化。

工作

- 必要條件

- 步驟 1：部署 Amazon SageMaker 模型
- 步驟 2：取得 SageMaker 模型端點
- 步驟 3：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 4：建立 Amazon Redshift ML 的模型
- 步驟 5：執行模型的預測

必要條件

若要完成本教學課程，您需要以下先決條件：

- 您已完成 Amazon Redshift ML 的[管理設定](#)。
- 您已下載[NYC計程車資料集](#)、[建立 Amazon S3 儲存貯體](#)，並將[資料上傳到 Amazon S3 儲存貯體](#)。
- 您必須訓練、部署 SageMaker 模型和端點，並取得 SageMaker 端點的名稱。[使用此 AWS CloudFormation 範本](#)自動佈建 AWS 帳戶中的所有 SageMaker 資源。

步驟 1：部署 Amazon SageMaker 模型

1. 若要部署模型，請前往 Amazon SageMaker 主控台，在導覽窗格中的筆記本下選擇筆記本執行個體。
2. 針對範本建立的 Jupyter 筆記本選擇開啟 Jupyter CloudFormation。
3. 選擇 `bring-your-own-model-remote-inference.ipynb`。
4. 將以下幾行取代為 Amazon S3 儲存貯體和字首，設定參數來將訓練輸入和輸出儲存在 Amazon S3 中。

```
data_location=f"s3://{bucket}/{prefix}/",
output_path=f"s3://{bucket}/{prefix}/output",
```

5. 選擇快轉按鈕以執行所有儲存格。

步驟 2：取得 SageMaker 模型端點

在 Amazon SageMaker 主控台的導覽窗格中推論下，選擇端點並尋找模型名稱。在 Amazon Redshift 中建立遠端推論模型時，必須複製模型的端點名稱。

步驟 3：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 在 Amazon Redshift 中執行下列 SQL 命令。如果 `rcf_taxi_data` 資料表存在，這些命令會捨棄資料表、建立相同名稱的資料表，然後將範例資料集載入資料表中。

```
DROP TABLE IF EXISTS public.rcf_taxi_data CASCADE;

CREATE TABLE public.rcf_taxi_data (ride_timestamp timestamp, nbr_passengers int);

COPY public.rcf_taxi_data
FROM
    's3://sagemaker-sample-files/datasets/tabular/anomaly_benchmark_taxi/
NAB_nyc_taxi.csv'
IAM_ROLE default
IGNOREHEADER 1
FORMAT AS CSV;
```

步驟 4：建立 Amazon Redshift ML 的模型

執行下列查詢，使用您在上一步取得的模型端點在 Amazon Redshift ML 中建立 SageMaker 模型。Replace (取代) `randomcutforest-xxxxxxxx` 使用您自己的 SageMaker 端點名稱。

```
CREATE MODEL public.remote_random_cut_forest
FUNCTION remote_fn_rcf(int)
RETURNS decimal(10, 6) SAGEMAKER '<randomcutforest-xxxxxxxx>' IAM_ROLE default;
```

檢查模型狀態 (選擇性)

您可以使用 SHOWMODEL 命令來了解模型何時準備就緒。

若要檢查模型狀態，請使用下列 SHOWMODEL 操作。

```
SHOW MODEL public.remote_random_cut_forest
```

輸出會顯示 SageMaker 端點和函數名稱。

```
+-----+-----+
|      Model Name      | remote_random_cut_forest |
+-----+-----+
|      Schema Name    | public                    |
+-----+-----+
```

Owner	awsuser
Creation Time	Wed, 15.06.2022 17:58:21
Model State	READY
PARAMETERS:	
Endpoint	<randomcutforest-xxxxxxxx>
Function Name	remote_fn_rcf
Inference Type	Remote
Function Parameter Types	int4
IAM Role	default-aws-iam-role

步驟 5：執行模型的預測

Amazon SageMaker Random Cut Forest 演算法旨在偵測資料集內的異常資料點。在此範例中，您的模型旨在偵測由於重要事件而導致的計程車乘車尖峰。您可以透過為每個資料點產生異常分數，使用模型來預測異常事件。

使用下列查詢來計算整個計程車資料集的異常分數。請注意，您會參考您在上一個步驟中 CREATEMODEL 陳述式中使用的函數。

```
SELECT
    ride_timestamp,
    nbr_passengers,
    public.remote_fn_rcf(nbr_passengers) AS score
FROM
    public.rcf_taxi_data;
```

檢查是高分和低分的異常 (選擇性)

執行下列查詢，以尋找分數大於平均分數三個標準差的任何資料點。

```
WITH score_cutoff AS (
    SELECT
        STDDEV(public.remote_fn_rcf(nbr_passengers)) AS std,
        AVG(public.remote_fn_rcf(nbr_passengers)) AS mean,
        (mean + 3 * std) AS score_cutoff_value
    FROM
        public.rcf_taxi_data
)
SELECT
    ride_timestamp,
```

```
    nbr_passengers,
    public.remote_fn_rcf(nbr_passengers) AS score
FROM
    public.rcf_taxi_data
WHERE
    score > (
        SELECT
            score_cutoff_value
        FROM
            score_cutoff
    )
ORDER BY
    2 DESC;
```

執行下列查詢，以尋找分數大於平均分數三個標準差的任何資料點。

```
WITH score_cutoff AS (
    SELECT
        STDDEV(public.remote_fn_rcf(nbr_passengers)) AS std,
        AVG(public.remote_fn_rcf(nbr_passengers)) AS mean,
        (mean - 3 * std) AS score_cutoff_value
    FROM
        public.rcf_taxi_data
)
SELECT
    ride_timestamp,
    nbr_passengers,
    public.remote_fn_rcf(nbr_passengers) AS score
FROM
    public.rcf_taxi_data
WHERE
    score < (
        SELECT
            score_cutoff_value
        FROM
            score_cutoff
    )
ORDER BY
    2 DESC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：建置 K 平均值叢集模型

在本教學課程中，您會使用 Amazon Redshift ML 來建立、訓練和部署以 [K 平均值演算法](#) 為基礎的機器學習模型。此演算法可解決您想在資料中探索分組的叢集問題。K 平均值有助於對尚未標記的資料進行分組。若要進一步了解 K 平均數叢集，請參閱 Amazon SageMaker 開發人員指南中的 [K 平均數叢集如何運作](#)。

您將使用 CREATEMODEL 操作從 Amazon Redshift 叢集建立 K 平均數模型。您可以使用 CREATEMODEL 命令匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATEMODEL 操作，以資料表或 SELECT 陳述式的形式指定訓練資料。

在本教學課程中，您可以在 [事件、語言和音調 \(GDELT\) 資料集的全球資料庫](#) 上使用 K 平均值，該資料庫會監控全球新聞，而且資料每天都會儲存一秒。K 平均值會將音調、動作者或位置相似的事件進行分組。資料會以多個檔案形式儲存在 Amazon Simple Storage Service 中 (儲存在兩個不同的資料夾中)。這些資料夾為歷史資料夾 (涵蓋 1979–2013 年) 及每日更新資料夾 (涵蓋 2013 年及以後的年份)。在此範例中，我們使用歷史格式，並帶入 1979 年的資料。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他叢集問題，例如對在串流服務上具有類似檢視習慣的客戶進行分組。您也可以使用 Redshift ML 來預測遞送服務的最佳託運中心數量。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的[管理設定](#)。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

1. 使用 [Amazon Redshift 查詢編輯器 v2](#) 來執行下列查詢。查詢會在公用結構描述中捨棄 `gdelt_data` 資料表 (如果資料表存在)，並在公用結構描述中建立相同名稱的資料表。

```
DROP TABLE IF EXISTS gdelt_data CASCADE;

CREATE TABLE gdelt_data (
  GlobalEventId bigint,
  SqlDate bigint,
  MonthYear bigint,
  Year bigint,
  FractionDate double precision,
  Actor1Code varchar(256),
  Actor1Name varchar(256),
  Actor1CountryCode varchar(256),
  Actor1KnownGroupCode varchar(256),
  Actor1EthnicCode varchar(256),
  Actor1Religion1Code varchar(256),
  Actor1Religion2Code varchar(256),
  Actor1Type1Code varchar(256),
  Actor1Type2Code varchar(256),
  Actor1Type3Code varchar(256),
  Actor2Code varchar(256),
  Actor2Name varchar(256),
  Actor2CountryCode varchar(256),
  Actor2KnownGroupCode varchar(256),
  Actor2EthnicCode varchar(256),
  Actor2Religion1Code varchar(256),
  Actor2Religion2Code varchar(256),
  Actor2Type1Code varchar(256),
  Actor2Type2Code varchar(256),
  Actor2Type3Code varchar(256),
  IsRootEvent bigint,
  EventCode bigint,
```

```
EventBaseCode bigint,  
EventRootCode bigint,  
QuadClass bigint,  
GoldsteinScale double precision,  
NumMentions bigint,  
NumSources bigint,  
NumArticles bigint,  
AvgTone double precision,  
Actor1Geo_Type bigint,  
Actor1Geo_FullName varchar(256),  
Actor1Geo_CountryCode varchar(256),  
Actor1Geo_ADM1Code varchar(256),  
Actor1Geo_Lat double precision,  
Actor1Geo_Long double precision,  
Actor1Geo_FeatureID bigint,  
Actor2Geo_Type bigint,  
Actor2Geo_FullName varchar(256),  
Actor2Geo_CountryCode varchar(256),  
Actor2Geo_ADM1Code varchar(256),  
Actor2Geo_Lat double precision,  
Actor2Geo_Long double precision,  
Actor2Geo_FeatureID bigint,  
ActionGeo_Type bigint,  
ActionGeo_FullName varchar(256),  
ActionGeo_CountryCode varchar(256),  
ActionGeo_ADM1Code varchar(256),  
ActionGeo_Lat double precision,  
ActionGeo_Long double precision,  
ActionGeo_FeatureID bigint,  
DATEADDED bigint  
);
```

2. 下列查詢會將範例資料載入 `gdelt_data` 資料表中。

```
COPY gdelt_data  
FROM 's3://gdelt-open-data/events/1979.csv'  
REGION 'us-east-1'  
IAM_ROLE default  
CSV  
DELIMITER '\t';
```

檢查訓練資料 (選擇性)

若要查看您的模型將在哪些資料上進行訓練，請使用下列查詢。

```
SELECT
  AvgTone,
  EventCode,
  NumArticles,
  Actor1Geo_Lat,
  Actor1Geo_Long,
  Actor2Geo_Lat,
  Actor2Geo_Long
FROM
  gdelt_data LIMIT 100;
```

步驟 2：建立機器學習模型

下列範例使用 CREATEMODEL 命令來建立模型，將資料分組為七個叢集。K 值是資料點將分散在其中的叢集數目。模型會將您的資料點分類成叢集，其中資料點彼此更相似。透過將資料點分成群組，K 平均值演算法會反覆判斷最佳的叢集中心。然後演算法會將每個資料點指派給最近的叢集中心。最接近屬於同一群組的相同叢集中心的成員。群組的成員會盡可能地與相同群組中的其他成員相似，並盡可能地和其他群組的成員不同。K 值是主觀的，取決於測量資料點之間相似性的方法。如果叢集分佈不均，您可以變更 K 值以平滑化叢集大小。

在下列範例中，將 amzn-s3-demo-bucket 取代為您自己的 Amazon S3 儲存貯體。

```
CREATE MODEL news_data_clusters
FROM
  (
    SELECT
      AvgTone,
      EventCode,
      NumArticles,
      Actor1Geo_Lat,
      Actor1Geo_Long,
      Actor2Geo_Lat,
      Actor2Geo_Long
    FROM
      gdelt_data
  ) FUNCTION news_monitoring_cluster
IAM_ROLE default
AUTO OFF
```

```

MODEL_TYPE KMEANS
PREPROCESSORS 'none'
HYPERPARAMETERS DEFAULT
EXCEPT
(K '7')
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');

```

檢查模型訓練的狀態 (選擇性)

您可以使用 SHOWMODEL 命令來了解模型何時準備就緒。

若要檢查模型狀態，請使用下列 SHOWMODEL 操作，並尋找 Model State 是否為 Ready。

```
SHOW MODEL NEWS_DATA_CLUSTERS;
```

當模型準備就緒時，先前操作的輸出應會顯示 Model State 為 Ready。以下是 SHOWMODEL 操作輸出的範例。

```

+-----+
+-----+
+
|      Model Name      |
news_data_clusters
+-----+
+-----+
+
|      Schema Name    |                                public
|
|      Owner          |                                awsuser
|
|      Creation Time  |                                Fri, 17.06.2022
16:32:19
|
|      Model State    |                                READY
|
|      train:msd      |                                2973.822754
|
|      train:progress |                                100.000000
|
|      train:throughput |                                237114.875000
|
|      Estimated Cost |                                0.004983
|

```



```

|
|
| TRAINING DATA:
|
| Query | SELECT AVGTONE, EVENTCODE, NUMARTICLES, ACTOR1GEO_LAT,
| ACTOR1GEO_LONG, ACTOR2GEO_LAT, ACTOR2GEO_LONG |
|
| FROM GDELT_DATA
|
|
| PARAMETERS:
|
| Model Type | kmeans
|
| Training Job Name |
| redshiftml-20220617163219978978-kmeans |
| Function Name |
| news_monitoring_cluster |
| Function Parameters | avgtone eventcode numarticles actor1geo_lat
| actor1geo_long actor2geo_lat actor2geo_long |
| Function Parameter Types | float8 int8 int8 float8 float8
| float8 float8 |
| IAM Role | default-aws-iam-
| role |
| S3 Bucket | amzn-s3-demo-
| bucket |
| Max Runtime | 5400
|
|
| HYPERPARAMETERS:
|
| feature_dim | 7
|
| k | 7
|
+-----+
+-----+
+

```

步驟 3：執行模型的預測

識別叢集

您可以找到模型在資料中識別的離散群組 (也稱為叢集)。比起任何其他叢集中心，叢集是更接近其叢集中心的一組資料點。由於 K 值代表模型中的叢集數目，因此也代表叢集中心的數目。下面的查詢會透過顯示與每個 `globaleventid` 相關聯的叢集來識別叢集。

```
SELECT
  globaleventid,
  news_monitoring_cluster (
    AvgTone,
    EventCode,
    NumArticles,
    Actor1Geo_Lat,
    Actor1Geo_Long,
    Actor2Geo_Lat,
    Actor2Geo_Long
  ) AS cluster
FROM
  gdelt_data;
```

檢查資料的分佈

您可以檢查叢集之間的資料分佈，以查看您選擇的 K 值是否會讓資料稍微均勻地分佈。使用下列查詢來判斷資料是否平均分佈在叢集中。

```
SELECT
  events_cluster,
  COUNT(*) AS nbr_events
FROM
  (
    SELECT
      globaleventid,
      news_monitoring_cluster(
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
      ) AS events_cluster
```

```
        FROM
            gdelt_data
    )
GROUP BY
    1;
```

請注意，如果叢集分佈不均，您可以變更 K 值以平滑化叢集大小。

判斷叢集中心

比起任何其他叢集中心，資料點會更接近其叢集中心。因此，尋找叢集中心可協助您定義叢集。

執行下列查詢，根據事件代碼的文章數目判斷叢集的中心。

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    eventcode,
    SUM(numArticles) AS numArticles
FROM
    gdelt_data
GROUP BY
    1,
    2;
```

顯示叢集中資料點的相關資訊

使用以下查詢傳回指派給第五個叢集的點的資料。選定的文章必須有兩個動作者。

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
```

```
        Actor2Geo_Long
    ) AS events_cluster,
    eventcode,
    actor1name,
    actor2name,
    SUM(numarticles) AS totalarticles
FROM
    gdelt_data
WHERE
    events_cluster = 5
    AND actor1name <> ' '
    AND actor2name <> ' '
GROUP BY
    1,
    2,
    3,
    4
ORDER BY
    5 desc;
```

顯示具有相同族裔代碼的動作者的事件資料

下列查詢會計算以正面語調撰寫有關事件的文章數目。該查詢也會要求兩個動作者具有相同的族裔代碼，並傳回每個事件分派給哪個叢集。

```
SELECT
    news_monitoring_cluster (
        AvgTone,
        EventCode,
        NumArticles,
        Actor1Geo_Lat,
        Actor1Geo_Long,
        Actor2Geo_Lat,
        Actor2Geo_Long
    ) AS events_cluster,
    SUM(numarticles) AS total_articles,
    eventcode AS event_code,
    Actor1EthnicCode AS ethnic_code
FROM
    gdelt_data
WHERE
    Actor1EthnicCode = Actor2EthnicCode
    AND Actor1EthnicCode <> ' ';
```

```
    AND Actor2EthnicCode <> ' '
    AND AvgTone > 0
GROUP BY
    1,
    3,
    4
HAVING
    (total_articles) > 4
ORDER BY
    1,
    2 ASC;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：建置多類別分類模型

在本教學課程中，您會使用 Amazon Redshift ML 建立可解決多類別分類問題的機器學習模型。多類別分類演算法將資料點分類為三個或多個類別之一。然後，您可以使用 CREATE MODEL 命令產生的 SQL 函數實作查詢。

您可以使用 CREATEMODEL 命令匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATEMODEL 操作來指定訓練資料，可以是資料表或 SELECT 陳述式。

若要按照本教學課程進行操作，您可以使用公共資料集 [電子商務銷售預測](#)，其中包括線上英國零售商的銷售資料。您產生的模型將以特殊客戶忠誠度計劃中最活躍的客戶為目標。透過多類別分類，您可以使用該模型來預測客戶在 13 個月之間有多少個月是活躍的。預測函數會指定預計會活躍 7 個月或更長時間的客戶來加入該計劃。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他多類別分類問題，例如從產品線預測暢銷產品。您還可以預測圖像包含哪些水果，例如選擇蘋果、梨子或柳橙。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的[管理設定](#)。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 來執行下列查詢。這些查詢會將範例資料載入 Amazon Redshift。

1. 以下範例會查詢名為 `ecommerce_sales` 的資料表。

```
CREATE TABLE IF NOT EXISTS ecommerce_sales (  
    invoiceno VARCHAR(30),  
    stockcode VARCHAR(30),  
    description VARCHAR(60),  
    quantity DOUBLE PRECISION,  
    invoicedate VARCHAR(30),  
    unitprice DOUBLE PRECISION,  
    customerid BIGINT,  
    country VARCHAR(25)  
);
```

2. 下列查詢會將[電子商務銷售預測資料集](#)中的範例資料複製到 `ecommerce_sales` 資料表中。

```
COPY ecommerce_sales  
FROM  
    's3://redshift-ml-multiclass/ecommerce_data.txt'  
IAM_ROLE default  
DELIMITER '\t'
```

```
IGNOREHEADER 1
REGION 'us-east-1'
MAXERROR 100;
```

分割資料

當您在 Amazon Redshift ML 中建立模型時，SageMaker 會自動將資料分割為訓練和測試集，以便 SageMaker 可以判斷模型準確性。透過在此步驟手動分割資料，您將能夠透過配置額外預測集來驗證模型的準確性。

使用下列SQL陳述式將資料分割為三組，以進行訓練、驗證和預測。

```
--creates table with all data
CREATE TABLE ecommerce_sales_data AS (
  SELECT
    t1.stockcode,
    t1.description,
    t1.invoicedate,
    t1.customerid,
    t1.country,
    t1.sales_amt,
    CAST(RANDOM() * 100 AS INT) AS data_group_id
  FROM
    (
      SELECT
        stockcode,
        description,
        invoicedate,
        customerid,
        country,
        SUM(quantity * unitprice) AS sales_amt
      FROM
        ecommerce_sales
      GROUP BY
        1,
        2,
        3,
        4,
        5
    ) t1
);
```

```
--creates training set
CREATE TABLE ecommerce_sales_training AS (
  SELECT
    a.customerid,
    a.country,
    a.stockcode,
    a.description,
    a.invoicedate,
    a.sales_amt,
    (b.nbr_months_active) AS nbr_months_active
  FROM
    ecommerce_sales_data a
  INNER JOIN (
    SELECT
      customerid,
      COUNT(
        DISTINCT(
          DATE_PART(y, CAST(invoicedate AS DATE)) || '-' || LPAD(
            DATE_PART(mon, CAST(invoicedate AS DATE)),
            2,
            '00'
          )
        )
      ) AS nbr_months_active
    FROM
      ecommerce_sales_data
    GROUP BY
      1
  ) b ON a.customerid = b.customerid
  WHERE
    a.data_group_id < 80
);

--creates validation set
CREATE TABLE ecommerce_sales_validation AS (
  SELECT
    a.customerid,
    a.country,
    a.stockcode,
    a.description,
    a.invoicedate,
    a.sales_amt,
    (b.nbr_months_active) AS nbr_months_active
  FROM
```



```
ecommerce_sales_data a
INNER JOIN (
  SELECT
    customerid,
    COUNT(
      DISTINCT(
        DATE_PART(y, CAST(invoicedate AS DATE)) || '-' || LPAD(
          DATE_PART(mon, CAST(invoicedate AS DATE)),
          2,
          '00'
        )
      )
    ) AS nbr_months_active
  FROM
    ecommerce_sales_data
  GROUP BY
    1
) b ON a.customerid = b.customerid
WHERE
  a.data_group_id BETWEEN 80
  AND 90
);

--creates prediction set
CREATE TABLE ecommerce_sales_prediction AS (
  SELECT
    customerid,
    country,
    stockcode,
    description,
    invoicedate,
    sales_amt
  FROM
    ecommerce_sales_data
  WHERE
    data_group_id > 90);
```

步驟 2：建立機器學習模型

在此步驟中，您可以使用 CREATEMODEL 陳述式來使用多類別分類建立機器學習模型。

下列查詢會使用 CREATEMODEL 操作建立具有訓練集的多類別分類模型。將 amzn-s3-demo-bucket 取代為您自己的 Amazon S3 儲存貯體。

```

CREATE MODEL ecommerce_customer_activity
FROM
  (
    SELECT
      customerid,
      country,
      stockcode,
      description,
      invoicedate,
      sales_amt,
      nbr_months_active
    FROM
      ecommerce_sales_training
  ) TARGET nbr_months_active FUNCTION predict_customer_activity IAM_ROLE default
PROBLEM_TYPE MULTICLASS_CLASSIFICATION SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  S3_GARBAGE_COLLECT OFF
);

```

在此查詢中，您可以將問題類型指定為 `Multiclass_Classification`。您為模型預測的目標是 `nbr_months_active`。SageMaker 完成模型訓練後，它會建立函數 `predict_customer_activity`，您將使用它在 Amazon Redshift 中進行預測。

顯示模型訓練的狀態 (選擇性)

您可以使用 `SHOWMODEL` 命令來了解模型何時準備就緒。

使用下列查詢傳回模型的各種指標，包括模型狀態和準確度。

```
SHOW MODEL ecommerce_customer_activity;
```

當模型準備就緒時，先前操作的輸出應會顯示 `Model State` 為 `Ready`。以下是 `SHOWMODEL` 操作輸出的範例。

```

+-----+
+-----+
+
|      Model Name      |
| ecommerce_customer_activity |
+-----+
+-----+
+

```

Schema Name	public
Owner	awsuser
Creation Time	Fri, 17.06.2022 19:02:15
Model State	READY
Training Job Status	
MaxAutoMLJobRuntimeReached	
validation:accuracy	0.991280
Estimated Cost	7.897689
TRAINING DATA:	
Query	SELECT CUSTOMERID, COUNTRY, STOCKCODE, DESCRIPTION, INVOICEDATE, SALES_AMT, NBR_MONTHS_ACTIVE FROM ECOMMERCE_SALES_TRAINING NBR_MONTHS_ACTIVE
Target Column	NBR_MONTHS_ACTIVE
PARAMETERS:	
Model Type	xgboost
Problem Type	MulticlassClassification
Objective	Accuracy
AutoML Job Name	redshiftml-20220617190215268770
Function Name	predict_customer_activity
Function Parameters	customerid country stockcode description
invoicedate sales_amt	
Function Parameter Types	int8 varchar varchar varchar
varchar float8	
IAM Role	default-aws-iam-role

bucket	S3 Bucket	amzn-s3-demo-
	Max Runtime	5400

+-----
+-----
+

步驟 3：執行模型的預測

以下查詢會顯示哪些客戶符合您客戶忠誠度計劃的資格。如果模型預測客戶至少有 7 個月的活躍時間，則模型會為忠誠度計劃選取該客戶。

```
SELECT
  customerid,
  predict_customer_activity(
    customerid,
    country,
    stockcode,
    description,
    invoicedate,
    sales_amt
  ) AS predicted_months_active
FROM
  ecommerce_sales_prediction
WHERE
  predicted_months_active >= 7
GROUP BY
  1,
  2
LIMIT
  10;
```

針對驗證資料執行預測查詢 (選擇性)

針對驗證資料執行下列預測查詢，以查看模型的準確度層級。

```
SELECT
  CAST(SUM(t1.match) AS decimal(7, 2)) AS predicted_matches,
  CAST(SUM(t1.nonmatch) AS decimal(7, 2)) AS predicted_non_matches,
  CAST(SUM(t1.match + t1.nonmatch) AS decimal(7, 2)) AS total_predictions,
  predicted_matches / total_predictions AS pct_accuracy
FROM
```

```
(
  SELECT
    customerid,
    country,
    stockcode,
    description,
    invoicedate,
    sales_amt,
    nbr_months_active,
    predict_customer_activity(
      customerid,
      country,
      stockcode,
      description,
      invoicedate,
      sales_amt
    ) AS predicted_months_active,
    CASE
      WHEN nbr_months_active = predicted_months_active THEN 1
      ELSE 0
    END AS match,
    CASE
      WHEN nbr_months_active <> predicted_months_active THEN 1
      ELSE 0
    END AS nonmatch
  FROM
    ecommerce_sales_validation
)t1;
```

預測有多少客戶錯失加入機會 (選擇性)

下列查詢會比較預測只會在 5 或 6 個月內處於活躍狀態的客戶數量。該模型會預測這些客戶將錯過忠誠度計劃。然後，查詢會將幾乎不會錯過該計劃的數量與預測符合忠誠度計劃資格的數量進行比較。此查詢可用於提供是否要降低忠誠度計劃門檻的決定。您還可以定判斷是否有大量的客戶預計幾乎不會錯過該計劃。然後，您可以鼓勵這些客戶增加他們的活動，以獲得忠誠度計劃會員資格。

```
SELECT
  predict_customer_activity(
    customerid,
    country,
    stockcode,
    description,
    invoicedate,
```

```
        sales_amt
    ) AS predicted_months_active,
    COUNT(customerid)
FROM
    ecommerce_sales_prediction
WHERE
    predicted_months_active BETWEEN 5 AND 6
GROUP BY
    1
ORDER BY
    1 ASC
LIMIT
    10)
UNION
(SELECT
    NULL AS predicted_months_active,
    COUNT (customerid)
FROM
    ecommerce_sales_prediction
WHERE
    predict_customer_activity(
        customerid,
        country,
        stockcode,
        description,
        invoicedate,
        sales_amt
    ) >=7);
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)

- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：建置XGBoost模型

在本教學課程中，您會使用來自 Amazon S3 的資料建立模型，並使用 Amazon Redshift ML 透過該模型執行預測查詢。此XGBoost演算法是漸層增強型樹狀目錄演算法的最佳化實作。XGBoost 處理比其他梯度增強型樹狀演算法更多的資料類型、關係和分佈。您可以使用 XGBoost進行迴歸、二進位分類、多類別分類和排名問題。如需XGBoost演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的[XGBoost演算法](#)。

Amazon Redshift ML CREATE MODEL操作搭配 AUTO OFF選項目前支援XGBoost作為 MODEL_TYPE。您可以根據您的使用案例提供相關資訊 (例如目標和超參數) 做為 CREATE MODEL 命令的一部分。

在本教學課程中，您將使用[鈔票驗證資料集](#)，這是一個二進制分類問題，用於預測給定鈔票是真的還是偽造的。

使用案例範例

您可以使用 Amazon Redshift ML 來解決其他二進制分類問題，例如預測病患是健康狀態還是患有疾病。您也可以預測電子郵件是否為垃圾郵件。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：執行模型的預測

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的[管理設定](#)。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 來執行下列查詢。

下列查詢會建立兩個資料表、從 Amazon S3 載入資料，然後將資料分割為訓練集和測試集。您將使用訓練集來訓練模型並建立預測函數。然後，您將在測試集上測試預測函數。

```
--create training set table
CREATE TABLE banknoteauthentication_train(
    variance FLOAT,
    skewness FLOAT,
    curtosis FLOAT,
    entropy FLOAT,
    class INT
);

--Load into training table
COPY banknoteauthentication_train
FROM
    's3://redshiftbucket-ml-sagemaker/banknote_authentication/train_data/' IAM_ROLE
    default REGION 'us-west-2' IGNOREHEADER 1 CSV;

--create testing set table
CREATE TABLE banknoteauthentication_test(
    variance FLOAT,
    skewness FLOAT,
    curtosis FLOAT,
    entropy FLOAT,
    class INT
);

--Load data into testing table
COPY banknoteauthentication_test
FROM
    's3://redshiftbucket-ml-sagemaker/banknote_authentication/test_data/'
    IAM_ROLE default
    REGION 'us-west-2'
    IGNOREHEADER 1
    CSV;
```

步驟 2：建立機器學習模型

下列查詢會從您在上一個步驟中建立的訓練集，在 Amazon Redshift ML 中建立 XGBoost 模型。將 `amzn-s3-demo-bucket` 取代為您自己的 `S3_BUCKET`，這將存儲您的輸入資料集和其他 Redshift ML 成品。

```
CREATE MODEL model_banknoteauthentication_xgboost_binary
FROM
    banknoteauthentication_train
```



```

TARGET class
FUNCTION func_model_banknoteauthentication_xgboost_binary
IAM_ROLE default
AUTO OFF
MODEL_TYPE xgboost
OBJECTIVE 'binary:logistic'
PREPROCESSORS 'none'
HYPERPARAMETERS DEFAULT
EXCEPT(NUM_ROUND '100')
SETTINGS(S3_BUCKET 'amzn-s3-demo-bucket');

```

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOWMODEL 命令來了解模型何時準備就緒。

使用下列查詢來監控模型訓練進度。

```
SHOW MODEL model_banknoteauthentication_xgboost_binary;
```

如果模型為 READY，SHOWMODEL 操作也會提供 train:error 指標，如下列輸出範例所示。train:error 指標用於測量模型準確度，可測量到小數點後六位。值為 0 是最準確的，而值為 1 是最不準確的。

```

+-----+-----+
|      Model Name      | model_banknoteauthentication_xgboost_binary |
+-----+-----+
| Schema Name         | public                                       | |
| Owner                | awsuser                                     |
| Creation Time       | Tue, 21.06.2022 19:07:35                  |
| Model State         | READY                                       |
| train:error         |                                             | 0.000000 |
| Estimated Cost      |                                             | 0.006197 |
|                     |                                             |          |
| TRAINING DATA:    |                                             |          |
| Query              | SELECT *                                     |          |
|                     | FROM "BANKNOTEAUTHENTICATION_TRAIN"        |          |
| Target Column      | CLASS                                       |          |
|                     |                                             |          |
| PARAMETERS:        |                                             |          |
| Model Type          | xgboost                                     |          |
| Training Job Name   | redshiftml-20220621190735686935-xgboost    |          |
| Function Name       | func_model_banknoteauthentication_xgboost_binary |

```

Function Parameters	variance skewness curtosis entropy	
Function Parameter Types	float8 float8 float8 float8	
IAM Role	default-aws-iam-role	
S3 Bucket	amzn-s3-demo-bucket	
Max Runtime		5400
HYPERPARAMETERS:		
num_round		100
objective	binary:logistic	
+-----+-----+-----+		

步驟 3：執行模型的預測

檢查模型的準確度

下列預測查詢會使用在上一個步驟中建立的預測函數來檢查模型的準確度。在測試集上執行此查詢，可確保模型與訓練集的對應不會過於緊密。這種緊密的對應關係也稱為過度擬合，過度擬合可能會導致模型做出不可靠的預測。

```
WITH predict_data AS (
  SELECT
    class AS label,
    func_model_banknoteauthentication_xgboost_binary (variance, skewness, curtosis,
entropy) AS predicted,
    CASE
      WHEN label IS NULL THEN 0
      ELSE label
    END AS actual,
    CASE
      WHEN actual = predicted THEN 1 :: INT
      ELSE 0 :: INT
    END AS correct
  FROM
    banknoteauthentication_test
),
aggr_data AS (
  SELECT
    SUM(correct) AS num_correct,
    COUNT(*) AS total
  FROM
    predict_data
```

```
)
SELECT
    (num_correct :: FLOAT / total :: FLOAT) AS accuracy
FROM
    aggr_data;
```

預測真鈔和假鈔的數量

以下預測查詢會傳回測試集中真鈔和假鈔的預測數量。

```
WITH predict_data AS (
    SELECT
        func_model_banknoteauthentication_xgboost_binary(variance, skewness, curtosis,
        entropy) AS predicted
    FROM
        banknoteauthentication_test
)
SELECT
    CASE
        WHEN predicted = '0' THEN 'Original banknote'
        WHEN predicted = '1' THEN 'Counterfeit banknote'
        ELSE 'NA'
    END AS banknote_authentication,
    COUNT(1) AS count
FROM
    predict_data
GROUP BY
    1;
```

找到真鈔和假鈔的平均觀察值

下列預測查詢會傳回測試集中預測為真鈔和假鈔的每個鈔票特徵的平均值。

```
WITH predict_data AS (
    SELECT
        func_model_banknoteauthentication_xgboost_binary(variance, skewness, curtosis,
        entropy) AS predicted,
        variance,
        skewness,
        curtosis,
        entropy
    FROM
        banknoteauthentication_test
```

```
)
SELECT
  CASE
    WHEN predicted = '0' THEN 'Original banknote'
    WHEN predicted = '1' THEN 'Counterfeit banknote'
    ELSE 'NA'
  END AS banknote_authentication,
  TRUNC(AVG(variance), 2) AS avg_variance,
  TRUNC(AVG(skewness), 2) AS avg_skewness,
  TRUNC(AVG(kurtosis), 2) AS avg_kurtosis,
  TRUNC(AVG(entropy), 2) AS avg_entropy
FROM
  predict_data
GROUP BY
  1
ORDER BY
  2;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：建置迴歸模型

在本教學中，您會使用 Amazon Redshift ML 建立機器學習迴歸模型，並在模型上執行預測查詢。迴歸模型允許您預測數值結果，例如房屋的價格，或有多少人將使用城市的自行車租賃服務。您可以在 Amazon Redshift 中使用 CREATEMODEL 命令搭配訓練資料。然後，Amazon Redshift ML 會編譯模型，將訓練過的模型匯入 Redshift，並準備 SQL 預測函數。您可以在 Amazon Redshift 的 SQL 查詢中使用預測函數。

在本教學課程中，您將使用 Amazon Redshift ML 建置迴歸模型，以預測在一天中的任何特定時間使用多倫多市自行車共享服務的人數。模型的輸入包括假日和天氣條件。您將使用迴歸模型，因為您想要此問題的數值結果。

您可以使用 CREATEMODEL 命令匯出訓練資料、訓練模型，並將模型作為 SQL 函數在 Amazon Redshift 中提供。使用 CREATEMODEL 操作來指定訓練資料，可以是資料表或 SELECT 陳述式。

使用案例範例

您可以使用 Amazon Redshift ML 解決其他迴歸問題，例如預測客戶的終身價值。您也可以使用 Redshift ML 來預測最有利的價格和產生的產品收入。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：驗證模型

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的[管理設定](#)。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 來執行下列查詢。

1. 您必須建立三個資料表，才能將三個公開資料集載入 Amazon Redshift。資料集是[多倫多自行車乘客資料](#)、[歷史氣象資料](#)和[歷史假期資料](#)。在 Amazon Redshift 查詢編輯器中執行下列查詢，以建立名為 ridership、weather 和 holiday 的資料表。

```
CREATE TABLE IF NOT EXISTS ridership (  
    trip_id INT,  
    trip_duration_seconds INT,  
    trip_start_time timestamp,  
    trip_stop_time timestamp,  
    from_station_name VARCHAR(50),  
    to_station_name VARCHAR(50),  
    from_station_id SMALLINT,  
    to_station_id SMALLINT,  
    user_type VARCHAR(20)
```

```
);

CREATE TABLE IF NOT EXISTS weather (
    longitude_x DECIMAL(5, 2),
    latitude_y DECIMAL(5, 2),
    station_name VARCHAR(20),
    climate_id BIGINT,
    datetime_utc TIMESTAMP,
    weather_year SMALLINT,
    weather_month SMALLINT,
    weather_day SMALLINT,
    time_utc VARCHAR(5),
    temp_c DECIMAL(5, 2),
    temp_flag VARCHAR(1),
    dew_point_temp_c DECIMAL(5, 2),
    dew_point_temp_flag VARCHAR(1),
    rel_hum SMALLINT,
    rel_hum_flag VARCHAR(1),
    precip_amount_mm DECIMAL(5, 2),
    precip_amount_flag VARCHAR(1),
    wind_dir_10s_deg VARCHAR(10),
    wind_dir_flag VARCHAR(1),
    wind_spd_kmh VARCHAR(10),
    wind_spd_flag VARCHAR(1),
    visibility_km VARCHAR(10),
    visibility_flag VARCHAR(1),
    stn_press_kpa DECIMAL(5, 2),
    stn_press_flag VARCHAR(1),
    hmdx SMALLINT,
    hmdx_flag VARCHAR(1),
    wind_chill VARCHAR(10),
    wind_chill_flag VARCHAR(1),
    weather VARCHAR(10)
);

CREATE TABLE IF NOT EXISTS holiday (holiday_date DATE, description VARCHAR(100));
```

2. 下列查詢會將範例資料載入您在上一個步驟中建立的資料表。

```
COPY ridership
FROM
    's3://redshift-ml-bikesharing-data/bike-sharing-data/ridership/'
IAM_ROLE default
FORMAT CSV
```

```
IGNOREHEADER 1
DATEFORMAT 'auto'
TIMEFORMAT 'auto'
REGION 'us-west-2'
gzip;

COPY weather
FROM
  's3://redshift-ml-bikesharing-data/bike-sharing-data/weather/'
IAM_ROLE default
FORMAT csv
IGNOREHEADER 1
DATEFORMAT 'auto'
TIMEFORMAT 'auto'
REGION 'us-west-2'
gzip;

COPY holiday
FROM
  's3://redshift-ml-bikesharing-data/bike-sharing-data/holiday/'
IAM_ROLE default
FORMAT csv
IGNOREHEADER 1
DATEFORMAT 'auto'
TIMEFORMAT 'auto'
REGION 'us-west-2'
gzip;
```

3. 下列查詢會對 `ridership` 和 `weather` 資料集執行轉換，以移除偏差或異常。移除偏差和異常可改善模型準確度。該查詢會透過建立兩個名為 `ridership_view` 和 `weather_view` 的新檢視簡化資料表。

```
CREATE
OR REPLACE VIEW ridership_view AS
SELECT
  trip_time,
  trip_count,
  TO_CHAR(trip_time, 'hh24') :: INT trip_hour,
  TO_CHAR(trip_time, 'dd') :: INT trip_day,
  TO_CHAR(trip_time, 'mm') :: INT trip_month,
  TO_CHAR(trip_time, 'yy') :: INT trip_year,
  TO_CHAR(trip_time, 'q') :: INT trip_quarter,
  TO_CHAR(trip_time, 'w') :: INT trip_month_week,
```

```
TO_CHAR(trip_time, 'd') :: INT trip_week_day
FROM
(
    SELECT
        CASE
            WHEN TRUNC(r.trip_start_time) < '2017-07-01' :: DATE THEN
CONVERT_TIMEZONE(
                'US/Eastern',
                DATE_TRUNC('hour', r.trip_start_time)
            )
            ELSE DATE_TRUNC('hour', r.trip_start_time)
        END trip_time,
        COUNT(1) trip_count
    FROM
        ridership r
    WHERE
        r.trip_duration_seconds BETWEEN 60
        AND 60 * 60 * 24
    GROUP BY
        1
);

CREATE
OR REPLACE VIEW weather_view AS
SELECT
    CONVERT_TIMEZONE(
        'US/Eastern',
        DATE_TRUNC('hour', datetime_utc)
    ) daytime,
    ROUND(AVG(temp_c)) temp_c,
    ROUND(AVG(precip_amount_mm)) precip_amount_mm
FROM
    weather
GROUP BY
    1;
```

4. 下面的查詢會建立一個資料表，該資料表會將 `ridership_view` 和 `weather_view` 中的所有相關輸入屬性結合到 `trip_data` 資料表中。

```
CREATE TABLE trip_data AS
SELECT
    r.trip_time,
    r.trip_count,
```



```

r.trip_hour,
r.trip_day,
r.trip_month,
r.trip_year,
r.trip_quarter,
r.trip_month_week,
r.trip_week_day,
w.temp_c,
w.precip_amount_mm,CASE
    WHEN h.holiday_date IS NOT NULL THEN 1
    WHEN TO_CHAR(r.trip_time, 'D') :: INT IN (1, 7) THEN 1
    ELSE 0
END is_holiday,
ROW_NUMBER() OVER (
    ORDER BY
        RANDOM()
) serial_number
FROM
ridership_view r
JOIN weather_view w ON (r.trip_time = w.daytime)
LEFT OUTER JOIN holiday h ON (TRUNC(r.trip_time) = h.holiday_date);

```

檢視範例資料 (選擇性)

下列查詢會顯示資料表中的項目。您可以執行此操作，以確保資料表已正確製作。

```

SELECT *
FROM trip_data
LIMIT 5;

```

下列為上一個操作的輸出範例。

```

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+
|      trip_time      | trip_count | trip_hour | trip_day | trip_month | trip_year
| trip_quarter | trip_month_week | trip_week_day | temp_c | precip_amount_mm |
is_holiday | serial_number |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+

```

```

| 2017-03-21 22:00:00 |          47 |          22 |          21 |          3 |          17 |
|           1 |          3 |          3 |          1 |          0 |          0 |
|           1 |
| 2018-05-04 01:00:00 |          19 |          1 |          4 |          5 |          18 |
|           2 |          1 |          6 |         12 |          0 |          0 |
|           3 |
| 2018-01-11 10:00:00 |          93 |         10 |         11 |          1 |          18 |
|           1 |          2 |          5 |          9 |          0 |          0 |
|           5 |
| 2017-10-28 04:00:00 |          20 |          4 |         28 |         10 |          17 |
|           4 |          4 |          7 |         11 |          0 |          1 |
|           7 |
| 2017-12-31 21:00:00 |          11 |         21 |         31 |         12 |          17 |
|           4 |          5 |          1 |        -15 |          0 |          1 |
|           9 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+

```

顯示屬性之間的關聯 (選擇性)

確定相關性有助於測量屬性之間的關聯強度。關聯層級可協助您判斷哪些因素會影響您的目標輸出。在此教學課程中，目標輸出為 `trip_count`。

下列查詢會建立或取代 `sp_correlation` 程序。您可以使用名為 `sp_correlation` 的預存程序來顯示 Amazon Redshift 中資料表內屬性之間的關聯性。

```

CREATE OR REPLACE PROCEDURE sp_correlation(source_schema_name in varchar(255),
  source_table_name in varchar(255), target_column_name in varchar(255),
  output_temp_table_name inout varchar(255)) AS $$
DECLARE
  v_sql varchar(max);
  v_generated_sql varchar(max);
  v_source_schema_name varchar(255)=lower(source_schema_name);
  v_source_table_name varchar(255)=lower(source_table_name);
  v_target_column_name varchar(255)=lower(target_column_name);
BEGIN
  EXECUTE 'DROP TABLE IF EXISTS ' || output_temp_table_name;
  v_sql = '
SELECT
  'CREATE temp table ' || output_temp_table_name || ' AS SELECT ' || outer_calculation ||

```

```

    ' FROM (SELECT COUNT(1) number_of_items, SUM('||v_target_column_name||')
sum_target, SUM(POW('||v_target_column_name||',2)) sum_square_target, POW(SUM('||
v_target_column_name||'),2) square_sum_target,'||
inner_calculation||
    ' FROM (SELECT '||
column_name||
    ' FROM '||v_source_table_name||'))'
FROM
(
SELECT
DISTINCT
LISTAGG(outer_calculation,',') OVER () outer_calculation
,LISTAGG(inner_calculation,',') OVER () inner_calculation
,LISTAGG(column_name,',') OVER () column_name
FROM
(
SELECT
CASE WHEN atttypid=16 THEN 'DECODE('||column_name||',true,1,0)' ELSE
column_name END column_name
,atttypid
,'CAST(DECODE(number_of_items * sum_square_'||rn||' - square_sum_'||
rn||',0,null,(number_of_items*sum_target_'||rn||' - sum_target * sum_'||rn||
')/SQRT((number_of_items * sum_square_target - square_sum_target) *
(number_of_items * sum_square_'||rn||
' - square_sum_'||rn||')) AS numeric(5,2)) '||column_name
outer_calculation
,'sum('||column_name||') sum_'||rn||','||
'SUM(trip_count*'||column_name||') sum_target_'||rn||','||
'SUM(POW('||column_name||',2)) sum_square_'||rn||','||
'POW(SUM('||column_name||'),2) square_sum_'||rn inner_calculation
FROM
(
SELECT
row_number() OVER (order by a.attnum) rn
,a.attname::VARCHAR column_name
,a.atttypid
FROM pg_namespace AS n
INNER JOIN pg_class AS c ON n.oid = c.relnamespace
INNER JOIN pg_attribute AS a ON c.oid = a.attrelid
WHERE a.attnum > 0
AND n.nspname = '||v_source_schema_name||'
AND c.relname = '||v_source_table_name||'
AND a.atttypid IN (16,20,21,23,700,701,1700)
)
)

```

```

)
)';
EXECUTE v_sql INTO v_generated_sql;
EXECUTE v_generated_sql;
END;
$$ LANGUAGE plpgsql;

```

下面的查詢顯示目標列、`trip_count` 和我們的資料集中的其他數字屬性之間的相關性。

```

call sp_correlation(
  'public',
  'trip_data',
  'trip_count',
  'tmp_corr_table'
);

SELECT
  *
FROM
  tmp_corr_table;

```

下列範例顯示上一個 `sp_correlation` 操作的輸出。

```

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+
| trip_count | trip_hour | trip_day | trip_month | trip_year | trip_quarter |
| trip_month_week | trip_week_day | temp_c | precip_amount_mm | is_holiday | serial_number |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+
|          1 |          0.32 |          0.01 |          0.18 |          0.12 |          0.18 |
|          0 |          0.02 |          0.53 |          -0.07 |          -0.13 |          0 |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+

```

步驟 2：建立機器學習模型

1. 下列查詢會將您的資料分割為訓練集和驗證集，並指定 80% 的資料集用於訓練，而 20% 的資料集用於驗證。訓練集是 ML 模型的輸入，用於識別模型的最佳演算法。建立模型之後，您可以使用驗證集來驗證模型的準確度。

```
CREATE TABLE training_data AS
SELECT
    trip_count,
    trip_hour,
    trip_day,
    trip_month,
    trip_year,
    trip_quarter,
    trip_month_week,
    trip_week_day,
    temp_c,
    precip_amount_mm,
    is_holiday
FROM
    trip_data
WHERE
    serial_number > (
        SELECT
            COUNT(1) * 0.2
        FROM
            trip_data
    );

CREATE TABLE validation_data AS
SELECT
    trip_count,
    trip_hour,
    trip_day,
    trip_month,
    trip_year,
    trip_quarter,
    trip_month_week,
    trip_week_day,
    temp_c,
    precip_amount_mm,
    is_holiday,
    trip_time
```

```
FROM
  trip_data
WHERE
  serial_number <= (
    SELECT
      COUNT(1) * 0.2
    FROM
      trip_data
  );
```

2. 下列查詢會建立迴歸模型，以預測任何輸入日期和時間的 `trip_count` 值。在下列範例中，將 `amzn-s3-demo-bucket` 取代為您自己的 S3 儲存貯體。

```
CREATE MODEL predict_rental_count
FROM
  training_data TARGET trip_count FUNCTION predict_rental_count
IAM_ROLE default
PROBLEM_TYPE regression
OBJECTIVE 'mse'
SETTINGS (
  s3_bucket 'amzn-s3-demo-bucket',
  s3_garbage_collect off,
  max_runtime 5000
);
```

步驟 3：驗證模型

1. 使用下列查詢來輸出模型的各個層面，並在輸出中尋找均方誤差指標。均方誤差是迴歸問題的典型準確指標。

```
show model predict_rental_count;
```

2. 針對驗證資料執行下列預測查詢，以比較預測的行程計數與實際的行程計數。

```
SELECT
  trip_time,
  actual_count,
  predicted_count,
  (actual_count - predicted_count) difference
FROM
  (
```

```
SELECT
    trip_time,
    trip_count AS actual_count,
    PREDICT_RENTAL_COUNT (
        trip_hour,
        trip_day,
        trip_month,
        trip_year,
        trip_quarter,
        trip_month_week,
        trip_week_day,
        temp_c,
        precip_amount_mm,
        is_holiday
    ) predicted_count
FROM
    validation_data
)
LIMIT
    5;
```

3. 下列查詢會根據驗證資料計算均方誤差和均方根誤差。您可以使用均方誤差和均方根誤差來測量預測數值目標與實際數值答案之間的距離。一個好的模型在這兩個指標中的分數都很低。下列查詢會傳回這兩個指標的值。

```
SELECT
    ROUND(
        AVG(POWER((actual_count - predicted_count), 2)),
        2
    ) mse,
    ROUND(
        SQRT(AVG(POWER((actual_count - predicted_count), 2))),
        2
    ) rmse
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
```

```

        trip_year,
        trip_quarter,
        trip_month_week,
        trip_week_day,
        temp_c,
        precip_amount_mm,
        is_holiday
    ) predicted_count
FROM
    validation_data
);

```

4. 下列查詢會針對 2017-01-01 的每個行程時間計算行程計數中的百分比誤差。查詢會將行程時間從百分比誤差最低的時間排序到百分比誤差最高的時間。

```

SELECT
    trip_time,
    CAST(ABS(((actual_count - predicted_count) / actual_count)) * 100 AS DECIMAL
    (7,2)) AS pct_error
FROM
    (
        SELECT
            trip_time,
            trip_count AS actual_count,
            PREDICT_RENTAL_COUNT (
                trip_hour,
                trip_day,
                trip_month,
                trip_year,
                trip_quarter,
                trip_month_week,
                trip_week_day,
                temp_c,
                precip_amount_mm,
                is_holiday
            ) predicted_count
        FROM
            validation_data
    )
WHERE
    trip_time LIKE '2017-01-01 %:%:%:%%'
ORDER BY
    2 ASC;

```


相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：使用線性學習程式建置迴歸模型

在本教學課程中，您會使用來自 Amazon S3 的資料建立線性學習程式模型，並使用 Amazon Redshift ML 透過該模型執行預測查詢。SageMaker 線性學習器演算法可解決迴歸或多類別分類問題。若要進一步了解迴歸和多類別分類問題，請參閱 Amazon SageMaker 開發人員指南中的[機器學習範例問題類型](#)。在本教學課程中，您需解決迴歸問題。線性學習程式演算法可同時訓練許多模型，並自動判斷最佳化程度最高的模型。您可以在 Amazon Redshift 中使用 CREATEMODEL 操作，這會使用 建立線性學習器模型，SageMaker 並將預測函數傳送至 Amazon Redshift。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的[線性學習器演算法](#)。

您可以使用 CREATEMODEL 命令匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATEMODEL 操作來指定訓練資料，可以是資料表或 SELECT 陳述式。

線性學習程式模型可最佳化連續目標或離散目標。連續目標用於迴歸，而離散變數用於分類。某些方法會僅針對連續目標提供解決方案，例如迴歸方法。與單純的超參數最佳化技術 (例如 Naive Bayes 技術) 相比，線性學習程式演算法的速度更快。單純的最佳化技術會假定每個輸入變數都是獨立的。若要使用線性學習程式演算法，您必須提供代表輸入維度的資料欄，以及表示觀測值的資料列。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的[線性學習器演算法](#)。

在本教學課程中，您將建置一個線性學習程式模型，以預測鮑魚的年齡。您可以在[鮑魚資料集](#)上使用 CREATEMODEL 命令，來判斷鮑魚的實體測量之間的關係。然後，您可以使用該模型來判斷鮑魚的年齡。

使用案例範例

您可以使用線性學習程式和 Amazon Redshift ML 來解決其他迴歸問題，例如預測房屋的價格。您還可以使用 Redshift ML 來預測將使用城市自行車租賃服務的人數。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型
- 步驟 3：驗證模型

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的[管理設定](#)。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 來執行下列查詢。這些查詢會將範例資料載入 Redshift，並將資料分割為訓練集和驗證集。

1. 以下查詢會建立 `abalone_dataset` 資料表。

```
CREATE TABLE abalone_dataset (  
  id INT IDENTITY(1, 1),  
  Sex CHAR(1),  
  Length float,  
  Diameter float,  
  Height float,  
  Whole float,  
  Shucked float,  
  Viscera float,  
  Shell float,  
  Rings integer  
);
```

2. 下列查詢會將 Amazon S3 中[鮑魚資料集](#)的範例資料複製到您先前在 Amazon Redshift 中建立的 `abalone_dataset` 資料表。

```
COPY abalone_dataset
```

```
FROM
  's3://redshift-ml-multiclass/abalone.csv' REGION 'us-east-1' IAM_ROLE default CSV
  IGNOREHEADER 1 NULL AS 'NULL';
```

3. 透過手動分割資料，您將能夠透過配置額外預測集來驗證模型的準確性。下面的查詢會將資料分成兩組。abalone_training 資料表用於訓練，abalone_validation 資料表用於驗證。

```
CREATE TABLE abalone_training as
SELECT
  *
FROM
  abalone_dataset
WHERE
  mod(id, 10) < 8;

CREATE TABLE abalone_validation as
SELECT
  *
FROM
  abalone_dataset
WHERE
  mod(id, 10) >= 8;
```

步驟 2：建立機器學習模型

在此步驟中，您可以使用 CREATEMODEL 陳述式建立具有線性學習器演算法的機器學習模型。

下列查詢會使用 S3 儲存貯體建立具有 CREATEMODEL 操作的線性學習器模型。將 amzn-s3-demo-bucket 取代為您自己的 S3 儲存貯體。

```
CREATE MODEL model_abalone_ring_prediction
FROM
  (
    SELECT
      Sex,
      Length,
      Diameter,
      Height,
      Whole,
      Shucked,
      Viscera,
      Shell,
```

```

Rings AS target_label
FROM
  abalone_training
) TARGET target_label FUNCTION f_abalone_ring_prediction IAM_ROLE default
MODEL_TYPE LINEAR_LEARNER PROBLEM_TYPE REGRESSION OBJECTIVE 'MSE' SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  MAX_RUNTIME 15000
);

```

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOWMODEL 命令來了解模型何時準備就緒。

使用下列查詢來監控模型訓練進度。

```
SHOW MODEL model_abalone_ring_prediction;
```

當模型準備就緒時，先前操作的輸出看起來應與下列範例類似。請注意，輸出會提供 validation:mse 指標，即均方誤差。在下一個步驟中，您將使用均方誤差來驗證模型的準確度。

```

+-----+
+-----+
+
|      Model Name      |
| model_abalone_ring_prediction |
+-----+
+-----+
+
| Schema Name          | public
|
| Owner                 | awsuser
|
| Creation Time         | Thu, 30.06.2022 18:00:10
|
| Model State           | READY
|
| validation:mse        |
|                       | 4.168633 |
| Estimated Cost        |
|                       | 4.291608 |
|
|
|

```

```

| TRAINING DATA: |
|
| Query | SELECT SEX , LENGTH , DIAMETER , HEIGHT , WHOLE ,
SHUCKED , VISCERA , SHELL, RINGS AS TARGET_LABEL |
| | FROM ABALONE_TRAINING |
| Target Column | TARGET_LABEL |
|
|
| PARAMETERS: |
|
| Model Type | linear_learner |
| Problem Type | Regression |
| Objective | MSE |
| AutoML Job Name | redshiftml-20220630180010947843 |
| Function Name | f_abalone_ring_prediction |
| Function Parameters | sex length diameter height whole shucked viscera shell |
| Function Parameter Types | bpchar float8 float8 float8 float8 float8 float8 float8 |
| IAM Role | default-aws-iam-role |
| S3 Bucket | amzn-s3-demo-bucket |
| Max Runtime |
| | 15000 |
+-----+
+-----+
+

```

步驟 3：驗證模型

1. 下列預測查詢會計算均方誤差和均方根誤差，藉此驗證 `abalone_validation` 資料集上模型的準確性。

```

SELECT
    ROUND(AVG(POWER((tgt_label - predicted), 2)), 2) mse,

```

```

ROUND(SQRT(AVG(POWER((tgt_label - predicted), 2))), 2) rmse
FROM
(
    SELECT
        Sex,
        Length,
        Diameter,
        Height,
        Whole,
        Shucked,
        Viscera,
        Shell,
        Rings AS tgt_label,
        f_abalone_ring_prediction(
            Sex,
            Length,
            Diameter,
            Height,
            Whole,
            Shucked,
            Viscera,
            Shell
        ) AS predicted,
        CASE
            WHEN tgt_label = predicted then 1
            ELSE 0
        END AS match,
        CASE
            WHEN tgt_label <> predicted then 1
            ELSE 0
        END AS nonmatch
    FROM
        abalone_validation
) t1;

```

先前查詢的輸出看起來應該像下列範例。平均平方錯誤指標的值應類似於SHOWMODEL操作輸出顯示的validation:mse指標。

```

+-----+-----+
| mse |           rmse           |
+-----+-----+
| 5.1 | 2.2600000000000002 |

```

```
+-----+-----+-----+-----+
```

2. 使用下列查詢，在您的預測函數上執行 EXPLAIN_MODEL 操作。操作會傳回模型可解釋性報告。如需 EXPLAIN_MODEL 操作的詳細資訊，請參閱 Amazon Redshift 資料庫開發人員指南中的 [EXPLAIN_MODEL 函數](#)。

```
SELECT
  EXPLAIN_MODEL ('model_abalone_ring_prediction');
```

下列資訊是先前 EXPLAIN_MODEL 操作產生的模型可解釋性報告範例。每個輸入的值為 Shapley 值。Shapley 值代表每個輸入對模型預測的影響，值較高的輸入對預測的影響較大。在此範例中，值較高的輸入對預測鮑魚的年齡有較大的影響。

```
{
  "explanations": {
    "kernel_shap": {
      "label0": {
        "expected_value" :10.290688514709473,
        "global_shap_values": {
          "diameter" :0.6856910187882492,
          "height" :0.4415323937124035,
          "length" :0.21507476107609084,
          "sex" :0.448611774505744,
          "shell" :1.70426496893776,
          "shucked" :2.1181392924386994,
          "viscera" :0.342220754059912,
          "whole" :0.6711906974084011
        }
      }
    }
  },
  "version" : "1.0"
};
```

3. 針對尚未成熟的鮑魚，使用以下查詢來計算模型做出正確預測的百分比。未成熟的鮑魚有 10 個或更少的環，而正確預測的準確度與實際環數僅差不到一個環。

```
SELECT
  TRUNC(
    SUM(
      CASE
        WHEN ROUND(
```

```
        f_abalone_ring_prediction(
            Sex,
            Length,
            Diameter,
            Height,
            Whole,
            Shucked,
            Viscera,
            Shell
        ),
        0
    ) BETWEEN Rings - 1
    AND Rings + 1 THEN 1
    ELSE 0
END
) / CAST(COUNT(SHELL) AS FLOAT),
4
) AS prediction_pct
FROM
    abalone_validation
WHERE
    Rings <= 10;
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

教學課程：使用線性學習程式建置多類別分類模型

在本教學課程中，您會使用來自 Amazon S3 的資料建立線性學習程式模型，然後使用 Amazon Redshift ML 透過該模型執行預測查詢。SageMaker 線性學習器演算法可解決迴歸或分類問題。若要進一步了解迴歸和多類別分類問題，請參閱 Amazon SageMaker 開發人員指南中的[機器學習範例問題類型](#)。在本教學課程中，您將解決多類別分類問題。線性學習程式演算法可同時訓練許多模型，並自動判斷最佳化程度最高的模型。您可以在 Amazon Redshift 中使用 CREATEMODEL 操作，這會使用建立線性學習器模型，SageMaker 並將預測函數傳送至 Amazon Redshift。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的[線性學習器演算法](#)。

您可以使用 CREATEMODEL 命令匯出訓練資料、訓練模型、匯入模型，以及準備 Amazon Redshift 預測函數。使用 CREATEMODEL 操作來指定訓練資料，可以是資料表或 SELECT 陳述式。

線性學習程式模型可最佳化連續目標或離散目標。連續目標用於迴歸，而離散變數用於分類。某些方法會僅針對連續目標提供解決方案，例如迴歸方法。與單純的超參數最佳化技術 (例如 Naive Bayes 技術) 相比，線性學習程式演算法的速度更快。單純的最佳化技術會假定每個輸入變數都是獨立的。線性學習程式演算法可同時訓練許多模型，並選取最佳化程度最高的模型。類似的演算法是 XGBoost，它結合了一組更簡單和較弱模型的估計值，以進行預測。若要進一步了解 XGBoost，請參閱 Amazon SageMaker 開發人員指南中的[XGBoost 演算法](#)。

若要使用線性學習程式演算法，您必須提供代表輸入維度的資料欄，以及表示觀測值的資料列。如需線性學習器演算法的詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的[線性學習器演算法](#)。

在本教學課程中，您將建置一個線性學習程式模型，以預測指定區域的覆蓋類型。您可以從 UCIMachine Learning 儲存庫的 [Covertypes 資料集](#) 上使用 CREATEMODEL 命令。然後，您可以使用命令建立的預測函數來判斷荒野區域中的覆蓋類型。森林覆蓋類型通常是一個樹種。Redshift ML 將用於建立模型的輸入包括土壤類型、與道路的距離以及荒野區域指定。如需資料集的詳細資訊，請參閱 UCIMachine Learning 儲存庫中的 [Covertypes 資料集](#)。

使用案例範例

您可以使用 Amazon Redshift ML 解決線性學習程式的其他多類別分類問題，例如從影像預測植物的種類。您還可以預測客戶將購買的產品數量。

工作

- 必要條件
- 步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift
- 步驟 2：建立機器學習模型

- 步驟 3：驗證模型

必要條件

為完成此教學課程，您必須完成 Amazon Redshift ML 的[管理設定](#)。

步驟 1：將資料從 Amazon S3 載入到 Amazon Redshift

使用 [Amazon Redshift 查詢編輯器 v2](#) 來執行下列查詢。這些查詢會將範例資料載入 Redshift，並將資料分割為訓練集和驗證集。

1. 以下查詢會建立 covertime_data 資料表。

```
CREATE TABLE public.covertime_data (  
  elevation bigint ENCODE az64,  
  aspect bigint ENCODE az64,  
  slope bigint ENCODE az64,  
  horizontal_distance_to_hydrology bigint ENCODE az64,  
  vertical_distance_to_hydrology bigint ENCODE az64,  
  horizontal_distance_to_roadways bigint ENCODE az64,  
  hillshade_9am bigint ENCODE az64,  
  hillshade_noon bigint ENCODE az64,  
  hillshade_3pm bigint ENCODE az64,  
  horizontal_distance_to_fire_points bigint ENCODE az64,  
  wilderness_area1 bigint ENCODE az64,  
  wilderness_area2 bigint ENCODE az64,  
  wilderness_area3 bigint ENCODE az64,  
  wilderness_area4 bigint ENCODE az64,  
  soil_type1 bigint ENCODE az64,  
  soil_type2 bigint ENCODE az64,  
  soil_type3 bigint ENCODE az64,  
  soil_type4 bigint ENCODE az64,  
  soil_type5 bigint ENCODE az64,  
  soil_type6 bigint ENCODE az64,  
  soil_type7 bigint ENCODE az64,  
  soil_type8 bigint ENCODE az64,  
  soil_type9 bigint ENCODE az64,  
  soil_type10 bigint ENCODE az64,  
  soil_type11 bigint ENCODE az64,  
  soil_type12 bigint ENCODE az64,  
  soil_type13 bigint ENCODE az64,  
  soil_type14 bigint ENCODE az64,  
  soil_type15 bigint ENCODE az64,
```

```
soil_type16 bigint ENCODE az64,  
soil_type17 bigint ENCODE az64,  
soil_type18 bigint ENCODE az64,  
soil_type19 bigint ENCODE az64,  
soil_type20 bigint ENCODE az64,  
soil_type21 bigint ENCODE az64,  
soil_type22 bigint ENCODE az64,  
soil_type23 bigint ENCODE az64,  
soil_type24 bigint ENCODE az64,  
soil_type25 bigint ENCODE az64,  
soil_type26 bigint ENCODE az64,  
soil_type27 bigint ENCODE az64,  
soil_type28 bigint ENCODE az64,  
soil_type29 bigint ENCODE az64,  
soil_type30 bigint ENCODE az64,  
soil_type31 bigint ENCODE az64,  
soil_type32 bigint ENCODE az64,  
soil_type33 bigint ENCODE az64,  
soil_type34 bigint ENCODE az64,  
soil_type35 bigint ENCODE az64,  
soil_type36 bigint ENCODE az64,  
soil_type37 bigint ENCODE az64,  
soil_type38 bigint ENCODE az64,  
soil_type39 bigint ENCODE az64,  
soil_type40 bigint ENCODE az64,  
cover_type bigint ENCODE az64  
) DISTSTYLE AUTO;
```

2. 下列查詢會將 Amazon S3 中 [Covertime 資料集](#) 的範例資料複製到您先前在 Amazon Redshift 中建立的 `covertime_data` 資料表。

```
COPY public.covertime_data  
FROM  
    's3://redshift-ml-multiclass/covtype.data.gz' IAM_ROLE DEFAULT gzip DELIMITER ','  
    REGION 'us-east-1';
```

3. 透過手動分割資料，您將能夠透過配置額外測試集來驗證模型的準確性。下面的查詢會將資料分成三組。`covertime_training` 資料表用於訓練、`covertime_validation` 資料表用於驗證，而 `covertime_test` 資料表用於測試您的模型。您將使用訓練集來訓練模型，並使用驗證集來驗證模型的開發。然後，您可以使用測試集來測試模型的效能，並查看模型與資料集之間是否過度擬合或低度擬合。

```
CREATE TABLE public.covertime_data_prep AS
SELECT
  a.*,
  CAST (random() * 100 AS int) AS data_group_id
FROM
  public.covertime_data a;

--training dataset
CREATE TABLE public.covertime_training as
SELECT
  *
FROM
  public.covertime_data_prep
WHERE
  data_group_id < 80;

--validation dataset
CREATE TABLE public.covertime_validation AS
SELECT
  *
FROM
  public.covertime_data_prep
WHERE
  data_group_id BETWEEN 80
  AND 89;

--test dataset
CREATE TABLE public.covertime_test AS
SELECT
  *
FROM
  public.covertime_data_prep
WHERE
  data_group_id > 89;
```

步驟 2：建立機器學習模型

在此步驟中，您可以使用 `CREATE MODEL` 陳述式建立具有線性學習器演算法的機器學習模型。

下列查詢會使用 S3 儲存貯體建立具有 `CREATE MODEL` 操作的線性學習器模型。將 `amzn-s3-demo-bucket` 取代為您自己的 S3 儲存貯體。

```
CREATE MODEL forest_cover_type_model
FROM
  (
    SELECT
      Elevation,
      Aspect,
      Slope,
      Horizontal_distance_to_hydrology,
      Vertical_distance_to_hydrology,
      Horizontal_distance_to_roadways,
      Hillshade_9am,
      Hillshade_noon,
      Hillshade_3pm,
      Horizontal_Distance_To_Fire_Points,
      Wilderness_Area1,
      Wilderness_Area2,
      Wilderness_Area3,
      Wilderness_Area4,
      soil_type1,
      Soil_Type2,
      Soil_Type3,
      Soil_Type4,
      Soil_Type5,
      Soil_Type6,
      Soil_Type7,
      Soil_Type8,
      Soil_Type9,
      Soil_Type10,
      Soil_Type11,
      Soil_Type12,
      Soil_Type13,
      Soil_Type14,
      Soil_Type15,
      Soil_Type16,
      Soil_Type17,
      Soil_Type18,
      Soil_Type19,
      Soil_Type20,
      Soil_Type21,
      Soil_Type22,
      Soil_Type23,
      Soil_Type24,
      Soil_Type25,
```

```

        Soil_Type26,
        Soil_Type27,
        Soil_Type28,
        Soil_Type29,
        Soil_Type30,
        Soil_Type31,
        Soil_Type32,
        Soil_Type33,
        Soil_Type34,
        Soil_Type36,
        Soil_Type37,
        Soil_Type38,
        Soil_Type39,
        Soil_Type40,
        Cover_type
    from
        public.covertime_training
    ) TARGET cover_type FUNCTION predict_cover_type IAM_ROLE default MODEL_TYPE
    LINEAR_LEARNER PROBLEM_TYPE MULTICLASS_CLASSIFICATION OBJECTIVE 'Accuracy' SETTINGS (
        S3_BUCKET 'amzn-s3-demo-bucket',
        S3_GARBAGE_COLLECT OFF,
        MAX_RUNTIME 15000
    );

```

顯示模型訓練的狀態 (選擇性)

您可以使用 SHOWMODEL 命令來了解模型何時準備就緒。

使用下列查詢來監控模型訓練進度。

```
SHOW MODEL forest_cover_type_model;
```

當模型準備就緒時，先前操作的輸出看起來應與下列範例類似。請注意，輸出會提供 `validation:multiclass_accuracy` 指標，您可以在下列範例的右側檢視該指標。多類別準確度可測量模型正確分類的資料點百分比。在下一個步驟中，您將使用多類別準確度來驗證模型的準確度。

```

+-----+
+-----+
+
|           Key           |

```

Value

|

+-----

+-----

+

| Model Name | forest_cover_type_model

| Schema Name

| public

|

| Owner

| awsuser

|

| Creation Time

| Tue, 12.07.2022 20:24:32

|

Model State	READY
validation:multiclass_accuracy	
Estimated Cost	0.724952
	5.341750

| TRAINING DATA:

```

| Query
| SELECT ELEVATION, ASPECT, SLOPE,
HORIZONTAL_DISTANCE_TO_HYDROLOGY, VERTICAL_DISTANCE_TO_HYDROLOGY,
HORIZONTAL_DISTANCE_TO_ROADWAYS, HILLSHADE_9AM, HILLSHADE_NOON, HILLSHADE_3PM ,
HORIZONTAL_DISTANCE_TO_FIRE_POINTS, WILDERNESS_AREA1, WILDERNESS_AREA2,
WILDERNESS_AREA3, WILDERNESS_AREA4, SOIL_TYPE1, SOIL_TYPE2, SOIL_TYPE3, SOIL_TYPE4,
SOIL_TYPE5, SOIL_TYPE6, SOIL_TYPE7, SOIL_TYPE8, SOIL_TYPE9, SOIL_TYPE10 , SOIL_TYPE11,
SOIL_TYPE12 , SOIL_TYPE13 , SOIL_TYPE14, SOIL_TYPE15, SOIL_TYPE16, SOIL_TYPE17,
SOIL_TYPE18, SOIL_TYPE19, SOIL_TYPE20, SOIL_TYPE21, SOIL_TYPE22, SOIL_TYPE23,
SOIL_TYPE24, SOIL_TYPE25, SOIL_TYPE26, SOIL_TYPE27, SOIL_TYPE28, SOIL_TYPE29,
SOIL_TYPE30, SOIL_TYPE31, SOIL_TYPE32, SOIL_TYPE33, SOIL_TYPE34, SOIL_TYPE36,
SOIL_TYPE37, SOIL_TYPE38, SOIL_TYPE39, SOIL_TYPE40, COVER_TYPE |
| FROM PUBLIC.COVERTYPE_TRAINING

```

| Target Column | COVER_TYPE

PARAMETERS:	
Model Type	linear_learner
Problem Type	MulticlassClassification
Objective	Accuracy

```

|
| AutoML Job Name          | redshiftml-20220712202432187659
|
|
|
| Function Name           | predict_cover_type
|
|
|
| Function Parameters     | elevation aspect slope
horizontal_distance_to_hydrology vertical_distance_to_hydrology
horizontal_distance_to_roadways hillshade_9am hillshade_noon hillshade_3pm
horizontal_distance_to_fire_points wilderness_area1 wilderness_area2 wilderness_area3
wilderness_area4 soil_type1 soil_type2 soil_type3 soil_type4 soil_type5 soil_type6
soil_type7 soil_type8 soil_type9 soil_type10 soil_type11 soil_type12 soil_type13
soil_type14 soil_type15 soil_type16 soil_type17 soil_type18 soil_type19 soil_type20
soil_type21 soil_type22 soil_type23 soil_type24 soil_type25 soil_type26 soil_type27
soil_type28 soil_type29 soil_type30 soil_type31 soil_type32 soil_type33 soil_type34
soil_type36 soil_type37 soil_type38 soil_type39 soil_type40
|
| Function Parameter Types | int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8 int8
int8 int8 int8 int8 int8 int8 int8 int8 int8 int8

```

```
| IAM Role | default-aws-iam-role |
```

```
| S3 Bucket | amzn-s3-demo-bucket |
```

```
| Max Runtime |
```

```
15000 |
```

```
+-----+
+-----+
+
```

步驟 3：驗證模型

1. 下列預測查詢會透過計算多類別準確度來驗證 `covertime_validation` 資料集上模型的準確度。多類別準確度是模型預測正確的百分比。

```
SELECT
```

```
CAST(sum(t1.match) AS decimal(7, 2)) AS predicted_matches,  
CAST(sum(t1.nonmatch) AS decimal(7, 2)) AS predicted_non_matches,  
CAST(sum(t1.match + t1.nonmatch) AS decimal(7, 2)) AS total_predictions,  
predicted_matches / total_predictions AS pct_accuracy  
FROM  
(  
    SELECT  
        Elevation,  
        Aspect,  
        Slope,  
        Horizontal_distance_to_hydrology,  
        Vertical_distance_to_hydrology,  
        Horizontal_distance_to_roadways,  
        Hillshade_9am,  
        Hillshade_noon,  
        Hillshade_3pm,  
        Horizontal_Distance_To_Fire_Points,  
        Wilderness_Area1,  
        Wilderness_Area2,  
        Wilderness_Area3,  
        Wilderness_Area4,  
        soil_type1,  
        Soil_Type2,  
        Soil_Type3,  
        Soil_Type4,  
        Soil_Type5,  
        Soil_Type6,  
        Soil_Type7,  
        Soil_Type8,  
        Soil_Type9,  
        Soil_Type10,  
        Soil_Type11,  
        Soil_Type12,  
        Soil_Type13,  
        Soil_Type14,  
        Soil_Type15,  
        Soil_Type16,  
        Soil_Type17,  
        Soil_Type18,  
        Soil_Type19,  
        Soil_Type20,  
        Soil_Type21,  
        Soil_Type22,  
        Soil_Type23,
```

```
Soil_Type24,  
Soil_Type25,  
Soil_Type26,  
Soil_Type27,  
Soil_Type28,  
Soil_Type29,  
Soil_Type30,  
Soil_Type31,  
Soil_Type32,  
Soil_Type33,  
Soil_Type34,  
Soil_Type36,  
Soil_Type37,  
Soil_Type38,  
Soil_Type39,  
Soil_Type40,  
Cover_type AS actual_cover_type,  
predict_cover_type(  
    Elevation,  
    Aspect,  
    Slope,  
    Horizontal_distance_to_hydrology,  
    Vertical_distance_to_hydrology,  
    Horizontal_distance_to_roadways,  
    Hillshade_9am,  
    Hillshade_noon,  
    Hillshade_3pm,  
    Horizontal_Distance_To_Fire_Points,  
    Wilderness_Area1,  
    Wilderness_Area2,  
    Wilderness_Area3,  
    Wilderness_Area4,  
    soil_type1,  
    Soil_Type2,  
    Soil_Type3,  
    Soil_Type4,  
    Soil_Type5,  
    Soil_Type6,  
    Soil_Type7,  
    Soil_Type8,  
    Soil_Type9,  
    Soil_Type10,  
    Soil_Type11,  
    Soil_Type12,
```

```
Soil_Type13,  
Soil_Type14,  
Soil_Type15,  
Soil_Type16,  
Soil_Type17,  
Soil_Type18,  
Soil_Type19,  
Soil_Type20,  
Soil_Type21,  
Soil_Type22,  
Soil_Type23,  
Soil_Type24,  
Soil_Type25,  
Soil_Type26,  
Soil_Type27,  
Soil_Type28,  
Soil_Type29,  
Soil_Type30,  
Soil_Type31,  
Soil_Type32,  
Soil_Type33,  
Soil_Type34,  
Soil_Type36,  
Soil_Type37,  
Soil_Type38,  
Soil_Type39,  
Soil_Type40  
    ) AS predicted_cover_type,  
CASE  
    WHEN actual_cover_type = predicted_cover_type THEN 1  
    ELSE 0  
END AS match,  
CASE  
    WHEN actual_cover_type <> predicted_cover_type THEN 1  
    ELSE 0  
END AS nonmatch  
FROM  
    public.covertime_validation  
    ) t1;
```

先前查詢的輸出看起來應該像下列範例。多類別準確度指標的值應類似於SHOWMODEL操作輸出顯示的validation:multiclass_accuracy指標。

```
+-----+-----+-----+-----+
| predicted_matches | predicted_non_matches | total_predictions | pct_accuracy |
+-----+-----+-----+-----+
|           41211 |           16324 |           57535 | 0.71627704 |
+-----+-----+-----+-----+
```

2. 下面的查詢會預測 wilderness_area2 中最常見的覆蓋類型。此資料集包括四個荒野區域和七種覆蓋類型。一個荒野區域可以有多种覆蓋類型。

```
SELECT t1. predicted_cover_type, COUNT(*)
FROM
(
SELECT
  Elevation,
  Aspect,
  Slope,
  Horizontal_distance_to_hydrology,
  Vertical_distance_to_hydrology,
  Horizontal_distance_to_roadways,
  Hillshade_9am,
  Hillshade_noon,
  Hillshade_3pm ,
  Horizontal_Distance_To_Fire_Points,
  Wilderness_Area1,
  Wilderness_Area2,
  Wilderness_Area3,
  Wilderness_Area4,
  soil_type1,
  Soil_Type2,
  Soil_Type3,
  Soil_Type4,
  Soil_Type5,
  Soil_Type6,
  Soil_Type7,
  Soil_Type8,
  Soil_Type9,
  Soil_Type10 ,
  Soil_Type11,
  Soil_Type12 ,
  Soil_Type13 ,
  Soil_Type14,
  Soil_Type15,
```



```
Soil_Type16,  
Soil_Type17,  
Soil_Type18,  
Soil_Type19,  
Soil_Type20,  
Soil_Type21,  
Soil_Type22,  
Soil_Type23,  
Soil_Type24,  
Soil_Type25,  
Soil_Type26,  
Soil_Type27,  
Soil_Type28,  
Soil_Type29,  
Soil_Type30,  
Soil_Type31,  
Soil_Type32,  
Soil_Type33,  
Soil_Type34,  
Soil_Type36,  
Soil_Type37,  
Soil_Type38,  
Soil_Type39,  
Soil_Type40,  
predict_cover_type( Elevation,  
Aspect,  
Slope,  
Horizontal_distance_to_hydrology,  
Vertical_distance_to_hydrology,  
Horizontal_distance_to_roadways,  
Hillshade_9am,  
Hillshade_noon,  
Hillshade_3pm ,  
Horizontal_Distance_To_Fire_Points,  
Wilderness_Area1,  
Wilderness_Area2,  
Wilderness_Area3,  
Wilderness_Area4,  
soil_type1,  
Soil_Type2,  
Soil_Type3,  
Soil_Type4,  
Soil_Type5,  
Soil_Type6,
```

```

Soil_Type7,
Soil_Type8,
Soil_Type9,
Soil_Type10,
Soil_Type11,
Soil_Type12,
Soil_Type13,
Soil_Type14,
Soil_Type15,
Soil_Type16,
Soil_Type17,
Soil_Type18,
Soil_Type19,
Soil_Type20,
Soil_Type21,
Soil_Type22,
Soil_Type23,
Soil_Type24,
Soil_Type25,
Soil_Type26,
Soil_Type27,
Soil_Type28,
Soil_Type29,
Soil_Type30,
Soil_Type31,
Soil_Type32,
Soil_Type33,
Soil_Type34,
Soil_Type36,
Soil_Type37,
Soil_Type38,
Soil_Type39,
Soil_Type40) AS predicted_cover_type

```

```

FROM public.covertime_test
WHERE wilderness_area2 = 1)
t1
GROUP BY 1;

```

先前操作的輸出看起來應與下列範例類似。該輸出意味著該模型預測大部分覆蓋是覆蓋類型 1，並且有一些覆蓋類型 2 和 7。

```
+-----+-----+
```

```

| predicted_cover_type | count |
+-----+-----+
|                    2 |    564 |
|                    7 |     97 |
|                    1 |   2309 |
+-----+-----+

```

3. 下列查詢會顯示單一荒野區域中最常見的覆蓋類型。查詢會顯示該覆蓋類型的數量和覆蓋類型的荒野區域。

```

SELECT t1. predicted_cover_type, COUNT(*), wilderness_area
FROM
(
SELECT
  Elevation,
  Aspect,
  Slope,
  Horizontal_distance_to_hydrology,
  Vertical_distance_to_hydrology,
  Horizontal_distance_to_roadways,
  Hillshade_9am,
  Hillshade_noon,
  Hillshade_3pm ,
  Horizontal_Distance_To_Fire_Points,
  Wilderness_Area1,
  Wilderness_Area2,
  Wilderness_Area3,
  Wilderness_Area4,
  soil_type1,
  Soil_Type2,
  Soil_Type3,
  Soil_Type4,
  Soil_Type5,
  Soil_Type6,
  Soil_Type7,
  Soil_Type8,
  Soil_Type9,
  Soil_Type10 ,
  Soil_Type11,
  Soil_Type12 ,
  Soil_Type13 ,
  Soil_Type14,
  Soil_Type15,

```

```
Soil_Type16,  
Soil_Type17,  
Soil_Type18,  
Soil_Type19,  
Soil_Type20,  
Soil_Type21,  
Soil_Type22,  
Soil_Type23,  
Soil_Type24,  
Soil_Type25,  
Soil_Type26,  
Soil_Type27,  
Soil_Type28,  
Soil_Type29,  
Soil_Type30,  
Soil_Type31,  
Soil_Type32,  
Soil_Type33,  
Soil_Type34,  
Soil_Type36,  
Soil_Type37,  
Soil_Type38,  
Soil_Type39,  
Soil_Type40,  
predict_cover_type( Elevation,  
Aspect,  
Slope,  
Horizontal_distance_to_hydrology,  
Vertical_distance_to_hydrology,  
Horizontal_distance_to_roadways,  
Hillshade_9am,  
Hillshade_noon,  
Hillshade_3pm ,  
Horizontal_Distance_To_Fire_Points,  
Wilderness_Area1,  
Wilderness_Area2,  
Wilderness_Area3,  
Wilderness_Area4,  
soil_type1,  
Soil_Type2,  
Soil_Type3,  
Soil_Type4,  
Soil_Type5,  
Soil_Type6,
```

```
Soil_Type7,  
Soil_Type8,  
Soil_Type9,  
Soil_Type10,  
Soil_Type11,  
Soil_Type12,  
Soil_Type13,  
Soil_Type14,  
Soil_Type15,  
Soil_Type16,  
Soil_Type17,  
Soil_Type18,  
Soil_Type19,  
Soil_Type20,  
Soil_Type21,  
Soil_Type22,  
Soil_Type23,  
Soil_Type24,  
Soil_Type25,  
Soil_Type26,  
Soil_Type27,  
Soil_Type28,  
Soil_Type29,  
Soil_Type30,  
Soil_Type31,  
Soil_Type32,  
Soil_Type33,  
Soil_Type34,  
Soil_Type36,  
Soil_Type37,  
Soil_Type38,  
Soil_Type39,  
Soil_Type40) AS predicted_cover_type,  
CASE WHEN Wilderness_Area1 = 1 THEN 1  
      WHEN Wilderness_Area2 = 1 THEN 2  
      WHEN Wilderness_Area3 = 1 THEN 3  
      WHEN Wilderness_Area4 = 1 THEN 4  
      ELSE 0  
END AS wilderness_area  
  
FROM public.covertime_test)  
t1  
GROUP BY 1, 3  
ORDER BY 2 DESC
```

```
LIMIT 1;
```

先前操作的輸出看起來應與下列範例類似。

```
+-----+-----+-----+
| predicted_cover_type | count | wilderness_area |
+-----+-----+-----+
|                2 | 15738 |                1 |
+-----+-----+-----+
```

相關主題

如需 Amazon Redshift ML 的相關資訊，請參閱下列文件：

- [使用 Amazon Redshift ML 的成本](#)
- [CREATE MODEL 操作](#)
- [EXPLAIN_MODEL 函數](#)

如需機器學習的相關資訊，請參閱下列文件：

- [機器學習概述](#)
- [適用於新手和專家的機器學習](#)
- [什麼是機器學習預測的公平性和模型解釋性？](#)

Amazon Redshift ML 與 Amazon Bedrock 整合

本節說明如何將 Amazon Redshift ML 整合與 Amazon Bedrock 搭配使用。透過此功能，您可以使用叫用 Amazon Bedrock 模型 SQL，而且您可以使用來自 Amazon Redshift 資料倉儲的資料來建置生成性 AI 應用程式，例如文字產生、情緒分析或翻譯。

主題

- [建立或更新 Amazon Redshift ML 與 Amazon Bedrock 整合 IAM 的角色](#)
- [建立 Amazon Redshift ML 與 Amazon Bedrock 整合的外部模型](#)
- [使用外部模型與 Amazon Bedrock 進行 Amazon Redshift ML 整合](#)
- [Amazon Redshift ML 與 Amazon Bedrock 整合的提示工程](#)

建立或更新 Amazon Redshift ML 與 Amazon Bedrock 整合IAM的角色

本節示範如何建立IAM角色以與 Amazon Bedrock 搭配使用 Amazon Redshift ML 整合。

將下列政策新增至您與 Amazon Redshift ML 整合 Amazon Bedrock 搭配使用IAM的角色：

- AmazonBedrockFullAccess

若要允許 Amazon Redshift 擔任與其他 服務互動的角色，請將下列信任政策新增至該IAM角色：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "redshift.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

如果叢集或命名空間位於 中VPC，請遵循 中的指示[針對 Amazon Redshift ML 管理進行叢集和組態設定](#)。

如果您需要更嚴格的政策，您可以建立僅包含下列頁面中指定 Amazon Bedrock 許可的政策：

- [針對 Amazon Redshift ML 管理進行叢集和組態設定](#)
- [使用 Amazon Redshift 機器學習 \(ML \) 所需的許可](#)

如需建立IAM角色的相關資訊，請參閱 AWS Identity and Access Management 使用者指南 中的[IAM角色建立](#)。

建立 Amazon Redshift ML 與 Amazon Bedrock 整合的外部模型

本節說明如何在 Amazon Redshift 資料倉儲中建立外部模型，以用作 Amazon Bedrock 的介面。

若要從 Amazon Redshift 叫用 Amazon Bedrock 模型，您必須先執行 CREATE EXTERNAL MODEL 命令。此命令會在資料庫中建立外部模型物件，以及您用來使用 Amazon Bedrock 產生文字內容的相關使用者函數。

下列程式碼範例顯示基本 CREATE EXTERNAL MODEL 命令：

```
CREATE EXTERNAL MODEL llm_claude
FUNCTION llm_claude_func
IAM_ROLE '<IAM role arn>'
MODEL_TYPE BEDROCK
SETTINGS (
  MODEL_ID 'anthropic.claude-v2:1',
  PROMPT 'Summarize the following text:');
```

此 CREATE EXTERNAL MODEL 命令具有與 Amazon Bedrock 統一且一致的介面，適用於所有支援訊息的基礎模型（FM）。這是使用 CREATE EXTERNAL MODEL 命令或明確指定請求類型為時的預設選項 UNIFIED。如需詳細資訊，請參閱 Amazon Bedrock [API 文件中的 Converse](#) 文件。API

如果 FM 不支援訊息，則必須將 request_type 設定設為 RAW。將 request_type 設定為時 RAW，您必須根據選取的 FM，建構使用推論函數時傳送至 Amazon Bedrock 的請求。

CREATE EXTERNAL MODEL 命令的 PROMPT 參數是靜態提示。如果您需要應用程式的動態提示，則必須在使用推論函數時指定它。如需詳細資訊，請參閱 [Amazon Redshift ML 與 Amazon Bedrock 整合的提示工程](#) 下列內容。

如需 CREATE EXTERNAL MODEL 陳述式及其參數和設定的詳細資訊，請參閱 [CREATE EXTERNAL MODEL](#)。

使用外部模型與 Amazon Bedrock 進行 Amazon Redshift ML 整合

本節說明如何叫用外部模型來產生文字以回應提供的提示。若要叫用外部模型，請使用您透過建立的推論函數 CREATE EXTERNAL MODEL。

主題

- [使用 UNIFIED 請求類型模型進行推論](#)
- [使用 RAW 請求類型模型進行推論](#)
- [推論函數作為領導者限定函數](#)
- [推論函數用量備註](#)

使用UNIFIED請求類型模型進行推論

具有請求類型的模型的推論函數UNIFIED具有下列三個參數，這些參數會依序傳遞至函數：

- 輸入文字（必要）：此參數指定 Amazon Redshift 傳遞給 Amazon Bedrock 的輸入文字。
- 推論組態和其他模型請求欄位（選用）：Amazon Redshift 會將這些參數傳遞給 Converse 模型的對應參數API。

下列程式碼範例示範如何使用UNIFIED類型推論函數：

```
SELECT llm_claude_func(input_text, object('temperature', 0.7, 'maxtokens', 500))
FROM some_data;
```

使用RAW請求類型模型進行推論

具有請求類型的模型的推論函數RAW只有一個資料類型的參數SUPER。此參數的語法取決於使用的 Amazon Bedrock 模型。

下列程式碼範例示範如何使用RAW類型推論函數：

```
SELECT llm_titan_func(
  object(
    "inputText", "Summarize the following text: " | input_text,
    "textGenerationConfig", object("temperature", 0.5, "maxTokenCount", 500)
  )
)
FROM some_data;
```

推論函數作為領導者限定函數

當使用 Amazon Bedrock 模型的查詢未參考任何資料表時，推論函數可以作為領導者節點限定函數執行。如果您想要快速提出問題LLM，這會很有幫助。

下列程式碼範例示範如何使用僅限領導者的推論函數：

```
SELECT general_titan_llm_func('Summarize the benefits of LLM on data analytics in 100 words');
```

推論函數用量備註

搭配 Amazon Redshift ML 與 Amazon Bedrock 整合使用推論函數時，請注意下列事項：

- 所有 Amazon Bedrock 模型的參數名稱區分大小寫。如果您的參數與模型所需的參數不相符，Amazon Bedrock 可能會安靜地忽略它們。
- 推論查詢的輸送量受到不同區域中 Amazon Bedrock 所提供不同模型的執行期配額限制。如需詳細資訊，請參閱 [Amazon Bedrock 使用者指南 中的 Quotas for Amazon Bedrock](#)。
- 如果您需要保證且一致的輸送量，請考慮從 Amazon Bedrock 取得所需模型的佈建輸送量。如需詳細資訊，請參閱 Amazon [Bedrock 使用者指南中的使用 Amazon Bedrock 中的佈建輸送量增加模型調用容量](#)。
- 具有大量資料的推論查詢可能會遇到限流例外狀況。這是因為 Amazon Bedrock 的執行時間配額有限。Amazon Redshift 會多次重試請求，但查詢仍會受到限制，因為未佈建模型的輸送量可能有所不同。

Amazon Redshift ML 與 Amazon Bedrock 整合的提示工程

本節說明如何將靜態提示與外部模型搭配使用。

若要使用外部模型的靜態字首和字尾提示，請使用 CREATE EXTERNAL MODEL 陳述式的 PROMPT 和 SUFFIX 參數提供它們。這些提示會使用外部模型新增至每個查詢。

下列範例示範如何將字首和尾碼提示新增至外部模型：

```
CREATE EXTERNAL MODEL llm_claude
FUNCTION llm_claude_func
IAM_ROLE '<IAM role arn>'
MODEL_TYPE BEDROCK
SETTINGS (
  MODEL_ID 'anthropic.claude-v2:1',
  PROMPT 'Summarize the following text:',
  SUFFIX 'Respond in an analytic tone');
```

若要使用動態提示，您可以在函數輸入中串連推論函數，以便在使用推論函數時提供它們。下列範例示範如何搭配推論函數使用動態提示：

```
SELECT llm_claude_func('Summarize the following review:' | input_text | 'The review
should have formal tone.')
```

```
FROM some_data
```

查詢效能調校

Amazon Redshift 會根據結構化查詢語言 (SQL) 使用查詢，與系統中的資料和物件互動。資料處理語言 (DML) 是您SQL用來檢視、新增、變更和刪除資料的子集。資料定義語言 (DDL) 是您用來新增、變更和刪除資料庫物件SQL的子集，例如資料表和檢視。

設定系統後，您通常會使用DML最多，特別是用於擷取和檢視資料的 [SELECT](#) 命令。若要在 Amazon Redshift 中寫入有效的資料擷取查詢，請熟悉 [中概述的提示SELECT](#)並套用這些提示[設計資料表的 Amazon Redshift 最佳實務](#)，以最大限度地提高查詢效率。

若要了解 Amazon Redshift 如何處理查詢，請使用[查詢處理](#)和[查詢分析和改善](#)小節。然後您可以套用此資訊結合診斷工具來識別和消除查詢效能中的問題。

若要識別和解決使用 Amazon Redshift 查詢時可能遇到的部分最常見和最嚴重的問題，請使用[查詢疑難排解](#)小節。

主題

- [查詢處理](#)
- [查詢分析和改善](#)
- [查詢疑難排解](#)

查詢處理

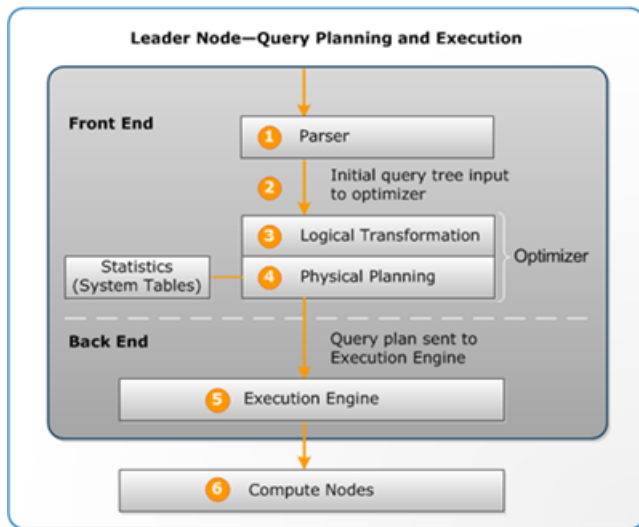
Amazon Redshift 透過剖析器和最佳化工具路由提交的SQL查詢，以開發查詢計劃。執行引擎接著會將查詢計畫轉譯為程式碼，並將該程式碼傳送至運算節點以供執行。

主題

- [查詢計劃和執行工作流程](#)
- [建立和解譯查詢計劃](#)
- [檢閱查詢計劃步驟](#)
- [影響查詢效能的因素](#)

查詢計劃和執行工作流程

下表提供查詢計畫和執行工作流程的整體檢視。



查詢計畫和執行工作流程會遵循這些步驟：

1. 領導節點會收到查詢並剖析 SQL。
2. 剖析器會產生初始查詢樹狀目錄，它是原始查詢的邏輯呈現。然後，Amazon Redshift 會將此查詢樹狀目錄輸入至查詢最佳化器。
3. 最佳化器會進行評估，並且在必要時重新編寫查詢以最佳化其效率。此程序有時會造成建立多個相關的查詢來取代單一查詢。
4. 最佳化器會產生一個查詢計畫 (或數個，如果先前的步驟產生多個查詢) 以利用最佳效能執行。查詢計畫會指定執行選項，例如：聯結類型、聯結順序、彙整選項和資料配送需求。

您可以使用 [EXPLAIN](#) 命令來檢視查詢計畫。查詢計畫為用於分析和調校複雜查詢的基礎工具。如需詳細資訊，請參閱[建立和解譯查詢計畫](#)。

5. 執行引擎會將查詢計畫轉譯為步驟、區段和串流：

步驟

每個步驟為查詢執行期間所需的個別操作。您可以結合步驟，以允許運算節點執行查詢、聯結或其他資料庫操作。

區段

可以透過單一程序完成的數個步驟組合，也是可由運算節點配量執行的最小的編譯單位。配量是 Amazon Redshift 中平行處理的單位。串流中的區段會平行執行。

串流

要透過可用運算節點配量打包的區段的集合。

執行引擎會根據步驟、區段和串流產生編譯的程式碼。相較於轉譯的程式碼，編譯的程式碼執行得更快速，並且使用較少運算容量。然後將此編譯的程式碼播送至運算節點。

Note

設定查詢的基準時，應該一律比較查詢的第二個執行的時間，因為第一個執行時間包含編譯程式碼的間接。如需詳細資訊，請參閱[影響查詢效能的因素](#)。

6. 運算節點配量會平行執行查詢區段。隨著此程序，Amazon Redshift 會利用最佳化的網路通訊、記憶體和磁碟管理，在查詢計畫步驟之間傳遞中繼結果。這也有助於加快查詢執行速度。

步驟 5 和 6 會對每個串流發生一次。引擎會為一個串流建立可執行的區段，並將它們傳送至運算節點。當該串流的區段完成時，引擎會產生下一個串流的區段。以此方式，引擎可以分析前一個串流中發生的動作 (例如，操作是否基於磁碟) 以影響下一個串流中區段的產生。

當運算節點完成時，它們會將查詢結果傳回至領導者節點供最終處理。領導者節點會將資料合併至單一結果集，並解決任何需要的排序或彙整。然後領導者節點會將結果傳回至用戶端。

Note

在查詢執行期間，運算節點可能會在必要時傳回一些資料至領導者節點。例如，如果您有子查詢具有子LIMIT句，則限制會套用至領導節點，然後再將資料重新分佈到叢集以進行進一步處理。

建立和解譯查詢計畫

您可以使用查詢計畫來取得執行查詢所需的個別操作的相關資訊。使用查詢計畫之前，建議您先了解 Amazon Redshift 如何處理處理中的查詢和建立查詢計畫。如需詳細資訊，請參閱[查詢計畫和執行工作流程](#)。

若要建立查詢計畫，請執行 [EXPLAIN](#) 命令，後面接著實際的查詢文字。查詢計畫可提供下列資訊：

- 執行引擎將執行的操作，請由下至上閱讀結果。
- 每個操作會執行的步驟類型。
- 每個操作中使用的資料表和資料欄。
- 每個操作中處理的資料量，就資料列的數量和資料寬度 (位元組) 而言。

- 操作的相對成本。成本是一種測量方式，會比較計畫內步驟的相對執行時間。成本不會提供實際執行時間或記憶體消耗的任何精確資訊，也不會提供執行計畫之間有意義的比較。但它可提供您查詢中耗費最多資源的操作的指示。

EXPLAIN 命令實際上不會執行查詢。它只會顯示如果查詢是在目前操作條件下執行時，Amazon Redshift 會執行的計畫。如果您變更資料表的結構描述或資料，並再次執行 [ANALYZE](#) 以更新統計資訊中繼資料，查詢計畫可能不同。

的查詢計畫輸出EXPLAIN是查詢執行的簡化、高階檢視。它不會說明平行查詢處理的詳細資訊。若要查看詳細資訊，請執行查詢本身，然後從 SVL_QUERY_SUMMARY 或 SVL_QUERY_REPORT 檢視取得查詢摘要資訊。如需使用這些檢視的相關資訊，請參閱[分析查詢摘要](#)。

下列範例顯示EVENT資料表上簡單 GROUP BY 查詢的EXPLAIN輸出：

```
explain select eventname, count(*) from event group by eventname;
```

```
QUERY PLAN
```

```
-----  
XN HashAggregate (cost=131.97..133.41 rows=576 width=17)  
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=17)
```

EXPLAIN 會針對每個操作傳回下列指標：

成本

用在計畫內比較操作的相對值。成本由兩個小數值組成，以兩個句點分隔，例如 `cost=131.97..133.41`。第一個值，在此情況下的 131.97，提供傳回此操作第一個資料列的相對成本。第二個值，在此情況下的 133.41，提供完成操作的相對成本。查詢計畫中的成本會在您讀取計畫時累積，因此此範例（131.97..133.41）中的 HashAggregate 成本包含其下方的 Seq 掃描成本（0.00..87.98）。

資料列

要傳回的估計資料列數量。在此範例中，掃描預期傳回 8798 個資料列。HashAggregate 運算子會自行傳回 576 列（從結果集中捨棄重複的事件名稱之後）。

Note

資料列估算是根據 ANALYZE 命令產生的可用統計資料。如果最近 ANALYZE 尚未執行，則預估值較不可靠。

寬度

估計的平均資料列寬度 (位元組)。在此範例中，平均資料列的寬度預期為 17 個位元組。

EXPLAIN 運算子

本節簡短說明您在EXPLAIN輸出中最常看到的運算子。如需運算子的完整清單，請參閱 SQL命令一節[EXPLAIN](#)中的。

循序掃描運算子

循序掃描運算子 (Seq Scan) 指出資料表掃描。序列掃描會從頭到尾依序掃描資料表中的每個資料欄，並評估每一列的查詢限制 (子WHERE句中)。

聯結運算子

Amazon Redshift 會根據要聯結資料表的實體設計、聯結所需的位置資料，以及查詢本身的特定需求來選取聯結運算子。

- 巢狀迴路

最差的最佳聯結，巢狀迴路主要用於交叉聯結 (笛卡兒乘積) 和一些對等聯結中。

- 雜湊聯結和雜湊

雜湊聯結和雜湊通常會較巢狀迴路聯結更快速，用於內部聯結和左右外部聯結。當聯結資料表中的聯結資料欄不是散發索引鍵也不是排序索引鍵時，會使用這些運算子。雜湊運算子會為聯結中的內部資料表建立雜湊表；雜湊聯結運算子會讀取外部資料表、雜湊聯結資料欄，並在內部雜湊表中尋找相符項目。

- 合併聯結

合併聯結通常是最快速的聯結，用於內部聯結和外部聯結。合併聯結不會用於完整聯結。當聯結資料表中的聯結資料欄為散發索引鍵也是排序索引鍵，並且少於 20% 的聯結資料表未排序時，會使用此運算子。它會依序讀取兩個排序的資料表，並尋找相符的資料列。若要檢視未排序資料列的百分比，請查詢 [SVV_TABLE_INFO](#) 系統資料表。

- 空間聯結

通常基於空間資料的鄰近程度快速聯結，用於 GEOMETRY 和 GEOGRAPHY 資料類型。

彙整運算子

查詢計劃在涉及彙總函數和 GROUP BY 操作的查詢中使用下列運算子。

- Aggregate

純量彙總函數的運算子，例如 AVG和 SUM。

- HashAggregate

未排序的分組彙整函數的運算子。

- GroupAggregate

排序的分組彙整函數的運算子。

排序運算子

查詢計畫會在查詢必須排序或合併結果集時使用下列運算子。

- 排序

評估 ORDER BY 子句和其他排序操作，例如UNION查詢和聯結、SELECTDISTINCT查詢和視窗函數所需的排序。

- 合併

根據衍生自平行操作中繼排序的結果，產生最終排序的結果。

UNION、 INTERSECT和 EXCEPT運算子

查詢計劃針對涉及使用 UNION、 INTERSECT和 設定操作的查詢使用下列運算子EXCEPT。

- Subquery

用於執行UNION查詢。

- Hash Intersect Distinct

用於執行INTERSECT查詢。

- SetOp 除了

用於執行 EXCEPT (或 MINUS) 查詢。

其他運算子

下列運算子也會經常出現在例行查詢的EXPLAIN輸出中。

- 唯一
移除SELECTDISTINCT查詢和UNION查詢的重複項目。
- 限制
處理LIMIT子句。
- 視窗
執行視窗函數。
- 結果
執行未牽涉任何資料表存取的純量函數。
- Subplan
用於特定子查詢。
- 網路
將中繼結果傳送至領導者節點供進一步處理。
- Materialize
儲存資料列以輸入至巢狀迴路聯結和一些合併聯結。

在 中加入 EXPLAIN

取決於查詢和基礎資料表的結構，查詢最佳化器會使用不同的聯結類型來擷取資料表資料。EXPLAIN輸出會參考聯結類型、使用的資料表，以及資料表資料在叢集間的分佈方式，以描述查詢的處理方式。

聯結類型範例

下列範例顯示查詢最佳化器可以使用的不同聯結類型。查詢計畫中使用的聯結類型取決於所牽涉資料表的實體設計。

範例：雜湊聯結兩個資料表

下列查詢會在CATID欄CATEGORY上聯結EVENT和。CATID是的分佈和排序金鑰，CATEGORY但不是的分佈和排序金鑰EVENT。雜湊聯結會使用EVENT作為外部資料表和CATEGORY內部資料表

來執行。因為 CATEGORY 是較小的資料表，規劃器會在查詢處理期間使用 DS_BCAST_ 將其複本廣播至運算節點 INNER。此範例中的聯結成本可說明計畫的多數累積成本。

```
explain select * from category, event where category.catid=event.catid;

          QUERY PLAN
-----
XN Hash Join DS_BCAST_INNER (cost=0.14..6600286.07 rows=8798 width=84)
  Hash Cond: ("outer".catid = "inner".catid)
    -> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=35)
    -> XN Hash (cost=0.11..0.11 rows=11 width=49)
        -> XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
```

Note

EXPLAIN 輸出中運算子的對齊縮排有時表示這些操作彼此不依賴，並且可以平行啟動。在上述範例中，雖然 EVENT 資料表上的掃描和雜湊操作已對齊，但 EVENT 掃描必須等到雜湊操作完全完成。

範例：合併聯結兩個資料表

下列查詢也會使用 SELECT *，但會聯結 SALES 和 LISTING LISTID 欄，其中 LISTID 已設定為兩個資料表的分佈和排序金鑰。已選擇合併聯結，且聯結不需要重新分發資料 (DS_DIST_NONE)。

```
explain select * from sales, listing where sales.listid = listing.listid;

          QUERY PLAN
-----
XN Merge Join DS_DIST_NONE (cost=0.00..6285.93 rows=172456 width=97)
  Merge Cond: ("outer".listid = "inner".listid)
    -> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497 width=44)
    -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=53)
```

下列範例示範相同查詢內不同類型的聯結。如上一個範例所示，SALES 和 LISTING 合併已聯結，但第三個資料表 EVENT 必須與合併聯結的結果結合雜湊。重申，雜湊聯結會衍生播送成本。

```
explain select * from sales, listing, event
where sales.listid = listing.listid and sales.eventid = event.eventid;

          QUERY PLAN
-----
```

```

XN Hash Join DS_BCAST_INNER (cost=109.98..3871130276.17 rows=172456 width=132)
  Hash Cond: ("outer".eventid = "inner".eventid)
    -> XN Merge Join DS_DIST_NONE (cost=0.00..6285.93 rows=172456 width=97)
      Merge Cond: ("outer".listid = "inner".listid)
        -> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497 width=44)
        -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=53)
    -> XN Hash (cost=87.98..87.98 rows=8798 width=35)
      -> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=35)

```

範例：聯結、彙整和排序

下列查詢會執行 SALES和 EVENT 資料表的雜湊聯結，接著會彙總和排序操作，以說明分組SUM函數和 ORDER BY 子句。初始的排序運算子會在運算節點上平行執行。然後 Network 運算子會傳送結果至領導者節點，在其中，Merge 運算子會產生最終排序的結果。

```

explain select eventname, sum(pricepaid) from sales, event
where sales.eventid=event.eventid group by eventname
order by 2 desc;

                                QUERY PLAN
-----
XN Merge (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
  Merge Key: sum(sales.pricepaid)
    -> XN Network (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
      Send to leader
        -> XN Sort (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
          Sort Key: sum(sales.pricepaid)
            -> XN HashAggregate (cost=2815366577.07..2815366578.51 rows=576
width=27)
              -> XN Hash Join DS_BCAST_INNER (cost=109.98..2815365714.80
rows=172456 width=27)
                Hash Cond: ("outer".eventid = "inner".eventid)
                  -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456
width=14)
                    -> XN Hash (cost=87.98..87.98 rows=8798 width=21)
                      -> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=21)

```

資料重新分佈

聯結的EXPLAIN輸出也會指定資料在叢集周圍移動的方式，以促進聯結。資料可以透過播送或重新配送來移動。在播送中，來自聯結一端的資料值會從每個運算節點複製到每個其他運算節點，使得每個運算節點最終有完整的資料副本。在重新配送中，參與的資料值是從其目前的配量傳送至新配量（可能在

不同的節點上)。資料一般會重新配送以符合參與聯結的其他資料表的散發索引鍵 (如果該散發索引鍵是其中一個聯結資料欄)。如果任何資料表都不具備其中一個聯結資料欄上的散發索引鍵，則會配送兩個資料表，或是將內部資料表播送至每個節點。

EXPLAIN 輸出也會參考內部和外部資料表。會先掃描內部資料表，並出現在較接近查詢計畫底端的位置。內部資料表為對其探測相符項目的資料表。它通常位於記憶體中，且通常是雜湊的來源資料表，如有可能，會是要聯結的兩個資料表中較小的那個。外部資料表為要對內部資料表比對的資料列的來源。通常是從磁碟讀取。查詢最佳化工具會根據ANALYZE命令最新執行的資料庫統計資料，選擇內部和外部資料表。查詢子FROM句中的資料表順序不會決定哪個資料表是內部資料表，哪個是外部資料表。

請在查詢計畫中使用下列屬性來識別資料將移動的方式，以輔助查詢：

- DS_BCAST_INNER

將整個內部資料表的副本播送至所有運算節點。

- DS_DIST_ALL_NONE

不需要重新分發，因為內部資料表已使用 DISTSTYLE 分發給每個節點ALL。

- DS_DIST_NONE

不重新配送任何資料表。共置聯結可行，因為會聯結對應的配量，而不需在節點間移動資料。

- DS_DIST_INNER

重新配送內部資料表。

- DS_DIST_OUTER

重新配送外部資料表。

- DS_DIST_ALL_INNER

整個內部資料表會重新分發至單一切片，因為外部資料表使用 DISTSTYLE ALL。

- DS_DIST_BOTH

重新配送這兩個資料表。

檢閱查詢計劃步驟

您可以執行 EXPLAIN命令來查看查詢計劃中的步驟。下列範例顯示SQL查詢並說明輸出。您可以由下至上閱讀查詢計畫，以查看用於執行查詢的每個邏輯操作。如需詳細資訊，請參閱[建立和解譯查詢計劃](#)。

```

explain
select eventname, sum(pricepaid) from sales, event
where sales.eventid = event.eventid
group by eventname
order by 2 desc;

```

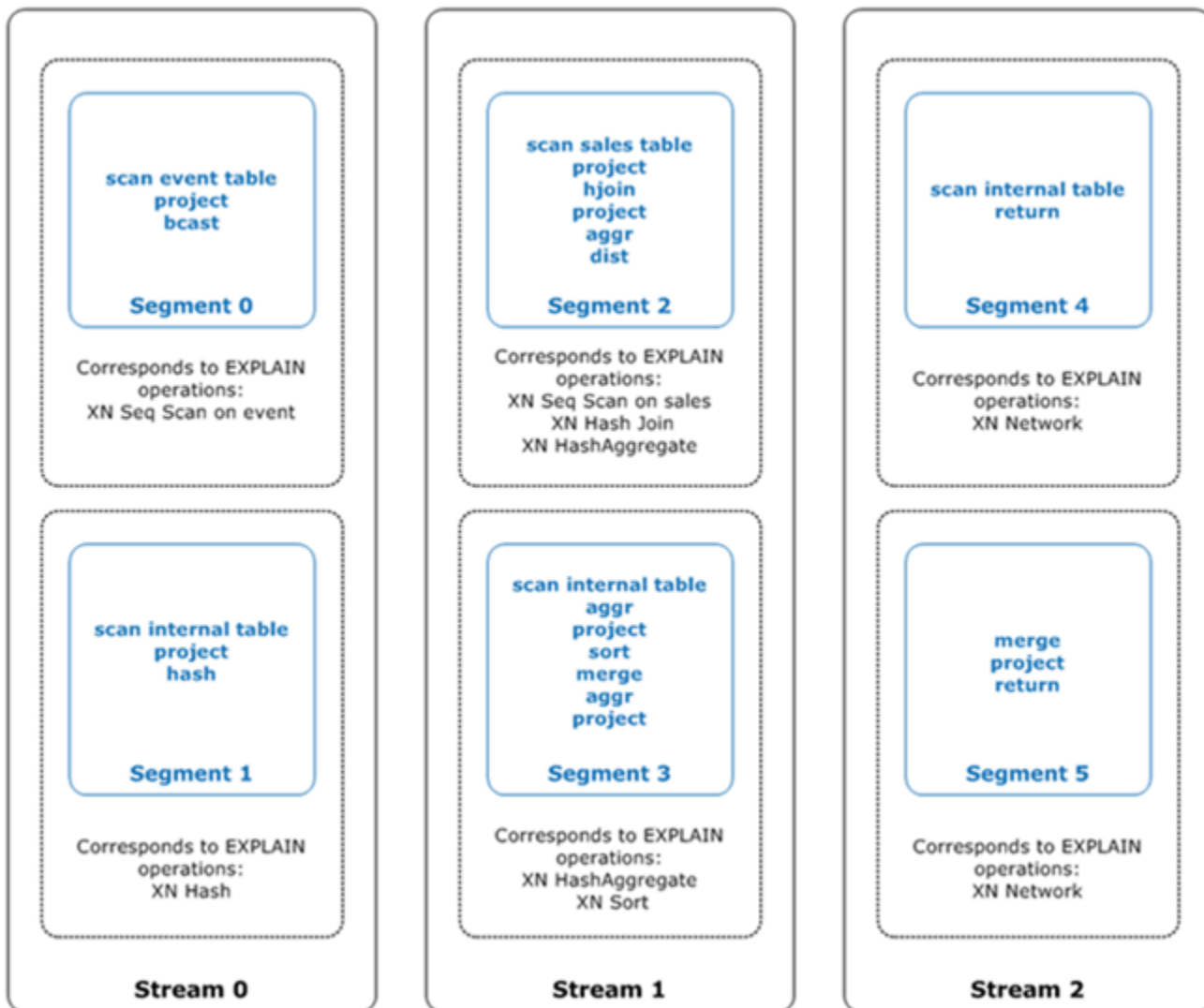
```

XN Merge (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
  Merge Key: sum(sales.pricepaid)
  -> XN Network (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
    Send to leader
    -> XN Sort (cost=1002815366604.92..1002815366606.36 rows=576 width=27)
      Sort Key: sum(sales.pricepaid)
      -> XN HashAggregate (cost=2815366577.07..2815366578.51 rows=576
width=27)
        -> XN Hash Join DS_BCAST_INNER (cost=109.98..2815365714.80
rows=172456 width=27)
          Hash Cond: ("outer".eventid = "inner".eventid)
          -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456
width=14)
            -> XN Hash (cost=87.98..87.98 rows=8798 width=21)
              -> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=21)

```

在產生查詢計畫的過程中，查詢最佳化工具會將計畫劃分為串流、區段和步驟。查詢最佳化工具會細分計畫，以準備將資料和查詢工作負載分配到運算節點。如需串流、區段和步驟的相關資訊，請參閱[查詢計畫和執行工作流程](#)。

下圖顯示先前的查詢和相關聯的查詢計畫。其顯示涉及的查詢操作如何映射至 Amazon Redshift 用於為運算節點配量產生編譯程式碼的步驟。每個查詢計畫操作會映射至區段內的多個步驟，並且有時映射至串流內的多個區段。



在此圖中，查詢最佳化工具會執行查詢計畫，如下所示：

1. 在 Stream 0 中，查詢會以循序掃描操作執行 Segment 0，以掃描 events 資料表。查詢會使用雜湊操作繼續處理 Segment 1，為聯結中的內部資料表建立雜湊表。
2. 在 Stream 1 中，查詢會以循序掃描操作執行 Segment 2，以掃描 sales 資料表。它會使用雜湊連結繼續處理 Segment 2，以聯結含有不是散發索引鍵也不是排序索引鍵之聯結資料欄的資料表。它會再次使用雜湊彙總繼續處理 Segment 2，以彙總結果。然後，查詢會 Segment 3 執行雜湊彙總操作來執行未排序的分組彙總函數，以及排序操作以評估 ORDER BY 子句和其他排序操作。
3. 在 Stream 2 中，查詢會在 Segment 4 和 Segment 5 中執行網路操作，以將中繼結果傳送到領導節點進一步處理。

查詢的最後一個區段會傳回資料。如果傳回集合已彙總或排序，則運算節點會各自將其中繼結果部分傳送到領導節點。領導節點會接著合併資料，以便將最終結果送回給請求的用戶端。

如需EXPLAIN運算子的詳細資訊，請參閱 [EXPLAIN](#)。

影響查詢效能的因素

有一些因素會影響查詢效能。您的資料、叢集和資料庫操作的下列層面，都會影響處理查詢的速度。

- 節點、處理器或配量的數量 – 運算節點會分割為配量。更多節點表示更多處理器和更多配量，透過在配量間並行執行部分的查詢，可讓您的查詢處理得更快速。不過，更多節點也表示更大的開支，因此必須尋找適合您的系統的成本與效能的平衡點。如需 Amazon Redshift 叢集架構的相關資訊，請參閱 [資料倉儲系統架構](#)。
- 節點類型 - Amazon Redshift 叢集可以使用數種節點類型之一。每個節點類型可提供不同的大小和限制，以幫助您適當地調整叢集的規模。節點大小決定叢集中每個節點的儲存容量、記憶體CPU、和價格。如需節點類型的相關資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 叢集概觀](#)。
- 資料配送 – Amazon Redshift 會根據資料表的配送樣式，將資料表資料儲存在運算節點。當您執行查詢時，查詢最佳化器會視需要將資料重新配送至運算節點，以執行任何聯結與彙整。選擇資料表的適當配送樣式，有助於降低重新配送步驟所帶來的影響，方法是在執行聯結之前，將資料放置在需要的位置。如需詳細資訊，請參閱 [查詢最佳化的資料分佈](#)。
- 資料排序排列 – Amazon Redshift 會根據資料表的排序索引鍵，以排序的順序將資料表資料儲存在磁碟上。查詢最佳化器和查詢處理器使用資料所在位置的資訊以減少必須掃描的區塊數量，藉以改善查詢速度。如需詳細資訊，請參閱 [排序金鑰](#)。
- 資料集大小 – 由於需要掃描和重新配送更多資料列，因此叢集中較多的資料量可能拖慢查詢的查詢效能。您可以透過定期清空和封存資料，以及使用述詞來限制查詢資料集，藉以減緩此影響。
- 並行操作 – 一次執行多個操作可能影響查詢效能。每個操作會使用可用查詢佇列中的一或多個位置，並使用與那些位置關聯的記憶體。如果其他操作正在執行，則可能沒有足夠的查詢佇列位置可供使用。在此情況下，查詢必須等候位置開放之後才可以開始處理。如需建立和設定查詢佇列的相關資訊，請參閱 [工作負載管理](#)。
- 查詢結構 – 寫入查詢的方式會影響其效能。請對程序寫入盡可能多的查詢，並傳回符合您需求的最少資料。如需詳細資訊，請參閱 [設計查詢的 Amazon Redshift 最佳實務](#)。
- 程式碼編譯 – Amazon Redshift 會為每個查詢執行計畫產生和編譯程式碼。

編譯的程式碼執行得更快速，因為它省去了使用解譯器的間接成本。第一次產生和編譯程式碼時，通常會有些間接成本。因此，第一次執行查詢的效能可能會有偏差。執行一次性查詢時的間接成本可

能特別明顯。請進行第二次執行查詢，以便判斷其一般效能。Amazon Redshift 使用無伺服器編譯服務，將查詢編譯擴展到 Amazon Redshift 叢集的運算資源之外。編譯的程式碼區段會在叢集本機上快取，而且會在幾乎無限制的快取中快取。此快取會在叢集重新啟動後持續存在。相同查詢的後續執行可以加快執行速度，因為它可以略過編譯階段。

快取在 Amazon Redshift 版本之間不相容，因此會清除編譯快取，並在版本升級後執行查詢時重新編譯程式碼。如果您的查詢具有嚴格的 SLAs，我們建議您預先執行查詢區段，以從叢集資料表掃描資料。這可讓 Amazon Redshift 快取基礎資料表資料，減少版本升級後的查詢規劃時間。透過使用可擴展的編譯服務，Amazon Redshift 可以並行編譯程式碼以提供一致的快速效能。工作負載加速的程度取決於查詢的複雜性和並行性。

查詢分析和改善

從 Amazon Redshift 資料倉儲擷取資訊需要對非常大量的資料執行複雜的查詢，這可能耗費冗長時間來處理。若要確保盡可能快速處理查詢，有一些工具可供您用來識別潛在的效能問題。

主題

- [查詢分析工作流程](#)
- [檢閱查詢提醒](#)
- [分析查詢計劃](#)
- [分析查詢摘要](#)
- [查詢效能改善](#)
- [診斷查詢以進行查詢調校](#)

查詢分析工作流程

如果查詢耗費的時間超過預期，請使用下列步驟來識別並更正可能對查詢效能帶來負面影響的問題。如果您不確定系統中有哪些查詢可能透過效能調校而獲益，請在[識別用於調校的最高候選項目查詢](#)中執行診斷查詢以開始。

1. 確定您的資料表是根據最佳實務而設計。如需詳細資訊，請參閱[設計資料表的 Amazon Redshift 最佳實務](#)。
2. 請查看是否可以刪除或封存資料表中任何不需要的資料。例如，假設您的查詢一律鎖定最近 6 個月的資料量，但是您的資料表中有最近 18 個月的資料。在此情況下，您可以刪除或封存較舊的資料，以減少需要掃描和配送的記錄數量。

3. 在查詢中對資料表執行 [VACUUM](#) 命令，以回收空間和重新排序資料列。如果未排序的區域較大，且查詢在聯結或述詞中使用排序索引鍵，則執行會VACUUM有所幫助。
4. 在查詢中對資料表執行 [ANALYZE](#) 命令，以確定統計資料是最新的。如果查詢中的任何資料表最近變更了許多大小，則執行會ANALYZE有所幫助。如果執行完整ANALYZE命令的時間過長，請在單一資料欄ANALYZE上執行，以減少處理時間。此方法仍會更新資料表大小統計資料；資料表大小是查詢計畫中的顯著因素。
5. 確定您的查詢已對每個類型的用戶端執行一次 (根據用戶端使用的連線通訊協定的類型而定)，使得該查詢會經過編譯和快取。這種方法會加速查詢的後續執行。如需詳細資訊，請參閱[影響查詢效能的因素](#)。
6. 請檢查[STL_ALERT_EVENT_LOG](#)資料表來識別和更正您的查詢的可能問題。如需詳細資訊，請參閱[檢閱查詢提醒](#)。
7. 執行 [EXPLAIN](#) 命令來取得查詢計畫，並用它來最佳化查詢。如需詳細資訊，請參閱[分析查詢計畫](#)。
8. 使用[SVL_QUERY_SUMMARY](#)和[SVL_QUERY_REPORT](#)檢視來取得摘要資訊，並用它來最佳化查詢。如需詳細資訊，請參閱[分析查詢摘要](#)。

有時應該快速執行的查詢會被強制等候其他長時間執行的查詢完成。在該情況下，對於查詢本身，您可能沒有可改善的項目，但您可以藉由對不同類型的查詢建立和使用查詢佇列來改善整體系統效能。若要獲得您的查詢佇列等候時間的概念，請參閱[檢閱查詢的佇列等候時間](#)。如需設定查詢佇列的相關資訊，請參閱[工作負載管理](#)。

檢閱查詢提醒

若要使用[STL_ALERT_EVENT_LOG](#)系統資料表來識別和更正您的查詢的潛在效能問題，請遵循這些步驟：

1. 執行下列動作來判斷您的查詢 ID：

```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

在 substring 欄位中檢查截斷的查詢文字，判斷要選取的 query 值。如果您已執行查詢超過一次，請使用來自具有較低 query 值資料列的 elapsed 值。那是編譯版本的資料列。如果您正在執行許多查詢，您可以提高用於確保查詢包含的LIMIT子句所使用的值。

2. 從 STL_ALERT_EVENT_LOG 選取查詢的資料列：

```
Select * from stl_alert_event_log where query = MyQueryID;
```

userid	query	slice	segment	step	pid	xid	event	solution	event_time
100	32359	4	0	0	8780	71195	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-10 17:40:50
100	32359	5	0	0	8781	71195	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-10 17:40:50
100	109142	4	0	0	8780	302411	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-24 20:32:28
100	109142	5	0	0	8781	302411	Very selective query filter:ratio=rows(2)/r	Review the choice of sort key to enable...	2015-02-24 20:32:28
100	109828	4	1	0	8746	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:27:52
100	109828	5	1	0	8747	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:27:52
100	109829	4	1	0	8760	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:28:01
100	109829	5	1	0	8761	304543	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-24 23:28:01
100	113910	4	1	0	8774	316848	Very selective query filter:ratio=rows(3)/r	Review the choice of sort key to enable...	2015-02-25 17:14:58

3. 評估您的查詢的結果。使用下表來找出任何遇到問題的潛在解決方案。

Note

並非所有查詢都有 STL_ALERT_EVENT_ 中的資料列LOG，只有已識別問題的資料列。

問題	事件值	解決方案值	建議的解決方案
查詢中資料表的統計資料遺漏或過時。	遺漏查詢規劃器統計資料	執行 ANALYZE 命令	請參閱 資料表統計資訊遺漏或過時 。
查詢計畫中有巢狀迴路聯結 (最差的最佳聯結)。	查詢計畫中的巢狀迴路聯結	檢閱聯結述詞以避免笛卡兒乘積	請參閱 巢狀迴圈 。
掃描已略過相當多標示為已刪除但未清空的資料列，或已插入但未遞交的資料列。	已掃描大量刪除的資料列	執行 VACUUM 命令以回收已刪除的空間	請參閱 幽靈資料列或未遞交的資料列 。
已針對雜湊聯結或彙整重新配送超過 1,000,000 個資料列。	在整個網路中分發大量資料列：RowCount 資料列已分發，以便處理彙總	檢閱散發索引鍵的選擇以共置聯結或彙整	請參閱 次佳資料分佈 。
已針對雜湊聯結播送超過 1,000,000 個資料列。	在網路間播送大量資料列	檢閱散發索引鍵的選擇以共置聯	請參閱 次佳資料分佈 。

問題	事件值	解決方案值	建議的解決方案
		結或並考慮使用 配送資料表	
DS_DIST_ALL_INNER 重新分發樣式在查詢計劃中表示，這會強制序列執行，因為整個內部資料表已重新分發至單一節點。	DS_DIST_ALL_INNER for Hash 在查詢計劃中加入	檢閱配送策略的選擇，以配送內部而非外部資料表	請參閱 次佳資料分佈 。

分析查詢計劃

執行 [EXPLAIN](#) 命令來取得查詢計畫。

分析查詢計畫之前，您應該熟悉如何閱讀它。如果您對於閱讀查詢計畫不熟悉，建議您閱讀[建立和解譯查詢計劃](#)之後再繼續。

若要分析查詢計畫提供的資料，請遵循這些步驟：

1. 識別具有最高成本的步驟。繼續進行其餘步驟時，請專注在那些。
2. 查看聯結類型：
 - 巢狀迴路：這類聯結的發生通常是因為省略了聯結條件。如需建議的解決方案，請參閱[巢狀迴路](#)。
 - 雜湊和雜湊聯結：當聯結資料表中的聯結資料欄不是散發索引鍵也不是排序索引鍵時，會使用雜湊聯結。如需建議的解決方案，請參閱[雜湊聯結](#)。
 - 合併聯結：不需變更。
3. 注意哪個資料表用於內部聯結，以及哪個用於外部聯結。查詢引擎一般會對內部聯結選擇較小的資料表，以及對外部聯結選擇較大的資料表。如果這類選擇未發生，那麼，您的統計資料很可能過時。如需建議的解決方案，請參閱[資料表統計資訊遺漏或過時](#)。
4. 查看是否有任何高成本的排序操作。如果有，請參閱[未排序或排序錯誤的資料列](#)以取得建議的解決方案。
5. 尋找具有高成本操作的下列播送運算子：
 - DS_BCAST_INNER：表示資料表已廣播至所有運算節點。這對於小資料表來說很好，但對於較大的資料表來說並不理想。
 - DS_DISTALL_INNER：表示所有工作負載都在單一切片上。
 - DS_DIST_BOTH：表示大量重新分發。

如需這些情況的建議解決方案，請參閱[次佳資料分佈](#)。

分析查詢摘要

若要取得較 [EXPLAIN](#) 所產生的查詢計畫更詳細的執行步驟和統計資料，請使用 [SVL_QUERY_SUMMARY](#) 和 [SVL_QUERY_REPORT](#) 系統檢視。

[SVL_QUERY_SUMMARY](#) 依串流提供查詢統計資料。您可以使用其提供的資訊來識別具有代價高昂步驟、長時間執行步驟和寫入磁碟步驟的問題。

[SVL_QUERY_REPORT](#) 系統檢視可讓您SUMMARY僅透過運算節點層而非串流，查看與 [SVL_QUERY](#) 相似的資訊。您可以使用配量層級資訊來偵測叢集間不平均的資料配送 (也稱為資料配送偏度)，它會強制部分節點執行較其他節點更多的工作，並影響查詢效能。

主題

- [使用 \[SVL_QUERY_SUMMARY\]\(#\) 檢視](#)
- [使用 \[SVL_QUERY_REPORT\]\(#\) 檢視](#)
- [將查詢計劃映射到查詢摘要](#)

使用 [SVL_QUERY_SUMMARY](#) 檢視

若要使用 依串流分析查詢摘要資訊[SVL_QUERY_SUMMARY](#)，請執行下列動作：

1. 執行下列查詢來判斷您的查詢 ID：

```
select query, elapsed, substring
from svl_qlog
order by query
desc limit 5;
```

在 `substring` 欄位中檢查截斷的查詢文字，判斷代表您的查詢的 `query` 值。如果您已執行查詢超過一次，請使用來自具有較低 `query` 值資料列的 `elapsed` 值。那是編譯版本的資料列。如果您正在執行許多查詢，您可以提高用於確保查詢包含的LIMIT子句所使用的值。

2. 從 [SVL_QUERY_SUMMARY](#) 選取查詢的資料列。依串流、區段和步驟排列結果：

```
select * from svl_query_summary where query = MyQueryID order by stm, seg, step;
```

以下是結果範例。

userid	query	stn	seg	step	maxtime	avgtime	rows	bytes	rate_row	rate_byte	label	is_diskbased	workmem	is_rscan	is_delayed_scan	rows_pre_filter
1249059		0	0	0	58	27	4	192			scan tbl=246 name=Internal Worktable	f		0 f	f	0
1249059		0	0	1	58	27	4	0			project	f		0 f	f	0
1249059		0	0	2	58	27	4	64			save tbl=249	f	481296384 f	f	f	0
1249059		1	1	0	20	20	1	48			scan tbl=250 name=Internal Worktable	f		0 f	f	0
1249059		1	1	1	20	20	1	0			dist	f		0 f	f	0
1249059		1	2	0	2275	1350	1	48			scan tbl=19221 name=Internal Worktable	f		0 f	f	0
1249059		1	2	1	2275	1350	1	0			project	f		0 f	f	0
1249059		1	2	2	2275	1350	1	16			save tbl=249	f	475004928 f	f	f	0
1249059		2	3	0	1640	792	5	80			scan tbl=249 name=Internal Worktable	f		0 f	f	0
1249059		2	3	1	1640	792	5	80			sort tbl=248	f	468713472 f	f	f	0
1249059		3	4	0	26	9	5	80			scan tbl=248 name=Internal Worktable	f		0 f	f	0
1249059		3	4	1	26	9	5	0			return	f		0 f	f	0
1249059		3	5	0	49	49	0	0			merge	f		0 f	f	0
1249059		3	5	1	49	49	0	0			project	f		0 f	f	0
1249059		3	5	2	49	49	0	0			return	f		0 f	f	0

3. 使用[將查詢計劃映射到查詢摘要](#)中的資訊，將步驟與查詢計畫中的操作映射。它們的資料列和位元組應該具有大約相同的值 (查詢計畫中的資料列 * 寬度)。如果不同，請參閱[資料表統計資訊遺漏或過時](#)以取得建議的解決方案。
4. 查看任何步驟的 `is_diskbased` 欄位是否具有 `t (true)` 值。如果系統沒有為查詢處理配置足夠的記憶體，雜湊、彙整和排序為可能將資料寫入至磁碟的運算子。

如果 `is_diskbased` 為 `true`，請參閱[配置給查詢的記憶體不足](#)以取得建議的解決方案。

5. 檢閱 `label` 欄位值，並查看步驟中是否有 `AGG-DIST-AGG` 序列。出現它表示兩個步驟彙整，其代價高昂。若要修正此問題，請變更 `GROUP BY` 子句以使用分發金鑰 (第一個金鑰，如果有多個金鑰)。
6. 檢閱每個區段的 `maxtime` 值 (區段中的所有步驟是相同的)。識別具有最高 `maxtime` 值的區段，並檢閱此區段中下列運算子的步驟。

Note

高的 `maxtime` 值並不一定代表區段有問題。雖然值很高，但該區段可能並未耗費大量時間處理。串流中的所有區段會同時開始計時。不過，部分下游區段必須等到取得來自上游的資料後才能執行。此影響可能使得它們看起來耗費了長時間，因為其 `maxtime` 值將同時包含等候時間和處理時間。

- `BCAST` 或 `DIST`：在這些情況下，高 `maxtime` 值可能是重新分發大量資料列的結果。如需建議的解決方案，請參閱[次佳資料分佈](#)。
- `HJOIN` (hash join)：如果與 `RETURN` 查詢中最後一個步驟的值相比，該步驟在 `rows` 欄位中具有非常高 `rows` 的值，請參閱[雜湊連結](#)以取得建議的解決方案。
- `SCAN/SORT`：尋找連結步驟前的步驟 `SCANMERGE` 順序 `SCAN`、`SORT`、。這個模式指出未排序的資料會經過掃描、排序，然後與資料表排序的區域合併。

看看相較於查詢中最後一個SCAN步驟中的資料列值，該RETURN步驟的資料列值是否有極高的值。這個模式指出執行引擎正在掃描稍後將捨棄的資料列，這樣做缺乏效率。如需建議的解決方案，請參閱[不足的限制性述詞](#)。

如果SCAN步驟maxtime的值很高，請參閱[次佳WHERE子句](#)以取得建議的解決方案。

如果SORT步驟rows的值不是零，請參閱[未排序或排序錯誤的資料列](#)以取得建議的解決方案。

7. 檢閱最後一個步驟之前 5–10 RETURN 個步驟的 rows 和 bytes 值，以了解傳回用戶端的資料量。此程序可以是一門藝術。

例如，在以下範例查詢摘要中，第三個PROJECT步驟提供 rows 值，但不是 bytes 值。透過查看具有相同rows值的 的上述步驟，您可以找到同時提供資料列和位元組資訊SCAN的步驟。

以下是範例結果。

userid	query	stm	seg	step	maxtime	avgtime	rows	bytes	rate_row	rate_byte	label	is_diskbased	workmem
1	187435	2	5	2	14307	12797	0	0			hash tbl=256	f	46871347
1	187435	3	6	0	531	308	387	229104			scan tbl=242 name=Internal Worktable	f	
1	187435	3	6	1	531	308	387	0			project	f	
1	187435	3	6	2	531	308	387	222912			save tbl=245	f	38063306
1	187435	4	7	0	390	390	0	0			scan tbl=238 name=Internal Worktable	f	
1	187435	4	7	1	390	390	0	0			dist	f	
1	187435	4	8	0	1218	1066	0	0			scan tbl=134954 name=Internal Worktable	f	
1	187435	4	8	1	1218	1066	0	0			project	f	
1	187435	4	8	2	1218	1066	0	0			save tbl=245	f	37434165
1	187435	5	9	0	171	83	387	222912			scan tbl=245 name=Internal Worktable	f	
1	187435	5	9	1	171	83	387	60120			dist	f	
1	187435	5	10	0	3579	3383	387	222912			scan tbl=134955 name=Internal Worktable	f	
1	187435	5	10	1	3579	3383	387	0			project	f	
1	187435	5	10	2	3579	3383	0	0			hjoin tbl=256	f	
1	187435	5	10	3	3579	3383	0	0			project	f	
1	187435	5	10	4	3579	3383	0	0			sort tbl=259	f	36805017
1	187435	6	11	0	10	7	0	0			scan tbl=259 name=Internal Worktable	f	
1	187435	6	11	1	10	7	0	0			return	f	
1	187435	6	12	0	9	9	0	0			merge	f	
1	187435	6	12	1	9	9	0	0			project	f	
1	187435	6	12	2	9	9	0	0			return	f	

如果您要傳回異常大的資料量，請參閱[非常大的結果集](#)以取得建議的解決方案。

8. 相較於其他步驟，查看 bytes 值是否相對於任何步驟中的 rows 值來得高。這個模式可能指出您正選取許多資料欄。如需建議的解決方案，請參閱[大型SELECT清單](#)。

使用 SVL_QUERY_REPORT 檢視

若要使用 依切片分析查詢摘要資訊 [SVL_QUERY_REPORT](#)，請執行下列動作：

1. 執行下列動作來判斷您的查詢 ID：

```
select query, elapsed, substring
from svl_qlog
```

```
order by query
desc limit 5;
```

在 substring 欄位中檢查截斷的查詢文字，判斷代表您的查詢的 query 值。如果您已執行查詢超過一次，請使用來自具有較低 query 值資料列的 elapsed_time 值。那是編譯版本的資料列。如果您一直執行許多查詢，您可以提高子LIMIT句所使用的值，以確保包含查詢。

- 從 SVL_QUERY_REPORT 選取查詢的資料列。依區段、步驟、經過時間和資料列排列結果：

```
select * from svl_query_report where query = MyQueryID order by segment, step,
elapsed_time, rows;
```

- 針對每個步驟，查看所有配量是否正處理大約相同數量的資料列：

userid	query	slice	segment	step	start_time	end_time	elapsed_time	rows	bytes	label	is
100	141696	2	0	0	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	scan tbl=95423 name=Internal Worktable	f
100	141696	5	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	420	1100	31700	bcast	f
100	141696	1	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	437	1099	31812	bcast	f
100	141696	3	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	490	1066	30108	bcast	f
100	141696	6	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	576	1108	32316	bcast	f
100	141696	4	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	583	1128	32484	bcast	f
100	141696	0	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	726	1079	30804	bcast	f
100	141696	7	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	bcast	f
100	141696	2	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2406	1068	31056	bcast	f
100	141696	2	1	0	2014-09-12 18:45:33	2014-09-12 18:45:33	3441	8798	253580	scan tbl=95423 name=Internal Worktable	f

也查看所有配量是否正耗費大約相同的時間量：

userid	query	slice	segment	step	start_time	end_time	elapsed_time	rows	bytes	label	is
100	141696	2	0	0	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	scan tbl=95423 name=Internal Worktable	f
100	141696	5	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	420	1100	31700	bcast	f
100	141696	1	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	437	1099	31812	bcast	f
100	141696	3	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	490	1066	30108	bcast	f
100	141696	6	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	576	1108	32316	bcast	f
100	141696	4	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	583	1128	32484	bcast	f
100	141696	0	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	726	1079	30804	bcast	f
100	141696	7	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2109	1150	33300	bcast	f
100	141696	2	0	2	2014-09-12 18:45:33	2014-09-12 18:45:33	2406	1068	31056	bcast	f
100	141696	2	1	0	2014-09-12 18:45:33	2014-09-12 18:45:33	3441	8798	253580	scan tbl=95423 name=Internal Worktable	f

這些值的差異大可能表示由於此特定查詢的次佳配送樣式造成的資料配送偏度。如需建議的解決方案，請參閱[次佳資料分佈](#)。

將查詢計劃映射到查詢摘要

分析查詢摘要時，您可以將查詢計畫中的操作對應至查詢摘要中的步驟（由標籤欄位值識別），以取得更多詳細資訊。下表映射查詢計畫操作以查詢摘要步驟。

查詢計劃操作	標籤欄位值	描述
Aggregate HashAggregate GroupAggregate	AGGR	評估彙總函數和 GROUP BY 條件。
DS_BCAST_INNER	BCAST (廣播)	將整個資料表或某組資料列 (例如來自資料表篩選的一組資料列) 播送至所有節點。
不會出現在查詢計畫中	DELETE	刪除資料表中的資料列。
DS_DIST_NONE DS_DIST_ALL_NONE DS_DIST_INNER DS_DIST_ALL_INNER DS_DIST_ALL_BOTH	DIST (分佈)	針對平行聯結目的或其他平行處理，將資料列配送至節點。
HASH	HASH	建置要用於雜湊聯結的雜湊表。
雜湊聯結	HJOIN (hash 聯結)	執行兩個資料表或中繼結果集的雜湊聯結。
不會出現在查詢計畫中	INSERT	將資料列插入至資料表。
限制	LIMIT	將LIMIT子句套用至結果集。
Merge	MERGE	合併衍生自平行排序或聯結操作的資料列。
合併聯結	MJOIN (合併聯結)	執行兩個資料表或中繼結果集的合併聯結。

查詢計劃操作	標籤欄位值	描述
巢狀迴路	NLOOP (巢狀迴圈)	執行兩個資料表或中繼結果集的巢狀迴路聯結。
不會出現在查詢計畫中	PARSE	將字串剖析為二進位值以進行載入。
專案	PROJECT	評估表達式。
網路	RETURN	將資料列傳回至領導者或用戶端。
不會出現在查詢計畫中	SAVE	具體化資料列以使於下一個處理步驟。
循序掃描	SCAN	掃描資料表或中繼結果集。
Sort	SORT	根據其他後續操作 (例如聯結或彙總) 的需求或滿足 ORDER BY 子句, 對資料列或中間結果集進行排序。
唯一	UNIQUE	套用SELECTDISTINCT子句或移除其他操作所需的重複項目。
視窗	WINDOW	計算彙整和排名視窗函數。

查詢效能改善

以下是影響 Amazon Redshift 查詢效能的一些常見問題，其中包含診斷和解決方法的說明。

主題

- [資料表統計資訊遺漏或過時](#)
- [巢狀迴圈](#)
- [雜湊聯結](#)
- [幽靈資料列或未遞交的資料列](#)

- [未排序或排序錯誤的資料列](#)
- [次佳資料分佈](#)
- [配置給查詢的記憶體不足](#)
- [次佳WHERE子句](#)
- [不足的限制性述詞](#)
- [非常大的結果集](#)
- [大型SELECT清單](#)

資料表統計資訊遺漏或過時

如果資料表統計資料遺漏或過時，您可能會看見下列：

- EXPLAIN 命令中的警告訊息會產生。
- STL_ALERT_EVENT_ 中遺失統計資料警示事件LOG。如需詳細資訊，請參閱[檢閱查詢提醒](#)。

若要修正此問題，請執行 [ANALYZE](#)。

巢狀迴圈

如果存在巢狀迴圈，您可能會在 STL_ALERT_EVENT_ 中看到巢狀迴圈警示事件LOG。您也可以透過執行[識別具有巢狀迴圈的查詢](#)中的查詢，來識別此類型的事件。如需詳細資訊，請參閱[檢閱查詢提醒](#)。

若要修正此問題，請檢閱您的查詢以取得交叉聯結並在可能時加以移除。交叉聯結是沒有聯結條件的聯結，會造成兩個資料表的笛卡兒乘積。它們通常會以巢狀迴圈聯結的形式執行，這是可能的聯結類型中最慢的。

雜湊聯結

如果出現雜湊聯結，您可能會看見下列：

- 查詢計畫中的雜湊和雜湊聯結操作。如需詳細資訊，請參閱[分析查詢計劃](#)。
- 在 SVL_QUERY_ 中具有最高最長時間值的區段中的HJOIN步驟SUMMARY。如需詳細資訊，請參閱[使用 SVL_QUERY_SUMMARY 檢視](#)。

若要修正此問題，您可以採取兩個方法：

- 如果可能，重新寫入查詢以使用合併聯結。指定同時為散發索引鍵和排序索引鍵的聯結資料欄即可執行此動作。
- 如果 SVL_QUERY_SUMMARY 中的 HJOIN 步驟與查詢中最後一個 RETURN 步驟中的資料列值相比，資料列欄位中的值非常高，請檢查是否可以重寫查詢以在唯一資料欄上聯結。當查詢未在唯一資料欄上聯結時，例如主要索引鍵，它會增加聯結中牽涉的資料列數量。

幽靈資料列或未遞交的資料列

如果存在 ghost 資料列或未認可資料列，您可能會在 STL_ALERT_EVENT_LOG 中看到警示事件，指出過度的 ghost 資料列。如需詳細資訊，請參閱[檢閱查詢提醒](#)。

若要修正此問題，您可以採取兩個方法：

- 查看 Amazon Redshift 主控台的載入索引標籤，以瞭解任何查詢資料表上的作用中載入操作。如果您看見作用中的載入操作，採取動作之前，請等候那些操作完成。
- 如果沒有作用中載入操作，請在查詢資料表上執行 [VACUUM](#) 來移除刪除的資料列。

未排序或排序錯誤的資料列

如果存在未排序或錯誤列，您可能會在 STL_ALERT_EVENT_ 中看到非常選擇性的篩選條件警示事件 LOG。如需詳細資訊，請參閱[檢閱查詢提醒](#)。

您也可以透過執行 [識別具有資料扭曲或未排序資料列的資料表](#) 中的查詢，查看您的查詢中是否任何資料表有大型未排序的區域。

若要修正此問題，您可以採取兩個方法：

- 在查詢資料表上執行 [VACUUM](#) 來重新排序資料列。
- 檢閱查詢資料表上的排序索引鍵，以查看是否有可以進行的任何改善。請記得評估此查詢的效能與其他重要查詢和整體系統的效能，再進行任何變更。如需詳細資訊，請參閱[排序金鑰](#)。

次佳資料分佈

如果資料配送為次佳，您可能看見下列：

- 序列執行、大型廣播或大型分佈提醒事件會顯示在 STL_ALERT_EVENT_ 中 LOG。如需詳細資訊，請參閱[檢閱查詢提醒](#)。

- 針對指定步驟，配量未處理大約相同數量的資料列。如需詳細資訊，請參閱[使用 SVL_QUERY_REPORT 檢視](#)。
- 針對指定步驟，配量未耗費大約相同的時間量。如需詳細資訊，請參閱[使用 SVL_QUERY_REPORT 檢視](#)。

如果前述中無一成立，您也可以透過執行 [識別具有資料扭曲或未排序資料列的資料表](#) 中的查詢，查看您的查詢中是否有任何資料表有資料偏度。

若要修正此問題，查看查詢中資料表的分佈樣式，並確認是否可以進行任何改善。請記得評估此查詢的效能與其他重要查詢和整體系統的效能，再進行任何變更。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

配置給查詢的記憶體不足

如果分配給查詢的記憶體不足，您可能會在 SVL_QUERY_SUMMARY 中看到 is_diskbased 一個值為 true 的步驟。如需詳細資訊，請參閱[使用 SVL_QUERY_SUMMARY 檢視](#)。

若要修正此問題，請透過暫時增加它所使用查詢位置的數量，為查詢配置更多記憶體。Workload Management (WLM) 會保留查詢佇列中的插槽，等同於為佇列設定的並行層級。例如，具有並行層級 5 的佇列有 5 個位置。指派給佇列的記憶體會平均配置到每個位置。對一個查詢指派數個位置可讓該查詢存取所有那些位置的記憶體。如需如何暫時增加查詢位置的相關資訊，請參閱[wlm_query_slot_count](#)。

次佳WHERE子句

如果您的WHERE子句導致過度掃描資料表，您可能會在 區段中看到一個SCAN步驟，其maxtime值最高為 SVL_QUERY_SUMMARY。如需詳細資訊，請參閱[使用 SVL_QUERY_SUMMARY 檢視](#)。

若要修正此問題，請根據最大資料表的主要排序資料欄，將WHERE子句新增至查詢。這種方法可縮短降低掃描時間。如需詳細資訊，請參閱[設計資料表的 Amazon Redshift 最佳實務](#)。

不足的限制性述詞

如果您的查詢具有限制性不足的述詞，您可能會在 SVL_QUERY_SUMMARY 中看到具有最高maxtime值的區段中，該SCAN值與查詢中最後一個RETURN步驟中的rows值rows相比非常高。如需詳細資訊，請參閱[使用 SVL_QUERY_SUMMARY 檢視](#)。

若要修正此問題，請嘗試新增述詞至查詢，或讓現有述詞更具限制性來縮小列輸出。

非常大的結果集

如果您的查詢傳回非常大的結果集，請考慮重新寫入查詢，以使用 [UNLOAD](#) 來寫入結果至 Amazon S3。此方法利用平行處理來改善RETURN步驟的效能。如需檢查非常大型結果集的相關資訊，請參閱[使用 SVL_QUERY_SUMMARY 檢視](#)。

大型SELECT清單

如果您的查詢SELECT清單異常大，您可能會在 SVL_QUERY_ 中看到相對於任何步驟 bytes（與其他步驟相比）rows值較高的值SUMMARY。這個高 bytes 值可能是您正選取許多資料欄的指標。如需詳細資訊，請參閱[使用 SVL_QUERY_SUMMARY 檢視](#)。

若要修正此問題，請檢閱您要選取的資料欄，並查看是否可移除任何項目。

診斷查詢以進行查詢調校

使用下列查詢來識別查詢的問題或可能影響查詢效能的基礎資料表。建議您使用這些查詢結合[查詢分析和改善](#)中討論的查詢調校程序。

Note

這些查詢適用於 Amazon Redshift 佈建叢集。這些查詢不適用於 Redshift Serverless 工作群組。

主題

- [識別用於調校的最高候選項目查詢](#)
- [識別具有資料扭曲或未排序資料列的資料表](#)
- [識別具有巢狀迴圈的查詢](#)
- [檢閱查詢的佇列等候時間](#)
- [依資料表檢閱查詢提醒](#)
- [識別具有遺漏統計資訊的資料表](#)

識別用於調校的最高候選項目查詢

下列查詢會識別過去 7 天中已執行的前 50 個最耗時的陳述式。您可以使用結果來識別需要異常長時間的查詢。您也可以識別經常執行的查詢（在結果集中出現超過一次的項目）。這些查詢往往是進行調校以改善系統效能的良好候選項目。

此查詢也提供與所識別的每個查詢關聯的提醒事件計數。這些提醒提供的詳細資訊可供您用來改善查詢的效能。如需詳細資訊，請參閱[檢閱查詢提醒](#)。

```
select trim(database) as db, count(query) as n_qry,
max(substring (qrytext,1,80)) as qrytext,
min(run_minutes) as "min" ,
max(run_minutes) as "max",
avg(run_minutes) as "avg", sum(run_minutes) as total,
max(query) as max_query_id,
max(starttime)::date as last_run,
sum(alerts) as alerts, aborted
from (select userid, label, stl_query.query,
trim(database) as database,
trim(querytxt) as qrytext,
md5(trim(querytxt)) as qry_md5,
starttime, endtime,
(datediff(seconds, starttime,endtime)::numeric(12,2))/60 as run_minutes,
alrt.num_events as alerts, aborted
from stl_query
left outer join
(select query, 1 as num_events from stl_alert_event_log group by query ) as alrt
on alrt.query = stl_query.query
where userid <> 1 and starttime >= dateadd(day, -7, current_date))
group by database, label, qry_md5, aborted
order by total desc limit 50;
```

識別具有資料扭曲或未排序資料列的資料表

下列查詢會識別具有不均資料配送 (資料偏度) 或高比例未排序資料列的資料表。

低的 skew 值指出資料表資料已正確配送。如果資料表有 4.00 或更高的 skew 值，請考慮修改其資料配送樣式。如需詳細資訊，請參閱[次佳資料分佈](#)。

如果資料表的 pct_unsorted 值大於 20%，請考慮執行 [VACUUM](#) 命令。如需詳細資訊，請參閱[未排序或排序錯誤的資料列](#)。

同時另外檢閱每個資料表的 mbytes 和 pct_of_total 值。這些資料欄會識別資料表的大小，以及資料表耗用的原始磁碟空間百分比。原始磁碟空間包括 Amazon Redshift 保留供內部使用的空間，因此會大於名目磁碟容量，它是可供使用者使用的磁碟空間容量。使用此資訊來確保您的可用磁碟空間至少是您最大資料表的 2.5 倍。有此可用空間可讓系統在處理複雜查詢時將中繼結果寫入至磁碟。

```
select trim(pgn.nspname) as schema,
```

```

trim(a.name) as table, id as tableid,
decode(pgc.reldiststyle,0, 'even',1,det.distkey ,8,'all') as distkey,
  dist_ratio.ratio::decimal(10,4) as skew,
det.head_sort as "sortkey",
det.n_sortkeys as "#sks", b.mbytes,
decode(b.mbytes,0,0,((b.mbytes/part.total::decimal)*100)::decimal(5,2)) as
  pct_of_total,
decode(det.max_enc,0,'n','y') as enc, a.rows,
decode( det.n_sortkeys, 0, null, a.unsorted_rows ) as unsorted_rows ,
decode( det.n_sortkeys, 0, null, decode( a.rows,0,0, (a.unsorted_rows::decimal(32)/
a.rows)*100) )::decimal(5,2) as pct_unsorted
from (select db_id, id, name, sum(rows) as rows,
sum(rows)-sum(sorted_rows) as unsorted_rows
from stv_tbl_perm a
group by db_id, id, name) as a
join pg_class as pgc on pgc.oid = a.id
join pg_namespace as pgn on pgn.oid = pgc.relnamespace
left outer join (select tbl, count(*) as mbytes
from stv_blocklist group by tbl) b on a.id=b.tbl
inner join (select attrelid,
min(case attisdistkey when 't' then attname else null end) as "distkey",
min(case attsortkeyord when 1 then attname else null end ) as head_sort ,
max(attsortkeyord) as n_sortkeys,
max(attencodingtype) as max_enc
from pg_attribute group by 1) as det
on det.attrelid = a.id
inner join ( select tbl, max(mbytes)::decimal(32)/min(mbytes) as ratio
from (select tbl, trim(name) as name, slice, count(*) as mbytes
from svv_diskusage group by tbl, name, slice )
group by tbl, name ) as dist_ratio on a.id = dist_ratio.tbl
join ( select sum(capacity) as total
from stv_partitions where part_begin=0 ) as part on 1=1
where mbytes is not null
order by mbytes desc;

```

識別具有巢狀迴圈的查詢

下列查詢可識別已針對巢狀迴路記錄提醒事件的查詢。如需如何修正巢狀迴路條件的資訊，請參閱[巢狀迴圈](#)。

```

select query, trim(querytxt) as SQL, starttime
from stl_query
where query in (

```



```
select distinct query
from stl_alert_event_log
where event like 'Nested Loop Join in the query plan%'
order by starttime desc;
```

檢閱查詢的佇列等候時間

下列查詢顯示最近的查詢在執行之前等候查詢佇列中開放位置的時間。如果您看見較高的等候時間趨勢，您可能想要修改您的查詢佇列組態以獲得更好的傳輸量。如需詳細資訊，請參閱[實作手冊 WLM](#)。

```
select trim(database) as DB , w.query,
substring(q.querytxt, 1, 100) as querytxt, w.queue_start_time,
w.service_class as class, w.slot_count as slots,
w.total_queue_time/1000000 as queue_seconds,
w.total_exec_time/1000000 exec_seconds, (w.total_queue_time+w.total_Exec_time)/1000000
as total_seconds
from stl_wlm_query w
left join stl_query q on q.query = w.query and q.userid = w.userid
where w.queue_start_Time >= dateadd(day, -7, current_Date)
and w.total_queue_Time > 0 and w.userid >1
and q.starttime >= dateadd(day, -7, current_Date)
order by w.total_queue_time desc, w.queue_start_time desc limit 35;
```

依資料表檢閱查詢提醒

下列查詢可識別已為其記錄提醒事件的資料表，也可識別最常引發的提醒類型。

如果具有已識別資料表資料列的 minutes 值較高，請檢查資料表，以查看它是否需要例行維護，例如對它執行 [ANALYZE](#) 或 [VACUUM](#)。

如果資料列 count 的值很高，但 table 值為 Null，請針對關聯 event 值 STL_ALERT_EVENT_LOG 執行查詢，以調查警示頻繁發出的原因。

```
select trim(s.perm_table_name) as table,
(sum(abs(datediff(seconds, s.starttime, s.endtime)))/60)::numeric(24,0) as minutes,
trim(split_part(l.event, ':', 1)) as event, trim(l.solution) as solution,
max(l.query) as sample_query, count(*)
from stl_alert_event_log as l
left join stl_scan as s on s.query = l.query and s.slice = l.slice
and s.segment = l.segment and s.step = l.step
where l.event_time >= dateadd(day, -7, current_Date)
```

```
group by 1,3,4
order by 2 desc,6 desc;
```

識別具有遺漏統計資訊的資料表

下列查詢提供您要對遺漏統計資料的資料表執行的查詢計數。如果此查詢傳回任何資料列，請查看 `plannode` 值來判斷受影響的資料表，然後對其執行 [ANALYZE](#)。

```
select substring(trim(plannode),1,100) as plannode, count(*)
from stl_explain
where plannode like '%missing statistics%'
group by plannode
order by 2 desc;
```

查詢疑難排解

此小節提供快速的參考，用於識別和解決使用 Amazon Redshift 查詢時可能遇到的部分最常見和最嚴重的問題。

主題

- [連線失敗](#)
- [查詢懸置](#)
- [查詢耗費太長時間](#)
- [載入失敗](#)
- [載入耗費太長時間](#)
- [載入資料不正確](#)
- [設定JDBC擷取大小參數](#)

這些建議可做為進行故障排除的起點。您也可以參閱下列資源以取得更詳細的資訊。

- [存取 Amazon Redshift 叢集和資料庫](#)
- [自動資料表最佳化](#)
- [在 Amazon Redshift 中載入資料](#)

- [教學課程：從 Amazon S3 載入資料](#)

連線失敗

由於下列原因，您的查詢連線可能會失敗。我們建議使用以下疑難排解方法。

用戶端無法連線至伺服器

如果您使用的是 SSL 或 伺服器憑證，請在疑難排解連線問題時，先移除此複雜性。然後在找到解決方案時新增 SSL 或 伺服器憑證。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[設定連線的安全性選項](#)。

連線遭拒

一般來說，收到錯誤訊息指出無法建立連線時，它表示存取叢集的許可發生問題。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[連線遭拒或失敗](#)。

查詢懸置

您的查詢可以因下列原因掛起或停止回應。我們建議使用以下疑難排解方法。

對資料庫的連線已丟棄

減少最大傳輸單位的大小 (MTU)。MTU 大小決定可透過網路連線在一個乙太網路框架中傳輸的封包大小上限，以位元組為單位。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[已中斷與資料庫的連線](#)。

對資料庫的連線逾時

執行長查詢時，您對資料庫的用戶端連線似乎已暫停或逾時，例如 COPY 命令。在此情況下，您可能會發現 Amazon Redshift 主控台顯示查詢已完成，但用戶端工具本身仍似乎在執行查詢。取決於連線停止的時間，查詢的結果可能遺漏或不完整。當中繼網路元件終止閒置的連線時，會發生此影響。如需詳細資訊，請前往《Amazon Redshift 管理指南》中的[防火牆逾時問題](#)。

用戶端 out-of-memory 錯誤發生在 ODBC

如果您的用戶端應用程式使用 ODBC 連線，且查詢建立的結果集太大而無法容納記憶體，您可以使用游標將結果集串流至用戶端應用程式。如需詳細資訊，請參閱 [DECLARE](#) 和 [使用游標時的效能考量](#)。

用戶端 out-of-memory 錯誤發生在 JDBC

當您嘗試透過JDBC連線擷取大型結果集時，您可能會遇到用戶端 out-of-memory錯誤。如需詳細資訊，請參閱[設定JDBC擷取大小參數](#)。

可能有死鎖

如果有可能的死鎖，請嘗試下列：

- 檢視 [STV_LOCKS](#) 和 [STL_TR_CONFLICT](#) 系統資料表來尋找牽涉到對超過一個資料表進行更新的衝突。
- 使用 [PG_CANCEL_BACKEND](#) 函數來取消一或多個有衝突的查詢。
- 使用 [PG_TERMINATE_BACKEND](#) 函數來終止工作階段，它可強制終止的工作階段中任何目前執行中的交易釋出所有鎖定和復原交易。
- 排程並行寫入操作時務必謹慎。如需詳細資訊，請參閱[管理並行寫入操作](#)。

查詢耗費太長時間

由於下列原因，您的查詢可能需要太長的時間。我們建議使用以下疑難排解方法。

資料表未進行優化

設定資料表的排序索引鍵、配送樣式和壓縮編碼，以善加利用平行處理。如需詳細資訊，請參閱 [自動資料表最佳化](#)

查詢正在寫入磁碟

您的查詢可能會在查詢執行的至少一部分寫入至磁碟。如需詳細資訊，請參閱[查詢效能改善](#)。

查詢必須等候其他查詢完成

您可以透過建立查詢佇列和指派不同類型的查詢至適當的佇列，來改善整體系統效能。如需詳細資訊，請參閱[工作負載管理](#)。

查詢未進行優化

分析解釋計畫以尋找重新寫入查詢或最佳化資料庫的機會。如需詳細資訊，請參閱[建立和解譯查詢計劃](#)。

查詢需要更多記憶體來執行

如果特定查詢需要更多記憶體，您可以透過增加[wlm_query_slot_count](#)來增加可用記憶體。

資料庫需要執行VACUUM命令

每當您新增、刪除或修改大量資料列時，請執行 VACUUM 命令，除非您以排序索引鍵順序載入資料。此 VACUUM 命令會重新組織您的資料，以維護排序順序並還原效能。如需詳細資訊，請參閱 [清空資料表](#)。

疑難排解長期執行查詢的其他資源

以下是有助於調整查詢的系統檢視主題和其他文章節：

- [STV_INFLIGHT](#) 系統檢視會顯示叢集上執行的查詢。將它與 [STV_RECENTS](#) 一起使用，以確定當前正在執行或最近完成的查詢會很有幫助。
- [SYS_QUERY_HISTORY](#) 對於疑難排解很有用。它會顯示具有相關屬性的 DDL/DML 查詢，例如其目前狀態，例如 `running` 或 `failed`、執行每個屬性所花費的時間，以及查詢是否在並行擴展叢集上執行。
- [STL_QUERYTEXT](#) 會擷取 SQL 命令的查詢文字。此外，將 [STL_QUERYTEXT](#) 連結至 [STV_SV_QUERY_INFLIGHT](#) 的 `INFLIGHT` 會顯示更多查詢中繼資料。
- 交易鎖定衝突可能是查詢效能問題的可能來源。如需目前在資料表上保留鎖定之交易的相關資訊，請參閱 [SVV_TRANSACTIONS](#)。
- [識別最適合調整的查詢](#) 會提供疑難排解查詢，協助您判斷最近執行的查詢最耗時。這可以幫助您將精力集中在需要改進的查詢上。
- 如果您想要進一步探索查詢管理並瞭解如何管理查詢佇列，[工作負載管理](#) 顯示如何執行此操作。工作負載管理是一項進階功能，我們建議您在大多數情況下自動化工作負載。

載入失敗

您的資料載入可能會失敗，原因如下。我們建議使用以下疑難排解方法。

資料來源位於不同 AWS 區域

根據預設，在 COPY 命令中指定的 Amazon S3 儲存貯體或 Amazon DynamoDB 資料表必須與叢集位於相同的 AWS 區域中。如果您的資料和您的叢集位於不同的區域，您會收到類似以下的錯誤：

```
The bucket you are attempting to access must be addressed using the specified endpoint.
```

如果有可能，請確定您的叢集和您的資料來源位於相同區域。您可以使用 [REGION](#) 選項搭配 COPY 命令來指定不同的區域。

Note

如果您的叢集和資料來源位於不同 AWS 區域，則會產生資料傳輸成本。您的延遲也會更高。

COPY 命令失敗

查詢 `STL_LOAD_ERRORS` 以探索特定載入期間發生的錯誤。如需詳細資訊，請參閱 [STL_LOAD_ERRORS](#)。

載入耗費太長時間

由於下列原因，您的載入操作可能需要太長的時間。我們建議使用以下疑難排解方法。

COPY 從單一檔案載入資料

將您的載入資料分割至多個檔案。當您從多個檔案載入所有資料時，Amazon Redshift 將被迫執行序列化載入，其速度非常慢。檔案的數量應該是您的叢集中配量數量的倍數，而檔案應該是大約相等的大小，壓縮後介於 1 MB 與 1 GB 之間。如需詳細資訊，請參閱 [設計查詢的 Amazon Redshift 最佳實務](#)。

載入操作使用多個 COPY 命令

如果您使用多個並行 COPY 命令從多個檔案載入一個資料表，Amazon Redshift 會強制執行序列化載入，速度會慢得多。在此情況下，請使用單一 COPY 命令。

載入資料不正確

您的 COPY 操作可以下列方式載入不正確的資料。我們建議使用以下疑難排解方法。

載入錯誤的檔案

使用物件字首來指定資料檔案可能導致讀取不需要的檔案。相反地，請使用資訊清單檔案來指定要載入的確切檔案。如需詳細資訊，請參閱 [Example: COPY from Amazon S3 using a manifest](#) COPY 範例中命令和 [copy_from_s3_manifest_file](#) 的選項。

設定 JDBC 擷取大小參數

根據預設，JDBC 驅動程式會一次收集查詢的所有結果。因此，當您嘗試透過 JDBC 連線擷取大型結果集時，您可能會遇到用戶端 out-of-memory 錯誤。若要讓用戶端以批次擷取結果集，而不是在單一 all-or-nothing 擷取中擷取結果集，請在用戶端應用程式中設定 JDBC 擷取大小參數。

Note

不支援擷取大小ODBC。

為求最佳效能，請將擷取大小設定為不會導致記憶體不足錯誤的最高值。較低的擷取大小值會造成更多伺服器來回行程，進而延長執行時間。伺服器會保留資源，包括WLM查詢插槽和相關聯的記憶體，直到用戶端擷取整個結果集或取消查詢為止。適當地調校擷取大小時，那些資源會更快速釋出，使得它們可供其他查詢使用。

Note

如果您需要擷取大型資料集，我們建議您使用 [UNLOAD](#) 陳述式將資料傳輸至 Amazon S3。當您使用 UNLOAD，運算節點會平行運作，以加快資料傳輸。

如需設定JDBC擷取大小參數的詳細資訊，請前往 PostgreSQL 文件中的 [根據游標取得結果](#)。

工作負載管理

您可以將 Amazon Redshift WLM 設定為使用自動或 WLM 手動 執行 WLM。

透過 Amazon Redshift，您可以管理和排定並行查詢和使用者工作負載的優先順序，以最佳化效能和資源使用率。工作負載管理（WLM）可讓您定義佇列、使用者群組和其他建構，以控制配置給不同類型查詢或使用者的資源。

下列各節概述 Amazon Redshift 中的特定工作負載管理功能，並引導您完成其組態和監控。

自動 WLM

若要最大化系統輸送量並有效使用資源，您可以讓 Amazon Redshift 管理資源的分割方式，以使用自動 執行並行查詢 WLM。自動 WLM 管理執行查詢所需的資源。Amazon Redshift 會決定多少個查詢並行執行和分配多少記憶體給每個分派的查詢。當您想要 Amazon Redshift 管理資源的分割方式以執行並行查詢 WLM 時，請使用 Auto。如需詳細資訊，請參閱 [實作自動 WLM](#)。

使用並行擴展和自動 WLM，您可以支援幾乎無限制的並行使用者和並行查詢，同時具有一致的快速查詢效能。如需詳細資訊，請參閱 [並行擴展](#)。

Note

在大多數情況下，我們建議您使用自動 WLM。如果您使用的是手動 WLM 而且想要從 遷移至自動 WLM，請參閱 [從手動遷移 WLM 至自動 WLM](#)。

使用 Auto WLM，可以為佇列中的工作負載定義查詢優先順序。如需查詢優先順序的相關資訊，請參閱 [查詢優先順序](#)。

手動 WLM

您可能有多個工作階段或使用者同時執行查詢。有些查詢可能會長時間使用叢集資源，並影響其他查詢的效能。手動 WLM 可協助管理特殊使用案例。當您想要對並行進行更多控制 WLM 時，請使用 手動。

您可以修改 WLM 組態來為長時間執行的查詢和短時間執行的查詢建立個別佇列，藉此管理系統效能。在執行時間，您可以根據使用者群組或查詢群組，將查詢路由至這些佇列。

您可以設定規則，以根據執行查詢的使用者，或您指定的標籤，將查詢路由至特定的佇列。您也可以設定要配置給每個佇列的記憶體數量，讓大型查詢在記憶體較多的佇列中執行。您也可以設定查詢監控規則（QMR），以限制長時間執行的查詢。如需詳細資訊，請參閱 [實作手冊 WLM](#)。

Note

我們建議您設定手動WLM查詢佇列，總共有 15 個或更少的查詢插槽。如需詳細資訊，請參閱[並行層級](#)。

請注意，對於手動WLM組態，您可以配置給佇列的插槽上限為 50 個。不過，這並不表示在自動WLM組態中，Amazon Redshift 叢集一律會同時執行 50 個查詢。這可能會根據叢集上的記憶體需求或其他類型的資源配置而變更。

主題

- [切換WLM模式](#)
- [修改WLM組態](#)
- [實作自動 WLM](#)
- [實作手冊 WLM](#)
- [並行擴展](#)
- [短期查詢加速](#)
- [WLM 佇列指派規則](#)
- [將查詢指派給佇列](#)
- [WLM 動態和靜態組態屬性](#)
- [WLM 查詢監控規則](#)
- [WLM 系統資料表和檢視](#)

切換WLM模式

您可以使用 Amazon Redshift WLM 主控台啟用自動或手動：

1. 選擇 [切換WLM模式](#)。
2. 若要將其設定為自動 WLM，請選擇自動 WLM。使用此選擇時，最多使用八個佇列來管理查詢，而且 Memory (記憶體) 和 Concurrency on main (主要叢集的並行) 欄位都設為 Auto (自動)。此外，查詢的預設優先順序會設定為 Normal。
3. 若要使用 Amazon Redshift 主控台啟用手動組態，請切換到手動 WLM。使用此選擇時，請指定用來管理查詢的佇列，以及 Memory (記憶體) 和 Concurrency on main (主要叢集的並行) 欄位值。使用手動組態時，您最多可以設定八個查詢佇列，並設定每個佇列中可同時執行的查詢數。

修改WLM組態

修改WLM組態的最簡單方法是使用 Amazon Redshift 主控台。您也可以使用 AWS CLI 或 Amazon Redshift API。

當您在自動和手動 之間切換叢集時WLM，您的叢集會進入 pending reboot 狀態。變更要在下次叢集重新啟動後才會生效。

如需修改WLM組態的詳細資訊，請參閱 Amazon Redshift [管理指南中的設定工作負載管理](#)。

從手動遷移WLM至自動 WLM

若要最大化系統輸送量並最有效地使用資源，建議您WLM為佇列設定自動。考慮採取下列方法，設定從手動順利轉換WLM至自動 WLM。

若要從手動遷移WLM至自動WLM並使用查詢優先順序，建議您建立新的參數群組，然後將該參數群組連接至叢集。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 參數群組](#)。

Important

若要變更參數群組或從手動切換到自動，WLM需要重新開機叢集。如需詳細資訊，請參閱 [WLM 動態和靜態組態屬性](#)。

讓我們舉一個範例，其中有三個手動WLM佇列。每個ETL工作負載、分析工作負載和資料科學工作負載各一個。ETL 工作負載每 6 小時執行一次，分析工作負載會全天執行一次，而資料科學工作負載可能會隨時飆升。使用手動 時WLM，您可以根據您了解每個工作負載對業務的重要性，指定每個工作負載佇列取得的記憶體和並行。指定記憶體和並行不僅很難釐清，也會導致叢集資源被靜態分割，因而在僅執行一部分工作負載時造成浪費。

您可以使用自動WLM查詢優先順序來指示工作負載的相對優先順序，避免上述問題。對於這個範例，請依照下列步驟進行：

- 建立新的參數群組並切換至自動WLM模式。
- 新增三種工作負載的佇列：ETL工作負載、分析工作負載和資料科學工作負載。針對與手動WLM模式搭配使用的每個工作負載使用相同的使用者群組。
- 將ETL工作負載的優先順序設定為 High，將分析工作負載設定為 Normal，將資料科學設定為 Low。這些優先順序反映了不同工作負載或使用群組的業務優先順序。

- 或者，為分析或資料科學佇列啟用並行擴展，以便即使ETL工作負載每 6 小時執行一次，這些佇列中的查詢也能獲得一致的效能。

運用查詢優先順序，當只有分析工作負載在叢集上執行時，它可取得整個系統。這可以產生高輸送量和最佳的系統使用率。不過，當ETL工作負載開始時，它就有方向權，因為其優先順序較高。作為ETL工作負載一部分執行的查詢在入院期間除了優先資源配置之外，還會在入院後獲得優先權。因此，無論系統上執行的其他項目為何，ETL工作負載都會有可預測的執行。高優先順序工作負載能獲得可預測的效能，代價是其他較低優先順序工作負載的執行時間更久，因為它們的查詢要等候較重要的查詢先完成；或者，當它們與較高優先順序的查詢同時執行時，所獲得的資源較少。Amazon Redshift 所用的排程演算法可使較低優先順序的查詢不會面臨資源耗盡，而是持續取得進展，只是速度較慢而已。

Note

- 自動中無法使用逾時欄位WLM。反之，請使用QMR規則 `query_execution_time`。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。
- QMR 動作 HOP不適用於自動 WLM。請改用 `change priority` 動作。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。
- 叢集使用自動WLM和手動WLM佇列的方式不同，這可能會導致您的組態混淆。例如，您可以在自動WLM佇列中設定優先順序屬性，但無法在手動WLM佇列中設定。因此，請避免在參數群組中混合自動WLM佇列和手動WLM佇列。請改為在遷移至自動時建立新的參數群組WLM。

實作自動 WLM

透過自動工作負載管理 (WLM)，Amazon Redshift 可管理查詢並行和記憶體配置。您可以使用服務類別識別碼 100-107 建立最多八個佇列。每個佇列都有優先順序。如需詳細資訊，請參閱[查詢優先順序](#)。

自動WLM決定查詢所需的資源量，並根據工作負載調整並行。當系統中有需要大量資源的查詢時 (例如，大型資料表之間的雜湊聯結)，並行較低。提出較輕型的查詢時 (例如，插入、刪除、掃描或簡易的彙總)，並行較高。

自動WLM與短查詢加速 (SQA) 分開，並以不同的方式評估查詢。自動WLM並協同SQA運作，即使長時間執行，也能完成短執行和輕量型查詢，資源密集型查詢也會處於作用中狀態。如需的詳細資訊 SQA，請參閱 [短期查詢加速](#)。

Amazon Redshift WLM 會透過參數群組啟用自動：

- 如果您的叢集使用預設參數群組，Amazon Redshift 會WLM為其啟用自動。
- 如果您的叢集使用自訂參數群組，您可以設定叢集以啟用自動 WLM。建議您為自動WLM組態建立個別的參數群組。

若要設定 WLM，請在可以與一或多個叢集相關聯的`wlm_json_configuration`參數群組中編輯參數。如需詳細資訊，請參閱[修改WLM組態](#)。

您可以在WLM組態中定義查詢佇列。您可以將其他查詢佇列新增至預設WLM組態，最多總共八個使用者佇列。您可以為每個查詢佇列設定下列項目：

- 優先順序
- 並行擴展模式
- 使用者群組
- 查詢群組
- 查詢監控規則

優先順序

您可以透過設定優先順序值來定義查詢在工作負載中的相對重要性。為佇列指定優先順序，與佇列關聯的所有查詢就會繼承其優先順序。如需詳細資訊，請參閱[查詢優先順序](#)。

並行擴展模式

當並行擴展啟用時，當您需要更多叢集容量以執行增加的並行讀取及寫入查詢時，Amazon Redshift 將會自動新增額外的叢集容量。您的使用者會看到最新資料，無論查詢是執行於主要叢集或並行擴展叢集。

您可以透過設定WLM佇列來管理要傳送至並行擴展叢集的查詢。當您為佇列啟用並行擴展時，合格查詢將傳送到並行擴展叢集，而非在佇列中等待。如需詳細資訊，請參閱[並行擴展](#)。

使用者群組

您可以指定每一個使用者群組名稱或使用萬用字元，將一組使用者群組指派給佇列。當所列的使用者群組中有一個成員執行查詢時，該查詢會在相應佇列中執行。可指派給佇列的使用者群組數沒有固定的限制。如需詳細資訊，請參閱[根據使用者群組將查詢指派給佇列](#)。

查詢群組

您可以指定每一個查詢群組名稱或使用萬用字元，將一組查詢群組指派給佇列。查詢群組只是一個標籤。在執行時間，您可以將查詢群組標籤指派給一系列查詢。指派給所列之查詢群組的任何查詢會在相應佇列中執行。可指派給佇列的查詢群組數沒有固定的限制。如需詳細資訊，請參閱[將查詢指派給查詢群組](#)。

萬用字元

如果在WLM佇列組態中啟用萬用字元，您可以個別指派使用者群組和查詢群組給佇列，也可以使用 Unix shell 樣式萬用字元。模式比對不區分大小寫。

例如，'*' 萬用字元符合任意數目的字元。因此，如果您將 dba_* 新增至佇列的使用者群組清單，任何使用者執行查詢只要是屬於以 dba_ 為名稱開頭的群組，都指派到該佇列。例如 dba_admin 或 DBA_primary。 '?' 萬用字元符合任何單一字元。因此，如果佇列包含使用者群組 dba?1，則名為 dba11 和 dba21 的使用者群組符合，但 dba12 不符。

預設情況下，不會啟用萬用字元。

查詢監控規則

查詢監控規則會定義WLM佇列的指標型效能界限，並指定查詢超出這些界限時要採取的動作。例如，對於短期執行查詢專用的佇列，您可以建立規則，以將執行時間超過 60 秒的查詢取消。若要追蹤設計不良的查詢，您可以使用另一個規則來記錄含有巢狀迴圈的查詢。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。

檢查是否有自動 WLM

若要檢查是否WLM已啟用自動，請執行下列查詢。如果查詢至少傳回一行，則會WLM啟用自動。

```
select * from stv_wlm_service_class_config
where service_class >= 100;
```

下列查詢顯示通過每個查詢佇列 (服務類別) 的查詢數。也顯示平均執行時間、第 90 個百分位數的查詢數和等待時間，以及平均等待時間。自動WLM查詢使用服務類別 100 到 107。

```
select final_state, service_class, count(*), avg(total_exec_time),
percentile_cont(0.9) within group (order by total_queue_time), avg(total_queue_time)
```

```
from stl_wlm_query where userid >= 100 group by 1,2 order by 2,1;
```

若要尋找哪些查詢是由自動執行WLM並順利完成，請執行下列查詢。

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class >= 100 and a.final_state = 'Completed'
order by b.query desc limit 5;
```

查詢優先順序

使用 Amazon Redshift，您可以使用工作負載管理（WM）管理並行查詢和工作負載之間的查詢優先順序和資源配置。下列各節詳細說明如何設定 WM 查詢佇列、定義記憶體配置和並行擴展等佇列屬性，以及實作根據您的工作負載需求量身打造的優先順序規則。

並非所有查詢都具有相同的重要性，通常某个工作負載或某一組使用者的效能會比較重要。如果您已啟用[自動 WLM](#)，您可以設定優先順序值，定義工作負載中查詢的相對重要性。為佇列指定優先順序，與佇列關聯的所有查詢就會繼承其優先順序。將使用者群組和查詢群組對應到佇列，即可將查詢關聯至佇列。您可以設定以下優先順序（從最高優先順序到最低優先順序列出）：

1. HIGHEST
2. HIGH
3. NORMAL
4. LOW
5. LOWEST

當優先順序不同的查詢爭用相同資源時，管理員使用這些優先順序來顯示其工作負載的相對重要性。Amazon Redshift 會採用優先順序來將查詢放入系統，以及決定分配給查詢的資源量。依預設，查詢的優先順序設定為 NORMAL。

超級使用者可使用一個額外的優先順序 CRITICAL，這是比 HIGHEST 更高的優先順序。若要設定此優先順序，您可使用函數 [CHANGE_QUERY_PRIORITY](#)、[CHANGE_SESSION_PRIORITY](#) 和 [CHANGE_USER_PRIORITY](#)。若要將使用這些函數的許可授予給資料庫使用者，您可以建立預存程序並授予許可給使用者。如需範例，請參閱[CHANGE_SESSION_PRIORITY](#)。

Note

一次只能執行一個 CRITICAL 查詢。

讓我們舉一個範例，其中擷取、轉換、載入（ETL）工作負載的優先順序高於分析工作負載的優先順序。ETL 工作負載每 6 小時執行一次，分析工作負載則全天執行一次。當只有分析工作負載在叢集上執行時，它可取得整個系統，產生高輸送量和最佳的系統使用率。但是，當 ETL 工作負載開始時，它就有辦法，因為它具有更高的優先順序。作為 ETL 工作負載一部分執行的查詢，在入院期間取得路徑，在入院後也取得優先資源配置。因此，無論系統上執行的其他項目為何，ETL 工作負載都會有可預測的執行。因此，它提供可預測的效能，以及管理員為其商業使用者提供服務層級協議（SLAs）的能力。

在指定叢集中，高優先順序工作負載能獲得可預測的效能，代價是其他較低優先順序的工作負載。較低優先順序的工作負載可能會執行得更久，因為它們的查詢要等候較重要的查詢先完成。或者，當它們與較高優先順序的查詢同時執行時，所獲得的資源較少，所以要執行得更久。較低優先順序的查詢不會面臨資源耗盡，只是以較慢的速度取得進展。

在前面的範例中，管理員可以為分析工作負載啟用[並行擴展](#)。這樣做可讓工作負載維持其輸送量，即使 ETL 工作負載以高優先順序執行。

設定佇列優先順序

如果您已啟用自動 WLM，則每個佇列都有優先順序值。查詢會根據使用者群組和查詢群組而路由到佇列。從將佇列優先順序設定為 NORMAL 開始。請根據與佇列之使用者群組和查詢群組關聯的工作負載，設定更高或更低的優先順序。

您可以在 Amazon Redshift 主控台上變更佇列的優先順序。在 Amazon Redshift 主控台上，工作負載管理頁面會顯示佇列並啟用佇列屬性的編輯，例如優先順序。若要使用 CLI 或 API 操作設定優先順序，請使用 `wlm_json_configuration` 參數。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[設定工作負載管理](#)。

下列 `wlm_json_configuration` 範例定義三個使用者群組 (`ingest`、`reporting` 和 `analytics`)。來自這些群組的使用者所提交的查詢分別以 `highest`、`normal` 和 `low` 的優先順序執行。

```
[
  {
    "user_group": [
      "ingest"
    ],
    "priority": "highest",
    "queue_type": "auto"
  },
  {
    "user_group": [
      "reporting"
```

```

    ],
    "priority": "normal",
    "queue_type": "auto"
  },
  {
    "user_group": [
      "analytics"
    ],
    "priority": "low",
    "queue_type": "auto",
    "auto_wlm": true
  }
]

```

使用查詢監控規則變更查詢優先順序

查詢監控規則（QMR）可讓您根據查詢執行時的行為變更查詢的優先順序。除了動作之外，您還可以在QMR述詞中指定優先順序屬性來達成此目的。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。

例如，您可以定義規則，來取消任何分類為 high 優先順序且執行超過 10 分鐘的查詢。

```

"rules" :[
  {
    "rule_name":"rule_abort",
    "predicate":[
      {
        "metric_name":"query_cpu_time",
        "operator":">",
        "value":600
      },
      {
        "metric_name":"query_priority",
        "operator":"=",
        "value":"high"
      }
    ],
    "action":"abort"
  }
]

```

另一個範例是定義規則以將目前優先順序為 normal 且溢出超過 1 TB 到磁碟之任何查詢的優先順序變更為 lowest。


```

"rules":[
  {
    "rule_name":"rule_change_priority",
    "predicate":[
      {
        "metric_name":"query_temp_blocks_to_disk",
        "operator":">",
        "value":1000000
      },
      {
        "metric_name":"query_priority",
        "operator":"=",
        "value":"normal"
      }
    ],
    "action":"change_query_priority",
    "value":"lowest"
  }
]

```

監控查詢優先順序

若要顯示等待中和執行中查詢的優先順序，請檢視 `stv_wlm_query_state` 系統資料表中的 `query_priority` 欄。

query	service_cl	wlm_start_time	state	queue_time
2673299	102	2019-06-24 17:35:38.866356	QueuedWaiting	265116
Highest				
2673236	101	2019-06-24 17:35:33.313854	Running	0
Highest				
2673265	102	2019-06-24 17:35:33.523332	Running	0
High				
2673284	102	2019-06-24 17:35:38.477366	Running	0
Highest				
2673288	102	2019-06-24 17:35:38.621819	Running	0
Highest				
2673310	103	2019-06-24 17:35:39.068513	QueuedWaiting	62970
High				

2673303	102	2019-06-24 17:35:38.968921	QueuedWaiting	162560	
Normal					
2673306	104	2019-06-24 17:35:39.002733	QueuedWaiting	128691	
Lowest					

若要列出已完成查詢的優先順序，請檢視 `stl_wlm_query` 系統資料表中的 `query_priority` 欄。

```
select query, service_class as svclass, service_class_start_time as starttime,
       query_priority
from stl_wlm_query order by 3 desc limit 10;
```

query	svclass	starttime	query_priority
2723254	100	2019-06-24 18:14:50.780094	Normal
2723251	102	2019-06-24 18:14:50.749961	Highest
2723246	102	2019-06-24 18:14:50.725275	Highest
2723244	103	2019-06-24 18:14:50.719241	High
2723243	101	2019-06-24 18:14:50.699325	Low
2723242	102	2019-06-24 18:14:50.692573	Highest
2723239	101	2019-06-24 18:14:50.668535	Low
2723237	102	2019-06-24 18:14:50.661918	Highest
2723236	102	2019-06-24 18:14:50.643636	Highest

為了最佳化工作負載的輸送量，Amazon Redshift 可能會修改使用者所提交查詢的優先順序。Amazon Redshift 會使用進階機器學習演算法來判斷此最佳化何時有利於您的工作負載，並在符合下列所有條件時自動套用。

- WLM 已啟用自動。
- 只會定義一個WLM佇列。
- 您尚未定義可設定查詢優先順序的查詢監控規則（QMRs）。這些規則包括QMR指標 `query_priority` 或 QMR 動作 `change_query_priority`。如需詳細資訊，請參閱 [WLM 查詢監控規則](#)。

實作手冊 WLM

透過手動 WLM，您可以修改WLM組態，為長時間執行的查詢和短時間執行的查詢建立個別佇列，藉此管理系統效能和使用者體驗。

當使用者在 Amazon Redshift 中執行查詢時，會將查詢路由至查詢佇列。每個查詢佇列包含一些查詢槽。每個佇列都配置有叢集的一部分可用的記憶體。佇列的記憶體再分配給佇列的查詢槽。您可以啟用 Amazon Redshift，以使用自動管理查詢並行 WLM。如需詳細資訊，請參閱[實作自動 WLM](#)。

或者，您可以為每個查詢佇列設定 WLM 屬性。這麼做可以指定各個槽分配記憶體的方式，以及在執行期將查詢路由至特定佇列的方式。您也可以設定 WLM 屬性來取消長時間執行的查詢。

根據預設，Amazon Redshift 會設定以下查詢佇列：

- 一個超級使用者佇列

超級使用者佇列會保留給超級使用者專用，因此無法進行設定。只有在需要執行會影響系統的查詢，或基於故障診斷目的，才應該使用此佇列。例如，當您需要取消使用者長期執行的查詢，或需要將使用者新增至資料庫時，請使用此佇列。請勿用來執行例行性查詢。此佇列不會出現主控台，而是在資料庫的系統資料表中以第五個佇列出現。若要在超級使用者佇列中執行查詢，使用者必須以超級使用者身分登入，且必須使用預先定義的 `superuser` 查詢群組來執行查詢。

- 一個預設使用者佇列

預設佇列最初會設定為同時執行五個查詢。使用手動時 WLM，您可以變更預設佇列的並行、逾時和記憶體配置屬性，但無法指定使用者群組或查詢群組。預設佇列必須是 WLM 組態中的最後一個佇列。任何未路由至其他佇列的查詢會在預設佇列中執行。

查詢佇列在 WLM 組態中定義。WLM 組態是參數群組中的可編輯參數

(`wlm_json_configuration`)，可以與一或多個叢集建立關聯。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[設定工作負載管理](#)。

您可以將其他查詢佇列新增至預設 WLM 組態，最多總共八個使用者佇列。您可以為每個查詢佇列設定下列項目：

- 並行擴展模式
- 並行層級
- 使用者群組
- 查詢群組
- WLM 要使用的記憶體百分比
- WLM 逾時
- WLM 查詢佇列跳轉
- 查詢監控規則

並行擴展模式

當並行擴展啟用時，當您需要更多叢集容量以執行增加的並行讀取及寫入查詢時，Amazon Redshift 將會自動新增額外的叢集容量。使用者會看到最新資料，無論查詢是執行於主要叢集或並行擴展叢集。

您可以透過設定WLM佇列來管理要傳送至並行擴展叢集的查詢。當您為佇列啟用並行擴展時，合格查詢將傳送到並行擴展叢集，而非在佇列中等待。如需詳細資訊，請參閱[並行擴展](#)。

並行層級

佇列中的查詢會同時執行，直到達到WLM針對該佇列定義的查詢槽計數或並行層級為止。後續的查詢需要在佇列中等待。

Note

WLM 並行層級與可對叢集建立的並行使用者連線數目不同。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[連線到叢集](#)。

在建議使用的自動WLM組態中，並行層級設定為 Auto。Amazon Redshift 會動態地將記憶體配置給查詢，然後決定要同時執行多少個查詢。這是基於執行中和佇列查詢所需的資源。Auto WLM無法設定。如需詳細資訊，請參閱[實作自動 WLM](#)。

在手動WLM組態中，Amazon Redshift 會為每個佇列靜態配置固定數量的記憶體。佇列的記憶體會在查詢插槽之間平均分割。說明如下，如果佇列分配到 20% 的叢集記憶體，並且具有 10 個插槽，則每個查詢都會分配到 2% 的叢集記憶體。無論同時執行的查詢數目為何，記憶體配置都會保持固定。因為此固定的記憶體配置，如果查詢在槽計數為 5 時可完全在記憶體中執行，當槽計數增加到 20 時，就可能將中間結果寫入磁碟。在這種情況下，每個查詢的佇列記憶體共用會從 1/5 減少到 1/20。增加的磁碟 I/O 會降低效能。

所有使用者定義佇列上的插槽計數上限為 50 個。這會限制所有佇列 (包括預設佇列) 的總插槽數。唯一不受限制限制的佇列是保留的超級使用者佇列。

根據預設，手動WLM佇列的並行層級為 5。在某些情況下，較高的並行層級可能有利於工作負載，例如：

- 如果許多較小的查詢被迫等待長期執行的查詢，請建立槽計數較高的另一個佇列，並將較小的查詢指派至該佇列。並行層級較高的佇列配置給每個查詢槽的記憶體較少，但較小的查詢所需的記憶體不多。

Note

如果您啟用短查詢加速（SQA），WLM會自動將短查詢優先於長時間執行的查詢，因此大多數工作流程不需要個別的短查詢佇列。如需詳細資訊，請參閱[短期查詢加速](#)。

- 如果您有多個查詢，每個存取單一切片上的資料，請設定個別WLM佇列以同時執行這些查詢。Amazon Redshift 會將並行查詢指派給不同配量，以便在多個配量上平行執行多個查詢。例如，假設查詢是在散發索引鍵上搭配述詞執行的簡單彙總，則查詢的資料會位於單一配量上。

手動WLM範例

此範例是簡單的手動WLM案例，示範如何配置插槽和記憶體。您可以使用WLM三個佇列實作手冊，如下所示：

- 資料擷取佇列 – 這是針對擷取資料而設定的。這會分配 20% 的叢集記憶體，並且具有 5 個插槽。隨後，5 個查詢可以在佇列中同時運行，並且每個查詢會分配到 4% 的記憶體。
- 資料科學家佇列 - 這是專為記憶體密集型查詢而設計的。這會分配 40% 的叢集記憶體，並且具有 5 個插槽。隨後，5 個查詢可以同時運行，並且每個查詢會分配到 8% 的記憶體。
- 預設佇列 – 這是針對組織中大多數使用者所設計的。這包括銷售和會計群組，這些群組通常具有不複雜的簡短或中型執行查詢。它會分配到 40% 的叢集記憶體，並且具有 40 個插槽。40 個查詢可以在此佇列中同時執行，而每個查詢會分配到 1% 的記憶體。這是可配置給此佇列的插槽數目上限，因為所有佇列之間的限制為 50 個。

如果您正在自動執行，WLM且工作負載需要超過 15 個查詢才能平行執行，建議您開啟並行擴展。這是因為將查詢槽計數增加到 15 以上可能會導致系統資源爭用，並限制單一叢集的整體輸送量。使用並行擴展，您可以平行執行數百個查詢，最多可達設定的並行擴展叢集數目。並行擴展叢集數目由 [max_concurrency_scaling_clusters](#) 控制。如需並行擴展的相關資訊，請參閱 [並行擴展](#)。

如需詳細資訊，請參閱[查詢效能改善](#)。

使用者群組

您可以指定每一個使用者群組名稱或使用萬用字元，將一組使用者群組指派給佇列。當所列的使用者群組中有一個成員執行查詢時，該查詢會在相應佇列中執行。可指派給佇列的使用者群組數沒有固定的限制。如需詳細資訊，請參閱[根據使用者群組將查詢指派給佇列](#)。

查詢群組

您可以指定每一個查詢群組名稱或使用萬用字元，將一組查詢群組指派給佇列。查詢群組只是一個標籤。在執行時間，您可以將查詢群組標籤指派給一系列查詢。指派給所列之查詢群組的任何查詢會在相應佇列中執行。可指派給佇列的查詢群組數沒有固定的限制。如需詳細資訊，請參閱[將查詢指派給查詢群組](#)。

萬用字元

如果在WLM佇列組態中啟用萬用字元，您可以個別指派使用者群組和查詢群組給佇列，也可以使用 Unix shell 樣式萬用字元。模式比對不區分大小寫。

例如，'*' 萬用字元符合任意數目的字元。因此，如果您將 dba_* 新增至佇列的使用者群組清單，任何使用者執行查詢只要是屬於以 dba_ 為名稱開頭的群組，都指派到該佇列。例如 dba_admin 或 DBA_primary。 '?' 萬用字元符合任何單一字元。因此，如果佇列包含使用者群組 dba?1，則名為 dba11 和 dba21 的使用者群組符合，但 dba12 不符。

依預設，萬用字元處於關閉狀態。

WLM 要使用的記憶體百分比

在自動WLM組態中，記憶體百分比設定為 **auto**。如需詳細資訊，請參閱[實作自動 WLM](#)。

在手動WLM組態中，若要指定配置給查詢的可用記憶體量，您可以設定 WLM Memory Percent to Use 參數。根據預設，每個使用者定義佇列都配置有相等份額的記憶體，可供使用者定義的查詢使用。例如，假設您有四個使用者定義佇列，則會配置可用記憶體的 25% 給每一個佇列。超級使用者佇列有其自己配置的記憶體，無法修改。若要變更配置，請將記憶體的整數百分比指派給每個佇列，總計最多 100%。任何未配置的記憶體由 Amazon Redshift 管理並可暫時提供給佇列 (如果佇列要求更多記憶體來進行處理)。

例如，假設您設定四個佇列，則可以如下配置記憶體：20%、30%、15%、15%。剩餘 20% 未配置，由服務管理。

WLM 逾時

WLM 逾時 (max_execution_time) 已棄用。反之，請使用 建立查詢監控規則 (QMR) query_execution_time，以限制查詢的已耗用執行時間。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。

若要限制特定WLM佇列中的查詢允許使用的時間量，您可以為每個佇列設定WLM逾時值。逾時參數指定 Amazon Redshift 在取消或轉跳查詢之前，等待查詢開始執行的時間量 (以毫秒為單位)。逾時以查詢執行時間為基礎，不包括在佇列中等待所花的時間。

WLM 嘗試跳轉 [CREATE TABLE AS \(CTAS \)](#) 陳述式和唯讀查詢，例如SELECT陳述式。無法轉跳的查詢會取消。如需詳細資訊，請參閱[WLM 查詢佇列跳轉](#)。

WLM 逾時不適用於已達到傳回狀態的查詢。若要檢視查詢的狀態，請參閱 [STV_WLM_QUERY_STATE](#) 系統資料表。COPY 陳述式和維護操作，例如 ANALYZE和 VACUUM，不受WLM逾時限制。

WLM 逾時的函數類似於[statement_timeout](#)組態參數。不同之處在於，當statement_timeout組態參數套用至整個叢集時，WLM逾時是WLM組態中單一佇列特有的。

如果 [statement_timeout](#) 也指定，則會使用 statement_timeout 和 WLM timeout (max_execution_time) 的較低值。

查詢監控規則

查詢監控規則會定義WLM佇列的指標型效能界限，並指定查詢超出這些界限時要採取的動作。例如，對於短期執行查詢專用的佇列，您可以建立規則，以將執行時間超過 60 秒的查詢取消。若要追蹤設計不良的查詢，您可以使用另一個規則來記錄含有巢狀迴圈的查詢。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。

WLM 查詢佇列跳轉

使用 Amazon Redshift，您可以透過啟用 WLM (工作負載管理) 查詢佇列跳轉來管理工作負載並行和資源配置。此功能允許查詢在資源可用時暫時從指派佇列「跳出」至較高優先順序佇列，從而改善整體查詢效能和系統使用率。下列各節提供在 Amazon Redshift 中設定和使用WLM查詢佇列跳轉的詳細指引。

由於[WLM逾時](#)或查詢[監控規則 \(QMR \) 跳轉動作](#)，[可以跳轉查詢](#)。您只能在手動WLM組態中跳轉查詢。

跳出查詢時，WLM會根據佇列[WLM指派規則 嘗試將查詢路由至下一個相符的佇列](#)。如果查詢不符合其他任何佇列定義，則會取消此查詢。不會指派給預設佇列。

WLM 逾時動作

下表摘要說明具有WLM逾時的不同類型查詢的行為。

查詢類型	動作
INSERT、UPDATE和 DELETE	取消
使用者定義的函數 (UDFs)	取消
UNLOAD	取消
COPY	繼續執行
維護操作	繼續執行
處於 returning 狀態的唯讀查詢	繼續執行
處於 running 狀態的唯讀查詢	重新指派或重新啟動
CREATE TABLE AS (CTAS)、SELECT INTO	重新指派或重新啟動

WLM 逾時佇列跳轉

WLM 逾時時跳出下列類型的查詢：

- 處於 WLM 狀態的唯讀查詢，例如 SELECT 陳述式 running。若要尋找查詢 WLM 的狀態，請檢視 [STV_WLM_QUERY_STATE](#) 系統資料表上的資料 STATE 欄。
- CREATE TABLE AS (CTAS) 陳述式。WLM 佇列跳轉支援使用者定義和系統產生的 CTAS 陳述式。
- SELECT INTO 陳述式。

不受 WLM 逾時限制的查詢會繼續在原始佇列中執行，直到完成為止。下列類型的查詢不會 WLM 逾時：

- COPY 陳述式
- 維護操作，例如 ANALYZE 和 VACUUM
- 已達 WLM 狀態的唯讀查詢，例如 SELECT 陳述式 returning。若要尋找查詢 WLM 的狀態，請檢視 [STV_WLM_QUERY_STATE](#) 系統資料表上的資料 STATE 欄。

不符合 WLM 逾時跳轉資格的查詢會在逾時時取消。下列類型的查詢不符合 WLM 逾時跳轉的資格：

- INSERT、UPDATE和 DELETE陳述式
- UNLOAD 陳述式
- 使用者定義的函數（UDFs）

WLM 重新指派和重新啟動的逾時查詢

當查詢跳轉且找不到相符的佇列時，就會取消此查詢。

跳出查詢並找到相符佇列時，會WLM嘗試將查詢重新指派給新的佇列。如果無法重新指派查詢，則會在新佇列中重新啟動此查詢，如下所述。

只有在下列所有條件成立時，才會重新指派查詢：

- 找到相符的查詢。
- 新佇列有足夠可用的槽可執行查詢。如果 [wlm_query_slot_count](#) 參數設定的值大於 1，查詢可能需要多個槽。
- 新佇列可用的記憶體至少與查詢目前使用的記憶體一樣多。

如果重新指派查詢，此查詢會在新佇列中繼續執行。中間結果會保留下來，因此對於總執行時間影響極小。

如果無法重新指派查詢，則會取消查詢，並於新佇列中重新啟動查詢。將會刪除中間結果。查詢會在佇列中等待，然後在有足夠的槽可用時開始執行。

QMR 躍點動作

下表總結了具有QMR躍點動作的不同類型查詢的行為。

查詢類型	動作
COPY	繼續執行
維護操作	繼續執行
使用者定義的函數（UDFs）	繼續執行
UNLOAD	重新指派或繼續執行
INSERT、UPDATE和 DELETE	重新指派或繼續執行

查詢類型	動作
處於 returning 狀態的唯讀查詢	重新指派或繼續執行
處於 running 狀態的唯讀查詢	重新指派或重新啟動
CREATE TABLE AS (CTAS)、SELECT INTO	重新指派或重新啟動

若要尋找所跳出的查詢QMR是否已重新指派、重新啟動或取消，請查詢[STL_WLM_RULE_ACTION](#)系統日誌表。

QMR 躍點動作已重新指派和重新啟動的查詢

當查詢跳轉且找不到相符的佇列時，就會取消此查詢。

跳出查詢並找到相符佇列時，會WLM嘗試將查詢重新指派給新的佇列。如果無法重新指派查詢，此查詢會在新佇列中重新啟動，或在原始佇列中繼續執行，如下所述。

只有在下列所有條件成立時，才會重新指派查詢：

- 找到相符的查詢。
- 新佇列有足夠可用的槽可執行查詢。如果 [wlm_query_slot_count](#) 參數設定的值大於 1，查詢可能需要多個槽。
- 新佇列可用的記憶體至少與查詢目前使用的記憶體一樣多。

如果重新指派查詢，此查詢會在新佇列中繼續執行。中間結果會保留下來，因此對於總執行時間影響極小。

如果無法重新指派查詢，此查詢會重新啟動或在原始佇列中繼續執行。如果重新啟動查詢，則會取消查詢，並於新佇列中重新啟動查詢。將會刪除中間結果。查詢會在佇列中等待，然後在有足夠的槽可用時開始執行。

教學課程：設定手動工作負載管理（WLM）佇列

使用 Amazon Redshift，您可以設定手動工作負載管理（WLM）佇列，為不同類型的查詢和使用者优先順序並配置資源。手動WLM佇列可讓您控制特定佇列的記憶體和並行設定，確保關鍵工作負載接收必要的資源，同時防止低優先順序查詢壟斷系統。下列各節會引導您完成在 Amazon Redshift 中建立和設定手動WLM佇列的程序，以符合工作負載管理需求。

概觀

建議您在 Amazon Redshift 中設定自動工作負載管理 (WLM)。如需自動的詳細資訊 WLM，請參閱 [工作負載管理](#)。不過，如果您需要多個 WLM 佇列，本教學課程會逐步引導您在 Amazon Redshift 中設定手動工作負載管理 (WLM) 的程序。透過設定手動 WLM，您可以改善叢集中的查詢效能和資源配置。

Amazon Redshift 將使用者查詢路由到佇列以進行處理。WLM 定義這些查詢如何路由至佇列。Amazon Redshift 預設有兩個用於查詢的佇列：一個給超級使用者使用，一個給使用者使用。超級使用者佇列無法設定，且一次只能處理一個佇列。只有為了故障排除的目的才需要保留此佇列。使用者佇列一次最多可以處理五個查詢，但您可以視需要變更佇列的並行層級以改變設定。

如果有多個使用者對資料庫執行查詢，您可能會發現另一種設定更有效率。例如，如果某些使用者執行資源密集型操作，例如 VACUUM，這些操作可能會對較不密集的查詢產生負面影響，例如報告。您可以考慮加入額外的佇列，並設定它們處理不同的工作負載。

預估時間：75 分鐘

評估成本：50 美分

必要條件

您需要 Amazon Redshift 叢集、範例 TICKIT 資料庫和 Amazon Redshift RSQL 用戶端工具。如果您尚未設定這些設定，請前往 [Amazon Redshift 入門指南](#) 和 [Amazon Redshift RSQL](#)。

章節

- [第 1 節：了解預設佇列處理行為](#)
- [第 2 節：修改 WLM 查詢佇列組態](#)
- [第 3 節：根據使用者群組和查詢群組將查詢路由至佇列](#)
- [第 4 節：使用 `wlm_query_slot_count` 暫時覆寫佇列中的並行層級](#)
- [第 5 節：清理資源](#)

第 1 節：了解預設佇列處理行為

在您開始設定手動之前 WLM，了解 Amazon Redshift 中佇列處理的預設行為很有用。您將在本節建立兩個資料庫檢視，這些檢視會傳回多個系統資料表的資訊。然後，您將執行一些測試查詢，以查看預設情況下如何路由查詢。如需系統資料表的相關資訊，請參閱 [系統資料表和檢視參考](#)。

步驟 1：建立 WLM_QUEUE_STATE_VW 檢視

在此步驟中，您會建立名為 WLM_QUEUE_STATE_VW 的檢視。此檢視會回傳下列系統資料表的資訊。

- [STV_WLM_CLASSIFICATION_CONFIG](#)
- [STV_WLM_SERVICE_CLASS_CONFIG](#)
- [STV_WLM_SERVICE_CLASS_STATE](#)

您可以在教學課程中使用此檢視來監控變更WLM組態後佇列的情況。下表說明 WLM_QUEUE_STATE_VW 檢視傳回的資料。

資料欄	描述
佇列	與表示佇列的資料列關聯的數量。佇列數量決定了佇列在資料庫中的順序。
description	此值描述佇列是僅用於某些使用者群組、某些查詢群組、或是所有類型的查詢。
slots	配置給佇列的槽數量。
mem	配置給佇列的記憶體數量 (單位為 MB/槽)。
max_execution_time	在終止查詢之前允許查詢執行的時間。
user_*	指出WLM組態中是否允許萬用字元以符合使用者群組的值。
query_*	指出WLM組態中是否允許萬用字元以符合查詢群組的值。
queued	佇列中正在等待處理的查詢數量。
執行中	目前執行中的查詢數。
executed	已執行的佇列數量。

建立 WLM_QUEUE_STATE_VW 檢視

1. 開啟 [Amazon Redshift RSQL](#) 並連線至您的TICKIT範例資料庫。如果您沒有此資料庫，請參閱[必要條件](#)。
2. 執行下列查詢以建立 WLM_QUEUE_STATE_VW 檢視。

```
create view WLM_QUEUE_STATE_VW as
select (config.service_class-5) as queue
, trim (class.condition) as description
, config.num_query_tasks as slots
, config.query_working_mem as mem
, config.max_execution_time as max_time
, config.user_group_wild_card as "user_*"
, config.query_group_wild_card as "query_*"
, state.num_queued_queries queued
, state.num_executing_queries executing
, state.num_executed_queries executed
from
STV_WLM_CLASSIFICATION_CONFIG class,
STV_WLM_SERVICE_CLASS_CONFIG config,
STV_WLM_SERVICE_CLASS_STATE state
where
class.action_service_class = config.service_class
and class.action_service_class = state.service_class
and config.service_class > 4
order by config.service_class;
```

3. 執行下列查詢以查看檢視中包含的資訊。

```
select * from wlm_queue_state_vw;
```

以下是結果範例。

query	description	slots	mem	max_time	user_*
query_*	queued executing executed				
0	(super user) and (query group: superuser)	1	357	0	false
false	0 0 0				
1	(querytype:any)	5	836	0	false
false	0 1 160				

步驟 2：建立 WLM_QUERY_STATE_VW 檢視

在此步驟中，您會建立名為 WLM_QUERY_STATE_VW 的檢視。此檢視會回傳 [STV_WLM_QUERY_STATE](#) 系統資料表的資訊。

您將在整個教學課程中使用此檢視來監控執行中的佇列。下表說明 WLM_QUERY_STATE_VW 檢視傳回的資料。

資料欄	描述
query	查詢 ID。
佇列	佇列數量。
slot_count	配置給查詢的槽數量。
start_time	查詢開始的時間。
state	查詢的狀態，例如 executing (執行中)。
queue_time	查詢在佇列中花費的時間 (微秒)。
exec_time	查詢已執行的微秒數。

若要建立 WLM_QUERY_STATE_VW 檢視

1. 在中 RSQL，執行下列查詢以建立 WLM_QUERY_STATE_VW 檢視。

```
create view WLM_QUERY_STATE_VW as
select query, (service_class-5) as queue, slot_count, trim(wlm_start_time) as
start_time, trim(state) as state, trim(queue_time) as queue_time, trim(exec_time) as
exec_time
from stv_wlm_query_state;
```

2. 執行下列查詢以查看檢視中包含的資訊。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

```

query | queue | slot_count | start_time          | state      | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----
+-----+
1249 |      1 |           1 | 2014-09-24 22:19:16 | Executing | 0          | 516

```

步驟 3：執行測試查詢

在此步驟中，您可以從 中的多個連線執行查詢，RSQL並檢閱系統資料表，以判斷如何路由查詢進行處理。

在此步驟中，您需要開啟兩個RSQL視窗：

- 在RSQL視窗 1 中，您可以使用您在本教學課程中建立的檢視，執行監控佇列和查詢狀態的查詢。
- 在RSQL視窗 2 中，您會執行長時間執行的查詢，以變更您在RSQL視窗 1 中找到的結果。

執行測試查詢

1. 開啟兩個RSQL視窗。如果您已經開啟一個視窗，則只需再開啟第二個視窗。這兩個連線皆可使用相同的使用者帳戶。
2. 在RSQL視窗 1 中，執行下列查詢。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

```

query | queue | slot_count | start_time          | state      | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----
+-----+
1258 |      1 |           1 | 2014-09-24 22:21:03 | Executing | 0          | 549

```

此查詢會傳回自我參考的結果。目前正在執行的查詢是來自此檢視的 SELECT 陳述式。此檢視的查詢一律會至少傳回一個結果。比較此結果與在下一步驟中啟動的長時間執行查詢所產生的結果。

3. 在RSQL視窗 2 中，從TICKIT範例資料庫執行查詢。此查詢應執行約一分鐘，以便您有時間探索之前建立的 WLM_QUEUE_STATE_VW 檢視和 WLM_QUERY_STATE_VW 檢視的結果。在某些情況下，您可能發覺查詢執行時間不夠讓您查詢這兩個檢視。在這些情況下，您可以在 `l.listid` 上提高篩選條件的值，讓它執行更久。

Note

為了縮短查詢執行時間並改善系統效能，Amazon Redshift 會將特定查詢類型的結果快取在領導者節點的記憶體中。啟用結果快取後，後續的查詢會執行得更快。若要防止查詢執行過快，可停用目前工作階段的結果快取。

若要關閉目前工作階段的結果快取，將 [enable_result_cache_for_session](#) 參數設為 off，如下所示。

```
set enable_result_cache_for_session to off;
```

在RSQL視窗 2 中，執行下列查詢。

```
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <
100000;
```

- 在RSQL視窗 1 中，查詢 WLM_QUEUE_STATE_VW 和 WLM_QUERY_STATE_VW，並將結果與您先前的結果進行比較。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

```
query | description | slots | mem | max_time | user_* |
query_* | queued | executing | executed
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
0 | (super user) and (query group: superuser) | 1 | 357 | 0 | false |
false | 0 | 0 | 0
1 | (querytype:any) | 5 | 836 | 0 | false |
false | 0 | 2 | 163

query | queue | slot_count | start_time | state | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
1267 | 1 | 1 | 2014-09-24 22:22:30 | Executing | 0 | 684
1265 | 1 | 1 | 2014-09-24 22:22:36 | Executing | 0 | 4080859
```


請注意，您之前的查詢結果與此步驟的結果之間有以下差異：

- 目前在 `WLM_QUERY_STATE_VW` 中有兩個資料列。其中一個結果是在此檢視上執行SELECT操作的自我參考查詢。第二個結果是上一步驟中長時間執行的查詢。
- `WLM_QUEUE_STATE_VW` 中的執行資料欄已從 1 增加為 2。此欄位值表示佇列中有兩個正在執行的查詢。
- 每次在佇列中執行查詢，`executed` 欄位都會遞增。

`WLM_QUEUE_STATE_VW` 檢視有助於取得佇列的整體檢視，以及每個佇列中處理的查詢數量。`WLM_QUERY_STATE_VW` 檢視有助於取得目前正在執行之個別查詢的詳細檢視。

第 2 節：修改WLM查詢佇列組態

現在您已了解佇列預設的運作方式，您可以了解如何使用手動設定查詢佇列WLM。在本節中，您將為叢集建立及設定新的參數群組。您會建立另外兩個使用者佇列，並根據查詢的使用者群組或查詢群組標籤，將佇列設為接受查詢。任何未路由到這兩個佇列之一的查詢，都會在執行時路由到預設佇列。

在參數群組中建立手動WLM組態

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台<https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單中，選擇組態，然後選擇工作負載管理以顯示工作負載管理頁面。
3. 選擇 Create (建立) 以顯示 Create parameter group (建立參數群組) 視窗。
4. 在參數群組名稱和描述中輸入 **WLMTutorial**，然後選擇建立以建立參數群組。

Note

Parameter group name (參數群組名稱) 會在建立時轉換成全部小寫的格式。

5. 在 Workload management (工作負載管理) 頁面上，選擇 **wlmtutorial** 參數群組來顯示詳細資訊頁面，其中包含 Parameters (參數) 和 Workload management (工作負載管理) 的標籤。
6. 確認您位於工作負載管理索引標籤上，然後選擇切換WLM模式以顯示並行設定視窗。
7. 選擇手動 WLM，然後選擇儲存以切換至手動 WLM。
8. 選擇 Edit workload queues (編輯工作負載佇列)。
9. 選擇 Add queue (新增佇列) 兩次來新增兩個佇列。現在有三個佇列：Queue 1 (佇列 1)、Queue 2 (佇列 2) 和 Default queue (預設佇列)。

10. 輸入每個佇列的資訊如下：

- 對於佇列 1，在記憶體 (%) 輸入 **30**、在主體的並行輸入 **2**，以及在查詢群組輸入 **test**。將其他設定保留為其預設值。
- 對於佇列 2，在記憶體 (%) 輸入 **40**、在主體的並行輸入 **3**，以及在使用者群組輸入 **admin**。將其他設定保留為其預設值。
- 不要對 Default queue (預設佇列) 做任何變更。WLM 會將未配置的記憶體指派給預設佇列。

11. 選擇 Save (儲存) 儲存設定。

接下來，將具有手動WLM組態的參數群組與叢集建立關聯。

將參數群組與手動WLM組態與叢集建立關聯

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽選單上，選擇叢集，然後選擇叢集來顯示您叢集的清單。
3. 選擇您的叢集，例如 `examplecluster`，以顯示叢集的詳細資訊。然後選擇屬性索引標籤以顯示該叢集的屬性。
4. 在資料庫組態區段中，選擇編輯、編輯參數群組，以顯示參數群組視窗。
5. 對於參數群組，請選擇先前建立的 `wlmtutorial` 參數群組。
6. 選擇儲存變更以關聯參數群組。

叢集會使用變更的參數群組進行修改。但是，您需要重新啟動叢集，才能將變更套用到資料庫。

7. 選擇您的叢集，然後在動作選擇重新啟動。

叢集重新啟動之後，狀態會回到 Available (可用)。

第 3 節：根據使用者群組和查詢群組將查詢路由至佇列

現在，您的叢集已與新的參數群組建立關聯，且已設定 WLM。接下來，執行一些查詢，以了解 Amazon Redshift 如何將查詢路由傳送到佇列來處理。

步驟 1：檢視資料庫中的查詢佇列組態

首先，確認資料庫具有您預期的 WLM 組態。

檢視查詢佇列組態

1. 開啟RSQL並執行下列查詢。查詢使用您在 中建立的 WLM_QUEUESTATE__VW 檢視 [步驟 1：建立 WLM_QUEUE_STATE_VW 檢視](#)。如果叢集在重新啟動之前有已連線到資料庫的工作階段，則需要重新連線。

```
select * from wlm_queue_state_vw;
```

以下是結果範例。

query	description	slots	mem	max_time	user_*
query_*	queued	executing	executed		
0	(super user) and (query group: superuser)	1	357	0	false
false	0	0	0		
1	(query group: test)	2	627	0	false
false	0	0	0		
2	(suser group: admin)	3	557	0	false
false	0	0	0		
3	(querytype:any)	5	250	0	false
false	0	1	0		

比較這些結果與您在 [步驟 1：建立 WLM_QUEUE_STATE_VW 檢視](#) 收到的結果。請注意，現在有兩個額外的佇列。佇列 1 現在是 test 查詢群組的佇列，佇列 2 是 admin 使用者群組的佇列。

佇列 3 現在是預設佇列。清單中的最後一個佇列一定是預設佇列。如果查詢未指定使用者群組或查詢群組，依預設會將查詢路由傳送到此佇列。

2. 執行下列查詢來確認您的查詢現在是在佇列 3 執行。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
2144	3	1	2014-09-24 23:49:59	Executing	0	550430

步驟 2：使用查詢群組佇列執行查詢

使用查詢群組佇列執行查詢

1. 執行以下查詢將其路由到 test 查詢群組。

```
set query_group to test;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. 從另一個RSQL視窗中，執行下列查詢。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

query	queue	slot_count	start_time	state	queue_time	exec_time
2168	1	1	2014-09-24 23:54:18	Executing	0	6343309
2170	3	1	2014-09-24 23:54:24	Executing	0	847

查詢已路由到測試查詢群組，即現在的佇列 1。

3. 選取佇列狀態檢視中的所有項目。

```
select * from wlm_queue_state_vw;
```

您會看到類似以下的結果。

query	description	slots	mem	max_time	user_*
query_*	queued	executing	executed		
0	(super user) and (query group: superuser)	1	357	0	false
false	0	0	0		
1	(query group: test)	2	627	0	false
false	0	1	0		
2	(suser group: admin)	3	557	0	false
false	0	0	0		
3	(querytype:any)	5	250	0	false
false	0	1	0		

4. 現在，重設查詢群組並再次執行長時間查詢：

```
reset query_group;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

5. 對檢視執行查詢以查看結果。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

```
query | description | slots | mem | max_time | user_* |
query_* | queued | executing | executed
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
      0 | (super user) and (query group: superuser) |      1 | 357 |          0 | false |
false |      0 |          0 |          0
      1 | (query group: test) |      2 | 627 |          0 | false |
false |      0 |          0 |          1
      2 | (suser group: admin) |      3 | 557 |          0 | false |
false |      0 |          0 |          0
      3 | (querytype:any) |      5 | 250 |          0 | false |
false |      0 |          2 |          5

query | queue | slot_count | start_time | state | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
2186 |      3 |          1 | 2014-09-24 23:57:52 | Executing | 0 | 649
2184 |      3 |          1 | 2014-09-24 23:57:48 | Executing | 0 | 4137349
```

結果應該是查詢現在再次在佇列 3 中執行。

步驟 3：建立資料庫使用者和群組

您必須先在資料庫中建立使用者群組並在群組中新增使用者，才可以在此佇列中執行查詢。然後使用 RSQL 新使用者的憑證使用登入，然後執行查詢。您需要以超級使用者的身分 (例如管理員使用者) 執行查詢，才能建立資料庫使用者。

建立新的資料庫使用者和使用者群組

1. 在 資料庫中，建立名為 的新資料庫使用者，adminwlm方法是在 RSQL 視窗中執行下列命令。

```
create user adminwlm createuser password '123Admin';
```

2. 然後執行下列命令來建立新使用者群組，並將您的新使用者 adminwlm 加入群組。

```
create group admin;
alter group admin add user adminwlm;
```

步驟 4：使用使用者群組佇列執行查詢

接下來您要執行查詢，並將其路由到使用者群組佇列。當您要將查詢路由到您設定用來處理理想執行之查詢類型的佇列，請這麼做。

使用使用者群組佇列執行查詢

1. 在RSQL視窗 2 中，執行下列查詢以切換至 adminwlm帳戶，並以該使用者身分執行查詢。

```
set session authorization 'adminwlm';
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. 在RSQL視窗 1 中，執行下列查詢，以查看要路由查詢的查詢佇列。

```
select * from wlm_query_state_vw;
select * from wlm_queue_state_vw;
```

以下為結果範例。

query	description	slots	mem	max_time	user_*
query_*	queued	executing	executed		
0	(super user) and (query group: superuser)	1	357	0	false
false	0	0	0		
1	(query group: test)	2	627	0	false
false	0	0	1		
2	(suser group: admin)	3	557	0	false
false	0	1	0		

```

    3 | (querytype:any) | 5 | 250 | 0 | false |
false | 0 | 1 | 8

query | queue | slot_count | start_time | state | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----
+-----+
2202 | 2 | 1 | 2014-09-25 00:01:38 | Executing | 0 | 4885796
2204 | 3 | 1 | 2014-09-25 00:01:43 | Executing | 0 | 650

```

執行此查詢的佇列是佇列 2，即 admin 使用者佇列。每次您以此使用者身分登入並執行查詢時，這些查詢就會在佇列 2 中執行，除非您指定要使用其他查詢群組。所選佇列取決於佇列指派規則。如需詳細資訊，請參閱[WLM 佇列指派規則](#)。

3. 現在請從RSQL視窗 2 執行下列查詢。

```

set query_group to test;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;

```

4. 在RSQL視窗 1 中，執行下列查詢，以查看要路由查詢的查詢佇列。

```

select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;

```

以下為結果範例。

```

query | description | slots | mem | max_time | user_* |
query_* | queued | executing | executed
-----+-----+-----+-----+-----+-----+-----
+-----+
    0 | (super user) and (query group: superuser) | 1 | 357 | 0 | false |
false | 0 | 0 | 0
    1 | (query group: test) | 2 | 627 | 0 | false |
false | 0 | 1 | 1
    2 | (suser group: admin) | 3 | 557 | 0 | false |
false | 0 | 0 | 1
    3 | (querytype:any) | 5 | 250 | 0 | false |
false | 0 | 1 | 10

query | queue | slot_count | start_time | state | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----
+-----+
2218 | 1 | 1 | 2014-09-25 00:04:30 | Executing | 0 | 4819666

```

```
2220 | 3 | 1 | 2014-09-25 00:04:35 | Executing | 0 | 685
```

5. 完成後，請重設查詢群組。

```
reset query_group;
```

第 4 節：使用 wlm_query_slot_count 暫時覆寫佇列中的並行層級

有時候使用者可能暫時需要較多資源來處理特定查詢。如果是這種情況，他們可以使用 wlm_query_slot_count 組態設定來暫時覆寫查詢佇列配置槽的方式。插槽是記憶體單位CPU，用於處理查詢。當您偶爾在叢集中取得大量資源的查詢時，例如在資料庫中執行VACUUM操作時，您可以覆寫插槽計數。

您可能發現使用者經常需要為某些類型的查詢設定 wlm_query_slot_count。如果是這樣，請考慮調整 WLM組態，並為使用者提供更符合其查詢需求的佇列。如需使用槽計數暫時覆寫並行層級的相關資訊，請參閱 [wlm_query_slot_count](#)。

步驟 1：使用 wlm_query_slot_count 覆寫並行層級

在本教學課程中，我們會執行相同的長時間執行SELECT查詢。我們會以 adminwlm 使用者的身分執行，使用 wlm_query_slot_count 增加查詢可使用的槽數量。

使用 wlm_query_slot_count 覆寫並行層級

1. 增加查詢的限制，以確保您有足夠的時間查詢 WLM_QUERYSTATE_VW 檢視並查看結果。

```
set wlm_query_slot_count to 3;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

2. 現在，向管理員使用者查詢 WLM_QUERY_STATE_VW，以查看查詢的執行方式。

```
select * from wlm_query_state_vw;
```

以下是結果範例。

```
query | queue | slot_count | start_time          | state      | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----
+-----+
2240 | 2 | 1 | 2014-09-25 00:08:45 | Executing | 0 | 3731414
2242 | 3 | 1 | 2014-09-25 00:08:49 | Executing | 0 | 596
```


請注意，查詢的槽計數是 3。此計數表示查詢會將三個槽都用來處理查詢，將佇列中的所有資源配置給該查詢。

3. 現在，執行下列查詢。

```
select * from WLM_QUEUE_STATE_VW;
```

以下是結果範例。

query	description	slots	mem	max_time	user_*
query_*	queued	executing	executed		
0	(super user) and (query group: superuser)	1	357	0	false
false	0	0	0		
1	(query group: test)	2	627	0	false
false	0	0	4		
2	(suser group: admin)	3	557	0	false
false	0	1	3		
3	(querytype:any)	5	250	0	false
false	0	1	25		

wlm_query_slot_count 組態的設定只在目前的工作階段中有效。如果該工作階段過期，或其他使用者執行查詢，則會使用WLM組態。

4. 重設槽計數，然後重新執行測試。

```
reset wlm_query_slot_count;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

以下為結果範例。

query	description	slots	mem	max_time	user_*
query_*	queued	executing	executed		
0	(super user) and (query group: superuser)	1	357	0	false
false	0	0	0		
1	(query group: test)	2	627	0	false
false	0	0	2		

```

  2 | (suser group: admin) | 3 | 557 | 0 | false |
false | 0 | 1 | 2
  3 | (querytype:any) | 5 | 250 | 0 | false |
false | 0 | 1 | 14

query | queue | slot_count | start_time | state | queue_time | exec_time
-----+-----+-----+-----+-----+-----+-----
+-----+
2260 | 2 | 1 | 2014-09-25 00:12:11 | Executing | 0 | 4042618
2262 | 3 | 1 | 2014-09-25 00:12:15 | Executing | 0 | 680

```

步驟 2：從不同工作階段執行查詢

接下來，請從不同工作階段執行查詢。

從不同工作階段執行查詢

1. 在RSQL視窗 1 和 2 中，執行下列動作以使用測試查詢群組。

```
set query_group to test;
```

2. 在RSQL視窗 1 中，執行下列長時間執行的查詢。

```
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

3. 由於長時間執行的查詢仍在RSQL視窗 1 中，請執行下列動作。這些命令會提高槽計數來使用佇列的所有槽，然後開始執行長時間執行的查詢。

```
set wlm_query_slot_count to 2;
select avg(l.priceperticket*s.qtysold) from listing l, sales s where l.listid <40000;
```

4. 開啟第三個RSQL視窗並查詢檢視以查看結果。

```
select * from wlm_queue_state_vw;
select * from wlm_query_state_vw;
```

以下為結果範例。

```
query | description | slots | mem | max_time | user_* |
query_* | queued | executing | executed
```

```

-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
  0 | (super user) and (query group: superuser) |      1 | 357 |      0 | false |
false |      0 |      0 |      0
  1 | (query group: test) |      2 | 627 |      0 | false |
false |      1 |      1 |      2
  2 | (suser group: admin) |      3 | 557 |      0 | false |
false |      0 |      0 |      3
  3 | (querytype:any) |      5 | 250 |      0 | false |
false |      0 |      1 |     18

query | queue | slot_count | start_time          | state          | queue_time |
exec_time
-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
2286 |      1 |          2 | 2014-09-25 00:16:48 | QueuedWaiting | 3758950    | 0
2282 |      1 |          1 | 2014-09-25 00:16:33 | Executing    | 0          |
19335850
2288 |      3 |          1 | 2014-09-25 00:16:52 | Executing    | 0          | 666

```

請注意，第一個查詢使用已配置給佇列 1 的其中一個槽來執行查詢。此外，請注意有一個查詢在佇列中等待 (其中，queued 是 1，state 是 QueuedWaiting)。第一個查詢完成後，第二個查詢就會開始執行。發生此執行是因為兩個查詢都路由到 test 查詢群組，第二個查詢必須等到有足夠的槽才能開始處理。

第 5 節：清理資源

叢集只要執行就會繼續產生費用。完成本教學課程後，請按照《Amazon Redshift 入門指南》中的[尋找其他資源和重設環境](#)中的步驟，將環境恢復到先前的狀態。

如需的詳細資訊 WLM，請參閱 [工作負載管理](#)。

並行擴展

使用並行擴展功能，您可以藉由持續的快速查詢效能，支援數千個並行使用者和並行查詢。開啟並行擴展時，Amazon Redshift 會自動新增額外的叢集容量以處理增加的讀取及寫入查詢。使用者會看到最新資料，無論查詢是執行於主要叢集或並行擴展叢集。

您可以透過設定 WLM 佇列來管理要傳送至並行擴展叢集的查詢。當您開啟並行擴展時，合格查詢會傳送到並行擴展叢集，而非在佇列中等待。

我們只會向您收取並行擴展叢集實際執行查詢的費用。如需定價的相關資訊，包括費用應計方式與最低費用，請參閱[並行擴展定價](#)。

主題

- [並行擴展功能](#)
- [並行擴展的限制](#)
- [AWS 區域 用於並行擴展](#)
- [並行擴展候選項目](#)
- [設定並行擴展佇列](#)
- [監控並行擴展](#)
- [並行擴展系統檢視](#)

並行擴展功能

當您開啟WLM佇列的並行擴展時，它適用於讀取操作，例如儀表板查詢。也適用於常用的寫入操作，例如用於資料擷取和處理的陳述式。

寫入操作的並行擴展功能

並行擴展支援常用的寫入操作，例如擷取、轉換和載入（ETL）陳述式。當您想要在叢集收到大量要求時維持一致的回應時間時，寫入操作的並行擴展特別有用。這會改善在主叢集上爭用資源的寫入操作輸送量。

並行擴展支援 COPY、INSERT、UPDATE、DELETE和 CREATE TABLE AS（CTAS）陳述式。此外，並行擴展支援不使用彙總MV的具體化檢視重新整理。不支援其他資料處理語言（DML）陳述式和資料定義語言（DDL）陳述式。當不受支援的寫入陳述式，例如CREATE沒有TABLE AS，包含在受支援寫入陳述式之前的明確交易中時，任何寫入陳述式都不會在並行擴展叢集上執行。

當您為並行擴展累積點數時，此點數累積適用於讀取與寫入操作。

並行擴展的限制

以下是使用 Amazon Redshift 並行擴展的限制：

- 不支援對使用交錯排序索引鍵的資料表進行查詢。
- 不支援對暫存資料表進行查詢。

- 它不支援存取受限制性網路或虛擬私有雲端 (VPC) 組態保護之外部資源的查詢。
- 它不支援包含 Python 使用者定義函數 (UDFs) 和 Lambda 的查詢UDFs。
- 它不支援存取系統資料表、PostgreSQL 目錄資料表或無備份資料表的查詢。
- 當有限制性IAM政策許可時，它不支援COPY或UNLOAD查詢存取外部資源。這包括套用至資源的許可，例如 Amazon S3 儲存貯體或 DynamoDB 資料表或來源。IAM 來源可以包括下列項目：
 - `aws:sourceVpc` – 來源 VPC。
 - `aws:sourceVpce` – 來源VPC端點。
 - `aws:sourceIp` – 來源 IP 地址。

在某些情況下，您可能需要移除限制資源或來源的許可，以便將存取資源的 COPY和 UNLOAD查詢傳送至並行擴展叢集。

如需資源政策的詳細資訊，請參閱 AWS Identity and Access Management 使用者指南中的[政策類型](#)，以及[使用儲存貯體政策 控制VPC端點的存取](#)。

- Amazon Redshift 寫入操作的並行擴展不支援 DDL操作，例如 CREATETABLE或 ALTER TABLE。
- 它不支援 ANALYZE COPY命令。
- 它不支援目標資料表上的寫入操作，其中 DISTSTYLE 設定為 ALL。
- 它不支援COPY下列檔案格式：
 - Parquet
 - ORC
- 不支援對具有身分資料欄的資料表進行寫入操作。
- Amazon Redshift 僅支援對 Amazon Redshift RA3節點進行寫入操作並行擴展。其他節點類型不支援寫入操作的並行擴展。

AWS 區域 用於並行擴展

透過 Amazon Redshift，您可以使用並行擴展來管理跨 Redshift 叢集的並行工作負載需求。本主題詳細介紹哪些區域可與 Amazon Redshift 搭配使用並行擴展。

並行擴展適用於以下 AWS 區域：

- 美國東部 (維吉尼亞北部) 區域 (us-east-1)
- 美國東部 (俄亥俄) 區域 (us-east-2)
- 美國西部 (加利佛尼亞北部) 區域 (us-west-1)

- 美國西部 (奧勒岡) 區域 (us-west-2)
- 亞太區域 (孟買) 區域 (ap-south-1)
- 亞太區域 (首爾) 區域 (ap-northeast-2)
- 亞太區域 (新加坡) 區域 (ap-southeast-1)
- 亞太區域 (雪梨) 區域 (ap-southeast-2)
- 亞太區域 (東京) 區域 (ap-northeast-1)
- 加拿大 (中部) 區域 (ca-central-1)
- 中國 (北京) 區域 (cn-north-1)
- 中國 (寧夏) 區域 (cn-northwest-1)
- 歐洲 (法蘭克福) 區域 (eu-central-1)
- 歐洲 (愛爾蘭) 區域 (eu-west-1)
- 歐洲 (倫敦) 區域 (eu-west-2)
- 歐洲 (巴黎) 區域 (eu-west-3)
- 歐洲 (斯德哥爾摩) 區域 (eu-north-1)
- 歐洲 (蘇黎世) 區域 (eu-central-2)
- 歐洲 (西班牙) 區域 (eu-south-2)
- 南美洲 (聖保羅) 區域 (sa-east-1)
- AWS GovCloud (美國東部)

並行擴展候選項目

使用 Amazon Redshift，您可以橫向擴展查詢處理，以加速並行查詢的執行。下列主題說明 Amazon Redshift 用來判斷要路由至並行擴展的查詢的條件。

只有在主要叢集符合以下要求時，查詢才會路由至並行擴展叢集：

- EC2-VPC 平台。
- 節點類型必須為 dc2.8xlarge、dc2.large、ra3.large、ra3.xlplus、ra3.4xlarge 或 ra3.16xlarge。只有 Amazon Redshift RA3 節點支援寫入操作的並行擴展。
- 對於具有 ra3.xlplus、ra3.4xlarge 或 ra3.16xlarge 節點類型的叢集，最多 32 個運算節點。此外，最初建立叢集時，主要叢集的節點數不得超過 32 個節點。例如，如果叢集最初建立時為 40 個節點，即使叢集目前具備 20 個節點，也不符合並行擴展的需求。相反地，如果 DC2 叢集目前有 40 個節點，但最初建立的節點為 20 個，則確實符合並行擴展的需求。

- 非單一節點叢集。

設定並行擴展佇列

使用 Amazon Redshift，您可以透過設定並行擴展來管理並行和系統資源。並行擴展佇列可讓您設定可同時執行的查詢或使用使用者工作階段數量限制。下一節說明如何在 Amazon Redshift 中啟用並行擴展佇列，讓您有效地處理並行查詢和使用使用者工作階段。

您可以在工作負載管理器（WLM）佇列中啟用並行擴展，將查詢路由至並行擴展叢集。若要開啟佇列的並行擴展，請將 Concurrency Scaling mode (並行擴展模式) 值設定為 auto (自動)。

當路由至啟用並行擴展之佇列的查詢數量超過佇列的並行容量時，無論該容量是手動設定還是自動決定，合格的查詢都會傳送至並行擴展叢集。當主叢集上可用的佇列插槽時，查詢會路由至主叢集並在主叢集上執行。與任何 WLM 佇列一樣，您可以根據使用者群組，或根據將查詢指派給佇列中 [定義的相符條件，將查詢路由至並行擴展佇列](#)。您也可以藉由定義 [WLM 查詢監控規則](#) 來路由查詢。例如，您可以將耗時超過 5 秒的所有查詢路由至並行擴展佇列。請記住，佇列行為可能會有所不同，取決於您使用的是自動 WLM 還是手動 WLM。如需詳細資訊，請參閱 [實作自動 WLM](#) 或 [實作手動 WLM](#)。

並行擴展叢集的預設數目為一。可使用的並行擴展叢集數目由 [max_concurrency_scaling_clusters](#) 控制。

監控並行擴展

透過 Amazon Redshift，您可以監控和管理並行擴展，以最佳化資料倉儲工作負載的效能和成本效益。並行擴展可讓 Amazon Redshift 在工作負載需求增加時自動新增額外的叢集容量，並在需求減少時移除該容量。下一節提供監控 Amazon Redshift 叢集並行擴展的指引。

若要查看查詢是執行於主要叢集或並行擴展叢集，您可以瀏覽至 Amazon Redshift 主控台叢集中的叢集，然後選擇一個叢集。然後選擇查詢監控索引標籤和工作負載並行，檢視有關執行中查詢和佇列查詢的資訊。

若要尋找執行時間，請查詢 `concurrency_scaling_status` 資料欄上的 `STL_QUERY` 資料表和篩選條件。以下查詢可針對執行於並行擴展叢集與執行於主要叢集的查詢，比較其佇列時間與執行時間。

```
SELECT w.service_class AS queue
, CASE WHEN q.concurrency_scaling_status = 1 THEN 'concurrency scaling cluster' ELSE
  'main cluster' END as concurrency_scaling_status
, COUNT( * ) AS queries
, SUM( q.aborted ) AS aborted
```

```
, SUM( ROUND( total_queue_time::NUMERIC / 1000000,2) ) AS queue_secs
, SUM( ROUND( total_exec_time::NUMERIC / 1000000,2) ) AS exec_secs
FROM stl_query q
JOIN stl_wlm_query w
USING (userid,query)
WHERE q.userid > 1
AND q.starttime > '2019-01-04 16:38:00'
AND q.endtime < '2019-01-04 17:40:00'
GROUP BY 1,2
ORDER BY 1,2;
```

根據您的需求調整 starttime 和 endtime 值。

並行擴展系統檢視

透過 Amazon Redshift，您可以使用並行擴展系統檢視來監控和管理叢集中的並行擴展活動。下一節說明查詢這些系統檢視，並解譯結果，以有效利用 Amazon Redshift 環境中的並行擴展。

一組具有字首的系統檢視SVCS提供系統日誌資料表中有關主要和並行擴展叢集查詢的詳細資訊。

下列檢視具有與對應STL檢視相似的資訊SVL：

- [SVCS_ALERT_EVENT_LOG](#)
- [SVCS_COMPILE](#)
- [SVCS_EXPLAIN](#)
- [SVCS_PLAN_INFO](#)
- [SVCS_QUERY_SUMMARY](#)
- [SVCS_STREAM_SEGS](#)

以下檢視專屬於並行擴展。

- [SVCS_CONCURRENCY_SCALING_USAGE](#)

如需並行擴展的相關資訊，請參閱《Amazon Redshift 管理指南》中的下列主題。

- [檢視並行擴展資料](#)
- [檢視查詢執行期間的叢集效能](#)
- [檢視查詢詳細資訊](#)

短期查詢加速

短查詢加速 (SQA) 會優先處理選取的短執行查詢，然後再處理較長的查詢。SQA 在專用空間中執行短執行查詢，這樣SQA查詢就不會被迫在較長查詢後的佇列中等待。SQA 只會優先處理短執行中且位於使用者定義佇列中的查詢。使用 SQA時，短執行查詢會開始更快速地執行，使用者會更快看到結果。

如果您啟用 SQA，您可以減少專用於執行簡短查詢的工作負載管理 (WLM) 佇列。此外，長時間執行的查詢不需要處理佇列中插槽的簡短查詢，因此您可以將WLM佇列設定為使用較少的查詢插槽。使用較低的並行性時，就大部分工作負載而言，查詢傳輸量會增加，且整體系統效能會改善。

[CREATE TABLE AS](#) (CTAS) 陳述式和唯讀查詢，例如[SELECT](#)陳述式，符合的資格SQA。

Amazon Redshift 採用機器學習演算法來分析每一個合格查詢，並預測查詢的執行時間。根據預設，會根據叢集工作負載的分析，WLM動態指派SQA最長執行時間的值。或者，您可以指定固定值 1–20 秒。如果查詢的預測執行時間少於定義或動態指派SQA的最長執行時間，且查詢正在WLM佇列中等待，會將查詢與WLM佇列SQA分開，並排程優先順序執行。如果查詢的執行時間超過SQA最長執行時間，會根據WLM佇列指派規則將查詢WLM移至第一個相符的佇列。[WLM](#) 隨著時間的推移，預測會隨著從查詢模式中SQA學習而改善。

SQA 預設會在預設參數群組中啟用，並針對所有新的參數群組啟用。若要在 Amazon Redshift 主控台SQA中停用，請編輯參數群組的WLM組態，然後取消選取啟用短查詢加速。最佳實務是，建議使用 15 個或更少的WLM查詢槽計數，以維持最佳整體系統效能。如需修改WLM組態的相關資訊，請參閱 Amazon Redshift [管理指南](#) 中的[設定工作負載](#)管理。

短期查詢最長執行時間

當您啟用時SQA，會將短查詢的最長執行時間WLM預設為動態。建議您保留動態設定，以SQA達最長執行時間。您可以指定固定值 1–20 秒以覆寫預設設定。

在某些情況下，您可以考慮使用不同的值來達到SQA最大執行時間值，以改善系統效能。在此情況下，請分析工作負載，以找出大部分短期執行查詢的最長執行時間。下列查詢傳回第 70 個百分位數附近之查詢的最長執行時間。

```
select least(greatest(percentile_cont(0.7)
within group (order by total_exec_time / 1000000) + 2, 2), 20)
from stl_wlm_query
where userid >= 100
and final_state = 'Completed';
```

決定適合工作負載的最長執行時間值之後，就不需要變更，除非工作負載大幅改變。

監控 SQA

若要檢查SQA是否已啟用，請執行下列查詢。如果查詢傳回資料列，SQA則會啟用。

```
select * from stv_wlm_service_class_config
where service_class = 14;
```

下列查詢顯示通過每個查詢佇列 (服務類別) 的查詢數。也顯示平均執行時間、第 90 個百分位數的查詢數和等待時間，以及平均等待時間。SQA 查詢用於服務類別 14。

```
select final_state, service_class, count(*), avg(total_exec_time),
percentile_cont(0.9) within group (order by total_queue_time), avg(total_queue_time)
from stl_wlm_query where userid >= 100 group by 1,2 order by 2,1;
```

若要尋找哪些查詢已由擷取SQA並順利完成，請執行下列查詢。

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class = 14 and a.final_state = 'Completed'
order by b.query desc limit 5;
```

若要尋找SQA已提取但逾時的查詢，請執行下列查詢。

```
select a.queue_start_time, a.total_exec_time, label, trim(querytxt)
from stl_wlm_query a, stl_query b
where a.query = b.query and a.service_class = 14 and a.final_state = 'Evicted'
order by b.query desc limit 5;
```

如需移出查詢的相關資訊，以及可對查詢執行之更普遍的規則式動作的詳細資訊，請參閱 [WLM 查詢監控規則](#)。

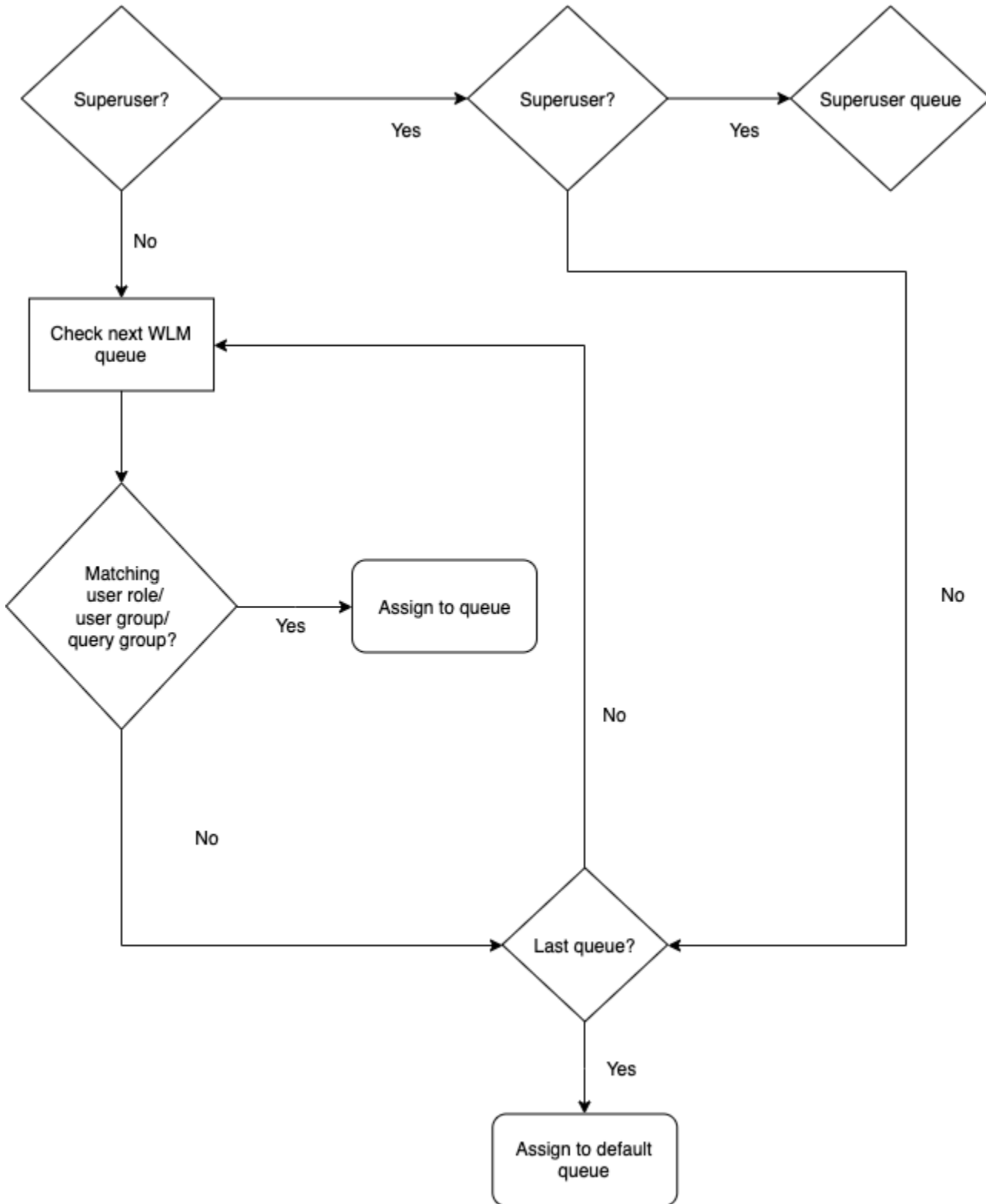
WLM 佇列指派規則

使用 Amazon Redshift，您可以透過在工作負載管理 (WLM) 組態中定義佇列指派規則，來控制使用者查詢的記憶體CPU和資源配置。下一節說明建立和管理WLM佇列指派規則，以實現有效的資源配置，並符合 Amazon Redshift 中各種工作負載的服務層級協議。

當使用者執行查詢時，會根據佇列WLM指派規則，將查詢指派給第一個相符的WLM佇列：

1. 如果使用者以超級使用者身分登入，且在標示為超級使用者的查詢群組中執行查詢，此查詢會指派給超級使用者佇列。
2. 如果使用者是角色的一部份、屬於列出的使用者群組，或在列出的查詢群組內執行查詢，此查詢會指派給第一個相符的佇列。
3. 如果查詢不符合任何條件，則會將查詢指派給預設佇列，這是WLM組態中定義的最後一個佇列。

下圖說明這些規則如何運作。

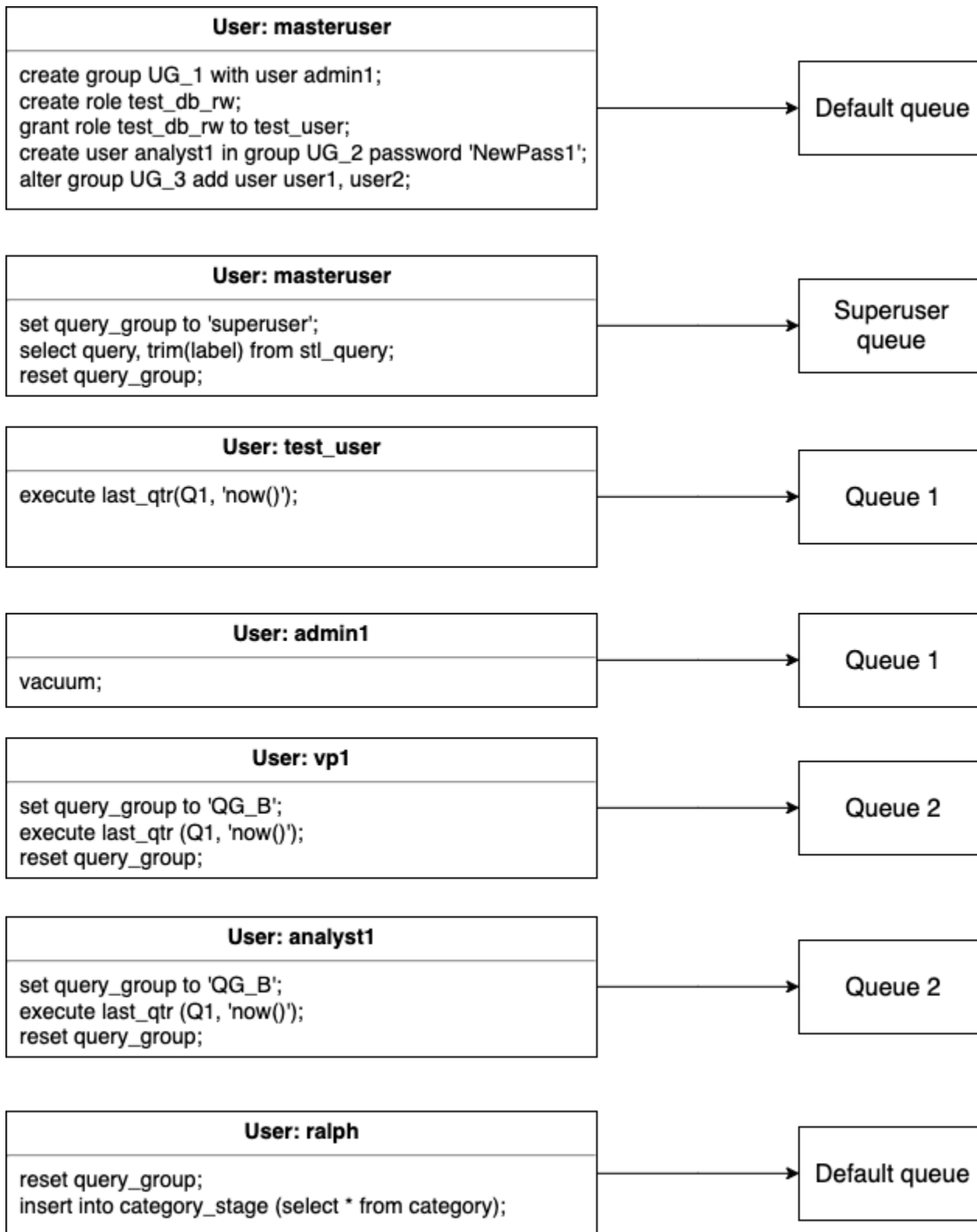


佇列指派範例

下表顯示具有超級使用者佇列和四個使用者定義佇列的WLM組態。

佇列	並行數量	使用者角色	User Groups (使用者群組)	Query Groups (查詢群組)
超級使用者	1			superuser
1	5	test_db_rw	UG_1	
2	5			QG_B
3	5		UG_2	QG_C
預設	5			

下圖示範如何根據使用者群組和查詢群組，將查詢指派給上表中的佇列。如需在執行時間如何將查詢指派給使用者群組和查詢群組的資訊，請參閱本節稍後的[將查詢指派給佇列](#)。



在此範例中，WLM會進行下列指派：

1. 第一組陳述式示範將使用者指派給使用者群組的三種方法。陳述式是由使用者執行 `adminuser`，該使用者不是任何WLM佇列中列出的使用者群組的成員。未設定任何查詢群組，所以會將陳述式路由至預設佇列。
2. 使用者 `adminuser` 是超級使用者，且查詢群組設為 `'superuser'`，所以會將查詢指派給超級使用者佇列。
3. 使用者 `test_user` 會獲派佇列 1 所列出的角色 `test_db_rw`，所以查詢會指派給佇列 1。
4. 使用者 `admin1` 是佇列 1 所列出之使用者群組的成員，所以查詢會指派給佇列 1。
5. 使用者 `vp1` 不是任何列出之使用者群組的成員。查詢群組設為 `'QG_B'`，所以會將查詢指派給佇列 2。
6. 使用者 `analyst1` 是佇列 3 中所列出之使用者群組的成員，但 `'QG_B'` 符合佇列 2，所以會將此查詢指派給佇列 2。
7. 使用者 `ralph` 不是任何列出之使用者群組的成員，且該查詢群組已重設，所以沒有相符的佇列。會將該查詢指派給預設佇列。

將查詢指派給佇列

使用 Amazon Redshift，您可以管理工作負載並行，並透過將查詢指派給佇列來排定查詢的優先順序。佇列可讓您 CPU 為不同類型的查詢或使用者配置資源，例如記憶體，並確保關鍵查詢優先於較不重要的查詢。下列各節說明如何建立佇列、設定其屬性，以及根據您定義的條件指派傳入的查詢。

下列範例會根據使用者角色、使用者群組和查詢群組，將查詢指派給佇列。

根據使用者角色將查詢指派給佇列

如果將使用者指派給某個角色，且該角色已附加至佇列，則該使用者執行的查詢會指派至該佇列。下列範例會建立名為 `sales_rw` 的使用者角色，並將使用者 `test_user` 指派給該角色。

```
create role sales_rw;  
grant role sales_rw to test_user;
```

您也可以將一個角色明確授予另一個角色，來合併兩個角色的許可。將巢狀角色指派給使用者，會將這兩個角色的許可授予使用者。

```
create role sales_rw;  
create role sales_ro;  
grant role sales_ro to role sales_rw;
```

```
grant role sales_rw to test_user;
```

若要查看叢集中已授予角色的使用者清單，請查詢 `SVV_USER_GRANTS` 資料表。若要查看叢集中已授予角色的角色清單，請查詢 `SVV_ROLE_GRANTS` 資料表。

```
select * from svv_user_grants;
select * from svv_role_grants;
```

根據使用者群組將查詢指派給佇列

如果使用者群組名稱列在佇列定義中，則由該使用者群組的成員所執行的查詢會指派給相應佇列。下列範例會建立使用者群組，並使用SQL命令 [CREATE USER](#)、[CREATE GROUP](#)和將使用者新增至群組[ALTER GROUP](#)。

```
create group admin_group with user admin246, admin135, sec555;
create user vp1234 in group ad_hoc_group password 'vpPass1234';
alter group admin_group add user analyst44, analyst45, analyst46;
```

將查詢指派給查詢群組

您可以在執行時間將查詢指派給適當的查詢群組，以便將查詢指派給佇列。使用 `SET`命令開始查詢群組。

```
SET query_group TO group_label
```

在此，*group_label* 是在WLM組態中列出的查詢群組標籤。

您在 `SET query_group` 命令之後執行的所有查詢，將會以指定之查詢群組的成員身分執行，直到您重設查詢群組或結束目前的登入工作階段為止。如需有關設定和重設 Amazon Redshift 物件的資訊，請參閱 SQL命令參考[RESET](#)中的 [SET](#)和。

您指定的查詢群組標籤必須包含在目前的WLM組態中；否則，`SET query_group` 命令不會影響查詢佇列。

`TO` 子句中定義的標籤會擷取到查詢日誌中，您可以利用此標籤來進行故障診斷。如需查詢群組組態參數的相關資訊，請參閱組態參考中的 [query_group](#)。

下列範例在查詢群組 'priority' 中執行兩個查詢，然後重設查詢群組。


```
set query_group to 'priority';
select count(*)from stv_blocklist;
select query, elapsed, substring from svl_qlog order by query desc limit 5;
reset query_group;
```

將查詢指派給超級使用者佇列

若要將查詢指派給超級使用者佇列，請以超級使用者身分登入 Amazon Redshift，然後在超級使用者群組中執行查詢。完成時，請重設查詢群組，後續的查詢就不會在超級使用者佇列中執行。

以下範例指派兩個命令在超級使用者佇列中執行。

```
set query_group to 'superuser';

analyze;
vacuum;
reset query_group;
```

若要檢視超級使用者清單，請查詢 PG_USER 系統目錄表。

```
select * from pg_user where usesuper = 'true';
```

WLM 動態和靜態組態屬性

WLM 組態屬性是動態或靜態的。您可在不重新啟動叢集的情形下，將動態屬性套用至資料庫，但靜態屬性需要重新啟動叢集才能讓變生效。但假如您同時變更了動態與靜態屬性，則必須重新啟動叢集，所有屬性變更才會生效。無論變更的屬性是動態或是靜態，都是如此。

套用動態屬性時，叢集狀態為 modifying。在自動 WLM 和手動之間切換 WLM 是靜態變更，需要叢集重新啟動才能生效。

下表指出使用自動或手動時，哪些 WLM 屬性是動態 WLM 或靜態的 WLM。

WLM 屬性	自動 WLM	手動 WLM
查詢群組	動態	靜態
查詢群組萬用字元	動態	靜態

WLM 屬性	自動 WLM	手動 WLM
使用者群組	動態	靜態
使用者群組萬用字元	動態	靜態
使用者角色	動態	靜態
使用者角色萬用字元	動態	靜態
主要叢集的並行	不適用	動態
並行擴展模式	動態	動態
啟用短期查詢加速	不適用	動態
短期查詢最長執行時間	動態	動態
要使用的記憶體百分比	不適用	動態
逾時	不適用	動態
優先順序	動態	不適用
新增或移除佇列	動態	靜態

如果您修改查詢監控規則（QMR），則變更會自動發生，而不需要修改叢集。

Note

使用手動時WLM，如果變更逾時值，則新值會套用至在變更值後開始執行的任何查詢。如果變更要使用的記憶體並行或百分比，Amazon Redshift 將動態變更至新組態。如此，目前執行中的查詢將不會受變更影響。如需詳細資訊，請參閱[WLM動態記憶體配置](#)。

主題

- [WLM 動態記憶體配置](#)
- [動態WLM範例](#)

WLM 動態記憶體配置

在每個佇列中，WLM會建立與佇列並行層級相等的多個查詢插槽。配置給查詢槽的記憶體數量等於配置給佇列的記憶體百分比除以槽計數。如果您變更記憶體配置或並行，Amazon Redshift 會動態管理轉換至新WLM組態的作業。因此，作用中查詢可以使用目前配置的記憶體數量，執行到完成為止。同時，Amazon Redshift 會確保記憶體總用量絕不超過 100% 可用的記憶體。

工作負載管理員使用下列程序來管理轉移：

1. WLM 重新計算每個新查詢插槽的記憶體配置。
2. 如果執行中的查詢未主動使用查詢插槽，會WLM移除該插槽，讓該記憶體可用於新的插槽。
3. 如果查詢插槽正在使用中，會WLM等待查詢完成。
4. 當作用中查詢完成時，就會移除空的槽，並釋放相關的記憶體。
5. 只要有足夠的記憶體可供新增一或多個槽，就會新增新的槽。
6. 當變更完成時執行的所有查詢，插槽計數等於新的並行層級，且轉換至新WLM組態完成。

事實上，變生效時正在執行的查詢會繼續使用原始的記憶體配置。變生效時已排入佇列的查詢會在有新的槽可用時，路由傳送至新的槽。

如果在轉換過程中變更WLM動態屬性，WLM 會從目前狀態立即開始轉換為新組態。若要檢視轉移的狀態，請查詢 [STV_WLM_SERVICE_CLASS_CONFIG](#) 系統資料表。

動態WLM範例

使用 Amazon Redshift，您可以使用動態 WLM（工作負載管理）自動管理 Amazon Redshift 叢集中的工作負載分佈和資源分配。動態WLM是工作負載管理（WLM）組態的範例，可根據工作負載需求動態調整記憶體配置，從而實現最佳的並行和效能。下一節提供實作和設定 Amazon Redshift WLM 叢集動態的詳細資訊。

假設您的叢集WLM使用下列動態屬性設定兩個佇列。

佇列	並行數量	要使用的記憶體 %
1	4	50%
2	4	50%

現在，假設您的叢集有 200 GB 的記憶體可供查詢處理使用。(此為任意數字，僅供示範用途。) 如下列方程式所示，每個槽配置 25 GB。

$$(200 \text{ GB} * 50\%) / 4 \text{ slots} = 25 \text{ GB}$$

接下來，您將變更為WLM使用下列動態屬性。

佇列	並行數量	要使用的記憶體 %
1	3	75%
2	4	25%

如下列方程式所示，佇列 1 中每個槽的新記憶體配置為 50 GB。

$$(200 \text{ GB} * 75\%) / 3 \text{ slots} = 50 \text{ GB}$$

假設套用新組態時，查詢 A1、A2、A3 和 A4 在執行中，而查詢 B1、B2、B3 和 B4 會排入佇列。WLM 動態重新設定查詢插槽，如下所示。

步驟	執行中的查詢	目前槽計數	目標槽計數	配置的記憶體	可用的記憶體量
1	A1、A2、A3、A4	4	0	100 GB	50 GB
2	A2、A3、A4	3	0	75 GB	75 GB
3	A3、A4	2	0	50 GB	100 GB
4	A3、A4、B1	2	1	100 GB	50 GB
5	A4、B1	1	1	75 GB	75 GB
6	A4、B1、B2	1	2	125 GB	25 GB
7	B1、B2	0	2	100 GB	50 GB
8	B1、B2、B3	0	3	150 GB	0 GB

1. WLM 重新計算每個查詢插槽的記憶體配置。佇列 1 最初配置 100 GB。新的佇列總共配置 150 GB，所以新佇列立即有 50 GB 可用。佇列 1 現在使用四個槽，且新的並行層級是三個槽，所以不會新增任何槽。
2. 當一個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有三個槽和 75 GB 的可用記憶體。新的組態規定每個新的槽要有 50 GB，但新的並行層級是三個槽，所以不會新增任何槽。
3. 當第二個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有兩個槽和 100 GB 的可用記憶體。
4. 使用 50 GB 的可用記憶體新增一個槽。佇列 1 現在有三個槽和 50 GB 的可用記憶體。排入佇列的查詢現在可以路由至新的槽。
5. 當第三個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有兩個槽和 75 GB 的可用記憶體。
6. 使用 50 GB 的可用記憶體新增一個槽。佇列 1 現在有三個槽和 25 GB 的可用記憶體。排入佇列的查詢現在可以路由至新的槽。
7. 當第四個查詢完成，就會移除槽並釋出 25 GB。佇列 1 現在有兩個槽和 50 GB 的可用記憶體。
8. 使用 50 GB 的可用記憶體新增一個槽。佇列 1 現在有三個槽，各有 50 GB，所有可用的記憶體都已配置。

轉移完成，所有查詢槽都可供排入佇列的查詢使用。

WLM 查詢監控規則

在 Amazon Redshift 工作負載管理 (WLM) 中，查詢監控規則會定義 WLM 佇列的指標型效能界限，並指定查詢超出這些界限時要採取的動作。例如，對於短期執行查詢專用的佇列，您可以建立規則，以將執行時間超過 60 秒的查詢取消。若要追蹤設計不良的查詢，您可以使用另一個規則來記錄含有巢狀迴圈的查詢。

您可以在工作負載管理 (WLM) 組態中定義查詢監控規則。每一個佇列最多可以定義 25 個規則，而全部佇列合計以 25 個規則為限制。每個規則最多包含三個條件 (或述詞) 和一個動作。述詞由指標、比較條件 (=、< 或 >) 和值所組成。如果符合任何規則的所有述詞，就會觸發該規則的動作。可能的規則動作包括記錄、跳轉和中止，如下所述。

給定佇列中的規則僅套用至該佇列中執行的查詢。各規則彼此無關。

WLM 每 10 秒評估一次指標。如果在同一期間觸發多個規則，會 WLM 選擇具有最嚴重動作的規則。如果兩個規則的動作具有相同的嚴重性，會根據規則名稱依字母順序 WLM 執行規則。如果動作是跳轉或中止，則會記錄動作，並從佇列中移出查詢。如果動作是記錄，則查詢會在佇列中繼續執行。WLM 每個規則每個查詢僅啟動一個日誌動作。如果佇列包含其他規則，那些規則仍然有效。如果動作是跳轉，

且將查詢路由至另一個佇列，則會套用新佇列的規則。如需針對特定查詢執行的查詢監控和追蹤動作的相關資訊，請參閱 [短期查詢加速](#) 中的範例集合。

符合所有規則的述詞時，會將資料列WLM寫入 [STL_WLM_RULE_ACTION](#) 系統資料表。此外，Amazon Redshift 會將目前執行中查詢的查詢指標記錄到 [STV_QUERY_METRICS](#)。已完成之查詢的指標儲存在 [STL_QUERY_METRICS](#) 中。

定義查詢監控規則

您可以在WLM組態中建立查詢監控規則，而這些規則會定義成叢集參數群組定義的一部分。

您可以使用 [建立規則](#)，AWS Management Console 或使用 [以程式設計方式建立規則JSON](#)。

Note

如果您選擇以程式設計方式建立規則，強烈建議使用主控台來產生JSON您在參數群組定義中包含的。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [使用主控台建立或修改查詢監控規則](#) 和 [使用 AWS CLI設定參數值](#)。

若要定義查詢監控規則，請指定下列元素：

- 規則名稱 – 規則名稱在WLM組態中必須是唯一的。規則名稱最多為 32 個英數字元或底線，且不可含有空格和問號。每個佇列最多可以有 25 個規則，而全部佇列合計最多 25 規則。
- 一或多個述詞 – 每個規則最多可以有三個述詞。如果符合任何規則的所有述詞，就會觸發相關聯的動作。述詞由指標名稱、運算子 (=、< 或 >) 和值所定義。例如，`query_cpu_time > 100000`。如需指標清單及各種指標的範例值，請參閱本節後續的 [已佈建 Amazon Redshift 的查詢監控指標](#)。
- 動作 – 如果觸發多個規則，會WLM選擇具有最嚴重動作的規則。依嚴重性遞增順序，可能的動作如下：
 - 日誌 – 在 `STL_WLM_RULE_ACTION` 系統資料表中記錄查詢的相關資訊。只需要寫入日誌記錄時，請使用「記錄」動作。WLM 每個規則最多會為每個查詢建立一個日誌。在日誌動作之後，其他規則仍然有效，WLM並繼續監控查詢。
 - Hop (僅適用於手動 WLM) – 記錄動作並將查詢跳至下一個相符佇列。如果沒有另一個相符的佇列，則會取消查詢。QMR 只跳 [CREATE TABLE AS \(CTAS\)](#) 陳述式和唯讀查詢，例如 `SELECT` 陳述式。如需詳細資訊，請參閱 [WLM 查詢佇列跳轉](#)。
 - 中止 – 記錄動作並終止查詢。QMR 不會停止 `COPY` 陳述式和維護操作，例如 `ANALYZE` 和 `VACUUM`。
 - 變更優先順序 (僅適用於自動 WLM) – 變更查詢的優先順序。

若要限制查詢的執行時間，建議您建立查詢監控規則，而不是使用WLM逾時。例如，您可以以`max_execution_time`設定為 50,000 毫秒，如下列JSON程式碼片段所示。

```
"max_execution_time": 50000
```

但是，我們建議您改為定義等效的查詢監控規則。下列範例示範將 `query_execution_time` 設為 50 秒的查詢監控規則：

```
"rules":
[
  {
    "rule_name": "rule_query_execution",
    "predicate": [
      {
        "metric_name": "query_execution_time",
        "operator": ">",
        "value": 50
      }
    ],
    "action": "abort"
  }
]
```

如需建立或修改查詢監控規則的步驟，請參閱《Amazon Redshift 管理指南》中的[使用主控台建立或修改查詢監控規則](#)和 [wlm_json_configuration](#) 參數中的屬性。


在下列主題中，您可以找到查詢監控規則的詳細資訊：

- [已佈建 Amazon Redshift 的查詢監控指標](#)
- [查詢監控規則範本](#)
- [使用主控台建立規則](#)
- [設定工作負載管理](#)
- [查詢監控規則的系統資料表和檢視](#)

已佈建 Amazon Redshift 的查詢監控指標

下表描述查詢監控規則中使用的指標。(這些指標不同於 [STV_QUERY_METRICS](#) 和 [STL_QUERY_METRICS](#) 系統資料表中儲存的指標。)

對於給定的指標，將會在查詢層級或區段層級追蹤效能臨界值。如需區段和步驟的相關資訊，請參閱[查詢計劃和執行工作流程](#)。

 Note

[WLM 逾時](#) 參數不同於查詢監控規則。

指標	名稱	描述
查詢CPU時間	query_cpu_time	CPU 查詢使用的時間，以秒為單位。CPU time 與不同Query execution time。 有效值為 0-999,999。
已讀取的區塊	query_blocks_read	查詢所讀取的 1 MB 資料區塊數。 有效值為 0-1,048,575。
掃描的資料列數	scan_row_count	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。 有效值為 0-999,999,999,999,999。
查詢執行時間	query_execution_time	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。 有效值為 0-86,399。
查詢佇列時間	query_queue_time	在佇列中等待的時間 (秒)。 有效值為 0 至 86,399。
CPU 用量	query_cpu_usage_percent	查詢使用的CPU容量百分比。 有效值為 0-6,399。

指標	名稱	描述
記憶體到磁碟	query_temp_blocks_to_disk	用來寫入中間結果的暫時磁碟空間，以 1 MB 區塊為單位。 有效值為 0–319,815,679。
CPU 偏移	cpu_skew	任何層的最大CPU用量與所有層的平均CPU用量比率。此指標在區段層級中定義。 有效值為 0–99。
I/O 扭曲	io_skew	任何分割讀取區塊數上限 (I/O) 與所有分割平均讀取區塊數的比率。此指標在區段層級中定義。 有效值為 0–99。
已聯結的資料列	join_row_count	聯結步驟中處理的資料列數。 有效值為 0–999,999,999,999,999。
巢狀迴圈聯結資料列數	nested_loop_join_row_count	巢狀迴圈聯結中的數字或資料列。 有效值為 0–999,999,999,999,999。
傳回的資料列數	return_row_count	查詢傳回的資料列數。 有效值為 0–999,999,999,999,999。
區段執行時間	segment_execution_time	單一區段經過的執行時間 (秒)。為了避免或減少取樣錯誤，請在規則中包含 <code>segment_execution_time > 10</code> 。 有效值為 0–86,388。
Spectrum 掃描的資料列數	spectrum_scan_row_count	Amazon Redshift Spectrum 查詢在 Amazon S3 中掃描的資料列數。 有效值為 0–999,999,999,999,999。

指標	名稱	描述
Spectrum 掃描大小	spectrum_scan_size_mb	Amazon Redshift Spectrum 查詢在 Amazon S3 中掃描的資料大小 (以 MB 為單位)。 有效值為 0–999,999,999,999,999。
查詢優先順序	query_priority	查詢的優先順序。 有效值 為 HIGHEST、HIGH、NORMAL、LOW、LOWEST。 使用大於 (>) 和小於 (<) 運算子比較 query_priority 時，HIGHEST 大於 HIGH、HIGH 大於 NORMAL，以此類推。

Note

- query_queue_time 述詞不支援跳轉動作。也就是說，會忽略定義為在符合 query_queue_time 述詞時進行跳轉的規則。
- 區段執行時間太短可能會導致某些指標取樣錯誤，例如 io_skew 和 query_cpu_usage_percent。為了避免或減少取樣錯誤，請在規則中包含區段執行時間。segment_execution_time > 10 是很理想的起點。

[SVL_QUERY_METRICS](#) 檢視顯示已完成之查詢的指標。[SVL_QUERY_METRICS_SUMMARY](#) 檢視顯示已完成之查詢的指標最大值。請使用這些檢視中的值來協助決定臨界值，以定義查詢監控規則。

Amazon Redshift Serverless 的查詢監控指標

下表描述 Amazon Redshift Serverless 查詢監控規則中使用的指標。

指標	名稱	描述
查詢CPU時間	max_query_cpu_time	CPU 查詢使用的時間，以秒為單位。CPU time 與不同 Query execution time。 有效值為 0–999,999。

指標	名稱	描述
已讀取的區塊	max_query_blocks_read	查詢所讀取的 1 MB 資料區塊數。 有效值為 0–1,048,575。
掃描的資料列數	max_scan_row_count	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。 有效值為 0–999,999,999,999,999。
查詢執行時間	max_query_execution_time	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。如果查詢超過設定的執行時間，Amazon Redshift Serverless 會停止查詢。 有效值為 0–86,399。
查詢佇列時間	max_query_queue_time	在佇列中等待的時間 (秒)。 有效值為 0 至 86,399。
CPU 用量	max_query_cpu_usage_percent	查詢使用的CPU容量百分比。 有效值為 0–6,399。
記憶體到磁碟	max_query_temp_blocks_to_disk	用來寫入中間結果的暫時磁碟空間，以 1 MB 區塊為單位。 有效值為 0–319,815,679。
已聯結的資料列	max_join_row_count	聯結步驟中處理的資料列數。 有效值為 0–999,999,999,999,999。
巢狀迴圈聯結資料列數	max_nested_loop_join_row_count	巢狀迴圈聯結中的數字或資料列。 有效值為 0–999,999,999,999,999。

Note

- `max_query_queue_time` 述詞不支援跳轉動作。也就是說，會忽略定義為在符合 `max_query_queue_time` 述詞時進行跳轉的規則。
- 區段執行時間太短可能會導致某些指標取樣錯誤，例如 `max_io_skew` 和 `max_query_cpu_usage_percent`。

查詢監控規則範本

當您使用 Amazon Redshift 主控台新增規則時，您可以選擇從預先定義的範本建立規則。Amazon Redshift 會建立包含一組述詞的新規則，並以預設值填入述詞。預設動作為記錄。您可以修改述詞和動作，以符合您的使用案例。

下表列出可用的範本。

範本名稱	述詞	描述
巢狀迴路聯結	<code>nested_loop_join_row_count > 100</code>	巢狀迴圈聯結可能表示聯結述詞不完整，通常會產生非常大的傳回集 (笛卡兒乘積)。請使用較小的資料列數，以儘早發現潛在的失控查詢。
查詢傳回的大量的資料列數	<code>return_row_count > 1000000</code>	如果您將佇列專用於簡單、短期執行的查詢，您可以包含規則來找出傳回較多資料列數的查詢。範本以 1 百萬個資料列為預設值。對於某些系統，您可能認為 1 百萬個資料列太多，或在較大的系統中，十億或更多個資料列可能太多。
一同加入大量的資料列數	<code>join_row_count > 1000000000</code>	涉及非常多資料列的聯結步驟可能表示需要更嚴格的篩選條件。範本以十億個資料列為預設值。對於用於快速、簡單查詢的隨機操作 (一次性) 佇列，您可以使用較小的數目。
寫入中繼結果時出現高磁碟用量	<code>query_template_blocks_to_disk > 100000</code>	目前執行查詢時，查詢執行引擎會將中繼結果寫入磁碟 (溢出的記憶體) RAM。這種情況通常是惡意查詢所引起，這通常也是使用最多磁碟空間的查詢。可接受的磁碟用量臨界值根據叢集節

範本名稱	述詞	描述
		點類型和節點數而有所不同。範本以 100,000 個區塊或 100 GB 為預設值。對於小型叢集，您可以使用較小的數目。
查詢執行時間長且 I/O 高度扭曲	segment_execution_time > 120 和 io_skew > 1.30	當一個節點配量的 I/O 速率比其他配量高出許多時，就會發生 I/O 扭曲。以扭曲 1.30 (平均值的 1.3 倍) 為起點應該太高。I/O 扭曲偏高不一定是問題，但加上執行查詢時間太長時，就可能表示分佈樣式或排序索引鍵有問題。

查詢監控規則的系統資料表和檢視

符合所有規則的述詞時，會將資料列 WLM 寫入 [STL_WLM_RULE_ACTION](#) 系統資料表。這一系列包含觸發規則的查詢和所產生動作的詳細資訊。

此外，Amazon Redshift 會在下列系統資料表和檢視中記錄查詢指標。

- [STV_QUERY_METRICS](#) 資料表會顯示目前執行之查詢的指標。
- [STL_QUERY_METRICS](#) 資料表會記錄以完成之查詢的指標。
- [SVL_QUERY_METRICS](#) 檢視顯示已完成之查詢的指標。
- [SVL_QUERY_METRICS_SUMMARY](#) 檢視顯示已完成之查詢的指標最大值。

WLM 系統資料表和檢視

WLM 根據內部定義的 WLM 服務類別設定查詢佇列。Amazon Redshift 會根據這些服務類別以及 WLM 組態中定義的佇列來建立多個內部佇列。在系統資料表中，佇列和服務類別這兩個詞通常互通使用。超級使用者佇列使用服務類別 5。使用者定義的查詢使用服務類別 6 以上。

您可以使用 WLM 特定的系統資料表來檢視查詢、佇列和服務類別的狀態。查詢下列系統資料表來執行下列動作：

- 檢視工作負載管理員正在追蹤哪些查詢及已配置什麼資源。
- 查看已將查詢指派給哪一個佇列。
- 檢視工作負載管理員目前追蹤之查詢的狀態。

資料表名稱	描述
STL_WLM_ERROR	包含 WLM 相關錯誤事件的日誌。
STL_WLM_QUERY	列出正在追蹤的查詢 WLM。
STV_WLM_CLASSIFICATION_CONFIG	顯示目前的分類規則 WLM。
STV_WLM_QUERY_QUEUE_STATE	記錄查詢佇列的目前狀態。
STV_WLM_QUERY_STATE	提供追蹤的查詢目前狀態的快照 WLM。
STV_WLM_QUERY_TASK_STATE	包含查詢任務的目前狀態。
STV_WLM_SERVICE_CLASS_CONFIG	記錄的服務類別組態 WLM。
STV_WLM_SERVICE_CLASS_STATE	包含服務類別的目前狀態。
STL_WLM_RULE_ACTION	記錄因與使用者定義佇列相關聯的 WLM 查詢監控規則所產生的動作詳細資訊。
STV_WLM_QMR_CONFIG	記錄 WLM 查詢監控規則的組態 (QMR)。

您可以在系統資料表中使用任務 ID 來追蹤查詢。以下範例示範如何取得最近提交之使用者查詢的任務 ID：

```
select task from stl_wlm_query where exec_start_time =(select max(exec_start_time) from
  stl_wlm_query);

task
-----
137
(1 row)
```

下列範例顯示目前正在執行或在各種服務類別 (佇列) 中等待的查詢。此查詢很適用於追蹤 Amazon Redshift 的整體並行工作負載：

```
select * from stv_wlm_query_state order by query;
```

```
xid |task|query|service_| wlm_start_ | state |queue_ | exec_
   |   |   |class  | time      |      |time   | time
-----+-----+-----+-----+-----+-----+-----+-----
2645| 84 | 98 | 3      | 2010-10-... |Returning| 0 | 3438369
2650| 85 | 100 | 3     | 2010-10-... |Waiting | 0 | 1645879
2660| 87 | 101 | 2     | 2010-10-... |Executing| 0 | 916046
2661| 88 | 102 | 1     | 2010-10-... |Executing| 0 | 13291
(4 rows)
```

WLM 服務類別 IDs

下表列出IDs指派給服務類別的。

ID	服務類別
1–4	保留以供系統使用。
5	供超級使用者佇列使用。
6–13	由WLM組態中定義的手動WLM佇列使用。
14	供短期查詢加速使用。
15	保留給 Amazon Redshift 執行的維護活動。
100–107	當 auto_wlm 為 true 時，由自動WLM佇列使用。

資料庫安全

您可以控制哪些使用者可以存取哪些資料庫物件來管理資料庫安全性。使用者可以指派角色或群組，而您授予使用者、角色或群組的許可會決定他們可以存取哪些資料庫物件。

主題

- [Amazon Redshift 安全性概觀](#)
- [預設的資料庫使用者許可](#)
- [超級使用者](#)
- [使用者](#)
- [群組](#)
- [結構描述](#)
- [角色型存取控制 \(RBAC \)](#)
- [資料列層級安全性](#)
- [中繼資料安全性](#)
- [動態資料遮罩](#)
- [限定範圍權限](#)

對資料庫物件的存取取決於您授予使用者或角色的許可。下列準則彙總資料庫安全性的運作方式：

- 依預設，只對物件擁有者授予許可。
- Amazon Redshift 資料庫使用者是可以連接至資料庫的具名使用者。使用者會以兩種方式獲得許可：直接明確地或隱含地將這些許可指派給帳戶、成為已獲得許可的群組成員。
- 群組是可以集體獲指派許可，以簡化安全維護的使用者集合。
- 結構描述是資料庫資料表和其他資料庫物件的集合。結構描述與檔案系統目錄相似，但結構描述無法巢狀化。使用者可獲得存取單一結構描述或多個結構描述的權限。

此外，Amazon Redshift 還採用下列功能，讓您更好地控制哪些使用者可以存取哪些資料庫物件：

- 角色型存取控制 (RBAC) 可讓您將許可指派給角色，然後您可以套用至使用者，讓您控制大型使用者群組的許可。與群組不同，角色可以繼承其他角色的許可。

資料列層級安全性 (RLS) 可讓您定義限制存取所選資料列的政策，然後將這些政策套用至使用者或群組。

動態資料遮罩 (DDM) 會在查詢執行期進行轉換，進一步保護您的資料，以便允許使用者存取資料，而不會洩露敏感詳細資訊。

如需安全性實作的範例，請參閱[控制使用者和群組存取的範例](#)。

如需保護資料的相關資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift](#) 中的安全性。

Amazon Redshift 安全性概觀

Amazon Redshift 資料庫安全性不同於其他類型的 Amazon Redshift 安全性。除了本節描述的資料庫安全性外，Amazon Redshift 還提供這些功能來管理安全性：

- 登入憑證 — 對 Amazon Redshift AWS 管理主控台的存取由 AWS 您的帳戶許可控制。如需詳細資訊，請參閱[登入資料](#)。
- 存取管理 — 若要控制對特定 Amazon Redshift 資源的存取，您可以定義 AWS Identity and Access Management (IAM) 帳戶。(如需詳細資訊，請參閱[控制 Amazon Redshift 資源的存取](#))。
- 叢集安全群組 — 若要授予其他使用者 Amazon Redshift 叢集的傳入存取，您可以定義叢集安全群組，並將它與叢集建立關聯。如需詳細資訊，請參閱 [Amazon Redshift 叢集安全群組](#)。
- VPC — 若要使用虛擬聯網環境來保護叢集的存取，您可以在 Amazon Virtual Private Cloud () 中啟動叢集VPC。如需詳細資訊，請參閱在[虛擬私有雲端 \(VPC \) 中管理叢集](#)。
- 叢集加密 — 若要加密您的所有使用者建立的資料表中的資料，您可以在啟動叢集時開啟叢集加密。如需詳細資訊，請參閱 [Amazon Redshift 叢集](#)。
- SSL 連線 — 若要加密SQL用戶端與叢集之間的連線，您可以使用安全通訊端層 (SSL) 加密。如需詳細資訊，請參閱[使用 連線至叢集SSL](#)。
- 載入資料加密 — 若要在您將資料表載入資料檔案上傳至 Amazon S3 時加密這些檔案，您可以使用伺服器端加密或用戶端加密。當您從伺服器端加密的資料載入時，Amazon S3 會以透明的方式處理解密。當您從用戶端加密資料載入時，Amazon Redshift COPY命令會在載入資料表時解密資料。如需詳細資訊，請參閱[將加密資料上傳到 Amazon S3](#)。
- 傳輸中的資料 — 為了保護您的 AWS Cloud 內傳輸中的資料，Amazon Redshift 使用加速SSL的硬體與 Amazon S3 或 Amazon DynamoDB 通訊，以進行 COPY、UNLOAD、備份和還原操作。
- 資料欄層級存取控制 — 若要讓資料的資料欄層級在 Amazon Redshift 中存取控制，請使用資料欄層級授權及撤銷陳述式，而不需實作檢視型存取控制或使用其他系統。

- 資料列層級安全控制 — 若要具有 Amazon Redshift 中資料的資料列層級安全控制，請建立政策並將政策附加至角色或使用者，限制他們存取政策中定義的資料列。

預設的資料庫使用者許可

當您建立資料庫物件時，您是其擁有者。依預設，只有超級使用者或物件擁有者才能查詢物件、修改物件，或授予物件許可。對於要使用物件的使用者，您必須將必要的許可授予使用者或包含使用者的群組。資料庫超級使用者具有與資料庫擁有者相同的許可。

Amazon Redshift 支援下列許可：

SELECT、INSERT、UPDATE、DELETE、REFERENCES、CREATE、TEMPORARY 和 USAGE。不同的許可與不同的物件類型相關聯。如需有關 Amazon Redshift 支援的資料庫物件許可資訊，請參閱 [GRANT](#) 命令。

只有擁有者才有修改或銷毀物件的許可。

根據預設，所有使用者都具有資料庫結構描述的 PUBLIC CREATE 和 USAGE 許可。若要禁止使用者在資料庫的結構描述中建立物件，請使用 PUBLIC REVOKE 命令移除該許可。

若要撤銷先前授予的許可，請使用 [REVOKE](#) 命令。物件擁有者的許可，例如 DROP、GRANT 和 REVOKE 許可是隱含的，無法授予或撤銷。物件擁有者可以撤銷自己的普通許可，例如，使自己以及其他只能讀取資料表。無論 GRANT 和 REVOKE 命令為何，超級使用者都會保留所有許可。

超級使用者

資料庫超級使用者具有與所有資料庫之資料庫擁有者相同的許可。

admin 使用者 (即您啟動叢集時所建立的使用者) 是超級使用者。

您必須是超級使用者才能建立超級使用者。

Amazon Redshift 系統資料表和系統檢視分為只有超級使用者才能看見或所有使用者都能看見。只有超級使用者才能查詢指定為「超級使用者才能看見」的系統資料表和系統檢視。如需相關資訊，請參閱 [SYS 監控檢視](#)。

超級使用者可以檢視所有目錄資料表。如需相關資訊，請參閱 [系統目錄資料表](#)。

資料庫超級使用者會略過所有許可檢查。無論 GRANT 和 REVOKE 命令為何，超級使用者都會保留所有許可。使用超級使用者角色時必須謹慎。建議您以不是超級使用者的角色執行大部分工作。您可以使用更嚴格的許可建立管理員角色。如需有關建立角色的詳細資訊，請參閱 [角色型存取控制 \(RBAC\)](#)。

若要建立新的資料庫超級使用者，請以超級使用者身分登入資料庫，並使用 CREATEUSER 許可發出 CREATEUSER 命令或 ALTERUSER 命令。

```
CREATE USER adminuser CREATEUSER PASSWORD '1234Admin';  
ALTER USER adminuser CREATEUSER;
```

若要建立、修改或捨棄超級使用者，請使用相同的命令來管理使用者。如需詳細資訊，請參閱[建立、更改和刪除使用者](#)。

使用者

您可以使用 Amazon Redshift SQL 命令 和 ALTER 來建立 CREATEUSER 和管理資料庫使用者 USER。或者，您可以使用自訂 Amazon Redshift JDBC 或 ODBC 驅動程式來設定 SQL 用戶端。這些驅動程式會管理隨著資料庫登入的一部分建立使用者資料庫和暫時密碼的程序。

驅動程式會根據 AWS Identity and Access Management (IAM) 身分驗證來驗證資料庫使用者。如果您已在 之外管理使用者身分 AWS，您可以使用符合 SAML 2.0 標準的身分提供者 IdP) 來管理對 Amazon Redshift 資源的存取。您可以使用 IAM 角色來設定 IdP AWS，並允許聯合使用者產生臨時資料庫憑證並登入 Amazon Redshift 資料庫。如需詳細資訊，請參閱[使用 IAM 身分驗證來產生資料庫使用者憑證](#)。

Amazon Redshift 使用者帳戶只能由資料庫超級使用者建立和捨棄。使用者在登入 Amazon Redshift 時會進行身份驗證。他們可以擁有資料庫和資料庫物件 (例如資料表)。他們也可以將這些物件的許可授予使用者、群組和結構描述，以控制誰有權存取哪個物件。具有 CREATEDATABASE 權限的使用者可以建立資料庫，並授予這些資料庫的許可。超級使用者具有所有資料庫的資料庫所有權。

建立、更改和刪除使用者

資料庫使用者適用於資料倉儲叢集 (而且不是根據個別資料庫)。

- 若要建立使用者，請使用 [CREATE USER](#) 命令。
- 若要建立超級使用者，請使用 [CREATE USER](#) 命令搭配 CREATEUSER 選項。
- 若要移除現有的使用者，請使用 [DROP USER](#) 命令。
- 若要變更使用者 (例如變更密碼)，請使用 [ALTER USER](#) 命令。
- 若要檢視使用者清單，請查詢 PG_USER 目錄表。

```
select * from pg_user;
```

```

username | usesysid | usecreatedb | usesuper | usecatupd | passwd | valuntil |
useconfig
-----+-----+-----+-----+-----+-----+-----
+-----+
rdsdb    |         1 | t           | t         | t         | ***** |          |
masteruser |       100 | t           | t         | f         | ***** |          |
dwuser    |       101 | f           | f         | f         | ***** |          |
simpleuser |       102 | f           | f         | f         | ***** |          |
poweruser |       103 | f           | t         | f         | ***** |          |
dbuser    |       104 | t           | f         | f         | ***** |          |
(6 rows)

```

群組

群組是全部被授予與群組相關聯之任何許可的使用者集合。您可以使用群組來指派許可。例如，您可以為銷售、管理和支援建立不同的群組，並給與每個群組中的使用者適當權限，讓他們可在工作時存取所需的資料。您可以授予或撤銷群組層級的許可，而且那些變更將適用於群組的所有成員，但超級使用者除外。

若要檢視所有使用者群組，請查詢 PG_GROUP 系統目錄表：

```
select * from pg_group;
```

例如，若要依群組列出所有資料庫使用者，請執行下列 SQL。

```

SELECT u.usesysid
 ,g.groname
 ,u.username
FROM pg_user u
LEFT JOIN pg_group g ON u.usesysid = ANY (g.groplist)

```

建立、更改和刪除群組

只有超級使用者才能建立、更改或捨棄群組。

您可以執行下列動作：

- 若要建立群組，請使用 [CREATE GROUP](#) 命令。
- 若要將使用者新增至現有群組或從中移除使用者，請使用 [ALTER GROUP](#) 命令。

- 若要刪除群組，請使用 [DROP GROUP](#) 命令。此命令只會捨棄群組，不會捨棄其成員使用者。

控制使用者和群組存取的範例

此範例會建立使用者群組和使用者，然後將連接至 Web 應用程式用戶端之 Amazon Redshift 資料庫的各種許可授予他們。此範例採用三個使用者群組：Web 應用程式的一般使用者、Web 應用程式的進階使用者，以及 Web 開發人員。

1. 建立將在其中指派使用者的群組。下列一組命令會建立三個不同的使用者群組：

```
create group webappusers;  
  
create group webpowerusers;  
  
create group webdevusers;
```

2. 建立數個具有不同許可的資料庫使用者，並將他們新增至群組。

- a. 建立兩個使用者並將其新增至WEBAPPUSERS群組：

```
create user webappuser1 password 'webAppuser1pass'  
in group webappusers;  
  
create user webappuser2 password 'webAppuser2pass'  
in group webappusers;
```

- b. 建立 Web 開發人員使用者並將其新增至WEBDEVUSERS群組：

```
create user webdevuser1 password 'webDevuser2pass'  
in group webdevusers;
```

- c. 建立一個超級使用者。此使用者將具有建立其他使用者的管理權限：

```
create user webappadmin password 'webAppadminpass1'  
createuser;
```

3. 建立要與 Web 應用程式所使用之資料庫資料表相關聯的結構描述，並將各種使用者群組存取授予這個結構描述：

- a. 建立WEBAPP結構描述：

```
create schema webapp;
```

b. 授予 WEBAPPUSERS 群組 USAGE 許可：

```
grant usage on schema webapp to group webappusers;
```

c. 授予 WEBPOWERUSERS 群組 USAGE 許可：

```
grant usage on schema webapp to group webpowerusers;
```

d. 授予 WEBDEVUSERS 群組 ALL 許可：

```
grant all on schema webapp to group webdevusers;
```

基本使用者和群組現在已完成設定。您現在可以修改使用者和群組。

4. 例如，下列命令會變更的 `search_path` 參數 WEBAPPUSER1。

```
alter user webappuser1 set search_path to webapp, public;
```

SEARCH_PATH 會指定資料庫物件的結構描述搜尋順序，例如資料表和函數，而該物件是由未指定結構描述的簡單名稱所參考。

5. 您也可以在建立群組之後將使用者新增至群組，例如 WEBAPPUSER2 新增至 WEBPOWERUSERS 群組：

```
alter group webpowerusers add user webappuser2;
```

結構描述

資料庫包含一或多個具名結構描述。資料庫中的每個結構描述都包含資料表和其他類型的具名物件。根據預設，資料庫具有名為的單一結構描述 PUBLIC。您可以使用結構描述，在常見名稱下將資料庫物件分組。結構描述與檔案系統目錄相似，但結構描述無法巢狀化。

相同的資料庫物件名稱可在相同資料庫的不同結構描述中使用，而不會發生衝突。例如，MY_SCHEMA 和 YOUR_SCHEMA 都可以包含名為的資料表 MYTABLE。具有必要許可的使用者可以跨資料庫中的多個結構描述存取物件。

依預設，物件是在資料庫搜尋路徑中的第一個結構描述內建立。如需相關資訊，請參閱本節後續的[搜尋路徑](#)。

結構描述可以協助在多使用者環境中採用下列方式解決組織和並行問題：

- 讓多個開發人員可在同一個資料庫中工作，而彼此不會互相干擾。
- 將資料庫物件組織成邏輯群組，讓它們更便於管理。
- 讓應用程式能夠將其物件放入個別結構描述中，以便其名稱將不會與其他應用程式使用的物件名稱發生衝突。

搜尋路徑

搜尋路徑是在 `search_path` 參數中搭配逗號分隔的結構描述名稱清單定義的。當以未包括結構描述限定詞的簡單名稱參考物件 (如資料表或函數) 時，此搜尋路徑會指定搜尋結構描述的順序。

如果在未指定目標結構描述的情況下建立物件，則物件會新增至搜尋路徑中列出的第一個結構描述。當名稱相同的物件存在於不同結構描述時，未指定結構描述的物件名稱將參考搜尋路徑中的第一個結構描述，其中包含具有該名稱的物件。

若要變更目前工作階段的預設結構描述，請使用 [SET](#) 命令。

如需詳細資訊，請參閱《組態參考》中的 [search_path](#) 說明。

建立、更改和刪除結構描述

任何使用者都可以建立結構描述，以及更改或捨棄他們擁有的結構描述。

您可以執行下列動作：

- 若要建立結構描述，請使用 [CREATE SCHEMA](#) 命令。
- 若要變更結構描述的擁有者，請使用 [ALTER SCHEMA](#) 命令。
- 若要刪除結構描述及其物件，請使用 [DROP SCHEMA](#) 命令。
- 若要在結構描述內建立資料表，請建立格式為 `schema_name.table_name` 的資料表。

若要檢視所有結構描述的清單，請查詢 `PG_NAMESPACE` 系統目錄表：

```
select * from pg_namespace;
```

若要檢視屬於結構描述的資料表清單，請查詢 `PG_TABLE_DEF` 系統目錄資料表。例如，下列查詢會傳回 `PG_CATALOG` 結構描述中的資料表清單。

```
select distinct(tablename) from pg_table_def
where schemaname = 'pg_catalog';
```

結構描述型許可

結構描述型許可是由結構描述的擁有者決定：

- 根據預設，所有使用者都具有資料庫結構描述的 PUBLIC CREATE和 USAGE許可。若要禁止使用者在資料庫的結構描述中建立物件，請使用 PUBLIC [REVOKE](#)命令移除該許可。
- 除非物件擁有者授予USAGE許可，否則使用者無法存取他們不擁有的結構描述中的任何物件。
- 如果已授予使用者由其他使用者所建立結構描述的CREATE許可，則這些使用者可以在該結構描述中建立物件。

角色型存取控制（RBAC）

透過使用角色型存取控制（RBAC）來管理 Amazon Redshift 中的資料庫許可，您可以簡化 Amazon Redshift 中安全許可的管理。您可以控制使用者在廣泛或精細層級上執行的動作，以確保敏感資料的存取安全。您也可以控制使用者存取通常僅限超級使用者存取的工作。透過將不同許可指派給不同角色，並將其指派給不同的使用者，您可以更精細地控制使用者存取權。

獲派角色的使用者只能執行其授權指派角色所指定的工作。例如，具有 CREATETABLE和 DROPTABLE許可的指派角色的使用者，僅獲授權執行這些任務。您可以透過授予不同層級的安全許可給不同的使用者，以控制使用者存取權，讓他們能存取工作所需的資料。

RBAC 根據使用者的角色需求，將最低許可原則套用至使用者，無論涉及的物件類型為何。授予和撤銷許可是在角色層級上執行的，而不需要更新個別資料庫物件的許可。

使用 RBAC，您可以建立具有執行命令的許可角色，這些命令用於要求超級使用者許可。使用者只要獲得包含這些許可的角色授權，就可以執行這些命令。同樣地，您也可以建立角色來限制對某些命令的存取，並將角色指派給已獲得該角色授權的超級使用者或使用者。

若要了解 Amazon Redshift 的運作方式RBAC，請觀看下列影片：在 [Amazon Redshift 中介紹角色型存取控制（RBAC）](#)。

角色階層

角色是您可以指派給使用者或其他角色的許可集合。您可以將系統或資料庫許可指派給角色。使用者會繼承指派角色的許可。

在中RBAC，使用者可以有巢狀角色。您可以將角色授予使用者和角色。將角色授予使用者時，您可以使用此角色包含的所有權限來授權使用者。將角色 r1 授予使用者時，您可以使用 r1 的許可來授權使用者。使用者現在擁有 r1 的許可，以及他們已有的任何現有許可。

將角色 (r1) 授予另一個角色 (r2) 時，您可以使用 r1 的所有許可來授權 r2。此外，將 r2 授予另一個角色 (r3) 時，r3 的許可會結合 r1 和 r2 的許可。角色階層會讓 r2 從 r1 繼承許可。Amazon Redshift 會透過每個角色授權來傳播許可。將 r1 授予 r2，然後再將 r2 授予 r3，即可將這三個角色的所有許可授權給 r3。因此，透過將 r3 授予使用者，使用者就會擁有這三個角色的所有許可。

Amazon Redshift 不允許建立角色授權循環。當巢狀角色被指派回角色階層中較早的角色 (例如 r3 被指派回 r1) 時，就會發生角色授權循環。如需建立角色及管理角色指派的相關資訊，請參閱 [在中管理角色 RBAC](#)。

角色指派

具有 CREATEROLE 許可的超級使用者和一般使用者可以使用 CREATEROLE 陳述式來建立角色。超級使用者和角色管理員可以使用 GRANTROLE 陳述式將角色授予其他人。他們可以使用 REVOKEROLE 陳述式來撤銷其他人的角色，以及使用 DROPROLE 陳述式來捨棄角色。角色管理員包括角色擁有者和已取得 ADMINOPTION 許可之角色的使用者。

只有超級使用者或角色管理員可以授予和撤銷角色。您可以對一個或多個角色或使用者授予或撤銷一個或多個角色。使用 GRANTROLE 陳述式中的 WITHADMINOPTION 選項，為所有授予的角色提供管理選項。

Amazon Redshift 支援不同的角色指派組合，例如授予多個角色或擁有多個承授者。WITH ADMIN OPTION 僅適用於使用者，不適用於角色。同樣地，請使用 REVOKEROLE 陳述式中的 WITHADMINOPTION 選項，從受讓者移除角色和管理授權。與 ADMIN 搭配使用時 OPTION，只會從角色撤銷管理授權。

下列範例會從 user2 中撤銷 sample_role2 角色的管理授權。

```
REVOKE ADMIN OPTION FOR sample_role2 FROM user2;
```

如需建立角色及管理角色指派的相關資訊，請參閱 [在中管理角色 RBAC](#)。

Amazon Redshift 系統定義角色

Amazon Redshift 會提供使用特定許可定義的系統定義角色。系統專屬角色會以 sys: 字首開頭。只有具有適當存取權的使用者才能修改系統定義角色或建立自訂的系統定義角色。您無法針對自訂系統定義角色使用 sys: 字首。

下表摘要說明角色及其許可。

角色名稱	描述		
sys:monitor	此角色具有存取目錄或系統資料表的許可。		
sys:operator	此角色具有存取目錄或系統資料表、分析、清除或取消查詢的許可。		
sys:dba	此角色具有建立結構描述、建立資料表、捨棄結構描述、捨棄資料表和截斷資料表的許可。其有權建立或取代預存程序、捨棄程序、建立或替代函數、建立或取代外部函數、建立檢視和捨棄檢視。此外，此角色會繼承 sys:operator 角色的所有許可。		
sys:superuser	此角色具有 的系統許可 RBAC 中定義的所有受支援的系統權限。		
sys:secadmin	<ul style="list-style-type: none"> 此角色有權建立使用者、修改使用者、捨棄使用者、建立角色、捨棄角色和授予角色許可。 此角色具有RLS開啟 或OFF開啟關 		

角色名稱	描述		
	<p>係的許可，以及管理 RLS和 DDM政策 DROP (CREATE、ATTACH、DETACH和) 的許可ALTER。此外，請注意，預設會將 EXPLAIN RLS、RLS IGNORE 和 EXPLAINMASKING 許可授予此角色。</p> <ul style="list-style-type: none"> 只有將許可明確授予該角色時，此角色才能存取使用者資料表。 		

用於資料共用的系統定義角色和使用者

Amazon Redshift 會建立與資料共用和資料共用取用者相對應的角色和使用者以供內部使用。每個內部角色名稱和使用者名稱都有預留的命名空間字首 `ds:`。它們具有下列格式：

名稱	描述		
<code>ds:sharename</code>	與資料共用對應的系統角色。		
<code>ds:sharename_consumer</code>	與資料共用取用者對應的系統使用者。		

會為每個資料共用建立資料共用角色。它保留目前授予資料共用的所有許可。會為每個資料共用的取用者建立資料共用使用者。它被授予單一資料共用角色的許可。新增至多個資料共用的取用者會為每個資料共用建立資料共用使用者。

這些使用者和角色是資料共用正常運作的必要條件。它們無法修改或捨棄，也無法存取或用於客戶執行的任何任務。您可以安全地忽略它們。如需資料共用的詳細資訊，請參閱 [Amazon Redshift 中的跨叢集共用資料](#)。

Note

您無法使用 `ds:` 字首來建立使用者定義的角色或使用者。

的系統許可 RBAC

下列是您可以對角色授予或撤銷的系統許可清單。

Command	您必須具有下列其中一種執行命令的許可		
CREATE ROLE	<ul style="list-style-type: none"> 超級使用者。 具有 CREATEROLE 許可的使用者。 		
DROP ROLE	<ul style="list-style-type: none"> 超級使用者。 角色擁有者，可以是建立角色的使用者，或是已透過 WITHADMIN OPTION 許可授予角色的使用者。 		
CREATE USER	<ul style="list-style-type: none"> 超級使用者。 具有 CREATEUSER 許可的使用者。這些使用者無法建立超級使用者。 		
DROP USER	<ul style="list-style-type: none"> 超級使用者。 具有 DROPUSER 許可的使用者。 		

Command	您必須具有下列其中一種執行命令的許可		
ALTER USER	<ul style="list-style-type: none"> • 超級使用者。 • 具有 ALTERUSER 許可的使用者。這些使用者無法將使用者變更為超級使用者或將超級使用者變更為使用者。 • 想要變更自己密碼的目前使用者。 		
CREATE SCHEMA	<ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATESCH EMA許可的使用者。 		
DROP SCHEMA	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROPSCHEM A許可的使用者。 • 結構描述擁有者。 		
ALTER DEFAULT PRIVILEGES	<ul style="list-style-type: none"> • 超級使用者。 • 具有 ALTERDEFA ULTPRIVILEGES許 可的使用者。 • 變更自身預設存取許可的使用者 • 為有權存取的結構描述設定許可的使用者。 		

Command	您必須具有下列其中一種執行命令的許可		
CREATE TABLE	<ul style="list-style-type: none">• 超級使用者。• 具有 CREATE TABLE 許可的使用者。• 具有結構描述 CREATE 許可的使用者。		
DROP TABLE	<ul style="list-style-type: none">• 超級使用者。• 具有 DROP TABLE 許可的使用者。• 具有結構描述 USAGE 許可的資料表擁有者。		
ALTER TABLE	<ul style="list-style-type: none">• 超級使用者。• 具有 ALTER TABLE 許可的使用者。• 具有結構描述 USAGE 許可的資料表擁有者。		

Command	您必須具有下列其中一種執行命令的許可		
CREATE 或 REPLACE FUNCTION	<ul style="list-style-type: none"> • 對於 CREATE FUNCTION : <ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACEFUNCTION許可的使用者。 • 具有語言USAGE許可的使用者。 • 對於 REPLACE FUNCTION : <ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACEFUNCTION許可的使用者。 • 函數擁有者。 		
CREATE 或 REPLACE EXTERNAL FUNCTION	<ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACEEXTERNALFUNCTION許可的使用者。 		
DROP FUNCTION	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROPFUNCTION許可的使用者。 • 函數擁有者。 		

Command	您必須具有下列其中一種執行命令的許可
CREATE 或 REPLACE PROCEDURE	<ul style="list-style-type: none"> • 對於 CREATE PROCEDURE : <ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACE PROCEDURE 許可的使用者。 • 具有語言 USAGE 許可的使用者。 • 對於 REPLACE PROCEDURE : <ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACE PROCEDURE 許可的使用者。 • 程序擁有者。
DROP PROCEDURE	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROP PROCEDURE 許可的使用者。 • 程序擁有者。

Command	您必須具有下列其中一種執行命令的許可		
CREATE 或 REPLACE VIEW	<ul style="list-style-type: none"> • 對於 CREATE VIEW : <ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACEVIEW許可的使用者。 • 具有結構描述 CREATE許可的使用者。 • 對於 REPLACE VIEW : <ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE OR REPLACEVIEW許可的使用者。 • 檢視擁有者。 		
DROP VIEW	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROPVIEW許可的使用者。 • 檢視擁有者。 		

Command	您必須具有下列其中一種執行命令的許可		
CREATE MODEL	<ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE MODEL 系統許可的使用者，他們應該能夠讀取的關係 CREATE MODEL。 • 具有 CREATE MODEL 許可的使用者。 		
DROP MODEL	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROP MODEL 許可的使用者。 • 模型擁有者。 • 結構描述擁有者。 		
CREATE DATASHARE	<ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATE DATASHARE 許可的使用者。 • 資料庫擁有者。 		

Command	您必須具有下列其中一種執行命令的許可		
ALTER DATASHARE	<ul style="list-style-type: none"> • 超級使用者。 • 具有 ALTERDATA SHARE許可的使用者。 • 擁有 ALTER或 資料共用ALL許可的使用者。 • 若要將特定物件新增至資料共用，這些使用者必須要有物件的許可。使用者應該是物件的擁有者SELECT，或對物件具有 USAGE、或 ALL許可。 		
DROP DATASHARE	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROPDATAS HARE許可的使用者。 • 資料庫擁有者。 		
CREATE LIBRARY	<ul style="list-style-type: none"> • 超級使用者。 • 具有 CREATELIBRARY許可或指定語言許可的使用者。 		
DROP LIBRARY	<ul style="list-style-type: none"> • 超級使用者。 • 具有 DROPLIBRARY許可的使用者。 • 程式庫擁有者。 		

Command	您必須具有下列其中一種執行命令的許可		
ANALYZE	<ul style="list-style-type: none"> • 超級使用者。 • 具有 ANALYZE 許可的使用者。 • 關係的擁有者。 • 共用資料表的資料庫擁有者。 		
CANCEL	<ul style="list-style-type: none"> • 超級使用者取消自己的查詢。 • 超級使用者取消使用者的查詢。 • 具有取消使用者查詢 CANCEL 許可的使用者。 • 使用者取消自己的查詢。 		
TRUNCATE TABLE	<ul style="list-style-type: none"> • 超級使用者。 • 具有 TRUNCATE TABLE 許可的使用者。 • 資料表擁有者。 		
VACUUM	<ul style="list-style-type: none"> • 超級使用者。 • 具有 VACUUM 許可的使用者。 • 資料表擁有者。 • 共用資料表的資料庫擁有者。 		

Command	您必須具有下列其中一種執行命令的許可		
IGNORE RLS	<ul style="list-style-type: none"> 超級使用者。 sys:secadmin 角色中的使用者。 		
EXPLAIN RLS	<ul style="list-style-type: none"> 超級使用者。 sys:secadmin 角色中的使用者。 		
EXPLAIN MASKING	<ul style="list-style-type: none"> 超級使用者。 sys:secadmin 角色中的使用者。 		

資料庫物件許可

除了系統許可外，Amazon Redshift 還包含可定義存取選項的資料庫物件許可。其中包含的選項包括能夠讀取資料表和檢視中的資料、寫入資料、建立資料表及捨棄資料表。如需詳細資訊，請參閱[GRANT](#)。

透過使用 RBAC，您可以將資料庫物件許可指派給角色，類似於使用系統許可的方式。然後，您可以將角色指派給使用者、授權具有系統許可的使用者，以及使用資料庫許可授權使用者。

ALTER DEFAULT PRIVILEGES 適用於 RBAC

使用 ALTERDEFAULTPRIVILEGES 陳述式來定義要套用至指定使用者未來建立之物件的預設存取許可集。根據預設，使用者只能變更自己擁有的預設存取許可。使用 RBAC，您可以設定角色的預設存取權限。如需詳細資訊，請參閱 [ALTER DEFAULT PRIVILEGES](#) 命令。

RBAC 可讓您將資料庫物件許可指派給角色，類似於系統許可。然後，您可以將角色指派給使用者、使用系統和/或資料庫許可授權使用者。

中角色用量的考量 RBAC

使用 RBAC 角色時，請考慮下列事項：

- Amazon Redshift 不允許循環的角色授權。您不能將 r1 授予 r2，然後又將 r2 授予 r1。

- RBAC 適用於原生 Amazon Redshift 物件和 Amazon Redshift Spectrum 資料表。
- 身為 Amazon Redshift 管理員，您可以將叢集RBAC升級至最新的維護修補程式以開始使用。
- 只有具有CREATEROLE系統許可的超級使用者和使用者才能建立角色。
- 只有超級使用者和角色管理員可以修改和捨棄角色。
- 角色名稱不能與使用者名稱相同。
- 角色名稱不能包含無效字元，例如「:\n」。
- 角色名稱不能是保留字，例如 PUBLIC。
- 角色名稱不能以預設角色的保留字首開頭 (sys:)
- 您無法在將具有 RESTRICT 參數的角色授予其他角色時捨棄該角色。預設設定為 RESTRICT。當您嘗試捨棄繼承另一個角色的角色時，Amazon Redshift 會擲回錯誤。
- 沒有角色管理員許可的使用者無法予與或撤銷角色。
- RBAC 未完全支援系統資料表和檢視。RBAC 系統資料表和檢視的許可不會透過升級、降級或調整大小而持續存在。建議使用 [Amazon Redshift 系統定義角色](#) 來管理系統資料表和檢視許可。如需系統資料表的詳細資訊，請前往 [系統資料表和檢視參考](#)。

在 中管理角色 RBAC

欲執行下列動作，請使用下列命令：

- 若要建立角色，請使用 [CREATE ROLE](#) 命令。
- 若要重新命名角色或變更角色的擁有者，請使用 [ALTER ROLE](#) 命令。
- 若要刪除角色，請使用 [DROP ROLE](#) 命令。
- 若要將角色授予使用者，請使用 [GRANT](#) 命令。
- 若要撤銷使用者的角色，請使用 [REVOKE](#) 命令。
- 若要將系統許可授予角色，請使用 [GRANT](#) 命令。
- 若要撤銷角色的系統許可，請使用 [REVOKE](#) 命令。

若要檢視叢集或工作群組中的角色清單，請參閱 [SVV_ROLES](#)。

教學課程：使用 建立角色和查詢 RBAC

使用 RBAC，您可以建立具有執行命令的許可角色，這些命令用於要求超級使用者許可。使用者只要獲得包含這些許可的角色授權，就可以執行這些命令。

在本教學課程中，您可以使用角色型存取控制（RBAC）來管理您建立之資料庫中的許可。然後，您會連線至資料庫，並從兩個不同的角色查詢資料庫，以測試的功能RBAC。

您建立和用於查詢資料庫的兩個角色是 `sales_ro` 和 `sales_rw`。您可以建立 `sales_ro` 角色，並以具有 `sales_ro` 角色的使用者身分查詢資料。`sales_ro` 使用者只能使用 `SELECT` 命令，但無法使用 `UPDATE` 命令。然後，您以具有 `sales_rw` 角色的使用者身分建立 `sales_rw` 角色和查詢資料。`sales_rw` 使用者可以使用 `SELECT` 命令和 `UPDATE` 命令。

此外，您可以建立角色以限制對特定命令的存取，並將角色指派給超級使用者或使用者。

工作

- [必要條件](#)
- [步驟 1：建立管理員使用者](#)
- [步驟 2：設定結構描述](#)
- [步驟 3：建立唯讀使用者](#)
- [步驟 4：以唯讀使用者身分查詢資料](#)
- [步驟 5：建立讀寫使用者](#)
- [步驟 6：以具有繼承唯讀角色的使用者身分查詢資料](#)
- [步驟 7：將更新和插入許可授予讀寫角色](#)
- [步驟 8：以讀寫使用者身分查詢資料](#)
- [步驟 9：以管理員使用者身分分析和清空資料庫中的資料表](#)
- [步驟 10：以讀寫使用者身分截斷資料表](#)
- [的系統函數 RBAC \(選用\)](#)
- [的系統檢視 RBAC \(選用\)](#)
- [搭配使用資料列層級安全性 RBAC \(選用\)](#)

必要條件

- 建立與 TICKIT 範例資料庫一起載入的 Amazon Redshift 叢集或無伺服器工作群組。若要建立無伺服器工作群組，請參閱 [開始使用 Redshift Serverless 資料倉儲](#)。若要建立叢集，請參閱 [建立範例 Amazon Redshift 叢集](#)。如需 TICKIT 範例資料庫的詳細資訊，請參閱 [範本資料庫](#)。
- 具有超級使用者或角色管理員許可的使用者存取權。只有超級使用者或角色管理員可以授予或撤銷角色。如需所需許可的詳細資訊 RBAC，請參閱 [的系統許可 RBAC](#)。
- 請參閱 [中角色用量的考量 RBAC](#)。

步驟 1：建立管理員使用者

若要設定本教學課程，請在此步驟中建立資料庫管理員角色，並將其連接至資料庫管理員使用者。您必須將資料庫管理員建立為超級使用者或角色管理員。

在 Amazon Redshift [查詢編輯器 v2](#) 中執行所有查詢。

1. 若要建立管理員角色 db_admin，請使用下列範例。

```
CREATE ROLE db_admin;
```

2. 若要建立名為 dbadmin 的資料庫使用者，請使用下列範例。

```
CREATE USER dbadmin PASSWORD 'Test12345';
```

3. 若要將名為 sys : dba 的系統定義角色授予 db_admin 角色，請使用下列範例。授予 sys : dba 角色時，dbadmin 使用者可以建立結構描述和資料表。如需詳細資訊，請參閱 [Amazon Redshift 系統定義角色](#)。

步驟 2：設定結構描述

在此步驟中，您以資料庫管理員身分連線至資料庫。然後，您可以建立兩個結構描述，並將資料新增至這些結構描述。

1. 使用查詢編輯器 v2 以 dbadmin 使用者身分連線至 dev 資料庫。如需連線至資料庫的詳細資訊，請參閱 [使用查詢編輯器 v2](#)。
2. 若要建立銷售和行銷資料庫結構描述，請使用下列範例。

```
CREATE SCHEMA sales;  
CREATE SCHEMA marketing;
```

3. 若要在銷售結構描述中建立和插入值至資料表，請使用下列範例。

```
CREATE TABLE sales.cat(  
  catid smallint,  
  catgroup varchar(10),  
  catname varchar(10),  
  catdesc varchar(50)  
);  
INSERT INTO sales.cat(SELECT * FROM category);
```



```
CREATE TABLE sales.dates(  
  dateid smallint,  
  caldate date,  
  day char(3),  
  week smallint,  
  month char(5),  
  qtr char(5),  
  year smallint,  
  holiday boolean  
);  
INSERT INTO sales.dates(SELECT * FROM date);  
  
CREATE TABLE sales.events(  
  eventid integer,  
  venueid smallint,  
  catid smallint,  
  dateid smallint,  
  eventname varchar(200),  
  starttime timestamp  
);  
INSERT INTO sales.events(SELECT * FROM event);  
  
CREATE TABLE sales.sale(  
  salesid integer,  
  listid integer,  
  sellerid integer,  
  buyerid integer,  
  eventid integer,  
  dateid smallint,  
  qtysold smallint,  
  pricepaid decimal(8,2),  
  commission decimal(8,2),  
  saletime timestamp  
);  
INSERT INTO sales.sale(SELECT * FROM sales);
```

4. 若要在行銷結構描述中建立和插入值至資料表，請使用下列範例。

```
CREATE TABLE marketing.cat(  
  catid smallint,  
  catgroup varchar(10),  
  catname varchar(10),  
  catdesc varchar(50)  
);
```

```
INSERT INTO marketing.cat(SELECT * FROM category);

CREATE TABLE marketing.dates(
  dateid smallint,
  caldate date,
  day char(3),
  week smallint,
  month char(5),
  qtr char(5),
  year smallint,
  holiday boolean
);
INSERT INTO marketing.dates(SELECT * FROM date);

CREATE TABLE marketing.events(
  eventid integer,
  venueid smallint,
  catid smallint,
  dateid smallint,
  eventname varchar(200),
  starttime timestamp
);
INSERT INTO marketing.events(SELECT * FROM event);

CREATE TABLE marketing.sale(
  marketingid integer,
  listid integer,
  sellerid integer,
  buyerid integer,
  eventid integer,
  dateid smallint,
  qtysold smallint,
  pricepaid decimal(8,2),
  commission decimal(8,2),
  saletime timestamp
);
INSERT INTO marketing.sale(SELECT * FROM marketing);
```

步驟 3：建立唯讀使用者

在此步驟中，您會為唯讀角色建立唯讀角色和 Salesanalyst 使用者。銷售分析師只需要對銷售結構描述中資料表的唯讀存取權，才能完成他們指派的任務，以尋找導致最高佣金的事件。

1. 以 dbadmin 使用者身分連線至資料庫。
2. 若要建立 sales_ro 角色，請使用下列範例。

```
CREATE ROLE sales_ro;
```

3. 若要建立 Salesanalyst 使用者，請使用下列範例。

```
CREATE USER salesanalyst PASSWORD 'Test12345';
```

4. 若要授予 sales_ro 角色用量，並選取對銷售結構描述物件的存取，請使用下列範例。

```
GRANT USAGE ON SCHEMA sales TO ROLE sales_ro;
GRANT SELECT ON ALL TABLES IN SCHEMA sales TO ROLE sales_ro;
```

5. 若要授予 salesanalyst 使用者 sales_ro 角色，請使用下列範例。

```
GRANT ROLE sales_ro TO salesanalyst;
```

步驟 4：以唯讀使用者身分查詢資料

在此步驟中，Salesanalyst 使用者會從銷售結構描述查詢資料。然後，Salesanalyst 使用者嘗試更新行銷結構描述中的資料表和讀取資料表。

1. 以 Salesanalyst 使用者身分連線至資料庫。
2. 若要尋找最高佣金的 10 個銷售，請使用下列範例。

```
SET SEARCH_PATH TO sales;
SELECT DISTINCT events.dateid, sale.commission, cat.catname
FROM sale, events, dates, cat
WHERE events.dateid=dates.dateid AND events.dateid=sale.dateid AND events.catid =
  cat.catid
ORDER BY 2 DESC LIMIT 10;
```

```
+-----+-----+-----+
| dateid | commission | catname |
+-----+-----+-----+
| 1880 | 1893.6 | Pop |
| 1880 | 1893.6 | Opera |
| 1880 | 1893.6 | Plays |
| 1880 | 1893.6 | Musicals |
```

```

| 1861 | 1500 | Plays |
| 2003 | 1500 | Pop |
| 1861 | 1500 | Opera |
| 2003 | 1500 | Plays |
| 1861 | 1500 | Musicals |
| 1861 | 1500 | Pop |
+-----+-----+-----+

```

3. 若要從銷售結構描述中的事件資料表中選取 10 個事件，請使用下列範例。

```
SELECT * FROM sales.events LIMIT 10;
```

```

+-----+-----+-----+-----+-----+-----+-----+
| eventid | venueid | catid | dateid | eventname | starttime |
+-----+-----+-----+-----+-----+-----+-----+
| 4836 | 73 | 9 | 1871 | Soulfest | 2008-02-14 19:30:00 |
| 5739 | 41 | 9 | 1871 | Fab Faux | 2008-02-14 19:30:00 |
| 627 | 229 | 6 | 1872 | High Society | 2008-02-15 14:00:00 |
| 2563 | 246 | 7 | 1872 | Hamlet | 2008-02-15 20:00:00 |
| 7703 | 78 | 9 | 1872 | Feist | 2008-02-15 14:00:00 |
| 7903 | 90 | 9 | 1872 | Little Big Town | 2008-02-15 19:30:00 |
| 7925 | 101 | 9 | 1872 | Spoon | 2008-02-15 19:00:00 |
| 8113 | 17 | 9 | 1872 | Santana | 2008-02-15 15:00:00 |
| 463 | 303 | 8 | 1873 | Tristan und Isolde | 2008-02-16 19:00:00 |
| 613 | 236 | 6 | 1873 | Pal Joey | 2008-02-16 15:00:00 |
+-----+-----+-----+-----+-----+-----+-----+

```

4. 若要嘗試更新事件 1 的事件名稱，請執行下列範例。此範例將導致許可遭拒錯誤，因為 Salesanalyst 使用者只有銷售結構描述中事件資料表的SELECT許可。若要更新事件資料表，您必須將 sales_ro 角色許可授予 UPDATE。如需授予更新資料表許可的詳細資訊，請參閱的 UPDATE 參數[GRANT](#)。如需 UPDATE 命令的詳細資訊，請參閱 [UPDATE](#)。

```

UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;

```

```
ERROR: permission denied for relation events
```

5. 若要嘗試從行銷結構描述中的事件資料表中選取全部，請使用下列範例。此範例會導致許可遭拒錯誤，因為 Salesanalyst 使用者只有銷售結構描述中事件資料表的SELECT許可。若要從行銷結構描述中的事件資料表中選取資料，您必須在行銷結構描述中的事件資料表上授予 sales_ro 角色 SELECT許可。

```
SELECT * FROM marketing.events;
```

```
ERROR: permission denied for schema marketing
```

步驟 5：建立讀寫使用者

在此步驟中，負責在銷售結構描述中建置擷取、轉換和載入（ETL）管道以進行資料處理的銷售工程師將擁有唯讀存取權，但稍後將擁有讀取和寫入存取權來執行其任務。

1. 以 dbadmin 使用者身分連線至資料庫。
2. 若要在銷售結構描述中建立 sales_rw 角色，請使用下列範例。

```
CREATE ROLE sales_rw;
```

3. 若要建立 Salesengineer 使用者，請使用下列範例。

```
CREATE USER salesengineer PASSWORD 'Test12345';
```

4. 若要授予 sales_rw 角色用量，並藉由指派 sales_ro 角色給銷售結構描述的物件來選取存取權，請使用下列範例。如需角色如何在 Amazon Redshift 中繼承許可的詳細資訊，請參閱 [角色階層](#)。

```
GRANT ROLE sales_ro TO ROLE sales_rw;
```

5. 若要將 sales_rw 角色指派給 Salesengineer 使用者，請使用下列範例。

```
GRANT ROLE sales_rw TO salesengineer;
```

步驟 6：以具有繼承唯讀角色的使用者身分查詢資料

在此步驟中，Salesengineer 使用者會嘗試在獲得讀取許可之前更新事件資料表。

1. 以 Salesengineer 使用者身分連線至資料庫。
2. Salesengineer 使用者可以成功從銷售結構描述的事件資料表讀取資料。若要從銷售結構描述中的事件資料表中選取事件 1 的事件，請使用下列範例。

```
SELECT * FROM sales.events where eventid=1;
```

```
+-----+-----+-----+-----+-----+-----+
| eventid | venueid | catid | dateid | eventname | starttime |
+-----+-----+-----+-----+-----+-----+
|      1 |     305 |     8 |   1851 | Gotterdammerung | 2008-01-25 14:30:00 |
+-----+-----+-----+-----+-----+-----+
```

3. 若要嘗試從行銷結構描述中的事件資料表中選取全部，請使用下列範例。salesengineer 使用者沒有行銷結構描述中資料表的許可，因此此查詢會導致許可遭拒的錯誤。若要從行銷結構描述中的事件資料表中選取資料，您必須在行銷結構描述中的事件資料表上授予 sales_rw 角色SELECT許可。

```
SELECT * FROM marketing.events;
```

```
ERROR: permission denied for schema marketing
```

4. 若要嘗試更新事件 1 的事件名稱，請執行下列範例。此範例將導致許可遭拒錯誤，因為 Salesengineer 使用者在銷售結構描述的事件資料表上只有選取許可。若要更新事件資料表，您必須將 sales_rw 角色許可授予 UPDATE。

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;
```

```
ERROR: permission denied for relation events
```

步驟 7：將更新和插入許可授予讀寫角色

在此步驟中，您將更新和插入許可授予 sales_rw 角色。

1. 以 dbadmin 使用者身分連線至資料庫。
2. 若要將 UPDATE、INSERT和 DELETE許可授予 sales_rw 角色，請使用下列範例。

```
GRANT UPDATE, INSERT, ON ALL TABLES IN SCHEMA sales TO role sales_rw;
```

步驟 8：以讀寫使用者身分查詢資料

在此步驟中，Salesengineer 會在授予其角色插入和更新許可後成功更新資料表。接下來，銷售工程師會嘗試分析和清空事件資料表，但無法這麼做。

1. 以 Salesengineer 使用者身分連線至資料庫。

2. 若要更新 eventid 1 的事件名稱，請執行下列範例。

```
UPDATE sales.events
SET eventname = 'Comment event'
WHERE eventid = 1;
```

3. 若要檢視在上一個查詢中所做的變更，請使用下列範例，從銷售結構描述中的事件資料表中選取事件 1 的事件。

```
SELECT * FROM sales.events WHERE eventid=1;
```

```
+-----+-----+-----+-----+-----+-----+
| eventid | venueid | catid | dateid | eventname | starttime |
+-----+-----+-----+-----+-----+-----+
|      1 |      305 |      8 |   1851 | Comment event | 2008-01-25 14:30:00 |
+-----+-----+-----+-----+-----+-----+
```

4. 若要分析銷售結構描述中的更新事件表，請使用下列範例。此範例會導致許可遭拒錯誤，因為 Salesengineer 使用者沒有必要的許可，而且不是銷售結構描述中事件資料表的擁有者。若要分析事件資料表，您必須使用 GRANT 命令 ANALYZE 將 sales_rw 角色許可授予。如需 ANALYZE 命令的詳細資訊，請參閱 [ANALYZE](#)。

```
ANALYZE sales.events;
```

```
ERROR: skipping "events" --- only table or database owner can analyze
```

5. 若要清空更新的事件資料表，請使用下列範例。此範例會導致許可遭拒錯誤，因為 Salesengineer 使用者沒有必要的許可，而且不是銷售結構描述中事件資料表的擁有者。若要清空事件資料表，您必須使用 GRANT 命令授予 sales_rw VACUUM 角色許可。如需 VACUUM 命令的詳細資訊，請參閱 [VACUUM](#)。

```
VACUUM sales.events;
```

```
ERROR: skipping "events" --- only table or database owner can vacuum it
```

步驟 9：以管理員使用者身分分析和清空資料庫中的資料表

在此步驟中，dbadmin 使用者會分析和清空所有資料表。使用者在此資料庫具有管理員許可，因此可以執行這些命令。

1. 以 dbadmin 使用者身分連線至資料庫。
2. 若要分析銷售結構描述中的事件資料表，請使用下列範例。

```
ANALYZE sales.events;
```

3. 若要清空銷售結構描述中的事件資料表，請使用下列範例。

```
VACUUM sales.events;
```

4. 若要分析行銷結構描述中的事件資料表，請使用下列範例。

```
ANALYZE marketing.events;
```

5. 若要清空行銷結構描述中的事件資料表，請使用下列範例。

```
VACUUM marketing.events;
```

步驟 10：以讀寫使用者身分截斷資料表

在此步驟中，銷售工程師使用者會嘗試截斷銷售結構描述中的事件資料表，但只有在 dbadmin 使用者授予截斷許可時才會成功。

1. 以 Salesengineer 使用者身分連線至資料庫。
2. 若要嘗試從銷售結構描述中的事件資料表中刪除所有資料列，請使用下列範例。此範例會導致錯誤，因為 Salesengineer 使用者沒有必要的許可，而且不是銷售結構描述中事件資料表的擁有者。若要截斷事件資料表，您必須使用 GRANT 命令授予 sales_rw TRUNCATE 角色許可。如需 TRUNCATE 命令的詳細資訊，請參閱 [TRUNCATE](#)。

```
TRUNCATE sales.events;
```

```
ERROR: must be owner of relation events
```

3. 以 dbadmin 使用者身分連線至資料庫。
4. 若要將截斷的資料表權限授予 sales_rw 角色，請使用下列範例。

```
GRANT TRUNCATE TABLE TO role sales_rw;
```

5. 使用查詢編輯器 v2 以 Salesengineer 使用者身分連線至資料庫。

6. 若要從銷售結構描述中的事件資料表讀取前 10 個事件，請使用下列範例。

```
SELECT * FROM sales.events ORDER BY eventid LIMIT 10;
```

```
+-----+-----+-----+-----+-----+
+-----+
| eventid | venueid | catid | dateid |          eventname          |          starttime          |
|         |         |      |        |                             |                             |
+-----+-----+-----+-----+-----+
+-----+
|         1 |      305 |      8 |   1851 | Comment event              | 2008-01-25
14:30:00 |
|         2 |      306 |      8 |   2114 | Boris Godunov              | 2008-10-15
20:00:00 |
|         3 |      302 |      8 |   1935 | Salome                      | 2008-04-19
14:30:00 |
|         4 |      309 |      8 |   2090 | La Cenerentola (Cinderella) | 2008-09-21
14:30:00 |
|         5 |      302 |      8 |   1982 | Il Trovatore                | 2008-06-05
19:00:00 |
|         6 |      308 |      8 |   2109 | L Elisir d Amore            | 2008-10-10
19:30:00 |
|         7 |      309 |      8 |   1891 | Doctor Atomic               | 2008-03-06
14:00:00 |
|         8 |      302 |      8 |   1832 | The Magic Flute              | 2008-01-06
20:00:00 |
|         9 |      308 |      8 |   2087 | The Fly                      | 2008-09-18
19:30:00 |
|        10 |      305 |      8 |   2079 | Rigoletto                   | 2008-09-10
15:00:00 |
+-----+-----+-----+-----+-----+
+-----+
```

7. 若要截斷銷售結構描述中的事件資料表，請使用下列範例。

```
TRUNCATE sales.events;
```

8. 若要從銷售結構描述中的更新事件資料表讀取資料，請使用下列範例。

```
SELECT * FROM sales.events ORDER BY eventid LIMIT 10;
```

```
+-----+-----+-----+-----+-----+
+-----+
```

```
| eventid | venueid | catid | dateid |          eventname          |          starttime          |
|
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```

建立行銷結構描述的唯一讀和讀寫角色 (選用)

在此步驟中，您會為行銷結構描述建立唯一讀和讀寫角色。

1. 以 dbadmin 使用者身分連線至資料庫。
2. 若要為行銷結構描述建立唯一讀和讀寫角色，請使用下列範例。

```
CREATE ROLE marketing_ro;

CREATE ROLE marketing_rw;

GRANT USAGE ON SCHEMA marketing TO ROLE marketing_ro, ROLE marketing_rw;

GRANT SELECT ON ALL TABLES IN SCHEMA marketing TO ROLE marketing_ro;

GRANT ROLE marketing_ro TO ROLE marketing_rw;

GRANT INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA marketing TO ROLE marketing_rw;

CREATE USER marketinganalyst PASSWORD 'Test12345';

CREATE USER marketingengineer PASSWORD 'Test12345';

GRANT ROLE marketing_ro TO marketinganalyst;

GRANT ROLE marketing_rw TO marketingengineer;
```

的系統函數 RBAC (選用)

Amazon Redshift 有兩個函數，可提供有關其他群組或角色的使用者成員資格和角色成員資格的系統資訊：`role_is_member_of` 和 `user_is_member_of`。這些函數可供超級使用者和一般使用者使用。超級使用者可以檢查所有角色成員資格。一般使用者只能檢查其已獲授予存取權之角色的成員資格。

使用 `role_is_member_of` 函數

1. 以 Salesengineer 使用者身分連線至資料庫。
2. 若要檢查 sales_rw 角色是否為 sales_ro 角色的成員，請使用下列範例。

```
SELECT role_is_member_of('sales_rw', 'sales_ro');
```

```
+-----+
| role_is_member_of |
+-----+
| true              |
+-----+
```

3. 若要檢查 sales_ro 角色是否為 sales_rw 角色的成員，請使用下列範例。

```
SELECT role_is_member_of('sales_ro', 'sales_rw');
```

```
+-----+
| role_is_member_of |
+-----+
| false             |
+-----+
```

使用 user_is_member_of 函數

1. 以 Salesengineer 使用者身分連線至資料庫。
2. 下列範例會嘗試檢查 Salesanalyst 使用者的使用者成員資格。此查詢會導致錯誤，因為 Salesengineer 無法存取 salesanalyst。若要成功執行此命令，請以 Salesanalyst 使用者身分連線至資料庫，並使用 範例。

```
SELECT user_is_member_of('salesanalyst', 'sales_ro');
```

```
ERROR
```

3. 以超級使用者身分連線至資料庫。
4. 若要在以超級使用者身分連線時檢查 Salesanalyst 使用者的成員資格，請使用下列範例。

```
SELECT user_is_member_of('salesanalyst', 'sales_ro');
```

```
+-----+
| user_is_member_of |
+-----+
```

```
| true          |
+-----+
```

5. 以 dbadmin 使用者身分連線至資料庫。
6. 若要檢查 Salesengineer 使用者的成員資格，請使用下列範例。

```
SELECT user_is_member_of('salesengineer', 'sales_ro');

+-----+
| user_is_member_of |
+-----+
| true              |
+-----+

SELECT user_is_member_of('salesengineer', 'marketing_ro');

+-----+
| user_is_member_of |
+-----+
| false             |
+-----+

SELECT user_is_member_of('marketinganalyst', 'sales_ro');

+-----+
| user_is_member_of |
+-----+
| false             |
+-----+
```

的系統檢視 RBAC (選用)

若要透過角色檢視角色、將角色指派給使用者、角色階層和資料庫物件的權限，請使用 Amazon Redshift 的系統檢視。這些檢視可供超級使用者和一般使用者使用。超級使用者可以檢查所有角色詳細資訊。一般使用者只能檢查其已獲授予存取權之角色的詳細資訊。

1. 若要檢視叢集中明確授予角色的使用者清單，請使用下列範例。

```
SELECT * FROM svv_user_grants;
```

2. 若要檢視叢集中明確授予角色的角色清單，請使用下列範例。

```
SELECT * FROM svv_role_grants;
```

如需系統檢視的完整清單，請參閱 [SVV 中繼資料檢視](#)。

搭配 使用資料列層級安全性 RBAC (選用)

若要對敏感資料進行精細存取控制，請使用資料列層級安全性 (RLS)。如需的詳細資訊RLS，請參閱 [資料列層級安全性](#)。

在本節中，您可以建立RLS政策，讓使用者salesengineer只能檢視cat資料表中具有 Major League Baseball catdesc值的資料列。然後，您以salesengineer使用者身分查詢資料庫。

1. 以salesengineer使用者身分連線至資料庫。
2. 若要檢視cat資料表中的前 5 個項目，請使用下列範例。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;
```

catid	catgroup	catname	catdesc
1	Sports	MLB	Major League Baseball
2	Sports	NHL	National Hockey League
3	Sports	NFL	National Football League
4	Sports	NBA	National Basketball Association
5	Sports	MLS	Major League Soccer

3. 以dbadmin使用者身分連線至資料庫。
4. 若要為cat資料表中的資料catdesc欄建立RLS政策，請使用下列範例。

```
CREATE RLS POLICY policy_mlb_engineer
WITH (catdesc VARCHAR(50))
USING (catdesc = 'Major League Baseball');
```

5. 若要將RLS政策連接至sales_rw角色，請使用下列範例。

```
ATTACH RLS POLICY policy_mlb_engineer ON sales.cat TO ROLE sales_rw;
```

6. 若要變更資料表以開啟 RLS，請使用下列範例。

```
ALTER TABLE sales.cat ROW LEVEL SECURITY ON;
```

7. 以salesengineer使用者身分連線至資料庫。

8. 若要嘗試檢視cat資料表中的前 5 個項目，請使用下列範例。請注意，只有在資料catdesc欄為時，才會顯示項目Major League Baseball。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
|      1 | Sports   | MLB     | Major League Baseball |
+-----+-----+-----+-----+
```

9. 以salesanalyst使用者身分連線至資料庫。

10. 若要嘗試檢視cat資料表中的前 5 個項目，請使用下列範例。請注意，不會顯示任何項目，因為已套用預設拒絕所有政策。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
```

11. 以dbadmin使用者身分連線至資料庫。

12. 若要授予sales_ro角色IGNORERLS許可，請使用下列範例。這會授予salesanalyst使用者忽略RLS政策的許可，因為它們是sales_ro角色的成員。

```
GRANT IGNORE RLS TO ROLE sales_ro;
```

13以salesanalyst使用者身分連線至資料庫。

14若要檢視cat資料表中的前 5 個項目，請使用下列範例。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
|     1 | Sports   | MLB     | Major League Baseball    |
|     2 | Sports   | NHL     | National Hockey League    |
|     3 | Sports   | NFL     | National Football League  |
|     4 | Sports   | NBA     | National Basketball Association |
|     5 | Sports   | MLS     | Major League Soccer       |
+-----+-----+-----+-----+
```

15以dbadmin使用者身分連線至資料庫。

16若要撤銷sales_ro角色的IGNORERLS許可，請使用下列範例。

```
REVOKE IGNORE RLS FROM ROLE sales_ro;
```

17以salesanalyst使用者身分連線至資料庫。

18若要嘗試檢視cat資料表中的前 5 個項目，請使用下列範例。請注意，不會顯示任何項目，因為已套用預設拒絕所有政策。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
```

19以dbadmin使用者身分連線至資料庫。

20若要從cat資料表分離RLS政策，請使用下列範例。

```
DETACH RLS POLICY policy_mlb_engineer ON cat FROM ROLE sales_rw;
```

21以salesanalyst使用者身分連線至資料庫。

22若要嘗試檢視cat資料表中的前 5 個項目，請使用下列範例。請注意，不會顯示任何項目，因為已套用預設拒絕所有政策。

```
SELECT *
FROM sales.cat
ORDER BY catid ASC
LIMIT 5;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
|      1 | Sports   | MLB     | Major League Baseball    |
|      2 | Sports   | NHL     | National Hockey League    |
|      3 | Sports   | NFL     | National Football League  |
|      4 | Sports   | NBA     | National Basketball Association |
|      5 | Sports   | MLS     | Major League Soccer       |
+-----+-----+-----+-----+
```

23以dbadmin使用者身分連線至資料庫。

24若要捨棄RLS政策，請使用下列範例。

```
DROP RLS POLICY policy_mlb_engineer;
```

25若要移除 RLS，請使用下列範例。

```
ALTER TABLE cat ROW LEVEL SECURITY OFF;
```

相關主題

如需的詳細資訊RBAC，請參閱下列文件：

- [角色階層](#)
- [角色指派](#)
- [資料庫物件許可](#)
- [ALTER DEFAULT PRIVILEGES 適用於 RBAC](#)

資料列層級安全性

在 Amazon Redshift 中使用資料列層級安全性 (RLS)，您可以對敏感資料進行精細的存取控制。您可以根據資料庫物件層級定義的安全政策，決定哪些使用者或角色可以存取結構描述或資料表內的特定資料記錄。除了資料欄層級的安全性之外，您還可以將使用者許可授予資料欄子集，並使用 RLS 政策進一步限制對可見資料欄特定資料列的存取。如需資料欄層級安全的相關資訊，請參閱 [欄位層級存取控制的使用須知](#)。

當您在資料表上執行 RLS 政策時，您可以在使用者執行查詢時限制傳回的結果集。

建立 RLS 政策時，您可以指定運算式，以決定 Amazon Redshift 是否傳回查詢中資料表中的任何現有資料列。透過建立 RLS 政策來限制存取，您不必在查詢中新增或外部化其他條件。

建立 RLS 政策時，建議您建立簡單的政策，並避免政策中的複雜陳述式。定義 RLS 政策時，請勿在以政策為基礎的政策定義中使用過多的資料表聯結。

當政策參照查閱資料表時，除了政策所在的資料表之外，Amazon Redshift 還會掃描其他表格。對於已連接 RLS 政策的使用者，以及未連接任何政策的使用者，相同的查詢之間會有效能差異。

在 SQL 陳述式中使用 RLS 政策

在 SQL 陳述式中使用 RLS 政策時，Amazon Redshift 會套用下列規則：

- Amazon Redshift 預設會將 RLS 政策套用至 SELECT、UPDATE 和 DELETE 陳述式。
- 對於 SELECT 和 UNLOAD，Amazon Redshift 會根據您定義的政策篩選資料列。
- 對於 UPDATE，Amazon Redshift 只會更新您可見的資料列。如果政策限制了資料表中的資料列子集，您就無法將其更新。
- 對於 DELETE，您只能刪除您可見的資料列。如果政策限制了資料表中的資料列子集，您就無法將其刪除。對於 TRUNCATE，您仍然可以截斷資料表。
- 對於 CREATE TABLE LIKE，使用 LIKE 選項建立的資料表不會從來源資料表繼承許可設定。同樣地，目標資料表不會從來源資料表繼承 RLS 政策。

每個使用者結合多個政策

RLS 在 Amazon Redshift 中，支援為每個使用者和物件連接多個政策。當為使用者定義多個政策時，Amazon Redshift 會根據資料表 RLSCONJUNCTIONTYPE 的設定，使用 AND 或 OR 語法套用所有政策。如需結合類型的更多相關資訊，請參閱 [ALTER TABLE](#)。

資料表上的多個政策都可以與您建立關聯。您可以直接附加多個政策，或者您屬於多個角色，而這些角色附加了不同的政策。

當多個政策應該限制指定關係中的資料列存取時，您可以將RLSCONJUNCTIONTYPE關係設定為AND。請考量下列範例。Alice 只能看到具有的 "catname" NBA作為指定政策的運動事件。

```
-- Create an analyst role and grant it to a user named Alice.
CREATE ROLE analyst;
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
GRANT ROLE analyst TO alice;

-- Create an RLS policy that only lets the user see sports.
CREATE RLS POLICY policy_sports
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Sports');

-- Create an RLS policy that only lets the user see NBA.
CREATE RLS POLICY policy_nba
WITH (catname VARCHAR(10))
USING (catname = 'NBA');

-- Attach both to the analyst role.
ATTACH RLS POLICY policy_sports ON category TO ROLE analyst;
ATTACH RLS POLICY policy_nba ON category TO ROLE analyst;

-- Activate RLS on the category table with AND CONJUNCTION TYPE.
ALTER TABLE category ROW LEVEL SECURITY ON CONJUNCTION TYPE AND;

-- Change session to Alice.
SET SESSION AUTHORIZATION alice;

-- Select all from the category table.
SELECT catgroup, catname
FROM category;
```

catgroup	catname
Sports	NBA

(1 row)

當多個政策應允許使用者在指定關係中看到更多資料列時，使用者可以設定與 OR RLS CONJUNCTION TYPE 的關係。請考量下列範例。Alice 只能看到「音樂會」和「體育」做為指定政策。

```
-- Create an analyst role and grant it to a user named Alice.
CREATE ROLE analyst;
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
GRANT ROLE analyst TO alice;

-- Create an RLS policy that only lets the user see concerts.
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');

-- Create an RLS policy that only lets the user see sports.
CREATE RLS POLICY policy_sports
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Sports');

-- Attach both to the analyst role.
ATTACH RLS POLICY policy_concerts ON category TO ROLE analyst;
ATTACH RLS POLICY policy_sports ON category TO ROLE analyst;

-- Activate RLS on the category table with OR CONJUNCTION TYPE.
ALTER TABLE category ROW LEVEL SECURITY ON CONJUNCTION TYPE OR;

-- Change session to Alice.
SET SESSION AUTHORIZATION alice;

-- Select all from the category table.
SELECT catgroup, count(*)
FROM category
GROUP BY catgroup ORDER BY catgroup;

  catgroup | count
-----+-----
  Concerts |    3
   Sports  |    5
(2 rows)
```

RLS 政策所有權和管理

身為超級使用者、安全管理員或具有 `sys : secadmin` 角色的使用者，您可以建立、修改或管理資料表的所有RLS政策。在物件層級上，您可以開啟或關閉資料列層級安全，而不必修改資料表的結構描述定義。

若要開始使用資料列層級安全性，以下是您可以使用的SQL陳述式：

- 使用 `ALTER TABLE` 陳述式來開啟或關閉RLS資料表。如需詳細資訊，請參閱[ALTER TABLE](#)。
- 使用 `CREATE RLS POLICY` 陳述式為一或多個資料表建立安全政策，並在政策中指定一或多個使用者或角色。

如需詳細資訊，請參閱[CREATE RLS POLICY](#)。

- 使用 `ALTER RLS POLICY` 陳述式來變更政策，例如變更政策定義。您可以針對多個資料表或檢視使用相同的政策。

如需詳細資訊，請參閱[ALTER RLS POLICY](#)。

- 使用 `ATTACH RLS POLICY` 陳述式將政策連接至一或多個關係、一或多個使用者或角色。

如需詳細資訊，請參閱[ATTACH RLS POLICY](#)。

- 使用 `DETACH RLS POLICY` 陳述式將政策與一或多個關係、一或多個使用者或角色分離。

如需詳細資訊，請參閱[DETACH RLS POLICY](#)。

- 使用 `DROP RLS POLICY` 陳述式捨棄政策。

如需詳細資訊，請參閱[DROP RLS POLICY](#)。

- 使用 `GRANT` 和 `REVOKE` 陳述式明確授予和撤銷參考查詢資料表之RLS政策的SELECT許可。如需詳細資訊，請參閱 [GRANT](#) 和 [REVOKE](#)。

若要監控所建立的證測，`sys:secadmin` 可以檢視 [SVV_RLS_POLICY](#) 和 [SVV_RLS_ATTACHED_POLICY](#)。

若要列出 RLS受保護的關係，`sys : secadmin` 可以檢視 `SVV_RLS_RELATION`。

若要追蹤參考RLS受保護關係之查詢上的RLS政策應用程式，超級使用者、`sys : operator` 或任何具有系統許可的使用者`ACCESSSYSTEMTABLE`都可以檢視 [SVV_RLS_APPLIED_POLICY](#)。請注意，在預設情況下，`sys:secadmin` 不會授予這些許可。

若要查詢具有連接RLS政策的資料表，但未看到它們，您可以將許可授予`IGNORERLS`任何使用者。超級使用者或 `sys : secadmin` 的使用者會自動獲得許可 `IGNORE RLS`。如需詳細資訊，請參閱[GRANT](#)。

若要解釋`EXPLAIN`計劃中查詢RLS的政策篩選條件，以疑難排解 RLS相關查詢，您可以將許可授予`EXPLAINRLS`任何使用者。如需詳細資訊，請參閱 [GRANT](#) 和 [EXPLAIN](#)。

政策相依物件和原則

為了為應用程式提供安全性，並防止政策物件變得過時或無效，Amazon Redshift 不允許捨棄或變更 RLS 政策參考的物件。

下列列出 Amazon Redshift 追蹤 RLS 的政策結構描述物件相依性。

- 追蹤目標資料表的結構描述物件相依性時，Amazon Redshift 會遵循下列規則：
 - 當您捨棄目標資料表時，Amazon Redshift 會將政策與關係、使用者、角色或公開分離。
 - 當您重新命名目標資料表名稱時，附加的政策不會受到影響。
 - 如果您先將政策捨棄或分離，您只能捨棄政策定義中參照的目標資料表資料欄。當指定 CASCADE 選項時，這也適用。您可以捨棄目標資料表中的其他資料欄。
 - 您無法重命名目標資料表的參照資料欄。若要重新命名參照的資料欄，請先將政策分離。當指定 CASCADE 選項時，這也適用。
 - 即使您指定 CASCADE 選項，也無法變更所參考資料欄的類型。
- 追蹤查閱資料表的結構描述物件相依性時，Amazon Redshift 會遵循下列規則：
 - 您無法捨棄查閱資料表。若要捨棄查閱資料表，請先捨棄參照查詢資料表的政策。
 - 您無法重新命名查閱資料表。若要重新命名查閱資料表，請先捨棄參照查詢資料表的政策。當指定 CASCADE 選項時，這也適用。
 - 您無法捨棄政策定義中使用的查閱資料表資料欄。若要捨棄政策定義中使用的查閱資料表資料欄，請先捨棄參照查詢資料表的政策。當 ALTER TABLE DROP COLUMN 陳述式中指定 CASCADE 選項時，這也適用。您可以捨棄查閱資料表中的其他資料欄。
 - 您無法重命名查閱資料表的參照資料欄。若要重新命名參照的資料欄，請先捨棄參照查詢資料表的政策。當指定 CASCADE 選項時，這也適用。
 - 您無法變更參照資料欄的類型。
- 捨棄使用者或角色時，Amazon Redshift 會自動分離附加到該使用者或角色的所有政策。
- 當您在 DROP SCHEMA 陳述式中使用 CASCADE 選項時，Amazon Redshift 也會捨棄結構描述中的關聯。還會捨棄任何其他結構描述中依賴於已捨棄結構描述中關係的關係。對於政策中查詢資料表的關係，Amazon Redshift 會失敗 DROP SCHEMA DDL。對於陳述 DROPSHEMA 式捨棄的任何關係，Amazon Redshift 會分離連接到這些關係的所有政策。
- 您只能在捨棄政策時捨棄查詢函數 (政策定義中參照的函數)。當指定 CASCADE 選項時，這也適用。
- 將政策附加至資料表時，Amazon Redshift 會檢查此資料表是否為不同政策中的查閱資料表。如果是這種情況，Amazon Redshift 將不允許在此資料表中附加政策。

- 建立RLS政策時，Amazon Redshift 會檢查此資料表是否為任何其他RLS政策的目標資料表。如果是這種情況，Amazon Redshift 將不允許在此資料表上建立政策。

範例

下列範例說明如何追蹤結構描述相依性。

```
-- The CREATE and ATTACH policy statements for `policy_events` references some
-- target and lookup tables.
-- Target tables are tickit_event_redshift and target_schema.target_event_table.
-- Lookup table is tickit_sales_redshift.
-- Policy `policy_events` has following dependencies:
-- table tickit_sales_redshift column eventid, qtysold
-- table tickit_event_redshift column eventid
-- table target_event_table column eventid
-- schema public and target_schema
CREATE RLS POLICY policy_events
WITH (eventid INTEGER)
USING (
    eventid IN (SELECT eventid FROM tickit_sales_redshift WHERE qtysold <3)
);

ATTACH RLS POLICY policy_events ON tickit_event_redshift TO ROLE analyst;

ATTACH RLS POLICY policy_events ON target_schema.target_event_table TO ROLE consumer;
```

使用RLS政策的考量和限制

考量事項

以下是使用RLS政策的考量事項：

- Amazon Redshift 會將RLS政策套用至 SELECT、UPDATE或 DELETE陳述式。
- Amazon Redshift 不會將RLS政策套用至 INSERT、COPY、ALTERTABLEAPPEND陳述式。
- 資料列層級安全可搭配資料欄層級安全來保護您的資料。
- 當您的 Amazon Redshift 叢集位於支援 的最新一般可用版本RLS，但降級至較早版本時，當您在已連接RLS政策的基礎資料表上執行查詢時，Amazon Redshift 會傳回錯誤。sys : secadmin 可以撤銷被授予受限制政策的使用者存取權、關閉RLS資料表，以及捨棄政策。
- 當 為來源關係RLS開啟時，Amazon Redshift 支援超級使用者、明確授予系統許可的使用者RLS 或 sys : secadmin IGNORE 角色的ALTERTABLEAPPEND陳述式。在此情況下，您可以執行

ALERTABLEAPPEND陳述式，透過從現有來源資料表移動資料，將資料列附加到目標資料表。Amazon Redshift 會將所有元組從來源關係移動到目標關係。目標關係RLS的狀態不會影響ALERTABLEAPPEND陳述式。

- 若要協助從其他資料倉儲系統移轉，您可以指定變數名稱和值，以設定和擷取用於連線的自訂工作階段內容變數。

下列範例會設定資料列層級安全（RLS）政策的工作階段內容變數。

```
-- Set a customized context variable.
SELECT set_config('app.category', 'Concerts', FALSE);

-- Create a RLS policy using current_setting() to get the value of a customized
  context variable.
CREATE RLS POLICY policy_categories
WITH (catgroup VARCHAR(10))
USING (catgroup = current_setting('app.category', FALSE));

-- Set correct roles and attach the policy on the target table to one or more roles.
ATTACH RLS POLICY policy_categories ON tickit_category_redshift TO ROLE analyst, ROLE
  dbadmin;
```

如需如何設定和擷取自訂工作階段內容變數的詳細資訊，請前往 [SET](#)、[SET_CONFIG](#)、[SHOW](#)、[CURRENT_SETTING](#)和 [RESET](#)。如需修改伺服器組態的一般詳細資訊，請前往 [修改伺服器組態](#)。

Important

在RLS政策中使用工作階段內容變數時，安全政策依賴於叫用政策的使用者或角色。在RLS政策中使用工作階段內容變數時，請小心避免安全漏洞。

- 在後續FETCH陳述式SETSESSIONAUTHORIZATION之間DECLAREFETCH或在之間使用變更工作階段使用者，不會根據DECLARE當時的使用者政策重新整理已預備的計劃。當游標與RLS受保護的資料表搭配使用時，避免變更工作階段使用者。
- 當檢視物件內的基礎物件受到RLS保護時，連接到執行查詢之使用者的政策會套用至個別的基礎物件。這與物件層級許可檢查不同，檢視擁有者的許可會根據檢視基礎物件進行檢查。您可以在其EXPLAIN計劃輸出中檢視查詢的RLS受保護關係。
- 在連接到使用者之關係RLS的政策中參考使用者定義函數（UDF）時，使用者必須擁有的EXECUTE許可UDF才能查詢關係。

- 資料列層級安全性可能會限制最佳查詢效果。建議您在大型資料集上部署 保護RLS的檢視之前，先仔細評估查詢效能。
- 套用至最新繫結視觀表的資料列層級安全政策，可能會推入聯合資料表。這些RLS政策可能會在外部處理引擎日誌中顯示。

限制

以下是使用 RLS 政策時的限制：

- Amazon Redshift 支援某些政策的SELECT陳述式，這些RLS政策具有複雜的聯結，但不支援 UPDATE或DELETE陳述式。如果使用 UPDATE或 DELETE陳述式，Amazon Redshift 會傳回下列錯誤：

```
ERROR: One of the RLS policies on target relation is not supported in UPDATE/DELETE.
```

- 每當使用者定義函數（UDF）在連接到使用者之關係RLS的政策中參考時，使用者必須擁有的 EXECUTE 許可UDF，才能查詢該關係。
- 不支援相關子查詢。Amazon Redshift 會傳回以下錯誤：

```
ERROR: RLS policy could not be rewritten.
```

- RLS 政策無法連接至外部資料表。
- Amazon Redshift 不支援與 進行資料共用RLS。如果關係尚未針對資料共用RLS關閉，則取用者叢集上的查詢會失敗，並出現下列錯誤：

```
RLS-protected relation "rls_protected_table" cannot be accessed via datasharing query.
```

您可以使用 `ALTER TABLE` 命令搭配 參數 `RLS` 來關閉資料共用

`ROWLEVELSECURITYOFFFOR DATASHARES`。如需使用 `ALTER TABLE` 啟用或停用的詳細資訊 RLS，請前往 [ALTER TABLE](#)。

- 在跨資料庫查詢中，Amazon Redshift 區塊會讀取到 RLS受保護的關係。具有 `IGNORERLS`許可的使用者可以使用跨資料庫查詢來存取受保護的關係。當沒有 `IGNORE RLS` 許可的使用者透過跨資料庫查詢存取 RLS受保護的關係時，會出現下列錯誤：

```
RLS-protected relation "rls_protected_table" cannot be accessed via cross-database query.
```


- ALTER RLS POLICY 僅支援使用 USING (using_predicate_exp) 子句修改RLS政策。執行 ALTER RLS 時，您無法修改具有 WITH 子句RLS的政策POLICY。
- 如果下列任一組態選項的值不符合工作階段的預設值，您就無法查詢已開啟資料列層級安全的關係：
 - enable_case_sensitive_super_attribute
 - enable_case_sensitive_identifier
 - downcase_delimited_identifier

如果您嘗試在 上查詢與資料列層級安全性的關係，並看到訊息「RLS受保護關係不支援案例敏感度與其預設值不同的工作階段層級組態」，請考慮重設工作階段的組態選項。

- 當您佈建的叢集或無伺服器命名空間具有任何資料列層級安全性政策時，一般使用者會無法使用下列命令：

```
ALTER <current_user> SET enable_case_sensitive_super_attribute/  
enable_case_sensitive_identifier/downcase_delimited_identifier
```

建立RLS政策時，建議您變更一般使用者的預設組態選項設定，以符合政策建立時的工作階段組態選項設定。具有 ALTERUSER權限的超級使用者和使用者可以使用參數群組設定或 ALTERUSER命令來執行此操作。如需有關參數群組的詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 參數群組](#)。如需有關 ALTERUSER命令的資訊，請參閱 [ALTER USER](#)。

- 使用 [CREATE VIEW](#) 命令的一般使用者，無法取代具有資料列層級安全政策的視觀表和近期繫結視觀表。若要將檢視或取代LBVs為RLS政策，請先分離連接到它們的任何RLS政策，取代檢視或LBVs，然後重新連接政策。具有的超級使用者和使用者可以CREATEVIEW在檢視上使用，或LBVs搭配 RLS政策sys:secadmin permission使用，而無需分離政策。
- 具有資料列層級安全政策的視觀表無法參考系統資料表和系統視觀表。
- 無法RLS保護一般檢視所參考的繫結後期檢視。
- RLS無法在相同的查詢中存取來自資料湖的受保護關係和巢狀資料。

RLS 效能的最佳實務

以下是最佳實務，以確保 Amazon Redshift 在 保護的資料表上具有更好的效能RLS。

運算子和函數的安全性

查詢RLS受保護的資料表時，某些運算子或函數的使用可能會導致效能降低。Amazon Redshift 會將運算子和函數分類為安全或不安全的查詢RLS受保護資料表。當函數或運算子沒有根據輸入觀察到的任何副作用時，會將其分類為 RLS-安全。特別是，RLS安全函數或運算子不能是下列其中一項：

- 輸出一個輸入值，或任何依賴於輸入值的值，無論有沒有錯誤訊息。
- 失敗或傳回依賴於輸入值的錯誤。

RLS-不安全的運算子包括：

- 算術運算子 — +、-、/、*、%。
- 文字運算子 - LIKE和 SIMILAR TO。
- 轉換運算子。
- UDFs.

使用下列SELECT陳述式來檢查運算子和函數的安全性。

```
SELECT proname, proc_is_qls_safe(oid) FROM pg_proc;
```

Amazon Redshift 在規劃 RLS受保護資料表上的查詢時，會對包含RLS不安全運算子和函數的使用者述詞的評估順序施加限制。參考 RLS- 不安全的運算子或函數的查詢可能會在查詢 RLS- 受保護的資料表時造成效能降低。當 Amazon Redshift 無法將RLS不安全的述詞向下推送到基礎資料表掃描，以利用排序金鑰時，效能可能會大幅降低。為了提升效能，請避免使用利用排序金鑰的RLS不安全述詞進行查詢。若要確認 Amazon Redshift 能夠推低運算子和函數，您可以搭配系統許可 使用 EXPLAIN 陳述式EXPLAINRLS。

結果快取

為了縮短查詢執行期並改善系統效能，Amazon Redshift 會將特定查詢類型的結果快取在領導者節點的記憶體中。

當未受保護資料表的所有條件都為 true 且下列所有條件都為 true 時，Amazon Redshift 會使用快取結果進行新的查詢掃描 RLS- 受保護資料表：

- 政策中的資料表或檢視尚未經過修改。
- 政策不會使用每次執行時都必須評估的函數，例如 GETDATE或 CURRENT_USER。

為了獲得更好的效能，請避免使用不符合前述條件的政策述詞。

如需 Amazon Redshift 中結果快取的詳細資訊，請參閱 [結果快取](#)。

複雜政策

為了獲得更好的效能，請避免在聯結多個資料表的子查詢中使用複雜的政策。

資料列層級安全 end-to-end範例

以下範例 end-to-end說明超級使用者如何建立部分使用者和角色。然後，具有 secadmin 角色的使用者會建立、連接、分離和捨棄RLS政策。此範例會使用票卷範例資料庫。如需詳細資訊，請參閱《Amazon Redshift 入門指南》中的[將資料從 Amazon S3 載入到 Amazon Redshift](#)。

```
-- Create users and roles referenced in the policy statements.
CREATE ROLE analyst;
CREATE ROLE consumer;
CREATE ROLE dbadmin;
CREATE ROLE auditor;
CREATE USER bob WITH PASSWORD 'Name_is_bob_1';
CREATE USER alice WITH PASSWORD 'Name_is_alice_1';
CREATE USER joe WITH PASSWORD 'Name_is_joe_1';
CREATE USER molly WITH PASSWORD 'Name_is_molly_1';
CREATE USER bruce WITH PASSWORD 'Name_is_bruce_1';
GRANT ROLE sys:secadmin TO bob;
GRANT ROLE analyst TO alice;
GRANT ROLE consumer TO joe;
GRANT ROLE dbadmin TO molly;
GRANT ROLE auditor TO bruce;
GRANT ALL ON TABLE tickit_category_redshift TO PUBLIC;
GRANT ALL ON TABLE tickit_sales_redshift TO PUBLIC;
GRANT ALL ON TABLE tickit_event_redshift TO PUBLIC;

-- Create table and schema referenced in the policy statements.
CREATE SCHEMA target_schema;
GRANT ALL ON SCHEMA target_schema TO PUBLIC;
CREATE TABLE target_schema.target_event_table (LIKE tickit_event_redshift);
GRANT ALL ON TABLE target_schema.target_event_table TO PUBLIC;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check the tuples visible to analyst alice.
-- Should contain all 3 categories.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;
```

```
-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');

SELECT polddb, polname, polalias, polatts, polqual, polenabled, polmodifiedby FROM
  svv_qls_policy WHERE polddb = CURRENT_DATABASE();

ATTACH RLS POLICY policy_concerts ON tickit_category_redshift TO ROLE analyst, ROLE
  dbadmin;

ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;

SELECT * FROM svv_qls_attached_policy;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check that tuples with only `Concert` category will be visible to analyst alice.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to consumer joe.
SET SESSION AUTHORIZATION joe;

-- Although the policy is attached to a different role, no tuples will be
-- visible to consumer joe because the default deny all policy is applied.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to dbadmin molly.
SET SESSION AUTHORIZATION molly;

-- Check that tuples with only `Concert` category will be visible to dbadmin molly.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Check that EXPLAIN output contains RLS SecureScan to prevent disclosure of
```

```
-- sensitive information such as RLS filters.
EXPLAIN SELECT catgroup, count(*) FROM tickit_category_redshift GROUP BY catgroup ORDER
  BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

-- Grant IGNORE RLS permission so that RLS policies do not get applicable to role
  dbadmin.
GRANT IGNORE RLS TO ROLE dbadmin;

-- Grant EXPLAIN RLS permission so that anyone in role auditor can view complete
  EXPLAIN output.
GRANT EXPLAIN RLS TO ROLE auditor;

-- Change session to dbadmin molly.
SET SESSION AUTHORIZATION molly;

-- Check that all tuples are visible to dbadmin molly because `IGNORE RLS` is granted
  to role dbadmin.
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to auditor bruce.
SET SESSION AUTHORIZATION bruce;

-- Check explain plan is visible to auditor bruce because `EXPLAIN RLS` is granted to
  role auditor.
EXPLAIN SELECT catgroup, count(*) FROM tickit_category_redshift GROUP BY catgroup ORDER
  BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

DETACH RLS POLICY policy_concerts ON tickit_category_redshift FROM ROLE analyst, ROLE
  dbadmin;

-- Change session to analyst alice.
SET SESSION AUTHORIZATION alice;

-- Check that no tuples are visible to analyst alice.
-- Although the policy is detached, no tuples will be visible to analyst alice
-- because of default deny all policy is applied if the table has RLS on.
```

```
SELECT catgroup, count(*)
FROM tickit_category_redshift
GROUP BY catgroup ORDER BY catgroup;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

CREATE RLS POLICY policy_events
WITH (eventid INTEGER) AS ev
USING (
    ev.eventid IN (SELECT eventid FROM tickit_sales_redshift WHERE qtysold <3)
);

ATTACH RLS POLICY policy_events ON tickit_event_redshift TO ROLE analyst;
ATTACH RLS POLICY policy_events ON target_schema.target_event_table TO ROLE consumer;

RESET SESSION AUTHORIZATION;

-- Can not cannot alter type of dependent column.
ALTER TABLE target_schema.target_event_table ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_event_redshift ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_sales_redshift ALTER COLUMN eventid TYPE float;
ALTER TABLE tickit_sales_redshift ALTER COLUMN qtysold TYPE float;

-- Can not cannot rename dependent column.
ALTER TABLE target_schema.target_event_table RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_event_redshift RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_sales_redshift RENAME COLUMN eventid TO renamed_eventid;
ALTER TABLE tickit_sales_redshift RENAME COLUMN qtysold TO renamed_qtysold;

-- Can not drop dependent column.
ALTER TABLE target_schema.target_event_table DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_event_redshift DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_sales_redshift DROP COLUMN eventid CASCADE;
ALTER TABLE tickit_sales_redshift DROP COLUMN qtysold CASCADE;

-- Can not drop lookup table.
DROP TABLE tickit_sales_redshift CASCADE;

-- Change session to security administrator bob.
SET SESSION AUTHORIZATION bob;

DROP RLS POLICY policy_concerts;
DROP RLS POLICY IF EXISTS policy_events;
```

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY OFF;

RESET SESSION AUTHORIZATION;

-- Drop users and roles.
DROP USER bob;
DROP USER alice;
DROP USER joe;
DROP USER molly;
DROP USER bruce;
DROP ROLE analyst;
DROP ROLE consumer;
DROP ROLE auditor FORCE;
DROP ROLE dbadmin FORCE;
```

中繼資料安全性

如同 Amazon Redshift 的資料列層級安全性一樣，中繼資料安全性可讓您更精細地控制中繼資料。如果針對已佈建的叢集或無伺服器工作群組啟用中繼資料安全性，則使用者可以查看其具有檢視存取權之物件的中繼資料。中繼資料安全性可讓您根據需求分開可見性。例如，您可以使用單一資料倉儲來集中所有資料儲存。但是，如果您為多個扇區儲存資料，則管理安全性會較為棘手。啟用中繼資料安全性後，可以設定可見性。一個扇區的使用者對其物件有更高可見性，同時可限制對另一個磁區使用者的檢視權。中繼資料安全性支援所有物件類型，例如結構描述、資料表、視觀表、具體化視觀表、預存程序、使用者定義函數和機器學習模型。

使用者可以在下列情況下查看物件的中繼資料：

- 如果將物件存取權授予使用者。
- 如果將物件存取權授予使用者所屬的群組或角色。
- 物件是公有的。
- 使用者是資料庫物件的擁有者。

若要啟用中繼資料安全，請使用 [ALTERSYSTEM](#) 命令。以下是如何使用 ALTERSYSTEM 命令與中繼資料安全性的語法。

```
ALTER SYSTEM SET metadata_security=[true|t|on|false|f|off];
```

當您啟用中繼資料安全性時，擁有必要權限的所有使用者都有權查看他們可存取之物件的相關中繼資料。如果您只想讓特定使用者查看中繼資料安全性，請將 ACCESS CATALOG 權限授予某角色，然後再將該角色指派給使用者。如需有關使用角色進一步控制安全性的詳細資訊，請參閱[角色型存取控制](#)。

下列範例示範如何將 ACCESS CATALOG 權限授予某角色，然後再將該角色指派給使用者。如需授予許可的詳細資訊，請參閱 [GRANT](#) 命令。

```
CREATE ROLE sample_metadata_viewer;  
  
GRANT ACCESS CATALOG TO ROLE sample_metadata_viewer;  
  
GRANT ROLE sample_metadata_viewer to salesadmin;
```

如果您偏好使用已定義的角色，則[系統定義的角色](#) operator、secadmin、dba 和 superuser 所有角色都具有檢視物件中繼資料的必要權限。預設情況下，超級使用者可以查看完整的目錄。

```
GRANT ROLE operator to sample_user;
```

如果您使用角色來控制中繼資料安全性，則可以存取角色型存取控制隨附的所有系統檢視和功能。例如，您可以查詢 [SVV_ROLES](#) 檢視以查看所有角色。若要查看使用者是否為角色或群組的成員，請使用 [USER_IS_MEMBER_OF](#) 函數。如需 SVV 檢視的完整清單，請參閱[SVV 中繼資料檢視](#)。如需系統資訊函數清單，請參閱[系統資訊函數](#)。

動態資料遮罩

在 Amazon Redshift 中使用動態資料遮罩 (DDM)，您可以保護資料倉儲中的敏感資料。您可以操控 Amazon Redshift 在查詢時向使用者顯示敏感資料的方式，而無需在資料庫中進行轉換。您可以透過將自訂混淆規則套用至指定使用者或角色的遮罩政策，來控制資料的存取。如此一來，您可以回應不斷變化的隱私權需求，而不必變更基礎資料或編輯 SQL 查詢。

動態資料遮罩原則會隱藏、混淆或虛擬化符合指定格式的資料。當附加至資料表時，遮罩運算式會套用至其一或多個資料欄。您可以進一步修改遮罩政策，只將政策套用至特定使用者，或套用至可以使用[角色型存取控制 \(RBAC\)](#) 建立的使用者定義角色。此外，您可以在建立遮罩政策時，使用條件式資料欄來套用至 DDM 儲存格層級。如需條件式遮罩的詳細資訊，請參閱[條件式動態資料遮罩](#)。

您可以將不同混淆層級的多個遮罩政策套用至資料表中的相同資料欄，並將期指派給不同的角色。當您有不同角色且搭配套用至一個資料欄的不同政策時，為避免發生衝突，您可以為每個應用程式設定優先順序。如此一來，您就可以控制指定使用者或角色可以存取的資料。DDM 政策可以透過使用

SQL、Python 或中寫入的使用者定義函數來部分或完全編輯資料，或雜湊資料 AWS Lambda。透過使用雜湊來遮罩資料，您可以在不存取潛在敏感資訊的情況下對此資料套用聯結。

SQL 用於管理動態資料遮罩政策的 命令

您可以執行下列動作來建立、連接、分離和刪除動態資料遮罩政策：

- 若要建立DDM政策，請使用 [CREATE MASKING POLICY](#) 命令。

以下是使用 SHA-2 雜湊函數建立遮罩政策的範例。

```
CREATE MASKING POLICY hash_credit
WITH (credit_card varchar(256))
USING (sha2(credit_card + 'testSalt', 256));
```

- 若要變更現有DDM政策，請使用 [ALTER MASKING POLICY](#) 命令。

下列是變更現有遮罩政策的範例。

```
ALTER MASKING POLICY hash_credit
USING (sha2(credit_card + 'otherTestSalt', 256));
```

- 若要將資料表上的DDM政策連接至一或多個使用者或角色，請使用 [ATTACH MASKING POLICY](#) 命令。

下列是將遮罩政策附加至資料欄/角色組的範例。

```
ATTACH MASKING POLICY hash_credit
ON credit_cards (credit_card)
TO ROLE science_role
PRIORITY 30;
```

PRIORITY 子句決定當多個政策連接到同一資料欄時，哪些遮罩政策會套用至使用者工作階段。例如，如果上述範例中的使用者在相同的信用卡資料欄中附加了另一個遮罩政策，且優先順序為 20，那麼 science_role 的政策就是適用的政策，因為其具有較高的優先順序 30。

- 若要將資料表上的DDM政策與一或多個使用者或角色分離，請使用 [DETACH MASKING POLICY](#) 命令。

下列是將遮罩政策從資料欄/角色組分離的範例。

```
DETACH MASKING POLICY hash_credit
```

```
ON credit_cards(credit_card)
FROM ROLE science_role;
```

- 若要從所有資料庫捨棄DDM政策，請使用 [DROP MASKING POLICY](#) 命令。

以下是從所有資料庫捨棄遮罩政策的範例。

```
DROP MASKING POLICY hash_credit;
```

動態資料遮罩政策階層

附加多項遮罩政策時，請考慮下列事項：

- 您可以將多個遮罩政策附加至單一資料欄。
- 當查詢適用多個遮罩政策時，則會套用附加至每個資料欄且優先順序最高的政策。請考量下列範例。

```
ATTACH MASKING POLICY partial_hash
ON credit_cards(address, credit_card)
TO ROLE analytics_role
PRIORITY 20;
```

```
ATTACH MASKING POLICY full_hash
ON credit_cards(credit_card, ssn)
TO ROLE auditor_role
PRIORITY 30;
```

```
SELECT address, credit_card, ssn
FROM credit_cards;
```

執行SELECT陳述式時，同時具有分析和稽核員角色的使用者會看到套用partial_hash遮罩政策的地址欄。他們看到套用full_hash遮罩政策的信用卡和SSN資料欄，因為該full_hash政策在信用卡資料欄具有較高的優先順序。

- 如果附加遮罩政策時未指定優先順序，則預設的優先順序為 0。
- 您無法將兩個政策附加至具有相同優先順序的相同資料欄。
- 您無法將兩個政策附加到使用者和欄或角色和欄的相同組合。
- 當多個遮罩政策在連接至相同使用者或角色時沿相同SUPER路徑套用時，只有最高優先順序的連接才會生效。請考慮下列範例。

第一個範例顯示在相同路徑上附加的兩個遮罩政策，優先順序較高的政策會生效。

```
ATTACH MASKING POLICY hide_name
ON employees(col_person.name)
TO PUBLIC
PRIORITY 20;

ATTACH MASKING POLICY hide_last_name
ON employees(col_person.name.last)
TO PUBLIC
PRIORITY 30;

--Only the hide_last_name policy takes effect.
SELECT employees.col_person.name FROM employees;
```

第二個範例顯示連接到相同SUPER物件中不同路徑的兩個遮罩政策，政策之間沒有衝突。兩項附件同時採用。

```
ATTACH MASKING POLICY hide_first_name
ON employees(col_person.name.first)
TO PUBLIC
PRIORITY 20;

ATTACH MASKING POLICY hide_last_name
ON employees(col_person.name.last)
TO PUBLIC
PRIORITY 20;

--Both col_person.name.first and col_person.name.last are masked.
SELECT employees.col_person.name FROM employees;
```

若要確認哪個遮罩政策適用於指定的使用者和資料欄，或角色和資料欄的組合，具有 [sys:secadmin](#) 角色的使用者可以在 [SVV_ATTACHED_MASKING_POLICY](#) 系統檢視中查詢資料欄/角色或資料欄/使用者配對。如需詳細資訊，請參閱[動態資料遮罩系統檢視](#)。

搭配SUPER資料類型路徑使用動態資料遮罩

Amazon Redshift 支援將動態資料遮罩政策連接至SUPER類型資料欄的路徑。如需SUPER資料類型的詳細資訊，請參閱 [Amazon Redshift 中的半結構化資料](#)。

將遮罩政策連接至SUPER類型資料欄的路徑時，請考慮下列事項。

- 將遮罩政策連接至資料欄上的路徑時，該資料欄必須定義為SUPER資料類型。您只能將遮罩政策套用至SUPER路徑上的純量值。您無法將遮罩政策套用至複雜的結構或陣列。
- 只要SUPER路徑沒有衝突，您可以將不同的遮罩政策套用至單一SUPER資料欄上的多個純量值。例如，SUPER路徑a.b和a.b.c衝突，因為它們位於相同的路徑上，而a.b是父項a.b.c。SUPER路徑a.b.c和a.b.d不會衝突。
- 在使用者查詢執行期套用政策之前，Amazon Redshift無法檢查遮罩政策附加的路徑是否存在於資料中，且為預期類型的路徑。例如，當您將遮罩TEXT值的遮罩政策附加到包含INT值的SUPER路徑時，Amazon Redshift會嘗試在路徑中投放該值的類型。

在這種情況下，Amazon Redshift在執行階段的行為取決於您查詢SUPER物件的組態設定。依預設，Amazon Redshift處於通量模式，並會像NULL指定路徑一樣解決遺失的路徑和無效的轉換SUPER。如需SUPER相關組態設定的詳細資訊，請參閱 [SUPER 組態](#)。

- SUPER是無結構描述類型，這表示Amazon Redshift無法確認指定SUPER路徑中值的存在。如果您將遮罩政策連接到不存在的SUPER路徑，且Amazon Redshift處於寬鬆模式，Amazon Redshift會將路徑解析為NULL值。建議您在將遮罩政策連接至SUPER資料欄路徑時，考慮SUPER物件的預期格式及其具有非預期屬性的可能性。如果您認為資料SUPER欄中可能有非預期的結構描述，請考慮將遮罩政策直接連接至資料SUPER欄。您可以使用SUPER類型資訊函數來檢查屬性和類型，並使用OBJECT_TRANSFORM來遮罩值。如需SUPER類型資訊函數的詳細資訊，請參閱 [SUPER 類型資訊函數](#)。

範例

將遮罩政策連接至SUPER路徑

下列範例會將多個遮罩政策連接到一欄中的多種SUPER類型路徑。

```
CREATE TABLE employees (  
    col_person SUPER  
);  
  
INSERT INTO employees  
VALUES  
    (  
        json_parse(''  
            {  
                "name": {  
                    "first": "John",
```

```
        "last": "Doe"
    },
    "age": 25,
    "ssn": "111-22-3333",
    "company": "Company Inc."
  }
)
),
(
  json_parse('
    {
      "name": {
        "first": "Jane",
        "last": "Appleseed"
      },
      "age": 34,
      "ssn": "444-55-7777",
      "company": "Organization Org."
    }
  ')
)
;
GRANT ALL ON ALL TABLES IN SCHEMA "public" TO PUBLIC;

-- Create the masking policies.

-- This policy converts the given name to all uppercase letters.
CREATE MASKING POLICY mask_first_name
WITH(first_name TEXT)
USING ( UPPER(first_name) );

-- This policy replaces the given name with the fixed string 'XXXX'.
CREATE MASKING POLICY mask_last_name
WITH(last_name TEXT)
USING ( 'XXXX'::TEXT );

-- This policy rounds down the given age to the nearest 10.
CREATE MASKING POLICY mask_age
WITH(age INT)
USING ( (FLOOR(age::FLOAT / 10) * 10)::INT );

-- This policy converts the first five digits of the given SSN to 'XXX-XX'.
CREATE MASKING POLICY mask_ssn
WITH(ssn TEXT)
```

```

USING ( 'XXX-XX-':TEXT || SUBSTRING(ssn::TEXT FROM 8 FOR 4) );

-- Attach the masking policies to the employees table.
ATTACH MASKING POLICY mask_first_name
ON employees(col_person.name.first)
TO PUBLIC;

ATTACH MASKING POLICY mask_last_name
ON employees(col_person.name.last)
TO PUBLIC;

ATTACH MASKING POLICY mask_age
ON employees(col_person.age)
TO PUBLIC;

ATTACH MASKING POLICY mask_ssn
ON employees(col_person.ssn)
TO PUBLIC;

-- Verify that your masking policies are attached.
SELECT
    policy_name,
    TABLE_NAME,
    priority,
    input_columns,
    output_columns
FROM
    svv_attached_masking_policy;

    policy_name | table_name | priority |          input_columns          |
output_columns
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
mask_age       | employees |         0 | ["col_person.\"age\""]         |
["col_person.\"age\""]
mask_first_name | employees |         0 | ["col_person.\"name\".\"first\""] |
["col_person.\"name\".\"first\""]
mask_last_name | employees |         0 | ["col_person.\"name\".\"last\""] |
["col_person.\"name\".\"last\""]
mask_ssn       | employees |         0 | ["col_person.\"ssn\""]         |
["col_person.\"ssn\""]
(4 rows)

-- Observe the masking policies taking effect.

```

```

SELECT col_person FROM employees ORDER BY col_person.age;

-- This result is formatted for ease of reading.
      col_person
-----
{
  "name": {
    "first": "JOHN",
    "last": "XXXX"
  },
  "age": 20,
  "ssn": "XXX-XX-3333",
  "company": "Company Inc."
}
{
  "name": {
    "first": "JANE",
    "last": "XXXX"
  },
  "age": 30,
  "ssn": "XXX-XX-7777",
  "company": "Organization Org."
}

```

以下是SUPER一些無效遮罩政策附件路徑的範例。

```

-- This attachment fails because there is already a policy
-- with equal priority attached to employees.name.last, which is
-- on the same SUPER path as employees.name.
ATTACH MASKING POLICY mask_ssn
ON employees(col_person.name)
TO PUBLIC;
ERROR: DDM policy "mask_last_name" is already attached on relation "employees" column
"col_person."name"."last"" with same priority

-- Create a masking policy that masks DATETIME objects.
CREATE MASKING POLICY mask_date
WITH(INPUT DATETIME)
USING ( INPUT );

-- This attachment fails because SUPER type columns can't contain DATETIME objects.
ATTACH MASKING POLICY mask_date
ON employees(col_person.company)

```

```
TO PUBLIC;
ERROR: cannot attach masking policy for output of type "timestamp without time zone"
to column "col_person."company"" of type "super
```

以下是將遮罩政策連接到不存在SUPER路徑的範例。預設情況下，Amazon Redshift 會將路徑解析到 NULL。

```
ATTACH MASKING POLICY mask_first_name
ON employees(col_person.not_exists)
TO PUBLIC;

SELECT col_person FROM employees LIMIT 1;

-- This result is formatted for ease of reading.
    col_person
-----
{
  "name": {
    "first": "JOHN",
    "last": "XXXX"
  },
  "age": 20,
  "ssn": "XXX-XX-3333",
  "company": "Company Inc.",
  "not_exists": null
}
```

條件式動態資料遮罩

您可以在遮罩運算式中使用條件運算式建立遮罩政策，以在儲存格層級上遮罩資料。例如，您可以建立一個遮罩政策，根據該資料列中其他資料欄的值，將不同的遮罩套用至值。

以下是使用條件式資料遮罩來建立並附加遮罩政策的範例，該遮罩政策會修改涉及詐騙的部分信用卡號碼，同時完全隱藏所有其他信用卡號碼。您必須是超級使用者或具有 [sys:secadmin](#) 角色才能執行此範例。

```
--Create an analyst role.
CREATE ROLE analyst;

--Create a credit card table. The table contains an is_fraud boolean column,
--which is TRUE if the credit card number in that row was involved in a fraudulent
transaction.
```



```
CREATE TABLE credit_cards (id INT, is_fraud BOOLEAN, credit_card_number VARCHAR(16));

--Create a function that partially redacts credit card numbers.
CREATE FUNCTION REDACT_CREDIT_CARD (credit_card VARCHAR(16))
RETURNS VARCHAR(16) IMMUTABLE
AS $$
    import re
    regexp = re.compile("^[0-9]{6}[0-9]{5,6}([0-9]{4})")

    match = regexp.search(credit_card)
    if match != None:
        first = match.group(1)
        last = match.group(2)
    else:
        first = "000000"
        last = "0000"

    return "{}XXXXX{}".format(first, last)
$$ LANGUAGE plpythonu;

--Create a masking policy that partially redacts credit card numbers if the is_fraud
value for that row is TRUE,
--and otherwise blanks out the credit card number completely.
CREATE MASKING POLICY card_number_conditional_mask
    WITH (fraudulent BOOLEAN, pan varchar(16))
    USING (CASE WHEN fraudulent THEN REDACT_CREDIT_CARD(pan)
            ELSE Null
        END);

--Attach the masking policy to the credit_cards/analyst table/role pair.
ATTACH MASKING POLICY card_number_conditional_mask ON credit_cards (credit_card_number)
USING (is_fraud, credit_card_number)
TO ROLE analyst PRIORITY 100;
```

動態資料遮罩系統檢視

超級使用者、具有 `sys:operator` 角色的使用者以及具有 `ACCESSSYSTEMTABLE` 許可的使用者都可以存取下列 DDM 相關的系統檢視。

- [SVV_MASKING_POLICY](#)

使用 `SVV_MASKING_POLICY` 檢視在叢集或工作群組上建立的所有遮罩政策。

- [SVV_ATTACHED_MASKING_POLICY](#)

使用 `SVV_ATTACHED_MASKING_POLICY` 來檢視目前連線的資料庫中附加政策的所有關係和使用者或角色。

- [`SYS_APPLIED_MASKING_POLICY_LOG`](#)

使用 `SYS_APPLIED_MASKING_POLICY_LOG` 在參考 DDM 受保護關係的查詢上追蹤遮罩政策的應用程式。

以下是您可以使用系統檢視找到的一些資訊範例。

```
--Select all policies associated with specific users, as opposed to roles
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE grantee_type = 'user';

--Select all policies attached to a specific user
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE grantee = 'target_grantee_name';

--Select all policies attached to a given table
SELECT policy_name,
       schema_name,
       table_name,
       grantee
FROM svv_attached_masking_policy
WHERE table_name = 'target_table_name'
      AND schema_name = 'target_schema_name';

--Select the highest priority policy attachment for a given role
SELECT samp.policy_name,
       samp.priority,
       samp.grantee,
       samp.policy_expression
FROM svv_masking_policy AS samp
JOIN svv_attached_masking_policy AS samp
```

```
    ON samp.policy_name = smp.policy_name
WHERE
    samp.grantee_type = 'role' AND
    samp.policy_name = mask_get_policy_for_role_on_column(
        'target_schema_name',
        'target_table_name',
        'target_column_name',
        'target_role_name')
ORDER BY samp.priority desc
LIMIT 1;

--See which policy a specific user will see on a specific column in a given relation
SELECT samp.policy_name,
       samp.priority,
       samp.grantee,
       smp.policy_expression
FROM svv_masking_policy AS smp
JOIN svv_attached_masking_policy AS samp
    ON samp.policy_name = smp.policy_name
WHERE
    samp.grantee_type = 'role' AND
    samp.policy_name = mask_get_policy_for_user_on_column(
        'target_schema_name',
        'target_table_name',
        'target_column_name',
        'target_user_name')
ORDER BY samp.priority desc;

--Select all policies attached to a given relation.
SELECT policy_name,
       schema_name,
       relation_name,
       database_name
FROM sys_applied_masking_policy_log
WHERE relation_name = 'relation_name'
AND schema_name = 'schema_name';
```

使用動態資料遮罩時的考量

使用動態資料遮罩時，請考量下列內容：

- 查詢從資料表建立的物件 (例如檢視) 時，使用者會根據自己的遮罩政策 (而非建立物件之使用者的政策) 來看到結果。例如，具有分析師角色的使用者查詢由 secadmin 建立的檢視時，就會看到附加到分析師角色的遮罩政策的結果。
- 為了防止EXPLAIN命令可能暴露敏感遮罩政策篩選條件，只有具有 SYS_EXPLAIN_DDM 許可的使用者才能查看套用至EXPLAIN輸出中的遮罩政策。根據預設，使用者沒有 SYS_EXPLAIN_DDM 許可。

以下是將許可授予角色的語法。

```
GRANT EXPLAIN MASKING TO ROLE rolename
```

如需 EXPLAIN命令的詳細資訊，請參閱 [EXPLAIN](#)。

- 具有不同角色的使用者可以根據所使用的篩選條件或聯結條件看到不同的結果。例如，如果執行 SELECT命令的使用者套用了混淆該資料欄的遮罩政策，則使用特定資料欄值在資料表上執行命令將會失敗。
- DDM 政策必須在任何述詞操作或投影之前套用。遮罩政策可能包含下列項目：
 - 低成本常數操作，例如將值轉換為 null
 - 適度的成本操作，例如HMAC雜湊
 - 高成本操作，例如呼叫外部 Lambda 使用者定義函數

如此一來，我們建議您盡可能使用簡單遮罩運算式。

- 您可以針對具有資料列層級安全政策的角色使用DDM政策，但請注意，RLS政策是在 之前套用的 DDM。動態資料遮罩表達式將無法讀取受到 保護的資料列RLS。如需 的詳細資訊RLS，請參閱 [資料列層級安全性](#)。
- 使用 [COPY](#)命令從 parquet 複製到受保護的目標資料表時，您應該明確指定COPY陳述式中的資料欄。如需使用 對應資料欄的詳細資訊COPY，請參閱 [欄映射選項](#)。
- DDM 政策無法連接至下列關係：
 - 系統表格和目錄
 - 外部資料表
 - 資料共用資料表
 - 具體化視觀表
 - 跨資料庫關係
 - 暫時資料表
 - 相關查詢

- DDM 政策可以包含查詢資料表。子USING句中可能存在查詢資料表。下列關係類型無法用作查閱資料表：
 - 系統表格和目錄
 - 外部資料表
 - 資料共用資料表
 - 檢視、具體化視觀表和近期繫結視觀表
 - 跨資料庫關係
 - 暫時資料表
 - 相關查詢

以下是將遮罩政策附加至查閱資料表的範例。

```
--Create a masking policy referencing a lookup table
CREATE MASKING POLICY lookup_mask_credit_card WITH (credit_card TEXT) USING (
  CASE
    WHEN
      credit_card IN (SELECT credit_card_lookup FROM credit_cards_lookup)
    THEN '000000XXXX0000'
    ELSE REDACT_CREDIT_CARD(credit_card)
  END
);

--Provides access to the lookup table via a policy attached to a role
GRANT SELECT ON TABLE credit_cards_lookup TO MASKING POLICY lookup_mask_credit_card;
```

- 如果遮罩政策會產生與目標資料欄類型和大小不相容的輸出，則您無法附加該遮罩政策。例如，您無法將輸出 12 個字元長字串的遮罩政策連接至 VARCHAR (10) 欄。Amazon Redshift 支援以下例外情況：
 - 具有輸入類型的遮罩政策INTN可以連接到大小INTM為 $M < N$ 的政策。例如，BIGINT (INT8) 輸入政策可以連接到小型 (INT4) 資料欄。
 - 具有輸入類型 NUMERIC或 的遮罩政策，一律DECIMAL可以連接至資料FLOAT欄。
- DDM 政策無法與資料共用搭配使用。如果資料共用的資料生產者將DDM政策連接到資料共用中的資料表，則嘗試查詢資料表的資料取用者的使用者無法存取資料表。附接DDM政策的資料表無法新增至資料共用。
- 如果您的以下任何組態選項的值與工作階段的預設值不相符，則無法查詢已連接DDM政策的關係：
 - enable_case_sensitive_super_attribute
 - enable_case_sensitive_identifier

- `downcase_delimited_identifier`

如果您嘗試查詢已連接DDM政策的關係，並看到訊息「DDM受保護關係不支援案例敏感度與其預設值不同的工作階段層級組態」，請考慮重設工作階段的組態選項。

- 當您佈建的叢集或無伺服器命名空間具有任何動態資料遮罩政策時，一般使用者會無法使用下列命令：

```
ALTER <current_user> SET enable_case_sensitive_super_attribute/  
enable_case_sensitive_identifier/downcase_delimited_identifier
```

建立DDM政策時，建議您變更一般使用者的預設組態選項設定，以符合政策建立時的工作階段組態選項設定。具有 ALTERUSER 權限的超級使用者和使用者可以使用參數群組設定或 ALTERUSER 命令來執行此操作。如需有關參數群組的詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 參數群組](#)。如需有關 ALTERUSER 命令的資訊，請參閱 [ALTER USER](#)。

- 具有連接DDM政策的檢視和繫結後期檢視無法由一般使用者使用 [CREATE VIEW](#) 命令取代。若要將檢視或取代LBVs為DDM政策，請先分離連接到它們的任何DDM政策，取代檢視或LBVs，然後重新連接政策。具有 `sys:secadmin` 許可的超級使用者和使用者可以CREATEVIEW在檢視上使用，或LBVs搭配DDM政策使用，而無需分離政策。
- 具有附加DDM政策的檢視無法參考系統資料表和檢視。近期繫結視觀表可以參考系統資料表和視觀表。
- 附接DDM政策的繫結後期檢視無法參考資料湖中的巢狀資料，例如JSON文件。
- 如果任何檢視參考了該晚繫結檢視，則晚繫結檢視無法連接DDM政策。
- DDM 連接至繫結後期檢視的政策會依資料欄名稱附加。在查詢時，Amazon Redshift 會驗證附加到近期繫結視觀表的所有遮罩政策是否已成功套用，而且近期繫結視觀表的輸出資料欄類型是否符合附加的遮罩政策的類型。如果驗證失敗，Amazon Redshift 會傳回查詢的錯誤訊息。
- 您可以在建立DDM政策時使用自訂工作階段內容變數。下列範例會設定DDM政策的工作階段內容變數。

```
-- Set a customized context variable.  
SELECT set_config('app.city', 'XXXX', FALSE);  
  
-- Create a MASKING policy using current_setting() to get the value of a customized  
  context variable.  
CREATE MASKING POLICY city_mask  
WITH (city VARCHAR(30))  
USING (current_setting('app.city')::VARCHAR(30));
```

```
-- Attach the policy on the target table to one or more roles.  
ATTACH MASKING POLICY city_mask  
ON tickit_users_redshift(city)  
TO ROLE analyst, ROLE dbadmin;
```

如需如何設定和擷取自訂工作階段內容變數的詳細資訊，請前往 [SET](#)、[SET_CONFIG](#)、[SHOW](#)、[CURRENT_SETTING](#)和 [RESET](#)。如需修改伺服器組態的一般詳細資訊，請前往 [修改伺服器組態](#)。

Important

在DDM政策中使用工作階段內容變數時，安全政策依賴於叫用政策的使用者或角色。在DDM政策中使用工作階段內容變數時，請小心避免安全漏洞。

動態資料遮罩 end-to-end範例

以下是 end-to-end 示範如何建立遮罩政策並將其連接至資料欄的範例。這些政策可讓使用者存取資料欄並查看不同的值，視附加至其角色的政策中的混淆程度而定。您必須是超級使用者或具有 [sys:secadmin](#) 角色才能執行此範例。

建立遮罩政策

首先，建立一個資料表，並填入信用卡值。

```
--create the table  
CREATE TABLE credit_cards (  
    customer_id INT,  
    credit_card TEXT  
);  
  
--populate the table with sample values  
INSERT INTO credit_cards  
VALUES  
    (100, '4532993817514842'),  
    (100, '4716002041425888'),  
    (102, '5243112427642649'),  
    (102, '6011720771834675'),  
    (102, '6011378662059710'),  
    (103, '373611968625635')  
;
```

```
--run GRANT to grant permission to use the SELECT statement on the table
GRANT SELECT ON credit_cards TO PUBLIC;

--create two users
CREATE USER regular_user WITH PASSWORD '1234Test!';

CREATE USER analytics_user WITH PASSWORD '1234Test!';

--create the analytics_role role and grant it to analytics_user
--regular_user does not have a role
CREATE ROLE analytics_role;

GRANT ROLE analytics_role TO analytics_user;
```

接下來，建立要套用至分析角色的遮罩政策。

```
--create a masking policy that fully masks the credit card number
CREATE MASKING POLICY mask_credit_card_full
WITH (credit_card VARCHAR(256))
USING ('000000XXXX0000'::TEXT);

--create a user-defined function that partially obfuscates credit card data
CREATE FUNCTION REDACT_CREDIT_CARD (credit_card TEXT)
RETURNS TEXT IMMUTABLE
AS $$
    import re
    regexp = re.compile("^[0-9]{6}[0-9]{5,6}([0-9]{4})")

    match = regexp.search(credit_card)
    if match != None:
        first = match.group(1)
        last = match.group(2)
    else:
        first = "000000"
        last = "0000"

    return "{}XXXXX{}".format(first, last)
$$ LANGUAGE plpythonu;

--create a masking policy that applies the REDACT_CREDIT_CARD function
CREATE MASKING POLICY mask_credit_card_partial
WITH (credit_card VARCHAR(256))
USING (REDACT_CREDIT_CARD(credit_card));
```



```
--confirm the masking policies using the associated system views
SELECT * FROM svv_masking_policy;

SELECT * FROM svv_attached_masking_policy;
```

附加遮罩政策

將遮罩政策附加至信用卡資料表。

```
--attach mask_credit_card_full to the credit card table as the default policy
--all users will see this masking policy unless a higher priority masking policy is
  attached to them or their role
ATTACH MASKING POLICY mask_credit_card_full
ON credit_cards(credit_card)
TO PUBLIC;

--attach mask_credit_card_partial to the analytics role
--users with the analytics role can see partial credit card information
ATTACH MASKING POLICY mask_credit_card_partial
ON credit_cards(credit_card)
TO ROLE analytics_role
PRIORITY 10;

--confirm the masking policies are applied to the table and role in the associated
  system view
SELECT * FROM svv_attached_masking_policy;

--confirm the full masking policy is in place for normal users by selecting from the
  credit card table as regular_user
SET SESSION AUTHORIZATION regular_user;

SELECT * FROM credit_cards;

--confirm the partial masking policy is in place for users with the analytics role by
  selecting from the credit card table as analytics_user
SET SESSION AUTHORIZATION analytics_user;

SELECT * FROM credit_cards;
```

修改遮罩政策

下一節說明如何修改動態資料遮罩政策。

```
--reset session authorization to the default
RESET SESSION AUTHORIZATION;

--alter the mask_credit_card_full policy
ALTER MASKING POLICY mask_credit_card_full
USING ('0000000000000000'::TEXT);

--confirm the full masking policy is in place after altering the policy, and that
results are altered from '000000XXXX0000' to '0000000000000000'
SELECT * FROM credit_cards;
```

分離並捨棄遮罩政策

下列區段說明如何移除資料表中的所有動態資料遮罩政策，以分離和捨棄遮罩政策。

```
--reset session authorization to the default
RESET SESSION AUTHORIZATION;

--detach both masking policies from the credit_cards table
DETACH MASKING POLICY mask_credit_card_full
ON credit_cards(credit_card)
FROM PUBLIC;

DETACH MASKING POLICY mask_credit_card_partial
ON credit_cards(credit_card)
FROM ROLE analytics_role;

--drop both masking policies
DROP MASKING POLICY mask_credit_card_full;

DROP MASKING POLICY mask_credit_card_partial;
```

限定範圍權限

範圍許可可讓您將許可授予資料庫或結構描述內類型的所有物件的使用者或角色。具有範圍許可的使用者和角色在資料庫或結構描述中的所有目前和未來物件上具有指定的許可。

您可以在 [中檢視資料庫層級範圍許可的範圍](#)[SVV_DATABASE_PRIVILEGES](#)。您可以在 [中檢視結構描述層級範圍許可的範圍](#)[SVV_SCHEMA_PRIVILEGES](#)。

如需套用限定範圍的權限的詳細資訊，請參閱 [GRANT](#) 和 [REVOKE](#)。

使用限定範圍的權限考量

使用限定範圍的權限時，請考量下列事項：

- 您可以使用範圍許可，將資料庫或結構描述範圍的許可授予或撤銷給指定的使用者或角色。
- 您無法將範圍許可授予使用者群組。
- 授予或撤銷限定範圍的權限會變更範圍中所有目前和未來物件的權限。
- 範圍許可和物件層級許可彼此獨立運作。例如，使用者將在下列兩種情況下維護資料表的許可。
 - 在資料表 `schema1.table1` SELECT 上授予使用者，並在 `schema1` 上授予 SELECT 範圍許可。然後，使用者已 SELECT 撤銷結構描述結構描述1 中所有資料表的。使用者保留 SELECT `schema1.table1`。
 - 在資料表 `schema1.table1` SELECT 上授予使用者，並在 `schema1` 上授予 SELECT 範圍許可。然後，使用者已 SELECT 撤銷 `schema1.table1` 的。使用者保留 SELECT `schema1.table1`。
- 若要授予或撤銷限定範圍的權限，您必須符合下列其中一項條件：
 - 超級使用者。
 - 具有該權限授予選項的使用者。如需授予選項的詳細資訊，請前往 [中的 WITH GRANT OPTION 參數 GRANT](#)。
- 限定範圍的權限只能授予或撤銷已連線資料庫的物件，或從資料共用匯入的資料庫撤銷。
- 您可以使用範圍許可，在從資料共用建立的資料庫上設定預設許可。被授予共用資料庫限定範圍權限的取用者端資料共用使用者，將自動獲得新增到生產者端資料共用之任何新物件的那些權限。
- 生產者可以將結構描述內物件的範圍許可授予資料共用。（預覽）

SQL 參考

透過 Amazon Redshift，您可以利用 SQL 有效率地查詢和分析儲存在資料倉儲中的大量資料。SQL 參考涵蓋 SQL 命令、資料類型、函數、運算子等的語法和用量，可讓您擷取見解並做出資料驅動的決策。您可以參考此參考來撰寫最佳化查詢、建立和管理資料庫物件，以及執行複雜的資料轉換。下列參考提供在 Amazon Redshift SQL 環境中使用以滿足資料分析需求的全面詳細資訊。

主題

- [Amazon Redshift SQL](#)
- [使用 SQL](#)
- [SQL 命令](#)
- [SQL 函數參考](#)
- [保留字](#)

Amazon Redshift SQL

主題

- [SQL 領導節點上支援的 函數](#)
- [Amazon Redshift 和 PostgreSQL](#)

Amazon Redshift 是以業界標準為基礎建置 SQL，並新增了管理非常大型資料集的功能，並支援這些資料的高效能分析和報告。

Note

單一 Amazon Redshift SQL 陳述式的大小上限為 16 MB。

SQL 領導節點上支援的 函數

某些 Amazon Redshift 查詢會分配到運算節點上執行，其他的查詢則是只在領導節點上執行。

每當查詢參考使用者建立的資料表或系統資料表（具有 STL 或 STV 字首的資料表，以及具有 SVL 或 SVV 字首的系統檢視）時，領導節點就會分發 SQL 給運算節點。僅參考目錄資料表（具有 PG 字首的

資料表，例如DEF位於領導節點上的 PG_TABLE_) 或不參考任何資料表的查詢，僅在領導節點上執行。

某些 Amazon Redshift SQL函數僅支援領導節點，運算節點不支援。使用領導者節點函數的查詢必須完全在領導者節點上執行，而不是在運算節點上，否則會傳回錯誤。

必須只在領導節點上執行的每個函數，其文件包含備註，說明如果函數參考使用者定義的資料表或 Amazon Redshift 系統資料表，將會傳回錯誤。關於只在領導節點上執行的功能，如需功能清單，請參閱 [僅限領導節點函數](#)。

範例

下列範例使用範例TICKIT資料庫。如需範例資料庫的詳細資訊，請前往 [範本資料庫](#)。

CURRENT_SCHEMA

CURRENT_SCHEMA 函數是僅限領導者節點的函數。在此範例中，查詢並未參考資料表，因此只會在領導節點上執行。

```
select current_schema();
```

```
current_schema
-----
public
```

在下一個範例中，查詢參考了系統目錄資料表，因此只會在領導節點上執行。

```
select * from pg_table_def
where schemaname = current_schema() limit 1;
```

```
schemaname | tablename | column | type | encoding | distkey | sortkey | notnull
-----+-----+-----+-----+-----+-----+-----+-----
public     | category | catid  | smallint | none      | t        |          | 1 | t
```

在下一個範例中，查詢參考了位於運算節點上的 Amazon Redshift 系統資料表，因此會傳回錯誤。

```
select current_schema(), userid from users;
```

```
INFO:  Function "current_schema()" not supported.
ERROR: Specified types or functions (one per INFO message) not supported on Amazon
Redshift tables.
```

SUBSTR

SUBSTR 也是領導者節點限定函數。在以下範例中，因查詢未參考資料表，所以只會在領導節點上執行。

```
SELECT SUBSTR('amazon', 5);
```

```
+-----+
| substr |
+-----+
| on     |
+-----+
```

在下列範例中，查詢會參照位於運算節點上的資料表。這會導致錯誤。

```
SELECT SUBSTR(catdesc, 1) FROM category LIMIT 1;
```

```
ERROR: SUBSTR() function is not supported (Hint: use SUBSTRING instead)
```

若要成功執行先前的查詢，請使用 [SUBSTRING](#)。

```
SELECT SUBSTRING(catdesc, 1) FROM category LIMIT 1;
```

```
+-----+
|          substring          |
+-----+
| National Basketball Association |
+-----+
```

FACTORIAL()

FACTORIAL () 是僅限領導者節點的函數。在以下範例中，因查詢未參考資料表，所以只會在領導節點上執行。

```
SELECT FACTORIAL(5);
```

```
factorial
-----
120
```

在下列範例中，查詢會參照位於運算節點上的資料表。使用查詢編輯器 v2 執行時，這會導致錯誤。

```
create table t(a int);
insert into t values (5);
select factorial(a) from t;
```

```
ERROR: Specified types or functions (one per INFO message) not supported on Redshift tables.
```

```
Info: Function "factorial(bigint)" not supported.
```

Amazon Redshift 和 PostgreSQL

主題

- [Amazon Redshift 和 PostgreSQL JDBC和 ODBC](#)
- [以不同方式建置的功能](#)
- [不支援的 PostgreSQL 功能](#)
- [不支援的 PostgreSQL 資料類型](#)
- [不支援的 PostgreSQL 函數](#)

Amazon Redshift 是以 Postgre 為基礎SQL。Amazon Redshift 和 PostgreSQL 在設計和開發資料倉儲應用程式時，您必須注意許多非常重要的差異。

Amazon Redshift 專為線上分析處理（OLAP）和商業智慧（BI）應用程式而設計，這些應用程式需要對大型資料集進行複雜的查詢。由於它解決了非常不同的需求，Amazon Redshift 使用的特殊資料儲存結構描述和查詢執行引擎與 PostgreSQL 實作完全不同。例如，線上交易處理（OLTP）應用程式通常會將資料以資料列形式存放，Amazon Redshift 會使用特殊的資料壓縮編碼將資料存放於資料欄，以獲得最佳記憶體用量和磁碟 I/O。某些 PostgreSQL 功能適用於較小規模的OLTP處理，例如次要索引和高效的單列資料操作操作，已省略以提高效能。

如需 Amazon Redshift 資料倉儲系統架構的詳細說明，請參閱[Amazon Redshift 架構](#)。

PostgreSQL 9.x 包含 Amazon Redshift 不支援的某些功能。此外，您必須注意 Amazon Redshift SQL 和 PostgreSQL 之間的重要差異。本節重點介紹 Amazon Redshift 和 PostgreSQL 之間的差異，並提供開發充分利用 Amazon Redshift SQL實作的資料倉儲指南。

Amazon Redshift 和 PostgreSQL JDBC和 ODBC

由於 Amazon Redshift 是以 Postgre 為基礎SQL，因此先前建議使用 JDBC4 Postgresql 驅動程式 8.4.703 版和 psqIODBC 9.x 版驅動程式。如果您目前正在使用這些驅動程式，我們建議今後移至新

的 Amazon Redshift 專屬驅動程式。如需驅動程式和設定連線的詳細資訊，請參閱 [JDBC Amazon Redshift 管理指南](#) 中的 [和 Amazon Redshift 的 ODBC 驅動程式](#)。

若要避免使用擷取大型資料集時發生用戶端 out-of-memory 錯誤 JDBC，您可以透過設定擷取大小參數，讓用戶端以批次方式 JDBC 擷取資料。如需詳細資訊，請參閱 [設定 JDBC 擷取大小參數](#)。

Amazon Redshift 無法辨識 JDBC maxRows 參數。請改為指定 [LIMIT](#) 子句來限制結果集。您也可以使用 [OFFSET](#) 子句，來跳到結果集中的指定起點。

以不同方式建置的功能

許多 Amazon Redshift SQL 語言元素具有不同的效能特性，並使用語法和語義，與同等 PostgreSQL 實作截然不同。

Important

請勿假設 Amazon Redshift 和 PostgreSQL 共同的元素語義相同。請務必參考《Amazon Redshift 開發人員指南》的 [SQL 命令](#)，以了解細微的差異。

一個特別的例子是 [VACUUM](#) 命令，這是用來清理和重整資料表。VACUUM 功能與 PostgreSQL 版本不同，並使用不同的參數集。如需在 Amazon Redshift VACUUM 中使用的詳細資訊 [清空資料表](#)，請參閱。

資料庫管理的功能和工具也經常有所不同。例如，Amazon Redshift 會維持一組系統資料表和檢視，這些項目提供關於系統運作狀況的資訊。如需更多資訊，請參閱 [SYS 監控檢視](#)。

下列清單包含一些在 Amazon Redshift 中以不同方式實作 SQL 的功能範例。

- [CREATE TABLE](#)

Amazon Redshift 不支援資料表空間、資料表分割、繼承和某些限制。的 Amazon Redshift 實作 CREATE TABLE 可讓您定義資料表的排序和分佈演算法，以最佳化平行處理。

Amazon Redshift Spectrum 支援使用 [CREATE EXTERNAL TABLE](#) 命令來分割資料表。

- [ALTER TABLE](#)

僅 ALTER COLUMN 支援動作子集。

ADD COLUMN 支援在每個 ALTER TABLE 陳述式中僅新增一欄。

- [COPY](#)

Amazon Redshift COPY命令具有高度專門性，可從 Amazon S3 儲存貯體和 Amazon DynamoDB 資料表載入資料，並促進自動壓縮。如需詳細資訊，請參閱 [在 Amazon Redshift 中載入資料](#) 一節和 COPY命令參考。

- [VACUUM](#)

的參數VACUUM完全不同。例如，PostgreSQL 中的預設VACUUM操作只會回收空間，並使其可供重複使用；不過，Amazon Redshift 中的預設VACUUM操作為 VACUUM FULL，這會回收磁碟空間並替代所有資料列。

- 比較字串VARCHAR值時，會忽略值中的追蹤空格。如需詳細資訊，請參閱[多餘空格的意義](#)。

不支援的 PostgreSQL 功能

Amazon Redshift 不支援這些 PostgreSQL 功能。

Important

請勿假設 Amazon Redshift 和 PostgreSQL 共同的元素語義相同。請務必參考《Amazon Redshift 開發人員指南》的 [SQL 命令](#)，以了解細微的差異。

- 不支援查詢工具 psql。支援 [Amazon Redshift RSQL](#) 用戶端。
- 資料表分割 (範圍與列表分割)
- 資料表空間
- 限制
 - 唯一
 - 外部索引鍵
 - 主索引鍵
 - 檢查限制
 - 排除限制

允許唯一、主索引鍵和外部索引鍵的限制，但這些只是參考資訊。系統不會強制執行這些限制，不過查詢規劃器會使用這些限制。

- 資料庫角色

- 繼承
- PostgreSQL 系統資料欄

Amazon Redshift SQL不會隱含定義系統資料欄。不過，下列 PostgreSQL 系統資料欄名稱不能用作使用者定義資料欄的名稱：oid、tableoid、xmin、xminxmax、xmax和ctid。如需詳細資訊，請參閱 <https://www.postgresql.org/docs/8.0/static/ddl-system-columns.html>。

- 索引
- NULLS Window 函數中的子句
- 定序

Amazon Redshift 不支援地區特定或使用者定義的定序句。請參閱 [定序序列](#)。

- 值表達式
 - 註標表達式
 - 陣列建構子
 - 列建構子
- 觸發
- 外部資料的管理 (SQL/MED)
- 資料表函式
- VALUES 用作常數資料表的清單
- 序列
- 全文檢索搜尋
- RULE 和 TRIGGER許可。

當您執行 `GRANT` 或 `REVOKE` 時，Amazon Redshift 會授予 `GRANT ALL` 或撤銷這些許可 `ALL`，但 `RULE` 和 `TRIGGER` 許可的存在與否不會以任何方式影響受讓者的存取許可。

不支援的 PostgreSQL 資料類型

一般而言，如果查詢嘗試使用不支援的資料類型 (包括明確或隱含轉換)，將會傳回錯誤。不過，使用不支援資料類型的某些查詢，將可在領導節點上執行，但無法在運算節點上執行。請參閱 [SQL 領導節點上支援的函數](#)。

如需支援的資料類型清單，請參閱 [資料類型](#)。

Amazon Redshift 不支援這些 PostgreSQL 資料類型。

- 陣列
- BIT, BIT VARYING
- BYTEA
- 複合類型
- 日期/時間類型
- 列舉類型
- 幾何類型
- HSTORE
- JSON
- 網路位址類型
- 數值類型
 - SERIAL, BIGSERIAL, SMALLSERIAL
 - MONEY
- 物件識別碼類型
- 虛擬類型
- 範圍類型
- 特殊字元類型
 - "char" - 單位元組的內部類型 (其中名為 char 的資料類型會用引號括住)。
 - name - 物件名稱的內部類型。

如需這些類型的詳細資訊，請參閱 PostgreSQL 文件中的[特殊字元類型](#)。

- 文字搜尋類型
- TXID_SNAPSHOT
- UUID
- XML

不支援的 PostgreSQL 函數

許多未排除的函式具有不同的語意或用法。例如，某些支援的函式只會在領導節點上執行。此外，某些不支援的函式在領導節點上執行時，不會傳回錯誤。這些函數在某些情況中不會傳回錯誤的事實，不應視為 Amazon Redshift 支援這些函數。

⚠ Important

請勿假設 Amazon Redshift 和 PostgreSQL 共同的元素語義相同。請務必參考《Amazon Redshift 資料庫開發人員指南》的 [SQL 命令](#)，以了解細微的差異。

如需詳細資訊，請參閱[SQL 領導節點上支援的 函數](#)。

Amazon Redshift 不支援這些 PostgreSQL 函數。

- 存取權限查詢函式
- 建議鎖函式
- 彙總函數
 - STRING_AGG()
 - ARRAY_AGG()
 - EVERY()
 - XML_AGG()
 - CORR()
 - COVAR_POP()
 - COVAR_SAMP()
 - REGR_AVGX(), REGR_AVGY()
 - REGR_COUNT()
 - REGR_INTERCEPT()
 - REGR_R2 ()
 - REGR_SLOPE()
 - REGR_SXX(), REGR_SXY(), REGR_SYY()
- 陣列函數和運算子
- 備份控制函式
- 註解資訊函式
- 資料庫物件位置函式
- 資料庫物件大小函式
- 日期/時間函式與運算子
 - CLOCK_TIMESTAMP()

- JUSTIFY_DAYS(), JUSTIFY_HOURS(), JUSTIFY_INTERVAL()
- PG_SLEEP ()
- TRANSACTION_TIMESTAMP()
- ENUM 支援函數
- 幾何函式與運算子
- 通用檔案存取函式
- IS DISTINCT FROM
- 網路位址函式與運算子
- 數學函式
 - DIV()
 - SETSEED()
 - WIDTH_BUCKET()
- 結果集傳回函式
 - GENERATE_SERIES()
 - GENERATE_SUBSCRIPTS()
- 範圍函式與運算子
- 復原控制函式
- 復原資訊函式
- ROLLBACK TO SAVEPOINT函數
- 結構描述可見性查詢函式
- 伺服器發訊函式
- 快照同步函式
- 序列處理函式
- 字串函數
 - BIT_LENGTH()
 - OVERLAY()
 - CONVERT ()、CONVERT_FROM ()、CONVERT_TO ()
 - ENCODE()
 - FORMAT()
 - QUOTE_NULLABLE()

- REGEXP_MATCHES()
- REGEXP_SPLIT_TO_ARRAY ()
- REGEXP_SPLIT_TO_TABLE ()
- 系統目錄資訊函式
- 系統資訊函數
 - CURRENT_CATALOG CURRENT_QUERY()
 - INET_CLIENT_ADDR()
 - INET_CLIENT_PORT()
 - INET_SERVER_ADDR() INET_SERVER_PORT()
 - PG_CONFLOAD_TIME_ ()
 - PG_IS_OTHERTEMP_SCHEMA ()
 - PG_LISTENING_CHANNELS ()
 - PG_MYTEMP_SCHEMA ()
 - PG_POSTMASTERSTART_TIME ()
 - PG_TRIGGER_DEPTH ()
 - SHOW VERSION()
- 文字搜尋函式與運算子
- 交易IDs和快照函數
- 觸發函數
- XML 函數

使用 SQL

主題

- [SQL 參考慣例](#)
- [基本元素](#)
- [表達式](#)
- [條件](#)

SQL 語言包含您用來處理資料庫和資料庫物件的命令和函數。此語言也會針對資料類型、表達式和常值的使用，來強制執行規則。

SQL 參考慣例

本節說明用於撰寫SQL參考章節所述SQL表達式、命令和函數語法的慣例。

字元	描述
CAPS	大寫字詞為關鍵字。
[]	方括號是用來表示選用的引數。方括號中的多個引數，代表您可以選擇任何數目的引數。此外，方括號中不同行的引數，代表 Amazon Redshift 剖析器期望引數的排序，等於在語法中所列出的順序。如需範例，請參閱 SELECT 。
{ }	大括號表示您需要選擇大括號內的其中一個引數。
	直立線符號會將您可選擇的引數隔開。
斜體	斜體的字表示預留位置。您必須插入適當的值來取代斜體字詞。
...	省略符號表示您可以重複前一個元素。
'	單引號中的字詞表示您必須輸入引用內容。

基本元素

主題

- [名稱與識別碼](#)
- [文字](#)
- [Null](#)
- [資料類型](#)
- [定序序列](#)

本節針對資料庫物件名稱、常值、null 及資料類型，說明了使用的規則。

名稱與識別碼

名稱可用來識別資料庫物件，包括資料表和資料欄，以及使用者和密碼。名稱和識別碼這兩個詞可互換使用。識別碼有兩種：標準識別碼和引號 (或區隔) 識別碼。識別符只能包含 UTF-8 個可列印字

元。ASCII 標準和分隔識別碼中的字母不區分大小寫，且在資料庫中折疊為小寫。在查詢結果中，欄的名稱預設會以小寫傳回。若要以大寫傳回欄的名稱，請將 [describe_field_name_in_uppercase](#) 組態參數設定為 **true**。

標準識別碼

標準SQL識別碼遵循一組規則，且必須：

- 從ASCII單位元組字母字元或底線字元，或長度為兩到四個位元組的 UTF-8 個多位元組字元開始。
- 後續字元可以是ASCII單位元組英數字元、底線或美元符號，或長度為兩到四個位元組的 UTF-8 個多位元組字元。
- 讓長度介於 1 到 127 個位元組之間，不包括區隔識別碼的引號。
- 不包含引號和空格。
- 不是保留SQL的關鍵字。

區隔識別碼

區隔識別碼 (也稱為引號識別碼) 是以雙引號 (") 開頭和結尾。如果使用區隔識別碼，則每次參考該物件，都必須使用雙引號。識別符可以包含雙引號本身以外的任何標準 UTF-8 個可列印字元。因此，您可以建立欄或資料表的名稱，其中包含無效字元，例如空格或百分比符號。

ASCII 分隔識別碼中的字母不區分大小寫，並摺疊為小寫。若要在字串中使用雙引號，必須在其前面加上另一個雙引號字元。

區分大小寫的識別碼

區分大小寫的識別碼 (也稱為混合大小寫識別碼) 可以同時包含大寫和小寫字母。若要使用區分大小寫的識別碼，您可以將組態 `enable_case_sensitive_identifier` 設定為 `true`。您可以為叢集或工作階段設定此組態。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [預設參數值](#) 和 [enable_case_sensitive_identifier](#)。

系統欄名稱

下列 PostgreSQL 系統資料欄名稱無法用作使用者定義資料欄中的資料欄名稱。如需詳細資訊，請參閱 <https://www.postgresql.org/docs/8.0/static/ddl-system-columns.html>。

- `oid`
- `tableoid`

- xmin
- cmin
- xmax
- cmax
- ctid

範例

下表顯示區隔識別碼的範例、產生的輸出和說明：

語法	結果	說明
"group"	群組	GROUP 是保留單字，因此在識別符內使用該單字需要雙引號。
""WHERE""	"where"	WHERE 也是保留字。若要在字串中包含雙引號，請使用額外的雙引號字元，來逸出每個雙引號字元。
"This name"	this name	必須使用雙引號來保留空格。
"This ""IS IT""	this "is it"	包夾 IS IT 的引號，前後必須再加上另一個引號，才能成為名稱的一部分。

若要建立名為 group 的資料表，而且此資料表包含名為 this "is it" 的欄：

```
create table "group" (
  "This ""IS IT"" char(10));
```

下列的查詢會傳回相同的結果：

```
select "This ""IS IT""
from "group";

this "is it"
-----
(0 rows)
```

```
select "this ""is it""
from "group";

this "is it"
-----
(0 rows)
```

下列的完整 `table.column` 語法，也會傳回相同的結果：

```
select "group"."this ""is it""
from "group";

this "is it"
-----
(0 rows)
```

下列 `CREATETABLE` 命令會建立資料欄名稱中具有斜線的資料表：

```
create table if not exists city_slash_id(
    "city/id" integer not null,
    state char(2) not null);
```

文字

常值或常數是固定的資料值，由一系列的字元或數值常數構成。Amazon Redshift 支援多種類型的常值，包括：

- 整數的數值常值、小數和符點數。如需詳細資訊，請參閱 [整數常值與浮點字面值](#)。
- 字元常值，也稱為字串、字元字串或字元常數
- 日期時間 (datetime) 與間隔常值，搭配 datetime 資料類型使用。如需詳細資訊，請參閱 [日期、時間和時間戳記常值](#) 和 [間隔資料類型和常值](#)。

Null

如果列所在的欄遺漏、未知或不適用，則此欄為 null 值，或是包含 null。Null 會出現在不受主索引鍵或 NOTNULL 限制條件限制的任何資料類型欄位中。Null 不等於 0 值，也不等於空字串。

任何包含 null 的數學表達式，所求得的結果一律為 null。給予 null 引數或運算元時，除了串連以外的所有運算子，都會傳回 null。

若要測試 null，請使用比較條件 IS NULL 和 IS NOT NULL。由於 null 代表缺少資料，因此 null 不等於任何值，也不等於另一個 null。

資料類型

主題

- [多位元組字元](#)
- [數值類型](#)
- [字元類型](#)
- [日期時間 \(Datetime\) 類型](#)
- [布林值 \(Boolean\) 類型](#)
- [HLLSKETCH 類型](#)
- [SUPER 類型](#)
- [VARBYTE 類型](#)
- [類型相容性與轉換](#)

Amazon Redshift 所儲存或擷取的每個值，都具有資料類型，而資料類型具有一組固定的相關屬性。資料類型會在資料表建立時宣告，用來限制欄或引數可包含的一組值。

下表列出您可以在 Amazon Redshift 中使用的資料類型。

資料類型	Aliases	描述
SMALLINT	INT2	帶正負號的 2 位元組整數
INTEGER	INT, INT4	帶正負號的 4 位元組整數
BIGINT	INT8	帶正負號的 8 位元組整數
DECIMAL	NUMERIC	可選擇精確度 (有效位數) 的精確數值
REAL	FLOAT4	單精度浮點數
DOUBLE PRECISION	FLOAT8, FLOAT	雙精度浮點數

資料類型	Aliases	描述
CHAR	CHARACTER, NCHAR, BPCHAR	固定長度的字元字串
VARCHAR	CHARACTER VARYING, NVARCHAR, TEXT	可變長度的字元字串 (使用者定義的限制)
DATE		日曆日期 (年、月、日)
TIME	TIME WITHOUT TIME ZONE	一天中的時間
TIMETZ	TIME WITH TIME ZONE	一天中的時間，含時區
TIMESTAMP	TIMESTAMP WITHOUT TIME ZONE	日期和時間 (未使用時區)
TIMESTAMPTZ	TIMESTAMP WITH TIME ZONE	日期和時間 (包含時區)
INTERVAL YEAR 至 MONTH		以年到月順序表示的持續時間
INTERVAL DAY 至 SECOND		以天到第二個順序表示的持續時間
BOOLEAN	BOOL	邏輯布林值 (true/false)
HLLSKETCH		與草圖搭配使用 HyperLogLog 的類型。
SUPER		超集資料類型，包含所有純量類型的 Amazon Redshift，包括複雜的類型，例如 ARRAY 和 STRUCTS。
VARBYTE	VARBINARY, BINARY VARYING	可變長度二進位值
GEOMETRY		空間資料

資料類型	Aliases	描述
GEOGRAPHY		空間資料

Note

如需 "char" (請注意，char 會用引號括住) 等不支援資料類型的相關資訊，請參閱[不支援的 PostgreSQL 資料類型](#)。

多位元組字元

VARCHAR 資料類型支援最多 4 個位元組的 UTF-8 個多位元組字元。不支援 5 個位元組或更長的字元。若要計算包含多位元組字元的資料 VARCHAR 欄大小，請將字元數乘以每個字元的位元組數。例如，如果字串有四個中文字元，且每個字元長度為三個位元組，則您將需要一 VARCHAR (12) 欄來存放字串。

VARCHAR 資料類型不支援下列無效的 UTF-8 碼點：

0xD800 - 0xDFFF (位元組序列：ED A0 80 - ED BF BF)

CHAR 資料類型不支援多位元組字元。

數值類型

主題

- [整數類型](#)
- [DECIMAL 或 NUMERIC 類型](#)
- [有關使用 128 位元 DECIMAL 或 NUMERIC 資料欄的備註](#)
- [浮點類型](#)
- [數值的計算](#)
- [整數常值與浮點字面值](#)
- [數值類型範例](#)

數值資料類型包括整數、小數和符點數。

整數類型

使用 SMALLINT、INTEGER 和 BIGINT 資料類型來存放各種範圍的整數。您不能儲存超出每種類型允許範圍的值。

名稱	儲存	範圍
SMALLINT 或 INT2	2 位元組	-32768 到 +32767
INTEGER、INT 或 INT4	4 位元組	-2147483648 到 +2147483647
BIGINT 或 INT8	8 位元組	-9223372036854775808 到 9223372036854775807

DECIMAL 或 NUMERIC 類型

使用 DECIMAL 或 NUMERIC 資料類型，以使用者定義的精確度儲存值。DECIMAL 和 NUMERIC 關鍵字可互換。在本文件中，小數是此資料類型的首選用詞。數值一詞通常是用來指稱整數、小數和浮點資料類型。

儲存	範圍
變數，未壓縮 DECIMAL 類型最多 128 位元。	128 位元帶正負號的整數，具備最高 38 個位數的精確度。

透過指定精確度和規模，在資料表中定義資料 DECIMAL 欄：

```
decimal(precision, scale)
```

precision

整個值中有效位數的總數：小數點兩邊的位數數量。例如，數字 48.2891 的精確度 (有效位數) 為 6，小數位數為 4。如果未指定，預設的精確度為 18，最高精確度為 38。

如果在輸入值中，小數點左邊的位數數目，超過資料欄的精確度減去其小數位數，就無法將此值複製 (或插入或更新) 到資料欄中。此規則適用於超出資料欄定義範圍之外的任何值。例如，numeric(5,2) 欄的值，其允許的範圍為 -999.99 到 999.99。

scale

數值小數部分中，位於小數點右邊的小數位數數目。整數的小數位數為 0。在資料欄的規格中，小數位數的值必須小於或等於精確度的值。如果未指定，預設的小數位數為 0，最大的小數位數為 37。

如果載入資料表的輸入值，其小數位數大於資料欄的小數位數，則此值會四捨五入至指定的小數位數。例如，SALES資料表中的資料PRICEPAID欄是 DECIMAL (8 , 2) 資料欄。如果將 DECIMAL (8 , 4) 值插入資料PRICEPAID欄，則值會四捨五入至 2 的尺度。

```
insert into sales
values (0, 8, 1, 1, 2000, 14, 5, 4323.8951, 11.00, null);

select pricepaid, salesid from sales where salesid=0;

pricepaid | salesid
-----+-----
4323.90 |      0
(1 row)
```

不過，從資料表所選取值的明確轉換結果，不會四捨五入。

Note

您可以插入 DECIMAL (19 , 0) 欄的最大正值為 9223372036854775807 ($2^{63} - 1$)。負數值上限為 -9223372036854775808。例如，如果試圖插入數值 9999999999999999999 (19 個 9)，將會造成溢位錯誤。無論小數點的位置為何，Amazon Redshift 可以表示為DECIMAL數字的最大字串為 9223372036854775807。例如，您可以載入 DECIMAL (19 , 18) 資料欄的最大值為 9.223372036854775807。這些規則是由於具有 19 個或更少有效位數的精度DECIMAL值會以 8 位元組整數的形式儲存在內部，而具有 20 到 38 個有效位數的精度DECIMAL值則以 16 位元組整數的形式儲存。

有關使用 128 位元DECIMAL或NUMERIC資料欄的備註

除非您確定應用程式需要該精確度，否則請勿任意將最大精確度指派給DECIMAL資料欄。128 位元值使用的磁碟空間是 64 位元值的兩倍，而且可以縮短查詢執行時間。

浮點類型

使用 REAL和 DOUBLE PRECISION 資料類型來儲存具有變數精確度的數值。這些是不精確的類型，代表某些數值會以近似值儲存，因此在儲存和傳回特定值時，可能會造成些微的出入。如果您需要確切的儲存和計算（例如的貨幣金額），請使用 DECIMAL 資料類型。

REAL 根據二進位浮點數運算IEEE的標準 754，表示單精度浮點格式。它具有大約 6 位數的精確度，並且範圍約為 1E-37 到 1E+37。您也可以將此資料類型指定為 FLOAT4。

DOUBLE PRECISION 根據二進位浮點運算IEEE的標準 754，表示雙精度浮點格式。它具有大約 15 位數的精確度，並且範圍約為 1E-307 到 1E+308。您也可以將此資料類型指定為 FLOAT或 FLOAT8。

除了普通的數值之外，浮點類型還有幾個特殊值。在 中使用單引號時，請在這些值周圍使用它們 SQL：

- NaN – not-a-number
- Infinity— 無窮大
- -Infinity— 負無窮大

例如，若要在資料表day_charge的 欄中插入 not-a-number，請customer_activity執行下列 SQL：

```
insert into customer_activity(day_charge) values('NaN');
```

數值的計算

在本文中，計算是指二進位數學運算：加、減、乘和除。本節說明這些操作的預期傳回類型，以及用於在涉及DECIMAL資料類型時判斷精確度和規模的特定公式。

在查詢處理作業期間計算數值時，可能會遇到無法進行計算的情況，而且查詢會傳回數值溢位錯誤。您也可能會遇到計算值的小數位數改變或出乎意料的情況。針對某些運算，您可以使用明確轉換 (類型提升) 或 Amazon Redshift 設定參數，來解決這些問題。

如需使用 SQL函數進行類似運算的結果資訊，請參閱 [彙總函數](#)。

計算的傳回類型

根據 Amazon Redshift 中所支援的一組數值資料類型，下表顯示了加法、減法、乘法和除法運算預期的傳回類型。表格左側的第一欄代表計算中的第一個運算元，最上面的列代表第二個運算元。

運算元 1	運算元 2	傳回類型
INT2	INT2	INT2
INT2	INT4	INT4
INT2	INT8	INT8
INT2	DECIMAL	DECIMAL
INT2	FLOAT4	FLOAT8
INT2	FLOAT8	FLOAT8
INT4	INT4	INT4
INT4	INT8	INT8
INT4	DECIMAL	DECIMAL
INT4	FLOAT4	FLOAT8
INT4	FLOAT8	FLOAT8
INT8	INT8	INT8
INT8	DECIMAL	DECIMAL
INT8	FLOAT4	FLOAT8
INT8	FLOAT8	FLOAT8
DECIMAL	DECIMAL	DECIMAL
DECIMAL	FLOAT4	FLOAT8
DECIMAL	FLOAT8	FLOAT8

運算元 1	運算元 2	傳回類型
FLOAT4	FLOAT8	FLOAT8
FLOAT8	FLOAT8	FLOAT8

計算DECIMAL結果的精確度和規模

下表摘要說明計算所得精確度的規則，並在數學運算傳回DECIMAL結果時擴展。在這個表格中，p1 和 s1 代表計算式中第一個運算元的精確度和小數位數，p2 和 s2 代表第二個運算元的精確度和小數位數。(無論這些計算如何，結果的最高精確度為 38、結果的最大小數位數為 38。)

作業	結果的精確度與小數位數
+ 或 -	擴展 = $\max(s1, s2)$ 精確度 = $\max(p1-s1, p2-s2)+1+scale$
*	擴展 = $s1+s2$ 精確度 = $p1+p2+1$
/	擴展 = $\max(4, s1+p2-s2+1)$ 精確度 = $p1-s1+ s2+scale$

例如，SALES資料表中的 PRICEPAID和 COMMISSION資料欄都是 DECIMAL (8 , 2) 資料欄。如果您PRICEPAID除以 COMMISSION (反之亦然) ，則會套用公式，如下所示：

```
Precision = 8-2 + 2 + max(4,2+8-2+1)
= 6 + 2 + 9 = 17
```

```
Scale = max(4,2+8-2+1) = 9
```

```
Result = DECIMAL(17,9)
```

下列計算是計算對 UNION、 和 等設定運算子EXCEPT或 INTERSECT和 等函數對DECIMAL值執行之操作所產生的精確度COALESCE和規模的一般規則DECODE：

```
Scale = max(s1,s2)
Precision = min(max(p1-s1,p2-s2)+scale,19)
```

例如，具有一 DECIMAL (7 , 2) 欄的DEC1資料表會與具有一 DECIMAL (15 , 3) 欄的DEC2資料表聯結，以建立DEC3資料表。的結構描述DEC3顯示資料欄變成 NUMERIC (15 , 3) 資料欄。

```
create table dec3 as select * from dec1 union select * from dec2;
```

結果

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'dec3';
```

column	type	encoding	distkey	sortkey
c1	numeric(15,3)	none	f	0

在上述的範例中，會如下套用公式：

```
Precision = min(max(7-2,15-3) + max(2,3), 19)
= 12 + 3 = 15
```

```
Scale = max(2,3) = 3
```

```
Result = DECIMAL(15,3)
```

關於除法運算的備註

對於部門操作， divide-by-zero條件會傳回錯誤。

在計算出精確度和小數位數之後，會套用 100 個小數位數的限制。如果計算結果的小數位數大於 100，除的結果會如下設定小數位數：

- 精確度 = precision - (scale - max_scale)
- 擴展 = max_scale

如果計算出的精確度大於最高精確度 (38)，則精確度會降低為 38，而小數位數會變成下列算式的結果：
 $\max((38 + \text{scale} - \text{precision}), \min(4, 100))$

溢位狀況

會針對所有數值運算檢查溢位。DECIMAL 精確度為 19 或以下的資料會儲存為 64 位元整數。DECIMAL 精確度大於 19 的資料會儲存為 128 位元整數。所有DECIMAL值的最大精確度為 38，最大規模為 37。當數值超出這些限值時，會出現溢位錯誤，這會同時發生於中間的和最終的結果集：

- 當特定資料值不符合轉換函式所要求的精確度或指定的小數位數時，明確轉換作業會產生執行時期的溢位錯誤。例如，您無法從SALES資料表（DECIMAL8，2）欄）中的PRICEPAID資料欄轉換所有值，並傳回 DECIMAL（7，3）結果：

```
select pricepaid::decimal(7,3) from sales;
ERROR: Numeric data overflow (result precision)
```

發生此錯誤是因為無法轉換資料PRICEPAID欄中某些較大的值。

- 在乘法運算所產生的結果中，其小數位數是每個運算元小數位數的總和。例如，如果兩個運算元都有 4 個小數位數，結果的小數位數是 8 個，使得小數點的左邊只有 10 個位數。因此，在將兩個都擁有大量小數位數的大數值相乘時，就會相當容易產生溢位狀況。

下列範例會導致溢位錯誤。

```
SELECT CAST(1 AS DECIMAL(38, 20)) * CAST(10 AS DECIMAL(38, 20));
ERROR: 128 bit numeric data overflow (multiplication)
```

您可以使用除法而不是乘法來解決溢位錯誤。使用以下範例除以 1 除以原始除數。

```
SELECT CAST(1 AS DECIMAL(38, 20)) / (1 / CAST(10 AS DECIMAL(38, 20)));
+-----+
| ?column? |
+-----+
| 10      |
+-----+
```

使用 INTEGER和 DECIMAL類型的數值計算

當計算中的其中一個運算元具有INTEGER資料類型，而另一個運算元為時DECIMAL，INTEGER運算元會隱含轉換為 DECIMAL：

- INT2（SMALLINT）轉換為 DECIMAL（5，0）
- INT4（INTEGER）轉換為 DECIMAL（10，0）

- INT8 (BIGINT) 轉換為 DECIMAL (19 , 0)

例如，如果您將 SALES.COMMISSION、DECIMAL (8 , 2) 欄和 SALES. QTY SOLD SMALLINT 欄相乘，則此計算會轉換為：

```
DECIMAL(8,2) * DECIMAL(5,0)
```

整數常值與浮點字面值

表示數字的常值或常數，可以是整數或浮點數。

整數常值

整數常值是數字 0 – 9 的序列，在數字前面可以選擇性地加上正號 (+) 或負號 (-)。

語法

```
[ + | - ] digit ...
```

範例

有效的整數包括下列範例：

```
23  
-555  
+17
```

浮點字面值

浮點字面值 (也稱為小數、數值或分數常值) 是數字的序列，可包含小數點，以及選擇性的包含指數標記。

語法

```
[ + | - ] digit ... [ . ] [ digit ... ]  
[ e | E [ + | - ] digit ... ]
```

引數

e | E

e 或 E 代表該數字是以科學記號指定。

範例

有效的浮點字面值包括下列範例：

```
3.14159
-37.
2.0e19
-2E-19
```

數值類型範例

CREATE TABLE 陳述式

下列CREATETABLE陳述式示範不同數值資料類型的宣告：

```
create table film (
  film_id integer,
  language_id smallint,
  original_language_id smallint,
  rental_duration smallint default 3,
  rental_rate numeric(4,2) default 4.99,
  length smallint,
  replacement_cost real default 25.00);
```

試圖插入超出範圍的整數

下列範例會嘗試將值 33000 插入資料SMALLINT欄。

```
insert into film(language_id) values(33000);
```

的範圍SMALLINT是 -32768 至 +32767，因此 Amazon Redshift 傳回錯誤。

```
An error occurred when executing the SQL command:
insert into film(language_id) values(33000)

ERROR: smallint out of range [SQL State=22003]
```

將小數值插入整數資料欄

下列範例會將十進位值插入資料INT欄。

```
insert into film(language_id) values(1.5);
```

會插入此數值，但四捨五入為整數值 2。

成功地插入小數，因為其小數位數已經過四捨五入

下列的範例所插入的小數值，具有高於資料欄的精確度。

```
insert into film(rental_rate) values(35.512);
```

在此例中，將 35.51 這個值插入了資料欄。

試圖插入超出範圍的小數值

在此例中，350.10 這個值超出範圍。DECIMAL 資料欄中值的位數等於資料欄的精確度減去其規模 (RENTAL_RATE 資料欄為 4 減去 2)。換句話說，DECIMAL(4,2) 資料欄允許的範圍是 -99.99 到 99.99。

```
insert into film(rental_rate) values (350.10);
ERROR: numeric field overflow
DETAIL: The absolute value is greater than or equal to 10^2 for field with precision
4, scale 2.
```

將變數精確度值插入資料REAL欄

下列範例會將變數精確度值插入資料REAL欄。

```
insert into film(replacement_cost) values(1999999.99);

insert into film(replacement_cost) values(1999.99);

select replacement_cost from film;

+-----+
| replacement_cost |
+-----+
| 2000000          |
| 1999.99          |
+-----+
```

1999999.99 值會轉換為 2000000，以符合 REAL 欄的精確度要求。數值 1999.99 會依原狀載入。

字元類型

主題

- [儲存與範圍](#)
- [CHAR 或 CHARACTER](#)
- [VARCHAR 或 CHARACTER VARYING](#)
- [NCHAR 和 NVARCHAR 類型](#)
- [TEXT 和 BPCHAR 類型](#)
- [多餘空格的意義](#)
- [字元類型範例](#)

字元資料類型包括 CHAR (字元) 和 VARCHAR (字元變化) 。

儲存與範圍

CHAR 和 VARCHAR 資料類型是以位元組為單位定義，而非字元。資料CHAR欄只能包含單位元組字元，因此 CHAR (10) 資料欄可以包含長度上限為 10 個位元組的字串。VARCHAR 可包含多位元組字元，每個字元最多四個位元組。例如，VARCHAR (12) 欄可以包含 12 個單位元組字元、6 個雙位元組字元、4 個三位元組字元或 3 個四位元組字元。

名稱	儲存	範圍 (資料欄的寬度)
CHAR、 CHARACTER或 NCHAR	字串的長度，包括多餘的空格 (如果有的話)	4096 位元組
VARCHAR、 CHARACTER VARYING或 NVARCHAR	4 位元組 + 字元的總位元組數，其中每個字元都可以是 1 到 4 個位元組。	65535 位元組 (64K -1)
BPCHAR	已轉換為固定長度 CHAR (256) 。	256 位元組

名稱	儲存	範圍 (資料欄的寬度)
TEXT	已轉換為 VARCHAR (256) 。	260 位元組

Note

CREATE TABLE 語法支援字元資料類型的MAX關鍵字。例如：

```
create table test(col1 varchar(max));
```

MAX 設定會將欄的寬度定義為 4096 位元組，CHAR或 65535 位元組VARCHAR。

CHAR 或 CHARACTER

使用 CHAR或 CHARACTER欄來存放固定長度字串。這些字串會加上空白，因此 CHAR (10) 欄一律佔用 10 個位元組的儲存空間。

```
char(10)
```

沒有長度規格的資料CHAR欄會產生 CHAR (1) 資料欄。

VARCHAR 或 CHARACTER VARYING

使用 VARCHAR或 CHARACTERVARYING欄來存放具有固定限制的變數長度字串。這些字串不會加上空白，因此 VARCHAR (120) 欄最多包含 120 個單位元組字元、60 個雙位元組字元、40 個三位元組字元或 30 個四位元組字元。

```
varchar(120)
```

如果您在CREATETABLE陳述式中使用沒有長度規格符的VARCHAR資料類型，預設長度為 256。如果在運算式中使用，系統會使用輸入運算式來確定輸出的大小 (最多 65535)。

NCHAR 和 NVARCHAR 類型

您可以使用 NCHAR 和 NVARCHAR 類型（也稱為 NATIONALCHARACTER 和 NATIONALCHARACTERVARYING 類型）建立資料欄。這些類型分別轉換為 CHAR 和 VARCHAR 類型，並儲存在指定的位元組數中。

沒有長度規格 NCHAR 的欄會轉換為 CHAR (1) 欄。

沒有長度規格的 NVARCHAR 欄會轉換為 VARCHAR (256) 欄。

TEXT 和 BPCHAR 類型

您可以使用資料 TEXT 欄建立 Amazon Redshift 資料表，但會轉換為接受變數長度值的 VARCHAR (256) 資料欄，最多 256 個字元。

您可以建立 Amazon Redshift 資料欄，其中使用 BPCHAR (空白填充字元) 類型，Amazon Redshift 會轉換為固定長度 CHAR (256) 資料欄。

多餘空格的意義

CHAR 和 VARCHAR 資料類型最多可以存放長度為 n 個位元組的字串。如果試圖將較長的字串，儲存到具有這些類型的資料欄中，將會造成錯誤 (除非多出來的字串全部都是空格 (空白)，此時字串會被截斷至最大長度)。如果字串短於長度上限，則 CHAR 值會填入空白，但 VARCHAR 值會存放字串而不留空白。

CHAR 值中的追蹤空白一律不明顯。當您比較兩個值，不包含在 LENGTH 計算中，並在將 CHAR 值轉換為 CHAR 其他字串類型時將其移除時，它們會被忽略。

追蹤 VARCHAR 和 CHAR 值中的空格在比較值時被視為不明顯。

長度計算會傳回長度中包含尾端空格的 VARCHAR 字元字串長度。多餘的空格不會列入固定長度字元字串的長度計算。

字元類型範例

CREATE TABLE 陳述式

下列 CREATE TABLE 陳述式示範 VARCHAR 和 CHAR 資料類型的使用：

```
create table address(  
address_id integer,
```

```
address1 varchar(100),
address2 varchar(50),
district varchar(20),
city_name char(20),
state char(2),
postal_code char(5)
);
```

下列的範例使用此資料表。

可變長度字元字串中的多餘空格

因為 ADDRESS1 是資料 VARCHAR 欄，所以第二個插入地址中的尾端空白不明顯。換句話說，下列這兩個插入的地址是相符合的。

```
insert into address(address1) values('9516 Magnolia Boulevard');
insert into address(address1) values('9516 Magnolia Boulevard ');
```

```
select count(*) from address
where address1='9516 Magnolia Boulevard';
```

```
count
-----
2
(1 row)
```

如果 ADDRESS1 資料欄是資料 CHAR 欄，且已插入相同的值，則 COUNT (*) 查詢會識別相同字元字串，並傳回 2。

LENGTH 函數的結果

LENGTH 函數會識別 VARCHAR 欄中的尾隨空白：

```
select length(address1) from address;
```

```
length
-----
23
25
(2 rows)
```

CITY_Augusta NAME 資料欄中的值，即CHAR資料欄，無論輸入字串中的任何後置空白，一律會傳回 7 個字元的長度。

超過資料欄長度的值

字元字串不會為了配合資料欄宣告的寬度而遭到截斷：

```
insert into address(city_name) values('City of South San Francisco');
ERROR: value too long for type character(20)
```

這個問題的解決方法，是將值轉換為符合資料欄的大小：

```
insert into address(city_name)
values('City of South San Francisco'::char(20));
```

在這個例子中，字串 (City of South San Fr) 的前 20 個字元會載入資料欄。

日期時間 (Datetime) 類型

主題

- [儲存與範圍](#)
- [DATE](#)
- [TIME](#)
- [TIMETZ](#)
- [TIMESTAMP](#)
- [TIMESTAMPTZ](#)
- [日期時間 \(Datetime\) 類型範例](#)
- [日期、時間和時間戳記常值](#)
- [間隔資料類型和常值](#)

日期時間資料類型包括 DATE、TIME、TIMESTAMP、TIMETZ和 TIMESTAMPTZ。

儲存與範圍

名稱	儲存	範圍	解析度
DATE	4 位元組	4713 BC 到 294276 AD	1 天

名稱	儲存	範圍	解析度
TIME	8 位元組	00:00:00 至 24:00:00	1 毫秒
TIMETZ	8 位元組	00:00:00+1459 至 00:00:00+1459	1 毫秒
TIMESTAMP	8 位元組	4713 BC 到 294276 AD	1 毫秒
TIMESTAMP TZ	8 位元組	4713 BC 到 294276 AD	1 毫秒

DATE

使用 DATE 資料類型來存放不含時間戳記的簡單行事曆日期。

TIME

TIME 是 的別名TIMEWITHOUTTIMEZONE。

使用 TIME 資料類型來儲存一天的時間。

TIME 資料欄最多可儲存六位數精度的值，以用於分數秒。

根據預設，使用者資料表和 Amazon Redshift 系統資料表中的TIME值皆為國際標準時間（UTC）。

TIMETZ

TIMETZ 是 的別名TIMEWITHTIMEZONE。

使用 TIMETZ 資料類型來存放具有時區的每日時間。

TIMETZ 資料欄最多可儲存六位數精度的值，以用於分數秒。

依預設，TIMETZ 值位於使用者資料表和 Amazon Redshift 系統資料表UTC中。

TIMESTAMP

TIMESTAMP 是 的別名TIMESTAMPWITHOUTTIMEZONE。

使用 TIMESTAMP 資料類型來儲存完整的時間戳記值，其中包含日期和時間。

TIMESTAMP 資料欄最多可儲存六位數精度的值，以用於分數秒。

如果您將日期插入資料TIMESTAMP欄，或具有部分時間戳記值的日期，則值會隱含地轉換為完整的時間戳記值。此完整時間戳記值對於缺少的小時、分鐘和秒具有預設值 (00)。輸入字串中的時區值會遭到忽略。

依預設，TIMESTAMP 值位於使用者資料表和 Amazon Redshift 系統資料表UTC中。

TIMESTAMPTZ

TIMESTAMPTZ 是 的別名TIMESTAMPWITHTIMEZONE。

使用TIMESTAMPTZ資料類型輸入完整的時間戳記值，包括日期、一天中的時間和時區。當輸入值包含時區時，Amazon Redshift 會使用時區將值轉換為 UTC 並儲存該UTC值。

若要查看受支援時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

若要查看受支援時區縮寫的清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

您也可以在此時區[IANA資料庫](#)中找到時區的最新資訊。

下表提供時區格式的範例。

格式	範例
dd mon hh:mi:ss yyyy tz	1997 年 12 月 17 日 07 : 37 : 16 PST
mm/dd/yyyy hh:mi:ss.ss tz	12/17/1997 07 : 37 : 16.00 PST
mm/dd/yyyy hh:mi:ss.ss tz	12/17/1997 07:37:16.00 美國時間/太平洋時區
yyyy-mm-dd hh : mi : ss+/-tz	1997-12-17 07:37:16-08
dd.mm.yyyy hh:mi:ss tz	17.12.1997 07 : 37 : 16.00 PST

TIMESTAMPTZ 資料欄可儲存最多六位數的精確度值，以用於分數秒。

如果您將日期插入資料TIMESTAMPTZ欄，或具有部分時間戳記的日期，則值會隱含轉換為完整的时间戳記值。此完整時間戳記值對於缺少的小時、分鐘和秒具有預設值 (00)。

TIMESTAMPTZ 值位於使用者資料表UTC中。

日期時間 (Datetime) 類型範例

接下來，您可以找到使用 Amazon Redshift 支援的日期時間類型的範例。

日期範例

以下範例插入具有不同格式的日期並顯示輸出。

```
create table datetable (start_date date, end_date date);
```

```
insert into datetable values ('2008-06-01','2008-12-31');
```

```
insert into datetable values ('Jun 1,2008','20081231');
```

```
select * from datetable order by 1;
```

```
start_date | end_date  
-----  
2008-06-01 | 2008-12-31  
2008-06-01 | 2008-12-31
```

如果您將時間戳記值插入資料DATE欄，則會忽略該時間部分，而且只會載入日期。

時間範例

下列範例插入具有不同格式和顯示輸出TIMETIMETZ的值。

```
create table timetable (start_time time, end_time timetz);
```

```
insert into timetable values ('19:11:19','20:41:19 UTC');
```

```
insert into timetable values ('191119', '204119 UTC');
```

```
select * from timetable order by 1;
```

```
start_time | end_time  
-----
```

```
19:11:19 | 20:41:19+00
19:11:19 | 20:41:19+00
```

時間戳記範例

如果您將日期插入 `TIMESTAMP` 或 `TIMESTAMPTZ` 資料欄，則時間預設為午夜。例如，如果插入常值 `20081231`，則儲存的值為 `2008-12-31 00:00:00`。

若要變更目前工作階段的時區，請利用 `SET` 指令來設定 `timezone` 組態參數。

下列範例會插入具有不同格式的時間戳記，並顯示產生的資料表。

```
create table tstamp(timeofday timestamp, timeofdaytz timestamptz);

insert into tstamp values('Jun 1,2008 09:59:59', 'Jun 1,2008 09:59:59 EST' );
insert into tstamp values('Dec 31,2008 18:20', 'Dec 31,2008 18:20');
insert into tstamp values('Jun 1,2008 09:59:59 EST', 'Jun 1,2008 09:59:59');

SELECT * FROM tstamp;
```

```
+-----+-----+
|      timeofday      |      timeofdaytz      |
+-----+-----+
| 2008-06-01 09:59:59 | 2008-06-01 14:59:59+00 |
| 2008-12-31 18:20:00 | 2008-12-31 18:20:00+00 |
| 2008-06-01 09:59:59 | 2008-06-01 09:59:59+00 |
+-----+-----+
```

日期、時間和時間戳記常值

以下是 Amazon Redshift 所支援日期、時間和時間戳記常值的規則。

日期

下列輸入日期是您可以載入 Amazon Redshift 資料表 `DATE` 的資料類型之常值日期值的所有有效範例。假設預設 `MDY` `DateStyle` 模式有效。此模式表示在字串中月份值位於日期值之前，例如 `1999-01-08` 和 `01/02/00`。

Note

載入資料表時，日期或時間戳記常值必須用引號括住。

輸入的日期	完整日期
January 8, 1999	January 8, 1999
1999-01-08	January 8, 1999
1/8/1999	January 8, 1999
01/02/00	2000 年 1 月 2 日
2000-Jan-31	2000 年 1 月 31 日
Jan-31-2000	2000 年 1 月 31 日
31-Jan-2000	2000 年 1 月 31 日
20080215	2008 年 2 月 15 日
080215	2008 年 2 月 15 日
2008.366	2008 年 12 月 31 日 (日期的 3 位數部分必須介於 001 到 366 之間)

Times

下列輸入時間都是您可以載入 TIME 和 TIMETZ 資料類型的常值時間值的有效範例，這些值可載入 Amazon Redshift 資料表。

輸入時間	說明 (時間的部分)
04:05:06.789	4:05 AM 又 6.789 秒
04 : 05 : 06	4:05 AM 又 6 秒
04 : 05	4:05 AM 整
040506	4:05 AM 又 6 秒
04:05 AM	4:05 AM 整 ; AM 為選用

輸入時間	說明 (時間的部分)
04:05 PM	4:05 PM 整 ; 小時值必須小於 12。
16 : 05	4:05 PM 整

時間戳記

下列輸入時間戳記是您可以載入 Amazon Redshift 資料表的 `TIMESTAMP` 和 `TIMESTAMPTZ` 資料類型之常值時間值的所有有效範例。有效的日期常值全都可以和下列的時間常值合併。

輸入的時間戳記 (串接的日期和時間)	說明 (時間的部分)
20080215 04:05:06.789	4:05 AM 又 6.789 秒
20080215 04:05:06	4:05 AM 又 6 秒
20080215 04:05	4:05 AM 整
20080215 040506	4:05 AM 又 6 秒
20080215 04:05 AM	4:05 AM 整 ; AM 為選用
20080215 04:05 PM	4:05 PM 整 ; 小時值必須小於 12。
20080215 16:05	4:05 PM 整
20080215	午夜 (預設)

特殊的日期時間 (Datetime) 值

下列的特殊值可做為日期時間 (datetime) 常值和日期函式的引數使用。這些值需使用單引號，而且會在查詢處理作業進行期間，轉換為一般的時間戳記值。

特殊值	描述
<code>now</code>	轉換為目前交易的開始時間，並傳回毫秒精確度的時間戳記。

特殊值	描述
today	轉換為適當的日期，並傳回時間戳記，其中時間的部分全部以 0 表示。
tomorrow	轉換為適當的日期，並傳回時間戳記，其中時間的部分全部以 0 表示。
yesterday	轉換為適當的日期，並傳回時間戳記，其中時間的部分全部以 0 表示。

下列範例顯示 `now` 和 `today` 如何使用 `DATEADD` 函數。

```
select dateadd(day,1,'today');
```

```
date_add
```

```
-----  
2009-11-17 00:00:00
```

```
(1 row)
```

```
select dateadd(day,1,'now');
```

```
date_add
```

```
-----  
2009-11-17 10:45:32.021394
```

```
(1 row)
```

間隔資料類型和常值

您可以使用間隔資料類型，以 `seconds`、`minutes`、`hours`、`days`、`months` 和 `years` 等單位來儲存持續時間。間隔資料類型和常值可用於日期時間計算，例如將間隔新增至日期和時間戳記、加總間隔，以及從日期或時間戳記中減去間隔。間隔常值可以用作資料表中間隔資料類型資料欄的輸入值。

間隔資料類型的語法

若要指定間隔資料類型，以年和月為單位來儲存持續時間：

```
INTERVAL year_to_month_qualifier
```

若要指定間隔資料類型以天數、小時、分鐘和秒為單位來儲存持續時間：

```
INTERVAL day_to_second_qualifier [ (fractional_precision) ]
```

間隔常值的語法

若要指定間隔常值以定義以年和月為單位的持續時間：

```
INTERVAL quoted-string year_to_month_qualifier
```

若要指定間隔常值以定義以天、小時、分鐘和秒為單位的持續時間：

```
INTERVAL quoted-string day_to_second_qualifier [ (fractional_precision) ]
```

引數

引號字串

指定正或負數值，將數量和日期時間單位指定為輸入字串。如果引號字串僅包含數值，則 Amazon Redshift 會從 `year_to_month_qualifier` 或 `day_to_second_qualifier` 確定單位。例如，`'23'` MONTH 代表 1 year 11 months，`'-2'` DAY 代表 -2 days 0 hours 0 minutes 0.0 seconds，`'1-2'` MONTH 代表 1 year 2 months，代表，`'13 day 1 hour 1 minute 1.123 seconds'` SECOND 代表 13 days 1 hour 1 minute 1.123 seconds。如需間隔輸出格式的詳細資訊，請參閱 [間隔樣式](#)。

`year_to_month_qualifier`

指定間隔的範圍。如果您使用限定詞，並建立時間單位小於限定詞的間隔，Amazon Redshift 會截斷並捨棄間隔的較小部分。`year_to_month_qualifier` 的有效值為：

- YEAR
- MONTH
- YEAR TO MONTH

`day_to_second_qualifier`

指定間隔的範圍。如果您使用限定詞，並建立時間單位小於限定詞的間隔，Amazon Redshift 會截斷並捨棄間隔的較小部分。`day_to_second_qualifier` 的有效值為：

- DAY
- HOUR

- MINUTE
- SECOND
- DAY TO HOUR
- DAY TO MINUTE
- DAY TO SECOND
- HOUR TO MINUTE
- HOUR TO SECOND
- MINUTE TO SECOND

INTERVAL 常值的輸出會截斷為指定的最小INTERVAL元件。例如，使用MINUTE限定詞時，Amazon Redshift 會捨棄小於 的時間單位MINUTE。

```
select INTERVAL '1 day 1 hour 1 minute 1.123 seconds' MINUTE
```

產生的值會截斷為 '1 day 01:01:00'。

分數_精確度

選擇性參數，指定間隔中允許的分數位數。只有當您的間隔包含 時，才應指定 fractional_precision 引數SECOND。例如，SECOND(3)會建立僅允許三個小數位的間隔，例如 1.234 秒。分數位數上限為六位數。

工作階段組態會interval_forbid_composite_literals判斷在使用 YEAR TO MONTH和 DAY TO SECOND部分指定間隔時是否傳回錯誤。如需詳細資訊，請參閱[interval_forbid_composite_literals](#)。

間隔算術

您可以使用間隔值搭配其他日期時間值來執行算術操作。下表說明可用的操作，以及每個操作的結果資料類型。

Note

根據方程式中涉及的最小時間單位，可以產生 date和 timestamp 結果的操作會這樣做。例如，當您將 interval新增至結果時date，date如果是 YEAR TO MONTH間隔，則表示，如果是 DAY TO SECOND間隔，則表示時間戳記。

第一個運算元為 `interval` 的操作會針對指定的第二個運算元 `interval` 產生下列結果：

運算子	日期	時間戳記	Interval (間隔)	數值
-	N/A	N/A	Interval (間隔)	N/A
+	日期	日期/時間戳記	Interval (間隔)	N/A
*	N/A	N/A	N/A	Interval (間隔)
/	N/A	N/A	N/A	Interval (間隔)

第一個運算元為 `date` 的操作會針對指定的第二個運算元 `date` 產生下列結果：

運算子	日期	時間戳記	Interval (間隔)	數值
-	數值	Interval (間隔)	日期/時間戳記	日期
+	N/A	N/A	N/A	N/A

第一個運算元為 `timestamp` 的操作會針對指定的第二個運算元 `timestamp` 產生下列結果：

運算子	日期	時間戳記	Interval (間隔)	數值
-	數值	Interval (間隔)	時間戳記	時間戳記
+	N/A	N/A	N/A	N/A

間隔樣式

您可以使用 SQL [the section called “SET”](#) 命令來變更間隔值的輸出顯示格式。當您在 `interval` 資料類型時 SQL，請將它轉換為文字，以查看預期的間隔樣式，例如 `YEAR TO MONTH::text`。此 `IntervalStyle` 值 SET 的可用值為：

- `postgres` – 遵循 PostgreSQL 樣式。此為預設值。
- `postgres_verbose` – 遵循 PostgreSQL 詳細描述樣式。
- `sql_standard` – 遵循 SQL 標準間隔常值樣式。

下列命令會將間隔樣式設定為 `sql_standard`。

```
SET IntervalStyle to 'sql_standard';
```

postgres 輸出格式

以下是postgres間隔樣式的輸出格式。每個數值可以是負數。

```
'<numeric> <unit> [, <numeric> <unit> ...]'
```

```
select INTERVAL '1-2' YEAR TO MONTH::text
```

```
varchar
```

```
-----
```

```
1 year 2 mons
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text
```

```
varchar
```

```
-----
```

```
1 day 02:03:04.5678
```

postgres_verbose 輸出格式

postgres_verbose 語法類似於 postgres，但 postgres_verbose 輸出也包含時間單位。

```
'[@] <numeric> <unit> [, <numeric> <unit> ...] [direction]'
```

```
select INTERVAL '1-2' YEAR TO MONTH::text
```

```
varchar
```

```
-----
```

```
@ 1 year 2 mons
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text
```

```
varchar
```

```
-----
```

```
@ 1 day 2 hours 3 mins 4.56 secs
```

sql_standard 輸出格式

間隔年到月值格式如下。在間隔之前指定負號表示間隔是負值，並套用於整個間隔。

```
'[-]yy-mm'
```

間隔天數到第二個值的格式如下。

```
'[-]dd hh:mm:ss.ffffff'
```

```
SELECT INTERVAL '1-2' YEAR TO MONTH::text
```

```
varchar
-----
1-2
```

```
select INTERVAL '1 2:3:4.5678' DAY TO SECOND::text
```

```
varchar
-----
1 2:03:04.5678
```

間隔資料類型的範例

下列範例示範如何搭配資料表使用INTERVAL資料類型。

```
create table sample_intervals (y2m interval month, h2m interval hour to minute);
insert into sample_intervals values (interval '20' month, interval '2 days
1:1:1.123456' day to second);
select y2m::text, h2m::text from sample_intervals;
```

```
      y2m      |      h2m
-----+-----
1 year 8 mons | 2 days 01:01:00
```

```
update sample_intervals set y2m = interval '2' year where y2m = interval '1-8' year to
month;
select * from sample_intervals;
```



```

y2m | h2m
-----+-----
2 years | 2 days 01:01:00

```

```

delete from sample_intervals where h2m = interval '2 1:1:0' day to second;
select * from sample_intervals;

```

```

y2m | h2m
-----+-----

```

間隔常值的範例

下列範例會在間隔樣式設定為的情況下執行postgres。

下列範例示範如何建立 1 年的INTERVAL常值。

```

select INTERVAL '1' YEAR

```

```

intervaly2m
-----
1 years 0 mons

```

如果您指定超出限定符的引號字串，則剩餘時間單位會從間隔截斷。在下列範例中，13 個月的間隔會變成 1 年又 1 個月，但剩餘的 1 個月會因為YEAR限定符而被淘汰。

```

select INTERVAL '13 months' YEAR

```

```

intervaly2m
-----
1 years 0 mons

```

如果您使用低於間隔字串的限定符，則會包含剩餘單位。

```

select INTERVAL '13 months' MONTH

```

```

intervaly2m
-----
1 years 1 mons

```

在間隔中指定精確度會將分數位數截斷為指定的精確度。

```
select INTERVAL '1.234567' SECOND (3)
```

```
intervald2s
-----
0 days 0 hours 0 mins 1.235 secs
```

如果您未指定精確度，Amazon Redshift 會使用 6 的最高精確度。

```
select INTERVAL '1.23456789' SECOND
```

```
intervald2s
-----
0 days 0 hours 0 mins 1.234567 secs
```

下列範例示範如何建立範圍間隔。

```
select INTERVAL '2:2' MINUTE TO SECOND
```

```
intervald2s
-----
0 days 0 hours 2 mins 2.0 secs
```

限定符會指定您要指定的單位。例如，即使下列範例使用與上一個範例相同的 '2 : 2' 引號字串，Amazon Redshift 仍會因為限定符而識別它使用不同的時間單位。

```
select INTERVAL '2:2' HOUR TO MINUTE
```

```
intervald2s
-----
0 days 2 hours 2 mins 0.0 secs
```

也支援每個單位的縮寫和複數。例如，5s、5 second和5 seconds是相等的間隔。支援的單位為年、月、小時、分鐘和秒。

```
select INTERVAL '5s' SECOND
```

```
intervald2s
-----
```

```
0 days 0 hours 0 mins 5.0 secs
```

```
select INTERVAL '5 HOURS' HOUR
```

```
intervald2s
```

```
-----
0 days 5 hours 0 mins 0.0 secs
```

```
select INTERVAL '5 h' HOUR
```

```
intervald2s
```

```
-----
0 days 5 hours 0 mins 0.0 secs
```

不含限定詞語法的間隔常值範例

Note

下列範例示範使用不含 YEAR TO MONTH或 DAY TO SECOND 限定詞的間隔常值。如需搭配限定詞使用建議間隔常值的資訊，請參閱 [間隔資料類型和常值](#)。

使用間隔常值來表示指定的時間期間，例如 12 hours 或 6 months。您可以在需要表示日期時間的條件和表達式中，使用這些間隔常值。

間隔常值表示為INTERVAL關鍵字與數值數量和支援日期部分的組合，例如 INTERVAL '7 days'或 INTERVAL '59 minutes'。您可以串連幾個數量和單位，來組成更精確的間隔時間，例如：INTERVAL '7 days, 3 hours, 59 minutes'。也支援每種單位的縮寫和複數，例如 5 s、5 second 和 5 seconds 是相同的間隔時間。

如果未指定日期部分，則間隔值代表秒。您可以指定小數格式的數量值 (例如：0.5 days)。

下列範例顯示不同間隔值的一連串計算。

以下內容為指定日期增加 1 秒。

```
select caldate + interval '1 second' as dateplus from date
where caldate='12-31-2008';
dateplus
```

```
-----
```

```
2008-12-31 00:00:01
(1 row)
```

以下內容為指定日期增加 1 分鐘。

```
select caldate + interval '1 minute' as dateplus from date
where caldate='12-31-2008';
dateplus
-----
2008-12-31 00:01:00
(1 row)
```

以下內容為指定日期增加 3 個小時又 35 分鐘。

```
select caldate + interval '3 hours, 35 minutes' as dateplus from date
where caldate='12-31-2008';
dateplus
-----
2008-12-31 03:35:00
(1 row)
```

以下內容為指定日期增加 52 週。

```
select caldate + interval '52 weeks' as dateplus from date
where caldate='12-31-2008';
dateplus
-----
2009-12-30 00:00:00
(1 row)
```

以下內容為指定日期增加 1 週 1 小時 1 分鐘又 1 秒。

```
select caldate + interval '1w, 1h, 1m, 1s' as dateplus from date
where caldate='12-31-2008';
dateplus
-----
2009-01-07 01:01:01
(1 row)
```

以下內容為指定日期增加 12 個小時 (半天)。

```
select caldate + interval '0.5 days' as dateplus from date
where caldate='12-31-2008';
dateplus
-----
2008-12-31 12:00:00
(1 row)
```

以下內容從 2023 年 2 月 15 日減去 4 個月，結果是 2022 年 10 月 15 日。

```
select date '2023-02-15' - interval '4 months';
?column?
-----
2022-10-15 00:00:00
```

以下內容從 2023 年 3 月 31 日減去 4 個月，結果是 2022 年 11 月 30 日。計算有將一個月中的天數納入考量。

```
select date '2023-03-31' - interval '4 months';
?column?
-----
2022-11-30 00:00:00
```

布林值 (Boolean) 類型

使用 BOOLEAN 資料類型將 true 和 false 值儲存在單位元組資料欄中。下表說明 Boolean 值的三種可能狀態，以及導致狀態的字面值。無論輸入的字串為何，Boolean 資料欄都會分別將「t」和「f」儲存和輸出為 true 與 false。

State	有效的常值	儲存
True	TRUE 't' 'true' 'y' 'yes' '1'	1 位元組
False	FALSE 'f' 'false' 'n' 'no' '0'	1 位元組

State	有效的常值	儲存
不明	NULL	1 位元組

您可以使用 IS 比較來檢查布林值，但只能作為 WHERE 子句中的述詞。您無法將 IS 比較與 SELECT 清單中的布林值搭配使用。

範例

您可以使用資料 BOOLEAN 欄來為 CUSTOMER 資料表中的每個客戶儲存「作用中/非作用中」狀態。

```
create table customer(
custid int,
active_flag boolean default true);
```

```
insert into customer values(100, default);
```

```
select * from customer;
custid | active_flag
-----+-----
  100 | t
```

如果 CREATETABLE 未在陳述式中指定預設值（ true 或 false ），插入預設值表示插入 null。

在此範例中，查詢會從 USERS 表中選取喜歡運動但不喜歡戲劇的使用者：

```
select firstname, lastname, likesports, liketheatre
from users
where likesports is true and liketheatre is false
order by userid limit 10;
```

```
firstname | lastname | likesports | liketheatre
-----+-----+-----+-----
Lars      | Ratliff  | t          | f
Mufutau   | Watkins  | t          | f
Scarlett  | Mayer    | t          | f
Shafira   | Glenn    | t          | f
Winifred  | Cherry   | t          | f
Chase     | Lamb     | t          | f
Liberty   | Ellison  | t          | f
```

```
Aladdin | Haney | t | f
Tashya | Michael | t | f
Lucian | Montgomery | t | f
(10 rows)
```

下列範例會從資料表中選取使用者，而該USERS資料表不知道他們是否喜歡搖滾音樂。

```
select firstname, lastname, likerock
from users
where likerock is unknown
order by userid limit 10;
```

```
firstname | lastname | likerock
-----+-----+-----
Rafael    | Taylor   |
Vladimir | Humphrey |
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett  | Mayer    |
(10 rows)
```

下列範例會傳回錯誤，因為它在SELECT清單中使用 IS 比較。

```
select firstname, lastname, likerock is true as "check"
from users
order by userid limit 10;
```

```
[Amazon](500310) Invalid operation: Not implemented
```

下列範例成功，因為它在SELECT清單中使用相等的比較 (=)，而不是 IS 比較。

```
select firstname, lastname, likerock = true as "check"
from users
order by userid limit 10;
```

```
firstname | lastname | check
-----+-----+-----
```

Rafael	Taylor	
Vladimir	Humphrey	
Lars	Ratliff	true
Barry	Roy	
Reagan	Hodge	true
Victor	Hernandez	true
Tamekah	Juarez	
Colton	Roy	false
Mufutau	Watkins	
Naida	Calderon	

HLLSKETCH 類型

使用 HLLSKETCH 資料類型進行 HyperLogLog 草圖。Amazon Redshift 支援稀疏或密集的 HyperLogLog 草圖表示法。草圖一開始是稀疏的，當密集格式更有效地減少使用的記憶體佔用時，會切換到密集。

以下列 JSON 格式匯入、匯出或列印 HyperLogLog 草圖時，Amazon Redshift 會自動轉換稀疏草圖。

```
{"logm":15,"sparse":{"indices":[4878,9559,14523],"values":[1,2,1]}}
```

Amazon Redshift 使用 Base64 格式的字串表示法來表示密集 HyperLogLog 草圖。

Amazon Redshift 會以 Base64 格式使用以下字串表示來表示密集 HyperLogLog 草圖。

```
"ABAABA..."
```

在原始壓縮中使用時，HLLSKETCH 物件的大小上限為 24,580 位元組。

SUPER 類型

使用 SUPER 資料類型將半結構化資料或文件儲存為值。

半結構化資料不符合 SQL 資料庫中使用的關聯式資料模型的剛性和表格式結構。它包含參考資料中不同實體的標籤。它們可以包含複雜的值，例如陣列、巢狀結構和其他與序列化格式相關聯的複雜結構，例如 JSON。SUPER 資料類型是一組無結構描述陣列和結構值，其中包含所有其他純量類型的 Amazon Redshift。

SUPER 資料類型支援個別 SUPER 物件最多 16 MB 的資料。如需 SUPER 資料類型的詳細資訊，包括在資料表中實作資料類型的範例，請參閱 [Amazon Redshift 中的半結構化資料](#)。

SUPER 大於 1MB 的物件只能從下列檔案格式擷取：

- Parquet
- JSON
- TEXT
- CSV

SUPER 資料類型具有下列屬性：

- Amazon Redshift 純量值：
 - Null
 - 布林值
 - 數字，例如 smallint、integer、bigint、decimal 或浮點數 (例如 float4 或 float8)
 - 字串值，如 varchar 或 char
- 複雜值：
 - 值的陣列，包括純量或複雜
 - 一種結構，也稱為元組或物件，它是屬性名稱和值 (純量或複雜) 的映射

這兩種類型的複雜值中的任何一種都包含它們自己的純量或複雜值，而沒有任何正規性限制。

SUPER 資料類型支援半結構化資料以無結構描述的形式持久性。雖然階層式資料模型可能會變更，但舊版本的資料可以共存在同一 SUPER 欄中。

Amazon Redshift 使用 PartiQL 啟用陣列和結構的導覽功能。Amazon Redshift 也會使用 PartiQL 語法在 SUPER 陣列上反覆運算。如需詳細資訊，請參閱 [Navigation \(導覽\)](#) 和 [解除巢狀化查詢](#)。

Amazon Redshift 使用動態輸入來處理無結構描述 SUPER 資料，而無需在查詢中使用資料類型之前宣告資料類型。如需詳細資訊，請參閱 [動態類型](#)。

您可以將動態資料遮罩政策套用至 SUPER 類型資料欄路徑上的 scalar 值。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。如需有關搭配 SUPER 資料類型使用動態資料遮罩的資訊，請參閱 [搭配 SUPER 資料類型路徑使用動態資料遮罩](#)。

VARBYTE 類型

使用 VARBYTE、VARBINARY 或 BINARYVARYING 資料欄來存放具有固定限制的可變長度二進位值。

```
varbyte [ (n) ]
```

位元組數上限 (n) 的範圍為 1 – 16,777,216。預設值為 64,000。

您可能想要使用VARBYTE資料類型的一些範例如下：

- 聯結VARBYTE資料欄上的資料表。
- 建立包含VARBYTE資料欄的具體化檢視。支援逐步重新整理包含VARBYTE資料欄的具體化檢視。不過，VARBYTE除了 COUNT、MINMAX和 GROUP BY 欄上的彙總函數不支援增量重新整理。

為了確保所有位元組都是可列印的字元，Amazon Redshift 會使用十六進位格式來列印VARBYTE值。例如，下列 會將十六進位字串SQL轉換為6162二進位值。即使傳回的值是二進位值，結果仍會列印為十六進位 6162。

```
select from_hex('6162');

 from_hex
-----
 6162
```

Amazon Redshift 支援在 VARBYTE和 之間轉換下列資料類型：

- CHAR
- VARCHAR
- SMALLINT
- INTEGER
- BIGINT

使用 CHAR和 VARCHAR UTF-8 格式投放時。如需 UTF-8 格式的詳細資訊，請參閱 [TO_VARBYTE](#)。從 SMALLINT、轉換時INTEGER，會維護原始資料類型的位元組BIGINT數。也就是 的兩個位元組 SMALLINT、 的四個位元組INTEGER，以及 的八個位元組BIGINT。

下列SQL陳述式會將VARCHAR字串轉換為 VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 616263。

```
select 'abc'::varbyte;

 varbyte
-----
 616263
```

下列SQL陳述式會將欄中CHAR的值轉換為 VARBYTE。此範例會建立具有 CHAR (10) 欄 (c) 的資料表，並插入小於 10 長度的字元值。產生的轉換將結果以空格字元 (hex'20') 填入定義的欄大小。即使傳回的值是二進位值，結果仍會列印為十六進位。

```
create table t (c char(10));
insert into t values ('aa'), ('abc');
select c::varbyte from t;
      c
-----
61612020202020202020
61626320202020202020
```

下列SQL陳述式會將SMALLINT字串轉換為 VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 0005，即兩個位元組或四個十六進位字元。

```
select 5::smallint::varbyte;

varbyte
-----
0005
```

下列SQL陳述式會將 轉換為 INTEGER VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 00000005，即四個位元組或八個十六進位字元。

```
select 5::int::varbyte;

varbyte
-----
00000005
```

下列SQL陳述式會將 轉換為 BIGINT VARBYTE。即使傳回的值是二進位值，結果仍會列印為十六進位 0000000000000005，即八個位元組或 16 個十六進位字元。

```
select 5::bigint::varbyte;

varbyte
-----
0000000000000005
```

支援VARBYTE資料類型的 Amazon Redshift 功能包括：

- [VARBYTE 運算子](#)
- [CONCAT](#)
- [LEN](#)
- [LENGTH 函數](#)
- [OCTET_LENGTH](#)
- [SUBSTRING 函數](#)
- [FROM_HEX](#)
- [TO_HEX](#)
- [FROM_VARBYTE](#)
- [TO_VARBYTE](#)
- [GETBIT](#)
- [載入VARBYTE資料類型的資料欄](#)
- [卸載VARBYTE資料類型的資料欄](#)

搭配 Amazon Redshift 使用VARBYTE資料類型的限制

以下是將VARBYTE資料類型與 Amazon Redshift 搭配使用時的限制：

- Amazon Redshift Spectrum 僅支援 Parquet 和 ORC 檔案的VARBYTE資料類型。
- Amazon Redshift 查詢編輯器和 Amazon Redshift 查詢編輯器 v2 尚未完全支援VARBYTE資料類型。因此，使用VARBYTE運算式時請使用不同的SQL用戶端。

作為使用查詢編輯器的解決方法，如果您的資料長度低於 64 KB 且內容有效 UTF-8，您可以將 VARBYTE值轉換為 VARCHAR，例如：

```
select to_varbyte('6162', 'hex')::varchar;
```

- 您無法搭配 Python 或 Lambda 使用者定義函數 () 使用VARBYTE資料類型UDFs。
- 您無法從HLLSKETCH資料欄建立VARBYTE資料欄或在資料VARBYTE欄 APPROXIMATECOUNTDISTINCT上使用。
- VARBYTE 大於 1 MB 的值只能從下列檔案格式擷取：
 - Parquet
 - 文字

- 逗號分隔值 (CSV)

類型相容性與轉換

接下來的內容將會說明在 Amazon Redshift 中，類型轉換規則與資料類型相容性的運作方式。

相容性

在資料庫各種操作的作業期間，會進行資料類型的比對，以及字面值與常數和資料類型的比對，包括下列的操作：

- 資料表上的資料操作語言 (DML) 操作
- UNION、INTERSECT和 EXCEPT查詢
- CASE 表達式
- 評估述詞，例如 LIKE和 IN
- 評估執行資料比較或擷取的SQL函數
- 數學運算子的比較

這些操作的結果，取決於類型轉換規則和資料類型的相容性。相容性表示不一定需要 one-to-one符合特定值和特定資料類型。由於某些資料類型是相容的，因此可進行隱含轉換或強制轉換 (如需詳細資訊，請參閱 [隱含轉換類型](#))。當資料類型不相容時，有時您可以使用明確的轉換函式，來將值從一種資料類型轉換為另一種。

一般相容性與轉換規則

請注意下列的相容性與轉換規則：

- 一般而言，屬於相同類型類別的資料類型 (例如不同的數值資料類型)，彼此可以相容和隱含轉換。

例如，進行隱含轉換時，您可以將小數值插入整數資料欄。小數會經過四捨五入而變成整數。或者，您可以從日期中擷取 2008 等數值，然後將該數值插入整數資料欄。
- 數值資料類型會強制執行嘗試插入 out-of-range值時發生的溢位條件。例如，精確度為 5 的小數值，不符合精確度定義為 4 的小數資料欄。小數的整數或完整部分一律不會遭到截斷，但是小數的小數部分可以適當地向上或向下四捨五入。不過，從資料表所選取值的明確轉換結果，不會四捨五入。
- 不同類型的字元字串相容；包含單位元組資料和CHAR資料VARCHAR欄字串的字串資料欄字串是可比較且隱含可轉換的。VARCHAR 包含多位元組資料的字串是無法比較的。此外，如果字串是適當的常

值，則可以將字元字串轉換為日期、時間、時間戳記或數值；會忽略開頭或結尾的所有空格。相反地，您也可以將日期、時間、時間戳記或數值，轉換為固定長度或可變長度的字元字串。

Note

您想要轉換為數值類型的字元字串，必須包含表示數字的字元。例如，您可以將字串 '1.0' 或 '5.9' 轉換為小數值，但無法將字串 'ABC' 轉換為任何數值類型。

- 如果您將DECIMAL值與字元字串進行比較，Amazon Redshift 會嘗試將字元字串轉換為DECIMAL值。比較所有其他數值和字元字串時，數值會轉換為字元字串。若要強制執行相反的轉換（例如，將字元字串轉換為整數，或將DECIMAL值轉換為字元字串），請使用明確函數，例如 [CAST](#)。
- 若要將 64 位元DECIMAL或NUMERIC值轉換為更高的精確度，您必須使用明確的轉換函數，例如 CAST或 CONVERT函數。
- 將 DATE或 轉換為 TIMESTAMP TIMESTAMPTZ，或TIME將 轉換為 時TIMETZ，時區會設定為目前的工作階段時區。工作階段時區預設為 UTC。如需設定工作階段時區的相關資訊，請參閱 [timezone](#)。
- 同樣地，TIMESTAMPTZ 會根據TIMESTAMP目前的工作階段時區轉換為 DATETIME、或。工作階段時區預設為 UTC。在轉換之後，會去掉時區的資訊。
- 代表指定時區時間戳記的字元字串會使用TIMESTAMPTZ目前工作階段時區轉換為，這是UTC預設的。同樣地，代表指定時區時間的字元字串會使用TIMETZ目前工作階段時區轉換為，這是UTC預設的。

隱含轉換類型

隱含轉換有兩種：

- 任務中的隱含轉換，例如在 INSERT或 UPDATE命令中設定值。
- 表達式中的隱含轉換，例如在WHERE子句中執行比較。

下表列出了可以在指派敘述或表達式中隱含轉換的資料類型。您也可以使用明確轉換函式來進行這些轉換。

轉換前的類型	轉換後的類型
BIGINT (INT8)	BOOLEAN

轉換前的類型	轉換後的類型
	CHAR
	DECIMAL (NUMERIC)
	DOUBLE PRECISION (FLOAT8)
	INTEGER (INT, INT4)
	REAL (FLOAT4)
	SMALLINT (INT2)
	VARCHAR
CHAR	VARCHAR
DATE	CHAR
	VARCHAR
	TIMESTAMP
	TIMESTAMPTZ
DECIMAL (NUMERIC)	BIGINT (INT8)
	CHAR
	DOUBLE PRECISION (FLOAT8)
	INTEGER (INT, INT4)
	REAL (FLOAT4)
	SMALLINT (INT2)
	VARCHAR
DOUBLE PRECISION (FLOAT8)	BIGINT (INT8)

轉換前的類型	轉換後的類型
	CHAR DECIMAL (NUMERIC) INTEGER (INT, INT4) REAL (FLOAT4) SMALLINT (INT2) VARCHAR
INTEGER (INT, INT4)	BIGINT (INT8) BOOLEAN CHAR DECIMAL (NUMERIC) DOUBLE PRECISION (FLOAT8) REAL (FLOAT4) SMALLINT (INT2) VARCHAR
REAL (FLOAT4)	BIGINT (INT8) CHAR DECIMAL (NUMERIC) INTEGER (INT, INT4) SMALLINT (INT2) VARCHAR

轉換前的類型	轉換後的類型
SMALLINT (INT2)	BIGINT (INT8)
	BOOLEAN
	CHAR
	DECIMAL (NUMERIC)
	DOUBLE PRECISION (FLOAT8)
	INTEGER (INT, INT4)
	REAL (FLOAT4)
	VARCHAR
TIMESTAMP	CHAR
	DATE
	VARCHAR
	TIMESTAMPTZ
	TIME
TIMESTAMPTZ	CHAR
	DATE
	VARCHAR
	TIMESTAMP
	TIMETZ
TIME	VARCHAR
	TIMETZ

轉換前的類型	轉換後的類型
	INTERVAL DAY 至 SECOND
TIMETZ	VARCHAR
	TIME
GEOMETRY	GEOGRAPHY
GEOGRAPHY	GEOMETRY

Note

TIMESTAMPTZ、TIMESTAMP、TIMETZ、DATE TIME 或 字元字串之間的隱含轉換會使用目前的工作階段時區。關於設定目前的時區，詳細資訊請參閱 [timezone](#)。

GEOMETRY 和 GEOGRAPHY 資料類型無法隱含轉換為任何其他資料類型，彼此除外。如需詳細資訊，請參閱 [CAST 函數](#)。

VARBYTE 資料類型無法隱含轉換為任何其他資料類型。如需詳細資訊，請參閱 [CAST 函數](#)。

針對SUPER資料類型使用動態輸入

Amazon Redshift 使用動態輸入來處理無結構描述SUPER資料，而無需在查詢中使用資料類型之前宣告。動態輸入會使用導覽至SUPER資料欄的結果，而不必明確地將其轉換為 Amazon Redshift 類型。如需針對SUPER資料類型使用動態輸入的詳細資訊，請參閱 [動態類型](#)。

您可以將SUPER值轉換為其他資料類型和從其他資料類型轉換，但有一些例外狀況。如需詳細資訊，請參閱 [限制](#)。

定序序列

Amazon Redshift 不支援地區特定或使用者定義的定序句。一般而言，如果缺少對資料值進行排序和比較時的地區特定規則，可能會影響到所有內容中任何述詞的結果。例如，ORDERBY 表達式和函數，例如 MIN、MAX 和 RANK 傳回結果，是根據不考慮地區設定特定字元的資料二進位 UTF8 排序。

表達式

主題

- [簡易表達式](#)
- [複合運算式](#)
- [表達式清單](#)
- [賦值子查詢](#)
- [函數表達式](#)

表達式結合了一個或多個值、運算子或求值的函式。表達式的資料類型通常是其組成項目的資料類型。

簡易表達式

簡易表達式包括：

- 常數或常值
- 資料欄名稱或資料欄參考
- 純量函式
- 彙總 (集) 函式
- 視窗函式
- 賦值子查詢

簡易表達式的範例包括：

```
5+12
dateid
sales.qtysold * 100
sqrt (4)
max (qtysold)
(select max (qtysold) from sales)
```

複合運算式

複合表達式是由算術運算子聯結的一連串的簡單表達式。在複合表達式中使用的簡易表達式，必須傳回數值。

語法

```
expression
```

```
operator  
expression | (compound_expression)
```

引數

運算式

求值的簡易表達式。

operator

複合的算術表達式可使用下列的運算子來建構，依此優先順序使用：

- ()：用來控制求值順序的括號
- +、-：正號和負號/運算子
- ^、|/、||/：乘冪、平方根、立方根
- *、/、%：乘法、除法和模除運算子
- @：絕對值
- +、-：加和減
- 和、|、#、~、<<、>>：AND、OR、NOT、向左移、向右移位元運算子
- ||：串接

(compound_expression)

複合運算式可以使用括號來巢狀化。

範例

複合運算式的範例包括下列。

```
('SMITH' || 'JONES')  
sum(x) / y  
sqrt(256) * avg(column)  
rank() over (order by qtysold) / 100  
(select (pricepaid - commission) from sales where dateid = 1882) * (qtysold)
```

某些函式也可以嵌套於其他函式內。例如，任何純量函式都可以嵌套在另一個純量函式之內。下列的範例會傳回一組數字的絕對值總和：

```
sum(abs(qtysold))
```

視窗函式不能做為彙總函式或其他視窗函式的引數使用。下列的表達式會傳回錯誤：

```
avg(rank() over (order by qtysold))
```

視窗函式可以嵌套彙總函式。下列的表達式會求出值集合的總和，然後加以排序：

```
rank() over (order by sum(qtysold))
```

表達式清單

表達式清單是表達式的組合，並會出現在成員資格和比較條件（WHERE 子句）和 GROUP BY 子句中。

語法

```
expression , expression , ... | (expression , expression , ...)
```

引數

運算式

求值的簡易表達式。表達式清單可以包含用英文逗號分隔的一個或多個表達式，或是用英文逗號分隔的一組或多組表達式。如果包含多組表達式，則每組都必須包含相同數量的表達式，並且用括號分隔。每組中的表達式數量，都必須符合條件中運算子前面的表達式數目。

範例

下列是條件中的表達式清單範例：

```
(1, 5, 10)  
( 'THESE', 'ARE', 'STRINGS' )  
( ('one', 'two', 'three'), ('blue', 'yellow', 'green') )
```

每組中的表達式數量，都必須符合陳述式前半部中的數目：

```
select * from venue
```

```
where (venuecity, venuestate) in (('Miami', 'FL'), ('Tampa', 'FL'))
order by venueid;
```

venueid	venue name	venuecity	venuestate	venueseats
28	American Airlines Arena	Miami	FL	0
54	St. Pete Times Forum	Tampa	FL	0
91	Raymond James Stadium	Tampa	FL	65647

(3 rows)

賦值子查詢

純量子查詢是括號中的一般SELECT查詢，只會傳回一個值：一列和一欄。執行此查詢後，傳回的值會在外層的查詢中使用。如果子查詢傳回 0 列，則子查詢表達式的值為 null。如果子查詢傳回超過一列，Amazon Redshift 會傳回錯誤。子查詢可以參照父查詢傳來的變數，此變數會在子查詢的任何一次叫用期間中，做為常數使用。

您可以在叫用表達式的大多數陳述式中，使用賦值子查詢。賦值子查詢在下列的情況中並非有效的表達式：

- 做為表達式的預設值
- 在 GROUP BY 和 HAVING子句中

範例

下列的子查詢會針對 2008 年一整年，計算出每次銷售平均支付的價格，外層的查詢接著會使用輸出中的此值，來比較每季的銷售平均價格：

```
select qtr, avg(pricepaid) as avg_saleprice_per_qtr,
(select avg(pricepaid)
from sales join date on sales.dateid=date.dateid
where year = 2008) as avg_saleprice_yearly
from sales join date on sales.dateid=date.dateid
where year = 2008
group by qtr
order by qtr;
```

qtr	avg_saleprice_per_qtr	avg_saleprice_yearly
1	647.64	642.28
2	646.86	642.28
3	636.79	642.28

```
4      |          638.26 |          642.28
(4 rows)
```

函數表達式

語法

任何內建的函式皆可做為表達式使用。函式呼叫的語法，是函式的名稱後面接著括號中的引數列表。

```
function ( [expression [, expression...]] )
```

引數

函數

任何內建函式。如需一些範例函數，請參閱[SQL 函數參考](#)。

運算式

符合函式所預期資料類型和參數數目的任何表達式。

範例

```
abs (variable)
select avg (qtysold + 3) from sales;
select dateadd (day,30,caldate) as plus30days from date;
```


條件

主題

- [語法](#)
- [比較條件](#)
- [邏輯條件](#)
- [模式比對條件](#)
- [BETWEEN 範圍條件](#)
- [Null 條件](#)
- [EXISTS 條件](#)

- [IN 條件](#)

條件是一個或多個表達式和邏輯運算子 (判斷值為 true、false 或 unknown) 的陳述式。條件有時也稱為述詞。

 Note

所有字串比較和LIKE模式比對都區分大小寫。例如，「A」和「a」不符。不過，您可以使用 ILIKE 述詞進行不區分大小寫的模式比對。

語法

```
comparison_condition
| logical_condition
| range_condition
| pattern_matching_condition
| null_condition
| EXISTS_condition
| IN_condition
```

比較條件

比較條件表示兩個值之間的邏輯關係。所有比較條件都是二元運算子，具有 Boolean 傳回類型。Amazon Redshift 支援下表中描述的比較運算子：

運算子	語法	描述
<	a < b	a 值小於 b 值。
>	a > b	a 值大於 b 值。
<=	a <= b	a 值小於或等於 b 值。
>=	a >= b	a 值大於或等於 b 值。
=	a = b	a 值等於 b 值。
<> 或 !=	a <> b or a != b	a 值不等於 b 值。

運算子	語法	描述
ANY SOME	a = ANY(subquery)	a 值等於子查詢所傳回的任何值。
ALL	a <> ALL or != ALL (subquery))	a 值不等於子查詢所傳回的任何值。
IS TRUE FALSE UNKNOWN	a IS TRUE	值 a 為布林值 TRUE。

使用須知

= ANY | SOME

ANY 和 SOME 關鍵字是 IN 條件的同義詞，如果對於傳回一或多個值的子查詢傳回的至少一個值比較為 true，則傳回 true。Amazon Redshift 僅支援 ANY 和的 =（等於）條件 SOME。不支援不等式條件。

Note

不支援 ALL 述詞。

<> ALL

ALL 關鍵字與 NOT IN（請參閱 [IN 條件](#) 條件）同義詞，如果表達式未包含在子查詢的結果中，則傳回 true。Amazon Redshift 僅支援的 <> 或 !=（不等於）條件 ALL。不支援其他比較條件。

IS TRUE/FALSE/UNKNOWN

非零值等於 TRUE，0 等於 FALSE，而 null 等於 UNKNOWN。請參閱 [布林值 \(Boolean\) 類型](#) 資料類型。

範例

下列是比較條件的一些簡單範例：

```
a = 5
```

```
a < b
min(x) >= 5
qtysold = any (select qtysold from sales where dateid = 1882
```

下列查詢會從VENUE資料表傳回超過 10000 個座位的場地：

```
select venueid, venuename, venueseats from venue
where venueseats > 10000
order by venueseats desc;
```

venueid	venuename	venueseats
83	FedExField	91704
6	New York Giants Stadium	80242
79	Arrowhead Stadium	79451
78	INVESCO Field	76125
69	Dolphin Stadium	74916
67	Ralph Wilson Stadium	73967
76	Jacksonville Municipal Stadium	73800
89	Bank of America Stadium	73298
72	Cleveland Browns Stadium	73200
86	Lambeau Field	72922
...		

(57 rows)

此範例會從喜歡搖滾音樂的USERS資料表中選取使用者 (USERID)：

```
select userid from users where likerock = 't' order by 1 limit 5;
```

```
userid
-----
3
5
6
13
16
(5 rows)
```

此範例會從USERS資料表中選取使用者 (USERID)，其中不知道他們是否喜歡搖滾音樂：

```
select firstname, lastname, likerock
from users
```

```
where likerock is unknown
order by userid limit 10;
```

```

firstname | lastname | likerock
-----+-----+-----
Rafael    | Taylor   |
Vladimir | Humphrey |
Barry     | Roy      |
Tamekah   | Juarez   |
Mufutau   | Watkins  |
Naida     | Calderon |
Anika     | Huff     |
Bruce     | Beck     |
Mallory   | Farrell  |
Scarlett | Mayer    |
(10 rows)
```

具有TIME資料欄的範例

下列範例資料表 TIME_TEST 具有資料欄 TIME_VAL (類型 TIME) , 並已插入三個值。

```
select time_val from time_test;
```

```

time_val
-----
20:00:00
00:00:00.5550
00:58:00
```

下列範例會擷取每個 timetz_val 中的小時數。

```
select time_val from time_test where time_val < '3:00';
```

```

time_val
-----
00:00:00.5550
00:58:00
```

下列範例會比較兩個時間常值。

```
select time '18:25:33.123456' = time '18:25:33.123456';
```

```
?column?
-----
```

```
t
```

具有TIMETZ資料欄的範例

下列範例資料表 TIMETZ_TEST 具有一欄 TIMETZ_VAL (類型 TIMETZ) ，並插入三個值。

```
select timetz_val from timetz_test;
```

```
timetz_val
-----
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例只會選取小於 TIMETZ 的值 3:00:00 UTC。將值轉換為 後，會進行比較 UTC。

```
select timetz_val from timetz_test where timetz_val < '3:00:00 UTC';
```

```
timetz_val
-----
00:00:00.5550+00
```

下列範例會比較兩個 TIMETZ 常值。比較時會忽略時區。

```
select time '18:25:33.123456 PST' < time '19:25:33.123456 EST';
```

```
?column?
-----
t
```

邏輯條件

邏輯條件會合併兩個條件的結果，來產生單一結果。所有的邏輯條件都是二元運算子，具有 Boolean 傳回類型。

語法

```
expression
{ AND | OR }
expression
```

NOT *expression*

邏輯條件使用三種值的布林邏輯，其中 null 值代表未知的關係。下表說明邏輯條件的結果，其中 E1 和 E2 表示表達式：

E1	E2	E1 AND E2	E1 OR E2	NOT E2
TRUE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	TRUE
TRUE	UNKNOWN	UNKNOWN	TRUE	UNKNOWN
FALSE	TRUE	FALSE	TRUE	
FALSE	FALSE	FALSE	FALSE	
FALSE	UNKNOWN	FALSE	UNKNOWN	
UNKNOWN	TRUE	UNKNOWN	TRUE	
UNKNOWN	FALSE	FALSE	UNKNOWN	
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	

NOT 運算子會在之前進行評估AND，而AND運算子會在 OR 運算子之前進行評估。如果使用任何括號，就可以覆蓋這個預設的評估順序。

範例

下列範例USERNAME會從使用者喜歡拉斯維加斯和運動的USERS資料表傳回 USERID和：

```
select userid, username from users
where likevegas = 1 and likesports = 1
order by userid;
```

```
userid | username
-----+-----
1 | JSG99FHE
67 | TWU10MZT
87 | DUF19VXU
```

```

92 | HYP36WEQ
109 | FPL38HZK
120 | DMJ24GUZ
123 | QZR22XGQ
130 | ZQC82ALK
133 | LBN45WCH
144 | UCX04JKN
165 | TEY680EB
169 | AYQ83HGO
184 | TVX65AZX
...
(2128 rows)

```

下一個範例 USERNAME 會從使用者喜歡拉斯維加斯、運動或兩者的 USERS 資料表傳回 USERID 和。此查詢會傳回前一個範例的所有輸出資料，加上只喜歡拉斯維加斯或運動的使用者。

```

select userid, username from users
where likevegas = 1 or likesports = 1
order by userid;

```

```

userid | username
-----+-----
1 | JSG99FHE
2 | PGL08LJI
3 | IFT66TXU
5 | AEB55QTM
6 | NDQ15VBM
9 | MSD36KVR
10 | WKW41AIW
13 | QTF33MCG
15 | OWU78MTR
16 | ZMG93CDD
22 | RHT62AGI
27 | K0Y02CVE
29 | HUH27PKK
...
(18968 rows)

```

下列的查詢在 OR 條件周圍加上括號，以找出在紐約或加州上演「馬克白」的場地：

```

select distinct venuename, venuecity
from venue join event on venue.venueid=event.venueid
where (venuestate = 'NY' or venuestate = 'CA') and eventname='Macbeth'

```

```
order by 2,1;

venueName          |   venueCity
-----+-----
Geffen Playhouse   | Los Angeles
Greek Theatre      | Los Angeles
Royce Hall         | Los Angeles
American Airlines Theatre | New York City
August Wilson Theatre | New York City
Belasco Theatre   | New York City
Bernard B. Jacobs Theatre | New York City
...
```

如果移除此範例中的括號，將會改變查詢的邏輯和結果。

下列的範例使用 NOT 運算子：

```
select * from category
where not catid=1
order by 1;

catid | catgroup | catname |          catdesc
-----+-----+-----+-----
2 | Sports  | NHL     | National Hockey League
3 | Sports  | NFL     | National Football League
4 | Sports  | NBA     | National Basketball Association
5 | Sports  | MLS     | Major League Soccer
...
```

下列範例使用 NOT 條件，後接 AND 條件：

```
select * from category
where (not catid=1) and catgroup='Sports'
order by catid;

catid | catgroup | catname |          catdesc
-----+-----+-----+-----
2 | Sports  | NHL     | National Hockey League
3 | Sports  | NFL     | National Football League
4 | Sports  | NBA     | National Basketball Association
5 | Sports  | MLS     | Major League Soccer
(4 rows)
```

模式比對條件

主題

- [LIKE](#)
- [SIMILAR 至](#)
- [POSIX 運算子](#)

模式比對運算子會依據條件表達式中所指定的模式，來搜尋字串，並根據是否找到符合的結果，傳回 true 或 false。Amazon Redshift 使用三種方法進行模式比對：

- LIKE 表達式

LIKE 運算子會比較字串表達式，例如資料欄名稱，以及使用萬用字元 % (百分比) 和 _ (底線) 的模式。LIKE 模式比對一律會涵蓋整個字串。LIKE 會執行區分大小寫的比對，並 ILIKE 執行區分大小寫的比對。

- SIMILAR TO 規則運算式

SIMILAR TO 運算子會將字串表達式與 SQL 標準規則表達式模式進行比對，其中可以包含一組模式比對中繼符，其中包含 LIKE 運算子支援的兩個中繼符。SIMILAR TO 會比對整個字串，並執行區分大小寫的比對。

- POSIX 樣式規則運算式

POSIX 規則運算式提供比 LIKE 和 SIMILAR TO 運算子更強大的模式比對方法。POSIX 規則表達式模式可以比對字串的任何部分，並執行區分大小寫的比對。

使用 SIMILAR TO 或 POSIX 運算子進行規則運算式比對，運算成本高昂。我們建議您 LIKE 盡可能使用，特別是在處理大量資料列時。例如，下列查詢的功能相同，但使用的查詢 LIKE 執行速度比使用規則運算式的查詢快上數倍：

```
select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die%';
```

LIKE

LIKE 運算子會比較字串表達式，例如資料欄名稱，以及使用萬用字元 % (百分比) 和 _ (底線) 的模式。LIKE 模式比對一律會涵蓋整個字串。若要比對字串中任意位置的序列，模式必須以百分比符號開頭和結尾。

LIKE 區分大小寫；區分ILIKE大小寫。

語法

```
expression [ NOT ] LIKE | ILIKE pattern [ ESCAPE 'escape_char' ]
```

引數

運算式

有效的 UTF-8 字元表達式，例如資料欄名稱。

LIKE | ILIKE

LIKE 會執行區分大小寫的模式比對。ILIKE 會針對單位元組 UTF-8 (ASCII) 字元執行不區分大小寫的模式比對。若要針對多位元組字元執行不區分大小寫的模式比對，請在 運算式和模式上使用 [LOWER](#) 函數，並具有 LIKE 條件。

與比較述詞相反，例如 = 和 <>，述LIKEILIKE詞不會隱含地忽略尾端空格。若要忽略尾端空格，請使用 RTRIM或明確地將資料CHAR欄轉換為 VARCHAR。

~~ 運算子等於 LIKE，而 ~~* 等於 ILIKE。!~~ 和 !~~* 運算子也相當於 NOTLIKE和 NOT ILIKE。

pattern

具有要比對之模式的有效 UTF-8 字元表達式。

escape_char

字元表達式，將會用來逸出模式中的中繼字元。預設值為兩個反斜線 (「\」)。

如果模式不包含元字元，則模式僅代表字串本身；在這種情況下，其LIKE作用與等於運算子相同。

其中一個字元表達式可以是 CHAR或 VARCHAR 資料類型。如果不同，Amazon Redshift 會將 pattern 轉換為 expression 的資料類型。

LIKE 支援下列模式比對中繼字元：

運算子	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。

範例

下表顯示使用 比對模式的範例LIKE：

表達式	傳回值
'abc' LIKE 'abc'	True
'abc' LIKE 'a%'	True
'abc' LIKE '_B_'	False
'abc' ILIKE '_B_'	True
'abc' LIKE 'c%'	False

下列範例會找出名稱以「E」開頭的所有城市：

```
select distinct city from users
where city like 'E%' order by city;
city
-----
East Hartford
East Lansing
East Rutherford
East St. Louis
Easthampton
Easton
Eatontown
Eau Claire
...
```

下列範例會找出姓氏中包含「ten」的使用者：

```
select distinct lastname from users
where lastname like '%ten%' order by lastname;
lastname
-----
Christensen
Wooten
...
```

以下範例示範如何比對多個模式。

```
select distinct lastname from tickit.users
where lastname like 'Chris%' or lastname like '%Wooten' order by lastname;
lastname
-----
Christensen
Christian
Wooten
...
```

下列範例會找出名稱中第 3 個和第 4 個字元為「ea」的城市。命令使用 ILIKE 來示範不敏感案例：

```
select distinct city from users where city ilike '__EA%' order by city;
city
-----
Brea
Clearwater
Great Falls
Ocean City
Olean
Wheaton
(6 rows)
```

下列的範例使用預設的逸出字串 (\\)，來搜尋包含「_」的字串 (文字 start 後跟底線 _):

```
select tablename, "column" from pg_table_def
where "column" like '%start\\_%'
limit 5;
```

tablename	column
stl_s3client	start_time
stl_tr_conflict	xact_start_ts
stl_undone	undo_start_ts
stl_unload_log	start_time
stl_vacuum_detail	start_row

(5 rows)

下列的範例將「^」指定為逸出字元，然後使用該逸出字元來搜尋包含「_」的字串 (文字 start 後跟底線 _):

```
select tablename, "column" from pg_table_def
where "column" like '%start^_%' escape '^'
limit 5;
```

tablename	column
stl_s3client	start_time
stl_tr_conflict	xact_start_ts
stl_undone	undo_start_ts
stl_unload_log	start_time
stl_vacuum_detail	start_row

(5 rows)

下列範例使用 `~~*` 運算子對以「Ag」開頭的城市進行不區分大小寫（`ILIKE`）搜尋。

```
select distinct city from users where city ~~* 'Ag%' order by city;
```

```
city
-----
Agat
Agawam
Agoura Hills
Aguadilla
```

SIMILAR 至

`SIMILAR TO` 運算子會比對字串表達式，例如資料欄名稱，並具有SQL標準規則表達式模式。SQL 規則表達式模式可以包含一組模式相符的元字元，包括 [LIKE](#) 運算子支援的兩個元。

只有在 `TO SIMILAR` 運算子的模式符合整個字串時，`TO` 運算子才會傳回 `true`，與POSIX規則表達式行為不同，其中模式可以符合字串的任何部分。

`SIMILAR` 執行區分大小寫的比對。

Note

使用 `SIMILAR TO` 進行規則表達式比對的運算成本高昂。我們建議您 `LIKE` 盡可能使用，特別是在處理大量資料列時。例如，下列查詢的功能相同，但使用的查詢 `LIKE` 執行速度比使用規則運算式的查詢快上數倍：

```
select count(*) from event where eventname SIMILAR TO '%(Ring|Die)%';
```

```
select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die%';
```

語法

```
expression [ NOT ] SIMILAR TO pattern [ ESCAPE 'escape_char' ]
```

引數

運算式

有效的 UTF-8 字元表達式，例如資料欄名稱。

SIMILAR 至

SIMILAR 對運算式 中的整個字串執行區分大小寫的模式比對。

pattern

有效的 UTF-8 字元表達式，代表SQL標準規則表達式模式。

escape_char

字元表達式，將會用來逸出模式中的中繼字元。預設值為兩個反斜線 (「\」)。

如果模式未包含任何中繼字元，則模式只代表字串本身。

其中一個字元表達式可以是 CHAR 或 VARCHAR 資料類型。如果不同，Amazon Redshift 會將 pattern 轉換為 expression 的資料類型。

SIMILAR TO 支援下列模式比對中繼字元：

運算子	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。
	表示交替 (兩種替代選項中的任一種)。
*	重複前一個項目 0 次或更多次。

運算子	描述
+	重複前一個項目 1 次或更多次。
?	重複前一個項目 0 次或 1 次。
{m}	重複前一個項目 m 次。
{m,}	重複前一個項目 m 次或更多次。
{m,n}	重複前一個項目至少 m 次，而且不超過 n 次。
()	括號可將多個項目分組為單一邏輯項目。
[...]	括號表達式會指定字元類別，就像在POSIX規則表達式中一樣。

範例

下表顯示使用 SIMILAR TO 比對模式的範例：

表達式	傳回值
'abc' SIMILAR TO 'abc'	True
'abc' SIMILAR TO '_b_'	True
'abc' SIMILAR TO '_A_'	False
'abc' SIMILAR TO '%(b d)%'	True
'abc' SIMILAR TO '(b c)%'	False
'AbcAbcdefgfg12efgfg12' SIMILAR TO '((Ab)?c)+d((efg)+(12))+'	True
'aaaaaab11111xy' SIMILAR TO 'a{6}_ [0-9]{5}(x y){2}'	True
'\$0.87' SIMILAR TO '\$[0-9]+(.[0-9][0-9])?'	True

下列範例會找出名稱中包含「E」或「H」的城市：

```
SELECT DISTINCT city FROM users
WHERE city SIMILAR TO '%E|H%' ORDER BY city LIMIT 5;
```

city
Agoura Hills
Auburn Hills
Benton Harbor
Beverly Hills
Chicago Heights

下列的範例使用預設的逸出字串 (「\\」)，來搜尋包含「_」的字串：

```
SELECT tablename, "column" FROM pg_table_def
WHERE "column" SIMILAR TO '%start\\_%'
ORDER BY tablename, "column" LIMIT 5;
```

tablename	column
stcs_abort_idle	idle_start_time
stcs_abort_idle	txn_start_time
stcs_analyze_compression	start_time
stcs_auto_worker_levels	start_level
stcs_auto_worker_levels	start_wlm_occupancy

下列的範例將「^」指定為逸出字元，然後使用該逸出字元來搜尋包含「_」的字串：

```
SELECT tablename, "column" FROM pg_table_def
WHERE "column" SIMILAR TO '%start^_%' ESCAPE '^'
ORDER BY tablename, "column" LIMIT 5;
```

tablename	column
stcs_abort_idle	idle_start_time
stcs_abort_idle	txn_start_time
stcs_analyze_compression	start_time
stcs_auto_worker_levels	start_level
stcs_auto_worker_levels	start_wlm_occupancy

POSIX 運算子

POSIX 規則表達式是指定相符模式的一系列字元。如果字串是規則運算式所描述之規則集的成員，就會比對規則運算式。

POSIX 規則運算式提供比 [LIKE](#) 和 [SIMILAR TO](#) 運算子更強大的模式比對方法。POSIX 規則表達式模式可以符合字串的任何部分，與 SIMILAR TO 運算子不同，只有在其模式符合整個字串時，才會傳回 true。

Note

使用 POSIX 運算子進行規則運算式比對的成本非常昂貴。我們建議您盡可能使用 [LIKE](#)，特別是在處理大量資料列時。例如，下列查詢的功能相同，但使用的查詢 [LIKE](#) 執行速度比使用規則運算式的查詢快上數倍：

```
select count(*) from event where eventname ~ '.*(Ring|Die).*';
select count(*) from event where eventname LIKE '%Ring%' OR eventname LIKE '%Die%';
```

語法

```
expression [ ! ] ~ pattern
```

引數

運算式

有效的 UTF-8 字元表達式，例如資料欄名稱。

!

否定運算子。不符合規則運算式。

~

針對 expression 的任何子字串，進行區分大小寫的比對。

Note

~~ 是 [LIKE](#) 的同義詞。

pattern

代表規則運算式模式的字串常值。

如果模式未包含萬用字元，則模式只代表字串本身。

若要搜尋包含中繼字元 (例如「. * | ? 」等) 的字串，請使用兩個反斜線 (「 \\」) 來逸出字元。與 SIMILAR TO 和不同 LIKE，POSIX 規則運算式語法不支援使用者定義的逸出字元。

其中一個字元表達式可以是 CHAR 或 VARCHAR 資料類型。如果不同，Amazon Redshift 會將 pattern 轉換為 expression 的資料類型。

所有字元表達式可以是 CHAR 或 VARCHAR 資料類型。如果運算式的資料類型不同，Amazon Redshift 會將其轉換為 expression 的資料類型。

POSIX 模式比對支援下列元字元：

POSIX	描述
.	比對任一個單一字元。
*	比對出現 0 次或更多次。
+	比對出現 1 次或更多次。
?	比對出現 0 次或 1 次。
	指定交替比對；例如，E H 代表 E 或 H。
^	符合 beginning-of-line 字元。
\$	符合 end-of-line 字元。
\$	比對字串結尾。
[]	方括號用來表示符合的清單，應符合清單中的一個表達式。插入符號 (^) 會出現在不符合清單的前面，此清單會比對除了清單中所顯示表達式以外的所有字元。
()	括號可將多個項目分組為單一邏輯項目。

POSIX	描述
{m}	重複前一個項目 m 次。
{m,}	重複前一個項目 m 次或更多次。
{m,n}	重複前一個項目至少 m 次，而且不超過 n 次。
[: :]	比對字元類別中的任何POSIX字元。在下列字元類別中，Amazon Redshift 僅支援ASCII字元：[:alnum:]、[:alpha:]、[:lower:]、[:upper:]

Amazon Redshift 支援下列POSIX字元類別。

字元類別	描述
[[:alnum:]]	所有ASCII英數字元
[[:alpha:]]	所有ASCII字母字元
[[:blank:]]	所有空白字元
[[:cntrl:]]	所有控制字元 (非列印)
[[:digit:]]	所有數字位數
[[:lower:]]	所有小寫ASCII字母字元
[[:punct:]]	所有標點符號字元
[[:space:]]	所有空格字元 (非列印)
[[:upper:]]	所有大寫ASCII字母字元
[[:xdigit:]]	所有的有效十六進位字元

Amazon Redshift 在規則運算式中支援下列受到 Perl 影響的運算子。使用兩個反斜線 (「\\」) 來逸出運算子。

運算子	描述	同等的字元類別表達式
<code>\\d</code>	數字字元	<code>[[:digit:]]</code>
<code>\\D</code>	非數字字元	<code>[^[:digit:]]</code>
<code>\\w</code>	字詞字元	<code>[[:word:]]</code>
<code>\\W</code>	非字詞字元	<code>[^[:word:]]</code>
<code>\\s</code>	空白字元	<code>[[:space:]]</code>
<code>\\S</code>	非空白字元	<code>[^[:space:]]</code>
<code>\\b</code>	界限單字	

範例

下表顯示使用POSIX運算子比對模式的範例：

表達式	傳回值
<code>'abc' ~ 'abc'</code>	True
<code>'abc' ~ 'a'</code>	True
<code>'abc' ~ 'A'</code>	False
<code>'abc' ~ '.*(b d).*'</code>	True
<code>'abc' ~ '(b c).*'</code>	True
<code>'AbcAbcdefgfg12efgfg12' ~ '((Ab)?c)+d((efg)+(12))+'</code>	True
<code>'aaaaaab11111xy' ~ 'a{6}.[1]{5} (x y){2}'</code>	True
<code>'\$0.87' ~ '\\\$[0-9]+(\\. [0-9] [0-9])?'</code>	True

表達式	傳回值
'ab c' ~ '[:,space:]'	True
'ab c' ~ '\\s'	True
' ' ~ '\\S'	False

下列範例會找出名稱中包含 E 或 H 的城市：

```
SELECT DISTINCT city FROM users
WHERE city ~ '.*E.*|.H.*' ORDER BY city LIMIT 5;
```

```
      city
-----
Agoura Hills
Auburn Hills
Benton Harbor
Beverly Hills
Chicago Heights
```

下列範例會找出名稱中不包含 E 或 H 的所有城市：

```
SELECT DISTINCT city FROM users WHERE city !~ '.*E.*|.H.*' ORDER BY city LIMIT 5;
```

```
      city
-----
Aberdeen
Abilene
Ada
Agat
Agawam
```

下列的範例使用逸出字串 (「\\」) 來搜尋包含英文句點的字串。

```
SELECT venueid FROM venue
WHERE venueid ~ '.*\\..*'
ORDER BY venueid;
```

```
venueid
```

```

-----
St. Pete Times Forum
Jobing.com Arena
Hubert H. Humphrey Metrodome
U.S. Cellular Field
Superpages.com Center
E.J. Nutter Center
Bernard B. Jacobs Theatre
St. James Theatre

```

BETWEEN 範圍條件

BETWEEN 條件會使用關鍵字 BETWEEN 和 AND，來檢定表達式是否在一系列值的範圍內。

語法

```
expression [ NOT ] BETWEEN expression AND expression
```

表達式可以是數值、字元或日期時間 (datetime) 資料類型，但這些類型必須相容。範圍包含端點。

範例

第一個範例會計算有多少交易已登錄售出 2、3 或 4 張票券：

```

select count(*) from sales
where qtysold between 2 and 4;

count
-----
104021
(1 row)

```

範圍條件包含開頭值與結尾值。

```

select min(dateid), max(dateid) from sales
where dateid between 1900 and 1910;

min | max
-----+-----
1900 | 1910

```

範圍條件中第一個表達式的值，必須小於第二個表達式的值。由於表達式的值，下列的範例一律會傳回 0 列：

```
select count(*) from sales
where qtysold between 4 and 2;

count
-----
0
(1 row)
```

不過，套用 NOT 修改程式會反轉邏輯，並產生所有資料列的計數：

```
select count(*) from sales
where qtysold not between 4 and 2;

count
-----
172456
(1 row)
```

下列的查詢會傳回擁有 20,000 到 50,000 個座位的場地清單：

```
select venueid, venuename, venueseats from venue
where venueseats between 20000 and 50000
order by venueseats desc;

venueid |          venuename          | venueseats
-----+-----+-----
116 | Busch Stadium                |    49660
106 | Rangers BallPark in Arlington |    49115
96  | Oriole Park at Camden Yards  |    48876
...
(22 rows)
```

下列範例示範使用 BETWEEN 作為日期值：

```
select salesid, qtysold, pricepaid, commission, saletime
from sales
where eventid between 1000 and 2000
      and saletime between '2008-01-01' and '2008-01-03'
order by saletime asc;
```

salesid	qtysold	pricepaid	commission	saletime
65082	4	472	70.8	1/1/2008 06:06
110917	1	337	50.55	1/1/2008 07:05
112103	1	241	36.15	1/2/2008 03:15
137882	3	1473	220.95	1/2/2008 05:18
40331	2	58	8.7	1/2/2008 05:57
110918	3	1011	151.65	1/2/2008 07:17
96274	1	104	15.6	1/2/2008 07:18
150499	3	135	20.25	1/2/2008 07:20
68413	2	158	23.7	1/2/2008 08:12

請注意，雖然 BETWEEN 的範圍包含，但日期預設為時間值 00:00:00。範例查詢的唯一有效 1 月 3 日列是銷售時間為 1/3/2008 00:00:00 的列。

Null 條件

null 條件會檢定 null (缺少值或是值未知)。

語法

```
expression IS [ NOT ] NULL
```

引數

運算式

任何表達式，例如資料欄。

IS NULL

表達式的值如果是 null 則為 true，表達式的值如果包含值，則為 false。

IS NOT NULL

表達式的值如果是 null 則為 false，表達式的值如果包含值，則為 true。

範例

此範例指出 QTYSOLD 欄位中 SALES 資料表包含 null 的次數：

```
select count(*) from sales
where qtysold is null;
```

```
count
-----
0
(1 row)
```

EXISTS 條件

EXISTS 條件測試子查詢中是否存在資料列，如果子查詢至少傳回一個資料列，則傳回 true。如果指定 NOT，則如果子查詢沒有傳回任何資料列，則條件會傳回 true。

語法

```
[ NOT ] EXISTS (table_subquery)
```

引數

EXISTS

當 table_subquery 傳回至少一行時，其值為 true。

NOT EXISTS

當 table_subquery 未傳回任何行時，其值為 true。

table_subquery

子查詢，會評估包含一個或多個欄和一行或多行的資料表。

範例

此範例會針對具有任何類型銷售的日期，傳回所有的日期識別碼，一次一個：

```
select dateid from date
where exists (
select 1 from sales
where date.dateid = sales.dateid
)
order by dateid;

dateid
-----
1827
1828
1829
```


...

IN 條件

IN 條件會檢定值是否隸屬於一組值或子查詢。

語法

```
expression [ NOT ] IN (expr_list | table_subquery)
```

引數

運算式

數值、字元或日期時間 (datetime) 表達式，會根據 `expr_list` 或 `table_subquery` 進行評估，而且必須與該清單或子查詢的資料類型相容。

`expr_list`

用英文逗號分隔的一個或多個表達式，或是用英文逗號分隔的一組或多組表達式 (用括號括住)。

`table_subquery`

子查詢，會評估包含一列或多列的資料表，但是其選擇清單中只限包含一個欄。

IN | NOT IN

如果表達式是表達式清單或查詢的成員，IN 會傳回 true。NOT 如果表達式不是成員，IN 會傳回 true。在下列情況下，IN 和 NOT IN 傳回 NULL，且不會傳回任何資料列：如果表達式產生 null；或者沒有相符的 `expr_list` 或 `table_subquery` 值，且至少其中一個比較資料列產生 null。

範例

只有這些列出的值，才會讓下列條件傳回 true：

```
qtysold in (2, 4, 5)
date.day in ('Mon', 'Tues')
date.month not in ('Oct', 'Nov', 'Dec')
```

大型 IN 清單的最佳化

為了實現最佳化的查詢效能，包含超過 10 個值的 IN 清單，會在內部轉換為純量陣列。包含不到 10 個值的 IN 清單，會轉換為一系列的 OR 述

詞。SMALLINT、INTEGER、BIGINT、REAL、DOUBLEPRECISION、BOOLEAN、CHAR、VARCHAR 和 TIMESTAMPTZ 資料類型支援此最佳化。

查看查詢的EXPLAIN輸出，以查看此最佳化的效果。例如：

```
explain select * from sales
QUERY PLAN
-----
XN Seq Scan on sales (cost=0.00..6035.96 rows=86228 width=53)
Filter: (salesid = ANY ('{1,2,3,4,5,6,7,8,9,10,11}'::integer[]))
(2 rows)
```

SQL 命令

SQL 語言包含您用來建立和操作資料庫物件、執行查詢、載入資料表和修改資料表資料的命令。

Amazon Redshift 是以 Postgre 為基礎SQL。Amazon Redshift 和 PostgreSQL 在設計和開發資料倉儲應用程式時，您必須注意許多重要的差異。如需 Amazon Redshift 與 Postgre 有何SQL不同的詳細資訊SQL，請參閱 [Amazon Redshift 和 PostgreSQL](#)。

Note

單一SQL陳述式的大小上限為 16 MB。

主題

- [ABORT](#)
- [ALTER DATABASE](#)
- [ALTER DATASHARE](#)
- [ALTER DEFAULT PRIVILEGES](#)
- [ALTER EXTERNAL VIEW \(預覽\)](#)
- [ALTER FUNCTION](#)
- [ALTER GROUP](#)
- [ALTER IDENTITY PROVIDER](#)
- [ALTER MASKING POLICY](#)
- [ALTER MATERIALIZED VIEW](#)
- [ALTER RLS POLICY](#)

- [ALTER ROLE](#)
- [ALTER PROCEDURE](#)
- [ALTER SCHEMA](#)
- [ALTER SYSTEM](#)
- [ALTER TABLE](#)
- [ALTER TABLE APPEND](#)
- [ALTER USER](#)
- [ANALYZE](#)
- [ANALYZE COMPRESSION](#)
- [ATTACH MASKING POLICY](#)
- [ATTACH RLS POLICY](#)
- [BEGIN](#)
- [CALL](#)
- [CANCEL](#)
- [CLOSE](#)
- [COMMENT](#)
- [COMMIT](#)
- [COPY](#)
- [CREATE DATABASE](#)
- [CREATE DATASHARE](#)
- [CREATE EXTERNAL FUNCTION](#)
- [CREATE EXTERNAL MODEL](#)
- [CREATE EXTERNAL SCHEMA](#)
- [CREATE EXTERNAL TABLE](#)
- [CREATE EXTERNAL VIEW](#)
- [CREATE FUNCTION](#)
- [CREATE GROUP](#)
- [CREATE IDENTITY PROVIDER](#)
- [CREATE LIBRARY](#)
- [CREATE MASKING POLICY](#)

- [CREATE MATERIALIZED VIEW](#)
- [CREATE MODEL](#)
- [CREATE PROCEDURE](#)
- [CREATE RLS POLICY](#)
- [CREATE ROLE](#)
- [CREATE SCHEMA](#)
- [CREATE TABLE](#)
- [CREATE TABLE AS](#)
- [CREATE USER](#)
- [CREATE VIEW](#)
- [DEALLOCATE](#)
- [DECLARE](#)
- [DELETE](#)
- [DESC DATASHARE](#)
- [DESC IDENTITY PROVIDER](#)
- [DETACH MASKING POLICY](#)
- [DETACH RLS POLICY](#)
- [DROP DATABASE](#)
- [DROP DATASHARE](#)
- [DROP EXTERNAL VIEW \(預覽 \)](#)
- [DROP FUNCTION](#)
- [DROP GROUP](#)
- [DROP IDENTITY PROVIDER](#)
- [DROP LIBRARY](#)
- [DROP MASKING POLICY](#)
- [DROP MODEL](#)
- [DROP MATERIALIZED VIEW](#)
- [DROP PROCEDURE](#)
- [DROP RLS POLICY](#)
- [DROP ROLE](#)

- [DROP SCHEMA](#)
- [DROP TABLE](#)
- [DROP USER](#)
- [DROP VIEW](#)
- [END](#)
- [EXECUTE](#)
- [EXPLAIN](#)
- [FETCH](#)
- [GRANT](#)
- [INSERT](#)
- [INSERT \(外部資料表 \)](#)
- [LOCK](#)
- [MERGE](#)
- [PREPARE](#)
- [REFRESH MATERIALIZED VIEW](#)
- [RESET](#)
- [REVOKE](#)
- [ROLLBACK](#)
- [SELECT](#)
- [SELECT INTO](#)
- [SET](#)
- [SET SESSION AUTHORIZATION](#)
- [SET SESSION CHARACTERISTICS](#)
- [SHOW](#)
- [SHOW COLUMNS](#)
- [SHOW EXTERNAL TABLE](#)
- [SHOW DATABASES](#)
- [SHOW MODEL](#)
- [SHOW DATASHARES](#)
- [SHOW PROCEDURE](#)

- [SHOW SCHEMAS](#)
- [SHOW TABLE](#)
- [SHOW TABLES](#)
- [SHOW VIEW](#)
- [START TRANSACTION](#)
- [TRUNCATE](#)
- [UNLOAD](#)
- [UPDATE](#)
- [VACUUM](#)

ABORT

停止目前執行的交易，並捨棄該交易所做的所有更新。ABORT 對已完成的交易沒有影響。

此命令會執行與ROLLBACK命令相同的函數。如需相關資訊，請參閱 [ROLLBACK](#)。

語法

```
ABORT [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。

TRANSACTION

選用關鍵字；WORK以及 TRANSACTION是同義字。

範例

下列範例會建立資料表，然後開始交易，將資料插入資料表中。然後ABORT，命令會復原資料插入，讓資料表保持空白。

下列命令會建立名為 MOVIE_ 的範例資料表GROSS：

```
create table movie_gross( name varchar(30), gross bigint );
```

下一個命令集會開始交易，將兩個資料列插入資料表中：

```
begin;

insert into movie_gross values ( 'Raiders of the Lost Ark', 23400000);

insert into movie_gross values ( 'Star Wars', 10000000 );
```

接著，以下命令會從資料表中選取資料，表示該資料已成功插入：

```
select * from movie_gross;
```

命令輸出會顯示這兩個資料列都已成功插入：

```
      name          | gross
-----+-----
Raiders of the Lost Ark | 23400000
Star Wars           | 10000000
(2 rows)
```

現在這個命令會將資料變更轉返為交易開始時的狀態：

```
abort;
```

從資料表選取資料現在會顯示空白資料表：

```
select * from movie_gross;

 name | gross
-----+-----
(0 rows)
```

ALTER DATABASE

變更資料庫的屬性。

所需權限

若要使用 ALTER DATABASE，需要下列其中一個權限。

- 超級使用者
- 具有 ALTERDATABASE 權限的使用者
- 資料庫擁有者

語法

```
ALTER DATABASE database_name
{ RENAME TO new_name
| OWNER TO new_owner
| CONNECTION LIMIT { limit | UNLIMITED }
| COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }
| ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT }
| INTEGRATION {{SET REFRESH_INTERVAL <interval>} | REFRESH {{ ALL | INERROR } TABLES
[IN SCHEMA schema [, ...]] | TABLE schema.table [, ...]}}
```

參數

database_name

要修改的資料庫名稱。通常您會修改目前未連接的資料庫；無論何種情況下，變更都只會在後續工作階段中生效。您可以變更目前資料庫的擁有者，但無法將它重新命名：

```
alter database tickit rename to newtickit;
ERROR:  current database may not be renamed
```

RENAME 至

重新命名指定的資料庫。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。您無法重新命名 dev、padb_harvest、template0 或 template1 或 sys:internal 資料庫，也無法重新命名目前的資料庫。只有資料庫擁有者或 [superuser \(p. 791\)](#) 可以重新命名資料庫；非超級使用者擁有者也必須具有 CREATEDB 權限。

new_name

新的資料庫名稱。

OWNER 至

變更所指定資料庫的擁有者。您可以變更目前資料庫或其他資料庫的擁有者。但只有超級使用者可以變更擁有者。

new_owner

新的資料庫擁有者。新的擁有者必須是具備寫入權限的現有資料庫使用者。如需使用者權限的相關資訊，請參閱 [GRANT](#)。

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字來允許並行連線的數量上限。另外也可能限制每位使用者的連線數目。如需詳細資訊，請參閱 [CREATE USER](#)。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 [STV_SESSIONS](#) 系統畫面。

Note

如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }

指定字串搜尋或比較是否區分大小寫的子句。

您可以變更目前空資料庫的區分大小寫設定。

您必須擁有目前資料庫的權限，才能變更區分大小寫の設定。具有 CREATEDATABASE 權限的超級使用者或資料庫擁有者也可以變更資料庫大小寫敏感度。

ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT }

指定對資料庫執行查詢時所用隔離層級的子句。

- SERIALIZABLE 隔離 – 為並行交易提供完整的序列化功能。如需詳細資訊，請參閱 [可序列化隔離](#)。
- SNAPSHOT 隔離 – 提供隔離層級，防止更新和刪除衝突。

如需隔離層級的相關資訊，請參閱 [CREATE DATABASE](#)。

修改資料庫的隔離層級時，請考慮下列項目：

- 您必須擁有目前資料庫的超級使用者或 CREATEDATABASE 權限，才能變更資料庫隔離層級。
- 您無法變更 dev 資料庫的隔離層級。
- 您無法變更交易區塊中的隔離等級。

- 如果其他使用者連線到資料庫，修改隔離層級命令就會失敗。
- 修改隔離層級命令可以修改目前工作階段的隔離層級設定。

```
INTEGRATION REFRESH { ALL | INERROR } TABLES [ IN SCHEMA 結構描述 [ , ... ] ] |  
TABLE 結構描述.table [ , ... ] }
```

指定 Amazon Redshift 是否會重新整理指定結構描述或資料表中的所有資料表或發生錯誤之資料表的子句。重新整理會觸發指定結構描述或資料表中的資料表，從來源資料庫完全複製。

如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的[使用零ETL整合](#)。如需整合狀態的詳細資訊，請參閱 [SVV_INTEGRATION_TABLE_STATE](#) 和 [SVV_INTEGRATION](#)。

```
INTEGRATION { SET REFRESH_INTERVAL <interval> }
```

REFRESH_SET INTERVAL 子句會設定大約的時間間隔，以秒為單位，將資料從零ETL來源重新整理至目標資料庫。對於來源類型為 Aurora My、Aurora Postgre 或 RDS My 的零ETL整合SQL，此值可以設定為 0–432SQL, 000 秒 (5 天) SQL。

如需使用零ETL整合建立資料庫的詳細資訊，請參閱 [Amazon Redshift 管理指南 中的在 Amazon Redshift 中建立目的地資料庫](#)。

使用須知

ALTER DATABASE 命令會套用至後續工作階段，而非目前的工作階段。您必須重新連線到修改過的資料庫，才會看見變更的效果。

範例

下列範例將名為 TICKIT_SANDBOX 的資料庫重新命名為 TICKIT_TEST：

```
alter database tickit_sandbox rename to tickit_test;
```

下列範例會將TICKIT資料庫（目前資料庫）的擁有者變更為 DWUSER：

```
alter database tickit owner to dwuser;
```

下列範例會變更 sampledb 資料庫的資料庫大小寫設定：

```
ALTER DATABASE sampledb COLLATE CASE_INSENSITIVE;
```

下列範例會變更為 **sampledb** 的資料庫與SNAPSHOT隔離層級。

```
ALTER DATABASE sampledb ISOLATION LEVEL SNAPSHOT;
```

下列範例會重新整理 **sample_integration_db** 零ETL整合 **schema2.sample_table2** 中資料庫中的資料表 **schema1.sample_table1** 和 。

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH TABLES schema1.sample_table1,
schema2.sample_table2;
```

下列範例會重新整理零ETL整合中的所有同步和失敗資料表。

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH ALL tables;
```

下列範例會將零ETL整合的重新整理間隔設定為 600 秒。

```
ALTER DATABASE sample_integration_db INTEGRATION SET REFRESH_INTERVAL 600;
```

下列範例會重新整理結構描述 **ErrorState** 中 中的所有資料表 **sample_schema**。

```
ALTER DATABASE sample_integration_db INTEGRATION REFRESH INERROR TABLES in SCHEMA
sample_schema;
```

ALTER DATASHARE

變更資料共用的定義。您可以使用 新增或移除物件ALTERDATASHARE。您只能變更目前資料庫中的資料共用。在資料共用中新增或移除相關聯資料庫中的物件。在要新增或移除的資料共用物件上有所需許可的資料共用擁有者可以修改資料共用。

所需權限

以下是 ALTER 的必要權限DATASHARE：

- 超級使用者。
- 具有 ALTERDATASHARE權限的使用者。
- 在資料共用上具有 ALTER或 ALL權限的使用者。
- 若要將特定物件新增至資料共用，使用者必須要有物件的權限。在此情況下，使用者應該是物件的擁有者SELECT，或在物件上具有 USAGE、或 ALL權限。

語法

下列語法說明如何在資料共用中新增或移除物件。

```
ALTER DATASHARE datashare_name { ADD | REMOVE } {
TABLE schema.table [, ...]
| SCHEMA schema [, ...]
| FUNCTION schema.sql_udf (argtype,...) [, ...]
| ALL TABLES IN SCHEMA schema [, ...]
| ALL FUNCTIONS IN SCHEMA schema [, ...] }
```

下列語法說明如何設定資料共用的屬性。

```
ALTER DATASHARE datashare_name {
[ SET PUBLICACCESSIBLE [=] TRUE | FALSE ]
[ SET INCLUDENEW [=] TRUE | FALSE FOR SCHEMA schema ] }
```

參數

datashare_name

要修改的資料共用名稱。

ADD | REMOVE

指定是否要在資料共用中新增或移除物件的子句。

TABLE 結構描述 .table 【、...】

指定結構描述中要新增至資料共用的資料表或檢視名稱。

SCHEMA 結構描述 【、...】

要新增至資料共用的結構描述名稱。

FUNCTION 結構描述 .sql_udf (argtype , ...) 【、...】

使用者定義SQL函數的名稱，其中包含要新增至資料共用的引數類型。

ALL TABLES IN SCHEMA 結構描述 【、...】

指定是否將指定結構描述中的所有資料表和檢視新增至資料共用的子句。

ALL FUNCTIONS IN SCHEMA 結構描述 【、...】 }

指定將指定結構描述中所有函數新增至資料共用的子句。

[SET PUBLICACCESSIBLE [=] TRUE | FALSE]

該子句會指定資料共用是否可以共用至可公開存取的叢集。

SET INCLUDENEW 【 [=] TRUE | FALSE FOR SCHEMA 結構描述 】

指定是否將指定結構描述中建立的任何未來資料表、檢視或SQL使用者定義函數（UDFs）新增至資料共用的子句。指定的結構描述SQLUDFs中的目前資料表、檢視或不會新增至資料共用。只有超級使用者可以為每個「資料共用-結構描述」配對修改此屬性。根據預設，子INCLUDENEW句為false。

ALTER DATASHARE 用量備註

- 下列使用者可以修改資料共用：
 - 超級使用者
 - 資料共用的擁有者
 - 在資料共用上具有 ALTER或 ALL權限的使用者
- 若要將特定物件新增至資料共用，使用者必須要有物件的正確權限。使用者應該是物件的擁有者 SELECT，或在物件上具有 USAGE、或 ALL權限。
- 您可以共用結構描述、資料表、常規檢視、繫結後期檢視、具體化檢視和SQL使用者定義函數（UDFs）。在結構描述中新增物件之前，請先將結構描述新增至資料共用。

當您新增結構描述時，Amazon Redshift 不會在其下方新增所有物件。您必須明確地新增這些物件。

- 建議您在開啟公開存取設定的情況下建立 AWS Data Exchange 資料共用。
- 一般而言，建議您不要變更 AWS Data Exchange 資料共用，以使用 ALTERDATASHARE陳述式關閉公有可存取性。如果您這麼做，AWS 帳戶 則具有資料共用存取權的 會在叢集可公開存取時失去存取權。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。有關此建議的例外情況，請參閱以下內容。

下列範例顯示資料 AWS Data Exchange 共用建立時關閉設定時的錯誤。

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
ERROR: Alter of ADX-managed datashare salesshare requires session variable
datashare_break_glass_session_var to be set to value 'c670ba4db22f4b'
```

若要允許變更 AWS Data Exchange 資料共用以關閉可公開存取的設定，請設定下列變數，然後再次執行ALTERDATASHARE陳述式。

```
SET datashare_break_glass_session_var to 'c670ba4db22f4b';
```

```
ALTER DATASHARE salesshare SET PUBLICACCESSIBLE FALSE;
```

在此情況下，Amazon Redshift 會產生隨機的一次性值，以設定工作階段變數以允許 ALTERDATASHARESETPUBLICACCESSIBLEFALSE AWS Data Exchange 資料共用。

範例

下列範例會將 public 結構描述新增至資料共用 salesshare。

```
ALTER DATASHARE salesshare ADD SCHEMA public;
```

下列範例會將 public.tickit_sales_redshift 資料表新增至 salesshare 資料共用。

```
ALTER DATASHARE salesshare ADD TABLE public.tickit_sales_redshift;
```

下列範例會將所有資料表新增至 salesshare 資料共用。

```
ALTER DATASHARE salesshare ADD ALL TABLES IN SCHEMA PUBLIC;
```

下列範例會將 public.tickit_sales_redshift 資料表從 salesshare 資料共用中移除。

```
ALTER DATASHARE salesshare REMOVE TABLE public.tickit_sales_redshift;
```

ALTER DEFAULT PRIVILEGES

定義要套用至指定使用者未來建立之物件的預設存取許可集。根據預設，使用者只能變更自己擁有的預設存取許可。只有超級使用者才能為其他使用者指定預設許可。

您可將預設權限套用至角色、使用者或使用者群組。您可以針對在目前資料庫中建立的所有物件，或在指定的結構描述中建立的物件，全域設定預設許可。

預設許可僅適用於新的物件。執行 ALTERDEFAULTPRIVILEGES 不會變更現有物件的許可。若要授予資料庫或結構描述中任何使用者建立的所有目前和未來物件的許可，請參閱 [範圍許可](#)。

若要檢視有關資料庫使用者預設權限的資訊，請查詢 [PG_DEFAULT_ACL](#) 系統目錄資料表。

如需權限的相關資訊，請參閱 [GRANT](#)。

所需權限

以下是 ALTER DEFAULT 的必要權限PRIVILEGES：

- 超級使用者
- 具有 ALTERDEFAULTPRIVILEGES權限的使用者
- 變更自身預設存取權限的使用者
- 為其有權存取的結構描述設定權限的使用者

語法

```
ALTER DEFAULT PRIVILEGES
  [ FOR USER target_user [, ...] ]
  [ IN SCHEMA schema_name [, ...] ]
  grant_or_revoke_clause
```

where *grant_or_revoke_clause* is one of:

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | TRUNCATE } [,...] |
  ALL [ PRIVILEGES ] }
  ON TABLES
  TO { user_name [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]
```

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
  ON FUNCTIONS
  TO { user_name [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]
```

```
GRANT { EXECUTE | ALL [ PRIVILEGES ] }
  ON PROCEDURES
  TO { user_name [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]
```

```
REVOKE [ GRANT OPTION FOR ] { { SELECT | INSERT | UPDATE | DELETE | REFERENCES |
  TRUNCATE } [,...] | ALL [ PRIVILEGES ] }
  ON TABLES
  FROM user_name [, ...] [ RESTRICT ]
```

```
REVOKE { { SELECT | INSERT | UPDATE | DELETE | REFERENCES | TRUNCATE } [,...] | ALL
[ PRIVILEGES ] }
ON TABLES
FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ] { EXECUTE | ALL [ PRIVILEGES ] }
ON FUNCTIONS
FROM user_name [, ...] [ RESTRICT ]

REVOKE { EXECUTE | ALL [ PRIVILEGES ] }
ON FUNCTIONS
FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ] { EXECUTE | ALL [ PRIVILEGES ] }
ON PROCEDURES
FROM user_name [, ...] [ RESTRICT ]

REVOKE { EXECUTE | ALL [ PRIVILEGES ] }
ON PROCEDURES
FROM { ROLE role_name | GROUP group_name | PUBLIC } [, ...] [ RESTRICT ]
```

參數

FOR USER *target_user*

選用。為其定義預設權限的使用者名稱。只有超級使用者能夠指定其他使用者的預設權限。預設值是目前使用者。

IN SCHEMA *schema_name*

選用。如果出現 IN SCHEMA 子句，指定的預設權限會套用至在指定的 *schema_name* 中建立的新物件。在此情況下，目標的使用者或使用者群組 ALTERDEFAULTPRIVILEGES 必須具有指定結構描述 CREATE 的權限。結構描述專屬的預設權限會新增至現有的全域預設權限。根據預設，預設權限會全域套用至整個資料庫。

GRANT

授予指定使用者或群組由指定使用者建立之所有新資料表和檢視、函數或預存程序的一組權限。您可以使用 [GRANT](#) 命令的 GRANT 子句來設定相同的權限和選項。

WITH GRANT OPTION

指出接收權限的使用者可以接著將相同的權限授予他人的子句。您無法 WITHGRANTOPTION 將授予群組或 PUBLIC。

TO user_name | ROLE role_name | GROUP group_name

使用者、角色或使用者群組的名稱，將對其套用指定的預設權限。

REVOKE

對於指定使用者所建立的所有新資料表、函數或預存程序，要撤銷指定使用者或群組的權限集。您可以使用 [REVOKE](#) 命令的 REVOKE 子句來設定相同的權限和選項。

GRANT OPTION FOR

這個子句只會撤銷對其他使用者授予指定權限的選項，而不會撤銷權限本身。您無法 GRANT OPTION 從 群組或 撤銷 PUBLIC。

FROM user_name | ROLE role_name | GROUP group_name

使用者、角色或使用者群組的名稱，預設將撤銷其指定權限。

RESTRICT

RESTRICT 選項只會撤銷使用者直接授予的那些權限。此為預設值。

範例

假設您想要允許使用者群組中的任何使用者 report_readers 檢視使用者 建立的所有資料表和檢視 report_admin。在此情況下，請以超級使用者身分執行下列命令。

```
alter default privileges for user report_admin grant select on tables to group
report_readers;
```

在下列範例中，第一個命令會針對您建立的所有新資料表和檢視授予 SELECT 權限。

```
alter default privileges grant select on tables to public;
```

下列範例會將您在 sales 結構描述中建立的所有新資料表和檢視 INSERT 的權限授予 sales_admin 使用者群組。

```
alter default privileges in schema sales grant insert on tables to group sales_admin;
```

下列範例會反轉上述範例中的 ALTER DEFAULT PRIVILEGES 命令。

```
alter default privileges in schema sales revoke insert on tables from group
sales_admin;
```

根據預設，PUBLIC使用者群組具有所有新使用者定義函數的執行許可。若要撤銷新功能的 public 執行許可，然後僅將執行許可授予 dev_test 使用者群組，請執行下列命令。

```
alter default privileges revoke execute on functions from public;
alter default privileges grant execute on functions to group dev_test;
```

ALTER EXTERNAL VIEW (預覽)

這是適用於 Amazon Redshift 的資料目錄中的發行前版本文件檢視，屬於預覽版本。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。
2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。AWS 區域 會列出您帳戶中目前的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的 [建立叢集](#)。
5. 選擇建立叢集按鈕以建立預覽叢集。

Note

preview_2023 軌跡是最近可用的預覽軌跡。此追蹤僅支援使用RA3節點類型建立叢集。不支援節點類型DC2和任何較舊的節點類型。

6. 預覽叢集可用時，請使用SQL用戶端載入和查詢資料。

Data Catalog 視觀表預覽功能僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

您也可以建立預覽工作群組來測試 Data Catalog 視觀表。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱 [建立預覽工作群組](#)。

使用 ALTEREXTERNALVIEW 命令更新您的外部檢視。視您使用的參數而定，其他引擎，SQL 例如 Amazon Athena 和 Amazon EMR Spark，也可以參考此檢視可能會受到影響。如需有關 Data Catalog 視觀表的詳細資訊，請參閱 [建立 Data Catalog 視觀表 \(預覽\)](#)。

語法

```
ALTER EXTERNAL VIEW schema_name.view_name
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
  external_schema_name.view_name}
[FORCE] { AS (query_definition) | REMOVE DEFINITION }
```

參數

schema_name.view_name

連接至 AWS Glue 資料庫的結構描述，後面接著檢視的名稱。

*catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |
external_schema_name.view_name*

變更視觀表時要使用的結構描述標記法。您可以指定使用您建立的 AWS Glue Data Catalog Glue 資料庫，或您建立的外部結構描述。如需詳細資訊 [CREATEEXTERNALSCHEMA](#)，請參閱 [CREATEDATABASE](#) 和。

FORCE

即使資料表中參考的物件與其他 SQL 引擎不一致，是否仍 AWS Lake Formation 應更新檢視的定義。如果 Lake Formation 更新檢視，則視觀表會被視為其他 SQL 引擎過時，直到這些引擎也更新為止。

AS query_definition

Amazon Redshift 執行以變更檢視的SQL查詢定義。

REMOVE DEFINITION

是否捨棄並重新建立檢視。必須捨棄檢視並重新建立，才能將其標記為 PROTECTED。

範例

下列範例會變更為 sample_schema.glue_data_catalog_view 的 Data Catalog 視觀表。

```
ALTER EXTERNAL VIEW sample_schema.glue_data_catalog_view
FORCE
REMOVE DEFINITION
```

ALTER FUNCTION

重新命名函數或變更擁有者。函數名稱和資料類型都是必要的。只有擁有者或超級使用者可以重新命名函數。只有超級使用者可以變更函數的擁有者。

語法

```
ALTER FUNCTION function_name ( { [ py_arg_name py_arg_data_type | sql_arg_data_type ]
[ , ... ] } )
    RENAME TO new_name
```

```
ALTER FUNCTION function_name ( { [ py_arg_name py_arg_data_type | sql_arg_data_type ]
[ , ... ] } )
    OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
```

參數

function_name

要變更的函數名稱。在目前的搜尋路徑中指定函數的名稱，或使用格式 schema_name.function_name 來使用特定的結構描述。

py_arg_name py_arg_data_type | sql_arg_data_type

選用。Python 使用者定義函數的輸入引數名稱和資料類型清單，或SQL使用者定義函數的輸入引數資料類型清單。

new_name

使用者定義函數的新名稱。

new_owner | CURRENT_USER | SESSION_USER

使用者定義函數的新擁有者。

範例

下列範例會將函數的名稱從 `quarterly_revenue` 變更為 `first_quarter_revenue`。

```
ALTER FUNCTION first_quarter_revenue(bigint, numeric, int)
    RENAME TO quarterly_revenue;
```

下列範例會將 `quarterly_revenue` 函數的擁有者變更為 `etl_user`。

```
ALTER FUNCTION quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

ALTER GROUP

變更使用者群組。使用此命令可將使用者新增至群組、從群組中刪除使用者，或重新命名群組。

語法

```
ALTER GROUP group_name
{
  ADD USER username [, ... ] |
  DROP USER username [, ... ] |
  RENAME TO new_name
}
```

參數

group_name

要修改的使用者群組名稱。

ADD

將使用者新增至使用者群組。

DROP

從使用者群組中移除使用者。

username

要新增至群組或從群組中刪除的使用者名稱。

RENAME 至

重新命名使用者群組。開頭為兩個底線的群組名稱保留供 Amazon Redshift 內部使用。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

new_name

使用者群組的新名稱。

範例

下列範例DWUSER會將名為 的使用者新增至 ADMIN_GROUP 群組。

```
ALTER GROUP admin_group
ADD USER dwuser;
```

下列範例會將群組 ADMIN_GROUP 重新命名為 ADMINISTRATORS。

```
ALTER GROUP admin_group
RENAME TO administrators;
```

下列範例會將兩個使用者新增至群組 ADMIN_GROUP。

```
ALTER GROUP admin_group
ADD USER u1, u2;
```

下列範例會從群組 ADMIN_ 捨棄兩個使用者GROUP。

```
ALTER GROUP admin_group
DROP USER u1, u2;
```

ALTER IDENTITY PROVIDER

修改身分提供者以指派新參數和值。當您執行此命令時，所有先前設定的參數值都會先刪除，然後再指派新值。只有超級使用者可以修改身分提供者。

語法

```
ALTER IDENTITY PROVIDER identity_provider_name
[PARAMETERS parameter_string]
[NAMESPACE namespace]
[IAM_ROLE iam_role]
[DISABLE | ENABLE]
```

參數

identity_provider_name

新身分提供者的名稱。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

parameter_string

包含正確格式化JSON物件的字串，其中包含特定身分提供者所需的參數和值。

命名空間

組織命名空間。

iam_role

提供與 IAM Identity Center 連線許可IAM的角色。此參數僅適用於 *identity-provider* 類型為 時 AWSIDC。

DISABLE 或 ENABLE

開啟或關閉身分提供者。預設值為 ENABLE

範例

下列範例會修改名為 *oauth_standard* 的身分提供者。當 Microsoft Azure AD 是身分提供者時，它特別適用於。

```
ALTER IDENTITY PROVIDER oauth_standard
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
"client_id":"87f4aa26-78b7-410e-bf29-57b39929ef9a",
"client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]}'
```

下列範例示範如何設定 identity-provider 命名空間。如果 Microsoft Azure AD 遵循如先前範例的陳述式，或另一個身分提供者，則這可以適用於 Microsoft Azure AD。如果您透過受管應用程式設定連線，也可以套用到將現有 Amazon Redshift 佈建叢集或 Amazon Redshift Serverless 工作群組連線至 IAM Identity Center 的情況。

```
ALTER IDENTITY PROVIDER "my-redshift-idc-application"  
NAMESPACE 'MYCO';
```

下列範例會設定 IAM 角色，並在使用案例中運作，以設定與 IAM Identity Center 的 Redshift 整合。

```
ALTER IDENTITY PROVIDER "my-redshift-idc-application"  
IAM_ROLE 'arn:aws:iam::123456789012:role/myadministratorrole';
```

如需從 Redshift 設定與 IAM Identity Center 連線的詳細資訊，請參閱[使用 IAM Identity Center 連線 Redshift，為使用者提供單一登入體驗。](#)

停用身分提供者

下列範例陳述式說明如何停用身分提供者。停用時，身分提供者的聯合使用者必須先再次啟用，才能登入叢集。

```
ALTER IDENTITY PROVIDER "redshift-idc-app" DISABLE;
```

ALTER MASKING POLICY

修改現有的動態資料遮罩政策。如需動態資料遮罩的相關資訊，請參閱[動態資料遮罩](#)。

超級使用者和具有 sys.secadmin 角色的使用者或角色可以修改遮罩政策。

語法

```
ALTER MASKING POLICY policy_name  
USING (masking_expression);
```

參數

policy_name

遮罩政策的名稱。這必須是資料庫中已存在的遮罩政策名稱。

masking_expression

用於轉換目標資料欄的SQL表達式。它可以使用資料操作函數編寫，例如字串操作函數，或結合以SQL、Python 或 編寫的使用者定義函數 AWS Lambda。

運算式必須與原始運算式的輸入資料欄和資料類型相符。例如，如果原始遮罩政策的輸入資料欄是 `sample_1 FLOAT`和 `sample_2 VARCHAR(10)`，則您無法將遮罩政策變更為使用第三個資料欄，或讓政策使用 `FLOAT`和 `BOOLEAN`。如果您使用常數做為遮罩運算式，則必須將其明確轉換為符合輸入類型的類型。

您必須擁有遮罩表達式中使用的任何使用者定義函數的USAGE許可。

ALTER MATERIALIZED VIEW

變更具體化視觀表的屬性。

語法

```
ALTER MATERIALIZED VIEW mv_name
[ AUTO REFRESH { YES | NO } ]
[ ROW LEVEL SECURITY { ON | OFF } [ CONJUNCTION TYPE { AND | OR } ] [FOR DATASHARES] ];
```

參數

mv_name

具體化視觀表的名稱。

AUTO REFRESH { YES | NO }

開啟或關閉具體化視觀表自動重新整理的子句。如需具體化視觀表自動重新整理的相關資訊，請參閱 [重新整理具體化視觀表](#)。

ROW LEVEL SECURITY { ON | OFF } 【 CONJUNCTION TYPE { AND | 或 } 】 【 FOR DATASHARES 】

開啟或關閉關係的資料列層級安全性的子句。

開啟關係的資料列層級安全性時，您只能讀取資料列層級安全政策允許您存取的資料列。如果沒有任何政策授予您關係存取權，您就無法看到關聯中的任何資料列。只有具有 `sys:secadmin` 角色

的超級使用者和使用者或角色才能設定ROWLEVELSECURITY子句。如需詳細資訊，請參閱[資料列層級安全性](#)。

- **【 CONJUNCTION TYPE { AND | OR } 】**

一個子句，讓您為關係選擇資料列層級安全政策的結合類型。當多個資料列層級安全政策連接至關係時，您可以將政策與 AND 或 OR 子句結合。根據預設，Amazon Redshift 會將RLS政策與 AND 子句結合。具有 sys:secadmin 角色的超級使用者、使用者或角色可以使用此子句來定義關係之資料列層級安全政策的組合類型。如需詳細資訊，請參閱[每個使用者結合多個政策](#)。

- FOR DATASHARES

子句，用於判斷 RLS 受保護的關係是否可以透過資料共用存取。根據預設，無法透過資料共用存取 RLS 保護的關係。使用此子句執行的 ALTER MATERIALIZED VIEW ROW LEVEL SECURITY 命令只會影響關係的資料共用可存取性屬性。ROW LEVEL SECURITY 屬性不會變更。

如果您透過資料共用存取 RLS 保護關係，則該關係在取用者端資料共用資料庫中不會具有資料列層級安全性。關係會保留其 RLS 屬性在生產者端。

範例

下列範例可讓 tickets_mv 具體化視觀表自動重新整理。

```
ALTER MATERIALIZED VIEW tickets_mv AUTO REFRESH YES
```

ALTER RLS POLICY

修改資料表上現有的資料列層級安全性政策。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以修改政策。

語法

```
ALTER RLS POLICY policy_name  
USING ( using_predicate_exp );
```

參數

policy_name

政策的名稱。

USING (using_predicate_exp)

指定套用至查詢子WHERE句的篩選條件。Amazon Redshift 會在查詢層級使用者述詞之前套用政策述詞。例如，**current_user = 'joe' and price > 10** 會限制 Joe 只能查看價格大於 \$10 的記錄。

表達式可存取CREATERLSPOLICY陳述式子WITH句中宣告的變數，該變數用於建立具有 name policy_name 的政策。

範例

下列範例會變更RLS政策。

```
-- First create an RLS policy that limits access to rows where catgroup is 'concerts'.
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'concerts');

-- Then, alter the RLS policy to only show rows where catgroup is 'piano concerts'.
ALTER RLS POLICY policy_concerts
USING (catgroup = 'piano concerts');
```

ALTER ROLE

重新命名角色或變更擁有者。如需 Amazon Redshift 系統定義角色的清單，請參閱 [the section called “Amazon Redshift 系統定義角色”](#)。

所需的許可

以下是 ALTER 的必要許可ROLE：

- 超級使用者
- 具有 ALTERROLE許可的使用者

語法

```
ALTER ROLE role [ WITH ]
  { { RENAME TO role } | { OWNER TO user_name } }[, ...]
```

```
[ EXTERNALID TO external_id ]
```

參數

角色

要修改的角色名稱。

RENAME 至

角色的新名稱。

OWNER TO *user_name*

角色的新擁有者。

EXTERNALID 至 *external_id*

角色的新外部 ID，與身分提供者相關聯。如需詳細資訊，請參閱 [Amazon Redshift 的原生身分提供者 \(IdP\) 聯合](#)。

範例

下列範例會將角色的名稱從 `sample_role1` 變更為 `sample_role2`。

```
ALTER ROLE sample_role1 WITH RENAME TO sample_role2;
```

下列範例會變更角色的擁有者。

```
ALTER ROLE sample_role1 WITH OWNER TO user1
```

的語法 ALTER ROLE 類似於 ALTER PROCEDURE 以下內容。

```
ALTER PROCEDURE first_quarter_revenue(bigint, numeric) RENAME TO quarterly_revenue;
```

下列範例會將程序的擁有者變更為 `etl_user`。

```
ALTER PROCEDURE quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

下列範例會使用與身分提供者相關聯的新外部 ID 來更新角色 `sample_role1`。

```
ALTER ROLE sample_role1 EXTERNALID TO "XYZ456";
```

ALTER PROCEDURE

重新命名程序或變更擁有者。程序名稱和資料類型 (或簽章) 都是必要的。只有擁有者或超級使用者才能重新命名程序。只有超級使用者才能變更程序的擁有者。

語法

```
ALTER PROCEDURE sp_name [ ( [ [ argname ] [ argmode ] argtype [, ...] ] ) ]  
    RENAME TO new_name
```

```
ALTER PROCEDURE sp_name [ ( [ [ argname ] [ argmode ] argtype [, ...] ] ) ]  
    OWNER TO { new_owner | CURRENT_USER | SESSION_USER }
```

參數

sp_name

要修改的程序的名称。在目前的搜尋路徑中僅指定程序的名称，或使用格式 `schema_name.sp_procedure_name` 來使用專屬結構描述。

[*argname*] [*argmode*] *argtype*

引數名称、引數模式和資料類型的清單。只有用於識別預存程序的輸入資料類型是必要的。或者，您可以提供用於建立程序的完整簽章，包括輸入和輸出參數及其模式。

new_name

預存程序的新名称。

new_owner | CURRENT_USER | SESSION_USER

預存程序的新擁有者。

範例

下列範例會將程序的名称從 `first_quarter_revenue` 變更為 `quarterly_revenue`。

```
ALTER PROCEDURE first_quarter_revenue(volume INOUT bigint, at_price IN numeric,
```

```
result OUT int) RENAME TO quarterly_revenue;
```

此範例相當於下列命令。

```
ALTER PROCEDURE first_quarter_revenue(bigint, numeric) RENAME TO quarterly_revenue;
```

下列範例會將程序的擁有者變更為 `etl_user`。

```
ALTER PROCEDURE quarterly_revenue(bigint, numeric) OWNER TO etl_user;
```

ALTER SCHEMA

變更現有結構描述的定義。使用此命令可重新命名結構描述或變更結構描述的擁有者。例如，重新命名現有的結構描述，以便在您打算建立該結構描述的新版本時，保留該結構描述的備份複本。如需結構描述的相關資訊，請參閱 [CREATE SCHEMA](#)。

若要檢視設定的結構描述配額，請參閱 [SVV_SCHEMA_QUOTA_STATE](#)。

若要檢視超出結構描述配額的記錄，請參閱 [STL_SCHEMA_QUOTA_VIOLATIONS](#)。

所需權限

以下是 ALTER 的必要權限SCHEMA：

- 超級使用者
- 具有 ALTERSCHEMA權限的使用者
- 結構描述擁有者

當您變更結構描述名稱時，請務必更新使用舊名稱的物件 (例如預存程序或具體化視觀表)，才能使用新名稱。

語法

```
ALTER SCHEMA schema_name
{
  RENAME TO new_name |
  OWNER TO new_owner |
  QUOTA { quota [MB | GB | TB] | UNLIMITED }
```

```
}
```

參數

schema_name

要修改的資料庫結構描述名稱。

RENAME 至

重新命名結構描述的子句。

new_name

結構描述的新名稱。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

OWNER 至

變更結構描述擁有者的子句。

new_owner

結構描述的新擁有者。

QUOTA

所指定之結構描述可以使用的磁碟空間數上限。此空間是所指定之結構描述下所有資料表的集體大小。Amazon Redshift 會將選取的值轉換為 MB。GB 是您未指定值時的預設測量單位。

如需設定結構描述配額的相關資訊，請參閱 [CREATE SCHEMA](#)。

範例

下列範例會將SALES結構描述重新命名為 US_SALES。

```
alter schema sales  
rename to us_sales;
```

下列範例會將 US_SALES 結構描述的所有權授予使用者 DWUSER。

```
alter schema us_sales  
owner to dwuser;
```

下列範例會將配額變更為 300 GB，並移除配額。

```
alter schema us_sales QUOTA 300 GB;  
alter schema us_sales QUOTA UNLIMITED;
```

ALTER SYSTEM

變更 Amazon Redshift 叢集或 Redshift Serverless 工作群組的系統層級組態選項。

所需權限

下列其中一個使用者類型可以執行 ALTERSYSTEM命令：

- 超級使用者
- 管理員使用者

語法

```
ALTER SYSTEM SET system-level-configuration = {true| t | on | false | f | off}
```

參數

system-level-configuration

系統層級組態 有效值：data_catalog_auto_mount 和 metadata_security。

{true| t | on | false | f | off}

用於啟用或停用系統層級組態的值。true、t 或 on 表示要啟用組態。false、f 或 off 表示要停用組態。

使用須知

對於佈建的叢集，data_catalog_auto_mount 的變更會在下次重新啟動叢集時生效。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[重新啟動叢集](#)。

對於無伺服器工作群組，data_catalog_auto_mount 的變更不會立即生效。

範例

下列範例會開啟自動掛載 AWS Glue Data Catalog。


```
ALTER SYSTEM SET data_catalog_auto_mount = true;
```

下列範例會開啟中繼資料安全性。

```
ALTER SYSTEM SET metadata_security = true;
```

設定預設身分命名空間

此範例專用於與身分提供者合作。您可以整合 Redshift 與 IAM Identity Center 和身分提供者，以集中 Redshift 和其他 AWS 服務的身分管理。

下列範例示範如何設定系統的預設身分命名空間。隨後執行此操作會更輕鬆地執行 GRANT 和 CREATE 陳述式，因為您不需要將命名空間作為每個身分的字首。

```
ALTER SYSTEM SET default_identity_namespace = 'MYC0';
```

執行命令之後，您可以執行如下所示的陳述式：

```
GRANT SELECT ON TABLE mytable TO alice;

GRANT UPDATE ON TABLE mytable TO salesrole;

CREATE USER bob password 'md50c983d1a624280812631c5389e60d48c';
```

設定預設身分命名空間的效果是，每個身分都不需要它作為字首。在此範例中，alice 會取代為 MYC0:alice。這會在包含任何身分時發生。如需搭配 Redshift 使用身分提供者的詳細資訊，請參閱 [將 Redshift 與 IAM Identity Center 連線，為使用者提供單一登入體驗。](#)

如需使用 IAM Identity Center 進行 Redshift 組態相關設定的詳細資訊，請參閱 [SET](#) 和 [ALTER IDENTITY PROVIDER](#)。

ALTER TABLE

此命令會變更 Amazon Redshift 資料表或 Amazon Redshift Spectrum 外部資料表的定義。此命令會更新 [CREATE TABLE](#) 或 [CREATE EXTERNAL TABLE](#) 設定的值和屬性。

您無法 ALTER TABLE 在交易區塊內的外部資料表上執行（BEGIN ... END）。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

ALTER TABLE 會鎖定用於讀取和寫入操作的資料表，直到包含ALTER TABLE 操作的交易完成為止，除非文件中特別說明您可以在資料表變更時查詢資料或執行資料表上的其他操作。

所需權限

修改資料表的使用者需要適當的權限，才能成功執行命令。視ALTER TABLE 命令而定，需要下列其中一個權限。

- 超級使用者
- 具有 ALTER TABLE 權限的使用者
- 具有結構描述上USAGE 權限的資料表擁有者

語法

```
ALTER TABLE table_name
{
ADD table_constraint
| DROP CONSTRAINT constraint_name [ RESTRICT | CASCADE ]
| OWNER TO new_owner
| RENAME TO new_name
| RENAME COLUMN column_name TO new_name
| ALTER COLUMN column_name TYPE updated_varchar_data_type_size
| ALTER COLUMN column_name ENCODE new_encode_type
| ALTER COLUMN column_name ENCODE encode_type,
| ALTER COLUMN column_name ENCODE encode_type, .....;
| ALTER DISTKEY column_name
| ALTER DISTSTYLE ALL
| ALTER DISTSTYLE EVEN
| ALTER DISTSTYLE KEY DISTKEY column_name
| ALTER DISTSTYLE AUTO
| ALTER [COMPOUND] SORTKEY ( column_name [,...] )
| ALTER SORTKEY AUTO
| ALTER SORTKEY NONE
| ALTER ENCODE AUTO
| ADD [ COLUMN ] column_name column_type
  [ DEFAULT default_expr ]
  [ ENCODE encoding ]
  [ NOT NULL | NULL ]
  [ COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE } ] |
| DROP [ COLUMN ] column_name [ RESTRICT | CASCADE ]
| ROW LEVEL SECURITY { ON | OFF } [ CONJUNCTION TYPE { AND | OR } ] [ FOR DATASHARES ]}
```

where *table_constraint* is:

```
[ CONSTRAINT constraint_name ]
{ UNIQUE ( column_name [, ... ] )
| PRIMARY KEY ( column_name [, ... ] )
| FOREIGN KEY ( column_name [, ... ] )
  REFERENCES reftable [ ( refcolumn ) ]}
```

The following options apply only to external tables:

```
SET LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }
| SET FILE FORMAT format |
| SET TABLE PROPERTIES ('property_name'='property_value')
| PARTITION ( partition_column=partition_value [, ...] )
  SET LOCATION { 's3://bucket/folder' | 's3://bucket/manifest_file' }
| ADD [IF NOT EXISTS]
  PARTITION ( partition_column=partition_value [, ...] ) LOCATION
  { 's3://bucket/folder' | 's3://bucket/manifest_file' }
  [, ... ]
| DROP PARTITION ( partition_column=partition_value [, ...] )
```

若要減少執行ALTER TABLE命令的時間，您可以合併ALTER TABLE命令的某些子句。

Amazon Redshift 支援下列子ALTER TABLE句組合：

```
ALTER TABLE tablename ALTER SORTKEY (column_list), ALTER DISTKEY column_Id;
ALTER TABLE tablename ALTER DISTKEY column_Id, ALTER SORTKEY (column_list);
ALTER TABLE tablename ALTER SORTKEY (column_list), ALTER DISTSTYLE ALL;
ALTER TABLE tablename ALTER DISTSTYLE ALL, ALTER SORTKEY (column_list);
```

參數

table_name

要修改的資料表名稱。僅指定資料表的名稱，或使用格式 *schema_name.table_name* 來使用專屬結構描述。外部資料表必須以外部結構描述名稱限定。如果您使用 ALTER TABLE陳述式重新命名檢視或變更其擁有者，也可以指定檢視名稱。資料表名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 個多位元組字元，最多四個位元組。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

ADD table_constraint

新增指定限制條件至資料表的子句。如需有效 table_constraint 值得說明，請參閱 [CREATE TABLE](#)。

Note

您無法將主索引鍵限制條件新增至可為 null 的資料欄。如果資料欄最初是使用 NOTNULL 限制條件建立的，您可以新增主金鑰限制條件。

DROP CONSTRAINT traint_name

從資料表中刪除具名限制條件的子句。若要刪除限制條件，請指定限制條件名稱，而非限制條件類型。若要檢視資料表限制條件名稱，請執行下列查詢。

```
select constraint_name, constraint_type
from information_schema.table_constraints;
```

RESTRICT

僅移除指定限制條件的子句。RESTRICT 是 DROP 的選項 CONSTRAINT。RESTRICT 無法與 搭配使用 CASCADE。

CASCADE

此子句會移除指定限制條件及相依於該限制條件的任何項目。CASCADE 是 DROP 的選項 CONSTRAINT。CASCADE 無法與 搭配使用 RESTRICT。

OWNER 至 new_擁有者

此子句會將資料表 (或檢視) 的擁有者變更為 new_owner 值。

RENAME 至 new_name

此子句會將資料表 (或檢視) 重新命名為 new_name 中指定的值。資料表名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。

您無法將永久資料表重新命名為開頭為 '#' 的名稱。開頭為 '#' 的資料表名稱代表暫時資料表。

您無法重新命名外部資料表。

ALTER COLUMN column_name TYPE updated_varchar_data_type_size

變更定義為VARCHAR資料類型之資料欄大小的子句。此子句僅支援變更VARCHAR資料類型的大小。考量下列限制：

- 您無法變更具有壓縮編碼 BYTEDICT、RUNLENGTH、TEXT255或 的資料欄TEXT32K。
- 您無法將大小降低至小於現有資料的大小上限。
- 您無法修改含預設值的資料欄。
- 您無法使用 UNIQUE、PRIMARY KEY或 FOREIGN 變更資料欄KEY。
- 您無法變更交易區塊中的資料欄 (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

ALTER COLUMN column_name ENCODE new_encode_type

變更資料欄壓縮編碼的子句。如果您為資料欄指定壓縮編碼，則資料表不再設定為 ENCODE AUTO。如需壓縮編碼的詳細資訊，請參閱 [資料欄壓縮以減少儲存資料的大小](#)。

當您變更資料欄的壓縮編碼時，資料表仍可供查詢。

考量下列限制：

- 您無法將資料欄的編碼修改為目前資料行定義的編碼。
- 您無法在具有交錯排序索引鍵的資料表中修改資料行的編碼。

ALTER COLUMN column_name ENCODE encode_type 、 ALTER COLUMN column_name ENCODE encode_type 、 ；

該子句可變更單一命令中多個資料欄的壓縮編碼。如需壓縮編碼的詳細資訊，請參閱 [資料欄壓縮以減少儲存資料的大小](#)。

當您變更資料欄的壓縮編碼時，資料表仍可供查詢。

考量下列限制：

- 您無法在單一命令中多次將資料欄變更為相同或不同的編碼類型。
- 您無法將資料欄的編碼修改為目前資料行定義的編碼。
- 您無法在具有交錯排序索引鍵的資料表中修改資料行的編碼。

ALTER DISTSTYLE ALL

將資料表的現有分佈樣式變更為 ALL 的子句。考慮下列各項：

- DISTSTYLE、ALTER SORTKEY和 ALTER VACUUM無法在相同的資料表上同時執行。
 - 如果 VACUUM 目前正在執行，則執行 會ALTERDISTSTYLEALL傳回錯誤。

- 如果 ALTER DISTSTYLE ALL 正在執行，則背景真空不會在資料表上啟動。
- 具有交錯排序索引鍵和暫存資料表的資料表不支援 ALTERDISTSTYLEALL 命令。
- 如果分佈樣式先前定義為 AUTO，則資料表不再是自動資料表最佳化的候選者。

如需 DISTSTYLE 的詳細資訊 ALL，請參閱 [CREATE TABLE](#)。

ALTER DISTSTYLE EVEN

將資料表的現有分佈樣式變更為 EVEN 的子句。考慮下列各項：

- DISTSYTLE、ALTER SORTKEY 和 ALTER VACUUM 無法在相同的資料表上同時執行。
 - 如果 VACUUM 目前正在執行，則執行 會 ALTERDISTSTYLEEVEN 傳回錯誤。
 - 如果 ALTER DISTSTYLE EVEN 正在執行，則背景真空不會在資料表上啟動。
- 具有交錯排序索引鍵和暫存資料表的資料表不支援 ALTERDISTSTYLEEVEN 命令。
- 如果分佈樣式先前定義為 AUTO，則資料表不再是自動資料表最佳化的候選者。

如需 DISTSTYLE 的詳細資訊 EVEN，請參閱 [CREATE TABLE](#)。

ALTER DISTKEY column_name 或 ALTER DISTSTYLE KEY DISTKEY column_name

會變更用作資料表的分佈索引鍵的資料欄的子句。考慮下列各項：

- VACUUM 和 ALTER DISTKEY 無法在相同的資料表上同時執行。
 - 如果 VACUUM 已在執行中，則 會 ALTERDISTKEY 傳回錯誤。
 - 如果 ALTER DISTKEY 正在執行，則背景真空不會在資料表上啟動。
 - 如果 ALTER DISTKEY 正在執行，則前景真空會傳回錯誤。
- 您一次只能在資料表上執行一個 ALTERDISTKEY 命令。
- 具有交錯排序索引鍵的資料表不支援 ALTERDISTKEY 命令。
- 如果分佈樣式先前定義為 AUTO，則資料表不再是自動資料表最佳化的候選者。

指定 DISTSTYLE 時 KEY，資料會由 DISTKEY 資料欄中的值分佈。如需 的詳細資訊 DISTSTYLE，請參閱 [CREATE TABLE](#)。

ALTER DISTSTYLE AUTO

將資料表的現有分佈樣式變更為 的子句 AUTO。

當您將分佈樣式變更為 時 AUTO，資料表的分佈樣式會設定為下列：

- 具有 的小資料表 DISTSTYLEALL 會轉換為 AUTO (ALL)。
- 具有 的小資料表 DISTSTYLEEVEN 會轉換為 AUTO (ALL)。
- 具有 的小資料表 DISTSTYLEKEY 會轉換為 AUTO (ALL)。

- 具有的大型資料表DISTSTYLEALL會轉換為 AUTO (EVEN) 。
- 具有的大型資料表DISTSTYLEEVEN會轉換為 AUTO (EVEN) 。
- 具有的大型資料表DISTSTYLEKEY會轉換為 AUTO (KEY) DISTKEY，並保留。在這種情況下，Amazon Redshift 不會對資料表進行任何變更。

如果 Amazon Redshift 判斷新的分佈樣式或索引鍵可以改善查詢效能，那麼 Amazon Redshift 可能會在未來變更資料表的分佈樣式或索引鍵。例如，Amazon Redshift 可能會將具有 AUTO (KEY) DISTSTYLE 的資料表轉換為 AUTO (EVEN)，反之亦然。如需修改分佈索引鍵時行為 (包括資料重新分佈和鎖定) 的相關資訊，請參閱 [Amazon Redshift Advisor 建議](#)。

如需 DISTSTYLE 的詳細資訊AUTO，請參閱 [CREATE TABLE](#)。

若要檢視資料表的分佈樣式，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱[SVV_TABLE_INFO](#)。若要檢視資料表的 Amazon Redshift Advisor 建議，請查詢 SVV_ALTER_TABLE_RECOMMENDATIONS 系統目錄檢視。如需詳細資訊，請參閱[SVV_ALTER_TABLE_RECOMMENDATIONS](#)。若要檢視 Amazon Redshift 採取的動作，請查詢 SVL_AUTO_WORKER_ACTION系統目錄檢視。如需詳細資訊，請參閱[SVL_AUTO_WORKER_ACTION](#)。

ALTER 【COMPOUND】 SORTKEY (column_name 【, ...】)

此子句可變更或新增用於資料表的排序索引鍵。ALTER SORTKEY 不支援暫存資料表。

當您變更排序索引鍵時，新排序索引鍵或原始排序索引鍵中欄的壓縮編碼可能會變更。如果沒有明確定義資料表的編碼，則 Amazon Redshift 會自動指定壓縮編碼，如下所示：

- 定義為排序索引鍵的資料欄會被指派RAW壓縮。
- 定義為 BOOLEAN、 REAL或 DOUBLE PRECISION 資料類型的資料欄會被指派RAW壓縮。
- 定義為 SMALLINT、 INTEGER、 BIGINT、 DECIMAL、 DATE、 TIME、 TIMESTAMP、 TIMETZ或的資料欄TIMESTAMPTZ會指派AZ64壓縮。
- 定義為 CHAR或 VARCHAR 且被指派LZO壓縮的資料欄。

考慮下列各項：

- 每個資料表的排序索引鍵最多可定義 400 個資料欄。
- 您可以將交錯排序索引鍵修改為複合排序索引鍵或無排序索引鍵。但是，您無法將複合排序索引鍵修改為交錯排序索引鍵。
- 如果排序索引鍵先前定義為 AUTO，則資料表不再是自動資料表最佳化的候選者。
- Amazon Redshift 建議對定義為排序索引鍵的資料欄使用RAW編碼 (不壓縮)。當您變更資料欄以將其選擇為排序索引鍵時，資料欄的壓縮會變更為RAW壓縮 (無壓縮)。這可能會增加資料

表所需的儲存空間。資料表大小的增加幅度取決於特定的資料表定義和資料表內容。如需壓縮的相關資訊，請參閱 [壓縮編碼](#)。

當資料載入資料表之後，資料就會依據排序索引鍵的排序載入。當您修改排序索引鍵時，Amazon Redshift 會重新排列資料。如需的詳細資訊 SORTKEY，請參閱 [CREATE TABLE](#)。

ALTER SORTKEY AUTO

將目標資料表的排序索引鍵變更或新增至的子句 AUTO。ALTER SORTKEY AUTO 不支援暫存資料表。

當您將排序索引鍵變更為時 AUTO，Amazon Redshift 會保留資料表的現有排序索引鍵。

如果 Amazon Redshift 判斷新的排序索引鍵可以改善查詢效能，那麼 Amazon Redshift 可能會在未來變更資料表的排序索引鍵。

如需 SORTKEY 的詳細資訊 AUTO，請參閱 [CREATE TABLE](#)。

若要檢視資料表的排序索引鍵，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱 [SVV_TABLE_INFO](#)。若要檢視資料表的 Amazon Redshift Advisor 建議，請查詢 SVV_ALTER_TABLE_RECOMMENDATIONS 系統目錄檢視。如需詳細資訊，請參閱 [SVV_ALTER_TABLE_RECOMMENDATIONS](#)。若要檢視 Amazon Redshift 採取的動作，請查詢 SVL_AUTO_WORKER_ACTION 系統目錄檢視。如需詳細資訊，請參閱 [SVL_AUTO_WORKER_ACTION](#)。

ALTER SORTKEY NONE

移除目標資料表的排序索引鍵的子句。

如果排序索引鍵先前定義為 AUTO，則資料表不再是自動資料表最佳化的候選者。

ALTER ENCODE AUTO

將目標資料表資料欄的編碼類型變更為的子句 AUTO。當您將編碼變更為時 AUTO，Amazon Redshift 會保留資料表中資料欄的現有編碼類型。然後，如果 Amazon Redshift 判斷新的編碼類型可以改善查詢效能，Amazon Redshift 就可以變更資料表資料欄的編碼類型。

如果您修改一個或多個欄以指定編碼，則 Amazon Redshift 不會再自動調整資料表中所有資料欄的編碼。這些資料欄會保留目前的編碼設定。

下列動作不會影響資料表 ENCODE AUTO 的設定：

- 重新命名資料表。
- 變更資料表的 DISTSTYLE 或 SORTKEY 設定。

- 使用 ENCODE 設定新增或捨棄資料欄。
- 使用 COPY 命令 COMPUPDATE 的選項。如需詳細資訊，請參閱 [資料載入操作](#)。

若要檢視資料表的編碼，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱 [SVV_TABLE_INFO](#)。

RENAME COLUMN column_name TO new_name

此子句會將資料欄重新命名為 new_name 中指定的值。資料欄名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

ADD 【 COLUMN 】 column_name

將指定名稱的資料欄新增至資料表的子句。每個 ALTER TABLE 陳述式只能新增一欄。

您無法新增 欄，該欄是 資料表的分佈索引鍵（DISTKEY）或排序索引鍵（SORTKEY）。

您無法使用 ALTER TABLE ADD COLUMN 命令來修改下表和資料欄屬性：

- UNIQUE
- PRIMARY KEY
- REFERENCES（外部金鑰）
- IDENTITY 或 GENERATED BY DEFAULT AS IDENTITY

資料欄名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。單一資料表中可定義的資料欄數目上限為 1,600 個。

以下限制適用於新增資料欄至外部資料表時：

- 您無法將資料欄新增至具有資料欄限制 DEFAULT、ENCODE、NOT NULL 或 的外部資料表 NULL。
- 您無法將資料欄新增至使用 AVRO 檔案格式定義的外部資料表。
- 若啟用虛擬資料欄，則單一外部資料表中可定義的資料欄數目上限為 1,598 個。若未啟用虛擬資料欄，則單一資料表中可定義的資料欄數目上限為 1,600 個。

如需詳細資訊，請參閱 [CREATE EXTERNAL TABLE](#)。

column_type

要新增之資料欄的資料類型。對於 CHAR 和 VARCHAR 資料欄，您可以使用 MAX 關鍵字，而不是宣告長度上限。MAX 會將 的最大長度設定為 4,096 位元組，CHAR 或 設定為 65,535 位元組 VARCHAR。GEOMETRY 物件的大小上限為 1,048,447 個位元組。

如需有關 Amazon Redshift 支援資料類型的資訊，請參閱 [資料類型](#)。

DEFAULT default_expr

此子句會指派資料欄的預設資料值。default_expr 的資料類型必須符合資料欄的資料類型。DEFAULT 值必須是無變數表達式。不允許子查詢、目前資料表中其他資料欄的交叉參考，以及使用者定義的功能。

default_expr 用於任何未指定資料欄值INSERT的操作。若未指定預設值，則資料欄的預設值為null。

如果COPY操作在具有 DEFAULT值和NOTNULL限制條件的資料欄上遇到 null 欄位，則COPY命令會插入 default_expr 的值。

DEFAULT 不支援外部資料表。

ENCODE 編碼

資料欄的壓縮編碼。根據預設，如果您未為資料表中的任何資料欄指定壓縮編碼，或指定資料表 ENCODEAUTO的選項，Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。

如果您為資料表中的任何資料欄指定壓縮編碼，或如果您未指定資料表ENCODEAUTO的選項，Amazon Redshift 會自動將壓縮編碼指派給您未指定壓縮編碼的資料欄，如下所示：

- 根據預設，暫存資料表中的所有資料欄都會指派RAW壓縮。
- 定義為排序索引鍵的資料欄會被指派RAW壓縮。
- 定義為 BOOLEAN、REAL、DOUBLEPRECISION、 GEOMETRY或 GEOGRAPHY資料類型的資料欄會被指派RAW壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMESTAMP、TIMETZ或 的資料欄TIMESTAMPTZ會指派AZ64壓縮。
- 定義為 CHAR、 VARCHAR或 的資料欄VARBYTE會被指派LZO壓縮。

Note

如果您不想壓縮資料欄，請明確指定RAW編碼。

支援以下 [compression encodings \(p. 54\)](#)：

- AZ64
- BYTEDICT
- DELTA
- DELTA32K

- LZO
- MOSTLY8
- MOSTLY16
- MOSTLY32
- RAW (無壓縮)
- RUNLENGTH
- TEXT255
- TEXT32K
- ZSTD

ENCODE 不支援外部資料表。

NOT NULL | NULL

NOT NULL 指定不允許資料欄包含 null 值。NULL是預設值，指定資料欄接受 null 值。

NOT NULL 和 NULL不支援外部資料表。

COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }

指定資料欄上的字串搜尋或比較是 CASE_SENSITIVE 還是 CASE_ 的子句INSENSITIVE。預設值與資料庫目前的區分大小寫設定相同。

若要尋找資料庫定序資訊，請使用下列命令：

```
SELECT db_collation();

db_collation
-----
case_sensitive
(1 row)
```

DROP 【 COLUMN 】 column_name

要從資料表中刪除的資料欄名稱。

您無法刪除資料表中的最後一個資料欄。資料表至少必須有一個資料欄。

您無法捨棄屬於 資料表分發金鑰 (DISTKEY) 或排序金鑰 (SORTKEY) 的資料欄。如果 RESTRICT資料欄有任何相依物件，例如檢視、主金鑰、外部金鑰或UNIQUE限制，則的預設行為 DROPCOLUMN為。

以下限制適用於從外部資料表刪除資料欄時：

- 若資料欄做為分割區使用，則無法從外部資料表刪除該資料欄。
- 您無法從使用AVRO檔案格式定義的外部資料表捨棄資料欄。
- RESTRICT 外部資料表CASCADE會忽略 和。
- 除非您將政策捨棄或中斷連結，否則您無法捨棄政策定義中參照的政策資料表資料欄。當指定CASCADE選項時，這也適用。您可以捨棄政策資料表中的其他資料欄。

如需詳細資訊，請參閱[CREATE EXTERNAL TABLE](#)。

RESTRICT

與 DROP 搭配使用時COLUMN，RESTRICT表示不會捨棄要捨棄的資料欄，在這些情況下：

- 如果定義的檢視參考要捨棄的資料欄
- 如果外部索引鍵參考資料欄
- 如果資料欄為分段索引鍵的一部分

RESTRICT 無法與 搭配使用CASCADE。

RESTRICT 外部資料表CASCADE會忽略 和。

CASCADE

與 DROP 搭配使用時COLUMN，會移除指定的資料欄，以及與該資料欄相關的任何項目。CASCADE 無法與 搭配使用RESTRICT。

RESTRICT 外部資料表CASCADE會忽略 和。

以下選項只適用於外部資料表。

```
SET LOCATION { 's3 : //bucket/folder /' | 's3 : //bucket/manifest_file ' }
```

包含資料檔案的 Amazon S3 資料夾路徑，或包含 Amazon S3 物件路徑清單的資訊清單檔案。儲存貯體必須位於 Amazon Redshift 叢集所在的同一 AWS 區域。如需支援的 AWS 區域清單，請參閱 [Amazon Redshift Spectrum 限制](#)。如需使用資訊清單檔案的詳細資訊，請參閱 CREATE EXTERNAL TABLE [參數](#) LOCATION參考中的。

SET FILE FORMAT 格式


外部資料檔案的檔案格式。

有效格式如下：

- AVRO
- PARQUET
- RCFILE
- SEQUENCEFILE
- TEXTFILE

SET TABLE PROPERTIES ('property_name '='property_value ')

此子句會設定外部資料表的資料表屬性的資料表定義。

 Note

資料表屬性區分大小寫。

'numRows'='row_count '

設定資料表定義 numRows 值的 屬性。若要明確更新外部資料表的統計資料，請設定 numRows 屬性以指示資料表的大小。Amazon Redshift 不會分析外部資料表來產生查詢最佳化工具用來產生查詢計劃的資料表統計資料。如果未設定外部資料表的資料表統計資料，則 Amazon Redshift 會產生查詢執行計畫。此計畫是根據外部資料表為較大資料表，而本機資料表為較小資料表的假設。

'skip.header.line.count'='line_count'


此屬性會設定每個來源檔案開頭要略過的資料列數。

PARTITION (partition_column =partition_value 【 , ...】 SET LOCATION { 's3 : //bucket /folder ' | 's3 : //bucket /manifest_file ' }

此子句會設定一個或多個分割區資料欄的新位置。

ADD 【 IF NOT EXISTS 】 PARTITION (partition_column =partition_value 【 , ...】) LOCATION { 's3 : //bucket /folder ' | 's3 : //bucket /manifest_file ' } 【 , ... 】

新增一或多個分割區的字句。您可以使用單一 ... ALTER TABLE ADD陳述式來指定多個 PARTITION子句。

 Note

如果您使用 AWS Glue 目錄，則可以使用單一ALTER TABLE陳述式新增最多 100 個分割區。

IF NOT EXISTS子句指出，如果指定的分割區已存在，則命令不應進行任何變更。也指出命令應該傳回分割區已存在的訊息，而不是在發生錯誤的情況下終止。此子句在指令碼時很有用，因此如果ALTER TABLE嘗試新增已存在的分割區，指令碼不會失敗。

DROP PARTITION (partition_column =partition_value 【、...】)

刪除指定分割區的字句。刪除分割區只會修改外部資料表中繼資料。Amazon S3 上的資料不受影響。

ROW LEVEL SECURITY { ON | OFF } 【 CONJUNCTION TYPE { AND | 或 } 】 【 FOR DATASHARES 】

開啟或關閉關係的資料列層級安全性的字句。

開啟關係的資料列層級安全性時，您只能讀取資料列層級安全政策允許您存取的資料列。如果沒有任何政策授予您關係存取權，您就無法看到關聯中的任何資料列。只有具有 sys:secadmin 角色的超級使用者和使用者或角色才能設定ROWLEVELSECURITY子句。如需詳細資訊，請參閱[資料列層級安全性](#)。

- 【 CONJUNCTION TYPE { AND | OR } 】

一個子句，讓您為關係選擇資料列層級安全政策的結合類型。當多個資料列層級安全政策連接至關係時，您可以將政策與 AND 或 OR 子句結合。根據預設，Amazon Redshift 會將RLS政策與 AND 子句結合。具有 sys:secadmin 角色的超級使用者、使用者或角色可以使用此子句來定義關係之資料列層級安全政策的組合類型。如需詳細資訊，請參閱[每個使用者結合多個政策](#)。

- FOR DATASHARES

字句，用於判斷 RLS受保護的關係是否可以透過資料共用存取。根據預設，無法透過資料共用存取 RLS保護的關係。使用此子句執行的ALTER TABLE ROWLEVELSECURITY命令只會影響關係的資料共用可存取性屬性。ROW LEVEL SECURITY 屬性不會變更。

如果您讓 RLS受保護的關係可透過資料共用存取，則該關係在取用者端資料共用資料庫中沒有資料列層級安全性。關係會保留其RLS屬性在生產者端。

範例

如需示範如何使用 ALTER TABLE命令的範例，請參閱以下內容。

- [ALTER TABLE 範例](#)
- [ALTER EXTERNAL TABLE 範例](#)
- [ALTER TABLE ADD 和 DROP COLUMN 範例](#)

ALTER TABLE 範例

下列範例示範 ALTER TABLE 命令的基本用量。

重新命名資料表或檢視

下列命令會將USERS資料表重新命名為 USERS_BKUP：

```
alter table users
rename to users_bkup;
```

您也可以使用此類型的命令來重新命名檢視。

變更資料表或檢視的擁有者

下列命令會將VENUE資料表擁有者變更為使用者 DWUSER：

```
alter table venue
owner to dwuser;
```

以下命令會建立檢視，然後變更其擁有者：

```
create view vdate as select * from date;
alter table vdate owner to vuser;
```

重新命名欄位

下列命令會將VENUE資料表中的VENUESEATS資料欄重新命名為 VENUESIZE：

```
alter table venue
rename column venueseats to venuesize;
```

丟棄資料表限制條件

若要捨棄資料表限制條件，像是主索引鍵、外部索引鍵或唯一限制條件，請先找到限制條件的內部名稱。然後在ALTER TABLE命令中指定限制名稱。下列範例會尋找CATEGORY資料表的限制，然後捨棄名為 的主索引鍵category_pkey。

```
select constraint_name, constraint_type
from information_schema.table_constraints
```

```
where constraint_schema = 'public'  
and table_name = 'category';
```

```
constraint_name | constraint_type  
-----+-----  
category_pkey  | PRIMARY KEY
```

```
alter table category  
drop constraint category_pkey;
```

變更資料VARCHAR欄

若要節省儲存空間，您可以先使用資料VARCHAR欄定義資料表，其中包含目前資料需求所需的大小下限。之後若要容納較長的字串，則可以變更資料表以增加資料欄的大小。

下列範例會將EVENTNAME資料欄的大小增加為 VARCHAR (300)。

```
alter table event alter column eventname type varchar(300);
```

修改資料欄的壓縮編碼。

您可以修改資料欄的壓縮編碼。您可以在下方找到一組示範此方法的範例。這些範例的資料表定義如下。

```
create table t1(c0 int encode lzo, c1 bigint encode zstd, c2 varchar(16) encode lzo, c3  
varchar(32) encode zstd);
```

下列陳述式會將資料欄 c0 的壓縮編碼從LZO編碼變更為AZ64編碼。

```
alter table t1 alter column c0 encode az64;
```

下列陳述式會將資料欄 c1 的壓縮編碼從 Zstandard 編碼變更為AZ64編碼。

```
alter table t1 alter column c1 encode az64;
```

下列陳述式會將資料欄 c2 的壓縮編碼從LZO編碼變更為位元組字典編碼。

```
alter table t1 alter column c2 encode bytedict;
```

下列陳述式會將資料欄 c3 的壓縮編碼從 Zstandard 編碼修改為 Runlength 編碼。


```
alter table t1 alter column c3 encode runlength;
```

變更DISTSTYLEKEYDISTKEY資料欄

下列範例示範如何變更資料表DISTKEY的 DISTSTYLE和。

建立具有EVEN分佈樣式的資料表。SVV_TABLE_INFO 檢視顯示 DISTSTYLE是 EVEN。

```
create table inventory(
  inv_date_sk int4 not null ,
  inv_item_sk int4 not null ,
  inv_warehouse_sk int4 not null ,
  inv_quantity_on_hand int4
) diststyle even;

Insert into inventory values(1,1,1,1);

select "table", "diststyle" from svv_table_info;
```

table	diststyle
inventory	EVEN

將資料表變更為 DISTKEY inv_warehouse_sk。SVV_TABLE_INFO 檢視會將資料inv_warehouse_sk欄顯示為產生的分發金鑰。

```
alter table inventory alter diststyle key distkey inv_warehouse_sk;

select "table", "diststyle" from svv_table_info;
```

table	diststyle
inventory	KEY(inv_warehouse_sk)

將資料表變更為 DISTKEY inv_item_sk。SVV_TABLE_INFO 檢視會將資料inv_item_sk欄顯示為產生的分發金鑰。

```
alter table inventory alter distkey inv_item_sk;

select "table", "diststyle" from svv_table_info;
```

```

table | diststyle
-----+-----
inventory | KEY(inv_item_sk)

```

將資料表變更為 DISTSTYLE ALL

下列範例示範如何將資料表變更為 DISTSTYLE ALL。

建立具有 EVEN 分佈樣式的資料表。SVV_TABLE_INFO 檢視顯示 DISTSTYLE 是 EVEN。

```

create table inventory(
  inv_date_sk int4 not null ,
  inv_item_sk int4 not null ,
  inv_warehouse_sk int4 not null ,
  inv_quantity_on_hand int4
) diststyle even;

Insert into inventory values(1,1,1,1);

select "table", "diststyle" from svv_table_info;

```

```

table | diststyle
-----+-----
inventory | EVEN

```

將資料表變更為 DISTSTYLE ALL。SVV_TABLE_INFO 檢視會顯示已變更的 DISTSYTLE。

```

alter table inventory alter diststyle all;

select "table", "diststyle" from svv_table_info;

```

```

table | diststyle
-----+-----
inventory | ALL

```

變更資料表 SORTKEY

您可以修改資料表以具有複合排序索引鍵或沒有排序鍵。

在下列資料表定義中，資料表 t1 是以交錯排序索引鍵來定義。

```

create table t1 (c0 int, c1 int) interleaved sortkey(c0, c1);

```

下列命令會將資料表從交錯排序索引鍵修改為複合排序索引鍵。

```
alter table t1 alter sortkey(c0, c1);
```

下列命令會修改資料表以移除交錯排序索引鍵。

```
alter table t1 alter sortkey none;
```

在下列資料表定義中，資料表 t1 是以 c0 作為交錯排序索引鍵來定義。

```
create table t1 (c0 int, c1 int) sortkey(c0);
```

下列命令會將資料表 t1 修改為複合排序索引鍵。

```
alter table t1 alter sortkey(c0, c1);
```

將資料表變更為 ENCODE AUTO

下列範例示範如何將資料表變更為 ENCODE AUTO。

此範例的資料表定義如下。資料欄 c0 是以編碼類型 定義 AZ64，而資料欄 c1 是以編碼類型 定義 LZ0。

```
create table t1(c0 int encode AZ64, c1 varchar encode LZ0);
```

對於此表格，下列陳述式會將編碼變更為 AUTO。

```
alter table t1 alter encode auto;
```

下列範例示範如何變更資料表以移除 ENCODE AUTO 設定。

此範例的資料表定義如下。資料表資料欄不需編碼即可定義。在此情況下，編碼預設為 ENCODE AUTO。

```
create table t2(c0 int, c1 varchar);
```

對於此表格，下列陳述式會將資料欄 c0 的編碼變更為 LZ0。資料表編碼不再設定為 ENCODE AUTO。

```
alter table t2 alter column c0 encode lzo;;
```

修改資料列層級安全性控制

資料表的下列命令會RLS關閉：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY OFF;
```

資料表的下列命令會RLS開啟：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;
```

下列命令會針對資料表RLS開啟，並使其可透過資料共用存取：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;  
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY FOR DATASHARES OFF;
```

下列命令會針對資料表RLS開啟，並使其無法透過資料共用存取：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON;  
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY FOR DATASHARES ON;
```

下列命令會RLS開啟，並將資料表的RLS聯合類型設定為 OR：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON CONJUNCTION TYPE OR;
```

下列命令會RLS開啟，並將資料表AND的RLS聯合類型設定為：

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON CONJUNCTION TYPE AND;
```

ALTER EXTERNAL TABLE 範例

下列範例使用位於美國東部（維吉尼亞北部）區域（us-east-1）中的 Amazon S3 儲存貯體，AWS 區域以及範例為 CREATE 建立的範例資料表TABLE。如需如何搭配外部資料表使用分割區的詳細資訊，請參閱 [分割 Redshift Spectrum 外部資料表](#)。

下列範例會將 SPECTRUM.SALES 外部 numRows 資料表的資料表屬性設定為 170,000 列。

```
alter table spectrum.sales  
set table properties ('numRows'='170000');
```

下列範例會變更 SPECTRUM.SALES 外部資料表的位置。

```
alter table spectrum.sales
set location 's3://redshift-downloads/ticket/spectrum/sales/';
```

下列範例會將 SPECTRUM.SALES 外部資料表的格式變更為 Parquet。

```
alter table spectrum.sales
set file format parquet;
```

下列範例會為資料表 SPECTRUM.SALES_ 新增一個分割區PART。

```
alter table spectrum.sales_part
add if not exists partition(saledate='2008-01-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01/';
```

下列範例為資料表 SPECTRUM.SALES_ 新增三個分割區PART。

```
alter table spectrum.sales_part add if not exists
partition(saledate='2008-01-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01/'
partition(saledate='2008-02-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-02/'
partition(saledate='2008-03-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-03/';
```

下列範例會變更 SPECTRUM.SALES_PART 以捨棄具有的分割區 saledate='2008-01-01'。

```
alter table spectrum.sales_part
drop partition(saledate='2008-01-01');
```

以下範例會為具有 saledate='2008-01-01' 的分割區設定新的 Amazon S3 路徑。

```
alter table spectrum.sales_part
partition(saledate='2008-01-01')
set location 's3://redshift-downloads/ticket/spectrum/sales_partition/
saledate=2008-01-01/';
```

下列範例會將 sales_date 的名稱變更為 transaction_date。

```
alter table spectrum.sales rename column sales_date to transaction_date;
```

下列範例會將資料欄映射設定為使用最佳化資料欄（ORC）格式的外部資料表的位置映射。

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='position');
```

下列範例會將資料欄映射設定為使用 ORC 格式的外部資料表的名稱映射。

```
alter table spectrum.orc_example
set table properties('orc.schema.resolution'='name');
```

ALTER TABLE ADD 和 DROP COLUMN 範例

下列範例示範如何使用 ALTER TABLE 新增和捨棄基本資料表資料欄，以及如何捨棄具有相依物件的資料欄。

ADD 然後 DROP 是基本資料欄

下列範例會將獨立 FEEDBACK_SCORE 資料欄新增至 USERS 資料表。此欄僅包含整數，且此欄的預設值為 NULL（無意見回饋分數）。

首先，查詢 PG_TABLE_DEF 目錄資料表以檢視 USERS 資料表的結構描述：

column	type	encoding	distkey	sortkey
userid	integer	delta	true	1
username	character(8)	lzo	false	0
firstname	character varying(30)	text32k	false	0
lastname	character varying(30)	text32k	false	0
city	character varying(30)	text32k	false	0
state	character(2)	bytedict	false	0
email	character varying(100)	lzo	false	0
phone	character(14)	lzo	false	0
likesports	boolean	none	false	0
liketheatre	boolean	none	false	0
likeconcerts	boolean	none	false	0
likejazz	boolean	none	false	0
likeclassical	boolean	none	false	0
likeopera	boolean	none	false	0
likerock	boolean	none	false	0

likevegas	boolean	none	false	0
likebroadway	boolean	none	false	0
likemusicals	boolean	none	false	0

現在新增 `feedback_score` 資料欄：

```
alter table users
add column feedback_score int
default NULL;
```

從 選取 `FEEDBACK_SCORE` 欄 `USERS`，以確認已新增：

```
select feedback_score from users limit 5;
```

```
feedback_score
-----
NULL
NULL
NULL
NULL
NULL
```

捨棄資料欄以恢復原始 DDL：

```
alter table users drop column feedback_score;
```

捨棄具有相依物件的欄位

下列範例會捨棄具有相依物件的資料欄。結果會將相依物件一併刪除。

若要開始，請再次將 `FEEDBACK_SCORE` 欄新增至 `USERS` 資料表：

```
alter table users
add column feedback_score int
default NULL;
```

接下來，從名為 `USERS_` 的 `USERS` 資料表建立檢視 `VIEW`：

```
create view users_view as select * from users;
```

現在，請嘗試從USERS資料表捨棄 FEEDBACK_SCORE 欄。此DROP陳述式使用預設行為 (RESTRICT) ：

```
alter table users drop column feedback_score;
```

Amazon Redshift 會顯示錯誤訊息，指出無法捨棄資料欄，因為有其他物件與其相依。

請嘗試再次捨棄 FEEDBACK_SCORE 欄，這次CASCADE指定捨棄所有相依物件：

```
alter table users  
drop column feedback_score cascade;
```

ALTER TABLE APPEND

會從現有來源資料表移出資料，將資料列附加到目標資料表。來源資料表中的資料會移至目標資料表中相符的資料欄。資料欄的順序並不重要。資料成功附加至目標資料表後，來源資料表就會是空的。ALTER TABLE APPEND 通常比類似 [CREATE TABLE AS](#) 或 [INSERT INTO](#) 操作快得多，因為資料是移動的，而不是重複的。

Note

ALTER TABLE APPEND 在來源資料表和目標資料表之間移動資料區塊。為了改善效能，ALTER TABLE APPEND 不會將壓縮儲存作為附加操作的一部分。因此，儲存空間的使用量會暫時增加。若要回收空間，請執行 [VACUUM](#) 操作。

同名的資料欄也必須有一模一樣的資料欄屬性。如果來源資料表或目標資料表包含不存在於其他資料表中的資料欄，請使用 IGNOREEXTRA 或 FILLTARGET 參數指定應如何管理額外的資料欄。

您無法附加身分資料欄。若兩個資料表都包含身分資料欄，命令就會失敗。如果只有一個資料表具有身分資料欄，請包含 FILLTARGET 或 IGNOREEXTRA 參數。如需詳細資訊，請參閱 [ALTER TABLE APPEND 用量備註](#)。

您可以附加 GENERATED BY DEFAULT AS IDENTITY 資料欄。您可以使用您提供的值 IDENTITY 來更新定義為 GENERATED BY DEFAULT AS 的資料欄。如需詳細資訊，請參閱 [ALTER TABLE APPEND 用量備註](#)。

目標資料表必須是永久資料表。不過，來源可以是永久資料表，也可以是為串流擷取設定的具體化視觀表。兩個物件必須使用相同的分佈樣式和分佈索引鍵 (如有定義的話)。若物件經過排序，則兩個物件必須使用相同的排序樣式，並且定義相同的資料欄做為排序索引鍵。

ALTER TABLE APPEND 命令會在操作完成後立即自動遞交。此命令無法轉返。您無法在交易區塊 ALTER TABLE APPEND 內執行 (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

所需權限

根據 ALTER TABLE APPEND 命令，需要下列其中一個權限：

- 超級使用者
- 具有 ALTER TABLE 系統權限的使用者
- 來源資料表上具有 DELETE 和 SELECT 權限的使用者，以及目標資料表上具有 INSERT 權限的使用者

語法

```
ALTER TABLE target_table_name APPEND FROM [ source_table_name
| source_materialized_view_name ]
[ IGNOREEXTRA | FILLTARGET ]
```

從具體化視觀表附加只有在為 [將擷取串流至具體化視觀表](#) 設定具體化視觀表的情況下才有作用。

參數

target_table_name

要附加資料列的資料表名稱。僅指定資料表的名稱，或使用格式 `schema_name.table_name` 來使用專屬結構描述。目標資料表必須是現有的永久資料表。

FROM source_table_name

提供要附加之資料列的資料表名稱。僅指定資料表的名稱，或使用格式 `schema_name.table_name` 來使用專屬結構描述。來源資料表必須是現有的永久資料表。

FROM source_materialized_view_name

提供要附加之資料列的具體化視觀表名稱。從具體化視觀表附加只有在為 [將擷取串流至具體化視觀表](#) 設定具體化視觀表的情況下才有作用。來源具體化視觀表必須已存在。

IGNOREEXTRA

此關鍵字會指定，若來源資料表包含了目標資料表中沒有的資料欄，則應捨棄額外資料欄中的資料。您無法 IGNOREEXTRA 搭配使用 FILLTARGET。

FILLTARGET

關鍵字，指定如果目標資料表包含來源資料表中不存在的資料欄，則如果已定義資料欄值，則資料欄應填入資料 [DEFAULT](#) 欄值，或 NULL。您無法 IGNOREEXTRA 搭配使用 FILLTARGET。

ALTER TABLE APPEND 用量備註

ALTER TABLE APPEND 只會將相同的資料欄從來源資料表移至目標資料表。資料欄的順序並不重要。

如果來源資料表或目標資料表包含額外的資料欄，請 IGNOREEXTRA 根據下列規則使用 FILLTARGET 或：

- 如果來源資料表包含不存在於目標資料表中的資料欄，請包含 IGNOREEXTRA。此命令會略過來源資料表中額外的資料欄。
- 如果目標資料表包含來源資料表中不存在的資料欄，請包含 FILLTARGET。如果已定義資料欄值或 IDENTITY 值，則命令會填入目標資料表中的額外資料欄，或 NULL。
- 如果來源資料表和目標資料表都包含額外的資料欄，則命令會失敗。您無法同時使用 FILLTARGET 和 IGNOREEXTRA。

如果兩個資料表中包含的資料欄名稱相同但屬性不同，則命令會失敗。名稱類似的資料欄必須都有下列屬性：

- 資料類型
- 資料欄大小
- 壓縮編碼
- 非 null
- 排序樣式
- 排序索引鍵資料欄
- 分佈樣式
- 分佈索引鍵資料欄

您無法附加身分資料欄。如果來源資料表和目標資料表都有身分資料欄，則命令會失敗。如果只有來源資料表具有身分資料欄，請包含 IGNOREEXTRA 參數，以便忽略身分資料欄。如果只有目標資料表具有身分資料欄，請包含 FILLTARGET 參數，以便根據資料表定義的 IDENTITY 子句填入身分資料欄。如需詳細資訊，請參閱 [DEFAULT](#)。

您可以使用 `ALTER TABLE APPEND` 陳述式附加預設身分資料欄。如需詳細資訊，請參閱 [CREATE TABLE](#)。

ALTER TABLE APPEND 範例

假設您的組織維護資料表 `SALES_MONTHLY`，以擷取目前的銷售交易。您想要每月將資料從交易資料表移至 `SALES` 資料表。

您可以使用下列 `INSERT INTO` 和 `TRUNCATE` 命令來完成任務。

```
insert into sales (select * from sales_monthly);
truncate sales_monthly;
```

不過，您可以使用 `ALTER TABLE APPEND` 命令更有效率地執行相同的操作。

首先，查詢 [PG_TABLE_DEF](#) 系統目錄資料表以確認兩個資料表擁有相同的資料欄，且資料欄的屬性相同。

```
select trim(tablename) as table, "column", trim(type) as type,
encoding, distkey, sortkey, "notnull"
from pg_table_def where tablename like 'sales%';
```

table	column	type	encoding	distkey	sortkey	notnull
sales	salesid	integer	lzo	false	0	true
sales	listid	integer	none	true	1	true
sales	sellerid	integer	none	false	2	true
sales	buyerid	integer	lzo	false	0	true
sales	eventid	integer	mostly16	false	0	true
sales	dateid	smallint	lzo	false	0	true
sales	qtysold	smallint	mostly8	false	0	true
sales	pricepaid	numeric(8,2)	delta32k	false	0	false

```

sales      | commission | numeric(8,2)          | delta32k | false | 0 |
false
sales      | saletime   | timestamp without time zone | lzo      | false | 0 |
false
salesmonth | salesid    | integer               | lzo      | false | 0 |
true
salesmonth | listid     | integer               | none     | true  | 1 |
true
salesmonth | sellerid   | integer               | none     | false | 2 |
true
salesmonth | buyerid   | integer               | lzo      | false | 0 |
true
salesmonth | eventid    | integer               | mostly16 | false | 0 |
true
salesmonth | dateid     | smallint              | lzo      | false | 0 |
true
salesmonth | qtysold    | smallint              | mostly8  | false | 0 |
true
salesmonth | pricepaid  | numeric(8,2)          | delta32k | false | 0 |
false
salesmonth | commission | numeric(8,2)          | delta32k | false | 0 |
false
salesmonth | saletime   | timestamp without time zone | lzo      | false | 0 |
false

```

接著查看各資料表的大小。

```

select count(*) from sales_monthly;
count
-----
  2000
(1 row)

select count(*) from sales;
count
-----
412,214
(1 row)

```

現在請執行下列ALTER TABLE APPEND命令。

```
alter table sales append from sales_monthly;
```

再次查看各資料表的大小。SALES_MONTHLY 資料表現在有 0 個資料列，且SALES資料表已成長 2000 個資料列。

```
select count(*) from sales_monthly;
 count
-----
      0
(1 row)

select count(*) from sales;
 count
-----
414214
(1 row)
```

如果來源資料表具有比目標資料表更多的資料欄，請指定 IGNOREEXTRA 參數。下列範例使用 IGNOREEXTRA 參數，在附加至SALES資料表時忽略 SALES_LISTING 資料表中的額外資料欄。

```
alter table sales append from sales_listing ignoreextra;
```

如果目標資料表的資料欄多於來源資料表，請指定 FILLTARGET 參數。下列範例使用 FILLTARGET 參數來填入 SALES_REPORT 資料表中不存在MONTH的欄SALES。

```
alter table sales_report append from sales_month filltarget;
```

下列範例顯示如何使用ALERTABLEAPPEND具體化視觀表作為來源的範例。

```
ALTER TABLE target_tbl APPEND FROM my_streaming_materialized_view;
```

此範例中的資料表和具體化視觀表名稱為範例。從具體化視觀表附加只有在為 [將擷取串流至具體化視觀表](#) 設定具體化視觀表的情況下才有作用。這會將來源具體化視觀表中的所有記錄移至與具體化視觀表具有相同結構描述的目標資料表，並使具體化視觀表保持不變。這與資料來源為資料表時的行為相同。

ALTER USER

變更資料庫使用者。

所需權限

以下是 ALTER 的必要權限USER：

- 超級使用者
- 具有 ALTERUSER 權限的使用者
- 想要變更自己密碼的目前使用者

語法

```
ALTER USER username [ WITH ] option [, ... ]
```

where *option* is

```
CREATEDB | NOCREATEDB  
| CREATEUSER | NOCREATEUSER  
| SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }  
| PASSWORD { 'password' | 'md5hash' | DISABLE }  
[ VALID UNTIL 'expiration_date' ]  
| RENAME TO new_name |  
| CONNECTION LIMIT { limit | UNLIMITED }  
| SESSION TIMEOUT limit | RESET SESSION TIMEOUT  
| SET parameter { TO | = } { value | DEFAULT }  
| RESET parameter  
| EXTERNALID external_id
```

參數

username

使用者的名稱。

WITH

選用的關鍵字。

CREATEDB | NOCREATEDB

CREATEDB 此選項可讓使用者建立新的資料庫。NOCREATEDB 為預設值。

CREATEUSER | NOCREATEUSER

CREATEUSER 此選項會建立具有所有資料庫權限的超級使用者，包括 CREATE USER。預設值為 NOCREATEUSER。如需詳細資訊，請參閱[superuser](#)。

SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }

此子句會指定使用者對 Amazon Redshift 系統資料表和檢視的存取層級。

具有 SYSLOG ACCESS RESTRICTED 許可的普通使用者只能在使用者可見的系統資料表和檢視中查看該使用者產生的資料列。預設值為 RESTRICTED。

具有 SYSLOG ACCESS UNRESTRICTED 許可的普通使用者可以查看使用者可見系統資料表和檢視中的所有資料列，包括其他使用者產生的資料列。UNRESTRICTED 不會讓一般使用者存取超級使用者可見的資料表。只有超級使用者可看見超級使用者可查看的資料表。

Note

若使用者擁有不受限制的系統資料表存取權限，該使用者就能查看其他使用者產生的資料。例如，STL_QUERY 和 STL_QUERYTEXT 包含 INSERT、UPDATE 和 DELETE 陳述式的全文，其中可能包含敏感的使用者產生資料。

所有使用者都可看見 SVV_TRANSACTIONS 中的所有資料列。

如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

PASSWORD { 'password' | 'md5hash' | DISABLE }

設定使用者的密碼。

根據預設，使用者可以變更自己擁有的密碼，除非密碼遭到停用。若要停用使用者的密碼，請指定 DISABLE。停用使用者密碼時，會從系統中刪除密碼，使用者只能使用臨時 AWS Identity and Access Management (IAM) 使用者憑證登入。如需詳細資訊，請參閱[使用 IAM 身分驗證來產生資料庫使用者憑證](#)。只有超級使用者能夠啟用或停用密碼，您無法停用超級使用者的密碼。若要啟用密碼，請執行 ALTER USER 並指定密碼。

如需使用 PASSWORD 參數的詳細資訊，請參閱[CREATE USER](#)。

VALID UNTIL 'expiration_date'

指定密碼有過期日期。使用 'infinity' 值就不會有過期日期。此參數的有效資料類型為時間戳記。

只有超級使用者可以使用此參數。

RENAME 至

重新命名使用者。

new_name

使用者的新名稱。如需有效名稱的相關資訊，請參閱[名稱與識別碼](#)。

⚠ Important

重新命名使用者時，您也必須重設使用者的密碼。重設密碼不一定要與先前的密碼不同。使用者名稱會做為密碼加密的一部分使用，因此使用者重新命名時，密碼也會遭到清除。在密碼重設前，使用者將無法登入。例如：

```
alter user newuser password 'EXAMPLENewPassword11';
```

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字來允許並行連線的數量上限。另外也可能限制每個資料庫的連線數目。如需詳細資訊，請參閱 [CREATE DATABASE](#)。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 [STV_SESSIONS](#) 系統畫面。

ℹ Note

如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

SESSION TIMEOUT 限制 | RESET SESSION TIMEOUT

工作階段保持非作用中或閒置的時間上限 (以秒為單位)。範圍是 60 秒 (一分鐘) 到 1,728,000 秒 (20 天)。如果未為使用者設定工作階段逾時，則會套用叢集設定。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 中的配額和限制](#)。

當您設定工作階段逾時時，只會套用至新的工作階段。

若要檢視有關作用中使用者工作階段的資訊 (包括開始時間、使用者名稱和工作階段逾時)，請查詢 [STV_SESSIONS](#) 系統檢視。若要檢視有關使用者工作階段歷史記錄的資訊，請查詢 [STL_SESSIONS](#) 檢視。若要擷取有關資料庫使用者的資訊 (包括工作階段逾時值)，請查詢 [SVL_USER_INFO](#) 檢視。

SET

將指定使用者執行之所有工作階段的組態參數設定為新的預設值。

RESET

將指定使用者的組態參數重設為原始預設值。

parameter

要設定或重設的參數名稱。

值

參數的新值。

DEFAULT

將指定使用者執行之所有工作階段的組態參數設定為預設值。

EXTERNALID external_id

與身分提供者相關聯的使用者識別碼。使用者必須停用密碼。如需詳細資訊，請參閱 [Amazon Redshift 的原生身分提供者 \(IdP\) 聯合](#)。

使用須知

- 嘗試修改 rdsdb - 您不能修改名為 rdsdb 的使用者。
- 建立未知密碼 - 使用 AWS Identity and Access Management (IAM) 身分驗證來建立資料庫使用者憑證時，您可能想要建立只能使用臨時憑證登入的超級使用者。您無法停用超級使用者的密碼，但您可以使用隨機產生的MD5雜湊字串建立未知的密碼。

```
alter user iam_superuser password 'md51234567890123456780123456789012';
```

- 設定 search_path - 當您使用 ALTERUSER命令設定 [search_path](#) 參數時，修改會在指定的使用者下次登入時生效。如果您想要變更目前使用者和工作階段的 search_path 值，請使用 SET命令。
- 設定時區 - 當您SETTIMEZONE搭配 ALTERUSER命令使用時，修改會在指定的使用者下次登入時生效。
- 使用動態資料遮罩和資料列層級安全性政策 - 當您佈建的叢集或無伺服器命名空間具有任何動態資料遮罩或資料列層級安全性政策時，一般使用者會無法使用下列命令：

```
ALTER <current_user> SET enable_case_sensitive_super_attribute/  
enable_case_sensitive_identifer/downcase_delimited_identifer
```

只有具有 ALTERUSER權限的超級使用者和使用者才能設定這些組態選項。如需有關資料列層級安全性詳細資訊，請參閱 [資料列層級安全性](#)。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。

範例

下列範例提供使用者建立資料庫ADMIN的權限：

```
alter user admin createdb;
```

下列範例會將使用者的密碼設定為 `ADMINadminPass9` 並設定密碼的過期日期和時間：

```
alter user admin password 'adminPass9'  
valid until '2017-12-31 23:59';
```

下列範例會將使用者重新命名ADMIN為 SYSADMIN：

```
alter user admin rename to sysadmin;
```

下列範例會將使用者的閒置工作階段逾時更新為 300 秒。

```
ALTER USER dbuser SESSION TIMEOUT 300;
```

重設使用者的閒置工作階段逾時。重設時會套用叢集設定。您必須是資料庫超級使用者才能執行此命令。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 中的配額和限制](#)。

```
ALTER USER dbuser RESET SESSION TIMEOUT;
```

下列範例會更新名為 bob 之使用者的外部 ID。命名空間為 myco_aad。如果命名空間未與已註冊的身分提供者相關聯，則會導致錯誤。

```
ALTER USER myco_aad:bob EXTERNALID "ABC123" PASSWORD DISABLE;
```

下列範例會為特定資料庫使用者執行的所有工作階段設定時區。這會變更後續工作階段的時區，但不會變更目前工作階段的時區。

```
ALTER USER odie SET TIMEZONE TO 'Europe/Zurich';
```

下列範例會設定bob允許使用者開啟的資料庫連線數目上限。

```
ALTER USER bob CONNECTION LIMIT 10;
```

ANALYZE

更新資料表統計資訊以供查詢規劃器使用。

所需權限

以下是的必要權限ANALYZE：

- 超級使用者
- 具有 ANALYZE 權限的使用者
- 關係的擁有者
- 共用資料表的資料庫擁有者

語法

```
ANALYZE [ VERBOSE ]  
[ [ table_name [ ( column_name [, ...] ) ] ]  
[ PREDICATE COLUMNS | ALL COLUMNS ]
```

參數

VERBOSE

傳回有關ANALYZE操作之進度資訊訊息的子句。若您不指定資料表，此選項會很實用。

table_name

您可以分析特定資料表，包括臨時資料表。您可以利用結構描述名稱限定資料表。您可以選擇性的指定 table_name 來分析單一資料表。您無法使用單一 table_name 陳述式指定多個 ANALYZE table_name。如果您未指定 table_name 值，則目前連接的資料庫中所有資料表都會加以分析，包括系統目錄中的永久性資料表。如果自上次變更的資料列百分比ANALYZE低於分析閾值，Amazon Redshift 會略過分析資料表。如需詳細資訊，請參閱[分析閾值](#)。

您不需要分析 Amazon Redshift 系統資料表（STL 和 STV 資料表）。

column_name

若您指定 table_name，則您也可以在資料表中指定一個或多個資料欄（括號內以資料欄分隔的清單）。若指定了資料欄清單，則只會分析列出的資料欄。

PREDICATE COLUMNS | ALL COLUMNS

表示是否ANALYZE應僅包含述詞資料欄的子句。指定 僅PREDICATECOLUMNS分析在先前查詢中用作述詞或可能用作述詞的候選資料欄。指定 ALL COLUMNS 以分析所有資料欄。預設值為 ALL COLUMNS。

若下列任一條件為真，表示資料欄包含在一組述詞資料欄中：

- 資料欄已在查詢中做為篩選的一部分、聯結條件或 group by 子句使用。
- 資料欄是分佈索引鍵。
- 資料欄是排序索引鍵的一部分。

如果沒有資料欄標示為述詞資料欄，例如因為尚未查詢資料表，即使指定 PREDICATE COLUMNS，也會分析所有資料欄。發生這種情況時，Amazon Redshift 可能會回應訊息，例如找不到 " 的述詞欄 *table-name*"。分析所有資料欄。如需述詞資料欄的相關資訊，請參閱 [分析資料表](#)。

使用須知

Amazon Redshift 會自動在您使用下列命令建立的資料表ANALYZE上執行：

- CREATE TABLE AS
- CREATE TEMP TABLE AS
- SELECT INTO

您無法分析外部資料表。

第一次建立這些資料表時，您不需要在這些資料表上執行 ANALYZE 命令。若您修改這些資料表，則應依照與其他資料表相同的方式進行分析。

分析閾值

為了縮短處理時間並改善整體系統效能，如果自上次ANALYZE命令執行以來變更的資料列百分比低於 [analyze_threshold_percent](#) 參數指定的分析閾值，Amazon Redshift 會略過ANALYZE資料表。analyze_threshold_percent 預設為 10。若要變更目前工作階段的 analyze_threshold_percent，請執行 [SET](#) 命令。下列範例會將 analyze_threshold_percent 變更為 20%。

```
set analyze_threshold_percent to 20;
```

若要在只有少量資料列變更時分析資料表，請將 `analyze_threshold_percent` 設定為任何更小的數。例如，如果將 `analyze_threshold_percent` 設為 0.01，則 100,000,000 資料列的資料表中變更的資料列大於 10,000 列資料時，便不會略過資料表。

```
set analyze_threshold_percent to 0.01;
```

如果因為資料表不符合分析閾值而ANALYZE略過資料表，Amazon Redshift 會傳回下列訊息。

```
ANALYZE SKIP
```

若要分析所有資料表，即使沒有任何資料列變更，請將 `analyze_threshold_percent` 設為 0。

若要檢視ANALYZE操作結果，請查詢[STL_ANALYZE](#)系統資料表。

如需分析資料表的相關資訊，請參閱 [分析資料表](#)。

範例

分析TICKIT資料庫中的所有資料表並傳回進度資訊。

```
analyze verbose;
```

僅分析LISTING資料表。

```
analyze listing;
```

分析VENUE資料表中的 VENUEID和 VENUENAME資料欄。

```
analyze venue(venueid, venueid);
```

僅分析VENUE資料表中的述詞資料欄。

```
analyze venue predicate columns;
```

ANALYZE COMPRESSION

執行壓縮分析並產生報告，當中針對分析的資料表提供建議的壓縮編碼。對於每欄，報告會包含與 RAW編碼相比，磁碟空間可能減少的估計值。

語法

```
ANALYZE COMPRESSION  
[ [ table_name ]  
[ ( column_name [, ...] ) ] ]  
[COMPROWS numrows]
```

參數

table_name

您可以分析特定資料表的壓縮，包括臨時資料表。您可以利用結構描述名稱限定資料表。您可以選擇性的指定 *table_name* 來分析單一資料表。如果您未指定 *table_name* 值，目前連接的資料庫中所有資料表都會加以分析。您無法使用單一 ANALYZE COMPRESSION 陳述式指定多個 *table_name*。

column_name

若您指定 *table_name*，則您也可以指定一個或多個資料欄 (括號內以資料欄分隔的清單)。

COMPROWS

資料列數，做為壓縮分析的樣本大小。分析是以每個資料配量中的列為對象。例如，如果您指定 COMPROWS 1000000 (1,000,000) 且系統包含總共 4 個層，則每個層不會讀取和分析超過 250,000 列。如果 COMPROWS 未指定，則每個切片的樣本數預設為 100,000。COMPROWS 小於每個切片 100,000 列預設值的值會自動升級至預設值。但是，如果資料表中的資料量不足，無法產生有意義的樣本，則壓縮分析不會產生建議。如果 COMPROWS 數字大於資料表中的資料列數目，ANALYZE COMPRESSION 命令仍會繼續並針對所有可用的資料列執行壓縮分析。如果未指定資料表，則使用 COMPROWS 會導致錯誤。

numrows

資料列數，做為壓縮分析的樣本大小。*numrows* 可接受的範圍是介於 1000 到 1000000000 (1,000,000,000) 之間的數字。

使用須知

ANALYZE COMPRESSION 會取得專屬資料表鎖定，以防止同時讀取和寫入資料表。資料表閒置時，才執行 ANALYZE COMPRESSION 命令。

根據資料表內容的範例，執行 ANALYZE COMPRESSION 以取得資料欄編碼結構的建議。ANALYZE COMPRESSION 是建議工具，不會修改資料表的資料欄編碼。您可以透過重新建立資料表，或建立具

有相同結構描述的新資料表，套用建議的編碼。利用適當的編碼機制重新建立未經壓縮的資料表，可大幅減少在磁碟上佔用的空間。此方法節省了磁碟空間，並提高 I/O 繫結工作負載的查詢效能。

ANALYZE COMPRESSION 會略過實際分析階段，並在指定為的任何資料欄上直接傳回原始編碼類型 SORTKEY。這樣做是因為當 SORTKEY 資料欄壓縮得比其他資料欄高得多時，範圍限制掃描可能會效能不佳。

範例

下列範例僅顯示 LISTING 資料表中資料欄的編碼和預估降低百分比：

```
analyze compression listing;
```

Table	Column	Encoding	Est_reduction_pct
listing	listid	az64	40.96
listing	sellerid	az64	46.92
listing	eventid	az64	53.37
listing	dateid	raw	0.00
listing	numtickets	az64	65.66
listing	priceperticket	az64	72.94
listing	totalprice	az64	68.05
listing	listtime	az64	49.74

下列範例會分析 SALES 資料表中的 COMMISSION、QTYSOLD 和 SALETIME 資料欄。

```
analyze compression sales(qtysold, commission, saletime);
```

Table	Column	Encoding	Est_reduction_pct
sales	salesid	N/A	0.00
sales	listid	N/A	0.00
sales	sellerid	N/A	0.00
sales	buyerid	N/A	0.00
sales	eventid	N/A	0.00
sales	dateid	N/A	0.00
sales	qtysold	az64	83.06
sales	pricepaid	N/A	0.00
sales	commission	az64	71.85
sales	saletime	az64	49.63

ATTACH MASKING POLICY

將現有的動態資料遮罩政策附加至資料欄。如需動態資料遮罩的相關資訊，請參閱 [動態資料遮罩](#)。

超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以附加遮罩政策。

語法

```
ATTACH MASKING POLICY policy_name
  ON { relation_name }
  ( {output_columns_names | output_path} ) [ USING ( {input_column_names | input_path
}) ]
  TO { user_name | ROLE role_name | PUBLIC }
  [ PRIORITY priority ];
```

參數

`policy_name`

欲附加的遮罩政策名稱。

`relation_name`

欲附加遮罩政策的關係名稱。

`output_column_names`

要套用遮罩政策的資料欄名稱。

`output_paths`

遮罩政策將套用之SUPER物件的完整路徑，包括資料欄名稱。例如，對於與名為 `person` 的SUPER類型資料欄的關係 `person`，`output_path` 可能是 `person.name.first_name`。

`input_column_names`

將作為遮罩政策輸入的資料欄名稱。此為選用參數。如果未指定，遮罩政策會使用 `output_column_names` 做為輸入。

`input_paths`

遮罩政策將作為輸入的SUPER物件的完整路徑。此為選用參數。如果未指定，遮罩政策會使用 `output_path` 做為輸入。

user_name

要附加遮罩政策的使用者名稱。您無法將兩個政策附加到使用者和欄或角色和欄的相同組合。您可以將一個政策附加至使用者，並將另一個政策附加至使用者的角色。在此情況下，會套用優先順序較高的政策。

您只能在單一ATTACHMASKINGPOLICY命令PUBLIC中設定 user_name、 role_name 和 之一。

role_name

要附加遮罩政策的角色名稱。您無法將兩個政策附加至相同的資料欄/角色配對。您可以將一個政策附加至使用者，並將另一個政策附加至使用者的角色。在此情況下，會套用優先順序較高的政策。

您只能在單一ATTACHMASKINGPOLICY命令PUBLIC中設定 user_name、 role_name 和 之一。

PUBLIC

將遮罩政策附加至存取資料表的所有使用者。您必須為連接至特定資料欄/使用者或資料欄/角色對的其他遮罩政策提供比套用PUBLIC政策更高的優先順序。

您只能在單一ATTACHMASKINGPOLICY命令PUBLIC中設定 user_name、 role_name 和 之一。

priority

遮罩政策的優先順序。當多個遮罩政策套用至指定使用者的查詢時，會套用優先順序最高的政策。

您無法將兩個不同的政策附加到具有相同優先順序的同個資料欄，即使這兩個策略附加到不同的使用者或角色也是如此。只要附加政策的使用者或角色每次都不同，您就可以將相同的政策多次附加至同一組資料表、輸出資料欄、輸入資料欄和優先順序參數。

您無法對資料欄套用與附加至該資料欄的另一個政策具有相同優先順序的政策，即使用於不同角色也一樣。此欄位為選用欄位。如果未指定優先順序，遮罩政策會預設套用值為 0 的優先順序。

ATTACH RLS POLICY

將資料表上的資料列層級安全性政策附加至一或多個使用者或角色。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以附加政策。

語法

```
ATTACH RLS POLICY policy_name ON [TABLE] table_name [, ...]
```

```
TO { user_name | ROLE role_name | PUBLIC } [, ...]
```

參數

`policy_name`

政策的名稱。

ON **【TABLE】** `table_name` **【, ...】**

連接至關係之資料列層級安全政策的名稱。

至 { `user_name` | ROLE `role_name` | PUBLIC } **【, ...】**

指定政策是否附加至一或多個指定的使用者或角色。

使用須知

使用 ATTACH RLS POLICY 陳述式時，請遵守下列事項：

- 附加的資料表應具有政策建立陳述式子WITH句中列出的所有資料欄。
- Amazon Redshift RLS 不支援將RLS政策連接至下列物件：
 - 目錄表
 - 跨資料庫關係
 - 外部資料表
 - 暫時資料表
 - 查詢表
- 您無法將RLS政策連接至超級使用者或具有 `sys:secadmin` 許可的使用者。

範例

下列範例會將資料表上的政策附加至角色。

```
ATTACH RLS POLICY policy_concerts ON tickit_category_redshift TO ROLE analyst, ROLE dbadmin;
```

BEGIN

開始交易。與同義字STARTTRANSACTION。

交易是單一邏輯工作單位，無論是由一個命令或多個命令所組成。一般而言，交易中的所有命令會在資料庫的快照上執行，其開始時間是由 `transaction_snapshot_begin` 系統組態參數的值組所決定。

根據預設，個別 Amazon Redshift 操作（查詢、DDL 陳述式、負載）會自動提交至資料庫。如果您想要暫停對操作的遞交，直到後續工作完成為止，您需要使用 `BEGIN` 陳述式開啟交易，然後執行所需的命令，然後使用 [COMMIT](#) 或 [END](#) 陳述式關閉交易。若有需要，您可以使用 [ROLLBACK](#) 陳述式停止進行中的交易。此行為的例外狀況是 [TRUNCATE](#) 命令，它會遞交本身執行所在的交易，而且無法轉返。

語法

```
BEGIN [ WORK | TRANSACTION ] [ ISOLATION LEVEL option ] [ READ WRITE | READ ONLY ]
```

```
START TRANSACTION [ ISOLATION LEVEL option ] [ READ WRITE | READ ONLY ]
```

Where *option* is

```
SERIALIZABLE  
| READ UNCOMMITTED  
| READ COMMITTED  
| REPEATABLE READ
```

Note: `READ UNCOMMITTED`, `READ COMMITTED`, and `REPEATABLE READ` have no operational impact and map to `SERIALIZABLE` in Amazon Redshift. You can see database isolation levels on your cluster by querying the `stv_db_isolation_level` table.

參數

WORK

選用的關鍵字。

TRANSACTION

選用關鍵字；`WORK` 以及 `TRANSACTION` 是同義字。

ISOLATION LEVEL SERIALIZABLE

可序列化隔離是預設支援的操作，因此無論陳述式中是否包含此語法，交易的行為都一樣。如需詳細資訊，請參閱[管理並行寫入操作](#)。不支援任何其他隔離層級。

Note

SQL 該標準定義了四個層級的交易隔離，以防止讀取髒污（交易讀取並行未承諾交易所寫入的資料），不可重複的讀取（其中交易重新讀取先前讀取的資料，並發現資料在初次讀取後被另一個遞交的交易所變更），和假體讀取（其中交易重新執行查詢，傳回一組符合搜尋條件的列，然後，會發現一組資料列因另一個最近遞交的交易而變更）：

- 讀取未遞交資料：已變更讀取、不可重複讀取及幽靈讀取都有可能發生。
- 讀取已遞交資料：不可重複讀取及幽靈讀取都有可能發生。
- 可重複讀取：可能發生幽靈讀取。
- 可序列化：防止已變更讀取、不可重複讀取及幽靈讀取。

雖然您可以使用四種交易隔離層級中的任一種，但是 Amazon Redshift 會以可序列化的方式處理所有隔離層級。

READ WRITE

為交易提供讀取和寫入許可。

READ ONLY

為交易提供唯讀許可。

範例

下列範例會啟動可序列化交易區塊：

```
begin;
```

下列範例會啟動具有可序列化隔離層級和讀取與寫入許可的交易區塊：

```
begin read write;
```

CALL

執行預存程序。CALL 命令必須包含程序名稱和輸入引數值。您必須使用 CALL 陳述式呼叫預存程序。

Note

CALL 不能成為任何常規查詢的一部分。

語法

```
CALL sp_name ( [ argument ] [, ...] )
```

參數

sp_name

要執行的程序的名稱。

argument

輸入引數的值。此參數也可以是函數名稱，例如 `pg_last_query_id()`。您無法將查詢用作 CALL 引數。

使用須知

Amazon Redshift 預存程序支援巢狀和遞迴呼叫，如下所述。此外，請確定您的驅動程式支援為 up-to-date，也如下所述。

主題

- [巢狀呼叫](#)
- [驅動程式支援](#)

巢狀呼叫

Amazon Redshift 預存程序支援巢狀和遞迴呼叫。允許的巢狀層級數目上限為 16。巢狀呼叫可以將商業邏輯封裝成較小的程序，供多個發起人共用。

如果您呼叫具有輸出參數的巢狀程序，內部程序必須定義 INOUT 引數。在此情況下，非常數變數會傳入內部程序中。OUT 不允許引數。發生此行為是因為需要變數來保留內部呼叫的輸出。

內層和外層程序的關係記錄在 [SVL_STORED_PROC_CALL](#) 的 `from_sp_call` 欄。

下列範例顯示透過INOUT引數傳遞變數至巢狀程序呼叫。

```
CREATE OR REPLACE PROCEDURE inner_proc(INOUT a int, b int, INOUT c int) LANGUAGE
  plpgsql
AS $$
BEGIN
  a := b * a;
  c := b * c;
END;
$$;

CREATE OR REPLACE PROCEDURE outer_proc(multiplier int) LANGUAGE plpgsql
AS $$
DECLARE
  x int := 3;
  y int := 4;
BEGIN
  DROP TABLE IF EXISTS test_tbl;
  CREATE TEMP TABLE test_tbl(a int, b varchar(256));
  CALL inner_proc(x, multiplier, y);
  insert into test_tbl values (x, y::varchar);
END;
$$;

CALL outer_proc(5);

SELECT * from test_tbl;
 a | b
-----+-----
 15 | 20
(1 row)
```

驅動程式支援

建議您將 Java Database Connectivity (JDBC) 和 Open Database Connectivity (ODBC) 驅動程式升級至支援 Amazon Redshift 預存程序的最新版本。

如果您的用戶端工具使用透過CALL陳述式傳遞至伺服器的驅動程式API操作，則您可能可以使用現有的驅動程式。傳回的輸出參數 (若有) 是一列的結果集。

最新版本的 Amazon Redshift JDBC和ODBC驅動程式具有儲存程序探索的中繼資料支援。對於自訂 Java 應用程式。還支援 CallableStatement。如需驅動程式的詳細資訊，請參閱 [Amazon Redshift 管理指南中的使用SQL用戶端工具連線至 Amazon Redshift 叢集](#)。

下列範例示範如何使用JDBC驅動程式的不同API操作進行預存程序呼叫。

```
void statement_example(Connection conn) throws SQLException {
    statement.execute("CALL sp_statement_example(1)");
}

void prepared_statement_example(Connection conn) throws SQLException {
    String sql = "CALL sp_prepared_statement_example(42, 84)";
    PreparedStatement pstmt = conn.prepareStatement(sql);
    pstmt.execute();
}

void callable_statement_example(Connection conn) throws SQLException {
    CallableStatement cstmt = conn.prepareCall("CALL sp_create_out_in(?,?)");
    cstmt.registerOutParameter(1, java.sql.Types.INTEGER);
    cstmt.setInt(2, 42);
    cstmt.executeQuery();
    Integer out_value = cstmt.getInt(1);
}
```

範例

下列範例呼叫程序名稱 test_sp1。

```
call test_sp1(3,'book');
INFO: Table "tmp_tbl" does not exist and will be skipped
INFO: min_val = 3, f2 = book
```

下列範例呼叫程序名稱 test_sp12。

```
call test_sp2(2,'2019');

      f2          | column2
-----+-----
2019+2019+2019+2019 | 2
(1 row)
```

CANCEL

取消目前正在執行的資料庫查詢。

此CANCEL命令需要執行中查詢的程序 ID 或工作階段 ID，並顯示確認訊息，以確認查詢已取消。

所需權限

以下是 的必要權限CANCEL：

- 超級使用者取消自己的查詢
- 超級使用者取消使用者的查詢
- 具有取消使用者查詢CANCEL權限的使用者
- 使用者取消自己的查詢

語法

```
CANCEL process_id [ 'message' ]
```

參數

process_id

若要取消在 Amazon Redshift 叢集中執行的查詢，請使用對應至您要取消之查詢[STV_RECENTS](#)的 *pid* (程序 ID)。

若要取消在 Amazon Redshift Serverless 工作群組中執行的查詢，請使用對應至您要取消之查詢[session_idSYS_QUERY_HISTORY](#)的。

'*message*'

選用的確認訊息，會在查詢取消完成時顯示。如果您未指定訊息，Amazon Redshift 會顯示預設訊息進行驗證。您必須將訊息用單引號括住。

使用須知

您無法透過指定查詢 ID 來取消查詢；您必須指定查詢的程序 ID (PID) 或工作階段 ID 。您只能取消目前由您的使用者執行的查詢。超級使用者可以取消所有查詢。

如果多個工作階段中的查詢在同一個資料表上保留鎖定，您可以使用 [PG_TERMINATE_BACKEND](#) 函數終止其中一個工作階段。這麼做可強制終止的工作階段中任何目前執行中的交易釋出所有鎖定和復原交易。若要檢視目前的鎖定情形，可查詢 [STV_LOCKS](#) 系統資料表。

在某些內部事件之後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 PID。如果 PID 已變更，您可能會收到下列錯誤訊息。


```
Session <PID> does not exist. The session PID might have changed. Check the
stl_restarted_sessions system table for details.
```

若要尋找新的 PID，請查詢 oldpid 欄上的 [STL_RESTARTED_SESSIONS](#) 系統資料表和篩選條件。

```
select oldpid, newpid from stl_restarted_sessions where oldpid = 1234;
```

範例

若要取消 Amazon Redshift 叢集中目前正在執行的查詢，請先擷取您要取消之查詢的程序 ID。若要判斷 IDs 所有目前執行中查詢的程序，請輸入下列命令：

```
select pid, starttime, duration,
trim(user_name) as user,
trim (query) as querytxt
from stv_recents
where status = 'Running';
```

pid	starttime	duration	user	querytxt
802	2008-10-14 09:19:03.550885	132	dwuser	select venue venue from venue where venuestate='FL', where venuecity not in ('Miami' , 'Orlando');
834	2008-10-14 08:33:49.473585	1250414	dwuser	select * from listing;
964	2008-10-14 08:30:43.290527	326179	dwuser	select sellerid from sales where qtysold in (8, 10);

檢查查詢文字，以判斷哪個程序 ID (PID) 對應至您要取消的查詢。

輸入下列命令以使用 PID 802 取消該查詢：

```
cancel 802;
```

查詢執行所在的工作階段會顯示以下訊息：

```
ERROR: Query (168) cancelled on user's request
```

其中 168 是查詢 ID (不是用來取消查詢的處理程序 ID)。

或者，您可以指定顯示自訂確認訊息來取代預設訊息。若要指定自訂訊息，請在CANCEL命令結尾將訊息包含在單一引號中：

```
cancel 802 'Long-running query';
```

查詢執行所在的工作階段會顯示以下訊息：

```
ERROR: Long-running query
```

CLOSE

(選用) 關閉與開啟游標相關聯的所有可用資源。[END](#)、和 [COMMITROLLBACK](#)會自動關閉游標，因此不需要使用 CLOSE命令來明確關閉游標。

如需詳細資訊，請參閱 [DECLARE](#)、[FETCH](#)。

語法

```
CLOSE cursor
```

參數

cursor

要關閉的游標名稱。

CLOSE 範例

以下命令會關閉游標並執行遞交，進而結束交易：

```
close movie_cursor;  
commit;
```

COMMENT

建立或變更有關資料庫物件的註解。

語法

```
COMMENT ON
```

```
{  
TABLE object_name |  
COLUMN object_name.column_name |  
CONSTRAINT constraint_name ON table_name |  
DATABASE object_name |  
VIEW object_name  
}  
IS 'text' | NULL
```

參數

object_name

要加上註解之資料庫物件的名稱。您可以新增註解至下列物件：

- TABLE
- COLUMN (也需要 *column_name*)。
- CONSTRAINT (也需要 *traint_name* 和 *table_name*)。
- DATABASE
- VIEW
- SCHEMA

IS '文字' | NULL

您要對指定物件新增或取代的註解文字。文字字串是資料類型 TEXT。以單引號括住註解。將值設定為 NULL 以移除評論文字。

column_name

要加上註解之資料欄的名稱。的參數 COLUMN。接在 *object_name* 中指定的資料表後面。

constraint_name

要加上註解之限制條件的名稱。的參數 CONSTRAINT。

table_name

包含限制條件的資料表名稱。的參數 CONSTRAINT。

使用須知

您必須是超級使用者或資料庫物件的擁有者，才能新增或更新註解。

資料庫的註解僅適用於目前資料庫。如果您嘗試對不同資料庫加上註解，則會顯示警告訊息。對不存在的資料庫加上註解時，也會顯示同樣的警告。

不支援對外部資料表、外部資料欄和延遲繫結檢視資料欄的註解。

範例

下列範例會將註解新增至SALES資料表。

```
COMMENT ON TABLE sales IS 'This table stores tickets sales data';
```

下列範例會在SALES資料表上顯示註解。

```
select obj_description('public.sales'::regclass);

obj_description
-----
This table stores tickets sales data
```

下列範例會從SALES資料表中移除註解。

```
COMMENT ON TABLE sales IS NULL;
```

下列範例會將註解新增至SALES資料表的資料EVENTID欄。

```
COMMENT ON COLUMN sales.eventid IS 'Foreign-key reference to the EVENT table.';
```

下列範例會在SALES資料表的資料EVENTID欄（資料欄編號 5）上顯示註解。

```
select col_description( 'public.sales'::regclass, 5::integer );

col_description
-----
Foreign-key reference to the EVENT table.
```

下列範例會將描述性註解新增至EVENT資料表。

```
comment on table event is 'Contains listings of individual events.';
```

若要檢視註解，請查詢 PG_DESCRIPTION 系統目錄。下列範例會傳回EVENT資料表的描述。

```
select * from pg_catalog.pg_description
where objoid =
(select oid from pg_class where relname = 'event'
and relnamespace =
(select oid from pg_catalog.pg_namespace where nsname = 'public') );

objoid | classoid | objsubid | description
-----+-----+-----+-----
116658 |      1259 |          0 | Contains listings of individual events.
```

COMMIT

將目前交易遞交至資料庫。此命令會使交易的資料庫更新變成永久有效。

語法

```
COMMIT [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。預存程序內不支援使用此關鍵字。

TRANSACTION

選用的關鍵字。WORK 和 TRANSACTION是同義詞。預存程序內不支援使用這兩個關鍵字。

如需COMMIT在預存程序中使用的資訊，請參閱 [管理交易](#)。

範例

以下每個範例都會將目前交易遞交至資料庫：

```
commit;
```

```
commit work;
```

```
commit transaction;
```

COPY

從資料檔案或 Amazon DynamoDB 資料表，將資料載入資料表。檔案可以位於 Amazon Simple Storage Service (Amazon S3) 儲存貯體、Amazon EMR叢集，或使用 Secure Shell (SSH) 連線存取的遠端主機。

Note

Amazon Redshift Spectrum 外部資料表處於唯讀狀態。您無法COPY前往外部資料表。

COPY 命令會將輸入資料附加為資料表的額外資料列。

任何來源的單一輸入資料列最大為 4 MB。

主題

- [所需的許可](#)
- [COPY 語法](#)
- [必要參數](#)
- [選用的參數](#)
- [COPY 命令的使用備註和其他資源](#)
- [COPY 命令範例](#)
- [COPYJOB\(預覽\)](#)
- [COPY 參數參考](#)
- [使用須知](#)
- [COPY 範例](#)

所需的許可

若要使用 COPY命令，您必須具有 Amazon Redshift 資料表[INSERT](#)的權限。

COPY 語法

```
COPY table-name
```

```
[ column-list ]  
FROM data_source  
authorization  
[ [ FORMAT ] [ AS ] data_format ]  
[ parameter [ argument ] [, ... ] ]
```

您可以使用最少三個參數來執行COPY操作：資料表名稱、資料來源，以及存取資料的授權。

Amazon Redshift 擴展COPY了 命令的功能，可讓您從多個資料來源以多種資料格式載入資料、控制載入資料的存取、管理資料轉換和管理載入操作。

下列各節會呈現所需的COPY命令參數，依函數分組選用參數。其中也會描述每個參數，並解釋如何搭配運用各種選項。您可以利用依字母順序排列的參數清單，直接跳到參數描述。

必要參數

此COPY命令需要三個元素：

- [Table Name](#)
- [Data Source](#)
- [Authorization](#)

最簡單的COPY命令會使用下列格式。

```
COPY table-name  
FROM data-source  
authorization;
```

下列範例會建立名為 的資料表CATDEMO，然後從名為 的 Amazon S3 中的資料檔案中載入包含範例資料的資料表category_pipe.txt。

```
create table catdemo(catid smallint, catgroup varchar(10), catname varchar(10), catdesc  
varchar(50));
```

在下列範例中，COPY命令的資料來源是名為 的 category_pipe.txt Amazon S3 儲存貯體tickit資料夾中名為 的資料檔案redshift-downloads。COPY 命令已獲授權，可透過 AWS Identity and Access Management (IAM) 角色存取 Amazon S3 儲存貯體。如果您的叢集具有已連接 Amazon S3 的現有IAM角色，您可以在下列COPY命令中取代角色的 Amazon Resource Name (ARN) 並執行該角色。

```
copy catdemo
from 's3://redshift-downloads/ticket/category_pipe.txt'
iam_role 'arn:aws:iam::<aws-account-id>:role/<role-name>'
region 'us-east-1';
```

如需如何使用COPY命令載入範例資料的完整指示，包括從其他 AWS 區域載入資料的指示，請參閱 Amazon Redshift 入門指南中的[從 Amazon S3 載入範例資料](#)。

table-name

COPY 命令的目標資料表名稱。此資料表必須已存在於資料庫中。此資料表可以是暫時性或持久性。COPY 命令會將新的輸入資料附加至資料表中的任何現有資料列。

FROM 資料來源

在目標資料表中載入之來源資料的位置。資訊清單檔案可與一些資料來源一起指定。

最常用的資料儲存庫是 Amazon S3 儲存貯體。您也可以從位於 Amazon EMR 叢集中的資料檔案、Amazon EC2 執行個體，或叢集可以使用 SSH 連線存取的遠端主機載入，也可以直接從 DynamoDB 資料表載入。

- [COPY 從 Amazon S3](#)
- [COPY 從 Amazon EMR](#)
- [COPY 從遠端主機 \(SSH \)](#)
- [COPY 從 Amazon DynamoDB](#)

授權

指出叢集用於身分驗證和授權以存取其他 AWS 資源的方法的子句。COPY 命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2。您可以參考連接至叢集 IAM 的角色，或為 IAM 使用者提供存取金鑰 ID 和秘密存取金鑰，以提供該授權。

- [授權參數](#)
- [角色類型存取控制](#)
- [金鑰型存取控制](#)

選用的參數

您可以選擇性地指定將欄位資料 COPY 映射至目標資料表中資料欄的方式、定義來源資料屬性，以讓 COPY 命令正確讀取和剖析來源資料，以及管理 COPY 命令在載入程序期間執行的操作。

- [欄映射選項](#)
- [資料格式參數](#)
- [資料轉換參數](#)
- [資料載入操作](#)

欄映射

根據預設，會以與資料檔案中發生之欄位相同的順序，將欄位值COPY插入目標資料表的欄。如果預設資料欄順序無法運作，您可以指定資料欄清單，或使用JSONPath運算式將來源資料欄位映射到目標資料欄。

- [Column List](#)
- [JSONPaths File](#)

資料格式參數

您可以從以固定寬度、字元分隔、逗號分隔值（CSV）或JSON格式的文字檔案，或從 Avro 檔案載入資料。

根據預設，COPY命令預期來源資料在字元分隔的 UTF-8 文字檔案中。預設分隔符號是縱線字元（|）。如果來源資料是其他格式，請使用下列參數來指定資料格式。

- [FORMAT](#)
- [CSV](#)
- [DELIMITER](#)
- [FIXEDWIDTH](#)
- [SHAPEFILE](#)
- [AVRO](#)
- [JSON](#)
- [ENCRYPTED](#)
- [BZIP2](#)
- [GZIP](#)
- [LZOP](#)
- [PARQUET](#)
- [ORC](#)

- [ZSTD](#)

資料轉換參數

載入資料表時，COPY會嘗試隱含地將來源資料中的字串轉換為目標資料欄的資料類型。如果您需要指定不同於預設行為的轉換，或預設轉換造成錯誤，您可以指定下列參數來管理資料轉換。

- [ACCEPTANYDATE](#)
- [ACCEPTINVCHARS](#)
- [BLANKSASNULL](#)
- [DATEFORMAT](#)
- [EMPTYASNULL](#)
- [ENCODING](#)
- [ESCAPE](#)
- [EXPLICIT_IDS](#)
- [FILLRECORD](#)
- [IGNOREBLANKLINES](#)
- [IGNOREHEADER](#)
- [NULL AS](#)
- [REMOVEQUOTES](#)
- [ROUNDEC](#)
- [TIMEFORMAT](#)
- [TRIMBLANKS](#)
- [TRUNCATECOLUMNS](#)

資料載入操作

指定下列參數來管理載入操作的預設行為，以進行故障排除或縮短載入時間。

- [COMPROWS](#)
- [COMPUPDATE](#)
- [IGNOREALLERRORS](#)
- [MAXERROR](#)

- [NOLOAD](#)
- [STATUPDATE](#)

COPY 命令的使用備註和其他資源

如需如何使用 COPY 命令的詳細資訊，請參閱下列主題：

- [使用須知](#)
- [教學課程：從 Amazon S3 載入資料](#)
- [載入資料的 Amazon Redshift 最佳實務](#)
- [使用 COPY 命令載入資料表](#)
 - [從 Amazon S3 載入資料](#)
 - [從 Amazon 載入資料 EMR](#)
 - [從遠端主機載入資料](#)
 - [從 Amazon DynamoDB 資料表載入資料](#)
- [針對資料載入進行故障診斷](#)

COPY 命令範例

如需示範如何 COPY 從不同來源、不同格式和使用不同 COPY 選項的更多範例，請參閱 [COPY 範例](#)。

COPYJOB(預覽)

這是預覽版本中的 autcopy (SQLCOPYJOB) 的發行前說明文件。文件和功能會隨時變更。我們建議僅在測試環境中使用此功能，不要在生產環境中使用。公開預覽將於 2024 年 10 月 31 日結束。預覽叢集將在預覽版結束的兩週後自動移除。如需預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

如需有關在檢視中使用此命令的資訊，請參閱 [從 Amazon S3 載入具有連續檔案擷取的資料表 \(預覽\)](#)。

管理將資料載入表格的 COPY 指令。該 COPYJOB 命令是命令的延伸，可自動從 Amazon S3 儲存貯體載入資料。COPY 當您建立 COPY 任務時，Amazon Redshift 會偵測是否在指定路徑中建立新的 Amazon S3 檔案，然後自動載入這些檔案，而無需您介入。載入資料時，會使用原始 COPY 指令中使用的相同參數。Amazon Redshift 會追蹤載入的檔案，以確認檔案只載入一次。

Note

如需有關COPY指令的資訊，包括用法、參數和權限，請參閱[COPY](#)。

所需的許可

要運行的COPY命令 COPYJOB，您必須具有被加載的表的INSERT權限。

使用COPY命令指定的IAM角色必須具有存取要載入之資料的權限。如需詳細資訊，請參閱[IAM COPY、UNLOAD和的許可 CREATE LIBRARY](#)。

語法

建立複製任務。COPY命令的參數會與複製工作一起儲存。

```
COPY copy-command JOB CREATE job-name  
[AUTO ON | OFF]
```

變更複製任務的組態。

```
COPY JOB ALTER job-name  
[AUTO ON | OFF]
```

執行複製任務。會使用儲存的COPY指令參數。

```
COPY JOB RUN job-name
```

列出所有複製任務。

```
COPY JOB LIST
```

顯示複製任務的詳細資訊。

```
COPY JOB SHOW job-name
```

刪除複製任務。

```
COPY JOB DROP job-name
```

參數

copy-command

將資料從 Amazon S3 載入到亞馬 Amazon Redshift 的 COPY 命令。該子句包含 COPY 定義 Amazon S3 儲存貯體、目標資料表、IAM 角色和載入資料時使用的其他參數的參數。支援 Amazon S3 資料載入的所有 COPY 命令參數，但以下情況除外：

- COPYJOB 不會將預先存在的檔案夾內擷取命令所指向的 COPY 檔案。只有在建立時間戳記之 COPYJOB 後建立的檔案才會擷取。
- 您無法使用 MAXERROR 或 IGNOREALLERRORS 選項指定指 COPY 令。
- 您不能指定資訊清單檔案。COPYJOB 需要指定的 Amazon S3 位置來監控新建立的檔案。
- 您不能使用諸如 Access 和秘密密鑰之類的授權類型來指定 COPY 命令。僅支援使 COPY 用該 IAM_ROLE 參數進行授權的命令。如需詳細資訊，請參閱 [授權參數](#)。
- COPYJOB 不支援與叢集相關聯的預設 IAM 角色。您必須 IAM_ROLE 在指 COPY 令中指定。

如需詳細資訊，請參閱 [COPY 從 Amazon S3](#)。

job-name

用來參照工作的工 COPY 作名稱。

[AUTO 開啟 | OFF]

指出是否將 Amazon S3 資料自動載入 Amazon Redshift 資料表的子句。

- Amazon Redshift 何時 ON 會監控新建立檔案的來源 Amazon S3 路徑，如果找到，則會使用任務定義中的 COPY 參數執行 COPY 命令。此為預設值。
- 什麼時候 OFF，Amazon Redshift 不會 COPYJOB 自動運行。

使用須知

COPY 命令的選項在運行時之前不會驗證。例如，無效 IAM_ROLE 或 Amazon S3 資料來源會在 COPYJOB 啟動時導致執行階段錯誤。

如果叢集已暫停，則不 COPYJOBS 會執行。

若要查詢載入的 COPY 指令檔案和載入錯誤，請參閱

[STL_LOAD_COMMITS](#) [STL_LOAD_ERRORS](#)，，[STL_LOADERROR_DETAIL](#)。如需詳細資訊，請參閱 [驗證資料已正確載入](#)。

範例

下列範例顯示建立COPYJOB以從 Amazon S3 儲存貯體載入資料的方式。

```
COPY public.target_table
FROM 's3://amzn-s3-demo-bucket/staging-folder'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyLoadRoleName'
JOB CREATE my_copy_job_name
AUTO ON;
```

COPY 參數參考

COPY 有許多參數可在許多情況下使用。但是，並非每種情況都支援所有參數。例如，若要從 ORC 或 PARQUET 檔案載入，支援的參數數量有限。如需詳細資訊，請參閱[COPY 從資料欄資料格式](#)。

主題

- [資料來源](#)
- [授權參數](#)
- [欄映射選項](#)
- [資料格式參數](#)
- [檔案壓縮參數](#)
- [資料轉換參數](#)
- [資料載入操作](#)
- [依字母排序的參數清單](#)

資料來源

您可以從 Amazon S3 儲存貯體、Amazon EMR 叢集或叢集可以使用 SSH 連線存取的遠端主機的文字檔案載入資料。您也可以直接從 DynamoDB 資料表載入資料。

任何來源的單一輸入資料列最大為 4 MB。

若要將資料表中的資料匯出至 Amazon S3 中的一組檔案，請使用 [UNLOAD](#) 命令。

主題

- [COPY 從 Amazon S3](#)
- [COPY 從 Amazon EMR](#)
- [COPY 從遠端主機 \(SSH \)](#)

- [COPY 從 Amazon DynamoDB](#)

COPY 從 Amazon S3

若要從位於一或多個 S3 儲存貯體中的檔案載入資料，請使用 FROM 子句來指示檔案在 Amazon S3 中 COPY 的位置。您可以提供資料檔案的物件路徑作為 FROM 子句的一部分，也可以提供包含 Amazon S3 物件路徑清單的清單檔案的位置。COPY 從 Amazon S3 使用 HTTPS 連線。確定 S3 IP 範圍已新增至您的允許清單。若要進一步了解所需的 S3 IP 範圍，請參閱[網路隔離](#)。

Important

如果存放資料檔案的 Amazon S3 儲存貯體與叢集不位於相同的 AWS 區域，您必須使用 [REGION](#) 參數來指定資料所在的區域。

主題

- [語法](#)
- [範例](#)
- [選用的參數](#)
- [不支援的參數](#)

語法

```
FROM { 's3://objectpath' | 's3://manifest_file' }  
authorization  
| MANIFEST  
| ENCRYPTED  
| REGION [AS] 'aws-region'  
| optional-parameters
```

範例

下列範例使用物件路徑以從 Amazon S3 載入資料。

```
copy customer  
from 's3://amzn-s3-demo-bucket/customer'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

下列範例使用資訊清單檔案以從 Amazon S3 載入資料。

```
copy customer
from 's3://amzn-s3-demo-bucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

參數

FROM

載入的資料來源。如需 Amazon S3 檔案編碼的相關資訊，請參閱[資料轉換參數](#)。

's3://copy_from_s3_objectpath'

指定含有資料之 Amazon S3 物件的路徑 Amazon S3 - 例如 's3://amzn-s3-demo-bucket/custdata.txt'。s3://copy_from_s3_objectpath 參數可以參考單一檔案，或一組有相同金鑰前綴的物件或資料夾。例如，custdata.txt 這個名稱是參考一些實體檔案的金鑰前綴：custdata.txt、custdata.txt.1、custdata.txt.2、custdata.txt.bak 等。金鑰前綴也可以參考一些資料夾。例如，'s3://amzn-s3-demo-bucket/custfolder' 參考資料夾 custfolder、custfolder_1、custfolder_2 等。如果金鑰前綴參考多個資料夾，則會載入這些資料夾中的所有檔案。如果金鑰字首符合檔案和資料夾，例如 custfolder.log，也會 COPY 嘗試載入檔案。如果金鑰字首可能會導致 COPY 嘗試載入不需要的檔案，請使用資訊清單檔案。如需詳細資訊，請參閱下列 [copy_from_s3_manifest_file](#)。

Important

如果存放資料檔案的 S3 儲存貯體與叢集不位於相同的 AWS 區域，您必須使用 [REGION](#) 參數來指定資料所在的區域。

如需詳細資訊，請參閱[從 Amazon S3 載入資料](#)。

's3://copy_from_s3_manifest_file'

指定資訊清單檔案 (列出要載入的資料檔案) 的 Amazon S3 物件金鑰。's3://copy_from_s3_manifest_file' 引數必須明確參考單一檔案 - 例如 's3://amzn-s3-demo-bucket/manifest.txt'。無法參考金鑰前綴。

資訊清單是 JSON 格式的文字檔案，列出要從 Amazon S3 載入的每個檔案 URL 的。URL 包含檔案的儲存貯體名稱和完整物件路徑。在資訊清單中指定的檔案可以位於不同的儲存貯體中，但所有

儲存貯體都必須與 Amazon Redshift 叢集位於相同的 AWS 區域中。如果某個檔案列出兩次，則該檔案會載入兩次。下列範例顯示載入三個檔案之JSON清單清單的。

```
{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata.1","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket1/custdata.2","mandatory":true},
    {"url":"s3://amzn-s3-demo-bucket2/custdata.1","mandatory":false}
  ]
}
```

雙引號字元是必要的，且必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。資訊清單中的每個項目可以選擇性包含 mandatory 旗標。如果 mandatory 設定為 true，如果找不到該項目的檔案，會COPY終止；否則，COPY會繼續。mandatory 的預設值為 false。

從資料檔案以 ORC或 Parquet 格式載入時，需要一個meta欄位，如下列範例所示。

```
{
  "entries":[
    {
      "url":"s3://amzn-s3-demo-bucket1/orc/2013-10-04-custdata",
      "mandatory":true,
      "meta":{
        "content_length":99
      }
    },
    {
      "url":"s3://amzn-s3-demo-bucket2/orc/2013-10-05-custdata",
      "mandatory":true,
      "meta":{
        "content_length":99
      }
    }
  ]
}
```

即使已指定 ENCRYPTED、GZIP、或 LZOPZSTD選項，也不得加密BZIP2或壓縮資訊清單檔案。COPY 如果找不到指定的資訊清單檔案或未正確形成資訊清單檔案，會傳回錯誤。

如果使用資訊清單檔案，則必須使用 COPY命令指定 MANIFEST 參數。如果未指定 MANIFEST 參數，會COPY假設使用 指定的檔案FROM是資料檔案。

如需詳細資訊，請參閱[從 Amazon S3 載入資料](#)。

authorization

此COPY命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2。您可以參考連接至叢集的 AWS Identity and Access Management (IAM) 角色 (以角色為基礎的存取控制)，或提供使用者的存取憑證 (以金鑰為基礎的存取控制)，以提供該授權。為了提高安全性和靈活性，我們建議您使用IAM角色型存取控制。如需詳細資訊，請參閱[授權參數](#)。

MANIFEST

指定使用清單檔案來指出要從 Amazon S3 載入的資料檔案。如果使用 MANIFEST 參數，會從 's3://copy_from_s3_manifest_file' 所參考資訊清單中列出的檔案COPY載入資料。如果找不到資訊清單檔案，或未正確形成，會COPY失敗。如需詳細資訊，請參閱[使用資訊清單指定資料檔案](#)。

ENCRYPTED

此子句指定 Amazon S3 上的輸入檔案是以用戶端加密搭配客戶管理金鑰來加密。如需詳細資訊，請參閱[從 Amazon S3 載入加密的資料檔案](#)。請勿指定輸入檔案ENCRYPTED是否使用 Amazon S3 伺服器端加密 (SSE-KMS 或 SSE-S3) 進行加密。COPY 會自動讀取伺服器端加密檔案。

如果您指定 ENCRYPTED 參數，也必須指定 [MASTER_SYMMETRIC_KEY](#) 參數或在 [CREDENTIALS](#) 字串中包含 `master_symmetric_key` 值。

如果加密的檔案為壓縮格式，請新增 GZIP、LZOP、BZIP2或 ZSTD 參數。

即使已指定 ENCRYPTED選項，也不得加密清單檔案和JSONPaths檔案。

MASTER_SYMMETRIC_KEY 'root_key'

用於將 Amazon S3 上的資料檔案加密的根對稱金鑰。如果指定 MASTER_SYMMETRIC_KEY，也必須指定 [ENCRYPTED](#) 參數。MASTER_SYMMETRIC_KEY 無法與 CREDENTIALS 參數搭配使用。如需詳細資訊，請參閱[從 Amazon S3 載入加密的資料檔案](#)。

如果加密的檔案為壓縮格式，請新增 GZIP、LZOP、BZIP2或 ZSTD 參數。

REGION 【AS】 'aws-region'

指定來源資料所在的 AWS 區域。REGION 當包含資料 AWS 的資源不在與 Amazon Redshift 叢集相同的區域中時，需要COPY從 Amazon S3 儲存貯體或 DynamoDB 資料表進行。Amazon S3

aws_region 的值必須符合 [Amazon Redshift 區域與端點](#) 資料表所列的區域。

如果指定 REGION 參數，則所有資源，包括資訊清單檔案或多個 Amazon S3 儲存貯體，都必須位於指定的區域中。

 Note

跨區域傳輸資料需要為含有資料的 Amazon S3 儲存貯體或 DynamoDB 資料表另外付費。如需定價的詳細資訊，請參閱 Amazon S3 定價頁面上的OUT從 Amazon S3 到另一個 AWS 區域的資料傳輸，以及 [Amazon DynamoDB 定價](#) 頁面上的資料傳輸OUT。 [Amazon S3](#)

依預設，會COPY假設資料與 Amazon Redshift 叢集位於相同的區域。

選用的參數

您可以選擇性地COPY從 Amazon S3 使用 指定下列參數：

- [欄映射選項](#)
- [資料格式參數](#)
- [資料轉換參數](#)
- [資料載入操作](#)

不支援的參數

您無法COPY從 Amazon S3 將下列參數與 搭配使用：

- SSH
- READRATIO

COPY 從 Amazon EMR

您可以使用 COPY命令，從EMR設定為將文字檔案寫入叢集的 Hadoop 分散式檔案系統（HDFS），以固定寬度檔案、字元分隔檔案、CSV檔案、JSON格式化檔案或 Avro 檔案的形式平行載入資料。

主題

- [語法](#)

- [範例](#)
- [參數](#)
- [支援的參數](#)
- [不支援的參數](#)

語法

```
FROM 'emr://emr_cluster_id/hdfs_filepath'  
authorization  
[ optional_parameters ]
```

範例

下列範例會從 Amazon EMR 叢集載入資料。

```
copy sales  
from 'emr://j-SAMPLE2B500FC/myoutput/part-*'   
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

參數

FROM

載入的資料來源。

'emr://emr_cluster_id/hdfs_file_path'


Amazon EMR 叢集的唯一識別符，以及參考 COPY 命令資料檔案 HDFS 的檔案路徑。HDFS 資料檔案名稱不得包含萬用字元星號 (*) 和問號 (?)。

Note

Amazon EMR 叢集必須繼續執行，直到 COPY 操作完成。如果在 COPY 操作完成之前變更或刪除任何 HDFS 資料檔案，則可能會出現非預期的結果，或者 COPY 操作可能會失敗。

您可以在 `hdfs_file_path` 引數中使用萬用字元星號 (*) 和問號 (?) 來指定要載入的多個檔案。例如，`'emr://j-SAMPLE2B500FC/myoutput/part*'` 表示檔案 `part-0000`、`part-0001`，以

此類推。如果檔案路徑不含萬用字元，則視為字串常值。如果您只指定資料夾名稱，會COPY嘗試載入資料夾中的所有檔案。

 Important

如果使用萬用字元或只使用資料夾名稱，請確認不會載入不需要的檔案。例如，某些程序可能將日誌檔案寫入至輸出資料夾。

如需詳細資訊，請參閱[從 Amazon 載入資料 EMR](#)。

authorization

此COPY命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2。您可以參考[連接至叢集的 AWS Identity and Access Management \(IAM\) 角色](#)（以角色為基礎的存取控制），或提供使用者的存取憑證（以金鑰為基礎的存取控制），以提供該授權。為了提高安全性和靈活性，我們建議您使用IAM角色型存取控制。如需詳細資訊，請參閱[授權參數](#)。

支援的參數

您可以選擇COPY從 Amazon 使用 指定下列參數EMR：

- [欄映射選項](#)
- [資料格式參數](#)
- [資料轉換參數](#)
- [資料載入操作](#)

不支援的參數

您無法將下列參數與COPY來自 Amazon 的 搭配使用EMR：

- ENCRYPTED
- MANIFEST
- REGION
- READRATIO
- SSH

COPY 從遠端主機 (SSH)

您可以使用 COPY 命令從一或多個遠端主機平行載入資料，例如 Amazon Elastic Compute Cloud (AmazonEC2) 執行個體或其他電腦。COPY 使用 Secure Shell (SSH) 連線至遠端主機，並在遠端主機上執行命令以產生文字輸出。遠端主機可以是 EC2 Linux 執行個體，或是設定為接受 SSH 連線的另一個 Unix 或 Linux 電腦。Amazon Redshift 可以連線至多個主機，也可以開啟每個主機的多個 SSH 連線。Amazon Redshift 會透過每個連線傳送一個唯一的命令，將文字輸出產生到主機的標準輸出，然後 Amazon Redshift 會像讀取文字檔一樣讀取該輸出。

使用 FROM 子句來指定資訊清單檔案的 Amazon S3 物件金鑰，提供 COPY 用來開啟 SSH 連線和執行遠端命令的資訊。

主題

- [語法](#)
- [範例](#)
- [參數](#)
- [選用的參數](#)
- [不支援的參數](#)

Important

如果存放資訊清單檔案的 S3 儲存貯體與叢集不位於同一個 AWS 區域，您必須使用 REGION 參數來指定儲存貯體所在的區域。

語法

```
FROM 's3://'ssh_manifest_file' }  
authorization  
SSH  
| optional-parameters
```

範例

下列範例使用資訊清單檔案，使用 從遠端主機載入資料SSH。

```
copy sales
```

```
from 's3://amzn-s3-demo-bucket/ssh_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
ssh;
```

參數

FROM

載入的資料來源。

```
's3://copy_from_ssh_manifest_file'
```

COPY 命令可以使用連線到多個主機SSH，並可以建立與每個主機的多個SSH連線。COPY 會透過每個主機連線執行命令，然後將命令的輸出平行載入資料表。s3 : //copy_from_ssh_manifest_file 引數指定資訊清單檔案的 Amazon S3 物件索引鍵，提供資訊COPY用於開啟SSH連線和執行遠端命令。

s3://copy_from_ssh_manifest_file 引數必須明確參考單一檔案；不能是金鑰字首。如下列範例所示：

```
's3://amzn-s3-demo-bucket/ssh_manifest.txt'
```

資訊清單檔案是 Amazon Redshift 用來連線至主機JSON的文字檔案。資訊清單檔案會指定SSH主機端點，以及在主機上執行的命令，以將資料傳回 Amazon Redshift。您可以選擇在每個項目中包含主機公有金鑰、登入使用者名稱及 mandatory 旗標。下列範例顯示建立兩個SSH連線的資訊清單檔案：

```
{
  "entries": [
    {
      "endpoint": "<ssh_endpoint_or_IP>",
      "command": "<remote_command>",
      "mandatory": true,
      "publickey": "<public_key>",
      "username": "<host_user_name>"
    },
    {
      "endpoint": "<ssh_endpoint_or_IP>",
      "command": "<remote_command>",
      "mandatory": true,
      "publickey": "<public_key>",
      "username": "<host_user_name>"
    }
  ]
}
```

資訊清單檔案包含每個SSH連線的一個"entries"建構。您可以對單一主機建立多個連線，也可以對多台主機建立多個連線。如圖所示，欄位名稱和值都需要雙引號字元。引號字元必須是簡單引號(0x22)，而不是斜向或「智慧型」引號。唯一不需要雙引號字元的值是 "mandatory" 欄位的布林值 true 或 false。

下列清單描述資訊清單檔案中的欄位。

端點

主機URL的地址或 IP 地址，例如 "ec2-111-222-333.compute-1.amazonaws.com" 或 "198.51.100.0"。

command

主機執行的命令，會產生文字輸出或 gzip、lzop、bzip2 或 zstd 格式的二進位輸出。命令可以是使用者 "host_user_name" 有許可執行的任何命令。命令可能像是列印檔案這麼簡單，也可能是查詢資料庫或啟動指令碼。輸出（文字檔案、gzip 二進位檔案、lzop 二進位檔案或 bzip2 二進位檔案）必須採用 Amazon Redshift COPY命令可以擷取的形式。如需詳細資訊，請參閱[準備您的輸入資料](#)。

publickey

(選用) 主機的公有金鑰。如果提供，Amazon Redshift 會使用公有金鑰來識別主機。如果未提供公有金鑰，Amazon Redshift 不會嘗試識別主機。例如，若遠端主機的公有金鑰是 ssh-rsa AbcCbaxxx...Example root@amazon.com，請在公有金鑰欄位中輸入下列文字："AbcCbaxxx...Example"

mandatory

(選用) 子句，指出如果連線嘗試失敗，COPY命令是否應該失敗。預設值為 false。如果 Amazon Redshift 無法成功建立至少一個連線，則COPY命令會失敗。

使用者名稱

(選用) 用來登入主機系統並執行遠端命令的使用者名稱。使用者登入名稱與用來將 Amazon Redshift 叢集公有金鑰新增至主機授權金鑰檔案的登入必須相同。預設使用者名為 redshift。

如需建立資訊清單檔案的相關資訊，請參閱[載入資料程序](#)。

若要COPY從遠端主機執行，必須使用 COPY命令指定 SSH 參數。如果未指定 SSH 參數，會 COPY假設使用 指定的檔案FROM是資料檔案，且 將會失敗。

如果您使用自動壓縮，COPY命令會執行兩個資料讀取操作，這表示它將執行遠端命令兩次。第一次讀取操作會提供資料樣本進行壓縮分析，第二次讀取操作就會實際載入資料。如果遠端命令的兩

次執行可能造成問題，請停用自動壓縮。若要停用自動壓縮，請執行 COMPUPDATE 參數設定為的 COPY 命令 OFF。如需詳細資訊，請參閱 [利用自動壓縮載入資料表](#)。

如需 COPY 從使用的詳細程序 SSH，請參閱 [從遠端主機載入資料](#)。

authorization

此 COPY 命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2。您可以參考連接至叢集的 AWS Identity and Access Management (IAM) 角色（以角色為基礎的存取控制），或提供使用者的存取憑證（以金鑰為基礎的存取控制），以提供該授權。為了提高安全性和靈活性，我們建議您使用 IAM 角色型存取控制。如需詳細資訊，請參閱 [授權參數](#)。

SSH

指定要使用 SSH 通訊協定從遠端主機載入資料的子句。如果您指定 SSH，也必須使用 [s3://copy_from_ssh_manifest_file](#) 引數提供資訊清單檔案。

Note

如果您使用從主機使用遠端中的私有 IP 地址進行 SSH 複製 VPC，VPC 則必須啟用增強型 VPC 路由。如需增強型 VPC 路由的詳細資訊，請參閱 [Amazon Redshift 增強型 VPC 路由](#)。

選用的參數

您可以從選擇性地使用 COPY 指定下列參數 SSH：

- [欄映射選項](#)
- [資料格式參數](#)
- [資料轉換參數](#)
- [資料載入操作](#)

不支援的參數

您無法將下列參數與 COPY 搭配使用 SSH：

- ENCRYPTED
- MANIFEST

- READRATIO

COPY 從 Amazon DynamoDB

若要從現有的 DynamoDB 資料表載入資料，請使用 FROM 子句來指定 DynamoDB 資料表名稱。

主題

- [語法](#)
- [範例](#)
- [選用的參數](#)
- [不支援的參數](#)

Important

如果 DynamoDB 資料表與 Amazon Redshift 叢集不位於相同的區域，您必須使用 REGION 參數來指定資料所在的區域。

語法

```
FROM 'dynamodb://table-name'  
authorization  
READRATIO ratio  
| REGION [AS] 'aws_region'  
| optional-parameters
```

範例

下列範例從 DynamoDB 資料表載入資料。

```
copy favoritemovies from 'dynamodb://ProductCatalog'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
readratio 50;
```

參數

FROM

載入的資料來源。

'dynamodb://table-name'

包含資料之 DynamoDB 資料表的名稱，例如 'dynamodb://ProductCatalog'。如需 DynamoDB 屬性如何映射至 Amazon Redshift 欄的詳細資訊，請參閱[從 Amazon DynamoDB 資料表載入資料](#)。

DynamoDB 資料表名稱對 AWS 帳戶是唯一的，由 AWS 存取憑證識別。

authorization

此 COPY 命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、DynamoDB 和 Amazon EC2。您可以參考連接至叢集的 AWS Identity and Access Management (IAM) 角色（以角色為基礎的存取控制），或提供使用者的存取憑證（以金鑰為基礎的存取控制），以提供該授權。為了提高安全性和靈活性，我們建議您使用 IAM 角色型存取控制。如需詳細資訊，請參閱[授權參數](#)。

READRATIO 【AS】 比率

DynamoDB 資料表的佈建輸送量中用於資料載入的百分比。READRATIO 從 COPY DynamoDB 需要的。無法搭配 COPY 來自 Amazon S3 的使用。強烈建議將此比率設定為小於平均未用佈建輸送量的值。有效值為整數 1–200。

Important

READRATIO 將設為 100 或更高，可讓 Amazon Redshift 消耗 DynamoDB 資料表佈建的輸送量，這在 COPY 工作階段期間會嚴重降低相同資料表並行讀取操作的效能。寫入流量不受影響。在 Amazon Redshift 無法滿足資料表的佈建輸送量的罕見情況下，允許使用高於 100 的值來排解此問題。如果您持續將資料從 DynamoDB 載入至 Amazon Redshift，請考慮將您的 DynamoDB 資料表組織為時間序列，以將即時流量與 COPY 操作分開。

選用的參數

您可以選擇 COPY 從 Amazon DynamoDB 使用 指定下列參數：

- [欄映射選項](#)
- 支援下列資料轉換參數：
 - [ACCEPTANYDATE](#)
 - [BLANKSASNULL](#)
 - [DATEFORMAT](#)

- [EMPTYASNULL](#)
- [ROUNDEC](#)
- [TIMEFORMAT](#)
- [TRIMBLANKS](#)
- [TRUNCATECOLUMNS](#)
- [資料載入操作](#)

不支援的參數

您無法COPY從 DynamoDB 將下列參數與 搭配使用：

- 所有資料格式參數
- ESCAPE
- FILLRECORD
- IGNOREBLANKLINES
- IGNOREHEADER
- NULL
- REMOVEQUOTES
- ACCEPTINVCHARS
- MANIFEST
- ENCRYPTED

授權參數

此COPY命令需要授權才能存取其他 AWS 資源中的資料，包括 Amazon S3、Amazon EMR、Amazon DynamoDB 和 Amazon EC2。您可以參考連接至叢集的 [AWS Identity and Access Management \(IAM\) 角色](#) (角色型存取控制)，以提供該授權。您可以在 Amazon S3 上加密載入資料。

下列主題提供身分驗證選項的詳細資訊和範例：

- [IAM COPY、UNLOAD和 的許可 CREATE LIBRARY](#)
- [角色類型存取控制](#)
- [金鑰型存取控制](#)

使用下列其中一項來提供COPY命令的授權：

- [IAM_ROLE](#) 參數
- [ACCESS_KEY_ID and SECRET_ACCESS_KEY](#) 參數
- [CREDENTIALS](#) 子句

```
IAM_ROLE { 預設 | 'arn : aws : iam : : <AWS ##-id> : 角色 / <role-name>' }
```

使用預設關鍵字讓 Amazon Redshift 在COPY命令執行時，使用設定為預設並與叢集相關聯的IAM角色。

針對叢集用於身分驗證和授權IAM的角色，請使用 Amazon Resource Name (ARN)。如果您指定 IAM_ROLE，則無法使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY、SESSION_TOKEN 或 CREDENTIALS。

下列顯示 IAM_ROLE 參數的語法。

```
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
```

如需詳細資訊，請參閱[角色類型存取控制](#)。

```
ACCESS_KEY_ID 'access-key-id ' SECRET_ACCESS_KEY 'secret-access-key'
```

不建議使用此授權方法。

Note

強烈建議您指定 IAM_ROLE 參數，而不是以純文字形式提供存取憑證，而是使用角色型身分驗證。如需詳細資訊，請參閱[角色類型存取控制](#)。

```
SESSION_TOKEN '暫時金鑰 '
```

用於暫時存取登入資料的工作階段字符。指定 SESSION_TOKEN 時，您還必須使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY 來提供臨時存取金鑰憑證。如果您指定 SESSION_TOKEN，則無法使用 IAM_ROLE 或 CREDENTIALS。如需詳細資訊，請參閱 IAM 使用者指南[暫時安全憑證](#)中的。

Note

除了建立暫時安全登入資料，強烈建議使用角色型身分驗證。當您使用IAM角色授權時，Amazon Redshift 會自動為每個工作階段建立臨時使用者憑證。如需詳細資訊，請參閱[角色類型存取控制](#)。

以下顯示 SESSION__TOKENID 和 ACCESS_KEY_ 參數的 SECRETACCESS_KEY 參數語法。

```
ACCESS_KEY_ID '<access-key-id>'
SECRET_ACCESS_KEY '<secret-access-key>'
SESSION_TOKEN '<temporary-token>';
```

如果您指定 SESSION_TOKEN，則無法使用 CREDENTIALS或 IAM_ROLE。

【WITH】 CREDENTIALS 【AS】 'credentials-args'

指出叢集在存取包含資料檔案或資訊清單檔案的其他 AWS 資源時將使用的方法的子句。您無法將 CREDENTIALS 參數與 IAM_ROLE 或 ACCESS_KEY_ID 和 SECRET_ACCESS_ 搭配使用KEY。

Note

為了提高彈性，我們建議您使用 [IAM_ROLE](#) 參數，而非 CREDENTIALS 參數。

(選用) 如果使用 [ENCRYPTED](#) 參數，則 credentials-args 字串也提供加密金鑰。

credentials-args 字串區分大小寫，且不得包含空格。

關鍵字WITH和 AS 是選用項目，且會被忽略。

您可指定為 [role-based access control](#) 或 [key-based access control](#)。無論哪種情況，IAM角色或使用者都必須具有存取指定 AWS 資源所需的許可。如需詳細資訊，請參閱[IAM COPY、UNLOAD和的許可 CREATE LIBRARY](#)。

Note

為了保護您的 AWS 憑證和保護敏感資料，我們強烈建議使用角色型存取控制。

若要指定角色型存取控制，請依下列格式提供 credentials-args 字串。

```
'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
```

若要使用暫時字符登入資料，您必須提供暫時存取金鑰 ID、暫時私密存取金鑰及暫時字符。credentials-args 字串採用下列格式。

```
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;token=<temporary-token>'
```

如需詳細資訊，請參閱[暫時安全憑證](#)。

如果使用 [ENCRYPTED](#) 參數，Credential-args 字串會以下列格式顯示，其中 *<root-key>* 是用來加密檔案的根金鑰值。

```
CREDENTIALS
'<credentials-args>;master_symmetric_key=<root-key>'
```

例如，下列COPY命令使用角色型存取控制搭配加密金鑰。

```
copy customer from 's3://amzn-s3-demo-bucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::<account-id>:role/<role-name>;master_symmetric_key=<root-key>'
```

下列COPY命令顯示具有加密金鑰的角色型存取控制。

```
copy customer from 's3://amzn-s3-demo-bucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>;master_symmetric_key=<root-key>'
```

欄映射選項

依預設，會以與資料檔案中欄位相同的順序，將值COPY插入目標資料表的資料欄。如果預設資料欄順序無法運作，您可以指定資料欄清單或使用JSONPath運算式將來源資料欄位映射到目標資料欄。

- [Column List](#)
- [JSONPaths File](#)

資料欄清單

您可以指定逗號分隔的欄名稱清單，以便將來源資料欄位載入特定的目標欄。這些資料欄可以在 COPY 陳述式中以任何順序排列，但從平面檔案載入時，例如在 Amazon S3 儲存貯體中，其順序必須符合來源資料的順序。

從 Amazon DynamoDB 資料表載入時，順序並不重要。COPY 命令會將從 DynamoDB 資料表擷取的項目中的屬性名稱與 Amazon Redshift 資料表中的資料欄名稱相符。如需詳細資訊，請參閱 [從 Amazon DynamoDB 資料表載入資料](#)

欄清單的格式如下。

```
COPY tablename (column1 [,column2, ...])
```

如果目標資料表中的資料欄從資料欄清單中省略，則會 COPY 載入目標資料欄的 [DEFAULT](#) 表達式。

如果目標欄沒有預設值，則會 COPY 嘗試載入 NULL。

如果 COPY 嘗試 NULL 將指派給定義為 NOT 的欄 NULL，則 COPY 命令會失敗。

如果資料 [IDENTITY](#) 欄包含在資料欄清單中，則必須 [EXPLICIT_IDS](#) 指定資料欄；如果省略資料 IDENTITY 欄，則無法指定 EXPLICIT_IDS。如果未指定任何資料欄清單，命令的行為就好像指定了完整的順序資料欄清單，如果還未指定 EXPLICIT_IDS，則會省略資料 IDENTITY 欄。

如果使用 GENERATED BY DEFAULT AS 定義資料欄 IDENTITY，則可以複製該資料欄。值是利用您提供的值來產生或更新。不需要 EXPLICIT_IDS 選項。COPY 不會更新身分高浮水印。如需詳細資訊，請參閱 [GENERATED BY DEFAULT AS IDENTITY](#)。

JSONPaths 檔案

從 JSON 或 Avro 格式的資料檔案載入時，JSON 或 Avro 來源資料中的資料元素 COPY 會自動對應至目標資料表中的資料欄。它會比對 Avro 結構定義中的欄位名稱與目標資料表或欄清單中的欄名稱，以達到這個目標。

在某些情況下，您的欄名稱和欄位名稱不相符，或者您需要映射至資料階層中的更深層級。在這些情況下，您可以使用 JSONPaths 檔案將 JSON 或 Avro 資料元素明確映射至資料欄。

如需詳細資訊，請參閱 [JSONPaths 檔案](#)。

資料格式參數

根據預設，COPY 命令預期來源資料為字元分隔 UTF-8 文字。預設分隔符號是縱線字元 (|)。如果來源資料是其他格式，請使用下列參數來指定資料格式：

- [FORMAT](#)
- [CSV](#)
- [DELIMITER](#)
- [FIXEDWIDTH](#)
- [SHAPEFILE](#)
- [AVRO](#)
- [JSON](#)
- [PARQUET](#)
- [ORC](#)

除了標準資料格式之外，COPY還支援COPY來自 Amazon S3 的下列資料欄資料格式：

- [ORC](#)
- [PARQUET](#)

COPY 資料欄格式支援特定限制。如需詳細資訊，請參閱[COPY 從資料欄資料格式](#)。

資料格式參數

FORMAT 【AS】

(選用) 識別資料格式關鍵字。引FORMAT數說明如下。

CSV QUOTE 【【AS】 'quote_character'】

啟用輸入資料中的CSV格式。若要自動逸出分隔符號、新行字元和歸位字元，請在 QUOTE 參數指定的字元中包含 欄位。預設引號字元為雙引號 (")。在欄位內使用引號字元時，請多加一個引號字元來逸出此字元。例如，假設引號字元是雙引號，若要插入 A "quoted" word 字串，則輸入檔案應該包含 "A ""quoted"" word" 字串。使用 CSV 參數時，預設分隔符號為逗號 (,)。您可以使用 DELIMITER 參數指定不同的分隔符號。

以引號括住欄位時，分隔字元和引號字元之間的空格會被忽略。如果分隔字元是空格字元 (例如 Tab 字元)，則不會將分隔字元視為空格。

CSV 無法與 FIXEDWIDTH、REMOVEQUOTES或 搭配使用ESCAPE。

QUOTE 【AS】 'quote_character'

選用。指定使用 CSV 參數時要用作引號字元的字元。預設值為雙引號 (")。如果您使用 QUOTE 參數來定義雙引號以外的引號字元，則不需要在欄位中逸出雙引號。QUOTE 參數只能與 CSV 參數搭配使用。AS 關鍵字為選用。

DELIMITER 【AS】 ['delimiter_char']

指定用來分隔輸入檔案中欄位的字元，例如管道字元 | ()、逗號 , ()、標籤 (\t) 或多個字元，例如 |~|。支援不可列印的字元。字元也可以以八進位表示為 UTF-8 碼單位。對於八進位數字，請使用格式 '\ddd'，其中 'd' 是八進位數字 (0-7)。預設分隔符號為管道字元 (|)，除非使用 CSV 參數，在這種情況下，預設分隔符號為逗號 (,)。AS 關鍵字為選用。DELIMITER 無法與 搭配使用FIXEDWIDTH。

FIXEDWIDTH 'fixedwidth_spec'

從每個欄寬是固定長度 (而不是以分隔字元隔開的欄) 的檔案載入資料。fixedwidth_spec 是字串，指定使用者定義的欄標籤和欄寬。欄標籤可以是文字字串或整數 (視使用者的選擇而定)。欄標籤與欄名稱無關。標籤/寬度配對的順序必須完全符合資料表欄的順序。FIXEDWIDTH 無法與 CSV 或 搭配使用DELIMITER。在 Amazon Redshift 中，CHAR 和 VARCHAR資料欄的長度是以位元組表示，因此在準備要載入的檔案時，請確定您指定的資料欄寬度可容納多位元組字元的二進位長度。如需詳細資訊，請參閱[字元類型](#)。

fixedwidth_spec 的格式如下所示：

```
'colLabel1:colWidth1,colLabel:colWidth2, ...'
```

SHAPEFILE SIMPLIFY 【【AUTO】 【'公差'】】

啟用輸入資料中的SHAPEFILE格式。依預設，Shapefile 的第一個資料欄是 GEOMETRY 或 IDENTITY 資料欄。所有後續的資料欄都會遵循 Shapefile 中指定的順序。

您無法SHAPEFILE搭配 FIXEDWIDTH、 REMOVEQUOTES或 使用 ESCAPE。

若要搭配 GEOGRAPHY 物件使用 COPY FROM SHAPEFILE，請先擷取至 GEOMETRY 欄中，然後將物件轉換為 GEOGRAPHY 物件。

SIMPLIFY 【容差】

(選用) 使用 Ramer-Douglas-Peucker演算法和指定的容差，簡化擷取過程中的所有幾何。

SIMPLIFY AUTO 【容差】

(選用) 僅簡化大於最大幾何大小的幾何。如果未超過指定的公差，此簡化會使用 Ramer-Douglas-Peucker 演算法和自動計算公差。此演算法會在指定公差內計算儲存物件的大小。tolerance 值是選用的。

如需載入 Shapefile 的範例，請參閱 [將 Shapefile 載入 Amazon Redshift](#)。

AVRO 【AS】 'avro_option'

指定來源資料是 Avro 格式。

COPY 這些服務和通訊協定支援 Avro 格式：

- Amazon S3
- Amazon EMR
- 遠端主機 (SSH)

COPY DynamoDB 不支援 Avro 的。

Avro 是資料序列化通訊協定。Avro 來源檔案包含結構描述來定義資料的結構。Avro 結構描述類型必須是 record。COPY 接受使用預設未壓縮轉碼器以及 deflate 和 snappy 壓縮轉碼器建立的 Avro 檔案。如需 Avro 的相關資訊，請前往 [Apache Avro](#)。

avro_option 的有效值如下所示：

- 'auto'
- 'auto ignorecase'
- 's3://*jsonpaths_file*'

預設值為 'auto'。

COPY 會自動將 Avro 來源資料中的資料元素映射至目標資料表中的資料欄。它會比對 Avro 結構描述中的欄位名稱與目標資料表中的欄名稱。'auto' 的比對區分大小寫，而 'auto ignorecase' 的比對不區分大小寫。

Amazon Redshift 資料表中的欄名稱一律為小寫，因此當您使用 'auto' 選項時，相符欄位名稱也必須是小寫。如果欄位名稱並非全部小寫，您可以使用 'auto ignorecase' 選項。使用預設 'auto' 引數時，只會 COPY 辨識結構中欄位的第一層或外部欄位。

若要將欄名稱明確映射至 Avro 欄位名稱，您可以使用 [JSONPaths 檔案](#)。

根據預設，COPY會嘗試將目標資料表中的所有資料欄與 Avro 欄位名稱比對。若要載入欄子集，您可以選擇性指定欄清單。如果目標資料表中的資料欄從資料欄清單中省略，會COPY載入目標資料欄的`DEFAULT`表達式。如果目標欄沒有預設值，會COPY嘗試載入 NULL。如果資料欄包含在資料欄清單中，但 Avro 資料中COPY找不到相符的欄位，會COPY嘗試載入 NULL 資料欄。

如果COPY嘗試NULL將指派給定義為 NOT 的欄NULL，則COPY命令會失敗。

Avro 結構描述

Avro 來源資料檔案包含結構描述來定義資料的結構。COPY 會讀取屬於 Avro 來源資料檔案一部分的結構描述，以將資料元素對應至目標資料表資料欄。下列範例顯示 Avro 結構描述。

```
{
  "name": "person",
  "type": "record",
  "fields": [
    {"name": "id", "type": "int"},
    {"name": "guid", "type": "string"},
    {"name": "name", "type": "string"},
    {"name": "address", "type": "string"}]
}
```

Avro 結構描述是使用 JSON 格式定義的。頂層JSON物件包含三個名稱值對，其中包含名稱或索引鍵、"type"、"name"和"fields"。

內含物件陣列的 "fields" 金鑰對會定義資料結構中每個欄位的名稱和資料類型。根據預設，COPY會自動將欄位名稱與資料欄名稱相符。欄名稱一律為小寫，因此相符的欄位名稱也必須是小寫，除非您指定 'auto ignorecase' 選項。不符合欄名稱的任何欄位名稱會被忽略。順序並不重要。在上一個範例中，會COPY映射到資料欄名稱 id、guid、name 和 address。

使用預設 'auto' 引數，只會將第一級物件與資料欄COPY比對。若要映射到結構描述中的更深層，或者如果欄位名稱和資料欄名稱不相符，請使用 JSONPaths 檔案定義映射。如需詳細資訊，請參閱[JSONPaths 檔案](#)。

如果與金鑰相關聯的值是複雜的 Avro 資料類型，例如位元組、陣列、記錄、地圖或連結，會將值COPY載入為字串。此處，字串是資料的JSON表示法。COPY 會將 Avro 列舉資料類型載入為字串，其中內容是類型的名稱。如需範例，請參閱[COPY 從JSON格式](#)。

Avro 檔案標頭 (包括結構描述和檔案中繼資料) 的大小上限為 1 MB。

單一 Avro 資料區塊的大小上限為 4 MB。這不同於資料列大小上限。如果超過單一 Avro 資料區塊的大小上限，即使產生的資料列大小小於 4 MB 的資料列大小限制，COPY命令也會失敗。

在計算列大小時，Amazon Redshift 會在內部將縱線字元 (|) 計算兩次。如果輸入資料包含非常多的縱線字元，即使資料區塊小於 4 MB，資料列大小仍可能超過 4 MB。

JSON 【AS】 'json_option'

來源資料格式為 JSON 。

JSON COPY 從這些服務和通訊協定支援 的 格式：

- Amazon S3
- COPY 從 Amazon EMR
- COPY 從 SSH

JSON COPY DynamoDB 不支援 的 。

json_option 的有效值如下所示：

- 'auto'
- 'auto ignorecase'
- 's3://*jsonpaths_file*'
- 'noshred'

預設值為 'auto'。Amazon Redshift 不會在載入JSON文件時將JSON結構的屬性分割為多個資料欄。

根據預設，COPY會嘗試將目標資料表中的所有資料欄與JSON欄位名稱索引鍵比對。若要載入欄子集，您可以選擇性指定欄清單。如果JSON欄位名稱索引鍵不是全部小寫，您可以使用 'auto ignorecase' 選項或 [JSONPaths 檔案](#) 將資料欄名稱明確對應至JSON欄位名稱索引鍵。

如果目標資料表中的資料欄從資料欄清單中省略，則會COPY載入目標資料欄的[DEFAULT](#)表達式。如果目標欄沒有預設值，會COPY嘗試載入 NULL。如果資料欄包含在資料欄清單中，但COPY找不到相符的欄位JSON，會COPY嘗試載入NULL資料欄。

如果COPY嘗試NULL將指派給定義為 NOT 的欄NULL，則COPY命令會失敗。

COPY 會將JSON來源資料中的資料元素映射到目標資料表中的資料欄。它會將來源名稱值對中的物件金鑰 (或名稱) 比對目標資料表中的欄名稱來達成此目標。

請參閱每個 json_option 值的下列詳細資訊：

'auto'

使用此選項，比對會區分大小寫。Amazon Redshift 資料表中的資料欄名稱一律小寫，因此當您使用 'auto' 選項時，相符JSON的欄位名稱也必須小寫。

'auto ignorecase'

使用此選項，比對不會區分大小寫。Amazon Redshift 資料表中的資料欄名稱一律為小寫，因此當您使用 'auto ignorecase' 選項時，對應的JSON欄位名稱可以是小寫、大寫或混合大小寫。

's3://jsonpaths_file'

使用此選項，COPY會使用具名JSONPaths檔案，將JSON來源資料中的資料元素映射至目標資料表中的資料欄。`s3://jsonpaths_file` 引數必須是明確參考單一檔案的 Amazon S3 物件金鑰。例如 `'s3://amzn-s3-demo-bucket/jsonpaths.txt'`。引數不能是索引鍵字首。如需使用 JSONPaths 檔案的詳細資訊，請參閱 [the section called “JSONPaths 檔案”](#)。

在某些情況下，由 `jsonpaths_file` 指定的檔案具有與 `copy_from_s3_objectpath` 為資料檔指定的路徑相同的字首。如果是這樣，會將JSONPaths檔案COPY讀取為資料檔案並傳回錯誤。例如，假設資料檔案使用物件路徑，`s3://amzn-s3-demo-bucket/my_data.json`且JSONPaths檔案為 `s3://amzn-s3-demo-bucket/my_data.jsonpaths`。在此情況下，會COPY嘗試將 載入`my_data.jsonpaths`為資料檔案。

'noshred'

使用此選項時，Amazon Redshift 不會在載入JSON文件時將JSON結構的屬性分割為多個資料欄。

JSON 資料檔案

JSON 資料檔案包含一組物件或陣列。COPY 會將每個JSON物件或陣列載入目標資料表中的一列。對應至資料列的每個物件或陣列都必須是獨立的根層級結構；也就是說，它不得是另一個JSON結構的成員。

JSON 物件的開頭和結尾為 braces (`{ }`)，並包含未排序的名稱值對集合。每一對名稱和值以冒號分隔，而配對以逗號分隔。根據預設，名稱值對中的物件金鑰 (或名稱) 必須符合資料表中相應欄的名稱。Amazon Redshift 資料表中的資料欄名稱一律為小寫，因此相符JSON的欄位名稱索引鍵也必須為小寫。如果您的資料欄名稱和JSON索引鍵不相符，請使用 [the section called “JSONPaths 檔案”](#)明確對應至索引鍵。

JSON 物件中的順序並不重要。不符合欄名稱的任何名稱會被忽略。下列顯示簡單JSON物件的結構。

```
{
```

```
"column1": "value1",  
"column2": value2,  
"notacolumn" : "ignore this value"  
}
```

JSON 陣列以括號 (**[]**) 開頭和結尾，並包含以逗號分隔的有序值集合。如果您的資料檔案使用陣列，您必須指定一個JSONPaths檔案，將值與資料欄比對。下列顯示簡單JSON陣列的結構。

```
["value1", value2]
```

JSON 必須格式良好。例如，不能以逗號或其他任何字元 (空格除外) 來分隔物件或陣列。必須以雙引號字元括住字串。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。

單一JSON物件或陣列的大小上限為 4 MB，包括支架或括號。這不同於資料列大小上限。如果超過單一JSON物件或陣列的大小上限，即使產生的資料列大小小於 4 MB 的資料列大小限制，COPY命令也會失敗。

在計算列大小時，Amazon Redshift 會在內部將縱線字元 (|) 計算兩次。如果輸入資料包含非常多的縱線字元，即使物件小於 4 MB，資料列大小仍可能超過 4 MB。

COPY \n會以新行字元載入，並以\t標籤字元載入。若要載入反斜線，請加上反斜線 (\\) 來逸出。

COPY 會搜尋格式良好、有效JSON物件或陣列的指定JSON來源。如果在尋找可用JSON結構之前或在有效JSON物件或陣列之間COPY遇到任何非空格字元，COPY會傳回每個執行個體的錯誤。這些錯誤會計入MAXERROR錯誤計數。當錯誤計數等於或超過時MAXERROR，COPY會失敗。

對於每個錯誤，Amazon Redshift 會在 STL_LOAD_ERRORS 系統資料表中記錄資料列。LINE_NUMBER 欄會記錄造成錯誤的JSON物件最後一行。

如果指定 IGNOREHEADER，會COPY忽略JSON資料中指定的行數。JSON 資料中的新行字元一律會計入IGNOREHEADER計算。

COPY 預設會將空字串載入為空欄位。如果指定 EMPTYASNULL，會將 CHAR和 VARCHAR 欄位的空字串COPY載入為 NULL。其他資料類型的空白字串，例如 INT一律會載入 NULL。

不支援下列選項JSON：

- CSV
- DELIMITER
- ESCAPE

- FILLRECORD
- FIXEDWIDTH
- IGNOREBLANKLINES
- NULL AS
- READRATIO
- REMOVEQUOTES

如需詳細資訊，請參閱[COPY 從JSON格式](#)。如需JSON資料結構的詳細資訊，請前往 <https://www.json.org>。

JSONPaths 檔案

如果您從 JSON 格式化或 Avro 來源資料載入，則依預設，會將來源資料中的第一層資料元素 COPY 映射至目標資料表中的資料欄。它會將名稱值對中的每個名稱 (或物件金鑰) 比對目標資料表中的欄名稱來達成此目標。

如果您的資料欄名稱和物件索引鍵不相符，或要映射到資料階層中的更深層，您可以使用 JSONPaths 檔案將 JSON 或 Avro 資料元素明確映射到資料欄。JSONPaths 檔案會比對目標資料表或資料欄清單中的資料欄順序，將 JSON 資料元素映射至資料欄。

JSONPaths 檔案只能包含單一 JSON 物件 (非陣列)。JSON 物件是名稱值對。物件金鑰 (名稱值對中的名稱) 必須是 "jsonpaths"。名稱值對中的值是 JSONPath 表達式的陣列。每個 JSONPath 表達式都參考 JSON 資料階層或 Avro 結構描述中的單一元素，類似於 XPath 表達式如何參考 XML 文件中的元素。如需詳細資訊，請參閱[JSONPath 表達式](#)。

若要使用 JSONPaths 檔案，請將 JSON 或 AVRO 關鍵字新增至 COPY 命令。使用以下格式指定 JSONPaths 檔案的 S3 儲存貯體名稱和物件路徑。

```
COPY tablename
FROM 'data_source'
CREDENTIALS 'credentials-args'
FORMAT AS { AVRO | JSON } 's3://jsonpaths_file';
```

s3://jsonpaths_file 值必須是明確參考單一檔案的 Amazon S3 物件金鑰，例如 's3://amzn-s3-demo-bucket/jsonpaths.txt'。它不能是索引鍵字首。

在某些情況下，如果您是從 Amazon S3 載入，則 jsonpaths_file 指定的檔案具有與 copy_from_s3_objectpath 為資料檔指定的路徑相同的字首。如果是這樣，會將 JSONPaths

檔案COPY讀取為資料檔案並傳回錯誤。例如，假設資料檔案使用物件路徑，`s3://amzn-s3-demo-bucket/my_data.json`且JSONPaths檔案為 `s3://amzn-s3-demo-bucket/my_data.jsonpaths`。在此情況下，會COPY嘗試將 `my_data.jsonpaths` 載入為資料檔案。

如果金鑰名稱是 以外的任何字串"jsonpaths"，則COPY命令不會傳回錯誤，但會忽略 `jsonpaths_file`，而是使用 'auto' 引數。

如果發生以下任何情況，COPY命令會失敗：

- 格式JSON不正確。
- 有一個以上的JSON物件。
- 物件外面存在空格以外的任何字元。
- 陣列元素是空字串或不是字串。

MAXERROR 不適用於 JSONPaths 檔案。

即使已指定 [ENCRYPTED](#) 選項，也不得加密JSONPaths檔案。

如需詳細資訊，請參閱[COPY 從JSON格式](#)。

JSONPath 表達式

JSONPaths 檔案使用JSONPath運算式將資料欄位映射至目標資料欄。每個JSONPath表達式對應至 Amazon Redshift 目標資料表中的一欄。如果使用資料欄清單，則JSONPath陣列元素的順序必須符合目標資料表或資料欄清單中的資料欄順序。

如圖所示，欄位名稱和值都需要雙引號字元。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。

如果在JSON資料中找不到JSONPath運算式所參考的物件元素，會COPY嘗試載入NULL值。如果參考的物件格式不正確，會COPY傳回載入錯誤。

如果在JSON或Avro資料中找不到JSONPath運算式所參考的陣列元素，COPY會失敗並出現下列錯誤：從來源資料中不存在JSONPaths的 `Invalid JSONPath format: Not an array or index out of range`。移除任何陣列元素，並確認來源資料中的陣列是否形成良好。

JSONPath 運算式可以使用括號符號或點符號，但您無法混合符號。下列範例使用括號符號顯示JSONPath表達式。

```
{
  "jsonpaths": [
    "$['venueName']",
    "$['venueCity']",
    "$['venueState']",
    "$['venueSeats']"
  ]
}
```

下列範例顯示使用點符號的JSONPath表達式。

```
{
  "jsonpaths": [
    "$.venueName",
    "$.venueCity",
    "$.venueState",
    "$.venueSeats"
  ]
}
```

在 Amazon Redshift COPY 語法中，JSONPath 表達式必須指定 JSON 或 Avro 階層式資料結構中單一名稱元素的明確路徑。Amazon Redshift 不支援可能解析為不明確路徑或多個名稱 JSONPath 元素的任何元素，例如萬用字元或篩選條件表達式。

如需詳細資訊，請參閱[COPY 從JSON格式](#)。

JSONPaths 搭配 Avro Data 使用

下列範例示範多個層級的 Avro 結構描述。

```
{
  "name": "person",
  "type": "record",
  "fields": [
    {"name": "id", "type": "int"},
    {"name": "guid", "type": "string"},
    {"name": "isActive", "type": "boolean"},
    {"name": "age", "type": "int"},
    {"name": "name", "type": "string"},
    {"name": "address", "type": "string"},
    {"name": "latitude", "type": "double"},
    {"name": "longitude", "type": "double"},
  ]
}
```

```
{
  "name": "tags",
  "type": {
    "type" : "array",
    "name" : "inner_tags",
    "items" : "string"
  }
},
{
  "name": "friends",
  "type": {
    "type" : "array",
    "name" : "inner_friends",
    "items" : {
      "name" : "friends_record",
      "type" : "record",
      "fields" : [
        {"name" : "id", "type" : "int"},
        {"name" : "name", "type" : "string"}
      ]
    }
  }
},
{"name": "randomArrayItem", "type": "string"}
]
```

下列範例顯示使用 AvroPath 運算式參考先前結構描述JSONPaths的檔案。

```
{
  "jsonpaths": [
    "$.id",
    "$.guid",
    "$.address",
    "$.friends[0].id"
  ]
}
```

此JSONPaths範例包含下列元素：

jsonpaths

包含運算式的JSON AvroPath物件名稱。

[...]

括號會括住包含路徑元素的JSON陣列。

\$

貨幣符號表示 Avro 結構描述中的根元素，即 "fields" 陣列。

\$.id",

AvroPath 表達式的目標。在此例子中，目標是 "fields" 陣列中名稱為 "id" 的元素。表達式以逗號分隔。

\$.friends[0].id"

方括號表示陣列索引。JSONPath 運算式使用以零為基礎的索引，因此此運算式參考 "friends" 陣列中名稱為的第一個元素 "id"。

Avro 結構描述語法需要使用內部欄位來定義記錄和陣列資料類型的結構。AvroPath 運算式會忽略內部欄位。例如，欄位 "friends" 定義名為 "inner_friends" 的陣列，此陣列接著定義名為 "friends_record" 的記錄。參考欄位的 AvroPath 表達式 "id" 可以忽略額外的欄位，以直接參考目標欄位。下列 AvroPath 表達式參考屬於 "friends" 陣列的兩個欄位。

```
$.friends[0].id"  
$.friends[0].name"
```

單欄資料格式參數

除了標準資料格式之外，COPY還支援COPY來自 Amazon S3 的下列資料欄資料格式。COPY 具有特定限制，支援資料欄格式。如需詳細資訊，請參閱[COPY 從資料欄資料格式](#)。

ORC

從使用 Optimized Row Columnar (ORC) 檔案格式的檔案中載入資料。

PARQUET

從採用 Parquet 檔案格式的檔案載入資料。

檔案壓縮參數

您可以指定下列參數，以便從壓縮資料檔案載入。

檔案壓縮參數

BZIP2

此值指定一或多個 bzip2 壓縮格式的輸入檔案 (.bz2 檔案)。COPY 操作會讀取每個壓縮檔案，並在資料載入時取消壓縮資料。

GZIP

此值指定一或多個 gzip 壓縮格式的輸入檔案 (.gz 檔案)。COPY 操作會讀取每個壓縮檔案，並在資料載入時取消壓縮資料。

LZOP

此值指定一或多個 lzop 壓縮格式的輸入檔案 (.lzo 檔案)。COPY 操作會讀取每個壓縮檔案，並在資料載入時取消壓縮資料。

Note

COPY 不支援使用 lzop --filter 選項壓縮的檔案。

ZSTD

此值指定一或多個壓縮 Zstandard 格式的輸入檔案 (.zst 檔案)。COPY 操作會讀取每個壓縮檔案，並在資料載入時取消壓縮資料。

Note

ZSTD 僅支援 COPY 來自 Amazon S3 的。

資料轉換參數

載入資料表時，COPY 會嘗試隱含地將來源資料中的字串轉換為目標資料欄的資料類型。如果您需要指定不同於預設行為的轉換，或預設轉換造成錯誤，您可以指定下列參數來管理資料轉換。如需這些參數語法的詳細資訊，請參閱 [COPY 語法](#)。

- [ACCEPTANYDATE](#)
- [ACCEPTINVCHARS](#)
- [BLANKSASNULL](#)

- [DATEFORMAT](#)
- [EMPTYASNULL](#)
- [ENCODING](#)
- [ESCAPE](#)
- [EXPLICIT_IDS](#)
- [FILLRECORD](#)
- [IGNOREBLANKLINES](#)
- [IGNOREHEADER](#)
- [NULL AS](#)
- [REMOVEQUOTES](#)
- [ROUNDEC](#)
- [TIMEFORMAT](#)
- [TRIMBLANKS](#)
- [TRUNCATECOLUMNS](#)

資料轉換參數

ACCEPTANYDATE

允許載入任何日期格式而不會產生錯誤，包括無效格式，例如 `00/00/00 00:00:00`。此參數僅適用於 `TIMESTAMP` 和 `DATE` 資料欄。一律 `ACCEPTANYDATE` 搭配 `DATEFORMAT` 參數使用。如果資料的日期格式與 `DATEFORMAT` 規格不符，Amazon Redshift 會將 `NULL` 值插入該欄位中。

ACCEPTINVCHARS **【AS】** **【'replacement_char'】**

即使資料包含無效 UTF-8 個字元，也能將資料載入資料 `VARCHAR` 欄。指定 `ACCEPTINVCHARS` 時，會將每個無效 UTF-8 字元 `COPY` 取代為相等長度的字串，其中包含 `replacement_char` 指定的字元。例如，假設替換字元是 '^'，則會以 '^'^'^' 取代無效的三位元組字元。

取代字元可以是除 `NULL` 以外的任何 ASCII 字元。預設值為問號 (?)。如需有關無效 UTF-8 個字元的資訊，請參閱 [多位元組字元載入錯誤](#)。

`COPY` 傳回包含無效 UTF-8 個字元的列數，並為每個受影響的列將項目新增至 [STL_REPLACEMENTS](#) 系統資料表，每個節點層最多 100 個列。其他無效 UTF-8 個字元也會被取代，但這些取代事件不會記錄下來。

如果 ACCEPTINVCHARS 未指定，則每當遇到無效的 UTF-8 字元時，會 COPY 傳回錯誤。

ACCEPTINVCHARS 僅適用於 VARCHAR 欄。

BLANKSASNULL

載入空白欄位，僅包含空格字元，如 NULL。此選項僅適用於 CHAR 和 VARCHAR 資料欄。其他資料類型的空白欄位，例如 INT 一律會載入 NULL。例如，包含連續三個空格字元（且沒有其他字元）的字串會載入為 NULL。預設行為（不使用此選項時）是依原狀載入空白字元。

DATEFORMAT 【AS】 { 'dateformat_string' | 'auto' }

如果未指定 DATEFORMAT 任何，則預設格式為 'YYYY-MM-DD'。例如，另一種有效格式為 'MM-DD-YYYY'。

如果 COPY 命令無法辨識日期或時間值的格式，或者您的日期或時間值使用不同的格式，請使用 'auto' 引數搭配 DATEFORMAT 或 TIMEFORMAT 參數。'auto' 引數會識別使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援的數種格式。'auto' 關鍵字區分大小寫。如需詳細資訊，請參閱 [搭配 DATEFORMAT 和使用自動辨識 TIMEFORMAT](#)。

日期格式可以包含時間資訊（時、分、秒），但此資訊會遭到忽略。AS 關鍵字為選用。如需詳細資訊，請參閱 [DATEFORMAT 和 TIMEFORMAT 字串](#)。

EMPTYASNULL

表示 Amazon Redshift 應該將空 CHAR/VARCHAR 欄位載入為 NULL。其他資料類型的空白欄位，例如 INT 一律會載入 NULL。當資料包含連續兩個分隔符號，且分隔符號之間沒有字元時，就形成空欄位。EMPTYASNULL 和 NULL AS "（空字串）會產生相同的行為。

ENCODING 【AS】 file_encoding

指定載入資料的編碼類型。COPY 命令會在載入期間將指定編碼中的資料轉換為 UTF-8。

file_encoding 的有效值如下所示：

- UTF8
- UTF16
- UTF16LE
- UTF16BE
- ISO88591

預設值為 UTF8。

來源檔案名稱必須使用 UTF-8 編碼。

即使為載入資料指定了不同的編碼，下列檔案必須使用 UTF-8 編碼：

- 資訊清單檔案
- JSONPaths 檔案

搭配下列參數提供的引數字串必須使用 UTF-8：

- FIXEDWIDTH 'fixedwidth_spec '
- ACCEPTINVCHARS 'replacement_char '
- DATEFORMAT 'dateformat_string '
- TIMEFORMAT 'timeformat_string '
- NULL AS 'null_string '

固定寬度資料檔案必須使用 UTF-8 編碼。欄位寬度是根據字元數，而不是位元組數。

所有載入資料必須使用指定的編碼。如果 COPY 遇到不同的編碼，它會略過 檔案並傳回錯誤。

如果您指定 UTF16，則資料必須具有位元組順序標記（BOM）。如果您知道您的 UTF-16 資料是小端點（LE）還是大端點（BE），則無論是否存在 UTF16BE，都可以使用 UTF16LE 或 BOM。

若要使用 ISO-8859-1 編碼，請指定 ISO88591。如需詳細資訊，請參閱 Wikipedia 中的 [ISO/IEC 8859-1](#)。

ESCAPE

指定此參數時，輸入資料中的反斜線字元 (\) 就視為逸出字元。反斜線字元後面緊接的字元即使通常做為特殊用途，一樣會載入資料表中成為目前欄值的一部分。例如，當分隔符號字元、引號、內嵌的換行符號字元或逸出字元本身是欄值的正當部分時，您可以使用此參數來逸出這些字元。

如果您將 ESCAPE 參數與 REMOVEQUOTES 參數結合指定，您可以逸出並保留可能移除的引號（' 或 "）。預設 null 字串 (\N) 不受影響，但也可以在輸入資料中以 \\N 逸出。只要您不指定具有 NULL AS 參數的替代 null 字串，\N 並 \\N 產生相同的結果。

Note

控制字元 0x00 (NUL) 無法逸出，應該從輸入資料中移除或轉換。此字元被視為記錄結尾 (EOR) 標記，導致記錄的其餘部分被截斷。

您無法將 ESCAPE 參數用於 FIXEDWIDTH 載入，也無法指定逸出字元本身；逸出字元一律為反斜線字元。您還必須確定輸入資料包含的逸出字元是在適當的位置。

以下是指定 ESCAPE 參數時輸入資料和產生的載入資料的一些範例。第 4 列的結果假設參數也 REMOVEQUOTES 已指定。輸入資料由縱線分隔的兩個欄位組成：

```
1|The quick brown fox\[newline]
jumped over the lazy dog.
2| A\\B\\C
3| A \| B \| C
4| 'A Midsummer Night\'s Dream'
```

載入第 2 欄的資料如下所示：

```
The quick brown fox
jumped over the lazy dog.
A\\B\\C
A|B|C
A Midsummer Night's Dream
```

Note

使用者必須負責將逸出字元套用至載入的輸入資料。此需求的一個例外是，當您重新載入先前使用 ESCAPE 參數卸載的資料時。在此情況下，資料一定包含必要的逸出字元。

ESCAPE 參數不會解譯八進位、十六進位、Unicode 或其他逸出序列符號。例如，如果您的來源資料包含八進位換行值（\012），而且您嘗試使用 ESCAPE 參數載入此資料，Amazon Redshift 會將該值載入資料表 012，而不會將此值解譯為逸出的換行換行值。

在源自 Microsoft Windows 平台的資料中，若要逸出新行字元，您可能需要使用兩個逸出字元：一個用於換行字元，另一個用於新行字元。或者，您可以在載入檔案之前移除換行字元（例如，使用 dos2unix 公用程式）。

EXPLICIT_IDS

如果您想要使用資料表來源資料檔案中的明確值覆寫自動產生的值，請將 EXPLICIT_IDS 與具有 IDENTITY 資料欄的資料表搭配使用。如果命令包含資料欄清單，則該清單必須包含資料 IDENTITY 欄才能使用此參數。EXPLICIT_IDS 值的資料格式必須符合 CREATETABLE 定義指定的 IDENTITY 格式。

當您使用 EXPLICIT_IDS 選項對資料表執行 COPY 命令時，Amazon Redshift 不會檢查資料表中 IDENTITY 資料欄的唯一性。

如果使用 GENERATED BY DEFAULT AS 定義資料欄 IDENTITY，則可以複製該資料欄。這是利用您提供的值來產生或更新。不需要 EXPLICIT_IDS 選項。COPY 不會更新身分高浮水印。

如需使用 EXPLICIT_ 的 COPY 命令範例 IDS，請參閱 [VENUE 載入資料 IDENTITY 欄的明確值](#)。

FILLRECORD

當某些記錄的結尾缺少連續的欄時，允許載入資料檔案。遺失的資料欄會載入為 NULLs。對於文字和 CSV 格式，如果缺少的資料欄是 VARCHAR 資料欄，則會載入零長度字串，而不是 NULLs。若要從文字和 NULLs 載入資料 VARCHAR 欄 CSV，請指定 EMPTYASNULL 關鍵字。NULL 取代只有在資料欄定義允許時才有效 NULLs。

例如，如果資料表定義包含四個 nullable CHAR 資料欄，且記錄包含值 apple, orange, banana, mango，則 COPY 命令可以載入並填入僅包含值的記錄 apple, orange。遺失 CHAR 的值會載入為 NULL 值。

IGNOREBLANKLINES

忽略資料檔案中只含有換行字元的空白行，而不嘗試載入這些空白行。

IGNOREHEADER 【AS】 number_rows

將指定的 number_rows 視為檔案標頭而不載入。使用 IGNOREHEADER 略過平行載入中所有檔案中的檔案標頭。

NULL AS 'null_string'

將符合 null_string 的欄位載入為 NULL，其中 null_string 可以是任何字串。如果您的資料包含 null 終止器，也稱為 NUL (UTF-8 0000) 或二進位零 (0x000)，會 COPY 將其視為任何其他字元。例如，包含 '1' || NUL || '2' 的記錄會複製為長度為 3 位元組的字串。如果欄位僅包含 NUL，您可以使用 NULL AS 指定 '\0' 或 '\000'，例如 NULL AS '\0' 或 NULL AS '\000'。如果欄位包含結尾為 NUL 的字串，且已指定 NULL AS，則會 NUL 在結尾插入結尾為 NUL 的字串。請勿在 null_string 值使用 '\n' (換行符號)。Amazon Redshift 保留 '\n' 用作行分隔符號。預設 null_string 為 '\N'。

Note

如果您嘗試將 null 載入定義為 NOT 的欄中 NULL，COPY 命令將會失敗。

REMOVEQUOTES

在傳入的資料中移除括住字串的引號。引號內的所有字元 (包括分隔符號) 都會保留。如果字串具有起始的單引號或雙引號，但沒有對應的結束標記，則COPY命令無法載入該列並傳回錯誤。下表以一些簡單的範例示範含有引號的字串及結果載入的值。

輸入字串	使用 REMOVEQUOTES 選項載入的值
"分隔符號是縱線 () 字元"	分隔符號是縱線 () 字元
'黑色'	黑色
"白色"	白色
藍色'	藍色'
藍色'	不會載入值：錯誤情況
"藍色	不會載入值：錯誤情況
'''黑色'''	''黑色''
''	<white space>

ROUNDEC

當輸入值的小數位數超過欄的小數位數時，將數值四捨五入。根據預設，COPY會在必要時截斷值以符合資料欄的規模。例如，如果20.259將值載入DECIMAL(8,2)資料欄，20.25預設會將值截斷為COPY。如果指定ROUNDEC，會將值COPY四捨五入為20.26。INSERT命令一律會在必要時四捨五入值，以符合資料欄的規模，因此具有ROUNDEC參數的COPY命令的行為與INSERT命令相同。

TIMEFORMAT 【AS】 {'timeformat_string' | 'auto' | 'epochsecs' | 'epochmillisecs' }

指定時間格式。如果未指定TIMEFORMAT任何，則預設格式為YYYY-MM-DD HH:MI:SS資料TIMESTAMP欄或YYYY-MM-DD HH:MI:SSOF資料TIMESTAMPTZ欄，其中OF是國際標準時間(UTC)的偏移。timeformat_string中不可包含時區指標。若要載入與預設格式不同的TIMESTAMPTZ資料，請指定「自動」；如需詳細資訊，請參閱[搭配DATEFORMAT和使用自動辨識TIMEFORMAT](#)。如需timeformat_string的相關資訊，請參閱[DATEFORMAT和TIMEFORMAT字串](#)。

'auto' 引數會識別使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援的數種格式。如果 COPY 命令無法辨識日期或時間值的格式，或者您的日期和時間值使用彼此不同的格式，請使用 'auto' 引數搭配 DATEFORMAT 或 TIMEFORMAT 參數。如需詳細資訊，請參閱[搭配 DATEFORMAT 和 使用自動辨識 TIMEFORMAT](#)。

如果您的來源資料表示為 epoch 時間，則表示自 1970 年 1 月 1 日起的秒數或毫秒數，請 UTC 指定 'epochsecs' 或 'epochmillisecs'。

'auto'、'epochsecs' 和 'epochmillisecs' 關鍵字區分大小寫。

AS 關鍵字為選用。

TRIMBLANKS

從 VARCHAR 字串中移除尾端空白字元。此參數僅適用於具有 VARCHAR 資料類型的資料欄。

TRUNCATECOLUMNS

將欄的資料截斷為適當的字元數，以符合欄規格。僅適用於 VARCHAR 或 CHAR 資料類型的資料欄，以及大小為 4 MB 或以下的資料列。

資料載入操作

指定下列參數來管理載入操作的預設行為，以進行故障排除或縮短載入時間。

- [COMPROWS](#)
- [COMPUPDATE](#)
- [IGNOREALLERRORS](#)
- [MAXERROR](#)
- [NOLOAD](#)
- [STATUPDATE](#)

參數

COMPROWS 數量

指定列數做為壓縮分析的樣本大小。分析是以每個資料配量中的列為對象。例如，假設您指定 COMPROWS 1000000 (1,000,000)，而系統總共包含四個配量，則每個配量最多讀取和分析 250,000 列。

如果COMPROWS未指定，則每個切片的樣本數預設為 100,000。COMPROWS 小於每個切片預設值 100,000 列的值會自動升級至預設值。但是，如果載入的資料量不足以構成有意義的樣本，則不會執行自動壓縮。

如果COMPROWS數字大於輸入檔案中的資料列數目，COPY命令仍會繼續並對所有可用的資料列執行壓縮分析。此引數可接受的範圍是介於 1000 到 2147483647 (2,147,483,647) 之間的數字。

COMPUPDATE 【 PRESET | { ON | TRUE } | { OFF | FALSE } 】,

控制是否在 期間自動套用壓縮編碼COPY。

當 COMPUPDATE為 時PRESET，如果目標資料表為空，COPY則命令會為每個資料欄選擇壓縮編碼；即使資料欄已有 以外的編碼RAW。可以取代目前指定的資料欄編碼。每個資料欄的編碼是以資料欄的資料類型為基礎。不會對任何資料取樣。Amazon Redshift 會自動指派壓縮編碼，如下所示：

- 定義為排序索引鍵的資料欄會被指派RAW壓縮。
- 定義為 BOOLEAN、 REAL或 DOUBLE PRECISION 資料類型的資料欄會被指派RAW壓縮。
- 定義為 SMALLINT、 INTEGER、 、 BIGINTDECIMAL、 TIMESTAMP、 DATE或 的資料欄 TIMESTAMPTZ會被指派AZ64壓縮。
- 定義為 CHAR或 VARCHAR 且被指派LZO壓縮的資料欄。

COMPUPDATE 省略 時，只有在目標資料表為空，且您尚未為任何資料欄指定編碼（ 除外RAW）時，COPY命令才會選擇每個資料欄的壓縮編碼。每個欄的編碼是由 Amazon Redshift 決定。不會對任何資料取樣。

當 COMPUPDATE 為 ON（ 或 TRUE ）或 COMPUPDATE 時，如果沒有 選項，COPY則命令會套用自動壓縮，如果資料表為空；即使資料表資料欄已有 以外的編碼RAW。可以取代目前指定的資料欄編碼。每個資料欄的編碼會根據取樣資料的分析。如需詳細資訊，請參閱[利用自動壓縮載入資料表](#)。

當 COMPUPDATE為 OFF（ 或 FALSE ）時，會停用自動壓縮。不會變更資料欄編碼。

如需關於分析壓縮的系統資料表資訊，請參閱 [STL_ANALYZE_COMPRESSION](#)。

IGNOREALLERRORS

您可以指定此選項來忽略載入作業期間發生的所有錯誤。

如果您指定 IGNOREALLERRORS選項，則無法指定 MAXERROR選項。您無法指定欄式格式 IGNOREALLERRORS的選項，包括 ORC和 Parquet。

MAXERROR 【AS】 error_count

如果載入傳回 error_count 個或更多錯誤，載入會失敗。如果載入傳回的錯誤較少，則會繼續並傳回INFO訊息，指出無法載入的資料列數。當某些列因為格式錯誤或資料有其他不一致情形而無法載入到資料表時，使用此參數可讓載入繼續進行。

如果要讓載入在發生第一個錯誤時立即失敗，請將此值設為 0 或 1。AS 關鍵字為選用。MAXERROR 預設值為 0，限制為 100000。

由於 Amazon Redshift 的平行性質MAXERROR，實際報告的錯誤數目可能大於指定的。如果 Amazon Redshift 叢集中的任何節點偵測到MAXERROR已超過，則每個節點都會報告其遇到的所有錯誤。

NOLOAD

檢查資料檔案的有效性，而不實際載入資料。執行實際資料載入之前，請使用 NOLOAD 參數來確定資料檔案載入時沒有任何錯誤。COPY 使用 NOLOAD 參數執行比載入資料快得多，因為它只會剖析檔案。

STATUPDATE 【{ ON | TRUE } | { OFF | FALSE }】

在成功COPY命令結束時，管理最佳化工具統計資料的自動運算和重新整理。根據預設，如果未使用 STATUPDATE 參數，則在資料表最初為空白時，會自動更新統計資料。

每當將資料擷取到非空白資料表時，我們建議透過執行 [ANALYZE](#) 命令或使用 STATUPDATE ON 引數來更新統計資料。

STATUPDATE 開啟（或 TRUE）時，統計資料會自動更新，無論資料表初始是否為空白。STATUPDATE 如果使用，目前使用者必須是資料表擁有者或超級使用者。如果 STATUPDATE未指定，則只需要INSERT許可。

透過 STATUPDATEOFF（或 FALSE），統計資料永遠不會更新。

如需其他資訊，請參閱 [分析資料表](#)。

依字母排序的參數清單

下列清單提供每個COPY命令參數描述的連結，依字母順序排序。

- [ACCEPTANYDATE](#)
- [ACCEPTINVCHARS](#)

- [ACCESS_KEY_ID and SECRET_ACCESS_KEY](#)
- [AVRO](#)
- [BLANKSASNULL](#)
- [BZIP2](#)
- [COMPROWS](#)
- [COMPUPDATE](#)
- [CREDENTIALS](#)
- [CSV](#)
- [DATEFORMAT](#)
- [DELIMITER](#)
- [EMPTYASNULL](#)
- [ENCODING](#)
- [ENCRYPTED](#)
- [ESCAPE](#)
- [EXPLICIT_IDS](#)
- [FILLRECORD](#)
- [FIXEDWIDTH](#)
- [FORMAT](#)
- [FROM](#)
- [GZIP](#)
- [IAM_ROLE](#)
- [IGNOREALLERRORS](#)
- [IGNOREBLANKLINES](#)
- [IGNOREHEADER](#)
- [JSON](#)
- [LZOP](#)
- [MANIFEST](#)
- [MASTER_SYMMETRIC_KEY](#)
- [MAXERROR](#)

- [NOLOAD](#)
- [NULL AS](#)
- [READRATIO](#)
- [REGION](#)
- [REMOVEQUOTES](#)
- [ROUNDEC](#)
- [SESSION_TOKEN](#)
- [SHAPEFILE](#)
- [SSH](#)
- [STATUPDATE](#)
- [TIMEFORMAT](#)
- [SESSION_TOKEN](#)
- [TRIMBLANKS](#)
- [TRUNCATECOLUMNS](#)
- [ZSTD](#)

使用須知

主題

- [存取其他 AWS 資源的許可](#)
- [COPY 搭配 Amazon S3 存取點別名使用](#)
- [從 Amazon S3 載入多位元組資料](#)
- [載入 GEOMETRY 或 GEOGRAPHY 資料類型的資料欄](#)
- [載入 HLLSKETCH 資料類型](#)
- [載入 VARBYTE 資料類型的資料欄](#)
- [讀取多個檔案時發生錯誤](#)
- [COPY 從 JSON 格式](#)
- [COPY 從資料欄資料格式](#)
- [DATEFORMAT 和 TIMEFORMAT 字串](#)

• [搭配 DATEFORMAT 和 使用自動辨識 TIMEFORMAT](#)

存取其他 AWS 資源的許可

若要在叢集與其他 AWS 資源之間移動資料，例如 Amazon S3、Amazon DynamoDB EMR、Amazon 或 Amazon EC2，您的叢集必須具有存取資源和執行必要動作的許可。例如，若要從 Amazon S3 載入資料，COPY 必須具有儲存貯體的LIST存取權，以及儲存貯體物件的GET存取權。如需最低許可的相關資訊，請參閱 [IAM COPY、UNLOAD 和 的許可 CREATE LIBRARY](#)。

叢集必須經過驗證，才能獲得授權來存取資源。您可以選擇下列任一種身分驗證方法：

- [角色類型存取控制](#) – 對於角色型存取控制，您可以指定叢集用於身分驗證和授權的 AWS Identity and Access Management (IAM) 角色。為了保護您的 AWS 憑證和敏感資料，我們強烈建議使用角色型身分驗證。
- [金鑰型存取控制](#) – 對於金鑰型存取控制，您可以純文字形式提供使用者的 AWS 存取憑證（存取金鑰 ID 和秘密存取金鑰）。

角色類型存取控制

透過角色型存取控制，您的叢集會暫時代表您擔任IAM角色。然後，根據角色獲予的授權，叢集即可存取所需的 AWS 資源。

建立IAM角色類似於將許可授予使用者，因為它是 AWS 具有許可政策的身分，用於決定身分在 中可以和不可以執行的動作 AWS。但是，角色可以由任何需要此角色的實體來擔任，而不必與一個使用者產生獨特的關聯。此外，角色沒有任何相關聯的登入資料 (密碼或存取金鑰)。反之，如果角色與叢集相關聯，則會動態建立存取金鑰並將該金鑰提供給叢集。

我們建議您使用角色型存取控制，因為它除了保護您的 AWS 憑證之外，還提供對 AWS 資源和敏感使用者資料的更安全、更精細的存取控制。

角色型身分驗證提供下列優點：

- 您可以使用 AWS 標準IAM工具來定義IAM角色，並將角色與多個叢集建立關聯。修改角色的存取政策時，會將變更自動套用至所有使用該角色的叢集。
- 您可以定義精細IAM政策，授予特定叢集和資料庫使用者存取特定 AWS 資源和動作的許可。
- 您的叢集會在執行時間取得臨時工作階段登入資料，並視需要重新整理登入資料，直到操作完成。如果您使用金鑰型臨時登入資料，萬一臨時登入資料在操作完成之前就過期，則操作會失敗。
- 您的存取金鑰 ID 和秘密存取金鑰 ID 不會儲存在SQL程式碼中或傳輸。

若要使用角色型存取控制，您必須先使用 Amazon Redshift 服務IAM角色類型建立角色，然後將角色連接至叢集。角色至少必須具備 [IAM COPY](#)、[UNLOAD](#)和 [的許可 CREATE LIBRARY](#)中列出的許可。如需建立IAM角色並將其連接至叢集的步驟，請參閱 [Amazon Redshift 管理指南 中的授權 Amazon Redshift 代表您存取 AWS 其他服務](#)。

您可以使用 Amazon Redshift 管理主控台、或將角色新增至叢集CLI，或檢視與叢集相關聯的角色API。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的[將IAM角色與叢集建立關聯](#)。

當您建立IAM角色時，會IAM傳回角色的 Amazon Resource Name (ARN)。若要指定IAM角色，ARN請將 [IAM_ROLE](#) 參數或 [CREDENTIALS](#) 參數提供給角色。

例如，假設下列角色已附加至叢集。

```
"IamRoleArn": "arn:aws:iam::0123456789012:role/MyRedshiftRole"
```

下列COPY命令範例使用 IAM_ROLE 參數搭配上ARN一個範例中的，以進行 Amazon S3 的身分驗證和存取。

```
copy customer from 's3://amzn-s3-demo-bucket/mydata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

下列COPY命令範例使用 CREDENTIALS 參數來指定IAM角色。

```
copy customer from 's3://amzn-s3-demo-bucket/mydata'
credentials
'aws_iam_role=arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

此外，超級使用者可以將 ASSUMEROLE 權限授予資料庫使用者和群組，以提供COPY操作角色的存取權。如需相關資訊，請參閱 [GRANT](#)。

金鑰型存取控制

使用金鑰型存取控制，您可以為IAM有權存取包含資料之 AWS 資源的使用者提供存取金鑰 ID 和秘密存取金鑰。您可以將 [ACCESS_KEY_ID and SECRET_ACCESS_KEY](#) 參數一起使用，或使用 [CREDENTIALS](#) 參數。

Note

我們強烈建議使用IAM角色進行身分驗證，而不是提供純文字存取金鑰 ID 和秘密存取金鑰。如果您選擇以金鑰為基礎的存取控制，切勿使用 AWS 您的帳戶（根）憑證。一律建立IAM使用

者，並提供該使用者的存取金鑰 ID 和秘密存取金鑰。如需建立IAM使用者的步驟，請參閱在 [AWS 帳戶中建立IAM使用者](#)。

若要使用 ACCESS_KEY_ID 和 SECRET_ACCESS_ 進行身分驗證KEY，請取代 `<access-key-id>` 以及 `<secret-access-key>` 授權使用者的存取金鑰 ID 和完整秘密存取金鑰，如下所示。

```
ACCESS_KEY_ID '<access-key-id>'
SECRET_ACCESS_KEY '<secret-access-key>';
```

若要使用 CREDENTIALS 參數進行身分驗證，請取代 `<access-key-id>` 以及 `<secret-access-key>` 授權使用者的存取金鑰 ID 和完整秘密存取金鑰，如下所示。

```
CREDENTIALS
'aws_access_key_id=<access-key-id>;aws_secret_access_key=<secret-access-key>';
```

IAM 使用者至少必須擁有 中列出的許可 [IAM COPY、UNLOAD](#) 和 的許可 [CREATE LIBRARY](#)。

暫時安全憑證

如果使用金鑰型存取控制，您可以利用臨時安全性登入資料，以進一步限制使用者有何權限存取您的資料。角色型身分驗證會自動使用臨時登入資料。

Note

強烈建議使用 [role-based access control](#)，而不要建立臨時登入資料並以純文字提供存取金鑰 ID 和私密存取金鑰。角色型存取控制會自動使用暫時憑證。

暫時安全憑證提供加強的安全性，因為有效期限較短，且過期之後不能重複使用。與字符一起產生的存取金鑰 ID 和私密存取金鑰一定要有字符才能使用，而具有這些暫時安全憑證的使用者只能在憑證到期之前存取您的資源。

若要授予使用者暫時存取 資源的權限，您可以呼叫 AWS Security Token Service (AWS STS) API 操作。操作會 AWS STS API傳回暫時性安全憑證，其中包含安全權杖、存取金鑰 ID 和秘密存取金鑰。如果某使用者需要暫時存取您的資源，您可以將臨時安全性登入資料發行給該使用者。這些使用者可以是現有IAM使用者，也可以是非AWS使用者。如需建立臨時安全憑證的詳細資訊，請參閱 IAM 使用者指南中的 [使用臨時安全憑證](#)。

您可以將 [ACCESS_KEY_ID](#) and [SECRET_ACCESS_KEY](#) 參數與 [SESSION_TOKEN](#) 參數或 [CREDENTIALS](#) 參數一起使用。您也必須提供字符隨附的存取金鑰 ID 和私密存取金鑰。

若要使用 `ACCESS_KEY_ID`、`SECRET_ACCESS_KEY` 和 `SESSION_` 驗證 `TOKEN`，請取代 `<temporary-access-key-id>`、`<temporary-secret-access-key>` 和 `<temporary-token>` 如下所示。

```
ACCESS_KEY_ID '<temporary-access-key-id>'
SECRET_ACCESS_KEY '<temporary-secret-access-key>'
SESSION_TOKEN '<temporary-token>';
```

若要使用 驗證 `CREDENTIALS`，請在憑證字串 `session_token=<temporary-token>` 中包含，如下所示。

```
CREDENTIALS
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-access-key>;session_token=<temporary-token>';
```

下列範例顯示具有臨時安全憑證的 `COPY` 命令。

```
copy table-name
from 's3://objectpath'
access_key_id '<temporary-access-key-id>'
secret_access_key '<temporary-secret-access-key>'
session_token '<temporary-token>';
```

下列範例會使用臨時憑證和檔案加密載入 `LISTING` 資料表。

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listings_pipe.txt'
access_key_id '<temporary-access-key-id>'
secret_access_key '<temporary-secret-access-key>'
session_token '<temporary-token>'
master_symmetric_key '<root-key>'
encrypted;
```

下列範例使用具有臨時憑證和檔案加密的 `CREDENTIALS` 參數載入 `LISTING` 資料表。

```
copy listing
from 's3://amzn-s3-demo-bucket/data/listings_pipe.txt'
```

```
credentials
'aws_access_key_id=<temporary-access-key-id>;aws_secret_access_key=<temporary-secret-
access-key>;session_token=<temporary-token>;master_symmetric_key=<root-key>'
encrypted;
```

Important

暫時安全憑證在整個 COPY 或 UNLOAD 操作期間都必須有效。如果臨時安全性登入資料在操作期間過期，則命令會失敗，而交易會回復。例如，如果臨時安全憑證在 15 分鐘後過期，且 COPY 操作需要一小時，COPY 操作會在完成之前失敗。如果您使用角色型存取，臨時安全性登入資料會自動重新整理，直到操作完成。

IAM COPY、UNLOAD 和 的許可 CREATE LIBRARY

CREDENTIALS 參數參考 IAM 的角色或使用者必須至少具有下列許可：

- 對於 COPY 來自 Amazon S3 LIST 的，如果使用 Amazon S3 儲存貯體和正在載入的 GET Amazon S3 物件，以及資訊清單檔案的許可。
- 對於 COPY 具有 JSON 格式化資料的 Amazon S3 EMR、Amazon 和遠端主機（SSH），如果使用 Amazon S3 上的 LIST 和 GET JSONPaths 檔案的許可。
- COPY 從 DynamoDB 對的許可，SCAN 以及正在載入的 DESCRIBE DynamoDB 資料表。
- 對於 COPY 來自 Amazon EMR 叢集的，Amazon EMR 叢集上 ListInstances 動作的許可。
- 將資料檔案卸載至 Amazon S3 儲存貯體的 UNLOAD Amazon S3 LIST、GET 和 PUT 許可。
- 從 CREATE LIBRARY Amazon S3 取得的 Amazon S3 LIST 儲存貯體和匯入 GET 之 Amazon S3 物件的許可。

Note

如果您收到錯誤訊息 `S3ServiceException: Access Denied`，當您執行 COPY、UNLOAD 或 CREATE LIBRARY 命令時，您的叢集沒有 Amazon S3 的適當存取權限。

您可以透過將 IAM 政策連接至叢集、使用者或使用者所屬群組 IAM 的角色來管理 IAM 許可。例如，`AmazonS3ReadOnlyAccess` 受管政策會授予 LIST Amazon S3 資源和 GET 許可。如需 IAM 政策的詳細資訊，請參閱 IAM 使用者指南 中的 [管理 IAM 政策](#)。

COPY 搭配 Amazon S3 存取點別名使用

COPY 支援 Amazon S3 存取點別名。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的[為您的存取點使用儲存貯體型別名](#)。

從 Amazon S3 載入多位元組資料

如果您的資料包含非ASCII多位元組字元（例如中文或西里爾文字元），您必須將資料載入VARCHAR資料欄。VARCHAR資料類型支援4位元組UTF-8個字元，但CHAR資料類型只接受單位元組ASCII字元。您無法將五位元組或更長的字元載入Amazon Redshift資料表。如需詳細資訊，請參閱[多位元組字元](#)。

載入 GEOMETRY或 GEOGRAPHY 資料類型的資料欄

您可以從字元分隔文字檔案中的資料COPY到GEOMETRY或GEOGRAPHY資料欄，例如CSV檔案。資料必須是已知二進位格式（WKB或EWKB）或已知文字格式（WKT或EWKT）的十六進位形式，且符合COPY命令的單一輸入列大小上限。如需詳細資訊，請參閱[COPY](#)。

如需有關如何從Shapefile中載入的資訊，請參閱[將 Shapefile 載入 Amazon Redshift](#)。

如需GEOMETRY或GEOGRAPHY資料類型的相關資訊，請參閱[在 Amazon Redshift 中查詢空間資料](#)。

載入HLLSKETCH資料類型

您只能以Amazon Redshift支援的稀疏或密集格式複製HLL草圖。若要在HyperLogLog草圖上使用COPY命令，請使用Base64格式進行密集草HyperLogLog圖，並使用JSON格式進行稀疏HyperLogLog草圖。如需詳細資訊，請參閱[HyperLogLog 函數](#)。

下列範例使用CREATETABLE和將資料從CSV檔案匯入資料表COPY。首先，範例t1會使用CREATE建立資料表TABLE。

```
CREATE TABLE t1 (sketch hllsketch, a bigint);
```

然後，它會使用COPY將資料從CSV檔案匯入資料表t1。

```
COPY t1 FROM s3://amzn-s3-demo-bucket/unload/' IAM_ROLE  
'arn:aws:iam::0123456789012:role/MyRedshiftRole' NULL AS 'null' CSV;
```

載入VARBYTE資料類型的資料欄

您可以從 CSV、Parquet 和 ORC 格式的檔案載入資料。對於 CSV，資料會從以十六進位表示的檔案中載入VARBYTE。您無法使用 FIXEDWIDTH選項載入VARBYTE資料。COPY 不支援 ADDQUOTES或 REMOVEQUOTES選項。資料VARBYTE欄無法用作分割區資料欄。

讀取多個檔案時發生錯誤

COPY 命令是原子和交易的。換句話說，即使COPY命令從多個檔案讀取資料，整個程序也會被視為單一交易。如果讀取檔案COPY時發生錯誤，它會自動重試，直到程序逾時（請參閱 [statement timeout](#)）或長時間無法從 Amazon S3 下載資料（介於 15 到 30 分鐘之間），確保每個檔案僅載入一次。如果COPY命令失敗，整個交易都會取消，且所有變更都會復原。如需如何處理載入錯誤的相關資訊，請參閱[針對資料載入進行故障診斷](#)。

成功啟動COPY命令後，如果工作階段終止，例如用戶端中斷連線，則不會失敗。不過，如果COPY命令位於 ... BEGIN END交易區塊中，但由於工作階段終止而未完成，則整個交易，包括 COPY，都會復原。如需交易的相關資訊，請參閱 [BEGIN](#)。

COPY 從JSON格式

JSON 資料結構由一組物件或陣列組成。JSON 物件以 braces 開頭和結尾，並包含未排序的名稱值對集合。每一個名稱和值以冒號分隔，而配對以逗號分隔。名稱是以雙引號括住的字串。引號字元必須是簡單引號 (0x22)，而不是斜向或「智慧型」引號。

JSON 陣列以括號開頭和結尾，並包含以逗號分隔的有序值集合。值可以是雙引號中的字串、數字、布林值為 true 或 false、null、JSON物件或陣列。

JSON 物件和陣列可以巢狀排列，啟用階層式資料結構。下列範例顯示具有兩個有效物件JSON的資料結構。

```
{
  "id": 1006410,
  "title": "Amazon Redshift Database Developer Guide"
}
{
  "id": 100540,
  "name": "Amazon Simple Storage Service User Guide"
}
```

以下顯示與兩個JSON陣列相同的資料。

```
[
  1006410,
  "Amazon Redshift Database Developer Guide"
]
[
  100540,
  "Amazon Simple Storage Service User Guide"
]
```

COPY 的選項 JSON

您可以在COPY搭配使用 JSON 格式資料時指定下列選項：

- 'auto' – COPY會自動從 JSON 檔案載入欄位。
- 'auto ignorecase' – COPY會自動從JSON檔案載入欄位，同時忽略欄位名稱的案例。
- s3://jsonpaths_file – COPY使用 JSONPaths 檔案剖析JSON來源資料。JSONPaths 檔案是文字檔案，其中包含名稱與JSONPath表達式陣列"jsonpaths"配對的單一JSON物件。如果名稱是以外的任何字串"jsonpaths"，COPY會使用 'auto' 引數，而不是 檔案JSONPaths。

如需示範如何使用 'auto'、'auto ignorecase' 或 JSONPaths 檔案以及使用 JSON 物件或陣列載入資料的範例，請參閱 [從JSON範例複製](#)。

JSONPath 選項

在 Amazon Redshift COPY 語法中，JSONPath表達式會使用括號符號或點符號，指定JSON階層式資料結構中單一名稱元素的明確路徑。Amazon Redshift 不支援可能解析為不明確路徑或多個名稱JSONPath元素的任何元素，例如萬用字元或篩選條件表達式。因此，Amazon Redshift 無法剖析複雜、多層級的資料結構。

以下是使用括號表示法，具有JSONPath表達式JSONPaths的檔案範例。貨幣符號 (\$) 代表根層級結構。

```
{
  "jsonpaths": [
    "$['id']",
    "$['store']['book']['title']",
    "$['location'][0]"
  ]
}
```


在上述範例中，`['location'][0]` 參考陣列的第一個元素。JSON 使用以零為基礎的陣列索引。陣列索引必須是正整數 (大於或等於零)。

下列範例顯示使用點符號的上一個JSONPaths檔案。

```
{
  "jsonpaths": [
    "$.id",
    "$.store.book.title",
    "$.location[0]"
  ]
}
```

`jsonpaths` 陣列中不能混用方括號標記法和點標記法。方括號標記法和點標記法中都可以使用方括號來參考陣列元素。

使用點符號時，JSONPath運算式不能包含下列字元：

- 一般單引號 (')
- 句點，或點 (.)
- 方括號 ([])，除非用來參考陣列元素

如果JSONPath運算式所參考之名稱值對中的值是物件或陣列，則整個物件或陣列會載入為字串，包括括號或括號。例如，假設JSON您的資料包含下列物件。

```
{
  "id": 0,
  "guid": "84512477-fa49-456b-b407-581d0d851c3c",
  "isActive": true,
  "tags": [
    "nisi",
    "culpa",
    "ad",
    "amet",
    "voluptate",
    "reprehenderit",
    "veniam"
  ],
  "friends": [
    {
      "id": 0,
```

```

        "name": "Martha Rivera"
    },
    {
        "id": 1,
        "name": "Renaldo"
    }
]
}

```

`['tags']` 然後，JSONPath表達式會傳回下列值。

```
["nisi","culpa","ad","amet","voluptate","reprehenderit","veniam"]
```

`['friends'][1]` 然後，JSONPath表達式會傳回下列值。

```
{"id": 1,"name": "Renaldo"}
```

`jsonpaths` 陣列中的每個JSONPath表達式對應至 Amazon Redshift 目標資料表中的一欄。`jsonpaths` 陣列元素的順序必須符合目標資料表或欄清單 (如果使用欄清單) 中的欄順序。

如需示範如何使用 'auto' 引數或JSONPaths檔案以及使用JSON物件或陣列載入資料的範例，請參閱 [從JSON範例複製](#)。

如需有關如何複製多個JSON檔案的資訊，請參閱 [使用資訊清單指定資料檔案](#)。

中的逸出字元 JSON

`COPY \n`會以新行字元載入，並以`\t`標籤字元載入。若要載入反斜線，請加上反斜線 (`\\`) 來逸出。

例如，假設您在 儲存貯體 JSON 中名為 `escape.json`的檔案中具有下列項目`s3://amzn-s3-demo-bucket/json/`。

```

{
  "backslash": "This is a backslash: \\",
  "newline": "This sentence\n is on two lines.",
  "tab": "This sentence \t contains a tab."
}

```

執行下列命令來建立ESCAPES資料表並載入 JSON。

```
create table escapes (backslash varchar(25), newline varchar(35), tab varchar(35));
```

```
copy escapes from 's3://amzn-s3-demo-bucket/json/escape.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as json 'auto';
```

查詢ESCAPES資料表以檢視結果。

```
select * from escapes;

    backslash      |      newline      |      tab
-----+-----+-----
This is a backslash: \ | This sentence      | This sentence      contains a tab.
                        : is on two lines.

(1 row)
```

遺失數值精確度

將數字從JSON格式的資料檔案載入定義為數字資料類型的資料欄時，可能會失去精確度。電腦系統中無法精確表示某些浮點值。因此，您從JSON檔案複製的資料可能無法如預期般四捨五入。為了避免遺失精確度，建議您使用下列其中一個替代方法：

- 以字串來表示數字，用雙引號括住值。
- 使用 [ROUNDEC](#) 來四捨五入數字，而不是截斷數字。
- 不使用 JSON 或 Avro 檔案，而是使用 CSV、字元分隔或固定寬度的文字檔案。

COPY 從資料欄資料格式

COPY 可以下列資料欄格式從 Amazon S3 載入資料：

- ORC
- Parquet

如需 COPY 從資料欄資料格式使用的範例，請參閱 [COPY 範例](#)。

COPY 支援欄式格式化資料，考量如下：

- Amazon S3 儲存貯體必須與 Amazon Redshift 資料庫位於相同的 AWS 區域中。
- 若要透過 VPC 端點存取 Amazon S3 資料，請使用 IAM 政策和 IAM 角色設定存取權，如 [Amazon Redshift 管理指南 中的使用 Amazon Redshift Spectrum 搭配增強 VPC 型路由](#) 所述。
- COPY 不會自動套用壓縮編碼。

- 僅支援下列COPY參數：
 - [ACCEPTINVCHARS](#) 從 ORC或 Parquet 檔案複製時。
 - [FILLRECORD](#)
 - [FROM](#)
 - [IAM_ROLE](#)
 - [CREDENTIALS](#)
 - [STATUPDATE](#)
 - [MANIFEST](#)
 - [EXPLICIT_IDS](#)
- 如果在載入時COPY遇到錯誤，命令會失敗。ACCEPTANYDATE 資料欄資料類型MAXERROR不支援和。
- 錯誤訊息會傳送至SQL用戶端。有些錯誤會記錄在 STL_LOAD_ERRORS 和 STL_ 中ERROR。
- COPY 會以與資料欄資料檔案中的資料欄相同的順序，將值插入目標資料表的資料欄。目標資料表的欄數和資料檔案的欄數必須相符。
- 如果您為 COPY 操作指定的檔案包含下列其中一個延伸模組，我們會解壓縮資料，而不需要新增任何參數：
 - .gz
 - .snappy
 - .bz2
- COPY 從 Parquet 和ORC檔案格式使用 Redshift Spectrum 和儲存貯體存取。若要COPY將用於這些格式，請確定沒有IAM政策封鎖使用 Amazon S3 預先簽章的 URLs。Amazon Redshift URLs 產生的預先簽章有效期為 1 小時，以便 Amazon Redshift 有足夠的時間從 Amazon S3 儲存貯體載入所有檔案。從COPY資料欄資料格式掃描的每個檔案URL都會產生唯一的預先簽章。對於包含 s3:signatureAge 動作的儲存貯體政策，請務必將值設定為至少 3,600,000 毫秒。如需詳細資訊，請參閱[使用 Amazon Redshift Spectrum 搭配增強型VPC路由](#)。

DATEFORMAT 和 TIMEFORMAT字串

COPY 命令會使用 DATEFORMAT和 TIMEFORMAT選項來剖析來源資料中的日期和時間值。DATEFORMAT 和 TIMEFORMAT是格式化字串，必須符合來源資料的日期和時間值格式。例如，具有日期值的COPY命令載入來源資料Jan-01-1999必須包含下列DATEFORMAT字串：

```
COPY ...  
COPY
```

```
DATEFORMAT AS 'MON-DD-YYYY'
```

如需管理COPY資料轉換的詳細資訊，請參閱[資料轉換參數](#)。

DATEFORMAT 和 TIMEFORMAT字串可包含日期時間分隔符號（例如 '-'、'/' 或 ':'），以及下表中的日期部分和時間部分格式。

Note

如果您無法將日期或時間值的格式與下列日期區段和時間區段比對，或者您有使用彼此不同格式的日期和時間值，請使用 'auto' 引數搭配 DATEFORMAT或 TIMEFORMAT 參數。'auto' 引數會識別使用 DATEFORMAT或 TIMEFORMAT字串時不支援的數種格式。如需詳細資訊，請參閱[搭配 DATEFORMAT和 使用自動辨識 TIMEFORMAT](#)。

日期部分或時間部分	意義
YY	年，不含世紀
YYYY	年，包含世紀
MM	月，以數字表示
MON	月，以名稱表示 (縮寫名稱或完整名稱)
DD	月中的日，以數字表示
HH 或 HH24	小時 (24 小時制)
	<div data-bbox="852 1430 980 1465" data-label="Section-Header"> <h4>Note</h4> </div> <div data-bbox="899 1480 1456 1663" data-label="Text"> <p>在SQL函數的DATETIME格式字串中，HH 與 相同HH12。不過，在 DATEFORMAT和 的TIMEFORMAT字串中COPY，HH 與 相同HH24。</p> </div>
HH12	小時 (12 小時制)
MI	分鐘

日期部分或時間部分	意義
SS	秒鐘
AM 或 PM	正午指標 (用於 12 小時制)

預設日期格式為 YYYY-MM-DD。沒有時區 (TIMESTAMP) 格式的預設時間戳記為 YYYY-MM-DD HH : MI : SS。具有時區 (TIMESTAMPTZ) 格式的預設時間戳記為 YYYY-MM-DD HH : MI : SSOFF，其中 OFF 是 的偏移 UTC (例如 -8 : 00。您無法在 timeformat_string 中包含時區規格符 (TZ、tz 或 OF)。秒 (SS) 欄位也支援微秒層級的分數秒詳細資訊。若要載入與預設格式不同的 TIMESTAMPTZ 資料，請指定「自動」。

以下是您在來源資料中遇到的一些範例日期或時間，以及它們的對應 DATEFORMAT 或 TIMEFORMAT 字串。

來源資料日期或時間範例	DATEFORMAT 或 TIMEFORMAT 語法
03/31/2003	DATEFORMAT AS 'MM/DD/YYYY'
2003 年 3 月 31 日	DATEFORMAT AS 'MON DD, YYYY'
03.31.2003 18:45:05	TIMEFORMAT AS 'MM.DD。YYYY HH : MI : SS'
03.31.2003 18:45:05.123456	

範例

如需使用的範例 TIMEFORMAT，請參閱 [載入時間戳記或日期戳記](#)。

搭配 DATEFORMAT 和使用自動辨識 TIMEFORMAT

如果您將指定 'auto' 為 DATEFORMAT 或 TIMEFORMAT 參數的引數，Amazon Redshift 會自動識別並轉換來源資料中的日期格式或時間格式。下列顯示一個範例。

```
copy favoritemovies from 'dynamodb://ProductCatalog'
```

```
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
dateformat 'auto';
```

與 DATEFORMAT 和的 'auto' 引數搭配使用時 TIMEFORMAT，會 COPY 識別並轉換 中資料表中列出的日期和時間格式 [DATEFORMAT 和 TIMEFORMAT 字串](#)。此外，引 'auto' 數會識別使用 DATEFORMAT 和 TIMEFORMAT 字串時不支援的下列格式。

格式	有效輸入字串的範例
ISO 8601	2019-02-11T05:09:12.195Z
凱撒曆	J2451187
BC	Jan-08-95 BC
YYYYMMDD HHMISS	19960108 040809
YYMMDD HHMISS	960108 040809
YYYY.DDD	1996.008
YYYY-MM-DD HH : MI : SS。 SSS	1996-01-08 04:05:06.789
DD Mon HH : MI : SS YYYY TZ	1997 年 12 月 17 日 07 : 37 : 16 PST
MM/DD/YYYY HH : MI : SS.SS TZ	12/17/1997 07 : 37 : 16.00 PST
YYYY-MM-DD HH : MI : SS+/-TZ	1997-12-17 07:37:16-08
https : /DD.MM。 YYYY HH : MI : SS TZ	12.17.1997 07 : 37 : 16.00 PST

自動辨識不支援 epoch 秒和 epoch 毫秒。

若要測試日期或時間戳記值是否會自動轉換，請使用 CAST 函數嘗試將字串轉換為日期或時間戳記值。例如，下列命令測試時間戳記值 'J2345678 04:05:06.789'：

```
create table formattest (test char(21));
insert into formattest values('J2345678 04:05:06.789');
select test, cast(test as timestamp) as timestamp, cast(test as date) as date from
formattest;
```

test	timestamp	date
J2345678 04:05:06.789	1710-02-23 04:05:06	1710-02-23

如果DATE資料欄的來源資料包含時間資訊，則會截斷時間元件。如果TIMESTAMP資料欄的來源資料省略時間資訊，則時間元件會使用 00 : 00 : 00。

COPY 範例

Note

這些範例包含換行以方便閱讀。請勿在 credentials-args 字串中包含換行或空格。

主題

- [FAVORITEMOVIES 從 DynamoDB 資料表載入](#)
- [LISTING 從 Amazon S3 儲存貯體載入](#)
- [LISTING 從 Amazon EMR叢集載入](#)
- [使用資訊清單指定資料檔案](#)
- [LISTING 從管道分隔檔案載入 \(預設分隔符號 \)](#)
- [LISTING 使用 Parquet 格式的資料欄資料載入](#)
- [LISTING 使用ORC格式的資料欄資料載入](#)
- [EVENT 使用 選項載入](#)
- [VENUE 從固定寬度資料檔案載入](#)
- [CATEGORY 從CSV檔案載入](#)
- [VENUE 載入資料IDENTITY欄的明確值](#)
- [TIME 從管道分隔GZIP檔案載入](#)
- [載入時間戳記或日期戳記](#)
- [從含有預設值的檔案載入資料](#)

- [COPY 具有 ESCAPE選項的資料](#)
- [從JSON範例複製](#)
- [從 Avro 複製的範例](#)
- [COPY 使用 ESCAPE選項為 準備檔案](#)
- [將 Shapefile 載入 Amazon Redshift](#)
- [COPY 具有 NOLOAD選項的 命令](#)
- [COPY 具有多位元組分隔符號的 命令，以及 ENCODING選項](#)

FAVORITEMOVIES 從 DynamoDB 資料表載入

包含建立名為電影 AWS SDKs 的 DynamoDB 資料表的簡單範例。(關於此範例，請參閱 [DynamoDB 入門](#)。) 下列範例會將來自 DynamoDB MOVIES資料表的資料載入 Amazon Redshift 資料表。Amazon Redshift 資料表必須已存在於資料庫中。

```
copy favoritemovies from 'dynamodb://Movies'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
readratio 50;
```

LISTING 從 Amazon S3 儲存貯體載入

下列範例LISTING會從 Amazon S3 儲存貯體載入。COPY 命令會載入/data/listing/資料夾中的所有檔案。

```
copy listing  
from 's3://amzn-s3-demo-bucket/data/listing/'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

LISTING 從 Amazon EMR叢集載入

下列範例會從 Amazon EMR叢集中的 lzop 壓縮檔案載入具有 Tab 分隔資料的SALES資料表。COPY 會載入以 開頭的myoutput/資料夾中的每個檔案part-。

```
copy sales  
from 'emr://j-SAMPLE2B500FC/myoutput/part-*'   
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
delimiter '\t' lzop;
```

下列範例會在 Amazon JSON EMR叢集中載入具有格式化資料的SALES資料表。COPY 會載入myoutput/json/資料夾中的每個檔案。

```
copy sales
from 'emr://j-SAMPLE2B500FC/myoutput/json/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
JSON 's3://amzn-s3-demo-bucket/jsonpaths.txt';
```

使用資訊清單指定資料檔案

您可以使用資訊清單來確保您的COPY命令從 Amazon S3 載入所有必要檔案，以及僅載入必要檔案。需要從不同儲存貯體載入多個檔案，或載入不共用相同字首的檔案時，您也可以使用資訊清單。

例如，假設您需要載入下列三個檔案：custdata1.txt、custdata2.txt 和 custdata3.txt。您可以使用下列命令指定字首，以載入 amzn-s3-demo-bucket 中開頭為 custdata 的所有檔案：

```
copy category
from 's3://amzn-s3-demo-bucket/custdata'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如果因為錯誤而只有兩個檔案存在，則只會COPY載入這兩個檔案並成功完成，導致資料載入不完整。如果儲存貯體也包含使用相同字首的不需要檔案，例如名為 custdata.backup 的檔案，也會COPY載入該檔案，導致載入不需要的資料。

若要確保載入所有必要檔案並防止載入不需要的檔案，您可以使用資訊清單檔案。資訊清單是 JSON格式化的文字檔案，列出命令要處理的檔案COPY。例如，下列資訊清單會載入上述範例中的三個檔案。

```
{
  "entries":[
    {
      "url":"s3://amzn-s3-demo-bucket/custdata.1",
      "mandatory":true
    },
    {
      "url":"s3://amzn-s3-demo-bucket/custdata.2",
      "mandatory":true
    },
    {
      "url":"s3://amzn-s3-demo-bucket/custdata.3",
      "mandatory":true
    }
  ]
}
```

```
]
}
```

選用的mandatory旗標會指出如果檔案不存在，是否COPY應該終止。預設值為 false。無論任何強制性設定為何，如果找不到檔案，都會COPY終止。在此範例中，如果找不到任何檔案，會COPY傳回錯誤。如果您僅指定金鑰前綴 (例如 custdata.backup)，則可能已挑選的不必要檔案會被忽略，因為這些檔案不在資訊清單上。

從 ORC或 Parquet 格式的資料檔案載入時，需要一個meta欄位，如下列範例所示。

```
{
  "entries":[
    {
      "url":"s3://amzn-s3-demo-bucket1/orc/2013-10-04-custdata",
      "mandatory":true,
      "meta":{
        "content_length":99
      }
    },
    {
      "url":"s3://amzn-s3-demo-bucket2/orc/2013-10-05-custdata",
      "mandatory":true,
      "meta":{
        "content_length":99
      }
    }
  ]
}
```

以下範例使用名為 cust.manifest 的資訊清單。

```
copy customer
from 's3://amzn-s3-demo-bucket/cust.manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as orc
manifest;
```

您可以使用資訊清單從不同儲存貯體載入檔案，或載入不共用相同字首的檔案。下列範例顯示 JSON 以檔案名稱開頭為日期戳的 檔案載入資料。

```
{
```

```

"entries": [
  {"url":"s3://amzn-s3-demo-bucket/2013-10-04-custdata.txt","mandatory":true},
  {"url":"s3://amzn-s3-demo-bucket/2013-10-05-custdata.txt","mandatory":true},
  {"url":"s3://amzn-s3-demo-bucket/2013-10-06-custdata.txt","mandatory":true},
  {"url":"s3://amzn-s3-demo-bucket/2013-10-07-custdata.txt","mandatory":true}
]
}

```

只要儲存貯體與叢集位於相同 AWS 區域，清單就可以列出位於不同儲存貯體中的檔案。

```

{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata1.txt","mandatory":false},
    {"url":"s3://amzn-s3-demo-bucket2/custdata1.txt","mandatory":false},
    {"url":"s3://amzn-s3-demo-bucket2/custdata2.txt","mandatory":false}
  ]
}

```

LISTING 從管道分隔檔案載入（預設分隔符號）

下列範例是非常簡單的案例，其中沒有指定任何選項，且輸入檔案包含預設分隔符號，即縱線字元（'|'）。

```

copy listing
from 's3://amzn-s3-demo-bucket/data/listings_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';

```

LISTING 使用 Parquet 格式的資料欄資料載入

下列範例從 Amazon S3 上名為 parquet 的資料夾載入資料。

```

copy listing
from 's3://amzn-s3-demo-bucket/data/listings/parquet/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as parquet;

```

LISTING 使用ORC格式的資料欄資料載入

下列範例從 Amazon S3 上名為 orc 的資料夾載入資料。

```

copy listing

```

```
from 's3://amzn-s3-demo-bucket/data/listings/orc/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as orc;
```

EVENT 使用 選項載入

下列範例會將管道分隔資料載入EVENT資料表，並套用下列規則：

- 如果使用成對的引號來括住任何字元字串，則會移除引號。
- 空白字串和包含空白的字串都會載入為NULL值。
- 如果傳回 5 個以上的錯誤，載入會失敗。
- 時間戳記值必須符合指定的格式；例如，有效時間戳記為 2008-09-26 05:43:12。

```
copy event
from 's3://amzn-s3-demo-bucket/data/allevnts_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
removequotes
emptyasnull
blanksasnull
maxerror 5
delimiter '|'
timeformat 'YYYY-MM-DD HH:MI:SS';
```

VENUE 從固定寬度資料檔案載入

```
copy venue
from 's3://amzn-s3-demo-bucket/data/venue_fw.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth 'venueid:3,venueid:25,venueid:12,venueid:2,venueid:6';
```

上述範例假設資料檔案的格式與所顯示的樣本資料相同。在下列樣本中，空格充當預留位置，讓所有欄都是規格所指定的相同寬度：

```
1 Toyota Park           Bridgeview  IL0
2 Columbus Crew Stadium Columbus    OH0
3 RFK Stadium           Washington  DC0
4 CommunityAmerica BallparkKansas City KS0
5 Gillette Stadium      Foxborough MA68756
```

CATEGORY 從CSV檔案載入

假設您想要CATEGORY使用下表所示的值載入。

catid	catgroup	catname	catdesc
12	Shows	Musicals	Musical theatre
13	Shows	Plays	All "non-musical" theatre
14	Shows	Opera	All opera, light, and "rock" opera
15	Concerts	Classical	All symphony, concerto, and choir concerts

下列範例顯示文字檔案的內容，欄位值以逗號分隔。

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,All "non-musical" theatre
14,Shows,Opera,All opera, light, and "rock" opera
15,Concerts,Classical,All symphony, concerto, and choir concerts
```

如果您使用 DELIMITER 參數載入檔案來指定逗號分隔輸入，則COPY命令會失敗，因為某些輸入欄位包含逗號。您可以使用 CSV 參數，並在引號字元中包含逗號的欄位來避免此問題。如果引號括住的字串內出現引號字元，則需要多加一個引號字元才能將其逸出。預設引號字元是雙引號，所以您需要多加一個雙引號來逸出每一個雙引號。新的輸入檔案如下所示。

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,"All ""non-musical"" theatre"
14,Shows,Opera,"All opera, light, and ""rock"" opera"
15,Concerts,Classical,"All symphony, concerto, and choir concerts"
```

假設檔案名稱為 category_csv.txt，您可以使用下列COPY命令載入檔案：

```
copy category
from 's3://amzn-s3-demo-bucket/data/category_csv.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
csv;
```

或者，若要避免需要逸出輸入中的雙引號，您可以使用 QUOTE AS 參數指定不同的引號字元。例如，下列的 category_csv.txt 版本使用 '%' 做為引號字元。

```
12,Shows,Musicals,Musical theatre
13,Shows,Plays,%All "non-musical" theatre%
14,Shows,Opera,%All opera, light, and "rock" opera%
15,Concerts,Classical,%All symphony, concerto, and choir concerts%
```

下列 COPY 命令使用 QUOTE AS 載入 category_csv.txt：

```
copy category
from 's3://amzn-s3-demo-bucket/data/category_csv.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
csv quote as '%';
```

VENUE 載入資料 IDENTITY 欄的明確值

下列範例假設建立 VENUE 資料表時，至少指定一個資料欄（例如 venueid 資料欄）為資料 IDENTITY 欄。此命令會覆寫資料 IDENTITY 欄自動產生值的預設 IDENTITY 行為，而是從 venue.txt 檔案載入明確值。使用 EXPLICIT_IDS 選項時，Amazon Redshift 不會檢查是否有重複 IDENTITY 值載入資料表。

```
copy venue
from 's3://amzn-s3-demo-bucket/data/venue.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
explicit_ids;
```

TIME 從管道分隔 GZIP 檔案載入

下列範例會從以管道分隔 GZIP 的檔案載入 TIME 資料表：

```
copy time
from 's3://amzn-s3-demo-bucket/data/timerows.gz'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
gzip
delimiter '|';
```

載入時間戳記或日期戳記

下列範例載入含有格式化時間戳記的資料。

Note

TIMEFORMAT 的 HH:MI:SS 也可以支援超出 SS 到微秒層級詳細資訊的小秒數。此範例中使用的檔案 time.txt 包含一行，即 2009-01-12 14:15:57.119568。

```
copy timestamp1
from 's3://amzn-s3-demo-bucket/data/time.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
timeformat 'YYYY-MM-DD HH:MI:SS';
```

此複製的結果如下所示：

```
select * from timestamp1;
c1
-----
2009-01-12 14:15:57.119568
(1 row)
```

從含有預設值的檔案載入資料

下列範例使用 TICKIT 資料庫中 VENUE 資料表的變化。請考慮使用下列陳述式定義的 VENUE_NEW 資料表：

```
create table venue_new(
venueid smallint not null,
venueid varchar(100) not null,
venuecity varchar(30),
venuestate char(2),
venuestate integer not null default '1000');
```

請考慮不含資料 VENUESEATS 欄值的 venue_noseats.txt 資料檔案，如下列範例所示：

```
1|Toyota Park|Bridgeview|IL|
2|Columbus Crew Stadium|Columbus|OH|
3|RFK Stadium|Washington|DC|
4|CommunityAmerica Ballpark|Kansas City|KS|
5|Gillette Stadium|Foxborough|MA|
6|New York Giants Stadium|East Rutherford|NJ|
```



```
7|BMO Field|Toronto|ON|
8|The Home Depot Center|Carson|CA|
9|Dick's Sporting Goods Park|Commerce City|CO|
10|Pizza Hut Park|Frisco|TX|
```

下列COPY陳述式會從 檔案成功載入資料表，並將DEFAULT值（'1000'）套用至省略的資料欄：

```
copy venue_new(venueid, venue_name, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_noseats.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

現在檢視已載入的資料表：

```
select * from venue_new order by venueid;
venueid |          venue_name          |   venuecity   | venuestate | venue_seats
-----+-----+-----+-----+-----
1 | Toyota Park                  | Bridgeview    | IL         |          1000
2 | Columbus Crew Stadium        | Columbus      | OH         |          1000
3 | RFK Stadium                  | Washington    | DC         |          1000
4 | CommunityAmerica Ballpark    | Kansas City   | KS         |          1000
5 | Gillette Stadium             | Foxborough    | MA         |          1000
6 | New York Giants Stadium      | East Rutherford | NJ         |          1000
7 | BMO Field                    | Toronto       | ON         |          1000
8 | The Home Depot Center        | Carson         | CA         |          1000
9 | Dick's Sporting Goods Park   | Commerce City | CO         |          1000
10 | Pizza Hut Park                | Frisco        | TX         |          1000
(10 rows)
```

針對下列範例，除了假設檔案中不包含VENUESEATS資料之外，也假設不包含VENUENAME資料：

```
1||Bridgeview|IL|
2||Columbus|OH|
3||Washington|DC|
4||Kansas City|KS|
5||Foxborough|MA|
6||East Rutherford|NJ|
7||Toronto|ON|
8||Carson|CA|
9||Commerce City|CO|
10||Frisco|TX|
```

使用相同的資料表定義，下列COPY陳述式會失敗，因為沒有為指定DEFAULT值VENUENAME，且VENUENAME是資料NOTNULL欄：

```
copy venue(venueid, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

現在請考慮使用資料IDENTITY欄的VENUE資料表變化：

```
create table venue_identity(
venueid int identity(1,1),
venueid varchar(100) not null,
venuecity varchar(30),
venuestate char(2),
venueid integer not null default '1000');
```

與上一個範例一樣，假設資料VENUSEATS欄在來源檔案中沒有對應的值。下列COPY陳述式會成功載入資料表，包括預先定義的IDENTITY資料值，而不是自動產生這些值：

```
copy venue(venueid, venueid, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' explicit_ids;
```

此陳述式失敗，因為它不包含IDENTITY資料欄（VENUID從資料欄清單中遺失），但包含EXPLICIT_IDS參數：

```
copy venue(venueid, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' explicit_ids;
```

此陳述式失敗，因為它不包含EXPLICIT_IDS參數：

```
copy venue(venueid, venueid, venuecity, venuestate)
from 's3://amzn-s3-demo-bucket/data/venue_pipe.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|';
```

COPY 具有 ESCAPE 選項的資料

下列範例示範如何載入符合分隔符號字元 (在此例子中是縱線字元) 的字元。在輸入檔案中，請確定您要載入的所有縱線字元 (|) 都以反斜線字元 (\) 逸出。然後使用 ESCAPE 參數載入 檔案。

```
$ more redshiftinfo.txt
1|public\|event\|dwuser
2|public\|sales\|dwuser

create table redshiftinfo(infoid int,tableinfo varchar(50));

copy redshiftinfo from 's3://amzn-s3-demo-bucket/data/redshiftinfo.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter '|' escape;

select * from redshiftinfo order by 1;
infoid |      tableinfo
-----+-----
1      | public|event|dwuser
2      | public|sales|dwuser
(2 rows)
```

如果沒有 ESCAPE 參數，此 COPY 命令會失敗並發生錯誤 `Extra column(s) found`。

Important

如果您使用 COPY 具有 ESCAPE 參數的 載入資料，您還必須使用 UNLOAD 命令指定 ESCAPE 參數，以產生互惠輸出檔案。同樣地，如果您 UNLOAD 使用 ESCAPE 參數，則需要在 COPY 相同的資料 ESCAPE 時使用。

從 JSON 範例複製

在下列範例中，您會使用下列資料載入 CATEGORY 資料表。

CATID	CATGROUP	CATNAME	CATDESC
1	運動	MLB	美國職棒大聯盟
2	運動	NHL	National Hockey League

CATID	CATGROUP	CATNAME	CATDESC
3	運動	NFL	National Football League
4	運動	NBA	National Basketball Association
5	Concerts	Classical	All symphony, concerto, and choir concerts

主題

- [使用「自動」選項從JSON資料載入](#)
- [使用「自動忽略案例」選項從JSON資料載入](#)
- [使用 JSONPaths 檔案從JSON資料載入](#)
- [使用 JSONPaths 檔案從JSON陣列載入](#)

使用「自動」選項從JSON資料載入

若要使用 'auto' 選項從JSON資料載入，JSON資料必須包含一組物件。金鑰名稱必須符合欄名稱，但順序並不重要。以下顯示一個名為 category_object_auto.json 之檔案的內容。

```
{
  "catdesc": "Major League Baseball",
  "catid": 1,
  "catgroup": "Sports",
  "catname": "MLB"
}
{
  "catgroup": "Sports",
  "catid": 2,
  "catname": "NHL",
  "catdesc": "National Hockey League"
}
{
  "catid": 3,
  "catname": "NFL",
  "catgroup": "Sports",
  "catdesc": "National Football League"
}
{
  "bogus": "Bogus Sports LLC",
```

```
"catid": 4,
"catgroup": "Sports",
"catname": "NBA",
"catdesc": "National Basketball Association"
}
{
"catid": 5,
"catgroup": "Shows",
"catname": "Musicals",
"catdesc": "All symphony, concerto, and choir concerts"
}
```

若要從上一個範例中JSON的資料檔案載入，請執行下列COPY命令。

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_auto.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 'auto';
```

使用「自動忽略案例」選項從JSON資料載入

若要使用 'auto ignorecase' 選項從JSON資料載入，JSON資料必須包含一組物件。索引鍵名稱的大小寫不必符合欄名稱，順序也不重要。以下顯示一個名為 category_object_auto-ignorecase.json 之檔案的內容。

```
{
  "CatDesc": "Major League Baseball",
  "CatID": 1,
  "CatGroup": "Sports",
  "CatName": "MLB"
}
{
  "CatGroup": "Sports",
  "CatID": 2,
  "CatName": "NHL",
  "CatDesc": "National Hockey League"
}
{
  "CatID": 3,
  "CatName": "NFL",
  "CatGroup": "Sports",
  "CatDesc": "National Football League"
}
```

```
{
  "bogus": "Bogus Sports LLC",
  "CatID": 4,
  "CatGroup": "Sports",
  "CatName": "NBA",
  "CatDesc": "National Basketball Association"
}
{
  "CatID": 5,
  "CatGroup": "Shows",
  "CatName": "Musicals",
  "CatDesc": "All symphony, concerto, and choir concerts"
}
```

若要從上一個範例中JSON的資料檔案載入，請執行下列COPY命令。

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_auto ignorecase.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 'auto ignorecase';
```

使用 JSONPaths 檔案從JSON資料載入

如果JSON資料物件未直接對應至資料欄名稱，您可以使用 JSONPaths 檔案將JSON元素對應至資料欄。順序與JSON來源資料無關，但JSONPaths檔案表達式的順序必須符合資料欄順序。假設您有下列資料檔案，名稱為 `category_object_paths.json`。

```
{
  "one": 1,
  "two": "Sports",
  "three": "MLB",
  "four": "Major League Baseball"
}
{
  "three": "NHL",
  "four": "National Hockey League",
  "one": 2,
  "two": "Sports"
}
{
  "two": "Sports",
  "three": "NFL",
```

```

    "one": 3,
    "four": "National Football League"
  }
  {
    "one": 4,
    "two": "Sports",
    "three": "NBA",
    "four": "National Basketball Association"
  }
  {
    "one": 6,
    "two": "Shows",
    "three": "Musicals",
    "four": "All symphony, concerto, and choir concerts"
  }
}

```

下列名為 JSONPaths 的檔案會將來源資料 category_jsonpath.json 對應至資料表資料欄。

```

{
  "jsonpaths": [
    "$['one']",
    "$['two']",
    "$['three']",
    "$['four']"
  ]
}

```

若要從上一個範例中 JSON 的資料檔案載入，請執行下列 COPY 命令。

```

copy category
from 's3://amzn-s3-demo-bucket/category_object_paths.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 's3://amzn-s3-demo-bucket/category_jsonpath.json';

```

使用 JSONPaths 檔案從 JSON 陣列載入

若要從由一組陣列組成的 JSON 資料載入，您必須使用 JSONPaths 檔案將陣列元素對應至資料欄。假設您有下列資料檔案，名稱為 category_array_data.json。

```

[1,"Sports","MLB","Major League Baseball"]
[2,"Sports","NHL","National Hockey League"]
[3,"Sports","NFL","National Football League"]

```

```
[4,"Sports","NBA","National Basketball Association"]
[5,"Concerts","Classical","All symphony, concerto, and choir concerts"]
```

下列名為 JSONPaths 的檔案會將來源資料 category_array_jsonpath.json 映射至資料表資料欄。

```
{
  "jsonpaths": [
    "$[0]",
    "$[1]",
    "$[2]",
    "$[3]"
  ]
}
```

若要從上一個範例中 JSON 的資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://amzn-s3-demo-bucket/category_array_data.json'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
json 's3://amzn-s3-demo-bucket/category_array_jsonpath.json';
```

從 Avro 複製的範例

在下列範例中，您會使用下列資料載入 CATEGORY 資料表。

CATID	CATGROUP	CATNAME	CATDESC
1	運動	MLB	美國職棒大聯盟
2	運動	NHL	National Hockey League
3	運動	NFL	National Football League
4	運動	NBA	National Basketball Association
5	Concerts	Classical	All symphony, concerto, and choir concerts

主題

- [使用 'auto' 選項從 Avro 資料載入](#)
- [使用 'auto ignorecase' 選項從 Avro 資料載入](#)
- [使用 JSONPaths 檔案從 Avro 資料載入](#)

使用 'auto' 選項從 Avro 資料載入

若要使用 'auto' 引數從 Avro 資料載入，Avro 結構描述中的欄位名稱必須符合欄名稱。使用 'auto' 引數時，順序並不重要。以下顯示一個名為 `category_auto.avro` 之檔案的結構描述。

```
{
  "name": "category",
  "type": "record",
  "fields": [
    {"name": "catid", "type": "int"},
    {"name": "catdesc", "type": "string"},
    {"name": "catname", "type": "string"},
    {"name": "catgroup", "type": "string"},
  ]
}
```

Avro 檔案中的資料是二進位格式，無法直接閱讀。下列顯示 `category_auto.avro` 檔案中資料的 JSON 表示法。

```
{
  "catid": 1,
  "catdesc": "Major League Baseball",
  "catname": "MLB",
  "catgroup": "Sports"
}
{
  "catid": 2,
  "catdesc": "National Hockey League",
  "catname": "NHL",
  "catgroup": "Sports"
}
{
  "catid": 3,
  "catdesc": "National Basketball Association",
  "catname": "NBA",
  "catgroup": "Sports"
}
{
```

```
"catid": 4,  
"catdesc": "All symphony, concerto, and choir concerts",  
"catname": "Classical",  
"catgroup": "Concerts"  
}
```

若要從上一個範例中的 Avro 資料檔案載入，請執行下列COPY命令。

```
copy category  
from 's3://amzn-s3-demo-bucket/category_auto.avro'  
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'  
format as avro 'auto';
```

使用 'auto ignorecase' 選項從 Avro 資料載入

若要使用 'auto ignorecase' 引數從 Avro 資料載入，Avro 結構描述中欄位名稱的大小寫不必符合欄名稱的大小寫。使用 'auto ignorecase' 引數時，順序並不重要。以下顯示一個名為 category_auto-ignorecase.avro 之檔案的結構描述。

```
{  
  "name": "category",  
  "type": "record",  
  "fields": [  
    {"name": "CatID", "type": "int"},  
    {"name": "CatDesc", "type": "string"},  
    {"name": "CatName", "type": "string"},  
    {"name": "CatGroup", "type": "string"},  
  ]  
}
```

Avro 檔案中的資料是二進位格式，無法直接閱讀。下列顯示category_auto-ignorecase.avro檔案中資料的JSON表示法。

```
{  
  "CatID": 1,  
  "CatDesc": "Major League Baseball",  
  "CatName": "MLB",  
  "CatGroup": "Sports"  
}  
{  
  "CatID": 2,  
  "CatDesc": "National Hockey League",
```

```
"CatName": "NHL",
"CatGroup": "Sports"
}
{
  "CatID": 3,
  "CatDesc": "National Basketball Association",
  "CatName": "NBA",
  "CatGroup": "Sports"
}
{
  "CatID": 4,
  "CatDesc": "All symphony, concerto, and choir concerts",
  "CatName": "Classical",
  "CatGroup": "Concerts"
}
```

若要從上一個範例中的 Avro 資料檔案載入，請執行下列COPY命令。

```
copy category
from 's3://amzn-s3-demo-bucket/category_auto-ignorecase.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format as avro 'auto ignorecase';
```

使用 JSONPaths 檔案從 Avro 資料載入

如果 Avro 結構描述中的欄位名稱未直接對應至資料欄名稱，您可以使用 JSONPaths 檔案將結構描述元素對應至資料欄。JSONPaths 檔案表達式的順序必須符合資料欄順序。

假設您有一個名為 `category_paths.avro` 的資料檔案，其中包含的資料與上述範例相同，但使用下列結構描述。

```
{
  "name": "category",
  "type": "record",
  "fields": [
    {"name": "id", "type": "int"},
    {"name": "desc", "type": "string"},
    {"name": "name", "type": "string"},
    {"name": "group", "type": "string"},
    {"name": "region", "type": "string"}
  ]
}
```

下列名為 JSONPaths 的檔案會將來源資料 `category_path.avropath` 對應至資料表資料欄。

```
{
  "jsonpaths": [
    "$['id']",
    "$['group']",
    "$['name']",
    "$['desc']"
  ]
}
```

若要從上一個範例中的 Avro 資料檔案載入，請執行下列 COPY 命令。

```
copy category
from 's3://amzn-s3-demo-bucket/category_object_paths.avro'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
format avro 's3://amzn-s3-demo-bucket/category_path.avropath ';
```

COPY 使用 ESCAPE 選項為 準備檔案

下列範例說明如何在使用 COPY 命令搭配 ESCAPE 參數將資料匯入 Amazon Redshift 資料表之前，將資料準備為「逸出」新行字元。在不準備資料以分隔新行字元的情況下，Amazon Redshift 會在您執行 COPY 命令時傳回載入錯誤，因為新行字元通常用作記錄分隔符號。

例如，假設您想要將一個檔案或外部資料表中的一欄複製到 Amazon Redshift 資料表。如果檔案或資料欄包含 XML 格式化的內容或類似資料，您需要確保作為內容一部分的所有新行字元 (`\n`) 都使用反斜線字元 (`\`) 逸出。

包含內嵌換行字元的檔案或資料表可提供相當簡單的比對模式。每一個內嵌的換行字元很可能都接在 `>` 字元後面，且之間可能有幾個空格字元 (`'` 或 Tab 字元)，如下列範例所示 (在名為 `nlTest1.txt` 的文字檔案中)。

```
$ cat nlTest1.txt
<xml start>
<newline characters provide>
<line breaks at the end of each>
<line in content>
</xml>|1000
<xml>
</xml>|2000
```

在下列範例中，您可以執行文字處理公用程式來預先處理來源檔案，並在需要的地方插入逸出字元。(當欄資料複製到 Amazon Redshift 資料表時，會將 | 字元主要做為分隔符號來分隔欄資料。)

```
$ sed -e ':a;N;$!ba;s/>[[[:space:]]*\n/>\\\n/g' nlTest1.txt > nlTest2.txt
```

同樣地，您可以使用 Perl 來執行類似的操作：

```
cat nlTest1.txt | perl -p -e 's/>\s*\n/>\\\n/g' > nlTest2.txt
```

為了方便將 nlTest2.txt 檔案中的資料載入 Amazon Redshift，我們在 Amazon Redshift 中建立一個兩欄資料表。第一欄 c1 是一個字元欄，可存放來自 nlTest2.txt 檔案的 XML 格式化內容。第二欄 c2 存放從同一個檔案載入的整數值。

執行 sed 命令後，您可以使用 ESCAPE 參數，將 nlTest2.txt 檔案的資料正確載入 Amazon Redshift 資料表。

Note

當您在 COPY 命令中包含 ESCAPE 參數時，它會逸出包含反斜線字元（包括新行）的多個特殊字元。

```
copy t2 from 's3://amzn-s3-demo-bucket/data/nlTest2.txt'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
escape
delimiter as '|';
```

```
select * from t2 order by 2;
```

```
c1          | c2
-----+-----
<xml start>
<newline characters provide>
<line breaks at the end of each>
<line in content>
</xml>
| 1000
<xml>
</xml>      | 2000
(2 rows)
```

您可以用類似的方法來準備從外部資料庫匯出的資料檔案。例如，透過 Oracle 資料庫，您可以在要複製到 Amazon Redshift 的資料表中每個受影響的資料欄上使用 REPLACE 函數。

```
SELECT c1, REPLACE(c2, \n',\\n' ) as c2 from my_table_with_xml
```

此外，許多定期處理大量資料的資料庫匯出和擷取、轉換、載入（ETL）工具提供用於指定逸出字元和分隔字元的選項。

將 Shapefile 載入 Amazon Redshift

下列範例示範如何使用載入 Esri 形狀檔 COPY。如需載入 Shapefile 的相關資訊，請參閱 [將 Shapefile 載入 Amazon Redshift](#)。

載入 Shapefile

下列步驟說明如何使用 COPY 命令從 Amazon S3 擷取 OpenStreetMap 資料。此範例假設來自 [Geofabrik 下載網站](#) 的挪威形狀檔案封存已上傳至您 AWS 區域中的私有 Amazon S3 儲存貯體。 .shp、.shx 和 .dbf 檔案必須共用相同的 Amazon S3 字首和檔案名稱。

不使用簡化擷取資料

以下命令會建立資料表並擷取資料，這些資料無需任何簡化即可符合最大幾何大小。在偏好的 GIS 軟體 gis_osm_natural_free_1.shp 中開啟，並檢查此層中的資料欄。依預設，IDENTITY 或 GEOMETRY 資料欄為第一欄。GEOMETRY 欄為第一個時，您可以建立資料表，如下所示。

```
CREATE TABLE norway_natural (  
  wkb_geometry GEOMETRY,  
  osm_id BIGINT,  
  code INT,  
  fclass VARCHAR,  
  name VARCHAR);
```

或者，當資料 IDENTITY 欄為第一個時，您可以建立資料表，如下所示。

```
CREATE TABLE norway_natural_with_id (  
  fid INT IDENTITY(1,1),  
  wkb_geometry GEOMETRY,  
  osm_id BIGINT,  
  code INT,  
  fclass VARCHAR,  
  name VARCHAR);
```

現在您可以使用 擷取資料COPY。

```
COPY norway_natural FROM 's3://bucket_name/shapefiles/norway/
gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural' completed, 83891 record(s) loaded successfully
```

或者，您可以如下所示擷取資料。

```
COPY norway_natural_with_id FROM 's3://bucket_name/shapefiles/norway/
gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural_with_id' completed, 83891 record(s) loaded
successfully.
```

使用簡化擷取資料

下列命令會建立資料表並嘗試擷取資料，這些資料若不經簡化就無法符合最大幾何大小。檢查 `gis_osm_water_a_free_1.shp` Shapefile 並建立適當資料表，如下所示。

```
CREATE TABLE norway_water (
  wkb_geometry GEOMETRY,
  osm_id BIGINT,
  code INT,
  fclass VARCHAR,
  name VARCHAR);
```

COPY 命令執行時，會導致錯誤。

```
COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
ERROR: Load into table 'norway_water' failed. Check 'stl_load_errors' system table
for details.
```

查詢 STL_LOAD_ERRORS 顯示幾何太大。

```
SELECT line_number, btrim(colname), btrim(err_reason) FROM stl_load_errors WHERE query
= pg_last_copy_id();
```

```

line_number |      btrim      |
-----+-----
+-----+-----
      1184705 | wkb_geometry | Geometry size: 1513736 is larger than maximum supported
size: 1048447

```

為了克服這個問題，SIMPLIFY AUTO 參數會新增至 COPY 命令，以簡化幾何。

```

COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
SIMPLIFY AUTO
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';

INFO: Load into table 'norway_water' completed, 1989196 record(s) loaded successfully.

```

若要檢視已簡化的列和幾何，請查詢 SVL_SPATIAL_SIMPLIFY。

```

SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
query | line_number | maximum_tolerance | initial_size | simplified | final_size |
final_tolerance
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
      20 |      1184704 |                -1 |      1513736 | t         |      1008808 |
1.276386653895e-05
      20 |      1664115 |                -1 |      1233456 | t         |      1023584 |
6.11707814796635e-06

```

使用 SIMPLIFY AUTO max_tolerance，且容差低於自動計算的容差，可能會導致擷取錯誤。在此情況下，請使用 MAXERROR 來忽略錯誤。

```

COPY norway_water FROM 's3://bucket_name/shapefiles/norway/gis_osm_water_a_free_1.shp'
FORMAT SHAPEFILE
SIMPLIFY AUTO 1.1E-05
MAXERROR 2
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';

INFO: Load into table 'norway_water' completed, 1989195 record(s) loaded successfully.
INFO: Load into table 'norway_water' completed, 1 record(s) could not be loaded.
Check 'stl_load_errors' system table for details.

```

SVL_SPATIAL_SIMPLIFY 再次查詢以識別 COPY 無法載入的記錄。


```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
query | line_number | maximum_tolerance | initial_size | simplified | final_size |
final_tolerance
-----+-----+-----+-----+-----+-----+-----
+-----+
    29 |      1184704 |          1.1e-05 |      1513736 | f          |           0 |
           0
    29 |      1664115 |          1.1e-05 |      1233456 | t          |      794432 |
    1.1e-05
```

在這個範例中，第一筆記錄無法容納，因此 `simplified` 欄顯示為 `false`。第二筆記錄在給定的公差範圍內載入。但是，最終大小大於使用自動計算的公差而不指定最大公差。

從壓縮的 Shapefile 載入

Amazon Redshift COPY 支援從壓縮形狀檔擷取資料。所有 Shapefile 元件必須具有相同的 Amazon S3 字首和相同的壓縮字尾。例如，假設您想要從上述範例中載入資料。在此情況下，`gis_osm_water_a_free_1.shp.gz`、`gis_osm_water_a_free_1.dbf.gz` 和 `gis_osm_water_a_free_1.shx.gz` 檔案必須共用相同的 Amazon S3 目錄。COPY 命令需要 GZIP 選項，且 FROM 子句必須指定正確的壓縮檔案，如下所示。

```
COPY norway_natural FROM 's3://bucket_name/shapefiles/norway/compressed/
gis_osm_natural_free_1.shp.gz'
FORMAT SHAPEFILE
GZIP
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural' completed, 83891 record(s) loaded successfully.
```

以不同的欄順序將資料載入到資料表中

如果您的資料表沒有將 `GEOMETRY` 作為第一欄，則可以使用欄映射將欄映射到目標資料表。例如，建立一個將 `osm_id` 指定為第一欄的資料表。

```
CREATE TABLE norway_natural_order (
    osm_id BIGINT,
    wkb_geometry GEOMETRY,
    code INT,
    fclass VARCHAR,
    name VARCHAR);
```

然後使用欄映射擷取 Shapefile。

```
COPY norway_natural_order(wkb_geometry, osm_id, code, fclass, name)
FROM 's3://bucket_name/shapefiles/norway/gis_osm_natural_free_1.shp'
FORMAT SHAPEFILE
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/MyRoleName';
INFO: Load into table 'norway_natural_order' completed, 83891 record(s) loaded
successfully.
```

將資料載入至具有地理欄的資料表

如果您有一個包含 GEOGRAPHY 欄的資料表，則首先擷取至 GEOMETRY 欄中，然後將物件轉換為 GEOGRAPHY 物件。例如，將 Shapefile 複製到 GEOMETRY 欄之後，請更改資料表以新增 GEOGRAPHY 資料類型的欄。

```
ALTER TABLE norway_natural ADD COLUMN wkb_geography GEOGRAPHY;
```

然後將幾何轉換為地理。

```
UPDATE norway_natural SET wkb_geography = wkb_geometry::geography;
```

您也可以選擇捨棄 GEOMETRY 欄。

```
ALTER TABLE norway_natural DROP COLUMN wkb_geometry;
```

COPY 具有 NOLOAD 選項的 命令

若要在實際載入資料之前驗證資料檔案，請使用 NOLOAD 選項搭配 COPY 命令。Amazon Redshift 會剖析輸入檔案，並顯示發生的任何錯誤。下列範例使用 NOLOAD 選項，且資料表中不會實際載入任何資料列。

```
COPY public.zipcode1
FROM 's3://amzn-s3-demo-bucket/mydata/zipcode.csv'
DELIMITER ';'
IGNOREHEADER 1 REGION 'us-east-1'
NOLOAD
CREDENTIALS 'aws_iam_role=arn:aws:iam::123456789012:role/myRedshiftRole';

Warnings:
Load into table 'zipcode1' completed, 0 record(s) loaded successfully.
```

COPY 具有多位元組分隔符號的 命令，以及 ENCODING 選項

下列範例LATIN1會從包含多位元組資料的 Amazon S3 檔案載入。COPY 命令會以八進位形式指定分隔符號\302\246\303\254，以分隔輸入檔案中編碼為 ISO-8859-1 的欄位。若要在 UTF-8 中指定相同的分隔符號，請指定 DELIMITER '|'。

```
COPY latin1
FROM 's3://amzn-s3-demo-bucket/multibyte/myfile'
IAM_ROLE 'arn:aws:iam::123456789012:role/myRedshiftRole'
DELIMITER '\302\246\303\254'
ENCODING ISO88591
```

CREATE DATABASE

建立新的資料庫。

若要建立資料庫，您必須是超級使用者或具有 CREATEDB 權限。若要建立與零ETL整合相關聯的資料庫，您必須是超級使用者或同時擁有 CREATEDB 和 CREATEUSER 權限。

您無法在交易區塊CREATEDATABASE內執行 (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

語法

```
CREATE DATABASE database_name
[ { [ WITH ]
  [ OWNER [=] db_owner ]
  [ CONNECTION LIMIT { limit | UNLIMITED } ]
  [ COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE } ]
  [ ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT } ]
}
| { [ WITH PERMISSIONS ] FROM DATASHARE datashare_name ] OF [ ACCOUNT account_id ]
NAMESPACE namespace_guid }
| { FROM { { ARN '<arn>' } { WITH DATA CATALOG SCHEMA '<schema>' | WITH NO DATA
CATALOG SCHEMA } }
| { INTEGRATION '<integration_id>' [ DATABASE '<source_database>' ] [ SET
{REFRESH_INTERVAL <interval>} ] } }
| { IAM_ROLE {default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' } }
]
```

參數

database_name

新資料庫的名稱。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

WITH

選用的關鍵字。

OWNER

指定資料庫擁有者。

=

選用字元。

db_owner

資料庫擁有者的使用者名稱。

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字來允許並行連線的數量上限。另外也可能限制每位使用者的連線數目。如需詳細資訊，請參閱 [CREATE USER](#)。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 [STV_SESSIONS](#) 系統畫面。

Note

如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

COLLATE { CASE_SENSITIVE | CASE_INSENSITIVE }

指定字串搜尋或比較是 CASE_SENSITIVE 或 CASE_ 的子句 INSENSITIVE。預設值為 CASE_SENSITIVE。

ISOLATION LEVEL { SERIALIZABLE | SNAPSHOT }

指定對資料庫執行查詢時所用隔離層級的子句。

- SERIALIZABLE 隔離 – 為並行交易提供完整的序列化功能。如需詳細資訊，請參閱 [可序列化隔離](#)。

- **SNAPSHOT 隔離** – 提供隔離層級，防止更新和刪除衝突。這是佈建叢集或無伺服器命名空間中建立的資料庫的預設值。

您可以檢式資料庫正在執行的並行模型，如下所示：

- 查詢 `STV_DB_ISOLATION_LEVEL` 目錄檢視。如需詳細資訊，請參閱 [STV__ISOLATION LEVEL](#)。

```
SELECT * FROM stv_db_isolation_level;
```

- 查詢 `PG_DATABASE_INFO` 檢視。

```
SELECT datname, datconfig FROM pg_database_info;
```

每個資料庫的隔離層級都會顯示在 `concurrency_model` 索引鍵旁邊。的值1表示 SNAPSHOT。的值2表示 SERIALIZABLE。

在 Amazon Redshift 資料庫中，SERIALIZABLE和 SNAPSHOT隔離都是可序列化隔離層級的類型。也就是說，根據 SQL 標準，會阻止骯髒的讀取、不可重複的讀取和假體讀取。這兩個隔離層級可保證在交易開始時，交易可在存在之資料的快照上操作，而且其他交易都不能變更該快照。但是，SNAPSHOT隔離不提供完整的序列化功能，因為它無法防止不同資料表列上的寫入偏移插入和更新。

下列案例說明使用SNAPSHOT隔離層級的寫入偏移更新。名為 Numbers 的資料表包含名為 digits 的資料欄，其中包含 0 和 1 值。每個使用者的UPDATE陳述式不會與其他使用者重疊。但是，0 和 1 值會交換。SQL 其執行的 遵循此時間表並包含這些結果：

時間	使用者 1 動作	使用者 2 動作
1	BEGIN;	
2		BEGIN;
3	SELECT * FROM 數字;	

時間	使用者 1 動作	使用者 2 動作
	<pre>digits - ----- 0 1</pre>	
4		<p>SELECT * FROM數字 ;</p> <pre>digits ----- 0 1</pre>
5	<p>UPDATE SET數字 = 0 WHERE 位數 = 1 ;</p>	
6	<p>SELECT * FROM 數字 ;</p> <pre>digits - ----- 0 0</pre>	
7	<p>COMMIT</p>	
8		<p>更新數字SET數字 = 1 WHERE位數 = 0 ;</p>

時間	使用者 1 動作	使用者 2 動作
9		SELECT * FROM 數字 ; <pre> digits ----- 1 1 </pre>
10		COMMIT;
11	SELECT * FROM 數字 ; <pre> digits - ----- 1 0 </pre>	
12		SELECT * FROM 數字 ; <pre> digits ----- 1 0 </pre>

如果使用可序列化隔離執行相同的案例，Amazon Redshift 會因為可序列化違規而終止使用者 2，並傳回錯誤 1023。如需詳細資訊，請參閱[如何修正可序列化隔離錯誤](#)。在這種情況下，只有使用者 1 可以成功提交。並非所有工作負載都需要要求可序列化隔離，在這種情況下，快照隔離就足以成為資料庫的目標隔離層級。

```
FROM ARN '<ARN>'
```

ARN 用來建立 AWS Glue 資料庫的資料庫。

```
{ DATA CATALOG SCHEMA '<schema>' | WITH NO DATA CATALOG SCHEMA }
```

Note

此參數只有在您的CREATEDATABASE命令也使用 FROM ARN 參數時才適用。

指定是否使用結構描述建立資料庫，以協助存取 AWS Glue Data Catalog中的物件。

```
FROM INTEGRATION '<integration_id>' DATABASE 【 '<source_database>' 】 【 SET  
{ REFRESH_INTERVAL <interval> } 】
```

指定是否使用零ETL整合識別符建立資料庫。您可以從 integration_id SVV_INTEGRATION 系統檢視擷取。對於 Aurora PostgreSQL ETL零整合，您也需要指定名稱，該source_database名稱也可以從 SVV_ 擷取INTEGRATION。REFRESH_SET INTERVAL 子句會設定大約的時間間隔，以秒為單位，將資料從零ETL來源重新整理至目標資料庫。對於來源類型為 Aurora MySQL、Aurora Postgre 或 RDS My 的零ETL整合SQL，預設值為零（0）秒SQL。

如需範例，請參閱[建立資料庫以接收零ETL整合的結果](#)。如需使用零ETL整合建立資料庫的詳細資訊，請參閱 [Amazon Redshift 管理指南 中的在 Amazon Redshift 中建立目的地資料庫](#)。

```
IAM_ROLE { 預設 | 'SESSION' | 'arn : aws : iam : <AWS ##-id> : 角色/<role-name>' }
```

Note

此參數只有在您的CREATEDATABASE命令也使用 FROM ARN 參數時才適用。

如果您在執行CREATEDATABASE命令時指定與叢集相關聯的IAM角色，Amazon Redshift 會在資料庫上執行查詢時使用該角色的憑證。

指定default關鍵字表示使用設定為預設並與叢集相關聯的IAM角色。

如果您使用聯合身分連線到 Amazon Redshift 叢集，並從使用此命令建立的外部結構描述存取資料表，請使用 'SESSION'。如需使用聯合身分的範例，請參閱[使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取](#)，其中會說明如何設定聯合身分。

針對叢集用於身分驗證和授權IAM的角色，使用 Amazon Resource Name (ARN)。至少，IAM角色必須具有許可，才能在要存取的 Amazon S3 儲存貯體上執行LIST操作，以及在儲存貯體包含的

Amazon S3 物件上執行GET操作。若要進一步了解使用 建立資料庫時使用 IAM_ROLE AWS Glue Data Catalog 進行資料共用，請參閱以[取用者身分使用 Lake Formation 管理的資料共用](#)。

下列顯示單一 的 IAM_ROLE 參數字串語法ARN。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

您可以連結角色，讓您的叢集擔任另一個IAM角色，可能屬於另一個帳戶。您最多可以鏈結 10 個角色。如需詳細資訊，請參閱[Amazon Redshift Spectrum 中的鏈結IAM角色](#)。

若要擔任此IAM角色，請連接類似下列的IAM許可政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessSecret",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-rds-secret-VNenFy"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetRandomPassword",
        "secretsmanager:ListSecrets"
      ],
      "Resource": "*"
    }
  ]
}
```

如需建立IAM角色以搭配聯合查詢使用的步驟，請參閱 [建立密碼和IAM角色以使用聯合查詢](#)。

Note

不要在鏈結的角色清單中包含空格。

以下顯示鏈結三個角色的語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

CREATE DATABASE 搭配資料共用使用的語法

下列語法說明從資料共用建立資料庫的CREATEDATABASE命令，用於在相同 AWS 帳戶中共用資料。

```
CREATE DATABASE database_name  
[ [ WITH PERMISSIONS ] FROM DATASHARE datashare_name ] OF [ ACCOUNT account_id ]  
NAMESPACE namespace_guid
```

下列語法說明從資料共用建立資料庫的CREATEDATABASE命令，用於跨 AWS 帳戶共用資料。

```
CREATE DATABASE database_name  
[ [ WITH PERMISSIONS ] FROM DATASHARE datashare_name ] OF ACCOUNT account_id  
NAMESPACE namespace_guid
```

CREATE DATABASE 搭配資料共用使用的參數

FROM DATASHARE

指出資料共用所在位置的關鍵字。

datashare_name

建立取用者資料庫所用的資料共用名稱。

WITH PERMISSIONS

指定從資料共用建立的資料庫需要物件層級權限，才能存取個別資料庫物件。如果沒有此子句，在資料庫上授予USAGE許可的使用者或角色將自動存取資料庫中的所有資料庫物件。

NAMESPACE namespace_guid

指定資料共用所屬之生產者命名空間的值。

ACCOUNT account_id

指定資料共用所屬之生產者帳戶的值。

用於資料共用CREATEDATABASE的 用量備註

身為資料庫超級使用者，當您使用從 AWS 帳戶中的資料共用CREATEDATABASE建立資料庫時，請指定 NAMESPACE選項。ACCOUNT 選項為選用。當您使用從跨 AWS 帳戶的資料共用CREATEDATABASE建立資料庫時，請同時指定 ACCOUNT和NAMESPACE來自生產者。

您只能為取用者叢集上的一個資料共用建立一個取用者資料庫。您無法建立參照相同資料共用的多個取用者資料庫。

CREATE DATABASE 從 AWS Glue Data Catalog

若要使用資料庫 建立 AWS Glue 資料庫ARN，請在CREATEDATABASE命令ARN中指定。

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA;
```

您也可以選擇性地將值提供給 IAM_ROLE 參數。如需參數和接受值的相關資訊，請參閱[參數](#)。

以下是示範如何使用ARNIAM角色從 建立資料庫的範例。

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA  
IAM_ROLE <iam-role-arn>
```

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH NO DATA CATALOG SCHEMA  
IAM_ROLE default;
```

您也可以使用 DATA CATALOG 建立資料庫SCHEMA。

```
CREATE DATABASE sampledb FROM ARN <glue-database-arn> WITH DATA CATALOG SCHEMA  
<sample_schema> IAM_ROLE default;
```

建立資料庫以接收零ETL整合的結果

若要使用零ETL整合身分建立資料庫，請在CREATEDATABASE命令integration_id中指定。

```
CREATE DATABASE destination_db_name FROM INTEGRATION 'integration_id';
```

例如，首先，從 SVV_ 擷取整合 IDINTEGRATION ；

```
SELECT integration_id FROM SVV_INTEGRATION;
```

然後使用擷取的其中一個整合 ID 來建立接收零ETL整合的資料庫。

```
CREATE DATABASE sampledb FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111';
```

需要零ETL整合來源資料庫時，例如指定。

```
CREATE DATABASE sampledb FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'  
DATABASE 'sourcedb';
```

您也可以設定資料庫的重新整理間隔。例如，若要將來自零ETL整合來源的資料重新整理間隔設定為 7,200 秒：

```
CREATE DATABASE myacct_mysql FROM INTEGRATION 'a1b2c3d4-5678-90ab-cdef-EXAMPLE11111'  
SET REFRESH_INTERVAL 7200;
```

查詢 SVV_INTEGRATION 目錄檢視，以取得零ETL整合的相關資訊，例如 integration_id、target_database、source、fresh_interval 等。

```
SELECT * FROM svv_integration;
```

CREATE DATABASE 限制

Amazon Redshift 會針對資料庫強制執行以下限制：

- 每個叢集最多 60 個使用者定義的資料庫。
- 資料庫名稱最多 127 個位元組。
- 資料庫名稱不能是保留字。

資料庫定序

Collation 是一組規則，定義資料庫引擎如何比較和排序 中的字元類型資料SQL。不區分大小寫的定序是最常用的定序。Amazon Redshift 使用不區分大小寫的定序來協助從其他資料倉儲進行移轉。透過

不區分大小寫定序的原生支援，Amazon Redshift 會繼續使用重要的調整或最佳化方法，例如分佈索引鍵、排序索引鍵或範圍限制掃描。

COLLATE 子句會指定資料庫中所有 CHAR 和 VARCHAR 資料欄的預設定序。如果指定 CASE_INSENSITIVE，則所有 CHAR 或 VARCHAR 資料欄都會使用不區分大小寫的定序。如需有關定序的資訊，請參閱 [定序序列](#)。

在不區分大小寫的資料欄中插入或擷取的資料將會保留其原始大小寫。但是所有基於比較的字串操作 (包括排序和分組) 也都不區分大小寫。模式比對操作，例如 LIKE 述詞、類似 和規則表達式函數，也不區分大小寫。

下列 SQL 操作支援適用的定序語義：

- 比較運算子：=、<>、<、<=、>、>=。
- LIKE 運算子
- ORDER BY 子句
- GROUP BY 子句
- 使用字串比較的彙總函數，例如 MIN 和 MAX LISTAGG
- 視窗函數，例如 PARTITION BY 子句和 ORDER BY 子句
- 純量函數 greatest () 和 least ()、STRPOS ()、REGEXP_COUNT ()、REGEXP_REPLACE ()、REGEXP_INSTR ()、_
- 相異子句
- UNION、INTERSECT 和 EXCEPT
- IN LIST

對於外部查詢，包括 Amazon Redshift Spectrum 和 Aurora PostgreSQL 聯合查詢，VARCHAR 或 CHAR 欄的定序與目前的資料庫層級定序相同。

下列範例會查詢 Amazon Redshift Spectrum 資料表：

```
SELECT ci_varchar FROM spectrum.test_collation
WHERE ci_varchar = 'AMAZON';

ci_varchar
-----
amazon
Amazon
```

```
AMAZON
AmaZon
(4 rows)
```

如需如何使用資料庫定序建立資料表的資訊，請參閱 [CREATE TABLE](#)。

如需COLLATE函數的資訊，請參閱 [COLLATE 函數](#)。

資料庫定序限制

以下是在 Amazon Redshift 中使用資料庫定序時的限制：

- 所有系統資料表或檢視 (包括 PG 目錄資料表和 Amazon Redshift 系統資料表) 都會區分大小寫。
- 當取用者資料庫和生產者資料庫具有不同的資料庫層級定序時，Amazon Redshift 不支援跨資料庫和跨叢集查詢。
- Amazon Redshift 在僅限領導節點查詢中不支援不區分大小寫的定序。

下列範例顯示不支援的不區分大小寫查詢，以及 Amazon Redshift 傳送的錯誤：

```
SELECT collate(username, 'case_insensitive') FROM pg_user;
ERROR: Case insensitive collation is not supported in leader node only query.
```

- Amazon Redshift 不支援區分大小寫和不區分大小寫的資料欄之間進行互動，例如比較、函數、聯結或設定操作。

下列範例顯示區分大小寫和不區分大小寫的資料行互動時的錯誤：

```
CREATE TABLE test
  (ci_col varchar(10) COLLATE case_insensitive,
   cs_col varchar(10) COLLATE case_sensitive,
   cint int,
   cbigint bigint);
```

```
SELECT ci_col = cs_col FROM test;
ERROR: Query with different collations is not supported yet.
```

```
SELECT concat(ci_col, cs_col) FROM test;
ERROR: Query with different collations is not supported yet.
```

```
SELECT ci_col FROM test UNION SELECT cs_col FROM test;
```

```
ERROR: Query with different collations is not supported yet.
```

```
SELECT * FROM test a, test b WHERE a.ci_col = b.cs_col;  
ERROR: Query with different collations is not supported yet.
```

```
Select Coalesce(ci_col, cs_col) from test;  
ERROR: Query with different collations is not supported yet.
```

```
Select case when cint > 0 then ci_col else cs_col end from test;  
ERROR: Query with different collations is not supported yet.
```

- Amazon Redshift 不支援SUPER資料類型的定序。不支援在不區分大小寫的資料庫中建立SUPER資料欄，以及 SUPER和不區分大小寫的資料欄之間的互動。

下列範例會建立資料表，並以 SUPER作為不區分大小寫的資料庫中的資料類型：

```
CREATE TABLE super_table (a super);  
ERROR: SUPER column is not supported in case insensitive database.
```

下列範例使用不區分大小寫的字串查詢資料，並與SUPER資料進行比較：

```
CREATE TABLE test_super_collation  
(s super, c varchar(10) COLLATE case_insensitive, i int);
```

```
SELECT s = c FROM test_super_collation;  
ERROR: Coercing from case insensitive string to SUPER is not supported.
```

若要讓這些查詢正常運作，請使用 COLLATE函數來轉換一欄的定序，以符合另一欄。如需詳細資訊，請參閱[COLLATE 函數](#)。

範例

建立資料庫

下列範例會建立名為 的資料庫TICKIT，並將所有權授予使用者 DWUSER。

```
create database tickit  
with owner dwuser;
```

若要檢視資料庫的詳細資訊，請查詢 PG_DATABASE_INFO 目錄表。

```
select datname, datdba, datconlimit
from pg_database_info
where datdba > 1;
```

datname	datdba	datconlimit
admin	100	UNLIMITED
reports	100	100
ticket	100	100

下列範例會建立名為 **sampledb** 的資料庫，其中包含 SNAPSHOT 隔離層級。

```
CREATE DATABASE sampledb ISOLATION LEVEL SNAPSHOT;
```

下列範例會從資料共用 saleshare 中建立資料庫 sales_db。

```
CREATE DATABASE sales_db FROM DATASHARE saleshare OF NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

資料庫定序範例

建立不區分大小寫的資料庫

下列範例會建立 sampledb 資料庫、建立 T1 資料表，以及將資料插入 T1 資料表。

```
create database sampledb collate case_insensitive;
```

連線至您剛使用 SQL 用戶端建立的新資料庫。使用 Amazon Redshift 查詢編輯器 v2 時，請在編輯器中選擇 sampledb。使用 時 RSQL，請使用如下所示的命令。

```
\connect sampledb;
```

```
CREATE TABLE T1 (
  col1 Varchar(20) distkey sortkey
);
```

```
INSERT INTO T1 VALUES ('bob'), ('john'), ('Mary'), ('JOHN'), ('Bob');
```


然後查詢會尋找包含 John 結果。

```
SELECT * FROM T1 WHERE col1 = 'John';

col1
-----
john
JOHN
(2 row)
```

以不區分大小寫的順序排序

下列範例顯示資料表 T1 的不區分大小寫排序。Bob 和 bob 或 John 和 john 的排序是不確定的，因為其在不區分大小寫的資料欄中是相等的。

```
SELECT * FROM T1 ORDER BY 1;

col1
-----
bob
Bob
JOHN
john
Mary
(5 rows)
```

同樣地，下列範例顯示 GROUP BY 子句不區分大小寫的排序。Bob 和 bob 是同等的，屬於同一組。結果中會顯示哪一個是不確定的。

```
SELECT col1, count(*) FROM T1 GROUP BY 1;

col1 | count
-----+-----
Mary | 1
bob  | 2
JOHN | 2
(3 rows)
```

在不區分大小寫的資料欄上使用視窗函數進行查詢

下列範例會在不區分大小寫的資料欄上查詢視窗函數。

```
SELECT col1, rank() over (ORDER BY col1) FROM T1;
```

```
col1 | rank
-----+-----
bob  |    1
Bob  |    1
john |    3
JOHN |    3
Mary |    5
(5 rows)
```

使用DISTINCT關鍵字查詢

下列範例會查詢具有 DISTINCT 關鍵字的 T1 資料表。

```
SELECT DISTINCT col1 FROM T1;
```

```
col1
-----
bob
Mary
john
(3 rows)
```

使用 UNION 子句查詢

下列範例顯示資料表 T1 和 UNION 的結果 T2。

```
CREATE TABLE T2 AS SELECT * FROM T1;
```

```
SELECT col1 FROM T1 UNION SELECT col1 FROM T2;
```

```
col1
-----
john
bob
Mary
(3 rows)
```

CREATE DATASHARE

在目前資料庫中建立新的資料共用。此資料共用的擁有者是 CREATE DATASHARE 命令的發行者。

Amazon Redshift 會將每個資料共用與單一 Amazon Redshift 資料庫相關聯。您只可以在資料共用中新增相關聯資料庫中的物件。您可以在同一個 Amazon Redshift 資料庫上建立多個資料共用。

如需有關資料共用的資訊，請參閱 [管理資料共用工作](#)。

若要檢視有關資料庫的資訊，請使用 [SHOW DATASHARES](#)。

所需權限

以下是 CREATE 的必要權限 DATASHARE：

- 超級使用者
- 具有 CREATEDATASHARE 權限的使用者
- 資料庫擁有者

語法

```
CREATE DATASHARE datashare_name  
[[SET] PUBLICACCESSIBLE [=] TRUE | FALSE ];
```

參數

datashare_name

資料共用的名稱。資料共用名稱必須是叢集命名空間中唯一的名稱。

[[SET] PUBLICACCESSIBLE]

該子句會指定資料共用是否可以共用至可公開存取的叢集。

SET PUBLICACCESSIBLE 的預設值為 FALSE。

使用須知

依預設，資料共用的擁有者只能有共用，不能有共用中的物件。

只有超級使用者和資料庫擁有者可以使用 `CREATEDATASHARE`，並將 `ALTER` 權限委派給其他使用者或群組。

範例

下列範例會建立 `salesshare` 資料共用。

```
CREATE DATASHARE salesshare;
```

下列範例會建立 AWS Data Exchange 管理的 `demoshare` 資料共用。

```
CREATE DATASHARE demoshare SET PUBLICACCESSIBLE TRUE, MANAGEDBY ADX;
```

CREATE EXTERNAL FUNCTION

根據 AWS Lambda Amazon Redshift 建立純量使用者定義函數 (UDF)。如需 Lambda 使用者定義函數的相關資訊，請參閱 [Scalar Lambda UDFs](#)。

所需權限

以下是 `CREATE EXTERNAL FUNCTION` 的必要權限：

- 超級使用者
- 具有 `CREATE [OR] REPLACE EXTERNALFUNCTION` 權限的使用者

語法

```
CREATE [ OR REPLACE ] EXTERNAL FUNCTION external_fn_name ( [data_type] [, ...] )  
RETURNS data_type  
{ VOLATILE | STABLE }  
LAMBDA 'lambda_fn_name'  
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }  
RETRY_TIMEOUT milliseconds  
MAX_BATCH_ROWS count  
MAX_BATCH_SIZE size [ KB | MB ];
```

參數

或 REPLACE

該子句會指定已有相同名稱和輸入引數資料類型 (或簽章) 的函數存在時，取代現有函數。您可以將函數取代為定義一組相同資料類型的新函數。您必須是超級使用者才能取代函數。

如果您定義的函數與現有函數同名，但簽章不同，則會建立新函數。換言之，函數名稱將會過載。如需詳細資訊，請參閱[多載函數名稱](#)。

external_fn_name

外部函數的名稱。如果您指定結構描述名稱 (例如 myschema.myfunction)，則會使用指定的結構描述建立函數。否則，函數會在目前結構描述中建立。如需有效名稱的相關資訊，請參閱[名稱與識別碼](#)。

建議您使用 在所有UDF名稱前面加上 f_。Amazon Redshift 會保留UDF名稱的f_字首。使用f_字首，有助於確保UDF您的姓名不會與現在或未來的 Amazon Redshift 的任何內建SQL函數名稱衝突。如需詳細資訊，請參閱[防止UDF命名衝突](#)。

data_type

輸入引數的資料類型。如需詳細資訊，請參閱[資料類型](#)。

RETURNS data_type

函數所傳回值的資料類型。RETURNS 資料類型可以是任何標準 Amazon Redshift 資料類型。如需詳細資訊，請參閱[Python UDF 資料類型](#)。

VOLATILE | STABLE

通知查詢最佳化工具有關函數的波動情形。

若要得到最理想的最佳化，請將函數標示為最嚴格的有效波動類別。從最低嚴格程度開始，依嚴格程度排列的波動類別如下所示：

- VOLATILE
- STABLE

VOLATILE

假設引數相同，即使是針對單一陳述式中的資料列，函數也可能在後續呼叫中傳回不同的結果。查詢最佳化工具無法假設波動函數的行為。使用波動函數的查詢必須重新評估每個輸入的函數。

STABLE

假設引數相同，函數一定會針對單一陳述式內處理的連續呼叫傳回相同結果。在不同陳述式中呼叫函數時，函數可能傳回不同結果。此類別可這麼做，所以最佳化工具可以減少在單個陳述式中調用函數的次數。

請注意，如果選擇的嚴格性對函數無效，則最佳化工具可能會有根據此嚴格性而略過某些呼叫的風險。這可能會導致不正確的結果集。

Lambda 目前不支援 IMMUTABLE子句UDFs。

LAMBDA 'lambda_fn_name'

Amazon Redshift 呼叫的函數名稱。

如需建立 AWS Lambda 函數的步驟，請參閱 AWS Lambda 開發人員指南 中的[使用 主控台建立 Lambda 函數](#)。

如需 Lambda 函數所需權限的相關資訊，請參閱《AWS Lambda 開發人員指南》中的[AWS Lambda 權限](#)。

IAM_ROLE { 預設 | 'arn : aws : iam : : <AWS #-id> : 角色/<role-name>' }

使用預設關鍵字讓 Amazon Redshift 在CREATEEXTERNALFUNCTION命令執行時，使用設定為預設並與叢集相關聯的IAM角色。

針對叢集用於身分驗證和授權IAM的角色，請使用 Amazon Resource Name (ARN)。CREATE EXTERNAL FUNCTION 命令有權透過此IAM角色叫用 Lambda 函數。如果您的叢集具有現有IAM角色，並具有叫用已連接 Lambda 函數的許可，您可以取代角色的 ARN。如需詳細資訊，請參閱[設定 Lambda 的授權參數 UDFs](#)。

下列顯示 IAM_ROLE 參數的語法。

```
IAM_ROLE 'arn:aws:iam::aws-account-id:role/role-name'
```

RETRY_TIMEOUT 毫秒

Amazon Redshift 用於重試退避延遲的總時間 (以毫秒為單位)。

Amazon Redshift 不會針對任何失敗的查詢立即重試，而是執行退選，並在重試之間等待一段時間。然後，Amazon Redshift 會重試重新執行失敗查詢的請求，直到所有延遲的總和等於或超過您指定的 RETRY_TIMEOUT 值。預設值為 20,000 毫秒。

調用 Lambda 函數時，Amazon Redshift 會針對接收到錯誤 (例如 TooManyRequestsException、EC2ThrottledException 和 ServiceException) 的查詢重試。

您可以將 `RETRY_TIMEOUT` 參數設定為 0 毫秒，以防止 Lambda 的任何重試UDF。

MAX_BATCH_ROWS 計數

Amazon Redshift 在單一批次請求中針對單一 Lambda 調用傳送的資料列數目上限。

此參數的最小值為 1。最大值為 `INT_MAX`或 2, 147, 483, 647。

此為選用參數。預設值為 `INT_MAX`或 2, 147, 483, 647。

MAX_BATCH_SIZE 大小 【KB | MB】

Amazon Redshift 在單一批次請求中針對單一 Lambda 調用傳送的資料承載大小上限。

此參數的最小值為 1 KB。最大值為 5 MB。

此參數的預設值為 5 MB。

您可以選擇 KB 和 MB。如果未設定測量單位，Amazon Redshift 會預設為使用 KB。

使用須知

當您建立 Lambda 時，請考慮下列事項UDFs：

- 輸入引數上的 Lambda 函數調用順序不是固定或可保證的。其可能會因執行查詢的執行個體而有所不同，具體取決於叢集組態。
- 函數不能保證一次且僅一次地套用到每個輸入引數。Amazon Redshift 與 之間的互動 AWS Lambda 可能會導致具有相同輸入項目的重複呼叫。

範例

以下是使用純量 Lambda 使用者定義函數 () 的範例UDFs。

使用 Node.js Lambda 函數的 Scalar Lambda UDF範例

下列範例會建立名為 `exfunc_sum` 的外部函數，該函數會接受兩個整數做為輸入引數。該函數會傳回作為整數輸出的總和。要呼叫的 Lambda 函數名稱為 `lambda_sum`。用於此 Lambda 函數的語言是 Node.js 12.x。請務必指定IAM角色。此範例使用 `'arn:aws:iam::123456789012:user/johndoe'` 作為IAM角色。

```
CREATE EXTERNAL FUNCTION exfunc_sum(INT,INT)
RETURNS INT
VOLATILE
```

```
LAMBDA 'lambda_sum'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';
```

Lambda 函數會接受請求承載並逐一查看每一列。單列中的所有值都會加入以計算該列的總和，而這會保存在回應陣列中。結果陣列中的資料列數會與請求承載中接收的資料列數相似。

JSON 回應承載必須在 '結果' 欄位中具有結果資料，才能讓外部函數識別。在傳送至 Lambda 函數的請求中，引數欄位會包含資料承載。在批次處理請求的情況下，資料承載中可以有多個資料列。下列 Lambda 函數會逐一查看請求資料承載中的所有資料列。也會單獨逐一查看單一資料列中的所有值。

```
exports.handler = async (event) => {
  // The 'arguments' field in the request sent to the Lambda function contains the
  data payload.
  var t1 = event['arguments'];

  // 'len(t1)' represents the number of rows in the request payload.
  // The number of results in the response payload should be the same as the number
  of rows received.
  const resp = new Array(t1.length);

  // Iterating over all the rows in the request payload.
  for (const [i, x] of t1.entries())
  {
    var sum = 0;
    // Iterating over all the values in a single row.
    for (const y of x) {
      sum = sum + y;
    }
    resp[i] = sum;
  }
  // The 'results' field should contain the results of the lambda call.
  const response = {
    results: resp
  };
  return JSON.stringify(response);
};
```

下列範例會呼叫具有常值的外部函數。

```
select exfunc_sum(1,2);
exfunc_sum
-----
```



```
3
(1 row)
```

下列範例會建立名為 `t_sum` 的資料表，其中包含整數資料類型的兩個資料欄 `c1` 和 `c2`，並插入兩列資料。然後透過傳遞此資料表的資料欄名稱來呼叫外部函數。這兩個資料表資料列會在請求承載中以批次請求的形式傳送，以做為單一 Lambda 調用。

```
CREATE TABLE t_sum(c1 int, c2 int);
INSERT INTO t_sum VALUES (4,5), (6,7);
SELECT exfunc_sum(c1,c2) FROM t_sum;
  exfunc_sum
-----
          9
         13
(2 rows)
```

使用 `RETRY_TIMEOUT` 屬性的 Scalar Lambda UDF 範例

在下一節中，您可以找到如何在 Lambda 中使用 `RETRY_TIMEOUT` 屬性的範例 UDFs。

AWS Lambda 函數具有您可以為每個函數設定的並行限制。如需並行限制的詳細資訊，請參閱 AWS Lambda 開發人員指南中的[管理 Lambda 函數的並行](#)，以及 AWS 運算部落格上的[管理 AWS Lambda 函數並行](#)後文。

當 Lambda 提供的請求數量 UDF 超過並行限制時，新請求會收到 `TooManyRequestsException` 錯誤。Lambda UDF 會重試此錯誤，直到傳送至 Lambda 函數的請求之間的所有延遲總和等於或超過您設定的 `RETRY_TIMEOUT` 值。預設 `RETRY_TIMEOUT` 值為 20,000 毫秒。

下列範例會使用名為 `exfunc_sleep_3` 的 Lambda 函數。該函數會接受請求承載、逐一查看每一個資料列，並將輸入轉換為大寫。然後休眠 3 秒鐘後傳回結果。用於此 Lambda 函數的語言是 Python 3.8。

結果陣列中的資料列數會與請求承載中接收的資料列數相似。JSON 回應承載必須在 `results` 欄位中具有結果資料，才能由外部函數識別。在傳送至 Lambda 函數的請求中，`arguments` 欄位會包含資料承載。在批次處理請求的情況下，資料承載中可以有多個資料列。

此函數的並行限制在預留並行中特別設為 1，以示範 `RETRY_TIMEOUT` 屬性的使用。當屬性設定為 1 時，Lambda 函數一次只能處理一個請求。

```
import json
import time
```

```
def lambda_handler(event, context):
    t1 = event['arguments']
    # 'len(t1)' represents the number of rows in the request payload.
    # The number of results in the response payload should be the same as the number of
    rows received.
    resp = [None]*len(t1)

    # Iterating over all rows in the request payload.
    for i, x in enumerate(t1):
        # Iterating over all the values in a single row.
        for j, y in enumerate(x):
            resp[i] = y.upper()

    time.sleep(3)
    ret = dict()
    ret['results'] = resp
    ret_json = json.dumps(ret)
    return ret_json
```

接下來，另外兩個範例說明 `RETRY_TIMEOUT` 屬性。他們各自叫用單一 Lambda UDF。叫用 Lambda 時UDF，每個範例都會執行相同的SQL查詢，以同時UDF從兩個並行資料庫工作階段叫用 Lambda。叫用 Lambda 的第一個查詢由 UDF提供時UDF，第二個查詢會收到 `TooManyRequestsException` 錯誤。此結果是因為您在 中將預留並行設定為 UDF 1。如需如何為 Lambda 函數設定保留並行的相關資訊，請參閱[設定保留並行](#)。

第一個範例如下，將 Lambda 的 `RETRY_TIMEOUT` 屬性UDF設定為 0 毫秒。如果 Lambda 請求從 Lambda 函數收到任何例外狀況，Amazon Redshift 不會進行任何重試。發生此結果是因為 `RETRY_TIMEOUT` 屬性設定為 0。

```
CREATE OR REPLACE EXTERNAL FUNCTION exfunc_upper(varchar)
RETURNS varchar
VOLATILE
LAMBDA 'exfunc_sleep_3'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test'
RETRY_TIMEOUT 0;
```

將 `RETRY_TIMEOUT` 設定為 0 時，您可以從不同的資料庫工作階段執行下列兩個查詢，以查看不同的結果。

使用 Lambda 的第一個SQL查詢已成功UDF執行。

```
select exfunc_upper('Varchar');
```

```

exfunc_upper
-----
VARCHAR
(1 row)

```

同時從不同資料庫工作階段執行的第二個查詢會收到 `TooManyRequestsException` 錯誤。

```

select exfunc_upper('Varchar');
ERROR:  Rate Exceeded.; Exception: TooManyRequestsException; ShouldRetry: 1
DETAIL:
-----
error:  Rate Exceeded.; Exception: TooManyRequestsException; ShouldRetry: 1
code:      32103
context:query:      0
location:  exfunc_client.cpp:102
process:   padbmaster [pid=26384]
-----

```

第二個範例如下，將 Lambda 的 `RETRY_TIMEOUT` 屬性UDF設定為 3,000 毫秒。即使第二個查詢同時執行，Lambda UDF 仍會重試，直到總延遲為 3,000 毫秒為止。因此，兩個查詢都會成功執行。

```

CREATE OR REPLACE EXTERNAL FUNCTION exfunc_upper(varchar)
RETURNS varchar
VOLATILE
LAMBDA 'exfunc_sleep_3'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test'
RETRY_TIMEOUT 3000;

```

將 `RETRY_TIMEOUT` 設定為 3,000 毫秒時，您可以從不同的資料庫工作階段執行下列兩個查詢，以查看相同的結果。

執行 Lambda 的第一個SQL查詢已成功UDF執行。

```

select exfunc_upper('Varchar');
exfunc_upper
-----
VARCHAR
(1 row)

```

第二個查詢會同時執行，Lambda UDF 會重試，直到總延遲為 3,000 毫秒為止。

```

select exfunc_upper('Varchar');

```

```

exfunc_upper
-----
VARCHAR
(1 row)

```

使用 Python Lambda 函數的 Scalar Lambda UDF 範例

下列範例會建立名為 `exfunc_multiplication` 且會乘以數字並傳回整數的外部函數。此範例包含 Lambda 回應中的成功和 `error_msg` 欄位。當乘法結果中有整數溢位，且 `error_msg` 訊息設定為 `Integer multiplication overflow` 時，成功欄位會設定為 `false`。`exfunc_multiplication` 函數採用三個整數作為輸入參數，並傳回總和作為整數輸出。

要呼叫的 Lambda 函數名稱為 `lambda_multiplication`。用於此 Lambda 函數的語言是 Python 3.8。請務必指定 IAM 角色。

```

CREATE EXTERNAL FUNCTION exfunc_multiplication(int, int, int)
RETURNS INT
VOLATILE
LAMBDA 'lambda_multiplication'
IAM_ROLE 'arn:aws:iam::123456789012:role/Redshift-Exfunc-Test';

```

Lambda 函數會接受請求承載並逐一查看每一列。單列中的所有值都會相乘以計算該列的結果，而這會保存在回應清單中。此範例使用預設為 `true` 的布林成功值。如果資料列的乘法結果有整數溢位，則成功值會設定為 `false`。然後迭代循環會中斷。

建立回應承載時，如果成功值為 `false`，則下列 Lambda 函數會在承載中新增 `error_msg` 欄位。也會將錯誤訊息設定為 `Integer multiplication overflow`。如果成功值為 `true`，則結果資料會新增到結果欄位中。結果陣列中的資料列數 (如果有的話) 會與請求承載中接收的資料列數相似。

在傳送至 Lambda 函數的請求中，引數欄位會包含資料承載。在批次處理請求的情況下，資料承載中可以有多個資料列。下列 Lambda 函數會逐一查看請求資料承載中的所有資料列，並個別逐一查看單一資料列中的所有值。

```

import json
def lambda_handler(event, context):
    t1 = event['arguments']
    # 'len(t1)' represents the number of rows in the request payload.
    # The number of results in the response payload should be the same as the number of
    rows received.
    resp = [None]*len(t1)

```

```

# By default success is set to 'True'.
success = True
# Iterating over all rows in the request payload.
for i, x in enumerate(t1):
    mul = 1
    # Iterating over all the values in a single row.
    for j, y in enumerate(x):
        mul = mul*y

    # Check integer overflow.
    if (mul >= 9223372036854775807 or mul <= -9223372036854775808):
        success = False
        break
    else:
        resp[i] = mul
ret = dict()
ret['success'] = success
if not success:
    ret['error_msg'] = "Integer multiplication overflow"
else:
    ret['results'] = resp
ret_json = json.dumps(ret)

return ret_json

```

下列範例會呼叫具有常值的外部函數。

```

SELECT exfunc_multiplication(8, 9, 2);
   exfunc_multiplication
-----
                144
(1 row)

```

下列範例會建立名為 t_multi 的資料表，其中包含整數資料類型的三個資料欄 c1、c2 和 c3。外部函數會透過傳遞此資料表的資料欄名稱來呼叫。資料會以這種方式插入，進而導致整數溢位，以顯示錯誤的傳播方式。

```

CREATE TABLE t_multi (c1 int, c2 int, c3 int);
INSERT INTO t_multi VALUES (2147483647, 2147483647, 4);
SELECT exfunc_multiplication(c1, c2, c3) FROM t_multi;
DETAIL:
-----

```

```
error: Integer multiplication overflow
code:      32004context:
context:
query:     38
location:  exfunc_data.cpp:276
process:   query2_16_38 [pid=30494]
-----
```

CREATE EXTERNAL MODEL

主題

- [的先決條件 CREATE EXTERNAL MODEL](#)
- [所需權限](#)
- [成本控制](#)
- [CREATE EXTERNAL MODEL 語法](#)
- [CREATE EXTERNAL MODEL 參數和設定](#)
- [CREATE EXTERNAL MODEL 推論函數參數](#)

的先決條件 CREATE EXTERNAL MODEL

使用CREATEEXTERNALMODEL陳述式之前，請先完成 中的先決條件[使用 Amazon Redshift ML 的叢集設定](#)。以下是先決條件的概要。

- 使用 AWS 管理主控台或 AWS 命令列介面 () 建立 Amazon Redshift 叢集AWS CLI。
- 在建立叢集時連接 AWS Identity and Access Management (IAM) 政策。
- 若要允許 Amazon Redshift 和 Amazon Bedrock 擔任與其他 服務互動的角色，請將適當的信任政策新增至該IAM角色。

如需IAM角色、信任政策和其他先決條件的詳細資訊，請參閱 [使用 Amazon Redshift ML 的叢集設定](#)。

所需權限

以下是 CREATE EXTERNAL 的必要權限MODEL：

- 超級使用者
- 具有 CREATEEXTERNALMODEL權限的使用者
- 具有 GRANTCREATEEXTERNALMODEL權限的角色

成本控制

Amazon Redshift ML 會使用現有的叢集資源建立預測模型，因此您不必支付額外費用。不過，根據您選取的模型使用 Amazon Bedrock 時 AWS 需付費。如需詳細資訊，請參閱[使用 Amazon Redshift 資料 API](#)。

CREATE EXTERNAL MODEL 語法

以下是 CREATEEXTERNALMODEL 陳述式的完整語法。

```
CREATE EXTERNAL MODEL model_name
FUNCTION function_name
IAM_ROLE {default/'arn:aws:iam::<account-id>:role/<role-name>'}
MODEL_TYPE BEDROCK
SETTINGS (
  MODEL_ID model_id
  [, PROMPT 'prompt prefix']
  [, SUFFIX 'prompt suffix']
  [, REQUEST_TYPE {RAW|UNIFIED}]
  [, RESPONSE_TYPE {VARCHAR|SUPER}]
);
```

此 CREATE EXTERNAL MODEL 命令會建立推論函數，供您用來產生內容。

以下是使用的 CREATE EXTERNAL MODEL 建立的推論函數語法 REQUEST_TYPE RAW：

```
SELECT inference_function_name(request_super)
[FROM table];
```

以下是使用的 CREATE EXTERNAL MODEL 建立的推論函數語法 REQUEST_TYPE UNIFIED：

```
SELECT inference_function_name(input_text, [, inference_config [,
  additional_model_request_fields]])
[FROM table];
```

如需有關如何使用推論函數的資訊，請參閱[使用外部模型與 Amazon Bedrock 進行 Amazon Redshift ML 整合](#)。

CREATE EXTERNAL MODEL 參數和設定

本節說明 CREATE EXTERNAL MODEL 命令的參數和設定。

主題

- [CREATE EXTERNAL MODEL 參數](#)
- [CREATE EXTERNAL MODEL 設定](#)

CREATE EXTERNAL MODEL 參數

model_name

外部模型的名稱。結構描述中的模型名稱必須是唯一的。

FUNCTION function_name (data_type 【 , ...】)

所CREATE EXTERNAL MODEL建立推論函數的名稱。您可以使用推論函數將請求傳送至 Amazon Bedrock，並擷取 ML 產生的文字。

IAM_ROLE { 預設 | 'arn : aws : iam : : <account-id> : role/<role-name>' }

Amazon Redshift 用來存取 Amazon Bedrock IAM的角色。如需IAM角色的相關資訊，請參閱 [建立或更新 Amazon Redshift ML 與 Amazon Bedrock 整合IAM的角色](#)。

MODEL_TYPE BEDROCK

指定模型類型。唯一有效的值為 BEDROCK。

SETTINGS (MODEL_ID model_id 【 , ...】)

指定外部模型設定。如需詳細資訊，請參閱以下章節。

CREATE EXTERNAL MODEL 設定

MODEL_ID model_id

外部模型的識別符，例如 anthropic.claude-v2。如需有關 Amazon Bedrock 模型的資訊 IDs，請參閱 [Amazon Bedrock 模型 IDs](#)。

PROMPT 「提示字首」

指定靜態提示，Amazon Redshift 會新增至每個推論請求的開頭。僅支援 REQUEST_TYPE的 UNIFIED。

SUFFIX 「提示字尾」

指定靜態提示，Amazon Redshift 會新增至每個推論請求的結尾。僅支援 REQUEST_TYPE的 UNIFIED。

REQUEST_TYPE { RAW | UNIFIED }

指定傳送至 Amazon Bedrock 的請求格式。有效值包括以下項目：

- **RAW**：推論函數會將輸入視為單一超值，並一律傳回超值。超值的格式專屬於選取的 Amazon Bedrock 模型。Super 是一種預測模型，結合了多個演算法，以產生單一、改善的預測。
- **UNIFIED**：推論函數使用統一 API。所有模型都具有與 Amazon Bedrock 統一且一致的介面。這適用於所有支援訊息的模型。此為預設值。

如需詳細資訊，請參閱 Amazon Bedrock [API 文件中的 Converse](#) 文件。 API

RESPONSE_TYPE { VARCHAR | SUPER }

指定回應的格式。如果 REQUEST_TYPE 是 RAW，RESPONSE_TYPE 則需要，且唯一有效的值是 SUPER。對於所有其他 REQUEST_TYPE 值，預設值為 VARCHAR，且 RESPONSE_TYPE 為選用。有效值包括以下項目：

- **VARCHAR**：Amazon Redshift 只會傳回模型所產生的文字回應。
- **SUPER**：Amazon Redshift 會將模型 JSON 所產生的完整回應傳回為超級回應。這包括文字回應，以及停止原因等資訊，以及模型輸入和輸出字符用量。Super 是一種預測模型，結合了多個演算法，以產生單一、改善的預測。

CREATE EXTERNAL MODEL 推論函數參數

本節說明 CREATE EXTERNAL MODEL 命令建立之推論函數的有效參數。

CREATE EXTERNAL MODEL REQUEST_TYPE 的推論函數參數 RAW

使用 建立 REQUEST_TYPE 的推論函數 RAW 有一個超級輸入引數，並一律傳回超級資料類型。輸入的語法超級遵循從 Amazon Bedrock 選取的特定模型請求的語法。

CREATE EXTERNAL MODEL REQUEST_TYPE 的推論函數參數 UNIFIED

input_text

Amazon Redshift 傳送至 Amazon Bedrock 的文字。

inference_config

包含 Amazon Redshift 傳送至 Amazon Bedrock 之選用參數的超值。這些可能包括下列項目：

- maxTokens
- stopSequences

- 溫度
- topP

這些參數都是選用的，且全部區分大小寫。如需有關這些參數的資訊，請參閱 Amazon Bedrock 參考 [InferenceConfiguration](#) 中的 API。

CREATE EXTERNAL SCHEMA

在目前資料庫中建立新的外部結構描述。您可以使用此外部結構描述來連線至 Amazon RDS for PostgreSQL 或 Amazon Aurora PostgreSQL-Compatible Edition 資料庫。您也可以建立外部結構描述，在外部資料目錄中參考資料庫，例如 AWS Glue、Athena，或在 Apache Hive 中繼存放區中的資料庫，例如 Amazon EMR。

此結構描述的擁有者是 CREATE EXTERNAL SCHEMA 命令的發行者。若要轉移外部結構描述的所有權，請使用 [ALTER SCHEMA](#) 來變更擁有者。若要將結構描述的存取權授予其他使用者或使用者群組，請使用 [GRANT](#) 命令。

您無法將 GRANT 或 REVOKE 命令用於外部資料表上的許可。這時請改為在外部結構描述授予和撤銷許可。

Note

如果您目前在 Amazon Athena 資料目錄中有 Redshift Spectrum 外部資料表，則可以將 Athena 資料目錄遷移到 AWS Glue Data Catalog。若要搭配 Redshift Spectrum 使用 AWS Glue Data Catalog，您可能需要變更 AWS Identity and Access Management (IAM) 政策。如需詳細資訊，請參閱 Athena 使用者指南中的 [升級至 AWS Glue 資料目錄](#)。

若要檢視外部結構描述的詳細資訊，請查詢 [SVV_EXTERNAL_SCHEMAS](#) 系統畫面。

語法

下列語法說明使用外部資料目錄參考資料的 CREATE EXTERNAL SCHEMA 命令。如需詳細資訊，請參閱 [Amazon Redshift Spectrum](#)。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM [ [ DATA CATALOG ] | HIVE METASTORE | POSTGRES | MYSQL | KINESIS | MSK |
      REDSHIFT ]
[ DATABASE 'database_name' ]
```

```
[ SCHEMA 'schema_name' ]
[ REGION 'aws-region' ]
[ IAM_ROLE [ default | 'SESSION' | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' ] ]
[ AUTHENTICATION [ none | iam | mtls ] ]
[ AUTHENTICATION_ARN 'acm-certificate-arn' | SECRET_ARN 'ssm-secret-arn' ]
[ URI ['hive_metastore_uri' [ PORT port_number ] | 'hostname' [ PORT port_number ] |
'msk bootstrap URL' ] ]
[ CLUSTER_ARN 'arn:aws:kafka:<region>:<AWS ##-id>:cluster/msk/<cluster uuid>' ]
[ CATALOG_ROLE [ 'SESSION' | 'catalog-role-arn-string' ] ]
[ CREATE EXTERNAL DATABASE IF NOT EXISTS ]
[ CATALOG_ID 'Amazon Web Services account ID containing Glue or Lake Formation
database' ]
```

下列語法說明使用聯合查詢 RDSPOSTGRES 或 Aurora Postgre 來參考資料的 CREATEEXTERNALSCHEMA 命令 SQL。您也可以建立參照串流來源的外部結構描述，例如 Kinesis Data Streams。如需詳細資訊，請參閱 [使用 Amazon Redshift 中的聯合查詢來查詢資料](#)。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM POSTGRES
DATABASE 'federated_database_name' [SCHEMA 'schema_name']
URI 'hostname' [ PORT port_number ]
IAM_ROLE [ default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' ]
SECRET_ARN 'ssm-secret-arn'
```

下列語法說明使用聯合查詢至 RDS MySQL 或 Aurora My 來參考資料的 CREATEEXTERNALSCHEMA 命令 SQL。如需詳細資訊，請參閱 [使用 Amazon Redshift 中的聯合查詢來查詢資料](#)。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] local_schema_name
FROM MYSQL
DATABASE 'federated_database_name'
URI 'hostname' [ PORT port_number ]
IAM_ROLE [ default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' ]
SECRET_ARN 'ssm-secret-arn'
```

下列語法說明用於參考 Kinesis 串流中資料的 CREATEEXTERNALSCHEMA 命令。如需詳細資訊，請參閱 [將擷取串流至具體化視觀表](#)。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] schema_name
FROM KINESIS
IAM_ROLE [ default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' ]
```

下列語法說明用於參考 Amazon Managed Streaming for Apache Kafka 叢集的 CREATEEXTERNALSCHEMA 命令，及其要擷取的主題。若要連線，請提供代理程式 URI。如需詳細資訊，請參閱[將擷取串流至具體化視觀表](#)。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] schema_name
FROM MSK
[ IAM_ROLE [ default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' ] ]
URI 'msk bootstrap URL'
AUTHENTICATION [ none | iam | mtls ]
[ AUTHENTICATION_ARN 'acm-certificate-arn' | SECRET_ARN 'ssm-secret- arn' ];
```

下列語法說明使用跨資料庫查詢來參考資料的 CREATEEXTERNALSCHEMA 命令。

```
CREATE EXTERNAL SCHEMA local_schema_name
FROM REDSHIFT
DATABASE 'redshift_database_name' SCHEMA 'redshift_schema_name'
```

參數

IF NOT EXISTS

此子句會指出，若指定的結構描述已存在，則命令不應進行任何變更，且應傳回結構描述存在的訊息，而不是在發生錯誤的情況下終止。此子句在指令碼時很有用，因此如果 CREATEEXTERNALSCHEMA 嘗試建立已存在的結構描述，指令碼不會失敗。

local_schema_name

新外部結構描述的名稱。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

```
FROM [ DATA CATALOG ] | HIVE METASTORE | POSTGRES | MYSQL | KINESIS | MSK |
REDSHIFT
```

指出外部資料庫所在位置的關鍵字。

DATA CATALOG 表示外部資料庫是在 Athena 資料目錄或 中定義的 AWS Glue Data Catalog。

如果外部資料庫是在不同 AWS 區域的外部資料目錄中定義，則需要 REGION 參數。DATA CATALOG 為預設值。

HIVE METASTORE 表示外部資料庫是在 Apache Hive 中繼存放區中定義的。如果指定 HIVE METASTORE，URI 則需要。

POSTGRES 表示外部資料庫是在 RDS PostgreSQL 或 Aurora Postgre 中定義的 SQL。

MYSQL 表示外部資料庫已在 RDS MySQL 或 Aurora My 中定義SQL。

KINESIS 表示資料來源是來自 Kinesis Data Streams 的串流。

MSK 表示資料來源是 Amazon MSK佈建或無伺服器叢集。

FROM REDSHIFT

指出資料庫位於 Amazon Redshift 中的關鍵字。

DATABASE 'redshift_database_name' 'SCHEMAredshift_schema_name '

Amazon Redshift 資料庫的名稱。

redshift_schema_name 表示 Amazon Redshift 中的結構描述。預設的 redshift_schema_name 為 public。

DATABASE 'federated_database_name '

表示支援的 PostgreSQL 或 MySQL 資料庫引擎中外部資料庫名稱的關鍵字。

【SCHEMA 'schema_name '】

schema_name 表示支援的 PostgreSQL 資料庫引擎中的結構描述。預設 schema_name 是 public。


當您將聯合查詢設定為支援的 MySQL 資料庫引擎SCHEMA時，無法指定。

REGION 'aws-region '

如果在 Athena 資料型錄或 中定義外部資料庫 AWS Glue Data Catalog，則定義資料庫所在的 AWS 區域。如果資料庫是在外部資料目錄中定義，則需要此參數。

URI 【 'hive_metastore_uri' 【 PORT port_number 】 | 'hostname' 【 PORT port_number 】 | 'msk bootstrap URL' 】

支援的 PostgreSQL 或 MySQL 資料庫引擎的主機名稱URI和 port_number。hostname 是複本集的前端節點。端點必須可從 Amazon Redshift 叢集連接 (可路由)。預設 PostgreSQL port_number 為 5432。預設的 MySQL port_number 為 3306。

 Note

支援的 PostgreSQL 或我的SQL資料庫引擎必須與 Amazon Redshift 叢集VPC位於相同位置，且安全群組會連結 Amazon Redshift 和 RDS url-rsPostgreSQL 或 Aurora PostgreSQL。此外，您可以使用增強型VPC路由來設定跨VPC使用案例。如需詳細資訊，請參閱 [Redshift 受管VPC端點](#)。

指定 hive 中繼存放區 URI

如果資料庫位於 Hive 中繼存放區中，請指定中繼存放區的 URI 和連接埠號碼。預設連接埠號碼為 9083。

URI 不包含通訊協定規格 ("http://")。有效範例URI : uri '172.10.10.10'。

指定代理程式URI以進行串流擷取

包含 bootstrap-broker URI 可讓您連線至 Amazon MSK 叢集並接收串流資料。如需詳細資訊並查看範例，請參閱 [從 Amazon Managed Streaming for Apache Kafka 開始串流擷取](#)。

IAM_ROLE 【預設 | 'SESSION' | 'arn : aws : iam : : <AWS ##-id> : 角色 / <role-name>']

使用預設關鍵字讓 Amazon Redshift 在 CREATEEXTERNALSCHEMA 命令執行時，使用設定為預設並與叢集相關聯的 IAM 角色。

如果您使用聯合身分連線到 Amazon Redshift 叢集，並從使用此命令建立的外部結構描述存取資料表，請使用 'SESSION'。如需詳細資訊，請參閱 [使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取](#)，其中會說明如何設定聯合身分。請注意，只有在使用 建立結構描述時 ARN，才能使用此組態 'SESSION' 來取代 DATA CATALOG。

針對叢集用於身分驗證和授權 IAM 的角色，請使用 Amazon Resource Name (ARN)。至少，IAM 角色必須具有許可，才能在要存取的 Amazon S3 儲存貯體上執行 LIST 操作，以及在儲存貯體包含的 Amazon S3 物件上執行 GET 操作。

下列顯示單一的 IAM_ROLE 參數字串語法 ARN。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

您可以連結角色，讓您的叢集擔任另一個 IAM 角色，可能屬於另一個帳戶。您最多可以鏈結 10 個角色。如需鏈結角色的範例，請參閱 [Amazon Redshift Spectrum 中的鏈結 IAM 角色](#)。

在此 IAM 角色中，連接類似下列的 IAM 許可政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessSecret",
      "Effect": "Allow",
      "Action": [
```

```

        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
    ],
    "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-
rds-secret-VNenFy"
  },
  {
    "Sid": "VisualEditor1",
    "Effect": "Allow",
    "Action": [
      "secretsmanager:GetRandomPassword",
      "secretsmanager:ListSecrets"
    ],
    "Resource": "*"
  }
]
}

```

如需建立IAM角色以搭配聯合查詢使用的步驟，請參閱 [建立密碼和IAM角色以使用聯合查詢](#)。

Note

不要在鏈結的角色清單中包含空格。

以下顯示鏈結三個角色的語法。

```

IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-
account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'

```

```
SECRET_ARN 'ssm-secret-arn'
```

使用建立的支援 PostgreSQL 或 MySQL 資料庫引擎秘密的 Amazon Resource Name (ARN) AWS Secrets Manager。如需有關如何建立和擷取ARN秘密的資訊，請參閱 AWS Secrets Manager 使用者指南 中的[建立基本秘密](#)和[擷取秘密值秘密](#)。

```
CATALOG_ROLE [ 'SESSION' | catalog-role-arn-string]
```

使用聯合身分透過 'SESSION' 連接到 Amazon Redshift 叢集，以便對資料目錄進行身份驗證和授權。如需完成聯合身分步驟的相關資訊，請參閱[使用聯合身分管理 Amazon Redshift 對本機資](#)

[源和 Amazon Redshift Spectrum 外部資料表的存取](#)。請注意，只有在 `CREATE EXTERNAL DATABASE` 中建立結構描述時，才能使用 'SESSION' 角色 DATA CATALOG。

使用 Amazon Resource Name ARN 作為叢集用於資料目錄身分驗證和授權 IAM 的角色。

如果未指定 CATALOG_ROLE，Amazon Redshift 會使用指定的 IAM_ROLE。目錄角色必須具有在 AWS Glue 或 Athena 中存取資料目錄的許可。如需詳細資訊，請參閱 [IAM Amazon Redshift Spectrum 的政策](#)。

下列顯示單一的 CATALOG_ROLE 參數字串語法 ARN。

```
CATALOG_ROLE 'arn:aws:iam::aws-account-id:role/<catalog-role>'
```

您可以連結角色，讓您的叢集擔任另一個 IAM 角色，可能屬於另一個帳戶。您最多可以鏈結 10 個角色。如需詳細資訊，請參閱 [Amazon Redshift Spectrum 中的鏈結 IAM 角色](#)。

Note

鏈結的角色清單不得包含空格。

以下顯示鏈結三個角色的語法。

```
CATALOG_ROLE 'arn:aws:iam::aws-account-id:role/<catalog-role-1-name>,arn:aws:iam::aws-account-id:role/<catalog-role-2-name>,arn:aws:iam::aws-account-id:role/<catalog-role-3-name>'
```

CREATE EXTERNAL DATABASE IF NOT EXISTS

如果指定的外部資料庫不存在，則會使用 DATABASE 引數指定的名稱建立外部資料庫的子句。如果指定的外部資料庫存在，則此命令不會進行任何變更。在這種情況下，此命令會傳回外部資料庫存在的訊息，並不會因錯誤而終止。

Note

您無法 NOT EXISTS 搭配 CREATE EXTERNAL DATABASE 使用 HIVE IF METASTORE。若要在為 CREATE EXTERNAL DATABASE 啟用 Data Catalog NOT EXISTS 的情況下使用 IF AWS Lake Formation，您需要 Data Catalog 的 CREATE_DATABASE 許可。

CATALOG_ID '包含 Glue 或 Lake Formation 資料庫的 Amazon Web Services 帳戶 ID '

儲存資料目錄資料庫的帳戶 ID。

只有在您打算連接到 Amazon Redshift 叢集或 Amazon Redshift Serverless，並透過設定下列任一項來使用聯合身分進行資料目錄的身分驗證和授權時，才能指定 CATALOG_ID：

- CATALOG_ROLE 至 'SESSION'
- IAM_ROLE 設定為 'SESSION'，'CATALOG_ROLE' 設定為其預設值

如需完成聯合身分步驟的相關資訊，請參閱[使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取](#)。

AUTHENTICATION

為串流擷取定義的驗證類型。具有驗證類型的串流擷取會與 Amazon Managed Streaming for Apache Kafka 搭配運作。AUTHENTICATION 類型如下：

- 無 – 指定不需要身分驗證。這相當於上的未驗證存取MSK。
- iam – 指定IAM身分驗證。當您選擇此選項時，請確定IAM角色具有IAM身分驗證的許可。如需定義外部結構描述的相關資訊，請參閱 [從 Amazon Managed Streaming for Apache Kafka \(Amazon MSK \) 開始串流擷取](#)。
- mtls – 指定相互傳輸層安全透過促進用戶端和伺服器之間的身分驗證來提供安全通訊。在此情況下，用戶端為 Redshift，伺服器為 Amazon MSK。如需使用 m 設定串流擷取的詳細資訊TLS，請參閱 [使用 mTLS 進行身分驗證，以便從 Amazon 擷取 Redshift 串流 MSK](#)。

AUTHENTICATION_ARN

Amazon Redshift 使用 ARN Amazon 進行 mtls 身分驗證的 AWS Certificate Manager 憑證的 MSK。當您選擇發行的憑證時，ACM主控台中ARN會提供。

CLUSTER_ARN

對於串流擷取，CLUSTER_ARN 是串流來源的 Amazon Managed Streaming for Apache Kafka 叢集的叢集識別符。使用 CLUSTER_時ARN，它需要包含 kafka:GetBootstrapBrokers 許可 IAM的角色政策。此選項用於向後相容性。目前，我們建議您使用 bootstrap-broker URI選項來連線至 Amazon Managed Streaming for Apache Kafka 叢集。如需詳細資訊，請參閱[串流擷取](#)。

使用須知

如需使用 Athena 資料目錄時的限制，請參閱 AWS 一般參考中的 [Athena 限制](#)。

如需使用時的限制 AWS Glue Data Catalog，請參閱 [AWS Glue 限制](#) AWS 一般參考。

這些限制不適用於 Hive 中繼存放區。

每個資料庫最多 9,900 個結構描述。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [配額和限制](#)。

若要取消註冊結構描述，請使用 [DROP SCHEMA](#) 命令。

若要檢視外部結構描述的詳細資訊，請查詢下列系統檢視：

- [SVV_EXTERNAL_SCHEMAS](#)
- [SVV_EXTERNAL_TABLES](#)
- [SVV_EXTERNAL_COLUMNS](#)

範例

以下範例使用美國西部 (奧勒岡) 區域中資料目錄中名為 `sampledb` 的資料庫建立外部結構描述。將此範例與 Athena 或 AWS Glue 資料目錄搭配使用。

```
create external schema spectrum_schema
from data catalog
database 'sampledb'
region 'us-west-2'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole';
```

以下範例會建立外部結構描述，並建立名為 `spectrum_db` 的新外部資料庫。

```
create external schema spectrum_schema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole'
create external database if not exists;
```

以下範例使用名為 `hive_db` 的 Hive 中繼存放區資料庫建立外部結構描述。

```
create external schema hive_schema
from hive metastore
database 'hive_db'
uri '172.10.10.10' port 99
```

```
iam_role 'arn:aws:iam::123456789012:role/MySpectrumRole';
```

以下範例會鏈結角色，以使用 myS3Role 角色存取 Amazon S3，並使用 myAthenaRole 存取資料目錄。如需詳細資訊，請參閱[Amazon Redshift Spectrum 中的鏈結IAM角色](#)。

```
create external schema spectrum_schema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myRedshiftRole,arn:aws:iam::123456789012:role/myS3Role'
catalog_role 'arn:aws:iam::123456789012:role/myAthenaRole'
create external database if not exists;
```

下列範例會建立參考 Aurora PostgreSQL 資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] myRedshiftSchema
FROM POSTGRES
DATABASE 'my_aurora_db' SCHEMA 'my_aurora_schema'
URI 'endpoint to aurora hostname' PORT 5432
IAM_ROLE 'arn:aws:iam::123456789012:role/MyAuroraRole'
SECRET_ARN 'arn:aws:secretsmanager:us-east-2:123456789012:secret:development/MyTestDatabase-AbCdEf'
```

下列範例會建立外部結構描述，以參照在取用者叢集上匯入的 sales_db。

```
CREATE EXTERNAL SCHEMA sales_schema FROM REDSHIFT DATABASE 'sales_db' SCHEMA 'public';
```

下列範例會建立參考 Aurora MySQL 資料庫的外部結構描述。

```
CREATE EXTERNAL SCHEMA [IF NOT EXISTS] myRedshiftSchema
FROM MYSQL
DATABASE 'my_aurora_db'
URI 'endpoint to aurora hostname'
IAM_ROLE 'arn:aws:iam::123456789012:role/MyAuroraRole'
SECRET_ARN 'arn:aws:secretsmanager:us-east-2:123456789012:secret:development/MyTestDatabase-AbCdEf'
```

CREATE EXTERNAL TABLE

在指定的結構描述中建立新的外部資料表。所有外部資料表都必須經由外部結構描述建立。外部結構描述和外部資料表不支援搜尋路徑。如需詳細資訊，請參閱[CREATE EXTERNAL SCHEMA](#)。

除了使用 `CREATE EXTERNAL TABLE` 命令建立的外部資料表之外，Amazon Redshift 也可以參考 AWS Glue 或 AWS Lake Formation 目錄中定義的外部資料表，或 Apache Hive 中繼存放區。使用 [CREATE EXTERNAL SCHEMA](#) 命令註冊外部目錄中定義的外部資料庫，並且在 Amazon Redshift 中將外部資料表提供使用。如果外部資料表存在於 AWS Glue 或 AWS Lake Formation 目錄或 Hive 中繼存放區中，則不需要使用 `CREATE EXTERNAL TABLE` 建立資料表。若要檢視外部資料表，請查詢 [SVV_EXTERNAL_TABLES](#) 系統畫面。

透過執行 `CREATE EXTERNAL TABLE AS` 命令，您可以根據查詢中的資料欄定義建立外部資料表，並將該查詢的結果寫入 Amazon S3。結果為 Apache Parquet 或分隔文字格式。如果外部資料表有一個或多個分割區索引鍵，Amazon Redshift 會根據這些分割區索引鍵來分割新檔案，並自動將新的分割區註冊到外部類別目錄中。如需 `CREATE EXTERNAL TABLE AS` 的詳細資訊，請參閱 [使用須知](#)。

您可以使用與其他 Amazon Redshift `SELECT` 資料表相同的語法來查詢外部資料表。您也可以使用 `INSERT` 語法，將新檔案寫入 Amazon S3 上外部資料表的位置。如需詳細資訊，請參閱 [INSERT \(外部資料表\)](#)。

若要使用外部資料表建立檢視，請在 [CREATE VIEW](#) 陳述式中包含 `WITH NO SCHEMABINDING` 子句。

您無法在交易 `CREATE EXTERNAL TABLE` 內執行 (`BEGIN ... END`)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

所需權限

您必須為外部結構描述的擁有者或超級使用者，始可建立外部資料表。若要轉移外部結構描述的所有權，請使用 `ALTER SCHEMA` 變更擁有者。外部資料表的存取權是由外部結構描述的存取權所控制。您無法對外部資料表上的許可執行 [GRANT](#) 或 [REVOKE](#)。相反地，請在外部結構描述 `USAGE` 上授予或撤銷。

[使用須知](#) 具有有關外部資料表特定權限的其他資訊。

語法

```
CREATE EXTERNAL TABLE
external_schema.table_name
(column_name data_type [, ...] )
[ PARTITIONED BY (col_name data_type [, ...] ) ]
[ { ROW FORMAT DELIMITED row_format |
  ROW FORMAT SERDE 'serde_name'
  [ WITH SERDEPROPERTIES ( 'property_name' = 'property_value' [, ...] ) ] } ]
```

```
STORED AS file_format
LOCATION { 's3://bucket/folder/' | 's3://bucket/manifest_file' }
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
```

以下是 CREATE EXTERNAL TABLE AS 的語法。

```
CREATE EXTERNAL TABLE
external_schema.table_name
[ PARTITIONED BY (col_name [, ... ] ) ]
[ ROW FORMAT DELIMITED row_format ]
STORED AS file_format
LOCATION { 's3://bucket/folder/' }
[ TABLE PROPERTIES ( 'property_name'='property_value' [, ...] ) ]
AS
{ select_statement }
```

參數

external_schema.table_name

要建立的資料表名稱，以外部結構描述名稱限定。外部資料表必須建立在外部結構描述中。如需詳細資訊，請參閱[CREATE EXTERNAL SCHEMA](#)。

資料表名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 個多位元組字元，最多四個位元組。Amazon Redshift 會強制執行每個叢集 9,900 個資料表的限制，包括使用者定義的臨時資料表，以及 Amazon Redshift 在查詢處理或系統維護期間建立的臨時資料表。您也可以選擇使用資料庫名稱來限定資料表名稱。在以下範例中，資料庫名稱為 `spectrum_db`，外部結構描述名稱為 `spectrum_schema`，而資料表名稱為 `test`。

```
create external table spectrum_db.spectrum_schema.test (c1 int)
stored as parquet
location 's3://amzn-s3-demo-bucket/myfolder/';
```

如果指定的資料庫或結構描述不存在，則不會建立資料表，而且陳述式會傳回錯誤。您無法在系統資料庫 `template0`、`template1`、`padb_harvest` 和 `sys:internal` 中建立資料表或檢視。

資料表名稱對於指定的結構描述來說必須是唯一的。

如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

(column_name data_type)

要建立的每個資料欄的名稱和資料類型。

資料欄名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 個多位元組字元，最多四個位元組。您無法指定資料欄名稱 "\$path" 或 "\$size"。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

Amazon Redshift 預設會以虛擬資料欄 \$path 和 \$size 建立外部資料表。您可以藉由將 spectrum_enable_pseudo_columns 組態參數設定為 false，以停用工作階段的虛擬資料欄建立。如需詳細資訊，請參閱 [虛擬資料欄](#)。

若已啟用虛擬資料欄，則單一資料表中可定義的資料欄數目上限為 1,598 個。若未啟用虛擬資料欄，則單一資料表中可定義的資料欄數目上限為 1,600 個。

若您要建立「寬資料表」，則務必確定在載入和查詢處理期間，您的資料欄清單未超過中繼結果的資料列寬度界限。如需詳細資訊，請參閱 [使用須知](#)。

對於 CREATE EXTERNAL TABLE AS 命令，不需要資料欄清單，因為資料欄是從查詢衍生而來。

data_type

支援以下 [資料類型](#)：

- SMALLINT (INT2)
- INTEGER (INT, INT4)
- BIGINT (INT8)
- DECIMAL (NUMERIC)
- REAL (FLOAT4)
- DOUBLE PRECISION (FLOAT8)
- BOOLEAN (BOOL)
- CHAR (CHARACTER)
- VARCHAR (CHARACTER VARYING)
- VARBYTE (CHARACTER VARYING) – 可與 Parquet 和 ORC 資料檔案搭配使用，且僅適用於非分割資料表。
- DATE – 只能與文字、Parquet 或 ORC 資料檔案搭配使用，或用作分割區資料欄。
- TIMESTAMP

對於 DATE，您可以使用如下所述的格式。使用數字表示的月份值支援下列格式：

- mm-dd-yyyy 例如：05-01-2017。此為預設值。
- yyyy-mm-dd，其中年份由 2 個以上的數字表示。例如：2017-05-01。

使用三個字母縮寫表示的月份值支援下列格式：

- mmm-dd-yyyy 例如：may-01-2017。此為預設值。
- dd-mmm-yyyy，其中年份由 2 個以上的數字表示。例如：01-may-2017。
- yyyy-mmm-dd，其中年份由 2 個以上的數字表示。例如：2017-may-01。

對於始終小於 100 的年份值，年份的計算方式如下：

- 如果年份小於 70，則該年份的計算方式為年份加上 2000。例如，使用 mm-dd-yyyy 格式的日期 05-01-17 會轉換成 05-01-2017。
- 如果年份小於 100 且大於 69，則該年份的計算方式為年份加上 1900。例如，使用 mm-dd-yyyy 格式的日期 05-01-89 會轉換成 05-01-1989。
- 對於以兩個數字表示的年份值，請在前面加上零以使用 4 個數字表示年份。

文字檔案中的時間戳記值格式必須為 yyyy-mm-dd HH:mm:ss.SSSSSS，如以下的時間戳記值所示：2017-05-01 11:30:59.000000。

資料 VARCHAR 欄的長度以位元組為單位定義，而非字元。例如，VARCHAR (12) 欄可以包含 12 個單位元組字元或 6 個雙位元組字元。查詢外部資料表時，結果會截斷以配合定義的資料欄大小，而不會傳回錯誤。如需詳細資訊，請參閱[儲存與範圍](#)。

為獲得最佳效能，建議您指定可配合您的資料的最小資料欄大小。若要尋找資料欄中值的位元組大小上限，請使用 [OCTET_LENGTH](#) 函數。下列範例會傳回電子郵件資料欄中值的大小上限。

```
select max(octet_length(email)) from users;
```

```
max
---
62
```

PARTITIONED BY (col_name data_type [, ...])

此子句會定義包含一個或多個分割區資料欄的分割資料表。每種指定的組合都會另外使用一個資料目錄，這樣可在某些情況下改善查詢效能。資料表資料內並未包含分割資料欄。如果您對 col_name 使用的值與資料表資料欄相同，則會發生錯誤。

建立分割的資料表後，請使用 ... [ALTER TABLE](#) `ADDPARTITION` 陳述式來變更資料表，以將新分割區註冊至外部目錄。當您新增分割區時，會定義 Amazon S3 上包含分割區資料的子資料夾位置。

例如，如果使用 `spectrum.lineitem_part` 定義資料表 `PARTITIONED BY (l_shipdate date)`，請執行下列 `ALTER TABLE` 命令來新增分割區。

```
ALTER TABLE spectrum.lineitem_part ADD PARTITION (l_shipdate='1992-01-29')
LOCATION 's3://spectrum-public/lineitem_partition/l_shipdate=1992-01-29';
```

如果您使用的是 `CREATE EXTERNAL TABLE AS`，則不需要執行 `ALTER TABLE...ADD PARTITION`。Amazon Redshift 會自動在外部目錄中註冊新的分割區。Amazon Redshift 也會根據資料表中定義的一個或多個分割區索引鍵，自動將對應的資料寫入 Amazon S3 中的分割區。

若要檢視分割區，請查詢 [SVV_EXTERNAL_PARTITIONS](#) 系統畫面。

Note

對於 `CREATE EXTERNAL TABLE AS` 命令，您不需要指定分割區資料欄的資料類型，因為此資料欄衍生自查詢。

ROW FORMAT DELIMITED rowformat

此子句會指定基礎資料的格式。rowformat 可能的值如下：

- `LINES TERMINATED` 依 '分隔符號'
- `FIELDS TERMINATED` 依 '分隔符號'

指定 'delimiter' 的單一 ASCII 字元。您可以使用八進位字元來指定非列印 ASCII 字元，格式 '`\ddd`' `d` 為八進位位數（0-7）到 '\177'。下列範例使用八進位指定 BEL（鐘型）字元。

```
ROW FORMAT DELIMITED FIELDS TERMINATED BY '\007'
```

`ROW FORMAT` 如果省略，則預設格式為 `DELIMITED FIELDS TERMINATED BY '\A'`（標題開頭）和 `LINES TERMINATED BY '\n'`（新行）。

`ROW FORMAT SERDE 'serde_name'`，【`WITH SERDEPROPERTIES ('property_name' = 'property_value' [, ...])`】

指定基礎資料的 `SERDE` 格式的子句。

'serde_name'

的名稱 SerDe。您可以指定下列格式：

- org.apache.hadoop.hive.serde2.RegexSerDe
- com.amazonaws.glue.serde.GrokSerDe
- org.apache.hadoop.hive.serde2.OpenCSVSerde

此參數支援 O 的下列 SerDe 屬性 OpenCSVSerde：

```
'wholeFile' = 'true'
```

將 wholeFile 屬性設定為 true，以正確剖析 OpenCSV 請求的引號字串中的新行字元 (\n)。

- org.openx.data.jsonserde.JsonSerDe
 - JSON SERDE 也支援 Ion 檔案。
 - JSON 必須格式良好。
 - Ion 和 中的時間戳記 JSON 必須使用 ISO8601 格式。
 - 此參數支援 的下列 SerDe 屬性 JsonSerDe：

```
'strip.outer.array'='true'
```

處理包含在外部括號 ([...]) 中的一個非常大陣列的 Ion/JSON 檔案，就像在陣列中包含多個 JSON 記錄一樣。

- com.amazon.ionhiveserde.IonHiveSerDe

除了資料類型之外，Amazon ION 格式還提供文字和二進位格式。對於參考 ION 格式資料的外部資料表，您可以將外部資料表中的每個資料欄對應至 ION 格式資料中的對應元素。如需詳細資訊，請參閱 [Amazon Ion](#)。您還需要指定輸入和輸出格式。

WITH SERDEPROPERTIES ('property_name' = 'property_value' [, ...])]

選擇性地指定屬性名稱和值，並以逗號分隔。

ROW FORMAT 如果省略，則預設格式為 DELIMITED FIELDS TERMINATED BY '\A' (標題開頭) 和 LINES TERMINATED BY '\n' (新行)。

STORED AS file_format

資料檔案的檔案格式。

有效格式如下：

- PARQUET
- RCFILE (ColumnarSerDe 僅適用於 的資料，而非 LazyBinaryColumnarSerDe)
- SEQUENCEFILE
- TEXTFILE (適用於文字檔案，包括JSON檔案)。
- ORC
- AVRO
- INPUTFORMAT 'input_format_classname ' OUTPUTFORMAT 'output_format_classname '

CREATE EXTERNAL TABLE AS 命令僅支援兩種檔案格式 TEXTFILE和 PARQUET。

對於 INPUTFORMAT和 OUTPUTFORMAT，指定類別名稱，如下列範例所示。

```
'org.apache.hadoop.mapred.TextInputFormat'
```

```
LOCATION { 's3 : //bucket/folder /' | 's3 : //bucket/manifest_file ' }
```

包含資料檔案的 Amazon S3 儲存貯體或資料夾路徑，或包含 Amazon S3 物件路徑清單的資訊清單檔案。儲存貯體必須與 Amazon Redshift 叢集位於相同的 AWS 區域。如需支援的 AWS 區域清單，請參閱 [Amazon Redshift Spectrum 限制](#)。

如果路徑指定的是儲存貯體或資料夾，例如 's3://amzn-s3-demo-bucket/custdata/'，則 Redshift Spectrum 會掃描指定儲存貯體或資料夾以及任何子資料夾裡的檔案。Redshift Spectrum 會忽略隱藏檔案以及開頭為句號或底線的檔案。

如果路徑指定的是資訊清單檔案，則 's3://bucket/manifest_file' 引數必須明確參考單一檔案，例如 's3://amzn-s3-demo-bucket/manifest.txt'。無法參考金鑰前綴。

資訊清單是JSON格式的文字檔案，列出要從 Amazon S3 載入的每個檔案URL的，以及檔案的大小，以位元組為單位。URL 包含 檔案的儲存貯體名稱和完整物件路徑。在資訊清單中指定的檔案可以位於不同的儲存貯體中，但所有儲存貯體都必須與 Amazon Redshift 叢集位於相同的 AWS 區域中。如果某個檔案列出兩次，則該檔案會載入兩次。下列範例顯示載入三個檔案的JSON清單清單的。

```
{
  "entries": [
    {"url": "s3://amzn-s3-demo-bucket1/custdata.1", "meta": { "content_length":
      5956875 } },
```

```

    {"url":"s3://amzn-s3-demo-bucket1/custdata.2", "meta": { "content_length":
5997091 } },
    {"url":"s3://amzn-s3-demo-bucket2/custdata.1", "meta": { "content_length":
5978675 } }
  ]
}

```

您可以讓包含特定檔案變為強制性。若要這麼做，請在資訊清單中的檔案層級包含 `mandatory` 選項。當您查詢缺少必要檔案的外部資料表時，`SELECT` 陳述式會失敗。確定外部資料表定義中包含的所有檔案均存在。如果並非全部存在，則會出現錯誤，顯示找不到第一個強制性檔案。下列範例顯示 資訊清單JSON 的，其中 `mandatory` 選項設定為 `true`。

```

{
  "entries": [
    {"url":"s3://amzn-s3-demo-bucket1/custdata.1", "mandatory":true, "meta":
{ "content_length": 5956875 } },
    {"url":"s3://amzn-s3-demo-bucket1/custdata.2", "mandatory":false, "meta":
{ "content_length": 5997091 } },
    {"url":"s3://amzn-s3-demo-bucket2/custdata.1", "meta": { "content_length":
5978675 } }
  ]
}

```

若要參考使用 建立的檔案 `UNLOAD`，您可以使用使用 建立的清單 `UNLOAD` 與 `MANIFEST` 參數。資訊清單檔案與 [COPY 從 Amazon S3](#) 的資訊清單檔案相容，但使用不同的索引鍵。未使用的索引鍵會加以忽略。

`TABLE PROPERTIES ('property_name'='property_value' [, ...])`

此子句會設定資料表屬性的資料表定義。

Note

資料表屬性區分大小寫。

`'compression_type'='value'`

如果檔案名稱未包含副檔名，此屬性會設定要使用的壓縮類型。如果您設定此屬性，但是有副檔名，則會忽略副檔名並使用屬性所設定的值。壓縮類型的有效值如下所示：

- `bzip2`

- gzip
- 無
- snappy

'data_cleansing_enabled'='true / false'

此屬性會設定資料表的資料處理是否開啟。當 'data_cleansing_enabled' 設定為 true 時，資料表的資料處理為開啟狀態。當 'data_cleansing_enabled' 設定為 false 時，資料表的資料處理為關閉狀態。以下是此屬性所控制的資料表層級資料處理屬性清單：

- column_count_mismatch_handling
- invalid_char_handling
- numeric_overflow_handling
- replacement_char
- surplus_char_handling

如需範例，請參閱 [資料處理範例](#)。

'invalid_char_handling'='value'

指定查詢結果包含無效 UTF-8 字元值時要執行的動作。您可以指定下列動作：

DISABLED

不執行無效的字元處理。

FAIL

取消傳回包含無效 UTF-8 值之資料的查詢。

SET_TO_NULL

將無效的 UTF-8 值取代為 null。

DROP_ROW

將資料列中的每個值取代為 null。

REPLACE

以您使用 replacement_char 指定的取代字元取代無效字元。

'replacement_char'='character'

指定當您將 invalid_char_handling 設定為 REPLACE 時要使用的取代字元。

'numeric_overflow_handling'='value'

指定當ORC資料包含大於資料欄定義（例如，BIGINTint16）的整數（例如，SMALLINT或int64）時要執行的動作。您可以指定下列動作：

DISABLED

無效字元處理已關閉。

FAIL

當資料包含無效字元時，取消查詢。

SET_TO_NULL

將無效字符設定為 null。

DROP_ROW

將資料列中的每個值設定為 null。

'surplus_bytes_handling'='value'

指定如何處理要載入的資料，該資料超過包含VARBYTE資料之資料欄定義的資料類型長度。根據預設，Redshift Spectrum 會針對超出欄寬度的資料，將值設定為 null。

您可以指定當查詢傳回超過資料類型長度的資料時，執行下列動作：

SET_TO_NULL

以 null 取代超過欄寬的資料。

DISABLED

不執行多餘位元組處理。

FAIL

取消傳回資料超出欄寬的查詢。

DROP_ROW

捨棄包含超出欄寬之資料的所有資料列。

TRUNCATE

如果字元超過為欄定義的字元數目上限，則移除字元。

'surplus_char_handling'='value'

指定如何處理載入的資料，該資料超過包含 VARCHAR、CHAR 或 字串資料之資料欄定義的資料類型長度。根據預設，Redshift Spectrum 會針對超出欄寬度的資料，將值設定為 null。

您可以指定當查詢傳回超過欄寬的資料時，執行下列動作：

SET_TO_NULL

以 null 取代超過欄寬的資料。

DISABLED

不執行多餘字元處理。

FAIL

取消傳回資料超出欄寬的查詢。

DROP_ROW

將資料列中的每個值取代為 null。

TRUNCATE

如果字元超過為欄定義的字元數目上限，則移除字元。

'column_count_mismatch_handling'='value'

識別檔案包含的資料列值是否少於或多於外部資料表定義中指定的欄數。此屬性僅適用於未壓縮的文字檔案格式。您可以指定下列動作：

DISABLED

欄計數不相符處理已關閉。

FAIL

如果偵測到資料欄計數不相符，則查詢失敗。

SET_TO_NULL

將遺失值填入，NULL 並忽略每列中的其他值。

DROP_ROW

從掃描中捨棄包含欄計數不相符錯誤的所有資料列。

```
'numRows'='row_count'
```

設定資料表定義 numRows 值的 屬性。若要明確更新外部資料表的統計資料，請設定 numRows 屬性以指示資料表的大小。Amazon Redshift 不會分析外部資料表來產生查詢最佳化工具用來產生查詢計劃的資料表統計資料。如果資料表統計資訊沒有為外部資料表進行設定，則 Amazon Redshift 會以「外部資料表較大而本機資料表較小」的假設來產生查詢執行計畫。

```
'skip.header.line.count'='line_count'
```

此屬性會設定每個來源檔案開頭要略過的資料列數。

```
'serialization.null.format'=''
```

此屬性會指定，欄位中提供的文字有完全相符項目時，Spectrum 應傳回 NULL 值。


```
'orc.schema.resolution'='mapping_type'
```

為使用ORC資料格式的資料表設定資料欄映射類型的屬性。將針對其他所有日期格式忽略此屬性。

資料欄映射類型的有效值如下所示：

- name
- position

如果忽略 orc.schema.resolution 屬性，則預設會依名稱映射資料欄。如果 orc.schema.resolution 設為 'name' 或 'position' 以外的任何值，則會依位置映射資料欄。如需資料欄映射的相關資訊，請參閱 [將外部資料表資料欄映射至ORC資料欄](#)。

 Note

COPY 命令只會依位置對應至ORC資料檔案。orc.schema.resolution 資料表屬性不會影響COPY命令行為。

```
'write.parallel'='on/off'
```

設定 CREATE EXTERNAL TABLE AS 是否應該平行寫入資料的 屬性。預設情況下，CREATEEXTERNALTABLEAS 會根據叢集中的切片數目，將資料平行寫入多個檔案。預設選項為開啟。當 'write.parallel' 設定為關閉時，CREATEEXTERNALTABLEAS 會依序寫入一或多個資料檔案至 Amazon S3。此資料表屬性也適用於相同外部資料表中的任何後續INSERT 陳述式。

```
'write.maxfilesize.mb'='size'
```

設定 CREATE EXTERNAL TABLE AS 寫入 Amazon S3 之每個檔案的大小上限（以 MB 為單位）的屬性。大小必須是介於 5 到 6200 之間的有效整數。預設的檔案大小上限為 6,200 MB。此資料表屬性也適用於相同外部資料表中的任何後續 INSERT 陳述式。

```
'write.kms.key.id'='value'
```

您可以指定 AWS Key Management Service 金鑰來啟用 Amazon S3 物件的伺服器端加密（SSE），其中 值為下列其中一項：

- auto 使用存放在 Amazon S3 儲存貯體中的預設 AWS KMS 金鑰。
- 您指定用來加密資料的 kms-key。

```
select_statement
```

透過定義任何查詢，將一或多個列插入外部資料表的陳述式。查詢產生的所有列都會根據表格定義，以文字或 Parquet 格式寫入到 Amazon S3。

範例

您可在 [範例](#) 取得範例集合。

使用須知

本主題包含 [CREATE EXTERNAL TABLE](#) 的使用須知。您無法使用與標準 Amazon Redshift 資料表相同的資源來檢視 Amazon Redshift Spectrum 資料表的詳細資訊，例如 [PG_TABLE_DEF](#)、CLASS、PG_ [STV_TBL_PERM](#) 或 information_schema。如果您的商業智慧或分析工具無法識別 Redshift Spectrum 外部資料表，請將您的應用程式設定為查詢 [SVV_EXTERNAL_TABLES](#) 與 [SVV_EXTERNAL_COLUMNS](#)。

CREATE EXTERNAL TABLE AS

在某些情況下，您可以在 AWS Glue Data Catalog、AWS Lake Formation external Catalog 或 Apache Hive 中繼存放區上執行 CREATE EXTERNAL TABLE AS 命令。在這種情況下，您可以使用 AWS Identity and Access Management（IAM）角色來建立外部結構描述。此 IAM 角色必須同時具有 Amazon S3 上的讀取和寫入許可。

如果您使用 Lake Formation 目錄，IAM 角色必須具有在目錄中建立資料表的許可。在此案例中，它也必須具有目標 Amazon S3 路徑上的資料湖位置許可。此 IAM 角色會成為新 AWS Lake Formation 資料表的擁有者。

為了確保檔案名稱是唯一的，Amazon Redshift 依預設會針對每個上傳到 Amazon S3 的檔案名稱使用下列格式。

```
<date>_<time>_<microseconds>_<query_id>_<slice-number>_part_<part-number>.<format>.
```

例如，20200303_004509_810669_1007_0001_part_00.parquet。

執行 CREATE EXTERNAL TABLE AS 命令時，請考慮下列事項：

- Amazon S3 位置必須為空。
- Amazon Redshift 僅在使用 STORED AS 子句時支援 PARQUET 和 TEXTFILE 格式。
- 您不需要定義欄位定義清單。新外部資料表的資料欄名稱和資料欄資料類型直接衍生自 SELECT 查詢。
- 您不需要在 PARTITIONED BY 子句中定義分割區資料欄的資料類型。如果您指定分割區金鑰，則此欄的名稱必須存在於 SELECT 查詢結果中。當具有多個分割區資料欄時，查詢中的順序 SELECT 並不重要。Amazon Redshift 使用 BY PARTITIONED 子句中定義的順序來建立外部資料表。
- Amazon Redshift 會根據分割區索引鍵值，自動將輸出檔案分割到分割區資料夾中。依預設，Amazon Redshift 會從輸出檔案中移除分割區欄位。
- 不支援 LINES TERMINATED BY 'delimiter' 子句。
- 不支援 ROW FORMAT SERDE 'serde_name' 子句。
- 不支援使用資訊清單檔案。因此，您無法定義 Amazon S3 上資訊清單檔案的 LOCATION 子句。
- Amazon Redshift 會在命令結束時自動更新 'numRows' 資料表屬性。
- 'compression_type' 資料表屬性僅接受 PARQUET 檔案格式的 'none' 或 'snappy'。
- Amazon Redshift 不允許外部 SELECT 查詢中的 LIMIT 子句。反之，您可以使用巢狀 LIMIT 子句。
- 您可以使用 STL_UNLOAD_LOG 追蹤每個 CREATE EXTERNAL TABLE AS 操作寫入 Amazon S3 的檔案。

建立和查詢外部資料表的許可

若要建立外部資料表，請確定您是外部結構描述或超級使用者的擁有者。若要轉移外部結構描述的所有權，請使用 [ALTER SCHEMA](#)。下列範例會將 spectrum_schema 結構描述的擁有者變更為 newowner。

```
alter schema spectrum_schema owner to newowner;
```

若要執行 Redshift Spectrum 查詢，您需要以下許可：

- 結構描述使用許可
- 在目前資料庫建立暫時資料表的許可

下列範例可在結構描述 `spectrum_schema` 上授予使用許可至 `spectrumusers` 使用者群組。

```
grant usage on schema spectrum_schema to group spectrumusers;
```

下列範例可在資料庫 `spectrumdb` 上授予臨時許可至 `spectrumusers` 使用者群組。

```
grant temp on database spectrumdb to group spectrumusers;
```

虛擬資料欄

Amazon Redshift 預設會以虛擬資料欄 `$path` 和 `$size` 建立外部資料表。選擇這些欄位以檢視 Amazon S3 上資料檔案的路徑，以及由查詢傳回的每列資料檔案大小。`$path` 與 `$size` 欄位名稱必須以雙引號分隔。SELECT * 子句不會傳回虛擬欄。您必須在查詢中明確包含 `$path` 和 `$size` 欄位名稱，如以下範例所示。

```
select "$path", "$size"
from spectrum.sales_part
where saledate = '2008-12-01';
```

您可以藉由將 `spectrum_enable_pseudo_columns` 組態參數設定為 `false`，以停用工作階段的虛擬資料欄建立。

Important

選擇 `$size` 或 `$path` 會產生費用，因為 Redshift Spectrum 會掃描 Amazon S3 中的資料檔案以判斷結果集的大小。如需詳細資訊，請參閱 [Amazon Redshift 定價](#)。

設定資料處理選項

您可以設定資料表參數，為要在外部資料表中查詢的資料指定輸入處理，包括：

- 包含 VARCHAR、CHAR 和 字串資料的欄中的剩餘字元。如需詳細資訊，請參閱外部資料表屬性 `surplus_char_handling`。

- 資料欄中包含 VARCHAR、CHAR 和 字串資料的字元無效。如需詳細資訊，請參閱外部資料表屬性 `invalid_char_handling`。
- 當您 REPLACE 指定外部資料表屬性 時要使用的取代字元 `invalid_char_handling`。
- 在包含整數和十進位資料的資料欄中進行轉換溢位處理。如需詳細資訊，請參閱外部資料表屬性 `numeric_overflow_handling`。
- 在包含 VARBYTE 資料的資料欄中，`Surplus_bytes_handling` 可指定多餘位元組的輸入處理。如需詳細資訊，請參閱外部資料表屬性 `surplus_bytes_handling`。

範例

下列範例會在名為 `SALES` 的 Amazon Redshift 外部結構描述 SALES 中建立名為 `spectrum` 的資料表。該資料位於 Tab 鍵分隔的文字檔案中。TABLE PROPERTIES 子句會將 `numRows` 屬性設定為 170,000 列。

根據您用來執行 CREATE EXTERNAL 的身分 TABLE，您可能需要設定 IAM 許可。最佳實務是，建議您將許可政策連接至 IAM 角色，然後視需要將其指派給使用者和群組。如需詳細資訊，請參閱 [Amazon Redshift 中的身分和存取管理](#)。

```
create external table spectrum.sales(  
  salesid integer,  
  listid integer,  
  sellerid integer,  
  buyerid integer,  
  eventid integer,  
  saledate date,  
  qty sold smallint,  
  pricepaid decimal(8,2),  
  commission decimal(8,2),  
  saletime timestamp)  
row format delimited  
fields terminated by '\t'  
stored as textfile  
location 's3://redshift-downloads/ticket/spectrum/sales/'  
table properties ('numRows'='170000');
```

下列範例會建立使用 JsonSerDe 以 JSON 格式參考資料的資料表。

```
create external table spectrum.cloudtrail_json (  
  event_version int,  
  event_id bigint,
```

```

event_time timestamp,
event_type varchar(10),
awsregion varchar(20),
event_name varchar(max),
event_source varchar(max),
requesttime timestamp,
useragent varchar(max),
recipientaccountid bigint)
row format serde 'org.openx.data.jsonserde.JsonSerDe'
with serdeproperties (
'dots.in.keys' = 'true',
'mapping.requesttime' = 'requesttimestamp'
) location 's3://amzn-s3-demo-bucket/json/cloudtrail';

```

下列 CREATE EXTERNAL TABLE AS 範例會建立非分割外部資料表。然後，它會將SELECT查詢的結果作為 Apache Parquet 寫入目標 Amazon S3 位置。

```

CREATE EXTERNAL TABLE spectrum.lineitem
STORED AS parquet
LOCATION 'S3://amzn-s3-demo-bucket/cetas/lineitem/'
AS SELECT * FROM local_lineitem;

```

下列範例會建立分割的外部資料表，並在SELECT查詢中包含分割區資料欄。

```

CREATE EXTERNAL TABLE spectrum.partitioned_lineitem
PARTITIONED BY (l_shipdate, l_shipmode)
STORED AS parquet
LOCATION 'S3://amzn-s3-demo-bucket/cetas/partitioned_lineitem/'
AS SELECT l_orderkey, l_shipmode, l_shipdate, l_partkey FROM local_table;

```

如需外部資料目錄中現有資料庫的清單，請查詢 [SVV_EXTERNAL_DATABASES](#) 系統畫面。

```

select eskind,databasename,esoptions from svv_external_databases order by databasename;

```

```

eskind | databasename | esoptions
-----+-----
+-----
      1 | default      | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
      1 | sampledb     | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}

```

```
1 | spectrumdb | {"REGION":"us-
west-2","IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

若要檢視外部資料表的詳細資訊，請查詢 [SVV_EXTERNAL_TABLES](#) 和 [SVV_EXTERNAL_COLUMNS](#) 系統畫面。

下列範例會查詢 SVV_EXTERNAL_TABLES 檢視。

```
select schemaname, tablename, location from svv_external_tables;
```

schemaname	tablename	location
spectrum	sales	s3://redshift-downloads/ticket/spectrum/sales
spectrum	sales_part	s3://redshift-downloads/ticket/spectrum/sales_partition

下列範例會查詢 SVV_EXTERNAL_COLUMNS 檢視。

```
select * from svv_external_columns where schemaname like 'spectrum%' and tablename = 'sales';
```

schemaname	tablename	columnname	external_type	columnnum	part_key
spectrum	sales	salesid	int	1	0
spectrum	sales	listid	int	2	0
spectrum	sales	sellerid	int	3	0
spectrum	sales	buyerid	int	4	0
spectrum	sales	eventid	int	5	0
spectrum	sales	saledate	date	6	0
spectrum	sales	qtysold	smallint	7	0
spectrum	sales	pricepaid	decimal(8,2)	8	0
spectrum	sales	commission	decimal(8,2)	9	0
spectrum	sales	saletime	timestamp	10	0

若要檢視資料表分割區，請使用下列查詢。

```
select schemaname, tablename, values, location
from svv_external_partitions
where tablename = 'sales_part';
```

schemaname	tablename	values	location
spectrum	sales_part	["2008-01-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01
spectrum	sales_part	["2008-02-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-02
spectrum	sales_part	["2008-03-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-03
spectrum	sales_part	["2008-04-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-04
spectrum	sales_part	["2008-05-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-05
spectrum	sales_part	["2008-06-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-06
spectrum	sales_part	["2008-07-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-07
spectrum	sales_part	["2008-08-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-08
spectrum	sales_part	["2008-09-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-09
spectrum	sales_part	["2008-10-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-10
spectrum	sales_part	["2008-11-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-11
spectrum	sales_part	["2008-12-01"]	s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-12

下列範例會傳回外部資料表相關的資料檔案大小總和。

```
select distinct "$path", "$size"
from spectrum.sales_part;
```

\$path	\$size
s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01/	1616
s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-02/	1444
s3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-02/	1444

資料分割範例

若要建立以日期分割的外部資料表，請執行以下命令。

```
create external table spectrum.sales_part(  
salesid integer,  
listid integer,  
sellerid integer,  
buyerid integer,  
eventid integer,  
dateid smallint,  
qtysold smallint,  
pricepaid decimal(8,2),  
commission decimal(8,2),  
saletime timestamp)  
partitioned by (saledate date)  
row format delimited  
fields terminated by '|'   
stored as textfile  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/'  
table properties ('numRows'='170000');
```

若要新增分割區，請執行下列ALTER TABLE命令。

```
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-01-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-01/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-02-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-02/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-03-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-03/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-04-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-04/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-05-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-05/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-06-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-06/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-07-01')  
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-07/';  
alter table spectrum.sales_part  
add if not exists partition (saledate='2008-08-01')
```

```
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-08/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-09-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-09/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-10-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-10/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-11-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-11/';
alter table spectrum.sales_part
add if not exists partition (saledate='2008-12-01')
location 's3://redshift-downloads/ticket/spectrum/sales_partition/saledate=2008-12/';
```

若要從分割資料表選取資料，請執行下列查詢。

```
select top 10 spectrum.sales_part.eventid, sum(spectrum.sales_part.pricepaid)
from spectrum.sales_part, event
where spectrum.sales_part.eventid = event.eventid
    and spectrum.sales_part.pricepaid > 30
    and saledate = '2008-12-01'
group by spectrum.sales_part.eventid
order by 2 desc;
```

eventid	sum
914	36173.00
5478	27303.00
5061	26383.00
4406	26252.00
5324	24015.00
1829	23911.00
3601	23616.00
3665	23214.00
6069	22869.00
5638	22551.00

若要檢視外部資料表分割區，請查詢 [SVV_EXTERNAL_PARTITIONS](#) 系統畫面。

```
select schemaname, tablename, values, location from svv_external_partitions
where tablename = 'sales_part';
```



```

schemaname | tablename | values          | location
-----+-----+-----
+-----
spectrum   | sales_part | ["2008-01-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-01
spectrum   | sales_part | ["2008-02-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-02
spectrum   | sales_part | ["2008-03-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-03
spectrum   | sales_part | ["2008-04-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-04
spectrum   | sales_part | ["2008-05-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-05
spectrum   | sales_part | ["2008-06-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-06
spectrum   | sales_part | ["2008-07-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-07
spectrum   | sales_part | ["2008-08-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-08
spectrum   | sales_part | ["2008-09-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-09
spectrum   | sales_part | ["2008-10-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-10
spectrum   | sales_part | ["2008-11-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-11
spectrum   | sales_part | ["2008-12-01"] | s3://redshift-downloads/ticket/spectrum/
sales_partition/saledate=2008-12

```

資料列格式範例

下列顯示指定AVRO格式儲存之資料檔案ROWFORMATSERDE參數的範例。

```

create external table spectrum.sales(salesid int, listid int, sellerid int,
  buyerid int, eventid int, dateid int, qtysold int, pricepaid decimal(8,2), comment
  VARCHAR(255))
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES ('avro.schema.literal'='{\"namespace\": \"dory.sample\", \"name\":
  \"dory_avro\", \"type\": \"record\", \"fields\": [{\"name\": \"salesid\", \"type\": \"int
  \"},
  {\"name\": \"listid\", \"type\": \"int\"},
  {\"name\": \"sellerid\", \"type\": \"int\"},
  {\"name\": \"buyerid\", \"type\": \"int\"},
  {\"name\": \"eventid\", \"type\": \"int\"},

```

```
{\"name\": \"dateid\", \"type\": \"int\"},
{\"name\": \"qtysold\", \"type\": \"int\"},
{\"name\": \"pricepaid\", \"type\": {\"type\": \"bytes\", \"logicalType\": \"decimal\",
  \"precision\": 8, \"scale\": 2}}, {\"name\": \"comment\", \"type\": \"string\"}}}')
STORED AS AVRO
location 's3://amzn-s3-demo-bucket/avro/sales' ;
```

下列顯示使用 指定ROWFORMATSERDE參數的範例 RegEx。

```
create external table spectrum.types(
cbigint bigint,
cbigint_null bigint,
cint int,
cint_null int)
row format serde 'org.apache.hadoop.hive.serde2.RegexSerDe'
with serdeproperties ('input.regex'='([^\x01]+)\x01([^\x01]+)\x01([^\x01]+)\x01([^\x01]+)')
stored as textfile
location 's3://amzn-s3-demo-bucket/regex/types';
```

下列顯示使用 Grok 指定ROWFORMATSERDE參數的範例。

```
create external table spectrum.grok_log(
timestamp varchar(255),
pid varchar(255),
loglevel varchar(255),
programe varchar(255),
message varchar(255))
row format serde 'com.amazonaws.glue.serde.GrokSerDe'
with serdeproperties ('input.format'='[DFEWI], \\[%{TIMESTAMP_ISO8601:timestamp} #
%{POSINT:pid:int}\\] *(?<loglevel>:DEBUG|FATAL|ERROR|WARN|INFO) -- +%{DATA:programe}:
%{GREEDYDATA:message}')
```

```
stored as textfile
location 's3://DOC-EXAMPLE-BUCKET/grok/logs';
```

下面範例會示範在 S3 儲存貯體中定義 Amazon S3 伺服器存取日誌。您可以使用 Redshift Spectrum 查詢 Amazon S3 存取日誌。

```
CREATE EXTERNAL TABLE spectrum.mybucket_s3_logs(
bucketowner varchar(255),
bucket varchar(255),
requestdatetime varchar(2000),
```

```

remoteip varchar(255),
requester varchar(255),
requested varchar(255),
operation varchar(255),
key varchar(255),
requesturi_operation varchar(255),
requesturi_key varchar(255),
requesturi_httpprotoversion varchar(255),
httpstatus varchar(255),
errorcode varchar(255),
bytessent bigint,
objectsize bigint,
totaltime varchar(255),
turnaroundtime varchar(255),
referrer varchar(255),
useragent varchar(255),
versionid varchar(255)
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
'input.regex' = '([^ ]*) ([^ ]*) \\[(.??)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
\\"([^ ]*)\\s*([^ ]*)\\s*([^ ]*)\\" (- |[^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
([^ ]*) (\\"[^\\"]*"\\") ([^ ]*).*$')
LOCATION 's3://amzn-s3-demo-bucket/s3logs';

```

下列顯示指定ION格式資料ROWFORMATSERDE參數的範例。

```

CREATE EXTERNAL TABLE tbl_name (columns)
ROW FORMAT SERDE 'com.amazon.ionhiveserde.IonHiveSerDe'
STORED AS
INPUTFORMAT 'com.amazon.ionhiveserde.formats.IonInputFormat'
OUTPUTFORMAT 'com.amazon.ionhiveserde.formats.IonOutputFormat'
LOCATION 's3://amzn-s3-demo-bucket/prefix'

```

資料處理範例

下列範例會存取檔案：[spi_global_rankings.csv](#)。您可以將 `spi_global_rankings.csv` 檔案上傳至 Amazon S3 儲存貯體以嘗試下列範例。

以下範例會建立外部結構描述 `schema_spectrum_uddh` 和資料庫 `spectrum_db_uddh`。在中 `aws-account-id`，輸入 AWS 您的帳戶 ID 和 `role-name` 輸入您的 Redshift Spectrum 角色名稱。

```
create external schema schema_spectrum_uddh
from data catalog
database 'spectrum_db_uddh'
iam_role 'arn:aws:iam::aws-account-id:role/role-name'
create external database if not exists;
```

以下範例會在外部結構描述 `schema_spectrum_uddh` 中建立外部資料表 `soccer_league`。

```
CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league
(
  league_rank smallint,
  prev_rank smallint,
  club_name varchar(15),
  league_name varchar(20),
  league_off decimal(6,2),
  league_def decimal(6,2),
  league_spi decimal(6,2),
  league_nspi integer
)
ROW FORMAT DELIMITED
  FIELDS TERMINATED BY ','
  LINES TERMINATED BY '\n\\1'
stored as textfile
LOCATION 's3://spectrum-uddh/league/'
table properties ('skip.header.line.count'='1');
```

檢查 `soccer_league` 資料表中的列數。

```
select count(*) from schema_spectrum_uddh.soccer_league;
```

列數會隨其顯示。

```
count
645
```

下列查詢會顯示前 10 名的俱樂部。由於分會的字串中有 Barcelona 無效的字元，NULL 因此會顯示名稱的。

```
select league_rank, club_name, league_name, league_nspi
from schema_spectrum_uddh.soccer_league
```

```
where league_rank between 1 and 10;
```

```
league_rank club_name league_name league_nspi
1 Manchester City Barclays Premier Lea 34595
2 Bayern Munich German Bundesliga 34151
3 Liverpool Barclays Premier Lea 33223
4 Chelsea Barclays Premier Lea 32808
5 Ajax Dutch Eredivisie 32790
6 Atletico Madrid Spanish Primera Divi 31517
7 Real Madrid Spanish Primera Divi 31469
8 NULL Spanish Primera Divi 31321
9 RB Leipzig German Bundesliga 31014
10 Paris Saint-Ger French Ligue 1 30929
```

下列範例會修改 `soccer_league` 資料表，以指定 `invalid_char_handling`、`replacement_char` 和 `data_cleansing_enabled` 外部資料表屬性來插入問號 (?) 替代非預期字元。

```
alter table schema_spectrum_uddh.soccer_league
set table properties
('invalid_char_handling'='REPLACE','replacement_char'='?', 'data_cleansing_enabled'='true');
```

下列範例會針對排名 1 到 10 的團隊查詢 `soccer_league` 資料表。

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

由於資料表屬性已修改，因此結果會顯示前 10 名的俱樂部，並在俱樂部 Barcelona 的第八列包含問號 (?) 替代字元。

```
league_rank club_name league_name league_nspi
1 Manchester City Barclays Premier Lea 34595
2 Bayern Munich German Bundesliga 34151
3 Liverpool Barclays Premier Lea 33223
4 Chelsea Barclays Premier Lea 32808
5 Ajax Dutch Eredivisie 32790
6 Atletico Madrid Spanish Primera Divi 31517
7 Real Madrid Spanish Primera Divi 31469
8 Barcel?na Spanish Primera Divi 31321
9 RB Leipzig German Bundesliga 31014
```

```
10 Paris Saint-Ger French Ligue 1 30929
```

下列範例會修改資料表 `soccer_league` 來指定 `invalid_char_handling` 外部資料表屬性，以捨棄含有非預期字元的資料列。

```
alter table schema_spectrum_uddh.soccer_league
set table properties
('invalid_char_handling'='DROP_ROW','data_cleansing_enabled'='true');
```

下列範例會針對排名 1 到 10 的團隊查詢 `soccer_league` 資料表。

```
select league_rank,club_name,league_name,league_nspi
from schema_spectrum_uddh.soccer_league
where league_rank between 1 and 10;
```

結果會顯示熱門俱樂部，不包括俱樂部 Barcelona 的第八列。

league_rank	club_name	league_name	league_nspi
1	Manchester City	Barclays Premier Lea	34595
2	Bayern Munich	German Bundesliga	34151
3	Liverpool	Barclays Premier Lea	33223
4	Chelsea	Barclays Premier Lea	32808
5	Ajax	Dutch Eredivisie	32790
6	Atletico Madrid	Spanish Primera Divi	31517
7	Real Madrid	Spanish Primera Divi	31469
9	RB Leipzig	German Bundesliga	31014
10	Paris Saint-Ger	French Ligue 1	30929

CREATE EXTERNAL VIEW

Data Catalog 視觀表預覽功能僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

在 Data Catalog 中建立檢視。Data Catalog 檢視是單一檢視結構描述，可搭配 Amazon Athena 和 Amazon 等其他 SQL 引擎使用 EMR。您可以從您選擇的引擎中查詢檢視。如需 Data Catalog 檢視的詳細資訊，請參閱 [建立 Data Catalog 檢視](#)。

語法

```
CREATE EXTERNAL VIEW schema_name.view_name [ IF NOT EXISTS ]  
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |  
 external_schema_name.view_name}  
AS query_definition;
```

參數

schema_name.view_name

連接至 AWS Glue 資料庫的結構描述，後面接著檢視的名稱。

PROTECTED

指定只有在 *query_definition* 中的查詢可以成功完成時，CREATEEXTERNALVIEW 命令才應完成。

IF NOT EXISTS

如果檢視不存在，則建立檢視。

catalog_name.schema_name.view_name | *awsdatacatalog.dbname.view_name* |
external_schema_name.view_name

建立視觀表時所使用的結構描述標記法。您可以指定使用您建立的 AWS Glue Data Catalog Glue 資料庫，或您建立的外部結構描述。如需詳細資訊 [CREATEEXTERNALSCHEMA](#)，請參閱 [CREATEDATABASE](#) 和。

query_definition

Amazon Redshift 執行以變更檢視的 SQL 查詢定義。

範例

下列範例會建立名為 `sample_schema.glue_data_catalog_view` 的 Data Catalog 視觀表。

```
CREATE EXTERNAL PROTECTED VIEW sample_schema.glue_data_catalog_view IF NOT EXISTS
```

```
AS SELECT * FROM sample_database.remote_table "remote-table-name";
```

CREATE FUNCTION

使用子SQLSELECT句或 Python 程式建立新的純量使用者定義函數 (UDF)。

如需詳細資訊和範例，請參閱 [Amazon Redshift 中的使用者定義函數](#)。

所需權限

您必須具有下列其中一種執行 CREATE OR REPLACE 的許可FUNCTION：

- 對於 CREATE FUNCTION：
 - 超級使用者可以使用受信任和不受信任的語言來建立函數。
 - 具有 CREATE 【 OR 】 REPLACE FUNCTION權限的使用者可以使用受信任的語言建立函數。
- 對於 REPLACE FUNCTION：
 - 超級使用者
 - 具有 CREATE 【 OR 】 REPLACE FUNCTION權限的使用者
 - 函數擁有者

語法

```
CREATE [ OR REPLACE ] FUNCTION f_function_name
( { [py_arg_name py_arg_data_type |
sql_arg_data_type ] [ , ... ] } )
RETURNS data_type
{ VOLATILE | STABLE | IMMUTABLE }
AS $$
  { python_program | SELECT_clause }
$$ LANGUAGE { plpythonu | sql }
```

參數

或 REPLACE

指定已有相同名稱和輸入引數資料類型 (或簽章) 的函數存在時，取代現有函數。您可以將函數取代為定義一組相同資料類型的新函數。您必須是超級使用者才能取代函數。

如果您定義的函數與現有函數同名，但簽章不同，則會建立新函數。換言之，函數名稱將會過載。如需詳細資訊，請參閱[多載函數名稱](#)。

f_function_name

函數的名稱。如果您指定結構描述名稱 (例如 myschema.myfunction)，則會使用指定的結構描述建立函數。否則，函數會在目前結構描述中建立。如需有效名稱的相關資訊，請參閱[名稱與識別碼](#)。

建議您使用 在所有UDF名稱前面加上 f_。Amazon Redshift 保留UDF名稱的f_字首，因此使用f_字首，您可以確保UDF您的姓名不會與任何現有或未來的 Amazon Redshift 內建SQL函數名稱衝突。如需詳細資訊，請參閱[防止UDF命名衝突](#)。

若輸入引數的資料類型不同，則您可以定義多個擁有相同函數名稱的函數。換言之，函數名稱將會過載。如需詳細資訊，請參閱[多載函數名稱](#)。

py_arg_name py_arg_data_type | sql_arg_data_type

對於 Python UDF，輸入引數名稱和資料類型的清單。對於 SQL UDF，列出沒有引數名稱的資料類型。在 Python 中UDF，使用引數名稱參考引數。在 SQL 中UDF，根據引數清單中的引數順序，使用 \$1、\$2 等引數。

對於 SQL UDF，輸入和傳回資料類型可以是任何標準 Amazon Redshift 資料類型。對於 Python UDF，輸入和傳回資料類型可以是 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLEPRECISION、BOOLEANCHAR、VARCHAR、DATE或 TIMESTAMP。此外，Python 使用者定義函數 (UDFs) 支援資料類型 ANYELEMENT。這會根據執行時提供的對應引數資料類型，自動轉換為標準資料類型。如果多個引數使用 ANYELEMENT，它們都會根據清單中的第一個ANYELEMENT引數，在執行時間解析為相同的資料類型。如需詳細資訊，請參閱[Python UDF 資料類型](#) 和 [資料類型](#)。

您最多可以指定 32 個引數。

RETURNS data_type

函數所傳回值的資料類型。RETURNS 資料類型可以是任何標準 Amazon Redshift 資料類型。此外，Python UDFs可以使用的資料類型ANYELEMENT，其會根據執行時提供的引數自動轉換為標準資料類型。如果您ANYELEMENT為傳回資料類型指定，則至少有一個引數必須使用 ANYELEMENT。實際傳回資料類型符合呼叫函數時為ANYELEMENT引數提供的資料類型。如需詳細資訊，請參閱[Python UDF 資料類型](#)。

VOLATILE | STABLE | IMMUTABLE

通知查詢最佳化工具有關函數的波動情形。

如果您將函數標示為最嚴格的有效波動類別，將會得到最理想的最佳化結果。不過，如果類別太嚴格，則最佳化工具可能會錯誤地略過某些呼叫，導致產生不正確的結果集。從最低嚴格程度開始，依嚴格程度排列的波動類別如下所示：

- VOLATILE
- STABLE
- IMMUTABLE

VOLATILE

假設引數相同，即使是針對單一陳述式中的資料列，函數也可能在後續呼叫中傳回不同的結果。查詢最佳化工具無法對波動函數的行為做出任何假設，因此，使用波動函數的查詢必須針對每個輸入資料列重新評估函數。

STABLE

假設引數相同，函數一定會針對單一陳述式內處理的所有資料列傳回相同結果。在不同陳述式中呼叫函數時，函數可能傳回不同結果。此類別可讓最佳化工具將單一陳述式中的多次函數呼叫最佳化，成為陳述式的單一呼叫。

IMMUTABLE

假設引數相同，函數一律傳回相同結果，而且永遠不變。當查詢呼叫具有常數引數的 IMMUTABLE 函數時，最佳化工具會預先評估函數。

AS \$\$ statement \$\$

包圍要執行之陳述式的結構。常值關鍵字 AS \$\$ 和 \$\$ 是必要的。

Amazon Redshift 會要求您使用稱為 \$ 符號引用的格式包圍函數中的陳述式。包圍範圍當中的任何內容都會原封不動傳遞。您不需要逸出任何特殊字元，因為字串的內容是逐字撰寫。

使用 \$ 符號引用時，您會使用一組 \$\$ 符號配對表示要執行之陳述式的開頭和結尾，如以下範例所示。

```
$$ my statement $$
```

您也可以選擇在每組配對的兩個 \$ 符號之間指定字串來協助識別陳述式。您在包圍配對的開頭和結尾中使用的字串必須相同。此字串區分大小寫，且遵循與未加引號的識別碼相同的限制條件，不過後者不可包含 \$ 符號。下列範例使用字串 test。

```
$test$ my statement $test$
```

如需美元報價的詳細資訊，請參閱 PostgreSQL 文件的 [Lexical Structure](#) 下的「美元報價字串常數」。

python_program

傳回值的有效可執行 Python 程式。隨函數傳入的陳述式必須符合縮排要求，如 Python 網站上的《Python 程式碼格式指南》<https://www.python.org/dev/peps/pep-0008/#indentation> 所述。如需詳細資訊，請參閱的 [Python 語言支援 UDFs](#)。

SQL_clause

SQL SELECT 子句。

SELECT 子句不能包含下列任一類型的子句：

- FROM
- INTO
- WHERE
- GROUP 依 排序
- ORDER 依 排序
- LIMIT

```
LANGUAGE { plpythonu | sql }
```

若是 Python，指定 plpythonu。針對 SQL，指定 sql。您必須具有 SQL 或 plpythonu 使用語言的許可。如需詳細資訊，請參閱 [UDF 安全性和許可](#)。

使用須知

巢狀函數

您可以從內呼叫其他 SQL 使用者定義的函數 SQL (UDF) UDF。當您執行 CREATEFUNCTION 命令時，巢狀函數必須存在。Amazon Redshift 不會追蹤的相依性UDFs，因此如果您捨棄巢狀函數，Amazon Redshift 不會傳回錯誤。不過，如果巢狀函數不存在，UDF 將會失敗。例如，下列函數會呼叫 SELECT 子句中的 f_sql_greater 函數。

```
create function f_sql_commission (float, float )
```

```
returns float
stable
as $$
  select f_sql_greater ($1, $2)
$$ language sql;
```

UDF 安全性和權限

若要建立 UDF，您必須具有 SQL 或 ppythonu (Python) 使用語言的許可。根據預設，USAGE ON LANGUAGE SQL 會授予 PUBLIC。不過，您必須明確 LANGUAGE PLPYTHONU 地將 USAGE ON 授予特定使用者或群組。

若要撤銷 的使用 SQL，請先撤銷 的使用 PUBLIC。然後 SQL 僅將 上的用量授予允許建立 SQL 的特定使用者或群組 UDFs。下列範例 SQL 會從 撤銷 上的用量，PUBLIC 然後將用量授與使用者群組 udf_devs。

```
revoke usage on language sql from PUBLIC;
grant usage on language sql to group udf_devs;
```

若要執行 UDF，您必須具有每個函數的執行許可。根據預設，UDFs 會將新的執行許可授予 PUBLIC。若要限制用量，請撤銷 對函數 PUBLIC 的執行許可。然後將許可授予特定個人或群組。

下列範例 f_py_greater 會從 撤銷對函數的執行許可，PUBLIC 然後將用量授予使用者群組 udf_devs。

```
revoke execute on function f_py_greater(a float, b float) from PUBLIC;
grant execute on function f_py_greater(a float, b float) to group udf_devs;
```

根據預設，超級使用者具備所有權限。

如需詳細資訊，請參閱 [GRANT](#) 和 [REVOKE](#)。

範例

Scalar Python UDF 範例

下列範例會建立 Python UDF，比較兩個整數並傳回較大的值。

```
create function f_py_greater (a float, b float)
```

```
    returns float
stable
as $$
    if a > b:
        return a
    return b
$$ language plpythonu;
```

下列範例會查詢SALES資料表並呼叫新f_py_greater函數，以傳回 COMMISSION或 20% 的 PRICEPAID，以較大者為準。

```
select f_py_greater (commission, pricepaid*0.20) from sales;
```

純量SQLUDF範例

下列範例會建立一個函數，比較兩個數字並傳回較大的值。

```
create function f_sql_greater (float, float)
    returns float
stable
as $$
    select case when $1 > $2 then $1
        else $2
    end
$$ language sql;
```

下列查詢會呼叫新f_sql_greater函數來查詢SALES資料表，並傳回 COMMISSION或 20%PRICEPAID，以較大者為準。

```
select f_sql_greater (commission, pricepaid*0.20) from sales;
```

CREATE GROUP

定義新的使用者群組。只有超級使用者才能建立群組。

語法

```
CREATE GROUP group_name
[ [ WITH ] [ USER username ] [, ...] ]
```

參數

group_name

新使用者群組的名稱。開頭為兩個底線的群組名稱保留供 Amazon Redshift 內部使用。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

WITH

選擇性語法，用於指示 CREATE 的其他參數GROUP。

USER

在群組中新增一個或多個使用者。

username

要新增至群組的使用者名稱。

範例

下列範例會建立名為 ADMIN_GROUP 的使用者群組，其中包含兩個使用者 ADMIN1和 ADMIN2。

```
create group admin_group with user admin1, admin2;
```

CREATE IDENTITY PROVIDER

定義新的身分提供者。只有超級使用者可以建立身分提供者。

語法

```
CREATE IDENTITY PROVIDER identity_provider_name TYPE type_name  
NAMESPACE namespace_name  
[PARAMETERS parameter_string]  
[APPLICATION_ARN arn]  
[IAM_ROLE iam_role]
```

參數

identity_provider_name

新身分提供者的名稱。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

type_name

要與之連接的身分提供者。Azure 是目前唯一支援的身分提供者。

namespace_name

命名空間 這是身分提供者目錄的唯一速記識別碼。

parameter_string

包含正確格式化JSON物件的字串，其中包含身分提供者所需的參數和值。

arn

IAM Identity Center 受管應用程式的 Amazon 資源名稱 (ARN)。此參數僅適用於 identity-provider 類型為 時 AWSIDC。

iam_role

提供許可以連線至 IAM Identity Center IAM的角色。此參數僅適用於 identity-provider 類型為 時 AWSIDC。

範例

下列範例會使用 azure 建立名為 oauth_standard TYPE 的身分提供者，以與 Microsoft Azure Active Directory (AD) 建立通訊。

```
CREATE IDENTITY PROVIDER oauth_standard TYPE azure
NAMESPACE 'aad'
PARAMETERS '{"issuer":"https://sts.windows.net/2sdfdsf-d475-420d-b5ac-667adad7c702/",
"client_id":"87f4aa26-78b7-410e-bf29-57b39929ef9a",
"client_secret":"BUAH~ewrqewrqwerUUY^%tHe1oNZShoiU7",
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift"]}
}'
```

您可以將 IAM Identity Center 受管應用程式與現有的佈建叢集或 Amazon Redshift Serverless 工作群組連線。這可讓您透過 IAM Identity Center 管理對 Redshift 資料庫的存取。若要這麼做，請執行類似下列範例的SQL命令。您必須是資料庫管理員。

```
CREATE IDENTITY PROVIDER "redshift-idc-app" TYPE AWSIDC
NAMESPACE 'awsidc'
APPLICATION_ARN 'arn:aws:sso::123456789012:application/ssoins-12345f67fe123d4/apl-a0b0a12dc123b1a4'
```

```
IAM_ROLE 'arn:aws:iam::123456789012:role/MyRedshiftRole';
```

ARN 在此情況下，應用程式會識別要連線的受管應用程式。您可以執行 `SELECT * FROM SVV_IDENTITY_PROVIDERS;`

如需使用 CREATE IDENTITY 的詳細資訊 PROVIDER，包括其他範例，請參閱 [Amazon Redshift 的原始身分提供者 \(IdP\) 聯合](#)。如需從 Redshift 設定與 IAM Identity Center 連線的詳細資訊，請參閱 [使用 IAM Identity Center 連接 Redshift，為使用者提供單一登入體驗](#)。

CREATE LIBRARY

安裝 Python 程式庫，該程式庫可供使用者在建立使用 [CREATE FUNCTION](#) 命令的使用者定義函數 (UDF) 時加以整合。使用者安裝的程式庫的總和大小不得超過 100 MB。

CREATE LIBRARY 無法在交易區塊內執行 (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

Amazon Redshift 支援 Python 2.7 版。如需詳細資訊，請參閱 www.python.org。

如需詳細資訊，請參閱 [範例：匯入自訂 Python 程式庫模組](#)。

所需權限

以下是 CREATE 的必要權限 LIBRARY：

- 超級使用者
- 具有 CREATE LIBRARY 權限或具有指定語言權限的使用者

語法

```
CREATE [ OR REPLACE ] LIBRARY library_name LANGUAGE plpythonu
FROM
{ 'https://file_url'
| 's3://bucketname/file_name'
authorization
  [ REGION [AS] 'aws_region' ]
  IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
}
```


參數

或 REPLACE

指定已有相同名稱的程式庫存在時，取代現有程式庫。REPLACE 立即遞交。如果UDF依賴程式庫的同時執行，即使UDF是在交易中執行，UDF也可能會失敗或傳回非預期的結果。您必須是擁有者或超級使用者才能取代程式庫。

library_name

要安裝的程式庫名稱。您建立的程式庫不可包含與 Python 標準程式庫模組或 Amazon Redshift 預先安裝 Python 模組同名的模組。如果現有的使用者安裝程式庫使用與所安裝程式庫相同的 Python 套件，則您必須先捨棄現有的程式庫，才能安裝新的程式庫。如需詳細資訊，請參閱[Python 語言支援 UDFs](#)。

LANGUAGE 普皮索諾

要使用的語言。Python (plpythonu) 是唯一支援的語言。Amazon Redshift 支援 Python 2.7 版。如需詳細資訊，請參閱 www.python.org。

FROM

程式庫檔案的位置。您可以指定 Amazon S3 儲存貯體和物件名稱，也可以指定 URL 從公有網站下載檔案。程式庫必須封裝為 .zip 檔案。如需詳細資訊，請參閱 Python 文件中的[建構和安裝 Python 模組](#)。

https://file_url

從公有網站URL下載檔案的。最多可URL包含三個重新導向。以下是檔案的範例URL。

```
'https://www.example.com/pylib.zip'
```

s3://bucket_name/file_name

單一 Amazon S3 物件的路徑，物件當中包含程式庫檔案。以下是 Amazon S3 物件路徑的範例。

```
's3://amzn-s3-demo-bucket/my-pylib.zip'
```

如果您指定 Amazon S3 儲存貯體，則必須同時提供有下載檔案許可之 AWS 使用者的登入資料。

⚠ Important

如果 Amazon S3 儲存貯體與 Amazon Redshift 叢集不位於相同的 AWS 區域，您必須使用 REGION 選項來指定資料所在的 AWS 區域。aws_region 的值必須符合 COPY 命令 [REGION](#) 參數描述中資料表中列出的 AWS 區域。

authorization

此子句指出叢集在身分驗證和授權存取包含程式庫檔案的 Amazon S3 儲存貯體時使用的方法。您的叢集必須具有使用 LIST 和 GET 動作存取 Amazon S3 的許可。

授權的語法與 COPY 命令授權相同。如需詳細資訊，請參閱 [授權參數](#)。

```
IAM_ROLE { default | 'arn:aws:iam::<AWS ##-id>:role/<role-name>' }
```

使用預設關鍵字讓 Amazon Redshift 在 CREATE LIBRARY 命令執行時，使用設定為預設並與叢集相關聯的 IAM 角色。

針對叢集用於身分驗證和授權 IAM 的角色，使用 Amazon Resource Name (ARN)。如果您指定 IAM_ROLE，則無法使用 ACCESS_KEY_ID 和 SECRET_ACCESS_KEY、SESSION_TOKEN 或 CREDENTIALS。

或者，如果 Amazon S3 儲存貯體使用的是伺服器端加密，請提供 credentials-args 字串中的加密金鑰。如果您使用的是臨時安全登入資料，請提供 credentials-args 字串中的暫時字符。

如需詳細資訊，請參閱 [暫時安全憑證](#)。

REGION 【AS】 aws_region

Amazon S3 儲存貯體所在的 AWS 區域。REGION 當 Amazon S3 儲存貯體不在與 Amazon Redshift 叢集相同的 AWS 區域中時，是必要的。aws_region 的值必須符合 COPY 命令 [REGION](#) 參數描述中資料表中列出的 AWS 區域。

根據預設，會 CREATE LIBRARY 假設 Amazon S3 儲存貯體與 Amazon Redshift 叢集位於相同的 AWS 區域。

範例

以下兩個範例會安裝 [urlparse](#) Python 模組，它會封裝成名為 urlparse3-1.0.3.zip 的檔案。

下列命令會從已上傳至美國東部區域的 Amazon S3 儲存貯體的 `f_urlparse` 套件安裝名為 `f_urlparse` 的 UDF 程式庫。

```
create library f_urlparse
language plpythonu
from 's3://amzn-s3-demo-bucket/urlparse3-1.0.3.zip'
credentials 'aws_iam_role=arn:aws:iam::<aws-account-id>:role/<role-name>'
region as 'us-east-1';
```

下列範例會從網站上的程式庫檔案安裝名為 `f_urlparse` 的程式庫。

```
create library f_urlparse
language plpythonu
from 'https://example.com/packages/urlparse3-1.0.3.zip';
```

CREATE MASKING POLICY

建立新的動態資料遮罩政策，以模糊指定格式的資料。如需動態資料遮罩的相關資訊，請參閱 [動態資料遮罩](#)。

超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以建立遮罩政策。

語法

```
CREATE MASKING POLICY
  policy_name [IF NOT EXISTS]
  WITH (input_columns)
  USING (masking_expression);
```

參數

`policy_name`

遮罩政策的名稱。遮罩政策的名稱不能與資料庫中已存在的另一個遮罩政策相同。

`input_columns`

使用格式的資料欄名稱元組 (`col1` 類型、`col2` 類型 ...)。

資料欄名稱會用來做為遮罩表示式的輸入。資料欄名稱不一定要與要遮罩的資料欄名稱相符，但輸入和輸出資料類型必須相符。

masking_expression

用於轉換目標資料欄的 SQL 表達式。它可以使用資料操作函數編寫，例如字串操作函數，或與使用者定義的函數一起寫入 SQL、Python 或 AWS Lambda。您可以針對具有多個輸出的遮罩政策包含資料欄運算式的元組。如果您使用常數做為遮罩運算式，則必須將其明確轉換為符合輸入類型的類型。

您必須擁有遮罩表達式中使用的任何使用者定義函數的 USAGE 許可。

CREATE MATERIALIZED VIEW

根據一個或多個 Amazon Redshift 資料表建立具體化視觀表。您也可以讓具體化視觀表以使用 Spectrum 或聯合查詢建立的外部資料表為依據。如需 Spectrum 的詳細資訊，請參閱 [Amazon Redshift Spectrum](#)。如需聯合查詢的詳細資訊，請參閱 [使用 Amazon Redshift 中的聯合查詢來查詢資料](#)。

語法

```
CREATE MATERIALIZED VIEW mv_name
[ BACKUP { YES | NO } ]
[ table_attributes ]
[ AUTO REFRESH { YES | NO } ]
AS query
```

參數

BACKUP

指定具體化檢視是否應包含在自動和手動叢集快照中的子句。

對於不包含關鍵資料的具體化檢視，請指定 BACKUP NO，以在建立快照並從快照還原時節省處理時間，並減少 Amazon Simple Storage Service 上的儲存空間。BACKUP NO 設定不會影響資料自動複寫至叢集內其他節點，因此在節點故障的情況下，會還原具有指定 BACKUP NO 的具體化檢視。預設值為 BACKUP YES。

table_attributes

此子句指定如何分配具體化檢視中的資料，其中包括：

- 具體化檢視的分佈樣式，其格式為 DISTSTYLE { EVEN | ALL | KEY }。如果您省略此子句，則分佈樣式為 EVEN。如需詳細資訊，請參閱 [分佈樣式](#)。

- 具體化檢視的分佈索引鍵，其格式為 `DISTKEY (distkey_identifier)`。如需詳細資訊，請參閱[指定分佈樣式](#)。
- 具體化檢視的排序索引鍵，其格式為 `SORTKEY (column_name [, ...])`。如需詳細資訊，請參閱[排序金鑰](#)。

AS query

定義具體化視觀表及其內容的有效 `SELECT` 陳述式。查詢的結果集會定義具體化檢視的資料行與資料列。如需有關建立具體化檢視時有何限制的資訊，請參閱[限制](#)。

此外，查詢中使用的特定SQL語言建構會決定具體化視觀表是否可以遞增或完全重新整理。如需重新整理方法的相關資訊，請參閱[REFRESH MATERIALIZED VIEW](#)。如需增量重新整理之限制的相關資訊，請參閱[增量重新整理的限制](#)。

如果查詢包含不支援增量重新整理的SQL命令，Amazon Redshift 會顯示訊息，指出具體化檢視將使用完整重新整理。訊息不一定會顯示，具體取決於SQL用戶端應用程式。檢查 [STV_MV_INFO](#) 的 `state` 資料欄，以查看具體化檢視所使用的重新整理類型。

AUTO REFRESH

該子句會定義是否應使用基本資料表中的最新變更自動重新整理具體化視觀表。預設值為 `NO`。如需詳細資訊，請參閱[重新整理具體化視觀表](#)。

使用須知

若要建立具體化檢視，您必須具有下列權限：

- `CREATE` 結構描述的權限。
- 基礎資料表上的資料表層級或資料欄層級 `SELECT` 權限，以建立具體化檢視。如果您擁有特定資料欄層級的權限，就可以僅針對這些資料欄建立具體化視觀表。

資料共用中具體化視觀表的增量重新整理

共用基礎資料表時，Amazon Redshift 支援消費者資料共用中具體化視觀表的自動和增量重新整理。增量重新整理是 Amazon Redshift 識別在上一次重新整理後基礎資料表或資料表中發生的變更，並僅更新具體化檢視中對應的記錄的操作。這比完全重新整理更快執行，並改善工作負載效能。您不需要變更具體化視觀表定義，即可利用增量重新整理。

利用具體化視觀表進行增量重新整理需要注意幾個限制：

- 具體化檢視只能參考一個資料庫，無論是本機或遠端。
- 增量重新整理僅適用於新的具體化視觀表。因此，您必須捨棄現有的具體化檢視，並重新建立這些檢視，才能進行增量重新整理。

如需在資料共用中建立具體化檢視的詳細資訊，請參閱[使用 Amazon Redshift 資料共用 中的檢視](#)，其中包含數個查詢範例。

DDL 具體化視觀表或基礎資料表的更新

在 Amazon Redshift 中使用具體化視觀表時，請遵循這些有關資料定義語言（DDL）更新具體化視觀表或基礎資料表的用量備註。

- 您可以將資料欄新增到基礎資料表，而不影響任何參考該基礎資料表的具體化視觀表。
- 某些操作也可能使具體化檢視進入完全無法重新整理的狀態。例如：重新命名或卸除資料欄、變更資料欄類型，以及變更結構描述名稱等操作。這類具體化檢視可以進行查詢，但無法重新整理。在此情況下，您必須卸除並重新建立具體化檢視。
- 一般而言，您無法變更具體化視觀表的定義（其SQL陳述式）。
- 您無法將具體化檢視重新命名。

限制

您無法定義參考或包含以下任何一項的具體化檢視：

- 標準檢視，或系統資料表和檢視。
- 暫存資料表。
- 使用者定義的函數。
- ORDER BY、LIMIT或 OFFSET子句。
- 對基礎資料表的近期繫結參考。換句話說，定義具體化視觀表SQL查詢中參考的任何基礎資料表或相關資料欄必須存在且必須是有效的。
- 領導者節點限定函數：
CURRENT_SCHEMA、CURRENT_SCHEMAS、HAS_DATABASE_PRIVILEGE、HAS_SCHEMA_PRIVILEGE
- 當具體化視觀表定義包含可變函數或外部結構描述時，您就無法使用 AUTOREFRESHYES選項。當您在另一個具體化視觀表上定義具體化視觀表時，也無法使用此選項。
- 您不需要在具體化視觀表上手動執行 [ANALYZE](#)。目前僅透過 AUTO 發生這種情況ANALYZE。如需詳細資訊，請參閱[分析資料表](#)。

範例

下列範例會從三個加入和彙總的基礎資料表建立具體化視觀表。每一列代表具有售票數的類別。查詢 tickets_mv 具體化檢視時，您可以直接存取 tickets_mv 具體化檢視中的預先計算資料。

```
CREATE MATERIALIZED VIEW tickets_mv AS
  select  catgroup,
         sum(qtysold) as sold
  from    category c, event e, sales s
  where   c.catid = e.catid
  and     e.eventid = s.eventid
  group by catgroup;
```

下列範例會建立類似於先前範例的具體化檢視，並使用彙總函數 MAX ()。

```
CREATE MATERIALIZED VIEW tickets_mv_max AS
  select  catgroup,
         max(qtysold) as sold
  from    category c, event e, sales s
  where   c.catid = e.catid
  and     e.eventid = s.eventid
  group by catgroup;
```

```
SELECT name, state FROM STV_MV_INFO;
```

下列範例使用 UNIONALL子句來加入 Amazon Redshift public_sales資料表和 Redshift Spectrum spectrum.sales資料表，以建立材料檢視mv_sales_vw。如需有關 Amazon Redshift Spectrum CREATEEXTERNALTABLE命令的資訊，請參閱 [CREATE EXTERNAL TABLE](#)。Redshift Spectrum 外部資料表會參考 Amazon S3 上的資料。

```
CREATE MATERIALIZED VIEW mv_sales_vw as
  select salesid, qtysold, pricepaid, commission, saletime from public.sales
  union all
  select salesid, qtysold, pricepaid, commission, saletime from spectrum.sales
```

下列範例會根據聯合查詢外部資料表建立具體化檢視 mv_fq。如需聯合查詢的詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#)。

```
CREATE MATERIALIZED VIEW mv_fq as select firstname, lastname from apg.mv_fq_example;
```

```
select firstname, lastname from mv_fq;
  firstname | lastname
-----+-----
   John    |   Day
   Jane    |   Doe
(2 rows)
```

下列範例顯示具體化檢視的定義。

```
SELECT pg_catalog.pg_get_viewdef('mv_sales_vw'::regclass::oid, true);

pg_get_viewdef
-----
create materialized view mv_sales_vw as select a from t;
```

下列範例示範如何在具體化檢視定義AUTOREFRESH中設定，並指定DISTSTYLE。首先，建立一個簡單的基底資料表。

```
CREATE TABLE baseball_table (ball int, bat int);
```

然後建立具體化視觀表。

```
CREATE MATERIALIZED VIEW mv_baseball DISTSTYLE ALL AUTO REFRESH YES AS SELECT ball AS
baseball FROM baseball_table;
```

現在您可以查詢 mv_baseball 具體化視觀表。若要檢查具體化視觀表AUTOREFRESH是否已開啟，請參閱 [STV_MV_INFO](#)。

下列範例會建立參照另一個資料庫中來源資料表的具體化視觀表。假設包含來源資料表的資料庫 database_A 與您在 database_B 中建立的具體化視觀表位於相同叢集或工作群組中。(您可以將範例取代為自己的資料庫。) 首先，在 database_A 中建立一個名為 cities 的資料表，其中包含 cityname 資料欄。將資料欄的資料類型設為 VARCHAR。建立來源資料表之後，在 database_B 中執行下列命令，以建立其來源為 cities 資料表的具體化視觀表。請務必在 FROM 子句中指定來源資料表的資料庫和結構描述：

```
CREATE MATERIALIZED VIEW cities_mv AS
SELECT  cityname
FROM    database_A.public.cities;
```

查詢您建立的具體化視觀表。查詢會擷取其原始來源為 database_A 中 cities 資料表的記錄：


```
select * from cities_mv;
```

當您執行SELECT陳述式時，city_mv 會傳回記錄。只有在執行REFRESH陳述式時，才會從來源資料表重新整理記錄。此外，請注意，您無法直接在具體化視觀表中更新記錄。如需有關重新整理具體化視觀表中資料的資訊，請參閱 [REFRESH MATERIALIZED VIEW](#)。

如需具體化檢視概觀的詳細資訊，以及用於重新整理和捨棄具體化檢視的SQL命令，請參閱下列主題：

- [Amazon Redshift 中的具體化視觀表](#)
- [REFRESH MATERIALIZED VIEW](#)
- [DROP MATERIALIZED VIEW](#)

CREATE MODEL

主題

- [必要條件](#)
- [所需權限](#)
- [成本控制](#)
- [完整 CREATE MODEL](#)
- [參數](#)
- [使用須知](#)
- [使用案例](#)

必要條件

使用CREATEMODEL陳述式之前，請先完成 中的先決條件[使用 Amazon Redshift ML 的叢集設定](#)。以下是先決條件的概要。

- 使用 AWS 管理主控台或 AWS 命令列介面 () 建立 Amazon Redshift 叢集AWS CLI。
- 在建立叢集時連接 AWS Identity and Access Management (IAM) 政策。
- 若要允許 Amazon Redshift 和 SageMaker 擔任角色以與其他 服務互動，請將適當的信任政策新增至IAM角色。

如需IAM角色、信任政策和其他先決條件的詳細資訊，請參閱 [使用 Amazon Redshift ML 的叢集設定](#)。

接下來，您可以找到 CREATEMODEL 陳述式的不同使用案例。

- [簡單 CREATE MODEL](#)
- [CREATE MODEL 使用使用者指南](#)
- [CREATE XGBoost 具有的模型 AUTO OFF](#)
- [自備模型 \(BYOM\) - 本機推論](#)
- [使用您自己的模型 \(BYOM\) - 遠端推論](#)
- [CREATE MODEL 使用 K-MEANS](#)
- [完整 CREATE MODEL](#)

所需權限

以下是 CREATE 的必要權限MODEL：

- 超級使用者
- 具有 CREATEMODEL 權限的使用者
- 具有 GRANTCREATEMODEL 權限的角色

成本控制

Amazon Redshift ML 會使用現有的叢集資源建立預測模型，因此您不必支付額外費用。但是，如果您需要調整叢集大小或想要訓練模型，則可能會產生額外費用。Amazon Redshift ML 使用 Amazon SageMaker 來訓練模型，這些模型確實有額外的關聯成本。有一些方法可以控制額外成本，例如限制訓練所耗用的時間上限，或限制用於訓練模型的訓練範例數量。如需詳細資訊，請參閱[使用 Amazon Redshift 資料 API](#)。

完整 CREATE MODEL

下列摘要說明完整CREATEMODEL語法的基本選項。

完整CREATEMODEL語法

以下是 CREATEMODEL 陳述式的完整語法。

Important

使用 CREATEMODEL 陳述式建立模型時，請依照下列語法中關鍵字順序進行。

```

CREATE MODEL model_name
FROM { table_name | ( select_statement ) | 'job_name' }
[ TARGET column_name ]
FUNCTION function_name [ ( data_type [, ...] ) ]
[ RETURNS data_type ]
  -- supported only for BYOM
[ SAGEMAKER 'endpoint_name':['model_name']]
  -- supported only for BYOM remote inference
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
[ AUTO ON / OFF ]
  -- default is AUTO ON
[ MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER | KMEANS | FORECAST } ]
  -- not required for non AUTO OFF case, default is the list of all supported types
  -- required for AUTO OFF
[ PROBLEM_TYPE ( REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION ) ]
  -- not supported when AUTO OFF
[ OBJECTIVE ( 'MSE' | 'Accuracy' | 'F1' | 'F1_Macro' | 'AUC' |
             'reg:squarederror' | 'reg:squaredlogerror' | 'reg:logistic' |
             'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' |
             'binary:hinge',
             'multi:softmax' | 'RMSE' | 'WAPE' | 'MAPE' | 'MASE' |
             'AverageWeightedQuantileLoss' ) ]
  -- for AUTO ON: first 5 are valid
  -- for AUTO OFF: 6-13 are valid
  -- for FORECAST: 14-18 are valid
[ PREPROCESSORS 'string' ]
  -- required for AUTO OFF, when it has to be 'none'
  -- optional for AUTO ON
[ HYPERPARAMETERS { DEFAULT | DEFAULT EXCEPT ( Key 'value' (, ...) ) } ]
  -- support XGBoost hyperparameters, except OBJECTIVE
  -- required and only allowed for AUTO OFF
  -- default NUM_ROUND is 100
  -- NUM_CLASS is required if objective is multi:softmax (only possible for AUTO OFF)
[ SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket', |
    -- required
  TAGS 'string', |
    -- optional
  KMS_KEY_ID 'kms_string', |
    -- optional
  S3_GARBAGE_COLLECT on / off, |
    -- optional, default is on.
  MAX_CELLS integer, |

```

```

-- optional, default is 1,000,000
MAX_RUNTIME integer (, ...) |
-- optional, default is 5400 (1.5 hours)
HORIZON integer, |
-- required if creating a forecast model
FREQUENCY integer, |
-- required if creating a forecast model
PERCENTILES string, |
-- optional if creating a forecast model
MAX_BATCH_ROWS integer -- optional for BYOM remote inference
) ]

```

參數

`model_name`

模型的名稱。結構描述中的模型名稱必須是唯一的。

`FROM { table_name | (select_query) | 'job_name' }`

指定訓練資料的 `table_name` 或查詢。它們可以是系統中現有的資料表，也可以是括號隨附的 Amazon Redshift 相容 SELECT 查詢，也就是 ()。查詢結果中至少必須有兩個資料列。

`TARGET column_name`

成為預測目標的資料欄名稱。資料欄必須存在於 FROM 子句中。

`FUNCTION function_name (data_type [, ...])`

要建立的函數名稱和輸入引數的資料類型。您可以在資料庫中提供結構描述的結構描述名稱，而不是函數名稱。

`RETURNS data_type`

從模型函數傳回的資料類型。傳回的 SUPER 資料類型僅適用於 BYOM 具有遠端推論的。

`SAGEMAKER 'endpoint_name' [: 'model_name']`

Amazon SageMaker 端點的名稱。如果端點名稱指向多模型端點，請新增要使用的模型名稱。端點必須託管在 AWS 區域與 Amazon Redshift 叢集相同的 中。

`IAM_ROLE { 預設 | 'arn : aws : iam : : <account-id> : role/<role-name>' }`

使用預設關鍵字讓 Amazon Redshift 在 CREATE MODEL 命令執行時，使用設定為預設並與叢集相關聯的 IAM 角色。或者，您可以指定 IAM 角色 ARN 的 來使用該角色。

【 AUTO ON / OFF 】

開啟或關閉CREATEMODEL自動探索前置處理器、演算法和超參數選擇。建立預測模型時在上指定表示使用AutoPredictor，其中Amazon Forecast會將演算法的最佳組合套用至資料集中的每個時間序列。

MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER | KMEANS | FORECAST }

(選擇性) 指定模型類型。您可以指定是否要訓練特定模型類型的模型，例如 XGBoost、多層 Perceptron (MLP) KMEANS、或 Linear Learner，這些都是 Amazon SageMaker Autopilot 支援的演算法。如果未指定參數，則會在訓練期間搜尋所有支援的模型類型，以取得最佳模型。您也可以 Redshift ML 中建立預測模型，以建立準確的時間序列預測。

PROBLEM_TYPE (REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION)

(選擇性) 指定問題類型。如果您知道問題類型，則可以將 Amazon Redshift 限制為僅搜尋該特定模型類型的最佳模型。如果未指定此參數，則訓練期間會根據您的資料探索問題類型。

OBJECTIVE ('MSE' | 'Accuracy' | 'F1Macro' | 'AUC' | 'reg : squarederror' | 'reg : squaredlogerror' | 'reg : logistic' | 'reg : pseudohubererror' | 'reg : tweedie' | 'binary : logistic' | 'binary : hinge' | 'multi : softmax' | 'RMSE' | 'WAPE' | 'MAPE' | 'MASE' | 'AverageWeightedQuantileLoss')

(選擇性) 指定用來測量機器學習系統預測品質的目標指標名稱。此指標會在訓練期間進行最佳化，從資料中提供模型參數值的最佳估計值。如果您未明確指定指標，預設行為是自動將 MSE：用於迴歸、F1：用於二進位分類、準確性：用於多類別分類。如需目標的詳細資訊，請參閱 XGBOOST 文件中的 Amazon SageMaker API 參考和[學習任務參數](#)中的 [utoMLJob](#) 目標。這些值 RMSE、WAPE、MASE、MAPE 和 AverageWeightedQuantileLoss 僅適用於預測模型。如需詳細資訊，請參閱 [CreateAutoPredictor](#) API 操作。

PREPROCESSORS '字串'

(選擇性) 指定特定資料欄集的特定預處理器組合。格式是 的清單 columnSets，且要套用至每組資料欄的適當轉換。Amazon Redshift 會將特定轉換器清單中的所有轉換器套用至對應 中的所有資料欄 ColumnSet。例如，若要 OneHotEncoder 將 Imputer 套用至資料欄 t1 和 t2，請使用下列範例命令。

```
CREATE MODEL customer_churn
FROM customer_data
TARGET 'Churn'
FUNCTION predict_churn
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
PROBLEM_TYPE BINARY_CLASSIFICATION
```

```

OBJECTIVE 'F1'
PREPROCESSORS '['
...
  {"ColumnSet": [
    "t1",
    "t2"
  ],
  "Transformers": [
    "OneHotEncoder",
    "Imputer"
  ]
},
  {"ColumnSet": [
    "t3"
  ],
  "Transformers": [
    "OneHotEncoder"
  ]
},
  {"ColumnSet": [
    "temp"
  ],
  "Transformers": [
    "Imputer",
    "NumericPassthrough"
  ]
}
]'
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket'
)

```

HYPERPARAMETERS { DEFAULT | DEFAULTEXCEPT (鍵「值」 (, ..)) }

指定預設XGBoost參數是否由使用者指定的值使用或覆寫。這些值必須用單引號括住。以下是XGBoost及其預設值的參數範例。

參數名稱	參數值	預設值	備註
num_class	Integer	多類 別分	N/A

參數名稱	參數值	預設值	備註
		類所需。	
num_round	Integer	100	N/A
tree_method	字串	Auto	N/A
max_depth	Integer	6	[0 , 10]
min_child_weight	Float	1	MinValue : 0、 MaxValue : 120
子範例	Float	1	MinValue : 0.5、 MaxValue : 1
Gamma	Float	0	MinValue : 0、 MaxValue : 5
alpha	Float	0	MinValue : 0 , MaxValue : 1000
eta	Float	0.3	MinValue : 0.1、 MaxValue : 0.5
colsample_bylevel	Float	1	MinValue : 0.1 , MaxValue : 1
colsample_bynode	Float	1	MinValue : 0.1 , MaxValue : 1
colsample_bytree	Float	1	MinValue : 0.5、 MaxValue : 1
lambda	Float	1	MinValue : 0 , MaxValue : 1000
max_delta_step	Integer	0	[0, 10]

SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket', | TAGS 'string', | KMSKEY_ID 'kms_string', | S3_GARBAGE_COLLECT on / off, | MAX_CELLS 整數, | MAX_RUNTIME (, ...), | HORIZON整數, | FREQUENCY forecast_frequency, | 字串PERCENTILES陣列)

S3_BUCKET 子句指定用於儲存中繼結果的 Amazon S3 位置。

(選用) TAGS 參數是以逗號分隔的鍵值對清單，可用來標記在 Amazon SageMaker；和 Amazon Forecast 中建立的資源。標籤可協助您整理資源和配置成本。配對中的值是可選的，因此您可以使用 key=value 格式或直接建立索引鍵來建立標籤。如需 Amazon Redshift 中標籤的相關資訊，請參閱[標記概觀](#)。

(選用) KMS_KEY_ID 指定 Amazon Redshift 是否使用伺服器端加密搭配 AWS KMS 金鑰來保護靜態資料。傳輸中的資料會受到 Secure Sockets Layer () 的保護SSL。

(選用) S3_GARBAGE_COLLECT { ON | OFF } 指定 Amazon Redshift 是否對用於訓練模型和模型的結果資料集執行垃圾收集。如果設定為 OFF，則用於訓練模型和模型的資料集會保留在 Amazon S3 中，並可用於其他用途。如果設定為 ON，Amazon Redshift 會在訓練完成後刪除 Amazon S3 中的成品。預設值為 ON。

(選用) MAX_CELLS 指定訓練資料中的儲存格數目。此值是記錄數目 (訓練查詢或資料表中) 乘以欄數的乘積。預設值為 1,000,000。

(選用) MAX_RUNTIME 指定訓練的時間上限。視資料集大小而定，訓練工作通常會更快完成。這會指定訓練應耗用的時間上限。預設值為 5,400 (90 分鐘)。

HORIZON 指定預測模型可以傳回的預測數目上限。模型一經過訓練，就無法再變更此整數。如果訓練預測模型，則需要此參數。

FREQUENCY 指定您希望預測的精細時間單位。可用選項為 Y | M | W | D | H | 30min | 15min | 10min | 5min | 1min。如果訓練預測模型，則需要此參數。

(選用) PERCENTILES 是以逗號分隔的字串，指定用於訓練預測器的預測類型。預測類型可以是 0.01 到 0.99 之間的分位數，增量單位為 0.01 或更高。您也可以使用平均值指定平均預測。您最多可以指定五種預測類型。

MAX_BATCH_ROWS 整數

(選用) Amazon Redshift 在單一批次請求中針對單一 SageMaker 調用傳送的資料列數目上限。僅支援BYOM具有遠端推論的。此參數的最小值為 1。最大值為 INT_MAX、或 2, 147, 483, 647。只有在輸入和傳回的資料類型皆為時，才需要此參數SUPER。預設值為 INT_MAX、或 2, 147, 483, 647。

使用須知

使用 CREATE 時MODEL，請考慮下列事項：

- CREATE MODEL 陳述式會以非同步模式運作，並在訓練資料匯出至 Amazon S3 時傳回。在 Amazon 中進行訓練的剩餘步驟 SageMaker 會在背景中進行。訓練正在進行中時，對應的推論函數會顯示，但無法執行。您可以查詢 [STV_毫升_MODEL INFO](#) 以查看訓練狀態。
- 根據預設，在自動模式中，訓練最多可在背景執行 90 分鐘，而且可以延長。若要取消訓練，只要執行 [DROP MODEL](#) 命令即可。
- 您用來建立模型的 Amazon Redshift 叢集，以及用於暫存訓練資料和模型成品的 Amazon S3 儲存貯體，必須位於相同 AWS 區域。
- 在模型訓練期間，Amazon Redshift 會將中繼成品 SageMaker 存放在您提供的 Amazon S3 儲存貯體中。根據預設，Amazon Redshift 會在CREATEMODEL操作結束時執行垃圾收集。Amazon Redshift 會從 Amazon S3 中移除這些物件。若要在 Amazon S3 上保留這些成品，請設定 S3_GARBAGE COLLECTOFF選項。
- 您必須在子FROM句中提供的訓練資料中使用至少 500 列。
- 使用 CREATEMODEL陳述式時，您最多只能指定 FROM { table_name | (select_query) } 子句中的 256 個功能（輸入）資料欄。
- 對於 AUTO ON，您可以用作訓練集的欄類型為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE、BOOLEANCHAR、VARCHAR、DATE、TIMETZTIMESTAMP和 TIMESTAMPTZ。對於 AUTO OFF，您可以用作訓練集的欄類型為 SMALLINT、INTEGER、BIGINT、DECIMAL、REAL、DOUBLE和 BOOLEAN。
- 您不能使用 DECIMAL、DATE、TIME、TIMETZ、TIMESTAMP、TIMESTAMPTZ、GEOMETRY、GEOGRAPHY、H 或 VARBYTE作為目標欄類型。
- 若要改善模型精確度，請執行下列其中一項操作：
 - 當您在FROM子句中指定訓練資料時，請盡可能在CREATEMODEL命令中新增更多相關資料欄。
 - 針對 MAX_RUNTIME 和 MAX_ 使用較大的值CELLS。此參數的值越大，訓練模型的成本也會增加。
- 一旦計算訓練資料並匯出至 Amazon S3 儲存貯體，CREATEMODEL陳述式執行就會傳回。在此之後，您可以使用 SHOWMODEL命令檢查訓練的狀態。當在背景中訓練的模型失敗時，您可以使用 SHOW 檢查錯誤MODEL。您無法重試失敗的模型。使用 DROPMODEL移除失敗的模型，並重新建立新的模型。如需 SHOW 的詳細資訊MODEL，請參閱 [SHOW MODEL](#)。
- Local BYOM支援與 Amazon Redshift ML 支援的非BYOM案例相同的模型類型。Amazon Redshift 支援純 XGBoost (使用 1XGBoost.0 版或更新版本)、不含前置處理器的KMEANS模型，以及

XGBOOST Amazon SageMaker Autopilot 訓練的 /MLP/Linear Learner 模型。它使用 Autopilot 指定的 Amazon SageMaker Neo 也支援的前置處理器支援後者。

- 如果您的 Amazon Redshift 叢集已針對虛擬私有雲端 (VPC) 啟用增強路由，請務必為 VPC 叢集所在的 SageMaker VPC 建立 Amazon S3 VPC 端點和端點。這樣做可讓流量在 期間在這些服務 VPC 之間執行 CREATE 您的 MODEL。如需詳細資訊，請參閱 [SageMaker 釐清任務 Amazon VPC 子網路和安全群組](#)。

使用案例

下列使用案例示範如何使用 CREATE MODEL 來滿足您的需求。

簡單 CREATE MODEL

下列摘要說明 CREATE MODEL 語法的基本選項。

簡單 CREATE MODEL 語法

```
CREATE MODEL model_name
FROM { table_name | ( select_query ) }
TARGET column_name
FUNCTION prediction_function_name
IAM_ROLE { default }
SETTINGS (
    S3_BUCKET 'amzn-s3-demo-bucket',
    [ MAX_CELLS integer ]
)
```

簡單 CREATE MODEL 參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FROM { *table_name* | (*select_query*) }

指定訓練資料的 *table_name* 或查詢。它們可以是系統中現有的資料表，也可以是括號隨附的 Amazon Redshift 相容 SELECT 查詢，也就是 ()。查詢結果中至少必須有兩個資料列。

TARGET *column_name*

成為預測目標的資料欄名稱。資料欄必須存在於 FROM 子句中。

FUNCTION prediction_function_name

指定 要由 產生的 Amazon Redshift 機器學習函數名稱，CREATEMODEL並用來使用此模型進行預測的值。該函數會在與模型物件相同的結構描述中建立，並且可以多載。

Amazon Redshift 機器學習支援模型，例如用於迴歸和分類的 Xtreme 梯度增強樹 (XGBoost) 模型。

IAM_ROLE { 預設 | 'arn : aws : iam : : <account-id> : role/<role-name>' }

使用預設關鍵字讓 Amazon Redshift 在CREATMODEL命令執行時，使用設定為預設並與叢集相關聯的IAM角色。或者，您可以指定IAM角色ARN的 來使用該角色。

S3_BUCKET 'amzn-s3-demo-bucket'

您先前建立的 Amazon S3 儲存貯體名稱，用於在 Amazon Redshift 和 之間共用訓練資料和成品 SageMaker。Amazon Redshift 會在卸載訓練資料之前，在此儲存貯體中建立一個子資料夾。訓練完成後，Amazon Redshift 會刪除建立的子資料夾及其內容。

MAX_CELLS 整數

從FROM子句匯出的儲存格數目上限。預設值為 1,000,000。

儲存格數目是訓練資料 (由FROM子句資料表或查詢產生) 中資料列數目乘以資料欄數目的乘積。如果訓練資料中的儲存格數量超過 max_cells 參數指定的數量，會CREATMODEL減少子FROM句訓練資料，以減少低於 MAX_ 的訓練集大小CELLS。允許更大的訓練資料集可以產生更高的準確性，但也可能意味著模型需要更長的訓練時間和更高的成本。

如需 Amazon Redshift 使用成本的資訊，請參閱 [使用 Amazon Redshift ML 的成本](#)。

如需各種儲存格數目的成本和免費試用的相關資訊，請參閱 [Amazon Redshift 定價](#)。

CREATE MODEL 使用使用者指南

接下來，除了 中所述的選項CREATMODEL之外，您還可以找到 的選項描述[簡單 CREATE MODEL](#)。

根據預設，CREATE MODEL 會搜尋特定資料集的預先處理和模型的最佳組合。您可能需要模型的額外控制項目或對其引入其他領域知識 (例如問題類型或目標)。在客戶流失案例中，如果「客戶不活躍」的結果很少見，則 F1 目標通常高於準確度目標。因為高準確度模型可能會一直預測到「客戶很活躍」，此結果雖然具有高準確度，但商業價值很小。如需 F1 目標的相關資訊，請參閱 Amazon SageMaker API 參考 中的[utoMLJob目標](#)。

然後，會CREATE MODEL遵循您在指定方面的建議，例如目標。同時，CREATE MODEL會自動探索最佳前置處理器和最佳超參數。

CREATE MODEL 使用使用者指南語法

CREATE MODEL 在您可以指定的各個方面以及 Amazon Redshift 自動探索的各個方面提供更多彈性。

```
CREATE MODEL model_name
FROM { table_name | ( select_statement ) }
TARGET column_name
FUNCTION function_name
IAM_ROLE { default }
[ MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER } ]
[ PROBLEM_TYPE ( REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION ) ]
[ OBJECTIVE ( 'MSE' | 'Accuracy' | 'F1' | 'F1Macro' | 'AUC' ) ]
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket', |
  S3_GARBAGE_COLLECT { ON | OFF }, |
  KMS_KEY_ID 'kms_key_id', |
  MAX_CELLS integer, |
  MAX_RUNTIME integer (, ...)
)
```

CREATE MODEL 使用使用者指南參數

MODEL_TYPE { XGBOOST | MLP | LINEAR_LEARNER }

(選擇性) 指定模型類型。您可以指定是否要訓練特定模型類型的模型，例如 XGBoost、多層 Perceptron (MLP) 或 Linear Learner，這些都是 Amazon SageMaker Autopilot 支援的演算法。如果未指定參數，則會在訓練期間搜尋所有支援的模型類型，以取得最佳模型。

PROBLEM_TYPE (REGRESSION | BINARY_CLASSIFICATION | MULTICLASS_CLASSIFICATION)

(選擇性) 指定問題類型。如果您知道問題類型，則可以將 Amazon Redshift 限制為僅搜尋該特定模型類型的最佳模型。如果未指定此參數，則訓練期間會根據您的資料探索問題類型。

OBJECTIVE ('MSE' | 'Accuracy' | 'F1' | 'F1Macro' | 'AUC')

(選擇性) 指定用來測量機器學習系統預測品質的目標指標名稱。此指標會在訓練期間進行最佳化，從資料中提供模型參數值的最佳估計值。如果您未明確指定指標，預設行為是自動將 MSE：用於

迴歸、F1：用於二進位分類、準確性：用於多類別分類。如需目標的詳細資訊，請參閱 Amazon SageMaker API 參考 中的 [utoMLJob](#) 目標。

MAX_CELLS 整數

(選擇性) 指定訓練資料中的儲存格數目。此值是記錄數目 (訓練查詢或資料表中) 乘以欄數的乘積。預設值為 1,000,000。

MAX_RUNTIME 整數

(選擇性) 指定訓練的時間上限。視資料集大小而定，訓練工作通常會更快完成。這會指定訓練應耗用的時間上限。預設值為 5,400 (90 分鐘)。

S3_GARBAGE_COLLECT { ON | OFF }

(選擇性) 指定 Amazon Redshift 是否對用於訓練模型的結果資料集和模型執行垃圾回收。如果設定為 OFF，則用於訓練模型和模型的資料集會保留在 Amazon S3 中，並可用於其他用途。如果設定為 ON，Amazon Redshift 會在訓練完成後刪除 Amazon S3 中的成品。預設值為 ON。

KMS_KEY_ID 'kms_key_id'

(選擇性) 指定 Amazon Redshift 是否使用伺服器端加密搭配 AWS KMS 索引鍵來保護靜態資料。傳輸中的資料會受到 Secure Sockets Layer () 的保護 SSL。

PREPROCESSORS '字串'

(選擇性) 指定特定資料欄集的特定預處理器組合。格式是 的清單 columnSets，且要套用至每組資料欄的適當轉換。Amazon Redshift 會將特定轉換器清單中的所有轉換器套用至對應 中的所有資料欄 ColumnSet。例如，若要 OneHotEncoder 將 Imputer 套用至資料欄 t1 和 t2，請使用下列範例命令。

```
CREATE MODEL customer_churn
FROM customer_data
TARGET 'Churn'
FUNCTION predict_churn
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
PROBLEM_TYPE BINARY_CLASSIFICATION
OBJECTIVE 'F1'
PREPROCESSORS '[
...
{"ColumnSet": [
    "t1",
    "t2"
]},
"Transformers": [
```

```
        "OneHotEncoder",
        "Imputer"
    ]
},
{"ColumnSet": [
    "t3"
],
"Transformers": [
    "OneHotEncoder"
]
},
{"ColumnSet": [
    "temp"
],
"Transformers": [
    "Imputer",
    "NumericPassthrough"
]
}
]'
SETTINGS (
S3_BUCKET 'amzn-s3-demo-bucket'
)
```

Amazon Redshift 支援以下轉換：

- OneHotEncoder – 通常用於將離散值編碼為具有一個非零值的二進位向量。此轉換適用於許多機器學習模型。
- OrdinalEncoder – 將離散值編碼為單一整數。此轉換器適用於特定機器學習模型，例如 MLP 和 Linear Learner。
- NumericPassthrough – 將輸入如原樣傳遞至模型中。
- Imputer – 填入遺漏值，而不是數字 (NaN) 值。
- ImputerWithIndicator – 填入缺少的值和 NaN 值。此轉換也會建立指出是否有任何值遺失和填入的指示器。
- Normalizer – 將值標準化，可改善許多機器學習演算法的效能。
- DateTimeVectorizer – 建立向量內嵌，代表可用於機器學習模型的日期時間資料類型資料欄。
- PCA – 將資料投影到較低的維度空間，以減少功能數量，同時盡可能保留資訊。
- StandardScaler – 透過移除平均值和擴展至單位差異來標準化特徵。

- MinMax – 透過將每個功能擴展到指定範圍來轉換功能。

Amazon Redshift ML 會儲存經過訓練的轉換，並在預測查詢中自動套用這些轉換。從模型產生預測時，您不需要指定這些轉換。

CREATE XGBoost 具有的模型 AUTO OFF

與預設 AUTOOFFCREATEMODEL的目標大致不同CREATEMODEL。

身為進階使用者，其已知道您想要使用的模型類型，且超參數在訓練這些模型時要使用，您可以使用 CREATE MODEL 搭配 AUTOOFF來關閉前置處理器和超參數的CREATEMODEL自動探索。若要這樣做，您必須明確指定模型類型。XGBoost 當 AUTO 設為時，目前是唯一支援的模型類型OFF。您可以指定超參數。Amazon Redshift 會針對您指定的任何超參數使用預設值。

CREATE XGBoost 具有語法AUTOOFF的模型

```
CREATE MODEL model_name
FROM { table_name | (select_statement) }
TARGET column_name
FUNCTION function_name
IAM_ROLE { default }
AUTO OFF
MODEL_TYPE XGBOOST
OBJECTIVE { 'reg:squarederror' | 'reg:squaredlogerror' | 'reg:logistic' |
            'reg:pseudohubererror' | 'reg:tweedie' | 'binary:logistic' | 'binary:hinge'
            |
            'multi:softmax' | 'rank:pairwise' | 'rank:ndcg' }
HYPERPARAMETERS DEFAULT EXCEPT (
    NUM_ROUND '10',
    ETA '0.2',
    NUM_CLASS '10',
    (, ...)
)
PREPROCESSORS 'none'
SETTINGS (
    S3_BUCKET 'amzn-s3-demo-bucket', |
    S3_GARBAGE_COLLECT { ON | OFF }, |
    KMS_KEY_ID 'kms_key_id', |
    MAX_CELLS integer, |
    MAX_RUNTIME integer (, ...)
)
```

CREATE XGBoost 具有AUTOOFF參數的模型

AUTO OFF

關閉CREATEMODEL自動探索前置處理器、演算法和超參數選擇。

MODEL_TYPE XGBOOST

指定使用 XGBOOST 來訓練模型。

OBJECTIVE str

指定演算法辨識的目標。Amazon Redshift 可支援

reg:squarederror、reg:squaredlogerror、reg:logistic、reg:pseudohubererror、reg:tweedie、binary:logistic

如需這些目標的詳細資訊，請參閱 XGBoost 文件中的[學習任務參數](#)。

HYPERPARAMETERS { DEFAULT | DEFAULTEXCEPT (鍵「值」 (, ..)) }

指定預設XGBoost參數是否由使用者指定的值使用或覆寫。這些值必須用單引號括住。以下是XGBoost及其預設值的參數範例。

參數名稱	參數值	預設值	備註
num_class	Integer	多類 別分 類所 需。	N/A
num_round	Integer	100	N/A
tree_meth od	字串	Auto	N/A
max_depth	Integer	6	[0 , 10]
min_child _weight	Float	1	MinValue : 0、 MaxValue : 120
子範例	Float	1	MinValue : 0.5 , MaxValue : 1
Gamma	Float	0	MinValue : 0、 MaxValue : 5

參數名稱	參數值	預設值	備註
alpha	Float	0	MinValue : 0 , MaxValue : 1000
eta	Float	0.3	MinValue : 0.1、 MaxValue : 0.5
colsample _byleve	Float	1	MinValue : 0.1 , MaxValue : 1
colsample _bynode	Float	1	MinValue : 0.1 , MaxValue : 1
colsample _bytree	Float	1	MinValue : 0.5 , MaxValue : 1
lambda	Float	1	MinValue : 0 , MaxValue : 1000
max_delta _step	Integer	0	[0, 10]

下列範例會準備 的資料XGBoost。

```

DROP TABLE IF EXISTS abalone_xgb;

CREATE TABLE abalone_xgb (
length_val float,
diameter float,
height float,
whole_weight float,
shucked_weight float,
viscera_weight float,
shell_weight float,
rings int,
record_number int);

COPY abalone_xgb
FROM 's3://redshift-downloads/redshift-ml/abalone_xg/'
REGION 'us-east-1'
IAM_ROLE default

```

```
IGNOREHEADER 1 CSV;
```

下列範例會建立具有指定進階選項的XGBoost模型，例如 MODEL_OBJECTIVE、TYPE和PREPROCESSORS。

```
DROP MODEL abalone_xgboost_multi_predict_age;

CREATE MODEL abalone_xgboost_multi_predict_age
FROM ( SELECT length_val,
             diameter,
             height,
             whole_weight,
             shucked_weight,
             viscera_weight,
             shell_weight,
             rings
       FROM abalone_xgb WHERE record_number < 2500 )
TARGET rings FUNCTION ml_fn_abalone_xgboost_multi_predict_age
IAM_ROLE default
AUTO OFF
MODEL_TYPE XGBOOST
OBJECTIVE 'multi:softmax'
PREPROCESSORS 'none'
HYPERPARAMETERS DEFAULT EXCEPT (NUM_ROUND '100', NUM_CLASS '30')
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');
```

下列範例會使用推論查詢來對記錄編號大於 2500 的魚預測年齡。其使用從上述命令中建立的函數 ml_fn_abalone_xgboost_multi_predict_age。

```
select ml_fn_abalone_xgboost_multi_predict_age(length_val,
                                               diameter,
                                               height,
                                               whole_weight,
                                               shucked_weight,
                                               viscera_weight,
                                               shell_weight)+1.5 as age
from abalone_xgb where record_number > 2500;
```

自備模型 (BYOM) - 本機推論

Amazon Redshift ML 支援使用自有模型 (BYOM) 進行本機推論。

下列摘要說明 CREATE MODEL 語法的選項BYOM。您可以在 Amazon Redshift 之外與 Amazon 搭配使用訓練的模型 SageMaker ，在 Amazon Redshift 本機進行資料庫內推論。

CREATE MODEL 本機推論的語法

下列說明本機推論的CREATEMODEL語法。

```
CREATE MODEL model_name
FROM ('job_name' | 's3_path' )
FUNCTION function_name ( data_type [, ...] )
RETURNS data_type
IAM_ROLE { default }
[ SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket', | --required
  KMS_KEY_ID 'kms_string') --optional
];
```

Amazon Redshift 目前僅支援 的預先訓練 XGBoost、MLP和 Linear Learner 模型BYOM。您可以使用此路徑，匯入直接在 Amazon 中訓練的 SageMaker Autopilot 和模型，SageMaker 以進行本機推論。

CREATE MODEL 本機推論的參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FROM ('*job_name*' | '*s3_path*')

job_name 使用 Amazon SageMaker 任務名稱作為輸入。任務名稱可以是 Amazon SageMaker 訓練任務名稱或 Amazon SageMaker Autopilot 任務名稱。任務必須在擁有 Amazon Redshift 叢集的相同 AWS 帳戶中建立。若要尋找任務名稱，請啟動 Amazon SageMaker。在訓練下拉式選單中，選擇訓練工作。

'*s3_path*' 會指定 .tar.gz 模型成品檔案的 S3 位置，這會在建立模型時使用。

FUNCTION *function_name* (*data_type* 【 , ...】)

要建立的函數名稱和輸入引數的資料類型。您可以提供結構描述名稱。

RETURNS *data_type*

函數所傳回值的資料類型。

```
IAM_ROLE { 預設 | 'arn : aws : iam : : <account-id> : role/<role-name>' }
```

使用預設關鍵字讓 Amazon Redshift 在 CREATE MODEL 命令執行時，使用設定為預設並與叢集相關聯的 IAM 角色。

針對叢集用於身分驗證和授權 IAM 的角色，使用 Amazon Resource Name (ARN)。

```
SETTINGS ( S3_BUCKET 'amzn-s3-demo-bucket' , | KMS_KEY_ID 'kms_string' )
```

S3_BUCKET 子句指定用於儲存中繼結果的 Amazon S3 位置。

(選用) KMS_KEY_ID 子句指定 Amazon Redshift 是否使用伺服器端加密搭配 AWS KMS 金鑰來保護靜態資料。傳輸中的資料會受到 Secure Sockets Layer () 的保護 SSL。

如需詳細資訊，請參閱 [CREATE MODEL 使用使用者指南](#)。

CREATE MODEL 本機推論範例

下列範例會建立先前在 Amazon Redshift SageMaker 之外在 Amazon 中訓練的模型。由於 Amazon Redshift ML 支援本機推論的模型類型，因此下列 會 CREATE MODEL 建立可在 Amazon Redshift 本機使用的函數。您可以提供 SageMaker 訓練任務名稱。

```
CREATE MODEL customer_churn
FROM 'training-job-customer-churn-v4'
FUNCTION customer_churn_predict (varchar, int, float, float)
RETURNS int
IAM_ROLE default
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');
```

建立模型之後，您可以使用函數 customer_churn_predict 與指定的引數類型來進行預測。

使用您自己的模型 (BYOM) - 遠端推論

Amazon Redshift ML 也支援使用自有模型 (BYOM) 進行遠端推論。

下列摘要說明 CREATE MODEL 語法的選項 BYOM。

CREATE MODEL 遠端推論的語法

下列說明遠端推論的 CREATE MODEL 語法。

```
CREATE MODEL model_name
```

```

FUNCTION function_name ( data_type [, ...] )
RETURNS data_type
SAGEMAKER 'endpoint_name'[:'model_name']
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
[SETTINGS (MAX_BATCH_ROWS integer)];

```

CREATE MODEL 遠端推論的參數

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

FUNCTION fn_name (**【data_type】** **【, ...】**)

函數的名稱和輸入引數的資料類型。請參閱所有支援資料類型的[資料類型](#)。[Geography](#)、[geometry](#) 和 [hllsketch](#) 不受支援。

您也可以使用兩段式表示法在結構描述內提供函數名稱，例如 `myschema.myfunction`。

RETURNS data_type

函數所傳回值的資料類型。請參閱所有支援資料類型的[資料類型](#)。[Geography](#)、[geometry](#) 和 [hllsketch](#) 不受支援。

SAGEMAKER '*endpoint_name*' **【 : '*model_name*'】**

Amazon SageMaker 端點的名稱。如果端點名稱指向多模型端點，請新增要使用的模型名稱。端點必須託管在與 Amazon Redshift 叢集相同的 AWS 區域中。若要尋找您的端點，請啟動 Amazon SageMaker。在推論下拉式功能表中，選擇端點。

IAM_ROLE { 預設 | 'arn:aws:iam::<account-id>:role/<role-name>' }

使用預設關鍵字讓 Amazon Redshift 在 CREATE MODEL 命令執行時，使用設定為預設並與叢集相關聯的 IAM 角色。或者，您可以指定 IAM 角色 ARN 的來使用該角色。

MAX_BATCH_ROWS 整數

Amazon Redshift 在單一批次請求中針對單一 SageMaker 調用傳送的資料列數目上限。僅支援 BYOM 具有遠端推論的。批次中的實際資料列數目也取決於輸入大小，但小於或等於此值。此參數的最小值為 1。最大值為 INT_MAX、或 2, 147, 483, 647。只有在輸入和傳回的資料類型皆為時，才需要此參數 SUPER。預設值為 INT_MAX、或 2, 147, 483, 647。

當模型部署到 SageMaker 端點時，會在 Amazon Redshift 中 SageMaker 建立模型的資訊。然後透過外部函數執行推斷。您可以使用 SHOW MODEL 命令來檢視 Amazon Redshift 叢集上的模型資訊。

CREATE MODEL 適用於遠端推論用量備註

使用 CREATEMODEL進行遠端推論之前，請考慮下列事項：

- 端點必須由擁有 Amazon Redshift 叢集的相同 AWS 帳戶託管。
- 請確定 Amazon SageMaker 端點有足夠的資源來容納來自 Amazon Redshift 的推論呼叫，或者 Amazon SageMaker 端點可以自動擴展。
- 如果您不是使用SUPER資料類型做為輸入，則模型只會接受逗號分隔值（CSV）格式的輸入，對應 text/CSV中的內容類型 SageMaker。
- 如果您不是使用SUPER資料類型做為輸入，則模型的輸出是建立函數時所指定類型的單一值。輸出的格式為逗號分隔值（CSV），透過 text/CSV中的內容類型 SageMaker。VARCHAR資料類型不能為引號，且不能包含新行，且每個輸出都必須在新行中。
- 模型接受 null 作為空字串。
- 當輸入資料類型為 時SUPER，僅支援一個輸入引數。
- 當輸入資料類型為 時SUPER，傳回的資料類型也必須為 SUPER。
- MAX當輸入和傳回的資料類型皆為 時，需要 _BATCH_ROWSSUPER。
- 當輸入資料類型為端點調用的內容類型application/json時SUPER，則 MAX_BATCH_ROWS 為 1或application/jsonlines所有其他情況下為。
- 當傳回的資料類型是端點調用的接受類型application/json時SUPER，即 MAX_BATCH_ROWS 是 1或application/jsonlines所有其他情況下。

CREATE MODEL 適用於遠端推論範例

下列範例會建立使用 SageMaker 端點進行預測的模型。確定端點正在執行以進行預測，並在 CREATEMODEL命令中指定其名稱。

```
CREATE MODEL remote_customer_churn
FUNCTION remote_fn_customer_churn_predict (varchar, int, float, float)
RETURNS int
SAGEMAKER 'customer-churn-endpoint'
IAM_ROLE default;
```

下列範例會使用大型語言模型模型（）建立具有BYOM遠端推論的 LLM。LLMs 託管在 Amazon SageMaker Jumpstart 上接受並傳回application/json內容類型，且JSON每次調用都支援單一。輸入和傳回的資料類型必須為 SUPER，且 MAX_BATCH_ROWS 必須設定為 1。

```
CREATE MODEL sample_super_data_model
FUNCTION sample_super_data_model_predict(super)
RETURNS super
SAGEMAKER 'sample_super_data_model_endpoint'
IAM_ROLE default
SETTINGS (MAX_BATCH_ROWS 1);
```

CREATE MODEL 使用 K-MEANS

Amazon Redshift 支援可將未標示資料分組的 K 平均值演算法。此演算法可解決您想在資料中探索分組的叢集問題。未分類的資料會根據其相似性和差異進行分組和分割。

CREATE MODEL 使用 K MEANS語法

```
CREATE MODEL model_name
FROM { table_name | ( select_statement ) }
FUNCTION function_name
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS 'string'
HYPERPARAMETERS DEFAULT EXCEPT ( K 'val' [, ...] )
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  KMS_KEY_ID 'kms_string', |
  -- optional
  S3_GARBAGE_COLLECT on / off, |
  -- optional
  MAX_CELLS integer, |
  -- optional
  MAX_RUNTIME integer
  -- optional);
```

CREATE MODEL 使用 K 參數MEANS

AUTO OFF

關閉CREATEMODEL自動探索前置處理器、演算法和超參數選擇。

MODEL_TYPE KMEANS

指定使用 KMEANS 來訓練模型。

PREPROCESSORS '字串'

指定特定資料欄集的特定預處理器組合。格式是 的清單columnSets，且要套用至每組資料欄的適當轉換。Amazon Redshift 支援 3 個 K 平均值前置處理器，即 StandardScaler MinMax、和 NumericPassthrough。如果您不想套用任何 K 平均值的預處理，請 NumericPassthrough 明確選擇作為轉換器。如需轉換器的相關資訊，請參閱 [CREATE MODEL 使用使用者指南參數](#)。

K 平均值演算法會使用歐幾里德距離來計算相似性。預處理資料可確保模型的特徵保持相同的比例並產生可靠的結果。

HYPERPARAMETERS DEFAULT EXCEPT (K 'val' [, ...])

指定是否使用 K 平均值參數。使用 K 平均值演算法時，您必須指定 K 參數。如需詳細資訊，請參閱 Amazon SageMaker 開發人員指南中的 [K-Means 超參數](#)

下列範例會準備 K 平均值的資料。

```
CREATE MODEL customers_clusters
FROM customers
FUNCTION customers_cluster
IAM_ROLE default
AUTO OFF
MODEL_TYPE KMEANS
PREPROCESSORS '[
{
  "ColumnSet": [ "*" ],
  "Transformers": [ "NumericPassthrough" ]
}
]'
HYPERPARAMETERS DEFAULT EXCEPT ( K '5' )
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket');

select customer_id, customers_cluster(...) from customers;
customer_id | customers_cluster
-----
12345          1
12346          2
12347          4
12348
```


CREATE MODEL 使用預測

Redshift ML 中的預測模型會使用 Amazon Forecast 來建立準確的時間序列預測。這樣做可讓您使用一段時間內的歷史資料來預測未來事件。Amazon Forecast 的常見使用案例包括使用零售產品資料來決定如何為庫存定價格、製造數量資料來預測要訂購多少項目，以及使用 Web 流量資料來預測 Web 伺服器可能接收的流量。

[Amazon Forecast 的配額限制](#)會在 Amazon Redshift 預測模型中強制執行。例如，預測的最大數量為 100，但可調整。捨棄預測模型不會自動刪除 Amazon Forecast 中的相關聯資源。如果您刪除 Redshift 叢集，所有相關聯的模型也會被刪除。

請注意，預測模型目前僅適用於下列區域：

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太區域 (孟買) (ap-south-1)
- 亞太區域 (首爾) (ap-northeast-2)
- 亞太區域 (新加坡) (ap-southeast-1)
- 亞太區域 (雪梨) (ap-southeast-2)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (法蘭克福) (eu-central-1)
- 歐洲 (愛爾蘭) (eu-west-1)

CREATE MODEL 使用預測語法

```
CREATE [ OR REPLACE ] MODEL forecast_model_name
FROM { table_name | ( select_query ) }
TARGET column_name
IAM_ROLE { default | 'arn:aws:iam::<account-id>:role/<role-name>' }
AUTO ON
MODEL_TYPE FORECAST
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket',
  HORIZON integer,
  FREQUENCY forecast_frequency
  [PERCENTILES '0.1', '0.5', '0.9']
```

```
)
```

CREATE MODEL 使用預測參數

`forecast_model_name`

模型的名稱。模型名稱必須是唯一的。

```
FROM { table_name | ( select_query ) }
```

指定訓練資料的 `table_name` 或查詢。這可以是系統中現有的資料表，也可以是括號隨附的 Amazon Redshift 相容 SELECT 查詢。資料表或查詢結果至少必須有三個資料欄：(1) 指定時間序列名稱的 `varchar` 資料欄。每個資料集可以有多個時間序列；(2) 一個日期時間資料欄；以及 (3) 要預測的目標資料欄。此目標資料欄必須是整數或浮點數。如果您提供的資料集包含三個以上的資料欄，Amazon Redshift 會假設所有其他資料欄都是相關時間序列的一部分。請注意，相關的時間序列必須是整數或浮點數類型。如需相關時間序列的相關資訊，請參閱 [使用相關時間序列資料集](#)。

`TARGET column_name`

成為預測目標的資料欄名稱。資料欄必須存在於 FROM 子句中。

```
IAM_ROLE { 預設 | 'arn : aws : iam : : <account-id> : role/<role-name>' }
```

使用預設關鍵字讓 Amazon Redshift 在 CREATE MODEL 命令執行時，使用設定為預設並與叢集相關的 IAM 角色。或者，您可以指定 IAM 角色 ARN 的來使用該角色。

`AUTO` 開啟

開啟演算法和超參數選擇的 CREATE MODEL 自動探索。在建立預測模型時指定 `表示` 使用預測 AutoPredictor，其中 Amazon Forecast 會將演算法的最佳組合套用至資料集中的每個時間序列。

`MODEL_TYPE FORECAST`

指定使用 FORECAST 來訓練模型。

```
S3_BUCKET 'amzn-s3-demo-bucket'
```

您之前建立的 Amazon Simple Storage Service 儲存貯體名稱，用於在 Amazon Redshift 和 Amazon Forecast 之間共用訓練資料和成品。Amazon Redshift 會在卸載訓練資料之前，在此儲存貯體中建立一個子資料夾。訓練完成後，Amazon Redshift 會刪除建立的子資料夾及其內容。

`HORIZON` 整數

預測模型可傳回的預測數量上限。模型一經過訓練，就無法再變更此整數。

FREQUENCY forecast_frequency

指定您想要的預測細微程度。可用選項為 Y | M | W | D | H | 30min | 15min | 10min | 5min | 1min。如果您正在訓練預測模型，則需要此選項。

PERCENTILES 字串

逗號分隔的字串，可指定用來訓練預測器的預測類型。預測類型可以是 0.01 到 0.99 之間的分位數，增量單位為 0.01 或更高。您也可以使用平均值指定平均預測。您最多可以指定五種預測類型。

下列範例說明如何建立簡單預測模型。

```
CREATE MODEL forecast_example
FROM forecast_electricity_
TARGET target
IAM_ROLE 'arn:aws:iam::<account-id>:role/<role-name>'
AUTO ON
MODEL_TYPE FORECAST
SETTINGS (S3_BUCKET 'amzn-s3-demo-bucket',
          HORIZON 24,
          FREQUENCY 'H',
          PERCENTILES '0.25,0.50,0.75,mean',
          S3_GARBAGE_COLLECT OFF);
```

建立預測模型後，您可以使用預測資料建立新資料表。

```
CREATE TABLE forecast_model_results as SELECT Forecast(forecast_example)
```

然後，您可以查詢新資料表以取得預測。

```
SELECT * FROM forecast_model_results
```

CREATE PROCEDURE

為目前的資料庫建立新的預存程序或取代現有的程序。

如需詳細資訊和範例，請參閱 [在 Amazon Redshift 中建立預存程序](#)。

所需權限

您必須具有下列其中一種執行 CREATE OR REPLACE 的許可 PROCEDURE：

- 對於 CREATE PROCEDURE：
 - 超級使用者
 - 在建立預存程序的結構描述上具有 CREATE 和 USAGE 權限的使用者
- 對於 REPLACE PROCEDURE：
 - 超級使用者
 - 程序擁有者

語法

```
CREATE [ OR REPLACE ] PROCEDURE sp_procedure_name
  ( [ [ argname ] [ argmode ] argtype [, ...] ] )
[ NONATOMIC ]
AS $$
  procedure_body
$$ LANGUAGE plpgsql
[ { SECURITY INVOKER | SECURITY DEFINER } ]
[ SET configuration_parameter { TO value | = value } ]
```

參數

或 REPLACE

此子句指定如果已有一個程序的名稱和輸入引數資料類型 (或簽章) 與此程序相同，則取代現有程序。您只能將程序取代為定義一組相同資料類型的新程序。

如果您定義的程序與現有程序同名，但簽章不同，則會建立新程序。換言之，程序名稱是過載。如需詳細資訊，請參閱[多載程序名稱](#)。

sp_procedure_name

程序的名稱。如果您指定結構描述名稱 (例如 **myschema.myprocedure**)，則會在指定的結構描述中建立程序。否則會在目前結構描述中建立程序。如需有效名稱的相關資訊，請參閱[名稱與識別碼](#)。

我們建議您在所有預存程序名稱前加上 sp_。Amazon Redshift 會保留預存程序名稱的 sp_ 字首。您可以使用 sp_ 字首，以確保您的預存程序不會與任何現有或未來的 Amazon Redshift 內建預存程序或函數名稱發生衝突。如需詳細資訊，請參閱[命名預存程序](#)。

如果輸入引數的資料類型 (或簽章) 不同，您可以定義多個同名的程序。換言之，在此情況下程序名稱是過載。如需詳細資訊，請參閱[多載程序名稱](#)

[argname] [argmode] argtype

引數名稱、引數模式和資料類型的清單。僅資料類型是必要的。名稱和模式是選用的，其位置可調換。

引數模式可以是 IN、OUT 或 INOUT。預設值為 IN。

您可以使用 OUT 和 INOUT 引數，從程序呼叫傳回一或多個值。有 OUT 或 INOUT 引數時，程序呼叫會傳回一個包含 n 資料欄的結果列，其中 n 是 OUT 或 INOUT 引數的總數。

INOUT 引數是同時輸入和輸出引數。輸入引數包含 IN 和 INOUT 引數，而輸出引數包含 OUT 和 INOUT 引數。

OUT 引數不會指定為 CALL 陳述式的一部分。在預存程序 CALL 陳述式中指定 INOUT 引數。INOUT 從巢狀呼叫傳遞和傳回值，以及傳回時，引數非常有用 `refcursor`。如需 `refcursor` 類型的詳細資訊，請參閱 [游標](#)。

引數資料類型可以是任何標準 Amazon Redshift 資料類型。此外，引數資料類型還可以是 `refcursor`。

您最多可以指定 32 個輸入引數和 32 個輸出引數。

AS \$\$ procedure_body \$\$

包圍要執行之程序的結構。常值關鍵字 AS \$\$ 和 \$\$ 是必要的。

Amazon Redshift 會要求您使用稱為 \$ 符號引用的格式包圍程序中的陳述式。包圍範圍當中的任何內容都會原封不動傳遞。您不需要逸出任何特殊字元，因為字串的內容是逐字撰寫。

使用 \$ 符號引用時，您會使用一組 \$\$ 符號配對表示要執行之陳述式的開頭和結尾，如以下範例所示。

```
$$ my statement $$
```

您也可以選擇在每組配對的兩個 \$ 符號之間指定字串來協助識別陳述式。您在包圍配對的開頭和結尾中使用的字串必須相同。此字串區分大小寫，且遵循與未加引號的識別碼相同的限制條件，不過後者不可包含 \$ 符號。下列範例使用字串 `test`。

```
$test$ my statement $test$
```

此語法也適用於巢狀 \$ 符號引用。如需美元報價的詳細資訊，請參閱 PostgreSQL 文件的 [Lexical Structure](#) 下的「美元報價字串常數」。

procedure_body

一組有效的 PL/pgSQL 陳述式。PL/pgSQL 陳述式使用程序建構增強SQL命令，包括迴圈和條件式表達式，以控制邏輯流程。大多數SQL命令都可以在程序內文中使用，包括資料修改語言（DML），例如 COPY、UNLOAD和 INSERT，以及資料定義語言（DDL），例如 CREATE TABLE。如需詳細資訊，請參閱[PL/pgSQL 語言參考](#)。

LANGUAGE plpgsql

語言值。指定 plpgsql。您必須具有語言的使用許可，才能使用 plpgsql。如需詳細資訊，請參閱[GRANT](#)。

NONATOMIC

在非原子交易模式中建立預存程序。NONATOMIC 模式會自動遞交程序中的陳述式。此外，當NONATOMIC程序內發生錯誤時，如果錯誤是由例外區塊處理，則不會重新擲回錯誤。如需詳細資訊，請參閱[管理交易](#)和[RAISE](#)。

將預存程序定義為 NONATOMIC 時，請考慮下列事項：

- 當您巢狀化預存程序呼叫時，必須以相同的交易模式建立所有程序。
- 在 NONATOMIC 模式下建立程序時，不支援 SECURITY DEFINER 選項和 SET configuration_parameter 選項。
- 處理隱含遞交時，任何已開啟的 (明確或隱含) 游標皆會自動關閉。因此，您必須先開啟明確的交易，才能開始游標迴圈，以確保迴圈的反覆運算SQL中的任何 不會隱含遞交。

SECURITY INVOKER | SECURITY DEFINER

指定 NONATOMIC 時不支援 SECURITY DEFINER 選項。

程序的安全模式決定程序在執行時間的存取權限。程序必須有存取基礎資料庫物件的許可。

對於 SECURITY INVOKER 模式，程序會使用呼叫程序之使用者的權限。使用者必須具有基礎資料庫物件的明確權限。預設值為 SECURITY INVOKER。

對於 SECURITY DEFINER 模式，程序會使用程序擁有者的權限。程序擁有者定義為在執行階段擁有該程序的使用者，而不一定是最初定義程序的使用者。呼叫程序的使用者必須有程序的執行權限，但不需要基礎物件的任何權限。

SET configuration_parameter { TO value | = value }

指定 NONATOMIC 時不支援這些選項。

SET 子句會在輸入程序時將指定的 `configuration_parameter` 設定為指定的值。當程序離開時，此子句會接著將 `configuration_parameter` 還原到先前的值。

使用須知

如果使用 SECURITYDEFINER 選項建立預存程序，則在預存程序中叫用 CURRENT_USER 函數時，Amazon Redshift 會傳回預存程序擁有者的使用者名稱。

範例

Note

如果在執行這些範例時，您遇到類似以下的錯誤：

```
ERROR: 42601: [Amazon](500310) unterminated dollar-quoted string at or near "$$
```

請參閱 [Amazon Redshift 中的預存程序概觀](#)。

下列範例建立具有兩個輸入參數的程序。

```
CREATE OR REPLACE PROCEDURE test_sp1(f1 int, f2 varchar(20))
AS $$
DECLARE
    min_val int;
BEGIN
    DROP TABLE IF EXISTS tmp_tbl;
    CREATE TEMP TABLE tmp_tbl(id int);
    INSERT INTO tmp_tbl values (f1),(10001),(10002);
    SELECT INTO min_val MIN(id) FROM tmp_tbl;
    RAISE INFO 'min_val = %, f2 = %', min_val, f2;
END;
$$ LANGUAGE plpgsql;
```

Note

當您撰寫預存程序時，我們建議您採用保護敏感值的最佳作法：

不要在預存程序邏輯中對任何敏感資訊進行硬式編碼。例如，請勿在儲存程序內文的 CREATEUSER 陳述式中指派使用者密碼。這會造成安全性風險，因為硬式編碼值可以記錄為

目錄資料表中的結構描述資料。應改為透過參數將敏感值 (例如密碼) 當做引數傳遞給預存程序。

如需預存程序的詳細資訊，請參閱 [CREATEPROCEDURE](#) 和在 [Amazon Redshift 中建立預存程序](#)。如需目錄資料表的相關資訊，請參閱 [系統目錄資料表](#)。

下列範例會建立具有一個 IN 參數、一個 OUT 參數和一個 INOUT 參數的程序。

```
CREATE OR REPLACE PROCEDURE test_sp2(f1 IN int, f2 INOUT varchar(256), out_var OUT
  varchar(256))
AS $$
DECLARE
  loop_var int;
BEGIN
  IF f1 is null OR f2 is null THEN
    RAISE EXCEPTION 'input cannot be null';
  END IF;
  DROP TABLE if exists my_etl;
  CREATE TEMP TABLE my_etl(a int, b varchar);
  FOR loop_var IN 1..f1 LOOP
    insert into my_etl values (loop_var, f2);
    f2 := f2 || '+' || f2;
  END LOOP;
  SELECT INTO out_var count(*) from my_etl;
END;
$$ LANGUAGE plpgsql;
```

CREATE RLS POLICY

建立新的資料列層級安全性政策，以提供資料庫物件的精細存取權。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以建立政策。

語法

```
CREATE RLS POLICY policy_name
[ WITH (column_name data_type [, ...]) [ [AS] relation_alias ] ]
USING ( using_predicate_exp )
```


參數

policy_name

政策的名稱。

WITH (column_name data_type [、 ...])

指定參考政策附加資料表資料欄的 column_name 和 data_type。

只有在RLS政策未參考附加政策之資料表的任何資料欄時，您才能省略WITH子句。

AS relation_alias

為要連接RLS政策的資料表指定選用別名。

USING (using_predicate_exp)

指定套用至查詢子WHERE句的篩選條件。Amazon Redshift 會在查詢層級使用者述詞之前套用政策述詞。例如，**current_user = 'joe' and price > 10** 會限制 Joe 只能查看價格大於 \$10 的記錄。

使用須知

使用 CREATE RLS POLICY 陳述式時，請遵循下列事項：

- Amazon Redshift 支援篩選條件，這些篩選條件可能是查詢子WHERE句的一部分。
- 所有附加至資料表的政策都必須使用相同的資料表別名建立。
- 您不需要查詢資料表的SELECT許可。當您建立政策時，Amazon Redshift 會授予個別政策查詢表的SELECT許可。查閱資料表是在政策定義中使用的資料表物件。
- Amazon Redshift 資料列層級安全性不支援政策定義中的下列物件類型：目錄資料表、跨資料庫關係、外部資料表、一般檢視、繫結延遲檢視、開啟RLS政策的資料表，以及暫存資料表。

範例

下列SQL陳述式會建立CREATERLSPOLICY範例的資料表、使用者和角色。

```
-- Create users and roles reference in the policy statements.  
CREATE ROLE analyst;  
  
CREATE ROLE consumer;
```

```
CREATE USER bob WITH PASSWORD 'Name_is_bob_1';

CREATE USER alice WITH PASSWORD 'Name_is_alice_1';

CREATE USER joe WITH PASSWORD 'Name_is_joe_1';

GRANT ROLE sys:secadmin TO bob;

GRANT ROLE analyst TO alice;

GRANT ROLE consumer TO joe;

GRANT ALL ON TABLE tickit_category_redshift TO PUBLIC;
```

以下範例會建立稱為 `policy_concerts` 的政策。

```
CREATE RLS POLICY policy_concerts
WITH (catgroup VARCHAR(10))
USING (catgroup = 'Concerts');
```

CREATE ROLE

建立作為權限集合的新自訂角色。如需 Amazon Redshift 系統定義角色的清單，請參閱 [the section called “Amazon Redshift 系統定義角色”](#)。查詢 [SVV_ROLES](#) 以檢視叢集或工作群組中目前建立的角色。

可以建立的角色數量有配額限制。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 中的配額和限制](#)。

所需的許可

以下是 CREATE 的必要權限ROLE。

- 超級使用者
- 具有 CREATEROLE 權限的使用者

語法

```
CREATE ROLE role_name
[ EXTERNALID external_id ]
```

參數

role_name

角色的名稱。角色名稱必須是唯一的，且不能與任何使用者名稱相同。角色名稱不能是保留字。

具有 CREATEROLE 權限的超級使用者或一般使用者可以建立角色。非超級使用者但已授予 USAGE 角色 WITHGRANTOPTION 和 ALTER 權限的使用者，可以將此角色授予任何人。

EXTERNALID external_id

與身分提供者相關聯的角色識別碼。如需詳細資訊，請參閱 [Amazon Redshift 的原生身分提供者 \(IdP\) 聯合](#)。

範例

下列範例會建立 sample_role1 角色。

```
CREATE ROLE sample_role1;
```

下列範例會使用與身分提供者相關聯的外部 ID 來建立角色 sample_role1。

```
CREATE ROLE sample_role1 EXTERNALID "ABC123";
```

CREATE SCHEMA

為目前資料庫定義新的結構描述。

所需權限

以下是 CREATE 的必要權限 SCHEMA：

- 超級使用者
- 具有 CREATESCHEMA 權限的使用者

語法

```
CREATE SCHEMA [ IF NOT EXISTS ] schema_name [ AUTHORIZATION username ]  
            [ QUOTA {quota [MB | GB | TB] | UNLIMITED} ] [ schema_element [ ... ]
```

```
CREATE SCHEMA AUTHORIZATION username [ QUOTA {quota [MB | GB | TB] | UNLIMITED} ]  
[ schema_element [ ... ] ]
```

參數

IF NOT EXISTS

此子句會指出，若指定的結構描述已存在，則命令不應進行任何變更，且應傳回結構描述存在的訊息，而不是在發生錯誤的情況下終止。

此子句在指令碼時很有用，因此如果CREATESCHEMA嘗試建立已存在的結構描述，指令碼不會失敗。

schema_name

新結構描述的名稱。結構描述名稱不可以是 PUBLIC。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

Note

[search_path](#) 組態參數中結構描述的清單會決定，在沒有結構描述名稱的情況下參考名稱相同的物件時，這些物件的優先順序。

AUTHORIZATION

將所有權提供給指定使用者的子句。

username

結構描述擁有者的名稱。

schema_element

要在結構描述內建立的一個或多個物件的定義。

QUOTA

所指定之結構描述可以使用的磁碟空間數上限。這個空間是集體磁碟使用量。它包含所有永久資料表、指定結構描述下的具體化檢視，以及在每個運算節點上ALL分佈的所有資料表複本。結構描述配額不會將暫時資料表納入考量，此暫時資料表是作為暫時命名空間或結構描述一部分而建立的。

若要檢視設定的結構描述配額，請參閱 [SVV_SCHEMA_QUOTA_STATE](#)。

若要檢視超出結構描述配額的記錄，請參閱 [STL_SCHEMA_QUOTA_VIOLATIONS](#)。

Amazon Redshift 會將選取的值轉換為 MB。GB 是您未指定值時的預設測量單位。

您必須是資料庫超級使用者，才能設定和變更結構描述配額。非超級使用者但具有 CREATE SCHEMA 許可的使用者可以建立具有定義配額的結構描述。當您在未定義配額的情況下建立結構描述時，結構描述會有無限的配額。當您將配額設為低於結構描述所使用的目前值時，除非您釋放磁碟空間，否則 Amazon Redshift 不允許進一步擷取。DELETE 陳述式會從資料表中刪除資料，且僅在 VACUUM 執行時釋放磁碟空間。

Amazon Redshift 會在確認交易之前檢查每筆交易是否存在配額違規情況。Amazon Redshift 會根據設定的配額，檢查每個已修改結構描述的大小 (結構描述中所有資料表使用的磁碟空間)。因為配額違規檢查發生在交易結束時，所以在確定之前，大小限制可以暫時超過交易內的配額。當交易超過配額時，Amazon Redshift 會停止交易、禁止後續導入，並還原所有變更，直到您釋放磁碟空間為止。由於背景 VACUUM 和內部清除，在您取消交易後檢查結構描述時，結構描述可能尚未完全完成。

在例外情況下，Amazon Redshift 會忽略配額違規，並在特定情況下遞交交易。Amazon Redshift 會針對僅包含下列一或多個陳述式的交易執行此操作，其中相同交易中沒有 INSERT 或 COPY 擷取陳述式：

- DELETE
- TRUNCATE
- VACUUM
- DROP TABLE
- ALTER TABLE APPEND 只有在將資料從完整結構描述移至其他非完整結構描述時

UNLIMITED

Amazon Redshift 不會限制結構描述總大小的增長。

限制

Amazon Redshift 會對結構描述強制實施下列限制。

- 每個資料庫最多 9900 個結構描述。

範例

下列範例會建立名為 US_SALES 的結構描述，並將所有權授予使用者 DWUSER。

```
create schema us_sales authorization dwuser;
```

下列範例會建立名為 US_ 的結構描述 SALES，將擁有權授予使用者 DWUSER，並將配額設定為 50 GB。

```
create schema us_sales authorization dwuser QUOTA 50 GB;
```

若要檢視新的結構描述，請查詢 PG_NAMESPACE 目錄表，如下所示。

```
select nspname as schema, username as owner
from pg_namespace, pg_user
where pg_namespace.nspowner = pg_user.usesysid
and pg_user.username = 'dwuser';
```

```
   schema | owner
-----+-----
 us_sales | dwuser
(1 row)
```

下列範例會建立 US_SALES 結構描述，或者不執行任何動作，如果訊息已經存在，則傳回訊息。

```
create schema if not exists us_sales;
```

CREATE TABLE

在目前資料庫中建立新的資料表。您可以定義資料欄清單，每個資料欄都會保留不同類型的資料。資料表的擁有者是 CREATE TABLE 命令的發行者。

所需權限

以下是 CREATE 的必要權限 TABLE：

- 超級使用者
- 具有 CREATETABLE 權限的使用者

語法

```
CREATE [ [LOCAL ] { TEMPORARY | TEMP } ] TABLE
[ IF NOT EXISTS ] table_name
```

```
( { column_name data_type [column_attributes] [ column_constraints ]
  | table_constraints
  | LIKE parent_table [ { INCLUDING | EXCLUDING } DEFAULTS ] }
[, ... ] )
[ BACKUP { YES | NO } ]
[table_attributes]

where column_attributes are:
  [ DEFAULT default_expr ]
  [ IDENTITY ( seed, step ) ]
  [ GENERATED BY DEFAULT AS IDENTITY ( seed, step ) ]
  [ ENCODE encoding ]
  [ DISTKEY ]
  [ SORTKEY ]
  [ COLLATE CASE_SENSITIVE | COLLATE CASE_INSENSITIVE ]

and column_constraints are:
  [ { NOT NULL | NULL } ]
  [ { UNIQUE | PRIMARY KEY } ]
  [ REFERENCES reftable [ ( refcolumn ) ] ]

and table_constraints are:
  [ UNIQUE ( column_name [, ... ] ) ]
  [ PRIMARY KEY ( column_name [, ... ] ) ]
  [ FOREIGN KEY ( column_name [, ... ] ) REFERENCES reftable [ ( refcolumn ) ]

and table_attributes are:
  [ DISTSTYLE { AUTO | EVEN | KEY | ALL } ]
  [ DISTKEY ( column_name ) ]
  [ [COMPOUND | INTERLEAVED ] SORTKEY ( column_name [,...]) | [ SORTKEY AUTO ] ]
  [ ENCODE AUTO ]
```

參數

LOCAL

選用。雖然陳述式中可接受此關鍵字，但是在 Amazon Redshift 中沒有作用。

TEMPORARY | TEMP

建立臨時資料表的關鍵字，只能在目前工作階段中看見。資料表會在建立所在的工作階段結束時自動捨棄。臨時資料表可與永久資料表同名。臨時資料表是以不同的工作階段專屬結構描述建立 (您

無法指定此結構描述的名稱)。此臨時結構描述會成為搜尋路徑中的第一個結構描述，因此，除非您以結構描述名稱限定資料表名稱來存取永久資料表，否則臨時資料表的優先順序高於永久資料表。如需結構描述和優先順序的相關資訊，請參閱 [search_path](#)。

Note

根據預設，資料庫使用者有權透過其在PUBLIC群組中的自動成員資格來建立暫存資料表。若要拒絕此權限給使用者，請從 PUBLIC 群組撤銷TEMP權限，然後明確地將TEMP權限僅授予特定使用者或使用者群組。

IF NOT EXISTS

此子句會指出，若指定的資料表已存在，則命令不應進行任何變更，且應傳回資料表存在的訊息，而不是在發生錯誤的情況下停止。請注意，現有資料表可能與這裡建立的資料表完全不一樣；只有資料表名稱進行比較。

此子句在指令碼時很有用，因此如果CREATETABLE嘗試建立已存在的資料表，指令碼不會失敗。

table_name

要建立的資料表名稱。

Important

若您指定 '#' 開頭的資料表名稱，所建立的資料表會是臨時資料表。以下是範例：

```
create table #newtable (id int);
```

您也可以使用 '#' 參考資料表。例如：

```
select * from #newtable;
```

資料表名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 個多位元組字元，最多四個位元組。Amazon Redshift 會依節點類型強制執行每個叢集的資料表數目配額限制，包括使用者定義的臨時資料表，以及 Amazon Redshift 在查詢處理或系統維護期間建立的臨時資料表。或者，資料表名稱也可透過資料庫和結構描述名稱來限定。在以下範例中，資料庫名稱為 tickit，結構描述名稱為 public，而資料表名稱為 test。


```
create table tickit.public.test (c1 int);
```

如果資料庫或結構描述不存在，則不會建立資料表，而且陳述式會傳回錯誤。您無法在系統資料庫 `template0`、`template1`、`padb_harvest` 或 `sys:internal` 中建立資料表或視觀表。

若提供結構描述名稱，則會在該結構描述中建立新資料表 (假設建立者具有存取結構描述的權限)。資料表名稱對於該結構描述來說必須是唯一的。如果未指定結構描述，則會使用目前資料庫結構描述建立資料表。如果您要建立臨時資料表，就不能指定結構描述名稱，因為臨時資料表會採用特殊結構描述。

若臨時資料表是在不同的工作階段中建立的話，在同一個資料庫中可同時有多個同名的臨時資料表存在，因為資料表會指派至不同的結構描述。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

column_name

要在新資料表中建立的資料欄名稱。資料欄名稱的長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。您可以使用 UTF-8 個多位元組字元，最多四個位元組。單一資料表中可定義的資料欄數目上限為 1,600 個。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

Note

若您要建立「寬資料表」，則務必注意，在載入和查詢處理期間，您的資料欄清單未超過中繼結果的資料列寬度界限。如需詳細資訊，請參閱 [使用須知](#)。

data_type

要建立之資料欄的資料類型。對於 CHAR 和 VARCHAR 資料欄，您可以使用 MAX 關鍵字，而不是宣告長度上限。MAX 會將的長度上限設定為 4,096 位元組 CHAR，或設定為 65535 位元組 VARCHAR。GEOMETRY 物件的大小上限為 1,048,447 個位元組。

如需有關 Amazon Redshift 支援資料類型的資訊，請參閱 [資料類型](#)。

DEFAULT default_expr

此子句會指派資料欄的預設資料值。default_expr 的資料類型必須符合資料欄的資料類型。DEFAULT 值必須是無變數表達式。不允許子查詢、目前資料表中其他資料欄的交叉參考，以及使用者定義的功能。

default_expr 運算式用於任何未指定資料欄值 INSERT 的操作。若未指定預設值，則資料欄的預設值為 null。

如果具有定義資料欄清單COPY的操作省略了具有DEFAULT值的資料欄，則COPY命令會插入 default_expr 的值。

IDENTITY (種子 , 步驟)

指定資料欄為IDENTITY資料欄的子句。資料IDENTITY欄包含唯一的自動產生的值。IDENTITY 資料欄的資料類型必須為 INT或 BIGINT。

當您使用 INSERT 或 INSERT INTO [tablename] VALUES() 陳述式新增資料列時，這些值會從指定為 seed 的值開始，並依指定為 step 的數字遞增。

使用 INSERT INTO [tablename] SELECT * FROM 或 COPY 陳述式載入資料表時，會並行載入資料並將其分發至節點片段。為確保身分值是唯一的，Amazon Redshift 會在建立身分值時略過一些值。身分值是唯一的，但順序可能不符合來源檔案中的順序。

GENERATED DEFAULT AS IDENTITY (種子 , 步驟)

指定資料欄為預設IDENTITY資料欄，並可讓您自動為資料欄指派唯一值的子句。IDENTITY 資料欄的資料類型必須為 INT或 BIGINT。當您新增沒有值的資料列時，這些值會從指定為 seed 的值開始，並依指定為 step 的數字遞增。如需如何產生值的詳細資訊，請參閱[IDENTITY](#)。

此外，在 INSERT、UPDATE或 期間COPY，您可以提供不含 EXPLICIT_ 的值IDS。Amazon Redshift 會使用該值插入身分資料欄，而不是使用系統產生的值。此值可以是複本、小於種子的值，或是介於步驟值之間的值。Amazon Redshift 不會檢查資料欄中的值是否是唯一的。提供一值並不會影響下一個系統產生的值。

Note

如果您需要資料欄中的唯一性，請不要新增複本值。改為新增小於種子或介於步驟值之間的唯一值。

請記住下列有關預設身分資料欄的事項：

- 預設身分資料欄為 NOT NULL。NULL 無法插入。
- 若要將產生的值插入至預設身分資料欄，請使用關鍵字 DEFAULT。

```
INSERT INTO tablename (identity-column-name) VALUES (DEFAULT);
```

- 置換預設身分資料欄的值並不會影響下一個產生的值。
- 您無法使用 ALTERTABLEADDCOLUMN陳述式新增預設身分資料欄。
- 您可以使用 ALTERTABLEAPPEND陳述式附加預設身分資料欄。

ENCODE 編碼

資料欄的壓縮編碼。ENCODE AUTO 是資料表的預設值。Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。如果您為資料表中的任何資料欄指定壓縮編碼，則資料表不再設定為 ENCODE AUTO。Amazon Redshift 不再自動管理資料表中所有資料欄的壓縮編碼。您可以指定資料表 ENCODE AUTO 的選項，讓 Amazon Redshift 自動管理資料表中所有資料欄的壓縮編碼。

Amazon Redshift 會自動將初始壓縮編碼指派給您未指定壓縮編碼的資料欄，如下所示：

- 根據預設，暫存資料表中的所有資料欄都會指派 RAW 壓縮。
- 定義為排序索引鍵的資料欄會被指派 RAW 壓縮。
- 定義為 BOOLEAN、REAL、DOUBLEPRECISION、GEOMETRY 或 GEOGRAPHY 資料類型的資料欄會被指派 RAW 壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMAL、DATE、TIME、TIMESTAMP、TIMETZ 或 的資料欄 TIMESTAMPTZ 會指派 AZ64 壓縮。
- 定義為 CHAR、VARCHAR 或 的資料欄 VARBYTE 會被指派 LZ0 壓縮。

Note

如果您不想壓縮資料欄，請明確指定 RAW 編碼。

支援以下 [compression encodings \(p. 54\)](#)：

- AZ64
- BYTEDICT
- DELTA
- DELTA32K
- LZ0
- MOSTLY8
- MOSTLY16
- MOSTLY32
- RAW (無壓縮)
- RUNLENGTH
- TEXT255
- TEXT32K

- ZSTD

DISTKEY

此關鍵字會指定資料欄是資料表的分佈索引鍵。資料表中只能有一個資料欄是分佈索引鍵。您可以使用 `DISTKEY (column_name)` 語法，在資料欄名稱後面或作為資料表定義的一部分使用 `DISTKEY` 關鍵字。兩種方法的效果一樣。如需詳細資訊，請參閱本主題稍後的 `DISTSTYLE` 參數。

分發金鑰欄的資料類型可以是：

`BOOLEAN`、`REAL`、`DOUBLEPRECISION`、`SMALLINTINTEGER`、`BIGINT`、`DECIMAL`、`DATE`、`TIMESTAMP` 或 `CHAR`、`TIMESTAMPTZ` 或 `VARCHAR`。

SORTKEY

此關鍵字會指定資料欄是資料表的排序索引鍵。當資料載入資料表時，資料會依指定為排序索引鍵的一個或多個資料欄排序。您可以在資料欄名稱後面使用 `SORTKEY` 關鍵字來指定單一資料欄排序索引鍵，或者您可以使用 `SORTKEY (column_name [, ...])` 語法，將一或多個資料欄指定為資料表的排序索引鍵資料欄。此語法只會建立複合排序索引鍵。

每個資料表最多可定義 400 個 `SORTKEY` 資料欄。

排序索引鍵資料欄的資料類型可以是：

`BOOLEAN`、`REAL`、`DOUBLEPRECISIONSMALLINT`、`INTEGER`、`BIGINT`、`DECIMAL`、`DATE`、`TIMESTAMP` 或 `CHAR`、`TIMESTAMPTZ` 或 `VARCHAR`。

COLLATE CASE_SENSITIVE | COLLATE CASE_INSENSITIVE

指定資料欄上的字串搜尋或比較是 `CASE_SENSITIVE` 還是 `CASE_INSENSITIVE` 的子句。預設值與資料庫目前的區分大小寫設定相同。

若要尋找資料庫定序資訊，請使用下列命令：

```
SELECT db_collation();

db_collation
-----
case_sensitive
(1 row)
```

NOT NULL | NULL

`NOT NULL` 指定不允許資料欄包含 `null` 值。`NULL` 是預設值，指定資料欄接受 `null` 值。`IDENTITY` 資料欄 `NOTNULL` 預設為宣告。

UNIQUE

此關鍵字會指定資料欄只能包含唯一值。唯一資料表限制條件的行為與資料欄限制條件的行為相同，但多了可橫跨多個資料欄的額外功能。若要定義唯一的資料表限制條件，請使用 `UNIQUE (column_name [, ...])` 語法。

Important

唯一限制條件僅供參考，系統不會強制執行它們。

PRIMARY KEY

此關鍵字會指定資料欄是資料表的主索引鍵。只能使用資料欄定義將一個資料欄定義為主索引鍵。若要使用多欄主索引鍵定義資料表限制，請使用 `PRIMARYKEY (column_name [, ...])` 語法。

將資料欄識別為主索引鍵即可提供有關結構描述設計的中繼資料。主索引鍵表示，其他資料表可倚賴這組資料欄做為資料列的唯一識別碼。一個資料表中可指定一個主索引鍵，無論是資料欄限制條件或資料表限制條件都可行。主索引鍵限制條件應命名一組資料欄，這組資料欄有別於由為相同資料表所定義的任何唯一限制條件命名的其他資料欄組。

`PRIMARY KEY` 欄位也定義為 `NOT NULL`。

Important

主索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

References reftable [(refcolumn)]

此子句會指定外部索引鍵限制條件，表示資料欄包含的值只能是符合參考資料表中某一資料列之參考資料欄的值。參考資料欄應為參考資料表中唯一或主索引鍵限制條件的資料欄。

Important

外部索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

LIKE parent_table 【 { INCLUDING | EXCLUDING } DEFAULTS 】

指定現有資料表的子句，新資料表會自動從中複製資料欄名稱、資料類型和NOTNULL限制條件。新資料表和父資料表是分開的，對父資料表所做的變更不會套用至新資料表。只有在指定 INCLUDING DEFAULTS 時，才會複製複製複製的資料欄定義的預設表達式。預設行為是執行預設表達式，如此一來，新資料表中所有資料欄的預設值都是 null。

使用 LIKE 選項建立的資料表不會繼承主要和外部金鑰限制。分佈樣式、排序索引鍵BACKUP、和 NULL 屬性會依LIKE資料表繼承，但您無法在 CREATE TABLE ... 中明確設定 LIKE 陳述式。

BACKUP { YES | NO }

此子句會指定資料表是否應包含在自動化和手動叢集快照中。

對於不包含關鍵資料的暫存資料表等資料表，請指定 BACKUP NO，以在建立快照和從快照還原時節省處理時間，並減少 Amazon Simple Storage Service 上的儲存空間。BACKUP NO 設定不會影響資料自動複寫至叢集內其他節點，因此在節點失敗的情況下，會還原具有指定 BACKUP NO 的資料表。預設值為 BACKUP YES。

DISTSTYLE { AUTO | EVEN | KEY | ALL }

定義整個資料表的資料分佈樣式的關鍵字。Amazon Redshift 會根據資料表上指定的分佈樣式，將資料表的資料列分散到運算節點。預設值為 AUTO。

您為資料表選取的分佈樣式會影響資料庫的整體效能。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。可能的分佈樣式如下：

- AUTO：Amazon Redshift 會根據資料表資料指派最佳分佈樣式。例如，如果指定AUTO分佈樣式，Amazon Redshift 最初會將ALL分佈樣式指派給小型資料表。當資料表變大時，Amazon Redshift 可能會將分佈樣式變更為 KEY，選擇主索引鍵（或複合主索引鍵的一欄）作為 DISTKEY。如果資料表變大，且任何資料欄都不適合成為 DISTKEY，Amazon Redshift 會將分佈樣式變更為 EVEN。分佈樣式的變更會發生在背景中，對使用者查詢的影響最小。

若要檢視套用至資料表的分佈樣式，請查詢 PG_CLASS 系統目錄資料表。如需詳細資訊，請參閱[檢視分佈樣式](#)。

- EVEN：資料表中的資料平均分散到循環配置中叢集中的節點。資料列IDs用於決定分佈，且大約相同的資料列數目會分佈到每個節點。
- KEY：資料由DISTKEY資料欄中的值分佈。當您將聯結資料表的聯結資料欄設定為分佈索引鍵時，兩個資料表的聯結資料列會在運算節點上並存。資料並存時，最佳化工具就能更有效率地執行聯結。如果您指定 DISTSTYLE KEY，則必須為資料表或作為資料DISTKEY欄定義的一部分命名資料欄。如需詳細資訊，請參閱本主題稍早的 DISTKEY 參數。

- ALL：整個資料表的副本會分發給每個節點。此分佈樣式可確保每個節點上都有任何聯結所需的所有資料列，但儲存空間的需求也會倍增，並且會增加資料表的負載和維護次數。ALL 與某些維度資料表搭配使用時，分佈可以改善執行時間，因為KEY分佈不適當，但效能改善必須權衡維護成本。

DISTKEY (column_name)

此限制條件會指定要做為資料表分佈索引鍵的資料欄。您可以使用 DISTKEY (column_name) 語法，在資料欄名稱之後或作為資料表定義的一部分使用DISTKEY關鍵字。兩種方法的效果一樣。如需詳細資訊，請參閱本主題稍早的 DISTSTYLE 參數。

【COMPOUND | INTERLEAVED】 SORTKEY (column_name [, ...]) | 【 SORTKEY AUTO 】

指定資料表的一個或多個排序索引鍵。當資料載入資料表時，資料會依指定為排序索引鍵的資料欄排序。您可以在資料欄名稱後面使用SORTKEY關鍵字來指定單一資料欄排序索引鍵，或使用SORTKEY (column_name [, ...])語法來指定一或多個資料欄作為資料表的排序索引鍵資料欄。

您可以選擇性地指定COMPOUND或INTERLEAVED排序樣式。如果您SORTKEY使用資料欄指定，則預設值為 COMPOUND。如需詳細資訊，請參閱[排序金鑰](#)。

如果您未指定任何排序金鑰選項，則預設為 AUTO。

您最多可以定義 400 個COMPOUNDSORTKEY資料欄或每個資料表 8 個 INTERLEAVEDSORTKEY資料欄。

AUTO

指定 Amazon Redshift 會根據資料表資料指派最佳排序索引鍵。例如，如果指定AUTO排序索引鍵，Amazon Redshift 最初不會將排序索引鍵指派給資料表。如果 Amazon Redshift 判斷排序索引鍵可以改善查詢效能，那麼 Amazon Redshift 可能會在變更資料表的排序索引鍵。資料表的實際排序是透過自動資料表排序來完成的。如需詳細資訊，請參閱[自動資料表排序](#)。

Amazon Redshift 不會修改已具有排序或分佈索引鍵的資料表。有一個例外是，如果資料表具有從未在 中使用的分發金鑰JOIN，則如果 Amazon Redshift 確定存在更好的金鑰，則金鑰可能會變更。

若要檢視資料表的排序索引鍵，請查詢 SVV_TABLE_INFO 系統目錄檢視。如需詳細資訊，請參閱[SVV_TABLE_INFO](#)。若要檢視資料表的 Amazon Redshift Advisor 建議，請查詢 SVV_ALTER_TABLE_RECOMMENDATIONS 系統目錄檢視。如需詳細資訊，請參閱[SVV_ALTER_TABLE_RECOMMENDATIONS](#)。若要檢視 Amazon Redshift 採取

的動作，請查詢 `SVL_AUTO_WORKER_ACTION` 系統目錄檢視。如需詳細資訊，請參閱 [SVL_AUTO_WORKER_ACTION](#)。

COMPOUND

指定使用複合索引鍵排序資料，該索引鍵是由列出的所有資料欄組成，並依其列出順序排列。當查詢依據排序資料欄的順序掃描資料列時，複合排序索引鍵最實用。當查詢依賴次要排序資料欄時，使用複合索引鍵排序的效能優勢就會降低。每個資料表最多可定義 400 個 `COMPOUNDSORTKEY` 資料欄。

INTERLEAVED

指定使用交錯排序索引鍵排序資料。最多可以為交錯排序索引鍵指定八個資料欄。

交錯排序索引鍵對排序索引鍵中的每個資料欄 (或資料欄子集) 都提供相等的權重，所以查詢不會取決於排序索引鍵中資料欄的順序。當查詢使用一個或多個次要排序資料欄時，交錯排序可大幅改善查詢效能。交錯排序在執行資料載入和清空操作時，會產生很少的額外負荷成本，

Important

不要在具有依序增加屬性 (如身分資料欄、日期或時間戳記) 的資料欄上使用交錯排序索引鍵。

ENCODE AUTO

讓 Amazon Redshift 能夠自動調整資料表中所有資料欄的編碼類型，以最佳化查詢效能。 `ENCODE AUTO` 會保留您在建立資料表時指定的初始編碼類型。然後，如果 Amazon Redshift 判斷新的編碼類型可以改善查詢效能，Amazon Redshift 就可以變更資料表資料欄的編碼類型。 `ENCODE AUTO` 如果您未在資料表的任何資料欄上指定編碼類型，則為預設值。

UNIQUE (column_name [, ...])

此限制條件會指定，資料表中一個包含一個或多個資料欄的群組只能包含唯一值。唯一資料表限制條件的行為與資料欄限制條件的行為相同，但多了可橫跨多個資料欄的額外功能。在唯一限制條件的細節中，不會將 `null` 值視為相等。每個唯一資料表限制條件必須命名一組資料欄，這組資料欄有別於為資料表所定義的任何其他唯一或主索引鍵限制條件所命名的資料欄組。

Important

唯一限制條件僅供參考，系統不會強制執行它們。

PRIMARY KEY (column_name [, ...])

此限制條件會指定，資料表中的一個或多個資料欄只能包含唯一的 (不重複) 非 null 值。將一組資料欄識別為主索引鍵也會提供有關結構描述設計的中繼資料。主索引鍵表示，其他資料表可倚賴這組資料欄做為資料列的唯一識別碼。一個資料表中可指定一個主索引鍵，無論是單一資料欄限制條件或資料表限制條件都可行。主索引鍵限制條件應命名一組資料欄，這組資料欄有別於由為相同資料表所定義的任何唯一限制條件命名的其他資料欄組。

Important

主索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

FOREIGN KEY (column_name [, ...]) REFERENCES 可參考 [(refcolumn)]

此限制條件會指定外部索引鍵限制條件，其要求新資料表中一個包含一個或多個資料欄的群組包含的值，只能是符合參考資料表中某一資料列之參考資料欄的值。如果省略 refcolumn，則會使用 reftable 的主索引鍵。參考資料欄必須為參考資料表中唯一或主索引鍵限制條件的資料欄。

Important

外部索引鍵條件限制僅供參考。系統不會強制執行這些限制，不過規劃器會使用這些限制。

使用須知

唯一性、主索引鍵和外部索引鍵限制僅供參考，Amazon Redshift 不會在您填入資料表時強制執行它們。例如，如果您將資料插入具有相依性的資料表中，即使插入違反限制也可以成功執行。不過，主要金鑰和外部金鑰會用作規劃提示，如果您 ETL 的程序或應用程式中的其他程序強制執行其完整性，則應宣告這些提示。如需如何捨棄具有相依性之資料表的資訊，請參閱 [DROP TABLE](#)。

限制和配額

建立資料表時，請考量下列限制。

- 根據節點類型，叢集中的資料表數目上限有所限制。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的[限制](#)。
- 資料表名稱的字元數上限為 127 個。
- 單一資料表中可定義的資料欄數目上限為 1,600 個。

- 您可以在單一資料表中定義的SORTKEY資料欄數目上限為 400。

欄位層級設定和資料表層級設定的摘要

在資料欄層級或資料表層級可設定數種屬性和設定。某些情況下，在資料欄層級或資料表層級設定屬性或限制條件的效果相同。而有些情況下則會產生不同的結果。

下列清單摘要說明資料欄層級和資料表層級設定：

DISTKEY

在資料欄層級或資料表層級設定的效果並無差異。

如果 DISTKEY 設定為 `DISTKEY`，無論是在資料欄層級還是在資料表層級，DISTSTYLE 都必須設定為 `DISTKEY` 或完全不設定。DISTSTYLE 只能在資料表層級設定。

SORTKEY

如果在資料欄層級設定，SORTKEY 必須是單一資料欄。如果在資料表層級 SORTKEY 設定，則一或多個資料欄可以組成複合或交錯的複合排序索引鍵。

COLLATE CASE_SENSITIVE | COLLATE CASE_INSENSITIVE

Amazon Redshift 不支援修改資料欄的區分大小寫組態。當您將新的資料欄附加到資料表時，Amazon Redshift 會使用預設值來區分大小寫。附加新資料欄時，Amazon Redshift 不支援 COLLATE 關鍵字。

如需如何使用資料庫定序建立資料庫的資訊，請參閱 [CREATE DATABASE](#)。

如需 COLLATE 函數的資訊，請參閱 [COLLATE 函數](#)。

UNIQUE

在資料欄層級，一個或多個金鑰可以設定為 UNIQUE；UNIQUE 限制條件個別套用至每個資料欄。如果在資料表層級設定 UNIQUE，則一個或多個資料欄可能會構成複合 UNIQUE 限制條件。

PRIMARY KEY

如果在資料欄層級設定，PRIMARY KEY 必須是單一資料欄。如果在資料表層級設定 PRIMARY KEY，則一或多個資料欄可以組成複合主索引鍵。

FOREIGN KEY

無論是在資料欄層級還是在資料表層級設定 FOREIGN KEY，都不會有任何效果差異。在資料欄層級的語法為單純的 REFERENCES reftable [(refcolumn)]。

傳入資料的分佈

當傳入資料的雜湊分佈機制與目標資料表的機制相同時，不需要在資料載入時實際分佈資料。例如，如果新資料表設定了分佈索引鍵，而且要從另一個分佈在相同索引鍵資料欄上的資料表插入資料，則會使用相同的節點和分割就地載入資料。不過，如果來源和目標資料表都設定為EVEN分佈，資料會重新分佈到目標資料表。

寬資料表

您可能可以建立非常寬的資料表，但無法在資料表上執行查詢處理，例如 INSERT 或 SELECT 陳述式。具有固定寬度資料欄的資料表，例如 CHAR，其寬度上限為 64KB - 1（或 65535 位元組）。如果資料表包含 VARCHAR 資料欄，則資料表可以具有較大的宣告寬度，而不會傳回錯誤，因為資料 VARCHARS 欄不會將完整的宣告寬度貢獻至計算的查詢處理限制。資料 VARCHAR 欄的有效查詢處理限制會根據多種因素而有所不同。

如果資料表對於插入或選取操作來說太寬，您會收到下列錯誤。

```
ERROR: 8001
DETAIL: The combined length of columns processed in the SQL statement
exceeded the query-processing limit of 65535 characters (pid:7627)
```

範例

如需示範如何使用 CREATETABLE 命令的範例，請參閱 [範例](#) 主題。

範例

下列範例示範 Amazon Redshift CREATETABLE 陳述式中的各種資料欄和資料表屬性。如需 CREATE 的詳細資訊 TABLE，包括參數定義，請參閱 [CREATE TABLE](#)。

許多範例使用 TICKIT 範例資料集中的資料表和資料。如需詳細資訊，請參閱 [範本資料庫](#)。

您可以在 CREATE TABLE 命令中使用資料庫名稱和結構描述名稱作為資料表名稱的字首。例如：dev_database.public.sales。資料庫名稱必須是您所連線的資料庫。任何在另一個資料庫中建立資料庫物件的嘗試都會失敗，並顯示操作無效錯誤。

建立含有分佈索引鍵、複合排序索引鍵和壓縮的資料表

下列範例會在 TICKIT 資料庫中建立 SALES 資料表，並針對數個資料欄定義壓縮。LISTID 會宣告為分發金鑰，LISTID 並將 SELLERID 宣告為多欄複合排序金鑰。另外也會為資料表定義主索引鍵和外部索引鍵限制條件。在建立範例中的資料表之前，如果 UNIQUE 沒有限制，您可能需要將限制新增至外部索引鍵參考的每個資料欄。

```

create table sales(
salesid integer not null,
listid integer not null,
sellerid integer not null,
buyerid integer not null,
eventid integer not null encode mostly16,
dateid smallint not null,
qtysold smallint not null encode mostly8,
pricepaid decimal(8,2) encode delta32k,
commission decimal(8,2) encode delta32k,
saletime timestamp,
primary key(salesid),
foreign key(listid) references listing(listid),
foreign key(sellerid) references users(userid),
foreign key(buyerid) references users(userid),
foreign key(dateid) references date(dateid))
distkey(listid)
compound sortkey(listid,sellerid);

```

結果如下：

schemaname	tablename	column	type	encoding	distkey
	sortkey	notnull			
public	sales	salesid	integer	lzo	false
	0	true			
public	sales	listid	integer	none	true
	1	true			
public	sales	sellerid	integer	none	false
	2	true			
public	sales	buyerid	integer	lzo	false
	0	true			
public	sales	eventid	integer	mostly16	false
	0	true			
public	sales	dateid	smallint	lzo	false
	0	true			
public	sales	qtysold	smallint	mostly8	false
	0	true			
public	sales	pricepaid	numeric(8,2)	delta32k	false
	0	false			
public	sales	commission	numeric(8,2)	delta32k	false
	0	false			

```
public      | sales      | saletime   | timestamp without time zone | lzo      | false
|           | 0 | false
```

下列範例會使用不區分大小寫的資料欄 col1 建立資料表 t1。

```
create table T1 (
  col1 Varchar(20) collate case_insensitive
);

insert into T1 values ('bob'), ('john'), ('Tom'), ('JOHN'), ('Bob');
```

查詢資料表：

```
select * from T1 where col1 = 'John';

col1
-----
john
JOHN
(2 rows)
```

使用交錯排序索引鍵建立資料表

下列範例會使用交錯排序索引鍵建立CUSTOMER資料表。

```
create table customer_interleaved (
  c_custkey      integer      not null,
  c_name         varchar(25)  not null,
  c_address      varchar(25)  not null,
  c_city         varchar(10)  not null,
  c_nation       varchar(15)  not null,
  c_region       varchar(12)  not null,
  c_phone        varchar(15)  not null,
  c_mktsegment   varchar(10)  not null)
diststyle all
interleaved sortkey (c_custkey, c_city, c_mktsegment);
```

使用 IF 建立資料表 NOT EXISTS

下列範例會建立CITIES資料表，或 不執行任何動作，如果訊息已經存在，則傳回訊息：

```
create table if not exists cities(
```

```
cityid integer not null,
city varchar(100) not null,
state char(2) not null);
```

建立具有ALL分佈的資料表

下列範例會建立具有ALL分佈的VENUE資料表。

```
create table venue(
venueid smallint not null,
venuename varchar(100),
venuecity varchar(30),
venuestate char(2),
venueseats integer,
primary key(venueid))
diststyle all;
```

建立具有EVEN分佈的資料表

下列範例會建立名為 `MYEVENT` 的資料表，`MYEVENT` 其中包含三個資料欄。

```
create table myevent(
eventid int,
eventname varchar(200),
eventcity varchar(30))
diststyle even;
```

資料表會均勻分佈且不會排序。資料表沒有宣告DISTKEY或SORTKEY資料欄。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'myevent';
```

column	type	encoding	distkey	sortkey
eventid	integer	lzo	f	0
eventname	character varying(200)	lzo	f	0
eventcity	character varying(30)	lzo	f	0

(3 rows)

建立LIKE另一個資料表的暫存資料表

下列範例會建立名為 `TEMPEVENT` 的暫存資料表，此資料表會從EVENT資料表繼承其資料欄。

```
create temp table tempevent(like event);
```

此資料表也會繼承其父資料表的 DISTKEY 和 SORTKEY 屬性：

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'tempevent';
```

column	type	encoding	distkey	sortkey
eventid	integer	none	t	1
venueid	smallint	none	f	0
catid	smallint	none	f	0
dateid	smallint	none	f	0
eventname	character varying(200)	lzo	f	0
starttime	timestamp without time zone	bytedict	f	0

(6 rows)

使用資料 IDENTITY 欄建立資料表

下列範例會建立名為 VENUE_ 的資料表 IDENTITY，其中具有名為的 IDENTITY 欄 VENUEID。此資料欄從 0 開始，並隨每筆記錄遞增 1。VENUEID 也會宣告為資料表的主要索引鍵。

```
create table venue_ident(venueid bigint identity(0, 1),
venueid varchar(100),
venuecity varchar(30),
venuestate char(2),
venuestate integer,
primary key(venueid));
```

使用預設 IDENTITY 資料欄建立資料表

以下範例會建立名為 t1 的資料表。此資料表具有名為的資料 IDENTITY 欄 hist_id 和名為的預設 IDENTITY 資料欄 base_id。

```
CREATE TABLE t1(
  hist_id BIGINT IDENTITY NOT NULL, /* Cannot be overridden */
  base_id BIGINT GENERATED BY DEFAULT AS IDENTITY NOT NULL, /* Can be overridden */
  business_key varchar(10) ,
  some_field varchar(10)
);
```

將資料列插入至資料表顯示已同時產生 `hist_id` 及 `base_id` 值。

```
INSERT INTO T1 (business_key, some_field) values ('A','MM');
```

```
SELECT * FROM t1;
```

hist_id	base_id	business_key	some_field
1	1	A	MM

插入第二列顯示已產生 `base_id` 的預設值。

```
INSERT INTO T1 (base_id, business_key, some_field) values (DEFAULT, 'B','MNOP');
```

```
SELECT * FROM t1;
```

hist_id	base_id	business_key	some_field
1	1	A	MM
2	2	B	MNOP

插入第三列顯示 `base_id` 的值不需要是唯一的。

```
INSERT INTO T1 (base_id, business_key, some_field) values (2,'B','MNNN');
```

```
SELECT * FROM t1;
```

hist_id	base_id	business_key	some_field
1	1	A	MM
2	2	B	MNOP
3	2	B	MNNN

建立具有DEFAULT資料欄值的資料表

下列範例會建立宣告每欄預設值的CATEGORYDEF資料表：

```
create table categorydef(
  catid smallint not null default 0,
```



```
catgroup varchar(10) default 'Special',
catname varchar(10) default 'Other',
catdesc varchar(50) default 'Special events',
primary key(catid));

insert into categorydef values(default,default,default,default);
```

```
select * from categorydef;
```

```
 catid | catgroup | catname |   catdesc
-----+-----+-----+-----
      0 | Special  | Other   | Special events
(1 row)
```

DISTSTYLE、DISTKEY和 SORTKEY選項

下列範例顯示 DISTKEY、 SORTKEY和 DISTSTYLE選項的運作方式。在此範例中，COL1是分發金鑰；因此，分發樣式必須設定為 KEY或未設定為。根據預設，資料表沒有排序索引鍵，因此不會排序：

```
create table t1(col1 int distkey, col2 int) diststyle key;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't1';
```

```
column | type   | encoding | distkey | sortkey
-----+-----+-----+-----+-----
col1   | integer | az64     | t       | 0
col2   | integer | az64     | f       | 0
```

在下列範例中，會將同一個資料欄定義為分佈索引鍵和排序索引鍵。同樣地，分佈樣式必須設定為 KEY 或未設定為。

```
create table t2(col1 int distkey sortkey, col2 int);
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't2';
```

```
column | type   | encoding | distkey | sortkey
-----+-----+-----+-----+-----
```

```
col1 | integer | none | t | 1
col2 | integer | az64 | f | 0
```

在下列範例中，沒有資料欄設定為分佈索引鍵，COL2設定為排序索引鍵，分佈樣式設定為 ALL：

```
create table t3(col1 int, col2 int sortkey) diststyle all;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't3';
```

Column	Type	Encoding	DistKey	SortKey
col1	integer	az64	f	0
col2	integer	none	f	1

在下列範例中，分佈樣式設定為 EVEN，而且未明確定義排序索引鍵；因此資料表會平均分佈，但不會排序。

```
create table t4(col1 int, col2 int) diststyle even;
```

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 't4';
```

column	type	encoding	distkey	sortkey
col1	integer	az64	f	0
col2	integer	az64	f	0

使用 ENCODEAUTO 選項建立資料表

下列範例會建立具有自動壓縮編碼的 t1 資料表。ENCODE AUTO 當您未指定任何資料欄的編碼類型時，是資料表的預設值。

```
create table t1(c0 int, c1 varchar);
```

下列範例會指定來建立 t2 具有自動壓縮編碼 ENCODE 的資料表 AUTO。

```
create table t2(c0 int, c1 varchar) encode auto;
```

下列範例會指定 來建立 t3 具有自動壓縮編碼 ENCODE 的資料表 AUTO。資料欄的定義初始編碼類型 c0 為 DELTA。如果其他編碼提供更好的查詢效能，Amazon Redshift 可以變更編碼。

```
create table t3(c0 int encode delta, c1 varchar) encode auto;
```

下列範例會指定 來建立 t4 具有自動壓縮編碼 ENCODE 的資料表 AUTO。資料欄的初始編碼 c0 為 定義 DELTA，資料欄的初始編碼 c1 為 定義 LZO。如果其他編碼提供更好的查詢效能，Amazon Redshift 可以變更這些編碼。

```
create table t4(c0 int encode delta, c1 varchar encode lzo) encode auto;
```

CREATE TABLE AS

主題

- [語法](#)
- [參數](#)
- [CTAS 用量備註](#)
- [CTAS 範例](#)

根據查詢建立新資料表。此資料表的擁有者是發出命令的使用者。

載入的新資料表包含命令中的查詢所定義的資料。資料表資料欄的名稱和資料類型會與查詢的輸出資料欄相關聯。CREATE TABLE AS (CTAS) 命令會建立新的資料表，並評估要載入新資料表的查詢。

語法

```
CREATE [ [ LOCAL ] { TEMPORARY | TEMP } ]  
TABLE table_name  
[ ( column_name [, ... ] ) ]  
[ BACKUP { YES | NO } ]  
[ table_attributes ]  
AS query  
  
where table_attributes are:  
[ DISTSTYLE { AUTO | EVEN | ALL | KEY } ]  
[ DISTKEY( distkey_identifier ) ]  
[ [ COMPOUND | INTERLEAVED ] SORTKEY( column_name [, ...] ) ]
```

參數

LOCAL

雖然陳述式中可接受此選用關鍵字，但是在 Amazon Redshift 中沒有作用。

TEMPORARY | TEMP

建立臨時資料表。臨時資料表會在其建立所在的工作階段結束時自動捨棄。

table_name

要建立的資料表名稱。

Important

若您指定 '#' 開頭的資料表名稱，所建立的資料表會是臨時資料表。例如：

```
create table #newtable (id) as select * from oldtable;
```

資料表名稱長度上限為 127 個位元組；超過此長度的名稱會截斷至 127 個位元組。Amazon Redshift 會依節點類型強制執行每個叢集的資料表數量配額限制。資料表名稱也可透過資料庫和結構描述名稱來限定，如下表所示。

```
create table tickit.public.test (c1) as select * from oldtable;
```

在此範例中，tickit 是資料庫名稱，而 public 是結構描述名稱。如果資料庫或結構描述不存在，則陳述式會傳回錯誤。

若提供結構描述名稱，則會在該結構描述中建立新資料表 (假設建立者具有存取結構描述的權限)。資料表名稱對於該結構描述來說必須是唯一的。如果未指定結構描述，則會使用目前資料庫結構描述建立資料表。如果您要建立臨時資料表，就不能指定結構描述名稱，因為臨時資料表會採用特殊結構描述。

若臨時資料表是在不同的工作階段中建立的話，則允許在同一個資料庫中同時有多個同名的臨時資料表存在。這些資料表會指派至不同的結構描述。

column_name

新資料表中資料欄的名稱。若未提供任何資料欄名稱，則會採用查詢的輸出資料欄名稱做為資料欄名稱。表達式會使用預設資料欄名稱。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

BACKUP { YES | NO }

此子句會指定資料表是否應包含在自動化和手動叢集快照中。

對於不包含關鍵資料的暫存資料表等資料表，請指定 BACKUP NO，以在建立快照和從快照還原時節省處理時間，以及減少 Amazon Simple Storage Service 上的儲存空間。BACKUP NO 設定不會影響資料自動複寫至叢集內其他節點，因此在節點失敗的情況下，會還原具有指定 BACKUP NO 的資料表。預設值為 BACKUP YES。

DISTSTYLE { AUTO | EVEN | KEY | ALL }

定義整個資料表的資料分佈樣式的關鍵字。Amazon Redshift 會根據資料表上指定的分佈樣式，將資料表的資料列分散到運算節點。預設值為 DISTSTYLE AUTO。

您為資料表選取的分佈樣式會影響資料庫的整體效能。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

- AUTO：Amazon Redshift 會根據資料表資料指派最佳分佈樣式。若要檢視套用至資料表的分佈樣式，請查詢 PG_CLASS 系統目錄資料表。如需詳細資訊，請參閱[檢視分佈樣式](#)。
- EVEN：資料表中的資料平均分散到循環配置中叢集中的節點。資料列IDs用於決定分佈，且大約相同的資料列數目會分佈到每個節點。這是預設分佈方法。
- KEY：資料由DISTKEY資料欄中的值分佈。當您將聯結資料表的聯結資料欄設定為分佈索引鍵時，兩個資料表的聯結資料列會在運算節點上並存。資料並存時，最佳化工具就能更有效率地執行聯結。如果您指定 DISTSTYLE KEY，則必須命名資料DISTKEY欄。
- ALL：整個資料表的副本會分發給每個節點。此分佈樣式可確保每個節點上都有任何聯結所需的所有資料列，但儲存空間的需求也會倍增，並且會增加資料表的負載和維護次數。ALL 與某些維度資料表搭配使用時，分佈可以改善執行時間，因為KEY分佈不適當，但效能改善必須權衡維護成本。

DISTKEY (欄)

指定分佈索引鍵的資料欄名稱或位置編號。使用資料表的選用資料欄清單中所指定的名稱，或是查詢的選取清單中指定的名稱。或者，使用位置編號，其中選取的第一個資料欄為 1、第二個為 2，以此類推。資料表中只能有一個資料欄是分佈索引鍵：

- 如果您將資料欄宣告為資料DISTKEY欄，DISTSTYLE則必須設定為 KEY或完全不設定為。
- 如果您沒有宣告資料DISTKEY欄，您可以將 DISTSTYLE設定為 EVEN。
- 如果您未指定 DISTKEY或 DISTSTYLE，會根據子SELECT句的查詢計畫來CTAS決定新資料表的分佈樣式。如需詳細資訊，請參閱[欄位和資料表屬性的繼承](#)。

您可以將同一個資料欄定義為分佈索引鍵和排序索引鍵；此方法主要目的是在所指的資料欄要與查詢中的資料欄聯結時加速聯結。

【 COMPOUND | INTERLEAVED 】 SORTKEY (column_name [, ...])

指定資料表的一個或多個排序索引鍵。當資料載入資料表時，資料會依指定為排序索引鍵的資料欄排序。

您可以選擇性地指定COMPOUND或INTERLEAVED排序樣式。預設值為 COMPOUND。如需詳細資訊，請參閱[排序金鑰](#)。

您最多可以定義 400 個COMPOUNDSORTKEY資料欄或每個資料表 8 個資料 INTERLEAVEDSORTKEY欄。

如果您未指定 SORTKEY，會根據子SELECT句的查詢計畫CTAS，決定新資料表的排序索引鍵。如需詳細資訊，請參閱[欄位和資料表屬性的繼承](#)。

COMPOUND

指定使用複合索引鍵排序資料，該索引鍵是由列出的所有資料欄組成，並依其列出順序排列。當查詢依據排序資料欄的順序掃描資料列時，複合排序索引鍵最實用。當查詢依賴次要排序資料欄時，使用複合索引鍵排序的效能優勢就會降低。每個資料表最多可定義 400 個 COMPOUNDSORTKEY資料欄。

INTERLEAVED

指定使用交錯排序索引鍵排序資料。最多可以為交錯排序索引鍵指定八個資料欄。

交錯排序索引鍵對排序索引鍵中的每個資料欄 (或資料欄子集) 都提供相等的權重，所以查詢不會取決於排序索引鍵中資料欄的順序。當查詢使用一個或多個次要排序資料欄時，交錯排序可大幅改善查詢效能。交錯排序在執行資料載入和清空操作時，會產生很少的額外負荷成本，

AS query

Amazon Redshift 支援的任何查詢 (SELECT 陳述式)。

CTAS 用量備註

限制

Amazon Redshift 會依節點類型強制執行每個叢集的資料表數量配額限制。

資料表名稱的字元數上限為 127 個。

單一資料表中可定義的資料欄數目上限為 1,600 個。

欄位和資料表屬性的繼承

CREATE TABLE AS (CTAS) 資料表不會從建立的資料表繼承限制、身分資料欄、預設資料欄值或主索引鍵。

您無法指定CTAS資料表的資料欄壓縮編碼。Amazon Redshift 會自動指派壓縮編碼，如下所示：

- 定義為排序索引鍵的資料欄會被指派RAW壓縮。
- 定義為 BOOLEAN、REAL、DOUBLEPRECISION、GEOMETRY或 GEOGRAPHY資料類型的資料欄會被指派RAW壓縮。
- 定義為 SMALLINT、INTEGER、BIGINT、DECIMALDATE、TIME、TIMESTAMP、或 TIMETZ的資料欄TIMESTAMPTZ會指派AZ64壓縮。
- 定義為 CHAR、VARCHAR或 的資料欄VARBYTE會被指派LZO壓縮。

如需詳細資訊，請參閱 [壓縮編碼](#) 和 [資料類型](#)。

若要明確指派資料欄編碼，請使用 [CREATE TABLE](#)。

CTAS 根據子SELECT句的查詢計畫，決定新資料表的分佈樣式和排序索引鍵。

對於複雜的查詢，例如包含聯結、彙總、依子句排序或限制子句的查詢，CTAS 會盡最大努力根據查詢計畫選擇最佳分佈樣式和排序金鑰。

Note

為了在大型資料集或複雜查詢上發揮最佳效能，建議您使用一般資料集進行測試。

您通常可以透過CTAS檢查查詢計劃來預測選擇哪個分佈金鑰和排序金鑰，以查看查詢最佳化工具選擇哪些資料欄進行排序和分發資料。如果查詢計劃的頂端節點是從單一資料表（ XN 序列掃描 ）進行簡單的循序掃描，則 CTAS通常會使用來源資料表的分佈樣式和排序索引鍵。如果查詢計畫的頂端節點是任何其他序列掃描（ 例如 XN Limit、XN Sort HashAggregate、XN 等 ），CTAS 會盡最大努力根據查詢計畫選擇最佳分佈樣式和排序索引鍵。

例如，假設您使用下列子SELECT句類型建立五個資料表：

- 簡單 select 陳述式
- 限制子句

- 使用 依子句排序 LISTID
- 使用的子句順序 QTYSOLD
- 依子句分組的SUM彙總函數。

下列範例顯示每個CTAS陳述式的查詢計劃。

```
explain create table sales1_simple as select listid, dateid, qtysold from sales;
          QUERY PLAN
```

```
-----
XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=8)
(1 row)
```

```
explain create table sales2_limit as select listid, dateid, qtysold from sales limit
100;
```

```
          QUERY PLAN
```

```
-----
XN Limit (cost=0.00..1.00 rows=100 width=8)
-> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=8)
(2 rows)
```

```
explain create table sales3_orderbylistid as select listid, dateid, qtysold from sales
order by listid;
```

```
          QUERY PLAN
```

```
-----
XN Sort (cost=1000000016724.67..1000000017155.81 rows=172456 width=8)
Sort Key: listid
-> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=8)
(3 rows)
```

```
explain create table sales4_orderbyqty as select listid, dateid, qtysold from sales
order by qtysold;
```

```
          QUERY PLAN
```

```
-----
XN Sort (cost=1000000016724.67..1000000017155.81 rows=172456 width=8)
Sort Key: qtysold
-> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=8)
(3 rows)
```



```
explain create table sales5_groupby as select listid, dateid, sum(qtysold) from sales
group by listid, dateid;
```

QUERY PLAN

```
-----
XN HashAggregate (cost=3017.98..3226.75 rows=83509 width=8)
  -> XN Seq Scan on sales (cost=0.00..1724.56 rows=172456 width=8)
(2 rows)
```

若要檢視每個資料表的分發金鑰和排序金鑰，請查詢 PG_TABLE_DEF 系統目錄表，如下所示。

```
select * from pg_table_def where tablename like 'sales%';
```

tablename	column	distkey	sortkey
sales	salesid	f	0
sales	listid	t	0
sales	sellerid	f	0
sales	buyerid	f	0
sales	eventid	f	0
sales	dateid	f	1
sales	qtysold	f	0
sales	pricepaid	f	0
sales	commission	f	0
sales	saletime	f	0
sales1_simple	listid	t	0
sales1_simple	dateid	f	1
sales1_simple	qtysold	f	0
sales2_limit	listid	f	0
sales2_limit	dateid	f	0
sales2_limit	qtysold	f	0
sales3_orderbylistid	listid	t	1
sales3_orderbylistid	dateid	f	0
sales3_orderbylistid	qtysold	f	0
sales4_orderbyqty	listid	t	0
sales4_orderbyqty	dateid	f	0
sales4_orderbyqty	qtysold	f	1
sales5_groupby	listid	f	0
sales5_groupby	dateid	f	0
sales5_groupby	sum	f	0

下表顯示結果摘要。為了簡化起見，我們省略了說明計畫中的成本、資料列和寬度詳細資訊。

資料表	CTAS SELECT 陳述式	說明計劃頂端節點	分佈索引鍵	排序索引鍵
S1_SIMPLE	<code>select listid, dateid, qtysold from sales</code>	XN Seq Scan on sales ...	LISTID	DATEID
S2_LIMIT	<code>select listid, dateid, qtysold from sales limit 100</code>	XN Limit ...	無 (EVEN)	無
S3ORDER_B Y_LISTID	<code>select listid, dateid, qtysold from sales order by listid</code>	XN Sort ... Sort Key: listid	LISTID	LISTID
S4ORDER_B Y_QTY	<code>select listid, dateid, qtysold from sales order by qtysold</code>	XN Sort ... Sort Key: qtysold	LISTID	QTYSOLD
S5GROUP_B Y	<code>select listid, dateid, sum(qtysold) from sales group by listid, dateid</code>	XN HashAggre gate ...	無 (EVEN)	無

您可以在 CTAS 陳述式中明確指定分佈樣式和排序索引鍵。例如，下列陳述式會使用 EVEN 分佈建立資料表，並指定 SALESID 作為排序索引鍵。

```
create table sales_disteven
diststyle even
sortkey (salesid)
as
select eventid, venueid, dateid, eventname
from event;
```

壓縮編碼

ENCODE AUTO 會用作資料表的預設值。Amazon Redshift 會自動管理資料表中所有資料欄的壓縮編碼。

傳入資料的分佈

當傳入資料的雜湊分佈機制與目標資料表的機制相同時，不需要在資料載入時實際分佈資料。例如，如果新資料表設定了分佈索引鍵，而且要從另一個分佈在相同索引鍵資料欄上的資料表插入資料，則會使用相同的節點和分割就地載入資料。不過，如果來源和目標資料表都設定為EVEN分佈，資料會重新分佈到目標資料表。

自動ANALYZE操作

Amazon Redshift 會自動分析您使用CTAS命令建立的資料表。第一次建立這些資料表時，您不需要在這些資料表上執行 ANALYZE 命令。若您修改這些資料表，則應依照與其他資料表相同的方式進行分析。

CTAS 範例

下列範例會為資料表建立名為 EVENT_BACKUP 的EVENT資料表：

```
create table event_backup as select * from event;
```

產生的資料表會繼承EVENT資料表中的分佈和排序索引鍵。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'event_backup';
```

column	type	encoding	distkey	sortkey
catid	smallint	none	false	0
dateid	smallint	none	false	1
eventid	integer	none	true	0
eventname	character varying(200)	none	false	0
starttime	timestamp without time zone	none	false	0
venueid	smallint	none	false	0

下列命令EVENTDISTSORT會建立名為 的新資料表，方法是從EVENT資料表中選取四欄。新資料表會依 EVENTID和 分佈EVENTID並排序DATEID：

```
create table eventdistsort
```

```

distkey (1)
sortkey (1,3)
as
select eventid, venueid, dateid, eventname
from event;

```

結果如下所示：

```

select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdistsort';

```

column	type	encoding	distkey	sortkey
eventid	integer	none	t	1
venueid	smallint	none	f	0
dateid	smallint	none	f	2
eventname	character varying(200)	none	f	0

您可以使用分佈和排序索引鍵的資料欄名稱建立完全相同的資料表。例如：

```

create table eventdistsort1
distkey (eventid)
sortkey (eventid, dateid)
as
select eventid, venueid, dateid, eventname
from event;

```

以下陳述式會將均勻分佈套用至資料表，但不會定義明確的排序索引鍵。

```

create table eventdisteven
diststyle even
as
select eventid, venueid, dateid, eventname
from event;

```

資料表不會從EVENT資料表（EVENTID）繼承排序索引鍵，因為已為新資料表指定EVEN分佈。新資料表沒有排序索引鍵，也沒有分佈索引鍵。

```

select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdisteven';

```

column	type	encoding	distkey	sortkey
eventid	integer	none	f	0
venueid	smallint	none	f	0
dateid	smallint	none	f	0
eventname	character varying(200)	none	f	0

以下陳述式會套用均勻分佈，並定義排序索引鍵：

```
create table eventdistevensort diststyle even sortkey (venueid)
as select eventid, venueid, dateid, eventname from event;
```

產生的資料表有排序索引鍵，但沒有分佈索引鍵。

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'eventdistevensort';
```

column	type	encoding	distkey	sortkey
eventid	integer	none	f	0
venueid	smallint	none	f	1
dateid	smallint	none	f	0
eventname	character varying(200)	none	f	0

下列陳述式會重新分配傳入資料中不同索引鍵欄上的EVENT資料表，該資料會在EVENTID資料欄上排序，並且未定義任何SORTKEY資料欄；因此資料表不會排序。

```
create table venuedistevent distkey(venueid)
as select * from event;
```

結果如下所示：

```
select "column", type, encoding, distkey, sortkey
from pg_table_def where tablename = 'venuedistevent';
```

column	type	encoding	distkey	sortkey
eventid	integer	none	f	0
venueid	smallint	none	t	0
catid	smallint	none	f	0
dateid	smallint	none	f	0
eventname	character varying(200)	none	f	0

```
starttime | timestamp without time zone | none | f | 0
```

CREATE USER

建立新的資料庫使用者。視權限和角色而定，資料庫使用者可以擷取資料、執行命令，以及在資料庫中執行其他動作。您必須是資料庫超級使用者才能執行此命令。

所需權限

以下是 CREATE 的必要權限USER：

- 超級使用者
- 具有 CREATEUSER 權限的使用者

語法

```
CREATE USER name [ WITH ]  
PASSWORD { 'password' | 'md5hash' | 'sha256hash' | DISABLE }  
[ option [ ... ] ]
```

where *option* can be:

```
CREATEDB | NOCREATEDB  
| CREATEUSER | NOCREATEUSER  
| SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }  
| IN GROUP groupname [, ... ]  
| VALID UNTIL 'abstime'  
| CONNECTION LIMIT { limit | UNLIMITED }  
| SESSION TIMEOUT limit  
| EXTERNALID external_id
```

參數

name

欲建立的使用者名稱。使用者名稱不可以是 PUBLIC。如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。

WITH

選用的關鍵字。WITH 被 Amazon Redshift 忽略

```
PASSWORD { 'password' | 'md5hash' | 'sha256hash' | DISABLE }
```

設定使用者的密碼。

根據預設，使用者可以變更自己擁有的密碼，除非密碼遭到停用。若要停用使用者的密碼，請指定 `DISABLE`。停用使用者的密碼時，會從系統中刪除密碼，使用者只能使用臨時 AWS Identity and Access Management (IAM) 使用者憑證登入。如需詳細資訊，請參閱[使用IAM身分驗證來產生資料庫使用者憑證](#)。只有超級使用者能夠啟用或停用密碼，您無法停用超級使用者的密碼。若要啟用密碼，請執行 [ALTER USER](#) 並指定密碼。

您可以指定純文字、MD5雜湊字串或SHA256雜湊字串的密碼。

Note

當您使用 AWS Management Console AWS CLI、或 Amazon Redshift 啟動新叢集時 API，您必須為初始資料庫使用者提供純文字密碼。您之後可以使用 [ALTER USER](#) 變更密碼。

若是純文字，密碼必須滿足下列限制條件：

- 長度必須為 8 到 64 個字元。
- 至少須包含一個大寫字母、一個小寫字母和一個數字。
- 它可以使用ASCII代碼為 33–126 的任何ASCII字元，但 ' (單引號)、" (雙引號)、\、/ 或 @ 除外。

作為以純文字傳遞CREATEUSER密碼參數更安全的替代方案，您可以指定包含密碼和使用名稱的字串MD5雜湊。

Note

當您指定MD5雜湊字串時，CREATEUSER命令會檢查是否有有效的MD5雜湊字串，但不會驗證字串的密碼部分。在此情況下可能會建立密碼，例如空字串，但此密碼無法用來登入資料庫。

若要指定MD5密碼，請遵循下列步驟：

1. 串連密碼和使用名稱。

例如，密碼 `ez` 和使用者 `user1` 的串連字串為 `ezuser1`。

- 將串連字串轉換為 32 個字元 MD5 雜湊字串。您可以使用任何 MD5 公用程式來建立雜湊字串。下列範例使用 Amazon Redshift [MD5 函數](#) 和串連運算子 (||) 來傳回 32 字元 MD5-hash 字串。

```
select md5('ez' || 'user1');

md5
-----
153c434b4b77c89e6b94f12c5393af5b
```

- 在 MD5 雜湊字串前面串連 'md5'，並提供串連字串作為 md5hash 引數。

```
create user user1 password 'md5153c434b4b77c89e6b94f12c5393af5b';
```

- 使用登入憑證登入資料庫。

例如，以 user1 的身分和密碼 ez 登入。

另一個安全替代方案是指定密碼字串的 SHA-256 雜湊；或者您可以提供自己的有效 SHA-256 摘要和用於建立摘要的 256 位元鹽。

- 摘要 — 雜湊函數的輸出。
- Salt — 隨機產生且可與密碼結合的資料，有助於減少雜湊函數輸出中的模式。

```
'sha256|Mypassword'
```

```
'sha256|digest|256-bit-salt'
```

在下列範例中，Amazon Redshift 會產生並管理 Salt。

```
CREATE USER admin PASSWORD 'sha256|Mypassword1';
```

在下列範例中，會提供用於建立摘要的有效 SHA-256 位元摘要和 256 位元鹽。

要指定密碼並使用自己的 Salt 對其進行雜湊，請按照以下步驟操作：

- 建立 256 位元 Salt。您可以透過使用任何十六進位字串產生器產生 64 個字元長的字串來取得 Salt。在此範例中，Salt 是 c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6。

2. 使用 FROM_HEX 函數將鹽轉換為二進位。這是因為SHA2函數需要鹽的二進位表示法。請參閱以下陳述式。

```
SELECT
FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6');
```

3. 使用 CONCAT函數將 salt 附加至您的密碼。在此範例中，密碼為 Mypassword1。請參閱以下陳述式。

```
SELECT
CONCAT('Mypassword1',FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6'));
```

4. 使用 SHA2函數從您的密碼和 salt 組合建立摘要。請參閱以下陳述式。

```
SELECT
SHA2(CONCAT('Mypassword1',FROM_HEX('c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6')),256);
```

5. 使用先前步驟中的摘要和 Salt，建立使用者。請參閱以下陳述式。

```
CREATE USER admin PASSWORD 'sha256|
821708135fcc42eb3afda85286dee0ed15c2c461d000291609f77eb113073ec2|
c721bff5d9042cf541ff7b9d48fa8a6e545c19a763e3710151f9513038b0f6c6';
```

6. 使用登入憑證登入資料庫。

例如，以 admin 的身分和密碼 Mypassword1 登入。

如果您在未指定雜湊函數的情況下以純文字設定密碼，則會使用使用者名稱作為 salt MD5 產生摘要。

CREATEDB | NOCREATEDB

CREATEDB 此選項可讓新使用者建立資料庫。預設值為 NOCREATEDB。

CREATEUSER | NOCREATEUSER

CREATEUSER 此選項會建立具有所有資料庫權限的超級使用者，包括 CREATE USER。預設值為 NOCREATEUSER。如需詳細資訊，請參閱[superuser](#)。

SYSLOG ACCESS { RESTRICTED | UNRESTRICTED }

此子句會指定使用者對 Amazon Redshift 系統資料表和檢視的存取層級。

具有 SYSLOG ACCESS RESTRICTED 許可的普通使用者只能在使用者可見的系統資料表和檢視中查看該使用者產生的資料列。預設值為 RESTRICTED。

具有 SYSLOG ACCESS UNRESTRICTED 許可的普通使用者可以查看使用者可見系統資料表和檢視中的所有資料列，包括其他使用者產生的資料列。UNRESTRICTED 不會讓一般使用者存取超級使用者可見的資料表。只有超級使用者可看見超級使用者可查看的資料表。

Note

若使用者擁有不受限制的系統資料表存取權限，該使用者就能查看其他使用者產生的資料。例如，STL_QUERY 和 STL_QUERYTEXT 包含 INSERT、UPDATE 和 DELETE 陳述式的全文，其中可能包含敏感的使用者產生資料。

所有使用者都可看見 SVV_TRANSACTIONS 中的所有資料列。

如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

IN GROUP 群組名稱

指定使用者所屬現有群組的名稱。可列出多個群組名稱。

VALID UNTIL abstime

VALID UNTIL 選項會設定絕對時間，在此之後，使用者的密碼就不再有效。根據預設，密碼沒有時間限制。

CONNECTION LIMIT { limit | UNLIMITED }

允許使用者同時開啟的資料庫連線數目上限。超級使用者不受此限制規範。使用 UNLIMITED 關鍵字來允許並行連線的數量上限。另外也可能限制每個資料庫的連線數目。如需詳細資訊，請參閱[CREATE DATABASE](#)。預設值為 UNLIMITED。若要檢視目前連線數目，請查詢 [STV_SESSIONS](#) 系統畫面。

Note

如果同時套用使用者和資料庫連線數目限制，則必須在使用者嘗試連線時，在不超過這兩項限制的情況下提供一個未使用的連線位置。

SESSION TIMEOUT 限制

工作階段保持非作用中或閒置的時間上限 (以秒為單位)。範圍是 60 秒 (一分鐘) 到 1,728,000 秒 (20 天)。如果未為使用者設定工作階段逾時，則會套用叢集設定。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 中的配額和限制](#)。

當您設定工作階段逾時時，只會套用至新的工作階段。

若要檢視有關作用中使用者工作階段的資訊 (包括開始時間、使用者名稱和工作階段逾時)，請查詢 [STV_SESSIONS](#) 系統檢視。若要檢視有關使用者工作階段歷史記錄的資訊，請查詢 [STL_SESSIONS](#) 檢視。若要擷取有關資料庫使用者的資訊 (包括工作階段逾時值)，請查詢 [SVL_USER_INFO](#) 檢視。

EXTERNALID external_id

與身分提供者相關聯的使用者識別碼。使用者必須停用密碼。如需詳細資訊，請參閱 [Amazon Redshift 的原生身分提供者 \(IdP\) 聯合](#)。

使用須知

根據預設，所有使用者在結構描述上都有 PUBLIC CREATE 和 USAGE 權限。若要禁止使用者在資料庫的 PUBLIC 結構描述中建立物件，請使用 REVOKE 命令來移除該權限。

使用 IAM 身分驗證建立資料庫使用者憑證時，您可能想要建立只能使用臨時憑證登入的超級使用者。您無法停用超級使用者的密碼，但您可以使用隨機產生的 MD5 雜湊字串建立未知的密碼。

```
create user iam_superuser password 'md5A1234567890123456780123456789012' createuser;
```

無論 `enable_case_sensitive_identifier` 組態選項的設定為何，都會保留以雙引號括住的 `username` 大小寫。如需詳細資訊，請參閱 [enable_case_sensitive_identifier](#)。

範例

以下命令會建立名為 `dbuser` 的使用者與密碼 "abcD1234"、建立資料庫的權限，以及 30 個連線限制。

```
create user dbuser with password 'abcD1234' createdb connection limit 30;
```

查詢 `PG_USER_INFO` 目錄資料表，以檢視資料庫使用者的詳細資訊。

```
select * from pg_user_info;
```

```

username | usesysid | usecreatedb | usesuper | usecatupd | passwd | valuntil |
useconfig | useconlimit
-----+-----+-----+-----+-----+-----+-----
+-----+-----
rdsdb    |          1 | true        | true     | true      | ***** | infinity |
|
adminuser |         100 | true        | true     | false     | ***** |          |
| UNLIMITED
dbuser    |         102 | true        | false    | false     | ***** |          |
| 30

```

在以下範例中，帳戶密碼的有效期限至 2017 年 6 月 10 日。

```
create user dbuser with password 'abcD1234' valid until '2017-06-10';
```

以下範例會建立使用者，以及包含特殊字元且區分大小寫的密碼。

```
create user newman with password '@AbC4321!';
```

若要在 MD5 密碼中使用反斜線（`\'`），請在來源字串中使用反斜線來逸出反斜線。以下範例會建立名為 slashpass 的使用者，並使用單一反斜線（`\'`）做為密碼。

```
select md5('\|'|'slashpass');

md5
-----
0c983d1a624280812631c5389e60d48c
```

使用 md5 密碼建立使用者。

```
create user slashpass password 'md50c983d1a624280812631c5389e60d48c';
```

下列範例會建立將閒置工作階段逾時設定為 120 秒的使用者 dbuser。

```
CREATE USER dbuser password 'abcD1234' SESSION TIMEOUT 120;
```

以下範例會建立名為 bob 的使用者。命名空間為 myco_aad。這只是範例。若要成功執行命令，您必須擁有已註冊的身分提供者。如需詳細資訊，請參閱 [Amazon Redshift 的原生身分提供者 \(IdP\) 聯合](#)。

```
CREATE USER myco_aad:bob EXTERNALID "ABC123" PASSWORD DISABLE;
```

CREATE VIEW

在資料庫中建立檢視。檢視不會實際具體化；定義檢視的查詢會在每次查詢中參考檢視時執行。若要使用外部資料表建立檢視，請包含 `WITH NO SCHEMABINDING` 子句。

若要建立標準檢視，您需要存取基礎資料表或基礎檢視。若要查詢標準檢視，您需要選取檢視本身的許可，但不需要選取基礎資料表的許可。如果您建立參照其他結構描述中資料表或視觀表的視觀表，或是建立參照具體化視觀表的視觀表，則需要使用權限。若要查詢近期繫結視觀表，您需要選取近期繫結視觀表本身的權限。您也應該確保延遲繫結檢視的擁有者擁有參考物件（資料表、檢視或使用使用者定義函數）的選取許可。如需近期繫結視觀表的詳細資訊，請參閱 [使用須知](#)。

所需的許可

若要使用 `CREATE VIEW`，需要下列其中一個許可。

- 若要使用 `CREATE` **【OR】** `REPLACE` 建立檢視VIEW：
 - 超級使用者
 - 具有 `CREATE` **【】** `REPLACE VIEW` 許可的使用者
- 若要使用 `CREATE OR REPLACE` 取代現有檢視VIEW：
 - 超級使用者
 - 具有 `CREATE` **【OR】** `REPLACE VIEW` 許可的使用者
 - 檢視擁有者

如果使用者想要存取包含使用者定義函數的檢視，使用者必須擁有該函數的 `EXECUTE` 許可。

語法

```
CREATE [ OR REPLACE ] VIEW name [ ( column_name [, ...] ) ] AS query  
[ WITH NO SCHEMA BINDING ]
```

參數

或 REPLACE

如果有同名的檢視存在，則會取代該檢視。您只能將檢視取代為使用相同的資料欄名稱和資料類型產生一組相同資料欄的新查詢。CREATE OR 會REPLACEVIEW鎖定檢視以進行讀取和寫入，直到操作完成為止。

當檢視被替換時，其他屬性 (例如所有權和授予的權限) 會保留下來。

name

檢視的名稱。如果有提供結構描述名稱 (例如 myschema.myview)，則會使用指定的結構描述建立檢視。否則，檢視會在目前結構描述中建立。檢視名稱必須與相同結構描述中的任何其他檢視或資料表名稱不同。

若您指定 '#' 開頭的檢視名稱，所建立的檢視會是臨時檢視，而且只可在目前工作階段中看見。

如需有效名稱的相關資訊，請參閱 [名稱與識別碼](#)。您無法在系統資料庫 template0、template1 和 padb_harvest 或 sys:internal中建立資料表或視觀表。

column_name

選用的名稱清單，將用於檢視中的資料欄。若未提供任何資料欄名稱，則會從查詢衍生資料欄名稱。單一檢視中可定義的資料欄數目上限為 1,600 個。

query

評估為資料表的查詢 (以SELECT陳述式的形式)。此資料表會定義檢視中的資料欄和資料列。

WITH 否 SCHEMA BINDING

此子句會指定檢視不會與底層資料庫物件繫結，像是資料表和使用使用者定義的函數。因此，檢視和其參考的物件之間並無相依性。即使參考的物件不存在，您仍可以建立檢視。由於沒有相依性，您可以捨棄或修改參考物件，而不會影響檢視。Amazon Redshift 在查詢檢視之前不會檢查相依性。若要檢視近期繫結視觀表的詳細資訊，請執行 [PG_GET_LATE_BINDING_VIEW_COLS](#) 函數。

當您包含 WITH NO SCHEMABINDING子句時，SELECT陳述式中參考的資料表和檢視必須使用結構描述名稱來驗證。檢視建立時，結構描述必須存在，即使參考的資料表不存在也一樣。例如，下列陳述式會傳回錯誤。

```
create view myevent as select eventname from event
with no schema binding;
```

下列陳述式會成功執行。

```
create view myevent as select eventname from public.event
with no schema binding;
```

Note

您無法在檢視中進行更新、插入或刪除操作。

使用須知

近期繫結檢視

在對檢視進行查詢之前，近期繫結檢視不會檢查底層資料庫物件，像是資料表和其他檢視。因此，您可以修改或捨棄底層物件，而不會捨棄和重新建立檢視。若您捨棄底層物件，對近期繫結檢視的查詢將會失敗。若對近期繫及檢視的查詢參考底層物件中不存在的資料欄，則查詢將會失敗。

若您捨棄後再次建立近期繫結檢視的基礎資料表或檢視，則會建立具有預設存取許可的新物件。您可能需要對將查詢檢視的使用者授予基礎物件的許可。

若要建立繫結延遲檢視，請包含 WITH NO SCHEMABINDING 子句。以下範例會建立不含結構描述繫結的檢視。

```
create view event_vw as select * from public.event
with no schema binding;
```

```
select * from event_vw limit 1;
```

eventid	venueid	catid	dateid	eventname	starttime
2	306	8	2114	Boris Godunov	2008-10-15 20:00:00

以下範例顯示您可以修改基礎資料表，但不重新建立檢視。

```
alter table event rename column eventname to title;
```

```
select * from event_vw limit 1;
```

```

eventid | venueid | catid | dateid | title           | starttime
-----+-----+-----+-----+-----+-----
      2 |     306 |     8 |   2114 | Boris Godunov | 2008-10-15 20:00:00

```

您只能在近期繫結檢視中參考 Amazon Redshift Spectrum 外部資料表。近期繫結檢視的應用方式之一，就是查詢 Amazon Redshift 和 Redshift Spectrum 這兩種資料表。例如，您可以使用 [UNLOAD](#) 命令將舊資料封存至 Amazon S3。然後建立參考 Amazon S3 中資料的 Redshift Spectrum 外部資料表，並建立查詢這兩種資料表的檢視。下列範例使用 UNIONALL 子句來加入 Amazon Redshift SALES 資料表和 Redshift Spectrum SPECTRUM.SALES 資料表。

```

create view sales_vw as
select * from public.sales
union all
select * from spectrum.sales
with no schema binding;

```

如需建立 Redshift Spectrum 外部資料表的相關資訊，包括 SPECTRUM.SALES 資料表，請參閱 [開始使用 Amazon Redshift Spectrum](#)。

當您從近期繫結檢視建立標準檢視時，標準檢視的定義會包含進行標準檢視時的近期繫結檢視定義。系統不會追蹤近期繫結檢視的相依性，因此不會在標準檢視中追蹤對近期繫結檢視所做的變更。

若要更新標準檢視以參考後期繫結檢視的最新定義，請執行 CREATE OR, REPLACEVIEW 並使用您用來建立標準檢視的初始檢視定義。

請參閱下列範例，了解如何從近期繫結檢視建立標準檢視。

```

create view sales_vw_lbv as
select * from public.sales
with no schema binding;

show view sales_vw_lbv;
                                Show View DDL statement
-----
create view sales_vw_lbv as select * from public.sales with no schema binding;
(1 row)

create view sales_vw as
select * from sales_vw_lbv;

```



```
show view sales_vw;
```

Show View DDL statement

```
-----  
SELECT sales_vw_lbv.price, sales_vw_lbv."region" FROM (SELECT sales.price,  
sales."region" FROM sales) sales_vw_lbv;  
(1 row)
```

請注意，標準檢視的DDL陳述式中顯示的繫結後期檢視會在建立標準檢視時定義，且之後不會更新您對繫結後期檢視所做的任何變更。

範例

範例命令使用一組名為TICKIT資料庫的物件和資料範例。如需詳細資訊，請參閱[範本資料庫](#)。

下列命令會從名為 `event` 的資料表建立名為 `myevent` 的檢視EVENT。

```
create view myevent as select eventname from event  
where eventname = 'LeAnn Rimes';
```

下列命令會從名為 `users` 的資料表建立名為 `myuser` 的檢視USERS。

```
create view myuser as select lastname from users;
```

下列命令會從名為 `users` 的資料表建立或取代名為 `myuser` 的檢視USERS。

```
create or replace view myuser as select lastname from users;
```

以下範例會建立不含結構描述繫結的檢視。

```
create view myevent as select eventname from public.event  
with no schema binding;
```

DEALLOCATE

解除配置預備陳述式。

語法

```
DEALLOCATE [PREPARE] plan_name
```

參數

PREPARE

此關鍵字為選用並且會予以忽略。

`plan_name`

要解除配置之預備陳述式的名稱。

使用須知

DEALLOCATE 用於將先前準備的SQL陳述式重新配置。若您未明確解除配置預備陳述式，則會在目前工作階段結束時將它解除配置。如需預備陳述式的相關資訊，請參閱 [PREPARE](#)。

另請參閱

[EXECUTE](#), [PREPARE](#)

DECLARE

定義新的資料指標。使用資料指標可從完整查詢的結果集中一次擷取幾個資料列。

擷取資料指標的第一列時，整個結果集會在領導節點上、記憶體中或磁碟上具體化 (如有需要)。由於在大型結果集內使用資料指標可能會對效能造成負面影響，因此建議您盡量使用替代方式。如需詳細資訊，請參閱[使用游標時的效能考量](#)。

您必須在交易區塊內宣告資料指標。每個工作階段中一次只能開啟一個資料指標。

如需詳細資訊，請參閱 [FETCH](#)、[CLOSE](#)。

語法

```
DECLARE cursor_name CURSOR FOR query
```

參數

`cursor_name`

新資料指標的名稱。

`query`

填入游標的SELECT陳述式。

DECLARE CURSOR 用量備註

如果您的用戶端應用程式使用ODBC連線，且查詢建立的結果集太大而無法容納記憶體，您可以使用游標將結果集串流至用戶端應用程式。當您使用資料指標時，整個結果集會在領導節點上具體化，然後您的用戶端就能以遞增方式擷取結果。

Note

若要ODBC在 Microsoft Windows 中啟用游標，請在ODBCDSN您用於 Amazon Redshift 的中啟用 Use Declare/Fetch 選項。我們建議您使用ODBCDSN選項對話方塊中的快取大小欄位，在多節點叢集上將ODBC快取大小設定為 4,000 或更高，以將往返次數降至最低。在單一節點叢集上，將 Cache Size (快取大小) 設定為 1,000。

由於使用資料指標可能會對效能造成負面影響，因此建議您盡量使用替代方式。如需詳細資訊，請參閱[使用游標時的效能考量](#)。

在滿足以下限制的情況下，可支援 Amazon Redshift 資料指標：

- 每個工作階段中一次只能開啟一個資料指標。
- 游標必須在交易（ BEGIN ... ）中使用END。
- 所有資料指標累積的結果集大小上限受限於叢集節點類型。若您需要更大的結果集，可將大小調整為 XL 或 8XL 節點組態。

如需詳細資訊，請參閱[游標限制條件](#)。

游標限制條件

擷取資料指標的第一列時，整個結果集會在領導節點上具體化。若結果集無法納入記憶體內，則會視需要寫入磁碟中。為保護領導節點的完整性，Amazon Redshift 會根據叢集的節點類型對所有資料指標結果集的大小強制執行限制條件。

下表說明每個叢集節點類型的結果集大小總計上限。結果集大小上限的單位是 MB。

節點類型	每個叢集的結果集上限 (MB)
DC2 Large 多個節點	192,000

節點類型	每個叢集的結果集上限 (MB)
DC2 Large 單一節點	8,000
DC2 8XL 多個節點	3,200,000
RA3 16XL 多個節點	14,400,000
RA3 4XL 多個節點	3,200,000
RA3 XLPLUS 多個節點	1,000,000
RA3 XLPLUS 單一節點	64,000
RA3 LARGE 多個節點	240,000
RA3 LARGE 單一節點	8,000
Amazon Redshift Serverless	150,000

若要檢視叢集的使用中資料指標組態，請以超級使用者身分查詢 [STV_CURSOR_CONFIGURATION](#) 系統資料表。若要檢視使用中資料指標的狀態，請查詢 [STV_ACTIVE_CURSORS](#) 系統資料表。使用者只能看見自己擁有之資料指標的資料列，但超級使用者可檢視所有資料指標。

使用游標時的效能考量

由於資料指標會在開始傳回結果至用戶端之前，於領導節點上將整個結果集具體化，因此在非常大型的結果集內使用資料指標會對效能產生負面影響。強烈建議您不要在非常大型的結果集內使用資料指標。在某些情況下，例如當您的應用程式使用ODBC連線時，游標可能是唯一可行的解決方案。不過我們建議您盡量使用以下替代方式：

- 使用 [UNLOAD](#) 匯出大型資料表。當您使用時UNLOAD，運算節點會平行運作，以將資料直接傳輸至 Amazon Simple Storage Service 上的資料檔案。如需詳細資訊，請參閱 [在 Amazon Redshift 卸載數據](#)。
- 在用戶端應用程式中設定JDBC擷取大小參數。如果您使用 JDBC 連線且遇到用戶端 out-of-memory 錯誤，您可以透過設定JDBC擷取大小參數，讓用戶端以較小的批次擷取結果集。如需詳細資訊，請參閱 [設定JDBC擷取大小參數](#)。

DECLARE CURSOR 範例

下列範例宣告名為 `lollapalooza` 的游標，`LOLLAPALOOZA` 以選取 Lollapalooza 事件的銷售資訊，然後使用游標從結果集擷取資料列：

```
-- Begin a transaction

begin;

-- Declare a cursor

declare lollapalooza cursor for
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
where sales.eventid = event.eventid
and eventname='Lollapalooza';

-- Fetch the first 5 rows in the cursor lollapalooza:

fetch forward 5 from lollapalooza;

 eventname |          starttime          | costperticket | qtysold
-----+-----+-----+-----
Lollapalooza | 2008-05-01 19:00:00 | 92.00000000 | 3
Lollapalooza | 2008-11-15 15:00:00 | 222.00000000 | 2
Lollapalooza | 2008-04-17 15:00:00 | 239.00000000 | 3
Lollapalooza | 2008-04-17 15:00:00 | 239.00000000 | 4
Lollapalooza | 2008-04-17 15:00:00 | 239.00000000 | 1
(5 rows)

-- Fetch the next row:

fetch next from lollapalooza;

 eventname |          starttime          | costperticket | qtysold
-----+-----+-----+-----
Lollapalooza | 2008-10-06 14:00:00 | 114.00000000 | 2

-- Close the cursor and end the transaction:

close lollapalooza;
commit;
```

下列範例會使用資料表中的所有結果在參考游標上循環：

```
CREATE TABLE tbl_1 (a int, b int);
INSERT INTO tbl_1 values (1, 2),(3, 4);

CREATE OR REPLACE PROCEDURE sp_cursor_loop() AS $$
DECLARE
    target record;
    curs1 cursor for select * from tbl_1;
BEGIN
    OPEN curs1;
    LOOP
        fetch curs1 into target;
        exit when not found;
        RAISE INFO 'a %', target.a;
    END LOOP;
    CLOSE curs1;
END;
$$ LANGUAGE plpgsql;

CALL sp_cursor_loop();

SELECT message
  from svl_stored_proc_messages
  where querytxt like 'CALL sp_cursor_loop()%';

message
-----
  a 1
  a 3
```

DELETE

刪除資料表中的資料列。

Note

單一SQL陳述式的大小上限為 16 MB。

語法

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ... ] ]  
DELETE [ FROM ] { table_name | materialized_view_name }  
  [ { USING } table_name, ... ]  
  [ WHERE condition ]
```

參數

WITH 子句

指定一或多個的選用子句 *common-table-expressions*。請參閱 [WITH 子句](#)。

FROM

FROM 關鍵字是選用的，但指定 USING 子句時除外。陳述式 `delete from event;` 和 `delete event;` 是從 EVENT 資料表中移除所有資料列的等效操作。

Note

若要刪除資料表中的所有資料列，請對資料表執行 [TRUNCATE](#)。TRUNCATE 比更有效率 DELETE，而且不需要 VACUUM 和 ANALYZE。不過，請注意 `TRUNCATE` 遞交執行交易的。

table_name

暫時性或持久性資料表。只有資料表的擁有者或具有 DELETE 權限的使用者，才能從資料表中刪除資料列。

請考慮在大型資料表上使用 TRUNCATE 命令進行快速不合格的刪除操作；請參閱 [TRUNCATE](#)。

Note

從資料表中刪除大量資料列之後：

- 清空資料表以回收儲存空間和重新排序資料列。
- 分析資料表以更新查詢規劃器的統計資訊。

materialized_view_name

具體化視觀表。此DELETE陳述式適用於 所使用的具體化檢視將擷取串流至具體化視觀表。只有具體化視觀表的擁有者或具有具體化視觀表DELETE權限的使用者，才能從中刪除資料列。

您無法在具體化檢視DELETE上執行，以使用未授予使用者IGNORERLS許可的列層級安全 (RLS) 政策進行串流擷取。發生例外狀況：如果執行 的使用者DELETE已IGNORERLS授予，則表示執行成功。如需詳細資訊，請參閱[RLS政策擁有權和管理](#)。

USING table_name , ...

當子WHERE句條件中參考其他資料表時，USING關鍵字用於引入資料表清單。例如，下列陳述式會從滿足 EVENT和 EVENT資料表之聯結條件的SALES資料表中刪除所有列。SALES 資料表必須在FROM清單中明確命名：

```
delete from event using sales where event.eventid=sales.eventid;
```

如果您在 USING 子句中重複目標資料表名稱，DELETE操作會執行自我聯結。您可以在 WHERE 子句中使用子查詢，而不是USING語法，作為撰寫相同查詢的替代方法。

WHERE 條件

此選子句會限制僅刪除符合條件的資料列。例如，條件可以是資料欄上的限制、聯結條件，或根據查詢結果的條件。查詢可以參考 DELETE命令目標以外的資料表。例如：

```
delete from t1
where col1 in(select col2 from t2);
```

如未指定任何條件，則會刪除資料表中的所有資料列。

範例

從CATEGORY資料表中刪除所有資料列：

```
delete from category;
```

從CATEGORY資料表中刪除CATID值介於 0 和 9 之間的資料列：

```
delete from category
where catid between 0 and 9;
```


從LISTING資料表中刪除SELLERID值不存在於SALES資料表中的資料列：

```
delete from listing
where listing.sellerid not in(select sales.sellerid from sales);
```

下列兩個查詢都會根據CATEGORY資料表的聯結和CATID資料欄的其他限制，從EVENT資料表中刪除一行：

```
delete from category
using event
where event.catid=category.catid and category.catid=9;
```

```
delete from category
where catid in
(select category.catid from category, event
where category.catid=event.catid and category.catid=9);
```

下列查詢會刪除 mv_cities 具體化視觀表中的所有資料列。此範例中的具體化視觀表名稱為範例：

```
delete from mv_cities;
```

DESC DATASHARE

顯示使用 ALTER 新增至資料共用中的資料庫物件清單DATASHARE。Amazon Redshift 會顯示資料表、檢視和函數的名稱、資料庫、結構描述及類型。

有關資料共用物件的其他資訊，可以使用系統檢視找到。如需詳細資訊，請參閱 [SVV_DATASHARE_OBJECTS](#) 和 [SVV_DATASHARES](#)。

語法

```
DESC DATASHARE datashare_name [ OF [ ACCOUNT account_id ] NAMESPACE namespace_guid ]
```

參數

datashare_name

資料共用的名稱。

NAMESPACE namespace_guid

指定資料共用所用命名空間的值。當您以取用者叢集管理員DESCDATAHSARE身分執行時，請指定參數NAMESPACE以檢視傳入資料共用。

ACCOUNT account_id

指定資料共用所屬帳戶的值。

使用須知

身為消費者帳戶管理員，當您執行 DESC DATASHARE 以查看 AWS 帳戶中的傳入資料共用時，請指定 NAMESPACE 選項。當您執行 DESC DATASHARE 以查看跨 AWS 帳戶的傳入資料共用時，請指定 ACCOUNT 和 NAMESPACE 選項。

範例

下列範例顯示生產者叢集上有關輸出資料共用的資訊。

```
DESC DATASHARE salesshare;

producer_account |          producer_namespace          | share_type | share_name |
object_type |          object_name          | include_new
-----+-----+-----+-----+
+-----+-----+-----+-----+
123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare |
TABLE          | public.tickit_sales_redshift |
123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | OUTBOUND   | salesshare |
SCHEMA         | public                          | t
```

下列範例顯示取用者叢集上有關輸入資料共用的資訊。

```
DESC DATASHARE salesshare of ACCOUNT '123456789012' NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';

producer_account |          producer_namespace          | share_type | share_name |
object_type |          object_name          | include_new
-----+-----+-----+-----+
+-----+-----+-----+-----+
123456789012    | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND    | salesshare |
table          | public.tickit_sales_redshift |
```

```
123456789012 | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d | INBOUND | salesshare |
schema      | public |
(2 rows)
```

DESC IDENTITY PROVIDER

顯示身分提供者的相關資訊。只有超級使用者可以描述身分提供者。

語法

```
DESC IDENTITY PROVIDER identity_provider_name
```

參數

identity_provider_name

身分提供者的名稱。

範例

下列範例顯示身分提供者的相關資訊。

```
DESC IDENTITY PROVIDER azure_idp;
```

輸出範例。

```
uid | name | type | instanceid | namespace |
      |      |      |            |           |
      |      |      |            |           |
      |      |      |            |           |
      |      |      |            |           |
      |      |      |            |           |
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+
126692 | azure_idp | azure | e40d4bb2-7670-44ae-bfb8-5db013221d73 | aad |
{"issuer":"https://login.microsoftonline.com/e40d4bb2-7670-44ae-bfb8-5db013221d73/
v2.0", "client_id":"871c010f-5e61-4fb1-83ac-98610a7e9110", "client_secret":',
"audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift", "https://
analysis.windows.net/powerbi/connector/AWSRDS"]} | t
(1 row)
```

DETACH MASKING POLICY

從資料欄中分離已附加的動態資料遮罩政策。如需動態資料遮罩的相關資訊，請參閱 [動態資料遮罩](#)。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以分離遮罩政策。

語法

```
DETACH MASKING POLICY policy_name
ON { table_name }
( output_column_names )
FROM { user_name | ROLE role_name | PUBLIC };
```

參數

policy_name

欲分離的遮罩政策名稱。

table_name

要將遮罩政策分離的資料表名稱。

output_column_names

已附加遮罩政策的資料欄名稱。

user_name

已附加遮罩政策的使用者名稱。

您只能在單一DETACHMASKINGPOLICY陳述式PUBLIC中設定 user_name、 role_name 和 之一。

role_name

已附加遮罩政策的角色名稱。

您只能在單一DETACHMASKINGPOLICY陳述式PUBLIC中設定 user_name、 role_name 和 之一。

PUBLIC

顯示政策已附加至資料表中的所有使用者。

您只能在單一DETACHMASKINGPOLICY陳述式PUBLIC中設定 user_name、 role_name 和 之一。

DETACH RLS POLICY

將資料表上的資料列層級安全性政策從一或多個使用者或角色中分離。

超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以分離政策。

語法

```
DETACH RLS POLICY policy_name ON [TABLE] table_name [, ...]  
FROM { user_name | ROLE role_name | PUBLIC } [, ...]
```

參數

`policy_name`

政策的名稱。

ON **【TABLE】** `table_name` **【, ...】**

分離資料列層級安全性政策的資料表或檢視。

FROM { `user_name` | ROLE `role_name` | PUBLIC } **【、...】**

指定政策是否要從一或多個指定的使用者或角色中分離。

使用須知

使用 DETACH RLS POLICY 陳述式時，請遵守下列事項：

- 您可以從關係、使用者、角色或公用項目中分離政策。

範例

下列範例會將資料表上的政策與角色分離。

```
DETACH RLS POLICY policy_concerts ON ticket_category_redshift FROM ROLE analyst, ROLE  
dbadmin;
```

DROP DATABASE

捨棄資料庫。

您無法在交易區塊DROPDATABASE內執行 (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

語法

```
DROP DATABASE database_name
```

參數

database_name

要捨棄之資料庫的名稱。您無法捨棄 dev、padb_harvest、template0、template1 或 sys:internal 資料庫，也無法捨棄目前的資料庫。

若要捨棄外部資料庫，請捨棄外部結構描述。如需詳細資訊，請參閱 [DROP SCHEMA](#)。

DROP DATABASE 用量備註

使用DROPDATABASE陳述式時，請考慮下列事項：

- 一般而言，建議您不要使用 DROPDATABASE陳述式捨棄包含 AWS Data Exchange 資料共用的資料庫。如果您這樣做，AWS 帳戶 具有資料共用存取權的 會失去存取權。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。

下列範例顯示捨棄包含 AWS Data Exchange 資料共用的資料庫時發生錯誤。

```
DROP DATABASE test_db;  
ERROR:  Drop of database test_db that contains ADX-managed datashare(s)  
        requires session variable datashare_break_glass_session_var to be set to value  
        'ce8d280c10ad41'
```

若要允許捨棄資料庫，請設定下列變數，然後再次執行DROPDATABASE陳述式。

```
SET datashare_break_glass_session_var to 'ce8d280c10ad41';
```

```
DROP DATABASE test_db;
```

在此情況下，Amazon Redshift 會產生隨機的一次性值，以設定工作階段變數，以允許包含 AWS Data Exchange 資料共用的DROPDATABASE資料庫。

範例

下列範例會捨棄名為 TICKIT_ 的資料庫TEST：

```
drop database tickit_test;
```

DROP DATASHARE

捨棄資料共用。此命令無法還原。

只有超級使用者或資料共用擁有者可以捨棄資料共用。

所需權限

以下是 DROP 的必要權限DATASHARE：

- 超級使用者
- 具有 DROPDATASHARE權限的使用者
- 資料共用擁有者

語法

```
DROP DATASHARE datashare_name;
```

參數

datashare_name

要捨棄的資料共用名稱。

DROP DATASHARE 用量備註

使用DROPDATASHARE陳述式時，請考慮下列事項：

- 一般而言，建議您不要使用 DROPDATASHARE陳述式捨棄 AWS Data Exchange 資料共用。如果您這樣做，AWS 帳戶 具有資料共用存取權的 會失去存取權。執行此類修改可能會違反 AWS Data Exchange中的資料產品條款。

下列範例顯示捨棄 AWS Data Exchange 資料共用時的錯誤。

```
DROP DATASHARE salesshare;  
ERROR: Drop of ADX-managed datashare salesshare requires session variable  
datashare_break_glass_session_var to be set to value '620c871f890c49'
```

若要允許捨棄 AWS Data Exchange 資料共用，請設定下列變數，然後再次執行 DROPDATASHARE 陳述式。

```
SET datashare_break_glass_session_var to '620c871f890c49';
```

```
DROP DATASHARE salesshare;
```

在此情況下，Amazon Redshift 會產生隨機的一次性值，以設定工作階段變數以允許 DROPDATASHARE AWS Data Exchange 資料共用。

範例

以下範例會捨棄名為 salesshare 的資料共用。

```
DROP DATASHARE salesshare;
```

DROP EXTERNAL VIEW (預覽)

這是適用於 Amazon Redshift 的資料目錄中的發行前版本文件檢視，屬於預覽版本。文件和功能會隨時變更。我們建議僅搭配測試叢集使用此功能，不要在生產環境中使用。如需預覽版條款與條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

您可以在預覽版中建立 Amazon Redshift 叢集，以測試 Amazon Redshift 的新功能。您無法在生產環境中使用這些功能，也無法將預覽叢集移至生產叢集或其他軌道上的叢集。如需了解預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

建立預覽版叢集

1. 登入 AWS Management Console 並在 開啟 Amazon Redshift 主控台 <https://console.aws.amazon.com/redshiftv2/>。

2. 在導覽功能表上，選擇佈建叢集儀表板，然後選擇叢集。AWS 區域 列出您目前 中帳戶的叢集。每個叢集的屬性子集會在清單中分欄顯示。
3. 叢集清單頁面上會顯示一個介紹預覽版的橫幅。選擇建立預覽叢集按鈕以開啟 [建立叢集] 頁面。
4. 輸入叢集的內容。選擇預覽軌道，其中包含您想要測試的功能。建議您輸入叢集名稱，以表示叢集位於預覽軌道上。針對您要測試的功能選擇叢集選項，包括標記為 -preview 的選項。如需有關建立叢集的一般資訊，請參閱《Amazon Redshift 管理指南》中的 [建立叢集](#)。
5. 選擇建立叢集按鈕以建立預覽叢集。

Note

preview_2023 軌跡是最近可用的預覽軌跡。此追蹤僅支援使用RA3節點類型建立叢集。不支援節點類型DC2和任何較舊的節點類型。

6. 預覽叢集可用時，請使用SQL用戶端載入和查詢資料。

Data Catalog 視觀表預覽功能僅適用於以下區域。

- 美國東部 (俄亥俄) (us-east-2)
- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (加利佛尼亞北部) (us-west-1)
- 亞太區域 (東京) (ap-northeast-1)
- 歐洲 (愛爾蘭) (eu-west-1)
- 歐洲 (斯德哥爾摩) (eu-north-1)

您也可以建立預覽工作群組來測試 Data Catalog 視觀表。您無法在生產環境中使用這些功能，也無法將工作群組移至另一個工作群組。如需了解預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。如需如何建立預覽工作群組的指示，請參閱 <https://docs.aws.amazon.com/redshift/latest/mgmt/serverless-workgroup-preview.html>。

從資料庫中捨棄外部檢視。捨棄外部檢視會將其從與檢視相關聯的所有SQL引擎中移除，例如 Amazon Athena 和 Amazon EMR Spark。此命令無法反轉。如需有關 Data Catalog 視觀表的詳細資訊，請參閱 [建立 Data Catalog 視觀表 \(預覽\)](#)。

語法

```
DROP EXTERNAL VIEW schema_name.view_name [ IF EXISTS ]
```

```
{catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |  
external_schema_name.view_name}
```

參數

schema_name.view_name

連接至 AWS Glue 資料庫的結構描述，後面接著檢視的名稱。

IF EXISTS

只有在檢視存在時才捨棄檢視。

```
catalog_name.schema_name.view_name | awsdatacatalog.dbname.view_name |  
external_schema_name.view_name
```

捨棄視觀表時所使用的結構描述標記法。您可以指定使用您建立的 AWS Glue Data Catalog Glue 資料庫，或您建立的外部結構描述。如需詳細資訊[CREATEEXTERNALSCHEMA](#)，請參閱[CREATEDATABASE](#)和。

query_definition

Amazon Redshift 執行以變更檢視的 SQL 查詢定義。

範例

下列範例會捨棄名為 sample_schema.glue_data_catalog_view 的 Data Catalog 視觀表。

```
DROP EXTERNAL VIEW sample_schema.glue_data_catalog_view IF EXISTS
```

DROP FUNCTION

從資料庫移除使用者定義的函數 (UDF)。必須指定函數的簽章 (或引數資料類型的清單)，因為可能有多個同名但簽章不同的函數存在。您無法捨棄 Amazon Redshift 內建函數。

此命令無法還原。

所需權限

以下是 DROP 的必要權限FUNCTION：

- 超級使用者

- 具有 DROPFUNCTION 權限的使用者
- 函數擁有者

語法

```
DROP FUNCTION name
( [arg_name] arg_type  [, ...] )
[ CASCADE | RESTRICT ]
```

參數

name

要移除的函數名稱。

arg_name

輸入引數的名稱。DROP FUNCTION 會忽略引數名稱，因為只需要引數資料類型來判斷函數的身分。

arg_type

輸入引數的資料類型。您可以提供最多包含 32 種資料類型的逗號分隔清單。

CASCADE

此關鍵字指定自動捨棄取決於函數的物件，例如檢視。

若要建立不依賴於函數的檢視，請在檢視定義中包含 WITH NO SCHEMABINDING 子句。如需詳細資訊，請參閱 [CREATE VIEW](#)。

RESTRICT

此關鍵字指定，若有任何物件取決於函數，則不捨棄函數並傳回訊息。這是預設動作。

範例

下列範例會捨棄名為 f_sqrt 的函數：

```
drop function f_sqrt(int);
```

若要移除具有相依性的函數，請使用 CASCADE 選項，如下列範例所示：

```
drop function f_sqrt(int)cascade;
```

DROP GROUP

刪除使用者群組。此命令無法還原。此命令不會刪除群組中的個別使用者。

請參閱 `DROP USER` 以刪除個別使用者。

語法

```
DROP GROUP name
```

參數

`name`

要刪除的使用者群組名稱。

範例

下列範例會刪除 `guests` 使用者群組：

```
DROP GROUP guests;
```

如果群組有物件上的任何權限，則您無法捨棄群組。如果您嘗試捨棄這類群組，則會收到下列錯誤。

```
ERROR: group "guests" can't be dropped because the group has a privilege on some object
```

如果群組具有物件的權限，您必須在捨棄群組之前撤銷權限。若要尋找 `guests` 群組具有權限的物件，請使用下列範例。如需範例中使用的中繼資料檢視的詳細資訊，請參閱 [SVV_RELATION_PRIVILEGES](#)。

```
SELECT DISTINCT namespace_name, relation_name, identity_name, identity_type
FROM svv_relation_privileges
WHERE identity_type='group' AND identity_name='guests';
```

```
+-----+-----+-----+-----+
| namespace_name | relation_name | identity_name | identity_type |
+-----+-----+-----+-----+
| public         | table1        | guests        | group         |
```

```
+-----+-----+-----+-----+
| public      | table2      | guests      | group       |
+-----+-----+-----+-----+
```

下列範例會撤銷 `public` 使用者群組的 `guests` 結構描述中所有資料表的所有權限，然後捨棄群組。

```
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM GROUP guests;
DROP GROUP guests;
```

DROP IDENTITY PROVIDER

刪除身分提供者。此命令無法還原。只有超級使用者可以捨棄身分提供者。

語法

```
DROP IDENTITY PROVIDER identity_provider_name [ CASCADE ]
```

參數

`identity_provider_name`

要刪除的身分提供者名稱。

`CASCADE`

刪除身分提供者時，刪除附加至身分提供者的使用者和角色。

範例

下列範例會刪除 `oauth_provider` 身分提供者。

```
DROP IDENTITY PROVIDER oauth_provider;
```

如果您捨棄身分提供者，某些使用者可能無法登入，或使用設定為使用身分提供者的用戶端工具。

DROP LIBRARY

從資料庫移除自訂 Python 程式庫。只有程式庫擁有者或超級使用者可以捨棄程式庫。

`DROP LIBRARY` 無法在交易區塊內執行（`BEGIN ... END`）。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

此命令無法還原。DROP LIBRARY 命令會立即遞交。如果UDF依賴程式庫的 同時執行，即使 UDF是在交易中執行，也可能會UDF失敗。

如需詳細資訊，請參閱[CREATE LIBRARY](#)。

所需權限

以下是 的必要權限DROPLIBRARY：

- 超級使用者
- 具有 DROPLIBRARY權限的使用者
- 程式庫擁有者

語法

```
DROP LIBRARY library_name
```

參數

library_name

程式庫的名稱。

DROP MASKING POLICY

從所有資料庫中捨棄動態資料遮罩政策。您無法捨棄仍然附加至一或多個資料表的遮罩政策。如需動態資料遮罩的相關資訊，請參閱 [動態資料遮罩](#)。

超級使用者和具有 sys:secadmin 角色的使用者或角色可以捨棄遮罩政策。

語法

```
DROP MASKING POLICY policy_name;
```

參數

policy_name

要捨棄的遮罩政策名稱。

DROP MODEL

從資料庫移除模型。只有模型擁有者或超級使用者可以捨棄模型。

DROP MODEL 也會刪除衍生自此模型的所有相關預測函數、與該模型相關的所有 Amazon Redshift 成品，以及與該模型相關的所有 Amazon S3 資料。當模型仍在 Amazon 中訓練時 SageMaker，DROPMODEL 會取消這些操作。

此命令無法還原。DROP MODEL 命令會立即遞交。

所需的許可

以下是 DROP 的必要許可MODEL：

- 超級使用者
- 具有 DROPMODEL 許可的使用者
- 模型擁有者
- 結構描述擁有者

語法

```
DROP MODEL [ IF EXISTS ] model_name
```

參數

IF EXISTS

此子句會指出，若指定的結構描述已存在，則命令不應進行任何變更，且應傳回結構描述存在的訊息。

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

範例

下列範例會捨棄模型 demo_ml.customer_churn。

```
DROP MODEL demo_ml.customer_churn
```

DROP MATERIALIZED VIEW

移除具體化檢視。

如需具體化檢視的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

語法

```
DROP MATERIALIZED VIEW [ IF EXISTS ] mv_name [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

指定要檢查具名具體化檢視是否存在的子句。如果具體化檢視不存在，則 DROP MATERIALIZED VIEW 命令會傳回錯誤訊息。這個子句在編寫指令碼時相當有用，可用來避免指令碼在您卸除不存在的具體化檢視時失敗。

mv_name

要卸除的具體化檢視名稱。

CASCADE

表示自動捨棄具體化視觀表所依賴之物件的子句，例如其他視觀表。

RESTRICT

如果任何物件依賴於具體化檢視，則表示不捨棄該檢視的子句。此為預設值。

使用須知

只有具體化檢視的擁有者可以在該檢視上使用 DROP MATERIALIZED VIEW。超級使用者或特別授予 DROP 權限的使用者可能是此項目的例外狀況。

當您為具體化視觀表撰寫捨棄陳述式，並存在具有相符名稱的視觀表時，會導致錯誤，指示您使用 DROP VIEW。即使在您使用 DROP MATERIALIZED VIEW IF EXISTS 的情況下，也會發生錯誤。

範例

以下範例會卸除 `tickets_mv` 具體化檢視。


```
DROP MATERIALIZED VIEW tickets_mv;
```

DROP PROCEDURE

捨棄程序。若要捨棄程序，程序名稱和輸入引數資料類型 (簽章) 都是必要的。或者，您可以包含完整的引數資料類型，包括OUT引數。若要尋找程序的簽章，請使用 [SHOW PROCEDURE](#) 命令。如需程序簽章的相關資訊，請參閱 [PG_PROC_INFO](#)。

所需權限

以下是 DROP 的必要權限PROCEDURE：

- 超級使用者
- 具有 DROPPROCEDURE權限的使用者
- 程序擁有者

語法

```
DROP PROCEDURE sp_name ( [ [ argname ] [ argmode ] argtype [, ...] ] )
```

參數

sp_name

要移除的程序的名称。

argname

輸入引數的名称。DROP PROCEDURE 會忽略引數名称，因為只需要引數資料類型來判斷程序的身分。

argmode

引數的模式，可以是 IN、OUT或 INOUT。OUT 引數是選用的，因為它們不會用來識別預存程序。

argtype

輸入引數的資料類型。如需支援的資料類型清單，請參閱 [資料類型](#)。

範例

下列範例捨棄名為 `quarterly_revenue` 的預存程序。

```
DROP PROCEDURE quarterly_revenue(volume INOUT bigint, at_price IN numeric,result OUT int);
```

DROP RLS POLICY

捨棄所有資料庫中所有資料表的資料列層級安全性政策。

超級使用者和具有 `sys:secadmin` 角色的使用者或角色可以捨棄政策。

語法

```
DROP RLS POLICY [ IF EXISTS ] policy_name [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

指出指定政策是否已存在的子句。

policy_name

政策的名稱。

CASCADE

指示在捨棄政策之前，自動將政策從所有附加的資料表分離的子句。

RESTRICT

指出政策在附加至某些資料表時不要捨棄政策。此為預設值。

範例

下列範例會捨棄資料列層級安全性政策。

```
DROP RLS POLICY policy_concerts;
```

DROP ROLE

從資料庫中移除角色。只有建立角色的角色擁有者、具有 WITHADMIN 選項的使用者或超級使用者才能捨棄角色。

如果角色已授予使用者或相依於此角色的其他角色，則您無法捨棄角色。

所需權限

以下是 DROP 的必要權限ROLE：

- 超級使用者
- 角色擁有者是建立角色的使用者，或是已授予具有 WITH ADMIN OPTION 權限角色的使用者。

語法

```
DROP ROLE role_name [ FORCE | RESTRICT ]
```

參數

role_name

角色的名稱。

[FORCE | RESTRICT]

預設設定為 RESTRICT。當您嘗試捨棄繼承另一個角色的角色時，Amazon Redshift 會擲回錯誤。如果存在任何角色指派，請使用 FORCE 移除所有角色指派。

範例

下列範例會捨棄 `sample_role` 角色。

```
DROP ROLE sample_role FORCE;
```

下列範例會嘗試捨棄已授予具有預設RESTRICT選項的使用者的角色 `sample_role1`。

```
CREATE ROLE sample_role1;  
GRANT sample_role1 TO user1;
```

```
DROP ROLE sample_role1;  
ERROR: cannot drop this role since it has been granted on a user
```

若要成功捨棄已授予使用者的 `sample_role1`，請使用 `FORCE` 選項。

```
DROP ROLE sample_role1 FORCE;
```

下列範例會嘗試捨棄具有其他角色的 `sample_role2` 角色，這些角色依賴於具有預設 `RESTRICT` 選項的角色。

```
CREATE ROLE sample_role1;  
CREATE ROLE sample_role2;  
GRANT sample_role1 TO sample_role2;  
DROP ROLE sample_role2;  
ERROR: cannot drop this role since it depends on another role
```

若要成功捨棄具有其他角色的 `sample_role2`，請使用 `FORCE` 選項。

```
DROP ROLE sample_role2 FORCE;
```

DROP SCHEMA

刪除結構描述。針對外部結構描述，您也可以捨棄與結構描述相關聯的外部資料庫。此命令無法還原。

所需權限

以下是 `DROP` 的必要權限 `SCHEMA`：

- 超級使用者
- 結構描述擁有者
- 具有 `DROPSHEMA` 權限的使用者

語法

```
DROP SCHEMA [ IF EXISTS ] name [, ...]  
[ DROP EXTERNAL DATABASE ]  
[ CASCADE | RESTRICT ]
```

參數

IF EXISTS

此子句會指出，若指定的結構描述不存在，則命令不應進行任何變更，且應傳回結構描述不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在指令碼時很有用，因此如果對不存在的結構描述DROPSHEMA執行，指令碼不會失敗。

name

要捨棄的結構描述名稱。您可以指定多個結構描述名稱，以逗號分隔。

DROP EXTERNAL DATABASE

此子句指出是否捨棄外部結構描述，如果存在則捨棄與外部結構描述相關聯的外部資料庫。如果外部資料庫不存在，此命令會傳回訊息，說明外部資料庫不存在。如果已捨棄多個外部資料庫，則會捨棄與指定結構描述相關聯的所有資料庫。

如果外部資料庫包含相依物件，例如資料表，也請包含捨棄相依物件CASCADE的選項。

捨棄外部資料庫時，也會捨棄資料庫中與資料庫相關聯的任何其他外部結構描述。也會捨棄在使用資料庫的其他外部結構描述中所定義的資料表。

DROP EXTERNAL DATABASE 不支援存放在HIVE中繼存放區的外部資料庫。

CASCADE

此關鍵字指出自動捨棄結構描述中的所有物件。如果指定 DROP EXTERNAL DATABASE，則外部資料庫中的所有物件也會捨棄。

RESTRICT

此關鍵字指出，若結構描述或外部資料庫含任何物件則不捨棄。這是預設動作。

範例

下列範例會刪除名為 S_ 的結構描述SALES。此範例使用 RESTRICT作為安全機制，因此如果結構描述包含任何物件，就不會將其刪除。在此情況下，您需要先刪除結構描述物件，再刪除結構描述。

```
drop schema s_sales restrict;
```

下列範例會刪除名為 S_SALES 的結構描述，以及所有依賴該結構描述的物件。

```
drop schema s_sales cascade;
```

下列範例會在 S_SALES 結構描述存在時捨棄它，或者如果它不存在，則不會執行任何動作，如果它沒有，則傳回訊息。

```
drop schema if exists s_sales;
```

下列範例會刪除名為 S_SPECTRUM 的外部結構描述，以及與其相關聯的外部資料庫。此範例使用 `RESTRICT` 因此如果結構描述和資料庫包含任何物件，就不會將其刪除。在此情況下，您會需要先刪除相依物件，再刪除結構描述和資料庫。

```
drop schema s_spectrum drop external database restrict;
```

以下範例會刪除名為多個結構描述和與其相關聯的外部資料庫，以及任何相依物件。

```
drop schema s_sales, s_profit, s_revenue drop external database cascade;
```

DROP TABLE

從資料庫移除資料表。

如果您嘗試清空資料列資料表，而不移除資料表，請使用 `DELETE` 或 `TRUNCATE` 命令。

`DROP TABLE` 移除目標資料表上存在的限制。可以使用單一 `DROPTABLE` 命令移除多個資料表。

`DROP TABLE` 無法在交易（`BEGIN ...`）內執行具有外部資料表的 `END`。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

若要尋找將 `DROP` 權限授予群組的範例，請參閱 [GRANT 範例](#)。

所需權限

以下是 `DROP` 的必要權限 `TABLE`：

- 超級使用者
- 具有 `DROPTABLE` 權限的使用者
- 具有結構描述上 `USAGE` 權限的資料表擁有者

語法

```
DROP TABLE [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

此子句會指出，若指定的資料表不存在，則命令不應進行任何變更，且應傳回資料表不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在指令碼時很有用，因此如果對不存在的資料表DROPTABLE執行，指令碼不會失敗。

name

要捨棄的資料表名稱。

CASCADE

此子句指出，自動捨棄取決於資料表的物件，例如檢視。

若要建立不依賴於其他資料庫物件的檢視，例如檢視和資料表，請在檢視定義中包含 WITH NO SCHEMABINDING子句。如需詳細資訊，請參閱[CREATE VIEW](#)。

RESTRICT

此子句指出，若有任何物件相依於資料表，則不捨棄資料表。這是預設動作。

範例

丟棄無相依性的資料表

下列範例會建立和捨棄名為的資料表FEEDBACK，該資料表沒有相依性：

```
create table feedback(a int);  
  
drop table feedback;
```

若資料表包含的資料欄為檢視或其他資料表所參考，Amazon Redshift 會顯示以下訊息。

```
Invalid operation: cannot drop table feedback because other objects depend on it
```

同步丟棄兩個資料表

下列命令集會建立FEEDBACK資料表和BUYERS資料表，然後使用單一命令捨棄兩個資料表：

```
create table feedback(a int);

create table buyers(a int);

drop table feedback, buyers;
```

丟棄具有相依性的資料表

下列步驟說明如何捨棄FEEDBACK使用CASCADE交換機呼叫的資料表。

首先，FEEDBACK使用 CREATETABLE命令建立名為 的簡單資料表：

```
create table feedback(a int);
```

接下來，使用 CREATEVIEW命令來建立名為 FEEDBACK_VIEW 的檢視，該檢視依賴資料表 FEEDBACK：

```
create view feedback_view as select * from feedback;
```

下列範例會捨棄資料表FEEDBACK，也會捨棄檢視 FEEDBACK_VIEW，因為 FEEDBACK_VIEW 取決於資料表 FEEDBACK：

```
drop table feedback cascade;
```

檢視資料表的相依性

若要傳回資料表的相依性，請使用下列範例。Replace (取代) *my_schema* 以及 *my_table* 使用您自己的結構描述和資料表。

```
SELECT dependent_ns.nspname as dependent_schema
, dependent_view.relname as dependent_view
, source_ns.nspname as source_schema
, source_table.relname as source_table
, pg_attribute.attname as column_name
FROM pg_depend
JOIN pg_rewrite ON pg_depend.objid = pg_rewrite.oid
JOIN pg_class as dependent_view ON pg_rewrite.ev_class = dependent_view.oid
JOIN pg_class as source_table ON pg_depend.refobjid = source_table.oid
```



```

JOIN pg_attribute ON pg_depend.refobjid = pg_attribute.attrelid
      AND pg_depend.refobjsubid = pg_attribute.attnum
JOIN pg_namespace dependent_ns ON dependent_ns.oid = dependent_view.relnamespace
JOIN pg_namespace source_ns ON source_ns.oid = source_table.relnamespace
WHERE
source_ns.nspname = 'my_schema'
AND source_table.relname = 'my_table'
AND pg_attribute.attnum > 0
ORDER BY 1,2
LIMIT 10;

```

若要捨棄 *my_table* 及其相依性，請使用下列範例。此範例也會傳回已捨棄資料表的所有相依性。

```

DROP TABLE my_table CASCADE;

SELECT dependent_ns.nspname as dependent_schema
, dependent_view.relname as dependent_view
, source_ns.nspname as source_schema
, source_table.relname as source_table
, pg_attribute.attname as column_name
FROM pg_depend
JOIN pg_rewrite ON pg_depend.objid = pg_rewrite.oid
JOIN pg_class as dependent_view ON pg_rewrite.ev_class = dependent_view.oid
JOIN pg_class as source_table ON pg_depend.refobjid = source_table.oid
JOIN pg_attribute ON pg_depend.refobjid = pg_attribute.attrelid
      AND pg_depend.refobjsubid = pg_attribute.attnum
JOIN pg_namespace dependent_ns ON dependent_ns.oid = dependent_view.relnamespace
JOIN pg_namespace source_ns ON source_ns.oid = source_table.relnamespace
WHERE
source_ns.nspname = 'my_schema'
AND source_table.relname = 'my_table'
AND pg_attribute.attnum > 0
ORDER BY 1,2
LIMIT 10;

```

```

+-----+-----+-----+-----+-----+
| dependent_schema | dependent_view | source_schema | source_table | column_name |
+-----+-----+-----+-----+-----+

```

使用 IF 捨棄資料表 EXISTS

下列範例會在 FEEDBACK 資料表存在時捨棄資料表，或者如果資料表不存在，則不會執行任何動作，如果資料表不存在，則傳回訊息：

```
drop table if exists feedback;
```

DROP USER

從資料庫捨棄使用者。可以使用單一DROPUSER命令捨棄多個使用者。您必須是資料庫超級使用者，或具有執行此命令的DROPUSER許可。

語法

```
DROP USER [ IF EXISTS ] name [, ... ]
```

參數

IF EXISTS

此子句會指出，若指定的使用者不存在，則命令不應進行任何變更，且應傳回使用者不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在指令碼時很有用，因此如果對不存在的使用者DROPUSER執行，指令碼不會失敗。

name

要移除的使用者名稱。您可以指定多個使用者，並以逗號分隔每個使用者名稱。

使用須知

您不能捨棄名為 rdsdb 的使用者或通常命名為 awsuser 或 admin 的資料庫管理員使用者。

若使用者擁有任何資料庫物件，像是結構描述、資料庫、資料表或檢視，或是使用者具有資料庫、資料表、資料欄或群組的任何權限，則無法捨棄使用者。如果您嘗試捨棄這類使用者，則會收到下列錯誤之一。

```
ERROR: user "username" can't be dropped because the user owns some object [SQL State=55006]
```

```
ERROR: user "username" can't be dropped because the user has a privilege on some object [SQL State=55006]
```

如需如何尋找資料庫使用者擁有之物件的詳細指示，請參閱知識中心內的[如何解決 Amazon Redshift 中的「使用者無法捨棄」錯誤？](#)。

Note

Amazon Redshift 在捨棄使用者之前，只會檢查目前資料庫。DROP USER 如果使用者擁有資料庫物件，或對另一個資料庫中的物件具有任何權限，則不會傳回錯誤。若您捨棄的使用者擁有其他資料庫中的物件，則這些物件的擁有者會變更為「不明」。

若使用者擁有物件，請先捨棄物件或將其所有權變更為其他使用者，再捨棄原始使用者。如果使用者具有物件的權限，請先撤銷權限，再捨棄使用者。以下範例顯示在捨棄使用者前，先捨棄物件、變更所有權，以及撤銷權限。

```
drop database dwdatabase;
alter schema dw owner to dwadmin;
revoke all on table dwtable from dwuser;
drop user dwuser;
```

範例

以下範例會捨棄名為 paulo 的使用者：

```
drop user paulo;
```

以下範例會捨棄兩個使用者，paulo 和 martha：

```
drop user paulo, martha;
```

以下範例會捨棄使用者 paulo (如存在的話)，或不執行任何動作，並於該使用者帳戶不存在時傳回訊息：

```
drop user if exists paulo;
```

DROP VIEW

從資料庫移除檢視。可以使用單一DROPVIEW命令捨棄多個檢視。此命令無法還原。

所需權限

以下是 DROP 的必要權限VIEW：

- 超級使用者
- 具有 DROPVIEW 權限的使用者
- 檢視擁有者

語法

```
DROP VIEW [ IF EXISTS ] name [, ... ] [ CASCADE | RESTRICT ]
```

參數

IF EXISTS

此子句會指出，若指定的檢視不存在，則命令不應進行任何變更，且應傳回檢視不存在的訊息，而不是在發生錯誤的情況下終止。

此子句在指令碼時很有用，因此如果對不存在的檢視 DROPVIEW 執行，指令碼不會失敗。

name

要移除的檢視名稱。

CASCADE

此子句指出，自動捨棄取決於檢視的物件，例如其他檢視。

若要建立不依賴於其他資料庫物件的檢視，例如檢視和資料表，請在檢視定義中包含 WITH NO SCHEMABINDING 子句。如需詳細資訊，請參閱 [CREATE VIEW](#)。

請注意，如果您包含 CASCADE，且資料庫物件的捨棄計數執行至十個或更多，則資料庫用戶端可能不會列出摘要結果中的所有捨棄物件。這通常是因為 SQL 用戶端工具對傳回的結果具有預設限制。

RESTRICT

此子句指出，若有任何物件相依於檢視，則不捨棄檢視。這是預設動作。

範例

下列範例會捨棄名為 event 的檢視：

```
drop view event;
```

若要移除具有相依性的檢視，請使用 CASCADE 選項。例如，假設我們從名為 `event` 的資料表開始 EVENT。然後，我們會使用 CREATEVIEW 命令建立 EVENT 資料表的 `Eventview` 檢視，如下列範例所示：

```
create view eventview as
select dateid, eventname, catid
from event where catid = 1;
```

現在我們建立另一個檢視，名為 `myeventview`，這是根據第一個檢視 `eventview` 所建立：

```
create view myeventview as
select eventname, catid
from eventview where eventname <> ' ';
```

此時已建立兩個檢視：`eventview` 和 `myeventview`。

`myeventview` 檢視是子檢視，`eventview` 是其父檢視。

若要刪除 `eventview` 檢視，明顯可使用下列命令：

```
drop view eventview;
```

請注意，如果您在此情況下執行此命令，將會收到下列錯誤：

```
drop view eventview;
ERROR: can't drop view eventview because other objects depend on it
HINT: Use DROP ... CASCADE to drop the dependent objects too.
```

若要解決此情況，請執行下列命令 (如錯誤訊息中所建議)：

```
drop view eventview cascade;
```

`eventview` 和 `myeventview` 現在都已成功捨棄。

下列範例會捨棄 `eventview` 檢視 (如存在的話)，或不執行任何動作，並於檢視不存在時傳回訊息：

```
drop view if exists eventview;
```

END

遞交目前交易。執行與COMMIT命令完全相同的函數。

如需詳細文件，請參閱 [COMMIT](#)。

語法

```
END [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。

TRANSACTION

選用關鍵字；WORK以及 TRANSACTION是同義字。

範例

以下範例全都會結束交易區塊並遞交交易：

```
end;
```

```
end work;
```

```
end transaction;
```

在這些命令之後，Amazon Redshift 會結束交易區塊並遞交變更。

EXECUTE

執行先前預備的陳述式。

語法

```
EXECUTE plan_name [ (parameter [, ...]) ]
```

參數

plan_name

要執行的預備陳述式名稱。

parameter

預備陳述式之參數的實際值。這必須是在建立準備陳述式的PREPARE命令中產生與為此參數位置指定的資料類型相容之類型值的表達式。

使用須知

EXECUTE 用於執行先前準備的陳述式。由於準備的陳述式僅在工作階段的持續時間內存在，因此準備的陳述式必須由目前工作階段中較早執行的PREPARE陳述式建立。

如果先前的PREPARE陳述式指定了一些參數，則必須將一組相容的參數傳遞給陳述EXECUTE式，否則 Amazon Redshift 會傳回錯誤。與函數不同的是，預備陳述式不會根據指定的參數類型或數目過載；預備陳述式的名稱必須在資料庫工作階段內是唯一的。

為準備好的陳述式發出EXECUTE命令時，Amazon Redshift 可能會選擇性地修改查詢執行計畫（以根據指定的參數值改善效能），然後再執行準備好的陳述式。此外，每次新執行準備好的陳述式時，Amazon Redshift 可能會根據隨EXECUTE陳述式指定的不同參數值，再次修改查詢執行計畫。若要檢查 Amazon Redshift 為任何指定EXECUTE陳述式選擇的查詢執行計畫，請使用 [EXPLAIN](#)命令。

如需建立及使用預備陳述式的範例和詳細資訊，請參閱 [PREPARE](#)。

另請參閱

[DEALLOCATE](#), [PREPARE](#)

EXPLAIN

顯示查詢陳述式的執行計畫，但不執行查詢。如需有關查詢分析工作流程的資訊，請參閱 [查詢分析工作流程](#)。

語法

```
EXPLAIN [ VERBOSE ] query
```

參數

VERBOSE

顯示完整查詢計畫，不只是摘要而已。

query

要說明的查詢陳述式。查詢可以是 SELECT、INSERT、CREATETABLEASUPDATE、或 DELETE 陳述式。

使用須知

EXPLAIN 效能有時會受到建立暫存資料表所需的時間所影響。例如，使用常見子表達最佳化的查詢需要建立和分析暫存資料表，才能傳回 EXPLAIN 輸出。查詢計畫取決於臨時資料表的結構描述和統計資訊。因此，這類查詢的 EXPLAIN 命令執行時間可能比預期更長。

EXPLAIN 您只能將用於下列命令：

- SELECT
- SELECT INTO
- CREATE TABLE AS
- INSERT
- UPDATE
- DELETE

如果您將其用於其他 SQL 命令，例如資料定義語言（DDL）或資料庫操作，則 EXPLAIN 命令將會失敗。

Amazon Redshift 會使用 EXPLAIN 輸出相對單位成本來選擇查詢計畫。Amazon Redshift 會比較各種資源估算值的大小來決定計畫。

查詢計劃和執行步驟

特定 Amazon Redshift 查詢陳述式的執行計畫會將查詢的執行和計算細分成一系列分散的步驟和資料表操作，這些最後會產生查詢的最終結果集。如需查詢計劃的資訊，請參閱 [查詢處理](#)。

下表摘要說明 Amazon Redshift 在開發使用者提交執行的任何查詢執行計畫時可使用的步驟。

EXPLAIN 運算子	查詢執行步驟	描述
-------------	--------	----

SCAN:

Sequential Scan	scan	Amazon Redshift 關聯式掃描或資料庫掃描運算子或步驟。從頭到尾依序掃描整個資料表；如果使用子WHERE句指定，也會評估每一列（篩選條件）的查詢限制。也用於執行 INSERT、UPDATE 和 DELETE 陳述式。
-----------------	------	--

JOINS：Amazon Redshift 會根據要聯結之資料表的實體設計、聯結所需的資料位置，以及查詢本身的特定屬性，使用不同的聯結運算子。子查詢掃描 -- 子查詢掃描和附加會用來執行 UNION 查詢。

巢狀迴路	nloop	最差聯結；主要用於交叉聯結 (笛卡兒乘積；無聯結條件) 和一些不相等聯結中。
雜湊聯結	hjoin	同樣用於內部聯結和左右外部聯結，且通常會較巢狀迴路聯結更快速。雜湊聯結會讀取外部資料表、雜湊聯結資料欄，並在內部雜湊表中尋找相符項目。步驟可溢寫至磁碟。(hjoin 的內部輸出可以是磁碟型的雜湊步驟。)
合併聯結	mjoin	同樣用於內部聯結和外部聯結 (針對在聯結資料欄上進行分佈和排序的聯結資料表)。通常是速度最快的 Amazon Redshift 聯結演算法 (不納入其他成本考量的情況下)。

AGGREGATION：用於涉及彙總函數和 GROUP BY 操作之查詢的運算子和步驟。

Aggregate	aggr	純量彙整函數的運算子/步驟。
HashAggregate	aggr	分組彙整函數的運算子/步驟。可從磁碟透過溢寫至磁碟的雜湊資料表操作。
GroupAggregate	aggr	有時會針對分組彙整查詢選擇的運算子，在 force_hash_grouping 設定的 Amazon Redshift 組態設定為關閉的情況下。

SORT：查詢必須排序或合併結果集時使用的運算子和步驟。

EXPLAIN 運算子	查詢執行步驟	描述
Sort	sort	排序會執行 ORDER BY 子句指定的排序，以及 UNIONs和 聯結等其他操作。可從磁碟操作。
合併	merge	根據衍生自平行執行之操作的中繼排序結果，產生查詢的最終排序結果。
EXCEPT、INTERSECT和 UNION操作：		
SetOp 除了 【Distinct】	hjoin	用於EXCEPT查詢。根據輸入雜湊可以是磁碟型的事實，可從磁碟操作。
Hash Intersect [Distinct]	hjoin	用於INTERSECT查詢。根據輸入雜湊可以是磁碟型的事實，可從磁碟操作。
Append [All Distinct]	save	附加與子查詢掃描搭配使用，以實作 UNION 和 UNION ALL 查詢。可根據 "save" 從磁碟操作。
其他：		
Hash	hash	用於內部聯結和左右外部聯結 (對雜湊聯結提供輸入)。Hash 運算子會為聯結的內部資料表建立雜湊資料表 (內部資料表是在兩個資料表的聯結中，要查看其中是否有相符項目的資料表，通常是兩個資料表中較小的)。
限制	limit	評估 LIMIT 子句。
Materialize	save	具體化資料列以輸入至巢狀迴路聯結和一些合併聯結。可從磁碟操作。
--	parse	在載入時用來剖析文字輸入資料。
--	project	用來重新安排資料欄和表達式，也就是專案資料。
結果	--	執行未牽涉任何資料表存取的純量函數。
--	return	將資料列傳回至領導者或用戶端。

EXPLAIN 運算子	查詢執行步驟	描述
Subplan	--	用於特定子查詢。
唯一	unique	消除來自 SELECTDISTINCT和 UNION查詢的重複項目。
視窗	window	運算彙整和排名視窗函數。可從磁碟操作。

網路操作：

Network (Broadcast)	bcast	Broadcast 也是 Join Explain 運算子和步驟的屬性。
Network (Distribute)	dist	將資料列分佈至運算節點供資料倉儲叢集進行平行處理。
Network (Send to Leader)	return	將結果傳回領導者供進一步處理。

DML 操作（修改資料的運算子）：

Insert (using Result)	insert	插入資料。
Delete (Scan + Filter)	刪除	刪除資料。可從磁碟操作。
Update (Scan + Filter)	delete, insert	實作為 delete 和 Insert。

使用 EXPLAIN for RLS

如果查詢包含受資料列層級安全（RLS）政策約束的資料表，EXPLAIN會顯示特殊RLS SecureScan節點。Amazon Redshift 也會將相同的節點類型記錄到 STL_EXPLAIN 系統資料表。EXPLAIN 不會顯示適用於 dim_tbl 的RLS述詞。RLS SecureScan 節點類型可做為執行計畫包含目前使用者看不到之其他操作的指標。

下列範例說明RLS SecureScan 節點。

```
EXPLAIN
SELECT D.cint
FROM fact_tbl F INNER JOIN dim_tbl D ON F.k_dim = D.k
WHERE F.k_dim / 10 > 0;
```

QUERY PLAN

```

-----
XN Hash Join DS_DIST_ALL_NONE (cost=0.08..0.25 rows=1 width=4)
  Hash Cond: ("outer".k_dim = "inner"."k")
  -> *XN* *RLS SecureScan f (cost=0.00..0.14 rows=2 width=4)*
      Filter: ((k_dim / 10) > 0)
  -> XN Hash (cost=0.07..0.07 rows=2 width=8)
      -> XN Seq Scan on dim_tbl d (cost=0.00..0.07 rows=2 width=8)
          Filter: (("k" / 10) > 0)

```

若要完整調查受約束的查詢計劃RLS，Amazon Redshift 會提供EXPLAINRLS系統許可。已授予此許可的使用者可以檢查完整的查詢計劃，其中也包含述RLS詞。

下列範例說明RLS SecureScan 節點下方的額外 Seq 掃描，也包含RLS政策述詞（`k_dim > 1`）。

```

EXPLAIN SELECT D.cint
FROM fact_tbl F INNER JOIN dim_tbl D ON F.k_dim = D.k
WHERE F.k_dim / 10 > 0;

```

QUERY PLAN

```

-----
XN Hash Join DS_DIST_ALL_NONE (cost=0.08..0.25 rows=1 width=4)
  Hash Cond: ("outer".k_dim = "inner"."k")
  *-> XN RLS SecureScan f (cost=0.00..0.14 rows=2 width=4)
      Filter: ((k_dim / 10) > 0)*
      -> *XN* *Seq Scan on fact_tbl rls_table (cost=0.00..0.06 rows=5 width=8)
          Filter: (k_dim > 1)*
  -> XN Hash (cost=0.07..0.07 rows=2 width=8)
      -> XN Seq Scan on dim_tbl d (cost=0.00..0.07 rows=2 width=8)
          Filter: (("k" / 10) > 0)

```

將EXPLAINRLS許可授予使用者時，Amazon Redshift 會記錄完整的查詢計畫，包括 STL_EXPLAIN 系統資料表中的述RLS詞。未授予此許可時執行的查詢，將在沒有RLS內部的情況下進行記錄。授予或移除EXPLAINRLS許可不會變更 Amazon Redshift 針對先前查詢記錄到 STL_EXPLAIN 的內容。

AWS Lake Formation-RLS 受保護的 Redshift 關係

下列範例說明 LF SecureScan 節點，您可以用來檢視 Lake Formation RLS關係。

```


EXPLAIN
SELECT *
FROM lf_db.public.t_share
WHERE a > 1;

```

QUERY PLAN

```
-----
XN LF SecureScan t_share (cost=0.00..0.02 rows=2 width=11)
(2 rows)
```

範例

 Note

在這些範例中，範例輸出可能因 Amazon Redshift 組態而有所不同。

下列範例會傳回 VENUENAME 從 EVENT 和 VENUE 資料表中選取 EVENTID、VENUEID、EVENTNAME 和 之查詢的查詢計畫：

```
explain
select eventid, eventname, event.venueid, venueid, venueid
from event, venue
where event.venueid = venue.venueid;
```

QUERY PLAN

```
-----
XN Hash Join DS_DIST_OUTER (cost=2.52..58653620.93 rows=8712 width=43)
Hash Cond: ("outer".venueid = "inner".venueid)
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=23)
-> XN Hash (cost=2.02..2.02 rows=202 width=22)
-> XN Seq Scan on venue (cost=0.00..2.02 rows=202 width=22)
(5 rows)
```

以下範例會傳回相同查詢的查詢計畫，但包含詳細輸出：

```
explain verbose
select eventid, eventname, event.venueid, venueid, venueid
from event, venue
where event.venueid = venue.venueid;
```

QUERY PLAN

```
-----
{HASHJOIN
:startup_cost 2.52
```

```

:total_cost 58653620.93
:plan_rows 8712
:plan_width 43
:best_pathkeys <>
:dist_info DS_DIST_OUTER
:dist_info.dist_keys (
TARGETENTRY
{
VAR
:varno 2
:varattno 1
...

XN Hash Join DS_DIST_OUTER (cost=2.52..58653620.93 rows=8712 width=43)
Hash Cond: ("outer".venueid = "inner".venueid)
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798 width=23)
-> XN Hash (cost=2.02..2.02 rows=202 width=22)
-> XN Seq Scan on venue (cost=0.00..2.02 rows=202 width=22)
(519 rows)

```

下列範例會傳回 CREATE TABLE AS (CTAS) 陳述式的查詢計劃：

```

explain create table venue_nonulls as
select * from venue
where venueseats is not null;

QUERY PLAN
-----
XN Seq Scan on venue (cost=0.00..2.02 rows=187 width=45)
Filter: (venueseats IS NOT NULL)
(2 rows)

```

FETCH

使用資料指標擷取資料列。如需宣告資料指標的相關資訊，請參閱 [DECLARE](#)。

FETCH 根據游標內的目前位置擷取資料列。建立資料指標時，它會定位在第一列前面。在之後 FETCH，游標會位於擷取的最後一列。如果從可用資料列的結尾 FETCH 執行，例如遵循 FETCH ALL，則游標會保留在最後一個資料列之後。

FORWARD 0 在不移動游標的情況下擷取目前的資料列；也就是說，它會擷取最近擷取的資料列。如果資料指標定位在第一列前面或最後一列後面，則不會傳回任何資料列。

擷取資料指標的第一列時，整個結果集會在領導節點上、記憶體中或磁碟上具體化 (如有需要)。由於在大型結果集內使用資料指標可能會對效能造成負面影響，因此建議您盡量使用替代方式。如需詳細資訊，請參閱[使用游標時的效能考量](#)。

如需詳細資訊，請參閱 [DECLARE](#)、[CLOSE](#)。

語法

```
FETCH [ NEXT | ALL | {FORWARD [ count | ALL ] } ] FROM cursor
```

參數

NEXT

擷取下一列。此為預設值。

ALL

擷取所有剩餘的資料列。(與 FORWARD 相同ALL。) ALL 不支援單節點叢集。

FORWARD 【計數 | ALL】

擷取下一個 *count* 資料列，或所有剩餘的資料列。FORWARD 0 會擷取目前資料列。若是單一節點叢集，*count* 的最大值為 1000。FORWARD ALL 不支援單節點叢集。

cursor

新資料指標的名稱。

FETCH 範例

下列範例宣告名為 `LOLLAPALOOZA` 的游標，LOLLAPALOOZA 以選取 Lollapalooza 事件的銷售資訊，然後使用游標從結果集擷取資料列：

```
-- Begin a transaction

begin;

-- Declare a cursor

declare lollapalooza cursor for
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
```

```

where sales.eventid = event.eventid
and eventname='Lollapalooza';

-- Fetch the first 5 rows in the cursor lollapalooza:

fetch forward 5 from lollapalooza;

  eventname |          starttime          | costperticket | qtysold
-----+-----+-----+-----
Lollapalooza | 2008-05-01 19:00:00 | 92.00000000 | 3
Lollapalooza | 2008-11-15 15:00:00 | 222.00000000 | 2
Lollapalooza | 2008-04-17 15:00:00 | 239.00000000 | 3
Lollapalooza | 2008-04-17 15:00:00 | 239.00000000 | 4
Lollapalooza | 2008-04-17 15:00:00 | 239.00000000 | 1
(5 rows)

-- Fetch the next row:

fetch next from lollapalooza;

  eventname |          starttime          | costperticket | qtysold
-----+-----+-----+-----
Lollapalooza | 2008-10-06 14:00:00 | 114.00000000 | 2

-- Close the cursor and end the transaction:

close lollapalooza;
commit;

```

GRANT

定義使用者或角色的存取權限。

許可包括存取選項，例如能夠讀取資料表和檢視中的資料、寫入資料、建立資料表及捨棄資料表。使用此命令可提供資料表、資料庫、結構描述、函數、程序、語言或資料欄的特定許可。若要撤銷資料庫物件的許可，請使用 [REVOKE](#) 命令。

權限也包括下列資料共用生產者存取選項：

- 授予取用者命名空間和帳戶的資料共用存取權。
- 透過在資料共用中新增或移除物件，授予變更資料共用的權限。
- 透過從資料共用中新增或移除取用者命名空間，授予資料共用的權限。

資料共用取用者存取選項如下：

- 授予使用者對從資料共用建立的資料庫，或指向此類資料庫的外部結構描述的完全存取權。
- 授予使用者從資料共用建立之資料庫的物件層級權限，就像您對本機資料庫物件一樣。若要授予此層級的許可，您必須在從資料共用建立資料庫時使用 WITHPERMISSIONS 子句。如需詳細資訊，請參閱 [CREATE DATABASE](#)。

如需資料共用權限的詳細資訊，請參閱 [叢集內和叢集之間的資料共用](#)。

您也可以授予角色來管理資料庫許可，並控制使用者可以對您的資料執行哪些動作。透過定義角色並將角色指派給使用者，您可以限制使用者可以採取的動作，例如將使用者限制為僅 CREATETABLE 和 INSERT 命令。如需 CREATEROLE 命令的詳細資訊，請參閱 [the section called “CREATE ROLE”](#)。Amazon Redshift 具有一些系統定義的角色，您也可以使用這些角色將特定許可授予使用者。如需詳細資訊，請參閱 [the section called “Amazon Redshift 系統定義角色”](#)。

您只能對使用 ON SCHEMA 語法的資料庫使用者和使用者群組授予外部結構描述的 GRANT 或 REVOKEUSAGE 許可。EXTERNAL SCHEMA 搭配使用 ON 時 AWS Lake Formation，您只能使用 GRANT 和 AWS Identity and Access Management (IAM) 角色的 REVOKE 許可。如需許可的清單，請參閱語法。

對於儲存的程序，您可以授予的唯一許可是 EXECUTE。

您無法在交易區塊內執行 GRANT (在外部資源上) (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

若要查看使用者被授予的資料庫許可，請使用 [HAS_DATABASE_PRIVILEGE](#)。若要查看使用者被授予的結構描述許可，請使用 [HAS_SCHEMA_PRIVILEGE](#)。若要查看使用者被授予的資料表許可，請使用 [HAS_TABLE_PRIVILEGE](#)。

語法

```
GRANT { { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | ALTER | TRUNCATE }
[,...] | ALL [ PRIVILEGES ] }
  ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
  TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]

GRANT { { CREATE | TEMPORARY | TEMP | ALTER } [,...] | ALL [ PRIVILEGES ] }
  ON DATABASE db_name [, ...]
```

```

    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
    [, ...]

GRANT { { CREATE | USAGE | ALTER } [, ...] | ALL [ PRIVILEGES ] }
    ON SCHEMA schema_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
    [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
    FUNCTIONS IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
    [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
    ON { PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...] | ALL
    PROCEDURES IN SCHEMA schema_name [, ...] }
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
    [, ...]

GRANT USAGE
    ON LANGUAGE language_name [, ...]
    TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
    [, ...]

```

授予資料表的資料欄層級權限

下列語法適用於 Amazon Redshift 資料表和檢視上的資料欄層級許可。

```

GRANT { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
    ( column_name [, ...] ) }
    ON { [ TABLE ] table_name [, ...] }

    TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]

```

授予ASSUMEROLE許可

以下是授予具有指定角色之使用者和群組的ASSUMEROLE許可語法。若要開始使用 ASSUMEROLE 權限，請參閱 [授予ASSUMEROLE許可的使用備註](#)。

```

GRANT ASSUMEROLE
    ON { 'iam_role' [, ...] | default | ALL }
    TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]

```

```
FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]
```

授予 Redshift Spectrum 與 Lake Formation 整合的許可

以下是 Redshift Spectrum 與 Lake Formation 整合的語法。

```
GRANT { SELECT | ALL [ PRIVILEGES ] } ( column_list )
  ON EXTERNAL TABLE schema_name.table_name
  TO { IAM_ROLE iam_role } [, ...] [ WITH GRANT OPTION ]

GRANT { { SELECT | ALTER | DROP | DELETE | INSERT } [, ...] | ALL [ PRIVILEGES ] }
  ON EXTERNAL TABLE schema_name.table_name [, ...]
  TO { { IAM_ROLE iam_role } [, ...] | PUBLIC } [ WITH GRANT OPTION ]

GRANT { { CREATE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
  ON EXTERNAL SCHEMA schema_name [, ...]
  TO { IAM_ROLE iam_role } [, ...] [ WITH GRANT OPTION ]
```

授予資料共用許可

生產者端資料共用權限

以下是使用 向使用者ALTER或角色GRANT授予或SHARE許可的語法。使用者可以使用 ALTER 許可來變更資料共用，或使用 SHARE許可將用量授予取用者。ALTER 和 SHARE是您可以在資料共用上授予使用者和角色的唯一許可。

```
GRANT { ALTER | SHARE } ON DATASHARE datashare_name
  TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]
```

以下是在 Amazon Redshift 上使用 GRANT 進行資料共用使用許可的語法。您可以使用 USAGE許可將資料共用的存取權授予取用者。您無法將此許可授予使用者或使用者群組。此許可也不支援 GRANT陳述式WITHGRANTOPTION的。只有具有先前授予FOR資料共用SHARE許可的使用者或使用者群組，才能執行此類型的GRANT陳述式。

```
GRANT USAGE
  ON DATASHARE datashare_name
  TO NAMESPACE 'namespaceGUID' | ACCOUNT 'accountnumber' [ VIA DATA CATALOG ]
```

以下是如何將資料共用使用權授予 Lake Formation 帳戶的範例。

```
GRANT USAGE ON DATASHARE salesshare TO ACCOUNT '123456789012' VIA DATA CATALOG;
```

取用者端資料共用權限

以下是從GRANT資料共用建立之特定資料庫或結構描述的資料共用使用許可語法。

取用者存取從資料共用建立的資料庫所需的進一步許可，取決於用於從資料共用建立資料庫的CREATEDATABASE命令是否使用 WITHPERMISSIONS子句。如需CREATEDATABASE命令和 WITHPERMISSIONS子句的詳細資訊，請參閱 [CREATE DATABASE](#)。

不使用 WITHPERMISSIONS子句建立的資料庫

當您USAGE在沒有子WITHPERMISSIONS句的情況下，在從資料共用建立的資料庫上授予時，您不需要另外授予共用資料庫中物件的許可。在沒有子WITHPERMISSIONS句的情況下，在從資料共用建立的資料庫上授予使用權限的實體會自動存取資料庫中的所有物件。

使用 WITHPERMISSIONS子句建立的資料庫

當您在透過具有 WITHPERMISSIONS子句的資料共用建立共用資料庫的資料庫USAGE上授予時，仍必須授予共用資料庫中資料庫物件的相關許可，才能存取它們，就像您授予本機資料庫物件的許可一樣。若要將權限授予從資料共用建立之資料庫中的物件，請使用三部分語法 `database_name.schema_name.object_name`。若要將權限授予指向共用資料庫內共用結構描述的外部結構描述中的物件，請使用兩部分語法 `schema_name.object_name`。

```
GRANT USAGE ON { DATABASE shared_database_name [, ...] | SCHEMA shared_schema }  
TO { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

授予限定範圍權限

範圍許可可讓您將許可授予資料庫或結構描述內類型的所有物件的使用者或角色。具有範圍許可的使用者和角色在資料庫或結構描述中的所有目前和未來物件上具有指定的許可。

您可以在 [中](#) 檢視資料庫層級範圍許可的範圍 [SVV_DATABASE_PRIVILEGES](#)。您可以在 [中](#) 檢視結構描述層級範圍許可的範圍 [SVV_SCHEMA_PRIVILEGES](#)。

以下是將限定範圍權限授予使用者或角色的語法。如需有關範圍許可的詳細資訊，請參閱 [限定範圍權限](#)。

```
GRANT { CREATE | USAGE | ALTER } [, ...] | ALL [ PRIVILEGES ] }  
FOR SCHEMAS IN  
DATABASE db_name
```

```

TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

GRANT
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
  [, ...] } | ALL [PRIVILEGES] } }
FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name} [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name | } [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
FOR PROCEDURES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
TO { username [ WITH GRANT OPTION ] | ROLE role_name | } [, ...]

GRANT USAGE
FOR LANGUAGES IN
{DATABASE db_name}
TO { username [ WITH GRANT OPTION ] | ROLE role_name } [, ...]

```

請注意，範圍許可不會區分函數的許可和程序的許可。例如，下列陳述式bob會授予結構描述 中函數和程序的EXECUTE許可Sales_schema。

```
GRANT EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema TO bob;
```

授予機器學習許可

以下是 Amazon Redshift 上的機器學習模型許可語法。

```

GRANT CREATE MODEL
  TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]

GRANT { EXECUTE | ALL [ PRIVILEGES ] }
  ON MODEL model_name [, ...]

  TO { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
  [, ...]

```

授予角色許可

以下是在 Amazon Redshift 上使用授予角色許可的語法。

```
GRANT { ROLE role_name } [, ...] TO { { user_name [ WITH ADMIN OPTION ] } |  
  ROLE role_name }[, ...]
```

以下是在 Amazon Redshift 上對角色授予系統許可的語法。

```
GRANT  
  {  
    { CREATE USER | DROP USER | ALTER USER |  
      CREATE SCHEMA | DROP SCHEMA |  
      ALTER DEFAULT PRIVILEGES |  
      ACCESS CATALOG |  
      CREATE TABLE | DROP TABLE | ALTER TABLE |  
      CREATE OR REPLACE FUNCTION | CREATE OR REPLACE EXTERNAL FUNCTION |  
      DROP FUNCTION |  
      CREATE OR REPLACE PROCEDURE | DROP PROCEDURE |  
      CREATE OR REPLACE VIEW | DROP VIEW |  
      CREATE MODEL | DROP MODEL |  
      CREATE DATASHARE | ALTER DATASHARE | DROP DATASHARE |  
      CREATE LIBRARY | DROP LIBRARY |  
      CREATE ROLE | DROP ROLE |  
      TRUNCATE TABLE  
      VACUUM | ANALYZE | CANCEL }[, ...]  
  }  
  | { ALL [ PRIVILEGES ] }  
TO { ROLE role_name } [, ...]
```

針對資料列層級安全性政策篩選授予解釋許可

以下是授予許可的語法，以解釋EXPLAIN計劃中查詢的資料列層級安全政策篩選條件。您可以使用REVOKE陳述式撤銷權限。

```
GRANT EXPLAIN RLS TO ROLE rolename
```

以下是授予許可以略過查詢資料列層級安全性政策的語法。

```
GRANT IGNORE RLS TO ROLE rolename
```

將RLS查詢資料表的許可授予政策物件

以下是對指定資料列層級安全性政策授予許可的語法。

```
GRANT SELECT ON [ TABLE ] table_name [, ...]  
TO RLS POLICY policy_name [, ...]
```

參數

SELECT

准許使用SELECT陳述式從資料表或檢視中選取資料。還需要 SELECT許可才能參考 UPDATE或 DELETE操作的現有資料欄值。

INSERT

准許使用INSERT陳述式或COPY陳述式將資料載入資料表。

UPDATE

准許使用UPDATE陳述式更新資料表欄。UPDATE 操作也需要 SELECT 許可，因為它們必須參考資料表資料欄來決定要更新的資料列，或計算資料欄的新值。

DELETE

允許刪除資料表中的資料列。DELETE 操作也需要 SELECT 許可，因為它們必須參考資料表資料欄來決定要刪除的資料列。

DROP

准許捨棄資料表。此許可適用於 Amazon Redshift 和 AWS Glue Data Catalog 已啟用 Lake Formation 的。

REFERENCES

授予建立外部索引鍵限制條件的許可。您需要在被動參考資料表與主動參考資料表上授予此許可；否則，使用者就無法建立限制條件。

ALTER

根據資料庫物件而定，將下列許可授予使用者或使用者群組：

- 對於資料表，會ALTER授予許可來變更資料表或檢視。如需詳細資訊，請參閱[ALTER TABLE](#)。
- 對於資料庫，會ALTER授予修改資料庫的許可。如需詳細資訊，請參閱[ALTER DATABASE](#)。
- 對於結構描述，ALTER會授予許可來變更結構描述。如需詳細資訊，請參閱[ALTER SCHEMA](#)。

- 對於外部資料表，ALTER會授予許可，以變更已啟用 Lake Formation AWS Glue Data Catalog 的 中的資料表。此許可僅適用於使用 Lake Formation 時。

TRUNCATE

允許截斷資料表 如果沒有此許可，只有資料表的擁有者或超級使用者可以截斷資料表。如需 TRUNCATE命令的詳細資訊，請參閱 [the section called “TRUNCATE”](#)。

ALL [PRIVILEGES]

一次將所有可用的許可授予指定的使用者或角色。PRIVILEGES 關鍵字為選用。

GRANT ALL ON SCHEMA不會授予外部結構描述的CREATE許可。

您可以授予ALL許可給已啟用 Lake Formation AWS Glue Data Catalog 的 中的資料表。在此情況下，個別許可（例如 ALTER、SELECT等）會記錄在 Data Catalog 中。

Note

Amazon Redshift 不支援 RULE和 TRIGGER許可。如需詳細資訊，請前往 [不支援的 PostgreSQL 功能](#)。

ASSUMEROLE

准許對具有指定角色的使用者、角色或群組執行 COPY UNLOAD EXTERNAL 、 、 FUNCTION 和 CREATEMODEL命令。執行指定的命令時，使用者、角色或群組會擔任該角色。若要開始使用 ASSUMEROLE許可，請參閱 [授予ASSUMEROLE許可的使用備註](#)。

ON TABLE 【 】 table_name

授予資料表或檢視的指定許可。TABLE 關鍵字為選用。您可在一個陳述式中列出多個資料表和檢視。

ON ALL TABLES IN SCHEMA 結構描述_名稱

授予所參考結構描述中所有資料表和檢視的指定許可。

(column_name 【 , ...】) ON TABLE table_name

將指定的許可授予 Amazon Redshift 資料表或檢視的指定欄PUBLIC上的使用者、群組或。

(column_list) ON EXTERNAL TABLE schema_name.table_name

將指定的許可授予參考結構描述中 Lake Formation 資料表指定資料欄上的IAM角色。

ON EXTERNAL TABLE schema_name.table_name

將指定的許可授予參考結構描述中指定 Lake Formation 資料表上的IAM角色。

ON EXTERNAL SCHEMA 結構描述名稱

將指定的許可授予參考結構描述上的IAM角色。

ON iam_role

將指定的許可授予IAM角色。

TO username

指出接收許可的使用者。

至 IAM_ROLE iam_role

指示接收許可IAM的角色。

WITH GRANT OPTION

指出接收許可的使用者可以接著將相同的許可授予他人。WITH GRANT OPTION 無法授予 群組或 PUBLIC。

ROLE role_name

將許可授予角色。

GROUP group_name

將許可授予使用者群組。可以使用逗號分隔清單來指定多個使用者群組。

PUBLIC

對所有使用者授予指定的許可，包括之後建立的使用者。PUBLIC 代表一律包含所有使用者的群組。個別使用者的許可包含授予 的許可總和PUBLIC、授予使用者所屬任何群組的許可，以及個別授予使用者的任何許可。

授予 PUBLIC Lake Formation EXTERNALTABLE會導致將許可授予 Lake Formation 每個群組。

CREATE

根據資料庫物件而定，將下列許可授予使用者或使用者群組：

- 對於資料庫，CREATE 允許使用者在資料庫中建立結構描述。
- 對於結構描述，CREATE 允許使用者在結構描述中建立物件。若要重新命名物件，使用者必須具有 CREATE許可，並擁有要重新命名的物件。

- CREATE Amazon Redshift Spectrum 外部結構描述SCHEMA不支援 ON。若要授予外部結構描述中外部資料表的使用量，SCHEMA請將 USAGE ON 授予需要存取的使用者。只有外部結構描述的擁有者或超級使用者才可在外部結構描述中建立外部資料表。若要轉移外部結構描述的所有權，請使用 [ALTER SCHEMA](#) 來變更擁有者。

TEMPORARY | TEMP

授予許可以在指定的資料庫中建立臨時資料表。若要執行 Amazon Redshift Spectrum 查詢，資料庫使用者必須具有在資料庫中建立臨時資料表的許可。

Note

根據預設，使用者會獲得許可，透過其在PUBLIC群組中的自動成員資格來建立臨時資料表。若要移除任何使用者建立暫存資料表的許可，請從 PUBLIC群組撤銷TEMP許可。然後將建立暫存資料表的許可明確授予特定使用者或使用者群組。

ON DATABASE db_name

授予資料庫的指定許可。

USAGE

授予特定結構描述的USAGE許可，讓使用者可存取該結構描述中的物件。必須針對本機 Amazon Redshift 結構描述單獨授予這些物件的特定動作（例如，SELECT或資料表上的UPDATE許可）。根據預設，所有使用者都擁有 結構描述的 PUBLIC CREATE和 USAGE許可。

當您使用 ON SCHEMA語法USAGE授予外部結構描述時，您不需要單獨授予外部結構描述中物件的動作。對應的目錄許可會控制外部結構描述物件的精細許可。

ON SCHEMA 結構描述名稱

授予結構描述的指定許可。

GRANT CREATE Amazon Redshift Spectrum 外部結構描述SCHEMA不支援 ON SCHEMA和 GRANT ALL ON 中的CREATE許可。若要授予外部結構描述中外部資料表的使用量，SCHEMA請將 USAGE ON 授予需要存取的使用者。只有外部結構描述的擁有者或超級使用者才可在外部結構描述中建立外部資料表。若要轉移外部結構描述的所有權，請使用 [ALTER SCHEMA](#) 來變更擁有者。

EXECUTE ON ALL FUNCTIONS IN SCHEMA 結構描述名稱

授予所參考結構描述中所有函數的指定許可。

Amazon Redshift 不支援 GRANT 或 pg_catalog 命名空間中定義的 pg_proc 內建項目的 REVOKE 陳述式。

EXECUTE ON PROCEDURE procedure_name

授予特定預存程序的 EXECUTE 許可。由於預存程序名稱可以過載，因此您必須包含程序的引數清單。如需詳細資訊，請參閱 [命名預存程序](#)。

EXECUTE ON ALL PROCEDURES IN SCHEMA 結構描述名稱

授予所參考結構描述中所有預存程序的指定許可。

USAGE ON LANGUAGE language_name

授予語言的 USAGE 許可。

執行 [CREATE FUNCTION](#) 命令來建立使用者定義函數 (UDFs) 需要 USAGE ON LANGUAGE 許可。如需詳細資訊，請參閱 [UDF 安全性和許可](#)。

需要 USAGE ON LANGUAGE 許可才能透過執行 [CREATE PROCEDURE](#) 命令來建立預存程序。如需詳細資訊，請參閱 [預存程序的安全和權限](#)。

對於 Python UDFs，請使用 plpythonu。對於 SQL UDFs，請使用 sql。若為預存程序，請使用 plpgsql。

FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]

指定授予許可的 SQL 命令。您可以指定 ALL 授予 COPY、FUNCTION、UNLOAD EXTERNAL 和 CREATE MODEL 陳述式的許可。本條款僅適用於授予 ASSUME ROLE 許可。

ALTER

ALTER 准許使用者從資料共用新增或移除物件，或設定屬性 PUBLICACCESSIBLE。如需詳細資訊，請參閱 [ALTER DATASHARE](#)。

SHARE

授予使用者和使用者群組將資料取用者新增至資料共用的許可。需要此許可才能讓特定取用者 (帳戶或命名空間) 從其叢集存取資料共用。取用者可以是相同或不同的 AWS 帳戶，具有與全域唯一識別碼 () 指定的相同或不同的叢集命名空間 GUID。

ON DATASHARE datashare_name

授予參考資料共用的指定許可。如需有關取用者存取控制精細度的資訊，請參閱 [Amazon Redshift 中不同層級的資料共用](#)。

USAGE

當USAGE授予相同帳戶中的取用者帳戶或命名空間時，帳戶內的特定取用者帳戶或命名空間可以唯讀方式存取資料共用和資料共用的物件。

至 NAMESPACE '叢集命名空間 GUID'

表示相同帳戶中的命名空間，取用者可以在其中接收資料共用的指定許可。命名空間使用 128 位元英數字元 GUID。

結束 ACCOUNT 'accountnumber' 【 VIA DATA CATALOG 】

表示另一個帳戶的編號，其中的取用者可以接收資料共用的指定許可。指定 DATA 「VIACATALOG」表示您正在將資料共用的使用權限授予 Lake Formation 帳戶。省略此參數表示您將使用權授予擁有叢集的帳戶。

ON DATABASE shared_database_name> 【 , ...】

在指定資料共用中建立的指定資料庫上授予指定使用許可。

ON SCHEMA shared_schema

在指定資料共用中建立的指定結構描述上授予指定許可。

FOR { SCHEMAS | TABLES | FUNCTIONS | PROCEDURES | LANGUAGES } IN

指定要被授予權限的資料庫物件。IN 後面的參數定義授予權限的範圍。

CREATE MODEL

將CREATEMODEL許可授予特定使用者或使用者群組。

ON MODEL model_name

授予特定模型的EXECUTE許可。

ACCESS CATALOG

授予檢視角色可存取之物件之相關中繼資料的權限。

{ role } [, ...]

要授予另一個角色、使用者或的角色PUBLIC。

PUBLIC 代表一律包含所有使用者的群組。個別使用者的許可包含授予的許可總和PUBLIC、授予使用者所屬任何群組的許可，以及個別授予使用者的任何許可。

至 { user_name WITH ADMIN OPTION 【 】 } | 角色 } 【 , ...】

使用 WITH ADMIN OPTION、另一個角色或 將指定的角色授予指定的使用者PUBLIC。

WITH ADMIN OPTION 子句提供所有授予角色的管理選項給所有授予者。

EXPLAIN RLS TO ROLE 角色名稱

准許向角色解釋EXPLAIN計劃中查詢的資料列層級安全政策篩選條件。

IGNORE RLS 結束ROLE角色名稱

授予許可，以略過對角色查詢資料列層級安全性政策。

使用須知

若要進一步了解 的用量備註GRANT，請參閱 [the section called “使用須知”](#)。

範例

如需如何使用的範例GRANT，請參閱 [the section called “範例”](#)。

使用須知

若要授予物件的權限，您必須符合下列條件之一：

- 身為物件擁有者。
- 身為超級使用者。
- 具有該物件和權限的授予權限。

例如，下列命令可讓使用者 HR 在員工資料表上執行SELECT命令，並為其他使用者授予和撤銷相同的權限。

```
grant select on table employees to HR with grant option;
```

HR 無法授予除 以外的任何操作權限SELECT，也無法授予員工以外的任何其他資料表。

另一個範例是，下列命令可讓使用者 HR 在員工資料表上執行ALTER命令，並為其他使用者授予和撤銷相同的權限。

```
grant ALTER on table employees to HR with grant option;
```

HR 無法授予除 以外的任何操作權限ALTER，也無法授予員工以外的任何其他資料表。

在檢視上授予權限並不表示具有基礎資料表的權限。同樣地，在結構描述上授予權限並不表示具有結構描述中資料表的權限。請改為明確授與基礎資料表的存取權限。

若要將權限授予 AWS Lake Formation 資料表，與資料表外部結構描述相關聯的IAM角色必須具有將權限授予外部資料表的許可。下列範例會建立具有關聯IAM角色 的外部結構描述myGrantor。此IAM角色myGrantor具有將許可授予他人的許可。GRANT 命令會使用與外部結構描述myGrantor相關聯的IAM角色許可，將許可授予IAM角色 myGrantee。

```
create external schema mySchema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myGrantor'
create external database if not exists;
```

```
grant select
on external table mySchema.mytable
to iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

如果您GRANTALL擁有IAM角色的許可，個別許可會在相關的已啟用 Lake Formation 的資料目錄中授予。例如，下列 GRANT ALL 會導致授予的個別權限 ALTER (SELECT、DELETE、DROP和 INSERT) 顯示在 Lake Formation 主控台中。

```
grant all
on external table mySchema.mytable
to iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

超級使用者可以存取所有物件，無論設定物件權限的 GRANT和 REVOKE命令為何。

欄位層級存取控制的使用須知

下列使用須知適用於 Amazon Redshift 資料表和檢視上的資料欄層級權限。這些須知會描述資料表；除非我們明確提到例外，否則相同的須知會適用於檢視。

- 對於 Amazon Redshift 資料表，您只能授予資料欄層級的 SELECT和 UPDATE權限。對於 Amazon Redshift 檢視，您只能授予資料欄層級SELECT的權限。
- ALL 關鍵字是資料表GRANT上資料欄層級內容中使用的 SELECT和 UPDATE權限組合的同義詞。
- 如果您沒有資料表中所有資料欄SELECT的權限，執行 SELECT * 操作只會傳回您有權存取的資料欄。使用檢視時，SELECT* 操作會嘗試存取檢視中的所有資料欄。如果您沒有存取所有資料欄的許可，這些查詢會失敗，但出現許可遭拒的錯誤。

- SELECT * 在下列情況下，不會擴展為僅可存取的資料欄：
 - 您無法使用 SELECT * 建立僅可存取資料欄的一般檢視。
 - 您無法建立僅限使用 SELECT * 存取資料欄的具體化檢視。
- 如果您在資料表SELECT或檢視上具有或 UPDATE 權限，並新增資料欄，則您在資料表或檢視上仍然具有相同的權限，因此擁有其所有資料欄。
- 只有資料表的擁有者或超級使用者可以授與資料欄層級權限。
- 資料欄層級權限不支援 WITHGRANTOPTION 子句。
- 您無法同時在資料表層級和資料欄層級擁有相同的權限。例如，使用者 data_scientist 不能同時擁有資料表上的 SELECT 權限 employee 和資料欄上的 SELECT 權限 employee.department。對資料表和資料表內的資料欄授與相同權限時，請考慮下列結果：
 - 如果使用者具有資料表的資料表層級權限，則在資料欄層級授與相同權限就沒有作用。
 - 如果使用者具有資料表的資料表層級權限，對資料表的一或多個資料欄撤銷相同的權限，則會傳回錯誤。請改為在資料表層級撤銷權限。
 - 如果使用者具有資料欄層級權限，在資料表層級授與相同的權限，則會傳回錯誤。
 - 如果使用者具有資料欄層級權限，在資料表層級撤銷相同的權限，則會撤銷資料表上所有資料欄的資料欄和資料表權限。
- 您無法授與延遲繫結檢視的資料欄層級權限。
- 若要建立具體化檢視，您必須在基礎資料表上具有資料表層級 SELECT 權限。即使您擁有特定資料欄層級的權限，也無法針對這些資料欄建立具體化檢視。不過，您可以將 SELECT 權限授予具體化檢視的資料欄，類似於一般檢視。
- 若要查詢資料欄層級權限的授予，請使用 [PG_ATTRIBUTE_INFO](#) 檢視。

授予 ASSUMEROLE 許可的使用備註

下列用量備註適用於在 Amazon Redshift 中授予 ASSUMEROLE 許可。

您可以使用 ASSUMEROLE 許可來控制、或等命令上資料庫使用者 COPYUNLOADEXTERNALFUNCTION、角色或群組 IAM 的角色存取許可 CREATE MODEL。將 ASSUMEROLE 許可授予 IAM 角色的使用者、角色或群組後，使用者、角色或群組可以在執行命令時擔任該角色。ASSUMEROLE 許可可讓您視需要授予對適當命令的存取權。

只有資料庫超級使用者可以授予或撤銷使用者、角色和群組的 ASSUMEROLE 許可。超級使用者一律會保留 ASSUMEROLE 許可。

若要啟用使用者、角色和群組的 ASSUMEROLE 許可，超級使用者會執行下列兩個動作：

- 在叢集上執行下列陳述式一次：

```
revoke assumerole on all from public for all;
```

- 將 ASSUMEROLE 許可授予適當命令的使用者、角色和群組。

您可以在授予 ASSUMEROLE 許可時，在 ON 子句中指定角色鏈結。您可以使用逗號來分隔角色鏈中的角色，例如：Role1,Role2,Role3。如果在授予 ASSUMEROLE 許可時指定角色鏈結，您必須在執行 ASSUMEROLE 許可授予的操作時指定角色鏈結。執行 ASSUMEROLE 許可授予的操作時，您無法在角色鏈中指定個別角色。例如，如果使用者、角色或群組被授予角色鏈結 Role1,Role2,Role3，則您無法指定僅 Role1 執行操作。

如果使用者嘗試執行 COPY、UNLOAD、EXTERNAL 或 CREATE MODEL 操作 FUNCTION，但尚未獲得 ASSUMEROLE 許可，則會出現類似下列訊息。

```
ERROR: User awsuser does not have ASSUMEROLE permission on IAM role  
"arn:aws:iam::123456789012:role/RoleA" for COPY
```

若要列出已透過 ASSUMEROLE 許可授予 IAM 角色和命令存取權的使用者，請參閱 [HAS_ASSUMEROLE_PRIVILEGE](#)。若要列出已授予您指定使用者 IAM 的角色和命令許可，請參閱 [PG_GET_IAMROLE_BY_USER](#)。若要列出已授予您指定 IAM 角色存取權的使用者、角色和群組，請參閱 [PG_GETGRANTEE_BY_IAM_ROLE](#)。

授予機器學習許可的使用須知

您無法直接授予或撤銷與 ML 函數相關的許可。ML 函數屬於 ML 模型，而且許可是透過模型控制的。相反地，您可以授予 ML 模型相關的許可。下列範例會示範如何授予許可給所有使用者，以便執行與模型 customer_churn 相關聯的 ML 函數。

```
GRANT EXECUTE ON MODEL customer_churn TO PUBLIC;
```

您也可以將 ML 模型的所有許可授予使用者 customer_churn。

```
GRANT ALL on MODEL customer_churn TO ml_user;
```

如果結構描述中有 ML 函數，則授予 ML 函數相關的 EXECUTE 許可將會失敗，即使該 ML 函數已經具有透過 GRANT EXECUTE ON MODEL 取得的 EXECUTE 許可也是一樣。我們建議您在使用 CREATE MODEL 命令時，使用個別的結構描述，將 ML 函數本身保留在不同的結構描述中。下列範例示範如何執行此動作。


```
CREATE MODEL ml_schema.customer_churn
FROM customer_data
TARGET churn
FUNCTION ml_schema.customer_churn_prediction
IAM_ROLE default
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket'
);
```

範例

下列範例會將SALES資料表上的 SELECT 權限授予使用者 fred。

```
grant select on table sales to fred;
```

下列範例會將 QA_TICKIT 結構描述中所有資料表SELECT的權限授予使用者 fred。

```
grant select on all tables in schema qa_tickit to fred;
```

下列範例會將結構描述 QA_TICKIT 上的所有結構描述權限授予使用者群組 QA_USERS。結構描述權限為 CREATE和 USAGE。USAGE 會授予使用者對結構描述中物件的存取權，但不會授予許可，例如這些物件SELECT上的 INSERT或 。分別授與每個物件的權限。

```
create group qa_users;
grant all on schema qa_tickit to group qa_users;
```

下列範例會將 QA_TICKIT 結構描述中SALES資料表上的所有權限授予群組 QA_ 中的所有使用者 USERS。

```
grant all on table qa_tickit.sales to group qa_users;
```

下列範例會將 QA_TICKIT 結構描述中SALES資料表上的所有權限授予群組 QA_USERS和 RO_ 中的所有使用者USERS。

```
grant all on table qa_tickit.sales to group qa_users, group ro_users;
```

下列範例會將 QA_TICKIT 結構描述中SALES資料表上的DROP權限授予群組 QA_ 中的所有使用者 USERS。

```
grant drop on table qa_tickit.sales to group qa_users;>
```

以下一系列命令說明，結構描述的存取權不會授予結構描述中資料表的權限。

```
create user schema_user in group qa_users password 'Abcd1234';
create schema qa_tickit;
create table qa_tickit.test (col1 int);
grant all on schema qa_tickit to schema_user;
```

```
set session authorization schema_user;
select current_user;
```

```
current_user
-----
schema_user
(1 row)
```

```
select count(*) from qa_tickit.test;
```

```
ERROR: permission denied for relation test [SQL State=42501]
```

```
set session authorization dw_user;
grant select on table qa_tickit.test to schema_user;
set session authorization schema_user;
select count(*) from qa_tickit.test;
```

```
count
-----
0
(1 row)
```

以下一系列命令說明，檢視的存取權不代表能夠存取其基礎資料表。名為 VIEW_USER 的使用者無法從 DATE 資料表中選取，但此使用者已在 VIEW_ 上獲得所有許可 DATE。

```
create user view_user password 'Abcd1234';
create view view_date as select * from date;
grant all on view_date to view_user;
```

```
set session authorization view_user;
select current_user;
```

```
current_user
-----
view_user
(1 row)
```

```
select count(*) from view_date;
```

```
count
-----
365
(1 row)
```

```
select count(*) from date;
```

```
ERROR: permission denied for relation date
```

下列範例會將 `cust_profile` 資料表的 `cust_name` 和 `cust_phone` 資料欄 `SELECT` 的權限授予使用者 `user1`。

```
grant select(cust_name, cust_phone) on cust_profile to user1;
```

下列範例會將 `cust_name` 和 `cust_phone` 資料欄上的 `SELECT` 權限，以及 `cust_profile` 資料表資料欄 `cust_contact_preference` 上的 `UPDATE` 權限授予 `sales_group` 群組。

```
grant select(cust_name, cust_phone), update(cust_contact_preference) on cust_profile to
group sales_group;
```

下列範例顯示 `ALL` 關鍵字的使用，將資料表的三個資料欄上的 `SELECT` 和 `UPDATE` 權限授予 `cust_profilesales_admin` 群組。

```
grant ALL(cust_name, cust_phone, cust_contact_preference) on cust_profile to group
sales_admin;
```

下列範例會將cust_profile_vw檢視資料cust_name欄上的SELECT權限授予user2使用者。

```
grant select(cust_name) on cust_profile_vw to user2;
```

授予資料共用存取權的範例

下列範例顯示從GRANT資料共用建立的特定資料庫或結構描述的資料共用使用許可。

在下列範例中，生產者端管理員會將salesshare資料共用的USAGE許可授予指定的命名空間。

```
GRANT USAGE ON DATASHARE salesshare TO NAMESPACE  
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

在下列範例中，消費者端管理員將的USAGE許可授予 sales_db Bob。

```
GRANT USAGE ON DATABASE sales_db TO Bob;
```

在下列範例中，消費者端管理員將sales_schema結構描述的GRANTUSAGE許可授予Analyst_role角色。sales_schema是指向sales_db的外部結構描述。

```
GRANT USAGE ON SCHEMA sales_schema TO ROLE Analyst_role;
```

此時，Bob和Analyst_role可以存取sales_schema和sales_db中的所有資料庫物件。

下列範例顯示授予共用資料庫中物件的其他物件層級權限。只有在用來建立共用資料庫的CREATEDATABASE命令使用WITHPERMISSIONS子句時，才需要這些額外的許可。如果CREATEDATABASE命令未使用WITH PERMISSIONS，USAGE則在共用資料庫中授予會授予該資料庫中所有物件的完整存取權。

```
GRANT SELECT ON sales_db.sales_schema.tickit_sales_redshift to Bob;
```

授予限定範圍權限的範例

下列範例會將Sales_db資料庫中所有目前和未來結構描述的使用權授予Sales角色。

```
GRANT USAGE FOR SCHEMAS IN DATABASE Sales_db TO ROLE Sales;
```

下列範例會將Sales_db資料庫中所有目前和未來資料表的SELECT許可授予使用者alice，並授予許可alice，將中資料表的範圍許可授予Sales_db其他使用者。

```
GRANT SELECT FOR TABLES IN DATABASE Sales_db TO alice WITH GRANT OPTION;
```

下列範例會將Sales_schema結構描述中函數的EXECUTE許可授予使用者 bob。

```
GRANT EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema TO bob;
```

下列範例會將 ShareDb 資料庫的 ShareSchema 結構描述中所有資料表的所有權限授予 Sales 角色。當指定結構描述時，您可以使用兩部分格式 database.schema 指定結構描述的資料庫。

```
GRANT ALL FOR TABLES IN SCHEMA ShareDb.ShareSchema TO ROLE Sales;
```

以下範例與前面的範例是相同的。您可以使用 DATABASE 關鍵字來指定資料庫，而不是使用兩部分格式。

```
GRANT ALL FOR TABLES IN SCHEMA ShareSchema DATABASE ShareDb TO ROLE Sales;
```

授予 ASSUMEROLE 權限的範例

以下是授予 ASSUMEROLE 權限的範例。

下列範例顯示超級使用者在叢集上執行一次的REVOKE陳述式，以啟用使用者和群組的 ASSUMEROLE 權限。然後，超級使用者將ASSUMEROLE權限授予適當命令的使用者和群組。如需啟用使用者和群組專用ASSUMEROLE權的資訊，請參閱 [授予ASSUMEROLE許可的使用備註](#)。

```
revoke assumerole on all from public for all;
```

下列範例會將 ASSUMEROLE 權限授予使用者reg_user1，Redshift-S3-Read讓IAM角色執行 COPY操作。

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-S3-Read'  
to reg_user1 for copy;
```

下列範例會將 ASSUMEROLE 權限授予reg_user1IAM角色鏈 的使用者RoleA，RoleB以執行 UNLOAD 操作。

```
grant assumerole  
on 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB'
```

```
to reg_user1
for unload;
```

以下是使用IAM角色鏈 RoleA、 的UNLOAD命令範例RoleB。

```
unload ('select * from venue limit 10')
to 's3://companyb/redshift/venue_pipe_'
iam_role 'arn:aws:iam::123456789012:role/RoleA,arn:aws:iam::210987654321:role/RoleB';
```

下列範例會將 ASSUMEROLE 權限授予使用者reg_user1 , Redshift-Exfunc讓IAM角色建立外部函數。

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-Exfunc'
to reg_user1 for external function;
```

下列範例會將 ASSUMEROLE 權限授予IAM角色的使用者reg_user1 , Redshift-model以建立機器學習模型。

```
grant assumerole on 'arn:aws:iam::123456789012:role/Redshift-ML'
to reg_user1 for create model;
```

授予ROLE權限的範例

下列範例會將 sample_role1 授予 user1。

```
CREATE ROLE sample_role1;
GRANT ROLE sample_role1 TO user1;
```

下列範例使用 將 sample_role1 WITH ADMIN 授予 user1OPTION , 設定 user1 的目前工作階段 , 而 user1 會將 sample_role1 授予 user2。

```
GRANT ROLE sample_role1 TO user1 WITH ADMIN OPTION;
SET SESSION AUTHORIZATION user1;
GRANT ROLE sample_role1 TO user2;
```

下列範例會將 sample_role1 授予 sample_role2。

```
GRANT ROLE sample_role1 TO ROLE sample_role2;
```

下列範例會將 `sample_role2` 授予 `sample_role3` 和 `sample_role4`。然後嘗試將 `sample_role3` 授予 `sample_role1`。

```
GRANT ROLE sample_role2 TO ROLE sample_role3;
GRANT ROLE sample_role3 TO ROLE sample_role2;
ERROR: cannot grant this role, a circular dependency was detected between these roles
```

下列範例會將 `CREATEUSER` 系統權限授予 `sample_role1`。

```
GRANT CREATE USER TO ROLE sample_role1;
```

下列範例會將系統定義角色 `sys:dba` 授予 `user1`。

```
GRANT ROLE sys:dba TO user1;
```

下列範例會嘗試將循環相依性中的 `sample_role3` 授予 `sample_role2`。

```
CREATE ROLE sample_role3;
GRANT ROLE sample_role2 TO ROLE sample_role3;
GRANT ROLE sample_role3 TO ROLE sample_role2; -- fail
ERROR: cannot grant this role, a circular dependency was detected between these roles
```

INSERT

主題

- [語法](#)
- [參數](#)
- [使用須知](#)
- [INSERT 範例](#)

將新資料列插入資料表中。您可以插入具有 `VALUES` 語法的單一資料列、具有 `VALUES` 語法的多資料列，或查詢結果定義的一或多個資料列（`INSERT INTO...SELECT`）。

Note

我們強烈鼓勵您使用 [COPY](#) 命令來載入大量資料。使用個別 `INSERT` 陳述式填入資料表可能會非常慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用

INSERTINTOSELECT或 [CREATE TABLE AS](#) 來改善效能。如需使用 COPY命令載入資料表的詳細資訊，請參閱 [在 Amazon Redshift 中載入資料](#)。

Note

單一SQL陳述式的大小上限為 16 MB。

語法

```
INSERT INTO table_name [ ( column [, ...] ) ]  
{DEFAULT VALUES |  
VALUES ( { expression | DEFAULT } [, ...] )  
[, ( { expression | DEFAULT } [, ...] )  
[, ...] ] |  
query }
```

參數

table_name

暫時性或持久性資料表。只有資料表的擁有者或具有INSERT權限的使用者才能插入資料列。如果您使用查詢子句插入資料列，則必須具有查詢中命名SELECT資料表的權限。

Note

使用 INSERT (外部資料表) 將SELECT查詢的結果插入外部目錄上的現有資料表。如需詳細資訊，請參閱[INSERT \(外部資料表 \)](#)。

欄位

您可以將值插入資料表的一個或多個資料欄。您可以依任意順序列出目標資料欄名稱。如果您未指定資料欄清單，則要插入的值必須對應至CREATETABLE陳述式中宣告的順序中的資料表資料欄。若要插入的值數目少於資料表中的資料欄數，則會在前 n 個資料欄中載入。

宣告的預設值或 Null 值會載入陳述INSERT式中未列出的任何資料欄 (隱含或明確)。

DEFAULT VALUES

若在建立資料表時對資料表中的資料欄指派預設值，請使用這些關鍵字插入完全由預設值組成的資料列。若有任何資料欄未包含預設值，則會將 null 插入這些資料欄。如果宣告任何資料欄 NOTNULL，則INSERT陳述式會傳回錯誤。

VALUES

使用此關鍵字插入一個或多個資料列，每列包含一個或多個值。每一列的VALUES清單必須與資料欄清單相符。若要插入多個資料列，請使用逗號分隔符號分隔每份表達式清單。請勿重複VALUES關鍵字。多列INSERT陳述式的所有VALUES清單必須包含相同數量的值。

運算式

單一值，或判斷值為單一值的表達式。每個值都必須與其插入所在資料欄的資料類型相容。若值的資料類型與資料欄的宣告資料類型不相符，則會在可能的情況下自動轉換成相容的資料類型。例如：

- 十進位值1.1會以 的形式插入資料INT欄1。
- 十進位值100.8976會以 的形式插入 DEC (5 , 2) 欄中100.90。

您可以在表達式中包含類型轉換語法，藉此明確將值轉換成相容的資料類型。例如，如果資料表 T1 COL1中的資料欄是 CHAR (3) 資料欄：

```
insert into t1(col1) values('Incomplete'::char(3));
```

此陳述式會將值 Inc 插入資料欄。

對於單列INSERTVALUES陳述式，您可以使用純量子查詢作為表達式。子查詢的結果會插入適當的資料欄中。

Note

不支援將子查詢作為多列INSERTVALUES陳述式的表達式。

DEFAULT

使用此關鍵字可依資料表建立時所定義，插入資料欄的預設值。若資料欄沒有預設值，則會插入 null。如果該資料欄在CREATETABLE陳述式中沒有為其指派明確的預設值，則您無法將預設值插入有NOTNULL限制的資料欄。

query

藉由定義任何查詢將一個或多個資料列插入資料表。查詢產生的所有資料列都會插入資料表中。查詢必須傳回與資料表中資料欄相容的資料欄清單，但資料欄名稱不需相符。

使用須知

Note

我們強烈鼓勵您使用 [COPY](#) 命令來載入大量資料。使用個別INSERT陳述式填入資料表可能會非常慢。或者，如果您的資料已存在於其他 Amazon Redshift 資料庫資料表中，請使用 [INSERTINTOSELECT](#) 或 [CREATE TABLE AS](#) 來改善效能。如需使用 COPY命令載入資料表的詳細資訊，請參閱 [在 Amazon Redshift 中載入資料](#)。

插入值的資料格式必須符合CREATETABLE定義指定的資料格式。

將大量新資料列插入資料表之後：

- 清空資料表以回收儲存空間和重新排序資料列。
- 分析資料表以更新查詢規劃器的統計資訊。

當值插入DECIMAL資料欄且超過指定的規模時，載入的值會視需要四捨五入。例如，當的值20.259插入 DECIMAL (8 , 2) 資料欄時，儲存的值為 20.26。

您可以插入 GENERATED BY DEFAULT AS IDENTITY資料欄。您可以使用您提供的值IDENTITY來更新定義為 GENERATED BY DEFAULT AS 的資料欄。如需詳細資訊，請參閱[GENERATED BY DEFAULT AS IDENTITY](#)。

INSERT 範例

TICKIT 資料庫中的CATEGORY資料表包含下列資料列：

catid	catgroup	catname	catdesc
1	Sports	MLB	Major League Baseball
2	Sports	NHL	National Hockey League
3	Sports	NFL	National Football League
4	Sports	NBA	National Basketball Association

```

5 | Sports | MLS | Major League Soccer
6 | Shows | Musicals | Musical theatre
7 | Shows | Plays | All non-musical theatre
8 | Shows | Opera | All opera and light opera
9 | Concerts | Pop | All rock and pop music concerts
10 | Concerts | Jazz | All jazz singers and bands
11 | Concerts | Classical | All symphony, concerto, and choir concerts
(11 rows)

```

建立與資料表具有類似結構描述的 CATEGORY_STAGE CATEGORY 資料表，但定義資料欄的預設值：

```

create table category_stage
(catid smallint default 0,
catgroup varchar(10) default 'General',
catname varchar(10) default 'General',
catdesc varchar(50) default 'General');

```

下列INSERT陳述式會從CATEGORY資料表中選取所有列，並將其插入 CATEGORY_STAGE 資料表。

```

insert into category_stage
(select * from category);

```

查詢前後的括號是選用的。

此命令會將新資料列插入 CATEGORY_STAGE 資料表，並依下列順序為每個資料欄指定值：

```

insert into category_stage values
(12, 'Concerts', 'Comedy', 'All stand-up comedy performances');

```

您也可以插入結合特定值與預設值的新資料列：

```

insert into category_stage values
(13, 'Concerts', 'Other', default);

```

執行以下查詢來傳回插入的資料列：

```

select * from category_stage
where catid in(12,13) order by 1;

```

```

catid | catgroup | catname |          catdesc
-----+-----+-----+-----
    12 | Concerts | Comedy | All stand-up comedy performances
    13 | Concerts | Other  | General
(2 rows)

```

下列範例顯示一些多列INSERTVALUES陳述式。第一個範例會插入兩列的特定CATID值，以及兩列中其他資料欄的預設值。

```

insert into category_stage values
(14, default, default, default),
(15, default, default, default);

select * from category_stage where catid in(14,15) order by 1;
catid | catgroup | catname | catdesc
-----+-----+-----+-----
    14 | General  | General | General
    15 | General  | General | General
(2 rows)

```

下一個範例會插入包含不同的特定與預設值組合的三個資料列：

```

insert into category_stage values
(default, default, default, default),
(20, default, 'Country', default),
(21, 'Concerts', 'Rock', default);

select * from category_stage where catid in(0,20,21) order by 1;
catid | catgroup | catname | catdesc
-----+-----+-----+-----
     0 | General  | General | General
    20 | General  | Country | General
    21 | Concerts | Rock    | General
(3 rows)

```

VALUES 此範例中的第一組 會產生與DEFAULTVALUES指定單列INSERT陳述式相同的結果。

下列範例顯示資料表具有IDENTITY資料欄時INSERT的行為。首先，建立新的CATEGORY資料表版本，然後從 將資料列插入其中CATEGORY：

```
create table category_ident
```

```
(catid int identity not null,
catgroup varchar(10) default 'General',
catname varchar(10) default 'General',
catdesc varchar(50) default 'General');

insert into category_ident(catgroup,catname,catdesc)
select catgroup,catname,catdesc from category;
```

請注意，您無法將特定整數值插入資料CATIDIDENTITY欄。IDENTITY 資料欄值會自動產生。

下列範例示範子查詢無法用作多列INSERTVALUES陳述式中的表達式：

```
insert into category(catid) values
((select max(catid)+1 from category)),
((select max(catid)+2 from category));

ERROR: can't use subqueries in multi-row VALUES
```

下列範例會示範使用 WITH SELECT 子句將資料從 venue 資料表填入暫存資料表中的插入操作。如需 venue 資料表的相關資訊，請參閱 [範本資料庫](#)。

首先，建立暫存資料表 #venuetemp。

```
CREATE TABLE #venuetemp AS SELECT * FROM venue;
```

列出 #venuetemp 資料表中的資料列。

```
SELECT * FROM #venuetemp ORDER BY venueid;
```

venueid	venue name	venue city	venue state	venue seats
1	Toyota Park	Bridgeview	IL	0
2	Columbus Crew Stadium	Columbus	OH	0
3	RFK Stadium	Washington	DC	0
4	CommunityAmerica Ballpark	Kansas City	KS	0
5	Gillette Stadium	Foxborough	MA	68756
...				

使用 WITH SELECT 子句在 #venuetemp 資料表中插入 10 個重複的資料列。

```
INSERT INTO #venuetemp (WITH venuecopy AS (SELECT * FROM venue) SELECT * FROM venuecopy
ORDER BY 1 LIMIT 10);
```

列出 #venuetemp 資料表中的資料列。

```
SELECT * FROM #venuetemp ORDER BY venueid;
```

venueid	venue name	venue city	venue state	venue seats
1	Toyota Park	Bridgeview	IL	0
1	Toyota Park	Bridgeview	IL	0
2	Columbus Crew Stadium	Columbus	OH	0
2	Columbus Crew Stadium	Columbus	OH	0
3	RFK Stadium	Washington	DC	0
3	RFK Stadium	Washington	DC	0
4	CommunityAmerica Ballpark	Kansas City	KS	0
4	CommunityAmerica Ballpark	Kansas City	KS	0
5	Gillette Stadium	Foxborough	MA	68756
5	Gillette Stadium	Foxborough	MA	68756
...				

INSERT (外部資料表)

將SELECT查詢結果插入外部目錄上的現有外部資料表 AWS Lake Formation，例如 AWS Glue、或 Apache Hive 中繼存放區。使用與CREATEEXTERNALSCHEMA命令相同的 AWS Identity and Access Management (IAM) 角色，與外部目錄和 Amazon S3 互動。

對於非分割資料表，INSERT (外部資料表) 命令會根據指定的資料表屬性和檔案格式，將資料寫入資料表中定義的 Amazon S3 位置。

對於分割資料表，INSERT (外部資料表) 會根據資料表中指定的分割金鑰，將資料寫入 Amazon S3 位置。INSERT 操作完成後，它也會自動在外部目錄中註冊新的分割區。

您無法在交易區塊內執行 INSERT (外部資料表) (BEGIN ... END)。如需交易的相關資訊，請參閱 [可序列化隔離](#)。

語法

```
INSERT INTO external_schema.table_name
{ select_statement }
```

參數

`external_schema.table_name`

現有外部資料結構描述的名稱和要插入的目標外部資料表。

`select_statement`

透過定義任何查詢，將一或多個列插入外部資料表的陳述式。所有查詢產生的列都會根據資料表定義，以文字或 Parquet 的格式被寫入 Amazon S3。查詢必須傳回與外部資料表中的欄位資料類型相容的欄位清單。不過，欄位名稱不一定要相符。

使用須知

SELECT 查詢中的資料欄數必須與資料欄和分割區資料欄的總和相同。每個資料欄的位置和資料類型必須與外部資料表的資料類型相符。分割區資料欄的位置必須在SELECT查詢結尾，其順序必須與在CREATEEXTERNALTABLE命令中定義的順序相同。欄位名稱不一定要相符。

在某些情況下，您可能想要在 Data Catalog AWS Glue 或 Hive 中繼存放區上執行 INSERT (外部資料表) 命令。如果是 AWS Glue，用於建立外部結構描述IAM的角色必須同時具有 Amazon S3 和 的讀取和寫入許可 AWS Glue。如果您使用 AWS Lake Formation 目錄，此IAM角色會成為新 Lake Formation 資料表的擁有者。此IAM角色必須至少具有下列許可：

- SELECT、INSERT、外部資料表上的UPDATE許可
- 外部資料表 Amazon S3 路徑上的資料位置許可

為了確保檔案名稱是唯一的，Amazon Redshift 依預設會針對每個上傳到 Amazon S3 的檔案名稱使用下列格式。

```
<date>_<time>_<microseconds>_<query_id>_<slice-number>_part_<part-number>.<format>.
```

例如，20200303_004509_810669_1007_0001_part_00.parquet。

執行 INSERT (外部資料表) 命令時，請考慮下列事項：

- TEXTFILE 不支援格式為 PARQUET或 以外的外部資料表。
- 此命令支援現有的資料表屬性，如 'write.parallel'、'write.maxfilesize.mb'、'compression_type' 和 'serialization.null.format'。若要更新這些值，請執行 ALTERTABLESETTABLEPROPERTIES命令。

- 'numRows' 資料表屬性會在INSERT操作結束時自動更新。如果不是由 CREATE EXTERNAL TABLE AS 操作建立，則必須定義資料表屬性或將其新增至資料表。
- 外部SELECT查詢不支援子LIMIT句。反之，請使用巢狀LIMIT子句。
- 您可以使用 [STL_UNLOAD_LOG](#) 資料表來追蹤每個 INSERT (外部資料表) 操作寫入 Amazon S3 的檔案。
- Amazon Redshift 僅支援 INSERT (外部資料表) 的 Amazon S3 標準加密。

INSERT (外部資料表) 範例

下列範例會將SELECT陳述式的結果插入外部資料表。

```
INSERT INTO spectrum.lineitem
SELECT * FROM local_lineitem;
```

下列範例使用靜態分割，將SELECT陳述式的結果插入分割的外部資料表。SELECT 陳述式中的分割區資料欄為硬式編碼。分割區欄位必須位於查詢結尾。

```
INSERT INTO spectrum.customer
SELECT name, age, gender, 'May', 28 FROM local_customer;
```

下列範例使用動態分割，將SELECT陳述式的結果插入分割的外部資料表。分割區欄位並非硬式編碼。若已新增新的分割區，則資料會自動新增至現有的分割區資料夾或新的資料夾。

```
INSERT INTO spectrum.customer
SELECT name, age, gender, month, day FROM local_customer;
```

LOCK

限制對資料庫資料表的存取。此命令只有在交易區塊內執行時才有意義。

此LOCK命令會取得 " EXCLUSIVE"ACCESS 模式中的資料表層級鎖定，必要時會等待任何衝突鎖定解除。若以此方式明確鎖定資料表，則從其他交易或工作階段嘗試時，會造成資料表的讀取和寫入等待。若某位使用者建立明確的資料表鎖定，則會暫時阻止其他使用者從該資料表選取資料或將資料載入其中。當包含 LOCK命令的交易完成時，鎖定即會釋放。

參考資料表的命令 (例如寫入操作) 會以隱含方式取得限制較少的資料表鎖定。例如，若使用者嘗試從資料表讀取資料時，有另一位使用者正在更新資料表，則讀取的資料會是已遞交資料的快照 (在某些情況下，若查詢違反可序列化隔離規則，則會停止)。請參閱 [管理並行寫入操作](#)。

有些DDL操作，例如 DROPTABLE和 TRUNCATE，會建立專屬鎖定。這些操作會阻止資料讀取。

若發生鎖定衝突，Amazon Redshift 會顯示錯誤訊息，提醒進行發生衝突之交易的使用者。收到鎖定衝突的交易會停止。每次發生鎖定衝突時，Amazon Redshift 都會在 [STL_TR_CONFLICT](#) 資料表中寫入一項記錄。

語法

```
LOCK [ TABLE ] table_name [, ...]
```

參數

TABLE

選用的關鍵字。

table_name

要鎖定的資料表名稱。您可以使用逗號分隔的資料表名稱清單鎖定多個資料表。不過您無法鎖定檢視。

範例

```
begin;  
  
lock event, sales;  
  
...
```

MERGE

有條件地將來源資料表中的資料列合併到目標資料表中。傳統上，這只能透過單獨使用多個插入、更新或刪除陳述式來實現。如需MERGE可讓您合併的 操作的詳細資訊，請參閱 [UPDATE](#)、[DELETE](#)和 [INSERT](#)。

語法

```
MERGE INTO target_table  
USING source_table [ [ AS ] alias ]  
ON match_condition  
[ WHEN MATCHED THEN { UPDATE SET col_name = { expr } [,...] | DELETE }
```

```
WHEN NOT MATCHED THEN INSERT [ ( col_name [,...] ) ] VALUES ( { expr } [, ...] ) |  
REMOVE DUPLICATES ]
```

參數

target_table

MERGE 陳述式合併的暫時或永久資料表。

source_table

提供資料列以合併至 target_table 的臨時或永久性資料表。source_table 也可以是 Spectrum 資料表。

alias

source_table 暫時替代名稱。

此為選用參數。以 AS 開頭的 alias 也是選用項目。

match_condition

指定來源資料表資料欄與目標資料表資料欄之間的相等述詞，這些資料欄的作用是判斷 source_table 中的資料列是否可與 target_table 中的資料列相符。如果符合條件，會針對該資料列 MERGE 執行 matched_clause。否則，會針對該資料列 MERGE 執行 not_matched_clause。

WHEN MATCHED

指定當來源資料列與目標資料列之間的比對條件評估為 True 時，要執行的動作。您可以指定 UPDATE 動作或 DELETE 動作。

UPDATE

更新 target_table 中的相符資料列。只會更新 col_name 中您指定的值。

DELETE

刪除 target_table 中的相符資料列。

WHEN NOT MATCHED

指定當比對條件評估為 False 或未知時要執行的動作。您只能為此子句指定 INSERT 插入動作。

INSERT

根據 match_condition，從 source_table 插入不符合 target_table 中任何資料列的 target_table 資料列。您可以依任意順序列出目標 col_name。如果您不提供任何 col_name 值，則預設順序是按所有資料表資料欄的宣告順序排序。

col_name

您要修改的一個或多個資料欄名稱。不要在指定目標資料欄時包含資料表名稱。

expr

定義 col_name 新值的運算式。

REMOVE DUPLICATES

指定 MERGE 命令以簡化模式執行。簡化模式有以下要求：

- target_table 和 source_table 必須具有相同數目的資料欄和相容的資料欄類型。
- 從您的 MERGE 命令省略 WHEN 子句和 UPDATE 和 INSERT 子句。
- 在 MERGE 命令中使用 REMOVEDUPLICATES 子句。

在簡化模式中，MERGE 會執行下列動作：

- target_table 中具有 source_table 中相符項目的資料列會更新，以符合 source_table 中的值。
- source_table 中沒有 target_table 中相符項目的資料列會插入到 target_table 中。
- 當 target_table 中的多個資料列與 source_table 中的相同資料列相符時，重複的資料列會移除。Amazon Redshift 會保留一個資料列並對其進行更新。與 source_table 中的資料列不相符的重複資料列保持不變。

使用 REMOVE DUPLICATES 可提供比使用 WHENMATCHED 和 WHEN NOT 更好的效能 MATCHED。REMOVE DUPLICATES 如果 target_table 和 source_table 相容，建議您使用 `REMOVE DUPLICATES`，而您不需要在 target_table 中保留重複的資料列。

使用須知

- 若要執行 MERGE 陳述式，您必須同時是 source_table 和 target_table 的擁有者，或擁有這些資料表的 SELECT 許可。此外，視 MERGE 陳述式中包含的操作而定 DELETE，您必須具有 target_table UPDATE、和 INSERT 許可。
- target_table 不能是系統資料表、目錄資料表或外部資料表。
- source_table 和 target_table 不能是相同資料表。
- 您無法在 MERGE 陳述式中使用 WITH 子句。
- target_table 的資料列無法比對 source_table 中的多個資料列。

請思考下列範例：

```
CREATE TABLE target (id INT, name CHAR(10));
```

```
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (1, 'Bob'), (2, 'John');
INSERT INTO source VALUES (1, 'Tony'), (1, 'Alice'), (3, 'Bill');

MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN UPDATE SET id = source.id, name = source.name
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);
ERROR: Found multiple matches to update the same tuple.

MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN DELETE
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);
ERROR: Found multiple matches to update the same tuple.
```

在這兩個MERGE陳述式中，操作會失敗，因為source資料表中有多個ID值為的資料列1。

- match_condition 和 expr 無法部分參考SUPER類型資料欄。例如，如果您的SUPER類型物件是陣列或結構，則無法使用該資料欄的個別元素作為 match_condition 或 expr，但您可以使用整個資料欄。

請思考下列範例：

```
CREATE TABLE IF NOT EXISTS target (key INT, value SUPER);
CREATE TABLE IF NOT EXISTS source (key INT, value SUPER);

INSERT INTO target VALUES (1, JSON_PARSE('{"key": 88}'));
INSERT INTO source VALUES (1, ARRAY(1, 'John')), (2, ARRAY(2, 'Bill'));

MERGE INTO target USING source ON target.key = source.key
WHEN matched THEN UPDATE SET value = source.value[0]
WHEN NOT matched THEN INSERT VALUES (source.key, source.value[0]);
ERROR: Partial reference of SUPER column is not supported in MERGE statement.
```

如需SUPER類型的詳細資訊，請參閱[SUPER類型](#)。

- 如果 source_table 很大，則將 target_table 和 source_table 中的聯接資料欄定義為分佈索引鍵可以提高性能。
- 若要使用 REMOVE DUPLICATES 子句，您需要 target_table SELECT 的 INSERT、和 DELETE 許可。
- source_table 可以是檢視或子查詢。以下是 MERGE 陳述式的範例，其中 source_table 是移除重複資料列的子查詢。

```
MERGE INTO target
USING (SELECT id, name FROM source GROUP BY 1, 2) as my_source
ON target.id = my_source.id
WHEN MATCHED THEN UPDATE SET id = my_source.id, name = my_source.name
WHEN NOT MATCHED THEN INSERT VALUES (my_source.id, my_source.name);
```

範例

下列範例會建立兩個資料表，然後對其執行MERGE操作、更新目標資料表中的相符資料列，以及插入不相符的資料列。然後，它會將另一個值插入來源資料表並執行另一個MERGE操作，這次會刪除相符的資料列，並從來源資料表插入新的資料列。

首先建立並填入來源和目標資料表。

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (101, 'Bob'), (102, 'John'), (103, 'Susan');
INSERT INTO source VALUES (102, 'Tony'), (103, 'Alice'), (104, 'Bill');

SELECT * FROM target;
 id | name
-----+-----
 101 | Bob
 102 | John
 103 | Susan
(3 rows)

SELECT * FROM source;
 id | name
-----+-----
 102 | Tony
 103 | Alice
 104 | Bill
(3 rows)
```

接下來，將來源資料表合併到目標資料表，使用相符的資料列更新目標資料表，並從來源資料表插入不相符的資料列。

```
MERGE INTO target USING source ON target.id = source.id
```

```
WHEN MATCHED THEN UPDATE SET id = source.id, name = source.name
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);

SELECT * FROM target;
 id | name
-----+-----
 101 | Bob
 102 | Tony
 103 | Alice
 104 | Bill
(4 rows)
```

請注意，識別碼值為 102 和 103 的資料列會更新，以符合目標資料表中的名稱值。此外，ID 值為 104 且名稱值為 Bill 的新資料列會插入目標資料表中。

接下來，在來源資料表中插入新資料列。

```
INSERT INTO source VALUES (105, 'David');

SELECT * FROM source;
 id | name
-----+-----
 102 | Tony
 103 | Alice
 104 | Bill
 105 | David
(4 rows)
```

最後，執行合併操作，刪除目標資料表中的相符資料列，並插入不相符的資料列。

```
MERGE INTO target USING source ON target.id = source.id
WHEN MATCHED THEN DELETE
WHEN NOT MATCHED THEN INSERT VALUES (source.id, source.name);

SELECT * FROM target;
 id | name
-----+-----
 101 | Bob
 105 | David
(2 rows)
```

識別碼值為 102、103 和 104 的資料列會從目標資料表中刪除，而識別碼值為 105 且名稱值為 David 的新資料列會插入目標資料表中。

下列範例顯示使用 REMOVEDUPLICATES 子句之 MERGE 命令的簡化語法。

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (30, 'Tony'), (11, 'Alice'), (23, 'Bill');
INSERT INTO source VALUES (23, 'David'), (22, 'Clarence');

MERGE INTO target USING source ON target.id = source.id REMOVE DUPLICATES;

SELECT * FROM target;
id | name
----+-----
30 | Tony
11 | Alice
23 | David
22 | Clarence
(4 rows)
```

下列範例顯示使用 REMOVEDUPLICATES 子句的 MERGE 命令簡化語法，如果其在 source_table 中有相符的資料列，則從 target_table 中移除重複的資料列。

```
CREATE TABLE target (id INT, name CHAR(10));
CREATE TABLE source (id INT, name CHAR(10));

INSERT INTO target VALUES (30, 'Tony'), (30, 'Daisy'), (11, 'Alice'), (23, 'Bill'),
(23, 'Nikki');
INSERT INTO source VALUES (23, 'David'), (22, 'Clarence');

MERGE INTO target USING source ON target.id = source.id REMOVE DUPLICATES;

SELECT * FROM target;
id | name
----+-----
30 | Tony
30 | Daisy
11 | Alice
23 | David
22 | Clarence
```

(5 rows)

MERGE 執行後，target_table 中只有一列 ID 值為 23。由於 source_table 中沒有 ID 值為 30 的資料列，所以 ID 值為 30 的兩個重複資料列會保留在 target_table 中。

另請參閱

[INSERT](#), [UPDATE](#), [DELETE](#)

PREPARE

準備陳述式供執行。

PREPARE 會建立準備好的陳述式。執行PREPARE陳述式時，會剖析SELECT、重寫和規劃指定的陳述式 INSERT (UPDATE、或 DELETE)。然後，在為準備好的陳述式發出EXECUTE命令時，Amazon Redshift 可能會選擇性地修改查詢執行計畫（以根據指定的參數值改善效能），然後再執行準備好的陳述式。

語法

```
PREPARE plan_name [ (datatype [, ...] ) ] AS statement
```

參數

plan_name

給予此特定預備陳述式的任意名稱。此名稱在單一工作階段內必須是唯一的，且後續會用來執行或解除配置先前的預備陳述式。

datatype

預備陳述式的參數資料類型。若要在預備陳述式本身內參考參數，請使用 \$1、\$2，以此類推。

statement

任何 SELECT、UPDATE、INSERT 或 DELETE 陳述式。

使用須知

預備陳述式可採用參數：這些值會在陳述式執行時替換到其中。若要將參數包含在準備好的陳述式中，請在PREPARE陳述式中提供資料類型清單，並在陳述式中，使用符號 \$1、\$2、... 依位置參考參數執行陳述式時，請在EXECUTE陳述式中指定這些參數的實際值。如需詳細資訊，請參閱[EXECUTE](#)。

預備陳述式只會在目前工作階段期間內存在。當工作階段結束時，就會捨棄預備陳述式，因此必須再次建立它才能再度使用。這也表示，單一預備陳述式無法供多個同步的資料庫用戶端使用；不過，每個用戶端可建立自己要使用的預備陳述式。您可以使用 DEALLOCATE 命令手動移除準備的陳述式。

在使用單一工作階段執行大量類似的陳述式時，預備陳述式能獲得最大效能。如前所述，每次重頭執行預備陳述式時，Amazon Redshift 都會根據指定的參數值修改查詢執行計畫以提升效能。若要檢查 Amazon Redshift 為任何特定EXECUTE陳述式選擇的查詢執行計畫，請使用 [EXPLAIN](#) 命令。

如需 Amazon Redshift 為進行查詢最佳化所收集的查詢計畫和統計資訊的相關資訊，請參閱 [ANALYZE](#) 命令。

範例

建立暫存資料表、準備INSERT陳述式，然後執行它：

```
DROP TABLE IF EXISTS prep1;
CREATE TABLE prep1 (c1 int, c2 char(20));
PREPARE prep_insert_plan (int, char)
AS insert into prep1 values ($1, $2);
EXECUTE prep_insert_plan (1, 'one');
EXECUTE prep_insert_plan (2, 'two');
EXECUTE prep_insert_plan (3, 'three');
DEALLOCATE prep_insert_plan;
```

準備SELECT陳述式，然後執行它：

```
PREPARE prep_select_plan (int)
AS select * from prep1 where c1 = $1;
EXECUTE prep_select_plan (2);
EXECUTE prep_select_plan (3);
DEALLOCATE prep_select_plan;
```

另請參閱

[DEALLOCATE](#), [EXECUTE](#)

REFRESH MATERIALIZED VIEW

重新整理具體化檢視。

當您建立具體化視觀表時，其內容會反映當時基礎資料庫資料表或資料表的狀態。即使應用程式對基底資料表中的資料進行變更，具體化檢視中的資料仍會維持不變。如要更新具體化視觀表中的資料，您可以隨時使用 REFRESH MATERIALIZED VIEW 陳述式。當您執行此陳述式時，Amazon Redshift 會識別在基底資料表或資料表中發生的變更，然後將這些變更套用到具體化視觀表。

如需具體化檢視的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

語法

```
REFRESH MATERIALIZED VIEW mv_name
```

參數

mv_name

要重新整理的具體化檢視名稱。

使用須知

只有具體化檢視的擁有者才能對該檢視執行 REFRESH MATERIALIZED VIEW 操作。此外，擁有者必須具有基礎基礎資料表的 SELECT 權限，才能成功執行 REFRESH MATERIALIZED VIEW。

REFRESH MATERIALIZED VIEW 命令會以自己所屬的交易形式執行。遵循 Amazon Redshift 交易語意，以決定 REFRESH 命令可以看到基礎資料表中的哪些資料，或是當 REFRESH 命令進行變更時，在 Amazon Redshift 中執行的其他交易是否會顯示。

- 針對累加式具體化檢視，REFRESH MATERIALIZED VIEW 只會使用已遞交的基底資料表資料列。因此，如果重新整理操作在相同交易中的資料處理語言（DML）陳述式之後執行，則重新整理時不會看到該DML陳述式的變更。
- 針對具體化視觀表的完全重新整理，REFRESH MATERIALIZED VIEW 會根據一般 Amazon Redshift 交易語意，查看重新整理交易可見的所有基底資料表資料列。
- 視輸入引數類型而定，Amazon Redshift 仍支援下列具有特定輸入引數類型的函數的具體化檢視增量重新整理：DATE（時間戳記）、DATE_PART（日期、時間、間隔、時區）、DATE_TRUNC（時間戳記、間隔）。
- 基礎資料表位於資料共用中的具體化視觀表支援累加式重新整理。

Amazon Redshift 中的某些操作會和具體化視觀表互動。即使定義具體化視觀表的查詢僅使用符合增量重新整理資格SQL的功能，這些操作中的某些操作可能會強制REFRESH MATERIALIZED VIEW操作完全重新計算具體化視觀表。例如：

- 如果具體化檢視並未重新整理，則背景 vacuum 操作可能會遭到封鎖。在內部定義的閾值期間後，便會允許執行 vacuum 操作。發生此 vacuum 操作時，任何依存的具體化檢視都會標記為在下一次重新整理時重新進行運算 (即使這些具體化檢視為累加式也一樣)。如需的相關資訊VACUUM，請參閱 [VACUUM](#)。如需事件和狀態變更的相關資訊，請參閱 [STL_MV_STATE](#)。
- 基礎資料表上的某些使用者起始操作會強制將具體化視觀表在下次執行REFRESH操作時完全重新計算。此類操作的範例包括手動調用 VACUUM、傳統調整大小、ALTERDISTKEY操作、ALTERSORTKEY操作和截斷操作。在某些情況下，自動操作也可能導致在下次執行REFRESH操作時完全重新計算具體化視觀表。例如，自動清空刪除操作可能會導致完整重新計算。如需事件和狀態變更的相關資訊，請參閱 [STL_MV_STATE](#)。

資料共用中具體化視觀表的增量重新整理

共用基礎資料表時，Amazon Redshift 支援消費者資料共用中具體化視觀表的自動和增量重新整理。增量重新整理是 Amazon Redshift 識別在上一次重新整理後基礎資料表或資料表中發生的變更，並僅更新具體化檢視中對應的記錄的操作。如需此行為的詳細資訊，請參閱 [CREATE MATERIALIZED VIEW](#)。

增量重新整理的限制

Amazon Redshift 目前不支援使用下列任何SQL元素透過查詢定義的具體化檢視增量重新整理：

- OUTER JOIN (RIGHT、LEFT或 FULL)。
- 設定操作：UNION、INTERSECT、EXCEPT、MINUS。
- UNION ALL 當它發生在子查詢中，且查詢中存在彙總函數或 GROUP BY 子句時。
- 彙總函數：
MEDIAN、PERCENTILE_CONT、LISTAGG、STDDEV_SAMP、STDDEV_POP、APPROXIMATECOUNT、APPROXIMATE PERCENTILE和 位元彙總函數。

Note

支援 COUNT、SUM、MAX、MIN和 AVG彙總函數。

- DISTINCT 彙總函數，例如 DISTINCT SUM、COUNT DISTINCT 等。

- 視窗函數。
- 使用暫存資料表進行查詢最佳化的查詢，例如最佳化通用子運算式。
- 子查詢
- 在定義具體化視觀表的查詢中，參照下列查詢中的格式的外部資料表。
 - Delta Lake
 - Hudi

使用參照預覽軌跡上其他格式的外部資料表定義的具體化視觀表，支援累加式重新整理。如需詳細資訊，請參閱 Amazon Redshift 管理指南中的[建立預覽叢集](#)。如需設定預覽工作群組的詳細資訊，請參閱 Amazon Redshift 管理指南中的[建立預覽工作群組](#)。

- 可互變函數，例如日期時間函數RANDOM和非STABLE使用者定義的函數。
- 如需零ETL整合增量重新整理的限制，請參閱[使用零ETL整合與 Amazon Redshift 時的考量](#)事項。

如需具體化視觀表限制的詳細資訊，包括背景操作對VACUUM具體化視觀表重新整理操作的影響，請參閱 [使用須知](#)。

範例

以下範例會重新整理 tickets_mv 具體化檢視。

```
REFRESH MATERIALIZED VIEW tickets_mv;
```

RESET

將組態參數的值還原為其預設值。

您可以重設單一指定參數，或一次還原所有參數。若要將參數設定為特定值，請使用 [SET](#) 命令。若要顯示參數的目前值，請使用 [SHOW](#) 命令。

語法

```
RESET { parameter_name | ALL }
```

下列陳述式會將工作階段內容變數的值設定為 NULL。

```
RESET { variable_name | ALL }
```

參數

parameter_name

要重設的參數名稱。如需更多有關參數的文件，請參閱 [修改伺服器組態](#)。

ALL

重設所有執行期參數，包括所有工作階段內容變數。

variable

要重設的變數名稱 如果 的值RESET是工作階段內容變數，Amazon Redshift 會將其設定為 NULL。

範例

以下範例會將 query_group 參數重設為其預設值：

```
reset query_group;
```

以下範例會將所有執行時間參數重設為其預設值。

```
reset all;
```

下列範例會重設內容變數。

```
RESET app_context.user_id;
```

REVOKE

從使用者或角色移除存取許可，例如建立、捨棄或更新資料表的許可。

您只能使用 ON SCHEMA語法對資料庫使用者和角色進行外部結構描述的 GRANT或 REVOKEUSAGE許可。將 ON EXTERNALSCHEMA與 搭配使用時 AWS Lake Formation，您只能使用 GRANT和 AWS Identity and Access Management (IAM) 角色的REVOKE許可。如需許可的清單，請參閱語法。

對於預存程序，PUBLIC依預設會將 USAGE ON LANGUAGEplpgsql許可授予。EXECUTE 根據預設，ON PROCEDURE許可只會授予擁有者和超級使用者。

在REVOKE命令中指定您要移除的許可。若要授予許可，請使用 [GRANT](#) 命令。

語法

```
REVOKE [ GRANT OPTION FOR ]
{ { SELECT | INSERT | UPDATE | DELETE | DROP | REFERENCES | ALTER | TRUNCATE } [,...] |
  ALL [ PRIVILEGES ] }
ON { [ TABLE ] table_name [, ...] | ALL TABLES IN SCHEMA schema_name [, ...] }
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
{ { CREATE | TEMPORARY | TEMP | ALTER } [,...] | ALL [ PRIVILEGES ] }
ON DATABASE db_name [, ...]
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
{ { CREATE | USAGE | ALTER } [,...] | ALL [ PRIVILEGES ] }
ON SCHEMA schema_name [, ...]
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
EXECUTE
    ON FUNCTION function_name ( [ [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
{ { EXECUTE } [,...] | ALL [ PRIVILEGES ] }
    ON PROCEDURE procedure_name ( [ [ argname ] argtype [, ...] ] ) [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

```
REVOKE [ GRANT OPTION FOR ]
USAGE
    ON LANGUAGE language_name [, ...]
    FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

```
[ RESTRICT ]
```

授予資料表的資料欄層級許可

下列語法適用於 Amazon Redshift 資料表和檢視上的資料欄層級許可。

```
REVOKE { { SELECT | UPDATE } ( column_name [, ...] ) [, ...] | ALL [ PRIVILEGES ]
        ( column_name [, ...] ) }
        ON { [ TABLE ] table_name [, ...] }
        FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
        [ RESTRICT ]
```

撤銷ASSUMEROLE許可

以下是從具有指定角色的使用者和群組撤銷ASSUMEROLE許可的語法。

```
REVOKE ASSUMEROLE
        ON { 'iam_role' [, ...] | default | ALL }
        FROM { user_name | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
        FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL }
```

撤銷 Redshift Spectrum 使用 Lake Formation 的許可

以下是 Redshift Spectrum 與 Lake Formation 整合的語法。

```
REVOKE [ GRANT OPTION FOR ]
{ SELECT | ALL [ PRIVILEGES ] } ( column_list )
  ON EXTERNAL TABLE schema_name.table_name
  FROM { IAM_ROLE iam_role } [, ...]

REVOKE [ GRANT OPTION FOR ]
{ { SELECT | ALTER | DROP | DELETE | INSERT } [, ...] | ALL [ PRIVILEGES ] }
  ON EXTERNAL TABLE schema_name.table_name [, ...]
  FROM { { IAM_ROLE iam_role } [, ...] | PUBLIC }

REVOKE [ GRANT OPTION FOR ]
{ { CREATE | ALTER | DROP } [, ...] | ALL [ PRIVILEGES ] }
  ON EXTERNAL SCHEMA schema_name [, ...]
  FROM { IAM_ROLE iam_role } [, ...]
```

撤銷資料共用許可

生產者端資料共用權限

以下是使用 從使用者ALTER或角色REVOKE移除或SHARE許可的語法。權限已被撤銷的使用者無法再變更資料共用，或將使用權授予取用者。

```
REVOKE { ALTER | SHARE } ON DATASHARE datashare_name
FROM { username [ WITH GRANT OPTION ] | ROLE role_name | GROUP group_name | PUBLIC }
[, ...]
```

以下是使用 REVOKE移除取用者對資料共用存取權的語法。

```
REVOKE USAGE
ON DATASHARE datashare_name
FROM NAMESPACE 'namespaceGUID' [, ...] | ACCOUNT 'accountnumber' [ VIA DATA CATALOG ]
[, ...]
```

以下是將資料共用使用權從 Lake Formation 帳戶撤銷的範例。

```
REVOKE USAGE ON DATASHARE salesshare FROM ACCOUNT '123456789012' VIA DATA CATALOG;
```

取用者端資料共用權限

以下是從資料共用建立之特定資料庫或結構描述的資料共用使用許可REVOKE語法。從使用 WITHPERMISSIONS子句建立的資料庫撤銷使用許可不會撤銷您授予使用者或角色的任何其他許可，包括授予基礎物件的物件層級許可。如果您重新授予該使用者或角色的使用權限，他們會保留他們在您撤銷使用權之前擁有的其他所有權限。

```
REVOKE USAGE ON { DATABASE shared_database_name [, ...] | SCHEMA shared_schema }
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
```

撤銷限定範圍權限

範圍許可可讓您將許可授予資料庫或結構描述內類型的所有物件的使用者或角色。具有範圍許可的使用者和角色在資料庫或結構描述內所有目前和未來物件具有指定的許可。

您可以在 中檢視資料庫層級範圍許可的範圍[SVV_DATABASE_PRIVILEGES](#)。您可以在 中檢視結構描述層級範圍許可的範圍[SVV_SCHEMA_PRIVILEGES](#)。

以下是撤銷使用者和角色之限定範圍權限的語法。如需範圍許可的詳細資訊，請參閱 [限定範圍權限](#)。

```
REVOKE [ GRANT OPTION ]
```



```

{ CREATE | USAGE | ALTER } [,...] | ALL [ PRIVILEGES ] }
FOR SCHEMAS IN
DATABASE db_name
FROM { username | ROLE role_name } [, ...]

REVOKE [ GRANT OPTION ]
{ { SELECT | INSERT | UPDATE | DELETE | DROP | ALTER | TRUNCATE | REFERENCES }
  [, ...] } | ALL [PRIVILEGES] } }
FOR TABLES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name} [, ...]

REVOKE [ GRANT OPTION ] { EXECUTE | ALL [ PRIVILEGES ] }
FOR FUNCTIONS IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name | } [, ...]

REVOKE [ GRANT OPTION ] { EXECUTE | ALL [ PRIVILEGES ] }
FOR PROCEDURES IN
{SCHEMA schema_name [DATABASE db_name ] | DATABASE db_name }
FROM { username | ROLE role_name | } [, ...]

REVOKE [ GRANT OPTION ] USAGE
FOR LANGUAGES IN
{DATABASE db_name}
FROM { username | ROLE role_name } [, ...]

```

請注意，範圍許可不會區分函數的許可和程序的許可。例如，下列陳述式會在結構描述 bob 中從 撤銷函數和程序的EXECUTE許可Sales_schema。

```
REVOKE EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema FROM bob;
```

撤銷機器學習許可

以下是 Amazon Redshift 上的機器學習模型許可語法。

```

REVOKE [ GRANT OPTION FOR ]
  CREATE MODEL FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
  [ RESTRICT ]

REVOKE [ GRANT OPTION FOR ]
  { EXECUTE | ALL [ PRIVILEGES ] }
  ON MODEL model_name [, ...]

```

```
FROM { username | ROLE role_name | GROUP group_name | PUBLIC } [, ...]
[ RESTRICT ]
```

撤銷角色許可

以下是在 Amazon Redshift 上使用撤銷角色許可的語法。

```
REVOKE [ ADMIN OPTION FOR ] { ROLE role_name } [, ...] FROM { user_name } [, ...]
```

```
REVOKE { ROLE role_name } [, ...] FROM { ROLE role_name } [, ...]
```

以下是在 Amazon Redshift 上對角色撤銷系統許可的語法。

```
REVOKE
{
  { CREATE USER | DROP USER | ALTER USER |
    CREATE SCHEMA | DROP SCHEMA |
    ALTER DEFAULT PRIVILEGES |
    ACCESS CATALOG |
    CREATE TABLE | DROP TABLE | ALTER TABLE |
    CREATE OR REPLACE FUNCTION | CREATE OR REPLACE EXTERNAL FUNCTION |
    DROP FUNCTION |
    CREATE OR REPLACE PROCEDURE | DROP PROCEDURE |
    CREATE OR REPLACE VIEW | DROP VIEW |
    CREATE MODEL | DROP MODEL |
    CREATE DATASHARE | ALTER DATASHARE | DROP DATASHARE |
    CREATE LIBRARY | DROP LIBRARY |
    CREATE ROLE | DROP ROLE
    TRUNCATE TABLE
    VACUUM | ANALYZE | CANCEL }[, ...]
}
| { ALL [ PRIVILEGES ] }
FROM { ROLE role_name } [, ...]
```

針對資料列層級安全性政策篩選撤銷解釋許可

以下是撤銷許可的語法，以解釋 EXPLAIN 計劃中查詢的資料列層級安全政策篩選條件。您可以使用 REVOKE 陳述式撤銷權限。

```
REVOKE EXPLAIN RLS FROM ROLE rolename
```

以下是授予許可以略過查詢資料列層級安全性政策的語法。

```
REVOKE IGNORE RLS FROM ROLE rolename
```

以下是從指定資料列層級安全性政策撤銷許可的語法。

```
REVOKE SELECT ON [ TABLE ] table_name [, ...]  
FROM RLS POLICY policy_name [, ...]
```

參數

GRANT OPTION FOR

只會撤銷對其他使用者授予指定許可的選項，而不會撤銷許可本身。您無法GRANTOPTION從群組或撤銷PUBLIC。

SELECT

撤銷使用SELECT陳述式從資料表或檢視選取資料的許可。

INSERT

撤銷使用INSERT陳述式或COPY陳述式將資料載入資料表的許可。

UPDATE

撤銷使用UPDATE陳述式更新資料表欄的許可。

DELETE

撤銷從資料表中刪除資料列的許可。

REFERENCES

撤銷建立外部索引鍵限制條件的許可。您應在被動參考資料表與主動參考資料表上撤銷此許可。

TRUNCATE

撤銷截斷資料表的許可。如果沒有此許可，只有資料表的擁有者或超級使用者可以截斷資料表。如需TRUNCATE命令的詳細資訊，請參閱[the section called "TRUNCATE"](#)。

ALL [PRIVILEGES]

從指定的使用者或群組一次撤銷所有可用許可。PRIVILEGES關鍵字為選用。

Note

Amazon Redshift 不支援 RULE 和 TRIGGER 許可。如需詳細資訊，請前往 [不支援的 PostgreSQL 功能](#)。

ALTER

根據資料庫物件而定，將下列許可從使用者或使用者群組中撤銷：

- 對於資料表，會 ALTER 撤銷變更資料表或檢視的許可。如需詳細資訊，請參閱 [ALTER TABLE](#)。
- 對於資料庫，會 ALTER 撤銷修改資料庫的許可。如需詳細資訊，請參閱 [ALTER DATABASE](#)。
- 對於結構描述，ALTER 會授予撤銷以變更結構描述。如需詳細資訊，請參閱 [ALTER SCHEMA](#)。
- 對於外部資料表，會 ALTER 撤銷在已啟用 Lake Formation AWS Glue Data Catalog 的中變更資料表的許可。此許可僅適用於使用 Lake Formation 時。

DROP

撤銷捨棄資料表的許可。此許可適用於 Amazon Redshift 和 AWS Glue Data Catalog 已啟用 Lake Formation 的。

ASSUMEROLE

撤銷從具有指定角色的使用者 COPYUNLOADEXTERNALFUNCTION、角色或群組執行、或 CREATEMODEL 命令的許可。

ON TABLE 【 】 table_name

撤銷資料表或檢視的指定許可。TABLE 關鍵字為選用。

ON ALL TABLES IN SCHEMA 結構描述名稱

撤銷所參考結構描述中所有資料表的指定許可。

(column_name 【 , ... 】) ON TABLE table_name

從 Amazon Redshift 資料表或檢視的指定資料欄 PUBLIC 上的使用者、群組或撤銷指定的許可。

(column_list) ON EXTERNAL TABLE schema_name.table_name

在參考結構描述的 Lake Formation 資料表的指定資料欄上，從 IAM 角色撤銷指定的許可。

ON EXTERNAL TABLE schema_name.table_name

撤銷參考結構描述中指定 Lake Formation 資料表上 IAM 角色的指定許可。

ON EXTERNAL SCHEMA 結構描述名稱

從參考結構描述上的IAM角色撤銷指定的許可。

FROM IAM_ROLE iam_role

指示IAM角色失去許可。

ROLE role_name

從指定的角色中撤銷許可。

GROUP group_name

從指定的使用者群組中撤銷許可。

PUBLIC

從所有使用者中撤銷指定許可。PUBLIC 代表一律包含所有使用者的群組。個別使用者的許可包含授予的許可總和PUBLIC、授予使用者所屬任何群組的許可，以及個別授予使用者的任何許可。

從 Lake Formation PUBLIC 外部資料表撤銷會導致從 Lake Formation 每個群組撤銷許可。

CREATE

根據資料庫物件而定，將下列許可從使用者或群組中撤銷：

- 對於資料庫，使用 CREATE子句REVOKE可防止使用者在資料庫中建立結構描述。
- 對於結構描述，使用 CREATE子句REVOKE可防止使用者在結構描述中建立物件。若要重新命名物件，使用者必須具有 CREATE許可，並擁有要重新命名的物件。

Note

根據預設，所有使用者在結構描述上都有 PUBLIC CREATE和 USAGE許可。

TEMPORARY | TEMP

撤銷在指定資料庫中建立臨時資料表的許可。

Note

根據預設，使用者會獲得許可，透過其在PUBLIC群組中的自動成員資格來建立臨時資料表。若要移除任何使用者建立臨時資料表的許可，請從PUBLIC群組撤銷TEMP許可，然後明確授予許可，以針對特定使用者或使用者群組建立臨時資料表。

ON DATABASE db_name

撤銷指定資料庫的許可。

USAGE

撤銷特定結構描述內物件的USAGE許可，讓使用者無法存取這些物件。這些物件的特定動作必須個別撤銷（例如函數的EXECUTE許可）。

Note

根據預設，所有使用者在結構描述上都有 PUBLIC CREATE和 USAGE許可。

ON SCHEMA 結構描述名稱

撤銷指定結構描述的許可。您可以使用結構描述許可來控制資料表的建立；資料庫的CREATE許可僅控制結構描述的建立。

RESTRICT

僅撤銷使用者直接授予的許可。這是預設行為。

EXECUTE ON PROCEDURE procedure_name

撤銷特定預存程序的EXECUTE許可。由於預存程序名稱可以過載，因此您必須包含程序的引數清單。如需詳細資訊，請參閱[命名預存程序](#)。

EXECUTE ON ALL PROCEDURES IN SCHEMA 程序名稱

撤銷所參考結構描述中所有程序的指定許可。

USAGE ON LANGUAGE language_name

撤銷語言的USAGE許可。對於 Python 使用者定義函數（UDFs），請使用 plpythonu。對於 SQL UDFs，請使用 sql。若為預存程序，請使用 plpgsql。

若要建立 UDF，您必須具有 SQL或 plpythonu（Python）使用語言的許可。根據預設，USAGE ON LANGUAGE SQL 會授予 PUBLIC。不過，您必須明確LANGUAGE PLPYTHONU 地將 USAGE ON 授予特定使用者或群組。

若要撤銷的使用SQL，請先撤銷的使用PUBLIC。然後SQL僅將上的用量授予允許建立 SQL 的特定使用者或群組UDFs。下列範例SQL會從 撤銷 上的用量，PUBLIC然後將用量授與使用者群組 udf_devs。

```
revoke usage on language sql from PUBLIC;  
grant usage on language sql to group udf_devs;
```

如需詳細資訊，請參閱[UDF 安全性和許可](#)。

若要撤銷預存程序的使用，請先從 撤銷使用PUBLIC。然後僅將 p1pgsql 使用權授予獲得許可建立預存程序的特定使用者或群組。如需詳細資訊，請參閱[預存程序的安全和權限](#)。

```
FOR { ALL | COPY | UNLOAD | EXTERNAL FUNCTION | CREATE MODEL } [, ...]
```

指定撤銷許可的SQL命令。您可以指定 ALL撤銷 COPY、FUNCTION、UNLOAD EXTERNAL 和 CREATEMODEL陳述式的許可。此子句僅適用於撤銷ASSUMEROLE許可。

ALTER

撤銷使用者或使用者群組的ALTER許可，允許不擁有資料共用的使用者或使用者群組變更資料共用。需要此許可才能從資料共用新增或移除物件，或設定 屬性 PUBLICACCESSIBLE。如需詳細資訊，請參閱[ALTER DATASHARE](#)。

SHARE

撤銷使用者和使用者群組將取用者新增至資料共用的許可。撤銷此許可需要阻止特定取用者從其叢集存取資料共用。

```
ON DATASHARE datashare_name
```

授予參考資料共用的指定許可。

```
FROM 使用者名稱
```

指出失去許可的使用者。

```
FROM GROUP group_name
```

指出失去許可的使用者群組。

```
WITH GRANT OPTION
```

指出失去許可的使用者可以接著撤銷其他人的相同許可。您無法WITHGRANTOPTION撤銷 群組或的 PUBLIC。

USAGE

當同一帳戶內的消費者帳戶或命名空間遭到USAGE撤銷時，帳戶內指定的消費者帳戶或命名空間無法以唯讀方式存取資料共用和資料共用的物件。

撤銷USAGE許可會撤銷取用者對資料共用的存取權。

FROM NAMESPACE '叢集命名空間 GUID'

指出其中取用者失去資料共用許可的相同帳戶中的命名空間。命名空間使用 128 位元英數全域唯一識別碼 (GUID)。

FROM ACCOUNT 'accountnumber' 【 VIA DATA CATALOG 】

指出其中取用者失去資料共用許可的其他帳戶中的帳戶編號。指定「VIACATALOG」表示您要從 Lake Formation DATA 帳戶撤銷資料共用的使用。省略帳號代表您要從擁有叢集的帳戶撤銷。

ON DATABASE shared_database_name> 【 , ...】

在指定資料共用中建立的指定資料庫上撤銷指定使用許可。

ON SCHEMA shared_schema

在指定資料共用中建立的指定結構描述上撤銷指定許可。

FOR { SCHEMAS | TABLES | FUNCTIONS | PROCEDURES | LANGUAGES } IN

指定要撤銷權限的資料庫物件。IN 後面的參數定義撤銷權限的範圍。

CREATE MODEL

撤銷在指定資料庫中建立機器學習模型的CREATEMODEL許可。

ON MODEL model_name

撤銷特定模型的EXECUTE許可。

ACCESS CATALOG

撤銷權限，以檢視角色可存取之物件的相關中繼資料。

【 ADMIN OPTION FOR 】 { 角色 } 【 , ...】

從具有的指定使用者撤銷的角色WITHADMINOPTION。

FROM { 角色 } 【、...】

您從中撤銷指定角色的角色。

使用須知

若要進一步了解的用量備註REVOKE，請參閱 [the section called “使用須知”](#)。

範例

如需如何使用的範例REVOKE，請參閱 [the section called “範例”](#)。

使用須知

若要撤銷物件的權限，您必須符合下列條件之一：

- 身為物件擁有者。
- 身為超級使用者。
- 具有該物件和權限的授予權限。

例如，下列命令可讓使用者 HR 在員工資料表上執行SELECT命令，並為其他使用者授予和撤銷相同的權限。

```
grant select on table employees to HR with grant option;
```

HR 無法撤銷除 以外的任何操作，SELECT或員工以外任何其他資料表上的權限。

超級使用者可以存取所有物件，無論設定物件權限的 GRANT和 REVOKE命令為何。

PUBLIC 代表一律包含所有使用者的群組。根據預設，的所有成員在結構描述上PUBLIC都有 PUBLIC CREATE和 USAGE權限。若要限制任何使用者對PUBLIC結構描述的許可，您必須先撤銷PUBLIC結構描述PUBLIC上的所有許可，然後將許可授予特定使用者或群組。下列範例控制PUBLIC結構描述中的資料表建立權限。

```
revoke create on schema public from public;
```

若要從 Lake Formation 資料表撤銷權限，與資料表外部結構描述相關聯的IAM角色必須具有撤銷外部資料表權限的許可。下列範例會建立具有關聯IAM角色 的外部結構描述myGrantor。IAM 角色myGrantor具有撤銷他人許可的許可。REVOKE 命令會使用與外部結構描述myGrantor相關聯的IAM角色許可，來撤銷IAM角色 的許可myGrantee。

```
create external schema mySchema
from data catalog
database 'spectrum_db'
iam_role 'arn:aws:iam::123456789012:role/myGrantor'
create external database if not exists;
```

```
revoke select
on external table mySchema.mytable
from iam_role 'arn:aws:iam::123456789012:role/myGrantee';
```

Note

如果IAM角色在已啟用 Lake Formation AWS Glue Data Catalog 的中也有ALL許可，則不會撤銷ALL許可。只有 SELECT 許可會遭撤銷。您可以在 Lake Formation 控制台中檢視 Lake Formation 許可。

撤銷ASSUMEROLE許可的使用備註

下列用量備註適用於在 Amazon Redshift 中撤銷ASSUMEROLE權限。

只有資料庫超級使用者可以撤銷使用者和群組ASSUMEROLE的權限。超級使用者一律會保留ASSUMEROLE 權限。

若要啟用使用者和群組的 ASSUMEROLE 權限，超級使用者會在叢集上執行下列陳述式一次。在將ASSUMEROLE權限授予使用者和群組之前，超級使用者必須在叢集上執行下列陳述式一次。

```
revoke assumerole on all from public for all;
```

撤銷機器學習許可的使用須知

您無法直接授予或撤銷與 ML 函數相關的許可。ML 函數屬於 ML 模型，而且許可是透過模型控制的。相反地，您可以撤銷 ML 模型相關的許可。下列範例會示範如何從與模型 `customer_churn` 相關聯的所有使用者中撤銷執行許可。

```
REVOKE EXECUTE ON MODEL customer_churn FROM PUBLIC;
```

您也可以從 ML 模型 `customer_churn` 的使用者中撤銷所有許可。

```
REVOKE ALL on MODEL customer_churn FROM ml_user;
```

如果結構描述中有 ML 函數，則授予或撤銷 ML 函數相關的 EXECUTE 許可將會失敗，即使該 ML 函數已經具有透過 GRANT EXECUTE ON MODEL 取得的 EXECUTE 許可也是一樣。我們建議您在使用

CREATE MODEL 命令時，使用個別的結構描述，將 ML 函數本身保留在不同的結構描述中。下列範例示範如何執行此動作。

```
CREATE MODEL ml_schema.customer_churn
FROM customer_data
TARGET churn
FUNCTION ml_schema.customer_churn_prediction
IAM_ROLE default
SETTINGS (
  S3_BUCKET 'amzn-s3-demo-bucket'
);
```

範例

下列範例會從GUESTS使用者群組撤銷SALES資料表上的INSERT權限。此命令可防止的成員使用INSERT命令將資料GUESTS載入SALES資料表。

```
revoke insert on table sales from group guests;
```

下列範例會從使用者撤銷 QA_TICKIT 結構描述中SELECT所有資料表的權限fred。

```
revoke select on all tables in schema qa_tickit from fred;
```

下列範例會撤銷使用者 bobr 從檢視選取的權限。

```
revoke select on table eventview from bobr;
```

下列範例會撤銷從所有使用者在TICKIT資料庫中建立暫存資料表的權限。

```
revoke temporary on database tickit from public;
```

下列範例會從使用者撤銷cust_profile資料表的 SELECT cust_name和 cust_phone資料欄的權限user1。

```
revoke select(cust_name, cust_phone) on cust_profile from user1;
```

下列範例會從 sales_group群組撤銷 cust_name和 cust_phone資料欄上的SELECT權限，以及cust_profile資料表資料cust_contact_preference欄上的UPDATE權限。

```
revoke select(cust_name, cust_phone), update(cust_contact_preference) on cust_profile
from group sales_group;
```

下列範例顯示 ALL 關鍵字的使用，從 cust_profile sales_admin 群組撤銷資料表三欄上的 SELECT 和 UPDATE 權限。

```
revoke ALL(cust_name, cust_phone, cust_contact_preference) on cust_profile from group
sales_admin;
```

下列範例會從 user2 使用者撤銷 cust_profile_vw 檢視資料 cust_name 欄上的 SELECT 權限。

```
revoke select(cust_name) on cust_profile_vw from user2;
```

從資料共用建立的資料庫撤銷 USAGE 許可的範例

下列範例會從 13b8833d-17c6-4f16-8fe4-1a018f5ed00d 命名空間撤銷對 salesshare 資料共用的存取權。

```
REVOKE USAGE ON DATASHARE salesshare FROM NAMESPACE
'13b8833d-17c6-4f16-8fe4-1a018f5ed00d';
```

下列範例 sales_db 會從撤銷 上的 USAGE 許可 Bob。

```
REVOKE USAGE ON DATABASE sales_db FROM Bob;
```

sales_schema 來自的 之 的下列範例 REVOKE USAGE 許可 Analyst_role。

```
REVOKE USAGE ON SCHEMA sales_schema FROM ROLE Analyst_role;
```

撤銷限定範圍權限的範例

下列範例會將 Sales_db 資料庫中所有目前和未來結構描述的使用權從 Sales 角色撤銷。

```
REVOKE USAGE FOR SCHEMAS IN DATABASE Sales_db FROM ROLE Sales;
```

下列範例會撤銷從使用者 授予 Sales_db 資料庫中所有目前和未來資料表 SELECT 許可的能力 alice。會 alice 保留對 中所有資料表的存取權 Sales_db。

```
REVOKE GRANT OPTION SELECT FOR TABLES IN DATABASE Sales_db FROM alice;
```

下列範例會從使用者 撤銷Sales_schema結構描述中函數的EXECUTE許可bob。

```
REVOKE EXECUTE FOR FUNCTIONS IN SCHEMA Sales_schema FROM bob;
```

下列範例會從 Sales 角色撤銷 ShareDb 資料庫之 ShareSchema 結構描述中所有資料表的所有權限。當指定結構描述時，您也可以使用兩部分格式 database.schema 指定結構描述的資料庫。

```
REVOKE ALL FOR TABLES IN SCHEMA ShareDb.ShareSchema FROM ROLE Sales;
```

以下範例與前面的範例是相同的。您可以使用 DATABASE 關鍵字來指定結構描述資料庫，而不是使用兩部分格式。

```
REVOKE ALL FOR TABLES IN SCHEMA ShareSchema DATABASE ShareDb FROM ROLE Sales;
```

撤銷ASSUMEROLE權限的範例

以下是撤銷ASSUMEROLE權限的範例。

超級使用者必須在叢集上執行下列陳述式一次，以啟用使用者和群組的 ASSUMEROLE 權限使用：

```
revoke assumerole on all from public for all;
```

下列陳述式會撤銷ASSUMEROLE使用者 reg_user1 對於所有操作之所有角色的權限。

```
revoke assumerole on all from reg_user1 for all;
```

撤銷ROLE權限的範例

下列範例會對 sample_role2 撤銷 sample_role1。

```
CREATE ROLE sample_role2;  
GRANT ROLE sample_role1 TO ROLE sample_role2;  
REVOKE ROLE sample_role1 FROM ROLE sample_role2;
```

下列範例會撤銷 user1 的系統權限。

```
GRANT ROLE sys:DBA TO user1;  
REVOKE ROLE sys:DBA FROM user1;
```

下列範例會從 user1 撤銷 sample_role1 和 sample_role2。

```
CREATE ROLE sample_role1;
CREATE ROLE sample_role2;
GRANT ROLE sample_role1, ROLE sample_role2 TO user1;
REVOKE ROLE sample_role1, ROLE sample_role2 FROM user1;
```

下列範例ADMINOPTION會從 user1 撤銷 sample_role2。

```
GRANT ROLE sample_role2 TO user1 WITH ADMIN OPTION;
REVOKE ADMIN OPTION FOR ROLE sample_role2 FROM user1;
REVOKE ROLE sample_role2 FROM user1;
```

下列範例會從 sample_role5 撤銷 sample_role1 和 sample_role2。

```
CREATE ROLE sample_role5;
GRANT ROLE sample_role1, ROLE sample_role2 TO ROLE sample_role5;
REVOKE ROLE sample_role1, ROLE sample_role2 FROM ROLE sample_role5;
```

下列範例會將 CREATESCHEMA和 DROP SCHEMA 系統權限撤銷為 sample_role1。

```
GRANT CREATE SCHEMA, DROP SCHEMA TO ROLE sample_role1;
REVOKE CREATE SCHEMA, DROP SCHEMA FROM ROLE sample_role1;
```

ROLLBACK

停止目前交易，並捨棄該交易所做的所有更新。

此命令會執行與 [ABORT](#) 命令相同的功能。

語法

```
ROLLBACK [ WORK | TRANSACTION ]
```

參數

WORK

選用的關鍵字。預存程序內不支援使用此關鍵字。

TRANSACTION

選用的關鍵字。WORK 和 TRANSACTION是同義詞。預存程序內不支援使用這兩個關鍵字。

如需ROLLBACK在預存程序中使用的資訊，請參閱 [管理交易](#)。

範例

下列範例會建立資料表，然後開始交易，將資料插入資料表中。然後ROLLBACK，命令會復原資料插入，讓資料表保持空白。

下列命令會建立名為 MOVIE_ 的範例資料表GROSS：

```
create table movie_gross( name varchar(30), gross bigint );
```

下一個命令集會開始交易，將兩個資料列插入資料表中：

```
begin;

insert into movie_gross values ( 'Raiders of the Lost Ark', 23400000);

insert into movie_gross values ( 'Star Wars', 10000000 );
```

接著，以下命令會從資料表中選取資料，表示該資料已成功插入：

```
select * from movie_gross;
```

命令輸出會顯示這兩個資料列都已成功插入：

```
name          | gross
-----+-----
Raiders of the Lost Ark | 23400000
Star Wars      | 10000000
(2 rows)
```

現在這個命令會將資料變更轉返為交易開始時的狀態：

```
rollback;
```

從資料表選取資料現在會顯示空白資料表：

```
select * from movie_gross;
```

```
name | gross  
-----+-----  
(0 rows)
```

SELECT

從資料表、檢視及使用者定義的函數中刪除資料列。

Note

單一 SQL 陳述式的大小上限為 16 MB。

語法

```
[ WITH with_subquery [, ...] ]  
SELECT  
[ TOP number | [ ALL | DISTINCT ]  
* | expression [ AS output_name ] [, ...] ]  
[ FROM table_reference [, ...] ]  
[ WHERE condition ]  
[ [ START WITH expression ] CONNECT BY expression ]  
[ GROUP BY expression [, ...] ]  
[ HAVING condition ]  
[ QUALIFY condition ]  
[ { UNION | ALL | INTERSECT | EXCEPT | MINUS } query ]  
[ ORDER BY expression [ ASC | DESC ] ]  
[ LIMIT { number | ALL } ]  
[ OFFSET start ]
```

主題

- [WITH 子句](#)
- [SELECT 清單](#)
- [FROM 子句](#)
- [WHERE 子句](#)

- [GROUP BY 子句](#)
- [HAVING 子句](#)
- [QUALIFY 子句](#)
- [UNION、INTERSECT 和 EXCEPT](#)
- [ORDER BY 子句](#)
- [CONNECT BY 子句](#)
- [子查詢範例](#)
- [相互關聯子查詢](#)

WITH 子句

WITH 子句是選用的子句，位於查詢中的 SELECT 前面。WITH 子句會定義一個或多個 `common_table_expressions`。每個通用資料表運算式 (CTE) 都會定義一個暫存資料表，與檢視定義類似。您可以在 FROM 子句中參考這些暫存資料表。這些資料表僅會在其所屬的查詢執行時使用。WITH 子句中的每個 CTE 都會指定資料表名稱、選用的資料欄名稱清單，以及判斷值為資料表的查詢表達式 (SELECT 陳述式)。當您在定義暫存資料表的相同查詢運算式的 FROM 子句中參考暫存資料表名稱時，CTE 是遞迴的。

WITH 子句子查詢是定義資料表時較有效率的方式，可在執行單一查詢的過程中使用。在所有任何情況下，於 SELECT 陳述式的本體中使用子查詢都可產生相同的結果，但 WITH 子句子查詢對於寫入和讀取來說可能較為簡單。參考多次的 WITH 子句子查詢會盡可能最佳化為通用子表達式；也就是說，或許可以評估 WITH 子查詢一次並重複使用其結果 (請注意，通用子表達式不限於 WITH 子句中所定義者)。

語法

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
```

其中 `common_table_expression` 可以是非遞迴或遞迴的。以下是非遞迴形式：

```
CTE_table_name [ ( column_name [, ...] ) ] AS ( query )
```

以下是 `common_table_expression` 的遞迴形式：

```
CTE_table_name ( column_name [, ...] ) AS ( recursive_query )
```

參數

RECURSIVE

將查詢識別為遞迴 CTE 的關鍵字。如果 WITH 子句中定義的 `common_table_expression` 是遞迴的，則需要此關鍵字。即使 WITH 子句包含多個遞迴 CTE，您也只能指定一次 RECURSIVE 關鍵字（緊接在 WITH 關鍵字之後）。一般而言，遞迴 CTE 是具有兩個部分的 UNION ALL 子查詢。

`common_table_expression`

定義可在 [FROM 子句](#) 中參照的暫存資料表，且僅在執行該資料表所屬的查詢期間使用此暫存資料表。

`CTE_table_name`

此臨時資料表的唯一名稱會定義 WITH 子句子查詢的結果。您無法在單一 WITH 子句內使用重複的名稱。每個子查詢都必須有可在 [FROM 子句](#) 中參考的資料表名稱。

`column_name`

WITH 子句子查詢的輸出資料欄名稱清單，以逗號分隔。指定的資料欄名稱數目必須等於或少於子查詢所定義的資料欄數目。對於非遞迴的 CTE 而言，`column_name` 子句是選擇性的。對於遞迴 CTE，`column_name` 清單是必要的。

`query`

Amazon Redshift 支援的任何 SELECT 查詢。請參閱[SELECT](#)。

`recursive_query`

由兩個 SELECT 子查詢組成的 UNION ALL 查詢：

- 第一個 SELECT 子查詢沒有對相同 `CTE_table_name` 的遞迴參考。這會傳回一個結果集，也就是遞迴的初始種子。此部分稱為初始成員或種子成員。
- 第二個 SELECT 子查詢會在其 FROM 子句中參考相同的 `CTE_table_name`。這就是所謂的遞迴成員。`recursive_query` 包含一個 WHERE 條件來結束 `recursive_query`。

使用須知

您可以在下列 SQL 陳述式中使用 WITH 子句：

- SELECT
- SELECT INTO

- CREATE TABLE AS
- CREATE VIEW
- DECLARE
- EXPLAIN
- INSERT INTO...SELECT
- PREPARE
- 更UPDATE (在 WHERE 子句子查詢中。您無法在子查詢中定義遞迴 CTE。遞迴 CTE 必須位於 UPDATE 子句之前。)
- DELETE

如果查詢的 FROM 子句包含 WITH 子句，但未參考 WITH 子句定義的任何資料表，則會忽略 WITH 子句，而查詢會照常執行。

WITH 子句子查詢定義的資料表只能在 WITH 子句開始的 SELECT 查詢範圍內參考。例如，您可以在 SELECT 清單、WHERE 子句或 HAVING 子句中，子查詢的 FROM 子句內參考這類資料表。您無法在子查詢中使用 WITH 子句，並於主查詢或其他子查詢的 FROM 子句內參考其資料表。此查詢模式會針對 WITH 子句資料表產生 `relation table_name doesn't exist` 形式的錯誤訊息。

您無法在 WITH 子句子查詢內指定另一個 WITH 子句。

您無法對 WITH 子句子查詢定義的資料表進行向前參考。例如，以下查詢會傳回錯誤訊息，因為資料表 W1 的定義中有對資料表 W2 的向前參考：

```
with w1 as (select * from w2), w2 as (select * from w1)
select * from sales;
ERROR:  relation "w2" does not exist
```

WITH 子句子查詢不一定包含 SELECT INTO 陳述式；不過您可以在 SELECT INTO 陳述式中使用 WITH 子句。

遞迴一般資料表表達式

遞迴 common table expression (CTE) 是參考其本身的 CTE。遞迴 CTE 在查詢階層式資料時非常有用，例如顯示員工與管理者之間責任關係的組織圖。請參閱[範例：遞迴 CTE](#)。

另一個常見的用途是多層級材料表，像是產品由許多元件組成，而每個元件本身也包含其他元件或次要組件時。

請務必在遞迴查詢的第二個 SELECT 子查詢中加入 WHERE 子句，以限制遞迴的深度。如需範例，請參閱範例：[遞迴 CTE](#)。否則，會發生類似以下內容的錯誤：

- Recursive CTE out of working buffers.
- Exceeded recursive CTE max rows limit, please add correct CTE termination predicates or change the max_recursion_rows parameter.

Note

`max_recursion_rows` 是一個參數，可設定遞迴 CTE 可傳回的最大資料列數，以防止無限遞迴迴圈。我們建議您不要將此值變更為大於預設值的值。這樣可以防止查詢中的無限遞迴佔用叢集中過多空間的問題。

您可以指定遞迴 CTE 結果的排序順序和限制。您可以在遞迴 CTE 的最終結果中包含群組依據和相異選項。

您無法在子查詢內指定 WITH RECURSIVE 子句。`recursive_query` 成員不能包含排序依據或限制子句。

範例

下列範例顯示包含 WITH 子句的最簡單查詢案例。名為 VENUECOPY 的 WITH 查詢會從 VENUE 資料表選取所有資料列。主查詢會接著從 VENUECOPY 選取所有資料列。VENUECOPY 資料表僅在此查詢期間存在。

```
with venuecopy as (select * from venue)
select * from venuecopy order by 1 limit 10;
```

venueid	venuename	venuecity	venuestate	venueSeats
1	Toyota Park	Bridgeview	IL	0
2	Columbus Crew Stadium	Columbus	OH	0
3	RFK Stadium	Washington	DC	0
4	CommunityAmerica Ballpark	Kansas City	KS	0
5	Gillette Stadium	Foxborough	MA	68756
6	New York Giants Stadium	East Rutherford	NJ	80242
7	BMO Field	Toronto	ON	0
8	The Home Depot Center	Carson	CA	0
9	Dick's Sporting Goods Park	Commerce City	CO	0

```
v      10 | Pizza Hut Park      | Frisco      | TX      |      0
(10 rows)
```

下列範例顯示 WITH 子句，它會產生兩個資料表，分別名為 VENUE_SALES 和 TOP_VENUES。第二個 WITH 查詢資料表會從第一個資料表選取。接著主查詢區塊的 WHERE 子句會包含限制 TOP_VENUES 資料表的子查詢。

```
with venue_sales as
(select venuename, venuecity, sum(pricepaid) as venue_name_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
group by venue_name_sales, venuecity),

top_venues as
(select venue_name_sales
from venue_sales
where venue_name_sales > 800000)

select venue_name_sales, venuecity, venuestate,
sum(qtysold) as venue_qty,
sum(pricepaid) as venue_sales
from sales, venue, event
where venue.venueid=event.venueid and event.eventid=sales.eventid
and venue_name_sales in(select venue_name_sales from top_venues)
group by venue_name_sales, venuecity, venuestate
order by venue_name_sales;
```

venue_name_sales	venuecity	venuestate	venue_qty	venue_sales
August Wilson Theatre	New York City	NY	3187	1032156.00
Biltmore Theatre	New York City	NY	2629	828981.00
Charles Playhouse	Boston	MA	2502	857031.00
Ethel Barrymore Theatre	New York City	NY	2828	891172.00
Eugene O'Neill Theatre	New York City	NY	2488	828950.00
Greek Theatre	Los Angeles	CA	2445	838918.00
Helen Hayes Theatre	New York City	NY	2948	978765.00
Hilton Theatre	New York City	NY	2999	885686.00
Imperial Theatre	New York City	NY	2702	877993.00
Lunt-Fontanne Theatre	New York City	NY	3326	1115182.00
Majestic Theatre	New York City	NY	2549	894275.00
Nederlander Theatre	New York City	NY	2934	936312.00
Pasadena Playhouse	Pasadena	CA	2739	820435.00

```
Winter Garden Theatre | New York City | NY | 2838 | 939257.00
(14 rows)
```

以下兩個範例將示範根據 WITH 子句子查詢的資料表參考範圍規則。第一個查詢會執行，但第二個會失敗，並產生預期的錯誤。第一個查詢會在主查詢的 SELECT 清單內包含 WITH 子句子查詢。WITH 子句定義的資料表 (HOLIDAYS) 會在 SELECT 清單中子查詢的 FROM 子句中參考：

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t'))
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join date on sales.dateid=date.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;
```

caldate	daysales	dec25sales
2008-12-25	70402.00	70402.00
2008-12-31	12678.00	70402.00

(2 rows)

第二個查詢會失敗，因為它會嘗試參考主查詢以及 SELECT 清單子查詢中的 HOLIDAYS 資料表。而主查詢參考超出範圍。

```
select caldate, sum(pricepaid) as daysales,
(with holidays as (select * from date where holiday ='t'))
select sum(pricepaid)
from sales join holidays on sales.dateid=holidays.dateid
where caldate='2008-12-25') as dec25sales
from sales join holidays on sales.dateid=holidays.dateid
where caldate in('2008-12-25','2008-12-31')
group by caldate
order by caldate;
```

ERROR: relation "holidays" does not exist

範例：遞迴 CTE

以下是遞迴 CTE 的範例，此範例會傳回直接或間接向 John 報告的員工。遞迴查詢包含 WHERE 子句，可將遞迴深度限制為少於 4 個層級。

```
--create and populate the sample table
create table employee (
  id int,
  name varchar (20),
  manager_id int
);

insert into employee(id, name, manager_id) values
(100, 'Carlos', null),
(101, 'John', 100),
(102, 'Jorge', 101),
(103, 'Kwaku', 101),
(110, 'Liu', 101),
(106, 'Mateo', 102),
(110, 'Nikki', 103),
(104, 'Paulo', 103),
(105, 'Richard', 103),
(120, 'Saanvi', 104),
(200, 'Shirley', 104),
(201, 'Sofía', 102),
(205, 'Zhang', 104);

--run the recursive query
with recursive john_org(id, name, manager_id, level) as
( select id, name, manager_id, 1 as level
  from employee
  where name = 'John'
  union all
  select e.id, e.name, e.manager_id, level + 1 as next_level
  from employee e, john_org j
  where e.manager_id = j.id and level < 4
  )
select distinct id, name, manager_id from john_org order by manager_id;
```

以下為查詢結果。

id	name	manager_id
101	John	100
102	Jorge	101
103	Kwaku	101
110	Liu	101
201	Sofía	102

106	Mateo	102
110	Nikki	103
104	Paulo	103
105	Richard	103
120	Saanvi	104
200	Shirley	104
205	Zhang	104

以下是 John 的部門組織結構圖。

SELECT 清單

主題

- [語法](#)
- [參數](#)
- [使用須知](#)
- [範例](#)

SELECT 清單會指出您要查詢傳回的資料欄、函數及表達式的名稱。清單查詢的輸出。

如需 SQL 函數的相關資訊，請參閱 [SQL 函數參考](#)。如需運算式的相關資訊，請參閱 [條件式運算式](#)。

語法

```
SELECT
[ TOP number ]
[ ALL | DISTINCT ] * | expression [ AS column_alias ] [, ...]
```

參數

TOP number

TOP 會採用正整數做為其引數，此引數定義傳回至用戶端的資料列數。TOP 子句的行為與 LIMIT 子句的行為相同。傳回的資料列數是固定的，但資料列集卻不是。若要傳回一致的資料列集，請搭配 ORDER BY 子句使用 TOP 或 LIMIT。

ALL

您未指定 DISTINCT 時，用來定義預設行為的備援關鍵字。SELECT ALL * 的意義與 SELECT * 相同 (選取所有資料欄的所有資料列並保留重複項目)。

DISTINCT

此選項會根據一個或多個資料欄中相符的值，從結果集中消除重複的資料列。

Note

如果您的應用程式允許無效的外部索引鍵或主索引鍵，可能會導致查詢傳回不正確的結果。例如，如果主索引鍵資料行不包含所有唯一值，則 `SELECT DISTINCT` 查詢可能會傳回重複的資料列。如需詳細資訊，請參閱[定義表格條件約束](#)。

* (星號)

傳回資料表的整個內容 (所有資料欄和所有資料列)。

運算式

表達式是由查詢所參考資料表中的一個或多個資料欄構成。表達式可包含 SQL 函數。例如：

```
avg(datediff(day, listtime, saletime))
```

AS column_alias

資料欄的暫時名稱，會在最終結果集中使用。AS 關鍵字為選用。例如：

```
avg(datediff(day, listtime, saletime)) as avgwait
```

若您沒有為表達式指定非簡單資料欄名稱的別名，結果集將會套用預設名稱至該資料欄。

Note

別名在目標清單中定義之後立即直接辨識。您可以在相同目標清單中後續定義的其他表達式後面使用別名。下列的範例示範了這一點。

```
select clicks / impressions as probability, round(100 * probability, 1) as percentage from raw_data;
```

側邊別名參考的好處在於，您在相同目標清單中建構更複雜的表達式時，不需要重複有別名的表達式。當 Amazon Redshift 剖析此類型參考時，會直接內嵌先前定義的別名。若 FROM

子句中定義了與先前具有別名之表達式同名的資料欄，則 FROM 子句中的資料欄優先順序較高。例如，在上方查詢中，若 raw_data 資料表中有名為 'probability' 的資料欄，目標清單中第二個表達式內的 'probability' 會參考該資料欄，而不是 'probability' 這個別名。

使用須知

TOP 是 SQL 延伸模組；它提供了 LIMIT 行為的替代方式。您無法在相同查詢中同時使用 TOP 和 LIMIT。

範例

下列範例會從 SALES 資料表中傳回 10 個資料列。雖然查詢使用 TOP 子句，但仍然會傳回無法預測的資料列集，因為沒有指定 ORDER BY 子句。

```
select top 10 *
from sales;
```

下列查詢具同等功能，但使用 LIMIT 子句而非 TOP 子句：

```
select *
from sales
limit 10;
```

下列範例會使用 TOP 子句從 SALES 資料表傳回前 10 列，並依 QTYSOLD 資料欄遞減排序。

```
select top 10 qtysold, sellerid
from sales
order by qtysold desc, sellerid;
```

```
qtysold | sellerid
-----+-----
8 |      518
8 |      520
8 |      574
8 |      718
8 |      868
8 |     2663
8 |     3396
8 |     3726
8 |     5250
```

```
8 |      6216
(10 rows)
```

下列範例會從 SALES 資料表傳回前兩個 QTYSOLD 和 SELLERID 值，並依 QTYSOLD 資料欄排序：

```
select top 2 qtysold, sellerid
from sales
order by qtysold desc, sellerid;
```

```
qtysold | sellerid
-----+-----
8 |      518
8 |      520
(2 rows)
```

下列範例顯示 CATEGORY 資料表中不同類別群組的清單：

```
select distinct catgroup from category
order by 1;
```

```
catgroup
-----
Concerts
Shows
Sports
(3 rows)
```

```
--the same query, run without distinct
select catgroup from category
order by 1;
```

```
catgroup
-----
Concerts
Concerts
Concerts
Shows
Shows
Shows
Sports
Sports
Sports
Sports
```

```
Sports
(11 rows)
```

下列範例會傳回 2008 年 12 月的不同週數組。如果沒有 DISTINCT 子句，陳述式會傳回 31 個資料列，或是針對每月的每一天傳回 1 列。

```
select distinct week, month, year
from date
where month='DEC' and year=2008
order by 1, 2, 3;
```

```
week | month | year
-----+-----+-----
49 | DEC   | 2008
50 | DEC   | 2008
51 | DEC   | 2008
52 | DEC   | 2008
53 | DEC   | 2008
(5 rows)
```

FROM 子句

查詢中的 FROM 子句列出資料表參考 (資料表、檢視和子查詢)，此為選取資料的來源位置。若列出多個資料表參考，則必須在 FROM 子句或 WHERE 子句中使用適當的語法聯結資料表。若未指定聯結條件，則系統會將查詢當做交叉聯結 (笛卡兒乘積) 處理。

主題

- [語法](#)
- [參數](#)
- [使用須知](#)
- [PIVOT 和 UNPIVOT 範例](#)
- [JOIN 範例](#)

語法

```
FROM table_reference [, ...]
```

其中 *table_reference* 是下列其中一項：

```

with_subquery_table_name [ table_alias ]
table_name [ * ] [ table_alias ]
( subquery ) [ table_alias ]
table_reference [ NATURAL ] join_type table_reference
  [ ON join_condition | USING ( join_column [, ...] ) ]
table_reference PIVOT (
  aggregate(expr) [ [ AS ] aggregate_alias ]
  FOR column_name IN ( expression [ AS ] in_alias [, ...] )
) [ table_alias ]
table_reference UNPIVOT [ INCLUDE NULLS | EXCLUDE NULLS ] (
  value_column_name
  FOR name_column_name IN ( column_reference [ [ AS ]
  in_alias ] [, ...] )
) [ table_alias ]
UNPIVOT expression AS value_alias [ AT attribute_alias ]

```

可選 `table_alias` 可用於為表和複雜表引用提供臨時名稱，如果需要，它們的列也可以使用，如下所示：

```
[ AS ] alias [ ( column_alias [, ...] ) ]
```

參數

`with_subquery_table_name`

[WITH 子句](#) 中子查詢所定義的資料表。

`table_name`

資料表或檢視的名稱。

`alias`

資料表或檢視的暫時替代名稱。必須為衍生自子查詢的資料表提供別名。在其他資料表參考中，別名是選用的。AS 關鍵字一律為選用。資料表別名提供了方便在查詢的其他部分中識別資料表的捷徑，例如 WHERE 子句。例如：

```
select * from sales s, listing l
where s.listid=l.listid
```

`column_alias`

資料表或檢視中資料欄的暫時替代名稱。

subquery

判斷值為資料表的查詢表達式。資料表只會在查詢期間存在，通常會為其命名或提供別名。不過，不需要別名。您也可以為衍生自子查詢的資料表定義資料欄名稱。當您想要將子查詢的結果與其他資料表聯結時，以及您想要在查詢中的其他位置選取或限制這些資料欄時，為資料欄指定別名就很重要。

子查詢可包含 ORDER BY 子句，但是，若未指定 LIMIT 或 OFFSET 子句，則此子句不一定有作用。

NATURAL

定義聯結，此聯結會自動使用兩個資料表中所有同名資料欄的配對做為聯結資料欄。不需要明確的聯結條件。例如，若 CATEGORY 和 EVENT 資料表都有名為 CATID 的資料欄，則這兩個資料表的 natural 聯結會是透過其 CATID 資料欄的聯結。

Note

若已指定 NATURAL 聯結，但是要聯結的資料表中並沒有同名的資料欄配對，則查詢會預設為交叉聯結。

join_type

指定下列其中一種聯結類型：

- [INNER] JOIN
- LEFT [OUTER] JOIN
- RIGHT [OUTER] JOIN
- FULL [OUTER] JOIN
- CROSS JOIN

交叉聯結是沒有限定的聯結，會傳回兩個資料表的笛卡兒乘積。

內部和外部聯結為限定聯結。它們會以隱含方式 (在 natural 聯結中)、在 FROM 子句中使用 ON 或 USING 語法，或使用 WHERE 子句條件限定。

內部聯結只會根據聯結條件或聯結資料欄清單傳回相符的資料列。外部聯結會傳回對等內部聯結傳回的所有資料列，加上「左側」資料表和/或「右側」資料表的不相符資料列。左側資料表是最先

列出的資料表，右側資料表是其次列出的資料表。不相符的資料列包含要填入輸出資料欄中空處的 NULL 值。

ON join_condition

聯結規格的類型，其中聯結資料欄會做為條件陳述，後面接著 ON 關鍵字。例如：

```
sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
```

USING (join_column [, ...])

聯結規格的類型，其中聯結資料欄會在括號內列出。若指定了多個聯結資料欄，則會以逗號分隔。USING 關鍵字必須放在清單前面。例如：

```
sales join listing
using (listid,eventid)
```

PIVOT

將輸出從資料列旋轉為資料欄，以便以易於閱讀的格式表示資料表資料。輸出會橫跨多個資料欄表示。PIVOT 類似於具使用彙總的 GROUP BY 查詢，並且會使用彙總運算式來指定輸出格式。但是，與 GROUP BY 相反，其結果會以資料欄傳回，而不是資料列。

如需示範如何使用 PIVOT 和 UNPIVOT 進行查詢的範例，請參閱 [PIVOT 和 UNPIVOT 範例](#)。

UNPIVOT

使用 UNPIVOT 將資料欄旋轉為資料列 — 運算子會將輸入資料表或查詢結果的結果欄轉換為列，以便於讀取輸出。UNPIVOT 會將其輸入資料欄的資料合併為兩個結果資料欄：名稱資料欄和值資料欄。名稱資料欄包含來自輸入中做為資料列項目的資料欄名稱。值資料欄包含來自輸入資料欄的值，例如彙總的結果。例如，各種類別中的項目計數。

使用 UNPIVOT (SUPER) 取消樞紐的物件 — 您可以執行取消樞紐的物件，其中運算式是參照另一個 FROM 子句項目的 SUPER 運算式。如需詳細資訊，請參閱 [物件取消樞紐](#)。它也有示範如何查詢半結構化資料的範例，例如 JSON 格式的資料。

使用須知

聯結資料欄必須採用可比較的資料類型。

NATURAL 或 USING 聯結只會針對中繼結果集內每個聯結資料欄配對保留一個。

使用 ON 語法的聯結則會保留其中繼結果集內的兩個聯結資料欄。

另請參閱[WITH 子句](#)。

PIVOT 和 UNPIVOT 範例

PIVOT 和 UNPIVOT 是 FROM 子句中的參數，這兩個參數會分別將查詢輸出從資料列旋轉為資料欄，以及將資料欄旋轉為資料列。這會以易於閱讀的格式呈現資料表式查詢結果。下列範例會使用測試資料和查詢來顯示如何使用這些參數。

如需這些參數與其他參數的相關資訊，請參閱[FROM 子句](#)。

PIVOT 範例

設定範例資料表和資料，並用其來執行後續的範例查詢。

```
CREATE TABLE part (  
    partname varchar,  
    manufacturer varchar,  
    quality int,  
    price decimal(12, 2)  
);  
  
INSERT INTO part VALUES ('prop', 'local parts co', 2, 10.00);  
INSERT INTO part VALUES ('prop', 'big parts co', NULL, 9.00);  
INSERT INTO part VALUES ('prop', 'small parts co', 1, 12.00);  
  
INSERT INTO part VALUES ('rudder', 'local parts co', 1, 2.50);  
INSERT INTO part VALUES ('rudder', 'big parts co', 2, 3.75);  
INSERT INTO part VALUES ('rudder', 'small parts co', NULL, 1.90);  
  
INSERT INTO part VALUES ('wing', 'local parts co', NULL, 7.50);  
INSERT INTO part VALUES ('wing', 'big parts co', 1, 15.20);  
INSERT INTO part VALUES ('wing', 'small parts co', NULL, 11.80);
```

partname 上的 PIVOT 搭配 price 上的 AVG 彙總。

```
SELECT *  
FROM (SELECT partname, price FROM part) PIVOT (  
    AVG(price) FOR partname IN ('prop', 'rudder', 'wing')  
);
```


此查詢結果為下列輸出。

```
prop   | rudder | wing
-----+-----+-----
10.33  | 2.71   | 11.50
```

在上一個範例中，結果會轉換為資料欄。下列範例會顯示以資料列 (不是資料欄) 傳回平均價格的 GROUP BY 查詢。

```
SELECT partname, avg(price)
FROM (SELECT partname, price FROM part)
WHERE partname IN ('prop', 'rudder', 'wing')
GROUP BY partname;
```

此查詢結果為下列輸出。

```
partname | avg
-----+-----
prop     | 10.33
rudder   | 2.71
wing     | 11.50
```

以 manufacturer 作為隱含資料欄的 PIVOT 範例。

```
SELECT *
FROM (SELECT quality, manufacturer FROM part) PIVOT (
    count(*) FOR quality IN (1, 2, NULL)
);
```

此查詢結果為下列輸出。

```
manufacturer | 1 | 2 | null
-----+---+---+-----
local parts co | 1 | 1 | 1
big parts co   | 1 | 1 | 1
small parts co | 1 | 0 | 2
```

未在 PIVOT 定義中參考的輸入資料表資料欄會隱含地新增至結果資料表。上一個範例中的 manufacturer 資料欄就是這種情況。此範例也會顯示 NULL 是 IN 運算子的有效值。

PIVOT 在上述範例中會傳回類似下列查詢的資訊，其中包括 GROUP BY。不同之處在於 PIVOT 傳回資料欄 2 的值 0 和製造商 small parts co。GROUP BY 查詢不包含對應的資料列。在大多數情況下，如果資料列沒有給定資料欄的輸入資料，PIVOT 會插入 NULL。但是，計數彙總不會傳回 NULL，0 是預設值。

```
SELECT manufacturer, quality, count(*)
FROM (SELECT quality, manufacturer FROM part)
WHERE quality IN (1, 2) OR quality IS NULL
GROUP BY manufacturer, quality
ORDER BY manufacturer;
```

此查詢結果為下列輸出。

manufacturer	quality	count
big parts co		1
big parts co	2	1
big parts co	1	1
local parts co	2	1
local parts co	1	1
local parts co		1
small parts co	1	1
small parts co		2

PIVOT 運算子會接受彙總運算式上及 IN 運算子每個值上的選用別名。使用別名來自訂資料欄名稱。如果沒有彙總別名，則只會使用 IN 清單別名。否則，彙總別名會附加至資料欄名稱，並加上底線來分隔名稱。

```
SELECT *
FROM (SELECT quality, manufacturer FROM part) PIVOT (
    count(*) AS count FOR quality IN (1 AS high, 2 AS low, NULL AS na)
);
```

此查詢結果為下列輸出。

manufacturer	high_count	low_count	na_count
local parts co	1	1	1
big parts co	1	1	1
small parts co	1	0	2

設定下列範例資料表和資料，並用其來執行後續的範例查詢。資料表示飯店集合的預訂日期。

```
CREATE TABLE bookings (  
    booking_id int,  
    hotel_code char(8),  
    booking_date date,  
    price decimal(12, 2)  
);  
  
INSERT INTO bookings VALUES (1, 'FOREST_L', '02/01/2023', 75.12);  
INSERT INTO bookings VALUES (2, 'FOREST_L', '02/02/2023', 75.00);  
INSERT INTO bookings VALUES (3, 'FOREST_L', '02/04/2023', 85.54);  
  
INSERT INTO bookings VALUES (4, 'FOREST_L', '02/08/2023', 75.00);  
INSERT INTO bookings VALUES (5, 'FOREST_L', '02/11/2023', 75.00);  
INSERT INTO bookings VALUES (6, 'FOREST_L', '02/14/2023', 90.00);  
  
INSERT INTO bookings VALUES (7, 'FOREST_L', '02/21/2023', 60.00);  
INSERT INTO bookings VALUES (8, 'FOREST_L', '02/22/2023', 85.00);  
INSERT INTO bookings VALUES (9, 'FOREST_L', '02/27/2023', 90.00);  
  
INSERT INTO bookings VALUES (10, 'DESERT_S', '02/01/2023', 98.00);  
INSERT INTO bookings VALUES (11, 'DESERT_S', '02/02/2023', 75.00);  
INSERT INTO bookings VALUES (12, 'DESERT_S', '02/04/2023', 85.00);  
  
INSERT INTO bookings VALUES (13, 'DESERT_S', '02/05/2023', 75.00);  
INSERT INTO bookings VALUES (14, 'DESERT_S', '02/06/2023', 34.00);  
INSERT INTO bookings VALUES (15, 'DESERT_S', '02/09/2023', 85.00);  
  
INSERT INTO bookings VALUES (16, 'DESERT_S', '02/12/2023', 23.00);  
INSERT INTO bookings VALUES (17, 'DESERT_S', '02/13/2023', 76.00);  
INSERT INTO bookings VALUES (18, 'DESERT_S', '02/14/2023', 85.00);  
  
INSERT INTO bookings VALUES (19, 'OCEAN_WV', '02/01/2023', 98.00);  
INSERT INTO bookings VALUES (20, 'OCEAN_WV', '02/02/2023', 75.00);  
INSERT INTO bookings VALUES (21, 'OCEAN_WV', '02/04/2023', 85.00);  
  
INSERT INTO bookings VALUES (22, 'OCEAN_WV', '02/06/2023', 75.00);  
INSERT INTO bookings VALUES (23, 'OCEAN_WV', '02/09/2023', 34.00);  
INSERT INTO bookings VALUES (24, 'OCEAN_WV', '02/12/2023', 85.00);  
  
INSERT INTO bookings VALUES (25, 'OCEAN_WV', '02/13/2023', 23.00);  
INSERT INTO bookings VALUES (26, 'OCEAN_WV', '02/14/2023', 76.00);  
INSERT INTO bookings VALUES (27, 'OCEAN_WV', '02/16/2023', 85.00);
```

```

INSERT INTO bookings VALUES (28, 'CITY_BLD', '02/01/2023', 98.00);
INSERT INTO bookings VALUES (29, 'CITY_BLD', '02/02/2023', 75.00);
INSERT INTO bookings VALUES (30, 'CITY_BLD', '02/04/2023', 85.00);

INSERT INTO bookings VALUES (31, 'CITY_BLD', '02/12/2023', 75.00);
INSERT INTO bookings VALUES (32, 'CITY_BLD', '02/13/2023', 34.00);
INSERT INTO bookings VALUES (33, 'CITY_BLD', '02/17/2023', 85.00);

INSERT INTO bookings VALUES (34, 'CITY_BLD', '02/22/2023', 23.00);
INSERT INTO bookings VALUES (35, 'CITY_BLD', '02/23/2023', 76.00);
INSERT INTO bookings VALUES (36, 'CITY_BLD', '02/24/2023', 85.00);

```

在此查詢範例中，預訂記錄會計算為每週的總數。每週的結束日期會變成資料欄名稱。

```

SELECT * FROM
  (SELECT
    booking_id,
    (date_trunc('week', booking_date::date) + '5 days'::interval)::date as enddate,
    hotel_code AS "hotel code"
  FROM bookings
 ) PIVOT (
   count(booking_id) FOR enddate IN ('2023-02-04', '2023-02-11', '2023-02-18')
 );

```

此查詢結果為下列輸出。

hotel code	2023-02-04	2023-02-11	2023-02-18
FOREST_L	3	2	1
DESERT_S	4	3	2
OCEAN_WV	3	3	3
CITY_BLD	3	1	2

Amazon Redshift 不支援使用 CROSSTAB 在多個資料欄上轉移。但是，您可以使用類似於 PIVOT 的彙總方式將資料列資料變更為資料欄，並使用類似於以下內容的查詢。這會使用與前一個範例相同的預訂範例資料。

```

SELECT
  booking_date,
  MAX(CASE WHEN hotel_code = 'FOREST_L' THEN 'forest is booked' ELSE '' END) AS
  FOREST_L,

```

```

MAX(CASE WHEN hotel_code = 'DESERT_S' THEN 'desert is booked' ELSE '' END) AS
DESERT_S,
MAX(CASE WHEN hotel_code = 'OCEAN_WV' THEN 'ocean is booked' ELSE '' END) AS
OCEAN_WV
FROM bookings
GROUP BY booking_date
ORDER BY booking_date asc;

```

範例查詢結果會顯示預訂日期，並列在指出已預訂飯店的短語旁邊。

booking_date	forest_l	desert_s	ocean_wv
2023-02-01	forest is booked	desert is booked	ocean is booked
2023-02-02	forest is booked	desert is booked	ocean is booked
2023-02-04	forest is booked	desert is booked	ocean is booked
2023-02-05		desert is booked	
2023-02-06		desert is booked	

以下是 PIVOT 的使用須知：

- PIVOT 可套用至資料表、子查詢和通用資料表運算式 (CTE)。PIVOT 無法套用至任何 JOIN 運算式、遞迴 CTE、PIVOT 或 UNPIVOT 運算式。SUPER 非巢狀運算式和 Redshift Spectrum 巢狀資料表也不支援。
- PIVOT 支援 COUNT、SUM、MIN、MAX 和 AVG 彙總函數。
- PIVOT 彙總運算式必須是受支援彙總函式的呼叫。不支援彙總頂端的複雜運算式。彙總引數不能包含對 PIVOT 輸入資料表以外資料表的參考。也不支援父查詢的相關參考。彙總參數可以包含子查詢。這些可以在內部或 PIVOT 輸入資料表上相互關聯。
- PIVOT IN 清單值不能是資料欄參考或子查詢。每個值必須是與 FOR 資料欄參考相容的類型。
- 如果 IN 清單值沒有別名，則 PIVOT 會產生預設資料欄名稱。對於常數 IN 值，如 'abc' 或 5，預設資料欄名稱是常值本身。對於任何複雜的運算式，資料欄名稱都是標準的 Amazon Redshift 預設名稱，例如 ?column?。

UNPIVOT 範例

設定範例資料，並用其來執行後續的範例。

```

CREATE TABLE count_by_color (quality varchar, red int, green int, blue int);

INSERT INTO count_by_color VALUES ('high', 15, 20, 7);

```

```
INSERT INTO count_by_color VALUES ('normal', 35, NULL, 40);
INSERT INTO count_by_color VALUES ('low', 10, 23, NULL);
```

UNPIVOT 在輸入欄紅色、綠色和藍色上。

```
SELECT *
FROM (SELECT red, green, blue FROM count_by_color) UNPIVOT (
    cnt FOR color IN (red, green, blue)
);
```

此查詢結果為下列輸出。

```
color | cnt
-----+-----
red   | 15
red   | 35
red   | 10
green | 20
green | 23
blue  | 7
blue  | 40
```

依預設，輸入資料欄中的 NULL 值會略過，且不會產生結果資料列。

下列範例會顯示包括 INCLUDE NULLS 的 UNPIVOT。

```
SELECT *
FROM (
    SELECT red, green, blue
    FROM count_by_color
) UNPIVOT INCLUDE NULLS (
    cnt FOR color IN (red, green, blue)
);
```

以下為其輸出。

```
color | cnt
-----+-----
red   | 15
red   | 35
red   | 10
green | 20
```

```
green |
green | 23
blue  | 7
blue  | 40
blue  |
```

如果設定 INCLUDING NULLS 參數，NULL 輸入值會產生結果列。

以 quality 作為隱含資料欄的 The following query shows UNPIVOT。

```
SELECT *
FROM count_by_color UNPIVOT (
    cnt FOR color IN (red, green, blue)
);
```

此查詢結果為下列輸出。

```
quality | color | cnt
-----+-----+-----
high    | red   | 15
normal  | red   | 35
low     | red   | 10
high    | green | 20
low     | green | 23
high    | blue  | 7
normal  | blue  | 40
```

未在 UNPIVOT 定義中參考的輸入資料表資料欄會隱含地新增至結果資料表。在範例中，quality 資料欄就是這種情況。

下列範例顯示 IN 清單中具有值別名的 UNPIVOT。

```
SELECT *
FROM count_by_color UNPIVOT (
    cnt FOR color IN (red AS r, green AS g, blue AS b)
);
```

上述查詢結果為下列輸出。

```
quality | color | cnt
-----+-----+-----
high    | r     | 15
```

normal	r	35
low	r	10
high	g	20
low	g	23
high	b	7
normal	b	40

UNPIVOT 運算符會接受每個 IN 清單值上的選用別名。每個別名都會提供每個 value 資料欄中的資料定義。

以下是 UNPIVOT 的使用須知。

- UNPIVOT 可套用至資料表、子查詢和通用資料表運算式 (CTE)。UNPIVOT 無法套用至任何 JOIN 運算式、遞迴 CTE、PIVOT 或 UNPIVOT 運算式。SUPER 非巢狀運算式和 Redshift Spectrum 巢狀資料表也不支援。
- UNPIVOT IN 清單必須只包含輸入資料表資料欄參考。IN 清單欄必須具有與之相容的通用類型。UNPIVOT 值資料欄具有這種通用類型。UNPIVOT 名稱資料欄的類型為 VARCHAR。
- 如果 IN 清單值沒有別名，UNPIVOT 會使用資料欄名稱做為預設值。

JOIN 範例

SQL JOIN 子句用於根據通用欄位，結合兩個或多個資料表中的資料。結果可能會或可能不會改變，具體取決於指定的聯結方法。如需 JOIN 子句語法的相關資訊，請參閱 [參數](#)。

下列範例會使用 TICKET 範例資料中的資料。如需資料庫結構描述的相關資訊，請參閱 [範本資料庫](#)。若要了解如何載入範例資料，請參閱 [Amazon Redshift 入門指南中的載入資料](#)。

下列查詢是 LISTING 資料表和 SALES 資料表之間的內部聯結 (沒有 JOIN 關鍵字)，其中 LISTING 資料表中的 LISTID 是介於 1 和 5 之間。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示 LISTID 1、4 和 5 符合條件。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing, sales
where listing.listid = sales.listid
and listing.listid between 1 and 5
group by 1
order by 1;
```

listid	price	comm
1	728.00	109.20

4	76.00	11.40
5	525.00	78.75

下列查詢是左側外部聯結。未在另一個資料表中找到相符項目時，左和右外部聯結會保留來自其中一個聯結資料表的值。左側和右側資料表分別是語法中最先和其次列出的資料表。NULL 值會用來填入結果集中的「空處」。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示，LISTID 2 和 3 並未產生任何銷售。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing left outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;
```

listid	price	comm
1	728.00	109.20
2	NULL	NULL
3	NULL	NULL
4	76.00	11.40
5	525.00	78.75

下列查詢是右側外部聯結。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示 LISTID 1、4 和 5 符合條件。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing right outer join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;
```

listid	price	comm
1	728.00	109.20
4	76.00	11.40
5	525.00	78.75

下列查詢是完全聯結。未在另一個資料表中找到相符項目時，完全聯結會保留來自聯結資料表的值。左側和右側資料表分別是語法中最先和其次列出的資料表。NULL 值會用來填入結果集中的「空處」。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。結果顯示，LISTID 2 和 3 並未產生任何銷售。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
group by 1
order by 1;
```

listid	price	comm
1	728.00	109.20
2	NULL	NULL
3	NULL	NULL
4	76.00	11.40
5	525.00	78.75

下列查詢是完全聯結。此查詢會比對 LISTING 資料表 (左側資料表) 和 SALES 資料表 (右側資料表) 中 LISTID 資料欄的值。只有不會產生任何銷售 (ListID 2 和 3) 的資料列才會顯示在結果中。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from listing full join sales on sales.listid = listing.listid
where listing.listid between 1 and 5
and (listing.listid IS NULL or sales.listid IS NULL)
group by 1
order by 1;
```

listid	price	comm
2	NULL	NULL
3	NULL	NULL

下列範例是一個內部聯結搭配 ON 子句。在此情況下，不會傳回 NULL 資料列。

```
select listing.listid, sum(pricepaid) as price, sum(commission) as comm
from sales join listing
on sales.listid=listing.listid and sales.eventid=listing.eventid
where listing.listid between 1 and 5
group by 1
order by 1;
```

listid	price	comm
1	728.00	109.20
4	76.00	11.40

5 | 525.00 | 78.75

下列查詢是 LISTING 資料表和 SALES 資料表的交叉聯結或笛卡爾聯結，並且使用述詞來限制結果。此查詢會在 SALES 資料表和 LISTING 資料表中比對 LISTID 資料欄值，以找出兩個資料表中的 LISTID 1、2、3、4 和 5。結果顯示 20 個符合條件的資料列。

```
select sales.listid as sales_listid, listing.listid as listing_listid
from sales cross join listing
where sales.listid between 1 and 5
and listing.listid between 1 and 5
order by 1,2;
```

```
sales_listid | listing_listid
-----+-----
1            | 1
1            | 2
1            | 3
1            | 4
1            | 5
4            | 1
4            | 2
4            | 3
4            | 4
4            | 5
5            | 1
5            | 1
5            | 2
5            | 2
5            | 3
5            | 3
5            | 4
5            | 4
5            | 5
5            | 5
```

下列範例是兩個資料表之間的自然聯結。在這種情況下，兩個資料表中的 listid、sellerid、eventid 和 dateid 資料欄會具有相同的名稱和資料類型，因此會被用來作為聯結資料欄。結果限制為 5 個資料列。

```
select listid, sellerid, eventid, dateid, numtickets
from listing natural join sales
order by 1
```

```
limit 5;
```

listid	sellerid	eventid	dateid	numtickets
113	29704	4699	2075	22
115	39115	3513	2062	14
116	43314	8675	1910	28
118	6079	1611	1862	9
163	24880	8253	1888	14

下列範例是兩個資料表之間的聯結和 USING 子句。在這種情況下，資料欄 listid 和 eventid 會被用來作為聯結資料欄。結果限制為 5 個資料列。

```
select listid, listing.sellerid, eventid, listing.dateid, numtickets
from listing join sales
using (listid, eventid)
order by 1
limit 5;
```

listid	sellerid	eventid	dateid	numtickets
1	36861	7872	1850	10
4	8117	4337	1970	8
5	1616	8647	1963	4
5	1616	8647	1963	4
6	47402	8240	2053	18

以下查詢為 FROM 子句中兩個子查詢的內部聯結。查詢會尋找不同類別活動 (演奏會和表演) 的已售出和未售出票券數目。FROM 子句子查詢是資料表子查詢，可傳回多個資料欄和資料列。

```
select catgroup1, sold, unsold
from
(select catgroup, sum(qtysold) as sold
from category c, event e, sales s
where c.catid = e.catid and e.eventid = s.eventid
group by catgroup) as a(catgroup1, sold)
join
(select catgroup, sum(numtickets)-sum(qtysold) as unsold
from category c, event e, sales s, listing l
where c.catid = e.catid and e.eventid = s.eventid
and s.listid = l.listid
group by catgroup) as b(catgroup2, unsold)
```

```
on a.catgroup1 = b.catgroup2
order by 1;
```

catgroup1	sold	unsold
Concerts	195444	1067199
Shows	149905	817736

WHERE 子句

WHERE 子句包含聯結資料表或套用述詞至資料表中資料欄的條件。資料表可以藉由在 WHERE 子句或 FROM 子句中使用適當的語法進行內部聯結。外部連結條件必須在 FROM 子句中指定。

語法

```
[ WHERE condition ]
```

條件

任何產生布林值結果的搜尋條件，例如，資料表資料欄的聯結條件或述詞。以下範例為有效的聯結條件：

```
sales.listid=listing.listid
sales.listid<>listing.listid
```

以下範例對於資料表中的資料欄是有效的條件：

```
catgroup like 'S%'
venue seats between 20000 and 50000
eventname in('Jersey Boys','Spamalot')
year=2008
length(catdesc)>25
date_part(month, caldate)=6
```

條件可分成簡單和複雜；若是複雜條件，您可以使用括號來隔離邏輯單位。在下列範例中，聯結條件會以括號包圍。

```
where (category.catid=event.catid) and category.catid in(6,7,8)
```

使用須知

您無法在 WHERE 子句中使用別名來參考選取清單表達式。

您無法限制 WHERE 子句中彙整函數的結果；請使用 HAVING 子句達成此目的。

WHERE 子句中限制的資料欄必須衍生自 FROM 子句中的資料表參考。

範例

以下查詢使用不同 WHERE 子句限制的組合，包括 SALES 和 EVENT 資料表的聯結條件、EVENTNAME 資料欄上的述詞，以及 STARTTIME 資料欄上的兩個述詞。

```
select eventname, starttime, pricepaid/qtysold as costperticket, qtysold
from sales, event
where sales.eventid = event.eventid
and eventname='Hannah Montana'
and date_part(quarter, starttime) in(1,2)
and date_part(year, starttime) = 2008
order by 3 desc, 4, 2, 1 limit 10;
```

eventname	starttime	costperticket	qtysold
Hannah Montana	2008-06-07 14:00:00	1706.00000000	2
Hannah Montana	2008-05-01 19:00:00	1658.00000000	2
Hannah Montana	2008-06-07 14:00:00	1479.00000000	1
Hannah Montana	2008-06-07 14:00:00	1479.00000000	3
Hannah Montana	2008-06-07 14:00:00	1163.00000000	1
Hannah Montana	2008-06-07 14:00:00	1163.00000000	2
Hannah Montana	2008-06-07 14:00:00	1163.00000000	4
Hannah Montana	2008-05-01 19:00:00	497.00000000	1
Hannah Montana	2008-05-01 19:00:00	497.00000000	2
Hannah Montana	2008-05-01 19:00:00	497.00000000	4

(10 rows)

WHERE 子句中 Oracle 樣式的外部聯結

為了提供 Oracle 相容性，Amazon Redshift 在 WHERE 子句聯結條件中支援 Oracle 外部聯結運算子 (+)。此運算子主要僅用於定義外部聯結條件；請勿嘗試在其他內容中使用它。大多數情況下，此運算子的其他用途都會加以忽略，且不會顯示任何訊息。

外部聯結會傳回對等內部聯結傳回的所有資料列，加上其中一個資料表或兩個資料表的不相符資料列。在 FROM 子句中，您可以指定左、右和完整外部聯結。在 WHERE 子句中，您只能指定左和右外部聯結。

若要對 TABLE1 和 TABLE2 進行外部聯結並從 TABLE1 傳回不相符的資料列 (左外部聯結)，請在 FROM 子句中指定 TABLE1 LEFT OUTER JOIN TABLE2，或在 WHERE 子句中對來自 TABLE2 的所有聯結資料欄套用 (+) 運算子。對於 TABLE1 中在 TABLE2 中沒有相符資料列的所有資料列，查詢的結果會對包含 TABLE2 中資料欄的任何選取清單表達式包含 null。

若要對 TABLE2 中在 TABLE1 中沒有相符資料列的所有資料列產生相同的行為，請在 FROM 子句中指定 TABLE1 RIGHT OUTER JOIN TABLE2，或在 WHERE 子句中對來自 TABLE1 的所有聯結資料欄套用 (+) 運算子。

基本語法

```
[ WHERE {  
  [ table1.column1 = table2.column1(+) ]  
  [ table1.column1(+) = table2.column1 ]  
}
```

第一個條件相當於：

```
from table1 left outer join table2  
on table1.column1=table2.column1
```

第二個條件相當於：

```
from table1 right outer join table2  
on table1.column1=table2.column1
```

Note

這裡顯示的語法採用一個聯結資料欄配對說明簡單的 equijoin 案例。不過，其他類型的比較條件和多個聯結資料欄配對同樣有效。

例如，下列 WHERE 子句會定義兩個資料欄配對之間的外部聯結。(+) 運算子在這兩個條件中必須連接至相同資料表：

```
where table1.col1 > table2.col1(+)
and table1.col2 = table2.col2(+)
```

使用須知

盡可能使用標準 FROM 子句 OUTER JOIN 語法，而不要在 WHERE 子句中使用 (+) 運算子。包含 (+) 運算子的查詢受到下列規則限制：

- 您只能在 WHERE 子句中使用 (+) 運算子，而且只能參考資料表或檢視的資料欄。
- 您無法將 (+) 運算子套用至表達式。不過，表達式可包含使用 (+) 運算子的資料欄。例如，下列聯結條件會傳回語法錯誤：

```
event.eventid*10(+)=category.catid
```

不過，下列聯結條件有效：

```
event.eventid(+)*10=category.catid
```

- 您無法在同時包含 FROM 子句聯結語法的查詢區塊中使用 (+) 運算子。
- 若兩個資料表是透過多個聯結條件聯結，您必須在所有條件中使用 (+) 運算子，或完全不使用。採用混合語法樣式的聯結會做為內部聯結執行，但不會產生警告。
- 若您將外部查詢中的資料表與產生自內部查詢的資料表聯結，則 (+) 運算子不會產生外部聯結。
- 若要使用 (+) 運算子將資料表與其本身進行外部聯結，您必須在 FROM 子句中定義資料表別名，並且在聯結條件中參考這些別名：

```
select count(*)
from event a, event b
where a.eventid(+)=b.catid;
```

```
count
-----
8798
(1 row)
```

- 您無法將包含 (+) 運算子的聯結條件與 OR 條件或 IN 條件結合。例如：

```
select count(*) from sales, listing
where sales.listid(+)=listing.listid or sales.salesid=0;
ERROR: Outer join operator (+) not allowed in operand of OR or IN.
```


- 在與兩個以上資料表進行外部聯結的 WHERE 子句中，(+) 運算子只能對特定資料表套用一次。在以下範例中，SALES 資料表無法在兩個連續聯結中使用 (+) 運算子參考。

```
select count(*) from sales, listing, event
where sales.listid(+)=listing.listid and sales.dateid(+)=date.dateid;
ERROR:  A table may be outer joined to at most one other table.
```

- 若 WHERE 子句的外部聯結條件會將 TABLE2 的資料欄與常數進行比較，則將 (+) 運算子套用至資料欄。若您未包含運算子，就會消除 TABLE1 中的外部聯結資料列 (當中限制的資料欄會包含 null)。請參閱下方範例一節。

範例

以下聯結查詢會在 LISTID 資料欄上指定 SALES 和 LISTING 資料表的左外部聯結：

```
select count(*)
from sales, listing
where sales.listid = listing.listid(+);

count
-----
172456
(1 row)
```

下列對等查詢會產生相同的結果，但使用 FROM 子句聯結語法：

```
select count(*)
from sales left outer join listing on sales.listid = listing.listid;

count
-----
172456
(1 row)
```

SALES 資料表未包含 LISTING 資料表中所有清單的記錄，因為並非所有清單都產生銷售。以下查詢會將 SALES 和 LISTING 進行外部聯結，並從 LISTING 傳回資料列，即使 SALES 資料表回報特定清單 ID 沒有銷售。PRICE 和 COMM 資料欄衍生自 SALES 資料表，其結果集中的不相符資料列會包含 null。

```
select listing.listid, sum(pricepaid) as price,
sum(commission) as comm
```

```

from listing, sales
where sales.listid(+) = listing.listid and listing.listid between 1 and 5
group by 1 order by 1;

```

```

listid | price | comm
-----+-----+-----
1 | 728.00 | 109.20
2 |      |
3 |      |
4 | 76.00 | 11.40
5 | 525.00 | 78.75
(5 rows)

```

請注意，使用 WHERE 子句聯結運算子時，資料表在 FROM 子句中的順序並不重要。

WHERE 子句中較複雜的外部聯結條件範例，就是條件由兩個資料表資料欄之間的比較，以及與常數的比較所構成：

```

where category.catid=event.catid(+) and eventid(+)=796;

```

請注意，(+) 運算子會在兩處使用：一處是資料表之間的對等比較中，另一處是 EVENTID 資料欄的比較條件中。此語法的結果是在評估 EVENTID 的限制時，保留外部聯結資料列。若您從 EVENTID 限制中移除 (+) 運算子，則查詢會將此限制視為篩選條件，而非外部聯結條件的一部分。接著就會從結果集中消除 EVENTID 包含 null 的外部聯結資料列。

以下是說明此行為的完整查詢：

```

select catname, catgroup, eventid
from category, event
where category.catid=event.catid(+) and eventid(+)=796;

```

```

catname | catgroup | eventid
-----+-----+-----
Classical | Concerts |
Jazz | Concerts |
MLB | Sports |
MLS | Sports |
Musicals | Shows | 796
NBA | Sports |
NFL | Sports |
NHL | Sports |
Opera | Shows |

```

```
Plays | Shows    |
Pop   | Concerts |
(11 rows)
```

使用 FROM 子句語法的對等查詢如下所示：

```
select catname, catgroup, eventid
from category left join event
on category.catid=event.catid and eventid=796;
```

若您從此查詢的 WHERE 子句版本中移除第二個 (+) 運算子，則只會傳回 1 個資料列 (eventid=796 的資料列)。

```
select catname, catgroup, eventid
from category, event
where category.catid=event.catid(+) and eventid=796;
```

```
catname | catgroup | eventid
-----+-----+-----
Musicals | Shows    | 796
(1 row)
```

GROUP BY 子句

GROUP BY 子句會識別查詢的分組資料欄。分組資料欄必須在查詢使用標準函數運算彙整時宣告，像是 SUM、AVG 和 COUNT。如需詳細資訊，請參閱 [彙總函數](#)。

語法

```
GROUP BY group_by_clause [, ...]

group_by_clause := {
    expr |
    GROUPING SETS ( () | group_by_clause [, ...] ) |
    ROLLUP ( expr [, ...] ) |
    CUBE ( expr [, ...] )
}
```

參數

expr

在查詢的選取清單中，資料欄或表達式的清單必須符合非彙整表達式的清單。例如，請考量以下簡單查詢。

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by listid, eventid
order by 3, 4, 2, 1
limit 5;
```

listid	eventid	revenue	numtix
89397	47	20.00	1
106590	76	20.00	1
124683	393	20.00	1
103037	403	20.00	1
147685	429	20.00	1

(5 rows)

在此查詢中，選取清單是由兩個彙整表達式所構成。第一個使用 SUM 函數，第二個使用 COUNT 函數。其餘兩個資料欄 LISTID 和 EVENTID 必須宣告為分組資料欄。

GROUP BY 子句中的表達式也可以使用序數來參考選取清單。例如，前一個範例可縮減如下。

```
select listid, eventid, sum(pricepaid) as revenue,
count(qtysold) as numtix
from sales
group by 1,2
order by 3, 4, 2, 1
limit 5;
```

listid	eventid	revenue	numtix
89397	47	20.00	1
106590	76	20.00	1
124683	393	20.00	1
103037	403	20.00	1
147685	429	20.00	1

```
(5 rows)
```

GROUPING SETS/ROLLUP/CUBE

您可以使用彙總延伸項目 GROUPING SETS、ROLLUP 和 CUBE，在單一陳述式中執行多個 GROUP BY 操作的工作。如需彙總延伸項目及相關函數的相關資訊，請參閱 [彙總延伸項目](#)。

彙總延伸項目

Amazon Redshift 支援彙總延伸項目，可在單一陳述式中執行多個 GROUP BY 操作的工作。

彙總延伸項目的範例會使用 orders 資料表，該資料表會保留電子公司的銷售資料。您可以執行下列操作來建立 orders：

```
CREATE TABLE ORDERS (  
  ID INT,  
  PRODUCT CHAR(20),  
  CATEGORY CHAR(20),  
  PRE_OWNED CHAR(1),  
  COST DECIMAL  
);  
  
INSERT INTO ORDERS VALUES  
  (0, 'laptop',      'computers',    'T', 1000),  
  (1, 'smartphone', 'cellphones',   'T', 800),  
  (2, 'smartphone', 'cellphones',   'T', 810),  
  (3, 'laptop',     'computers',    'F', 1050),  
  (4, 'mouse',      'computers',    'F', 50);
```

GROUPING SETS

在單一陳述式中計算一或多個群組集。群組集是單一 GROUP BY 子句的集合，也就是一組 0 個或多個資料行，您可以以此將查詢的結果集分組。GROUP BY GROUPING SETS 等同於在由不同資料欄分組的一個結果集上執行 UNION ALL 查詢。例如，GROUP BY GROUPING SETS((a), (b)) 等同於 GROUP BY a UNION ALL GROUP BY b。

下列範例會傳回訂單資料表的產品根據產品類別和銷售產品種類進行分組的成本。

```
SELECT category, product, sum(cost) as total
```

```
FROM orders
GROUP BY GROUPING SETS(category, product);
```

category	product	total
computers		2100
cellphones		1610
	laptop	2050
	smartphone	1610
	mouse	50

(5 rows)

ROLLUP

假設有一個階層，其中之前的資料欄被視為後續資料欄的父項。ROLLUP 會依提供的資料欄將資料分組，並傳回額外的小計資料列 (代表所有分組資料欄層級的總計)，以及已分組的資料列。例如，您可以使用 GROUP BY ROLLUP((a), (b)) 傳回一個先按 a 分組，然後按 b 分組的結果集 (假設 b 是 a 的子區段)。ROLLUP 也會傳回包含整個結果集的資料列，而不會將資料欄分組。

GROUP BY ROLLUP((a), (b)) 等於 GROUP BY GROUPING SETS((a,b), (a), ())。

下列範例會傳回訂單資料表產品先依類別分組，然後依產品分組的成本，其中以產品做為類別的細項。

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY ROLLUP(category, product) ORDER BY 1,2;
```

category	product	total
cellphones	smartphone	1610
cellphones		1610
computers	laptop	2050
computers	mouse	50
computers		2100
		3710

(6 rows)

CUBE

依提供的資料欄將資料分組，並傳回額外的小計資料列 (代表所有分組資料欄層級的總計)，以及已分組的資料列。CUBE 會傳回與 ROLLUP 相同的資料列，同時針對 ROLLUP 未涵蓋的每個分組資料欄組合

新增額外的小計資料列。例如，您可以使用 GROUP BY CUBE ((a), (b)) 傳回一個先按 a 分組，然後按 b 分組的結果集 (假設 b 是 a 的子區段)，然後再獨自按 b 分組。CUBE 也會傳回包含整個結果集的資料列，而不會將資料欄分組。

GROUP BY CUBE((a), (b)) 等於 GROUP BY GROUPING SETS((a, b), (a), (b), ())。

下列範例會傳回訂單資料表產品先依類別分組，然後依產品分組的成本，其中以產品做為類別的細項。與前面的 ROLLUP 範例不同，陳述式會傳回每個分組資料欄組合的結果。

```
SELECT category, product, sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 1,2;
```

category	product	total
cellphones	smartphone	1610
cellphones		1610
computers	laptop	2050
computers	mouse	50
computers		2100
	laptop	2050
	mouse	50
	smartphone	1610
		3710

(9 rows)

GROUPING/GROUPING_ID 函數

ROLLUP 和 CUBE 會將 NULL 值新增至結果集，以指出小計資料列。例如，GROUP BY ROLLUP((a), (b)) 會傳回 b 分組資料欄中值為 NULL 的一或多個資料列，以指出這些資料列是分組資料欄中欄位的小計。這些 NULL 值僅用於滿足傳回元組的格式。

當您在儲存 NULL 值本身的關係上使用 ROLLUP 和 CUBE 執行 GROUP BY 操作時，這可能會產生資料列看起來具有相同分組資料欄的結果集。返回上一個範例，如果 b 分組資料欄包含一個儲存的 NULL 值，GROUP BY ROLLUP((a), (b)) 會在 b 分組資料欄中 (不是小計) 傳回值為 NULL 的資料列。

若要區分 ROLLUP 和 CUBE 建立的 NULL 值，以及儲存在資料表本身的 NULL 值，您可以使用 GROUPING 函數或其別名 GROUPING_ID。GROUPING 採用單個分組集作為其引數，並且對於結果集中的每一個資料列傳回對應於該位置中分組資料欄的 0 或 1 位元值，然後將該值轉換為整數。如果該位置中的值是由彙總延伸項目建立的 NULL 值，則 GROUPING 會傳回 1。對於所有其他值，包括儲存的 NULL 值，則傳回 0。

例如，GROUPING(category, product) 可以針對指定的資料列傳回下列值，視該資料列的分組資料欄值而定。就此範例而言，資料表中的所有 NULL 值都是彙總延伸項目所建立的 NULL 值。

類別資料欄	產品資料欄	GROUPING 函數位元值	十進制值
非 NULL	非 NULL	00	0
非 NULL	NULL	01	1
NULL	非 NULL	10	2
NULL	NULL	11	3

GROUPING 函數會以下列格式顯示在查詢的 SELECT 清單部分中。

```
SELECT ... [GROUPING( expr )...] ...
GROUP BY ... {CUBE | ROLLUP| GROUPING SETS} ( expr ) ...
```

下列範例與前面的 CUBE 範例相同，但為其分組集增加了 GROUPING 函數。

```
SELECT category, product,
       GROUPING(category) as grouping0,
       GROUPING(product) as grouping1,
       GROUPING(category, product) as grouping2,
       sum(cost) as total
FROM orders
GROUP BY CUBE(category, product) ORDER BY 3,1,2;
```

```

      category      |      product      | grouping0 | grouping1 | grouping2 |
-----+-----+-----+-----+-----
+-----
cellphones          | smartphone        |          0 |          0 |          0 |
1610
cellphones          |                   |          0 |          1 |          1 |
1610
computers           | laptop            |          0 |          0 |          0 |
2050
```



```

computers      | mouse      |          0 |          0 |          0 |
50
computers      |            |          0 |          1 |          1 |
2100
                | laptop     |          1 |          0 |          2 |
2050
                | mouse      |          1 |          0 |          2 |
50
                | smartphone |          1 |          0 |          2 |
1610
                |            |          1 |          1 |          3 |
3710
(9 rows)

```

部份 ROLLUP 與 CUBE

您只能使用小計的一部分來執行 ROLLUP 和 CUBE 操作。

部分 ROLLUP 和 CUBE 操作的語法如下。

```
GROUP BY expr1, { ROLLUP | CUBE }( expr2, [, ...] )
```

在這裡，GROUP BY 子句僅在表達式 *expr2* 和以後的層級上建立小計資料列。

下列範例會顯示訂單資料表上的部份 ROLLUP 和 CUBE 操作，並先依產品是否為二手產品分組，然後在類別與產品資料欄上執行 ROLLUP 和 CUBE。

```

SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY pre_owned, ROLLUP(category, product) ORDER BY 4,1,2,3;

```

pre_owned	category	product	group_id	total
F	computers	laptop	0	1050
F	computers	mouse	0	50
T	cellphones	smartphone	0	1610
T	computers	laptop	0	1000
F	computers		2	1100
T	cellphones		2	1610
T	computers		2	1000
F			6	1100

```
T          |          |          |          | 6 | 2610
(9 rows)
```

```
SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
FROM orders
GROUP BY pre_owned, CUBE(category, product) ORDER BY 4,1,2,3;
```

pre_owned	category	product	group_id	total
F	computers	laptop	0	1050
F	computers	mouse	0	50
T	cellphones	smartphone	0	1610
T	computers	laptop	0	1000
F	computers		2	1100
T	cellphones		2	1610
T	computers		2	1000
F		laptop	4	1050
F		mouse	4	50
T		laptop	4	1000
T		smartphone	4	1610
F			6	1100
T			6	2610

(13 rows)

由於二手資料欄不包含在 ROLLUP 和 CUBE 作業中，因此沒有包含所有其他資料列的總計資料列。

串連分組

您可以串連多個 GROUPING SETS/ROLLUP/CUBE 子句，以計算不同層次的小計。串連分組會傳回所提供分組集的笛卡爾乘積。

串連 GROUPING SETS/ROLLUP/CUBE 子句的語法如下。

```
GROUP BY {ROLLUP|CUBE|GROUPING SETS}(expr1[, ...]),
        {ROLLUP|CUBE|GROUPING SETS}(expr1[, ...])[, ...]
```

請考慮下列範例，看看小型串連分組如何產生大型的最終結果集。

```
SELECT pre_owned, category, product,
       GROUPING(category, product, pre_owned) as group_id,
       sum(cost) as total
```

```
FROM orders
GROUP BY CUBE(category, product), GROUPING SETS(pre_owned, ())
ORDER BY 4,1,2,3;
```

pre_owned	category	product	group_id	total
F	computers	laptop	0	1050
F	computers	mouse	0	50
T	cellphones	smartphone	0	1610
T	computers	laptop	0	1000
	cellphones	smartphone	1	1610
	computers	laptop	1	2050
	computers	mouse	1	50
F	computers		2	1100
T	cellphones		2	1610
T	computers		2	1000
	cellphones		3	1610
	computers		3	2100
F		laptop	4	1050
F		mouse	4	50
T		laptop	4	1000
T		smartphone	4	1610
		laptop	5	2050
		mouse	5	50
		smartphone	5	1610
F			6	1100
T			6	2610
			7	3710

(22 rows)

巢狀分組

您可以使用 GROUPING SETS/ROLLUP/CUBE 操作作為您的 GROUPING SETS expr，形成一個巢狀分組。巢狀 GROUPING SETS 中的子分組會被扁平化。

巢狀分組的語法如下。

```
GROUP BY GROUPING SETS({ROLLUP|CUBE|GROUPING SETS}(expr[, ...])[, ...])
```

請考量下列範例。

```
SELECT category, product, pre_owned,
       GROUPING(category, product, pre_owned) as group_id,
```

```

    sum(cost) as total
FROM orders
GROUP BY GROUPING SETS(ROLLUP(category), CUBE(product, pre_owned))
ORDER BY 4,1,2,3;

```

category	product	pre_owned	group_id	total
cellphones			3	1610
computers			3	2100
	laptop	F	4	1050
	laptop	T	4	1000
	mouse	F	4	50
	smartphone	T	4	1610
	laptop		5	2050
	mouse		5	50
	smartphone		5	1610
		F	6	1100
		T	6	2610
			7	3710
			7	3710

(13 rows)

請注意，由於 ROLLUP(category) 和 CUBE(product, pre_owned) 都包含分組集 ()，因此代表總計的資料列將重複。

使用須知

- GROUP BY 子句最多支援 64 個分組集。如果是 ROLLUP 和 CUBE，或是 GROUPING SETS、ROLLUP 和 CUBE 的某些組合，則此限制會套用至隱含的分組集數目。例如，GROUP BY CUBE((a), (b)) 計為 4 個分組集，而不是 2 個。
- 使用彙總延伸項目時，您無法使用常數作為分組資料欄。
- 您無法建立包含重複資料欄的分組集。

HAVING 子句

HAVING 子句會將條件套用至查詢傳回的中繼分組結果集。

語法

```
[ HAVING condition ]
```

例如，您可以限制 SUM 函數的結果：

```
having sum(pricepaid) >10000
```

HAVING 會在套用所有 WHERE 子句條件且完成 GROUP BY 操作之後套用。

條件本身會採用與任何 WHERE 子句條件相同的形式。

使用須知

- HAVING 子句條件中參考的任何資料欄必須是分組資料欄，或是參考彙整函數結果的資料欄。
- 在 HAVING 子句中，您無法指定：
 - 參考選取清單項目的序數。只有 GROUP BY 和 ORDER BY 子句接受序數。

範例

以下查詢會依名稱計算售票總金額，然後消除總金額低於 800,000 USD 的活動。HAVING 條件會套用至選取清單中彙整函數的結果：sum(pricepaid)。

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
group by 1
having sum(pricepaid) > 800000
order by 2 desc, 1;
```

eventname	sum
Mamma Mia!	1135454.00
Spring Awakening	972855.00
The Country Girl	910563.00
Macbeth	862580.00
Jersey Boys	811877.00
Legally Blonde	804583.00

以下查詢會計算類似的結果集。不過，在此情況下，HAVING 條件會套用至選取清單中未指定的彙整：sum(qtysold)。未銷售超過 2,000 張票的活動將從最終結果中消除。

```
select eventname, sum(pricepaid)
from sales join event on sales.eventid = event.eventid
```

```
group by 1
having sum(qtysold) >2000
order by 2 desc, 1;
```

eventname	sum
Mamma Mia!	1135454.00
Spring Awakening	972855.00
The Country Girl	910563.00
Macbeth	862580.00
Jersey Boys	811877.00
Legally Blonde	804583.00
Chicago	790993.00
Spamalot	714307.00

以下查詢會依名稱計算售票總金額，然後消除總金額低於 800,000 USD 的活動。HAVING 條件會套用至使用別名 pp for 的選取清單中彙總函式的結果sum(pricepaid)。

```
select eventname, sum(pricepaid) as pp
from sales join event on sales.eventid = event.eventid
group by 1
having pp > 800000
order by 2 desc, 1;
```

eventname	pp
Mamma Mia!	1135454.00
Spring Awakening	972855.00
The Country Girl	910563.00
Macbeth	862580.00
Jersey Boys	811877.00
Legally Blonde	804583.00


QUALIFY 子句

QUALIFY 子句會根據使用者指定的搜尋條件，篩選先前計算的視窗函數結果。您可以使用子句將篩選條件套用至視窗函數的結果，而不需要使用子查詢。

其類似於 [HAVING 子句](#)，該子句適用於從 WHERE 子句進一步篩選資料列的條件。QUALIFY 和 HAVING 之間的區別在於，從 QUALIFY 子句篩選的結果可以基於在資料上執行窗口函數的結果。您可以在一個查詢中同時使用 QUALIFY 和 HAVING 子句。

語法

```
QUALIFY condition
```

 Note

如果您直接在 FROM 子句之後使用 QUALIFY 子句，則 FROM 關係名稱必須在 QUALIFY 子句之前指定別名。

範例

本節中的範例使用以下範例資料。

```
create table store_sales (ss_sold_date date, ss_sold_time time,
                          ss_item text, ss_sales_price float);
insert into store_sales values ('2022-01-01', '09:00:00', 'Product 1', 100.0),
                              ('2022-01-01', '11:00:00', 'Product 2', 500.0),
                              ('2022-01-01', '15:00:00', 'Product 3', 20.0),
                              ('2022-01-01', '17:00:00', 'Product 4', 1000.0),
                              ('2022-01-01', '18:00:00', 'Product 5', 30.0),
                              ('2022-01-02', '10:00:00', 'Product 6', 5000.0),
                              ('2022-01-02', '16:00:00', 'Product 7', 5.0);
```

下列範例示範如何找到每天 12:00 之後售出的兩個最昂貴的物品。

```
SELECT *
FROM store_sales ss
WHERE ss_sold_time > time '12:00:00'
QUALIFY row_number()
OVER (PARTITION BY ss_sold_date ORDER BY ss_sales_price DESC) <= 2
```

ss_sold_date	ss_sold_time	ss_item	ss_sales_price
2022-01-01	17:00:00	Product 4	1000
2022-01-01	18:00:00	Product 5	30
2022-01-02	16:00:00	Product 7	5

然後，您可以找到每天售出的最後一件物品。

```

SELECT *
FROM store_sales ss
QUALIFY last_value(ss_item)
OVER (PARTITION BY ss_sold_date ORDER BY ss_sold_time ASC
      ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) = ss_item;

```

ss_sold_date	ss_sold_time	ss_item	ss_sales_price
2022-01-01	18:00:00	Product 5	30
2022-01-02	16:00:00	Product 7	5

下列範例會傳回與上一個查詢相同的記錄，也就是每天售出的最後一個項目，但不使用 `QUALIFY` 子句。

```

SELECT * FROM (
  SELECT *,
  last_value(ss_item)
  OVER (PARTITION BY ss_sold_date ORDER BY ss_sold_time ASC
        ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING) ss_last_item
  FROM store_sales ss
)
WHERE ss_last_item = ss_item;

```

ss_sold_date	ss_sold_time	ss_item	ss_sales_price	ss_last_item
2022-01-02	16:00:00	Product 7	5	Product 7
2022-01-01	18:00:00	Product 5	30	Product 5

UNION、INTERSECT 和 EXCEPT

主題

- [語法](#)
- [參數](#)
- [集合運算子的評估順序](#)
- [使用須知](#)
- [範例 UNION 查詢](#)
- [範例 UNION ALL 查詢](#)
- [範例 INTERSECT 查詢](#)

- [範例 EXCEPT 查詢](#)

UNION、INTERSECT 和 EXCEPT 集合運算子可用來比較和合併兩種不同查詢表達式的結果。例如，如果您想知道哪些網站使用者同時是買方和賣家，但其使用者名稱儲存在不同的資料欄或資料表中，您可以找出這兩種類型使用者的交集。如果您想知道哪些網站使用者是買方，但不是賣家，您可以使用 EXCEPT 運算子找出兩份使用者清單之間的差異。如果您想要建構所有使用者的清單，但不考慮角色，您可以使用 UNION 運算子。

語法

```
query
{ UNION [ ALL ] | INTERSECT | EXCEPT | MINUS }
query
```

參數

query

此查詢表達式會以其選取清單形式，對應至接在 UNION、INTERSECT 或 EXCEPT 運算子後面的另一個查詢表達式。兩個表達式必須包含採用相容資料類型的相同輸出資料欄數，否則就無法比較和合併這兩個結果集。集合操作不允許在不同類別的資料類型之間進行隱含轉換；如需詳細資訊，請參閱 [類型相容性與轉換](#)。

您可以建構包含無限查詢表達式數目的查詢，並將它們與 UNION、INTERSECT 和 EXCEPT 運算子的任意組合連結。例如，假設資料表 T1、T2 和 T3 包含相容的資料欄集，則以下查詢結構有效：

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

UNION

此集合操作會從兩個查詢表達式傳回資料列，無論資料列衍生自其中一個或兩個表達式。

INTERSECT

此集合操作會傳回衍生自兩個查詢表達式的資料列。未由兩個表達式傳回的資料列則會遭到捨棄。

EXCEPT | MINUS

此集合操作會傳回衍生自兩個查詢表達式之一的資料列。若要限定結果，資料列必須存在第一個結果資料表中，但不能存在第二個資料表中。MINUS 和 EXCEPT 是一模一樣的同義詞。

ALL

ALL 關鍵字會保留 UNION 所產生的任何重複資料列。未使用 ALL 關鍵字時的預設行為是捨棄這些重複項目。不支援 INTERSECT ALL、EXCEPT ALL 和 MINUS ALL。

集合運算子的評估順序

UNION 和 EXCEPT 集合運算子為左關聯。若未指定括號來影響優先順序，則會從左到右評估這些集合運算子的組合。例如，在下列查詢中，T1 和 T2 的 UNION 會先評估，然後在 UNION 結果上執行 EXCEPT 操作：

```
select * from t1
union
select * from t2
except
select * from t3
order by c1;
```

在相同查詢中使用運算子組合時，INTERSECT 運算子的優先順序高於 UNION 和 EXCEPT 運算子。例如，下列查詢會先評估 T2 和 T3 的交集，再將結果與 T1 進行聯集：

```
select * from t1
union
select * from t2
intersect
select * from t3
order by c1;
```

加入括號就可以強制執行不同的評估順序。在下列案例中，T1 和 T2 的聯集結果會與 T3 交集，而查詢可能會產生不同的結果。

```
(select * from t1
union
select * from t2)
intersect
(select * from t3)
```

```
order by c1;
```

使用須知

- 集合操作查詢的結果中傳回的資料欄名稱，是來自第一個查詢表達式的資料表中的資料欄名稱 (或別名)。這些資料欄名稱可能會造成誤導，因為資料欄中的值是從任一邊集合運算子的資料表衍生，所以建議您為結果集提供有意義的別名。
- 先於集合運算子的查詢表達式不應包含 ORDER BY 子句。只有在包含集合運算子的查詢結尾使用 ORDER BY 子句時，該子句才會產生有意義的排序結果。在此情況下，ORDER BY 子句會套用至所有集合操作的最終結果。最外層的查詢也可包含標準 LIMIT 和 OFFSET 子句。
- 當集合運算子查詢傳回小數結果時，對應的結果資料欄就會提升，以傳回相同的精確度和小數位數。例如，在以下查詢中，T1.REVENUE 是 DECIMAL(10,2) 資料欄，而 T2.REVENUE 是 DECIMAL(8,4) 資料欄，小數結果會提升為 DECIMAL(12,4)：

```
select t1.revenue union select t2.revenue;
```

小數位數為 4，因為這是兩個資料欄的小數位數上限。精確度為 12，因為 T1.REVENUE 要求小數點左邊有 8 位數 ($12 - 4 = 8$)。此類型提升可確保 UNION 兩邊的所有值都能納入結果中。若是 64 位元值，最高結果精確度為 19，而結果小數位數上限為 18。若是 128 位元值，最高結果精確度為 38，而結果小數位數上限為 37。

若產生的資料類型超過 Amazon Redshift 精確度和小數位數限制，查詢就會傳回錯誤。

- 在集合操作中，若每個對應資料欄配對的這兩個資料值為等於或兩者皆為 NULL，則這兩個資料列會視為相同。例如，若資料表 T1 和 T2 都包含一個資料欄和一個資料列，而該資料列在兩個資料表中都是 NULL，則對這些資料表執行 INTERSECT 操作就會傳回該資料列。

範例 UNION 查詢

在下列 UNION 查詢中，SALES 資料表中的資料列會與 LISTING 資料表中的資料列合併。會從每個資料表選取三個相容的資料欄；在此情況下，對應的資料欄會有相同的名稱和資料類型。

最終結果集是依 LISTING 資料表中的第一欄排序，且限於擁有最高 LISTID 值的 5 個資料列。

```
select listid, sellerid, eventid from listing
union select listid, sellerid, eventid from sales
order by listid, sellerid, eventid desc limit 5;

listid | sellerid | eventid
```

```

-----+-----+-----
1 |    36861 |    7872
2 |    16002 |    4806
3 |    21461 |    4256
4 |     8117 |    4337
5 |     1616 |    8647
(5 rows)

```

以下範例說明如何將常值新增至 UNION 查詢的輸出，以便查看結果集中的每個資料列是由哪個查詢表達式所產生。查詢會將來自第一個查詢表達式的資料列識別為 "B" (表示買方)，來自第二個查詢表達式的資料列識別為 "S" (表示賣家)。

查詢會識別價值 10,000 USD 以上的票券交易買方和賣家。UNION 運算子任一邊的兩個查詢表達式之間唯一的差異，就是 SALES 資料表的聯結資料欄。

```

select listid, lastname, firstname, username,
pricepaid as price, 'S' as buyorsell
from sales, users
where sales.sellerid=users.userid
and pricepaid >=10000
union
select listid, lastname, firstname, username, pricepaid,
'B' as buyorsell
from sales, users
where sales.buyerid=users.userid
and pricepaid >=10000
order by 1, 2, 3, 4, 5;

```

```

listid | lastname | firstname | username | price   | buyorsell
-----+-----+-----+-----+-----+-----
209658 | Lamb    | Colette   | VOR15LYI | 10000.00 | B
209658 | West    | Kato      | ELU81XAA | 10000.00 | S
212395 | Greer   | Harlan    | GX071KOC | 12624.00 | S
212395 | Perry   | Cora      | YWR73YNZ | 12624.00 | B
215156 | Banks   | Patrick   | ZNQ69CLT | 10000.00 | S
215156 | Hayden  | Malachi   | BBG56AKU | 10000.00 | B
(6 rows)

```

以下範例使用 UNION ALL 運算子，因為重複的資料列 (若找到) 需保留在結果中。若是特定活動 ID 系列，查詢會針對與各個活動相關聯的每筆銷售傳回 0 或更多資料列，並針對該活動的每份清單傳回 0 或 1。LISTING 和 EVENT 資料表中每個資料列的活動 ID 都是唯一的，但 SALES 資料表中相同的活動和清單 ID 組合可能會有多筆銷售。

結果集中的第三個資料欄會識別資料列的來源。若是來自 SALES 資料表，則會在 SALESROW 資料欄中標示為「Yes (是)」(SALESROW 是 SALES.LISTID 的別名)。若資料列來自 LISTING 資料表，則會在 SALESROW 資料欄中標示為「No (否)」。

在此情況下，結果集會包含活動 7787、清單 500 的三個銷售資料列。換句話說，此清單與活動組合發生了三筆不同的交易。另外兩份清單 501 和 502 並未產生任何銷售，因此查詢針對這些清單 ID 產生的唯一資料列是來自 LISTING 資料表 (SALESROW = 'No')。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;
```

```
eventid | listid | salesrow
-----+-----+-----
7787 | 500 | No
7787 | 500 | Yes
7787 | 500 | Yes
7787 | 500 | Yes
6473 | 501 | No
5108 | 502 | No
(6 rows)
```

若您執行相同查詢，但未使用 ALL 關鍵字，結果只會保留其中一項銷售交易。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;
```

```
eventid | listid | salesrow
-----+-----+-----
7787 | 500 | No
7787 | 500 | Yes
```

```
6473 |    501 | No
5108 |    502 | No
(4 rows)
```

範例 UNION ALL 查詢

以下範例使用 UNION ALL 運算子，因為重複的資料列 (若找到) 需保留在結果中。若是特定活動 ID 系列，查詢會針對與各個活動相關聯的每筆銷售傳回 0 或更多資料列，並針對該活動的每份清單傳回 0 或 1。LISTING 和 EVENT 資料表中每個資料列的活動 ID 都是唯一的，但 SALES 資料表中相同的活動和清單 ID 組合可能會有多筆銷售。

結果集中的第三個資料欄會識別資料列的來源。若是來自 SALES 資料表，則會在 SALESROW 資料欄中標示為「Yes (是)」(SALESROW 是 SALES.LISTID 的別名)。若資料列來自 LISTING 資料表，則會在 SALESROW 資料欄中標示為「No (否)」。

在此情況下，結果集會包含活動 7787、清單 500 的三個銷售資料列。換句話說，此清單與活動組合發生了三筆不同的交易。另外兩份清單 501 和 502 並未產生任何銷售，因此查詢針對這些清單 ID 產生的唯一資料列是來自 LISTING 資料表 (SALESROW = 'No')。

```
select eventid, listid, 'Yes' as salesrow
from sales
where listid in(500,501,502)
union all
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;
```

```
eventid | listid | salesrow
-----+-----+-----
7787 |    500 | No
7787 |    500 | Yes
7787 |    500 | Yes
7787 |    500 | Yes
6473 |    501 | No
5108 |    502 | No
(6 rows)
```

若您執行相同查詢，但未使用 ALL 關鍵字，結果只會保留其中一項銷售交易。

```
select eventid, listid, 'Yes' as salesrow
```

```

from sales
where listid in(500,501,502)
union
select eventid, listid, 'No'
from listing
where listid in(500,501,502)
order by listid asc;

```

```

eventid | listid | salesrow
-----+-----+-----
7787 | 500 | No
7787 | 500 | Yes
6473 | 501 | No
5108 | 502 | No
(4 rows)

```

範例 INTERSECT 查詢

比較下列範例與第一個 UNION 範例。這兩個範例的唯一差異在於使用的集合運算子，但結果非常不同。只有其中一個資料列相同：

```

235494 | 23875 | 8771

```

這是在限制的 5 個資料列結果中，唯一同時存在兩個資料表中的資料列。

```

select listid, sellerid, eventid from listing
intersect
select listid, sellerid, eventid from sales
order by listid desc, sellerid, eventid
limit 5;

```

```

listid | sellerid | eventid
-----+-----+-----
235494 | 23875 | 8771
235482 | 1067 | 2667
235479 | 1589 | 7303
235476 | 15550 | 793
235475 | 22306 | 7848
(5 rows)

```

以下查詢會尋找三月份同時於紐約市和洛杉磯的場館舉行的活動 (有售票)。兩個查詢表達式之間唯一的差異就是 VENUECITY 資料欄的限制條件。

```

select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='Los Angeles'
intersect
select distinct eventname from event, sales, venue
where event.eventid=sales.eventid and event.venueid=venue.venueid
and date_part(month,starttime)=3 and venuecity='New York City'
order by eventname asc;

```

```

eventname
-----
A Streetcar Named Desire
Dirty Dancing
Electra
Running with Annalise
Hairspray
Mary Poppins
November
Oliver!
Return To Forever
Rhinoceros
South Pacific
The 39 Steps
The Bacchae
The Caucasian Chalk Circle
The Country Girl
Wicked
Woyzeck
(16 rows)

```

範例 EXCEPT 查詢

TICKIT 資料庫中的 CATEGORY 資料表包含以下 11 個資料列：

catid	catgroup	catname	catdesc
1	Sports	MLB	Major League Baseball
2	Sports	NHL	National Hockey League
3	Sports	NFL	National Football League
4	Sports	NBA	National Basketball Association
5	Sports	MLS	Major League Soccer
6	Shows	Musicals	Musical theatre
7	Shows	Plays	All non-musical theatre


```

 8 | Shows      | Opera      | All opera and light opera
 9 | Concerts   | Pop        | All rock and pop music concerts
10 | Concerts   | Jazz       | All jazz singers and bands
11 | Concerts   | Classical  | All symphony, concerto, and choir concerts
(11 rows)

```

假設 CATEGORY_STAGE 資料表 (臨時資料表) 包含一個額外的資料列：

```

catid | catgroup | catname | catdesc
-----+-----+-----+-----
 1 | Sports   | MLB      | Major League Baseball
 2 | Sports   | NHL      | National Hockey League
 3 | Sports   | NFL      | National Football League
 4 | Sports   | NBA      | National Basketball Association
 5 | Sports   | MLS      | Major League Soccer
 6 | Shows    | Musicals | Musical theatre
 7 | Shows    | Plays    | All non-musical theatre
 8 | Shows    | Opera    | All opera and light opera
 9 | Concerts | Pop      | All rock and pop music concerts
10 | Concerts | Jazz     | All jazz singers and bands
11 | Concerts | Classical | All symphony, concerto, and choir concerts
12 | Concerts | Comedy   | All stand up comedy performances
(12 rows)

```

傳回兩個資料表之間的差異。換句話說，會傳回 CATEGORY_STAGE 資料表而非 CATEGORY 資料表中的資料列：

```

select * from category_stage
except
select * from category;

catid | catgroup | catname | catdesc
-----+-----+-----+-----
12 | Concerts | Comedy | All stand up comedy performances
(1 row)

```

以下對等查詢使用同義詞 MINUS。

```

select * from category_stage
minus
select * from category;

```

```
catid | catgroup | catname |          catdesc
-----+-----+-----+-----
12 | Concerts | Comedy | All stand up comedy performances
(1 row)
```

若您將 SELECT 表達式的順序反轉，則查詢不會傳回任何資料列。

ORDER BY 子句

主題

- [語法](#)
- [參數](#)
- [使用須知](#)
- [ORDER BY 的範例](#)

ORDER BY 子句會排序查詢的結果集。

語法

```
[ ORDER BY expression [ ASC | DESC ] ]
[ NULLS FIRST | NULLS LAST ]
[ LIMIT { count | ALL } ]
[ OFFSET start ]
```

參數

運算式

此表達式會排序查詢結果集的順序，通常是藉由指定選取清單中的一個或多個資料欄。結果會根據二進位 UTF-8 順序傳回。您還可以指定下列項目：

- 不在選取清單中的資料欄
- 由查詢所參考資料表中的一個或多個資料欄構成的表達式
- 代表選取清單項目位置的序數 (或是，若沒有選取清單的話，則為資料欄在資料表中的位置)
- 定義選取清單項目的別名

當 ORDER BY 子句包含多個表達式時，結果集會根據第一個表達式排序，然後對擁有與第一個表達式相符之值的資料列套用第二個表達式，以此類推。

ASC | DESC

此選項會定義表達式的排序順序，如下所示：

- ASC：遞增 (例如，數值從低到高，字元字串 'A' 到 'Z')。若未指定選項，資料會預設為遞增排序。
- DESC：遞減 (數值從高到低，字串 'Z' 到 'A')。

NULLS FIRST | NULLS LAST

這些選項指定 NULL 值應該排序在最前 (在非 null 值之前) 或排序在最後 (在非 null 值之後)。根據預設，依 ASC 順序排序時，NULL 值排在最後面，而依 DESC 順序排序時，則排在最前面。

LIMIT number | ALL

此選項會控制查詢傳回的排序資料列數。LIMIT 數字必須是正整數；最大值為 2147483647。

LIMIT 0 不會傳回任何資料列。您可以使用此語法進行測試：查看查詢執行情形 (不顯示任何資料列)，或從資料表傳回資料欄清單。若您使用 LIMIT 0 傳回資料欄清單，則 ORDER BY 子句是多餘的。預設值為 LIMIT ALL。

OFFSET start

此選項會指定先略過 start 之前的資料列數，再開始傳回資料列。OFFSET 數字必須是正整數；最大值為 2147483647。搭配 LIMIT 選項使用時，會先略過 OFFSET 資料列，再開始計算傳回的 LIMIT 資料列。如果未使用 LIMIT 選項，則結果集中的資料列數會減掉略過的資料列數。OFFSET 子句略過的資料列仍須經過掃描，因此使用較大的 OFFSET 值可能會導致效率不佳。

使用須知

請注意以下使用 ORDER BY 子句的預期行為：

- NULL 值會視為「高於」所有其他值。使用預設的遞增排序順序時，NULL 值會排列在最後面。若要變更此行為，請使用 NULLS FIRST 選項。
- 若查詢未包含 ORDER BY 子句，系統傳回的結果集當中就不會有可預測的資料列排列順序。執行相同的查詢兩次，可能會傳回依不同順序排列的結果集。
- LIMIT 和 OFFSET 選項可在沒有 ORDER BY 子句的情況下使用；不過，若要傳回一致的資料列集，請使用這些選項搭配 ORDER BY。
- 在任何平行系統中，例如 Amazon Redshift，當 ORDER BY 子句無法產生唯一排列順序時，資料列的順序便會不確定。也就是說，如果 ORDER BY 表達式產生重複的值，則這些資料列的傳回順序可能與其他系統不同，也可能隨著每次執行 Amazon Redshift 而有所不同。

- Amazon Redshift 不支援 ORDER BY 子句中的字符文字。

ORDER BY 的範例

從 CATEGORY 資料表傳回全部 11 列，並依第二個資料欄 CATGROUP 排序。若結果擁有相同的 CATGROUP 值，則依字元字串的長度排列 CATDESC 資料欄的值。然後，依 CATID 和 CATNAME 欄排序。

```
select * from category order by 2, length(catdesc), 1, 3;
```

catid	catgroup	catname	catdesc
10	Concerts	Jazz	All jazz singers and bands
9	Concerts	Pop	All rock and pop music concerts
11	Concerts	Classical	All symphony, concerto, and choir conce
6	Shows	Musicals	Musical theatre
7	Shows	Plays	All non-musical theatre
8	Shows	Opera	All opera and light opera
5	Sports	MLS	Major League Soccer
1	Sports	MLB	Major League Baseball
2	Sports	NHL	National Hockey League
3	Sports	NFL	National Football League
4	Sports	NBA	National Basketball Association

(11 rows)

從 SALES 資料表傳回選取的欄，並依最高 QTYSOLD 值排序。將結果限制為前 10 個資料列：

```
select salesid, qtysold, pricepaid, commission, saletime from sales
order by qtysold, pricepaid, commission, salesid, saletime desc
limit 10;
```

salesid	qtysold	pricepaid	commission	saletime
15401	8	272.00	40.80	2008-03-18 06:54:56
61683	8	296.00	44.40	2008-11-26 04:00:23
90528	8	328.00	49.20	2008-06-11 02:38:09
74549	8	336.00	50.40	2008-01-19 12:01:21
130232	8	352.00	52.80	2008-05-02 05:52:31
55243	8	384.00	57.60	2008-07-12 02:19:53
16004	8	440.00	66.00	2008-11-04 07:22:31
489	8	496.00	74.40	2008-08-03 05:48:55
4197	8	512.00	76.80	2008-03-23 11:35:33

```
16929 |      8 |    568.00 |    85.20 | 2008-12-19 02:59:33
(10 rows)
```

使用 LIMIT 0 語法會傳回一份資料欄清單，但沒有資料列：

```
select * from venue limit 0;
venueid | venue name | venue city | venue state | venue seats
-----+-----+-----+-----+-----
(0 rows)
```

CONNECT BY 子句

CONNECT BY 子句會指定階層中資料列之間的關係。您可以使用 CONNECT BY，透過將資料表聯結至自身並處理階層式資料，以階層順序選取資料列。例如，您可以使用它來遞迴循環組織結構圖和列出資料。

階層式查詢會以下列順序處理：

1. 如果 FROM 子句具有聯結，則會先處理它。
2. CONNECT BY 子句會受到評估。
3. WHERE 子句會受到評估。

語法

```
[START WITH start_with_conditions]
CONNECT BY connect_by_conditions
```

Note

雖然 START 和 CONNECT 不是保留字，但如果您在查詢中使用 START 和 CONNECT 做為資料表別名，請使用分隔識別碼 (雙引號) 或 AS，以避免在執行階段失敗。

```
SELECT COUNT(*)
FROM Employee "start"
CONNECT BY PRIOR id = manager_id
START WITH name = 'John'
```

```
SELECT COUNT(*)
FROM Employee AS start
CONNECT BY PRIOR id = manager_id
START WITH name = 'John'
```

參數

start_with_conditions

指定階層根資料列的條件

connect_by_conditions

指定階層父項資料列與子資料列之間關係的條件。至少有一個條件必須使用用來參照父資料列的一元運算子來限定。

```
PRIOR column = expression
-- or
expression > PRIOR column
```

運算子

您可以在 CONNECT BY 查詢中使用以下運算子。

LEVEL

傳回階層中目前資料列層級的虛擬資料欄。針對根資料列傳回 1，針對根資料列的子項傳回 2，依此類推。

PRIOR

一元運算子，用於評估階層中目前資料列之父資料列的運算式。

範例

下列範例是 CONNECT BY 查詢，此查詢會傳回直接或間接向 John 報告的員工人數，不超過 4 個層級。

```
SELECT id, name, manager_id
FROM employee
WHERE LEVEL < 4
```

```
START WITH name = 'John'  
CONNECT BY PRIOR id = manager_id;
```

以下為查詢結果。

id	name	manager_id
101	John	100
102	Jorge	101
103	Kwaku	101
110	Liu	101
201	Sofia	102
106	Mateo	102
110	Nikki	103
104	Paulo	103
105	Richard	103
120	Saanvi	104
200	Shirley	104
205	Zhang	104

此範例的資料表定義：

```
CREATE TABLE employee (  
  id INT,  
  name VARCHAR(20),  
  manager_id INT  
);
```

以下是插入到資料表中的資料列。

```
INSERT INTO employee(id, name, manager_id) VALUES  
(100, 'Carlos', null),  
(101, 'John', 100),  
(102, 'Jorge', 101),  
(103, 'Kwaku', 101),  
(110, 'Liu', 101),  
(106, 'Mateo', 102),  
(110, 'Nikki', 103),  
(104, 'Paulo', 103),  
(105, 'Richard', 103),  
(120, 'Saanvi', 104),  
(200, 'Shirley', 104),
```

```
(201, 'Sofía', 102),
(205, 'Zhang', 104);
```

以下是 John 的部門組織結構圖。

子查詢範例

下列範例顯示將子查詢納入 SELECT 查詢的不同方式。請參閱 [JOIN 範例](#)，了解另一個使用子查詢的範例。

SELECT 清單子查詢

以下範例包含 SELECT 清單中的子查詢。此子查詢為純量：它只會傳回一個資料欄和一個值，該值會在從外部查詢傳回的每個資料列的結果中重複出現。查詢會比較子查詢運算的 Q1SALES 值與 2008 年另兩季 (2 和 3) 的銷售數字，如外部查詢所定義。

```
select qtr, sum(pricepaid) as qtrsales,
(select sum(pricepaid)
from sales join date on sales.dateid=date.dateid
where qtr='1' and year=2008) as q1sales
from sales join date on sales.dateid=date.dateid
where qtr in('2','3') and year=2008
group by qtr
order by qtr;
```

```
qtr | qtrsales | q1sales
-----+-----+-----
2   | 30560050.00 | 24742065.00
3   | 31170237.00 | 24742065.00
(2 rows)
```

WHERE 子句子查詢

以下範例包含 WHERE 子句中的資料表子查詢。此子查詢會產生多個資料列。在此情況下，資料列只會包含一個資料欄，但資料表子查詢可包含多個資料欄和資料列，就像任何其他資料表一樣。

查詢會尋找票券銷售量最高的前 10 名賣家。前 10 名清單受到子查詢的限制，會移除居住在有售票場地之城市的使用者。此查詢可透過不同的方式撰寫；例如，子查詢可重寫為主查詢內的聯結。

```
select firstname, lastname, city, max(qtysold) as maxsold
from users join sales on users.userid=sales.sellerid
```



```

where users.city not in(select venuecity from venue)
group by firstname, lastname, city
order by maxsold desc, city desc
limit 10;

```

firstname	lastname	city	maxsold
Noah	Guerrero	Worcester	8
Isadora	Moss	Winooski	8
Kieran	Harrison	Westminster	8
Heidi	Davis	Warwick	8
Sara	Anthony	Waco	8
Bree	Buck	Valdez	8
Evangeline	Sampson	Trenton	8
Kendall	Keith	Stillwater	8
Bertha	Bishop	Stevens Point	8
Patricia	Anderson	South Portland	8

(10 rows)

WITH 子句子查詢

請參閱[WITH 子句](#)。

相互關聯子查詢

以下範例包含 WHERE 子句中的相互關聯子查詢；這類子查詢的資料欄與外部查詢產生的資料欄之間包含一項或多項相互關聯。在此情況下，相互關聯為 `where s.listid=l.listid`。針對外部查詢產生的每個資料列，子查詢會執行以限定或取消限定資料列。

```

select salesid, listid, sum(pricepaid) from sales s
where qtysold=
(select max(numtickets) from listing l
where s.listid=l.listid)
group by 1,2
order by 1,2
limit 5;

```

salesid	listid	sum
27	28	111.00
81	103	181.00
142	149	240.00
146	152	231.00

```
194 | 210 | 144.00
(5 rows)
```

不支援的相互關聯子查詢模式

查詢規劃器會使用一種查詢重寫方法，稱為子查詢解除相互關聯，在 MPP 環境中最佳化數種相互關聯子查詢模式以供執行。有幾種類型的相互關聯子查詢採用了 Amazon Redshift 無法解除相互關聯也不支援的模式。包含下列相互關聯參考的查詢會傳回錯誤：

- 略過查詢區塊的相互關聯參考，也稱為「略過層級相互關聯參考」。例如，在下列查詢中，包含相互關聯參考的區塊和略過的區塊會以 NOT EXISTS 述詞連接：

```
select event.eventname from event
where not exists
(select * from listing
where not exists
(select * from sales where event.eventid=sales.eventid));
```

在此案例中略過的區塊是對 LISTING 資料表的子查詢。相互關聯參考會將 EVENT 和 SALES 資料表相互關聯。

- 來自子查詢的相互關聯參考，它是外部查詢中 ON 子句的一部分：

```
select * from category
left join event
on category.catid=event.catid and eventid =
(select max(eventid) from sales where sales.eventid=event.eventid);
```

ON 子句包含從子查詢中 SALES 對外部查詢中 EVENT 的相互關聯參考。

- Null 敏感的相互關聯會參考 Amazon Redshift 系統資料表。例如：

```
select attrelid
from stv_locks sl, pg_attribute
where sl.table_id=pg_attribute.attrelid and 1 not in
(select 1 from pg_opclass where sl.lock_owner = opcname);
```

- 來自子查詢內的相互關聯參考，當中包含視窗函數。

```
select listid, qtysold
from sales s
where qtysold not in
```

```
(select sum(numtickets) over() from listing l where s.listid=l.listid);
```

- GROUP BY 資料欄中對相互關聯子查詢結果的參考。例如：

```
select listing.listid,
(select count (sales.listid) from sales where sales.listid=listing.listid) as list
from listing
group by list, listing.listid;
```

- 子查詢中使用彙整函數和 GROUP BY 子句的相互關聯參考，會透過 IN 述詞連接至外部查詢。(此限制不會套用至 MIN 和 MAX 彙整函數)。例如：

```
select * from listing where listid in
(select sum(qtysold)
from sales
where numtickets>4
group by salesid);
```

SELECT INTO

選取任何查詢所定義的資料列，然後將它們插入新的資料表中。您可以指定是否建立臨時或永久資料表。

語法

```
[ WITH with_subquery [, ...] ]
SELECT
[ TOP number ] [ ALL | DISTINCT ]
* | expression [ AS output_name ] [, ...]
INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table
[ FROM table_reference [, ...] ]
[ WHERE condition ]
[ GROUP BY expression [, ...] ]
[ HAVING condition [, ...] ]
[ { UNION | INTERSECT | { EXCEPT | MINUS } } [ ALL ] query ]
[ ORDER BY expression
[ ASC | DESC ]
[ LIMIT { number | ALL } ]
[ OFFSET start ]
```

如需有關此命令之參數的詳細資訊，請參閱 [SELECT](#)。

範例

從EVENT資料表中選取所有列，並建立NEWEVENT資料表：

```
select * into newevent from event;
```

將彙總查詢的結果選取為稱為的暫存資料表PROFITS：

```
select username, lastname, sum(pricepaid-commission) as profit
into temp table profits
from sales, users
where sales.sellerid=users.userid
group by 1, 2
order by 3 desc;
```

SET

設定伺服器組態參數的值。使用 SET 命令僅在目前工作階段或交易的持續時間覆寫設定。

使用 [RESET](#) 命令將參數恢復為其預設值。

您可以採用數種方式來變更伺服器組態參數。如需詳細資訊，請參閱[修改伺服器組態](#)。

語法

```
SET { [ SESSION | LOCAL ]
{ SEED | parameter_name } { TO | = }
{ value | 'value' | DEFAULT } |
SEED TO value }
```

以下陳述式會設定工作階段內容變數的值。

```
SET { [ SESSION | LOCAL ]
variable_name { TO | = }
{ value | 'value' }
```

參數

SESSION

指出目前工作階段的設定有效。預設值。

variable_name

指定為工作階段設定的內容變數名稱。

命名慣例是由點分隔的兩部分名稱，例如 identifier.identifier。只允許使用一個點分隔符號。使用符合 Amazon Redshift 標準識別碼規則的 identifier，如需詳細資訊，請參閱 [名稱與識別碼](#)。不允許使用分隔的識別碼。

LOCAL

指出目前交易的設定有效。

SEED TO 值

設定RANDOM函數用於產生隨機數字的內部種子。

SET SEED 會取得介於 0 和 1 之間的數值，並將此數字乘以 $(2^{31}-1)$ 以搭配 [RANDOM 函數](#) 函數使用。如果您在進行多次RANDOM呼叫SETSEED之前使用，會以可預測的順序RANDOM產生數字。

parameter_name

要設定的參數名稱。如需參數的詳細資訊，請參閱 [修改伺服器組態](#)。

值

新參數值。使用單引號將值設定為特定字串。如果使用 SET SEED，此參數會包含 SEED值。

DEFAULT

將參數設定為預設值。

範例

變更目前工作階段的參數

下列範例會設定 datestyle：

```
set datestyle to 'SQL,DMY';
```

設定查詢群組以進行工作負載管理

如果查詢群組在佇列定義中列為叢集WLM組態的一部分，您可以將 QUERY_GROUP 參數設定為列出的查詢群組名稱。後續查詢會指派至相關聯的查詢佇列。QUERY_GROUP 設定在工作階段期間或直到遇到 QUERY_RESET GROUP 命令為止保持有效。

此範例會在查詢群組 'priority' 中執行兩個查詢，然後重設查詢群組。

```
set query_group to 'priority';
select tbl, count(*)from stv_blocklist;
select query, elapsed, substring from svl_qlog order by query desc limit 5;
reset query_group;
```

如需詳細資訊，請參閱[工作負載管理](#)。

變更工作階段的預設身分命名空間

資料庫使用者可以設定 `default_identity_namespace`。此範例說明如何在目前工作階段的持續時間內SET SESSION使用覆寫設定，然後顯示新的身分提供者值。當您搭配 Redshift 和 IAM Identity Center 使用身分提供者時，最常使用此功能。如需搭配 Redshift 使用身分提供者的詳細資訊，請參閱將 [Redshift 與 IAM Identity Center 連線，為使用者提供單一登入體驗](#)。

```
SET SESSION default_identity_namespace = 'MYCO';

SHOW default_identity_namespace;
```

執行命令之後，您可以執行如下所示的GRANT陳述式或CREATE陳述式：

```
GRANT SELECT ON TABLE mytable TO alice;

GRANT UPDATE ON TABLE mytable TO salesrole;

CREATE USER bob password 'md50c983d1a624280812631c5389e60d48c';
```

在此執行個體中，設定預設身分命名空間的效果相當於在每個身分前面加上命名空間。在此範例中，alice 會取代為 MYCO:alice。如需使用 IAM Identity Center 進行 Redshift 組態相關設定的詳細資訊，請參閱 [ALTER SYSTEM](#)和 [ALTER IDENTITY PROVIDER](#)。

設定查詢群組的標籤

QUERY_GROUP 參數會為在SET命令之後於相同工作階段中執行的一或多個查詢定義標籤。反之，此標籤會在執行查詢時記錄，並可用來限制從 STL_QUERY 和 STV_INFLIGHT 系統資料表和 SVL_QLOG 檢視傳回的結果。

```
show query_group;
query_group
```

```

-----
unset
(1 row)

set query_group to '6 p.m.';

show query_group;
query_group
-----
6 p.m.
(1 row)

select * from sales where salesid=500;
salesid | listid | sellerid | buyerid | eventid | dateid | ...
-----+-----+-----+-----+-----+-----+-----
500 | 504 | 3858 | 2123 | 5871 | 2052 | ...
(1 row)

reset query_group;

select query, trim(label) querygroup, pid, trim(querytxt) sql
from stl_query
where label = '6 p.m.';
query | querygroup | pid | sql
-----+-----+-----+-----
57 | 6 p.m. | 30711 | select * from sales where salesid=500;
(1 row)

```

查詢群組標籤是實用的機制，方便用來隔離指令碼中執行的個別查詢或查詢群組。您不需要根據他們的來識別和追蹤查詢IDs；您可以根據他們的標籤來追蹤查詢。

設定產生亂數的 Seed 值

下列範例使用 SEED 選項 SET，讓 RANDOM 函數以可預測的順序產生數字。

首先，傳回三個 RANDOM 整數，而不先設定 SEED 值：

```

select cast (random() * 100 as int);
int4
-----
6
(1 row)

```

```
select cast (random() * 100 as int);
int4
-----
68
(1 row)

select cast (random() * 100 as int);
int4
-----
56
(1 row)
```

現在，將 SEED 值設定為 .25，然後傳回三個額外的RANDOM數字：

```
set seed to .25;

select cast (random() * 100 as int);
int4
-----
21
(1 row)

select cast (random() * 100 as int);
int4
-----
79
(1 row)

select cast (random() * 100 as int);
int4
-----
12
(1 row)
```

最後，將 SEED 值重設為 .25，並確認 RANDOM傳回與先前三個呼叫相同的結果：

```
set seed to .25;

select cast (random() * 100 as int);
int4
-----
21
```



```
(1 row)

select cast (random() * 100 as int);
int4
-----
79
(1 row)

select cast (random() * 100 as int);
int4
-----
12
(1 row)
```

以下範例會設定自訂內容變數。

```
SET app_context.user_id TO 123;
SET app_context.user_id TO 'sample_variable_value';
```

SET SESSION AUTHORIZATION

設定目前工作階段的使用者名稱。

例如，您可以使用 SETSESSIONAUTHORIZATION 命令，以無權限使用者身分暫時執行工作階段或交易，以測試資料庫存取。您必須是資料庫超級使用者才能執行此命令。

語法

```
SET [ LOCAL ] SESSION AUTHORIZATION { user_name | DEFAULT }
```

參數

LOCAL

指出目前交易的設定有效。省略此參數會指出目前工作階段的設定有效。

user_name

要設定的使用者名稱。使用者名稱可做為識別碼或字串常值寫入。

DEFAULT

將工作階段使用者名稱設為預設值。

範例

以下範例會將目前工作階段的使用者名稱設為 `dwuser`：

```
SET SESSION AUTHORIZATION 'dwuser';
```

以下範例會將目前交易的使用者名稱設為 `dwuser`：

```
SET LOCAL SESSION AUTHORIZATION 'dwuser';
```

此範例會將目前工作階段的使用者名稱設為預設使用者名稱：

```
SET SESSION AUTHORIZATION DEFAULT;
```

SET SESSION CHARACTERISTICS

此命令已棄用。

SHOW

顯示伺服器組態參數的目前值。如果SET命令生效，則此值可能特定於目前的工作階段。如需組態參數的清單，請參閱 [組態參考](#)。

語法

```
SHOW { parameter_name | ALL }
```

以下陳述式會顯示工作階段內容變數的目前值。如果該變數不存在，Amazon Redshift 會擲回錯誤。

```
SHOW variable_name
```

參數

`parameter_name`

顯示指定之參數的目前值。

ALL

顯示所有參數的目前值。

variable_name

顯示指定變數的目前值。

範例

以下範例會顯示 query_group 參數的值：

```
show query_group;

query_group

unset
(1 row)
```

以下範例會顯示所有參數與其值的清單：

```
show all;
name          | setting
-----+-----
datestyle     | ISO, MDY
extra_float_digits | 0
query_group   | unset
search_path   | $user,public
statement_timeout | 0
```

下列範例會顯示指定變數的目前值。

```
SHOW app_context.user_id;
```

SHOW COLUMNS

顯示資料表中的資料欄清單，以及某些資料欄屬性。

每個輸出資料列都包含以逗號分隔的資料庫名稱、結構描述名稱、資料表名稱、資料欄名稱、序號位置、資料欄預設值、可為 Null、資料類型、字元最大長度、數值有效位數及備註。如需這些屬性的相關資訊，請參閱 [SVV_ALL_COLUMNS](#)。

如果SHOWCOLUMNS命令會產生超過 10,000 個資料欄，則會傳回錯誤。

語法

```
SHOW COLUMNS FROM TABLE database_name.schema_name.table_name [LIKE 'filter_pattern']  
[LIMIT row_limit ]
```

參數

database_name

包含所要列出資料表的資料庫名稱。

若要在 中顯示資料表 AWS Glue Data Catalog，請指定 (`awsdatacatalog`) 作為資料庫名稱，並確保系統組態 `data_catalog_auto_mount` 設定為 `true`。如需詳細資訊，請參閱 [ALTER SYSTEM](#)。

schema_name

包含所要列出資料表的結構描述名稱。

若要顯示 AWS Glue Data Catalog 資料表，請提供 AWS Glue 資料庫名稱作為結構描述名稱。

table_name

包含所要列出資料欄的資料表名稱。

filter_pattern

有效的 UTF-8 字元表達式，具有與資料表名稱相符的模式。LIKE 選項會執行區分大小寫的比對，以支援下列模式比對元字元：

中繼字元	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。

如果 `filter_pattern` 不包含中繼字元，則模式僅代表字串本身；在這種情況下，其 LIKE 作用與等於運算子相同。

row_limit

傳回的最大資料列數。`row_limit` 可以是 0 到 10,000。

範例

下列範例顯示 Amazon Redshift 資料庫中名為 `dev` 且在結構描述 `public` 和資料表 `tb` 中的資料欄。

```
SHOW COLUMNS FROM TABLE dev.public.tb;
```

```

database_name | schema_name | table_name | column_name | ordinal_position
| column_default | is_nullable | data_type | character_maximum_length |
numeric_precision | remarks
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+-----
dev          | public      | tb         | col         |          1 |
| YES        | integer    |             |             |          32 |

```

下列範例顯示名為 `awsdatacatalog` 的 AWS Glue Data Catalog 資料庫中 `awsdatacatalog` 的資料表，其位於結構描述 `batman` 和資料表 `nation`。輸出限制為 2 個資料列。

```
SHOW COLUMNS FROM TABLE awsdatacatalog.batman.nation LIMIT 2;
```

```

database_name | schema_name | table_name | column_name | ordinal_position
| column_default | is_nullable | data_type | character_maximum_length |
numeric_precision | remarks
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+-----
awsdatacatalog | batman      | nation     | n_nationkey |          1 |
|              | integer    |             |             |             |
awsdatacatalog | batman      | nation     | n_name      |          2 |
|              | character  |             |             |             |

```

SHOW EXTERNAL TABLE

顯示外部資料表的定義，包括資料表屬性和資料欄屬性。您可以使用 `SHOW EXTERNAL TABLE` 陳述式的輸出來重新建立資料表。

如需外部資料表建立的相關資訊，請參閱 [CREATE EXTERNAL TABLE](#)。

語法

```
SHOW EXTERNAL TABLE [external_database].external_schema.table_name [ PARTITION ]
```

參數

external_database

相關聯外部資料庫的名稱。此為選用參數。

external_schema

相關聯外部結構描述的名稱。

table_name

要顯示的資料表名稱。

PARTITION

顯示將分割區新增至資料表定義的ALTER TABLE 陳述式。

範例

以下範例基於外部資料表定義如下：

```
CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (  
    csmallint smallint,  
    cint int,  
    cbigint bigint,  
    cfloat float4,  
    cdouble float8,  
    cchar char(10),  
    cvarchar varchar(255),  
    cdecimal_small decimal(18,9),  
    cdecimal_big decimal(30,15),  
    ctimestamp TIMESTAMP,  
    cboolean boolean,  
    cstring varchar(16383)  
)  
PARTITIONED BY (cdate date, ctime TIMESTAMP)  
STORED AS PARQUET  
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';
```

以下是資料表的SHOWEXTERNALTABLE命令和輸出範

例my_schema.alldatatypes_parquet_test_partitioned。

```
SHOW EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned;
```

```
"CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (  
  csmallint smallint,  
  cint int,  
  cbigint bigint,  
  cfloat float4,  
  cdouble float8,  
  cchar char(10),  
  cvarchar varchar(255),  
  cdecimal_small decimal(18,9),  
  cdecimal_big decimal(30,15),  
  ctimestamp timestamp,  
  cboolean boolean,  
  cstring varchar(16383)  
)  
PARTITIONED BY (cdate date, ctime timestamp)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'  
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'  
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'  
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_test_partitioned';"
```

以下是相同資料表的SHOWEXTERNALTABLE命令和輸出範例，但參數中也指定了資料庫。

```
SHOW EXTERNAL TABLE my_database.my_schema.alldatatypes_parquet_test_partitioned;
```

```
"CREATE EXTERNAL TABLE my_database.my_schema.alldatatypes_parquet_test_partitioned (  
  csmallint smallint,  
  cint int,  
  cbigint bigint,  
  cfloat float4,  
  cdouble float8,  
  cchar char(10),  
  cvarchar varchar(255),  
  cdecimal_small decimal(18,9),  
  cdecimal_big decimal(30,15),  
  ctimestamp timestamp,  
  cboolean boolean,  
  cstring varchar(16383)  
)  
PARTITIONED BY (cdate date, ctime timestamp)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
```

```
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'  
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'  
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';"
```

以下是使用 PARTITION 參數時SHOWEXTERNALTABLE命令和輸出的範例。輸出包含將分割區新增至資料表定義的ALTERTABLE陳述式。

```
SHOW EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned PARTITION;
```

```
"CREATE EXTERNAL TABLE my_schema.alldatatypes_parquet_test_partitioned (  
  csmallint smallint,  
  cint int,  
  cbigint bigint,  
  cfloat float4,  
  cdouble float8,  
  cchar char(10),  
  cvarchar varchar(255),  
  cdecimal_small decimal(18,9),  
  cdecimal_big decimal(30,15),  
  ctimestamp timestamp,  
  cboolean boolean,  
  cstring varchar(16383)  
)  
PARTITIONED BY (cdate date)  
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'  
STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat'  
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.parquet.MapredParquetOutputFormat'  
LOCATION 's3://amzn-s3-demo-bucket/alldatatypes_parquet_partitioned';  
ALTER TABLE my_schema.alldatatypes_parquet_test_partitioned ADD IF NOT  
  EXISTS PARTITION (cdate='2021-01-01') LOCATION 's3://amzn-s3-demo-bucket/  
alldatatypes_parquet_partitioned2/cdate=2021-01-01';  
ALTER TABLE my_schema.alldatatypes_parquet_test_partitioned ADD IF NOT  
  EXISTS PARTITION (cdate='2021-01-02') LOCATION 's3://amzn-s3-demo-bucket/  
alldatatypes_parquet_partitioned2/cdate=2021-01-02';"
```

SHOW DATABASES

顯示來自指定帳戶 ID 的資料庫。

語法

```
SHOW DATABASES FROM
```



```
DATA CATALOG [ ACCOUNT '<id1>', '<id2>', ... ]
[ LIKE '<expression>' ]
[ IAM_ROLE default | 'SESSION' | 'arn:aws:iam::<account-id>:role/<role-name>' ]
```

參數

ACCOUNT '<id1>','<id2>', ...

要從中列出資料庫 AWS Glue Data Catalog 的帳戶。省略此參數表示 Amazon Redshift 應該顯示擁有叢集之帳戶中的資料庫。

LIKE '<expression>'

從資料庫清單中篩選符合您指定運算式的資料庫。此參數支援使用萬用字元 % (百分比) 和 _ (底線) 的模式。

IAM_ROLE 預設 | 'SESSION' | 'arn : aws : iam : : <account-id> : role/<role-name>'

如果您在執行SHOWDATABASES命令時指定與叢集相關聯的IAM角色，Amazon Redshift 會在資料庫上執行查詢時使用該角色的憑證。

指定default關鍵字表示使用設定為預設並與叢集相關聯的IAM角色。

如果您使用聯合身分連線到 Amazon Redshift 叢集，並從使用 [the section called “CREATE DATABASE”](#) 命令建立的外部資料庫存取資料表，請使用 'SESSION'。如需使用聯合身分的範例，請參閱[使用聯合身分管理 Amazon Redshift 對本機資源和 Amazon Redshift Spectrum 外部資料表的存取](#)，其中會說明如何設定聯合身分。

針對叢集用於身分驗證和授權IAM的角色，使用 Amazon Resource Name (ARN)。至少，IAM角色必須具有許可，才能在要存取的 Amazon S3 儲存貯體上執行LIST操作，以及在儲存貯體包含的 Amazon S3 物件上執行GET操作。若要進一步了解從 AWS Glue Data Catalog 為資料共用和使用 IAM_ 建立的資料庫ROLE，請參閱以[取用者身分使用 Lake Formation 管理的資料共用](#)。

下列顯示單一的 IAM_ROLE 參數字串語法ARN。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-name>'
```

您可以連結角色，讓您的叢集擔任另一個IAM角色，可能屬於另一個帳戶。您最多可以鏈結 10 個角色。如需詳細資訊，請參閱[Amazon Redshift Spectrum 中的鏈結IAM角色](#)。

若要擔任此IAM角色，請連接類似下列的IAM許可政策。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AccessSecret",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": "arn:aws:secretsmanager:us-west-2:123456789012:secret:my-
rds-secret-VNenFy"
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetRandomPassword",
        "secretsmanager:ListSecrets"
      ],
      "Resource": "*"
    }
  ]
}
```

如需建立IAM角色以搭配聯合查詢使用的步驟，請參閱 [建立密碼和IAM角色以使用聯合查詢](#)。

Note

不要在鏈結的角色清單中包含空格。

以下顯示鏈結三個角色的語法。

```
IAM_ROLE 'arn:aws:iam::<aws-account-id>:role/<role-1-name>,arn:aws:iam::<aws-
account-id>:role/<role-2-name>,arn:aws:iam::<aws-account-id>:role/<role-3-name>'
```

範例

下列範例會顯示帳戶識別碼 123456789012 中的所有 Data Catalog 資料庫。

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012'
```

```

catalog_id | database_name | database_arn
| type      |              | target_database
|          | location | parameters
-----+-----+-----
+-----+-----+-----
+-----+-----+-----
+-----+-----+-----
123456789012 | database1 | arn:aws:glue:us-east-1:123456789012:database/database1
| Data Catalog |
|          |          |
123456789012 | database2 | arn:aws:glue:us-east-1:123456789012:database/database2
| Data Catalog | arn:aws:redshift:us-
east-1:123456789012:datashare:035c45ea-61ce-86f0-8b75-19ac6102c3b7/database2 |
|

```

以下是示範如何在使用IAM角色的憑證時顯示帳戶 ID 123456789012 中所有 Data Catalog 資料庫的範例。

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012' IAM_ROLE default;
```

```
SHOW DATABASES FROM DATA CATALOG ACCOUNT '123456789012' IAM_ROLE <iam-role-arn>;
```

SHOW MODEL

顯示有關機器學習模型的實用資訊，包括其狀態、用於建立模型的參數，以及具有輸入引數類型的預測函數。您可以使用 `SHOW MODEL` 中的資訊來重新建立模型。如果基礎資料表已變更，`CREATE MODEL` 使用相同的 SQL 陳述式執行會產生不同的模型。傳回的資訊因模型擁有者和具有 `EXECUTE` 權限的使用者 `SHOW MODEL` 而異。`SHOW MODEL` 當模型是從 Amazon Redshift 訓練或模型為 BYOM 模型時，會顯示不同的輸出。

語法

```
SHOW MODEL ( ALL | model_name )
```

參數

ALL

傳回使用者可以使用的所有模型及其結構描述。

model_name

模型的名稱。結構描述中的模型名稱必須是唯一的。

使用須知

SHOW MODEL 命令會傳回下列項目：

- 模型名稱。
- 建立模型所在的結構描述。
- 模型的擁有者。
- 模型建立時間。
- 模型的狀態，例如 READY、TRAINING 或 FAILED。
- 失敗模型的原因訊息。
- 如果模型已完成訓練，則會出現驗證錯誤。
- 衍生非BYOM方法模型所需的預估成本。只有模型的擁有者可以檢視此資訊。
- 使用者指定的參數及其值的清單，特別是下列各項：
 - 指定的TARGET資料欄。
 - 模型類型 AUTO 或 XGBoost。
 - 問題類型，例如 REGRESSION、BINARY_CLASSIFICATION、MULTICLASS_CLASSIFICATION。此參數是特有的AUTO。
 - 建立模型的 Amazon SageMaker 訓練任務或 Amazon SageMaker Autopilot 任務的名稱。您可以使用此任務名稱來尋找有關 Amazon 上模型的詳細資訊 SageMaker。
 - 目標，例如 MSE、F1、準確性。此參數是特有的AUTO。

- 所建立函數的名稱。
- 推論類型 (本機或遠端)。
- 預測函數輸入引數。
- 未攜帶您自己的模型之模型的預測函數輸入引數類型 (BYOM)。
- 預測函數的傳回類型。此參數是特有的BYOM。
- 具有遠端推論之BYOM模型的 Amazon SageMaker 端點名稱。
- IAM 角色。只有模型的擁有者可以看到此項目。
- 使用的 S3 儲存貯體。只有模型的擁有者可以看到此項目。
- 索引 AWS KMS 鍵，如果已提供索引鍵。只有模型的擁有者可以看到此項目。
- 模型可以執行的時間上限。
- 如果模型類型不是 AUTO，Amazon Redshift 也會顯示提供的超參數清單及其值。

您也可以檢視其他目錄資料表SHOWMODEL中由提供的一些資訊，例如 pg_proc。Amazon Redshift 會傳回 pg_proc 目錄資料表中註冊的預測函數相關資訊。此資訊包括用於預測函數的輸入引數名稱及其類型。Amazon Redshift 會在 SHOW MODEL 命令中傳回相同的資訊。

```
SELECT * FROM pg_proc WHERE proname ILIKE '%<function_name>%';
```

範例

下列範例顯示「顯示模型」的輸出。

```
SHOW MODEL ALL;
```

Schema Name	Model Name
public	customer_churn

customer_churn 的擁有者可以看到以下輸出。僅具有 EXECUTE 權限的使用者看不到 IAM 角色、Amazon S3 儲存貯體，以及模式的預估成本。

```
SHOW MODEL customer_churn;
```

Key	Value
Model Name	customer_churn

```

Schema Name          | public
Owner                | 'owner'
Creation Time        | Sat, 15.01.2000 14:45:20
Model State          | READY
validation:F1        | 0.855
Estimated Cost       | 5.7
                     |
TRAINING DATA:      |
Table                | customer_data
Target Column        | CHURN
                     |
PARAMETERS:          |
Model Type           | auto
Problem Type         | binary_classification
Objective            | f1
Function Name        | predict_churn
Function Parameters   | age zip average_daily_spend average_daily_cases
Function Parameter Types | int int float float
IAM Role             | 'iam_role'
KMS Key              | 'kms_key'
Max Runtime          | 36000

```

SHOW DATASHARES

顯示叢集中來自相同帳戶或跨帳戶的輸入和輸出共用。如果您未指定資料共用名稱，Amazon Redshift 會顯示叢集中所有資料庫中的所有資料共用。擁有 ALTER 和 SHARE 權限的使用者可以看到他們擁有權限的共用。

語法

```
SHOW DATASHARES [ LIKE 'namepattern' ]
```

參數

LIKE

選用子句，將指定的名稱模式與資料共用的描述進行比較。使用此子句時，Amazon Redshift 只會顯示名稱符合指定名稱模式的資料共用。

namepattern

請求的資料共用名稱或名稱的一部分會使用萬用字元進行比對。

範例

下列範例會顯示叢集中的輸入和輸出共用。

```
SHOW DATASHARES;
SHOW DATASHARES LIKE 'sales%';
```

share_name	share_owner	source_database	consumer_database	share_type	createdate	is_publicaccessible	share_acl	producer_account	producer_namespace
'salesshare'	100	dev		outbound	2020-12-09 01:22:54.	False		123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d

SHOW PROCEDURE

顯示特定預存程序的定義，包括其簽章。您可以使用的輸出SHOWPROCEDURE來重新建立預存程序。

語法

```
SHOW PROCEDURE sp_name [( [ [ argname ] [ argmode ] argtype [, ...] ] )]
```

參數

sp_name

要顯示的程序的名稱。

[argname] [argmode] argtype

用來識別預存程序的輸入引數類型。或者，您可以包含完整的引數資料類型，包括OUT引數。如果預存程序的名稱是唯一的（亦即不過載），則此為選用部分。

範例

下列範例顯示程序 test_sp12 的定義。

```
show procedure test_sp2(int, varchar);
```

Stored Procedure Definition

```
-----  
CREATE OR REPLACE PROCEDURE public.test_sp2(f1 integer, INOUT f2 character varying, OUT  
  character varying)  
LANGUAGE plpgsql  
AS $_$  
DECLARE  
  out_var alias for $3;  
  loop_var int;  
BEGIN  
  IF f1 is null OR f2 is null THEN  
    RAISE EXCEPTION 'input cannot be null';  
  END IF;  
  CREATE TEMP TABLE etl(a int, b varchar);  
  FOR loop_var IN 1..f1 LOOP  
    insert into etl values (loop_var, f2);  
    f2 := f2 || '+' || f2;  
  END LOOP;  
  SELECT INTO out_var count(*) from etl;  
END;  
$_$
```

(1 row)

SHOW SCHEMAS

顯示資料庫中的結構描述清單，以及一些結構描述屬性。

每個輸出列包含資料庫名稱、結構描述名稱、結構描述擁有者、結構描述類型、結構描述 ACL、來源資料庫和結構描述選項。如需這些屬性的相關資訊，請參閱 [SVV_ALL_SCHEMAS](#)。

如果SHOWSCHEMAS命令可能導致超過 10,000 個結構描述，則會傳回錯誤。

語法

```
SHOW SCHEMAS FROM DATABASE database_name [LIKE 'filter_pattern'] [LIMIT row_limit ]
```

參數

database_name

包含所要列出資料表的資料庫名稱。

若要在中顯示資料表 AWS Glue Data Catalog，請指定（awsdatacatalog）作為資料庫名稱，並確保系統組態 data_catalog_auto_mount 設定為 true。如需詳細資訊，請參閱 [ALTER SYSTEM](#)。

filter_pattern

有效的 UTF-8 字元表達式，具有符合結構描述名稱的模式。LIKE 選項會執行區分大小寫的比對，以支援下列模式比對元字元：

中繼字元	描述
%	比對任何 0 的序列或更多字元。
_	比對任一個單一字元。

如果 filter_pattern 不包含中繼字元，則模式只會代表字串本身；在這種情況下，LIKE 的作用與等於運算子相同。

row_limit

傳回的最大資料列數。row_limit 可以是 0 到 10,000。

範例

下面的範例會顯示名為 dev 的 Amazon Redshift 資料庫的結構描述。

```
SHOW SCHEMAS FROM DATABASE dev;
```

```

database_name |      schema_name      | schema_owner | schema_type |      schema_acl
              | source_database | schema_option
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
dev          | pg_automv           |              | local      |
              |                    |              |            |
dev          | pg_catalog          |              | local      | jpuser=UC/
jpuser~=U/jpuser |                    |              |            |
dev          | public              |              | local      | jpuser=UC/
jpuser~=UC/jpuser |                    |              |            |
dev          | information_schema  |              | local      | jpuser=UC/
jpuser~=U/jpuser |                    |              |            |

```

```
dev | schemad79cd6d93bf043 | 1 | local |
```

下列範例顯示名為 `awsdatacatalog` 的 AWS Glue Data Catalog 資料庫中的結構描述。輸出資料列數目上限為 5。

```
SHOW SCHEMAS FROM DATABASE awsdatacatalog LIMIT 5;
```

```

database_name | schema_name | schema_owner | schema_type | schema_acl |
source_database | schema_option
-----+-----+-----+-----+-----
+-----+-----
awsdatacatalog | 000_too_many_glue_db | | EXTERNAL | |
|
awsdatacatalog | 123_default | | EXTERNAL | |
|
awsdatacatalog | adhoc | | EXTERNAL | |
|
awsdatacatalog | all_shapes_10mb | | EXTERNAL | |
|
awsdatacatalog | all_shapes_1g | | EXTERNAL | |
|

```

SHOW TABLE

顯示資料表的定義，包括資料表屬性、資料表條件限制、資料欄屬性和資料欄條件限制。您可以使用 `SHOWTABLE` 陳述式的輸出來重新建立資料表。

如需建立資料表的相關資訊，請參閱 [CREATE TABLE](#)。

語法

```
SHOW TABLE [schema_name.]table_name
```

參數

`schema_name`

(選擇性) 相關結構描述的名稱。

`table_name`

要顯示的資料表名稱。

範例

以下是資料表 `sales` 的 `SHOWTABLE` 輸出範例。

```
show table sales;
```

```
CREATE TABLE public.sales (  
  salesid integer NOT NULL ENCODE az64,  
  listid integer NOT NULL ENCODE az64 distkey,  
  sellerid integer NOT NULL ENCODE az64,  
  buyerid integer NOT NULL ENCODE az64,  
  eventid integer NOT NULL ENCODE az64,  
  dateid smallint NOT NULL,  
  qtysold smallint NOT NULL ENCODE az64,  
  pricepaid numeric(8,2) ENCODE az64,  
  commission numeric(8,2) ENCODE az64,  
  saletime timestamp without time zone ENCODE az64  
)  
DISTSTYLE KEY SORTKEY ( dateid );
```

以下是結構描述 `category` 中資料表 `SHOWTABLE` 輸出的範例。

```
show table public.category;
```

```
CREATE TABLE public.category (  
  catid smallint NOT NULL distkey,  
  catgroup character varying(10) ENCODE lzo,  
  catname character varying(10) ENCODE lzo,  
  catdesc character varying(50) ENCODE lzo  
) DISTSTYLE KEY SORTKEY ( catid );
```

下列範例會建立具有主索引鍵的資料表 `foo`。

```
create table foo(a int PRIMARY KEY, b int);
```

`SHOW TABLE` 結果會顯示包含 `foo` 資料表所有屬性的建立陳述式。

```
show table foo;
```

```
CREATE TABLE public.foo ( a integer NOT NULL ENCODE az64, b integer ENCODE az64,  
PRIMARY KEY (a) ) DISTSTYLE AUTO;
```

SHOW TABLES

顯示結構描述中的資料表清單，以及一些資料表屬性。

每個輸出列都包含資料庫名稱、結構描述名稱、資料表名稱、資料表類型、資料表 ACL 和備註。如需這些屬性的相關資訊，請參閱 [SVV_ALL_TABLES](#)。

如果 SHOWTABLES 命令會產生超過 10,000 個資料表，則會傳回錯誤。

語法

```
SHOW TABLES FROM SCHEMA database_name.schema_name [LIKE 'filter_pattern']  
[LIMIT row_limit ]
```

參數

database_name

包含所要列出資料表的資料庫名稱。

若要在中顯示資料表 AWS Glue Data Catalog，請指定 (`awsdatacatalog`) 作為資料庫名稱，並確保系統組態 `data_catalog_auto_mount` 設定為 `true`。如需詳細資訊，請參閱 [ALTER SYSTEM](#)。

schema_name

包含所要列出資料表的結構描述名稱。

若要顯示 AWS Glue Data Catalog 資料表，請提供 AWS Glue 資料庫名稱作為結構描述名稱。

filter_pattern

有效的 UTF-8 字元表達式，具有與資料表名稱相符的模式。LIKE 選項會執行區分大小寫的比對，以支援下列模式比對元字元：

中繼字元	描述
%	比對任何 0 的序列或更多字元。

中繼字元	描述
_	比對任一個單一字元。

如果 `filter_pattern` 不包含中繼字元，則模式只會代表字串本身；在這種情況下，其LIKE作用與等於運算子相同。

`row_limit`

傳回的最大資料列數。`row_limit` 可以是 0 到 10,000。

範例

下列範例顯示 Amazon Redshift 資料庫中名為 `dev` 且在結構描述 `public` 中的資料表。

```
SHOW TABLES FROM SCHEMA dev.public;
```

database_name	schema_name	table_name	table_type	table_acl	remarks
dev	public	tb	TABLE		
dev	public	tb2	TABLE		
dev	public	tb3	TABLE		

下列範例顯示 結構描述 中名為 的 AWS Glue Data Catalog 資料庫中 `awsdatacatalog` 的資料表 `batman`。

```
SHOW TABLES FROM SCHEMA awsdatacatalog.batman;
```

database_name	schema_name	table_name	table_type	table_acl	remarks
awsdatacatalog	batman	nation	EXTERNAL		
awsdatacatalog	batman	part	EXTERNAL		
awsdatacatalog	batman	partsupp	EXTERNAL		
awsdatacatalog	batman	region	EXTERNAL		
awsdatacatalog	batman	supplier	EXTERNAL		
awsdatacatalog	batman	automount_nation	EXTERNAL		

SHOW VIEW

顯示檢視的定義，包括具體化視觀表和近期繫結檢視。您可以使用 SHOW VIEW 陳述式的輸出來重新建立檢視。

語法

```
SHOW VIEW [schema_name.]view_name
```

參數

schema_name

(選擇性) 相關結構描述的名稱。

view_name

要顯示的檢視名稱。

範例

以下為 LA_Venues_v 檢視的檢視定義：

```
create view LA_Venues_v as select * from venue where venuecity='Los Angeles';
```

以下是先前定義之檢視的SHOWVIEW命令和輸出範例。

```
show view LA_Venues_v;
```

```
SELECT venue.venueid,  
venue.venueName,  
venue.venueCity,  
venue.venueState,  
venue.venueSeats  
FROM venue WHERE ((venue.venueCity)::text = 'Los Angeles'::text);
```

以下為結構描述 public 中檢視 public.Sports_v 的檢視定義。

```
create view public.Sports_v as select * from category where catgroup='Sports';
```

以下是先前定義之檢視的SHOWVIEW命令和輸出範例。

```
show view public.Sports_v;
```

```
SELECT category.catid,  
category.catgroup,  
category.catname,  
category.catdesc  
FROM category WHERE ((category.catgroup)::text = 'Sports'::text);
```

START TRANSACTION

BEGIN 函數的同義詞。

請參閱 [BEGIN](#)。

TRUNCATE

刪除資料表中的所有資料列而不執行資料表掃描：此操作是不合格DELETE操作的更快替代方案。若要執行TRUNCATE命令，您必須擁有 TRUNCATE TABLE 許可、資料表的擁有者或超級使用者。若要授予截斷資料表的權限，請使用 [GRANT](#) 命令。

TRUNCATE 比更有效率DELETE，而且不需要 VACUUM和 ANALYZE。不過，請注意 會 TRUNCATE遞交執行交易的。

語法

```
TRUNCATE [ TABLE ] table_name
```

此命令也適用於具體化視觀表。

```
TRUNCATE materialized_view_name
```

參數

TABLE

選用的關鍵字。

table_name

暫時性或持久性資料表。只有資料表的擁有者或超級使用者可將它截斷。

您可以截斷任何資料表，包括外部索引鍵限制條件中參考的資料表。

您不需要在截斷資料表之後將它清空。

materialized_view_name

具體化視觀表。

您可以截斷用於 [將擷取串流至具體化視觀表](#) 的具體化視觀表。

使用須知

TRUNCATE 命令會遞交執行該命令的交易；因此，您無法復原TRUNCATE操作，且TRUNCATE命令可能會在遞交時遞交其他操作。

範例

使用 TRUNCATE命令從CATEGORY資料表中刪除所有資料列：

```
truncate category;
```

嘗試復原TRUNCATE操作：

```
begin;

truncate date;

rollback;

select count(*) from date;
count
-----
0
(1 row)
```

DATE 資料表在ROLLBACK命令之後會保持空白，因為TRUNCATE命令會自動遞交。

下列範例使用 TRUNCATE 命令，從具體化檢視中刪除所有資料列。

```
truncate my_materialized_view;
```

它會刪除具體化視觀表中的所有記錄，並保持具體化視觀表及其結構描述不變。在查詢中，具體化視觀表名稱是一個範例。

UNLOAD

使用 Amazon S3 伺服器端加密 (SSE-S3)JSON，將查詢結果卸載到一個或多個文字或 Amazon S3 上的 Apache 實木複合地板檔案。您也可以指定伺服器端加密 AWS Key Management Service 使用客戶管理的金鑰進行 key (SSE-KMS) 或用戶端加密。

依預設，卸載檔案的格式為以豎線分隔 (|) 的文字。

您可以通過設置參數來管理 Amazon S3 上的文件大小，並通過擴展名管理文件 MAXFILESIZE 數量。確定 S3 IP 範圍已新增至您的允許清單。若要進一步了解所需的 S3 IP 範圍，請參閱[網路隔離](#)。

您可以將 Amazon Redshift 查詢的結果以 Apache Parquet 卸載至您的 Amazon S3 資料湖，這是一種用於分析的高效率開放單欄式儲存格式。相較於文字格式，Parquet 格式的卸載速度提升達 2 倍，並且在 Amazon S3 中使用的儲存空間減少達 6 倍。這可以讓您將已在 Amazon S3 中執行的資料轉換和資料充實，以開放的格式儲存到您的 Amazon S3 資料湖。然後，您可以使用 Redshift 頻譜和其他分析數據 AWS Amazon Athena，Amazon 和 Amazon EMR 等服務 SageMaker。

若要取得有關使用 UNLOAD 指令的更多資訊和範例案例，請參閱 [〈〉 在 Amazon Redshift 卸載數據](#)。

所需的權限和許可

若要讓 UNLOAD 命令成功，至少需要資料庫中資料的 SELECT 權限，以及寫入 Amazon S3 位置的權限。所需的權限與 COPY 命令類似。如需有關 COPY 指令權限的資訊，請參閱[存取其他 AWS 資源的許可](#)。

語法

```
UNLOAD ('select-statement')  
TO 's3://object-path/name-prefix'  
authorization  
[ option, ...]
```

```

where authorization is
IAM_ROLE { default | 'arn:aws:iam::<AWS #-id-1>:role/<role-name>[,arn:aws:iam::<AWS #
#-id-2>:role/<role-name>][,...]' }

where option is
| [ FORMAT [ AS ] ] CSV | PARQUET | JSON
| PARTITION BY ( column_name [, ... ] ) [ INCLUDE ]
| MANIFEST [ VERBOSE ]
| HEADER
| DELIMITER [ AS ] 'delimiter-char'
| FIXEDWIDTH [ AS ] 'fixedwidth-spec'
| ENCRYPTED [ AUTO ]
| BZIP2
| GZIP
| ZSTD
| ADDQUOTES
| NULL [ AS ] 'null-string'
| ESCAPE
| ALLOWOVERWRITE
| CLEANPATH
| PARALLEL [ { ON | TRUE } | { OFF | FALSE } ]
| MAXFILESIZE [AS] max-size [ MB | GB ]
| ROWGROUPSIZE [AS] size [ MB | GB ]
| REGION [AS] 'aws-region' }
| EXTENSION 'extension-name'

```

參數

('select-statement')

SELECT查詢。查詢的結果會卸載。在大多數情況下，通過在查詢中指定 BY 子句，以排序順序卸載數據是值得的。ORDER此方式可節省重新載入資料時，排序資料所需的時間。

查詢必須用單引號括住，如下所示：

```
('select * from venue order by venueid')
```

Note

如果您的查詢包含引號 (例如為了將常值括住)，請將常值放在兩組單引號內，您也必須將查詢放在單引號之間：

```
('select * from venue where venuestate='NV''')
```

TO 's3://object-path/name-prefix'

Amazon S3 上 Amazon Redshift 寫入輸出檔案物件的位置的完整路徑 (包括儲存貯體名稱) , 包括資訊清單檔案 (如果MANIFEST已指定)。物件名稱前面會加上 name-prefix。如果您使用 PARTITION BY , 會自動視需要將一個斜線 (/) 新增到 name-prefix 值的結尾。為了增加安全性 , 請使用UNLOAD連接連接到 Amazon S3。HTTPS依預設 , UNLOAD會為每個磁碟片段寫入一或多個檔案。UNLOAD將切片編號和零件號碼附加到指定的名稱前置詞 , 如下所示 :

<object-path>/<name-prefix><slice-number>_part_<part-number>.

如果MANIFEST指定 , 資訊清單檔案會寫入如下 :

<object_path>/<name_prefix>manifest.

如果PARALLEL已指定OFF , 資料檔案會寫入如下 :

<object_path>/<name_prefix><part-number>.

UNLOAD使用 Amazon S3 伺服器端加密 (SSE) 自動建立加密檔案 , 包括資訊清單檔案 (如果使MANIFEST用的話)。此命COPY令會在載入作業期間自動讀取伺服器端加密檔案。您可以使用 Amazon S3 主控台或API從儲存貯體透明地下載伺服器端加密檔案。如需詳細資訊 , 請參閱[使用伺服器端加密保護資料](#)。

若要使用 Amazon S3 用戶端加密 , 請指定ENCRYPTED選項。

 Important

REGION當 Amazon S3 存儲桶不在同一個時 , 是必需的 AWS 區域 作為 Amazon Redshift 數據庫。

authorization

此命UNLOAD令需要授權才能將資料寫入 Amazon S3。該UNLOAD命令使用命COPY令用於授權的參數相同。如需詳細資訊 , 請參閱COPY指令[授權參數](#)語法參考中的。

`IAM_ROLE` {默認 | 'ARN : AWN : IAM : : *<AWS ##-id-1>* : 角色/*<role-name>*'

使用預設關鍵字可讓 Amazon Redshift 使用設定為預設並在 UNLOAD 命令執行時與叢集關聯的 IAM 角色。

將 Amazon 資源名稱 (ARN) 用於叢集用於身份驗證和授權的 IAM 角色。如果指定 `IAM_ROLE` , 則無法使用 `ACCESS_KEY_ID` 和 `SECRET_ACCESS_KEY_TOKEN`、`SESSION_` 或 `CREDENTIALS`。該 `IAM_ROLE` 可以鏈接。如需詳細資訊, 請參閱 Amazon Redshift 管理指南中的 [鏈結IAM角色](#)。

`FORMAT`[如] `CSV` | `PARQUET` | `JSON`

關鍵字, 用來指定取代預設格式的卸載格式。

如果 `CSV`, 請使用逗號 (,) 字元作為預設分隔符號, 以 `CSV` 格式卸載至文字檔案。如果欄位包含分隔符號、雙引號、新行字元或歸位字元, 已卸載檔案中的欄位便會括在雙引號中。資料欄位中的雙引號會使用額外的雙引號進行逸出。卸載零個資料列時, Amazon Redshift 可能會寫入空的 Amazon S3 物件。

何時 `PARQUET`, 卸載到 Apache 實木複合地板 1.0 版格式的文件。依預設, 會使用 `SNAPPY` 壓縮來壓縮每個資料列群組。如需 Apache Parquet 格式的相關資訊, 請參閱 [Parquet](#)。

何時 `JSON`, 卸載到每一行都包含一個 `JSON` 對象的 `JSON` 文件, 代表查詢結果中的完整記錄。當查詢結果包含 `SUPER` 資料行 `JSON` 時, Amazon Redshift 支援以巢狀方式寫入。若要建立有效的 `JSON` 物件, 查詢中每個資料行的名稱必須是唯一的。在 `JSON` 檔案中, 布林值會卸載為 `t` 或 `f`, 且 `NULL` 值會卸載為 `null`。卸載零個資料列時, Amazon Redshift 不會寫入 Amazon S3 物件。

`FORMAT` 和 `AS` 關鍵字是選擇性的。您不能 `CSV` 與 `FIXEDWIDTH` 或一起使用 `ADDQUOTES`。您不能 `PARQUET` 與 `DELIMITER`、`FIXEDWIDTH`、`NULL AS` `ADDQUOTES`、`ESCAPE`、`HEADERGZIPBZIP2`、或搭配使用 `ZSTD`。`PARQUET` 只有 `ENCRYPTED` 在伺服器端加密時才支援 AWS Key Management Service 鍵 (`SSE-KMS`)。您無法搭 `JSON` 配 `DELIMITER`、`HEADER`、`FIXEDWIDTH` `ADDQUOTES` `ESCAPE`、或 `NULL AS` 使用。

`PARTITION` 通過 (列名 [, ...]) `INCLUDE`

指定卸載操作的分割區索引鍵。UNLOAD 依照 Apache Hive 慣例, 自動根據分割區索引鍵值將輸出檔案分割為分割區資料夾。例如, 屬於 2019 年分割區和九月分割區的 Parquet 檔案便會具有以下字首: `s3://amzn-s3-demo-bucket/my_prefix/year=2019/month=September/000.parquet`。

`column_name` 的值必須是正在卸載查詢結果中的資料行。

如果您使用 `INCLUDE` 選項指定 `PARTITION BY`, 則不會從卸載的檔案中移除磁碟分割欄。

Amazon Redshift 不支持 BY 子句中的字符串文PARTITION字。

MANIFEST [VERBOSE]

建立資訊清單檔案，以明確列出UNLOAD處理程序所建立之資料檔案的詳細資訊。資訊清單是JSON格式的文字檔案，其URL中列出寫入 Amazon S3 的每個檔案。

如果MANIFEST使用VERBOSE選項指定，資訊清單會包含下列詳細資訊：

- 每個欄的欄名稱和資料類型 CHARVARCHAR，以及用於、或NUMERIC資料類型的維度。對於CHAR和VARCHAR數據類型，維度是長度。對於DECIMAL或NUMERIC資料類型，尺寸為精確度和刻度。
- 卸載至每個檔案的資料列計數。如果已指定HEADER選項，列計數會包含標題行。
- 卸載的所有檔案的檔案大小總計以及卸載至所有檔案的資料列計數總計。如果已指定HEADER選項，列計數會包含標題行。
- 作者。作者一律是 "Amazon Redshift"。

您VERBOSE只能指定下列項目MANIFEST。

資訊清單檔案會寫入與卸載檔案相同的 Amazon S3 路徑字首，其格式為

<object_path_prefix>manifest。例如，如果UNLOAD指定 Amazon S3 路徑前綴 's3://amzn-s3-demo-bucket/venue_'，則資訊清單檔案位置為 's3://amzn-s3-demo-bucket/venue_manifest'。

HEADER

在每個輸出檔案上方新增包含資料欄名稱的標題行。文字轉換選項 (例如 CSVDELIMITER、ADDQUOTESESCAPE、和) 也會套用至標題行。您不能使HEADER用FIXEDWIDTH。

DELIMITERS '分隔符字符'

指定用於分隔輸出檔案中欄位的單一ASCII字元，例如直線字元 (|)、逗號 (,) 或定位點 (\t)。文字檔的預設分隔符號為縱線字元。CSV檔案的預設分隔符號是逗號字元。AS 關鍵字為選用。您不能使DELIMITER用FIXEDWIDTH。如果資料包含分隔符號字元，則需要指定逸出分隔符號的ESCAPE選項，或ADDQUOTES使用將資料用雙引號括住。或者，指定資料中未包含的分隔符號。

FIXEDWIDTH'固定寬度規格'

將資料卸載至每個欄寬是固定長度 (而不是以分隔字元隔開) 的檔案。fixedwidth_spec 是字串，指定資料欄數和欄寬。AS 關鍵字為選用。因為FIXEDWIDTH不會截斷資料，所以UNLOAD陳述式中每個資料行的規格必須至少與該資料行的最長項目長度相同。fixedwidth_spec 的格式如下所示：

```
'colID1:colWidth1,colID2:colWidth2, ...'
```

您不能FIXEDWIDTH與DELIMITER或一起使用HEADER。

ENCRYPTED [AUTO]

指定 Amazon S3 上的輸出檔案將使用 Amazon S3 伺服器端加密或用戶端加密進行加密。如果 MANIFEST 指定，資訊清單檔案也會加密。如需詳細資訊，請參閱[卸載加密的資料檔案](#)。如果您未指定 ENCRYPTED 參數，請使用 Amazon S3 伺服器端加密 UNLOAD 自動建立加密檔案 AWS-受管加密金鑰 (SSE-S3)。

對於 ENCRYPTED，您可能想要使用伺服器端加密與 AWS KMS 鍵 (SSE-KMS)。如果是這種情況，請使用 [KMS_KEY_ID](#) 參數來提供金鑰 ID。您不能將 [CREDENTIALS](#) 參數與 KMS_KEY_ID 參數搭配使用。如果您使用 KMS_KEY_ID 執行資料 UNLOAD 命令，則可以在不指定索引鍵的情況下對相同資料執行 COPY 作業。

如要搭配客戶提供的對稱金鑰使用用戶端加密卸載至 Amazon S3，請透過以下兩種方式中的其中一種來提供金鑰。如要提供金鑰，請使用 [MASTER_SYMMETRIC_KEY](#) 參數，或是 [CREDENTIALS](#) 登入資料字串的 master_symmetric_key 部分。如果您使用根對稱金鑰卸載資料，請務必在對加密資料執行 COPY 作業時提供相同的金鑰。

UNLOAD 使用客戶提供的金鑰 (SSE-C) 不支援 Amazon S3 伺服器端加密。

如果使 ENCRYPTED AUTO 用，則指 UNLOAD 令會擷取預設值 AWS KMS 在目標 Amazon S3 儲存貯體屬性上加密金鑰，並使用 AWS KMS 索引鍵。如果值區沒有預設值 AWS KMS 加密金鑰，使用 Amazon Redshift 伺服器端加密功能，UNLOAD 自動建立加密檔案 AWS-受管加密金鑰 (SSE-S3)。您不能將此選項與 KMS_KEY_ID、_ 或包含主對稱密鑰的 MASTER_SYMMETRIC_KEY CREDENTIALS 一起使用。

KMSKEY_ID '密鑰識別碼'

指定的密鑰 ID AWS Key Management Service (AWS KMS) 用於在 Amazon S3 上加密資料檔案的金鑰。如需詳細資訊，請參閱[什麼是 AWS Key Management Service?](#) 如果指定 KMS_KEY_ID，也必須指定 [ENCRYPTED](#) 參數。如果指定 KMS_KEY_ID，則無法使用 CREDENTIALS 參數進行驗證。請改用 [IAM_ROLE](#) 或 [ACCESS_KEY_ID and SECRET_ACCESS_KEY](#)。

MASTERSYMMETRIC_KEY '根密鑰'

指定要用來加密 Amazon S3 上資料檔案的根對稱金鑰。如果指定 MASTER_SYMMETRIC_KEY，也必須指定 [ENCRYPTED](#) 參數。您不能將 MASTER_SYMMETRIC_KEY 與 CREDENTIALS 參數搭配使用。如需詳細資訊，請參閱[從 Amazon S3 載入加密的資料檔案](#)。

BZIP2

將資料卸載至每個分割中的一個或多個 bzip2 壓縮檔。產生的每個檔案都會附加 .bz2 副檔名。

GZIP

將資料卸載至每個分割中的一個或多個 gzip 壓縮檔。產生的每個檔案都會附加 .gz 副檔名。

ZSTD

將資料卸載至每個分割中的一個或多個 Zstandard 壓縮檔。產生的每個檔案都會附加 .zst 副檔名。

ADDQUOTES

在每個卸載的資料欄位前後加上引號，如此 Amazon Redshift 就能卸載包含分隔符號本身的資料值。例如，如果分隔符號是逗號，您可以成功卸載並重新載入下列資料：

```
"1","Hello, World"
```

如果沒有新增雙引號，Hello, World 字串將會剖析為兩個不同欄位。

某些輸出格式不支持ADDQUOTES。

如果使用ADDQUOTES，則必須REMOVEQUOTES在中指定COPY如果重新載入資料。

NULLAS '空字符串'

指定一個字串，代表卸載檔案中的 null 值。若使用此選項，所有輸出檔案都會包含指定的字串，用以取代所選取資料中找到的任何 null 值。若未指定此選項，則 null 值會卸載為：

- 分隔符號輸出的零長度字串
- 固定寬度輸出的空白字串

若針對固定寬度卸載指定 null 字串，而輸出資料欄的寬度小於 null 字串的寬度，則會發生下列行為：

- 非字元資料欄的輸出為空欄位
- 針對字元資料欄回報錯誤

與其他使用者定義字串代表空值的資料類型不同，Amazon Redshift 會使用格式匯出SUPER資料欄，並根據JSON格式決定將其表示為空值。JSON因此，SUPER資料欄會忽略指UNLOAD令中使用的 NULL [AS] 選項。

ESCAPE

對於CHAR和分隔的卸載文件中的VARCHAR列，轉義字符 (\) 被放置在每次出現以下字符之前：

- 換行：\n
- 歸位字元：\r
- 針對卸載資料指定的分隔符號字元。
- 逸出字元：\
- 引號字元："或 ' (如果指令中同時ESCAPE指定了 UNLOAD AND)。ADDQUOTES

Important

如果您使用COPY帶ESCAPE選項的 a 載入資料，您還必須使用指UNLOAD令指定ESCAPE選項，以產生往復式輸出檔案。同樣，如果您UNLOAD使用該ESCAPE選項，則需要COPY在相同的數據ESCAPE時使用。

ALLOWOVERWRITE

依預設，如UNLOAD果找到可能會覆寫的檔案，就會失敗。如果ALLOWOVERWRITE已指定，則UNLOAD會覆寫現有檔案，包括資訊清單檔案。

CLEANPATH

在將檔案卸載到指定位置之前，CLEANPATH此選項會移除 TO 子句中指定之 Amazon S3 路徑中的現有檔案。

如果您包含 PARTITION BY 子句，則僅從分區文件夾中刪除現有文件，以接收UNLOAD操作生成的新文件。

您必須具有 Amazon S3 儲存貯體的 s3:DeleteObject 許可。如需詳細資訊，請參閱《Amazon Simple Storage Service 使用者指南》中的 [Amazon S3 的政策和許可](#)。您使用該CLEANPATH 選項刪除的文件將被永久刪除，無法恢復。如果目標 Amazon S3 儲存貯體已啟用版本控制，則UNLOAD使用該CLEANPATH選項不會移除先前版本的檔案。

如果您指定選CLEANPATH項，則無法指定ALLOWOVERWRITE選項。

PARALLEL

依預設，UNLOAD會根據叢集中的磁碟片段數目，parallel 寫入資料到多個檔案。預設選項為「開啟」或TRUE。如果PARALLEL是OFF或FALSE，依序UNLOAD寫入一個或多個數據文件，完全根

據 ORDER BY 子句排序，如果使用一個。資料檔案大小上限為 6.2 GB。因此，例如，如果卸載 13.4 GB 的資料，UNLOAD會建立以下三個檔案。

```
s3://amzn-s3-demo-bucket/key000    6.2 GB
s3://amzn-s3-demo-bucket/key001    6.2 GB
s3://amzn-s3-demo-bucket/key002    1.0 GB
```

Note

該UNLOAD命令被設計為使用 parallel 處理。我們建議在大多數情況下保持PARALLEL啟用狀態，特別是如果使用COPY命令來載入表格的檔案。

MAXFILESIZE[如] 最大尺寸 [MB | GB]

指定在 Amazon S3 中UNLOAD建立的檔案大小上限。指定介於 5 MB 和 6.2 GB 之間的小數值。AS 關鍵字為選用。預設單位為 MB。如果MAXFILESIZE未指定，預設檔案大小上限為 6.2 GB。資訊清單檔案的大小 (如果使用的話) 不會受到影響MAXFILESIZE。

ROWGROUPSIZE[如] 大小 [MB | GB]

指定資料列群組的大小。選擇較大的大小可減少資料列群組的數目，進而減少網路通訊量。請指定介於 32 MB 到 128 MB 之間的整數值。AS 關鍵字為選用。預設單位為 MB。

如果ROWGROUPSIZE未指定，則預設大小為 32 MB。若要使用此參數，儲存格式必須是鑲木地板，且節點類型必須是 ra3.4xlarge、ra3.16xlarge 或 dc2.8xlarge。

REGION[AS] 'aws-region'

指定 AWS 區域 目標 Amazon S3 儲存桶所在的位置。REGION對於不在UNLOAD相同的 Amazon S3 儲存桶是必需的 AWS 區域 作為 Amazon Redshift 數據庫。

aw_ 區域的值必須符合 AWS 列於 [Amazon Redshift 區域和端點](#) 資料表中的區域 AWS 一般參考。

依預設，UNLOAD假設目標 Amazon S3 儲存貯體位於同一個儲存貯體 AWS 區域 作為 Amazon Redshift 數據庫。

EXTENSION'擴充功能名稱'

指定要附加到卸載檔案名稱的副檔名。Amazon Redshift 不會執行任何驗證，因此您必須確認指定的副檔名是否正確。如果您使用的是壓縮方法，例如GZIP，您仍然必須.gz在擴展參數中指定。如

果您不提供任何副檔名，Amazon Redshift 不會在文件名中添加任何內容。如果您指定壓縮方法而不提供副檔名，Amazon Redshift 只會將壓縮方法的副檔名新增至檔案名稱。

使用須知

用ESCAPE於所有分隔文字UNLOAD作業

UNLOAD使用分隔符號時，您的資料可以包含該分隔符號或ESCAPE選項說明中列出的任何字元。在此情況下，您必須搭配UNLOAD陳述式使用ESCAPE選項。如果不搭配使用選ESCAPE項UNLOAD，使用卸載資料的後續COPY作業可能會失敗。

Important

我們強烈建議您始終ESCAPE與UNLOAD和COPY語句一起使用。如果您確定資料未包含任何分隔符號或可能需要逸出的其他字元，則為例外。

浮點數精確度喪失

您可能遇到後續卸載並重新載入的浮點資料精確度喪失的情況。

Limit 子句

查SELECT詢不能在外部使用LIMIT子句SELECT。例如，下列UNLOAD陳述式會失敗。

```
unload ('select * from venue limit 10')
to 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

請改用巢狀LIMIT子句，如下列範例所示。

```
unload ('select * from venue where venueid in
(select venueid from venue order by venueid desc limit 10)')
to 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

您也可以使用SELECT... INTO 或 CREATE TABLE AS 使用LIMIT子句填入資料表，然後從該資料表中卸載。

卸載數GEOMETRY據類型的列

您只能將GEOMETRY欄卸載為文字或CSV格式。您無法使用該FIXEDWIDTH選項卸載GEOMETRY數據。資料會以十六進位形式的延伸已知二進位 (EWKB) 格式卸載。如果資料大小超過 4 MB，則會發生警告，因為資料稍後無法載入到資料表中。

卸載數HLLSKETCH據類型的列

您只能將HLLSKETCH欄卸載為文字或CSV格式。您無法使用該FIXEDWIDTH選項卸載HLLSKETCH數據。對於密集 HyperLogLog 草圖，資料會以 Base64 格式卸載，或以稀疏 HyperLogLog 草圖的JSON格式卸載資料。如需詳細資訊，請參閱[HyperLogLog 函數](#)。

下列範例會將包含HLLSKETCH欄的表格匯出至檔案。

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
(5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

UNLOAD ('select * from hll_table') TO 's3://amzn-s3-demo-bucket/unload/'
IAM_ROLE 'arn:aws:iam::0123456789012:role/MyRedshiftRole' NULL AS 'null' ALLOWOVERWRITE
CSV;
```

卸載數VARBYTE據類型的列

您只能將VARBYTE欄卸載為文字或CSV格式。資料會以十六進位形式卸載。您無法使用該FIXEDWIDTH選項卸載VARBYTE數據。不支援UNLOADCSV的ADDQUOTES選項。資料行不能是PARTITIONED BY資料行。

FORMATAS PARQUET 條款

使用 FORMAT AS 時，請注意以下注意事項PARQUET：

- 卸載至 Parquet 不會使用檔案層級壓縮。每個資料列群組都會使用壓縮SNAPPY。
- 如果MAXFILESIZE未指定，預設檔案大小上限為 6.2 GB。您可以使MAXFILESIZE用指定 5 MB 至 6.2 GB 的檔案大小。實際檔案大小會在寫入檔案時估算，因此可能不會與您指定的數字完全相等。

為了最大化掃描效能，Amazon Redshift 會嘗試建立包含 32-MB 相等大小資料列群組的 Parquet 檔案。您指定的MAXFILESIZE值會自動四捨五入至最接近的 32 MB 倍數。例如，如果您指定

MAXFILESIZE 200 MB，則卸載的每個實木複合地板檔案大約為 192 MB (32 MB 的資料列群組 x 6 = 192 MB)。

- 如果資料行使用TIMESTAMPTZ資料格式，則只會卸載時間戳記值。不會卸載時區資訊。
- 請不要指定以底線 (_) 或句號 (.) 字元開頭的檔案名稱字首。Redshift Spectrum 會將以這些字元開頭的檔案視為隱藏檔案並進行忽略。

PARTITIONBY 子句

使用 PARTITION BY 時，請注意以下注意事項：

- 分割區資料行不會包含在輸出檔案中。
- 請務必在UNLOAD陳述式中使用的SELECT查詢中包含分割區資料行。您可以在UNLOAD命令中指定任意數量的分割區欄。但是，其中一個限制是檔案內至少應包含一個非分割區資料行。
- 如果分割區索引鍵的值為 Null，Amazon Redshift 會自動將該資料卸載至稱為 `partition_column=__HIVE_DEFAULT_PARTITION__` 的預設分割區。
- 此命UNLOAD令不會對外部目錄進行任何呼叫。要將新分區註冊為現有外部表的一部分，請使用單獨的 ALTERTABLE... ADDPARTITION... 命令。或者，您可以運行CREATEEXTERNALTABLE命令將卸載的數據註冊為新的外部表。您也可以使用 AWS Glue 爬行者程式來填入您的資料目錄。如需詳細資訊，請參閱 < [定義搜尋器](#) > AWS Glue 開發人員指南。
- 如果您使用MANIFEST此選項，Amazon Redshift 只會在根 Amazon S3 資料夾中產生一個資訊清單檔案。
- 您可以用作分區索引鍵的資料欄資料類型為 SMALLINTINTEGERBIGINTDECIMAL、REAL、BOOLEAN、CHAR、VARCHAR、DATE、和 TIMESTAMP。

使用ASSUMEROLE權限授與UNLOAD作業IAM角色的存取權

若要為特定使用者和群組提供UNLOAD作業IAM角色的存取權，超級使用者可以將IAM角色的ASSUMEROLE權限授與使用者和群組。如需相關資訊，請參閱 [GRANT](#)。

UNLOAD不支援 Amazon S3 存取點別名

您無法將 Amazon S3 存取點別名與UNLOAD命令搭配使用。

範例

若要取得展示如何使用UNLOAD指令的範例，請參閱 < [UNLOAD例子](#) >。

UNLOAD例子

這些範例會示範UNLOAD指令的各種參數。範TICKIT例資料用於許多範例中。如需詳細資訊，請參閱[範本資料庫](#)。

Note

這些範例包含換行以方便閱讀。請勿在 credentials-args 字串中包含換行或空格。

卸載VENUE到以管道分隔的文件 (默認分隔符)

下列範例會卸載資料VENUE表並將資料寫入 : s3://amzn-s3-demo-bucket/unload/

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

依預設，UNLOAD會為每個磁碟片段寫入一或多個檔案。假設有一個雙節點叢集，且每個節點有兩個分割，則上方範例會在 amzn-s3-demo-bucket 中建立這些檔案：

```
unload/0000_part_00
unload/0001_part_00
unload/0002_part_00
unload/0003_part_00
```

為了能更清楚區分輸出檔案，您可以在位置中包含字首。下列範例會卸載資料VENUE表並將資料寫入 : s3://amzn-s3-demo-bucket/unload/venue_pipe_

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

結果會是 unload 資料夾中的這四個檔案，同樣假設有四個分割。

```
venue_pipe_0000_part_00
venue_pipe_0001_part_00
venue_pipe_0002_part_00
venue_pipe_0003_part_00
```

卸載LINEITEM表到分區的實木複合地板文件

下l_shipdate列範例會以 Parquet 格式卸載資料LINEITEM表，並依資料行分割。

```
unload ('select * from lineitem')
to 's3://amzn-s3-demo-bucket/lineitem/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
PARQUET
PARTITION BY (l_shipdate);
```

假設有四個分割，則結果產生的 Parquet 檔案會以動態方式分成不同的資料夾。

```
s3://amzn-s3-demo-bucket/lineitem/l_shipdate=1992-01-02/0000_part_00.parquet
                                0001_part_00.parquet
                                0002_part_00.parquet
                                0003_part_00.parquet
s3://amzn-s3-demo-bucket/lineitem/l_shipdate=1992-01-03/0000_part_00.parquet
                                0001_part_00.parquet
                                0002_part_00.parquet
                                0003_part_00.parquet
s3://amzn-s3-demo-bucket/lineitem/l_shipdate=1992-01-04/0000_part_00.parquet
                                0001_part_00.parquet
                                0002_part_00.parquet
                                0003_part_00.parquet
...
```

Note

在某些情況下，UNLOAD命令使用的INCLUDE選項顯示在下面的SQL語句。

```
unload ('select * from lineitem')
to 's3://amzn-s3-demo-bucket/lineitem/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
PARQUET
PARTITION BY (l_shipdate) INCLUDE;
```

在這些情況下，l_shipdate 資料欄也在 Parquet 檔案的資料中。否則，l_shipdate 資料欄資料不在 Parquet 檔案中。

將VENUE表格卸載至檔案 JSON

下列範例會卸載資料VENUE表，並以JSON格式將資料寫入。s3://amzn-s3-demo-bucket/unload/

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
JSON;
```

以下是從VENUE表中的示例行。

venueid	venue name	venue city	venue state	venue seats
1	Pinewood Racetrack	Akron	OH	0
2	Columbus "Crew" Stadium	Columbus	OH	0
4	Community, Ballpark, Arena	Kansas City	KS	0

卸載到之後JSON，檔案的格式類似於以下內容。

```
{"venueid":1,"venue name":"Pinewood
Racetrack","venue city":"Akron","venue state":"OH","venue seats":0}
{"venueid":2,"venue name":"Columbus \"Crew\" Stadium
","venue city":"Columbus","venue state":"OH","venue seats":0}
{"venueid":4,"venue name":"Community, Ballpark, Arena","venue city":"Kansas
City","venue state":"KS","venue seats":0}
```

卸載VENUE到文件 CSV

下列範例會卸載資料VENUE表，並以CSV格式將資料寫入。s3://amzn-s3-demo-bucket/unload/

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
CSV;
```

假設該VENUE表包含以下行。

venueid	venueName	venueCity	venueState	venueSeats
1	Pinewood Racetrack	Akron	OH	0
2	Columbus "Crew" Stadium	Columbus	OH	0
4	Community, Ballpark, Arena	Kansas City	KS	0

卸載檔案看起來類似下列。

```
1,Pinewood Racetrack,Akron,OH,0
2,"Columbus ""Crew"" Stadium",Columbus,OH,0
4,"Community, Ballpark, Arena",Kansas City,KS,0
```

使用分隔符卸載VENUE到CSV文件

下列範例會卸載資料VENUE表，並使用直線字元 (|) 做為分隔符號以CSV格式寫入資料。卸載的檔案會寫入到 `s3://amzn-s3-demo-bucket/unload/`。此範例中的VENUE表格包含第一列 (Pinewood Race|track) 值中的直線字元。它這樣做是為了表明結果中的值用雙引號括住。雙引號會逸出雙引號，而整個欄位會用雙引號括住。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
CSV DELIMITER AS '|';
```

假設該VENUE表包含以下行。

venueid	venueName	venueCity	venueState	venueSeats
1	Pinewood Race track	Akron	OH	0
2	Columbus "Crew" Stadium	Columbus	OH	0
4	Community, Ballpark, Arena	Kansas City	KS	0

卸載檔案看起來類似下列。

```
1|"Pinewood Race|track"|Akron|OH|0
2|"Columbus ""Crew"" Stadium"|Columbus|OH|0
4|Community, Ballpark, Arena|Kansas City|KS|0
```


使用清單VENUE文件卸載

若要建立資訊清單檔案，請包含選MANIFEST項。下列範例會卸載資料VENUE表，並將資訊清單檔案與資料檔案一起寫入 s3://amzn-s3-demo-bucket/venue_pipe_ :

Important

如果您使用MANIFEST選項卸載檔案，則在載入檔案時應該搭配COPY指令使用該MANIFEST選項。如果您使用相同的前綴來加載文件，但未指定MANIFEST選項，則COPY會失敗，因為它假定清單文件是數據文件。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

結果會是這五個檔案：

```
s3://amzn-s3-demo-bucket/venue_pipe_0000_part_00
s3://amzn-s3-demo-bucket/venue_pipe_0001_part_00
s3://amzn-s3-demo-bucket/venue_pipe_0002_part_00
s3://amzn-s3-demo-bucket/venue_pipe_0003_part_00
s3://amzn-s3-demo-bucket/venue_pipe_manifest
```

以下示範資訊清單檔案的內容。

```
{
  "entries": [
    {"url": "s3://amzn-s3-demo-bucket/ticket/venue_0000_part_00"},
    {"url": "s3://amzn-s3-demo-bucket/ticket/venue_0001_part_00"},
    {"url": "s3://amzn-s3-demo-bucket/ticket/venue_0002_part_00"},
    {"url": "s3://amzn-s3-demo-bucket/ticket/venue_0003_part_00"}
  ]
}
```

卸載 VENUE MANIFEST VERBOSE

當您指定MANIFESTVERBOSE選項時，資訊清單檔案包含下列區段：

- `entries` 區段會列出每個檔案的 Amazon S3 路徑、檔案大小和資料列計數。
- `schema` 區段會列出每個資料欄的資料欄名稱、資料類型和維度。
- `meta` 區段會顯示所有檔案的檔案大小總計和資料列計數。

下列範例會使用 `MANIFESTVERBOSE` 選項卸載資料表。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload_venue_folder/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest verbose;
```

以下示範資訊清單檔案的內容。

```
{
  "entries": [
    {"url": "s3://amzn-s3-demo-bucket/venue_pipe_0000_part_00", "meta":
    { "content_length": 32295, "record_count": 10 }},
    {"url": "s3://amzn-s3-demo-bucket/venue_pipe_0001_part_00", "meta":
    { "content_length": 32771, "record_count": 20 }},
    {"url": "s3://amzn-s3-demo-bucket/venue_pipe_0002_part_00", "meta":
    { "content_length": 32302, "record_count": 10 }},
    {"url": "s3://amzn-s3-demo-bucket/venue_pipe_0003_part_00", "meta":
    { "content_length": 31810, "record_count": 15 }}
  ],
  "schema": {
    "elements": [
      {"name": "venueid", "type": { "base": "integer" }},
      {"name": "venueName", "type": { "base": "character varying", 25 }},
      {"name": "venueCity", "type": { "base": "character varying", 25 }},
      {"name": "venueState", "type": { "base": "character varying", 25 }},
      {"name": "venueSeats", "type": { "base": "character varying", 25 }}
    ]
  },
  "meta": {
    "content_length": 129178,
    "record_count": 55
  },
  "author": {
    "name": "Amazon Redshift",
    "version": "1.0.0"
  }
}
```

```
}
```

VENUE使用標題卸載

下列範例會卸載VENUE標題列。

```
unload ('select * from venue where venueseats > 75000')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
header
parallel off;
```

以下示範資訊具有標題列的輸出檔案的內容。

```
venueid|venueName|venueCity|venueState|venueseats
6|New York Giants Stadium|East Rutherford|NJ|80242
78|INVESCO Field|Denver|CO|76125
83|FedExField|Landover|MD|91704
79|Arrowhead Stadium|Kansas City|MO|79451
```

卸載VENUE到較小的文件

根據預設，檔案大小的上限為 6.2 GB。如果卸載資料大於 6.2 GB，則會為每個 6.2 GB 資料區段 UNLOAD 建立一個新檔案。若要建立較小的檔案，請包括 MAXFILESIZE 參數。假設上一個範例中的資料大小為 20 GB，下列 UNLOAD 命令會建立 20 個檔案，每個檔案大小為 1 GB。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
maxfilesize 1 gb;
```

順序卸載 VENUE

若要依序卸載，請指定。PARALLEL OFF UNLOAD 然後一次寫入一個文件，每個文件最多 6.2 GB。

下列範例會卸載資料 VENUE 表，並依序將資料寫入。s3://amzn-s3-demo-bucket/unload/

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/venue_serial_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off;
```

結果會產生一個名為 `venue_serial_000` 的檔案。

如果卸載資料大於 6.2 GB，則會為每個 6.2 GB 資料區段 UNLOAD 建立一個新檔案。下列範例會卸載資料 LINEORDER 表，並依序將資料寫入。 `s3://amzn-s3-demo-bucket/unload/`

```
unload ('select * from lineorder')
to 's3://amzn-s3-demo-bucket/unload/lineorder_serial_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
parallel off gzip;
```

結果會是下方的一系列檔案。

```
lineorder_serial_0000.gz
lineorder_serial_0001.gz
lineorder_serial_0002.gz
lineorder_serial_0003.gz
```

為了能更清楚區分輸出檔案，您可以在位置中包含字首。下列範例會卸載資料 VENUE 表並將資料寫入：`s3://amzn-s3-demo-bucket/venue_pipe_`

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/unload/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

結果會是 unload 資料夾中的這四個檔案，同樣假設有四個分割。

```
venue_pipe_0000_part_00
venue_pipe_0001_part_00
venue_pipe_0002_part_00
venue_pipe_0003_part_00
```

VENUE 從卸載文件加載

要從一組卸載文件中加載表，只需使用 COPY 命令反轉過程即可。下列範例會建立新資料表 LOADVENUE，並從上一個範例中建立的資料檔載入表格。

```
create table loadvenue (like venue);

copy loadvenue from 's3://amzn-s3-demo-bucket/venue_pipe_' iam_role
'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

如果您使用MANIFEST此選項建立包含卸載檔案的資訊清單檔案，您可以使用相同的資訊清單檔案載入資料。您可以使用帶有該MANIFEST選項的COPY命令來執行此操作。下列範例會使用資訊清單檔案載入資料。

```
copy loadvenue
from 's3://amzn-s3-demo-bucket/venue_pipe_manifest' iam_role
  'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest;
```

卸載VENUE到加密文件

下列範例會使用 VENUE AWS KMS 索引鍵。如果您使用ENCRYPTED選項指定資訊清單檔案，資訊清單檔案也會加密。如需詳細資訊，請參閱[卸載加密的資料檔案](#)。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_encrypt_kms'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
kms_key_id '1234abcd-12ab-34cd-56ef-1234567890ab'
manifest
encrypted;
```

下列範例會使用根對稱金鑰將VENUE資料表卸載至一組加密檔案。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_encrypt_cmk'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key 'EXAMPLEMASTERKEYtkbjk/OpCwtYSx/M4/t7DMCDIK722'
encrypted;
```

VENUE從加密檔案載入

若要從UNLOAD搭配ENCRYPT選項使用建立的一組檔案載入表格，請使用COPY指令反轉程序。使用該命令，使用ENCRYPTED選項並指定用於該UNLOAD命令的相同根對稱金鑰。下列範例會從前一個範例中建立的加密資料檔案載入資料LOADVENUE表。

```
create table loadvenue (like venue);

copy loadvenue
from 's3://amzn-s3-demo-bucket/venue_encrypt_manifest'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
master_symmetric_key 'EXAMPLEMASTERKEYtkbjk/OpCwtYSx/M4/t7DMCDIK722'
```

```
manifest
encrypted;
```

將VENUE資料卸載到以 Tab 字元分隔的檔案

```
unload ('select venueid, venuename, venueseats from venue')
to 's3://amzn-s3-demo-bucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t';
```

輸出資料檔案看起來像這樣：

```
1 Toyota Park Bridgeview IL 0
2 Columbus Crew Stadium Columbus OH 0
3 RFK Stadium Washington DC 0
4 CommunityAmerica Ballpark Kansas City KS 0
5 Gillette Stadium Foxborough MA 68756
...
```

卸載VENUE到固定寬度的數據文件

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_fw_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
fixedwidth as 'venueid:3,venuename:39,venuecity:16,venuestate:2,venueseats:6';
```

輸出資料檔案看起來會像下面這樣。

```
1 Toyota Park           Bridgeview  IL0
2 Columbus Crew Stadium Columbus   OH0
3 RFK Stadium           Washington  DC0
4 CommunityAmerica BallparkKansas City  KS0
5 Gillette Stadium      Foxborough MA68756
...
```

卸載VENUE到一組以 Tab 字元分GZIP隔的壓縮檔

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter as '\t'
```

```
gzip;
```

卸載VENUE到GZIP壓縮的文本文件

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_tab_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
extension 'txt.gz'
gzip;
```

卸載包含分隔符號的資料

此範例使用選ADDQUOTES項來卸載逗號分隔的資料，其中一些實際資料欄位包含逗號。

首先，請建立包含引號的資料表。

```
create table location (id int, location char(64));

insert into location values (1,'Phoenix, AZ'),(2,'San Diego, CA'),(3,'Chicago, IL');
```

然後，使用該ADDQUOTES選項卸載數據。

```
unload ('select id, location from location')
to 's3://amzn-s3-demo-bucket/location_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
delimiter ',' addquotes;
```

卸載的資料檔案看起來像這樣：

```
1,"Phoenix, AZ"
2,"San Diego, CA"
3,"Chicago, IL"
...
```

卸載聯結查詢的結果

以下範例會卸載包含視窗函數之聯結查詢的結果。

```
unload ('select venuename, venuestate, caldate, pricepaid,
sum(pricepaid) over(partition by venuename, venuestate
order by caldate rows between 3 preceding and 3 following) as winsum
from sales join date on sales.dateid=date.dateid
```

```

join event on event.eventid=sales.eventid
join venue on event.venueid=venue.venueid
order by 1,2')
to 's3://amzn-s3-demo-bucket/ticket/winsum'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';

```

輸出檔案看起來像這樣：

```

Atlanta|GA|2008-01-04|363.00|1362.00
Atlanta|GA|2008-01-05|233.00|2030.00
Atlanta|GA|2008-01-06|310.00|3135.00
Atlanta|GA|2008-01-08|166.00|8338.00
Atlanta|GA|2008-01-11|268.00|7630.00
...

```

使用 NULL AS 卸載

UNLOAD 默認情況下輸出 null 值作為空字符串。下列範例說明如何使用 NULL AS 來取代空值的文字字符串。

對於這些示例，我們將一些 null 值添加到 VENUE 表中。

```

update venue set venuestate = NULL
where venuecity = 'Cleveland';

```

從 VENUE 其中選取 VENUESTATE 為 null，以驗證資料行是否包含 NULL。

```

select * from venue where venuestate is null;

```

venueid	venue name	venuecity	venuestate	venueseats
22	Quicken Loans Arena	Cleveland		0
101	Progressive Field	Cleveland		43345
72	Cleveland Browns Stadium	Cleveland		73200

現 UNLOAD 在，使 VENUE 用 NULL AS 選項用字符串 'fred' 替換空值的表。

```

unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
null as 'fred';

```


下列來自卸載檔案的範例顯示，null 值已取代為 fred。事實證明，的某些值也為 null，並被替換為 fred。VENUSEATS 即使的資料類型 VENUSEATS 為整數，請將值 UNLOAD 轉換為卸載檔案中的文字，然後 COPY 將它們轉換回整數。如果您要卸載到固定寬度的檔案，NULLAS 字串不得大於欄位寬度。

```
248|Charles Playhouse|Boston|MA|0
251|Paris Hotel|Las Vegas|NV|fred
258|Tropicana Hotel|Las Vegas|NV|fred
300|Kennedy Center Opera House|Washington|DC|0
306|Lyric Opera House|Baltimore|MD|0
308|Metropolitan Opera|New York City|NY|0
  5|Gillette Stadium|Foxborough|MA|5
  22|Quicken Loans Arena|Cleveland|fred|0
101|Progressive Field|Cleveland|fred|43345
...
```

若要從卸載檔案載入表格，請使用具有相同 NULL AS 選項的 COPY 指令。

Note

如果您嘗試將空值載入定義為的資料行 NOTNULL，則 COPY 命令會失敗。

```
create table loadvenuenuLLs (like venue);

copy loadvenuenuLLs from 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
null as 'fred';
```

要驗證列包含 null，而不僅僅是空字符串，請從中選擇 LOADVENUENUALLS 並過濾 null。

```
select * from loadvenuenuLLs where venuestate is null or venueseats is null;
```

venueid	venueName	venueCity	venueState	venueSeats
72	Cleveland Browns Stadium	Cleveland		73200
253	Mirage Hotel	Las Vegas	NV	
255	Venetian Hotel	Las Vegas	NV	
22	Quicken Loans Arena	Cleveland		0
101	Progressive Field	Cleveland		43345
251	Paris Hotel	Las Vegas	NV	

...

您可以UNLOAD使用預設 AS 行為包含空值的資料表，然後COPY使用預設 NULL AS 行為將資料回到資料表中；不過，目標資料表中的任何非數值欄位都包含空字串，而非空值。NULL默認情況下，空值UNLOAD轉換為空字符串（空格或零長度）。COPY將空字串轉換NULL為數值欄，但將空字串插入非數值欄中。下列範例會示範如何執行UNLOAD後續COPY使用預設 NULL AS 行為。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' allowoverwrite;

truncate loadvenuenuLLs;
copy loadvenuenuLLs from 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole';
```

在這種情況下，當您篩選空值時，只有VENUESEATS包含空值的列。凡在表中包VENUESTATE含空值 (VENUE)，VENUESTATE在目標表 (LOADVENUENULLS) 中包含空字符串。

```
select * from loadvenuenuLLs where venuestate is null or venuesats is null;
```

venueid	venueName	venueCity	venueState	venuesats
253	Mirage Hotel	Las Vegas	NV	
255	Venetian Hotel	Las Vegas	NV	
251	Paris Hotel	Las Vegas	NV	
...				

若要將空字串載入至非數值欄NULL，請包括EMPTYASNULL或選BLANKSASNULL項。也可以兩者都使用。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' allowoverwrite;

truncate loadvenuenuLLs;
copy loadvenuenuLLs from 's3://amzn-s3-demo-bucket/nulls/'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole' EMPTYASNULL;
```

要驗證列是否包含NULL，而不僅僅是空白或空字符串，請從中選擇LOADVENUENULLS並過濾 null。

```
select * from loadvenuenuLLs where venuestate is null or venueseats is null;
```

venueid	venueName	venuecity	venuestate	venueseats
72	Cleveland Browns Stadium	Cleveland		73200
253	Mirage Hotel	Las Vegas	NV	
255	Venetian Hotel	Las Vegas	NV	
22	Quicken Loans Arena	Cleveland		0
101	Progressive Field	Cleveland		43345
251	Paris Hotel	Las Vegas	NV	
...				

使ALLOWOVERWRITE用參數卸載

依預設，UNLOAD不會覆寫目標值區中的現有檔案。例如，如果您執行相同的UNLOAD陳述式兩次，而不修改目的地值區中的檔案，則第二個陳述式UNLOAD會失敗。若要覆寫現有檔案 (包括資訊清單檔案)，請指定ALLOWOVERWRITE選項。

```
unload ('select * from venue')
to 's3://amzn-s3-demo-bucket/venue_pipe_'
iam_role 'arn:aws:iam::0123456789012:role/MyRedshiftRole'
manifest allowoverwrite;
```

使用PARALLEL和MANIFEST參數卸載EVENT表

您可以 parallel UNLOAD 建立資料表並產生資訊清單檔案。Amazon S3 資料檔案都是在相同層級建立的，名稱字尾為模式 0000_part_00。清單檔案與資料檔案位於相同的資料夾層級，並加上字尾 manifest。以下內容將SQL卸載EVENT表並創建具有基本名稱的文件 parallel

```
unload ('select * from myticket1.event')
to 's3://amzn-s3-demo-bucket/parallel'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
parallel on
manifest;
```

Amazon S3 檔案清單類似以下內容。

Name	Last modified	Size
parallel0000_part_00 -	August 2, 2023, 14:54:39 (UTC-07:00)	52.1 KB
parallel0001_part_00 -	August 2, 2023, 14:54:39 (UTC-07:00)	53.4 KB

```
parallel0002_part_00 - August 2, 2023, 14:54:39 (UTC-07:00) 52.1 KB
parallel0003_part_00 - August 2, 2023, 14:54:39 (UTC-07:00) 51.1 KB
parallel0004_part_00 - August 2, 2023, 14:54:39 (UTC-07:00) 54.6 KB
parallel0005_part_00 - August 2, 2023, 14:54:39 (UTC-07:00) 53.4 KB
parallel0006_part_00 - August 2, 2023, 14:54:39 (UTC-07:00) 54.1 KB
parallel0007_part_00 - August 2, 2023, 14:54:39 (UTC-07:00) 55.9 KB
parallelmanifest      - August 2, 2023, 14:54:39 (UTC-07:00) 886.0 B
```

parallelmanifest 檔案內容類似以下內容。

```
{
  "entries": [
    {"url": "s3://amzn-s3-demo-bucket/parallel0000_part_00", "meta": { "content_length":
53316 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0001_part_00", "meta": { "content_length":
54704 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0002_part_00", "meta": { "content_length":
53326 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0003_part_00", "meta": { "content_length":
52356 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0004_part_00", "meta": { "content_length":
55933 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0005_part_00", "meta": { "content_length":
54648 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0006_part_00", "meta": { "content_length":
55436 }},
    {"url": "s3://amzn-s3-demo-bucket/parallel0007_part_00", "meta": { "content_length":
57272 }}
  ]
}
```

使用PARALLELOFF和MANIFEST參數卸載EVENT表

您可以連續UNLOAD一個表 (PARALLELOFF) 並生成清單文件。Amazon S3 資料檔案都是在相同層級建立的，名稱字尾為模式 0000。清單檔案與資料檔案位於相同的資料夾層級，並加上字尾 manifest。

```
unload ('select * from myticket1.event')
to 's3://amzn-s3-demo-bucket/serial'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
```

```
parallel off
manifest;
```

Amazon S3 檔案清單類似以下內容。

Name	Last modified	Size
serial0000	- August 2, 2023, 15:54:39 (UTC-07:00)	426.7 KB
serialmanifest	- August 2, 2023, 15:54:39 (UTC-07:00)	120.0 B

serialmanifest 檔案內容類似以下內容。

```
{
  "entries": [
    {"url": "s3://amzn-s3-demo-bucket/serial000", "meta": { "content_length": 436991 }}
  ]
}
```

使用 PARTITION BY 和 MANIFEST 參數卸載 EVENT 表

您可以 UNLOAD 按分區劃分表並生成清單文件。Amazon S3 中會建立一個新資料夾，其中包含子分割區資料夾，而子資料夾中的資料檔案名稱模式類似於 0000_par_00。清單檔案與名為 manifest 的子資料夾位於相同的資料夾層級。

```
unload ('select * from myticket1.event')
to 's3://amzn-s3-demo-bucket/partition'
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
partition by (eventname)
manifest;
```

Amazon S3 檔案清單類似以下內容。

Name	Type	Last modified	Size
partition	Folder		

在 partition 資料夾中的是具有分割區名稱和清單檔案的子文件夾。下面顯示的是資料夾 partition 中資料夾清單的底端，類似以下內容。

Name	Type	Last modified	Size
...			
eventname=Zucchero/	Folder		
eventname=Zumanity/	Folder		
eventname=ZZ Top/	Folder		
manifest	-	August 2, 2023, 15:54:39 (UTC-07:00)	467.6 KB

eventname=Zucchero/ 資料夾中的資料檔案類似於以下內容。

Name	Last modified	Size
0000_part_00 -	August 2, 2023, 15:59:19 (UTC-07:00)	70.0 B
0001_part_00 -	August 2, 2023, 15:59:16 (UTC-07:00)	106.0 B
0002_part_00 -	August 2, 2023, 15:59:15 (UTC-07:00)	70.0 B
0004_part_00 -	August 2, 2023, 15:59:17 (UTC-07:00)	141.0 B
0006_part_00 -	August 2, 2023, 15:59:16 (UTC-07:00)	35.0 B
0007_part_00 -	August 2, 2023, 15:59:19 (UTC-07:00)	108.0 B

manifest 檔案內容底部類似以下內容。

```
{
  "entries": [
    ...
    {"url":"s3://amzn-s3-demo-bucket/partition/eventname=Zucchero/007_part_00", "meta":
  { "content_length": 108 }},
    {"url":"s3://amzn-s3-demo-bucket/partition/eventname=Zumanity/007_part_00", "meta":
  { "content_length": 72 }}
  ]
}
```

使用MAXFILESIZE、ROWGROUPSIZE和MANIFEST參數卸載EVENT表格

您可以 parallel UNLOAD 建立資料表並產生資訊清單檔案。Amazon S3 資料檔案都是在相同層級建立的，名稱字尾為模式 0000_part_00。產生的 Parquet 資料檔案會限制為 256 MB，資料列群組大小為 128 MB。清單檔案與資料檔案位於相同的資料夾層級，並加上字尾 manifest。

```
unload ('select * from myticket1.event')
to 's3://amzn-s3-demo-bucket/eventsize'
```

```
iam_role 'arn:aws:iam::123456789012:role/MyRedshiftRole'
maxfilesize 256 MB
rowgroupsize 128 MB
parallel on
parquet
manifest;
```

Amazon S3 檔案清單類似以下內容。

Name	Type	Last modified	Size
eventsize0000_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	24.5 KB
eventsize0001_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	24.8 KB
eventsize0002_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	24.4 KB
eventsize0003_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	24.0 KB
eventsize0004_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	25.3 KB
eventsize0005_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	24.8 KB
eventsize0006_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	25.0 KB
eventsize0007_part_00.parquet	parquet	August 2, 2023, 17:35:21 (UTC-07:00)	25.6 KB
eventsizemanifest	-	August 2, 2023, 17:35:21 (UTC-07:00)	958.0 B

eventsizemanifest 檔案內容類似以下內容。

```
{
  "entries": [
    {"url": "s3://amzn-s3-demo-bucket/eventsize0000_part_00.parquet", "meta":
    { "content_length": 25130 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0001_part_00.parquet", "meta":
    { "content_length": 25428 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0002_part_00.parquet", "meta":
    { "content_length": 25025 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0003_part_00.parquet", "meta":
    { "content_length": 24554 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0004_part_00.parquet", "meta":
    { "content_length": 25918 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0005_part_00.parquet", "meta":
    { "content_length": 25362 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0006_part_00.parquet", "meta":
    { "content_length": 25647 }},
    {"url": "s3://amzn-s3-demo-bucket/eventsize0007_part_00.parquet", "meta":
    { "content_length": 26256 }}
  ]
}
```

```
]
}
```

UPDATE

主題

- [語法](#)
- [參數](#)
- [使用須知](#)
- [UPDATE 陳述式範例](#)

在條件滿足時，更新一個或多個資料表資料欄中的值。

Note

單一SQL陳述式的大小上限為 16 MB。

語法

```
[ WITH [RECURSIVE] common_table_expression [, common_table_expression , ...] ]
    UPDATE table_name [ [ AS ] alias ] SET column = { expression | DEFAULT }
    [, ...]

[ FROM fromlist ]
[ WHERE condition ]
```

參數

WITH 子句

指定一或多個的選用子句common-table-expressions。請參閱 [WITH 子句](#)。

table_name

暫時性或持久性資料表。只有資料表的擁有者或具有UPDATE權限的使用者可以更新資料列。如果您使用 FROM子句或從表達式或條件中的資料表中選取，您必須在這些資料表上具有 SELECT 權限。您無法在此處為資料表提供別名；不過，您可以在 FROM子句中指定別名。

Note

Amazon Redshift Spectrum 外部資料表處於唯讀狀態。您無法UPDATE外部資料表。

別名

目標資料表的暫時替代名稱。別名是選擇性項目。AS 關鍵字一律為選用。

SET 資料欄 =

您要修改的一個或多個資料欄。未列出的資料欄會保留其目前值。不要在目標資料欄的規格中包含資料表名稱。例如，UPDATE tab SET tab.col = 1 無效。

運算式

此表達式會為所指定資料欄定義新值。

DEFAULT

使用指派給CREATETABLE陳述式中資料欄的預設值更新資料欄。

FROM 資料表清單

您可以透過參考其他資料表中的資訊更新資料表。在 FROM 子句中列出這些其他資料表，或使用子查詢作為WHERE條件的一部分。FROM 子句中列出的資料表可以具有別名。如果您需要在清單中包含UPDATE陳述式的目標資料表，請使用別名。

WHERE 條件

此選用子句會限制僅更新符合條件的資料列。當條件傳回時true，會更新指定的SET資料欄。條件可以是資料欄上簡單的述詞，或是根據子查詢結果的條件。

您可以為子查詢中的任何資料表命名，包括的目標資料表UPDATE。

使用須知

在更新資料表中的大量資料列之後：

- 清空資料表以回收儲存空間和重新排序資料列。
- 分析資料表以更新查詢規劃器的統計資訊。

UPDATE 陳述式的 FROM子句不支援左、右和完整外部聯結；它們傳回下列錯誤：

```
ERROR: Target table must be part of an equijoin predicate
```

如果您需要指定外部聯結，請在 UPDATE 陳述式的 WHERE 子句中使用子查詢。

如果您的 UPDATE 陳述式需要自我加入目標資料表，您需要指定聯結條件，以及符合更新操作資料列資格的 WHERE 子句條件。大致而言，當目標資料表與其本身或其他資料表聯結時，最佳實務會是使用子查詢清楚區分聯結條件和限定更新資料列的條件。

UPDATE 當組態參數 `error_on_nondeterministic_update` 設定為 `true` 時，每列具有多個相符項目的查詢會擲出錯誤。如需詳細資訊，請參閱 [error_on_nondeterministic_update](#)。

您可以更新 GENERATED BY DEFAULT AS IDENTITY 資料欄。定義為 GENERATED BY DEFAULT AS 的資料欄 IDENTITY 可以使用您提供的值進行更新。如需詳細資訊，請參閱 [GENERATED BY DEFAULT AS IDENTITY](#)。

UPDATE 陳述式範例

如需下列範例中所用資料表的相關資訊，請參閱 [範本資料庫](#)。

TICKIT 資料庫中的 CATEGORY 資料表包含下列資料列：

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
| 5      | Sports   | MLS      | Major League Soccer      |
| 11     | Concerts | Classical | All symphony, concerto, and choir concerts |
| 1      | Sports   | MLB      | Major League Baseball    |
| 6      | Shows    | Musicals  | Musical theatre          |
| 3      | Sports   | NFL      | National Football League  |
| 8      | Shows    | Opera    | All opera and light opera |
| 2      | Sports   | NHL      | National Hockey League    |
| 9      | Concerts | Pop      | All rock and pop music concerts |
| 4      | Sports   | NBA      | National Basketball Association |
| 7      | Shows    | Plays    | All non-musical theatre   |
| 10     | Concerts | Jazz     | All jazz singers and bands |
+-----+-----+-----+-----+
```

根據值的範圍更新資料表

根據資料 CATGROUP 欄內的值範圍更新資料 CATID 欄。

```
UPDATE category
```

```
SET catgroup='Theatre'
WHERE catid BETWEEN 6 AND 8;

SELECT * FROM category
WHERE catid BETWEEN 6 AND 8;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
| 6     | Theatre | Musicals | Musical theatre          |
| 7     | Theatre | Plays   | All non-musical theatre  |
| 8     | Theatre | Opera   | All opera and light opera |
+-----+-----+-----+-----+
```

根據目前值更新資料表

根據目前的CATGROUP值更新 CATNAME和 CATDESC資料欄：

```
UPDATE category
SET catdesc=default, catname='Shows'
WHERE catgroup='Theatre';

SELECT * FROM category
WHERE catname='Shows';
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname | catdesc |
+-----+-----+-----+-----+
| 6     | Theatre | Shows   | NULL    |
| 7     | Theatre | Shows   | NULL    |
| 8     | Theatre | Shows   | NULL    |
+-----+-----+-----+-----+)
```

在此情況下，由於建立資料表時未定義預設值，因此資料CATDESC欄設定為 null。

執行下列命令，將CATEGORY資料表資料設回原始值：

```
TRUNCATE category;

COPY category
FROM 's3://redshift-downloads/ticket/category_pipe.txt'
DELIMITER '|'
IGNOREHEADER 1
```

```
REGION 'us-east-1'
IAM_ROLE default;
```

根據WHERE子句子查詢的結果更新資料表

根據子查詢在子WHERE句中的結果更新CATEGORY資料表：

```
UPDATE category
SET catdesc='Broadway Musical'
WHERE category.catid IN
(SELECT category.catid FROM category
JOIN event ON category.catid = event.catid
JOIN venue ON venue.venueid = event.venueid
JOIN sales ON sales.eventid = event.eventid
WHERE venuecity='New York City' AND catname='Musicals');
```

檢視更新的資料表：

```
SELECT * FROM category ORDER BY catid;
```

catid	catgroup	catname	catdesc
2	Sports	NHL	National Hockey League
3	Sports	NFL	National Football League
4	Sports	NBA	National Basketball Association
5	Sports	MLS	Major League Soccer
6	Shows	Musicals	Broadway Musical
7	Shows	Plays	All non-musical theatre
8	Shows	Opera	All opera and light opera
9	Concerts	Pop	All rock and pop music concerts
10	Concerts	Jazz	All jazz singers and bands
11	Concerts	Classical	All symphony, concerto, and choir concerts

根據WITH子句子查詢的結果更新資料表

若要根據使用 WITH子句的子查詢結果更新CATEGORY資料表，請使用下列範例。

```
WITH u1 as (SELECT catid FROM event ORDER BY catid DESC LIMIT 1)
UPDATE category SET catid='200' FROM u1 WHERE u1.catid=category.catid;

SELECT * FROM category ORDER BY catid DESC LIMIT 1;
```

```

+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
| 200   | Concerts | Pop     | All rock and pop music concerts |
+-----+-----+-----+-----+

```

根據聯結條件的結果更新資料表

根據CATEGORY資料表中的相符資料列，更新EVENT資料表中的原始 11 資料CATID列：

```

UPDATE category SET catid=100
FROM event
WHERE event.catid=category.catid;

SELECT * FROM category ORDER BY catid;

```

```

+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
| 2     | Sports   | NHL     | National Hockey League   |
| 3     | Sports   | NFL     | National Football League |
| 4     | Sports   | NBA     | National Basketball Association |
| 5     | Sports   | MLS     | Major League Soccer      |
| 10    | Concerts | Jazz    | All jazz singers and bands |
| 11    | Concerts | Classical | All symphony, concerto, and choir concerts |
| 100   | Concerts | Pop     | All rock and pop music concerts |
| 100   | Shows    | Plays   | All non-musical theatre   |
| 100   | Shows    | Opera   | All opera and light opera |
| 100   | Shows    | Musicals | Broadway Musical         |
+-----+-----+-----+-----+

```

請注意，EVENT資料表會列在FROM子句中，而目標資料表的聯結條件會在WHERE子句中定義。只有四個資料列符合更新資格。這四個資料列是其CATID值原本為 6、7、8 和 9 的資料列；只有這四個類別會顯示在EVENT表格中：

```

SELECT DISTINCT catid FROM event;

```

```

+-----+
| catid |
+-----+
| 6     |
| 7     |

```

```
| 8      |
| 9      |
+-----+
```

透過擴展先前的範例並將另一個條件新增至WHERE子句，來更新CATEGORY資料表中的原始 11 列。由於資料CATGROUP欄的限制，只有一列符合更新的資格（但四列符合聯結的資格）。

```
UPDATE category SET catid=100
FROM event
WHERE event.catid=category.catid
AND catgroup='Concerts';

SELECT * FROM category WHERE catid=100;
```

```
+-----+-----+-----+-----+
| catid | catgroup | catname |          catdesc          |
+-----+-----+-----+-----+
| 100   | Concerts | Pop     | All rock and pop music concerts |
+-----+-----+-----+-----+
```

撰寫此範例的替代方法如下：

```
UPDATE category SET catid=100
FROM event JOIN category cat ON event.catid=cat.catid
WHERE cat.catgroup='Concerts';
```

此方法的好處在於，聯結條件會與任何其他限定更新資料列的條件清楚區分。請注意子FROM句中CATCATEGORY資料表的別名。

子FROM句中的外部聯結更新

上一個範例顯示UPDATE陳述式子FROM句中指定的內部聯結。下列範例傳回錯誤，因為子FROM句不支援目標資料表的外部聯結：

```
UPDATE category SET catid=100
FROM event LEFT JOIN category cat ON event.catid=cat.catid
WHERE cat.catgroup='Concerts';
ERROR: Target table must be part of an equijoin predicate
```

如果UPDATE陳述式需要外部聯結，您可以將外部聯結語法移至子查詢：

```
UPDATE category SET catid=100
```

```

FROM
(SELECT event.catid FROM event LEFT JOIN category cat ON event.catid=cat.catid)
 eventcat
WHERE category.catid=eventcat.catid
AND catgroup='Concerts';

```

更新子SET句中另一個資料表的資料欄

若要更新 listing TICKIT 範例資料庫中具有 值的 資料表 sales 資料表，請使用下列範例。

```

SELECT listid, numtickets FROM listing WHERE sellerid = 1 ORDER BY 1 ASC LIMIT 5;

```

```

+-----+-----+
| listid | numtickets |
+-----+-----+
| 100423 | 4          |
| 108334 | 24         |
| 117150 | 4          |
| 135915 | 20         |
| 205927 | 6          |
+-----+-----+

```

```

UPDATE listing
SET numtickets = sales.sellerid
FROM sales
WHERE sales.sellerid = 1 AND listing.sellerid = sales.sellerid;

```

```

SELECT listid, numtickets FROM listing WHERE sellerid = 1 ORDER BY 1 ASC LIMIT 5;

```

```

+-----+-----+
| listid | numtickets |
+-----+-----+
| 100423 | 1          |
| 108334 | 1          |
| 117150 | 1          |
| 135915 | 1          |
| 205927 | 1          |
+-----+-----+

```

VACUUM

在指定的資料表中或目前資料庫的所有資料表中，重新排序資料列並回收空間。

Note

只有具有必要資料表權限的使用者才能有效地清除資料表。如果 VACUUM 在沒有必要的資料表許可的情況下執行，則操作已成功完成，但沒有效果。如需有效執行的有效資料表許可清單 VACUUM，請參閱下列必要權限章節。

Amazon Redshift 會自動排序資料並在背景 VACUUMDELETE 執行。這可減少執行 VACUUM 命令的需要。如需詳細資訊，請參閱[清空資料表](#)。

根據預設，會 VACUUM 略過任何資料表的排序階段，其中已排序資料表列的 95% 以上。跳過排序階段可以大幅改善 VACUUM 效能。若要變更單一資料表的預設排序或刪除閾值，請在執行時包含資料表名稱和 TO 閾值 PERCENT 參數 VACUUM。

使用者可以在清空資料表時加以存取。您可以在將資料表清空時執行查詢和寫入操作，但當資料操作語言（DML）命令和真空同時執行時，兩者可能需要更長的時間。如果您在真空期間執行 UPDATE 和 DELETE 陳述式，系統效能可能會降低。VACUUM DELETE 暫時封鎖更新和刪除操作。

Amazon Redshift 會自動在背景執行 DELETEONLY 吸塵。當使用者執行資料定義語言（DDL）操作時，自動清空操作會暫停，例如 ALTER TABLE。

Note

Amazon Redshift VACUUM 命令語法和行為與 PostgreSQL VACUUM 操作大不相同。例如，Amazon Redshift 中的預設 VACUUM 操作為 VACUUM FULL，這會回收磁碟空間並重新排序所有資料列。相反地，PostgreSQL 中的預設 VACUUM 操作只會回收空間，並使其可供重複使用。

如需詳細資訊，請參閱[清空資料表](#)。

所需權限

以下是的必要權限 VACUUM：

- 超級使用者
- 具有 VACUUM 權限的使用者
- 資料表擁有者

- 共用資料表的資料庫擁有者

語法

```
VACUUM [ FULL | SORT ONLY | DELETE ONLY | REINDEX | RECLUSTER ]  
[ [ table_name ] [ TO threshold PERCENT ] [ BOOST ] ]
```

參數

FULL

排序指定的資料表（或目前資料庫中的所有資料表），並擷取先前 UPDATE 和 DELETE 操作標記為刪除的資料列所佔用的磁碟空間。VACUUM FULL 為預設值。

完整清空並不會對交錯資料表執行重新建立索引。若要在完整清空之後重新建立交錯資料表的索引，請使用 [VACUUM REINDEX](#) 選項。

根據預設，會 VACUUM FULL 略過已排序至少 95% 的任何資料表的排序階段。如果 VACUUM 能夠略過排序階段，它會在刪除階段執行 DELETE ONLY 並回收空間，這樣就不會將至少 95% 的剩餘資料列標記為刪除。

如果未達到排序閾值（例如，如果 90% 的資料列已排序）並 VACUUM 執行完整排序，則它也會執行完整的刪除操作，從 100% 的已刪除資料列中復原空間。

您只能變更單一資料表的預設清空閾值。若要變更單一資料表的預設真空閾值，請包含資料表名稱和 TO 閾值 PERCENT 參數。

SORT ONLY

排序指定的資料表（或目前資料庫中的所有資料表），但不回收已刪除資料列釋出的空間。當回收磁碟空間不重要，但重新排序新資料列很重要時，此選項很實用。當未排序的區域不包含大量已刪除的資料列，且未跨越整個排序區域時，SORT ONLY 真空會減少真空操作的經過時間。若應用程式沒有磁碟空間限制條件，但須倚賴與保持資料表資料列排序狀態相關聯的查詢最佳化，則可受益於這類清空。

根據預設，會 VACUUM SORT ONLY 略過已排序至少 95% 的任何資料表。若要變更單一資料表的預設排序閾值，請在執行時包含資料表名稱和 TO 閾值 PERCENT 參數 VACUUM。

DELETE ONLY

Amazon Redshift 會自動在背景執行 DELETE ONLY 吸塵，因此您很少需要執行 DELETE ONLY 吸塵。

會VACUUMDELETE回收被先前 UPDATE和 DELETE操作標記為刪除的資料列佔用的磁碟空間，並壓縮資料表以釋放耗用空間。DELETE ONLY 真空操作不會排序資料表資料。

當回收磁碟空間很重要，但重新排序新資料列不重要時，此選項可縮短清空操作的經過時間。當您的查詢效能已達最佳狀態，而不需要重新排序資料列來最佳化查詢效能時，此選項也很實用。

依預設，VACUUM DELETE ONLY 會回收空間，讓至少 95% 的剩餘資料列不會標記為刪除。若要變更單一資料表的預設刪除閾值，請在執行時包含資料表名稱和 TOthresholdPERCENT 參數 VACUUM。

某些操作 (如 ALTER TABLE APPEND) 可能造成資料表遭切割成片段。當您使用 DELETE ONLY 子句時，清空操作會回收片段資料表的空間。重組操作同樣會套用 95% 的閾值。

REINDEX tablename

分析交錯排序索引鍵欄中的值分佈，然後執行完整VACUUM操作。REINDEX 如果使用，則需要資料表名稱。

VACUUM REINDEX 花費的時間遠超過，VACUUMFULL因為它會進行額外的傳遞來分析交錯的排序金鑰。交錯資料表的排序和合併操作可能需要更長的時間，因為交錯排序可能需要重新排列比複合排序更多的資料列。

如果VACUUMREINDEX操作在完成之前終止，則下一個會在執行完整真空操作之前VACUUM繼續重新索引操作。

VACUUM REINDEX 不支援 TO 閾值 PERCENT。

table_name

要清空的資料表名稱。如果您未指定資料表名稱，則清空操作會套用至目前資料庫中的所有資料表。您可以指定任何使用者建立的永久或臨時資料表。此命令對於其他物件並無意義，例如檢視和系統資料表。

如果您包含 TO 閾值PERCENT參數，則需要資料表名稱。

RECLUSTER 資料表名稱

排序資料表中未排序的部分。已依自動資料表排序排序的資料表部分會保持不變。此命令不會將新排序的資料與已排序的區域合併。也不會回收標記為刪除的所有空間。當此命令完成時，資料表可能不會完全排序，如 SVV_TABLE_ 中的unsorted欄位所示INFO。

建議您VACUUMRECLUSTER將用於經常擷取和查詢且僅存取最新資料的大型資料表。

VACUUM RECLUSTER 不支援 TO 閾值 PERCENT。RECLUSTER 如果使用 `table_name`，則需要資料表名稱。

VACUUM RECLUSTER 在具有交錯排序索引鍵的資料表和具有ALL分佈樣式的資料表上不支援。

`table_name`

要清空的資料表名稱。您可以指定任何使用者建立的永久或臨時資料表。此命令對於其他物件並無意義，例如檢視和系統資料表。

結束閾值 PERCENT

指定上述閾值的子句，其會VACUUM略過排序階段，以及用於在刪除階段中回收空間的目標閾值。排序閾值是清空之前，指定資料列中已依照排序順序排列的總列數百分比。刪除閾值是清空之後，未標記為要進行刪除的總列數百分比下限。

由於只有在資料表中排序資料列的百分比低於排序閾值時，VACUUM才會重新排序資料列，因此 Amazon Redshift 通常可以大幅縮短VACUUM時間。同樣地，當 VACUUM 沒有限制從 100% 標記為刪除的資料列中回收空間時，通常可以略過僅包含幾個已刪除資料列的重寫區塊。

例如，如果您為閾值指定 75，則如果資料表的 75% 或更多資料列已按排序順序排列，則會 VACUUM略過排序階段。對於刪除階段，VACUUMS會設定回收磁碟空間的目標，使得至少 75% 的資料表資料列在真空之後不會標記為刪除。閾值必須是介於 0 到 100 之間的整數。預設為 95。如果您指定 100 的值，VACUUM一律會排序資料表，除非資料表已完全排序，並從標記為刪除的所有資料列中回收空間。如果您指定 0 的值，則VACUUM絕不會排序資料表，也絕不會回收空間。

如果您包含 TO 閾值PERCENT參數，則還必須指定資料表名稱。如果省略資料表名稱，會 VACUUM失敗。

您無法搭配使用 TO 閾值PERCENT參數REINDEX。

BOOST

使用可用的其他資源執行VACUUM命令，例如記憶體和磁碟空間。透過 BOOST選項，在一個視窗中VACUUM操作，並在VACUUM操作期間封鎖並行刪除和更新。使用 BOOST選項執行會爭用系統資源，這可能會影響查詢效能。在系統上負載很亮VACUUMBOOST時執行，例如在維護操作期間。

使用 BOOST選項時，請考慮下列事項：

- 指定 BOOST 時，需要 `table_name` 值。
- BOOST 不支援 REINDEX。

- BOOST 會以 DELETE 忽略 ONLY。

使用須知

對於大多數 Amazon Redshift 應用程式，建議您執行完整清空。如需詳細資訊，請參閱[清空資料表](#)。

在執行清空操作之前，請注意以下行為：

- 您無法在交易區塊 VACUUM 內執行 (BEGIN ... END)。如需交易的相關資訊，請參閱[可序列化隔離](#)。
- 您可以隨時在叢集上執行一個 VACUUM 命令。若您嘗試同時執行多個清空操作，Amazon Redshift 會傳回錯誤。
- 當資料庫清空時，資料庫可能會稍微擴大。在沒有刪除的資料列可回收，或資料表的新排序順序導致資料壓縮比率降低時，這是預期的行為。
- 在清空操作期間，查詢效能預期會受到某種程度的影響。清空操作完成後，就會恢復正常效能。
- 在清空操作期間，同時的寫入操作會繼續執行，但不建議在清空時執行寫入操作。較有效率的方式，是在執行清空之前，先完成寫入操作。此外，在清空操作開始之後寫入的任何資料，都無法藉由該次操作清空。在此情況下，將需要執行第二次清空操作。
- 若載入或插入操作已在進行中，則清空操作可能無法開始執行。清空操作會暫時需要資料表的獨佔存取權，以開始執行。需要此獨佔存取權的時間很短，因此清空操作不會長時間封鎖同時執行的載入和插入操作。
- 若沒有可對特殊資料表執行的工作，則會略過清空操作；不過，在發現可略過該操作的過程中，可能會產生一些額外成本。若您知道這是原始資料表，或未達清空閾值，則不要對它執行清空操作。
- 小型資料表上的 DELETE ONLY 真空操作可能不會減少用於存放資料的區塊數量，特別是當資料表具有大量資料欄或叢集在每個節點使用大量切片時。這些清空操作會對每個分割的每個資料欄新增一個區塊，用來負責同時插入資料表的工作，而此額外負荷有可能超過回收磁碟空間所減少的區塊數。例如，若 8 個節點叢集上的 10 欄資料表在清空前佔用了 1000 個區塊，則清空將不會減少實際的區塊數，除非因為刪除的資料列而回收了超過 80 個區塊的磁碟空間 (每個資料區塊都使用 1 MB)。

如果符合以下任何條件，則自動清空操作會暫停：

- 使用者執行資料定義語言 (DDL) ALTER 操作，例如 TABLE，需要對目前正在進行自動吸塵的資料表進行專屬鎖定。
- 叢集中任何資料表 VACUUM 上的使用者觸發程序 (一次只能 VACUUM 執行一個)。
- 高叢集負載的期間。

範例

根據預設的 95% 清空閾值回收空間和資料庫，並排序所有資料表中的資料列。

```
vacuum;
```

根據預設的 95% 閾值來回收空間並重新排序SALES資料表中的資料列。

```
vacuum sales;
```

一律回收SALES表中的空間並重新排序資料列。

```
vacuum sales to 100 percent;
```

只有在少於 75% 的資料列已排序時，才能重新排序SALES資料表中的資料列。

```
vacuum sort only sales to 75 percent;
```

回收SALES資料表中的空間，讓至少有 75% 的剩餘資料列在真空之後不會標記為刪除。

```
vacuum delete only sales to 75 percent;
```

重新索引，然後對LISTING資料表進行吸塵。

```
vacuum reindex listing;
```

下列命令會傳回錯誤。

```
vacuum reindex listing to 75 percent;
```

叢集，然後清空LISTING資料表。

```
vacuum recluster listing;
```

叢集，然後使用 BOOST選項對LISTING資料表進行吸塵。

```
vacuum recluster listing boost;
```

SQL 函數參考

主題

- [僅限領導節點函數](#)
- [彙總函數](#)
- [陣列函數](#)
- [位元彙整函數](#)
- [條件式運算式](#)
- [資料類型格式化函數](#)
- [日期和時間函數](#)
- [雜湊函數](#)
- [HyperLogLog 函數](#)
- [JSON 函數](#)
- [機器學習函數](#)
- [數學函數](#)
- [物件函數](#)
- [空間函數](#)
- [字串函數](#)
- [SUPER 類型資訊函數](#)
- [VARBYTE 函數與運算子](#)
- [範圍函數](#)
- [系統管理函數](#)
- [系統資訊函數](#)

Amazon Redshift 支援許多延伸到 SQL 標準的函數，以及標準彙總函數、純量函數和視窗函數。

Note

Amazon Redshift 是以 Postgre 為基礎 SQL。Amazon Redshift 和 PostgreSQL 在設計和開發資料倉儲應用程式時，您必須注意許多非常重要的差異。如需 Amazon Redshift 與 Postgre 有何 SQL 不同的詳細資訊 SQL，請參閱 [Amazon Redshift 和 PostgreSQL](#)。

僅限領導節點函數

某些 Amazon Redshift 查詢會分配到運算節點上執行；其他的查詢則是只在領導者節點上執行。

當查詢參考使用者建立的資料表或系統資料表（具有 STL 或 STV 字首的資料表和具有 SVL 或 SVV 字首的系統檢視）時，領導節點會分發 SQL 到運算節點。僅參考目錄資料表（具有 PG 字首的資料表，例如 PG_TABLE_DEF）或不參考任何資料表的查詢，僅在領導節點上執行。

某些 Amazon Redshift SQL 函數僅支援領導節點，運算節點不支援。使用領導者節點函數的查詢必須完全在領導者節點上執行，而不是在運算節點上，否則會傳回錯誤。

每個僅限於領導者節點的函數在文件中包含備註，說明如果函數參照使用者定義的資料表或 Amazon Redshift 系統資料表，將會傳回錯誤。

如需詳細資訊，請參閱 [SQL 領導節點上支援的函數](#)。

下列 SQL 函數是僅限領導者節點的函數，運算節點不支援：

系統資訊函數

- CURRENT_SCHEMA
- CURRENT_SCHEMAS
- HAS_DATABASE_PRIVILEGE
- HAS_SCHEMA_PRIVILEGE
- HAS_TABLE_PRIVILEGE

字串函數

- SUBSTR

數學函數

- FACTORIAL()

下列僅限於領導者節點的函數已被棄用並且不再受支援：

日期函數

- AGE

- CURRENT_TIME
- CURRENT_TIMESTAMP
- LOCALTIME
- ISFINITE
- NOW

字串函數

- GETBIT
- GET_BYTE
- SET_BIT
- SET_BYTE
- 結束_ASCII

彙總函數

主題

- [ANY_VALUE 函數](#)
- [APPROXIMATE PERCENTILE_DISC 函數](#)
- [AVG 函數](#)
- [COUNT 函數](#)
- [LISTAGG 函數](#)
- [MAX 函數](#)
- [MEDIAN 函數](#)
- [MIN 函數](#)
- [PERCENTILE_CONT 函數](#)
- [STDDEV_SAMP 和 STDDEV_POP 函數](#)
- [SUM 函數](#)
- [VAR_SAMP 和 VAR_POP 函數](#)

彙總函數從一組輸入值計算單一結果值。

SELECT 使用彙總函數的 陳述式可包含兩個選子句：GROUPBY 和 HAVING。這些子句的語法如下所示（使用 COUNT函數作為範例）：

```
SELECT count (*) expression FROM table_reference
WHERE condition [GROUP BY expression ] [ HAVING condition]
```

GROUP BY 子句會依指定資料欄或資料欄中的唯一值彙總和分組結果。HAVING 子句會將傳回的結果限制為特定彙總條件為 true 的資料列，例如計數 (*) > 1。子HAVING句的使用方式與根據資料欄的值WHERE限制資料列的方式相同。如需這些額外子句的範例，請參閱 [COUNT](#)。

彙總函數不接受巢狀彙總函數或範圍函數做為引數。

ANY_VALUE 函數

ANY_VALUE 函數會從輸入表達式值中傳回任何值，但無法確定。如果輸入表達式不會傳回任何資料列，此函數會傳回 NULL。如果輸入表達式中有 NULL 值，函數也會傳回 NULL。

語法

```
ANY_VALUE( [ DISTINCT | ALL ] expression )
```

引數

DISTINCT | ALL

指定 DISTINCT或 從輸入表達式值ALL傳回任何值。DISTINCT 引數沒有效果，因此會被忽略。

expression

函數運算的目標欄或運算式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- BOOLEAN
- CHAR
- VARCHAR

- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- INTERVAL YEAR 至 MONTH
- INTERVAL DAY 至 SECOND
- VARBYTE
- SUPER
- HLLSKETCH
- GEOMETRY
- GEOGRAPHY

傳回值

傳回與 expression 相同的資料類型。

使用須知

如果指定資料欄 ANY_VALUE 函數的陳述式也包含第二個資料欄參考，則第二個資料欄必須出現在 GROUP BY 子句中或包含在彙總函數中。

範例

這些範例使用《Amazon Redshift 入門指南》中的[步驟 4：從 Amazon S3 載入範例資料](#)中建立的事件表格。下列範例會傳回事件名稱為 Eagles 的任何 dateid 的執行個體。

```
select any_value(dateid) as dateid, eventname from event where eventname = 'Eagles'
group by eventname;
```

以下是結果。

```
dateid | eventname
-----+-----
1878   | Eagles
```

下列範例會傳回事件名稱為 Eagles 或 Cold War Kids 的任何 dateid 的執行個體。

```
select any_value(dateid) as dateid, eventname from event where eventname in('Eagles',
'Cold War Kids') group by eventname;
```

以下是結果。

```
dateid | eventname
-----+-----
 1922  | Cold War Kids
 1878  | Eagles
```

APPROXIMATE PERCENTILE_DISC 函數

APPROXIMATE PERCENTILE_DISC 是假設離散分佈模型的反分佈函數。它採用百分位數值和排序規格，且會傳回給定集裡的一個元素。近似法可讓函數執行較快，其相對錯誤率低到約 0.5%。

對於指定的百分位數值，PERCENTILE_APPROXIMATE DISC 使用四分位數摘要演算法來近似 ORDER BY 子句中表達式的離散百分位數。APPROXIMATE PERCENTILE_DISC 傳回累積分佈值最小的值（相對於相同的排序規格），大於或等於百分位數。

語法

```
APPROXIMATE PERCENTILE_DISC ( percentile )
WITHIN GROUP ( ORDER BY expr )
```

引數

percentile

介於 0 和 1 之間的數值常數。計算時會忽略 Null。

WITHIN GROUP (ORDER BY *expr*)

此子句指定要排序和計算百分位數的數值或日期/時間值。

傳回值

與WITHINGROUP子句中的 ORDER BY 表達式相同的資料類型。

使用須知

如果 APPROXIMATE PERCENTILE_DISC 陳述式包含 GROUP BY 子句，則結果集會受到限制。限制會根據節點類型和節點數目而不同。如果超出限制，函數會失敗並傳回下列錯誤。

GROUP BY limit for approximate percentile_disc exceeded.

如果您需要評估的組數超過限制所允許，請考慮使用 [PERCENTILE_CONT 函數](#)。

範例

下列範例傳回排名前 10 個日期的銷售數量、銷售總計及第五個百分位數值。

```
select top 10 date.caldate,
count(totalprice), sum(totalprice),
approximate percentile_disc(0.5)
within group (order by totalprice)
from listing
join date on listing.dateid = date.dateid
group by date.caldate
order by 3 desc;
```

caldate	count	sum	percentile_disc
2008-01-07	658	2081400.00	2020.00
2008-01-02	614	2064840.00	2178.00
2008-07-22	593	1994256.00	2214.00
2008-01-26	595	1993188.00	2272.00
2008-02-24	655	1975345.00	2070.00
2008-02-04	616	1972491.00	1995.00
2008-02-14	628	1971759.00	2184.00
2008-09-01	600	1944976.00	2100.00
2008-07-29	597	1944488.00	2106.00
2008-07-23	592	1943265.00	1974.00

AVG 函數

AVG 函數會傳回輸入表達式值的平均值（算術平均值）。AVG 函數使用數值並忽略NULL值。

語法

```
AVG ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER
- BIGINT
- NUMERIC
- DECIMAL
- REAL
- DOUBLE PRECISION
- SUPER

DISTINCT | ALL

使用引數時DISTINCT，函數會先從指定的表達式中消除所有重複的值，然後再計算平均值。使用引數 ALL，函數會保留運算式中的所有重複值，以計算平均值。ALL 為預設值。

資料類型

AVG 函數支援的引數類型為 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL、DOUBLE PRECISION和 SUPER。

AVG 函數支援的傳回類型為：

- BIGINT 針對任何整數類型引數
- DOUBLE PRECISION 針對浮點數引數
- 針對任何其他引數類型，傳回與運算式相同的資料類型。

具有 NUMERIC或 DECIMAL引數的AVG函數結果的預設精確度為 38。結果的小數位數和引數的小數位數相同。例如，DEC (5 , 2) 欄AVG的傳回 DEC (38 , 2) 資料類型。

範例

從SALES資料表尋找每筆交易的平均售出數量：

```
select avg(qtysold)from sales;

avg
-----
2
```

```
(1 row)
```

尋找所有列表所列出的平均總價：

```
select avg(numtickets*priceperticket) as avg_total_price from listing;
```

```
avg_total_price
-----
3034.41
(1 row)
```

尋找平均支付價格，按月份分組，依遞減順序排列：

```
select avg(pricepaid) as avg_price, month
from sales, date
where sales.dateid = date.dateid
group by month
order by avg_price desc;
```

```
avg_price | month
-----+-----
659.34 | MAR
655.06 | APR
645.82 | JAN
643.10 | MAY
642.72 | JUN
642.37 | SEP
640.72 | OCT
640.57 | DEC
635.34 | JUL
635.24 | FEB
634.24 | NOV
632.78 | AUG
(12 rows)
```

COUNT 函數

COUNT 函數會計算運算式定義的資料列。

COUNT 函數有下列變化。

- COUNT (*) 會計算目標資料表中的所有資料列，無論是否包含 null。

- COUNT (表達式) 會計算特定資料欄或表達式中具有非NULL值的資料列數目。
- COUNT (DISTINCT 表達式) 會計算資料欄或表達式中不同非NULL值的數量。
- APPROXIMATE COUNT DISTINCT 近似資料欄或運算式中不同非NULL值的數量。

語法

```
COUNT( * | expression )
```

```
COUNT ( [ DISTINCT | ALL ] expression )
```

```
APPROXIMATE COUNT ( DISTINCT expression )
```

引數

expression

函數運算的目標欄或表達式。COUNT 函數支援所有引數資料類型。

DISTINCT | ALL

使用引數時DISTINCT，函數會先從指定的表達式中消除所有重複的值，然後再進行計數。使用引數 ALL，函數會保留運算式中的所有重複值，以進行計數。ALL 為預設值。

APPROXIMATE

與 搭配使用時APPROXIMATE，COUNTDISTINCT函數會使用 HyperLogLog 演算法來估算資料欄或運算式中不同的非NULL值數量。使用 APPROXIMATE 關鍵字的查詢執行速度更快，相對錯誤較低，約為 2%。如果每個查詢或每一組 (若有 group by 子句) 傳回數百萬個以上的大量相異值，則查詢一定要採用近似法。如果相異值較少 (數千個)，則近似法可能比精確計數更慢。APPROXIMATE 只能與 COUNT 搭配使用DISTINCT。

傳回類型

COUNT 函數會傳回 BIGINT。

範例

計算佛羅里達州的所有使用者人數：

```
select count(*) from users where state='FL';
```

```
count
-----
510
```

從EVENT資料表中計算所有事件名稱：

```
select count(eventname) from event;
```

```
count
-----
8798
```

從EVENT資料表中計算所有事件名稱：

```
select count(all eventname) from event;
```

```
count
-----
8798
```

從IDsEVENT資料表中計算所有唯一的場地：

```
select count(distinct venueid) as venues from event;
```

```
venues
-----
204
```

計算每個賣方列出整批銷售門票超過四張的次數。結果依賣方 ID 分組：

```
select count(*), sellerid from listing
where numtickets > 4
group by sellerid
order by 1 desc, 2;
```

```
count | sellerid
-----+-----
12    |    6386
11    |   17304
```



```

11      |      20123
11      |      25428
...

```

下列範例比較 `count` 和 `APPROXIMATE` 的傳回值 `COUNT` 和執行時間 `COUNT`。

```

select count(distinct pricepaid) from sales;

count
-----
 4528

Time: 48.048 ms

select approximate count(distinct pricepaid) from sales;

count
-----
 4553

Time: 21.728 ms

```

LISTAGG 函數

對於查詢中的每個群組，`LISTAGG` 彙總函數會根據 `ORDER BY` 表達式排序該群組的資料列，然後將值串連成單一字串。

語法

```

LISTAGG( [DISTINCT] aggregate_expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]

```

引數

DISTINCT

此子句在串連值之前會從指定的運算式中消除重複值。忽略結尾空格。例如，字串 `'a'` 和 `'a '` 被視為重複。`LISTAGG` 會使用遇到的第一個值。如需詳細資訊，請參閱 [多餘空格的意義](#)。

aggregate_expression

任何有效運算式 (例如欄名) , 用於提供要彙總的值。NULL 值和空字串會被忽略。

delimiter

用來區隔串連值的字串常數。預設值為 NULL。

WITHIN GROUP (ORDER BY order_list)

此子句指定彙總值的排序順序。

傳回值

VARCHAR (MAX)。如果結果集大於VARCHAR大小上限 , LISTAGG會傳回下列錯誤 :

```
Invalid operation: Result size exceeds LISTAGG limit
```

使用須知

- 如果陳述式包含多個使用WITHINGROUP子句的LISTAGG函數 , 則每個WITHINGROUP子句都必須使用相同的 ORDER BY 值。

例如 , 下列陳述式會傳回錯誤。

```
SELECT LISTAGG(sellerid)
WITHIN GROUP (ORDER BY dateid) AS sellers,
LISTAGG(dateid)
WITHIN GROUP (ORDER BY sellerid) AS dates
FROM sales;
```

下列陳述式會成功執行。

```
SELECT LISTAGG(sellerid)
WITHIN GROUP (ORDER BY dateid) AS sellers,
LISTAGG(dateid)
WITHIN GROUP (ORDER BY dateid) AS dates
FROM sales;
```

```
SELECT LISTAGG(sellerid)
WITHIN GROUP (ORDER BY dateid) AS sellers,
LISTAGG(dateid) AS dates
```

```
FROM sales;
```

範例

下列範例會彙總賣方 IDs，依賣方 ID 排序。

```
SELECT LISTAGG(sellerid, ', ')
WITHIN GROUP (ORDER BY sellerid)
FROM sales
WHERE eventid = 4337;
```

listagg

380, 380, 1178, 1178, 1178, 2731, 8117, 12905, 32043, 32043, 32043, 32432, 32432,
38669, 38750, 41498, 45676, 46324, 47188, 47188, 48294

下列範例使用 DISTINCT 來傳回唯一賣方的清單 IDs。

```
SELECT LISTAGG(DISTINCT sellerid, ', ')
WITHIN GROUP (ORDER BY sellerid)
FROM sales
WHERE eventid = 4337;
```

listagg

380, 1178, 2731, 8117, 12905, 32043, 32432, 38669, 38750, 41498, 45676, 46324, 47188,
48294

下列範例 IDs 會依日期順序彙總賣方。

```
SELECT LISTAGG(sellerid, ', ')
WITHIN GROUP (ORDER BY dateid)
FROM sales
WHERE eventid = 4337;
```

listagg

41498, 47188, 47188, 1178, 1178, 1178, 380, 45676, 46324, 48294, 32043, 32043, 32432,
12905, 8117, 38750, 2731, 32432, 32043, 380, 38669

下列範例以縱線分隔清單傳回 ID 為 660 的買方的銷售日期。

```
SELECT LISTAGG(
  (SELECT caldate FROM date WHERE date.dateid=sales.dateid), ' | '
)
WITHIN GROUP (ORDER BY sellerid DESC, salesid ASC)
FROM sales
WHERE buyerid = 660;

          listagg
-----
2008-07-16 | 2008-07-09 | 2008-01-01 | 2008-10-26
```

下列範例會傳回買方 660、661 和 662 IDs 的逗號分隔銷售清單。

```
SELECT buyerid,
LISTAGG(salesid,', ')
WITHIN GROUP (ORDER BY salesid) AS sales_id
FROM sales
WHERE buyerid BETWEEN 660 AND 662
GROUP BY buyerid
ORDER BY buyerid;

buyerid |          sales_id
-----+-----
660     | 32872, 33095, 33514, 34548
661     | 19951, 20517, 21695, 21931
662     | 3318, 3823, 4215, 51980, 53202, 55908, 57832, 171603
```

MAX 函數

MAX 函數會傳回一組資料列中的最大值。DISTINCT 或 ALL 可能被使用，但不會影響結果。

語法

```
MAX ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER
- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- VARBYTE
- SUPER

DISTINCT | ALL

使用引數時DISTINCT，函數會在計算最大值之前，從指定的表達式消除所有重複值。使用引數ALL時，函數會保留運算式中的所有重複值，以計算最大值。ALL 為預設值。

資料類型

傳回與 expression 相同的資料類型。MIN 函數的布林值相等值是 [BOOL_AND 函數](#)，而布林值相等值MAX是 [BOOL_OR 函數](#)。

範例

從所有銷售中尋找最高支付價格：

```
select max(pricepaid) from sales;

max
-----
12624.00
```

```
(1 row)
```

從所有銷售中尋找每張門票的最高支付價格：

```
select max(pricepaid/qtysold) as max_ticket_price
from sales;
```

```
max_ticket_price
-----
2500.000000000
(1 row)
```

MEDIAN 函數

計算值範圍的中值。忽略範圍內的 NULL 值。

MEDIAN 是假設連續分佈模型的反分佈函數。

MEDIAN 是 的特殊案例 [PERCENTILE_CONT](#)。

語法

```
MEDIAN(median_expression)
```

引數

median_expression

函數運算的目標欄或表達式。

資料類型

傳回類型取決於 *median_expression* 的資料類型。下表顯示每一個 *median_expression* 資料類型的傳回類型。

輸入類型	傳回類型
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE

輸入類型	傳回類型
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

使用須知

如果 `median_expression` 引數是定義的最高精確度為 38 位數的 DECIMAL 資料類型，MEDIAN 則可能會傳回不正確的結果或錯誤。如果 MEDIAN 函數的傳回值超過 38 位數，則結果會截斷為適合，這會導致精確度損失。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些情況，建議使用精確度較低的資料類型，或將 `median_expression` 引數轉換為較低精確度。

如果陳述式包含以排序方式彙總函數的多個呼叫（LISTAGG、PERCENTILE_ 或 MEDIAN）CONT，則它們都必須使用相同的 ORDER BY 值。請注意，會在表達式值上 MEDIAN 套用隱含順序。

例如，下列陳述式會傳回錯誤。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;
```

An error occurred when executing the SQL command:

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;
```

ERROR: within group ORDER BY clauses for aggregate functions must be the same

下列陳述式會成功執行。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
```

```

MEDIAN(salesid)
FROM sales
GROUP BY salesid, pricepaid;

```

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

下列範例顯示 MEDIAN會產生與 PERCENTILE_CONT (0.5) 相同的結果。

```

SELECT TOP 10 DISTINCT sellerid, qtysold,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold),
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid, qtysold;

```

sellerid	qtysold	percentile_cont	median
2	2	2	2
26	1	1	1
33	1	1	1
38	1	1	1
43	1	1	1
48	2	2	2
48	3	3	3
77	4	4	4
85	4	4	4
95	2	2	2

下列範例會找出每個 sellerid 售出的中位數量。

```

SELECT sellerid,
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid
ORDER BY sellerid
LIMIT 10;

```

sellerid	median
----------	--------


```

|      1 |      1.5 |
|      2 |      2 |
|      3 |      2 |
|      4 |      2 |
|      5 |      1 |
|      6 |      1 |
|      7 |      1.5 |
|      8 |      1 |
|      9 |      4 |
|     12 |      2 |
+-----+-----+

```

若要驗證第一個 sellerid 的先前查詢結果，請使用下列範例。

```

SELECT qty sold
FROM sales
WHERE sellerid=1;

```

```

+-----+
| qty sold |
+-----+
|      2 |
|      1 |
+-----+

```

MIN 函數

MIN 函數會傳回一組資料列中的最小值。DISTINCT 或 ALL 可能被使用，但不會影響結果。

語法

```

MIN ( [ DISTINCT | ALL ] expression )

```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT
- INTEGER

- BIGINT
- DECIMAL
- REAL
- DOUBLE PRECISION
- CHAR
- VARCHAR
- DATE
- TIMESTAMP
- TIMESTAMPTZ
- TIME
- TIMETZ
- VARBYTE
- SUPER

DISTINCT | ALL

使用引數時DISTINCT，函數會在計算最小值之前，從指定的表達式消除所有重複值。使用引數ALL，函數會保留運算式中的所有重複值，以計算最小值。ALL 為預設值。

資料類型

傳回與 expression 相同的資料類型。MIN 函數的布林值相等為 [BOOL_AND 函數](#)，而布林值相等MAX 為 [BOOL_OR 函數](#)。

範例

從所有銷售中尋找最低支付價格：

```
select min(pricepaid) from sales;

min
-----
20.00
(1 row)
```

從所有銷售中尋找每張門票的最低支付價格：

```
select min(pricepaid/qtysold)as min_ticket_price
from sales;
```

```
min_ticket_price
-----
20.000000000
(1 row)
```

PERCENTILE_CONT 函數

PERCENTILE_CONT 是假設連續分佈模型的反分佈函數。它採用百分位數值和排序規格，且會傳回插入值，該值將根據排序規格落入給定的百分位數值。

PERCENTILE_CONT 會在值排序後計算值之間的線性插補。此函數在列根據排序規格來排序後，使用彙總群組中的百分位數值 (P) 和非 Null 列數 (N) 來計算列號。此列號 (RN) 是根據公式 $RN = (1 + (P * (N - 1)))$ 來計算。彙總函數的最終結果是以列號 $CRN = \text{CEILING}(RN)$ 到 $FRN = \text{FLOOR}(RN)$ 各列的值之間的線性插值來計算。

最終結果如下。

如果 $(CRN = FRN = RN)$ ，則結果為 (value of expression from row at RN)

否則結果如下：

$(CRN - RN) * (\text{value of expression for row at FRN}) + (RN - FRN) * (\text{value of expression for row at CRN})$.

語法

```
PERCENTILE_CONT(percentile)
WITHIN GROUP(ORDER BY expr)
```

引數

percentile

介於 0 到 1 之間的數值常數。計算中會忽略 NULL 值。

expr

指定要排序和計算百分位數的數值或日期/時間值。

傳回值

傳回類型取決於子WITHINGROUP句中 ORDER BY 表達式的資料類型。下表顯示每個 ORDER BY 表達式資料類型的傳回類型。

輸入類型	傳回類型
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP
TIMESTAMPTZ	TIMESTAMPTZ

使用須知

如果 ORDER BY 表達式是定義的最高精確度為 38 位數的 DECIMAL 資料類型，PERCENTILE_CONT 可能會傳回不正確的結果或錯誤。如果 PERCENTILE_CONT 函數的傳回值超過 38 位數，則結果會截斷為適合，這會導致精確度損失。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些條件，我們建議您使用精確度較低的資料類型，或將 ORDER BY 表達式轉換為精確度較低的資料類型。

如果陳述式包含以排序方式彙總函數的多個呼叫（LISTAGG、PERCENTILE_ 或 MEDIAN）CONT，則它們都必須使用相同的 ORDER BY 值。請注意，會在表達式值上 MEDIAN 套用的隱含順序。

例如，下列陳述式會傳回錯誤。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
FROM sales
GROUP BY salesid, pricepaid;
```

```
An error occurred when executing the SQL command:
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(pricepaid)
```

```
FROM sales
GROUP BY salesid, pricepaid;

ERROR: within group ORDER BY clauses for aggregate functions must be the same
```

下列陳述式會成功執行。

```
SELECT TOP 10 salesid, SUM(pricepaid),
PERCENTILE_CONT(0.6) WITHIN GROUP(ORDER BY salesid),
MEDIAN(salesid)
FROM sales
GROUP BY salesid, pricepaid;
```

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

下列範例顯示 PERCENTILE_CONT (0.5) 會產生與 相同的結果。 MEDIAN

```
SELECT TOP 10 DISTINCT sellerid, qtysold,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold),
MEDIAN(qtysold)
FROM sales
GROUP BY sellerid, qtysold;
```

sellerid	qtysold	percentile_cont	median
2	2	2	2
26	1	1	1
33	1	1	1
38	1	1	1
43	1	1	1
48	2	2	2
48	3	3	3
77	4	4	4
85	4	4	4
95	2	2	2

下列範例會尋找SALES資料表中每個 sellerid 的售出數量 PERCENTILE_CONT (0.5) 和 PERCENTILE_CONT (0.75) 。

```

SELECT sellerid,
PERCENTILE_CONT(0.5) WITHIN GROUP(ORDER BY qtysold) as pct_05,
PERCENTILE_CONT(0.75) WITHIN GROUP(ORDER BY qtysold) as pct_075
FROM sales
GROUP BY sellerid
ORDER BY sellerid
LIMIT 10;

```

```

+-----+-----+-----+
| sellerid | pct_05 | pct_075 |
+-----+-----+-----+
|         1 |      1.5 |      1.75 |
|         2 |       2 |      2.25 |
|         3 |       2 |       3 |
|         4 |       2 |       2 |
|         5 |       1 |      1.5 |
|         6 |       1 |       1 |
|         7 |      1.5 |      1.75 |
|         8 |       1 |       1 |
|         9 |       4 |       4 |
|        12 |       2 |      3.25 |
+-----+-----+-----+

```

若要驗證第一個 sellerid 的先前查詢結果，請使用下列範例。

```

SELECT qtysold
FROM sales
WHERE sellerid=1;

```

```

+-----+
| qtysold |
+-----+
|         2 |
|         1 |
+-----+

```

STDDEV_SAMP 和 STDDEV_POP 函數

STDDEV_SAMP 和 STDDEV_POP 函數會傳回一組數值（整數、小數點或浮點）的範例和母群體標準差。STDDEV_SAMP 函數的結果相當於相同一組值之範例變異數的平方根。

STDDEV_SAMP 和 STDDEV 是相同函數的同義詞。

語法

```
STDDEV_SAMP | STDDEV ( [ DISTINCT | ALL ] expression)
STDDEV_POP ( [ DISTINCT | ALL ] expression)
```

表達式必須為整數、小數或浮點數資料類型。不論表達式的資料類型為何，此函數的傳回類型都是雙精確度數字。

Note

標準差是採用浮點運算來計算，所得結果可能稍不精確。

使用須知

當範例標準差（STDDEV 或 STDDEV_SAMP）計算為由單一值組成的表達式時，函數的結果NULL 不是 0。

範例

下列查詢會傳回VENUE資料表資料VENUESEATS欄中的值平均值，接著是相同值集的樣本標準差和母群體標準差。VENUESEATS 是資料INTEGER欄。結果的小數位數簡化到 2 位數。

```
select avg(venueseats),
       cast(stddev_samp(venueseats) as dec(14,2)) stddevsamp,
       cast(stddev_pop(venueseats) as dec(14,2)) stddevpop
from venue;
```

```
avg | stddevsamp | stddevpop
-----+-----+-----
17503 | 27847.76 | 27773.20
(1 row)
```

下列查詢會傳回SALES資料表中資料COMMISSION欄的範例標準差。COMMISSION 是資料DECIMAL欄。結果的小數位數簡化到 10 位數。

```
select cast(stddev(commission) as dec(18,10))
from sales;

stddev
```

```
-----
130.3912659086
(1 row)
```

下列查詢會將資料COMMISSION欄的範例標準差轉換為整數。

```
select cast(stddev(commission) as integer)
from sales;

stddev
-----
130
(1 row)
```

下列查詢會傳回資料COMMISSION欄的範例標準差和範例變異的平方根。這些計算的結果相同。

```
select
cast(stddev_samp(commission) as dec(18,10)) stddevsamp,
cast(sqrt(var_samp(commission)) as dec(18,10)) sqrtvarsamp
from sales;

stddevsamp | sqrtvarsamp
-----+-----
130.3912659086 | 130.3912659086
(1 row)
```

SUM 函數

SUM 函數會傳回輸入資料欄或表達式值的總和。SUM 函數使用數值並忽略NULL值。

語法

```
SUM ( [ DISTINCT | ALL ] expression )
```

引數

expression

函數運算的目標欄或表達式。expression 是下列其中一種資料類型：

- SMALLINT

- INTEGER
- BIGINT
- NUMERIC
- DECIMAL
- REAL
- DOUBLE PRECISION
- SUPER

DISTINCT | ALL

使用引數 DISTINCT 時，函數會先從指定的表達式消除所有重複值，然後再計算總和。使用引數 ALL，函數會保留運算式中的所有重複值，以計算總和。ALL 為預設值。

資料類型

SUM 函數支援的引數類型為 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL、DOUBLE PRECISION 和 SUPER。

SUM 函數支援的傳回類型為

- BIGINT 適用於 BIGINT、SMALLINT 和 INTEGER 引數
- NUMERIC 適用於引 NUMERIC 數
- DOUBLE PRECISION 適用於浮點數引數
- 針對任何其他引數類型，傳回與運算式相同的資料類型。

具有 NUMERIC 或 DECIMAL 引數的 SUM 函數結果的預設精確度為 38。結果的小數位數和引數的小數位數相同。例如，DEC (5 , 2) 欄 SUM 的傳回 DEC (38 , 2) 資料類型。

範例

尋找從 SALES 資料表支付的所有佣金總和：

```
select sum(commission) from sales;

sum
-----
16614814.65
```

```
(1 row)
```

尋找佛羅里達州的所有會場的座位數：

```
select sum(venueSeats) from venue
where venuestate = 'FL';
```

```
sum
-----
250411
(1 row)
```

尋找五月售出的座位數：

```
select sum(qtysold) from sales, date
where sales.dateid = date.dateid and date.month = 'MAY';
```

```
sum
-----
32291
(1 row)
```

VAR_SAMP 和 VAR_POP 函數

VAR_SAMP 和 VAR_POP 函數會傳回一組數值（整數、小數點或浮點）的範例和母群體差異。VAR_SAMP 函數的結果相當於相同值集的平方樣本標準差。

VAR_SAMP 和 VARIANCE 是相同函數的同義詞。

語法

```
VAR_SAMP | VARIANCE ( [ DISTINCT | ALL ] expression )
VAR_POP ( [ DISTINCT | ALL ] expression )
```

表達式必須為整數、小數或浮點數資料類型。不論表達式的資料類型為何，此函數的傳回類型都是雙精確度數字。

Note

這些函數的結果可能隨著資料倉儲叢集而有所不同，視每個案例中的叢集組態而定。

使用須知

當範例變異數 (VARIANCE 或 VAR_SAMP) 針對由單一值組成的表達式計算時，函數的結果NULL不是 0。

範例

下列查詢會傳回LISTING資料表中資料NUMTICKETS欄的四捨五入範例和母群體差異。

```
select avg(numtickets),
round(var_samp(numtickets)) varsamp,
round(var_pop(numtickets)) varpop
from listing;
```

```
avg | varsamp | varpop
-----+-----+-----
10 |      54 |      54
(1 row)
```

下列查詢執行同樣的計算，但將結果轉換為小數值。

```
select avg(numtickets),
cast(var_samp(numtickets) as dec(10,4)) varsamp,
cast(var_pop(numtickets) as dec(10,4)) varpop
from listing;
```

```
avg | varsamp | varpop
-----+-----+-----
10 | 53.6291 | 53.6288
(1 row)
```

陣列函數

接下來，您可以找到 Amazon Redshift SQL 支援存取和操作陣列的陣列函數說明。

主題

- [陣列函數](#)
- [array_concat 函數](#)
- [array_flatten 陣列](#)

- [get_array_length 陣列](#)
- [split_to_array 陣列](#)
- [子陣列函數](#)

陣列函數

建立SUPER資料類型的陣列。

語法

```
ARRAY( [ expr1 ] [ , expr2 [ , ... ] ] )
```

引數

expr1, expr2

日期和時間類型以外的任何 Amazon Redshift 資料類型的表達式，因為 Amazon Redshift 不會將日期和時間類型轉換為SUPER資料類型。引數不需要是相同的資料類型。

傳回類型

陣列函數會傳回SUPER資料類型。

範例

下列範例顯示數值陣列和不同資料類型的陣列。

```
--an array of numeric values
select array(1,50,null,100);
      array
-----
 [1,50,null,100]
(1 row)

--an array of different data types
select array(1,'abc',true,3.14);
      array
-----
 [1,"abc",true,3.14]
```

```
(1 row)
```

array_concat 函數

array_concat 函數串連兩個陣列以建立一個陣列，其中包含第一個陣列中的所有元素，後跟第二個陣列中的所有元素。這兩個引數必須是有效的陣列。

語法

```
array_concat( super_expr1, super_expr2 )
```

引數

super_expr1

指定要串連的兩個陣列中第一個的值。

super_expr2

指定要串連的兩個陣列中第二個的值。

傳回類型

array_concat 函數會傳回SUPER資料值。

範例

下列範例顯示串連相同類型的兩個陣列，以及串連不同類型的兩個陣列。

```
-- concatenating two arrays
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY(10003,10004));
           array_concat
-----
 [10001,10002,10003,10004]
(1 row)

-- concatenating two arrays of different types
SELECT ARRAY_CONCAT(ARRAY(10001,10002),ARRAY('ab','cd'));
           array_concat
-----
 [10001,10002,"ab","cd"]
(1 row)
```

array_flatten 陣列

將多個陣列合併為單一SUPER類型的陣列。

語法

```
array_flatten( super_expr1, super_expr2, ... )
```

引數

super_expr1、*super_expr2*

陣列形式的有效SUPER表達式。

傳回類型

`array_flatten` 函數會傳回SUPER資料值。

範例

下列範例顯示 `array_flatten` 函數。

```
SELECT ARRAY_FLATTEN(ARRAY(ARRAY(1,2,3,4),ARRAY(5,6,7,8),ARRAY(9,10)));
      array_flatten
-----
 [1,2,3,4,5,6,7,8,9,10]
(1 row)
```

get_array_length 陣列

傳回指定陣列的長度。GET_ARRAY_LENGTH 函數會傳回指定物件或SUPER陣列路徑的陣列長度。

語法

```
get_array_length( super_expr )
```

引數

super_expr

陣列形式的有效SUPER表達式。

傳回類型

`get_array_length` 函數會傳回 BIGINT。

範例

下列範例顯示 `get_array_length` 函數。

```
SELECT GET_ARRAY_LENGTH(ARRAY(1,2,3,4,5,6,7,8,9,10));
get_array_length
-----
                10
(1 row)
```

split_to_array 陣列

使用分隔符號做為選擇性的參數。如果沒有分隔符號，則預設值為逗號。

語法

```
split_to_array( string, delimiter )
```

引數

string

要分割的輸入字串。

delimiter

將根據其分割輸入字串的選用值。預設為逗號。

傳回類型

`split_to_array` 函數會傳回 SUPER 資料值。

範例

下列範例顯示 `split_to_array` 函數。

```
SELECT SPLIT_TO_ARRAY('12|345|6789', '|');
split_to_array
-----
```

```
["12", "345", "6789"]  
(1 row)
```

子陣列函數

操控陣列以傳回輸入陣列的子集。

語法

```
SUBARRAY( super_expr, start_position, length )
```

引數

super_expr

陣列形式的有效SUPER表達式。

start_position

陣列中開始擷取的位置，從索引位置 0 開始。負位置從陣列的末尾向後計數。

長度

要擷取的元素數 (子字串的長度)。

傳回類型

子陣列函數會傳回SUPER資料值。

範例

下列為子陣列函數的範例。

```
SELECT SUBARRAY(ARRAY('a', 'b', 'c', 'd', 'e', 'f'), 2, 3);  
subarray  
-----  
["c", "d", "e"]  
(1 row)
```

位元彙整函數

位元彙總函數會運算位元操作，以執行整數欄以及可轉換或四捨五入為整數值的彙總。

主題

- [在位元彙總NULLs中使用](#)
- [DISTINCT 支援位元彙總](#)
- [位元函數的概述範例](#)
- [BIT_AND 函數](#)
- [BIT_OR 函數](#)
- [BOOL_AND 函數](#)
- [BOOL_OR 函數](#)

在位元彙總NULLs中使用

當您將位元函數套用至可空值的資料欄時，任何NULL值都會在計算函數結果之前消除。如果沒有資料列符合彙總資格，位元函數會傳回 NULL。同樣行為適用於一般彙總函數。以下是範例。

```
select sum(venueSeats), bit_and(venueSeats) from venue
where venueSeats is null;
```

```
sum | bit_and
-----+-----
null |      null
(1 row)
```

DISTINCT 支援位元彙總

與其他彙總函數一樣，位元函數也支援 DISTINCT 關鍵字。

不過，DISTINCT 搭配這些函數使用 不會影響結果。值的第一個執行個體足以滿足位元AND或 OR 操作。如果正在評估的運算式中存在重複值，則沒有任何區別。

由於DISTINCT處理可能會產生一些查詢執行額外負荷，我們建議您不要DISTINCT與位元函數搭配使用。

位元函數的概述範例

在下文中，您可以找到一些概述範例，展示如何使用位元函數。您還可以找到具有每個函數描述的特定代碼範例。

位元函數的範例是以TICKIT範例資料庫為基礎。TICKIT 範例資料庫中的USERS資料表包含數個布林值資料欄，指出每個使用者是否已知喜歡不同類型的事件，例如運動、劇院、歌劇等。範例如下。

```
select userid, username, lastname, city, state,
likesports, liketheatre
from users limit 10;
```

```
userid | username | lastname | city | state | likesports | liketheatre
-----+-----+-----+-----+-----+-----+-----
1 | JSG99FHE | Taylor | Kent | WA | t | t
9 | MSD36KVR | Watkins | Port Orford | MD | t | f
```

假設USERS資料表的新版本是以不同的方式建置。在此新版本中，單一整數欄定義 (以二進位格式) 每一個使用者喜歡或不喜歡的八種活動。在這種設計中，每個位置代表一種類型的事件。喜歡所有八種類型的使用者將全部八位設定為 1 (如下表的第一列)。如果使用者不喜歡任何活動，則八個位元全部設為 0 (請看第二列)。接下來的第三列表示只喜歡運動和爵士樂的使用者。

USER	SPORTS	THEATRE	JAZZ	OPERA	ROCK	VEGAS	BROADW	CLASSICAL
使用者 1	1	1	1	1	1	1	1	1
使用者 2	0	0	0	0	0	0	0	0
使用者 3	1	0	1	0	0	0	0	0

在資料庫資料表中，這些二進位值可以整數形式存放在單一資料LIKES欄中，如下所示。

使用者	二進位值	儲存的值 (整數)
使用者 1	11111111	255
使用者 2	00000000	0
使用者 3	10100000	160

BIT_AND 函數

BIT_AND 函數會對單一整數資料欄或運算式中的所有值執行位元AND操作。此函數會彙總每一個二進位值 (對應於表達式中的每一個整數值) 的每一個位元。

0 如果所有值的位元都未設定為 1，則 BIT_AND 函數會傳回的結果。如果所有值之中有一或多個位元設定為 1，此函數會傳回整數值。此整數是對應於那些位元的二進位值的數字。

例如，資料表有一欄包含四個整數值：3、7、10 及 22。這些整數以二進位格式表示如下：

Integer	二進位值
3	11
7	111
10	1010
22	10110

此資料集上的 BIT_AND 操作會發現所有位元都只設定為 1 second-to-last位置。結果為 00000010 的二進位值，代表整數值 2。因此，BIT_AND 函數會傳回 2。

語法

```
BIT_AND ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須具有 INT、INT2或 INT8 資料類型。函數會傳回相等的 INT2、INT或 INT8 資料類型。

DISTINCT | ALL

使用引數時DISTINCT，函數會先消除指定運算式的所有重複值，再計算結果。使用引數 ALL，函數會保留所有重複的值。ALL 為預設值。如需詳細資訊，請參閱[DISTINCT 支援位元彙總](#)。

範例

由於整數欄中儲存有意義的商業資訊，您可以使用位元函數來擷取和彙總該資訊。下列查詢會將 BIT_AND 函數套用至名為 `userlikes` 的資料表中的 LIKES 資料欄，USERLIKES 並依資料 CITY 欄將結果分組。

```
select city, bit_and(likes) from userlikes group by city
order by city;
city          | bit_and
-----+-----
Los Angeles  |      0
Sacramento   |      0
San Francisco |      0
San Jose     |     64
Santa Barbara |    192
(5 rows)
```

您可將這些結果解釋如下：

- Santa Barbara 的整數值 192 轉換為二進位值 11000000。換言之，此城市的所有使用者都喜歡運動和戲劇，但並非所有使用者還喜歡其他任何一種活動。
- 整數 64 轉換為 01000000。因此，對於 San Jose 中的使用者而言，他們全部都喜歡的一種活動只有戲劇。
- 其他三個城市的值為 0，表示那些城市的所有使用者沒有共同的「愛好」。

BIT_OR 函數

BIT_OR 函數對單一整數資料欄或表達式中的所有值執行位元 OR 操作。此函數會彙總每一個二進位值 (對應於表達式中的每一個整數值) 的每一個位元。

例如，假設資料表有一欄包含四個整數值：3、7、10 及 22。這些整數以二進位格式表示如下：

Integer	二進位值
3	11
7	111
10	1010
22	10110

如果您將 BIT_OR 函數套用至整數值集，操作會尋找在每個位置1找到的任何值。在此案例中，至少一個值的最後五個位置有 1，於是產生二進位結果 00011111；因此，函數傳回 31 (亦即 16 + 8 + 4 + 2 + 1)。

語法

```
BIT_OR ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須具有 INT、INT2或 INT8資料類型。函數會傳回相等的 INT2、INT或 INT8 資料類型。

DISTINCT | ALL

使用引數 DISTINCT時，函數會先消除指定運算式的所有重複值，再計算結果。使用引數 ALL，函數會保留所有重複的值。ALL 為預設值。如需詳細資訊，請參閱[DISTINCT 支援位元彙總](#)。

範例

下列查詢會將 BIT_OR 函數套用至名為 的資料表中的資料LIKES欄，USERLIKES並依資料CITY欄將結果分組。

```
select city, bit_or(likes) from userlikes group by city
order by city;
city          | bit_or
-----+-----
Los Angeles  |    127
Sacramento   |    255
San Francisco |    255
San Jose     |    255
Santa Barbara |    255
(5 rows)
```

對於列出的其中四個城市，至少有一個使用者喜歡所有活動類型 (255=11111111)。以 Los Angeles 來說，至少有一個使用者喜歡所有活動類型，但運動除外 (127=01111111)。

BOOL_AND 函數

BOOL_AND 函數會在單一布林值或整數資料欄或運算式上運作。此函數會將類似的邏輯套用至 BIT_AND 和 BIT_OR 函數。此函數的傳回類型為布林值 (true 或 false)。

如果集合中的所有值都是 true，BOOL_AND 函數會傳回 true (t)。如果任何值為 false，此函數會傳回 false (f)。

語法

```
BOOL_AND ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須具有 BOOLEAN 或 整數資料類型。函數的傳回類型為 BOOLEAN。

DISTINCT | ALL

使用引數 DISTINCT 時，函數會先消除指定運算式的所有重複值，再計算結果。使用引數 ALL，函數會保留所有重複的值。ALL 為預設值。如需詳細資訊，請參閱 [DISTINCT 支援位元彙總](#)。

範例

您可以對布林值表達式或整數表達式使用布林值函數。例如，下列查詢傳回資料庫中標準 USERS 資料表的結果 TICKIT，其中包含數個布林值資料欄。

BOOL_AND 函數 false 會傳回所有五列。其中每個州並非所有使用者都喜歡運動。

```
select state, bool_and(likesports) from users
group by state order by state limit 5;
```

```
state | bool_and
-----+-----
AB    | f
AK    | f
AL    | f
AZ    | f
BC    | f
(5 rows)
```

BOOL_OR 函數

BOOL_OR 函數會在單一布林值或整數資料欄或運算式上運作。此函數會將類似的邏輯套用至 BIT_AND 和 BIT_OR 函數。此函數的傳回類型為布林值 (true、false 或 NULL)。

如果集合中的一或多個值為 true，BOOL_OR 函數會傳回 true (t)。如果集合中的所有值都是 false，函數會傳回 false (f)。NULL 如果值未知，則可以傳回。

語法

```
BOOL_OR ( [DISTINCT | ALL] expression )
```

引數

expression

函數運算的目標欄或表達式。此表達式必須具有 BOOLEAN 或 整數資料類型。函數的傳回類型為 BOOLEAN。

DISTINCT | ALL

使用引數 DISTINCT 時，函數會先消除指定運算式的所有重複值，再計算結果。使用引數 ALL，函數會保留所有重複的值。ALL 為預設值。請參閱 [DISTINCT 支援位元彙總](#)。

範例

您可以對布林值運算式或整數運算式使用布林值函數。例如，下列查詢傳回資料庫中標準 USERS 資料表的結果 TICKIT，其中包含數個布林值資料欄。

BOOL_OR 函數 true 會傳回所有五列。其中每個州至少有一個使用者喜歡運動。

```
select state, bool_or(likesports) from users
group by state order by state limit 5;
```

```
state | bool_or
-----+-----
AB    | t
AK    | t
AL    | t
AZ    | t
BC    | t
(5 rows)
```

下列範例傳回 NULL。

```
SELECT BOOL_OR(NULL = '123')
           bool_or
-----
NULL
```

條件式運算式

主題

- [CASE 條件式表達式](#)
- [DECODE 函數](#)
- [GREATEST 和 LEAST 函數](#)
- [NVL 和 COALESCE函數](#)
- [NVL2 函數](#)
- [NULLIF 函數](#)

Amazon Redshift 支援某些延伸到 SQL 標準的條件式表達式。

CASE 條件式表達式

CASE 表達式是條件式表達式，類似於以其他語言找到的 if/then/else 陳述式。CASE 用於在有多個條件時指定結果。在SQL運算式有效CASE時使用，例如在 SELECT 命令中。

CASE 表達式有兩種類型：簡單和已搜尋。

- 在簡單CASE表達式中，將表達式與值進行比較。找到相符項目時，會套用THEN子句中的指定動作。如果找不到相符項目，則會套用ELSE子句中的動作。
- 在搜尋的CASE表達式中，每個表達式CASE都會根據布林值表達式進行評估，且陳述CASE式會傳回第一個相符的 CASE。如果在WHEN子句中找不到相符項目，則會傳回ELSE子句中的動作。

語法

用於比對條件的簡單CASE陳述式：

```
CASE expression
  WHEN value THEN result
```



```
[WHEN...]  
[ELSE result]  
END
```

用於評估每個條件的搜尋CASE陳述式：

```
CASE  
  WHEN condition THEN result  
  [WHEN ...]  
  [ELSE result]  
END
```

引數

運算式

欄名或任何有效表達式。

值

與表達式相比較的值，例如數值常數或字元字串。

result

評估表達式或布林值條件時傳回的目標值或表達式。所有結果運算式的資料類型必須轉換為單個輸出類型。

condition

評估 true 或 false 的布林值運算式。如果條件為 true，則CASE表達式的值是遵循條件的結果，而且不會處理CASE表達式的其餘部分。如果條件為 false，則會評估任何後續WHEN子句。如果沒有WHEN條件結果為 true，則CASE表達式的值是ELSE子句的結果。如果省略ELSE子句，且沒有條件為 true，則結果為 null。

範例

下列範例使用範例TICKIT資料的VENUE資料表和SALES資料表。如需詳細資訊，請參閱[範本資料庫](#)。

使用簡單的CASE表達式，在針對VENUE資料表的查詢Big Apple中New York City取代。以 other 取代其他所有城市名稱。

```
select venuecity,
```

```

case venuecity
  when 'New York City'
    then 'Big Apple' else 'other'
  end
from venue
order by venueid desc;

```

venuecity	case
Los Angeles	other
New York City	Big Apple
San Francisco	other
Baltimore	other
...	

使用搜尋的CASE表達式，根據個別票證銷售PRICEPAID的值指派群組號碼：

```

select pricepaid,
  case when pricepaid <10000 then 'group 1'
        when pricepaid >10000 then 'group 2'
        else 'group 3'
  end
from sales
order by 1 desc;

```

pricepaid	case
12624	group 2
10000	group 3
10000	group 3
9996	group 1
9988	group 1
...	

DECODE 函數

DECODE 表達式會根據平等條件的結果，將特定值取代為另一個特定值或預設值。此操作等同於簡單CASE表達式或陳述式 IF-THEN-ELSE的操作。

語法

```
DECODE ( expression, search, result [, search, result ]... [ ,default ] )
```

這種表達式有助於將儲存於資料表中的縮寫或代碼，取代為報告所需的有意義商業值。

參數

運算式

您要比較之值的來源，例如資料表中的欄。

search

與來源表達式相比較的目標值，例如數值或字元字串。搜尋表達式必須評估為單一固定值。您不能指定會評估為一系列值的表達式，例如 `age between 20 and 29`；對於您要取代的每一個值，您需要指定個別的搜尋/結果對。

搜尋表達式的所有實例必須是相同或相容的資料類型。expression 和 search 參數也必須相容。

result

當表達式符合搜尋值時，查詢所傳回的替換值。您必須在DECODE運算式中包含至少一個搜尋/結果對。

結果表達式的所有實例必須是相同或相容的資料類型。result 和 default 參數也必須相容。

default

搜尋條件失敗時用於案例的選用預設值。如果您未指定預設值，表示DECODE式會傳回 NULL。

使用須知

如果表達式值和搜尋值都是 NULL，則DECODE結果是對應的結果值。如需這樣使用函數的相關說明，請參閱「範例」一節。

以此方式使用時，DECODE 類似於 [NVL2 函數](#)，但有一些差異。如需這些差異的說明，請參閱NVL2 用量備註。

範例

當 datetable 的 caldate 欄中存在值 2008-06-01 時，下列範例會以 June 1st, 2008 取代此值。此範例會以 取代所有其他的 caldate 值NULL。

```
select decode(caldate, '2008-06-01', 'June 1st, 2008')
from datetable where month='JUN' order by caldate;
```

```

case
-----
June 1st, 2008

...
(30 rows)

```

下列範例使用 DECODE 運算式，將 CATEGORY 資料表中的五個縮寫 CATNAME 資料欄轉換為全名，並將資料欄中的其他值轉換為 Unknown。

```

select catid, decode(catname,
'NHL', 'National Hockey League',
'MLB', 'Major League Baseball',
'MLS', 'Major League Soccer',
'NFL', 'National Football League',
'NBA', 'National Basketball Association',
'Unknown')
from category
order by catid;

```

```

catid | case
-----+-----
1     | Major League Baseball
2     | National Hockey League
3     | National Football League
4     | National Basketball Association
5     | Major League Soccer
6     | Unknown
7     | Unknown
8     | Unknown
9     | Unknown
10    | Unknown
11    | Unknown
(11 rows)

```

使用 DECODE 表達式尋找科羅拉多州和內華達州的 VENUESEATS 欄中具有 NULL 的場地；將 NULLs 轉換為零。如果資料 VENUESEATS 欄不是 NULL，則傳回 1 作為結果。

```

select venuename, venuestate, decode(venueSeats, null, 0, 1)
from venue
where venuestate in('NV', 'CO')
order by 2, 3, 1;

```

```

venuestate | case
-----+-----+
Coors Field | CO | 1
Dick's Sporting Goods Park | CO | 1
Ellie Caulkins Opera House | CO | 1
INVESCO Field | CO | 1
Pepsi Center | CO | 1
Ballys Hotel | NV | 0
Bellagio Hotel | NV | 0
Caesars Palace | NV | 0
Harrahs Hotel | NV | 0
Hilton Hotel | NV | 0
...
(20 rows)

```

GREATEST 和 LEAST 函數

從含有任何數量的表達式清單中傳回最大值或最小值。

語法

```

GREATEST (value [, ...])
LEAST (value [, ...])

```

參數

expression_list

逗號分隔的表達式清單，例如欄名。表達式必須全部可轉換為常見資料類型。NULL 系統會忽略清單中的值。如果所有表達式都評估為 NULL，則結果為 NULL。

傳回值

從提供的表達式清單中傳回最大（適用於 GREATEST）或最小（適用於 LEAST）值。

範例

下列範例按字母順序傳回 firstname 或 lastname 的最高值。

```
select firstname, lastname, greatest(firstname,lastname) from users
```

```
where userid < 10
order by 3;
```

```
  firstname | lastname | greatest
-----+-----+-----
Lars       | Ratliff  | Ratliff
Reagan    | Hodge   | Reagan
Colton    | Roy     | Roy
Barry     | Roy     | Roy
Tamekah   | Juarez  | Tamekah
Rafael    | Taylor  | Taylor
Victor    | Hernandez | Victor
Vladimir | Humphrey | Vladimir
Mufutau   | Watkins | Watkins
(9 rows)
```

NVL 和 COALESCE 函數

傳回一系列運算式中不為 null 的第一個運算式的值。找到非 Null 值時，就不會評估清單中剩餘的運算式。

NVL 與 COALESCE 相同。它們是同義詞。本主題說明語法，並包含兩者的範例。

語法

```
NVL( expression, expression, ... )
```

的語法 COALESCE 相同：

```
COALESCE( expression, expression, ... )
```

如果所有表達式都是 Null，則結果為 Null。

當您想要在主要值遺失或為 null 時傳回次要值，這些函數非常有用。例如，查詢可能會傳回三個可用電話號碼中的第一個：行動電話號碼、住家或公司。函數中運算式的順序決定評估的順序。

引數

運算式

要評估 Null 狀態的表達式，例如欄名。

傳回類型

Amazon Redshift 會根據輸入運算式判斷傳回值的資料類型。如果輸入運算式的資料類型沒有一般類型，則會傳回錯誤。

範例

如果清單包含整數運算式，該函數傳回一個整數。

```
SELECT COALESCE(NULL, 12, NULL);
```

```
coalesce  
-----  
12
```

此範例與上一個範例相同，除了使用 `NVL` 之外，會傳回相同的結果。

```
SELECT NVL(NULL, 12, NULL);
```

```
coalesce  
-----  
12
```

下列範例會傳回字串類型。

```
SELECT COALESCE(NULL, 'Amazon Redshift', NULL);
```

```
coalesce  
-----  
Amazon Redshift
```

下列範例會導致錯誤，因為運算式清單中的資料類型不同。在這種情況下，清單中同時存在字串類型和數字類型。

```
SELECT COALESCE(NULL, 'Amazon Redshift', 12);  
ERROR: invalid input syntax for integer: "Amazon Redshift"
```

在此範例中，您可以建立包含 `START_DATE` 和 `END_DATE` 資料欄的資料表、插入包含 `null` 值的資料列，然後將 `NVL` 運算式套用至兩個資料欄。

```
create table datetable (start_date date, end_date date);
```

```
insert into datetable values ('2008-06-01','2008-12-31');
insert into datetable values (null,'2008-12-31');
insert into datetable values ('2008-12-31',null);
```

```
select nvl(start_date, end_date)
from datetable
order by 1;
```

```
coalesce
-----
2008-06-01
2008-12-31
2008-12-31
```

NVL 運算式的預設資料欄名稱為 COALESCE。下列查詢會傳回相同的結果：

```
select coalesce(start_date, end_date)
from datetable
order by 1;
```

對於下列範例查詢，您可以建立包含飯店預訂範例資訊的表格，並插入數列。有些記錄包含 null 值。

```
create table booking_info (booking_id int, booking_code character(8), check_in date,
check_out date, funds_collected numeric(12,2));
```

插入下列範例資料。有些記錄沒有 check_out 日期或 funds_collected 金額。

```
insert into booking_info values (1, 'OCEAN_WV', '2023-02-01','2023-02-03',100.00);
insert into booking_info values (2, 'OCEAN_WV', '2023-04-22','2023-04-26',120.00);
insert into booking_info values (3, 'DSRT_SUN', '2023-03-13','2023-03-16',125.00);
insert into booking_info values (4, 'DSRT_SUN', '2023-06-01','2023-06-03',140.00);
insert into booking_info values (5, 'DSRT_SUN', '2023-07-10',null,null);
insert into booking_info values (6, 'OCEAN_WV', '2023-08-15',null,null);
```

下列查詢會傳回日期清單。如果 check_out 日期不可用，它會列出 check_in 日期。

```
select coalesce(check_out, check_in)
from booking_info
order by booking_id;
```


結果如下。請注意，最後兩則記錄顯示 `check_in` 日期。

```
coalesce
-----
2023-02-03
2023-04-26
2023-03-16
2023-06-03
2023-07-10
2023-08-15
```

如果您預期查詢會傳回特定函數或資料欄的 `null` 值，您可以使用 `NVL` 運算式將 `null` 取代為其他一些值。例如，彙總函數，例如 `SUM`，會在沒有要評估的資料列時傳回 `null` 值，而不是零。您可以使用 `NVL` 運算式，將這些 `null` 值取代為 `700.0`。加總 `funds_collected` 的結果不是 485 而是 1885，因為兩個有 `null` 的列被替換為 `700`。

```
select sum(nvl(funds_collected, 700.0)) as sumresult from booking_info;

sumresult
-----
1885
```

NVL2 函數

根據指定的表達式評估為 `NULL` 或 `NOT`，傳回兩個值之一 `NULL`。

語法

```
NVL2 ( expression, not_null_return_value, null_return_value )
```

引數

運算式

要評估 `Null` 狀態的表達式，例如欄名。

`not_null_return_value`

如果表達式評估為 `NOT`，則傳回的值 `NULL`。`not_null_return_value` 值的資料類型必須與 `expression` 相同，或可隱含地轉換為該資料類型。

null_return_value

如果表達式評估為 `NULL`，傳回的值 `NULL`。 `null_return_value` 值的資料類型必須與 `expression` 相同，或可隱含地轉換為該資料類型。

傳回類型

`NVL2` 傳回類型決定如下：

- 如果 `not_null_return_value` 或 `null_return_value` 為 `Null`，則傳回 `not-null` 表達式的資料類型。

如果 `not_null_return_value` 和 `null_return_value` 都不是 `Null`：

- 如果 `not_null_return_value` 和 `null_return_value` 有相同的資料類型，則傳回該資料類型。
- 如果 `not_null_return_value` 和 `null_return_value` 有不同的數值資料類型，則傳回最小可相容的數值資料類型。
- 如果 `not_null_return_value` 和 `null_return_value` 有不同的日期時間資料類型，則傳回時間戳記資料類型。
- 如果 `not_null_return_value` 和 `null_return_value` 有不同的字元資料類型，則傳回 `not_null_return_value` 的資料類型。
- 如果 `not_null_return_value` 和 `null_return_value` 有混合的數值和非數值資料類型，則傳回 `not_null_return_value` 的資料類型。

Important

前兩個案例中傳回 `not_null_return_value` 的資料類型，而 `null_return_value` 會隱含地轉換為該資料類型。如果資料類型不相容，函數會失敗。

使用須知

[DECODE 函數](#) 可以類似表達 `NVL2` 式和搜尋參數皆為 `null` 的方式使用。差別在於，對於 `DECODE`，傳回將同時具有結果參數的值和資料類型。相反地，對於 `NVL2`，傳回將具有 `not_null_return_value` 或 `null_return_value` 參數的值，以函數選取的值為準，但資料類型為 `not_null_return_value`。

例如，假設資料欄 1 為 `NULL`，下列查詢將傳回相同的值。不過，`DECODE` 傳回值資料類型將為 `INTEGER`，`NVL2` 傳回值資料類型將為 `VARCHAR`。

```
select decode(column1, null, 1234, '2345');
select nvl2(column1, '2345', 1234);
```

範例

下列範例修改一些範例資料，然後評估兩個欄位來提供使用者的適當聯絡資訊：

```
update users set email = null where firstname = 'Aphrodite' and lastname = 'Acevedo';
```

```
select (firstname + ' ' + lastname) as name,
nvl2(email, email, phone) AS contact_info
from users
where state = 'WA'
and lastname like 'A%'
order by lastname, firstname;
```

```
name          contact_info
-----+-----
Aphrodite Acevedo (906) 632-4407
Caldwell Acevedo Nunc.sollicitudin@Duisac.ca
Quinn Adams    vel@adipiscingligulaAenean.com
Kamal Aguilar  quis@vulputaterisusa.com
Samson Alexander hendrerit.neque@indolorFusce.ca
Hall Alford    ac.mattis@vitaediamProin.edu
Lane Allen     et.netus@risusDonec.org
Xander Allison ac.facilisis.facilisis@Infaucibus.com
Amaya Alvarado dui.nec.tempus@eudui.edu
Vera Alvarez   at.arcu.Vestibulum@pellentesque.edu
Yetta Anthony  enim.sit@risus.org
Violet Arnold  ad.litora@at.com
August Ashley  consectetuer.euismod@Phasellus.com
Karyn Austin   ipsum.primis.in@Maurisblanditenim.org
Lucas Ayers    at@elitpretiumet.com
```

NULLIF 函數

語法

NULLIF 運算式會比較兩個引數，如果引數相等，則傳回 null。如果不相等，則傳回第一個引數。此表達式是 NVL 或 COALESCE 表達式的反轉。

```
NULLIF ( expression1, expression2 )
```

引數

expression1、expression2

比較的目標欄或表達式。傳回類型與第一個表達式的類型相同。NULLIF 結果的預設資料欄名稱是第一個表達式的資料欄名稱。

範例

在下列範例中，查詢會傳回字串 `first`，因為引數不相等。

```
SELECT NULLIF('first', 'second');  
  
case  
-----  
first
```

在下列範例中，查詢會傳回 NULL，因為字串常值引數相等。

```
SELECT NULLIF('first', 'first');  
  
case  
-----  
NULL
```

在下列範例中，查詢會傳回 1，因為整數引數不相等。

```
SELECT NULLIF(1, 2);  
  
case  
-----  
1
```

在下列範例中，查詢會傳回 NULL，因為整數引數相等。

```
SELECT NULLIF(1, 1);  
  
case  
-----  
NULL
```

在下列範例中，當 LISTID 和 SALESID 值相符時，查詢會傳回 null：

```
select nullif(listid,salesid), salesid
from sales where salesid<10 order by 1, 2 desc;
```

listid	salesid
4	2
5	4
5	3
6	5
10	9
10	8
10	7
10	6
	1

(9 rows)

您可以使用 NULLIF 來確保空字串一律傳回為 null。在下面的範例中，NULLIF 表達式會傳回 null 值或包含至少一個字元的字串。

```
insert into category
values(0, '', 'Special', 'Special');

select nullif(catgroup, '') from category
where catdesc='Special';
```

catgroup
null

(1 row)

NULLIF 會忽略尾隨空白。如果字串不是空的，但包含空白，NULLIF 仍會傳回 null：

```
create table nulliftest(c1 char(2), c2 char(2));

insert into nulliftest values ('a','a ');

insert into nulliftest values ('b','b');

select nullif(c1,c2) from nulliftest;
```

```
c1
-----
null
null
(2 rows)
```

資料類型格式化函數

主題

- [CAST 函數](#)
- [CONVERT 函數](#)
- [TO_CHAR](#)
- [TO_DATE 函數](#)
- [收件人_NUMBER](#)
- [TEXT_TO_INT_ALT](#)
- [TEXT_TO_NUMERIC_ALT](#)
- [日期時間格式字串](#)
- [數值格式字串](#)
- [數值資料的 Teradata 樣式格式化字元](#)

資料類型格式化函數可讓您輕鬆轉換值的資料類型。範本對於其中每一個函數，第一個引數一律為要格式化的值，第二個引數包含新格式的範本。Amazon Redshift 支援多種資料類型格式化函數。

CAST 函數

CAST 函數會將一種資料類型轉換為另一種相容的資料類型。例如，您可以將字串轉換為日期，或將數字類型轉換為字串。CAST 會執行執行期轉換，這表示轉換不會變更來源資料表中的值資料類型。它僅在查詢的上下文中進行更改。

CAST 該函數與 [非常相似the section called “CONVERT”](#)，因為它們都會將一種資料類型轉換為另一種資料類型，但它們的命名不同。

某些資料類型需要使用 CAST 或 CONVERT 函數明確轉換為其他資料類型。其他資料類型可以隱含轉換，作為另一個命令的一部分，而無需使用 CAST 或 CONVERT。請參閱 [類型相容性與轉換](#)。

語法

使用兩種同等的語法格式中的任何一種，將運算式從一種資料類型轉換為另一種資料類型。

```
CAST ( expression AS type )  
expression :: type
```

引數

運算式

任何評估為一或多個值的表達式，例如欄名或常值。轉換 Null 值會傳回 Null。表達式不能包含空格或空字串。

type

其中一個支援的[資料類型](#)。

傳回類型

CAST 傳回類型引數指定的資料類型。

Note

如果您嘗試執行有問題的轉換，例如失去精確度的DECIMAL轉換，Amazon Redshift 會傳回錯誤，如下所示：

```
select 123.456::decimal(2,1);
```

或導致溢位的INTEGER轉換：

```
select 12345678::smallint;
```

範例

部分範例使用範例[TICKIT資料庫](#)。如需設定範例資料的詳細資訊，請參閱[載入資料](#)。

下列兩個查詢相同。都是將小數值轉換為整數：

```
select cast(pricepaid as integer)  
from sales where salesid=100;  
  
pricepaid
```

```
-----
162
(1 row)
```

```
select pricepaid::integer
from sales where salesid=100;
```

```
pricepaid
-----
162
(1 row)
```

以下產生類似的結果。它不需要執行範例資料：

```
select cast(162.00 as integer) as pricepaid;
```

```
pricepaid
-----
162
(1 row)
```

在此範例中，時間戳記資料欄中的值會轉換為日期，因此會從每個結果中移除時間：

```
select cast(saletime as date), salesid
from sales order by salesid limit 10;
```

```
 saletime | salesid
-----+-----
2008-02-18 |      1
2008-06-06 |      2
2008-06-06 |      3
2008-06-09 |      4
2008-08-31 |      5
2008-07-16 |      6
2008-06-26 |      7
2008-07-10 |      8
2008-07-22 |      9
2008-08-06 |     10
(10 rows)
```

如果您未CAST如上一個範例所示使用，則結果將包含時間：2008-02-18 02:36:48。

下列查詢會將可變字元資料轉換為日期。它不需要執行範例資料。

```
select cast('2008-02-18 02:36:48' as date) as mysaletime;
```

```
mysaletime
```

```
-----
```

```
2008-02-18
```

```
(1 row)
```

在此範例中，日期欄中的值轉換為時間戳記：

```
select cast(caldate as timestamp), dateid
from date order by dateid limit 10;
```

caldate	dateid
2008-01-01 00:00:00	1827
2008-01-02 00:00:00	1828
2008-01-03 00:00:00	1829
2008-01-04 00:00:00	1830
2008-01-05 00:00:00	1831
2008-01-06 00:00:00	1832
2008-01-07 00:00:00	1833
2008-01-08 00:00:00	1834
2008-01-09 00:00:00	1835
2008-01-10 00:00:00	1836

(10 rows)

在與上一個範例類似的案例中，您可以使用 [TO_CHAR](#) 取得對輸出格式的額外控制。

在此範例中，整數轉換為字元字串：

```
select cast(2008 as char(4));
```

```
bpchar
```

```
-----
```

```
2008
```

在此範例中，DECIMAL (6 , 3) 值會轉換為 DECIMAL (4 , 1) 值：

```
select cast(109.652 as decimal(4,1));
```

```
numeric
-----
109.7
```

此範例顯示更複雜的運算式。SALES 資料表中的資料PRICEPAID欄 (DECIMAL8 , 2) 資料欄) 會轉換為 DECIMAL (38 , 2) 資料欄 , 並將值乘以 100000000000000000000 :

```
select salesid, pricepaid::decimal(38,2)*100000000000000000000
as value from sales where salesid<10 order by salesid;
```

salesid	value
1	7280000000000000000000000000.00
2	7600000000000000000000000000.00
3	3500000000000000000000000000.00
4	1750000000000000000000000000.00
5	1540000000000000000000000000.00
6	3940000000000000000000000000.00
7	7880000000000000000000000000.00
8	1970000000000000000000000000.00
9	5910000000000000000000000000.00

(9 rows)

Note

您無法在GEOMETRY資料類型上執行 CAST或 CONVERT操作 , 以將其變更為其他資料類型。不過 , 您可以以延伸的已知二進位 (EWKB) 格式提供字串常值的十六進位表示法 , 作為接受GEOMETRY引數之函數的輸入。例如 , 以下 ST_AsText 函數預期 GEOMETRY 資料類型。

```
SELECT ST_AsText('01010000000000000000000000001C400000000000002040');
```

```
st_astext
-----
POINT(7 8)
```

您也可以明確指定 GEOMETRY 資料類型。

```
SELECT ST_AsText('010100000000000000000000000014400000000000001840'::geometry);
```

```
st_astext
-----
POINT(5 6)
```

CONVERT 函數

與[CAST函數](#)一樣，CONVERT函數會將一種資料類型轉換為另一種相容的資料類型。例如，您可以將字串轉換為日期，或將數字類型轉換為字串。CONVERT 會執行執行期轉換，這表示轉換不會變更來源資料表中的值資料類型。它僅在查詢的上下文中進行更改。

某些資料類型需要使用 CONVERT函數明確轉換為其他資料類型。其他資料類型可以隱含轉換，作為另一個命令的一部分，而無需使用 CAST或 CONVERT。請參閱[類型相容性與轉換](#)。

語法

```
CONVERT ( type, expression )
```

引數

type

其中一個支援的[資料類型](#)。

運算式

任何評估為一或多個值的表達式，例如欄名或常值。轉換 Null 值會傳回 Null。表達式不能包含空格或空字串。

傳回類型

CONVERT 傳回類型引數指定的資料類型。

Note

如果您嘗試執行有問題的轉換，例如失去精確度的DECIMAL轉換，Amazon Redshift 會傳回錯誤，如下所示：

```
SELECT CONVERT(decimal(2,1), 123.456);
```

或導致溢位的INTEGER轉換：

```
SELECT CONVERT(smallint, 12345678);
```

範例

部分範例使用範例[TICKIT資料庫](#)。如需設定範例資料的詳細資訊，請參閱[載入資料](#)。

下列查詢使用 CONVERT函數將一欄小數轉換為整數

```
SELECT CONVERT(integer, pricepaid)
FROM sales WHERE salesid=100;
```

這個範例會將整數轉換為字元字串。

```
SELECT CONVERT(char(4), 2008);
```

此範例會將目前日期和時間轉換為可變字元資料類型：

```
SELECT CONVERT(VARCHAR(30), GETDATE());
```

```
getdate
-----
2023-02-02 04:31:16
```

這個範例會將 saletime 欄轉換成僅限時間，並從每一列移除日期。

```
SELECT CONVERT(time, saletime), salesid
FROM sales order by salesid limit 10;
```

如需將時間戳記從某個時區轉換為另一個時區的詳細資訊，請參閱[CONVERT_TIMEZONE 函數](#)。如需其他日期和時間函數，請參閱[日期和時間函數](#)。

下列範例會將可變字元資料轉換成日期時間物件。

```
SELECT CONVERT(datetime, '2008-02-18 02:36:48') as mysaletime;
```

Note

您無法在 GEOMETRY 資料類型上執行 CAST 或 CONVERT 操作，以將其變更為其他資料類型。不過，您可以以延伸的已知二進位（EWKB）格式提供字串常值的十六進位表示法，作為接受 GEOMETRY 引數之函數的輸入。例如，以下 ST_AsText 函數預期 GEOMETRY 資料類型。

```
SELECT ST_AsText('01010000000000000000000001C40000000000002040');
```

```
st_astext
-----
POINT(7 8)
```

您也可以明確指定 GEOMETRY 資料類型。

```
SELECT ST_AsText('010100000000000000000000144000000000001840'::geometry);
```

```
st_astext
-----
POINT(5 6)
```

TO_CHAR

TO_CHAR 會將時間戳記或數值表達式轉換為字元字串資料格式。

語法

```
TO_CHAR (timestamp_expression | numeric_expression , 'format')
```

引數

timestamp_expression

產生 TIMESTAMP 或 TIMESTAMPTZ 類型值或可隱含強制時間戳記的值的表達式。

numeric_expression

此表達式產生數值資料類型的值，或可隱含地強制轉換為數值類型的值。如需詳細資訊，請參閱[數值類型](#)。TO_CHAR 會在數字字串的左側插入空格。

Note

TO_CHAR 不支援 128 位元DECIMAL值。

format

新值的格式。關於有效的格式，請參閱[日期時間格式字串](#)和[數值格式字串](#)。

傳回類型**VARCHAR****範例**

下列範例會將時間戳記轉換為日期和時間的值，其格式為月份名稱填滿九個字元、星期名稱以及月份的日期編號。

```
select to_char(timestamp '2009-12-31 23:15:59', 'MONTH-DY-DD-YYYY HH12:MIPM');
```

```
to_char
-----
DECEMBER -THU-31-2009 11:15PM
```

下列範例會將時間戳記轉換為具有年份天數的值。

```
select to_char(timestamp '2009-12-31 23:15:59', 'DDD');
```

```
to_char
-----
365
```

下列範例會將時間戳記轉換為一週的ISO天數。

```
select to_char(timestamp '2022-05-16 23:15:59', 'ID');
```

```
to_char
-----
1
```

以下範例會從日期擷取月份名稱。

```
select to_char(date '2009-12-31', 'MONTH');
```

```
to_char
```

```
-----  
DECEMBER
```

下列範例會將EVENT資料表中的每個STARTTIME值轉換為由小時、分鐘和秒組成的字串。

```
select to_char(starttime, 'HH12:MI:SS')  
from event where eventid between 1 and 5  
order by eventid;
```

```
to_char
```

```
-----  
02:30:00  
08:00:00  
02:30:00  
02:30:00  
07:00:00
```

下列範例將整個時間戳記值轉換成另一種格式。

```
select starttime, to_char(starttime, 'MON-DD-YYYY HH12:MIPM')  
from event where eventid=1;
```

```
      starttime      |      to_char  
-----+-----  
2008-01-25 14:30:00 | JAN-25-2008 02:30PM
```

下列範例將時間戳記常值轉換為字元字串。

```
select to_char(timestamp '2009-12-31 23:15:59', 'HH24:MI:SS');
```

```
to_char
```

```
-----  
23:15:59
```

下列範例會將小數轉換為字元字串。

```
select to_char(125.8, '999.99');
```

```
to_char
-----
125.80
```

下列範例會將小數轉換為字元字串。

```
select to_char(125.8, '999D99');

to_char
-----
125.80
```

下列範例會將數字轉換為開頭為零的字元字串。

```
select to_char(125.8, '0999D99');

to_char
-----
0125.80
```

下列範例將數字轉換為結尾帶負號的字元字串。

```
select to_char(-125.8, '999D99S');

to_char
-----
125.80-
```

下列範例會將數字轉換為在指定位置具有正號或負號的字元字串。

```
select to_char(125.8, '999D99SG');

to_char
-----
125.80+
```

下列範例會將數字轉換為在指定位置具有正符號的字元字串。

```
select to_char(125.8, 'PL999D99');
```



```
to_char
-----
+ 125.80
```

下列範例將數字轉換為帶有貨幣符號的字元字串。

```
select to_char(-125.88, '$S999D99');

to_char
-----
$-125.88
```

下列範例會將數字轉換為字元字串，貨幣符號位於指定位置。

```
select to_char(-125.88, 'S999D99L');

to_char
-----
-125.88$
```

下列範例使用千（逗號）分隔符號將數字轉換為字元字串。

```
select to_char(1125.8, '9,999.99');

to_char
-----
1,125.80
```

下列範例將數字轉換為字元字串，並使用角括號代表負數。

```
select to_char(-125.88, '$999D99PR');

to_char
-----
$<125.88>
```

下列範例將數字轉換為 Roman 數值字串。

```
select to_char(125, 'RN');
```

```
to_char
-----
CXXV
```

下列範例會將日期轉換為百年代碼。

```
select to_char(date '2020-12-31', 'CC');
```

```
to_char
-----
21
```

下列範例會顯示星期幾。

```
SELECT to_char(current_timestamp, 'FMDay, FMDD HH12:MI:SS');
```

```
to_char
-----
Wednesday, 31 09:34:26
```

下列範例顯示數字的序號字尾。

```
SELECT to_char(482, '999th');
```

```
to_char
-----
482nd
```

下列範例將 sales 資料表中的支付價格減去佣金。然後，將差額捨進並轉換為羅馬數字，並顯示於 to_char 欄：

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'rn') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;
```

salesid	pricepaid	commission	difference	to_char
1	728.00	109.20	618.80	dcxix
2	76.00	11.40	64.60	lxv
3	350.00	52.50	297.50	ccxcviii

4	175.00	26.25	148.75	cxlix
5	154.00	23.10	130.90	cxxxi
6	394.00	59.10	334.90	cccxxxv
7	788.00	118.20	669.80	dclxx
8	197.00	29.55	167.45	clxvii
9	591.00	88.65	502.35	dii
10	65.00	9.75	55.25	lv

下列範例將 to_char 欄所顯示的差額值加上貨幣符號：

```
select salesid, pricepaid, commission, (pricepaid - commission)
as difference, to_char(pricepaid - commission, 'l99999D99') from sales
group by sales.pricepaid, sales.commission, salesid
order by salesid limit 10;
```

salesid	pricepaid	commission	difference	to_char
1	728.00	109.20	618.80	\$ 618.80
2	76.00	11.40	64.60	\$ 64.60
3	350.00	52.50	297.50	\$ 297.50
4	175.00	26.25	148.75	\$ 148.75
5	154.00	23.10	130.90	\$ 130.90
6	394.00	59.10	334.90	\$ 334.90
7	788.00	118.20	669.80	\$ 669.80
8	197.00	29.55	167.45	\$ 167.45
9	591.00	88.65	502.35	\$ 502.35
10	65.00	9.75	55.25	\$ 55.25

下列範例列出每一筆銷售完成的世紀。

```
select salesid, saletime, to_char(saletime, 'cc') from sales
order by salesid limit 10;
```

salesid	saletime	to_char
1	2008-02-18 02:36:48	21
2	2008-06-06 05:00:16	21
3	2008-06-06 08:26:17	21
4	2008-06-09 08:38:52	21
5	2008-08-31 09:17:02	21
6	2008-07-16 11:59:24	21
7	2008-06-26 12:56:06	21
8	2008-07-10 02:12:36	21

```

9 | 2008-07-22 02:23:17 | 21
10 | 2008-08-06 02:51:55 | 21

```

下列範例會將EVENT資料表中的每個STARTTIME值轉換為由小時、分鐘、秒和時區組成的字串。

```

select to_char(starttime, 'HH12:MI:SS TZ')
from event where eventid between 1 and 5
order by eventid;

```

```

to_char
-----
02:30:00 UTC
08:00:00 UTC
02:30:00 UTC
02:30:00 UTC
07:00:00 UTC

```

下列範例顯示秒、毫秒和微秒的格式。

```

select sysdate,
to_char(sysdate, 'HH24:MI:SS') as seconds,
to_char(sysdate, 'HH24:MI:SS.MS') as milliseconds,
to_char(sysdate, 'HH24:MI:SS.US') as microseconds;

```

```

timestamp          | seconds | milliseconds | microseconds
-----+-----+-----+-----
2015-04-10 18:45:09 | 18:45:09 | 18:45:09.325 | 18:45:09:325143

```

TO_DATE 函數

TO_DATE 將字元字串表示的日期轉換為DATE資料類型。

Note

TO_DATE 不支援具有 Q (季度編號) 的格式字串。

語法

```
TO_DATE(string, format)
```

```
TO_DATE(string, format, is_strict)
```

引數

string

要轉換的字串。

format

字串常值，定義輸出 string 的日期部分格式。如需有效日、月和年格式的清單，請參閱[日期時間格式字串](#)。

is_strict

選用的布林值，指定如果輸入日期值超出範圍，是否會傳回錯誤。當 is_strict 設定為 TRUE 時，如果有超出範圍的值，就會傳回錯誤。當 is_strict 設定為 FALSE (預設值) 時，就會接受溢位值。

傳回類型

TO_DATE DATE會根據格式值傳回。

如果轉換成 format 失敗，則會傳回錯誤。

範例

下列SQL陳述式會將日期轉換為02 Oct 2001日期資料類型。

```
select to_date('02 Oct 2001', 'DD Mon YYYY');
```

```
to_date
-----
2001-10-02
(1 row)
```

下列SQL陳述式會將字串轉換為20010631日期。

```
select to_date('20010631', 'YYYYMMDD', FALSE);
```

結果是 2001 年 7 月 1 日，因為 6 月只有 30 天。

```
to_date
```

```
-----  
2001-07-01
```

下列SQL陳述式會將字串轉換為20010631日期：

```
to_date('20010631', 'YYYYMMDD', TRUE);
```

結果是錯誤，因為六月只有 30 天。

```
ERROR: date/time field date value out of range: 2001-6-31
```

收件人_NUMBER

TO_NUMBER 會將字串轉換為數值（十進位）值。

Note

建議您在格式字串FM中使用來抑制填充空白和零。如需有效格式的清單，請參閱 [數值格式字串](#)。

語法

```
to_number(string, format)
```

引數

string

要轉換的字串。格式必須是文字值。

format

第二個引數是格式字串，指出如何剖析字元字串來建立數值。例如，格式 'FM99D999' 指定要轉換的字串包含五位數，且第三個位置是小數點。例如，`to_number('12.345', 'FM99D999')` 會將以數值傳回 12.345。如需有效格式的清單，請參閱 [數值格式字串](#)。

傳回類型

TO_NUMBER 傳回DECIMAL數字。

如果轉換成 `format` 失敗，則會傳回錯誤。

範例

下列範例將字串 `12,454.8-` 轉換為數字：

```
select to_number('12,454.8-', 'FM99G999D9S');

to_number
-----
-12454.8
```

下列範例將字串 `$ 12,454.88` 轉換為數字：

```
select to_number('$ 12,454.88', 'FML99G999D99');

to_number
-----
12454.88
```

下列範例將字串 `$ 2,012,454.88` 轉換為數字：

```
select to_number('$ 2,012,454.88', 'FML9,999,999.99');

to_number
-----
2012454.88
```

TEXT_TO_INT_ALT

`TEXT_TO_INT_ALT` 使用 Teradata 格式格式將字元字串轉換為整數。結果中的分數數字會被截斷。

語法

```
TEXT_TO_INT_ALT (expression [ , 'format'])
```

引數

運算式

產生一或多個 `CHAR` 或 `VARCHAR` 值的表達式，例如資料欄名稱或常值字串。轉換 `Null` 值會傳回 `Null`。該函數將空白或空字串轉換為 `0`。

format

字串常值，定義輸入運算式的格式。如需您可以指定之格式化字元的相關資訊，請參閱[數值資料的 Teradata 樣式格式化字元](#)。

傳回類型

TEXT_TO_INT_ALT 傳回 INTEGER 值。

轉換結果的小數部分被截斷。

如果轉換成您指定的 format 詞語不成功，Amazon Redshift 就會傳回錯誤訊息。

範例

下列範例會將輸入 expression 字串 '123-' 轉換為整數 -123。

```
select text_to_int_alt('123-');
```

```
text_to_int_alt
-----
          -123
```

下列範例會將輸入 expression 字串 '2147483647+' 轉換成整數 2147483647。

```
select text_to_int_alt('2147483647+');
```

```
text_to_int_alt
-----
2147483647
```

下列範例會將指數輸入 expression 字串 '-123E-2' 轉換為整數 -1。

```
select text_to_int_alt('-123E-2');
```

```
text_to_int_alt
-----
          -1
```

下列範例會將輸入 expression 字串 '2147483647+' 轉換成整數 2147483647。


```
select text_to_int_alt('2147483647');
```

```
text_to_int_alt
-----
2147483647
```

以下範例會將 format 詞語 '999S' 的輸入 expression 字串 '123{' 轉換為整數 1230。S 字元表示帶符號的分區十進位值。如需詳細資訊，請參閱[數值資料的 Teradata 樣式格式化字元](#)。

```
select text_to_int_alt('123{', '999S');
```

```
text_to_int_alt
-----
      1230
```

下列範例會將格式片語為 'C9 (I) USD123' 的輸入表達式字串 " 轉換為整數 123。請參閱[數值資料的 Teradata 樣式格式化字元](#)。

```
select text_to_int_alt('USD123', 'C9(I)');
```

```
text_to_int_alt
-----
      123
```

下列範例會指定資料表欄做為輸入 expression。

```
select text_to_int_alt(a), text_to_int_alt(b) from t_text2int order by 1;
```

```
text_to_int_alt | text_to_int_alt
-----+-----
      -123 |          -123
      -123 |          -123
       123 |           123
       123 |           123
```

以下是此範例的表格定義和插入陳述式。

```
create table t_text2int (a varchar(200), b char(200));
```

```
insert into t_text2int VALUES('123', '123'),('123.123', '123.123'), ('-123', '-123'),  
('123-', '123-');
```

TEXT_TO_NUMERIC_ALT

TEXT_TO_NUMERIC_ALT 執行 Teradata 樣式的轉換操作，將字元字串轉換為數值資料格式。

語法

```
TEXT_TO_NUMERIC_ALT (expression [, 'format'] [, precision, scale])
```

引數

運算式

評估為一或多個CHAR或VARCHAR值的表達式，例如資料欄名稱或常值。轉換 Null 值會傳回 Null。空白或空字串會轉換為 0。

format

字串常值，定義輸入運算式的格式。如需詳細資訊，請參閱[數值資料的 Teradata 樣式格式化字元](#)。

precision

數字結果中的位數。預設值為 38。

scale

數值結果中小數點右邊的位數。預設值為 0。

傳回類型

TEXT_TO_NUMERIC_ALT 傳回DECIMAL數字。

如果轉換成您指定的 format 詞語不成功，Amazon Redshift 就會傳回錯誤訊息。

Amazon Redshift 會將輸入 expression 字串轉換為數值類型，並以您在精確度選項中為該類型指定的最高 precision。如果數值的長度超過您為 precision 指定的值，Amazon Redshift 會根據下列規則四捨五入數值：

- 如果轉換結果的長度超過您在 format 詞語中指定的長度，Amazon Redshift 會傳回錯誤訊息。

- 如果將結果轉換為數值，則結果會四捨五入為最接近的值。如果小數部分剛好位於上部和下部轉換結果的中間，則結果將四捨五入到最接近的偶數。

範例

下列範例會將輸入 expression 字串 '1.5' 轉換為數值 '2'。因為陳述式未指定 scale，所以 scale 預設為 0，而轉換結果不包含分數結果。由於 0.5 位於 1 和 2 之間，因此轉換結果將四捨五入為偶數值 2。

```
select text_to_numeric_alt('1.5');
```

```
text_to_numeric_alt
-----
                    2
```

下列範例會將輸入 expression 字串 '2.51' 轉換為數值 3。因為陳述式未指定 scale 值，所以 scale 預設為 0，而轉換結果不包含分數結果。由於 0.51 比 2 更接近 3，因此轉換結果將四捨五入為值 3。

```
select text_to_numeric_alt('2.51');
```

```
text_to_numeric_alt
-----
                    3
```

下列範例會將 precision 為 10 且 scale 為 2 的輸入 expression 字串 123.52501 轉換為數值 123.53。

```
select text_to_numeric_alt('123.52501', 10, 2);
```

```
text_to_numeric_alt
-----
                123.53
```

下列範例會將具有格式片語 '999S' 的輸入表達式字串 '123{' 轉換為數字 1230。S 字元表示帶符號的分區十進位值。如需詳細資訊，請參閱[數值資料的 Teradata 樣式格式化字元](#)。

```
select text_to_numeric_alt('123{', '999S');
```

```
text_to_int_alt
-----
```


日期部分或時間部分	意義
BC 或 B.C.、AD 或 A.D.、b.c. 或 bc、ad 或 a.d。	大寫和小寫的紀元標記
CC	兩位數世紀編號
YYYY、YYY、YY、Y	4 位數、3 位數、2 位數、1 位數的年編號
Y、YYY	含逗號的 4 位數年編號
IYYY、IYY、IY、I	4 位數、3 位數、2 位數、1 位數國際標準化組織 (ISO) 年編號
Q	季編號 (1 到 4)
MONTH、月、月	月名稱 (大寫、大小寫混合、小寫、以空格填補為 9 個字元)
MON、週一、週一	縮寫月名稱 (大寫、大小寫混合、小寫、以空格填補為 3 個字元)
MM	月編號 (01-12)
RM、rm	以羅馬數字表示的月數 (I–XII , 其中 I 是 1 月、大寫或小寫)
W	月的第幾週 (1-5 ; 第一週以月初第一天起算。)
WW	年的週編號 (1-53 ; 第一週以年初第一天起算。)
IW	ISO 週數 (新年的第一個星期四是第 1 週。)
DAY、日、日	日名稱 (大寫、大小寫混合、小寫、以空格填補為 9 個字元)
DY、Dy、dy	縮寫日名稱 (大寫、大小寫混合、小寫、以空格填補為 3 個字元)
DDD	年的第幾日 (001-366)

日期部分或時間部分	意義
IDDD	ISO 8601 週編號年的日 (001-371 ; 一年的第 1 天是第一-ISO週的星期一)
DD	月的第幾日 , 以數字表示 (01-31)
D	星期幾 (1-7 ; 星期日是 1)
	<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p> Note</p> <p>D 日期部分的行為與用於日期時間函數 DATE_PART 和 的星期 (DOW) 日期部分的行為不同EXTRACT。DOW 是以整數 0-6 為基礎 , 其中星期日为 0。如需詳細資訊 , 請參閱日期或時間戳記函數的日期部分。</p> </div>
ID	ISO 週一 (1) 至週日 (7) , 每週 8601 天
J	羅馬曆日 (自紀元前 4712 年 1 月 1 日起算的天數)
HH24	小時 (24 小時制 , 00-23)
HH 或 HH12	小時 (12 小時制 , 01-12)
MI	分鐘 (00-59)
SS	秒 (00-59)
MS	毫秒 (.000)
US	微秒 (.000000)
AM 或 PM、A.M. 或 P.M.、a.m. 或 p.m.、am 或 pm	大寫和小寫正午指標 (用於 12 小時制)
TZ、tz	大寫和小寫時區縮寫 ; TIMESTAMPTZ僅適用於

日期部分或時間部分	意義
OF	與 的偏移UTC ; TIMESTAMPTZ僅適用於

Note

您必須以單引號括住日期時間分隔符號 (例如 '-'、'/' 或 ':')，但必須以雙引號括住上表所列的「日期部分」和「時間部分」。

範例

如需將日期格式化為字串的範例，請參閱[TO_CHAR](#)。

數值格式字串

您可以在下面找到數字格式字串的參考。

下列格式字串適用於 TO_NUMBER 和 TO_ 等函數CHAR。

- 如需將字串格式化為數字的範例，請參閱[收件人_NUMBER](#)。
- 如需將數字格式化為字串的範例，請參閱[TO_CHAR](#)。

格式	描述
9	含指定位數的數值。
0	開頭為零的數值。
.(點)、D	小數點。
, (逗號)	千位分隔符號。
CC	世紀代碼。例如，21 世紀從 2001-01-01 開始 (僅支援 TO_CHAR)。
FM	填補模式。禁止填補空格和零。
PR	角括號中的負值。

格式	描述
S	緊貼於數字的正負號。
L	指定位置中的貨幣符號。
G	群組分隔符號。
MI	在小於 0 的數字中，位於指定位置的減號。
PL	在大於 0 的數字中，位於指定位置的加號。
SG	指定位置中的加號或減號。
RN	介於 1 到 3999 之間的羅馬數字（僅支援 TO_CHAR）。
TH 或 th	序號字尾。不轉換小於零的小數或值。

數值資料的 Teradata 樣式格式化字元

接下來，您可以找到 TEXT_TO_INT_ALT 和 TEXT_TO_NUMERIC_ALT 函數如何解譯輸入表達式字串中的字元。您還可以找到可在 format 詞語中指定的字元清單。此外，您還可以找到 Teradata 樣式格式化和 Amazon Redshift 格式選項之間差異的說明。

格式	描述
G	輸入 expression 字串中不支援做為群組分隔符號。您無法在 format 詞語中指定此字元。
D	<p>基數符號。您可以在 format 詞語中指定此字元。此字元相當於 .(句號)。</p> <p>基數符號不能出現在包含下列任何字元的 format 詞語中：</p> <ul style="list-style-type: none"> .(句點) S (大寫 's') V (大寫 'v')

格式	描述
/, : %	<p>插入字元 / (正斜線)、逗號 (,)、:(冒號) 和 % (百分比符號)。</p> <p>您不能在 format 詞語中包含這些字元。</p> <p>Amazon Redshift 會忽略輸入 expression 字串中的這些字元。</p>
.	<p>句點作為基底數字元，即小數點。</p> <p>此字元不能出現在以包含下列任何字元的 format 詞語中：</p> <ul style="list-style-type: none"> • D (大寫 'd') • S (大寫 's') • V (大寫 'v')
B	<p>您不能在 format 詞語中包含空格字元 (B)。在輸入 expression 字串中，開頭和結尾空格被忽略，並且不允許數字之間的空格。</p>
+ -	<p>您不能在 format 詞語中包含加號 (+) 或減號 (-)。但是，如果加號 (+) 和減號 (-) 出現在輸入 expression 字串中，則會隱含地剖析為數值的一部分。</p>
V	<p>小數點位置指示器。</p> <p>此字元不能出現在以包含下列任何字元的 format 詞語中：</p> <ul style="list-style-type: none"> • D (大寫 'd') • .(句點)

格式	描述
Z	抑制零的十進位數字。Amazon Redshift 修剪開頭零。Z 字元不能跟在 9 字元後面。如果小數部分包含 9 字元，則 Z 字元必須位於基數字元的左側。
9	小數位。
CHAR (n)	對於此格式，您可以指定下列選項： <ul style="list-style-type: none">CHAR 由 Z 或 9 個字元組成。Amazon Redshift 不支援CHAR值中的 + (加號) 或 - (減號)。n 是整數常數 I 或 F。對於 I，這是顯示數字或整數資料之整數部分所需的字元數。對於 F，這是顯示數值資料的小數部分所需的字元數。
-	連字號 (-) 字元。 您不能在 format 詞語中包含此字元。 Amazon Redshift 會忽略輸入 expression 字串中的這個字元。

格式	描述
S	<p>帶符號的分區十進位值。S 字元必須位於 format 詞語中最後一個十進位數字之後。輸入 expression 字串的最後一個字元和對應的數字轉換在 用於有符號區十進位、Teradata 樣式數值資料格式的資料格式字元 中列出。</p> <p>S 字元不能出現在包含下列任何字元的 format 詞語中：</p> <ul style="list-style-type: none"> • + (加號) • .(句點) • D (大寫 'd') • Z (大寫 'z') • F (大寫 'f') • E (大寫 'e')
E	<p>指數表示法。輸入 expression 字串可以包含指數字元。您不能在 format 詞語中指定 E 作為指數字元。</p>
FN9	Amazon Redshift 不支援。
FNE	Amazon Redshift 不支援。
\$、USD、美元	<p>美元符號 (\$)、ISO 貨幣符號 (USD) 和貨幣名稱美元。</p> <p>ISO 貨幣符號 USD 和貨幣名稱美元區分大小寫。Amazon Redshift 僅支援 USD 貨幣。輸入表達式字串可包含 USD 貨幣符號與數值之間的空格，例如「\$ 123E2」或「123E2 \$」。</p>
L	<p>貨幣符號。此貨幣符號字元只能在 format 詞語中出現一次。您無法指定重複的貨幣符號字元。</p>

格式	描述
C	ISO 貨幣符號。此貨幣符號字元只能在 format 詞語中出現一次。您無法指定重複的貨幣符號字元。
N	貨幣完整名稱。此貨幣符號字元只能在 format 詞語中出現一次。您無法指定重複的貨幣符號字元。
O	雙重貨幣符號。您無法在 format 詞語中指定此字元。
U	雙ISO貨幣符號。您無法在 format 詞語中指定此字元。
A	完整的雙重貨幣名稱。您無法在 format 詞語中指定此字元。

用於有符號區十進位、Teradata 樣式數值資料格式的資料格式字元

您可以在 TEXT_TOINT_ALT 和 TEXT_TO_NUMERIC_ALT 函數的格式片語中使用下列字元，作為已簽章的區域十進位值。

輸入字串的最後一個字元	數值轉換
{ 或 0	n ... 0
A 或 1	n ... 1
B 或 2	n ... 2
C 或 3	n ... 3
D 或 4	n ... 4
E 或 5	n ... 5
F 或 6	n ... 6

輸入字串的最後一個字元	數值轉換
G 或 7	n ... 7
H 或 8	n ... 8
I 或 9	n ... 9
}	-n ... 0
J	-n ... 1
K	-n ... 2
L	-n ... 3
M	-n ... 4
N	-n ... 5
O	-n ... 6
P	-n ... 7
Q	-n ... 8
R	-n ... 9

日期和時間函數

在此節中，您可以找到 Amazon Redshift 支援的日期和時間 scalar 函數之相關資訊。

主題

- [日期和時間函數的摘要](#)
- [交易中日期與時間函數](#)
- [已取代的僅限領導節點函數](#)
- [+ \(串連\) 運算子](#)
- [ADD_MONTHS 函數](#)
- [AT TIME ZONE 函數](#)

- [CONVERT_TIMEZONE 函數](#)
- [CURRENT_DATE 函數](#)
- [DATE_CMP 函數](#)
- [DATE_CMP_TIMESTAMP 函數](#)
- [DATE_CMP_TIMESTAMPPTZ 函數](#)
- [DATEADD 函數](#)
- [DATEDIFF 函數](#)
- [DATE_PART 函數](#)
- [DATE_PART_YEAR 函數](#)
- [DATE_TRUNC 函數](#)
- [EXTRACT 函數](#)
- [GETDATE 函數](#)
- [INTERVAL_CMP 函數](#)
- [LAST_DAY 函數](#)
- [MONTHS_BETWEEN 函數](#)
- [NEXT_DAY 函數](#)
- [SYSDATE 函數](#)
- [TIMEOFDAY 函數](#)
- [TIMESTAMP_CMP 函數](#)
- [TIMESTAMP_CMP_DATE 函數](#)
- [TIMESTAMP_CMP_TIMESTAMPPTZ 函數](#)
- [TIMESTAMPPTZ_CMP 函數](#)
- [TIMESTAMPPTZ_CMP_DATE 函數](#)
- [TIMESTAMPPTZ_CMP_TIMESTAMP 函數](#)
- [TIMEZONE 函數](#)
- [TO_TIMESTAMP 函數](#)
- [TRUNC 函數](#)
- [日期或時間戳記函數的日期部分](#)

日期和時間函數的摘要

函式	語法	傳回值
<p>+ (串連) 運算子</p> <p>將日期與 + 符號兩側的時間串連起來，並傳回 TIMESTAMP 或 TIMESTAMPTZ。</p>	date + time	TIMESTAMP 或 TIMESTAMP Z
<p>ADD_MONTHS</p> <p>將指定幾個月新增至日期或時間戳記。</p>	ADD_MONTHS ({date timestamp}, integer)	TIMESTAMP
<p>AT TIME ZONE</p> <p>指定要透過 TIMESTAMP 或 TIMESTAMPTZ 表達式來使用哪一個時區。</p>	AT TIME ZONE 'timezone'	TIMESTAMP 或 TIMESTAMP Z
<p>CONVERT_TIMEZONE</p> <p>可將時間戳記從一個時區轉換為另一個時區。</p>	CONVERT_TIMEZONE (['timezone',] 'timezone', timestamp)	TIMESTAMP
<p>CURRENT_DATE</p> <p>傳回目前工作階段時區中的日期 (預設為 UTC) 做為目前交易的開始。</p>	CURRENT_DATE	DATE
<p>DATE_CMP</p> <p>比較兩個日期並傳回 0 (如果日期是相同的)、1 (如果 date1 較大)，且 -1 (如果 date2 較大)。</p>	DATE_CMP (date1, date2)	INTEGER
<p>DATE_CMP_TIMESTAMP</p> <p>比較日期與時間並傳回 0 (如果值是相同的)、1 (如果 date 較大)，且 -1 (如果 timestamp 較大)。</p>	DATE_CMP_TIMESTAMP (date, timestamp)	INTEGER
<p>DATE_CMP_TIMESTAMPTZ</p>	DATE_CMP_TIMESTAMPTZ (date, timestamptz)	INTEGER

函式	語法	傳回值
<p>比較日期與含時區的時間戳記並傳回 0 (如果值是相同的)、1 (如果 date 較大), 且 -1 (如果 timestampz 較大)。</p> <p>DATE_PART_YEAR</p> <p>從日期擷取年份。</p>	DATE_PART_YEAR (date)	INTEGER
<p>DATEADD</p> <p>透過指定間隔來增量日期或時間。</p>	DATEADD (datepart, interval, {date time timetz timestamp})	TIMESTAMP、TIME 或 TIMETZ
<p>DATEDIFF</p> <p>傳回兩個日期或時間的差異，做為給定日期的部分 (例如日或月)。</p>	DATEDIFF (datepart, {date time timetz timestamp}, {date time timetz timestamp})	BIGINT
<p>DATE_PART</p> <p>從日期或時間擷取日期部分值。</p>	DATE_PART (datepart, {date timestamp})	DOUBLE
<p>DATE_TRUNC</p> <p>根據日期部分來截斷時間戳記。</p>	DATE_TRUNC ('datepart', timestamp)	TIMESTAMP
<p>EXTRACT</p> <p>從 timestamp、timestampz、time 或 timetz 中擷取日期或時間部分。</p>	EXTRACT (datepart FROM source)	INTEGER or DOUBLE
<p>GETDATE</p> <p>傳回目前工作階段時區中的目前日期和時間 (預設為 UTC)。括號是必要的。</p>	GETDATE()	TIMESTAMP

函式	語法	傳回值
<p>INTERVAL_CMP</p> <p>比較兩個間隔並傳回 0 (如果間隔是相同的)、1 (如果 interval1 較大), 且 -1 (如果 interval2 較大)。</p>	INTERVAL_CMP (interval1, interval2)	INTEGER
<p>LAST_DAY</p> <p>傳回某月最後一天的日期, 其中包含 date。</p>	LAST_DAY(date)	DATE
<p>MONTHS_BETWEEN</p> <p>傳回兩個日期之間有幾個月。</p>	MONTHS_BETWEEN (date, date)	FLOAT8
<p>NEXT_DAY</p> <p>傳回晚於 date 的第一個 day 執行個體的日期。</p>	NEXT_DAY (date, day)	DATE
<p>SYSDATE</p> <p>傳回日期和時間 (以 UTC 表示) 做為目前交易的開始。</p>	SYSDATE	TIMESTAMP
<p>TIMEOFDAY</p> <p>傳回目前工作階段時區中的目前工作日 (預設為 UTC) 做為字串值。</p>	TIMEOFDAY()	VARCHAR
<p>TIMESTAMP_CMP</p> <p>比較兩個時間戳記並傳回 0 (如果時間戳記是相同的)、1 (如果 timestamp1 較大), 且 -1 (如果 timestamp2 較大)。</p>	TIMESTAMP_CMP (timestamp1, timestamp2)	INTEGER
<p>TIMESTAMP_CMP_DATE</p> <p>比較時間戳記與日期並傳回 0 (如果值是相同的)、1 (如果 timestamp 較大), 且 -1 (如果 date 較大)。</p>	TIMESTAMP_CMP_DATE (timestamp, date)	INTEGER

函式	語法	傳回值
<p>TIMESTAMP_CMP_TIMESTAMPTZ</p> <p>比較時間戳記與含時區的時間戳記並傳回 0 (如果值是相同的)、1 (如果 timestamp 較大), 且 -1 (如果 timestamptz 較大)。</p>	<p>TIMESTAMP_CMP_TIME STAMPTZ (timestamp, timestamptz)</p>	INTEGER
<p>TIMESTAMPTZ_CMP</p> <p>比較兩個含時區的時間戳記並傳回 0 (如果值是相同的)、1 (如果 timestamptz1 較大), 且 -1 (如果 timestamptz2 較大)。</p>	<p>TIMESTAMPTZ_CMP (timestamptz1, timestamptz2)</p>	INTEGER
<p>TIMESTAMPTZ_CMP_DATE</p> <p>比較含時區的時間戳記值與日期並傳回 0 (如果值是相同的)、1 (如果 timestamptz 較大), 且 -1 (如果 date 較大)。</p>	<p>TIMESTAMPTZ_CMP_DATE (timestamptz, date)</p>	INTEGER
<p>TIMESTAMPTZ_CMP_TIMESTAMP</p> <p>比較含時區的時間戳記與時間戳記並傳回 0 (如果值是相同的)、1 (如果 timestamptz 較大), 且 -1 (如果 timestamp 較大)。</p>	<p>TIMESTAMPTZ_CMP_TIMESTAMP (timestamptz, timestamp)</p>	INTEGER
<p>TIMEZONE</p> <p>傳回時間戳記, 做為指定時區和時間戳記值。</p>	<p>TIMEZONE ('timezone' { timestamp timestamptz })</p>	TIMESTAMP 或 TIMESTAMP TZ
<p>TO_TIMESTAMP</p> <p>傳回含時區的時間戳記, 做為指定時間戳記和時區格式。</p>	<p>TO_TIMESTAMP ('timestamp', 'format')</p>	TIMESTAMP TZ
<p>TRUNC</p> <p>截斷時間戳記並傳回日期。</p>	<p>TRUNC(timestamp)</p>	DATE

Note

不會將閏秒視為歷經時間的計算中。

交易中日期與時間函數

當您在交易區塊 (BEGIN ... END) 中執行下列函數時，函數會傳回目前交易的開始時間，而不是目前陳述式的開始。

- SYSDATE
- TIMESTAMP
- CURRENT_DATE

下列函數一律會傳回目前陳述式的開始日期或時間 (即使他們在交易區塊中)。

- GETDATE
- TIMEOFDAY

已取代的僅限領導節點函數

下列日期函數已棄用，因為他們僅在領導節點上執行。如需詳細資訊，請參閱 [僅限領導節點函數](#)。

- AGE。請改用 [DATEDIFF 函數](#)。
- CURRENT_TIME。請改用 [GETDATE 函數](#) 或 [SYSDATE](#)。
- CURRENT_TIMESTAMP。請改用 [GETDATE 函數](#) 或 [SYSDATE](#)。
- LOCALTIME。請改用 [GETDATE 函數](#) 或 [SYSDATE](#)。
- LOCALTIMESTAMP。請改用 [GETDATE 函數](#) 或 [SYSDATE](#)。
- ISFINITE
- NOW。請改用 [GETDATE 函數](#) 或 [SYSDATE](#)。

+ (串連) 運算子

將 DATE 串連到 + 符號兩側的 TIME 或 TIMETZ，並傳回 TIMESTAMP 或 TIMESTAMPTZ。

語法

```
date + {time | timetz}
```

引數的順序可以反轉。例如，`time + date`。

引數

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。。

time

TIME 資料類型的欄，或是隱含評估為 TIME 類型的運算式。

timetz

TIMETZ 資料類型的欄，或是隱含評估為 TIMETZ 類型的運算式。

傳回類型

TIMESTAMP，如果輸入是 `date + time`。

TIMESTAMPTZ，如果輸入是 `date + timetz`。

範例

範例設定

若要設定範例中使用的 `TIME_TEST` 和 `TIMTZ_TEST` 資料表，請使用下列命令。

```
create table time_test(time_val time);

insert into time_test values
('20:00:00'),
('00:00:00.5550'),
('00:58:00');

create table timetz_test(timetz_val timetz);

insert into timetz_test values
('04:00:00+00'),
```

```
( '00:00:00.5550+00' ),
( '05:58:00+00' );
```

具有時間欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME) , 其中插入了三個值。

```
select time_val from time_test;
```

```
time_val
-----
20:00:00
00:00:00.5550
00:58:00
```

下列範例會串連日期常值和 TIME_VAL 欄。

```
select date '2000-01-02' + time_val as ts from time_test;
```

```
ts
-----
2000-01-02 20:00:00
2000-01-02 00:00:00.5550
2000-01-02 00:58:00
```

下列範例會串連日期常值和時間常值。

```
select date '2000-01-01' + time '20:00:00' as ts;
```

```
ts
-----
2000-01-01 20:00:00
```

下列範例會串連時間常值和日期常值。

```
select time '20:00:00' + date '2000-01-01' as ts;
```

```
ts
-----
2000-01-01 20:00:00
```

具有 TIMTZ 欄的範例

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ) , 其中插入了三個值。

```
select timetz_val from timetz_test;
```

```
timetz_val
-----
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會串連日期常值和 TIMETZ_VAL 欄。

```
select date '2000-01-01' + timetz_val as ts from timetz_test;
```

```
ts
-----
2000-01-01 04:00:00+00
2000-01-01 00:00:00.5550+00
2000-01-01 05:58:00+00
```

下列範例會串連 TIMETZ_VAL 欄和日期常值。

```
select timetz_val + date '2000-01-01' as ts from timetz_test;
```

```
ts
-----
2000-01-01 04:00:00+00
2000-01-01 00:00:00.5550+00
2000-01-01 05:58:00+00
```

下列範例會串連 DATE 常值和 TIMETZ 常值。此範例會傳回預設為 UTC 時區的 TIMESTAMPTZ。UTC 比 PST 早 8 小時 , 因此結果比輸入時間早 8 小時。

```
select date '2000-01-01' + timetz '20:00:00 PST' as ts;
```

```
ts
-----
2000-01-02 04:00:00+00
```

ADD_MONTHS 函數

ADD_MONTHS 會將指定幾個月新增至日期或時間戳記值或運算式。 [DATEADD](#) 函數提供類似功能。

語法

```
ADD_MONTHS( {date | timestamp}, integer)
```

引數

date | timestamp

DATE 或 TIMESTAMP 資料類型的欄，或是隱含評估為 DATE 或 TIMESTAMP 類型的運算式。如果日期是某個月的最後一天，或者如果產生的月份是較短的月份，則函數會在結果中傳回該月的最後一天。若是其他日期，結果會包含與日期表達式一樣의相同天數。

integer

INTEGER 資料類型的值。使用負數可減少日期中的月份。

傳回類型

TIMESTAMP

範例

以下查詢會使用 TRUNC 函數中的 ADD_MONTHS 函數。TRUNC 函數會從 ADD_MONTHS 結果移除某日時間。ADD_MONTHS 函數會從 CALDATE 欄中新增 12 個月至每個值。在 CALDATE 欄中的值是日期。

```
select distinct trunc(add_months(caldate, 12)) as calplus12,
trunc(caldate) as cal
from date
order by 1 asc;
```

```
calplus12 | cal
-----+-----
2009-01-01 | 2008-01-01
2009-01-02 | 2008-01-02
2009-01-03 | 2008-01-03
...
(365 rows)
```

下列範例會使用 ADD_MONTHS 函數，將 1 個月加到 timestamp。

```
select add_months('2008-01-01 05:07:30', 1);
```

```
add_months
-----
2008-02-01 05:07:30
```

下列範例示範當 ADD_MONTHS 函數對含有月份的日期進行操作的行為，而這些月份的天數皆不同時。此範例顯示函數如何處理在 3 月 31 日增加 1 個月，以及在 4 月 30 日增加 1 個月。4 月有 30 天，因此在 3 月 31 日的基礎上加上 1 個月就得到 4 月 30 日。5 月有 31 天，因此在 4 月 30 日的基礎上加上 1 個月就得到 5 月 31 日。

```
select add_months('2008-03-31',1);

add_months
-----
2008-04-30 00:00:00

select add_months('2008-04-30',1);

add_months
-----
2008-05-31 00:00:00
```

AT TIME ZONE 函數

AT TIME ZONE 指定要透過 TIMESTAMP 或 TIMESTAMPTZ 表達式來使用哪一個時區。

語法

```
AT TIME ZONE 'timezone'
```

引數

timezone

傳回值的 TIMEZONE。您可以將時區指定為時區名稱 (例如 **'Africa/Kampala'** 或 **'Singapore'**) 或做為時區縮寫 (例如 **'UTC'** 或 **'PDT'**)。

若要查看受支援時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```


若要查看受支援時區縮寫的清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

如需詳細資訊和範例，請參閱 [時區使用須知](#)。

傳回類型

與 `TIMESTAMP` 表達式搭配使用時的 `TIMESTAMPTZ`。與 `TIMESTAMPTZ` 表達式搭配使用時的 `TIMESTAMP`。

範例

下列範例會轉換不含時區的時間戳記值，並將其解譯為 MST 時間 (POSIX 中的 UTC+7)。此範例會傳回 UTC 時區的 `TIMESTAMPTZ` 資料類型值。如果您將預設時區設定為 UTC 以外的時區，可能會看到不同的結果。

```
SELECT TIMESTAMP '2001-02-16 20:38:40' AT TIME ZONE 'MST';
```

```
timezone
```

```
-----  
2001-02-17 03:38:40+00
```

下列範例會使用含時區值的輸入時間戳記，其中指定的時區是 EST (POSIX 中的 UTC+5) 且會將其轉換為 MST (POSIX 中的 UTC+7)。此範例會傳回 `TIMESTAMP` 資料類型的值。

```
SELECT TIMESTAMPTZ '2001-02-16 20:38:40-05' AT TIME ZONE 'MST';
```

```
timezone
```

```
-----  
2001-02-16 18:38:40
```

CONVERT_TIMEZONE 函數

`CONVERT_TIMEZONE` 可將時間戳記從一個時區轉換為另一個時區。此功能會根據日光節約時間自動調整。

語法

```
CONVERT_TIMEZONE( ['source_timezone',] 'target_timezone', 'timestamp')
```

引數

source_timezone

(選用) 目前時間戳記的時區。預設值為 UTC。如需詳細資訊，請參閱 [時區使用須知](#)。

target_timezone

新時間戳記的時區。如需詳細資訊，請參閱 [時區使用須知](#)。

timestamp

時間戳記欄或運算式會隱性轉換為時間戳記。

傳回類型

TIMESTAMP

時區使用須知

來源時區或 target_ 時區/時區可以指定為時區名稱 (例如「非洲/坎帕拉」或「新加坡」) ，也可以指定為時區縮寫 (例如「世界標準時間」或「PDT」) 。您不必將時區名稱轉換為名稱或縮寫轉換為縮寫。例如，您可以從來源時區名稱「Singapore」中選擇時間戳記，並將其轉換為時區縮寫「PDT」的時間戳記。

Note

使用時區名稱或時區縮寫的結果可能會因當地季節時間 (例如夏令時間) 而有所不同。

使用時區名稱

若要檢視目前和完整的時區名稱清單，請執行下列命令。

```
select pg_timezone_names();
```

每一個資料列都包含一個逗號分隔的字串，其中包含時區名稱、縮寫、UTC 時差，以及時區是否會遵循日光節約時間 (t 或 f) 的指標。以下列程式碼片段為例，其將顯示兩個產生的資料列。第一行是時區 Europe/Paris ，縮寫 CET ，與 UTC 的 01:00:00 偏移量，並 f 表示它不會觀察日光節約時間。第二行是時區 Israel ，縮寫 IST ，與 UTC 02:00:00 偏移量，並 f 表示它不會觀察日光節約時間。

```
pg_timezone_names
-----
(Europe/Paris,CET,01:00:00,f)
(Israel,IST,02:00:00,f)
```

執行 SQL 陳述式以取得整個清單並尋找時區名稱。大約會傳回 600 個資料列。雖然部分傳回的時區名稱為大寫首字母或縮寫 (例如, GB、PRC、ROK), CONVERT_TIMEZONE 功能仍會將其視為時區名稱, 而非時區縮寫。

如果您使用時區名稱指定時區, CONVERT_TIMZONE 會自動調整夏令時 (DST) 或任何其他本地季節性協定, 例如夏令時間、標準時間或冬令時間, 這些時區在「timestamp」指定的日期和時間內生效。例如, 'Europe/London' 在冬天代表 UTC, 在夏天增加一小時。

使用時區縮寫

若要檢視目前和完整的時區縮寫清單, 請執行下列命令。

```
select pg_timezone_abbrevs();
```

結果都會包含一個逗號分隔的字串, 其中包含時區縮寫、UTC 時差, 以及時區是否會遵循日光節約時間的指標 (t 或 f)。以下列程式碼片段為例, 其將顯示兩個產生的資料列。第一列包含太平洋夏令時間的縮寫 (PDT), 與 UTC 的時差為 -07:00:00, 以及表示其遵循日光節約時間的 t。第二列包含太平洋標準時間的縮寫 PST, 與 UTC 的 -08:00:00 偏移量, 並表示它不會觀察 f 到節省日光的時間。

```
pg_timezone_abbrevs
-----
(PDT,-07:00:00,t)
(PST,-08:00:00,f)
```

執行 SQL 陳述式以取得整個清單, 並根據其時差和日光節約時間指標來尋找縮寫。大約會傳回 200 個資料列。

時區縮寫代表與 UTC 的固定偏差。如果您使用時區縮寫指定時區, CONVERT_TIMZONE 會使用與 UTC 相比的固定偏移量, 並且不會針對任何本地季節性通訊協定進行調整。

使用 POSIX 樣式格式

POSIX 樣式時區規格的格式是 STDoffset 或 STDoffsetDST, 其中 STD 是時區縮寫, offset 是 UTC 以西的小時偏移量, DST 是選用的日光節約時區縮寫。日光節約時間假設為較給定偏移早一小時。

POSIX 樣式時區格式使用格林威治以西的正偏移，與 ISO-8601 慣例不同，此是使用格林威治以東的正偏移。

以下是 POSIX 樣式時區的範例：

- PST8
- PST8PDT
- EST5
- EST5EDT

Note

Amazon Redshift 不會驗證 POSIX 樣式時區規格，所以可能會將時區設為無效值。例如，即使將時區設為無效值，下列命令不會傳回錯誤。

```
set timezone to 'xxx36';
```

範例

許多範例都使用 TICKIT 範例資料集。如需詳細資訊，請參閱[範本資料庫](#)。

以下範例會將時間戳記值從預設 UTC 時區轉換為 PST。

```
select convert_timezone('PST', '2008-08-21 07:23:54');

convert_timezone
-----
2008-08-20 23:23:54
```

以下範例會將 LISTTIME 欄位中的時間戳記值從預設 UTC 時區轉換為 PST。即使時間戳記是在日光節約時間期間，其會轉換為標準時間，因為目標時區是指定為縮寫 (PST)。

```
select listtime, convert_timezone('PST', listtime) from listing
where listid = 16;

listtime      | convert_timezone
-----+-----
```

```
2008-08-24 09:36:12      2008-08-24 01:36:12
```

以下範例會將時間戳記 LISTTIME 欄位從預設 UTC 時區轉換為 US/Pacific 時區。目標時區使用時區名稱，且時間戳記是在日光節約時間期間，因此函數會傳回日光時間。

```
select listtime, convert_timezone('US/Pacific', listtime) from listing
where listid = 16;
```

```
listtime      | convert_timezone
-----+-----
2008-08-24 09:36:12 | 2008-08-24 02:36:12
```

以下範例會將時間戳記字串從 EST 轉換為 PST：

```
select convert_timezone('EST', 'PST', '20080305 12:25:29');
```

```
convert_timezone
-----
2008-03-05 09:25:29
```

以下範例會將時間戳記轉換為美國東部標準時間，因為目標時區使用時區名稱 (America/New_York) 且時間戳記是在標準時間期間。

```
select convert_timezone('America/New_York', '2013-02-01 08:00:00');
```

```
convert_timezone
-----
2013-02-01 03:00:00
(1 row)
```

以下範例會將時間戳記轉換為美國東部日光時間，因為目標時區使用時區名稱 (America/New_York) 且時間戳記是在日光時間期間。

```
select convert_timezone('America/New_York', '2013-06-01 08:00:00');
```

```
convert_timezone
-----
2013-06-01 04:00:00
(1 row)
```

以下範例示範的是偏移的使用。

```
SELECT CONVERT_TIMEZONE('GMT', 'NEWZONE +2', '2014-05-17 12:00:00') as newzone_plus_2,
CONVERT_TIMEZONE('GMT', 'NEWZONE-2:15', '2014-05-17 12:00:00') as newzone_minus_2_15,
CONVERT_TIMEZONE('GMT', 'America/Los_Angeles+2', '2014-05-17 12:00:00') as la_plus_2,
CONVERT_TIMEZONE('GMT', 'GMT+2', '2014-05-17 12:00:00') as gmt_plus_2;
```

newzone_plus_2	newzone_minus_2_15	la_plus_2	gmt_plus_2
2014-05-17 10:00:00	2014-05-17 14:15:00	2014-05-17 10:00:00	2014-05-17 10:00:00

(1 row)

CURRENT_DATE 函數

CURRENT_DATE 傳回目前工作階段時區中的日期 (預設為 UTC)，使用預設格式：YYYY-MM-DD。

Note

CURRENT_DATE 傳回目前交易的日期 (而不是目前陳述式的開始)。考慮以下場景：您在 10/01/08 23:59 啟動包含多個陳述式的交易，而包含 CURRENT_DATE 的陳述式在 10/02/08 00:00 執行。CURRENT_DATE 傳回 10/01/08，而不是 10/02/08。

語法

```
CURRENT_DATE
```

傳回類型

DATE

範例

下列範例會傳回目前的日期 (函式執行的 AWS 區域 位置)。

```
select current_date;
```

date
2008-10-01

下列範例會建立資料表，插入 todays_date 欄預設值為 CURRENT_DATE 的列，然後選取資料表中的所有列。

```
CREATE TABLE insert_dates(  
    label varchar(128) NOT NULL,  
    todays_date DATE DEFAULT CURRENT_DATE);
```

```
INSERT INTO insert_dates(label)  
VALUES('Date row inserted');
```

```
SELECT * FROM insert_dates;
```

```
label          | todays_date  
-----+-----  
Date row inserted | 2023-05-10
```

DATE_CMP 函數

DATE_CMP 會比較兩個日期。該函數會傳回 0 (如果日期是相同的)、1 (如果 date1 較大)，且 -1 (如果 date2 較大)。

語法

```
DATE_CMP(date1, date2)
```

引數

date1

DATE 資料類型的欄，或是評估為 DATE 類型的運算式。

date2

DATE 資料類型的欄，或是評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

下列查詢會比較 CALDATE 欄位的 DATE 值與日期 2008 年 1 月 4 日，並傳回 CALDATE 中的值是早於 (-1)、等於 (0) 或晚於 (1) 2008 年 1 月 4 日：

```
select caldate, '2008-01-04',
date_cmp(caldate, '2008-01-04')
from date
order by dateid
limit 10;
```

caldate	?column?	date_cmp
2008-01-01	2008-01-04	-1
2008-01-02	2008-01-04	-1
2008-01-03	2008-01-04	-1
2008-01-04	2008-01-04	0
2008-01-05	2008-01-04	1
2008-01-06	2008-01-04	1
2008-01-07	2008-01-04	1
2008-01-08	2008-01-04	1
2008-01-09	2008-01-04	1
2008-01-10	2008-01-04	1

(10 rows)

DATE_CMP_TIMESTAMP 函數

DATE_CMP_TIMESTAMP 會比較日期與時間戳記並傳回 0 (如果值是相同的)、1 (如果 date 較大)，以及 -1 (如果 timestamp 較大)。

語法

```
DATE_CMP_TIMESTAMP(date, timestamp)
```

引數

date

DATE 資料類型的欄，或是評估為 DATE 類型的運算式。

timestamp

TIMESTAMP 資料類型的欄，或是評估為 TIMESTAMP 類型的運算式。

傳回類型

INTEGER

範例

例如，下列範例會比較日期 2008-06-18 與 LISTTIME。LISTTIME 欄的值是時間戳記。在此日期前所做的清單會傳回 1，在此日期後所做的清單會傳回 -1。

```
select listid, '2008-06-18', listtime,
       date_cmp_timestamp('2008-06-18', listtime)
from listing
order by 1, 2, 3, 4
limit 10;
```

listid	?column?	listtime	date_cmp_timestamp
1	2008-06-18	2008-01-24 06:43:29	1
2	2008-06-18	2008-03-05 12:25:29	1
3	2008-06-18	2008-11-01 07:35:33	-1
4	2008-06-18	2008-05-24 01:18:37	1
5	2008-06-18	2008-05-17 02:29:11	1
6	2008-06-18	2008-08-15 02:08:13	-1
7	2008-06-18	2008-11-15 09:38:15	-1
8	2008-06-18	2008-11-09 05:07:30	-1
9	2008-06-18	2008-09-09 08:03:36	-1
10	2008-06-18	2008-06-17 09:44:54	1

(10 rows)

DATE_CMP_TIMESTAMPTZ 函數

DATE_CMP_TIMESTAMPTZ 會比較日期與含時區的時間戳記，並傳回 0 (如果值是相同的)、1 (如果 date 較大)，以及 -1 (如果 timestamptz 較大)。

語法

```
DATE_CMP_TIMESTAMPTZ(date, timestamptz)
```

引數

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。。

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

傳回類型

INTEGER

範例

例如，下列範例會比較日期 2008-06-18 與 LISTTIME。在此日期前所做的清單會傳回 1，在此日期後所做的清單會傳回 -1。

```
select listid, '2008-06-18', CAST(listtime AS timestamptz),
date_cmp_timestamptz('2008-06-18', CAST(listtime AS timestamptz))
from listing
order by 1, 2, 3, 4
limit 10;
```

listid	?column?	timestamptz	date_cmp_timestamptz
1	2008-06-18	2008-01-24 06:43:29+00	1
2	2008-06-18	2008-03-05 12:25:29+00	1
3	2008-06-18	2008-11-01 07:35:33+00	-1
4	2008-06-18	2008-05-24 01:18:37+00	1
5	2008-06-18	2008-05-17 02:29:11+00	1
6	2008-06-18	2008-08-15 02:08:13+00	-1
7	2008-06-18	2008-11-15 09:38:15+00	-1
8	2008-06-18	2008-11-09 05:07:30+00	-1
9	2008-06-18	2008-09-09 08:03:36+00	-1
10	2008-06-18	2008-06-17 09:44:54+00	1

(10 rows)

DATEADD 函數

透過指定間隔來增量 DATE、TIME、TIMETZ 或 TIMESTAMP 值。

語法

```
DATEADD( datepart, interval, {date|time|timetz|timestamp} )
```

引數

datepart

函數對其執行的日期部分 (例如，年、月、日或小時)。如需詳細資訊，請參閱 [日期或時間戳記函數的日期部分](#)。

間隔

指定要新增至目標表達式之間隔 (例如, 天數) 的整數。負整數會減去間隔。

date|time|timetz|timestamp

DATE、TIME、TIMETZ 或 TIMESTAMP 欄或隱含轉換為 DATE、TIME、TIMETZ 或 TIMESTAMP 的運算式。DATE、TIME、TIMETZ 或 TIMESTAMP 運算式必須包含指定的日期部分。

傳回類型

TIMESTAMP 或 TIME 或 TIMETZ 取決於輸入資料類型。

具有 DATE 欄的範例

下列範例會向 DATE 資料表中存在的 11 月份的每個日期增加 30 天。

```
select dateadd(day,30,caldate) as novplus30
from date
where month='NOV'
order by dateid;

novplus30
-----
2008-12-01 00:00:00
2008-12-02 00:00:00
2008-12-03 00:00:00
...
(30 rows)
```

下列範例會將 18 個月增加至常值日期值。

```
select dateadd(month,18,'2008-02-28');

date_add
-----
2009-08-28 00:00:00
(1 row)
```

DATEADD 函數的預設欄名稱為 DATE_ADD。日期值的預設時間戳記為 00:00:00。

下列範例會將 30 分鐘增加至未指定時間戳記的日期值。

```
select dateadd(m,30,'2008-02-28');

date_add
-----
2008-02-28 00:30:00
(1 row)
```

您可以用全名或縮寫來表示日期部分。在這種情況下，m 代表分鐘，而不是月。

具有 TIME 欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME)，其中插入了三個值。

```
select time_val from time_test;

time_val
-----
20:00:00
00:00:00.5550
00:58:00
```

下列範例會將 5 分鐘增加至 TIME_TEST 資料表中的每個 TIME_VAL。

```
select dateadd(minute,5,time_val) as minplus5 from time_test;

minplus5
-----
20:05:00
00:05:00.5550
01:03:00
```

下列範例會將 8 小時增加至常值時間值。

```
select dateadd(hour, 8, time '13:24:55');

date_add
-----
21:24:55
```

下列範例會顯示時間超過 24:00:00 或在 00:00:00 以下的時間。

```
select dateadd(hour, 12, time '13:24:55');

date_add
-----
01:24:55
```

具有 TIMTZ 欄的範例

這些範例中的輸出值是以 UTC 為預設時區。

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ) , 其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
-----
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會將 5 分鐘增加至 TIMETZ_TEST 資料表中的每個 TIMETZ_VAL。

```
select dateadd(minute,5,timetz_val) as minplus5_tz from timetz_test;

minplus5_tz
-----
04:05:00+00
00:05:00.5550+00
06:03:00+00
```

下列範例會將 2 小時加入常值 timetz 值。

```
select dateadd(hour, 2, timetz '13:24:55 PST');

date_add
-----
23:24:55+00
```

具有 TIMESTAMP 欄的範例

這些範例中的輸出值是以 UTC 為預設時區。

下列範例資料表 `TIMESTAMP_TEST` 有一個 `TIMESTAMP_VAL` 欄 (類型為 `TIMESTAMP`)，其中插入了三個值。

```
SELECT timestamp_val FROM timestamp_test;
```

```
timestamp_val
-----
1988-05-15 10:23:31
2021-03-18 17:20:41
2023-06-02 18:11:12
```

以下範例僅將 20 年增加至 2000 年之前的 `TIMESTAMP_TEST` 中的 `TIMESTAMP_VAL` 值。

```
SELECT dateadd(year,20,timestamp_val)
FROM timestamp_test
WHERE timestamp_val < to_timestamp('2000-01-01 00:00:00', 'YYYY-MM-DD HH:MI:SS');
```

```
date_add
-----
2008-05-15 10:23:31
```

下列範例將 5 秒加入不帶秒指示器的常值時間戳記值。

```
SELECT dateadd(second, 5, timestamp '2001-06-06');
```

```
date_add
-----
2001-06-06 00:00:05
```

使用須知

`DATEADD(month, ...)` 和 `ADD_MONTHS` 函數處理落在月底之日期的方式會所有不同：

- `ADD_MONTHS`：如果您要新增的日期是某個月的最後一天，則產生的月份總是產生月份的最後一天，不論該月份的長短。例如，4 月 30 號 + 1 個月是 5 月 31 號。

```
select add_months('2008-04-30',1);
```

```
add_months
-----
2008-05-31 00:00:00
```

```
(1 row)
```

- **DATEADD**：如果您要新增之日期中的天數少於結果月份，則結果將會是結果月份的相對應天數，而不會是該月的最後一天。例如，4 月 30 號 + 1 個月是 5 月 30 號。

```
select dateadd(month,1,'2008-04-30');

date_add
-----
2008-05-30 00:00:00
(1 row)
```

DATEADD 函數使用 `dateadd (月份, 12...)` 或 `dateadd (年份, 1...)` 時，處理閏年日期 02-29 的方式會不同。

```
select dateadd(month,12,'2016-02-29');

date_add
-----
2017-02-28 00:00:00

select dateadd(year, 1, '2016-02-29');

date_add
-----
2017-03-01 00:00:00
```

DATEDIFF 函數

DATEDIFF 傳回兩個日期或時間表達式之日期部分的差異。

語法

```
DATEDIFF( datepart, {date|time|timetz|timestamp}, {date|time|timetz|timestamp} )
```

引數

datepart

函數運作的日期或時間值 (年、月或日、小時、分鐘、秒、毫秒或微秒) 的特定部分。如需詳細資訊，請參閱 [日期或時間戳記函數的日期部分](#)。

特別是，DATEDIFF 會決定兩個運算式間相距的日期部分邊界數。例如，假設您正在計算兩個日期 12-31-2008 和 01-01-2009 之間的年度差異。在這種情況下，函數傳回 1 年，儘管這些日期只相隔一天。如果您正在尋找兩個時間戳記間時數的差異，01-01-2009 8:30:00 和 01-01-2009 10:00:00，結果為 2 小時。如果您正在尋找兩個時間戳記間時數的差異，8:30:00 和 10:00:00，結果為 2 小時。

date|time|timetz|timestamp

DATE、TIME、TIMETZ 或 TIMESTAMP 欄，或隱含轉換為 DATE、TIME、TIMETZ 或 TIMESTAMP 的運算式。運算式必須同時包含指定的日期部分或時間部分。如果第二個日期或時間晚於第一個日期或時間，結果為正值。如果第二個日期或時間早於第一個日期或時間，結果為負值。

傳回類型

BIGINT

具有 DATE 欄的範例

下列範例會尋找在兩個常值日期值間週次的差異。

```
select datediff(week, '2009-01-01', '2009-12-31') as numweeks;

numweeks
-----
52
(1 row)
```

下列範例會尋找兩個常值日期值之間的差異 (以小時為單位)。如果您未提供日期的時間值，則預設值為 00:00:00。

```
select datediff(hour, '2023-01-01', '2023-01-03 05:04:03');

date_diff
-----
53
(1 row)
```

下列範例會尋找兩個常值 TIMESTAMETZ 值之間的差異 (以天為單位)。

```
Select datediff(days, 'Jun 1,2008 09:59:59 EST', 'Jul 4,2008 09:59:59 EST')
```



```
date_diff
-----
33
```

以下範例找出資料表中同一列兩個日期之間的差異 (以天為單位)。

```
select * from date_table;

start_date | end_date
-----+-----
2009-01-01 | 2009-03-23
2023-01-04 | 2024-05-04
(2 rows)

select datediff(day, start_date, end_date) as duration from date_table;

duration
-----
      81
     486
(2 rows)
```

下列範例會尋找在過去和今日日期中常值間季次的差異。此範例假設目前日期為 2008 年 6 月 5 日。您可以用全名或縮寫來表示日期部分。DATEDIFF 函數的預設欄名稱為 DATE_DIFF。

```
select datediff(qtr, '1998-07-01', current_date);

date_diff
-----
40
(1 row)
```

下列範例會聯結 SALES 和 LISTING 資歷表，來計算在他們列出和門票售出後所經的天數：清單 1000 到 1005。這些清單銷售的最長等待時間是 15 天，而最短時間是不到一天 (0 天)。

```
select priceperticket,
datediff(day, listtime, saletime) as wait
from sales, listing where sales.listid = listing.listid
and sales.listid between 1000 and 1005
order by wait desc, priceperticket desc;
```

```

priceperticket | wait
-----+-----
 96.00         |    15
 123.00        |    11
 131.00        |     9
 123.00        |     6
 129.00        |     4
 96.00         |     4
 96.00         |     0
(7 rows)

```

此範例會計算賣家等待所有門票特價的平均小時數。

```

select avg(datediff(hours, listtime, saletime)) as avgwait
from sales, listing
where sales.listid = listing.listid;

avgwait
-----
 465
(1 row)

```

具有 TIME 欄的範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME) , 其中插入了三個值。

```

select time_val from time_test;

time_val
-----
20:00:00
00:00:00.5550
00:58:00

```

下列範例會尋找 TIME_VAL 欄與時間常值之間的小時數差異。

```

select datediff(hour, time_val, time '15:24:45') from time_test;

date_diff
-----
      -5

```

```
15
15
```

下列範例會尋找兩個常值時間值之間的分鐘數差異。

```
select datediff(minute, time '20:00:00', time '21:00:00') as nummins;

nummins
-----
60
```

具有 TIMTZ 欄的範例

下列範例資料表 TIMETZ_TEST 有一個 TIMETZ_VAL 欄 (類型為 TIMETZ) , 其中插入了三個值。

```
select timetz_val from timetz_test;

timetz_val
-----
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會尋找 TIMETZ 常值與 timetz_val 之間的小時數差異。

```
select datediff(hours, timetz '20:00:00 PST', timetz_val) as numhours from timetz_test;

numhours
-----
0
-4
1
```

下列範例會尋找兩個常值 TIMTZ 值之間的小時數差異。

```
select datediff(hours, timetz '20:00:00 PST', timetz '00:58:00 EST') as numhours;

numhours
-----
1
```

DATE_PART 函數

DATE_PART 會從運算式擷取日期部分值。DATE_PART 是 PGDATE_PART 函數的同義詞。

語法

```
DATE_PART(datepart, {date|timestamp})
```

引數

datepart

函數所操作的日期值的特定部分 (例如年、月或日) 的識別碼常值或字串。如需詳細資訊，請參閱 [日期或時間戳記函數的日期部分](#)。

{date|timestamp}

日期欄、時間戳記欄，或會隱性轉換為日期或時間戳記的運算式。Date 或 timestamp 中的欄或運算式必須包含 datepart 中指定的日期部分。

傳回類型

DOUBLE

範例

DATE_PART 函數的預設欄名稱為 pgdate_part。

如需其中一些範例中所使用資料的相關資訊，請參閱 [範本資料庫](#)。

下列範例會從時間戳記常值尋找分鐘。

```
SELECT DATE_PART(minute, timestamp '20230104 04:05:06.789');
```

```
pgdate_part
-----
          5
```

下列範例會從時間戳記常值中尋找週數。週數計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 [ISO 8601](#)。

```
SELECT DATE_PART(week, timestamp '20220502 04:05:06.789');
```

```
pgdate_part
-----
          18
```

下列範例會從時間戳記常值尋找月份中的日期。

```
SELECT DATE_PART(day, timestamp '20220502 04:05:06.789');

pgdate_part
-----
          2
```

下列範例會從時間戳記常值尋找星期幾。週數計算是從 0-6 開始的整數，以星期日開始。

```
SELECT DATE_PART(dayofweek, timestamp '20220502 04:05:06.789');

pgdate_part
-----
          1
```

下列範例會從時間戳記常值中尋找世紀。世紀計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 [ISO 8601](#)。

```
SELECT DATE_PART(century, timestamp '20220502 04:05:06.789');

pgdate_part
-----
         21
```

下列範例會從時間戳記常值中尋找千年。千年計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 [ISO 8601](#)。

```
SELECT DATE_PART(millennium, timestamp '20220502 04:05:06.789');

pgdate_part
-----
          3
```

下列範例會從時間戳記常值中尋找微秒。微秒計算遵循 ISO 8601 標準。如需詳細資訊，請參閱 Wikipedia 中的 [ISO 8601](#)。

```
SELECT DATE_PART(microsecond, timestamp '20220502 04:05:06.789');
```

```
pgdate_part
-----
      789000
```

下列範例會從日期常值中尋找月份。

```
SELECT DATE_PART(month, date '20220502');
```

```
pgdate_part
-----
          5
```

下列範例會將 DATE_PART 函數套用至資料表中的欄。

```
SELECT date_part(w, listtime) AS weeks, listtime
FROM listing
WHERE listid=10
```

```
weeks |      listtime
-----+-----
    25 | 2008-06-17 09:44:54
(1 row)
```

您可以將日期部分以全名或縮寫表示，在此情況下，w 代表週。

週日期部分的天會傳回從 0–6 的整數，從星期日開始。使用 DATE_PART 含小數點 (DAYOFWEEK)，來檢視星期六的事件。

```
SELECT date_part(dow, starttime) AS dow, starttime
FROM event
WHERE date_part(dow, starttime)=6
ORDER BY 2,1;
```

```
dow |      starttime
-----+-----
    6 | 2008-01-05 14:00:00
    6 | 2008-01-05 14:00:00
    6 | 2008-01-05 14:00:00
```

```
6 | 2008-01-05 14:00:00
...
(1147 rows)
```

DATE_PART_YEAR 函數

DATE_PART_YEAR 函數從日期擷取年份。

語法

```
DATE_PART_YEAR(date)
```

引數

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

下列範例會從日期常值中尋找年份。

```
SELECT DATE_PART_YEAR(date '20220502 04:05:06.789');

date_part_year
-----
2022
```

下列範例從 CALDATE 欄位中擷取年份。在 CALDATE 欄中的值是日期。如需此範例中所使用資料的相關資訊，請參閱 [範本資料庫](#)。

```
select caldate, date_part_year(caldate)
from date
order by
dateid limit 10;
```

```
caldate | date_part_year
-----+-----
2008-01-01 |          2008
2008-01-02 |          2008
2008-01-03 |          2008
2008-01-04 |          2008
2008-01-05 |          2008
2008-01-06 |          2008
2008-01-07 |          2008
2008-01-08 |          2008
2008-01-09 |          2008
2008-01-10 |          2008
(10 rows)
```

DATE_TRUNC 函數

DATE_TRUNC 函數會根據您指定的日期部分 (例如小時、天或月) 來截斷時間戳記運算式或常值。

語法

```
DATE_TRUNC('datepart', timestamp)
```

引數

datepart

時間戳記值截斷目標的日期部分。輸入 timestamp 被截斷為輸入 datepart 的精確度。例如，month 截斷為每個月的第一天。有效格式如下：

- microsecond, microseconds
- millisecond, milliseconds
- second, seconds
- minute, minutes
- hour, hours
- day, days
- week, weeks
- month, months
- quarter, quarters
- year, years

- decade, decades
- century, centuries
- millennium, millennia

如需某些格式縮寫的相關資訊，請參閱 [日期或時間戳記函數的日期部分](#)

timestamp

時間戳記欄或運算式會隱性轉換為時間戳記。

傳回類型

TIMESTAMP

範例

將輸入時間戳記截斷為秒。

```
SELECT DATE_TRUNC('second', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:06
```

將輸入時間戳記截斷為分鐘。

```
SELECT DATE_TRUNC('minute', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:05:00
```

將輸入時間戳記截斷為小時。

```
SELECT DATE_TRUNC('hour', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 04:00:00
```

將輸入時間戳記截斷為天。

```
SELECT DATE_TRUNC('day', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-30 00:00:00
```

將輸入時間戳記截斷為一個月的第一天。

```
SELECT DATE_TRUNC('month', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

將輸入時間戳記截斷為季度的第一天。

```
SELECT DATE_TRUNC('quarter', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-04-01 00:00:00
```

將輸入時間戳記截斷為一年的第一天。

```
SELECT DATE_TRUNC('year', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2020-01-01 00:00:00
```

將輸入時間戳記截斷為世紀的第一天。

```
SELECT DATE_TRUNC('millennium', TIMESTAMP '20200430 04:05:06.789');
date_trunc
2001-01-01 00:00:00
```

將輸入時間戳記截斷為一週的星期一。

```
select date_trunc('week', TIMESTAMP '20220430 04:05:06.789');
date_trunc
2022-04-25 00:00:00
```

在下列範例中，DATE_TRUNC 函數會使用「週」日期部分來傳回每週星期一的日期。

```
select date_trunc('week', saletime), sum(pricepaid) from sales where
saletime like '2008-09%' group by date_trunc('week', saletime) order by 1;
```

date_trunc	sum
-----+-----	
2008-09-01	2474899
2008-09-08	2412354
2008-09-15	2364707

```
2008-09-22 | 2359351
2008-09-29 | 705249
```

EXTRACT 函數

該 EXTRACT 函數從時間戳，TIMSTAMPTZ，時間，時間，時間，時間，間隔年到月，或間隔天到第二個值返回日期或時間部分。範例包括時間戳記中的日、月、年、時、分、秒、毫秒或微秒。

語法

```
EXTRACT(datepart FROM source)
```

引數

datepart

要擷取的日期或時間的分欄，例如日、月、年、小時、分鐘、秒、毫秒或微秒。對於可能的值，請參閱 [日期或時間戳記函數的日期部分](#)。

source

評估為「時間戳記」、「TIMSTAMPTZ」、「時間」、「時間」、「時間」、「年到月的間隔」或「從日到秒的間隔」資料類型的資料欄或運算式。

傳回類型

INTEGER 如果來源值評估為資料類型「時間戳記」、「時間」、「時間」、「年到月的間隔」或「從日到秒的間隔」。

如果 source 值計算為資料類型 TIMSTAMPTZ，則為 DOUBLE PRECISION。

TIMESTAMP 範例

下列範例會判斷特價時，售價是 10,000 USD 或更多的週次。此範例使用 TICKIT 資料。如需詳細資訊，請參閱 [範本資料庫](#)。

```
select salesid, extract(week from saletime) as weeknum
from sales
where pricepaid > 9999
order by 2;

salesid | weeknum
```

```

-----+-----
159073 |      6
160318 |      8
161723 |     26

```

下列範例會從常值 timestamp 值傳回分鐘值。

```

select extract(minute from timestamp '2009-09-09 12:08:43');

date_part
-----
8

```

下列範例會從常值 timestamp 值傳回毫秒值。

```

select extract(ms from timestamp '2009-09-09 12:08:43.101');

date_part
-----
101

```

TIMESTAMPTZ 範例

下列範例會從常值 timestamptz 傳回年份值。

```

select extract(year from timestamptz '1.12.1997 07:37:16.00 PST');

date_part
-----
1997

```

TIME 範例

下列範例資料表 TIME_TEST 有一個 TIME_VAL 欄 (類型為 TIME) , 其中插入了三個值。

```

select time_val from time_test;

time_val
-----
20:00:00
00:00:00.5550

```

```
00:58:00
```

下列範例會擷取每個 `time_val` 的分鐘。

```
select extract(minute from time_val) as minutes from time_test;
```

```
minutes
-----
      0
      0
     58
```

下列範例會擷取每個 `time_val` 中的小時數。

```
select extract(hour from time_val) as hours from time_test;
```

```
hours
-----
     20
      0
      0
```

下列範例會從常值值擷取毫秒。

```
select extract(ms from time '18:25:33.123456');
```

```
date_part
-----
     123
```

TIMETZ 範例

下列範例資料表 `TIMETZ_TEST` 有一個 `TIMETZ_VAL` 欄 (類型為 `TIMETZ`)，其中插入了三個值。

```
select timetz_val from timetz_test;
```

```
timetz_val
-----
04:00:00+00
00:00:00.5550+00
05:58:00+00
```

下列範例會擷取每個 `timetz_val` 的小時。

```
select extract(hour from timetz_val) as hours from time_test;
```

```
hours
-----
      4
      0
      5
```

下列範例會從常值值擷取毫秒。在處理擷取之前，常值不會轉換為 UTC。

```
select extract(ms from timetz '18:25:33.123456 EST');
```

```
date_part
-----
      123
```

下列範例會從常值 `timetz` 值傳回 UTC 的時區偏移小時。

```
select extract(timezone_hour from timetz '1.12.1997 07:37:16.00 PDT');
```

```
date_part
-----
      -7
```

具有間隔年到月和間隔日到第二的例子

下列範例會擷取定義 36 小時 (即 1 天 12 小時) 的 1 間隔天到秒的第二天部分。

```
select EXTRACT('days' from INTERVAL '36 hours' DAY TO SECOND)
```

```
date_part
-----
      1
```

下列範例會從定義 15 個月 (即 1 年 3 個月) 的年份擷取月份部分。

```
select EXTRACT('month' from INTERVAL '15 months' YEAR TO MONTH)
```

```

date_part
-----
3

```

下面的示例提取6從 30 個月，即 2 年 6 個月的月份部分。

```
select EXTRACT('month' from INTERVAL '30' MONTH)
```

```

date_part
-----
6

```

下列範例會2從 50 小時 (即 2 天 2 小時) 擷取小時部分。

```
select EXTRACT('hours' from INTERVAL '50' HOUR)
```

```

date_part
-----
2

```

下列範例會11從 1 小時 11 分鐘 11.123 秒擷取的分鐘部分。

```
select EXTRACT('minute' from INTERVAL '70 minutes 70.123 seconds' MINUTE TO SECOND)
```

```

date_part
-----
11

```

下面的例子1.11從 1 天 1 小時 1 分鐘 1.11 秒提取秒部分。

```
select EXTRACT('seconds' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND)
```

```

date_part
-----
1.11

```

下面的例子提取在一個 INTERVAL 的總小時數。每個零件都被萃取並加入到總計中。

```
select EXTRACT('days' from INTERVAL '50' HOUR) * 24 + EXTRACT('hours' from INTERVAL
'50' HOUR)
```

```
?column?
```

```
-----  
50
```

下列範例會擷取間隔中的總秒數。每個零件都被萃取並加入到總計中。

```
select EXTRACT('days' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 86400 +  
       EXTRACT('hours' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 3600 +  
       EXTRACT('minutes' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND) * 60 +  
       EXTRACT('seconds' from INTERVAL '1 day 1:1:1.11' DAY TO SECOND)
```

```
?column?
```

```
-----  
90061.11
```

GETDATE 函數

GETDATE 傳回目前工作階段時區中的目前日期和時間 (預設為 UTC)。它會傳回目前陳述式的開始日期或時間，即使是在交易區塊中也一樣。

語法

```
GETDATE()
```

括號是必要的。

傳回類型

TIMESTAMP

範例

下列範例中會使用 GETDATE 函數來傳回目前日期的完整時間戳記。

```
select getdate();
```

```
timestamp
```

```
-----  
2008-12-04 16:10:43
```

下列範例中會使用 TRUNC 函數中的 GETDATE 函數來傳回不含時間的目前日期。


```
select trunc(getdate());
```

```
trunc  
-----  
2008-12-04
```

INTERVAL_CMP 函數

INTERVAL_CMP 會比較兩個間隔並傳回 1 (如果第一個間隔較大)、-1 (如果第二個間隔較大)，且 0 (如果間隔相等)。如需詳細資訊，請參閱 [不含限定詞語法的間隔常值範例](#)。

語法

```
INTERVAL_CMP(interval1, interval2)
```

引數

interval1

間隔常值。

interval2

間隔常值。

傳回類型

INTEGER

範例

下列範例會比較 3 days 和 1 year 的值。

```
select interval_cmp('3 days','1 year');
```

```
interval_cmp  
-----  
-1
```

此範例會將值 7 days 與 1 week 進行比較。

```
select interval_cmp('7 days','1 week');

interval_cmp
-----
0
```

下列範例會比較 1 year 和 3 days 的值。

```
select interval_cmp('1 year','3 days');

interval_cmp
-----
1
```

LAST_DAY 函數

LAST_DAY 會傳回某月最後一天的日期，其中包含 date。不論 date 引數的資料類型為何，傳回類型一律是 DATE。

如需擷取特定日期部分的相關資訊，請參閱[DATE_TRUNC 函數](#)。

語法

```
LAST_DAY( { date | timestamp } )
```

引數

date | timestamp

資料類型 DATE 或 TIMESTAMP 的欄，或是隱含評估為 DATE 或 TIMESTAMP 類型的運算式。

傳回類型

DATE

範例

下列範例傳回目前月份之最後一天的日期。

```
select last_day(sysdate);
```

```
last_day
-----
2014-01-31
```

下列範例傳回該月份之最後 7 天每天所售的門票數。SALETIME 欄中的值是時間戳記。

```
select datediff(day, saletime, last_day(saletime)) as "Days Remaining", sum(qtysold)
from sales
where datediff(day, saletime, last_day(saletime)) < 7
group by 1
order by 1;
```

```
days remaining | sum
-----+-----
              0 | 10140
              1 | 11187
              2 | 11515
              3 | 11217
              4 | 11446
              5 | 11708
              6 | 10988
```

(7 rows)

MONTHS_BETWEEN 函數

MONTHS_BETWEEN 可判斷兩個日期之間有幾個月。

如果第一個日期晚於第二個日期，結果為正值，否則結果為負值。

如果引數為 null，則結果為 NULL。

語法

```
MONTHS_BETWEEN( date1, date2 )
```

引數

date1

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

date2

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

FLOAT8

結果的整個數字部分是基於日期的年份和月份值的差異。結果的小數部分是透過日期的天和時間戳記值計算而得，且一個月假設為 31 天。

如果 date1 和 date2 都含有某個月份中的相同日期 (例如，1/15/14 和 2/15/14) 或某月的最後一天 (例如，8/31/14 和 9/30/14)，則結果是根據日期之年份和月份值的整個數值，而不論時間戳記部分符合的情況 (若有)。

範例

下列範例傳回 1/18/1969 和 3/18/1969 的月份。

```
select months_between('1969-01-18', '1969-03-18')
as months;

months
-----
-2
```

下列範例傳回 1/18/1969 和 1/18/1969 的月份。

```
select months_between('1969-01-18', '1969-01-18')
as months;

months
-----
0
```

下列範例傳回事件之第一個和最後一個顯示間的月份。

```
select eventname,
min(starttime) as first_show,
max(starttime) as last_show,
months_between(max(starttime),min(starttime)) as month_diff
```

```

from event
group by eventname
order by eventname
limit 5;

```

eventname	first_show	last_show	month_diff
.38 Special	2008-01-21 19:30:00.0	2008-12-25 15:00:00.0	11.12
3 Doors Down	2008-01-03 15:00:00.0	2008-12-01 19:30:00.0	10.94
70s Soul Jam	2008-01-16 19:30:00.0	2008-12-07 14:00:00.0	10.7
A Bronx Tale	2008-01-21 19:00:00.0	2008-12-15 15:00:00.0	10.8
A Catered Affair	2008-01-08 19:30:00.0	2008-12-19 19:00:00.0	11.35

NEXT_DAY 函數

NEXT_DAY 會傳回在給定日期之後的指定那天之第一個執行個體的日期。

如果 day 值與給定日期是一週中的同一天，則會傳回該日的下一個出現時間。

語法

```
NEXT_DAY( { date | timestamp }, day )
```

引數

date | timestamp

資料類型 DATE 或 TIMESTAMP 的欄，或是隱含評估為 DATE 或 TIMESTAMP 類型的運算式。

天

包含任何天之名稱的字串。大寫並不重要。

有效值如下。

天	值
週日	日、週日、星期日
週一	一、週一、星期一
週二	二、週二、星期二

天	值
週三	三、週三、星期三
週四	四、週四、星期四
週五	五、週五、星期五
週六	六、週六、星期六

傳回類型

DATE

範例

下列範例請求會傳回在 8/20/2014 後第一個星期二的日期。

```
select next_day('2014-08-20', 'Tuesday');
```

```
next_day
-----
2014-08-26
```

下列範例會傳回 2008 年 1 月 1 日之後第一個星期二的 5:54:44 的日期。

```
select listtime, next_day(listtime, 'Tue') from listing limit 1;
```

```
listtime          | next_day
-----+-----
2008-01-01 05:54:44 | 2008-01-08
```

以下範例會取得第三季目標行銷日期。

```
select username, (firstname || ' ' || lastname) as name,
eventname, caldate, next_day (caldate, 'Monday') as marketing_target
from sales, date, users, event
where sales.buyerid = users.userid
and sales.eventid = event.eventid
and event.dateid = date.dateid
and date.qtr = 3
```

```
order by marketing_target, eventname, name;
```

username	name	eventname	caldate	
marketing_target				
-----+-----+-----+-----				
+-----				
MB026QSG	Callum Atkinson	.38 Special	2008-07-06	2008-07-07
WCR50YIU	Erasmus Alvarez	A Doll's House	2008-07-03	2008-07-07
CKT700IE	Hadassah Adkins	Ana Gabriel	2008-07-06	2008-07-07
VVG070U0	Nathan Abbott	Armando Manzanero	2008-07-04	2008-07-07
GEW77SII	Scarlet Avila	August: Osage County	2008-07-06	2008-07-07
ECR71CVS	Caryn Adkins	Ben Folds	2008-07-03	2008-07-07
KUW82CYU	Kaden Aguilar	Bette Midler	2008-07-01	2008-07-07
WZE78DJZ	Kay Avila	Bette Midler	2008-07-01	2008-07-07
HXY04NVE	Dante Austin	Britney Spears	2008-07-02	2008-07-07
URY81YWF	Wilma Anthony	Britney Spears	2008-07-02	2008-07-07

SYSDATE 函數

SYSDATE 傳回目前工作階段時區中的目前日期和時間 (預設為 UTC)。

Note

SYSDATE 傳回目前交易的日期和時間 (而不是目前陳述式的開始)。

語法

```
SYSDATE
```

此函數不需引數。

傳回類型

TIMESTAMP

範例

下列範例中會使用 SYSDATE 函數來傳回目前日期的完整時間戳記。

```
select sysdate;
```

```
timestamp
-----
2008-12-04 16:10:43.976353
```

下列範例中會使用 TRUNC 函數中的 SYSDATE 函數來傳回不含時間的目前日期。

```
select trunc(sysdate);

trunc
-----
2008-12-04
```

在發出查詢且當日期早於 120 天時，下列查詢會傳回落於該日期間之日期的特價資訊。

```
select salesid, pricepaid, trunc(saletime) as saletime, trunc(sysdate) as now
from sales
where saletime between trunc(sysdate)-120 and trunc(sysdate)
order by saletime asc;
```

salesid	pricepaid	saletime	now
91535	670.00	2008-08-07	2008-12-05
91635	365.00	2008-08-07	2008-12-05
91901	1002.00	2008-08-07	2008-12-05
...			

TIMEOFDAY 函數

TIMEOFDAY 是特別的別名，會用來傳回週間日、日期和時間做為字串值。它會傳回目前陳述式的當日時間字串，即使是在交易區塊中也一樣。

語法

```
TIMEOFDAY()
```

傳回類型

VARCHAR

範例

下列範例會透過使用 TIMEOFDAY 函數來傳回目前日期和時間。


```
select timeofday();
```

```
timeofday
```

```
-----
```

```
Thu Sep 19 22:53:50.333525 2013 UTC
```

TIMESTAMP_CMP 函數

比較兩個時間戳記的值並傳回整數。如果時間戳記相同，則函數會傳回 0。如果第一個時間戳記較大，則函數會傳回 1。如果第二個時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMP_CMP(timestamp1, timestamp2)
```

引數

timestamp1

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

timestamp2

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較時間戳記，並顯示比較結果。

```
SELECT TIMESTAMP_CMP('2008-01-24 06:43:29', '2008-01-24 06:43:29'),  
       TIMESTAMP_CMP('2008-01-24 06:43:29', '2008-02-18 02:36:48'), TIMESTAMP_CMP('2008-02-18  
       02:36:48', '2008-01-24 06:43:29');
```

```
timestamp_cmp | timestamp_cmp | timestamp_cmp  
-----+-----+-----  
              0 |             -1 |              1
```

例如，下列範例會比較清單的 LISTTIME 與 SALETIME。所有清單的 TIMESTAMP_CMP 的值是 -1，因為特價的時間戳記是在清單的時間戳記後。

```
select listing.listid, listing.listtime,
sales.saletime, timestamp_cmp(listing.listtime, sales.saletime)
from listing, sales
where listing.listid=sales.listid
order by 1, 2, 3, 4
limit 10;
```

listid	listtime	saletime	timestamp_cmp
1	2008-01-24 06:43:29	2008-02-18 02:36:48	-1
4	2008-05-24 01:18:37	2008-06-06 05:00:16	-1
5	2008-05-17 02:29:11	2008-06-06 08:26:17	-1
5	2008-05-17 02:29:11	2008-06-09 08:38:52	-1
6	2008-08-15 02:08:13	2008-08-31 09:17:02	-1
10	2008-06-17 09:44:54	2008-06-26 12:56:06	-1
10	2008-06-17 09:44:54	2008-07-10 02:12:36	-1
10	2008-06-17 09:44:54	2008-07-16 11:59:24	-1
10	2008-06-17 09:44:54	2008-07-22 02:23:17	-1
12	2008-07-25 01:45:49	2008-08-04 03:06:36	-1

(10 rows)

此範例顯示若是在相同的時間戳記時，TIMESTAMP_CMP 會傳回 0：

```
select listid, timestamp_cmp(listtime, listtime)
from listing
order by 1 , 2
limit 10;
```

listid	timestamp_cmp
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

(10 rows)

TIMESTAMP_CMP_DATE 函數

TIMESTAMP_CMP_DATE 會比較時間戳記值和日期。如果時間戳記和日期值相同，則函數會傳回 0。如果時間戳記依時間順序較大，則函數會傳回 1。如果日期較大，則函數會傳回 -1。

語法

```
TIMESTAMP_CMP_DATE(timestamp, date)
```

引數

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

例如，下列範例會比較日期 2008-06-18 與 LISTTIME。在此日期後所做的清單會傳回 1，在此日期前所做的清單會傳回 -1。LISTTIME 值是時間戳記。

```
select listid, listtime,
timestamp_cmp_date(listtime, '2008-06-18')
from listing
order by 1, 2, 3
limit 10;
```

listid	listtime	timestamp_cmp_date
1	2008-01-24 06:43:29	-1
2	2008-03-05 12:25:29	-1
3	2008-11-01 07:35:33	1

```

 4 | 2008-05-24 01:18:37 |          -1
 5 | 2008-05-17 02:29:11 |          -1
 6 | 2008-08-15 02:08:13 |           1
 7 | 2008-11-15 09:38:15 |           1
 8 | 2008-11-09 05:07:30 |           1
 9 | 2008-09-09 08:03:36 |           1
10 | 2008-06-17 09:44:54 |          -1

```

(10 rows)

TIMESTAMP_CMP_TIMESTAMPTZ 函數

TIMESTAMP_CMP_TIMESTAMPTZ 將時間戳記運算式值與含時區運算式的時間戳記比較。如果時間戳記和含時區值的時間戳記相同，則函數會傳回 0。如果時間戳記依時間順序較大，則函數會傳回 1。如果含時區的時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMP_CMP_TIMESTAMPTZ(timestamp, timestamptz)
```

引數

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較時間戳記與含時區的時間戳記，並顯示比較結果。

```

SELECT TIMESTAMP_CMP_TIMESTAMPTZ('2008-01-24 06:43:29', '2008-01-24 06:43:29+00'),
       TIMESTAMP_CMP_TIMESTAMPTZ('2008-01-24 06:43:29', '2008-02-18 02:36:48+00'),
       TIMESTAMP_CMP_TIMESTAMPTZ('2008-02-18 02:36:48', '2008-01-24 06:43:29+00');

```

```

timestamp_cmp_timestamptz | timestamp_cmp_timestamptz | timestamp_cmp_timestamptz
-----+-----+-----

```

0 | -1 | 1

TIMESTAMPTZ_CMP 函數

TIMESTAMPTZ_CMP 會比較含時區值的兩個時間戳記值並傳回整數。如果時間戳記相同，則函數會傳回 0。如果第一個時間戳記依時間順序較大，則函數會傳回 1。如果第二個時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMPTZ_CMP(timestamptz1, timestamptz2)
```

引數

timestamptz1

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

timestamptz2

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較含時區的時間戳記，並顯示比較結果。

```
SELECT TIMESTAMPTZ_CMP('2008-01-24 06:43:29+00', '2008-01-24 06:43:29+00'),
       TIMESTAMPTZ_CMP('2008-01-24 06:43:29+00', '2008-02-18 02:36:48+00'),
       TIMESTAMPTZ_CMP('2008-02-18 02:36:48+00', '2008-01-24 06:43:29+00');
```

```
timestamptz_cmp | timestamptz_cmp | timestamptz_cmp
-----+-----+-----
0 | -1 | 1
```

TIMESTAMPTZ_CMP_DATE 函數

TIMESTAMPTZ_CMP_DATE 會比較時間戳記值和日期的值。如果時間戳記和日期值相同，則函數會傳回 0。如果時間戳記依時間順序較大，則函數會傳回 1。如果日期較大，則函數會傳回 -1。

語法

```
TIMESTAMPTZ_CMP_DATE(timestamptz, date)
```

引數

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

date

DATE 資料類型的欄，或是隱含評估為 DATE 類型的運算式。

傳回類型

INTEGER

範例

下列範例會比較做為含時區之時間戳記的 LISTTIME 與日期 2008-06-18。在此日期後所做的清單會傳回 1，在此日期前所做的清單會傳回 -1。

```
select listid, CAST(listtime as timestamptz) as tstz,
timestamp_cmp_date(tstz, '2008-06-18')
from listing
order by 1, 2, 3
limit 10;
```

listid	tstz	timestamp_cmp_date
1	2008-01-24 06:43:29+00	-1
2	2008-03-05 12:25:29+00	-1
3	2008-11-01 07:35:33+00	1
4	2008-05-24 01:18:37+00	-1
5	2008-05-17 02:29:11+00	-1
6	2008-08-15 02:08:13+00	1
7	2008-11-15 09:38:15+00	1
8	2008-11-09 05:07:30+00	1
9	2008-09-09 08:03:36+00	1
10	2008-06-17 09:44:54+00	-1

(10 rows)

TIMESTAMPTZ_CMP_TIMESTAMP 函數

TIMESTAMPTZ_CMP_TIMESTAMP 將含時區運算式的時間戳記值與時間戳記運算式比較。如果含時區的時間戳記和時間戳記值相同，則函數會傳回 0。如果含時區的時間戳記依時間順序較大，則函數會傳回 1。如果時間戳記較大，則函數會傳回 -1。

語法

```
TIMESTAMPTZ_CMP_TIMESTAMP(timestamptz, timestamp)
```

引數

timestamptz

TIMESTAMPTZ 資料類型的欄，或是隱含評估為 TIMESTAMPTZ 類型的運算式。

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

傳回類型

INTEGER

範例

下列範例會將含時區的時間戳記與時間戳記進行比較並顯示比較結果。

```
SELECT TIMESTAMPTZ_CMP_TIMESTAMP('2008-01-24 06:43:29+00', '2008-01-24 06:43:29'),
       TIMESTAMPTZ_CMP_TIMESTAMP('2008-01-24 06:43:29+00', '2008-02-18 02:36:48'),
       TIMESTAMPTZ_CMP_TIMESTAMP('2008-02-18 02:36:48+00', '2008-01-24 06:43:29');
```

timestamptz_cmp_timestamp	timestamptz_cmp_timestamp	timestamptz_cmp_timestamp
0	-1	1

TIMEZONE 函數

TIMEZONE 傳回時間戳記，做為指定時區和時間戳記值。

如需有關如何設定時區的資訊和範例，請參閱 [timezone](#)。

如需有關如何轉換時區的資訊和範例，請參閱 [CONVERT_TIMEZONE](#)。

語法

```
TIMEZONE('timezone', { timestamp | timestamptz })
```

引數

timezone

傳回值的時區。您可以將時區指定為時區名稱 (例如 `'Africa/Kampala'` 或 `'Singapore'`) 或做為時區縮寫 (例如 `'UTC'` 或 `'PDT'`)。若要查看受支援時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

若要查看受支援時區縮寫的清單，請執行下列命令。

```
select pg_timezone_abbrevs();
```

如需詳細資訊和範例，請參閱 [時區使用須知](#)。

timestamp | timestamptz

造成 `TIMESTAMP` 或 `TIMESTAMPTZ` 類型的運算式會隱含地強制轉換為時間戳記或含時區的時間戳記。

傳回類型

與 `TIMESTAMP` 表達式搭配使用時的 `TIMESTAMPTZ`。

與 `TIMESTAMPTZ` 表達式搭配使用時的 `TIMESTAMP`。

範例

以下使用 `PST` 時區的時間戳記 `2008-06-17 09:44:54` 傳回 `UTC` 時區的時間戳記。

```
SELECT TIMEZONE('PST', '2008-06-17 09:44:54');
```

```
timezone
-----
2008-06-17 17:44:54+00
```

以下使用 `UTC` 時區的時間戳記 `2008-06-17 09:44:54+00` 傳回 `PST` 時區的時間戳記。


```
SELECT TIMEZONE('PST', timestampz('2008-06-17 09:44:54+00'));
```

```
timezone
```

```
-----  
2008-06-17 01:44:54
```

TO_TIMESTAMP 函數

TO_TIMESTAMP 會將 TIMESTAMP 字串轉換為 TIMESTAMPTZ。如需 Amazon Redshift 的其他日期和時間函數清單，請參閱[日期和時間函數](#)。

語法

```
to_timestamp(timestamp, format)
```

```
to_timestamp (timestamp, format, is_strict)
```

引數

timestamp

字串，代表 format 指定的格式中的時間戳記值。如果此引數保留為空白，則時間戳記值預設為 0001-01-01 00:00:00。

format

字串常值，定義 timestamp 值的格式。不支援包含時區 (TZ、tz 或 OF) 的格式做為輸出。請參閱[日期時間格式字串](#) 以取得有效的時間戳記格式。

is_strict

選用的 Boolean 值，指定如果輸入時間戳值超出範圍是否回傳錯誤。當 is_strict 設定為 TRUE 時，如果有超出範圍的值，就會傳回錯誤。當 is_strict 設定為 FALSE (預設值) 時，會接受溢位值。

傳回類型

TIMESTAMPTZ

範例

以下範例示範如何使用 TO_TIMESTAMP 函數，將 TIMESTAMP 字串轉換成 TIMESTAMPTZ。

```
select sysdate, to_timestamp(sysdate, 'YYYY-MM-DD HH24:MI:SS') as second;
```

```
timestamp                | second
-----
2021-04-05 19:27:53.281812 | 2021-04-05 19:27:53+00
```

可以傳遞日期的 TO_TIMESTAMP 部分。其餘日期部分設定為預設值。時間包含在輸出中：

```
SELECT TO_TIMESTAMP('2017', 'YYYY');
```

```
to_timestamp
-----
2017-01-01 00:00:00+00
```

以下 SQL 陳述式將字串 '2011-12-18 24:38:15' 轉換為 TIMESTAMPTZ。結果是 TIMESTAMPTZ 落在第二天，因為小時數超過 24 小時：

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS');
```

```
to_timestamp
-----
2011-12-19 00:38:15+00
```

下列 SQL 陳述式會將字串「2011-12-18 24:38:15」轉換為 TIMESTAMPTZ。結果會產生錯誤，因為時間戳記中的時間值超過 24 小時：

```
SELECT TO_TIMESTAMP('2011-12-18 24:38:15', 'YYYY-MM-DD HH24:MI:SS', TRUE);
```

```
ERROR:  date/time field time value out of range: 24:38:15.0
```

TRUNC 函數

截斷 TIMESTAMP 並傳回 DATE。

這個函數也可以截斷數字。如需詳細資訊，請參閱 [TRUNC 函數](#)。

語法

```
TRUNC(timestamp)
```

引數

timestamp

TIMESTAMP 資料類型的欄，或是隱含評估為 TIMESTAMP 類型的運算式。

若要傳回以 00:00:00 為時間的時間戳記值，請將函數結果轉換為 TIMESTAMP。

傳回類型

DATE

範例

以下範例從 SYSDATE 函數的結果傳回日期部分 (其會傳回時間戳記)。

```
SELECT SYSDATE;
```

```
+-----+
|      timestamp      |
+-----+
| 2011-07-21 10:32:38.248109 |
+-----+
```

```
SELECT TRUNC(SYSDATE);
```

```
+-----+
|   trunc   |
+-----+
| 2011-07-21 |
+-----+
```

下列範例會將 TRUNC 函數套用至 TIMESTAMP 欄。傳回類型為日期。

```
SELECT TRUNC(starttime) FROM event
ORDER BY eventid LIMIT 1;
```

```
+-----+
|   trunc   |
+-----+
| 2008-01-25 |
+-----+
```

下列範例會傳回時間為 00:00:00 的時間戳記值，方法是將 TRUNC 函數結果轉換 TIMESTAMP。

```
SELECT CAST((TRUNC(SYSDATE)) AS TIMESTAMP);
```

```
+-----+
|      trunc      |
+-----+
| 2011-07-21 00:00:00 |
+-----+
```

日期或時間戳記函數的日期部分

下表識別日期部分和時間部分名稱和縮寫，系統接受它們做為以下函數的引數：

- DATEADD
- DATEDIFF
- DATE_PART
- EXTRACT

日期部分或時間部分	縮寫
millennium, millennia	mil, mils
century, centuries	c, cent, cents
decade, decades	dec, decs
epoch	epoch (由 EXTRACT 支援)
year, years	y, yr, yrs
quarter, quarters	qtr, qtrs
month, months	mon, mons
week, weeks	w
週中的日	dayofweek, dow, dw, weekday (由 DATE_PART 和 EXTRACT 函數 所支援)

日期部分或時間部分	縮寫
	傳回從 0–6 的整數，從星期日開始。
	<div style="border: 1px solid #add8e6; border-radius: 10px; padding: 10px;"> <p>Note</p> <p>DOW 日期部分與週中的日 (日期部分用來表示日期時間格式是字串) 表現不同 (D)。D 是基於整數 1–7，其中星期日為 1。如需詳細資訊，請參閱 日期時間格式字串。</p> </div>
年中的日	dayofyear, doy, dy, yearday (由 EXTRACT 支援)
day, days	d
hour, hours	h, hr, hrs
minute, minutes	m, min, mins
second, seconds	s, sec, secs
millisecond, milliseconds	ms, msec, msecs, msecond, mseconds, millisec, millisecs, millisecon
microsecond, microseconds	microsec, microsecs, microsecond, usecond, useconds, us, usec, usecs
timezone, timezone_hour, timezone_minute	由 EXTRACT 函數僅針對含時區 (TIMESTAMP TZ) 的時間戳記所支援。

含秒、毫秒和微秒結果的差異

當不同日期函數指定秒、毫秒或微秒做為日期部分時，減去查詢結果的差異：

- [EXTRACT](#) 函數會傳回整數僅做為指定日期部分，而忽略更高或更低層級的日期部分。若指定的日期部分為秒，則毫秒或微秒不會包含在結果中。若指定的日期部分為豪秒，則秒或微秒不會包含在內。若指定的日期部分為微秒，則秒或豪秒不會包含在內。
- [DATE_PART](#) 函數會傳回時間戳記的完整秒部分，不論指定的日期部分為何，視需要傳回十進位值或整數。

例如，比較以下查詢的結果：

```
create table seconds(micro timestamp);

insert into seconds values('2009-09-21 11:10:03.189717');

select extract(sec from micro) from seconds;

date_part
-----
3

select date_part(sec, micro) from seconds;

pgdate_part
-----
3.189717
```

CENTURY、EPOCH、DECADE 和 MIL 備註

CENTURY 或 CENTURIES

Amazon Redshift 會解釋 CENTURY 來以年 ####1 開始並以年 ###0 結束：

```
select extract (century from timestamp '2000-12-16 12:21:13');
date_part
-----
20

select extract (century from timestamp '2001-12-16 12:21:13');
date_part
-----
21
```

EPOCH

EPOCH 的 Amazon Redshift 實作相對於 1970-01-01 00:00:00.000000，與叢集所在的時區無關。您可能想要根據叢集所在的時區，彌補結果的時數差異。

下列範例示範以下方法：

1. 建立名為 `EVENT_EXAMPLE` 且根據 `EVENT` 資料欄的資料表。此 `CREATE AS` 命令會使用 `DATE_PART` 函數來建立日期欄 (依預設, 名為 `PGDATE_PART`), 以儲存每個事件的 epoch 值。
2. 從 `PG_TABLE_DEF` 選取 `EVENT_EXAMPLE` 的欄和資料類型。
3. 從 `EVENT_EXAMPLE` 資料表選取 `EVENTNAME`、`STARTTIME` 和 `PGDATE_PART`, 來檢視不同的日期和時間格式。
4. 從 `EVENT_EXAMPLE` 中按現狀選取 `EVENTNAME` 和 `STARTTIME`。使用 1 秒間隔將 `PGDATE_PART` 中的 epoch 值轉換為不含時區的時間戳記, 並在名為 `CONVERTED_TIMESTAMP` 的欄位中傳回結果。

```
create table event_example
as select eventname, starttime, date_part(epoch, starttime) from event;

select "column", type from pg_table_def where tablename='event_example';
```

column	type
eventname	character varying(200)
starttime	timestamp without time zone
pgdate_part	double precision

(3 rows)

```
select eventname, starttime, pgdate_part from event_example;
```

eventname	starttime	pgdate_part
Mamma Mia!	2008-01-01 20:00:00	1199217600
Spring Awakening	2008-01-01 15:00:00	1199199600
Nas	2008-01-01 14:30:00	1199197800
Hannah Montana	2008-01-01 19:30:00	1199215800
K.D. Lang	2008-01-01 15:00:00	1199199600
Spamalot	2008-01-02 20:00:00	1199304000
Macbeth	2008-01-02 15:00:00	1199286000
The Cherry Orchard	2008-01-02 14:30:00	1199284200
Macbeth	2008-01-02 19:30:00	1199302200
Demi Lovato	2008-01-02 19:30:00	1199302200

```
select eventname,
starttime,
```

```
timestamp with time zone 'epoch' + pgdate_part * interval '1 second' AS
  converted_timestamp
from event_example;
```

eventname	starttime	converted_timestamp
Mamma Mia!	2008-01-01 20:00:00	2008-01-01 20:00:00
Spring Awakening	2008-01-01 15:00:00	2008-01-01 15:00:00
Nas	2008-01-01 14:30:00	2008-01-01 14:30:00
Hannah Montana	2008-01-01 19:30:00	2008-01-01 19:30:00
K.D. Lang	2008-01-01 15:00:00	2008-01-01 15:00:00
Spamalot	2008-01-02 20:00:00	2008-01-02 20:00:00
Macbeth	2008-01-02 15:00:00	2008-01-02 15:00:00
The Cherry Orchard	2008-01-02 14:30:00	2008-01-02 14:30:00
Macbeth	2008-01-02 19:30:00	2008-01-02 19:30:00
Demi Lovato	2008-01-02 19:30:00	2008-01-02 19:30:00
...		

DECADE 或 DECADES

Amazon Redshift 會根據一般日曆解釋 DECADE 或 DECADES DATEPART。例如，因為一般日曆是從年份 1 開始，第一個十年 (十年 1) 為 0001-01-01 至 0009-12-31，而第二個十年 (十年 2) 為 0010-01-01 到 0019-12-31。例如，十年 201 橫跨 2000-01-01 到 2009-12-31：

```
select extract(decade from timestamp '1999-02-16 20:38:40');
date_part
-----
200

select extract(decade from timestamp '2000-02-16 20:38:40');
date_part
-----
201

select extract(decade from timestamp '2010-02-16 20:38:40');
date_part
-----
202
```

MIL 或 MILS

Amazon Redshift 會解釋 MIL 來以年 #001 的第一天開始並以年 #000 的最後一天結束：


```
select extract (mil from timestamp '2000-12-16 12:21:13');
date_part
-----
2

select extract (mil from timestamp '2001-12-16 12:21:13');
date_part
-----
3
```

雜湊函數

主題

- [CHECKSUM 函數](#)
- [farmFingerprint64 函數](#)
- [FUNC_SHA1 函數](#)
- [FNV_HASH 函數](#)
- [MD5 函數](#)
- [SHA 函數](#)
- [SHA1 函數](#)
- [SHA2 函數](#)
- [MURMUR3_32_HASH](#)

雜湊函數是一種數學函數，可將數字輸入值轉換成另一個值。

CHECKSUM 函數

計算用來建立雜湊索引的檢查總和值。

語法

```
CHECKSUM(expression)
```

引數

運算式

輸入表達式必須是 VARCHAR、INTEGER或 DECIMAL資料類型。

傳回類型

CHECKSUM 函數會傳回整數。

範例

下列範例會計算資料COMMISSION欄的檢查總和值：

```
select checksum(commission)
from sales
order by salesid
limit 10;
```

```
checksum
-----
10920
1140
5250
2625
2310
5910
11820
2955
8865
975
(10 rows)
```

farmFingerprint64 函數

使用 Fingerprint64 函數計算輸入引數的 farmhash 值。

語法

```
farmFingerprint64(expression)
```

引數

運算式

輸入運算式必須是 VARCHAR 或 VARBYTE 資料類型。

傳回類型

farmFingerprint64 函數會傳回 BIGINT。

範例

下列範例會傳回輸入 Amazon Redshift 為 VARCHAR 資料類型之的 farmFingerprint64 值。

```
SELECT farmFingerprint64('Amazon Redshift');
```

```
farmfingerprint64
-----
8085098817162212970
```

下列範例會傳回輸入 Amazon Redshift 為 VARBYTE 資料類型之的 farmFingerprint64 值。

```
SELECT farmFingerprint64('Amazon Redshift'::varbyte);
```

```
farmfingerprint64
-----
8085098817162212970
```

FUNC_SHA1 函數

SHA1 函數的同義詞。

請參閱 [SHA1 函數](#)。

FNV_HASH 函數

計算所有基本資料類型的 64 位元 FNV-1a 非加密雜湊函數。

語法

```
FNV_HASH(value [, seed])
```

引數

值

要進行雜湊的輸入值。Amazon Redshift 使用值的二進位表示法來雜湊輸入值；例如，使用 4 個位元組雜湊INTEGER值，並使用 8 個位元組雜湊BIGINT值。此外，雜湊CHAR和VARCHAR輸入不會忽略尾端空格。

seed

雜湊函數的BIGINT種子為選用。如果未提供，Amazon Redshift 會使用預設FNV種子。這會允許組合多個欄位的雜湊，而無須進行轉換或串連。

傳回類型

BIGINT

範例

下列範例會傳回數字的FNV雜湊、字串 'Amazon Redshift'，以及兩者的串連。

```
select fnv_hash(1);
       fnv_hash
-----
-5968735742475085980
(1 row)
```

```
select fnv_hash('Amazon Redshift');
       fnv_hash
-----
7783490368944507294
(1 row)
```

```
select fnv_hash('Amazon Redshift', fnv_hash(1));
       fnv_hash
-----
-2202602717770968555
```

```
(1 row)
```

使用須知

- 若要計算具有多個資料欄之資料表的雜湊，您可以計算第一欄的FNV雜湊，並將其作為種子傳遞至第二欄的雜湊。然後，它會將第二欄的FNV雜湊作為種子傳遞到第三欄的雜湊。

以下範例會建立種子來雜湊包含多個欄位的資料表。

```
select fnv_hash(column_3, fnv_hash(column_2, fnv_hash(column_1))) from sample_table;
```

- 相同屬性可以用來運算字串串連的雜湊。

```
select fnv_hash('abcd');
       fnv_hash
-----
-281581062704388899
(1 row)
```

```
select fnv_hash('cd', fnv_hash('ab'));
       fnv_hash
-----
-281581062704388899
(1 row)
```

- 雜湊函數會使用輸入的類型來判斷要雜湊的位元組數。如有必要，其會使用轉換來強制使用特定類型。

以下範例使用不同的輸入類型來產生不同結果。

```
select fnv_hash(1::smallint);
       fnv_hash
-----
589727492704079044
(1 row)
```

```
select fnv_hash(1);
       fnv_hash
-----
-5968735742475085980
```

```
(1 row)
```

```
select fnv_hash(1::bigint);
       fnv_hash
-----
-8517097267634966620
(1 row)
```

MD5 函數

使用MD5密碼編譯雜湊函數將可變長度字串轉換為 32 個字元的字串，該字串是 128 位元總和檢查碼十六進位值的文字表示。

語法

```
MD5(string)
```

引數

string

可變長度字串。

傳回類型

MD5 函數會傳回 32 字元字串，該字串是 128 位元總和檢查碼十六進位值的文字表示。

範例

下列範例顯示字串 'Amazon Redshift' 的 128 位元值：

```
select md5('Amazon Redshift');
       md5
-----
f7415e33f972c03abd4f3fed36748f7a
(1 row)
```

SHA 函數

SHA1 函數的同義詞。

請參閱 [SHA1 函數](#)。

SHA1 函數

SHA1 函數使用SHA1密碼編譯雜湊函數，將可變長度字串轉換為 40 個字元的字串，該字串是 160 位元總和檢查碼十六進位值的文字表示。

語法

SHA1 是 [SHA 函數](#) 和 [FUNC_SHA1 函數](#) 的同義字。

```
SHA1(string)
```

引數

string

可變長度字串。

傳回類型

SHA1 函數會傳回 40 字元字串，該字串是 160 位元總和檢查碼十六進位值的文字表示。

範例

下列範例傳回字詞 'Amazon Redshift' 的 160 位元值：

```
select sha1('Amazon Redshift');
```

SHA2 函數

SHA2 函數使用SHA2密碼編譯雜湊函數，將可變長度字串轉換為字元字串。字串是檢查總和之十六進位值的文字表示法，具有指定的位元數。

語法

```
SHA2(string, bits)
```

引數

string

可變長度字串。

integer

雜湊函數中的位元數。有效值為 0 (與 256 相同)、224、256、384 和 512。

傳回類型

SHA2 函數會傳回字元字串，該字串是總和檢查碼十六進位值的文字表示，如果位元數目無效，則傳回空字串。

範例

下列範例傳回字詞 'Amazon Redshift' 的 256 位元值：

```
select sha2('Amazon Redshift', 256);
```

MURMUR3_32_HASH

MURMUR3_32_HASH 函數會運算所有常見資料類型的 32 位元 Murmur3A 非加密雜湊，包括數值和字串類型。

語法

```
MURMUR3_32_HASH(value [, seed])
```

引數

值

要進行雜湊的輸入值。Amazon Redshift 雜湊輸入值的二進位表示。這種行為類似於 [FNV_HASH 函數](#)，但該值被轉換為 [Apache Iceberg 32 位元 Murmur3 雜湊規格](#) 指定的二進位表示。

seed

雜湊函數的 INT 種子。此為選用引數。如果未指定，Amazon Redshift 將使用預設種子 0。這會允許組合多個欄位的雜湊，而無須進行轉換或串連。

傳回類型

函數會傳回 INT。

範例

下列範例會分別傳回數字、字串 'Amazon Redshift' 及兩個值串連之後的 Murmur3 雜湊。

```
select MURMUR3_32_HASH(1);

      MURMUR3_32_HASH
-----
      1392991556
(1 row)
```

```
select MURMUR3_32_HASH('Amazon Redshift');

      MURMUR3_32_HASH
-----
     -1563580564
(1 row)
```

```
select MURMUR3_32_HASH('Amazon Redshift', MURMUR3_32_HASH(1));

      MURMUR3_32_HASH
-----
     -1346554171
(1 row)
```

使用須知

如要運算包含多個欄位資料表的雜湊，您可以運算第一個欄位的 Murmur3 雜湊，然後將其做為種子傳遞給第二個欄位的雜湊。然後，其會將第二個欄位的 Murmur3 雜湊做為種子傳遞至第三個欄位的雜湊。

以下範例會建立種子來雜湊包含多個欄位的資料表。

```
select MURMUR3_32_HASH(column_3, MURMUR3_32_HASH(column_2, MURMUR3_32_HASH(column_1)))
from sample_table;
```

相同屬性可以用來運算字串串連的雜湊。

```
select MURMUR3_32_HASH('abcd');
```

```
      MURMUR3_32_HASH  
-----  
1139631978  
(1 row)
```

```
select MURMUR3_32_HASH('cd', MURMUR3_32_HASH('ab'));
```

```
      MURMUR3_32_HASH  
-----  
1711522338  
(1 row)
```

雜湊函數會使用輸入的類型來判斷要雜湊的位元組數。如有必要，其會使用轉換來強制使用特定類型。

以下範例使用不同的輸入類型來產生不同結果。

```
select MURMUR3_32_HASH(1, MURMUR3_32_HASH(1));
```

```
      MURMUR3_32_HASH  
-----  
-1193428387  
(1 row)
```

```
select MURMUR3_32_HASH(1);
```

```
      MURMUR3_32_HASH  
-----  
1392991556  
(1 row)
```

```
select MURMUR3_32_HASH(1, MURMUR3_32_HASH(2));
```

```
      MURMUR3_32_HASH  
-----  
1179621905  
(1 row)
```

HyperLogLog 函數

接下來，您可以找到SQL該 Amazon Redshift 支援的 HyperLogLog 函數說明。

主題

- [HLL 函數](#)
- [HLL_CREATE_SKETCH 函數](#)
- [HLL_CARDINALITY 函數](#)
- [HLL_COMBINE 函數](#)
- [HLL_COMBINE_SKETCHES 函數](#)

HLL 函數

HLL 函數會傳回輸入表達式值的 HyperLogLog 基數。HLL 函數適用於資料類型以外的任何 HLLSKETCH資料類型。HLL 函數會忽略NULL值。當資料表中沒有資料列或所有資料列都是時 NULL，產生的基數為 0。

語法

```
HLL (aggregate_expression)
```

引數

aggregate_expression

提供要彙總之值的任何有效運算式，例如欄名。此函數支援除 HLLSKETCH、GEOMETRY、GEOGRAPHY和 以外的任何資料類型作為輸入VARBYTE。

傳回類型

HLL 函數會傳回 BIGINT或 INT8值。

範例

下列範例會傳回資料表 `a_table` 中資料欄 `an_int` 的基數。

```
CREATE TABLE a_table(an_int INT);
INSERT INTO a_table VALUES (1), (2), (3), (4);
```

```
SELECT hll(an_int) AS cardinality FROM a_table;
cardinality
-----
4
```

HLL_CREATE_SKETCH 函數

HLL_CREATE_SKETCH 函數傳回封裝輸入表達式值的HLLSKETCH資料類型。HLL_CREATE_SKETCH 函數適用於任何資料類型，並忽略NULL值。當資料表中沒有資料列或所有資料列都是時NULL，產生的草圖沒有索引值對，例如 {"version":1,"logm":15,"sparse":{"indices":[],"values":[]}}。

語法

```
HLL_CREATE_SKETCH (aggregate_expression)
```

引數

aggregate_expression

提供要彙總之值的任何有效運算式，例如欄名。NULL 會忽略 值。此函數支援除 HLLSKETCH、GEOMETRY、GEOGRAPHY和 以外的任何資料類型做為輸入VARBYTE。

傳回類型

HLL_CREATE_SKETCH 函數會傳回HLLSKETCH值。

範例

下列範例傳回資料表 *an_int*中資料欄的HLLSKETCH類型*a_table*。JSON 物件用於在匯入、匯出或列印草圖時表示稀疏 HyperLogLog草圖。字串表示法（Base64 格式）用於表示密集 HyperLogLog 草圖。

```
CREATE TABLE a_table(an_int INT);
INSERT INTO a_table VALUES (1), (2), (3), (4);

SELECT hll_create_sketch(an_int) AS sketch FROM a_table;
sketch
-----
{"version":1,"logm":15,"sparse":{"indices":
[20812342,20850007,22362299,47158030],"values":[1,2,1,1]}}
```

```
(1 row)
```

HLL_CARDINALITY 函數

HLL_CARDINALITY 函數會傳回輸入HLLSKETCH資料類型的基數。

語法

```
HLL_CARDINALITY (hllsketch_expression)
```

引數

hllsketch_expression

任何評估為 HLLSKETCH類型的有效表達式，例如資料欄名稱。輸入值是HLLSKETCH資料類型。

傳回類型

HLL_CARDINALITY 函數會傳回 BIGINT或 INT8值。

範例

下列範例會傳回資料表 `hll_table` 中資料欄 `sketch` 的基數。

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
(5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

SELECT hll_cardinality(sketch) AS cardinality FROM hll_table;
cardinality
-----
6
4
(2 rows)
```

HLL_COMBINE 函數

HLL_COMBINE 彙總函數會傳回合併所有輸入HLLSKETCH值的HLLSKETCH資料類型。

兩個或多個 HyperLogLog 草圖的組合是新的 HLLSKETCH，其封裝了每個輸入草圖所代表不同值的結合相關資訊。在合併草圖之後，Amazon Redshift 會擷取兩個或多個資料集的聯集基數。如需如何合併多個草圖的相關資訊，請參閱 [範例：從合併多個 HyperLogLog 草圖傳回草圖](#)。

語法

```
HLL_COMBINE (hllsketch_expression)
```

引數

hllsketch_expression

任何評估為 HLLSKETCH 類型的有效表達式，例如資料欄名稱。輸入值是 HLLSKETCH 資料類型。

傳回類型

HLL_COMBINE 函數傳回 HLLSKETCH 類型。

範例

下列範例會傳回資料表 中的組合 HLLSKETCH 值 `hll_table`。

```
CREATE TABLE a_table(an_int INT, b_int INT);
INSERT INTO a_table VALUES (1,1), (2,1), (3,1), (4,1), (1,2), (2,2), (3,2), (4,2),
(5,2), (6,2);

CREATE TABLE hll_table (sketch HLLSKETCH);
INSERT INTO hll_table select hll_create_sketch(an_int) from a_table group by b_int;

SELECT hll_combine(sketch) AS sketches FROM hll_table;
sketches
-----
{"version":1,"logm":15,"sparse":{"indices":
[20812342,20850007,22362299,40314817,42650774,47158030],"values":[1,2,1,3,2,1]}}
(1 row)
```

HLL_COMBINE_SKETCHES 函數

HLL_COMBINE_SKETCHES 是一種純量函數，作為輸入兩個 HLLSKETCH 值，並將其合併為單一 HLLSKETCH。

兩個或多個 HyperLogLog 草圖的組合是新的 HLLSKETCH，其封裝了每個輸入草圖所代表不同值的結合相關資訊。

語法

```
HLL_COMBINE_SKETCHES (hllsketch_expression1, hllsketch_expression2)
```

引數

hllsketch_expression1 和 *hllsketch_expression2*

任何評估為 HLLSKETCH 類型的有效表達式，例如資料欄名稱。

傳回類型

HLL_COMBINE_SKETCHES 函數傳回 HLLSKETCH 類型。

範例

下列範例會傳回資料表 中的組合 HLLSKETCH 值 *tbl1_table*。

```
WITH tbl1(x, y)
  AS (SELECT Hll_create_sketch(1),
           Hll_create_sketch(2)
       UNION ALL
       SELECT Hll_create_sketch(3),
           Hll_create_sketch(4)
       UNION ALL
       SELECT Hll_create_sketch(5),
           Hll_create_sketch(6)
       UNION ALL
       SELECT Hll_create_sketch(7),
           Hll_create_sketch(8)),
  tbl2(x, y)
  AS (SELECT Hll_create_sketch(9),
           Hll_create_sketch(10)
       UNION ALL
       SELECT Hll_create_sketch(11),
           Hll_create_sketch(12)
       UNION ALL
       SELECT Hll_create_sketch(13),
           Hll_create_sketch(14)
       UNION ALL
```

```
SELECT H11_create_sketch(15),
       H11_create_sketch(16)
UNION ALL
SELECT H11_create_sketch(NULL),
       H11_create_sketch(NULL)),
tbl3(x, y)
AS (SELECT *
     FROM tbl1
     UNION ALL
     SELECT *
     FROM tbl2)
SELECT H11_combine_sketches(x, y)
FROM tbl3;
```

JSON 函數

主題

- [JSON_PARSE 函數](#)
- [CAN_JSON_PARSE 函數](#)
- [JSON_SERIALIZE 函數](#)
- [JSON_SERIALIZE_TO_VARBYTE 函數](#)
- [文字型JSON函數](#)

Note

建議您使用下列函數來使用 JSON：

- [JSON_PARSE 函數](#)
- [CAN_JSON_PARSE 函數](#)
- [JSON_SERIALIZE 函數](#)
- [JSON_SERIALIZE_TO_VARBYTE 函數](#)

使用 `JSON_PARSE`，您只需在擷取時將JSON文字轉換為SUPER類型值一次，之後即可在SUPER值上操作。Amazon Redshift 剖析SUPER值比更有效率VARCHAR，這是文字型JSON函數的輸出。如需使用SUPER資料類型的詳細資訊，請前往 [Amazon Redshift 中的半結構化資料](#)。

當您需要存放一組相對較小的鍵值對時，可以透過將資料儲存為 JSON 格式來節省空間。由於 JSON 字串可以儲存在單一資料欄中，因此使用 JSON 可能比以表格格式儲存資料更有效率。例如，假設您有一個稀疏的資料表，您需要有多個資料欄才能完全代表所有可能的屬性，但大部分的資料欄值是 NULL 針對任何給定的資料列或任何給定的資料欄。透過使用 JSON 進行儲存，您或許可以在單一 JSON 字串中將索引鍵：值對中資料的資料存放在資料列中，並消除稀疏填入的資料表資料欄。

此外，您可以輕鬆修改 JSON 字串，以便在 JSON 結構描述變更時儲存其他索引鍵：值對，而無需將資料欄新增至資料表。

建議您 JSON 謹慎使用。JSON 不是存放較大資料集的好選擇，因為透過將不同的資料儲存在單一資料欄中，JSON 不使用 Amazon Redshift 資料欄存放區架構。雖然 Amazon Redshift 支援 CHAR 和 VARCHAR 欄的 JSON 函數，但建議使用 SUPER 處理 JSON 序列化格式的資料。SUPER 使用剖析後的無結構描述表示法，可有效率地查詢階層資料。如需 SUPER 資料類型的詳細資訊，請參閱 [Amazon Redshift 中的半結構化資料](#)。

JSON 使用 UTF-8 編碼的文字字串，因此 JSON 字串可以儲存為 CHAR 或 VARCHAR 資料類型。

JSON 根據下列規則 JSON，字串必須正確格式化為：

- 根層級 JSON 可以是 JSON 物件或 JSON 陣列。JSON 物件是一組由捲曲的括號所包圍的逗號分隔鍵：值對。

例如 {"one":1, "two":2}

- JSON 陣列是由括號括住的一組逗號分隔值。

以下是範例：["first", {"one":1}, "second", 3, null]

- JSON 陣列使用以零為基礎的索引；陣列中的第一個元素位於位置 0。在 JSON 索引鍵：值對中，索引鍵是雙引號中的字串。
- JSON 值可以是下列任何一項：
 - JSON 物件
 - 陣列
 - string
 - 使用雙引號表示
 - number
 - 包括整數、小數和浮點數
 - boolean
 - null

- 空的物件和空的陣列是有效的JSON值。
- JSON 欄位區分大小寫。
- 會忽略JSON結構元素（例如 { }, []）之間的空格。

Amazon Redshift JSON函數和 Amazon Redshift COPY命令使用相同的方法來使用JSON格式化的資料。如需使用的詳細資訊JSON，請參閱 [COPY 從JSON格式](#)

JSON_PARSE 函數

JSON_PARSE 函數會以 JSON 格式剖析資料，並將其轉換為SUPER表示式。

若要使用 INSERT或 UPDATE命令擷取SUPER資料類型，請使用 JSON_PARSE 函數。當您使用 JSON_PARSE () 將JSON字串剖析為SUPER值時，會套用某些限制。如需其他資訊，請參閱 [的剖析選項 SUPER](#)。

語法

```
JSON_PARSE( {json_string | binary_value} )
```

引數

json_string

傳回序列化JSON為 VARBYTE或 VARCHAR類型的表達式。

binary_value

VARBYTE 類型二進位值。

傳回類型

SUPER

範例

若要將JSON陣列轉換為[10001,10002,"abc"]SUPER資料類型，請使用下列範例。

```
SELECT JSON_PARSE('[10001,10002,"abc"]');
```

```
+-----+
| json_parse |
```

```
+-----+
| [10001,10002,"abc"] |
+-----+
```

若要確保函數將JSON陣列轉換為SUPER資料類型，請使用下列範例。如需詳細資訊，請參閱 [JSON_TYPEOF 函數](#)

```
SELECT JSON_TYPEOF(JSON_PARSE('[10001,10002,"abc"]'));
```

```
+-----+
| json_typeof |
+-----+
| array       |
+-----+
```

CAN_JSON_PARSE 函數

如果true結果可以使用 CAN_ 函數轉換為SUPER值，JSON_PARSEJSON_PARSE 函數會以 JSON 格式剖析資料並傳回。

語法

```
CAN_JSON_PARSE( {json_string | binary_value} )
```

引數

json_string

傳回序列JSON化VARCHAR格式的表達式。

binary_value

VARBYTE 類型二進位值。

傳回類型

BOOLEAN

使用須知

- CAN_JSON_PARSE 傳回空白字串的 false。當輸入引數為 null NULL 時，它會傳回。

範例

下列範例顯示使用 CASE 條件在正確形成的JSON陣列上執行的 CAN_JSON_PARSE。它傳回 true，因此 Amazon Redshift 會在範例值上執行 JSON_PARSE 函數。

```
SELECT CASE
    WHEN CAN_JSON_PARSE('[10001,10002,"abc"']')
    THEN JSON_PARSE('[10001,10002,"abc"']')
END;

case
-----
'[10001,10002,"abc"']'
```

下列範例顯示 CAN_JSON_PARSE 在使用 CASE 條件JSON的格式以外的值上執行。它傳回 false，因此 Amazon Redshift 會改為傳回CASE條件子ELSE句中的區段。

```
SELECT CASE
    WHEN CAN_JSON_PARSE('This is a string.')
    THEN JSON_PARSE('This is a string.')
    ELSE 'This is not JSON.'
END;

case
-----
"This is not JSON."
```

JSON_SERIALIZE 函數

JSON_SERIALIZE 函數會將SUPER運算式序列化為文字JSON表示法，以遵循 RFC 8259。如需該的詳細資訊RFC，請參閱 [JavaScript 物件記號 \(JSON\) 資料交換格式](#)。

SUPER 大小限制與區塊限制大致相同，且 VARCHAR 限制小於 SUPER 大小限制。因此，當JSON格式超過系統的 varchar 限制時，JSON_SERIALIZE 函數會傳回錯誤。若要檢查 SUPER 運算式的大小，請參閱 [JSON_SIZE](#) 函數。

語法

```
JSON_SERIALIZE(super_expression)
```

引數

`super_expression`

SUPER 運算式或欄。

傳回類型

VARCHAR

範例

若要將 SUPER 值序列化為字串，請使用下列範例。

```
SELECT JSON_SERIALIZE(JSON_PARSE('[10001,10002,"abc"]'));
```

```
+-----+
| json_serialize |
+-----+
| [10001,10002,"abc"] |
+-----+
```

JSON_SERIALIZE_TO_VARBYTE 函數

JSON_SERIALIZE_TO_VARBYTE 函數會將SUPER值轉換為類似於 JSON_SERIALIZE () 的JSON字串，但改為儲存在VARBYTE值中。

語法

```
JSON_SERIALIZE_TO_VARBYTE(super_expression)
```

引數

`super_expression`

SUPER 運算式或欄。

傳回類型

VARBYTE

範例

若要序列化 SUPER 值，並以 VARBYTE 格式傳回結果，請使用下列範例。

```
SELECT JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]'));
```

```
+-----+
|      json_serialize_to_varbyte      |
+-----+
| 5b31303030312c31303030322c22616263225d |
+-----+
```

若要序列化 SUPER 值並將結果轉換為 VARCHAR 格式，請使用下列範例。如需詳細資訊，請參閱[CAST 函數](#)。

```
SELECT CAST((JSON_SERIALIZE_TO_VARBYTE(JSON_PARSE('[10001,10002,"abc"]')))) AS VARCHAR);
```

```
+-----+
| json_serialize_to_varbyte |
+-----+
| [10001,10002,"abc"]      |
+-----+
```

文字型JSON函數

下列函數會將JSON值剖析為 VARCHAR。對於剖析 JSON，我們建議您改用下列函數，這些函數剖析 JSON值為 SUPER。Amazon Redshift 剖析SUPER值比 更有效率VARCHAR。

- [JSON_PARSE 函數](#)
- [CAN_JSON_PARSE 函數](#)
- [JSON_SERIALIZE 函數](#)
- [JSON_SERIALIZE_TO_VARBYTE 函數](#)

主題

- [IS_VALID_JSON 函數](#)
- [IS_VALID_JSON_ARRAY 函數](#)
- [JSON_ARRAY_LENGTH 函數](#)

- [JSON_EXTRACT_ARRAY_ELEMENT_TEXT 函數](#)
- [JSON_EXTRACT_PATH_TEXT 函數](#)

IS_VALID_JSON 函數

Note

CAN_JSON_PARSE 及其相關聯的函數剖析JSON值為 SUPER，Amazon Redshift 會比 更有效率地剖析值VARCHAR。

建議您使用 驗證JSON字串JSON，而不是使用 IS_VALID_ [CAN_JSON_PARSE 函數](#)。

IS_VALID_JSON 函數會驗證JSON字串。true 如果字串正確形成JSON或字串格式不正確false，函數會傳回布林值。若要驗證JSON陣列，請使用 [IS_VALID_JSON_ARRAY 函數](#)

如需詳細資訊，請參閱[JSON 函數](#)。

語法

```
IS_VALID_JSON('json_string')
```

引數

json_string

評估為字串的JSON字串或運算式。

傳回類型

BOOLEAN

範例

若要建立資料表並插入用於測試的JSON字串，請使用下列範例。

```
CREATE TABLE test_json(id int IDENTITY(0,1), json_strings VARCHAR);

-- Insert valid JSON strings --
INSERT INTO test_json(json_strings) VALUES
```

```

({'a':2}),
({'a':{'b':{'c':1}}}),
({'a': [1,2,"b"]});

-- Insert invalid JSON strings --
INSERT INTO test_json(json_strings) VALUES
('{{}}'),
('{1:"a"}'),
('[1,2,3]');

```

若要驗證上述範例中的字串，請使用下列範例。

```

SELECT id, json_strings, IS_VALID_JSON(json_strings)
FROM test_json
ORDER BY id;

```

id	json_strings	is_valid_json
0	{"a":2}	true
4	{"a":{"b":{"c":1}}}	true
8	{"a": [1,2,"b"]}	true
12	{{}}	false
16	{1:"a"}	false
20	[1,2,3]	false

IS_VALID_JSON_ARRAY 函數

Note

JSON_PARSE 及其相關聯的函數會將值剖析為 SUPER，Amazon Redshift 會比更有效率地剖析這些JSON值VARCHAR。

建議您使用剖析JSON字串[JSON_PARSE 函數](#)以取得SUPER值ARRAY，而不是使用ISVALID__JSON_。然後，使用 [IS_ARRAY 函數](#)函數來確認陣列已正確形成。

IS_VALID_JSON_ARRAY 函數會驗證JSON陣列。true 如果陣列正確形成JSON或陣列格式不正確false，函數會傳回布林值。若要驗證JSON字串，請使用 [IS_VALID_JSON 函數](#)

如需詳細資訊，請參閱[JSON 函數](#)。

語法

```
IS_VALID_JSON_ARRAY('json_array')
```

引數

json_array

評估為JSON陣列的字串或運算式。

傳回類型

BOOLEAN

範例

若要建立資料表並插入用於測試的JSON字串，請使用下列範例。

```
CREATE TABLE test_json_arrays(id int IDENTITY(0,1), json_arrays VARCHAR);

-- Insert valid JSON array strings --
INSERT INTO test_json_arrays(json_arrays)
VALUES('[]'),
(['a","b"]),
(['a',['b',1,['c',2,3,null]]]);

-- Insert invalid JSON array strings --
INSERT INTO test_json_arrays(json_arrays)
VALUES({'a':1}),
('a'),
(['1,2,']);
```

若要驗證上述範例中的字串，請使用下列範例。

```
SELECT json_arrays, IS_VALID_JSON_ARRAY(json_arrays)
FROM test_json_arrays ORDER BY id;
```

json_arrays	is_valid_json_array
[]	true
["a","b"]	true

["a",["b",1,["c",2,3,null]]]	true	
{"a":1}	false	
a	false	
[1,2,]	false	
+-----+	+-----+	

JSON_ARRAY_LENGTH 函數

Note

JSON_PARSE 及其相關聯的函數會將值剖析為 SUPER，Amazon Redshift 會比更有效率地剖析這些JSON值VARCHAR。

建議您使用 剖析JSON字串[JSON_PARSE 函數](#)以取得SUPER值LENGTH，而不是使用JSON_ARRAY_。然後，使用 [get_array_length 陣列](#)取得陣列的長度。

JSON_ARRAY_LENGTH 函數會傳回JSON字串外部陣列中的元素數目。如果 null_if_invalid 引數設定為 true且JSON字串無效，則函數會傳回 NULL，而不是傳回錯誤。

如需詳細資訊，請參閱[JSON 函數](#)。

語法

```
JSON_ARRAY_LENGTH('json_array' [, null_if_invalid ] )
```

引數

json_array

正確格式化的JSON陣列。

null_if_invalid

(選用) BOOLEAN指定輸入JSON字串無效NULL時是否要傳回的值，而不是傳回錯誤。若要在 JSON 無效NULL時傳回 ，請指定 true (t)。若要在 JSON 無效時傳回錯誤，請指定 false (f)。預設值為 false。

傳回類型

INTEGER

範例

若要傳回陣列中的元素個數，請使用下列範例。

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14]');
```

```
+-----+
| json_array_length |
+-----+
|                   5 |
+-----+
```

若要因為 JSON 無效而傳回錯誤，請使用下列範例。

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14]');
```

```
ERROR: invalid json array object [11,12,13,{"f1":21,"f2":[25,26]},14
```

若要將 `null_if_invalid` 設定為 `true`，因此傳回的陳述式 NULL 不會傳回無效的錯誤 JSON，請使用下列範例。

```
SELECT JSON_ARRAY_LENGTH('[11,12,13,{"f1":21,"f2":[25,26]},14', true);
```

```
+-----+
| json_array_length |
+-----+
| NULL              |
+-----+
```

JSON_EXTRACT_ARRAY_ELEMENT_TEXT 函數

Note

JSON_PARSE 及其相關聯的函數會將值剖析為 SUPER，Amazon Redshift 會比更有效率地剖析這些 JSON 值 VARCHAR。

建議您使用剖析 JSON 字串 [JSON_PARSE 函數](#) 以取得 SUPER 值 TEXT，而不是使用 ARRAY_JSONEXTRACT_ELEMENT__。然後，使用 `value[element position]` 語法，使用陣列索引查詢您想要的元素。如需查詢 SUPER 值中陣列元素的詳細資訊，請前往 [查詢半結構化資料](#)。

JSON_EXTRACT_ARRAY_ELEMENT_TEXT函數會使用零索引傳回JSON字串最外部陣列中的JSON陣列元素。陣列的第一個元素在位置 0。如果索引為負或超出範圍，JSON_EXTRACT_ARRAY_ELEMENT_TEXT 會傳回空字串。如果 null_if_invalid 引數設定為 true且JSON字串無效，則函數會傳回 NULL，而不是傳回錯誤。

如需詳細資訊，請參閱[JSON 函數](#)。

語法

```
JSON_EXTRACT_ARRAY_ELEMENT_TEXT('json string', pos [, null_if_invalid ] )
```

引數

json_string

正確格式化的JSON字串。

pos

INTEGER，代表要傳回之陣列元素的索引 (採用以零開始的陣列索引)。

null_if_invalid

(選用) BOOLEAN指定輸入JSON字串無效NULL時是否要傳回的值，而不是傳回錯誤。若要在 JSON 無效NULL時傳回 ，請指定 true (t)。若要在 JSON 無效時傳回錯誤，請指定 false (f)。預設值為 false。

傳回類型

VARCHAR

代表位置 所參考之JSON陣列元素的VARCHAR字串。

範例

若要傳回位置 2 的陣列元素 (即從零開始之陣列索引的第三個元素)，請使用下列範例。

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT(' [111,112,113]', 2);
```

```
+-----+
| json_extract_array_element_text |
+-----+
|                               113 |
+-----+
```

若要因為 JSON 無效而傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('["a",["b",1,["c",2,3,null,]]',1);

ERROR: invalid json array object ["a",["b",1,["c",2,3,null,]]]
```

若要將 `null_if_invalid` 設定為 `true`，因此陳述式傳回，NULL 而不是傳回無效的錯誤 JSON，請使用下列範例。

```
SELECT JSON_EXTRACT_ARRAY_ELEMENT_TEXT('["a",["b",1,["c",2,3,null,]]',1,true);

+-----+
| json_extract_array_element_text |
+-----+
| NULL                             |
+-----+
```

JSON_EXTRACT_PATH_TEXT 函數

Note

JSON_PARSE 及其相關聯的函數會將值剖析為 SUPER，Amazon Redshift 會比更有效率地剖析這些 JSON 值 VARCHAR。

建議您使用剖析 JSON 字串 [JSON_PARSE 函數](#) 以取得 SUPER 值 TEXT，而不是使用 JSONEXTRACT_PATH_。然後，使用 `value.attribute` 語法查詢您想要的元素。如需查詢 SUPER 值中陣列元素的詳細資訊，請前往 [查詢半結構化資料](#)。

JSON_EXTRACT_PATH_TEXT 函數會傳回 JSON 字串中一系列路徑元素所參考之鍵值對的值。JSON 路徑最多可巢狀至五個層級深。路徑元素區分大小寫。如果 JSON 字串中不存在路徑元素，JSON_EXTRACT_PATH_TEXT 會傳回 NULL。

如果 `null_if_invalid` 引數設定為 `true` 且 JSON 字串無效，則函數會傳回 NULL，而不是傳回錯誤。

JSON_EXTRACT_PATH_TEXT 的資料大小上限為 64KB。因此，如果任何JSON記錄大於 64KB，使用 JSON_EXTRACT_PATH_TEXT 處理它會導致錯誤。

如需其他JSON函數的資訊，請參閱 [JSON 函數](#)。如需使用的詳細資訊JSON，請參閱 [COPY 從JSON 格式](#)。

語法

```
JSON_EXTRACT_PATH_TEXT('json_string', 'path_elem' [, 'path_elem'[, ...] ]  
[, null_if_invalid ] )
```

引數

json_string

正確格式化的JSON字串。

path_elem

JSON 字串中的路徑元素。需要一個路徑元素。可指定其他路徑元素，最深可達五層。

null_if_invalid

(選用) BOOLEAN指定輸入JSON字串無效NULL時是否要傳回的值，而不是傳回錯誤。若要在 JSON 無效NULL時傳回，請指定 true (t)。若要在 JSON 無效時傳回錯誤，請指定 false (f)。預設值為 false。

在JSON字串中，Amazon Redshift 會識別\n為新行字元和\t標籤字元。若要載入反斜線，請加上反斜線 (\\) 來逸出。如需詳細資訊，請參閱 [中的逸出字元 JSON](#)。

傳回類型

VARCHAR

表示路徑元素參考JSON值的VARCHAR字串。

範例

若要傳回路徑 'f4'，'f6' 的值，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{ "f2": {"f3": 1}, "f4": {"f5": 99, "f6": "star"} }', 'f4', 'f6');
```

```
+-----+
| json_extract_path_text |
+-----+
| star                    |
+-----+
```

若要因為 JSON 無效而傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4','f6');
```

```
ERROR: invalid json object {"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}
```

若要將 `null_if_invalid` 設定為 `true`，因此陳述式會傳回 NULL 無效，JSON 而不是傳回錯誤，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}','f4',
'f6',true);
```

```
+-----+
| json_extract_path_text |
+-----+
| NULL                    |
+-----+
```

若要傳回路徑 `'farm'`、`'barn'`、`'color'` 的值，其中擷取的值位於第三層，請使用下列範例。此範例使用 lint JSON 工具格式化，以便於讀取。

```
SELECT JSON_EXTRACT_PATH_TEXT('{
  "farm": {
    "barn": {
      "color": "red",
      "feed stocked": true
    }
  }
}','farm','barn','color');
```

```
+-----+
| json_extract_path_text |
+-----+
| red                    |
+-----+
```

若要在遺失 'color' 元素時傳回 NULL，請使用下列範例。此範例使用 lint JSON 工具格式化。

```
SELECT JSON_EXTRACT_PATH_TEXT('{
  "farm": {
    "barn": {}
  }
}', 'farm', 'barn', 'color');
```

```
+-----+
| json_extract_path_text |
+-----+
| NULL                    |
+-----+
```

如果 JSON 有效，嘗試擷取缺少的元素會傳回 NULL。

若要傳回路徑 'house', 'appliances', 'washing machine', 'brand' 的值，請使用下列範例。

```
SELECT JSON_EXTRACT_PATH_TEXT('{
  "house": {
    "address": {
      "street": "123 Any St.",
      "city": "Any Town",
      "state": "FL",
      "zip": "32830"
    },
    "bathroom": {
      "color": "green",
      "shower": true
    },
    "appliances": {
      "washing machine": {
        "brand": "Any Brand",
        "color": "beige"
      },
      "dryer": {
        "brand": "Any Brand",
        "color": "white"
      }
    }
  }
}', 'house', 'appliances', 'washing machine', 'brand');
```



```
+-----+
| json_extract_path_text |
+-----+
| Any Brand              |
+-----+
```

下列範例會建立範例資料表，並填入SUPER值，然後傳回兩個資料列 'f2' 路徑的值。

```
CREATE TABLE json_example(id INT, json_text SUPER);

INSERT INTO json_example VALUES
(1, JSON_PARSE('{"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}')),
(2, JSON_PARSE('{
  "farm": {
    "barn": {
      "color": "red",
      "feed stocked": true
    }
  }
}')));

SELECT * FROM json_example;
id          | json_text
-----+-----
1           | {"f2":{"f3":1},"f4":{"f5":99,"f6":"star"}}
2           | {"farm":{"barn":{"color":"red","feed stocked":true}}}}

SELECT id, JSON_EXTRACT_PATH_TEXT(JSON_SERIALIZE(json_text), 'f2') FROM json_example;

id          | json_text
-----+-----
1           | {"f3":1}
2           |
```

機器學習函數

透過使用 Amazon Redshift 機器學習 (ML)，您可以使用SQL陳述式訓練 ML 模型，並在SQL查詢中叫用它們以進行預測。Amazon Redshift 模型可解釋性包含特徵重要性值，可協助您了解訓練資料中的每個屬性如何影響預測結果。

接下來，您可以找到SQL該 Amazon Redshift 支援的機器學習函數的說明。

主題

- [EXPLAIN_MODEL 函數](#)

EXPLAIN_MODEL 函數

EXPLAIN_MODEL 函數會傳回包含 JSON 格式模型可解釋性報告的SUPER資料類型。可解釋性報告包含有關所有模型特徵的 Shapley 值的資訊。

EXPLAIN_MODEL 函數目前僅支援 AUTO ON 或 AUTO OFF XGBoost 模型。

當可解釋性報告不可用時，函數會傳回顯示模型進度的狀態。其中包括 Waiting for training job to complete、Waiting for processing job to complete 和 Processing job failed。

當您執行CREATEMODEL陳述式時，解釋狀態會變成 Waiting for training job to complete。當模型經過訓練並傳送解釋請求時，解釋狀態會變成 Waiting for processing job to complete。當模型解釋成功完成後，即可獲得完整的可解釋性報告。否則狀態變成 Processing job failed。

當您執行CREATEMODEL陳述式時，您可以使用選用MAX_RUNTIME參數來指定訓練應花費的時間上限。一旦模型建立達到這段時間，Amazon Redshift 就會停止建立模型。如果您在建立自動駕駛模型時達到該時間限制，Amazon Redshift 將傳回目前為止最佳模型。模型訓練完成後，模型可解釋性就可用，因此如果 MAX_RUNTIME 設定的時間較短，則可解釋性報告可能無法使用。訓練時間會因模型複雜度、資料大小及其他因素而有所不同。

語法

```
EXPLAIN_MODEL ('schema_name.model_name')
```

引數

`schema_name`

結構描述的名稱。如果未指定 `schema_name`，則會選取目前的結構描述。

`model_name`

模型的名稱。結構描述中的模型名稱必須是唯一的。

傳回類型

EXPLAIN_MODEL 函數會傳回SUPER資料類型，如下所示。

```
{"version":"1.0","explanations":{"kernel_shap":{"label0":{"global_shap_values":{"x0":0.05,"x1":0.10,"x2":0.30,"x3":0.15},"expected_value":0.50}}}}
```

範例

下列範例會傳回解釋狀態 waiting for training job to complete。

```
select explain_model('customer_churn_auto_model');
           explain_model
-----
{"explanations":"waiting for training job to complete"}
(1 row)
```

當模型解釋成功完成時，完整的可解釋性報告如下。

```
select explain_model('customer_churn_auto_model');
           explain_model
-----
{"version":"1.0","explanations":{"kernel_shap":{"label0":{"global_shap_values":{"x0":0.05386043365892927,"x1":0.10801289723274592,"x2":0.23227865827017378,"x3":0.067668513394},"expected_value":0.50}}}}
(1 row)
```

由於 EXPLAIN_MODEL 函數傳回SUPER資料類型，因此您可以查詢可解釋性報告。透過這樣做，您可以擷取 global_shap_values、expected_value 或特定於特徵的 Shapley 值。

以下範例會擷取模型的 global_shap_values。

```
select json_table.report.explanations.kernel_shap.label0.global_shap_values from
(select explain_model('customer_churn_auto_model') as report) as json_table;
           global_shap_values
-----
{"state":0.10983770427197151,"account_length":0.1772441398408543,"area_code":0.0862682396863959}
(1 row)
```

以下範例會擷取特徵 x0 的 global_shap_values。

```
select json_table.report.explanations.kernel_shap.label0.global_shap_values.x0 from
(select explain_model('customer_churn_auto_model') as report) as json_table;
```

```

          x0
-----
 0.05386043365892927
(1 row)

```

如果模型是在一個特定的結構描述中建立的，且您有權存取建立的模型，那麼您可以查詢模型解釋，如下所示。

```

-- Check the current schema
SHOW search_path;
  search_path
-----
 $user, public
(1 row)
-- If you have the privilege to access the model explanation
-- in `test_schema`
SELECT explain_model('test_schema.test_model_name');
          explain_model
-----
 {"explanations":"waiting for training job to complete"}
(1 row)

```

數學函數

主題

- [數學運算子符號](#)
- [ABS 函數](#)
- [ACOS 函數](#)
- [ASIN 函數](#)
- [ATAN 函數](#)
- [ATAN2 函數](#)
- [CBRT 函數](#)
- [CEILING \(或 CEIL \) 函數](#)
- [COS 函數](#)
- [COT 函數](#)
- [DEGREES 函數](#)
- [DEXP 函數](#)

- [DLOG1 函數](#)
- [DLOG10 函數](#)
- [EXP 函數](#)
- [FLOOR 函數](#)
- [LN 函數](#)
- [LOG 函數](#)
- [MOD 函數](#)
- [PI 函數](#)
- [POWER 函數](#)
- [RADIANS 函數](#)
- [RANDOM 函數](#)
- [ROUND 函數](#)
- [SIN 函數](#)
- [SIGN 函數](#)
- [SQRT 函數](#)
- [TAN 函數](#)
- [TRUNC 函數](#)

本節描述 Amazon Redshift 中支援的數學運算子和函數。

數學運算子符號

下表列出支援的數學運算子。

支援的運算子

運算子	描述	範例	結果
+	加法	2 + 3	5
-	減法	2 - 3	-1
*	乘法	2 * 3	6
/	除法	4 / 2	2

運算子	描述	範例	結果
%	模數	5 % 4	1
^	指數	2.0 ^ 3.0	8
/	平方根	/ 25.0	5
/	立方根	/ 27.0	3
@	絕對值	@ -5.0	5
<<	位元左移	1 << 4	16
>>	位元右移	8 >> 2	2
&	位元 and	8 & 2	0

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要計算給定交易的已付佣金加 2.00 USD 手續費，請使用下列範例。

```
SELECT
    commission,
    (commission + 2.00) AS comm
FROM
    sales
WHERE
    salesid = 10000;
```

```
+-----+-----+
| commission | comm |
+-----+-----+
|      28.05 | 30.05 |
+-----+-----+
```

若要計算給定交易售價的 20%，請使用下列範例。

```
SELECT pricepaid, (pricepaid * .20) as twentypct
```

```
FROM sales
WHERE salesid=10000;
```

```
+-----+-----+
| pricepaid | twentypct |
+-----+-----+
|      187 |      37.4 |
+-----+-----+
```

若要使用 DEXP 函數以根據持續成長模式來預測門票銷售，請使用下列範例。在此範例中，子查詢傳回 2008 年銷售的門票數。此結果以指數方式乘以過去 10 年的持續成長率 5%。

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid AND year=2008)^(5::float/100)*10) AS qty10years;
```

```
+-----+
| qty10years |
+-----+
| 587.664019657491 |
+-----+
```

若要尋找日期 ID 大於或等於 2000 之銷售的支付總價和佣金，請使用下列範例。然後從支付總價中減去佣金總計。

```
SELECT SUM(pricepaid) AS sum_price, dateid,
SUM(commission) AS sum_comm, (SUM(pricepaid) - SUM(commission)) AS value
FROM sales
WHERE dateid >= 2000
GROUP BY dateid
ORDER BY dateid
LIMIT 10;
```

```
+-----+-----+-----+-----+
| sum_price | dateid | sum_comm | value |
+-----+-----+-----+-----+
| 305885 | 2000 | 45882.75 | 260002.25 |
| 316037 | 2001 | 47405.55 | 268631.45 |
| 358571 | 2002 | 53785.65 | 304785.35 |
| 366033 | 2003 | 54904.95 | 311128.05 |
| 307592 | 2004 | 46138.8 | 261453.2 |
| 333484 | 2005 | 50022.6 | 283461.4 |
| 317670 | 2006 | 47650.5 | 270019.5 |
```

```
| 351031 | 2007 | 52654.65 | 298376.35 |
| 313359 | 2008 | 47003.85 | 266355.15 |
| 323675 | 2009 | 48551.25 | 275123.75 |
+-----+-----+-----+-----+
```

ABS 函數

ABS 會計算數字的絕對值，其中數字可以是常值或評估為數字的表達式。

語法

```
ABS(number)
```

引數

number

數字或評估為數字的表達式。它可以是

SMALLINT、INTEGER、BIGINT、DECIMAL、FLOAT4、FLOAT8 或 SUPER 類型。

傳回類型

ABS 會傳回與其引數相同的資料類型。

範例

若要計算 -38 的絕對值，請使用下列範例。

```
SELECT ABS(-38);
```

```
+-----+
| abs |
+-----+
| 38 |
+-----+
```

若要計算 (14-76) 的絕對值，請使用下列範例。

```
SELECT ABS(14-76);
```



```
+-----+
| abs |
+-----+
| 62 |
+-----+
```

ACOS 函數

ACOS 是三角形函數，會傳回數字的弧餘弦。傳回值為弧度且介於 0 和 PI 之間。

語法

```
ACOS(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 -1 的反餘弦，請使用下列範例。

```
SELECT ACOS(-1);
```

```
+-----+
|      acos      |
+-----+
| 3.141592653589793 |
+-----+
```

若要將 .5 的反餘弦轉換為同等度數，請使用下列範例。

```
SELECT (ACOS(.5) * 180/(SELECT PI())) AS degrees;
```

```
+-----+
|      degrees      |
+-----+
| 60.00000000000001 |
+-----+
```

ASIN 函數

ASIN 是三角形函數，會傳回數字的弧正弦。傳回值為弧度且介於 $\pi/2$ 和 $-\pi/2$ 之間。

語法

```
ASIN(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的正弦，請使用下列範例。

```
SELECT ASIN(1) AS halfpi;
```

```
+-----+
|      halfpi      |
+-----+
| 1.5707963267948966 |
+-----+
```

若要將 .5 的反正弦轉換為同等度數，請使用下列範例。

```
SELECT (ASIN(.5) * 180/(SELECT PI())) AS degrees;
```

```
+-----+
|      degrees      |
+-----+
| 30.000000000000004 |
+-----+
```

ATAN 函數

ATAN 是三角形函數，會傳回數字的弧切線。傳回值為弧度且介於 $-\pi$ 和 π 之間。

語法

```
ATAN(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的反正切再乘以 4，請使用下列範例。

```
SELECT ATAN(1) * 4 AS pi;
```

```
+-----+
|      pi      |
+-----+
| 3.141592653589793 |
+-----+
```

若要將 1 的反正切轉換為同等度數，請使用下列範例。

```
SELECT (ATAN(1) * 180/(SELECT PI())) AS degrees;
```

```
+-----+
```

```
| degrees |
+-----+
|      45 |
+-----+
```

ATAN2 函數

ATAN2 是三角形函數，傳回一個數字的弧切線除以另一個數字。傳回值為弧度且介於 $\text{PI}/2$ 和 $-\text{PI}/2$ 之間。

語法

```
ATAN2(number1, number2)
```

引數

number1

DOUBLE PRECISION 數字。

number2

DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的反正切再乘以 4，請使用下列範例。

```
SELECT ATAN2(2,2) * 4 AS PI;
```

```
+-----+
|      pi      |
+-----+
| 3.141592653589793 |
+-----+
```

若要將 1 的的反正切 (計算結果為 0) 轉換為同等度數，請使用下列範例。

```
SELECT (ATAN2(1,0) * 180/(SELECT PI())) AS degrees;
```

```
+-----+
| degrees |
+-----+
|      90 |
+-----+
```

CBRT 函數

CBRT 函數是數學函數，可計算指定數字的立方根。

語法

```
CBRT(number)
```

引數

CBRT 會將DOUBLE PRECISION數字作為引數。

傳回類型

DOUBLE PRECISION

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要計算給定交易之已付佣金的立方根，請使用下列範例。

```
SELECT CBRT(commission) FROM sales WHERE salesid=10000;
```

```
+-----+
|      cbrt      |
+-----+
| 3.0383953904884344 |
+-----+
```

CEILING (或 CEIL) 函數

CEILING 或 CEIL函數用於將數字四捨五入到下一個整數。([FLOOR 函數](#) 將數字捨去到下一個整數。)

語法

```
{CEIL | CEILING}(number)
```

引數

number

數字或評估為數字的運算式。它可以是 SMALLINT、INTEGER、BIGINT、DECIMAL、FLOAT4、FLOAT8 或 SUPER 類型。

傳回類型

CEILING 並 CEIL 傳回與其引數相同的資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型會保留該 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 null。

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 [範本資料庫](#)。

若要計算給定銷售交易之已付佣金的上限，請使用下列範例。

```
SELECT CEILING(commission) FROM sales
WHERE salesid=10000;
```

```
+-----+
| ceiling |
+-----+
|      29 |
+-----+
```

COS 函數

COS 是傳回數字餘弦的三角函數。傳回值為弧度且介於 -1 和 1 (含) 之間。

語法

```
COS(double_precision)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

COS 函數會傳回DOUBLE PRECISION數字。

範例

若要傳回 0 的餘弦，請使用下列範例。

```
SELECT COS(0);
```

```
+-----+
|  COS  |
+-----+
|    1  |
+-----+
```

若要傳回 pi 的餘弦，請使用下列範例。

```
SELECT COS(PI());
```

```
+-----+
|  COS  |
+-----+
|   -1  |
+-----+
```

COT 函數

COT 是三角形函數，會傳回數字的餘切。輸入參數必須不是零。

語法

```
COT(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 1 的餘切，請使用下列範例。

```
SELECT COT(1);

+-----+
|      cot      |
+-----+
| 0.6420926159343306 |
+-----+
```

DEGREES 函數

將角度的弧度轉換為其同等度數。

語法

```
DEGREES(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 0.5 弧度的同等度數，請使用下列範例。

```
SELECT DEGREES(.5);

+-----+
| degrees |
+-----+
| 28.64788975654116 |
+-----+
```

若要將 PI 弧度轉換為度數，請使用下列範例。

```
SELECT DEGREES(pi());

+-----+
| degrees |
+-----+
| 180 |
+-----+
```

DEXP 函數

DEXP 函數會傳回雙精度數字的科學符號指數值。DEXP 和 EXP 函數之間的唯一區別在於的參數 DEXP 必須是 DOUBLE PRECISION。

語法

```
DEXP(number)
```

引數

number

輸入參數是 DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

使用 DEXP函數根據持續成長模式預測票證銷售。在此範例中，子查詢傳回 2008 年銷售的門票數。該結果乘以DEXP函數的結果，該函數指定 10 年期間的持續成長率為 7%。

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * DEXP((7::FLOAT/100)*10) qty2010;
```

```
+-----+
|      qty2010      |
+-----+
| 695447.4837722216 |
+-----+
```

DLOG1 函數

DLOG1 函數會傳回輸入參數的自然對數。[LN 函數](#) 的同義詞。

DLOG10 函數

DLOG10 傳回輸入參數的基礎 10 對數。

[LOG 函數](#) 的同義詞。

語法

```
DLOG10(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回數字 100 的以 10 為底的對數，請使用下列範例。

```
SELECT DLOG10(100);
```

```
+-----+
| dlog10 |
+-----+
|      2 |
+-----+
```

EXP 函數

EXP 函數會實作指數函數，以 e 求表達能力提升的數值表達式或自然對數的基礎。EXP 函數是的反轉 [LN 函數](#)。

語法

```
EXP(expression)
```

引數

運算式

運算式必須是 INTEGER、DECIMAL 或 DOUBLE PRECISION 資料類型。

傳回類型

DOUBLE PRECISION

範例

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 [範本資料庫](#)。

使用 EXP 函數根據持續成長模式預測票證銷售。在此範例中，子查詢傳回 2008 年銷售的門票數。該結果乘以 EXP 函數的結果，該結果指定 10 年期間的持續成長率為 7%。

```
SELECT (SELECT SUM(qtysold)
FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * EXP((7::FLOAT/100)*10) qty2018;
```

```
+-----+
|      qty2018      |
+-----+
| 695447.4837722216 |
+-----+
```

FLOOR 函數

FLOOR 函數會將數字四捨五入到下一個整數。

語法

```
FLOOR(number)
```

引數

number

數字或評估為數字的運算式。它可以是

SMALLINT、INTEGER、BIGINT、DECIMAL、FLOAT4、FLOAT8 或 SUPER 類型。

傳回類型

FLOOR 會傳回與其引數相同的資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要顯示使用 FLOOR函數之前和之後為指定銷售交易支付的佣金值，請使用下列範例。

```
SELECT commission
FROM sales
WHERE salesid=10000;

+-----+
| commission |
+-----+
|      28.05 |
```

```
+-----+
SELECT FLOOR(commission)
FROM sales
WHERE salesid=10000;
```

```
+-----+
| floor |
+-----+
|    28 |
+-----+
```

LN 函數

傳回輸入參數的自然對數。

[DLOG1 函數](#) 的同義詞。

語法

```
LN(expression)
```

引數

運算式

函數運算的目標欄或表達式。

Note

如果運算式參考 Amazon Redshift 使用者建立的資料表或 Amazon Redshift STL或STV系統資料表，則此函數會傳回某些資料類型的錯誤。

如果具有下列資料類型的表達式參考使用者建立的資料表或系統資料表，則會產生錯誤。具有這些資料類型的表達式只能在領導者節點上執行：

- BOOLEAN
- CHAR
- DATE
- DECIMAL 或 NUMERIC

- TIMESTAMP
- VARCHAR

具有下列資料類型的表達式會在使用者建立的資料表和STL/或STV系統資料表上成功執行：

- BIGINT
- DOUBLE PRECISION
- INTEGER
- REAL
- SMALLINT

傳回類型

LN 函數傳回與輸入運算式相同的類型。

範例

若要傳回數字 2.718281828 的自然對數或以 e 為底的對數，請使用下列範例。

```
SELECT LN(2.718281828);
```

```
+-----+
|          ln          |
+-----+
| 0.999999998311267 |
+-----+
```

請注意，答案幾乎等於 1。

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要傳回USERS資料表中使用者 ID 欄中值的自然對數，請使用下列範例。

```
SELECT username, LN(userid) FROM users ORDER BY userid LIMIT 10;
```

```
+-----+-----+
| username |          ln          |
+-----+-----+
| JSG99FHE |                   0 |
| PGL08LJI | 0.6931471805599453 |
| IFT66TXU | 1.0986122886681098 |
| XDZ38RDD | 1.3862943611198906 |
```

```
| AEB55QTM | 1.6094379124341003 |
| NDQ15VBM | 1.791759469228055 |
| OWY35QYB | 1.9459101490553132 |
| AZG78YIP | 2.0794415416798357 |
| MSD36KVR | 2.1972245773362196 |
| WKW41AIW | 2.302585092994046 |
+-----+-----+
```

LOG 函數

傳回數字的對數。

如果您使用此函數計算以 10 為底的對數，則也可以使用 [DLOG10 函數](#)。

語法

```
LOG([base, ]argument)
```

參數

base

(選用) 對數函數的底。這個數字必須是正數且不能等於 1。如果省略此參數，Amazon Redshift 會計算 *argument* 的以 10 為底的對數。

argument

對數函數的引數。此數字必須為正數。如果 *argument* 值是 1，則函數傳回 0。

傳回類型

LOG 函數會傳回DOUBLE PRECISION數字。

範例

若要尋找 100 的以 2 為底的對數，請使用下列範例。

```
SELECT LOG(2, 100);
+-----+
|      log      |
+-----+
| 6.643856189774725 |
+-----+
```

若要尋找 100 的以 10 為底的對數，請使用下列範例。請注意，如果您省略基本參數，Amazon Redshift 假設基數為 10。

```
SELECT LOG(100);
```

```
+-----+
| log |
+-----+
|  2  |
+-----+
```

MOD 函數

傳回兩個數字的餘數，也稱為模數運算。為了計算結果，第一個參數除以第二個參數。

語法

```
MOD(number1, number2)
```

引數

number1

第一個輸入參數是 INTEGER、SMALLINT、BIGINT 或 DECIMAL 數字。如果任一參數為 DECIMAL 類型，則另一個參數也必須為 DECIMAL 類型。如果任一參數為 INTEGER，則另一個參數可以是 INTEGER、SMALLINT 或 BIGINT。兩個參數也都可以是 SMALLINT 或 BIGINT，但如果一個參數是 BIGINT，則另一個參數不能是 SMALLINT。

number2

第二個參數是 INTEGER、SMALLINT、BIGINT 或 DECIMAL 數字。相同的資料類型規則適用於 number2 與 number1。

傳回類型

如果兩個輸入參數都是相同的類型，則 MOD 函數的傳回類型與輸入參數的數值類型相同。不過，如果任一輸入參數為 INTEGER，則傳回類型也會是 INTEGER。有效的傳回類型為 DECIMAL、INT、SMALLINT 和 BIGINT。

使用須知

您可以使用 % 作為模數運算子。

範例

若要傳回一個數字除以另一個數字時的餘數，請使用下列範例。

```
SELECT MOD(10, 4);
```

```
+-----+
| mod   |
+-----+
|    2  |
+-----+
```

若要在使用 MOD 函數時傳回 DECIMAL 結果，請使用下列範例。

```
SELECT MOD(10.5, 4);
```

```
+-----+
| mod   |
+-----+
| 2.5   |
+-----+
```

若要在執行 MOD 函數之前投放數字，請使用下列範例。如需詳細資訊，請參閱 [CAST 函數](#)。

```
SELECT MOD(CAST(16.4 AS INTEGER), 5);
```

```
+-----+
| mod   |
+-----+
|    1  |
+-----+
```

若要透過除以 2 來檢查第一個參數是否為偶數，請使用下列範例。

```
SELECT mod(5,2) = 0 AS is_even;
```

```
+-----+
| is_even |
+-----+
| false   |
+-----+
```

若要使用 % 做為模數運算子，請使用下列範例。

```
SELECT 11 % 4 as remainder;
```

```
+-----+
| remainder |
+-----+
|          3 |
+-----+
```

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要傳回CATEGORY資料表中奇數類別的資訊，請使用下列範例。

```
SELECT catid, catname
FROM category
WHERE MOD(catid,2)=1
ORDER BY 1,2;
```

```
+-----+-----+
| catid | catname |
+-----+-----+
|      1 | MLB     |
|      3 | NFL     |
|      5 | MLS     |
|      7 | Plays   |
|      9 | Pop     |
|     11 | Classical |
+-----+-----+
```

PI 函數

PI 函數會傳回 pi 的值，精確到 14 位小數。

語法

```
PI()
```

傳回類型

DOUBLE PRECISION

範例

若要傳回 pi 的值，請使用下列範例。

```
SELECT PI();

+-----+
|      pi      |
+-----+
| 3.141592653589793 |
+-----+
```

POWER 函數

POWER 函數是指數函數，可將數值表達式提升為第二個數值表達式的威力。例如，2 的三次方以 POWER(2,3) 計算，結果為 8。

語法

```
{POW | POWER}(expression1, expression2)
```

引數

expression1

要乘以次方的數值表達式。必須是 INTEGER、DECIMAL 或 FLOAT 資料類型。

expression2

expression1 要乘以的次方。必須是 INTEGER、DECIMAL 或 FLOAT 資料類型。

傳回類型

DOUBLE PRECISION

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

在下列範例中，該POWER函數會根據 2008 年售出的門票數量（子查詢的結果），用來預測未來 10 年的門票銷售情況。在此範例中，成長率設為每年 7%。

```
SELECT (SELECT SUM(qtysold) FROM sales, date
```

```
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100),10) qty2010;
```

```
+-----+
|      qty2010      |
+-----+
| 679353.7540885945 |
+-----+
```

下列範例是上一個範例的變化版，成長率每年 7%，但間隔設為月份 (10 年共 120 個月)。

```
SELECT (SELECT SUM(qtysold) FROM sales, date
WHERE sales.dateid=date.dateid
AND year=2008) * POW((1+7::FLOAT/100/12),120) qty2010;
```

```
+-----+
|      qty2010      |
+-----+
| 694034.54678046   |
+-----+
```

RADIANS 函數

RADIANS 函數會以弧度將角度轉換為其等效值。

語法

```
RADIANS(number)
```

引數

number

輸入參數是DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 180 度的同等弧度，請使用下列範例。

```
SELECT RADIANS(180);
```

```
+-----+
| radians |
+-----+
| 3.141592653589793 |
+-----+
```

RANDOM 函數

RANDOM 函數會產生介於 0.0 (含) 和 1.0 (專屬) 之間的隨機值。

語法

```
RANDOM()
```

傳回類型

DOUBLE PRECISION

使用須知

使用 [SET](#) 命令設定種子值RANDOM後呼叫 RANDOM ，以導致 以可預測的順序產生數字。

範例

若要計算 0 到 99 之間的隨機值，請使用下列範例。如果隨機數字為 0 到 1，此查詢會產生 0 到 100 的隨機數字。

```
SELECT CAST(RANDOM() * 100 AS INT);
```

```
+-----+
| int4 |
+-----+
| 59 |
+-----+
```

此範例使用 [SET](#) 命令來設定SEED值，以便 RANDOM產生可預測的數字序列。

若要傳回三個RANDOM整數而不設定SEED值，請使用下列範例。

```
SELECT CAST(RANDOM() * 100 AS INT);
```

```
+-----+
| int4 |
+-----+
|    6 |
+-----+

SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|   68 |
+-----+

SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|   56 |
+-----+
```

若要將 SEED 值設定為 .25，並傳回三個RANDOM數字，請使用下列範例。

```
SET SEED TO .25;
SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|   21 |
+-----+

SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|   79 |
+-----+

SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|   12 |
+-----+
```

若要將 SEED 值重設為 .25，並確認 RANDOM 傳回與前三次呼叫相同的結果，請使用下列範例。

```

SET SEED TO .25;
SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|  21 |
+-----+

SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|  79 |
+-----+

SELECT CAST(RANDOM() * 100 AS INT);
+-----+
| int4 |
+-----+
|  12 |
+-----+

```

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 [範本資料庫](#)。

若要從 SALES 資料表中擷取 10 個項目的統一隨機範例，請使用下列範例。

```

SELECT *
FROM sales
ORDER BY RANDOM()
LIMIT 10;

+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| salesid | listid | sellerid | buyerid | eventid | dateid | qty sold | pricepaid |
| commission | saletime |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| 45422 | 51114 | 5983 | 24482 | 4369 | 2118 | 1 | 195 |
| 29.25 | 2008-10-19 05:20:07 |
| 42481 | 47638 | 4573 | 6198 | 6479 | 1987 | 4 | 1140 |
| 171 | 2008-06-10 09:39:19 |

```

```

| 31494 | 34759 | 18895 | 4719 | 7753 | 2090 | 4 | 1024 |
153.6 | 2008-09-21 03:44:26 |
| 119388 | 136685 | 21815 | 41905 | 2071 | 1884 | 1 | 359 |
53.85 | 2008-02-27 10:43:10 |
| 166990 | 225037 | 18529 | 7628 | 746 | 2113 | 1 | 2009 |
301.35 | 2008-10-14 10:07:44 |
| 11146 | 12096 | 42685 | 6619 | 1876 | 2123 | 1 | 29 |
4.35 | 2008-10-24 06:23:54 |
| 148537 | 172056 | 15102 | 11787 | 6122 | 1923 | 2 | 480 |
72 | 2008-04-07 03:58:23 |
| 68945 | 78387 | 7359 | 18323 | 6636 | 1910 | 1 | 457 |
68.55 | 2008-03-25 08:31:03 |
| 52796 | 59576 | 9909 | 15102 | 7958 | 1951 | 1 | 479 |
71.85 | 2008-05-05 02:25:08 |
| 90684 | 103522 | 38052 | 21549 | 7384 | 2117 | 1 | 313 |
46.95 | 2008-10-18 05:43:11 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+

```

若要擷取 10 個商品的隨機樣本，但請依價格比例來選擇商品，請使用下列範例。例如，一個商品的價格如果是其他商品的兩倍，則出現在查詢結果的機率也是其他商品的兩倍。

```

SELECT *
FROM sales
ORDER BY -LOG(RANDOM()) / pricepaid
LIMIT 10;

+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| salesid | listid | sellerid | buyerid | eventid | dateid | qty sold | pricepaid |
commission | saletime |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+
| 158340 | 208208 | 17082 | 42018 | 1211 | 2160 | 4 | 6852 |
1027.8 | 2008-11-30 12:21:43 |
| 53250 | 60069 | 12644 | 7066 | 7942 | 1838 | 4 | 1528 |
229.2 | 2008-01-12 11:24:56 |
| 22929 | 24938 | 47314 | 6503 | 179 | 2000 | 3 | 741 |
111.15 | 2008-06-23 08:04:50 |
| 164980 | 221181 | 1949 | 19670 | 1471 | 1906 | 1 | 1330 |
199.5 | 2008-03-21 07:59:51 |
| 159641 | 211179 | 44897 | 16652 | 7458 | 2128 | 1 | 1019 |
152.85 | 2008-10-29 02:02:15 |

```



```

| 73143 | 83439 | 5716 | 5727 | 7314 | 1903 | 1 | 248 |
37.2 | 2008-03-18 11:07:42 |
| 84778 | 96749 | 46608 | 32980 | 3883 | 1999 | 2 | 958 |
143.7 | 2008-06-22 12:13:31 |
| 171096 | 232929 | 43683 | 8536 | 8353 | 1870 | 1 | 929 |
139.35 | 2008-02-13 01:36:36 |
| 74212 | 84697 | 39809 | 15569 | 5525 | 2105 | 2 | 896 |
134.4 | 2008-10-06 11:47:50 |
| 158011 | 207556 | 25399 | 16881 | 232 | 2088 | 2 | 2526 |
378.9 | 2008-09-19 06:00:26 |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+

```

ROUND 函數

ROUND 函數會將數字四捨五入到最接近的整數或小數。

ROUND 函數可以選擇性地包含第二個引數作為 INTEGER，以指出四捨五入的小數位數。當您未提供第二個引數時，函數會四捨五入為最接近的整數。指定第二個引數 integer 時，函數會捨入為具有 integer 小數位數的最接近的數字。

語法

```
ROUND(number [ , integer ] )
```

引數

number

數字或評估為數字的運算式。它可以是 DECIMAL、FLOAT8 或 SUPER 類型。Amazon Redshift 可以隱含轉換其他數字資料類型。

integer

(選用) INTEGER，表示沿任一方向捨入的小數位數。此引數不支援 SUPER 資料類型。

傳回類型

ROUND 會傳回與輸入號碼 相同的數值資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要將給定交易的已付佣金四捨五入至最接近的整數，請使用下列範例。

```
SELECT commission, ROUND(commission)
FROM sales WHERE salesid=10000;
```

```
+-----+-----+
| commission | round |
+-----+-----+
|      28.05 |    28 |
+-----+-----+
```

若要將給定交易的已付佣金四捨五入至第一位小數，請使用下列範例。

```
SELECT commission, ROUND(commission, 1)
FROM sales WHERE salesid=10000;
```

```
+-----+-----+
| commission | round |
+-----+-----+
|      28.05 |   28.1 |
+-----+-----+
```

若要以上一個範例的反方向延伸精確度，請使用下列範例。

```
SELECT commission, ROUND(commission, -1)
FROM sales WHERE salesid=10000;
```

```
+-----+-----+
| commission | round |
+-----+-----+
|      28.05 |    30 |
+-----+-----+
```

SIN 函數

SIN 是三角形函數，可傳回數字的正弦。傳回值介於 -1 和 1 之間。

語法

```
SIN(number)
```

引數

number

以弧度表示的 DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 $-\pi$ 的正弦，請使用下列範例。

```
SELECT SIN(-PI());
```

```
+-----+
|          sin          |
+-----+
| -0.000000000000000012246 |
+-----+
```

SIGN 函數

SIGN 函數會傳回數字的符號（正或負）。如果引數為正數、-1如果引數為負數，或0如果引數為0，則SIGN函數的結果為0。

語法

```
SIGN(number)
```

引數

number

數字或評估為數字的表達式。它可以是 DECIMAL、FLOAT8 或 SUPER 類型。Amazon Redshift 可以根據隱含轉換規則轉換其他資料類型。

傳回類型

SIGN 會傳回與輸入引數相同的數值資料類型。如果輸入是 DECIMAL，則輸出為 DECIMAL(1,0)。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列範例顯示表格 t2 中的 d 欄具有 DOUBLE PRECISION 類型，因為輸入是 DOUBLE PRECISION，而表格 t2 中的 n 欄具有 NUMERIC(1,0) 輸出，因為輸入是 NUMERIC。

```
CREATE TABLE t1(d DOUBLE PRECISION, n NUMERIC(12, 2));
INSERT INTO t1 VALUES (4.25, 4.25), (-4.25, -4.25);
CREATE TABLE t2 AS SELECT SIGN(d) AS d, SIGN(n) AS n FROM t1;
SELECT table_name, column_name, data_type FROM SVV_REDSHIFT_COLUMNS WHERE
  table_name='t1' OR table_name='t2';
```

table_name	column_name	data_type
t1	d	double precision
t1	n	numeric(12,2)
t2	d	double precision
t2	n	numeric(1,0)
t1	col1	character varying(20)

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 [範本資料庫](#)。

若要從 SALES 資料表判斷指定交易支付的佣金符號，請使用下列範例。

```
SELECT commission, SIGN(commission)
FROM sales WHERE salesid=10000;
```

commission	sign
28.05	1

SQRT 函數

SQRT 函數會傳回NUMERIC值的平方根。平方根是一個數字與其自身相乘以獲得給定值。

語法

```
SQRT(expression)
```

引數

運算式

運算式必須具有 INTEGER、DECIMAL 或 FLOAT 資料類型，或隱含轉換為這些資料類型的資料類型。expression 可以包含函數。

傳回類型

DOUBLE PRECISION

範例

若要傳回 16 的平方根，請使用下列範例。

```
SELECT SQRT(16);
```

```
+-----+  
| sqrt |  
+-----+  
|    4 |  
+-----+
```

若要使用隱含類型轉換傳回字串 16 的平方根，請使用下列範例。

```
SELECT SQRT('16');
```

```
+-----+  
| sqrt |  
+-----+  
|    4 |  
+-----+
```

若要在使用 ROUND 函數後傳回 16.4 的平方根，請使用下列範例。

```
SELECT SQRT(ROUND(16.4));
```

```
+-----+
|  sqrt  |
+-----+
|    4   |
+-----+
```

若要在指定圓的面積時傳回半徑的長度，請使用下列範例。例如，當以平方英吋為單位指定面積時，它會以英吋為單位計算半徑。樣本中的面積為 20。

```
SELECT SQRT(20/PI()) AS radius;
```

```
+-----+
|      radius      |
+-----+
| 2.5231325220201604 |
+-----+
```

下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 [範本資料庫](#)。

若要傳回 SALES 資料表中 COMMISSION 值的平方根，請使用下列範例。資料 COMMISSION 欄是資料 DECIMAL 欄。此範例顯示如何在具有更複雜條件式邏輯的查詢中使用函數。

```
SELECT SQRT(commission)
FROM sales WHERE salesid < 10 ORDER BY salesid;
```

```
+-----+
|      sqrt      |
+-----+
| 10.449880382090505 |
| 3.3763886032268267 |
| 7.245688373094719 |
| 5.123475382979799 |
| 4.806245936279167 |
| 7.687652437513028 |
| 10.871982339941507 |
| 5.4359911699707535 |
| 9.41541289588513 |
+-----+
```

若要傳回同一組COMMISSION值的四捨五入平方根，請使用下列範例。

```
SELECT ROUND(SQRT(commission))
FROM sales WHERE salesid < 10 ORDER BY salesid;
```

```
+-----+
| round |
+-----+
|    10 |
|     3 |
|     7 |
|     5 |
|     5 |
|     8 |
|    11 |
|     5 |
|     9 |
+-----+
```

TAN 函數

TAN 是三角形函數，會傳回數字的切線。輸入引數是一個數字 (以弧度表示)。

語法

```
TAN(number)
```

引數

number

DOUBLE PRECISION 數字。

傳回類型

DOUBLE PRECISION

範例

若要傳回 0 的正切，請使用下列範例。

```
SELECT TAN(0);
```

```
+-----+
| tan |
+-----+
|  0 |
+-----+
```

TRUNC 函數

TRUNC 函數會將數字截斷為上一個整數或小數。

TRUNC 函數可以選擇性地包含第二個引數作為 INTEGER，以指出四捨五入的小數位數。當您未提供第二個引數時，函數會四捨五入為最接近的整數。指定第二個引數 integer 時，函數會捨入為具有 integer 小數位數的最接近的數字。

這個函數也可以截斷 TIMESTAMP 並傳回 DATE。如需詳細資訊，請參閱[TRUNC 函數](#)。

語法

```
TRUNC(number [ , integer ])
```

引數

number

數字或評估為數字的運算式。它可以是 DECIMAL、FLOAT8 或 SUPER 類型。Amazon Redshift 可以根據隱含轉換規則轉換其他資料類型。

integer

(選用) INTEGER，表示精確度的小數位數 (任一方向)。如果未提供整數，數字會截斷為整數；如果指定整數，數字會截斷至指定的小數位數。不支援 SUPER 資料類型。

傳回類型

TRUNC 會傳回與輸入號碼 相同的資料類型。

當輸入為 SUPER 類型時，輸出會保留與輸入相同的動態類型，而靜態類型仍然是 SUPER 類型。當 SUPER 的動態類型不是數字時，Amazon Redshift 會傳回 NULL。

範例

下列一些範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

若要截斷給定銷售交易的已付佣金，請使用下列範例。

```
SELECT commission, TRUNC(commission)
FROM sales WHERE salesid=784;
```

```
+-----+-----+
| commission | trunc |
+-----+-----+
|      111.15 |    111 |
+-----+-----+
```

若要將同一個佣金值截斷至第一位小數，請使用下列範例。

```
SELECT commission, TRUNC(commission,1)
FROM sales WHERE salesid=784;
```

```
+-----+-----+
| commission | trunc |
+-----+-----+
|      111.15 |  111.1 |
+-----+-----+
```

若要使用第二個引數的負值截斷佣金，請使用下列範例。請注意 111.15 向下捨入為 110。

```
SELECT commission, TRUNC(commission,-1)
FROM sales WHERE salesid=784;
```

```
+-----+-----+
| commission | trunc |
+-----+-----+
|      111.15 |    110 |
+-----+-----+
```

物件函數

以下是 Amazon Redshift 支援用來建立 SUPER 類型 SQL 物件的物件函數：

主題

- [LOWER_ATTRIBUTE_NAMES 函數](#)
- [OBJECT 函數](#)
- [OBJECT_TRANSFORM 函數](#)

- [UPPER_ATTRIBUTE_NAMES 函數](#)

LOWER_ATTRIBUTE_NAMES 函數

使用與 [UPPER_ATTRIBUTE_NAMES](#) 相同的案例轉換常式，將 SUPER 值中所有適用的屬性名稱轉換為小寫 [LOWER 函數](#)。LOWER_ATTRIBUTE_NAMES 支援 UTF-8 個多位元組字元，每個字元最多四個位元組。

若要將 SUPER 屬性名稱轉換為大寫，請使用 [UPPER_ATTRIBUTE_NAMES 函數](#)。

語法

```
LOWER_ATTRIBUTE_NAMES(super_expression)
```

引數

super_expression

SUPER 表達式。

傳回類型

SUPER

使用須知

在 Amazon Redshift 中，資料欄識別碼傳統上不區分大小寫，並轉換為小寫。如果您從區分大小寫的資料格式擷取資料，例如 JSON，則資料可能包含混合大小寫屬性名稱。

請考量下列範例。

```
CREATE TABLE t1 (s) AS SELECT JSON_PARSE({'AttributeName': 'Value'});

SELECT s.AttributeName FROM t1;

attributename
-----
NULL

SELECT s."AttributeName" FROM t1;
```

```

attributename
-----
NULL

```

Amazon Redshift NULL 會傳回這兩個查詢。若要查詢 AttributeName，請使用 LOWER_ATTRIBUTE_NAMES 將資料屬性名稱轉換為小寫。請考量下列範例。

```
CREATE TABLE t2 (s) AS SELECT LOWER_ATTRIBUTE_NAMES(s) FROM t1;
```

```
SELECT s.attributename FROM t2;
```

```

attributename
-----
"Value"

```

```
SELECT s.AttributeName FROM t2;
```

```

attributename
-----
"Value"

```

```
SELECT s."attributename" FROM t2;
```

```

attributename
-----
"Value"

```

```
SELECT s."AttributeName" FROM t2;
```

```

attributename
-----
"Value"

```

使用混合大小寫物件屬性名稱的相關選項是 `enable_case_sensitive_super_attribute` 組態選項，可讓 Amazon Redshift 以 SUPER 屬性名稱識別大小寫。這可能是使用 LOWER_ATTRIBUTE_ 的替代解決方案 NAMES。如需的詳細資訊 `enable_case_sensitive_super_attribute`，請前往 [enable_case_sensitive_super_attribute](#)。

範例

將SUPER屬性名稱轉換為小寫

下列範例使用 LOWER_ATTRIBUTE_NAMES 來轉換資料表中所有SUPER值的屬性名稱。

```
-- Create a table and insert several SUPER values.
CREATE TABLE t (i INT, s SUPER);

INSERT INTO t VALUES
  (1, NULL),
  (2, 'A'::SUPER),
  (3, JSON_PARSE('{"AttributeName": "B"}')),
  (4, JSON_PARSE(
    '[{"Subobject": {"C": "C"},
      "Subarray": [{"D": "D"}, "E"]}'));

-- Convert all attribute names to lowercase.
UPDATE t SET s = LOWER_ATTRIBUTE_NAMES(s);

SELECT i, s FROM t ORDER BY i;
```

i	s
1	NULL
2	"A"
3	{"attributename": "B"}
4	[{"subobject": {"c": "C"}, "subarray": [{"d": "D"}, "E"]}]

觀察 LOWER_ATTRIBUTE_NAMES 如何運作。

- NULL 值和純量SUPER值，例如 "A" 保持不變。
- 在 SUPER 物件中，所有屬性名稱都會變更為小寫，但屬性值，例如 "B" 保持不變。
- LOWER_ATTRIBUTE_NAMES 會遞迴套用至巢狀於SUPER陣列內部或另一個SUPER物件內部的任何物件。

在具有重複屬性名稱的SUPER物件上使用 LOWER_ATTRIBUTE_NAMES

如果SUPER物件包含的屬性其名稱僅在其情況下不同，LOWER_ATTRIBUTE_NAMES 將引發錯誤。請考量下列範例。

```
SELECT LOWER_ATTRIBUTE_NAMES(JSON_PARSE('{\"A\": \"A\", \"a\": \"a\"}'));
```

```
error:   Invalid input
```

```
code:    8001
```

```
context: SUPER value has duplicate attributes after case conversion.
```

OBJECT 函數

建立SUPER資料類型的物件。

語法

```
OBJECT ( [ key1, value1 ], [ key2, value2 ... ] )
```

引數

key1, key2

評估以VARCHAR輸入字串的運算式。

value1, value2

任何 Amazon Redshift 資料類型的表達式，日期時間類型除外，因為 Amazon Redshift 不會將日期時間類型轉換為SUPER資料類型。如需日期時間類型的相關資訊，請參閱[日期時間 \(Datetime\) 類型](#)。

物件中的 value 運算式不需要是相同的資料類型。

傳回類型

SUPER

範例

```
-- Creates an empty object.
```

```
select object();
```

```
object
```

```
-----
```

```
{}
```

```
(1 row)
```

```

-- Creates objects with different keys and values.
select object('a', 1, 'b', true, 'c', 3.14);

object
-----
{"a":1,"b":true,"c":3.14}
(1 row)

select object('a', object('aa', 1), 'b', array(2,3), 'c', json_parse('{}'));

object
-----
{"a":{"aa":1},"b":[2,3],"c":{}}
(1 row)

-- Creates objects using columns from a table.
create table bar (k varchar, v super);
insert into bar values ('k1', json_parse('[1]')), ('k2', json_parse('{}'));
select object(k, v) from bar;

object
-----
{"k1":[1]}
{"k2":{}}
(2 rows)

-- Errors out because DATE type values can't be converted to SUPER type.
select object('k', '2008-12-31'::date);

ERROR:  OBJECT could not convert type date to super

```

OBJECT_TRANSFORM 函數

轉換SUPER物件。

語法

```

OBJECT_TRANSFORM(
  input
  [KEEP path1, ...]
  [SET
    path1, value1,

```

```
    ..., ...  
  ]  
)
```

引數

input

解析為SUPER類型物件的表達式。

KEEP

在這個子句中指定的所有路徑值都會保留，並轉移到輸出物件。

此子句是選用的。

path1、path2...

常數字串常值，採用以句點分隔的雙引號路徑元件的格式。例如，`'"a"."b"."c"'` 是有效的路徑值。這適用於 KEEP 和 SET 子句中的路徑參數。

SET

path 和 value 配對可修改現有路徑或新增路徑，並在輸出物件中設定該路徑的值。

此子句是選用的。

value1、value2...

解析為SUPER輸入值的表達式。請注意，數值、文字和布林類型可解析為 SUPER。

傳回類型

SUPER

使用須知

OBJECT_TRANSFORM 傳回SUPER類型物件，其中包含在 中指定的路徑值，KEEP 以及在 中指定的路徑和值對SET。

如果 KEEP 和 SET 都是空的，OBJECT_TRANSFORM 會傳回輸入。

如果輸入不是SUPER類型物件，OBJECT_TRANSFORM 會傳回輸入，而不論任何 KEEP 或 SET 值。

範例

下列範例會將SUPER物件轉換為另一個SUPER物件。

```
CREATE TABLE employees (  
    col_person SUPER  
);  
  
INSERT INTO employees  
VALUES  
    (  
        json_parse('  
            {  
                "name": {  
                    "first": "John",  
                    "last": "Doe"  
                },  
                "age": 25,  
                "ssn": "111-22-3333",  
                "company": "Company Inc.",  
                "country": "U.S."  
            }  
        ')  
    ),  
    (  
        json_parse('  
            {  
                "name": {  
                    "first": "Jane",  
                    "last": "Appleseed"  
                },  
                "age": 34,  
                "ssn": "444-55-7777",  
                "company": "Organization Org.",  
                "country": "Ukraine"  
            }  
        ')  
    )  
;  
  
SELECT  
    OBJECT_TRANSFORM(  
        col_person  
        KEEP
```



```
        "name"."first",
        "age",
        "company",
        "country"
    SET
        "name"."first", UPPER(col_person.name.first::TEXT),
        "age", col_person.age + 5,
        "company", 'Amazon'
    ) AS col_person_transformed
FROM employees;
```

--This result is formatted for ease of reading.

```
        col_person_transformed
-----
{
  "name": {
    "first": "JOHN"
  },
  "age": 30,
  "company": "Amazon",
  "country": "U.S."
}
{
  "name": {
    "first": "JANE"
  },
  "age": 39,
  "company": "Amazon",
  "country": "Ukraine"
}
```

UPPER_ATTRIBUTE_NAMES 函數

使用與 [相同的案例轉換常式](#)，將SUPER值中所有適用的屬性名稱轉換為大寫[UPPER 函數](#)。UPPER_ATTRIBUTE_NAMES 支援 UTF-8 個多位元組字元，每個字元最多四個位元組。

若要將SUPER屬性名稱轉換為小寫，請使用 [LOWER_ATTRIBUTE_NAMES 函數](#)。

語法

```
UPPER_ATTRIBUTE_NAMES(super_expression)
```

引數

`super_expression`

SUPER 表達式。

傳回類型

SUPER

範例

將SUPER屬性名稱轉換為大寫

下列範例使用 `UPPER_ATTRIBUTE_NAMES` 來轉換資料表中所有SUPER值的屬性名稱。

```
-- Create a table and insert several SUPER values.
CREATE TABLE t (i INT, s SUPER);

INSERT INTO t VALUES
  (1, NULL),
  (2, 'a'::SUPER),
  (3, JSON_PARSE('{"AttributeName": "b"}')),
  (4, JSON_PARSE(
    '[{"Subobject": {"c": "c"},
      "Subarray": [{"d": "d"}, "e"]}'));

-- Convert all attribute names to uppercase.
UPDATE t SET s = UPPER_ATTRIBUTE_NAMES(s);

SELECT i, s FROM t ORDER BY i;
```

i	s
1	NULL
2	"a"
3	{"ATTRIBUTENAME": "B"}
4	[{"SUBOBJECT": {"C": "c"}, "SUBARRAY": [{"D": "d"}, "e"]}]

觀察 `UPPER_ATTRIBUTE_NAMES` 的運作方式。

- NULL 值和純量SUPER值，例如 "a" 保持不變。

- 在 SUPER 物件中，所有屬性名稱都會變更為大寫，但屬性值，例如 "b" 保持不變。
- UPPER_ATTRIBUTE_NAMES 會遞迴套用到巢狀在 SUPER 陣列內部或另一個 SUPER 物件內部的任何物件。

在具有重複屬性名稱的 SUPER 物件上使用 UPPER_ATTRIBUTE_NAMES

如果 SUPER 物件包含的屬性名稱僅在其情況下不同，UPPER_ATTRIBUTE_NAMES 將引發錯誤。請考量下列範例。

```
SELECT UPPER_ATTRIBUTE_NAMES(JSON_PARSE('{ "A": "A", "a": "a" }'));
```

```
error:   Invalid input
```

```
code:    8001
```

```
context: SUPER value has duplicate attributes after case conversion.
```

空間函數

幾何物件之間的關係是以維度延伸九交模型 (DE-9IM) 為基礎。這個模型定義了如相等、包含和涵蓋等述詞。如需空間關係定義的相關資訊，請參閱 Wikipedia 中的 [DE-9IM](#)。

如需如何將空間資料與 Amazon Redshift 搭配使用的相關資訊，請參閱 [在 Amazon Redshift 中查詢空間資料](#)。

Amazon Redshift 提供了可與 GEOMETRY 和 GEOGRAPHY 資料類型一起使用的空間函數。以下列出支援 GEOGRAPHY 資料類型的函數：

- [ST_Area](#)
- [ST_AsEWKT](#)
- [JSON AsGeo](#)
- [ST_ EWKB AsHex](#)
- [ST_ 白板 AsHex](#)
- [ST_ AsText](#)
- [ST_Distance](#)
- [ST_GeogFromText](#)
- [ST_ 白板 GeogFrom](#)
- [ST_Length](#)

- [ST_NPoints](#)
- [ST_Perimeter](#)

以下列出 Amazon Redshift 支援的完整空間函數集。

主題

- [AddBBox](#)
- [DropBBox](#)
- [GeometryType](#)
- [H3_FromLongLat](#)
- [H3_FromPoint](#)
- [H3_Polyfill](#)
- [ST_AddPoint](#)
- [ST_Angle](#)
- [ST_Area](#)
- [ST_AsBinary](#)
- [ST_AsEWKB](#)
- [ST_AsEWKT](#)
- [JSON AsGeo](#)
- [ST_ 白板 AsHex](#)
- [ST_ EWKB AsHex](#)
- [ST_AsText](#)
- [ST_Azimuth](#)
- [ST_Boundary](#)
- [ST_Buffer](#)
- [ST_Centroid](#)
- [ST_Collect](#)
- [ST_Contains](#)
- [ST_ContainsProperly](#)
- [ST_ConvexHull](#)

- [ST_CoveredBy](#)
- [ST_Covers](#)
- [ST_Crosses](#)
- [ST_Dimension](#)
- [ST_Disjoint](#)
- [ST_Distance](#)
- [ST_DistanceSphere](#)
- [ST_DWithin](#)
- [ST_EndPoint](#)
- [ST_Envelope](#)
- [ST_Equals](#)
- [ST_ExteriorRing](#)
- [ST_Force2D](#)
- [ST_Force3D](#)
- [ST_Force3DM](#)
- [ST_Force3DZ](#)
- [ST_Force4D](#)
- [ST_GeoHash](#)
- [ST_GeogFromText](#)
- [ST_白板 GeogFrom](#)
- [ST_GeometryN](#)
- [ST_GeometryType](#)
- [ST_EWKB GeomFrom](#)
- [ST-尤克特 GeomFrom](#)
- [ST_GeomFromGeoHash](#)
- [JSON GeomFromGeo](#)
- [ST_GeomFromGeoSquare](#)
- [ST_GeomFromText](#)
- [ST_白板 GeomFrom](#)

- [ST_GeoSquare](#)
- [ST_N InteriorRing](#)
- [ST_Intersects](#)
- [ST_Intersection](#)
- [ST_逆時鐘 IsPolygon](#)
- [ST_密西西比 IsPolygon](#)
- [ST_IsClosed](#)
- [ST_IsCollection](#)
- [ST_IsEmpty](#)
- [ST_IsRing](#)
- [ST_IsSimple](#)
- [ST_IsValid](#)
- [ST_Length](#)
- [ST_LengthSphere](#)
- [ST_Length2D](#)
- [ST_LineFromMultiPoint](#)
- [ST_LineInterpolatePoint](#)
- [ST_M](#)
- [ST_MakeEnvelope](#)
- [ST_MakeLine](#)
- [ST_MakePoint](#)
- [ST_MakePolygon](#)
- [ST_MemSize](#)
- [ST_MMax](#)
- [ST_MMin](#)
- [ST_Multi](#)
- [ST_NDims](#)
- [ST_NPoints](#)
- [ST_NRings](#)

- [ST_NumGeometries](#)
- [ST_NumInteriorRings](#)
- [ST_NumPoints](#)
- [ST_Perimeter](#)
- [ST_Perimeter2D](#)
- [ST_Point](#)
- [ST_PointN](#)
- [ST_Points](#)
- [ST_Polygon](#)
- [ST_RemovePoint](#)
- [ST_Reverse](#)
- [ST_SetPoint](#)
- [ST_SetSRID](#)
- [ST_Simplify](#)
- [ST_SRID](#)
- [ST_StartPoint](#)
- [ST_Touches](#)
- [ST_Transform](#)
- [ST_Union](#)
- [ST_Within](#)
- [ST_X](#)
- [ST_XMax](#)
- [ST_XMin](#)
- [ST_Y](#)
- [ST_YMax](#)
- [ST_YMin](#)
- [ST_Z](#)
- [ST_ZMax](#)
- [ST_ZMin](#)

- [SupportsBBox](#)

AddBBox

AddBBox 會傳回支援使用預先計算的週框方塊編碼輸入幾何的副本。如需週框方塊支援的相關資訊，請參閱[邊界框](#)。

語法

```
AddBBox(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會傳回支援使用週框方塊編碼的輸入多邊形幾何的副本。

```
SELECT ST_AsText(AddBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
st_astext
-----
POLYGON((0 0,1 0,0 1,0 0))
```

DropBBox

DropbBox 會傳回不支援使用預先計算的邊界方框編碼的輸入幾何副本。如需週框方塊支援的相關資訊，請參閱[邊界框](#)。

語法

```
DropBBox(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回不支援使用週框方塊編碼之輸入多邊形幾何的副本。

```
SELECT ST_AsText(DropBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0)'))));
```

```
st_astext
-----
POLYGON((0 0,1 0,0 1,0 0))
```

GeometryType

GeometryType 以字串形式傳回輸入幾何的子類型。

語法

```
GeometryType(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

VARCHAR 表示 geom 的子類型。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回 2D、3DZ、4D 幾何的字串值	傳回 3DM 幾何的字串值	幾何子類型
POINT	POINTM	在 geom 是 POINT 子類型時傳回
LINestring	LINestringM	在 geom 是 LINestring 子類型時傳回
POLYGON	POLYGONM	在 geom 是 POLYGON 子類型時傳回
MULTIPOINT	MULTIPOINTM	在 geom 是 MULTIPOINT 子類型時傳回
MULTILINestring	MULTILINestringM	在 geom 是 MULTILINE STRING 子類型時傳回
MULTIPOLYGON	MULTIPOLYGONM	在 geom 是 MULTIPOLYGON 子類型時傳回
GEOMETRYCOLLECTION	GEOMETRYCOLLECTIONM	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會轉換多邊形的已知文字 (WKT) 表示法，並將 GEOMETRY 子類型做為字串傳回。

```
SELECT GeometryType(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))');
```

```
geometrytype
```

```
-----
```

```
POLYGON
```

H3_FromLongLat

H3_ 從輸入經度、緯度和解析度FromLongLat 傳回對應的 H3 儲存格 ID。如需 H3 索引的詳細資訊，請參閱 [H3](#)。

語法

```
H3_FromLongLat(longitude, lattitude, resolution)
```

引數

經度

DOUBLE PRECISION 資料類型的值，或是評估為 DOUBLE PRECISION 類型的表達式。

緯度

DOUBLE PRECISION 資料類型的值，或是評估為 DOUBLE PRECISION 類型的表達式。

解析度

INTEGER 資料類型的值，或是評估為 INTEGER 類型的表達式。該值表示 H3 網格系統的解析度。該值必須是介於 0 和 15 之間的整數。0 是最普遍的，15 是最好。

傳回類型

BIGINT— 代表 H3 儲存格 ID。

如果解析度超出範圍，則會傳回錯誤。

範例

下列 SQL 會從經度 0、緯度 0 和解析度 10 傳回 H3 儲存格 ID。

```
SELECT H3_FromLongLat(0, 0, 10);
```

```
h3_fromlonglat
```

```
-----  
623560421467684863
```

H3_FromPoint

H3_ 從輸入幾何點和解析度FromPoint 傳回對應的 H3 零件 ID。如需 H3 索引的詳細資訊，請參閱 [H3](#)。

語法

```
H3_FromPoint(geom, resolution)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。geom 必須是 POINT。

解析度

INTEGER 資料類型的值，或是評估為 INTEGER 類型的表達式。該值表示 H3 網格系統的解析度。該值必須是介於 0 和 15 之間的整數。0 是最普遍的，15 是最好。

傳回類型

BIGINT— 代表 H3 儲存格 ID。

如果 geom 不是 POINT，則會傳回錯誤。

如果解析度超出範圍，則會傳回錯誤。

如果 geom 為空白，則會傳回 NULL。

範例

下方的 SQL 從點 0,0 和解析度 10 傳回 H3 儲存格 ID。

```
SELECT H3_FromPoint(ST_GeomFromText('POINT(0 0)'), 10);
```

```
h3_frompoint
```

```
-----  
623560421467684863
```

H3_Polyfill

H3_Polyfill 會傳回對應於指定解析度之輸入多邊形中包含的六邊形和五邊形的對應 H3 儲存格 ID。如需 H3 索引的詳細資訊，請參閱 [H3](#)。

語法

```
H3_Polyfill(geom, resolution)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。geom 必須是 POLYGON。

解析度

INTEGER 資料類型的值，或是評估為 INTEGER 類型的表達式。該值表示 H3 網格系統的解析度。該值必須是介於 0 和 15 之間的整數。0 是最普遍的，15 是最好。

傳回類型

SUPER— 代表 H3 儲存格 ID 的清單。

如果 geom 不是 POLYGON，則會傳回錯誤。

如果解析度超出範圍，則會傳回錯誤。

如果 geom 為空白，則會傳回 NULL。

範例

下方的 SQL 會從多邊形和解析度 4 傳回 H3 儲存格 ID 的 SUPER 資料類型陣列。

```
SELECT H3_Polyfill(ST_GeomFromText('POLYGON((0 0, 0 1, 1 1, 1 0, 0 0))'), 4);
```

```
h3_polyfill  
-----
```

```
[596538848238895103, 596538805289222143, 596538856828829695, 596538813879156735, 59653792052595916
```

ST_AddPoint

ST_會AddPoint 傳回與加入點的輸入幾何圖形相同的線串幾何圖形。如果提供索引，則點會新增在索引位置。如果索引為 -1 或未提供，則點附加至 linestring。

索引是以零開始。結果的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

傳回幾何的維度與 geom1 值的維度相同。如果 geom1 和 geom2 具有不同的維度，則將 geom2 投影到 geom1 的維度。

語法

```
ST_AddPoint(geom1, geom2)
```

```
ST_AddPoint(geom1, geom2, index)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。該點可以是空點。

索引

資料類型 INTEGER 的值，代表以零開始的索引位置。

傳回類型

GEOMETRY

如果 geom1、geom2 或 index 為 Null，則會傳回 Null。

如果 geom2 是空點，則傳回 geom1 的副本。

如果 geom1 不是 LINESTRING，則會傳回錯誤。

如果 geom2 不是 POINT，則會傳回錯誤。

如果 index 超出範圍，則會傳回錯誤。索引位置的有效值是 -1 或是介於 0 與 ST_NumPoints(geom1) 之間的值。

範例

下列 SQL 會將一點新增到 linestring，使其成為一個封閉的 linestring。

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_AddPoint(g, ST_StartPoint(g))) FROM tmp;
```

```
st_asewkt
-----
SRID=4326;LINESTRING(0 0,10 0,10 10,5 5,0 5,0 0)
```

下列 SQL 會將一點新增到 linestring 中的特定位置。

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_AddPoint(g, ST_SetSRID(ST_Point(5, 10), 4326), 3)) FROM tmp;
```

```
st_asewkt
-----
SRID=4326;LINESTRING(0 0,10 0,10 10,5 10,5 5,0 5)
```

ST_Angle

ST_Angle 傳回以弧度為單位順時針測量的點之間的角度，如下所示：

- 如果輸入三個點，則將測量傳回的角度 P1-P2-P3，就像繞 P2 順時針從 P1 旋轉到 P3 獲得的角度一樣。
- 如果輸入四個點，則傳回由有向線 P1-P2 和 P3-P4 形成的順時針角度。如果輸入是退化情況 (也就是說，P1 等於 P2，或 P3 等於 P4)，則傳回 null。

傳回值以弧度為單位，範圍為 $[0, 2\pi)$ 。

ST_Angle 對輸入幾何的 2D 投影進行操作。

語法

```
ST_Angle(geom1, geom2, geom3)
```

```
ST_Angle(geom1, geom2, geom3, geom4)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

geom3

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

geom4

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

傳回類型

DOUBLE PRECISION.

如果 geom1 等於 geom2，或 geom2 等於 geom3，則傳回 null。

如果 geom1、geom2、geom3 或 geom4 為 null，則傳回 null。

如果任何 geom1、geom2、geom3 或 geom4 是空點，則傳回錯誤。

如果 geom1、geom2、geom3 和 geom4 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

範例

下列 SQL 會傳回三個輸入點轉換為度數的角度。

```
SELECT ST_Angle(ST_Point(1,1), ST_Point(0,0), ST_Point(1,0)) / Pi() * 180.0 AS angle;
```



```
angle
```

```
-----  
45
```

下列 SQL 會傳回四個輸入點轉換為度數的角度。

```
SELECT ST_Angle(ST_Point(1,1), ST_Point(0,0), ST_Point(1,0), ST_Point(2,0)) / Pi() *  
180.0 AS angle;
```

```
angle
```

```
-----  
225
```

ST_Area

對於輸入幾何，ST_Area 傳回 2D 投影的笛卡爾面積。面積單位與表示輸入幾何座標的單位相同。對於點、linestring、multipoint 和 multilinestring，函數會傳回 0。對於幾何集合，它會傳回集合中幾何面積的總和。

對於輸入地理，ST_Area 會傳回在球體 (由 SRID 決定) 上所計算輸入面積地理之 2D 投影的測地線面積。長度單位為平方公尺。對於點、multipoint 和線形地理，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中面積地理的面積總和。

語法

```
ST_Area(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

DOUBLE PRECISION

如果 *geo* 為 null，則傳回 null。

範例

下列 SQL 會傳回 multipolygon 的笛卡爾面積。

```
SELECT ST_Area(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20 10,10 0)))'));
```

```
st_area
-----
      100
```

以下 SQL 傳回地理中多邊形的面積。

```
SELECT ST_Area(ST_GeogFromText('polygon((34 35, 28 30, 25 34, 34 35))'));
```

```
st_area
-----
201824655743.383
```

下列 SQL 針對線性地理傳回零。

```
SELECT ST_Area(ST_GeogFromText('multipoint(0 0, 1 1, -21.32 121.2)'));
```

```
st_area
-----
      0
```

ST_AsBinary

ST_ 會AsBinary 傳回輸入幾何的十六進位已知二進位 (WKB) 表示法。對於 3DZ、3DM 和 4D 幾何圖形，ST_ AsBinary 使用開放地理空間聯盟 (OGC) 標準值作為幾何類型。

語法

```
ST_AsBinary(geom)
```


如果 precision 為 Null，則會傳回 Null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回 linestring 的 EWKT 表示法。

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(3.141592653589793
-6.283185307179586,2.718281828459045 -1.414213562373095)', 4326));
```

```
st_asewkt
-----
SRID=4326;LINESTRING(3.14159265358979 -6.28318530717959,2.71828182845905
-1.41421356237309)
```

以下 SQL 會傳回 linestring 的 EWKT 表示法。幾何的座標會使用六位數的精確度顯示。

```
SELECT ST_AsEWKT(ST_GeomFromText('LINESTRING(3.141592653589793
-6.283185307179586,2.718281828459045 -1.414213562373095)', 4326), 6);
```

```
st_asewkt
-----
SRID=4326;LINESTRING(3.14159 -6.28319,2.71828 -1.41421)
```

以下 SQL 會傳回地理的 EWKT 表示法。

```
SELECT ST_AsEWKT(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
st_asewkt
-----
SRID=4326;LINESTRING(110 40,2 3,-10 80,-7 9)
```

JSON AsGeo

AsGeoJSON 會傳回輸入幾何或地理位置的 GeoJSON 表示法。如需 GeoJSON 的相關資訊，請參閱 Wikipedia 中的 [GeoJSON](#)。

對於 3DZ 和 4D 幾何，輸出幾何是輸入 3DZ 或 4D 幾何的 3DZ 投影。也就是說，x、y 和 z 座標存在於輸出中。對於 3DM 幾何，輸出幾何是輸入 3DM 幾何的 2D 投影。也就是說，只有 x 和 y 座標存在於輸出中。

對於輸入的地理位置，ST_AsGeoJSON 會傳回輸入地理位置的 GeoJSON 表示法。使用指定的精確度顯示地理座標。

語法

```
ST_AsGeoJSON(geo)
```

```
ST_AsGeoJSON(geo, precision)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

precision

INTEGER 資料類型的值。對於幾何，使用指定的精確度 1-20 顯示 geo 座標。如果沒有指定 precision，則預設為 15。對於地理，使用指定的精確度顯示 geo 座標。如果沒有指定 precision，則預設為 15。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果 precision 為 Null，則會傳回 Null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回 linestring 的 GeoJSON 表示法。

```
SELECT ST_AsGeoJSON(ST_GeomFromText('LINESTRING(3.141592653589793  
-6.283185307179586,2.718281828459045 -1.414213562373095)'));
```

```
st_asgeojson
```

```
-----
{"type":"LineString","coordinates":[[[3.14159265358979,-6.28318530717959],
[2.71828182845905,-1.41421356237309]]]}
```

以下 SQL 會傳回 linestring 的 GeoJSON 表示法。幾何的座標會使用六位數的精確度顯示。

```
SELECT ST_AsGeoJSON(ST_GeomFromText('LINESTRING(3.141592653589793
-6.283185307179586,2.718281828459045 -1.414213562373095)'), 6);
```

```
st_asgeojson
```

```
-----
{"type":"LineString","coordinates":[[[3.14159,-6.28319],[2.71828,-1.41421]]]}
```

以下 SQL 會傳回地理的 GeoJSON 表示法。

```
SELECT ST_AsGeoJSON(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
st_asgeojson
```

```
-----
{"type":"LineString","coordinates":[[[110,40],[2,3],[-10,80],[-7,9]]]}
```

ST_ 白板 AsHex

ST_AsHex WKB 會使用 ASCII 十六進位字元 (0—9、A — F) 傳回輸入幾何或地理位置的十六進位已知二進位 (WKB) 表示。對於 3DZ、3DM 和 4D 幾何圖形或地理位置，ST_AsHex WKB 會針對幾何圖形或地理類型使用開放地理空間協會 (OGC) 標準值。

語法

```
ST_AsHexWKB(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

語法

```
ST_AsText(geo)
```

```
ST_AsText(geo, precision)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

precision

INTEGER 資料類型的值。對於幾何，使用指定的精確度 1-20 顯示 geo 座標。如果沒有指定 precision，則預設為 15。對於地理，使用指定的精確度顯示 geo 座標。如果沒有指定 precision，則預設為 15。

傳回類型

VARCHAR

如果 geo 為 null，則傳回 null。

如果 precision 為 Null，則會傳回 Null。

如果結果大於 64-KB VARCHAR，則會傳回錯誤。

範例

以下 SQL 會傳回 linestring 的 WKT 表示法。

```
SELECT ST_AsText(ST_GeomFromText('LINESTRING(3.141592653589793  
-6.283185307179586,2.718281828459045 -1.414213562373095)', 4326));
```

```
st_astext  
-----  
LINESTRING(3.14159265358979 -6.28318530717959,2.71828182845905 -1.41421356237309)
```

以下 SQL 會傳回 linestring 的 WKT 表示法。幾何的座標會使用六位數的精確度顯示。

```
SELECT ST_AsText(ST_GeomFromText('LINESTRING(3.141592653589793
-6.283185307179586,2.718281828459045 -1.414213562373095)', 4326), 6);
```

```
st_astext
-----
LINESTRING(3.14159 -6.28319,2.71828 -1.41421)
```

以下 SQL 會傳回地理的 WKT 表示法。

```
SELECT ST_AsText(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
st_astext
-----
LINESTRING(110 40,2 3,-10 80,-7 9)
```

ST_Azimuth

ST_Azimuth 會使用兩個輸入點的 2D 投影傳回以北為基礎的笛卡爾方位。

語法

```
ST_Azimuth(point1, point2)
```

引數

point1

GEOMETRY 資料類型的 POINT 值。point1 的空間參考系統識別碼 (SRID) 必須與 point2 的 SRID 相符。

point2

GEOMETRY 資料類型的 POINT 值。point2 的 SRID 必須與 point1 的 SRID 相符。

傳回類型

數字，為 DOUBLE PRECISION 資料類型的角度 (弧度)。值的範圍介於 0 (含) 到 2 pi (不含)。

如果 point1 或 point2 是空點，則會傳回錯誤。

如果 point1 或 point2 為 Null，則會傳回 Null。

如果 point1 和 point2 相等，則會傳回 Null。

如果 point1 或 point2 不是點，則會傳回錯誤。

如果 point1 和 point2 沒有空間參考系統識別碼 (SRID) 的值，則會傳回錯誤。

範例

以下 SQL 會傳回輸入點的方位。

```
SELECT ST_Azimuth(ST_Point(1,2), ST_Point(5,6));
```

```
st_azimuth
-----
0.7853981633974483
```

ST_Boundary

ST_Boundary 會傳回輸入幾何的邊界，如下所示：

- 如果輸入幾何是空的 (也就是說，它不包含任何點)，則會按原樣傳回。
- 如果輸入幾何是點或非空的多點，則會傳回空的幾何集合。
- 如果輸入為 linestring 或 multilinestring，則會傳回包含邊界上所有點的多點。多點可能是空的)。
- 如果輸入是沒有任何內環的多邊形，則會傳回表示其邊界的封閉 linestring。
- 如果輸入是具有內環的多邊形，或者是多重多邊形，則會傳回 multilinestring。Multilinestring 包含面積幾何中所有環的所有邊界，做為封閉 linestring。

為了確定點相等，ST_Boundary 對輸入幾何的 2D 投影進行操作。如果輸入幾何為空，則會以與輸入相同的維度傳回其副本。對於非空的 3DM 和 4D 幾何，它們的 m 座標會被捨棄。在 3DZ 和 4D multilinestring 的特殊情況下，multilinestring 邊界點的 z 座標會計算為具有相同 2D 投影之 linestring 邊界點之相異 z 值的平均值。

語法

```
ST_Boundary(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 GEOMETRYCOLLECTION，則會傳回錯誤。

範例

下列 SQL 傳回輸入多邊形的邊界作為 multilinestring。

```
SELECT ST_AsEWKT(ST_Boundary(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1)'))));
```

```
st_asewkt
```

```
-----  
MULTILINESTRING((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1))
```

ST_Buffer

ST_Buffer 傳回 2D 幾何，該幾何表示與投影在 xy 笛卡爾平面上的輸入幾何的距離小於或等於輸入距離的所有點。

語法

```
ST_Buffer(geom, distance)
```

```
ST_Buffer(geom, distance, number_of_segments_per_quarter_circle)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

distance

DOUBLE PRECISION 資料類型的值，表示緩衝區的距離 (或半徑)。

number_of_segments_per_quarter_circle

INTEGER 資料類型的值。此值決定了圍繞輸入幾何的每個頂點近似四分之一圓的點數。負值預設為零。預設值為 8。

傳回類型

GEOMETRY

ST_Buffer 函數會傳回 xy 笛卡爾平面中的二維 (2D) 幾何。

如果 geom 是 GEOMETRYCOLLECTION，則會傳回錯誤。

範例

以下 SQL 會傳回輸入 linestring 的緩衝區。

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('LINESTRING(1 2,5 2,5 8)'), 2));
```

```

      st_asewkt
POLYGON((-1 2,-0.96157056080646 2.39018064403226,-0.847759065022573
 2.76536686473018,-0.662939224605089 3.11114046603921,-0.414213562373093
 3.4142135623731,-0.111140466039201 3.66293922460509,0.234633135269824
 3.84775906502257,0.609819355967748 3.96157056080646,1 4,3 4,3 8,3.03842943919354
 8.39018064403226,3.15224093497743 8.76536686473018,3.33706077539491
 9.11114046603921,3.58578643762691 9.4142135623731,3.8888595339608
 9.66293922460509,4.23463313526982 9.84775906502257,4.60981935596775
 9.96157056080646,5 10,5.39018064403226 9.96157056080646,5.76536686473018
 9.84775906502257,6.11114046603921 9.66293922460509,6.4142135623731
 9.41421356237309,6.66293922460509 9.1111404660392,6.84775906502258
 8.76536686473017,6.96157056080646 8.39018064403225,7 8,7 2,6.96157056080646
 1.60981935596774,6.84775906502257 1.23463313526982,6.66293922460509
 0.888859533960796,6.41421356237309 0.585786437626905,6.1111404660392
 0.33706077539491,5.76536686473018 0.152240934977427,5.39018064403226
 0.0384294391935391,5 0,1 0,0.609819355967744 0.0384294391935391,0.234633135269821
 0.152240934977427,-0.111140466039204 0.337060775394909,-0.414213562373095
 0.585786437626905,-0.662939224605091 0.888859533960796,-0.847759065022574
 1.23463313526982,-0.961570560806461 1.60981935596774,-1 2))

```

以下 SQL 會傳回近似於圓的輸入點幾何圖形的緩衝區。由於此命令未指定每四分之一圓的線段數，因此函數使用八個線段的預設值來近似四分之一圓。

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('POINT(3 4)'), 2));
```

```

      st_asewkt
POLYGON((1 4,1.03842943919354 4.39018064403226,1.15224093497743
4.76536686473018,1.33706077539491 5.11114046603921,1.58578643762691
5.4142135623731,1.8888595339608 5.66293922460509,2.23463313526982
5.84775906502257,2.60981935596775 5.96157056080646,3 6,3.39018064403226
5.96157056080646,3.76536686473019 5.84775906502257,4.11114046603921
5.66293922460509,4.4142135623731 5.41421356237309,4.66293922460509
5.1111404660392,4.84775906502258 4.76536686473017,4.96157056080646 4.39018064403225,5
4,4.96157056080646 3.60981935596774,4.84775906502257 3.23463313526982,4.66293922460509
2.8888595339608,4.41421356237309 2.58578643762691,4.1111404660392
2.33706077539491,3.76536686473018 2.15224093497743,3.39018064403226 2.03842943919354,3
2,2.60981935596774 2.03842943919354,2.23463313526982 2.15224093497743,1.8888595339608
2.33706077539491,1.58578643762691 2.58578643762691,1.33706077539491
2.8888595339608,1.15224093497743 3.23463313526982,1.03842943919354 3.60981935596774,1
4))

```

以下 SQL 會傳回近似於圓的輸入點幾何圖形的緩衝區。由於此命令指定 3 作為每四分之一圓的線段數，因此函數使用三個線段來近似四分之一圓。

```
SELECT ST_AsEwkt(ST_Buffer(ST_GeomFromText('POINT(3 4)'), 2, 3));
```

```

      st_asewkt
POLYGON((1 4,1.26794919243112 5,2 5.73205080756888,3 6,4
5.73205080756888,4.73205080756888 5,5 4,4.73205080756888 3,4 2.26794919243112,3 2,2
2.26794919243112,1.26794919243112 3,1 4))

```

ST_Centroid

st_Centroid 會傳回表示幾何質心的點，如下所示：

- 對於 POINT 幾何，它會傳回座標為幾何中點座標平均值的點。
- 對於 LINESTRING 幾何，它會傳回一個點，其座標是幾何體各段中點的加權平均值，其中權重是幾何體各段的長度。
- 對於 POLYGON 幾何，它會傳回一個點，其座標是面積幾何三角剖分質心的加權平均值，其中權重是三角剖分中三角形的面積。

- 對於幾何集合，它會傳回幾何集合中最大拓撲維度的幾何質心的加權平均值。

語法

```
ST_Centroid(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 *geom* 為 Null，則會傳回 Null。

如果 *geom* 為空白，則會傳回 Null。

範例

以下 SQL 會傳回輸入 *linestring* 的中心點。

```
SELECT ST_AsEWKT(ST_Centroid(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9, -22 -33)', 4326)))
```

```
st_asewkt
```

```
-----  
SRID=4326;POINT(15.6965103455214 27.0206782881905)
```

ST_Collect

ST_Collect 有兩個變體。一個接受兩個幾何，一個接受彙總運算式。

ST_Collect 的第一個變體會從輸入幾何建立幾何。會保留輸入幾何的順序。此變體的運作方式如下：

- 如果兩個輸入幾何都是點，則傳回具有兩個點的 MULTIPOINT。
- 如果兩個輸入幾何都是 *linestring*，則傳回具有兩個 *linestring* 的 MULTILINESTRING。
- 如果兩個輸入幾何都是多邊形，則傳回具有兩個多邊形的 MULTIPOLYGON。

- 否則，會傳回具有兩個輸入幾何的 GEOMETRYCOLLECTION。

ST_Collect 的第二個變體會根據幾何欄中的幾何建立幾何。幾何形狀沒有確定的傳回順序。指定 WITHIN GROUP (ORDER BY ...) 子句以指定傳回幾何的順序。此變體的運作方式如下：

- 如果輸入彙總運算式中的所有非 NULL 列都是點，則傳回包含彙總運算式中所有點的多點。
- 如果彙總運算式中的所有非 NULL 列都是 linestring，則會傳回包含彙總運算式中所有 linestring 的 multilinestring。
- 如果彙總運算式中的所有非 NULL 列都是多邊形，則結果是傳回一個包含彙總運算式中所有多邊形的多重多邊形。
- 否則，會傳回包含彙總運算式中所有幾何的 GEOMETRYCOLLECTION。

ST_Collect 會傳回與輸入幾何相同維度的幾何。所有輸入幾何必須具有相同的維度。

語法

```
ST_Collect(geom1, geom2)
```

```
ST_Collect(aggregate_expression) [WITHIN GROUP (ORDER BY sort_expression1 [ASC | DESC]  
[, sort_expression2 [ASC | DESC] ...])]
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

aggregate_expression

GEOMETRY 資料類型的欄，或是評估為 GEOMETRY 類型的運算式。

[WITHIN GROUP (ORDER BY *sort_expression1* [ASC | DESC] [, *sort_expression2* [ASC | DESC] ...])]

選用子句，指定彙總值的排序順序。ORDER BY 子句包含排序運算式的清單。排序運算式是類似於查詢選取清單中的有效排序運算式 (例如欄名稱) 的運算式。您可以指定遞增 (ASC) 或遞減 (DESC) 順序。預設值為 ASC。

傳回類型

MULTIPOINT、MULTILINESTRING、MULTIPOLYGON 或 GEOMETRYCOLLECTION 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom1 或 geom2 為 null，則傳回 null。

如果 aggregate_expression 的所有列都為 null，則傳回 null。

如果 geom1 為 null，則傳回 geom2 的副本。同樣，如果 geom2 為 null，則傳回 geom1 的副本。

如果 geom1 和 geom2 具有不同的 SRID 值，則會傳回錯誤。

如果 aggregate_expression 中的兩個幾何具有不同的 SRID 值，則會傳回錯誤。

如果傳回的幾何大於 GEOMETRY 的大小上限，則會傳回錯誤。

如果 geom1 和 geom2 的維度不同，則傳回錯誤。

如果 aggregate_expression 中的兩個幾何具有不同的維度，則會傳回錯誤。

範例

下列 SQL 會傳回包含兩個輸入幾何的幾何集合。

```
SELECT ST_AsText(ST_Collect(ST_GeomFromText('LINESTRING(0 0,1 1)'),
  ST_GeomFromText('POLYGON((10 10,20 10,10 20,10 10))')));
```

```
st_astext
-----
GEOMETRYCOLLECTION(LINESTRING(0 0,1 1),POLYGON((10 10,20 10,10 20,10 10)))
```

下列 SQL 會將表格中的所有幾何圖形收集到幾何集合中。

```
WITH tbl(g) AS (SELECT ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT NULL::geometry UNION ALL
SELECT ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326))
```

```
SELECT ST_AsEWKT(ST_Collect(g)) FROM tbl;
```

```
st_astext
```

```
-----
```

```
SRID=4326;GEOMETRYCOLLECTION(POINT(1 2),LINESTRING(0 0,10 0),MULTIPOINT((13 4),(8 5),
(4 4)),POLYGON((0 0,10 0,0 10,0 0)))
```

下列 SQL 會收集資料表中依 id 欄分組並依此 ID 排序的所有幾何。在此範例中，產生的幾何會依 ID 分組，如下所示：

- id 1 – 多點中的點。
- id 2 – multilinestring 中的 linestring。
- id 3 – 幾何集合中的混合子類型。
- id 4 – 多重多邊形中的多邊形。
- id 5 – null，結果為 null。

```
WITH tbl(id, g) AS (SELECT 1, ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT 1, ST_GeomFromText('POINT(4 5)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(10 0,20 -5)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5))', 4326) UNION
ALL
SELECT 4, ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326) UNION ALL
SELECT 4, ST_GeomFromText('POLYGON((20 20,20 30,30 20,20 20))', 4326) UNION ALL
SELECT 1, NULL::geometry UNION ALL SELECT 2, NULL::geometry UNION ALL
SELECT 5, NULL::geometry UNION ALL SELECT 5, NULL::geometry)
SELECT id, ST_AsEWKT(ST_Collect(g)) FROM tbl GROUP BY id ORDER BY id;
```

```
id |
```

```
st_asewkt
```

```
----
```

```
+-----
1 | SRID=4326;MULTIPOINT((1 2),(4 5))
2 | SRID=4326;MULTILINESTRING((0 0,10 0),(10 0,20 -5))
3 | SRID=4326;GEOMETRYCOLLECTION(MULTIPOINT((13 4),(8 5),(4 4)),MULTILINESTRING((-1
-1,-2 -2),(-3 -3,-5 -5)))
```

```
4 | SRID=4326;MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((20 20,20 30,30 20,20 20)))
5 |
```

下列 SQL 會從幾何集中的資料表收集所有幾何。結果會依 id 的遞減順序排序，然後根據它們的最小和最大 x 座標按字典順序排列。

```
WITH tbl(id, g) AS (
SELECT 1, ST_GeomFromText('POINT(4 5)', 4326) UNION ALL
SELECT 1, ST_GeomFromText('POINT(1 2)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(10 0,20 -5)', 4326) UNION ALL
SELECT 2, ST_GeomFromText('LINESTRING(0 0,10 0)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTIPOINT(13 4,8 5,4 4)', 4326) UNION ALL
SELECT 3, ST_GeomFromText('MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5))', 4326) UNION
  ALL
SELECT 4, ST_GeomFromText('POLYGON((20 20,20 30,30 20,20 20))', 4326) UNION ALL
SELECT 4, ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))', 4326) UNION ALL
SELECT 1, NULL::geometry UNION ALL SELECT 2, NULL::geometry UNION ALL
SELECT 5, NULL::geometry UNION ALL SELECT 5, NULL::geometry)
SELECT ST_AsEWKT(ST_Collect(g) WITHIN GROUP (ORDER BY id DESC, ST_XMin(g), ST_XMax(g)))
  FROM tbl;
```

st_asewkt

```
-----
SRID=4326;GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0 0)),POLYGON((20 20,20 30,30
20,20 20)),MULTILINESTRING((-1 -1,-2 -2),(-3 -3,-5 -5)),MULTIPOINT((13 4),(8 5),(4
4)),LINESTRING(0 0,10 0),LINESTRING(10 0,20 -5),POINT(1 2),POINT(4 5))
```

ST_Contains

ST_Contains 會在第一個輸入幾何的 2D 投影包含第二個輸入幾何的 2D 投影時傳回 true。如果 B 中的每個點都是 A 中的點，且其內部包含非空白的交集，則幾何 A 包含幾何 B。

ST_Contains(A, B) 與 ST_Within(B, A) 相等。

語法

```
ST_Contains(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom1 比較，以判斷其是否包含在 geom1 中。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否包含第二個多邊形。

```
SELECT ST_Contains(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
  ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))');
```

```
st_contains
-----
false
```

ST_ContainsProperly

如 ContainsProperly 果兩個輸入幾何圖形都非空白，且第二個幾何圖形的 2D 投影的所有點都是第一個幾何圖形的 2D 投影的內部點，ST_ 會傳回 true。

語法

```
ST_ContainsProperly(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型不能是 GEOMETRYCOLLECTION。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型不能是 GEOMETRYCOLLECTION。此值會與 geom1 進行比較，以判斷其所有點是否都是 geom1 的內點。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 返回 st_Contains 和 ST_ 的值，ContainsProperly 其中輸入線串與輸入多邊形的內部和邊界相交 (但不是其外部)。多邊形包含 linestring，但未正確包含 linestring。

```
WITH tmp(g1, g2)
AS (SELECT ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0))'),
      ST_GeomFromText('LINESTRING(5 5,10 5,10 6,5 5)')) SELECT ST_Contains(g1, g2),
      ST_ContainsProperly(g1, g2)
FROM tmp;
```

```
st_contains | st_containsproperly
-----+-----
t          | f
```

ST_ConvexHull

ST_ConvexHull 傳回幾何，代表輸入幾何中包含的非空點的凸包。

對於空輸入，產生的幾何與輸入幾何相同。對於所有非空輸入，此函數對輸入幾何的 2D 投影進行操作。但是，輸出幾何的維度取決於輸入幾何的維度。更具體地說，當輸入幾何是非空的 3DM 或 3D 幾何時，會捨棄 m 座標。也就是說，傳回幾何的維度分別為 2D 或 3DZ。如果輸入是非空的 2D 或 3DZ 幾何，則產生的幾何具有相同的維度。

語法

```
ST_ConvexHull(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

凸包上的點數	幾何子類型
0	傳回 geom 的副本。
1	傳回 POINT 子類型。
2	傳回 LINESTRING 子類型。傳回 linestring 的兩個點按字典順序排序。
3 或以上	傳回沒有內環的 POLYGON 子類型。多邊形是順時針方向的，外環的第一個點是環的字典順序最小的點。

範例

以下 SQL 會傳回 linestring 的擴充已知文字 (EWKT) 表示法。在此情況下，傳回的凸包為多邊形。

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('LINESTRING(0 0,1 0,0 1,1 1,0.5 0.5)')))
as output;
```

```
output
-----
POLYGON((0 0,0 1,1 1,1 0,0 0))
```

以下 SQL 會傳回 linestring 的 EWKT 表示法。在此情況下，傳回的凸包為 linestring。

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('LINESTRING(0 0,1 1,0.2 0.2,0.6 0.6,0.5
0.5)'))) as output;
```

```
output
-----
LINESTRING(0 0,1 1)
```

以下 SQL 會傳回多點的 EWKT 表示法。在此情況下，傳回的凸包為點。

```
SELECT ST_AsEWKT(ST_ConvexHull(ST_GeomFromText('MULTIPOINT(0 0,0 0,0 0)'))) as output;
```

```
output
-----
POINT(0 0)
```

ST_CoveredBy

如 CoveredBy 果第一個輸入幾何圖形的 2D 投影被第二個輸入幾何圖形的 2D 投影覆蓋，ST_ 會傳回 true。如果兩個幾何皆非空白，且 A 中的每個點都是 B 中的點，則幾何 B 涵蓋幾何 A。

ST_CoveredBy (A,B) 相當於 ST_ 封面 (B,)。A

語法

```
ST_CoveredBy(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 *geom2* 比較，以判斷其是否涵蓋於 *geom2*。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 *geom1* 或 *geom2* 為 Null，則會傳回 Null。

如果 *geom1* 和 *geom2* 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 *geom1* 或 *geom2* 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第二個多邊形是否涵蓋第一個多邊形。

```
SELECT ST_CoveredBy(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),  
ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_coveredby  
-----  
true
```

ST_Covers

ST_Covers 會在第一個輸入幾何的 2D 投影涵蓋第二個輸入幾何的 2D 投影時傳回 true。如果兩個幾何皆非空白，且 B 中的每個點都是 A 中的點，則幾何 A 涵蓋幾何 B。

ST_封面 (A,B) 等同於 ST_CoveredBy (B,)。A

語法

```
ST_Covers(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 *geom1* 比較，以判斷其是否涵蓋 *geom1*。

傳回類型

BOOLEAN

如果 *geom1* 或 *geom2* 為 Null，則會傳回 Null。

如果 *geom1* 和 *geom2* 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 *geom1* 或 *geom2* 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否涵蓋第二個多邊形。

```
SELECT ST_Covers(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),  
ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_covers  
-----  
false
```

ST_Crosses

如果兩個輸入幾何的 2D 投影互相交叉，ST_Crosses 會傳回 true。

語法

```
ST_Crosses(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 *geom1* 或 *geom2* 為 Null，則會傳回錯誤。

如果 *geom1* 或 *geom2* 是幾何集合，則會傳回錯誤。

如果 *geom1* 和 *geom2* 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否與第二個多點交錯。在此範例中，多點與多邊形的內部和外部相交，這就是 ST_Crosses 傳回 true 的原因。

```
SELECT ST_Crosses (ST_GeomFromText('polygon((0 0,10 0,10 10,0 10,0 0))'),
  ST_GeomFromText('multipoint(5 5,0 0,-1 -1)));
```

```
st_crosses
-----
true
```

以下 SQL 會檢查第一個多邊形是否與第二個多點交錯。在此範例中，多點與多邊形的外部相交，但不與多邊形的內部相交，這就是 ST_Crosses 傳回 false 的原因。

```
SELECT ST_Crosses (ST_GeomFromText('polygon((0 0,10 0,10 10,0 10,0 0))'),
  ST_GeomFromText('multipoint(0 0,-1 -1)));
```

```
st_crosses
-----
false
```

ST_Dimension

ST_Dimension 會傳回輸入幾何的固有維度。「固有維度」是定義在幾何中子類型的維度值。

對於 3DM、3DZ 和 4D 幾何輸入，ST_Dimension 會傳回與 2D 幾何輸入相同的結果。

語法

```
ST_Dimension(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 geom 的固有維度。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的值	幾何子類型
0	在 geom 是 POINT 或 MULTIPOINT 子類型時傳回
1	在 geom 是 LINESTRING 或 MULTILINESTRING 子類型時傳回。
2	在 geom 是 POLYGON 或 MULTIPOLYGON 子類型時傳回

傳回的值	幾何子類型
0	在 geom 是空白的 GEOMETRYCOLLECTION 子類型時傳回
集合元件的最大維度	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會將四個點 LINESTRING 的已知文字 (WKT) 表示法轉換成 GEOMETRY 物件，並傳回 linestring 的維度。

```
SELECT ST_Dimension(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
29.31,77.29 29.07)'));
```

```
st_dimension
-----
1
```

ST_Disjoint

ST_Disjoint 會在兩個輸入幾何的 2D 投影沒有任何相同點時傳回 True。

語法

```
ST_Disjoint(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否與第二個多邊形斷續。

```
SELECT ST_Disjoint(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0)),(2 2,2 5,5 5,5 2,2 2)'), ST_Point(4, 4));
```

```
st_disjoint
-----
true
```

ST_Distance

對於輸入幾何，ST_Distance 會傳回兩個輸入幾何值的 2D 投影之間的最小歐幾里得距離。

對於 3DM、3DZ、4D 幾何，ST_Distance 會傳回兩個輸入幾何值的 2D 投影之間的歐幾里德距離。

對於輸入地理，ST_Distance 會傳回兩個 2D 點的測地距離。距離的單位是公尺。對於點和空點以外的地理，會傳回錯誤。

語法

```
ST_Distance(geo1, geo2)
```

引數

geo1

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。geo1 的資料類型必須與 geo2 相同。

geo2

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。geo2 的資料類型必須與 geo1 相同。

傳回類型

與輸入幾何或地理相同單位的 DOUBLE PRECISION。

如果 geo1 或 geo2 為 null 或空白，則傳回 null。

如果 geo1 和 geo2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geo1 或 geo2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會傳回兩個多邊形之間的距離。

```
SELECT ST_Distance(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
  ST_GeomFromText('POLYGON((-1 -3,-2 -1,0 -3,-1 -3))');
```

```
st_distance
-----
1.4142135623731
```

以下 SQL 使用 GEOGRAPHY 資料類型，傳回柏林的布蘭登堡門和德國國會大廈之間的距離 (以公尺為單位)。

```
SELECT ST_Distance(ST_GeogFromText('POINT(13.37761826722198 52.516411678282445)'),
  ST_GeogFromText('POINT(13.377950831464005 52.51705102546893)');
```

```
st_distance
-----
74.64129172609631
```

ST_DistanceSphere

ST_會DistanceSphere 傳回位於球面上的兩點幾何圖形之間的距離。

語法

```
ST_DistanceSphere(geom1, geom2)
```

```
ST_DistanceSphere(geom1, geom2, radius)
```

引數

geom1

球面上 GEOMETRY 資料類型的點值，單位為度。點的第一個座標是經度值。點的第二個座標是緯度值。對於 3DZ、3DM 或 4D 幾何，僅使用前兩個座標。

geom2

球面上 GEOMETRY 資料類型的點值，單位為度。點的第一個座標是經度值。點的第二個座標是緯度值。對於 3DZ、3DM 或 4D 幾何，僅使用前兩個座標。

radius

DOUBLE PRECISION 資料類型的球面半徑。如果沒有提供 radius，則球面會預設為地球，且會從橢球的全球測量系統 (WGS) 84 表示法運算半徑。

傳回類型

與半徑單位相同的 DOUBLE PRECISION。如果未提供半徑，則距離以公尺為單位。

如果 geom1 或 geom2 為 Null 或空白，則會傳回 Null。

如果沒有提供 radius，則結果的單位會是沿著地球表面的公尺數。

如果 radius 為負數，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 不是點，則會傳回錯誤。

範例

以下範例 SQL 會計算地球上兩點之間的距離 (以公里為單位)。


```
SELECT ROUND(ST_DistanceSphere(ST_Point(-122, 47), ST_Point(-122.1, 47.1))/ 1000, 0);
```

```
round
```

```
-----  
13
```

以下範例 SQL 會運算德國三個機場位置間的距離 (公里)：柏林泰格爾機場 (TXL)、慕尼黑國際機場 (MUC) 及法蘭克福國際機場 (FRA)。

```
WITH airports_raw(code,lon,lat) AS (  
  (SELECT 'MUC', 11.786111, 48.353889) UNION  
  (SELECT 'FRA', 8.570556, 50.033333) UNION  
  (SELECT 'TXL', 13.287778, 52.559722)),  
airports1(code,location) AS (SELECT code, ST_Point(lon, lat) FROM airports_raw),  
airports2(code,location) AS (SELECT * from airports1)  
SELECT (airports1.code || ' <-> ' || airports2.code) AS airports,  
round(ST_DistanceSphere(airports1.location, airports2.location) / 1000, 0) AS  
  distance_in_km  
FROM airports1, airports2 WHERE airports1.code < airports2.code ORDER BY 1;
```

airports	distance_in_km
FRA <-> MUC	299
FRA <-> TXL	432
MUC <-> TXL	480

ST_DWithin

ST_DWithin 會在兩個輸入幾何值的 2D 投影之間的歐幾里得距離不大於閾值時傳回 True。

語法

```
ST_DWithin(geom1, geom2, threshold)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

threshold

DOUBLE PRECISION 資料類型的值。這個值的單位是輸入引數。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 threshold 為負值，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查兩個多邊形間的距離是否在五個單位內。

```
SELECT ST_DWithin(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
  ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'),5);
```

```
st_dwithin
-----
true
```

ST_EndPoint

ST_EndPoint 返回輸入線串的最後一個點。結果的空間參考系統識別碼 (SRID) 值與輸入幾何的 SRID 值相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_EndPoint(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

如果 geom 不是 LINESTRING，則會傳回 Null。

範例

下列 SQL 將四點 LINESTRING 的擴充已知文字 (EWKT) 表示法傳回至 GEOMETRY 物件，並傳回 linestring 的終點。

```
SELECT ST_AsEWKT(ST_EndPoint(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0
5)',4326)));
```

```
st_asewkt
-----
SRID=4326;POINT(0 5)
```

ST_Envelope

ST_Envelope 會傳回輸入幾何的最小週框方塊，如下所示。

- 如果輸入幾何為空白，則傳回的幾何為輸入幾何的副本。
- 如果輸入幾何圖形的最小週框方塊退化為一點，則傳回的幾何為一點。
- 如果輸入幾何的最小週框方塊為一維，則會傳回兩點 linestring。
- 如果上述情況都不成立，則函數會傳回順時針方向的多邊形，其頂點是最小週框方塊的邊角。

所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

對於所有非空輸入，此函數對輸入幾何的 2D 投影進行操作。

語法

```
ST_Envelope(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 *geom* 為 Null，則會傳回 Null。

範例

下列 SQL 將四點 LINESTRING 的已知文字 (WKT) 表示法轉換為 GEOMETRY 物件，並傳回一個多邊形，其邊角是最小週框方塊。

```
SELECT ST_AsText(ST_Envelope(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0 0)),LINESTRING(20 10,20 0,10 0))')));
```

```
st_astext
```

```
-----  
POLYGON((0 0,0 10,20 10,20 0,0 0))
```

ST_Equals

ST_Equals 會在輸入幾何的 2D 投影在幾何上相等時傳回 True。如果幾何具有相同的點集合，且其內部包含非空白的交集，便可將幾何視為幾何相等。

語法

```
ST_Equals(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom1 比較，以判斷其是否與 geom1 相等。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查兩個多邊形是否幾何相等。

```
SELECT ST_Equals(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),
  ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_equals
-----
false
```

以下 SQL 會檢查兩個 linestring 是否幾何相等。

```
SELECT ST_Equals(ST_GeomFromText('LINESTRING(1 0,10 0)'), ST_GeomFromText('LINESTRING(1
  0,5 0,10 0)'));
```

```
st_equals
```

```
-----  
true
```

ST_ ExteriorRing

ST_ 會 ExteriorRing 傳回代表輸入多邊形外環的封閉線串。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_ExteriorRing(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

LINestring 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 *geom* 為 Null，則會傳回 Null。

如果 *geom* 不是多邊形，則會傳回 Null。

如果 *geom* 為空，則傳回空的多邊形。

範例

下列 SQL 會將多邊形的外環傳回為封閉的 linestring。

```
SELECT ST_AsEWKT(ST_ExteriorRing(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17  
7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12  
14,15 14,13 11,12 14))'))));
```

```
st_asewkt
```

```
-----  
LINESTRING(7 9,8 7,11 6,15 8,16 6,17 7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9)
```

ST_Force2D

ST_Force2D 會傳回輸入幾何的 2D 幾何。對於 2D 幾何，會傳回輸入的副本。對於 3DZ、3DM 和 4D 幾何，ST_Force2D 會將幾何投影至 xy 笛卡爾平面。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force2D(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 *geom* 為 Null，則會傳回 Null。

如果 *geom* 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DZ 幾何傳回 2D 幾何。

```
SELECT ST_AsEWKT(ST_Force2D(ST_GeomFromText('MULTIPOINT Z(0 1 2, EMPTY, 2 3 4, 5 6 7)')));
```

```
st_asewkt
```

```
-----  
MULTIPOINT((0 1),EMPTY,(2 3),(5 6))
```

ST_Force3D

ST_Force3D 是 ST_Force3DZ 的別名。如需詳細資訊，請參閱 [ST_Force3DZ](#)。

ST_Force3DM

ST_Force3DM 會傳回輸入幾何的 3DM 幾何。對於 2D 幾何，輸出幾何中非空點的 m 座標均設定為 0。對於 3DM 幾何，會傳回輸入幾何的副本。對於 3DZ 幾何，幾何會投影到 xy 笛卡爾平面，並且輸出幾何中非空點的 m 座標全部設定為 0。對於 4D 幾何，會將幾何投影至 xym 笛卡爾空間。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force3DM(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 *geom* 為 Null，則會傳回 Null。

如果 *geom* 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DZ 幾何傳回 3DM 幾何。

```
SELECT ST_AsEWKT(ST_Force3DM(ST_GeomFromText('MULTIPOINT Z(0 1 2, EMPTY, 2 3 4, 5 6 7)')));
```

```
st_asewkt
```



```
-----  
MULTIPOINT M ((0 1 0),EMPTY,(2 3 0),(5 6 0))
```

ST_Force3DZ

ST_Force3DZ 會從輸入幾何傳回 3DZ 幾何。對於 2D 幾何，輸出幾何中非空點的 z 座標均設定為 0。對於 3DM 幾何，幾何會投影到 xy 笛卡爾平面，並且輸出幾何中非空點的 z 座標全部設定為 0。對於 3DZ 幾何，會傳回輸入幾何的副本。對於 4D 幾何，會將幾何投影至 xyz 笛卡爾空間。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force3DZ(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DM 幾何傳回 3DZ 幾何。

```
SELECT ST_AsEWKT(ST_Force3DZ(ST_GeomFromText('MULTIPOINT M(0 1 2, EMPTY, 2 3 4, 5 6 7)')));
```

```
st_asewkt  
-----
```

```
MULTIPOINT Z ((0 1 0),EMPTY,(2 3 0),(5 6 0))
```

ST_Force4D

ST_Force4D 會傳回輸入幾何的 4D 幾何。對於 2D 幾何，輸出幾何中非空點的 z 和 m 座標均設定為 0。對於 3DM 幾何，輸出幾何中非空點的 z 座標均設定為 0。對於 3DZ 幾何，輸出幾何中非空點的 m 座標均設定為 0。對於 4D 幾何，會傳回輸入幾何的副本。輸入幾何中的空點在輸出幾何中仍然是空點。

語法

```
ST_Force4D(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 *geom* 為 Null，則會傳回 Null。

如果 *geom* 為空，則會傳回空的幾何。

範例

下列 SQL 會從 3DM 幾何傳回 4D 幾何。

```
SELECT ST_AsEWKT(ST_Force4D(ST_GeomFromText('MULTIPOINT M(0 1 2, EMPTY, 2 3 4, 5 6 7)')));
```

```
st_asewkt
-----
MULTIPOINT ZM ((0 1 0 2),EMPTY,(2 3 0 4),(5 6 0 7))
```

ST_GeoHash

ST_GeoHash 返回具有指定精確度的輸入點的 geohash 表示。預設精確度為 20。如需 geohash 定義的相關資訊，請參閱 Wikipedia 中的 [Geohash](#)。

語法

```
ST_GeoHash(geom)
```

```
ST_GeoHash(geom, precision)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

precision

INTEGER 資料類型的值。預設值為 20。

傳回類型

GEOMETRY

函數會傳回輸入點的 geohash 表示。

如果輸入點為空，該函數傳回 null。

如果輸入幾何不是點，則函數會傳回錯誤。

範例

以下 SQL 會傳回輸入點的 geohash 表示法。

```
SELECT ST_GeoHash(ST_GeomFromText('POINT(45 -45)'), 25) AS geohash;
```

```
geohash
```

```
-----  
m0000000000000000000000gzz
```

下列 SQL 傳回 null，因為輸入點是空的。

```
SELECT ST_GeoHash(ST_GeomFromText('POINT EMPTY'), 10) IS NULL AS result;
```

```
result  
-----  
true
```

ST_GeogFromText

ST_GeogFromText 構造從一個知名的文本 (WKT) 或輸入地理的擴展知名文本 (EWKT) 表示的地理對象。

語法

```
ST_GeogFromText(wkt_string)
```

引數

wkt_string

VARCHAR 資料類型的值，其為幾何的 WKT 或 EWKT 表示法。

傳回類型

GEOGRAPHY

如果 SRID 值設定為輸入中提供的值。如果未提供 SRID，則會設定為 4326。

如果 wkt_string 為 null，則傳回 null。

如果 wkt_string 無效，則會傳回錯誤。

範例

以下 SQL 會使用 SRID 值從地理物件建構多邊形。

```
SELECT ST_AsEWKT(ST_GeogFromText('SRID=4324;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))'));
```

```
st_asewkt
```

```
-----  
SRID=4324;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))
```

以下 SQL 從地理物件建構一個多邊形。SRID 的值設定為 4326。

```
SELECT ST_AsEWKT(ST_GeogFromText('POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))'));
```

```
st_asewkt
```

```
-----  
SRID=4326;POLYGON((0 0,0 1,1 1,10 10,1 0,0 0))
```

ST_ 白板 GeogFrom

ST_GeogFrom WKB 構造從輸入地理的十六進制知名二進制 (WKB) 表示的地理對象。

語法

```
ST_GeogFromWKB(wkb_string)
```

引數

wkb_string

VARCHAR 資料類型的值，其為地理的十六進位 WKB 表示法。

傳回類型

GEOGRAPHY

如果提供 SRID 值，則會將設為提供的值。如果未提供 SRID，則會設定為 4326。

如果 wkb_string 為 null，則傳回 null。

傳回類型

GEOMETRY

如果 `geom` 或 `index` 為 `Null`，則會傳回 `Null`。

如果 `index` 超出範圍，則會傳回錯誤。

範例

下列 SQL 會傳回幾何集合中的幾何。

```
WITH tmp1(idx) AS (SELECT 1 UNION SELECT 2),
tmp2(g) AS (SELECT ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0
0)),LINESTRING(20 10,20 0,10 0))')
SELECT idx, ST_AsEWKT(ST_GeometryN(g, idx)) FROM tmp1, tmp2 ORDER BY idx;
```

idx	st_asewkt
1	POLYGON((0 0,10 0,0 10,0 0))
2	LINESTRING(20 10,20 0,10 0)

ST_GeometryType

ST_ 以字串形式 `GeometryType` 傳回輸入幾何的子類型。

對於 3DM、3DZ 和 4D 幾何輸入，ST_ 會 `GeometryType` 傳回與 2D 幾何輸入相同的結果。

語法

```
ST_GeometryType(geom)
```

引數

`geom`

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

VARCHAR 表示 `geom` 的子類型。

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的字串值	幾何子類型
ST_Point	在 geom 是 POINT 子類型時傳回
ST_LineString	在 geom 是 LINESTRING 子類型時傳回
ST_Polygon	在 geom 是 POLYGON 子類型時傳回
ST_MultiPoint	在 geom 是 MULTIPOINT 子類型時傳回
ST_MultiLineString	在 geom 是 MULTILINESTRING 子類型時傳回
ST_MultiPolygon	在 geom 是 MULTIPOLYGON 子類型時傳回
ST_GeometryCollection	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會傳回輸入 linestring 幾何的子類型。

```
SELECT ST_GeometryType(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
29.31,77.29 29.07)'));
```

```
st_geometrytype
-----
ST_LineString
```

ST_EWKB GeomFrom

ST_GeomFrom EWKB 構造從一個輸入幾何的擴展眾所周知的二進制 (EWKB) 表示的幾何對象。

ST_GeomFrom EWKB 接受以 WKB 和 EWKB 十六進位格式撰寫的 3DZ、3 分 M 和 4D 幾何圖形。

引數

ewkt_string

VARCHAR 資料類型的值，或計算結果為 VARCHAR 類型的運算式，也就是幾何的 EWKT 表示法。

您可以使用 WKT 關鍵字 EMPTY 來指定空點、具有空點的多點或具有空點的幾何集合。以下範例會建立空點。

```
ST_GeomFromEWKT('SRID=4326;POINT EMPTY');
```

傳回類型

GEOMETRY

如果 ewkt_string 為 null，則傳回 null。

如果 ewkt_string 無效，則會傳回錯誤。

範例

下列 SQL 會從 EWKT 值建構 multilinestring，並傳回幾何。它也會傳回幾何的 ST_AsEWKT 結果。

```
SELECT ST_GeomFromEWKT('SRID=4326;MULTILINESTRING((1 0,1 0),(2 0,3 0),(4 0,5 0,6 0))')
as geom, ST_AsEWKT(geom);
```

geom

st_asewkt

```
+-----+
| SRID=4326;MULTILINESTRING((1 0,1 0),(2 0,3 0),(4 0,5 0,6 0))
```

ST_GeomFromGeoHash

ST_ 從輸入幾何的 Geohash 表示法 GeomFromGeoHash 構造一個幾何對象。ST_ 會 GeomFromGeoHash 傳回空間參考識別碼 (SRID) 為零 (0) 的二維 (2D) 幾何圖形。如需 geohash 格式的相關資訊，請參閱 Wikipedia 中的 [Geohash](#)。

語法

```
ST_GeomFromGeoHash(geohash_string)
```

```
ST_GeomFromGeoHash(geohash_string, precision)
```

引數

geohash_string

VARCHAR 資料類型的值，或計算結果為 VARCHAR 類型的運算式，也就是幾何的 geohash 表示法。

precision

INTEGER 資料類型的值，代表 geohash 的精確度。該值是用作精確度的 geohash 的字元數。如果未指定該值，則小於零或大於 *geohash_string* 長度，則會使用 *geohash_string* 長度。

傳回類型

GEOMETRY

如果 *geohash_string* 為 null，則傳回 null。

如果 *geohash_string* 無效，則會傳回錯誤。

範例

下列 SQL 會傳回高精確度的多邊形。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxcgyy4d0dbxqz0'));
```

```
st_asewkt
```

```
-----  
POLYGON((-115.172816 36.114646,-115.172816 36.114646,-115.172816 36.114646,-115.172816  
36.114646,-115.172816 36.114646))
```

以下 SQL 會傳回高精確度的點。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxcggy4d0dbxqz00'));
```

```
st_asewkt
-----
POINT(-115.172816 36.114646)
```

下列 SQL 會傳回低精確度的多邊形。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qq'));
```

```
st_asewkt
-----
POLYGON((-115.3125 35.15625,-115.3125 36.5625,-113.90625 36.5625,-113.90625
35.15625,-115.3125 35.15625))
```

下列 SQL 會傳回精確度為 3 的多邊形。

```
SELECT ST_AsText(ST_GeomFromGeoHash('9qqj7nmxcggy4d0dbxqz0', 3));
```

```
st_asewkt
-----
POLYGON((-115.3125 35.15625,-115.3125 36.5625,-113.90625 36.5625,-113.90625
35.15625,-115.3125 35.15625))
```

JSON GeomFromGeo

ST_GeomFromGeo JSON 構造從輸入幾何的 GeoJSON 表示形式的幾何對象。如需 GeoJSON 格式的相關資訊，請參閱 Wikipedia 中的 [GeoJSON](#)。

如果至少有一個點具有三個或更多座標，則產生的幾何圖形為 3DZ，其中對於僅具有兩個座標的點，Z 分量為零。如果輸入 GeoJSON 中的所有點都包含兩個坐標或為空，則 ST_GeomFromGeo JSON 返回一個二維幾何圖形。傳回幾何的空間參考系統識別碼 (SRID) 一律為 4326。

語法

```
ST_GeomFromGeoJSON(geojson_string)
```

引數

geojson_string

VARCHAR 資料類型的值，或計算結果為 VARCHAR 類型的運算式，也就是幾何的 GeoJSON 表示法。

傳回類型

GEOMETRY

如果 *geojson_string* 為 null，則傳回 null。

如果 *geojson_string* 無效，則會傳回錯誤。

範例

以下 SQL 傳回輸入 GeoJSON 中表示的二維幾何。

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"Point","coordinates":[1,2]}'));
```

```
st_asewkt
```

```
-----  
SRID=4326;POINT(1 2)
```

以下 SQL 傳回輸入 GeoJSON 中表示的 3DZ 幾何。

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{"type":"LineString","coordinates":[[1,2,3],[4,5,6],[7,8,9]]}'));
```

```
st_asewkt
```

```
-----  
SRID=4326;LINESTRING Z (1 2 3,4 5 6,7 8 9)
```

當輸入 GeoJSON 中只有一個點具有三個座標而所有其他點都具有兩個座標時，以下 SQL 傳回 3DZ 幾何。

```
SELECT ST_AsEWKT(ST_GeomFromGeoJSON('{ "type": "Polygon", "coordinates": [[[0, 0],[0, 1, 8],[1, 0],[0, 0]]]}'));
```

```
st_asewkt
-----
SRID=4326;POLYGON Z ((0 0 0,0 1 8,1 0 0,0 0 0))
```

ST_GeomFromGeoSquare

ST_GeomFromGeoSquare 傳回涵蓋由輸入土平方值所表示區域的幾何圖形。傳回的幾何永遠是二維的。若要計算 geosquare 值，請參閱 [ST_GeoSquare](#)。

語法

```
ST_GeomFromGeoSquare(geosquare)
```

```
ST_GeomFromGeoSquare(geosquare, max_depth)
```

引數

geosquare

BIGINT 資料類型的值或計算結果為 geosquare 值之 BIGINT 類型的運算式，該值描述在初始域上進行的細分序列以達到所需的平方。此值由 [ST_GeoSquare](#) 計算。

max_depth

INTEGER 資料類型的值，代表對初始域進行的域細分的最大數量。值必須等於或大於 1。

傳回類型

GEOMETRY

如果 geosquare 無效，則函數會傳回錯誤。

如果輸入 `max_depth` 不在範圍內，則函數會傳回錯誤。

範例

下列 SQL 會從 `geosquare` 值傳回幾何。

```
SELECT ST_AsText(ST_GeomFromGeoSquare(797852));
```

```
st_astext
```

```
-----
POLYGON((13.359375 52.3828125,13.359375 52.734375,13.7109375 52.734375,13.7109375
52.3828125,13.359375 52.3828125))
```

下列 SQL 會從 `geosquare` 值和最大深度 3 傳回幾何。

```
SELECT ST_AsText(ST_GeomFromGeoSquare(797852, 3));
```

```
st_astext
```

```
-----
POLYGON((0 45,0 90,45 90,45 45,0 45))
```

以下 SQL 首先透過指定 `x` 座標為經度、`y` 座標為緯度 (-122.3, 47.6) 來計算西雅圖的 `geosquare` 值。然後它傳回 `geosquare` 的多邊形。雖然輸出是二維幾何，但它可以用來計算經度和緯度的空間資料。

```
SELECT ST_AsText(ST_GeomFromGeoSquare(ST_GeoSquare(ST_Point(-122.3, 47.6))));
```

```
st_astext
```

```
-----
POLYGON((-122.335167014971 47.6080129947513,-122.335167014971
47.6080130785704,-122.335166931152 47.6080130785704,-122.335166931152
47.6080129947513,-122.335167014971 47.6080129947513))
```

ST_GeomFromText

`ST_` 從輸入幾何圖形的已知文字 (WKT) 表現法 `GeomFromText` 建構幾何圖形物件。

ST_GeomFromText 接受 3DZ、3DM 和 4D，其中幾何類型分別以 Z、M 或 ZM 作為前綴。

語法

```
ST_GeomFromText(wkt_string)
```

```
ST_GeomFromText(wkt_string, srid)
```

引數

wkt_string

VARCHAR 資料類型的值，其為幾何的 WKT 表示法。

您可以使用 WKT 關鍵字 EMPTY 來指定空點、具有空點的多點或具有空點的幾何集合。下列範例使用一個空點和一個非空點建立多點。

```
ST_GeomFromEWKT('MULTIPOINT(1 0,EMPTY)');
```

srid

INTEGER 資料類型的值，其為空間參考識別碼 (SRID)。如果有提供 SRID 值，則傳回的幾何便會包含此 SRID 值。否則，傳回幾何的 SRID 值會設為零 (0)。

傳回類型

GEOMETRY

如果 wkt_string 或 srid 為 Null，則會傳回 Null。

如果 srid 為負值，則會傳回 Null。

如果 wkt_string 無效，則會傳回錯誤。

如果 srid 無效，則會傳回錯誤。

範例

以下 SQL 會從 WKT 表示法和 SRID 值建構幾何物件。


```
ST_GeoSquare(geom, max_depth)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 POINT 子類型的運算式。點的 x 座標 (經度) 必須在以下範圍內：-180 - 180。點的 y 座標 (緯度) 必須在以下範圍內：-90 - 90。

max_depth

INTEGER 資料類型的值。包含該點的域被遞歸細分的最大次數。值必須為介於 1 到 32 之間的整數。預設值為 32。子實際最終的細分數小於或等於指定的 max_depth。

傳回類型

BIGINT

此函數傳回一個唯一值，用於識別輸入點所在的最終 geosquare。

如果輸入 geom 不是點，則函數會傳回錯誤。

如果輸入點為空，則傳回值對 [ST_GeomFromGeoSquare](#) 函數不是有效輸入。使用該 [ST_IsEmpty](#) 函數可防止以空點對 ST_GeoSquare 的調用。

如果輸入點不在範圍內，則函數會傳回錯誤。

如果輸入 max_depth 超出範圍，則函數會傳回錯誤。

範例

以下 SQL 會從輸入點傳回 geosquare。

```
SELECT ST_GeoSquare(ST_Point(13.5, 52.5));
```

```
st_geosquare
```

```
-----  
-4410772491521635895
```

下列 SQL 從最大深度為 10 的輸入點傳回 geosquare。

```
SELECT ST_GeoSquare(ST_Point(13.5, 52.5), 10);
```

```
st_geosquare
-----
797852
```

ST_ N InteriorRing

ST_ InteriorRing N 會傳回對應於索引位置處輸入多邊形內環的封閉線串。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_InteriorRingN(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

索引

INTEGER 資料類型的值，代表以一開始索引的環形位置。

傳回類型

LINestring 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom 或 index 為 Null，則會傳回 Null。

如果 index 超出範圍，則會傳回 Null。

如果 geom 不是多邊形，則會傳回 Null。

如果 geom 為空多邊形，則傳回 null。

範例

下列 SQL 會將多邊形的第二個環傳回為封閉的 linestring。

```
SELECT ST_AsEWKT(ST_InteriorRingN(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17
7,17 10,18 12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12
14,15 14,13 11,12 14))'),2));
```

```
st_asewkt
-----
LINESTRING(12 14,15 14,13 11,12 14)
```

ST_Intersects

ST_Intersects 會在兩個輸入幾何的 2D 投影至少擁有一個相同點時傳回 True。

語法

```
ST_Intersects(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否和第二個多邊形交集。

```
SELECT ST_Intersects(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(2 2,2 5,5 5,5 2,2 2)'), ST_GeomFromText('MULTIPOINT((4 4),(6 6))'));
```

```
st_intersects
-----
true
```

ST_Intersection

ST_Intersection 會傳回代表兩個幾何的點集交集的幾何。也就是說，它會傳回兩個輸入幾何之間共用的部分。

語法

```
ST_Intersection(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

如果 geom1 和 geom2 不共用任何空間 (它們不相交)，則傳回空的幾何。

如果 geom1 或 geom2 為空，則會傳回空的幾何。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

如果 geom1 或 geom2 不是二維 (2D) 幾何，則會傳回錯誤。

範例

下列 SQL 會傳回表示兩個輸入幾何交集的非空幾何。

```
SELECT ST_AsEWKT(ST_Intersection(ST_GeomFromText('polygon((0 0,100 100,0 200,0 0))'),
  ST_GeomFromText('polygon((0 0,10 0,0 10,0 0))')));
```

```
      st_asewkt
-----
POLYGON((0 0,0 10,5 5,0 0))
```

當傳遞不連接 (非相交) 的輸入幾何時，下列 SQL 會傳回空的幾何。

```
SELECT ST_AsEWKT(ST_Intersection(ST_GeomFromText('linestring(0 100,0 0)'),
  ST_GeomFromText('polygon((1 0,10 0,1 10,1 0))')));
```

```
      st_asewkt
-----
LINESTRING EMPTY
```

ST_逆時鐘 IsPolygon

ST_IsPolygon CCW 如果輸入多邊形或多重重多邊形的 2D 投影是逆時針方向返回 true。如果輸入幾何為點、linestring、多點或 multilinestring，則傳回 true。對於幾何圖形集合，如果集合中的所有幾何圖形都是逆時鐘方向，ST_IsPolygon CCW 會傳回 true。

語法

```
ST_IsPolygonCCW(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否逆時針方向。

```
SELECT ST_IsPolygonCCW(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17 7,17 10,18
12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12 14,15 14,13
11,12 14)))'));
```

```
st_ispolygonccw
-----
true
```

ST_ 密西西比 IsPolygon

如果輸入多邊形或多重重多邊形的 2D 投影是順時針方向，則 ST_IsPolygon CW 返回 true。如果輸入幾何為點、linestring、多點或 multilinestring，則傳回 true。對於幾何圖形集合，如果集合中的所有幾何圖形均為順時針方向，則 ST_IsPolygon CW 會傳回 true。

語法

```
ST_IsPolygonCW(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否順時針方向。

```
SELECT ST_IsPolygonCW(ST_GeomFromText('POLYGON((7 9,8 7,11 6,15 8,16 6,17 7,17 10,18
12,17 14,15 15,11 15,10 13,9 12,7 9),(9 9,10 10,11 11,11 10,10 8,9 9),(12 14,15 14,13
11,12 14))')));
```

```
st_ispolygonccw
```

```
-----
```

```
true
```

ST_IsClosed

如果關閉輸入幾何圖形的 2D 投影，ST_IsClosed 傳回 true。下列規則定義了封閉幾何：

- 輸入幾何是點或 multipoint。
- 輸入幾何是 linestring，且 linestring 的起始點和結束點一致。
- 輸入幾何是非空白的 multilinestring，且其所有的 linestring 都是封閉的。
- 輸入幾何是非空白的多邊形，所有多邊形的環都是非空白的，且其所有環的起始點和結束點都一致。
- 輸入幾何是非空白的 multipolygon，且其所有的多邊形都是封閉的。
- 輸入幾何是非空白的幾何集合，且其所有的元件都是封閉的。

語法

```
ST_IsClosed(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 是空點，則傳回 false。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否是封閉的。

```
SELECT ST_IsClosed(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
stisclosed
-----
true
```

ST_IsCollection

如果輸入幾何是下列其中一種子類型，ST_ 會IsCollection 傳回

true : GEOMETRYCOLLECTION、MULTIPOINTMULTILINESTRING、或。MULTIPOLYGON

語法

```
ST_IsCollection(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查多邊形是否是集合。

```
SELECT ST_IsCollection(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_iscollection
-----
false
```

ST_IsEmpty

ST_ 如果輸入幾何圖形為空，則IsEmpty 返回 true。如果幾何至少包含一個非空點，則該幾何不是空的。

ST_ 如果輸入幾何至少有一個非空點，則IsEmpty 返回 true。

語法

```
ST_IsEmpty(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會檢查指定多邊形是否空白。

```
SELECT ST_IsEmpty(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'));
```

```
st_isempty
-----
false
```

ST_IsRing

ST_ 如果輸入線串是一個環，則IsRing 返回 true。如果 linestring 是封閉且簡單的，則為環。

語法

```
ST_IsRing(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。幾何必須是 LINESTRING。

傳回類型

BOOLEAN

如果 geom 不是 LINESTRING，則會傳回錯誤。

範例

下列 SQL 會檢查指定的 linestring 是否為環。

```
SELECT ST_IsRing(ST_GeomFromText('linestring(0 0, 1 1, 1 2, 0 0)'));
```

```
st_isring
-----
true
```

ST_IsSimple

如果輸入幾何圖形的 2D 投影很簡單，則 ST_ 會IsSimple 傳回 true。如需簡單幾何定義的相關資訊，請參閱[幾何簡單性](#)。

語法

```
ST_IsSimple(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會檢查指定的 linestring 是否簡單。在這個範例中，它並不簡單，因為它具有自我相交。

```
SELECT ST_IsSimple(ST_GeomFromText('LINESTRING(0 0,10 0,5 5,5 -5)'));
```

```
st_issimple
-----
false
```

ST_IsValid

如果輸入幾何圖形的 2D 投影有效，ST_ 會IsValid 傳回 true。如需有效幾何定義的相關資訊，請參閱[幾何有效性](#)。

語法

```
ST_IsValid(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 `geom` 為 `Null`，則會傳回 `Null`。

範例

以下 SQL 會檢查指定多邊形是否有效。在此範例中，多邊形無效，因為多邊形的內部並未簡單地連接。

```
SELECT ST_IsValid(ST_GeomFromText('POLYGON((0 0,10 0,10 10,0 10,0 0),(5 0,10 5,5 10,0 5,5 0))'));
```

```
st_isvalid
-----
false
```

ST_Length

對於線性幾何，`ST_Length` 傳回 2D 投影的笛卡爾長度。長度單位與表示輸入幾何座標的單位相同。對於點、multipoint 和面積幾何，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中幾何的長度總和。

對於地理，`ST_Length` 會傳回在球體 (由 SRID 決定) 上所計算輸入線性地理之 2D 投影的測地線長度。長度的單位是公尺。對於點、multipoint 和面積地理，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中地理的長度總和。

語法

```
ST_Length(geo)
```

引數

`geo`

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

DOUBLE PRECISION

如果 `geo` 為 `null`，則傳回 `null`。

如果找不到 SRID 值，則會傳回錯誤。

範例

下列 SQL 會傳回 multilinestring 的笛卡爾長度。

```
SELECT ST_Length(ST_GeomFromText('MULTILINESTRING((0 0,10 0,0 10),(10 0,20 0,20 10))'));
```

```
st_length
```

```
-----
```

```
44.142135623731
```

以下 SQL 會傳回幾何中 linestring 的長度。

```
SELECT ST_Length(ST_GeogFromText('SRID=4326;LINESTRING(5 0,6 0,4 0)'));
```

```
st_length
```

```
-----
```

```
333958.472379804
```

下列 SQL 會傳回地理中某個點的長度。

```
SELECT ST_Length(ST_GeogFromText('SRID=4326;POINT(4 5)'));
```

```
st_length
```

```
-----
```

```
0
```

ST_LengthSphere

ST_LengthSphere 傳回以公尺為單位的線性幾何圖形的長度。對於點、多點和面積幾何圖形，ST_LengthSphere 會傳回 0。對於幾何圖形集合，ST_LengthSphere 傳回集合中線性幾何圖形的總長度 (以公尺為單位)。

ST_LengthSphere 會將輸入幾何圖形的每個點的座標解譯為經度和緯度 (以度為單位)。對於 3DZ、3DM 或 4D 幾何，僅使用前兩個座標。

語法

```
ST_LengthSphere(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

DOUBLE PRECISION 長度的單位是公尺。長度計算是根據地球的球形模型，其半徑是世界大地測量系統 (WGS) 84 地球橢球模型的地球平均半徑。

如果 *geom* 為 Null，則會傳回 Null。

範例

下列範例 SQL 會計算 linestring 長度 (以公尺為單位)。

```
SELECT ST_LengthSphere(ST_GeomFromText('LINESTRING(10 10,45 45)'));
```

```
st_lengthsphere
-----
5127736.08292556
```

ST_Length2D

ST_Length2D 是 ST_Length 的別名。如需詳細資訊，請參閱 [ST_Length](#)。

ST_LineFromMultiPoint

ST_LineFromMultiPoint 會從輸入多點幾何圖形傳回線串。點的順序會保留下來。所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_LineFromMultiPoint(geom)
```


引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 MULTIPOINT。

傳回類型

GEOMETRY

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則傳回 LINESTRING。

如果 geom 包含空點，則會忽略這些空點。

如果 geom 不是 MULTIPOINT，則會傳回錯誤。

範例

下列 SQL 會從 multipoint 建立 linestring。

```
SELECT ST_AsEWKT(ST_LineFromMultiPoint(ST_GeomFromText('MULTIPOINT(0 0,10 0,10 10,5 5,0 5)',4326)));
```

```
st_asewkt
```

```
-----  
SRID=4326;LINESTRING(0 0,10 0,10 10,5 5,0 5)
```

ST_LineInterpolatePoint

ST_ 沿著直線LineInterpolatePoint 返回一個點，距離直線的起點有分數的距離。

若要確定點相等，ST_ 會對輸入幾何圖形的 2D 投影進LineInterpolatePoint 行操作。如果輸入幾何為空，則會以與輸入相同的維度傳回其副本。對於 3DZ、3DM 和 4D 幾何，z 或 m 座標是點所在區段的 z 或 m 座標的平均值。

語法

```
ST_LineInterpolatePoint(geom, fraction)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型是 LINESTRING。

fraction

DOUBLE PRECISION 資料類型的值，表示沿線的 linestring 的點位置。該值是介於 0-1 (含) 範圍內的分數。

傳回類型

POINT 子類型的 GEOMETRY。

如果 geom 或 fraction 為 null，則傳回 null。

如果 geom 為空，則傳回空點。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 fraction 超出範圍，則會傳回錯誤。

如果 geom 不是 linestring，則會傳回錯誤。

範例

下列 SQL 會傳回沿 linestring 中間的點。

```
SELECT ST_AsEWKT(ST_LineInterpolatePoint(ST_GeomFromText('LINESTRING(0 0, 5 5, 7 7, 10 10)'), 0.50));
```

```
st_asewkt  
-----  
POINT(5 5)
```

下列 SQL 會傳回沿 linestring 90% 的點。

```
SELECT ST_AsEWKT(ST_LineInterpolatePoint(ST_GeomFromText('LINESTRING(0 0, 5 5, 7 7, 10 10)'), 0.90));
```

```
st_asewkt
-----
POINT(9 9)
```

ST_M

ST_M 會傳回輸入點的 m 座標。

語法

```
ST_M(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

m 座標的 DOUBLE PRECISION 值。

如果 *point* 是 Null，則會傳回 Null。

如果 *point* 是 2D 或 3DZ 點，則傳回 null。

如果 *point* 是空點，則傳回 null。

如果 *point* 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回 3DZ 幾何中點的 m 座標。

```
SELECT ST_M(ST_GeomFromEWKT('POINT M (1 2 3)'));
```

```
st_m
-----
3
```

以下 SQL 會傳回 4D 幾何中點的 m 座標。

```
SELECT ST_M(ST_GeomFromEWKT('POINT ZM (1 2 3 4)'));
```

```
st_m  
-----  
4
```

ST_MakeEnvelope

ST_MakeEnvelope 會傳回幾何圖形，如下所示：

- 如果輸入座標指定了一個點，則傳回的幾何為一個點。
- 如果輸入座標指定了直線，則傳回的幾何為 linestring。
- 否則，傳回的幾何為多邊形，其中輸入座標會指定方塊的左下角和右上角。

如果提供，則傳回幾何的空間參考系統識別碼 (SRID) 值將設定為輸入 SRID 值。

語法

```
ST_MakeEnvelope(xmin, ymin, xmax, ymax)
```

```
ST_MakeEnvelope(xmin, ymin, xmax, ymax, srid)
```

引數

xmin

DOUBLE PRECISION 資料類型的值。此值是方塊左下角的第一個座標。

ymin

DOUBLE PRECISION 資料類型的值。此值是方塊左下角的第二個座標。

xmax

DOUBLE PRECISION 資料類型的值。此值為方塊右上角的第一個座標。

ymax

DOUBLE PRECISION 資料類型的值。此值為方塊的右上角的第二個座標。

srid

INTEGER 資料類型的值，表示空間參考系統識別碼 (SRID)。如果未提供 SRID 值，則會將其設定為零。

傳回類型

POINT、LINESTRING 或 POLYGON 子類型的 GEOMETRY。

如果未設定 srid，則傳回幾何的 SRID 會設定為 srid 或零。

如果 xmin、ymin、xmax、ymax 或 srid 為 null，則傳回 null。

如果 srid 為負值，則會傳回錯誤。

範例

下列 SQL 會傳回一個多邊形，表示由四個輸入座標值定義的包絡線。

```
SELECT ST_AsEWKT(ST_MakeEnvelope(2,4,5,7));
```

```
st_astext
-----
POLYGON((2 4,2 7,5 7,5 4,2 4))
```

以下 SQL 會傳回一個多邊形，表示由四個輸入座標值和 SRID 值定義的包絡線。

```
SELECT ST_AsEWKT(ST_MakeEnvelope(2,4,5,7,4326));
```

```
st_astext
-----
SRID=4326;POLYGON((2 4,2 7,5 7,5 4,2 4))
```

ST_MakeLine

ST_ 從輸入幾何圖 MakeLine 形建立線串。

傳回幾何的維度與輸入幾何的維度相同。兩個輸入幾何必須具有相同的維度。

語法

```
ST_MakeLine(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT、LINESTRING 或 MULTIPOINT。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT、LINESTRING 或 MULTIPOINT。

傳回類型

LINESTRING 子類型的 GEOMETRY。

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 是空點或包含空點，則忽略這些空點。

如果 geom1 和 geom2 為空，則傳回空的 LINESTRING。

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom1 和 geom2 具有不同的 SRID 值，則會傳回錯誤。

如果 geom1 或 geom2 不是 POINT、LINESTRING 或 MULTIPOINT，則會傳回錯誤。

如果 geom1 和 geom2 具有不同的維度，則會傳回錯誤。

範例

以下 SQL 會從兩個輸入 linestring 建構 linestring。

```
SELECT ST_MakeLine(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27  
29.31,77.29 29.07)'), ST_GeomFromText('LINESTRING(88.29 39.07,88.42 39.26,88.27  
39.31,88.29 39.07)'));
```

```
st_makeline
```

```
-----
```

```
010200000008000000C3F5285C8F52534052B81E85EB113D407B14AE47E15A5340C3F5285C8F423D40E17A14AE4751
```

ST_MakePoint

ST_ 會MakePoint 傳回座標值為輸入值的點幾何圖形。

語法

```
ST_MakePoint(x, y)
```

```
ST_MakePoint(x, y, z)
```

```
ST_MakePoint(x, y, z, m)
```

引數

x

DOUBLE PRECISION 資料類型的值，表示第一個座標。

y

DOUBLE PRECISION 資料類型的值，表示第二個座標。

z

DOUBLE PRECISION 資料類型的值，表示第三個座標。

m

DOUBLE PRECISION 資料類型的值，表示第四個座標。

傳回類型

POINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值會設為 0。

如果 x、y、z 或 m 為 null，則傳回 null。

範例

以下 SQL 會傳回 POINT 子類型的 GEOMETRY 類型，其中包含提供的座標。

```
SELECT ST_AsText(ST_MakePoint(1,3));
```

```
st_astext
-----
POINT(1 3)
```

以下 SQL 會傳回 POINT 子類型的 GEOMETRY 類型，其中包含提供的座標。

```
SELECT ST_AsEWKT(ST_MakePoint(1, 2, 3));
```

```
st_asewkt
-----
POINT Z (1 2 3)
```

以下 SQL 會傳回 POINT 子類型的 GEOMETRY 類型，其中包含提供的座標。

```
SELECT ST_AsEWKT(ST_MakePoint(1, 2, 3, 4));
```

```
st_asewkt
-----
POINT ZM (1 2 3 4)
```

ST_MakePolygon

ST_MakePolygon 有兩個返回多邊形的變體。一個採用單一幾何，另一個採用兩個幾何。

- 第一個變體的輸入是定義輸出多邊形外環的 linestring。
- 第二個變體的輸入是 linestring 和 multilinestring。這兩個都是空的或封閉的。

輸出多邊形的外環的邊界是輸入 `linestring`，而多邊形內環的邊界是輸入 `multilinestring` 中的 `linestring`。如果輸入 `linestring` 為空，則會傳回空多邊形。會忽略 `multilinestring` 中的空 `linestring`。產生幾何的空間參考系統識別碼 (SRID) 是兩個輸入幾何的公共 SRID。

傳回幾何的維度與輸入幾何的維度相同。外環和內環必須具有相同的維度。

語法

```
ST_MakePolygon(geom1)
```

```
ST_MakePolygon(geom1, geom2)
```

引數

`geom1`

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。linestring 的值必須是封閉或空的。

`geom2`

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 MULTILINESTRING。

傳回類型

POLYGON 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 等於輸入的 SRID。

如果 `geom1` 或 `geom2` 為 null，則傳回 null。

如果 `geom1` 不是線串，則會傳回錯誤。

如果 `geom2` 不是 multilinestring，則會傳回錯誤。

如果 `geom1` 未封閉，則會傳回錯誤。

如果 `geom1` 是單點或未封閉，則會傳回錯誤。

如果 geom2 包含至少一個具有單一點或未封閉的 linestring，則會傳回錯誤。

如果 geom1 和 geom2 具有不同的 SRID 值，則會傳回錯誤。

如果 geom1 和 geom2 具有不同的維度，則會傳回錯誤。

範例

以下 SQL 會從輸入 linestring 傳回多邊形。

```
SELECT ST_AsText(ST_MakePolygon(ST_GeomFromText('LINESTRING(77.29 29.07,77.42
29.26,77.27 29.31,77.29 29.07)')));
```

```
st_astext
-----
POLYGON((77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07))
```

下列 SQL 會從封閉 linestring 和封閉 multilinestring 建立多邊形。Linestring 用於多邊形的外環。Multilinestring 中的 linestring 用於多邊形的內環。

```
SELECT ST_AsEWKT(ST_MakePolygon(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,0 10,0 0)'),
ST_GeomFromText('MULTILINESTRING((1 1,1 2,2 1,1 1),(3 3,3 4,4 3,3 3)'))));
```

```
st_astext
-----
POLYGON((0 0,10 0,10 10,0 10,0 0),(1 1,1 2,2 1,1 1),(3 3,3 4,4 3,3 3))
```

ST_MemSize

ST_MemSize 返回輸入幾何使用的存儲器空間量 (以字節為單位)。此大小取決於幾何的 Amazon Redshift 內部表示法，因此可能會在內部表示法變更時變更。您可以使用此大小，指出 Amazon Redshift 中幾何物件的相對大小。

語法

```
ST_MemSize(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 geom 的固有維度。

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會傳回幾何集合的記憶體大小。

```
SELECT ST_MemSize(ST_GeomFromText('GEOMETRYCOLLECTION(POLYGON((0 0,10 0,0 10,0 0)),LINESTRING(20 10,20 0,10 0))'))::varchar + ' bytes';
```

```
?column?
```

```
-----  
172 bytes
```

ST_MMax

ST_MMax 會傳回輸入幾何的 m 座標上限。

語法

```
ST_MMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

m 座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DZ 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DM 幾何中 linestring 的最大 m 座標。

```
SELECT ST_MMax(ST_GeomFromEWKT('LINESTRING M (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_mmax  
-----  
      8
```

以下 SQL 會傳回 4D 幾何中線串的最大 m 座標。

```
SELECT ST_MMax(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_mmax  
-----  
     11
```

ST_MMin

ST_MMin 會傳回輸入幾何的 m 座標下限。

語法

```
ST_MMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

m 座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DZ 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DM 幾何中 linestring 的最小 m 座標。

```
SELECT ST_MMin(ST_GeomFromEWKT('LINESTRING M (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_mmin  
-----  
2
```

以下 SQL 會傳回 4D 幾何中 linestring 的 m 座標下限。

```
SELECT ST_MMin(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_mmin  
-----  
3
```

ST_Multi

ST_Multi 會將幾何轉換為對應的多重類型。如果輸入幾何已經是多重類型或幾何集合，則會傳回其副本。如果輸入幾何是點、linestring 或多邊形，則會分別傳回包含輸入幾何的多點、multilinestring 或多重多邊形。

語法

```
ST_Multi(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

具有子類型 MULTIPOINT、MULTILINESTRING、MULTIPOLYGON 或 GEOMETRYCOLLECTION 的 GEOMETRY。

所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會從輸入多點傳回一個多點。

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('MULTIPOINT((1 2),(3 4))', 4326)));
```

```
st_asewkt
```

```
-----  
SRID=4326;MULTIPOINT((1 2),(3 4))
```

以下 SQL 會從輸入點傳回多點。

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('POINT(1 2)', 4326)));
```

```
st_asewkt
```

```
-----  
SRID=4326;MULTIPOINT((1 2))
```

下列 SQL 會從輸入幾何集合傳回幾何集合。

```
SELECT ST_AsEWKT(ST_Multi(ST_GeomFromText('GEOMETRYCOLLECTION(POINT(1 2),MULTIPOINT((1 2),(3 4)))', 4326)));
```

```
st_asewkt
```

```
-----  
SRID=4326;GEOMETRYCOLLECTION(POINT(1 2),MULTIPOINT((1 2),(3 4)))
```

ST_NDims

ST_NDims 會傳回幾何的座標維度。ST_NDims 不會考慮幾何的拓撲維度。相反地，它會根據幾何的維度傳回常數值。

語法

```
ST_NDims(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 *geom* 的固有維度。

如果 *geom* 為 Null，則會傳回 Null。

傳回的值如下。

傳回的值	輸入幾何的維度
2	2D
3	3DZ 或 3DM
4	4D

範例

下列 SQL 會傳回 2D linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING(0 0,1 1,2 2,0 0)'));
```

```
st_ndims
```

```
-----
```

```
2
```

以下 SQL 會傳回 3DZ linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING Z(0 0 3,1 1 3,2 2 3,0 0 3)'));
```

```
st_ndims
```

```
-----
```

```
3
```

下列 SQL 會傳回 3DM linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING M(0 0 4,1 1 4,2 2 4,0 0 4)'));
```

```
st_ndims
```

```
-----
```

```
3
```

下列 SQL 會傳回 4D linestring 的維度數。

```
SELECT ST_NDims(ST_GeomFromText('LINESTRING ZM(0 0 3 4,1 1 3 4,2 2 3 4,0 0 3 4)'));
```

```
st_ndims
```

```
-----
```

```
4
```

ST_NPoints

ST_NPoints 會回輸入幾何或地理中非空點的數量。

語法

```
ST_NPoints(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

INTEGER

如果 *geo* 是空點，則傳回 0。

如果 *geo* 為 null，則傳回 null。

範例

以下 SQL 會傳回 linestring 中點的數量。

```
SELECT ST_NPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
```

```
st_npoints
-----
4
```

以下 SQL 會傳回地理中 linestring 中點的數量。

```
SELECT ST_NPoints(ST_GeogFromText('LINESTRING(110 40, 2 3, -10 80, -7 9)'));
```

```
st_npoints
-----
4
```

ST_NRings

ST_NRings 會傳回輸入幾何中的環數。

語法

```
ST_NRings(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER

如果 geom 為 Null，則會傳回 Null。

傳回的值如下。

傳回的值	幾何子類型
0	在 geom 是 POINT、LINESTRING、MULTIPOINT 或 MULTILINESTRING 子類型時傳回
環的數量。	在 geom 是 POLYGON 或 MULTIPOLYGON 子類型時傳回
所有元件中環的數量	在 geom 是 GEOMETRYCOLLECTION 子類型時傳回

範例

以下 SQL 會傳回 multipolygon 中環的數量。

```
SELECT ST_NRings(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((0 0,-10 0,0 -10,0 0)))'));
```

```
st_nrings
```

```
-----
```

```
2
```

ST_NumGeometries

ST_ 會NumGeometries 傳回輸入幾何圖形中的幾何圖形數目。

語法

```
ST_NumGeometries(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER 代表 *geom* 中的幾何數量。

如果 *geom* 為 Null，則會傳回 Null。

如果 *geom* 是單一空幾何，則會傳回 0。

如果 *geom* 是單一非空幾何，則會傳回 1。

如果 *geom* 是 GEOMETRYCOLLECTION 或 MULTI 子類型，則會傳回幾何的數量。

範例

以下 SQL 會傳回輸入 multilinestring 中的幾何數。

```
SELECT ST_NumGeometries(ST_GeomFromText('MULTILINESTRING((0 0,1 0,0 5),(3 4,13 26))'));
```

```
st_numgeometries
```

```
-----
```

```
2
```

ST_NumInteriorRings

ST_會NumInteriorRings 傳回輸入多邊形幾何圖形中的環數。

語法

```
ST_NumInteriorRings(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER

如果 geom 為 Null，則會傳回 Null。

如果 geom 不是多邊形，則會傳回 Null。

範例

以下 SQL 會傳回輸入多邊形中的內部環數。

```
SELECT ST_NumInteriorRings(ST_GeomFromText('POLYGON((0 0,100 0,100 100,0 100,0 0),(1  
1,1 5,5 1,1 1),(7 7,7 8,8 7,7 7))'));
```

```
st_numinteriorrings
```

```
-----
```

```
2
```

ST_NumPoints

ST_會NumPoints 傳回輸入幾何圖形中的點數。

語法

```
ST_NumPoints(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER

如果 geom 為 Null，則會傳回 Null。

如果 geom 不是子類型 LINESTRING，則會傳回 Null。

範例

以下 SQL 會傳回輸入 linestring 中點的數量。

```
SELECT ST_NumPoints(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
29.31,77.29 29.07)'));
```

```
st_numpoints
-----
4
```

下列 SQL 會傳回 Null，因為輸入 geom 不是子類型 LINESTRING。

```
SELECT ST_NumPoints(ST_GeomFromText('MULTIPOINT(1 2,3 4)'));
```

```
st_numpoints
-----
```

ST_Perimeter

對於輸入面幾何，ST_Perimeter 會傳回 2D 投影的笛卡爾周長 (邊界的長度)。周長單位與表示輸入幾何座標的單位相同。對於點、multipoint 和線形幾何，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中幾何的周長總和。

對於輸入地理，ST_Perimeter 會傳回在球體 (由 SRID 決定) 上所計算輸入面積地理之 2D 投影的測地週長 (邊界長度)。周長的單位是公尺。對於點、multipoint 和線形地理，此函數會傳回零 (0)。當輸入為幾何集合時，此函數會傳回集合中地理的周長總和。

語法

```
ST_Perimeter(geo)
```

引數

geo

GEOMETRY 或 GEOGRAPHY 資料類型的值，或是評估為 GEOMETRY 或 GEOGRAPHY 類型的運算式。

傳回類型

DOUBLE PRECISION

如果 geo 為 null，則傳回 null。

如果找不到 SRID 值，則會傳回錯誤。

範例

下列 SQL 會傳回 multipolygon 的笛卡爾周長。

```
SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20 10,10 0)))'));
```

```
st_perimeter
```

```
-----  
68.2842712474619
```

下列 SQL 會傳回 multipolygon 的笛卡爾周長。

```
SELECT ST_Perimeter(ST_GeomFromText('MULTIPOLYGON(((0 0,10 0,0 10,0 0)),((10 0,20 0,20 10,10 0)))'));
```

```
st_perimeter
```

```
-----  
68.2842712474619
```

以下 SQL 會傳回地理中多邊形的周長。

```
SELECT ST_Perimeter(ST_GeogFromText('SRID=4326;POLYGON((0 0,1 0,0 1,0 0))'));
```

```
st_perimeter
```

```
-----  
378790.428393693
```

下列 SQL 會傳回地理中 linestring 的周長。

```
SELECT ST_Perimeter(ST_GeogFromText('SRID=4326;LINESTRING(5 0,10 0))');
```

```
st_perimeter
```

```
-----  
0
```

ST_Perimeter2D

ST_Perimeter2D 是 ST_Perimeter 的別名。如需詳細資訊，請參閱 [ST_Perimeter](#)。

ST_Point

ST_Point 會傳回輸入座標值中的點幾何。

語法

```
ST_Point(x, y)
```

引數

x

DOUBLE PRECISION 資料類型的值，表示第一個座標。

y

DOUBLE PRECISION 資料類型的值，表示第二個座標。

傳回類型

POINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值會設為 0。

如果 x 或 y 為 Null，則會傳回 Null。

範例

以下 SQL 會從輸入座標建構點幾何。

```
SELECT ST_AsText(ST_Point(5.0, 7.0));
```

```
st_astext
-----
POINT(5 7)
```

ST_PointN

ST_PointN 會傳回 linestring 中索引值指定的點。負索引值會從 linestring 結尾向後算起，以便 -1 是最後一個點。

傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_PointN(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

索引

資料類型 INTEGER 的值，代表 linestring 中某點的索引。

傳回類型

POINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值會設為 0。

如果 geom 或 index 為 Null，則會傳回 Null。

如果 index 超出範圍，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

如果 geom 不是 LINESTRING，則會傳回 Null。

範例

下列 SQL 將六點 LINESTRING 的擴充已知文字 (EWKT) 表示法傳回至 GEOMETRY 物件，並傳回 linestring 索引 5 中的點。

```
SELECT ST_AsEWKT(ST_PointN(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5,0 0)',4326), 5));
```

```
st_asewkt
-----
SRID=4326;POINT(0 5)
```

ST_Points

ST_Points 會傳回包含輸入幾何中所有非空點的多點幾何。ST_Points 不會移除輸入中重複的點，包括環形幾何的起點和終點。

語法

```
ST_Points(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

MULTIPOINT 子類型的 GEOMETRY。

傳回幾何的空間參考系統識別碼 (SRID) 值與 geom 相同。

如果 geom 為 Null，則會傳回 Null。

如果 geom 為空，則傳回空的多點。

範例

下列 SQL 範例會從輸入幾何建構多點幾何。結果為包含輸入幾何中非空點的多點幾何。

```
SELECT ST_AsEWKT(ST_Points(ST_SetSRID(ST_GeomFromText('LINESTRING(1 0,2 0,3 0)'),
4326)));
```

```
st_asewkt
-----
SRID=4326;MULTIPOINT((1 0),(2 0),(3 0))
```

```
SELECT ST_AsEWKT(ST_Points(ST_SetSRID(ST_GeomFromText('MULTIPOLYGON(((0 0,1 0,0 1,0
0)))'), 4326)));
```

```
st_asewkt
-----
SRID=4326;MULTIPOINT((0 0),(1 0),(0 1),(0 0))
```

ST_Polygon

ST_Polygon 會傳回多邊形幾何，該幾何的外部環是輸入的 linestring，且其值為空間參考系統識別碼 (SRID) 的輸入。

傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_Polygon(linestring, srid)
```

引數

linestring

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是表示 linestring 的 LINESTRING。linestring 的值必須是封閉的。

srid

INTEGER 資料類型的值，表示 SRID。

傳回類型

POLYGON 子類型的 GEOMETRY。

傳回幾何的 SRID 值會設為 srid。

如果 linestring 或 srid 為 Null，則會傳回 Null。

如果 linestring 不是 linestring，則會傳回錯誤。

如果 linestring 並非封閉，則會傳回錯誤。

如果 srid 為負值，則會傳回錯誤。

範例

以下 SQL 會使用 SRID 值建構多邊形。

```
SELECT ST_AsEWKT(ST_Polygon(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
29.31,77.29 29.07)'),4356));
```

```
st_asewkt
-----
SRID=4356;POLYGON((77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07))
```

ST_RemovePoint

ST_RemovePoint 傳回已移除索引位置之輸入幾何圖形的點的線串幾何圖形。

索引是以零開始。結果的空間參考系統識別碼 (SRID) 與輸入幾何相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_RemovePoint(geom, index)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

索引

資料類型 INTEGER 的值，代表以零開始的索引位置。

傳回類型

GEOMETRY

如果 geom 或 index 為 Null，則會傳回 Null。

如果 geom 不是子類型 LINESTRING，則會傳回錯誤。

如果 index 超出範圍，則會傳回錯誤。索引位置的有效值介於 0 與 ST_NumPoints(geom) 減 1 之間。

範例

下列 SQL 會移除 linestring 中的最後一個點。

```
WITH tmp(g) AS (SELECT ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0 5)',4326))
SELECT ST_AsEWKT(ST_RemovePoint(g, ST_NumPoints(g) - 1)) FROM tmp;
```

```
st_asewkt
```

```
-----
SRID=4326;LINESTRING(0 0,10 0,10 10,5 5)
```

ST_Reverse

ST_Reverse 會反轉線性和面積幾何的頂點順序。對於點或多點幾何，會傳回原始幾何的副本。對於幾何集合，ST_Reverse 會反轉集合中每個幾何的頂點順序。

傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_Reverse(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

所傳回幾何的空間參考系統識別碼 (SRID) 與輸入幾何的 SRID 相同。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會反轉 linestring 中點的順序。

```
SELECT ST_AsEWKT(ST_Reverse(ST_GeomFromText('LINESTRING(1 0,2 0,3 0,4 0)', 4326)));
```

```
st_asewkt
```

```
-----  
SRID=4326;LINESTRING(4 0,3 0,2 0,1 0)
```

ST_SetPoint

ST_SetPoint 返回相對於由索引指定的輸入線串的位置更新坐標的線串。新坐標是輸入點的坐標。

傳回幾何的維度與 geom1 值的維度相同。如果 geom1 和 geom2 具有不同的維度，則將 geom2 投影到 geom1 的維度。

語法

```
ST_SetPoint(geom1, index, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

索引

INTEGER 資料類型的值，代表索引的位置。0 是指 linestring 左邊的第一個點，1 是指第二點，以此類推。索引可以是負值。-1 是指 linestring 右側的第一個點，-2 是指 linestring 右側的第二個點，以此類推。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 POINT。

傳回類型

GEOMETRY

如果 geom2 是空點，則傳回 geom1。

如果 geom1、geom2 或 index 為 Null，則會傳回 Null。

如果 geom1 不是線串，則會傳回錯誤。

如果 index 不在有效的索引範圍內，則傳回錯誤。

如果 geom2 不是點，則會傳回錯誤。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

範例

下列 SQL 會傳回一個新的 linestring，在這裡我們將輸入 linestring 的第二個點設定為指定點。

```
SELECT ST_AsText(ST_SetPoint(ST_GeomFromText('LINESTRING(1 2, 3 2, 5 2, 1 2)'), 2,
  ST_GeomFromText('POINT(7 9)')));
```

```
st_astext
-----
```

```
LINESTRING(1 2,3 2,7 9,1 2)
```

下列 SQL 範例會傳回一個新的 `linestring`，在這裡我們將 `linestring` 右側的第三個點 (索引為負) 設定為指定點。

```
SELECT ST_AsText(ST_SetPoint(ST_GeomFromText('LINESTRING(1 2, 3 2, 5 2, 1 2)'), -3,  
ST_GeomFromText('POINT(7 9)')));
```

```
st_astext  
-----  
LINESTRING(1 2,7 9,5 2,1 2)
```

ST_SetSRID

`ST_SetSRID` 會傳回與輸入幾何相同的幾何，但其會使用空間參考系統識別碼 (SRID) 的值輸入進行更新。

語法

```
ST_SetSRID(geom, srid)
```

引數

`geom`

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

`srid`

INTEGER 資料類型的值，表示 SRID。

傳回類型

GEOMETRY

傳回幾何的 SRID 值會設為 `srid`。

如果 `geom` 或 `srid` 為 Null，則會傳回 Null。

如果 `srid` 為負值，則會傳回錯誤。

範例

以下 SQL 會設定 linestring 的 SRID 值。

```
SELECT ST_AsEWKT(ST_SetSRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27
29.31,77.29 29.07)'),50));
```

```
st_asewkt
```

```
-----
```

```
SRID=50;LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)
```

ST_Simplify

ST_Simplify 會使用具有指定公差的 Ramer-Douglas-Peucker 演算法，傳回輸入幾何的簡化副本。可能不會保留輸入幾何的拓樸。如需演算法的相關資訊，請參閱 Wikipedia 中的 [Ramer-Douglas-Peucker 演算法](#)。

當 ST_Simplify 計算距離以簡化幾何時，ST_Simplify 會對輸入幾何的 2D 投影進行操作。

語法

```
ST_Simplify(geom, tolerance)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

tolerance

DOUBLE PRECISION 資料類型的值，代表 Ramer-Douglas-Peucker 演算法的公差等級。如果 tolerance 為負數，則使用零。

傳回類型

GEOMETRY.

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

傳回幾何的維度與輸入幾何的維度相同。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 透過 Ramer-Douglas-Peucker 演算法使用歐幾里德距離容差 1 來簡化輸入 linestring。距離的單位與幾何座標的單位相同。

```
SELECT ST_AsEWKT(ST_Simplify(ST_GeomFromText('LINESTRING(0 0,1 2,1 1,2 2,2 1)'), 1));
```

```
st_asewkt
-----
LINESTRING(0 0,1 2,2 1)
```

ST_SRID

ST_SRID 會傳回輸入幾何的空間參考系統識別碼 (SRID)。如需 SRID 的相關資訊，請參閱 [在 Amazon Redshift 中查詢空間資料](#)。

語法

```
ST_SRID(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

INTEGER，代表 geom 的 SRID 值。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回設定為 SRID 4326 之 linestring 的 SRID 值。

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)',4326));
```

```
st_srid  
-----  
4326
```

下列 SQL 會傳回建構時未設定之 linestring 的 SRID 值。這會導致 SRID 值變成 0。

```
SELECT ST_SRID(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
```

```
st_srid  
-----  
0
```

ST_StartPoint

ST_StartPoint 返回輸入線串的第一個點。結果的空間參考系統識別碼 (SRID) 值與輸入幾何的 SRID 值相同。傳回幾何的維度與輸入幾何的維度相同。

語法

```
ST_StartPoint(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。子類型必須是 LINESTRING。

傳回類型

GEOMETRY

如果 *geom* 為 Null，則會傳回 Null。

如果 geom 為空白，則會傳回 Null。

如果 geom 不是 LINESTRING，則會傳回 Null。

範例

下列 SQL 將四點 LINESTRING 的擴充已知文字 (EWKT) 表示法傳回至 GEOMETRY 物件，並傳回 linestring 的起點。

```
SELECT ST_AsEWKT(ST_StartPoint(ST_GeomFromText('LINESTRING(0 0,10 0,10 10,5 5,0
5)',4326)));
```

```
st_asewkt
-----
SRID=4326;POINT(0 0)
```

ST_Touches

ST_Touches 會在兩個輸入幾何的 2D 投影接觸時傳回 true。如果兩個幾何圖形非空白、相交，且沒有共同的內部點，則它們會接觸。

語法

```
ST_Touches(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

下列 SQL 會檢查多邊形是否接觸 linestring。

```
SELECT ST_Touches(ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))'),
  ST_GeomFromText('LINESTRING(20 10,20 0,10 0)'));
```

```
st_touches
-----
t
```

ST_Transform

ST_Transform 會傳回一個新幾何，其座標在由輸入空間參考系統識別碼 (SRID) 定義的空間參考系統中進行轉換。

語法

```
ST_Transform(geom, srid)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

srid

INTEGER 資料類型的值，表示 SRID。

傳回類型

GEOMETRY.

傳回幾何的 SRID 值會設為 srid。

如果 geom 或 srid 為 Null，則會傳回 Null。

如果與輸入 geom 相關聯的 SRID 值不存在，則傳回錯誤。

如果 srid 不存在，則會傳回錯誤。

範例

下列 SQL 會轉換空白幾何集合的 SRID。

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('GEOMETRYCOLLECTION EMPTY', 3857),
4326));
```

```
st_asewkt
```

```
-----
SRID=4326;GEOMETRYCOLLECTION EMPTY
```

以下 SQL 會轉換 linestring 的 SRID。

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('LINESTRING(110 40, 2 3, -10 80, -7 9,
-22 -33)', 4326), 26918));
```

```
st_asewkt
```

```
-----
SRID=26918;LINESTRING(73106.6977300955 15556182.9688576,14347201.5059964
1545178.32934967,1515090.41262989 9522193.25115316,10491250.83295
2575457.28410878,5672303.72135968 -5233682.61176205)
```

下列 SQL 會轉換一個多邊形的 SRID。

```
SELECT ST_AsEWKT(ST_Transform(ST_GeomFromText('POLYGON Z ((-10 10 -7, -65 10 -6, -10 64
-5, -10 10 -7), (-11 11 5, -11 12 6, -12 11 7, -11 11 5))', 6989), 6317));
```

```
st_asewkt
```

```
-----  
SRID=6317;POLYGON Z ((6186430.2771091 -1090834.57212608  
1100247.33216237,2654831.67853801 -5693304.90741276 1100247.50581055,2760987.41750022  
-486836.575101877 5709710.44137268,6186430.2771091 -1090834.57212608  
1100247.33216237),(6146675.25029258 -1194792.63532103 1209007.1115113,6125027.87562215  
-1190584.81194058 1317403.77865723,6124888.99555252 -1301885.3455052  
1209007.49312929,6146675.25029258 -1194792.63532103 1209007.1115113))
```

ST_Union

ST_Union 會傳回代表兩個幾何聯集的幾何。也就是說，它會合併輸入幾何，以產生沒有重疊的結果幾何。

語法

```
ST_Union(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

GEOMETRY

傳回幾何的空間參考系統識別碼 (SRID) 值是輸入幾何的 SRID 值。

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 或 geom2 為空，則會傳回空的幾何。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合、linestring 或 multilinestring，則會傳回錯誤。

如果 geom1 或 geom2 不是二維 (2D) 幾何，則會傳回錯誤。

範例

下列 SQL 會傳回表示兩個輸入幾何聯集的非空幾何。

```
SELECT ST_AsEWKT(ST_Union(ST_GeomFromText('POLYGON((0 0,100 100,0 200,0 0))'),
  ST_GeomFromText('POLYGON((0 0,10 0,0 10,0 0))')));
```

```
      st_asewkt
-----
POLYGON((0 0,0 200,100 100,5 5,10 0,0 0))
```

ST_Within

ST_Within 會在第一個輸入幾何的 2D 投影位於第二個輸入幾何的 2D 投影中時傳回 true。

例如，如果 A 中的每個點都是 B 中的點，且其內部沒有非空白的交集，則幾何 A 便位於幾何 B 中。

ST_Within(A, B) 與 ST_Contains(B, A) 相等。

語法

```
ST_Within(geom1, geom2)
```

引數

geom1

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。這個值會和 geom2 比較，以判斷其是否位於 geom2 內。

geom2

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom1 或 geom2 為 Null，則會傳回 Null。

如果 geom1 和 geom2 的空間參考系統識別碼 (SRID) 值不同，則會傳回錯誤。

如果 geom1 或 geom2 是幾何集合，則會傳回錯誤。

範例

以下 SQL 會檢查第一個多邊形是否位於第二個多邊形中。

```
SELECT ST_Within(ST_GeomFromText('POLYGON((0 2,1 1,0 -1,0 2))'),  
ST_GeomFromText('POLYGON((-1 3,2 1,0 -3,-1 3))'));
```

```
st_within  
-----  
true
```

ST_X

ST_X 會傳回輸入點的第一個座標。

語法

```
ST_X(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

第一個座標的 DOUBLE PRECISION 值。

如果 point 是 Null，則會傳回 Null。

如果 point 是空點，則傳回 null。

如果 point 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回點的第一個座標。


```
SELECT ST_X(ST_Point(1,2));
```

```
st_x  
-----  
1.0
```

ST_XMax

ST_XMax 會傳回輸入幾何的第一個座標上限。

語法

```
ST_XMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第一個座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最大的第一個座標。

```
SELECT ST_XMax(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29  
29.07)'));
```

```
st_xmax  
-----
```

77.42

ST_XMin

ST_XMin 會傳回輸入幾何的第一個座標下限。

語法

```
ST_XMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第一個座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最小的第一個座標。

```
SELECT ST_XMin(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
```

```
st_xmin  
-----  
77.27
```

ST_Y

ST_Y 會傳回輸入點的第二個座標。

語法

```
ST_Y(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

第二個座標的 DOUBLE PRECISION 值。

如果 *point* 是 Null，則會傳回 Null。

如果 *point* 是空點，則傳回 null。

如果 *point* 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回點的第二個座標。

```
SELECT ST_Y(ST_Point(1,2));
```

```
st_y  
-----  
2.0
```

ST_YMax

ST_YMax 會傳回輸入幾何的第二個座標上限。

語法

```
ST_YMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第二個座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最大的第二個座標。

```
SELECT ST_YMax(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
```

```
st_ymax  
-----  
29.31
```

ST_YMin

ST_YMin 會傳回輸入幾何的第二個座標下限。

語法

```
ST_YMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

第二個座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

範例

以下 SQL 會傳回 linestring 最小的第二個座標。

```
SELECT ST_YMin(ST_GeomFromText('LINESTRING(77.29 29.07,77.42 29.26,77.27 29.31,77.29 29.07)'));
```

```
st_ymin  
-----  
29.07
```

ST_Z

ST_Z 會傳回輸入點的 z 座標。

語法

```
ST_Z(point)
```

引數

point

GEOMETRY 資料類型的 POINT 值。

傳回類型

m 座標的 DOUBLE PRECISION 值。

如果 point 是 Null，則會傳回 Null。

如果 point 是 2D 或 3DM 點，則傳回 null。

如果 point 是空點，則傳回 null。

如果 point 不是 POINT，則會傳回錯誤。

範例

以下 SQL 會傳回 3DZ 幾何中點的 z 座標。

```
SELECT ST_Z(ST_GeomFromEWKT('POINT Z (1 2 3)'));
```

```
st_z  
-----  
3
```

以下 SQL 會傳回 4D 幾何中點的 z 座標。

```
SELECT ST_Z(ST_GeomFromEWKT('POINT ZM (1 2 3 4)'));
```

```
st_z  
-----  
3
```

ST_ZMax

ST_ZMax 會傳回輸入幾何的 z 座標上限。

語法

```
ST_ZMax(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

z 座標上限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DM 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DZ 幾何中 linestring 的最大 z 座標。

```
SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING Z (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_zmax
-----
      8
```

以下 SQL 會傳回 4D 幾何中線串的最大 z 座標。

```
SELECT ST_ZMax(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_zmax
-----
     10
```

ST_ZMin

ST_ZMin 會傳回輸入幾何的 z 座標下限。

語法

```
ST_ZMin(geom)
```

引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

z 座標下限的 DOUBLE PRECISION 值。

如果 geom 為空白，則會傳回 Null。

如果 geom 為 Null，則會傳回 Null。

如果 geom 是 2D 或 3DM 幾何，則傳回 null。

範例

以下 SQL 會傳回 3DZ 幾何中 linestring 的最小 z 座標。

```
SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING Z (0 1 2, 3 4 5, 6 7 8)'));
```

```
st_zmin  
-----  
2
```

以下 SQL 會傳回 4D 幾何中 linestring 的 z 座標下限。

```
SELECT ST_ZMin(ST_GeomFromEWKT('LINESTRING ZM (0 1 2 3, 4 5 6 7, 8 9 10 11)'));
```

```
st_zmin  
-----  
2
```

SupportsBBox

如果輸入幾何支援使用預先計算的週框方塊進行編碼，SupportsBBox 會傳回 true。如需週框方塊支援的相關資訊，請參閱[邊界框](#)。

語法

```
SupportsBBox(geom)
```


引數

geom

GEOMETRY 資料類型的值，或是評估為 GEOMETRY 類型的表達式。

傳回類型

BOOLEAN

如果 geom 為 Null，則會傳回 Null。

範例

下列 SQL 會傳回 true，因為輸入點幾何支援使用週框方塊進行編碼。

```
SELECT SupportsBBox(AddBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
supportsbbox
-----
t
```

下列 SQL 會傳回 false，因為輸入點幾何不支援使用週框方塊進行編碼。

```
SELECT SupportsBBox(DropBBox(ST_GeomFromText('POLYGON((0 0,1 0,0 1,0 0))')));
```

```
supportsbbox
-----
f
```

字串函數

主題

- [|| \(串連\) 運算子](#)
- [ASCII 函數](#)
- [BPCHARCMP 函數](#)

- [BTRIM 函數](#)
- [BTTEXT_PATTERN_CMP 函數](#)
- [CHAR_LENGTH 函數](#)
- [CHARACTER_LENGTH 函數](#)
- [CHARINDEX 函數](#)
- [CHR 函數](#)
- [COLLATE 函數](#)
- [CONCAT 函數](#)
- [CRC32 函數](#)
- [DIFFERENCE 函數](#)
- [INITCAP 函數](#)
- [LEFT 和 RIGHT 函數](#)
- [LEN 函數](#)
- [LENGTH 函數](#)
- [LOWER 函數](#)
- [LPAD 和 RPAD 函數](#)
- [LTRIM 函數](#)
- [OCTETINDEX 函數](#)
- [OCTET_LENGTH 函數](#)
- [POSITION 函數](#)
- [QUOTE_IDENT 函數](#)
- [QUOTE_LITERAL 函數](#)
- [REGEXP_COUNT 函數](#)
- [REGEXP_INSTR 函數](#)
- [REGEXP_REPLACE 函數](#)
- [REGEXP_SUBSTR 函數](#)
- [REPEAT 函數](#)
- [REPLACE 函數](#)
- [REPLICATE 函數](#)
- [REVERSE 函數](#)

- [RTRIM 函數](#)
- [SOUNDEX 函數](#)
- [SPLIT_PART 函數](#)
- [STRPOS 函數](#)
- [STRTOL 函數](#)
- [SUBSTRING 函數](#)
- [TEXTLEN 函數](#)
- [TRANSLATE 函數](#)
- [TRIM 函數](#)
- [UPPER 函數](#)

字串函數處理和操作字元字串，或評估為字元字串的表達式。當這些函數中的 string 引數是常值時，必須以單引號括住。支援的資料類型包括 CHAR 和 VARCHAR。

下節提供所支援函數的函數名稱、語法及描述。字串的所有偏移都是以一開始。

已取代的僅限領導節點函數

下列字串函數已棄用，因為它們只在領導者節點上執行。如需詳細資訊，請參閱 [僅限領導節點函數](#)

- GET_BYTE
- SET_BIT
- SET_BYTE
- 結束_ASCII

|| (串連) 運算子

串連 || 符號兩側的兩個運算式，並傳回串連後的運算式。

類似於 [CONCAT 函數](#)。

Note

如果其中一個或兩個運算式都為 null，則串連的結果為 NULL。

語法

```
expression1 || expression2
```

引數

expression1

CHAR 字串、VARCHAR 字串、二進位運算式或計算結果為其中一種類型的運算式。

expression2

CHAR 字串、VARCHAR 字串、二進位運算式或計算結果為其中一種類型的運算式。

傳回類型

字串的傳回類型與輸入引數的類型相同。例如，串連兩個 VARCHAR 類型的字串會傳回一個 VARCHAR 類型的字串。

範例

下列範例使用TICKIT範例資料庫中的 USERS和 VENUE資料表。如需詳細資訊，請參閱[範本資料庫](#)。

若要從範例資料庫中的USERS資料表串連 FIRSTNAME和 LASTNAME 欄位，請使用下列範例。

```
SELECT (firstname || ' ' || lastname) as fullname
FROM users
ORDER BY 1
LIMIT 10;
```

```
+-----+
|  fullname  |
+-----+
| Aaron Banks |
| Aaron Booth |
| Aaron Browning |
| Aaron Burnett |
| Aaron Casey |
| Aaron Cash |
| Aaron Castro |
| Aaron Dickerson |
```

```
| Aaron Dixon      |
| Aaron Dotson    |
+-----+
```

若要串連可能包含 Null 的欄，請使用 [NVL](#) 和 [COALESCE](#) 函數表達式。下列範例會在遇到 NULL 0 時 NVL，使用 傳回。

```
SELECT (venueName || ' seats ' || NVL(venueSeats, 0)) as seating
FROM venue
WHERE venuestate = 'NV' or venuestate = 'NC'
ORDER BY 1
LIMIT 10;
```

```
+-----+
|          seating          |
+-----+
| Ballys Hotel seats 0     |
| Bank of America Stadium seats 73298 |
| Bellagio Hotel seats 0  |
| Caesars Palace seats 0  |
| Harrahs Hotel seats 0   |
| Hilton Hotel seats 0    |
| Luxor Hotel seats 0     |
| Mandalay Bay Hotel seats 0 |
| Mirage Hotel seats 0    |
| New York New York seats 0 |
+-----+
```

ASCII 函數

ASCII 函數會傳回您指定字串中第一個字元的 ASCII 程式碼或 Unicode 程式碼點。如果字串為空，該函數傳回 0。如果字串為空，它傳回 NULL。

語法

```
ASCII('string')
```

引數

string

CHAR 字串或 VARCHAR 字串。

傳回類型

INTEGER

範例

若要傳回 NULL，請使用下列範例。NULL 如果兩個引數相同，NULLIF則函數會傳回，因此ASCII函數的輸入引數為 NULL。如需詳細資訊，請參閱[NULLIF 函數](#)。

```
SELECT ASCII(NULLIF('', ''));
```

```
+-----+
| ascii |
+-----+
|  NULL |
+-----+
```

若要傳回ASCII程式碼 0，請使用下列範例。

```
SELECT ASCII('');
```

```
+-----+
| ascii |
+-----+
|     0 |
+-----+
```

若要傳回字 amazon 第一個字母的ASCII程式碼 97，請使用下列範例。

```
SELECT ASCII('amazon');
```

```
+-----+
| ascii |
+-----+
|    97 |
+-----+
```

若要傳回 Amazon ASCII 字的第一個字母代碼 65，請使用下列範例。

```
SELECT ASCII('Amazon');
```

```
+-----+
```

```
| ascii |  
+-----+  
|    65 |  
+-----+
```

BPCHARCMP 函數

比較兩個字串的值並傳回整數。如果字串相同，則函數會傳回 0。如果第一個字串的字母順序較大，則函數會傳回 1。如果第二個字串較大，則函數會傳回 -1。

如果是多位元組字元，則根據位元組編碼來比較。

[BTTEXT_PATTERN_CMP 函數](#) 的同義詞。

語法

```
BPCHARCMP(string1, string2)
```

引數

string1

CHAR 字串或 VARCHAR 字串。

string2

CHAR 字串或 VARCHAR 字串。

傳回類型

INTEGER

範例

下列範例使用TICKIT範例資料庫中的 USERS 資料表。如需詳細資訊，請參閱[範本資料庫](#)。

若要判斷使用者的名字是否大於USERS資料表前十個項目的使用者姓氏，請使用下列範例。對於 的字串FIRSTNAME晚於 字串的字母LASTNAME，函數會傳回 1。如果 LASTNAME 的字母順序晚於 FIRSTNAME，則函數會傳回 -1。

```
SELECT userid, firstname, lastname, BPCHARCMP(firstname, lastname)  
FROM users
```

```
ORDER BY 1, 2, 3, 4
LIMIT 10;
```

userid	firstname	lastname	bpcharcmp
1	Rafael	Taylor	-1
2	Vladimir	Humphrey	1
3	Lars	Ratliff	-1
4	Barry	Roy	-1
5	Reagan	Hodge	1
6	Victor	Hernandez	1
7	Tamekah	Juarez	1
8	Colton	Roy	-1
9	Mufutau	Watkins	-1
10	Naida	Calderon	1

若要傳回函數傳回的USERS資料表中的所有項目0，請使用下列範例。當 FIRSTNAME與 相同0時，函數會傳回 LASTNAME。

```
SELECT userid, firstname, lastname,
BPCHARCMP(firstname, lastname)
FROM users
WHERE BPCHARCMP(firstname, lastname)=0
ORDER BY 1, 2, 3, 4;
```

userid	firstname	lastname	bpcharcmp
62	Chase	Chase	0
4008	Whitney	Whitney	0
12516	Graham	Graham	0
13570	Harper	Harper	0
16712	Cooper	Cooper	0
18359	Chase	Chase	0
27530	Bradley	Bradley	0
31204	Harding	Harding	0

BTRIM 函數

BTRIM 函數會移除前尾空白，或移除符合選用指定字串的前尾字元，藉此修剪字串。

語法

```
BTRIM(string [, trim_chars ] )
```

引數

string

要修剪的輸入 VARCHAR 字串。

trim_chars

包含要比對之字元的 VARCHAR 字串。

傳回類型

BTRIM 函數會傳回 VARCHAR 字串。

範例

下列範例從字串 ' abc ' 中修剪開頭和結尾空格：

```
select '   abc   ' as untrim, btrim('   abc   ') as trim;

untrim   | trim
-----+-----
   abc   | abc
```

下列範例從字串 'xyzaxyzbxyzxyz' 中移除開頭和結尾 'xyz' 字串。開頭和結尾的 'xyz' 已移除，但出現在字串內的部分則未移除。

```
select 'xyzaxyzbxyzxyz' as untrim,
btrim('xyzaxyzbxyzxyz', 'xyz') as trim;

untrim   | trim
-----+-----
xyzaxyzbxyzxyz | axyzbxyzc
```

下列範例會從符合 trim_chars 清單 'tes' 中任何字元的字串 'setuphistorycassettes' 中移除開頭和結尾部分。任何出現在輸入字串開頭或結尾的 trim_chars 清單中另一個字元前的 t、e 或 s 都會被移除。

```
SELECT btrim('setuphistorycassettes', 'tes');
```

```
      btrim
```

```
-----  
      uphistoryca
```

BTTEXT_PATTERN_CMP 函數

BPCHARCMP 函數的同義字。

如需詳細資訊，請參閱 [BPCHARCMP 函數](#)。

CHAR_LENGTH 函數

LEN 函數的同義詞。

請參閱 [LEN 函數](#)。

CHARACTER_LENGTH 函數

LEN 函數的同義詞。

請參閱 [LEN 函數](#)。

CHARINDEX 函數

傳回指定子字串在一個字串內的位置。

如需相似函數，請參閱 [POSITION 函數](#) 和 [STRPOS 函數](#)。

語法

```
CHARINDEX( substring, string )
```

引數

substring

在 *string* 內要搜尋的子字串。

string

要搜尋的字串或欄。

傳回類型

INTEGER

CHARINDEX 函數會傳回INTEGER對應於子字串位置的（以一個為基礎，而非以零為基礎）。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。CHARINDEX 0 如果在字串中找不到子字串，會傳回。

範例

若要顯示字串 fish 在單字 dog 內的位置，請使用下列範例。

```
SELECT CHARINDEX('fish', 'dog');
```

```
+-----+
| charindex |
+-----+
|          0 |
+-----+
```

若要顯示字串 fish 在單字 dogfish 內的位置，請使用下列範例。

```
SELECT CHARINDEX('fish', 'dogfish');
```

```
+-----+
| charindex |
+-----+
|          4 |
+-----+
```

下列範例使用TICKIT範例資料庫中的 SALES 資料表。如需詳細資訊，請參閱[範本資料庫](#)。

若要從SALES資料表傳回佣金超過 999.00 的不同銷售交易數目，請使用下列範例。此命令透過檢查佣金值開頭的小數是否超過 4 位元來計算大於 999.00 的佣金。

```
SELECT DISTINCT CHARINDEX('.', commission), COUNT (CHARINDEX('.', commission))
FROM sales
WHERE CHARINDEX('.', commission) > 4
GROUP BY CHARINDEX('.', commission)
ORDER BY 1,2;
```

```
+-----+-----+
| charindex | count |
+-----+-----+
|          5 |    629 |
+-----+-----+
```

CHR 函數

CHR 函數會傳回符合輸入參數所指定之ASCII程式碼點值的字元。

語法

```
CHR(number)
```

引數

number

輸入參數是代表ASCII程式碼點值INTEGER的。

傳回類型

CHAR

如果ASCII字元符合輸入值，CHR函數會傳回CHAR字串。如果輸入號碼沒有ASCII相符項目，函數會傳回 NULL。

範例

若要傳回與ASCII程式碼點 0 對應的字元，請使用下列範例。請注意，CHR函數NULL會傳回輸入 0。

```
SELECT CHR(0);
```

```
+-----+
| chr |
+-----+
|    |
+-----+
```

若要傳回與ASCII程式碼點 65 對應的字元，請使用下列範例。

```
SELECT CHR(65);
```

```
+-----+
| chr  |
+-----+
| A    |
+-----+
```

若要傳回以大寫 A（ASCII 程式碼點 65）開頭的不同事件名稱，請使用下列範例。下列範例使用 TICKIT 範例資料庫中的 EVENT 資料表。如需詳細資訊，請參閱 [範本資料庫](#)。

```
SELECT DISTINCT eventname FROM event
WHERE SUBSTRING(eventname, 1, 1)=CHR(65) LIMIT 5;
```

```
+-----+
|          eventname          |
+-----+
| A Catered Affair           |
| As You Like It             |
| A Man For All Seasons      |
| Alan Jackson                |
| Armando Manzanero           |
+-----+
```

COLLATE 函數

COLLATE 函數會覆寫字串欄或表達式的定序。

如需如何使用資料庫定序建立資料表的資訊，請參閱 [CREATE TABLE](#)。

如需如何使用資料庫定序建立資料庫的資訊，請參閱 [CREATE DATABASE](#)。

語法

```
COLLATE( string, 'case_sensitive' | 'case_insensitive');
```

引數

string

您要覆寫的字串欄或運算式。

'case_sensitive' | 'case_insensitive'

定序名稱的字串常數。Amazon Redshift 只支援 case_sensitive 或 case_insensitive。

傳回類型

COLLATE 函數 CHAR 會根據第一個輸入表達式類型傳回 VARCHAR 或 。此函數只會變更第一個輸入引數的定序，而不會變更其輸出值。

範例

若要建立資料表 T 並將資料表 T 中的 col1 定義為 case_sensitive，請使用下列範例。

```
CREATE TABLE T ( col1 Varchar(20) COLLATE case_sensitive );  
  
INSERT INTO T VALUES ('john'),('JOHN');
```

當您執行第一個查詢時，Amazon Redshift 只會傳回 john。COLLATE 函數在 col1 上執行後，定序會變成 case_insensitive。第二個查詢會傳回 john 和 JOHN。

```
SELECT * FROM T WHERE col1 = 'john';  
  
+-----+  
| col1 |  
+-----+  
| john |  
+-----+  
  
SELECT * FROM T WHERE COLLATE(col1, 'case_insensitive') = 'john';  
  
+-----+  
| col1 |  
+-----+  
| john |  
| JOHN |  
+-----+
```

若要建立資料表 A 並將資料表 A 中的 col1 定義為 case_insensitive，請使用下列範例。

```
CREATE TABLE A ( col1 Varchar(20) COLLATE case_insensitive );
```

```
INSERT INTO A VALUES ('john'),('JOHN');
```

當您執行第一個查詢時，Amazon Redshift 會同時傳回 john 和 JOHN。COLLATE 函數在 col1 上執行後，定序會變成 case_sensitive。第二個查詢只會傳回 john。

```
SELECT * FROM A WHERE col1 = 'john';
```

```
+-----+
| col1 |
+-----+
| john |
| JOHN |
+-----+
```

```
SELECT * FROM A WHERE COLLATE(col1, 'case_sensitive') = 'john';
```

```
+-----+
| col1 |
+-----+
| john |
+-----+
```

CONCAT 函數

CONCAT 函數會串連兩個運算式，並傳回產生的運算式。若要串連兩個以上的表達式，請使用巢狀 CONCAT 函數。兩個表達式之間的串連運算子 (||) 會產生與 CONCAT 函數相同的結果。

語法

```
CONCAT ( expression1, expression2 )
```

引數

expression1、*expression2*

這兩個引數都可以是固定長度的字元字串、可變長度字串、二進位運算式或計算結果為這些輸入之一的運算式。

傳回類型

CONCAT 會傳回運算式。運算式的資料類型與輸入引數的類型相同。

如果輸入運算式的類型不同，Amazon Redshift 會嘗試以隱含方式輸入轉換其中一個運算式。如果無法轉換數值，系統會傳回一個錯誤。

使用須知

- 對於CONCAT函數和串連運算子，如果一個或兩個表達式都是 null，串連的結果是 null。

範例

下列範例串連兩個字元常值：

```
SELECT CONCAT('December 25, ', '2008');
```

```
concat
-----
December 25, 2008
(1 row)
```

下列查詢使用 || 運算子而非 CONCAT 產生相同的結果：

```
SELECT 'December 25, ' || '2008';
```

```
?column?
-----
December 25, 2008
(1 row)
```

下列範例使用另一個CONCAT函數內的巢狀CONCAT函數來串連三個字元字串：

```
SELECT CONCAT('Thursday, ', CONCAT('December 25, ', '2008'));
```

```
concat
-----
Thursday, December 25, 2008
(1 row)
```

若要串連可能包含的資料欄NULLs，請使用 [NVL](#) 和 [COALESCE](#) 函數，該資料欄會在遇到時傳回給定的值NULL。下列範例會在遇到 NULL 時 NVL，使用傳回 0。

```
SELECT CONCAT(venueName, CONCAT(' seats ', NVL(venueSeats, 0))) AS seating
FROM venue WHERE venueState = 'NV' OR venueState = 'NC'
```



```
ORDER BY 1
LIMIT 5;

seating
-----
Ballys Hotel seats 0
Bank of America Stadium seats 73298
Bellagio Hotel seats 0
Caesars Palace seats 0
Harrahs Hotel seats 0
(5 rows)
```

下列查詢會從VENUE資料表串連 CITY 和 STATE值：

```
SELECT CONCAT(venuecity, venuestate)
FROM venue
WHERE venueseats > 75000
ORDER BY venueseats;

concat
-----
DenverCO
Kansas CityMO
East RutherfordNJ
LandoverMD
(4 rows)
```

下列查詢使用巢狀CONCAT函數。查詢會與VENUE資料表的 CITY和 STATE值串連，但會以逗號和空格分隔產生的字串：

```
SELECT CONCAT(CONCAT(venuecity, ', '), venuestate)
FROM venue
WHERE venueseats > 75000
ORDER BY venueseats;

concat
-----
Denver, CO
Kansas City, MO
East Rutherford, NJ
Landover, MD
(4 rows)
```

下列範例會串連兩個二進位運算式。其中 abc 是二進位值 (以十六進位表示 616263) , 且 def 是二進位值 (使用十六進位表示 646566)。結果會自動顯示為二進位值的十六進位表示。

```
SELECT CONCAT('abc'::VARBYTE, 'def'::VARBYTE);
```

```
concat
```

```
-----
```

```
616263646566
```

CRC32 函數

CRC32 是用於偵測錯誤的函數。函數使用CRC32演算法來偵測來源和目標資料之間的變更。CRC32 函數會將變數長度字串轉換為 8 個字元的字串，該字串是 32 位元二進位序列十六進位值的文字表示。若要偵測來源和目標資料之間的變更，請在來源資料上使用 CRC32函數，並儲存輸出。然後，在目標資料上使用 CRC32函數，並將該輸出與來源資料的輸出進行比較。如果資料沒有被修改，輸出將是相同的，如果資料被修改，輸出將不同。

語法

```
CRC32(string)
```

引數

string

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

CRC32 函數會傳回 8 字元字串，該字串是 32 位元二進位序列十六進位值的文字表示。Amazon Redshift CRC32函數是以 CRC-32C 多項式為基礎。

範例

為了顯示字串 Amazon Redshift 的 8 位值。

```
SELECT CRC32('Amazon Redshift');
```

```
+-----+
```

```
|  crc32  |
```

```
+-----+
| f2726906 |
+-----+
```

DIFFERENCE 函數

DIFFERENCE 函數會比較兩個字串的 American Soundex 程式碼。該函數傳回 INTEGER，以指示 Soundex 代碼之間的比對字元的數量。

一個 Soundex 代碼是一個四個字元長的字串。Soundex 代碼代表單詞的發音方式，而不是拼寫方式。例如，Smith 和 Smyth 具有相同的 Soundex 代碼。

語法

```
DIFFERENCE(string1, string2)
```

引數

string1

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

string2

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

INTEGER

DIFFERENCE 函數傳回 0–4 INTEGER 的值，計算兩個字串的 American Soundex 程式碼中相符字元的數量。Soundex 程式碼有 4 個字元，因此當字串的 American Soundex 程式碼值的所有 4 個字元都相同時，DIFFERENCE 函數會傳回 0。如果兩個字串之一為空，則傳回 1。如果字串都不包含有效字元，該函數傳回 1。DIFFERENCE 函數只會轉換英文字母的小寫或大寫 ASCII 字元，包括 a–z 和 A–Z。DIFFERENCE 會忽略其他字元。

範例

若要比較字串 % 和 @ 的 Soundex 值，請使用下列範例。該函數傳回 1，因為字串都不包含有效的字元。

```
SELECT DIFFERENCE('%', 'e');
```

```
+-----+
| difference |
+-----+
|          1 |
+-----+
```

若要比較 Amazon 的 Soundex 值和一個空字串，請使用下列範例。該函數傳回 0，因為兩個字串中的一個是空的。

```
SELECT DIFFERENCE('Amazon', '');
```

```
+-----+
| difference |
+-----+
|          0 |
+-----+
```

若要比較字串 Amazon 和 Ama 的 Soundex 值，請使用下列範例。該函數傳回 2，因為字串的 Soundex 值的 2 個字元是相同的。

```
SELECT DIFFERENCE('Amazon', 'Ama');
```

```
+-----+
| difference |
+-----+
|          2 |
+-----+
```

若要比較字串 Amazon 和 +-*/%Amazon 的 Soundex 值，請使用下列範例。該函數傳回 4，因為字串的 Soundex 值的所有 4 個字元都是相同的。請注意，函數會忽略第二個字串中的無效字元 +-*/%。

```
SELECT DIFFERENCE('Amazon', '+-*/%Amazon');
```

```
+-----+
| difference |
+-----+
|          4 |
+-----+
```

若要比較字串 AC/DC 和 Ay See Dee See 的 Soundex 值，請使用下列範例。函數傳回 4，因為字串的 Soundex 值的所有 4 個字元都相同。

```
SELECT DIFFERENCE('AC/DC', 'Ay See Dee See');
```

```
+-----+
| difference |
+-----+
|           4 |
+-----+
```

INITCAP 函數

將指定字串中每個單字的第一個字母變成大寫。INITCAP 支援 UTF-8 個多位元組字元，每個字元最多四個位元組。

語法

```
INITCAP(string)
```

引數

string

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

VARCHAR

使用須知

INITCAP 函數會讓字串大寫中每個單字的第一個字母，而任何後續字母都會變成（或左）小寫。因此，必須了解哪些字元（除了空白字元）做為單字分隔符號。單字分隔符號字元是任何非英數字元，包括標點符號、記號及控制字元。以下所有字元都是單字分隔符號：

```
! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [ \ ] ^ _ ` { | } ~
```

Tab、新行字元、換頁、換行及歸位也是單字分隔符號。

範例

下列範例使用TICKIT範例資料庫中 CATEGORY和 USERS資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要將CATDESC欄中每個單字的縮寫大寫，請使用下列範例。

```
SELECT catid, catdesc, INITCAP(catdesc)
FROM category
ORDER BY 1, 2, 3;
```

```
+-----+-----+-----+
+-----+-----+
| catid |          catdesc          |          initcap          |
|       |                            |                            |
+-----+-----+-----+
+-----+-----+
|    1 | Major League Baseball    | Major League Baseball    |
|       |                            |                            |
|    2 | National Hockey League   | National Hockey League   |
|       |                            |                            |
|    3 | National Football League | National Football League |
|       |                            |                            |
|    4 | National Basketball Association | National Basketball Association |
|       |                            |                            |
|    5 | Major League Soccer      | Major League Soccer      |
|       |                            |                            |
|    6 | Musical theatre          | Musical Theatre          |
|       |                            |                            |
|    7 | All non-musical theatre  | All Non-Musical Theatre  |
|       |                            |                            |
|    8 | All opera and light opera | All Opera And Light Opera |
|       |                            |                            |
|    9 | All rock and pop music concerts | All Rock And Pop Music Concerts |
|       |                            |                            |
|   10 | All jazz singers and bands | All Jazz Singers And Bands |
|       |                            |                            |
|   11 | All symphony, concerto, and choir concerts | All Symphony, Concerto, And Choir Concerts |
+-----+-----+-----+
+-----+-----+-----+
```

若要顯示INITCAP函數在未開始單字時不會保留大寫字元，請使用下列範例。例如，字串 MLB 會變成 Mlb。

```
SELECT INITCAP(catname)
FROM category
ORDER BY catname;
```

```
+-----+
| initcap |
+-----+
| Classical |
| Jazz      |
| Mlb       |
| Mls       |
| Musicals  |
| Nba       |
| Nfl       |
| Nhl       |
| Opera     |
| Plays     |
| Pop       |
+-----+
```

若要將空格以外的非英數字元顯示為文字分隔符號，請使用下列範例。每個字串中的幾個字母將被大寫。

```
SELECT email, INITCAP(email)
FROM users
ORDER BY userid DESC LIMIT 5;
```

```
+-----+-----+
|          email          |          initcap          |
+-----+-----+
| urna.Ut@egetdictumplacerat.edu | Urna.Ut@Egetdictumplacerat.Edu |
| nibh.enim@egestas.ca          | Nibh.Enim@Egestas.Ca      |
| in@Donecat.ca                 | In@Donecat.Ca             |
| sodales@blanditviverraDonec.ca | Sodales@Blanditviverradonec.Ca |
| sociis.natoque.penatibus@vitae.org | Sociis.Natoque.Penatibus@Vitae.Org |
+-----+-----+
```

LEFT 和 RIGHT 函數

這些函數從字元字串最左邊或最右邊傳回指定的字元數。

數目以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。

語法

```
LEFT( string, integer )
```

```
RIGHT( string, integer )
```

引數

string

CHAR 字串、VARCHAR 字串或任何計算結果為 CHAR 或 VARCHAR 字串的運算式。

integer

正整數。

傳回類型

VARCHAR

範例

下列範例使用TICKIT範例資料庫中EVENT資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要從事件名稱傳回 1000 到 1005 IDs之間的事件名稱中最左側的 5 個字元和最右側的 5 個字元，請使用下列範例。

```
SELECT eventid, eventname,  
LEFT(eventname,5) AS left_5,  
RIGHT(eventname,5) AS right_5  
FROM event  
WHERE eventid BETWEEN 1000 AND 1005  
ORDER BY 1;
```

```
+-----+-----+-----+-----+  
| eventid | eventname | left_5 | right_5 |
```



```
+-----+-----+-----+-----+
| 1000 | Gypsy          | Gypsy | Gypsy |
| 1001 | Chicago        | Chica | icago |
| 1002 | The King and I | The K | and I |
| 1003 | Pal Joey       | Pal J | Joey   |
| 1004 | Grease         | Greas | rease  |
| 1005 | Chicago        | Chica | icago |
+-----+-----+-----+-----+
```

LEN 函數

以字元數傳回指定字串的長度。

語法

LEN 是 [LENGTH 函數](#)、[CHAR_LENGTH 函數](#)、[CHARACTER_LENGTH 函數](#) 和 的同義詞 [TEXTLEN 函數](#)。

```
LEN(expression)
```

引數

運算式

CHAR 字串、VARCHAR 字串、VARBYTE 運算式或隱含評估為 CHAR、VARCHAR 或 VARBYTE 類型的運算式。

傳回類型

INTEGER

LEN 函數會傳回整數，指出輸入字串中的字元數。

如果輸入字串是字元字串，LEN函數會傳回多位元組字串中的實際字元數，而不是位元組數。例如，需要 VARCHAR(12) 欄來儲存三個四位元組中文字元。該LEN函數會3傳回相同的字串。若要取得字串的長度 (以位元組為單位)，請使用 [OCTET_LENGTH 函數](#)。

使用須知

如果 *expression* 是 CHAR 字串，不計算結尾空格。

如果 expression 是 VARCHAR 字串，則計算結尾空格。

範例

若要傳回字串 français 中的位元組數和字元數，請使用下列範例。

```
SELECT OCTET_LENGTH('français'),
LEN('français');
```

```
+-----+-----+
| octet_length | len |
+-----+-----+
|           9 |  8 |
+-----+-----+
```

若要傳回字串中的位元組數和字元數，français 而不使用 OCTET_LENGTH 函數，請使用下列範例。如需詳細資訊，請參閱 [CAST 函數](#)。

```
SELECT LEN(CAST('français' AS VARBYTE)) as bytes, LEN('français');
```

```
+-----+-----+
| bytes | len |
+-----+-----+
|     9 |  8 |
+-----+-----+
```

若要傳回沒有結尾空格的字串 cat 中的字元數、具有三個結尾空格的 cat 中的字元數、將具有三個結尾空格的 cat 轉換為長度為 6 的 CHAR，以及將具有三個結尾空格的 cat 轉換為長度為 6 的 VARCHAR，請使用以下範例。請注意，該函數不會計算 CHAR 字串的結尾空格，但會計算 VARCHAR 字串的結尾空格。

```
SELECT LEN('cat'), LEN('cat '), LEN(CAST('cat ' AS CHAR(6))) AS len_char,
LEN(CAST('cat ' AS VARCHAR(6))) AS len_varchar;
```

```
+-----+-----+-----+-----+
| len | len | len_char | len_varchar |
+-----+-----+-----+-----+
|  3 |  6 |         3 |             6 |
+-----+-----+-----+-----+
```

下列範例使用 TICKIT 範例資料庫中 VENUE 資料表的資料。如需詳細資訊，請參閱 [範本資料庫](#)。

若要傳回VENUE資料表中 10 個最長的場地名稱，請使用下列範例。

```
SELECT venuename, LEN(venueName)
FROM venue
ORDER BY 2 DESC, 1
LIMIT 10;
```

venueName	len
Saratoga Springs Performing Arts Center	39
Lincoln Center for the Performing Arts	38
Nassau Veterans Memorial Coliseum	33
Jacksonville Municipal Stadium	30
Rangers BallPark in Arlington	29
University of Phoenix Stadium	29
Circle in the Square Theatre	28
Hubert H. Humphrey Metrodome	28
Oriole Park at Camden Yards	27
Dick's Sporting Goods Park	26

LENGTH 函數

LEN 函數的同義詞。

請參閱 [LEN 函數](#)。

LOWER 函數

將字串轉換成小寫。LOWER 支援 UTF-8 個多位元組字元，每個字元最多四個位元組。

語法

```
LOWER(string)
```

引數

string

評估為 VARCHAR 類型的 VARCHAR 字串或運算式。

傳回類型

string

LOWER 函數會傳回與輸入字串相同資料類型的字串。例如，如果輸入是 CHAR 字串，該函數將傳回 CHAR 字串。

範例

下列範例使用TICKIT範例資料庫中CATEGORY資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要將CATNAME欄中的VARCHAR字串轉換為小寫，請使用下列範例。

```
SELECT catname, LOWER(catname) FROM category ORDER BY 1,2;
```

```
+-----+-----+
| catname | lower |
+-----+-----+
| Classical | classical |
| Jazz      | jazz     |
| MLB       | mlb      |
| MLS       | mls      |
| Musicals  | musicals |
| NBA       | nba      |
| NFL       | nfl      |
| NHL       | nhl      |
| Opera     | opera    |
| Plays     | plays    |
| Pop       | pop      |
+-----+-----+
```

LPAD 和 RPAD 函數

這些函數根據指定的長度，將字元附加到字串的前面或後面。

語法

```
LPAD(string1, length, [ string2 ])
```

```
RPAD(string1, length, [ string2 ])
```

引數

string1

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

長度

整數，定義函數結果的長度。字串長度以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。如果 string1 比指定的長度更長，則會截斷 (從右邊)。如果 length 是零或負數，則函數結果為空字串。

string2

(選用) 附加到 string1 前面或後面的一或多個字元。如果未指定此引數，則使用空格。

傳回類型

VARCHAR

範例

下列範例使用 TICKIT 範例資料庫中 EVENT 資料表的資料。如需詳細資訊，請參閱 [範本資料庫](#)。

若要將一組指定的活動名稱截斷至 20 個字元，並在較短名稱的前面附加空格，請使用下列範例。

```
SELECT LPAD(eventname, 20) FROM event
WHERE eventid BETWEEN 1 AND 5 ORDER BY 1;
```

```
+-----+
|      lpad      |
+-----+
|           Salome |
|      Il Trovatore |
|      Boris Godunov |
|      Gotterdammerung |
|La Cenerentola (Cind |
+-----+
```

若要將同一組活動名稱截斷至 20 個字元，但在較短名稱的後面附加 0123456789，請使用下列範例。

```
SELECT RPAD(eventname, 20,'0123456789') FROM event
WHERE eventid BETWEEN 1 AND 5 ORDER BY 1;
```

```
+-----+
|      rpad      |
+-----+
| Boris Godunov0123456 |
| Gotterdammerung01234 |
| Il Trovatore01234567 |
| La Cenerentola (Cind |
| Salome01234567890123 |
+-----+
```

LTRIM 函數

從字串開頭修剪字元。刪除只包含在修剪字元清單中的字元的最長字串。當修剪字元沒有出現在輸入字串中時，就會完成修剪。

語法

```
LTRIM( string [, trim_chars] )
```

引數

string

要修剪的字串資料行、運算式或字串常值。

trim_chars

字串欄、運算式或字串常值，代表要從 string 開頭修剪的字元。如果未指定，則會使用空格作為修剪字元。

傳回類型

LTRIM 函數會傳回與輸入字串（CHAR 或 ）相同資料類型的字元字串VARCHAR。

範例

下列範例從 listime 欄中擷取年份。字串常值 '2008-' 中的修剪字元表示要從左側修剪的字元。如果使用修剪字元 '028-'，則可以得到相同的結果。

```
select listid, listtime, ltrim(listtime, '2008-')
from listing
order by 1, 2, 3
limit 10;
```

listid	listtime	ltrim
1	2008-01-24 06:43:29	1-24 06:43:29
2	2008-03-05 12:25:29	3-05 12:25:29
3	2008-11-01 07:35:33	11-01 07:35:33
4	2008-05-24 01:18:37	5-24 01:18:37
5	2008-05-17 02:29:11	5-17 02:29:11
6	2008-08-15 02:08:13	15 02:08:13
7	2008-11-15 09:38:15	11-15 09:38:15
8	2008-11-09 05:07:30	11-09 05:07:30
9	2008-09-09 08:03:36	9-09 08:03:36
10	2008-06-17 09:44:54	6-17 09:44:54

LTRIM 當 trim_chars 出現在字串 開頭時，會移除它們中的任何字元。下列範例會在字元 'C'、'D' 和 'G' 出現在 的開頭時修剪，VENUENAME這是資料VARCHAR欄。

```
select venueid, venuename, ltrim(venueid, 'CDG')
from venue
where venueid like '%Park'
order by 2
limit 7;
```

venueid	venueid	btrim
121	ATT Park	ATT Park
109	Citizens Bank Park	itizens Bank Park
102	Comerica Park	omerica Park
9	Dick's Sporting Goods Park	ick's Sporting Goods Park
97	Fenway Park	Fenway Park
112	Great American Ball Park	reat American Ball Park
114	Miller Park	Miller Park

下列範例使用修剪字元 2，這是從 venueid 欄擷取的。

```
select ltrim('2008-01-24 06:43:29', venueid)
from venue where venueid=2;
```

```
ltrim
-----
008-01-24 06:43:29
```

下列範例不會修剪任何字元，因為在 '0' 修剪字元之前找到 2。

```
select ltrim('2008-01-24 06:43:29', '0');
```

```
ltrim
-----
2008-01-24 06:43:29
```

下列範例會使用預設的空格修剪字元，並從字串的开頭修剪兩個空格。

```
select ltrim(' 2008-01-24 06:43:29');
```

```
ltrim
-----
2008-01-24 06:43:29
```

OCTETINDEX 函數

OCTETINDEX 函數會傳回字串中子字串的位置，以位元組數為單位。

語法

```
OCTETINDEX(substring, string)
```

引數

substring

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

string

CHAR 字串、VARCHAR 字串或隱含評估為 CHAR 或 VARCHAR 類型的運算式。

傳回類型

INTEGER

OCTETINDEX 函數會傳回對應於字串中子字串位置 INTEGER 的值，以位元組數為單位，其中字串中的第一個字元會計算為 1。如果字串不包含多位元組字元，則結果等於 CHARINDEX 函數的結果。如果 string 不包含 substring，則函數會傳回 0。如果 substring 為空，則函數會傳回 1。

範例

若要傳回字串 Amazon Redshift 中子字串 q 的位置，請使用下列範例。此範例會傳回 0，因為 substring 不在 string 中。

```
SELECT OCTETINDEX('q', 'Amazon Redshift');
```

```
+-----+
| octetindex |
+-----+
|           0 |
+-----+
```

若要傳回字串 Amazon Redshift 中空白子字串的位置，請使用下列範例。這個範例會傳回 1，因為 substring 是空的。

```
SELECT OCTETINDEX('', 'Amazon Redshift');
```

```
+-----+
| octetindex |
+-----+
|           1 |
+-----+
```

若要傳回字串 Amazon Redshift 中子字串 Redshift 的位置，請使用下列範例。這個範例會傳回 8，因為 substring 是從 string 的第八個位元組開始。

```
SELECT OCTETINDEX('Redshift', 'Amazon Redshift');
```

```
+-----+
| octetindex |
+-----+
|           8 |
+-----+
```

若要傳回字串 Amazon Redshift 中子字串 Redshift 的位置，請使用下列範例。這個範例會傳回 21，因為 string 的前六個字元是雙位元組字元。

```
SELECT OCTETINDEX('Redshift', 'Ἀμαζον Amazon Redshift');
```

```
+-----+
```

```
| octetindex |  
+-----+  
|          21 |  
+-----+
```

OCTET_LENGTH 函數

傳回指定字串的長度 (以位元組為單位)。

語法

```
OCTET_LENGTH(expression)
```

引數

運算式

CHAR 字串、VARCHAR 字串、VARBYTE 運算式或隱含評估為 CHAR、VARCHAR 或 VARBYTE 類型的運算式。

傳回類型

INTEGER

OCTET_LENGTH 函數會傳回整數，指出輸入字串中的位元組數。

如果輸入字串是字元字串，則 [LEN](#) 函數會傳回多位元組字串中的實際字元數，而不是位元組數。例如，需要 VARCHAR(12) 欄來儲存三個四位元組中文字元。OCTET_LENGTH 函數會12傳回該字串，而該LEN函數會3傳回相同字串。

使用須知

如果 *expression* 是 CHAR 字串，該函數傳回 CHAR 字串的長度。例如，CHAR(6) 輸入的輸出是 CHAR(6)。

如果 *expression* 是 VARCHAR 字串，則計算結尾空格。

範例

若要在具有三個結尾空格的字串 `français` 轉換為 CHAR 和 VARCHAR 類型時傳回位元組數，請使用下列範例。如需詳細資訊，請參閱 [CAST 函數](#)。

```
SELECT OCTET_LENGTH(CAST('français ' AS CHAR(15))) AS octet_length_char,
       OCTET_LENGTH(CAST('français ' AS VARCHAR(15))) AS octet_length_varchar;
```

```
+-----+-----+
| octet_length_char | octet_length_varchar |
+-----+-----+
|                15 |                11 |
+-----+-----+
```

若要傳回字串 `français` 中的位元組數和字元數，請使用下列範例。

```
SELECT OCTET_LENGTH('français'), LEN('français');
```

```
+-----+-----+
| octet_length | len |
+-----+-----+
|             9 |    8 |
+-----+-----+
```

若要在字串 `français` 轉換為 `VARBYTE` 時傳回位元組數，請使用下列範例。

```
SELECT OCTET_LENGTH(CAST('français' AS VARBYTE));
```

```
+-----+
| octet_length |
+-----+
|             9 |
+-----+
```

POSITION 函數

傳回指定子字串在一個字串內的位置。

如需相似函數，請參閱 [CHARINDEX 函數](#) 和 [STRPOS 函數](#)。

語法

```
POSITION(substring IN string )
```

引數

substring

在 string 內要搜尋的子字串。

string

要搜尋的字串或欄。

傳回類型

POSITION 函數會傳回INTEGER對應於子字串位置的（以一個為基礎，而非以零為基礎）。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。POSITION 0 如果在字串中找不到子字串，會傳回 0。

範例

若要顯示字串 fish 在單字 dog 內的位置，請使用下列範例。

```
SELECT POSITION('fish' IN 'dog');
```

```
+-----+
| position |
+-----+
|         0 |
+-----+
```

若要顯示字串 fish 在單字 dogfish 內的位置，請使用下列範例。

```
SELECT POSITION('fish' IN 'dogfish');
```

```
+-----+
| position |
+-----+
|         4 |
+-----+
```

下列範例使用TICKIT範例資料庫中的 SALES 資料表。如需詳細資訊，請參閱[範本資料庫](#)。

若要從SALES資料表傳回佣金超過 999.00 的不同銷售交易數目，請使用下列範例。此命令透過檢查佣金值開頭的小數是否超過 4 位元來計算大於 999.00 的佣金。

```
SELECT DISTINCT POSITION('.') IN commission, COUNT (POSITION('.') IN commission)
FROM sales
WHERE POSITION('.') IN commission > 4
GROUP BY POSITION('.') IN commission
ORDER BY 1,2;
```

```
+-----+-----+
| position | count |
+-----+-----+
|          5 | 629 |
+-----+-----+
```

QUOTE_IDENT 函數

QUOTE_IDENT 函數會將指定的字串傳回為字串，其中包含前導雙引號和後導雙引號。函數輸出可以用作SQL陳述式中的識別符。該函數適當地加倍任何嵌入的雙引號。

QUOTE_IDENT 只有在建立有效識別符所需的情況下新增雙引號，當字串包含非識別符字元或將折疊為小寫時。若要一律傳回單引號字串，請使用 [QUOTE_LITERAL](#)。

語法

```
QUOTE_IDENT(string)
```

引數

string

CHAR 或 VARCHAR 字串。

傳回類型

QUOTE_IDENT 函數會傳回與輸入字串 相同的字串類型。

範例

若要傳回含雙引號的字串 "CAT"，請使用下列範例。

```
SELECT QUOTE_IDENT('"CAT"');
```

```
+-----+
| quote_ident |
+-----+
```

```
+-----+
| ""CAT"" |
+-----+
```

下列範例使用TICKIT範例資料庫中CATEGORY資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要傳回被引號包圍的資料CATNAME欄，請使用下列範例。

```
SELECT catid, QUOTE_IDENT(catname)
FROM category
ORDER BY 1,2;
```

```
+-----+-----+
| catid | quote_ident |
+-----+-----+
| 1 | "MLB" |
| 2 | "NHL" |
| 3 | "NFL" |
| 4 | "NBA" |
| 5 | "MLS" |
| 6 | "Musicals" |
| 7 | "Plays" |
| 8 | "Opera" |
| 9 | "Pop" |
| 10 | "Jazz" |
| 11 | "Classical" |
+-----+-----+
```

QUOTE_LITERAL 函數

QUOTE_LITERAL 函數會將指定的字串傳回為單引號字串，以便在SQL陳述式中用作字串常值。如果輸入參數是數字，QUOTE_LITERAL 會將其視為字串。適當地將任何內嵌單引號和反斜線加倍。

語法

```
QUOTE_LITERAL(string)
```

引數

string

CHAR 或 VARCHAR 字串。

傳回類型

QUOTE_LITERAL 函數會傳回與輸入CHARVARCHAR字串 相同的資料類型或字串。

範例

若要傳回''CAT''具有SINGLE引號的字串，請使用下列範例。

```
SELECT QUOTE_LITERAL(''CAT'');
```

```
+-----+
| quote_literal |
+-----+
| ''CAT''      |
+-----+
```

下列範例使用TICKIT範例資料庫中CATEGORY資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要傳回由單一引號包圍的CATNAME欄，請使用下列範例。

```
SELECT catid, QUOTE_LITERAL(catname)
FROM category
ORDER BY 1,2;
```

```
+-----+-----+
| catid | quote_literal |
+-----+-----+
| 1     | 'MLB'         |
| 2     | 'NHL'         |
| 3     | 'NFL'         |
| 4     | 'NBA'         |
| 5     | 'MLS'         |
| 6     | 'Musicals'    |
| 7     | 'Plays'       |
| 8     | 'Opera'       |
| 9     | 'Pop'         |
| 10    | 'Jazz'        |
| 11    | 'Classical'   |
+-----+-----+
```

若要傳回由單一引號包圍的CATID欄，請使用下列範例。

```
SELECT QUOTE_LITERAL(catid), catname
```

```
FROM category
```

```
ORDER BY 1,2;
```

```
+-----+-----+
| quote_literal | catname |
+-----+-----+
| '1'           | MLB     |
| '10'          | Jazz    |
| '11'          | Classical |
| '2'           | NHL     |
| '3'           | NFL     |
| '4'           | NBA     |
| '5'           | MLS     |
| '6'           | Musicals |
| '7'           | Plays   |
| '8'           | Opera   |
| '9'           | Pop     |
+-----+-----+
```

REGEXP_COUNT 函數

在字串中搜尋規則運算式模式，並傳回整數指出所指定模式出現在字串中的次數。如果找不到相符項目，則函數會傳回 0。如需規則表達式的詳細資訊，請參閱 Wikipedia 中的 [POSIX 運算子](#) 和 [規則表達式](#)。

語法

```
REGEXP_COUNT( source_string, pattern [, position [, parameters ] ] )
```

引數

source_string

CHAR 或 VARCHAR 字串。

pattern

代表規則表達式模式的 UTF-8 字串常值。如需詳細資訊，請參閱 [POSIX 運算子](#)。

position

(選用) 正 INTEGER，表示在 *source_string* 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設值為 1。如果 *position* 小於 1，則從 *source_string* 的第一個字元開始搜尋。如果 *position* 大於 *source_string* 中的字元數，則結果為 0。

參數

(選用) 一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- p - 使用 Perl 相容規則運算式 (PCRE) 方言解譯模式。如需的詳細資訊PCRE，請參閱 Wikipedia 中的 [Perl 相容規則運算式](#)。

傳回類型

INTEGER

範例

若要計算三字母序列出現的次數，請使用下列範例。

```
SELECT REGEXP_COUNT('abcdefghijklmnopqrstuvwxy', '[a-z]{3}');
```

```
+-----+
| regexp_count |
+-----+
|              8 |
+-----+
```

若要使用不區分大小寫的比對來計算字串 FOX 的出現次數，請使用下列範例。

```
SELECT REGEXP_COUNT('the fox', 'FOX', 1, 'i');
```

```
+-----+
| regexp_count |
+-----+
|              1 |
+-----+
```

若要使用PCRE在方言中寫入的模式來尋找包含至少一個數字和一個小寫字母的單字，請使用下列範例。此範例使用?=運算子，其在 中具有特定的前瞻性表示法PCRE。此範例會計算此類字詞的出現次數，並使用區分大小寫的比對。

```
SELECT REGEXP_COUNT('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+', 1, 'p');
```

```
+-----+
| regexp_count |
+-----+
|           2 |
+-----+
```

若要使用PCRE在方言中寫入的模式來尋找包含至少一個數字和一個小寫字母的單字，請使用下列範例。它使用?=運算子，其在 中具有特定註釋PCRE。此範例會計算此類字詞的出現次數，但與前一個範例不同，因為它使用不區分大小寫的比對。

```
SELECT REGEXP_COUNT('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
1, 'ip');
```

```
+-----+
| regexp_count |
+-----+
|           3 |
+-----+
```

下列範例使用TICKIT範例資料庫中USERS資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要計算頂層網域名稱為 org 或 edu 的次數，請使用下列範例。

```
SELECT email, REGEXP_COUNT(email, '^[^.]*\.(org|edu)') FROM users
ORDER BY userid LIMIT 4;
```

```
+-----+-----+-----+-----+
|          email          | regexp_count |
+-----+-----+-----+-----+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu |           1 |
| Suspendisse.tristique@nonnisiAenean.edu       |           1 |
| amet.faucibus.ut@condimentumegetvolutpat.ca  |           0 |
| sed@lacusUt nec.ca                             |           0 |
+-----+-----+-----+-----+
```

REGEXP_INSTR 函數

在字串中搜尋規則表達式模式，並傳回整數指出相符子字串的開始位置或結尾位置。如果找不到相符項目，則函數會傳回 0。REGEXP_INSTR 類似於 [POSITION](#) 函數，但可讓您搜尋字串以取得規則表達模式。如需規則表達式的詳細資訊，請參閱 Wikipedia 中的 [POSIX 運算子](#) 和 [規則表達式](#)。

語法

```
REGEXP_INSTR( source_string, pattern [, position [, occurrence] [, option [, parameters ] ] ] )
```

引數

source_string

要搜尋的字串表達式，例如欄名。

pattern

代表規則表達式模式的 UTF-8 字串常值。如需詳細資訊，請參閱[POSIX 運算子](#)。

position

(選用) 正 INTEGER，表示在 source_string 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設值為 1。如果 position 小於 1，則從 source_string 的第一個字元開始搜尋。如果 position 大於 source_string 中的字元數，則結果為 0。

occurrence

(選用) 正 INTEGER，表示要使用哪一個出現的模式。REGEXP_INSTR 會略過前個 occurrence-1 相符項目。預設值為 1。如果 occurrence 小於 1 或大於 source_string 中的字元數，則忽略搜尋，且結果為 0。

option

(選用) 此值指出要傳回相符項目第一個字元的位置 (0)，還是相符項目後第一個字元的位置 (1)。非零值與 1 相同。預設值為 0。

參數

(選用) 一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- e - 使用子運算式擷取子字串。

如果模式包含子運算式，REGEXP_INSTR 會使用模式中的第一個子運算式來比對子字串。REGEXP_INSTR 僅考慮第一個子運算式；會忽略其他子運算式。如果模式沒有子運算式，REGEXP_INSTR 會忽略 'e' 參數。

- p - 使用 Perl 相容規則運算式 (PCRE) 方言解譯模式。如需的詳細資訊 PCRE，請參閱 Wikipedia 中的 [Perl 相容規則運算式](#)。

傳回類型

Integer

範例

下列範例使用TICKIT範例資料庫中USERS資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要搜尋網域名稱開頭的 @ 字元，並傳回第一個相符項目的開始位置，請使用以下範例。

```
SELECT email, REGEXP_INSTR(email, '@[^\.]*')
FROM users
ORDER BY userid LIMIT 4;
```

email	regexp_instr
Etiam.laoreet.libero@sodalesMaurisblandit.edu	21
Suspendisse.tristique@nonnisiAenean.edu	22
amet.faucibus.ut@condimentumegetvolutpat.ca	17
sed@lacusUtnecc.ca	4

若要搜尋單字 Center 的變體，並傳回第一個相符項目的開始位置，請使用以下範例。

```
SELECT venueid, REGEXP_INSTR(venueid, '[cC]ent(er|re)$')
FROM venue
WHERE REGEXP_INSTR(venueid, '[cC]ent(er|re)$') > 0
ORDER BY venueid LIMIT 4;
```

venueid	regexp_instr
The Home Depot Center	16
Izod Center	6
Wachovia Center	10
Air Canada Centre	12

若要使用不區分大小寫的比對邏輯來尋找字串 FOX 第一次出現的開始位置，請使用下列範例。

```
SELECT REGEXP_INSTR('the fox', 'FOX', 1, 1, 0, 'i');
```

```
+-----+
| regexp_instr |
+-----+
|           5 |
+-----+
```

若要使用以PCRE方言形式寫入的模式來尋找包含至少一個數字和一個小寫字母的單字，請使用下列範例。它使用?=運算子，其在 中具有特定的前瞻性表示法PCRE。此範例會尋找第二個此類字詞的開始位置。

```
SELECT REGEXP_INSTR('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
  1, 2, 0, 'p');
```

```
+-----+
| regexp_instr |
+-----+
|           21 |
+-----+
```

若要使用以PCRE方言形式寫入的模式來尋找包含至少一個數字和一個小寫字母的單字，請使用下列範例。它使用?=運算子，其在 中具有特定的前瞻性表示法PCRE。此範例會尋找第二個這類字詞的開始位置，但與前一個範例不同，因為它使用不區分大小寫的比對。

```
SELECT REGEXP_INSTR('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
  1, 2, 0, 'ip');
```

```
+-----+
| regexp_instr |
+-----+
|           15 |
+-----+
```

REGEXP_REPLACE 函數

在字串中搜尋規則表達式模式，並以指定的字串取代每一個出現的模式。REGEXP_REPLACE 類似於 [REPLACE 函數](#)，但可讓您搜尋字串以取得規則表達模式。如需規則表達式的詳細資訊，請參閱 Wikipedia 中的 [POSIX 運算子](#) 和 [規則表達式](#)。

REGEXP_REPLACE 類似於 [TRANSLATE 函數](#) 和 [REPLACE 函數](#)，但 會TRANSLATE進行多個單一字元取代，並將整個字串REPLACE替換為另一個字串，而 REGEXP_REPLACE 可讓您搜尋字串以取得規則表達模式。

語法

```
REGEXP_REPLACE( source_string, pattern [, replace_string [ , position [ , parameters ] ] ] )
```

引數

source_string

要搜尋的 CHAR 或 VARCHAR 字串運算式，例如欄名。

pattern

代表規則表達式模式的 UTF-8 字串常值。如需詳細資訊，請參閱[POSIX 運算子](#)。

replace_string

(選用) CHAR 或 VARCHAR 字串運算式 (例如欄名)，用於搜尋每一個出現的模式。預設為空字串 ("")。

position

(選用) 正整數，表示在 *source_string* 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設值為 1。如果 *position* 小於 1，則從 *source_string* 的第一個字元開始搜尋。如果 *position* 大於 *source_string* 中的字元數，則結果為 *source_string*。

參數

(選用) 一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- p - 使用 Perl 相容規則運算式 (PCRE) 方言解譯模式。如需的詳細資訊 PCRE，請參閱 Wikipedia 中的 [Perl 相容規則運算式](#)。

傳回類型

VARCHAR

如果 *pattern* 或 *replace_string* 為 NULL，函數會傳回 NULL。

範例

若要使用不區分大小寫的比對取代值 quick brown fox 內所有出現的字串 FOX，請使用下列範例。

```
SELECT REGEXP_REPLACE('the fox', 'FOX', 'quick brown fox', 1, 'i');
```

```
+-----+
| regexp_replace |
+-----+
| the quick brown fox |
+-----+
```

下列範例使用PCRE在方言中寫入的模式來尋找包含至少一個數字和一個小寫字母的單字。它使用 `?` 運算子，其在 `?` 中具有特定的前瞻性表示法PCRE。若要以值 `[hidden]` 取代這類字詞的每個出現值，請使用下列範例。

```
SELECT REGEXP_REPLACE('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+', '[hidden]', 1, 'p');
```

```
+-----+
|          regexp_replace          |
+-----+
| [hidden] plain A1234 [hidden] |
+-----+
```

下列範例使用PCRE在方言中寫入的模式來尋找包含至少一個數字和一個小寫字母的單字。它使用 `?` 運算子，其在 `?` 中具有特定的前瞻性表示法PCRE。若要以值 `[hidden]` 取代這類字詞的每個出現值，但與前面的範例不同之處在於它使用不區分大小寫的比對，請使用下列範例。

```
SELECT REGEXP_REPLACE('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+', '[hidden]', 1, 'ip');
```

```
+-----+
|          regexp_replace          |
+-----+
| [hidden] plain [hidden] [hidden] |
+-----+
```

下列範例使用TICKIT範例資料庫中USERS資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要從電子郵件地中刪除 `@` 和網域名稱，請使用下列範例。

```
SELECT email, REGEXP_REPLACE(email, '@.*\\.(org|gov|com|edu|ca)$')
FROM users
ORDER BY userid LIMIT 4;
```

```

+-----+-----+
|          email          |   regexp_replace   |
+-----+-----+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu | Etiam.laoreet.libero |
| Suspendisse.tristique@nonnisiAenean.edu       | Suspendisse.tristique |
| amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut     |
| sed@lacusUt nec.ca                             | sed                   |
+-----+-----+

```

若要以 `internal.company.com` 取代電子郵件地址的網域名稱，請使用下列範例。

```

SELECT email, REGEXP_REPLACE(email, '@.*\.[[:alpha:]]{2,3}', '@internal.company.com')
FROM users
ORDER BY userid LIMIT 4;

+-----+-----+
|          email          |   regexp_replace   |
+-----+-----+
| Etiam.laoreet.libero@sodalesMaurisblandit.edu | Etiam.laoreet.libero@internal.company.com |
| Suspendisse.tristique@nonnisiAenean.edu       | Suspendisse.tristique@internal.company.com |
| amet.faucibus.ut@condimentumegetvolutpat.ca   | amet.faucibus.ut@internal.company.com     |
| sed@lacusUt nec.ca                             | sed@internal.company.com                 |
+-----+-----+

```

REGEXP_SUBSTR 函數

搜尋規則運算式模式，傳回字串中的字元。REGEXP_SUBSTR 類似於 [SUBSTRING 函數](#) 函數，但可讓您搜尋字串以取得規則表達模式。如果函數不能比對規則運算式與字串中的任何字元，則傳回一個空字串。如需規則表達式的詳細資訊，請參閱 Wikipedia 中的 [POSIX 運算子](#) 和 [規則表達式](#)。

語法

```
REGEXP_SUBSTR( source_string, pattern [, position [, occurrence [, parameters ] ] ] )
```


引數

source_string

要搜尋的字串運算式。

pattern

代表規則表達式模式的 UTF-8 字串常值。如需詳細資訊，請參閱[POSIX 運算子](#)。

position

正整數，表示在 source_string 內開始搜尋的位置。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。預設為 1。如果 position 小於 1，則從 source_string 的第一個字元開始搜尋。如果 position 大於 source_string 中的字元數，則結果為空字串 ("")。

occurrence

正整數，表示要使用哪一個出現的模式。REGEXP_SUBSTR 會略過第一次出現的 -1 個相符項目。預設為 1。如果發生少於 1 個或大於 source_string 中的字元數，則會忽略搜尋，且結果為 NULL。

參數

一或多個字串常值，表示函數如何比對模式。可能值如下：

- c - 進行區分大小寫比對。預設是使用區分大小寫比對。
- i - 進行不區分大小寫比對。
- e - 使用子運算式擷取子字串。

如果模式包含子運算式，REGEXP_SUBSTR 會使用模式中的第一個子運算式來比對子字串。子運算式是用括號括起來的模式中的運算式。例如，對於模式 'This is a '，會比對第一個運算式和字串 'This is a (\\w+)' 後跟一個單詞。具有 e 參數的 REGEXP_SUBSTR 不會傳回模式，只會傳回子運算式內的字串。

REGEXP_SUBSTR 僅考慮第一個子運算式；會忽略其他子運算式。如果模式沒有子運算式，REGEXP_SUBSTR 會忽略 'e' 參數。

- p - 使用 Perl 相容規則運算式 (PCRE) 方言解譯模式。如需的詳細資訊 PCRE，請參閱 Wikipedia 中的 [Perl 相容規則運算式](#)。

傳回類型

VARCHAR

範例

下列範例傳回電子郵件地址在 @ 和網域域名之間的部分。查詢的 users 資料來自 Amazon Redshift 範例資料。如需詳細資訊，請參閱[範本資料庫](#)。

```
SELECT email, regexp_substr(email,'@[^.]*')
FROM users
ORDER BY userid LIMIT 4;
```

email	regexp_substr
Suspendisse.tristique@nonnisiAenean.edu	@nonnisiAenean
amet.faucibus.ut@condimentumegetvolutpat.ca	@condimentumegetvolutpat
sed@lacusUtnecc.ca	@lacusUtnecc
Cum@accumsan.com	@accumsan

下列範例會使用不區分大小寫的比對，傳回與第一次出現的字串 FOX 相對應的輸入部分。

```
SELECT regexp_substr('the fox', 'FOX', 1, 1, 'i');
```

```
regexp_substr
-----
fox
```

下列範例會使用不區分大小寫的比對，傳回與第二次出現的字串 FOX 相對應的輸入部分。結果為 NULL (空)，因為沒有第二次出現。

```
SELECT regexp_substr('the fox', 'FOX', 1, 2, 'i');
```

```
regexp_substr
-----
```

下列範例會傳回以小寫字母開頭之輸入的第一部分。這在功能上與沒有 c 參數的相同 SELECT 陳述式相同。

```
SELECT regexp_substr('THE SECRET CODE IS THE LOWERCASE PART OF 1931abc0EZ.', '[a-z]+',
1, 1, 'c');
```

```
regexp_substr
```

```
-----
abc
```

下列範例使用PCRE在方言中寫入的模式來尋找包含至少一個數字和一個小寫字母的單字。它使用?=運算子，其在 中具有特定的前瞻性表示法PCRE。此範例會傳回對應於第二個這類字詞的輸入部分。

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
1, 2, 'p');
```

```
regexp_substr
-----
a1234
```

下列範例使用PCRE在方言中寫入的模式來尋找包含至少一個數字和一個小寫字母的單字。它使用?=運算子，其在 中具有特定的前瞻性表示法PCRE。此範例會傳回與第二個此類字詞對應的輸入部分，但與前一個範例不同，因為它使用不區分大小寫的比對。

```
SELECT regexp_substr('passwd7 plain A1234 a1234', '(?=[^ ]*[a-z])(?=[^ ]*[0-9])[^ ]+',
1, 2, 'ip');
```

```
regexp_substr
-----
A1234
```

下列範例會使用子運算式，使用不區分大小寫的比對來尋找符合模式 'this is a (\\w+)' 的第二個字串。它傳回括號內的子運算式。

```
SELECT regexp_substr(
    'This is a cat, this is a dog. This is a mouse.',
    'this is a (\\w+)', 1, 2, 'ie');
```

```
regexp_substr
-----
dog
```

REPEAT 函數

將字串重複指定的次數。如果輸入參數是數值，會REPEAT將其視為字串。

[REPLICATE 函數](#) 的同義詞。

語法

```
REPEAT(string, integer)
```

引數

string

第一個輸入參數是要重複的字串。

integer

第二個參數是 INTEGER，表示字串重複的次數。

傳回類型

VARCHAR

範例

下列範例使用TICKIT範例資料庫中CATEGORY資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要重複CATEGORY資料表中資料CATID欄的值三次，請使用下列範例。

```
SELECT catid, REPEAT(catid,3)
FROM category
ORDER BY 1,2;
```

```
+-----+-----+
| catid | repeat |
+-----+-----+
| 1 | 111 |
| 2 | 222 |
| 3 | 333 |
| 4 | 444 |
| 5 | 555 |
| 6 | 666 |
| 7 | 777 |
| 8 | 888 |
| 9 | 999 |
| 10 | 101010 |
| 11 | 111111 |
+-----+-----+
```

REPLACE 函數

以其他指定的字元取代一組字元在現有字串內出現的所有地方。

REPLACE 類似於 [TRANSLATE 函數](#) 和 [REGEXP_REPLACE 函數](#)，除了 TRANSLATE 會進行多個單一字元取代，而 REGEXP_REPLACE 可讓您搜尋字串以取得規則表達模式，同時將整個字串 REPLACE 取代為另一個字串。

語法

```
REPLACE(string, old_chars, new_chars)
```

引數

string

要搜尋的 CHAR 或 VARCHAR 字串

old_chars

要取代的 CHAR 或 VARCHAR 字串

new_chars

新的 CHAR 或 VARCHAR 字串，用來取代 *old_string*。

傳回類型

VARCHAR

如果 *old_chars* 或 *new_chars* 為 NULL，則結果為 NULL。

範例

下列範例使用 TICKIT 範例資料庫中 CATEGORY 資料表的資料。如需詳細資訊，請參閱 [範本資料庫](#)。

若要在 Shows Theatre CATGROUP 欄位中將字串轉換為 `Theatre`，請使用下列範例。

```
SELECT catid, catgroup, REPLACE(catgroup, 'Shows', 'Theatre')  
FROM category  
ORDER BY 1,2,3;
```

```
+-----+-----+-----+
| catid | catgroup | replace |
+-----+-----+-----+
|    1  | Sports  | Sports  |
|    2  | Sports  | Sports  |
|    3  | Sports  | Sports  |
|    4  | Sports  | Sports  |
|    5  | Sports  | Sports  |
|    6  | Shows   | Theatre |
|    7  | Shows   | Theatre |
|    8  | Shows   | Theatre |
|    9  | Concerts| Concerts|
|   10  | Concerts| Concerts|
|   11  | Concerts| Concerts|
+-----+-----+-----+
```

REPLICATE 函數

REPEAT 函數的同義字。

請參閱 [REPEAT 函數](#)。

REVERSE 函數

REVERSE 函數在字串上運作，並以相反順序傳回字元。例如，`reverse('abcde')` 傳回 `edcba`。此函數適用於數值和日期資料類型，以及字元資料類型；不過，在大部分情況下，字元字串有實用值。

語法

```
REVERSE( expression )
```

引數

運算式

具有字元、日期、時間戳記或數值資料類型的運算式，代表字元反轉的目標。所有運算式都會隱含轉換為 VARCHAR 字串。CHAR 字串中的結尾空格會遭到忽略。

傳回類型

VARCHAR

範例

下列範例使用TICKIT範例資料庫中 USERS和 SALES資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要從USERS資料表中選取五個不同的城市名稱及其對應的反向名稱，請使用下列範例。

```
SELECT DISTINCT city AS cityname, REVERSE(cityname)
FROM users
ORDER BY city LIMIT 5;
```

```
+-----+-----+
| cityname | reverse |
+-----+-----+
| Aberdeen | needrebA |
| Abilene  | enelibA |
| Ada      | adA     |
| Agat     | tagA    |
| Agawam   | mawagA  |
+-----+-----+
```

若要選取五個銷售IDs及其對應的反向IDs投射作為字元字串，請使用下列範例。

```
SELECT salesid, REVERSE(salesid)
FROM sales
ORDER BY salesid DESC LIMIT 5;
```

```
+-----+-----+
| salesid | reverse |
+-----+-----+
| 172456 | 654271 |
| 172455 | 554271 |
| 172454 | 454271 |
| 172453 | 354271 |
| 172452 | 254271 |
+-----+-----+
```

RTRIM 函數

RTRIM 函數會從字串結尾修剪一組指定的字元。移除只包含修剪字元清單中字元的最長字串。當修剪字元沒有出現在輸入字串中時，就會完成修剪。

語法

```
RTRIM( string, trim_chars )
```

引數

string

要修剪的字串資料行、運算式或字串常值。

trim_chars

字串欄、運算式或字串常值，代表要從 string 結尾修剪的字元。如果未指定，則會使用空格作為修剪字元。

傳回類型

與 string 引數的資料類型相同的字串。

範例

下列範例從字串 ' abc ' 中修剪開頭和結尾空格：

```
select '   abc   ' as untrim, rtrim('   abc   ') as trim;
```

```
untrim    | trim
-----+-----
   abc    |   abc
```

下列範例從字串 'xyzaxyzbxyzcxyz' 中移除結尾 'xyz' 字串。結尾的 'xyz' 已移除，但出現在字串內的部分則未移除。

```
select 'xyzaxyzbxyzcxyz' as untrim,
rtrim('xyzaxyzbxyzcxyz', 'xyz') as trim;
```

```
untrim          | trim
-----+-----
xyzaxyzbxyzcxyz | xyzaxyzbxyzc
```

下列範例會從符合 trim_chars 清單 'tes' 中任何字元的字串 'setuphistorycassettes' 中移除結尾部分。任何出現在輸入字串結尾的 trim_chars 清單中另一個字元前的 t、e 或 s 都會被移除。


```
SELECT rtrim('setuphistorycassettes', 'tes');
```

```
      rtrim  
-----  
setuphistoryca
```

下列範例會從出現VENUENAME的 結尾修剪字元 'Park'：

```
select venueid, venuename, rtrim(venueName, 'Park')  
from venue  
order by 1, 2, 3  
limit 10;
```

venueid	venueName	rtrim
1	Toyota Park	Toyota
2	Columbus Crew Stadium	Columbus Crew Stadium
3	RFK Stadium	RFK Stadium
4	CommunityAmerica Ballpark	CommunityAmerica Ballp
5	Gillette Stadium	Gillette Stadium
6	New York Giants Stadium	New York Giants Stadium
7	BMO Field	BMO Field
8	The Home Depot Center	The Home Depot Cente
9	Dick's Sporting Goods Park	Dick's Sporting Goods
10	Pizza Hut Park	Pizza Hut

請注意，當字元出現在 結尾k時Pa， 會RTRIM移除任何字元 r、 、 或 VENUENAME。

SOUNDEX 函數

SOUNDEX 函數會傳回 American Soundex 值，該值包含輸入字串的第一個字母，後面接著 3 位數的聲音編碼，這些聲音代表您指定的字串的英文發音。例如，Smith 和 Smyth 具有相同的 Soundex 值。

語法

```
SOUNDEX(string)
```

引數

string

您可以指定要轉換為 American Soundex 代碼值的 CHAR 或 VARCHAR 字串。

傳回類型

VARCHAR (4)

使用須知

SOUNDEX 函數只會轉換英文字母的小寫和大寫ASCII字元，包括 a-z 和 A-Z。SOUNDEX會忽略其他字元。SOUNDEX 傳回以空格分隔的多個單字字串的單一 Soundex 值。

```
SELECT SOUNDEX('AWS Amazon');
```

```
+-----+
| soundex |
+-----+
| A252   |
+-----+
```

SOUNDEX 如果輸入字串不包含任何英文字母，會傳回空字串。

```
SELECT SOUNDEX('+-*/%');
```

```
+-----+
| soundex |
+-----+
|         |
+-----+
```

範例

若要傳回 Amazon 的 Soundex 值，請使用下列範例。

```
SELECT SOUNDEX('Amazon');
```

```
+-----+
| soundex |
+-----+
```

```
| A525 |  
+-----+
```

若要傳回 smith 和 smyth 的 Soundex 值，請使用下列範例。請注意，聲音值是相同的。

```
SELECT SOUNDEX('smith'), SOUNDEX('smyth');
```

```
+-----+-----+  
| smith | smyth |  
+-----+-----+  
| S530  | S530  |  
+-----+-----+
```

SPLIT_PART 函數

在指定分隔符號之處分割字串，並傳回指定位置的部分。

語法

```
SPLIT_PART(string, delimiter, position)
```

引數

string

要分割的字串資料欄、運算式或字串常值。字串可以是 CHAR 或 VARCHAR。

delimiter

分隔符號字串，表示輸入 string 的部分。

如果 delimiter 是常值，請以單引號括住。

position

要傳回之字串部分的位置 (從 1 起算)。必須是大於 0 的整數。如果位置大於字串部分的數目，SPLIT_PART 會傳回空字串。如果在 string 中找不到 delimiter，則傳回的值包含指定部分的內容，這可能是整個 string 或空值。

傳回類型

A CHAR 或 VARCHAR 字串，與字串參數相同。

範例

下列範例會使用 \$ 分隔符號將字串常值分割成多個部分，並傳回第二部分。

```
select split_part('abc$def$ghi','$',2)
```

```
split_part
-----
def
```

以下範例將使用 \$ 分隔符號將字串常值分割成多個部分。它傳回一個空字串，因為沒有找到部分 4。

```
select split_part('abc$def$ghi','$',4)
```

```
split_part
-----
```

以下範例將使用 # 分隔符號將字串常值分割成多個部分。它傳回整個字串，這是第一部分，因為沒有找到分隔符號。

```
select split_part('abc$def$ghi','#',1)
```

```
split_part
-----
abc$def$ghi
```

下列範例會將時間戳記欄位分割LISTTIME為年、月和日元件。

```
select listtime, split_part(listtime,'-',1) as year,
       split_part(listtime,'-',2) as month,
       split_part(split_part(listtime,'-',3),' ',1) as day
from listing limit 5;
```

listtime	year	month	day
2008-03-05 12:25:29	2008	03	05
2008-09-09 08:03:36	2008	09	09
2008-09-26 05:43:12	2008	09	26
2008-10-04 02:00:30	2008	10	04
2008-01-06 08:33:11	2008	01	06

下列範例會選取LISTTIME時間戳記欄位，並在 '-' 字元上將其分割以取得月份（LISTTIME字串的第二部分），然後計算每月的項目數：

```
select split_part(listtime, '-', 2) as month, count(*)
from listing
group by split_part(listtime, '-', 2)
order by 1, 2;
```

month	count
01	18543
02	16620
03	17594
04	16822
05	17618
06	17158
07	17626
08	17881
09	17378
10	17756
11	12912
12	4589

STRPOS 函數

傳回子字串在指定字串內的位置。

如需相似函數，請參閱 [CHARINDEX 函數](#) 和 [POSITION 函數](#)。

語法

```
STRPOS(string, substring )
```

引數

string

第一個輸入參數是要搜尋的 CHAR 或 VARCHAR 字串。

substring

第二個參數是在 string 內要搜尋的子字串。

傳回類型

INTEGER

STRPOS 函數會傳回INTEGER對應於子字串位置的（以一個為基礎，而非以零為基礎）。位置以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。

使用須知

STRPOS 0 如果在字串 中找不到子字串， 會傳回 。

```
SELECT STRPOS('dogfish', 'fist');
```

```
+-----+
| strpos |
+-----+
|      0 |
+-----+
```

範例

若要顯示 fish 內 dogfish 的位置，請使用下列範例。

```
SELECT STRPOS('dogfish', 'fish');
```

```
+-----+
| strpos |
+-----+
|      4 |
+-----+
```

下列範例使用TICKIT範例資料庫中SALES資料表的資料。如需詳細資訊，請參閱[範本資料庫](#)。

若要從SALES資料表傳回COMMISSION超過 999.00 的銷售交易數量，請使用下列範例。

```
SELECT DISTINCT STRPOS(commission, '.'),
COUNT (STRPOS(commission, '.'))
FROM sales
WHERE STRPOS(commission, '.') > 4
GROUP BY STRPOS(commission, '.')
ORDER BY 1, 2;
```

```
+-----+-----+
| strpos | count |
+-----+-----+
|      5 |   629 |
+-----+-----+
```

STRTOL 函數

將指定底數之數字的字串表達式轉換為等同的整數值。轉換的值必須在帶正負號的 64 位元範圍內。

語法

```
STRTOL(num_string, base)
```

引數

num_string

要轉換之數字的字串表達式。如果 num_string 是空的 ('')，或開頭為 Null 字元 ('\0')，則轉換的值為 0。如果 num_string 是包含 NULL 值的資料欄，則 STRTOL 會傳回 NULL。字串的开頭可以有任意數量的空格，後面可選擇接著單一加號 '+' 或減號 '-' 來表示正或負。預設為 '+'。如果 base 是 16，字串可選擇以 '0x' 開頭。

base

INTEGER 介於 2 和 36 之間。

傳回類型

BIGINT

如果 num_string 為空，則函數傳回 NULL。

範例

若要將字串和基值對轉換為整數，請使用下列範例。

```
SELECT STRTOL('0xf',16);

+-----+
| strtol |
+-----+
```

```
| 15 |  
+-----+
```

```
SELECT STRTOL('abcd1234',16);
```

```
+-----+  
| strtol |  
+-----+  
| 2882343476 |  
+-----+
```

```
SELECT STRTOL('1234567', 10);
```

```
+-----+  
| strtol |  
+-----+  
| 1234567 |  
+-----+
```

```
SELECT STRTOL('1234567', 8);
```

```
+-----+  
| strtol |  
+-----+  
| 342391 |  
+-----+
```

```
SELECT STRTOL('110101', 2);
```

```
+-----+  
| strtol |  
+-----+  
| 53 |  
+-----+
```

```
SELECT STRTOL('\0', 2);
```

```
+-----+  
| strtol |  
+-----+  
| 0 |  
+-----+
```


SUBSTRING 函數

根據指定的開始位置傳回字串的子集。

如果輸入是字串，則提取的開始位置和字元數是以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。如果輸入是二進位運算式，則開始位置和提取的子字串是以位元組為基礎。您不能指定負長度，但可以指定負的開始位置。

語法

```
SUBSTRING(character_string FROM start_position [ FOR number_characters ] )
```

```
SUBSTRING(character_string, start_position, number_characters )
```

```
SUBSTRING(binary_expression, start_byte, number_bytes )
```

```
SUBSTRING(binary_expression, start_byte )
```

引數

character_string

供進行搜尋的字串。非字元資料類型視為字串。

start_position

在字串內要開始擷取的位置，從 1 開始。*start_position* 以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。這可以是負數。

number_characters

要擷取的字元數 (子字串的長度)。*number_characters* 以字元數為基礎，而不是位元組，所以多位元組字元視為單一字元。這不能是負數。

binary_expression

VARBYTE 要搜尋之資料類型的 *binary_expression*。

start_byte

在二進位運算式內要開始擷取的位置，從 1 開始。這可以是負數。

number_bytes

要擷取的位元組數，即子字串的長度。此數字不可以是負數。

傳回類型

VARCHAR 或 VARBYTE取決於輸入。

使用須知

以下是如何使用 `start_position` 和 `number_characters` 從字串中的各個位置提取子字串的一些範例。

下列範例傳回從第六個字元開始的四個字元的字串。

```
select substring('caterpillar',6,4);
substring
-----
pill
(1 row)
```

如果 `start_position + number_characters` 超過字串的長度，會從 `start_position` SUBSTRING傳回子字串，直到字串結束。例如：

```
select substring('caterpillar',6,8);
substring
-----
pillar
(1 row)
```

如果 `start_position` 為負數或 0，SUBSTRING則函數會傳回從字串的第一個字元開始的子字串，長度為 `start_position + number_characters - 1`。例如：

```
select substring('caterpillar',-2,6);
substring
-----
cat
(1 row)
```

如果 `start_position + number_characters - 1` 小於或等於零，SUBSTRING會傳回空字串。例如：

```
select substring('caterpillar',-5,4);
substring
-----
```

```
(1 row)
```

範例

下列範例會從LISTING資料表中的LISTTIME字串傳回月份：

```
select listid, listtime,
substring(listtime, 6, 2) as month
from listing
order by 1, 2, 3
limit 10;
```

listid	listtime	month
1	2008-01-24 06:43:29	01
2	2008-03-05 12:25:29	03
3	2008-11-01 07:35:33	11
4	2008-05-24 01:18:37	05
5	2008-05-17 02:29:11	05
6	2008-08-15 02:08:13	08
7	2008-11-15 09:38:15	11
8	2008-11-09 05:07:30	11
9	2008-09-09 08:03:36	09
10	2008-06-17 09:44:54	06

```
(10 rows)
```

下列範例與上述相同，但使用 FROM...FOR 選項：

```
select listid, listtime,
substring(listtime from 6 for 2) as month
from listing
order by 1, 2, 3
limit 10;
```

listid	listtime	month
1	2008-01-24 06:43:29	01
2	2008-03-05 12:25:29	03
3	2008-11-01 07:35:33	11
4	2008-05-24 01:18:37	05
5	2008-05-17 02:29:11	05
6	2008-08-15 02:08:13	08
7	2008-11-15 09:38:15	11

```

      8 | 2008-11-09 05:07:30 | 11
      9 | 2008-09-09 08:03:36 | 09
     10 | 2008-06-17 09:44:54 | 06
(10 rows)

```

您無法SUBSTRING使用 來預測擷取可能包含多位元組字元的字串字首，因為您需要根據位元組數而非字元數來指定多位元組字串的長度。若要根據以位元組為單位的長度擷取字串的開始區段，您可以CAST字串為 VARCHAR (byte_length) 來截斷字串，其中 byte_length 是必要的長度。下列範例從 'Fourscore and seven' 字串中擷取前 5 個位元組。

```

select cast('Fourscore and seven' as varchar(5));

varchar
-----
Fours

```

下列範例顯示二進位值 abc 的負開始位置。因為開始位置是 -3，所以子字串是從二進位值的開頭提取的。結果自動顯示為二進位子字串的十六進位表示形式。

```

select substring('abc'::varbyte, -3);

substring
-----
616263

```

下列範例顯示 1 作為二進位值 abc 的開始位置。因為沒有指定長度，所以字串從開始位置提取到字串的結尾。結果自動顯示為二進位子字串的十六進位表示形式。

```

select substring('abc'::varbyte, 1);

substring
-----
616263

```

下列範例顯示 3 作為二進位值 abc 的開始位置。因為沒有指定長度，所以字串是從開始位置提取到字串的結尾。結果自動顯示為二進位子字串的十六進位表示形式。

```

select substring('abc'::varbyte, 3);

substring

```

```
-----
63
```

下列範例顯示 2 作為二進位值 abc 的開始位置。字串從開始位置提取到位置 10，但字串的結尾位於位置 3。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, 2, 10);

 substring
-----
6263
```

下列範例顯示 2 作為二進位值 abc 的開始位置。字串從開始位置提取 1 個位元組。結果自動顯示為二進位子字串的十六進位表示形式。

```
select substring('abc'::varbyte, 2, 1);

 substring
-----
62
```

下列範例會傳回輸入字串 Silva, Ana 中最後一個空格之後出現的名字 Ana。

```
select reverse(substring(reverse('Silva, Ana'), 1, position(' ' IN reverse('Silva,
Ana'))))

 reverse
-----
Ana
```

TEXTLEN 函數

LEN 函數的同義詞。

請參閱 [LEN 函數](#)。

TRANSLATE 函數

對於給定的表達式，以指定的替換值取代所有出現的指定字元。現有字元依位置映射至 characters_to_replace 和 characters_to_substitute 引數中的替換字元。如果 characters_to_replace 引

數中指定的字元數比 `characters_to_substitute` 引數更多，傳回值中會省略 `characters_to_replace` 引數中額外的字元。

TRANSLATE 類似於 [REPLACE 函數](#) 和 [REGEXP_REPLACE 函數](#)，但會將整個字串 REPLACE 替換為另一個字串，而 REGEXP_REPLACE 可讓您搜尋字串以取得規則表達模式，同時 TRANSLATE 進行多個單一字元取代。

如果任何引數為 Null，則傳回值為 NULL。

語法

```
TRANSLATE( expression, characters_to_replace, characters_to_substitute )
```

引數

運算式

要轉換的表達式。

`characters_to_replace`

包含要取代之字元的字串。

`characters_to_substitute`

包含替換字元的字串。

傳回類型

VARCHAR

範例

若要取代字串中的幾個字元，請使用下列範例。

```
SELECT TRANSLATE('mint tea', 'inea', 'osin');
```

```
+-----+
| translate |
+-----+
| most tin  |
+-----+
```

下列範例使用 TICKIT 範例資料庫中 USERS 資料表的資料。如需詳細資訊，請參閱 [範本資料庫](#)。

若要對某欄的所有值，以點取代 at 符號 (@)，請使用下列範例。

```
SELECT email, TRANSLATE(email, '@', '.') as obfuscated_email
FROM users LIMIT 10;
```

email	obfuscated_email
Cum@accumsan.com	Cum.accumsan.com
lorem.ipsum@Vestibulumante.com	lorem.ipsum.Vestibulumante.com
non.justo.Proin@ametconsectetuer.edu	non.justo.Proin.ametconsectetuer.edu
non.ante.bibendum@porttitorTellus.org	non.ante.bibendum.porttitorTellus.org
eros@blanditatnisi.org	eros.blanditatnisi.org
augue@Donec.ca	augue.Donec.ca
cursus@pedeacurna.edu	cursus.pedeacurna.edu
at@Duis.com	at.Duis.com
quam@facilisisvitaeorci.ca	quam.facilisisvitaeorci.ca
mi.lorem@nunc.edu	mi.lorem.nunc.edu

若要對某欄的所有值，以點底線取代空格並剔除點，請使用下列範例。

```
SELECT city, TRANSLATE(city, ' ', '_')
FROM users
WHERE city LIKE 'Sain%' OR city LIKE 'St%'
GROUP BY city
ORDER BY city;
```

city	translate
Saint Albans	Saint_AlbanS
Saint Cloud	Saint_Cloud
Saint Joseph	Saint_Joseph
Saint Louis	Saint_Louis
Saint Paul	Saint_Paul
St. George	St_George
St. Marys	St_Marys
St. Petersburg	St_Petersburg
Stafford	Stafford
Stamford	Stamford
Stanton	Stanton
Starkville	Starkville

```
| Statesboro      | Statesboro      |  
| Staunton       | Staunton        |  
| Steubenville   | Steubenville    |  
| Stevens Point  | Stevens_Point   |  
| Stillwater     | Stillwater      |  
| Stockton       | Stockton        |  
| Sturgis        | Sturgis         |  
+-----+-----+
```

TRIM 函數

以空白或指定的字元來修剪字串。

語法

```
TRIM( [ BOTH | LEADING | TRAILING ] [trim_chars FROM ] string )
```

引數

BOTH | LEADING | TRAILING

(選用) 指定從何處修剪字元。使用 BOTH 移除前導和後導字元，使用 LEADING 僅移除前導字元，並使用 TRAILING 僅移除後導字元。如果省略此參數，會同時修剪開頭和結尾字元。

trim_chars

(選用) 要從字串中修剪的字元。如果省略此參數，則會修剪空格。

string

要修剪的字串。

傳回類型

TRIM 函數會傳回 VARCHAR 或 CHAR 字串。如果您搭配 SQL 命令使用 TRIM 函數，Amazon Redshift 會隱含地將結果轉換為 VARCHAR。如果您將 SELECT 清單中的 TRIM 函數用於 SQL 函數，Amazon Redshift 不會隱含轉換結果，而且您可能需要執行明確的轉換，以避免資料類型不相符錯誤。如需明確轉換的相關資訊，請參閱 [CAST 函數](#) 及 [CONVERT 函數](#)。

範例

若要從字串 dog 中修剪開頭和結尾空格，請使用下列範例。


```
SELECT TRIM('  dog ');
```

```
+-----+
| btrim |
+-----+
| dog   |
+-----+
```

若要從字串 `dog` 中修剪開頭和結尾空格，請使用下列範例。

```
SELECT TRIM(BOTH FROM '  dog ');
```

```
+-----+
| btrim |
+-----+
| dog   |
+-----+
```

若要從字串 `"dog"` 中移除開頭雙引號，請使用下列範例。

```
SELECT TRIM(LEADING '"' FROM "dog");
```

```
+-----+
| ltrim |
+-----+
| dog"  |
+-----+
```

若要從字串 `"dog"` 中移除結尾的雙引號，請使用下列範例。

```
SELECT TRIM(TRAILING '"' FROM "dog");
```

```
+-----+
| rtrim |
+-----+
| "dog  |
+-----+
```

TRIM 當 `trim_chars` 出現在字串的開頭或結尾時，會移除它們中的任何字元。下列範例會在字元 'C'、'D' 和 'G' 出現在的開頭或結尾時修剪 `VENUENAME`，這是資料 `VARCHAR` 欄。如需詳細資訊，請參閱 [範本資料庫](#)。

```
SELECT venueid, venuename, TRIM('CDG' FROM venuename)
FROM venue
WHERE venuename LIKE '%Park'
ORDER BY 2
LIMIT 7;
```

venueid	venuename	btrim
121	AT&T Park	AT&T Park
109	Citizens Bank Park	itizens Bank Park
102	Comerica Park	omerica Park
9	Dick's Sporting Goods Park	ick's Sporting Goods Park
97	Fenway Park	Fenway Park
112	Great American Ball Park	reat American Ball Park
114	Miller Park	Miller Park

UPPER 函數

將字串轉換成大寫。UPPER 支援 UTF-8 個多位元組字元，每個字元最多四個位元組。

語法

```
UPPER(string)
```

引數

string

輸入參數是 VARCHAR 字串或任何其他資料類型，例如 CHAR，可以隱含轉換為 VARCHAR。

傳回類型

UPPER 函數會傳回與輸入字串相同資料類型的字元字串。例如，如果輸入是 VARCHAR 字串，該函數將傳回 VARCHAR 字串。

範例

下列範例使用 TICKIT 範例資料庫中 CATEGORY 資料表的資料。如需詳細資訊，請參閱 [範本資料庫](#)。

若要將 CATNAME 欄位轉換為大寫，請使用下列各項。

```
SELECT catname, UPPER(catname)
FROM category
ORDER BY 1,2;
```

```
+-----+-----+
| catname | upper |
+-----+-----+
| Classical | CLASSICAL |
| Jazz      | JAZZ      |
| MLB       | MLB       |
| MLS       | MLS       |
| Musicals  | MUSICALS  |
| NBA       | NBA       |
| NFL       | NFL       |
| NHL       | NHL       |
| Opera     | OPERA     |
| Plays     | PLAYS     |
| Pop       | POP       |
+-----+-----+
```

SUPER 類型資訊函數

接下來，您可以找到 Amazon Redshift SQL 支援的類型資訊函數描述，從 SUPER 資料類型的輸入衍生動態資訊。

主題

- [DECIMAL_PRECISION 函數](#)
- [DECIMAL_SCALE 函數](#)
- [IS_ARRAY 函數](#)
- [IS_BIGINT 函數](#)
- [IS_BOOLEAN 函數](#)
- [IS_CHAR 函數](#)
- [IS_DECIMAL 函數](#)
- [IS_FLOAT 函數](#)
- [IS_INTEGER 函數](#)
- [IS_OBJECT 函數](#)
- [IS_SCALAR 函數](#)

- [IS_SMALLINT 函數](#)
- [IS_VARCHAR 函數](#)
- [JSON_SIZE 函數](#)
- [JSON_TYPEOF 函數](#)
- [SIZE](#)

DECIMAL_PRECISION 函數

檢查要儲存的最大小數位數的精確度。這個數字包括小數點左邊和右邊的數字。精確度的範圍是從 1 到 38，預設值為 38。

語法

```
DECIMAL_PRECISION(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

INTEGER

範例

若要將 DECIMAL_PRECISION 函數套用至資料表 t，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_PRECISION(s) FROM t;

+-----+
| decimal_precision |
+-----+
|                6 |
```

```
+-----+
```

DECIMAL_SCALE 函數

檢查要儲存到小數點右邊的小數位數。刻度的範圍是從 0 到精確度點，預設值為 0。

語法

```
DECIMAL_SCALE(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

INTEGER

範例

若要將 DECIMAL_SCALE 函數套用至資料表 t，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (3.14159);

SELECT DECIMAL_SCALE(s) FROM t;
```

```
+-----+
| decimal_scale |
+-----+
|                5 |
+-----+
```

IS_ARRAY 函數

檢查變數是否為陣列。如果變數是陣列，函數會傳回 true。該函數還包括空陣列。否則，函數會針對所有其他值傳回 false，包括 null。

語法

```
IS_ARRAY(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_ARRAY 函數檢查 [1,2] 是否為陣列，請使用下列範例。

```
SELECT IS_ARRAY(JSON_PARSE('[1,2]'));
```

```
+-----+
| is_array |
+-----+
| true     |
+-----+
```

IS_BIGINT 函數

檢查值是否為 BIGINT。IS_BIGINT 函數 true 會針對 64 位元範圍內的規模 0 數量傳回。否則，函數會針對所有其他值傳回 false，包括 null 和浮點數。

IS_BIGINT 函數是 IS_ 的超集 INTEGER。

語法

```
IS_BIGINT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要BIGINT使用 IS_BIGINT 函數檢查 5 是否為 ，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_BIGINT(s) FROM t;
```

```
+---+-----+
| s | is_bigint |
+---+-----+
| 5 | true      |
+---+-----+
```

IS_BOOLEAN 函數

檢查值是否為 BOOLEAN。IS_BOOLEAN 函數 true 會傳回恆定JSON布林值。函數會針對所有其他值傳回 false，包括 null。

語法

```
IS_BOOLEAN(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要BOOLEAN使用 IS_BOOLEAN 函數檢查 TRUE 是否為 ，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (TRUE);

SELECT s, IS_BOOLEAN(s) FROM t;
```

```
+-----+-----+
|  s   | is_boolean |
+-----+-----+
| true | true      |
+-----+-----+
```

IS_CHAR 函數

檢查值是否為 CHAR。IS_CHAR 函數 true 會針對只有 ASCII 個字元的字串傳回，因為 CHAR 類型只能儲存 ASCII 格式的字元。函數會針對所有其他值傳回 false。

語法

```
IS_CHAR(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要 CHAR 使用 IS_CHAR 函數檢查 t 是否為，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('t');

SELECT s, IS_CHAR(s) FROM t;
```



```
+-----+-----+
| s | is_char |
+-----+-----+
| "t" | true   |
+-----+-----+
```

IS_DECIMAL 函數

檢查值是否為 DECIMAL。IS_DECIMAL 函數 true 會針對非浮點的數字傳回。函數會針對所有其他值傳回 false，包括 null。

IS_DECIMAL 函數是 IS_ 的超集 BIGINT。

語法

```
IS_DECIMAL(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要 DECIMAL 使用 IS_DECIMAL 函數檢查 1.22 是否為 ，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (1.22);

SELECT s, IS_DECIMAL(s) FROM t;

+-----+-----+
| s   | is_decimal |
+-----+-----+
| 1.22 | true      |
+-----+-----+
```

IS_FLOAT 函數

檢查一個值是否為浮點數。IS_FLOAT 函數 true 會傳回浮點數 (FLOAT4 和 FLOAT8)。函數會針對所有其他值傳回 false。

IS_DECIMAL 集 IS_FLOAT 集不相交。

語法

```
IS_FLOAT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要 FLOAT 使用 IS_FLOAT 函數檢查 2.22::FLOAT 是否為 ，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES(2.22::FLOAT);

SELECT s, IS_FLOAT(s) FROM t;
```

```
+-----+-----+
|  s    | is_float |
+-----+-----+
| 2.22e+0 | true    |
+-----+-----+
```

IS_INTEGER 函數

對於 32 位元範圍內的小數位數為 0 的數字，則傳回 true；對於其他任何數字 (包括 null 和浮點數)，則傳回 false。

IS_INTEGER 函數是 IS_SMALLINT 函數的超集。

語法

```
IS_INTEGER(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要INTEGER使用 IS_INTEGER 函數檢查 5 是否為 ，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_INTEGER(s) FROM t;
```

```
+---+-----+
| s | is_integer |
+---+-----+
| 5 | true      |
+---+-----+
```

IS_OBJECT 函數

檢查變數是否為物件。IS_OBJECT 函數true會傳回物件，包括空物件。函數會針對所有其他值傳回false，包括null。

語法

```
IS_OBJECT(super_expression)
```

引數

`super_expression`

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 `IS_OBJECT` 函數檢查 `{"name": "Joe"}` 是否為物件，請使用下列範例。

```
CREATE TABLE t(s super);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_OBJECT(s) FROM t;
```

s	is_object
<code>{"name": "Joe"}</code>	true

IS_SCALAR 函數

檢查變數是否為純量。 `IS_SCALAR` 函數 `true` 會傳回任何非陣列或物件的值。函數會針對所有其他值傳回 `false`，包括 `null`。

一組 `IS_ARRAY`、`IS_OBJECT` 和 `IS_SCALAR` 涵蓋除了 `null` 以外的所有值。

語法

```
IS_SCALAR(super_expression)
```

引數

`super_expression`

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要使用 IS_SCALAR 函數檢查 {"name": "Joe"} 是否為純量，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (JSON_PARSE('{"name": "Joe"}'));

SELECT s, IS_SCALAR(s.name) FROM t;
```

```
+-----+-----+
|      s      | is_scalar |
+-----+-----+
| {"name":"Joe"} | true      |
+-----+-----+
```

IS_SMALLINT 函數

檢查變數是否為 SMALLINT。IS_SMALLINT 函數 true 會針對 16 位元範圍內的規模 0 數量傳回。函數會針對所有其他值傳回 false，包括 null 和浮點數。

語法

```
IS_SMALLINT(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回

BOOLEAN

範例

若要 SMALLINT 使用 IS_SMALLINT 函數檢查 5 是否為，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES (5);

SELECT s, IS_SMALLINT(s) FROM t;
```

```
+---+-----+
| s | is_smallint |
+---+-----+
| 5 | true        |
+---+-----+
```

IS_VARCHAR 函數

檢查變數是否為 VARCHAR。IS_VARCHAR 函數 true 會傳回所有字串。函數會針對所有其他值傳回 false。

IS_VARCHAR 函數是 IS_CHAR 函數的超集。

語法

```
IS_VARCHAR(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

BOOLEAN

範例

若要 VARCHAR 使用 IS_VARCHAR 函數檢查 abc 是否為 `CHAR`，請使用下列範例。

```
CREATE TABLE t(s SUPER);

INSERT INTO t VALUES ('abc');

SELECT s, IS_VARCHAR(s) FROM t;
```

```

+-----+-----+
|  s   | is_varchar |
+-----+-----+
| "abc" | true      |
+-----+-----+

```

JSON_SIZE 函數

當序列化為字串時，JSON_SIZE 函數會傳回指定SUPER表達式中的位元組數。

語法

```
JSON_SIZE(super_expression)
```

引數

super_expression

SUPER 常數或運算式。

傳回類型

INTEGER

JSON_SIZE 函數會傳回 `INTEGER` 指出輸入字串中的位元組數。此值與字元數目不同。例如，黑點 UTF-8 字元 # 的大小為 3 個位元組，即使其為 1 個字元。

使用須知

JSON_SIZE (*x*) 在功能上與 OCTET_LENGTH (JSON_) 相同SERIALIZE。不過請注意，當提供的SUPER表達式在序列化時超過系統VARCHAR限制時，JSON_SERIALIZE 會傳回錯誤。JSON_SIZE 沒有此限制。

範例

若要傳回序列化為字串的 SUPER 值的長度，請使用下列範例。

```
SELECT JSON_SIZE(JSON_PARSE('[10001,10002,"#"]'));
```

```
+-----+
```

```
| json_size |
+-----+
|          19 |
+-----+
```

請注意，提供的SUPER表達式長度為 17 個字元，但 # 是 3 位元組字元，因此 JSON_SIZE 會傳回 19。

JSON_TYPEOF 函數

JSON_TYPEOF 純量函數會根據值的動態類型，傳回VARCHAR具有布林值、數字、字串、物件、陣列或 null SUPER值的。

語法

```
JSON_TYPEOF(super_expression)
```

引數

super_expression

SUPER 運算式或欄。

傳回類型

VARCHAR

範例

若要[1, 2]使用 JSON_TYPEOF 函數檢查陣列JSON的 類型，請使用下列範例。

```
SELECT JSON_TYPEOF(ARRAY(1,2));
```

```
+-----+
| json_typeof |
+-----+
| array       |
+-----+
```

若要{"name":"Joe"}使用 JSON_TYPEOF 函數檢查物件JSON的 類型，請使用下列範例。

```
SELECT JSON_TYPEOF(JSON_PARSE('{"name":"Joe"}'));
```



```
+-----+
| json_typeof |
+-----+
| object      |
+-----+
```

SIZE

以 INTEGER 形式傳回 SUPER 類型常數或運算式的二進位記憶體內大小。

語法

```
SIZE(super_expression)
```

引數

super_expression

SUPER 類型常數或運算式。

傳回類型

INTEGER

範例

若要使用 SIZE 取得數種 SUPER 類型表達式的記憶體內大小，請使用下列範例。

```
CREATE TABLE test_super_size(a SUPER);

INSERT INTO test_super_size
VALUES
  (null),
  (TRUE),
  (JSON_PARSE('[0,1,2,3]')),
  (JSON_PARSE('{ "a":0, "b":1, "c":2, "d":3 }'))
;

SELECT a, SIZE(a)
FROM test_super_size
ORDER BY 2, 1;
```

```

+-----+-----+
|          a          | size |
+-----+-----+
| true                |    4 |
| NULL                |    4 |
| [0,1,2,3]           |   23 |
| {"a":0,"b":1,"c":2,"d":3} |   52 |
+-----+-----+

```

VARBYTE 函數與運算子

支援 VARBYTE 資料類型的 Amazon Redshift 函數和運算子包括：

- [VARBYTE 運算子](#)
- [FROM_HEX](#)
- [FROM_VARBYTE](#)
- [GETBIT](#)
- [TO_HEX](#)
- [TO_VARBYTE](#)
- [CONCAT](#)
- [LEN](#)
- [LENGTH 函數](#)
- [OCTET_LENGTH](#)
- [SUBSTRING 函數](#)

VARBYTE 運算子

下表會列出 VARBYTE 運算子。此運算子使用資料類型 VARBYTE 的二進位值。如果一個或兩個輸入都為 null，則結果為 null。

支援的運算子

運算子	描述	傳回類型
<	小於	BOOLEAN
<=	小於或等於	BOOLEAN

運算子	描述	傳回類型
=	等於	BOOLEAN
>	大於	BOOLEAN
>=	大於或等於	BOOLEAN
!= 或 <>	不等於	BOOLEAN
	串連	VARBYTE
+	串連	VARBYTE
~	位元 not	VARBYTE
&	位元 and	VARBYTE
	位元 or	VARBYTE
#	位元 xor	VARBYTE

範例

在下列範例中，'a'::VARBYTE 的值為 61，'b'::VARBYTE 的值為 62。:: 將字串轉換為 VARBYTE 資料類型。如需轉換資料類型的詳細資訊，請參閱 [CAST](#)。

若要使用 < 運算子比較 'a' 是否小於 'b'，請使用下列範例。

```
SELECT 'a'::VARBYTE < 'b'::VARBYTE AS less_than;
```

```
+-----+
| less_than |
+-----+
| true      |
+-----+
```

若要使用 = 運算子來比較 'a' 是否等於 'b'，請使用下列範例。

```
SELECT 'a'::VARBYTE = 'b'::VARBYTE AS equal;
```

```
+-----+
| equal |
+-----+
| false |
+-----+
```

若要使用 `||` 運算子連接兩個二進位值，請使用下列範例。

```
SELECT 'a'::VARBYTE || 'b'::VARBYTE AS concat;
```

```
+-----+
| concat |
+-----+
| 6162 |
+-----+
```

若要使用 `+` 運算子連接兩個二進位值，請使用下列範例。

```
SELECT 'a'::VARBYTE + 'b'::VARBYTE AS concat;
```

```
+-----+
| concat |
+-----+
| 6162 |
+-----+
```

若要使用 `FROM_VARBYTE` 函數求反輸入二進位值的每一位，請使用下列範例。字串 'a' 評估為 01100001。如需詳細資訊，請參閱[FROM_VARBYTE](#)。

```
SELECT FROM_VARBYTE(~'a'::VARBYTE, 'binary');
```

```
+-----+
| from_varbyte |
+-----+
| 10011110 |
+-----+
```

若要在兩個輸入二進位值上套用 `&` 運算子，請使用下列範例。字串 'a' 評估為 01100001 且 'b' 評估為 01100010。

```
SELECT FROM_VARBYTE('a'::VARBYTE & 'b'::VARBYTE, 'binary');
```

```
+-----+
| from_varbyte |
+-----+
|    01100000 |
+-----+
```

FROM_HEX 函數

FROM_HEX 將十六進位轉換為二進位值。

語法

```
FROM_HEX(hex_string)
```

引數

hex_string

要轉換的 VARCHAR 或 TEXT 資料類型十六進位字串。格式必須是文字值。

傳回類型

VARBYTE

範例

若要將 '6162' 的十六進位表示轉換為二進位值，請使用下列範例。結果會自動顯示為二進位值的十六進位表示。

```
SELECT FROM_HEX('6162');

+-----+
| from_hex |
+-----+
|    6162 |
+-----+
```

FROM_VARBYTE 函數

FROM_VARBYTE 會將二進位值轉換成指定格式的字元字串。

語法

```
FROM_VARBYTE(binary_value, format)
```

引數

binary_value

VARBYTE 資料類型的二進位值。

format

傳回字元字串的格式。不區分大小寫的有效值為 hex、binary、utf8 (utf-8 和 utf_8) 和 base64。

傳回類型

VARCHAR

範例

若要將二進位值 'ab' 轉換為十六進位，請使用下列範例。

```
SELECT FROM_VARBYTE('ab', 'hex');
```

```
+-----+
| from_varbyte |
+-----+
|           6162 |
+-----+
```

若要傳回 '4d' 的二進位表示法，請使用下列範例。'4d' 的二進位表示法是字元字串 01001101。

```
SELECT FROM_VARBYTE(FROM_HEX('4d'), 'binary');
```

```
+-----+
| from_varbyte |
+-----+
|      01001101 |
+-----+
```

GETBIT 函數

GETBIT 傳回指定索引處的二進位值的位元值。

語法

```
GETBIT(binary_value, index)
```

引數

binary_value

VARBYTE 資料類型的二進位值。

索引

傳回之二進位值中位元的索引編號。二進位值是從 0 開始的位元組，從最右邊的位元 (最低有效位元) 到最左邊的位元 (最高有效位元) 進行索引。

傳回類型

INTEGER

範例

若要傳回二進位值 `from_hex('4d')` 索引 2 處的位元，請使用以下範例。'4d' 的二進位表示法是 01001101。

```
SELECT GETBIT(FROM_HEX('4d'), 2);
```

```
+-----+
| getbit |
+-----+
|      1 |
+-----+
```

若要在 `from_hex('4d')` 傳回的二進位值的八個索引位置傳回位元，請使用下列範例。'4d' 的二進位表示法是 01001101。

```
SELECT GETBIT(FROM_HEX('4d'), 7), GETBIT(FROM_HEX('4d'), 6),
```

```
GETBIT(FROM_HEX('4d'), 5), GETBIT(FROM_HEX('4d'), 4),
GETBIT(FROM_HEX('4d'), 3), GETBIT(FROM_HEX('4d'), 2),
GETBIT(FROM_HEX('4d'), 1), GETBIT(FROM_HEX('4d'), 0);
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| getbit | getbit | getbit | getbit | getbit | getbit | getbit | getbit |
+-----+-----+-----+-----+-----+-----+-----+-----+
|      0 |      1 |      0 |      0 |      1 |      1 |      0 |      1 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

TO_HEX 函數

TO_HEX 將數字或二進位值轉換為十六進位表示法。

語法

```
TO_HEX(value)
```

引數

值

要轉換的數字或二進位值 (VARBYTE)。

傳回類型

VARCHAR

範例

若要將數字轉換為其十六進位表示法，請使用下列範例。

```
SELECT TO_HEX(2147676847);
```

```
+-----+
| to_hex |
+-----+
| 8002f2af |
+-----+
```

若要將 'abc' 的 VARBYTE 表示轉換為十六進位數字，請使用下列範例。


```
SELECT TO_HEX('abc'::VARBYTE);
```

```
+-----+
| to_hex |
+-----+
| 616263 |
+-----+
```

若要建立資料表、將 'abc' 的 VARBYTE 表示法插入至十六進位數字，然後選取含值的欄，請使用下列範例。

```
CREATE TABLE t (vc VARCHAR);
INSERT INTO t SELECT TO_HEX('abc'::VARBYTE);
SELECT vc FROM t;
```

```
+-----+
|  vc  |
+-----+
| 616263 |
+-----+
```

若要顯示將 VARBYTE 值轉換為 VARCHAR 時格式為 UTF-8，請使用下列範例。

```
CREATE TABLE t (vc VARCHAR);
INSERT INTO t SELECT 'abc'::VARBYTE::VARCHAR;
```

```
SELECT vc FROM t;
```

```
+-----+
| vc  |
+-----+
| abc |
+-----+
```

TO_VARBYTE 函數

TO_VARBYTE 會將指定格式的字串轉換為二進位值。

語法

```
TO_VARBYTE(string, format)
```

引數

string

CHAR 或 VARCHAR 字串。

format

輸入字串的格式。不區分大小寫的有效值為 hex、binary、utf8 (utf-8 和 utf_8) 和 base64。

傳回類型

VARBYTE

範例

若要將十六進位 6162 轉換為二進位值，請使用下列範例。結果會自動顯示為二進位值的十六進位表示。

```
SELECT TO_VARBYTE('6162', 'hex');
```

```
+-----+
| to_varbyte |
+-----+
|          6162 |
+-----+
```

若要傳回 4d 的二進位表示法，請使用下列範例。'4d' 的二進位表示是 01001101。

```
SELECT TO_VARBYTE('01001101', 'binary');
```

```
+-----+
| to_varbyte |
+-----+
|           4d |
+-----+
```

若要將 UTF-8 的 'a' 字串轉換為二進位值，請使用下列範例。結果會自動顯示為二進位值的十六進位表示。

```
SELECT TO_VARBYTE('a', 'utf8');
```

```
+-----+
| to_varbyte |
+-----+
|          61 |
+-----+
```

若要將十六進位的 '4' 字串轉換為二進位值，請使用下列範例。如果十六進位字串長度是奇數，則在前面加上 0 以形成有效的十六進位數字。

```
SELECT TO_VARBYTE('4', 'hex');
```

```
+-----+
| to_varbyte |
+-----+
|          04 |
+-----+
```

範圍函數

使用範圍函數，您可以更有效地建立分析業務查詢。範圍函數對結果集的一個分割區或「視窗」執行運算，然後針對該視窗中的每一列傳回一個值。反之，非視窗函數對結果集的每一列執行計算。不同於彙總結果列的群組函數，範圍函數會保留運算式中的所有列。

傳回的值是利用該視窗中列集的值來計算。對於資料表的每一列，視窗會定義用於計算其他屬性的列集。使用視窗規格（OVER子句）定義視窗，並以三個主要概念為基礎：

- 視窗分割，形成資料列群組（PARTITION子句）
- 視窗排序，定義每個分割區內的資料列順序或序列（ORDER BY子句）
- 相對於每一列定義的視窗規格，以進一步限制資料列集（ROWS規格）

除最終 ORDER BY 子句外，視窗函數是查詢中執行的最後一組操作。所有聯結和所有 WHERE、GROUP BY 和 HAVING 子句都會在處理視窗函數之前完成。因此，視窗函數只能出現在選取清單或 ORDER BY 子句中。您可以在具有不同窗框子句的單一查詢中使用多個範圍函數。您也可以在其他純量表達式中使用視窗函數，例如 CASE。

無法巢狀視窗函數。例如，彙總函數 [SUM](#) 可以出現在視窗函數內 [SUM](#)，但視窗函數 SUM 無法出現在另一個視窗函數內 SUM。由於視窗函數巢狀在另一個視窗函數中，因此不支援下列項目。

```
SELECT SUM(SUM(selectcol) OVER (PARTITION BY ordercol)) OVER (Partition by ordercol)
FROM t;
```

範圍函數語法摘要

範圍函數遵循標準語法，如下所示。

```
function (expression) OVER (
[ PARTITION BY expr_list ]
[ ORDER BY order_list [ frame_clause ] ] )
```

其中，`function` 是本節所述其中一個函數。

`expr_list` 如下。

```
expression | column_name [, expr_list ]
```

`order_list` 如下。

```
expression | column_name [ ASC | DESC ]
[ NULLS FIRST | NULLS LAST ]
[, order_list ]
```

`frame_clause` 如下。

```
ROWS
{ UNBOUNDED PRECEDING | unsigned_value PRECEDING | CURRENT ROW } |
{ BETWEEN
{ UNBOUNDED PRECEDING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW}
AND
{ UNBOUNDED FOLLOWING | unsigned_value { PRECEDING | FOLLOWING } | CURRENT ROW }}
```

引數

函數

範圍函數。如需詳細資訊，請參閱個別函數描述。

OVER

此子句定義視窗規格。子OVER句對於視窗函數是強制性的，並區分視窗函數與其他SQL函數。

PARTITION BY expr_list

(選用) PARTITION BY 子句會將結果集細分為分割區，就像 GROUP BY 子句一樣。如果有分割區子句，則會對每一個分割區的列來計算函數。如果未指定分割區子句，則單一分割區包含整個資料表，且會針對這整個資料表來計算函數。

排名函數 DENSE_RANK、NTILE、RANK 和 ROW_NUMBER 需要結果集中所有資料列的全域比較。使用 PARTITION BY 子句時，查詢最佳化工具可根據分割區將工作負載分散到多個層，以平行執行每個彙總。如果 PARTITION BY 子句不存在，彙總步驟必須在單一切片上連續執行，這可能會對效能產生重大負面影響，特別是對於大型叢集。

Amazon Redshift 不支援 PARTITION BY 子句中的字串常值。

ORDER BY 訂單_清單

(選用) 視窗函數會套用至根據 ORDER BY 中的順序規格排序的每個分割區中的資料列。此 ORDER BY 子句與 frame_clause 中的 ORDER BY 子句不同且完全無關。ORDER BY 子句可以在沒有 PARTITION BY 子句的情況下使用。

對於排名函數，ORDER BY 子句會識別排名值的量值。對於彙總函數，在為每一個窗框計算彙總函數之前，分割的列必須排序。如需範圍函數的詳細資訊，請參閱[範圍函數](#)。

順序清單中需要欄識別碼或可評估為欄識別碼的欄表達式。常數或常數表達式都不能用來替代欄名。

NULLS 值會視為自己的群組，並根據 NULLSFIRST 或 NULLSLAST 選項進行排序和排序。根據預設，NULL 值會 ASC 依順序排序和排序，並依 DESC 順序排序和排序。

Amazon Redshift 不支援 ORDER BY 子句中的字串常值。

如果省略 ORDER BY 子句，資料列的順序是非確定性的。

Note

在任何平行系統中，例如 Amazon Redshift，當 ORDER BY 子句未產生資料的唯一和總排序時，資料列的順序是非確定性的。也就是說，如果 ORDER BY 表達式產生重複的值（部分排序），則這些資料列的傳回順序可能會因執行一次 Amazon Redshift 而有所不同。於是，範圍函數可能傳回非預期或不一致的結果。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

column_name

分割或排序所依據的欄名。

ASC | DESC

此選項會定義表達式的排序順序，如下所示：

- ASC：遞增（例如，數值為從低到高，字元字串為 'A' 到 'Z'）。若未指定選項，資料會預設為遞增排序。
- DESC：遞減（數值為從高到低；字串為 'Z' 到 'A'）。

NULLS FIRST | NULLS LAST

指定NULLS是否應該先排序、在非空值之前排序，還是在非空值之後最後排序的選項。根據預設，NULLS 會ASC按順序排序和排序最後，並按DESC順序排序和排序。

frame_clause

對於彙總函數，當使用 ORDER BY 時，影格子句會進一步精簡函數視窗中的資料列集。它可讓您在排序的結果內包含或排除資料列組。框架子句由ROWS關鍵字和相關聯的規格符組成。

窗框子句不適用於排名函數。此外，當彙總函數的子句中未使用 ORDER BY 子句時，就不需要影格子句。如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。

未指定 ORDER BY 子句時，表示未繫結隱含影格，相當於 ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING。

ROWS

此子句定義視窗框作法是指定相對於目前列的實體位移。

此子句指定目前視窗或分割區中的列，以便與目前列的值結合。此子句使用引數來指定列位置，可能在目前列之前或之後。所有視窗框都以目前列為參考點。隨著視窗框在分割區中向前滑動，每一列會輪流變成目前列。

窗框可能是一組簡單的列，最遠到達且包含目前列。

```
{UNBOUNDED PRECEDING | offset PRECEDING | CURRENT ROW}
```

也可能是兩個邊界之間的一個列集。

```
BETWEEN
```

```
{ UNBOUNDED PRECEDING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }  
AND  
{ UNBOUNDED FOLLOWING | offset { PRECEDING | FOLLOWING } | CURRENT ROW }
```

UNBOUNDED PRECEDING 表示該視窗從分割區的第一列開始；偏移PRECEDING表示該視窗在目前資料列之前啟動了相當於偏移值的數個資料列。UNBOUNDED PRECEDING 為預設值。

CURRENT ROW 表示視窗在目前資料列開始或結束。

UNBOUNDED FOLLOWING 表示該視窗結束於分割區的最後一列；偏移FOLLOWING表示該視窗結束的列數相當於目前列之後的偏移值。

offset 表示目前列之前或之後的實體列數。在此案例中，*offset* 必須是評估為正數值的常數。例如，5 會在目前資料列之後FOLLOWING結束 5 資料列的影格。

BETWEEN 若未指定，則影格會隱含地由目前資料列繫結。例如，ROWS 5 PRECEDING 等於 ROWS BETWEEN 5 PRECEDING AND CURRENT ROW。此外，ROWS UNBOUNDED FOLLOWING 等於 ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING。

Note

您不能指定開始邊界大於結束邊界的窗框。例如，您不能指定下列任何窗框：

```
between 5 following and 5 preceding  
between current row and 2 preceding  
between 3 following and current row
```

範圍函數的資料唯一排序

如果視窗函數的 ORDER BY 子句未產生資料的唯一和總排序，則資料列的順序是非確定的。如果 ORDER BY 運算式產生重複的值（部分排序），則這些資料列的傳回順序可能會在多個執行中有所不同。在這種情況下，範圍函數也可能傳回非預期或不一致的結果。

例如，以下查詢會在多次執行中傳回不同的結果。會發生這些不同的結果，因為 order by dateid不會產生SUM視窗函數資料的唯一排序。

```
select dateid, pricepaid,  
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
```

```

from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
-----+-----+-----
1827 |   1730.00 |   1730.00
1827 |    708.00 |   2438.00
1827 |    234.00 |   2672.00
...

select dateid, pricepaid,
sum(pricepaid) over(order by dateid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid |   sumpaid
-----+-----+-----
1827 |    234.00 |    234.00
1827 |    472.00 |    706.00
1827 |    347.00 |   1053.00
...

```

在這種情況下，將第二個 ORDER BY 資料欄新增至視窗函數可以解決問題。

```

select dateid, pricepaid,
sum(pricepaid) over(order by dateid, pricepaid rows unbounded preceding) as sumpaid
from sales
group by dateid, pricepaid;

dateid | pricepaid | sumpaid
-----+-----+-----
1827 |    234.00 |  234.00
1827 |    337.00 |  571.00
1827 |    347.00 |  918.00
...

```

支援的函數

Amazon Redshift 支援兩種範圍函數：彙總和排名。

以下是支援的彙總函數：

- [AVG 視窗函數](#)

- [COUNT 視窗函數](#)
- [CUME_DIST 視窗函數](#)
- [DENSE_RANK 視窗函數](#)
- [FIRST_VALUE 視窗函數](#)
- [LAG 視窗函數](#)
- [LAST_VALUE 視窗函數](#)
- [LEAD 視窗函數](#)
- [LISTAGG 視窗函數](#)
- [MAX 視窗函數](#)
- [MEDIAN 視窗函數](#)
- [MIN 視窗函數](#)
- [NTH_VALUE 視窗函數](#)
- [PERCENTILE_CONT 視窗函數](#)
- [PERCENTILE_DISC 視窗函數](#)
- [RATIO_TO_REPORT 視窗函數](#)
- [STDDEV_SAMP 和 STDDEV_POP 視窗函數](#) ([STDDEV_SAMP](#) 和 [STDDEV](#)是同義詞)
- [SUM 視窗函數](#)
- [VAR_SAMP 和 VAR_POP 視窗函數](#) ([VAR_SAMP](#) 和 [VARIANCE](#)是同義詞)

以下是支援的排名函數：

- [DENSE_RANK 視窗函數](#)
- [NTILE 視窗函數](#)
- [PERCENT_RANK 視窗函數](#)
- [RANK 視窗函數](#)
- [ROW_NUMBER 視窗函數](#)

範圍函數範例的範例資料表

您可以在每個函數說明中找到特定的範圍函數範例。某些範例使用名為 `WINDSALES` 的資料表，其中包含 11 列，如下所示。

SALESID	DATEID	SELLERID	BUYERID	QTY	QTY_SHIPPED
30001	8/2/2003	3	B	10	10
10001	12/24/2003	1	C	10	10
10005	12/24/2003	1	A	30	
40001	1/9/2004	4	A	40	
10006	1/18/2004	1	C	10	
20001	2/12/2004	2	B	20	20
40005	2/12/2004	4	A	10	10
20002	2/16/2004	2	C	20	20
30003	4/18/2004	3	B	15	
30004	4/18/2004	3	B	20	
30007	9/7/2004	3	C	30	

下列指令碼會建立並填入範例WINSALES資料表。

```
CREATE TABLE winsales(  
  salesid int,  
  dateid date,  
  sellerid int,  
  buyerid char(10),  
  qty int,  
  qty_shipped int);  
  
INSERT INTO winsales VALUES  
  (30001, '8/2/2003', 3, 'b', 10, 10),  
  (10001, '12/24/2003', 1, 'c', 10, 10),  
  (10005, '12/24/2003', 1, 'a', 30, null),  
  (40001, '1/9/2004', 4, 'a', 40, null),  
  (10006, '1/18/2004', 1, 'c', 10, null),
```

```
(20001, '2/12/2004', 2, 'b', 20, 20),
(40005, '2/12/2004', 4, 'a', 10, 10),
(20002, '2/16/2004', 2, 'c', 20, 20),
(30003, '4/18/2004', 3, 'b', 15, null),
(30004, '4/18/2004', 3, 'b', 20, null),
(30007, '9/7/2004', 3, 'c', 30, null);
```

AVG 視窗函數

AVG 視窗函數會傳回輸入表達式值的平均值（算術平均值）。AVG 函數使用數值並忽略NULL值。

語法

```
AVG ( [ALL ] expression ) OVER
(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list
                frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL時，函數會保留運算式中的所有重複值以進行計數。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION 依 *expr_list* 排序

以一或多個表達式定義AVG函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱[範圍函數語法摘要](#)。

資料類型

AVG 函數支援的引數類型為 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL和 DOUBLE PRECISION。

AVG 函數支援的傳回類型為：

- BIGINT 適用於 SMALLINT或 INTEGER引數
- NUMERIC 適用於引BIGINT數
- DOUBLE PRECISION 適用於浮點數引數

範例

下列範例會依日期計算銷售數量的移動平均數；依日期 ID 和銷售 ID 排序結果：

```
select salesid, dateid, sellerid, qty,
avg(qty) over
(order by dateid, salesid rows unbounded preceding) as avg
from winsales
order by 2,1;
```

salesid	dateid	sellerid	qty	avg
30001	2003-08-02	3	10	10
10001	2003-12-24	1	10	10
10005	2003-12-24	1	30	16
40001	2004-01-09	4	40	22
10006	2004-01-18	1	10	20
20001	2004-02-12	2	20	20
40005	2004-02-12	4	10	18
20002	2004-02-16	2	20	18
30003	2004-04-18	3	15	18
30004	2004-04-18	3	20	18
30007	2004-09-07	3	30	19

(11 rows)

如需WINDSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

COUNT 視窗函數

COUNT 視窗函數會計算運算式定義的資料列。

COUNT 函數有兩種變化。COUNT (*) 會計算目標資料表中的所有資料列，無論是否包含 null。COUNT (運算式) 會計算特定資料欄或運算式中具有非NULL值的資料列數目。

語法

```
COUNT ( * | [ ALL ] expression) OVER  
(  
[ PARTITION BY expr_list ]  
[ ORDER BY order_list  
frame_clause ]  
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL時，函數會保留運算式中的所有重複值以進行計數。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION 依 expr_list 排序

以一或多個表達式定義COUNT函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱 [範圍函數語法摘要](#)。

資料類型

COUNT 函數支援所有引數資料類型。

COUNT 函數支援的傳回類型為 BIGINT。

範例

下列範例顯示從資料視窗開頭的所有列的銷售 ID、數量和計數：

```
select salesid, qty,
count(*) over (order by salesid rows unbounded preceding) as count
from winsales
order by salesid;
```

```
salesid | qty | count
-----+-----+-----
10001 | 10 | 1
10005 | 30 | 2
10006 | 10 | 3
20001 | 20 | 4
20002 | 20 | 5
30001 | 10 | 6
30003 | 15 | 7
30004 | 20 | 8
30007 | 30 | 9
40001 | 40 | 10
40005 | 10 | 11
(11 rows)
```

如需WINSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

下列範例顯示如何從資料視窗開頭計算非 null 列的銷售 ID、數量和計數。（在WINSALES表格中，QTY_SHIPPED 欄包含一些 NULLs。）

```
select salesid, qty, qty_shipped,
count(qty_shipped)
over (order by salesid rows unbounded preceding) as count
from winsales
order by salesid;
```

```
salesid | qty | qty_shipped | count
-----+-----+-----+-----
```

```

10001 | 10 | 10 | 1
10005 | 30 |  | 1
10006 | 10 |  | 1
20001 | 20 | 20 | 2
20002 | 20 | 20 | 3
30001 | 10 | 10 | 4
30003 | 15 |  | 4
30004 | 20 |  | 4
30007 | 30 |  | 4
40001 | 40 |  | 4
40005 | 10 | 10 | 5
(11 rows)

```

CUME_DIST 視窗函數

計算視窗或分割區內值的累積分佈。假定為遞增排序，使用此公式來決定累積分佈：

$$\text{count of rows with values } \leq x / \text{count of rows in the window or partition}$$

其中 x 等於 ORDER BY 子句中指定資料欄目前資料列中的值。以下資料集示範此公式的使用：

Row#	Value	Calculation	CUME_DIST
1	2500	(1)/(5)	0.2
2	2600	(2)/(5)	0.4
3	2800	(3)/(5)	0.6
4	2900	(4)/(5)	0.8
5	3100	(5)/(5)	1.0

傳回值範圍是 >0 至 1 (含)。

語法

```

CUME_DIST (
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)

```

引數

OVER

用於指定視窗分割的子句。OVER 子句不能包含窗框規格。

PARTITION BY partition_expression

選用。設定OVER子句中每個群組記錄範圍的表達式。

ORDER BY 訂單_清單

要計算累積分佈的表達式。表達式必須為數值資料類型，或可隱含地轉換為數值資料類型。如果省略 ORDER BY，則所有資料列的傳回值為 1。

如果 ORDER BY 未產生唯一的排序，則資料列的順序是非確定的。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

傳回類型

FLOAT8

範例

以下範例計算每一個賣方的數量累積分佈：

```
select sellerid, qty, cume_dist()
over (partition by sellerid order by qty)
from winsales;
```

sellerid	qty	cume_dist
1	10.00	0.33
1	10.64	0.67
1	30.37	1
3	10.04	0.25
3	15.15	0.5
3	20.75	0.75
3	30.55	1
2	20.09	0.5
2	20.12	1
4	10.12	0.5
4	40.23	1

如需WINSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

DENSE_RANK 視窗函數

DENSE_RANK 視窗函數會根據子OVER句中的 ORDER BY 表達式，決定值群組中值的排名。如果存在選用的 PARTITION BY 子句，則會為每個資料列群組重設排名。在排名準則中有相等值的列獲得相同排名。DENSE_RANK 函數RANK在某個方面與不同：如果兩個或更多資料列綁定，排序值的序列中不會有間隙。例如，假設兩列都排名 1，則下一個排名為 2。

您可以在相同的查詢中使用具有不同 PARTITION BY 和 ORDER BY 子句的排序函數。

語法

```
DENSE_RANK() OVER  
(  
[ PARTITION BY expr_list ]  
[ ORDER BY order_list ]  
)
```

引數

()

此函數不接受引數，但需要空括號。

OVER

DENSE_RANK 函數的視窗子句。

PARTITION 依 *expr_list* 排序

(選用) 一或多個用於定義視窗的運算式。

ORDER BY 訂單_清單

(選用) 排名值所根據的運算式。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。如果省略 ORDER BY，則傳回值1適用於所有資料列。

如果 ORDER BY 未產生唯一的排序，則資料列的順序是非確定的。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

傳回類型

INTEGER

範例

下列範例使用範圍函數的範例資料表。如需詳細資訊，請參閱[範圍函數範例的範例資料表](#)。

以下範例依銷售數量排序資料表，並將密集排名和一般排名指派給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, qty,
DENSE_RANK() OVER(ORDER BY qty DESC) AS d_rnk,
RANK() OVER(ORDER BY qty DESC) AS rnk
FROM winsales
ORDER BY 2,1;
```

salesid	qty	d_rnk	rnk
10001	10	5	8
10006	10	5	8
30001	10	5	8
40005	10	5	8
30003	15	4	7
20001	20	3	4
20002	20	3	4
30004	20	3	4
10005	30	2	2
30007	30	2	2
40001	40	1	1

請注意，在相同查詢中並排使用 DENSE_RANK 和 RANK 函數時，指派給相同資料列的排名差異。

下列範例會依 sellerid 分割資料表，並依數量排序每一個分割區，然後指派密集排名給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, sellerid, qty,
DENSE_RANK() OVER(PARTITION BY sellerid ORDER BY qty DESC) AS d_rnk
FROM winsales
ORDER BY 2,3,1;
```

salesid	sellerid	qty	d_rnk
10001	1	10	2

10006	1	10	2
10005	1	30	1
20001	2	20	1
20002	2	20	1
30001	3	10	4
30003	3	15	3
30004	3	20	2
30007	3	30	1
40005	4	10	2
40001	4	40	1

若要成功使用最後一個範例，請使用下列命令將資料列插入WINSALES資料表。此列與另一列具有相同的 buyid、sellerid 和 qtysold。這會導致最後一個範例中的兩列綁定，因此會顯示 DENSE_RANK 和 RANK函數之間的差異。

```
INSERT INTO winsales VALUES(30009, '2/2/2003', 3, 'b', 20, NULL);
```

下列範例會依 buyerid 和 sellerid 分割資料表，並依數量排序每一個分割區，然後同時指派密集排名和一般排名給每一列。(選用) 套用範圍函數之後對結果進行排序。

```
SELECT salesid, sellerid, qty, buyerid,
DENSE_RANK() OVER(PARTITION BY buyerid, sellerid ORDER BY qty DESC) AS d_rnk,
RANK() OVER (PARTITION BY buyerid, sellerid ORDER BY qty DESC) AS rnk
FROM winsales
ORDER BY rnk;
```

salesid	sellerid	qty	buyerid	d_rnk	rnk
20001	2	20	b	1	1
30007	3	30	c	1	1
10006	1	10	c	1	1
10005	1	30	a	1	1
20002	2	20	c	1	1
30009	3	20	b	1	1
40001	4	40	a	1	1
30004	3	20	b	1	1
10001	1	10	c	1	1
40005	4	10	a	2	2
30003	3	15	b	2	3
30001	3	10	b	3	4

```
+-----+-----+-----+-----+-----+-----+
```

FIRST_VALUE 視窗函數

指定一組有序的資料列，FIRST_VALUE 會傳回指定表達式相對於視窗框中第一列的值。

如需有關選取視窗框中最後一列的資訊，請參閱[LAST_VALUE 視窗函數](#)。

語法

```
FIRST_VALUE( expression ) [ IGNORE NULLS | RESPECT NULLS ]  
OVER (  
  [ PARTITION BY expr_list ]  
  [ ORDER BY order_list frame_clause ]  
)
```

引數

運算式

函數運算的目標欄或表達式。

IGNORE NULLS

當此選項與 FIRST_ 搭配使用時 VALUE，函數會傳回影格中的第一個值 NULL (如果 NULL 所有值都是 NULL)。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。RESPECT NULLS 如果您未指定，預設支援 IGNORE NULLS。

OVER

引進函數的視窗子句。

PARTITION 依 *expr_list* 排序

以一或多個表達式定義函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY 子句，ORDERBY 會排序整個資料表。如果您指定 ORDER BY 子句，則還必須指定 *frame_clause*。

FIRST_VALUE 函數的結果取決於資料的排序。在下列情況中，結果不確定：

- 未指定 ORDER BY 子句時，且分割區包含兩個不同的運算式值
- 當表達式評估為與 ORDER BY 清單中相同值對應的不同值時。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果中包含或排除資料列組，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱 [範圍函數語法摘要](#)。

傳回類型

這些函數支援使用基本 Amazon Redshift 資料類型的運算式。傳回類型與運算式的資料類型相同。

範例

下列範例使用範例TICKIT資料的VENUE資料表。如需詳細資訊，請參閱[範本資料庫](#)。

下列範例會傳回VENUE資料表中每個場地的座位容量，以及依容量排序的結果（從高到低）。FIRST_VALUE 函數用於選取與影格中第一列對應的場地名稱：在此情況下，是座位數量最高的列。結果會依狀態分割，因此當VENUESTATE值變更時，會選取新的第一個值。視窗框無界限，對於每一個分割區的第一列，選取的第一個值都相同。

以加利佛尼亞來說，Qualcomm Stadium 的座位數最多 (70561)，因此，對於 CA 分割區中的所有列，此名稱是第一個值。

```
select venuestate, venueseats, venueName,
first_value(venueName)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;
```

venuestate	venueseats	venueName	first_value
CA	70561	Qualcomm Stadium	Qualcomm Stadium
CA	69843	Monster Park	Qualcomm Stadium
CA	63026	McAfee Coliseum	Qualcomm Stadium
CA	56000	Dodger Stadium	Qualcomm Stadium

CA	45050	Angel Stadium of Anaheim	Qualcomm Stadium
CA	42445	PETCO Park	Qualcomm Stadium
CA	41503	AT&T Park	Qualcomm Stadium
CA	22000	Shoreline Amphitheatre	Qualcomm Stadium
CO	76125	INVESCO Field	INVESCO Field
CO	50445	Coors Field	INVESCO Field
DC	41888	Nationals Park	Nationals Park
FL	74916	Dolphin Stadium	Dolphin Stadium
FL	73800	Jacksonville Municipal Stadium	Dolphin Stadium
FL	65647	Raymond James Stadium	Dolphin Stadium
FL	36048	Tropicana Field	Dolphin Stadium
...			

下列範例顯示 IGNORENULLS 選項的使用，並依賴於將新資料列新增至 VENUE 資料表：

```
insert into venue values(2000,null,'Stanford','CA',90000);
```

此新資料列包含 VENUENAME 資料欄 NULL 的值。現在重複本節前面顯示的 FIRST_VALUE 查詢：

```
select venuestate, venueseats, venuename,
first_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;
```

venuestate	venueseats	venuename	first_value
CA	90000	NULL	NULL
CA	70561	Qualcomm Stadium	NULL
CA	69843	Monster Park	NULL
...			

由於新資料列包含最高 VENUESEATS 值 (90000)，且其 VENUENAME 為 NULL，因此 FIRST_VALUE 函數會 NULL 傳回 CA 分割區。若要在函數評估中忽略像這樣的資料列，請將 IGNORENULLS 選項新增至函數引數：

```
select venuestate, venueseats, venuename,
first_value(venuename) ignore nulls
over(partition by venuestate
order by venueseats desc
```

```
rows between unbounded preceding and unbounded following)
from (select * from venue where venuestate='CA')
order by venuestate;
```

venuestate	venue seats	venue name	first_value
CA	90000	NULL	Qualcomm Stadium
CA	70561	Qualcomm Stadium	Qualcomm Stadium
CA	69843	Monster Park	Qualcomm Stadium
...			

LAG 視窗函數

LAG 視窗函數會傳回分割區中目前資料列上方（之前）指定位移的資料列值。

語法

```
LAG (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

引數

value_expr

函數運算的目標欄或表達式。

offset

選擇性參數，指定在目前列之前要傳回值的列數。位移可以是常數整數，或評估為整數的表達式。如果您不指定位移，Amazon Redshift 會使用 1 做為預設值。位移 0 表示目前列。

IGNORE NULLS

選擇性規格，指出 Amazon Redshift 在決定要使用的列時應該略過 Null 值。如果 IGNORE NULLS 未列出，則會包含空值。

Note

您可以使用 NVL 或 COALESCE 表達式將 null 值取代為另一個值。如需詳細資訊，請參閱 [NVL 和 COALESCE 函數](#)。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。RESPECT NULLS 如果您未指定，預設支援 IGNORE NULLS。

OVER

指定視窗分割和排序。OVER 子句不能包含窗框規格。

PARTITION 依 window_partition 排序

選擇性引數，可設定OVER子句中每個群組的記錄範圍。

ORDER BY window_ordering

排序每一個分割區內的列。

LAG 視窗函數支援使用任何 Amazon Redshift 資料類型的表達式。傳回類型與 value_expr 的類型相同。

範例

下列範例顯示銷售給買方 ID 為 3 之買方的門票數量，以及買方 3 購買門票的時間。為了比較買方 3 的每次銷售與前次銷售，查詢會傳回每次銷售的前次銷售數量。因為 2008/1/16 之前沒有購買，所以第一個先前銷售數量值為 Null：

```
select buyerid, saletime, qtysold,
lag(qtysold,1) over (order by buyerid, saletime) as prev_qtysold
from sales where buyerid = 3 order by buyerid, saletime;
```

buyerid	saletime	qtysold	prev_qtysold
3	2008-01-16 01:06:09	1	
3	2008-01-28 02:10:01	1	1
3	2008-03-12 10:39:53	1	1
3	2008-03-13 02:56:07	1	1
3	2008-03-29 08:21:39	2	1
3	2008-04-27 02:39:01	1	2
3	2008-08-16 07:04:37	2	1
3	2008-08-22 11:45:26	2	2
3	2008-09-12 09:11:25	1	2
3	2008-10-01 06:22:37	1	1
3	2008-10-20 01:55:51	2	1
3	2008-10-28 01:30:40	1	2

(12 rows)

LAST_VALUE 視窗函數

指定一組有序的資料列，LAST_VALUE 函數會傳回相對於影格中最後一列的表達式值。

如需有關選取框架中第一列的資訊，請參閱[FIRST_VALUE 視窗函數](#)。

語法

```
LAST_VALUE( expression ) [ IGNORE NULLS | RESPECT NULLS ]  
OVER (  
  [ PARTITION BY expr_list ]  
  [ ORDER BY order_list frame_clause ]  
)
```

引數

運算式

函數運算的目標欄或表達式。

IGNORE NULLS

函數會傳回框架中最後一個值，但不是 NULL (或 NULL 所有值皆為 NULL)。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。RESPECT NULLS 如果您未指定，預設支援 IGNORE NULLS。

OVER

引進函數的視窗子句。

PARTITION BY *expr_list*

以一或多個表達式定義函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY 子句，ORDERBY 會排序整個資料表。如果您指定 ORDER BY 子句，則還必須指定 *frame_clause*。

結果取決於資料的順序。在下列情況中，結果不確定：

- 未指定 ORDER BY 子句時，且分割區包含兩個不同的運算式值
- 當表達式評估為與 ORDER BY 清單中相同值對應的不同值時。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果中包含或排除資料列組，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱 [範圍函數語法摘要](#)。

傳回類型

這些函數支援使用基本 Amazon Redshift 資料類型的運算式。傳回類型與運算式的資料類型相同。

範例

下列範例使用範例TICKIT資料的VENUE資料表。如需詳細資訊，請參閱[範本資料庫](#)。

下列範例會傳回VENUE資料表中每個場地的座位容量，以及依容量排序的結果（從高到低）。LAST_VALUE 函數用於選取對應於影格中最後一列的場地名稱：在此情況下，是座位數量最少的列。結果會依狀態分割，因此當VENUESTATE值變更時，會選取新的最後一個值。視窗框無界限，對於每一個分割區的第一列，選取的最後一個值都相同。

以加利佛尼亞來說，分割區中的每一列列都傳回 Shoreline Amphitheatre，因為其座位數最少(22000)。

```
select venuestate, venueseats, venuename,
last_value(venuename)
over(partition by venuestate
order by venueseats desc
rows between unbounded preceding and unbounded following)
from (select * from venue where venueseats >0)
order by venuestate;
```

venuestate	venueseats	venuename	last_value
CA	70561	Qualcomm Stadium	Shoreline Amphitheatre
CA	69843	Monster Park	Shoreline Amphitheatre
CA	63026	McAfee Coliseum	Shoreline Amphitheatre
CA	56000	Dodger Stadium	Shoreline Amphitheatre
CA	45050	Angel Stadium of Anaheim	Shoreline Amphitheatre
CA	42445	PETCO Park	Shoreline Amphitheatre

CA		41503		AT&T Park		Shoreline Amphitheatre
CA		22000		Shoreline Amphitheatre		Shoreline Amphitheatre
CO		76125		INVESCO Field		Coors Field
CO		50445		Coors Field		Coors Field
DC		41888		Nationals Park		Nationals Park
FL		74916		Dolphin Stadium		Tropicana Field
FL		73800		Jacksonville Municipal Stadium		Tropicana Field
FL		65647		Raymond James Stadium		Tropicana Field
FL		36048		Tropicana Field		Tropicana Field
...						

LEAD 視窗函數

LEAD 視窗函數會在分割區中目前資料列下方（之後）的指定偏移中傳回資料列的值。

語法

```
LEAD (value_expr [, offset ])
[ IGNORE NULLS | RESPECT NULLS ]
OVER ( [ PARTITION BY window_partition ] ORDER BY window_ordering )
```

引數

value_expr

函數運算的目標欄或表達式。

offset

選擇性參數，指定在目前列下方要傳回值的列數。位移可以是常數整數，或評估為整數的表達式。如果您不指定位移，Amazon Redshift 會使用 1 做為預設值。位移 0 表示目前列。

IGNORE NULLS

選擇性規格，指出 Amazon Redshift 在決定要使用的列時應該略過 Null 值。如果 IGNORE NULLS 未列出，則會包含空值。

Note

您可以使用 NVL 或 COALESCE 表達式將 null 值取代為另一個值。如需詳細資訊，請參閱 [NVL 和 COALESCE 函數](#)。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。RESPECT NULLS 如果您未指定，預設支援 IGNORE NULLS。

OVER

指定視窗分割和排序。OVER 子句不能包含窗框規格。

PARTITION BY window_partition

選擇性引數，可設定OVER子句中每個群組的記錄範圍。

ORDER BY window_ordering

排序每一個分割區內的列。

LEAD 視窗函數支援使用任何 Amazon Redshift 資料類型的表達式。傳回類型與 value_expr 的類型相同。

範例

下列範例提供SALES資料表中於 2008 年 1 月 1 日和 2008 年 1 月 2 日售出之門票的事件佣金，以及針對後續售出之門票銷售支付的佣金。下列範例使用TICKIT範例資料庫。如需詳細資訊，請參閱[範本資料庫](#)。

```
SELECT eventid, commission, saletime, LEAD(commission, 1) over ( ORDER BY saletime ) AS
next_comm
FROM sales
WHERE saletime BETWEEN '2008-01-09 00:00:00' AND '2008-01-10 12:59:59'
LIMIT 10;
```

eventid	commission	saletime	next_comm
1664	13.2	2008-01-09 01:00:21	69.6
184	69.6	2008-01-09 01:00:36	116.1
6870	116.1	2008-01-09 01:02:37	11.1
3718	11.1	2008-01-09 01:05:19	205.5
6772	205.5	2008-01-09 01:14:04	38.4
3074	38.4	2008-01-09 01:26:50	209.4
5254	209.4	2008-01-09 01:29:16	26.4
3724	26.4	2008-01-09 01:40:09	57.6

```

| 5303 | 57.6 | 2008-01-09 01:40:21 | 51.6 |
| 3678 | 51.6 | 2008-01-09 01:42:54 | 43.8 |
+-----+-----+-----+-----+

```

下列範例提供 SALES 資料表中事件的通訊差異上限，以及針對相同事件的後續銷售所支付的票證銷售佣金。此範例示範如何 LEAD 搭配 GROUP BY 子句使用。由於彙總子句中不允許視窗函數，因此此範例會使用子查詢。下列範例使用 TICKIT 範例資料庫。如需詳細資訊，請參閱 [範本資料庫](#)。

```

SELECT eventid, eventname, max(next_comm_diff) as max_commission_difference
FROM
(
  SELECT sales.eventid, eventname, commission - LEAD(commission, 1) over (ORDER BY
sales.eventid, saletime) AS next_comm_diff
  FROM sales JOIN event ON sales.eventid = event.eventid
)
GROUP BY eventid, eventname
ORDER BY eventid

LIMIT 10

```

```

| eventid | eventname | max_commission_difference |
+-----+-----+-----+
| 1 | Gotterdammerung | 7.95 |
| 2 | Boris Godunov | 227.85 |
| 3 | Salome | 1350.9 |
| 4 | La Cenerentola (Cinderella) | 790.05 |
| 5 | Il Trovatore | 214.05 |
| 6 | L'Elisir d'Amore | 510.9 |
| 7 | Doctor Atomic | 180.6 |
| 9 | The Fly | 147 |
| 10 | Rigoletto | 186.6 |
+-----+-----+-----+

```

LISTAGG 視窗函數

對於查詢中的每個群組，LISTAGG 視窗函數會根據 ORDER BY 表達式排序該群組的資料列，然後將值串連成單一字串。

語法

```

LISTAGG( [DISTINCT] expression [, 'delimiter' ] )
[ WITHIN GROUP (ORDER BY order_list) ]

```

```
OVER ( [PARTITION BY partition_expression] )
```

引數

DISTINCT

(選用) 此子句在串連值之前會從指定的表達式中消除重複值。結尾空格會忽略，所以字串 'a' 和 'a ' 視為重複值。LISTAGG 會使用遇到的第一個值。如需詳細資訊，請參閱[多餘空格的意義](#)。

aggregate_expression

任何有效表達式 (例如欄名)，用於提供要彙總的值。NULL 值和空字串會被忽略。

delimiter

(選用) 用來區隔串連值的字串常數。預設值為 NULL。

WITHIN GROUP (ORDER BY order_list)

(選用) 此子句指定彙總值的排序順序。只有在 ORDER BY 提供唯一的排序時，才能確定。預設是彙總所有列並傳回單一值。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗順序或視窗影格規格。

PARTITION BY partition_expression

(選用) 設定OVER子句中每個群組的記錄範圍。

傳回值

VARCHAR (MAX)。如果結果集大於VARCHAR大小上限 (64K – 1 或 65535)，則 會LISTAGG傳回下列錯誤：

```
Invalid operation: Result size exceeds LISTAGG limit
```

範例

下列範例使用 WINDSALES 資料表。如需WINDSALES資料表的說明，請參閱[範圍函數範例的範例資料表](#)。

下列範例會傳回賣方 ID 排序的賣方 清單。

```
select listagg(sellerid)
within group (order by sellerid)
over() from winsales;
```

```
listagg
-----
11122333344
...
...
11122333344
11122333344
(11 rows)
```

下列範例會傳回依日期排序IDs的買方 B 賣方清單。

```
select listagg(sellerid)
within group (order by dateid)
over () as seller
from winsales
where buyerid = 'b' ;
```

```
seller
-----
3233
3233
3233
3233
```

下列範例以逗號分隔清單傳回買方 B 的銷售日期。

```
select listagg(dateid,',')
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid = 'b';
```

```
dates
-----
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
```

```
2003-08-02,2004-04-18,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-04-18,2004-02-12
```

下列範例使用 DISTINCT 來傳回買方 B 的唯一銷售日期清單。

```
select listagg(distinct dateid,',' )
within group (order by sellerid desc,salesid asc)
over () as dates
from winsales
where buyerid = 'b';
```

```
          dates
-----
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
2003-08-02,2004-04-18,2004-02-12
```

下列範例會傳回 IDs 每個買方 ID 的逗號分隔銷售清單。

```
select buyerid,
listagg(salesid,',' )
within group (order by salesid)
over (partition by buyerid) as sales_id
from winsales
order by buyerid;
```

```
+-----+-----+
| buyerid |      sales_id      |
+-----+-----+
| a       | 10005,40001,40005 |
| a       | 10005,40001,40005 |
| a       | 10005,40001,40005 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| b       | 20001,30001,30003,30004 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
| c       | 10001,10006,20002,30007 |
+-----+-----+
```


MAX 視窗函數

MAX 視窗函數會傳回輸入表達式值的最大值。MAX 函數使用數值並忽略NULL值。

語法

```
MAX ( [ ALL ] expression ) OVER  
(  
[ PARTITION BY expr_list ]  
[ ORDER BY order_list frame_clause ]  
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL，函數會保留運算式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

此子句指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION 依 *expr_list* 排序

以一或多個表達式定義MAX函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱[範圍函數語法摘要](#)。

資料類型

接受任何資料類型做為輸入。傳回與 expression 相同的資料類型。

範例

下列範例顯示資料視窗開頭的銷售 ID、數量和最大數量：

```
select salesid, qty,
max(qty) over (order by salesid rows unbounded preceding) as max
from winsales
order by salesid;
```

salesid	qty	max
10001	10	10
10005	30	30
10006	10	30
20001	20	30
20002	20	30
30001	10	30
30003	15	30
30004	20	30
30007	30	30
40001	40	40
40005	10	40

(11 rows)

如需WINDSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

下列範例顯示受限窗框中的 salesid、數量和最大數量：

```
select salesid, qty,
max(qty) over (order by salesid rows between 2 preceding and 1 preceding) as max
from winsales
order by salesid;
```

salesid	qty	max
10001	10	
10005	30	10
10006	10	30
20001	20	30
20002	20	20
30001	10	20
30003	15	20
30004	20	15

```
30007 | 30 | 20
40001 | 40 | 30
40005 | 10 | 40
(11 rows)
```

MEDIAN 視窗函數

計算視窗或分割區內值範圍的中位數。NULL 會忽略 範圍內的值。

MEDIAN 是假設連續分佈模型的反分佈函數。

語法

```
MEDIAN ( median_expression )
OVER ( [ PARTITION BY partition_expression ] )
```

引數

median_expression

此表達式 (例如欄名) 提供要決定中位數的值。表達式必須為數值或日期時間資料類型，或可隱含地轉換為這種資料類型。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗順序或視窗影格規格。

PARTITION BY partition_expression

選用。設定OVER子句中每個群組記錄範圍的表達式。

資料類型

傳回類型取決於 median_expression 的資料類型。下表顯示每一個 median_expression 資料類型的傳回類型。

輸入類型	傳回類型
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL
FLOAT, DOUBLE	DOUBLE

輸入類型	傳回類型
DATE	DATE

使用須知

如果 `median_expression` 引數是定義的最高精確度為 38 位數的 DECIMAL 資料類型，MEDIAN 則可能會傳回不正確的結果或錯誤。如果 MEDIAN 函數的傳回值超過 38 位數，則結果會截斷為適合，這會導致精確度損失。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些情況，建議使用精確度較低的資料類型，或將 `median_expression` 引數轉換為較低精確度。

例如，具有 DECIMAL 引數的 SUM 函數會傳回預設精確度 38 位數。結果的小數位數和引數的小數位數相同。例如，DECIMAL (5 , 2) 資料欄 SUM 的 會傳回 DECIMAL (38 , 2) 資料類型。

下列範例會在 SUM 函數的 `median_expression` 引數中使用 MEDIAN 函數。資料 PRICEPAID 欄的資料類型為 DECIMAL (8 , 2) ，因此 SUM 函數會傳回 DECIMAL (38 , 2) 。

```
select salesid, sum(pricepaid), median(sum(pricepaid))
over() from sales where salesid < 10 group by salesid;
```

為了避免可能的精確度損失或溢位錯誤，請將結果轉換為精確度較低的 DECIMAL 資料類型，如下列範例所示。

```
select salesid, sum(pricepaid), median(sum(pricepaid)::decimal(30,2))
over() from sales where salesid < 10 group by salesid;
```

範例

以下範例計算每一個賣方的銷售數量中位數：

```
select sellerid, qty, median(qty)
over (partition by sellerid)
from winsales
order by sellerid;

sellerid qty median
-----
1 10 10.0
```

```
1 10 10.0
1 30 10.0
2 20 20.0
2 20 20.0
3 10 17.5
3 15 17.5
3 20 17.5
3 30 17.5
4 10 25.0
4 40 25.0
```

如需WINDSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

MIN 視窗函數

MIN 視窗函數會傳回輸入表達式值的最小值。MIN 函數使用數值並忽略NULL值。

語法

```
MIN ( [ ALL ] expression ) OVER
(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL，函數會保留運算式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION BY *expr_list*

以一或多個表達式定義MIN函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱 [範圍函數語法摘要](#)。

資料類型

接受任何資料類型做為輸入。傳回與 expression 相同的資料類型。

範例

下列範例顯示資料視窗開頭的銷售 ID、數量和最小數量：

```
select salesid, qty,  
min(qty) over  
(order by salesid rows unbounded preceding)  
from winsales  
order by salesid;
```

```
salesid | qty | min  
-----+-----+-----  
10001 | 10 | 10  
10005 | 30 | 10  
10006 | 10 | 10  
20001 | 20 | 10  
20002 | 20 | 10  
30001 | 10 | 10  
30003 | 15 | 10  
30004 | 20 | 10  
30007 | 30 | 10  
40001 | 40 | 10  
40005 | 10 | 10  
(11 rows)
```

如需WINSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

下列範例顯示受限窗框中的銷售 ID、數量和最小數量：

```
select salesid, qty,  
min(qty) over
```

```
(order by salesid rows between 2 preceding and 1 preceding) as min
from winsales
order by salesid;
```

```
salesid | qty | min
-----+-----+-----
10001 | 10 |
10005 | 30 | 10
10006 | 10 | 10
20001 | 20 | 10
20002 | 20 | 10
30001 | 10 | 20
30003 | 15 | 10
30004 | 20 | 10
30007 | 30 | 15
40001 | 40 | 20
40005 | 10 | 30
(11 rows)
```

NTH_VALUE 視窗函數

NTH_VALUE 視窗函數會傳回相對於視窗第一列的視窗框指定資料列的表達式值。

語法

```
NTH_VALUE (expr, offset)
[ IGNORE NULLS | RESPECT NULLS ]
OVER
( [ PARTITION BY window_partition ]
  [ ORDER BY window_ordering
             frame_clause ] )
```

引數

expr

函數運算的目標欄或表達式。

offset

相對於視窗中的第一列，決定要傳回表達式的列號。offset 可以是常數或表達式，且必須為大於 0 的正整數。

IGNORE NULLS

選擇性規格，指出 Amazon Redshift 在決定要使用的列時應該略過 Null 值。如果 IGNORE NULLS 未列出，則會包含空值。

RESPECT NULLS

指出 Amazon Redshift 應該包含 null 值來決定要使用的列。RESPECT NULLS 如果您未指定，預設支援 IGNORE NULLS。

OVER

指定視窗分割、排序及視窗框。

PARTITION 依 window_partition 排序

設定 OVER 子句中每個群組的記錄範圍。

ORDER BY window_ordering

排序每一個分割區內的列。如果省略 ORDER BY，則預設規格會包含分割區中的所有資料列。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的規格子句。窗框子句在排序的結果中包含或排除資料列組，以調整函數視窗中的一個列集。框架子句由 ROWS 關鍵字和相關聯的規格符組成。請參閱 [範圍函數語法摘要](#)。

NTH_VALUE window 函數支援使用任何 Amazon Redshift 資料類型的表達式。傳回類型與 expr 的類型相同。

範例

下列範例顯示加利佛尼亞、佛羅里達及紐約的前三大會場的座位數，並對照這些州其他會場的座位數：

```
select venuestate, venuename, venueseats,
nth_value(venueseats, 3)
ignore nulls
over(partition by venuestate order by venueseats desc
rows between unbounded preceding and unbounded following)
as third_most_seats
from (select * from venue where venueseats > 0 and
venuestate in('CA', 'FL', 'NY'))
order by venuestate;
```

venuestate	venuename	venueseats	third_most_seats
------------	-----------	------------	------------------


```

-----+-----+-----+-----
CA      | Qualcomm Stadium          |      70561 |      63026
CA      | Monster Park              |      69843 |      63026
CA      | McAfee Coliseum           |      63026 |      63026
CA      | Dodger Stadium            |      56000 |      63026
CA      | Angel Stadium of Anaheim  |      45050 |      63026
CA      | PETCO Park                 |      42445 |      63026
CA      | AT&T Park                  |      41503 |      63026
CA      | Shoreline Amphitheatre    |      22000 |      63026
FL      | Dolphin Stadium           |      74916 |      65647
FL      | Jacksonville Municipal Stadium |      73800 |      65647
FL      | Raymond James Stadium     |      65647 |      65647
FL      | Tropicana Field           |      36048 |      65647
NY      | Ralph Wilson Stadium      |      73967 |      20000
NY      | Yankee Stadium            |      52325 |      20000
NY      | Madison Square Garden     |      20000 |      20000
(15 rows)

```

NTILE 視窗函數

NTILE 視窗函數會將分割區中的排序資料列分割為盡可能相同大小的指定排名群組數目，並傳回指定資料列所屬的群組。

語法

```

NTILE (expr)
OVER (
  [ PARTITION BY expression_list ]
  [ ORDER BY order_list ]
)

```

引數

expr

每一個分割區的排名群組數，且結果必須為整數值 (大於 0)。expr 引數必須不可為 Null。

OVER

用於指定視窗分割和排序的子句。OVER 子句不能包含窗框規格。

PARTITION 依 *window_partition* 排序

選用。OVER 子句中每個群組的記錄範圍。

ORDER BY window_ordering

選用。此表達式排序每一個分割區內的列。如果省略 ORDER BY 子句，則排名行為相同。

如果 ORDER BY 未產生唯一的排序，資料列的順序是非確定的。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

傳回類型

BIGINT

範例

下列範例將 2008 年 8 月 26 日 Hamlet 門票的支付價格分成四個排名群組。結果集有 17 列，幾乎平均分散於排名 1 到 4：

```
select eventname, caldate, pricepaid, ntile(4)
over(order by pricepaid desc) from sales, event, date
where sales.eventid=event.eventid and event.dateid=date.dateid and eventname='Hamlet'
and caldate='2008-08-26'
order by 4;
```

eventname	caldate	pricepaid	ntile
Hamlet	2008-08-26	1883.00	1
Hamlet	2008-08-26	1065.00	1
Hamlet	2008-08-26	589.00	1
Hamlet	2008-08-26	530.00	1
Hamlet	2008-08-26	472.00	1
Hamlet	2008-08-26	460.00	2
Hamlet	2008-08-26	355.00	2
Hamlet	2008-08-26	334.00	2
Hamlet	2008-08-26	296.00	2
Hamlet	2008-08-26	230.00	3
Hamlet	2008-08-26	216.00	3
Hamlet	2008-08-26	212.00	3
Hamlet	2008-08-26	106.00	3
Hamlet	2008-08-26	100.00	4
Hamlet	2008-08-26	94.00	4
Hamlet	2008-08-26	53.00	4
Hamlet	2008-08-26	25.00	4

(17 rows)

PERCENT_RANK 視窗函數

計算給定資料列的百分比排行。使用此公式決定百分比排名：

$$(x - 1) / (\text{the number of rows in the window or partition} - 1)$$

其中 x 是目前列的排名。以下資料集示範此公式的使用：

```
Row# Value Rank Calculation PERCENT_RANK
1 15 1 (1-1)/(7-1) 0.0000
2 20 2 (2-1)/(7-1) 0.1666
3 20 2 (2-1)/(7-1) 0.1666
4 20 2 (2-1)/(7-1) 0.1666
5 30 5 (5-1)/(7-1) 0.6666
6 30 5 (5-1)/(7-1) 0.6666
7 40 7 (7-1)/(7-1) 1.0000
```

傳回值範圍是 0 至 1 (含)。任何集合中的第一列的 PERCENT_RANK 為 0。

語法

```
PERCENT_RANK ( )
OVER (
[ PARTITION BY partition_expression ]
[ ORDER BY order_list ]
)
```

引數

()

此函數不接受引數，但需要空括號。

OVER

用於指定視窗分割的子句。OVER 子句不能包含窗框規格。

PARTITION BY *partition_expression*

選用。設定 OVER 子句中每個群組記錄範圍的表達式。

ORDER BY 訂單_清單

選用。要計算百分比排名的表達式。表達式必須為數值資料類型，或可隱含地轉換為數值資料類型。如果省略 ORDER BY，則所有資料列的傳回值為 0。

如果 ORDER BY 未產生唯一的排序，資料列的順序是非確定的。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

傳回類型

FLOAT8

範例

以下範例計算每一個賣方的銷售數量百分比排名：

```
select sellerid, qty, percent_rank()  
over (partition by sellerid order by qty)  
from winsales;
```

```
sellerid qty  percent_rank  
-----
```

```
1  10.00  0.0  
1  10.64  0.5  
1  30.37  1.0  
3  10.04  0.0  
3  15.15  0.33  
3  20.75  0.67  
3  30.55  1.0  
2  20.09  0.0  
2  20.12  1.0  
4  10.12  0.0  
4  40.23  1.0
```

如需WINSALES資料表的說明，請參閱[範圍函數範例的範例資料表](#)。

PERCENTILE_CONT 視窗函數

PERCENTILE_CONT 是假設連續分佈模型的反分佈函數。它採用百分位數值和排序規格，且會傳回插入值，該值將根據排序規格落入給定的百分位數值。

PERCENTILE_CONT 會在值排序後計算值之間的線性插補。此函數在列根據排序規格來排序後，使用彙總群組中的百分位數值 (P) 和非 Null 列數 (N) 來計算列號。此列號 (RN) 是根據公式 $RN = (1 + (P * (N - 1)))$ 來計算。彙總函數的最終結果是以列號 $CRN = CEILING(RN)$ 到 $FRN = FLOOR(RN)$ 各列的值之間的線性插值來計算。

最終結果如下。

如果 (CRN = FRN = RN) , 則結果為 (value of expression from row at RN)

否則結果如下 :

$(CRN - RN) * (\text{value of expression for row at FRN}) + (RN - FRN) * (\text{value of expression for row at CRN})$.

您只能在PARTITION子句中指定OVER子句。如果PARTITION指定了 , 則對於每一列 , PERCENTILE_CONT 會傳回介於指定分割區中一組值之間的指定百分位數值。

語法

```
PERCENTILE_CONT ( percentile )
WITHIN GROUP (ORDER BY expr)
OVER ( [ PARTITION BY expr_list ] )
```

引數

percentile

介於 0 和 1 之間的數值常數。計算時會忽略 Null。

WITHIN GROUP (ORDER BY *expr*)

指定要排序和計算百分位數的數值或日期/時間值。

OVER

指定視窗分割。OVER 子句不能包含視窗順序或視窗影格規格。

PARTITION 依 *expr*

選擇性引數 , 可設定OVER子句中每個群組的記錄範圍。

傳回值

傳回類型取決於WITHINGROUP子句中 ORDER BY 表達式的資料類型。下表顯示每個 ORDER BY 表達式資料類型的傳回類型。

輸入類型	傳回類型
INT2, INT4, INT8, NUMERIC, DECIMAL	DECIMAL

輸入類型	傳回類型
FLOAT, DOUBLE	DOUBLE
DATE	DATE
TIMESTAMP	TIMESTAMP

使用須知

如果 ORDER BY 表達式是定義的最高精確度為 38 位數的 DECIMAL 資料類型，PERCENTILE_CONT 可能會傳回不正確的結果或錯誤。如果 PERCENTILE_CONT 函數的傳回值超過 38 位數，則結果會截斷為適合，這會導致精確度損失。在插補期間，如果中間結果超過最大精確度，則會發生數值溢位，且函數會傳回錯誤。為了避免這些條件，我們建議您使用精確度較低的資料類型，或將 ORDER BY 表達式轉換為精確度較低的資料類型。

例如，具有 DECIMAL 引數的 SUM 函數會傳回預設精確度 38 位數。結果的小數位數和引數的小數位數相同。例如，DECIMAL (5 , 2) 資料欄 SUM 的 會傳回 DECIMAL (38 , 2) 資料類型。

下列範例使用 PERCENTILE_CONT SUM 函數 ORDER BY 子句中的 函數。資料 PRICEPAID 欄的資料類型為 DECIMAL (8 , 2) ，因此 SUM 函數會傳回 DECIMAL (38 , 2) 。

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid) desc) over()
from sales where salesid < 10 group by salesid;
```

為了避免可能的精確度損失或溢位錯誤，請將結果轉換為精確度較低的 DECIMAL 資料類型，如下列範例所示。

```
select salesid, sum(pricepaid), percentile_cont(0.6)
within group (order by sum(pricepaid)::decimal(30,2) desc) over()
from sales where salesid < 10 group by salesid;
```

範例

下列範例使用 WINDSALES 資料表。如需 WINDSALES 資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
```

```
over() as median from winsales;
```

sellerid	qty	median
1	10	20.0
1	10	20.0
3	10	20.0
4	10	20.0
3	15	20.0
2	20	20.0
3	20	20.0
2	20	20.0
3	30	20.0
1	30	20.0
4	40	20.0

(11 rows)

```
select sellerid, qty, percentile_cont(0.5)
within group (order by qty)
over(partition by sellerid) as median from winsales;
```

sellerid	qty	median
2	20	20.0
2	20	20.0
4	10	25.0
4	40	25.0
1	10	10.0
1	10	10.0
1	30	10.0
3	10	17.5
3	15	17.5
3	20	17.5
3	30	17.5

(11 rows)

下列範例會計算華盛頓州賣方的售票銷售額 PERCENTILE_CONT 和 PERCENTILE_DISC。

```
SELECT sellerid, state, sum(qtysold*pricepaid) sales,
percentile_cont(0.6) within group (order by sum(qtysold*pricepaid)::decimal(14,2) )
desc) over(),
percentile_disc(0.6) within group (order by sum(qtysold*pricepaid)::decimal(14,2) )
desc) over()
```

```

from sales s, users u
where s.sellerid = u.userid and state = 'WA' and sellerid < 1000
group by sellerid, state;

```

sellerid	state	sales	percentile_cont	percentile_disc
127	WA	6076.00	2044.20	1531.00
787	WA	6035.00	2044.20	1531.00
381	WA	5881.00	2044.20	1531.00
777	WA	2814.00	2044.20	1531.00
33	WA	1531.00	2044.20	1531.00
800	WA	1476.00	2044.20	1531.00
1	WA	1177.00	2044.20	1531.00

(7 rows)

PERCENTILE_DISC 視窗函數

PERCENTILE_DISC 是假設離散分佈模型的反分佈函數。它採用百分位數值和排序規格，且會傳回給定集裡的一個元素。

對於指定的百分位數值 P，PERCENTILE_DISC 會排序 ORDER BY 子句中表達式的值，並傳回具有大於或等於 P 之最小累積分佈值（相對於相同排序規格）的值。

您只能在 PARTITION 子句中指定 OVER 子句。

語法

```

PERCENTILE_DISC ( percentile )
WITHIN GROUP ( ORDER BY expr )
OVER ( [ PARTITION BY expr_list ] )

```

引數

percentile

介於 0 和 1 之間的數值常數。計算時會忽略 Null。

WITHIN GROUP (ORDER BY expr)

指定要排序和計算百分位數的數值或日期/時間值。

OVER

指定視窗分割。OVER 子句不能包含視窗順序或視窗影格規格。

PARTITION 依 expr

選擇性引數，可設定OVER子句中每個群組的記錄範圍。

傳回值

與WITHINGROUP子句中的 ORDER BY 表達式相同的資料類型。

範例

下列範例使用 WINSALES 資料表。如需WINSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

```
SELECT sellerid, qty, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY qty)
OVER() AS MEDIAN FROM winsales;
```

sellerid	qty	median
3	10	20
1	10	20
1	10	20
4	10	20
3	15	20
2	20	20
2	20	20
3	20	20
1	30	20
3	30	20
4	40	20

```
SELECT sellerid, qty, PERCENTILE_DISC(0.5)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS MEDIAN FROM winsales;
```

sellerid	qty	median
4	10	10
4	40	10
3	10	15

3	15	15
3	20	15
3	30	15
2	20	20
2	20	20
1	10	10
1	10	10
1	30	10

若要尋找 PERCENTILE_DISC (0.25) 和 PERCENTILE_DISC (0.75) 作為依賣方 ID 分割的數量，請使用下列範例。

```
SELECT sellerid, qty, PERCENTILE_DISC(0.25)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS quartile1 FROM winsales;
```

sellerid	qty	quartile1
4	10	10
4	40	10
2	20	20
2	20	20
3	10	10
3	15	10
3	20	10
3	30	10
1	10	10
1	10	10
1	30	10

```
SELECT sellerid, qty, PERCENTILE_DISC(0.75)
WITHIN GROUP (ORDER BY qty)
OVER(PARTITION BY sellerid) AS quartile3 FROM winsales;
```

sellerid	qty	quartile3
3	10	20
3	15	20
3	20	20

```

| 3      | 30 | 20      |
| 4      | 10 | 40      |
| 4      | 40 | 40      |
| 2      | 20 | 20      |
| 2      | 20 | 20      |
| 1      | 10 | 30      |
| 1      | 10 | 30      |
| 1      | 30 | 30      |
+-----+-----+-----+

```

RANK 視窗函數

RANK 視窗函數會根據子OVER句中的 ORDER BY 表達式，決定值群組中值的排名。如果存在選用的 PARTITION BY 子句，則會為每個資料列群組重設排名。在排名準則中有相等值的列獲得相同排名。Amazon Redshift 將綁定的行數添加到綁定的排名中以計算下一個排名，因此排名可能不是連續的數字。例如，假設兩列都排名 1，則下一個排名為 3。

RANK [DENSE_RANK 視窗函數](#) 在一個方面與不同：對於 DENSE_RANK，如果兩個或多個資料列綁定，排序值的序列中沒有間隙。例如，假設兩列都排名 1，則下一個排名為 2。

您可以在相同的查詢中使用具有不同 PARTITION BY 和 ORDER BY 子句的排序函數。

語法

```

RANK () OVER
(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list ]
)

```

引數

()

此函數不接受引數，但需要空括號。

OVER

RANK 函數的視窗子句。

PARTITION 依 *expr_list* 排序

選用。一或多個用於定義視窗的表達式。

ORDER BY 訂單_清單

選用。定義排名值所根據的欄。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。如果省略 ORDER BY，則所有資料列的傳回值為 1。

如果 ORDER BY 未產生唯一的排序，資料列的順序是非確定的。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

傳回類型

INTEGER

範例

下列範例會依銷售數量排序資料表 (預設為遞增)，並將排名指派給每一列。最高排名的值為 1。套用範圍函數結果之後排序結果：

```
select salesid, qty,
rank() over (order by qty) as rnk
from winsales
order by 2,1;
```

```
salesid | qty | rnk
-----+-----+-----
10001 | 10 | 1
10006 | 10 | 1
30001 | 10 | 1
40005 | 10 | 1
30003 | 15 | 5
20001 | 20 | 6
20002 | 20 | 6
30004 | 20 | 6
10005 | 30 | 9
30007 | 30 | 9
40001 | 40 | 11
(11 rows)
```

請注意，此範例中的外部 ORDER BY 子句包含資料欄 2 和 1，以確保 Amazon Redshift 在每次執行此查詢時都會持續傳回排序結果。例如，銷售 10001 IDs 和 10006 的資料列具有相同的 QTY 和 RNK 值。依第 1 欄排序最終結果集可確保列 10001 一定位於 10006 之前。如需 WINSALES 資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

在以下範例中，範圍函數反向排序 (order by qty desc)。現在，最高排名值適用於最大值QTY。

```
select salesid, qty,
rank() over (order by qty desc) as rank
from winsales
order by 2,1;
```

salesid	qty	rank
10001	10	8
10006	10	8
30001	10	8
40005	10	8
30003	15	7
20001	20	4
20002	20	4
30004	20	4
10005	30	2
30007	30	2
40001	40	1

(11 rows)

如需WINSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

下列範例會將資料表依 進行分割SELLERID，並依數量（以遞減順序）排序每個分割區，並將排名指派給每一列。套用範圍函數結果之後排序結果。

```
select salesid, sellerid, qty, rank() over
(partition by sellerid
order by qty desc) as rank
from winsales
order by 2,3,1;
```

salesid	sellerid	qty	rank
10001	1	10	2
10006	1	10	2
10005	1	30	1
20001	2	20	1
20002	2	20	1
30001	3	10	4
30003	3	15	3
30004	3	20	2

```

30007 |      3 | 30 | 1
40005 |      4 | 10 | 2
40001 |      4 | 40 | 1
(11 rows)

```

RATIO_TO_REPORT 視窗函數

計算視窗或分割區內值與值總和的比率。使用下列公式決定報告值的比率：

$$\text{value of ratio_expression argument for the current row} / \text{sum of ratio_expression argument for the window or partition}$$

以下資料集示範此公式的使用：

```

Row# Value Calculation RATIO_TO_REPORT
1 2500 (2500)/(13900) 0.1798
2 2600 (2600)/(13900) 0.1870
3 2800 (2800)/(13900) 0.2014
4 2900 (2900)/(13900) 0.2086
5 3100 (3100)/(13900) 0.2230

```

傳回值範圍是 0 至 1 (含)。如果 `ratio_expression` 為 NULL，則傳回值為 NULL。如果 `partition_expression` 中的值是唯一的，則函數將傳回該值。

語法

```

RATIO_TO_REPORT ( ratio_expression )
OVER ( [ PARTITION BY partition_expression ] )

```

引數

ratio_expression

此表達式 (例如欄名) 提供要決定比率的值。表達式必須為數值資料類型，或可隱含地轉換為數值資料類型。

您不能在 `ratio_expression` 中使用其他任何分析函數。

OVER

用於指定視窗分割的子句。OVER 子句不能包含視窗順序或視窗影格規格。

PARTITION BY partition_expression

選用。設定OVER子句中每個群組記錄範圍的表達式。

傳回類型

FLOAT8

範例

下列範例使用 WINSALES 資料表。如需如何建立WINSALES資料表的資訊，請參閱 [範圍函數範例的範例資料表](#)。

下列範例會將 ratio-to-report賣方數量的每一列值計算為所有賣方數量的總數。

```
select sellerid, qty, ratio_to_report(qty)
over()
from winsales
order by sellerid;
```

sellerid	qty	ratio_to_report
1	30	0.13953488372093023
1	10	0.046511627906976744
1	10	0.046511627906976744
2	20	0.09302325581395349
2	20	0.09302325581395349
3	30	0.13953488372093023
3	20	0.09302325581395349
3	15	0.06976744186046512
3	10	0.046511627906976744
4	10	0.046511627906976744
4	40	0.18604651162790697

以下範例依分割區計算每一個賣方的銷售數量比例。

```
select sellerid, qty, ratio_to_report(qty)
over(partition by sellerid)
from winsales;
```

sellerid	qty	ratio_to_report
2	20	0.5

```
2      20      0.5
4      40      0.8
4      10      0.2
1      10      0.2
1      30      0.6
1      10      0.2
3      10      0.13333333333333333
3      15      0.2
3      20      0.26666666666666666
3      30      0.4
```

ROW_NUMBER 視窗函數

根據OVER子句中的 ORDER BY 表達式，在列群組中指派目前列的順序編號，從 1 開始計數。如果存在選用的 PARTITION BY 子句，則會為每個資料列群組重設順序編號。具有相同值的 ORDER BY 表達式資料列會以非確定方式收到不同的資料列編號。

語法

```
ROW_NUMBER() OVER(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list ]
)
```

引數

()

此函數不接受引數，但需要空括號。

OVER

ROW_NUMBER 函數的視窗函數子句。

PARTITION 依 *expr_list* 排序

選用。將結果分割成多組列的一或多個欄運算式。

ORDER BY 訂單_清單

選用。定義集合中資料列順序的一或多個資料行運算式。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

如果 ORDER BY 不會產生唯一的排序或省略，則資料列的順序是非確定性的。如需詳細資訊，請參閱[範圍函數的資料唯一排序](#)。

傳回類型

BIGINT

範例

下列範例使用 WINDSALES 資料表。如需 WINDSALES 資料表的描述，請參閱[範圍函數範例的範例資料表](#)。

下列範例會依 QTY (遞增順序) 排序資料表，然後將資料列編號指派給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, sellerid, qty,
ROW_NUMBER() OVER(
  ORDER BY qty ASC) AS row
FROM winsales
ORDER BY 4,1;
```

salesid	sellerid	qty	row
30001	3	10	1
10001	1	10	2
10006	1	10	3
40005	4	10	4
30003	3	15	5
20001	2	20	6
20002	2	20	7
30004	3	20	8
10005	1	30	9
30007	3	30	10
40001	4	40	11

下列範例會依 分割資料表SELLERID，並依 QTY (以遞增順序) 排序每個分割區，然後將資料列編號指派給每一列。套用範圍函數結果之後排序結果。

```
SELECT salesid, sellerid, qty,
ROW_NUMBER() OVER(
  PARTITION BY sellerid
  ORDER BY qty ASC) AS row_by_seller
FROM winsales
ORDER BY 2,4;
```

salesid	sellerid	qty	row_by_seller
---------	----------	-----	---------------

```

-----+-----+-----+-----
 10001 |      1 | 10 | 1
 10006 |      1 | 10 | 2
 10005 |      1 | 30 | 3
 20001 |      2 | 20 | 1
 20002 |      2 | 20 | 2
 30001 |      3 | 10 | 1
 30003 |      3 | 15 | 2
 30004 |      3 | 20 | 3
 30007 |      3 | 30 | 4
 40005 |      4 | 10 | 1
 40001 |      4 | 40 | 2

```

下列範例顯示不使用選用子句時的結果。

```

SELECT salesid, sellerid, qty, ROW_NUMBER() OVER() AS row
FROM winsales
ORDER BY 4,1;

```

```

salesid  sellerid  qty  row
-----+-----+-----+-----
 30001 |      3 | 10 | 1
 10001 |      1 | 10 | 2
 10005 |      1 | 30 | 3
 40001 |      4 | 40 | 4
 10006 |      1 | 10 | 5
 20001 |      2 | 20 | 6
 40005 |      4 | 10 | 7
 20002 |      2 | 20 | 8
 30003 |      3 | 15 | 9
 30004 |      3 | 20 | 10
 30007 |      3 | 30 | 11

```

STDDEV_SAMP 和 STDDEV_POP 視窗函數

STDDEV_SAMP 和 STDDEV_POP 視窗函數會傳回一組數值（整數、小數或浮點）的範例和母群體標準差。另請參閱[STDDEV_SAMP 和 STDDEV_POP 函數](#)。

STDDEV_SAMP 和 STDDEV 是相同函數的同義詞。

語法

```

STDDEV_SAMP | STDDEV | STDDEV_POP

```

```
( [ ALL ] expression ) OVER  
(  
  [ PARTITION BY expr_list ]  
  [ ORDER BY order_list  
                               frame_clause ]  
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL，函數會保留運算式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION 依 *expr_list* 排序

以一或多個表達式定義函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱[範圍函數語法摘要](#)。

資料類型

STDDEV 函數支援的引數類型為 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL 和 DOUBLE PRECISION。

無論表達式的資料類型為何，STDDEV函數的傳回類型都是雙精度數字。

範例

下列範例示範如何使用 STDDEV_POP 和 VAR_POP 函數作為視窗函數。查詢會計算SALES資料表中 PRICEPAID值的母群體變異數和母群體標準差。

```
select salesid, dateid, pricepaid,
round(stddev_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as stddevpop,
round(var_pop(pricepaid) over
(order by dateid, salesid rows unbounded preceding)) as varpop
from sales
order by 2,1;
```

salesid	dateid	pricepaid	stddevpop	varpop
33095	1827	234.00	0	0
65082	1827	472.00	119	14161
88268	1827	836.00	248	61283
97197	1827	708.00	230	53019
110328	1827	347.00	223	49845
110917	1827	337.00	215	46159
150314	1827	688.00	211	44414
157751	1827	1730.00	447	199679
165890	1827	4192.00	1185	1403323
...				

樣本標準差和變異數函數可如法泡製。

SUM 視窗函數

SUM 視窗函數會傳回輸入資料欄或表達式值的總和。SUM 函數使用數值並忽略NULL值。

語法

```
SUM ( [ ALL ] expression ) OVER
(
  [ PARTITION BY expr_list ]
  [ ORDER BY order_list
                frame_clause ]
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL，函數會保留運算式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION 依 expr_list 排序

以一或多個表達式定義SUM函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由ROWS關鍵字和相關聯的規格符組成。請參閱 [範圍函數語法摘要](#)。

資料類型

SUM 函數支援的引數類型為 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL和 DOUBLE PRECISION。

SUM 函數支援的傳回類型為：

- BIGINT 適用於 SMALLINT或 INTEGER引數
- NUMERIC 適用於BIGINT引數
- DOUBLE PRECISION 適用於浮點引數

範例

下列範例會建立銷售數量的累積 (滾動) 總和，依日期和銷售 ID 排序：

```
select salesid, dateid, sellerid, qty,  
sum(qty) over (order by dateid, salesid rows unbounded preceding) as sum  
from winsales  
order by 2,1;
```

```
salesid | dateid | sellerid | qty | sum  
-----+-----+-----+----+-----
```

```

30001 | 2003-08-02 |      3 | 10 | 10
10001 | 2003-12-24 |      1 | 10 | 20
10005 | 2003-12-24 |      1 | 30 | 50
40001 | 2004-01-09 |      4 | 40 | 90
10006 | 2004-01-18 |      1 | 10 | 100
20001 | 2004-02-12 |      2 | 20 | 120
40005 | 2004-02-12 |      4 | 10 | 130
20002 | 2004-02-16 |      2 | 20 | 150
30003 | 2004-04-18 |      3 | 15 | 165
30004 | 2004-04-18 |      3 | 20 | 185
30007 | 2004-09-07 |      3 | 30 | 215
(11 rows)

```

如需WINSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

以下範例會依日期建立銷售數量的累積 (滾動) 總和、依賣方 ID 分割結果，然後在分割區內依日期和銷售 ID 排序結果：

```

select salesid, dateid, sellerid, qty,
sum(qty) over (partition by sellerid
order by dateid, salesid rows unbounded preceding) as sum
from winsales
order by 2,1;

```

```

salesid | dateid | sellerid | qty | sum
-----+-----+-----+----+----
30001 | 2003-08-02 |      3 | 10 | 10
10001 | 2003-12-24 |      1 | 10 | 10
10005 | 2003-12-24 |      1 | 30 | 40
40001 | 2004-01-09 |      4 | 40 | 40
10006 | 2004-01-18 |      1 | 10 | 50
20001 | 2004-02-12 |      2 | 20 | 20
40005 | 2004-02-12 |      4 | 10 | 50
20002 | 2004-02-16 |      2 | 20 | 40
30003 | 2004-04-18 |      3 | 15 | 25
30004 | 2004-04-18 |      3 | 20 | 45
30007 | 2004-09-07 |      3 | 30 | 75
(11 rows)

```

下列範例會依 SELLERID和 SALESID資料欄順序，依序編號結果集中的所有資料列：

```

select salesid, sellerid, qty,
sum(1) over (order by sellerid, salesid rows unbounded preceding) as rownum

```

```

from winsales
order by 2,1;

salesid | sellerid | qty | rownum
-----+-----+-----+-----
10001 |      1 |  10 |      1
10005 |      1 |  30 |      2
10006 |      1 |  10 |      3
20001 |      2 |  20 |      4
20002 |      2 |  20 |      5
30001 |      3 |  10 |      6
30003 |      3 |  15 |      7
30004 |      3 |  20 |      8
30007 |      3 |  30 |      9
40001 |      4 |  40 |     10
40005 |      4 |  10 |     11
(11 rows)

```

如需WINDSALES資料表的說明，請參閱 [範圍函數範例的範例資料表](#)。

下列範例會依序編號結果集中的所有資料列、依分割結果SELLERID，並依 SELLERID和 SALESID 排序結果：

```

select salesid, sellerid, qty,
sum(1) over (partition by sellerid
order by sellerid, salesid rows unbounded preceding) as rownum
from winsales
order by 2,1;

salesid | sellerid | qty | rownum
-----+-----+-----+-----
10001 |      1 |  10 |      1
10005 |      1 |  30 |      2
10006 |      1 |  10 |      3
20001 |      2 |  20 |      1
20002 |      2 |  20 |      2
30001 |      3 |  10 |      1
30003 |      3 |  15 |      2
30004 |      3 |  20 |      3
30007 |      3 |  30 |      4
40001 |      4 |  40 |      1
40005 |      4 |  10 |      2
(11 rows)

```

VAR_SAMP 和 VAR_POP 視窗函數

VAR_SAMP 和 VAR_POP 視窗函數會傳回一組數值（整數、小數點或浮點）的範例和總體差異。另請參閱[VAR_SAMP 和 VAR_POP 函數](#)。

VAR_SAMP 和 VARIANCE 是相同函數的同義詞。

語法

```
VAR_SAMP | VARIANCE | VAR_POP  
( [ ALL ] expression ) OVER  
(  
  [ PARTITION BY expr_list ]  
  [ ORDER BY order_list  
                                frame_clause ]  
)
```

引數

expression

函數運算的目標欄或表達式。

ALL

使用引數 ALL，函數會保留運算式中的所有重複值。ALL 為預設值。不支援 DISTINCT。

OVER

指定彙總函數的視窗子句。OVER 子句會將視窗彙總函數與一般設定彙總函數區分開來。

PARTITION 依 *expr_list* 排序

以一或多個表達式定義函數的視窗。

ORDER BY 訂單_清單

排序每一個分割區內的列。如果未指定 PARTITION BY，ORDERBY 會使用整個資料表。

frame_clause

如果 ORDER BY 子句用於彙總函數，則需要明確的影格子句。窗框子句在排序的結果內包含或排除列集，以調整函數視窗中的一個列集。框架子句由 ROWS 關鍵字和相關聯的規格符組成。請參閱[範圍函數語法摘要](#)。

資料類型

VARIANCE 函數支援的引數類型為 SMALLINT、INTEGER、BIGINT、NUMERIC、DECIMAL、REAL 和 DOUBLE PRECISION。

無論表達式的資料類型為何，VARIANCE 函數的傳回類型都是雙精度數字。

系統管理函數

主題

- [CHANGE_QUERY_PRIORITY](#)
- [CHANGE_SESSION_PRIORITY](#)
- [CHANGE_USER_PRIORITY](#)
- [CURRENT_SETTING](#)
- [PG_CANCEL_BACKEND](#)
- [PG_TERMINATE_BACKEND](#)
- [REBOOT_CLUSTER](#)
- [SET_CONFIG](#)

Amazon Redshift 支援多個系統管理函數。

CHANGE_QUERY_PRIORITY

CHANGE_QUERY_PRIORITY 可讓超級使用者修改正在執行或在工作負載管理中等待的查詢的優先順序 (WLM)。

此函數讓超級使用者可以立即變更系統中任何查詢的優先順序。只有一個查詢、使用者或工作階段可以使用優先順序 CRITICAL 執行。

語法

```
CHANGE_QUERY_PRIORITY(query_id, priority)
```

引數

query_id

要變更其優先順序之查詢的查詢識別碼。需要 INTEGER 值。

priority

要指派給查詢的新優先順序。這個引數必須是具備 CRITICAL、HIGHEST、HIGH、NORMAL、LOW 或 LOWEST 值的字串。

傳回類型

無

範例

若要在 STV_WLM_QUERY_STATE 系統資料表 query_priority 中顯示資料欄，請使用下列範例。

```
SELECT query, service_class, query_priority, state
FROM stv_wlm_query_state WHERE service_class = 101;
```

```
+-----+-----+-----+-----+
| query | service_class | query_priority | state |
+-----+-----+-----+-----+
| 1076 | 101 | Lowest | Running |
| 1075 | 101 | Lowest | Running |
+-----+-----+-----+-----+
```

若要顯示超級使用者執行函數 change_query_priority 以將優先順序變更為 CRITICAL 的結果，請使用下列範例。

```
SELECT CHANGE_QUERY_PRIORITY(1076, 'Critical');
```

```
+-----+
| change_query_priority |
+-----+
| Succeeded to change query priority. Priority changed from Lowest to Critical. |
+-----+
```

CHANGE_SESSION_PRIORITY

CHANGE_SESSION_PRIORITY 可讓超級使用者立即變更系統中任何工作階段的優先順序。只有一個工作階段、使用者或查詢可以使用優先順序 CRITICAL 執行。

語法

```
CHANGE_SESSION_PRIORITY(pid, priority)
```

引數

pid

要變更其優先順序之工作階段的程序識別碼。值 -1 表示目前工作階段。需要 INTEGER 值。

priority

要指派給工作階段的新優先順序。這個引數必須是具備 CRITICAL、HIGHEST、HIGH、NORMAL、LOW 或 LOWEST 值的字串。

傳回類型

無

範例

若要傳回處理目前工作階段之伺服器處理程序的程序識別碼，請使用下列範例。

```
SELECT pg_backend_pid();
```

```
+-----+
| pg_backend_pid |
+-----+
|           30311 |
+-----+
```

在此範例中，將目前工作階段的優先順序變更為 LOWEST。

```
SELECT CHANGE_SESSION_PRIORITY(30311, 'Lowest');
```

```
+-----+
+
|
|           change_session_priority
+-----+
+
```

```
| Succeeded to change session priority. Changed session (pid:30311) priority to lowest.
|
+-----+
+
```

在此範例中，將目前工作階段的優先順序變更為 HIGH。

```
SELECT CHANGE_SESSION_PRIORITY(-1, 'High');

+-----+
+
|          change_session_priority
|
+-----+
+
| Succeeded to change session priority. Changed session (pid:30311) priority from
| lowest to high. |
+-----+
+
```

若要建立變更工作階段優先順序的預存程序，請使用下列範例。將執行此預存程序的許可授予給資料庫使用者 `test_user`。

```
CREATE OR REPLACE PROCEDURE sp_priority_low(pid IN int, result OUT varchar)
AS $$
BEGIN
    SELECT CHANGE_SESSION_PRIORITY(pid, 'low') into result;
END;
$$ LANGUAGE plpgsql
SECURITY DEFINER;
GRANT EXECUTE ON PROCEDURE sp_priority_low(int) TO test_user;
```

接著名為 `test_user` 的資料庫使用者呼叫程序。

```
CALL sp_priority_low(pg_backend_pid());

+-----+
|          result          |
+-----+
| Success. Change session (pid:13155) priority to low. |
+-----+
```

CHANGE_USER_PRIORITY

CHANGE_USER_PRIORITY 可讓超級使用者修改正在執行或在工作負載管理 () 中等待的使用者發出的所有查詢的優先順序WLM。只有一個使用者、工作階段或查詢可以使用優先順序 CRITICAL 執行。

語法

```
CHANGE_USER_PRIORITY(user_name, priority)
```

引數

user_name

要變更其查詢優先順序的資料庫使用者名稱。

priority

要指派給由 *user_name* 發出之所有查詢的新優先順序。這個引數必須是具備 CRITICAL、HIGHEST、HIGH、NORMAL、LOW、LOWEST 或 RESET 值的字串。只有超級使用者才能將優先順序變更為 CRITICAL。將優先順序變更為 RESET，會移除 *user_name* 的優先順序設定。

傳回類型

無

範例

若要將使用者 `analysis_user` 的優先順序變更為 LOWEST，請使用下列範例。

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'lowest');
```

```
+-----+
|               change_user_priority               |
+-----+
| Succeeded to change user priority. Changed user (analysis_user) priority to lowest. |
+-----+
```

若要將優先順序變更為 LOW，請使用下列範例。

```
SELECT CHANGE_USER_PRIORITY('analysis_user', 'low');
```

```

+-----+
+
|           change_user_priority
|
+-----+
+
| Succeeded to change user priority. Changed user (analysis_user) priority from Lowest
  to low. |
+-----+
+

```

若要重設優先順序，請使用下列範例。

```

SELECT CHANGE_USER_PRIORITY('analysis_user', 'reset');

+-----+
|           change_user_priority           |
+-----+
| Succeeded to reset priority for user (analysis_user). |
+-----+

```

CURRENT_SETTING

`CURRENT_SETTING` 傳回指定組態參數的目前值。

此函數相當於 [SHOW](#) 命令。

語法

```
current_setting('parameter')
```

以下陳述式會傳回指定工作階段內容變數的目前值。

```
current_setting('variable_name')
current_setting('variable_name'[, error_if_undefined])
```

引數

parameter

要顯示的參數值。如需組態參數的清單，請參閱[組態參考](#)

variable_name

要顯示的變數名稱。這必須是工作階段內容變數的字串常數。

error_if_undefined

(選用) Boolean 值，指定變數名稱不存在時的行為。當 `error_if_undefined` 設定為 `TRUE` (這是預設設定)，Amazon Redshift 擲回錯誤。當 `error_if_undefined` 設定為 `FALSE`，Amazon Redshift 傳回 `NULL`。Amazon Redshift 僅支援工作階段內容變數的 `error_if_undefined` 參數。當輸入是組態參數時，不能使用此選項。

傳回類型

傳回 `CHAR` 或 `VARCHAR` 字串。

範例

若要傳回 `query_group` 參數的目前設定，請使用下列範例。

```
SELECT CURRENT_SETTING('query_group');
```

```
+-----+
| current_setting |
+-----+
| unset          |
+-----+
```

若要傳回變數 `app_context.user_id` 的目前設定，請使用下列範例。

```
SELECT CURRENT_SETTING('app_context.user_id', FALSE);
```

PG_CANCEL_BACKEND

取消查詢。PG_CANCEL_ 在BACKEND功能上等同於 [CANCEL](#) 命令。您可以取消使用者目前正在執行的查詢。超級使用者可以取消任何查詢。

語法

```
pg_cancel_backend( pid )
```

引數

pid

要取消之查詢的程序 ID (PID)。您無法藉由指定查詢 ID 來取消查詢；您必須指定查詢的處理程序 ID。需要 INTEGER 值。

傳回類型

無

使用須知

如果有多個工作階段中的查詢在同一資料表上保持鎖定，您可以使用 [PG_TERMINATE_BACKEND](#) 函數來終止其中一個工作階段，這樣可強制終止工作階段中任何目前正在執行的交易，以解除所有鎖定並恢復交易。查詢 PG_LOCKS 目錄資料表以檢視目前保留的鎖定。如果您因為查詢位於交易區塊 (BEGIN ... END) 而無法取消查詢，您可以使用 PG_TERMINATE_BACKEND 函數終止執行查詢的工作階段。

範例

若要取消目前執行中的查詢，請先擷取您要取消之查詢的處理程序 ID。若要判斷IDs所有目前執行中查詢的程序，請執行下列命令。

```
SELECT pid, TRIM(starttime) AS start,
duration, TRIM(user_name) AS user,
SUBSTRING(query,1,40) AS querytxt
FROM stv_recents
WHERE status = 'Running';
```

pid	starttime	duration	user	querytxt
802	2013-10-14 09:19:03.55	132	dwuser	select venue name from venue
834	2013-10-14 08:33:49.47	1250414	dwuser	select * from listing;
964	2013-10-14 08:30:43.29	326179	dwuser	select sellerid from sales

若要取消處理程序識別碼為 802 的查詢，請使用下列範例。

```
SELECT PG_CANCEL_BACKEND(802);
```


PG_TERMINATE_BACKEND

終止工作階段。您可以終止使用者所擁有的工作階段。超級使用者可以終止任何工作階段。

語法

```
pg_terminate_backend( pid )
```

引數

pid

要終止之工作階段的處理程序 ID。需要 INTEGER 值。

傳回類型

無

使用須知

如果您即將達到並行連線的限制，請使用 PG_TERMINATE_BACKEND 終止閒置工作階段並釋放連線。如需詳細資訊，請參閱 [Amazon Redshift 限制](#)。

如果多個工作階段中的查詢在相同資料表上保留鎖定，您可以使用 PG_TERMINATE_BACKEND 來終止其中一個工作階段，這會強制終止工作階段中任何目前執行的交易釋放所有鎖定並復原交易。查詢 PG_LOCKS 目錄資料表以檢視目前保留的鎖定。

如果查詢不在交易區塊中（BEGIN ... END），您可以使用 [CANCEL](#) 命令或 [PG_CANCEL_BACKEND](#) 函數取消查詢。

範例

若要查詢 SVV_TRANSACTIONS 資料表以檢視目前交易的所有有效鎖定，請使用下列範例。

```
SELECT * FROM svv_transactions;
```

```
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+
| txn_owner | txn_db |  xid  | pid  |      txn_start      | lock_mode  |
| lockable_object_type | relation | granted |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```

| rsuser      | dev      | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
|             |          | 51940 | true  |                      |                  |
| rsuser      | dev      | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
|             |          | 52000 | true  |                      |                  |
| rsuser      | dev      | 96178 | 8585 | 2017-04-12 20:13:07 | AccessShareLock | relation
|             |          | 108623 | true  |                      |                  |
| rsuser      | dev      | 96178 | 8585 | 2017-04-12 20:13:07 | ExclusiveLock   |
transactionid |          |       | true  |                      |                  |
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

若要終止保持鎖定的工作階段，請使用下列範例。

```
SELECT PG_TERMINATE_BACKEND(8585);
```

REBOOT_CLUSTER

重新啟動 Amazon Redshift 叢集，而不關閉與叢集的連線。您必須是資料庫超級使用者才能執行此命令。

完成此軟重新開機之後，Amazon Redshift 叢集會將錯誤傳回給使用者應用程式，並要求使用者應用程式重新提交任何因軟重新開機而中斷的交易或查詢。

語法

```
SELECT REBOOT_CLUSTER();
```

SET_CONFIG

將組態參數設為新的設定。

此函數等同於 中的 SET 命令SQL。

語法

```
SET_CONFIG('parameter', 'new_value' , is_local)
```

下列陳述式會將工作階段內容變數設定為新設定。

```
set_config('variable_name', 'new_value' , is_local)
```

引數

parameter

要設定的參數。

variable_name

要設定的變數名稱。

new_value

參數的新值。

is_local

若為 true，參數值僅套用至目前交易。有效值為 true 或 1 及 false 或 0。

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

若要僅針對目前交易將 query_group 參數的值設為 test，請使用下列範例。

```
SELECT SET_CONFIG('query_group', 'test', true);
```

```
+-----+
| set_config |
+-----+
| test      |
+-----+
```

若要設定工作階段內容變數，請使用下列範例。

```
SELECT SET_CONFIG('app.username', 'cuddy', FALSE);
```

系統資訊函數

Amazon Redshift 支援許多系統資訊函數。

主題

- [CURRENT_AWS_ACCOUNT](#)

- [CURRENT_DATABASE](#)
- [CURRENT_NAMESPACE](#)
- [CURRENT_SCHEMA](#)
- [CURRENT_SCHEMAS](#)
- [CURRENT_SESSION_ARN](#)
- [CURRENT_USER](#)
- [CURRENT_USER_ID](#)
- [DEFAULT_IAM_ROLE](#)
- [GET_MOUNTED_ROLE](#)
- [HAS_ASSUMEROLE_PRIVILEGE](#)
- [HAS_DATABASE_PRIVILEGE](#)
- [HAS_SCHEMA_PRIVILEGE](#)
- [HAS_TABLE_PRIVILEGE](#)
- [LAST_USER_QUERY_ID](#)
- [PG_BACKEND_PID](#)
- [PG_GET_COLS](#)
- [PG_GETGRANTEE_BY_IAM_ROLE](#)
- [PG_GET_IAMROLE_BY_USER](#)
- [PG_GET_LATE_BINDING_VIEW_COLS](#)
- [PG_GET_SESSION_ROLES](#)
- [PG_LAST_COPY_COUNT](#)
- [PG_LAST_COPY_ID](#)
- [PG_LAST_UNLOAD_ID](#)
- [PG_LAST_QUERY_ID](#)
- [PG_LAST_UNLOAD_COUNT](#)
- [SLICE_NUM](#) 函數
- [USER](#)
- [ROLE_ISMEMBER_OF](#)
- [USER_ISMEMBER_OF](#)
- [VERSION](#)

CURRENT_AWS_ACCOUNT

傳回與提交查詢之 Amazon Redshift 叢集相關聯的 AWS 帳戶。

語法

```
current_aws_account
```

傳回類型

傳回整數。

範例

下列查詢傳回目前資料庫的名稱。

```
select user, current_aws_account;
current_user | current_account
-----+-----
dwuser      | 987654321

(1 row)
```

CURRENT_DATABASE

傳回您目前連接的資料庫名稱。

語法

```
current_database()
```

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢傳回目前資料庫的名稱。

```
select current_database();

current_database
-----
```

```
ticket  
(1 row)
```

CURRENT_NAMESPACE

傳回目前 Amazon Redshift 叢集的叢集命名空間。Amazon Redshift 叢集命名空間是 Amazon Redshift 叢集的唯一 ID。

語法

```
current_namespace
```

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列查詢傳回目前命名空間的名稱。

```
select user, current_namespace;  
current_user | current_namespace  
-----+-----  
dwuser      | 86b5169f-01dc-4a6f-9fbb-e2e24359e9a8  
  
(1 row)
```

CURRENT_SCHEMA

傳回搜尋路徑前面的結構描述名稱。此結構描述將用於建立時未指定目標結構描述的任何資料表或其他具名物件。

語法

Note

這是領導者節點函數。如果此函數參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視，則傳回錯誤。

```
current_schema()
```

傳回類型

CURRENT_SCHEMA 傳回 CHAR或 VARCHAR字串。

範例

下列查詢傳回目前結構描述：

```
select current_schema();

current_schema
-----
public
(1 row)
```

CURRENT_SCHEMAS

傳回目前搜尋路徑中任何結構描述的名稱所構成的陣列。目前搜尋路徑是在 `search_path` 參數中定義。

語法

Note

這是領導者節點函數。如果此函數參考使用者建立的資料表、STL或 STV 系統資料表，或 SVV或 SVL系統檢視，則傳回錯誤。

```
current_schemas(include_implicit)
```

引數

include_implicit

若為 `true`，表示搜尋路徑應該包含任何隱含包含的系統結構描述。有效值為 `true` 和 `false`。通常，若為 `true`，此參數除了傳回目前的結構描述外，還會傳回 `pg_catalog` 結構描述。

傳回類型

傳回 CHAR或 VARCHAR字串。

範例

下列範例傳回目前搜尋路徑中的結構描述名稱，但不包括隱含包含的系統結構描述：

```
select current_schemas(false);

current_schemas
-----
{public}
(1 row)
```

下列範例傳回目前搜尋路徑中的結構描述名稱，包括隱含包含的系統結構描述：

```
select current_schemas(true);

current_schemas
-----
{pg_catalog,public}
(1 row)
```

CURRENT_SESSION_ARN

傳回目前授權ARN全域使用者的。全域使用者在 Redshift 帳戶、叢集和 Serverless 工作群組之間具有相同的身分。全域使用者透過 IAM Identity Center 或透過 IAM型工作階段身分驗證登入。Data lake 使用者是全域 AWS 使用者。

此函數通常用於使用多方視 AWS Glue 觀表的內容。如需使用 IAM Identity Center 和 Redshift 進行身分管理的詳細資訊，請參閱將 [Redshift 與 IAM Identity Center 連線，為使用者提供單一登入體驗](#)。如需多方瀏覽 Glue 檢視的詳細資訊，請在 [AWS Glue Data Catalog 中建立檢視](#)。

語法

```
current_session_arn()
```

傳回類型

傳回全域驗證使用者的VARCHAR字串或 null 值。

使用須知

不支援本機使用者，並導致 null 回應。

範例

下列查詢會傳回目前工作階段 的名稱ARN：

```
SELECT current_session_arn();

current_session_arn
-----
arn:aws:iam::123456789012:user/user
(1 row)
```

CURRENT_USER

傳回資料庫的目前「有效」使用者的使用者名稱，適用於檢查權限。通常，此使用者名稱與工作階段使用者相同；不過，超級使用者偶爾會改變這種情況。

Note

呼叫 CURRENT_ 時，請勿使用結尾括號USER。

語法

```
current_user
```

傳回類型

CURRENT_USER 傳回NAME資料類型，並可以轉換為 CHAR或 VARCHAR字串。

使用須知

如果使用 CREATE_PROCEDURE 命令SECURITYDEFINER的選項建立預存程序，則在預存程序中叫用 CURRENT_USER 函數時，Amazon Redshift 會傳回預存程序擁有者的使用者名稱。

範例

下列查詢傳回目前資料庫使用者的名稱：

```
select current_user;
```

```
current_user
-----
dwuser
(1 row)
```

CURRENT_USER_ID

傳回登入目前工作階段之 Amazon Redshift 使用者的唯一識別碼。

語法

```
CURRENT_USER_ID
```

傳回類型

CURRENT_USER_ID 函數會傳回整數。

範例

下列範例傳回此工作階段的使用者名稱和目前使用者 ID：

```
select user, current_user_id;

current_user | current_user_id
-----+-----
dwuser      |                1
(1 row)
```

DEFAULT_IAM_ROLE

傳回目前與 Amazon Redshift 叢集相關聯的預設IAM角色。如果沒有相關聯的任何預設IAM角色，則函數不會傳回任何。

語法

```
select default_iam_role();
```

傳回類型

傳回VARCHAR字串。

範例

下列範例會傳回目前與指定 Amazon Redshift 叢集相關聯的預設IAM角色，

```
select default_iam_role();
           default_iam_role
-----
arn:aws:iam::123456789012:role/myRedshiftRole
(1 row)
```

GET_MOUNTED_ROLE

當作為多方位 AWS Glue 檢視的一部分調用時，它允許傳回用於掛載 Lake Formation 結構描述或資料庫IAM的角色。多方程表示 SQL 支援多個查詢引擎，例如 Amazon EMR和 Redshift。如需多向 Glue 檢視的詳細資訊，請參閱 [AWS Glue Data Catalog 中的建立檢視](#)。

語法

```
get_mounted_role()
```

傳回類型

傳回VARCHAR字串或 null 值。

使用須知

此函數會針對外部 Lake Formation 檢視以外的任何使用案例傳回 null。

範例

下列查詢會傳回用來掛載 Lake Formation 資源的身分。

```
CREATE EXTERNAL PROTECTED VIEW external_schema.remote_view AS
SELECT mycol, get_mounted_role() FROM external_schema.remote_table;

mycol | get_mounted_role
-----
1      arn:aws:iam::123456789012:role/salesrole
(1 row)
```

HAS_ASSUMEROLE_PRIVILEGE

如果指定的使用者具有執行指定命令IAM的許可，則傳回布林值 `true (t)`。如果使用者沒有具有執行指定命令之權限的指定IAM角色，則函數會傳回 `false (f)`。如需權限的相關資訊，請參閱 [GRANT](#)。

語法

```
has_assumerole_privilege( [ user, ] iam_role_arn, cmd_type)
```

引數

使用者

要檢查IAM角色權限的使用者名稱。預設值是檢查目前使用者。超級使用者和使用者可以使用此函數。不過，使用者只能檢視自己的權限。

iam_role_arn

已授予命令權限IAM的角色。

cmd_type

已授與存取權的命令。有效值如下：

- COPY
- UNLOAD
- EXTERNAL FUNCTION
- CREATE MODEL

傳回類型

BOOLEAN

範例

下列查詢確認使用者 `reg_user1` 具有執行 COPY 命令之 Redshift-S3-Read 角色的許可。

```
select has_assumerole_privilege('reg_user1', 'arn:aws:iam::123456789012:role/Redshift-S3-Read', 'copy');
```

```
has_assumerole_privilege
-----
true
(1 row)
```

HAS_DATABASE_PRIVILEGE

如果使用者具有指定之資料庫的指定權限，則傳回 true。如需權限的相關資訊，請參閱 [GRANT](#)。

語法

Note

這是領導者節點函數。如果此函數參考使用者建立的資料表、STL或STV系統資料表，或SVV或SVL系統檢視，則會傳回錯誤。

```
has_database_privilege( [ user, ] database, privilege)
```

引數

使用者

要檢查資料庫權限的使用者名稱。預設值是檢查目前使用者。

資料庫

與權限相關聯的資料庫。

privilege

要檢查的權限。有效值如下：

- CREATE
- TEMPORARY
- TEMP

傳回類型

傳回 CHAR或 VARCHAR字串。

範例

下列查詢確認GUEST使用者在TICKIT資料庫中具有 TEMP 權限。

```
select has_database_privilege('guest', 'ticket', 'temp');

has_database_privilege
-----
true
(1 row)
```

HAS_SCHEMA_PRIVILEGE

如果使用者具有指定之結構描述的指定權限，則傳回 true。如需權限的相關資訊，請參閱 [GRANT](#)。

語法

Note

這是領導者節點函數。如果此函數參考使用者建立的資料表、STL或STV系統資料表，或SVV或SVL系統檢視，則會傳回錯誤。

```
has_schema_privilege( [ user, ] schema, privilege)
```

引數

使用者

要檢查結構描述權限的使用者名稱。預設值是檢查目前使用者。

結構描述

與權限相關聯的結構描述。

privilege

要檢查的權限。有效值如下：

- CREATE
- USAGE

傳回類型

傳回 CHAR或 VARCHAR字串。

範例

下列查詢確認GUEST使用者在PUBLIC結構描述上具有 CREATE權限：

```
select has_schema_privilege('guest', 'public', 'create');

has_schema_privilege
-----
true
(1 row)
```

HAS_TABLE_PRIVILEGE

如果使用者具有指定之資料表的指定權限，則傳回 true，否則傳回 false。

語法

Note

這是領導者節點函數。如果此函數參考使用者建立的資料表、STL或 STV 系統資料表，或 SVV或 SVL系統檢視，則會傳回錯誤。如需權限的相關資訊，請參閱 [GRANT](#)。

```
has_table_privilege( [ user, ] table, privilege)
```

引數

使用者

要檢查資料表權限的使用者名稱。預設值是檢查目前使用者。

表格

與權限相關聯的資料表。

privilege

要檢查的權限。有效值如下：

- SELECT
- INSERT
- UPDATE
- DELETE
- DROP
- REFERENCES

傳回類型

BOOLEAN

範例

下列查詢發現GUEST使用者在LISTING資料表上沒有SELECT權限。

```
select has_table_privilege('guest', 'listing', 'select');

has_table_privilege
-----
false
```

下列查詢會使用 `pg_tables` 和 `pg_user` 目錄資料表的輸出，列出資料表權限，包括選取、插入、更新和刪除。這只是範例。您可能必須從資料庫中指定結構描述名稱和資料表名稱。如需詳細資訊，請參閱[查詢目錄資料表](#)。

```
SELECT
  tablename
  ,username
  ,HAS_TABLE_PRIVILEGE(users.username, tablename, 'select') AS sel
  ,HAS_TABLE_PRIVILEGE(users.username, tablename, 'insert') AS ins
  ,HAS_TABLE_PRIVILEGE(users.username, tablename, 'update') AS upd
  ,HAS_TABLE_PRIVILEGE(users.username, tablename, 'delete') AS del
FROM
  (SELECT * from pg_tables
  WHERE schemaname = 'public' and tablename in ('event','listing')) as tables
  ,(SELECT * FROM pg_user) AS users;
```

tablename	username	sel	ins	upd	del
event	john	true	true	true	true


```

event      | sally   | false | false | false | false
event      | elsa    | false | false | false | false
listing    | john    | true  | true  | true  | true
listing    | sally   | false | false | false | false
listing    | elsa    | false | false | false | false

```

先前的查詢也包含交叉聯結。如需詳細資訊，請參閱[JOIN 範例](#)。若要查詢不在public結構描述中的資料表，請從WHERE子句中移除schemaname條件，並在查詢之前使用下列範例。

```
SET SEARCH_PATH to 'schema_name';
```

LAST_USER_QUERY_ID

傳回目前工作階段中最近完成的使用者查詢的查詢 ID。如果目前工作階段中未執行任何查詢，last_user_query_id 會傳回 -1。此函數不會傳回僅在引線節點上執行之查詢的查詢 ID。如需詳細資訊，請參閱[僅限領導節點函數](#)。

語法

```
last_user_query_id()
```

傳回類型

傳回整數。

範例

下列查詢傳回在目前工作階段中完成的使用者最後一個執行查詢的 ID。

```
select last_user_query_id();
```

結果如下。

```

last_user_query_id
-----
          5437
(1 row)

```

下列查詢傳回使用者在目前工作階段中執行的最近完成之查詢的查詢 ID 和文字。

```
select query_id, query_text from sys_query_history where query_id =
last_user_query_id();
```

結果如下。

```
query_id, query_text
-----
+-----+-----+-----+-----+-----+-----+
5556975 | select last_user_query_id() limit 100 --RequestID=<unique request ID>;
TraceID=<unique trace ID>
```

PG_BACKEND_PID

傳回處理目前工作階段之伺服器程序的程序 ID (PID)。

Note

PID 並非全域唯一。可隨著時間而重複使用。

語法

```
pg_backend_pid()
```

傳回類型

傳回整數。

範例

您可以建立 PG_BACKEND_PID 與日誌資料表的關聯，以擷取目前工作階段的資訊。例如，下列查詢會針對目前工作階段中完成的查詢傳回其查詢 ID 和一部分的查詢文字。

```
select query, substring(text,1,40)
from stl_querytext
where pid = PG_BACKEND_PID()
order by query desc;
```

```
query |          substring
-----+-----
```

```
14831 | select query, substring(text,1,40) from
14827 | select query, substring(path,0,80) as pa
14826 | copy category from 's3://dw-tickit/manif
14825 | Count rows in target table
14824 | unload ('select * from category') to 's3
(5 rows)
```

您可以將 PG_BACKEND_PID_ 與下列日誌資料表中的 pid 資料欄建立關聯（括號中註明例外狀況）：

- [STL_CONNECTION_LOG](#)
- [STL_DDLTEXT](#)
- [STL_ERROR](#)
- [STL_QUERY](#)
- [STL_QUERYTEXT](#)
- [STL_SESSIONS](#) (process)
- [STL_TR_CONFLICT](#)
- [STL_UTILITYTEXT](#)
- [STV_ACTIVE_CURSORS](#)
- [STV_INFLIGHT](#)
- [STV_LOCKS](#) (lock_owner_pid)
- [STV_RECENTS](#) (process_id)

PG_GET_COLS

傳回資料表或檢視定義的欄中繼資料。

語法

```
pg_get_cols('name')
```

引數

name

Amazon Redshift 資料表或檢視的名稱。如需詳細資訊，請參閱[名稱與識別碼](#)。

傳回類型

VARCHAR

使用須知

PG_GET_COLS 函數會傳回資料表或檢視定義中每一欄的一列。列包含結構描述名稱、關係名稱、欄名稱、資料類型及欄號的逗號分隔清單。的結果格式SQL取決於使用的SQL用戶端。

範例

下列範例會傳回在 結構描述SALES_VW中名為 的檢視結果，public以及由使用者在連線的資料庫中建立myticket1的結構描述sales中名為 的資料表結果dev。

下列範例會傳回名為 之檢視的資料欄中繼資料SALES_VW。

```
select pg_get_cols('sales_vw');
```

```
pg_get_cols
```

```
-----
(public,sales_vw,salesid,integer,1)
(public,sales_vw,listid,integer,2)
(public,sales_vw,sellerid,integer,3)
(public,sales_vw,buyerid,integer,4)
(public,sales_vw,eventid,integer,5)
(public,sales_vw,dateid,smallint,6)
(public,sales_vw,qtysold,smallint,7)
(public,sales_vw,pricepaid,"numeric(8,2)",8)
(public,sales_vw,commission,"numeric(8,2)",9)
(public,sales_vw,saletime,"timestamp without time zone",10)
```

下列範例會以資料表格式傳回SALES_VW檢視的資料欄中繼資料。

```
select * from pg_get_cols('sales_vw')
cols(view_schema name, view_name name, col_name name, col_type varchar, col_num int);
```

view_schema	view_name	col_name	col_type	col_num
public	sales_vw	salesid	integer	1
public	sales_vw	listid	integer	2
public	sales_vw	sellerid	integer	3
public	sales_vw	buyerid	integer	4
public	sales_vw	eventid	integer	5

public	sales_vw	dateid	smallint	6
public	sales_vw	qtysold	smallint	7
public	sales_vw	pricepaid	numeric(8,2)	8
public	sales_vw	commission	numeric(8,2)	9
public	sales_vw	saletime	timestamp without time zone	10

下列範例會以SALES資料表格式myticket1傳回結構描述中資料表的資料欄中繼資料。

```
select * from pg_get_cols('"myticket1"."sales"')
cols(view_schema name, view_name name, col_name name, col_type varchar, col_num int);
```

view_schema	view_name	col_name	col_type	col_num
myticket1	sales	salesid	integer	1
myticket1	sales	listid	integer	2
myticket1	sales	sellerid	integer	3
myticket1	sales	buyerid	integer	4
myticket1	sales	eventid	integer	5
myticket1	sales	dateid	smallint	6
myticket1	sales	qtysold	smallint	7
myticket1	sales	pricepaid	numeric(8,2)	8
myticket1	sales	commission	numeric(8,2)	9
myticket1	sales	saletime	timestamp without time zone	10

PG_GETGRANTEE_BY_IAM_ROLE

傳回授予指定IAM角色的所有使用者和群組。

語法

```
pg_get_grantee_by_iam_role('iam_role_arn')
```

引數

iam_role_arn

傳回已授予此IAM角色之使用者和群組的角色。

傳回類型

VARCHAR

使用須知

PG_GET_GRANTEE_BY_IAM_ROLE函數會為每個使用者或群組傳回一列。每一列都包含承授者名稱、承授者類型和授與的權限。承授者類型的可能值為 p (代表公用)、u (代表使用者) 和 g (代表群組)。

您必須是超級使用者才能使用此函數。

範例

下列範例表示IAM角色Redshift-S3-Write已授予 group1和 reg_user1。中的使用者只能為COPY操作group_1指定角色，而使用者只能reg_user1指定角色以執行UNLOAD操作。

```
select pg_get_grantee_by_iam_role('arn:aws:iam::123456789012:role/Redshift-S3-Write');
```

```
pg_get_grantee_by_iam_role
-----
(group_1,g,COPY)
(reg_user1,u,UNLOAD)
```

下列 PG_GET_GRANTEE_BY_IAM_ROLE函數範例會將結果格式化為資料表。

```
select grantee, grantee_type, cmd_type FROM
pg_get_grantee_by_iam_role('arn:aws:iam::123456789012:role/Redshift-S3-Write')
res_grantee(grantee text, grantee_type text, cmd_type text) ORDER BY 1,2,3;
```

```
grantee | grantee_type | cmd_type
-----+-----+-----
group_1 | g             | COPY
reg_user1 | u            | UNLOAD
```

PG_GET_IAMROLE_BY_USER

傳回授予使用者的所有IAM角色和命令權限。

語法

```
pg_get_iam_role_by_user('name')
```

引數

name

要傳回IAM角色的使用者名稱。

傳回類型

VARCHAR

使用須知

PG_GET_IAM_ROLE_BY_USER 函數會為每個角色集和命令權限傳回一列。資料列包含逗號分隔清單，其中包含使用者名稱、IAM角色和命令。

結果中的 default 值表示使用者可以指定任何可用的角色來執行顯示的命令。

您必須是超級使用者才能使用此函數。

範例

下列範例表示使用者可以reg_user1指定任何可用的IAM角色來執行COPY操作。使用者也可以指定UNLOAD操作Redshift-S3-Write的角色。

```
select pg_get_iam_role_by_user('reg_user1');
```

```
pg_get_iam_role_by_user
```

```
(reg_user1,default,COPY)
(reg_user1,arn:aws:iam::123456789012:role/Redshift-S3-Write,COPY|UNLOAD)
```

下列 PG_GET_IAM_ROLE_BY_USER 函數範例會將結果格式化為資料表。

```
select username, iam_role, cmd FROM pg_get_iam_role_by_user('reg_user1')
res_iam_role(username text, iam_role text, cmd text);
```

username	iam_role	cmd
reg_user1	default	None

```
reg_user1 | arn:aws:iam::123456789012:role/Redshift-S3-Read | COPY
```

PG_GET_LATE_BINDING_VIEW_COLS

傳回資料庫中所有近期繫結檢視的欄中繼資料。如需詳細資訊，請參閱 [近期繫結檢視](#)

語法

```
pg_get_late_binding_view_cols()
```

傳回類型

VARCHAR

使用須知

PG_GET_LATE_BINDING_VIEW_COLS 函數針對近期繫結檢視中的每一欄，各傳回一行。列包含結構描述名稱、關係名稱、欄名稱、資料類型及欄號的逗號分隔清單。

範例

下列範例傳回所有近期繫結檢視的欄中繼資料。

```
select pg_get_late_binding_view_cols();

pg_get_late_binding_view_cols
-----
(public,myevent,eventname,"character varying(200)",1)
(public,sales_lbv,salesid,integer,1)
(public,sales_lbv,listid,integer,2)
(public,sales_lbv,sellerid,integer,3)
(public,sales_lbv,buyerid,integer,4)
(public,sales_lbv,eventid,integer,5)
(public,sales_lbv,dateid,smallint,6)
(public,sales_lbv,qtysold,smallint,7)
(public,sales_lbv,pricepaid,"numeric(8,2)",8)
(public,sales_lbv,commission,"numeric(8,2)",9)
(public,sales_lbv,saletime,"timestamp without time zone",10)
(public,event_lbv,eventid,integer,1)
(public,event_lbv,venueid,smallint,2)
(public,event_lbv,catid,smallint,3)
(public,event_lbv,dateid,smallint,4)
```



```
(public,event_lbv,eventname,"character varying(200)",5)
(public,event_lbv,starttime,"timestamp without time zone",6)
```

下列範例以資料表格式傳回所有近期繫結檢視的欄中繼資料。

```
select * from pg_get_late_binding_view_cols() cols(view_schema name, view_name name,
  col_name name, col_type varchar, col_num int);
```

view_schema	view_name	col_name	col_type	col_num
public	sales_lbv	salesid	integer	1
public	sales_lbv	listid	integer	2
public	sales_lbv	sellerid	integer	3
public	sales_lbv	buyerid	integer	4
public	sales_lbv	eventid	integer	5
public	sales_lbv	dateid	smallint	6
public	sales_lbv	qtysold	smallint	7
public	sales_lbv	pricepaid	numeric(8,2)	8
public	sales_lbv	commission	numeric(8,2)	9
public	sales_lbv	saletime	timestamp without time zone	10
public	event_lbv	eventid	integer	1
public	event_lbv	venueid	smallint	2
public	event_lbv	catid	smallint	3
public	event_lbv	dateid	smallint	4
public	event_lbv	eventname	character varying(200)	5
public	event_lbv	starttime	timestamp without time zone	6

PG_GET_SESSION_ROLES

傳回目前登入使用者在工作階段角色。使用者在工作階段角色是由身分提供者 (IdP) 為登入使用者定義的群組。例如，[Microsoft Azure Active Directory \(Azure AD\)](#) 等身分提供者 (IdP) 會驗證使用者的身分，並在使用者登入程序期間提供使用者所屬的任何外部群組。這些外部群組會轉換為 Amazon Redshift 角色，並可在目前的工作階段期間使用。這些角色稱為工作階段角色。管理員可以授與工作階段角色類似其他 Amazon Redshift 角色的權限。如需使用角色的詳細資訊，請參閱[角色型存取控制 \(RBAC\)](#)。如需使用身分提供者 (IdP) 管理身分的相關資訊，請參閱《Amazon Redshift 管理指南》中的[適用於 Amazon Redshift 的原生身分提供者 \(IdP\) 聯合](#)。

若要檢視 Amazon Redshift 目錄中定義的角色，請查詢系統檢視 [SVV_ROLES](#)。

語法

```
pg_get_session_roles()
```

傳回類型

由兩個值組成的一組列。第一個值有兩個部分，由冒號 (:) 分隔，其中包含 `idp-namespace:role-name`。`idp-namespace` 是身分提供者 (IdP) 的命名空間。`role-name` 是身分提供者 (IdP) 中外部群組的名稱。第二個值包含 `role-id`，這是角色識別碼。

使用須知

`PG_GET_SESSION_ROLES` 函數會針對每個傳回的工作階段角色傳回一列。

範例

下列範例會針對 Azure Active Directory IdP 中的每個角色傳回一個資料列。傳回的資料欄會轉換為 `sess_roles`，具有欄 `name` 和 `roleid`。每個 `name` 都包含 Azure Active Directory 命名空間和 Azure Active Directory 中的群組名稱。

```
SELECT * FROM pg_get_session_roles() AS sess_roles(name name, roleid integer);
```

name	roleid
-----	-----
my_aad:test_group_1	106204
my_aad:test_group_2	106205
my_aad:test_group_3	106206
my_aad:test_group_4	106207
my_aad:test_group_5	106208

下列範例會針對目前登入 IAM 使用者所屬的每個 IAM 群組傳回一列。傳回的資料欄會轉換為 `sess_roles`，具有欄 `name` 和 `roleid`。每個 `name` 都包含 IAM 命名空間和 IAM 群組名稱。

```
SELECT * FROM pg_get_session_roles() AS sess_roles(name name, roleid integer);
```

name	roleid
-----	-----
IAM:myGroup	110332

PG_LAST_COPY_COUNT

傳回目前工作階段中上次 COPY 命令執行所載入的資料列數目。`PG_LAST_COPY_COUNT` 會更新為最後一個 COPY ID，這是最後一個 COPY 開始載入程序的查詢 ID，即使載入失敗。查詢 ID 和 COPY ID 會在 COPY 命令開始載入程序時更新。

如果因為語法錯誤或權限不足而COPY失敗，則不會更新 COPY ID，且 PG_LAST_COPY_COUNT 會傳回先前的計數COPY。如果在目前的工作階段中沒有執行COPY命令，或如果載入期間最後一個COPY失敗，PG_LAST_COPY_COUNT 會傳回 0。如需詳細資訊，請參閱[PG_LAST_COPY_ID](#)。

語法

```
pg_last_copy_count()
```

傳回類型

傳回 BIGINT。

範例

下列查詢會傳回目前工作階段中最新COPY命令載入的列數。

```
select pg_last_copy_count();

pg_last_copy_count
-----
                192497
(1 row)
```

PG_LAST_COPY_ID

傳回目前工作階段中最近完成之COPY命令的查詢 ID。如果目前工作階段中未執行任何COPY命令，PG_LAST_COPY_ID 會傳回 -1。

PG_LAST_COPY_ID 的值會在COPY命令開始載入程序時更新。如果因為載入資料無效而COPY失敗，則會更新 COPY ID，因此您可以在查詢 STL_LASTCOPY_ 資料表時使用 PGLOAD__ERRORSID。如果將COPY交易復原，則不會更新 COPY ID。

如果COPY命令因載入程序開始之前發生的錯誤而失敗，例如語法錯誤、存取錯誤、無效憑證或權限不足，則不會更新 COPY ID。如果在分析壓縮步驟期間COPY失敗，該步驟在成功連線後但在資料載入之前開始，則不會更新 COPY ID。

語法

```
pg_last_copy_id()
```

傳回類型

傳回整數。

範例

下列查詢會傳回目前工作階段中最新COPY命令的查詢 ID。

```
select pg_last_copy_id();

pg_last_copy_id
-----
              5437
(1 row)
```

下列查詢會聯結 STL_LOAD_ERRORS 至 STL_LOADERROR_DETAIL，以檢視目前工作階段中最近載入期間發生的詳細資訊錯誤：

```
select d.query, substring(d.filename,14,20),
d.line_number as line,
substring(d.value,1,16) as value,
substring(le.err_reason,1,48) as err_reason
from stl_loaderror_detail d, stl_load_errors le
where d.query = le.query
and d.query = pg_last_copy_id();

 query | substring          | line | value  | err_reason
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
      558| allusers_pipe.txt | 251 | 251    | String contains invalid or unsupported
UTF8 code
      558| allusers_pipe.txt | 251 | ZRU29FGR | String contains invalid or unsupported
UTF8 code
      558| allusers_pipe.txt | 251 | Kaitlin | String contains invalid or unsupported
UTF8 code
      558| allusers_pipe.txt | 251 | Walter  | String contains invalid or unsupported
UTF8 code
```

PG_LAST_UNLOAD_ID

傳回目前工作階段中最近完成之UNLOAD命令的查詢 ID。如果目前工作階段中未執行任何UNLOAD命令，PG_LAST_UNLOAD_ID 會傳回 -1。

PG_LAST_UNLOAD_ID 的值會在UNLOAD命令開始載入程序時更新。如果因為載入資料無效而UNLOAD失敗，則會更新 UNLOAD ID，因此您可以使用 UNLOAD ID 進行進一步調查。如果將UNLOAD交易復原，則不會更新 UNLOAD ID。

如果UNLOAD命令因載入程序開始之前發生的錯誤而失敗，例如語法錯誤、存取錯誤、無效憑證或權限不足，則不會更新 UNLOAD ID。

語法

```
PG_LAST_UNLOAD_ID()
```

傳回類型

傳回整數。

範例

下列查詢會傳回目前工作階段中最新UNLOAD命令的查詢 ID。

```
select PG_LAST_UNLOAD_ID();

PG_LAST_UNLOAD_ID
-----
                5437
(1 row)
```

PG_LAST_QUERY_ID

傳回目前工作階段中最近完成之查詢的查詢 ID。如果目前工作階段中未執行任何查詢，PG_LAST_QUERY_ID 會傳回 -1。PG_LAST_QUERY_ID 不會針對僅在領導節點上執行的查詢傳回查詢 ID。如需詳細資訊，請參閱[僅限領導節點函數](#)。

語法

```
pg_last_query_id()
```

傳回類型

傳回整數。

範例

下列查詢傳回目前工作階段中完成的最後一個查詢的 ID。

```
select pg_last_query_id();
```

結果如下。

```
pg_last_query_id
-----
                5437
(1 row)
```

下列查詢傳回目前工作階段中最近完成之查詢的查詢 ID 和文字。

```
select query, trim(querytxt) as sqlquery
from stl_query
where query = pg_last_query_id();
```

結果如下。

```
query | sqlquery
-----+-----
 5437 | select name, loadtime from stl_file_scan where loadtime > 1000000;
(1 rows)
```

PG_LAST_UNLOAD_COUNT

傳回目前工作階段中最後一個完成的UNLOAD命令所卸載的資料列數目。即使操作失敗 UNLOAD，PGLAST__UNLOAD_COUNT也會以最後一個的查詢 ID 更新。查詢 ID 會在 UNLOAD 完成時更新。如果因為語法錯誤或權限不足而UNLOAD失敗，PG_LAST_UNLOAD_COUNT 會傳回上一個的計數UNLOAD。如果在目前的工作階段中沒有完成UNLOAD任何命令，或在卸載操作期間最後一個UNLOAD失敗，PG_LAST_UNLOAD_COUNT 會傳回 0。

語法

```
pg_last_unload_count()
```

傳回類型

傳回 BIGINT。

範例

下列查詢會傳回目前工作階段中最新UNLOAD命令卸載的資料列數目。

```
select pg_last_unload_count();

pg_last_unload_count
-----
                192497
(1 row)
```

SLICE_NUM 函數

傳回整數，對應於列的資料在叢集內所在的配量編號。SLICE_NUM 不帶參數。

語法

```
SLICE_NUM()
```

傳回類型

SLICE_NUM 函數會傳回整數。

範例

下列範例顯示哪些切片包含EVENTS資料表前十EVENT列的資料：

```
select distinct eventid, slice_num() from event order by eventid limit 10;

eventid | slice_num
-----+-----
        1 |         1
        2 |         2
        3 |         3
        4 |         0
        5 |         1
        6 |         2
        7 |         3
        8 |         0
        9 |         1
       10 |         2
(10 rows)
```

下列範例會傳回程式碼 (10000) ，以顯示沒有FROM陳述式的查詢在領導節點上執行：

```
select slice_num();
slice_num
-----
10000
(1 row)
```

USER

CURRENT_ 的同義詞USER。請參閱 [CURRENT_USER](#)。

ROLE_ISMEMBER_OF

如果角色是另一個角色的成員，則傳回 true。超級使用者可以檢查所有角色的成員資格。擁有 ACCESS SYSTEM TABLE 許可的普通使用者可以檢查所有使用者的會員資格。否則，一般使用者只能檢查他們有權存取的角色。如果提供的角色不存在或目前使用者無法存取該角色，Amazon Redshift 就會發生錯誤。

語法

```
role_is_member_of( role_name, granted_role_name)
```

引數

role_name

角色的名稱。

granted_role_name

授與角色的名稱。

傳回類型

傳回 BOOLEAN。

範例

下列查詢確認該角色不是 role1 的成員，也不是 role2 的成員。

```
SELECT role_is_member_of('role1', 'role2');
```



```
role_is_member_of
-----
False
```

USER_ISMEMBER_OF

如果使用者是角色或群組的成員，則傳回 true。超級使用者可以檢查所有使用者的成員資格。屬於 `sys:secadmin` 或 `sys:superuser` 角色成員的一般使用者可以檢查所有使用者的成員資格。否則，一般使用者只能檢查自己。如果提供的身分不存在或目前的使用者無法存取角色，Amazon Redshift 就會傳送錯誤訊息。

語法

```
user_is_member_of( user_name, role_name | group_name )
```

引數

user_name

使用者的名稱。

role_name

角色的名稱。

group_name

群組名稱。

傳回類型

傳回 BOOLEAN。

範例

下列查詢確認使用者不是 `role1` 的成員。

```
SELECT user_is_member_of('reguser', 'role1');
```

```
user_is_member_of
-----
```

False

VERSION

VERSION 函數會傳回有關目前安裝版本的詳細資訊，並在結尾提供特定的 Amazon Redshift 版本資訊。

Note

這是領導者節點函數。如果此函數參考使用者建立的資料表、STL 或 STV 系統資料表，或 SVV 或 SVL 系統檢視，則傳回錯誤。

語法

```
VERSION()
```

傳回類型

傳回 CHAR 或 VARCHAR 字串。

範例

下列範例顯示目前叢集的叢集版本資訊：

```
select version();
```

```
version
```

```
-----  
PostgreSQL 8.0.2 on i686-pc-linux-gnu, compiled by GCC gcc (GCC) 3.4.2 20041017 (Red  
Hat 3.4.2-6.fc3), Redshift 1.0.12103
```

其中 1.0.12103 是叢集版本編號。

Note

若要強制將您的叢集版本更新至最新，請調整您的[維護時段](#)。

保留字

以下是 Amazon Redshift 保留字的清單。您可將保留字用於分隔的分隔識別碼 (雙引號)。

Note

雖然 START 和 CONNECT 不是保留字，但如果您在查詢中使用 START 和 CONNECT 做為資料表別名，請使用分隔識別碼或 AS，以避免在執行期失敗。

如需更多詳細資訊，請參閱 [名稱與識別碼](#)。

AES128
AES256
ALL
ALLOWOVERWRITE
ANALYSE
ANALYZE
AND
ANY
ARRAY
AS
ASC
AUTHORIZATION
AZ64
BACKUP
BETWEEN
BINARY
BLANKSASNULL
BOTH
BYTEDICT
BZIP2
CASE
CAST
CHECK
COLLATE
COLUMN
CONSTRAINT
CREATE
CREDENTIALS
CROSS
CURRENT_DATE

CURRENT_TIME
CURRENT_TIMESTAMP
CURRENT_USER
CURRENT_USER_ID
DEFAULT
DEFERRABLE
DEFLATE
DEFRAG
DELTA
DELTA32K
DESC
DISABLE
DISTINCT
DO
ELSE
EMPTYASNULL
ENABLE
ENCODE
ENCRYPT
ENCRYPTION
END
EXCEPT
EXPLICIT
FALSE
FOR
FOREIGN
FREEZE
FROM
FULL
GLOBALDICT256
GLOBALDICT64K
GRANT
GROUP
GZIP
HAVING
IDENTITY
IGNORE
ILIKE
IN
INITIALLY
INNER
INTERSECT
INTERVAL
INTO

IS
ISNULL
JOIN
LEADING
LEFT
LIKE
LIMIT
LOCALTIME
LOCALTIMESTAMP
LUN
LUNS
LZO
LZOP
MINUS
MOSTLY16
MOSTLY32
MOSTLY8
NATURAL
NEW
NOT
NOTNULL
NULL
NULLS
OFF
OFFLINE
OFFSET
OID
OLD
ON
ONLY
OPEN
OR
ORDER
OUTER
OVERLAPS
PARALLEL
PARTITION
PERCENT
PERMISSIONS
PIVOT
PLACING
PRIMARY
RAW
READRATIO

RECOVER
REFERENCES
REJECTLOG
RESORT
RESPECT
RESTORE
RIGHT
SELECT
SESSION_USER
SIMILAR
SNAPSHOT
SOME
SYSDATE
SYSTEM
TABLE
TAG
TDES
TEXT255
TEXT32K
THEN
TIMESTAMP
TO
TOP
TRAILING
TRUE
TRUNCATECOLUMNS
UNION
UNIQUE
UNNEST
UNPIVOT
USER
USING
VERBOSE
WALLET
WHEN
WHERE
WITH
WITHOUT

系統資料表和檢視參考

Amazon Redshift 具有許多系統資料表和檢視，其中包含系統如何運作的相關資訊。您可以使用查詢任何其他資料庫資料表的方式，來查詢這些系統資料表和檢視。本節示範一些系統資料表查詢的例子並說明：

- 系統如何產生不同類型的資料表和檢視
- 您可以從這些資料表取得什麼類型的資訊
- 如何將 Amazon Redshift 系統資料表聯結至目錄資料表
- 如何管理不斷增多的系統資料表日誌檔案

某些系統資料表只能由 AWS 人員用於診斷目的。以下章節討論系統管理員或其他資料庫使用者可以查詢實用資訊的系統資料表。

Note

自動或手動叢集備份 (快照) 不包括系統資料表。STL 系統檢視會保留七天的日誌歷史記錄。保留日誌不需要客戶執行任何動作，但如果您想要儲存日誌資料超過 7 天，則必須定期將其複製到其他資料表或卸載到 Amazon S3。

主題

- [系統資料表和檢視的類型](#)
- [系統資料表和檢視中資料的可見性](#)
- [遷移僅供佈建查詢以SYS監控檢視查詢](#)
- [使用SYS監控檢視改善查詢識別符追蹤](#)
- [系統資料表查詢、程序和工作階段 ID](#)
- [SVV 中繼資料檢視](#)
- [SYS 監控檢視](#)
- [用於移轉至 SYS 監視檢視的系統檢視對映](#)
- [系統監控 \(僅限已佈建\)](#)
- [系統目錄資料表](#)

系統資料表和檢視的類型

系統資料表和檢視有以下幾種類型：

- SVV 檢視包含有關資料庫物件的資訊，其中包含暫態STV資料表的參考。
- SYS 檢視用於監控佈建叢集和無伺服器工作群組的查詢和工作負載用量。
- STL 系統會從保留到磁碟的日誌產生檢視，以提供系統歷史記錄。
- STV 資料表是虛擬系統資料表，其中包含目前系統資料的快照。他們是建基於暫時性記憶體內資料，而不會保存到磁碟型日誌或一般資料表。
- SVCS 檢視提供主要和並行擴展叢集上查詢的詳細資訊。
- SVL 檢視提供主叢集查詢的詳細資訊。

系統資料表和檢視不會使用相同的一致性模型做為一般資料表。查詢問題時請務必注意此問題，特別是對於STV資料表和SVV檢視而言。例如一般資料表 t1 (包含欄 c1)，預期以下查詢不會傳回任何列：

```
select * from t1
where c1 > (select max(c1) from t1)
```

但是以下對系統資料表的查詢可能會傳回列：

```
select * from stv_exec_state
where currenttime > (select max(currenttime) from stv_exec_state)
```

此查詢可能會傳回列的原因是 currenttime 為暫時性，而查詢中的兩個參考在評估時可能不會傳回相同的值。

另一方面，下列查詢可能不會傳回任何列：

```
select * from stv_exec_state
where currenttime = (select max(currenttime) from stv_exec_state)
```

系統資料表和檢視中資料的可見性

系統資料表和檢視中資料的可見性分為兩個類別：使用者可查看和超級使用者可查看。

擁有超級使用者權限的使用者才可以查看「超級使用者可查看」類別裡資料表的資料。一般使用者可以查看「使用者可查看」資料表裡的資料。若要讓一般使用者存取超級使用者可見的資料表，請將該資料表上的SELECT權限授予一般使用者。如需詳細資訊，請參閱[GRANT](#)。

依預設，在大部份「使用者可查看」資料表中，一般使用者無法查看另一位使用者產生的列。如果為一般使用者提供 [SYSLOG ACCESS UNRESTRICTED](#)，則該使用者可以看到使用者可見資料表中的所有資料列，包括其他使用者產生的資料列。如需詳細資訊，請參閱 [ALTER USER](#) 或 [CREATE USER](#)。所有使用者都可看見 SVV_TRANSACTIONS 中的所有資料列。如需資料可見性的詳細資訊，請參閱 AWS re:Post 知識庫文章 [如何允許 Amazon Redshift 資料庫一般使用者從其他使用者檢視叢集的系統資料表資料？](#)

對於中繼資料檢視，Amazon Redshift 不允許被授予 SYSLOG ACCESS 的使用者查看 UNRESTRICTED。

Note

若使用者擁有不受限制的系統資料表存取權限，該使用者就能查看其他使用者產生的資料。例如，STL_QUERY 和 STL_QUERY_TEXT 包含 INSERT、UPDATE 和 DELETE 陳述式的全文，其中可能包含敏感的使用者產生資料。

超級使用者可以查看所有資料表中的所有列。若要讓一般使用者存取超級使用者可見的資料表，請將該資料表上的 [GRANT SELECT](#) 權限提供給一般使用者。

篩選系統產生的查詢

查詢相關的系統資料表和檢視，例如 SVL_QUERY_SUMMARY、SVL_QLOG 和其他，通常包含大量自動產生的陳述式，Amazon Redshift 會使用這些陳述式來監控資料庫的狀態。超級使用者可以看到這些系統產生的查詢，但這些查詢通常作用不大。若要在從使用 userid 欄的系統資料表或系統檢視中選取時篩選掉它們，請將條件新增至 `userid > 1` WHERE 子句。例如：

```
select * from svl_query_summary where userid > 1
```

遷移僅供佈建查詢以SYS監控檢視查詢

從已佈建的叢集遷移至 Amazon Redshift Serverless

如果您要將佈建的叢集遷移至 Amazon Redshift Serverless，您可能要使用下列系統檢視的查詢，這些檢視只會儲存來自佈建叢集的資料。

- 所有STL檢視
- 所有STV檢視
- 所有SVCS檢視
- 所有SVL檢視
- 某些SVV檢視
- 如需 Amazon Redshift Serverless 中不支援SVV檢視的完整清單，請參閱 Amazon Redshift 管理指南 中的 [使用 Amazon Redshift Serverless 監控查詢和工作負載](#) 底部的清單。

若要繼續使用查詢，請重新調整它們以使用SYS監控檢視中定義的資料欄，這些資料欄對應於僅供佈建檢視中的資料欄。若要查看僅供佈建檢視與SYS監控檢視之間的映射關係，請前往 [用於移轉至 SYS 監視檢視的系統檢視對映](#)

停留在已佈建叢集的同時更新查詢

如果您不是遷移到 Amazon Redshift Serverless，您可能仍想要更新現有的查詢。SYS 監控檢視旨在方便使用並降低複雜性，提供完整的指標系列，以進行有效的監控和疑難排解。使用 [SYS_QUERY_HISTORY](#) 和 [SYS_QUERY_DETAIL](#) 等SYS檢視合併多個僅供佈建檢視的資訊，您可以簡化查詢。

使用SYS監控檢視改善查詢識別符追蹤

SYS 監控檢視，例如 [SYS_QUERY_HISTORY](#) 和 [SYS_QUERY_DETAIL](#) 包含 query_id 資料欄，其中會保留使用者查詢的識別碼。同樣地，僅供佈建的檢視 (例如 [STL_QUERY](#) 和 [SVL_QLOG](#)) 包含查詢欄，此欄也包含查詢識別碼。不過，SYS系統檢視中記錄的查詢識別碼與僅供佈建檢視中記錄的識別碼不同。

SYS 檢視的 query_id 資料欄值與僅供佈建檢視的查詢資料欄值之間的差異如下：

- 在SYS檢視中，query_id 資料欄會以使用者的原始格式記錄使用者提交的查詢。Amazon Redshift 最佳化工具可能會將它們分解為子查詢以提高效能，但是您執行的單一查詢在 [SYS_QUERY_HISTORY](#) 中仍只有一列。如果您想查看個別子查詢，可以在 [SYS_QUERY_DETAIL](#) 中找到它們。
- 在僅佈建的檢視中，查詢欄會記錄子查詢層級的查詢。如果 Amazon Redshift 最佳化工具將您的原始查詢重寫為多個子查詢，對於您執行的單一查詢，[STL_QUERY](#) 中將有多列具有不同的查詢識別碼值。

當您將監控和診斷查詢從僅供佈建檢視遷移至SYS檢視時，請考慮此差異並相應地編輯您的查詢。如需 Amazon Redshift 如何處理查詢的相關資訊，請參閱[查詢計劃和執行工作流程](#)。

範例

如需 Amazon Redshift 如何在僅供佈建和SYS監控檢視中以不同方式記錄查詢的範例，請參閱下列範例查詢。這是按照您在 Amazon Redshift 中執行的方式所編寫的查詢。

```
SELECT
  s_name
  , COUNT(*) AS numwait
FROM
  supplier,
  lineitem l1,
  orders,
  nation
WHERE   s_suppkey = l1.l_suppkey
        AND o_orderkey = l1.l_orderkey
        AND o_orderstatus = 'F'
        AND l1.l_receiptdate > l1.l_commitdate
        AND EXISTS (SELECT
                      *
                    FROM
                      lineitem l2
                    WHERE  l2.l_orderkey = l1.l_orderkey
                          AND l2.l_suppkey <> l1.l_suppkey )
        AND NOT EXISTS (SELECT
                        *
                      FROM
                        lineitem l3
                      WHERE  l3.l_orderkey = l1.l_orderkey
                            AND l3.l_suppkey <> l1.l_suppkey
                            AND l3.l_receiptdate > l3.l_commitdate )
        AND s_nationkey = n_nationkey
        AND n_name = 'UNITED STATES'
GROUP BY
  s_name
ORDER BY
  numwait DESC
  , s_name LIMIT 100;
```

在此之下，Amazon Redshift 查詢最佳化工具會將上述使用者提交的查詢重寫為 5 個子查詢。

第一個子查詢建立一個暫存資料表來實現一個子查詢。

```
CREATE TEMP TABLE volt_tt_606590308b512(l_orderkey
                                         , l_suppkey
                                         , s_name ) AS SELECT
                                         l1.l_orderkey
                                         , l1.l_suppkey
                                         , public.supplier.s_name
FROM
    public.lineitem AS l1,
    public.nation,
    public.orders,
    public.supplier
WHERE l1.l_commitdate <

    l1.l_receiptdate
    AND l1.l_orderkey =
    public.orders.o_orderkey
    AND l1.l_suppkey =
    public.supplier.s_suppkey
    AND public.nation.n_name
    = 'UNITED STATES'::CHAR(8)
    AND
    public.nation.n_nationkey = public.supplier.s_nationkey
    AND
    public.orders.o_orderstatus = 'F'::CHAR(1);
```

第二個子查詢會從暫存資料表收集統計資料。

```
padb_fetch_sample: select count(*) from volt_tt_606590308b512;
```

第三個子查詢建立另一個暫存資料表來具體化另一個子查詢，引用上面建立的暫存資料表。

```
CREATE TEMP TABLE volt_tt_606590308c2ef(l_orderkey
                                         , l_suppkey) AS (SELECT
    volt_tt_606590308b512.l_orderkey
                                         ,
    volt_tt_606590308b512.l_suppkey
                                         FROM
    public.lineitem AS l2,
    volt_tt_606590308b512
    WHERE l2.l_suppkey <>
    volt_tt_606590308b512.l_suppkey
```

```

                                AND l2.l_orderkey =
volt_tt_606590308b512.l_orderkey)
                                EXCEPT distinct (SELECT
volt_tt_606590308b512.l_orderkey, volt_tt_606590308b512.l_suppkey
                                FROM public.lineitem AS
l3, volt_tt_606590308b512
                                WHERE l3.l_commitdate <
l3.l_receiptdate
                                AND l3.l_suppkey <>
volt_tt_606590308b512.l_suppkey
                                AND l3.l_orderkey =
volt_tt_606590308b512.l_orderkey);

```

第四個子查詢再次收集暫存資料表的統計資料。

```
padb_fetch_sample: select count(*) from volt_tt_606590308c2ef
```

最後一個子查詢使用上面建立的暫存資料表來產生輸出。

```

SELECT
  volt_tt_606590308b512.s_name AS s_name
  , COUNT(*) AS numwait
FROM
  volt_tt_606590308b512,
  volt_tt_606590308c2ef
WHERE   volt_tt_606590308b512.l_orderkey = volt_tt_606590308c2ef.l_orderkey
        AND volt_tt_606590308b512.l_suppkey = volt_tt_606590308c2ef.l_suppkey
GROUP BY
  1
ORDER BY
  2 DESC
  , 1 ASC LIMIT 100;

```

在僅供佈建的系統檢視 STL_ 中 QUERY , Amazon Redshift 會在子查詢層級記錄五列 , 如下所示 :

```

SELECT userid, xid, pid, query, querytxt::varchar(100);
FROM stl_query
WHERE xid = 48237350
ORDER BY xid, starttime;

userid |  xid  |  pid  |  query  |
querytxt

```

```

-----+-----+-----+-----
+-----+-----+-----+-----+-----+
 101 | 48237350 | 1073840810 | 12058151 | CREATE TEMP TABLE
volt_tt_606590308b512(l_orderkey, l_suppkkey, s_name) AS SELECT l1.l_orderkey, l1.l
 101 | 48237350 | 1073840810 | 12058152 | padb_fetch_sample: select count(*) from
volt_tt_606590308b512
 101 | 48237350 | 1073840810 | 12058156 | CREATE TEMP TABLE
volt_tt_606590308c2ef(l_orderkey, l_suppkkey) AS (SELECT volt_tt_606590308b512.l_or
 101 | 48237350 | 1073840810 | 12058168 | padb_fetch_sample: select count(*) from
volt_tt_606590308c2ef
 101 | 48237350 | 1073840810 | 12058170 | SELECT s_name , COUNT(*) AS numwait FROM
supplier, lineitem l1, orders, nation WHERE s_suppkkey = l1.
(5 rows)

```

在SYS監控檢視 SYS_QUERY_ 中HISTORY , Amazon Redshift 會記錄查詢 , 如下所示 :

```

SELECT user_id, transaction_id, session_id, query_id, query_text::varchar(100)
FROM sys_query_history
WHERE transaction_id = 48237350
ORDER BY start_time;

 user_id | transaction_id | session_id | query_id |
         | query_text
-----+-----+-----+-----+-----+
 101 |      48237350 | 1073840810 | 12058149 | SELECT s_name , COUNT(*) AS numwait
FROM supplier, lineitem l1, orders, nation WHERE s_suppkkey = l1.

```

在 SYS_QUERY_ 中DETAIL , 您可以使用來自 SYS_QUERY_ 的 query_id 值來尋找子查詢層級詳細資訊HISTORY。child_query_sequence 欄顯示子查詢的執行順序。如需 SYS_QUERY_ 中資料欄的詳細資訊DETAIL , 請參閱 [SYS_QUERY_DETAIL](#)。

```

select user_id,
       query_id,
       child_query_sequence,
       stream_id,
       segment_id,
       step_id,
       start_time,
       end_time,
       duration,
       blocks_read,
       blocks_write,

```

```

    local_read_io,
    remote_read_io,
    data_skewness,
    time_skewness,
    is_active,
    spilled_block_local_disk,
    spilled_block_remote_disk
from sys_query_detail
where query_id = 12058149
      and step_id = -1
order by query_id,
         child_query_sequence,
         stream_id,
         segment_id,
         step_id;

```

```

user_id | query_id | child_query_sequence | stream_id | segment_id | step_id |
start_time      |          |          end_time      | duration | blocks_read |
blocks_write | local_read_io | remote_read_io | data_skewness | time_skewness |
is_active | spilled_block_local_disk | spilled_block_remote_disk
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
      101 | 12058149 |          1 |          0 |          0 |      -1 |
2023-09-27 15:40:38.512415 | 2023-09-27 15:40:38.533333 |      20918 |          0 |
          0 |          0 |          0 |          0 |          44 | f
|
          0 |          0
      101 | 12058149 |          1 |          1 |          1 |      -1 |
2023-09-27 15:40:39.931437 | 2023-09-27 15:40:39.972826 |      41389 |          12 |
          0 |          12 |          0 |          0 |          77 | f
|
          0 |          0
      101 | 12058149 |          1 |          2 |          2 |      -1 |
2023-09-27 15:40:40.584412 | 2023-09-27 15:40:40.613982 |      29570 |          32 |
          0 |          32 |          0 |          0 |          25 | f
|
          0 |          0
      101 | 12058149 |          1 |          2 |          3 |      -1 |
2023-09-27 15:40:40.582038 | 2023-09-27 15:40:40.615758 |      33720 |          0 |
          0 |          0 |          0 |          0 |          1 | f
|
          0 |          0
      101 | 12058149 |          1 |          3 |          4 |      -1 |
2023-09-27 15:40:46.668766 | 2023-09-27 15:40:46.705456 |      36690 |          24 |
          0 |          15 |          0 |          0 |          17 | f
|
          0 |          0

```

101		12058149		1		4		5		-1	
2023-09-27	15:40:46.707209		2023-09-27	15:40:46.709176		1967		0		0	
0		0		0		0		18		f	
						0					
101		12058149		1		4		6		-1	
2023-09-27	15:40:46.70656		2023-09-27	15:40:46.71289		6330		0		0	
0		0		0		0		0		f	
						0					
101		12058149		1		5		7		-1	
2023-09-27	15:40:46.71405		2023-09-27	15:40:46.714343		293		0		0	
0		0		0		0		0		f	
						0					
101		12058149		2		0		0		-1	
2023-09-27	15:40:52.083907		2023-09-27	15:40:52.087854		3947		0		0	
0		0		0		0		35		f	
						0					
101		12058149		2		1		1		-1	
2023-09-27	15:40:52.089632		2023-09-27	15:40:52.091129		1497		0		0	
0		0		0		0		11		f	
						0					
101		12058149		2		1		2		-1	
2023-09-27	15:40:52.089008		2023-09-27	15:40:52.091306		2298		0		0	
0		0		0		0		0		f	
						0					
101		12058149		3		0		0		-1	
2023-09-27	15:40:56.882013		2023-09-27	15:40:56.897282		15269		0		0	
0		0		0		0		29		f	
						0					
101		12058149		3		1		1		-1	
2023-09-27	15:40:59.718554		2023-09-27	15:40:59.722789		4235		0		0	
0		0		0		0		13		f	
						0					
101		12058149		3		2		2		-1	
2023-09-27	15:40:59.800382		2023-09-27	15:40:59.807388		7006		0		0	
0		0		0		0		58		f	
						0					
101		12058149		3		3		3		-1	
2023-09-27	15:41:06.488685		2023-09-27	15:41:06.493825		5140		0		0	
0		0		0		0		56		f	
						0					
101		12058149		3		3		4		-1	
2023-09-27	15:41:06.486206		2023-09-27	15:41:06.497756		11550		0		0	
0		0		0		0		2		f	
						0					

101		12058149		3		4		5		-1	
2023-09-27	15:41:06.499201		2023-09-27	15:41:06.500851		1650		0		0	
0		0		0		0		15		f	
						0					
101		12058149		3		4		6		-1	
2023-09-27	15:41:06.498609		2023-09-27	15:41:06.500949		2340		0		0	
0		0		0		0		0		f	
						0					
101		12058149		3		5		7		-1	
2023-09-27	15:41:06.502945		2023-09-27	15:41:06.503282		337		0		0	
0		0		0		0		0		f	
						0					
101		12058149		4		0		0		-1	
2023-09-27	15:41:06.62899		2023-09-27	15:41:06.631452		2462		0		0	
0		0		0		0		22		f	
						0					
101		12058149		4		1		1		-1	
2023-09-27	15:41:06.632313		2023-09-27	15:41:06.63391		1597		0		0	
0		0		0		0		20		f	
						0					
101		12058149		4		1		2		-1	
2023-09-27	15:41:06.631726		2023-09-27	15:41:06.633813		2087		0		0	
0		0		0		0		0		f	
						0					
101		12058149		5		0		0		-1	
2023-09-27	15:41:12.571974		2023-09-27	15:41:12.584234		12260		0		0	
0		0		0		0		39		f	
						0					
101		12058149		5		0		1		-1	
2023-09-27	15:41:12.569815		2023-09-27	15:41:12.585391		15576		0		0	
0		0		0		0		4		f	
						0					
101		12058149		5		1		2		-1	
2023-09-27	15:41:13.758513		2023-09-27	15:41:13.76401		5497		0		0	
0		0		0		0		39		f	
						0					
101		12058149		5		1		3		-1	
2023-09-27	15:41:13.749		2023-09-27	15:41:13.772987		23987		0		0	
0		0		0		0		32		f	
						0					
101		12058149		5		2		4		-1	
2023-09-27	15:41:13.799526		2023-09-27	15:41:13.813506		13980		0		0	
0		0		0		0		62		f	
						0					

```

101 | 12058149 |          5 |          2 |          5 |      -1 |
2023-09-27 15:41:13.798823 | 2023-09-27 15:41:13.813651 |      14828 |          0 |
          0 |          0 |          0 |          0 |          0 | f
|          0 |          0
(28 rows)

```

系統資料表查詢、程序和工作階段 ID

分析系統資料表中出現的查詢、程序和工作階段 ID 時，請注意下列事項：

- 查詢 ID 值（在 `query_id` 和 等資料欄中 `query`）可以隨著時間重複使用。
- 程序 ID 或工作階段 ID 值（在 `process_id`、`pid` 和 等資料欄中 `session_id`）可以隨著時間重複使用。
- 交易 ID 值（在 `transaction_id` 和 等資料欄中 `xid`）是唯一的。

SVV 中繼資料檢視

SVV 檢視是 Amazon Redshift 中的系統檢視，其中包含資料庫物件的相關資訊。

Note

如果資料庫回應因任何原因失敗 ERROR，Amazon Redshift 會報告 WARNING，而不是。當您在資料共用中查詢物件時，Amazon Redshift 不會 ERROR 傳送訊息。

主題

- [SVV_ACTIVE_CURSORS](#)
- [SVV_ALL_COLUMNS](#)
- [SVV_ALL_SCHEMAS](#)
- [SVV_ALL_TABLES](#)
- [SVV_ALTER_TABLE_RECOMMENDATIONS](#)
- [SVV_ATTACHED_MASKING_POLICY](#)
- [SVV_COLUMNS](#)
- [SVV_COLUMN_PRIVILEGES](#)

- [SVV_DATABASE_PRIVILEGES](#)
- [SVV_DATASHARE_PRIVILEGES](#)
- [SVV_DATASHARES](#)
- [SVV_DATASHARE_CONSUMERS](#)
- [SVV_DATASHARE_OBJECTS](#)
- [SVV_DEFAULT_PRIVILEGES](#)
- [SVV_DISKUSAGE](#)
- [SVV_EXTERNAL_COLUMNS](#)
- [SVV_EXTERNAL_DATABASES](#)
- [SVV_EXTERNAL_PARTITIONS](#)
- [SVV_EXTERNAL_SCHEMAS](#)
- [SVV_EXTERNAL_TABLES](#)
- [SVV_FUNCTION_PRIVILEGES](#)
- [SVV_GEOGRAPHY_COLUMNS](#)
- [SVV_GEOMETRY_COLUMNS](#)
- [SVV_IAM_PRIVILEGES](#)
- [SVV_IDENTITY_PROVIDERS](#)
- [SVV_INTEGRATION](#)
- [SVV_INTEGRATION_TABLE_STATE](#)
- [SVV_INTERLEAVED_COLUMNS](#)
- [SVV_LANGUAGE_PRIVILEGES](#)
- [SVV_MASKING_POLICY](#)
- [SVV_ML_MODEL_INFO](#)
- [SVV_ML_MODEL_PRIVILEGES](#)
- [SVV_MV_DEPENDENCY](#)
- [SVV_MV_INFO](#)
- [SVV_QUERY_INFLIGHT](#)
- [SVV_QUERY_STATE](#)
- [SVV_REDSHIFT_COLUMNS](#)

- [SVV_REDSHIFT_DATABASES](#)
- [SVV_REDSHIFT_FUNCTIONS](#)
- [SVV_REDSHIFT_SCHEMA_QUOTA](#)
- [SVV_REDSHIFT_SCHEMAS](#)
- [SVV_REDSHIFT_TABLES](#)
- [SVV_RELATION_PRIVILEGES](#)
- [SVV_RLS_APPLIED_POLICY](#)
- [SVV_RLS_ATTACHED_POLICY](#)
- [SVV_RLS_POLICY](#)
- [SVV_RLS_RELATION](#)
- [SVV_ROLE_GRANTS](#)
- [SVV_ROLES](#)
- [SVV_SCHEMA_PRIVILEGES](#)
- [SVV_SCHEMA_QUOTA_STATE](#)
- [SVV_SYSTEM_PRIVILEGES](#)
- [SVV_TABLE_INFO](#)
- [SVV_TABLES](#)
- [SVV_TRANSACTIONS](#)
- [SVV_USER_GRANTS](#)
- [SVV_USER_INFO](#)
- [SVV_VACUUM_PROGRESS](#)
- [SVV_VACUUM_SUMMARY](#)

SVV_ACTIVE_CURSORS

SVV_ACTIVE_CURSORS 顯示目前開啟游標的詳細資訊。如需詳細資訊，請參閱[DECLARE](#)。

SVV所有使用者都可看見 `_ACTIVE_CURSORS`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。使用者只能檢視該使用者所開啟的游標。超級使用者可檢視所有游標。

資料表欄

欄名稱	資料類型	描述
user_id	integer	建立游標的使用者 ID。
cursor_name	varchar(128)	游標的名稱。
transaction_id	bigint(128)	交易的 ID。
session_id	integer	具有作用中游標的處理程序的 ID。
declare_time	timestamp	宣告游標的時間。
total_bytes	bigint	游標結果集的大小 (以位元組為單位)。
total_rows	bigint	游標結果集中的列數。
fetches_rows	bigint	目前從游標結果集擷取的列數。
cursor_storage_limit_used_percent	integer	指標目前使用的磁碟空間百分比。

SVV_ALL_COLUMNS

使用 SVV_ALL_COLUMNS 從 Amazon Redshift 資料表檢視資料欄聯合，如 SVV_REDSHIFT_COLUMNS 和所有外部資料表中所有外部資料欄的合併清單所示。如需 Amazon Redshift 欄的相關資訊，請參閱[SVV_REDSHIFT_COLUMNS](#)。

SVV 所有使用者都可看見 _ALL_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	資料庫的名稱。
schema_name	varchar(128)	結構描述的名稱。
table_name	varchar(128)	資料表的名稱。
column_name	varchar(128)	欄位的名稱。
ordinal_position	integer	資料表中欄位的位置。
column_default	varchar(4000)	欄位的預設值。
is_nullable	varchar(3)	指出欄位是否可為 Null 的值。 可能的值是 yes 和 no。
data_type	varchar(128)	欄的資料類型。
character_maximum_length	integer	欄位中的最大字元數。
numeric_precision	integer	數值精確度。
numeric_scale	integer	數值擴展。
remarks	varchar(256)	備註。

範例查詢

下列範例會傳回 SVV_ALL_ 的輸出 COLUMNS。

```
SELECT *
FROM svv_all_columns
WHERE database_name = 'tickit_db'
      AND TABLE_NAME = 'tickit_sales_redshift'
ORDER BY COLUMN_NAME,
      SCHEMA_NAME
LIMIT 5;
```

```

database_name | schema_name |      table_name      | column_name | ordinal_position |
| column_default | is_nullable | data_type | character_maximum_length |
numeric_precision | numeric_scale | remarks
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
tickit_db     | public     | tickit_sales_redshift | buyerid    | 4                |
|             | NO        | integer    |             |                  | 32
| 0           |           |            |             |                  |
tickit_db     | public     | tickit_sales_redshift | commission  | 9                |
|             | YES       | numeric    |             |                  | 8
| 2           |           |            |             |                  |
tickit_db     | public     | tickit_sales_redshift | dateid     | 7                |
|             | NO       | smallint   |             |                  | 16
| 0           |           |            |             |                  |
tickit_db     | public     | tickit_sales_redshift | eventid    | 5                |
|             | NO       | integer    |             |                  | 32
| 0           |           |            |             |                  |
tickit_db     | public     | tickit_sales_redshift | listid     | 2                |
|             | NO       | integer    |             |                  | 32
| 0           |           |            |             |                  |

```

SVV_ALL_SCHEMAS

使用 SVV_ALL_SCHEMAS 檢視 Amazon Redshift 結構描述的結合，如 SVV_REDSHIFT_SCHEMAS 和來自所有資料庫的所有外部結構描述的合併清單所示。如需 Amazon Redshift 結構描述的相關資訊，請參閱[SVV_REDSHIFT_SCHEMAS](#)。

SVV 所有使用者都可看見 _ALL_SCHEMAS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	結構描述所在之資料庫的名稱。
schema_name	varchar(128)	結構描述的名稱。

欄名稱	資料類型	描述
schema_owner	integer	結構描述擁有者的使用者 ID。如需有關使用者的資訊IDs，請參閱 PG_USER_INFO 。
schema_type	varchar(128)	結構描述的類型。可能的值是 external、local 和 shared schemas。
schema_acl	varchar(128)	字串，定義結構描述之指定使用者或使用者群組的許可。
source_database	varchar(128)	外部結構描述的來源資料庫名稱。
schema_option	varchar(256)	結構描述的選項。這是外部結構描述屬性。

範例查詢

下列範例傳回 SVV_ALL_ 的輸出SCHEMAS。

```
SELECT *
FROM svv_all_schemas
WHERE database_name = 'tickit_db'
ORDER BY database_name,
        SCHEMA_NAME;
```

```
database_name |      schema_name      | schema_owner | schema_type | schema_acl |
source_database | schema_option
-----+-----+-----+-----+-----
+-----+-----
tickit_db    |      public          |      1      |      shared  |             |
|
```


SVV_ALL_TABLES

使用 SVV_ALL_TABLES 檢視 Amazon Redshift 資料表的聯合，如 SVV_REDSHIFT_TABLES 和所有外部結構描述中所有外部資料表的合併清單所示。如需有關 Amazon Redshift 資料表的資訊，請參閱[SVV_REDSHIFT_TABLES](#)。

SVV所有使用者都可看見 _ALL_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	資料表所在之資料庫的名稱。
schema_name	varchar(128)	資料表的結構描述名稱。
table_name	varchar(128)	資料表的名稱。
table_acl	varchar(128)	字串，定義資料表之指定使用者或使用者群組的許可。
table_type	varchar(128)	資料表的類型。可能的值是 views、base tables、external tables 和 shared tables。
remarks	varchar(256)	備註。

範例查詢

下列範例傳回 SVV_ALL_ 的輸出 TABLES。

```
SELECT *
FROM svv_all_tables
WHERE database_name = 'ticket_db'
ORDER BY TABLE_NAME,
        SCHEMA_NAME
LIMIT 5;
```

```

database_name | schema_name |          table_name          | table_type | table_acl |
remarks
-----+-----+-----+-----+-----
+-----
  tickit_db   |   public   | tickit_category_redshift    |   TABLE   |           |
  tickit_db   |   public   |  tickit_date_redshift       |   TABLE   |           |
  tickit_db   |   public   |  tickit_event_redshift      |   TABLE   |           |
  tickit_db   |   public   | tickit_listing_redshift     |   TABLE   |           |
  tickit_db   |   public   |  tickit_sales_redshift      |   TABLE   |           |

```

如果 `table_acl` 值為 `null`，表示沒有明確授予對應資料表的存取權限。

SVV_ALTER_TABLE_RECOMMENDATIONS

記錄資料表目前的 Amazon Redshift Advisor 建議。此檢視會顯示所有資料表的建議，不論這些資料表是否定義為自動最佳化。若要檢視資料表是否已定義為自動最佳化，請參閱[SVV_TABLE_INFO](#)。項目只會針對目前工作階段資料庫中可見的資料表而顯示。在套用建議之後（無論是由 Amazon Redshift 或您執行），該建議將不再出現在檢視中。

SVV只有超級使用者可以看到 `_ALTER_TABLE_RECOMMENDATIONS`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	Description (描述)
<code>type</code>	character (30)	建議的類型。可能的值是 <code>distkey</code> 和 <code>sortkey</code> 。
<code>database</code>	character (128)	資料庫名稱。
<code>table_id</code>	integer	資料表識別碼。
<code>group_id</code>	integer	一組建議的群組編號。應該套用一組中的所有建議，以查看最大收益。對於排序索引鍵建議，可能的值為 <code>-1</code> ；對於分散索引鍵建議，可能的值為大於零的數字。

欄名稱	資料類型	Description (描述)
DDL	character (1024)	必須執行的SQL陳述式才能套用建議。
auto_eligible	character (1)	該值表示該建議是否符合 Amazon Redshift 自動執行的資格。如果值為 t，則指示為 true，如果 f 則為 false。

範例查詢

在下列範例中，結果中的列會顯示分散索引鍵和排序索引鍵的建議。這些列也會顯示建議是否符合 Amazon Redshift 自動套用的資格。

```
select type, database, table_id, group_id, ddl, auto_eligible
from svv_alter_table_recommendations;
```

```
type          | database | table_id | group_id | ddl
              |          |          |          |
              | auto_eligible
diststyle | db0      | 117884   | 2        | ALTER TABLE "sch"."dp21235_tbl_1" ALTER
DISTSTYLE KEY DISTKEY "c0"
              | f
diststyle | db0      | 117892   | 2        | ALTER TABLE "sch"."dp21235_tbl_1" ALTER
DISTSTYLE KEY DISTKEY "c0"
              | f
diststyle | db0      | 117885   | 1        | ALTER TABLE "sch"."catalog_returns"
ALTER DISTSTYLE KEY DISTKEY "cr_sold_date_sk", ALTER COMPOUND SORTKEY
("cr_sold_date_sk","cr_returned_time_sk") | t
sortkey   | db0      | 117890   | -1       | ALTER TABLE "sch"."customer_addresses"
ALTER COMPOUND SORTKEY ("ca_address_sk")
              | t
```

SVV_ATTACHED_MASKING_POLICY

使用 SVV_ATTACHED_MASKING_POLICY 來檢視目前連線資料庫上附加政策的所有關係和角色/使用者。

只有具有 `sys:secadmin` 角色的超級使用者和使用者才能檢視 `SVV_ATTACHED_MASKING_POLICY`。一般使用者將看到 0 列。

資料表欄

欄名稱	資料類型	描述
<code>policy_name</code>	text	連接至資料表的遮罩政策的名稱。
<code>schema_name</code>	text	連接政策之資料表的結構描述。
<code>table_name</code>	text	連接政策的資料表名稱。
<code>table_type</code>	text	連接政策的資料表類型。
<code>grantor</code>	text	連接政策的使用者名稱。
<code>grantee</code>	text	連接政策的使用者/角色名稱。
<code>grantee_type</code>	text	<code>grantee</code> 的類型。這可以是 <code>role</code> 、 <code>user</code> 或 <code>public</code> 。
<code>priority</code>	int	連接政策的優先順序。
<code>input_columns</code>	text	連接政策的輸入欄屬性。
<code>output_columns</code>	text	連接政策的輸出欄屬性。

內部函數

`SVV_ATTACHED_MASKING_POLICY` 支援下列內部函數：

`mask_get_policy_for_role_on_column`

取得套用至指定欄/角色配對的最高優先順序政策。

語法

```
mask_get_policy_for_role_on_column
```

```
(relschema,  
relname,  
colname,  
rolename);
```

參數

relschema

政策所在之結構描述的名稱。

relname

政策所在的資料表的名稱。

colname

連接政策之欄的名稱。

rolename

政策連接的角色的名稱。

mask_get_policy_for_user_on_column

取得套用至指定欄/使用者配對的最高優先順序政策。

語法

```
mask_get_policy_for_user_on_column  
    (relschema,  
    relname,  
    colname,  
    username);
```

參數

relschema

政策所在之結構描述的名稱。

relname

政策所在的資料表的名稱。

colname

連接政策之欄的名稱。

rolename

政策連接的使用者名稱。

SVV_COLUMNS

使用 SVV_COLUMNS 檢視本機和外部資料表和檢視資料欄的目錄資訊，包括[繫結時間較晚的檢視](#)。

SVV所有使用者都可看見 _COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

SVV_COLUMNS 檢視會從 [系統目錄資料表](#) (具有 PG 字首的資料表)

和[SVV_EXTERNAL_COLUMNS](#)系統檢視聯結資料表中繼資料。此系統目錄會說明 Amazon Redshift 資料庫資料表。SVV_EXTERNAL_COLUMNS 說明與 Amazon Redshift Spectrum 搭配使用的外部資料表。

所有使用者可以查看系統目錄資料表中的所有列。一般使用者只能查看 SVV_EXTERNAL_COLUMNS 檢視中已授予存取權的外部資料表的資料欄定義。雖然一般使用者可以看見系統目錄資料表中的資料表中繼資料，如果他們擁有該資料表，或獲授予存取權，他們只能夠從使用者定義的資料表選取資料。

資料表欄

欄名稱	資料類型	描述
table_catalog	text	資料表所在之目錄的名稱。
table_schema	text	資料表的結構描述名稱。
table_name	text	表格的名稱。
column_name	text	欄位的名稱。
ordinal_position	int	資料表中欄位的位置。
column_default	text	欄位的預設值。
is_nullable	text	指出欄位是否可為 Null 的值。

欄名稱	資料類型	描述
data_type	text	欄的資料類型。
character_maximum_length	int	欄位中的最大字元數。
numeric_precision	int	數值精確度。如果 data_type 欄是數值，則此欄會傳回整個值中的有效位數。
numeric_precision_radix	int	數值精準度基數。如果 data_type 欄是數值，則此欄會傳回 numeric_precision 和 numeric_scale 欄的基數。
numeric_scale	int	數值擴展。如果 data_type 欄是數值，則此欄會傳回十進位值中的有效位數。
datetime_precision	int	日期時間精準度。
interval_type	text	間隔類型。
interval_precision	text	間隔精準度。
character_set_catalog	text	字元集目錄。
character_set_schema	text	字元集結構描述。
character_set_name	text	字元集名稱。
collation_catalog	text	定序目錄。
collation_schema	text	定序結構描述。
collation_name	text	定序名稱。
domain_name	text	網域名稱。
remarks	text	備註。

SVV_COLUMN_PRIVILEGES

使用 SVV_COLUMN_PRIVILEGES 檢視明確授予目前資料庫中使用者、角色和群組的資料欄許可。

SVV下列使用者可以看到 _COLUMN_PRIVILEGES：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定關係所在的命名空間的名稱。
relation_name	text	關係的名稱。
column_name	text	欄位的名稱。
privilege_type	text	許可的類型。可能的值為 SELECT 或 UPDATE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。

範例查詢

下列範例顯示 SVV_COLUMN_ 的結果PRIVILEGES。

```
SELECT
  namespace_name,relation_name,COLUMN_NAME,privilege_type,identity_name,identity_type
FROM svv_column_privileges WHERE relation_name = 'lineitem';
```



```

namespace_name | relation_name | column_name | privilege_type | identity_name |
identity_type
-----+-----+-----+-----+-----
+-----
   public      | lineitem     | l_orderkey  | SELECT        | reguser      |
user
   public      | lineitem     | l_orderkey  | SELECT        | role1        |
role
   public      | lineitem     | l_partkey   | SELECT        | reguser      |
user
   public      | lineitem     | l_partkey   | SELECT        | role1        |
role

```

SVV_DATABASE_PRIVILEGES

使用 SVV_DATABASE_PRIVILEGES 檢視明確授予 Amazon Redshift 叢集中使用者、角色和群組的資料庫許可。

SVV下列使用者可以看到 _DATABASE_PRIVILEGES：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
database_name	text	資料庫的名稱。
privilege_type	text	許可的類型。對於具有 privilege_scope of 資料庫的許可，可能的值為 USAGE、CREATE、TEMPORARY、TEMP和 ALTER。對於資料庫以外的 privilege_scope 值，可能的值包含許可範圍上可用的任何許可類型。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。

欄名稱	資料類型	描述
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。
privilege_scope	text	<p>在 privilege_type 中指定的許可範圍。可能的值如下：</p> <ul style="list-style-type: none"> • DATABASE • SCHEMAS • TABLES • FUNCTIONS • LANGUAGES <p>如需有關範圍許可的資訊，請前往 限定範圍權限。</p>

範例查詢

下列範例顯示 SVV_DATABASE_ 的結果PRIVILEGES。

```
SELECT database_name,privilege_type,identity_name,identity_type,admin_option FROM
svv_database_privileges
WHERE database_name = 'test_db';
```

database_name	privilege_type	identity_name	identity_type	admin_option
test_db	CREATE	reguser	user	False
test_db	CREATE	role1	role	False
test_db	TEMP	public	public	False
test_db	TEMP	role1	role	False

SVV_DATASHARE_PRIVILEGES

使用 SVV_DATASHARE_PRIVILEGES 檢視明確授予 Amazon Redshift 叢集中使用者、角色和群組的資料共用許可。

SVV下列使用者可以看到 _DATASHARE_PRIVILEGES：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
datashare_name	text	資料共用的名稱。
privilege_type	text	許可的類型。可能的值為 ALTER 或 SHARE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_DATASHARE_ 的結果PRIVILEGES。

```
SELECT datashare_name,privilege_type,identity_name,identity_type,admin_option FROM
svv_datashare_privileges
WHERE datashare_name = 'demo_share';
```

datashare_name	privilege_type	identity_name	identity_type	admin_option
demo_share	ALTER	superuser	user	False
demo_share	ALTER	reguser	user	False

SVV_DATASHARES

使用 SVV_DATASHARES 檢視在叢集上建立的資料共用清單，以及與叢集共用的資料共用清單。

SVV下列使用者可以看到 _DATASHARES：

- 超級使用者
- 資料共用擁有者
- 具有資料共用 ALTER 或 USAGE許可的使用者

其他使用者看不到任何列。如需有關 ALTER和 USAGE許可的資訊，請參閱 [GRANT](#)。

資料表欄

欄名稱	資料類型	描述
share_name	varchar(128)	資料共用的名稱。
share_id	integer	資料共用的 ID。
share_owner	integer	資料共用的擁有者。
source_database	varchar(128)	此資料共用的來源資料庫。
consumer_database	varchar(128)	從此資料共用建立的消費者資料庫。
share_type	varchar(8)	資料共用的類型。可能的值為 INBOUND和 OUTBOUND。
createdate	沒有時區的時間戳記	建立資料共用的日期。
is_publicaccessible	boolean	屬性，指定資料共用是否可以共用至可公開存取的叢集。

欄名稱	資料類型	描述
share_acl	varchar(256)	字串，定義資料共用之指定使用者或使用者群組的許可。
producer_account	varchar(16)	資料共用生產者帳戶的 ID。
producer_namespace	varchar(64)	資料共用生產者叢集的唯一叢集識別碼。
managed_by	varchar(64)	指定管理資料共用 AWS 的服務的屬性。

使用須知

擷取其他中繼資料 – 使用 share_owner 欄中傳回的整數，您可以加入 `usesysid` [SVL_USER_INFO](#)，以取得資料共用擁有者的資料。這包括名稱和其他屬性。

範例查詢

下列範例會傳回 SVV_ 的輸出 DATASHARES。

```
SELECT share_owner, source_database, share_type, is_publicaccessible
FROM svv_datashares
WHERE share_name LIKE 'ticket_datashare%'
AND source_database = 'dev';
```

```
share_owner | source_database | share_type | is_publicaccessible
-----+-----+-----+-----
      100    |      dev       |  OUTBOUND |          True
(1 rows)
```

下列範例會傳回輸出資料共用的 SVV_DATASHARES 輸出。

```
SELECT share_name, share_owner, btrim(source_database), btrim(consumer_database),
share_type, is_publicaccessible, share_acl, btrim(producer_account),
btrim(producer_namespace), btrim(managed_by) FROM svv_datashares WHERE share_type =
'OUTBOUND';
```

```

  share_name | share_owner | source_database | consumer_database | share_type |
  is_publicaccessible | share_acl | producer_account | producer_namespace
  | managed_by
-----+-----+-----+-----
+-----+-----+-----+-----
+-----+-----+-----+-----
  salesshare | 1 | dev | | OUTBOUND |
  True | | 123456789012 | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
marketingshare | 1 | dev | | OUTBOUND |
  True | | 123456789012 | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |

```

下列範例會傳回傳入資料共用的 SVV_DATASHARES 輸出。

```

SELECT share_name, share_owner, btrim(source_database), btrim(consumer_database),
share_type, is_publicaccessible, share_acl, btrim(producer_account),
btrim(producer_namespace), btrim(managed_by) FROM svv_datashares WHERE share_type =
'INBOUND';

```

```

  share_name | share_owner | source_database | consumer_database | share_type |
  is_publicaccessible | share_acl | producer_account | producer_namespace
  | managed_by
-----+-----+-----+-----
+-----+-----+-----+-----
+-----+-----+-----+-----
  salesshare | | | | INBOUND |
  False | | 123456789012 | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d
  |
marketingshare | | | | INBOUND |
  False | | 123456789012 | 13b8833d-17c6-4f16-8fe4-1a018f5ed00d |
ADX

```

SVV_DATASHARE_CONSUMERS

使用 SVV_DATASHARE_CONSUMERS 檢視叢集上建立之資料共用的取用者清單。

SVV下列使用者可以看到 _DATASHARE_CONSUMERS：

- 超級使用者
- 資料共用擁有者
- 具有資料共用 ALTER 或 USAGE許可的使用者

其他使用者看不到任何列。如需有關 ALTER和 USAGE許可的資訊，請參閱 [GRANT](#)。

資料表欄

欄名稱	資料類型	描述
share_name	varchar(128)	資料共用的名稱。
consumer_account	varchar(16)	資料共用消費者的帳戶 ID。
consumer_namespace	varchar(64)	資料共用消費者叢集的唯一叢集識別碼。
share_date	沒有時區的時間戳記	共用資料共用的日期。

範例查詢

下列範例會傳回 SVV_DATASHARE_ 的輸出CONSUMERS。

```
SELECT count(*)
FROM svv_datashare_consumers
WHERE share_name LIKE 'tickit_datashare%';
```

1

SVV_DATASHARE_OBJECTS

使用 SVV_DATASHARE_OBJECTS 檢視在叢集上建立或與叢集共用的所有資料共用中的物件清單。

SVV所有使用者都可看見 _DATASHARE_OBJECTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需有關檢視資料共用清單的資訊，請參閱 [SVV_DATASHARES](#)。

資料表欄

欄名稱	資料類型	描述
share_type	varchar(8)	所指定資料共用的類型。可能的值為 OUTBOUND和 INBOUND。
share_name	varchar(128)	資料共用的名稱。
object_type	varchar(64)	所指定物件的類型。可能的值是 schemas、tables、views、late binding views、materialized views 以及 functions 。
object_name	varchar(512)	物件的名稱。物件名稱會延伸至包含結構描述名稱，例如 schema1.t1。
producer_account	varchar(16)	資料共用生產者帳戶的 ID。
producer_namespace	varchar(64)	資料共用生產者叢集的唯一叢集識別碼。
include_new	boolean	指定是否將指定結構描述中建立的任何未來資料表、檢視或SQL使用者定義函數（UDFs）新增至資料共用的屬性。此參數僅適用於OUTBOUND資料共用，且僅適用於資料共用中的結構描述類型。

範例查詢

下列範例會傳回 SVV_DATASHARE_ 的輸出OBJECTS。

```
SELECT share_type,
```



```

    btrim(share_name)::varchar(16) AS share_name,
    object_type,
    object_name
FROM svv_datashare_objects
WHERE share_name LIKE 'tickit_datashare%'
AND object_name LIKE '%tickit%'
ORDER BY object_name
LIMIT 5;

```

share_type	share_name	object_type	object_name
OUTBOUND	tickit_datashare	table	public.tickit_category_redshift
OUTBOUND	tickit_datashare	table	public.tickit_date_redshift
OUTBOUND	tickit_datashare	table	public.tickit_event_redshift
OUTBOUND	tickit_datashare	table	public.tickit_listing_redshift
OUTBOUND	tickit_datashare	table	public.tickit_sales_redshift

```
SELECT * FROM SVV_DATASHARE_OBJECTS WHERE share_name like 'sales%';
```

share_type	share_name	object_type	object_name	producer_account	producer_namespace	include_new
OUTBOUND	salesshare	schema	public	123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	t
OUTBOUND	salesshare	table	public.sales	123456789012	13b8833d-17c6-4f16-8fe4-1a018f5ed00d	

SVV_DEFAULT_PRIVILEGES

使用 SVV_DEFAULT_PRIVILEGES 在 Amazon Redshift 叢集中檢視使用者可存取的預設權限。

SVV下列使用者可以看到 _DEFAULT_PRIVILEGES :

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到授予他們的預設許可。

資料表欄

欄名稱	資料類型	描述
schema_name	text	結構描述的名稱。
object_type	text	物件的類型。可能的值為 RELATION、FUNCTION或 PROCEDURE。
owner_id	integer	擁有者 ID。可能的值是 user ID。
owner_name	text	擁有者的名稱。
owner_type	text	擁有者類型。可能的值是 user。
privilege_type	text	權限類型。可能的值為 INSERT、SELECT、UPDATE、DELETE、RULE、REFERENCETRIGGER、DROP和 EXECUTE。
grantee_id	integer	承授者 ID。可能的值是 user ID、role ID 和 group ID。
grantee_type	text	承授者類型。可能的值是 user、role 和 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組類型來說，永遠是 false。

範例查詢

下列範例會傳回 SVV_DEFAULT_ 的輸出PRIVILEGES。

```
SELECT * from svv_default_privileges;
```

```

schema_name | object_type | owner_id | owner_name | owner_type | privilege_type
| grantee_id | grantee_name | grantee_type | admin_option
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+
public | RELATION | 106 | u1 | user | UPDATE
| 107 | u2 | user | f |
public | RELATION | 106 | u1 | user | SELECT
| 107 | u2 | user | f |

```

SVV_DISKUSAGE

Amazon Redshift 會加入 PERM SVV_DISKUSAGE STVTBL 和 _ 資料表來建立 STV_BLOCKLIST 系統檢視。SVV_DISKUSAGE 檢視包含資料庫中資料表的資料分配資訊。

使用彙總查詢搭配 SVV_DISKUSAGE，如下列範例所示，以判斷每個資料庫、資料表、層或資料欄配置的磁碟區塊數量。每個資料區塊都使用 1 MB。您也可以使用 [STV_PARTITIONS](#) 來檢視磁碟使用率的摘要資訊。

SVV 只有超級使用者可以看到 _DISKUSAGE。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
db_id	integer	資料庫 ID。
name	character (72)	資料表名稱。
分割	integer	配置至資料表的資料分割。
col	integer	欄位的以零為起始的索引。您建立的每個資料表都有三個隱藏的資料欄附加在其中：INSERT_XID、DELETE_XID 和 ROW_ID (OID)。包含 3 個使用者定義欄位的資料表包含 6 個實際欄位，使用者定義的欄位的內部編號為 0、1 及 2。在此範例中 XID，INSERT_XID、DELETE_ 和 ROW_ID 資料欄分別編號為 3、4 和 5。
tbl	integer	表格 ID。

欄名稱	資料類型	描述
blocknum	integer	資料區塊的 ID。
num_values	integer	區塊所包含的值的數量。
minvalue	bigint	區塊所包含的值下限。
maxvalue	bigint	區塊所包含的值上限。
sb_pos	integer	磁碟上的超級區塊位置的內部識別碼。
pinned	integer	區塊是否固定至記憶體做為預載的一部分。0 = false ; 1 = true。預設為 false。
on_disk	integer	區塊是否已自動儲存於磁碟。0 = false ; 1 = true。預設為 false。
modified	integer	區塊是否已修改。0 = false ; 1 = true。預設為 false。
hdr_modified	integer	區塊標頭是否已修改。0 = false ; 1 = true。預設為 false。
unsorted	integer	區塊是否未排序。0 = false ; 1 = true。預設為 true。
tombstone	integer	供內部使用。
preferred_diskno	integer	區塊應處於開啟狀態的磁碟數量 (無論磁碟是否故障)。一旦磁碟修復，區塊將移回該磁碟。
temporary	integer	無論區塊是否包含暫存資料，例如來自暫存資料表或中繼查詢結果。0 = false ; 1 = true。預設為 false。
newblock	integer	指出區塊是否是新的 (true) 或不曾遞交至磁碟 (false)。0 = false ; 1 = true。

範例查詢

SVV_DISKUSAGE 每個配置的磁碟區塊包含一列，因此選取所有資料列的查詢可能會傳回非常大量的資料列。我們建議僅使用 SVV_彙總查詢DISKUSAGE。

傳回已配置給USERS資料表（EMAIL資料欄）中資料欄 6 的區塊數量上限：

```
select db_id, trim(name) as tablename, max(blocknum)
from svv_diskusage
where name='users' and col=6
group by db_id, name;
```

```
db_id | tablename | max
-----+-----+-----
175857 | users      | 2
(1 row)
```

下列查詢會傳回大型 10 欄資料表中所有資料欄的類似結果，稱為 SALESNEW。(欄位 10 到 12 的最後三列會供隱藏的中繼資料欄位使用。)

```
select db_id, trim(name) as tablename, col, tbl, max(blocknum)
from svv_diskusage
where name='salesnew'
group by db_id, name, col, tbl
order by db_id, name, col, tbl;
```

```
db_id | tablename | col | tbl | max
-----+-----+-----+-----+-----
175857 | salesnew  | 0  | 187605 | 154
175857 | salesnew  | 1  | 187605 | 154
175857 | salesnew  | 2  | 187605 | 154
175857 | salesnew  | 3  | 187605 | 154
175857 | salesnew  | 4  | 187605 | 154
175857 | salesnew  | 5  | 187605 | 79
175857 | salesnew  | 6  | 187605 | 79
175857 | salesnew  | 7  | 187605 | 302
175857 | salesnew  | 8  | 187605 | 302
175857 | salesnew  | 9  | 187605 | 302
175857 | salesnew  | 10 | 187605 | 3
175857 | salesnew  | 11 | 187605 | 2
175857 | salesnew  | 12 | 187605 | 296
(13 rows)
```

SVV_EXTERNAL_COLUMNS

使用 SVV_EXTERNAL_COLUMNS 檢視外部資料表中資料欄的詳細資訊。也使用 SVV_EXTERNAL_COLUMNS 進行跨資料庫查詢，以檢視使用者可存取之未連線資料庫資料表中所有資料欄的詳細資訊。

SVV所有使用者都可看見 _EXTERNAL_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
redshift_database_name	text	本機 Amazon Redshift 資料庫的名稱。
結構描述名稱	text	外部資料表之 Amazon Redshift 外部結構描述的名稱。
資料表名稱	text	外部資料表名稱。
columnname	text	欄位的名稱。
external_type	text	欄的資料類型。
columnnum	integer	外部資料欄號碼，從 1 開始。
part_key	integer	索引鍵的順序 (如果欄位為分割區索引鍵)。如果資料欄不是分割區，值為 0。
is_nullable	text	定義一個資料欄是否可為 null。有些值是 true、false 或 "" 空字串，代表沒有資訊。

SVV_EXTERNAL_DATABASES

使用 SVV_EXTERNAL_DATABASES 檢視外部資料庫的詳細資訊。

SVV所有使用者都可看見 `_EXTERNAL_DATABASES`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
eskind	integer	資料庫外部目錄的類型； 1 表示資料目錄， 2 表示 Hive 中繼存放區。
esoptions	text	資料庫所在的目錄詳細資訊。
databasename	text	外部目錄中資料庫之名稱。
location	text	資料庫的位置。
parameters	text	資料庫參數。

SVV_EXTERNAL_PARTITIONS

使用 SVV_EXTERNAL_PARTITIONS 檢視外部資料表中分割區的詳細資訊。

SVV所有使用者都可看見 `_EXTERNAL_PARTITIONS`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
結構描述名稱	text	Amazon Redshift 外部結構描述的名稱，適用於內含指定分割區的外部資料表。
資料表名稱	text	外部資料表名稱。
values	text	分割區的值。
location	text	分割區的位置。欄大小限制為 128 個字元。將會截斷較長的值。

欄名稱	資料類型	描述
input_format	text	輸入格式。
output_format	text	輸出格式。
serialization_lib	text	序列化程式庫。
serde_parameters	text	SerDe 參數。
已壓縮	integer	此值指出分割區是否已壓縮； 1 表示已壓縮， 0 表示未壓縮。
parameters	text	分割區屬性。

SVV_EXTERNAL_SCHEMAS

使用 SVV_EXTERNAL_SCHEMAS 檢視外部結構描述的相關資訊。如需詳細資訊，請參閱[CREATE EXTERNAL SCHEMA](#)。

SVV 所有使用者都可看見 _EXTERNAL_SCHEMAS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
esoid	oid	外部結構描述 ID。
eskind	smallint	外部結構描述的外部目錄類型：1 表示資料目錄，2 表示 Hive 中繼存放區，3 表示 Aurora PostgreSQL 或 Amazon RDS Postgre 的聯合查詢 SQL，4 表示本機 Amazon Redshift 資料庫的結構描述，5 表示遠端 Amazon Redshift 資料庫的結構描述，6 表示系統資料表的結構描述，8 表示遠端 MySQL 資料庫的結構描述，9 表示 Amazon Kinesis 資料串流的結構描述，和 10 表示 Amazon Managed Streaming for Apache Kafka 資料串流。

欄名稱	資料類型	描述
結構描述名稱	name	外部結構描述名稱。
esowner	integer	外部結構描述擁有者的使用者 ID。
databasename	text	外部資料庫名稱。
esoptions	text	外部結構描述選項。

範例

下列範例顯示外部結構描述的詳細資訊。

```
select * from svv_external_schemas;

esoid | eskind | schemaname | esowner | databasename | esoptions
-----+-----+-----+-----+-----
100133 |      1 | spectrum  |      100 | redshift     | {"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
```

SVV_EXTERNAL_TABLES

使用 SVV_EXTERNAL_TABLES 檢視外部資料表的詳細資訊；如需詳細資訊，請參閱 [CREATE EXTERNAL SCHEMA](#)。也使用 SVV_EXTERNAL_TABLES 進行跨資料庫查詢，以檢視使用者可存取之未連線資料庫上所有資料表的中繼資料。

SVV 所有使用者都可看見 _EXTERNAL_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
redshift_database_name	text	本機 Amazon Redshift 資料庫的名稱。
結構描述名稱	text	外部資料表之 Amazon Redshift 外部結構描述的名稱。
資料表名稱	text	外部資料表名稱。
tabletype	text	資料表的類型。有些值是 TABLE、VIEW、MATERIALIZED VIEW 或 "" 空白字串，不代表任何資訊。
location	text	資料表的位置。
input_format	text	輸入格式
output_format	text	輸出格式。
serialization_lib	text	序列化程式庫。
serde_parameters	text	SerDe 參數。
已壓縮	integer	此值指出資料表是否已壓縮； 1 表示已壓縮， 0 表示未壓縮。
parameters	text	資料表屬性。

範例

下列範例顯示詳細資料 `svv_external_tables`，其具有聯合查詢所使用之外部結構描述上的述詞。

```
select schemaname, tablename from svv_external_tables where schemaname = 'apg_tpch';
schemaname | tablename
-----+-----
```

```

apg_tpch | customer
apg_tpch | lineitem
apg_tpch | nation
apg_tpch | orders
apg_tpch | part
apg_tpch | partsupp
apg_tpch | region
apg_tpch | supplier
(8 rows)

```

SVV_FUNCTION_PRIVILEGES

使用 SVV_FUNCTION_PRIVILEGES 檢視明確授予目前資料庫中使用者、角色和群組的函數許可。

SVV下列使用者可以看到 _FUNCTION_PRIVILEGES :

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定函數所在的命名空間名稱。
function_name	text	函數的名稱。
argument_types	text	代表函數輸入引數類型的字串。
privilege_type	text	許可的類型。可能的值為 EXECUTE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。

欄名稱	資料類型	描述
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_FUNCTION_ 的結果PRIVILEGES。

```
SELECT
  namespace_name, function_name, argument_types, privilege_type, identity_name, identity_type, admin_option
FROM svv_function_privileges
WHERE identity_name IN ('role1', 'reguser');
```

```
namespace_name | function_name | argument_types | privilege_type |
identity_name | identity_type | admin_option
-----+-----+-----+-----+
+-----+-----+-----+-----+
public        | test_func1   | integer        | EXECUTE        |
role1         | role         | False          |                 |
public        | test_func2   | integer, character varying | EXECUTE        |
reguser       | user         | False          |                 |
```

SVV_GEOGRAPHY_COLUMNS

使用 SVV_GEOGRAPHY_COLUMNS 檢視資料倉儲中的GEOGRAPHY資料欄清單。此欄清單包含來自資料共用的欄。

SVV所有使用者都可看見 _GEOGRAPHY_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
f_table_catalog	varchar(128)	具有資料GEOGRAPHY欄之資料表所在的資料庫名稱。
f_table_schema	varchar(128)	具有資料GEOGRAPHY欄之資料表所在的結構描述名稱。
f_table_name	varchar(128)	GEOGRAPHY 資料欄存在的資料表名稱。
f_geography_column	varchar(128)	GEOGRAPHY 資料欄的名稱。
coord_dimension	integer	GEOGRAPHY 資料的維度。
srid	integer	GEOGRAPHY 資料的空間參考系統識別碼 (SRID)。
type	varchar(128)	空間地理資料類型名稱。

範例查詢

下列範例顯示 SVV_GEOGRAPHY_ 的結果COLUMNS。

```
SELECT * FROM svv_geography_columns;

f_table_catalog | f_table_schema | f_table_name | f_geography_column |
coord_dimension | srid | type
-----+-----+-----+-----+
+-----+-----+-----+-----+
dev           | public       | spatial_test | test_geography     | 2
| 0 | GEOGRAPHY
```

SVV_GEOMETRY_COLUMNS

使用 SVV_GEOMETRY_COLUMNS 檢視資料倉儲中的 GEOMETRY 資料欄清單。此欄清單包含來自資料共用的欄。

SVV 所有使用者都可看見 _GEOMETRY_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
f_table_catalog	varchar(128)	具有資料 GEOMETRY 欄之資料表所在的資料庫名稱。
f_table_schema	varchar(128)	具有資料 GEOMETRY 欄之資料表所在的結構描述名稱。
f_table_name	varchar(128)	GEOMETRY 資料欄存在的資料表名稱。
f_geometry_column	varchar(128)	GEOMETRY 資料欄的名稱。
coord_dimension	integer	GEOMETRY 資料的維度。
srid	integer	olumn 的空間參考系統識別碼 GEOMETRY (SRID)。
type	varchar(128)	空間幾何類型名稱。

範例查詢

下列範例顯示 SVV_GEOMETRY_ 的結果 COLUMNS。

```
SELECT * FROM svv_geometry_columns;
```

```
f_table_catalog | f_table_schema | f_table_name | f_geometry_column |
coord_dimension | srid | type
```

```

-----+-----+-----+-----
+-----+-----+-----
dev          | public          | accomodations | shape          | 2
  | 0      | GEOMETRY
dev          | public          | zipcode       | wkb_geometry   | 2
  | 0      | GEOMETRY

```

SVV_IAM_PRIVILEGES

使用 SVV_IAM_PRIVILEGES 檢視使用者、角色和群組的明確授予IAM權限。

SVV下列使用者可以看到 _IAM_PRIVILEGES :

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取的项目。

資料表欄

欄名稱	資料類型	描述
iam_arn	text	命名空間的名稱。
command_type	text	權限類型。可能的值為 COPY、MODEL、UNLOAD CREATE 或 EXTERNAL FUNCTION。
identity_id	integer	身分 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分名稱。
identity_type	text	身分類型。可能的值是 user、role、group 或 public。

範例查詢

下列範例顯示 SVV_IAM_ 的結果PRIVILEGES。

```
SELECT * from SVV_IAM_PRIVILEGES ORDER BY IDENTITY_ID;
 iam_arn          | command_type | identity_id | identity_name | identity_type
-----+-----+-----+-----+-----
 default-aws-iam-role | COPY          |          0 | public        | public
 default-aws-iam-role | UNLOAD        |          0 | public        | public
 default-aws-iam-role | CREATE MODEL  |          0 | public        | public
 default-aws-iam-role | EXFUNC        |          0 | public        | public
 default-aws-iam-role | COPY          |         106 | u1            | user
 default-aws-iam-role | UNLOAD        |         106 | u1            | user
 default-aws-iam-role | CREATE MODEL  |         106 | u1            | user
 default-aws-iam-role | EXFUNC        |         106 | u1            | user
 default-aws-iam-role | COPY          |       118413 | r1            | role
 default-aws-iam-role | UNLOAD        |       118413 | r1            | role
 default-aws-iam-role | CREATE MODEL  |       118413 | r1            | role
 default-aws-iam-role | EXFUNC        |       118413 | r1            | role
(12 rows)
```

SVV_IDENTITY_PROVIDERS

SVV_IDENTITY_PROVIDERS 檢視會傳回身分提供者的名稱和其他屬性。如需如何建立身分提供者的相關資訊，請參閱[CREATE IDENTITY PROVIDER](#)。

SVV只有超級使用者可以看到 IDENTITY_PROVIDERS。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
uid	integer	已註冊身分提供者的唯一 ID。
name	text	身分提供者名稱。
type	text	身分提供者類型。

欄名稱	資料類型	描述
instanceid	text	相同類型的執行個體之間的唯一差異化因素。
namespc	text	身分提供者的命名空間前置詞。
params	text	具有身分提供者參數的JSON物件。
已啟用	bool	指出身分提供者是否已啟用。

範例查詢

若要檢視身分提供者內容，請在建立身分提供者之後執行如下查詢。

```
SELECT name, type, instanceid, namespc, params, enabled
FROM svv_identity_providers
ORDER BY 1;
```

範例輸出包含參數描述。

```

      name          | type |          instanceid          | namespc |
                  |      |                               |         |
                  |      |                               |         |
                  |      |                               |         |
                  |      |                               |         |
                  |      |                               |         |
                  |      |                               |         |
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
rs5517_azure_idp | azure | e40d4bb2-7670-44ae-bfb8-5db013221d73 | abc      |
{"issuer":"https://login.microsoftonline.com/e40d4bb2-7670-44ae-bfb8-5db013221d73/
v2.0", "client_id":"871c010f-5e61-4fb1-83ac-98610a7e9110", "client_secret":,
 "audience":["https://analysis.windows.net/powerbi/connector/AmazonRedshift", "https://
analysis.windows.net/powerbi/connector/AWSRDS"]} | t
(1 row)
```

SVV_INTEGRATION

SVV_INTEGRATION 顯示整合組態的詳細資訊。

SVV只有超級使用者可以看到 _INTEGRATION。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需零ETL整合的相關資訊，請參閱[使用零ETL整合](#)。

資料表欄

欄名稱	資料類型	描述
integration_id	character (128)	與整合關聯的識別碼。
target_database	character (128)	Amazon Redshift 中接收整合資料的資料庫。
source	character (128)	整合的來源資料。可能的類型包括 MySQL 和 PostgreSQL 。
state	character (128)	整合的狀態。可能的值包括 PendingDbConnectState、SchemaDiscoveryState、CdcRefreshState 與 ErrorState 。
current_lag	bigint	整合的來源與目的地之間的目前延遲時間 (毫秒)。
last_replicated_checkpoint	character (128)	最後複寫的檢查點。
total_tables_replicated	integer	目前處於已複寫狀態的資料表總數。
total_tables_failed	integer	目前處於失敗狀態的資料表總數。
creation_time	timestamp	建立整合的時間 (UTC)。其定義為從整合建立目標資料庫的時間。
refresh_interval	integer	大約的時間間隔，以秒為單位，將資料從零ETL來源重新整理至目標資料庫。

欄名稱	資料類型	描述
source_database	character (128)	來源資料庫的名稱。

範例查詢

下列SQL命令會顯示目前定義的整合。

```
select * from svv_integration;
```

```

          integration_id          | target_database | source |      state
| current_lag |      last_replicated_checkpoint      | total_tables_replicated |
total_tables_failed |      creation_time          | refresh_interval | source_database
-----+-----+-----+-----
+-----+-----+-----+-----
+-----+-----+-----+-----
+-----+-----+-----+-----
          99108e72-1cfd-414f-8cc0-0216acefac77 |      perfdb      | MySQL | CdcRefreshState |
56606106 | {"txn_seq":9834,"txn_id":126597515} |      152      |
0          | 2023-09-19 21:05:27.520299|      720      | + mysourceetl

```

SVV_INTEGRATION_TABLE_STATE

SVV_INTEGRATION_TABLE_STATE 顯示資料表層級整合資訊的詳細資訊。

SVV_INTEGRATION_TABLE_STATE 只有超級使用者可以看到。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需詳細資訊，請參閱[使用零ETL整合](#)。

資料表欄

欄名稱	資料類型	描述
integration_id	character(128)	與整合關聯的識別碼。
target_database	character(128)	Amazon Redshift 資料庫的名稱。

欄名稱	資料類型	描述
schema_name	character(128)	Amazon Redshift 結構描述的名稱。
table_name	character(128)	資料表的名稱。
table_state	character(128)	資料表的狀態。可能值為 Synced、Failed、Deleted、ResyncRequired 和 ResyncInitiated 。
table_last_replicated_checkpoint	character(128)	目前同步的記錄座標。
reason	character(256)	最後狀態轉換的原因。常見的原因可能是資料表中不支援的資料類型、資料表沒有主索引鍵。若要進一步了解如何疑難排解常見問題，請參閱 疑難排解 Amazon Redshift 中的零ETL 整合 。
last_updated_timestamp	沒有時區的時間戳記	資料表上次更新的時間（UTC）。

範例查詢

下列SQL命令會顯示整合的日誌。

```
select * from svv_integration_table_state;

      integration_id          | target_database | schema_name |      table_name
| Table_state |table_last_replicated_checkpoint | reason | last_updated_timestamp
-----+-----+-----
+-----+-----+-----
4798e675-8f9f-4686-b05f-92c538e19629 | sample_test2  | sample     |
SampleTestChannel | Synced       | {"txn_seq":3,"txn_id":3122} |
2023-05-12 12:40:30.656625
```

SVV_INTERLEAVED_COLUMNS

使用 SVV_INTERLEAVED_COLUMNS 檢視協助判斷使用交錯排序索引鍵的資料表是否應該使用重新索引 [VACUUM REINDEX](#)。如需如何判斷執行頻率 VACUUM 以及執行時間的詳細資訊 VACUUMREINDEX，請參閱 [將真空時間降到最低](#)。

SVV 只有超級使用者可以看到 INTERLEAVED_COLUMNS。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
tbl	integer	表格 ID。
col	integer	欄位的以零為起始的索引。
interleaved_skew	numeric (9,2)	比例會指出在資料表之交錯排序索引鍵欄位中存在的扭曲量。值 1.00 表示沒有扭曲，值越大表示扭曲越嚴重。具有大型偏移的資料表應該使用 VACUUMREINDEX 命令重新索引。
last_reindex	timestamp	指定資料表上次 VACUUMREINDEX 執行的時間。如果 NULL 資料表從未重新索引，或者基礎系統日誌資料表 STL_ 自上次重新索引以來已輪換 VACUUM，則此值為。

範例查詢

若要識別可能需要重新建立索引的資料表，請執行下列查詢。

```
select tbl as tbl_id, stv_tbl_perm.name as table_name,
col, interleaved_skew, last_reindex
from svv_interleaved_columns, stv_tbl_perm
where svv_interleaved_columns.tbl = stv_tbl_perm.id
and interleaved_skew is not null;
```

```
tbl_id | table_name | col | interleaved_skew | last_reindex
-----+-----+-----+-----+-----
100068 | lineorder  | 0   | 3.65             | 2015-04-22 22:05:45
100068 | lineorder  | 1   | 2.65             | 2015-04-22 22:05:45
```

```

100072 | customer   | 0 |          | 1.65 | 2015-04-22 22:05:45
100072 | lineorder  | 1 |          | 1.00 | 2015-04-22 22:05:45
(4 rows)

```

SVV_LANGUAGE_PRIVILEGES

使用 SVV_LANGUAGE_PRIVILEGES 檢視明確授予目前資料庫中使用者、角色和群組的語言許可。

SVV 下列使用者可以看到 _LANGUAGE_PRIVILEGES：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
language_name	text	語言的名稱。
privilege_type	text	許可的類型。可能的值為 USAGE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_LANGUAGE_ 的結果 PRIVILEGES。

```

SELECT language_name, privilege_type, identity_name, identity_type, admin_option FROM
svv_language_privileges

```

```
WHERE identity_name IN ('role1', 'reguser');
```

language_name	privilege_type	identity_name	identity_type	admin_option
exfunc	USAGE	reguser	user	False
exfunc	USAGE	role1	role	False
plpythonu	USAGE	reguser	user	False

SVV_MASKING_POLICY

使用 SVV_MASKING_POLICY 檢視叢集上建立的所有遮罩政策。

只有具有 [sys:secadmin](#) 角色的超級使用者和使用者才能檢視 SVV_MASKING_POLICY。一般使用者將看到 0 列。

資料表欄

欄名稱	資料類型	描述
policy_database	text	建立遮罩政策所在的資料庫名稱。
policy_name	text	遮罩政策的名稱。
input_columns	text	CREATE POLICY 陳述式子 WITH 句中提供的屬性。
policy_expression	text	政策中使用的遮罩運算式。
policy_modified_by	text	上次修改政策的使用者名稱。
policy_modified_time	timestamp	政策建立或上次修改的時間戳記。

SVV_ML_MODEL_INFO

關於機器學習模型目前狀態的狀態資訊。

SVV 所有使用者都可看見 _ML_MODEL_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	char(128)	模型的資料庫。
schema_name	char(128)	模型的結構描述。
user_name	char(128)	模型的擁有者。
model_name	char(128)	模型的名稱。
life_cycle	char(20)	模型的生命週期狀態。
is_refreshable	integer	模型的狀態，指出如果訓練查詢中的原始資料表和欄仍然存在且使用者仍具有這些項目的許可，則模型是否可重新整理。可能的值是：1 (可重新整理) 和 0 (無法重新整理)。
model_state	char(128)	膜性的目前狀態。

範例查詢

下列查詢會顯示機器學習模型的目前狀態。

```
SELECT schema_name, model_name, model_state
FROM svv_ml_model_info;
```

```

schema_name |          model_name          |          model_state
-----+-----+-----
public      | customer_churn_auto_model    | Train Model On SageMaker In Progress
public      | customer_churn_xgboost_model | Model is Ready
(2 row)
```

SVV_ML_MODEL_PRIVILEGES

使用 SVV_ML_MODEL_PRIVILEGES 檢視明確授予叢集中使用者、角色和群組的機器學習模型許可。

SVV下列使用者可以看到 _ML_MODEL_PRIVILEGES：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	所指定機器學習模型所在的命名空間名稱。
model_name	text	機器學習模型的名稱。
model_version	integer	模型的版本號碼。
privilege_type	text	許可的類型。可能的值為 EXECUTE。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_ML_MODEL_ 的結果 PRIVILEGES。

```
SELECT
  namespace_name, model_name, model_version, privilege_type, identity_name, identity_type, admin_option
FROM svv_ml_model_privileges
```

```
WHERE model_name = 'test_model';
```

```

namespace_name | model_name | model_version | privilege_type | identity_name |
identity_type | admin_option
-----+-----+-----+-----+-----
+-----+-----+
      public   | test_model |          1    | EXECUTE       | reguser      |
user          | False
      public   | test_model |          1    | EXECUTE       | role1        |
role          | False

```

SVV_MV_DEPENDENCY

SVV_MV_DEPENDENCY 資料表顯示 Amazon Redshift 內其他具體化視觀表的具體化視觀表相依性。

如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

SVV 所有使用者都可看見 _MV_DEPENDENCY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	char(128)	包含所指定具體化視觀表的資料庫。
schema_name	char(128)	具體化視觀表的結構描述。
name	char(128)	具體化視觀表的名稱。
dependent_database_name	char(128)	此具體化視觀表相依的具體化視觀表資料庫。
dependent_schema_name	char(128)	此具體化視觀表所依賴的具體化視觀表結構描述。
dependent_name	char(128)	此具體化視觀表所依賴的具體化視觀表的名稱。

範例查詢

下列查詢會傳回輸出列，指出具體化視觀表 `mv_over_foo` 使用其定義中的具體化視觀表 `mv_foo` 做為相依性。

```
CREATE SCHEMA test_ivm_setup;
CREATE TABLE test_ivm_setup.foo(a INT);
CREATE MATERIALIZED VIEW test_ivm_setup.mv_foo AS SELECT * FROM test_ivm_setup.foo;
CREATE MATERIALIZED VIEW test_ivm_setup.mv_over_foo AS SELECT * FROM
  test_ivm_setup.mv_foo;

SELECT * FROM svv_mv_dependency;

  database_name | schema_name          | name          | dependent_database_name |
  dependent_schema_name | dependent_name
-----+-----+-----+-----
+-----+-----+-----+-----
dev           | test_ivm_setup       | mv_over_foo | dev |
test_ivm_setup | mv_foo
```

SVV_MV_INFO

SVV_MV_INFO 資料表包含每個具體化視觀表的列，資料是否過時，以及狀態資訊。

如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

SVV 所有使用者都可看見 `_MV_INFO`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>database_name</code>	char(128)	包含具體化視觀表的資料庫。
<code>schema_name</code>	char(128)	資料庫的結構描述。
<code>user_name</code>	char(128)	擁有具體化視觀表的使用者。
<code>name</code>	char(128)	具體化視觀表名稱。

欄名稱	資料類型	描述
is_stale	char(1)	t 表示具體化視觀表已過時。「過時」的具體化視觀表是基底資料表已更新，但具體化視觀表尚未重新整理時的具體化視觀表。如果重新整理自上一次重新啟動以來都並未執行，這項資訊可能會不準確。
state	integer	<p>具體化視觀表的狀態如下：</p> <ul style="list-style-type: none"> • 0 - 重新整理時，會完全重新計算具體化視觀表。 • 1 - 具體化視觀表為累加式。 • 101 - 具體化視觀表因遭到卸除的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 102 - 具體化視觀表因為資料欄類型變更而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 103 - 具體化視觀表因為重新命名的資料表而無法重新整理。 • 104 - 具體化視觀表因為重新命名的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 105 - 具體化視觀表因為重新命名的結構描述而無法重新整理。
autorewrite	char(1)	t 表示具體化視觀表適用於自動重新寫入查詢。
autorefresh	char(1)	t 表示可以自動重新整理具體化視觀表。

範例查詢

若要檢視所有具體化視觀表的狀態，請執行下列查詢。

```
select * from svv_mv_info;
```

此查詢傳回下列範例輸出。

```

database_name |      schema_name      | user_name | name | is_stale | state |
autorefresh  | autorewrite
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
dev          | test_ivm_setup        | catch-22 | mv   | f        | 1    |
      1 |          0
dev          | test_ivm_setup        | lotr     | old_mv | t        | 1    |
      0 |          1

```

SVV_QUERY_INFLIGHT

使用 SVV_QUERY_INFLIGHT 檢視來判斷目前在資料庫上執行的查詢。此檢視會聯合 [STV_INFLIGHT](#) 至 [STL_QUERYTEXT](#)。SVV_QUERY_INFLIGHT 不會顯示領導者節點查詢。如需詳細資訊，請參閱 [僅限領導節點函數](#)。

SVV 所有使用者都可看見 _QUERY_INFLIGHT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
分割	integer	查詢執行時所在的分割。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查

欄名稱	資料類型	描述
		詢，則此值通常會保持不變。您可以使用此資料欄來聯結至 STL_ERROR 資料表。
starttime	timestamp	查詢開始的時間。
suspended	integer	查詢是否已遭停用：0 = false；1 = true。
text	character(200)	查詢文字，以 200 個字元的方式增量。
sequence	integer	查詢陳述式區段的序列數。

範例查詢

下面的範例輸出顯示目前正在執行的兩個查詢，即 SVV_QUERY_INFLIGHT 查詢本身和查詢 428，其在資料表中分為三列。(在此範例輸出中開始時間和陳述式欄位遭到截斷。)

```
select slice, query, pid, starttime, suspended, trim(text) as statement, sequence
from svv_query_inflight
order by query, sequence;
```

```
slice|query| pid |      starttime      |suspended| statement | sequence
-----+-----+-----+-----+-----+-----+-----
1012 | 428 | 1658 | 2012-04-10 13:53:... |      0 | select ... |      0
1012 | 428 | 1658 | 2012-04-10 13:53:... |      0 | enueid ... |      1
1012 | 428 | 1658 | 2012-04-10 13:53:... |      0 | atname,... |      2
1012 | 429 | 1608 | 2012-04-10 13:53:... |      0 | select ... |      0
(4 rows)
```

SVV_QUERY_STATE

使用 SVV_QUERY_STATE 檢視目前執行查詢執行時間的相關資訊。

SVV_QUERY_STATE 檢視包含 STV_EXEC_STATE 資料表的資料子集。

SVV所有使用者都可看見 _QUERY_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

Note

只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
seg	integer	正在執行中的查詢區段數。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	正在執行中的查詢步驟數。步驟是查詢執行期的最小單位。每個步驟表示分散的工作單位，例如掃描資料表、傳回結果或排序資料。
maxtime	間隔	此步驟執行的時間量上限 (微秒)。
avgtime	間隔	此步驟執行的平均時間 (微秒)。
rows	bigint	執行中步驟產生的列數。
bytes	bigint	執行中步驟產生的位元組數。
cpu	bigint	供內部使用。
memory	bigint	供內部使用。
rate_row	double precision	Rows-per-second 速率，計算方式為將資料列加總並除以查詢開始到目前時間的秒數。
rate_byte	double precision	Bytes-per-second 速率，計算方式為將位元組加總並除以查詢開始到目前時間的秒數。
label	character(25)	查詢標籤：步驟的名稱，例如 scan 或 sort。

欄名稱	資料類型	描述
is_diskbased	character(1)	此查詢步驟是否以磁碟型操作方式執行：true (t) 或 false (f)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。
num_partitions	integer	雜湊資料表在雜湊步驟期間已分割的分割區數。此欄中的正數並不表示雜湊步驟以磁碟式操作方式執行。檢查 IS_DISKBASED 欄中的值，查看雜湊步驟是否為磁碟型。
is_rrscan	character(1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。預設為 false (f)。
is_delayed_scan	character(1)	若為 true (t)，表示已在該步驟上使用延遲的掃描。預設為 false (f)。

範例查詢

依步驟判斷查詢處理時間

下列查詢顯示查詢 ID 為 279 之查詢的每個步驟執行多久，以及 Amazon Redshift 處理的列多寡：

```
select query, seg, step, maxtime, avgtime, rows, label
from svv_query_state
where query = 279
order by query, seg, step;
```

此查詢會擷取查詢 279 的相關處理資訊，如下範例輸出所示：

```
query | seg | step | maxtime | avgtime | rows | label
-----+-----+-----+-----+-----+-----+-----
279 | 3 | 0 | 1658054 | 1645711 | 1405360 | scan
279 | 3 | 1 | 1658072 | 1645809 | 0 | project
279 | 3 | 2 | 1658074 | 1645812 | 1405434 | insert
279 | 3 | 3 | 1658080 | 1645816 | 1405437 | distribute
279 | 4 | 0 | 1677443 | 1666189 | 1268431 | scan
279 | 4 | 1 | 1677446 | 1666192 | 1268434 | insert
279 | 4 | 2 | 1677451 | 1666195 | 0 | agg
```



```
(7 rows)
```

判斷磁碟上是否有任何正在執行的作用中查詢

以下查詢顯示作用中的查詢目前是否正在磁碟上執行：

```
select query, label, is_diskbased from svv_query_state
where is_diskbased = 't';
```

此範例輸出顯示作用中的查詢目前是否正在磁碟上執行：

```
query | label          | is_diskbased
-----+-----+-----
1025  | hash tbl=142  |          t
(1 row)
```

SVV_REDSHIFT_COLUMNS

使用 SVV_REDSHIFT_COLUMNS 檢視使用者可存取的所有資料欄清單。這組欄包括叢集上的欄，以及遠端叢集所提供資料共用中的欄。

SVV 所有使用者都可看見 _REDSHIFT_COLUMNS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	包含欄的資料表所在之資料庫的名稱。
schema_name	varchar(128)	資料表的結構描述名稱。
table_name	varchar(128)	資料表的名稱。
column_name	varchar(128)	欄的名稱。
ordinal_position	integer	資料表中欄位的位置。
data_type	varchar(32)	欄的資料類型。

欄名稱	資料類型	描述
column_default	varchar(4000)	欄位的預設值。
is_nullable	varchar(3)	定義欄是否可為 null 的值。可能的值是 yes、no 或 "" (表示沒有資訊的空字串)。
編碼	varchar(128)	欄的編碼類型。
distkey	boolean	如果此欄是資料表的分散索引鍵，則為 true，否則為 false。
sortkey	integer	<p>指定排序索引鍵中欄順序的值。</p> <p>如果資料表使用的是複合排序索引鍵，則排序索引鍵部分的所有欄位會有正值，指示排序索引鍵中欄位的位置。</p> <p>如果資料表使用交錯排序索引鍵，那麼作為排序索引鍵一部分的每一欄都會有交替為正或負的值。在這裡，絕對值指出欄在排序索引鍵的位置。</p> <p>若 sortkey 為 0，則欄不是排序索引鍵的一部分。</p>
column_acl	varchar(128)	字串，定義欄之指定使用者或使用者群組的許可。
備註	varchar(256)	備註。

範例查詢

下列範例傳回 SVV_REDSHIFT_ 的輸出 COLUMNS。

```

SELECT *
FROM svv_redshift_columns
WHERE database_name = 'tickit_db'
      AND TABLE_NAME = 'tickit_sales_redshift'
ORDER BY COLUMN_NAME,
      TABLE_NAME,
      database_name
LIMIT 5;

```

database_name	schema_name	table_name	column_name	ordinal_position	data_type	column_default	is_nullable	encoding	distkey	sortkey	column_acl	remarks
tickit_db	public	tickit_sales_redshift	buyerid	4	integer		NO	az64	False	0		
tickit_db	public	tickit_sales_redshift	commission	9	numeric	(8,2)	YES	az64	False	0		
tickit_db	public	tickit_sales_redshift	dateid	6	smallint		NO	none	False	1		
tickit_db	public	tickit_sales_redshift	eventid	5	integer		NO	az64	False	0		
tickit_db	public	tickit_sales_redshift	listid	2	integer		NO	az64	True	0		

SVV_REDSHIFT_DATABASES

使用 SVV_REDSHIFT_DATABASES 檢視使用者可存取的所有資料庫清單。這包括叢集上的資料庫，以及從遠端叢集所提供資料共用建立的資料庫。

SVV所有使用者都可看見 _REDSHIFT_DATABASES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	資料庫的名稱。
database_owner	integer	資料庫擁有者使用者 ID。

欄名稱	資料類型	描述
database_type	varchar(32)	資料庫的類型。可能的類型為本機或共用資料庫。
database_acl	varchar(128)	此資訊僅供內部使用。
database_options	varchar(128)	資料庫的屬性。
database_isolation_level	varchar(128)	資料庫的隔離層級。可能的值包括：Snapshot Isolation 和 Serializable。

範例查詢

下列範例會傳回 SVV_REDSHIFT_ 的輸出 DATABASES。

```
select database_name, database_owner, database_type, database_options,
       database_isolation_level
from   svv_redshift_databases;
```

```
database_name | database_owner | database_type | database_options |
database_isolation_level
```

```
-----+-----+-----+-----+-----
dev      | 1           | local        | NULL             |
Serializable
```

SVV_REDSHIFT_FUNCTIONS

使用 SVV_REDSHIFT_FUNCTIONS 檢視使用者可存取的所有函數清單。這組函數包括叢集上的函數，以及遠端叢集所提供資料共用中的函數。

SVV 所有使用者都可看見 _REDSHIFT_FUNCTIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	具有這些函數之叢集所在的資料庫名稱。
schema_name	varchar(128)	指定給定函數之結構描述的名稱。
function_name	varchar(128)	所指定函數的名稱。
function_type	varchar(128)	函數的類型。可能的值是 regular functions、aggregate functions 和 stored procedure S。
argument_type	varchar(512)	字串；代表函數輸入引數的類型。
result_type	varchar(128)	函數傳回值的資料類型。

範例查詢

下列範例會傳回 SVV_REDSHIFT_ 的輸出FUNCTIONS。

```
SELECT *
FROM svv_redshift_functions
WHERE database_name = 'tickit_db'
      AND SCHEMA_NAME = 'public'
ORDER BY function_name
LIMIT 5;
```

```
database_name | schema_name |      function_name      | function_type |
argument_type | result_type
-----+-----+-----+-----+
+-----+-----+
   tickit_db |   public   |  shared_function  | REGULAR FUNCTION | integer,
integer |   integer
```

SVV_REDSHIFT_SCHEMA_QUOTA

顯示資料庫中每個結構描述的配額和目前磁碟使用量。

SVV所有使用者都可看見 `_REDSHIFT_SCHEMA_QUOTA`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

查詢已佈建叢集或 Redshift 無伺服器工作群組時，可使用此檢視。

資料表欄

欄名稱	資料類型	描述
database_name	character(128)	包含結構描述的資料庫。
schema_name	character(128)	結構描述的名稱。
schema_owner	integer	結構描述擁有者的內部使用者 ID。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。

範例查詢

下列範例會顯示名為 `sales_schema` 之結構描述的配額和目前磁碟使用量。

```
SELECT TRIM(SCHEMA_NAME) "schema_name", QUOTA, disk_usage FROM
  svv_redshift_schema_quota
WHERE SCHEMA_NAME = 'sales_schema';
```

```
schema_name | quota | disk_usage
-----+-----+-----
sales_schema | 2048 | 30
```

SVV_REDSHIFT_SCHEMAS

使用 SVV_REDSHIFT_SCHEMAS 檢視使用者可存取的所有結構描述清單。這組結構描述包括叢集上的結構描述，以及遠端叢集所提供之資料庫的結構描述。

SVV所有使用者都可看見 _REDSHIFT_SCHEMAS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	指定結構描述所在的資料庫名稱。
schema_name	varchar(128)	命名空間或結構描述名稱。
schema_owner	integer	結構描述擁有者的內部使用者 ID。
schema_type	varchar(16)	結構描述的類型。可能的值是 shared 和 local schemas。
schema_acl	varchar(128)	字串，定義結構描述之指定使用者或使用者群組的許可。
schema_option	varchar(128)	結構描述的選項。

範例查詢

下列範例傳回 SVV_REDSHIFT_ 的輸出SCHEMAS。

```
SELECT *
FROM svv_redshift_schemas
WHERE database_name = 'tickit_db'
ORDER BY database_name,
        SCHEMA_NAME;
```

```
database_name |      schema_name      | schema_owner | schema_type | schema_acl |
              | schema_option
```

```

-----+-----+-----+-----
+-----
  tickit_db |      public      |      1      |      shared      |

```

SVV_REDSHIFT_TABLES

使用 SVV_REDSHIFT_TABLES 檢視使用者可存取的所有資料表清單。這組資料表包括叢集上的資料表，以及遠端叢集所提供資料共用中的資料表。

SVV所有使用者都可看見 _REDSHIFT_TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database_name	varchar(128)	指定之資料表所在資料庫的名稱。
schema_name	varchar(128)	資料表的結構描述名稱。
table_name	varchar(128)	資料表的名稱。
table_type	varchar(128)	資料表的類型。可能的值是 views 和 tables。
table_acl	varchar(128)	字串，定義資料表之指定使用者或使用者群組的許可。
備註	varchar(128)	備註。
table_owner	varchar(128)	資料表的擁有者。

範例查詢

下列範例傳回 SVV_REDSHIFT_ 的輸出TABLES。

```

SELECT *
FROM svv_redshift_tables

```



```
WHERE database_name = 'tickit_db' AND TABLE_NAME LIKE 'tickit_%'
ORDER BY database_name,
TABLE_NAME;
```

database_name	schema_name	table_name	table_type	table_acl	remarks	table_owner
tickit_db	public	tickit_category_redshift	TABLE			
tickit_db	public	tickit_date_redshift	TABLE			
tickit_db	public	tickit_event_redshift	TABLE			
tickit_db	public	tickit_listing_redshift	TABLE			
tickit_db	public	tickit_sales_redshift	TABLE			
tickit_db	public	tickit_users_redshift	TABLE			
tickit_db	public	tickit_venue_redshift	TABLE			

如果 table_acl 值為 null，表示沒有明確授予對應資料表的存取權限。

SVV_RELATION_PRIVILEGES

使用 SVV_RELATION_PRIVILEGES 檢視明確授予目前資料庫中使用者、角色和群組的關係（資料表和檢視）許可。

SVV 下列使用者可以看到 _RELATION_PRIVILEGES：

- 超級使用者
- 具有 SYSLOGACCESSUNRESTRICTED 許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。如需資料可見性的詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
namespace_name	text	指定關係所在的命名空間的名稱。
relation_name	text	關係的名稱。
privilege_type	text	許可的類型。可能的值為 INSERT、SELECT、UPDATE、REFERENCES、DELETE 或 DROP。
identity_id	integer	身分的 ID。可能的值是 user ID、role ID 或 group ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user、role、group 或 public。
admin_option	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 False。

範例查詢

下列範例顯示 SVV_RELATION_ 的結果PRIVILEGES。

```
SELECT
  namespace_name,relation_name,privilege_type,identity_name,identity_type,admin_option
  FROM svv_relation_privileges
WHERE relation_name = 'orders' AND privilege_type = 'SELECT';
```

```

namespace_name | relation_name | privilege_type | identity_name | identity_type |
admin_option
-----+-----+-----+-----+-----+
public         | orders       | SELECT        | reguser      | user         |
False
public         | orders       | SELECT        | role1        | role         |
False
```

SVV_RLS_APPLIED_POLICY

使用 SVV_RLS_APPLIED_POLICY 追蹤參考RLS受保護關係之查詢上的RLS政策應用程式。

SVV下列使用者可以看到 _RLS_APPLIED_POLICY：

- 超級使用者
- 具有 sys:operator 角色的使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

請注意，sys:secadmin 未授予此系統許可。

資料表欄

欄名稱	資料類型	描述
使用者名稱	text	執行查詢的使用者名稱。
query	integer	查詢的 ID。
xid	long	交易的內容。
pid	integer	執行查詢的領導者程序。
recordtime	time	記錄查詢的時間。
command	char(1)	套用RLS政策的命令。可能的值包括：k 代表未知、s 代表選取、u 代表更新、i 代表插入、y 代表公用程式，以及 d 代表刪除。
datname	text	列層級安全政策所連接之關係的資料庫名稱。
relschema	text	列層級安全政策所連接之關係的結構描述名稱。
relname	text	列層級安全政策所連接的關係名稱。
polname	text	連接至關係的列層級安全政策名稱。
poldefault	char(1)	連接至關係之列層級安全政策的預設設定。可能的值為如果已套用預設 false 政策則為 f (代表 false)，如果已套用預設 true 政策則為 t (代表 true)。

範例查詢

下列範例顯示 SVV_RLS_APPLIED_ 的結果POLICY。若要查詢 SVV_RLS_APPLIED_POLICY，您必須具有 ACCESSSYSTEMTABLE 許可。

```
-- Check what RLS policies were applied to the run query.
SELECT username, command, datname, relschema, relname, polname, poldefault
FROM svv_qls_applied_policy
WHERE datname = CURRENT_DATABASE() AND query = PG_LAST_QUERY_ID();

username | command | datname | relschema | relname | polname
| poldefault
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
molly | s | tickit_db | public | tickit_category_redshift |
policy_concerts |
```

SVV_RLS_ATTACHED_POLICY

使用 SVV_RLS_ATTACHED_POLICY 檢視在目前連線的資料庫中連接一或多個資料列層級安全政策的所有關係和使用者的清單。

只有具備 sys:secadmin 角色的使用者才能查詢此檢視。

資料表欄

欄名稱	資料類型	描述
relschema	text	列層級安全政策所連接之關係的結構描述名稱。
relname	text	列層級安全政策所連接的關係名稱。
relkind	text	物件的類型，例如 table。
polname	text	連接至關係之列層級安全政策的名稱。
grantor	text	連接此政策的使用者名稱。
grantee	text	此政策已連接之使用者或角色的名稱。
granteekind	text	grantee 的類型。可能的值是 user 或 role。

欄名稱	資料類型	描述
is_pol_on	boolean	指出資料表上的列層級安全政策是開啟或關閉的參數。可能的值是 true 和 false。
is_rls_on	boolean	指出資料表上的列層級安全是開啟或關閉的參數。可能的值是 true 和 false。
rls_conjunction_type	character (3)	指出關係是否將RLS政策與 and或 結合的 參數or。

範例查詢

下列範例顯示 SVV_RLS_ATTACHED_ 的結果POLICY。

```
--Inspect the policy in SVV_RLS_ATTACHED_POLICY
SELECT * FROM svv_rls_attached_policy;

 relschema |      relname      | relkind |      polname      | grantor | grantee
 | granteekind | is_pol_on | is_rls_on | rls_conjunction_type
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
public    | tickit_category_redshift | table | policy_concerts | bob    | analyst
 | role      | True    | True    | and
public    | tickit_category_redshift | table | policy_concerts | bob    | dbadmin
 | role      | True    | True    | and
```

SVV_RLS_POLICY

使用 SVV_RLS_POLICY 檢視在 Amazon Redshift 叢集上建立的所有資料列層級安全政策清單。

SVV所有使用者都可看見 _RLS_POLICY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
poldb	text	列層級安全政策所在資料庫的名稱。

欄名稱	資料類型	描述
polname	text	列層級安全政策的名稱。
polalias	text	政策定義中使用的資料表別名。
polatts	text	提供給政策定義的屬性。
polqual	text	CREATE POLICY 陳述式子 USING 句中提供的政策條件。
polenabed	boolean	政策是否全域開啟。
polmodifiedby	text	建立或最近修改政策的使用者名稱。
polmodifiedtime	timestamp	建立或上次修改政策時的時間戳記。

範例查詢

下列範例顯示 SVV_RLS_ 的結果 POLICY。

```
-- Create some policies.
CREATE RLS POLICY pol1 WITH (a int) AS t USING ( t.a IS NOT NULL );
CREATE RLS POLICY pol2 WITH (c varchar(10)) AS t USING ( c LIKE '%public%');

-- Inspect the policy in SVV_RLS_POLICY
SELECT * FROM svv_qls_policy;

 polddb | polname | polalias | polatts | polenabed | polmodifiedby | polmodifiedtime
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
my_db | pol1 | t | [{"colname":"a","type":"integer"}] | t | policy_admin | 2022-02-11
14:40:49
my_db | pol2 | t | [{"colname":"c","type":"character varying(10)"}] | t | policy_admin | 2022-02-11
14:41:28
```

SVV_RLS_RELATION

使用 SVV_RLS_RELATION 檢視受到 RLS保護的所有關係清單。

SVV所有使用者都可看見 _RLS_RELATION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
datname	text	包含關聯的資料庫名稱。
relschema	text	包含關係之結構描述的名稱。
relname	text	關係的名稱。
relkind	text	關係的類型，例如資料表或檢視。
is_rls_on	boolean	指出關係是否受到 RLS保護的參數。
is_rls_data_share_on	boolean	指出關聯是否受到資料共用RLS保護的參數。
rls_conjunction_type	character(3)	指出關係是否將RLS政策與 and或 結合的 參數or。
rls_data_share_conjunction_type	character(3)	指出關係是否將RLS政策與資料共用結合and還是or透過資料共用的 參數。

範例查詢

下列範例顯示 SVV_RLS_ 的結果RELATION。

```
ALTER TABLE tickit_category_redshift ROW LEVEL SECURITY ON FOR DATASHARES;

--Inspect RLS state on the relations using SVV_RLS_RELATION.
```

```
SELECT datname, relschema, relname, relkind, is_rols_on, is_rols_datashare_on FROM
svv_rols_relation ORDER BY relname;
```

```

 datname | relschema |          relname          | relkind | is_rols_on |
 is_rols_datashare_on | rls_conjunction_type | rls_datashare_conjunction_type
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
 tickit_db |   public  | tickit_category_redshift | table   |          t |
 |          |          |          |          |          |
(1 row)
```

SVV_ROLE_GRANTS

使用 SVV_ROLE_GRANTS 檢視叢集中明確授予角色的角色清單。

SVV下列使用者可以看到 _ROLE_GRANTS：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
role_id	integer	角色的 ID。
role_name	text	角色的名稱。
granted_role_id	integer	授予角色的 ID。
granted_role_name	text	已授予角色的名稱。

範例查詢

下列範例會傳回 SVV_ROLE_ 的輸出GRANTS。

```
GRANT ROLE role1 TO ROLE role2;
GRANT ROLE role2 TO ROLE role3;
```



```
SELECT role_name, granted_role_name FROM svv_role_grants;
```

```

role_name | granted_role_name
-----+-----
  role2   |         role1
  role3   |         role2
(2 rows)

```

SVV_ROLES

使用 SVV_ROLES 檢視角色資訊。

所有使用者都可看見此資料表。

資料表欄

欄名稱	資料類型	描述
role_id	integer	規則 ID。
role_name	text	角色的名稱。
role_owner	text	角色擁有者的名稱。
external_id	text	第三方身分提供者中角色的唯一識別碼。

範例查詢

下列範例會傳回 SVV_ 的輸出 ROLES。

```
SELECT role_name,role_owner FROM svv_roles WHERE role_name IN ('role1', 'role2');
```

```

role_name | role_owner
-----+-----
  role1   | superuser
  role2   | superuser

```

SVV_SCHEMA_PRIVILEGES

使用 SVV_SCHEMA_PRIVILEGES 檢視明確授予目前資料庫中使用者、角色和群組的結構描述許可。

SVV下列使用者可以看到 `_SCHEMA_PRIVILEGES` :

- 超級使用者
- 具有 `ACCESSSYSTEMTABLE`許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
<code>namespace_name</code>	text	指定結構描述所在的命名空間名稱。
<code>privilege_type</code>	text	許可的類型。對於具有 <code>privilege_scope</code> 的結構描述的許可，可能的值為 <code>CREATE</code> 、 <code>USAGE</code> 和 <code>ALTER</code> 。對於結構描述以外的 <code>privilege_scope</code> 值，可能的值包含許可範圍上可用的任何許可類型。
<code>identity_id</code>	integer	身分的 ID。可能的值是 <code>user ID</code> 、 <code>role ID</code> 或 <code>group ID</code> 。
<code>identity_name</code>	text	身分的名稱。
<code>identity_type</code>	text	身分的類型。可能的值是 <code>user</code> 、 <code>role</code> 、 <code>group</code> 或 <code>public</code> 。
<code>admin_option</code>	boolean	指出使用者是否可以將許可授予其他使用者和角色的值。對於角色和群組識別類型，永遠是 <code>False</code> 。
<code>privilege_scope</code>	text	在 <code>privilege_type</code> 中指定的許可範圍。可能的值如下： <ul style="list-style-type: none"> • <code>SCHEMA</code> • <code>TABLES</code> • <code>FUNCTIONS</code>

欄名稱	資料類型	描述
		如需有關範圍許可的資訊，請前往 限定範圍權限 。

範例查詢

下列範例顯示 SVV_SCHEMA_ 的結果PRIVILEGES。

```
SELECT namespace_name, privilege_type, identity_name, identity_type, admin_option FROM
svv_schema_privileges
WHERE namespace_name = 'test_schema1';
```

namespace_name	privilege_type	identity_name	identity_type	admin_option
test_schema1	USAGE	reguser	user	False
test_schema1	USAGE	role1	role	False

SVV_SCHEMA_QUOTA_STATE

顯示每個結構描述的配額和目前磁碟使用量。

一般使用者可以查看他們擁有USAGE許可的結構描述資訊。超級使用者可以看到目前資料庫中所有結構描述的資訊。

SVV所有使用者都可看見 _SCHEMA_QUOTA_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
schema_id	integer	命名空間或結構描述 ID。

欄名稱	資料類型	描述
schema_name	character (128)	命名空間或結構描述名稱。
schema_owner	integer	結構描述擁有者的內部使用者 ID。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。
disk_usage_pct	double precision	結構描述目前從所設定配額使用的磁碟空間百分比。

範例查詢

下列範例會顯示結構描述的配額和目前磁碟使用量。

```
SELECT TRIM(SCHEMA_NAME) "schema_name", QUOTA, disk_usage, disk_usage_pct FROM
  svv_schema_quota_state
WHERE SCHEMA_NAME = 'sales_schema';
schema_name | quota | disk_usage | disk_usage_pct
-----+-----+-----+-----
sales_schema | 2048 | 30          | 1.46
(1 row)
```

SVV_SYSTEM_PRIVILEGES

SVV下列使用者可以看到 _SYSTEM_PRIVILEGES :

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到他們有權存取或擁有的身分。

資料表欄

欄名稱	資料類型	描述
system_privilege	text	系統許可的名稱。
identity_id	integer	身分的 ID。可能的值是 user ID 或 role ID。
identity_name	text	身分的名稱。
identity_type	text	身分的類型。可能的值是 user 或 role。

範例查詢

下列範例顯示指定參數的結果。

```
SELECT system_privilege,identity_name,identity_type FROM svv_system_privileges
WHERE system_privilege = 'ALTER TABLE' AND identity_name = 'sys:superuser';
```

```
system_privilege | identity_name | identity_type
-----+-----+-----
ALTER TABLE    | sys:superuser | role
```

SVV_TABLE_INFO

顯示資料庫中資料表的摘要資訊。檢視會篩選系統資料表並僅顯示使用者定義的資料表。

您可以使用 SVV_TABLE_INFO 檢視來診斷和解決可能影響查詢效能的資料表設計問題。這包括壓縮編碼、分散索引鍵、排序樣式、資料分散扭曲、資料表大小和統計資料的問題。SVV_TABLE_INFO 檢視不會傳回空白資料表的任何資訊。

SVV_TABLE_INFO 檢視摘要來自 [STV_BLOCKLIST](#)、[STV_TBL_PERM](#)、[STV_NODE_STORAGE_CAPACITY](#)和 [STV_SLICES](#) 系統資料表，以及來自 [PG_DATABASE](#)、[PG_ATTRIBUTE](#)、[PG_CLASS](#)、[PG_NAMESPACE](#) 和 [PG_TYPE](#) 目錄資料表的資訊。

SVV只有超級使用者可以看到 TABLE__INFO。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。若要允許使用者查詢檢視，請將 SVV_TABLE_INFO 上的SELECT許可授予使用者。

資料表欄

欄名稱	資料類型	描述
database	text	資料庫名稱。
schema	text	結構描述名稱。
table_id	oid	表格 ID。
table	text	資料表名稱。
encoded	text	值會指出任何欄位是否定義壓縮編碼。
diststyle	text	分散樣式或分散索引鍵欄 (如果鍵分佈有定義的話)。可能的值包括 EVEN、KEY(<i>column</i>)、ALL、AUTO(A)、AUTO(EVEN) 及 AUTO(KEY(<i>column</i>))。
sortkey1	text	排序索引鍵的第一欄 (如果排序索引鍵有定義的話)。可能的值包括 <i>column</i> 、AUTO(SORT KEY) 與 AUTO(SORT KEY(<i>column</i>))。
max_varchar	integer	使用 VARCHAR 資料類型的最大資料欄大小。
sortkey1_enc	character(32)	排序索引鍵中第一欄的壓縮編碼 (如果排序索引鍵有定義的話)。
sortkey_num	integer	定義為排序索引鍵的欄位編號。

欄名稱	資料類型	描述
size	bigint	資料表的大小 (以 1-MB 資料區塊表示)。
pct_used	numeric(10,4)	資料表使用的可用空間百分比。
empty	bigint	供內部使用。此欄不再使用，並會在未來版本中移除。
unsorted	numeric(5,2)	資料表中未排序之列的百分比。
stats_off	numeric(5,2)	數字會指出資料表統計資訊過時程度。0 為目前，100 為過時。
tbl_rows	numeric(38,0)	資料表中列總數。此值包含標示要進行刪除，但未清空的資料列。
skew_sortkey1	numeric(19,2)	最大非排序索引鍵欄位的大小與排序索引鍵第一欄大小的比率 (如果排序索引鍵有定義的話)。使用此值來評估排序索引鍵是否有效。
skew_rows	numeric(19,2)	資料列數最多的分割，與資料列數最少的分割，兩者之間的資料列數比。
estimated_visible_rows	numeric(38,0)	資料表中的估計資料列，此值不包含標記刪除的資料列。

欄名稱	資料類型	描述
risk_event	text	<p>資料表的相關風險資訊。該欄位會分成以下幾個部分：</p> <pre>risk_type xid timestamp</pre> <ul style="list-style-type: none"> risk_type ，其中 1 表示 COPY command with the EXPLICIT_IDS option 已執行。Amazon Redshift 不再檢查資料表中 IDENTITY 資料欄的唯一性。如需詳細資訊，請參閱 EXPLICIT_IDS。 xid 交易 ID 會說明風險。 COPY 命令執行 timestamp 時的。 <p>下列範例顯示欄位中的值。</p> <pre>1 1107 2019-06-22 07:16:11.292952</pre>
vacuum_sort_benefit	numeric(12,2)	在您的執行清空排序時掃描查詢效能的估計改善百分比上限。
create_time	沒有時區的時間戳記	建立資料表時的時間戳記。

範例查詢

下列範例顯示資料庫中所有使用者定義資料表的編碼、分佈樣式、排序和資料扭曲。此處的 "table" 必須使用雙引號括起來，因為這是保留字。

```
select "table", encoded, diststyle, sortkey1, skew_sortkey1, skew_rows
```



```
from svv_table_info
order by 1;
```

```
table          | encoded | diststyle          | sortkey1      | skew_sortkey1 | skew_rows
-----+-----+-----+-----+-----+-----
category      | N       | EVEN              |               |               |
date          | N       | ALL               | dateid        | 1.00         |
event         | Y       | KEY(eventid)     | dateid        | 1.00         | 1.02
listing       | Y       | KEY(listid)      | dateid        | 1.00         | 1.01
sales         | Y       | KEY(listid)      | dateid        | 1.00         | 1.02
users         | Y       | KEY(userid)      | userid        | 1.00         | 1.01
venue         | N       | ALL               | venueid       | 1.00         |
(7 rows)
```

SVV_TABLES

使用 SVV_TABLES 檢視本機和外部目錄中的資料表。

SVV 所有使用者都可看見 _TABLES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
table_catalog	text	資料表所在之目錄的名稱。
table_schema	text	資料表的結構描述名稱。
table_name	text	表格的名稱。
table_type	text	資料表的類型。可能的值是 views、external tables 和 base tables。
備註	text	備註。

SVV_TRANSACTIONS

記錄資料庫中目前保持鎖定之資料表的交易相關資訊。使用 SVV_TRANSACTIONS 檢視來識別開啟的交易並鎖定爭用問題。如需鎖定的相關資訊，請參閱 [管理並行寫入操作](#) 和 [LOCK](#)。

SVV所有使用者都可看見 _TRANSACTIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
txn_owner	text	交易之擁有者的名稱。
txn_db	text	與交易相關聯之資料庫的名稱。
xid	bigint	交易 ID。
pid	integer	與鎖定關聯的處理程序 ID。
txn_start	timestamp	交易的開始時間。
lock_mode	text	此程序保留或請求的鎖定模式名稱。如果 lock_mode 是 ExclusiveLock 且 granted 為 true (t)，則此交易 ID 是開放交易。
lockable_object_type	text	請求或保留鎖定的物件類型，relation (如果是資料表) 或 transactionid (如果是交易)。
關聯	integer	取得鎖定之資料表 (關聯) 的資料表 ID。如果為 NULL，則此值lockable_object_type 為 transactionid。

欄名稱	資料類型	描述
獲授予	boolean	值會指出鎖定是否獲授予 (t) 或待定 (f)。

範例查詢

下列命令顯示所有作用中的交易且每個交易請求的鎖定數。

```
select * from svv_transactions;
```

```

txn_
lockable_
owner | txn_db | xid   | pid |          txn_start          | lock_mode |
object_type | relation | granted
-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
root | dev | 438484 | 22223 | 2016-03-02 18:42:18.862254 | AccessShareLock |
relation | 100068 | t
root | dev | 438484 | 22223 | 2016-03-02 18:42:18.862254 | ExclusiveLock |
transactionid | | t
root | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | AccessShareLock |
relation | 50860 | t
root | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | AccessShareLock |
relation | 52310 | t
root | tickit | 438490 | 22277 | 2016-03-02 18:42:48.084037 | ExclusiveLock |
transactionid | | t
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
relation | 100068 | f
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | RowExclusiveLock |
relation | 16688 | t
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessShareLock |
relation | 100064 | t
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
relation | 100166 | t
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
relation | 100171 | t
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | AccessExclusiveLock |
relation | 100190 | t
root | dev | 438505 | 22378 | 2016-03-02 18:43:27.611292 | ExclusiveLock |
transactionid | | t
(12 rows)

```

(12 rows)

SVV_USER_GRANTS

使用 SVV_USER_GRANTS 檢視叢集中明確授予角色的使用者清單。

SVV下列使用者可以看到 _USER_GRANTS：

- 超級使用者
- 具有 ACCESSSYSTEMTABLE許可的使用者

其他使用者只能看到明確授予他們的角色。

資料表欄

欄名稱	資料類型	描述
user_id	integer	使用者的使用者 ID。
user_name	text	使用者的名稱。
role_id	integer	授予角色的角色 ID。
role_name	text	授予角色的角色名稱。
admin_option	boolean	指出使用者是否可以將角色授予其他使用者和角色的值。

範例查詢

下列查詢會將角色授予使用者，並顯示明確授予角色的使用者清單。

```
GRANT ROLE role1 TO reguser;
GRANT ROLE role2 TO reguser;
GRANT ROLE role1 TO superuser;
GRANT ROLE role2 TO superuser;

SELECT user_name,role_name,admin_option FROM svv_user_grants;
```

```

user_name | role_name | admin_option
-----+-----+-----
superuser | role1     | False
reguser   | role1     | False
superuser | role2     | False
reguser   | role2     | False

```

SVV_USER_INFO

您可以使用 SVV_USER_INFO 檢視擷取 Amazon Redshift 資料庫使用者的資料。

SVV 所有使用者都可看見 _USER_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_name	text	角色的使用者名稱。
user_id	integer	使用者的使用者 ID。
createdb	boolean	一個值，指出使用者是否具有建立資料庫的許可。
superuser	boolean	一個值，指出使用者是否為超級使用者。
catalog_update	boolean	一個值，指出使用者可以更新系統目錄。
connection_limit	text	使用者可以開啟的連線數量。
syslog_access	text	一個值，指出使用者是否具有系統日誌的存取權。兩個可能值為 RESTRICTED 和 UNRESTRICTED。RESTRICTED 表示不是超級使用者的使用者可以看到其記錄。UNRESTRICTED 表示不是超級使用者的使用者可以看到他們具有 SELECT 權限的系統檢視和資料表中的所有記錄。
last_ddl_timestamp	timestamp	上次資料定義語言 (DDL) 的時間戳記會建立使用者執行的陳述式。

欄名稱	資料類型	描述
session_timeout	integer	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。如需叢集的閒置或非作用中逾時設定詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制 。
external_user_id	text	第三方身分提供者中使用者的唯一識別碼。

範例查詢

下列命令會從 SVV_USER_ 擷取使用者資訊INFO。

```
SELECT * FROM SVV_USER_INFO;
```

SVV_VACUUM_PROGRESS

此檢視會傳回目前進行中的清空操作所需完成時間的預估值。

SVV只有超級使用者可以看到 VACUUM_PROGRESS。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_VACUUM_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

如需有關 SVV_VACUUM_ 的資訊SUMMARY，請參閱 [SVV_VACUUM_SUMMARY](#)。

如需有關 SVL_VACUUM_ 的資訊PERCENTAGE，請參閱 [SVL_VACUUM_PERCENTAGE](#)。

Note

只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
table_name	text	目前正受到清空或前次受到清空的資料表名稱 (如果沒有任何操作在進行中)。
status	text	<p>描述在清空操作過程中所做的目前活動：</p> <ul style="list-style-type: none"> • 初始化 • Sort • Merge • Delete • Select • 失敗 • 完成 • 略過 • 建置INTERLEAVEDSORTKEY順序
time_remaining_estimate	text	<p>目前清空操作所剩需完成的預估時間 (以分和秒表示)：例如，5m 10s。在清空完成其第一次的排序操作前都不會將預估時間傳回。如果沒有正在進行的真空，則上次執行的真空會在 Completed STATUS 欄中顯示為 Completed，並顯示空白的 TIME_REMAINING_ESTIMATE 欄。預估值隨著清空的進行通常會變得越來越精準。</p>

範例查詢

下列查詢間隔幾分鐘執行，顯示名為 `salesnew` 的大型資料表 SALESNEW 正在清空。

```
select * from svv_vacuum_progress;
```

```

table_name      |          status          | time_remaining_estimate
-----+-----+-----
salesnew       | Vacuum: initialize salesnew |

```

```
(1 row)
...
select * from svv_vacuum_progress;

table_name | status | time_remaining_estimate
-----+-----+-----
salesnew | Vacuum salesnew sort | 33m 21s
(1 row)
```

下列查詢顯示目前沒有在進行任何清空操作。要清空的最後一個資料表是SALES資料表。

```
select * from svv_vacuum_progress;

table_name | status | time_remaining_estimate
-----+-----+-----
sales | Complete |
(1 row)
```

SVV_VACUUM_SUMMARY

SVV_VACUUM_SUMMARY 檢視會加入 STL_VACUUM、STL_QUERY和 STV_TBL_PERM資料表，以摘要系統記錄的真空操作相關資訊。檢視會根據清空交易和資料表傳回一個資料列。檢視會記錄操作的經過時間、建立之排序分割區數、所需合併增量數和操作執行前後的資料列和區塊計數的差異。

SVV只有超級使用者可以看到 _VACUUM_SUMMARY。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_VACUUM_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

如需 SVV_VACUUM_ 的相關資訊PROGRESS，請參閱 [SVV_VACUUM_PROGRESS](#)。

如需有關 SVL_VACUUM_ 的資訊PERCENTAGE，請參閱 [SVL_VACUUM_PERCENTAGE](#)。

Note

只有在查詢已佈建的叢集時，才能使用此檢視。

資料表欄

欄名稱	資料類型	描述
table_name	text	清空的資料表名稱。
xid	bigint	VACUUM 操作的交易 ID。
sort_partitions	bigint	在清空操作的排序階段所建立之排序分割區數。
merge_increments	bigint	合併增量數，需要此項目來完成清空操作的合併階段。
elapsed_time	bigint	清空操作的經過執行期 (微秒)。
row_delta	bigint	清空前後資料表資料列數總計的差異。
sortedrow_delta	bigint	清空前後排序資料表資料列數的差異。
block_delta	integer	清空前後資料表區塊計數的差異。
max_merge_partitions	integer	會使用此欄位來分析表現並表示分割區數上限，也就是清空可以根據合併階段反覆運算為資料表處理的分割區數上限。(真空將未分類的區域排序為一個或多個已排序的分區。根據資料表中的欄數和目前的 Amazon Redshift 組態，合併階段可以在單次合併反覆運算中處理最大數量的分割區。如果已排序的分割區數目超過合併分割區的最大數目，合併階段仍然可以運作，但需要更多的合併反覆運算。)

範例查詢

下列查詢會傳回在三個不同資料表上清空操作的統計資訊。SALES 資料表已吸塵兩次。

```
select table_name, xid, sort_partitions as parts, merge_increments as merges,
elapsed_time, row_delta, sortedrow_delta as sorted_delta, block_delta
from svv_vacuum_summary
order by xid;
```

```
table_ | xid | parts | merges | elapsed_ | row_ | sorted_ | block_
```

name				time	delta	delta	delta
users	2985	1	1	61919653	0	49990	20
category	3982	1	1	24136484	0	11	0
sales	3992	2	1	71736163	0	1207192	32
sales	4000	1	1	15363010	-851648	-851648	-140

(4 rows)

SYS 監控檢視

監控檢視是 Amazon Redshift 中的系統檢視，用於監控佈建叢集和無伺服器工作群組的查詢和工作負載資源使用情況。這些檢視位於 `pg_catalog` 結構描述中。若要顯示這些檢視提供的資訊，請執行 SQLSELECT 陳述式。

除非另有說明，否則這些檢視可用於 Amazon Redshift 叢集和 Amazon Redshift Serverless 工作群組。

`SYS_SERVERLESS_USAGE` 只會收集 Amazon Redshift Serverless 的使用資料。

主題

- [SYS_ANALYZE_COMPRESSION_HISTORY](#)
- [SYS_ANALYZE_HISTORY](#)
- [SYS_APPLIED_MASKING_POLICY_LOG](#)
- [SYS_AUTO_TABLE_OPTIMIZATION](#)
- [SYS_CONNECTION_LOG](#)
- [SYS_COPY_JOB \(預覽\)](#)
- [SYS_COPY_REPLACEMENTS](#)
- [SYS_DATASHARE_CHANGE_LOG](#)
- [SYS_DATASHARE_CROSS_REGION_USAGE](#)
- [SYS_DATASHARE_USAGE_CONSUMER](#)
- [SYS_DATASHARE_USAGE_PRODUCER](#)
- [SYS_EXTERNAL_QUERY_DETAIL](#)
- [SYS_EXTERNAL_QUERY_ERROR](#)
- [SYS_INTEGRATION_ACTIVITY](#)
- [SYS_INTEGRATION_TABLE_STATE_CHANGE](#)

- [SYS_LOAD_DETAIL](#)
- [SYS_LOAD_ERROR_DETAIL](#)
- [SYS_LOAD_HISTORY](#)
- [SYS_MV_REFRESH_HISTORY](#)
- [SYS_MV_STATE](#)
- [SYS_PROCEDURE_CALL](#)
- [SYS_PROCEDURE_MESSAGES](#)
- [SYS_QUERY_DETAIL](#)
- [SYS_QUERY_HISTORY](#)
- [SYS_QUERY_TEXT](#)
- [SYS_RESTORE_LOG](#)
- [SYS_RESTORE_STATE](#)
- [SYS_SCHEMA_QUOTA_VIOLATIONS](#)
- [SYS_SERVERLESS_USAGE](#)
- [SYS_SESSION_HISTORY](#)
- [SYS_SPATIAL_SIMPLIFY](#)
- [SYS_STREAM_SCAN_ERRORS](#)
- [SYS_STREAM_SCAN_STATES](#)
- [SYS_TRANSACTION_HISTORY](#)
- [SYS_UDF_LOG](#)
- [SYS_UNLOAD_DETAIL](#)
- [SYS_UNLOAD_HISTORY](#)
- [SYS_USERLOG](#)
- [SYS_VACUUM_HISTORY](#)

SYS_ANALYZE_COMPRESSION_HISTORY

在 COPY 或 ANALYZECOMPRESSION 命令期間記錄壓縮分析操作的詳細資訊。

SYS 所有使用者都可看見 `_ANALYZE_COMPRESSION_HISTORY`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
start_time	timestamp	壓縮分析操作開始進行的時間。
transaction_id	bigint	壓縮分析操作的交易 ID。
table_id	integer	接受分析之資料表的 ID。
table_name	character(128)	接受分析之資料表的名稱。
column_position	integer	接受分析之資料表的資料欄索引，用於判斷壓縮編碼。
old_encoding	character(15)	壓縮分析之後的編碼類型。
new_encoding	character(15)	壓縮分析之後的編碼類型。
模式	character(14)	<p>可能值如下：</p> <p>PRESET</p> <p>指定 new_encoding 是由 Amazon Redshift COPY 命令根據資料欄資料類型決定。不會對任何資料取樣。</p> <p>ON</p> <p>指定 new_encoding 是由 Amazon Redshift COPY 命令根據範例資料分析來決定。</p> <p>ANALYZE ONLY</p> <p>指定 new_encoding 是由 Amazon Redshift ANALYZECOMPRESSION 命令根據範例資料分析來決定。然而，不會變更已分析資料欄的編碼類型。</p>

範例查詢

下列範例會使用相同工作階段中最後一個執行的COPY命令，檢查lineitem資料表上的壓縮分析詳細資訊。

```
select transaction_id, table_id, btrim(table_name) as table_name, column_position,
       old_encoding, new_encoding, mode
from sys_analyze_compression_history
where transaction_id = (select transaction_id from sys_query_history where query_id =
                       pg_last_copy_id()) order by column_position;
```

transaction_id	table_id	table_name	column_position	old_encoding	new_encoding	mode
8196	248126	lineitem	0	mostly32	mostly32	ON
8196	248126	lineitem	1	mostly32	lzo	ON
8196	248126	lineitem	2	lzo	lzo	ON
8196	248126	lineitem	3	delta	delta	ON
8196	248126	lineitem	4	bytedict	bytedict	ON
8196	248126	lineitem	5	mostly32	mostly32	ON
8196	248126	lineitem	6	delta	delta	ON
8196	248126	lineitem	7	delta	delta	ON
8196	248126	lineitem	8	lzo	lzo	ON
8196	248126	lineitem	9	runlength	runlength	ON
8196	248126	lineitem	10	delta	lzo	ON
8196	248126	lineitem	11	delta	delta	ON
8196	248126	lineitem	12	delta	delta	ON
8196	248126	lineitem	13	bytedict	zstd	ON

```

      8196      | 248126 | lineitem | 14 | bytedict | zstd
      | ON
      8196      | 248126 | lineitem | 15 | text255  | zstd
      | ON
(16 rows)

```

SYS_ANALYZE_HISTORY

記錄[ANALYZE](#)操作的詳細資訊。

SYS只有超級使用者才能看見 `_ANALYZE_HISTORY`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>user_id</code>	integer	產生項目的使用者之 ID。
<code>transaction_id</code>	long	交易的 ID。
<code>query_id</code>	long	SYS_QUERY_HISTORY 中的查詢識別碼。
<code>database_name</code>	char(30)	資料庫的名稱。
<code>table_name</code>	char(30)	資料表的名稱。
<code>table_id</code>	integer	資料表的 ID。
<code>is_automatic</code>	char(1)	如果操作預設包含 Amazon Redshift ANALYZE 操作，則值為 true (t)。如果命令明確執行，則值 ANALYZE 為 false (f)。
<code>status</code>	char(15)	分析命令的結果。可能的值為完整、略過和 Predicate Column。
<code>start_time</code>	timestamp	ANALYZE 操作開始執行時在 UTC 中的時間。
<code>end_time</code>	timestamp	ANALYZE 操作完成執行時的 UTC 中時間。

欄名稱	資料類型	描述
rows	double	資料表中的列總數
modified_rows	double	自上次ANALYZE操作以來修改的資料列總數。
analyze_threshold_percent	integer	analyze_threshold_percent 參數的值。
last_analyze_time	timestamp	先前分析資料表時UTC，中的時間。

範例查詢

```

user_id | transaction_id | database_name | schema_name | table_name |
table_id | is_automatic | Status | start_time | end_time
| rows | modified_rows | analyze_threshold_percent | last_analyze_time
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
      101 |          8006 |          dev |          public | test_table_562bf8dc
| 110427 |           f | Full | 2023-09-21 18:33:08.504646 | 2023-09-21
18:33:24.296498 | 5 |           5 |           0 | 2000-01-01
00:00:00

```

SYS_APPLIED_MASKING_POLICY_LOG

使用 SYS_APPLIED_MASKING_POLICY_LOG 在參考 DDM受保護關係的查詢上追蹤動態資料遮罩政策的應用程式。

SYS下列使用者可以看到 _APPLIEDMASKING_POLICY_LOG :

- 超級使用者
- 具有 sys:operator 角色的使用者

- 具有 ACCESSSYSTEMTABLE 許可的使用者

一般使用者將看到 0 列。

請注意，具有 sys:secadmin 角色的使用者看不到 SYS_APPLIEDMASKING_POLICY_LOG。

如需動態資料遮罩的詳細資訊，請前往 [動態資料遮罩](#)。

資料表欄

欄名稱	資料類型	描述
policy_name	text	遮罩政策的名稱。
user_id	text	執行查詢的使用者 ID。
record_time	timestamp	記錄系統檢視項目的時間。
session_id	int	程序 ID。
transaction_id	long	交易的 ID。
query_id	int	查詢 ID。
database_name	text	執行查詢的資料庫名稱。
relation_name	text	套用遮罩政策的資料表名稱。
schema_name	text	資料表所在的結構描述名稱。
attachment_id	long	連接的遮罩政策 ID。
relation_kind	text	套用遮罩政策的關係類型。可能值為 TABLE、VIEW、LATE BINDING VIEW 和 MATERIALIZED VIEW 。

範例查詢

下列範例顯示mask_credit_card_full遮罩政策已連接至credit_db.public.credit_cards資料表。

```
select policy_name, database_name, relation_name, schema_name, relation_kind
from sys_applied_masking_policy_log;

policy_name          | database_name | relation_name | schema_name | relation_kind
-----+-----+-----+-----+-----
mask_credit_card_full | credit_db     | credit_cards  | public      | table

(1 row)
```

SYS_AUTO_TABLE_OPTIMIZATION

記錄 Amazon Redshift 對自動最佳化而定義的資料表執行的自動操作。

SYS只有超級使用者可以看到 _AUTO_TABLE_OPTIMIZATION。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
transaction_id	long	交易識別碼。
session_id	int	執行更改命令之程序的工作階段識別碼。
table_id	int	資料表識別碼。
alter_table_type	character (32)	建議的類型。可能值為 distkey、sortkey 及 encode。
status	character (128)	建議的完成狀態。可能值為 Start、Complete、Skipped、Abort、Checkpoint 以及 Failed。

欄名稱	資料類型	描述
event_time	timestamp	狀態欄的時間戳記。
alter_from	character (200)	套用建議之前，資料表的先前分散樣式和排序索引鍵。該值會被截斷為 200 個字元的增量。

範例查詢

在下列範例中，結果中的列顯示 Amazon Redshift 所採取的動作。

```
SELECT table_id, alter_table_type, status, event_time, alter_from
FROM SYS_AUTO_TABLE_OPTIMIZATION;
```

table_id	alter_table_type	status	event_time	alter_from
118082	sortkey	Start	2020-08-22 19:42:20.727049	
118078	sortkey	Start	2020-08-22 19:43:54.728819	
118082	sortkey	Start	2020-08-22 19:42:52.690264	
118072	sortkey	Start	2020-08-22 19:44:14.793572	
118082	sortkey	Failed	2020-08-22 19:42:20.728917	
118078	sortkey	Complete	2020-08-22 19:43:54.792705	SORTKEY: None;
118086	sortkey	Complete	2020-08-22 19:42:00.72635	SORTKEY: None;
118082	sortkey	Complete	2020-08-22 19:43:34.728144	SORTKEY: None;
118072	sortkey	Skipped:Retry exceeds the maximum limit for a table.	2020-08-22 19:44:46.706155	
118086	sortkey	Start	2020-08-22 19:42:00.685255	
118082	sortkey	Start	2020-08-22 19:43:34.69531	

```

118072 | sortkey          | Start
| 2020-08-22 19:44:46.703331 |
118082 | sortkey          | Checkpoint: progress 14.755079%
| 2020-08-22 19:42:52.692828 |
118072 | sortkey          | Failed
| 2020-08-22 19:44:14.796071 |
116723 | sortkey          | Abort:This table is not AUTO.
| 2020-10-28 05:12:58.479233 |
110203 | distkey          | Abort:This table is not AUTO.
| 2020-10-28 05:45:54.67259   |

```

SYS_CONNECTION_LOG

記錄身分驗證嘗試以及連線和中斷連線。

SYS只有超級使用者才能看見 `_CONNECTION_LOG`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
事件	character(50)	連線或身分驗證事件。
record_time	timestamp	發生事件的時間。
remote_host	character(45)	遠端主機的名稱或 IP 地址。
remote_port	character(32)	遠端主機的連接埠號碼。
session_id	integer	與陳述式相關聯的處理程序 ID。
database_name	character(50)	資料庫名稱。
user_name	character(50)	使用者名稱。
auth_method	character(32)	身分驗證方法。

欄名稱	資料類型	描述
持續時間	integer	連線的持續時間，以微秒為單位。
ssl_version	character(50)	安全通訊端層 (SSL) 版本。
ssl_cipher	character(128)	SSL 密碼。
mtu	integer	最大傳輸單位 (MTU) 。
ssl_compression	character(64)	SSL 壓縮類型。
ssl_expansion	character(64)	SSL 擴充類型。
iam_auth_guid	character(36)	請求的IAM CloudTrail身分驗證 ID。
application_name	character(250)	工作階段之應用程式的初始或已更新名稱。
driver_version	character(64)	從第三方SQL用戶端工具連線至 Amazon Redshift 叢集的 ODBC或 JDBC驅動程式版本。
os_version	character(64)	連線到 Amazon Redshift 叢集之用戶端機器上的作業系統版本。
plugin_name	character(32)	連接至您的 Amazon Redshift 叢集時使用的外掛程式名稱。

欄名稱	資料類型	描述
protocol_version	integer	<p>Amazon Redshift 驅動程式在建立與伺服器的連線時使用的內部通訊協定版本。通訊協定版本會在驅動程式與伺服器之間進行交涉。該版本描述了可用的功能。有效值包含：</p> <ul style="list-style-type: none"> • 0 (BASE_SERVER_PROTOCOL_VERSION) • 1 (EXTENDED_RESULT_METADATA_SERVER_PROTOCOL_VERSION) – 若要為每個查詢儲存往返，伺服器會傳送額外的結果集中繼資料資訊。 • 2 (BINARY_PROTOCOL_VERSION) – 根據結果集的資料類型，伺服器會以二進位格式傳送資料。 • 3 (EXTENDED2_RESULT_METADATA_SERVER_PROTOCOL_VERSION) – 伺服器傳送資料欄的案例敏感度 (定序) 資訊。
global_session_id	character(36)	目前工作階段的全域唯一識別碼。工作階段 ID 在節點故障重新啟動後仍然存在。

範例查詢

若要檢視已開啟之連線的詳細資訊，請執行下列查詢。

```
select record_time, user_name, database_name, remote_host, remote_port
from sys_connection_log
where event = 'initiating session'
and session_id not in
(select session_id from sys_connection_log
where event = 'disconnecting session')
order by 1 desc;
```

```
record_time          | user_name   | database_name   | remote_host   | remote_port
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
2014-11-06 20:30:06 | rdsdb      | dev             | [local]      |
2014-11-06 20:29:37 | test001    | test           | 10.49.42.138 | 11111
```

```

2014-11-05 20:30:29 | rdsdb          | dev          | 10.49.42.138 | 33333
2014-11-05 20:28:35 | rdsdb          | dev          | [local]      |
(4 rows)

```

下列範例反映失敗的身分驗證嘗試，以及成功的連線和中斷連線。

```

select event, record_time, remote_host, user_name
from sys_connection_log order by record_time;

          event          |          record_time          | remote_host | user_name
-----+-----+-----+-----
authentication failure | 2012-10-25 14:41:56.96391    | 10.49.42.138 | john
authenticated          | 2012-10-25 14:42:10.87613    | 10.49.42.138 | john
initiating session     | 2012-10-25 14:42:10.87638    | 10.49.42.138 | john
disconnecting session  | 2012-10-25 14:42:19.95992    | 10.49.42.138 | john
(4 rows)

```

下列範例顯示ODBC驅動程式的版本、用戶端電腦上的作業系統，以及用來連線至 Amazon Redshift 叢集的外掛程式。在此範例中，使用的外掛程式用於使用登入名稱和密碼進行標準ODBC驅動程式驗證。

```

select driver_version, os_version, plugin_name from sys_connection_log;

driver_version          | os_version          | plugin_name
-----+-----+-----
Amazon Redshift ODBC Driver 1.4.15.0001 | Darwin 18.7.0 x86_64 | none
Amazon Redshift ODBC Driver 1.4.15.0001 | Linux 4.15.0-101-generic x86_64 | none

```

下列範例顯示用戶端電腦上的作業系統版本、驅動程式版本和通訊協定版本。

```

select os_version, driver_version, protocol_version from sys_connection_log;

os_version          | driver_version          | protocol_version
-----+-----+-----

```

```
-----+-----+-----
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
```

SYS_COPY_JOB (預覽)

這是自動複製的預先發行文件 COPY (SQL JOB)，其位於預覽版本中。文件和功能會隨時變更。我們建議僅在測試環境中使用此功能，不要在生產環境中使用。公開預覽將於 2024 年 10 月 31 日結束。預覽叢集將在預覽版結束的兩週後自動移除。如需預覽版條款和條件，請參閱 [AWS 服務條款](#) 中的 Beta 版和預覽版。

使用 SYS_COPY_JOB 檢視 COPYJOB 命令的詳細資訊。

此檢視包含已建立的 COPYJOB 命令。

SYS 所有使用者都可看見 _COPY_JOB。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
job_id	bigint	複製任務識別碼。
job_name	character(128)	複製任務的名稱。
iam_role	character(128)	COPY 陳述式中指定的 IAM 角色。
job_text	character(256)	COPY 陳述式的參數。
is_auto	integer	指示 COPY JOB 是否由 Amazon Redshift 自動執行。1 表示 true，0 表示 false。
on_error_suspend	integer	此資訊僅供內部使用。

SYS_COPY_REPLACEMENTS

顯示日誌，其中記錄的無效 UTF-8 字元已由 [COPY](#) 命令取代為 ACCEPTINVCHARS 選項。對於每個節點切片上需要至少更換一次的前 100 列，日誌項目會新增至 SYS_COPY_REPLACEMENTS。

您可以使用此檢視來查看有關無伺服器工作群組和已佈建叢集的資訊。

SYS 所有使用者都可看見 _COPY_REPLACEMENTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生查詢的使用者之 ID。
query_id	bigint	查詢 ID。用於聯結其他系統資料表與檢視的欄。
table_id	integer	資料表 ID。
file_name	character (256)	COPY 命令輸入檔案的完整路徑。
column_name	character (127)	第一個包含無效 UTF-8 字元的欄位。
line_number	bigint	輸入資料檔案中包含無效 UTF-8 字元的行號。-1 表示行號不可用，例如從資料欄資料檔案複製時。
raw_line	character (1024)	包含無效 UTF-8 字元的原始載入資料。

範例查詢

下列範例會傳回最近 COPY 操作的替換。

```
select query_idp, table_id, file_name, line_number, colname
from sys_copy_replacements
where query = pg_last_copy_id();
```



```

query_id | table_id | file_name
line_number | column_name
-----+-----+-----
+-----+-----+-----
   96   |    26   | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt
123 | city
   96   |    26   | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt
456 | city
   96   |    26   | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt
789 | city
   96   |    26   | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt
  12   | city
   96   |    26   | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt
 119 | city
...

```

SYS_DATASHARE_CHANGE_LOG

記錄用於追蹤生產者和消費者叢集上資料共用變更的合併檢視。

SYS所有使用者都可看見 `_DATASHARE_CHANGE_LOG`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>user_id</code>	integer	採取動作之使用者的 ID。
<code>user_name</code>	varchar(128)	採取動作之使用者的名稱。
<code>session_id</code>	integer	工作階段的 ID。
<code>transaction_id</code>	bigint	交易的 ID。
<code>share_id</code>	integer	受影響資料共用的 ID。

欄名稱	資料類型	描述
share_name	varchar(128)	資料共用的名稱。
source_database_id	integer	資料共用所屬資料庫的 ID。
source_database_name	varchar(128)	資料共用所屬資料庫的名稱。
consumer_database_id	integer	從資料共用匯入之資料庫的 ID。
consumer_database_name	varchar(128)	從資料共用匯入之資料庫的名稱。
arn	varchar(192)	支援匯入資料庫的資源ARN的。
record_time	timestamp	動作的時間戳記。
動作	varchar(128)	正在執行的動作。可能的值為REVOKEUSAGE共用資料庫、CREATEDATASHAREDROP、DATASHARE、GRANTALTERREVOKEALTER、GRANTSHARE、REVOKESHAREALTERADD、ALTERREMOVEALTERSET、GRANTUSAGE、REVOKEUSAGE、CREATEDATABASE、GRANT、或 DROP SHARED DATABASE ALTER SHARED DATABASE。
status	integer	動作的狀態。可能的值為 SUCCESS和 ERROR-ERROR CODE。
share_object_type	varchar(64)	新增至資料共用或從資料共用中移除之資料庫物件的類型。可能的值包括 schemas、tables、columns、functions 和 views。這是生產者叢集的欄位。

欄名稱	資料類型	描述
share_object_id	integer	新增至資料共用或從資料共用中移除之資料庫物件的 ID。這是生產者叢集的欄位。
share_object_name	varchar(128)	新增至資料共用或從資料共用中移除之資料庫物件的名稱。這是生產者叢集的欄位。
target_user_type	varchar(16)	授與權限的使用者或群組類型。這是生產者和消費者叢集的欄位。
target_user_id	integer	授與權限的使用者或群組 ID。這是生產者和消費者叢集的欄位。
target_user_name	varchar(128)	授與權限的使用者或群組名稱。這是生產者和消費者叢集的欄位。
consumer_account	varchar(16)	資料消費者的帳戶 ID。這是生產者叢集的欄位。
consumer_namespace	varchar(64)	資料消費者帳戶的命名空間。這是生產者叢集的欄位。
producer_account	varchar(16)	資料共用所屬生產者帳戶的帳戶 ID。這是消費者叢集的欄位。
producer_namespace	varchar(64)	資料共用所屬產品帳戶的命名空間。這是消費者叢集的欄位。
attribute_name	varchar(64)	資料共用或共用資料庫的屬性名稱。
attribute_value	varchar(128)	資料共用或共用資料庫的屬性值。
message	varchar(512)	動作失敗時的錯誤訊息。

範例查詢

下列範例顯示 SYS_DATASHARE_CHANGE_LOG 檢視。

```
SELECT DISTINCT action
FROM sys_datashare_change_log
WHERE share_object_name LIKE 'tickit%';

      action
-----
"ALTER DATASHARE ADD"
```

SYS_DATASHARE_CROSS_REGION_USAGE

使用 SYS_DATASHARE_CROSS_REGION_USAGE 檢視取得跨區域資料共用查詢所導致跨區域資料傳輸用量的摘要。SYS_DATASHARE_CROSS_REGION_USAGE 彙總區段層級的詳細資訊。

SYS 只有超級使用者可以看到 _DATASHARECROSS_REGION_USAGE。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
query_id	integer	查詢的 ID。使用此值來聯結各種其他系統資料表與檢視。
child_query_sequence	integer	重寫使用者查詢的順序，從 1 開始。
segment_id	bigint	區段的號碼。查詢包含多個區段，每個區段包含一或多個步驟。
start_time	time	UTC 資料傳輸開始的時間。
end_time	time	資料傳輸結束UTC的時間。
transferred_data	bigint	從生產者區域傳輸至消費者區域的資料位元組數量。
source_region	char(25)	查詢傳輸資料的來源生產者區域。

範例查詢

下列範例顯示 SYS_DATASHARE_CROSS_REGION_USAGE 檢視。

```
SELECT query, segment, transferred_data, source_region
from sys_datashare_cross_region_usage
where query = pg_last_query_id()
order by query,segment;
```

query	segment	transferred_data	source_region
200048	2	4194304	us-west-1
200048	2	4194304	us-east-2

SYS_DATASHARE_USAGE_CONSUMER

記錄資料共用的活動和使用情況。此檢視僅與消費者叢集相關。

SYS所有使用者都可看見 _DATASHARE_USAGE_CONSUMER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	發出要求的使用者 ID。
session_id	integer	執行查詢之領導者程序的 ID。
transaction_id	bigint	目前交易的內容。
request_id	varchar(50)	請求API呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
transaction_uid	varchar(50)	交易的唯一 ID。

欄名稱	資料類型	描述
record_time	timestamp	記錄動作的時間。
status	integer	請求API呼叫的狀態。
error_message	varchar(512)	錯誤的訊息。

範例查詢

下列範例顯示 SYS_DATASHARE_USAGE_CONSUMER 檢視。

```
SELECT request_type, status, trim(error) AS error
FROM sys_datashare_usage_consumer
```

```

request_type | status | error_message
-----+-----+-----
"GET RELATION" | 0 |

```

SYS_DATASHARE_USAGE_PRODUCER

記錄資料共用的活動和使用情況。此檢視僅與生產者叢集相關。

SYS所有使用者都可看見 _DATASHARE_USAGE_PRODUCER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
share_id	integer	資料共用的物件 ID (OID)。
share_name	varchar(128)	資料共用的名稱。

欄名稱	資料類型	描述
request_id	varchar(50)	請求API呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
object_type	varchar(64)	從資料共用共用的物件類型。可能的值包括 schemas、tables、columns、functions 和 views。
object_oid	integer	從資料共用共用的物件 ID。
object_name	varchar(128)	要從資料共用共用的物件名稱。
consumer_account	varchar(16)	資料共用共用到的消費者帳戶的帳戶。
consumer_namespace	varchar(64)	資料共用共用到的使用者帳戶的命名空間。
consumer_transaction_uid	varchar(50)	消費者叢集上陳述式的唯一交易 ID。
record_time	timestamp	記錄動作的時間。
status	integer	資料共用的狀態。
error_message	varchar(512)	錯誤的訊息。
consumer_region	char(64)	取用者叢集所在的區域。

範例查詢

下列範例顯示 SYS_DATASHARE_USAGE_PRODUCER 檢視。

```
SELECT DISTINCT
FROM sys_datashare_usage_producer
WHERE object_name LIKE 'tickit%';

    request_type
-----
"GET RELATION"
```

SYS_EXTERNAL_QUERY_DETAIL

使用 SYS_EXTERNAL_QUERY_DETAIL 在區段層級檢視查詢的詳細資訊。每一列代表來自特定WLM查詢的區段，其詳細資訊包括 Amazon S3 中外部資料表的已處理列數、已處理的位元組數和分割區資訊。此檢視中的每一列也會在 SYS_QUERY_DETAIL 檢視中有一個對應的項目，但此檢視具有與外部查詢處理相關的更多詳細資訊。

SYS所有使用者都可看見 _EXTERNAL_QUERY_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	外部查詢的查詢識別碼。
transaction_id	bigint	交易識別碼。
child_query_sequence	integer	重寫使用者查詢的順序。從 0 開始，類似於 segment_id。
segment_id	integer	查詢區段的區段識別碼。
source_type	character(32)	查詢的資料來源類型，對於 Redshift Spectrum 可以是 S3，對於聯合查詢可以是 PG。

欄名稱	資料類型	描述
start_time	timestamp	開始查詢的時間。
end_time	timestamp	查詢完成的時間。
duration	bigint	查詢花費的時間 (微秒)。
total_partitions	integer	Amazon S3 查詢所需的分割區數量。
qualified_partitions	integer	Amazon S3 查詢掃描的分割區數量。
scanned_files	bigint	要掃描的 Amazon S3 檔案數目。
returned_rows	bigint	Amazon S3 查詢的已掃描列數，或是聯合查詢的傳回列數。
returned_bytes	bigint	Amazon S3 查詢的已掃描位元組數，或是聯合查詢的傳回位元組數。
file_format	text	Amazon S3 檔案的檔案格式。
file_location	text	外部資料表的 Amazon S3 位置。
external_query_text	text	聯合查詢的區段層級查詢文字。
warning_message	character(4000)	執行查詢時顯示的警告訊息。
table_name	character(136)	正在操作之步驟的資料表名稱。
is_recursive	character(1)	指示是否對子資料夾遞迴掃描。

欄名稱	資料類型	描述
is_nested	character(1)	指出是否存取巢狀欄資料類型。
s3list_time	bigint	檔案列出的持續時間，以毫秒為單位。
get_partition_time	long	從 和 Apache Hive 列出 AWS Glue Data Catalog 和驗證指定外部物件分割區所花費的時間。

範例查詢

下列查詢會顯示外部查詢詳細資訊。

```
SELECT query_id,
       segment_id,
       start_time,
       end_time,
       total_partitions,
       qualified_partitions,
       scanned_files,
       returned_rows,
       returned_bytes,
       trim(external_query_text) query_text,
       trim(file_location) file_location
FROM sys_external_query_detail
ORDER BY query_id, start_time DESC
LIMIT 2;
```

輸出範例。

```
query_id | segment_id |      start_time      |      end_time
| total_partitions | qualified_partitions | scanned_files | returned_rows |
returned_bytes | query_text | file_location
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```

763251 |          0 | 2022-02-15 22:32:23.312448 | 2022-02-15 22:32:24.036023 |
      3 |          3 |          3 |          38203 |          2683414 |
      |
763254 |          0 | 2022-02-15 22:32:40.17103  | 2022-02-15 22:32:40.839313 |
      3 |          3 |          3 |          38203 |          2683414 |
      |

```

SYS_EXTERNAL_QUERY_ERROR

您可以查詢系統檢視 SYS_EXTERNAL_QUERY_ERROR 以取得 Redshift Spectrum 掃描錯誤的相關資訊。SYS_EXTERNAL_QUERY_ERROR 會顯示記錄錯誤的範例。預設值是每個查詢 10 個項目。

SYS 所有使用者都可看見 _EXTERNAL_QUERY_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生此列之使用者的識別碼。
query_id	bigint	產生此列之查詢的查詢識別碼。
file_location	char(256)	要查詢之資料的位置。
rowid	char(2100)	<p>錯誤在檔案內的位置。rowid 部分以：(冒號) 分隔，未來可能會添加其他部分。</p> <pre>row_offset :row_group :row_id</pre> <p>Row_offset 是檔案中列的偏移量 (以位元組為單位)，針對不支援的檔案格式設定為 -1。資料表分為 row_groups，每個群組都有具有不同 row_id 的列。</p>
column_name	char(127)	查詢傳回的欄名稱。
original_value	char(1024)	已查詢原始值。

欄名稱	資料類型	描述
modified_value	char(1024)	根據查詢中指定的資料處理組態選項傳回的修改值。
觸發條件	char(128)	查詢中指定的資料處理選項。
動作	char(128)	與查詢中指定的資料處理選項相關聯的動作。
action_value	char(128)	與查詢中指定的資料處理選項相關聯的動作參數值。
error_code	integer	查詢中指定之資料處理選項的結果代碼。

範例查詢

下列查詢會傳回執行資料處理操作的列清單。

```
SELECT * FROM sys_external_query_error;
```

此查詢會傳回類似以下的結果。

```

 user_id  query_id  file_location                                rowid
column_name  original_value  modified_value  trigger
action
  100     1574007  s3://spectrum-uddh/league/spi_global_rankings.0:0
league_name  Barclays Premier League  Barclays Premier Lea UNSPECIFIED
TRUNCATE
  100     1574007  s3://spectrum-uddh/league/spi_global_rankings.0:0
league_nspi  34595          32767          UNSPECIFIED
OVERFLOW_VALUE  199
  100     1574007  s3://spectrum-uddh/league/spi_global_rankings.0:1
league_nspi  34151          32767          UNSPECIFIED
OVERFLOW_VALUE  199
  100     1574007  s3://spectrum-uddh/league/spi_global_rankings.0:2
league_name  Barclays Premier League  Barclays Premier Lea UNSPECIFIED
TRUNCATE
  100     1574007  s3://spectrum-uddh/league/spi_global_rankings.0:2
league_nspi  33223          32767          UNSPECIFIED
OVERFLOW_VALUE  199

```

```

100      1574007  s3://spectrum-uddh/league/spi_global_rankings.0:3
league_name      Barclays Premier League   Barclays Premier Lea UNSPECIFIED
TRUNCATE                156
100      1574007  s3://spectrum-uddh/league/spi_global_rankings.0:3
league_nspi      32808                    32767                    UNSPECIFIED
OVERFLOW_VALUE   199
100      1574007  s3://spectrum-uddh/league/spi_global_rankings.0:4
league_nspi      32790                    32767                    UNSPECIFIED
OVERFLOW_VALUE   199
100      1574007  s3://spectrum-uddh/league/spi_global_rankings.0:5
league_name      Spanish Primera Division   Spanish Primera Divi UNSPECIFIED
TRUNCATE                156
100      1574007  s3://spectrum-uddh/league/spi_global_rankings.0:6
league_name      Spanish Primera Division   Spanish Primera Divi UNSPECIFIED
TRUNCATE                156

```

SYS_INTEGRATION_ACTIVITY

SYS_INTEGRATION_ACTIVITY 顯示已完成整合執行的詳細資訊。

SYS只有超級使用者可以看到 _INTEGRATION_ACTIVITY。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需零ETL整合的相關資訊，請參閱 Amazon Redshift 管理指南中的[使用零ETL整合](#)。

資料表欄

欄名稱	資料類型	描述
integration_id	character (128)	與整合關聯的識別碼。
target_database	character (128)	Amazon Redshift 中接收整合資料的資料庫。
source	character (128)	整合的來源資料。可能的類型包括 MySQL 和 PostgreSQL 。
checkpoint_name	character (128)	檢查點複寫 binlog 座標的名稱。

欄名稱	資料類型	描述
checkpoint_type	character (16)	檢查點的類型。可能的值包括：snapshot、cdc。
checkpoint_bytes	bigint	此檢查點中的位元組數。
last_commit_timestamp	timestamp	上次在此檢查點中遞交的時間戳記。
modified_tables	integer	檢查點中修改的資料表數目。
integration_start_time	time	此檢查點開始整合的時間（UTC）。
integration_end_time	time	此檢查點的整合結束時的時間（UTC）。

範例查詢

下列SQL命令會顯示整合的日誌。

```
select * from sys_integration_activity;

      integration_id          | target_database | source |
      checkpoint_name        | checkpoint_type | checkpoint_bytes |
      last_commit_timestamp  | modified_tables | integration_start_time |
      integration_end_time
-----+-----+-----
+-----+-----+-----
+-----+-----+-----
+-----
76b15917-afae-4447-b7fd-08e2a5acce7b | demo1          | MySQL | checkpoints/
checkpoint_3_241_3_510.json | cdc            | 762   | 2023-05-10
23:00:14.201 | 1              | 2023-05-10 23:00:45.054265 | 2023-05-10
23:00:46.339826
76b15917-afae-4447-b7fd-08e2a5acce7b | demo1          | MySQL | checkpoints/
checkpoint_3_16329_3_17839.json | cdc            | 13488 | 2023-05-11
01:33:57.411 | 2              | 2023-05-11 02:19:09.440121 | 2023-05-11
02:19:16.090492
76b15917-afae-4447-b7fd-08e2a5acce7b | demo1          | MySQL | checkpoints/
checkpoint_3_5103_3_5532.json | cdc            | 1657  | 2023-05-10
23:13:14.205 | 2              | 2023-05-10 23:13:23.545487 | 2023-05-10
23:13:25.652144
```

SYS_INTEGRATION_TABLE_STATE_CHANGE

SYS_INTEGRATION_TABLE_STATE_CHANGE 會顯示整合的資料表狀態變更日誌詳細資訊。

超級使用者可以查看此表格中的所有資料列。

如需詳細資訊，請參閱[使用零ETL整合](#)。

資料表欄

欄名稱	資料類型	描述
integration_id	character (128)	與整合關聯的識別碼。
database_name	character (128)	Amazon Redshift 資料庫的名稱。
schema_name	character (128)	Amazon Redshift 結構描述的名稱。
table_name	character (128)	資料表的名稱。
new_state	character (128)	資料表的狀態。可能值為 Synced、ResyncRequired、ResyncInitiated、Deleted、Failed 以及 ResyncDeleted。
table_last_replicated_checkpoint	character (128)	目前同步的記錄座標。
state_change_reason	character (256)	最後狀態轉換的原因。
record_time	timestamp	此記錄更新的時間 (UTC)。

範例查詢

下列SQL命令會顯示整合的日誌。

```
select * from sys_integration_table_state_change;
```

```

      integration_id          | database_name | schema_name | table_name
| new_state | table_last_replicated_checkpoint | state_change_reason |
record_time
-----+-----+-----+-----
+-----+-----+-----+-----
+-----+-----+-----+-----
99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb      | sbtest80t3s | sbtest79  |
Synced    | {"txn_seq":9834,"txn_id":126597515} |                | 2023-09-20
19:39:50.087868
99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb      | sbtest80t3s | sbtest56  |
Synced    | {"txn_seq":9834,"txn_id":126597515} |                | 2023-09-20
19:39:45.54005
99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb      | sbtest80t3s | sbtest50  |
Synced    | {"txn_seq":9834,"txn_id":126597515} |                | 2023-09-20
19:40:20.362504
99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb      | sbtest80t3s | sbtest18  |
Synced    | {"txn_seq":9834,"txn_id":126597515} |                | 2023-09-20
19:40:32.544084
99108e72-1cfd-414f-8cc0-0216acefac77 | perfdb      | sbtest40t3s | sbtest23  |
Synced    | {"txn_seq":9834,"txn_id":126597515} |                | 2023-09-20
15:49:05.186209

```

SYS_LOAD_DETAIL

傳回資訊以追蹤資料載入或對其進行故障診斷。

此檢視會在每一個資料檔案載入至資料庫資料表時記錄其進度。

所有使用者都可看見此檢視。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
query_id	integer	查詢 ID。
file_name	character(256)	要載入的檔案名稱。

欄名稱	資料類型	描述
bytes_scanned	integer	從 Amazon S3 中的檔案掃描的位元組數。
lines_scanned	integer	從載入檔案掃描的行數。此數目可能不符合實際載入的資料列數目。例如，根據 COPY 命令中的 MAXERROR 選項，負載可能會掃描但可容忍許多不良記錄。
record_time	timestamp	上次更新此項目的時間。
splits_scanned	此檔案的分割數目。	此檔案的分割數目。
start_time	timestamp	此檔案處理開始的時間。
end_time	timestamp	此檔案處理完成的時間。

範例查詢

下列範例會傳回上次 COPY 操作的詳細資訊。

```
select query_id, trim(file_name) as file, record_time
from sys_load_detail
where query_id = pg_last_copy_id();
```

```
query_id |          file          |          record_time
-----+-----+-----
 28554   | s3://dw-tickit/category_pipe.txt | 2013-11-01 17:14:52.648486
(1 row)
```

下列查詢包含 TICKIT 資料庫中資料表新載入的項目：

```
select query_id, trim(file_name), record_time
from sys_load_detail
where file_name like '%tickit%' order by query_id;
```

```
query_id |          btrim          |          record_time
-----+-----+-----
 22475   | tickit/allusers_pipe.txt | 2013-02-08 20:58:23.274186
```

```

22478 | tickit/venue_pipe.txt | 2013-02-08 20:58:25.070604
22480 | tickit/category_pipe.txt | 2013-02-08 20:58:27.333472
22482 | tickit/date2008_pipe.txt | 2013-02-08 20:58:28.608305
22485 | tickit/allevvents_pipe.txt| 2013-02-08 20:58:29.99489
22487 | tickit/listings_pipe.txt | 2013-02-08 20:58:37.632939
22593 | tickit/allusers_pipe.txt | 2013-02-08 21:04:08.400491
22596 | tickit/venue_pipe.txt | 2013-02-08 21:04:10.056055
22598 | tickit/category_pipe.txt | 2013-02-08 21:04:11.465049
22600 | tickit/date2008_pipe.txt | 2013-02-08 21:04:12.461502
22603 | tickit/allevvents_pipe.txt| 2013-02-08 21:04:14.785124
22605 | tickit/listings_pipe.txt | 2013-02-08 21:04:20.170594

```

(12 rows)

事實上，記錄寫入至此系統檢視的日誌檔案，並不表示已成功遞交載入，做為其包含交易的一部分。若要驗證負載遞交，請查詢 STL_UTILITYTEXT 檢視並尋找與COPY交易對應的COMMIT記錄。例如，此查詢LOADSTL會根據 SYS_DETAILQUERYSTLUTILITYTEXT：

```

select l.query_id,rtrim(l.file_name),q.xid
from sys_load_detail l, stl_query q
where l.query_id=q.query
and exists
(select xid from stl_utilitytext where xid=q.xid and rtrim("text")='COMMIT');

```

query_id	rtrim	xid
22600	tickit/date2008_pipe.txt	68311
22480	tickit/category_pipe.txt	68066
7508	allusers_pipe.txt	23365
7552	category_pipe.txt	23415
7576	allevvents_pipe.txt	23429
7516	venue_pipe.txt	23390
7604	listings_pipe.txt	23445
22596	tickit/venue_pipe.txt	68309
22605	tickit/listings_pipe.txt	68316
22593	tickit/allusers_pipe.txt	68305
22485	tickit/allevvents_pipe.txt	68071
7561	allevvents_pipe.txt	23429
7541	category_pipe.txt	23415
7558	date2008_pipe.txt	23428
22478	tickit/venue_pipe.txt	68065
526	date2008_pipe.txt	2572
7466	allusers_pipe.txt	23365

```

22482 | tickit/date2008_pipe.txt | 68067
22598 | tickit/category_pipe.txt | 68310
22603 | tickit/allevnts_pipe.txt | 68315
22475 | tickit/allusers_pipe.txt | 68061
  547 | date2008_pipe.txt       |  2572
22487 | tickit/listings_pipe.txt | 68072
 7531 | venue_pipe.txt          | 23390
 7583 | listings_pipe.txt       | 23445
(25 rows)

```

SYS_LOAD_ERROR_DETAIL

使用 `SYS_LOAD_ERROR_DETAIL` 檢視 COPY 命令錯誤的詳細資訊。每列代表一個 COPY 命令。它同時包含執行中和已完成的 COPY 命令。

SYS 所有使用者都可看見 `_LOAD_ERROR_DETAIL`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>user_id</code>	integer	提交副本之使用者的識別碼。
<code>query_id</code>	bigint	副本的查詢識別碼。
<code>transaction_id</code>	bigint	交易識別碼。
<code>session_id</code>	integer	執行副本之程序的處理程序識別碼。
<code>database_name</code>	character(64)	發出複製時，要將使用者連接至其中的資料庫名稱。
<code>table_id</code>	integer	資料表識別碼。
<code>start_time</code>	timestamp	複製開始的時間 (UTC)。
<code>file_name</code>	character(256)	用於載入之輸入檔案的完整路徑。

欄名稱	資料類型	描述
line_number	bigint	載入檔案中發生錯誤的行號。當您載入JSON檔案時，JSON物件最後一行的行編號會帶有錯誤。
column_name	character(127)	發生錯誤的欄位。
column_type	character(10)	包含錯誤之欄位的資料類型。
column_length	character(10)	欄長度 (如適用)。當資料類型具有限制長度時，會填入此欄位。例如，對於資料類型為 "character(3)" 的欄，此欄將包含值 "3"。
position	integer	欄位中錯誤的位置。
error_code	integer	錯誤代碼。
error_message	character(512)	錯誤的解釋。

範例查詢

下列查詢會顯示特定查詢複製命令的載入錯誤詳細資訊。

```
SELECT query_id,
       table_id,
       start_time,
       trim(file_name) AS file_name,
       trim(column_name) AS column_name,
       trim(column_type) AS column_type,
       trim(error_message) AS error_message
FROM sys_load_error_detail
WHERE query_id = 762949
ORDER BY start_time
LIMIT 10;
```

輸出範例。

```

query_id | table_id |          start_time          |          file_name
         | column_name | column_type |          error_message
-----+-----+-----+-----
+-----+-----+-----+-----+
+-----+-----+-----+-----+
762949 | 137885 | 2022-02-15 22:14:46.759151 | s3://load-test/copyfail/
wrong_format_000 | id | int4 | Invalid digit, Value 'a', Pos 0, Type:
Integer
762949 | 137885 | 2022-02-15 22:14:46.759151 | s3://load-test/copyfail/
wrong_format_001 | id | int4 | Invalid digit, Value 'a', Pos 0, Type:
Integer

```

SYS_LOAD_HISTORY

使用 SYS_LOAD_HISTORY 檢視 COPY 命令的詳細資訊。每一列代表一個 COPY 命令，其中包含某些欄位的累積統計資料。它同時包含執行中和已完成的 COPY 命令。

SYS 所有使用者都可看見 _LOAD_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交副本之使用者的識別碼。
query_id	bigint	副本的查詢識別碼。
transaction_id	bigint	交易識別碼。
session_id	integer	執行副本之程序的處理程序識別碼。
database_name	text	發出操作時，要將使用者連接至其中的資料庫名稱。
status	text	副本的狀態。有效值為 running、completed、aborted。

欄名稱	資料類型	描述
table_name	text	複製到的資料表的名稱。
start_time	timestamp	複製開始的時間。
end_time	timestamp	複製完成的時間。
duration	bigint	在COPY命令中花費的時間 (微秒)。
data_source	text	要複製檔案輸入的 Amazon S3 位置。
file_format	text	來源檔案格式。格式包括 csv、txt、json、avro、orc 或 parquet。
loaded_rows	bigint	複製到資料表的列數。
loaded_bytes	bigint	複製到資料表的位元組數。
source_file_count	integer	來源檔案中的檔案數。
source_file_bytes	bigint	來源檔案中的位元組數。
file_count_scanned	integer	從 Amazon S3 掃描的檔案數。
file_bytes_scanned	bigint	從 Amazon S3 中的檔案掃描的位元組數。
error_count	bigint	錯誤計數。
copy_job_id	bigint	複製任務識別碼。0 表示沒有任務識別碼。

範例查詢

下列查詢顯示特定複製命令的載入列、位元組、資料表和資料來源。

```

SELECT query_id,
       table_name,
       data_source,
       loaded_rows,
       loaded_bytes
FROM sys_load_history
WHERE query_id IN (6389,490791,441663,74374,72297)
ORDER BY query_id,
         data_source DESC;

```

輸出範例。

query_id	table_name	data_source	loaded_rows	loaded_bytes
6389	store_returns	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/store_returns/	287999764	1196240296158
72297	web_site	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/web_site/	54	43808
74374	ship_mode	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/ship_mode/	20	1320
441663	income_band	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/income_band/	20	2152
490791	customer_address	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/customer_address/	6000000	722924305

下列查詢顯示複製命令的載入列、位元組、資料表和資料來源。

```

SELECT query_id,
       table_name,
       data_source,
       loaded_rows,
       loaded_bytes
FROM sys_load_history
ORDER BY query_id DESC
LIMIT 10;

```

輸出範例。

query_id	table_name	loaded_rows	loaded_bytes	data_source
491058	web_site	54	43808	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/web_site/
490947	web_sales	720000376	22971988122819	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/web_sales/
490923	web_returns	71997522	96597496325	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/web_returns/
490918	web_page	3000	1320	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/web_page/
490907	warehouse	20	1320	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/warehouse/
490902	time_dim	86400	1320	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/time_dim/
490876	store_sales	2879987999	151666241887933	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/store_sales/
490870	store_returns	287999764	1196405607941	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/store_returns/
490865	store	1002	365507	s3://load-test/data-sources/tpcds/2.8.0/textfile/1T/store/

下列查詢顯示複製命令的每日載入列和位元組。

```
SELECT date_trunc('day',start_time) AS exec_day,
       SUM(loaded_rows) AS loaded_rows,
       SUM(loaded_bytes) AS loaded_bytes
FROM sys_load_history
GROUP BY exec_day
ORDER BY exec_day DESC;
```

輸出範例。

exec_day	loaded_rows	loaded_bytes
2022-01-20 00:00:00	6347386005	258329473070606
2022-01-19 00:00:00	19042158015	775198502204572
2022-01-18 00:00:00	38084316030	1550294469446883
2022-01-17 00:00:00	25389544020	1033271084791724
2022-01-16 00:00:00	19042158015	775222736252792


```
2022-01-15 00:00:00 | 19834245387 | 798122849155598
2022-01-14 00:00:00 | 75376544688 | 3077040926571384
```

SYS_MV_REFRESH_HISTORY

結果包括所有具體化視觀表之重新整理歷史記錄的相關資訊。結果包括重新整理類型 (例如手動或自動)，以及最近重新整理的狀態。

SYS所有使用者都可看見 `_MV_REFRESH_HISTORY`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>user_id</code>	integer	提交重新整理的使用者識別碼。
<code>session_id</code>	integer	執行具體化視觀表之程序的處理程序識別碼。
<code>transaction_id</code>	bigint	交易識別碼。
<code>database_name</code>	char(128)	包含具體化視觀表的資料庫。
<code>schema_name</code>	char(128)	具體化視觀表的結構描述。
<code>mv_id</code>	bigint	具體化視觀表的物件 ID。
<code>mv_name</code>	char(128)	具體化視觀表名稱。
<code>refresh_type</code>	char(32)	重新整理的類型，例如手動或自動。
<code>status</code>	text	重新整理的狀態。如需狀態的詳細資訊，請參閱 SVL_MV_REFRESH_STATUS 的狀態欄。
<code>start_time</code>	timestamp	重新整理的開始時間。

欄名稱	資料類型	描述
end_time	timestamp	重新整理的結束時間。
duration	bigint	重新整理具體化視觀表所花費的時間 (以微秒為單位)。

範例查詢

下列查詢顯示具體化視觀表的重新整理歷史記錄。

```
SELECT user_id,
       session_id,
       transaction_id,
       database_name,
       schema_name,
       mv_id,
       mv_name,
       refresh_type,
       status,
       start_time,
       end_time,
       duration
from sys_mv_refresh_history;
```

此查詢會傳回下列範例輸出：

```
user_id | session_id | transaction_id | database_name | schema_name |
mv_id  | mv_name    | refresh_type  | status        |
       | start_time | end_time      | duration
-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
          1 | 1073815659 |          15066 | dev           | test_stl_mv_refresh_schema |
203762 | mv_incremental | Manual        | MV was already updated
       | 2023-10-26 15:59:20.952179 | 2023-10-26 15:59:20.952866 |          687
          1 | 1073815659 |          15068 | dev           | test_stl_mv_refresh_schema |
203771 | mv_nonincremental | Manual        | MV was already updated
       | 2023-10-26 15:59:21.008049 | 2023-10-26 15:59:21.008658 |          609
```

```

1 | 1073815659 |          15070 | dev          | test_stl_mv_refresh_schema |
203779 | mv_refresh_error | Manual          | MV was already updated
      | 2023-10-26 15:59:21.064252 | 2023-10-26 15:59:21.064885 |          633
1 | 1073815659 |          15074 | dev          | test_stl_mv_refresh_schema
| 203762 | mv_incremental | Manual          | Refresh successfully updated MV
incrementally | 2023-10-26 15:59:29.693329 | 2023-10-26 15:59:43.482842 | 13789513
1 | 1073815659 |          15076 | dev          | test_stl_mv_refresh_schema |
203771 | mv_nonincremental | Manual          | Refresh successfully recomputed MV from
scratch | 2023-10-26 15:59:43.550184 | 2023-10-26 15:59:47.880833 | 4330649
1 | 1073815659 |          15078 | dev          | test_stl_mv_refresh_schema |
203779 | mv_refresh_error | Manual          | Refresh failed due to an internal error
      | 2023-10-26 15:59:47.949052 | 2023-10-26 15:59:52.494681 | 4545629

```

(6 rows)

SYS_MV_STATE

結果包括所有具體化視觀表狀態的相關資訊。它包括基底資料表資訊、結構描述屬性，以及有關最近事件的資訊，例如卸除欄。

SYS所有使用者都可看見 `_MV_STATE`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>user_id</code>	<code>bigint</code>	建立事件的使用者 ID。
<code>transaction_id</code>	<code>bigint</code>	事件的交易 ID。
<code>database_name</code>	<code>char(128)</code>	包含具體化視觀表的資料庫。
<code>event_desc</code>	<code>char(500)</code>	提示狀態變更的事件。範例值包括下列各項： <ul style="list-style-type: none"> 已變更資料欄類型 已捨棄資料欄 已重新命名資料欄 已變更結構描述名稱 小型資料表轉換

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> • TRUNCATE • Vacuum <p>請注意，此欄還有其他可能的值。</p>
start_time	timestamp	事件的開始時間。
base_table_database_name	char(128)	基底資料表的資料庫名稱。
base_table_schema	char(128)	基底資料表的結構描述。
base_table_name	char(128)	基底資料表的名稱。
mv_schema	char(128)	具體化視觀表的結構描述。
mv_name	char(128)	具體化視觀表的名稱。
state	character(32)	<p>具體化視觀表的變更後狀態如下：</p> <ul style="list-style-type: none"> • 重新運算 • 無法重新整理

範例查詢

下列查詢顯示具體化視觀表的狀態。

```
select * from sys_mv_state;
```

此查詢會傳回下列範例輸出：

```
user_id | transaction_id | database_name | event_desc | start_time
        | base_table_database_name | base_table_schema | base_table_name |
mv_schema | mv_name | state
```

```

-----+-----+-----+-----
+-----+-----+-----+-----
+-----+-----+-----+-----
106      | 12720          | tickit_db      | TRUNCATE          | 2023-07-26
14:59:12.788268 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_a1f3f862 | Recompute
106      | 12724          | tickit_db      | ALTER TABLE ALTER DISTSTYLE | 2023-07-26
14:59:51.409014 | tickit_db      | mv_schema      | test_table_58102435 |
mv_schema | materialized_view_ca746631 | Recompute
106      | 12720          | tickit_db      | Column was renamed | 2023-07-26
14:59:12.822928 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_5750a8d4 | Unrefreshable
106      | 12727          | tickit_db      | Table was renamed | 2023-07-26
15:00:08.051244 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_5750a8d4 | Unrefreshable
106      | 12720          | tickit_db      | Column was renamed | 2023-07-26
14:59:12.857755 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_5750a8d4 | Unrefreshable
106      | 12727          | tickit_db      | Table was renamed | 2023-07-26
15:00:08.051358 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_5ef0d754 | Unrefreshable
106      | 12720          | tickit_db      | TRUNCATE          | 2023-07-26
14:59:12.788159 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_5750a8d4 | Recompute
106      | 12720          | tickit_db      | Column was renamed | 2023-07-26
14:59:12.857799 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_a1f3f862 | Unrefreshable
106      | 12720          | tickit_db      | TRUNCATE          | 2023-07-26
14:59:12.788327 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_5ef0d754 | Recompute
106      | 12727          | tickit_db      | ALTER TABLE ALTER SORTKEY | 2023-07-26
15:00:08.006235 | tickit_db      | mv_schema      | test_table_58102435 |
mv_schema | materialized_view_ca746631 | Recompute
106      | 12720          | tickit_db      | Column was renamed | 2023-07-26
14:59:12.82297  | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_a1f3f862 | Unrefreshable
106      | 12727          | tickit_db      | Table was renamed | 2023-07-26
15:00:08.051321 | tickit_db      | mv_schema      | test_table_95d6d861 |
mv_schema | materialized_view_a1f3f862 | Unrefreshable

```

SYS_PROCEDURE_CALL

使用 SYS_PROCEDURE_CALL 檢視取得有關預存程序呼叫的資訊，包括開始時間、結束時間、預存程序呼叫的狀態，以及巢狀預存程序呼叫的呼叫階層。每次預存程序呼叫會接收查詢 ID。

SYS所有使用者都可看見 _PROCEDURE_CALL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
session_user_id	integer	使用者的識別碼，該使用者建立工作階段，並且是最上層預存程序呼叫的叫用者。
security_user_id	integer	使用者的識別碼，該使用者的權限用於執行預存程序中的陳述式。如果預存程序為DEFINER，則這會是預存程序的擁有者 user_id。
query_id	integer	預存程序呼叫的查詢識別碼。
query_text	char(4000)	預存程序呼叫查詢的文字。
start_time	timestamp	查詢開始執行UTC的時間。例如，時間戳記使用六位數精確度來表示小數秒。2009-06-12 11:29:19.131358。
end_time	timestamp	UTC 查詢完成執行的時間。時間戳記會使用六位數精確度來表示小數秒，例如：2009-06-12 11:29:19.131358。
status	char(10)	預存程序呼叫的狀態。系統停止或使用者取消預存程序時，

欄名稱	資料類型	描述
		會取消該值。如果預存程序呼叫執行到完成，則值為成功。
caller_procedure_query_id	integer	如果預存程序呼叫由另一個預存程序呼叫所調用，則此欄包含外層呼叫的查詢 ID。否則，欄位為 NULL。

範例查詢

下列查詢會傳回巢狀預存程序呼叫階層。

```
select query_id, datediff(seconds, start_time, end_time) as elapsed_time, status,
trim(query_text) as call, caller_procedure_query_id from sys_procedure_call;
```

輸出範例。

```
query_id | elapsed_time | status | call |
caller_procedure_query_id
-----+-----+-----+-----+-----
+-----+
      3087 |          18 | success | CALL proc_bd906c98c45443ffa165e9552056902d(1) |
          3085
      3085 |          18 | success | CALL proc_bd906c98c45443ffa165e9552056902d_2(1); |
(2 rows)
```

SYS_PROCEDURE_MESSAGES

SYS所有使用者都可看見 _PROCEDURE_MESSAGES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
transaction_id	bigint	交易識別碼。

欄名稱	資料類型	描述
query_id	integer	預存程序呼叫的查詢識別碼。
record_time	timestamp	產生訊息UTC的時間。
log_level	char(10)	產生訊息的記錄層級。可能的值為 LOG、INFO、WARNING、NOTICE和 EXCEPTION。
message	char(1024)	產生訊息的文字。
line_number	integer	產生訊息的行號。

範例查詢

下列查詢顯示 SYS_PROCEDURE_ 的範例輸出MESSAGES。

```
select transaction_id, query_id, record_time, log_level, trim(message), line_number
from sys_procedure_messages;
```

```
transaction_id | query_id |          record_time          | log_level |          btrim
               | line_number
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
      25267    |    80562 | 2023-07-17 14:38:31.910136 |  NOTICE  |
test_notice_msg_b9f1e749 |      8
      25267    |    80562 | 2023-07-17 14:38:31.910002 |    LOG    |
test_log_msg_833c7420   |      6
      25267    |    80562 | 2023-07-17 14:38:31.910111 |   INFO   |
test_info_msg_651373d9  |      7
      25267    |    80562 | 2023-07-17 14:38:31.910154 | WARNING  |
test_warning_msg_831c5747 |      9
(4 rows)
```


SYS_QUERY_DETAIL

使用 SYS_QUERY_DETAIL 在步驟層級檢視查詢的詳細資訊。每一列代表特定WLM查詢中包含詳細資訊的步驟。此檢視包含許多類型的查詢DML，例如 DDL、和 公用程式命令（例如，複製和卸載）。根據查詢類型的不同，某些欄可能不相關。例如，external_scanned_bytes 與內部資料表不相關。

SYS所有使用者都可看見 _QUERY_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	查詢識別碼。
child_query_sequence	integer	重寫使用者查詢的順序，從 1 開始。
stream_id	integer	查詢串流的資料流識別碼。
segment_id	integer	查詢執行區段的區段識別碼。
step_id	integer	區段中的步驟識別碼。
step_name	text	區段中的步驟名稱。可能的值為 aggregate、broadcast、delete、distribute、hash、hashjoininsert、limit、unique和 window。
table_id	integer	永久資料表掃描的資料表識別碼。
table_name	character(136)	正在操作之步驟的資料表名稱。

欄名稱	資料類型	描述
is_rrscan	character	指示步驟是否為掃描步驟的值。True (t) 表示使用了範圍限制掃描。
start_time	timestamp	查詢步驟開始的時間。
end_time	timestamp	查詢步驟完成的時間。
duration	bigint	步驟花費的時間 (微秒)。
提醒	text	提醒事件的說明。
input_bytes	bigint	目前步驟的輸入位元組。
input_rows	bigint	目前步驟的輸入列。
output_bytes	bigint	目前步驟的輸出位元組。
output_rows	bigint	目前步驟的輸出列。
blocks_read	bigint	步驟讀取的區塊數目。
blocks_write	bigint	步驟寫入的區塊數目。
local_read_IO	bigint	從本機磁碟快取讀取的區塊數目。
remote_read_IO	bigint	從遠端讀取的區塊數。
source	text	已掃描的資料庫物件類型。只有當列的 step_name 值為 scan 時，此欄才有值。
data_skewness	integer	輸出列在所有步驟之間分佈的偏態。它是一個在 0% 到 100% 的範圍內的數字。數字越大，分佈越不平衡。

欄名稱	資料類型	描述
time_skewness	integer	所有步驟之間執行時間分佈的偏態。它是一個在 0% 到 100% 的範圍內的數字。數字越大，分佈越不平衡。
is_active	character	步驟層級查詢的狀態。可能的值是 't'，顯示步驟正在執行中；或 'f'，表示步驟已完成執行。
spilled_block_local_disk	bigint	溢出到本機磁碟的區塊數目。
spilled_block_remote_disk	bigint	溢出到 Amazon Simple Storage Service 的區塊數。
step_attribute	character(64)	包含相關聯步驟的資訊。掃描步驟的可能值：multi-dimensional。

範例查詢

下列範例傳回 SYS_QUERY_ 的輸出DETAIL。

以下查詢顯示步驟層級的查詢中繼資料詳細資訊，包括步驟名稱、input_bytes、output_bytes、input_rows、output_rows。

```
SELECT query_id,
       child_query_sequence,
       stream_id,
       segment_id,
       step_id,
       trim(step_name) AS step_name,
       duration,
       input_bytes,
       output_bytes,
       input_rows,
       output_rows
FROM sys_query_detail
```

```
WHERE query_id IN (193929)
ORDER BY query_id,
         stream_id,
         segment_id,
         step_id DESC;
```

輸出範例。

query_id	child_query_sequence	stream_id	segment_id	step_id	step_name	duration	input_bytes	output_bytes	input_rows	output_rows
193929		2	0	0	3	hash				
37144	0	9350272	0	0	292196					
193929		5	0	0	3	hash				
9492	0	23360	0	0	1460					
193929		1	0	0	3	hash				
46809	0	9350272	0	0	292196					
193929		4	0	0	2	return				
7685	0	896	0	0	112					
193929		1	0	0	2	project				
46809	0	0	0	0	292196					
193929		2	0	0	2	project				
37144	0	0	0	0	292196					
193929		5	0	0	2	project				
9492	0	0	0	0	1460					
193929		3	0	0	2	return				
11033	0	14336	0	0	112					
193929		2	0	0	1	project				
37144	0	0	0	0	292196					
193929		1	0	0	1	project				
46809	0	0	0	0	292196					
193929		5	0	0	1	project				
9492	0	0	0	0	1460					
193929		3	0	0	1	aggregate				
11033	0	201488	0	0	14					
193929		4	0	0	1	aggregate				
7685	0	28784	0	0	14					
193929		5	0	0	0	scan				
9492	0	23360	292196	0	1460					
193929		4	0	0	0	scan				
7685	0	1344	112	0	112					

193929			2		0		0		0		scan	
37144		0		7304900		292196		292196				
193929			3		0		0		0		scan	
11033		0		13440		112		112				
193929			1		0		0		0		scan	
46809		0		7304900		292196		292196				
193929			5		0		0		-1			
9492		12288		0		0		0				
193929			1		0		0		-1			
46809		16384		0		0		0				
193929			2		0		0		-1			
37144		16384		0		0		0				
193929			4		0		0		-1			
7685		28672		0		0		0				
193929			3		0		0		-1			
11033		114688		0		0		0				

若要檢視資料庫中的資料表，從最常用到最少使用的順序，請使用下列範例。Replace (取代) `sample_data_dev` 使用您自己的資料庫。請注意，此查詢將在建立叢集時開始計算查詢，但是當資料倉儲缺少空間時，系統檢視資料不會儲存。

```
SELECT table_name, COUNT (DISTINCT query_id)
FROM SYS_QUERY_DETAIL
WHERE table_name LIKE 'sample_data_dev%'
GROUP BY table_name
ORDER BY COUNT(*) DESC;
```

```
+-----+-----+
|          table_name          | count |
+-----+-----+
| sample_data_dev.tickit.venue |     4 |
| sample_data_dev.myunload1.venue |     3 |
| sample_data_dev.tickit.listing |     1 |
| sample_data_dev.tickit.category |     1 |
| sample_data_dev.tickit.users   |     1 |
| sample_data_dev.tickit.date    |     1 |
| sample_data_dev.tickit.sales   |     1 |
| sample_data_dev.tickit.event   |     1 |
+-----+-----+
```

SYS_QUERY_HISTORY

使用 SYS_QUERY_HISTORY 檢視使用者查詢的詳細資訊。每一列代表一個使用者查詢，其中包含某些欄位的累計統計資料。此檢視包含許多類型的查詢，例如資料定義語言 (DDL)、資料處理語言 (DML)、複製、卸載和 Amazon Redshift Spectrum。它包含正在執行和已完成的查詢。

SYS所有使用者都可看見 _QUERY_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	查詢識別碼。
query_label	character(320)	查詢的簡短名稱。
transaction_id	bigint	交易識別碼。
session_id	integer	執行查詢之程序的處理程序識別碼。
database_name	character(128)	當查詢發出時，要將使用者連接至其中的資料庫名稱。
query_type	character(32)	查詢的類型，例如、SELECT、INSERT、UPDATE、UNLOAD、COPY、COMMAND、DDL、UTILITY、CTAS和 OTHER。
status	character(10)	查詢的狀態。有效值：planning、queued、running、returning、failed、canceled 和 success。
result_cache_hit	Boolean	指出查詢是否符合結果快取。

欄名稱	資料類型	描述
start_time	timestamp	開始查詢的時間。
end_time	timestamp	查詢完成的時間。
elapsed_time	bigint	查詢所花費的總時間 (微秒)。
queue_time	bigint	服務類別查詢佇列所花費的總時間 (微秒)。
execution_time	bigint	服務類別中執行的總時間 (微秒)。
error_message	character(512)	查詢失敗的原因。
returned_rows	bigint	傳回給用戶端的列數。
returned_bytes	bigint	傳回給用戶端的位元組數。
query_text	character(4000)	查詢字串。此字串可能會被截斷。
redshift_version	character(256)	查詢執行時的 Amazon Redshift 版本。
usage_limit	character(150)	查詢IDs達到的使用限制清單。
compute_type	varchar(32)	指出查詢是執行於主要叢集或並行擴展叢集。可能值為 primary (在主要叢集上執行查詢)、secondary (在次要叢集上執行查詢)，或 primary-scale (在並行叢集上執行查詢)。僅適用已佈建叢集。
compile_time	bigint	編譯查詢所花費的總時間 (微秒)。

欄名稱	資料類型	描述
planning_time	bigint	規劃查詢所花費的總時間 (微秒)。
lock_wait_time	bigint	等待關係鎖定所花費的總時間 (微秒)。
service_class_id	integer	<p>服務類別的 ID。如需服務類別的清單IDs，請前往 WLM 服務類別 IDs。</p> <p>此欄僅用於在佈建叢集上執行的查詢。對於在 Redshift Serverless 上執行的查詢，此欄包含 -1。</p>
service_class_name	character(64)	<p>服務類別名稱。</p> <p>此欄僅用於在佈建叢集上執行的查詢。對於在 Amazon Redshift Redshift Serverless 上執行的查詢，此欄為空。</p>

欄名稱	資料類型	描述
query_priority	character(20)	<p>查詢執行所在佇列的優先順序。可能的值如下：</p> <ul style="list-style-type: none"> • NULL • 最低 • low • normal • high • 最高 <p>NULL 表示查詢不支援查詢優先順序。</p> <p>此欄僅用於在佈建叢集上執行的查詢。對於在 Redshift Serverless 上執行的查詢，此欄為空。</p>
short_query_accelerated	character(10)	<p>查詢是否使用短查詢加速來加速 (SQA)。可能的值如下：</p> <ul style="list-style-type: none"> • true • false • NULL <p>此欄僅用於在佈建叢集上執行的查詢。對於在 Redshift Serverless 上執行的查詢，此欄為空。</p>
user_query_hash	character(40)	<p>從查詢產生的查詢雜湊，包括其查詢常值。具有相同查詢文字的重複查詢將具有相同的 user_query_hash 值。</p>

欄名稱	資料類型	描述
generic_query_hash	character(40)	從查詢產生的查詢雜湊，不包括其查詢常值。具有相同查詢文字但不同查詢常值的重複查詢，將具有相同的 generic_query_hash 值。
query_hash_version	integer	從查詢產生的查詢雜湊版本號碼。

範例查詢

下列查詢會傳回執行中和佇列中的查詢。

```
SELECT user_id,
       query_id,
       transaction_id,
       session_id,
       status,
       trim(database_name) AS database_name,
       start_time,
       end_time,
       result_cache_hit,
       elapsed_time,
       queue_time,
       execution_time
FROM sys_query_history
WHERE status IN ('running','queued')
ORDER BY start_time;
```

輸出範例。

```
user_id | query_id | transaction_id | session_id | status | database_name |
start_time | end_time | result_cache_hit | elapsed_time |
queue_time | execution_time
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

```

101 | 760705 | 852337 | 1073832321 | running | tpcds_1t |
2022-02-15 19:03:19.67849 | 2022-02-15 19:03:19.739811 | f |
61321 | 0 | 0

```

下列查詢會傳回特定查詢的查詢開始時間、結束時間、佇列時間、經歷時間、計劃時間及其他中繼資料。

```

SELECT user_id,
       query_id,
       transaction_id,
       session_id,
       status,
       trim(database_name) AS database_name,
       start_time,
       end_time,
       result_cache_hit,
       elapsed_time,
       queue_time,
       execution_time,
       planning_time,
       trim(query_text) as query_text
FROM sys_query_history
WHERE query_id = 3093;

```

輸出範例。

```

user_id | query_id | transaction_id | session_id | status | database_name |
start_time | end_time | result_cache_hit | elapsed_time |
queue_time | execution_time | planning_time | query_text
-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
106 | 3093 | 11759 | 1073750146 | success | dev |
2023-03-16 16:53:17.840214 | 2023-03-16 16:53:18.106588 | f |
266374 | 0 | 105725 | 136589 | select count(*) from item;

```

下列查詢列出 10 個最近的SELECT查詢。

```

SELECT query_id,
       transaction_id,
       session_id,

```

```

    start_time,
    elapsed_time,
    queue_time,
    execution_time,
    returned_rows,
    returned_bytes
FROM sys_query_history
WHERE query_type = 'SELECT'
ORDER BY start_time DESC limit 10;

```

輸出範例。

query_id	transaction_id	session_id	start_time	elapsed_time
queue_time	execution_time	returned_rows	returned_bytes	
526532	61093	1073840313	2022-02-09 04:43:24.149603	520571
0	481293	1	3794	
526520	60850	1073840313	2022-02-09 04:38:27.24875	635957
0	596601	1	3679	
526508	60803	1073840313	2022-02-09 04:37:51.118835	563882
0	503135	5	17216	
526505	60763	1073840313	2022-02-09 04:36:48.636224	649337
0	589823	1	652	
526478	60730	1073840313	2022-02-09 04:36:11.741471	14611321
0	14544058	0	0	
526467	60636	1073840313	2022-02-09 04:34:11.91463	16711367
0	16633767	1	575	
511617	617946	1074009948	2022-01-20 06:21:54.44481	9937090
0	9899271	100	12500	
511603	617941	1074259415	2022-01-20 06:21:45.71744	8065081
0	7582500	100	8889	
511595	617935	1074128320	2022-01-20 06:21:44.030876	1051270
0	1014879	1	72	
511584	617931	1074030019	2022-01-20 06:21:42.764088	609033
0	485887	100	8438	

下列查詢會顯示每日選擇查詢計數和平均查詢經歷時間。

```

SELECT date_trunc('day',start_time) AS exec_day,
       status,
       COUNT(*) AS query_cnt,

```

```

    AVG(datediff (microsecond,start_time,end_time)) AS elapsed_avg
FROM sys_query_history
WHERE query_type = 'SELECT'
AND start_time >= '2022-01-14'
AND start_time <= '2022-01-18'
GROUP BY exec_day,
         status
ORDER BY exec_day,
         status;

```

輸出範例。

exec_day	status	query_cnt	elapsed_avg
2022-01-14 00:00:00	success	5253	56608048
2022-01-15 00:00:00	success	7004	56995017
2022-01-16 00:00:00	success	5253	57016363
2022-01-17 00:00:00	success	5309	55236784
2022-01-18 00:00:00	success	8092	54355124

下列查詢會顯示日常查詢的經歷時間效能。

```

SELECT distinct date_trunc('day',start_time) AS exec_day,
       query_count.cnt AS query_count,
       Percentile_cont(0.5) within group(ORDER BY elapsed_time) OVER (PARTITION BY
exec_day) AS P50_runtime,
       Percentile_cont(0.8) within group(ORDER BY elapsed_time) OVER (PARTITION BY
exec_day) AS P80_runtime,
       Percentile_cont(0.9) within group(ORDER BY elapsed_time) OVER (PARTITION BY
exec_day) AS P90_runtime,
       Percentile_cont(0.99) within group(ORDER BY elapsed_time) OVER (PARTITION BY
exec_day) AS P99_runtime,
       Percentile_cont(1.0) within group(ORDER BY elapsed_time) OVER (PARTITION BY
exec_day) AS max_runtime
FROM sys_query_history
LEFT JOIN (SELECT date_trunc('day',start_time) AS day, count(*) cnt
          FROM sys_query_history
          WHERE query_type = 'SELECT'
          GROUP by 1) query_count
ON date_trunc('day',start_time) = query_count.day
WHERE query_type = 'SELECT'
ORDER BY exec_day;

```

輸出範例。

```

      exec_day      | query_count | p50_runtime | p80_runtime | p90_runtime |
p99_runtime | max_runtime
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
2022-01-14 00:00:00 |      5253 | 16816922.0 | 69525096.0 | 158524917.8 |
486322477.52 | 1582078873.0
2022-01-15 00:00:00 |      7004 | 15896130.5 | 71058707.0 | 164314568.9 |
500331542.07 | 1696344792.0
2022-01-16 00:00:00 |      5253 | 15750451.0 | 72037082.2 | 159513733.4 |
480372059.24 | 1594793766.0
2022-01-17 00:00:00 |      5309 | 15394513.0 | 68881393.2 | 160254700.0 |
493372245.84 | 1521758640.0
2022-01-18 00:00:00 |      8092 | 15575286.5 | 68485955.4 | 154559572.5 |
463552685.39 | 1542783444.0
2022-01-19 00:00:00 |      5860 | 16648747.0 | 72470482.6 | 166485138.2 |
492038228.67 | 1693483241.0
2022-01-20 00:00:00 |      1751 | 15422072.0 | 69686381.0 | 162315385.0 |
497066615.00 | 1439319739.0
2022-02-09 00:00:00 |         13 | 6382812.0 | 17616161.6 | 21197988.4 |
23021343.84 | 23168439.0

```

下列查詢顯示查詢類型分佈。

```

SELECT query_type,
       COUNT(*) AS query_count
FROM sys_query_history
GROUP BY query_type
ORDER BY query_count DESC;

```

輸出範例。

```

query_type | query_count
-----+-----
UTILITY   |      134486
SELECT    |      38537
DDL       |         4832
OTHER     |          768
LOAD     |          768
CTAS     |          748

```

COMMAND		92
---------	--	----

下列範例顯示多個查詢之間的查詢雜湊結果差異。請遵守下列查詢：

```
CREATE TABLE test_table (col1 INT);

INSERT INTO test_table VALUES (1),(2);

SELECT * FROM test_table;

SELECT * FROM test_table;

SELECT col1 FROM test_table;

SELECT * FROM test_table WHERE col1=1;

SELECT * FROM test_table WHERE col1=2;

SELECT query_id, TRIM(user_query_hash) AS user_query_hash, TRIM(generic_query_hash)
  AS generic_query_hash, TRIM(query_text) AS text FROM sys_query_history ORDER BY
  start_time
DESC LIMIT 10;
```

以下是範例輸出：

```
query_id | user_query_hash | generic_query_hash | text
-----+-----+-----+-----
24723049 | oPuFtjEPLTs=   | oPuFtjEPLTs=       | select query_id,
  trim(user_query_hash) as user_query_hash, trim(generic_query_hash) as
  generic_query_hash, query_hash_version, trim(query_text) as text from
  sys_query_history order by start_time\r\ndesc limit 20
24723045 | Gw2Kwdd8m2I=   | IwFRu8/XAKI=       | select * from test_table where col1=2
  limit 100
24723041 | LNw2vx0GDxo=   | IwFRu8/XAKI=       | select * from test_table where col1=1
  limit 100
24723036 | H+qep/c82Y8=   | H+qep/c82Y8=       | select col1 from test_table limit 100
24723033 | H+qep/c82Y8=   | H+qep/c82Y8=       | select * from test_table limit 100
24723029 | H+qep/c82Y8=   | H+qep/c82Y8=       | select * from test_table limit 100
24723023 | 50sirx9E1hU=   | u036Z1a/QYs=       | insert into test_table values (1),(2)
24723021 | YSVnlivZHeo=   | YSVnlivZHeo=       | create table test_table (col1 int)
```

SELECT * FROM test_table; 和 SELECT col1 FROM test_table; 具有相同的 user_query_hash 值，因為 test_table 只有一欄。SELECT * FROM test_table WHERE col1=1; 和 SELECT * FROM test_table WHERE col1=2; 具有不同的 user_query_hash 值，但相同的 generic_query_hash 值，因為兩個查詢在查詢常值 1 和 2 之外是相同的。

SYS_QUERY_TEXT

使用 SYS_QUERY_TEXT 檢視所有查詢的查詢文字。每一列代表查詢的查詢文字，最多 4000 個字元，以序列號 0 開始。當查詢陳述式包含超過 4000 個字元時，會藉由遞增每個列的序號來記錄陳述式的其他列。此檢視會記錄所有使用者查詢文字，例如 DDL、公用程式、Amazon Redshift 查詢和僅限領導者節點的查詢。

SYS 所有使用者都可看見 _QUERY_TEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
query_id	bigint	查詢識別碼。
transaction_id	bigint	與陳述式關聯的交易 ID。
session_id	integer	執行查詢之工作階段的處理程序識別碼。
start_time	timestamp	查詢開始的時間。
sequence	integer	當單一陳述式包含不只 4000 個字元時，會將該陳述式的其他列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character (4000)	以 4000 個字元增量的 SQL 查詢文字。此欄位可能包含特殊字元，例如反斜線 () 和換行符號 ()。

範例查詢

下列查詢會傳回執行中和佇列中的查詢。

```
SELECT user_id,
       query_id,
       transaction_id,
       session_id, start_time,
       sequence, trim(text) as text from sys_query_text
ORDER BY sequence;
```

輸出範例。

```
user_id | query_id | transaction_id | session_id |          start_time          |
sequence |          text          |
-----+-----+-----+-----+-----+-----
+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
100 |      4      |      1396      | 1073750220 | 2023-04-28 16:44:55.887184 |
0 | SELECT trim(text) as text, sequence FROM sys_query_text WHERE query_id =
pg_last_query_id() AND user_id > 1 AND start
_time > '2023-04-28 16:44:55.922705+00:00'::timestamp order by sequence;
```

下列查詢會傳回已從資料庫中的群組授與或撤銷的許可。

```
SELECT
  SPLIT_PART(text, ' ', 1) as grantrevoke,
  SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'GROUP'))), ' ', 2) as group,
  SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), ' '))), 'ON', 1) as type,
  SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'ON'))), ' ', 2) || ' ' ||
  SPLIT_PART((SUBSTRING(text, STRPOS(UPPER(text), 'ON'))), ' ', 3) as entity
FROM SYS_QUERY_TEXT
WHERE (text LIKE 'GRANT%' OR text LIKE 'REVOKE%') AND text LIKE '%GROUP%';
```

```
+-----+-----+-----+-----+
| grantrevoke | group   | type   | entity |
+-----+-----+-----+-----+
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
| GRANT       | bi_group | USAGE  | TABLE t1 |
| GRANT       | bi_group | SELECT | TABLE t1 |
```

```
| GRANT          | bi_group | SELECT | TABLE t1 |
+-----+-----+-----+-----+
```

SYS_RESTORE_LOG

使用 SYS_RESTORE_LOG 在傳統調整RA3節點大小期間監控叢集中每個資料表的遷移進度。它會在調整大小作業期間擷取資料遷移的歷史輸送量。如需傳統調整大小至RA3節點的詳細資訊，請參閱[傳統調整大小](#)。

SYS只有超級使用者可以看到 RESTORE__LOG。

資料表欄

欄名稱	資料類型	描述
event_time	timestamp	指出日誌項目何時記錄的時間戳記。
database_name	char(128)	資料庫的名稱。
schema_name	char(128)	結構描述的名稱。
table_name	char(128)	資料表的名稱。
table_id	integer	資料表的 ID。
動作	char(128)	輸入時所採取的動作。值可包括：已啟動移轉、檢查點、已繼續、已完成、已取消或重設。
table_size	long	資料表的大小
total_data_processed	long	到目前為止，資料表處理的資料大小 (以 MB 為單位)。
delta_data_processed	long	自上次 event_time 更新後所處理的資料大小，以 MB 為單位。這可協助您判斷自先前記錄的時間間隔後，已處理多少

欄名稱	資料類型	描述
		資料。您可以將其與 <code>table_size</code> 進行比較，以了解資料處理的速度。
<code>message</code>	<code>char(512)</code>	動作欄中值的詳細說明。
<code>redistribution_type</code>	<code>char(32)</code>	資料表的重新發佈類型。KEY 轉換或EVEN重新平衡任務。如需分佈樣式的詳細資訊，請參閱 分佈樣式 。

範例查詢

下列查詢會使用 `SYS_RESTORE_` 計算資料處理的輸送量LOG。

```
SELECT
  ROUND(sum(delta_data_processed) / 1024.0, 2) as data_processed_gb,
  ROUND(datediff(sec, min(event_time), max(event_time)) / 3600.0, 2) as duration_hr,
  ROUND(data_processed_gb/duration_hr, 2) as throughput_gb_per_hr
from sys_restore_log;
```

輸出範例。

```
data_processed_gb | duration_hr | throughput_gb_per_hr
-----+-----+-----
                0.91 |          8.37 |                0.11
(1 row)
```

下列查詢會顯示所有重新分發類型。

```
SELECT * from sys_restore_log ORDER BY event_time;
```

```
database_name | schema_name | table_name | table_id |
action        | total_data_processed | delta_data_processed | event_time
              | table_size | message | redistribution_type
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
```

```

dev          | schemaaaa877096d844d | customer_key          | 106424 |
Redistribution started          | 0 |
02:18:00.744977 | 325 | | Restore Distkey Table
dev          | schemaaaa877096d844d | dp30907_t2_autokey    | 106430 |
Redistribution started          | 0 |
02:18:02.756675 | 90 | | Restore Distkey Table
dev          | schemaaaa877096d844d | dp30907_t2_autokey    | 106430 |
Redistribution completed        | 90 | 90 | 2024-01-05
02:23:30.643718 | 90 | | Restore Distkey Table
dev          | schemaaaa877096d844d | customer_key          | 106424 |
Redistribution completed        | 325 | 325 | 2024-01-05
02:23:45.998249 | 325 | | Restore Distkey Table
dev          | schemaaaa877096d844d | dp30907_t1_even       | 106428 |
Redistribution started          | 0 |
02:23:46.083849 | 30 | | Rebalance Disteven Table
dev          | schemaaaa877096d844d | dp30907_t5_auto_even  | 106436 |
Redistribution started          | 0 |
02:23:46.855728 | 45 | | Rebalance Disteven Table
dev          | schemaaaa877096d844d | dp30907_t5_auto_even  | 106436 |
Redistribution completed        | 45 | 45 | 2024-01-05
02:24:16.343029 | 45 | | Rebalance Disteven Table
dev          | schemaaaa877096d844d | dp30907_t1_even       | 106428 |
Redistribution completed        | 30 | 30 | 2024-01-05
02:24:20.584703 | 30 | | Rebalance Disteven Table
dev          | schemaefd028a2a48a4c | customer_even          | 130512 |
Redistribution started          | 0 |
04:54:55.641741 | 190 | | Restore Disteven Table
dev          | schemaefd028a2a48a4c | customer_even          | 130512 |
Redistribution checkpointed     | 29.4342113157737 | 29.4342113157737 | 2024-01-05
04:55:04.770696 | 190 | | Restore Disteven Table
(8 rows)

```

SYS_RESTORE_STATE

使用 `SYS_RESTORE_STATE` 在傳統調整大小期間監控每個資料表的遷移進度。這在目標節點類型為時特別適用RA3。如需傳統調整大小至RA3節點的詳細資訊，請參閱[傳統調整大小](#)。

SYS只有超級使用者可以看到 `RESTORE__STATE`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交查詢之使用者的識別碼。
database_name	char(64)	資料庫中的資料表名稱。
schema_id	integer	資料表的結構描述 ID。
table_id	integer	資料表的 ID。
table_name	char(128)	資料表的名稱。
redistribution_status	char(128)	資料表重新分佈進度的狀態。可能值為 Completed、In progress 及 Pending。
percentage_redistributed	float	資料表重新分佈進度的百分比。可能值為 0 到 100%。例如，值 25 表示 25% 的資料已重新分配。
redistribution_type	char(32)	資料表的重新發佈類型。KEY 轉換或 EVEN 重新平衡任務。如需分佈樣式的詳細資訊，請參閱 分佈樣式 。

範例查詢

下列查詢會傳回執行中和佇列查詢的記錄。

```
SELECT * FROM sys_restore_state;
```

輸出範例。

```
userid | database_name | schema_id | table_id | table_name | redistribution_status
| precentage_redistributed | redistribution_type
```

```

-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
  1 | test1 | 124865 | 124878 | customer_key_4 | Pending
  | 0 | | Rebalance Disteven Table
  1 | dev | 124865 | 124874 | customer_key_3 | Pending
  | 0 | | Rebalance Disteven Table
  1 | dev | 124865 | 124870 | customer_key_2 | Completed
  | 100 | | Rebalance Disteven Table
  1 | dev | 124865 | 124866 | customer_key_1 | In progress
  | 13.52 | | Restore Distkey Table

```

以下提供資料處理狀態。

```

SELECT
    redistribution_status, ROUND(SUM(block_count) / 1024.0, 2) AS total_size_gb
FROM sys_restore_state sys inner join stv_tbl_perm stv
    on sys.table_id = stv.id
GROUP BY sys.redistribution_status;

```

輸出範例。

```

redistribution_status | total_size_gb
-----+-----
Completed            |          0.07
Pending              |          0.71
In progress          |          0.20
(3 rows)

```

SYS_SCHEMA_QUOTA_VIOLATIONS

超過結構描述配額時，記錄出現次數、交易 ID 和其他有用的資訊。此系統資料表為 [STL_SCHEMA_QUOTA_VIOLATIONS](#) 的翻譯。

所有使用者都可看見 r_SYSSHEMA_QUOTA_VIOLATIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
owner_id	integer	結構描述擁有者的 ID。

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
transaction_id	bigint	與陳述式關聯的交易 ID。
session_id	integer	與陳述式相關聯的處理程序 ID。
schema_id	integer	命名空間或結構描述 ID。
schema_name	character (128)	命名空間或結構描述名稱。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。
record_time	沒有時區的時間戳記	發生違規的時間。

範例查詢

下列查詢顯示配額違規的結果：

```
SELECT user_id, TRIM(schema_name) "schema_name", quota, disk_usage, record_time FROM
sys_schema_quota_violations WHERE SCHEMA_NAME = 'sales_schema' ORDER BY timestamp DESC;
```

此查詢會傳回所指定結構描述的下列範例輸出：

```
user_id| schema_name | quota | disk_usage | record_time
-----+-----+-----+-----+-----
104    | sales_schema | 2048  | 2798      | 2020-04-20 20:09:25.494723
(1 row)
```

SYS_SERVERLESS_USAGE

使用 SYS_SERVERLESS_USAGE 檢視 Amazon Redshift Serverless 資源用量的詳細資訊。此系統檢視不適用於已佈建的 Amazon Redshift 叢集。

此檢視包含無伺服器使用量摘要，包括使用多少運算容量來處理查詢，以及使用的 Amazon Redshift 託管儲存量 (1 分鐘精細程度)。運算容量是以 Redshift 處理單位 (RPU) 為單位，並針對每秒以 RPU-秒為單位執行的工作負載進行計量。RPU 用於處理資料倉儲中載入之資料的查詢、從 Amazon S3 資料湖中查詢，或使用聯合查詢從操作資料庫存取的查詢。Amazon Redshift Serverless 會將 SYS_SERVERLESS_USAGE 中的資訊保留 7 天。

如需運算成本帳單的範例，請參閱 [Amazon Redshift Serverless 的計費](#)。

SYS 只有超級使用者可以看到 _SERVERLESS_USAGE。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
start_time	timestamp	間隔開始的時間。
end_time	timestamp	間隔完成的時間。
compute_seconds	double precision	在此時間間隔內耗用的累積運算單位 (RPU) 秒。此值會計入為帳戶配置的基本 RPU 容量。
compute_capacity	double precision	在此時間間隔內配置的運算單位 (Redshift 處理單位，或 RPU) 平均數量。 Compute_capacity 值可以動態變更。
data_storage	integer	此時間間隔內使用的平均資料儲存空間 (MB)。 當從資料庫載入或刪除資料時，使用的資料儲存可以動態變化。

欄名稱	資料類型	描述
cross_region_transferred_data	integer	此時間間隔內，跨區域資料共用所傳輸的累計資料 (以位元組為單位)。
charged_seconds	integer	在此時間間隔內充電的累積運算單位 (RPU) 秒數。這是交易結束後計算，因此在交易執行時可以是 0。使用 charged_seconds 來計算 Amazon Redshift Serverless 工作群組的成本。此值會考慮為 Amazon Redshift Serverless 工作群組配置的 RPU 容量。

使用須知

- 在某些情況下，compute_seconds 為 0，但 charged_seconds 大於 0，反之亦然。這是由於在系統檢視中記錄資料的方式所產生的正常行為。若要更準確地呈現無伺服器使用情況詳細資訊，我們建議您彙總資料。

範例

若要透過查詢 charged_seconds 取得時間間隔內使用 RPU 的總時數費用，請執行下列查詢：

```
select trunc(start_time) "Day",
(sum(charged_seconds)/3600::double precision) * <Price for 1 RPU> as cost_incurred
from sys_serverless_usage
group by 1
order by 1
```

請注意，間隔期間可能會有閒置時間。閒置時間不會新增至 RPU 已耗用。

SYS_SESSION_HISTORY

使用 SYS_SESSION_HISTORY 檢視目前作用中工作階段和工作階段歷史記錄的相關資訊。

SYS所有使用者都可看見 `_SESSION_HISTORY`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
<code>user_id</code>	<code>char(50)</code>	產生項目之使用者的識別碼。
<code>session_id</code>	<code>integer</code>	與陳述式相關聯之工作階段的名稱。
<code>database_name</code>	<code>char(50)</code>	資料庫的名稱。
<code>status</code>	<code>char</code>	工作階段的狀態。可能值為 <code>active</code> 、 <code>timed out</code> 及 <code>closed</code> 。
<code>session_timeout</code>	<code>integer</code>	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。
<code>start_time</code>	<code>timestamp</code>	建立連線的時間戳記。
<code>end_time</code>	<code>timestamp</code>	連線停止的時間戳記。

範例

以下是 `SYS_SESSION_` 的範例輸出 HISTORY。

```
select * from sys_session_history;
 user_id | session_id | database_name | status | session_timeout |
 start_time          | end_time
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
          1 | 1073971370 | dev          | closed | 0              |
15:50:12.030104 | 2023-07-17 15:50:12.123218
          1 | 1073979694 | dev          | closed | 0              |
15:50:24.117947 | 2023-07-17 15:50:24.131859
          1 | 1073873049 | dev          | closed | 0              |
15:49:29.067398 | 2023-07-17 15:49:29.070294
          1 | 1073873086 | database18127a4a | closed | 0              |
15:49:29.119018 | 2023-07-17 15:49:29.125925
```

```

1 | 1073832112 | dev | closed | 0 | 2023-07-17
15:49:29.164688 | 2023-07-17 15:49:29.179631
1 | 1073987697 | dev | closed | 0 | 2023-07-17
15:49:29.26749 | 2023-07-17 15:49:29.273034
1 | 1073922429 | dev | closed | 0 | 2023-07-17
15:49:33.35315 | 2023-07-17 15:49:33.367499
1 | 1073766783 | dev | closed | 0 | 2023-07-17
15:49:45.38237 | 2023-07-17 15:49:45.396902
1 | 1073807506 | dev | active | 0 | 2023-07-17
15:51:48 |

```

SYS_SPATIAL_SIMPLIFY

您可以使用 COPY 命令查詢系統檢視 SYS_SPATIAL_SIMPLIFY，以取得簡化空間幾何物件的相關資訊。當您 COPY 在 shapefile 上使用時，您可以指定 SIMPLIFY AUTO、tolerance SIMPLIFY 和 SIMPLIFY AUTO max_tolerance 擷取選項。簡化的結果會在 SYS_SPATIAL_SIMPLIFY 系統檢視中摘要。

設定 SIMPLIFY AUTO max_tolerance 時，此檢視會包含超過大小上限的每個幾何的列。設定 SIMPLIFY tolerance 時，會儲存整個 COPY 操作的一列。此列參考 COPY 查詢 ID 和指定的簡化容差。

如需有關載入 shapefile 的相關資訊，請參閱 [將 Shapefile 載入 Amazon Redshift](#)。

SYS 所有使用者都可看見 _SPATIAL_SIMPLIFY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
query_id	bigint	產生此資料列的查詢（COPY 命令）ID。
line_number	bigint	指定 COPY SIMPLIFY AUTO 選項時，此值是 shapefile 中簡化記錄的記錄編號。
maximum_tolerance	double precision	COPY 命令中指定的距離容差值。這是使用 SIMPLIFY AUTO 選項的最大公差值，或使用 SIMPLIFY 選項的固定公差值。

欄名稱	資料類型	描述
initial_size	bigint	簡化之前 GEOMETRY 資料值的大小 (以位元組為單位)。
簡化	char(1)	指定 COPY SIMPLIFY AUTO 選項時，t 如果幾何已成功簡化，f 否則。如果使用給定的最大公差進行簡化後，幾何的大小仍大於幾何大小上限，則幾何可能無法成功簡化。
final_size	bigint	指定 COPY SIMPLIFY AUTO 選項時，這是簡化後的幾何位元組大小。
final_tolerance	double precision	為簡化選擇的最終公差。

範例查詢

下列查詢會傳回 COPY 簡化的記錄清單。

```
SELECT * FROM sys_spatial_simplify;
```

```

query_id | line_number | maximum_tolerance | initial_size | simplified | final_size |
final_tolerance
-----+-----+-----+-----+-----+-----+
+-----+
      20  |    1184704 |           -1 |    1513736 | t         |    1008808 |
1.276386653895e-05
      20  |    1664115 |           -1 |    1233456 | t         |    1023584 |
6.11707814796635e-06

```

SYS_STREAM_SCAN_ERRORS

記錄透過串流擷取載入之記錄的錯誤。

SYS 所有使用者都可看見 `_STREAM_SCAN_ERRORS`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
external_schema_name	character (128)	Kinesis 串流或 Amazon MSK主題結構描述的名稱。區分大小寫。
stream_name	character (255)	串流或主題的名稱。區分大小寫。
mv_name	character (128)	相關聯具體化視觀表的名稱。如果沒有，則為空。區分大小寫。
transaction_id	bigint	交易 ID。
query_id	bigint	查詢 ID。
stream_timestamp_type	character (1)	串流時間戳記的類型。區分大小寫。
stream_timestamp	沒有時區的時間戳記	記錄到達的時間。
record_time	沒有時區的時間戳記	記錄錯誤訊息的時間。
partition_id	character (128)	分割區/碎片 ID。區分大小寫。
position	character (128)	記錄的位置。這相當於 Kinesis 中的序號或 Amazon 中的偏移MSK。區分大小寫。
error_code	integer	錯誤代碼。

欄名稱	資料類型	描述
error_reason	character (128)	錯誤原因。區分大小寫。

SYS_STREAM_SCAN_STATES

記錄透過串流擷取載入之記錄的掃描狀態。

SYS所有使用者都可看見 `_STREAM_SCAN_STATES`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
external_schema_name	character (128)	外部結構描述名稱。區分大小寫。
stream_name	character (255)	串流名稱。區分大小寫。
mv_name	character (128)	相關聯具體化視觀表的名稱。如果沒有，則為空。區分大小寫。
transaction_id	bigint	交易 ID。
query_id	bigint	查詢 ID。
record_time	沒有時區的時間戳記	記錄資料的時間。

欄名稱	資料類型	描述
partition_id	character (128)	分割區或碎片 ID。區分大小寫。
latest_position	character (128)	批次中最後讀取的記錄的位置。這相當於 Kinesis 中的序號或 Amazon 中的偏移MSK。區分大小寫。
scanned_rows	bigint	批次中掃描的記錄數。
skipped_rows	bigint	批次中略過的記錄數。
scanned_bytes	bigint	批次中掃描的位元組數。
stream_record_time_min	沒有時區的時間戳記	批次中最早記錄的 Kinesis 或 Amazon MSK到達時間。
stream_record_time_max	沒有時區的時間戳記	批次中最新記錄的 Kinesis 或 Amazon MSK抵達時間。

下列查詢會顯示特定查詢的串流和主題資料。

```
select
  query_id,mv_name::varchar,external_schema_name::varchar,stream_name::varchar,sum(scanned_rows)
  total_records,
sum(scanned_bytes) total_bytes from sys_stream_scan_states where query in
(5401180,8601939) group by 1,2,3,4;
```

```

query_id | mv_name      | external_schema_name | stream_name | total_records |
total_bytes
-----+-----+-----+-----+-----
+-----+
5401180  | kinesistest | kinesis              | kinesisstream | 1493255696 |
3209006490704
```

```

8601939 | msktest | msk | mskstream | 14677023 |
31056580668
(2 rows)

```

SYS_TRANSACTION_HISTORY

使用 SYS_TRANSACTION_HISTORY 在追蹤查詢時查看交易的詳細資訊。

SYS 只有超級使用者可以看到 _TRANSACTION_HISTORY。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目的使用者之 ID。
transaction_id	bigint	交易的 ID。
isolation_level	text	交易的隔離層級。可能值為 Serializable 和 Snapshot Isolation。
status	text	交易的狀態。可能的狀態為 committed 和 rollback。
transaction_start_time	timestamp	交易的開始時間。
commit_start_time	timestamp	遞交的開始時間。
commit_end_time	timestamp	提交的結束時間。
blocks_committed	bigint	必須寫入為此遞交一部分的區塊數目。
undo_transaction_id	bigint	復原交易的 ID (如果有任何交易已復原)。否則，此值為 -1。

範例查詢

```
select * from sys_transaction_history order by transaction_start_time desc;
```

user_id	transaction_id	isolation_level	status	transaction_start_time	commit_start_time	commit_end_time	blocks_committed	undo_transaction_id
100	1310	Serializable	committed	2023-08-27 21:03:11.822205	2023-08-28 21:03:11.825287	2023-08-28 21:03:11.854883	17	-1
101	1345	Serializable	committed	2023-08-27 21:03:12.000278	2023-08-28 21:03:12.003736	2023-08-28 21:03:12.030061	17	-1
102	1367	Serializable	committed	2023-08-27 21:03:12.1532	2023-08-28 21:03:12.156124	2023-08-28 21:03:12.186468	17	-1
100	1370	Serializable	committed	2023-08-27 21:03:12.199316	2023-08-28 21:03:12.204854	2023-08-28 21:03:12.238186	24	-1
100	1408	Serializable	committed	2023-08-27 21:03:53.891107	2023-08-28 21:03:53.894825	2023-08-28 21:03:53.927465	17	-1
100	1409	Serializable	rolledback	2023-08-27 21:03:53.936431	2000-01-01 00:00:00	2023-08-28 21:04:08.712532	0	1409
101	1415	Serializable	committed	2023-08-27 21:04:24.283188	2023-08-28 21:04:24.289196	2023-08-28 21:04:24.374318	25	-1
101	1416	Serializable	committed	2023-08-27 21:04:24.38818	2023-08-28 21:04:24.391688	2023-08-28 21:04:24.415135	17	-1
100	1417	Serializable	rolledback	2023-08-27 21:04:24.424252	2000-01-01 00:00:00	2023-08-28 21:04:28.354826	0	1417
101	1418	Serializable	rolledback	2023-08-27 21:04:24.425195	2000-01-01 00:00:00	2023-08-28 21:04:28.680355	0	1418
100	1420	Serializable	committed	2023-08-27 21:04:28.697607	2023-08-28 21:04:28.702374	2023-08-28 21:04:28.735541	23	-1

```

101 |          1421 | Serializable | committed | 2023-08-27 21:04:28.744854 |
2023-08-28 21:04:28.749344 | 2023-08-28 21:04:28.779029 |          23 |
-1
101 |          1423 | Serializable | committed | 2023-08-27 21:04:28.78942 |
2023-08-28 21:04:28.791556 | 2023-08-28 21:04:28.817485 |          16 |
-1
100 |          1430 | Serializable | committed | 2023-08-27 21:04:28.917788 |
2023-08-28 21:04:28.919993 | 2023-08-28 21:04:28.944812 |          16 |
-1
102 |          1494 | Serializable | committed | 2023-08-27 21:04:37.029058 |
2023-08-28 21:04:37.033137 | 2023-08-28 21:04:37.062001 |          16 |
-1

```

SYS_UDF_LOG

記錄使用者定義函數（UDF）執行期間產生的系統定義錯誤和警告訊息。

SYS只有超級使用者可以看到 UDF__LOG。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
query_id	bigint	查詢識別碼。
function_name	text	使用者定義函數的名稱。
record_time	timestamp	建立記錄的時間。
sequence	integer	單一記錄訊息的順序。
message	text	記錄訊息文字。

範例查詢

下列範例顯示UDFs如何處理系統定義的錯誤。第一個區塊顯示傳回引數反轉之UDF函數的定義。當您執行函數並提供 0 做為引數時，函數會傳回錯誤。最後一個陳述式傳回以 SYS_UDF_ 記錄的錯誤訊息 LOG。

```
-- Create a function to find the inverse of a number.
CREATE OR REPLACE FUNCTION f_udf_inv(a int)

RETURNS float

IMMUTABLE AS $$return 1/a

$$ LANGUAGE plpythonu;

-- Run the function with 0 to create an error.
Select f_udf_inv(0);

-- Query SYS_UDF_LOG to view the message.
Select query_id, record_time, message::varchar from sys_udf_log;
```

query_id	record_time	message
2211	2023-08-23 15:53:11.360538	ZeroDivisionError: integer division or modulo by zero line 2, in f_udf_inv\n return 1/a\n

下列範例會將記錄和警告訊息新增至 `SYS_UDF_LOG`，UDF 以便除以零操作會產生警告訊息，而不是以錯誤訊息停止。

```
-- Create a function to find the inverse of a number and log a warning if you input 0.
CREATE OR REPLACE FUNCTION f_udf_inv_log(a int)
  RETURNS float IMMUTABLE
  AS $$
  import logging
  logger = logging.getLogger() #get root logger
  if a==0:
    logger.warning('You attempted to divide by zero.\nReturning zero instead of error.\n')
    return 0
  else:
    return 1/a
  $$ LANGUAGE plpythonu;

-- Run the function with 0 to trigger the warning.
Select f_udf_inv_log(0);

-- Query SYS_UDF_LOG to view the message.
```

```
Select query_id, record_time, message::varchar from sys_udf_log;
```

```

query_id |          record_time          |          message
-----+-----
+-----+-----
      0  | 2023-08-23 16:10:48.833503 | WARNING: You attempted to divide by zero.
\nReturning zero instead of error.\n

```

SYS_UNLOAD_DETAIL

使用 SYS_UNLOAD_DETAIL 檢視 UNLOAD 操作的詳細資訊。它會記錄 UNLOAD 陳述式所建立的每個檔案的一列。例如，如果 UNLOAD 建立 12 個檔案，SYS_UNLOAD_DETAIL 將包含 12 個對應的資料列。

SYS 所有使用者都可看見 _UNLOAD_DETAIL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	產生項目之使用者的識別碼。
query_id	integer	UNLOAD 命令的查詢識別碼。
session_id	integer	與查詢陳述式相關聯的處理程序 ID。
transaction_id	bigint	與查詢陳述式關聯的交易 ID。
file_name	character (1280)	檔案的完整 Amazon S3 物件路徑。
start_time	timestamp	交易開始的時間。
end_time	timestamp	交易完成的時間。
line_count	bigint	已卸載至檔案的行數 (列數)。
transfer_size	bigint	已傳輸的位元組數目。

欄名稱	資料類型	描述
file_format	character (10)	卸載檔案的檔案格式。

範例查詢

下列查詢顯示卸載查詢詳細資訊，包括卸載命令的格式、列和檔案計數。

```
select query_id, substring(file_name, 0, 50), transfer_size, file_format from
sys_unload_detail;
```

輸出範例。

```
query_id |                substring                | transfer_size
| file_format
-----+-----
+-----+-----
    9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0000_part_00.gz |
395886 | Text
    9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0001_part_00.gz |
406444 | Text
    9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0002_part_00.gz |
409431 | Text
    9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0003_part_00.gz |
403051 | Text
    9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0004_part_00.gz |
413592 | Text
    9272 | s3://amzn-s3-demo-bucket/my_unload_doc_venue0005_part_00.gz |
395689 | Text
(6 rows)
```

SYS_UNLOAD_HISTORY

使用 SYS_UNLOAD_HISTORY 檢視UNLOAD命令的詳細資訊。每一列代表一個UNLOAD命令，其中包含某些欄位的累積統計資料。它同時包含執行中和已完成的UNLOAD命令。

SYS所有使用者都可看見 _UNLOAD_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交卸載之使用者的識別碼。
query_id	bigint	UNLOAD 命令的查詢識別碼。
transaction_id	bigint	交易識別碼。
session_id	integer	執行卸載之程序的處理程序識別碼。
database_name	text	發出操作時，要將使用者連接至其中的資料庫名稱。
status	text	UNLOAD 命令的狀態。有效值包括：running、completed、aborted 與 unknown。
start_time	timestamp	開始卸載的時間。
end_time	timestamp	卸載完成的時間。
duration	bigint	在UNLOAD命令中花費的時間（微秒）。
file_format	text	輸出檔案的檔案格式。
compression_type	text	壓縮類型。
unloaded_location	text	已卸載檔案的 Amazon S3 位置。
unloaded_rows	bigint	列的數。
unloaded_files_count	bigint	輸出檔案的檔案計數。
unloaded_files_size	bigint	輸出檔案的檔案大小。
error_message	text	UNLOAD 命令的錯誤訊息。

範例查詢

下列查詢顯示卸載查詢詳細資訊，包括卸載命令的格式、列和檔案計數。

```
SELECT query_id,
       file_format,
       start_time,
       duration,
       unloaded_rows,
       unloaded_files_count
FROM sys_unload_history
ORDER BY query_id,
         file_format limit 100;
```

輸出範例。

```
query_id | file_format |          start_time          | duration | unloaded_rows |
unloaded_files_count
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
    527067 | Text      | 2022-02-09 05:18:35.844452 | 5932478 |          10 |
          1
```

SYS_USERLOG

記錄資料庫使用者之下列變更的詳細資訊：

- 建立使用者
- 捨棄使用者
- 更改使用者 (重新命名)
- 更改使用者 (更改屬性)

您可以查詢此檢視，以查看有關無伺服器工作群組和已佈建叢集的資訊。

SYS只有超級使用者可以看到 `_USERLOG`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	提交卸載之使用者的識別碼。
user_name	character(50)	受到變更影響之使用者的使用者名稱。
original_user_name	character(50)	重新命名動作中的原始使用者名稱。若是任何其他動作，此欄位空白。
動作	character(10)	發生的動作。有效值包括 alter、create、drop 和 rename。
has_create_db_privs	integer	如果為 true (值為 1)，則使用者具有建立資料庫許可。
is_superuser	integer	若為 true (值為 1)，使用者可以更新系統目錄。
has_update_catalog_privs	integer	若為 true (值為 1)，使用者可以更新系統目錄。
password_expiration	timestamp	密碼到期日。
session_id	integer	處理程序 ID。
transaction_id	bigint	交易 ID。
record_time	timestamp	查詢啟動時的 UTC 中時間。

範例查詢

下列範例會執行四個使用者動作，然後查詢 SYS_USERLOG 檢視。

```
CREATE USER userlog1 password 'Userlog1';
ALTER USER userlog1 createdb createuser;
```



```
ALTER USER userlog1 rename to userlog2;
DROP user userlog2;
```

```
SELECT user_id, user_name, original_user_name, action, has_create_db_privs,
       is_superuser from SYS_USERLOG order by record_time desc;
```

```
user_id | user_name | original_user_name | action | has_create_db_privs |
is_superuser
-----+-----+-----+-----+-----+-----
    108 | userlog2 |                   | drop   |                    1 | 1
    108 | userlog2 |       userlog1    | rename |                    1 | 1
    108 | userlog1 |                   | alter  |                    1 | 1
    108 | userlog1 |                   | create |                    0 | 0
(4 rows)
```

SYS_VACUUM_HISTORY

使用 SYS_VACUUM_HISTORY 檢視真空查詢的詳細資訊。如需 VACUUM 命令的資訊，請參閱 [VACUUM](#)。

SYS 所有使用者都可看見 _VACUUM_HISTORY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
user_id	integer	啟動查詢的使用者 ID。
transaction_id	long	VACUUM 陳述式的交易 ID。
query_id	long	VACUUM 陳述式的查詢識別碼。您可以將此資料表加入 SYS_QUERY_DETAIL 檢視，以查看為指定 VACUUM 交易執行的個別 SQL 陳述式。如果清空整個資料庫，則會清空個別交易中的每個資料表。對於自

欄名稱	資料類型	描述
		動化VACUUM操作，此值為 null。
database_name	text	資料庫的名稱。
schema_name	text	結構描述的名稱。
table_name	text	資料表的名稱。
table_id	integer	資料表的 ID。
vacuum_type	character	VACUUM 操作的類型。可能的值如下： <ul style="list-style-type: none">• Delete• Sort• Reindex• Recluster• Full 如需清空類型的相關資訊，請參閱 VACUUM 。
is_automatic	boolean	如果操作是自動清空則為 true。否則為 false。

欄名稱	資料類型	描述
status	character	描述在清空操作過程中所做的目前活動： <ul style="list-style-type: none"> • 初始化 • Sort • Merge • Delete • Select • 失敗 • 完成 • 略過 • 建置INTERLEAVEDSORTKEY順序
start_time	timestamp	清空操作開始的時間。
end_time	timestamp	清空操作結束的時間。如果操作正在進行中，則此欄位為空白。
record_time	timestamp	在 SYS_VACUUM_ 中記錄真空操作的時間HISTORY。
持續時間	integer	清空操作開始和結束之間的微秒數。如果清空操作正在進行中，則此欄位為空白。
rows_before_vacuum	bigint	資料表中的實際資料列數目加上任何仍在磁碟上儲存的已刪除資料列 (等待清空)。
size_before_vacuum	integer	清空操作開始之前的資料表大小，以 MB 為單位。

欄名稱	資料類型	描述
reclaimable_rows	bigint	清空操作在開始之前估計將回收的列數。
reclaimed_rows	bigint	清空操作回收的列數。
reclaimed_blocks	bigint	清空操作回收的區塊數。
sortedrows_before_vacuum	integer	清空操作開始之前資料表中已排序的列數。
sortedrows_after_vacuum	integer	清空操作完成後，資料表中額外排序的列數。這不包括計數的行sortedrows_before_vacuum。

用於移轉至 SYS 監視檢視的系統檢視對映

將 Amazon Redshift 佈建叢集遷移到 Amazon Redshift Serverless 時，您的監控或診斷查詢可能會參考僅在已佈建叢集上可用的系統檢視。您可以更新查詢以使用 SYS 監控檢視。此頁面提供僅啟動設定至 SYS 視觀表對應，供您在更新查詢時參照。

主題

- [SYS_QUERY_HISTORY](#)
- [SYS_QUERY_DETAIL](#)
- [SYS_RESTORE_LOG](#)
- [SYS_RESTORE_STATE](#)
- [SYS_TRANSACTION_HISTORY](#)
- [SYS_QUERY_TEXT](#)
- [SYS_CONNECTION_LOG](#)
- [SYS_SESSION_HISTORY](#)
- [SYS_LOAD_DETAIL](#)
- [SYS_LOAD_HISTORY](#)
- [SYS_LOAD_ERROR_DETAIL](#)

- [SYS_UNLOAD_HISTORY](#)
- [SYS_UNLOAD_DETAIL](#)
- [SYS_COPY_REPLACEMENTS](#)
- [SYS_DATASHARE_USAGE_CONSUMER](#)
- [SYS_DATASHARE_USAGE_PRODUCER](#)
- [SYS_DATASHARE_CROSS_REGION_USAGE](#)
- [SYS_DATASHARE_CHANGE_LOG](#)
- [SYS_EXTERNAL_QUERY_DETAIL](#)
- [SYS_EXTERNAL_QUERY_ERROR](#)
- [SYS_VACUUM_HISTORY](#)
- [SYS_ANALYZE_HISTORY](#)
- [SYS_ANALYZE_COMPRESSION_HISTORY](#)
- [SYS_MV_REFRESH_HISTORY](#)
- [SYS_MV_STATE](#)
- [SYS_PROCEDURE_CALL](#)
- [SYS_PROCEDURE_MESSAGES](#)
- [SYS_UDF_LOG](#)
- [SYS_USERLOG](#)
- [SYS_SCHEMA_QUOTA_VIOLATIONS](#)
- [SYS_SPATIAL_SIMPLIFY](#)

SYS_QUERY_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_QUERY_HISTORY](#) 中定義。

- [STL_DDLTEXT](#)
- [STL_ERROR](#)
- [STL_QUERY](#)
- [STL_UTILITYTEXT](#)
- [STL_WLM_QUERY](#)
- [STV_INFLIGHT](#)

- [STV_RECENTS](#)
- [STV_WLM_QUERY_STATE](#)
- [SVL_COMPILE](#)
- [SVL_MULTI_STATEMENT_VIOLATIONS](#)
- [SVL_QLOG](#)
- [SVL_QUERY_QUEUE_INFO](#)
- [SVL_STATEMENTTEXT](#)
- [SVL_TERMINATE](#)

SYS_QUERY_DETAIL

下列資料表中的部分或所有欄也會在 [SYS_QUERY_DETAIL](#) 中定義。

- [STL_AGGR](#)
- [STL_ALERT_EVENT_LOG](#)
- [STL_BCAST](#)
- [STL_DELETE](#)
- [STL_DIST](#)
- [STL_EXPLAIN](#)
- [STL_HASH](#)
- [STL_HASHJOIN](#)
- [STL_INSERT](#)
- [STL_LIMIT](#)
- [STL_MERGE](#)
- [STL_MERGEJOIN](#)
- [STL_NESTLOOP](#)
- [STL_PARSE](#)
- [STL_PLAN_INFO](#)
- [STL_PROJECT](#)
- [STL_QUERY_METRICS](#)
- [STL_RETURN](#)

- [STL_SAVE](#)
- [STL_SCAN](#)
- [STL_SORT](#)
- [STL_STREAM_SEGS](#)
- [STL_UNIQUE](#)
- [STL_WINDOW](#)
- [STV_EXEC_STATE](#)
- [STV_QUERY_METRICS](#)
- [SVCS_QUERY_SUMMARY](#)
- [SVL_QUERY_METRICS](#)
- [SVL_QUERY_METRICS_SUMMARY](#)
- [SVL_QUERY_REPORT](#)
- [SVL_QUERY_SUMMARY](#)
- [SVV_QUERY_STATE](#)

SYS_RESTORE_LOG

下列資料表中的部分或所有欄也會在 [SYS_RESTORE_LOG](#) 中定義。

- [SVL 還原 _ 更改表 _ 進度](#)

SYS_RESTORE_STATE

下列資料表中的部分或所有欄也會在 [SYS_RESTORE_STATE](#) 中定義。

- [STV_XRESTORE ALTER QUEUE STATE](#)

SYS_TRANSACTION_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_TRANSACTION_HISTORY](#) 中定義。

- [STL_COMMIT_STATS](#)
- [STL_TR_CONFLICT](#)

- [STL_UNDONE](#)

SYS_QUERY_TEXT

下列資料表中的部分或所有欄也會在 [SYS_QUERY_TEXT](#) 中定義。

- [STL_QUERYTEXT](#)

SYS_CONNECTION_LOG

下列資料表中的部分或所有欄也會在 [SYS_CONNECTION_LOG](#) 中定義。

- [STL_CONNECTION_LOG](#)

SYS_SESSION_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_SESSION_HISTORY](#) 中定義。

- [STL_SESSIONS](#)
- [STL_RESTARTED_SESSIONS](#)
- [STV_SESSIONS](#)

SYS_LOAD_DETAIL

下列資料表中的部分或所有欄也會在 [SYS_LOAD_DETAIL](#) 中定義。

- [STL_LOAD_COMMITS](#)

SYS_LOAD_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_LOAD_HISTORY](#) 中定義。

- [STL_LOAD_COMMITS](#)

SYS_LOAD_ERROR_DETAIL

下列資料表中的部分或所有欄也會在 [SYS_LOAD_ERROR_DETAIL](#) 中定義。

- [STL_LOADERROR_DETAIL](#)
- [STL_LOAD_ERRORS](#)

SYS_UNLOAD_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_UNLOAD_HISTORY](#) 中定義。

- [STL_UNLOAD_LOG](#)

SYS_UNLOAD_DETAIL

下列資料表中的部分或所有欄也會在 [SYS_UNLOAD_DETAIL](#) 中定義。

- [STL_UNLOAD_LOG](#)

SYS_COPY_REPLACEMENTS

下列資料表中的部分或所有欄也會在 [SYS_COPY_REPLACEMENTS](#) 中定義。

- [STL_REPLACEMENTS](#)

SYS_DATASHARE_USAGE_CONSUMER

下列資料表中的部分或所有欄也會在 [SYS_DATASHARE_USAGE_CONSUMER](#) 中定義。

- [SVL_DATASHARE_USAGE_CONSUMER](#)

SYS_DATASHARE_USAGE_PRODUCER

下列資料表中的部分或所有欄也會在 [SYS_DATASHARE_USAGE_PRODUCER](#) 中定義。

- [SVL_DATASHARE_USAGE_PRODUCER](#)

SYS_DATASHARE_CROSS_REGION_USAGE

下列資料表中的部分或所有欄也會在 [SYS_DATASHARE_CROSS_REGION_USAGE](#) 中定義。

- [SVL_DATASHARE_CROSS_REGION_USAGE](#)

SYS_DATASHARE_CHANGE_LOG

下列資料表中的部分或所有欄也會在 [SYS_DATASHARE_CHANGE_LOG](#) 中定義。

- [SVL_DATASHARE_CHANGE_LOG](#)

SYS_EXTERNAL_QUERY_DETAIL

下列資料表中的部分或所有欄也會在 [SYS_EXTERNAL_QUERY_DETAIL](#) 中定義。

- [SVL_FEDERATED_QUERY](#)
- [SVL_S3LIST](#)
- [SVL_S3QUERY](#)
- [SVL_S3QUERY_SUMMARY](#)

SYS_EXTERNAL_QUERY_ERROR

下列資料表中的部分或所有欄也會在 [SYS_EXTERNAL_QUERY_ERROR](#) 中定義。

- [SVL_SPECTRUM_SCAN_ERROR](#)

SYS_VACUUM_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_VACUUM_HISTORY](#) 中定義。

- [STL_VACUUM](#)
- [SVL_VACUUM_PERCENTAGE](#)
- [SVV_VACUUM_PROGRESS](#)
- [SVV_VACUUM_SUMMARY](#)

SYS_ANALYZE_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_ANALYZE_HISTORY](#) 中定義。

- [STL_ANALYZE](#)

SYS_ANALYZE_COMPRESSION_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_ANALYZE_COMPRESSION_HISTORY](#) 中定義。

- [STL_ANALYZE_COMPRESSION](#)

SYS_MV_REFRESH_HISTORY

下列資料表中的部分或所有欄也會在 [SYS_MV_REFRESH_HISTORY](#) 中定義。

- [SVL_MV_REFRESH_STATUS](#)

SYS_MV_STATE

下列資料表中的部分或所有欄也會在 [SYS_MV_STATE](#) 中定義。

- [STL_MV_STATE](#)

SYS_PROCEDURE_CALL

下列資料表中的部分或所有欄也會在 [SYS_PROCEDURE_CALL](#) 中定義。

- [SVL_STORED_PROC_CALL](#)

SYS_PROCEDURE_MESSAGES

下列資料表中的部分或所有欄也會在 [SYS_PROCEDURE_MESSAGES](#) 中定義。

- [SVL_STORED_PROC_MESSAGES](#)

SYS_UDF_LOG

下列資料表中的部分或所有欄也會在 [SYS_UDF_LOG](#) 中定義。

- [SVL_UDF_LOG](#)

SYS_USERLOG

下列資料表中的部分或所有欄也會在 [SYS_USERLOG](#) 中定義。

- [STL_USERLOG](#)

SYS_SCHEMA_QUOTA_VIOLATIONS

下列資料表中的部分或所有欄也會在 [SYS_SCHEMA_QUOTA_VIOLATIONS](#) 中定義。

- [STL_SCHEMA_QUOTA_VIOLATIONS](#)

SYS_SPATIAL_SIMPLIFY

下列資料表中的部分或所有欄也會在 [SYS_SPATIAL_SIMPLIFY](#) 中定義。

- [SVL_SPATIAL_SIMPLIFY](#)

系統監控 (僅限已佈建)

您可以在已佈建的叢集上查詢下列系統資料表和檢視。STL 和 STV 資料表和檢視包含在數個系統資料表中找到的資料子集。這些項目可讓您快速輕鬆存取在那些資料表中找到的常見查詢資料。

SVCS 檢視可提供主要叢集與並行擴展叢集查詢的詳細資訊。SVL 檢視僅針對在主叢集上執行的查詢提供資訊，但 SVL_STATEMENTTEXT 除外。SVL_STATEMENTTEXT 可以包含在並行擴展叢集以及主叢集上執行之查詢的資訊。

主題

- [STL 記錄的檢視](#)
- [STV用於快照資料的表格](#)
- [SVCS主要和並行擴展叢集的檢視](#)
- [主要叢集的 SVL 檢視](#)

STL 記錄的檢視

STL 系統檢視會從 Amazon Redshift 日誌檔案產生，以提供系統歷史記錄。

這些檔案位於資料倉儲叢集中的每一個節點上。STL 檢視會從日誌中取得資訊，並將其格式化為系統管理員可用的檢視。

日誌保留 – STL系統檢視會保留七天的日誌歷史記錄。所有叢集大小和節點類型都保證日誌保留，並且不受叢集工作負載變化的影響。日誌保留也不受叢集狀態的影響，例如叢集暫停時。只有在叢集是新的情況下，您才會有少於 7 天的日誌歷史記錄。您不需要採取任何動作來保留日誌，但您必須定期將日誌資料複製到其他資料表，或將其卸載至 Amazon S3，以保留超過 7 天的日誌資料。

主題

- [STL_AGGR](#)
- [STL_ALERT_EVENT_LOG](#)
- [STL_ANALYZE](#)
- [STL_ANALYZE_COMPRESSION](#)
- [STL_BCAST](#)
- [STL_COMMIT_STATS](#)
- [STL_CONNECTION_LOG](#)
- [STL_DDLTEXT](#)
- [STL_DELETE](#)
- [STL_DISK_FULL_DIAG](#)
- [STL_DIST](#)
- [STL_ERROR](#)
- [STL_EXPLAIN](#)
- [STL_FILE_SCAN](#)
- [STL_HASH](#)
- [STL_HASHJOIN](#)
- [STL_INSERT](#)
- [STL_LIMIT](#)
- [STL_LOAD_COMMITS](#)
- [STL_LOAD_ERRORS](#)
- [STL_LOADERROR_DETAIL](#)
- [STL_MERGE](#)
- [STL_MERGEJOIN](#)

- [STL_MV_STATE](#)
- [STL_NESTLOOP](#)
- [STL_PARSE](#)
- [STL_PLAN_INFO](#)
- [STL_PROJECT](#)
- [STL_QUERY](#)
- [STL_QUERY_METRICS](#)
- [STL_QUERYTEXT](#)
- [STL_REPLACEMENTS](#)
- [STL_RESTARTED_SESSIONS](#)
- [STL_RETURN](#)
- [STL_S3CLIENT](#)
- [STL_S3CLIENT_ERROR](#)
- [STL_SAVE](#)
- [STL_SCAN](#)
- [STL_SCHEMA_QUOTA_VIOLATIONS](#)
- [STL_SESSIONS](#)
- [STL_SORT](#)
- [STL_SSHCLIENT_ERROR](#)
- [STL_STREAM_SEGS](#)
- [STL_TR_CONFLICT](#)
- [STL_UNDONE](#)
- [STL_UNIQUE](#)
- [STL_UNLOAD_LOG](#)
- [STL_USAGE_CONTROL](#)
- [STL_USERLOG](#)
- [STL_UTILITYTEXT](#)
- [STL_VACUUM](#)
- [STL_WINDOW](#)
- [STL_WLM_ERROR](#)

- [STL_WLM_RULE_ACTION](#)
- [STL_WLM_QUERY](#)

STL_AGGR

分析查詢的彙總執行步驟。這些步驟會在彙總函數和 GROUP BY 子句的執行期間發生。

STL 所有使用者都可看見 _AGGR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_AGGR 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用 SYS 監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

欄名稱	資料類型	描述
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
slots	integer	雜湊儲存貯體的數目。
occupied	integer	包含記錄之插槽的數目。
maxlength	integer	最大插槽的大小。
tbl	integer	表格 ID。
is_diskbased	character(1)	若為 true (t) , 查詢是以磁碟型操作方式執行。若為 false (f) , 查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體的位元組數。
type	character(6)	步驟的類型。有效的 值如下 : <ul style="list-style-type: none"> • HASHED。表示步驟已使用分組且未排序的彙整。 • PLAIN。表示步驟已使用未分組的純量彙整。 • SORTED。表示步驟已使用分組且排序的彙整。
resizes	integer	此資訊僅供內部使用。
flushable	integer	此資訊僅供內部使用。

範例查詢

傳回 1 SLICE 和 239 TBL 的彙總執行步驟的相關資訊。

```
select query, segment, bytes, slots, occupied, maxlength, is_diskbased, workmem, type
from stl_aggr where slice=1 and tbl=239
order by rows
limit 10;
```



```

query | segment | bytes | slots | occupied | maxlength | is_diskbased | workmem |
type
-----+-----+-----+-----+-----+-----+-----+-----
+-----
  562 |      1 |      0 | 4194304 |      0 |      0 | f |      | 383385600 |
HASHED
  616 |      1 |      0 | 4194304 |      0 |      0 | f |      | 383385600 |
HASHED
  546 |      1 |      0 | 4194304 |      0 |      0 | f |      | 383385600 |
HASHED
  547 |      0 |      8 |      0 |      0 |      0 | f |      |      0 |
PLAIN
  685 |      1 |     32 | 4194304 |      1 |      0 | f |      | 383385600 |
HASHED
  652 |      0 |      8 |      0 |      0 |      0 | f |      |      0 |
PLAIN
  680 |      0 |      8 |      0 |      0 |      0 | f |      |      0 |
PLAIN
  658 |      0 |      8 |      0 |      0 |      0 | f |      |      0 |
PLAIN
  686 |      0 |      8 |      0 |      0 |      0 | f |      |      0 |
PLAIN
  695 |      1 |     32 | 4194304 |      1 |      0 | f |      | 383385600 |
HASHED
(10 rows)

```

STL_ALERT_EVENT_LOG

當查詢最佳化器識別可能表示效能問題的狀況時，請記錄一個提醒。使用 STL_ALERT_EVENT_LOG 檢視來識別改善查詢效能的機會。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱[查詢處理](#)。

STL所有使用者都可看見 _ALERT_EVENT_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_ALERT_EVENT_LOG 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
pid	integer	與陳述式和配量相關聯的處理程序 ID。PIDs 如果相同的查詢在多個切片上執行，則可能有多個查詢。
xid	bigint	與陳述式關聯的交易 ID。
事件	character (1024)	提醒事件的說明。
solution	character (1024)	建議的解決方案。
event_time	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

使用須知

您可以使用 STL_ALERT_EVENT_LOG 來識別查詢中的潛在問題，然後遵循 [中的實務查詢效能調校](#) 來最佳化資料庫設計並重新寫入查詢。STL_ALERT_EVENT_LOG 會記錄下列提醒：

- 找不到統計資訊

找不到統計資訊。執行ANALYZE下列資料載入或重大更新，並STATUPDATE搭配 COPY 操作使用。如需詳細資訊，請參閱[設計查詢的 Amazon Redshift 最佳實務](#)。

- 巢狀迴圈

巢狀迴路通常是 Cartesian 產品。評估您的查詢，以確保所有參與資料表均已有效聯結。

- 選擇性相當高的篩選條件

傳回的資料列與已掃描資料列的比率低於 0.05。已掃描列是 `rows_pre_user_filter` 的值，而傳回的列則是 [STL_SCAN](#) 系統檢視中的列值。表示查詢正在掃描例外狀況大量的資料列來決定結果集。這可能是由於找不到排序索引鍵或其不正確所致。如需詳細資訊，請參閱[排序金鑰](#)。

- 過多的幽靈資料列

掃描已略過相當多標示為已刪除但未清空的資料列，或已插入但未遞交的資料列。如需詳細資訊，請參閱[清空資料表](#)。

- 大型分佈

已重新配送超過 1,000,000 個資料列，進行雜湊聯結或彙整。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

- 大型廣播

已播送超過 1,000,000 個資料列，進行雜湊聯結。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

- 序列執行

`DS_DIST_ALL_INNER` 重新分發樣式在查詢計劃中表示，這會強制序列執行，因為整個內部資料表已重新分發至單一節點。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

範例查詢

下列查詢顯示四個查詢的提醒事件。

```
SELECT query, substring(event,0,25) as event,
substring(solution,0,25) as solution,
trim(event_time) as event_time from stl_alert_event_log order by query;
```

query	event	solution	event_time
6567	Missing query planner statist	Run the ANALYZE command	2014-01-03 18:20:58
7450	Scanned a large number of del	Run the VACUUM command to rec	2014-01-03 21:19:31
8406	Nested Loop Join in the query	Review the join predicates to	2014-01-04 00:34:22

```
29512 | Very selective query filter:r | Review the choice of sort key| 2014-01-06
22:00:00
```

(4 rows)

STL_ANALYZE

記錄 [ANALYZE](#) 操作的詳細資訊。

STL只有超級使用者可以看到 `_ANALYZE`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_ANALYZE_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	long	交易 ID。
database	char(30)	資料庫名稱。
table_id	integer	資料表 ID。
status	char(15)	分析命令的結果。可能值為 Full、Skipped 及 Predicate Column 。
rows	double	資料表中的列總數。
modified_rows	double	自上次ANALYZE操作以來修改的資料列總數。
threshold_percent	integer	analyze_threshold_percent 參數的值。
is_auto	char(1)	如果操作預設包含 Amazon Redshift 分析操作，則值為 true (t)。如果命令明確執行，則值ANALYZE為 false (f)。

欄名稱	資料類型	描述
starttime	timestamp	UTC 分析操作開始執行的時間。
endtime	timestamp	UTC 分析操作完成執行的時間。
prevtime	timestamp	先前分析資料表UTC的時間。
num_predicate_cols	integer	資料表中述詞資料欄的目前數目。
num_new_predicate_cols	integer	自前一個分析操作後資料表中新述詞資料欄的數目。
is_background	character(1)	如果分析是由自動分析操作執行，則值為 true (t)。否則值為 false (f)。
auto_analyze_phase	character(100)	保留供內部使用。
schema_name	char(128)	資料表的結構描述名稱。
table_name	char(136)	資料表的名稱。

範例查詢

下列範例會加入 STV_TBL_PERM 以顯示資料表名稱和執行詳細資訊。

```
select distinct a.xid, trim(t.name) as name, a.status, a.rows, a.modified_rows,
  a.starttime, a.endtime
from stl_analyze a
join stv_tbl_perm t on t.id=a.table_id
where name = 'users'
order by starttime;
```

```
xid      | name  | status          | rows  | modified_rows | starttime          |
endtime
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
```

```

1582 | users | Full          | 49990 |          | 49990 | 2016-09-22 22:02:23 |
2016-09-22 22:02:28
244287 | users | Full          | 24992 |          | 74988 | 2016-10-04 22:50:58 |
2016-10-04 22:51:01
244712 | users | Full          | 49984 |          | 24992 | 2016-10-04 22:56:07 |
2016-10-04 22:56:07
245071 | users | Skipped       | 49984 |          | 0      | 2016-10-04 22:58:17 |
2016-10-04 22:58:17
245439 | users | Skipped       | 49984 |          | 1982  | 2016-10-04 23:00:13 |
2016-10-04 23:00:13
(5 rows)

```

STL_ANALYZE_COMPRESSION

在 COPY 或 ANALYZECOMPRESSION 命令期間記錄壓縮分析操作的詳細資訊。

STL 所有使用者都可看見 _ANALYZE_COMPRESSION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在 SYS 監控檢視 中找到 [SYS_ANALYZE_COMPRESSION_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用 SYS 監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
start_time	timestamp	壓縮分析操作開始進行的時間。
xid	bigint	壓縮分析操作的交易 ID。
tbl	integer	接受分析之資料表的 ID。
資料表名稱	character(128)	接受分析之資料表的名稱。
col	integer	接受分析之資料表的資料欄索引，用於判斷壓縮編碼。
old_encoding	character(15)	壓縮分析之後的編碼類型。

欄名稱	資料類型	描述
new_encoding	character(15)	壓縮分析之後的編碼類型。
模式	character(14)	<p>可能值如下：</p> <p>PRESET</p> <p>指定 new_encoding 是由 Amazon Redshift COPY命令根據資料欄資料類型決定。不會對任何資料取樣。</p> <p>ON</p> <p>指定 new_encoding 是由 Amazon Redshift COPY命令根據範例資料分析來決定。</p> <p>ANALYZE ONLY</p> <p>指定 new_encoding 是由 Amazon Redshift ANALYZECOMPRESSION命令根據範例資料分析來決定。然而，不會變更已分析資料欄的編碼類型。</p>
best_compression_encoding	character(15)	提供最佳壓縮比率的編碼類型。
recommended_bytes	character(15)	採用新編碼所使用的位元組。
best_compression_bytes	character(15)	採用最佳壓縮編碼所使用的位元組。
ndv	bigint	取樣資料列中不同值的數量。

範例查詢

下列範例會檢查lineitem資料表上壓縮分析的詳細資訊，方法是在相同工作階段中執行的最後一個COPY命令。

```
select xid, tbl, btrim(tablename) as tablename, col, old_encoding, new_encoding,
       best_compression_encoding, mode
from stl_analyze_compression
where xid = (select xid from stl_query where query = pg_last_copy_id()) order by col;
```

xid	tbl	tablename	col	old_encoding	new_encoding	mode
5308	158961	\$lineitem	0	mostly32	az64	delta
	ON					
5308	158961	\$lineitem	1	mostly32	az64	az64
	ON					
5308	158961	\$lineitem	2	lzo	az64	az64
	ON					
5308	158961	\$lineitem	3	delta	az64	az64
	ON					
5308	158961	\$lineitem	4	bytedict	az64	bytedict
	ON					
5308	158961	\$lineitem	5	mostly32	az64	az64
	ON					
5308	158961	\$lineitem	6	delta	az64	az64
	ON					
5308	158961	\$lineitem	7	delta	az64	az64
	ON					
5308	158961	\$lineitem	8	lzo	lzo	lzo
	ON					
5308	158961	\$lineitem	9	runlength	runlength	runlength
	ON					
5308	158961	\$lineitem	10	delta	az64	az64
	ON					
5308	158961	\$lineitem	11	delta	az64	az64
	ON					
5308	158961	\$lineitem	12	delta	az64	az64
	ON					
5308	158961	\$lineitem	13	bytedict	bytedict	bytedict
	ON					
5308	158961	\$lineitem	14	bytedict	bytedict	bytedict
	ON					
5308	158961	\$lineitem	15	text255	text255	text255
	ON					

(16 rows)

STL_BCAST

記錄在執行播送資料的查詢步驟期間網路活動的相關資訊。網路流量是藉由資料列數、位元組數和封包數所擷取的，而這些資料列、位元組和封包是在特定配量上的特定步驟期間透過網路傳送的。步驟的持續時間是記錄的開始時間與結束時間之間的差異。

若要識別查詢中的廣播步驟，請在 SVL_QUERY_SUMMARY 檢視中尋找廣播標籤，或執行 EXPLAIN 命令，然後尋找包含廣播的步驟屬性。

STL 所有使用者都可看見 _BCAST。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

STL_BCAST 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用 SYS 監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

欄名稱	資料類型	描述
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
packets	integer	透過網路傳送的封包總數。

範例查詢

下列範例傳回查詢的播送資訊，其中具有一個或多個封包，而且查詢的開始與結束之間的差異為一秒或以上。

```
select query, slice, step, rows, bytes, packets, datediff(seconds, starttime, endtime)
from stl_bcast
where packets>0 and datediff(seconds, starttime, endtime)>0;
```

```
query | slice | step | rows | bytes | packets | date_diff
-----+-----+-----+-----+-----+-----+-----
  453 |     2 |     5 |     1 |   264 |         1 |         1
  798 |     2 |     5 |     1 |   264 |         1 |         1
 1408 |     2 |     5 |     1 |   264 |         1 |         1
 2993 |     0 |     5 |     1 |   264 |         1 |         1
 5045 |     3 |     5 |     1 |   264 |         1 |         1
 8073 |     3 |     5 |     1 |   264 |         1 |         1
 8163 |     3 |     5 |     1 |   264 |         1 |         1
 9212 |     1 |     5 |     1 |   264 |         1 |         1
 9873 |     1 |     5 |     1 |   264 |         1 |         1
(9 rows)
```

STL_COMMIT_STATS

提供與遞交效能相關的指標，包括各種遞交階段的計時，以及遞交的區塊數。查詢 STL_COMMIT_STATS 以確定交易的哪些部分用於遞交，以及正在發生多少佇列。

STL只有超級使用者可以看到 _COMMIT_STATS。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_TRANSACTION_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
xid	bigint	交易 ID 遞交中。
node	integer	節點號碼。-1 是領導者節點。
startqueue	timestamp	開始遞交佇列。
startwork	timestamp	開始遞交。
endflush	timestamp	結束髒區塊排清階段。
endstage	timestamp	結束中繼資料臨時階段。
endlocal	timestamp	結束本機遞交階段。
startglobal	timestamp	開始全域階段。
endtime	timestamp	結束遞交。
queuelen	bigint	遞交佇列中在此交易之前的交易數。
permblocks	bigint	進行此遞交時現有永久區塊的數目。
newblocks	bigint	進行此遞交時新的永久區塊的數目。
dirtyblocks	bigint	必須寫入為此遞交一部分的區塊數目。
headers	bigint	必須寫入為此遞交一部分的區塊標頭數目。
numxids	integer	作用中DML交易的數量。
oldestxid	bigint	最舊作用中DML交易XID的。
extwritel atency	bigint	此資訊僅供內部使用。

欄名稱	資料類型	描述
metadataawritten	int	此資訊僅供內部使用。
tombstone dblocks	bigint	此資訊僅供內部使用。
tossedblo cks	bigint	此資訊僅供內部使用。
batched_by	bigint	此資訊僅供內部使用。

範例查詢

```
select node, datediff(ms,startqueue,startwork) as queue_time,
datediff(ms, startwork, endtime) as commit_time, queuelen
from stl_commit_stats
where xid = 2574
order by node;
```

```
node | queue_time | commit_time | queuelen
-----+-----+-----+-----
-1 | 0 | 617 | 0
0 | 444950725641 | 616 | 0
1 | 444950725636 | 616 | 0
```

STL_CONNECTION_LOG

記錄身分驗證嘗試以及連線和中斷連線。

STL只有超級使用者可以看到 `_CONNECTION_LOG`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_CONNECTION_LOG](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
事件	character(50)	連線或身分驗證事件。
recordtime	timestamp	發生事件的時間。
remotehost	character(45)	遠端主機的名稱或 IP 地址。
remoteport	character(32)	遠端主機的連接埠號碼。
pid	integer	與陳述式相關聯的處理程序 ID。
dbname	character(50)	資料庫名稱。
使用者名稱	character(50)	使用者名稱。
authmethod	character(32)	身分驗證方法。
持續時間	integer	連線的持續時間，以微秒為單位。
sslversion	character(50)	安全通訊端層 (SSL) 版本。
sslcipher	character(128)	SSL 密碼。
mtu	integer	最大傳輸單位 (MTU) 。
sslcompression	character(64)	SSL 壓縮類型。
sslexpansion	character(64)	SSL 擴充類型。
iamauthguid	character(36)	請求的 IAM CloudTrail 身分驗證 ID。
application_name	character(250)	工作階段之應用程式的初始或已更新名稱。
os_version	character(64)	連線到 Amazon Redshift 叢集之用戶端機器上的作業系統版本。

欄名稱	資料類型	描述
driver_version	character(64)	從第三方SQL用戶端工具連線至 Amazon Redshift 叢集的 ODBC或 JDBC驅動程式版本。
plugin_name	character(32)	連接至您的 Amazon Redshift 叢集時使用的外掛程式名稱。
protocol_version	integer	Amazon Redshift 驅動程式在建立與伺服器的連線時使用的內部通訊協定版本。通訊協定版本會在驅動程式與伺服器之間進行交涉。該版本描述了可用的功能。有效值包含： <ul style="list-style-type: none"> • 0 (BASE_SERVER_PROTOCOL_VERSION) • 1 (EXTENDED_RESULT_METADATA_SERVER_PROTOCOL_VERSION) – 若要為每個查詢儲存往返，伺服器會傳送額外的結果集中繼資料資訊。 • 2 (BINARY_PROTOCOL_VERSION) – 根據結果集的資料類型，伺服器會以二進位格式傳送資料。 • 3 (EXTENDED2_RESULT_METADATA_SERVER_PROTOCOL_VERSION) – 伺服器傳送資料欄的案例敏感度 (定序) 資訊。
sessionid	character(36)	目前工作階段的全域唯一識別碼。工作階段 ID 在節點故障重新啟動後仍然存在。
compression	character(16)	用於連線的壓縮演算法。

範例查詢

若要檢視已開啟之連線的詳細資訊，請執行下列查詢。

```
select recordtime, username, dbname, remotehost, remoteport
from stl_connection_log
where event = 'initiating session'
and pid not in
(select pid from stl_connection_log
where event = 'disconnecting session')
```

```
order by 1 desc;
```

recordtime	username	dbname	remotehost	remoteport
2014-11-06 20:30:06	rdsdb	dev	[local]	
2014-11-06 20:29:37	test001	test	10.49.42.138	11111
2014-11-05 20:30:29	rdsdb	dev	10.49.42.138	33333
2014-11-05 20:28:35	rdsdb	dev	[local]	

(4 rows)

下列範例反映失敗的身分驗證嘗試，以及成功的連線和中斷連線。

```
select event, recordtime, remotehost, username
from stl_connection_log order by recordtime;
```

event	recordtime	remotehost	username
authentication failure	2012-10-25 14:41:56.96391	10.49.42.138	john
authenticated	2012-10-25 14:42:10.87613	10.49.42.138	john
initiating session	2012-10-25 14:42:10.87638	10.49.42.138	john
disconnecting session	2012-10-25 14:42:19.95992	10.49.42.138	john

(4 rows)

下列範例顯示ODBC驅動程式的版本、用戶端電腦上的作業系統，以及用來連線至 Amazon Redshift 叢集的外掛程式。在此範例中，使用的外掛程式用於使用登入名稱和密碼進行標準ODBC驅動程式驗證。

```
select driver_version, os_version, plugin_name from stl_connection_log;
```

driver_version	os_version	plugin_name

```
Amazon Redshift ODBC Driver 1.4.15.0001 | Darwin 18.7.0 x86_64 | none
Amazon Redshift ODBC Driver 1.4.15.0001 | Linux 4.15.0-101-generic x86_64 | none
```

下列範例顯示用戶端電腦上的作業系統版本、驅動程式版本和通訊協定版本。

```
select os_version, driver_version, protocol_version from stl_connection_log;
```

```
os_version          | driver_version          | protocol_version
-----+-----+-----
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
Linux 4.15.0-101-generic x86_64 | Redshift JDBC Driver 2.0.0.0 | 2
```

STL_DDLTEXT

擷取在系統上執行的下列DDL陳述式。

這些DDL陳述式包括下列查詢和物件：

- CREATE SCHEMA, TABLE, VIEW
- DROP SCHEMA, TABLE, VIEW
- ALTER SCHEMA, TABLE

另請參閱 [STL_QUERYTEXT](#)、[STL_UTILITYTEXT](#) 和 [SVL_STATEMENTTEXT](#)。這些檢視提供在系統上執行的SQL命令時間表；此歷史記錄對於疑難排解和建立所有系統活動的稽核追蹤非常有用。

使用 STARTTIME和 ENDTIME資料欄，找出在指定期間內記錄的陳述式。長SQL文字區塊會細分為 200 個字元；資料SEQUENCE欄會識別屬於單一陳述式的文字片段。

STL所有使用者都可看見 _DDLTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_QUERY_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。

欄名稱	資料類型	描述
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或使用 QUERY_SET GROUP 命令定義的標籤。如果查詢不是以檔案為基礎，或未設定 QUERY_GROUP 參數，則此欄位為空白。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回包含先前執行 DDL 陳述式的記錄。

```
select xid, starttime, sequence, substring(text,1,40) as text
from stl_ddltext order by xid desc, sequence;
```

以下是顯示四個 CREATE TABLE 陳述式的範例輸出。DDL 陳述式會出現在 text 欄中，該欄為了可讀性而截斷。

```
xid |          starttime          | sequence |          text
-----+-----+-----+-----
```

```

1806 | 2013-10-23 00:11:14.709851 |      0 | CREATE TABLE supplier ( s_supkey int4
N
1806 | 2013-10-23 00:11:14.709851 |      1 | s_comment varchar(101) NOT NULL )
1805 | 2013-10-23 00:11:14.496153 |      0 | CREATE TABLE region ( r_regionkey int4
N
1804 | 2013-10-23 00:11:14.285986 |      0 | CREATE TABLE partsupp ( ps_partkey int8
1803 | 2013-10-23 00:11:14.056901 |      0 | CREATE TABLE part ( p_partkey int8 NOT
N
1803 | 2013-10-23 00:11:14.056901 |      1 | ner char(10) NOT NULL , p_retailprice
nu
(6 rows)

```

重建已儲存 SQL

下列SQL列出儲存在 STL_text 資料欄中的資料列DDLTEXT。列依 xid 與 sequence 排序。如果原始 SQL 超過 200 個字元的多列，STL_DDLTEXT 可以包含的多列sequence。

```

SELECT xid, sequence, LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text)
END, '') WITHIN GROUP (ORDER BY sequence) as query_statement
FROM stl_ddltext GROUP BY xid, sequence ORDER BY xid, sequence;

```

```

xid      | sequence | query_statement
-----+-----+-----
7886671  | 0        | create external schema schema_spectrum_uddh\nfrom data catalog
\ndatabase 'spectrum_db_uddh'\niam_role ''\ncreate external database if not exists;
7886752  | 0        | CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league\n(\n
league_rank smallint,\n prev_rank  smallint,\n club_name  varchar(15),\n
league_name varchar(20),\n league_off  decimal(6,2),\n le
7886752  | 1        | ague_def  decimal(6,2),\n league_spi  decimal(6,2),\n
league_nspi smallint\n)\nROW FORMAT DELIMITED \n  FIELDS TERMINATED BY ',' \n
LINES TERMINATED BY '\\n\\l'\nstored as textfile\nLOCATION 's
7886752  | 2        | 3://mybucket-spectrum-uddh/'\ntable properties
('skip.header.line.count'='1');
...

```

若要重建SQL存放在 STL_text 欄中的 DDLTEXT，請執行下列SQL陳述式。它將來自資料text欄中一或多個區段的DDL陳述式組合在一起。在執行重建的之前SQL，請將任何 (\n) 特殊字元取代為SQL用戶端中的新行。下列SELECT陳述式的結果會依序組合三列SQL，以重建 query_statement 欄位中的。

```
SELECT LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END) WITHIN
GROUP (ORDER BY sequence) as query_statement
FROM stl_ddltext GROUP BY xid, endtime order by xid, endtime;
```

```
query_statement
```

```
-----
create external schema schema_spectrum_uddh\nfrom data catalog\ndatabase
'spectrum_db_uddh'\niam_role ''\ncreate external database if not exists;
CREATE EXTERNAL TABLE schema_spectrum_uddh.soccer_league\n(\n league_rank smallint,\n prev_rank smallint,\n club_name varchar(15),\n league_name varchar(20),\n league_off decimal(6,2),\n league_def decimal(6,2),\n league_spi decimal(6,2),\n league_nspi smallint\n)\nROW FORMAT DELIMITED \n FIELDS TERMINATED BY ',' \n
LINES TERMINATED BY '\\n\\l'\nstored as textfile\nLOCATION 's3://mybucket-spectrum-
uddh/'\ntable properties ('skip.header.line.count'='1');
```

STL_DELETE

分析查詢的刪除執行步驟。

STL所有使用者都可看見 `_DELETE`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_DELETE 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。

欄名稱	資料類型	描述
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。

範例查詢

為了在 STL_ 中建立資料列 DELETE，下列範例會將資料列插入 EVENT 資料表，然後刪除該資料列。

首先，將資料列插入 EVENT 資料表，並驗證是否已插入。

```
insert into event(eventid,venueid,catid,dateid,eventname)
values ((select max(eventid)+1 from event),95,9,1857,'Lollapalooza');
```

```
select * from event
where eventname='Lollapalooza'
order by eventid;
```

```
eventid | venueid | catid | dateid | eventname | starttime
-----+-----+-----+-----+-----+-----
  4274 |    102 |    9 |   1965 | Lollapalooza | 2008-05-01 19:00:00
  4684 |    114 |    9 |   2105 | Lollapalooza | 2008-10-06 14:00:00
  5673 |    128 |    9 |   1973 | Lollapalooza | 2008-05-01 15:00:00
  5740 |    51 |    9 |   1933 | Lollapalooza | 2008-04-17 15:00:00
```

```

5856 |      119 |      9 |   1831 | Lollapalooza | 2008-01-05 14:00:00
6040 |      126 |      9 |   2145 | Lollapalooza | 2008-11-15 15:00:00
7972 |      92  |      9 |   2026 | Lollapalooza | 2008-07-19 19:30:00
8046 |      65  |      9 |   1840 | Lollapalooza | 2008-01-14 15:00:00
8518 |      48  |      9 |   1904 | Lollapalooza | 2008-03-19 15:00:00
8799 |      95  |      9 |   1857 | Lollapalooza |
(10 rows)

```

現在，刪除您新增至EVENT資料表的列，並確認該列已刪除。

```

delete from event
where eventname='Lollapalooza' and eventid=(select max(eventid) from event);

```

```

select * from event
where eventname='Lollapalooza'
order by eventid;

```

```

eventid | venueid | catid | dateid | eventname |      starttime
-----+-----+-----+-----+-----+-----
4274 |      102 |      9 |   1965 | Lollapalooza | 2008-05-01 19:00:00
4684 |      114 |      9 |   2105 | Lollapalooza | 2008-10-06 14:00:00
5673 |      128 |      9 |   1973 | Lollapalooza | 2008-05-01 15:00:00
5740 |      51  |      9 |   1933 | Lollapalooza | 2008-04-17 15:00:00
5856 |      119 |      9 |   1831 | Lollapalooza | 2008-01-05 14:00:00
6040 |      126 |      9 |   2145 | Lollapalooza | 2008-11-15 15:00:00
7972 |      92  |      9 |   2026 | Lollapalooza | 2008-07-19 19:30:00
8046 |      65  |      9 |   1840 | Lollapalooza | 2008-01-14 15:00:00
8518 |      48  |      9 |   1904 | Lollapalooza | 2008-03-19 15:00:00
(9 rows)

```

然後，查詢 `stl_delete` 以查看刪除的執行步驟。在此範例中，查詢已傳回超過 300 個的資料列，因此以下輸出會縮短以便於顯示。

```

select query, slice, segment, step, tasknum, rows, tbl from stl_delete order by query;

```

```

query | slice | segment | step | tasknum | rows | tbl
-----+-----+-----+-----+-----+-----+-----
7 |      0 |      0 |      1 |      0 |      0 | 100000
7 |      1 |      0 |      1 |      0 |      0 | 100000
8 |      0 |      0 |      1 |      2 |      0 | 100001

```

```

 8 | 1 | 0 | 1 | 2 | 0 | 100001
 9 | 0 | 0 | 1 | 4 | 0 | 100002
 9 | 1 | 0 | 1 | 4 | 0 | 100002
10 | 0 | 0 | 1 | 6 | 0 | 100003
10 | 1 | 0 | 1 | 6 | 0 | 100003
11 | 0 | 0 | 1 | 8 | 0 | 100253
11 | 1 | 0 | 1 | 8 | 0 | 100253
12 | 0 | 0 | 1 | 0 | 0 | 100255
12 | 1 | 0 | 1 | 0 | 0 | 100255
13 | 0 | 0 | 1 | 2 | 0 | 100257
13 | 1 | 0 | 1 | 2 | 0 | 100257
14 | 0 | 0 | 1 | 4 | 0 | 100259
14 | 1 | 0 | 1 | 4 | 0 | 100259
...

```

STL_DISK_FULL_DIAG

磁碟已滿時記錄有關錯誤的日誌資訊。

STL 只有超級使用者可以看到 `_DISK_FULL_DIAG`。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
currenttime	bigint	自 2000 年 1 月 1 日起，產生錯誤的日期和時間 (以微秒為單位)。
node_num	bigint	節點的識別碼。
query_id	bigint	造成錯誤之查詢的識別碼。
temp_blocks	bigint	查詢建立的臨時區塊數。

範例查詢

下列範例會在發生磁碟已滿錯誤時，傳回所儲存資料的詳細資訊。

```
select * from stl_disk_full_diag
```

下列範例會將 `currenttime` 轉換為時間戳記。

```
select '2000-01-01'::timestamp + (currenttime/1000000.0)* interval '1 second' as
currenttime,node_num,query_id,temp_blocks from pg_catalog.stl_disk_full_diag;
```

currenttime	node_num	query_id	temp_blocks
2019-05-18 19:19:18.609338	0	569399	70982
2019-05-18 19:37:44.755548	0	569580	70982
2019-05-20 13:37:20.566916	0	597424	70869

STL_DIST

記錄在執行遞送資料的查詢步驟期間網路活動的相關資訊。網路流量是藉由資料列數、位元組數和封包數所擷取的，而這些資料列、位元組和封包是在特定配量上的特定步驟期間透過網路傳送的。步驟的持續時間是記錄的開始時間與結束時間之間的差異。

若要識別查詢中的分佈步驟，請在 `QUERY_SUMMARY` 檢視中尋找 `Dist` 標籤，或執行 `EXPLAIN` 命令，然後尋找包含 `Dist` 的步驟屬性。

STL 所有使用者都可看見 `_DIST`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

STL_DIST 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用 `SYS` 監控檢視 [SYS_QUERY_DETAIL](#)。`SYS` 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
<code>userid</code>	<code>integer</code>	產生項目的使用者之 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
packets	integer	透過網路傳送的封包總數。

範例查詢

下列範例傳回查詢的配送資訊，其中具有一個或多個封包，且持續時間大於零。

```
select query, slice, step, rows, bytes, packets,
datediff(seconds, starttime, endtime) as duration
from stl_dist
where packets>0 and datediff(seconds, starttime, endtime)>0
order by query
limit 10;
```

```
query | slice | step | rows | bytes | packets | duration
-----+-----+-----+-----+-----+-----+-----
```



```

567 | 1 | 4 | 49990 | 6249564 | 707 | 1
630 | 0 | 5 | 8798 | 408404 | 46 | 2
645 | 1 | 4 | 8798 | 408404 | 46 | 1
651 | 1 | 5 | 192497 | 9226320 | 1039 | 6
669 | 1 | 4 | 192497 | 9226320 | 1039 | 4
675 | 1 | 5 | 3766 | 194656 | 22 | 1
696 | 0 | 4 | 3766 | 194656 | 22 | 1
705 | 0 | 4 | 930 | 44400 | 5 | 1
111525 | 0 | 3 | 68 | 17408 | 2 | 1
(9 rows)

```

STL_ERROR

記錄 Amazon Redshift 資料庫引擎所產生的內部處理錯誤。STL_ERROR 不會記錄 SQL 錯誤或訊息。STL_ERROR 中的資訊對於疑難排解某些錯誤很有用。AWS 支援工程師可能會請您提供此資訊，做為疑難排解程序的一部分。

STL 所有使用者都可看見 _ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在 SYS 監控檢視 中找到 [SYS_QUERY_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用 SYS 監控檢視。

如需使用 Copy 命令載入資料時可能產生之錯誤代碼的清單，請參閱[載入錯誤參考](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
process	character(12)	擲出例外狀況的處理程序。
recordtime	timestamp	發生錯誤的時間。
pid	integer	處理程序 ID。 STL_QUERY 資料表包含 IDs 已完成查詢的程序 IDs 和唯一查詢。
errcode	integer	對應於錯誤類別的錯誤代碼。
file	character(90)	發生錯誤之來源檔案的名稱。

欄名稱	資料類型	描述
linenum	integer	來源檔案中發生錯誤的行號。
context	character(100)	錯誤的原因。
error	character(512)	錯誤訊息。

範例查詢

下列範例會從 STL_ 擷取錯誤資訊ERROR。

```
select process, errcode, linenum as line,
trim(error) as err
from stl_error;
```

```

   process | errcode | line | err
-----+-----+-----+-----
+-----+-----+-----+-----
padbmaster |    8001 |  194 | Path prefix: s3://redshift-downloads/testnulls/
venue.txt*
padbmaster |    8001 |  529 | Listing bucket=redshift-downloads prefix=tests/
category-csv-quotes
padbmaster |      2 |  190 | database "template0" is not currently accepting
connections
padbmaster |    32 | 1956 | pq_flush: could not send data to client: Broken pipe
(4 rows)
```

STL_EXPLAIN

顯示已提交執行之查詢的EXPLAIN計劃。

STL所有使用者都可看見 _EXPLAIN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_EXPLAIN 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別碼，其中節點會在查詢執行時映射至一個或多個步驟。
parentid	integer	父節點的計劃節點識別碼。父節點具有一些子節點。例如，合併聯結是已聯結資料表上掃描的父項。
plannode	character(400)	EXPLAIN 輸出的節點文字。參考運算節點上執行的計劃節點在 EXPLAIN輸出XN中以 作為字首。
info	character(400)	計劃節點的限定詞和篩選條件資訊。例如，聯結條件和WHE RE子句限制會包含在此欄位中。

範例查詢

請考慮彙總聯結查詢的下列EXPLAIN輸出：

```
explain select avg(datediff(day, listtime, saletime)) as avgwait
from sales, listing where sales.listid = listing.listid;
          QUERY PLAN
-----
XN Aggregate (cost=6350.30..6350.31 rows=1 width=16)
-> XN Hash Join DS_DIST_NONE (cost=47.08..6340.89 rows=3766 width=16)
    Hash Cond: ("outer".listid = "inner".listid)
-> XN Seq Scan on listing (cost=0.00..1924.97 rows=192497 width=12)
-> XN Hash (cost=37.66..37.66 rows=3766 width=12)
    -> XN Seq Scan on sales (cost=0.00..37.66 rows=3766 width=12)
(6 rows)
```

如果您執行此查詢，且查詢 ID 為 10，您可以使用 STL_EXPLAIN 資料表來查看EXPLAIN命令傳回的相同資訊類型：

```
select query,nodeid,parentid,substring(plannode from 1 for 30),
substring(info from 1 for 20) from stl_explain
where query=10 order by 1,2;
```

query	nodeid	parentid	substring	substring
10	1	0	XN Aggregate (cost=6717.61..6	
10	2	1	-> XN Merge Join DS_DIST_N0	Merge Cond:("outer"
10	3	2	-> XN Seq Scan on lis	
10	4	2	-> XN Seq Scan on sal	

(4 rows)

請考慮下列查詢：

```
select event.eventid, sum(pricepaid)
from event, sales
where event.eventid=sales.eventid
group by event.eventid order by 2 desc;
```

eventid	sum
289	51846.00
7895	51049.00
1602	50301.00
851	49956.00
7315	49823.00
...	

如果此查詢的 ID 為 15，則下列系統檢視查詢會傳回已完成的計劃節點。在此情況下，會保留節點的順序，以顯示執行的實際順序：

```
select query,nodeid,parentid,substring(plannode from 1 for 56)
from stl_explain where query=15 order by 1, 2 desc;
```

query	nodeid	parentid	substring
15	8	7	-> XN Seq Scan on eve
15	7	5	-> XN Hash(cost=87.98..87.9
15	6	5	-> XN Seq Scan on sales(cos
15	5	4	-> XN Hash Join DS_DIST_OUTER(cos
15	4	3	-> XN HashAggregate(cost=862286577.07..
15	3	2	-> XN Sort(cost=1000862287175.47..10008622871

```

15  | 2 | 1 | -> XN Network(cost=1000862287175.47..1000862287197.
15  | 1 | 0 | XN Merge(cost=1000862287175.47..1000862287197.46 rows=87
(8 rows)

```

下列查詢會擷取包含視窗函數IDs之任何查詢計劃的查詢：

```

select query, trim(plannode) from stl_explain
where plannode like '%Window%';

query|          btrim
-----+-----
26  | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
27  | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
(2 rows)

```

STL_FILE_SCAN

傳回使用 COPY命令載入資料時 Amazon Redshift 讀取的檔案。

查詢此檢視有助於對資料載入錯誤進行故障診斷。STL_FILE_SCAN 特別有助於找出平行資料載入中的問題，因為平行資料載入通常會使用單一COPY命令載入許多檔案。

STL所有使用者都可看見 _FILE_SCAN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_FILE_SCAN 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_LOAD_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。

欄名稱	資料類型	描述
分割	integer	識別執行查詢之配量的數字。
name	character(90)	已載入之檔案的完整路徑和名稱。
lines	bigint	從檔案讀取的行數。
位元組	bigint	從檔案讀取的位元組數。
loadtime	bigint	載入檔案所花費的時間 (以微秒為單位)。
curtime	Timestamp	代表 Amazon Redshift 開始處理檔案之時間的時間戳記。
is_partial	integer	如果 true (1) 表示輸入檔案在COPY操作期間分割為範圍的值。如果此值為 false (0) , 則輸入檔案不會分割。
start_offset	bigint	如果在COPY操作期間分割輸入檔案的值 , 會指出分割的偏移值 (以位元組為單位) 。如果檔案未分割 , 則此值為 0。

範例查詢

下列查詢擷取 Amazon Redshift 需要超過 1,000,000 微秒來讀取之任何檔案的名稱和載入時間。

```
select trim(name)as name, loadtime from stl_file_scan
where loadtime > 1000000;
```

此查詢傳回下列範例輸出。

```

      name                | loadtime
-----+-----
 listings_pipe.txt       | 9458354
 allusers_pipe.txt       | 2963761
 allevents_pipe.txt      | 1409135
 tickit/listings_pipe.txt | 7071087
 tickit/allevents_pipe.txt | 1237364
 tickit/allusers_pipe.txt | 2535138
 listings_pipe.txt       | 6706370
 allusers_pipe.txt       | 3579461
 allevents_pipe.txt      | 1313195
 tickit/allusers_pipe.txt | 3236060
```

```
ticket/listings_pipe.txt | 4980108
(11 rows)
```

STL_HASH

分析查詢的雜湊執行步驟。

STL所有使用者都可看見 _HASH。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_HASH 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。

欄名稱	資料類型	描述
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
slots	integer	雜湊儲存貯體的總數。
occupied	integer	包含記錄之插槽的總數。
maxlength	integer	最大插槽的大小。
tbl	integer	表格 ID。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體的位元組總數。
num_parts	integer	在雜湊步驟期間雜湊資料表已劃分的分割區總數。
est_rows	bigint	要雜湊之資料列的預估數目。
num_blocks_permitted	integer	此資訊僅供內部使用。
resizes	integer	此資訊僅供內部使用。
checksum	bigint	此資訊僅供內部使用。
runtime_filter_size	integer	執行時間篩選條件的大小，以位元組為單位。
max_runtime_filter_size	integer	執行時間篩選條件的大小上限，以位元組為單位。

範例查詢

下列範例會傳回雜湊中針對查詢 720 所使用之分割區數目的相關資訊，並指出已沒有步驟在磁碟上執行。

```
select slice, rows, bytes, occupied, workmem, num_parts, est_rows,
       num_blocks_permitted, is_diskbased
from stl_hash
where query=720 and segment=5
order by slice;
```

```
slice | rows | bytes | occupied | workmem | num_parts | est_rows |
num_blocks_permitted | is_diskbased
-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
      0 |  145 | 585800 |          1 | 88866816 |          16 |          1 |
52          |      | f      |          |          |          |          |
      1 |    0 |    0 |          0 |          0 |          16 |          1 |
52          |      | f      |          |          |          |          |
(2 rows)
```

STL_HASHJOIN

分析查詢的雜湊聯結執行步驟。

STL 所有使用者都可看見 _HASHJOIN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_HASHJOIN 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。
num_parts	integer	在雜湊步驟期間雜湊資料表已劃分的分割區總數。

欄名稱	資料類型	描述
join_type	integer	<p>步驟的聯結類型：</p> <ul style="list-style-type: none"> • 0. 查詢已使用內部聯結。 • 1. 查詢已使用左外部聯結。 • 2. 查詢已使用完整外部聯結。 • 3. 查詢已使用右外部聯結。 • 4. 查詢使用運算UNION子。 • 5. 查詢已使用 IN 條件。 • 6. 此資訊僅供內部使用。 • 7. 此資訊僅供內部使用。 • 8. 此資訊僅供內部使用。 • 9. 此資訊僅供內部使用。 • 10. 此資訊僅供內部使用。 • 11. 此資訊僅供內部使用。 • 12. 此資訊僅供內部使用。
hash_looped	character(1)	此資訊僅供內部使用。
switched_parts	character(1)	指出建置 (或外部) 和探測 (或內部) 端是否已切換。
used_prefetching	character(1)	此資訊僅供內部使用。
hash_segment	integer	對應雜湊步驟的區段。
hash_step	integer	對應雜湊步驟的步驟號碼。
checksum	bigint	此資訊僅供內部使用。
分佈	integer	此資訊僅供內部使用。

範例查詢

下列查詢會傳回雜湊連結中針對查詢 720 所使用的分割區數目。

```
select query, slice, tbl, num_parts
from stl_hashjoin
where query=720 limit 10;
```

```
query | slice | tbl | num_parts
-----+-----+-----+-----
  720 |     0 | 243 |         1
  720 |     1 | 243 |         1
(2 rows)
```

STL_INSERT

分析查詢的插入執行步驟。

STL 所有使用者都可看見 `_INSERT`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_INSERT 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。

欄名稱	資料類型	描述
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。
inserted_mega_value	character(1)	此資訊僅供內部使用。此資訊顯示指定的插入步驟是否已插入較大的值。大值將儲存在多個區塊中。區塊大小預設為 1 MB，在預設設定中，大值大於 1 MB。

範例查詢

下列範例會傳回最新查詢的插入執行步驟。

```
select slice, segment, step, tasknum, rows, tbl
from stl_insert
where query=pg_last_query_id();
```

```
 slice | segment | step | tasknum | rows | tbl
-----+-----+-----+-----+-----+-----
      0 |         2 |     2 |        15 | 24958 | 100548
      1 |         2 |     2 |        15 | 25032 | 100548
(2 rows)
```

STL_LIMIT

分析在SELECT查詢中使用LIMIT子句時發生的執行步驟。

STL所有使用者都可看見 `_LIMIT`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_LIMIT 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

為了在 STL_ 中產生資料列LIMIT，此範例會先使用 LIMIT子句針對VENUE資料表執行下列查詢。

```
select * from venue
order by 1
limit 10;
```

venueid	venue name	venue city	venue state	venue seats
1	Toyota Park	Bridgeview	IL	0
2	Columbus Crew Stadium	Columbus	OH	0
3	RFK Stadium	Washington	DC	0
4	CommunityAmerica Ballpark	Kansas City	KS	0
5	Gillette Stadium	Foxborough	MA	68756
6	New York Giants Stadium	East Rutherford	NJ	80242
7	BMO Field	Toronto	ON	0
8	The Home Depot Center	Carson	CA	0
9	Dick's Sporting Goods Park	Commerce City	CO	0
10	Pizza Hut Park	Frisco	TX	0

(10 rows)

接下來，執行下列查詢，以尋找您針對VENUE資料表執行之最後一個查詢的查詢 ID。

```
select max(query)
from stl_query;
```

```
max
-----
127128
(1 row)
```

或者，您可以執行下列查詢，以確認查詢 ID 對應於您先前執行的LIMIT查詢。

```
select query, trim(querytxt)
from stl_query
where query=127128;
```

```
query | btrim
```

```
-----+-----
127128 | select * from venue order by 1 limit 10;
(1 row)
```

最後，執行下列查詢，從 STL_LIMIT 資料表傳回有關LIMIT查詢的資訊。

```
select slice, segment, step, starttime, endtime, tasknum
from stl_limit
where query=127128
order by starttime, endtime;
```

```
slice | segment | step |          starttime          |          endtime          |
tasknum
-----+-----+-----+-----+-----+-----
+-----+
      1 |         1 |     3 | 2013-09-06 22:56:43.608114 | 2013-09-06 22:56:43.609383 |
    15
      0 |         1 |     3 | 2013-09-06 22:56:43.608708 | 2013-09-06 22:56:43.609521 |
    15
 10000 |         2 |     2 | 2013-09-06 22:56:43.612506 | 2013-09-06 22:56:43.612668 |
      0
(3 rows)
```

STL_LOAD_COMMITS

傳回資訊以追蹤資料載入或對其進行故障診斷。

此檢視會在每一個資料檔案載入至資料庫資料表時記錄其進度。

STL所有使用者都可看見 _LOAD_COMMITS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_LOAD_COMMITS 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_LOAD_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	針對此項目載入的配量。
name	character(256)	系統定義的值。
filename	character(256)	正在追蹤之檔案的名稱。
byte_offset	integer	此資訊僅供內部使用。
lines_scanned	integer	從載入檔案掃描的行數。此數目可能不符合實際載入的資料列數目。例如，根據 COPY 命令中的 MAXERROR 選項，負載可能會掃描但可容忍許多不良記錄。
錯誤	integer	此資訊僅供內部使用。
curtime	timestamp	上次更新此項目的時間。
status	integer	此資訊僅供內部使用。
file_format	character(16)	載入檔案的格式。可能的值如下： <ul style="list-style-type: none"> • Avro • JSON • ORC • Parquet • 文字
is_partial	integer	如果 true (1) 表示輸入檔案在 COPY 操作期間分割為範圍的值。如果此值為 false (0)，則輸入檔案不會分割。

欄名稱	資料類型	描述
start_offset	bigint	如果在COPY操作期間分割輸入檔案的值，會指出分割的偏移值（以位元組為單位）。每個檔案分割都記錄為具有對應start_offset值的單獨記錄。如果檔案未分割，則此值為0。
copy_job_id	bigint	複製任務識別碼。0表示沒有任務識別碼。

範例查詢

下列範例會傳回上次COPY操作的詳細資訊。

```
select query, trim(filename) as file, curtime as updated
from stl_load_commits
where query = pg_last_copy_id();
```

```
query |          file          |          updated
-----+-----+-----
28554 | s3://dw-tickit/category_pipe.txt | 2013-11-01 17:14:52.648486
(1 row)
```

下列查詢包含TICKIT資料庫中資料表新載入的項目：

```
select query, trim(filename), curtime
from stl_load_commits
where filename like '%tickit%' order by query;
```

```
query |          btrim          |          curtime
-----+-----+-----
22475 | tickit/allusers_pipe.txt | 2013-02-08 20:58:23.274186
22478 | tickit/venue_pipe.txt    | 2013-02-08 20:58:25.070604
22480 | tickit/category_pipe.txt | 2013-02-08 20:58:27.333472
22482 | tickit/date2008_pipe.txt | 2013-02-08 20:58:28.608305
22485 | tickit/allevvents_pipe.txt | 2013-02-08 20:58:29.99489
22487 | tickit/listings_pipe.txt | 2013-02-08 20:58:37.632939
22593 | tickit/allusers_pipe.txt | 2013-02-08 21:04:08.400491
22596 | tickit/venue_pipe.txt    | 2013-02-08 21:04:10.056055
22598 | tickit/category_pipe.txt | 2013-02-08 21:04:11.465049
22600 | tickit/date2008_pipe.txt | 2013-02-08 21:04:12.461502
22603 | tickit/allevvents_pipe.txt | 2013-02-08 21:04:14.785124
22605 | tickit/listings_pipe.txt | 2013-02-08 21:04:20.170594
```

(12 rows)

事實上，記錄寫入至此系統檢視的日誌檔案，並不表示已成功遞交載入，做為其包含交易的一部分。若要驗證負載遞交，請查詢 STL_UTILITYTEXT 檢視並尋找與COPY交易對應的COMMIT記錄。例如，此查詢LOADSTL會根據 STL_COMMITSQUERYSTLUTILITYTEXT：

```
select l.query,rtrim(l.filename),q.xid
from stl_load_commits l, stl_query q
where l.query=q.query
and exists
(select xid from stl_utilitytext where xid=q.xid and rtrim("text")='COMMIT');
```

query	rtrim	xid
22600	ticket/date2008_pipe.txt	68311
22480	ticket/category_pipe.txt	68066
7508	allusers_pipe.txt	23365
7552	category_pipe.txt	23415
7576	allevents_pipe.txt	23429
7516	venue_pipe.txt	23390
7604	listings_pipe.txt	23445
22596	ticket/venue_pipe.txt	68309
22605	ticket/listings_pipe.txt	68316
22593	ticket/allusers_pipe.txt	68305
22485	ticket/allevents_pipe.txt	68071
7561	allevents_pipe.txt	23429
7541	category_pipe.txt	23415
7558	date2008_pipe.txt	23428
22478	ticket/venue_pipe.txt	68065
526	date2008_pipe.txt	2572
7466	allusers_pipe.txt	23365
22482	ticket/date2008_pipe.txt	68067
22598	ticket/category_pipe.txt	68310
22603	ticket/allevents_pipe.txt	68315
22475	ticket/allusers_pipe.txt	68061
547	date2008_pipe.txt	2572
22487	ticket/listings_pipe.txt	68072
7531	venue_pipe.txt	23390
7583	listings_pipe.txt	23445

(25 rows)

以下範例反白顯示 is_partial 和 start_offset 欄值。

```
-- Single large file copy without scan range
SELECT count(*) FROM stl_load_commits WHERE query = pg_last_copy_id();
1

-- Single large uncompressed, delimited file copy with scan range
SELECT count(*) FROM stl_load_commits WHERE query = pg_last_copy_id();
16

-- Scan range offset logging in the file at 64MB boundary.
SELECT start_offset FROM stl_load_commits
WHERE query = pg_last_copy_id() ORDER BY start_offset;
0
67108864
134217728
201326592
268435456
335544320
402653184
469762048
536870912
603979776
671088640
738197504
805306368
872415232
939524096
1006632960
```

STL_LOAD_ERRORS

顯示所有 Amazon Redshift 載入錯誤的記錄。

STL_LOAD_ERRORS 包含所有 Amazon Redshift 載入錯誤的歷史記錄。如需可能載入錯誤和說明的完整清單，請參閱[載入錯誤參考](#)。

在您查詢 STL_LOAD_ERRORS 以尋找有關錯誤的一般資訊之後，[STL_LOADERERROR_DETAIL](#)查詢其他詳細資訊，例如發生剖析錯誤的確切資料列和資料欄。

STL所有使用者都可看見 _LOAD_ERRORS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_LOAD_ERRORS 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_LOAD_ERROR_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
分割	integer	發生錯誤的配量。
tbl	integer	表格 ID。
starttime	timestamp	載入UTC的開始時間為。
session	integer	執行載入之工作階段的工作階段 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
filename	character(256)	用於載入之輸入檔案的完整路徑。
line_number	bigint	載入檔案中發生錯誤的行號。對於COPY來自的JSON，具有錯誤的JSON物件最後一行的行號。
colname	character(127)	發生錯誤的欄位。
type	character(10)	欄位的資料類型。
col_length	character(10)	資料欄長度 (如適用)。當資料類型具有限制長度時，會填入此欄位。例如，對於資料類型為 "character(3)" 的資料欄，此資料欄將包含值 "3"。
position	integer	欄位中錯誤的位置。
raw_line	character(1024)	包含錯誤的原始載入資料。載入資料中的多位元組字元會取代為句點。

欄名稱	資料類型	描述
raw_field_value	char(1024)	導致剖析錯誤之欄位 "colname" 的預先剖析值。
err_code	integer	錯誤代碼。
err_reason	character(100)	錯誤的說明。
is_partial	integer	如果 true (1) 表示輸入檔案在COPY操作期間分割為範圍的值。如果此值為 false (0) , 則輸入檔案不會分割。
start_offset	bigint	如果在COPY操作期間分割輸入檔案的值 , 會指出分割的偏移值 (以位元組為單位)。如果檔案中的行號不明 , 則行號為 -1。如果檔案未分割 , 則此值為 0。
copy_job_id	bigint	複製任務識別碼。0 表示沒有任務識別碼。

範例查詢

下列查詢會聯結 STL_LOAD_ERRORS 至 STL_LOADERROR_DETAIL , 以檢視最近載入期間發生的詳細資訊錯誤。

```
select d.query, substring(d.filename,14,20),
d.line_number as line,
substring(d.value,1,16) as value,
substring(le.err_reason,1,48) as err_reason
from stl_loadererror_detail d, stl_load_errors le
where d.query = le.query
and d.query = pg_last_copy_id();
```

query	substring	line	value	err_reason
558	allusers_pipe.txt	251	251	String contains invalid or unsupported UTF8 code
558	allusers_pipe.txt	251	ZRU29FGR	String contains invalid or unsupported UTF8 code
558	allusers_pipe.txt	251	Kaitlin	String contains invalid or unsupported UTF8 code
558	allusers_pipe.txt	251	Walter	String contains invalid or

```
unsupported UTF8 code
```

下列範例使用 STL_LOAD_ERRORS 搭配 STV_TBL_PERM 建立新的檢視，然後使用該檢視來判斷將資料載入EVENT資料表時發生的錯誤：

```
create view loadview as
(select distinct tbl, trim(name) as table_name, query, starttime,
trim(filename) as input, line_number, colname, err_code,
trim(err_reason) as reason
from stl_load_errors sl, stv_tbl_perm sp
where sl.tbl = sp.id);
```

接下來，下列查詢實際上會傳回載入EVENT資料表時發生的最後一個錯誤：

```
select table_name, query, line_number, colname, starttime,
trim(reason) as error
from loadview
where table_name = 'event'
order by line_number limit 1;
```

查詢會傳回EVENT資料表上次發生的載入錯誤。如果未發生任何載入錯誤，則查詢不會傳回任何資料列。在此範例中，查詢會傳回單一錯誤：

```
table_name | query | line_number | colname | error | starttime
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
event | 309 | 0 | 5 | Error in Timestamp value or format [%Y-%m-%d %H:%M:%S] |
2014-04-22 15:12:44

(1 row)
```

如果COPY命令自動分割大型、未壓縮、以文字分隔的檔案資料以促進平行處理，則 line_number、is_partial 和 start_offset 資料欄會顯示與分割相關的資訊。(如果原始檔案中的行號不可用，則行號可能是未知的。)

```
--scan ranges information
SELECT line_number, POSITION, btrim(raw_line), btrim(raw_field_value),
btrim(err_reason), is_partial, start_offset FROM stl_load_errors
WHERE query = pg_last_copy_id();
```

```
--result
-1,51,"1008771|13463413|463414|2|28.00|38520.72|0.06|0.07|N0|1998-08-30|1998-09-25|
1998-09-04|TAKE BACK RETURN|RAIL|ans cajole sly","N0","Char length exceeds DDL
length",1,67108864
```

STL_LOADERROR_DETAIL

顯示使用 COPY 命令載入資料表時發生的資料剖析錯誤日誌。若要保留磁碟空間，針對每個載入操作，每個節點配量最多可記錄 20 個錯誤。

將資料列中的欄位載入至資料表時，若 Amazon Redshift 無法剖析該欄位，即會發生剖析錯誤。例如，如果資料表資料欄預期 integer 資料類型，但資料檔案在該欄位中包含一串字母，則其會導致剖析錯誤。

查詢後，查詢 STL_LOADERROR_DETAIL 以取得其他詳細資訊，例如發生剖析錯誤的確切資料列和資料欄，[STL_LOAD_ERRORS](#) 以尋找有關錯誤的一般資訊。

STL_LOADERROR_DETAIL 檢視包含所有資料欄，包括發生剖析錯誤的資料欄之前的和。使用 VALUE 欄位來查看此欄中實際剖析的資料值，包括正確剖析至錯誤的資料欄。

所有使用者都可看見此檢視。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_LOADERROR_DETAIL 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用 SYS 監控檢視 [SYS_LOAD_ERROR_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
分割	integer	發生錯誤的配量。
session	integer	執行載入之工作階段的工作階段 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
filename	character(256)	用於載入之輸入檔案的完整路徑。
line_number	bigint	載入檔案中發生錯誤的行號。
field	integer	發生錯誤的欄位。
colname	character(1024)	資料欄名稱。
value	character(1024)	欄位的已剖析資料值。(可能已截斷。) 載入資料中的多位元組字元會取代為句點。
is_null	integer	已剖析值是否為 null。
type	character(10)	欄位的資料類型。
col_length	character(10)	資料欄長度 (如適用)。當資料類型具有限制長度時，會填入此欄位。例如，對於資料類型為 "character(3)" 的資料欄，此資料欄將包含值 "3"。

範例查詢

下列查詢會聯結 STL_LOAD_ERRORS 至 STL_LOADERROR_DETAIL，以檢視載入 EVENT 資料表時發生剖析錯誤的詳細資訊，資料表 ID 為 100133：

```
select d.query, d.line_number, d.value,
le.raw_line, le.err_reason
from stl_loaderror_detail d, stl_load_errors le
where
d.query = le.query
and tbl = 100133;
```

下列範例輸出會顯示成功載入的資料欄，包括發生錯誤的資料欄。在此範例中，於第三個資料欄中發生剖析錯誤之前已成功載入兩個資料欄，此錯誤是對於預期整數的欄位，卻不正確地剖析字元字串。因為欄位預期整數，所以它將字串 "aaa" (未初始化的資料) 剖析為 null，並產生剖析錯誤。輸出顯示原始值、已剖析值和錯誤原因：

```

query | line_number | value | raw_line | err_reason
-----+-----+-----+-----+-----
4      |      3      | 1201 | 1201     | Invalid digit
4      |      3      | 126  | 126      | Invalid digit
4      |      3      |      | aaa      | Invalid digit
(3 rows)

```

當查詢加入 `STL_LOAD_ERRORS` 和 `STL_LOADERROR_` 時 `DETAIL`，它會顯示資料列中每個資料欄的錯誤原因，這只是表示該資料列中發生錯誤。結果中的最後一列是發生剖析錯誤的實際資料欄。

STL_MERGE

分析查詢的合併執行步驟。當合併平行操作 (例如排序和聯結) 的結果進行後續處理時，即會進行這些步驟。

STL 所有使用者都可看見 `_MERGE`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

STL_MERGE 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用 SYS 監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。

欄名稱	資料類型	描述
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。

範例查詢

下列範例會傳回 10 個合併執行結果。

```
select query, step, starttime, endtime, tasknum, rows
from stl_merge
limit 10;
```

```
query | step |          starttime          |          endtime          | tasknum | rows
-----+-----+-----+-----+-----+-----
    9 |    0 | 2013-08-12 20:08:14 | 2013-08-12 20:08:14 |    0 |    0
   12 |    0 | 2013-08-12 20:09:10 | 2013-08-12 20:09:10 |    0 |    0
   15 |    0 | 2013-08-12 20:10:24 | 2013-08-12 20:10:24 |    0 |    0
   20 |    0 | 2013-08-12 20:11:27 | 2013-08-12 20:11:27 |    0 |    0
   26 |    0 | 2013-08-12 20:12:28 | 2013-08-12 20:12:28 |    0 |    0
   32 |    0 | 2013-08-12 20:14:33 | 2013-08-12 20:14:33 |    0 |    0
   38 |    0 | 2013-08-12 20:16:43 | 2013-08-12 20:16:43 |    0 |    0
   44 |    0 | 2013-08-12 20:17:05 | 2013-08-12 20:17:05 |    0 |    0
   50 |    0 | 2013-08-12 20:18:48 | 2013-08-12 20:18:48 |    0 |    0
   56 |    0 | 2013-08-12 20:20:48 | 2013-08-12 20:20:48 |    0 |    0
(10 rows)
```

STL_MERGEJOIN

分析查詢的合併聯結執行步驟。

STL所有使用者都可看見 `_MERGEJOIN`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_MERGEJOIN 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。這是已在合併聯結中使用之內部資料表的 ID。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例會傳回最新查詢的合併聯結結果。

```
select sum(s.qtysold), e.eventname
from event e, listing l, sales s
where e.eventid=l.eventid
and l.listid= s.listid
group by e.eventname;

select * from stl_mergejoin where query=pg_last_query_id();
```

```
userid | query | slice | segment | step |          starttime          |          endtime          |
tasknum | rows | tbl
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
  100 | 27399 |    3 |      4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
    19 | 43428 | 240
  100 | 27399 |    0 |      4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
    19 | 43159 | 240
  100 | 27399 |    2 |      4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
    19 | 42778 | 240
  100 | 27399 |    1 |      4 |    4 | 2013-10-02 16:30:41 | 2013-10-02 16:30:41 |
    19 | 43091 | 240
```

STL_MV_STATE

STL_MV_STATE 檢視包含具體化檢視的每個狀態轉換的資料列。

如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

STL 所有使用者都可看見 _MV_STATE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到 [SYS_MV_STATE](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	bigint	建立事件的使用者 ID。

欄名稱	資料類型	描述
starttime	timestamp	事件的開始時間。
xid	bigint	事件的交易 ID。
event_desc	char(500)	提示狀態變更的事件。一些範例值包括如下： <ul style="list-style-type: none"> • 已變更資料欄類型 • 已捨棄資料欄 • 已重新命名資料欄 • 已變更結構描述名稱 • 小型資料表轉換 • TRUNCATE • Vacuum <p>請注意，此欄還有其他可能的值。</p>
db_name	char(128)	包含具體化視觀表的資料庫。
base_table_schema	char(128)	基底資料表的結構描述。
base_table_name	char(128)	基底資料表的名稱。
mv_schema	char(128)	具體化視觀表的結構描述。
mv_name	char(128)	具體化視觀表的名稱。
state	character(32)	具體化視觀表的變更後狀態如下： <ul style="list-style-type: none"> • 重新運算 • 無法重新整理

下表顯示 event_desc 和 state 範例組合。

event_desc	state
TRUNCATE	Recompute
TRUNCATE	Recompute
Small table conversion	Recompute
Vacuum	Recompute
Column was renamed	Unrefreshable
Column was dropped	Unrefreshable
Table was renamed	Unrefreshable
Column type was changed	Unrefreshable
Schema name was changed	Unrefreshable

範例查詢

若要檢視具體化視觀表的狀態轉換日誌，請執行下列查詢。

```
select * from stl_mv_state;
```

此查詢傳回下列範例輸出：

```
userid |          starttime          | xid |          event_desc          | db_name |
base_table_schema | base_table_name | mv_schema | mv_name |
state
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
+-----
    138 | 2020-02-14 02:21:25.578885 | 5180 | TRUNCATE | dev |
public | mv_base_table | public | mv_test |
Recompute
    138 | 2020-02-14 02:21:56.846774 | 5275 | Column was dropped | dev |
      | mv_base_table | public | mv_test |
Unrefreshable
    100 | 2020-02-13 22:09:53.041228 | 1794 | Column was renamed | dev |
      | mv_base_table | public | mv_test |
Unrefreshable
     1 | 2020-02-13 22:10:23.630914 | 1893 | ALTER TABLE ALTER SORTKEY | dev |
public | mv_base_table_sorted | public | mv_test |
Recompute
     1 | 2020-02-17 22:57:22.497989 | 8455 | ALTER TABLE ALTER DISTSTYLE | dev |
public | mv_base_table | public | mv_test |
Recompute
```

```

173 | 2020-02-17 22:57:23.591434 | 8504 | Table was renamed | dev |
      | mv_base_table | public | mv_test |
Unrefreshable
173 | 2020-02-17 22:57:27.229423 | 8592 | Column type was changed | dev |
      | mv_base_table | public | mv_test |
Unrefreshable
197 | 2020-02-17 22:59:06.212569 | 9668 | TRUNCATE | dev |
schemaf796e415850f4f | mv_base_table | schemaf796e415850f4f | mv_test |
Recompute
138 | 2020-02-14 02:21:55.705655 | 5226 | Column was renamed | dev |
      | mv_base_table | public | mv_test |
Unrefreshable
1 | 2020-02-14 02:22:26.292434 | 5325 | ALTER TABLE ALTER SORTKEY | dev |
public | mv_base_table_sorted | public | mv_test |
Recompute

```

STL_NESTLOOP

分析查詢的巢狀迴路聯結執行步驟。

STL所有使用者都可看見 `_NESTLOOP`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_NESTLOOP 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。

欄名稱	資料類型	描述
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
tbl	integer	表格 ID。
checksum	bigint	此資訊僅供內部使用。

範例查詢

由於下列查詢忽略加入CATEGORY資料表，因此會產生不建議的部分卡氏產品。在這裡顯示它，是為了說明巢狀迴路。

```
select count(event.eventname), event.eventname, category.catname, date.caldate
from event, category, date
where event.dateid = date.dateid
group by event.eventname, category.catname, date.caldate;
```

下列查詢顯示 STL_NESTLOOP 檢視中先前查詢的結果。

```
select query, slice, segment as seg, step,
datediff(msec, starttime, endtime) as duration, tasknum, rows, tbl
from stl_nestloop
where query = pg_last_query_id();
```

```
query | slice | seg | step | duration | tasknum | rows | tbl
```

```

-----+-----+-----+-----+-----+-----+-----+-----+-----
6028 |      0 |    4 |    5 |      41 |      22 | 24277 | 240
6028 |      1 |    4 |    5 |      26 |      23 | 24189 | 240
6028 |      3 |    4 |    5 |      25 |      23 | 24376 | 240
6028 |      2 |    4 |    5 |      54 |      22 | 23936 | 240

```

STL_PARSE

分析將字串剖析為二進位值以進行載入的查詢步驟。

STL所有使用者都可看見 `_PARSE`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_PARSE 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

欄名稱	資料類型	描述
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。

範例查詢

下列範例會傳回配量 1 和區段 0 的所有查詢步驟，其中字串已剖析為二進位值。

```
select query, step, starttime, endtime, tasknum, rows
from stl_parse
where slice=1 and segment=0;
```

query	step	starttime	endtime	tasknum	rows
669	1	2013-08-12 22:35:13	2013-08-12 22:35:17	32	192497
696	1	2013-08-12 22:35:49	2013-08-12 22:35:49	32	0
525	1	2013-08-12 22:32:03	2013-08-12 22:32:03	13	49990
585	1	2013-08-12 22:33:18	2013-08-12 22:33:19	13	202
621	1	2013-08-12 22:34:03	2013-08-12 22:34:03	27	365
651	1	2013-08-12 22:34:47	2013-08-12 22:34:53	35	192497
590	1	2013-08-12 22:33:28	2013-08-12 22:33:28	19	0
599	1	2013-08-12 22:33:39	2013-08-12 22:33:39	31	11
675	1	2013-08-12 22:35:26	2013-08-12 22:35:27	38	3766
567	1	2013-08-12 22:32:47	2013-08-12 22:32:48	23	49990
630	1	2013-08-12 22:34:17	2013-08-12 22:34:17	36	0
572	1	2013-08-12 22:33:04	2013-08-12 22:33:04	29	0
645	1	2013-08-12 22:34:37	2013-08-12 22:34:38	29	8798
604	1	2013-08-12 22:33:47	2013-08-12 22:33:47	37	0

(14 rows)

STL_PLAN_INFO

使用 STL_PLAN_INFO 檢視來查看一組資料列中查詢的 EXPLAIN 輸出。這是查看查詢計劃的其他方法。

STL所有使用者都可看見 `_PLAN_INFO`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_PLAN_INFO 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別碼，其中節點會在查詢執行時映射至一個或多個步驟。
segment	integer	識別查詢區段的號碼。
step	integer	識別查詢步驟的號碼。
locus	integer	執行步驟的位置。若在運算節點上則為 0，若在領導者節點上則為 1。
plannode	integer	計劃節點的列舉值。請查看下表以取得 plannode 的列舉值。 (中的資料PLANNODE欄 STL_EXPLAIN 包含計劃節點文字。)
startupcost	double precision	傳回此步驟之第一列的預估相對成本。
totalcost	double precision	執行步驟的預估相對成本。
rows	bigint	步驟將產生之資料列的預估數目。
位元組	bigint	步驟將產生之位元組的預估數目。

範例查詢

下列範例會比較使用 EXPLAIN 命令和查詢 STL_PLAN_INFO 檢視傳回的簡單 SELECT 查詢的查詢計劃。

```
explain select * from category;
QUERY PLAN
-----
XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
(1 row)

select * from category;
catid | catgroup | catname | catdesc
-----+-----+-----+-----
 1 | Sports | MLB | Major League Baseball
 3 | Sports | NFL | National Football League
 5 | Sports | MLS | Major League Soccer
...

select * from stl_plan_info where query=256;

query | nodeid | segment | step | locus | plannode | startupcost | totalcost
| rows | bytes
-----+-----+-----+-----+-----+-----+-----+-----
+-----
256 | 1 | 0 | 1 | 0 | 104 | 0 | 0.11 | 11 | 539
256 | 1 | 0 | 0 | 0 | 104 | 0 | 0.11 | 11 | 539
(2 rows)
```

在此範例中，104 PLANNODE 是指 CATEGORY 資料表的循序掃描。

```
select distinct eventname from event order by 1;

eventname
-----
.38 Special
3 Doors Down
70s Soul Jam
A Bronx Tale
...

explain select distinct eventname from event order by 1;
```

QUERY PLAN

```
-----
XN Merge (cost=1000000000136.38..1000000000137.82 rows=576 width=17)
```

```
Merge Key: eventname
```

```
-> XN Network (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
```

```
Send to leader
```

```
-> XN Sort (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
```

```
Sort Key: eventname
```

```
-> XN Unique (cost=0.00..109.98 rows=576 width=17)
```

```
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=17)
```

```
(8 rows)
```

```
select * from stl_plan_info where query=240 order by nodeid desc;
```

```
query | nodeid | segment | step | locus | plannode | startupcost |
totalcost | rows | bytes
```

```
-----+-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----+-----
240 | 5 | 0 | 0 | 0 | 104 | 0 | 87.98 | 8798 | 149566
240 | 5 | 0 | 1 | 0 | 104 | 0 | 87.98 | 8798 | 149566
240 | 4 | 0 | 2 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 4 | 0 | 3 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 4 | 1 | 0 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 4 | 1 | 1 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 3 | 1 | 2 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 3 | 2 | 0 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 2 | 2 | 1 | 0 | 123 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 1 | 3 | 0 | 0 | 122 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
(10 rows)
```

STL_PROJECT

包含用來評估表達式之查詢步驟的資料列。

STL所有使用者都可看見 `_PROJECT`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_PROJECT 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例會傳回查詢步驟的所有資料列，而這些查詢步驟是用來評估配量 0 和區段 1 的表達式。

```
select query, step, starttime, endtime, tasknum, rows
```

```
from stl_project
where slice=0 and segment=1;
```

query	step	starttime	endtime	tasknum	rows
86399	2	2013-08-29 22:01:21	2013-08-29 22:01:21	25	-1
86399	3	2013-08-29 22:01:21	2013-08-29 22:01:21	25	-1
719	1	2013-08-12 22:38:33	2013-08-12 22:38:33	7	-1
86383	1	2013-08-29 21:58:35	2013-08-29 21:58:35	7	-1
714	1	2013-08-12 22:38:17	2013-08-12 22:38:17	2	-1
86375	1	2013-08-29 21:57:59	2013-08-29 21:57:59	2	-1
86397	2	2013-08-29 22:01:20	2013-08-29 22:01:20	19	-1
627	1	2013-08-12 22:34:13	2013-08-12 22:34:13	34	-1
86326	2	2013-08-29 21:45:28	2013-08-29 21:45:28	34	-1
86326	3	2013-08-29 21:45:28	2013-08-29 21:45:28	34	-1
86325	2	2013-08-29 21:45:27	2013-08-29 21:45:27	28	-1
86371	1	2013-08-29 21:57:42	2013-08-29 21:57:42	4	-1
111100	2	2013-09-03 19:04:45	2013-09-03 19:04:45	12	-1
704	2	2013-08-12 22:36:34	2013-08-12 22:36:34	37	-1
649	2	2013-08-12 22:34:47	2013-08-12 22:34:47	38	-1
649	3	2013-08-12 22:34:47	2013-08-12 22:34:47	38	-1
632	2	2013-08-12 22:34:22	2013-08-12 22:34:22	13	-1
705	2	2013-08-12 22:36:48	2013-08-12 22:36:49	13	-1
705	3	2013-08-12 22:36:48	2013-08-12 22:36:49	13	-1
3	1	2013-08-12 20:07:40	2013-08-12 20:07:40	3	-1
86373	1	2013-08-29 21:57:58	2013-08-29 21:57:58	3	-1
107976	1	2013-09-03 04:05:12	2013-09-03 04:05:12	3	-1
86381	1	2013-08-29 21:58:35	2013-08-29 21:58:35	8	-1
86396	1	2013-08-29 22:01:20	2013-08-29 22:01:20	15	-1
711	1	2013-08-12 22:37:10	2013-08-12 22:37:10	20	-1
86324	1	2013-08-29 21:45:27	2013-08-29 21:45:27	24	-1

(26 rows)

STL_QUERY

傳回有關資料庫查詢的執行資訊。

Note

STL_QUERY 和 STL_QUERYTEXT 檢視僅包含查詢的相關資訊，不包含其他公用程式和DDL命令。如需 Amazon Redshift 執行之所有陳述式的清單和資訊，您也可以查詢 STL_DDLTEXT

和 STL_UTILITYTEXT 檢視。如需 Amazon Redshift 執行之所有陳述式的完整清單，您可以查詢 SVL_STATEMENTTEXT 檢視。

STL所有使用者都可看見 _QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_QUERY_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
label	character(320)	用於執行查詢的檔案名稱，或使用 QUERY_SET GROUP 命令定義的標籤。如果查詢不是以檔案為基礎，或未設定 QUERY_GROUP 參數，則此欄位值為 default。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。正常情況下，工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。在某些內部事件之後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 PID。如需詳細資訊，請參閱 STL_RESTARTED_SESSIONS 。
database	character(32)	當查詢發出時，要將使用者連接至其中的資料庫名稱。
querytxt	character(4000)	查詢的實際查詢文字。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

欄名稱	資料類型	描述
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 1 。如果查詢完成執行 (包括將結果傳回至用戶端)，則此資料欄包含 0 。如果用戶端在接收結果之前中斷連接，則查詢將標示為已取消 (1)，即使它已在後端成功完成也一樣。
insert_pr istine	integer	當目前查詢正在執行時，寫入查詢是否能夠執行。1 = 不允許寫入查詢。0 = 允許寫入查詢。此欄位適用於偵錯。
concurrent cy_scalin g_status	integer	指出查詢是執行於主要叢集或並行擴展叢集。可能的值如下： 0 - 執行於主要叢集 1 - 執行於並行擴展叢集 大於 1 - 執行於主要叢集

範例查詢

下列查詢會列出五個最新查詢。

```
select query, trim(querytxt) as sqlquery
from stl_query
order by query desc limit 5;
```

```
query |          sqlquery
-----+-----
129 | select query, trim(querytxt) from stl_query order by query;
128 | select node from stv_disk_read_speeds;
127 | select system_status from stv_gui_status
126 | select * from systable_topology order by slice
125 | load global dict registry
(5 rows)
```

下列查詢會針對 2013 年 2 月 15 日執行的查詢，依遞減順序傳回經過時間。

```
select query, datediff(seconds, starttime, endtime),
trim(querytxt) as sqlquery
from stl_query
where starttime >= '2013-02-15 00:00' and endtime < '2013-02-16 00:00'
order by date_diff desc;
```

query	date_diff	sqlquery
55	119	padb_fetch_sample: select count(*) from category
121	9	select * from svl_query_summary;
181	6	select * from svl_query_summary where query in(179,178);
172	5	select * from svl_query_summary where query=148;
...		

(189 rows)

下列查詢顯示查詢的佇列時間和執行時間。concurrency_scaling_status = 1 的查詢執行於並行擴展叢集。所有其他查詢皆執行於主要叢集。

```
SELECT w.service_class AS queue
      , q.concurrency_scaling_status
      , COUNT( * ) AS queries
      , SUM( q.aborted ) AS aborted
      , SUM( ROUND( total_queue_time::NUMERIC / 1000000,2 ) ) AS queue_secs
      , SUM( ROUND( total_exec_time::NUMERIC / 1000000,2 ) ) AS exec_secs
FROM stl_query q
     JOIN stl_wlm_query w
         USING (userid,query)
WHERE q.userid > 1
      AND service_class > 5
      AND q.starttime > '2019-03-01 16:38:00'
      AND q.endtime < '2019-03-01 17:40:00'
GROUP BY 1,2
ORDER BY 1,2;
```

STL_QUERY_METRICS

對於在使用者定義的查詢佇列（服務類別）中執行的查詢，包含指標資訊，例如處理的列數、CPU用量、輸入/輸出和磁碟使用量。若要檢視目前執行中之作用中查詢的指標，請參閱 [STV_QUERY_METRICS](#) 系統檢視。

查詢指標每間隔一秒鐘取樣一次。因此，相同查詢的不同執行可能會傳回稍微不同的時間。另外，可能不會記錄執行時間不到一秒的查詢區段。

STL_QUERY_METRICS 在查詢、區段和步驟層級追蹤和彙總指標。如需查詢區段和步驟的相關資訊，請參閱[查詢計劃和執行工作流程](#)。許多指標 (例如 max_rows、cpu_time 等等) 是跨節點配量加總的。如需節點配量的相關資訊，請參閱[資料倉儲系統架構](#)。

若要判斷資料列報告指標的層級，請檢查 segment 和 step_type 資料欄。

- 如果 segment 和 step_type 皆為 -1，則資料列報告查詢層級的指標。
- 如果 segment 不是 -1，且 step_type 是 -1，則資料列報告區段層級的指標。
- 如果 segment 和 step_type 皆不是 -1，則資料列報告步驟層級的指標。

[SVL_QUERY_METRICS](#) 檢視和 [SVL_QUERY_METRICS_SUMMARY](#) 檢視彙總此檢視中的資料，並以更易存取的形式呈現資訊。

STL 所有使用者都可看見 _QUERY_METRICS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
service_class	integer	服務類別的 ID。查詢佇列在WLM組態中定義。只會回報使用者定義之佇列的指標。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。如果區段值為 -1，指標區段值將累計至查詢層級。
step_type	integer	執行的步驟類型。如需步驟類型的說明，請參閱 步驟類型 。

欄名稱	資料類型	描述
starttime	timestamp	UTC 查詢開始執行的時間，其中 6 位數的精確度為分數秒。例如：2009-06-12 11:29:19.131358 。
slices	integer	叢集的分割數量。
max_rows	bigint	為步驟匯出的最大資料列數 (彙總所有分割)。
rows	bigint	步驟處理的資料列數。
max_cpu_time	bigint	使用的最長CPU時間，以微秒為單位。在區段層級，區段在所有層中使用的最長CPU時間。在查詢層級，任何查詢區段所使用的最長CPU時間。
cpu_time	bigint	CPU 使用時間，以毫秒為單位。在區段層級，所有層中區段的總CPU時間。在查詢層級，查詢所有層和區段CPU的時間總和。
max_block_s_read	bigint	區段讀取的 1 MB 區塊最大數量 (彙總所有分割)。在區段層級，所有分割的區段所讀取的 1 MB 區塊最大數量。在查詢層級，任何查詢區段所讀取的 1 MB 區塊最大數量。
blocks_read	bigint	查詢或區段所讀取的 1 MB 區塊數。
max_run_time	bigint	區段的最大經過時間 (以微秒為單位)。在區段層級，所有分割之區段的最大執行時間。在查詢層級，任何查詢區段的最大執行時間。
run_time	bigint	所有分割合計的總執行時間。執行時間不包含等待時間。 在區段層級，所有分割合計的區段執行時間。在查詢層級，所有分割與區段的查詢執行時間總和。因為此值為總和，因此執行時間與查詢執行時間無關。
max_block_s_to_disk	bigint	用來寫入中繼結果之磁碟空間的數量上限，以 MB 區塊表示。在區段層級，所有分割之區段已使用的最大磁碟空間量。在查詢層級，任何查詢區段已使用的最大磁碟空間量。

欄名稱	資料類型	描述
blocks_to_disk	bigint	查詢或區段用來寫入中繼結果之磁碟空間的數量上限，以 MB 區塊表示。
step	integer	執行的查詢步驟。
max_query_scan_size	bigint	查詢所掃描的最大資料大小，以 MB 為單位。在區段層級，所有分割之區段所掃描的最大資料大小。在查詢層級，任何查詢區段所掃描的最大資料大小。
query_scan_size	bigint	查詢所掃描的資料大小，以 MB 為單位。
query_priority	integer	查詢的優先順序。可能的值為 -1、0、1、2、3 和 4，其中 -1 表示不支援查詢優先順序。
query_queue_time	bigint	查詢排入佇列的時間 (毫秒)。
service_class_name	character (64)	服務類別的名稱。

範例查詢

若要尋找具有高CPU時間（超過 1,000 秒）的查詢，請執行下列查詢。

```
Select query, cpu_time / 1000000 as cpu_seconds
from stl_query_metrics where segment = -1 and cpu_time > 1000000000
order by cpu_time;
```

```
query | cpu_seconds
-----+-----
25775 |          9540
```

若要找出具有傳回超過一百萬個資料列之巢狀迴圈連結的使用中查詢，請執行下列查詢。

```
select query, rows
from stl_query_metrics
where step_type = 15 and rows > 1000000
order by rows;
```

```
query | rows
-----+-----
25775 | 2621562702
```

若要尋找已執行超過 60 秒且已使用不到 10 秒CPU的作用中查詢，請執行下列查詢。

```
select query, run_time/1000000 as run_time_seconds
from stl_query_metrics
where segment = -1 and run_time > 60000000 and cpu_time < 10000000;
```

```
query | run_time_seconds
-----+-----
25775 | 114
```

STL_QUERYTEXT

擷取SQL命令的查詢文字。

查詢 STL_QUERYTEXT 檢視以擷取SQL為下列陳述式記錄的：

- SELECT, SELECT INTO
- INSERT, UPDATE, DELETE
- COPY
- UNLOAD
- 執行 VACUUM和 產生的查詢 ANALYZE
- CREATE TABLE AS (CTAS)

若要查詢指定期間內這些陳述式的活動，請加入 STL_QUERYTEXT 和 STL_QUERY 檢視。

Note

STL_QUERY 和 STL_QUERYTEXT 檢視僅包含查詢的相關資訊，不包含其他公用程式和DDL命令。如需 Amazon Redshift 執行之所有陳述式的清單和資訊，您也可以查詢 STL_DDLTEXT 和 STL_UTILITYTEXT 檢視。如需 Amazon Redshift 執行之所有陳述式的完整清單，您可以查詢 SVL_STATEMENTTEXT 檢視。

另請參閱 [STL_DDLTEXT](#)、[STL_UTILITYTEXT](#) 和 [SVL_STATEMENTTEXT](#)。

STL所有使用者都可看見 `_QUERYTEXT`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_QUERY_TEXT](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。正常情況下，工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。在某些內部事件之後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 PID。如需詳細資訊，請參閱 STL_RESTARTED_SESSIONS 。您可以使用此欄來聯結至 STL_ERROR 檢視。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\) 和換行符號 (\n) 等特殊字元。

範例查詢

您可以使用 `PGBACKEND_PID ()` 函數來擷取目前工作階段的資訊。例如，下列查詢會針對目前工作階段中完成的查詢傳回其查詢 ID 和一部分的查詢文字。

```
select query, substring(text,1,60)
from stl_querytext
where pid = pg_backend_pid()
```



```
order by query desc;

query | substring
-----+-----
28262 | select query, substring(text,1,80) from stl_querytext where
28252 | select query, substring(path,0,80) as path from stl_unload_l
28248 | copy category from 's3://dw-tickit/manifest/category/1030_ma
28247 | Count rows in target table
28245 | unload ('select * from category') to 's3://dw-tickit/manifes
28240 | select query, substring(text,1,40) from stl_querytext where
(6 rows)
```

重建已儲存的 SQL

若要重建SQL存放在 STL_text 欄中的 QUERYTEXT，請執行SELECT陳述式，以SQL從text欄中的 1 個或多個部分建立。在執行重建的之前SQL，請將任何（\n）特殊字元取代為新行。下列SELECT陳述式的結果是在 query_statement SQL欄位中重建的資料列。

```
SELECT query, LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END)
  WITHIN GROUP (ORDER BY sequence) as query_statement, COUNT(*) as row_count
FROM stl_querytext GROUP BY query ORDER BY query desc;
```

例如，下列查詢會選取 3 個資料欄。查詢本身超過 200 個字元，並部分存放在 STL_ 中 QUERYTEXT。

```
select
1 AS a0123456789012345678901234567890123456789012345678901234567890,
2 AS b0123456789012345678901234567890123456789012345678901234567890,
3 AS b012345678901234567890123456789012345678901234
FROM stl_querytext;
```

在此範例中，查詢會儲存在 STL_text 欄中的 2 個部分（列）中QUERYTEXT。

```
select query, sequence, text
from stl_querytext where query=pg_last_query_id() order by query desc, sequence limit
10;
```

```
query | sequence |
      text
```

```

-----+-----
+-----
 45 |          0 | select\n1 AS
a012345678901234567890123456789012345678901234567890,\n2 AS
b012345678901234567890123456789012345678901234567890,\n3 AS
b012345678901234567890123456789012345678901234
 45 |          1 | \nFROM stl_querytext;

```

若要重建SQL存放在 STL_ 中的 QUERYTEXT，請執行下列 SQL。

```

select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
  within group (order by sequence) AS text
from stl_querytext where query=pg_last_query_id();

```

若要在用戶端SQL中使用產生的重建，請將任何 (\n) 特殊字元取代為新的行。

```

          text
-----
select\n1 AS a0123456789012345678901234567890123456789012345678901234567890,
\n2 AS b0123456789012345678901234567890123456789012345678901234567890,\n3 AS
b012345678901234567890123456789012345678901234\nFROM stl_querytext;

```

STL_REPLACEMENTS

顯示日誌，其中記錄的 是使用 ACCEPTINVCHARS 選項取代無效 UTF-8 個字元的 [COPY](#) 命令。對於每個節點切片上需要至少更換一次的前 100 列，日誌項目會新增至 STL_REPLACEMENTS。

STL 所有使用者都可看見 _REPLACEMENTS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

STL_NESTLOOP 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用 SYS 監控檢視 [SYS_COPY_REPLACEMENTS](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	發生取代的節點配量號碼。
tbl	integer	表格 ID。
starttime	timestamp	COPY 命令的開始時間UTC為。
session	integer	執行 COPY命令之工作階段的工作階段 ID。
filename	character (256)	完成COPY命令輸入檔案的路徑。
line_number	bigint	輸入資料檔案中包含無效 UTF-8 字元的行號。-1 表示行號不可用，例如從單欄式資料檔案複製時。
colname	character (127)	第一個包含無效 UTF-8 字元的欄位。
raw_line	character (1024)	原始載入包含無效 UTF-8 字元的資料。

範例查詢

下列範例會傳回最近COPY操作的替換。

```
select query, session, filename, line_number, colname
from stl_replacements
where query = pg_last_copy_id();
```

```
query | session | filename | line_number | colname
-----+-----+-----+-----+-----
  96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt |          251 | city
  96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt |          317 | city
  96 |    6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt |          569 | city
```

```

96 |      6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt |      623 | city
96 |      6314 | s3://DOC-EXAMPLE-BUCKET/allusers_pipe.txt |      694 | city
...

```

STL_RESTARTED_SESSIONS

若要在特定內部事件之後維持持續可用性，Amazon Redshift 可能會使用新的程序 ID () 重新啟動作用中工作階段PID。當 Amazon Redshift 重新啟動工作階段時，STL_RESTARTED_SESSIONS 會記錄新的 PID 和舊的 PID。

如需詳細資訊，請參閱本節中的下列範例。

STL所有使用者都可看見 _RESTARTED_SESSIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_SESSION_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
currenttime	timestamp	事件的時間。
dbname	character (50)	與工作階段相關聯之資料庫的名稱。
newpid	integer	已重新啟動之工作階段的處理程序 ID。
oldpid	integer	原始工作階段的處理程序 ID。
使用者名稱	character (50)	與工作階段相關聯之使用者的名稱。
remotehost	character (45)	遠端主機的名稱或 IP 地址。
remoteport	character (32)	遠端主機的連接埠號碼。
parkedtime	timestamp	此資訊僅供內部使用。

欄名稱	資料類型	描述
session_vars	character (2000)	此資訊僅供內部使用。

範例查詢

下列範例會加入 STL_RESTARTED_SESSIONS 與 STL_SESSIONS，以顯示已重新啟動工作階段的使用者名稱。

```
select process, stl_restarted_sessions.newpid, user_name
from stl_sessions
inner join stl_restarted_sessions on stl_sessions.process =
  stl_restarted_sessions.oldpid
order by process;

...
```

STL_RETURN

包含查詢中傳回步驟的詳細資訊。傳回步驟會將運算節點上完成之查詢的結果傳回至領導者節點。然後，領導者節點會合併資料，並將結果傳回至提出請求的用戶端。對於領導者節點上完成的查詢，傳回步驟會將結果傳回至用戶端。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱[查詢處理](#)。

STL所有使用者都可看見 _RETURN。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_RETURN 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
packets	integer	透過網路傳送的封包總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例顯示最新查詢中的哪些步驟已在每個配量上執行。

```
SELECT query, slice, segment, step, endtime, rows, packets
from stl_return where query = pg_last_query_id();
```

```

query | slice | segment | step |          endtime          | rows | packets
-----+-----+-----+-----+-----+-----+-----
      4 |      2 |        3 |     2 | 2013-12-27 01:43:21.469043 |     3 |        0
      4 |      3 |        3 |     2 | 2013-12-27 01:43:21.473321 |     0 |        0

```

```

 4 |      0 |      3 |      2 | 2013-12-27 01:43:21.469118 |      2 |      0
 4 |      1 |      3 |      2 | 2013-12-27 01:43:21.474196 |      0 |      0
 4 |      4 |      3 |      2 | 2013-12-27 01:43:21.47704  |      2 |      0
 4 |      5 |      3 |      2 | 2013-12-27 01:43:21.478593 |      0 |      0
 4 | 12811 |      4 |      1 | 2013-12-27 01:43:21.480755 |      0 |      0
(7 rows)

```

STL_S3CLIENT

記錄傳輸時間和其他效能指標。

使用 STL_S3CLIENT 資料表尋找從 Amazon S3 傳輸資料所花費的時間。

STL 所有使用者都可看見 _S3CLIENT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
recordtime	timestamp	記錄被記錄的時間。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。
http_method	character (64)	HTTP 對應至 Amazon S3 請求的方法名稱。
儲存貯體	character (64)	S3 儲存貯體名稱。
金鑰	character (256)	對應至 Amazon S3 物件的索引鍵。

欄名稱	資料類型	描述
transfer_size	bigint	已傳輸的位元組數目。
data_size	bigint	資料的位元組數目。此值與未壓縮資料的 transfer_size 相同。如果已使用壓縮，則這是未壓縮資料的大小。
start_time	bigint	自 2000 年 1 月 1 日起，傳輸開始的時間 (以微秒為單位)。
end_time	bigint	自 2000 年 1 月 1 日起，傳輸結束的時間 (以微秒為單位)。
transfer_time	bigint	傳輸所花費的時間 (以微秒為單位)。
compression_time	bigint	傳輸時間中未壓縮資料所花費的部分 (以微秒為單位)。
connect_time	bigint	從開始直到完成連線至遠端伺服器的時間 (以微秒為單位)。
app_connect_time	bigint	從開始到與遠端主機的SSL連線/交握完成為止的時間 (以毫秒為單位)。
retries	bigint	已嘗試傳輸的次數。
request_id	char(32)	從 Amazon S3 HTTP回應標頭請求 ID
extended_request_id	char(128)	Amazon S3 HTTP標頭回應 (x-amz-id-2) 的延伸請求 ID。
ip_address	char(64)	伺服器的 IP 地址 (ip V4 或 V6)。
is_partial	integer	如果 true (1) 表示輸入檔案在COPY操作期間分割為範圍的值。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果在COPY操作期間分割輸入檔案的值，會指出分割的偏移值 (以位元組為單位)。如果檔案未分割，則此值為 0。

範例查詢

下列查詢會傳回使用 COPY命令載入檔案所需的時間。

```
select slice, key, transfer_time
from stl_s3client
```



```
where query = pg_last_copy_id());
```

結果

slice	key	transfer_time
0	listing10M0003_part_00	16626716
1	listing10M0001_part_00	12894494
2	listing10M0002_part_00	14320978
3	listing10M0000_part_00	11293439
3371	prefix=listing10M;marker=	99395

下列查詢會將 `start_time` 和 `end_time` 轉換為時間戳記。

```
select userid,query,slice,pid,recordtime,start_time,end_time,
'2000-01-01'::timestamp + (start_time/1000000.0)* interval '1 second' as start_ts,
'2000-01-01'::timestamp + (end_time/1000000.0)* interval '1 second' as end_ts
from stl_s3client where query> -1 limit 5;
```

userid	query	slice	pid	recordtime	start_time	end_time	start_ts	end_ts
0	0	0	23449	2019-07-14 16:27:17.207839	616436837154256	616436837207838	2019-07-14 16:27:17.154256	2019-07-14 16:27:17.207838
0	0	0	23449	2019-07-14 16:27:17.252521	616436837208208	616436837252520	2019-07-14 16:27:17.208208	2019-07-14 16:27:17.25252
0	0	0	23449	2019-07-14 16:27:17.284376	616436837208460	616436837284374	2019-07-14 16:27:17.20846	2019-07-14 16:27:17.284374
0	0	0	23449	2019-07-14 16:27:17.285307	616436837208980	616436837285306	2019-07-14 16:27:17.20898	2019-07-14 16:27:17.285306
0	0	0	23449	2019-07-14 16:27:17.353853	616436837302216	616436837353851	2019-07-14 16:27:17.302216	2019-07-14 16:27:17.353851

STL_S3CLIENT_ERROR

記錄從 Amazon S3 載入檔案時配量遇到的錯誤。

使用 STL_S3CLIENT_ERROR 尋找從 Amazon S3 傳輸資料作為 COPY 命令的一部分時遇到的錯誤詳細資訊。

STL 所有使用者都可看見 _S3CLIENT_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。查詢 ID -1 供內部使用。
sliceid	integer	識別執行查詢之配量的數字。
recordtime	timestamp	記錄被記錄的時間。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。
http_method	character (64)	HTTP 對應至 Amazon S3 請求的方法名稱。
儲存貯體	character (64)	Amazon S3 儲存貯體名稱。
金鑰	character (256)	對應至 Amazon S3 物件的索引鍵。
error	character (1024)	錯誤訊息。
is_partial	integer	如果為 true (1)，表示輸入檔案在 COPY 操作期間分割為範圍的值。如果此值為 false (0)，則輸入檔案不會分割。
start_offset	bigint	如果在 COPY 操作期間分割輸入檔案的值，會指出分割的偏移值 (以位元組為單位)。如果檔案未分割，則此值為 0。

使用須知

如果看到多個錯誤具有「連線逾時」，則可能發生聯網問題。如果您使用的是增強型VPC路由，請確認叢集VPC和資料資源之間具有有效的網路路徑。如需詳細資訊，請參閱 [Amazon Redshift 增強型VPC路由](#)。

範例查詢

下列查詢會從目前工作階段期間完成的COPY命令傳回錯誤。

```
select query, sliceid, substring(key from 1 for 20) as file,
substring(error from 1 for 35) as error
from stl_s3client_error
where pid = pg_backend_pid()
order by query desc;
```

結果

query	sliceid	file	error
362228	12	part.tbl.25.159.gz	transfer closed with 1947655 bytes
362228	24	part.tbl.15.577.gz	transfer closed with 1881910 bytes
362228	7	part.tbl.22.600.gz	transfer closed with 700143 bytes r
362228	22	part.tbl.3.34.gz	transfer closed with 2334528 bytes
362228	11	part.tbl.30.274.gz	transfer closed with 699031 bytes r
362228	30	part.tbl.5.509.gz	Unknown SSL protocol error in conne
361999	10	part.tbl.23.305.gz	transfer closed with 698959 bytes r
361999	19	part.tbl.26.582.gz	transfer closed with 1881458 bytes
361999	4	part.tbl.15.629.gz	transfer closed with 2275907 bytes
361999	20	part.tbl.6.456.gz	transfer closed with 692162 bytes r

(10 rows)

STL_SAVE

包含查詢中儲存步驟的詳細資訊。儲存步驟會將輸入串流儲存至暫時性資料表。暫時性資料表是暫存資料表，其會在查詢執行期間儲存中繼結果。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱 [查詢處理](#)。

STL所有使用者都可看見 _SAVE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

STL_SAVE 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
tbl	integer	具體化的暫時性資料表的 ID。
is_diskbased	character(1)	查詢的這個步驟是否以磁碟型操作方式執行：true (t) 或 false (f)。
workmem	bigint	已指派給步驟之運作中記憶體之位元組數。

範例查詢

下列範例顯示最新查詢中的哪些儲存步驟已在每個配量上執行。

```
select query, slice, segment, step, tasknum, rows, tbl
from stl_save where query = pg_last_query_id();
```

```

query | slice | segment | step | tasknum | rows | tbl
-----+-----+-----+-----+-----+-----+-----
52236 |    3 |      0 |    2 |    21 |    0 | 239
52236 |    2 |      0 |    2 |    20 |    0 | 239
52236 |    2 |      2 |    2 |    20 |    0 | 239
52236 |    3 |      2 |    2 |    21 |    0 | 239
52236 |    1 |      0 |    2 |    21 |    0 | 239
52236 |    0 |      0 |    2 |    20 |    0 | 239
52236 |    0 |      2 |    2 |    20 |    0 | 239
52236 |    1 |      2 |    2 |    21 |    0 | 239
(8 rows)
```

STL_SCAN

分析查詢的資料表掃描步驟。此資料表中資料列的步驟號碼一律為 0，因為掃描是區段中的第一個步驟。

STL 所有使用者都可看見 `_SCAN`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_SCAN 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。

欄名稱	資料類型	描述
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
fetches	bigint	此資訊僅供內部使用。
type	integer	掃描類型的 ID。如需有效值的清單，請參閱下列資料表。
tbl	integer	表格 ID。
is_rrscan	character(1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。
is_delayed_scan	character(1)	此資訊僅供內部使用。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之後，但在套用使用者定義的查詢篩選條件之前已發出的資料列總數。
rows_pre_user_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之後，但在套用使用者定義的查詢篩選條件之前已處理的資料列數目。

欄名稱	資料類型	描述
perm_table_name	character(136)	對於永久資料表的掃描，指的是已掃描的資料表名稱。
is_rlf_scan	character(1)	若為 true (t)，表示已在步驟上使用資料列層級篩選。
is_rlf_scan_reason	integer	此資訊僅供內部使用。
num_em_blocks	integer	此資訊僅供內部使用。
checksum	bigint	此資訊僅供內部使用。
runtime_filtering	character(1)	如果為 true (t)，則表示會套用執行時間篩選條件。
scan_region	integer	此資訊僅供內部使用。
num_sortkey_as_predicate	integer	此資訊僅供內部使用。
row_fetcher_state	integer	此資訊僅供內部使用。
consumed_scan_ranges	bigint	此資訊僅供內部使用。
work_stealing_reason	bigint	此資訊僅供內部使用。
is_vectorized_scan	character(1)	此資訊僅供內部使用。

欄名稱	資料類型	描述
is_vectorized_scan_reason	integer	此資訊僅供內部使用。
row_fetcher_reason	bigint	此資訊僅供內部使用。
topology_signature	bigint	此資訊僅供內部使用。
use_tpm_partition	character(1)	此資訊僅供內部使用。
is_rrscan_expr	character(1)	此資訊僅供內部使用。
scanned_mega_value	character(1)	此資訊僅供內部使用。此資訊顯示指定的掃描步驟是否掃描了較大的值。大值將儲存在多個區塊中。區塊大小預設為 1 MB，在預設設定中，大值大於 1 MB。

掃描類型

類型 ID	描述
1	來自網路的資料。
2	已壓縮之共用記憶體中的永久使用者資料表。
3	臨時性全資料列資料表。
21	從 Amazon S3 載入檔案。
22	從 Amazon DynamoDB 載入資料表。
23	從遠端SSH連線載入資料。
24	從遠端叢集載入資料 (已排序區域)。這是用於調整大小。

類型 ID	描述
25	從遠端叢集載入資料 (未排序區域)。這是用於調整大小。
28	在多個資料表UNIONALL上使用 從時間序列檢視讀取資料。
29	從 Amazon S3 外部資料表讀取資料。
30	讀取 Amazon S3 外部資料表的分割區資訊。
33	從遠端 Postgres 資料表讀取資料。
36	從遠端我的SQL資料表讀取資料。
37	從遠端 Kinesis 串流讀取資料。

使用須知

在理想情況下，rows 應該相當接近 rows_pre_filter。rows 與 rows_pre_filter 之間的差異若很大，表示執行引擎正在掃描稍後將捨棄的資料列，這樣做缺乏效率。rows_pre_filter 與 rows_pre_user_filter 之間的差異是掃描中幽靈資料列的數目。執行 VACUUM 以移除標記為刪除的資料列。rows 與 rows_pre_user_filter 之間的差異是查詢所篩選的資料列數目。若有許多資料列已遭使用者篩選條件捨棄，請檢閱您選擇的排序資料欄，或若這是由於大型未排序區域所致，請執行清空。

範例查詢

下列範例顯示 rows_pre_filter 大於 rows_pre_user_filter，因為資料表已刪除未清空的資料列 (幽靈資料列)。

```
SELECT query, slice, segment, step, rows, rows_pre_filter, rows_pre_user_filter
from stl_scan where query = pg_last_query_id();
```

query	slice	segment	step	rows	rows_pre_filter	rows_pre_user_filter
42915	0	0	0	43159	86318	43159
42915	0	1	0	1	0	0
42915	1	0	0	43091	86182	43091
42915	1	1	0	1	0	0
42915	2	0	0	42778	85556	42778
42915	2	1	0	1	0	0

```

42915 |      3 |      0 |      0 | 43428 |      86856 |      43428
42915 |      3 |      1 |      0 |      1 |           0 |           0
42915 | 10000 |      2 |      0 |      4 |           0 |           0
(9 rows)

```

STL_SCHEMA_QUOTA_VIOLATIONS

記錄超過結構描述配額時的發生率、時間戳記XID、和其他實用資訊。

STL所有使用者都可看見 `_SCHEMA_QUOTA_VIOLATIONS`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_SCHEMA_QUOTA_VIOLATIONS](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	Description
ownerid	integer	結構描述擁有者的 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
userid	integer	產生項目的使用者之 ID。
schema_id	integer	命名空間或結構描述 ID。
schema_name	character (128)	命名空間或結構描述名稱。
配額	integer	結構描述可以使用的磁碟空間量 (以 MB 為單位)。
disk_usage	integer	結構描述目前使用的磁碟空間 (以 MB 為單位)。
disk_usage_pct	double precision	結構描述目前從所設定配額使用的磁碟空間百分比。

欄名稱	資料類型	Description
timestamp	沒有時區的 時間戳記	發生違規的時間。

範例查詢

下列查詢顯示配額違規的結果：

```
SELECT userid, TRIM(SCHEMA_NAME) "schema_name", quota, disk_usage, disk_usage_pct,
timestamp FROM
stl_schema_quota_violations WHERE SCHEMA_NAME = 'sales_schema' ORDER BY timestamp DESC;
```

此查詢會傳回所指定結構描述的下列範例輸出：

```
userid | schema_name | quota | disk_usage | disk_usage_pct | timestamp
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
104    | sales_schema | 2048 | 2798      | 136.62         | 2020-04-20
      20:09:25.494723
(1 row)
```

STL_SESSIONS

傳回使用者工作階段歷史記錄的相關資訊。

STL_SESSIONS 與 STV_SESSIONS 不同之處在於 STL_SESSIONS 包含工作階段歷史記錄，其中 STV_SESSIONS 包含目前的作用中工作階段。

STL 所有使用者都可看見 _SESSIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_SESSION_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
starttime	timestamp	工作階段開始UTC的時間。
endtime	timestamp	工作階段結束UTC的時間。
process	integer	工作階段的處理程序 ID。
user_name	character(50)	與工作階段相關聯的使用者名稱。
db_name	character(50)	與工作階段相關聯之資料庫的名稱。
timeout_sec	int	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。
timed_out	int	指出連線終止原因的值。它可以具有下列值： <ul style="list-style-type: none"> • 0：連線因未知錯誤而終止。 • 1：連線逾時。 • 2：用戶端終止連線。 • 3：Amazon Redshift 後端內部錯誤終止了連線。

範例查詢

若要檢視TICKIT資料庫的工作階段歷史記錄，請輸入下列查詢：

```
select starttime, process, user_name, timeout_sec, timed_out
from stl_sessions
where db_name='tickit' order by starttime;
```

此查詢傳回下列範例輸出：

```
starttime          | process | user_name          | timeout_sec | timed_out
```

```

-----+-----+-----+-----
+-----
2008-09-15 09:54:06.746705 | 32358 | dwuser | 120 | 1
2008-09-15 09:56:34.30275 | 32744 | dwuser | 60 | 1
2008-09-15 11:20:34.694837 | 14906 | dwuser | 0 | 0
2008-09-15 11:22:16.749818 | 15148 | dwuser | 0 | 0
2008-09-15 14:32:44.66112 | 14031 | dwuser | 0 | 0
2008-09-15 14:56:30.22161 | 18380 | dwuser | 0 | 0
2008-09-15 15:28:32.509354 | 24344 | dwuser | 0 | 0
2008-09-15 16:01:00.557326 | 30153 | dwuser | 120 | 1
2008-09-15 17:28:21.419858 | 12805 | dwuser | 0 | 0
2008-09-15 20:58:37.601937 | 14951 | dwuser | 60 | 1
2008-09-16 11:12:30.960564 | 27437 | dwuser | 60 | 1
2008-09-16 14:11:37.639092 | 23790 | dwuser | 3600 | 1
2008-09-16 15:13:46.02195 | 1355 | dwuser | 120 | 1
2008-09-16 15:22:36.515106 | 2878 | dwuser | 120 | 1
2008-09-16 15:44:39.194579 | 6470 | dwuser | 120 | 1
2008-09-16 16:50:27.02138 | 17254 | dwuser | 120 | 1
2008-09-17 12:05:02.157208 | 8439 | dwuser | 3600 | 0
(17 rows)

```

STL_SORT

顯示查詢的排序執行步驟，例如使用 ORDER BY 處理的步驟。

STL 所有使用者都可看見 _SORT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_SORT 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
位元組	bigint	步驟的所有輸出欄之大小 (位元組)。
tbl	integer	表格 ID。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體中的位元組總數。
checksum	bigint	此資訊僅供內部使用。

範例查詢

下列範例傳回配量 0 和區段 1 的排序結果。

```
select query, bytes, tbl, is_diskbased, workmem
from stl_sort
where slice=0 and segment=1;
```

```

query | bytes | tbl | is_diskbased | workmem
-----+-----+-----+-----+-----
 567 | 3126968 | 241 | f | 383385600
 604 | 5292 | 242 | f | 383385600
 675 | 104776 | 251 | f | 383385600
 525 | 3126968 | 251 | f | 383385600
 585 | 5068 | 241 | f | 383385600
 630 | 204808 | 266 | f | 383385600
 704 | 0 | 242 | f | 0
 669 | 4606416 | 241 | f | 383385600
 696 | 104776 | 241 | f | 383385600
 651 | 4606416 | 254 | f | 383385600
 632 | 0 | 256 | f | 0
 599 | 396 | 241 | f | 383385600
86397 | 0 | 242 | f | 0
 621 | 5292 | 241 | f | 383385600
86325 | 0 | 242 | f | 0
 572 | 5068 | 242 | f | 383385600
 645 | 204808 | 241 | f | 383385600
 590 | 396 | 242 | f | 383385600
(18 rows)

```

STL_SSHCLIENT_ERROR

記錄SSH用戶端看到的所有錯誤。

STL所有使用者都可看見 `_SSHCLIENT_ERROR`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
recordtime	timestamp	記錄錯誤的時間。
pid	integer	記錄錯誤的處理程序。

欄名稱	資料類型	描述
ssh_username	character (1024)	SSH 使用者名稱。
端點	character (1024)	SSH 端點。
command	character (4096)	完整的SSH命令。
error	character (1024)	錯誤訊息。

STL_STREAM_SEGS

列出串流與並行區段之間的關係。

此內容中的串流是 Amazon Redshift 串流。此系統檢視不屬於 [將擷取串流至具體化視觀表](#)。

STL所有使用者都可看見 _STREAM_SEGS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_STREAM_SEGS 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
串流	integer	查詢的並行區段集。

欄名稱	資料類型	描述
segment	integer	識別查詢區段的號碼。

範例查詢

若要針對最新查詢檢視串流與並行區段之間的關係，請輸入下列查詢：

```
select *
from stl_stream_segs
where query = pg_last_query_id();
```

```

query | stream | segment
-----+-----+-----
    10 |      1 |      2
    10 |      0 |      0
    10 |      2 |      4
    10 |      1 |      3
    10 |      0 |      1
(5 rows)
```

STL_TR_CONFLICT

顯示資訊以識別並解決交易與資料庫資料表的衝突。

當兩個以上使用者正在查詢和修改資料表中的資料列，以致其交易無法序列化時，即會發生交易衝突。交易若執行將中斷序列化的陳述式，其會遭到停止並進行復原。每次發生交易衝突時，Amazon Redshift 都會將資料列寫入 STL_TR_CONFLICT 系統資料表，其中包含已取消交易的詳細資訊。如需詳細資訊，請參閱[可序列化隔離](#)。

STL_TR_CONFLICT 僅供超級使用者檢視。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_TRANSACTION_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
xact_id	bigint	已復原之交易的交易 ID。
process_id	bigint	與已復原之交易相關聯的處理程序。
xact_start_ts	timestamp	交易開始時的時間戳記 (UTC)。
abort_time	timestamp	交易停止時的時間戳記 (UTC)。
table_id	bigint	發生衝突之資料表的資料表 ID。

範例查詢

若要傳回涉及特定資料表之衝突的相關資訊，請執行一個指定資料表 ID 的查詢：

```
select * from stl_tr_conflict where table_id=100234
order by xact_start_ts;
```

```
xact_id|process_|      xact_start_ts      |      abort_time      |table_
      |id      |      |      |      |      |
-----+-----+-----+-----+-----+
 1876 |  8551 | 2010-03-30 09:19:15.852326| 2010-03-30 09:20:17.582499|100234
 1928 | 15034 | 2010-03-30 13:20:00.636045| 2010-03-30 13:20:47.766817|100234
 1991 | 23753 | 2010-04-01 13:05:01.220059| 2010-04-01 13:06:06.94098  |100234
 2002 | 23679 | 2010-04-01 13:17:05.173473| 2010-04-01 13:18:27.898655|100234
(4 rows)
```

您可以從 錯誤訊息的 `DETAIL` 區段取得資料表 ID，以解決序列化違規 (錯誤 1023)。

STL_UNDONE

顯示已復原之交易的相關資訊。

STL 所有使用者都可看見 `_UNDONE`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在 `SYS` 監控檢視 中找到 [SYS_TRANSACTION_HISTORY](#)。`SYS` 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用 `SYS` 監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xact_id	bigint	復原交易的 ID。
xact_id_undone	bigint	已復原之交易的 ID。
undo_start_ts	timestamp	復原交易的開始時間。
undo_end_ts	timestamp	復原交易的結束時間。
table_id	bigint	受到復原交易影響之資料表的 ID。

範例查詢

若要檢視所有已復原之交易的精簡日誌，請輸入下列命令：

```
select xact_id, xact_id_undone, table_id from stl_undone;
```

此命令會傳回下列範例輸出：

```
xact_id | xact_id_undone | table_id
-----+-----+-----
1344 | 1344 | 100192
1326 | 1326 | 100192
1551 | 1551 | 100192
(3 rows)
```

STL_UNIQUE

在SELECT清單中使用DISTINCT函數或在 UNIONINTERSECT查詢中移除重複項目時，分析發生的執行步驟。

STL所有使用者都可看見 `_UNIQUE`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_UNIQUE 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
type	character(6)	步驟的類型。有效的 值如下： <ul style="list-style-type: none"> HASHED。表示步驟已使用分組且未排序的彙整。 PLAIN。表示步驟已使用未分組的純量彙整。

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> • SORTED。表示步驟已使用分組且排序的彙整。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
slots	integer	雜湊儲存貯體的總數。
workmem	bigint	已指派給步驟之運作中記憶體中的位元組總數。
max_buffers_used	bigint	在移至磁碟之前於雜湊表中使用的緩衝區數目上限。
resizes	integer	此資訊僅供內部使用。
occupied	integer	此資訊僅供內部使用。
flushable	integer	此資訊僅供內部使用。
used_unique_prefetching	character(1)	此資訊僅供內部使用。
位元組	bigint	步驟之所有輸出列的位元組數目。

範例查詢

假設您執行下列查詢：

```
select distinct eventname
from event order by 1;
```

假設前一個查詢的 ID 為 6313，下列範例會顯示唯一步驟針對區段 0 與 1 中之每個配量所產生的資料列數目。

```
select query, slice, segment, step, datediff(msec, starttime, endtime) as msec,
       tasknum, rows
from stl_unique where query = 6313
order by query desc, slice, segment, step;
```

```

query | slice | segment | step | msec | tasknum | rows
-----+-----+-----+-----+-----+-----+-----
6313 | 0 | 0 | 2 | 0 | 22 | 550
6313 | 0 | 1 | 1 | 256 | 20 | 145
6313 | 1 | 0 | 2 | 1 | 23 | 540
6313 | 1 | 1 | 1 | 42 | 21 | 127
6313 | 2 | 0 | 2 | 1 | 22 | 540
6313 | 2 | 1 | 1 | 255 | 20 | 158
6313 | 3 | 0 | 2 | 1 | 23 | 542
6313 | 3 | 1 | 1 | 38 | 21 | 146
(8 rows)

```

STL_UNLOAD_LOG

記錄卸載操作的詳細資訊。

STL_UNLOAD_LOG 會為UNLOAD陳述式建立的每個檔案記錄一列。例如，如果 UNLOAD 建立 12 個檔案，STL_UNLOAD_LOG 將包含 12 個對應的資料列。

STL所有使用者都可看見 _UNLOAD_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_UNLOAD_LOG 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視[SYS_UNLOAD_HISTORY](#)和[SYS_UNLOAD_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。
分割	integer	識別執行查詢之配量的數字。

欄名稱	資料類型	描述
pid	integer	與查詢陳述式相關聯的處理程序 ID。
路徑	character(1280)	檔案的完整 Amazon S3 物件路徑。
start_time	timestamp	交易的開始時間。
end_time	timestamp	交易的結束時間。
line_count	bigint	已卸載至檔案的行數 (列數)。
transfer_size	bigint	已傳輸的位元組數目。
file_format	character(10)	已卸載檔案的格式。

範例查詢

若要取得透過 UNLOAD 命令寫入 Amazon S3 的檔案清單，您可以在 UNLOAD 完成後呼叫 Amazon S3 清單操作。您也可以查詢 STL_UNLOAD_LOG。

下列查詢會傳回 為上次完成的查詢所建立檔案 UNLOAD 的路徑名稱：

```
select query, substring(path,0,40) as path
from stl_unload_log
where query = pg_last_query_id()
order by path;
```

此命令會傳回下列範例輸出：

```
query |          path
-----+-----
 2320 | s3://amzn-s3-demo-bucket/venue0000_part_00
 2320 | s3://amzn-s3-demo-bucket/venue0001_part_00
 2320 | s3://amzn-s3-demo-bucket/venue0002_part_00
 2320 | s3://amzn-s3-demo-bucket/venue0003_part_00
(4 rows)
```

STL_USAGE_CONTROL

STL_USAGE_CONTROL 檢視包含達到用量限制時記錄的資訊。如需使用限制的相關資訊，請參閱《Amazon Redshift 管理指南》中的[管理用量限制](#)。

STL只有超級使用者才能看見 _USAGE_CONTROL。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
eventtime	timestamp	查詢超過用量限制的時間（UTC）。
query	integer	查詢識別符。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
xid	bigint	交易識別符。
pid	integer	與查詢相關聯的處理程序識別碼。
usage_lim it_id	character(40)	Amazon Redshift 產生的通用唯一識別碼（UUID），例如 25d9297e-3e7b-41c8-9f4d-c4b6eb731c09。
feature_t ype	character(30)	超過用量限制的功能。可能的值包括 CONCURRENCY_SCALING 和 SPECTRUM。

範例查詢

下列SQL範例會傳回達到用量限制時記錄的一些資訊。

```
select query, pid, eventtime, feature_type
from stl_usage_control
order by eventtime desc
limit 5;
```

STL_USERLOG

記錄資料庫使用者之下列變更的詳細資訊：

- 建立使用者

- 捨棄使用者
- 更改使用者 (重新命名)
- 更改使用者 (更改屬性)

STL只有超級使用者可以看到 `_USERLOG`。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_USERLOG](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	受到變更影響之使用者的 ID。
使用者名稱	character(50)	受到變更影響之使用者的使用者名稱。
oldusername	character(50)	若為重新命名動作，這是原始使用者名稱。若是任何其他動作，此欄位空白。
動作	character(10)	發生的動作。有效值： <ul style="list-style-type: none"> • Alter • 建立 • Drop • 重新命名
usecreatedb	integer	若為 true (1)，表示使用者具備建立資料庫權限。
usesuper	integer	若為 true (1)，表示使用者是超級使用者。
usecatupd	integer	若為 true (1)，表示使用者可以更新系統目錄。
valuntil	timestamp	密碼到期日。
pid	integer	處理程序 ID。

欄名稱	資料類型	描述
xid	bigint	交易 ID。
recordtime	timestamp	UTC 查詢開始的時間。

範例查詢

下列範例會執行四個使用者動作，然後查詢 STL_USERLOG 檢視。

```
create user userlog1 password 'Userlog1';
alter user userlog1 createdb createuser;
alter user userlog1 rename to userlog2;
drop user userlog2;

select userid, username, oldusername, action, usecreatedb, usesuper from stl_userlog
order by recordtime desc;
```

```
userid | username | oldusername | action | usecreatedb | usesuper
-----+-----+-----+-----+-----+-----
  108 | userlog2 |             | drop   |             | 1
  108 | userlog2 | userlog1    | rename |             | 1
  108 | userlog1 |             | alter  |             | 1
  108 | userlog1 |             | create |             | 0
(4 rows)
```

STL_UTILITYTEXT

擷取資料庫上執行的非SELECTSQL命令文字。

查詢 STL_UTILITYTEXT 檢視以擷取下列在系統上執行的SQL陳述式子集：

- ABORT, BEGIN, COMMIT, END, ROLLBACK
- ANALYZE
- CALL

- CANCEL
- COMMENT
- CREATE, ALTER, DROP DATABASE
- CREATE, ALTER, DROP USER
- EXPLAIN
- GRANT, REVOKE
- LOCK
- RESET
- SET
- SHOW
- TRUNCATE

另請參閱 [STL_DDLTEXT](#)、[STL_QUERYTEXT](#) 和 [SVL_STATEMENTTEXT](#)。

使用 STARTTIME 和 ENDTIME 資料欄，找出在指定期間內記錄的陳述式。長 SQL 文字區塊會細分為 200 個字元長的行；該 SEQUENCE 欄會識別屬於單一陳述式的文字片段。

STL 所有使用者都可看見 _UTILITYTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在 SYS 監控檢視 中找到 [SYS_QUERY_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用 SYS 監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	交易 ID。
pid	integer	與查詢陳述式相關聯的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或使用 QUERY_SET GROUP 命令定義的標籤。如果查詢不是以檔案為基礎，或未設定 QUERY_GROUP 參數，則此欄位為空白。

欄名稱	資料類型	描述
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回已在 2012 年 1 月 26 日執行之「公用程式」命令的文字。在此情況下，執行了某些 SET 命令和 SHOWALL 命令：

```
select starttime, sequence, rtrim(text)
from stl_utilitytext
where starttime like '2012-01-26%'
order by starttime, sequence;
```

starttime	sequence	rtrim
2012-01-26 13:05:52.529235	0	show all;
2012-01-26 13:20:31.660255	0	SET query_group to ''
2012-01-26 13:20:54.956131	0	SET query_group to 'soldunsold.sql'
...		

重建已儲存 SQL

若要重建 SQL 存放在 STL_text 欄中的 UTILITYTEXT，請執行 SELECT 陳述式，SQL 從 text 欄中的 1 個或多個部分建立。在執行重建的之前 SQL，請將任何 (\n) 特殊字元取代為新行。下列 SELECT 陳述式的結果是在 query_statement SQL 欄位中重建的資料列。

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> • Started Sort Only • Started Ranged Partition • Started Reindex • Finished <p>已對資料表完成操作的時間。若要了解清空作業已對特定資料表花費了多長時間，請從特定交易 ID 和資料表 ID 的完成時間中減去啟動時間。</p> <ul style="list-style-type: none"> • Skipped <p>已略過資料表，因為已完全排序資料表，而且未標示任何資料列進行刪除。</p> <ul style="list-style-type: none"> • Skipped (delete only) <p>資料表已略過，因為 DELETEONLY 已指定，而且沒有資料列標記為刪除。</p> <ul style="list-style-type: none"> • Skipped (sort only) <p>資料表已略過，因為 SORT ONLY 已指定，且資料表已完全排序。</p> <ul style="list-style-type: none"> • Skipped (sort only, sorted>=xx%) <p>資料表已略過，因為 SORT ONLY 已指定，且資料表已按排序閾值或高於排序閾值排序。</p> <ul style="list-style-type: none"> • Skipped (0 rows) <p>已略過資料表，因為它是空的。</p> <ul style="list-style-type: none"> • VacuumBG <p>自動清空操作會在背景執行。當其他狀態自動執行時，此狀態會附加在其他狀態之前。例如，自動執行的僅刪除清空將具有狀態 [VacuumBG] Started Delete Only 的起始列。</p>

欄名稱	資料類型	描述
		如需VACUUM排序閾值設定的詳細資訊，請參閱 VACUUM 。
rows	bigint	資料表中的實際資料列數目加上任何仍在磁碟上儲存的已刪除資料列 (等待清空)。此欄顯示在對狀態為 Started 的資料列啟動清空之前的計數，以及在對狀態為 Finished 的資料列啟動清空之後的計數。
sortedrows	integer	資料表中已排序的資料列數目。此欄顯示在對 Status 欄中具有 Started 的資料列啟動清空之前的計數，以及在對 Status 欄中具有 Finished 的資料列啟動清空之後的計數。
blocks	integer	在清空操作 (狀態為 Started 的資料列) 之前，以及在清空操作 (Finished 欄) 之後，用來儲存資料表資料的資料區塊總數。每個資料區塊都使用 1 MB。
max_merge_partitions	integer	會使用此欄位來分析表現並表示分割區數上限，也就是清空可以根據合併階段反覆運算為資料表處理的分割區數上限。(清空會將未排序的區域排序為一個或多個已排序的分割區。根據資料表中的欄數和目前的 Amazon Redshift 組態，合併階段可以在單次合併迭代中處理最大數量的分割區。如果排序分割區的數量超過合併分割區的最大數量，合併階段仍然有效，但需要更多的合併迭代。)
eventtime	timestamp	清空操作啟動或完成的時間。
reclaimable_rows	bigint	目前 cutoff_xid 的可回收列數。此欄顯示 Redshift 在清空開始之前對於具有 Started 狀態之列的估計可回收列數，以及在清空之後對於具有 Finished 狀態之列剩餘的實際可回收列數。
reclaimable_space_mb	bigint	目前 cutoff_xid 的可回收空間 (以 MB 為單位)。此欄顯示 Redshift 在清空開始之前對於具有 Started 狀態之列的估計可回收空間量，以及在清空之後對於具有 Finished 狀態之列剩餘的實際可回收空間量。

欄名稱	資料類型	描述
cutoff_xid	bigint	VACUUM 操作的截止交易 ID。截止後的任何交易都不會包含在VACUUM操作中。
is_recluster	integer	如果為 1 (true) , VACUUM操作會執行叢集演算法 , 如果為 0 (false) , 則不是。

範例查詢

下列查詢報告資料表 108313 的清空統計資訊。在一系列插入和刪除之後已清空此資料表。

```
select xid, table_id, status, rows, sortedrows, blocks, eventtime,
       reclaimable_rows, reclaimable_space_mb
from stl_vacuum where table_id=108313 order by eventtime;
```

xid	table_id	status	rows	sortedrows	blocks	eventtime	reclaimable_rows	reclaimable_space_mb
14294	108313	Started	1950	408	28	2016-05-19	984	17
14294	108313	Finished	966	966	11	2016-05-19	0	0
15126	108313	Skipped(sorted>=95%)	966	966	11	2016-05-19	0	0

在開始時VACUUM，資料表包含存放在 28 個 1 MB 區塊中的 1,950 列。Amazon Redshift 估計，透過清空操作可以回收 984 個磁碟空間，即 17 個磁碟空間區塊。

在完成狀態的列中，資料ROWS欄會顯示 966 的值，資料BLOCKS欄值為 11，從 28 下降。清空回收了估計的磁碟空間量，清空操作完成後沒有剩餘可回收的列或空間。

在排序階段 (交易 15126) 中，清空能夠略過資料表，因為已依排序索引鍵順序插入資料列。

下列範例顯示大型INSERT操作後SALES資料表（此範例中的資料表 110116）上的SORTONLY真空統計資料：

```
vacuum sort only sales;
```

```
select xid, table_id, status, rows, sortedrows, blocks, eventtime
from stl_vacuum order by xid, table_id, eventtime;
```

```
xid |table_id|      status      | rows |sortedrows|blocks|      eventtime
-----+-----+-----+-----+-----+-----+-----
...
2925| 110116 |Started Sort Only|1379648|  172456 |  132 | 2011-02-24 16:25:21...
2925| 110116 |Finished          |1379648| 1379648 |  132 | 2011-02-24 16:26:28...
```

STL_WINDOW

分析執行視窗函數的查詢步驟。

STL所有使用者都可看見 `_WINDOW`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STL_WINDOW 僅包含主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴展叢集上執行的查詢，建議您使用SYS監控檢視 [SYS_QUERY_DETAIL](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
分割	integer	識別執行查詢之配量的數字。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
starttime	timestamp	UTC 查詢開始的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

欄名稱	資料類型	描述
endtime	timestamp	UTC 查詢完成的時間。總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。
tasknum	integer	指派執行步驟的查詢任務程序編號。
rows	bigint	已處理的列總數。
is_diskbased	character(1)	若為 true (t)，查詢是以磁碟型操作方式執行。若為 false (f)，查詢是在記憶體中執行。
workmem	bigint	已指派給步驟之運作中記憶體中的位元組總數。

範例查詢

下列範例傳回配量 0 和區段 3 的視窗函數結果。

```
select query, tasknum, rows, is_diskbased, workmem
from stl_window
where slice=0 and segment=3;
```

```
query | tasknum | rows | is_diskbased | workmem
-----+-----+-----+-----+-----
86326 |      36 | 1857 | f             | 95256616
   705 |      15 | 1857 | f             | 95256616
86399 |      27 | 1857 | f             | 95256616
   649 |      10 |    0 | f             | 95256616
(4 rows)
```

STL_WLM_ERROR

記錄所有 WLM 相關的錯誤。

STL 所有使用者都可看見 `_WLM_ERROR`。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
recordtime	timestamp	發生錯誤的時間。
pid	integer	產生錯誤之處理程序的 ID。
error_string	character(256)	錯誤說明。

STL_WLM_RULE_ACTION

記錄因與使用者定義佇列相關聯的WLM查詢監控規則而產生的動作詳細資訊。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。

STL所有使用者都可看見 _WLM_RULE_ACTION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行查詢的使用者。
query	integer	查詢 ID。
service_class	integer	服務類別的 ID。查詢佇列在WLM組態中定義。大於 5 的服務類別是使用者定義的佇列。
rule	character(256)	查詢監控規則的名稱。
動作	character(256)	產生的動作。可能的值如下： <ul style="list-style-type: none"> log hop(reassign) hop(restart) abort

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> change_query_priority 無 <p>none 值表示已符合規則的述詞，但此動作已被另一個規則取代為嚴重性更高的動作。</p>
recordtime	timestamp	動作登入 的時間UTC。
action_value	character(256)	<p>若 action 為 change_query_priority ，則可能的值會是 highest、high、normal、low 和 lowest。</p> <p>若 action 為 log、hop 或 abort ，則此值為是空白。</p>
service_class_name	character(64)	服務類別的名稱。

範例查詢

下列範例尋找查詢監控規則已停止的查詢。

```
Select query, rule
from stl_wlm_rule_action
where action = 'abort'
order by query;
```

STL_WLM_QUERY

包含由 處理之服務類別中每次嘗試執行查詢的記錄WLM。

STL所有使用者都可看見 _WLM_QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此資料表中的部分或全部資料也可以在SYS監控檢視 中找到[SYS_QUERY_HISTORY](#)。SYS 監控檢視中的資料已格式化，以便於使用和理解。建議您針對查詢使用SYS監控檢視。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	integer	查詢或子查詢的交易 ID。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可以與多個查詢 建立關聯IDs。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
service_class	integer	服務類別的 ID。如需服務類別 的清單IDs，請參閱 WLM 服務類別 IDs 。
slot_count	integer	WLM 查詢根據為佇列設定的並行層級使用的查詢槽數量。預設值為 1。如需詳細資訊，請參閱 wlm_query_slot_count 。
service_class_start_time	timestamp	查詢已指派給服務類別的時間。此時間位於UTC時區。
queue_start_time	timestamp	查詢進入服務類別佇列的時間。此時間位於UTC時區。
queue_end_time	timestamp	查詢離開服務類別佇列的時間。此時間位於UTC時區。
total_queue_time	bigint	查詢在佇列中花費的微秒總數
exec_start_time	timestamp	查詢開始在服務類別執行的時間。此時間位於UTC時區。
exec_end_time	timestamp	查詢在服務類別完成執行的時間。此時間位於UTC時區。
total_exec_time	bigint	查詢花費在執行的微秒數。

欄名稱	資料類型	描述
service_class_end_time	timestamp	查詢離開服務類別的時間。此時間位於UTC時區。
final_state	character(16)	保留以供系統使用。
est_peak_mem	bigint	保留以供系統使用。
query_priority	char(20)	查詢的優先順序。可能的值為 n/a、lowest、low、normal、high 和 highest，其中 n/a 表示不支援查詢優先順序。
service_class_name	character(64)	服務類別名稱。如需服務類別的詳細資訊，請參閱 WLM系統資料表和檢視 。

範例查詢

檢視佇列和執行中的平均查詢時間

下列查詢顯示大於 4 之服務類別的目前組態。如需服務類別的清單IDs，請參閱 [WLM 服務類別 IDs](#)。

下列查詢會傳回每個查詢花費在查詢佇列以及執行每個服務類別的平均時間 (以微秒為單位)。

```
select service_class as svc_class, count(*),
avg(datediff(microseconds, queue_start_time, queue_end_time)) as avg_queue_time,
avg(datediff(microseconds, exec_start_time, exec_end_time )) as avg_exec_time
from stl_wlm_query
where service_class > 4
group by service_class
order by service_class;
```

此查詢傳回下列範例輸出：

```
svc_class | count | avg_queue_time | avg_exec_time
-----+-----+-----+-----
5 | 20103 | 0 | 80415
5 | 3421 | 34015 | 234015
6 | 42 | 0 | 944266
7 | 196 | 6439 | 1364399
```

(4 rows)

檢視佇列和執行中的查詢時間上限

下列查詢會傳回每個查詢花費在任何查詢佇列以及執行每個服務類別的時間量上限 (以微秒為單位)。

```
select service_class as svc_class, count(*),
max(datediff(microseconds, queue_start_time, queue_end_time)) as max_queue_time,
max(datediff(microseconds, exec_start_time, exec_end_time )) as max_exec_time
from stl_wlm_query
where svc_class > 5
group by service_class
order by service_class;
```

svc_class	count	max_queue_time	max_exec_time
6	42	0	3775896
7	197	37947	16379473

(4 rows)

STV用於快照資料的表格

STV資料表是包含目前系統資料快照的虛擬系統表格。

主題

- [STV_ACTIVE_CURSORS](#)
- [STV_BLOCKLIST](#)
- [STV_CURSOR_CONFIGURATION](#)
- [STV__ ISOLATION LEVEL](#)
- [STV_EXEC_STATE](#)
- [STV_INFLIGHT](#)
- [STV_LOAD_STATE](#)
- [STV_LOCKS](#)
- [STV_毫升_MODEL INFO](#)
- [STV_MV_DEPS](#)
- [STV_MV_INFO](#)
- [STV_NODE_STORAGE_CAPACITY](#)

- [STV_PARTITIONS](#)
- [STV_QUERY_METRICS](#)
- [STV_RECENTS](#)
- [STV_SESSIONS](#)
- [STV_SLICES](#)
- [STV_STARTUP_RECOVERY_STATE](#)
- [STV_TBL_PERM](#)
- [STV_TBL_TRANS](#)
- [STV_WLM_CLASSIFICATION_CONFIG](#)
- [STV_WLM_QMR_CONFIG](#)
- [STV_WLM_QUERY_QUEUE_STATE](#)
- [STV_WLM_QUERY_STATE](#)
- [STV_WLM_QUERY_TASK_STATE](#)
- [STV_WLM_SERVICE_CLASS_CONFIG](#)
- [STV_WLM_SERVICE_CLASS_STATE](#)
- [STV_XRESTORE_ALTER_QUEUE_STATE](#)

STV_ACTIVE_CURSORS

STV_ACTIVE_CURSORS 顯示當前打開的游標的詳細信息。如需詳細資訊，請參閱[DECLARE](#)。

STV_ACTIVE_CURSORS 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。使用者只能檢視該使用者所開啟的游標。超級使用者可檢視所有游標。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
name	character (256)	游標名稱。

欄名稱	資料類型	描述
xid	bigint	交易細節。
pid	integer	執行查詢的前置程序。
starttime	timestamp	宣告游標的時間。
row_count	bigint	游標結果集中的資料列數。
byte_count	bigint	游標結果集中的位元組數。
fetched_rows	bigint	目前從游標結果集擷取的資料列數。

STV_BLOCKLIST

STV_BLOCKLIST 包含資料庫中每個磁碟片段、資料表或資料行所使用的 1 MB 磁碟區塊數。

如下列範例所示BLOCKLIST，使用彙總查詢搭配 STV_，判斷每個資料庫、資料表、磁碟片段或資料行配置的 1 MB 磁碟區塊數目。您也可以使用 [STV_PARTITIONS](#) 來檢視磁碟使用率的摘要資訊。

STV_BLOCKLIST 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

Note

STV_BLOCKLIST 僅記錄已佈建叢集或無伺服器命名空間所擁有的區塊。如果資料庫包含來自資料清單生產者共用的區塊，則這些區塊不會包含在 STV_BLOCKLIST 中。如需有關資料傳輸的詳細資訊，請移至 [Amazon Redshift 中的資料共用](#)

資料表欄

欄名稱	資料類型	描述
分割	integer	節點分割。

欄名稱	資料類型	描述
col	integer	欄位的以零為起始的索引。您建立的每個表格都會附加三個隱藏欄：INSERTDELETE_XID XID、_ 和 ROW_ID (OID)。包含 3 個使用者定義欄位的資料表包含 6 個實際欄位，使用者定義的欄位的內部編號為 0、1 及 2。在此範例中 XIDXID，INSERTDELETE_、_ 和 ROW_ID 欄分別編號為 3、4 和 5。
tbl	integer	資料庫資料表的資料表 ID。
blocknum	integer	資料區塊的 ID。
num_values	integer	區塊所包含的值的數量。
extended_limits	integer	供內部使用。
minvalue	bigint	區塊的最小資料值。儲存前 8 個字元做為非數值資料的 64 位元整數。用於磁碟掃描。
maxvalue	bigint	區塊的最大資料值。儲存前 8 個字元做為非數值資料的 64 位元整數。用於磁碟掃描。
sb_pos	integer	磁碟上的超級區塊位置的內部 Amazon Redshift 識別碼。
pinned	integer	區塊是否固定至記憶體做為預載的一部分。0 = false ; 1 = true。預設為 false。
on_disk	integer	區塊是否已自動儲存於磁碟。0 = false ; 1 = true。預設為 false。
modified	integer	區塊是否已修改。0 = false ; 1 = true。預設為 false。
hdr_modified	integer	區塊標頭是否已修改。0 = false ; 1 = true。預設為 false。
unsorted	integer	區塊是否未排序。0 = false ; 1 = true。預設為 true。
tombstone	integer	供內部使用。

欄名稱	資料類型	描述
preferred_diskno	integer	區塊應處於開啟狀態的磁碟數量 (無論磁碟是否故障)。一旦磁碟修復，區塊將移回該磁碟。
temporary	integer	無論區塊是否包含暫存資料，例如來自暫存資料表或中繼查詢結果。0 = false ; 1 = true。預設為 false。
newblock	integer	指出區塊是否是新的 (true) 或不曾遞交至磁碟 (false)。0 = false ; 1 = true。
num_readers	integer	每個區塊上的參考數量。
flags	integer	區塊標頭檔內部 Amazon Redshift 旗標。

範例查詢

STV_ 每個配置的磁碟區塊BLOCKLIST包含一個資料列，因此選取所有資料列的查詢可能會傳回非常大量的資料列。我們建議您僅使用 STV_ 的彙總查詢BLOCKLIST。

該[SVV_DISKUSAGE](#)視圖以更易於使用的格式提供類似的信息；但是，下面的示例演示了 STV_ BLOCKLIST 表的一種用法。

若要判斷資料表中每個資料行所使用的 1 MB 區塊數VENUE目，請輸入下列查詢：

```
select col, count(*)
from stv_blocklist, stv_tbl_perm
where stv_blocklist.tbl = stv_tbl_perm.id
and stv_blocklist.slice = stv_tbl_perm.slice
and stv_tbl_perm.name = 'venue'
group by col
order by col;
```

此查詢會傳回配置給資料VENUE表中每個資料行的 1 MB 區塊數目，如下列範例資料所示：

```
col | count
-----+-----
0 | 4
```

```

1 | 4
2 | 4
3 | 4
4 | 4
5 | 4
7 | 4
8 | 4
(8 rows)

```

以下查詢顯示資料表資料是否實際發佈至所有分割：

```

select trim(name) as table, stv_blocklist.slice, stv_tbl_perm.rows
from stv_blocklist, stv_tbl_perm
where stv_blocklist.tbl=stv_tbl_perm.id
and stv_tbl_perm.slice=stv_blocklist.slice
and stv_blocklist.id > 10000 and name not like '%#m%'
and name not like 'systable%'
group by name, stv_blocklist.slice, stv_tbl_perm.rows
order by 3 desc;

```

此查詢會產生下列範例輸出，顯示具有最多資料列之資料表的偶數資料分佈：

```

table | slice | rows
-----+-----+-----
listing | 13 | 10527
listing | 14 | 10526
listing | 8 | 10526
listing | 9 | 10526
listing | 7 | 10525
listing | 4 | 10525
listing | 17 | 10525
listing | 11 | 10525
listing | 5 | 10525
listing | 18 | 10525
listing | 12 | 10525
listing | 3 | 10525
listing | 10 | 10525
listing | 2 | 10524
listing | 15 | 10524
listing | 16 | 10524
listing | 6 | 10524
listing | 19 | 10524

```

```
listing |      1 | 10523
listing |      0 | 10521
...
(180 rows)
```

下列查詢可判斷是否有任何已刪除標記的區塊已遞交至磁碟：

```
select slice, col, tbl, blocknum, newblock
from stv_blocklist
where tombstone > 0;

slice | col |  tbl | blocknum | newblock
-----+-----+-----+-----+-----
4      |  0 | 101285 |      0    |      1
4      |  2 | 101285 |      0    |      1
4      |  4 | 101285 |      1    |      1
5      |  2 | 101285 |      0    |      1
5      |  0 | 101285 |      0    |      1
5      |  1 | 101285 |      0    |      1
5      |  4 | 101285 |      1    |      1
...
(24 rows)
```

STV_CURSOR_CONFIGURATION

STV_CURSOR_CONFIGURATION 顯示游標組態約束。如需詳細資訊，請參閱[游標限制條件](#)。

STV_CURSOR_CONFIGURATION 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
current_cursor_count	integer	目前已開啟的游標數量。
max_diskspace_usable	integer	可供游標使用之磁碟空間量，以 MB 為單位。此限制根據叢集的最大游標結果集大小而定。

欄名稱	資料類型	描述
current_d iskspace_ used	integer	目前游標使用之磁碟空間量，以 MB 為單位。

STV__ ISOLATION LEVEL

STV_DB ISOLATION _ LEVEL 顯示資料庫目前的隔離層級。如需隔離層級的相關資訊，請參閱 [CREATE DATABASE](#)。

STV_DB_ ISOLATION _ 對所有使用者LEVEL都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
db_name	characte (128)	資料庫名稱。
isolation _level	characte (20)	資料庫的隔離層級。可能的值包括 Serializable 和 Snapshot Isolation 。

STV_EXEC_STATE

使用 STV EXEC _ 資料STATE表來瞭解有關在計算節點上主動執行之查詢和查詢步驟的資訊。

此資訊通常僅用於排解工程設計問題。視圖 SVV __ STATE 和 QUERY __ 從 SVL QUERY _ 中 SUMMARY提取他們的信息 STV EXEC _ STATE。

STV_ EXEC _ STATE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_QUERY_DETAIL](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
分割	integer	步驟完成的節點分割。
segment	integer	執行查詢的區段。查詢區段是一系列的步驟。
step	integer	完成查詢區段的步驟。步驟是查詢執行的最小單位。
starttime	timestamp	步驟執行的時間。
currenttime	timestamp	目前時間。
tasknum	integer	指派完成步驟之查詢任務程序。
rows	bigint	處理的資料列數。
位元組	bigint	處理的位元組數。
label	char(256)	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 scan tbl=100448 name =user)。三位數表IDs通常指的是瞬態表的掃描。當您看見 tbl=0 時，通常是指常數值的掃描。
is_diskbased	char(1)	查詢的這個步驟是否以磁碟型操作方式完成：true (t) 或 false (f)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中完成。
workmem	bigint	已指派給步驟之運作中記憶體的位元組數。
num_parts	integer	雜湊資料表在雜湊步驟期間已分割的分割區數。此欄位中的正數並不表示雜湊步驟以磁碟式操作方式執行。檢查 IS_DISKBASED 資料行中的值，以查看雜湊步驟是否是以磁碟為基礎。

欄名稱	資料類型	描述
is_rrscan	char(1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。預設為 false (f)。
is_delayed_scan	char(1)	若為 true (t)，表示已在該步驟上使用延遲的掃描。預設為 false (f)。

範例查詢

Amazon Redshift 不 STATE 直接查詢 STV EXEC __，建議您查詢 SVL _ SUMMARY 或 QUERY _ SVV QUERY _ STATE 以更方便使用的格式取得 STV EXEC __ STATE 中的資訊。如需詳細資訊，請參閱 [SVL_QUERY_SUMMARY](#) 或 [SVV_QUERY_STATE](#) 資料表文件。

STV_INFLIGHT

使用 STV _ INFLIGHT 表格來判斷叢集上目前正在執行的查詢。如果您要進行疑難排解，檢查長時間執行查詢的狀態會很有幫助。

STV _ INFLIGHT 不僅顯示領導者節點查詢。如需詳細資訊，請參閱 [僅限領導節點函數](#)。STV _ INFLIGHT 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在 SYS 監視檢視中找到 [SYS_QUERY_HISTORY](#)。SYS 監視檢視中的資料格式化為更易於使用和理解。我們建議您使用 SYS 監視檢視來進行查詢。

使用 STV _ 疑難排解 INFLIGHT

如果您使用 STV _ INFLIGHT 來疑難排解查詢或查詢集合的效能，請注意下列事項：

- 長時間運行的開放交易通常會增加負載。這些開啟的交易可能會導致其他查詢的執行時間較長。
- 長時間執行 COPY 和 ETL 工作可能會影響叢集上執行的其他查詢，如果它們需要大量的計算資源。在大多數情況下，將這些長時間執行的工作移至低使用率的時間，可提高報告或分析工作負載的效能。
- 有跡象表明，提供相關信息給 STV _ INFLIGHT。其中包括 [STL_QUERYTEXT](#) 擷取指 SQL 令的查詢文字，以及 [SVV_QUERY_INFLIGHT](#) INFLIGHT 將 STV _ 連接到 STL _ QUERYTEXT。您也可以使用 STV _ [STV_RECENTS](#) INFLIGHT 進行故障排除。例如，STV _ RECENTS 可以指出特定查詢處於「執行中」或「完成」狀態。結合此資訊與來自 STV _ 的結果 INFLIGHT 可以提供有關查詢屬性和計算資源影響的詳細資訊。

您也可以使用 Amazon Redshift 主控台監控執行中的查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
分割	integer	查詢執行時所在的分割。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
label	character(320)	用於執行查詢的檔案名稱或使用 SET QUERY _ GROUP 指令定義的標籤。如果查詢不是以檔案為基礎或未設定 QUERY _ GROUP 參數，則此欄位為空白。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。您可以使用此資料欄來聯結至 STL_ERROR 資料表。
starttime	timestamp	查詢開始的時間。
text	character(100)	查詢文字，如果陳述式超過限制，將會截斷為 100 個字元。
suspended	integer	查詢是否已遭停用。0 = false ; 1 = true。
insert_privilege	integer	當目前查詢正在執行時，寫入查詢是否能夠執行。1 = 不允許寫入查詢。0 = 允許寫入查詢。此欄位適用於偵錯。
concurrency_scaling_status	integer	指出查詢是執行於主要叢集或並行擴展叢集，可能的值如下： 0 - 執行於主要叢集 1 - 執行於並行擴展叢集

範例查詢

若要檢視目前正在資料庫上執行的所有使用中查詢，請輸入下列查詢：

```
select * from stv_inflight;
```

下列範例輸出顯示目前正在執行的兩個查詢，包括 STV_INFLIGHT 查詢本身，以及從指令碼執行的查詢，名為avgwait.sql：

```
select slice, query, trim(label) querylabel, pid,
starttime, substring(text,1,20) querytext
from stv_inflight;
```

slice	query	querylabel	pid	starttime	querytext
1011	21		646	2012-01-26 13:23:15.645503	select slice, query,
1011	20	avgwait.sql	499	2012-01-26 13:23:14.159912	select avg(datediff(

(2 rows)

下列查詢會選擇幾個欄，包括 concurrency_scaling_status。此欄指出是否要將查詢傳送至並行擴展叢集。如果該值對於某些結果是 1，則表示正在使用並行擴展運算資源。如需詳細資訊，請參閱[並行擴展](#)。

```
select userid,
query,
pid,
starttime,
text,
suspended,
concurrency_scaling_status
from STV_INFLIGHT;
```

範例輸出會顯示傳送至並行擴展叢集的一個查詢。

query	pid	starttime	text	suspended
concurrency_scaling_status				
1234567	123456	2012-01-26 13:23:15.645503	select userid, query...	0

1

```
2345678 | 234567 | 2012-01-26 13:23:14.159912 | select avg(datediff(... 0
0
(2 rows)
```

如需疑難排解查詢效能的詳細提示，請參閱[查詢疑難排解](#)。

STV_LOAD_STATE

使用 STV_LOAD_STATE 表查找有關正在進行中COPY語句的當前狀態的信息。

指COPY令會在載入每百萬筆記錄之後更新此資料表。

STV_LOAD_STATE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
session	integer	執行負載PID的程序工作階段。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
分割	integer	節點分割數量。
pid	integer	處理程序 ID。工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系列查詢，則此值通常會保持不變。
recordtime	timestamp	記錄被記錄的時間。
bytes_to_load	bigint	此分割要載入的位元組總數。如果要載入的資料已壓縮，此數值為 0
bytes_loaded	bigint	此分割要載入的位元組數。如果要載入的資料已壓縮，此數值為資料解壓縮之後載入的位元組數。
bytes_to_load_compressed	bigint	此分割要載入的已壓縮資料的位元組總數。如果要載入的資料未壓縮，此數值為 0。

欄名稱	資料類型	描述
bytes_loaded_compressed	bigint	此分割要載入的已壓縮資料的位元組數。如果要載入的資料未壓縮，此數值為 0。
lines	integer	此分割要載入的行數。
num_files	integer	此分割要載入的檔案數。
num_files_complete	integer	此分割要載入的檔案數。
current_file	character (256)	此分割要載入的檔案名稱。
pct_complete	integer	此分割已完成載入的資料百分比。

範例查詢

若要檢視命 COPY 令的每個磁碟片段的進度，請鍵入下列查詢。此範例使用 PG_LAST_COPY_ID () 函數擷取最後一 COPY 個命令的資訊。

```
select slice , bytes_loaded, bytes_to_load , pct_complete from stv_load_state where
query = pg_last_copy_id();
```

```
slice | bytes_loaded | bytes_to_load | pct_complete
-----+-----+-----+-----
      2 |           0 |           0 |           0
      3 |    12840898 |    39104640 |           32
(2 rows)
```

STV_LOCKS

使用 STV_LOCKS 表格來檢視資料庫中資料表的任何目前更新。

Amazon Redshift 鎖定資料表以避免兩個使用者同時更新相同的資料表。當 STV_LOCKS 資料表顯示所有目前的資料表更新時，請查詢 [STL_TR_CONFLICT](#) 資料表以查看鎖定衝突的記錄。使用 [SVV_TRANSACTIONS](#) 檢視以識別開啟交易與鎖定爭用的問題。

STV_LOCKS 僅對超級用戶可見。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
table_id	bigint	取得鎖定之資料表的資料表 ID。
last_commit	timestamp	資料表中最近一次遞交的時間戳記。
last_update	timestamp	資料表最近一次更新的時間戳記。
lock_owner	bigint	與鎖定關聯的交易 ID。
lock_owner_pid	bigint	與鎖定關聯的處理程序 ID。
lock_owner_start_ts	timestamp	交易開始時的時間戳記。
lock_owner_end_ts	timestamp	交易結束時的時間戳記。
lock_status	character (22)	等待或保持鎖定的處理程序狀態。

範例查詢

若要檢視在目前交易中進行的所有鎖定，請輸入下列命令：

```
select table_id, last_update, lock_owner, lock_owner_pid from stv_locks;
```

此查詢會傳回以下範例輸出，顯示目前生效的三個鎖定：

```

table_id |                last_update                | lock_owner | lock_owner_pid
-----+-----+-----+-----
100004  | 2008-12-23 10:08:48.882319 |    1043    |          5656
100003  | 2008-12-23 10:08:48.779543 |    1043    |          5656
100140  | 2008-12-23 10:08:48.021576 |    1043    |          5656
(3 rows)

```

STV_毫升_MODEL INFO

關於機器學習模型目前狀態的狀態資訊。

STV_ML_MODEL_ 對所有使用者INFO都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
schema_name	char(128)	模型的命名空間。
user_name	char(128)	模型的擁有者。
model_name	char(128)	模型的名稱。
life_cycle	char(20)	模型的生命週期狀態。
is_refreshable	integer	模型的狀態，指出如果訓練查詢中的原始資料表和欄仍然存在且使用者仍具有這些項目的許可，則模型是否可重新整理。可能的值是：1 (可重新整理) 和 0 (無法重新整理)。
model_state	char(128)	膜性的目前狀態。

範例查詢

下列查詢會顯示機器學習模型的目前狀態。

```
SELECT schema_name, model_name, model_state
FROM stv_ml_model_info;
```

```

schema_name |          model_name          |          model_state
-----+-----+-----
public      | customer_churn_auto_model    | Train Model On SageMaker In Progress
public      | customer_churn_xgboost_model | Model is Ready
(2 row)
```

STV_MV_DEPS

STV_MV_DEPS 表格顯示具體化視觀表在 Amazon Redshift 內其他具體化視觀表的相依性。

如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

STV_MV_ 對所有使DEPS用者都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
db_name	char(128)	包含所指定具體化視觀表的資料庫。
結構描述	char(128)	具體化視觀表的結構描述。
name	char(128)	具體化視觀表的名稱。
ref_schema	char(128)	此具體化視觀表所依賴的具體化視觀表結構描述。
ref_name	char(128)	此具體化視觀表相依之具體化視觀表的名稱。
ref_datab ase_name	char(128)	此具體化視觀表相依的資料庫名稱。

範例查詢

下列查詢會傳回輸出列，指出具體化視觀表 mv_over_foo 使用其定義中的具體化視觀表 mv_foo 做為相依性。

```
CREATE SCHEMA test_ivm_setup;
CREATE TABLE test_ivm_setup.foo(a INT);
CREATE MATERIALIZED VIEW test_ivm_setup.mv_foo AS SELECT * FROM test_ivm_setup.foo;
CREATE MATERIALIZED VIEW test_ivm_setup.mv_over_foo AS SELECT * FROM
  test_ivm_setup.mv_foo;

SELECT * FROM stv_mv_deps;

 db_name | schema          | name          | ref_schema | ref_name |
 ref_database_name
-----+-----+-----+-----+-----+
+-----+
 dev     | test_ivm_setup | mv_over_foo | test_ivm_setup | mv_foo   | dev
```


STV_MV_INFO

STV_MV_INFO 表格包含每個具體化視觀表的資料列、資料是否過時，以及狀態資訊。

如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

STV_MV_ 對所有使INFO用者都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
db_name	char(128)	包含具體化視觀表的資料庫。
結構描述	char(128)	資料庫的結構描述。
name	char(128)	具體化視觀表名稱。
updated_upto_xid	bigint	保留供內部使用。
is_stale	char(1)	<p>t 表示具體化視觀表已過時。「過時」的具體化視觀表是基底資料表已更新，但具體化視觀表尚未重新整理時的具體化視觀表。如果重新整理自上一次重新啟動以來都並未執行，這項資訊可能會不準確。</p> <p>如果具體化視觀表相依於可變函數，is_stale 欄一律會設定為 t。當給出相同的引數或參數時，可變函數會傳回不同的結果。例如，傳回日期或時間戳記的大多數函數都是可變函數。</p>
owner_user_name	char(128)	擁有具體化視觀表的使用者。
state	integer	<p>具體化視觀表的狀態如下：</p> <ul style="list-style-type: none"> 0 - 重新整理時，會完全重新計算具體化視觀表。

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> • 1 - 具體化視觀表為累加式。 • 101 - 具體化視觀表因遭到卸除的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 102 - 具體化視觀表因為資料欄類型變更而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 103 - 具體化視觀表因為重新命名的資料表而無法重新整理。 • 104 - 具體化視觀表因為重新命名的資料欄而無法重新整理。即使未在具體化視觀表中使用此資料欄，也適用此限制條件。 • 105 - 具體化視觀表因為重新命名的結構描述而無法重新整理。
autorewrite	char(1)	t 表示具體化視觀表適用於自動重新寫入查詢。
autorefresh	char(1)	t 表示可以自動重新整理具體化視觀表。

範例查詢

若要檢視所有具體化視觀表的狀態，請執行下列查詢。

```
select * from stv_mv_info;
```

此查詢傳回下列範例輸出。

```

db_name |          schema          | name | updated_upto_xid | is_stale | owner_user_name
| state | autorefresh | autorewrite
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
dev      | test_ivm_setup          | mv   |          1031 | f        | catch-22
|      1 |           1 |           0
dev      | test_ivm_setup          | old_mv |          988 | t        | lotr
|      1 |           0 |           1

```

STV_NODE_STORAGE_CAPACITY

STV_NODE_STORAGE_CAPACITY 表格會顯示叢集中每個節點的總儲存容量和總使用容量的詳細資訊。它包含每個節點的一列。

STV_NODE_STORAGE_CAPACITY 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
節點	integer	節點號碼。
used	integer	節點上使用中的 1 MB 磁碟區塊數。對於RA3節點類型，使用的區塊包括本機快取區塊和 Amazon S3 中保存的區塊。
容量	integer	為節點佈建的總儲存容量 (以 1 MB 為單位)。容量包括 Amazon Redshift 在DC2節點類型上保留的空間，以供內部使用。容量大於名目節點容量，即可用於使用者資料的節點空間量。對於RA3節點類型，此容量與叢集的受管理儲存配額總計相同。如需按節點類型分類容量的相關資訊，請參閱《Amazon Redshift 管理指南》中的 節點類型詳細資訊 。

範例查詢

Note

下列範例的結果會根據叢集的節點規格而有所不同。將欄新增capacitySQLSELECT至您以擷取叢集的容量。

下列查詢會傳回 1 MB 磁碟區塊中已使用的空間和總容量。此範例在雙節點 dc2.8xlarge 叢集上執行。

```
select node, used from stv_node_storage_capacity order by node;
```

此查詢傳回下列範例輸出。

```
node | used
-----+-----
  0 | 30597
  1 | 27089
```

下列查詢會傳回 1 MB 磁碟區塊中已使用的空間和總容量。此範例在雙節點 ra3.16xlarge 叢集上執行。

```
select node, used from stv_node_storage_capacity order by node;
```

此查詢傳回下列範例輸出。

```
node | used
-----+-----
  0 | 30591
  1 | 27103
```

STV_PARTITIONS

使用 STV _ PARTITIONS 表格來瞭解 Amazon Redshift 的磁碟速度效能和磁碟使用率。

STV_ 每個邏輯磁碟區的每個節點PARTITIONS包含一個資料列。

STV_ PARTITIONS 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
owner	integer	擁有該分割區的磁碟節點。
託管	integer	實體連接至該分割區的節點。
diskno	integer	包含該分割區的磁碟。

欄名稱	資料類型	描述
part_begin	bigint	分割區的位移。原始裝置以邏輯分割為鏡像區塊的開放空間。
part_end	bigint	分割區的結尾。
used	integer	分割區上使用中的 1 MB 磁碟區塊數。
tossed	integer	已準備刪除，但因釋放其磁碟地址並不安全而未移除的區塊數。如果立即釋放其地址，等待中的交易可能會寫入磁碟上的相同位置。因此，這些被丟棄的區塊將在下次遞交時釋放。例如，當卸除資料表資料行、在作業期間或以磁碟為基礎的查詢 INSERT 作業期間，磁碟區塊可能會標示為 tossed。
容量	integer	1 MB 磁碟區塊的分割區總容量。
reads	bigint	從最近一次叢集重新啟動之後發生的讀取次數。
writes	bigint	從最近一次叢集重新啟動之後發生的寫入次數。
seek_forward	integer	非用於後續地址的請求獲得前一個請求地址的次數。
seek_back	integer	非用於前一個地址的請求獲得後續請求地址的次數。
is_san	integer	分割區是否屬於 SAN。有效值為 0 (false) 或 1 (true)。
失敗	integer	此欄位已淘汰。
mbps	integer	磁碟速度，以每秒的 MB 量計算。
mount	character (256)	通往裝置的目錄路徑。

範例查詢

以下查詢會傳回已使用的磁碟空間與容量 (以 1 MB 磁碟區塊為單位)，並以原始磁碟空間的百分比計算磁碟使用率。原始磁碟空間包括 Amazon Redshift 保留供內部使用的空間，因此會大於名目磁碟容量，它是可供使用者使用的磁碟空間容量。Amazon Redshift 管理主控台效能索引標籤上的已使用磁碟空間百分比指標會報告叢集使用的標稱磁碟容量百分比。建議您監控 Percentage of Disk Space Used (已使用磁碟空間百分比) 指標，以維護叢集名目磁碟容量的使用情形。

⚠ Important

強烈建議您不要超過叢集的名目磁碟容量。雖然在某些情況下，以技術而言是可能的，但超過名目磁碟容量會降低叢集的容錯能力，同時提高遺失資料的風險。

此範例執行於每節點有六個邏輯磁碟分割區的雙節點叢集。空間的使用非常平均地分散至各磁碟，每個磁碟大約已使用 25%。

```
select owner, host, diskno, used, capacity,
(used-tossed)/capacity::numeric *100 as pctused
from stv_partitions order by owner;
```

owner	host	diskno	used	capacity	pctused
0	0	0	236480	949954	24.9
0	0	1	236420	949954	24.9
0	0	2	236440	949954	24.9
0	1	2	235150	949954	24.8
0	1	1	237100	949954	25.0
0	1	0	237090	949954	25.0
1	1	0	236310	949954	24.9
1	1	1	236300	949954	24.9
1	1	2	236320	949954	24.9
1	0	2	237910	949954	25.0
1	0	1	235640	949954	24.8
1	0	0	235380	949954	24.8

(12 rows)

STV_QUERY_METRICS

包含在使用者定義查詢佇列 (服務類別) 中執行的作用中查詢的測量結果資訊，例如處理的資料列數目、使用狀CPU況、輸入/輸出以及磁碟使用量。若要檢視已完成之查詢的指標，請參閱 [STL_QUERY_METRICS](#) 系統資料表。

查詢指標每間隔一秒鐘取樣一次。因此，相同查詢的不同執行可能會傳回稍微不同的時間。另外，可能不會記錄執行時間不到一秒的查詢區段。

STV_QUERY_ 在查詢、區段和步驟層級METRICS追蹤和彙總量度。如需查詢區段和步驟的相關資訊，請參閱[查詢計劃和執行工作流程](#)。許多指標 (例如 max_rows、cpu_time 等等) 是跨節點配量加總的。如需節點配量的相關資訊，請參閱[資料倉儲系統架構](#)。

若要判斷資料列報告指標的層級，請檢查 segment 和 step_type 資料欄：

- 如果 segment 和 step_type 皆為 -1，則資料列報告查詢層級的指標。
- 如果 segment 不是 -1，且 step_type 是 -1，則資料列報告區段層級的指標。
- 如果 segment 和 step_type 皆不是 -1，則資料列報告步驟層級的指標。

STV_QUERY_METRICS 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_QUERY_DETAIL](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
service_class	integer	WLM查詢佇列 (服務類別) 的識別碼。查詢佇列是在配置中 WLM定義的。只會回報使用者定義之佇列的指標。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
starttime	timestamp	查詢開始執行UTC的時間，以 6 位數的精確度為小數秒。例如：2009-06-12 11:29:19.131358 。
slices	integer	叢集的分割數量。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。如果區段值為 -1，指標區段值將累計至查詢層級。
step_type	integer	執行的步驟類型。如需步驟類型的說明，請參閱 步驟類型 。
rows	bigint	步驟處理的資料列數。

欄名稱	資料類型	描述
max_rows	bigint	為步驟匯出的最大資料列數 (彙總所有分割)。
cpu_time	bigint	CPU使用時間，以微秒為單位。在區段層級，區段跨越所有切片的總CPU時間。在查詢層級，查詢所有切片和區段的CPU時間總和。
max_cpu_time	bigint	使用CPU時間上限，以微秒為單位。在區段層級，區段橫跨所有切片所使用的CPU時間上限。在查詢層級，任何查詢區段所使用的CPU時間上限。
blocks_read	bigint	查詢或區段所讀取的 1 MB 區塊數。
max_block_s_read	bigint	區段讀取的 1 MB 區塊最大數量 (彙總所有分割)。在區段層級，所有分割的區段所讀取的 1 MB 區塊最大數量。在查詢層級，任何查詢區段所讀取的 1 MB 區塊最大數量。
run_time	bigint	所有分割合計的總執行時間。執行時間不包含等待時間。 在區段層級，所有分割合計的區段執行時間。在查詢層級，所有分割與區段的查詢執行時間總和。因為此值為總和，因此執行時間與查詢執行時間無關。
max_run_time	bigint	區段的最大經過時間 (以微秒為單位)。在區段層級，所有分割之區段的最大執行時間。在查詢層級，任何查詢區段的最大執行時間。
max_block_s_to_disk	bigint	用於寫入中繼結果的磁碟空間最大數量，以 1 MB 區塊為單位。在區段層級，所有分割之區段已使用的最大磁碟空間量。在查詢層級，任何查詢區段已使用的最大磁碟空間量。
blocks_to_disk	bigint	查詢或區段用於寫入中繼結果的最大磁碟空間量，以 1 MB 區塊為單位。
step	integer	執行的查詢步驟。

欄名稱	資料類型	描述
max_query_scan_size	bigint	查詢所掃描的最大資料大小，以 MB 為單位。在區段層級，所有分割之區段所掃描的最大資料大小。在查詢層級，任何查詢區段所掃描的最大資料大小。
query_scan_size	bigint	查詢所掃描的資料大小，以 MB 為單位。
query_priority	integer	查詢的優先順序。可能的值為 -1、0、1、2、3 和 4，其中 -1 表示不支援查詢優先順序。
query_queue_time	bigint	查詢排入佇列的時間 (毫秒)。

步驟類型

下表列出與資料庫使用者相關的步驟類型。此表未列出僅供內部使用的步驟類型。如果步驟類型為 -1，則該步驟層級不會回報該指標。

Step type (步驟類型)	描述
1	掃描資料表
2	插入資料列
3	彙總資料列
6	排序步驟
7	合併步驟
8	分佈步驟
9	廣播分佈步驟
10	雜湊聯結
11	合併聯結
12	儲存步驟

Step type (步驟類型)	描述
14	Hash
15	巢狀迴路聯結
16	投射欄位與表達式
17	限制傳回的資料列數
18	唯一
20	刪除資料列
26	限制傳回的排序資料列數
29	運算視窗函式
32	UDF
33	唯一
37	從領導者節點將資料列傳回至用戶端
38	從運算節點將資料列傳回至領導節點
40	Spectrum 掃描。

範例查詢

若要尋找具有較CPU長時間 (超過 1,000 秒) 的作用中查詢，請執行下列查詢。

```
select query, cpu_time / 1000000 as cpu_seconds
from stv_query_metrics where segment = -1 and cpu_time > 1000000000
order by cpu_time;
```

```
query | cpu_seconds
-----+-----
25775 |          9540
```

若要找出具有傳回超過一百萬個資料列之巢狀迴圈聯結的使用中查詢，請執行下列查詢。

```
select query, rows
from stv_query_metrics
where step_type = 15 and rows > 1000000
order by rows;
```

```
query | rows
-----+-----
25775 | 1580225854
```

若要尋找執行超過 60 秒且使用CPU時間少於 10 秒的使用中查詢，請執行下列查詢。

```
select query, run_time/1000000 as run_time_seconds
from stv_query_metrics
where segment = -1 and run_time > 60000000 and cpu_time < 10000000;
```

```
query | run_time_seconds
-----+-----
25775 |                114
```

STV_RECENTS

使用 STV _ RECENTS 表找出有關當前活動和最近對數據庫運行查詢的信息。

STV _ RECENTS 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_QUERY_HISTORY](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

使用 STV _ 疑難排解 RECENTS

STV _ 對RECENTS於確定查詢或查詢集合是否正在運行或完成特別有用。它也會顯示查詢執行的持續時間。這對於了解哪些查詢長時間執行很有幫助。

您可以RECENTS將 STV _ 聯結至其他系統檢視，例如[STV_INFLIGHT](#)，以收集有關執行查詢的其他中繼資料。(範例查詢部分中有一個範例展示如何執行此操作。) 您也可以使用此檢視中的傳回記錄以及Amazon Redshift 主控台內的監控功能，進行即時疑難排解。

補充 STV _ RECENTS 包含的系統檢視[STL_QUERYTEXT](#)，可擷取SQL指令的查詢文字，以及[SVV_QUERY_INFLIGHT](#)INFLIGHT將 STV _ 連接到 STL _ QUERYTEXT。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
status	character(20)	查詢狀態。有效值為 Running 、 Done 。
starttime	timestamp	查詢開始的時間。
持續時間	integer	工作階段開始之後的時間 (微秒)。
user_name	character(50)	執行處理程序者的使用者名稱。
db_name	character(50)	資料庫的名稱。
query	character(600)	查詢文字，最多 600 個字元。任何額外的字元都會被截斷。
pid	integer	與查詢關聯之工作階段的處理程序 ID，已完成之查詢的此 ID 一律為 -1。

範例查詢

若要判斷目前正在資料庫上執行的有哪些查詢，請執行下列查詢：

```
select user_name, db_name, pid, query
from stv_recents
where status = 'Running';
```

下面的示例輸出顯示了在 TICKIT 數據庫上運行的單個查詢：

```
user_name | db_name | pid | query
-----+-----+-----+-----
dwuser    | tickit  | 19996 |select venueName, venueSeats from
venue where venueSeats > 50000 order by venueSeats desc;
```

下列範例傳回正在執行或在佇列中等待執行之查詢 (如果有) 的清單：

```
select * from stv_recents where status <> 'Done';
```

```

status |      starttime          | duration |user_name|db_name| query      | pid
-----+-----+-----+-----+-----+-----+-----
Running| 2010-04-21 16:11... | 281566454| dwuser  |tickit  | select ...| 23347

```

除非您正在執行多個並行查詢，而且其中部分查詢位於佇列中，否則此查詢不會傳回結果。

以下範例延伸之前的範例。在此情況中，真正「進行中」(執行中而非等待中)的查詢會排除在結果之外：

```

select * from stv_recents where status<>'Done'
and pid not in (select pid from stv_inflight);
...

```

如需疑難排解查詢效能的詳細提示，請參閱[查詢疑難排解](#)。

STV_SESSIONS

使用 STV_SESSIONS 表格來檢視 Amazon Redshift 作用中使用者工作階段的相關資訊。

若要檢視工作階段歷程記錄，請使用[STL_SESSIONS](#)表格而不是 STV_SESSIONS。

STV_SESSIONS 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_SESSION_HISTORY](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

資料表欄

欄名稱	資料類型	描述
starttime	timestamp	工作階段開始的時間。
process	integer	工作階段的處理程序 ID。
user_name	character(50)	與工作階段相關聯的使用者。
db_name	character(50)	與工作階段相關聯之資料庫的名稱。
timeout_sec	int	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。

範例查詢

若要執行快速檢查以查看是否有任何其他使用者目前已登入 Amazon Redshift，請輸入下列查詢：

```
select count(*)
from stv_sessions;
```

如果結果大於 1，則至少有一位其他使用者目前已登入資料庫。

若要檢視 Amazon Redshift 的所有使用中工作階段，請輸入下列查詢：

```
select *
from stv_sessions;
```

以下結果顯示 Amazon Redshift 上目前執行的四個作用中工作階段：

```

      starttime          | process | user_name          | db_name
      | timeout_sec
-----+-----+-----
+-----+-----+-----
2018-08-06 08:44:07.50 | 13779 | IAMA:aws_admin:admin_grp | dev
      | 0
2008-08-06 08:54:20.50 | 19829 | dwuser              | dev
      | 120
2008-08-06 08:56:34.50 | 20279 | dwuser              | dev
      | 120
2008-08-06 08:55:00.50 | 19996 | dwuser              | tickit
      | 0
(3 rows)
```

前置詞為的使用者名稱IAMA表示使用同盟單一登入登入的使用者。如需詳細資訊，請參閱[使用IAM驗證產生資料庫使用者認證](#)。

STV_SLICES

使用 STV_SLICES 表格可檢視磁碟片段與節點的目前對映。

STV_ 中的信息主要用SLICES於調查目的。

STV_SLICES 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
節點	integer	分割所在的叢集節點。
分割	integer	節點分割。
localslice	integer	此資訊僅供內部使用。
type	character (1)	此資訊僅供內部使用。

範例查詢

若要檢視哪些叢集節點映射哪些分割，請輸入下列查詢：

```
select node, slice from stv_slices;
```

此查詢傳回下列範例輸出：

```
node | slice
-----+-----
    0 |     2
    0 |     3
    0 |     1
    0 |     0
(4 rows)
```

STV_STARTUP_RECOVERY_STATE

記錄在叢集重新啟動操作期間暫時鎖定的資料表狀態。Amazon Redshift 會在處理資料表時暫時鎖定資料表，以解決叢集重新啟動後的過時交易。

STV__STARTUP_RECOVERY_STATE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
db_id	integer	資料庫 ID。
table_id	integer	表格 ID。
table_name	character(137)	資料表名稱

範例查詢

若要監控有哪些資料表被暫時鎖定，請在叢集重新啟動之後，執行以下查詢。

```
select * from STV_STARTUP_RECOVERY_STATE;
```

```

  db_id | tbl_id | table_name
-----+-----+-----
 100044 | 100058 | lineorder
 100044 | 100068 | part
 100044 | 100072 | customer
 100044 | 100192 | supplier
(4 rows)
```

STV_TBL_PERM

STV_TBL_PERM 資料表包含 Amazon Redshift 中永久資料表的相關資訊，包括使用者針對目前工作階段建立的暫存資料表。STV_TBL_PERM 包含所有數據庫中所有表的信息。

此資料表不同於 [STV_TBL_TRANS](#)，它包含系統在查詢處理過程中建立之暫時性資料庫的相關資訊。

STV_TBL_PERM 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
分割	integer	配置至資料表的節點分割。

欄名稱	資料類型	描述
id	integer	表格 ID。
name	character (72)	資料表名稱。
rows	bigint	分割中的資料列數。
sorted_rows	bigint	已在磁碟上排序之分割中的資料列數。如果此數字與數ROWS字不匹配，請將表真空吸塵以求助行。
temp	integer	該資料表是否為暫時資料表。0 = false ; 1 = true。
db_id	integer	用於建立資料表之資料庫的 ID。
insert_privilege	integer	供內部使用。
delete_privilege	integer	供內部使用。
backup	integer	用於指出該資料表是否包含在叢集快照中的值。0 = no ; 1 = yes。如需詳細資訊，請 BACKUP 參閱CREATETABLE指令的參數。
dist_style	integer	切片所屬資料表的分散樣式。如需這些值的詳細資訊，請參閱 檢視分佈樣式 。如需分散樣式的資訊，請參閱 分佈樣式 。
block_count	integer	分割使用的區塊數目。無法計算區塊計數時，值為 -1。

範例查詢

下列查詢會傳回不同資料表IDs和名稱的清單：

```
select distinct id, name
from stv_tbl_perm order by name;
```

```
   id  | name
-----+-----
```

```

100571 | category
100575 | date
100580 | event
100596 | listing
100003 | padb_config_harvest
100612 | sales
...

```

其他系統表使用 tableIDs，因此知道哪個表 ID 對應於某個表可能非常有用。在此示例中，用 SELECTDISTINCT於刪除重複項（表分佈在多個切片中）。

若要判斷資料表中每個資料行所使用的區塊數VENUE目，請輸入下列查詢：

```

select col, count(*)
from stv_blocklist, stv_tbl_perm
where stv_blocklist.tbl = stv_tbl_perm.id
and stv_blocklist.slice = stv_tbl_perm.slice
and stv_tbl_perm.name = 'venue'
group by col
order by col;

```

col	count
0	8
1	8
2	8
3	8
4	8
5	8
6	8
7	8

(8 rows)

使用須知

該ROWS列包括尚未吸塵（或已吸塵但使用選項）的已刪除行計數。SORT ONLY因此，當您直接查詢給定ROWS資料PERM表時，STVTBL_ 資料表中的資料行可能不符合 COUNT (*) 結果。SUM例如，如果從中刪除 2 行VENUE，則COUNT (*) 結果為 200，但SUM (ROWS) 結果仍然是 202：

```

delete from venue
where venueid in (1,2);

```

```
select count(*) from venue;
count
-----
200
(1 row)

select trim(name) tablename, sum(rows)
from stv_tbl_perm where name='venue' group by name;

tablename | sum
-----+-----
venue      | 202
(1 row)
```

要同步 STV _ TBL _ 中的數據 PERM，運行全真空 VENUE 表。

```
vacuum venue;

select trim(name) tablename, sum(rows)
from stv_tbl_perm
where name='venue'
group by name;

tablename | sum
-----+-----
venue      | 200
(1 row)
```

STV_TBL_TRANS

使用 STV _ TBL _ TRANS 表來找出有關目前在記憶體中的暫時性資料庫資料表的資訊。

暫時性資料表通常是暫存的資料列集，在查詢執行時做為中繼結果使用。STV_TBL_TRANS 不同 [STV_TBL_PERM](#) 於 STV_TBL_PERM 包含有關永久性資料庫表的信息。

STV_TBL_TRANS 僅對超級用戶可見。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
分割	integer	配置至資料表的節點分割。

欄名稱	資料類型	描述
id	integer	表格 ID。
rows	bigint	資料表中的資料列數。
size	bigint	配置至資料表的位元組數。
query_id	bigint	查詢 ID。
ref_cnt	integer	參考的數量。
from_suspended	integer	此資料表是否是在查詢 (目前已停用) 期間內所建立的。
prep_swap	integer	此暫時性資料表是否已準備好在需要時交換到磁碟。(此交換只會發生在記憶體量較低的情況。)

範例查詢

若要檢視查詢 ID 為 90 之查詢的暫時性資料表資訊，請輸入以下命令：

```
select slice, id, rows, size, query_id, ref_cnt
from stv_tbl_trans
where query_id = 90;
```

此查詢會傳回查詢 90 的暫時性資料表資訊，如下範例輸出所示：

```
slice | id | rows | size | query_ | ref_ | from_      | prep_
      |   |     |     | id     | cnt  | suspended  | swap
-----+-----+-----+-----+-----+-----+-----+-----
1013 | 95 | 0    | 0    | 90    | 4    | 0          | 0
 7   | 96 | 0    | 0    | 90    | 4    | 0          | 0
10   | 96 | 0    | 0    | 90    | 4    | 0          | 0
17   | 96 | 0    | 0    | 90    | 4    | 0          | 0
14   | 96 | 0    | 0    | 90    | 4    | 0          | 0
 3   | 96 | 0    | 0    | 90    | 4    | 0          | 0
1013 | 99 | 0    | 0    | 90    | 4    | 0          | 0
 9   | 96 | 0    | 0    | 90    | 4    | 0          | 0
 5   | 96 | 0    | 0    | 90    | 4    | 0          | 0
19   | 96 | 0    | 0    | 90    | 4    | 0          | 0
```

2	96	0	0	90	4	0	0
1013	98	0	0	90	4	0	0
13	96	0	0	90	4	0	0
1	96	0	0	90	4	0	0
1013	96	0	0	90	4	0	0
6	96	0	0	90	4	0	0
11	96	0	0	90	4	0	0
15	96	0	0	90	4	0	0
18	96	0	0	90	4	0	0

在此範例中，您可以看到涉及資料表 95、96 及 98 的查詢資料。因為零字元組分配給此資料表，所以此查詢可在記憶體中執行。

STV_WLM_CLASSIFICATION_CONFIG

包含的目前分類規則WLM。

STV_WLM_CLASSIFICATION_CONFIG 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
id	integer	服務類別 ID。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
condition	character(128)	查詢條件。
action_seq	integer	保留以供系統使用。
動作	character(64)	保留以供系統使用。
action_service_class	integer	動作發生時所在的服務類別。

範例查詢

```
select * from STV_WLM_CLASSIFICATION_CONFIG;
```

```

id | condition | action_seq | action |
  | action_service_class
-----+-----+-----
+-----+-----+-----
1 | (system user) and (query group: health) | 0 | assign |
  | 1
2 | (system user) and (query group: metrics) | 0 | assign |
  | 2
3 | (system user) and (query group: cmstats) | 0 | assign |
  | 3
4 | (system user) | 0 | assign |
  | 4
5 | (super user) and (query group: superuser) | 0 | assign |
  | 5
6 | (query group: querygroup1) | 0 | assign |
  | 6
7 | (user group: usergroup1) | 0 | assign |
  | 6
8 | (user group: usergroup2) | 0 | assign |
  | 7
9 | (query group: querygroup3) | 0 | assign |
  | 8
10 | (query group: querygroup4) | 0 | assign |
  | 9
11 | (user group: usergroup4) | 0 | assign |
  | 9
12 | (query group: querygroup*) | 0 | assign |
  | 10
13 | (user group: usergroup*) | 0 | assign |
  | 10
14 | (querytype: any) | 0 | assign |
  | 11
(4 rows)

```

STV_WLM_QMR_CONFIG

記錄查WLM詢監視規則 (QMR) 的組態。如需詳細資訊，請參閱[WLM 查詢監控規則](#)。

STV__WLM_QMR_CONFIG 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
service_class	integer	WLM查詢佇列 (服務類別) 的識別碼。查詢佇列是在配置中WLM定義的。只有使用者定義的佇列可以定義規則。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
rule_name	character(256)	查詢監控規則的名稱。
動作	character(256)	規則動作。可能值為 log、hop、abort 和 change_query_priority 。
metric_name	character(256)	指標的名稱。
metric_operator	character(256)	指標運算子。可能值為 >、=、<。
metric_value	double	觸發動作之指定指標的閾值。
action_value	character(256)	若 action 為 change_query_priority ，則可能的值會是 highest、high、normal、low 和 lowest。 若 action 為 log、hop 或 abort ，則此值為是空白。

範例查詢

若要檢視所有大於 5 (包括使用者定義佇列) 之服務類別的QMR規則定義，請執行下列查詢。如需服務類別的清單IDs，請參閱[WLM 服務類別 IDs](#)。

```
Select *
from stv_wlm_qmr_config
where service_class > 5
order by service_class;
```

STV_WLM_QUERY_QUEUE_STATE

記錄服務類別之查詢佇列的目前狀態。

STV_WLM_QUERY_QUEUE_STATE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_QUERY_HISTORY](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
position	integer	佇列中查詢的位置。下一個執行的是擁有最小 position 值的查詢。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可以與多個查詢相關聯IDs。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
slot_count	integer	WLM查詢插槽的數量。
start_time	timestamp	查詢進入佇列的時間。
queue_time	bigint	查詢在佇列中的時間 (微秒)。

範例查詢

下列查詢會顯示服務類別大於 4 的佇列中的查詢。

```
select * from stv_wlm_query_queue_state
where service_class > 4
order by service_class;
```

此查詢傳回下列範例輸出。

```
service_class | position | task | query | slot_count |          start_time          |
queue_time
```



```

-----+-----+-----+-----+-----+-----+-----
+-----
          5 |          0 | 455 | 476 |          5 | 2010-10-06 13:18:24.065838 |
20937257
          6 |          1 | 456 | 478 |          5 | 2010-10-06 13:18:26.652906 |
18350191
(2 rows)

```

STV_WLM_QUERY_STATE

記錄正在追蹤的查詢的目前狀態WLM。

STV_WLM_QUERY_STATE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_QUERY_HISTORY](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

資料表欄

欄名稱	資料類型	描述
qid	integer	查詢或子查詢的交易 ID。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可以與多個查詢相關聯IDs。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
service_class	integer	服務類別的 ID。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
slot_count	integer	WLM查詢插槽的數量。
wlm_start_time	timestamp	查詢進入系統資料表佇列或短期查詢佇列的時間。
state	character(16)	查詢或子查詢的目前狀態。 可能的值如下：

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> Classified - 查詢已指派給服務類別。 Completed - 查詢已完成執行。查詢執行成功或已取消。對於最終狀態，請檢查 STL_QUERY 的結果。 Dequeued - 僅供內部使用。 Evicted - 已從服務類別撤銷查詢以重新啟動。 Evicting - 正從服務類別撤銷查詢以重新啟動。 Initialized - 僅供內部使用。 Invalid - 僅供內部使用。 Queued - 查詢已傳送至查詢佇列，因為沒有可用的插槽可用來執行查詢佇列。 QueuedWaiting - 查詢正在查詢佇列中等待。 Rejected - 僅供內部使用。 Returning - 查詢正在將結果傳回給用戶端。 Running - 正在執行查詢。 TaskAssigned - 僅供內部使用。
queue_time	bigint	查詢在佇列中花費的時間 (微秒)。
exec_time	bigint	查詢已執行的微秒數。
query_priority	char(20)	查詢的優先順序。可能的值為 n/a、lowest、low、normal、high 和 highest，其中 n/a 表示不支援查詢優先順序。

範例查詢

下列查詢顯示大於 4 之服務類別中所有目前正在執行的查詢。如需服務類別的清單IDs，請參閱[WLM 服務類別 IDs](#)。

```
select xid, query, trim(state) as state, queue_time, exec_time
from stv_wlm_query_state
where service_class > 4;
```

此查詢傳回下列範例輸出：

```
xid      | query | state   | queue_time | exec_time
-----+-----+-----+-----+-----
100813  | 25942 | Running |           0 |    1369029
100074  | 25775 | Running |           0 |   2221589242
```

STV_WLM_QUERY_TASK_STATE

包含服務類別查詢任務的目前狀態。

STV_WLM_QUERY_TASK_STATE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
task	integer	用於透過工作負載管理員追蹤查詢的 ID。可以與多個查詢相關聯IDs。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
query	integer	查詢 ID。如果查詢已重新啟動，會將新的查詢 ID (而非新的任務 ID) 指派給該查詢。
slot_count	integer	WLM查詢插槽的數量。
start_time	timestamp	查詢開始執行的時間。
exec_time	bigint	查詢到目前為止已執行的時間 (微秒)。

範例查詢

下列查詢顯示大於 4 之服務類別中的查詢的目前狀態。如需服務類別的清單IDs，請參閱[WLM 服務類別 IDs](#)。

```
select * from stv_wlm_query_task_state
```

```
where service_class > 4;
```

此查詢傳回下列範例輸出：

```
service_class | task | query |          start_time          | exec_time
-----+-----+-----+-----+-----
          5   |  466 |   491 | 2010-10-06 13:29:23.063787 | 357618748
(1 row)
```

STV_WLM_SERVICE_CLASS_CONFIG

記錄的服務類別組態WLM。

STV_WLM __ SERVICE CLASS _ CONFIG 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
queueing_strategy	character(32)	保留以供系統使用。
num_query_tasks	integer	服務類別的目前實際並行層級。如果num_query_tasks 和target_num_query_tasks 不同，則動態WLM轉移正在進行中。的值-1表示已設定「自動 WLM」。
target_num_query_tasks	integer	由最近的WLM組態變更所設定的並行層級。
evictable	character(8)	保留以供系統使用。
eviction_threshold	bigint	保留以供系統使用。
query_working_mem	integer	指派至服務類別的目前實際運作中記憶體數量，以每節點、每插槽 MB 為單位。如果query_working_mem 和target_query_working_mem 不同，

欄名稱	資料類型	描述
		則動態WLM轉移正在進行中。的值-1表示已設定「自動」(WLMAuto)。
target_query_working_mem	integer	最近一次WLM組態變更所設定的工作記憶體容量，以每個插槽、每個節點為單位的 MB。
min_step_mem	integer	保留以供系統使用。
name	character(64)	服務類別的名稱。
max_execution_time	bigint	查詢在被終止之前可執行的時間 (微秒)。
user_group_wild_card	Boolean	如果TRUE，WLM佇列會將星號 (*) 視為組WLM態中使用者群組字串中的萬用字元。
query_group_wild_card	Boolean	如果TRUE，WLM佇列會將星號 (*) 視為組WLM態中查詢群組字串中的萬用字元。
concurrency_scaling	character(20)	描述並行擴展是 on 或 off。
query_priority	character(20)	查詢優先順序的值。
user_role_wild_card	Boolean	如果TRUE，WLM佇列會將星號 (*) 視為WLM組態中使用者使用者字串中的萬用字元。

範例查詢

第一個使用者定義的服務類別為服務類別 6，其名為 Service class #1。下列查詢顯示大於 4 之服務類別的目前組態。如需服務類別的清單IDs，請參閱[WLM 服務類別 IDs](#)。

```
select rtrim(name) as name,
num_query_tasks as slots,
query_working_mem as mem,
max_execution_time as max_time,
user_group_wild_card as user_wildcard,
query_group_wild_card as query_wildcard
from stv_wlm_service_class_config
```

```
where service_class > 4;
```

name	slots	mem	max_time	user_wildcard	query_wildcard
Service class for super user	1	535	0	false	false
Queue 1	5	125	0	false	false
Queue 2	5	125	0	false	false
Queue 3	5	125	0	false	false
Queue 4	5	627	0	false	false
Queue 5	5	125	0	true	true
Default queue	5	125	0	false	false

下面的查詢顯示了一個動態WLM轉換的狀態。在轉換進行時，`num_query_tasks` 與 `target_query_working_mem` 會進行更新，直到它們等於目標值。如需詳細資訊，請參閱[WLM 動態和靜態組態屬性](#)。

```
select rtrim(name) as name,
num_query_tasks as slots,
target_num_query_tasks as target_slots,
query_working_mem as memory,
target_query_working_mem as target_memory
from stv_wlm_service_class_config
where num_query_tasks > target_num_query_tasks
or query_working_mem > target_query_working_mem
and service_class > 5;
```

name	slots	target_slots	memory	target_mem
Queue 3	5	15	125	375
Queue 5	10	5	250	125

(2 rows)

STV_WLM_SERVICE_CLASS_STATE

包含服務類別的目前狀態。

STV_WLM__SERVICE_CLASS_STATE 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
service_class	integer	服務類別的 ID。如需服務類別的清單IDs，請參閱 WLM 服務類別 IDs 。
num_queued_queries	integer	目前佇列中的查詢數。
num_executing_queries	integer	目前執行中的查詢數。
num_serviced_queries	integer	曾經在服務類別中的查詢數。
num_executed_queries	integer	自 Amazon Redshift 重新啟動後執行的查詢數目。
num_evicted_queries	integer	在 Amazon Redshift 重新啟動之後已移出的查詢數量。收回查詢的一些原因包括WLM逾時、QMR躍點動作以及在並行擴展叢集上的查詢失敗。
num_concurrency_scaling_queries	integer	在 Amazon Redshift 重新啟動之後執行於並行擴展叢集的查詢數量。

範例查詢

下列查詢顯示大於 5 之服務類別的狀態。如需服務類別的清單IDs，請參閱[WLM 服務類別 IDs](#)。

```
select service_class, num_executing_queries,
num_executed_queries
from stv_wlm_service_class_state
where service_class > 5
order by service_class;
```

service_class	num_executing_queries	num_executed_queries
6	1	222
7	0	135
8	1	39

(3 rows)

STV_XRESTORE_ALTER_QUEUE_STATE

使用 `STV_XRESTORE__ALTER_QUEUE_STATE` 來監視傳統調整大小期間每個資料表的移轉進度。當目標節點類型為時，這特別適用RA3。有關經典調整為RA3節點大小的更多信息，請轉到[經典調整大小](#)。

`STV_XRESTORE__ALTER_QUEUE_STATE` 僅對超級用戶可見。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_RESTORE_STATE](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視檢視來進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	起始調整大小的使用者 ID。
db_id	integer	資料庫的識別碼。
結構描述	char(128)	結構描述的名稱。
table_name	char(128)	資料表的名稱。
tbl	integer	資料表的 ID。
status	char(64)	資料表的移轉進度狀態。可能的值如下。 <ul style="list-style-type: none"> • <code>Waiting</code> : 等待重新發佈開始 • <code>Applying</code> : 目前重新分配 • <code>Finished</code> : 已完成的重新分配
任務類型	integer	資料表的重新發佈類型。可能的值如下。 <ul style="list-style-type: none"> • 1: KEY • 2: EVEN <p>如需分配型式的更多資訊，請參閱 〈〉 分佈樣式。</p>

範例查詢

下列查詢顯示資料庫中正在等待調整大小、目前正在調整大小以及完成重新調整大小的資料表數目。

```
select db_id, status, count(*)
from stv_xrestore_alter_queue_state
group by 1,2 order by 3 desc
```

db_id	status	count
694325	Waiting	323
694325	Finished	60
694325	Applying	1

SVCS主要和並行擴展叢集的檢視

SVCS具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的資料表類似，不同STL之處在於STL表格僅針對在主叢集上執行的查詢提供資訊。

主題

- [SVCS_ALERT_EVENT_LOG](#)
- [SVCS_COMPILE](#)
- [SVCS_CONCURRENCY_SCALING_USAGE](#)
- [SVCS_EXPLAIN](#)
- [SVCS_PLAN_INFO](#)
- [SVCS_QUERY_SUMMARY](#)
- [SVCS_S3 LIST](#)
- [SVCS_S3 LOG](#)
- [SVCSSUMMARYPARTITION](#)
- [SVCSSUMMARYQUERY](#)
- [SVCS_STREAM_SEGS](#)
- [SVCS_UNLOAD_LOG](#)

SVCS_ALERT_EVENT_LOG

當查詢最佳化器識別可能表示效能問題的狀況時，請記錄一個提醒。此檢視是衍生自 STL__ALERT_EVENT_LOG 系統資料表，但不會針對在並行縮放叢集上執行的查詢顯示切片層級。使用 SVCS__ALERT_EVENT_LOG 表格來找出改善查詢效能的機會。

查詢包含多個區段，每個區段包含一或多個步驟。如需詳細資訊，請參閱[查詢處理](#)。

Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的資料表類似，不同STL之處在於STL表格僅針對在主叢集上執行的查詢提供資訊。

SVCS__ALERT_EVENT_LOG 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
segment	integer	識別查詢區段的號碼。
step	integer	執行的查詢步驟。
pid	integer	與陳述式和配量相關聯的處理程序 ID。如果相同的查詢在多個切片上執行，PIDs則可能會有多個查詢。
xid	bigint	與陳述式關聯的交易 ID。
事件	character (1024)	提醒事件的說明。
solution	character (1024)	建議的解決方案。

欄名稱	資料類型	描述
event_time	timestamp	查詢開始的時間。UTC總時間包括佇列和執行。秒小數部分的精確度為 6 位元。例如： 2009-06-12 11:29:19.131358 。

使用須知

您可以使用 SVCS __ ALERT EVENT _ LOG 來識別查詢中的潛在問題，然後遵循中的做法[查詢效能調校](#)來最佳化資料庫設計並重新撰寫查詢。SVCS_ALERT _ EVENT _ LOG 記錄以下警報：

- 找不到統計資訊

找不到統計資訊。執行ANALYZE下列資料載入或重大更新，並STATUPDATE與COPY作業搭配使用。如需詳細資訊，請參閱[設計查詢的 Amazon Redshift 最佳實務](#)。

- 巢狀迴圈

巢狀迴路通常是 Cartesian 產品。評估您的查詢，以確保所有參與資料表均已有效聯結。

- 選擇性相當高的篩選條件

傳回的資料列與已掃描資料列的比率低於 0.05。已掃描資料列是 rows_pre_user_filter 的值，而傳回的資料列則是 [STL_SCAN](#) 系統資料表中的資料列值。表示查詢正在掃描例外狀況大量的資料列來決定結果集。這可能是由於找不到排序索引鍵或其不正確所致。如需詳細資訊，請參閱[排序金鑰](#)。

- 過多的幽靈資料列

掃描已略過相當多標示為已刪除但未清空的資料列，或已插入但未遞交的資料列。如需詳細資訊，請參閱[清空資料表](#)。

- 大型分佈

已重新配送超過 1,000,000 個資料列，進行雜湊聯結或彙整。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

- 大型廣播

已播送超過 1,000,000 個資料列，進行雜湊聯結。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

- 序列執行

查詢計畫中指出 DS DIST ALL __ INNER 重新發佈樣式，因為整個內部資料表已重新分配至單一節點，因此會強制執行序列執行。如需詳細資訊，請參閱[查詢最佳化的資料分佈](#)。

範例查詢

下列查詢顯示四個查詢的提醒事件。

```
SELECT query, substring(event,0,25) as event,
substring(solution,0,25) as solution,
trim(event_time) as event_time from svcs_alert_event_log order by query;
```

query	event	solution	event_time
6567 18:20:58	Missing query planner statist	Run the ANALYZE command	2014-01-03
7450 21:19:31	Scanned a large number of del	Run the VACUUM command to rec	2014-01-03
8406 00:34:22	Nested Loop Join in the query	Review the join predicates to	2014-01-04
29512 22:00:00	Very selective query filter:r	Review the choice of sort key	2014-01-06

(4 rows)

SVCS_COMPILE

記錄查詢之每個查詢區段的編譯時間和位置，包括執行於擴展叢集的查詢以及執行於主要叢集的查詢。

Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的檢視類似，不同SVL之處在於SVL檢視僅針對在主叢集上執行的查詢提供資訊。

SVCS_COMPILE 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需有關 SCL _ 的資訊COMPILE，請參閱[SVL_COMPILE](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
query	integer	查詢 ID。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
segment	integer	要編譯的查詢區段。
locus	integer	執行區段的位置， 1 (若在運算節點上) 和 2 (若在領導節點上)。
starttime	timestamp	在世界協調時間 (UTC) 編譯開始的時間。
endtime	timestamp	編譯結束UTC的時間。
compile	integer	值為 0 表示已重複使用編譯， 1 表示已編譯區段。

範例查詢

在此範例中，查詢 35878 和 35879 會執行相同的陳述式。SQL 查詢 35878 的編譯欄位針對四個查詢區段顯示 1，此值表示該區段已編譯。查詢 35879 在每個區段的編譯欄位中顯示 0，表示該區段不需要重新編譯。

```
select userid, xid, pid, query, segment, locus,
datediff(ms, starttime, endtime) as duration, compile
from svcs_compile
where query = 35878 or query = 35879
order by query, segment;
```

userid	xid	pid	query	segment	locus	duration	compile
100	112780	23028	35878	0	1	0	0
100	112780	23028	35878	1	1	0	0
100	112780	23028	35878	2	1	0	0
100	112780	23028	35878	3	1	0	0

```

100 | 112780 | 23028 | 35878 |      4 |      1 |          0 |          0
100 | 112780 | 23028 | 35878 |      5 |      1 |          0 |          0
100 | 112780 | 23028 | 35878 |      6 |      1 |        1380 |          1
100 | 112780 | 23028 | 35878 |      7 |      1 |        1085 |          1
100 | 112780 | 23028 | 35878 |      8 |      1 |        1197 |          1
100 | 112780 | 23028 | 35878 |      9 |      2 |         905 |          1
100 | 112782 | 23028 | 35879 |       0 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       1 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       2 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       3 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       4 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       5 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       6 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       7 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       8 |      1 |          0 |          0
100 | 112782 | 23028 | 35879 |       9 |      2 |          0 |          0

```

(20 rows)

SVCS_CONCURRENCY_SCALING_USAGE

記錄並行擴展的用量期間。每個用量期間都是連續的持續時間，個別並行擴展叢集在此期間主動處理查詢。

SVCS_ CONCURRENCY SCALING _ USAGE 此表格對超級使用者可見。資料庫超級使用者可選擇將它開啟供所有使用者檢視。

資料表欄

欄名稱	資料類型	描述
start_time	沒有時區的時間戳記	用量期間開始的時間。
end_time	沒有時區的時間戳記	用量期間結束的時間。
queries	bigint	在此用量期間執行的查詢數量。
usage_in_seconds	numeric(27,0)	此用量期間的總秒數。

範例查詢

若要以秒為單位檢視特定期間的用量持續時間，請輸入以下查詢：

```
select * from svcs_concurrency_scaling_usage order by start_time;
```

```
start_time | end_time | queries | usage_in_seconds
```

```
-----+-----+-----+-----
2019-02-14 18:43:53.01063 | 2019-02-14 19:16:49.781649 | 48 | 1977
```

SVCS_EXPLAIN

顯示已提交以供執行之查詢的EXPLAIN計劃。

Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的資料表類似，不同STL之處在於STL表格僅針對在主叢集上執行的查詢提供資訊。

SVCS_EXPLAIN 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別碼，其中節點會在查詢執行時映射至一個或多個步驟。
parentid	integer	父節點的計劃節點識別碼。父節點具有一些子節點。例如，合併聯結是已聯結資料表上掃描的父項。
plannode	character(400)	EXPLAIN輸出中的節點文本。參照計算節點執行的計劃節點會在EXPLAIN輸出XN中加上前綴。

欄名稱	資料類型	描述
info	character(400)	計劃節點的限定詞和篩選條件資訊。例如，聯結條件和WHERE子句限制包含在此欄中。

範例查詢

請考慮彙總聯結查詢的下列EXPLAIN輸出：

```
explain select avg(datediff(day, listtime, saletime)) as avgwait
from sales, listing where sales.listid = listing.listid;
          QUERY PLAN
-----
XN Aggregate  (cost=6350.30..6350.31 rows=1 width=16)
-> XN Hash Join DS_DIST_NONE  (cost=47.08..6340.89 rows=3766 width=16)
    Hash Cond: ("outer".listid = "inner".listid)
-> XN Seq Scan on listing  (cost=0.00..1924.97 rows=192497 width=12)
-> XN Hash  (cost=37.66..37.66 rows=3766 width=12)
    -> XN Seq Scan on sales  (cost=0.00..37.66 rows=3766 width=12)

(6 rows)
```

如果您執行此查詢，且其查詢識別碼為 10，您可以使用 SVCS_ 資料EXPLAIN表來查看EXPLAIN命令傳回的相同類型資訊：

```
select query,nodeid,parentid,substring(plannode from 1 for 30),
substring(info from 1 for 20) from svcs_explain
where query=10 order by 1,2;

query| nodeid |parentid|          substring          |          substring
-----+-----+-----+-----+-----
10   |      1 |      0 |XN Aggregate  (cost=6717.61..6 |
10   |      2 |      1 | -> XN Merge Join DS_DIST_NO| Merge Cond:("outer"
10   |      3 |      2 |      -> XN Seq Scan on lis |
10   |      4 |      2 |      -> XN Seq Scan on sal |

(4 rows)
```

請考慮下列查詢：

```
select event.eventid, sum(pricepaid)
```



```

from event, sales
where event.eventid=sales.eventid
group by event.eventid order by 2 desc;

```

```

eventid |    sum
-----+-----
    289 | 51846.00
    7895 | 51049.00
    1602 | 50301.00
     851 | 49956.00
    7315 | 49823.00
...

```

如果此查詢的 ID 為 15，則下列系統資料表查詢會傳回已執行的計劃節點。在此情況下，會保留節點的順序，以顯示執行的實際順序：

```

select query,nodeid,parentid,substring(plannode from 1 for 56)
from svcs_explain where query=15 order by 1, 2 desc;

```

```

query|nodeid|parentid|          substring
-----+-----+-----+-----
15   |    8 |    7 |          -> XN Seq Scan on eve
15   |    7 |    5 |          -> XN Hash(cost=87.98..87.9
15   |    6 |    5 |          -> XN Seq Scan on sales(cos
15   |    5 |    4 |          -> XN Hash Join DS_DIST_OUTER(cos
15   |    4 |    3 |          -> XN HashAggregate(cost=862286577.07..
15   |    3 |    2 |          -> XN Sort(cost=1000862287175.47..10008622871
15   |    2 |    1 | -> XN Network(cost=1000862287175.47..1000862287197.
15   |    1 |    0 | XN Merge(cost=1000862287175.47..1000862287197.46 rows=87
(8 rows)

```

下列查詢會擷取包含視窗函數之任何查詢計劃的查詢：IDs

```

select query, trim(plannode) from svcs_explain
where plannode like '%Window%';

```

```

query|          btrim
-----+-----
26   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
27   | -> XN Window(cost=1000985348268.57..1000985351256.98 rows=170 width=33)
(2 rows)

```

SVCS_PLAN_INFO

使用 SVCS_PLAN_INFO 表格以一組資料列來查看查詢的 EXPLAIN 輸出。這是查看查詢計劃的其他方法。

Note

具有前置詞的系統檢視 SVCS 提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的資料表類似，不同 STL 之處在於 STL 表格僅針對在主叢集上執行的查詢提供資訊。

SVCS_PLAN_INFO 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
nodeid	integer	計劃節點識別碼，其中節點會在查詢執行時映射至一個或多個步驟。
segment	integer	識別查詢區段的號碼。
step	integer	識別查詢步驟的號碼。
locus	integer	執行步驟的位置。若在運算節點上則為 0，若在領導者節點上則為 1。
plannode	integer	計劃節點的列舉值。請查看下表以取得 plannode 的列舉值。(中的 PLANNODE 欄 SVCS_EXPLAIN 包含計劃節點文字。)
startupcost	double precision	傳回此步驟之第一列的預估相對成本。
totalcost	double precision	執行步驟的預估相對成本。
rows	bigint	步驟將產生之資料列的預估數目。

欄名稱	資料類型	描述
位元組	bigint	步驟將產生之位元組的預估數目。

範例查詢

下列範例會比較使用EXPLAIN命令並SELECT查詢 SVCS _ PLAN _ 資INFO料表所傳回之簡單查詢的查詢計劃。

```
explain select * from category;
QUERY PLAN
-----
XN Seq Scan on category (cost=0.00..0.11 rows=11 width=49)
(1 row)

select * from category;
catid | catgroup | catname | catdesc
-----+-----+-----+-----
1 | Sports | MLB | Major League Baseball
3 | Sports | NFL | National Football League
5 | Sports | MLS | Major League Soccer
...

select * from svcs_plan_info where query=256;

query | nodeid | segment | step | locus | plannode | startupcost | totalcost
| rows | bytes
-----+-----+-----+-----+-----+-----+-----+-----+-----
+-----
256 | 1 | 0 | 1 | 0 | 104 | 0 | 0.11 | 11 | 539
256 | 1 | 0 | 0 | 0 | 104 | 0 | 0.11 | 11 | 539
(2 rows)
```

在此範例中，PLANNODE104 指的是CATEGORY資料表的順序掃描。

```
select distinct eventname from event order by 1;

eventname
-----
.38 Special
3 Doors Down
```

```
70s Soul Jam
A Bronx Tale
...
```

```
explain select distinct eventname from event order by 1;
```

```
QUERY PLAN
```

```
-----
XN Merge (cost=1000000000136.38..1000000000137.82 rows=576 width=17)
```

```
Merge Key: eventname
```

```
-> XN Network (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
```

```
Send to leader
```

```
-> XN Sort (cost=1000000000136.38..1000000000137.82 rows=576
width=17)
```

```
Sort Key: eventname
```

```
-> XN Unique (cost=0.00..109.98 rows=576 width=17)
```

```
-> XN Seq Scan on event (cost=0.00..87.98 rows=8798
width=17)
```

```
(8 rows)
```

```
select * from svcs_plan_info where query=240 order by nodeid desc;
```

```
query | nodeid | segment | step | locus | plannode | startupcost |
totalcost | rows | bytes
```

```
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----
240 | 5 | 0 | 0 | 0 | 104 | 0 | 87.98 | 8798 | 149566
240 | 5 | 0 | 1 | 0 | 104 | 0 | 87.98 | 8798 | 149566
240 | 4 | 0 | 2 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 4 | 0 | 3 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 4 | 1 | 0 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 4 | 1 | 1 | 0 | 117 | 0 | 109.975 | 576 | 9792
240 | 3 | 1 | 2 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 3 | 2 | 0 | 0 | 114 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 2 | 2 | 1 | 0 | 123 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
240 | 1 | 3 | 0 | 0 | 122 | 1000000000136.38 | 1000000000137.82 | 576 | 9792
(10 rows)
```

SVCS_QUERY_SUMMARY

使用 SVCS_QUERY_SUMMARY 檢視來尋找有關執行查詢的一般資訊。

請注意，SVCSQUERY_ 中的資訊SUMMARY是從所有節點彙總的。

Note

SVCS_QUERY_SUMMARY 檢視僅包含 Amazon Redshift 完成的查詢相關資訊，而不包含其他公用程式和DDL命令。如需 Amazon Redshift 完成的所有陳述式 (包括和公用程式命令) 的完整清單DDL和資訊，您可以查詢 SVL_STATEMENTTEXT 檢視。

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的檢視類似，不同SVL之處在於SVL檢視僅針對在主叢集上執行的查詢提供資訊。

SVCS_QUERY_SUMMARY 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

此表格中的部分或全部資料也可以在SYS監視檢視中找到[SYS_QUERY_DETAIL](#)。SYS監視檢視中的資料格式化為更易於使用和理解。我們建議您使用SYS監視視圖進行查詢。

如需 SVL_QUERY_ 的資訊SUMMARY，請參閱[SVL_QUERY_SUMMARY](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
stm	integer	串流：查詢的並行區段集。一個查詢擁有一或多個串流。
seg	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	執行的查詢步驟。
maxtime	bigint	此步驟執行的時間量上限 (微秒)。
avgtime	bigint	此步驟執行的平均時間 (微秒)。
rows	bigint	查詢步驟中包含的資料列數。
bytes	bigint	查詢步驟中包含的資料位元組數。

欄名稱	資料類型	描述
rate_row	double precision	每資料列的查詢執行率。
rate_byte	double precision	每位元組的查詢執行率。
label	text	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 scan tbl=100448 name =user)。三位數表IDs通常指的是瞬態表的掃描。當您看見 tbl=0 時，通常是指常數值的掃描。
is_diskbased	character(1)	此查詢步驟在叢集的任何節點上是否以磁碟型操作方式執行：true (t) 或 false (f)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。
is_rrscan	character(1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。預設為 false (f)。
is_delayed_scan	character(1)	若為 true (t)，表示已在該步驟上使用延遲的掃描。預設為 false (f)。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之前已發出的資料列總數。

範例查詢

檢視查詢步驟的處理資訊

下列查詢顯示查詢 87 每個步驟的基本處理資訊：

```
select query, stm, seg, step, rows, bytes
from svcs_query_summary
where query = 87
order by query, seg, step;
```

此查詢會擷取查詢 87 的相關處理資訊，如下範例輸出所示：

query	stm	seg	step	rows	bytes
87	0	0	0	90	1890
87	0	0	2	90	360
87	0	1	0	90	360
87	0	1	2	90	1440
87	1	2	0	210494	4209880
87	1	2	3	89500	0
87	1	2	6	4	96
87	2	3	0	4	96
87	2	3	1	4	96
87	2	4	0	4	96
87	2	4	1	1	24
87	3	5	0	1	24
87	3	5	4	0	0

(13 rows)

判斷查詢步驟是否溢出至磁碟

下列查詢顯示查詢 ID 為 1025 的任何查詢步驟 (請參閱 [SVL_QLOG](#) 檢視來了解如何取得查詢的查詢 ID) 是否溢出至磁碟或查詢完全在記憶體內執行：

```
select query, step, rows, workmem, label, is_diskbased
from svcs_query_summary
where query = 1025
order by workmem desc;
```

此查詢傳回下列範例輸出：

query	step	rows	workmem	label	is_diskbased
1025	0	16000000	141557760	scan tbl=9	f
1025	2	16000000	135266304	hash tbl=142	t
1025	0	16000000	128974848	scan tbl=116536	f
1025	2	16000000	122683392	dist	f

(4 rows)

通過掃描 IS_ 的值 DISKBASED，您可以查看哪些查詢步驟進入磁碟。針對查詢 1025，雜湊步驟在磁碟上執行。步驟可能在包含雜湊、彙總或排序步驟的磁碟上執行。若只要檢視以磁碟為基礎的查詢步驟，請將 **and is_diskbased = 't'** 子句加入上述範例中的 SQL 陳述式。

SVCS_S3 LIST

使用 SVCS_S3 LIST 檢視來取得有關區段層級 Amazon Redshift Spectrum 查詢的詳細資訊。一個區段可以執行一個外部資料表掃描。此檢視衍生自 SVL_S3 LIST 系統檢視，但不會針對在並行擴展叢集上執行的查詢顯示切片層級。

Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的檢視類似，不同SVL之處在於SVL檢視僅針對在主叢集上執行的查詢提供資訊。

SVCS_S3 LIST 對所有使用者都可見。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需 SVL_S3 的相關資訊LIST，請參閱[SVL_S3LIST](#)。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。一個查詢包含多個區段。
節點	integer	節點號碼。
eventtime	timestamp	記錄事UTC件的時間。
儲存貯體	char(256)	Amazon S3 儲存貯體名稱。
字首	char(256)	Amazon S3 儲存貯體位置的字首。
recursive	char(1)	是否對子資料夾遞迴掃描。
retrieved_files	integer	所列檔案的數量。
max_file_size	bigint	所列檔案的檔案大小上限。

欄名稱	資料類型	描述
avg_file_size	double precision	所列檔案的平均檔案大小上限。
generated_splits	integer	檔案分割的數量。
avg_split_length	double precision	檔案分割的平均長度 (位元組)。
duration	bigint	檔案清單的期間 (微秒)。

範例查詢

下列範例會LIST針對最後執行的查詢查詢查詢查詢 SVCS _S3。

```
select *
from svcs_s3list
where query = pg_last_query_id()
order by query,segment;
```

SVCS_S3 LOG

使用 SVCS_S3 LOG 檢視取得有關區段層級 Redshift 頻譜查詢的疑難排解詳細資料。一個區段可以執行一個外部資料表掃描。此檢視衍生自 SVL_S3 LOG 系統檢視，但不會針對在並行擴展叢集上執行的查詢顯示切片層級。

Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的檢視類似，不同SVL之處在於SVL檢視僅針對在主叢集上執行的查詢提供資訊。

SVCS_S3 LOG 對所有使用者都可見。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需 SVL_S3 的相關資訊LOG，請參閱[SVL_S3LOG](#)。

資料表欄

欄名稱	資料類型	描述
pid	integer	處理程序 ID。
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
節點	integer	節點號碼。
eventtime	timestamp	記錄事件UTC的時間。
message	char(512)	日誌項目訊息。


範例查詢

下列範例會LOG針對上次執行的查詢查詢查詢查詢 SVCS _S3。

```
select *
from svcs_s3log
where query = pg_last_query_id()
order by query,segment;
```

SVCSSUMMARYPARTITION

使用 SVCS _S3 PARTITION _ SUMMARY 檢視可取得區段層級的 Redshift 頻譜查詢分割處理的摘要。一個區段可以執行一個外部資料表掃描。

 Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的檢視類似，不同SVL之處在於SVL檢視僅針對在主叢集上執行的查詢提供資訊。

SVCS_S3 PARTITION _ SUMMARY 對所有使用者都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

如需 SVL _S3 的相關資訊PARTITION，請參閱[SVL_S3PARTITION](#)。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
segment	integer	區段號碼。一個查詢包含多個區段。
指派	char(1)	節點間分割指派的類型。
min_start time	timestamp	分割區處理開始的時間。UTC
max_endtime	timestamp	分割區處理完成的時間。UTC
min_duration	bigint	節點針對此查詢所使用的最小分割處理時間 (微秒)。
max_duration	bigint	節點針對此查詢所使用的最大分割處理時間 (微秒)。
avg_duration	bigint	節點針對此查詢所使用的平均分割處理時間 (微秒)。
total_partitions	integer	外部資料表中分割區的總數。
qualified_partitions	integer	合格分割區總數。
min_assigned_partitions	integer	在一個節點上指派的最小分割區數量。

欄名稱	資料類型	描述
max_assigned_partitions	integer	在一個節點上指派的最大分割區數量。
avg_assigned_partitions	bigint	在一個節點上指派的平均分割區數量。

範例查詢

以下範例會取得前次執行查詢的分割區掃描詳細資訊。

```
select query, segment, assignment, min_starttime, max_endtime, min_duration,
       avg_duration
from svcs_s3partition_summary
where query = pg_last_query_id()
order by query, segment;
```

SVCSSUMMARYQUERY

使用 `SVCS_S3_QUERY_SUMMARY` 檢視可取得已在系統上執行的所有 Redshift 頻譜查詢 (S3 查詢) 的摘要。一個區段可以執行一個外部資料表掃描。

Note

具有前置詞的系統檢視 `SVCS` 提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的檢視類似，不同 `SVL` 之處在於 `SVL` 檢視僅針對在主叢集上執行的查詢提供資訊。

`SVCS_S3_QUERY_SUMMARY` 對所有使用者都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

如需 `SVL_S3` 的相關資訊 `QUERY`，請參閱 [SVL_S3QUERY](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行給定項目的使用者 ID。
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
starttime	timestamp	此區段中 UTC Redshift 頻譜查詢開始執行的時間。一個區段可以有一個外部資料表掃描。
endtime	timestamp	此區段中 UTC Redshift 頻譜查詢完成的時間。一個區段可以有一個外部資料表掃描。
elapsed	integer	此區段中的 Redshift Spectrum 查詢執行所需的時間長度 (微秒)。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 1 。如果查詢執行至完成，欄位會包含 0 。
external_table_name	char(136)	資料表外部名稱之名稱的內部格式，適用於外部資料表掃描。
file_format	character(16)	外部資料表資料的檔案格式。
is_partitioned	char(1)	若為 true (t)，此欄位值表示外部資料表已遭分割。
is_rrscan	char(1)	若為 true (t)，此欄位值表示已套用範圍限定的掃描。

欄名稱	資料類型	描述
is_nested	varchar(1)	若為 true (t)，此欄位值表示已存取巢狀資料欄資料類型。
s3_scanned_rows	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的列數。
s3_scanned_bytes	bigint	根據壓縮檔案，從 Amazon S3 經掃描並送至 Redshift Spectrum 層的位元組數。
s3query_returned_rows	bigint	從 Redshift Spectrum 層傳回至叢集的資料列數。
s3query_returned_bytes	bigint	從 Redshift Spectrum 層傳回至叢集的位元組數。傳回至 Amazon Redshift 的大量資料可能會影響系統效能。
files	integer	此 Redshift Spectrum 查詢已處理的檔案數。限制平行處理之好處的少數檔案。
files_max	integer	在某分割上處理的檔案數上限。
files_avg	integer	在某分割上處理的平均檔案數。
splits	bigint	為此區段處理的分割數。在此分割上處理的分割數。有了大型可分割資料檔案 (例如，大於 512 MB 的資料檔案)，Redshift Spectrum 會嘗試將檔案分割至多個 S3 請求來進行平行處理。
splits_max	integer	在此分割上處理的分割數上限。
splits_avg	bigint	在此分割上處理的平均分割數。
total_split_size	bigint	處理的所有分割大小總計。
max_split_size	bigint	處理的分割大小上限 (位元組)。

欄名稱	資料類型	描述
avg_split_size	bigint	處理的平均分割大小 (位元組)。
total_retries	bigint	此區段中 Redshift Spectrum 查詢的重試數總計。
max_retries	integer	一個個別處理檔案的重試數上限。
max_request_duration	bigint	個別檔案請求的最大持續期間 (微秒)。長時間執行的查詢可能表示存在瓶頸。
avg_request_duration	bigint	檔案請求的平均持續時間 (微秒)。
max_request_parallelism	integer	此 Redshift Spectrum 查詢在某個分割上的最大平行請求數。
avg_request_parallelism	double precision	此 Redshift Spectrum 查詢在某個分割上的平均平行請求數。
total_slowdown_count	bigint	外部資料表掃描期間發生，具有降速錯誤的 Amazon S3 請求總數。
max_slowdown_count	integer	在某分割上外部資料表掃描期間發生，具有降速錯誤的最大 Amazon S3 請求數。

範例查詢

以下範例會取得前次執行查詢的掃描步驟詳細資訊。

```
select query, segment, elapsed, s3_scanned_rows, s3_scanned_bytes,
       s3query_returned_rows, s3query_returned_bytes, files
from svcs_s3query_summary
```

```
where query = pg_last_query_id()
order by query,segment;
```

```
query | segment | elapsed | s3_scanned_rows | s3_scanned_bytes | s3query_returned_rows
| s3query_returned_bytes | files
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
4587 |      2 |   67811 |           0 |           0 |           0
|           0 |       0
4587 |      2 |  591568 |      172462 |    11260097 |           8513
|      170260 |       1
4587 |      2 |   216849 |           0 |           0 |           0
|           0 |       0
4587 |      2 |   216671 |           0 |           0 |           0
|           0 |       0
```

SVCS_STREAM_SEGS

列出串流與並行區段之間的關係。

Note

具有前置詞的系統檢視SVCS提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的資料表類似，不同STL之處在於STL表格僅針對在主叢集上執行的查詢提供資訊。

SVCS_STREAM_SEGS 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
串流	integer	查詢的並行區段集。
segment	integer	識別查詢區段的號碼。

範例查詢

若要針對最新查詢檢視串流與並行區段之間的關係，請輸入下列查詢：

```
select *
from svcs_stream_segs
where query = pg_last_query_id();
```

```
query | stream | segment
-----+-----+-----
    10 |      1 |      2
    10 |      0 |      0
    10 |      2 |      4
    10 |      1 |      3
    10 |      0 |      1
(5 rows)
```

SVCS_UNLOAD_LOG

使用 `SVCS_UNLOAD_` 獲取 UNLOAD 操作的詳細信息。

`SVCS_UNLOAD_` 會針對 UNLOAD 陳述式建立的每個檔案，LOG 記錄一個資料列。例如，如果 UNLOAD 創建了 12 個文件，`SVCS_UNLOAD_LOG` 包含 12 個相應的行。此檢視是衍生自 `STL_UNLOAD_LOG` 系統資料表，但不會針對在並行縮放叢集上執行的查詢顯示切片層級。

Note

具有前置詞的系統檢視 SVCS 提供有關主要和並行縮放叢集的查詢詳細資訊。這些檢視與具有前置詞的資料表類似，不同 STL 之處在於 STL 表格僅針對在主叢集上執行的查詢提供資訊。

`SVCS_UNLOAD_LOG` 對所有用戶都可以看到。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。
pid	integer	與查詢陳述式相關聯的處理程序 ID。
路徑	character(1280)	檔案的完整 Amazon S3 物件路徑。
start_time	timestamp	作業的開始時UNLOAD間。
end_time	timestamp	作業的結束時UNLOAD間。
line_count	bigint	已卸載至檔案的行數 (列數)。
transfer_size	bigint	已傳輸的位元組數目。
file_format	character(10)	已卸載檔案的格式。

範例查詢

若要取得透過UNLOAD命令寫入 Amazon S3 的檔案清單，您可以在UNLOAD完成後呼叫 Amazon S3 清單操作；不過，視您發出呼叫的速度而定，清單可能不完整，因為 Amazon S3 清單操作最終是一致的。要立即獲得完整的權威列表，請查詢 SVCS _ UNLOAD _ LOG。

下列查詢會傳回由針對上次完成查詢所建立之檔案UNLOAD的路徑名稱：

```
select query, substring(path,0,40) as path
from svcs_unload_log
where query = pg_last_query_id()
order by path;
```

此命令會傳回下列範例輸出：

```
query | path
-----+-----
2320 | s3://amzn-s3-demo-bucket/venue0000_part_00
2320 | s3://amzn-s3-demo-bucket/venue0001_part_00
2320 | s3://amzn-s3-demo-bucket/venue0002_part_00
2320 | s3://amzn-s3-demo-bucket/venue0003_part_00
```

(4 rows)

主要叢集的 SVL 檢視

SVL 檢視是 Amazon Redshift 中包含 STL 資料表和日誌之參考的系統檢視，可了解詳細資訊。

這些檢視可讓您快速輕鬆存取在那些資料表中找到的常見查詢資料。

Note

SVL_QUERY_SUMMARY 檢視僅包含 Amazon Redshift 執行之查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 執行的所有陳述式 (包括 DDL 和公用程式命令) 的完整清單和相關資訊，您可以查詢 SVL_STATEMENTTEXT 檢視。

主題

- [SVL_AUTO_WORKER_ACTION](#)
- [SVL_COMPILE](#)
- [SVL_DATASHARE_CHANGE_LOG](#)
- [SVL_DATASHARE_CROSS_REGION_USAGE](#)
- [SVL_DATASHARE_USAGE_CONSUMER](#)
- [SVL_DATASHARE_USAGE_PRODUCER](#)
- [SVL_FEDERATED_QUERY](#)
- [SVL_MULTI_STATEMENT_VIOLATIONS](#)
- [SVL_MV_REFRESH_STATUS](#)
- [SVL_QERROR](#)
- [SVL_QLOG](#)
- [SVL_QUERY_METRICS](#)
- [SVL_QUERY_METRICS_SUMMARY](#)
- [SVL_QUERY_QUEUE_INFO](#)
- [SVL_QUERY_REPORT](#)
- [SVL_QUERY_SUMMARY](#)
- [SVL 還原 _ 更改表 _ 進度](#)

- [SVL_S3LIST](#)
- [SVL_S3LOG](#)
- [SVL_S3PARTITION](#)
- [SVL_S3PARTITION_SUMMARY](#)
- [SVL_S3QUERY](#)
- [SVL_S3QUERY_SUMMARY](#)
- [SVL_S3RETRIES](#)
- [SVL_SPATIAL_SIMPLIFY](#)
- [SVL_SPECTRUM_SCAN_ERROR](#)
- [SVL_STATEMENTTEXT](#)
- [SVL_STORED_PROC_CALL](#)
- [SVL_STORED_PROC_MESSAGES](#)
- [SVL_TERMINATE](#)
- [SVL_UDF_LOG](#)
- [SVL_USER_INFO](#)
- [SVL_VACUUM_PERCENTAGE](#)

SVL_AUTO_WORKER_ACTION

記錄 Amazon Redshift 對自動最佳化而定義的資料表執行的自動操作。

只有超級使用者可以看到 SVL_AUTO_WORKER_ACTION。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
table_id	integer	資料表識別碼。
type	character (32)	建議的類型。可能的值是 distkey 和 sortkey。

欄名稱	資料類型	描述
status	character (128)	建議的完成狀態。可能的值為 Start、Complete、Skipped、Abort、Checkpoint 和 Failed
eventtime	timestamp	status 欄的時間戳記。
sequence	integer	截斷 previous_state 值的序列號。當單一 previous_state 包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
previous_state	character (200)	套用建議之前，資料表的先前分散樣式和排序索引鍵。該值會被截斷為 200 個字元的增量。

status 欄的值的一些範例如下：

- 略過：找不到資料表。
- 略過：建議是空的。
- 略過：套用排序索引鍵建議已停用。
- 略過：重試超過資料表的上限。
- 略過：資料表欄已變更。
- 中止：此資料表不是 AUTO。
- 中止：最近已轉換此資料表。
- 中止：此資料表超過資料表大小臨界值。
- 中止：此資料表已經是建議的樣式。
- 檢查點：進度為 **21.9963%**。

範例查詢

在下列範例中，結果中的列顯示 Amazon Redshift 所採取的動作。

```
select table_id, type, status, eventtime, sequence, previous_state
from SVL_AUTO_WORKER_ACTION;
```

```

table_id | type | status |
eventtime | sequence | previous_state |
-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+
 118082 | sortkey | Start | | 2020-08-22
19:42:20.727049 | 0 | |
 118078 | sortkey | Start | | 2020-08-22
19:43:54.728819 | 0 | |
 118082 | sortkey | Start | | 2020-08-22
19:42:52.690264 | 0 | |
 118072 | sortkey | Start | | 2020-08-22
19:44:14.793572 | 0 | |
 118082 | sortkey | Failed | | 2020-08-22
19:42:20.728917 | 0 | |
 118078 | sortkey | Complete | | 2020-08-22
19:43:54.792705 | 0 | SORTKEY: None;
 118086 | sortkey | Complete | | 2020-08-22
19:42:00.72635 | 0 | SORTKEY: None;
 118082 | sortkey | Complete | | 2020-08-22
19:43:34.728144 | 0 | SORTKEY: None;
 118072 | sortkey | Skipped:Retry exceeds the maximum limit for a table. | | 2020-08-22
19:44:46.706155 | 0 | |
 118086 | sortkey | Start | | 2020-08-22
19:42:00.685255 | 0 | |
 118082 | sortkey | Start | | 2020-08-22
19:43:34.69531 | 0 | |
 118072 | sortkey | Start | | 2020-08-22
19:44:46.703331 | 0 | |
 118082 | sortkey | Checkpoint: progress 14.755079% | | 2020-08-22
19:42:52.692828 | 0 | |
 118072 | sortkey | Failed | | 2020-08-22
19:44:14.796071 | 0 | |
 116723 | sortkey | Abort:This table is not AUTO. | | 2020-10-28
05:12:58.479233 | 0 | |
 110203 | distkey | Abort:This table is not AUTO. | | 2020-10-28
05:45:54.67259 | 0 | |

```

SVL_COMPILE

記錄查詢之每個查詢區段的編譯時間和位置。

所有使用者都可看見 SVL_COMPILE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

SVL_COMPILE 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 [SYS_QUERY_HISTORY](#)。SYS 監視檢視中的資料會格式化為更易於使用和理解。

如需 SVCS_COMPILE 的相關資訊，請參閱 [SVCS_COMPILE](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	與陳述式相關聯的處理程序 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
segment	integer	要編譯的查詢區段。
locus	integer	執行區段的位置。 1 (若在運算節點上) 和 2 (若在領導節點上)。
starttime	timestamp	編譯開始的時間，以 UTC 表示。
endtime	timestamp	編譯結束的時間，以 UTC 表示。
compile	integer	0 表示編譯是否重新使用， 1 表示區段是否編譯。

範例查詢

在此範例中，查詢 35878 和 35879 執行相同的 SQL 陳述式。查詢 35878 的編譯欄位針對四個查詢區段顯示 1，此值表示該區段已編譯。查詢 35879 在每個區段的編譯欄位中顯示 0，表示該區段不需要重新編譯。

```
select userid, xid, pid, query, segment, locus,
```

```
datediff(ms, starttime, endtime) as duration, compile
from svl_compile
where query = 35878 or query = 35879
order by query, segment;
```

userid	xid	pid	query	segment	locus	duration	compile
100	112780	23028	35878	0	1	0	0
100	112780	23028	35878	1	1	0	0
100	112780	23028	35878	2	1	0	0
100	112780	23028	35878	3	1	0	0
100	112780	23028	35878	4	1	0	0
100	112780	23028	35878	5	1	0	0
100	112780	23028	35878	6	1	1380	1
100	112780	23028	35878	7	1	1085	1
100	112780	23028	35878	8	1	1197	1
100	112780	23028	35878	9	2	905	1
100	112782	23028	35879	0	1	0	0
100	112782	23028	35879	1	1	0	0
100	112782	23028	35879	2	1	0	0
100	112782	23028	35879	3	1	0	0
100	112782	23028	35879	4	1	0	0
100	112782	23028	35879	5	1	0	0
100	112782	23028	35879	6	1	0	0
100	112782	23028	35879	7	1	0	0
100	112782	23028	35879	8	1	0	0
100	112782	23028	35879	9	2	0	0

(20 rows)

SVL_DATASHARE_CHANGE_LOG

記錄用於追蹤生產者和消費者叢集上資料共用變更的合併檢視。

所有使用者都可看見 SVL_DATASHARE_CHANGE_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_DATASHARE_CHANGE_LOG](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	採取動作之使用者的 ID。
使用者名稱	varchar(128)	採取動作的使用者名稱。
pid	integer	程序的 ID。
xid	bigint	交易的 ID。
share_id	integer	受影響資料共用的 ID。
share_name	varchar(128)	資料共用的名稱。
source_database_id	integer	資料共用所屬資料庫的 ID。
source_database_name	varchar(128)	資料共用所屬資料庫的名稱。
consumer_database_id	integer	從資料共用匯入之資料庫的 ID。
consumer_database_name	varchar(128)	從資料共用匯入之資料庫的名稱。
arn	varchar(192)	支援匯入資料庫之資源的 ARN。
recordtime	timestamp	動作的時間戳記。
動作	varchar(128)	正在執行的動作。可能的值包括創建數據保護，刪除數據清理，授予改變，撤銷改變，授予共享，撤銷份額，更改添加，更改刪除，更改集

欄名稱	資料類型	描述
		合，授予使用情況，撤銷使用情況，創建數據庫，授予或撤銷共享數據庫的使用量，刪除共享數據庫，改變共享數據庫。
status	integer	動作的狀態。可能的值為 SUCCESS 和 ERROR-ERROR CODE。
share_object_type	varchar(64)	新增至資料共用或從資料共用中移除之資料庫物件的類型。可能的值包括 schemas、tables、columns、functions 和 views。這是生產者叢集的欄位。
share_object_id	integer	新增至資料共用或從資料共用中移除之資料庫物件的 ID。這是生產者叢集的欄位。
share_object_name	varchar(128)	新增至資料共用或從資料共用中移除之資料庫物件的名稱。這是生產者叢集的欄位。
target_user_type	varchar(16)	授與權限的使用者或群組類型。這是生產者和消費者叢集的欄位。
target_userid	integer	授與權限的使用者或群組 ID。這是生產者和消費者叢集的欄位。
target_username	varchar(128)	授與權限的使用者或群組名稱。這是生產者和消費者叢集的欄位。
consumer_account	varchar(16)	資料消費者的帳戶 ID。這是生產者叢集的欄位。
consumer_namespace	varchar(64)	資料消費者帳戶的命名空間。這是生產者叢集的欄位。
producer_account	varchar(16)	資料共用所屬生產者帳戶的帳戶 ID。這是消費者叢集的欄位。
producer_namespace	varchar(64)	資料共用所屬產品帳戶的命名空間。這是消費者叢集的欄位。
attribute_name	varchar(64)	資料共用或共用資料庫的屬性名稱。

欄名稱	資料類型	描述
attribute_value	varchar(128)	資料共用或共用資料庫的屬性值。
message	varchar(512)	動作失敗時的錯誤訊息。

範例查詢

下列範例顯示 SVL_DATASHARE_CHANGE_LOG 檢視。

```
SELECT DISTINCT action
FROM svl_datashare_change_log
WHERE share_object_name LIKE 'tickit%';
```

```
      action
-----
"ALTER DATASHARE ADD"
```

SVL_DATASHARE_CROSS_REGION_USAGE

使用 SVL_DATASHARE_CROSS_REGION_USAGE 檢視，可取得跨區域資料共用查詢所造成之跨區域資料傳輸使用量的摘要。SVL_DATASHARE_CROSS_REGION_USAGE 彙總區段層級的詳細資料。

所有使用者都可看見 SVL_DATASHARE_CROSS_REGION_USAGE。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_DATASHARE_CROSS_REGION_USAGE](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢的 ID。使用此值來聯結各種其他系統資料表與檢視。
segment	bigint	區段的號碼。查詢包含多個區段，每個區段包含一或多個步驟。
start_time	time	資料傳輸開始的時間 (以 UTC 為單位)。
end_time	time	資料傳輸結束的時間 (以 UTC 為單位)。
transferred_data	bigint	從生產者區域傳輸至消費者區域的資料位元組數量。
source_region	char(25)	查詢傳輸資料的來源生產者區域。
recordtime	timestamp	記錄動作的時間。

範例查詢

下列範例顯示 SVL_DATASHARE_CROSS_REGION_USAGE 檢視。

```
SELECT query, segment, transferred_data, source_region
from svl_datashare_cross_region_usage
where query = pg_last_query_id()
order by query, segment;
```

```
query | segment | transferred_data | source_region
-----+-----+-----+-----
200048 | 2 | 4194304 | us-west-1
200048 | 2 | 4194304 | us-east-2
```

SVL_DATASHARE_USAGE_CONSUMER

記錄資料共用的活動和使用情況。此檢視僅與消費者叢集相關。

所有使用者都可看見 SVL_DATASHARE_USAGE_CONSUMER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_DATASHARE_USAGE_CONSUMER](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	發出要求的使用者 ID。
pid	integer	執行查詢之領導者程序的 ID。
xid	bigint	目前交易的內容。
request_id	varchar(50)	要求之 API 呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
transaction_uid	varchar(50)	交易的唯一 ID。
recordtime	timestamp	記錄動作的時間。
status	integer	要求的 API 呼叫的狀態。
error	varchar(512)	錯誤的訊息。

範例查詢

下列範例顯示 SVL_DATASHARE_USAGE_CONSUMER 檢視。

```
SELECT request_type, status, trim(error) AS error
FROM svl_datashare_usage_consumer
```

```

 request_type | status | error
-----+-----+-----
 "GET RELATION" | 0      |

```

SVL_DATASHARE_USAGE_PRODUCER

記錄資料共用的活動和使用情況。此檢視僅與生產者叢集相關。

所有使用者都可看見 SVL_DATASHARE_USAGE_PRODUCER。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_DATASHARE_USAGE_PRODUCER](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
share_id	integer	資料共用的物件 ID (OID)。
share_name	varchar(128)	資料共用的名稱。
request_id	varchar(50)	要求之 API 呼叫的唯一 ID。
request_type	varchar(25)	對生產者叢集發出的要求類型。
object_type	varchar(64)	從資料共用共用的物件類型。可能的值包括 schemas、tables、columns、functions 和 views。
object_oid	integer	從資料共用共用的物件 ID。
object_name	varchar(128)	要從資料共用共用的物件名稱。
consumer_account	varchar(16)	資料共用共用到的消費者帳戶的帳戶。
consumer_namespace	varchar(64)	資料共用共用到的使用者帳戶的命名空間。

欄名稱	資料類型	描述
consumer_transaction_uid	varchar(50)	消費者叢集上陳述式的唯一交易 ID。
recordtime	timestamp	記錄動作的時間。
status	integer	資料共用的狀態。
error	varchar(512)	錯誤的訊息。
consumer_region	char(64)	取用者叢集所在的區域。

範例查詢

下列範例顯示 SVL_DATASHARE_USAGE_PRODUCER 檢視。

```
SELECT DISTINCT request_type
FROM svl_datashare_usage_producer
WHERE object_name LIKE 'tickit%';
```

```
request_type
```

```
-----
```

```
"GET RELATION"
```

SVL_FEDERATED_QUERY

使用 SVL_FEDERATED_QUERY 檢視，可檢視有關聯合查詢呼叫的資訊。

所有使用者都可看見 SVL_FEDERATED_QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_EXTERNAL_QUERY_DETAIL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行查詢的使用者 ID。
xid	bigint	交易 ID。
pid	integer	執行查詢之領導者程序的 ID。
query	integer	聯合呼叫的查詢 ID。
sourcetype	character (32)	聯合呼叫來源類型，例如 "PG"。
recordtime	timestamp	傳送查詢以進行聯合的時間。使用 UTC。
querytext	character (4000)	傳送到遠端 PostgreSQL 引擎以執行的查詢字串。
num_rows	bigint	聯合查詢傳回的資料列數。
num_bytes	bigint	聯合查詢所傳回的位元組數量。
duration	bigint	從游標呼叫擷取資料列所花費的時間 (微秒)。這是執行聯合查詢所花費的時間，以及獲取結果。

範例查詢

若要顯示聯合查詢呼叫的相關資訊，請執行下列查詢。

```
select query, trim(sourcetype) as type, recordtime, trim(querytext) as "PG Subquery"
from svl_federated_query where query = 4292;
```

```
query | type | recordtime | pg subquery
```



```
-----+-----+-----  
+-----  
4292 | PG | 2020-03-27 04:29:58.485126 | SELECT "level" FROM functional.employees  
WHERE ("level" >= 6)  
(1 row)
```

SVL_MULTI_STATEMENT_VIOLATIONS

使用 SVL_MULTI_STATEMENT_VIOLATIONS 檢視來取得系統上執行的所有違反交易區塊限制的 SQL 命令的完整記錄。

當您執行 Amazon Redshift 限制在交易區塊或多重陳述式請求內的下列任何 SQL 命令時，就會發生違規：

- [CREATE DATABASE](#)
- [DROP DATABASE](#)
- [ALTER TABLE APPEND](#)
- [CREATE EXTERNAL TABLE](#)
- DROP EXTERNAL TABLE
- RENAME EXTERNAL TABLE
- ALTER EXTERNAL TABLE
- CREATE TABLESPACE
- DROP TABLESPACE
- [CREATE LIBRARY](#)
- [DROP LIBRARY](#)
- REBUILD CAT
- INDEX CAT
- REINDEX DATABASE
- [VACUUM](#)
- [GRANT](#)
- [COPY](#)

Note

如果此檢視中有任何項目，請變更對應的應用程式和 SQL 命令檔。我們建議您變更應用程式程式碼，將這些受限制 SQL 命令的使用移至交易區塊之外。如果您需要進一步協助，請聯絡 Sup AWS port 部門。

所有使用者都可看見 SVL_MULTI_STATEMENT_VIOLATIONS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_HISTORY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	造成違規的使用者 ID。
database	character(32)	使用者連接的資料庫名稱。
cmdname	character(20)	無法在交易區塊或多重陳述式要求內執行的命令名稱。例如，CREATE DATABASE、DROP DATABASE、ALTER TABLE APPEND、CREATE EXTERNAL TABLE、DROP EXTERNAL TABLE、RENAME EXTERNAL TABLE、ALTER EXTERNAL TABLE、CREATE LIBRARY、DROP LIBRARY、REBUILD CAT、INDEX CAT、REINDEX DATABASE、VACUUM、外部資源上的 GRANT、CLUSTER、COPY、CREATE TABLESPACE 以及 DROP TABLESPACE。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	陳述式的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。

欄名稱	資料類型	描述
starttime	timestamp	陳述式開始執行的確切時間，精確度為六位數的小數秒，例如： 2009-06-12 11:29:19.131358
endtime	timestamp	陳述式完成執行的確切時間，精確度為六位數的小數秒，例如： 2009-06-12 11:29:19.193640
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
type	varchar(10)	SQL 陳述式的類型： QUERY 、 DDL 或 UTILITY 。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回多個有違規的陳述式。

```
select * from svl_multi_statement_violations order by starttime asc;

userid | database | cmdname | xid | pid | label | starttime | endtime | sequence | type
| text
=====
1 | dev | CREATE DATABASE | 1034 | 5729 | label1 | ***** | ***** | 0 | DDL |
create table c(b int);
1 | dev | CREATE DATABASE | 1034 | 5729 | label1 | ***** | ***** | 0 | UTILITY |
create database b;
1 | dev | CREATE DATABASE | 1034 | 5729 | label1 | ***** | ***** | 0 | UTILITY |
COMMIT
...
```

SVL_MV_REFRESH_STATUS

針對具體化視觀表的重新整理活動，SVL_MV_REFRESH_STATUS 檢視會包含一個資料列。

如需具體化視觀表的相關資訊，請參閱 [Amazon Redshift 中的具體化視觀表](#)。

所有使用者都可看見 SVL_MV_REFRESH_STATUS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_MV_REFRESH_HISTORY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
db_name	char(128)	包含具體化視觀表的資料庫。
userid	bigint	執行重新整理的使用者 ID。
schema_name	char(128)	具體化視觀表的結構描述。
mv_name	char(128)	具體化視觀表名稱。
xid	bigint	重新整理的交易 ID。
starttime	timestamp	重新整理的開始時間。
endtime	timestamp	重新整理的結束時間。
status	text	<p>重新整理的狀態。範例值包括下列各項：</p> <ul style="list-style-type: none"> 重新整理以累加方式成功更新 MV <p>如果它是用於串流的具體化視觀表，則訊息可能會有關於記錄數目的其他限定詞。這些索引標籤包括以下項目：</p> <ul style="list-style-type: none"> 串流未傳回任何新資料 - 未擷取任何記錄。 從串流收到的所有記錄被略過 - 已擷取記錄，但由於錯誤，所有記錄都已略過。 一些串流記錄被略過-記錄被檢索，但由於錯誤，有些被略過。 <p>如果沒有限定元，則至少會擷取一筆記錄，且具體化視觀表中的所有記錄都可用。還有一個可能的限定詞：</p>

欄名稱	資料類型	描述
		<ul style="list-style-type: none"> 串流可能包含更多資料 - 在 Amazon Redshift 判斷沒有進一步的記錄可供使用之前，重新整理即結束。該串流可能是最新的，但尚未得到 Amazon Redshift 的確認。 重新整理成功從頭重新運算 MV 重新整理以累加方式將 MV 部分更新至作用中交易 已經更新 MV 重新整理失敗。已重新命名基底資料表資料欄 重新整理失敗。已變更基底資料表資料欄類型 重新整理失敗。已重新命名基底資料表 內部錯誤導致的重新整理失敗。 重新整理失敗。已捨棄基底資料表資料欄 重新整理失敗。已重新命名 MV 的結構描述 重新整理失敗。找不到 MV 自動重新整理因使用者工作負載過多而中止 重新整理失敗。可序列化隔離違規
refresh_type	char(32)	重新整理類型的定義。範例值包括手動和自動。

範例查詢

若要檢視具體化視觀表的重新整理狀態，請執行下列查詢。

```
select * from svl_mv_refresh_status;
```

此查詢傳回下列範例輸出：

```
db_name | userid | schema | name | xid | starttime |      |
         |        |        |      |    |           |      |
         |        |        |      |    |           |      |
refresh_type
```

```

-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
+-----
dev      |    169 | mv_schema | mv_test | 6640 | 2020-02-14 02:26:53.497935 |
2020-02-14 02:26:53.556156 | Refresh successfully recomputed MV from scratch |
Manual
dev      |    166 | mv_schema | mv_test | 6517 | 2020-02-14 02:26:39.287438 |
2020-02-14 02:26:39.349539 | Refresh successfully updated MV incrementally |
Auto
dev      |    162 | mv_schema | mv_test | 6388 | 2020-02-14 02:26:27.863426 |
2020-02-14 02:26:27.918307 | Refresh successfully recomputed MV from scratch |
Manual
dev      |    161 | mv_schema | mv_test | 6323 | 2020-02-14 02:26:20.020717 |
2020-02-14 02:26:20.080002 | Refresh successfully updated MV incrementally |
Auto
dev      |    161 | mv_schema | mv_test | 6301 | 2020-02-14 02:26:05.796146 |
2020-02-14 02:26:07.853986 | Refresh successfully recomputed MV from scratch |
Manual
dev      |    153 | mv_schema | mv_test | 6024 | 2020-02-14 02:25:18.762335 |
2020-02-14 02:25:20.043462 | MV was already updated |
Manual
dev      |    143 | mv_schema | mv_test | 5557 | 2020-02-14 02:24:23.100601 |
2020-02-14 02:24:23.100633 | MV was already updated |
Manual
dev      |    141 | mv_schema | mv_test | 5447 | 2020-02-14 02:23:54.102837 |
2020-02-14 02:24:00.310166 | Refresh successfully updated MV incrementally |
Auto
dev      |     1  | mv_schema | mv_test | 5329 | 2020-02-14 02:22:26.328481 |
2020-02-14 02:22:28.369217 | Refresh successfully recomputed MV from scratch |
Auto
dev      |    138 | mv_schema | mv_test | 5290 | 2020-02-14 02:21:56.885093 |
2020-02-14 02:21:56.885098 | Refresh failed. MV was not found |
Manual

```

SVL_QERROR

SVL_QERROR 檢視已棄用。

SVL_QLOG

SVL_QLOG 檢視包含針對資料庫執行之所有查詢的記錄。

Amazon Redshift 會從 [STL_QUERY](#) 資料表以可讀資訊子集的形式建立 SVL_QLOG 檢視。使用此資料表來找出最近執行之查詢的查詢 ID，或查看完成查詢需要多長時間。

所有使用者都可看見 SVL_QLOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_HISTORY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目的使用者之 ID。
query	integer	查詢 ID。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
xid	bigint	交易 ID。
pid	integer	與查詢相關聯的處理程序 ID。
starttime	timestamp	陳述式開始執行的確切時間，精確度為六位數的小數秒，例如： 2009-06-12 11:29:19.131358
endtime	timestamp	陳述式完成執行的確切時間，精確度為六位數的小數秒，例如： 2009-06-12 11:29:19.193640
elapsed	bigint	查詢執行的所需時間 (微秒)。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 1 。如果查詢執行至完成，欄位會包含 0 。因工作負載管理目的而取消且後續會再重新啟動的查詢也會在此欄位中擁有值 1 。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為 default。
substring	character(60)	遭截斷的查詢文字。

欄名稱	資料類型	描述
source_query	integer	如果查詢使用了結果快取，則為快取結果來源之查詢的查詢 ID。如果未使用結果快取，此欄位的值為 NULL。
concurrency_scaling_status_txt	text	查詢是執行於主要叢集或並行擴展叢集的描述。
from_sp_call	integer	如果是從預存程序呼叫查詢，則為程序呼叫的查詢 ID。如果查詢不是在預存程序中執行，則此欄位為 NULL。

範例查詢

下列範例傳回使用者 `userid = 100` 執行之最近五個資料庫查詢的查詢 ID、執行時間和遭截斷的查詢文字。

```
select query, pid, elapsed, substring from svl_qlog
where userid = 100
order by starttime desc
limit 5;
```

```
query | pid | elapsed | substring
-----+-----+-----+-----
187752 | 18921 | 18465685 | select query, elapsed, substring from svl_...
204168 | 5117 | 59603 | insert into testtable values (100);
187561 | 17046 | 1003052 | select * from pg_table_def where tablename...
187549 | 17046 | 1108584 | select * from STV_WLM_SERVICE_CLASS_CONFIG
187468 | 17046 | 5670661 | select * from pg_table_def where schemaname...
(5 rows)
```

下列範例會傳回 SQL 指令碼名稱 (LABEL 欄位) 和已取消查詢的歷經時間 (**aborted=1**)：

```
select query, elapsed, trim(label) querylabel
from svl_qlog where aborted=1;
```

```
query | elapsed | querylabel
-----+-----+-----
16 | 6935292 | alltickittablesjoin.sql
```


(1 row)

SVL_QUERY_METRICS

SVL_QUERY_METRICS 檢視顯示已完成之查詢的指標。此檢視衍生自 [STL_QUERY_METRICS](#) 系統資料表。請使用此檢視中的值來協助決定臨界值，以定義查詢監控規則。如需詳細資訊，請參閱 [WLM 查詢監控規則](#)。

所有使用者都可看見 SVL_QUERY_METRICS。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_DETAIL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
service_class	integer	WLM 查詢佇列 ID (服務類別)。查詢佇列定義於 WLM 組態。只會回報使用者定義之佇列的指標。如需服務類別 ID 的清單，請參閱 WLM 服務類別 IDs 。
dimension	varchar(24)	指標回報的維度。可能的值為查詢、區段和步驟。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。如果區段值為 0，指標區段值將累計至查詢層級。
step	integer	所執行步驟類型的 ID。步驟類型的說明會在 step_label 欄位中顯示。
step_label	varchar(30)	執行的步驟類型。
query_cpu_time	bigint	查詢所用的 CPU 時間 (秒)。CPU 時間與查詢執行時間不同。
query_blocks_read	bigint	查詢所讀取的 1 MB 區塊數。

欄名稱	資料類型	描述
query_execution_time	bigint	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。如需排入佇列的時間，請參閱 query_queue_time 。
query_cpu_usage_percent	bigint	查詢所用的 CPU 容量百分比。
query_temp_blocks_to_disk	bigint	查詢用來寫入中繼結果之磁碟空間量 (MB)。
segment_execution_time	bigint	單一區段經過的執行時間 (秒)。
cpu_skew	numeric(38,2)	任何分割 CPU 用量與所有分割之平均 CPU 用量的比率。此指標在區段層級中定義。
io_skew	numeric(38,2)	任何分割讀取區塊數上限 (I/O) 與所有分割平均讀取區塊數的比率。
scan_row_count	bigint	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。
join_row_count	bigint	聯結步驟中處理的資料列數。
nested_loop_join_row_count	bigint	巢狀迴圈聯結中的資料列數。
return_row_count	bigint	查詢傳回的資料列數。
spectrum_scan_row_count	bigint	Amazon S3 中的 Amazon Redshift Spectrum 掃描的列數。
spectrum_scan_size_mb	bigint	Amazon S3 中 Amazon Redshift Spectrum 掃描的資料量 (MB)。

欄名稱	資料類型	描述
query_queue_time	bigint	查詢排入佇列的時間 (秒)。

SVL_QUERY_METRICS_SUMMARY

SVL_QUERY_METRICS_SUMMARY 檢視顯示已完成之查詢的指標最大值。此檢視衍生自 [STL_QUERY_METRICS](#) 系統資料表。請使用此檢視中的值來協助決定臨界值，以定義查詢監控規則。如需 Amazon Redshift 查詢監控之規則和指標的相關資訊，請參閱 [WLM 查詢監控規則](#)。

所有使用者都可看見 SVL_QUERY_METRICS_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_DETAIL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行產生此項目之查詢的使用者 ID。
query	integer	查詢 ID。查詢欄可用於加入其他系統表格與檢視。
service_class	integer	WLM 查詢佇列 ID (服務類別)。查詢佇列定義於 WLM 組態。只會回報使用者定義之佇列的指標。如需服務類別 ID 的清單，請參閱 WLM 服務類別 IDs 。
query_cpu_time	bigint	查詢所用的 CPU 時間 (秒)。CPU 時間與查詢執行時間不同。
query_blocks_read	bigint	查詢所讀取的 1 MB 區塊數。
query_execution_time	bigint	查詢經過的執行時間 (秒)。執行時間不包括在佇列中等待所花的時間。
query_cpu_usage_percent	numeric(3,2)	查詢所用的 CPU 容量百分比。

欄名稱	資料類型	描述
query_temp_blocks_to_disk	bigint	查詢用來寫入中繼結果之磁碟空間量 (MB)。
segment_execution_time	bigint	單一區段經過的執行時間 (秒)。
cpu_skew	numeric(38,2)	任何分割 CPU 用量與所有分割之平均 CPU 用量的比率。此指標在區段層級中定義。
io_skew	numeric(38,2)	任何分割讀取區塊數上限 (I/O) 與所有分割平均讀取區塊數的比率。
scan_row_count	bigint	掃描步驟的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。
join_row_count	bigint	聯結步驟中處理的資料列數。
nested_loop_join_row_count	bigint	巢狀迴圈聯結中的資料列數。
return_row_count	bigint	查詢傳回的資料列數。
spectrum_scan_row_count	bigint	Amazon S3 中的 Amazon Redshift Spectrum 掃描的列數。
spectrum_scan_size_mb	bigint	Amazon S3 中 Amazon Redshift Spectrum 掃描的資料量 (MB)。
query_queue_time	bigint	查詢排入佇列的時間 (秒)。

SVL_QUERY_QUEUE_INFO

摘要說明在工作負載管理 (WLM) 查詢佇列或遞交佇列中花費時間之查詢的詳細資訊。

SVL_QUERY_QUEUE_INFO 檢視會篩選系統執行的查詢並僅顯示使用者執行的查詢。

SVL_QUERY_QUEUE_INFO 檢視會摘要從 [STL_QUERY](#)、[STL_WLM_QUERY](#) 和 [STL_COMMIT_STATS](#) 系統資料表中的資訊。

只有超級使用者可以看到 SVL_QUERY_QUEUE_INFO。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
database	text	當查詢發出時，要將使用者連接至其中的資料庫名稱。
query	integer	查詢 ID。
xid	bigint	交易 ID。
userid	integer	產生查詢的使用者之 ID。
querytxt	text	查詢文字的前 100 個字元。
queue_start_time	timestamp	查詢進入 WLM 佇列的時間 (以 UTC 表示)。
exec_start_time	timestamp	查詢執行開始的時間 (以 UTC 表示)。
service_class	integer	服務類別的 ID。服務類別定義於 WLM 組態檔案。
slots	integer	WLM 查詢插槽的數量。
queue_elapsed	bigint	查詢在 WLM 佇列中等待的時間 (秒)。
exec_elapsed	bigint	花費在執行查詢的時間 (秒)。
wlm_total_elapsed	bigint	查詢在 WLM 佇列所花的時間 (queue_elapsed)，加上執行查詢所花的時間 (exec_elapsed)。

欄名稱	資料類型	描述
commit_queue_elapsed	bigint	查詢在遞交佇列中等待的時間 (秒)。
commit_exec_time	bigint	查詢在遞交操作中所花的時間 (秒)。
service_class_name	character(64)	服務類別的名稱。

範例查詢

下列範例為查詢在 WLM 佇列中所花的時間。

```
select query, service_class, queue_elapsed, exec_elapsed, wlm_total_elapsed
from svl_query_queue_info
where wlm_total_elapsed > 0;
```

query	service_class	queue_elapsed	exec_elapsed	wlm_total_elapsed
2742669	6	2	916	918
2742668	6	4	197	201

(2 rows)

SVL_QUERY_REPORT

Amazon Redshift 從各種 Amazon Redshift STL 系統資料表的 UNION 中建立 SVL_QUERY_REPORT 檢視，來提供已完成查詢步驟的資訊。

此檢視會依分割和依步驟來劃分已完成查詢的資訊，此有助於針對在 Amazon Redshift 叢集中的節點和分割問題進行疑難排解。

所有使用者都可看見 SVL_QUERY_REPORT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_DETAIL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
分割	integer	執行步驟的資料分割。
segment	integer	區段號碼。 查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	已完成的查詢步驟。
start_time	timestamp	區段開始執行的確切時間 (以 UTC 表示)，精確度為 6 位數的小數秒。例如： 2012-12-12 11:29:19.131358
end_time	timestamp	區段完成執行的確切時間 (以 UTC 表示)，精確度為 6 位數的小數秒。例如： 2012-12-12 11:29:19.131467
elapsed_time	bigint	區段執行的所需時間 (微秒)。
rows	bigint	步驟產生的資料列數 (每分割)。此數字表示因步驟執行而造成之分割的資料列數，而不是步驟接收或處理的資料列數。也就是說，此為歷經該步驟且傳遞至下一個步驟之資料列數。
bytes	bigint	步驟產生的位元組數 (每分割)。
label	char(256)	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 scan tbl=100448 name =user)。三位數資料表 ID 通常是指暫時性資料表的掃描。當您看見 tbl=0 時，通常是指常數值的掃描。
is_diskbased	character (1)	查詢的這個步驟是否以磁碟型操作方式執行：true (t) 或 false (f)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。

欄名稱	資料類型	描述
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。此值為在執行期間為使用而配置的 query_working_mem 臨界值，而非實際所用的記憶體量
is_rrscan	character (1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。
is_delayed_scan	character (1)	若為 true (t)，表示已在該步驟上使用延遲的掃描。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之後，但在套用使用者定義的查詢篩選條件之前已發出的資料列總數。

範例查詢

下列查詢示範查詢 ID 為 279 的查詢之傳回列的資料扭曲。使用此查詢來決定資料庫資料是否在資料倉儲叢集中的分割中平均分散：

```
select query, segment, step, max(rows), min(rows),
case when sum(rows) > 0
then ((cast(max(rows) -min(rows) as float)*count(rows))/sum(rows))
else 0 end
from svl_query_report
where query = 279
group by query, segment, step
order by segment, step;
```

此查詢應會傳回與下列範例輸出類似的資料：

query	segment	step	max	min	case
279	0	0	19721687	19721687	0
279	0	1	19721687	19721687	0
279	1	0	986085	986084	1.01411202804304e-06
279	1	1	986085	986084	1.01411202804304e-06
279	1	4	986085	986084	1.01411202804304e-06
279	2	0	1775517	788460	1.00098637606408
279	2	2	1775517	788460	1.00098637606408


```

279 |      3 |    0 | 1775517 | 788460 | 1.00098637606408
279 |      3 |    2 | 1775517 | 788460 | 1.00098637606408
279 |      3 |    3 | 1775517 | 788460 | 1.00098637606408
279 |      4 |    0 | 1775517 | 788460 | 1.00098637606408
279 |      4 |    1 | 1775517 | 788460 | 1.00098637606408
279 |      4 |    2 |      1 |      1 | 0
279 |      5 |    0 |      1 |      1 | 0
279 |      5 |    1 |      1 |      1 | 0
279 |      6 |    0 |     20 |     20 | 0
279 |      6 |    1 |      1 |      1 | 0
279 |      7 |    0 |      1 |      1 | 0
279 |      7 |    1 |      0 |      0 | 0
(19 rows)

```

SVL_QUERY_SUMMARY

使用 SVL_QUERY_SUMMARY 檢視來尋找查詢執行的一般相關資訊。

SVL_QUERY_SUMMARY 檢視包含 SVL_QUERY_REPORT 檢視的資料子集。請留意，會從所有節點彙總 SVL_QUERY_SUMMARY 中的資訊。

Note

SVL_QUERY_SUMMARY 檢視僅包含 Amazon Redshift 執行之查詢的相關資訊，不包含其他公用程式和 DDL 命令的相關資訊。如需 Amazon Redshift 所完成全部陳述式的完整清單和相關資訊 (包含 DDL 和公用程式命令)，則可查詢 SVL_STATEMENTTEXT 檢視。

所有使用者都可看見 SVL_QUERY_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_DETAIL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVCS_QUERY_SUMMARY 的相關資訊，請參閱 [SVCS_QUERY_SUMMARY](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。可用於聯結其他系統資料表與檢視。
stm	integer	串流：查詢的並行區段集。一個查詢擁有一或多個串流。
seg	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
step	integer	執行的查詢步驟。
maxtime	bigint	此步驟執行的時間量上限 (微秒)。
avgtime	bigint	此步驟執行的平均時間 (微秒)。
rows	bigint	查詢步驟中包含的資料列數。
位元組	bigint	查詢步驟中包含的資料位元組數。
rate_row	double precision	每資料列的查詢執行率。
rate_byte	double precision	每位元組的查詢執行率。
label	text	步驟標籤，包含查詢步驟名稱與資料表 ID 及資料表名稱 (如適用) (例如 scan tbl=100448 name =user)。三位數資料表 ID 通常是指暫時性資料表的掃描。當您看見 tbl=0 時，通常是指常數值的掃描。
is_diskbased	character(1)	此查詢步驟在叢集的任何節點上是否以磁碟型操作方式執行：true (t) 或 false (f)。只有特定步驟會進入磁碟，例如雜湊、排序及彙總步驟。許多步驟類型一律在記憶體中執行。
workmem	bigint	已指派給查詢步驟之運作中記憶體數 (位元組)。
is_rrscan	character(1)	若為 true (t)，表示已在步驟上使用範圍限制掃描。預設為 false (f)。

欄名稱	資料類型	描述
is_delayed_scan	character(1)	若為 true (t)，表示已在該步驟上使用延遲的掃描。預設為 false (f)。
rows_pre_filter	bigint	對於永久資料表的掃描，指的是在篩選標記進行刪除的資料列 (幽靈資料列) 之前已發出的資料列總數。

範例查詢

檢視查詢步驟的處理資訊

下列查詢顯示查詢 87 每個步驟的基本處理資訊：

```
select query, stm, seg, step, rows, bytes
from svl_query_summary
where query = 87
order by query, seg, step;
```

此查詢會擷取查詢 87 的相關處理資訊，如下範例輸出所示：

```
query | stm | seg | step | rows | bytes
-----+-----+-----+-----+-----+-----
87    | 0   | 0   | 0   | 90   | 1890
87    | 0   | 0   | 2   | 90   | 360
87    | 0   | 1   | 0   | 90   | 360
87    | 0   | 1   | 2   | 90   | 1440
87    | 1   | 2   | 0   | 210494 | 4209880
87    | 1   | 2   | 3   | 89500 | 0
87    | 1   | 2   | 6   | 4     | 96
87    | 2   | 3   | 0   | 4     | 96
87    | 2   | 3   | 1   | 4     | 96
87    | 2   | 4   | 0   | 4     | 96
87    | 2   | 4   | 1   | 1     | 24
87    | 3   | 5   | 0   | 1     | 24
87    | 3   | 5   | 4   | 0     | 0
(13 rows)
```

判斷查詢步驟是否溢出至磁碟

下列查詢顯示查詢 ID 為 1025 的任何查詢步驟 (請參閱 [SVL_QLOG](#) 檢視來了解如何取得查詢的查詢 ID) 是否溢出至磁碟或查詢完全在記憶體內執行：

```
select query, step, rows, workmem, label, is_diskbased
from svl_query_summary
where query = 1025
order by workmem desc;
```

此查詢傳回下列範例輸出：

```
query| step|  rows  |  workmem  |  label          |  is_diskbased
-----+-----+-----+-----+-----+-----
1025 |  0  |16000000| 141557760 |scan tbl=9      | f
1025 |  2  |16000000| 135266304 |hash tbl=142    | t
1025 |  0  |16000000| 128974848 |scan tbl=116536| f
1025 |  2  |16000000| 122683392 |dist            | f
(4 rows)
```

透過掃描 IS_DISKBASED 的值，您可以檢視哪一個查詢步驟使用磁碟。針對查詢 1025，雜湊步驟在磁碟上執行。步驟可能在包含雜湊、彙總或排序步驟的磁碟上執行。若要只檢視磁碟型的查詢步驟，將 **and is_diskbased = 't'** 子句新增至以上範例的 SQL 陳述式。

SVL 還原 _ 更改表 _ 進度

在將傳統調整為 RA3 節點的大小時，使用 SVL_RESTORE_ALTER_TABLE_PROGRESS 來監視叢集中每個表格的移轉進度。它會在調整大小作業期間擷取資料遷移的歷史輸送量。有關經典調整為 RA3 節點的更多信息，請轉到[經典調整大小](#)。

SVL_ 還原 _ 更改表格 _ 進度僅對超級使用者可見。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_RESTORE_LOG](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

Note

進度為100.00%或ABORTED在 7 天後刪除的資料列。在傳統重新調整大小期間或之後放置的表格資料列仍可顯示在 SVL_RESTORE_ALTER_TABLE_PROGRESS 中。

資料表欄

欄名稱	資料類型	描述
tbl	integer	資料表的 ID。
進度	char(32)	資料表重新分佈進度的狀態。可能的值為「0.00%到」100.00% 和「訊息」的百分比ABORTED。ABORTED意味著重新分配在沒有完成的情況下停止，原因已在message列中解釋。
message	char(256)	與資料表重新發佈進度相關聯的訊息。

範例查詢

下列查詢會傳回執行中和佇列中的查詢。


```
select * from svl_restore_alter_table_progress;
```

```
tbl      | progress | message
-----+-----+-----
105614   | ABORTED  | Abort:Table no longer contains the prior dist key column.
105610   | ABORTED  | Abort:Table no longer contains the prior dist key column.
105594   | 0.00%    | Table waiting for alter diststyle conversion.
105602   | ABORTED  | Abort:Table no longer contains the prior dist key column.
105606   | ABORTED  | Abort:Table no longer contains the prior dist key column.
105598   | 100.00%  | Restored to distkey successfully.
```

SVL_S3LIST

使用 SVL_S3LIST 檢視以取得有關區段層級的 Amazon Redshift Spectrum 查詢詳細資訊。

所有使用者都可看見 SVL_S3LIST。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

 Note

SVL_S3LIST 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 [SYS_EXTERNAL_QUERY_DETAIL](#)。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。一個查詢包含多個區段。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
eventtime	timestamp	記錄事件的時間 (以 UTC 表示)。
儲存貯體	text	Amazon S3 儲存貯體名稱。
字首	text	Amazon S3 儲存貯體位置的字首。
recursive	char(1)	是否對子資料夾遞迴掃描。
retrieved_files	integer	所列檔案的數量。
max_file_size	bigint	所列檔案的檔案大小上限。
avg_file_size	double precision	所列檔案的平均檔案大小上限。
generated_splits	integer	檔案分割的數量。
avg_split_length	double precision	檔案分割的平均長度 (位元組)。
duration	bigint	檔案清單的期間 (微秒)。

範例查詢

以下範例會查詢前次執行查詢的 SVL_S3LIST。

```
select *
from svl_s3list
where query = pg_last_query_id()
order by query,segment;
```

SVL_S3LOG

使用 SVL_S3LOG 檢視以取得有關區段和節點分割層級的 Amazon Redshift Spectrum 查詢詳細資訊。

所有使用者都可看見 SVL_S3LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

SVL_S3LOG 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 [SYS_EXTERNAL_QUERY_DETAIL](#)。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
pid	integer	處理程序 ID。
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
eventtime	timestamp	步驟開始執行的時間 (以 UTC 表示)。
message	text	日誌項目訊息。

範例查詢

以下範例會查詢前次執行查詢的 SVL_S3LOG。

```
select *
from svl_s3log
where query = pg_last_query_id()
order by query,segment,slice;
```

SVL_S3PARTITION

使用 SVL_S3PARTITION 檢視以取得有關區段和節點分割層級的 Amazon Redshift Spectrum 分割區詳細資訊。

所有使用者都可看見 SVL_S3PARTITION。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

SVL_S3PARTITION 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 [SYS_EXTERNAL_QUERY_DETAIL](#)。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
starttime	沒有時區的時間戳記	分割區清除開始執行的時間 (以 UTC 表示)。
endtime	沒有時區的時間戳記	分割區清除完成的時間 (以 UTC 表示)。

欄名稱	資料類型	描述
duration	bigint	經過時間 (微秒)。
total_partitions	integer	分割區總數。
qualified_partitions	integer	合格分割區數。
assigned_partitions	integer	分割上指定分割區數。
指派	character	指派的類型。

範例查詢

以下範例會取得前次完成查詢的分割區詳細資訊。

```
SELECT query, segment,
       MIN(starttime) AS starttime,
       MAX(endtime) AS endtime,
       datediff(ms,MIN(starttime),MAX(endtime)) AS dur_ms,
       MAX(total_partitions) AS total_partitions,
       MAX(qualified_partitions) AS qualified_partitions,
       MAX(assignment) as assignment_type
FROM svl_s3partition
WHERE query=pg_last_query_id()
GROUP BY query, segment
```

```
query | segment |                starttime                |                endtime                | dur_ms |
total_partitions | qualified_partitions | assignment_type
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
99232 |      0 | 2018-04-17 22:43:50.201515 | 2018-04-17 22:43:54.674595 | 4473 |
      2526 |      334 | p
```

SVL_S3PARTITION_SUMMARY

使用 SVL_S3PARTITION_SUMMARY 檢視來取得區段層級的 Redshift Spectrum 查詢分割處理的摘要。

所有使用者都可看見 SVL_S3PARTITION_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

如需 SVCS_S3PARTITION 的相關資訊，請參閱 [SVCSSUMMARYPARTITION](#)。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
segment	integer	區段號碼。一個查詢包含多個區段。
指派	char(1)	節點間分割指派的類型。
min_start_time	timestamp	分割區處理開始的時間 (以 UTC 表示)。
max_endtime	timestamp	分割區處理完成的時間 (以 UTC 表示)。
min_duration	bigint	節點針對此查詢所使用的最小分割處理時間 (微秒)。
max_duration	bigint	節點針對此查詢所使用的最大分割處理時間 (微秒)。
avg_duration	bigint	節點針對此查詢所使用的平均分割處理時間 (微秒)。
total_partitions	integer	外部資料表中分割區的總數。
qualified_partitions	integer	合格分割區總數。

欄名稱	資料類型	描述
min_assigned_partitions	integer	在一個節點上指派的最小分割區數量。
max_assigned_partitions	integer	在一個節點上指派的最高分割區數量。
avg_assigned_partitions	bigint	在一個節點上指派的平均分割區數量。

範例查詢

以下範例會取得前次完成查詢的分割區掃描詳細資訊。

```
select query, segment, assignment, min_starttime, max_endtime, min_duration,
       avg_duration
from svl_s3partition_summary
where query = pg_last_query_id()
order by query, segment;
```

SVL_S3QUERY

使用 SVL_S3QUERY 檢視以取得有關區段和節點分割層級的 Amazon Redshift Spectrum 查詢詳細資訊。

所有使用者都可看見 SVL_S3QUERY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

Note

SVL_S3QUERY 僅包含在主叢集上執行的查詢。但不包含在並行擴縮叢集上執行的查詢。若要存取在主要和並行擴縮叢集上執行的查詢，建議您使用 SYS 監視檢視 [SYS_EXTERNAL_QUERY_DETAIL](#)。SYS 監視檢視中的資料會格式化為更易於使用和理解。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生給定項目的使用者 ID。
query	integer	查詢 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
starttime	timestamp	查詢開始執行的時間 (以 UTC 表示)。
endtime	timestamp	查詢執行完成的時間 (以 UTC 表示)
elapsed	integer	經過時間 (微秒)。
external_table_name	char(136)	s3 掃描步驟之外部資料表名稱的內部格式。
is_partitioned	char(1)	若為 true (t)，此欄位值表示外部資料表已遭分割。
is_rrscan	char(1)	若為 true (t)，此欄位值表示已套用範圍限定的掃描。
s3_scanned_rows	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的列數。
s3_scanned_bytes	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的位元組數。
s3query_returned_rows	bigint	從 Redshift Spectrum 層傳回至叢集的資料列數。

欄名稱	資料類型	描述
s3query_returned_bytes	bigint	從 Redshift Spectrum 層傳回至叢集的位元組數。
files	integer	在此分割上此 S3 掃描步驟已處理的檔案數。
splits	int	在此分割上處理的分割數。有了大型可分割資料檔案 (例如, 大於 512 MB 的資料檔案), Redshift Spectrum 會嘗試將檔案分割至多個 S3 請求來進行平行處理。
total_split_size	bigint	在此分割上處理的所有分割大小總計 (位元組)。
max_split_size	bigint	為此分割處理的分割大小上限 (位元組)。
total_retries	integer	處理檔案的重試數總計。
max_retries	integer	個別處理檔案的重試數上限。
max_request_duration	integer	個別 Redshift Spectrum 請求的最大持續期間 (微秒)。
avg_request_duration	double precision	Redshift Spectrum 請求的平均持續時間 (微秒)。
max_request_parallelism	integer	在此 S3 掃描步驟的分割上未完成 Redshift Spectrum 數上限。
avg_request_parallelism	double precision	在此 S3 掃描步驟的分割上平行 Redshift Spectrum 請求數平均。

範例查詢

以下範例會取得前次完成查詢的掃描步驟詳細資訊。

```
select query, segment, slice, elapsed, s3_scanned_rows, s3_scanned_bytes,
       s3query_returned_rows, s3query_returned_bytes, files
from svl_s3query
where query = pg_last_query_id()
order by query,segment,slice;
```

```
query | segment | slice | elapsed | s3_scanned_rows | s3_scanned_bytes |
s3query_returned_rows | s3query_returned_bytes | files
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
4587 |      2 |     0 |   67811 |           0 |           0 |
    0 |      0 |     0 |         0 |           0 |           0 |
4587 |      2 |     1 |  591568 |       172462 |  11260097 |
   8513 |      1 |     1 |  170260 |           1 |           1 |
4587 |      2 |     2 |  216849 |           0 |           0 |
    0 |      0 |     0 |         0 |           0 |           0 |
4587 |      2 |     3 |  216671 |           0 |           0 |
    0 |      0 |     0 |         0 |           0 |           0 |
```

SVL_S3QUERY_SUMMARY

使用 SVL_S3QUERY_SUMMARY 檢視以取得在系統上執行之所有 Amazon Redshift Spectrum 查詢 (S3 查詢) 摘要。SVL_S3QUERY_SUMMARY 會在區段層級從 SVL_S3QUERY 彙總詳細資訊。

所有使用者都可看見 SVL_S3QUERY_SUMMARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_EXTERNAL_QUERY_DETAIL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

如需 SVCS_S3QUERY_SUMMARY 的相關資訊，請參閱 [SVCSSUMMARYQUERY](#)。

資料表欄

欄名稱	資料類型	描述
userid	integer	執行給定項目的使用者 ID。

欄名稱	資料類型	描述
query	integer	查詢 ID。您可以使用此值來聯結各種其他系統資料表與檢視。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。
segment	integer	區段號碼。查詢包含多個區段，每個區段包含一或多個步驟。
step	integer	執行的查詢步驟。
starttime	timestamp	查詢開始執行的時間 (以 UTC 表示)。
endtime	timestamp	查詢完成的時間 (以 UTC 表示)。
elapsed	integer	查詢執行的所需時間長度 (微秒)。
aborted	integer	如果查詢已被系統中止或已被使用者取消，則此欄包含 1 。如果查詢執行至完成，欄位會包含 0 。
external_table_name	char(136)	資料表外部名稱之名稱的內部格式，適用於外部資料表掃描。
file_format	character(16)	外部資料表資料的檔案格式。
is_partitioned	char(1)	若為 true (t)，此欄位值表示外部資料表已遭分割。
is_rrscan	char(1)	若為 true (t)，此欄位值表示已套用範圍限定的掃描。
is_nested	char(1)	若為 true (t)，此欄位值表示已存取巢狀資料欄資料類型。
s3_scanned_rows	bigint	從 Amazon S3 經掃描並送至 Redshift Spectrum 層的列數。
s3_scanned_bytes	bigint	根據壓縮檔案，從 Amazon S3 經掃描並送至 Redshift Spectrum 層的位元組數。

欄名稱	資料類型	描述
s3query_returned_rows	bigint	從 Redshift Spectrum 層傳回至叢集的資料列數。
s3query_returned_bytes	bigint	從 Redshift Spectrum 層傳回至叢集的位元組數。傳回至 Amazon Redshift 的大量資料可能會影響系統效能。
files	integer	此 Redshift Spectrum 查詢已處理的檔案數。限制平行處理之好處的少數檔案。
files_max	integer	在某分割上處理的檔案數上限。
files_avg	integer	在某分割上處理的平均檔案數。
splits	int	為此區段處理的分割數。在此分割上處理的分割數。有了大型可分割資料檔案 (例如, 大於 512 MB 的資料檔案), Redshift Spectrum 會嘗試將檔案分割至多個 S3 請求來進行平行處理。
splits_max	int	在此分割上處理的分割數上限。
splits_avg	int	在此分割上處理的平均分割數。
total_split_size	bigint	處理的所有分割大小總計。
max_split_size	bigint	處理的分割大小上限 (位元組)。
avg_split_size	bigint	處理的平均分割大小 (位元組)。
total_retries	integer	個別處理檔案的重試數總計。
max_retries	integer	任何處理檔案的重試數上限。

欄名稱	資料類型	描述
max_request_duration	integer	個別檔案請求的最大持續期間 (微秒)。長時間執行的查詢可能表示存在瓶頸。
avg_request_duration	double precision	檔案請求的平均持續時間 (微秒)。
max_request_parallelism	integer	此 Redshift Spectrum 查詢在某個分割上的最大平行請求數。
avg_request_parallelism	double precision	此 Redshift Spectrum 查詢在某個分割上的平均平行請求數。
total_slowdown_count	bigint	外部資料表掃描期間發生，具有降速錯誤的 Amazon S3 請求總數。
max_slowdown_count	integer	在某分割上外部資料表掃描期間發生，具有降速錯誤的最大 Amazon S3 請求數。

範例查詢

以下範例會取得前次完成查詢的掃描步驟詳細資訊。

```
select query, segment, elapsed, s3_scanned_rows, s3_scanned_bytes,
       s3query_returned_rows, s3query_returned_bytes, files
from svl_s3query_summary
where query = pg_last_query_id()
order by query, segment;
```

```
query | segment | elapsed | s3_scanned_rows | s3_scanned_bytes | s3query_returned_rows
| s3query_returned_bytes | files
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
4587 |      2 | 67811 |          0 |          0 |          0 |
|          0 |          0
```

4587	2	591568	172462	11260097	8513
		170260	1		
4587	2	216849	0	0	0
		0	0		
4587	2	216671	0	0	0
		0	0		

SVL_S3RETRIES

使用 SVL_S3RETRIES 檢視來取得基於 Amazon S3 的 Amazon Redshift Spectrum 查詢失敗的資訊。

所有使用者都可看見 SVL_S3RETRIES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
query	integer	查詢 ID。
segment	integer	區段號碼。 查詢包含多個區段，每個區段包含一或多個步驟。查詢區段可平行執行。每個區段皆在單一程序中執行。
節點	integer	節點號碼。
分割	integer	特定區段執行依據的資料分割。
eventtime	timestamp without time zone	步驟開始執行的時間 (以 UTC 表示)。
retries	integer	查詢的重試次數。
successful_fetches	integer	傳回資料的次數。

欄名稱	資料類型	描述
file_size	bigint	檔案的大小 (以位元組為單位)。
location	text	資料表的位置。
message	text	錯誤訊息。

範例查詢

以下範例會擷取關於失敗的 S3 查詢的資料。

```
SELECT svl_s3retries.query, svl_s3retries.segment, svl_s3retries.node,
       svl_s3retries.slice, svl_s3retries.eventtime, svl_s3retries.retries,
       svl_s3retries.successful_fetches, svl_s3retries.file_size,
       btrim((svl_s3retries."location")::text) AS "location",
       btrim((svl_s3retries.message)::text)
AS message FROM svl_s3retries;
```

SVL_SPATIAL_SIMPLIFY

您可以使用 COPY 命令查詢系統檢視 SVL_SPATIAL_SIMPLIFY 以取得簡化空間幾何物件的資訊。當您在 Shapefile 上使用 COPY 時，您可以指定 SIMPLIFY tolerance、SIMPLIFY AUTO 和 SIMPLIFY AUTO max_tolerance 擷取選項。簡化的結果摘要在 SVL_SPATIAL_SIMPLIFY 系統檢視中。

設定 SIMPLIFY AUTO max_tolerance 時，此檢視會針對超出大小上限的每個幾何包含一列。設定 SIMPLIFY tolerance 時，會儲存整個 COPY 操作的一個列。此列參考 COPY 查詢 ID 和指定的簡化公差。

所有使用者都可看見 SVL_SPATIAL_SIMPLIFY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_SPATIAL_SIMPLIFY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
query	integer	產生此列的查詢 ID (COPY 命令)。
line_number	integer	當指定 COPY SIMPLIFY AUTO 選項時，這個值是 Shapefile 中簡化記錄的記錄編號。
maximum_tolerance	double	COPY 命令中指定的距離公差值。這是使用 SIMPLIFY AUTO 選項的最大公差值，或使用 SIMPLIFY 選項的固定公差值。
initial_size	integer	簡化之前 GEOMETRY 資料值的大小 (以位元組為單位)。
簡化	char(1)	指定 COPY SIMPLIFY AUTO 選項時，若已成功簡化幾何則為 t，否則為 f。如果使用給定的最大公差進行簡化後，幾何的大小仍大於幾何大小上限，則幾何可能無法成功簡化。
final_size	integer	指定 COPY SIMPLIFY AUTO 選項時，這是簡化後幾何的位元組大小。
final_tolerance	double	

範例查詢

下列查詢會傳回 COPY 簡化的記錄清單。

```
SELECT * FROM svl_spatial_simplify WHERE query = pg_last_copy_id();
 query | line_number | maximum_tolerance | initial_size | simplified | final_size |
final_tolerance
-----+-----+-----+-----+-----+-----+-----
      20 |    1184704 |          -1 |    1513736 | t         |    1008808 |
1.276386653895e-05
      20 |    1664115 |          -1 |    1233456 | t         |    1023584 |
6.11707814796635e-06
```

SVL_SPECTRUM_SCAN_ERROR

您可以查詢系統檢視 SVL_SPECTRUM_SCAN_ERROR，以取得有關 Redshift Spectrum 掃描錯誤的資訊。

所有使用者都可看見 SVL_SPECTRUM_SCAN_ERROR。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_EXTERNAL_QUERY_ERROR](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

顯示記錄錯誤的範例。預設值是每個查詢 10 個項目。

欄名稱	資料類型	描述
userid	integer	產生此列之使用者的 ID。
query	integer	產生此列之查詢的 ID。
location	character(128)	要查詢之資料的位置。
rowid	character(128)	錯誤在檔案內的位置。rowid 部分以 :(冒號) 分隔，未來可能會添加其他部分。 <div style="border: 1px solid #ccc; border-radius: 10px; padding: 5px; margin: 10px 0;"> <code>row_offset :row_group :row_id</code> </div> Row_offset 是檔案中列的偏移量 (以位元組為單位)，針對不支援的檔案格式設定為 -1。資料表分為 row_groups，每個群組都有具有不同 row_id 的列。
colname	character(128)	查詢傳回的欄名稱。
original_value	character(128)	已查詢原始值。
modified_value	character(128)	根據查詢中指定的資料處理組態選項傳回的修改值。
觸發條件	character(128)	查詢中指定的資料處理選項。

欄名稱	資料類型	描述
動作	character(128)	與查詢中指定的資料處理選項相關聯的動作。
action_value	character(128)	與查詢中指定的資料處理選項相關聯的動作參數值。
error_code	integer	查詢中指定之資料處理選項的結果代碼。

範例查詢

下列查詢會傳回執行資料處理操作的列清單。

```
SELECT * FROM svl_spectrum_scan_error;
```

此查詢會傳回類似以下的結果。

```

userid  query      location                                     rowid  colname
        original_value      modified_value      trigger      action
        action_valueerror_code
100     1574007    s3://spectrum-uddh/league/spi_global_rankings.0:0
        Barclays Premier League    Barclays Premier Lea UNSPECIFIED    TRUNCATE
        156
100     1574007    s3://spectrum-uddh/league/spi_global_rankings.0:0
        34595                        32767              UNSPECIFIED
OVERFLOW_VALUE                        199
100     1574007    s3://spectrum-uddh/league/spi_global_rankings.0:1
        34151                        32767              UNSPECIFIED
OVERFLOW_VALUE                        199
100     1574007    s3://spectrum-uddh/league/spi_global_rankings.0:2
        Barclays Premier League    Barclays Premier Lea UNSPECIFIED    TRUNCATE
        156
100     1574007    s3://spectrum-uddh/league/spi_global_rankings.0:2
        33223                        32767              UNSPECIFIED
OVERFLOW_VALUE                        199
100     1574007    s3://spectrum-uddh/league/spi_global_rankings.0:3
        Barclays Premier League    Barclays Premier Lea UNSPECIFIED    TRUNCATE
        156

```

```

100 1574007 s3://spectrum-uddh/league/spi_global_rankings.0:3 league_nspi
      32808          32767          UNSPECIFIED
OVERFLOW_VALUE          199
100 1574007 s3://spectrum-uddh/league/spi_global_rankings.0:4 league_nspi
      32790          32767          UNSPECIFIED
OVERFLOW_VALUE          199
100 1574007 s3://spectrum-uddh/league/spi_global_rankings.0:5 league_name
      Spanish Primera Division Spanish Primera Divi UNSPECIFIED TRUNCATE
      156
100 1574007 s3://spectrum-uddh/league/spi_global_rankings.0:6 league_name
      Spanish Primera Division Spanish Primera Divi UNSPECIFIED TRUNCATE
      156

```

SVL_STATEMENTTEXT

使用 SVL_STATEMENTTEXT 檢視來取得在系統上執行之所有 SQL 命令的完整記錄。

SVL_STATEMENTTEXT 檢視包含 [STL_DDLTEXT](#)、[STL_QUERYTEXT](#) 和 [STL_UTILITYTEXT](#) 資料表中所有資料列的聯集。此檢視也包含對 STL_QUERY 資料表的聯結。

所有使用者都可看見 SVL_STATEMENTTEXT。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_HISTORY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	產生項目之使用者的 ID。
xid	bigint	與陳述式關聯的交易 ID。
pid	integer	陳述式的處理程序 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位為空白。

欄名稱	資料類型	描述
starttime	timestamp	陳述試開始執行的確切時間，精確度為 6 位數的小數秒。例如： 2009-06-12 11:29:19.131358
endtime	timestamp	陳述試完成執行的確切時間，精確度為 6 位數的小數秒。例如： 2009-06-12 11:29:19.193640
sequence	integer	當單一陳述式包含不只 200 個字元時，會將該陳述式的其他資料列記錄下來。序列 0 是第一列，1 是第二列，以此類推。
type	varchar(10)	SQL 陳述式的類型： QUERY 、 DDL 或 UTILITY 。
text	character(200)	SQL 文字，以 200 個字元遞增。此欄位可能包含反斜線 (\) 和換行符號 (\n) 等特殊字元。

範例查詢

下列查詢會傳回在 2009 年 6 月 16 日所執行的 DDL 陳述式：

```
select starttime, type, rtrim(text) from svl_statementtext
where starttime like '2009-06-16%' and type='DDL' order by starttime asc;
```

```
starttime | type | rtrim
-----|-----|-----
2009-06-16 10:36:50.625097 | DDL | create table ddltest(c1 int);
2009-06-16 15:02:16.006341 | DDL | drop view allticketjoin;
2009-06-16 15:02:23.65285 | DDL | drop table sales;
2009-06-16 15:02:24.548928 | DDL | drop table listing;
2009-06-16 15:02:25.536655 | DDL | drop table event;
...
```

重建儲存的 SQL

若要重建儲存在 SVL_STATEMENTTEXT text 資料欄中的 SQL，則需執行 SELECT 陳述式以從 text 資料欄的一或多個部分建立 SQL。請先以新的一行取代任意 (\n) 特殊字元，再執行重建的 SQL。下列 SELECT 陳述式顯示的結果會是 query_statement 欄中重建的 SQL 資料列。


```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
  within group (order by sequence) AS query_statement
from SVL_STATEMENTTEXT where pid=pg_backend_pid();
```

例如，下列查詢會選取 3 個資料欄。該查詢本身的長度超過 200 個字元，且會儲存在 SVL_STATEMENTTEXT 的多個部分中。

```
select
1 AS a0123456789012345678901234567890123456789012345678901234567890,
2 AS b0123456789012345678901234567890123456789012345678901234567890,
3 AS b012345678901234567890123456789012345678901234
FROM stl_querytext;
```

在本範例中，查詢會儲存在 SVL_STATEMENTTEXT text 資料欄的 2 個部分 (資料列) 中。

```
select sequence, text from SVL_STATEMENTTEXT where pid = pg_backend_pid() order by
starttime, sequence;
```

```
sequence |
          text
-----+-----
          0 | select\n1 AS
a0123456789012345678901234567890123456789012345678901234567890,\n2 AS
b0123456789012345678901234567890123456789012345678901234567890,\n3 AS
b0123456789012345678901234567890123456789012345678901234
          1 | \nFROM stl_querytext;
```

若要重建儲存在 STL_STATEMENTTEXT 中的 SQL，請執行下列 SQL。

```
select LISTAGG(CASE WHEN LEN(RTRIM(text)) = 0 THEN text ELSE RTRIM(text) END, '')
  within group (order by sequence) AS text
from SVL_STATEMENTTEXT where pid=pg_backend_pid();
```

若要在用戶端中使用產生的重建 SQL，請以新的一行取代任意 (\n) 特殊字元。

```
text
```

```
-----
select\n1 AS a0123456789012345678901234567890123456789012345678901234567890,\n2 AS b0123456789012345678901234567890123456789012345678901234567890,\n3 AS\nb0123456789012345678901234567890123456789012345678901234567890\nFROM stl_querytext;
```

SVL_STORED_PROC_CALL

您可以查詢系統檢視 SVL_STORED_PROC_CALL，以取得預存程序呼叫的相關資訊，包括開始時間、結束時間及是否取消呼叫。每次預存程序呼叫會接收查詢 ID。

所有使用者都可看見 SVL_STORED_PROC_CALL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_PROCEDURE_CALL](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	使用者的 ID，該使用者的權限用於執行陳述式。如果此呼叫套疊於 SECURITY DEFINER 預存程序內，則此為該預存程序的擁有者的 userid。
session_userid	integer	使用者的 ID，該使用者建立工作階段，並且是最上層預存程序呼叫的叫用者。
query	integer	程序呼叫的查詢 ID。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為預設值。
xid	bigint	交易 ID。
pid	integer	處理程序 ID。通常，工作階段中的所有查詢都是在相同的處理程序中執行，所以若您在相同的工作階段中執行一系

欄名稱	資料類型	描述
		列查詢，則此值通常會保持不變。在特定內部事件後，Amazon Redshift 可能會重新啟動作用中工作階段並指派新的 pid 值。如需詳細資訊，請參閱 STL_RESTARTED_SESSIONS 。
database	character(32)	當查詢發出時，使用者連接的資料庫名稱。
querytxt	character(4000)	程序呼叫查詢的實際文字。
starttime	timestamp	查詢開始執行的 UTC 時間，小數秒的精確度為 6 位數，例如： 2009-06-12 11:29:19.131358 。
endtime	timestamp	查詢完成執行的 UTC 時間，小數秒的精確度為 6 位數，例如： 2009-06-12 11:29:19.131358 。
aborted	integer	如果預存程序已被系統停止或被使用者取消，則此欄包含 1。如果呼叫執行至完成，則此欄包含 0。
from_sp_call	integer	如果程序呼叫由另一個程序呼叫所叫用，則此欄包含外層呼叫的查詢 ID。否則此欄位為 NULL。

範例查詢

下列查詢傳回過去一天的預存程序呼叫的經歷時間 (依遞減順序) 和完成狀態。

```
select query, datediff(seconds, starttime, endtime) as elapsed_time, aborted,
trim(querytxt) as call from svl_stored_proc_call where starttime >= getdate() -
interval '1 day' order by 2 desc;
```

```
query | elapsed_time | aborted | call
-----+-----+-----+-----
+-----+-----+-----+-----
4166 |          7 |      0 | call search_batch_status(35, 'succeeded');
2433 |          3 |      0 | call test_batch (123456)
1810 |          1 |      0 | call prod_benchmark (123456)
1836 |          1 |      0 | call prod_testing (123456)
1808 |          1 |      0 | call prod_portfolio ('N', 123456)
1816 |          1 |      1 | call prod_portfolio ('Y', 123456)
```

SVL_STORED_PROC_MESSAGES

您可以查詢系統檢視 SVL_STORED_PROC_MESSAGES 來取得預存程序訊息的相關資訊。即使預存程序呼叫遭到取消，引發的訊息還是會記錄到日誌。每次預存程序呼叫會接收查詢 ID。如需如何設定記錄訊息最小層級的相關資訊，請參閱 [stored_proc_log_min_messages](#)。

所有使用者都可看見 SVL_STORED_PROC_MESSAGES。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_PROCEDURE_MESSAGES](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
userid	integer	使用者的 ID，該使用者的權限用於執行陳述式。如果此呼叫套疊於 SECURITY DEFINER 預存程序內，則此為該預存程序的擁有者的 userid。
session_userid	integer	使用者的 ID，該使用者建立工作階段，並且是最上層預存程序呼叫的叫用者。
pid	integer	程序 ID。
xid	bigint	程序呼叫查詢的交易 ID。
query	integer	程序呼叫的查詢 ID。
recordtime	timestamp	引發訊息的 UTC 時間。
loglevel	integer	引發訊息的日誌層級數值。可能的值：20 – 適用於 LOG 30 – 適用於 INFO 40 – 適用於 NOTICE 50 – 適用於 WARNING 60 – 適用於 EXCEPTION
loglevel_text	character(10)	對應到 loglevel 中數值的日誌層級。可能的值：LOG (記錄)、INFO (資訊)、NOTICE (通知)、WARNING (警告) 和 EXCEPTION (例外)。
message	character(1024)	引發訊息的文字。

欄名稱	資料類型	描述
linenum	integer	引發陳述式的行號。
querytext	character(500)	程序呼叫查詢的實際文字。
label	character(320)	用於執行查詢的檔案名稱，或以 SET QUERY_GROUP 命令定義的標籤。如果查詢不是檔案型，或未設定 QUERY_GROUP 參數，則此欄位值為預設值。
aborted	integer	如果預存程序已被系統停止或被使用者取消，則此欄包含 1。如果呼叫執行至完成，則此欄包含 0。
message_x id	bigint	提出訊息的交易 ID。

範例查詢

下列 SQL 陳述式示範如何使用 SVL_STORED_PROC_MESSAGES 來檢閱引發的訊息。

```
-- Create and run a stored procedure
CREATE OR REPLACE PROCEDURE test_procl(f1 int) AS
$$
BEGIN
    RAISE INFO 'Log Level: Input f1 is %',f1;
    RAISE NOTICE 'Notice Level: Input f1 is %',f1;
    EXECUTE 'select invalid';
    RAISE NOTICE 'Should not print this';

EXCEPTION WHEN OTHERS THEN
    raise exception 'EXCEPTION level: Exception Handling';
END;
$$ LANGUAGE plpgsql;

-- Call this stored procedure
CALL test_procl(2);

-- Show raised messages with level higher than INFO
SELECT query, recordtime, loglevel, loglevel_text, trim(message) as message, aborted
FROM svl_stored_proc_messages
WHERE loglevel > 30 AND query = 193 ORDER BY recordtime;
```

```

query |          recordtime          | loglevel | loglevel_text |          message
      |          | aborted
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
  193 | 2020-03-17 23:57:18.277196 |      40 | NOTICE      | Notice Level: Input f1
is 2  |          |      1
  193 | 2020-03-17 23:57:18.277987 |      60 | EXCEPTION    | EXCEPTION level:
Exception Handling |          |      1
(2 rows)

```

```

-- Show raised messages at EXCEPTION level
SELECT query, recordtime, loglevel, loglevel_text, trim(message) as message, aborted
FROM svl_stored_proc_messages
WHERE loglevel_text = 'EXCEPTION' AND query = 193 ORDER BY recordtime;

```

```

query |          recordtime          | loglevel | loglevel_text |          message
      |          | aborted
-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
  193 | 2020-03-17 23:57:18.277987 |      60 | EXCEPTION    | EXCEPTION level:
Exception Handling |          |      1

```

下列 SQL 陳述式示範如何使用 SVL_STORED_PROC_MESSAGES，在建立預存程序時使用 SET 選項來檢閱引發的訊息。因為 test_proc() 的最低日誌層級為 NOTICE，只有 NOTICE、WARNING 和 EXCEPTION 層級訊息會記錄在 SVL_STORED_PROC_MESSAGES 中。

```

-- Create a stored procedure with minimum log level of NOTICE
CREATE OR REPLACE PROCEDURE test_proc() AS
$$
BEGIN
    RAISE LOG 'Raise LOG messages';
    RAISE INFO 'Raise INFO messages';
    RAISE NOTICE 'Raise NOTICE messages';
    RAISE WARNING 'Raise WARNING messages';
    RAISE EXCEPTION 'Raise EXCEPTION messages';
    RAISE WARNING 'Raise WARNING messages again'; -- not reachable
END;
$$ LANGUAGE plpgsql SET stored_proc_log_min_messages = NOTICE;

-- Call this stored procedure
CALL test_proc();

```

```
-- Show the raised messages
SELECT query, recordtime, loglevel_text, trim(message) as message, aborted FROM
svl_stored_proc_messages
WHERE query = 149 ORDER BY recordtime;
```

query aborted	recordtime	loglevel_text	message
149 1	2020-03-16 21:51:54.847627	NOTICE	Raise NOTICE messages
149 1	2020-03-16 21:51:54.84766	WARNING	Raise WARNING messages
149 1	2020-03-16 21:51:54.847668	EXCEPTION	Raise EXCEPTION messages

(3 rows)

SVL_TERMINATE

記錄使用者取消或終止程序的時間。

SELECT PG_TERMINATE_BACKEND(pid)、SELECT PG_CANCEL_BACKEND(pid) 和 CANCEL pid 可在 SVL_TERMINATE 中建立日誌項目。

只有超級使用者可以看到 SVL_TERMINATE。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_QUERY_HISTORY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
pid	integer	已取消或終止之程序的程序 ID。
eventtime	timestamp	取消或終止程序的時間。
userid	integer	執行命令之使用者的使用者 ID。
type	string	終止的類型。它可以是 CANCEL 或 TERMINATE。

下列命令顯示最新的取消查詢。

```
select * from svl_terminate order by eventtime desc limit 1;
 pid |          eventtime          | userid | type
-----+-----+-----+-----
 8324 | 2020-03-24 09:42:07.298937 |      1 | CANCEL
(1 row)
```

SVL_UDF_LOG

記錄使用者定義函數 (UDF) 執行期間產生的系統定義的錯誤和警告訊息。

所有使用者都可看見 SVL_UDF_LOG。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_UDF_LOG](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
query	bigint	查詢 ID。您可以使用此 ID 來聯結各種其他系統資料表與檢視。
message	char(4096)	函數產生的訊息。
已建立	timestamp	日誌建立的時間。
回溯	char(4096)	若果可用，此值會提供 UDF 的堆疊回溯。如需詳細資訊，請參閱 Python Standard Library 的 traceback 。
funcname	character(256)	正在執行之 UDF 的名稱。
節點	integer	訊息產生的節點。
分割	integer	訊息產生的分割。

欄名稱	資料類型	描述
seq	integer	分割上訊息的順序。

範例查詢

以下範例說明 UDF 如何處理系統定義的錯誤。第一個區塊顯示傳回引數反向之 UDF 函數的定義。執行函數並提供 0 引數時，如第二個區塊所示，函數會傳回錯誤。第三個陳述式會讀取在 SVL_UDF_LOG 中記錄的錯誤訊息

```
-- Create a function to find the inverse of a number

CREATE OR REPLACE FUNCTION f_udf_inv(a int)
  RETURNS float IMMUTABLE
AS $$
  return 1/a
$$ LANGUAGE plpythonu;

-- Run the function with a 0 argument to create an error
Select f_udf_inv(0) from sales;

-- Query SVL_UDF_LOG to view the message

Select query, created, message::varchar
from svl_udf_log;

query |          created          | message
-----+-----
+-----+-----
  2211 | 2015-08-22 00:11:12.04819 | ZeroDivisionError: long division or modulo by
zero\nNone
```

下列範例會將記錄且警告訊息新增至 UDF，以至於除以零操作會導致警告訊息，而不是停止並出現錯誤訊息。

```
-- Create a function to find the inverse of a number and log a warning

CREATE OR REPLACE FUNCTION f_udf_inv_log(a int)
  RETURNS float IMMUTABLE
AS $$
  import logging
```

```

logger = logging.getLogger() #get root logger
if a==0:
    logger.warning('You attempted to divide by zero.\nReturning zero instead of error.
\n')
    return 0
else:
    return 1/a
$$ LANGUAGE plpythonu;

```

下列範例會執行函數，接著查詢 SVL_UDF_LOG，來檢視訊息。

```

-- Run the function with a 0 argument to trigger the warning
Select f_udf_inv_log(0) from sales;

-- Query SVL_UDF_LOG to view the message

Select query, created, message::varchar
from svl_udf_log;

query |          created          | message
-----+-----+-----
      0 | 2015-08-22 00:11:12.04819 | You attempted to divide by zero.
                                     Returning zero instead of error.

```

SVL_USER_INFO

您可以使用 SVL_USER_INFO 檢視來擷取關於 Amazon Redshift 資料庫使用者的資料。

只有超級使用者可以看到 SVL_USER_INFO。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
username	text	角色的使用者名稱。
usesysid	integer	使用者的使用者 ID。
usecreate db	boolean	一個值，指出使用者是否具有建立資料庫的許可。

欄名稱	資料類型	描述
usesuper	boolean	一個值，指出使用者是否為超級使用者。
usecatupd	boolean	一個值，指出使用者可以更新系統目錄。
useconnlimit	text	使用者可以開啟的連線數量。
syslogaccess	text	一個值，指出使用者是否具有系統日誌的存取權。兩個可能值為 RESTRICTED 和 UNRESTRICTED。RESTRICTED 表示不是超級使用者的使用者可以看到其記錄。UNRESTRICTED 表示不是超級使用者的使用者可以看到他們具有 SELECT 權限的系統檢視和資料表中的所有記錄。
last_ddl_ts	timestamp	使用者執行的最新資料定義語言 (DDL) create 陳述式的時間戳記。
sessiontimeout	integer	逾時前，工作階段保持非作用中或閒置的時間上限 (以秒為單位)。0 表示未設定逾時。如需叢集的閒置或非作用中逾時設定詳細資訊，請參閱《Amazon Redshift 管理指南》中的 Amazon Redshift 中的配額和限制 。
external_id	text	第三方身分提供者中使用者的唯一識別碼。

範例查詢

下列命令會從 SVL_USER_INFO 擷取使用者資訊。

```
SELECT * FROM SVL_USER_INFO;
```

SVL_VACUUM_PERCENTAGE

SVL_VACUUM_PERCENTAGE 檢視報告在執行清空後為資料表配置的資料區塊百分比。此百分比數顯示磁碟空間的回收量。如需清空公用程式的相關資訊，請參閱 [VACUUM](#) 命令。

只有超級使用者可以看到 SVL_VACUUM_PERCENTAGE。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

此資料表中的部份或所有資料也會在 SYS 監控檢視 [SYS_VACUUM_HISTORY](#) 中找到。SYS 監視檢視中的資料會格式化為更易於使用和理解。我們建議您使用 SYS 監控檢視進行查詢。

資料表欄

欄名稱	資料類型	描述
xid	bigint	清空陳述式的交易 ID。
table_id	integer	清空資料表的資料表 ID。
百分比	bigint	清空後資料區塊的百分比 (與在清空執行前資料表中區塊數相對)。

範例查詢

下列查詢顯示在資料表 100238 上特定操作的百分比：

```
select * from svl_vacuum_percentage
where table_id=100238 and xid=2200;
```

```
xid | table_id | percentage
-----+-----+-----
1337 | 100238 |          60
(1 row)
```

在此清空操作後，資料表包含原始區塊的 60%。

系統目錄資料表

主題

- [PG_ATTRIBUTE_INFO](#)
- [PG_CLASS_INFO](#)
- [PG_DATABASE_INFO](#)
- [PG_DEFAULT_ACL](#)
- [PG_EXTERNAL_SCHEMA](#)
- [PG_LIBRARY](#)
- [PG_PROC_INFO](#)
- [PG_STATISTIC_INDICATOR](#)

- [PG_TABLE_DEF](#)
- [PG_USER_INFO](#)
- [查詢目錄資料表](#)

系統目錄儲存結構描述中繼資料，例如資料表和資料欄的相關資訊。系統目錄資料表有 PG 字首。

Amazon Redshift 使用者可存取標準 PostgreSQL 目錄資料表。如需 PostgreSQL 系統目錄的詳細資訊，請參閱 [PostgreSQL 系統資料表](#)

PG_ATTRIBUTE_INFO

PG_ATTRIBUTE_INFO 是建立在 PostgreSQL 目錄資料表 PG_ATTRIBUTE 和內部目錄資料表 PG_ATTRIBUTE_ 上的 Amazon Redshift 系統檢視ACL。PG_ATTRIBUTE_INFO 包含資料表或檢視資料欄的詳細資訊，如果有的話，包括資料欄存取控制清單。

資料表欄

除了 PGATTRIBUTE_INFO 中的資料欄之外，PG__ 還顯示下列資料欄ATTRIBUTE。

欄名稱	資料類型	描述
attacl	aclitem[]	在此資料欄上特別授與的資料欄層級存取權限 (如果有的話)。

PG_CLASS_INFO

PG_CLASS_INFO 是建立在 PostgreSQL 目錄資料表 PG_CLASS 和 PG_CLASS_ 上的 Amazon Redshift 系統檢視EXTENDED。PG_CLASS_INFO 包含資料表建立時間和目前分佈樣式的詳細資訊。如需詳細資訊，請參閱 [查詢最佳化的資料分佈](#)。

所有使用者都可看見 PG_CLASS_INFO。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

除了 PGCLASS_INFO 中的資料欄之外，PG__ 還顯示下列資料欄CLASS。PG_CLASS 中的oid資料欄在 PG_CLASS_INFO 資料表relid中稱為。

欄名稱	資料類型	描述
relcreate ontime	timestamp	資料表建立UTC的時間。
releffect ivediststyle	integer	資料表的分佈樣式，或如果資料表使用自動分佈，則為 Amazon Redshift 指派的目前分佈樣式。

PG_CLASS_INFO 中的資料RELEFFECTIVEDISTSTYLE欄表示資料表目前的分佈樣式。如果資料表使用自動分佈，RELEFFECTIVEDISTSTYLE則為 10、11 或 12，這表示有效分佈樣式為 AUTO (ALL)、AUTO (EVEN) 或 AUTO () KEY。如果資料表使用自動分佈，則分佈樣式一開始可能會顯示 AUTO (ALL)，然後在資料表增長時變更為 AUTO (EVEN)，如果發現資料欄可作為分佈金鑰有用，則變更為 AUTO (KEY)。

下表提供RELEFFECTIVEDISTSTYLE欄中每個值的分佈樣式：

RELEFFECTIVEDISTSTYLE	目前分佈樣式
0	EVEN
1	KEY
8	ALL
10	AUTO (ALL)
11	AUTO (EVEN)
12	AUTO (KEY)

範例

下列查詢會傳回目錄中資料表的目前分佈樣式。

```
select relid as tableid,trim(nspname) as schemaname,trim(relname) as
tablename,relstyle,releffectivediststyle,
CASE WHEN "relstyle" = 0 THEN 'EVEN'::text
      WHEN "relstyle" = 1 THEN 'KEY'::text
```

```

WHEN "reldiststyle" = 8 THEN 'ALL'::text
WHEN "releffectivediststyle" = 10 THEN 'AUTO(ALL)'::text
WHEN "releffectivediststyle" = 11 THEN 'AUTO(EVEN)'::text
WHEN "releffectivediststyle" = 12 THEN 'AUTO(KEY)'::text ELSE '<<UNKNOWN>>'::text
END as diststyle,relcreationtime
from pg_class_info a left join pg_namespace b on a.relnamespace=b.oid;

```

```

tableid | schemaname | tablename | reldiststyle | releffectivediststyle | diststyle |
relcreationtime
-----+-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----+-----
3638033 | public    | customer  |      0 |          0 | EVEN      |
2019-06-13 15:02:50.666718
3638037 | public    | sales     |      1 |          1 | KEY       |
2019-06-13 15:03:29.595007
3638035 | public    | lineitem  |      8 |          8 | ALL       |
2019-06-13 15:03:01.378538
3638039 | public    | product   |      9 |         10 | AUTO(ALL) |
2019-06-13 15:03:42.691611
3638041 | public    | shipping  |      9 |         11 | AUTO(EVEN) |
2019-06-13 15:03:53.69192
3638043 | public    | support   |      9 |         12 | AUTO(KEY)  |
2019-06-13 15:03:59.120695
(6 rows)

```

PG_DATABASE_INFO

PG_DATABASE_INFO是延伸 PostgreSQL 目錄資料表 PG_ 的 Amazon Redshift 系統檢視 DATABASE。

所有使用者都可看見 PG_DATABASE_INFO。

資料表欄

除了 PGDATABASE_INFO 中的資料欄之外，PG__ 還包含下列資料欄DATABASE。PG_DATABASE 中的oid資料欄在 PG_DATABASE_INFO資料表datid中稱為。如需詳細資訊，請參閱 [PostgreSQL 文件](#)。

欄名稱	資料類型	描述
datid	oid	系統資料表內部使用的物件識別符 (OID)。
datconnlimit	text	您可以對此資料庫進行的並行連線數量上限。值為 -1 表示沒有限制。

PG_DEFAULT_ACL

儲存預設存取權限的相關資訊。如需預設存取權限的相關資訊，請參閱 [ALTER DEFAULT PRIVILEGES](#)。

所有使用者都可看見 PG_DEFAULT_ACL。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱 [系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
defacluser	integer	將所列權限套用至其中的使用者 ID。
defaclnamespace	oid	將預設權限所套用至其中的結構描述物件 ID。如果未指定任何結構描述，預設值會是 0。
defaclobjtype	character	將預設權限所套用至其中的物件類型。有效值如下： <ul style="list-style-type: none"> r-關聯 (資料表或檢視) f-函數 p-預存程序
defaclacl	aclitem[]	定義指定使用者或使用者群組的預設權限和物件類型的字串。 如果將此權限授予另一個使用者，字串格式如下： <code>{ username=privilegestring/grantor }</code> username

欄名稱	資料類型	描述
		<p>將權限授予至其中的使用者名稱。如果省略使用者名稱，則會將 權限授予 PUBLIC。</p> <p>如果將此權限授予另一個使用者群組，字串格式如下：</p> <pre>{ "group groupname=privilegestring/grantor" }</pre> <p>privilegestring</p> <p>指定授予哪一個權限的字串。</p> <p>有效的 值如下：</p> <ul style="list-style-type: none"> • r-SELECT (讀取) • a-INSERT (附加) • w-UPDATE (寫入) • d-DELETE • x-授予建立外部金鑰限制 (REFERENCES) 的權限。 • X-EXECUTE • *-表示接收上述權限的使用者可繼而將相同的權限授予其他人 (WITH GRANT OPTION)。 <p>grantor</p> <p>授予權限的使用者名稱。</p> <p>下列範例指出使用者admin已將包括 WITH GRANT 在內的所有權限OPTION授予使用者 dbuser。</p> <pre>dbuser=r*a*w*d*x*X*/admin</pre>

範例

下列查詢會傳回為資料庫定義之所有預設權限。

```
select pg_get_userbyid(d.defacluser) as user,
n.nspname as schema,
case d.defaclobjtype when 'r' then 'tables' when 'f' then 'functions' end
as object_type,
array_to_string(d.defaclacl, ' + ') as default_privileges
from pg_catalog.pg_default_acl d
left join pg_catalog.pg_namespace n on n.oid = d.defaclnamespace;
```

```
user | schema | object_type | default_privileges
-----+-----+-----+-----
admin | tickit | tables      | user1=r/admin + "group group1=a/admin" + user2=w/admin
```

上述範例中的結果顯示，對於使用者在tickit結構描述admin中建立的所有新資料表，會將SELECT 權限admin授予 user1、INSERT 權限授予 group1，以及 UPDATE 權限授予 user2。

PG_EXTERNAL_SCHEMA

儲存外部結構描述的相關資訊。

所有使用者都可看見 PG_EXTERNAL_SCHEMA。超級使用者可以看見所有資料列；一般使用者只能看見它們有權存取的中繼資料。如需詳細資訊，請參閱[CREATE EXTERNAL SCHEMA](#)。

資料表欄

欄名稱	資料類型	描述
esoid	oid	外部結構描述 ID。
eskind	integer	一種外部結構描述。
esdbname	text	外部資料庫名稱。
esoptions	text	外部結構描述選項。

範例

下列範例顯示外部結構描述的詳細資訊。

```
select esoid, nspname as schemaname, nspowner, esdbname as external_db, esoptions
from pg_namespace a,pg_external_schema b where a.oid=b.esoid;
```

```

esoid | schemaname | nspowner | external_db | esoptions
-----+-----+-----+-----
+-----+-----+-----+-----
100134 | spectrum_schema | 100 | spectrum_db |
{"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
100135 | spectrum | 100 | spectrumdb |
{"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}
100149 | external | 100 | external_db |
{"IAM_ROLE":"arn:aws:iam::123456789012:role/mySpectrumRole"}

```

PG_LIBRARY

儲存使用者定義之資料庫的相關資訊。

所有使用者都可看見 PG_LIBRARY。超級使用者可以看見所有資料列；一般使用者只能看見自己的資料。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
name	name	資料庫名稱。
language_oid	oid	保留以供系統使用。
file_store_id	integer	保留以供系統使用。
owner	integer	資料庫擁有者的使用者 ID。

範例

以下範例會傳回使用者安裝資料庫的資訊。

```

select * from pg_library;

name | language_oid | file_store_id | owner

```

```
-----+-----+-----+-----
f_urlparse |      108254 |      2000 |      100
```

PG_PROC_INFO

PG_PROC_INFO 是建置在 PostgreSQL 目錄資料表 PG_PROC 和內部目錄資料表 PG_PROC_ 上的 Amazon Redshift 系統檢視 EXTENDED。PG_PROC_INFO 包含有關預存程序和函數的詳細資訊，包括與輸出引數相關的資訊，如果有的話。

資料表欄

除了 PGPROC_INFO 中的資料欄之外，PG__ 還顯示下列資料欄 PROC。PG_PROC 中的 oid 資料欄在 PG_PROC_INFO 資料表 prooid 中稱為。

欄名稱	資料類型	描述
prooid	oid	函數或預存程序的物件 ID。
prokind	"char"	指出函數或預存程序類型的值。此值為 'f' 代表一般函數、'p' 代表預存程序、'a' 代表彙總函數。
proargmodes	"char"[]	具有程序引數模式的陣列，編碼為「i」表示 IN 引數，「o」表示 OUT 引數，「b」表示 INOUT 引數。如果所有引數都是 IN 引數，則此欄位為 NULL。與 proallargtypes 陣列中的位置對應的下標。
proallargtypes	oid[]	由程序引數的資料類型組成的陣列。此陣列包含所有類型的引數（包括 OUT 和 INOUT 引數）。不過，如果所有引數都是 IN 引數，則此欄位為 NULL。下標以 1 開始。相反地，proargtypes 中的下標以 0 開始。

PG_PROC_INFO 中的欄位 proargname 包含所有類型的引數名稱（包括 OUT 和 INOUT），如果有的話。

PG_STATISTIC_INDICATOR

儲存自上次以來插入或刪除的資料列數目的相關資訊 ANALYZE。PG_STATISTIC_INDICATOR 資料表會在 DML 操作後頻繁更新，因此統計資料為近似值。

PG_STATISTIC_INDICATOR 僅供超級使用者檢視。如需詳細資訊，請參閱[系統資料表和檢視中資料的可見性](#)。

資料表欄

欄名稱	資料類型	描述
stairelid	oid	資料表 ID
stairows	float	資料表中列總數。
staiins	float	自上次 以來插入的資料列數目ANALYZE。
staidels	float	自上次 以來已刪除或更新的列數ANALYZE。

範例

下列範例傳回自上次 以來資料表變更的資訊ANALYZE。

```
select * from pg_statistic_indicator;
```

stairelid	stairows	staiins	staidels
108271	11	0	0
108275	365	0	0
108278	8798	0	0
108280	91865	0	100632
108267	89981	49990	9999
108269	808	606	374
108282	152220	76110	248566

PG_TABLE_DEF

儲存資料表欄位的相關資訊。

PG_TABLE_DEF只會傳回使用者可見資料表的相關資訊。如果 PG_TABLE_DEF未傳回預期結果，請確認[search_path](#)參數已正確設定，以包含相關結構描述。

您可以使用 [SVV_TABLE_INFO](#) 來檢視資料表的更全面性資訊，包括資料配送偏度、索引鍵配送偏度、資料表大小、統計等方面的問題。

資料表欄

欄名稱	資料類型	描述
結構描述名稱	name	結構描述名稱。
資料表名稱	name	資料表名稱。
欄位	name	資料欄名稱。
type	text	欄位資料類型。
編碼	character(32)	欄位編碼。
distkey	boolean	若此欄位是資料表的分佈索引鍵，則為 true。
sortkey	integer	排序索引鍵中的欄位順序。如果資料表使用的是複合排序索引鍵，則排序索引鍵部分的所有欄位會有正值，指示排序索引鍵中欄位的位置。如果資料表使用的是交錯的排序索引鍵，則排序索引鍵部分的每個欄會有交替為正或負的值，其中絕對值表示排序索引鍵中的欄位置。若為 0，則欄位不是排序索引鍵的一部分。
notnull	boolean	如果資料欄有NOTNULL限制，則為 true。

範例

下列範例顯示 LINEORDER_COMPOUND 資料表的複合排序索引鍵資料欄。

```
select "column", type, encoding, distkey, sortkey, "notnull"
from pg_table_def
where tablename = 'lineorder_compound'
and sortkey <> 0;
```

```
column          | type      | encoding | distkey | sortkey | notnull
-----+-----+-----+-----+-----+-----
lo_orderkey    | integer  | delta32k | false   | 1       | true
lo_custkey     | integer  | none     | false   | 2       | true
lo_partkey     | integer  | none     | true    | 3       | true
```

```

lo_suppkey | integer | delta32k | false | 4 | true
lo_orderdate | integer | delta | false | 5 | true
(5 rows)

```

下列範例顯示 LINEORDER_INTERLEAVED 資料表的交錯排序索引鍵資料欄。

```

select "column", type, encoding, distkey, sortkey, "notnull"
from pg_table_def
where tablename = 'lineorder_interleaved'
and sortkey <> 0;

column          | type      | encoding | distkey | sortkey | notnull
-----+-----+-----+-----+-----+-----
lo_orderkey     | integer   | delta32k | false   | -1      | true
lo_custkey      | integer   | none     | false   | 2       | true
lo_partkey      | integer   | none     | true    | -3      | true
lo_suppkey      | integer   | delta32k | false   | 4       | true
lo_orderdate    | integer   | delta    | false   | -5      | true
(5 rows)

```

PG_TABLE__DEF 只會傳回包含在搜尋路徑中的結構描述資料表資訊。如需詳細資訊，請參閱[search_path](#)。

例如，假設您建立新的結構描述和新資料表，然後查詢 PG_TABLE_DEF。

```

create schema demo;
create table demo.demotable (one int);
select * from pg_table_def where tablename = 'demotable';

schemaname|tablename|column| type | encoding | distkey | sortkey | notnull
-----+-----+-----+-----+-----+-----+-----+-----

```

查詢未傳回新資料表的任何列。檢查 search_path 的設定。

```

show search_path;

 search_path
-----
 $user, public
(1 row)

```

將 demo 結構描述新增至搜尋路徑中並再次執行查詢。

```

set search_path to '$user', 'public', 'demo';

select * from pg_table_def where tablename = 'demotable';

schemaname| tablename | column|  type   | encoding | distkey| sortkey| notnull
-----+-----+-----+-----+-----+-----+-----+-----
demo      | demotable | one   | integer | none     | f      |      0 | f
(1 row)

```

PG_USER_INFO

PG_USER_INFO 是 Amazon Redshift 系統檢視，可顯示使用者資訊，例如使用者 ID 和密碼過期時間。

只有超級使用者才能看到 PG_USER_INFO。

資料表欄

PG_USER_INFO 包含下列資料欄。如需詳細資訊，請參閱 [PostgreSQL 文件](#)。

欄名稱	資料類型	描述
username	name	使用者名稱。
usesysid	integer	使用者 ID。
usecreatedb	boolean	如果使用者可以建立資料庫，則為 True。
usesuper	boolean	如果使用者是超級使用者，則為 True。
usecatupd	boolean	若為 true 表示使用者可以更新系統目錄。
passwd	text	密碼。
valuntil	abstime	密碼的過期日期和時間。
useconfig	text[]	執行期變數的工作階段預設值。
useconnlimit	text	使用者可以開啟的連線數量。

查詢目錄資料表

主題

- [目錄查詢範例](#)

通常，您可以將目錄資料表和檢視 (其名稱開頭為 **PG_** 的關係) 聯結至 Amazon Redshift 資料表和檢視。

目錄資料表使用 Amazon Redshift 不支援的各種資料類型。下列資料類型在將查詢對 Amazon Redshift 資料表聯結目錄資料表時受到支援：

- bool
- "char"
- float4
- int2
- int4
- int8
- name
- oid
- text
- varchar

如果您寫入加入查詢，其明確或隱含地參考擁有不受支援之資料類型的欄位，則查詢會傳回錯誤。除了 PG_SETTINGS 和LOCKS PG_ 資料表所使用的函數之外，某些目錄資料表中使用的SQL函數也不受支援。

例如，由於不支援的函數，無法在與 Amazon Redshift 資料表的聯結中查詢 PG_STATS 資料表。

下列目錄資料表和檢視提供的資訊很有用，您可以將其與 Amazon Redshift 資料表中的資訊結合在一起。因為資料類型和函數限制，部分資料表僅允許部分存取權。當您查詢可部分存取的資料表時，請小心地選取或參考其欄位。

以下資料表是可完整存取的，且未包含不受支援的函數類型：

- [pg_attribute](#)
- [pg_cast](#)

- [pg_depend](#)
- [pg_description](#)
- [pg_locks](#)
- [pg_opclass](#)

以下資料表是可部分存取的，且包含一些不受支援類型、函數和遭截斷的文字欄位。文字欄位中的值遭截斷為 `varchar(256)` 值。

- [pg_class](#)
- [pg_constraint](#)
- [pg_database](#)
- [pg_group](#)
- [pg_language](#)
- [pg_namespace](#)
- [pg_operator](#)
- [pg_proc](#)
- [pg_settings](#)
- [pg_statistic](#)
- [pg_tables](#)
- [pg_type](#)
- [pg_user](#)
- [pg_views](#)

未在此處列出的目錄資料表為不可存取或可能未供 Amazon Redshift 管理員使用。然而，如果您的查詢不包含對 Amazon Redshift 資料表的加入時，您可以開放地查詢任何目錄資料表或檢視。

您可以使用 Postgres 目錄資料表中的OID資料欄作為聯結資料欄。例如，聯結條件會將每個 PG_DATABASE 資料列的內部資料庫物件 ID 與 STV__ 資料表中可見的 DBTBL_PERMID 資料欄 `pg_database.oid = stv_tbl_perm.db_id` 相符。當您從資料表中選取時，資料OID欄是看不到的內部主索引鍵。目錄檢視沒有OID資料欄。

某些 Amazon Redshift 函數必須只能在運算節點上執行。如果查詢參考使用者建立的資料表，會在運算節點SQL上執行。

僅參考目錄資料表（具有 PG 字首的資料表，例如 PG_TABLE_DEF）或不參考任何資料表的查詢，僅在領導節點上執行。

如果使用運算節點函數的查詢未參考使用者定義的資料表或 Amazon Redshift 系統資料表傳回下列錯誤。

```
[Amazon](500310) Invalid operation: One or more of the used functions must be applied on at least one user created table.
```

目錄查詢範例

下列查詢顯示幾個方法，您可以用這些方法來查詢目錄資料表來取得 Amazon Redshift 資料庫的相關有用資訊。

檢視資料表 ID、資料庫、結構描述和資料表名稱

下列檢視定義會將 STV_TBL_PERM 系統資料表與 PG_NAMESPACE、PG_CLASS 和 PG_DATABASE 系統目錄資料表聯結，以傳回資料表 ID、資料庫名稱、結構描述名稱和資料表名稱。

```
create view tables_vw as
select distinct(stv_tbl_perm.id) table_id
,trim(pg_database.datname) db_name
,trim(pg_namespace.nspname) schema_name
,trim(pg_class.relname) table_name
from stv_tbl_perm
join pg_class on pg_class.oid = stv_tbl_perm.id
join pg_namespace on pg_namespace.oid = pg_class.relnamespace
join pg_database on pg_database.oid = stv_tbl_perm.db_id;
```

下列範例會回傳資料表 ID 117855 的資訊。

```
select * from tables_vw where table_id = 117855;
```

```
table_id | db_name   | schema_name | table_name
-----+-----+-----+-----
 117855 |         dev | public      | customer
```

根據 Amazon Redshift 資料表列出欄數

下列查詢會聯結一些目錄資料表，以瞭解每個 Amazon Redshift 資料表包含多少個欄。Amazon Redshift 資料表名稱同時儲存在 PG_TABLES 和 STV_TBL_ 中 PERM；可能的話，請使用 PG_TABLES 傳回 Amazon Redshift 資料表名稱。

此查詢不包含任何 Amazon Redshift 資料表。

```
select nspname, relname, max(attnum) as num_cols
from pg_attribute a, pg_namespace n, pg_class c
where n.oid = c.relnamespace and a.attrelid = c.oid
and c.relname not like '%pkey'
and n.nspname not like 'pg%'
and n.nspname not like 'information%'
group by 1, 2
order by 1, 2;
```

nspname	relname	num_cols
public	category	4
public	date	8
public	event	6
public	listing	8
public	sales	10
public	users	18
public	venue	5

(7 rows)

列出資料庫中的結構描述和資料表

下列查詢會將 STV_TBL_PERM 聯結至某些 PG 資料表，以傳回 TICKIT 資料庫中的資料表清單及其結構描述名稱（NSPNAME 欄）。查詢也會傳回每個資料表中的列總數。（當系統中多個結構描述的資料表名稱相同時，此查詢相當實用。）

```
select datname, nspname, relname, sum(rows) as rows
from pg_class, pg_namespace, pg_database, stv_tbl_perm
where pg_namespace.oid = relnamespace
and pg_class.oid = stv_tbl_perm.id
and pg_database.oid = stv_tbl_perm.db_id
and datname = 'tickit'
group by datname, nspname, relname
order by datname, nspname, relname;
```

```

datname | nspname | relname | rows
-----+-----+-----+-----
tickit  | public  | category |    11
tickit  | public  | date     |   365
tickit  | public  | event    |  8798
tickit  | public  | listing  | 192497
tickit  | public  | sales    | 172456
tickit  | public  | users    |  49990
tickit  | public  | venue    |    202
(7 rows)

```

列出資料表 IDs、資料類型、資料欄名稱和資料表名稱

下列查詢列出每個使用者資料表和其欄位的部分相關資訊：資料表 ID、資料表名稱、其欄位名稱和每個欄位的資料類型：

```

select distinct attrelid, rtrim(name), attname, typename
from pg_attribute a, pg_type t, stv_tbl_perm p
where t.oid=a.atttypid and a.attrelid=p.id
and a.attrelid between 100100 and 110000
and typename not in('oid','xid','tid','cid')
order by a.attrelid asc, typename, attname;

```

```

attrelid | rtrim | attname | typename
-----+-----+-----+-----
  100133 | users | likebroadway | bool
  100133 | users | likeclassical | bool
  100133 | users | likeconcerts | bool
...
  100137 | venue | venuestate | bpchar
  100137 | venue | venueid | int2
  100137 | venue | venueseats | int4
  100137 | venue | venuecity | varchar
...

```

為資料表中的每個欄位計數資料區塊數

下列查詢會將 STV_BLOCKLIST 資料表聯結至 PG_CLASS，以傳回 SALES 資料表中資料欄的儲存資訊。

```

select col, count(*)

```

```
from stv_blocklist s, pg_class p
where s.tbl=p.oid and relname='sales'
group by col
order by col;
```

col	count
0	4
1	4
2	4
3	4
4	4
5	4
6	4
7	4
8	4
9	8
10	4
12	4
13	8

(13 rows)

組態參考

您可以使用組態來協助自訂您的環境。使用 Amazon Redshift，您可以透過設定各種參數和設定來自訂和最佳化資料倉儲環境。組態參考概述可用的叢集屬性、資料庫參數和工作負載管理（WLM）組態選項。您可以參考此參考，根據其特定需求微調效能、安全性和資源配置。下列參考提供修改這些組態的詳細指引，以達成您想要的資料倉儲設定。

主題

- [修改伺服器組態](#)
- [analyze_threshold_percent](#)
- [cast_super_null_on_error](#)
- [datashare_break_glass_session_var](#)
- [datestyle](#)
- [default_geometry_encoding](#)
- [describe_field_name_in_uppercase](#)
- [downcase_delimited_identifier](#)
- [enable_case_sensitive_identifier](#)
- [enable_case_sensitive_super_attribute](#)
- [enable_numeric_rounding](#)
- [enable_result_cache_for_session](#)
- [enable_vacuum_boost](#)
- [error_on_nondeterministic_update](#)
- [extra_float_digits](#)
- [interval_forbid_composite_literals](#)
- [json_serialization_enable](#)
- [json_serialization_parse_nested_strings](#)
- [max_concurrency_scaling_clusters](#)
- [max_cursor_result_set_size](#)
- [mv_enable_aqmv_for_session](#)
- [navigate_super_null_on_error](#)

- [parse_super_null_on_error](#)
- [pg_federation_repeatable_read](#)
- [query_group](#)
- [search_path](#)
- [spectrum_enable_pseudo_columns](#)
- [enable_spectrum_oid](#)
- [spectrum_query_maxerror](#)
- [statement_timeout](#)
- [stored_proc_log_min_messages](#)
- [timezone](#)
- [use_fips_ssl](#)
- [wlm_query_slot_count](#)

修改伺服器組態

您可以採用下列方式來變更伺服器組態：

- 使用 [SET](#) 命令時，將僅覆寫目前工作階段持續時間的設定。

例如：

```
set extra_float_digits to 2;
```

- 修改叢集的參數群組設定。此參數群組設定包含您可設定的額外參數。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 參數群組](#)。
- 使用 [ALTER USER](#) 命令，將指定之使用者執行的所有工作階段的組態參數設定為新的值。

```
ALTER USER username SET parameter { TO | = } { value | DEFAULT }
```

使用 SHOW 命令來檢視目前的參數設定。使用 SHOW ALL 檢視您可以使用 [SET](#) 命令設定的所有設定。

```
SHOW ALL;
```

```
name | setting
```



```
-----+-----  
analyze_threshold_percent | 10  
datestyle                 | ISO, MDY  
extra_float_digits       | 2  
query_group              | default  
search_path              | $user, public  
statement_timeout        | 0  
timezone                 | UTC  
wlm_query_slot_count     | 1
```

Note

請注意，組態參數會套用至您在資料倉儲中連線的資料庫。

analyze_threshold_percent

值 (粗體為預設值)

10、0 到 100.0

描述

設定為了分析資料表而變更的資料列百分比臨界值。為了縮短處理時間並改善整體系統效能，Amazon Redshift ANALYZE 會略過變更資料列百分比低於 所指定的任何資料表 `analyze_threshold_percent`。例如，如果資料表包含 100,000,000 個資料列，且自上次以來有 9,000,000 個資料列變更 ANALYZE，則根據預設會略過資料表，因為變更的資料列中有不到 10%。若要在只有少量資料列變更時分析資料表，請將 `analyze_threshold_percent` 設定為任何更小的數。例如，如果將 `analyze_threshold_percent` 設為 0.01，則 100,000,000 資料列的資料表中變更的資料列大於 10,000 列資料時，便不會略過資料表。若要分析所有資料表，即使沒有任何資料列變更，請將 `analyze_threshold_percent` 設為 0。

您只能使用 SET 命令修改目前工作階段的 `analyze_threshold_percent` 參數。不能在參數群組中修改參數。

範例

```
set analyze_threshold_percent to 15;  
set analyze_threshold_percent to 0.01;
```

```
set analyze_threshold_percent to 0;
```

cast_super_null_on_error

值 (粗體為預設值)

on、off

描述

指定當您嘗試存取物件或陣列元素的不存在成員時，如果查詢是在預設的隨機模式下執行，Amazon Redshift 會傳回NULL值。

datashare_break_glass_session_var

值 (粗體為預設值)

沒有預設值。該值可以是 Amazon Redshift 在發生不建議執行的操作時產生的任何字元字串，如下所述。

描述

套用許可，允許通常不建議資料 AWS Data Exchange 共用的特定操作。

一般而言，建議您不要使用 `ALTER DATASHARE SET PUBLIC ACCESSIBLE` 陳述式捨棄 `DROP DATASHARE` 或修改 AWS Data Exchange 資料共用。若要允許捨棄或修改 AWS Data Exchange 資料共用以關閉可公開存取的設定，請將 `datashare_break_glass_session_var` 變數設定為一次性值。此一次性值由 Amazon Redshift 產生，並且會在初次嘗試有問題的操作後，在錯誤訊息中提供。

將變數設定為一次性產生的值後，請再次執行 `DROP DATASHARE` 或 `ALTER DATASHARE` 陳述式。

如需詳細資訊，請參閱 [ALTER DATASHARE 用量備註](#) 或 [DROP DATASHARE 用量備註](#)。

範例

```
set datashare_break_glass_session_var to '620c871f890c49';
```

datestyle

值 (粗體為預設值)

格式規格 (ISO、Postgres、或 德文) SQL , 以及年/月/日訂購 (DMY、MDY、YMD) 。

- ISO – 使用 HH : MM : SS YYYY-MM-DD 的日期樣式。
- Postgres – 使用 MM-DD HH : MM : SS 的日期樣式 YYYY。
- SQL – 使用 HH : MM : SS MM-DD-YYYY 的日期樣式。
- 德文 – 使用 HH : MM : SS DD-MM-YYYY 的日期樣式。

描述

設定日期和時間值的顯示格式，以及解譯模糊日期輸入值的規則。此字串有兩個參數，可以分開或同時變更。

範例

```
show datestyle;
DateStyle
-----
ISO, MDY
(1 row)

set datestyle to 'SQL,DMY';
```

default_geometry_encoding

值 (粗體為預設值)

1、2

描述

工作階段組態，指定此工作階段期間建立的空間幾何圖形是否使用週框方塊進行編碼。如果 `default_geometry_encoding` 是 1，則不會使用週框方塊對幾何圖形進行編碼。如果 `default_geometry_encoding` 是 2，則使用週框方塊對幾何圖形進行編碼。如需週框方塊支援的相關資訊，請參閱 [邊界框](#)。

describe_field_name_in_uppercase

值 (粗體為預設值)

off (false)、on (true)

描述

指定SELECT陳述式傳回的資料欄名稱是大寫還是小寫。如果此參數為開啟 (on)，會傳回大寫的資料欄名稱。如果此參數為關閉 (off)，會傳回小寫的資料欄名稱。無論 describe_field_name_in_uppercase 的設定為何，Amazon Redshift 都會以小寫方式儲存資料欄名稱。

範例

```
set describe_field_name_in_uppercase to on;

show describe_field_name_in_uppercase;

DESCRIBE_FIELD_NAME_IN_UPPERCASE
-----
on
```

downcase_delimited_identifier

值 (粗體為預設值)

on、off

描述

此組態即將淘汰。反之，請使用 `enable_case_sensitive_identifier`。

讓超級剖析器讀取大寫或混合大小寫JSON的欄位。也啟用聯合查詢支援支援的 PostgreSQL 資料庫，其中包含資料庫、結構描述、資料表和資料欄的混合大小寫名稱。若要使用區分大小寫的識別碼，請將此參數設定為 off。

使用須知

- 如果您使用的是資料列層級安全或動態資料遮罩功能，建議您在叢集或工作群組的參數群組中設定 `downcase_delimited_identifier` 值。這樣可確保在建立和附加政策的過程中 `downcase_delimited_identifier` 保持不變，然後查詢已套用政策的關係。如需有關資料列層級安全性詳細資訊，請參閱 [資料列層級安全性](#)。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。
- 當您將 `downcase_delimited_identifier` 設定為 `off` 並建立資料表時，您可以設定區分大小寫的資料欄名稱。當您將 `downcase_delimited_identifier` 設定為 `on` 並查詢資料表時，資料欄名稱會被變更為小寫。這可能會在 `downcase_delimited_identifier` 設定為 `off` 時產生不同的查詢結果。請思考下列範例：

```
SET downcase_delimited_identifier TO off;
--Amazon Redshift preserves case for column names and other identifiers.

--Create a table with two columns that are identical except for the case.
CREATE TABLE t ("c" int, "C" int);

INSERT INTO t VALUES (1, 2);

SELECT * FROM t;

 c | C
---+---
 1 | 2
(1 row)

SET enable_downcase_delimited_identifier TO on;
--Amazon Redshift no longer preserves case for column names and other identifiers.

SELECT * FROM t;

 c | c
---+---
 1 | 1
(1 row)
```

- 我們建議一般使用者查詢具有動態資料遮罩或資料列層級安全政策的資料表時，使用預設的 `downcase_delimited_identifier` 設定。如需取得資料列層級安全的相關資訊，請參閱 [資料列層級安全性](#)。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。

enable_case_sensitive_identifier

值 (粗體為預設值)

true、false

描述

組態值，用於決定資料庫、資料表和資料欄的名稱識別碼是否區分大小寫。以雙引號括住名稱識別碼時，識別碼的大小寫會保留。當您將 `enable_case_sensitive_identifier` 設定為 `true` 時，名稱識別碼的大小寫會保留。當您將 `enable_case_sensitive_identifier` 設定為 `false` 時，名稱識別碼的大小寫不會保留。

無論 `enable_case_sensitive_identifier` 組態選項的設定為何，都會保留以雙引號括住的 `username` 大小寫。

範例

下列範例說明如何為資料表和資料欄名稱建立及使用區分大小寫的識別碼。

```
-- To create and use case sensitive identifiers
SET enable_case_sensitive_identifier TO true;

-- Create tables and columns with case sensitive identifiers
CREATE TABLE "MixedCasedTable" ("MixedCasedColumn" int);

CREATE TABLE MixedCasedTable (MixedCasedColumn int);

-- Now query with case sensitive identifiers
SELECT "MixedCasedColumn" FROM "MixedCasedTable";

MixedCasedColumn
-----
(0 rows)

SELECT MixedCasedColumn FROM MixedCasedTable;

mixedcasedcolumn
-----
(0 rows)
```

下列範例說明未保留識別碼大小寫的時機。

```
-- To not use case sensitive identifiers
RESET enable_case_sensitive_identifier;

-- Mixed case identifiers are lowercased
CREATE TABLE "MixedCasedTable2" ("MixedCasedColumn" int);

CREATE TABLE MixedCasedTable2 (MixedCasedColumn int);

ERROR:  Relation "mixedcasedtable2" already exists

SELECT "MixedCasedColumn" FROM "MixedCasedTable2";

 mixedcasedcolumn
-----
(0 rows)

SELECT MixedCasedColumn FROM MixedCasedTable2;

 mixedcasedcolumn
-----
(0 rows)
```

使用須知

- 如果您對具體化視觀表使用自動重新整理，建議您在叢集或工作群組的參數群組中設定 `enable_case_sensitive_identifier` 值。這可確保在重新整理具體化視觀表時 `enable_case_sensitive_identifier` 保持不變。如需重新整理具體化視觀表的相關資訊，請參閱 [重新整理具體化視觀表](#)。如需在參數群組中設定組態值的相關資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 參數群組](#)。
- 如果您使用的是資料列層級安全或動態資料遮罩功能，建議您在叢集或工作群組的參數群組中設定 `enable_case_sensitive_identifier` 值。這樣可確保在建立和附加政策的過程中 `enable_case_sensitive_identifier` 保持不變，然後查詢已套用政策的關係。如需有關資料列層級安全性詳細資訊，請參閱 [資料列層級安全性](#)。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。
- 當您將 `enable_case_sensitive_identifier` 設定為 `on` 並建立資料表時，您可以設定區分大小寫的資料欄名稱。當您將 `enable_case_sensitive_identifier` 設定為 `off` 並查詢資料表

時，資料欄名稱會被變更為小寫。這可能會在 `enable_case_sensitive_identifier` 設定為 `on` 時產生不同的查詢結果。請思考下列範例：

```
SET enable_case_sensitive_identifier TO on;
--Amazon Redshift preserves case for column names and other identifiers.

--Create a table with two columns that are identical except for the case.
CREATE TABLE t ("c" int, "C" int);

INSERT INTO t VALUES (1, 2);

SELECT * FROM t;

 c | C
----+----
 1 | 2
(1 row)

SET enable_case_sensitive_identifier TO off;
--Amazon Redshift no longer preserves case for column names and other identifiers.

SELECT * FROM t;

 c | c
----+----
 1 | 1
(1 row)
```

- 我們建議一般使用者查詢具有動態資料遮罩或資料列層級安全政策的資料表時，使用預設的 `enable_case_sensitive_identifier` 設定。如需有關資料列層級安全性詳細資訊，請參閱 [資料列層級安全性](#)。如需動態資料遮罩的詳細資訊，請參閱 [動態資料遮罩](#)。

`enable_case_sensitive_super_attribute`

值 (粗體為預設值)

true、false

描述

一種組態值，可判斷使用非分隔屬性名稱導覽SUPER資料類型結構是否區分大小寫。當您enable_case_sensitive_super_attribute設定為 true 時，使用非分隔屬性名稱導覽SUPER類型結構會區分大小寫。當您將 值設定為 false 時，使用非分隔屬性名稱導覽SUPER類型結構不區分大小寫。

無論 enable_case_sensitive_super_attribute 組態選項的設定為何，以雙引號括住屬性名稱並將 enable_case_sensitive_identifier 設定為 true 時，大小寫一律會保留。

enable_case_sensitive_super_attribute 僅適用於具有SUPER資料類型的資料欄。對於所有其他資料欄，請考慮使用 enable_case_sensitive_identifier 代替。

如需SUPER資料類型的詳細資訊，請參閱 [SUPER 類型](#)和 [Amazon Redshift 中的半結構化資料](#)。

範例

下列範例顯示選取enable_case_sensitive_super_attribute已啟用且已停用SUPER值的結果。

```
--Create a table with a SUPER column.
CREATE TABLE tbl (col SUPER);

--Insert values.
INSERT INTO tbl VALUES (json_parse('{
  "A": "A", "a": "a"
}'));

SET enable_case_sensitive_super_attribute TO ON;

SELECT col.A FROM tbl;
  a
-----
  "A"
(1 row)

SELECT col.a FROM tbl;
  a
-----
  "a"
(1 row)
```

```
SET enable_case_sensitive_super_attribute TO OFF;

SELECT col.A FROM tbl;
  a
-----
"a"
(1 row)

SELECT col.a FROM tbl;
  a
-----
"a"
(1 row)
```

使用須知

- 檢視和具體化視觀表會遵循其建立時的 `enable_case_sensitive_super_attribute` 值。近期繫結檢視、預存程序和使用使用者定義函數都會遵循查詢時的 `enable_case_sensitive_super_attribute` 值。
- 如果您對具體化視觀表使用自動重新整理，建議您在叢集或工作群組的參數群組中設定 `enable_case_sensitive_identifier` value。這可確保在重新整理具體化視觀表時 `enable_case_sensitive_identifier` 保持不變。如需重新整理具體化視觀表的相關資訊，請參閱 [重新整理具體化視觀表](#)。如需在參數群組中設定組態值的相關資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 參數群組](#)。
- 無論 `enable_case_sensitive_super_attribute` 的值為何，陳述式結果中的資料欄名稱一律會變更為小寫。若要讓資料欄名稱區分大小寫，請啟用 `enable_case_sensitive_identifier`。
- 我們建議一般使用者查詢具有資料列層級安全政策的資料表時，使用預設的 `enable_case_sensitive_identifier` 設定。如需取得資料列層級安全的相關資訊，請參閱 [資料列層級安全性](#)。

enable_numeric_rounding

值 (粗體為預設值)

on (true)、ff (false)

描述

指定是否使用數值四捨五入。如果 `enable_numeric_rounding` 是 on，Amazon Redshift 會在將 NUMERIC 值轉換為其他數值類型時四捨五入值，例如 INTEGER 或 DECIMAL。如果 `enable_numeric_rounding` 是 off，Amazon Redshift 會在將 NUMERIC 值轉換為其他數值類型時截斷值。如需數值類型的相關資訊，請參閱 [數值類型](#)。

範例

```
--Create a table and insert the numeric value 1.5 into it.
CREATE TABLE t (a numeric(10, 2));

INSERT INTO t VALUES (1.5);

SET enable_numeric_rounding to ON;
--Amazon Redshift now rounds NUMERIC values when casting to other numeric types.

SELECT a::int FROM t;

 a
---
 2
(1 row)

SELECT a::decimal(10, 0) FROM t;

 a
---
 2
(1 row)

SELECT a::decimal(10, 5) FROM t;

 a
-----
1.50000
(1 row)

SET enable_numeric_rounding to OFF;
```

```
--Amazon Redshift now truncates NUMERIC values when casting to other numeric types.

SELECT a::int FROM t;

  a
---
  1
(1 row)

SELECT a::decimal(10, 0) FROM t;

  a
---
  1
(1 row)

SELECT a::decimal(10, 5) FROM t;

  a
-----
1.50000
(1 row)
```

enable_result_cache_for_session

值 (粗體為預設值)

on (true)、off (false)

描述

指定是否使用查詢結果快取。如果 `enable_result_cache_for_session` 為 on，當提交查詢時，Amazon Redshift 會檢查查詢結果的有效快取副本。如果在結果快取中找到符合者，Amazon Redshift 會使用快取的結果，不會執行查詢。如果 `enable_result_cache_for_session` 為 off，Amazon Redshift 會在提交查詢時忽略結果快取並執行所有查詢。

範例

```
SET enable_result_cache_for_session TO off;
```

```
--Amazon Redshift now ignores the results cache
```

enable_vacuum_boost

值 (粗體為預設值)

false、true

描述

指定是否針對工作階段中執行的所有VACUUM命令啟用真空增強選項。如果 **enable_vacuum_boost** 是 true , Amazon Redshift 會使用 BOOST 選項執行工作階段中的所有 VACUUM 命令。如果 **enable_vacuum_boost** 是 false , Amazon Redshift 預設不會使用 BOOST 選項執行。如需 BOOST 選項的詳細資訊, 請參閱 [VACUUM](#)。

error_on_nondeterministic_update

值 (粗體為預設值)

false、true

描述

指定每列具有多個相符項目的UPDATE查詢是否傳回錯誤。

範例

```
SET error_on_nondeterministic_update TO true;

CREATE TABLE t1(x1 int, y1 int);

CREATE TABLE t2(x2 int, y2 int);

INSERT INTO t1 VALUES (1,10), (2,20), (3,30);

INSERT INTO t2 VALUES (2,40), (2,50);

UPDATE t1 SET y1=y2 FROM t2 WHERE x1=x2;
```

```
ERROR: Found multiple matches to update the same tuple.
```

extra_float_digits

值 (粗體為預設值)

0、-15 到 2

描述

設定浮點值 (包括 float4 和 float8) 顯示的位數。該值會新增至標準位數 (FLT_DIG 或 DBL_DIG, 視情況而定)。該值最高可以設定為 2, 以包括部分有效數字。這對於輸出必須完全還原的浮點數資料特別有用。或者, 可將此值設為負數, 隱匿不要的數字。

範例

下列範例會將 extra_float_digits 設定為 -2。首先, 顯示目前的參數設定。

```
show all;
  name                | setting
-----+-----
analyze_threshold_percent | 10
datestyle              | ISO, MDY
extra_float_digits     | 2
query_group           | default
search_path            | $user, public
statement_timeout      | 0
timezone               | UTC
wlm_query_slot_count  | 1
```

然後, 將新值設定為 -2。

```
set extra_float_digits to -2;
```

最後顯示更新的參數設置。

```
show all;
```

name	setting
analyze_threshold_percent	10
datestyle	ISO, MDY
extra_float_digits	-2
query_group	default
search_path	\$user, public
statement_timeout	0
timezone	UTC
wlm_query_slot_count	1

interval_forbid_composite_literals

值 (粗體為預設值)

false、true

描述

修改包含 YEAR TO MONTH和 DAY TO SECOND部分的間隔值的工作階段組態。

如果 `interval_forbid_composite_literals` 是 true，則如果同時遇到具有 YEAR TO MONTH 和 DAY TO SECOND 部分的間隔，則會傳回錯誤。例如，以下 SQL 包含 SECOND 具有 INTERVAL DAY TO MONTH 和 YEAR TO SECOND 部分的 DAY TO。

```
SELECT INTERVAL '1 year 1 day' DAY TO SECOND;
ERROR: Interval Day To Second literal cannot contain year-month parts. Disable the GUC
interval_forbid_composite_literals to suppress this error and silently discard the
year-month part.
```

如果 `interval_forbid_composite_literals` 是 false，Amazon Redshift 會抑制錯誤，並從 YEAR TO SECOND 值截斷 INTERVAL DAY TO MONTH 部分。例如，以下 SQL 包含 SECOND 具有 INTERVAL DAY TO MONTH 和 YEAR TO SECOND 部分的 DAY TO。

```
SET interval_forbid_composite_literals to "false";
SELECT INTERVAL '1 year 1 day' DAY TO SECOND;
```

```
intervald2s
-----
```

```
1 days 0 hours 0 mins 0.0 secs
```

json_serialization_enable

值 (粗體為預設值)

false、true

描述

修改 ORC、**JSON**離子和 Parquet 格式資料的JSON序列化行為的工作階段組態。如果 `json_serialization_enable` 是 true，則所有頂層集合會自動序列化為 `JSON` 並傳回為 VARCHAR (65535)。非複雜資料欄不會受到影響或序列化。由於集合資料欄會序列化為 VARCHAR (65535)，因此無法再直接存取其巢狀子欄位作為查詢語法的一部分（亦即，在篩選子句中）。如果 `json_serialization_enable` 是 false，頂層集合不會序列化為 JSON。如需巢狀 JSON 序列化的詳細資訊，請參閱 [序列化複雜的嵌套 JSON](#)。

json_serialization_parse_nested_strings

值 (粗體為預設值)

false、true

描述

修改 ORC、**JSON**離子和 Parquet 格式資料的JSON序列化行為的工作階段組態。當 `json_serialization_parse_nested_strings` 和 `json_serialization_enable` 皆為 true 時，以複雜類型（例如，地圖、結構或陣列）儲存的字串值，如果有效，會直接剖析並寫入結果 JSON。如果 `json_serialization_parse_nested_strings` 為 false，巢狀複雜類型中的字串會序列化為逸出 JSON 字串。如需詳細資訊，請參閱 [序列化包含字JSON字符串的複雜類型](#)。

max_concurrency_scaling_clusters

值 (粗體為預設值)

1、0 到 10

描述

設定當並行擴展啟用時，允許的並行擴展叢集的最大數量。來看需要更多的並行擴展，請提高此數值。降低此數值可降低並行擴展叢集的用量，以及相關的帳單費用。

並行擴展叢集的數量上限是一項可調整的配額。如需詳細資訊，請參閱《Amazon Redshift 管理指南》中的 [Amazon Redshift 配額](#)。

max_cursor_result_set_size

值 (粗體為預設值)

0 (預設為上限值) - 14400000 MB

描述

`max_cursor_result_set_size` 參數已不再繼續使用。如需游標結果集大小的相關資訊，請參閱 [游標限制條件](#)。

mv_enable_aqmv_for_session

值 (粗體為預設值)

true、false

描述

指定 Amazon Redshift 是否可以在工作階段層級上執行具體化視觀表的自動查詢重寫。

navigate_super_null_on_error

值 (粗體為預設值)

on、off

描述

指定當您嘗試導覽物件或陣列元素的不存在成員時，如果查詢是在預設的隨機模式下執行，Amazon Redshift 會傳回NULL值。

parse_super_null_on_error

值 (粗體為預設值)

off、on

描述

指定當 Amazon Redshift 嘗試剖析物件或陣列元素的不存在成員時，如果查詢是在嚴格模式下執行，Amazon Redshift 會傳回NULL值。

pg_federation_repeatable_read

值 (粗體為預設值)

true、false

描述

指定 PostgreSQL 資料庫結果的聯合查詢交易隔離層級。

- 當 `pg_federation_repeatable_read` 為 true 時，聯合交易會使用REPEATABLEREAD隔離層級語義處理。此為預設值。
- 當 `pg_federation_repeatable_read` 為 false 時，聯合交易會使用READCOMMITTED隔離層級語義處理。

如需詳細資訊，請參閱下列內容：

- [使用 Amazon Redshift 存取聯合資料時的注意事項](#).
- [管理並行寫入操作](#).

範例

下列命令會將工作階段的 `pg_federation_repeatable_read` 設定為 on。show 命令會顯示已設定值的值。

```
set pg_federation_repeatable_read to on;
```

```
show pg_federation_repeatabl_read;

pg_federation_repeatabl_read
-----
on
```

query_group

值 (粗體為預設值)

沒有預設值；可以是任何字元字串。

描述

將使用者定義的標籤套用至在相同工作階段中執行的查詢 (相同查詢群組)。此標籤會在查詢日誌中擷取。您可以使用它來限制來自 STL_QUERY 和 STV_INFLIGHT 資料表和 SVL_QLOG 檢視的結果。例如，您可以將個別標籤套用至您執行的每個查詢，以唯一識別查詢，而不必查詢其 IDs。

此參數不存在於伺服器組態檔案中，必須在執行期使用 SET 命令設定。雖然您可以使用長字元字串作為標籤，但標籤在 STL_QUERY 資料表和 SVL_QLOG 檢視的 LABEL 欄中截斷為 30 個字元 (在 STV_ 中截斷為 15 個字元 INFLIGHT)。

以下範例將 query_group 設為 **Monday**，然後以此標籤執行數個查詢。

```
set query_group to 'Monday';
SET
select * from category limit 1;
...
...
select query, pid, substring, elapsed, label
from svl_qlog where label = 'Monday'
order by query;
```

query	pid	substring	elapsed	label
789	6084	select * from category limit 1;	65468	Monday
790	6084	select query, trim(label) from ...	1260327	Monday
791	6084	select * from svl_qlog where ..	2293547	Monday
792	6084	select count(*) from bigsales;	108235617	Monday
...				

search_path

值 (粗體為預設值)

'\$user', public, schema_names

以逗號分隔的現有結構描述名稱清單。如果有 '\$user'，清單會取代和 SESSION_USER 相同名稱的結構描述；如果沒有，則會忽略清單。

描述

當以簡單名稱參考物件 (如資料表或函數) 且不提供結構描述元件時，指定搜尋結構描述的順序：

- 外部結構描述和外部資料表不支援搜尋路徑。外部資料表必須以外部結構描述明確限定。
- 建立物件時若沒有指定特定目標結構描述，系統會將物件置於搜尋路徑中列出的第一個結構描述。如果搜尋路徑是空的，系統會傳回錯誤。
- 當其他結構描述中有同名的物件時，會使用搜尋路徑中先找到的那一個。
- 物件若不在搜尋路徑裡的任何結構描述中，則只能使用它的完整名稱 (指定包含它的結構描述) 來參考它。
- 一定會搜尋系統目錄結構描述 pg_catalog。如果路徑中有它，會依指定順序搜尋。如果沒有，會先搜尋它再搜尋路徑中的項目。
- 一定會搜尋目前工作階段的暫時資料表結構描述 pg_temp_nnn (如果存在)。可以使用 alias pg_temp 將它明確列在路徑中。如果路徑中沒有它，則會先搜尋它 (甚至在 pg_catalog 之前)。不過，只會搜尋暫時結構描述中的關係名稱 (資料表、名稱)。不會搜尋函數名稱。

範例

下列範例會建立結構描述，ENTERPRISE並將 search_path 設定為新的結構描述。

```
create schema enterprise;
set search_path to enterprise;
show search_path;

 search_path
-----
 enterprise
(1 row)
```

下列範例會將結構描述新增至ENTERPRISE預設 search_path。

```
set search_path to '$user', public, enterprise;
show search_path;
```

```

      search_path
-----
"$user", public, enterprise
(1 row)
```

下列範例會將資料表新增至FRONTIER結構描述 ENTERPRISE。

```
create table enterprise.frontier (c1 int);
```

當在相同資料庫中建立資料表 PUBLIC.FRONTIER 且使用者未在查詢中指定結構描述名稱時，PUBLIC.FRONTIER 優先於 ENTERPRISE.FRONTIER。

```
create table public.frontier(c1 int);
insert into enterprise.frontier values(1);
select * from frontier;
```

```

frontier
----
(0 rows)
```

```
select * from enterprise.frontier;
```

```

c1
----
1
(1 row)
```

spectrum_enable_pseudo_columns

值 (粗體為預設值)

true、false

描述

您可以藉由將 `spectrum_enable_pseudo_columns` 組態參數設定為 `false`，以停用工作階段的虛擬資料欄建立。

範例

下列命令會停用工作階段的虛擬資料欄建立。

```
set spectrum_enable_pseudo_columns to false;
```

`enable_spectrum_oid`

值 (粗體為預設值)

true、false

描述

您也可以將 `enable_spectrum_oid` 組態參數設定為 `false`，僅停用 `$spectrum_oid` 虛擬資料欄。

範例

下列命令會透過將 `enable_spectrum_oid` 組態參數設定為 `false` 來停用 `$spectrum_oid` 虛擬資料欄。

```
set enable_spectrum_oid to false;
```

`spectrum_query_maxerror`

值 (粗體為預設值)

-1、整數

描述

您可以指定整數來指出取消查詢之前可接受的錯誤數目上限。負值會關閉最誤資料處理上限。結果會記錄在 [SVL_SPECTRUM_SCAN_ERROR](#) 中。

範例

下列範例假設包含多餘字元和無效字元ORC的資料。my_string 的資料欄定義會指定 3 個字元的長度。以下是此範例的範例資料：

```
my_string
-----
abcdef
gh◆
ab
```

下列命令會將錯誤的數目上限設定為 1 並執行查詢。

```
set spectrum_query_maxerror to 1;
SELECT my_string FROM orc_data;
```

查詢會停止並將結果記錄到 [SVL_SPECTRUM_SCAN_ERROR](#)。

statement_timeout

值 (粗體為預設值)

0 (關閉限制)、x 毫秒

描述

停止任何超過指定毫秒數的陳述式。

statement_timeout 值是指查詢在 Amazon Redshift 將其終止前可執行的最長時間。此時間包括規劃、在工作負載管理 (WLM) 中佇列，以及執行時間。將此時間與WLM逾時 (max_execution_time) 和 QMR (query_execution_time) 進行比較，其中僅包含執行時間。

如果也指定WLM逾時 (max_execution_time) 作為WLM組態的一部分，則會使用 statement_timeout 和 max_execution_time 的較低值。如需詳細資訊，請參閱[WLM 逾時](#)。

範例

以下查詢因為時間超過 1 毫秒，所以會逾時並被取消。

```
set statement_timeout = 1;

select * from listing where listid>5000;
ERROR: Query (150) canceled on user's request
```

stored_proc_log_min_messages

值 (粗體為預設值)

LOG, INFO, NOTICE, WARNING, EXCEPTION

描述

指定引發預存程序訊息的最低記錄層級。系統會記錄指定層級或高於指定層級的訊息。預設值為 LOG (記錄所有訊息)。記錄層級從最高到最低的順序如下：

1. EXCEPTION
2. WARNING
3. NOTICE
4. INFO
5. LOG

例如，如果您指定的值NOTICE，則訊息只會記錄 NOTICE、WARNING和 EXCEPTION。

timezone

值 (粗體為預設值)

UTC，時區

語法

```
SET timezone { TO | = } [ time_zone | DEFAULT ]

SET time zone [ time_zone | DEFAULT ]
```


描述

設定目前工作階段的時區。時區可以是與通用協調時間（UTC）或時區名稱的偏移。

Note

您不能使用叢集參數群組設定此 `timezone` 組態參數。只能使用 `SET` 命令為目前的工作階段設定時區。若要設定特定資料庫使用者執行之所有工作階段的時區，請使用 [ALTER USER](#) 命令。`ALTER USER ...` 會 `SET TIMEZONE` 變更後續工作階段的時區，而不是目前工作階段的時區。

當您使用 `SET timezone (一個字) 命令` 搭配 `T0` 或 `設定時區時=`，您可以將 `time_zone` 指定為時區名稱、POSIX 樣式格式偏移或 ISO-8601 格式偏移，如下所示。

```
SET timezone { T0 | = } time_zone
```

當您使用不含 `T0` 或 `SET` 的時區命令 `設定時區時=`，您可以使用 `INTERVAL` 和時區名稱、POSIX 樣式格式偏移或 ISO-8601 格式偏移來指定 `time_zone`，如下所示。

```
SET time zone time_zone
```

時區格式

Amazon Redshift 支援以下時區格式：

- 時區名稱
- INTERVAL
- POSIX 樣式時區規格
- ISO-8601 偏移


由於時區縮寫，例如 `PST` 或 `PDT`，定義為來自的固定偏移 UTC，且不包含日光節約時間規則，因此 `SET` 命令不支援時區縮寫。

請參閱以下內容，了解時區格式的詳細資訊。

時區名稱 – 完整的時區名稱，例如 `America/New_York`。完整時區名稱可能包含日光節約時間規則。

以下是完整時區名稱的範例：

- Etc/Greenwich
- America/New_York
- CST6CDT
- GB

 Note

許多時區名稱，例如 EST、MSTNZ 和 UCT，也是縮寫。

若要查看有效時區名稱的清單，請執行下列命令。

```
select pg_timezone_names();
```

INTERVAL – 來自的偏移UTC。例如，PST是 -8 : 00 或 -8 小時。

以下是INTERVAL時區偏移的範例：

- -8:00
- -8 小時
- 30 分鐘

POSIX樣式格式 – 形式STDOffset或 中的時區規格STDOffsetDST，其中 STD 是時區縮寫，偏移是從以西數小時為單位的數值偏移UTC，並且DST是選用的日光節約區域縮寫。日光節約時間假設為較給定偏移早一小時。

POSIX樣式時區格式使用 Greenwich 西部的正偏移，與使用 Greenwich 東部的正偏移的 ISO-8601 慣例不同。

以下是 POSIX樣式時區的範例：

- PST8
- PST8PDT
- EST5
- EST5EDT

Note

Amazon Redshift 不會驗證 POSIX 樣式的時區規格，因此可以將時區設定為無效的值。例如，即使將時區設為無效值，下列命令不會傳回錯誤。

```
set timezone to 'xxx36';
```

ISO-8601 偏移 – 來自 UTC 的偏移，格式為 \pm [hh]:[mm]。

以下是 ISO-8601 偏移的範例：

- -8:00
- +7:30

範例

以下範例會將目前工作階段的時區設為 New York。

```
set timezone = 'America/New_York';
```

下列範例會將目前工作階段的時區設定為 UTC-8 (PST)。

```
set timezone to '-8:00';
```

下列範例使用 將時區INTERVAL設定為 PST。

```
set timezone interval '-8 hours'
```

下列範例會將目前工作階段的時區重設為系統預設時區 (UTC)。

```
set timezone to default;
```

若要設定資料庫使用者的時區，請使用 ... ALTER USER SET 陳述式。以下範例會將 dbuser 的時區設為 New York。新值將在使用者的後續工作階段中延用。

```
ALTER USER dbuser SET timezone to 'America/New_York';
```

use_fips_ssl

值 (粗體為預設值)

true、false

描述

指定是否使用 FIPS 合規 SSL 模式的參數群組值。如果 use_fips_ssl 是 true，則會使用 FIPS 合規 SSL 模式。如果 use_fips_ssl 是 false，則不會使用 FIPS 合規 SSL 模式。如需詳細資訊，請參閱 Amazon Redshift 管理指南 中的 [設定連線的安全選項](#)。

若要設定 Amazon Redshift 佈建叢集的參數，請參閱 Amazon Redshift 管理指南 中的 [關於參數群組](#)。若要設定 Redshift Serverless 的參數，請參閱 [Amazon Redshift 管理指南](#) 和 或 [Redshift Serverless 參考](#) 中的 [設定 FIPS 與 Amazon Redshift Serverless 相容的 SSL 連線](#)。 [CreateWorkgroup](#) [UpdateWorkgroup](#) API

wlm_query_slot_count

值 (粗體為預設值)

1、1 到 50 (不得超過服務類別可使用的槽數量 (並行層級))

描述

設定查詢所使用的查詢槽數量。

工作負載管理 (WLM) 根據為佇列設定的並行層級，保留服務類別中的插槽。例如，如果並行層級設定為 5，則服務類別會有 5 個插槽。WLM 會將服務類別的可用記憶體平均配置到每個插槽。如需詳細資訊，請參閱 [工作負載管理](#)。

Note

如果 wlm_query_slot_count 的值大於服務類別可使用的槽數量 (並行層級)，這時查詢會失敗。如果發生錯誤，請將 wlm_query_slot_count 減為允許的值。

對於效能受配置記憶體嚴重影響的操作，例如清空，增加 wlm_query_slot_count 的值可以提高效能。特別是，對於慢真空命令，請在 SVV_VACUUM_SUMMARY 檢視中檢查對應的記錄。如果您在

__SVVACUUM檢視中看到 `sort_partitions` 和 `merge_increments` SUMMARY的高值（接近或高於 100），請考慮在下次針對該資料表執行 Vacuum 時增加 `wlm_query_slot_count` 的值。

增加 `wlm_query_slot_count` 值會限制可執行的並行查詢數量。例如，假設服務類別的並行層級為 5，且 `wlm_query_slot_count` 設為 3。在 `wlm_query_slot_count` 為 3 的工作階段中執行查詢時，最多可以在相同服務類別中再執行 2 個並行的查詢。之後的查詢會在佇列中等待，直到執行中的查詢完成且槽空出來。

範例

使用 SET命令來設定目前工作階段持續時間內 `wlm_query_slot_count` 的值。

```
set wlm_query_slot_count to 3;
```

文件歷史記錄

Note

如需 Amazon Redshift 中新功能的描述，請參閱[新功能](#)。

下表說明 2018 年 5 月之後 Amazon Redshift 資料庫開發人員指南的重要文件變更。如需本文件更新通知，您可以訂閱RSS摘要。

API 版本：2012-12-01

如需《Amazon Redshift 管理指南》的變更清單，請參閱[Amazon Redshift 管理指南文件歷史記錄](#)。

如需新功能的相關資訊，包括每個版本的修正清單以及關聯的叢集版本編號，請參閱[叢集版本歷史記錄](#)。

變更	描述	日期
支援空間 3D 和 4D 幾何圖形以及新的空間函數	您現在可以使用額外的空間函數，且 3D 和 4D 幾何圖形支援已新增到某些函數中。	2021 年 8 月 19 日
支援資料欄壓縮編碼以實現自動資料表最佳化	您可以指定資料表E NCODEAUTO的選項，以自動管理資料表中所有資料欄的壓縮編碼。	2021 年 8 月 3 日
支援使用 Amazon Redshift Data 的多個SQL陳述式或具有參數的SQL陳述式 API	您現在可以使用 Amazon Redshift Data 執行多個SQL陳述式或具有參數的陳述式API。	2021 年 7 月 28 日
支援搭配資料欄層級覆寫使用不區分大小寫定序	您現在可以使用CREATEDATABASE陳述式中的COLLATE 子句來指定預設定序。	2021 年 6 月 24 日

支援跨帳戶共用資料	您現在可以跨 共用資料 AWS 帳戶。	2021 年 4 月 30 日
支援具有遞迴的階層式資料查詢 CTE	您現在可以在 中使用遞迴通用資料表表達式 (CTE) SQL。	2021 年 4 月 29 日
支援跨資料庫查詢	您現在可以在叢集中的各個資料庫上查詢資料。	2021 年 3 月 10 日
支援 COPY和 UNLOAD命令的精細存取控制	您現在可以將執行 COPY和 UNLOAD 命令的權限授予 Amazon Redshift 叢集中的特定使用者和群組，以建立更精細的存取控制政策。	2021 年 1 月 12 日
支援原生JSON和半結構化資料	您現在可以定義SUPER資料類型。	2020 年 12 月 9 日
支援 My 聯合查詢SQL	您現在可以將聯合查詢寫入支援的我的SQL引擎。	2020 年 12 月 9 日
支援資料共用	您可以在 Amazon Redshift 叢集之間共用資料。	2020 年 12 月 9 日
支援自動資料表最佳化	您現在可以定義自動的分佈和排序索引鍵。	2020 年 12 月 9 日
支援 Amazon Redshift ML	您現在可以建立、訓練和部署機器學習 (ML) 模型。	2020 年 12 月 8 日
支援具體化視觀表的自動重新整理和查詢重寫	您現在可以使用自動重新整理來保留具體化檢視 up-to-date，並且可以透過自動重寫來改善查詢效能。	2020 年 11 月 11 日

支援 TIME 和 TIMETZ 資料類型	您現在可以使用 TIME 和 TIMETZ 資料類型建立資料表。TIME 資料類型會儲存一天中沒有時區資訊的時間，並 TIMETZ 儲存一天中的時間，包括時區資訊	2020 年 11 月 11 日
支援 Lambda UDFs 和權杖化	您現在可以撰寫 Lambda UDFs 來啟用資料的外部權杖化。	2020 年 10 月 26 日
支援修改資料表的資料欄編碼	您現在可以修改資料表的資料欄編碼。	2020 年 10 月 20 日
支援跨資料庫查詢	Amazon Redshift 現在可查詢叢集中的各個資料庫。	2020 年 10 月 15 日
支援 HyperLogLog 草圖	Amazon Redshift 現在可儲存和處理 HyperLogLogSketches。	2020 年 10 月 2 日
支援 Apache Hudi 和 Delta Lake	建立 Redshift Spectrum 外部資料表的增強功能。	2020 年 9 月 24 日
支援查詢空間資料的增強功能	增強功能包括載入形狀檔和數個新的空間 SQL 函數。	2020 年 9 月 15 日
具體化視觀表支援外部資料表	您可以在 Amazon Redshift 中建立參考外部資料來源的具體化視觀表。	2020 年 6 月 19 日
支援寫入外部資料表	您可以執行 CREATE EXTERNAL TABLE AS SELECT 來寫入外部資料表，方法是寫入新的外部資料表 INSERT INTO，或將資料插入現有的外部資料表。	2020 年 6 月 8 日
支援結構描述的儲存控制	管理結構描述之儲存控制項的命令和檢視更新。	2020 年 6 月 2 日

支援聯合查詢一般可用性	更新有關使用聯合查詢來查詢資料的資訊。	2020 年 4 月 16 日
支援其他空間函數	新增其他空間函數的描述。	2020 年 4 月 2 日
支援具體化檢視的一般可用性	從叢集版本 1.0.13059 開始全面推出具體化檢視。	2020 年 2 月 19 日
支援資料欄層級權限	從叢集版本 1.0.13059 開始，即可使用資料欄層級的權限。	2020 年 2 月 19 日
ALTER TABLE	您可以使用 ALTERTABLE命令搭配 ALTERDISTSTYLEALL子句來變更資料表的分佈樣式。	2020 年 2 月 11 日
聯合查詢支援	更新指南，以使用更新的 CREATE EXTERNAL 描述聯合查詢SCHEMA。	2019 年 12 月 3 日
資料湖匯出支援	更新指南，說明 UNLOAD命令的新參數。	2019 年 12 月 3 日
空間資料支援	更新指南以說明空間資料的支援。	2019 年 11 月 21 日
新主控台支援	更新指南以說明新的 Amazon Redshift 主控台。	2019 年 11 月 11 日
自動資料表排序支援	Amazon Redshift 可以自動排序資料表資料。	2019 年 11 月 7 日
支援 VACUUM BOOST 選項	您可以在清空資料表時使用 BOOST選項。	2019 年 11 月 7 日
支援預設IDENTITY資料欄	您可以使用預設IDENTITY資料欄建立資料表。	2019 年 9 月 19 日
支援AZ64壓縮編碼	您可以使用AZ64壓縮編碼來編碼某些資料欄。	2019 年 9 月 19 日

支援查詢優先順序	您可以設定自動WLM佇列的查詢優先順序。	2019 年 8 月 22 日
的支援 AWS Lake Formation	您可以將 Lake Formation Data Catalog 與 Amazon Redshift Spectrum 搭配使用。	2019 年 8 月 8 日
COMPUPDATE PRESET	您可以使用 COPY命令COMPUPDATEPRESET，讓 Amazon Redshift 選擇壓縮編碼。	2019 年 6 月 13 日
ALTER COLUMN	您可以使用 ALTABLE命令ALTERCOLUMN來增加VARCHAR資料欄的大小。	2019 年 5 月 22 日
支援預存程序	您可以在 Amazon Redshift 中定義 PL/pgSQL 預存程序。	2019 年 4 月 24 日
支援自動工作負載管理 (WLM) 組態	您可以啟用 Amazon Redshift 搭配自動 執行WLM。	2019 年 4 月 24 日
UNLOAD 至 Zstandard	您可以使用 UNLOAD命令，將 Zstandard 壓縮套用至卸載至 Amazon S3 的文字和逗號分隔值 (CSV) 檔案。	2019 年 4 月 3 日
並行擴展	當並行擴展啟用時，當您需要更多叢集容量以執行增加的並行讀取查詢時，Amazon Redshift 將會自動新增額外的叢集容量。	2019 年 3 月 21 日
UNLOAD 至 CSV	您可以使用 UNLOAD命令卸載至格式化為CSV文字的檔案。	2019 年 3 月 13 日

AUTO 分佈樣式	若要啟用自動分佈，您可以使用 CREATE TABLE 陳述式指定AUTO分佈樣式。啟用自動分佈時，Amazon Redshift 會根據資料表資料來指派最佳分佈樣式。分佈是在幾秒內於背景中發生變化。	2019 年 1 月 23 日
COPY 從 Parquet 支援 SMALLINT	COPY 現在支援將 Parquet 格式檔案載入使用SMALLINT資料類型的欄。如需詳細資訊，請參閱 COPY Columnar Data Formats 中的	2019 年 1 月 2 日
DROP EXTERNAL DATABASE	您可以將 DROPEXTERNALDATABASE子句與 DROPSHEMA 命令一起包含，以捨棄外部資料庫。	2018 年 12 月 3 日
跨區域 UNLOAD	您可以透過指定 REGION 參數 UNLOAD，前往另一個 AWS 區域中的 Amazon S3 儲存貯體。	2018 年 10 月 31 日
自動清空刪除	Amazon Redshift 會自動在背景執行 VACUUMDELETE 操作，因此您很少需要執行 DELETEONLY吸塵。Amazon Redshift 會將 排程VACUUMDELETE在負載減少期間執行，並在負載過高期間暫停操作。	2018 年 10 月 31 日

自動分佈	當您未指定具有 CREATETABLE 陳述式的分佈樣式時，Amazon Redshift 會根據資料表資料指派最佳分佈樣式。分佈是在幾秒內於背景中發生變化。	2018 年 10 月 31 日
的精細存取控制 AWS Glue Data Catalog	您現在可以指定對存放在 中的資料的存取層級 AWS Glue Data Catalog。	2018 年 10 月 15 日
UNLOAD 使用資料類型	您可以使用 UNLOAD 命令指定 MANIFESTVERBOSE 選項，將中繼資料新增至資訊清單檔案，包括資料欄的名稱和資料類型、檔案大小和資料列計數。	2018 年 10 月 10 日
使用單一-ALTERTABLE陳述式新增多個分割區	對於 Redshift Spectrum 外部資料表，您可以在單一- ALTERTABLEADD 陳述式中結合多個 PARTITION 子句。如需詳細資訊，請參閱 更改外部資料表範例 。	2018 年 10 月 10 日
UNLOAD 含標頭	您可以使用 UNLOAD 命令指定 HEADER 選項，以新增在每個輸出檔案頂端包含資料欄名稱的標頭行。	2018 年 9 月 19 日
新的系統資料表和檢視	新增 SVL_S3Retries 、 SVL_USER_INFO 和 STL_DISK_FULL_DIAG 文件。	2018 年 8 月 31 日

在 Amazon Redshift Spectrum 中支援巢狀資料	您現在可以查詢 Amazon Redshift Spectrum 資料表中儲存的巢狀資料。如需詳細資訊，請參閱 教學課程：使用 Amazon Redshift Spectrum 查詢巢狀資料 。	2018 年 8 月 8 日
SQA 預設為開啟	所有新叢集現在都預設為啟用短查詢加速 (SQA)。SQA 使用機器學習來提供更高的效能、更快的結果，以及更好的查詢執行時間可預測性。如需詳細資訊，請參閱 短期查詢加速 。	2018 年 8 月 8 日
Amazon Redshift Advisor	針對如何提高叢集效能和降低運營成本，您現在可以從 Amazon Redshift Advisor 獲得量身訂做的建議。如需詳細資訊，請參閱 Amazon Redshift Advisor 。	2018 年 7 月 26 日
立即參考別名	現在，您可以在定義別名表達式後立即參考它。如需詳細資訊，請參閱 SELECT 列出 。	2018 年 7 月 18 日
建立外部資料表時指定壓縮類型	現在，您可以在使用 Amazon Redshift Spectrum 建立外部資料表時指定壓縮類型。如需詳細資訊，請參閱 建立外部資料表 。	2018 年 6 月 27 日
PG_LAST_UNLOAD_ID	為新的系統資訊函數新增的文件：PG_LASTUNLOAD_ID。如需詳細資訊，請參閱 PGLAST_UNLOAD_ID 。	2018 年 6 月 27 日

[ALTER TABLE RENAME COLUMN](#)

ALTER TABLE 現在支援為外部資料表重新命名資料欄。如需詳細資訊，請參閱[更改外部資料表範例](#)。

2018 年 6 月 7 日

舊版更新

下表會說明 2018 年 6 月之前，《Amazon Redshift 資料庫開發人員指南》每個版本的重要變更。

變更	描述	變更日期
COPY 從 Parquet 包含 SMALLINT	COPY 現在支援將 Parquet 格式檔案載入使用 SMALLINT 資料類型的欄。如需詳細資訊，請參閱 COPY 從資料欄資料格式	2019 年 1 月 2 日
COPY 從資料欄格式	COPY 現在支援從使用 Parquet 和 ORC 資料欄資料格式的 Amazon S3 上的檔案載入。如需詳細資訊，請參閱 COPY 從資料欄資料格式	2018 年 5 月 17 日
的動態最長執行時間 SQA	根據預設，工作負載管理 (WLM) 現在會根據叢集工作負載的分析，動態指派短查詢加速 (SQA) 最長執行時間的值。如需詳細資訊，請參閱 短期查詢最長執行時間 。	2018 年 5 月 17 日
STL_LOAD_ 中的新資料欄 COMMIT	STL_LOAD_COMMITS 系統資料表具有新的資料欄 file_format 。	2018 年 5 月 10 日
STL_HASHJOIN 和其他系統日誌資料表中的新資料欄	STL_HASHJOIN 系統資料表有三個新的資料欄：hash_segment、hash_step 和 checksum。此外，checksum 也新增至 STL_MERGEJOIN、STL_NESTLOOP、STL_HASH、STL_SCAN、STL_SORT、_LIMIT、STL_ 和 STL_PROJECT。	2018 年 5 月 17 日
STL_ 中的新資料欄 AGGR	STL_AGGR 系統資料表有兩個新的資料欄，resizes 以及 flushable 。	2018 年 4 月 19 日

變更	描述	變更日期
REGEX 函數的新選項	對於 REGEXP_INSTR 和 REGEXP_SUBSTR 函數，您現在可以指定要使用的相符項目，以及是否執行區分大小寫的相符項目。REGEXP_INSTR 也可讓您指定是否要傳回相符項目的第一個字元位置，或是相符項目結束後的第一個字元位置。	2018 年 3 月 22 日
系統資料表中的新資料欄	STL_COMMIT_STATS 系統資料表中新增 tombstone dblocks、tossedblocks、batched_by 資料欄。 STV_SLICES 系統檢視中新增 localslice 資料欄。	2018 年 3 月 22 日
新增和丟棄外部資料表中的資料欄	ALTER TABLE 現在支援 ADDCOLUMN和 DROP COLUMN Amazon Redshift Spectrum 外部資料表。	2018 年 3 月 22 日
Redshift Spectrum 新 AWS 區域	Redshift Spectrum 現已在孟買和聖保羅區域提供。如需支援的區域的清單，請參閱 Amazon Redshift Spectrum 區域 。	2018 年 3 月 22 日
資料表限制增加至 20,000	現在，8xlarge 叢集節點類型的資料表數目上限是 20,000。大型與特大型節點類型上限則是 9,900。如需詳細資訊，請參閱 限制和配額 。	2018 年 3 月 13 日
Redshift Spectrum 支援 JSON和 Ion	使用 Redshift Spectrum，您可以參考 JSON或 Ion 資料格式中具有純量資料的檔案。如需詳細資訊，請參閱 CREATE EXTERNAL TABLE 。	2018 年 2 月 26 日
IAM Redshift Spectrum 的角色鏈結	您可以連結 AWS Identity and Access Management (IAM) 角色，讓您的叢集可以擔任未連接到叢集的其他角色，包括屬於另一個 AWS 帳戶的角色。如需詳細資訊，請參閱 Amazon Redshift Spectrum 中的鏈結IAM 角色 。	2018 年 2 月 1 日
ADD PARTITION 支援 IF NOT EXISTS	的 ADD PARTITION 子句ALTERTABLE現在支援 IF NOTEXISTS選項。如需詳細資訊，請參閱 ALTER TABLE 。	2018 年 1 月 11 日

變更	描述	變更日期
DATE 外部資料表的資料	Redshift Spectrum 外部資料表現在支援DATE資料類型。如需詳細資訊，請參閱 CREATE EXTERNAL TABLE 。	2018 年 1 月 11 日
Redshift Spectrum 新 AWS 區域	Redshift Spectrum 現已在新加坡、雪梨、首爾、法蘭克福區域提供。如需支援的 AWS 區域清單，請參閱 Amazon Redshift Spectrum 區域 。	2017 年 11 月 16 日
Amazon Redshift 工作負載管理 (WLM) 中的短查詢加速	短查詢加速 (SQA) 會優先處理選取的短執行查詢，然後再處理較長的查詢。SQA 在專用空間中執行短執行查詢，這樣SQA查詢就不會強制在較長查詢後的佇列中等待。使用 SQA時，短時間查詢會開始更快速地執行，使用者會更快看到結果。如需詳細資訊，請參閱 短期查詢加速 。	2017 年 11 月 16 日
WLM 重新指派跳轉查詢	Amazon Redshift 工作負載管理 (WLM) 現在不會取消和重新啟動跳轉查詢，而是將合格的查詢重新指派給新的佇列。WLM 重新指派查詢時，它會將查詢移至新佇列並繼續執行，以節省時間和系統資源。不符合重新指派資格的跳轉查詢，則會重新啟動或取消。如需詳細資訊，請參閱 WLM 查詢佇列跳轉 。	2017 年 11 月 16 日
使用者存取系統日誌	根據預設，在大部份使用者可查看的系統日誌資料表中，一般使用者無法查看另一位使用者產生的資料列。若要允許一般使用者查看使用者可見資料表中的所有資料列，包括其他使用者產生的資料列，請執行 ALTER USER 或 並將 SYSLOG ACCESS 參數 CREATE USER 設定為 UNRESTRICTED。	2017 年 11 月 16 日
結果快取	有了 結果快取 ，當您執行查詢時，Amazon Redshift 會快取結果。當您再次執行查詢，Amazon Redshift 會檢查有沒有該查詢結果的有效快取副本。如果在結果快取中找到符合者，Amazon Redshift 會使用快取的結果，不會執行查詢。結果快取預設為開啟。若要關閉結果快取，請將 enable_result_cache_for_session 組態參數設為 off。	2017 年 11 月 16 日

變更	描述	變更日期
資料欄中繼資料函數	PG_GET_COLS 和 PG_GET_LATE_BINDIN_G_VIEW_COLS 會傳回 Amazon Redshift 資料表、檢視、晚期繫結檢視的欄中繼資料。	2017 年 11 月 16 日
WLM 的佇列跳轉 CTAS	Amazon Redshift 工作負載管理 (WLM) 現在支援 CREATE TABLE AS (CTAS) 陳述式的查詢佇列跳轉，以及唯讀查詢，例如 SELECT 陳述式。如需詳細資訊，請參閱 WLM 查詢佇列跳轉 。	2017 年 10 月 19 日
Amazon Redshift Spectrum 清單檔案	當您建立 Redshift Spectrum 外部資料表時，可以指定一個其中已列出 Amazon S3 上資料檔案位置的資訊清單檔案。如需詳細資訊，請參閱 CREATE EXTERNAL TABLE 。	2017 年 10 月 19 日
Amazon Redshift Spectrum 新 AWS 區域	Redshift Spectrum 現已在歐洲 (愛爾蘭) 和亞太區域 (東京) 區域提供。如需支援 AWS 區域的清單，請參閱 Amazon Redshift Spectrum 限制 。	2017 年 10 月 19 日
Amazon Redshift Spectrum 已新增檔案格式	您現在可以根據 Regex、Open CSV 和 Avro 資料檔案格式建立 Redshift Spectrum 外部資料表。如需詳細資訊，請參閱 CREATE EXTERNAL TABLE 。	2017 年 10 月 5 日
Amazon Redshift Spectrum 外部資料表的虛擬資料欄	您可以在 Redshift Spectrum 外部資料表中選取 \$path 和 \$size 虛擬欄，以檢視 Amazon S3 中的參考資料檔案的位置和大小。如需詳細資訊，請參閱 虛擬資料欄 。	2017 年 10 月 5 日
要驗證的函數 JSON	您可以使用 IS_VALID_JSON 和 IS_VALID_JSON_ARRAY 函數來檢查有效 JSON 格式。其他 JSON 函數現在具有選用的 null_if_invalid 引數。	2017 年 10 月 5 日
LISTAGG DISTINCT	您可以使用 DISTINCT 子句搭配 LISTAGG 彙總函數和 LISTAGG 視窗函數，以在串連之前從指定的表達式中刪除重複值。	2017 年 10 月 5 日

變更	描述	變更日期
以大寫名稱檢視資料欄	若要檢視大寫SELECT結果中的資料欄名稱，您可以將 describe_field_name_in_uppercase 組態參數設定為true。	2017年10月5日
略過外部資料表的標題行	您可以在 <code>skip.header.line.count</code> 命令中設定 CREATE EXTERNAL TABLE 屬性，來略過 Redshift Spectrum 資料檔案一開始的標題行。	2017年10月5日
掃描的資料列數	WLM 查詢監控規則使用 <code>scan_row_count</code> 指標來傳回掃描步驟中的資料列數。資料列計數，指的是在篩選標記要刪除的資料列 (幽靈資料列) 之前，且在套用使用者定義的查詢篩選條件之前所發出的資料列總數。如需詳細資訊，請參閱 已佈建 Amazon Redshift 的查詢監控指標 。	2017年9月21日
SQL 使用者定義的函數	純量SQL使用者定義函數 (UDF) 包含子SQLSELECT句，會在呼叫函數時執行，並傳回單一值。如需詳細資訊，請參閱 純量 SQL UDFs 。	2017年8月31日
近期繫結檢視	近期繫結的檢視不會與底層資料庫物件繫結，像是資料表、使用者定義的函數。因此，檢視和其參考的物件之間並無相依性。即使參考的物件不存在，您仍可以建立檢視。由於沒有相依性，您可以捨棄或修改參考物件，而不會影響檢視。Amazon Redshift 在查詢檢視之前不會檢查相依性。若要建立繫結延遲檢視，請使用您的 <code>CREATE VIEW</code> 陳述式指定 <code>WITH NO SCHEMABINDING</code> 子句。如需詳細資訊，請參閱 CREATE VIEW 。	2017年8月31日
OCTET_LENGTH 函數	OCTET_LENGTH 傳回指定字串的長度 (以位元組為單位)。	2017年8月18日
ORC 支援 和 Grok 檔案類型	Amazon Redshift Spectrum 現在支援 Redshift Spectrum 資料檔案的 ORC和 Grok 資料格式。如需詳細資訊，請參閱 Amazon Redshift Spectrum 中查詢的資料檔案 。	2017年8月18日

變更	描述	變更日期
RegexSerDe 現在支援	Amazon Redshift Spectrum 現在支援 RegexSerDe 資料格式。如需詳細資訊，請參閱 Amazon Redshift Spectrum 中查詢的資料檔案 。	2017 年 7 月 19 日
新增至 SVV_TABLES 和 SVV_ 的新資料欄 COLUMNS	domain_name 新增 remarks 和 SVV_COLUMNS 資料欄。 SVV_TABLES 新增 remarks 資料欄。	2017 年 7 月 19 日
SVV_TABLES 和 SVV_COLUMNS 系統檢視	SVV_TABLES 和 SVV_COLUMNS 系統檢視提供本機、外部資料表與檢視資料欄相關的資訊，以及其他詳細資訊。	2017 年 7 月 7 日
VPC Amazon Redshift Spectrum 與 Amazon EMR Hive 中繼存放區不再需要	Redshift Spectrum 刪除了使用 Amazon EMR Hive 中繼存放區時，Amazon Redshift 叢集和 Amazon EMR 叢集必須位於相同 VPC 子網路的要求。如需詳細資訊，請參閱 在 Amazon Redshift Spectrum 中使用外部目錄 。	2017 年 7 月 7 日
UNLOAD 至較小的檔案大小	根據預設，在 Amazon S3 上 UNLOAD 建立多個檔案，最大大小為 6.2 GB。若要建立較小的檔案，MAXFILESIZE 請使用 UNLOAD 命令指定。您可以指定 5 MB 至 6.2 GB 的檔案大小上限。如需詳細資訊，請參閱 UNLOAD 。	2017 年 7 月 7 日
TABLE PROPERTIES	您現在可以設定 CREATE EXTERNAL TABLE 或的 TABLEPROPERTIES numRows 參數 ALTER TABLE ，以更新資料表統計資料，以反映資料表中的資料列數目。	2017 年 6 月 6 日
ANALYZE PREDICATE COLUMNS	為了節省時間和叢集資源，您可以選擇僅分析可能用來做為述詞的資料列。當您 ANALYZE 使用 PREDICATE COLUMNS 子句執行時，分析操作只會包含聯結、篩選條件或依子句分組中使用的資料欄，或用作排序鍵或分佈鍵。如需詳細資訊，請參閱 分析資料表 。	2017 年 5 月 25 日

變更	描述	變更日期
IAM Amazon Redshift Spectrum 的政策	若要授予只使用 Redshift Spectrum 的 Amazon S3 儲存體存取權，您可以加入允許使用者代理程式 AWS Redshift/Spectrum 存取的條件。如需詳細資訊，請參閱 IAM Amazon Redshift Spectrum 的政策 。	2017 年 5 月 25 日
Amazon Redshift Spectrum 遞迴掃描	Redshift Spectrum 現在會掃描 Amazon S3 中指定資料夾以及子資料夾裡的檔案。如需詳細資訊，請參閱 Redshift Spectrum 的外部資料表 。	2017 年 5 月 25 日
查詢監控規則	使用 WLM 查詢監控規則，您可以為 WLM 佇列定義指標型效能界限，並指定當查詢超出這些界限時要採取的動作：日誌、跳轉或中止。您可以在工作負載管理 (WLM) 組態中定義查詢監控規則。如需詳細資訊，請參閱 WLM 查詢監控規則 。	2017 年 4 月 21 日
Amazon Redshift Spectrum	使用 Redshift Spectrum，可以有效率地查詢 Amazon S3 裡的檔案並擷取資料，無需將資料載入資料表。Redshift Spectrum 對於大型資料庫的查詢執行十分快速，因為 Redshift Spectrum 會直接掃描 Amazon S3 裡的檔案。大部分的處理發生在 Amazon Redshift Spectrum 層，大部分資料則保留在 Amazon S3。可以多個叢集同時查詢 Amazon S3 上的相同資料集，無需為每個叢集複製資料副本。如需詳細資訊，請參閱 Amazon Redshift Spectrum	2017 年 4 月 19 日

變更	描述	變更日期
新系統資料表支援 Redshift Spectrum	<p>新增下列新的系統檢視來支援 Redshift Spectrum :</p> <ul style="list-style-type: none"> • SVL_S3QUERY • SVL_S3QUERY_SUMMARY • SVV_EXTERNAL_COLUMNS • SVV_EXTERNAL_DATABASES • SVV_EXTERNAL_PARTITIONS • SVV_EXTERNAL_TABLES • PG_EXTERNAL_SCHEMA 	2017 年 4 月 19 日
APPROXIMATE PERCENTILE_DISC 彙總函數	現已可使用 APPROXIMATE PERCENTILE_DISC 彙總函數。	2017 年 4 月 4 日
伺服器端加密搭配 KMS	您現在可以使用具有 AWS Key Management Service 金鑰 (SSE-) 的伺服器端加密，將資料卸載至 Amazon S3KMS。此外， COPY 現在會從 Amazon S3 透明載入 KMS加密的資料檔案。如需詳細資訊，請參閱 UNLOAD 。	2017 年 2 月 9 日
新的授權語法	您現在可以使用 IAM_ROLE、MASTER_SYMMETRIC_KEY、ACCESSKEY_IDKEY、SECRET_ACCESS_ 和 SESSION_TOKEN 參數來提供 COPY、UNLOAD和 CREATELIBRARY命令的授權和存取資訊。新的授權語法為 CREDENTIALS 參數提供了一個更靈活的替代選項，而不是提供單一字串引數。如需詳細資訊，請參閱 授權參數 。	2017 年 2 月 9 日
提升結構描述限制	現在每個叢集最多可以建立 9,900 個結構描述。如需詳細資訊，請參閱 CREATE SCHEMA 。	2017 年 2 月 9 日

變更	描述	變更日期
預設資料表編碼	CREATE TABLE 然後， ALTER TABLE 將LZO壓縮編碼指派給大多數新的資料欄。定義為排序索引鍵的欄、定義為 BOOLEAN、REAL 或 DOUBLE PRECISION 資料類型的欄，以及臨時資料表，預設會指派RAW編碼。如需詳細資訊，請參閱 ENCODE 。	2017 年 2 月 6 日
ZSTD 壓縮編碼	Amazon Redshift 現在支援ZSTD資料欄壓縮編碼。	2017 年 1 月 19 日
PERCENTILE_CONT 和 MEDIAN 彙總函數	PERCENTILE_CONT 和 MEDIAN 現已是彙總函數也是視窗函數。	2017 年 1 月 19 日
使用者定義函數 (UDF) 使用者記錄	您可以使用 Python 記錄模組，在 中建立使用者定義的錯誤和警告訊息UDFs。您可以在查詢執行之後，查詢 SVL_UDF_LOG 系統檢視來擷取日誌記錄的訊息。如需使用者定義的資訊相關資訊，請參閱 在 Python 中記錄錯誤和警告 UDFs	2016 年 12 月 8 日
ANALYZE COMPRESSION 預估縮減	ANALYZE COMPRESSION 命令現在會報告每欄磁碟空間百分比減少的估計值。如需詳細資訊，請參閱 ANALYZE COMPRESSION 。	2016 年 11 月 10 日
連線限制	您現在可以設定允許使用者同時打開的資料庫連線數目限制。您也可以限制資料庫的並行連線數量。如需詳細資訊，請參閱 CREATE USER 和 CREATE DATABASE 。	2016 年 11 月 10 日
COPY 排序順序增強	COPY 當您以排序索引鍵順序載入資料時，現在會自動將新資料列新增至資料表的排序區域。如需啟用此強化的特定需求，請參閱 以排序索引鍵順序載入您的資料	2016 年 11 月 10 日
CTAS 具有壓縮	CREATE TABLE AS (CTAS) 現在會根據資料欄的資料類型，自動將壓縮編碼指派給新資料表。如需詳細資訊，請參閱 欄位和資料表屬性的繼承 。	2016 年 10 月 28 日

變更	描述	變更日期
時間戳記包含時區資料類型	對於時區 (TIMESTAMPTZ) 資料類型，Amazon Redshift 現在支援時間戳記。同時，也已新增多個新函數來支援新的資料類型。如需詳細資訊，請參閱 日期和時間函數 。	2016 年 9 月 29 日
分析閾值	為了縮短處理時間並改善 ANALYZE 操作的整體系統效能，如果自上次ANALYZE命令執行以來變更的資料列百分比低於 analyze_threshold_percent 參數指定的分析閾值，Amazon Redshift 會略過分析資料表。 <code>analyze_threshold_percent</code> 預設為 10。	2016 年 8 月 9 日
新的 STL_RESTARTED_SESSIONS 系統資料表	當 Amazon Redshift 重新啟動工作階段時，會 STL_RESTARTED_SESSIONS 記錄新的程序 ID (PID) 和舊的 PID。	2016 年 8 月 9 日
已更新日期和時間函數的文件	新增函數的摘要資訊和 日期和時間函數 的連結，並已更新函數的參考資訊使資訊一致。	2016 年 6 月 24 日
STL_CONNECTION_ 中的新資料欄LOG	STL_CONNECTION_LOG 系統資料表有兩個新的資料欄可追蹤SSL連線。如果您定期將稽核日誌載入 Amazon Redshift 資料表，則需要將以下新欄位新增到目標資料表： <code>sslcompression</code> 和 <code>sslexpansion</code> 。	2016 年 5 月 5 日
MD5-hash 密碼	您可以透過提供密碼和使用者名稱的 MD5-hash 字串來指定 CREATE USER 或 ALTER USER 命令的密碼。	2016 年 4 月 21 日
STV_TBL_ 中的新資料欄PERM	backup 系統檢視中的 STV_TBL_PERM 資料欄指出叢集快照中是否包含資料表。如需詳細資訊，請參閱 BACKUP 。	2016 年 4 月 21 日
不備份資料表	對於不包含關鍵資料的暫存資料表等資料表，您可以在 CREATE TABLE 或 CREATE TABLE AS 陳述式中指定 <code>BACKUP NO</code> ，以防止 Amazon Redshift 在自動或手動快照中包含資料表。使用不備份資料表可以在建立快照以及從快還原照時節省處理時間，並節省 Amazon S3 上的儲存空間。	2016 年 4 月 7 日

變更	描述	變更日期
VACUUM 刪除閾值	根據預設， VACUUM 命令現在會回收空間，使至少百分之 95 的剩餘資料列不會被標記為刪除。因此，與回收 100% 的已刪除資料列相比，刪除階段VACUUM通常需要的時間更少。您可以在執行VACUUM命令時包含 TO 閾值PERCENT參數，以變更單一資料表的預設閾值。	2016 年 4 月 7 日
SVV_TRANS ACTIONS 系統資料表	SVV_TRANSACTIONS 系統檢視會記錄資料庫中目前保持鎖定之資料表的交易相關資訊。	2016 年 4 月 7 日
使用IAM角色存取其他 AWS 資源	若要在叢集與其他 AWS 資源之間移動資料，例如 Amazon S3、DynamoDBAmazon EMR或 Amazon EC2，您的叢集必須具有存取資源和執行必要動作的許可。作為使用 COPY、UNLOAD或 CREATELIBRARY 命令提供存取金鑰對的更安全替代方案，您現在可以指定叢集用於身分驗證和授權IAM的角色。如需詳細資訊，請參閱 角色類型存取控制 。	2016 年 3 月 29 日
VACUUM 排序閾值	VACUUM 命令現在會略過任何資料表的排序階段，其中已對資料表資料列的 95% 以上進行排序。您可以在執行 VACUUM 命令時包含 TO 閾值PERCENT參數，以變更單一資料表的預設排序閾值。	2016 年 3 月 17 日
STL_CONNECTION_ 中的新資料欄LOG	STL_CONNECTION_LOG 系統資料表有三個新資料欄。如果您定期將稽核日誌載入 Amazon Redshift 資料表，則需要將以下新欄位新增到目標資料表：sslversion、sslcipher 和 mtu。	2016 年 3 月 17 日
UNLOAD 使用 bzip2 壓縮	您現在可以選擇使用 bzip2 壓縮進行 UNLOAD 。	2016 年 2 月 8 日

變更	描述	變更日期
ALTER TABLE APPEND	ALTER TABLE APPEND 會從現有來源資料表移出資料，將資料列附加到目標資料表。ALTER TABLE APPEND 通常比類似 CREATE TABLE AS 或 INSERT INTO 操作快得多，因為資料是移動的，而不是重複的。	2016 年 2 月 8 日
WLM 查詢佇列跳轉	如果工作負載管理 (WLM) 因為WLM逾時而取消唯讀查詢，例如SELECT陳述式，會WLM嘗試將查詢路由至下一個相符佇列。如需詳細資訊，請參閱 WLM 查詢佇列跳轉	2016 年 1 月 7 日
ALTER DEFAULT PRIVILEGES	您可以使用 ALTER DEFAULT PRIVILEGES 命令來定義日後由指定使用者建立的物件將套用的存取權預設設定。	2015 年 12 月 10 日
bzip2 檔案壓縮	COPY 命令支援從使用 bzip2 壓縮的檔案載入資料。	2015 年 12 月 10 日
NULLS FIRST 和 NULLS LAST	您可以指定 ORDER BY 子句是否應該NULLS在結果集中先排序還是最後排序。如需詳細資訊，請參閱 ORDER BY 子句 和 範圍函數語法摘要 。	2015 年 11 月 19 日
REGION 關鍵字 CREATE LIBRARY	如果包含UDF程式庫檔案的 Amazon S3 儲存貯體不位於與 Amazon Redshift 叢集相同的 AWS 區域，您可以使用 REGION選項來指定資料所在的區域。如需詳細資訊，請參閱 CREATE LIBRARY 。	2015 年 11 月 19 日
使用者定義的純量函數 (UDFs)	您現在可以建立自訂使用者定義的純量函數，以實作 Python 2.7 標準程式庫中 Amazon Redshift 支援的模組提供的非SQL處理功能，或UDFs您自己的基於 Python 程式設計語言的自訂功能。如需詳細資訊，請參閱 Amazon Redshift 中的使用者定義函數 。	2015 年 9 月 11 日

變更	描述	變更日期
WLM 組態中的動態屬性	WLM 組態參數現在支援動態套用某些屬性。其他屬性仍然是靜態變更，需要將關聯的叢集重新啟動才會套用組態變更。如需詳細資訊，請參閱 WLM 動態和靜態組態屬性 和 工作負載管理 。	2015 年 8 月 3 日
LISTAGG 函數	LISTAGG 函數 和 LISTAGG 視窗函數 會傳回字串，這個字串是將一組資料欄的值串連起來建立而成。	2015 年 7 月 30 日
已棄用的參數	已棄用 max_cursor_result_set_size 組態參數。游標結果集的大小限制是根據叢集的節點類型而定。如需詳細資訊，請參閱 游標限制條件 。	2015 年 7 月 24 日
修訂的COPY命令文件	大幅修訂 COPY 命令的參考資料，使資料易懂更容易取得。	2015 年 7 月 15 日
COPY 從 Avro 格式	此 COPY 命令支援使用從 Amazon S3、Amazon 和遠端主機上的資料檔案EMR，以 Avro 格式載入資料SSH。如需詳細資訊，請參閱 AVRO 和 從 Avro 複製的範例 。	2015 年 7 月 8 日
STV_STARTUP_RECOVERY_STATE	STV_STARTUP_RECOVERY_STATE 系統資料表會記錄在叢集重新啟動操作期間暫時鎖定的資料表狀態。Amazon Redshift 會在處理資料表時暫時鎖定資料表，以解決叢集重新啟動後的過時交易。	2015 年 5 月 25 日
ORDER 排序函數的 BY 選用	對於某些視窗排名函數，ORDERBY 子句現在是選用項目。如需詳細資訊，請參閱 CUME_DIST 視窗函數 、 DENSE_RANK 視窗函數 、 RANK 視窗函數 、 NTILE 視窗函數 、 PERCENT_RANK 視窗函數 、 ROW_NUMBER 視窗函數 。	2015 年 5 月 25 日

變更	描述	變更日期
交錯的排序索引鍵	交錯的排序索引鍵使排序索引鍵中的每個資料欄具有相等權重。若使用交錯的排序索引鍵來代替預設的複合索引鍵，當查詢對第二個排序資料欄使用限制性述詞，尤其是查詢大型資料表時，可以大幅改善查詢效能。在對相同資料表中不同資料欄套用多個篩選條件時，交錯排序也能改善效能。如需詳細資訊，請參閱 排序金鑰 和 CREATE TABLE 。	2015 年 5 月 11 日
已修訂的調校查詢效能主題	已擴充 查詢效能調校 的內容，增加用於分析查詢效能的新查詢以及更多範例。此外，也將主題修訂得更清楚更完整。 設計查詢的 Amazon Redshift 最佳實務 提供如何撰寫查詢以改善效能的詳細資訊。	2015 年 3 月 23 日
SVL_QUERY_QUEUE_INFO	SVL_QUERY_QUEUE_INFO 檢視會摘要查詢佇列或遞交佇列中花費時間的WLM查詢詳細資訊。	2015 年 2 月 19 日
SVV_TABLE_INFO	您可以使用 SVV_TABLE_INFO 檢視來診斷和解決可能影響查詢效能的資料表設計問題，包括壓縮編碼、分佈索引鍵、排序樣式、資料配送偏度、資料表大小、統計等方面的問題。	2015 年 2 月 19 日
UNLOAD 使用伺服器端檔案加密	UNLOAD 命令現在會自動使用 Amazon S3 伺服器端加密 (SSE) 來加密所有卸載資料檔案。伺服器端加密增加了另一層資料安全，且對效能幾乎沒有影響。	2014 年 10 月 31 日
CUME_DIST 視窗函數	CUME_DIST 視窗函數 會計算視窗或分割區內值的累積分佈。	2014 年 10 月 31 日
MONTHS_BETWEEN 函數	MONTHS_BETWEEN 函數 可判斷兩個日期之間有幾個月。	2014 年 10 月 31 日
NEXT_DAY 函數	NEXT_DAY 函數 會傳回在給定日期之後的指定那天 (星期幾) 後的第一個執行個體的日期。	2014 年 10 月 31 日
PERCENT_RANK 視窗函數	PERCENT_RANK 視窗函數 會計算給定資料列的百分比排行。	2014 年 10 月 31 日

變更	描述	變更日期
RATIO_TO_REPORT 視窗函數	RATIO_TO_REPORT 視窗函數 會計算視窗或分割區內值與值總和的比率。	2014 年 10 月 31 日
TRANSLATE 函數	TRANSLATE 函數 會用指定的取代值來取代給定表達式中出現的所有指定字元。	2014 年 10 月 31 日
NVL2 函數	根據指定的表達式評估為 NULL 或 NOT ， 會 NVL2 函數 傳回兩個值之一 NULL。	2014 年 10 月 16 日
MEDIAN 視窗函數	MEDIAN 視窗函數 會計算視窗或分割區內值範圍的中位數。	2014 年 10 月 16 日
GRANT 和 REVOKE 命令的 ON ALL TABLES IN SCHEMA schema_name 子句	GRANT 和 REVOKE 命令已更新為 ON ALL TABLES IN SCHEMA schema_name 子句。這個子句可讓您使用單一命令來變更結構描述中所有資料表的權限。	2014 年 10 月 16 日
適用於 DROP SCHEMA、TABLE、DROP USER 和 DROPVIEW 命令的 IF DROP EXISTS 子句	DROP SCHEMA 、 DROP USER 、 DROP TABLE 和 DROP VIEW 命令已更新為 IF EXISTS 子句。如果指定的物件不存在，這個子句可使命令不進行任何變更並傳回訊息，而不是以發生錯誤終止。	2014 年 10 月 16 日
適用於 CREATESCHEMA 和 CREATETABLE 命令的 IF NOTEXISTS 子句	CREATE SCHEMA 和 CREATE TABLE 命令已更新為 IF NOTEXISTS 子句。如果指定的物件已存在，這個子句可使命令不進行任何變更並傳回訊息，而不是以發生錯誤終止。	2014 年 10 月 16 日

變更	描述	變更日期
COPY 支援 UTF-16 編碼	COPY 命令現在支援從使用 UTF-16 編碼和 UTF-8 編碼的資料檔案載入。如需詳細資訊，請參閱 ENCODING 。	2014 年 9 月 29 日
新的工作負載管理教學	教學課程：設定手動工作負載管理 (WLM) 佇列 引導您完成設定工作負載管理 (WLM) 佇列的程序，以改善查詢處理和資源配置。	2014 年 9 月 25 日
AES 128 位元加密	COPY 命令現在支援 128 位元加密，以及使用 AES Amazon S3 AES 用戶端加密從資料檔案載入時支援 256 位元加密。如需詳細資訊，請參閱 從 Amazon S3 載入加密的資料檔案 。	2014 年 9 月 29 日
PG_LAST_UNLOAD_COUNT 函數	PG_LAST_UNLOAD_COUNT 函數會傳回最近 UNLOAD 操作中處理的列數。如需詳細資訊，請參閱 PG_LAST_UNLOAD_COUNT 。	2014 年 9 月 15 日
新的故障診斷查詢章節	查詢疑難排解 提供快速的參考，方便識別和解決您使用 Amazon Redshift 查詢時可能遇到的一些最常見和最嚴重的問題。	2014 年 7 月 7 日
新的載入資料教學	教學課程：從 Amazon S3 載入資料 將從頭到尾引導您完成從 Amazon S3 儲存貯體中的資料檔案將資料載入 Amazon Redshift 資料庫資料表的過程。	2014 年 7 月 1 日
PERCENTILE_CONT 視窗函數	PERCENTILE_CONT 視窗函數 是反向分發函數，假定連續的分發模型。它採用百分位數值和排序規格，且會傳回插入值，該值將根據排序規格落入給定的百分位數值。	2014 年 6 月 30 日
PERCENTILE_DISC 視窗函數	PERCENTILE_DISC 視窗函數 是反向分發函數，假定不連續的分發模型。它採用百分位數值和排序規格，且會傳回集裡的一個元素。	2014 年 6 月 30 日
GREATEST 和 LEAST 函數	GREATEST 和 LEAST 函數 函數會傳回表達式清單中的最大或最小值。	2014 年 6 月 30 日

變更	描述	變更日期
跨區域 COPY	COPY 命令支援從 Amazon Redshift 叢集以外區域中的 Amazon S3 儲存貯體或 Amazon DynamoDB 資料表載入資料。如需詳細資訊，請參閱 COPY 命令參考 REGION 中的。	2014 年 6 月 30 日
擴充最佳實務	Amazon Redshift 最佳實務 的內容已擴充、重新組織並移至導覽階層的頂端，使其更容易被找到。	2014 年 5 月 28 日
UNLOAD 至單一檔案	UNLOAD 命令可以透過新增 PARALLELOFF 選項，選擇性地將資料表資料序列卸載至 Amazon S3 上的單一檔案。如果資料的大小大於 6.2 GB 的最大檔案大小，會 UNLOAD 建立其他檔案。	2014 年 5 月 6 日
REGEXP 函數	REGEXP_COUNT 、 REGEXP_INSTR 、 REGEXP_REPLACE 函數會根據正規表達式模式比對來操作字串。	2014 年 5 月 6 日
COPY 從 Amazon EMR	COPY 命令支援直接從 Amazon EMR 叢集載入資料。如需詳細資訊，請參閱 從 Amazon 載入資料 EMR 。	2014 年 4 月 18 日
WLM 並行限制增加	您現在可以將工作負載管理 (WLM) 設定為在使用者定義的查詢佇列中同時執行最多 50 個查詢。此增加為使用者提供更高的靈活性，以透過修改 WLM 組態來管理系統效能。如需詳細資訊，請參閱 實作手冊 WLM	2014 年 4 月 18 日
新的組態參數用於管理游標大小	<code>max_cursor_result_set_size</code> 組態參數定義了針對大型查詢的每個游標結果集可以傳回的數據大小上限 (以 MB 為單位)。此參數值也會影響叢集的並行游標數，可讓您配置一個值來增加或減少叢集的游標數。 如需詳細資訊，請參閱本指南中的 DECLARE 和《Amazon Redshift 管理指南》中的 設定游標結果集的大小上限 。	2014 年 3 月 28 日
COPY 從 JSON 格式	此 COPY 命令支援使用 從 Amazon S3 上的資料檔案和遠端主機以 JSON 格式載入資料 SSH。如需詳細資訊，請參閱 COPY 從 JSON 格式 使用須知。	2014 年 3 月 25 日

變更	描述	變更日期
新系統資料表 STL_PLAN_INFO	STL_PLAN_INFO 資料表補充 EXPLAIN 命令，作為查看查詢計畫的另一種方式。	2014 年 3 月 25 日
新函數 REGEXP_SUBSTR	REGEXP_SUBSTR 函數 會搜索正規表達式模式，傳回從字串中擷取的字元。	2014 年 3 月 25 日
STL_COMMIT_ 的新資料欄STATS	STL_COMMIT_STATS 資料表有兩個新資料欄：numxids 和 oldestxid 。	2014 年 3 月 6 日
COPY 從 gzip 和 lzop 的SSH支援	透過SSH連線載入資料時， COPY 命令支援 gzip 和 lzop 壓縮。	2014 年 2 月 13 日
新函數	ROW_NUMBER 視窗函數 會傳回目前資料列的編號。 STRTOL 函數 將指定底數之數字的字串表達式轉換為等同的整數值。 PG_CANCEL_BACKEND 和 PG_TERMINATE_BACKEND 可讓使用者取消查詢和工作階段連線。新增 LAST_DAY 函數以便和 Oracle 相容。	2014 年 2 月 13 日
新的系統資料表	STL_COMMIT_STATS 系統資料表提供與遞交效能相關的指標，包括各種遞交階段的時間，以及遞交的區塊數。	2014 年 2 月 13 日
FETCH 使用單節點叢集	在單一節點的叢集上使用游標時，使用 FETCH 命令可以擷取的資料列數上限為 1000。FETCH FORWARD ALL 不支援單節點叢集。	2014 年 2 月 13 日
DS_DIST_ALL_INNER 重新分發策略	解釋計劃輸出中的 DSDIST__ALL_INNER 表示整個內部資料表已重新分佈至單一切片，因為外部資料表使用 DISTSTYLE ALL。如需詳細資訊，請參閱 聯結類型範例 和 評估查詢計畫 。	2014 年 1 月 13 日
用於查詢的新系統資料表	Amazon Redshift 新增新的系統資料表，客戶可用來評估查詢執行，以便進行調校和故障診斷。如需詳細資訊，請參閱 SVL_COMPILE , STL_SCAN , STL_RETURN , STL_SAVE STL_ALERT_EVENT_LOG .	2014 年 1 月 13 日

變更	描述	變更日期
單一節點的游標	單一節點的叢集現在支援游標。單一節點的叢集可同時有兩個游標，其結果集大小上限為 32 GB。在單一節點叢集上，我們建議將 ODBC Cache Size 參數設定為 1,000。如需詳細資訊，請參閱 DECLARE 。	2013 年 13 月 12 日
ALL 分佈樣式	ALL 分佈可能會大幅縮短特定類型查詢的執行時間。當資料表使用ALL分佈樣式時，資料表的副本會分佈到每個節點。由於資料表與其他每個資料表有效地共置，因此在查詢執行期間不需要重新配送。ALL 分佈不適用於所有資料表，因為它會增加儲存需求和載入時間。如需詳細資訊，請參閱 查詢最佳化的資料分佈 。	2013 年 11 月 11 日
COPY 從遠端主機	除了從 Amazon S3 上的資料檔案以及從 Amazon DynamoDB 資料表載入資料表之外，COPY命令還可以使用SSH連線從 Amazon EMR叢集、Amazon EC2執行個體和其他遠端主機載入文字資料。Amazon Redshift 使用多個同時SSH連線來平行讀取和載入資料。如需詳細資訊，請參閱 從遠端主機載入資料 。	2013 年 11 月 11 日
WLM 使用的記憶體百分比	您可以為工作負載管理 (WLM) 組態中的每個佇列指定特定記憶體百分比，以平衡工作負載。如需詳細資訊，請參閱 實作手冊 WLM 。	2013 年 11 月 11 日
APPROXIMATE COUNT(DISTINCT)	使用 APPROXIMATECOUNT (DISTINCT) 的查詢執行速度更快，相對錯誤約為 2%。APPROXIMATE COUNT (DISTINCT) 函數使用 HyperLogLog 演算法。如需詳細資訊，請參閱 COUNT 函數 。	2013 年 11 月 11 日
擷取最近查詢詳細資訊的新SQL函數	四個新SQL函數會擷取有關最近查詢和COPY命令的詳細資訊。新函數使查詢系統日誌資料表變得更加容易，且在許多情況下不必存取系統資料表即可提供必要的詳細資訊。如需詳細資訊，請參閱 PG_BACKEND_PID ， PG_LAST_COPY_ID ， PG_LAST_COPY_COUNT ， PG_LAST_QUERY_ID 。	2013 年 11 月 1 日

變更	描述	變更日期
MANIFEST 的選項 UNLOAD	UNLOAD 命令MANIFEST的選項 補充了COPY命令MANIFEST的選項。搭配 使用 MANIFEST選項，UNLOAD會自動建立清單檔案，明確列出卸載操作在 Amazon S3 上建立的資料檔案。然後，您可以將相同的資訊清單檔案與 COPY命令搭配使用，以載入資料。如需詳細資訊，請參閱 將資料卸載到 Amazon S3 和 UNLOAD例子 。	2013 年 11 月 1 日
MANIFEST 的選項 COPY	您可以使用 MANIFEST選項搭配 COPY 命令，明確列出將從 Amazon S3 載入的資料檔案。	2013 年 10 月 18 日
用於查詢的故障診斷的系統資料表	新增用於對查詢進行故障診斷的系統資料表的文件。 STL 記錄的檢視 本節現在包含下列系統資料表的文件：STL_AGGR、STL_BCAST、STL_DIST、STL_DELETE、STL_HASH、STL_HASHJOIN、STL_INSERT、STL_LIMIT、_、STL_、_MERGE、STL_MERGEJOIN、STL_NESTLOOP、STL_PARSE、STL_、_PROJECT、_、STL_、_SCAN、_、STL_SORT、STL_、_UNIQUE、_、STL_WINDOW。	2013 年 10 月 3 日
CONVERT_TIMEZONE 函數	CONVERT_TIMEZONE 函數 會將時間戳記從一個時區轉換為另一個時區，並可選擇自動調整日光節約時間。	2013 年 10 月 3 日
SPLIT_PART 函數	SPLIT_PART 函數 會在指定分隔符號之處分割字串，並傳回指定位置的部分。	2013 年 10 月 3 日
STL_USERLOG 系統資料表	當建立、更改或刪除資料庫使用者時， STL_USERLOG 會記錄發生變更的詳細資訊。	2013 年 10 月 3 日
LZO 資料欄編碼和 LZOP檔案壓縮。	LZO 資料欄壓縮編碼結合了非常高的壓縮比率和良好的效能。COPY 從 Amazon S3 支援從使用壓縮壓縮的檔案載入 LZOP 。	2013 年 9 月 19 日

變更	描述	變更日期
JSON、規則運算式和游標	新增了剖析JSON字串、使用規則運算式進行模式比對，以及使用游標透過ODBC連線擷取大型資料集的支援。如需詳細資訊，請參閱 JSON 函數 、 模式比對條件 及 DECLARE 。	2013 年 9 月 10 日
ACCEPTINVCHAR 的選項 COPY	您可以使用 COPY 命令指定 ACCEPTINVCHAR 選項，以成功載入包含無效 UTF-8 個字元的資料。	2013 年 8 月 29 日
CSV 的選項 COPY	COPY 命令現在支援從格式化CSV的輸入檔案載入。	2013 年 8 月 9 日
CRC32	CRC32 函數 會執行循環冗餘檢查。	2013 年 8 月 9 日
WLM 萬用字元	工作負載管理 (WLM) 支援萬用字元，可將使用者群組和查詢群組新增至佇列。如需詳細資訊，請參閱 萬用字元 。	2013 年 8 月 1 日
WLM 逾時	若要限制指定WLM佇列中查詢的允許使用時間，您可以設定每個佇列的WLM逾時值。如需詳細資訊，請參閱 WLM 逾時 。	2013 年 8 月 1 日
新COPY選項 'auto' 和 'epochseconds'	COPY 命令會執行日期和時間格式的自動辨識。新的時間格式 'epochsecs' 和 'epochmilliseconds' 可COPY啟用 epoch 格式載入資料。	2013 年 7 月 25 日
CONVERT_TIMEZONE 函數	CONVERT_TIMEZONE 函數 可將時間戳記從一個時區轉換為另一個時區。	2013 年 7 月 25 日
FUNC_SHA1 函數	會使用SHA1演算法 FUNC_SHA1 函數 轉換字串。	2013 年 7 月 15 日
max_execution_time	若要限制允許查詢使用的時間量，您可以將 max_execution_time 參數設定為WLM組態的一部分。如需詳細資訊，請參閱 修改WLM組態 。	2013 年 7 月 22 日

變更	描述	變更日期
四位元組 UTF-8 個字元	VARCHAR 資料類型現在支援 4 位元組 UTF-8 個字元。不支援 5 位元組或更長的 UTF-8 個字元。如需詳細資訊，請參閱 儲存與範圍 。	2013 年 7 月 18 日
SVL_QERROR	SVL_QERROR 系統檢視已棄用。	2013 年 7 月 12 日
修訂文件歷史記錄	Document History (文件歷史記錄) 頁面現在會顯示文件更新的日期。	2013 年 7 月 12 日
STL_UNLOAD_LOG	STL_UNLOAD_LOG 會記錄卸載操作的詳細資訊。	2013 年 7 月 5 日
JDBC 擷取大小參數	若要避免使用 擷取大型資料集時發生用戶端記憶體不足錯誤 JDBC，您可以透過設定擷取大小參數，讓用戶端以批次方式 JDBC 擷取資料。如需詳細資訊，請參閱 設定 JDBC 擷取大小參數 。	2013 年 6 月 27 日
UNLOAD 加密檔案	UNLOAD 現在支援將資料表的資料卸載到 Amazon S3 上已加密的檔案。	2013 年 5 月 22 日
暫時性憑證	COPY 和 UNLOAD 現在支援使用暫時性登入資料。	2013 年 11 月 4 日
補充說明	闡明並已擴充設計資料表和載入資料的討論。	2013 年 2 月 14 日
新增最佳實務	新增 設計資料表的 Amazon Redshift 最佳實務 和 載入資料的 Amazon Redshift 最佳實務 。	2013 年 2 月 14 日
闡明密碼限制	說明 CREATEUSER 和 ALTER 的密碼限制 USER，以及各種次要修訂。	2013 年 2 月 14 日
新指南	這是《Amazon Redshift 開發人員指南》的第一個版本。	2013 年 2 月 14 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。