MySQL 5.7 + JSON
created: 2015/10/26

日本オラクル株式会社
MySQL Global Business Unit

ORACLE®

Copyright © 2015 Oracle and/or its affiliates. All rights reserved. |

SAFE HARBOR STATEMENT

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメントするものではない為、購買決定を行う際の判断材料になさらないで下さい。

オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Program Agenda

1. Introduction
2. JSON datatype
3. Indexing of JSON data
4. Functions to handle JSON data
5. Misc Supporting Features

ORACLE®

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

MySQL.com | Downloads | Documentation | Developer Zone

MySQL Server | MySQL Enterprise Edition | Workbench | Utilities/Fabric | Cluster | Connectors

Documentation Library

Table of Contents

MySQL 5.7 Manual

MySQL 5.6 Manual

MySQL 5.5 Manual

MySQL 5.1 Manual

MySQL 5.0 Manual

Search manual:

Go

MySQL 5.7 Reference Manual :: 11 Data Types

Chapter 11 Data Types

Table of Contents [+/-]

11.1 Data Type Overview [+/-]

11.2 Numeric Types [+/-]

11.3 Date and Time Types [+/-]

11.4 String Types [+/-]

11.5 Extensions for Spatial Data [+/-]

11.6 The JSON Data Type

11.7 Data Type Default Values

11.8 Data Type Storage Requirements

11.9 Choosing the Right Type for a Column

11.10 Using Data Types from Other Database Engines

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Why JSON support in MySQL?

JSON（JavaScript Object Notation）ドキュメントデータへの
効率的なアクセスを可能にするネイティブJSONデータ･タイプ

- シリアルフォーマットの便利なオブジェクト
- 効果的にJSONデータを処理する為
- JavaScriptアプリケーションのネイティブサポート
- リレーショナルデータとスキーマレスデータのシームレスな統合
- 既存のデータベース・インフラストラクチャを新しいアプリケーションへの活用

ORACLE®

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Program Agenda

1. Introduction
2. JSON datatype
3. Indexing of JSON data
4. Functions to handle JSON data
5. Misc Supporting Features

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

ストレージオプション

- Text
  - 高速なinsert処理
  - ヒューマンリーダブル
  - 検証が必要
  - 構文解析が必要
  - Updateが困難
- Binary (JSON型)
  - 1度のみ検証
  - 高速な参照
  - インプレースUpdate
  - Insert処理が遅い
  - そのままでは解読不可

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

NEW JSONデータ型

- utf8mb4 文字セット
- 参照中心のワークロードに最適化
- INSERT時のみのパースと構文検証
- 効率の良い、バイナリーフォーマット
- ディクショナリー
  - ソートされたオブジェクトキー
  - インデックスによる高速データアクセス

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

NEW JSONデータ型: サポート

- JSONで表現する全てのデータ型をサポート
  - 数値, 文字列, bool(true,false)
  - オブジェクト, 配列
- 拡張
  - 日付(date), 時刻, 日付(datetime), タイムスタンプ
  - その他

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSONデータ型: Create and Insert

JSONデータ型を指定して、テーブル内に列を作成

```
CREATE TABLE employees (data JSON);
INSERT INTO employees VALUES ('{"id": 1, "name": "Jane"}');
INSERT INTO employees VALUES ('{"id": 2, "name": "Joe"}');

SELECT * FROM employees;
+----------------------------+
| data                       |
+----------------------------+
| {"id": 1, "name": "Jane"} |
| {"id": 2, "name": "Joe"}  |
+----------------------------+
2 rows in set (0,00 sec)
```

メモ:JSON配列は内カンマで区切られ、囲まれた値のリスト[と]の文字が含まれています。JSONオブジェクトは、{と}文字以内カンマで区切られ、囲まれたキー/値のペアのセットが含まれています。

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

utf8mb4 文字セット

JSONに変換した文字列はグローバル対応の一環として, utf8mb4の文字セットとutf8mb4_binの照合がDefaultです。utf8mb4_binがバイナリ照合であるため、JSON値の比較では、大文字と小文字が区別されます。

```
SET @j = JSON_OBJECT('key', 'value');

 [NEW57]> SELECT @j;
+-------------------+
| @j                |
+-------------------+
| {"key": "value"} |
+-------------------+
1 row in set (0.00 sec)

 [NEW57]> SELECT CHARSET(@j), COLLATION(@j);
+-------------+---------------+
| CHARSET(@j) | COLLATION(@j) |
+-------------+---------------+
| utf8mb4     | utf8mb4_bin   |
+-------------+---------------+
1 row in set (0.00 sec)
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSON Comparator

- 多相性挙動
- シームレスかつ一貫性のある比較
  - JSON vs JSON, JSON vs SQL
  - 異なるデータは常に等しくない
  - 自動的な型変換は行いません
- 堅牢性
- キャッシングの広範な使用

```
SELECT CAST(1 AS JSON) = 1;
+-----------------------+
| CAST(1 AS JSON) = 1 |
+-----------------------+
|                     1 |
+-----------------------+
1 row in set (0.00 sec)
```

JSON 1の値は1と等しい

```
SELECT CAST(1 AS JSON) = '1';
+-------------------------+
| CAST(1 AS JSON) = '1' |
+-------------------------+
|                       0 |
+-------------------------+
1 row in set (0.00 sec)
```

JSON 1の値は'1'と不等

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

TEXT/VARCHARと比較した優位性

1. JSONドキュメントデータ形式の検証

```
INSERT INTO employees VALUES ('some random text');

ERROR 3130 (22032): Invalid JSON text:
"Expect a value here." at position 0 in value
(or column) 'some random text'.
```

2. 効率の良い、バイナリーフォーマット
   オブジェクトメンバと配列要素への参照性能が向上

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Real Lifeデータを用いた検証

位置情報を含む、サンフランシスコの区, 市, 郡を
表現した地域データを用いた検証

- Via SF OpenData
- 206,000件のJSONデータ

[検証用テーブル]

```
CREATE TABLE features (
id INT NOT NULL auto_increment primary key,
feature JSON NOT NULL );
```

[検証用データ]

https://github.com/zemirco/sf-city-lots-json + small tweaks

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

```
{
   "type":"Feature",
   "geometry":{
      "type":"Polygon",
      "coordinates":[
         [
            [-122.42200352825247,37.80848009696725,0],
            [-122.42207601332528,37.808835019815085,0],
            [-122.42110217434865,37.808803534992904,0],
            [-122.42106256906727,37.80860105681814,0],
            [-122.42200352825247,37.80848009696725,0]
         ]
      ]
   },
   "properties":{
      "TO_ST":"0",
      "BLKLOT":"0001001",
      "STREET":"UNKNOWN",
      "FROM_ST":"0",
      "LOT_NUM":"001",
      "ST_TYPE":null,
      "ODD_EVEN":"E",
      "BLOCK_NUM":"0001",
      "MAPBLKLOT":"0001001"
   }
}
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

シンプルなパフォーマンス比較

JSON とText型 によるデータ比較
インデックスの無い、206,000件のドキュメントデータ参照

```
# as JSON type

SELECT DISTINCT
 feature->"$.type" as json_extract
FROM features;
+--------------+
| json_extract |
+--------------+
| "Feature"    |
+--------------+
1 row in set (1.25 sec)
```

```
# as TEXT type

SELECT DISTINCT
 feature->"$.type" as json_extract
FROM features;
+--------------+
| json_extract |
+--------------+
| "Feature"    |
+--------------+
1 row in set (12.85 sec)
```

JSON形式のバイナリ形式は、非常に効率の良い検索を実装しています。
テキスト型でのトラバーサルと比較し、約10倍以上のパフォーマンス。

※ MySQLの5.7.9以降で->演算子はJSON_EXTRACT()関数の別名(省略形)として機能します。
左側は列identieferで右側 は JSONパスになります。

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

シンプルなパフォーマンス比較

追加で json_extract()ファンクションにて条件を変更し参照

```
# as JSON type
[NEW57]> SELECT distinct
json_extract(feature,'$.type')
as feature FROM features;
+-----------+
| feature   |
+-----------+
| "Feature" |
+-----------+
1 row in set (1.25 sec)

# as JSON type
[NEW57]> SELECT count(feature)
FROM features where
json_extract(feature,'$.properties')
like '%BEACH%';
+----------------+
| count(feature) |
+----------------+
|            235 |
+----------------+
1 row in set (2.94 sec)
```

```
# as TEXT type
[NEW57]> SELECT distinct
json_extract(feature,'$.type')
as feature FROM features;
+-----------+
| feature   |
+-----------+
| "Feature" |
+-----------+
1 row in set (9.48 sec)

# as TEXT type
[NEW57]> SELECT count(feature)
FROM features where
json_extract(feature,'$.properties')
like '%BEACH%';
+----------------+
| count(feature) |
+----------------+
|            235 |
+----------------+
1 row in set (10.11 sec)
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Program Agenda

1. Introduction
2. JSON datatype
3. Indexing of JSON data
4. Functions to handle JSON data
5. Misc Supporting Features

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Generated Columns(生成列)概要

id	my_integer	my_integer_plus_one
1	10	11
2	20	21
3	30	31
4	40	41

列は定義に基づいて、
自動的に生成されます。

```
CREATE TABLE t1 (
 id INT NOT NULL PRIMARY KEY auto_increment,
 my_integer INT,
 my_integer_plus_one INT AS (my_integer+1)
);
```

参照のみ可能

```
UPDATE t1 SET my_integer_plus_one = 10 WHERE id = 1;
ERROR 3105 (HY000): The value specified for generated
column 'my_integer_plus_one' in table 't1' is not allowed.
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Generated ColumnsによるIndexサポート

206,000件のドキュメントデータに対してのテーブルスキャンから,
マテリアライズされた206,000件のデータへのインデックススキャン。

```
ALTER TABLE features ADD feature_type VARCHAR(30) AS (feature->"$.type");
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

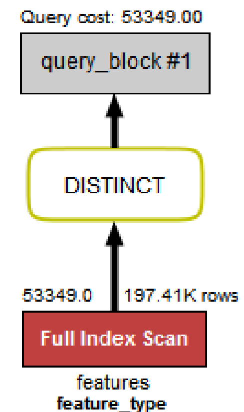
メタデータのみが変更されます(FAST). テーブルへのアクセスは不要

```
ALTER TABLE features ADD INDEX (feature_type);
Query OK, 0 rows affected (0.73 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

Indexのみ作成
テーブルの行変更は行わない

```
SELECT DISTINCT feature_type FROM features;
+--------------+
| feature_type |
+--------------+
| "Feature"    |
+--------------+
1 row in set (0.06 sec)
```

1.25秒から 0.06秒
実行時間の短縮



Copyright©2015 Oracle and/or its affiliates. All rights reserved.

参考) Generated ColumnsによるIndexサポート

JSON_EXTRACT()で実行する場合

```
ALTER TABLE features ADD feature_type VARCHAR(30) AS (json_extract(feature,'$.type'));
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0

ALTER TABLE features ADD INDEX (feature_type);
Query OK, 0 rows affected (1.59 sec)
Records: 0  Duplicates: 0  Warnings: 0

[NEW57]> desc features;
+--------------+-------------+------+-----+---------+-------------------+
| Field        | Type        | Null | Key | Default | Extra             |
+--------------+-------------+------+-----+---------+-------------------+
| id           | int(11)     | NO   | PRI | NULL    | auto_increment    |
| feature      | json        | NO   |     | NULL    |                   |
| feature_type | varchar(30) | YES  | MUL | NULL    | VIRTUAL GENERATED |
+--------------+-------------+------+-----+---------+-------------------+
3 rows in set (0.00 sec)

[NEW57]> SELECT distinct feature_type as feature FROM features;
+-----------+
| feature   |
+-----------+
| "Feature" |
+-----------+
1 row in set (0.09 sec)
```

1.25秒から 0.09秒
実行時間の短縮

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Generated ColumnsによるIndexサポート

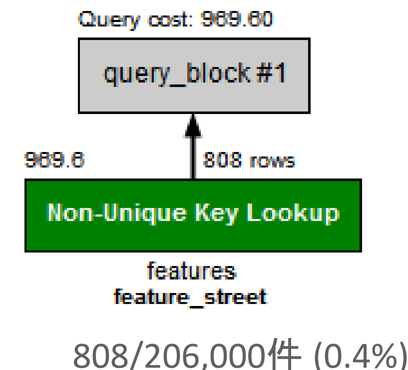
EXPLAINによるインデックス利用状況の確認

ALTER TABLE features ADD feature_street
VARCHAR(30) AS (json_extract(feature,'$.properties.STREET'));

ALTER TABLE features ADD INDEX (feature_street);

```
[NEW57]> desc features;
+----------------+-------------+------+-----+---------+-------------------+
| Field          | Type        | Null | Key | Default | Extra             |
+----------------+-------------+------+-----+---------+-------------------+
| id             | int(11)     | NO   | PRI | NULL    | auto_increment    |
| feature        | json        | NO   |     | NULL    |                   |
| feature_type   | varchar(30) | YES  | MUL | NULL    | VIRTUAL GENERATED |
| feature_street | varchar(30) | YES  | MUL | NULL    | VIRTUAL GENERATED |
+----------------+-------------+------+-----+---------+-------------------+
```

select feature from features where
json_extract(feature,'$.properties.STREET') = '"MARKET"';



808/206,000件 (0.4%)

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Generated Columns (続き)

- Functional Index(関数インデックス)を使用
- VIRTUAL (規定値)の他にSTOREDが利用可能

```
ALTER TABLE features ADD feature_type varchar(30) AS
(feature->"$.type") STORED;
Query OK, 206560 rows affected (4.70 sec)
Records: 206560  Duplicates: 0  Warnings: 0
```

- どちらの生成列もインデックス追加が可能

ORACLE
Copyright©2015 Oracle and/or its affiliates. All rights reserved.

使用可能なインデックスオプション

STORED	VIRTUAL
Primary and Secondary	Secondary Only
BTREE, Fulltext, GIS	BTREE Only
Mixed with fields	Mixed with virtual column only
Requires table rebuild	No table rebuild
Not Online	INSTANT Alter
	Faster Insert

Bottom Line: 主キー, FULLTEXTまたは仮想GISインデックスを必要とする場合を除きDefaultのVIRTUALを選択した方が良いケースが多い。

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Virtual vs. Stored パフォーマンスの違い

テーブルスキャンが発生する場合のパフォーマンス比較:

```
SELECT DISTINCT feature_type
FROM features;
+----------------+
| feature_type |
+----------------+
| "Feature"      |
+----------------+
```

```
VIRTUAL TEXT型 (9.89 sec)
STORED  TEXT型 (0.22 sec)
VIRTUAL JSON型 (0.85 sec)
STORED  JSON型 (0.24 sec)
```

Clarification: Since indexes are materialized (*stored*) themselves, the real-life case for STORED is when generating the column is computationally expensive *and* you can not use indexes effectively.

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Program Agenda

1. Introduction
2. JSON datatype
3. Indexing of JSON data
4. Functions to handle JSON data
5. Misc Supporting Features

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

JSONファンクション:Handle JSON Data

5.7では、CREATE, SEARCH, MODIFY and RETURN等の
JSONファンクションをサポートしています

- Info
  - JSON_VALID()
  - JSON_TYPE()
  - JSON_KEYS()
  - JSON_LENGTH()
  - JSON_DEPTH()
  - JSON_CONTAINS()
  - JSON_CONTAINS_PATH()
- Modify
  - JSON_REMOVE()
  - JSON_APPEND()
  - JSON_SET()
  - JSON_INSERT()
  - **JSON_REPLACE()**
  - JSON_ARRAY_INSERT()

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSONファンクション:Handle JSON Data

- Create
  - JSON_MERGE()
  - **JSON_ARRAY()**
  - **JSON_OBJECT()**
- Get data
  - **JSON_EXTRACT()**
  - **JSON_SEARCH()**
- Helper
  - JSON_QUOTE()
  - **JSON_UNQUOTE()**

参照）https://dev.mysql.com/doc/refman/5.7/en/json-functions.html

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSON Data Extract

JSONドキュメントからデータを抽出

```
SELECT json_extract(feature,'$.type') as
json_extract FROM features limit 1;
+--------------+
| json_extract |
+--------------+
| "Feature"    |
+--------------+

SELECT feature->"$.type" as
json_extract FROM features limit 1;
+--------------+
| json_extract |
+--------------+
| "Feature"    |
+--------------+
```

->演算子はJSON_EXTRACT()の省略形として機能し、 左側は列識別子で右側 は JSONパス

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

参考) JSON_EXTRACT()のエイリアス

- <field> **->** <JSON path expression string>
  data->'$.some.key[3].from.doc'
- Syntactic sugar over JSON_EXTRACT function
- SELECT * FROM employees WHERE data->'$.id'= 2;
- ALTER … ADD COLUMN id INT AS (data->'$.id') …
- CREATE VIEW .. AS SELECT data->'$.id', data->'$.name' FROM …
- UPDATE employees SET data->'$.name'='Samantha' WHERE …

※ MySQLの5.7.9以降で->演算子はJSON_EXTRACT()関数の別名(省略形)として機能します。
左側は列identieferで右側 は JSONパスになります。

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

参考) JSON_EXTRACT()のエイリアス:制限

	JSON_EXTRACT エイリアス	JSON_EXTRACT関数
Data source	A table's field only	Any JSON-typed value
Path expression	A string constant only	Any string-typed value
Number of path expressions	Only one	Multiple

ORACLE

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

Unquote JSON String

特殊文字のエスケープシーケンス

```
SELECT
 DISTINCT JSON_UNQUOTE(feature->"$.type")
 as feature_type FROM features;
+-----------------+
| feature_type    |
+-----------------+
| Feature         |
+-----------------+
```

```
SELECT DISTINCT json_extract(feature,'$.type'),
JSON_UNQUOTE (json_extract(feature,'$.type'))
as JSON_UNQUOTE FROM features;
+-------------------------------+--------------+
| json_extract(feature,'$.type') | JSON_UNQUOTE |
+-------------------------------+--------------+
| "Feature"                      | Feature      |
+-------------------------------+--------------+
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

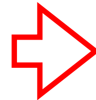
JSON Path Search

JSON形式のドキュメント内の指定された値のパスを返します。

To retrieve via: *[[database.]table.]column->"$<path spec>"*

feature: {"type": "Feature", "geometry": {"type": "Polygon", "coordinates": [[[-122.39790233801507, 37.790726654724864, 0], [-122.39823963293078, 37.79099174693105, 0], [-122.39835208359005, 37.79090296883558, 0], [-122.3986901921814, 37.79116869825866, 0], [-122.39823249443299, 37.7915300431353, 0], [-122.39756221186288, 37.79099545718336, 0], [-122.39790233801507, 37.790726654724864, 0]]]}, "properties": {"TO_ST": "425", "BLKLOT": "3709016", "STREET": "MARKET", "FROM_ST": "425", "LOT_NUM": "016", "ST_TYPE": "ST", "ODD_EVEN": "O", "BLOCK_NUM": "3709", "MAPBLKLOT": "3709014"}}


```
SELECT JSON_SEARCH(feature,
 'one', 'MARKET') AS
 extract_path
FROM features
WHERE id = 121254;
+-------------------------+
| extract_path            |
+-------------------------+
| "$.properties.STREET"   |
+-------------------------+
1 row in set (0.00 sec)
```



```
SELECT
feature->"$.properties.STREET"
 AS property_street
FROM features
WHERE id = 121254;
+-----------------+
| property_street |
+-----------------+
| "MARKET"        |
+-----------------+
1 row in set (0.00 sec)
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSON Array Creation

値の(空を含む)リストを評価し、それらの値を含むJSON配列を返します。

```
SELECT JSON_ARRAY(
id,feature->"$.properties.STREET",
feature->"$.type") AS json_array
FROM features ORDER BY RAND() LIMIT 3;
+----------------------------------+
| json_array                       |
+----------------------------------+
| [126213, "FOLSOM", "Feature"]    |
| [78171, "44TH", "Feature"]       |
| [148660, "UNDERWOOD", "Feature"] |
+----------------------------------+
3 rows in set (1.21 sec)
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSON Object Creation

キー/値のペアの（空を含む）リストを評価し、それらのペアを含むJSONオブジェクトを返します。任意のキー名がNULLであるか引数の数が奇数である場合、エラーが発生します。

```
SELECT JSON_OBJECT(
'id', id,
'street', feature->"$.properties.STREET",
'type', feature->"$.type"
) AS json_object FROM features ORDER BY RAND() LIMIT 3;
+----------------------------------------------------------+
| json_object                                              |
+----------------------------------------------------------+
| {"id": 181215, "type": "Feature", "street": "ROUSSEAU"} |
| {"id": 81215, "type": "Feature", "street": "QUINTARA"}  |
| {"id": 113788, "type": "Feature", "street": "NOE"}      |
+----------------------------------------------------------+
3 rows in set (2.02 sec)
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSON_REPLACE

JSON文書の既存の値を置き換え、その結果を返します。

```
SELECT json_extract(feature,'$.type'),JSON_ARRAY('feature',
'bug') as json_object FROM features LIMIT 1;
```

```
+--------------------------------+---------------------+
| json_extract(feature,'$.type') | json_object         |
+--------------------------------+---------------------+
| "Feature"                      | ["feature", "bug"]  |
+--------------------------------+---------------------+
```



```
SELECT JSON_REPLACE(feature, '$.type', JSON_ARRAY('feature',
'bug')) as json_object FROM features LIMIT 1;
+------------------------------------------------------+
| json_object                                          |
+------------------------------------------------------+
| {"type": ["feature", "bug"], "geometry": {"type": ..}} |
+------------------------------------------------------+
```

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Program Agenda

1. Introduction
2. JSON datatype
3. Indexing of JSON data
4. Functions to handle JSON data
5. Misc Supporting Features

ORACLE®

Copyright © 2014 Oracle and/or its affiliates. All rights reserved. |

JSON or Column(列)?

- JSONまたはColumnにするかは用途による
- 両方共にそれぞれのアプローチの利点

```
+--------------+-------------+---------------+
|   id         |     type    |    street     |
+--------------+-------------+---------------+
| 190838       | Feature     | FARALLONES    |
| 153676       | Feature     | ELLSWORTH     |
| 143094       | Feature     | JENNINGS      |
+--------------+-------------+---------------+

+--------------------------------------------------------------+
| json_object                                                  |
+--------------------------------------------------------------+
| {"id": 190838, "type": "Feature", "street": "FARALLONES"} |
| {"id": 153676, "type": "Feature", "street": "ELLSWORTH"}  |
| {"id": 143094, "type": "Feature", "street": "JENNINGS"}   |
+--------------------------------------------------------------+
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Storing as a Column

- アプリケーションに対し、スキーマの適用が容易
- スキーマにより長期的にアプリケーションの変更管理がコントロールしやすい。
  - 多くの順列を考慮しなくても良い
  - データにいくつかの制約設定を可能

```
+--------------+-------------+---------------+
|   id         |     type    |    street     |
+--------------+-------------+---------------+
| 190838       | Feature     | FARALLONES    |
| 153676       | Feature     | ELLSWORTH     |
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Storing as JSON

- スキーマ内でモデル化する事が困難であるデータを表現する為のより柔軟な方法。

- 例えば,多くのお客様へSaaSアプリケーション提供してる場合
- 強力なユースケースとしてカスタムフィールドをサポート
- 歴史的に エンティティ属性値モデル(EAV)が使われているが、必ずしも十分に機能していない場合等

```
+-----------------------------------------------------------+
| json_object                                               |
+-----------------------------------------------------------+
| {"id": 190838, "type": "Feature", "street": "FARALLONES"} |
| {"id": 153676, "type": "Feature", "street": "ELLSWORTH"}  |
```

ORACLE
Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Storing as JSON (続き.)

- 容易な非正規化
  特定の状況 において重要である最適化手法
- 労力を伴うスキーマ変更不要*
- 容易なプロトタイピング
  - 考慮すべき型 が少なくて済む
  - スキーマの強制不要,直ぐに値を格納

* MySQL 5.6からはオンラインDDLが実装されているため、
  MySQL5.6以降をご利用の場合は,以前程大きな影響はありません。

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

その他) Schema + Schemaless

ColumnとJSONの連携

```
CREATE TABLE pc_components (
 id INT NOT NULL PRIMARY KEY,
 description VARCHAR(60) NOT NULL,
 vendor VARCHAR(30) NOT NULL,
 serial_number VARCHAR(30) NOT NULL,
 attributes JSON NOT NULL
);
```

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

JSONドキュメントの圧縮

- 圧縮効率が良い
  - スキーマレスの為、キーは各ドキュメント内で繰り返されて使われている。
- 反復は圧縮効率を高めます。
- MySQL 5.7 では32KBと64KB pagesをサポート (improved compression)

ORACLE®

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

MySQL 5.7 Page 圧縮

- InnoDB MySQL 5.1のプラグインから圧縮を実装
- MySQL 5.7 からはシンプルで新しいページの圧縮方法を実装しました

  (制限:対応カーネル等が必要です)

ORACLE®

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

圧縮パフォーマンス (16K Page)

```
SELECT name, ((file_size-
allocated_size)*100)/file_size as compressed_pct
from information_schema.INNODB_SYS_TABLESPACES
WHERE name like 'test/features';
+----------------+-----------------+
| name           | compressed_pct |
+----------------+-----------------+
| test/features |         59.2634 |
+----------------+-----------------+
1 row in set (0.01 sec)
```

Using real-life data set from earlier

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

補足)　制限事項

- JSONの列に格納されるJSON文書のサイズは、
max_allowed_packetシステム変数の値に制限されています。

(While the server manipulates a JSON value internally in memory, it can be larger; the limit applies when the server stores it.)

- JSONの列はデフォルト値を持つことができません。

max_allowed_packet

Command-Line Format	--max_allowed_packet=#	
System Variable	Name	max_allowed_packet
	Variable Scope	Global
	Dynamic Variable	Yes
Permitted Values	Type	integer
	Default	4194304
	Min Value	1024
	Max Value	1073741824

The maximum size of one packet or any generated/intermediate string, or any parameter sent by the mysql_stmt_send_long_data() C API function. The default is 4MB.

参照： https://dev.mysql.com/doc/refman/5.7/en/json.html
https://dev.mysql.com/doc/refman/5.6/ja/server-system-variables.html#sysvar_max_allowed_packet

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

今後のロードマップ

- In-place partial update of JSON/BLOB *(performance)*
- Partial streaming of JSON/BLOB (*replication*)
- Full text and GIS index on virtual columns
  - Currently works for "STORED"
- Improved performance through condition pushdown
- Online alter for virtual columns
- Advanced JSON functions

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

参考)

- http://mysqlserverteam.com/
- http://mysqlserverteam.com/tag/json/
- https://dev.mysql.com/doc/refman/5.7/en/mysql-nutshell.html
- http://dev.mysql.com/doc/relnotes/mysql/5.7/en/
- https://dev.mysql.com/doc/refman/5.7/en/json.html
- https://dev.mysql.com/doc/refman/5.7/en/json-functions.html
- http://www.thecompletelistoffeatures.com

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

Hardware and Software

Engineered to Work Together

ORACLE

Copyright©2015 Oracle and/or its affiliates. All rights reserved.

ORACLE®