MySQL Document Store: Under the Hood

@MySQL Innovation Day 2016

日本オラクル株式会社
MySQL Global Business Unit
MySQL Principal Sales Consult /Shinya Sugiyama

ORACLE

MySQL

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

1. What is MySQL Document Store
2. Foundations of X DevAPI and X Protocol
3. MySQL Shell Overview
4. Features
5. Limitations
6. Future

What is MySQL Document Store

ドキュメントストアとしてのMySQL

DOCUMENT STORE

ドキュメントストア

- 形式・様式
- ツリー構造
- スキーマレス

```
{ '_id': 1,
  'name': 'Milk',
  'properties': {
     'volume_l': 1,
     'amount': 32
   }
}
```

- 高速な検索
- 更新の最適化
- オペレーション
  - 検索、更新、変更、削除
  - プロジェクション
  - フィルタリング
  - アグリゲーション

MYSQL DOCUMENT STORE

MySQLにおけるドキュメントストア

- MySQL 5.7.12に実装
- X DevAPI
  - MySQL Shell, Connector/J, C/.net and C/node.js
- X Protocol
  - X Pluginにより実装
- MySQL 5.7 GAから安定したインタフェースを利用可能
  - SQLサービス
  - JSONサポート (データ型・ファンクション)
    +Generated ColumnとIndex

Foundations of X DevAPI and X Protocol

X DevAPI and X Protocol概要

X DevAPI

X Plugin (MySQL) ⇔ X Protocol ⇔ X DevAPI (Driver)

- X Pluginを有効にする事で、X Protocol経由で通信可能
- ドキュメント及びテーブル共に処理可能
- NoSQLライクな構文でドキュメントに対しCRUD処理可能
- Fluent API
  - Connector/Node.js, Connector/J, Connector/.Net, MySQL Shell

```
select PLUGIN_NAME,PLUGIN_VERSION,PLUGIN_DESCRIPTION
from plugins where PLUGIN_NAME = 'mysqlx';
+-------------+----------------+--------------------+
| PLUGIN_NAME | PLUGIN_VERSION | PLUGIN_DESCRIPTION |
+-------------+----------------+--------------------+
| mysqlx      | 1.0            | X Plugin for MySQL |
+-------------+----------------+--------------------+
```

```
prod = sess.getSchema("prod")
res = prod.users.
    find("$.name = 'Milk'").
    fields(["name", "properties"])
```

X Protocol

A NEW PROTOCOL

The X Protocol focuses on:
- extensibility
- performance
- security

クエリ共通セット:
- CRUD ==大量に小さなPKを処理
- 独立した複数のクエリーを処理
- 大量のデータを処理
  - シャーディング

- 一般的なオペレーションに最適化
  - 同期、非同期をサポート (X DevAPI)
  - メッセージサイズ削減
- パイプライン方式
  - 応答を待たずにメッセージ送信可能
  - Expectationsによるエラー処理設定
- 警告・通知(global/local)

Pipelining messages is a core feature of the Mysqlx Protocol.
It sends messages to the server without waiting for a response to save latency. (no mandatory handshake)
参考) https://github.com/mysql/mysql-server/blob/5.7/rapid/plugin/x/protocol/mysqlx_expect.proto
https://dev.mysql.com/doc/internals/en/x-protocol-messages-message-structure.html

X Protocol

BUILDING BLOCKS

- Protobufによるシリアル処理
- SASL方式による認証
  - 認証フレームワーク(SASL)
  - PLAIN over TLS
  - MYSQL41
- TLSによる暗号化

protocol buffers: a language-neutral, platform-neutral, extensible way of serializing structured data for use in communications protocols, data storage, and more.

PROTOBUF

```
message Find {
required Collection collection = 2;
optional DataModel data_model = 3;
repeated Projection projection = 4;
optional Mysqlx.Expr.Expr criteria = 5;
repeated Mysqlx.Datatypes.Scalar args = 11;
optional Limit limit = 6;
repeated Order order = 7;
repeated Mysqlx.Expr.Expr grouping = 8;
optional Mysqlx.Expr.Expr grouping_criteria = 9;
};
```

```
static ngs::Authentication_handler_ptr create(ngs::Session *session)
  {
    return Authentication_handler::wrap_ptr(new Sasl_mysql41_auth((xpl::Session*)session));
  }
```

X Protocol

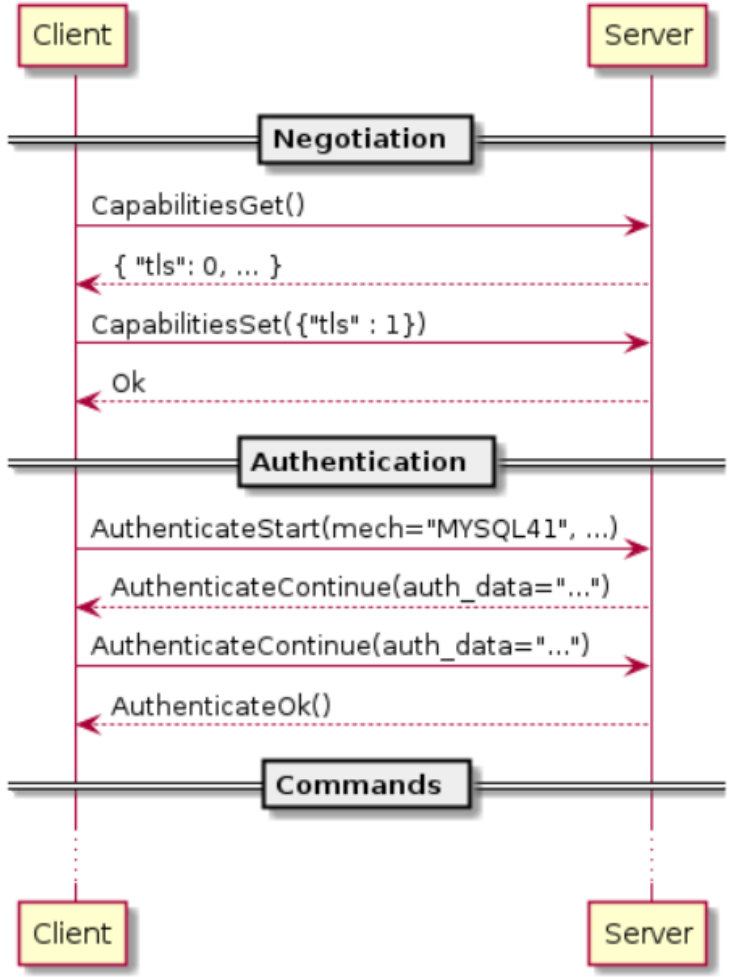
Stages of Session Setup

TLS extensionがサポートされていれば,CapabilitiesSetが成功

```
== Negotiation ==
Client -> Server: CapabilitiesGet()
Server --> Client: { "tls": 0, ... }

Client -> Server: CapabilitiesSet({"tls" : 1})
Server --> Client: Ok

== Authentication ==
Client -> Server: AuthenticateStart(mech="MYSQL41", ...)
Server --> Client: AuthenticateContinue(auth_data="...")
Client -> Server: AuthenticateContinue(auth_data="...")
Server --> Client: AuthenticateOk()

== Commands ==
...
```

参考) https://dev.mysql.com/doc/internals/en/x-protocol.html

TLS can be negotiated at connection setup. This negotiation step is optional and can be skipped.



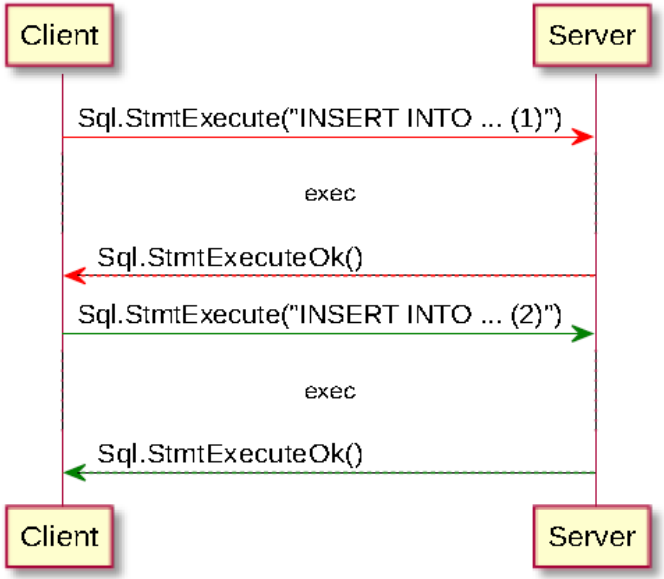
X Protocol

Query Time

Stage	Time
network path latency	1ms
exectime	0. 1ms

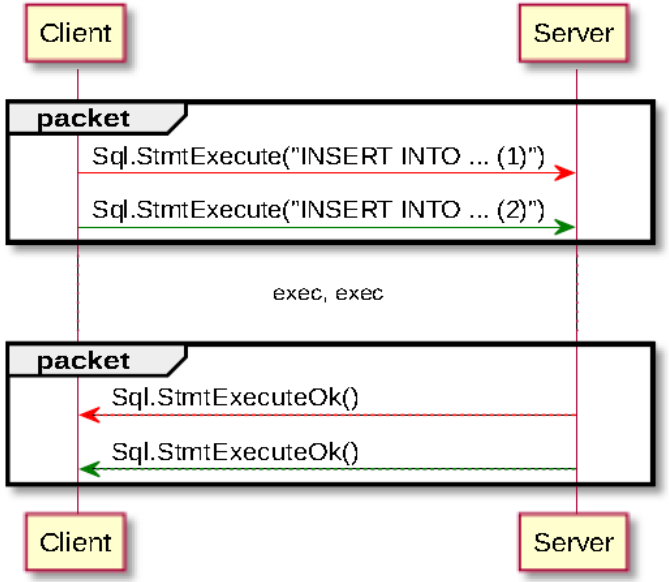
- CLASSIC REQUEST/RESPONSE

Total: 4x path + 2x exectime = **4.2ms**



- PIPELINING

Total: 2x path + 2x exectime = **2.2ms**



X Protocol

Default: Mysqlx.Expect::Open() without parameter == Mysqlx.Expect::Open([-no_error])

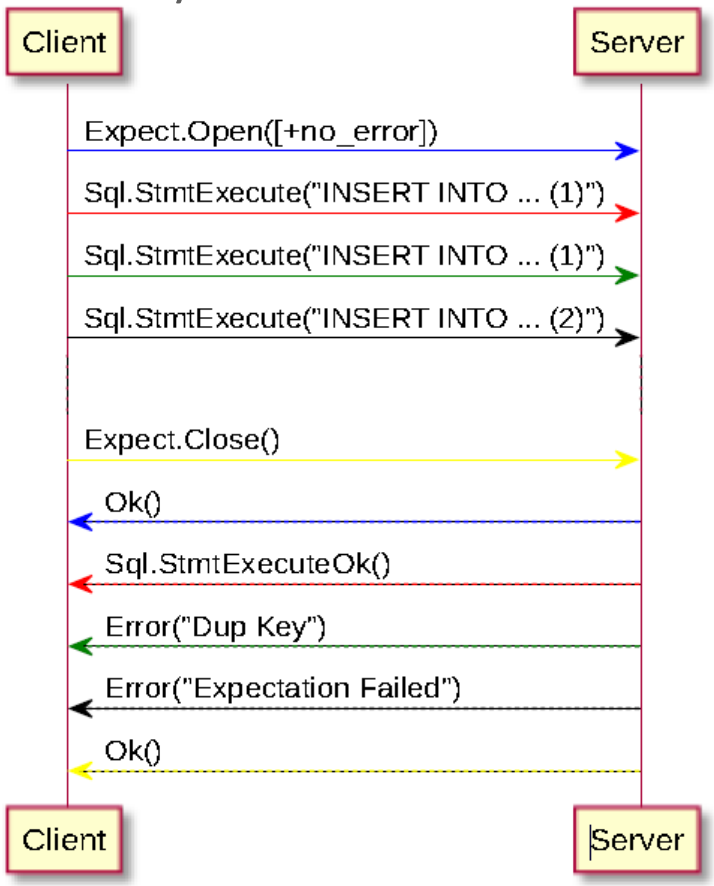
Managing Expectations

- パイプライン方式はエラー処理が難しい
  - 想定は通常間違っている
  - どのタイミングで失敗させる?
    例) 最初のエラーで失敗? エラーを無視して続行するか?
    Mysqlx.Expect::Open([+no_error]) | Mysqlx.Expect::Open([-no_error])

With expectations pipelined, the server will handle errors in a consistent, reliable way. In case error reporting isn't a major topic one can combine multi-row INSERT with pipelining and reduce the per-row network overhead. This is important in case the network is saturated.

- ステートメント
  - レプリケーションラグ?
  - 最大実行時間

Fail Early



error-msg: Expectation failed: %s

参考 : https://dev.mysql.com/doc/internals/en/x-protocol-expect-setting-expectations.html

X Protocol

Managing Expectations: Nesting

Nest (Expect blocks can be nested)

```
Expect.Open([+no_error])
  PrepStmt.Prepare(id=1, "INSERT INTO ... (?)")
  Expect.Open([+no_error])
    PrepStmt.Execute(id=1, values=[1])
    PrepStmt.Execute(id=1, values=[2])
  Expect.Close()
  PrepStmt.Close(id=1)
Expect.Close()
```

Condition	Key
no_error	1
schema_version	2
gtid_executed_contains	3
gtid_wait_less_than_ms	4

Mysqlx.Expect::Open([+no_error]):
Fail all messages of the block after the first message returning an error.

参考) https://dev.mysql.com/doc/internals/en/x-protocol-expect-conditions.html

X Protocol

Reduced Message Sizes

- ✓ Compression via zlib's DEFLATE method (Not in 5.7.12 yet)
- ✓ Space efficient variable length integer encoding
- ✓ Protocol buffer is compact binary format

```
message ColumnMeta {
optional name string = 1;
optional orig_name string = 2;
optional catalog string = 3;
// ...
}
```

- orig_name がnameと同じであれば、orig_nameは送信しない
- catalogが“def”であれば、catalogは送信しない

MySQL Shell Overview

MySQL Shell概要

MySQL Shell Overview

What is it?

- 一般的なスクリプト・インターフェースを介して利用可能なMySQLコマンドラインクライアント。

- PythonやJavaScriptなどのスクリプト言語でさまざまな製品と対話するための完全な開発用APIを提供。

- 開発APIを公開する等の用途の為に、ライブラリとして、MySQL WorkbenchおよびVisual Studio MySQLのプラグインの様に他の製品に組み込む事が出来ます。

参考) http://dev.mysql.com/doc/refman/5.7/en/mysql-shell-visual-studio.html

MySQL Shell Overview

Benefits

- シングルエントリーポイント
  - 最も一般的な利用例として、複数MySQLを一か所で利用する事が可能。
  - オペレーション毎に、複数製品を切り替える必要が無い。

- 標準的な言語
  - 同じ言語で異った製品を操作可能
  - 複数製品の詳細を個々に学ぶ必要が無く、各製品サポートに必要な、API拡張機能を同じ言語を利用して学ぶ必要があるのみ

Features

機能

Features

Multiple Language Support

- サポートされる言語
  - MySQL Shellは、次の言語でコードの実行をサポートしています :
    - JavaScript
    - Python
    - SQL

```
[root@misc01 admin]# mysqlsh --help | egrep -i "Start in"
  --sql                 Start in SQL mode using a node session.
  --sqlc                Start in SQL mode using a classic session.
  --js                  Start in JavaScript mode.
  --py                  Start in Python mode.
[root@misc01 admin]#
```

Features

Shell Commands

- シェルの対話モードでは特別なエスケープステートメントが利用可能です。

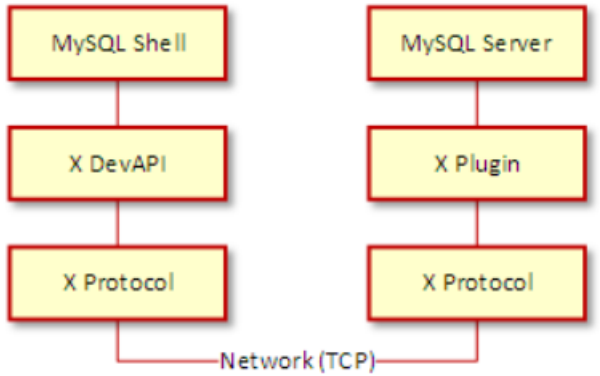
- 通常の文の処理ロジックから除外され以下の用途で利用されます:

–ヘルプの表示
–アクティブモード設定(language)
–ファイルからスクリプトの実行
–MySQL ShellのQuit
–セッションの確立
–Warning生成の有効・無効化
–セッションステータスのプリント
–保存されたセッションの管理

Features

Global Session

- セッションの確立:
  – コマンドライン引数
  – シェルコマンド
  – スクリプト



- コマンドライン引数若しくは,シェルコマンドでグローバルセッションが確立された時: 全てのシェルモードが利用可能です。

The Global Session is used to execute statements in SQL mode and the same session is available in both Python or JavaScript modes.

Features

Development API

開発APIは、MySQL製品群を JavaScriptとPython用に公開しています。
開発APIは、下記機能に対応しています:

–セッション処理
–スキーマ処理
–MySQLをドキュメントデータベースとして扱う: コレクション
–MySQLをリレーショナルデータベースとして扱う: テーブル
–テーブルとコレクションに対しての、CRUD処理
–SQLの実行
–結果の処理
–CRUDとSQL処理パラメータのバインディング

Features

Interactive Mode

- MySQLシェルが開始されると、それが有効な言語のラインインタープリタとなります。
- シェルコマンドの実行が可能。
- 未割り当てのCRUD＆SQL操作が自動実行されます:
  - CRUD and SQL実行時の、*execute()*のCallはオプションです。
- 結果の自動処理を行います:
  - 結果は自動的に画面に 表示されます。

Features

Batch Mode

- MySQLはサポートされている言語のいずれかで記述されたバッチ処理スクリプトをサポートします。

- JavaScriptとPythonのスクリプトは完全に指定する必要があり:
  - CRUDとSQLの自動実行はしない。結果は、スクリプトを使用し利用可能。
  - シェルコマンドは、スクリプトからサポートされていません。

Features

Output Formats

- サポートされる出力形式: Table, JSON, フォーマットJSON.
  - – Table
    デフォルト: ほぼ全てのSQLとCRUDオペレーション結果を表示。
  - – Formatted JSON
    JSONデータを人が理解しやすいようにフォーマットして表示。
  - – JSON
    他のツールと連携の為に、JSONデータをそのまま表示。

- MySQLシェルを起動するときに、デフォルトのフォーマットを、コマンドライン引数で変更することが可能です。

Features

Stored Sessions (~/.mysqlsh/stored_sessions.json)

永続的な場所にMySQLサーバ接続データを複数保持する事が可能

- 保存されたセッションは、名前に関連付けられています。
- 保存されたセッション名を利用してセッションを確立する事が可能です。
- 保存されたセッションは、シェルコマンドを使用して管理します。
- 保存されたセッションは、シェルモジュールを介して開発APIで利用可能です。

```
mysql-js> ¥addconn demo_conn demo_user@localhost:33060/NEW57
mysql-js> ¥lsconn
{
    "demo_conn": {
        "dbUser": "demo_user",
        "host": "localhost",
        "port": 33060,
        "schema": "NEW57"
    } ……
```

```
mysql-js> ¥connect $demo_conn
Closing old connection...
Using 'demo_conn' stored connection
```

Features

Application Log

- MySQLのシェルは、様々な問題の重要度に応じて、アプリケーションログファイルを生成するように構成することが出来ます。
- 実行中のMySQLシェルの状態を確認するために利用する事が出来ます。

```
shell>  echo $HOME
/root
shell>  tail /root/.mysqlsh/mysqlsh.log
2016-04-26 03:55:19: Error: Can't parse  Error (5115): Document is missing a required field
```

- ログは-log-levelコマンドライン引数を使用して、次のようなレベルで定義可能

```
shell>  mysqlsh --log-level=4
```

Log Level	Meaning
1	None, the default
2	Internal Error
3	Error
4	Warning

Log Level	Meaning
5	Info
6	Debug
7	Debug2
8	Debug3

Features

Backward Compatibility

```
# mysqlsh -u demo_user -ppassword --classic --schema=world
```

- MySQL Shellの主な機能としてはJavaScript及びPythonに対して スクリプティングインターフェースを提供し、Xプロトコル経由で MySQLをドキュメントデータベースとして処理する機能です。
- SQLモードは、SQL文をそのまま実行する為に追加されました。
- MySQLシェルはクラッシックなMySQLプロトコルを利用して MySQLに接続する事も可能です。
  - –SQLモードでは、SQL文をそのまま実行可能
  - –開発APIの利用は限定されます

Classic Session: Use this session type to interact with MySQL Servers that do not have the X Protocol enabled. The development API available for this type of session is very limited. For example, there are no CRUD operations, no collection handling, and binding is not supported.

Limitations

制限事項

Limitations
制限事項

- MySQLシェルはCLIではありません: MySQLシェルでは
サーバに付属のCLIで使用可能な多くの機能を利用出来ません。

- それらの多くは、将来的にMySQLシェルに入るかもしれませんが、
現状、その優先度は高くありません。

Future

将来的な機能拡張

Future

- Admin API:

Admin APIは、JavaScript とPython共に含まれ有効になっています。

– Group Replicationの設定
– MySQLインスタンスのクローニング
– MySQLインスタンスをまとめて管理

ユーティリティ統合

– MySQL Utilitiesに実装されている機能を、MySQL Shellを介して実行出来るように進めています。

- Development API:

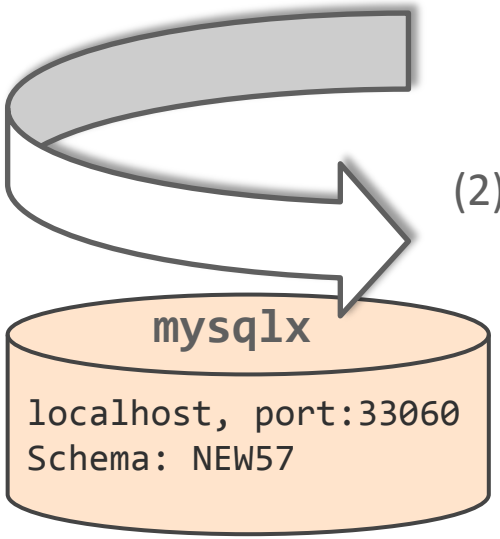
より良いユーザーエクスペリエンスを実現する為に、development API により多くの機能を実装し公開していきます。

DEMO) MySQL Shell

例) mysqlsh(--py)による,JSONデータのバッチ取り込み処理

Twitter API:
https://api.twitter.com/1.1/xxxx

OAuth1Session



(1)Twitter APIに接続しJSONフォーマットのTweetを取得

```
Shell> mysqlsh --py < demo_python_twitter.py
```

(2) 取得したJSONデータを.addでテーブルに追加

```
myDb = mySession.getSchema('NEW57')
省略…
timeline = json.loads(res.text)
for tweet in timeline:
myDb.X_PYTHON.add(tweet).execute()
```

```
+-------+-------------+------+-----+---------+-------------------+
| Field | Type        | Null | Key | Default | Extra             |
+-------+-------------+------+-----+---------+-------------------+
| doc   | json        | YES  |     | NULL    |                   |
| _id   | varchar(32) | NO   | PRI | NULL    | STORED GENERATED  |
| name  | varchar(64) | YES  | MUL | NULL    | STORED GENERATED  |
+-------+-------------+------+-----+---------+-------------------+
```

(3) Generated Columnを利用してデータ参照処理

参考) HOW TO WRITE YOUR OWN CLIENT

Spec:

http://dev.mysql.com/doc/internals/en/x-protocol.html

Message Def:

https://github.com/mysql/mysql-server/tree/5.7/rapid/plugin/x/protocol

Protobuf

https://developers.google.com/protocol-buffers/

```
$ protoc  -I  --python_out=... .../mysql.proto
```

有難うございました

Q&A①

Q: JSONデータの格納領域について?

A: MySQL5.7においてDefaultのInnoDBフォーマットになったBarracudaでは、
JSONデータ型においても、BINARYやTEXTカラムと同じように、データに対する
20バイトのポインタだけをクラスタインデックスに残し、すべてのデータを
オーバーフローページに格納するようになっています。

When a table is created with ROW_FORMAT=DYNAMIC or ROW_FORMAT=COMPRESSED, long variable-length column values (for VARBINARY, VARCHAR, BLOB, TEXT, and **JSON** columns) are stored fully off-page, and the clustered index record contains only a 20-byte pointer to the overflow page. InnoDB will also store long CHAR column values off-page if the column value is greater than or equal to 768 bytes, which can occur when the maximum byte length of the character set is greater than 3, as it is with utf8mb4, for example.

参照) http://dev.mysql.com/doc/refman/5.7/en/innodb-row-format-dynamic.html

ORACLE

Q&A②

Q: mysqlshにおける--sql 、--sqlcで接続した場合の違い?

--sql	Start in SQL mode using a node session.
--sqlc	Start in SQL mode using a classic session.

A:以下の例のように、X Protocol経由での接続かClassicなMySQL Protocol経由での接続になるかの違いがあります。X Protocol経由であれば、pipelines処理可能ですので、ネットワークLatencyの影響を受けにくくなる事が期待出来ます。

```
# mysqlsh -u demo_user -ppassword --classic --schema=NEW57
mysqlx: [Warning] Using a password on the command line interface can be insecure.
Creating a Classic Session to demo_user@localhost:3306/NEW57
Default schema `NEW57` accessible through db.

# mysqlsh -u demo_user -ppassword --sql --schema=NEW57
mysqlx: [Warning] Using a password on the command line interface can be insecure.
Creating a Node Session to demo_user@localhost:33060/NEW57
Default schema `NEW57` accessible through db.
```

参照: http://dev.mysql.com/doc/refman/5.7/en/mysql-shell-features-command-line-arguments-list.html


ORACLE®