MYSQL DOCUMENT STORE

UNDER THE HOOD

Jan Kneschke, mailto:jan.kneschke@oracle.com

Architect, MySQL Middleware and Clients

MySQL Innovation Day, April 2016

WHAT TO EXPECT

- What is MySQL Document Store
- Foundations of the X Protocol
- How X Plugin maps CRUD to SQL

DOCUMENT STORE

A *Document* is

- typed
- structured as a tree
- no schema enforced

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

DOCUMENT STORE

Store provides

- fast retrieval
- optimizations for updates
- operations
  - Find, Update, Modify, Delete
  - Projections
  - Filters
  - Aggregation

MYSQL DOCUMENT STORE

- part of MySQL 5.7.12
- X DevAPI
  - MySQL Shell, Connector/J, C/.net and C/node.js
- X Protocol
  - implemented by X Plugin
- using stable interfaces from MySQL 5.7 GA
  - SQL service
  - JSON support

MYSQL 5.7'S JSON SUPPORT

MySQL 5.7.8 added:

- JSON datatype
- JSON functions like
  - `JSON_EXTRACT()`, `JSON_OBJECT()`, ...
- Generated Virtual/Stored Columns
- Function Indexes
  - based on JSON datatype and functions

MYSQL 5.7'S JSON SUPPORT

EXAMPLES

```
CREATE TABLE product (
  doc JSON,
  _id CHAR(16) AS (JSON_EXTRACT(doc, "$._id") STORED),
  PRIMARY KEY(_id));
```

```
INSERT INTO product (doc) VALUES (
  "{ '_id': 1,
    'name': 'Milk',
    'properties': {
      'volume_l': 1,
      'amount': 32
    }
  }");
```

```
SELECT JSON_OBJECT("amount",
         JSON_EXTRACT(doc, "$.properties.amount"))
  FROM product
 WHERE JSON_EXTRACT(doc, '$.name') == "Milk"
```

X DEVAPI

- working with Documents and Tables
- fluent API

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

A NEW PROTOCOL

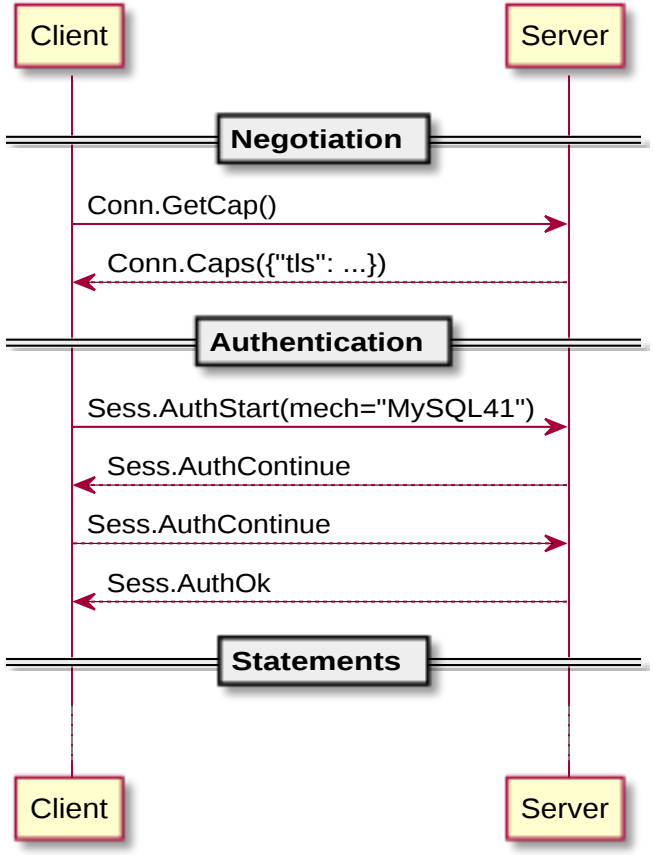
The common set of queries:

- CRUD == many small PK reads
- queries are independent
- plenty of data
  - sharding

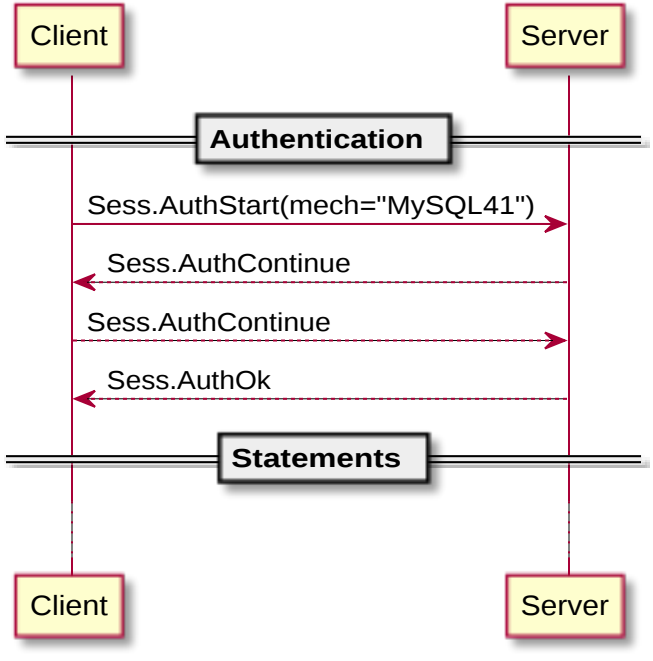
X PROTOCOL

- optimize for common operations
  - no manatory handshake
  - reduced message sizes
- pipelining
  - expectations
- notifications

X PROTOCOL


Client
Server
Negotiation
Conn.GetCap()
Conn.Caps({"tls": ...})
Authentication
Sess.AuthStart(mech="MySQL41")
Sess.AuthContinue
Sess.AuthContinue
Sess.AuthOk
Statements
Client
Server

X PROTOCOL


Client
Server
Authentication
Sess.AuthStart(mech="MySQL41")
Sess.AuthContinue
Sess.AuthContinue
Sess.AuthOk
Statements
Client
Server

REDUCED MESSAGE SIZES

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

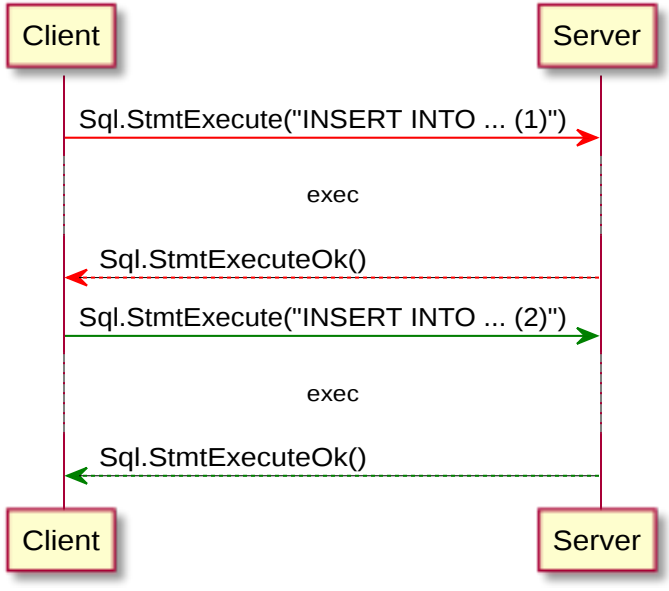
Note

- don’t send `orig_name` if it is equal to `name`
- don’t send catalog if it is “def”

QUERY TIME

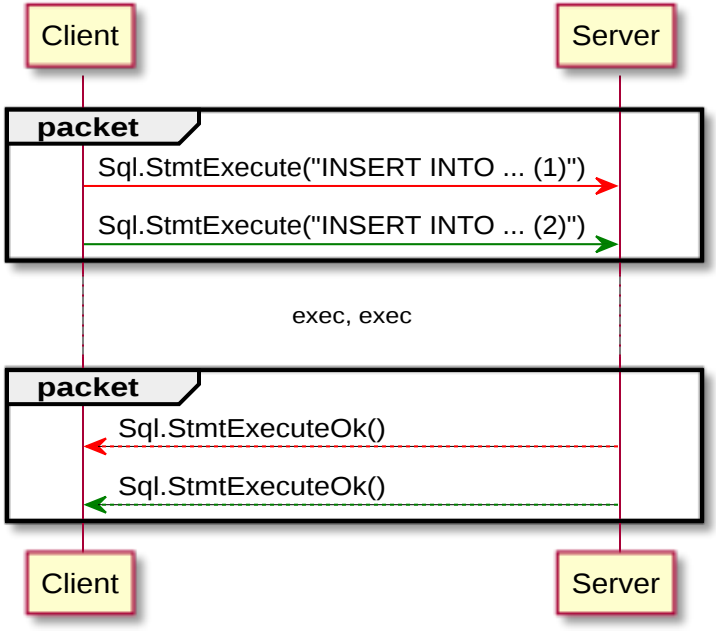
stage	time
network path latency	1ms
exectime	0.1ms

CLASSIC REQUEST/RESPONSE



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

PIPELINING



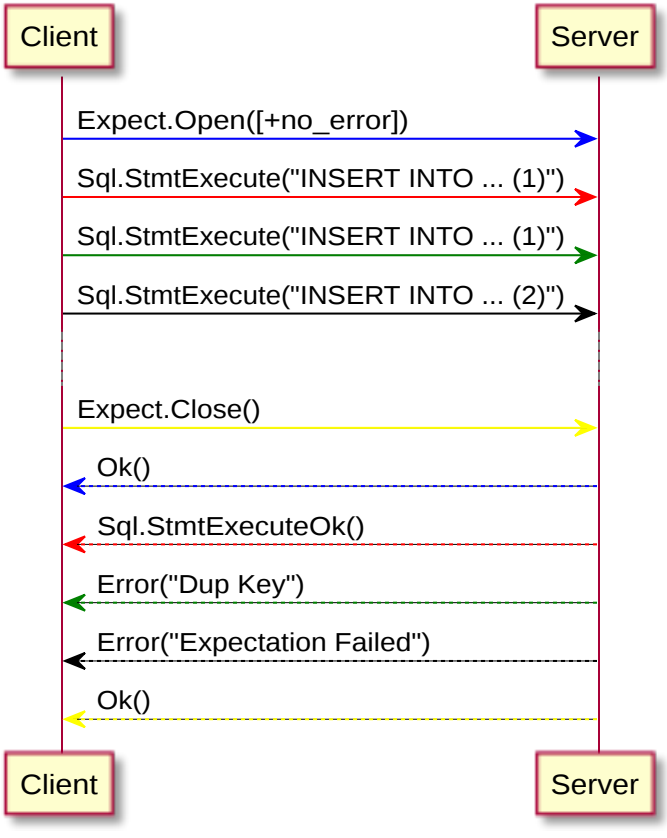
total: 2x path + 2x exectime = **2.2ms**

MANAGING EXPECTATIONS

- Pipelining makes error handling harder
  - Assumptions are usually wrong
  - Fail early? Fail late?
- Statement
  - Replication Lag?
  - Max Execution Time

MANAGING EXPECTATIONS

FAIL EARLY



MANAGING EXPECTATIONS III

NESTING

```
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()
```

MANAGING EXPECTATIONS IV

NESTING, PYTHON STYLE

```
try:
  prep = prepare(..)
  try:
    prep.execute([1])
    prep.execute([1])
  finally:
    prep.close()
except Exception:
  pass
```

NOTIFICATIONS

- server to client
- can appear at any time
- *local* events belong to current statement
  - warnings
  - status variable changes

NOTIFICATIONS

GLOBAL EVENTS

- async notification channel
- global events
  - server going down
  - connection timed out
  - membership changed: node left group
- foundation for message bus, pub/sub, ...

BUILDING BLOCKS

- `Protobuf` for serialization
- SASL methods for authentication
  - Simple Authentication Security Layer
  - `PLAIN` over TLS
  - `MYSQL41`
- TLS for encryption

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;
};
```

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

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

Q&A

Questions, please.