Introducing the MySQL Document Store

Mike Frank / マイク フランク
MySQL Global Business Unit
Product Management Director

ORACLE

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.

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Today's Agenda

1. Relational Databases, Document Databases and MySQL
2. MySQL JSON Support
3. Document Use Cases
4. The X DevAPI
5. Getting it all working together

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Relational Databases

- Data Integrity
  - Normalization
  - Constraints (foreign keys etc)
- Atomicity, Consistency, Isolation, Durability - ACID
  - Transactions
- SQL
  - Powerful, Optimizable Query Language
  - Declare what you want and the DB will find out the most efficient way to get it to you

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Plus…

- MySQL has been around since 1995
- Ubiquitous
- Pretty much a standard
- Scalable
- When there are issues, they are known and understood
- Large body of knowledge, from small to BIG deployments

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Document Databases

- Schemaless
  - no schema design, normalization, foreign keys, constraints, data types etc
  - faster initial development
- Flexible data structures
  - nested arrays and objects
  - some data is simply naturally unstructured or cannot be modeled efficiently in the relational model (hierarchies, product DB etc)
  - persist objects without ORMs

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Document Databases (Cont.)

- JSON
  - Closer to the frontend
  - "native" in JavaScript
  - Node.js and full stack JavaScript
- Easy to learn, easy to use

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Relational vs Document Databases

Why not both?

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

1 Relational Databases, Document Databases and MySQL
2 MySQL JSON Support
3 Document Use Cases
4 The X DevAPI
5 Getting it all working together

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

The New JSON Datatype

```
mysql> CREATE TABLE employees (data JSON);

mysql> INSERT INTO employees VALUES
 ('{"id": 1, "name": "Jane"}'),
 ('{"id": 2, "name": "Joe"}');

mysql> SELECT * FROM employees;
+-------------------------------------+
| data                                |
+-------------------------------------+
| {"id": 1, "name": "Jane"}           |
| {"id": 2, "name": "Joe"}            |
+-------------------------------------+
```

- Validation on INSERT
- No reparsing on SELECT
- Optimized for read
- Dictionary of sorted keys
- Can compare JSON/SQL
- Can convert JSON/SQL
- Supports all native JSON datatypes
- Also supports date, time, timestamp etc.

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL 5.7: JSON Support

- Native JSON datatype
- Store JSON values (objects, arrays and simple values) in MySQL tables
- Binary JSON storage format
- Conversion from "native" SQL types to and from JSON values
- JSON Manipulation functions
  - Extract contents (JSON_EXTRACT, JSON_KEYS etc)
  - Inspect contents (JSON_CONTAINS etc)
  - Modify contents (JSON_SET, JSON_INSERT, JSON_REMOVE etc)
  - Create arrays and objects (JSON_ARRAY, JSON_OBJECT)
  - Search objects (JSON_SEARCH)

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL 5.7: JSON Support (cont.)

- Inline SQL JSON path expressions

```
SELECT doc->'$.object.array[0].item' FROM some_table
```

- Boolean operators (compare JSON values etc)
  - foo = doc->'$.field'

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Generated Columns

```
CREATE TABLE order_lines
 (orderno integer,
  lineno integer,
  price decimal(10,2),
  qty   integer,
  sum_price decimal(10,2) GENERATED ALWAYS AS (qty * price) STORED );
```

- Column generated from the expression
- VIRTUAL: computed when read, not stored, indexable
- STORED: computed when inserted/updated, stored in SE, indexable
- Useful for:
  - Functional index
  - Materialized cache for complex conditions
  - Simplify query expression

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Functional Index

```
CREATE TABLE order_lines
 (orderno integer,
  lineno integer,
  price decimal(10,2),
  qty   integer,
  sum_price decimal(10,2) GENERATED ALWAYS AS (qty * price) VIRTUAL);

ALTER TABLE order_lines ADD INDEX  idx (sum_price);
```

- ***Online*** index creation
- Composite index on a mix of ordinary, virtual and stored columns

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Indexing JSON data

```
CREATE TABLE employees (data JSON);

ALTER TABLE employees
ADD COLUMN name VARCHAR(30) AS (JSON_UNQUOTE(data->"$.name"))
VIRTUAL,
ADD INDEX name_idx (name);
```

- Functional index approach
- Use inlined JSON path or JSON_EXTRACT to specify field to be indexed
- Support both VIRTUAL and STORED generated columns

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Generated column: STORED vs VIRTUAL

	Pros	Cons
STORED	• Fast retrieval	• Require table rebuild at creation • Update table data at INSERT/UPDATE • Require more storage space
VIRTUAL	• Metadata change only, instant • Faster INSERT/UPDATE, no change to table	• Compute when read, slower retrival

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Indexing Generated Column: STORED vs VIRTUAL

	Pros	Cons
STORED	• Primary & secondary index • B-TREE, Full text, R-TREE • Independent of SE • Online operation	• Duplication of data in base table and index
VIRTUAL	• Less storage • Online operation	• Secondary index only • B-TREE only • Require SE support

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

1 Relational Databases, Document Databases and MySQL
2 MySQL JSON Support
3 Document Use Cases
4 The X DevAPI
5 Getting it all working together

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Extracting JSON from a Relational DB

Relational In, Relational + Document Out

- Data stored in relational tables, but frontend uses JSON
- JSON directly maps to native data structures in many languages
  - Often easier for application code to use
  - JavaScript, Python, Ruby etc
  - In browser JavaScript

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Extracting JSON from a Relational DB

Relational In, Relational + Document Out

- SQL Functions to construct JSON
  - JSON_OBJECT(), JSON_ARRAY()
- Ex.:

```
SELECT JSON_OBJECT('cust_id', id, 'name', name, 'email', email) FROM customer;


CREATE VIEW customer_json AS
     SELECT JSON_OBJECT('cust_id', id, 'name', name, 'email', email) as doc
FROM customer;
SELECT * FROM customer_json;
```

- Updates and inserts still happen through the table columns

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Using MySQL as a JSON Document Container

Document In, Relational + Document Out

- Virtually Schemaless
  - Unstructured data
  - No clear, fixed structure for the data… records can have different fields
  - Often data that is not involved in *business rules*
  - Examples: "product_info", "properties", "options" etc
- Data does not map cleanly into a relational model (arrays, hierarchical data etc)
- Prototyping

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL as a JSON Document Container

Example: "properties" table

WIKIPEDIA

Users

user_properties
- up_user INT
- up_property VARBINARY(255)
- up_value BLOB

Indexes

page_props
- pp_page INT
- pp_propname VARBINARY(60)
- pp_value BLOB
- pp_sortkey FLOAT

Indexes

log_search
- ls_field VARBINARY(32)
- ls_value VARCHAR(255)
- ls_log_id INT

Indexes

https://www.mediawiki.org/wiki/Manual:Database_layout

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL as a JSON Document Container

Example: "product_info" table

product_id	attribute	value
9	size	M
9	color	red
9	fabric	cotton
11	flavour	strawberry
12	capacity	128GB
12	speed class	class 10
13	connectivity	Wi-Fi
13	storage	64GB
13	screen size	8.9"
13	resolution	2560 x 1600 (339 ppi)
13	battery life	12 hours

```
{
  "product_id": 9,
   "size" : "M",
   "color": "red",
   "fabric": "cotton"
},
{
  "product_id": 11,
  "flavour": "strawberry"
},
{
  "product_id": 12,
  "capacity": "128GB",
  "speed class": "class 10"
},
{
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL as a JSON Document Container

Document In, Relational + Document Out

- An ordinary MySQL table with a single JSON data column
- Generated columns allow SQL engine to look inside the JSON data
  - Virtual columns
  - Primary Keys
  - Indexes
  - Foreign Keys
- Writes on the JSON column
- Reads primarily from the JSON columns

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Hybrid Relational and JSON

Relational + Document In, Relational + Document Out

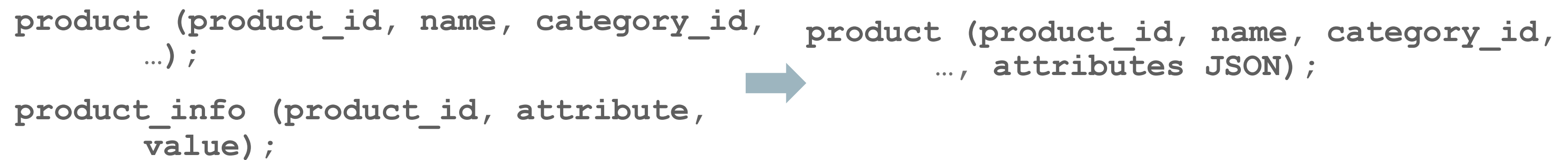
- Database is mostly relational
- Some parts of the database are unstructured or does not model cleanly as relational
- JSON columns in relational tables
- Queries can mix and match JSON and column data
- Evolution path for Document based applications

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Hybrid Relational and JSON

Relational + Document In, Relational + Document Out



```
product (product_id, name, category_id,
      …);

product_info (product_id, attribute,
      value);
```

```
product (product_id, name, category_id,
      …, attributes JSON);
```

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

1 Relational Databases, Document Databases and MySQL
2 MySQL JSON Support
3 Document Use Cases
4 The X DevAPI
5 Getting it all working together

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Document Operations via SQL

- Powerful
- Allows complex queries
- But… still difficult to use

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Document Operations via SQL

```
CREATE TABLE product (
  id VARCHAR(32) GENERATED ALWAYS AS (JSON_EXTRACT(doc, '$.id')) STORED,
  doc JSON
);

INSERT INTO product VALUES (1, '{…}');
SELECT * FROM product WHERE JSON_EXTRACT(doc, '$.field') = value;
etc.
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

The X DevAPI

- Abstraction over SQL
- Focused on 4 basic CRUD operations (Create, Read, Update, Delete)
- Fluent, Native Language API
- No knowledge of SQL needed
- X Protocol
  - CRUD requests encoded at protocol level
  - Request details "visible" (vs "opaque" SQL strings)

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Collection and Schema Operations

- Get a handle to a Schema

```
mydb = session.getSchema("mydb");
```

- Create a Collection

```
mydb.createCollection("products");
```

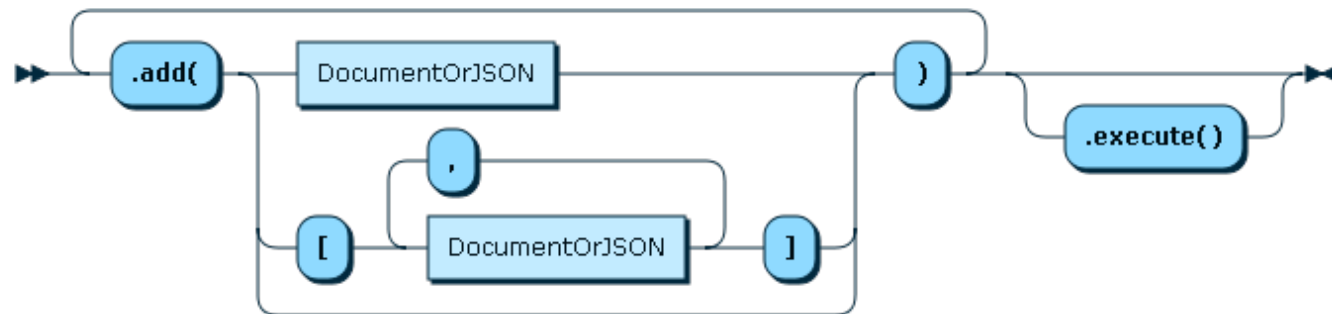
- Get a (local) reference to a Collection

```
products = mydb.getCollection("products");
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

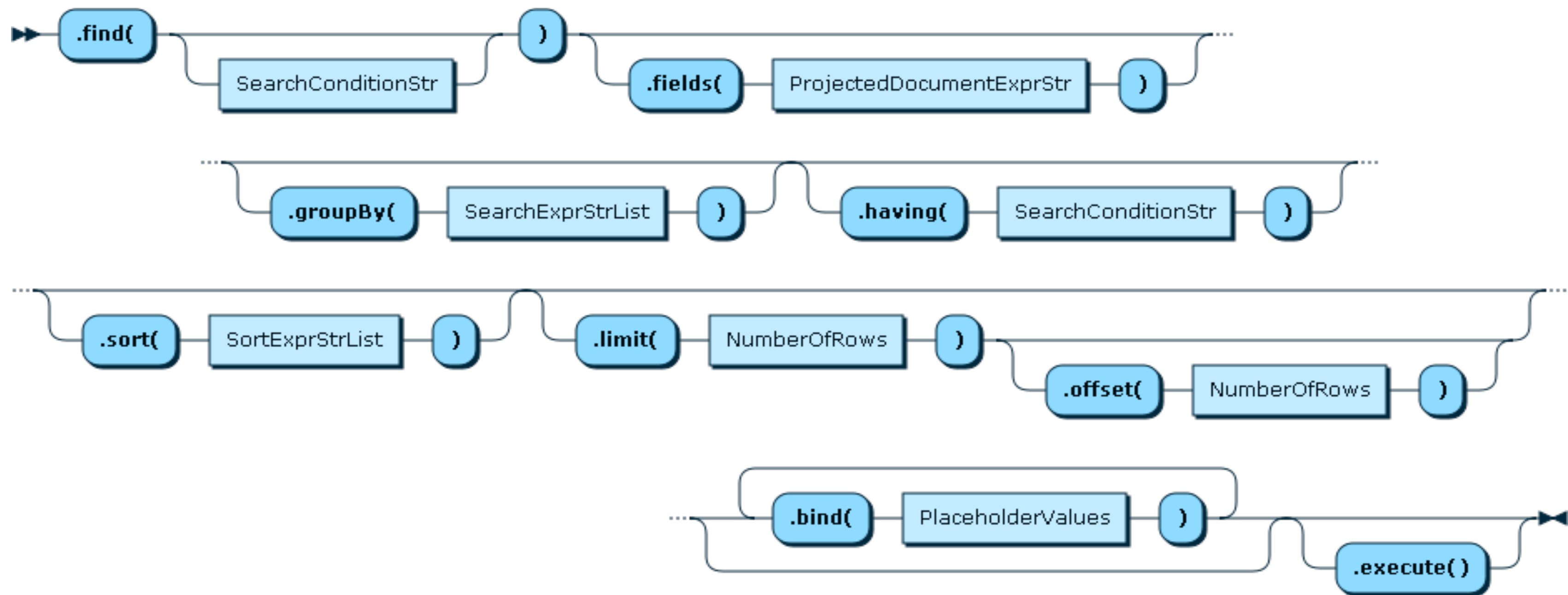
Add Document




```
products.add({"name":"bananas", "color":"yellow"}).execute();
```


Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Find Documents

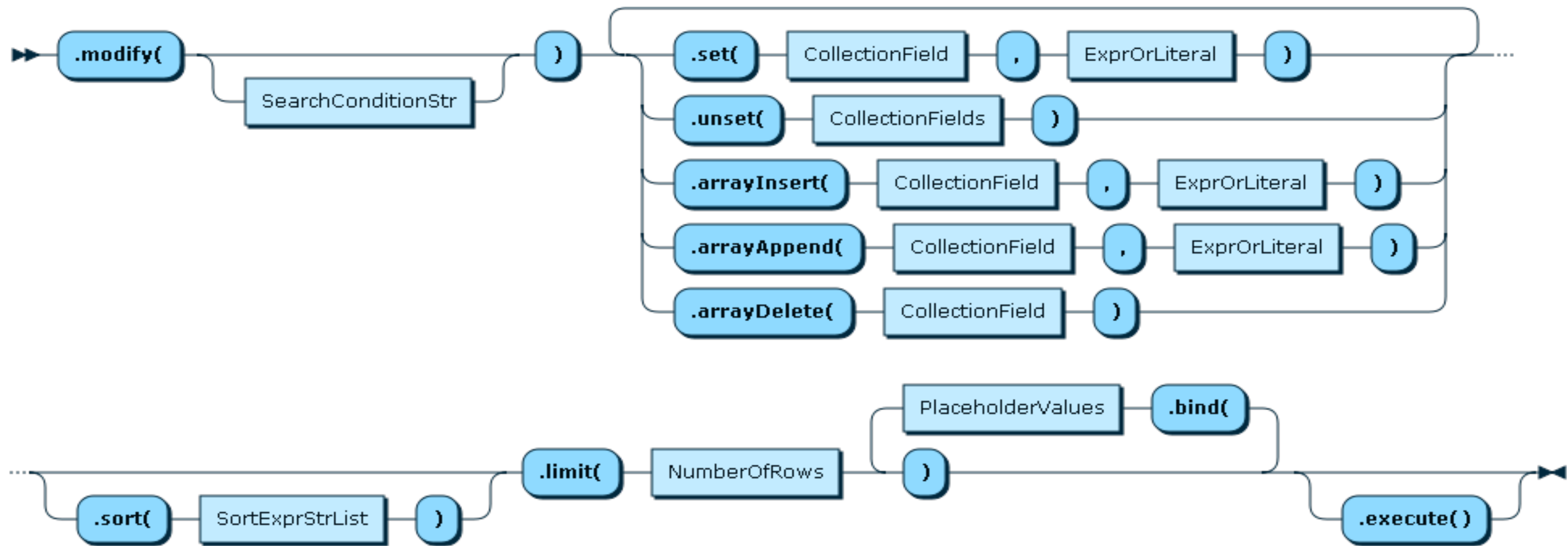



```
products.find("color = 'yellow'").sort(["name"]).execute();
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Modify Documents

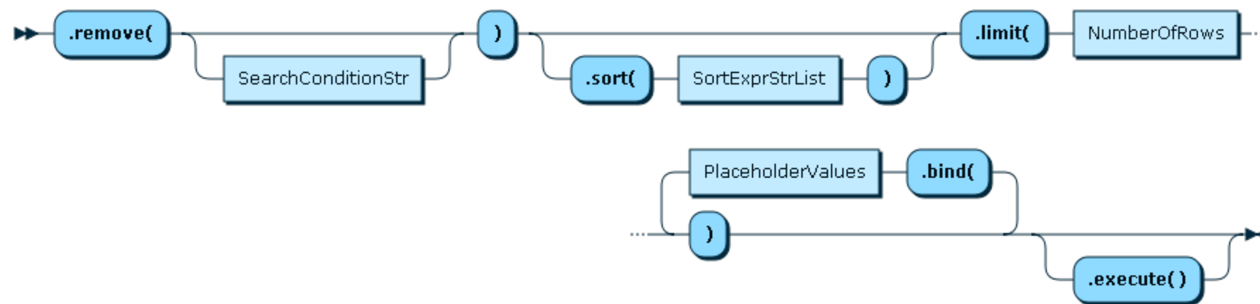



```
products.modify("product_id = 123").set("color", "red").execute();
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Remove Documents




```
products.remove("product_id = 123").execute();
```


Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

X DevAPI Sessions

- X Session
  - Stateless
  - CRUD only, no SQL
  - Abstracts the connection
- Node Session
  - Direct connection to a database node
  - Allows CRUD and SQL

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Other Operations on Collections

- Create an Index

```
db.post.createIndex("email").field("author.email", "text(30)", false)
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

CRUD Operations – NoSQL/Document and SQL/Relational

Operation	Document	Relational
Create	Collection.add()	Table.insert()
Read	Collection.find()	Table.select()
Update	Collection.modify()	Table.update()
Delete	Collection.remove()	Table.delete()

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

1 Relational Databases, Document Databases and MySQL
2 MySQL JSON Support
3 Document Use Cases
4 The X DevAPI
5 Getting it all working together

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

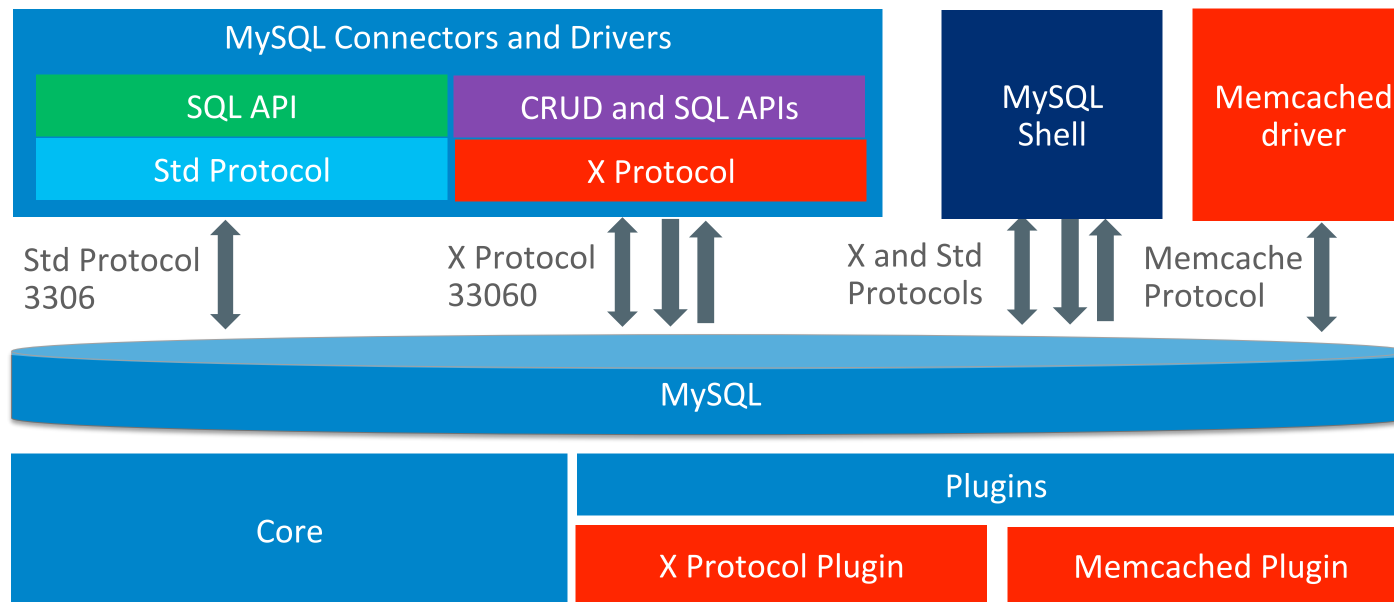
5.7.12 Development Preview Release

- MySQL 5.7.12 with Document Store plugin
- MySQL Shell 1.0.3
- Connector/J 7.0
- Connector/Net 7.0
- Connector/Node.js 1.0

ORACLE®

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL 5.7, Connectors, Drivers, and Protocols



ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

X DevAPI Connectors – MySQL Connector/Java 7.0

```
uri = "mysql:x://localhost:33060/test?user=user&password=mypwd"';

XSession session = new MysqlxSessionFactory().getSession(uri);

Schema schema = session.getDefaultSchema();

// document walkthrough

Collection coll = schema.createCollection("myBooks", true);

DbDoc newDoc = new DbDoc().add("isbn",
           new JsonString().setValue("12345"));

newDoc.add("title",
           new JsonString().setValue("Effi Briest"));

newDoc.add("author",
           new JsonString().setValue("Theodor Fontane"));

newDoc.add("currentlyReadingPage",
           new JsonNumber().setValue(String.valueOf(42)));

coll.add(newDoc).execute();

DocResult docs = coll.find("title = 'Effi Briest' and
currentlyReadingPage > 10").execute();
```

```
DbDoc book = docs.fetchOne();

System.err.println("Currently reading "
   + ((JsonString)book.get("title")).getString()
   + " on page "
   + ((JsonNumber)book.get("currentlyReadingPage")).getInteger());


// increment the page number and fetch it again

coll.modify("isbn = 12345").
  set("currentlyReadingPage",
    ((JsonNumber)book.get("currentlyReadingPage")).getInteger() +
1).execute();

docs = coll.find("title = 'Effi Briest' and currentlyReadingPage >
10").execute();

book = docs.fetchOne();

System.err.println("Currently reading "
  + ((JsonString)book.get("title")).getString()
  + " on page "
  + ((JsonNumber)book.get("currentlyReadingPage")).getInteger());
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

X DevAPI Connectors – MySQL Connector/Net 7.0

```
using (XSession session =
          MySQLX.GetSession("mysqlx://test:test@localhost:33060"))
{
  string schemaName = "test";
  Schema testSchema = session.GetSchema(schemaName);
  if (testSchema.ExistsInDatabase())
    session.DropSchema(schemaName);
  session.CreateSchema(schemaName);

  // insert some docs
  Collection coll = testSchema.CreateCollection("myDocs");
  var docs = new[]
  {
    new { _id = 1, title = "Book 1", pages = 20 },
    new { _id = 2, title = "Book 2", pages = 30 },
    new { _id = 3, title = "Book 3", pages = 40 },
    new { _id = 4, title = "Book 4", pages = 50 },
  };
```

```
  Result r = coll.Add(docs).Execute();
  Console.WriteLine("Docs added: " + r.RecordsAffected);

  // modify some values
  r = coll.Modify("_id = :ID").
       Bind("Id", 2).Set("pages", "25").Execute();
  Console.WriteLine("Docs modified: " + r.RecordsAffected);

  // remove a book
  r = coll.Remove("_id = :ID").Bind("Id", 4).Execute();
  Console.WriteLine("Docs removed: " + r.RecordsAffected);

  // list the results
  var result30orMore = coll.Find("pages > 20").
       OrderBy("pages DESC").Execute().FetchAll();
  foreach(var doc in result30orMore)
  {
    Console.WriteLine(doc.ToString());
  }
}
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

X DevAPI Connectors – MySQL Connector/Node.js 1.0 NEW!

```
const mysqlx = require('mysqlx');

mysqlx.getSession({
     host: 'localhost',
     dbUser: 'myuser',
     dbPassword: 'secret'
}).then(session => {
     const collection =
session.getSchema('myschema').getCollection('mycollection');
    return Promise.all([
       collection.add({ foo: "bar", something: { nested:
[1,2,3,4] } }).execute();
       session.close();
    ])
}).catch(err => {
            console.log(err);
});
```

```
const collection =
session.getSchema('myschema').getCollection('questions');
collection.find("answer == 42")
    .orderBy("foo DESC")
    .limit(10)
    .execute(doc => console.log(doc)) // print the document
received, callback called for each doc
    .then(() => console.log("All done") // Promise resolves
when all data is there
      .catch((err) => console.log("Oups, an error", err);
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL Shell

```
Copyright (c) 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type '\help', '\h' or '\?' for help.

Currently in JavaScript mode. Use \sql to switch to SQL mode and execute queries.
mysql-js> db.getCollections()
{
    "CountryInfo": <Collection:CountryInfo>
}
mysql-js> db.CountryInfo.find().limit(1)
[
    {
        "GNP": 828,
        "IndepYear": null,
        "Name": "Aruba",
        "_id": "ABW",
        "demographics": {
            "LifeExpectancy": 78.4000015258789,
            "Population": 103000
        },
        "geography": {
            "Continent": "North America",
            "Region": "Caribbean",
            "SurfaceArea": 193
        },
        "government": {
            "GovernmentForm": "Nonmetropolitan Territory of The Netherlands",
```

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

MySQL Plugin for VisualStudio

File Edit View Project Debug Team Tools Architecture Test Analyze Window Help

Server Explorer

- Azure (Not connected)
- Data Connections
  - localhost(test)
- Servers
- SharePoint Connections

MySQL Script.myjs*

Server: 5.7.12-log | User: test | Database: test | Session Option

```
var collection = session.sakila_x.getCollection('movies');
collection.find('rating = "R"').execute();
```

Result1

Row 3	8 Fields	Object
Row 4	8 Fields	Object
Row 5	8 Fields	Object
_id	0f3dca31c7724516873ebc226b0be05d	String
actors	System.Collections.Generic.List`1[Syst...	Collections.Generic.List`1[Object]
description	A Lackluster Display of a Dentist And ...	String
duration	92 min	String
language	English	String
rating	R	String
release_year	2006	String
title	ANACONDA CONFESSIONS	String
Row 6	8 Fields	Object
Row 7	8 Fields	Object
Row 8	8 Fields	Object
_id	8fcff3a58bda4015bf08ffe923489f10	String
actors	System.Collections.Generic.List`1[Syst...	Collections.Generic.List`1[Object]
description	A Astounding Story of a Dog And a Sq...	String
duration	119 min	String
language	English	String
rating	R	String
release_year	2006	String
title	APOCALYPSE FLAMINGOS	String

Grid View / Tree View / Text View

Output

Show output from: MySQL

```
[Information] - undefined
```

Solution Explorer

Solution 'Solution1' (0 projects)

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Resources

Topic	Link(s)
MySQL as a Document Database	http://dev.mysql.com/doc/refman/5.7/en/document-database.html
MySQL Shell	http://dev.mysql.com/doc/refman/5.7/en/mysql-shell.html http://dev.mysql.com/doc/refman/5.7/en/mysqlx-shell-tutorial-javascript.html http://dev.mysql.com/doc/refman/5.7/en/mysqlx-shell-tutorial-python.html
X Dev API	http://dev.mysql.com/doc/x-devapi-userguide/en/
X Plugin	http://dev.mysql.com/doc/refman/5.7/en/x-plugin.html
MySQL JSON	http://mysqlserverteam.com/tag/json/ https://dev.mysql.com/doc/refman/5.7/en/json.html https://dev.mysql.com/doc/refman/5.7/en/json-functions.html
Blogs	http://mysqlserverteam.com/category/docstore/

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

Thank You!

ORACLE

Copyright © 2016 Oracle and/or its affiliates. All rights reserved. |

ORACLE®