DevConf www.devconf.ru

Vittorio Cioe
MySQL Sr. Sales Consultant
vittorio.cioe@oracle.com

MySQL

NoSQL + SQL = MySQL Document Store

Agenda

- Introduction
- SQL or NoSQL, or… MySQL Document Store?
- Document Store: Under the hood
- Conclusion

Introduction

Who Am I?

- Used MySQL since 2006
- Working at Oracle/MySQL since 2017, covering Eastern Europe
- Previously working in the Security and Digital Transformation (API) space

- From Italy but based in Warsaw
- Love movies, travelling, cooking...


#1

MySQL is the #1 database for the web, used by 10 of the top 10 websites


MySQL Facts & Figures

#1 Most popular open source relational database*

*according to db-engines.com

#2 Most popular database behind only Oracle DB*

*according to db-engines.com

70% By 2022, more than 70% of new in-house applications will be developed on an Open Source RDBMS

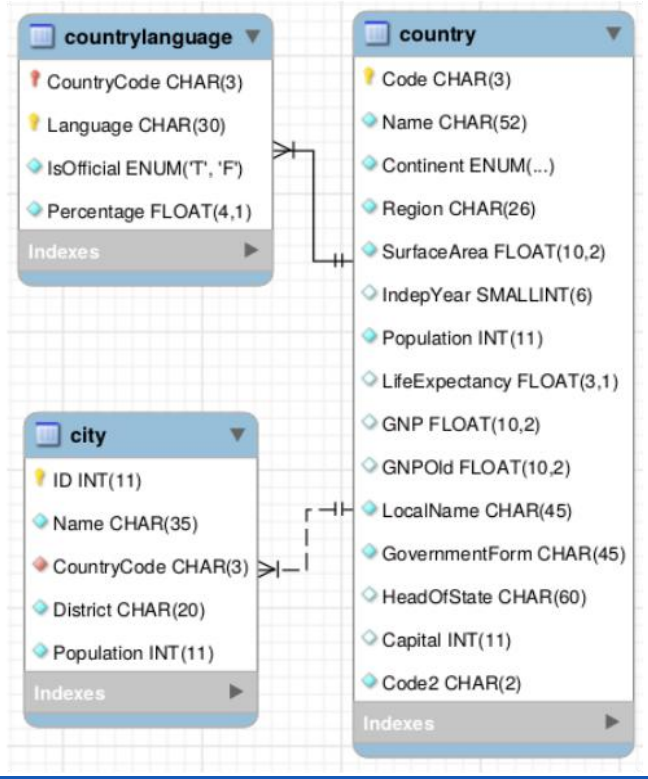
Source: Gartner, State of Relational Open Source RDBMSs 2018

50% By 2022, 50% of commercial RDBMS instances will have been converted to open source DBs

Source: Gartner, State of Relational Open Source RDBMSs 2018

SQL, NoSQL, or... MySQL Document Store?

How DBAs see data



How Developers see data

MySQL


```
{
   "GNP": 249704,
   "Name": "Belgium", "government" : {
      "GovernmentForm" :
         "Constitutional Monarchy, Federation", "HeadOfState" :
      "Philippe I"
      },
      "_id" : "BEL",
      "IndepYear": 1830, "demographics" : {
         "Population" : 10239000,
         "LifeExpectancy" : 77.8000030517578
      },
      "geography" : {
         "Region": "Western Europe", "SurfaceArea" : 30518,
      "Continent" : "Europe"
      }
   }
```

MySQL

DMBS or NoSQL ?

Tables or Collections?

- A collection can be viewed as a table with 2+ columns:
  - Primary key: `_id`
  - JSON document: `doc`
    - The document's `_id` field could be supplied or automatically generated as UUID
      - This field could be also used to populate the primary key
- Can add extra columns and indexes to a collection
- SQL, NoSQL, tables, collections, all can be used simultaneously
- Operations compatible with replication

JSON support is key to enable this choice

Native JSON Data Type

(Introduced in MySQL 5.7)

```
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)
```

JSON Data Type: what's good to know

- Binary Data type.
- Built-in JSON Document validation
- Can be searched using JSON Path (JSON_EXTRACT(column_name-> "$.type"))
- You can index (stored or virtual) specific JSON fields using Generated Columns
- Wide range of JSON functions, and increased in MySQL 8.0
- Quick document import with "util.importJSON(file, options)"
- Faster than TEXT data type.
  (Why? There is a table of pointers to the keys and values at the beginning of the binary representation, so you can look up a member in the middle of the document directly without scanning past all the members.)

DBMS or NoSQL ?

Why not both ?

MySQL

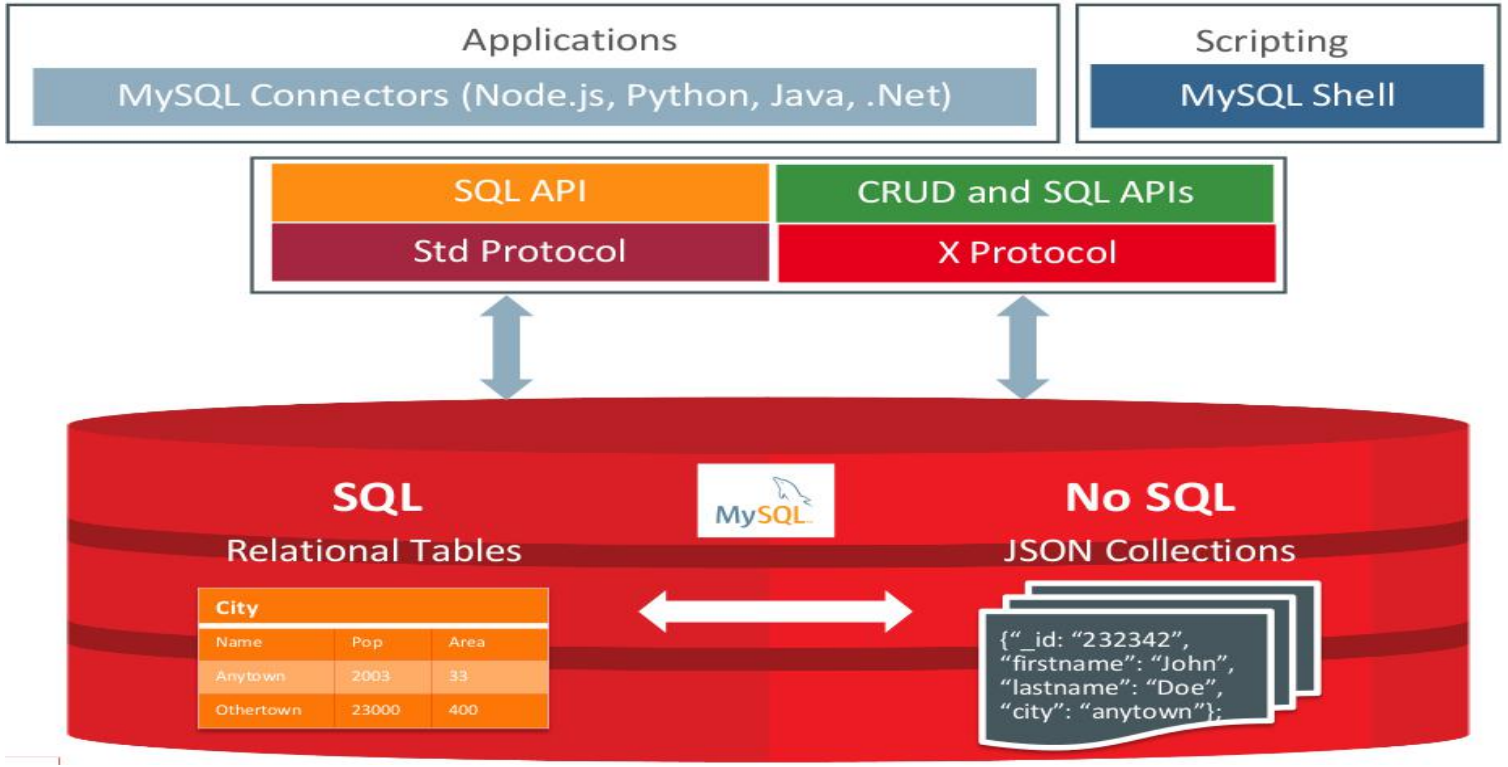
The MySQL Document Store !

The best of both worlds in one solution

MySQL Document Store

Document Store: Under the hood...

MySQL Document Store: Architecture



MySQL Document Store is Full ACID

ACID transactions = Don’t lose my data

What is a collection? it is an InnoDB table which has 2 columns, an id and a JSON object/document which contains the id itself, and it is accessed in a NoSQL fashion via the X-Dev API.

Since it relies on the proven MySQL InnoDB it brings strength, robustness & **full transactionality (including rollbacks!)**:

- **innodb_flush_log_at_trx_commit = 1**
- **innodb_doublewrite = ON**
- **sync_binlog = 1**
- **transaction_isolation = REPEATABLE-READ | READ-COMMITTED | ...**

We do care about your data!

X DevAPI

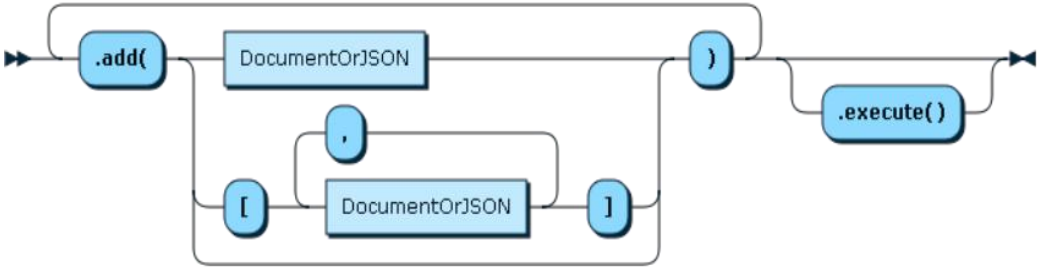
Use SQL, CRUD APIs – Document (NoSQL) and Relational (SQL), or "All of the Above"

All of this is in addition to the Classic APIs

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

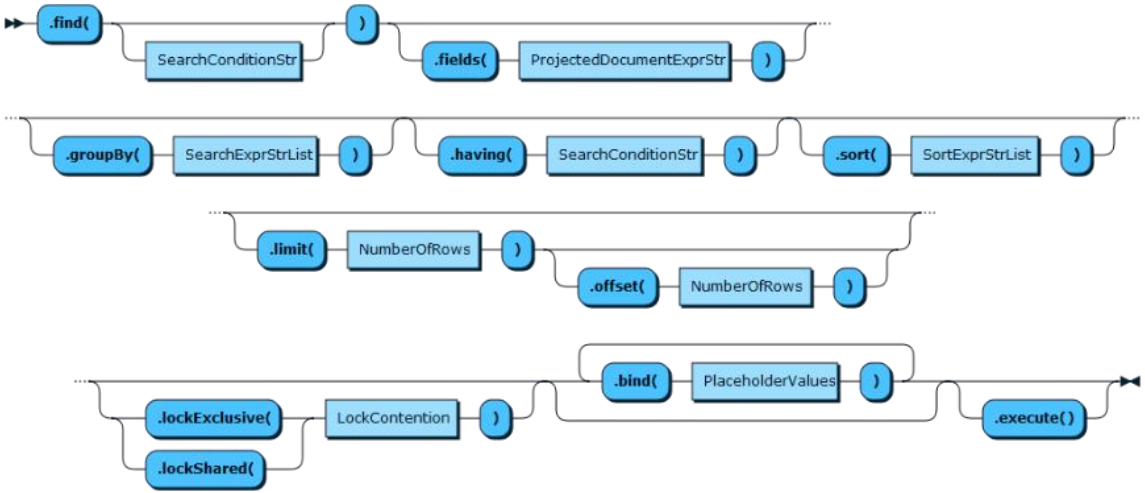
- Implemented in connectors for
  - C++, Java, .Net, Node.js, Python, PHP
  - working with Communities
- New MySQL Shell
  - Command Completion
  - Python, JavaScript & SQL modes
  - Admin functions
  - New Util object
  - A new high-level session concept that can scale from single MySQL Server to a multiple server environment
  - Non-blocking, asynchronous calls follow common language patterns
  - Supports CRUD operations

CRUD operations: CollectionAdd



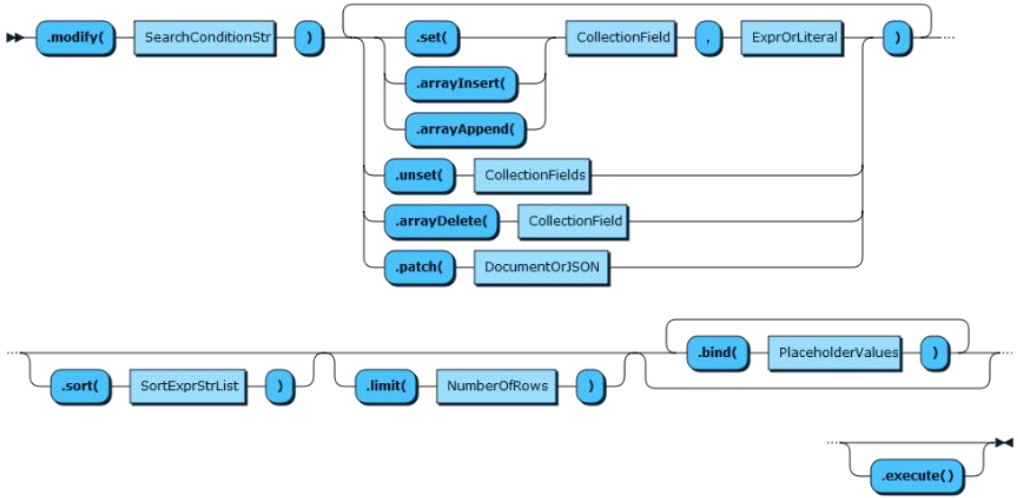

```
collection.add({ name: 'foo', age:
42 })
  .add({ name: 'bar', age: 23 })
  .execute()

collection.add([
  { name: 'baz', age: 50 },
  { name: 'qux', age: 25 }
]).execute()
```

CRUD operations: CollectionFind



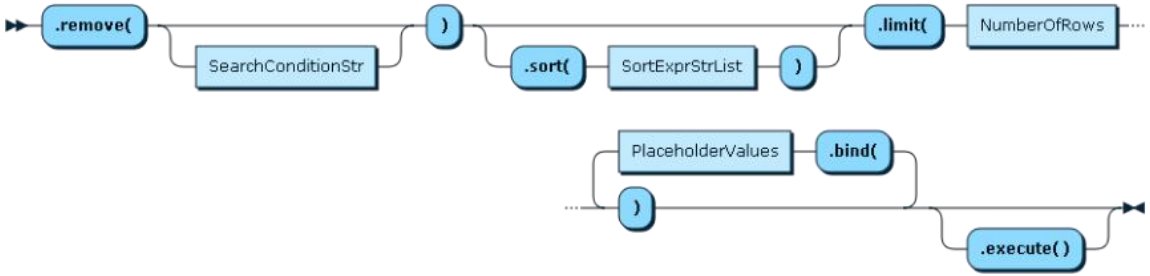

```
collection.find('name = :name')
      .bind('name', 'foo')
      .fields('COUNT(age) AS
      age')
      .groupBy('age')
      .having('age > 42')
      .sort('age DESC')
      .limit(10)
      .offset(5)
      .execute()
```

CRUD operations: CollectionModify



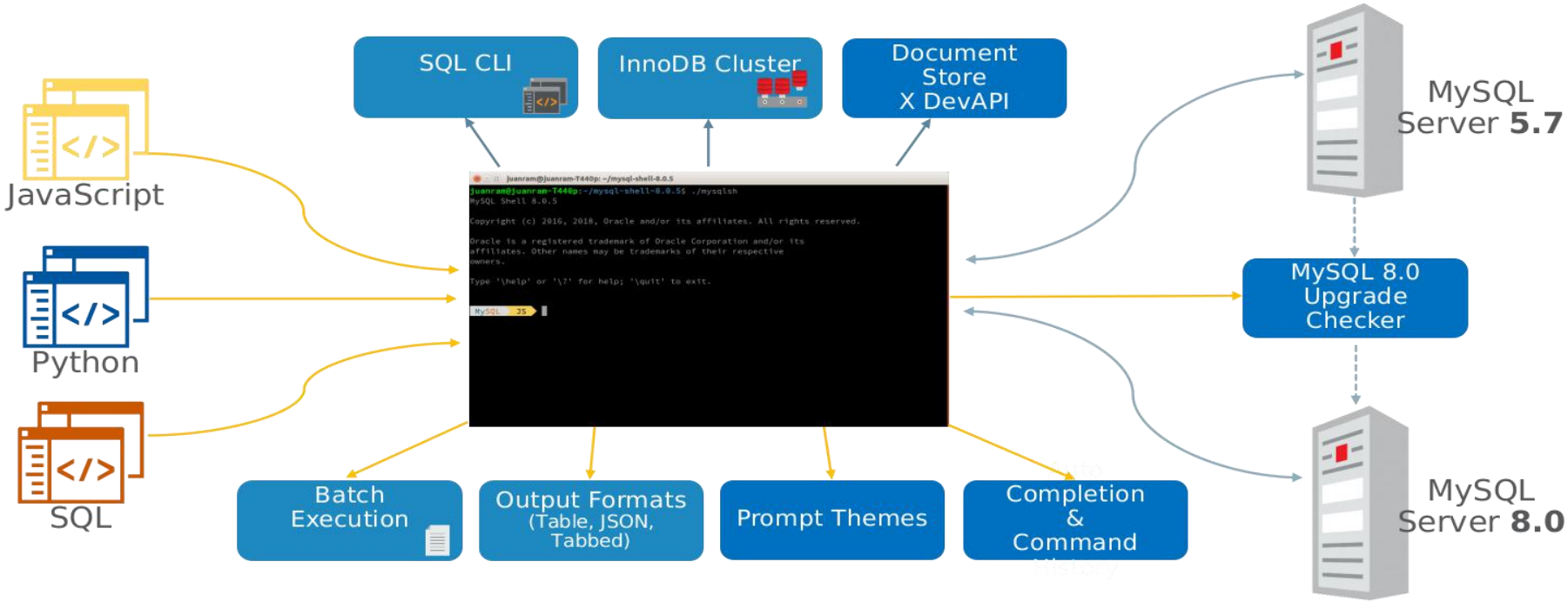

```
collection.modify('name
= :name')
  .bind('name', 'foo')
  .set('age', 42)
  .sort('name ASC')
  .limit(1)
  .execute()

collection.modify('name
= :name')
  .bind('name', 'bar')
  .patch({ age: 42, active:
  false })
  .sort('name DESC')
  .limit(1)
  .execute()
```

CRUD operations: CollectionRemove




```
collection.remove('name
= :name')
  .bind('name', 'foo')
  .sort('age ASC')
  .limit(1)
  .execute()
```

How to test and administer?
MySQL Shell 8.0.12+



MySQL Document Store cheat sheet

```
js> session.createSchema('name')
js> \use name
js> db.getCollections()
js> db.createCollection('myCollection')
js> db.getCollections()
js> db.myCollection.add({"param1":"value1", "param2":"value2"})
```

Create

```
js> db.myCollection.find()
js> db.myCollection.find().limit(1)
js> db.myCollection.find("_id = '00005af0184300000000000000002'")
```

Read

```
js> db.myCollection.modify("_id = '1234'").set("param","value")
```

Update

```
js> db.myCollection.remove("_id = '1234'")
```

Delete

```
js> session.startTransaction()
js> ...
js> session.rollback()
```

Document Store Example: Import Data

- Creating Collection
  db.createCollection('restaurants')
- Add single entry
  db.restaurants.add({ "_id" : "5ad5b645f88c5bb8fe3fd337", "address" : { "building" : "1007", "coord" : [-73.856077, 40.848447], "street" : "Morris Park Ave", "zipcode" : "10462" }, "borough" : "Bronx", "cuisine" : "Bakery", "grades" : [{ "date" : "2014-03-03T00:00:00Z", "grade" : "A", "score" : 2 }, { "date" : "2013-09-11T00:00:00Z", "grade" : "A", "score" : 6 }, { "date" : "2013-01-24T00:00:00Z", "grade" : "A", "score" : 10 }, { "date" : "2011-11-23T00:00:00Z", "grade" : "A", "score" : 9 }, { "date" : "2011-03-10T00:00:00Z", "grade" : "B", "score" : 14 }], "name" : "Morris Park Bake Shop", "restaurant_id" : "30075445" })
- Import a JSON bulk
  util.import_json("/tmp/products.json", {"schema": "mydb", "collection": "restaurants"})

Document Store Example: Transactions

- Initiate Transaction
  session.startTransaction()
  session.rollback()
- Do the work (even on multiple collections...)
  db.restaurants.add(
  db.restaurants.remove("_id='5ad5b645f88c5bb8fe3fd337'")
  db.restaurants.find("_id='5ad5b645f88c5bb8fe3fd337'")
  db.posts.modify("_id = '00005af018430000000000000002'").se("title","Hybrid database")
  db.posts.find("_id = '00005af018430000000000000002'")
- Confirm (or not...)
  session.commit()
  session.rollback()

Document Store Example: Indexes

- Create and index on a JSON field
  db.restaurants.createIndex('myIndex', {fields: [{field: "$.cuisine", type: "TEXT(20)"}]})
- Check how the create table changed
  \sql
  use mydb;
  show create table restaurants;
- Check how the explain plan changed
  EXPLAIN SELECT doc->>"$.name" AS name, doc->>"$.cuisine" AS cuisine FROM restaurants
  WHERE doc->>"$.cuisine" = 'Italian' \G

Document Store Example: NoSQL + SQL

- Use NoSQL for easy key-value data retrieving...

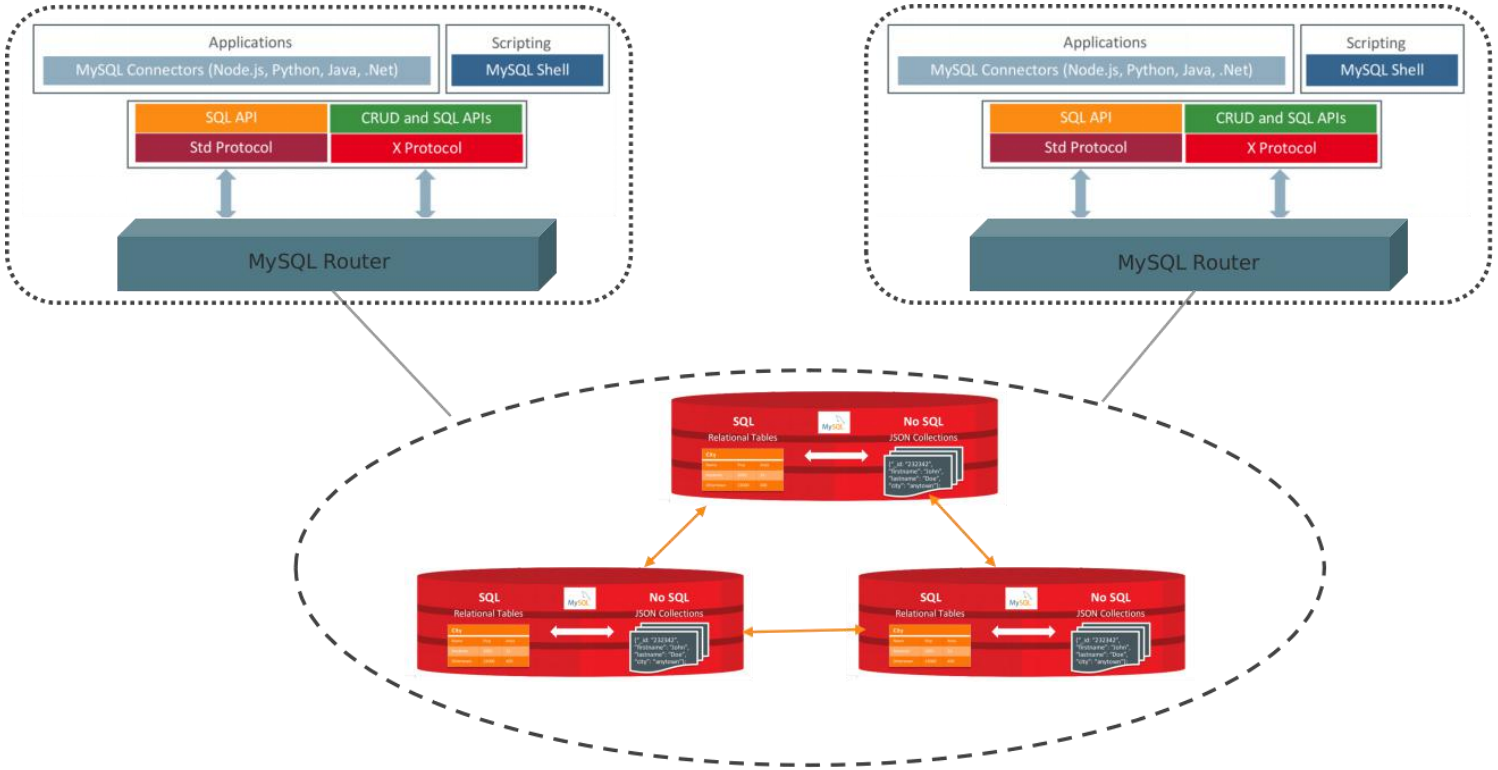
```
db.restaurants.find("_id='5ad5b645f88c5bb8fe3fd337'")
```

- ...and SQL for more complex aggregations

```
WITH cte1 AS (SELECT doc->>"$.name" AS name, doc->>"$.cuisine" AS cuisine,
(SELECT AVG(score) FROM json_table(doc, "$.grades[*]" COLUMNS (score INT PATH "$.score")) AS r) AS avg_score FROM restaurants) SELECT *,
RANK() OVER (PARTITION BY cuisine ORDER BY avg_score) AS `rank` FROM cte1 ORDER BY `rank`,
avg_score DESC LIMIT 30;
```

...on the same data set!!

MySQL DocStore: HA with InnoDB Cluster

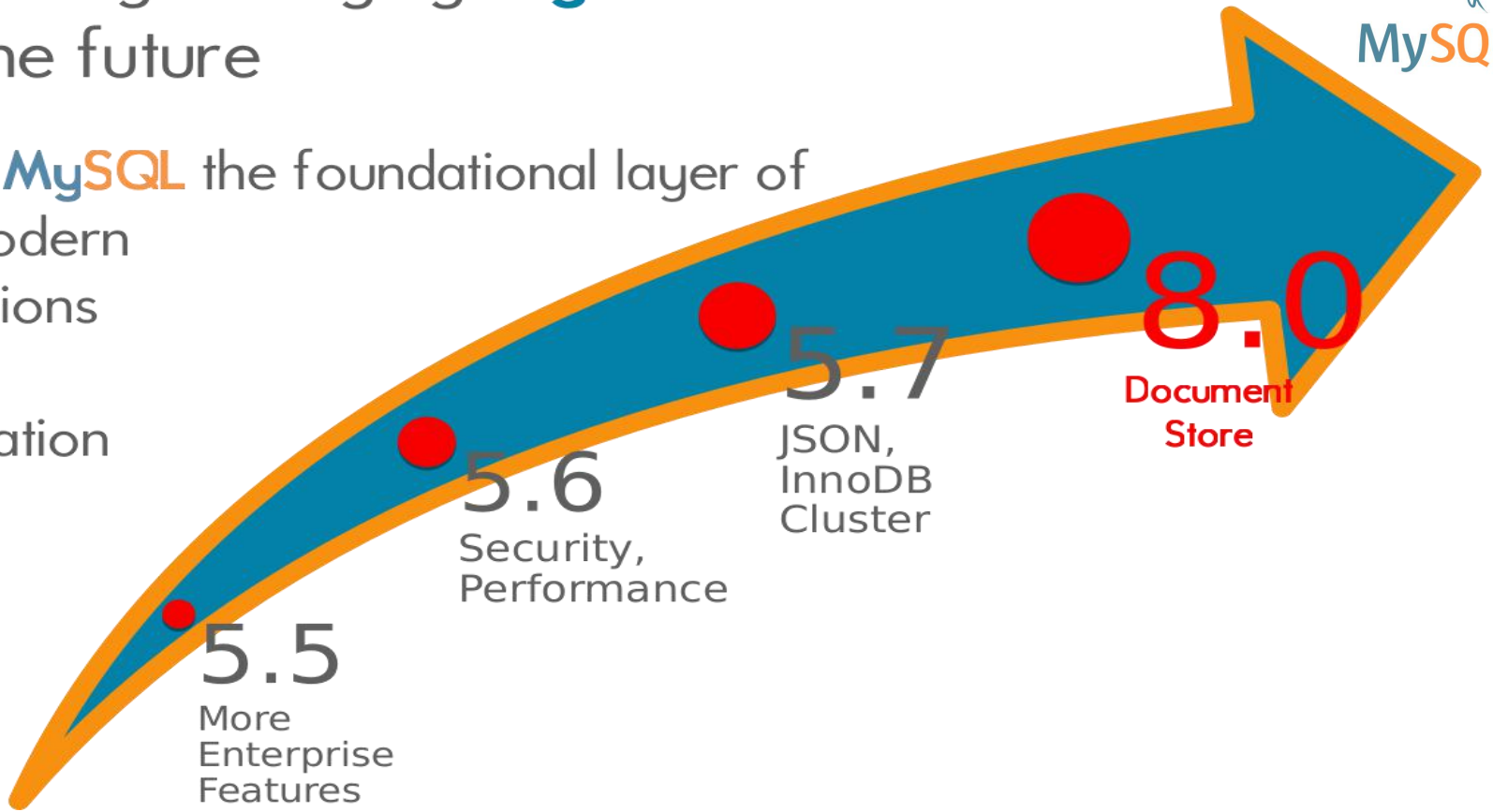


Conclusion


HEWLETT
PACKARD
48GX
8.0
MySQL
128K RAM
NOSQL
+ SQL
= MYSQL

Takeaways: bringing MySQL into the future

Making MySQL the foundational layer of your modern applications in your organization




MySQL
THANK YOU!!