Cross the border!! SQL + NoSQL = MySQL

MySQL Document Store

Revathi Rangachari
Technical Account Manager
revathi.rangachari@oracle.com

ORACLE

MySQL

Safe Harbor Statement

ORACLE®

Program Agenda

Introduction

Document Oriented Databases

What is a Document?

- A data structure that can represent complex information, similar to an Object
- Structure of the data is part of the document, no uniform structure
- **JSON** (=JavaScript Object Notation)
  - Compact, popular and standardized
  - Can be represented natively in many languages (JavaScript, Python etc)
- Other popular encoding formats are XML, YAML etc

JSON Document Example

```
{
      "_id": "IND",
      "Name": "India",
      "GNP": 211860,
      "IndepYear": 1947,
      "demographics": {
         "LifeExpectancy": 77.699,
         "Population": 1013662000
      },
      "geography": {
         "Continent": "Asia",
         "Region": "South & Central Asia",
         "SurfaceArea": 83859
      }
}
```

ORACLE

Document Oriented Databases

What is a Document in MySQL?

PDF
Adobe
W
X

Document Oriented Databases

Usability & Scalability

- **Schemaless**: No centralized database schema
  - Data model enforcement and validation (if any) at application layer
  - Simpler schema updates (no ALTER TABLE penalty)
- **NoSQL APIs**: Simpler programming interfaces
  - No specialized language for queries and data manipulation
  - Complex queries handled at application layer (no complex SELECTs, JOINs)
  - Document in, document out, manipulations at client side
- **Scalability**, but some drawbacks:
  - Limited database features (no foreign keys, no transactions, etc.)
  - Weak consistency guarantees

Why not…

- Have **both schema-less and schema** in the same technology stack?
- One that checks all the boxes of all stakeholders:

Developers:
[x] Schemaless or/and Schema
[x] Rapid Prototyping/Simpler APIs
[x] Document Model
[x] Transactions

Operations:
[x] Performance Management/Visibility
[x] Robust Replication, Backup, Restore
[x] Comprehensive Tooling Ecosystem
[x] Simpler application schema upgrades

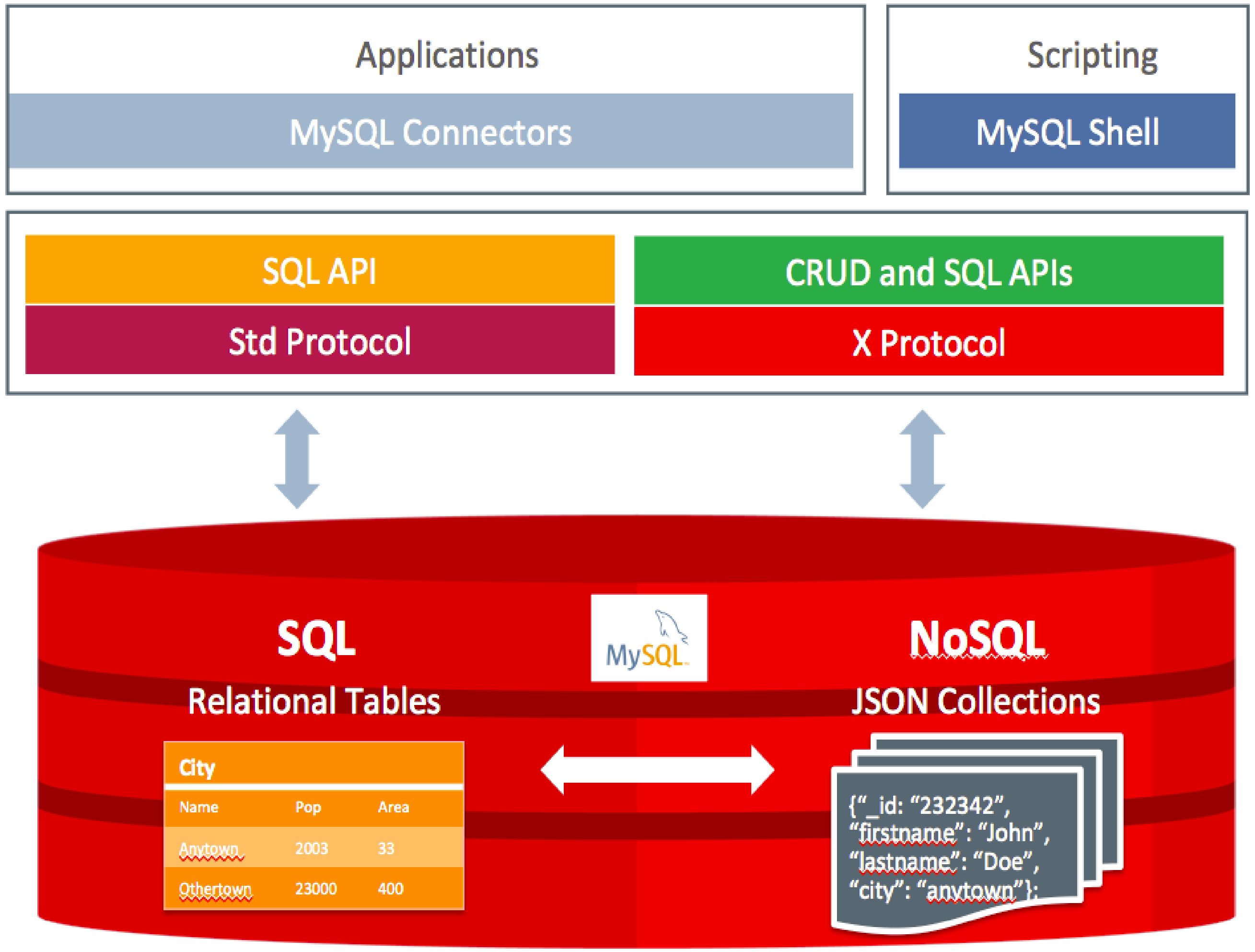
Business Owner:
[x] Don’t lose my data = ACID transactions
[x] Capture all my data = Extensible/Schemaless
[x] Products On Schedule/Time to Market = Rapid Development

MySQL Document Store

What is the MySQL Document Store?

"An easy, straight forward way to work with JSON documents in MySQL"

MySQL 8.0: Document Store Architecture



Scaling the Document Store

Scaling MySQL – What is available today?

- Vertical Scaling (scaling a single machine instance) – Available Today
  - Big improvements in MySQL 5.7 and 8.0
  - 1M QPS
  - Multi-TB databases
- Read Scale-Out – Available Today
  - Already solved since more than 10 years
  - Big companies run hundreds or thousands of async read slaves

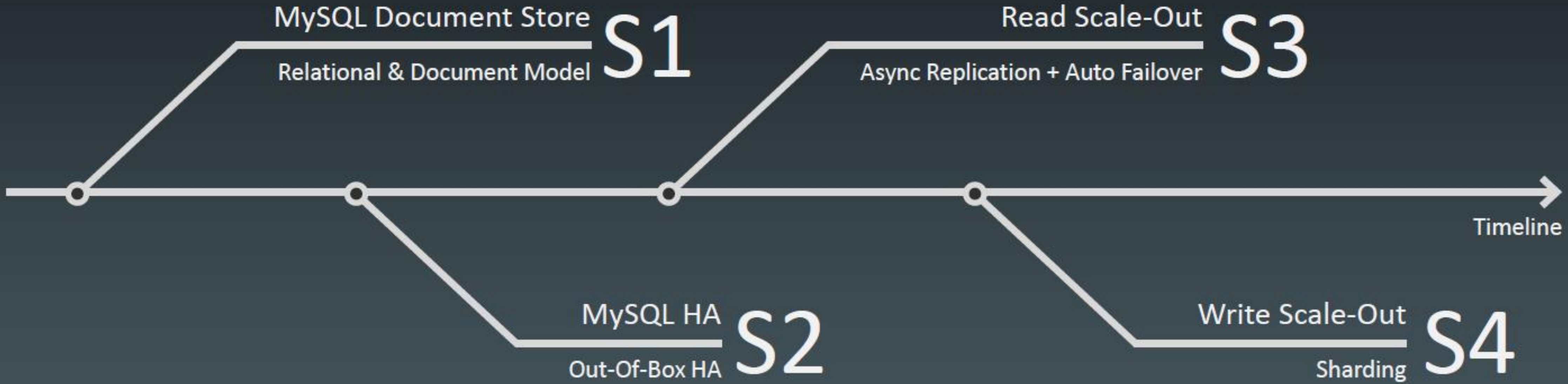
ORACLE

Scaling MySQL – Write Scale-Out

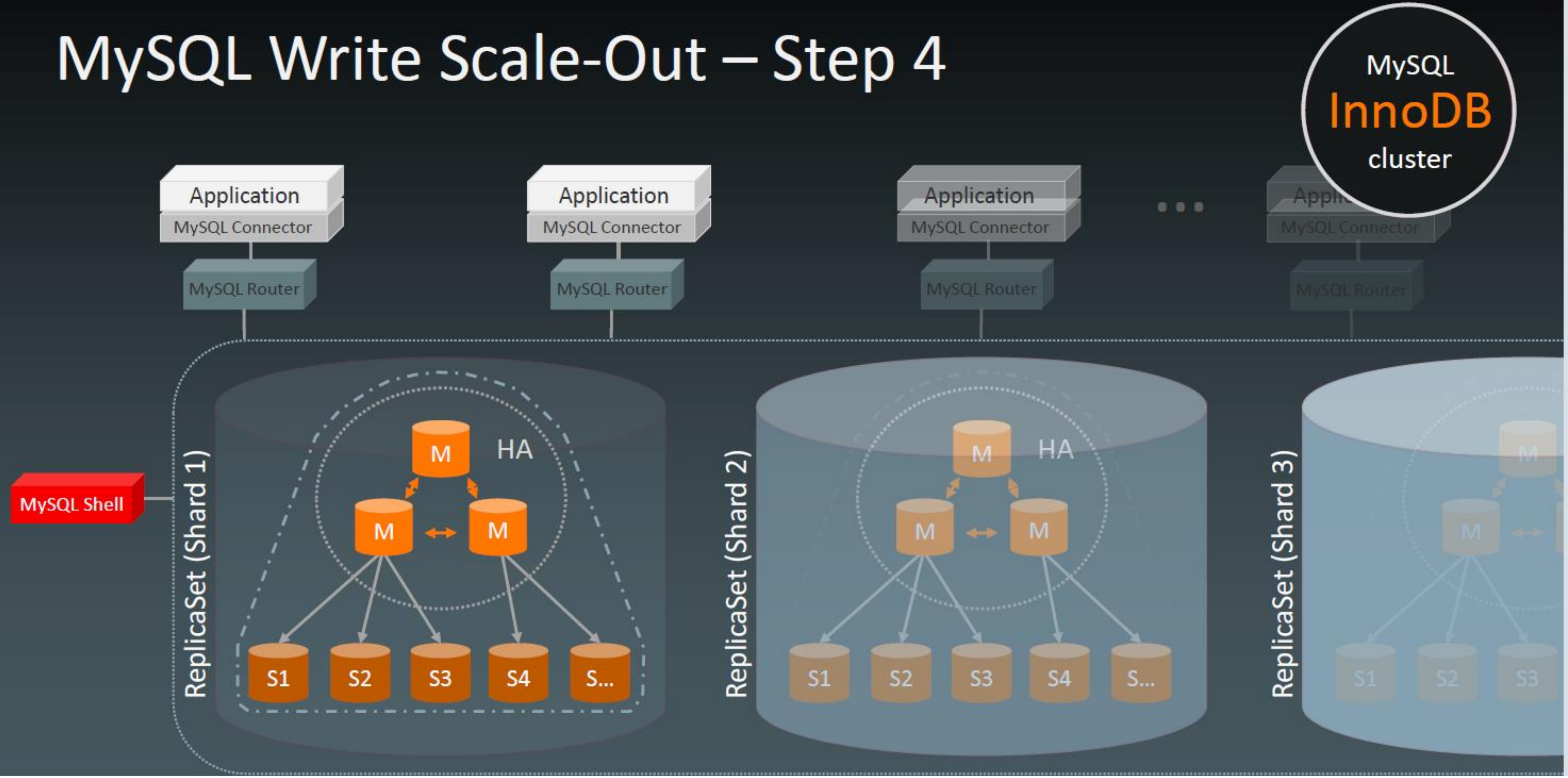
- Myth: *Relational databases don't scale for big data*
- Truth: Build your database using document model principles, and a RDBMS will scale as well!
  - Relationally designed databases are hard to scale **horizontally** (shard)
  - Foreign keys, transactional semantics, JOINs, strong global consistency, etc. … make it difficult to partition the data across servers

- **MySQL Document Store will make it easy to build big scale databases**
  - Applications and database are designed in a way to simplify sharding
  - Certain features are avoided (or used carefully)

MySQL Write Scale-Out – 4 Steps

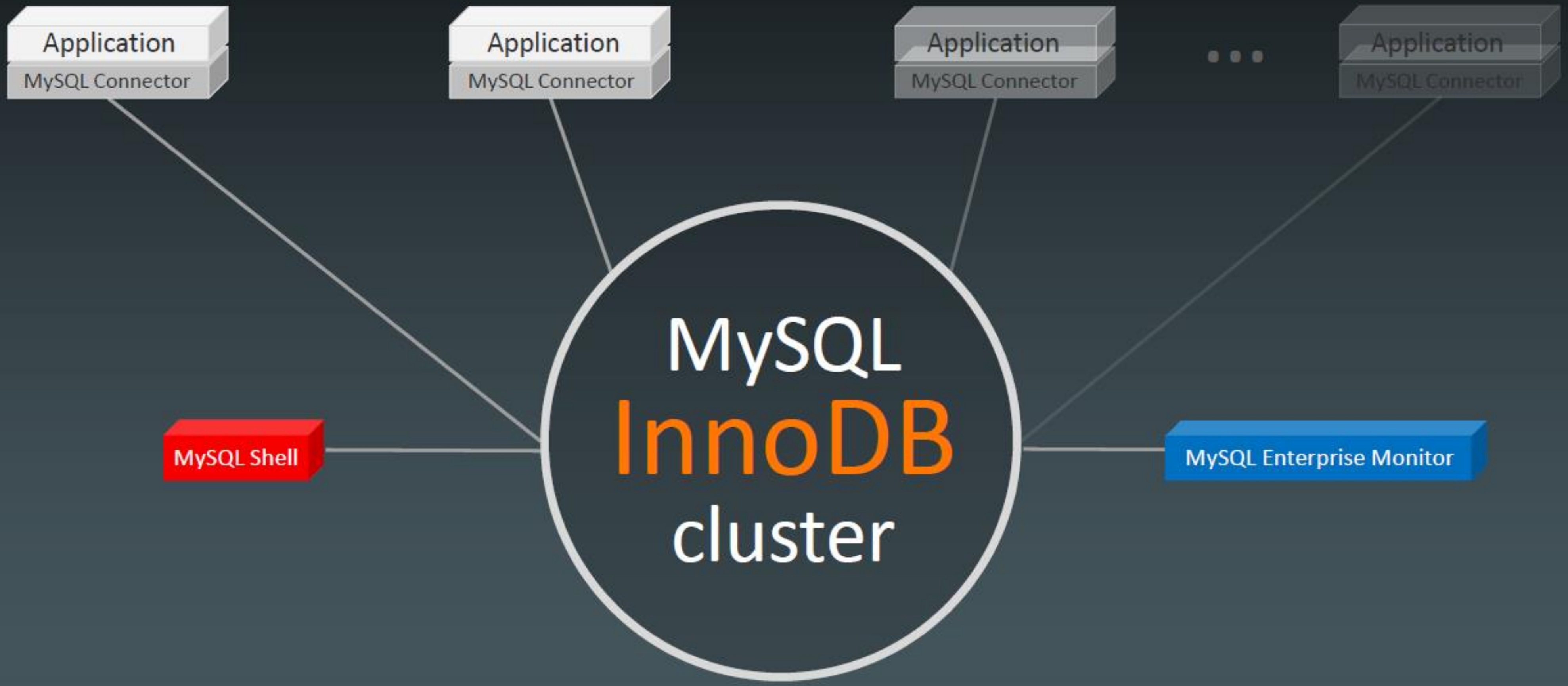


MySQL Write Scale-Out – Step 4



ORACLE

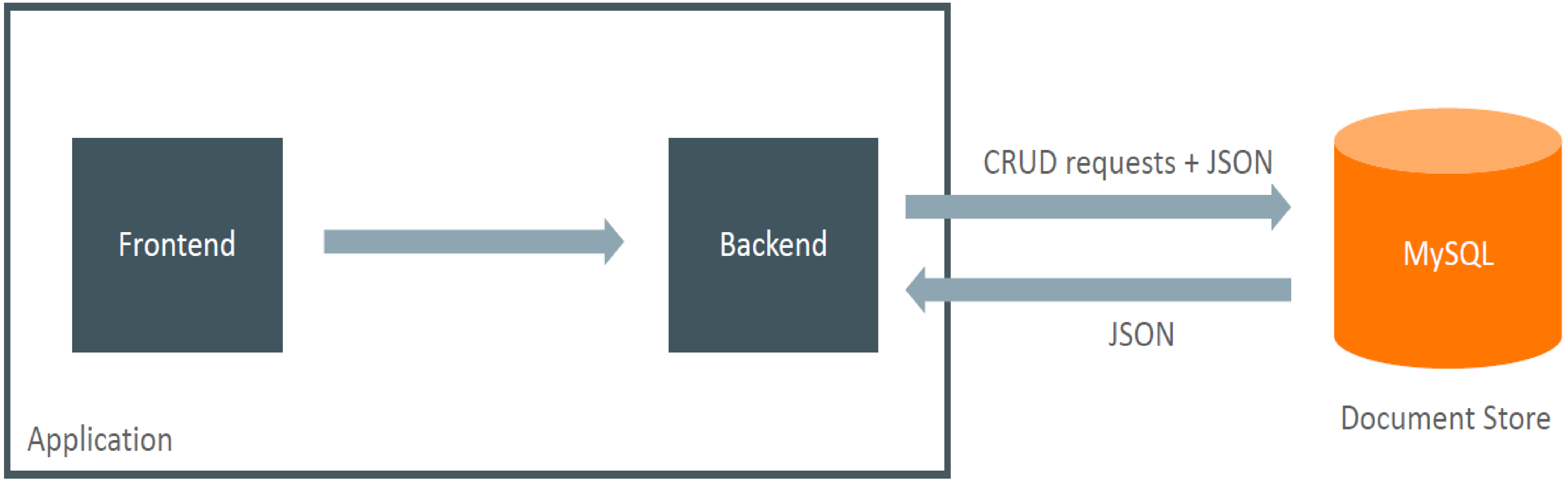
MySQL InnoDB Cluster – Architecture



How does it work?

How does the Document Store work?

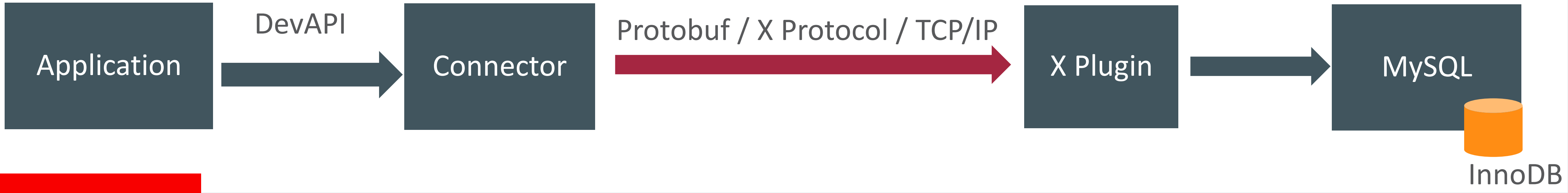
Architecture from the Application's POV



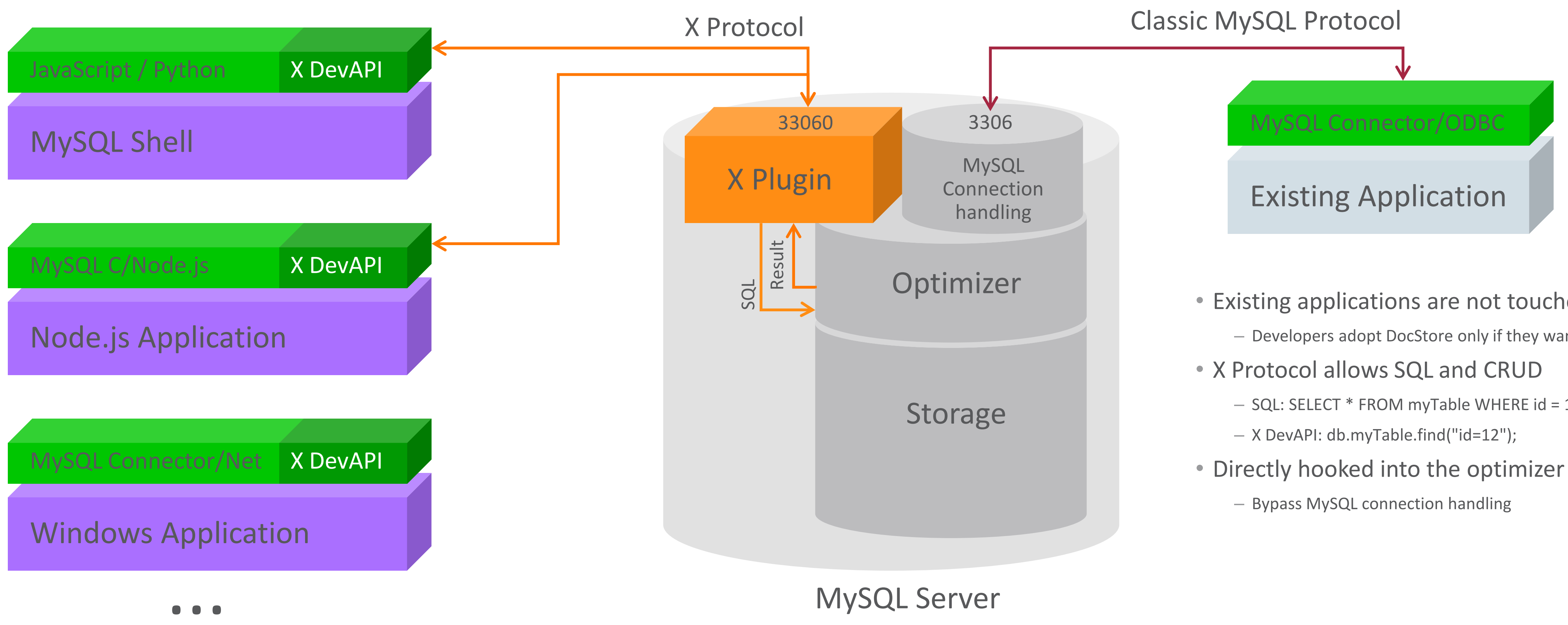
How it Works

Architecture - Components

- Applications use DevAPI connectors to write database operations in native code (instead of SQL)
- Connector translates DevAPI operations to X protocol document requests
- X Plugin translates document requests to SQL
- Results sent back to application as JSON documents



MySQL Document Store – How it works



- Existing applications are not touched
  - Developers adopt DocStore only if they want to
- X Protocol allows SQL and CRUD
  - SQL: SELECT * FROM myTable WHERE id = 12;
  - X DevAPI: db.myTable.find("id=12");
- Directly hooked into the optimizer
  - Bypass MySQL connection handling

Document Store DevAPI

- Commands serialized into Protobuf messages on the client side
- Transported via new "X Protocol" to the server
- Collections are stored as InnoDB tables
  - ACID compliance, transactions, replication, row locking etc all work as in plain MySQL

ORACLE

MySQL Document Store – Components

X Dev API

- New, modern, async developer API for CRUD and SQL operations on top of X Protocol
- Introduces Collections as new Schema obj.

X Protocol

- New MySQL client protocol based on top of industry standard (Protobuf)
- Works for both, CRUD and SQL operations

X Plugin

- Introduces X Protocol for relational- and document operations
- Maps CRUD operation to tables

MySQL Shell

- Offers interactive X DevAPI mode for app prototyping

ORACLE

The X DevAPI – A modern CRUD App Programming Interface

Document Store DevAPI

Overview

- A Document-oriented database built on top of MySQL
- Native language API
  - Write queries and DB code directly in JavaScript, Python, C#, PHP, Java, etc.
- CRUD methods to insert, query, modify and delete JSON documents
- Relational database aspects are abstracted when working with documents
  - Dev focuses on Collections versus tables, columns, or schema
  - Just documents in collections
  - Simplified interface for indexing document fields
- ...but relational tables can also be used

Document Store DevAPI

Main Features

- Introducing the **Collection** Schema Object
  - Abstraction of a table for storing JSON Documents
- Modern API using method chaining
  - db.products.find("name like :n").bind("n", searchString).execute().fetch_all();
- CRUD
  - .find(), .add(), .modify(), .remove()
- Indexing, Transactions, Row Locking, …

Example : Add, view JSON document

```
# create a schema to store the collection
        session.createSchema("items")

# create a collection
        db.createCollection("items_table")

# add JSON document to a collection
        db.items_table.add({"name":"washing machine","price":10000,"color":"white"})
        db.items_table.add({"name":"refrigerator","price":30000,"color":"steel grey"})
        db.items_table.add({"name":"samsung tv","price":40000,"color":"black"})
        db.items_table.add({"name":"MacBook","price":90000,"color":"ivory"})

# view the newly added items to the collection
         db.items_table.find()
```

Example : Modify JSON document

```
# create a schema to store the collection
        db.items_table.modify("name = 'samsung tv'").set("price", 15000)

# view the newly modified items
        db.items_table.find("name='samsung tv'")
[
  {
    "_id": "00005b6eba8e0000000000000003",
    "color": "black",
    "name": "samsung tv",
    "price": 15000
  }
]
```

Example : Delete JSON document

```
# delete document
        db.items_table.remove("name='washing machine'")
# view document after deletion
        db.items_table.find()
[
  {
    "_id": "00005b6eba8e0000000000000002",
    "color": "steel grey",
    "name": "refrigerator",
    "price": 30000
  },
  {
    "_id": "00005b6eba8e0000000000000003",
    "color": "black",
    "name": "samsung tv",
    "price": 15000
  }
]
```

Example : Read JSON document

# Search and list documents satisfying a condition

```
        db.items_table.find("price>25000")
[
  {
    "_id": "00005b6eba8e0000000000000002",
    "color": "steel grey",
    "name": "refrigerator",
    "price": 30000
  },
  {
    "_id": "00005b6eba8e0000000000000004",
    "color": "ivory",
    "name": "MacBook",
    "price": 90000
  }
]
```

Example: Comparing with raw SQL...

```
SELECT JSON_OBJECT(
                    'name', JSON_EXTRACT(doc,'$.name'),
                    'zip', JSON_EXTRACT(doc, '$.address.zip'))
          FROM `order`
          WHERE (JSON_UNQUOTE(JSON_EXTRACT(doc,'$.address.zip')) IN
                                                ('91234','94231'));
```

```
order.find("address.zip in ('91234', '94231')").
          patchFields({'name':'name', 'zip':'address.zip'});
```

Connectors for Applications

You can read much more about each of the products on the announcement blogs:

https://insidemysql.com/mysql-8-0-welcome-to-the-devapi/

- Java –
- .NET –
- Node.JS –
- C++ –
- Python –
- PHP –
- ODBC

Combining Document Store With Relational Model

SQL Interface to the Document Store

Document Store with SQL

- Available starting with 5.7
- JSON Datatype
- JSON Functions
- JSON Path Syntax
- JSON Indexing
- SQL Syntax Extensions

Document Store with SQL - JSON Datatype

- Store JSON data in table columns
- Validates format
- Internal binary format designed for faster lookup & partial updates
- Mix & Match with SQL
- Convert (cast) to and from string

Document Store with SQL

JSON Functions

- Construct JSON values
  - JSON_OBJECT('field', 'value', ...) → {"field": "value", ...}
  - JSON_ARRAY(1, 2, 3) → [1,2,3]
  - JSON_QUOTE('string')
- Query contents
  - JSON_EXTRACT('{"field": "value"}', '$.field') → "value"
  - JSON_CONTAINS('[1,2,3]', '3') → 1 *(true)*
  - JSON_KEYS(), JSON_CONTAINS_PATH(), JSON_LENGTH() etc

ORACLE®

Document Store with SQL

JSON Functions

- Modify JSON values
  - JSON_SET('{"name": "Alice"}', '$.name', 'Bob') → {"name": "Bob"}
  - JSON_INSERT(), JSON_APPEND(), JSON_ARRAY_APPEND() etc
- Aggregate rows into arrays or objects
  - JSON_ARRAYAGG(), JSON_OBJECTAGG()
  - SELECT JSON_ARRAYAGG(name) FROM users

    → ['alice', 'bob', …]

ORACLE

Document Store with SQL

JSON Path Syntax

- Refer to fields inside a JSON document

```
{ "field":
    { "array":
        [{"value": 123}]}}
$.field.array[0].value
```

- Use in JSON functions
  - JSON_EXTRACT(document,'$.address.zip')
- Inline JSON Path Syntax to refer to JSON contents in SQL
  - SELECT doc->>'$.description' FROM products

ORACLE

Document Store with SQL

JSON Indexing

- Index on specific values inside JSON documents
- Virtual columns allow indexes on JSON fields
  - Create a virtual column to "look in" a JSON document
  - Create index on the virtual column
- Foreign keys can also be created on virtual columns

Document Store with SQL

EXAMPLE: Query JSON Objects from Table Columns

```
SELECT JSON_OBJECT('id', cu.id,
             'name', cu.name,
             'email', cu.email,
             'city', ci.city) as customer
  FROM customer cu
  JOIN city ci ON ci.id = cu.city_id
```

ORACLE

Native Performance Comparison

Unindexed traversal of 206K documents

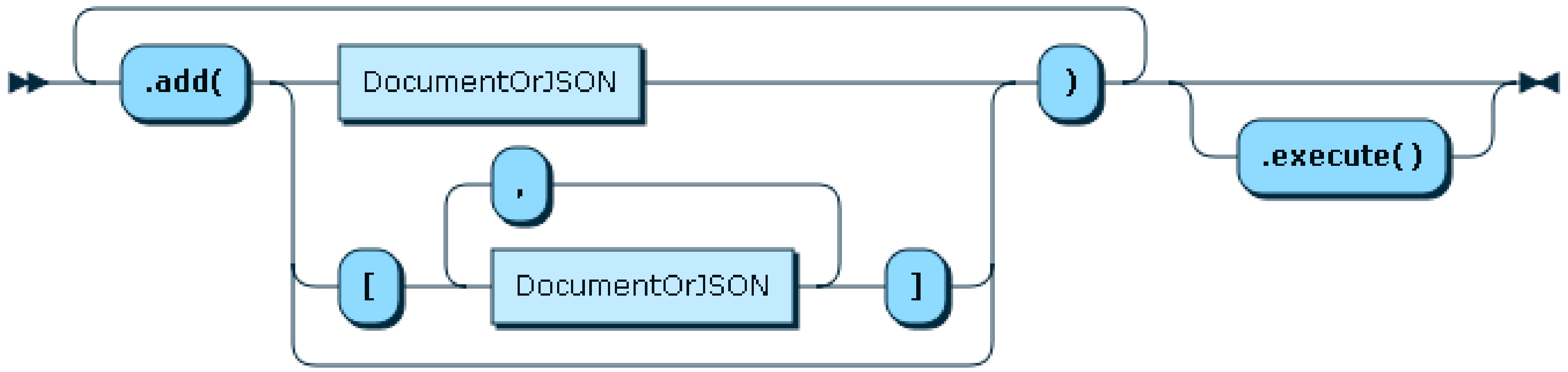
```
# as JSON type
SELECT DISTINCT
 feature->"$.type" as json_extract
FROM features;
+---------------+
| json_extract |
+---------------+
| "Feature"    |
+---------------+
1 row in set (1.25 sec)
```

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

Explanation: Binary format of JSON type is very efficient at searching. Storing as TEXT performs over 10x worse at traversal.

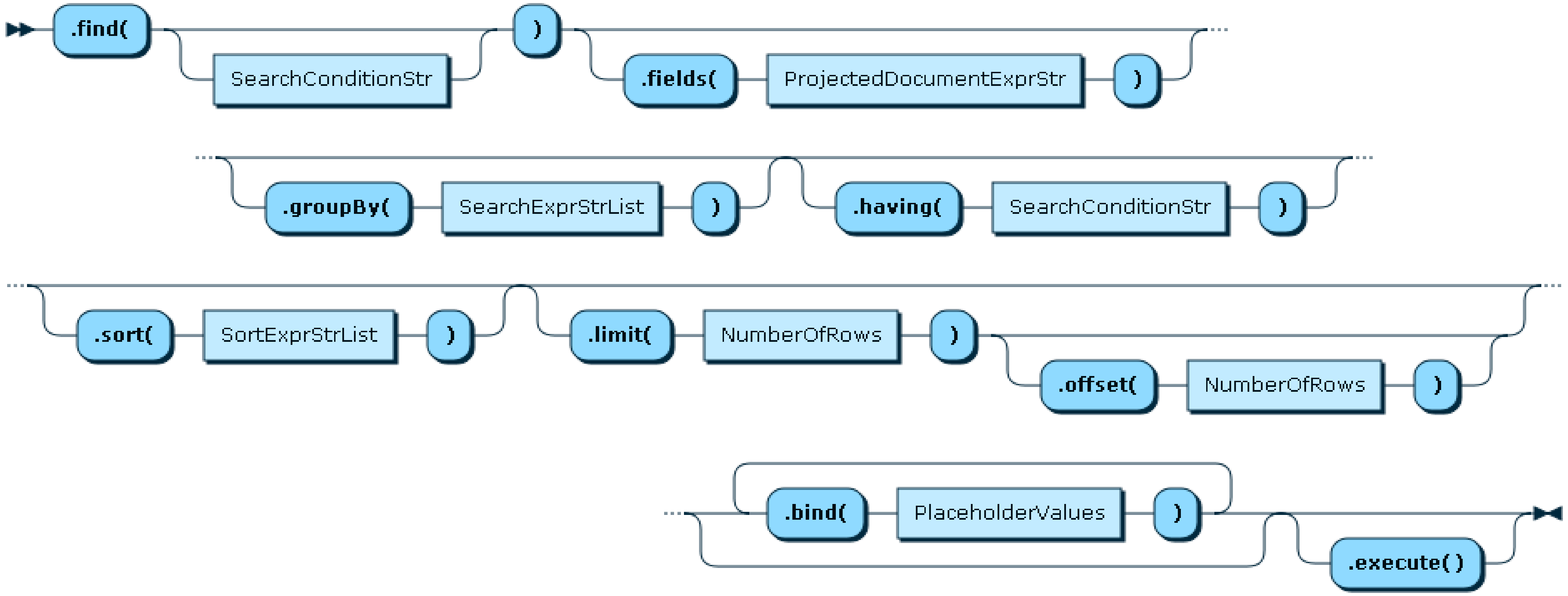
ORACLE

Collection Add Function – EBNF Notation



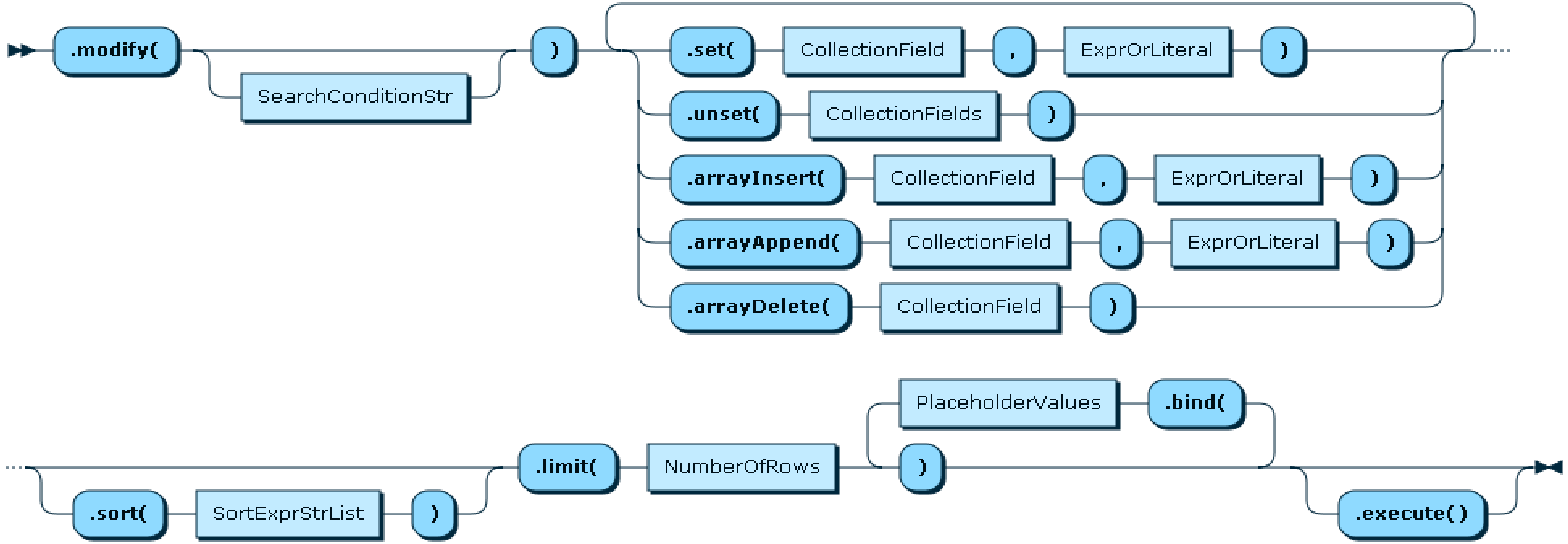
```
products.add({"name":"xyz", "dept":"IT"}).execute();
```

Collection Find Function



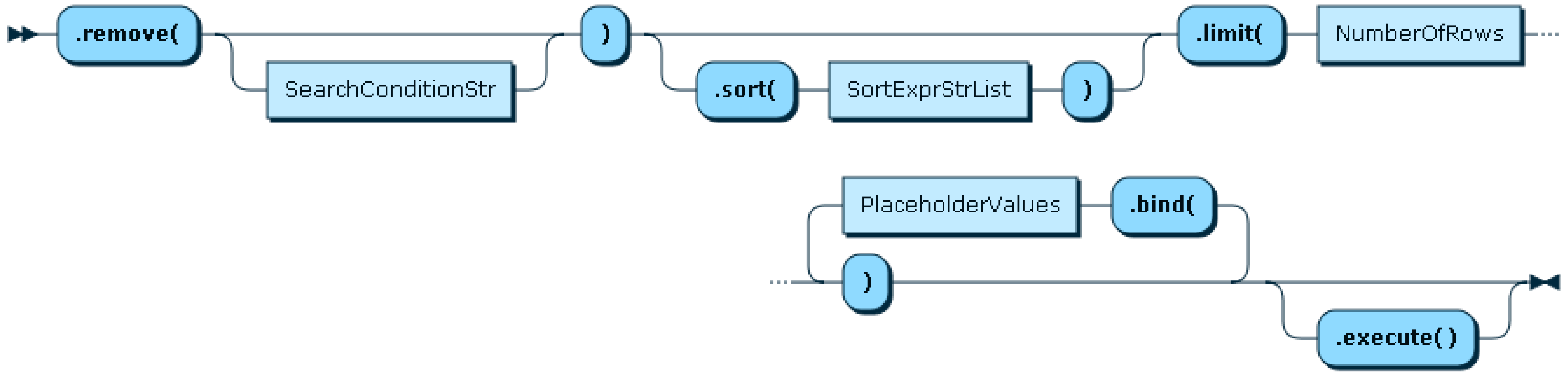
```
products.find("dept = ‘IT'").sort(["name"]).execute();
```

Collection Modify Function



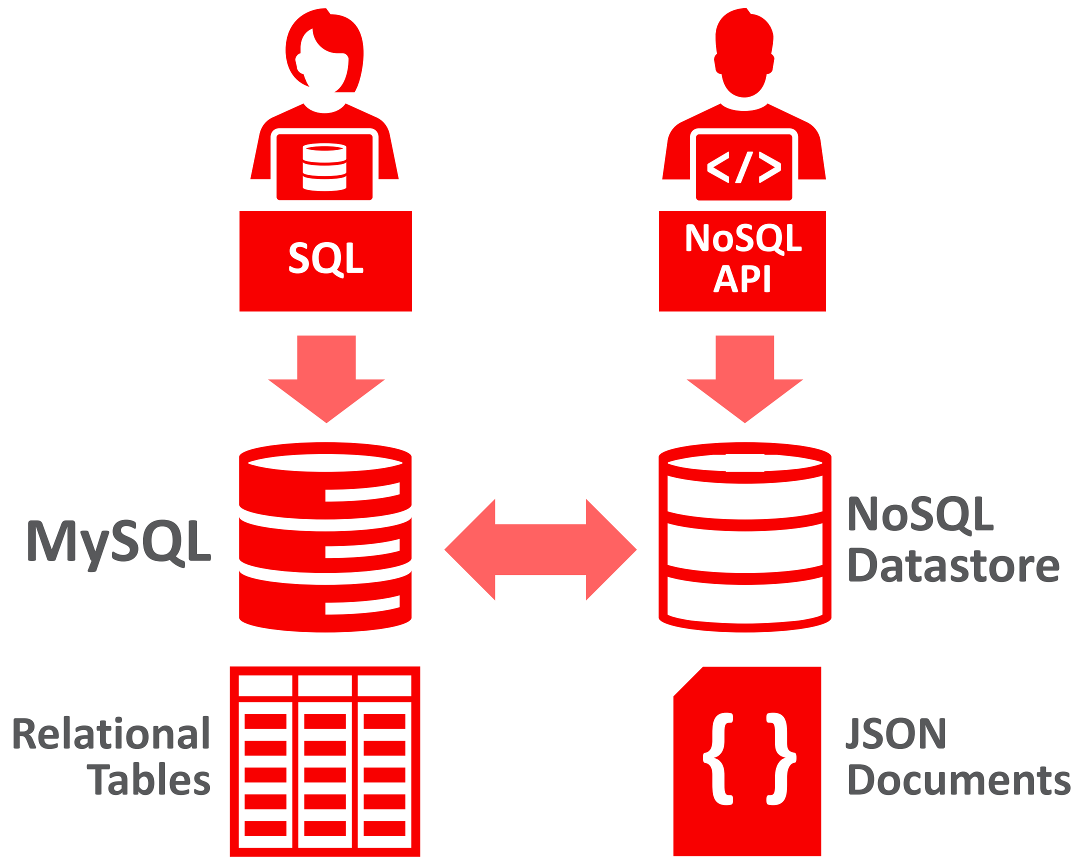
```
products.modify("product_id = 123").set("dept", "HR").execute();
```

Collection Remove Function



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

Pains running RDBMS + NoSQL Datastore

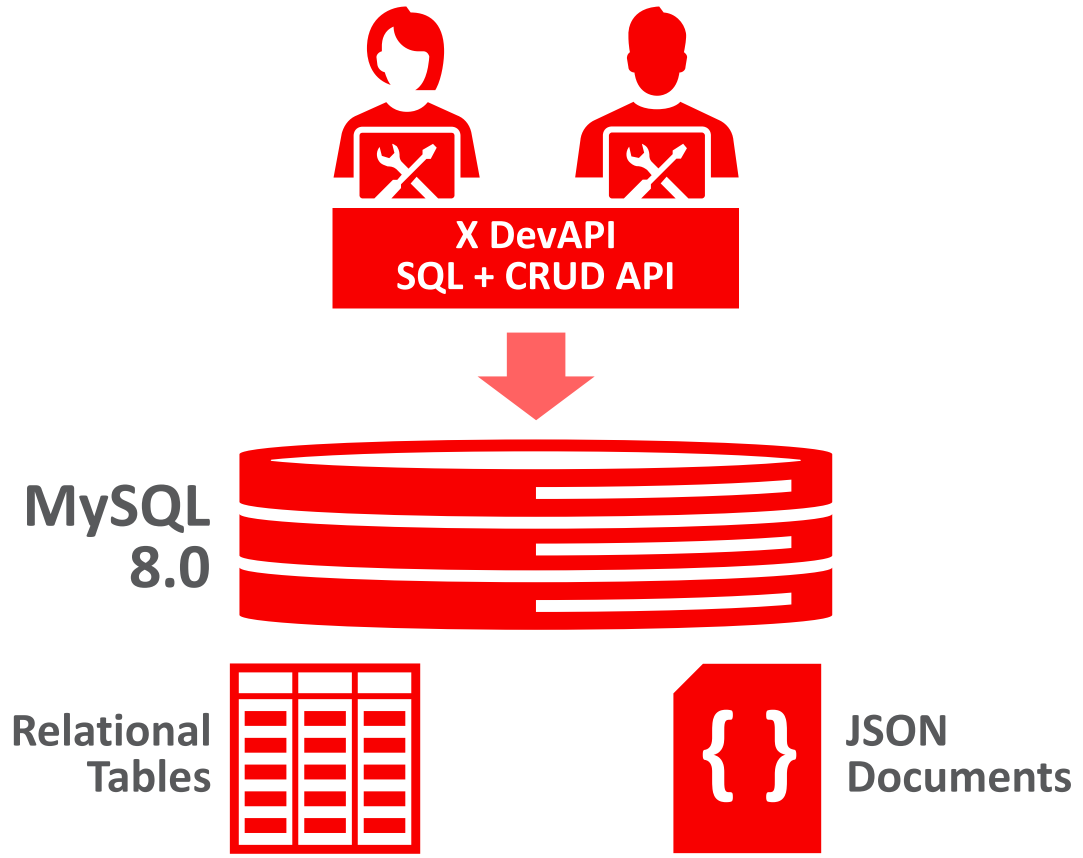


Developers
Required to learn multiple APIs

Data management
Difficult to keep data synchronization between tables and JSON documents

Operations
Required to manage multiple products with different tools

Solution by “SQL + NoSQL = MySQL”



For Developers
Unified API provides more flexibility

Data management
Single repository reliefs concerns on data synchronization

Operations
Managing single database with unified management tool

Getting Ready

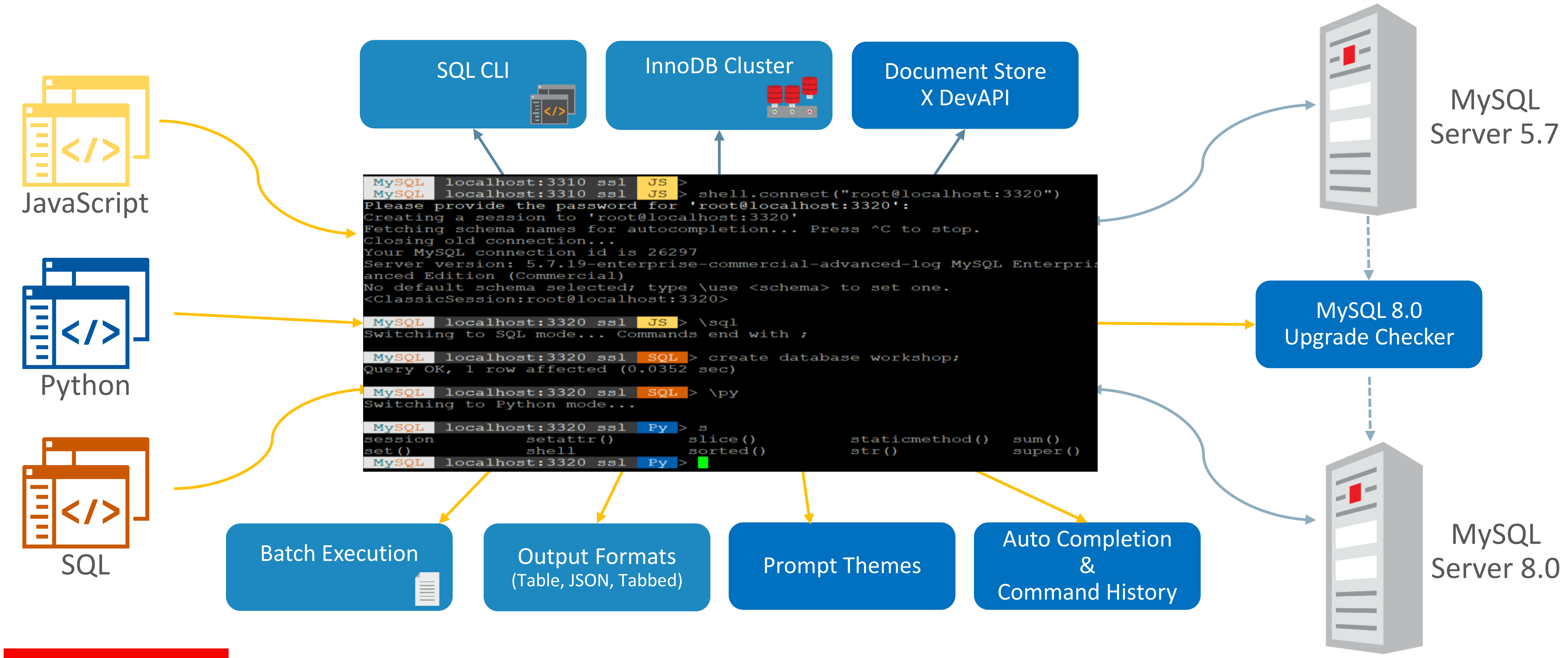
- MySQL Server 8.0
  - Binary downloads: https://dev.mysql.com/downloads/mysql/
  - MySQL Repos
    - https://dev.mysql.com/doc/mysql-yum-repo-quick-guide/en/
    - https://dev.mysql.com/doc/mysql-apt-repo-quick-guide/en/
- MySQL Shell 8.0
  - https://dev.mysql.com/downloads/shell/

Getting Ready

- Fast application prototyping using MySQL Shell
  - We will write a simple Document Store application in the MySQL Shell
  - This should give us a basic impression of how to code against MySQL Document Store
- Real world application development is done using MySQL Connectors
  - While the MySQL Shell is good for prototyping…
  - Real app development is done using an application framework, like Node.js

ORACLE

MySQL Shell 8.0



```
 MySQL  localhost:3310 ssl  JS >
 MySQL  localhost:3310 ssl  JS > shell.connect("root@localhost:3320")
Please provide the password for 'root@localhost:3320':
Creating a session to 'root@localhost:3320'
Fetching schema names for autocompletion... Press ^C to stop.
Closing old connection...
Your MySQL connection id is 26297
Server version: 5.7.19-enterprise-commercial-advanced-log MySQL Enterpri
anced Edition (Commercial)
No default schema selected; type \use <schema> to set one.
<ClassicSession:root@localhost:3320>

 MySQL  localhost:3320 ssl  JS > \sql
Switching to SQL mode... Commands end with ;

 MySQL  localhost:3320 ssl  SQL > create database workshop;
Query OK, 1 row affected (0.0352 sec)

 MySQL  localhost:3320 ssl  SQL > \py
Switching to Python mode...

 MySQL  localhost:3320 ssl  Py > s
session          setattr()        slice()          staticmethod()   sum()
set()            shell            sorted()         str()            super()
 MySQL  localhost:3320 ssl  Py >
```

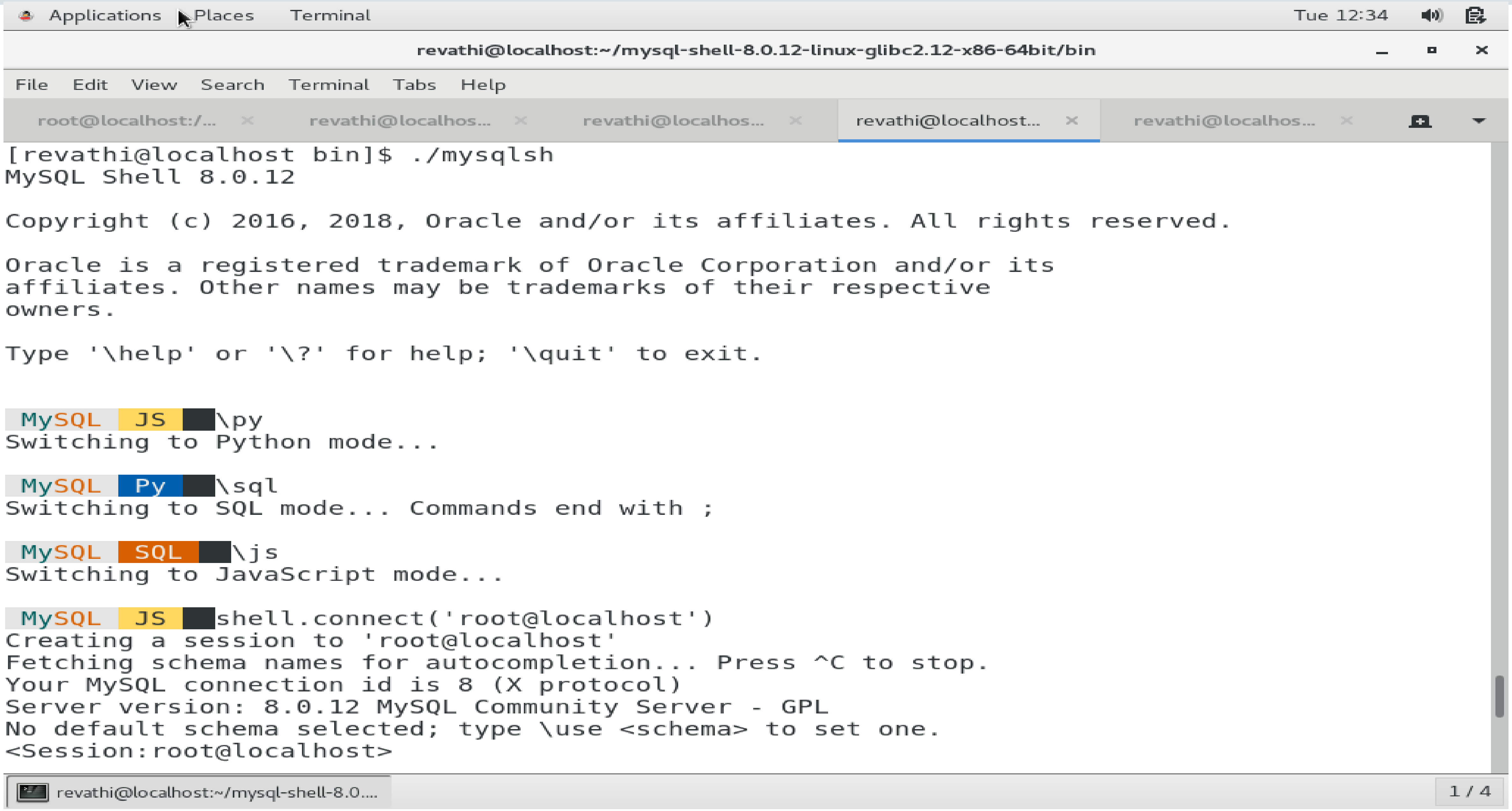
MySQL Document Store

Summary and Take Away

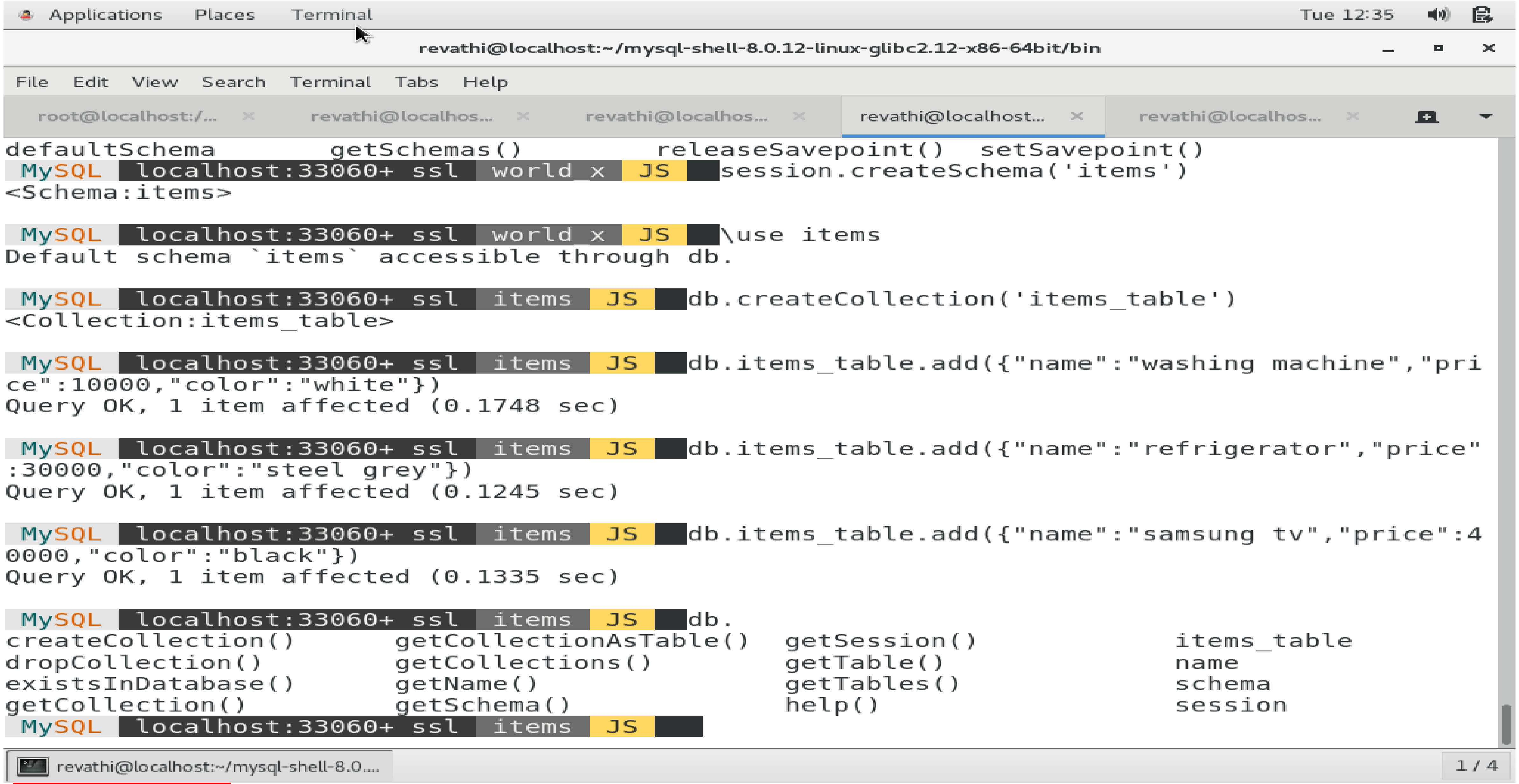
- New, modern way to develop database applications
- Combine best of relational and document oriented models
- MySQL InnoDB Cluster – Future proof for HA and scale-out deployments

- Blogs: mysqlserverteam.com/category/docstore/

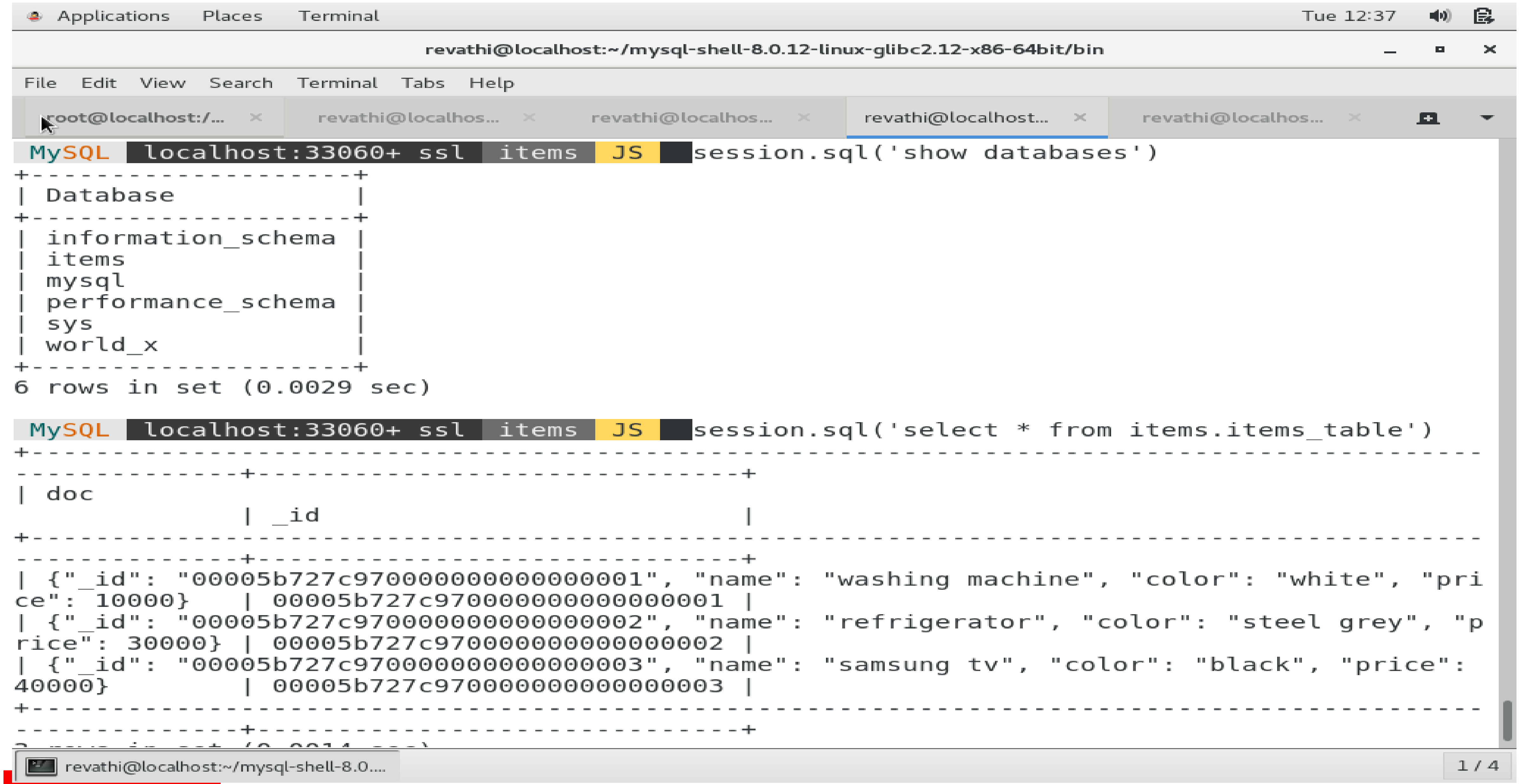
Demo!



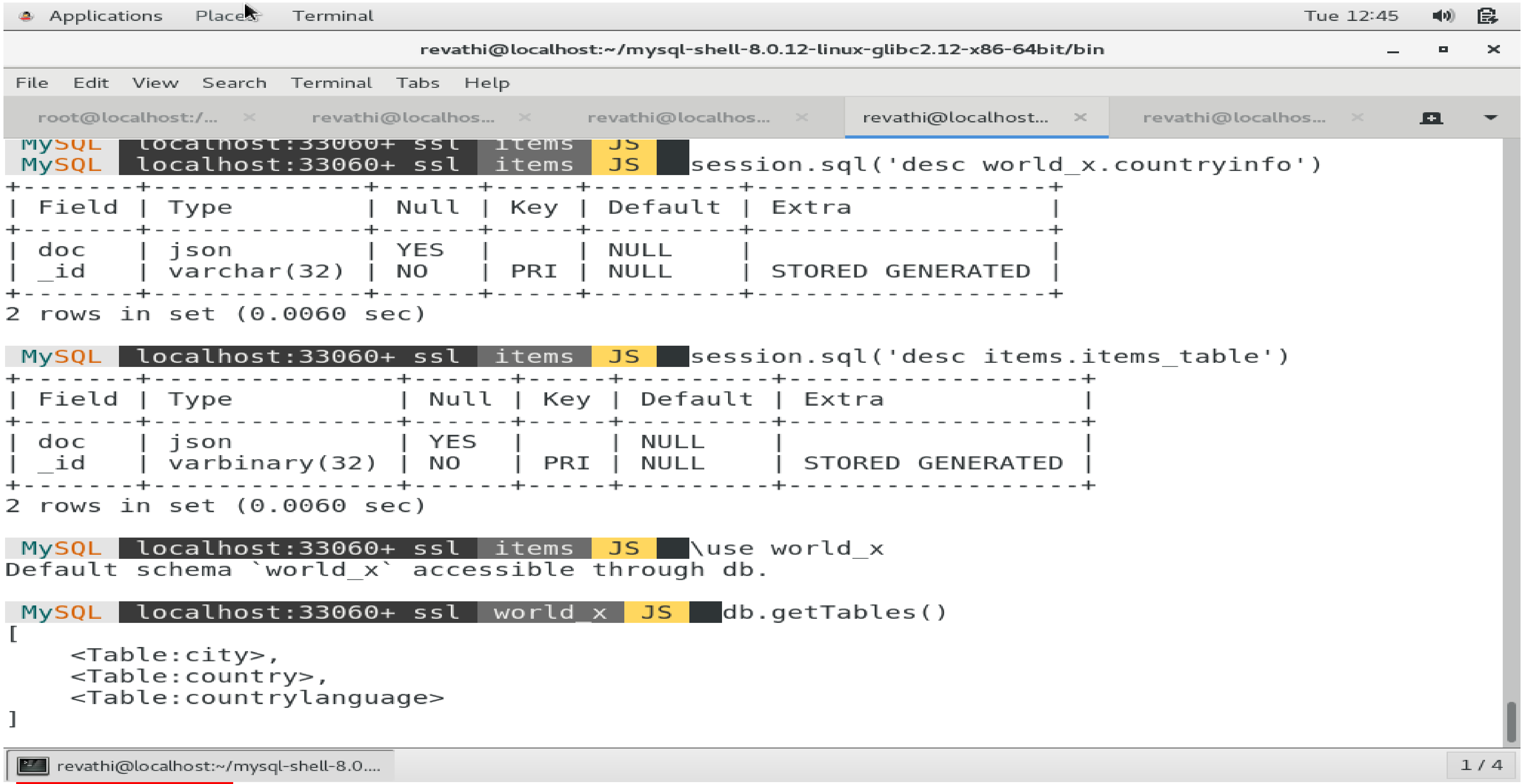
```
[revathi@localhost bin]$ ./mysqlsh
MySQL Shell 8.0.12

Copyright (c) 2016, 2018, 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' or '\?' for help; '\quit' to exit.


 MySQL  JS  \py
Switching to Python mode...

 MySQL  Py  \sql
Switching to SQL mode... Commands end with ;

 MySQL  SQL  \js
Switching to JavaScript mode...

 MySQL  JS  shell.connect('root@localhost')
Creating a session to 'root@localhost'
Fetching schema names for autocompletion... Press ^C to stop.
Your MySQL connection id is 8 (X protocol)
Server version: 8.0.12 MySQL Community Server - GPL
No default schema selected; type \use <schema> to set one.
<Session:root@localhost>
```



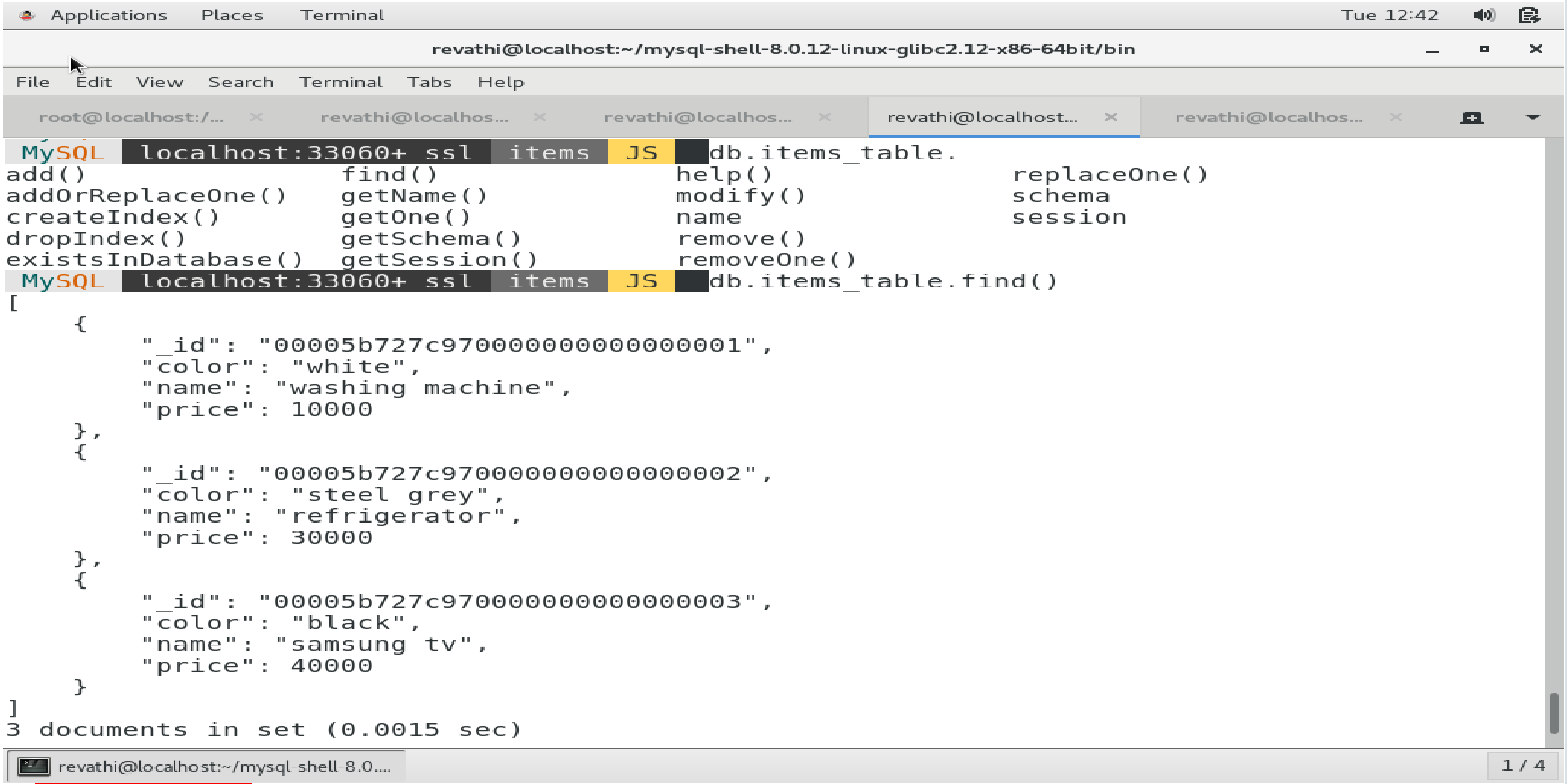
```
defaultSchema         getSchemas()          releaseSavepoint()  setSavepoint()
 MySQL  localhost:33060+ ssl  world_x  JS  session.createSchema('items')
<Schema:items>

 MySQL  localhost:33060+ ssl  world_x  JS  \use items
Default schema `items` accessible through db.

 MySQL  localhost:33060+ ssl  items  JS  db.createCollection('items_table')
<Collection:items_table>

 MySQL  localhost:33060+ ssl  items  JS  db.items_table.add({"name":"washing machine","pri
ce":10000,"color":"white"})
Query OK, 1 item affected (0.1748 sec)

 MySQL  localhost:33060+ ssl  items  JS  db.items_table.add({"name":"refrigerator","price"
:30000,"color":"steel grey"})
Query OK, 1 item affected (0.1245 sec)

 MySQL  localhost:33060+ ssl  items  JS  db.items_table.add({"name":"samsung tv","price":4
0000,"color":"black"})
Query OK, 1 item affected (0.1335 sec)

 MySQL  localhost:33060+ ssl  items  JS  db.
createCollection()        getCollectionAsTable()  getSession()            items_table
dropCollection()          getCollections()        getTable()              name
existsInDatabase()        getName()               getTables()             schema
getCollection()           getSchema()             help()                  session
 MySQL  localhost:33060+ ssl  items  JS
```



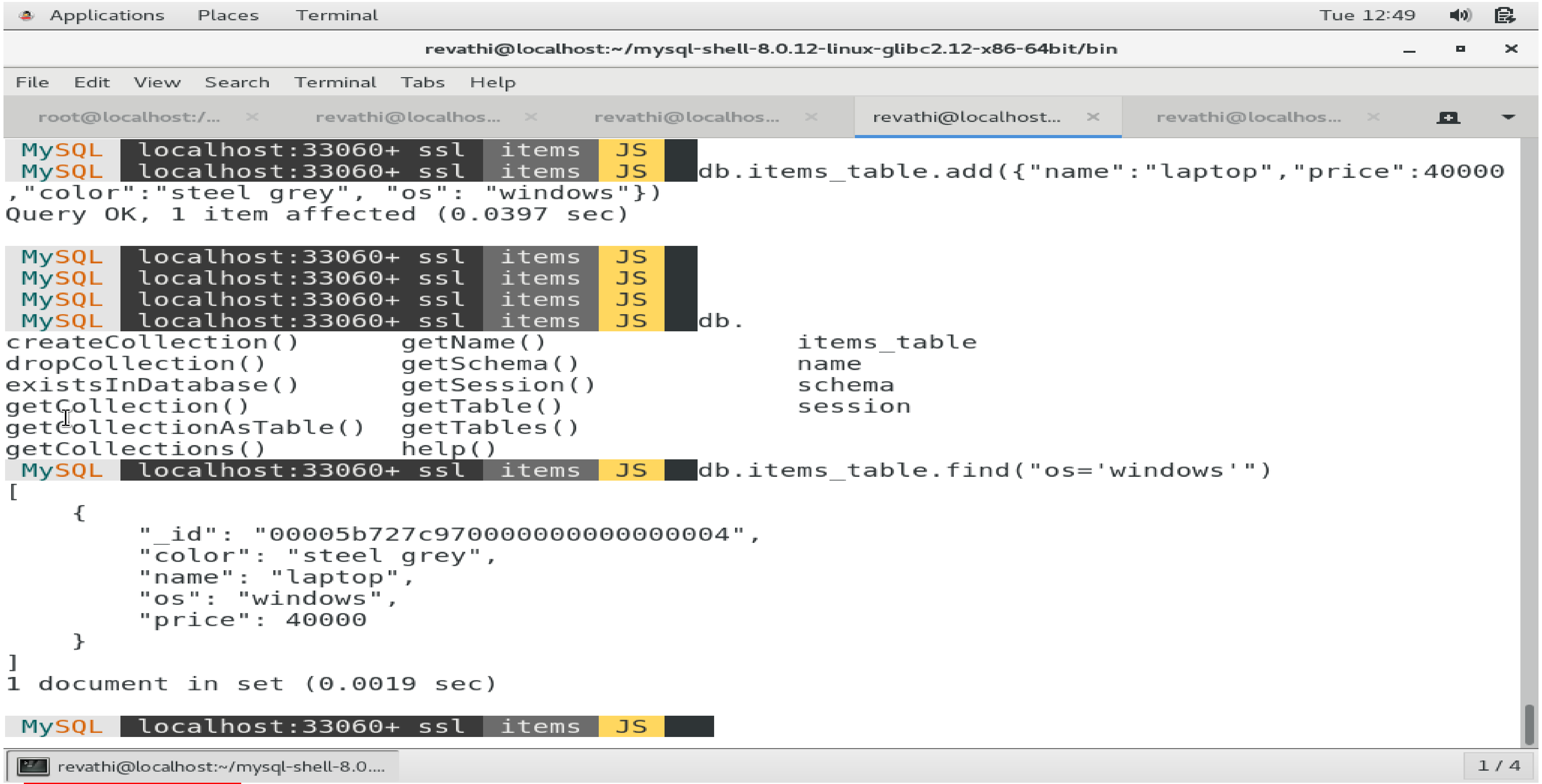
```
MySQL  localhost:33060+ ssl  items  JS  session.sql('show databases')
+--------------------+
| Database           |
+--------------------+
| information_schema |
| items              |
| mysql              |
| performance_schema |
| sys                |
| world_x            |
+--------------------+
6 rows in set (0.0029 sec)

MySQL  localhost:33060+ ssl  items  JS  session.sql('select * from items.items_table')
+------------------------------------------------------------------------------------------
--------------+-------------------------------+
| doc
              | _id                           |
+------------------------------------------------------------------------------------------
--------------+-------------------------------+
| {"_id": "00005b727c970000000000000001", "name": "washing machine", "color": "white", "pri
ce": 10000}   | 00005b727c970000000000000001 |
| {"_id": "00005b727c970000000000000002", "name": "refrigerator", "color": "steel grey", "p
rice": 30000} | 00005b727c970000000000000002 |
| {"_id": "00005b727c970000000000000003", "name": "samsung tv", "color": "black", "price":
40000}        | 00005b727c970000000000000003 |
+------------------------------------------------------------------------------------------
--------------+-------------------------------+
```



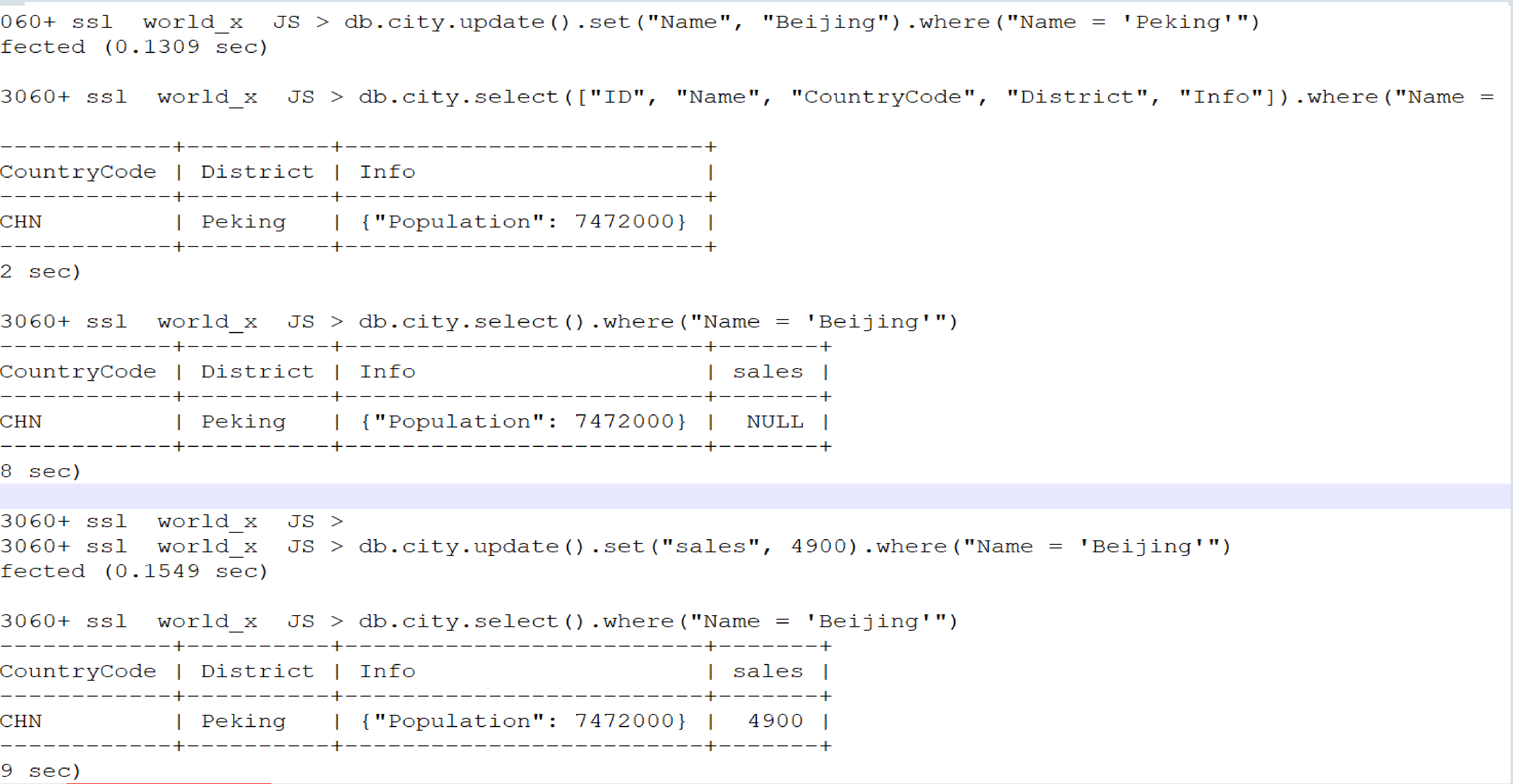
```
 MySQL  localhost:33060+ ssl  items  JS  session.sql('desc world_x.countryinfo')
+-------+---------------+------+-----+---------+-------------------+
| Field | Type          | Null | Key | Default | Extra             |
+-------+---------------+------+-----+---------+-------------------+
| doc   | json          | YES  |     | NULL    |                   |
| _id   | varchar(32)   | NO   | PRI | NULL    | STORED GENERATED  |
+-------+---------------+------+-----+---------+-------------------+
2 rows in set (0.0060 sec)

 MySQL  localhost:33060+ ssl  items  JS  session.sql('desc items.items_table')
+-------+-----------------+------+-----+---------+-------------------+
| Field | Type            | Null | Key | Default | Extra             |
+-------+-----------------+------+-----+---------+-------------------+
| doc   | json            | YES  |     | NULL    |                   |
| _id   | varbinary(32)   | NO   | PRI | NULL    | STORED GENERATED  |
+-------+-----------------+------+-----+---------+-------------------+
2 rows in set (0.0060 sec)

 MySQL  localhost:33060+ ssl  items  JS  \use world_x
Default schema `world_x` accessible through db.

 MySQL  localhost:33060+ ssl  world_x  JS  db.getTables()
[
    <Table:city>,
    <Table:country>,
    <Table:countrylanguage>
]
```


Applications Places Terminal
Tue 12:42
revathi@localhost:~/mysql-shell-8.0.12-linux-glibc2.12-x86-64bit/bin
File Edit View Search Terminal Tabs Help
root@localhost:/...
revathi@localhos...
revathi@localhos...
revathi@localhost...
revathi@localhos...
revathi@localhost:~/mysql-shell-8.0....
1 / 4

```
 MySQL  localhost:33060+ ssl  items  JS  db.items_table.
add()                    find()                   help()                   replaceOne()
addOrReplaceOne()        getName()                modify()                 schema
createIndex()            getOne()                 name                     session
dropIndex()              getSchema()              remove()
existsInDatabase()       getSession()             removeOne()
 MySQL  localhost:33060+ ssl  items  JS  db.items_table.find()
[
    {
        "_id": "00005b727c9700000000000000001",
        "color": "white",
        "name": "washing machine",
        "price": 10000
    },
    {
        "_id": "00005b727c9700000000000000002",
        "color": "steel grey",
        "name": "refrigerator",
        "price": 30000
    },
    {
        "_id": "00005b727c9700000000000000003",
        "color": "black",
        "name": "samsung tv",
        "price": 40000
    }
]
3 documents in set (0.0015 sec)
```



```
 MySQL  localhost:33060+ ssl  items  JS
 MySQL  localhost:33060+ ssl  items  JS  db.items_table.add({"name":"laptop","price":40000
,"color":"steel grey", "os": "windows"})
Query OK, 1 item affected (0.0397 sec)

 MySQL  localhost:33060+ ssl  items  JS
 MySQL  localhost:33060+ ssl  items  JS
 MySQL  localhost:33060+ ssl  items  JS
 MySQL  localhost:33060+ ssl  items  JS  db.
createCollection()        getName()                 items_table
dropCollection()          getSchema()               name
existsInDatabase()        getSession()              schema
getCollection()           getTable()                session
getCollectionAsTable()    getTables()
getCollections()          help()
 MySQL  localhost:33060+ ssl  items  JS  db.items_table.find("os='windows'")
[
    {
        "_id": "00005b727c9700000000000004",
        "color": "steel grey",
        "name": "laptop",
        "os": "windows",
        "price": 40000
    }
]
1 document in set (0.0019 sec)

 MySQL  localhost:33060+ ssl  items  JS
```



```
060+ ssl  world_x  JS > db.city.update().set("Name", "Beijing").where("Name = 'Peking'")
fected (0.1309 sec)

3060+ ssl  world_x  JS > db.city.select(["ID", "Name", "CountryCode", "District", "Info"]).where("Name =

-----------+----------+-------------------------+
CountryCode | District | Info                    |
-----------+----------+-------------------------+
CHN         | Peking   | {"Population": 7472000} |
-----------+----------+-------------------------+
2 sec)

3060+ ssl  world_x  JS > db.city.select().where("Name = 'Beijing'")
-----------+----------+-------------------------+-------+
CountryCode | District | Info                    | sales |
-----------+----------+-------------------------+-------+
CHN         | Peking   | {"Population": 7472000} |  NULL |
-----------+----------+-------------------------+-------+
8 sec)

3060+ ssl  world_x  JS >
3060+ ssl  world_x  JS > db.city.update().set("sales", 4900).where("Name = 'Beijing'")
fected (0.1549 sec)

3060+ ssl  world_x  JS > db.city.select().where("Name = 'Beijing'")
-----------+----------+-------------------------+-------+
CountryCode | District | Info                    | sales |
-----------+----------+-------------------------+-------+
CHN         | Peking   | {"Population": 7472000} |  4900 |
-----------+----------+-------------------------+-------+
9 sec)
```

ORACLE

```
Delete a row
============
 MySQL  localhost:33060+ ssl  world_x  JS > db.city.delete().where("ID = 4080")
Query OK, 1 item affected (0.0850 sec)
```