Safe Harbor

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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

MySQL Document Store *Tutorial*

Dave Stokes **David.Stokes@Oracle.com**

Community Manager
MySQL Community Team

Tutorial

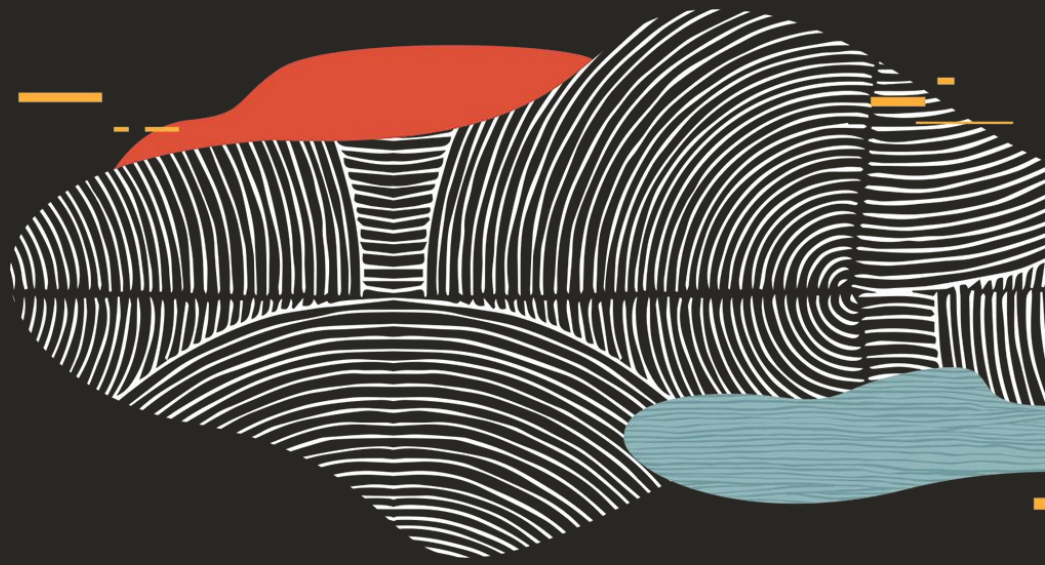
Workbook

A workbook with supporting text for the slide presentation and some extras is available at https://github.com/davidmstokes/preFOSDEM as a PDF file

Software

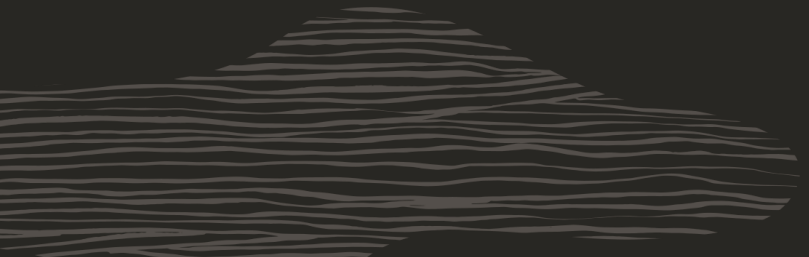
If you would like to follow along with the examples & perform the exercises, you will need to have a copies of MySQL Server and MySQL shell installed on your laptop (latest version please) plus a copy of the world_x database for the last part of the tutorial.

Our Goal



To show you how easy it is to use the MySQL Document Store!

No need to normalize data, setup schemas, or indexes BEFORE you can save your data.



Why this matters

"Go ahead a start coding and we will tell you want you are working on later…"

"Can we add this field to the data?"

"Uh, this SQL thing? Is it a pizza topping?"

What We Are Going to Do

- Too much of Dave talking
- Examples
- Exercises
- Much more Dave talking
- Your questions – always welcome

Benefits

One Database

No need to maintain the same data in SQL and JSON NoSQL.

Transactions

ACID Compliance for data consistence and easy management.

High Availability

Use MySQL Group Replication to scale reads and automatic failover in case master server fails.

Secure by Default

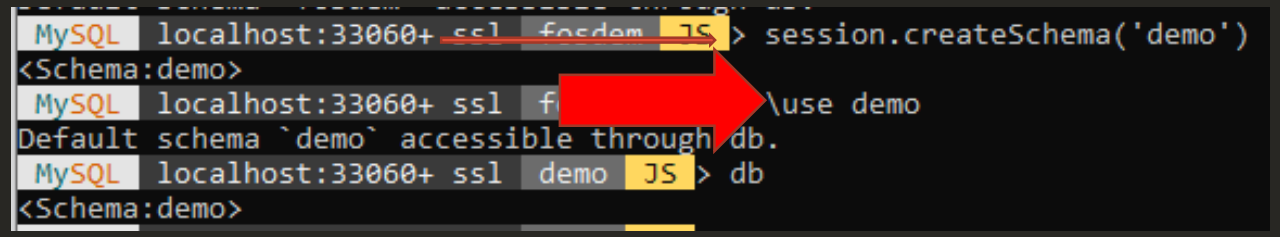
MySQL 8.0 is secure by default and there is the option to use the Enterprise Edition's Transparent Data Encryption.

So what does a NoSQL MySQL look like? Page 2

```
 MySQL  localhost:33060+ ssl  fosdem  JS > \use fosdem
Default schema `fosdem` accessible through db.
 MySQL  localhost:33060+ ssl  fosdem  JS > db
<Schema:fosdem>
 MySQL  localhost:33060+ ssl  fosdem  JS > db.createCollection('a')
<Collection:a>
 MySQL  localhost:33060+ ssl  fosdem  JS > db.a
<Collection:a>
 MySQL  localhost:33060+ ssl  fosdem  JS > db.a.add( { foo: "bar" } )
Query OK, 1 item affected (0.0062 sec)
 MySQL  localhost:33060+ ssl  fosdem  JS > db.a.find()
{
    "_id": "00005df0ed910000000000000002",
    "foo": "bar"
}
1 document in set (0.0005 sec)
```

- It is very simple to connect to a MySQL server, create a collection, and add a document without using SQL

Creating a Schema Page 3



- Once you have logged in a create a session, it is very simple to create a schema to house collections of document.

Now we can create a collection of documents

- Let us start with a collection to keep documents with information

```
MySQL  localhost:33060+                    > db.createCollection('names');
<Collection:names>
MySQL  localhost:33060+ ssl  demo  JS > db.names.add( { "name" : "Dave", home: "Texas"})
Query OK, 1 item affected (0.0145 sec)
MySQL  localhost:33060+ ssl  demo  JS > db.names.find()
{
    "_id": "00005e20838d0000000000000001",
    "home": "Texas",
    "name": "Dave"
}
1 document in set (0.0005 sec)
MySQL  localhost:33060+ ssl  demo  JS >
```

Data is added

- Data in the form a of JSON Document is added to the collection

```
MySQL  localhost:33060+ ssl  demo  JS > db.createCollection('names');
<Collection:names>
MySQL  localhost:33060+ ssl  demo  JS > db.names.add( { "name" : "Dave", home: "Texas"})
Query OK, 1 item affected (0.0145 sec)
MySQL  localhost:33060+ ssl  demo  JS > db.names.find()
{
    "_id": "00005e20838d0000000000000001",
    "home": "Texas",
    "name": "Dave"
}
1 document in set (0.0005 sec)
MySQL  localhost:33060+ ssl  demo  JS >
```

Query the data

- The data is returned in JSON format with the data entered previously and a new _id fields

```
MySQL  localhost:33060+ ssl  demo  JS > db.createCollection('names');
<Collection:names>
MySQL  localhost:33060+ ssl  demo  JS > db.names.add( { "name" : "Dave", home: "Texas"})
Query OK, 1 item affected (0.0145 sec)
MySQL  localhost:33060+ ssl  demo  JS > db.names.find()
{
    "_id": "00005e20838d0000000000000001",
    "home": "Texas",
    "name": "Dave"
}
1 document in set (0.0005 sec)
MySQL  localhost:33060+ ssl  demo  JS >
```

_id

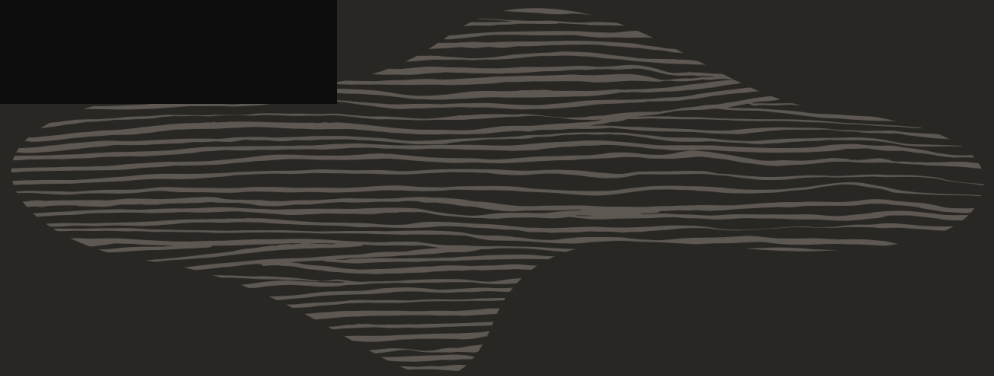
For now we are going to ignore the **_id** field

But it will be explained later

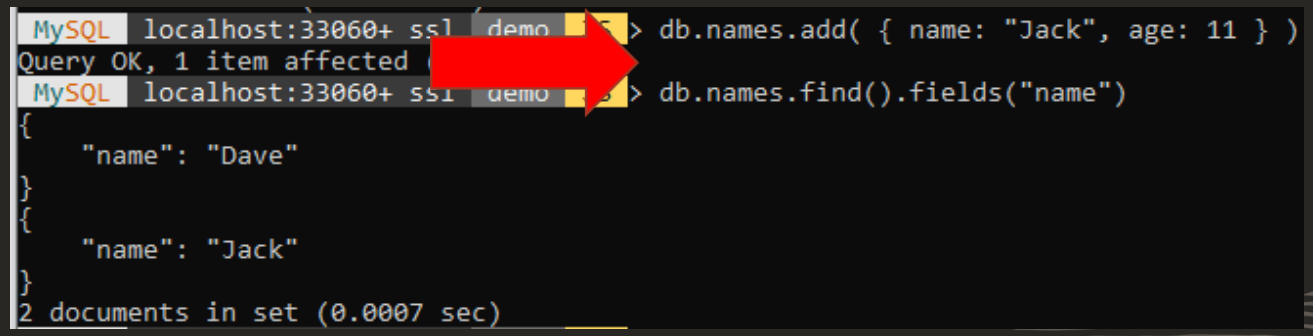
All this was done without an Structured Query Language (SQL)

- The X DevAPI uses modern programming structure, avoiding the often confusing syntax of SQL

```
MySQL  localhost:33060+ ssl  demo  JS > db.createCollection('names');
<Collection:names>
MySQL  localhost:33060+ ssl  demo  JS > db.names.add( { "name" : "Dave", home: "Texas"})
Query OK, 1 item affected (0.0145 sec)
MySQL  localhost:33060+ ssl  demo  JS > db.names.find()
{
    "_id": "00005e20838d0000000000000001",
    "home": "Texas",
    "name": "Dave"
}
1 document in set (0.0005 sec)
MySQL  localhost:33060+ ssl  demo  JS >
```



Adding another document is easy Page 4



- The nature of JSON Documents is schema less – so the new document can have different key/value pairs

The find() can provide selected key/values

- The **find()** is a very powerful tool

```
MySQL  localhost:33060+ ssl  demo  JS > db.names.add( { name: "Jack", age: 11 } )
Query OK, 1 item affected (0.0116 sec)
MySQL  localhost:33060+ ss                > db.names.find().fields("name")
{
    "name": "Dave"
}
{
    "name": "Jack"
}
2 documents in set (0.0007 sec)
```

Exercise 1

Create a schema named '**fosdem**', a collection named '**a**', and add your name and another piece of information. What is needed for quoting keys and what is needed for quoting values?

Time - 10 minutes

Exercise 1 Review – How did you do?

A Very Quick Explanation of JSON objects and JSON arrays

- Objects are bounded by {}, arrays are bounded by [].
- You can have arrays in objects, objects in arrays, in any combo you want.
- You can embed down as many levels as you want but discretion is advised as they get harder to read as you add more.
- The MySQL JSON data type will hold roughly 1GB per column.

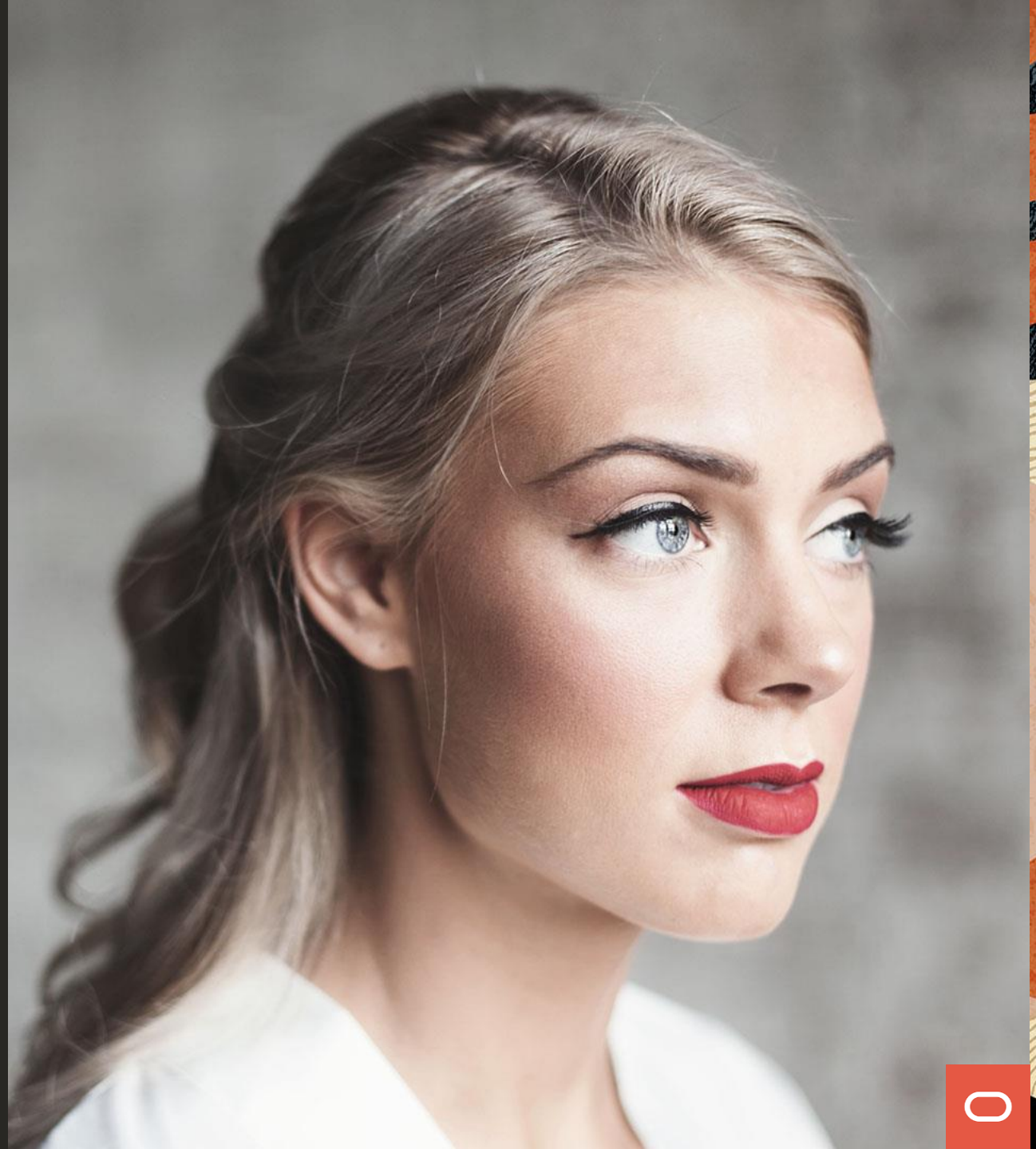
See https://www.json.org/json-en.html for more details on JSON.

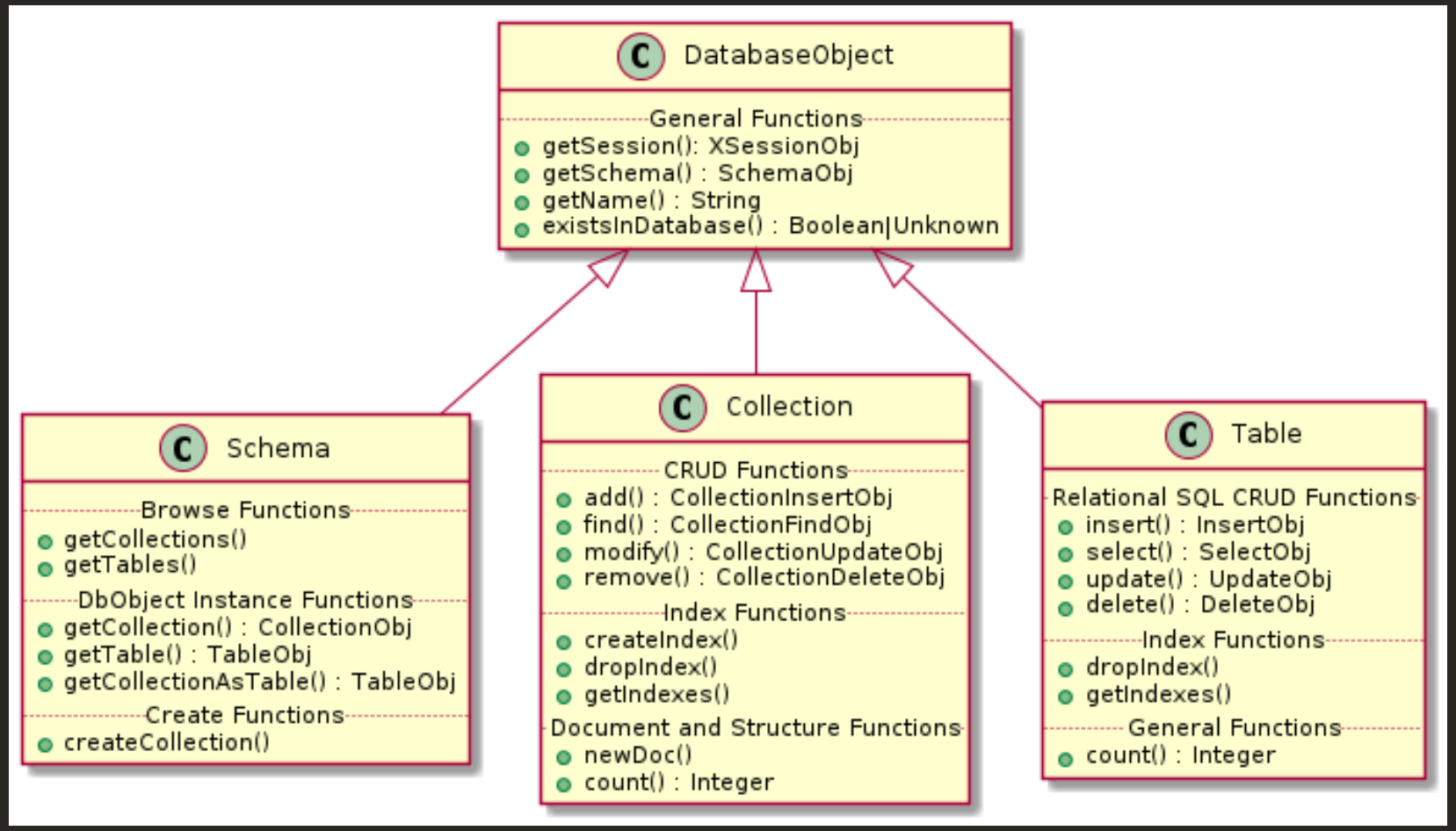
```
{
 "GNP": 249704,
 "_id": "BEL",
 "Name": "Belgium",
 "IndepYear": 1830,
 "geography": {
  "Region": "Western Europe",
  "Continent": "Europe",
  "SurfaceArea": 30518
 },
 "government": {
  "HeadOfState": "Albert II",
  "GovernmentForm": "Constitutional Monarchy, Federation"
 },
 "demographics": {
  "Population": 10239000,
  "LifeExpectancy": 77.80000305175781
 }
```


This is an example JSON document showing embedding

Ready for some technical stuff & a generic stock photo?

Then this is your *lucky* day!





The MySQL Document Store allows developers to work with SQL relational tables and schema-less JSON collections.

To make that possible MySQL has created the X Protocol and the X Dev API which puts a strong focus on CRUD by providing a fluent API allowing you to work with JSON documents in a natural way.

The X Protocol is a highly extensible and is optimized for CRUD as well as SQL API operations.

The X DevAPI is implemented by MySQL Shell and MySQL Connectors that support X Protocol.

The X DevAPI wraps several very powerful concepts in a simple API.

- A new high-level session concept enables you to write code that can transparently scale from single MySQL Server to a multiple server environment with InnoDB Cluster
- Read operations are simple and easy to understand.
- Non-blocking, asynchronous calls follow common host language patterns.

The X DevAPI introduces a new, modern and easy-to-learn way to work with your data.

- Documents are stored in Collections and have their dedicated CRUD operation set.
- Work with your existing domain objects or generate code based on structure definitions for strictly typed languages.
- Focus is put on working with data via CRUD operations.
- Modern practices and syntax styles are used to get away from traditional SQL-String-Building.

An X DevAPI session is a high-level database session concept that is different from working with traditional low-level MySQL connections.

Sessions can encapsulate one or more actual MySQL connections when using the X Protocol.

Use of this higher abstraction level decouples the physical MySQL setup from the application code. Sessions provide full support of X DevAPI and limited support of SQL.

So what is wrong with good ol' SQL?!?!

Descriptive Language

Impedance Mismatch

ORMS and other similar problems

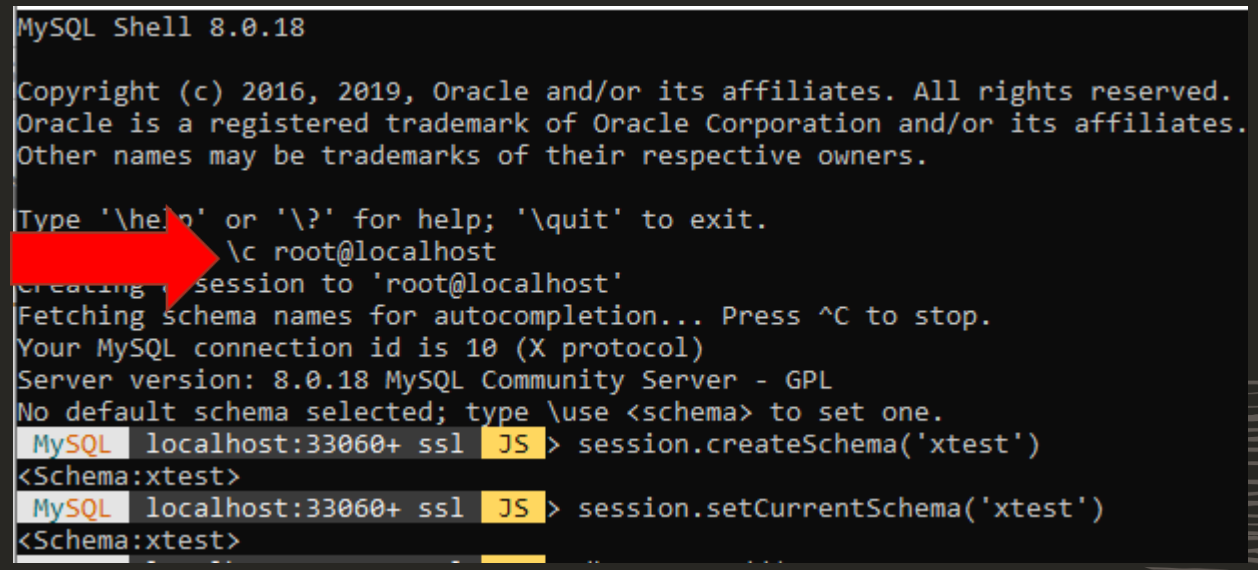
A New Port!

The X DevAPI listens on port 33060 instead of the older protocols 3306.

MySQL 8.0 has the X DevAPI enabled by default and it is optional on MySQL 5.7.

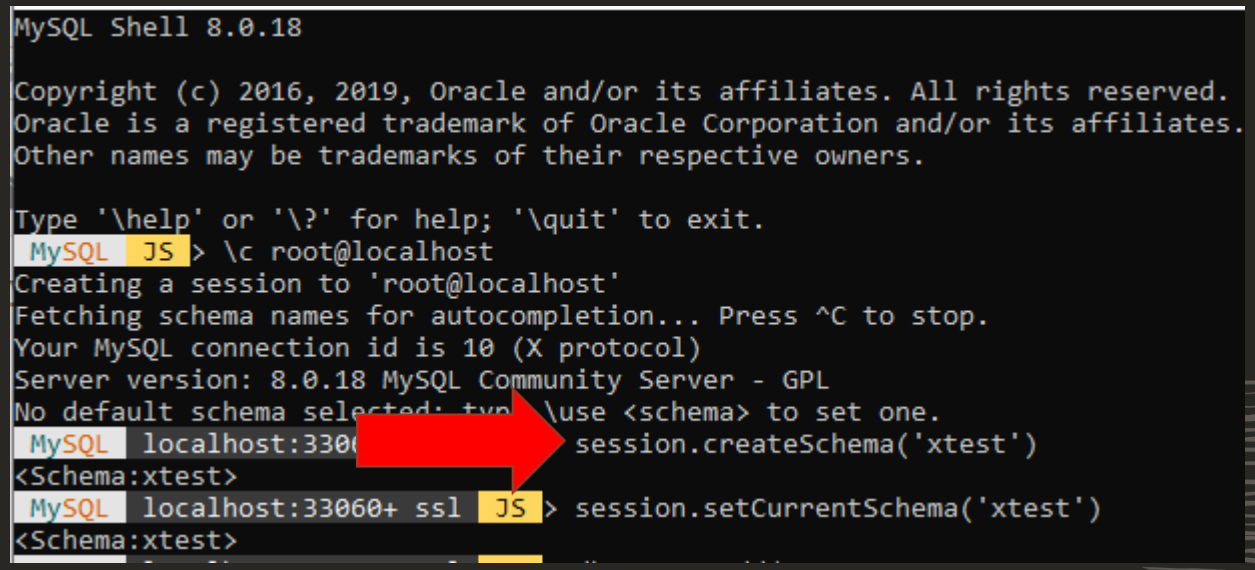
Creating a Session
Page 9

- To start using the MySQL Document store you have to create a session by logging into the server.

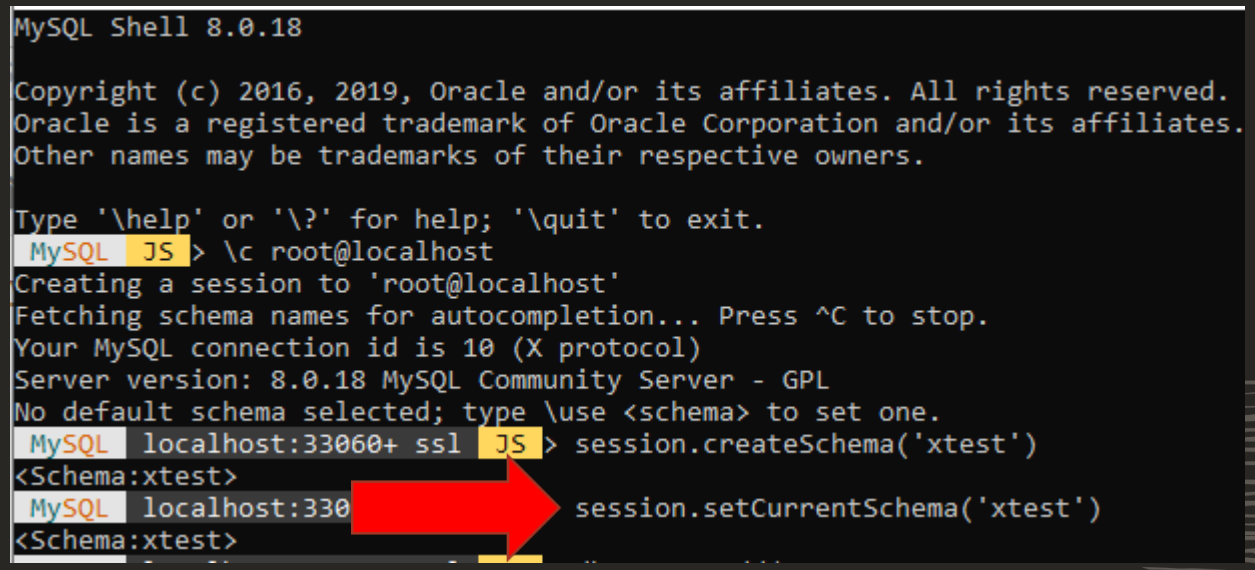


Creating a Schema

- Creating a schema is done with createSchema()



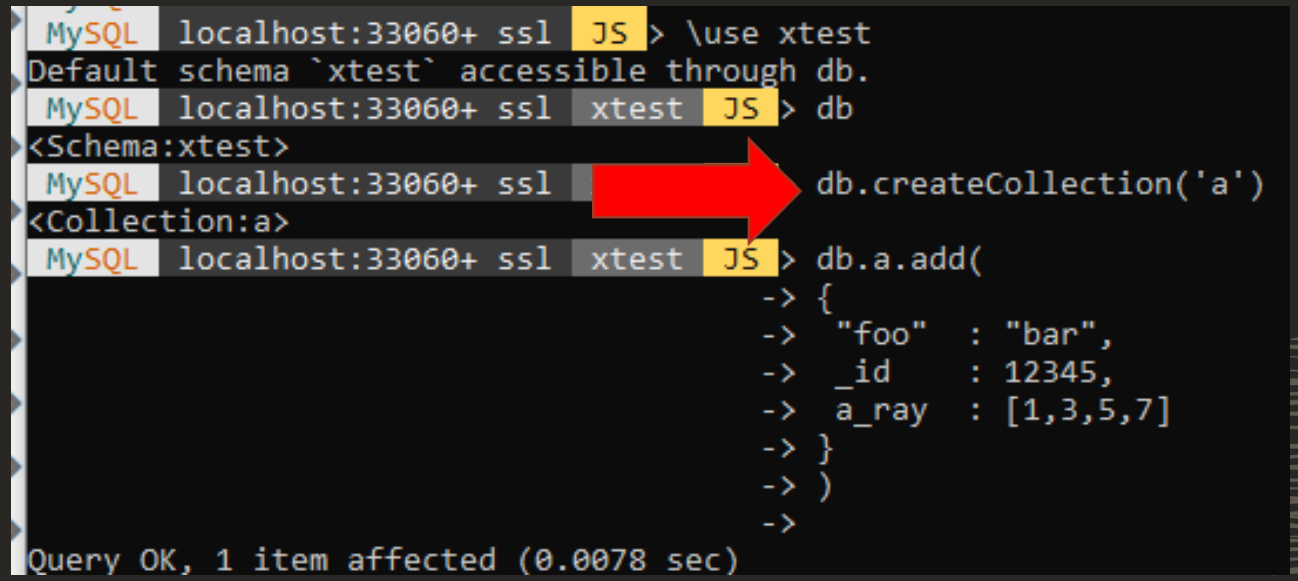
Pointing to a Schema



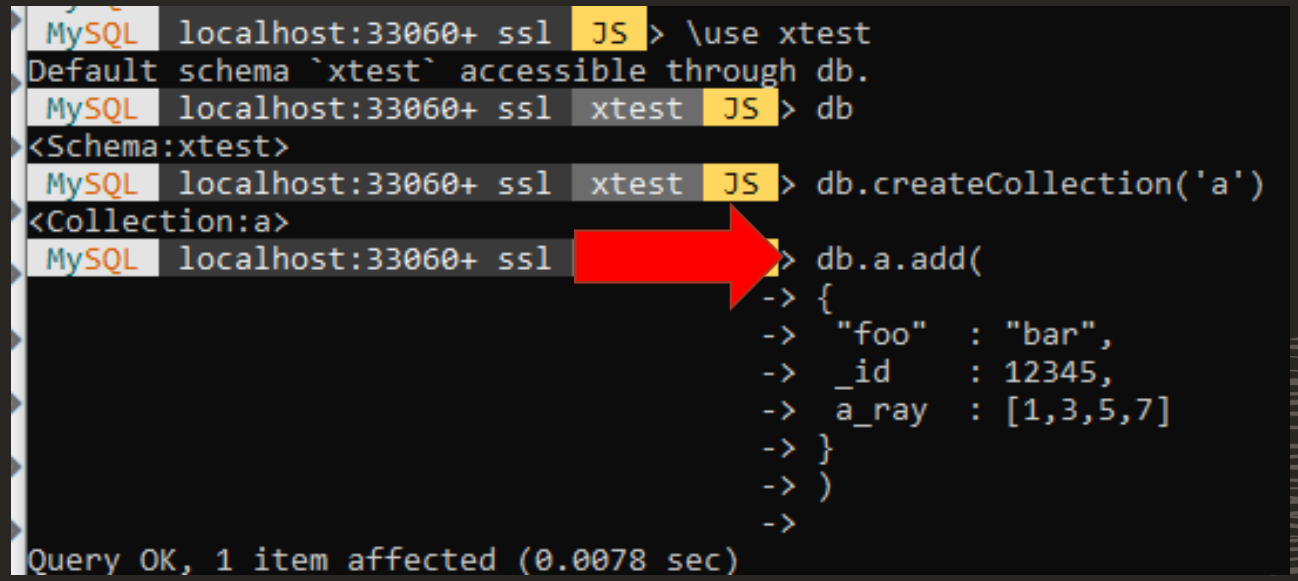
- To get the session to point to the new schema we can use \USE xtest or session.setCurrentSchema('xtest')

Create a Collection

- So let us create a new collection very creatively named ‘a’



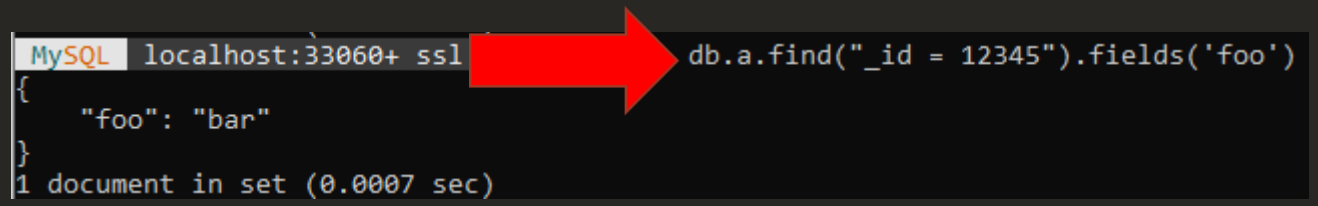
Create a Collection



- And add a document
- **Note:** the **_id** field is specified and there is an array

And let us find that particular record

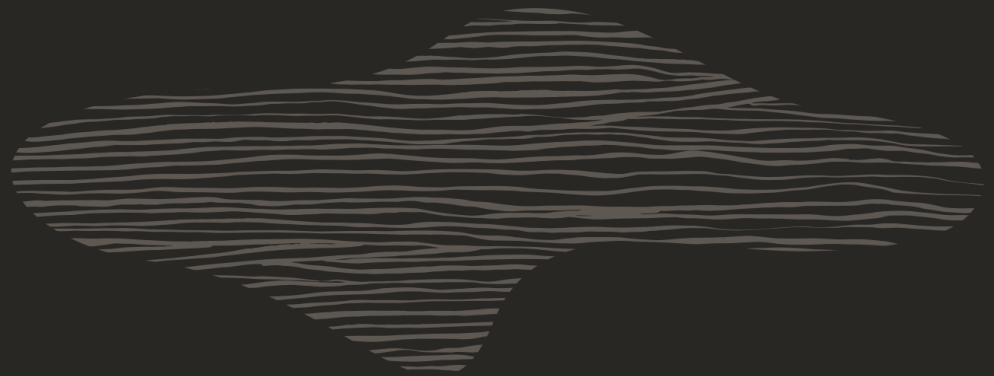
- We can specify the search parameters by putting them into the find()



- We can use modify() to add additional key/values to the document

And let us modify that particular record

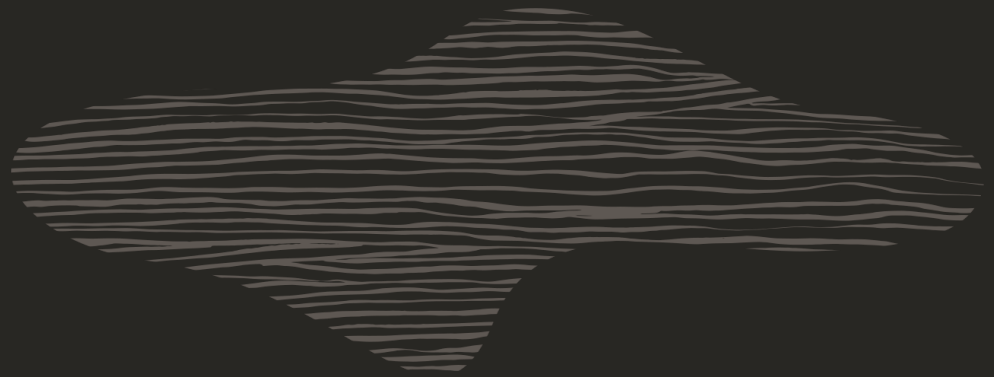
```
MySQL  localhost:33060+ ssl                > db.a.modify("_id = 12345").set('Location:', { city: 'Brussels', CountryCode: 'BEL'})
Query OK, 1 item affected (0.0046 sec)

Rows matched: 1  Changed: 1  Warnings: 0
```



And let us check that modification

```
MySQL  localhost:33060+ s          db.a.find("_id = 12345").fields('foo','Location')
{
    "foo": "bar",
    "Location": {
        "city": "Brussels",
        "CountryCode": "BEL"
    }
}
```

- Note that we are specifying two fields in this illustration



Exercise 2

Create a schema named **demox**, create a document collection in that schema named **a**, and create a document with your name and postal code as data.

Then modify the document to another key/value pair with shoeSize as the key and your shoe size as the data.

Time - 15 minutes

Exercise Review

What was your experience?

Group Exercise
Page 11
using ‘a’ collectio

Does db.a.find("_id = 12345").fields('foo',"Location") work with mixed single and double quotes?

What if you run db.a.modify("_id = 12345").set('location:', { city: 'Brussels', CountryCode: 'BEL'}) -- do you get two key/values or is it case insensitive?

.

Time - 15 minutes

CRUD operations are available as methods, which operate on Schema objects. The available Schema objects consist of Collection objects containing Documents, or Table objects consisting of rows and Collections containing Documents.

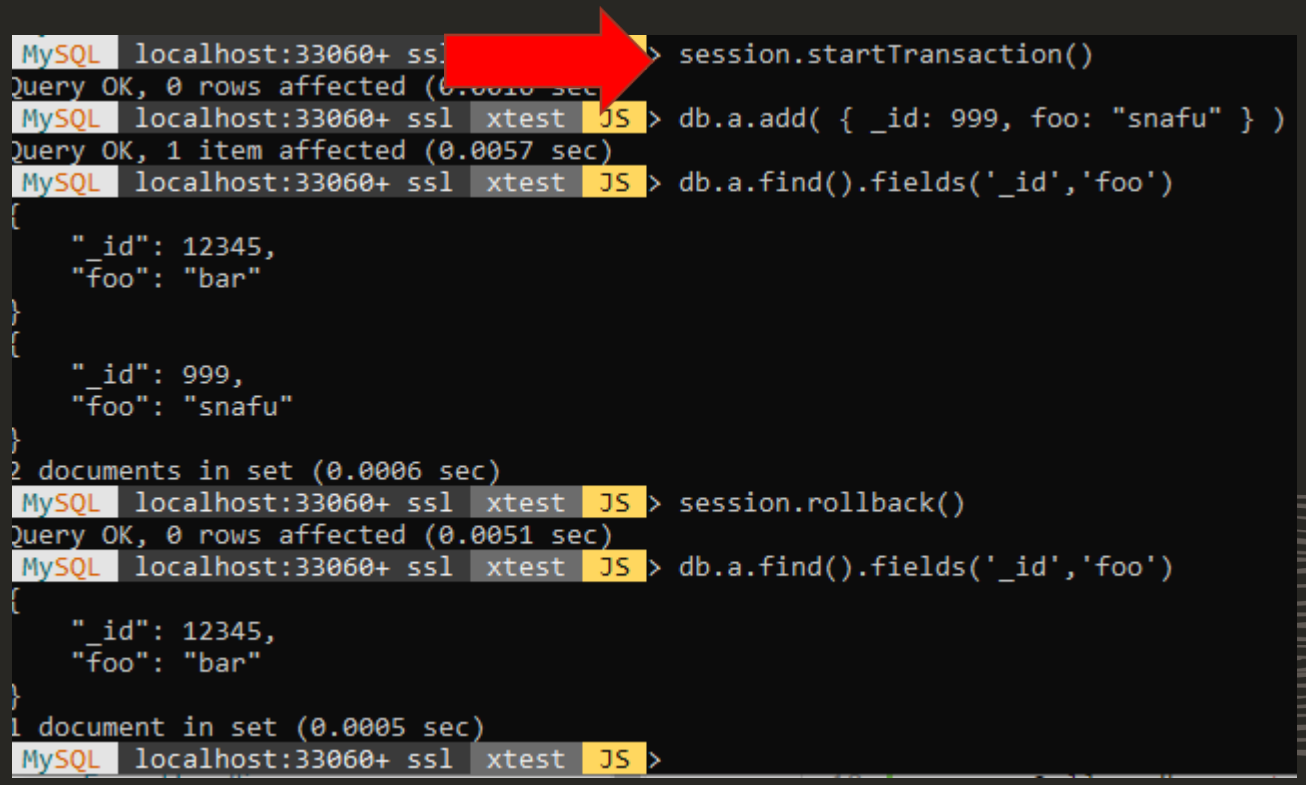
Operation	Document Store	Relational
Create	Connection.add()	Table.insert()
Read	Connection.find()	Table.select()
Update	Connection.modify()	Table.update()
Delete	Connection.remove()	Table.delete()

POP Quiz

What is the query for deleting a document from a collection named ‘z’ in a schema named ‘cars’ where the _id = 123?

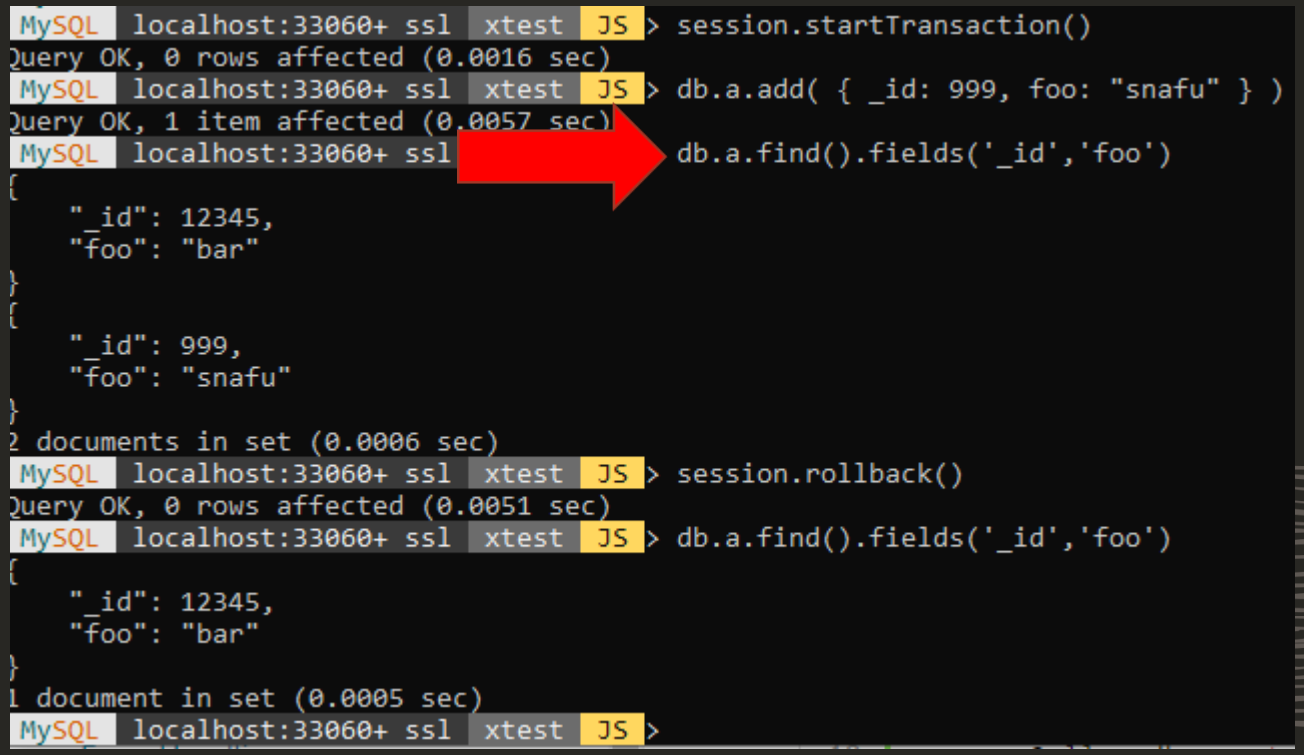
Transactions
Page 12

- Lets add another document to the ‘a’ collection



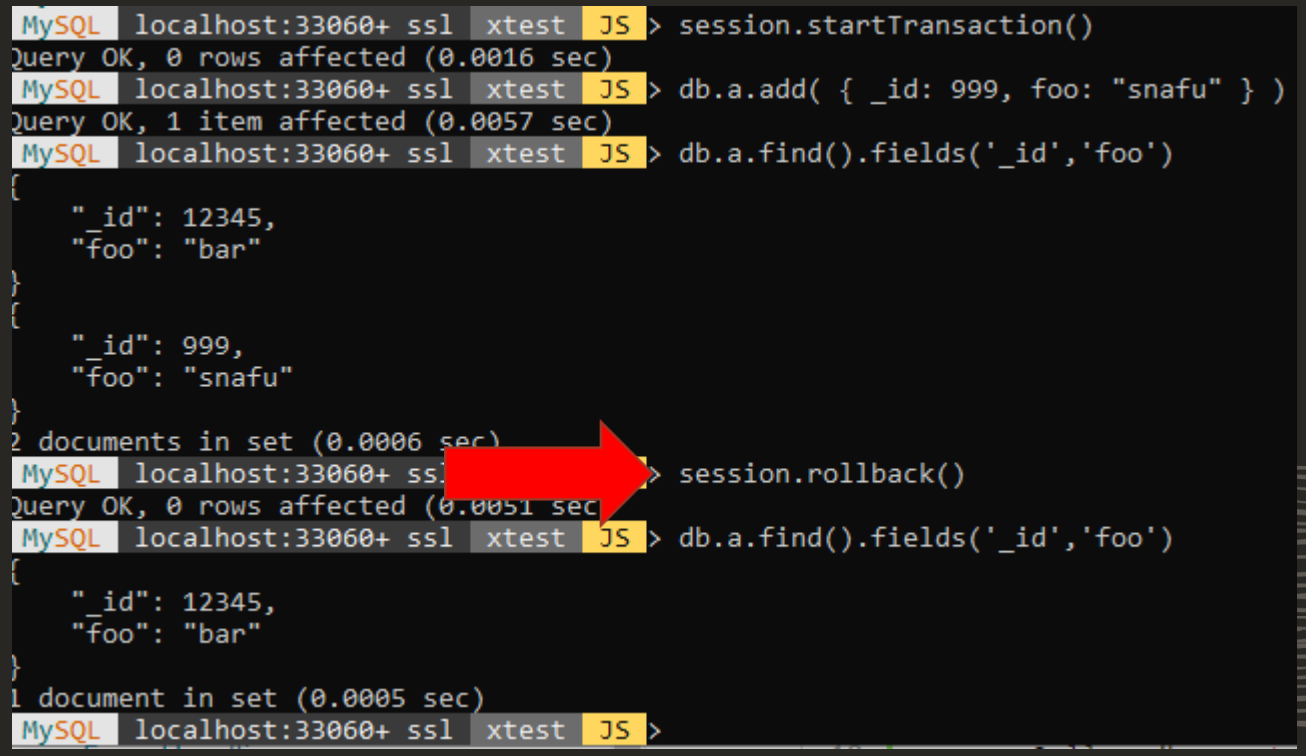
Transactions
Page 12

- Then we will use find() to check that that new document is in the collection



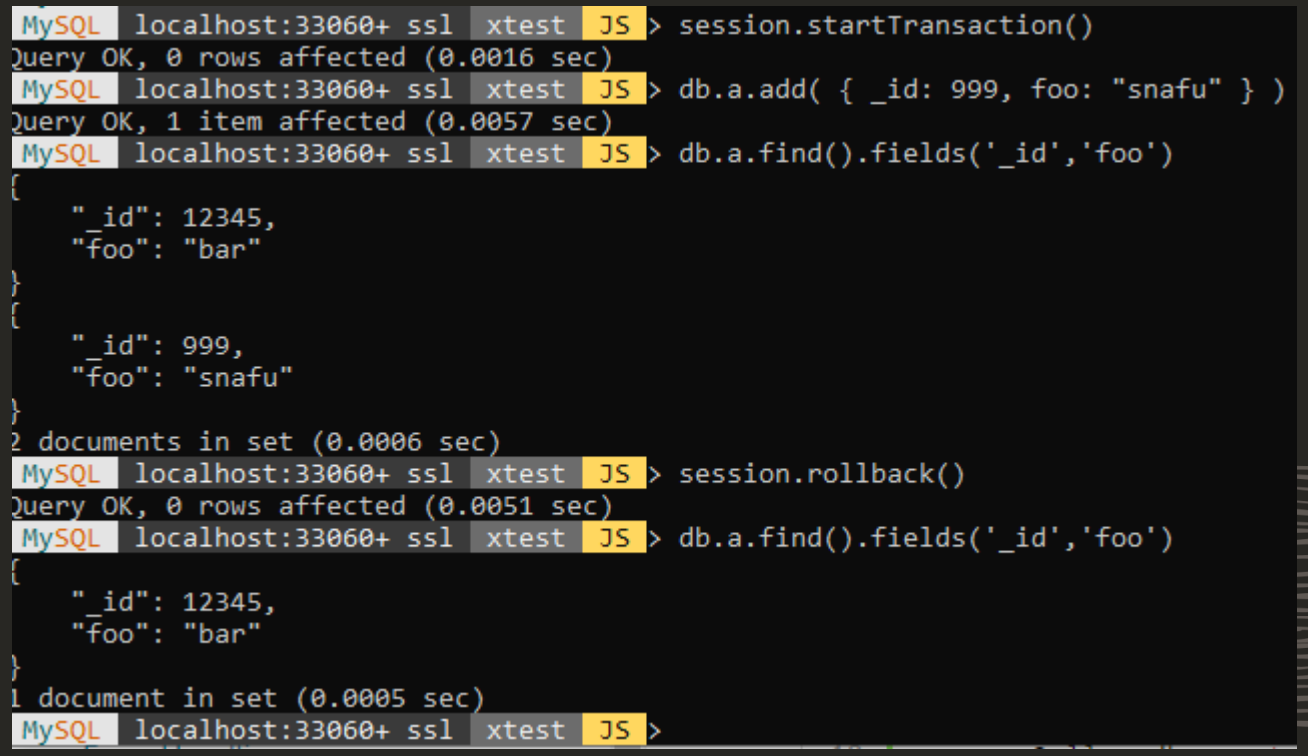
Transactions
Page 12

- Issuing a rollback removes the new document



Transactions
Page 12

- Any guesses as to what session.commit() does?



```
MySQL  localhost:33060+ ssl  world_x  JS > db.countryinfo.find().fields(['Name','geography.Region','IndepYear']).limit(3)
{
    "Name": "Aruba",
    "IndepYear": null,
    "geography.Region": "Caribbean"
}
{
    "Name": "Afghanistan",
    "IndepYear": 1919,
    "geography.Region": "Southern and Central Asia"
}
{
    "Name": "Angola",
    "IndepYear": 1975,
    "geography.Region": "Central Africa"
}
3 documents in set (0.0005 sec)
MySQL  localhost:33060+ ssl  world_x  JS > db.countryinfo.find("IndepYear > 1900").fields(['Name','geography.Region','IndepYear']).limit(3)
{
    "Name": "Afghanistan",
    "IndepYear": 1919,
    "geography.Region": "Southern and Central Asia"
}
{
    "Name": "Angola",
    "IndepYear": 1975,
    "geography.Region": "Central Africa"
}
{
    "Name": "Albania",
    "IndepYear": 1912,
    "geography.Region": "Southern Europe"
}
3 documents in set (0.0007 sec)
MySQL  localhost:33060+ ssl  world_x  JS >
```

Find things with find()

```
MySQL  localhost:33060+ ssl  world_x  JS > db.countryinfo.find("geography.Region = 'North America'").fields(['Name','geography.Region','IndepYear']).sort(['Name','IndepYear DESC'])
{
    "Name": "Bermuda",
    "IndepYear": null,
    "geography.Region": "North America"
}
{
    "Name": "Canada",
    "IndepYear": 1867,
    "geography.Region": "North America"
}
{
    "Name": "Greenland",
    "IndepYear": null,
    "geography.Region": "North America"
}
{
    "Name": "Saint Pierre and Miquelon",
    "IndepYear": null,
    "geography.Region": "North America"
}
{
    "Name": "United States",
    "IndepYear": 1776,
    "geography.Region": "North America"
}
5 documents in set (0.0012 sec)
MySQL  localhost:33060+ ssl  world_x  JS >
```

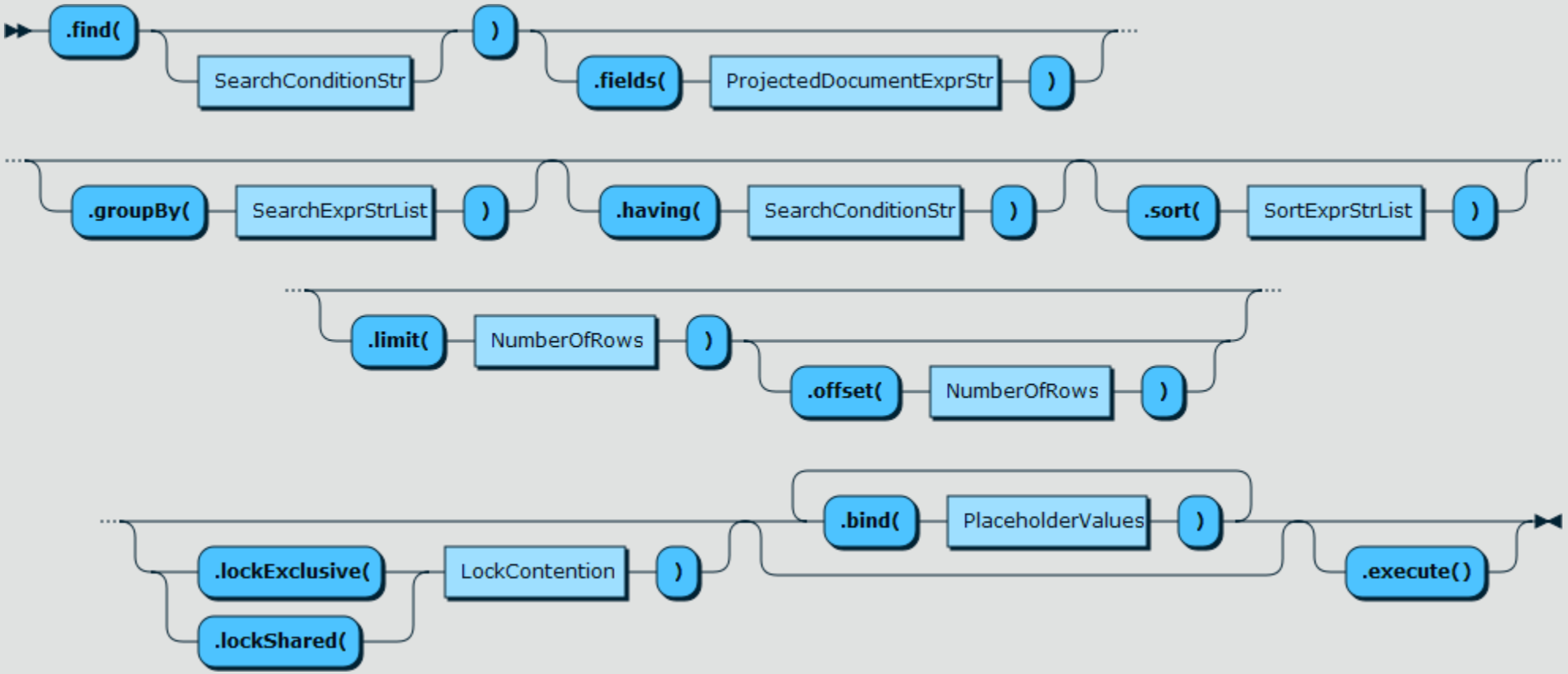
Find things with find()

What is the _id?

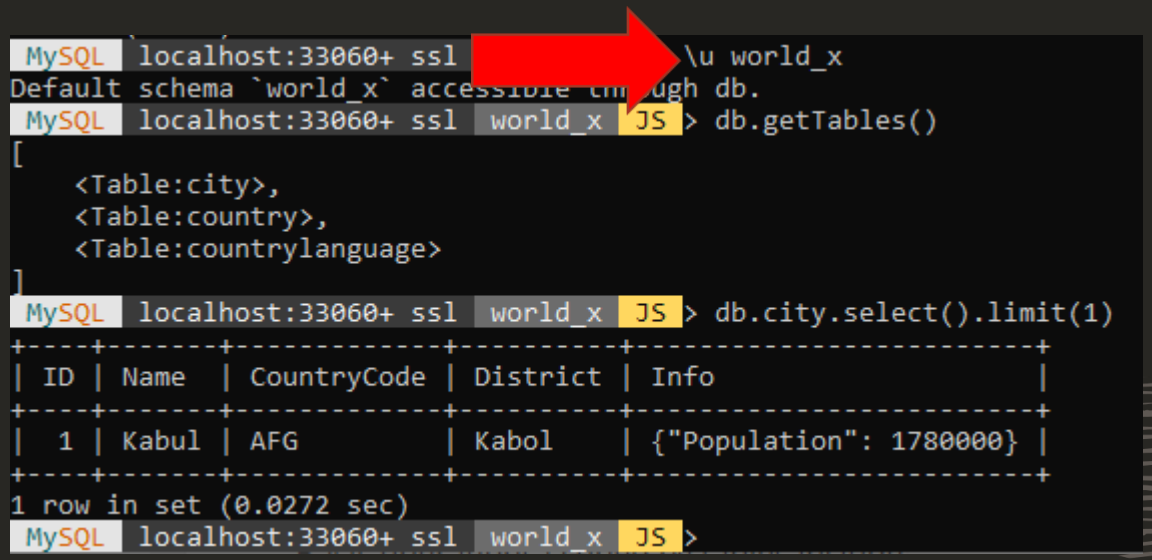
- The InnoDB used the _id field as a primary key for efficiency and, unlike other keys in the document, can not be changed. The document store will automatically assign a value but you can assign your own value but you need to make sure you do not duplicate these values or the server will complain. The _id number made of a 4 byte prefix, an 8 byte timestamp of the startup time of the server instance, and a 16 byte a per-instance automatically incremented integer value, which is hex encoded and has a range of 0 to 2**64-1. The initial value of serial is set to the auto_increment_offset system variable, and the increment of the value is set by the auto_increment_increment system variable.

- Users of multi-primary Group Replication or InnoDB cluster can depend on inserts to the same table from different instances do not have conflicting primary key values; assuming that the instances have the auto_increment_* system variables configured properly.

EBNF's are in the workbook

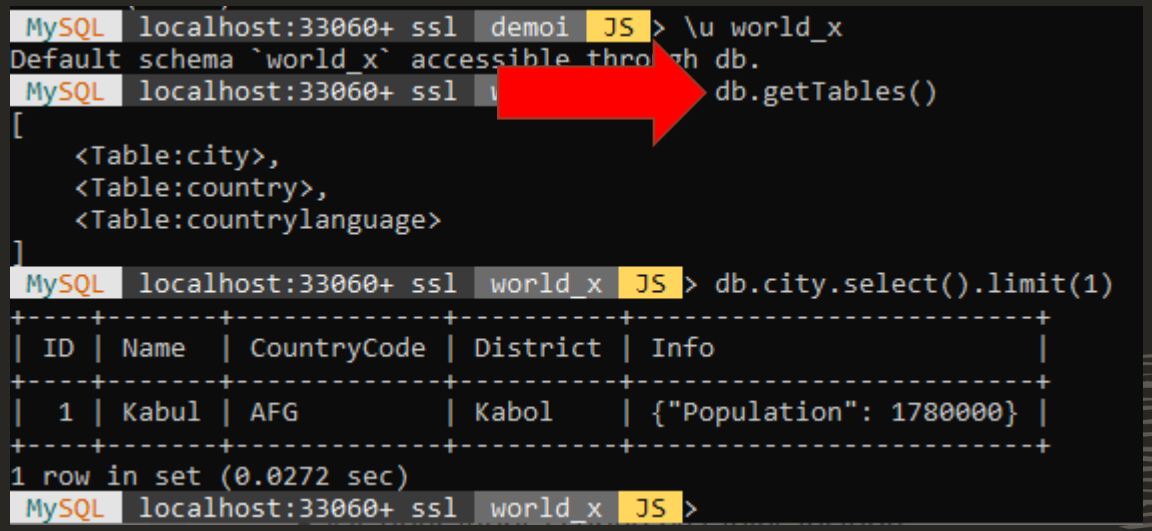


Tables
Page 22



- Yes you can access your relational tables from the MySQL Document Store too!

Tables
Page 22



- Where you can use db.getCollections() to find document collections, you can use db.getTables() to see the available tables!

Tables
Page 22

- But you use select() instead of find() to query

```
 MySQL  localhost:33060+ ssl  demoi  JS > \u world_x
Default schema `world_x` accessible through db.
 MySQL  localhost:33060+ ssl  world_x  JS > db.getTables()
[
    <Table:city>,
    <Table:country>,
    <Table:countrylanguage>
]
 MySQL  localhost:33060+ ssl                JS > db.city.select().limit(1)
+----+-------+-------------+----------+-------------------------+
| ID | Name  | CountryCode | District | Info                    |
+----+-------+-------------+----------+-------------------------+
|  1 | Kabul | AFG         | Kabol    | {"Population": 1780000} |
+----+-------+-------------+----------+-------------------------+
1 row in set (0.0272 sec)
 MySQL  localhost:33060+ ssl  world_x  JS >
```

Exercise Final

Exercise Final

How many documents are in the world_x schema matching the CountryCode = ‘BEL’?

db.city.select().where("CountryCode = 'BEL'")

What is missing?

- **Currently you can not do joins with the api**
  - (use the SQL())

Resources

- Using MySQL as a Document Store

  https://dev.mysql.com/doc/refman/8.0/en/document-store.html
- Documentation on X Devapi

  https://dev.mysql.com/doc/
- Connectors – C++, Java, JavaScript, .Net, Node.JS, Python, PHP

  https://dev.mysql.com/downloads/
- Books

  Introducing the MySQL Document Store – Dr. Charles Bell, MySQL & JSON – A Practical Programming Guide – Dave Stokes, MySQL Connector Python – Jesper Wisborg Krogh

Wrap up

Your reaction

David.Stokes@Oracle.com

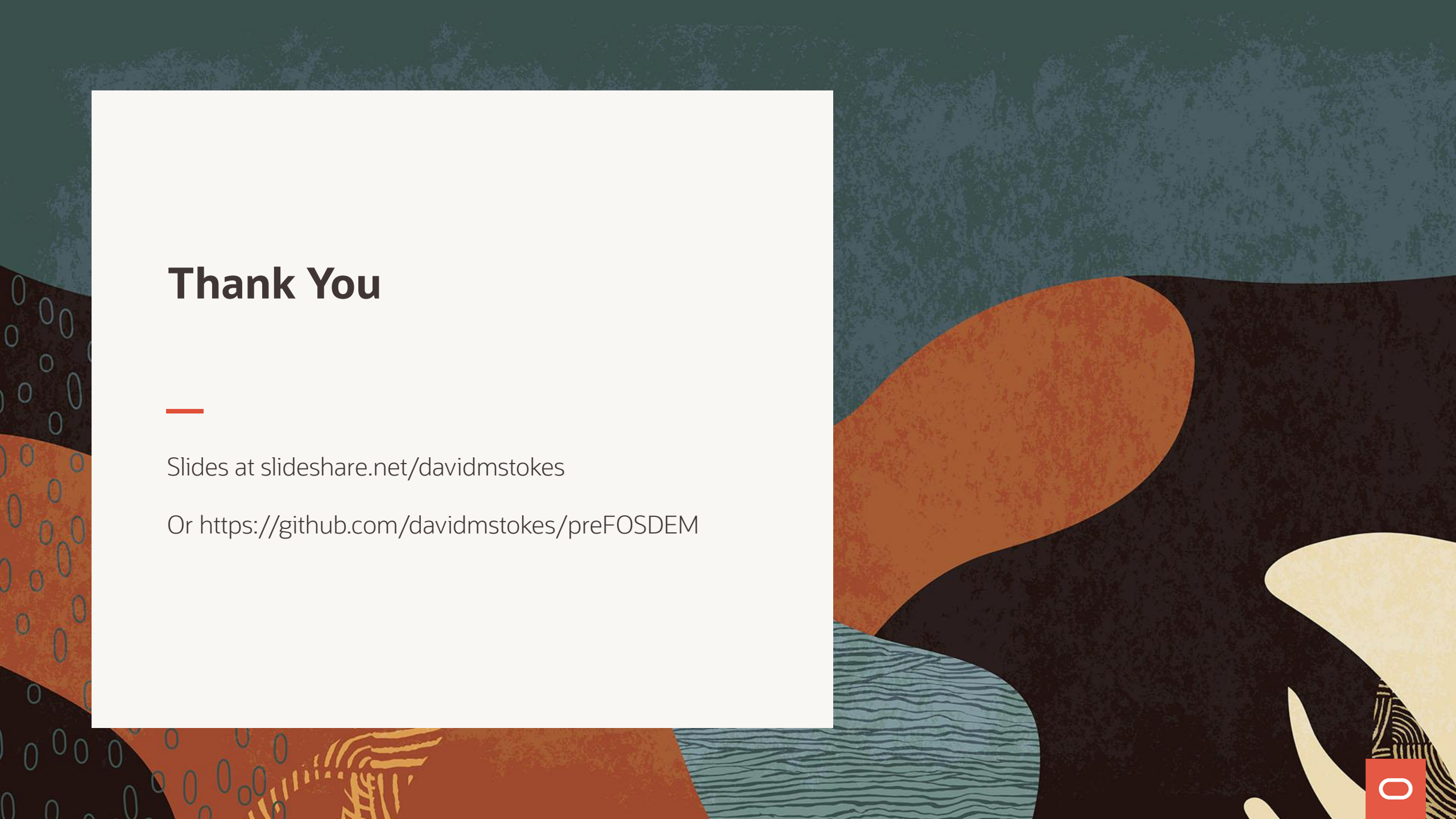
Reminder!

MySQL 5.6 end of life is February 5, 2021

Thank You

Slides at slideshare.net/davidmstokes

Or https://github.com/davidmstokes/preFOSDEM



Safe Harbor

The preceding 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, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.