Node.js and the MySQL Document Store

Live for the Code

Rui Quelhas
Software Developer
Oracle, MySQL
October 24, 2018

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

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

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

whoami

Rui Quelhas

MySQL Middleware and Clients team (Connectors)

Connector/Node.js Lead Developer

rui.quelhas@oracle.com

@ruiquelhas

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Outline

1. NoSQL and MySQL
2. MySQL Document Store API
3. MySQL Connector/Node.js
4. Final thoughts

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

NoSQL and MySQL

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

SQL, NoSQL, NewSQL, whatever

- Moving to NoSQL because:
  - RDBMS “do not scale” (High-availability, yada yada)
  - “Schemaless is agile” and developers don’t like SQL
- Only to later bump head-first into its limitations (lack of referential integrity, ACID is gone...)
- Some outcomes:
  1. NoSQL on top of SQL database engines (**Schemaless**)
  2. “New” sort-of hybrid database engines (**MyRocks**/**RocksDB**)
  3. Use and abuse of JSON columns
  4. Re-invent the wheel (**NewSQL**)

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

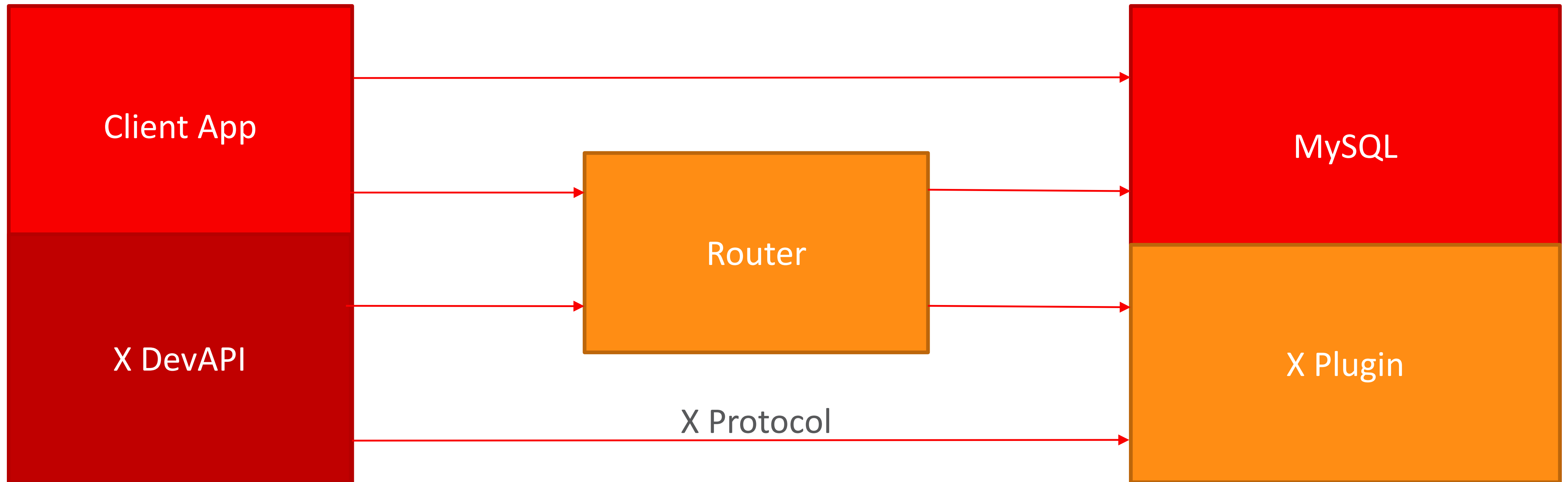
MySQL Document Store

- It’s now possible to save and retrieve unstructured data using a MySQL database (not talking about raw JSON)
- Data lives in a JSON column but everything is abstracted way from the user, which only deals with documents
- No need to worry about schemas and data types
- Keeping logical consistency and ACID (it’s MySQL after all)
- At the same time, a schema can still mix in traditional tables
- ...and more https://lefred.be/content/top-10-reasons-for-nosql-with-mysql/

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Component Overview



https://dev.mysql.com/doc/refman/8.0/en/document-store.html

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

X DevAPI [DEV5985]

- High-level database API to develop modern applications powered by **InnoDB Cluster** [TIP3065]
- Application gateway for the underlying document-store infrastructure (X Protocol, Router, X Plugin)
- Off-the-shelf support for **CRUD NoSQL** document operations
- Lower-level management via raw SQL and other advanced features
- Available in connectors for the most popular languages as well as the brand new MySQL Shell
- Documentation available at https://dev.mysql.com/doc/x-devapi-userguide/en/

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Router

- MySQL Router is an integral part of **InnoDB Cluster**
- Lightweight middleware that provides transparent routing between an application and back-end MySQL servers
- Can be used for a wide variety of use cases, but mainly to address high availability and scalability concerns
- Runs as a standalone component
- Packaged with the MySQL server for the usual platforms
- Documentation available at https://dev.mysql.com/doc/mysql-router/8.0/en/

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

X Plugin

- New MySQL server plugin (developed internally)
- Installed and enabled by default on MySQL 8.0 series
- Provides the document store infrastructure on top of plain MySQL servers
- Uses a different set of ports (**33060** instead of the usual MySQL **3306**)
- Implements a client-server protocol based on Google Protocol Buffers
- Accepts X DevAPI client requests and processes them using the core MySQL engine
- Documentation available at https://dev.mysql.com/doc/refman/8.0/en/document-store.html

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

X Protocol

- New client-server protocol based on Google Protocol Buffers (easier for **ANYONE** to extend and improve)
- Content awereness
  - CRUD expression trees to establish operation boundaries
  - Expectations to define conditions and create pipelined statements
  - Placeholder assignment and value escaping
- Security baked in
  - SSL/TLS by default
  - No information leaked to unauthenticated users (e.g. server version)
- Open source at https://dev.mysql.com/doc/internals/en/x-protocol.html

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

X Protocol

Message example

```
Mysqlx.Crud.Find {
  collection { name: "collection_name", schema: "schema_name" }
  data_model: DOCUMENT
  criteria {
    type: OPERATOR
    operator {
      name: "=="
      param {
        type: IDENT,
        identifier { name: "_id" }
      }
      param {
        type: LITERAL,
        literal {
          type: V_STRING,
          v_string: { value: "some_string" }
        }
      }
    }
  }
}
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Documents and Collections

- Documents in the MySQL Document store are just JSON
  - “JavaScript Object Notation”
  - Cross-platform serialization format (common for web services and applications)
  - Standardized as ECMA-404 (http://json.org)
  - Supported by a proper native MySQL data type
- Multiple documents are stored inside a Collection
  - Technically, an **InnoDB** table
  - One regular column of type JSON
  - Virtual columns on top for indexes

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

What does it look like?

SHOW CREATE TABLE

```
CREATE TABLE `docStoreCollection` (
  `doc` json DEFAULT NULL,
  `_id` varbinary(32) GENERATED ALWAYS AS
    (json_unquote(json_extract(`doc`,_utf8mb4'$._id'))) STORED NOT NULL,
  PRIMARY KEY (`_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

What does it look like?

DESCRIBE/SHOW INDEXES

Field	Type	Null	Key	Default	Extra
doc	json	YES		NULL	
_id	varbinary(32)	NO	PRI	NULL	STORED GENERATED

INDEX_NAME	COLUMN_NAME	INDEX_TYPE	IS_VISIBLE
PRIMARY	_id	BTREE	YES

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

MySQL Document Store API

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Schemaless Data

- X DevAPI introduces a modern CRUD API to work with schemaless data
- Semantic Document Store methods that allow to manage data without writing a single line of SQL

```
> users.add({ name: 'Rui' }).add({ name: 'Johannes' })
Query OK, 2 items affected (0.0373 sec)

> users.find()
[
    {
        "_id": "00005b50ced40000000000000001",
        "name": "Rui"
    },
    {
        "_id": "00005b50ced40000000000000002",
        "name": "Johannes"
    }
]
2 documents in set (0.0009 sec)
```

```
> users.modify('true').set('team', 'nodejs')
Query OK, 2 items affected (0.0751 sec)

> users.find('name = "Rui"')
[
    {
        "_id": "00005b50ced40000000000000001",
        "name": "Rui",
        "team": "nodejs"
    }
]
1 document in set (0.0026 sec)
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Document Store API

- DDL API to create new collections and list or remove existing ones
- CRUD API to insert, retrieve, update or remove documents in a collection
- Operate on multiple docs at the collection level or on individual docs
- Additional extensions: projections, aggregation, ordering, limits and skipping results
- Index manapgement API with support for regular and **SPATIAL** indexes
- Transactional API to begin, rollback or commit transactions as well as managing intermediate savepoints
- Document field locking API for granular consistency management

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Available Methods

Schema	Collection (multi-doc)	Collection (single-doc)	Session
• createCollection() • dropCollection()	• add() • find() • modify() • remove() • createIndex() • dropIndex()	• getOne() • addOrReplaceOne() • replaceOne() • removeOne()	• startTransaction() • rollback() • commit() • setSavepoint() • releaseSavepoint() • rollbackTo()

- Operational methods return proper database object or operation instances
- CRUD operations: **CollectionAdd**, **CollectionFind**, **CollectionModify**, **CollectionRemove**
- Operation boundaries - filtering criteria, computed projections, placeholder assignment, resultset ordering - established using an expression language

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Expression Strings

- Small subset of the SQL language
- Easy to grasp (everyone already knows SQL) but still powerful
- Expressive and human-readable
- Common standard between all the official connector implementations

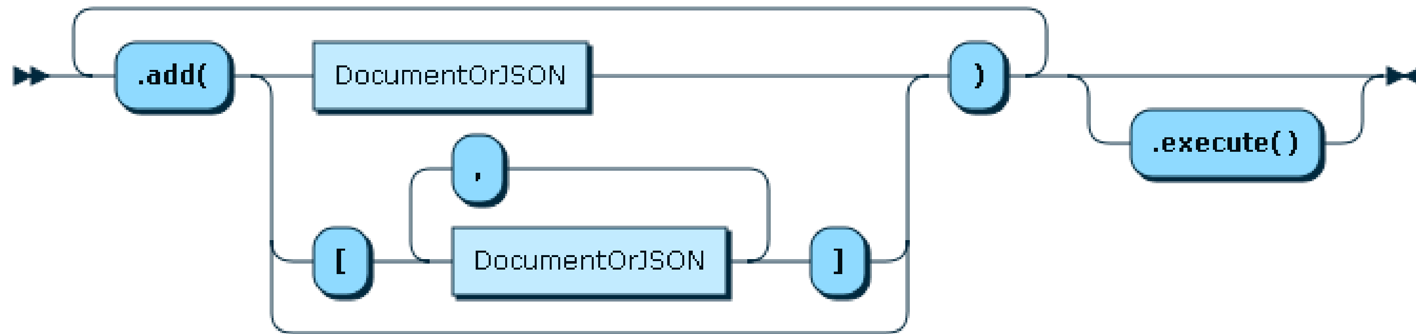
```
// JavaScript
collection
  .find("name = 'foo' AND age > 42")
  .fields("name", "age")
  .groupBy("name", "age")
  .sort("name ASC", "age DESC")
  .limit(4)
  .offset(2)
  .execute()
```

```
// Java
collection
  .find("name = 'foo' AND age > 42")
  .fields("name", "age")
  .groupBy("name", "age")
  .sort("name ASC", "age DESC")
  .limit(4)
  .offset(2)
  .execute()
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

CollectionAdd



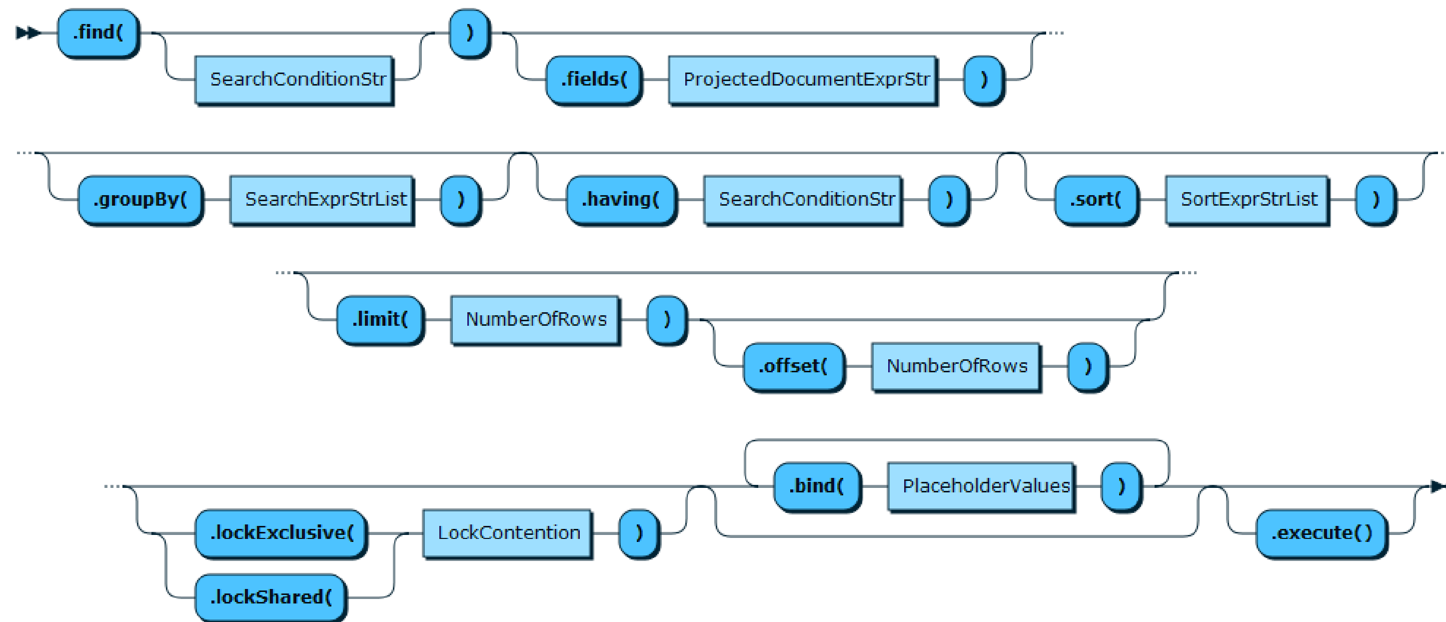

```
collection.add({ name: 'foo', age: 42 })
  .add({ name: 'bar', age: 23 })
  .execute()

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

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

CollectionFind

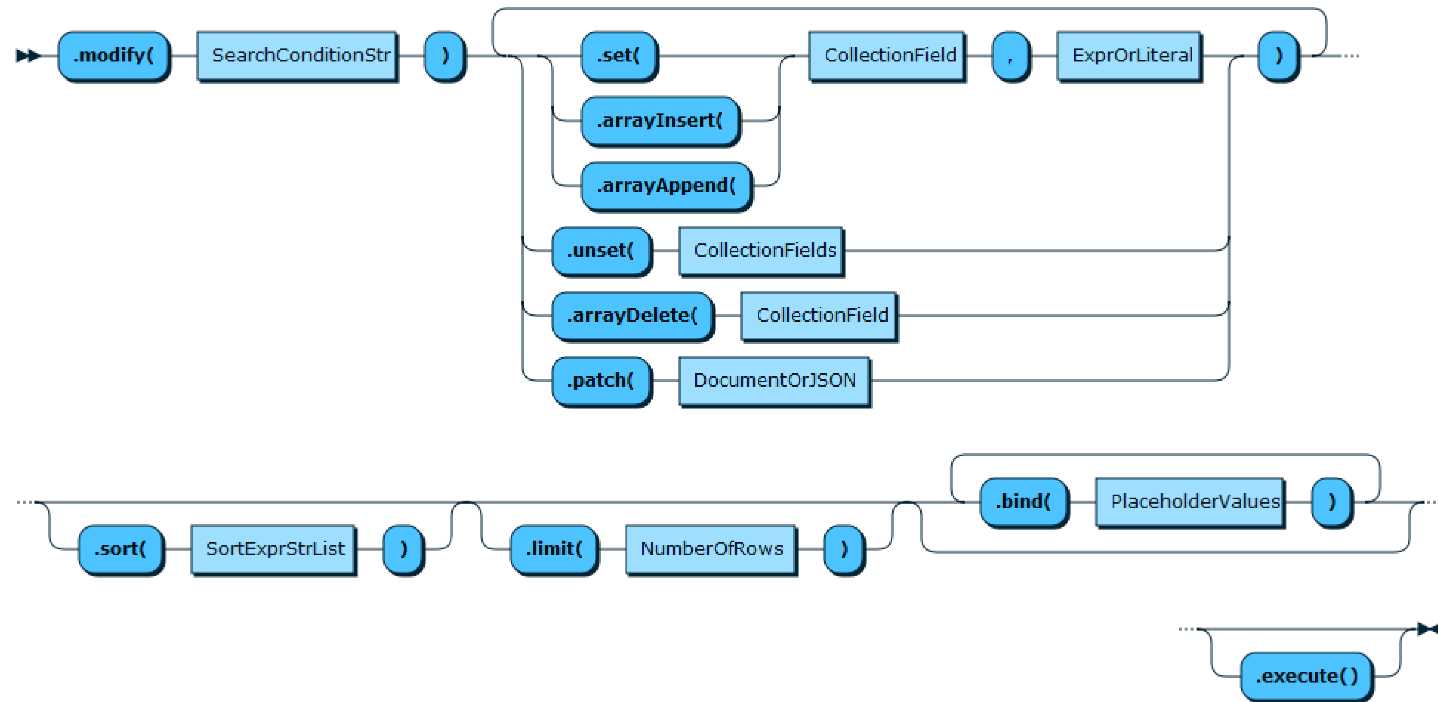



```
collection.find('name = :name')
  .bind('name', 'foo')
  .fields('COUNT(age) AS age')
  .groupBy('age')
  .having('age > 42')
  .sort('age DESC')
  .limit(10)
  .offset(5)
  .execute()
```

ORACLE®

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

CollectionModify



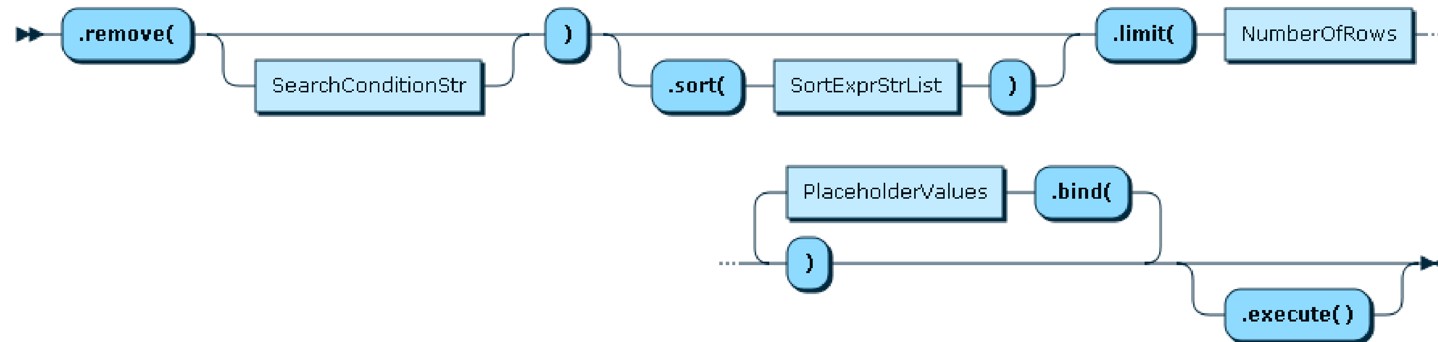

```
collection.modify('name = :name')
  .bind('name', 'foo')
  .set('age', 42)
  .sort('name ASC')
  .limit(1)
  .execute()

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

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

CollectionRemove




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

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

MySQL Connector/Node.js

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Overview

- Only Node.js driver for MySQL with out-of-the-box support for MySQL 8.0 series and, particularly, the Document Store
- Open source, available on GitHub and on npm
- Modern Node.js asynchronous API using ***Promise***s
- Greenfield X Protocol implementation leveraging the X DevAPI spec
- Document Store CRUD API
- Secure connections and multiple authentication mechanisms
- Connection pooling and failover
- Support for raw SQL and relational query builder as well

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Asynchronous API

- Operations are sent to the server when the ***execute()*** method is called
  - receives an optional query operation callback push-based cursor) which runs for each element in the result set
  - returns a ***Promise*** which resolves to a ***Result*** instance containing details and metadata about the operation, or fails with an ***Error*** (can be used with ***async/await)***

```
collection
  .find()
  .execute(doc => {
    // do something with the document
  })
  .then(result => {
    // result contains details about the operation
  })
  .catch(err => {
    // handle any errors
  })
```

```
collection
  .remove()
  .execute(/* no cursor will be needed */)
  .then(result => {
    // result contains details about the operation
  })
  .catch(err => {
    // handle any errors
  })
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Up and running

```
$ npm install --save --save-exact @mysql/xdevapi
```

```
const mysqlx = require('@mysql/xdevapi');
```

```
const session = await mysqlx.getSession({ user: 'root' });
const schema = session.getSchema('codeone2018');

if (!(await schema.existsInDatabase())) {
  await session.createSchema('codeone2018');
}

await schema.createCollection('sessions', { ReuseExistingObject: true });

const collections = await schema.getCollections();
console.log(collections); // [{ schema: 'codeone2018', collection: 'sessions' }]
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Adding new documents

```
// const collection = schema.getCollection('sessions');
await collection.add({ _id: 'DEV5986', title: 'This talk!', speakers: ['Rui Quelhas'] }).execute();

const batchA = collection
  .add({ _id: 'DEV5985', title: 'MySQL Connector/Node.js and the X DevAPI', speakers: ['Rui Quelhas'] })
  .add({ _id: 'DEV6233', title: 'Connector/J Beyond JDBC...', speakers: ['Filipe Silva'] });

const batchB = collection
  .add([{
    _id: 'DEV5959',
    title: 'Python and the MySQL Document Store',
    interactive: false,
    speakers: ['Jesper Krogh'],
    topics: ['python']
  }, {
    _id: 'HOL1706',
    title: 'Developing Modern Applications with the MySQL Document Store and Node.js',
    interactive: true,
    speakers: ['Jesper Krogh', 'Lig Isler-turmelle'],
    topics: ['node.js']
  }]);

await Promise.all([batchA.execute(), batchB.execute()]);
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Retrieving documents

```
// const collection = schema.getCollection('sessions');

// Retrieving documents with a given criteria
await collection.find('interactive = :interactive OR title LIKE :pattern')
  .bind({ interactive: true, pattern: '%Document Store%' })
  .fields('_id', 'speakers')
  .sort('_id ASC')
  .limit(2)
  .offset(1)
  .execute(console.log); // { _id: 'HOL1706', speakers: ['Jesper Krogh', 'Lig Isler-turmelle'] }


// Retrieving a specific document
const doc = await collection.getOne('DEV5985');
console.log(doc); // { _id: 'DEV5985', title: 'MySQL Connector/Node.js and the X DevAPI', speakers: ['Rui Quelhas'] }
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Modifying documents

```
// const collection = schema.getCollection('sessions');

await collection.modify('title LIKE :pattern AND "Rui Quelhas" IN speakers')
  .bind('pattern', '%talk%')
  .set('title', 'Node.js and the MySQL Document Store')
  .execute();

await collection.modify('_id = :id')
  .bind('id', 'DEV5986')
  .patch({ ongoing: true, topics: ['doc-store'] })
  .execute();

await collection.modify('ongoing = :ongoing')
  .bind('ongoing', true)
  .arrayAppend('topics', 'node.js')
  .execute();

await collection.find('"doc-store" IN topics')
  .fields(['ongoing', 'title', 'topics'])
  .execute(console.log); // { ongoing: true, title: 'Node.js and the...' topics: ['doc-store', 'node.js'] }
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Replacing and "Upserting"

```
// const collection = schema.getCollection('sessions');

const existing = 'DEV5959';

await collection.replaceOne(existing, { title: 'Python stuff', topics: ['doc-store'] });

let talk = await collection.getOne(existing);
console.log(talk); // { _id: 'DEV5959', title: 'Python stuff', topics: ['doc-store'] }

await collection.addOrReplaceOne('DEV5959', { interactive: true });

const talks = [];

await collection.find('interactive = true')
  .fields('_id')
  .execute(talk => talks.push(talk));

console.log(talks); // [{ _id: 'DEV5959'}, { _id: 'HOL1706' }]
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Removing documents

```
// const collection = schema.getCollection('sessions');

// Remove documents based in a given criteria
await collection.remove('_id LIKE :pattern')
  .bind({ pattern: 'DEV%' })
  .execute();

await collection.find()
  .fields(['speakers', 'topics'])
  .execute(console.log); // { speakers: ['Jesper Krogh', 'Lig Isler-turmelle'], topics: ['node.js', 'doc-store'] }

// Remove a specific document
await collection.removeOne('HOL1706');

const doc = await collection.getOne('HOL1706');
console.log(doc); // null
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Cross-collection atomic updates

```
try {
  const talks = [];

  await session.startTransaction();

  await schema.getCollection('sessions').find('"node.js" IN topics')
    .fields('_id')
    .lockExclusive(mysqlx.LockContention.DEFAULT) // default value as well
    .execute(talk => { talks.push(talk._id); });

  await schema.createCollection('products', { ReuseExistingObject: true });

  await schema.getCollection('products').add({ name: 'c-nodejs' }).execute();
  await schema.getCollection('products').modify('true')
    .set('talks', talks)
    .execute();

  await session.commit();
} catch (err) {
  await session.rollback();
}
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Creating secondary indexes

```
const collection = schema.getCollection('products'); // to comment

await collection.createIndex('name', {
  fields: [{
    field: '$.name', // column path
    required: true,
    type: 'TEXT(50)'
  }]
});

let indexes = [];

await session
  .sql(`SELECT INDEX_NAME FROM information_schema.STATISTICS WHERE TABLE_NAME = '${collection.getName()}'`)
  .execute(index => { indexes = indexes.concat(index); });

console.log(indexes); // ['PRIMARY', 'name']
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Cleaning up

```
// Delete existing collections
await schema.dropCollection('products');
await schema.dropCollection('sessions');

// Delete existing schemas
await session.dropSchema('codeone2018');

// Close the server session.
await session.close();
```

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Final thoughts

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Some takeaways

- Simplify and secure your database installation
- Reap the benefits of both SQL and NoSQL
- High-availability and scalability via **InnoDB Cluster**
- No secrets or black magic, it's just the regular JSON datatype
- Fast prototyping but always with maintainability in mind
- Stick to a modern simple Node.js workflow (async, npm)
- Deploy everywhere where Node.js excells (containers, microservices, serverless and FaaS, desktop, IoT, etc.)
- It's all open source

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Links and Resources

- https://github.com/mysql/mysql-connector-nodejs
- https://www.npmjs.com/package/@mysql/xdevapi
- https://dev.mysql.com/doc/dev/connector-nodejs/8.0/
- https://ruiquelhas.github.io/percona-live-europe-2017/
- https://www.mysql.com/news-and-events/web-seminars/mysql-document-store-and-node-js/
- https://www.mysql.com/news-and-events/web-seminars/mysql-document-store-and-node-js/
- https://insidemysql.com/mysql-document-store-crud-quick-start/

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

Q&A

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |

ORACLE

Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |