MySQL Document Store

Stuart Davey
MySQL Principal Sales Consultant
stuart.davey@oracle.com

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.

Program Agenda

1. MySQL Document Store
2. MySQL Document Store Demo
3. Non-Functional Requirements
    a. MySQL High Availability
    b. MySQL Enterprise Edition Advantage

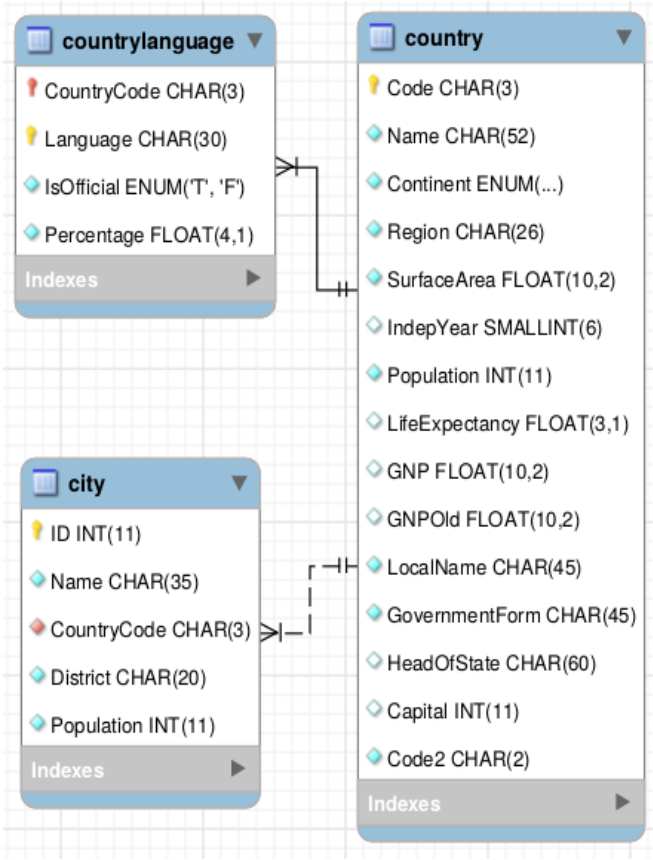
MySQL Document Store

Why Document Store?

How Developers See Data


```
{
    "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"
    }
}
```


How DBAs See Data



MySQL = NoSQL + SQL Dual Model

MySQL is a NoSQL Database

- A **Document Store** to be precise
  - Compare with MongoDB, CouchDB
- It is not:
  - Wide Column Store
  - Key value / Tuple Store
  - Graph Database
  - Object Database
  - Grid Database

MySQL is still a RDBMS

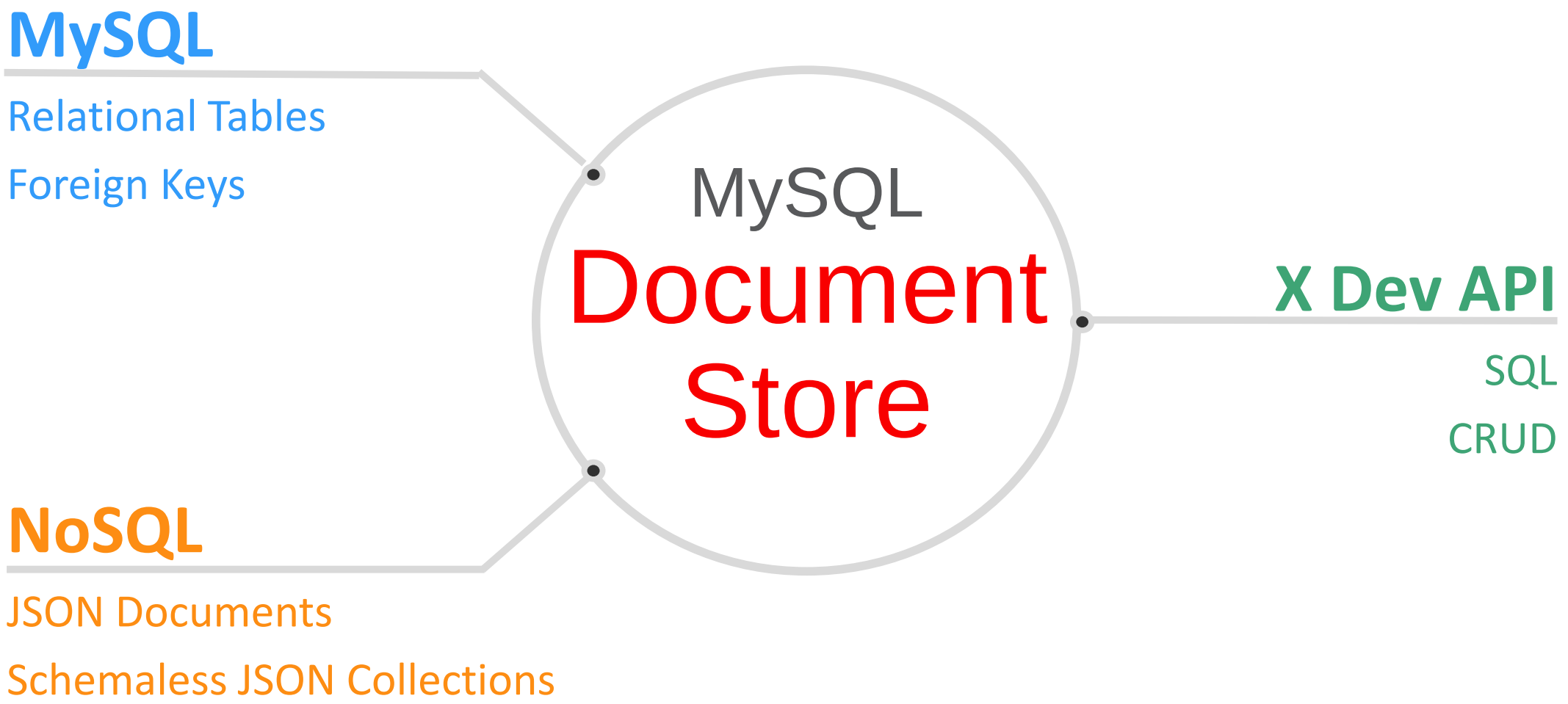
- Relational tabular structure
  - Foreign keys, indexing
  - Triggers, functions, stored procedures
  - DDL, DML using SQL
- Capable of great scale
  - Terabytes, users, transactions per sec
- Comes with complimentary tooling, features and support

MySQL 8.0: Document Store Features

MySQL = NoSQL + SQL

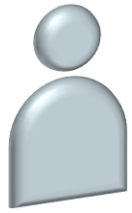
- Power of SQL & Flexibility of NoSQL
  - SQL Relational Tables
  - Schema-less JSON Documents
- Data Guarantees for JSON Documents
  - ACID Compliant
  - Multi-Document Transactional Support
- Consolidate Database Infrastructure
  - Single Database
  - Reduce Effort and Costs

MySQL 8.0: Document Store Elements



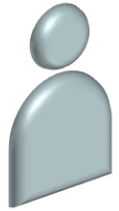
MySQL 8.0: Document Store Benefits

MySQL = NoSQL + SQL



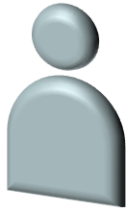
Developer

- Better supports Agile
  (rapid prototyping => MVP)
- Less code / complexity
  (but still transactional / ACID)
- No SQL to code



Operations

- Mainstream database
- Visibility: SQL Interface
- MySQL EE advantage:
  - Monitoring / Alerts
  - Backup and Restore
  - Support



Business Owner

- Time to market (MVP – Agile)
- Reduced costs
- Data integrity (ACID)
- Extract value from data
- Extend data (schema-less)
- MySQL EE advantage:
  - Availability
  - Security

MySQL = NoSQL + SQL

Document Store

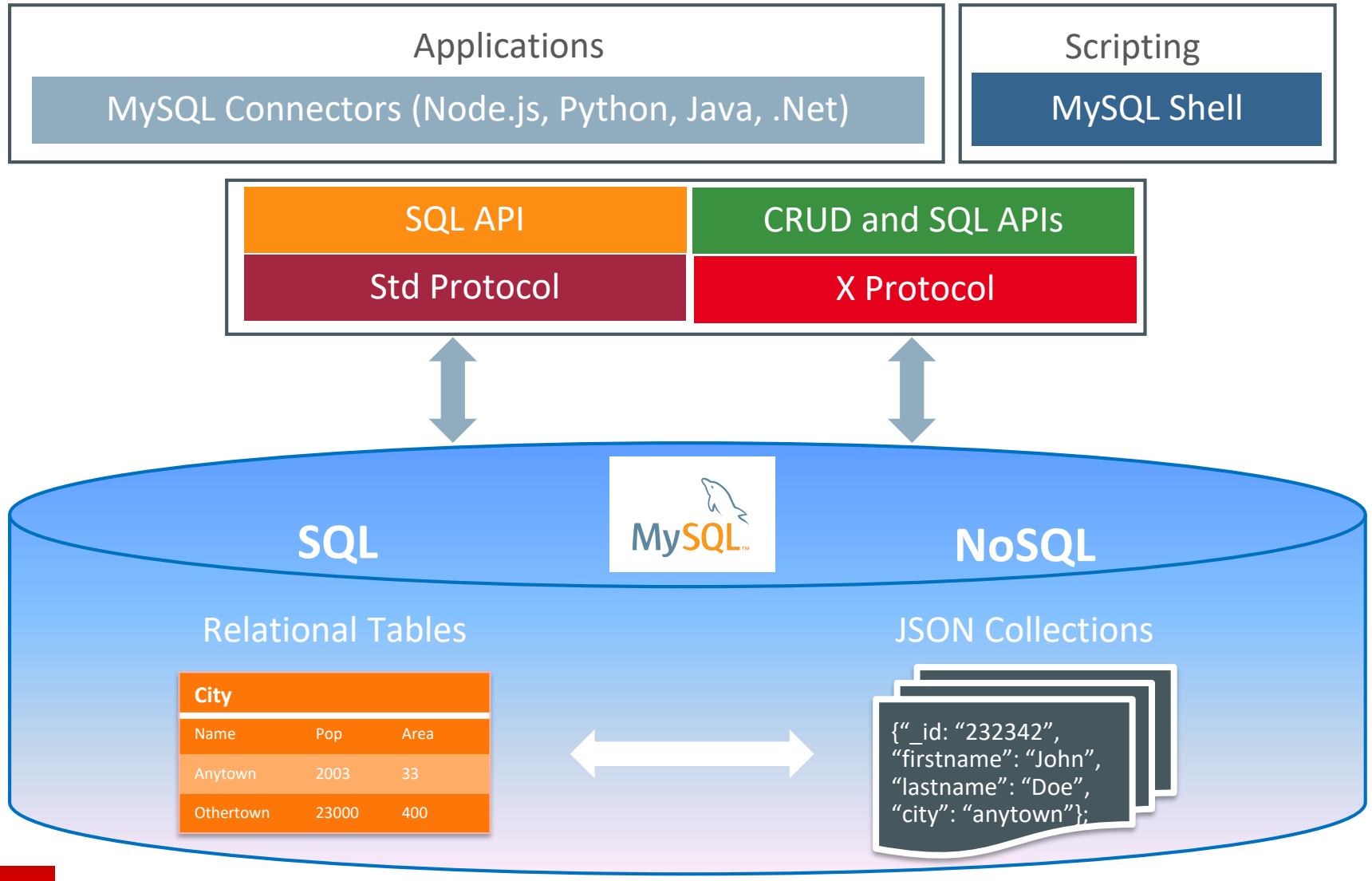
- JavaScript, Python, C++, Java and more
- CRUD: Create, Read, Update, Delete
- JSON Collections
- Schema-less
- More limited Read/Find

Relational Database

- SQL
- INSERT, SELECT, UPDATE, DELETE
- Tables
- Schema
- Powerful SELECT (e.g. join)

MySQL 8.0 : Mix and Match !

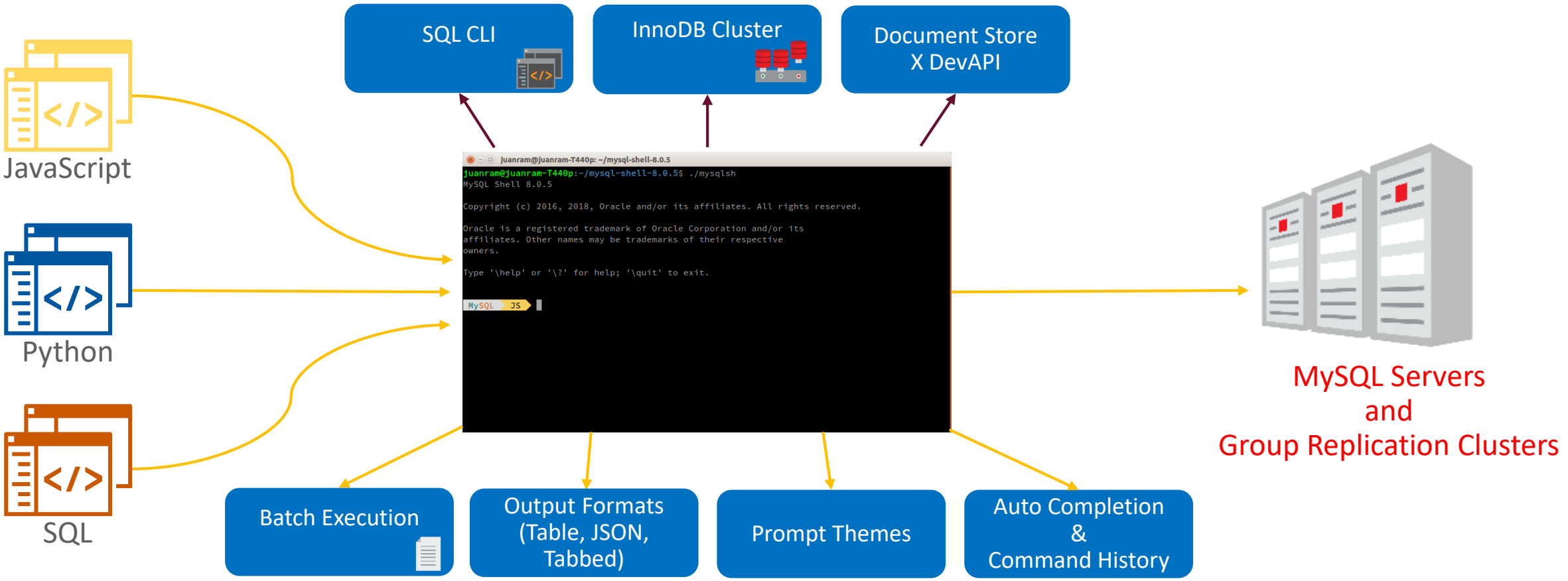
MySQL 8.0: Document Store Architecture



MySQL Document Store: Usage

- DevAPI (patterns)
  - Link your existing relational data with your document data in a single query using simple API patterns
- DevAPI based **Connectors**
  - **Node.js**, Python, PHP, .Net, ODBC, C++, Java
- DevAPI based **MySQL Shell**
  - Dev+Admin: JavaScript, Python and SQL
- New, async **client-server protocol** (X Protocol)
- And, the “good old MySQL Server”

MySQL Shell



MySQL Document Store Demonstration

MySQL 8.0: Demo Agenda

NoSQL + SQL = MySQL

- Using mysqlsh
- Xdevapi CRUD operations
- The SQL interface
- Xdevapi Indexing for Performance
- Joining collections and tables
- What this means for your code

MySQL 8.0: JSON Functions (SQL Interface)

NoSQL + SQL = MySQL

Functions for finding and getting paths and data

JSON_CONTAINS()

JSON_CONTAINS_PATH()

JSON_EXTRACT()

JSON_KEYS()

JSON_SEARCH()

Functions for changing data

JSON_ARRAY_APPEND()

JSON_ARRAY_INSERT()

JSON_INSERT()

JSON_MERGE()

JSON_MERGE_PATCH()

JSON_MERGE_PRESERVE()

JSON_REPLACE()

JSON_REMOVE()

JSON_SET()

Helper Functions

JSON_DEPTH()

JSON_LENGTH()

JSON_PRETTY()

JSON_UNQUOTE()

JSON_TYPE()

JSON_VALID()

JSON_STORAGE_SIZE()

JSON_STORAGE_FREE()

JSON & Non-JSON Output

JSON_OBJECT()

JSON_ARRAY()

JSON_TABLE()

Remember you can also CAST()

GeoJSON Functions

ST_AsGeoJSON()

ST_GeomFromGeoJSON()

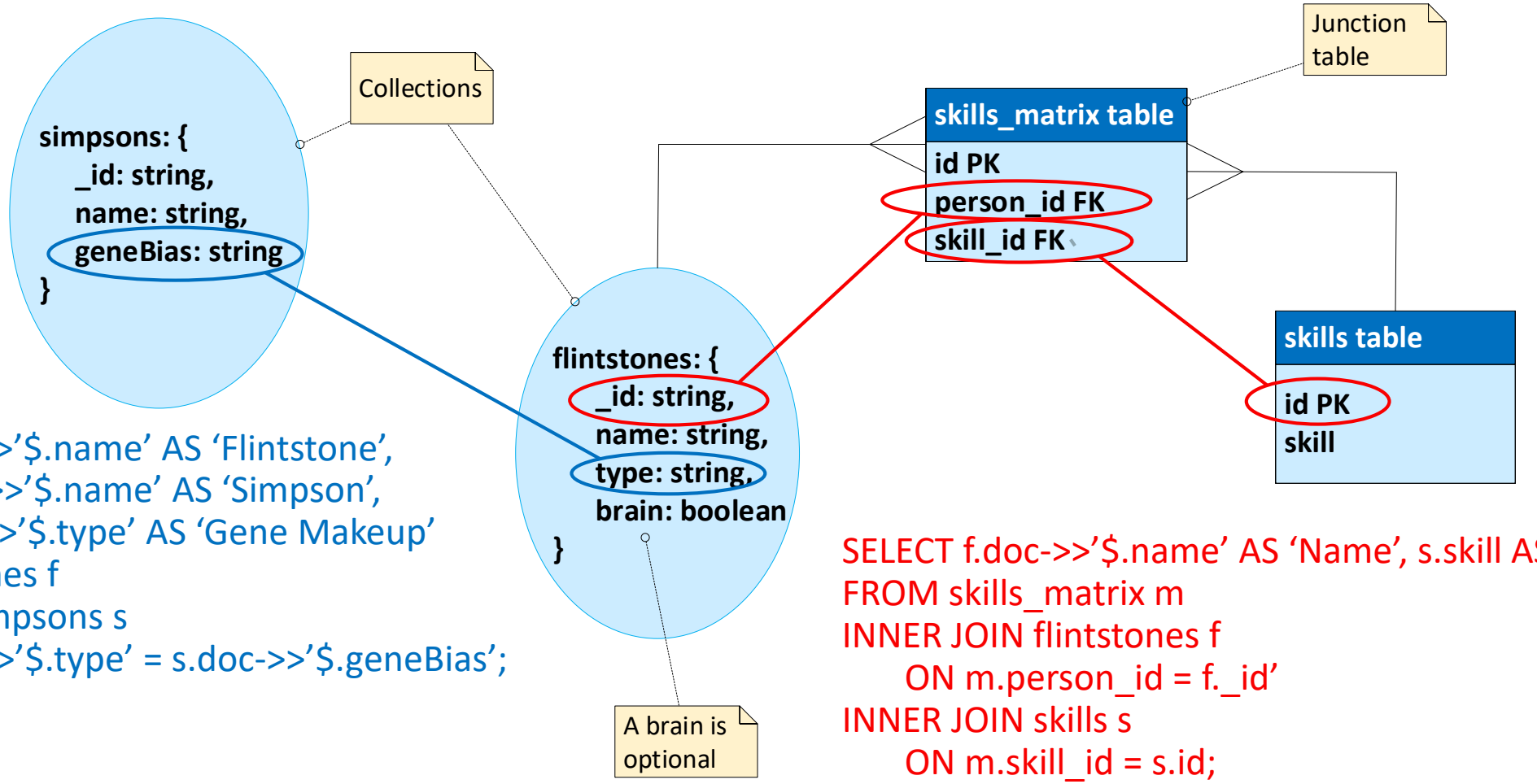
ST_GeomFromText()

Remember: for brevity use the operator -> instead of JSON_EXTRACT()

and ->> instead of JSON_UNQUOTE(JSON_EXTRACT())

MySQL 8.0: Joining Collections and Tables

NoSQL + SQL = MySQL




```
SELECT f.doc->>'$.name' AS 'Flintstone',
       s.doc->>'$.name' AS 'Simpson',
       f.doc->>'$.type' AS 'Gene Makeup'
FROM flintstones f
INNER JOIN simpsons s
    ON f.doc->>'$.type' = s.doc->>'$.geneBias';
```

```
SELECT f.doc->>'$.name' AS 'Name', s.skill AS 'Skill'
FROM skills_matrix m
INNER JOIN flintstones f
    ON m.person_id = f._id'
INNER JOIN skills s
    ON m.skill_id = s.id;
```

MySQL 8.0: Node.js Example (part 1 of 2)

NoSQL + SQL = MySQL

```
const express = require('express');
const bodyParser = require('body-parser');
const mysqlx = require('@mysql/xdevapi');

const app = express();
const port = 3000;
const client = mysqlx.getClient(
        {user: ‘appuser', host: 'localhost', password: ‘xyz12345()Q', port: 33060},
        {pooling: { enabled: true, maxIdleTime: 30000, maxSize: 25, queueTimeout: 10000}}
);

app.use(bodyParser.json());
app.listen(port, function() {
        console.log(`Example app listening on port ${port}!`);
});
```

Load xdevapi connector

Create a session connection pool

MySQL 8.0: Node.js Example (part 2 of 2)

NoSQL + SQL = MySQL

```
app.get('/nycfood/cuisines', async function(req, res) {
        try {
                let docArray = [];
                let session = await client.getSession();
                let col = await session.getSchema('nycfood').getCollection('outlets');
                let result = await col.find().fields('cuisine').groupBy('cuisine').sort('cuisine').execute(function(doc) {
                        docArray.push(doc);
                });
                result.getWarningsCount() > 0 ? res.send({"error": result.getWarnings()}) : res.send(docArray);
                session.close();
        } catch(err) {
                console.log(err);
        }
});
```

Get session connection from client pool

Get schema then collection from session

Perform ‘query’ (i.e. find())
Note: chaining and callback function

Response sends back
(returns) the docArray

MySQL 8.0: Document Store and Your Application Code

NoSQL + SQL = MySQL

- Xdevapi provides simpler, cleaner code
  - Uses the same syntax you are writing in (JS, Node.js, Python, Java, PHP, C++, .Net)
  - Not an inelegant set of strings littered with question marks representing SQL code
  - Fewer transformations – no taking objects from JSON and inserting into SQL
  - Consider the complexity of handling a collection which includes arrays using tables…
- Supports Agile development methods
  - Sprints, MVP completion, code more resistant to data model changes
- No need to become expert in SQL

Caveat: there will always be application use cases that demand the rigour of an RDBMS

Non Functional Requirements

MySQL 8.0: NFRs – You Can't Avoid Them

NoSQL + SQL = MySQL

- Availability
  - HA with InnoDB Cluster; automatic failover, follow the Master with MySQL Router
- Security (Enterprise Edition Advantage)
  - Authentication/Authorization integration, Transparent Date Encryption, Audit, Firewall, Data Masking
- Backup that's scalable, fast and reliable (Enterprise Edition Advantage)
- Performance enhancements – threads library plugin (Enterprise Edition Advantage)
- Support – traditional and consultative (Enterprise Edition Advantage)
- With Enterprise Edition you can also leverage other licensed Oracle products

Summary

MySQL 8.0: Document Store Recap

NoSQL + SQL = MySQL

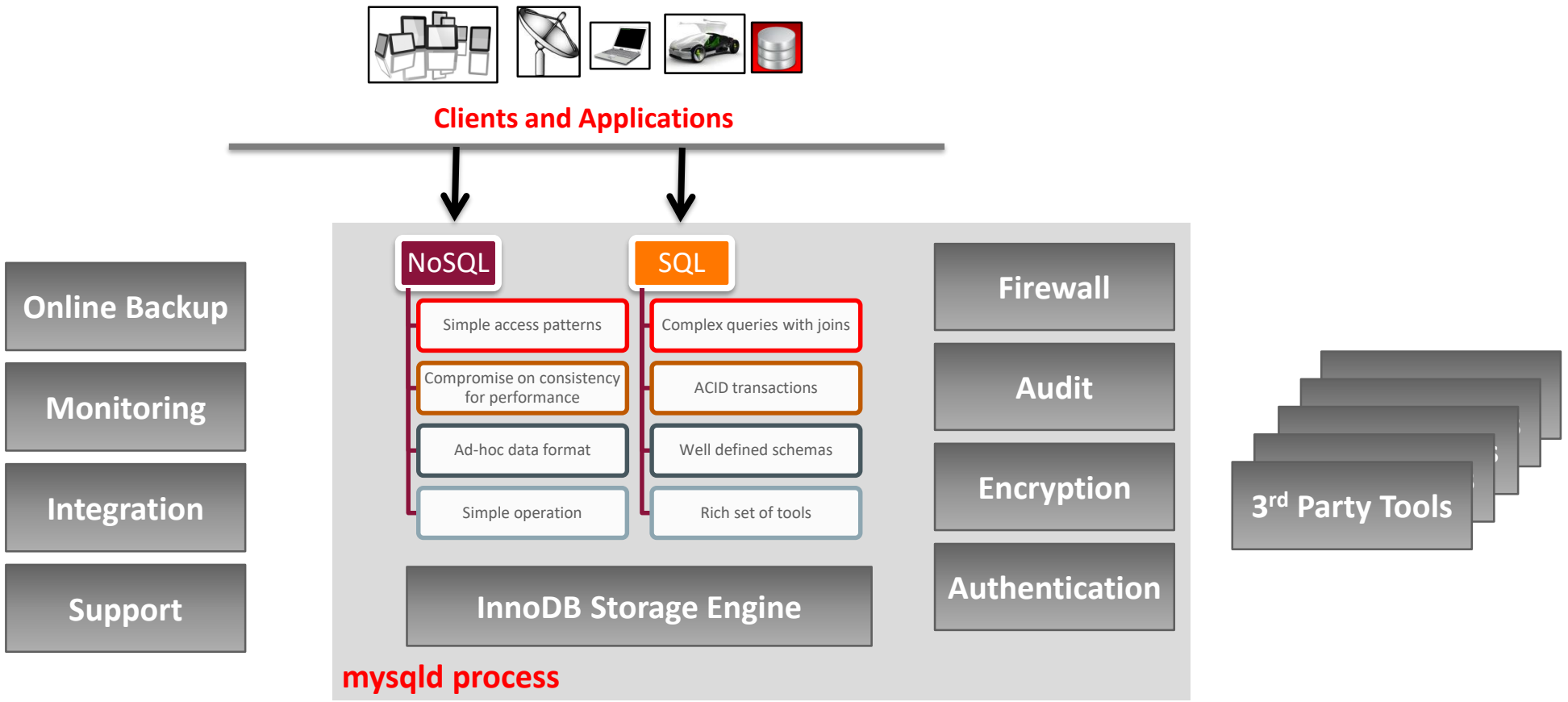
- Document oriented data storage for MySQL
  - Full **JSON document support** through **SQL** (inc. JSON functions) and the **new X DevAPI NoSQL interface**
- Schema-less and schema based data in the same technology stack
  - Use **COLLECTIONs** of documents & relational **TABLEs** together
- Rapid Prototyping & Simple CRUD APIs
  - **Modern APIs** using “method chaining” and asynchronous execution (e.g. promises, callbacks, etc.)
- Connectors for many different languages and frameworks
  - **Node.JS**, Java, NET, C++/C, PHP, Python

MySQL Document Store Top 10

Top 10 reasons you should consider using NoSQL with MySQL:

1. MySQL cares about your data ! NoSQL full ACID compliant
2. CRUD operations (SQL is not mandatory anymore)
3. Schema-less
4. Documents have the benefit of Data Integrity
5. Allows SQL (very important for analytics)
6. No 16MB limitation for Document
7. Simplest query syntax
8. Security
9. Simplify your DB infrastructure
10. Your MySQL DBAs already know how to manage/tune/scale MySQL

All-Bases-Covered = NoSQL + SQL + Enterprise Edition



This is the best of both worlds in one product !


ORACLE®