ORACLE

# MySQL 8.0 Document Store

## discovery of a new world

Frédéric Descamps
Community Manager
MySQL
October 2020

# Who am I ?

about.me/lefred


Copyright @ 2020 Oracle and/or its affiliates.

# Frédéric Descamps

- @lefred
- MySQL Evangelist
- Managing MySQL since 3.20 !
- devops believer
- living in Belgium 🇧🇪
- https://lefred.be


Copyright @ 2020 Oracle and/or its affiliates.

MySQL

#MySQL8isGreat


Copyright @ 2020 Oracle and/or its affiliates.

# MySQL is the DBMS of the Year 2019 !



Copyright @ 2020 Oracle and/or its affiliates.

# Happy 25th Anniversary MySQL


Copyright @ 2020 Oracle and/or its affiliates.

# Evolution...

## from LAMP stack to modern Web applications


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Web Applications 2000's

Who ?


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications 2000's

Who ?


Copyright @ 2020 Oracle and/or its affiliates.

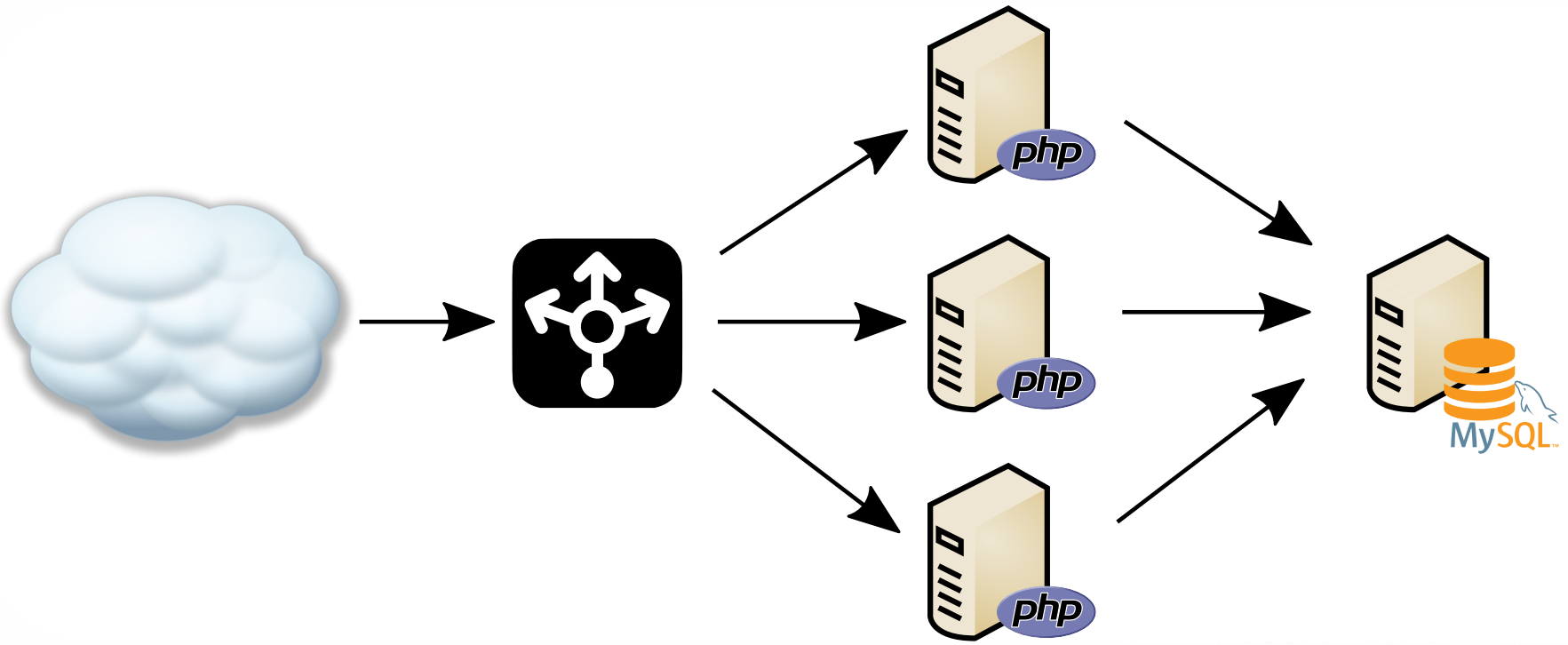
# Web Applications 2000's

How ?


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications 2000's

How ?





Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization
  - constraints (foreign keys, ...)


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization
  - constraints (foreign keys, ...)
- Atomicity, Consistency, Isolation, Durability


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization
  - constraints (foreign keys, ...)
- Atomicity, Consistency, Isolation, Durability
  - ACID compliant


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization
  - constraints (foreign keys, ...)
- Atomicity, Consistency, Isolation, Durability
  - ACID compliant
  - transactions


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization
  - constraints (foreign keys, ...)
- Atomicity, Consistency, Isolation, Durability
  - ACID compliant
  - transactions
- SQL


Copyright @ 2020 Oracle and/or its affiliates.

# Relational Databases

- Data Integrity
  - normalization
  - constraints (foreign keys, ...)
- Atomicity, Consistency, Isolation, Durability
  - ACID compliant
  - transactions
- SQL
  - powerfull query language


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

Who ?


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

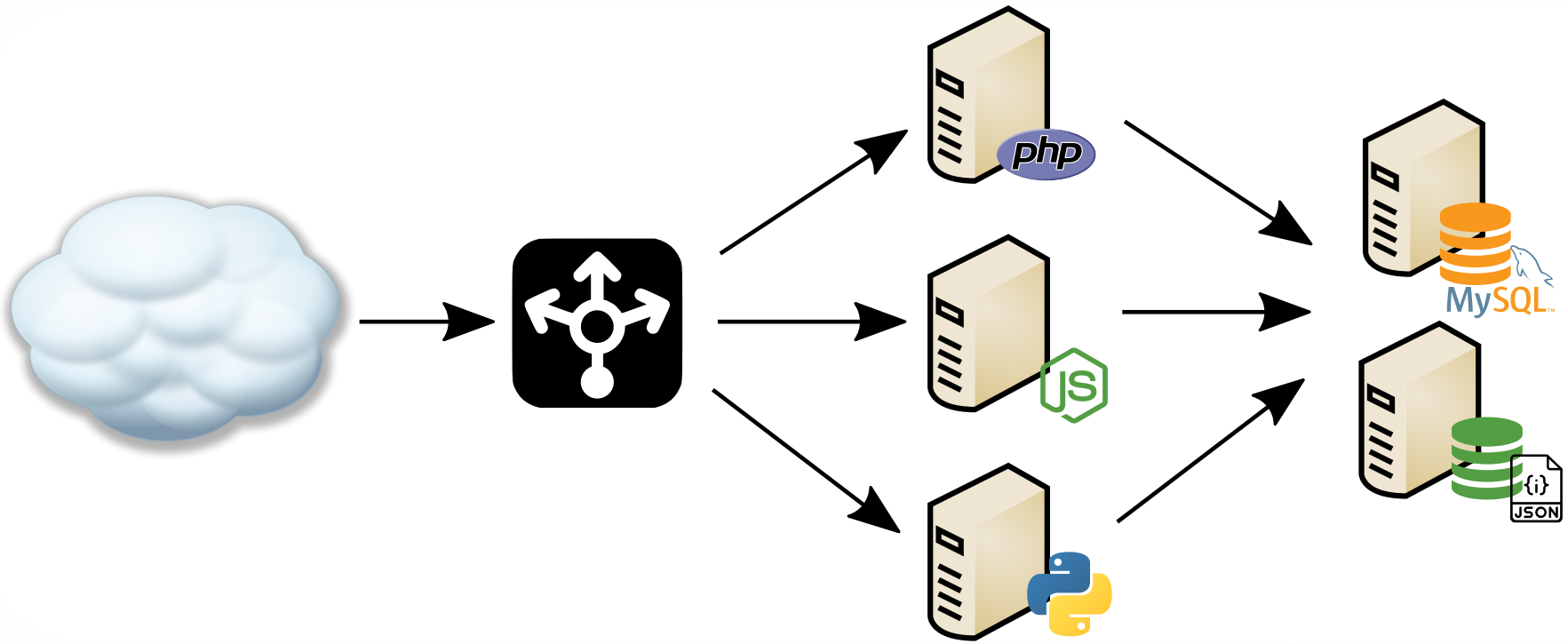
Who ?



Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

How ?





Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

Why ?


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

Why ?

Developers don't really like SQL anymore...

SQL

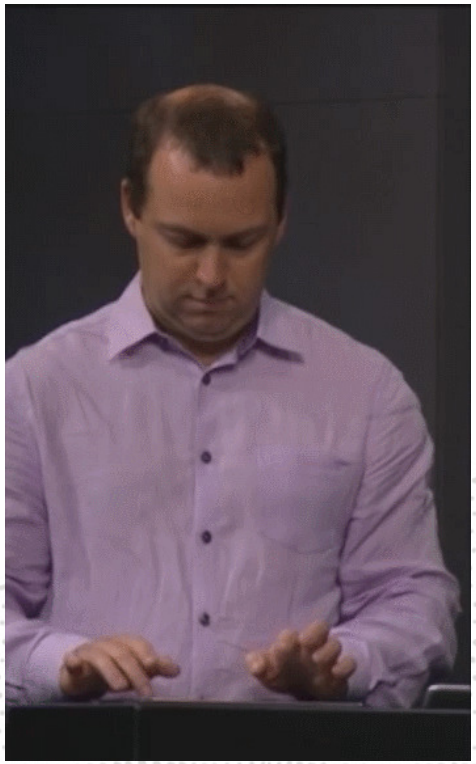
NoSQL


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

Why ?

SQL can be complicated and slows down the initial development




Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

Why ?

Developers don't have time to learn SQL, they need time to workout and have a nice look ;-)


Copyright @ 2020 Oracle and/or its affiliates.

# What do developers want ?

easy operations


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Use Objects / Documents

Developers want just to use objects


Copyright @ 2020 Oracle and/or its affiliates.

# Use Objects / Documents

Developers want just to use objects

- that's why they usually love what DBAs hate the most ORMs !


Copyright @ 2020 Oracle and/or its affiliates.

# Use Objects / Documents

Developers want just to use objects

- that's why they usually love what DBAs hate the most ORMs !

They want to deal with these objects easily (CRUD operations) and they don't want to think about schema design (slows down the initial development process).


Copyright @ 2020 Oracle and/or its affiliates.

# Use Objects / Documents

Developers want just to use objects

- that's why they usually love what DBAs hate the most ORMs !

They want to deal with these objects easily (CRUD operations) and they don't want to think about schema design (slows down the initial development process).

But they also want to keep their data safe and use transactions.


Copyright @ 2020 Oracle and/or its affiliates.

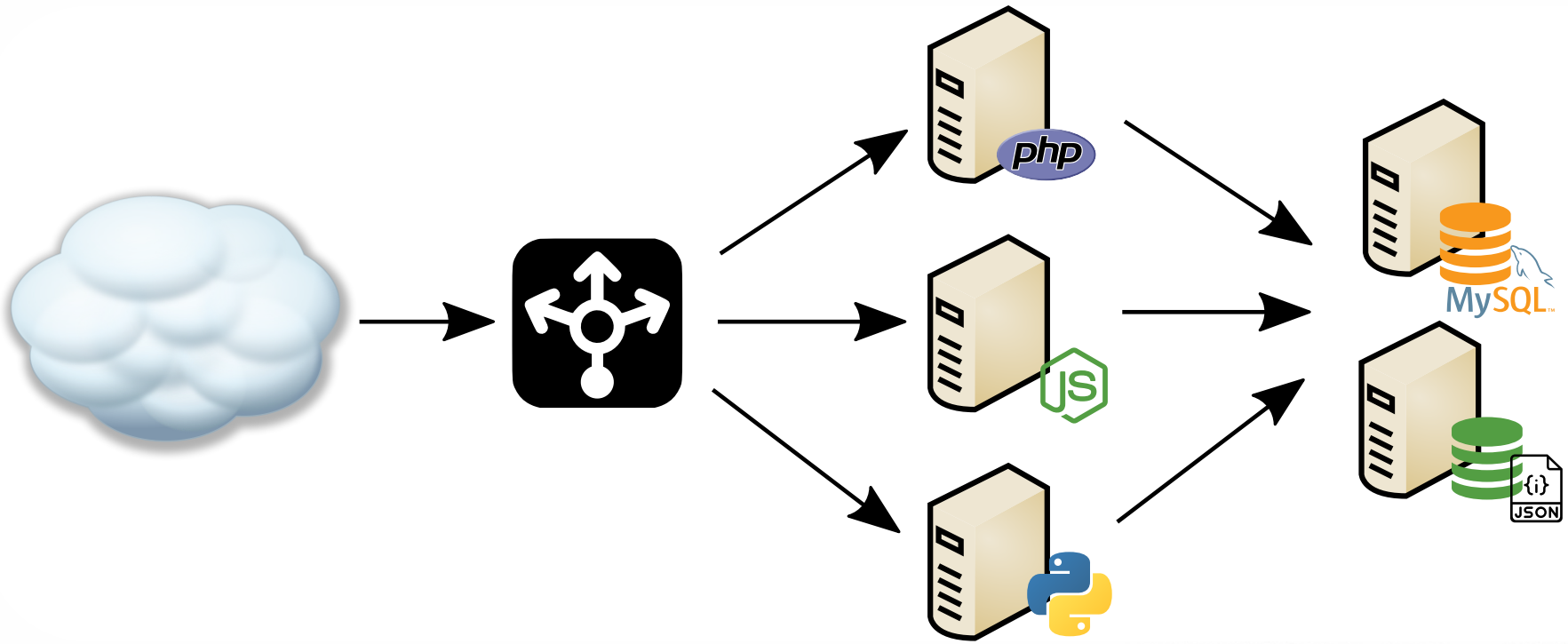
# Web Applications current days

How ?


Copyright @ 2020 Oracle and/or its affiliates.

# Web Applications current days

How ?





Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Databases

## JSON Document Store


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# NoSQL Document Store

- Schemaless


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, …


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, …
  - very quick initial development


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, …
  - very quick initial development
- Flexible data structure


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, …
  - very quick initial development
- Flexible data structure
  - embedded arrays or objects


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, ...
  - very quick initial development
- Flexible data structure
  - embedded arrays or objects
  - valid solution when natural data can't be modelized optimaly into a relational model


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, ...
  - very quick initial development
- Flexible data structure
  - embedded arrays or objects
  - valid solution when natural data can't be modelized optimaly into a relational model
  - objects persistance without the use of any ORM - *mapping oobject-oriented*


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, ...
  - very quick initial development
- Flexible data structure
  - embedded arrays or objects
  - valid solution when natural data can't be modelized optimaly into a relational model
  - objects persistance without the use of any ORM - *mapping oobject-oriented*
- JSON


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, ...
  - very quick initial development
- Flexible data structure
  - embedded arrays or objects
  - valid solution when natural data can't be modelized optimaly into a relational model
  - objects persistance without the use of any ORM - *mapping oobject-oriented*
- JSON
  - close to frontend

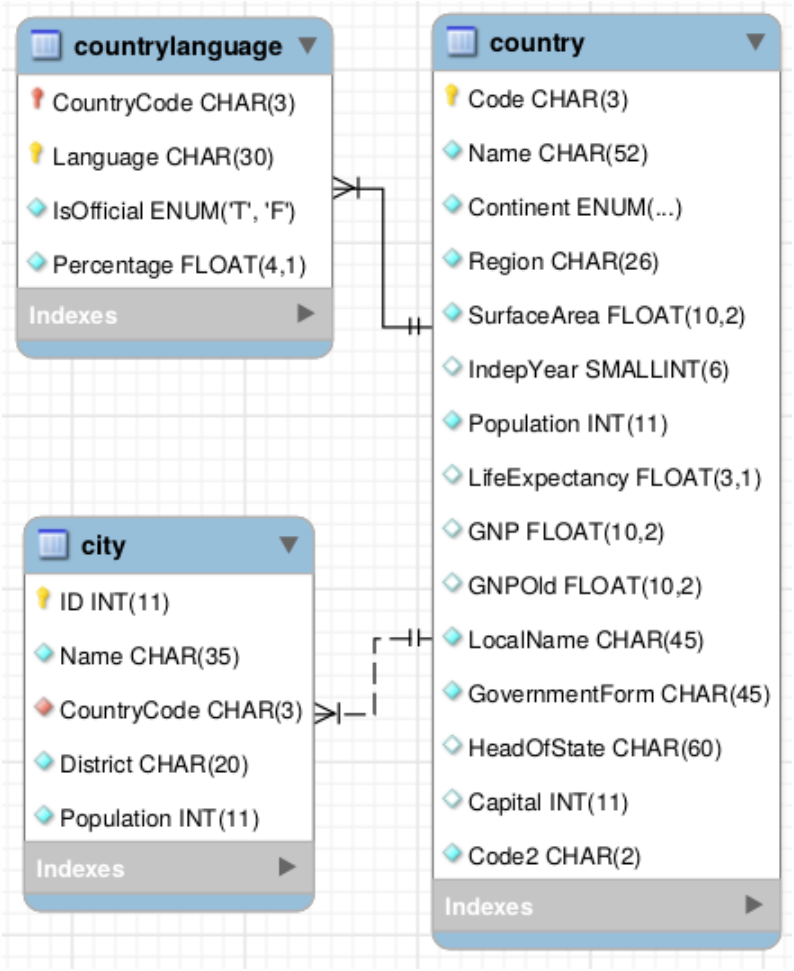

Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL Document Store

- Schemaless
  - no schema design, no normalization, no foreign keys, no data types, ...
  - very quick initial development
- Flexible data structure
  - embedded arrays or objects
  - valid solution when natural data can't be modelized optimaly into a relational model
  - objects persistance without the use of any ORM - *mapping oobject-oriented*
- JSON
  - close to frontend
  - easy to learn


Copyright @ 2020 Oracle and/or its affiliates.

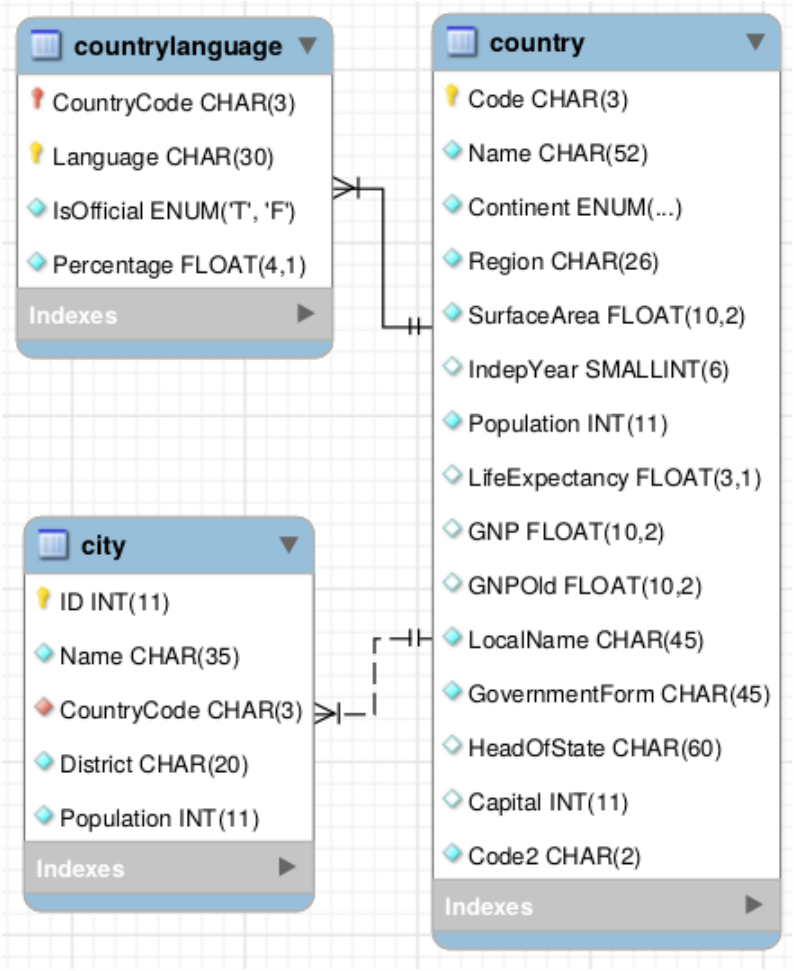
# How DBAs see data





Copyright @ 2020 Oracle and/or its affiliates.

# How DBAs see data

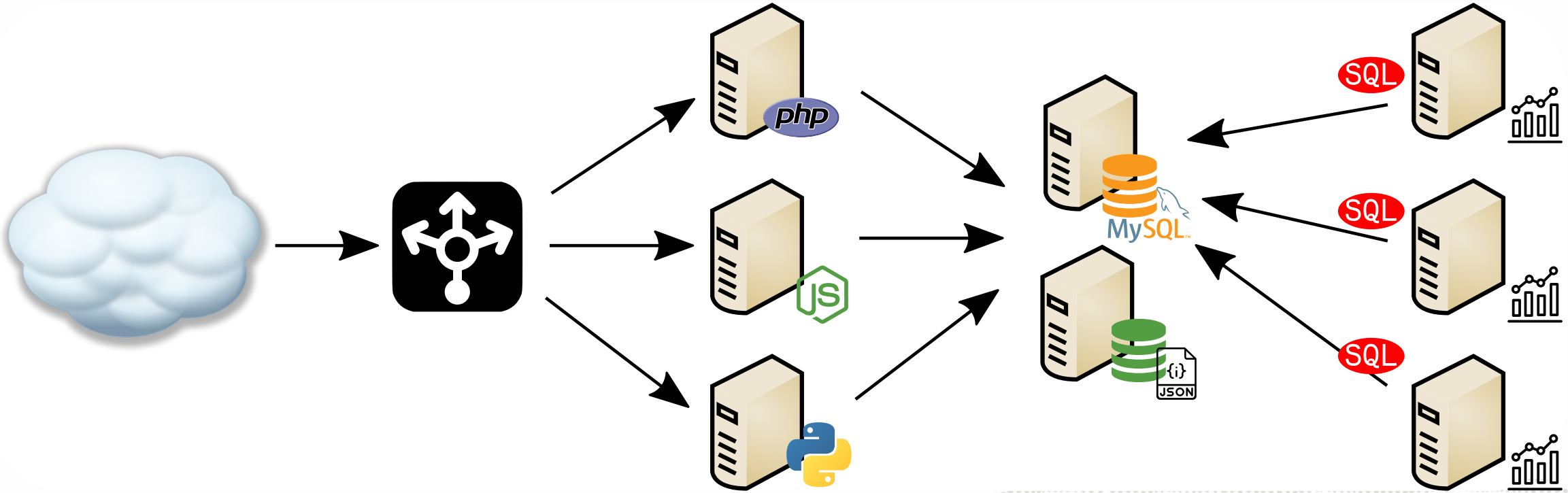



# How Developers see data

```
{
   "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"
   }
}
```


Copyright @ 2020 Oracle and/or its affiliates.

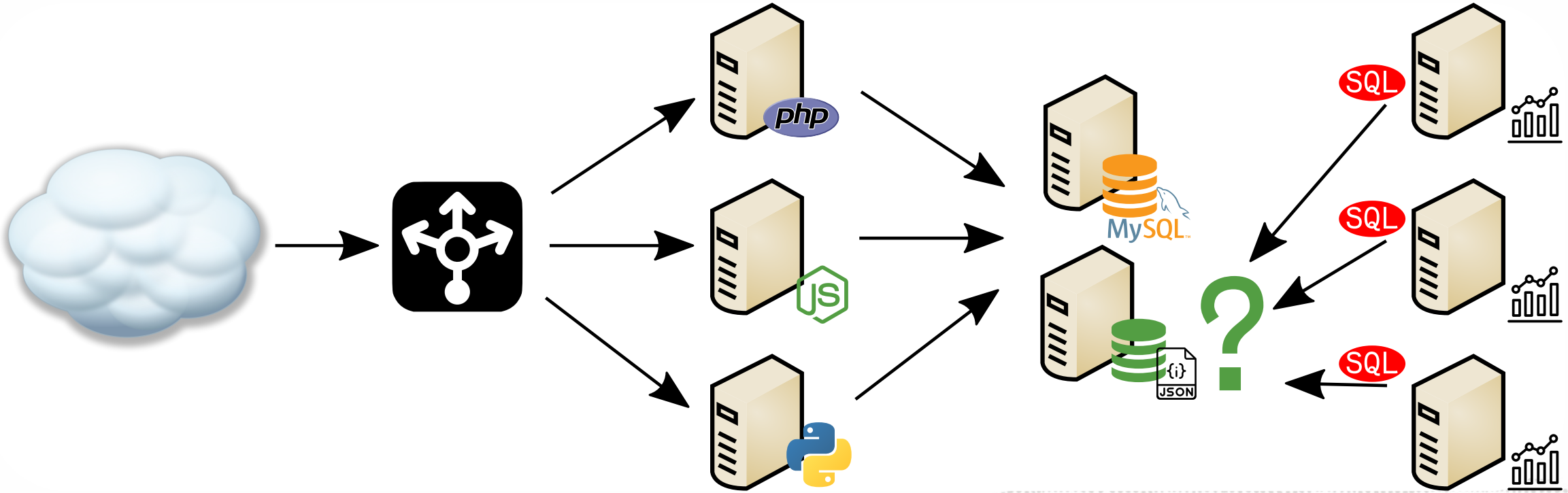
# And they still need to do Analytics





Copyright @ 2020 Oracle and/or its affiliates.

# … mmm…how ?





Copyright @ 2020 Oracle and/or its affiliates.

# Help needed !

Who ?

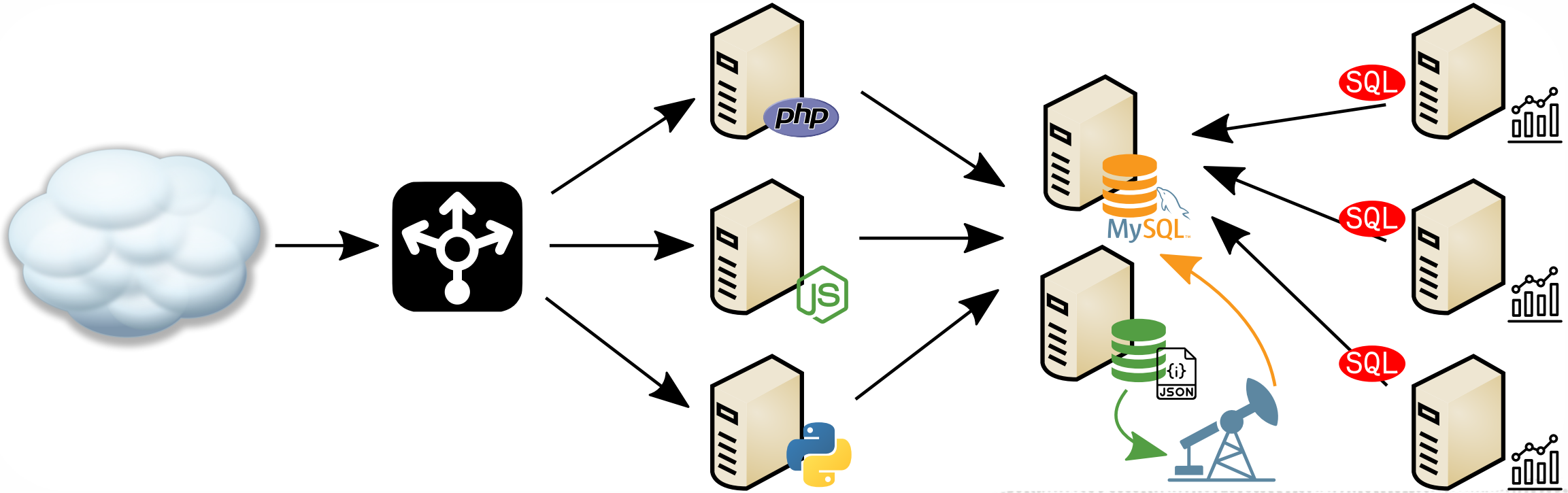

Copyright @ 2020 Oracle and/or its affiliates.

# Help needed !

Who ?


Copyright @ 2020 Oracle and/or its affiliates.

# How ?





Copyright @ 2020 Oracle and/or its affiliates.

**What if there was a way to provide both SQL and NoSQL on one stable platform that has proven stability on well know technology with a large Community and a diverse ecosystem ?**


Copyright @ 2020 Oracle and/or its affiliates.

# DBMS or NoSQL ?


Copyright @ 2020 Oracle and/or its affiliates.

# DBMS or NoSQL ?

## Why not both ?



Copyright @ 2020 Oracle and/or its affiliates.

# The MySQL Document Store !

## SQL is now optional ?!

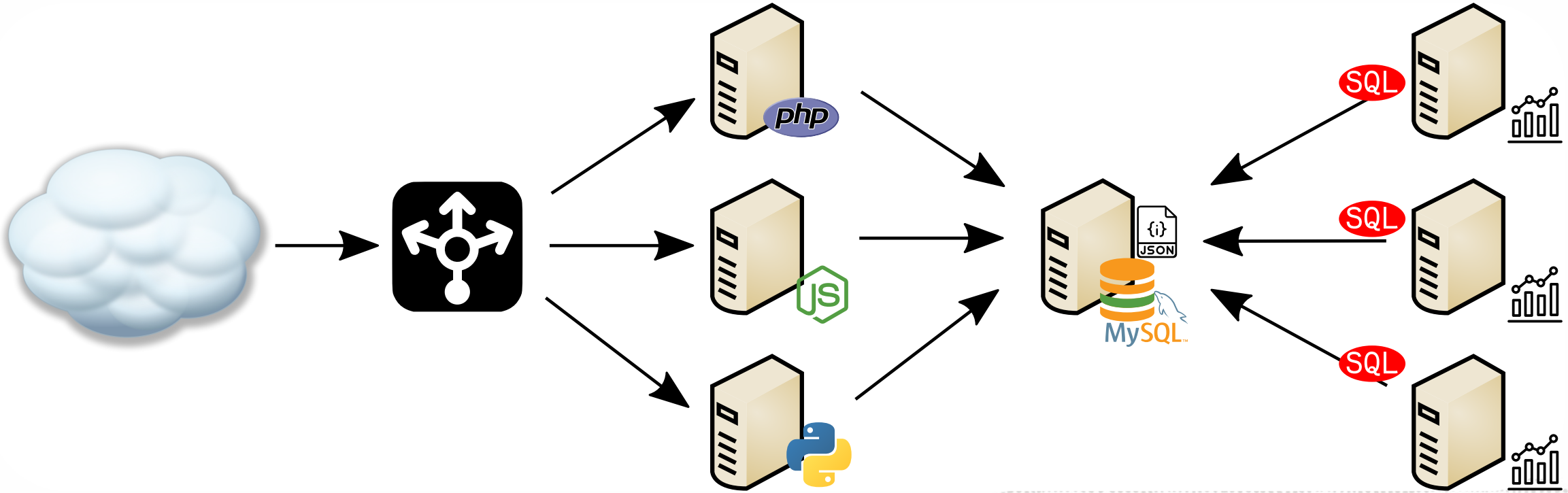

Copyright @ 2020 Oracle and/or its affiliates.

# SQL is now optional ?!



Copyright @ 2020 Oracle and/or its affiliates.

# Using MySQL Document Store !





Copyright @ 2020 Oracle and/or its affiliates.

# A solution for all

## Business Owner:

- [x] don't lose my data == ACID trx
- [x] capture all my data = extensible/schemaless
- [x] product on schedule/time to market = rapid developement

## Operations:

- [x] performance management/visibility
- [x] robust replication, backup, restore
- [x] comprehensive tooling ecosystem
- [x] simpler application schema upgrades

## Developers:

- [x] schemaless
- [x] rapid prototying/simpler APIs
- [x] document model
- [x] transactions


Copyright @ 2020 Oracle and/or its affiliates.

# the Solution

MySQL Document Store


Copyright @ 2020 Oracle and/or its affiliates.

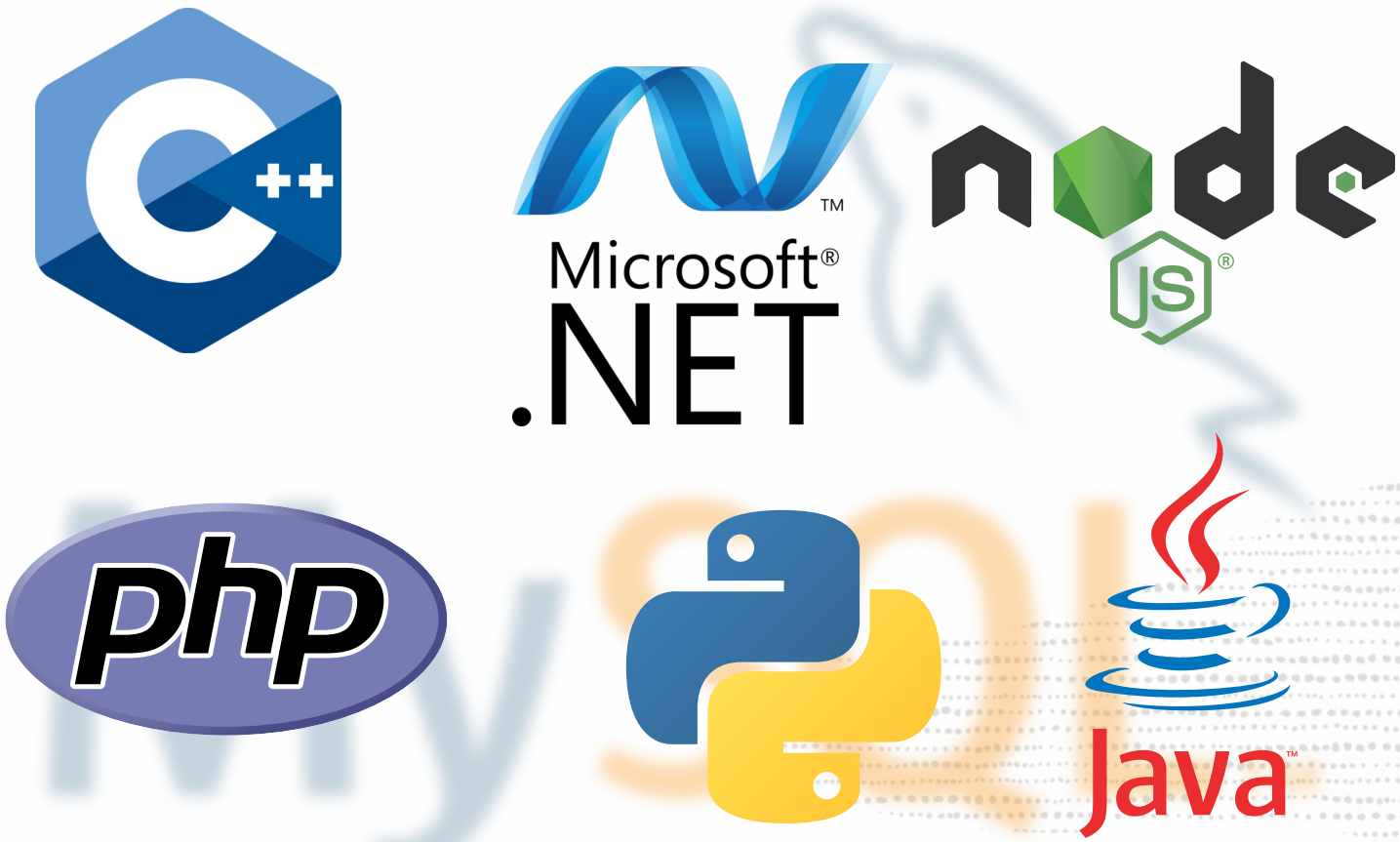
@lefred

# Built on the MySQL JSON Data type and Proven MySQL Server Technology

- Provides a schema flexible JSON Document Store
- **No** SQL required
- No need to define all possible attributes, tables, etc.
- Uses new X DevAPI
- Can leverage generated column to extract JSON values into materialized columns that can be indexed for fast SQL searches.
- Document can be ~1GB
  - It's a column in a row of a table
  - It cannot exceed `max_allowed_packet`
- Allows use of modern programming styles
  - No more embedded strings of SQL in your code
  - Easy to read
- Also works with relational Tables
- Proven MySQL Technology


Copyright @ 2020 Oracle and/or its affiliates.

# X Protocol Connectors





Copyright @ 2020 Oracle and/or its affiliates.

# X DevAPI

- We provide connectors for
  - C++, Java, .Net, Node.js, Python, PHP
  - working with Communities to help them supporting it too
- New MySQL Shell
  - Command Completion
  - Python, JavaScrips & 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


Copyright @ 2020 Oracle and/or its affiliates.

# Setup

## MySQL Document Store


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Installing MySQL Document Store

- install MySQL 8.0


Copyright @ 2020 Oracle and/or its affiliates.

# Installing MySQL Document Store

- install MySQL 8.0
- install MySQL Shell


Copyright @ 2020 Oracle and/or its affiliates.

# Installing MySQL Document Store

- install MySQL 8.0
- install MySQL Shell
- install MySQL Connector for your programming language


Copyright @ 2020 Oracle and/or its affiliates.

# Installing MySQL Document Store

- install MySQL 8.0
- install MySQL Shell
- install MySQL Connector for your programming language
  - `php-pecl-mysql-xdevapi` for PHP
  - `mysql-connector-python` for Python
  - ...


Copyright @ 2020 Oracle and/or its affiliates.

# Installing MySQL Document Store

- install MySQL 8.0
- install MySQL Shell
- install MySQL Connector for your programming language
  - `php-pecl-mysql-xdevapi` for PHP
  - `mysql-connector-python` for Python
  - ...

And nothing else, no need to install anything else or load any plugin, just be sure your firewall allows you to connect through port `33060` *(X Protocol)*.


Copyright @ 2020 Oracle and/or its affiliates.

# MySQL Database Service

X Protocol is also available in MDS !!



Copyright @ 2020 Oracle and/or its affiliates.

# Migration from MongoDB to MySQL DS

For this example, I will use the well known `restaurants` collection:

```
[root@myserver1 ~]# mongoexport -c restaurants > from_mongo.json
connected to: 127.0.0.1
exported 25359 records
```


Copyright @ 2020 Oracle and/or its affiliates.

# Migration from MongoDB to MySQL DS

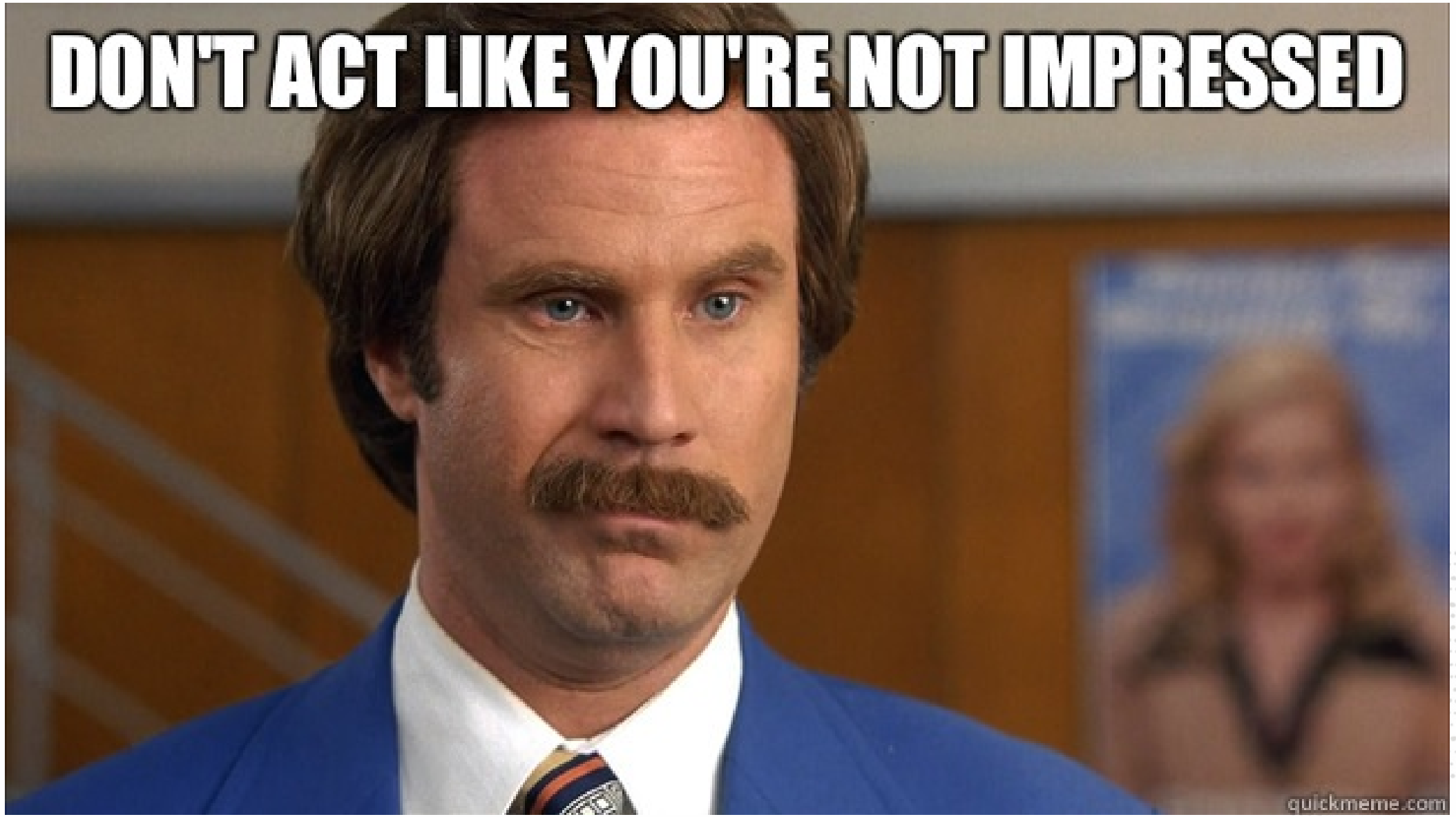
For this example, I will use the well known `restaurants` collection:

```
[root@myserver1 ~]# mongoexport -c restaurants > from_mongo.json
connected to: 127.0.0.1
exported 25359 records
```

```
MySQL  localhost:33060+  docstore  2018-09-09 20:01:45  Threads running: 2
JS util.importJson('/vagrant/from_mongo.json',{schema: 'docstore', collection: 'restaurants', convertBsonOid: true})
Importing from file "/vagrant/from_mongo.json" to collection `docstore`.`restaurants` in MySQL Server at localhost

.. 25359.. 25359
Processed 15.90 MB in 25359 documents in 11.4145 sec (2.22K documents/s)
Total successfully imported documents 25359 (2.22K documents/s)
```


Copyright @ 2020 Oracle and/or its affiliates.




 Copyright @ 2020 Oracle and/or its affiliates.


Copyright @ 2020 Oracle and/or its affiliates.

# Let's make a query



Copyright @ 2020 Oracle and/or its affiliates.

# Let's make a query


That's too much records to show in here... let's limit it


Copyright @ 2020 Oracle and/or its affiliates.

```
MySQL 127.0.0.1:33060 JS restaurants.find().limit(1)
[
    {
        "_id": "5943c83d1adc26055941640c",
        "address": {
            "building": "351",
            "coord": [
                -73.9851,
                40.7677
            ],
            "street": "West   57 Street",
            "zipcode": "10019"
        },
        "borough": "Manhattan",
        "cuisine": "Irish",
        "grades": [
            {
                "date": "2014-09-06T00:00:00Z",
                "grade": "A",
                "score": 2
            },
            {
                "date": "2013-07-22T00:00:00Z",
                "grade": "A",
                "score": 11
            },
            {
                "date": "2012-07-31T00:00:00Z",
                "grade": "A",
                "score": 12
            },
            {
                "date": "2011-12-29T00:00:00Z",
                "grade": "A",
                "score": 12
            }
        ],
        "name": "Dj Reynolds Pub And Restaurant",
        "restaurant_id": "30191841"
    }
]
1 document in set (0.08 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Some more examples

```
MySQL 127.0.0.1:33060 JS restaurants.find().fields(["name","cuisine"]).limit(2)
[
    {
        "cuisine": "Irish",
        "name": "Dj Reynolds Pub And Restaurant"
    },
    {
        "cuisine": "American",
        "name": "Riviera Caterer"
    }
]
2 documents in set (0.00 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Let's add a selection criteria:

```
MySQL 127.0.0.1:33060 JS restaurants.find("cuisine='Italian'").fields(["name","cuisine"]).limit(2)
[
    {
        "cuisine": "Italian",
        "name": "Philadelhia Grille Express"
    },
    {
        "cuisine": "Italian",
        "name": "Isle Of Capri Resturant"
    }
]
2 documents in set (0.00 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Syntax slightly different than MongoDB

```
MySQL 127.0.0.1:33060 JS restaurants.find("cuisine='French' AND borough!='Manhattan'").
fields(["name","cuisine","borough"]).limit(2)
[
    {
        "borough": "Queens",
        "cuisine": "French",
        "name": "La Baraka Restaurant"
    },
    {
        "borough": "Queens",
        "cuisine": "French",
        "name": "Air France Lounge"
    }
]
2 documents in set (0.00 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

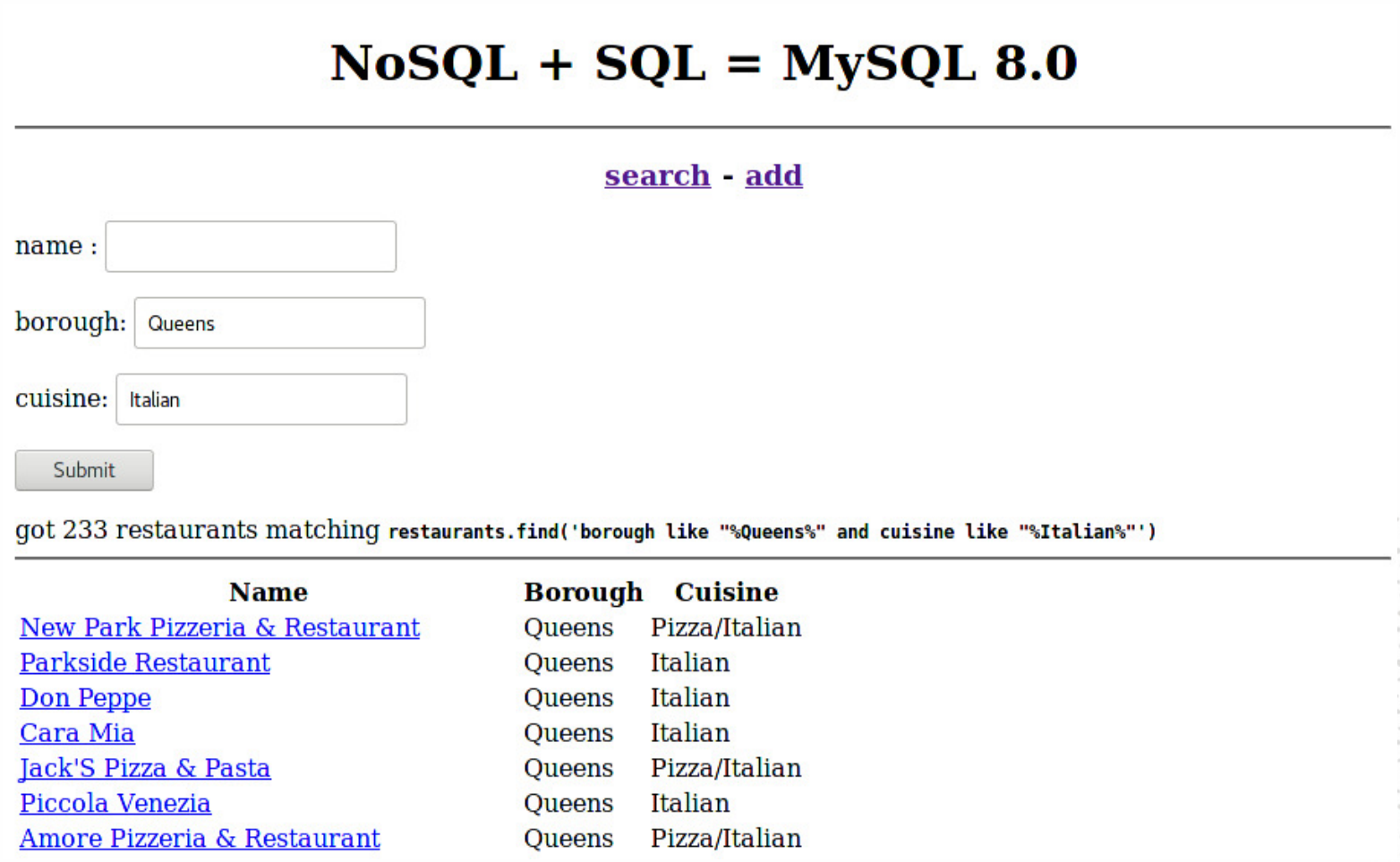
# Syntax slightly different than MongoDB

```
MySQL 127.0.0.1:33060 JS restaurants.find("cuisine='French' AND borough!='Manhattan'").
fields(["name","cuisine","borough"]).limit(2)
[
    {
        "borough": "Queens",
        "cuisine": "French",
        "name": "La Baraka Restaurant"
    },
    {
        "borough": "Queens",
        "cuisine": "French",
        "name": "Air France Lounge"
    }
]
2 documents in set (0.00 sec)
```

```
> db.restaurants.find({"cuisine": "French", "borough": { $not: /^Manhattan/} },
... {"_id":0, "name": 1, "cuisine": 1, "borough": 1}).limit(2)
{ "borough" : "Queens", "cuisine" : "French", "name" : "La Baraka Restaurant" }
{ "borough" : "Queens", "cuisine" : "French", "name" : "Air France Lounge" }
```


Copyright @ 2020 Oracle and/or its affiliates.

# And for developers ?





Copyright @ 2020 Oracle and/or its affiliates.

# And for developers ?

```
$session = mysql_xdevapi\getSession("mysqlx://fred:MyP@ssw0rd%@localhost");
$schema = $session->getSchema("docstore");
$collection = $schema->getCollection("restaurants");
$results = $collection->find($search)->execute()->fetchAll();
...
foreach ($results as $doc) {
   echo "<tr><td><a href='?id=${doc[_id]}'>${doc[name]}</a></td>";
   echo "<td>${doc[borough]}</td><td>${doc[cuisine]}</td></tr>";
}
```


Copyright @ 2020 Oracle and/or its affiliates.

# And for developers ?

```
$session = mysql_xdevapi\getSession("mysqlx://fred:MyP@ssw0rd%@localhost");
$schema = $session->getSchema("docstore");
$collection = $schema->getCollection("restaurants");
$results = $collection->find($search)->execute()->fetchAll();
...
foreach ($results as $doc) {
   echo "<tr><td><a href='?id=${doc[_id]}'>${doc[name]}</a></td>";
   echo "<td>${doc[borough]}</td><td>${doc[cuisine]}</td></tr>";
}
```

## Easy, using only CRUD operations !


Copyright @ 2020 Oracle and/or its affiliates.

# And for developers ?

```
$session = mysql_xdevapi\getSession("mysqlx://fred:MyP@ssw0rd%@localhost");
$schema = $session->getSchema("docstore");
$collection = $schema->getCollection("restaurants");
$results = $collection->find($search)->execute()->fetchAll();
...
foreach ($results as $doc) {
   echo "<tr><td><a href='?id=${doc[_id]}'>${doc[name]}</a></td>";
   echo "<td>${doc[borough]}</td><td>${doc[cuisine]}</td></tr>";
}
```

**Easy, using only CRUD operations !**
**Not a single SQL statement !**


Copyright @ 2020 Oracle and/or its affiliates.

# CRUD operations

```
MySQL 127.0.0.1:33060 JS restaurants.remove("cuisine='French' AND borough!='Manhattan'"
).limit(2)
Query OK, 2 items affected (0.16 sec)

MySQL 127.0.0.1:33060 JS restaurants.find("cuisine='French' AND borough!='Manhattan'").
fields(["name","cuisine","_id"]).limit(2)
[
    {
        "_id": "5943c83e1adc2605594170aa",
        "cuisine": "French",
        "name": "Bar Tabac"
    },
    {
        "_id": "5943c83e1adc260559417255",
        "cuisine": "French",
        "name": "Tournesol"
    }
]
2 documents in set (0.01 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# CRUD operations for collections

## Add a document

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

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


Copyright @ 2020 Oracle and/or its affiliates.

# CRUD operations for collections

## Modify a document

```
collection.modify('name = :name')
    .bind('name', 'fred')
    .set('age', 43)
    .sort('name ASC')
    .limit(1)
    .execute()
```

```
collection.modify('name = :name')
    .bind('name', 'fred')
    .patch({ age: 43, active: false })
    .sort('name DESC')
    .limit(1)
    .execute()
```


Copyright @ 2020 Oracle and/or its affiliates.

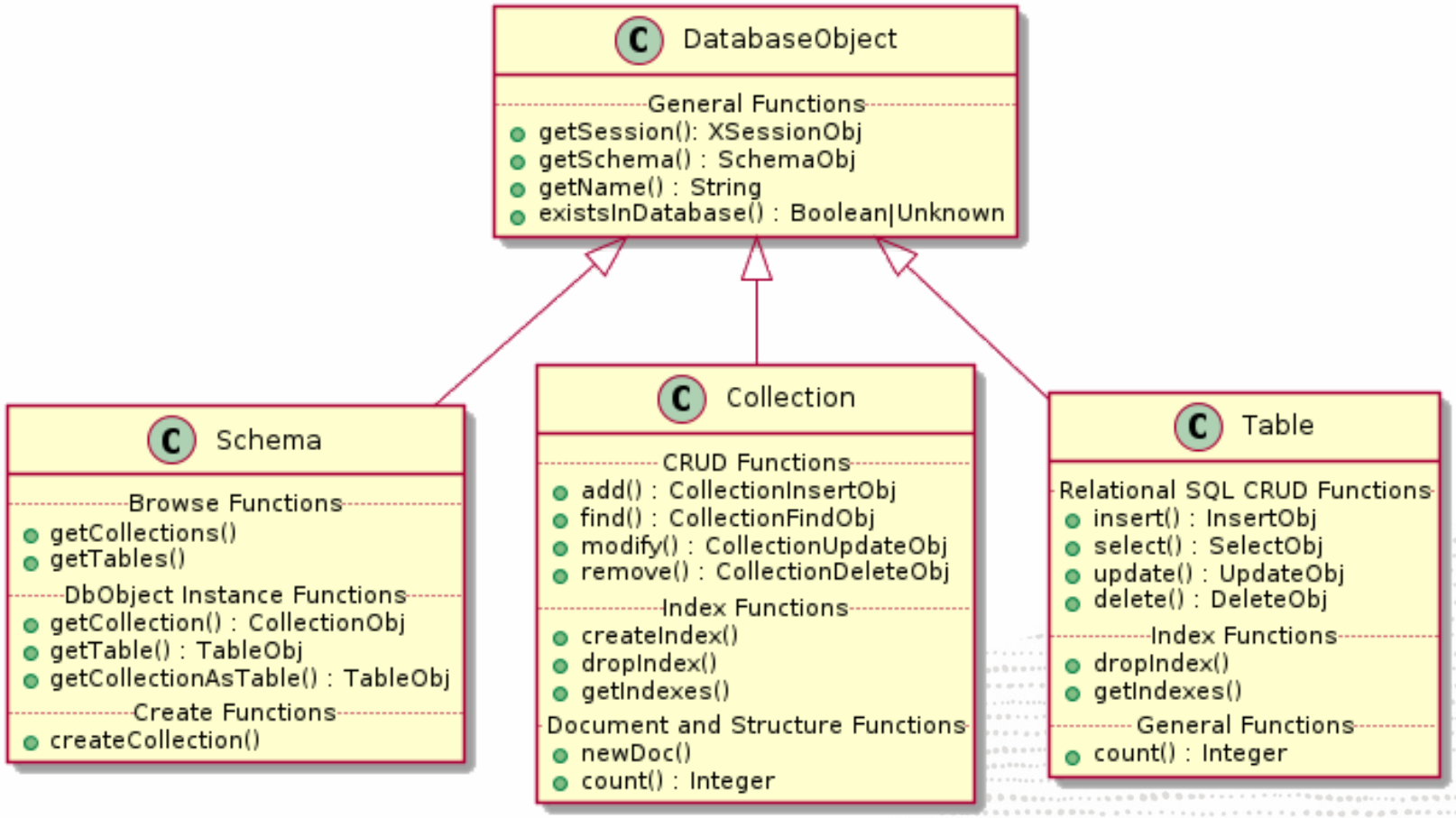
# CRUD operations for collections

## Remove a document

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


Copyright @ 2020 Oracle and/or its affiliates.

# MySQL Document Store Objects Summary





Copyright @ 2020 Oracle and/or its affiliates.

All you need to know is here:

https://dev.mysql.com/doc/x-devapi-userguide/en/crud-operations-overview.html

 Copyright @ 2020 Oracle and/or its affiliates.

# MySQL Document Store is Full ACID Compliant

we do care about your data


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Document Store Full ACID !

It relies on the proven MySQL InnoDB's strength & robustness:


Copyright @ 2020 Oracle and/or its affiliates.

# Document Store Full ACID !

It relies on the proven MySQL InnoDB's strength & robustness:

- `innodb_flush_log_at_trx_commit = 1`


Copyright @ 2020 Oracle and/or its affiliates.

# Document Store Full ACID !

It relies on the proven MySQL InnoDB's strength & robustness:

- `innodb_flush_log_at_trx_commit = 1`
- `innodb_doublewrite = ON`


Copyright @ 2020 Oracle and/or its affiliates.

# Document Store Full ACID !

It relies on the proven MySQL InnoDB's strength & robustness:

- `innodb_flush_log_at_trx_commit = 1`
- `innodb_doublewrite = ON`
- `sync_binlog = 1`


Copyright @ 2020 Oracle and/or its affiliates.

# Document Store Full ACID !

It relies on the proven MySQL InnoDB's strength & robustness:

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


Copyright @ 2020 Oracle and/or its affiliates.

# Document Store Full ACID !

It relies on the proven MySQL InnoDB's strength & robustness:

- `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 !


Copyright @ 2020 Oracle and/or its affiliates.

# Full ACID - Transactions support

```
MySQL localhost:33060+ fred JS session.startTransaction()
Query OK, 0 rows affected (0.0006 sec)

MySQL localhost:33060+ fred JS test.add({name: 'the René'})
Query OK, 1 item affected (0.1661 sec)

MySQL localhost:33060+ fred JS test.find()
[
    {
        "_id": "00005ade551100000000000001",
        "name": "fred"
    },
    {
        "_id": "00005ade551100000000000002",
        "name": "the René"
    }
]
2 documents in set (0.0019 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Full ACID - Transactions support

```
MySQL  localhost:33060+  fred  JS  session.rollback()
Query OK, 0 rows affected (0.0992 sec)

MySQL  localhost:33060+  fred  JS  test.find()
[
    {
        "_id": "00005ade551100000000000001",
        "name": "fred"
    }
]
1 document in set (0.0130 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# OK we have Document Store, CRUD and ACID

but what makes MySQL Document Store unique ?


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Challenge: list the best restaurant of each type of food and show the top 10, with the best one first !

don't forget that all these restaurants are just JSON documents


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL as SQL - aggregation

```
MySQL 127.0.0.1:33060+ docstore SQL 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 DESC) AS `rank`
                                FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10;
+----------------------+----------------------------------+-----------+------+
| name                 | cuisine                          | avg_score | rank |
+----------------------+----------------------------------+-----------+------+
| Juice It Health Bar  | Juice, Smoothies, Fruit Salads   |   75.0000 |    1 |
| Golden Dragon Cuisine| Chinese                          |   73.0000 |    1 |
| Palombo Pastry Shop  | Bakery                           |   69.0000 |    1 |
| Go Go Curry          | Japanese                         |   65.0000 |    1 |
| K & D Internet Inc   | Café/Coffee/Tea                  |   61.0000 |    1 |
| Koyla                | Middle Eastern                   |   61.0000 |    1 |
| Ivory D O S  Inc     | Other                            |   60.0000 |    1 |
| Espace               | American                         |   56.0000 |    1 |
| Tacos Al Suadero     | Mexican                          |   52.0000 |    1 |
| Rose Pizza           | Pizza                            |   52.0000 |    1 |
+----------------------+----------------------------------+-----------+------+
```


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL as SQL - aggregation

```
MySQL  127.0.0.1:33060+  docstore  SQL  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 DESC) AS `rank`
                                        FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10;
```

Common Table Expression (CTE)

| name | cuisine | avg_score | rank |
|---|---|---|---|
| Juice It Health Bar | Juice, Smoothies, Fruit Salads | 75.0000 | 1 |
| Golden Dragon Cuisine | Chinese | 73.0000 | 1 |
| Palombo Pastry Shop | Bakery | 69.0000 | 1 |
| Go Go Curry | Japanese | 65.0000 | 1 |
| K & D Internet Inc | Café/Coffee/Tea | 61.0000 | 1 |
| Koyla | Middle Eastern | 61.0000 | 1 |
| Ivory D O S  Inc | Other | 60.0000 | 1 |
| Espace | American | 56.0000 | 1 |
| Tacos Al Suadero | Mexican | 52.0000 | 1 |
| Rose Pizza | Pizza | 52.0000 | 1 |


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL as SQL - aggregation

```
MySQL  127.0.0.1:33060+  docstore  SQL  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 DESC) AS `rank`
                                        FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10;
```

Common Table Expression (CTE)

Window Function

| name | cuisine | avg_score | rank |
|---|---|---|---|
| Juice It Health Bar | Juice, Smoothies, Fruit Salads | 75.0000 | 1 |
| Golden Dragon Cuisine | Chinese | 73.0000 | 1 |
| Palombo Pastry Shop | Bakery | 69.0000 | 1 |
| Go Go Curry | Japanese | 65.0000 | 1 |
| K & D Internet Inc | Café/Coffee/Tea | 61.0000 | 1 |
| Koyla | Middle Eastern | 61.0000 | 1 |
| Ivory D O S  Inc | Other | 60.0000 | 1 |
| Espace | American | 56.0000 | 1 |
| Tacos Al Suadero | Mexican | 52.0000 | 1 |
| Rose Pizza | Pizza | 52.0000 | 1 |


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL as SQL - aggregation

```
MySQL  127.0.0.1:33060+  docstore  SQL
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 DESC) AS `rank`
FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10;
```

Common Table Expression (CTE)

Window Function

| name | cuisine | avg_score | rank |
|---|---|---|---|
| Juice It Health Bar | Juice, Smoothies, Fruit Salads | 75.0000 | 1 |
| Golden Dragon Cuisine | Chinese | 73.0000 | 1 |
| Palombo Pastry Shop | Bakery | 69.0000 | 1 |
| Go Go Curry | Japanese | 65.0000 | 1 |
| K & D Internet Inc | Café/Coffee/Tea | 61.0000 | 1 |
| Koyla | Middle Eastern | 61.0000 | 1 |
| Ivory D O S  Inc | Other | 60.0000 | 1 |
| Espace | American | 56.0000 | 1 |
| Tacos Al Suadero | Mexican | 52.0000 | 1 |
| Rose Pizza | Pizza | 52.0000 | 1 |


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL or SQL

You have the possibility to write clean and neat code:


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL or SQL

You have the possibility to write clean and neat code:

```
$results = $collection->find('cuisine like "italian"')->execute()->fetchAll();
```


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL or SQL

You have the possibility to write clean and neat code:

```
$results = $collection->find('cuisine like "italian"')->execute()->fetchAll();
```

Or use SQL only when it's really needed:


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL or SQL

You have the possibility to write clean and neat code:

```
$results = $collection->find('cuisine like "italian"')->execute()->fetchAll();
```

Or use SQL only when it's really needed:

```
$results = $session->sql('WITH cte1 AS (SELECT doc->>"$.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 docstore.restaurants)
  SELECT *, RANK()
  OVER ( PARTITION BY cuisine ORDER BY avg_score) AS `rank`
  FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10;')->execute();
```


Copyright @ 2020 Oracle and/or its affiliates.

# NoSQL or SQL

You have the possibility to write clean and neat code:

```
$results = $collection->find('cuisine like "italian"')->execute()->fetchAll();
```

Or use SQL only when it's really needed:

```
$results = $session->sql('WITH cte1 AS (SELECT doc->>"$.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 docstore.restaurants)
  SELECT *, RANK()
  OVER ( PARTITION BY cuisine ORDER BY avg_score) AS `rank`
  FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10;')->execute();
```

**All in the same MySQL X Session !**


Copyright @ 2020 Oracle and/or its affiliates.

# You can mix NoSQL & SQL as you want

```
MySQL 8.0.11  localhost:33060+  docstore  2018-09-11 12:11:00  Threads running: 2
JS session.sql("select doc->>'$.name', doc->>'$.cuisine' from restaurants where doc->>'$.cuisi
ne'='Italian' limit 2");
+--------------------+--------------------+
| doc->>'$.name'     | doc->>'$.cuisine'  |
+--------------------+--------------------+
| Marchis Restaurant | Italian            |
| Crystal Room       | Italian            |
+--------------------+--------------------+
2 rows in set (0.0021 sec)

MySQL 8.0.11  localhost:33060+  docstor…  2018-09-11 12:11:23  Threads running: 2
JS db.restaurants.find("cuisine='Italian'").fields('name','cuisine').limit(2)
[
    {
        "cuisine": "Italian",
        "name": "Marchis Restaurant"
    },
    {
        "cuisine": "Italian",
        "name": "Crystal Room"
    }
]
2 documents in set (0.0256 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Best of Both Worlds: JSON_TABLE

What are the maximum 10 ratings ever given to a restaurant?

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-29 11:08:29
SQL> SELECT doc->>'$.name', (SELECT MAX(x) FROM JSON_TABLE(doc, '$.grades[*].score' COLUMNS (x INT PATH '$')) jt) AS max_score
     FROM restaurants order by 2 desc limit 10;
+---------------------------------+-----------+
| doc->>'$.name'                  | max_score |
+---------------------------------+-----------+
| Murals On 54/Randolphs'S        |       131 |
| Baluchi'S Indian Food           |        98 |
| Bella Napoli                    |        98 |
| Gandhi                          |        92 |
| Concrete Restaurant             |        90 |
| West 79Th Street Boat Basin Cafe |        89 |
| D & Y Restaurant                |        86 |
| Spicy Shallot                   |        84 |
| Bistro Caterers                 |        84 |
| La Potencia Restaurant          |        82 |
+---------------------------------+-----------+
10 rows in set (0.9448 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Best of Both Worlds: JSON_TABLE

What are the maximum 10 ratings ever given to a restaurant?

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-29 11:08:29
SQL> SELECT doc->>'$.name', (SELECT MAX(x) FROM JSON_TABLE(doc, '$.grades[*].score' COLUMNS (x INT PATH '$')) jt) AS max_score
     FROM restaurants order by 2 desc limit 10;
+---------------------------------+-----------+
| doc->>'$.name'                  | max_score |
+---------------------------------+-----------+
| Murals On 54/Randolphs'S        |       131 |
| Baluchi'S Indian Food           |        98 |
| Bella Napoli                    |        98 |
| Gandhi                          |        92 |
| Concrete Restaurant             |        90 |
| West 79Th Street Boat Basin Cafe |        89 |
| D & Y Restaurant                |        86 |
| Spicy Shallot                   |        84 |
| Bistro Caterers                 |        84 |
| La Potencia Restaurant          |        82 |
+---------------------------------+-----------+
10 rows in set (0.9448 sec)
```

Cool… but my app only processes JSON !


Copyright @ 2020 Oracle and/or its affiliates.

# Best of Both Worlds: JSON_TABLE (2)

With JSON output:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-29 11:12:44
SQL > SELECT JSON_OBJECT("name",doc->>'$.name',"max", (SELECT MAX(x) FROM JSON_TABLE(doc, '$.grades[*].score' COLUMNS (x INT PATH '$')) jt))
    > FROM restaurants order by (SELECT MAX(x) FROM JSON_TABLE(doc, '$.grades[*].score' COLUMNS (x INT PATH '$')) jt) desc limit 10;
+-----------------------------------------------------------------------------------------------------------------------------------------+
| JSON_OBJECT("name",doc->>'$.name',"max", (SELECT MAX(x) FROM JSON_TABLE(doc, '$.grades[*].score' COLUMNS (x INT PATH '$')) jt)) |
+-----------------------------------------------------------------------------------------------------------------------------------------+
| {"max": 131, "name": "Murals On 54/Randolphs'S"}                                                                                        |
| {"max": 98, "name": "Bella Napoli"}                                                                                                     |
| {"max": 98, "name": "Baluchi'S Indian Food"}                                                                                            |
| {"max": 92, "name": "Gandhi"}                                                                                                           |
| {"max": 90, "name": "Concrete Restaurant"}                                                                                              |
| {"max": 89, "name": "West 79Th Street Boat Basin Cafe"}                                                                                 |
| {"max": 86, "name": "D & Y Restaurant"}                                                                                                 |
| {"max": 84, "name": "Spicy Shallot"}                                                                                                    |
| {"max": 84, "name": "Bistro Caterers"}                                                                                                  |
| {"max": 82, "name": "Cafe R"}                                                                                                           |
+-----------------------------------------------------------------------------------------------------------------------------------------+
10 rows in set (0.0571 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# And the DBA ?

MySQL Document Store


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Role of the MySQL Document Store DBA ?


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA ?

- find the non optimal lookups
- create the necessary indexes based on the results



Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA

## Looking for full table scans:

```
SELECT schema_name, sum_rows_examined, (sum_rows_examined/exec_count) avg_rows_call,
       format_time(total_latency) tot_lat, exec_count,
       format_time(total_latency/exec_count) AS latency_per_call, query_sample_text
  FROM sys.x$statements_with_full_table_scans AS t1
  JOIN performance_schema.events_statements_summary_by_digest AS t2
    ON t2.digest=t1.digest
 WHERE schema_name = "docstore" AND query_sample_text LIKE '%select%'
 ORDER BY (total_latency/exec_count) desc\G
```


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA

## Looking for full table scans:

```
SELECT schema_name, sum_rows_examined, (sum_rows_examined/exec_count) avg_rows_call,
       format_time(total_latency) tot_lat, exec_count,
       format_time(total_latency/exec_count) AS latency_per_call, query_sample_text
  FROM sys.x$statements_with_full_table_scans AS t1
  JOIN performance_schema.events_statements_summary_by_digest AS t2
    ON t2.digest=t1.digest
 WHERE schema_name = "docstore" AND query_sample_text LIKE '%select%'
 ORDER BY (total_latency/exec_count) desc\G
```

More info at https://lefred.be/content/mysql-8-0-if-i-should-optimize-only-one-query-on-my-application-which-one-should-it-be/


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (2)

```
MySQL 8.0.17 localhost:33060+ sys 2019-08-28 18:24:36
SQL > SELECT schema_name, sum_rows_examined, (sum_rows_examined/exec_count) avg_rows_call, format_time(total_latency) tot_lat, exec_count,
      format_time(total_latency/exec_count) AS latency_per_call, query_sample_text
    > FROM sys.x$statements_with_full_table_scans AS t1 JOIN performance_schema.events_statements_summary_by_digest AS t2
    >   ON t2.digest=t1.digest WHERE schema_name = "docstore" and query_sample_text like '%select%' ORDER BY (total_latency/exec_count) desc\G
*************************** 1. row ***************************
      schema_name: docstore
sum_rows_examined: 169550
    avg_rows_call: 169550.0000
          tot_lat: 1.46 s
       exec_count: 1
 latency_per_call: 1.46 s
query_sample_text: with cte1 AS (select doc->>"$.name" AS nam, doc->>"$.cuisine" as cuisine, (SELECT avg(score) from JSON_TABLE(doc, "$.grades
[*]"
COLUMNS (score INT PATH "$.score")) AS r) AS avg_score FROM restaurants) SELECT *, RANK() OVER ( PARTITION BY cuisine ORDER BY
avg_score DESC) AS `rank` FROM cte1 ORDER BY `rank`, avg_score DESC LIMIT 10
*************************** 2. row ***************************
      schema_name: docstore
sum_rows_examined: 50935
    avg_rows_call: 8489.1667
          tot_lat: 1.97 s
       exec_count: 6
 latency_per_call: 327.64 ms
query_sample_text: SELECT JSON_OBJECT('name', JSON_EXTRACT(doc,'$.name'),'cuisine', JSON_EXTRACT(doc,'$.cuisine')) AS doc FROM `docstore`.`res
taurants` WHERE (JSON_UNQUOTE(JSON_EXTRACT(doc,'$.cuisine')) IN ('Belgian','Chineese')) LIMIT 0, 2
```

name of the schema
used for my documents


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (3)

## Getting the Query Execution Plan:

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 18:27:13
SQL> EXPLAIN SELECT JSON_OBJECT('name', JSON_EXTRACT(doc,'$.name'),'cuisine', JSON_EXTRACT(doc,'$.cuisine')) AS doc FROM `docstore`.`restaurant
ts` WHERE (JSON_UNQUOTE(JSON_EXTRACT(doc,'$.cuisine')) IN ('Belgian','Chineese')) LIMIT 0, 2\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: restaurants
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 24026
     filtered: 100
        Extra: Using where
```


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (3)

## Getting the Query Execution Plan:

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 18:27:13
SQL> EXPLAIN SELECT JSON_OBJECT('name', JSON_EXTRACT(doc,'$.name'),'cuisine', JSON_EXTRACT(doc,'$.cuisine')) AS doc FROM `docstore`.`restaurant
ts` WHERE (JSON_UNQUOTE(JSON_EXTRACT(doc,'$.cuisine')) IN ('Belgian','Chineese')) LIMIT 0, 2\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: restaurants
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 24026
     filtered: 100
        Extra: Using where
```

It's indeed a full table scan !


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (4)

## MySQL supports also EXPLAIN ANALYZE !!

```
MySQL  localhost:33060+  docstore  2020-06-08 14:14:12
SQL  explain analyze SELECT JSON_OBJECT('name', JSON_EXTRACT(doc,'$.name'),'cuisine', JSON_EXTRACT(doc,'$.cuisi
ne')) AS doc FROM docstore.restaurants WHERE (JSON_UNQUOTE(JSON_EXTRACT(doc,'$.cuisine')) IN ('Belgian','Chineese
')) LIMIT 0,2\G
*************************** 1. row ***************************
EXPLAIN: -> Limit: 2 row(s)  (actual time=22.021..22.021 rows=0 loops=1)
    -> Filter: (json_unquote(json_extract(restaurants.doc,'$.cuisine')) in ('Belgian','Chineese'))  (cost=3559.30
 rows=24353) (actual time=22.019..22.019 rows=0 loops=1)
        -> Table scan on restaurants  (cost=3559.30 rows=24353) (actual time=0.049..13.195 rows=25357 loops=1)

1 row in set (0.0229 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (5)

## Create Indexes:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 19:58:01
JS restaurants.createIndex('cuisine_idx',{"fields": {"field": "$.cuisine", "required": true, type:"text(20)"}})
Query OK, 0 rows affected (3.0933 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (5)

## Create Indexes:

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 19:58:01
JS restaurants.createIndex('cuisine_idx',{"fields": {"field": "$.cuisine", "required": true, type:"text(20)"}})
Query OK, 0 rows affected (3.0933 sec)
```

For the "old" experienced MySQL DBAs, you can instead use:

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 20:01:58
SQL ALTER TABLE restaurants ADD COLUMN cuisine text GENERATED ALWAYS AS (doc->>"$.cuisine") VIRTUAL NOT NULL,
    ADD INDEX cuisine_idx(cuisine(20));
Query OK, 0 rows affected (0.7199 sec)

Records: 0  Duplicates: 0  Warnings: 0
```


Copyright @ 2020 Oracle and/or its affiliates.

# INFO: JSON Notation Shortcut

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 20:13:26
SQL SELECT json_unquote(json_extract(`doc`,_utf8mb4'$.name')) from restaurants limit 1;
+----------------------------------------------------+
| json_unquote(json_extract(`doc`,_utf8mb4'$.name')) |
+----------------------------------------------------+
| Morris Park Bake Shop                              |
+----------------------------------------------------+
1 row in set (0.1182 sec)
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 20:13:29
SQL SELECT json_unquote(doc->'$.name') from restaurants limit 1;
+-----------------------------+
| json_unquote(doc->'$.name') |
+-----------------------------+
| Morris Park Bake Shop       |
+-----------------------------+
1 row in set (0.0148 sec)
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 20:14:19
SQL SELECT doc->>'$.name' from restaurants limit 1;
+----------------------+
| doc->>'$.name'       |
+----------------------+
| Morris Park Bake Shop |
+----------------------+
1 row in set (0.0005 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# Role of the MySQL Document Store DBA (6)

## Result:

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 18:28:24
SQL> EXPLAIN SELECT JSON_OBJECT('name', JSON_EXTRACT(doc,'$.name'),'cuisine', JSON_EXTRACT(doc,'$.cuisine')) AS doc FROM `docstore`.`restaurant
ts` WHERE (JSON_UNQUOTE(JSON_EXTRACT(doc,'$.cuisine')) IN ('Belgian','Chineese')) LIMIT 0, 2\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: restaurants
   partitions: NULL
         type: range
possible_keys: cuisine_idx
          key: cuisine_idx
      key_len: 82
          ref: NULL
         rows: 2
     filtered: 100
        Extra: Using where
```


Copyright @ 2020 Oracle and/or its affiliates.

# For the curious: a collection in SQL:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 20:14:55
SQL> SHOW CREATE TABLE restaurants\G
*************************** 1. row ***************************
       Table: restaurants
Create Table: CREATE TABLE `restaurants` (
  `doc` json DEFAULT NULL,
  `_id` varbinary(32) GENERATED ALWAYS AS (json_unquote(json_extract(`doc`,_utf8mb4'$._id'))) STORED NOT NULL,
  `cuisine` text GENERATED ALWAYS AS (json_unquote(json_extract(`doc`,_utf8mb4'$.cuisine'))) VIRTUAL NOT NULL,
  PRIMARY KEY (`_id`),
  KEY `cuisine_idx` (`cuisine`(20))
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```


Copyright @ 2020 Oracle and/or its affiliates.

# Since MySQL 8.0.17

- JSON Array Indexes
- JSON Schema
  - `JSON_SCHEMA_VALID(<json schema>,<json doc>)`
  - `JSON_SCHEMA_VALIDATION_REPORT(<json schema>,<json doc>)`


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Array Indexes

- Index of a JSON array
  - A functional index over a JSON expression
  - The expression evaluates to an array
- Several index entries per row
  - One index entry per array element
  - General mechanism, currently used for JSON arrays

- Used to speed up array lookups
  - `JSON_CONTAINS(...)`
  - `JSON_OVERLAPS(...)`
  - `MEMBER OF (...)`


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Array Indexes

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-29 11:29:44
SQL select doc->>'$.name', doc->'$.grades[*].score' from test where 38 member of (doc->'$.grades[*].score');
+------------------------+--------------------------+
| doc->>'$.name'         | doc->'$.grades[*].score' |
+------------------------+--------------------------+
| Brunos On The Boulevard | [38, 10, 7, 13]          |
+------------------------+--------------------------+
1 row in set (0.0086 sec)
MySQL 8.0.17 localhost:33060+ docstore 2019-08-29 11:30:01
SQL EXPLAIN select doc->>'$.name', doc->>'$.grades[*].score' from test where 38 member of (doc->'$.grades[*].score')\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
   partitions: NULL
         type: ALL
possible_keys: NULL
          key: NULL
      key_len: NULL
          ref: NULL
         rows: 10
     filtered: 100
        Extra: Using where
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Array Indexes (2)

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-29 11:30:06
SQL> ALTER TABLE test ADD INDEX score_idx((CAST(doc->'$.grades[*].score' AS SIGNED ARRAY )));
Query OK, 0 rows affected (0.1210 sec)

Records: 0  Duplicates: 0  Warnings: 0
MySQL 8.0.17  localhost:33060+  docstore  2019-08-29 11:30:52
SQL> EXPLAIN select doc->>'$.name', doc->>'$.grades[*].score' from test where 38 member of (doc->'$.grades[*].score')\G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: test
   partitions: NULL
         type: ref
possible_keys: score_idx
          key: score_idx
      key_len: 9
          ref: const
         rows: 1
     filtered: 100
        Extra: Using where
1 row in set, 1 warning (0.0007 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON_VALUE (since 8.0.21)

This function is described in SQL 2016, chapter 6.27.

```
MySQL  localhost:33060+  docstore  2020-10-07 01:00:30
SQL  SELECT JSON_VALUE(doc, '$.address.zipcode' RETURNING UNSIGNED)
     FROM restaurants LIMIT 3;
+-----------------------------------------------------+
| JSON_VALUE(doc, '$.address.zipcode' RETURNING UNSIGNED) |
+-----------------------------------------------------+
|                                                NULL |
|                                               10462 |
|                                               11225 |
+-----------------------------------------------------+
3 rows in set (0.0004 sec)
MySQL  localhost:33060+  docstore  2020-10-07 01:01:38
SQL
```


Copyright @ 2020 Oracle and/or its affiliates.

# CHECK CONSTRAINTS

MySQL 8.0 supports Check Constraints:

```
MySQL  localhost:33060+  test  2019-10-25 14:38:45
SQL  CREATE TABLE japanese_food (
       id int auto_increment primary key,
       name varchar(20),
       note int CHECK (note > 0 AND note < 11));
Query OK, 0 rows affected (0.2753 sec)
```

```
MySQL  localhost:33060+  test  2019-10-25 14:39:30
SQL  INSERT INTO japanese_food VALUES (0, 'Sushi', 12);
ERROR: 3819: Check constraint 'japanese_food_chk_1' is violated.
MySQL  localhost:33060+  test  2019-10-25 14:43:34
SQL  INSERT INTO japanese_food VALUES (0, 'Sushi', -1);
ERROR: 3819: Check constraint 'japanese_food_chk_1' is violated.
MySQL  localhost:33060+  test  2019-10-25 14:43:57
SQL  INSERT INTO japanese_food VALUES (0, 'Sushi', 10);
Query OK, 1 row affected (0.0511 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Schema Validation

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 21:46:18
SQL > set @s='{"type": "object", "properties": { "cuisine": { "type": "string", "maxLength": 20} } }';
Query OK, 0 rows affected (0.0003 sec)
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 21:47:55
SQL > select doc->>"$.name", doc->>"$.cuisine", json_schema_valid(@s,doc) from test  ;
+--------------------------------+--------------------------------+-----------------------------+
| doc->>"$.name"                 | doc->>"$.cuisine"              | json_schema_valid(@s,doc)   |
+--------------------------------+--------------------------------+-----------------------------+
| Morris Park Bake Shop          | Bakery                         |                           1 |
| Wendy'S                        | Hamburgers                     |                           1 |
| Dj Reynolds Pub And Restaurant | Irish                          |                           1 |
| Riviera Caterer                | American                       |                           1 |
| Tov Kosher Kitchen             | Jewish/Kosher                  |                           1 |
| Brunos On The Boulevard        | American                       |                           1 |
| Kosher Island                  | Jewish/Kosher                  |                           1 |
| Wilken'S Fine Food             | Delicatessen                   |                           1 |
| Regina Caterers                | American                       |                           1 |
| Taste The Tropics Ice Cream    | Ice Cream, Gelato, Yogurt, Ices|                           0 |
+--------------------------------+--------------------------------+-----------------------------+
10 rows in set (0.0016 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Schema Validation (2)

```
MySQL 8.0.17 localhost:33060+ docstore 2019-08-28 21:51:31
SQL select doc->>"$.name", doc->>"$.cuisine", json_pretty(json_schema_validation_report(@s,doc)) from test where json_schema_valid(@s,doc)=0\G
*************************** 1. row ***************************
                              doc->>"$.name": Taste The Tropics Ice Cream
                           doc->>"$.cuisine": Ice Cream, Gelato, Yogurt, Ices
json_pretty(json_schema_validation_report(@s,doc)): {
  "valid": false,
  "reason": "The JSON document location '#/cuisine' failed requirement 'maxLength' at JSON Schema location '#/properties/cuisine'",
  "schema-location": "#/properties/cuisine",
  "document-location": "#/cuisine",
  "schema-failed-keyword": "maxLength"
}
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Schema Validation (3)

And the best of both worlds:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 21:56:26
SQL > ALTER TABLE test ADD CONSTRAINT `maxLengthCuisine`
      CHECK (JSON_SCHEMA_VALID('
        {"type": "object", "properties": { "cuisine":
          { "type": "string", "maxLength": 32} }, "required": ["cuisine"] }', `doc`) = 1);
Query OK, 10 rows affected (0.4174 sec)
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Schema Validation (3)

And the best of both worlds:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 21:56:26
SQL  ALTER TABLE test ADD CONSTRAINT `maxLengthCuisine`
     CHECK (JSON_SCHEMA_VALID('
       {"type": "object", "properties": { "cuisine":
         { "type": "string", "maxLength": 32} }, "required": ["cuisine"] }', `doc`) = 1);
Query OK, 10 rows affected (0.4174 sec)
```

And the result in action:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 22:04:19
JS  db.test.add({"name":"Test from lefred", "cuisine":"The Best Belgian-Italian-Junk Food in The World"})
ERROR: 3819: Check constraint 'maxLengthCuisine' is violated.
```


Copyright @ 2020 Oracle and/or its affiliates.

# JSON Schema Validation (3)

And the best of both worlds:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 21:56:26
SQL  ALTER TABLE test ADD CONSTRAINT `maxLengthCuisine`
     CHECK (JSON_SCHEMA_VALID('
       {"type": "object", "properties": { "cuisine":
         { "type": "string", "maxLength": 32} }, "required": ["cuisine"] }', `doc`) = 1);
Query OK, 10 rows affected (0.4174 sec)
```

And the result in action:

```
MySQL 8.0.17  localhost:33060+  docstore  2019-08-28 22:04:19
JS  db.test.add({"name":"Test from lefred", "cuisine":"The Best Belgian-Italian-Junk Food in The World"})
ERROR: 3819: Check constraint 'maxLengthCuisine' is violated.
```


Copyright @ 2020 Oracle and/or its affiliates.

# Conclusion

## what do I gain ?


Copyright @ 2020 Oracle and/or its affiliates.

@lefred

# Conclusion

This is the best of the two worlds in one product !

- Data integrity
- ACID Compliant
- Transactions
- SQL

- schemaless
- flexible data structure
- easy to start (CRUD)


Copyright @ 2020 Oracle and/or its affiliates.


http://lefred.be/content/top-10-reasons-for-nosql-with-mysql/


Copyright @ 2020 Oracle and/or its affiliates.

# Q & A



Copyright @ 2020 Oracle and/or its affiliates.