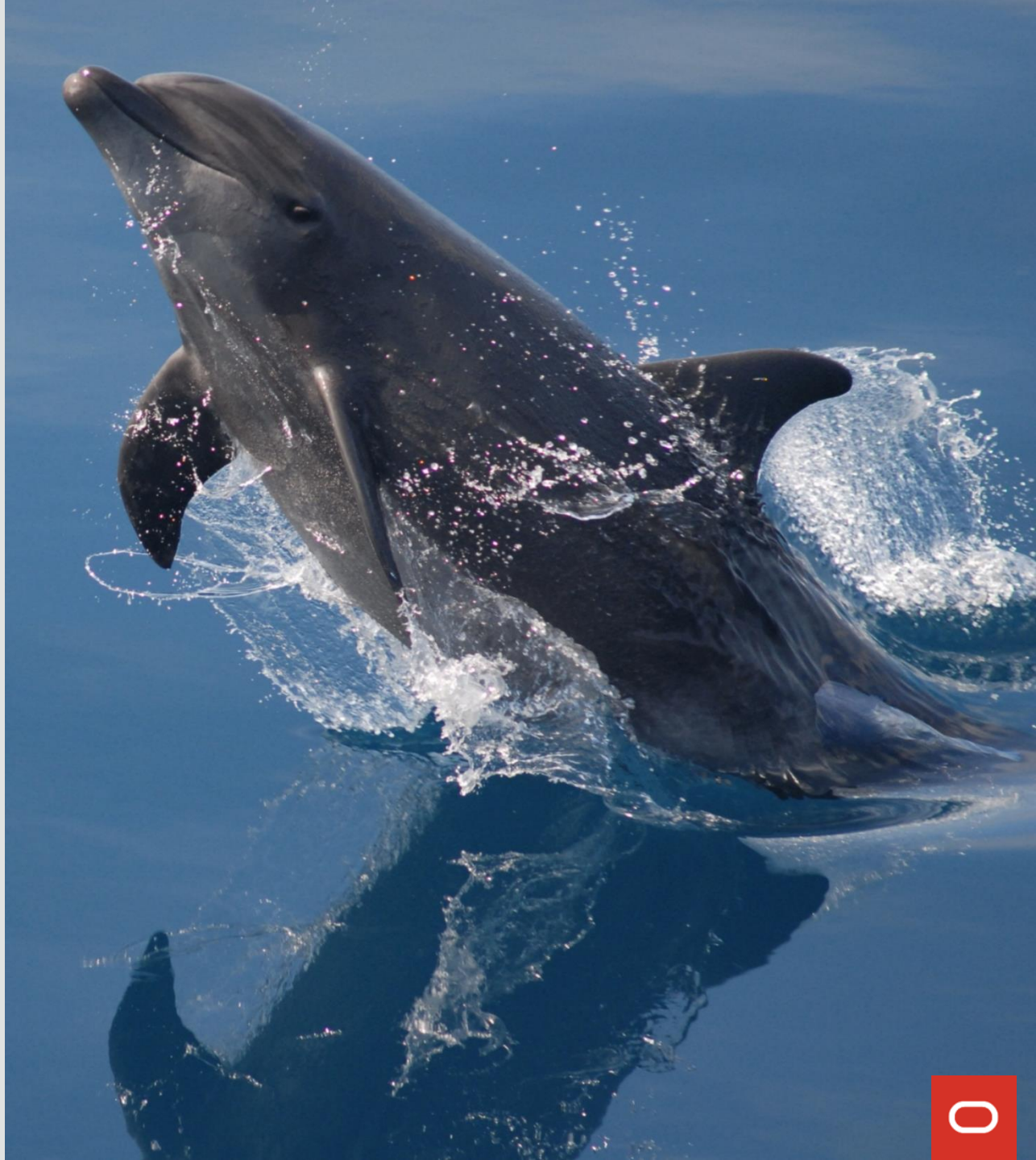
ORACLE

MySQL Goes to 9



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.

MySQL Enterprise Edition

Advanced Features

- Scalability
- High Availability
- Security
- Encryption + TDE
- Data Masking
- Firewall

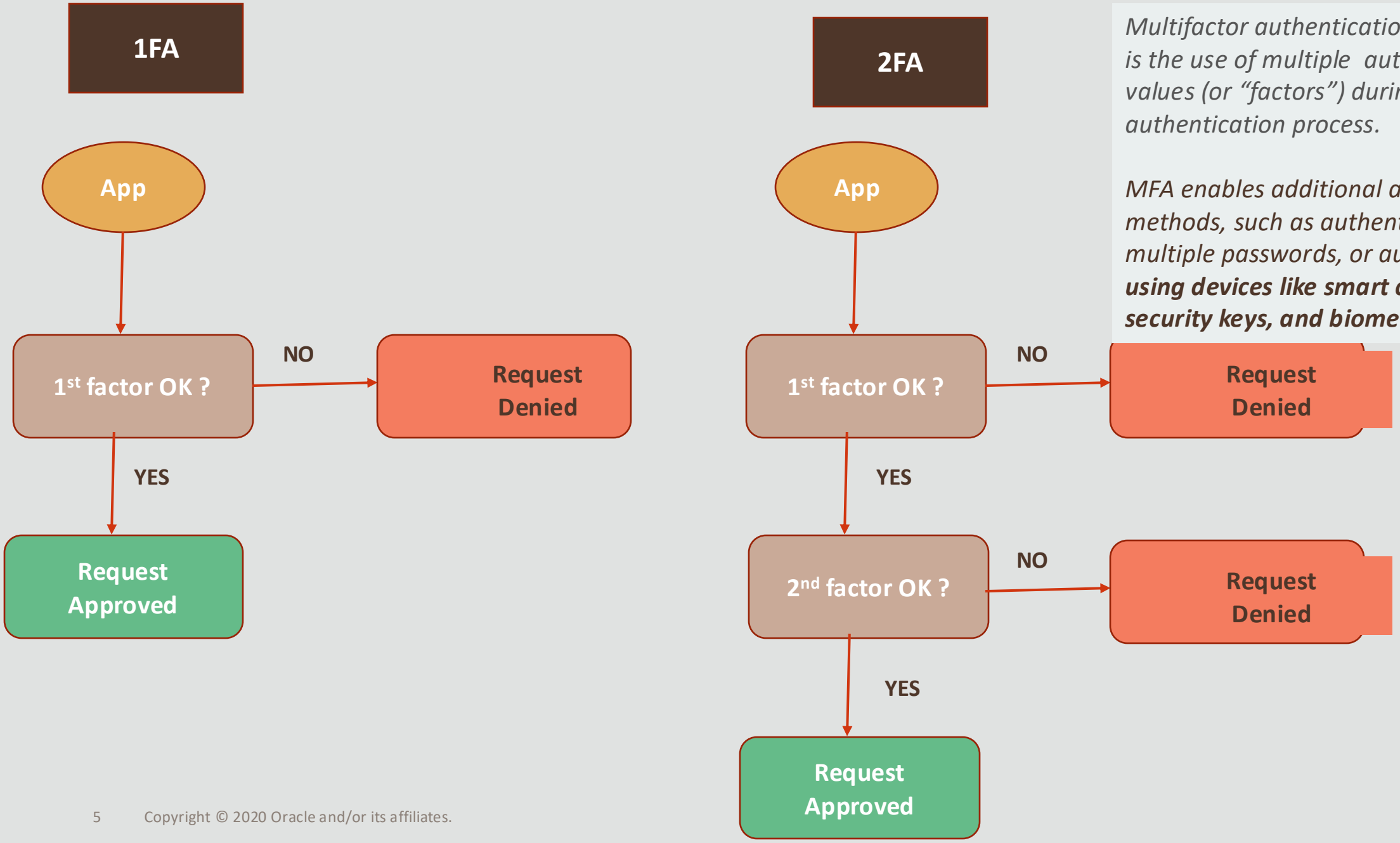
Management Tools

- Monitoring
- Backup
- Development
- Administration
- Migration

Support

- Technical Support
- Consultative Support
- Oracle Certifications

MySQL 9.0 Highlights



Multifactor authentication (MFA) is the use of multiple authentication values (or "factors") during the authentication process.

MFA enables additional authentication methods, such as authentication using multiple passwords, or authentication ***using devices like smart cards, security keys, and biometric readers****.*

Understanding Multifactor Authentication Support in MySQL 8.0.27

MFA -Elements

Something you know
- such as a **secret password** or passphrase

Something you have
- such as a **security key** or smart card.

Something you are
- a **biometric** characteristic such as a **fingerprint** or **facial scan**.

MFA –Authentication Policy

The **authentication_policy** system variable controls how many authentication factors can be used and the types of authentication permitted for each factor.

authentication_policy = ',authentication_fido,'*

SET GLOBAL authentication_policy=',*,';*

MFA –External Device Auth

The server side **authentication_fido** plugin enables authentication using devices.

authentication_fido also enables passwordless authentication, if it is the only authentication plugin used by an account.

server-side FIDO authentication plugin is included only in MySQL Enterprise Edition.

authentication_policy Value	Effective Policy
'*'	Permit only creating or altering accounts with one factor.
'*,*'	Permit only creating or altering accounts with two factors.
'*,*,*'	Permit only creating or altering accounts with three factors.
'*,'	Permit creating or altering accounts with one or two factors.
'*,,'	Permit creating or altering accounts with one, two, or three factors.
'*,*,'	Permit creating or altering accounts with two or three factors.
'*,auth_plugin'	Permit creating or altering accounts with two factors, where the first factor can be any authentication method, and the second factor must be the named plugin.
'auth_plugin,*,'	Permit creating or altering accounts with two or three factors, where the first factor must be the named plugin.
'auth_plugin,'	Permit creating or altering accounts with one or two factors, where the first factor must be the named plugin.
'auth_plugin,auth_plugin,auth_plugin'	Permits creating or altering accounts with three factors, where the factors must use the named plugins.

Using Multifactor Authentication

Suppose that you want an account to authenticate
first using the caching_sha2_password -default plugin, then
Second using the authentication_ldap_sasl SASL LDAP plugin.

Create User with MFA

```
CREATE USER ‘admin'@'localhost' IDENTIFIED WITH
caching_sha2_password BY '‘sha2_password’'
AND IDENTIFIED WITH authentication_ldap_sasl AS
'uid=u1_ldap,ou=People,dc=example,dc=com';
```

Login MySQL with MFA User with MFA

```
$> mysql --user=admin --password1 --password2
Enter password: (enter factor 1 password)

Enter password: (enter factor 2 password)
```

Suppose you want to add a third authentication factor.
This can be achieved by dropping and recreating the user with a third factor or by using ALTER USER *user* ADD *factor* syntax

```
ALTER USER 'alice'@'localhost' ADD 3 FACTOR IDENTIFIED WITH
authentication_fido;
```

changing the authentication method
from authentication_ldap_sasl to authetication_fido:

```
ALTER USER 'alice'@'localhost' MODIFY 2 FACTOR IDENTIFIED WITH
authentication_fido;
```

InnoDB GIPK mode

Since MySQL 8.0.30, MySQL supports generated invisible primary keys when running in ***GIPK mode*** *!*

GIPK mode is controlled by the sql_generate_invisible_primary_key *server system variable.*

When MySQL is running in GIPK mode, a primary key is added to a table by the server, the column and key name is always my_row_id.

InnoDB GIPK mode - example

```
MySQL > SELECT @@sql_generate_invisible_primary_key;
+-----------------------------------------+
| @@sql_generate_invisible_primary_key |
+-----------------------------------------+
|                                       1 |
+-----------------------------------------+

MySQL > CREATE TABLE rivierajug (name varchar(20), beers int unsigned);

MySQL > INSERT INTO rivierajug VALUES ('Yannis', 0), ('lefred',1);
Query OK, 2 rows affected (0.0073 sec)

MySQL > SELECT * FROM rivierajug;
+--------+-------+
| name   | beers |
+--------+-------+
| Yannis |     0 |
| lefred |     1 |
+--------+-------+
2 rows in set (0.0002 sec)
```

InnoDB GIPK mode - example (2)

```
MySQL > SHOW CREATE TABLE rivierajug\G
 *************************** 1. row ***************************
       Table: rivierajug
Create Table: CREATE TABLE `rivierajug` (
  `my_row_id` bigint unsigned NOT NULL AUTO_INCREMENT /*!80023 INVISIBLE */,
  `name` varchar(20) DEFAULT NULL,
  `beers` int unsigned DEFAULT NULL,
  PRIMARY KEY (`my_row_id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

```
MySQL > SELECT *, my_row_id FROM rivierajug;
+--------+-------+-----------+
| name   | beers | my_row_id |
+--------+-------+-----------+
| Yannis |     0 |         1 |
| lefred |     1 |         2 |
+--------+-------+-----------+
2 rows in set (0.0003 sec)
```

InnoDB GIPK mode - example (3)

It's also possible to hide it completely (for some legacy application that could rely on information_schema *and* SHOW CREATE TABLE*):*

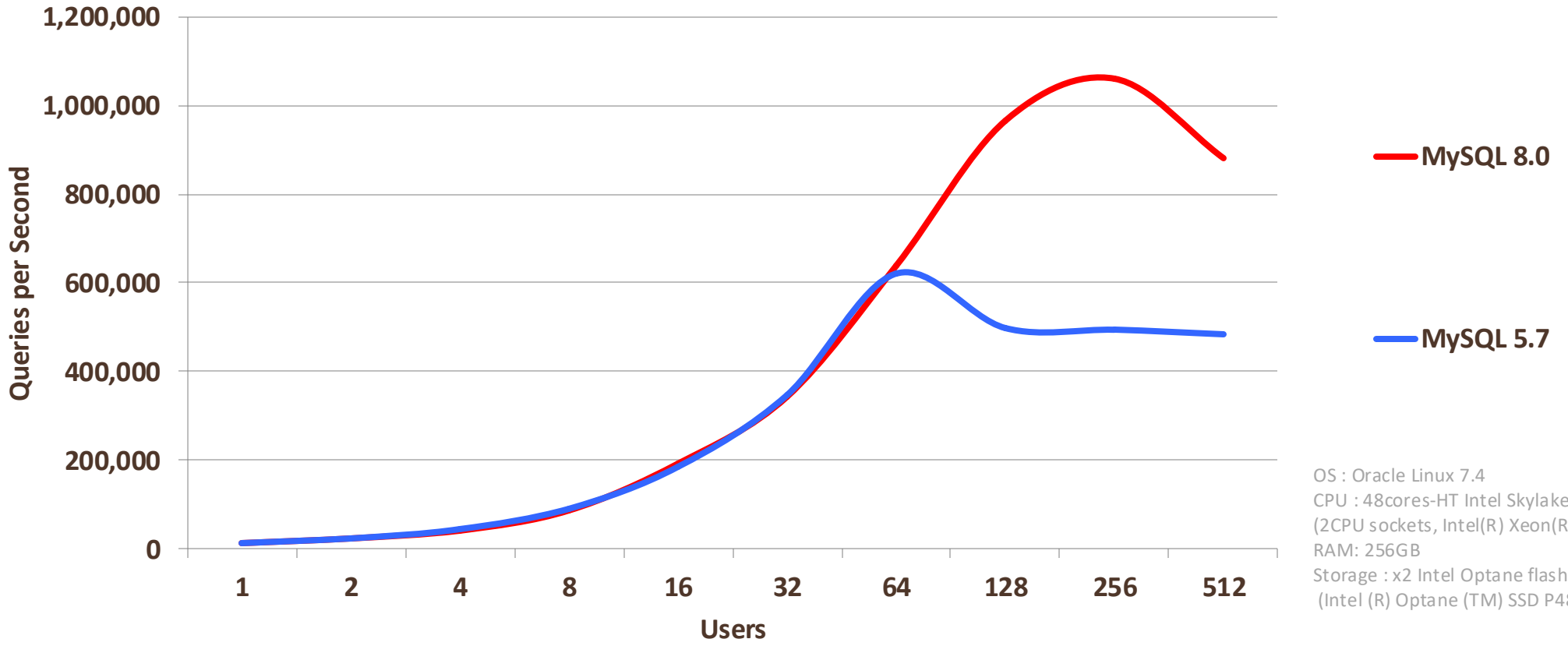
```
MySQL > SET show_gipk_in_create_table_and_information_schema = 0;

MySQL > SHOW CREATE TABLE rivierajug\G
 *************************** 1. row ***************************
       Table: rivierajug
Create Table: CREATE TABLE `rivierajug` (
  `name` varchar(20) DEFAULT NULL,
  `beers` int unsigned DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci
```

InnoDB GIPK mode - example (4)

```
MySQL > SELECT COLUMN_NAME, ORDINAL_POSITION, DATA_TYPE, COLUMN_KEY
        FROM INFORMATION_SCHEMA.COLUMNS
        WHERE TABLE_NAME = "rivierajug";
+-------------+------------------+-----------+------------+
| COLUMN_NAME | ORDINAL_POSITION | DATA_TYPE | COLUMN_KEY |
+-------------+------------------+-----------+------------+
| beers       |                3 | int       |            |
| name        |                2 | varchar   |            |
+-------------+------------------+-----------+------------+

MySQL > SET show_gipk_in_create_table_and_information_schema = 1;

MySQL > SELECT COLUMN_NAME, ORDINAL_POSITION, DATA_TYPE, COLUMN_KEY
        FROM INFORMATION_SCHEMA.COLUMNS
        WHERE TABLE_NAME = "rivierajug";
+-------------+------------------+-----------+------------+
| COLUMN_NAME | ORDINAL_POSITION | DATA_TYPE | COLUMN_KEY |
+-------------+------------------+-----------+------------+
| beers       |                3 | int       |            |
| my_row_id   |                1 | bigint    | PRI        |
| name        |                2 | varchar   |            |
+-------------+------------------+-----------+------------+
```

MySQL 8.0: SysBench IO Bound Read Only (Point Selects)

2x Faster than MySQL 5.7

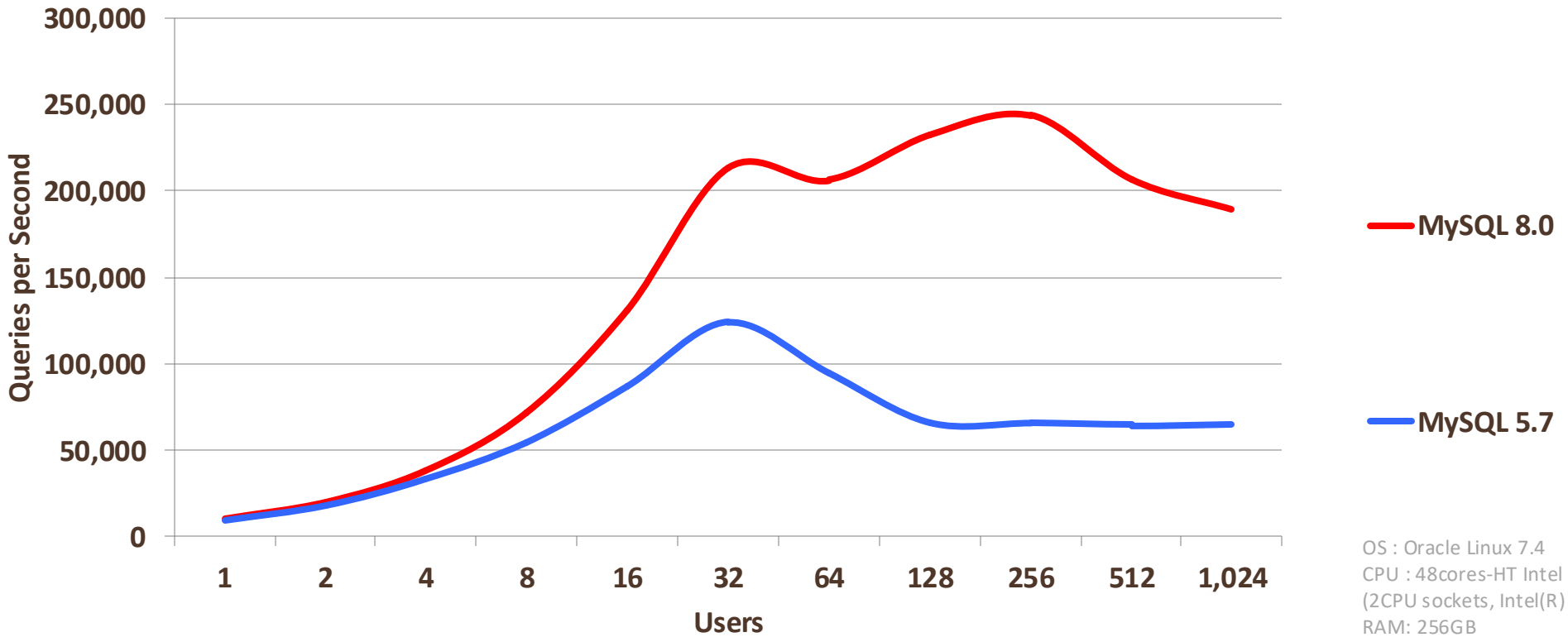


OS : Oracle Linux 7.4
CPU : 48cores-HT Intel Skylake 2.7Ghz
(2CPU sockets, Intel(R) Xeon(R) Platinum 8168 CPU)
RAM: 256GB
Storage : x2 Intel Optane flash devices
(Intel (R) Optane (TM) SSD P4800X Series)

http://dimitrik.free.fr/blog/posts/mysql-performance-80-and-sysbench-oltp_rw-updatenokey.html

MySQL 8.0: SysBench Read Write (update nokey)

2x Faster than MySQL 5.7

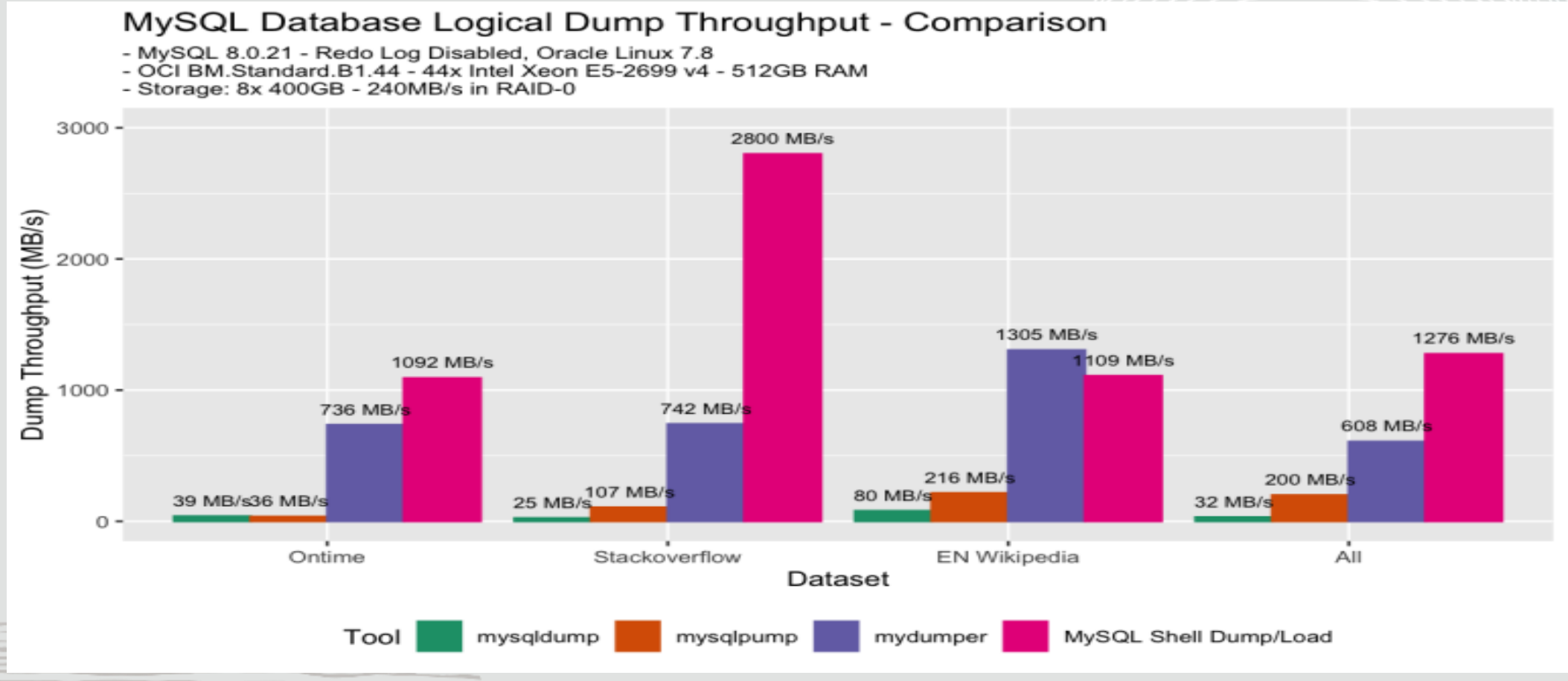


http://dimitrik.free.fr/blog/posts/mysql-performance-1m-iobound-qps-with-80-ga-on-intel-optane-ssd.html

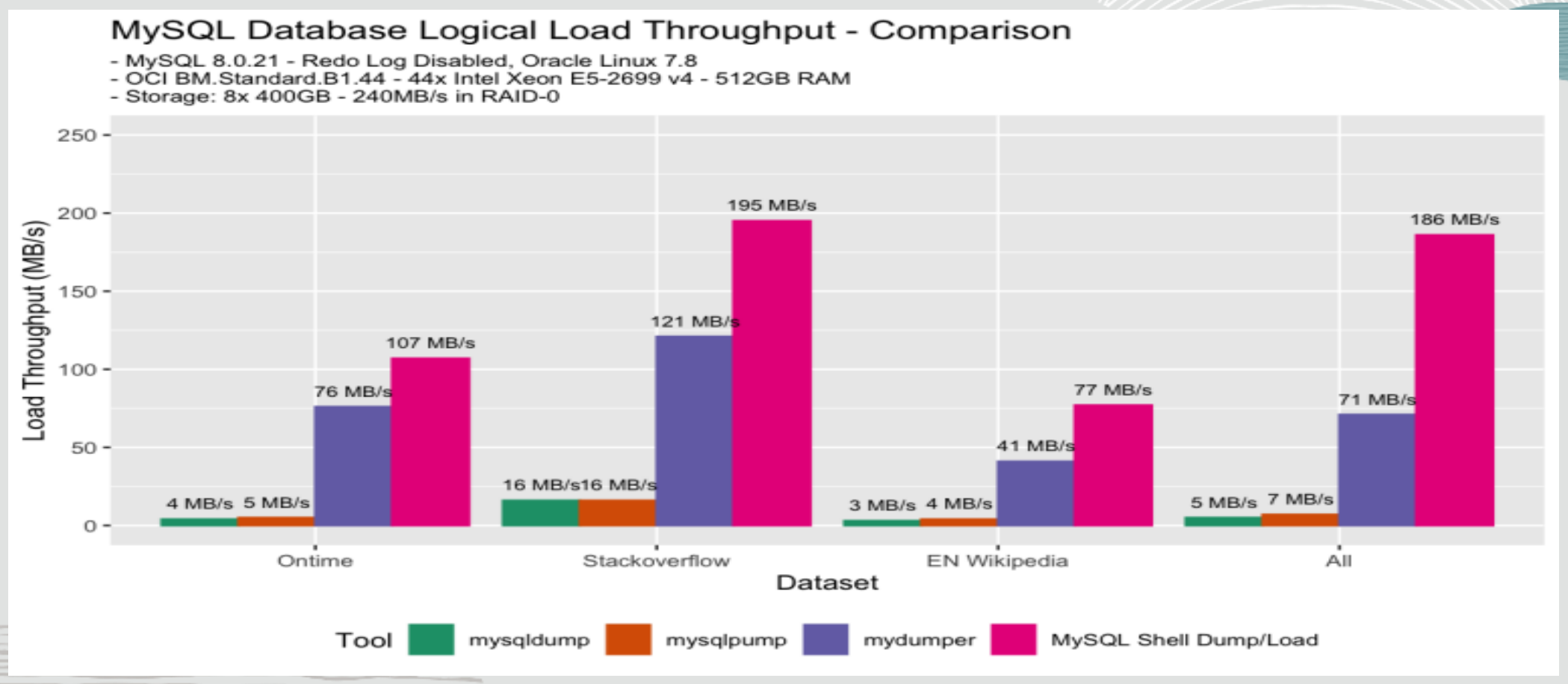
OS : Oracle Linux 7.4
CPU : 48cores-HT Intel Skylake 2.7Ghz
(2CPU sockets, Intel(R) Xeon(R) Platinum 8168 CPU)
RAM: 256GB
Storage : x2 Intel Optane flash devices
(Intel (R) Optane (TM) SSD P4800X Series)

MySQL Shell Dump & Load

MySQL Shell Changes – Dump Utility Benchmark



MySQL Shell Changes – Load Utility Benchmark



MySQL Shell Changes – Dump and Load Utilities Features

- Multi-threaded dump, splitting larger tables in smaller chunks, with speeds shown up to **3GB/S for dump** and **200 MB/S for load**!
- Concurrent loading capability while **Dump still in progress**
- Built-in **Compression** (zstd & gzip)
- Compatibility modes for OCI **MySQL Database Service (MDS)** makes migration to the cloud a piece of cake!

04 New Utilities for Logical Dump and Load

- `util.dumpInstance()`
  - Dump an entire database instance, including users
- `util.dumpSchemas()`
  - Dump a set of schemas
- `util.dumpTables()`
  - Dump a set of Tables

- `util.loadDump()`
  - Load a dump into a target database

MySQL Shell Changes – Instance Dump Example

```
 MySQL  localhost:33060+ ssl  JS > util.dumpInstance("/home/opc/backup/instance_backup.dmp",{threads:10})
Acquiring global read lock
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Writing users DDL
Writing DDL for schema `employees`
Writing DDL for view `employees`.`dept_emp_latest_date`
Writing DDL for view `employees`.`current_dept_emp`
Writing DDL for table `employees`.`dept_emp`
Writing DDL for table `employees`.`departments`
Writing DDL for table `employees`.`dept_manager`
Writing DDL for table `employees`.`salaries`
Writing DDL for table `employees`.`employees`
Preparing data dump for table `employees`.`departments`
Writing DDL for table `employees`.`titles`
Writing DDL for schema `test`
Data dump for table `employees`.`departments` will be chunked using column `dept_no`
Preparing data dump for table `employees`.`dept_emp`
Data dump for table `employees`.`dept_emp` will be chunked using column `emp_no`
Preparing data dump for table `employees`.`dept_manager`
Data dump for table `employees`.`dept_manager` will be chunked using column `emp_no`
Preparing data dump for table `employees`.`employees`
Data dump for table `employees`.`employees` will be chunked using column `emp_no`
Preparing data dump for table `employees`.`salaries`
Data dump for table `employees`.`salaries` will be chunked using column `emp_no`
Preparing data dump for table `employees`.`titles`
Data dump for table `employees`.`titles` will be chunked using column `emp_no`
Preparing data dump for table `test`.`trips`
Data dump for table `test`.`trips` will be chunked using column `id`
Preparing data dump for table `classicmodels`.`customers`
```

Dumping an entire instance with 10 parallel threads

MySQL Shell Changes – Instance Dump Example

```
Writing DDL for table `world`.`city`
Writing DDL for table `world`.`country`
Writing DDL for table `world`.`countrylanguage`
Data dump for table `employees`.`departments` will be written to 1 file
Data dump for table `employees`.`dept_manager` will be written to 1 file
Data dump for table `employees`.`dept_emp` will be written to 1 file
Data dump for table `employees`.`employees` will be written to 1 file
Data dump for table `employees`.`titles` will be written to 1 file
Data dump for table `classicmodels`.`customers` will be written to 1 file
Data dump for table `classicmodels`.`employees` will be written to 1 file
Data dump for table `classicmodels`.`offices` will be written to 1 file
Data dump for table `employees`.`salaries` will be written to 4 files
Data dump for table `classicmodels`.`orderdetails` will be written to 1 file
Data dump for table `classicmodels`.`orders` will be written to 1 file
Data dump for table `classicmodels`.`payments` will be written to 1 file
Data dump for table `classicmodels`.`productlines` will be written to 1 file
Data dump for table `world`.`city` will be written to 1 file
Data dump for table `classicmodels`.`products` will be written to 1 file
Data dump for table `world`.`country` will be written to 1 file
Data dump for table `world`.`countrylanguage` will be written to 1 file
Data dump for table `test`.`trips` will be written to 670 files
1 thds dumping – 98% (102.57M rows / ~103.75M rows), 1.89M rows/s, 148.30 MB/s uncompressed, 30.16 MB/s compressed
Duration: 00:00:53s
Schemas dumped: 4
Tables dumped: 18
Uncompressed data size: 7.86 GB
Compressed data size: 1.61 GB
Compression ratio: 4.9
Rows written: 102570941
Bytes written: 1.61 GB
Average uncompressed throughput: 146.34 MB/s
Average compressed throughput: 29.93 MB/s
```

Statistics for an instance of size 7.8 GB

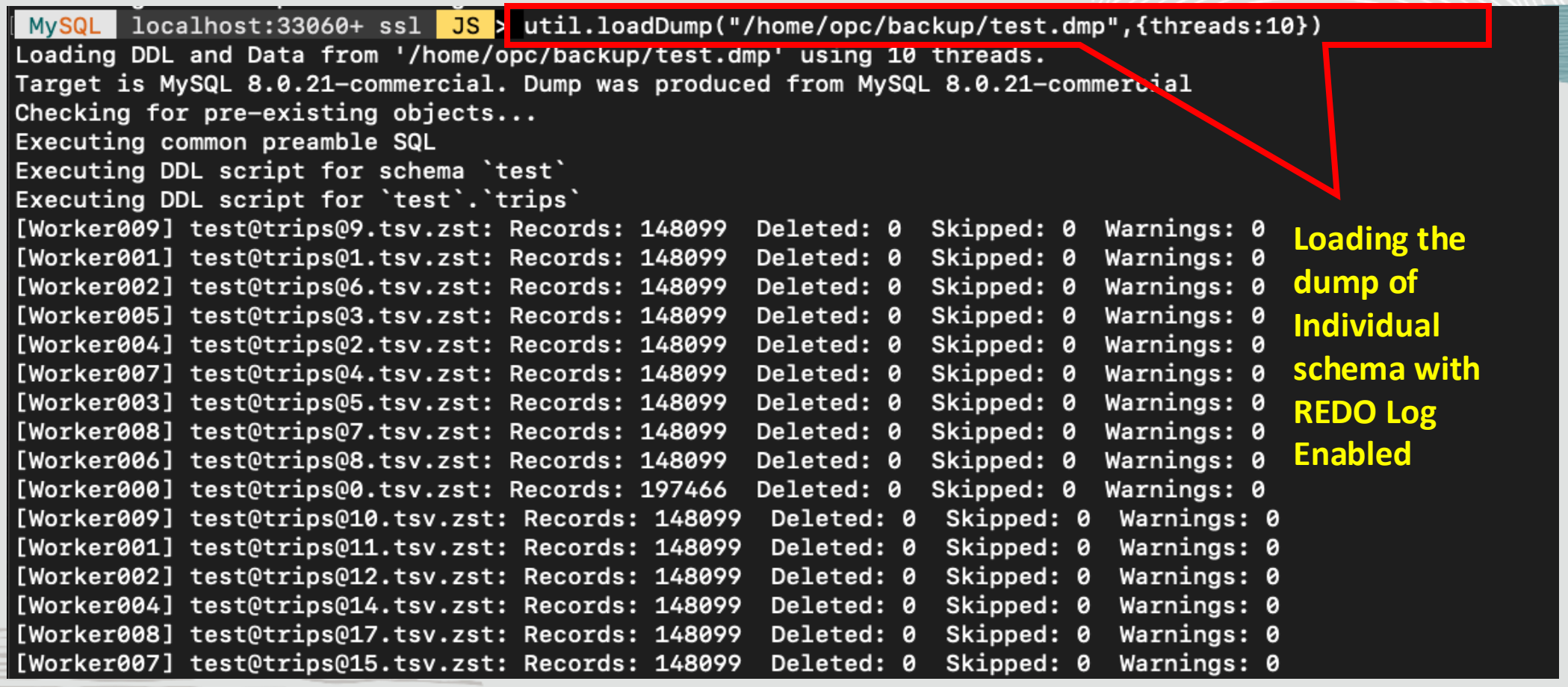
MySQL Shell Changes – Schema Dump Example

```
 MySQL  localhost:33060+ ssl  JS > util.dumpSchemas(["test"],"/home/opc/backup/test.dmp",{threads:10})
Acquiring global read lock
All transactions have been started
Locking instance for backup
Global read lock has been released
Writing global DDL files
Preparing data dump for table `test`.`trips`
Writing DDL for schema `test`
Writing DDL for table `test`.`trips`
Data dump for table `test`.`trips` will be chunked using column `id`
Running data dump using 10 threads.
NOTE: Progress information uses estimated values and may not be accurate.
Data dump for table `test`.`trips` will be written to 670 files
1 thds dumping - 98% (98.64M rows / ~99.83M rows), 1.90M rows/s, 149.11 MB/s uncompressed, 30.32 MB/s compressed
Duration: 00:00:51s
Schemas dumped: 1
Tables dumped: 1
Uncompressed data size: 7.71 GB
Compressed data size: 1.57 GB
Compression ratio: 4.9
Rows written: 98642760
Bytes written: 1.57 GB
Average uncompressed throughput: 148.92 MB/s
Average compressed throughput: 30.30 MB/s
 MySQL  localhost:33060+ ssl  JS >
```

Dumping an individual schema with 10 parallel threads

Statistics for an individual instance

MySQL Shell Changes - Load Dump



MySQL Document Store

SQL + NoSQL = MySQL

Document Oriented Databases

What is a Document?

A data structure that can represent complex information, similar to an Object

Structure of the data is part of the document, no uniform structure

JSON (=JavaScript Object Notation)

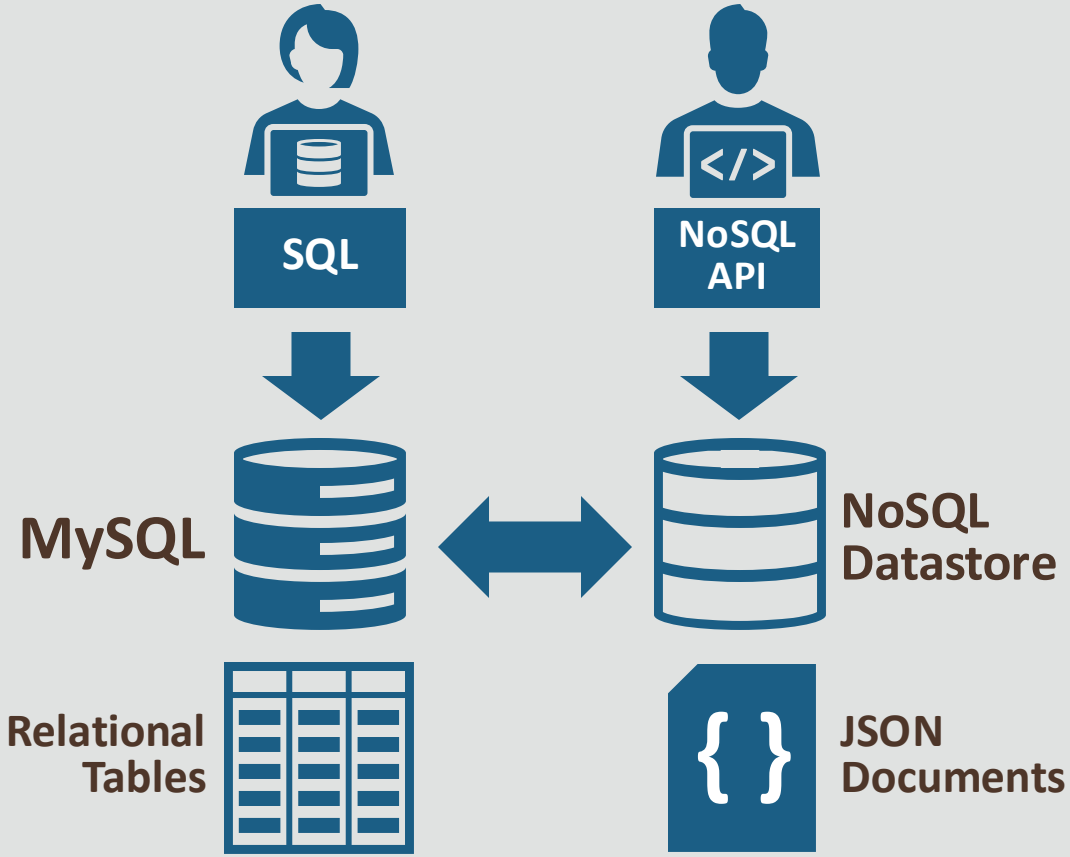
- Compact, popular and standardized
- Can be represented natively in many languages (JavaScript, Python etc)

Other popular encoding formats are XML, YAML etc

JSON Document Example

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

Pains running RDBMS plus NoSQL Datastore

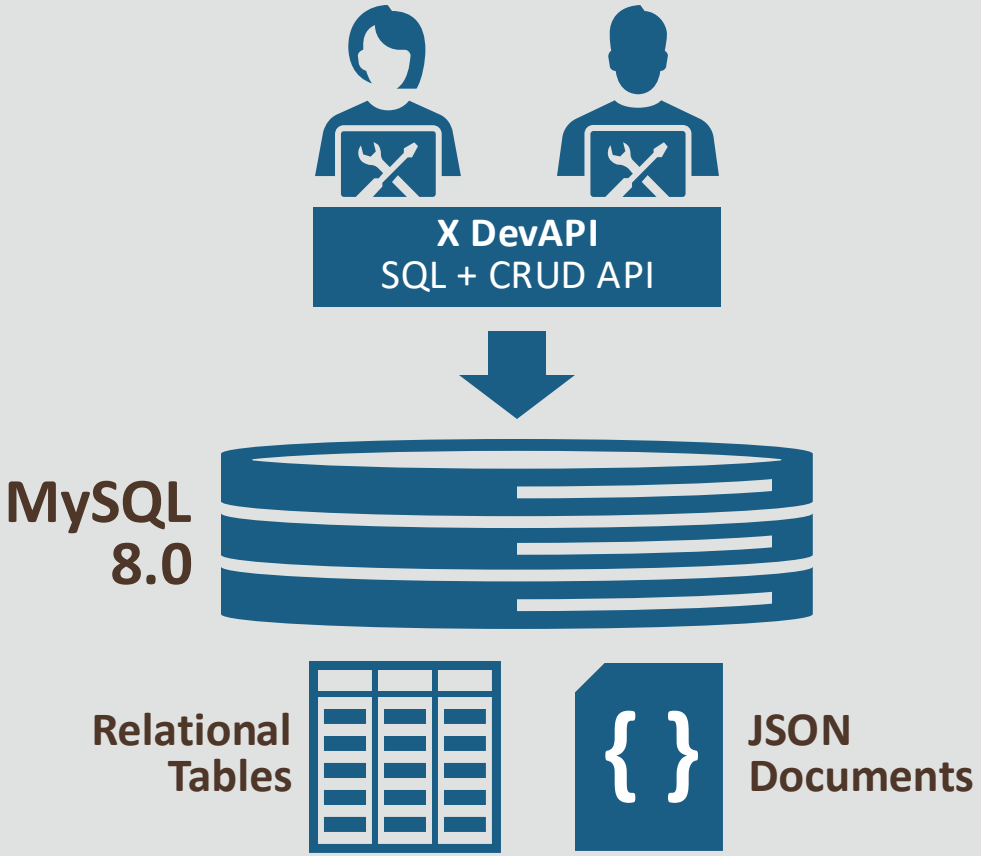


Developers
Required to learn multiple APIs

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

Operations
Required to manage multiple products with different tools

MySQL Document Store: SQL + NoSQL = MySQL

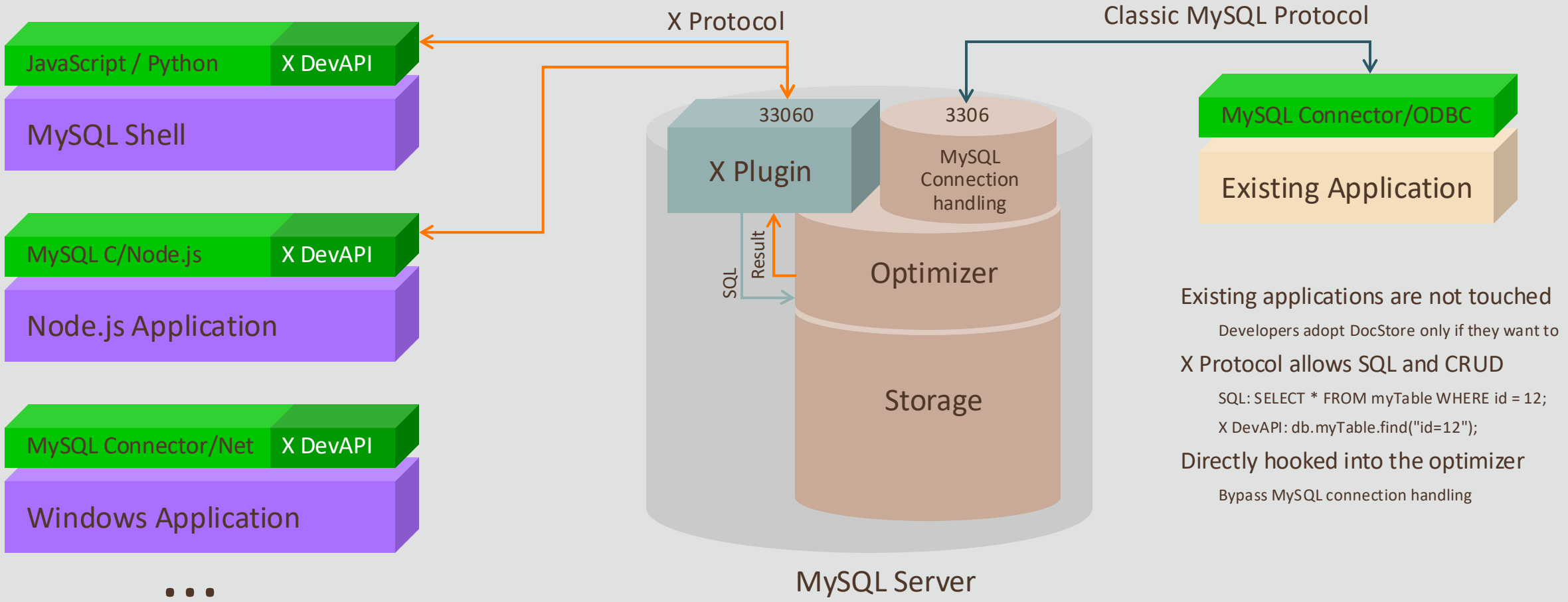


Flexibility for Developers
Unified API provides more flexibility

Reliable data management
Data is always in sync and you can “JOIN” tables and documents

Simplified Operations
Managing single database with unified management tool

MySQL Document Store – How it works



Existing applications are not touched

- Developers adopt DocStore only if they want to

X Protocol allows SQL and CRUD

- SQL: SELECT * FROM myTable WHERE id = 12;
- X DevAPI: db.myTable.find("id=12");

Directly hooked into the optimizer

- Bypass MySQL connection handling

How to use MySQL as NoSQL ?

Schema

- mysqlsh
- shell.connect("root@10.0.10.13")
- CREATE SCHEMA DocumentDB
- \use DocumentDB

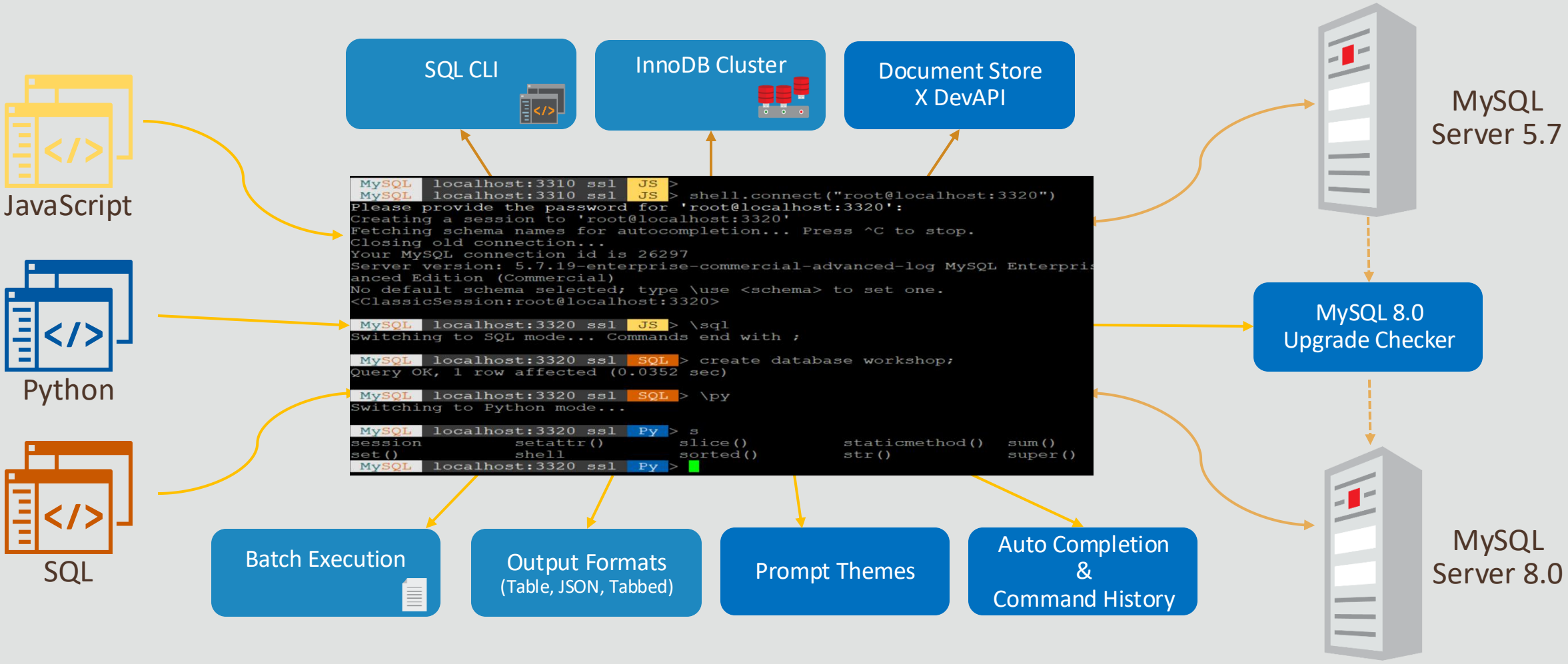
Collection

- db.createCollection("employee")
- db.getCollections()
- db.dropCollection("dept")

Document

- db.employee.add({"empname":"chandan","salary":1000,"dept":"HR"})
- db.employee.find()
- db.employee.find("empname='chandan'")
- db.employee.find(" empname='chandan' and dept='sales' ")
- db.employee.modify("empname='chandan'").set("empname","Radha")
- db.employee.remove("empname='Radha'")

MySQL Shell 8.0

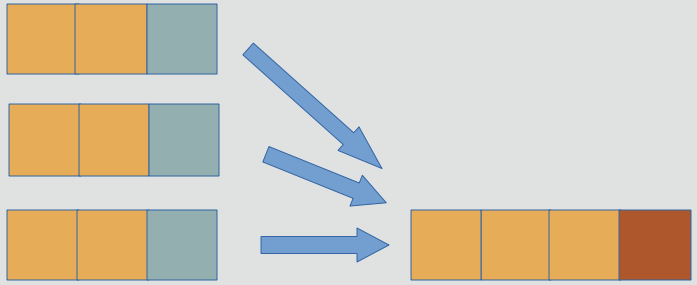


MySQL 8 - Window Functions and CTEs(Common Table Expression)

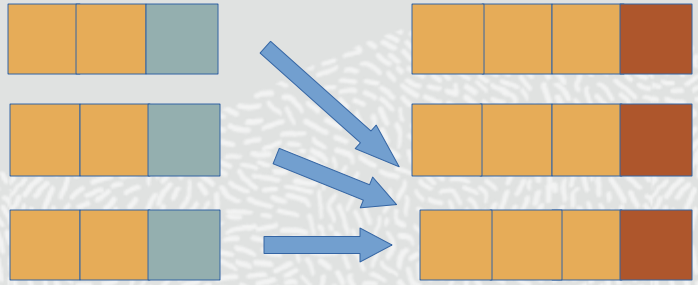
A window function performs a calculation across a set of rows that are related to the current row, similar to an aggregate function.

But unlike aggregate functions, a window function does not cause rows to become grouped into a single output row.

Window functions can access values of other rows “in the vicinity” of the current row.



Aggregate function



Window function

MySQL 8.0: CTEs & Window Functions

Before MySQL 8.0

```
INSERT INTO `tbl_ServiceRequestMaster_BeforeMySQL8`
      (
      ProblemId          ,Title              ,TechnicianName     ,
      CategoryId         ,OldState           ,CurrentState       ,
      LastUpdateTime     ,LocationId
      )

 SELECT DISTINCT
      A.RequestId        ,A.Title            ,A.ActionPerformedBy        ,
      A.CategoryId       ,A.OldStatename     ,A.NewStatename             ,
      A.LastUpdateTime   ,A.LocationId
 FROM  tbl_ServiceRequestHistory A ,(SELECT RequestId,CategoryId,MAX(LastUpdateTime) AS CurrentTime
                                     FROM tbl_ServiceRequestHistory
                                     GROUP BY RequestId,CategoryId
                                    )B
 WHERE A.RequestId=B.RequestId
   AND A.LastUpdateTime=B.CurrentTime
   AND A.CategoryId=B.CategoryId;
```

In MySQL 8.0

```
INSERT INTO `tbl_ServiceRequestMaster_AfterMySQL8`
      (
      ProblemId          ,Title              ,TechnicianName     ,
      CategoryId         ,OldState           ,CurrentState       ,
      LastUpdateTime     ,LocationId
      )
SELECT
      A.RequestId        ,A.Title            ,A.ActionPerformedBy        ,
      A.CategoryId       ,A.OldStatename     ,A.NewStatename             ,
      A.LastUpdateTime   ,A.LocationId

 FROM
  (
     SELECT *,
            ROW_NUMBER() OVER (PARTITION BY RequestId,CategoryId ORDER BY LastUpdateTime DESC)as Rownum
       FROM tbl_ServiceRequestHistory
  ) A
WHERE A.Rownum=1;
```

With Windowing you don't need to compute aggregation statistics separately and then join /correlate back to underlying data, hence performance of query improved

MySQL 8.0 – LATERAL DERIVED TABLES

- **LATERAL** keywords allows the sub-SELECT to refer to columns of FROM items that appear before it FROM clause.
- Without **LATERAL** ,each sub-select is evaluated independently and so can't cross reference any other FROM item.
- As of 8.0.14 version , a derived table can refer to(depend on) columns of preceding tables in same FROM clause.
- Part of SQL:1999 standard

```
SELECT ActorName
 FROM actor A , (  SELECT actor_id
                   FROM film_actor
                   WHERE actor_id=A.actor_id
                 )B;
```

A is outside table

Error Code: 1054. Unknown column 'A.actor_id' in 'where clause

```
SELECT A.ActorName ,B.film_id
 FROM actor A ,LATERAL (SELECT actor_id ,film_id
                        FROM film_actor
                        WHERE actor_id=A.actor_id
                       )B;
```

Glimpse of Lateral Derived Table

LATERAL keywords allows the sub-SELECT to refer to columns of FROM items that appear before it FROM clause.

Without **LATERAL** ,each sub-select is evaluated independently and so can't cross reference any other FROM item.

As of 8.0.14 version , a derived table can refer to(depend on) columns of preceding tables in same FROM clause.

Part of SQL:1999 standard

```
SELECT ActorName
 FROM actor A , (  SELECT actor_id
                   FROM film_actor
                   WHERE actor_id=A.actor_id
                 )B;
```

A is outside table

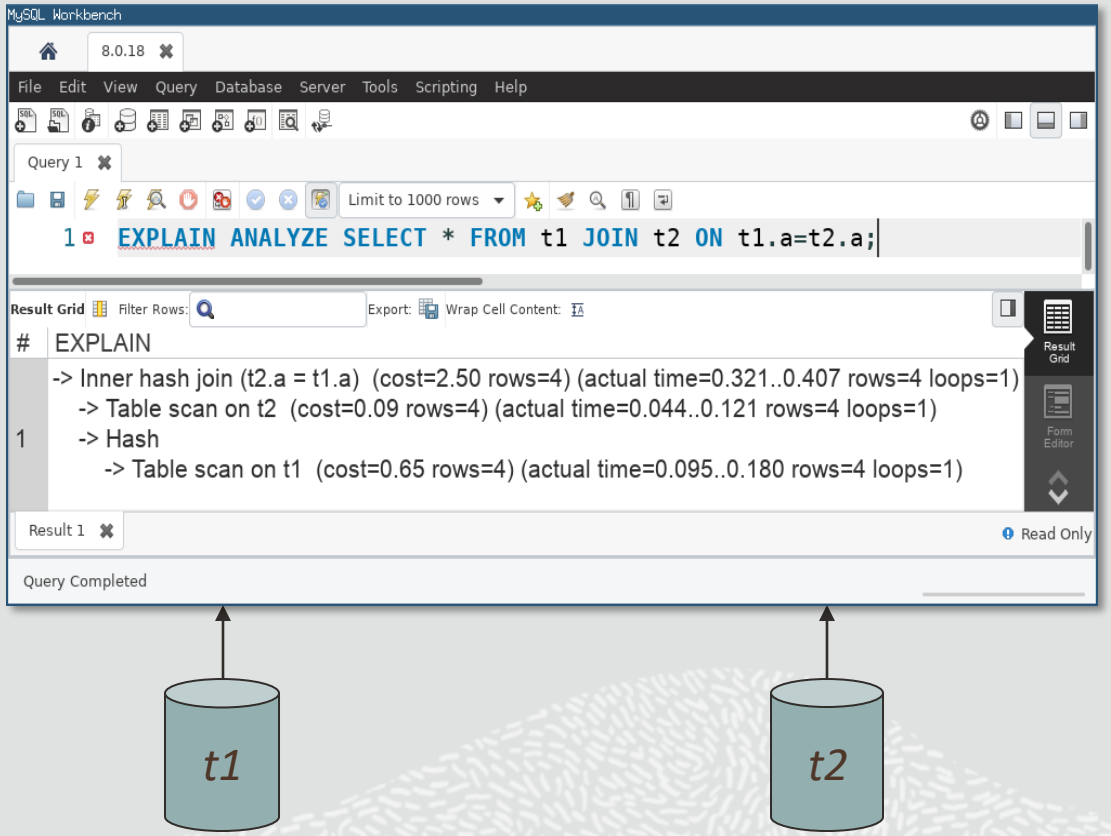
Error Code: 1054. Unknown column 'A.actor_id' in 'where clause

```
SELECT A.ActorName ,B.film_id
 FROM actor A ,LATERAL (SELECT actor_id ,film_id
                        FROM film_actor
                        WHERE actor_id=A.actor_id
                       )B;
```

MySQL 8 - EXPLAIN ANALYZE

- **Measurements**
  - Time (in ms) to first row
  - Time (in ms) to last row
  - Number of rows
  - Number of loops
- **Execute the query and dump the stats**
- **Built on EXPLAIN FORMAT=TREE**

EXPLAIN ANALYZE SELECT * FROM *t1* JOIN *t2* ON *t1.a* = *t2.a*;



Explain Format=Tree and Explain Analyze

An **EXPLAIN FORMAT=TREE** will show us the query plan and cost estimates:.
But it doesn't tell us if those estimates are correct, or on which operations in the query plan the time is actually spent.

```
EXPLAIN FORMAT=TREE
SELECT first_name, last_name, SUM(amount) AS total
FROM staff INNER JOIN payment
  ON staff.staff_id = payment.staff_id
    AND
    payment_date LIKE '2005-08%'
GROUP BY first_name, last_name;
```

```
-> Table scan on <temporary>
   -> Aggregate using temporary table
      -> Nested loop inner join  (cost=1757.30 rows=1787)
         -> Table scan on staff  (cost=3.20 rows=2)
         -> Filter: (payment.payment_date like '2005-08%')  (cost=117.43
rows=894)
            -> Index lookup on payment using idx_fk_staff_id
(staff_id=staff.staff_id)  (cost=117.43 rows=8043)
```

ANALYZE provide several new measurements here:

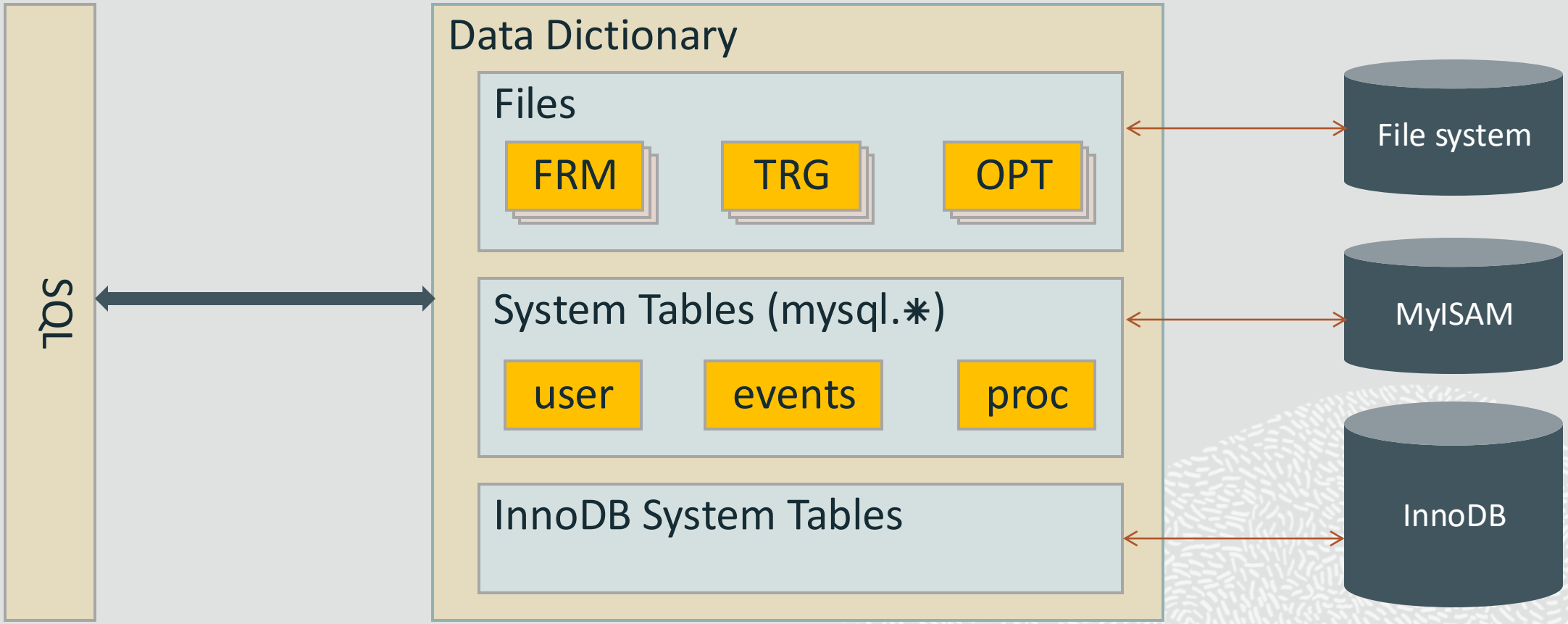
- Actual time to get first row (in milliseconds)
- Actual time to get all rows (in milliseconds)
- Actual number of rows read
- Actual number of loops

```
EXPLAIN ANALYZE
SELECT first_name, last_name, SUM(amount) AS total
FROM staff INNER JOIN payment
  ON staff.staff_id = payment.staff_id
    AND
    payment_date LIKE '2005-08%'
GROUP BY first_name, last_name;
```

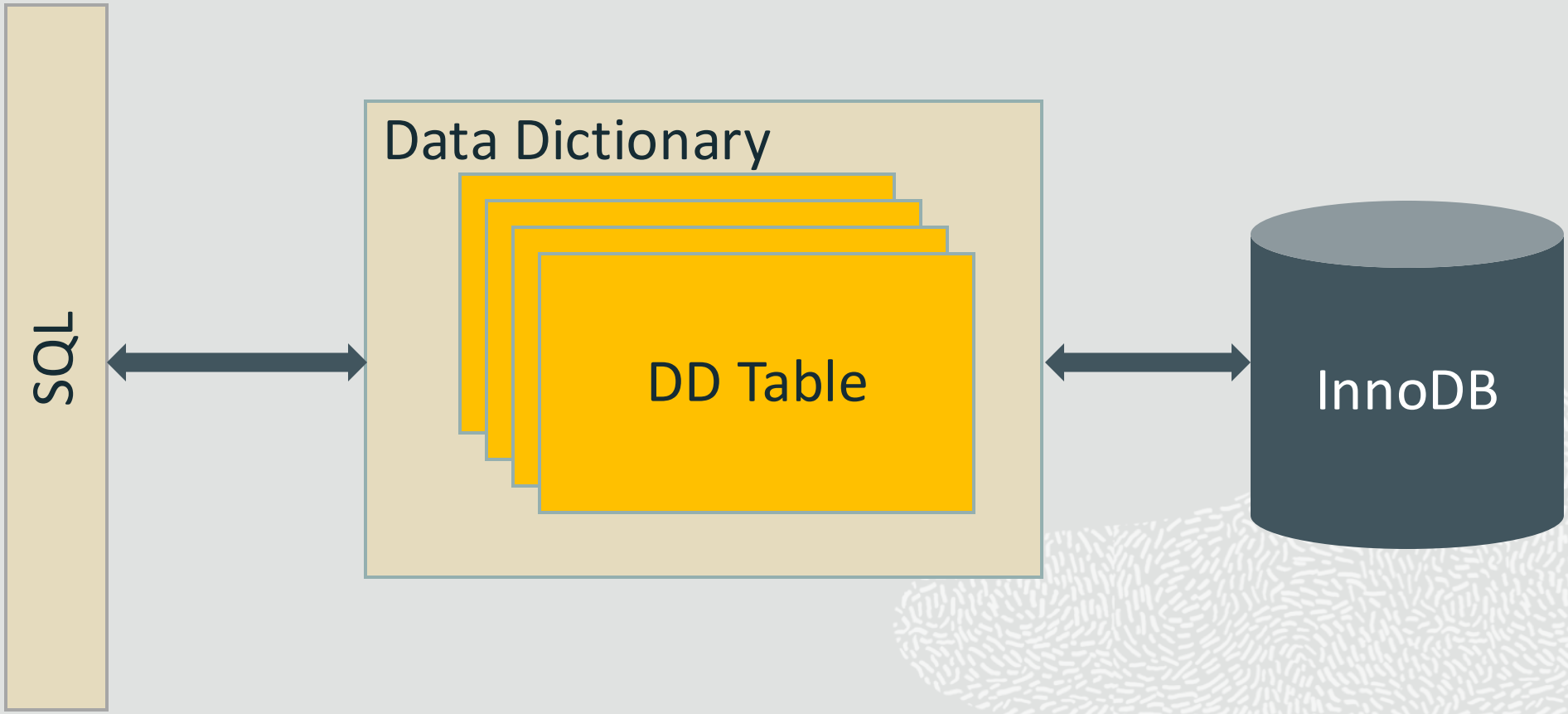
```
-> Table scan on <temporary>  (actual time=0.001..0.001 rows=2 loops=1)
   -> Aggregate using temporary table  (actual time=58.104..58.104 rows=2 loops=1)
      -> Nested loop inner join  (cost=1757.30 rows=1787) (actual time=0.816..46.135
rows=5687 loops=1)
         -> Table scan on staff  (cost=3.20 rows=2) (actual time=0.047..0.051 rows=2 loops=1)
         -> Filter: (payment.payment_date like '2005-08%')  (cost=117.43 rows=894) (actual
time=0.464..22.767 rows=2844 loops=2)
            -> Index lookup on payment using idx_fk_staff_id (staff_id=staff.staff_id)  (cost=117.43
rows=8043) (actual time=0.450..19.988 rows=8024 loops=2)
```

Data Dictionary

MySQL Data Dictionary before MySQL 8.0



Transactional Data Dictionary in MySQL 8.0



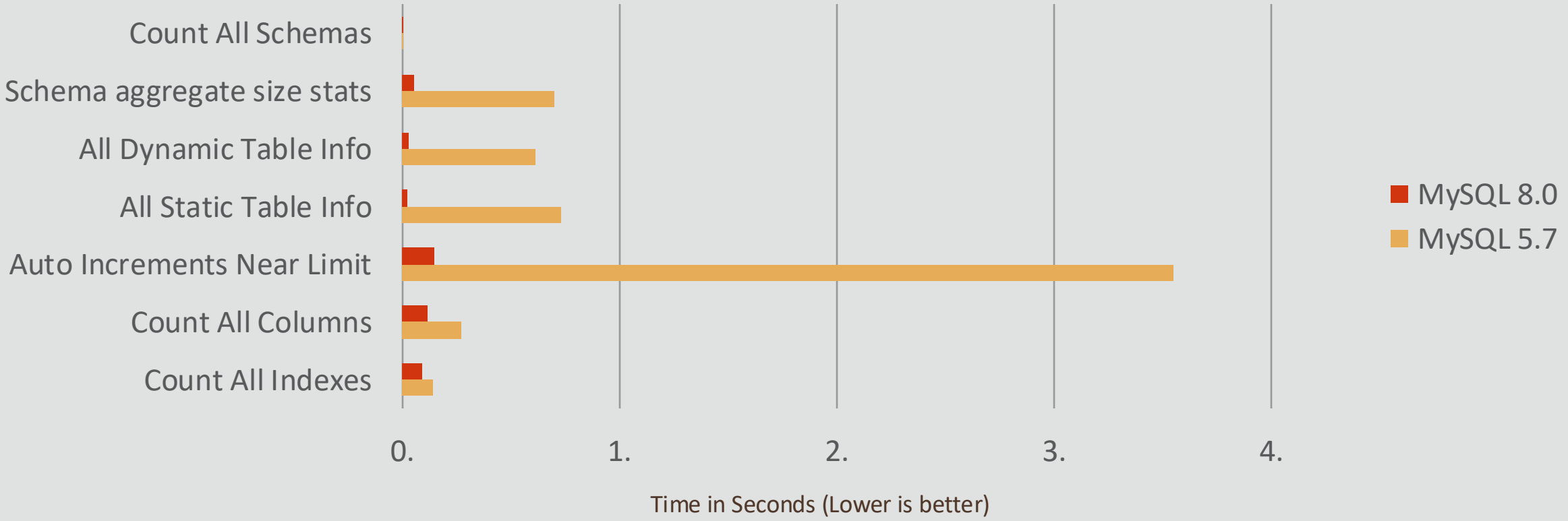
Performance Comparison

Over 30x faster!

MySQL 8.0: INFORMATION_SCHEMA

100 schemas times 50 tables (5000 tables)

30X Faster



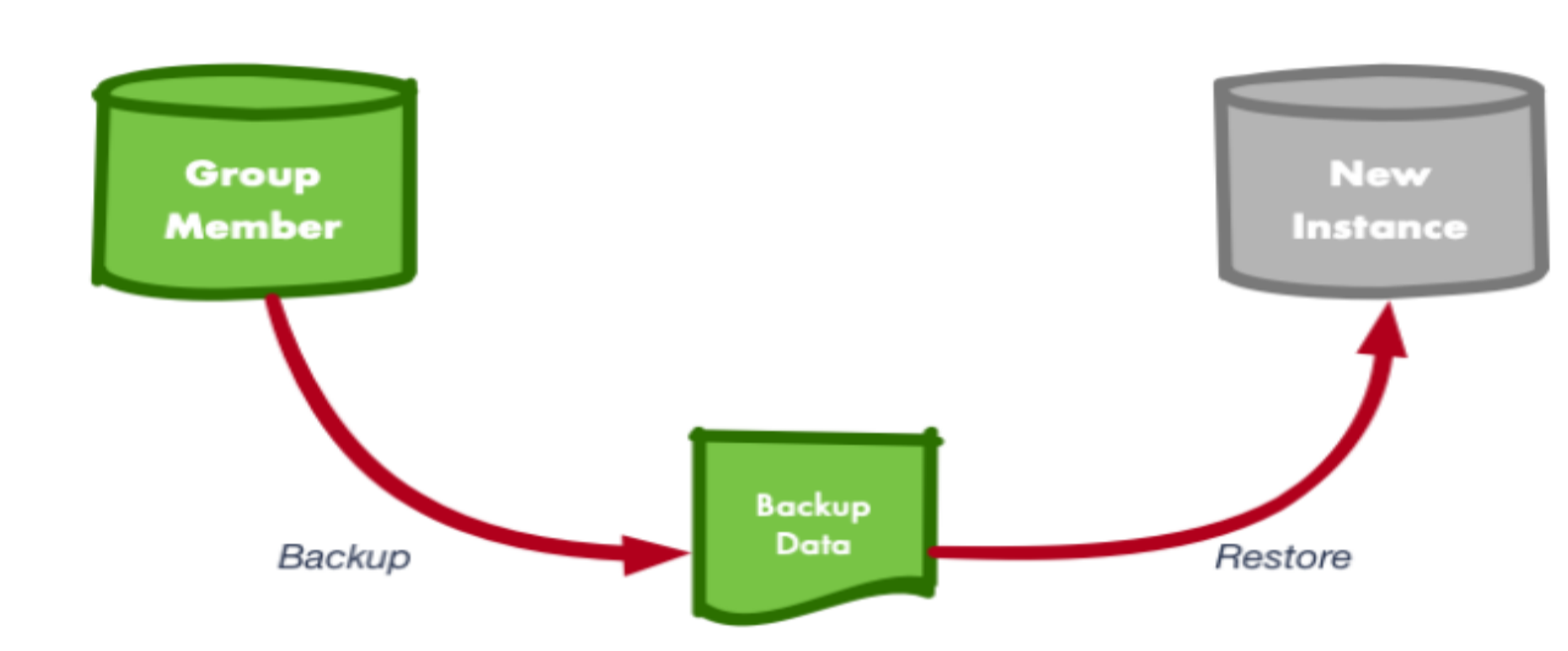
MySQL CLONE

Fast instance provisioning

Clone: Use Cases

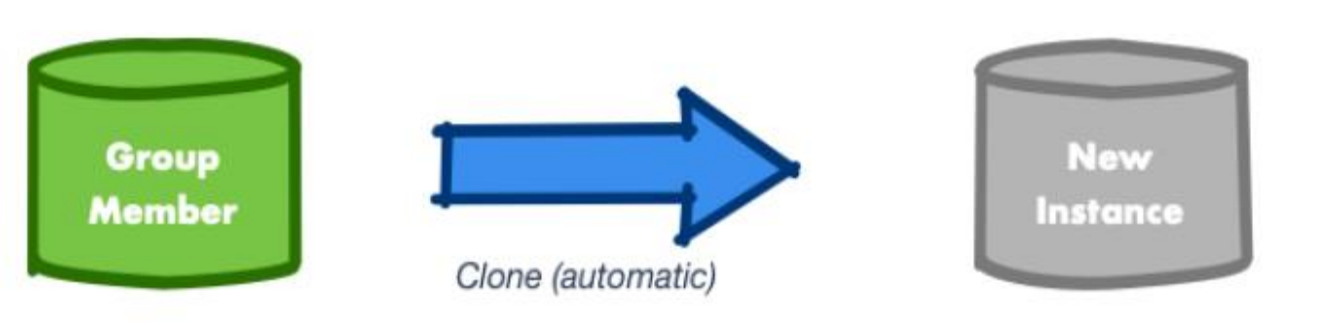
Provision a new replication node

Manual Process to build another DB Instance



Node Provisioning

Automatic Cloning



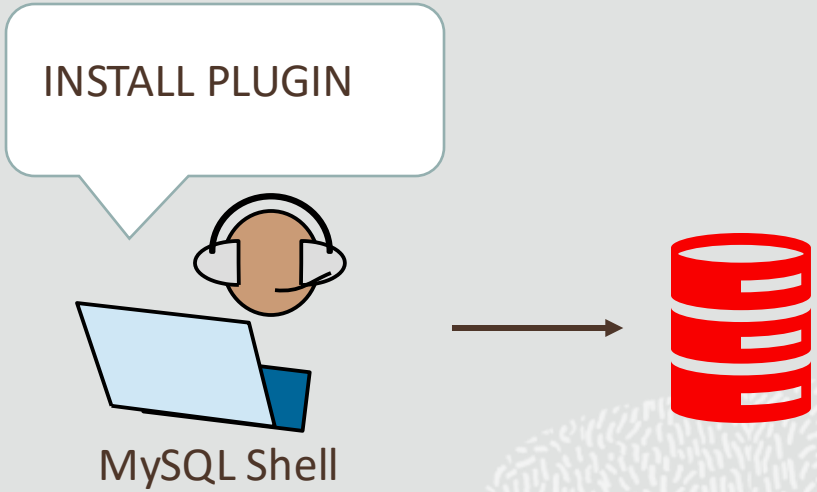
-Provision members that have no data.
-Takes Physical snapshot of data.
-Using standard MySQL connection (3306) .
-Built-in the MySQL Server.
-Fully integrated in MySQL InnoDB Cluster.

Clone Limitations

- Does not clone configuration values
- Does not clone binary logs
- Supports only InnoDB
- Blocks all concurrent DDLs on the Donor
- Will **FULLY** override recipient data
- Cloning can be done only to the **SAME** server version
- Doesn’t work over the X protocol

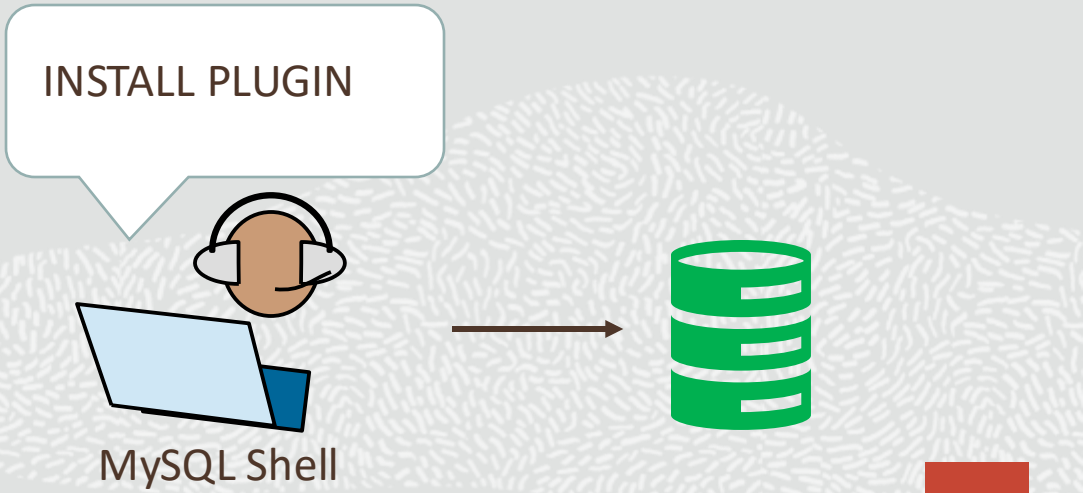
MySQL – 8 CLONE Setup the DONOR

```
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";

mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";

mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```



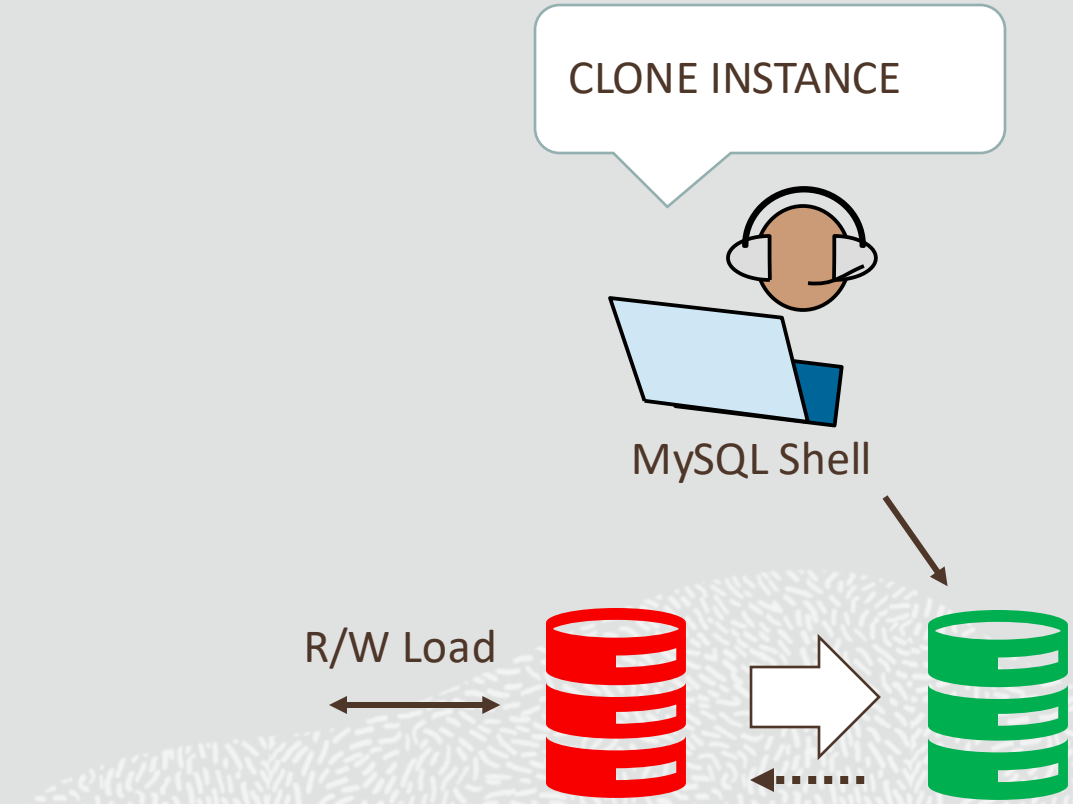
MySQL 8 - CLONE Setup the RECIPIENT

```
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";

mysql> SET GLOBAL clone_valid_donor_list = "donor.host.com:3306";

mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";

mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```



MySQL 8 - Connect to RECIPIENT and execute CLONE SQL statement

```
mysql> CLONE INSTANCE
    -> FROM clone_user@donor.host.com:3306
    -> IDENTIFIED BY "clone_password";
```



MySQL 8 - CLONE Check Status

```
mysql> select STATE, ...
    > from performance_schema.clone_status;
```

```
+-------------+----------------------+-------------+
| STATE       | START TIME           | DURATION    |
+-------------+----------------------+-------------+
| In Progress | 2019-07-17 17:23:26 |     4.84 m |
+-------------+----------------------+-------------+
```

MySQL 8 - CLONE Check Progress

```
mysql> select STATE, ...
    > from performance_schema.clone_progress;
```

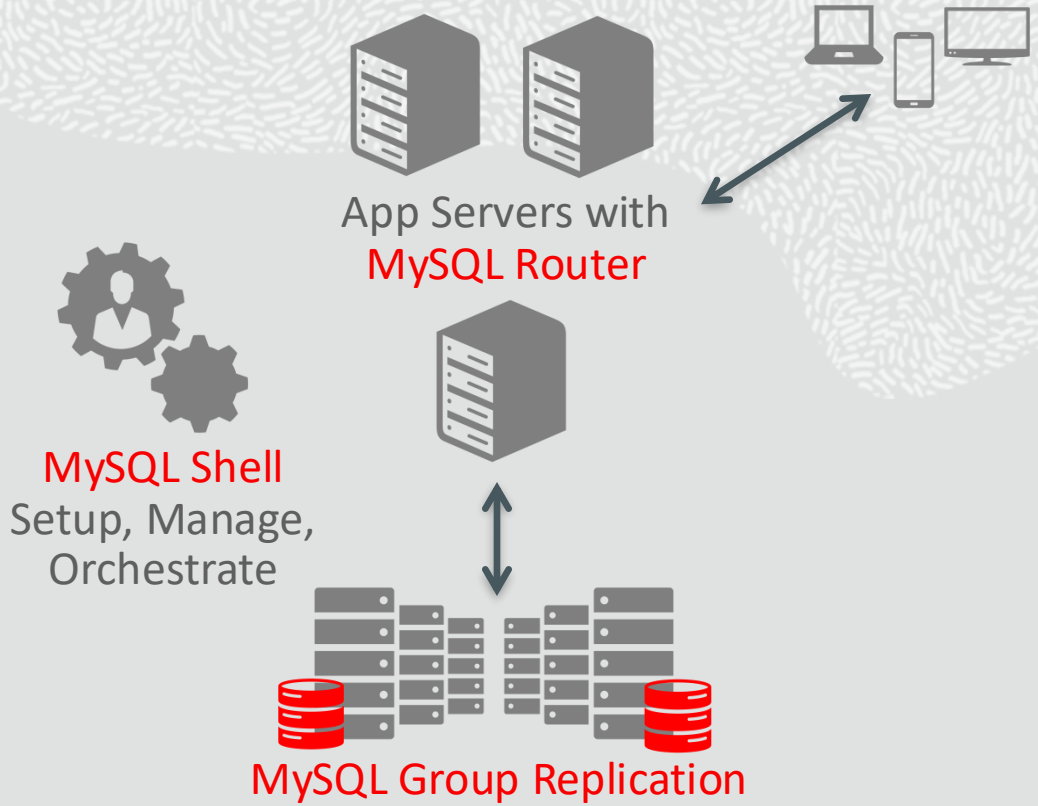
```
+-----------+-------------+------------+----------+-------------+----------+
| STAGE     | STATE       | START TIME | DURATION | Estimate    | Done(%)  |
+-----------+-------------+------------+----------+-------------+----------+
| DROP DATA |   Completed |   17:23:26 | 790.86 ms |       0 MB |     100% |
| FILE COPY |   Completed |   17:23:27 |  10.33 m  | 94,729 MB |     100% |
| PAGE COPY |   Completed |   17:33:47 |  15.91 s  | 11,885 MB |     100% |
| REDO COPY |   Completed |   17:34:03 |   1.07 s  |    293 MB |     100% |
| FILE SYNC | In Progress |   17:34:04 |  51.68 s  |       0 MB |       0% |
| RESTART   | Not Started |       NULL |      NULL |       0 MB |       0% |
| RECOVERY  | Not Started |       NULL |      NULL |       0 MB |       0% |
+-----------+-------------+------------+----------+-------------+----------+
```

MySQL InnoDB Cluster

High Availability - Out of the Box

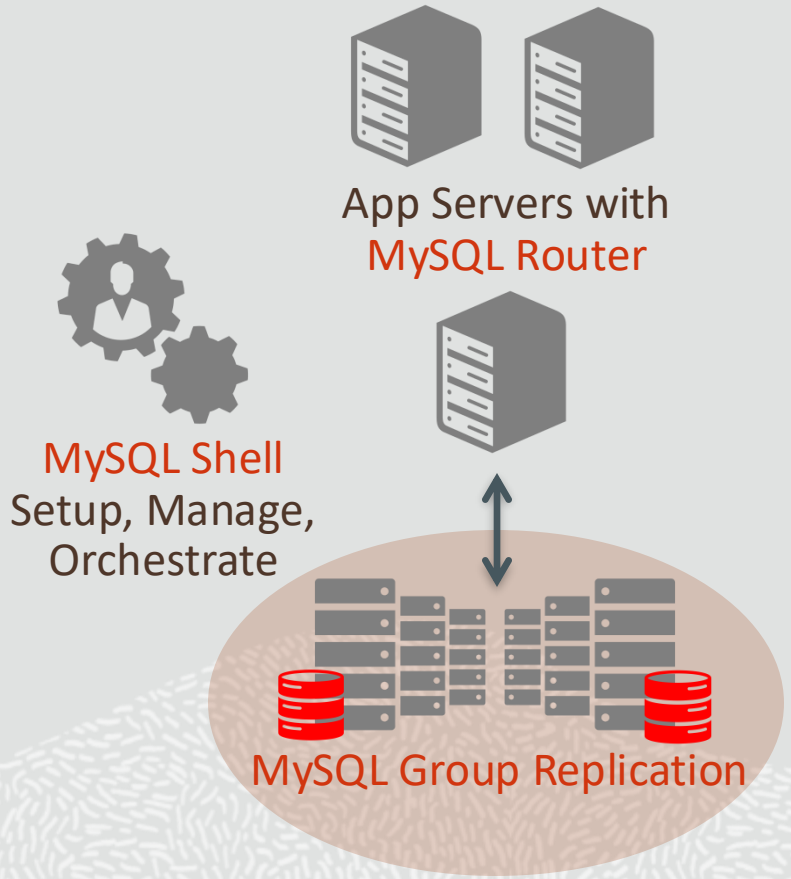
MySQL 8 - MySQL InnoDB Cluster

- MySQL Group Replication
  - High Availability
  - Elastic, Fault Tolerant, Self Healing
- MySQL Router
  - Connection Routing, Load Balancing
- MySQL Shell
  - Easy Setup & Administration



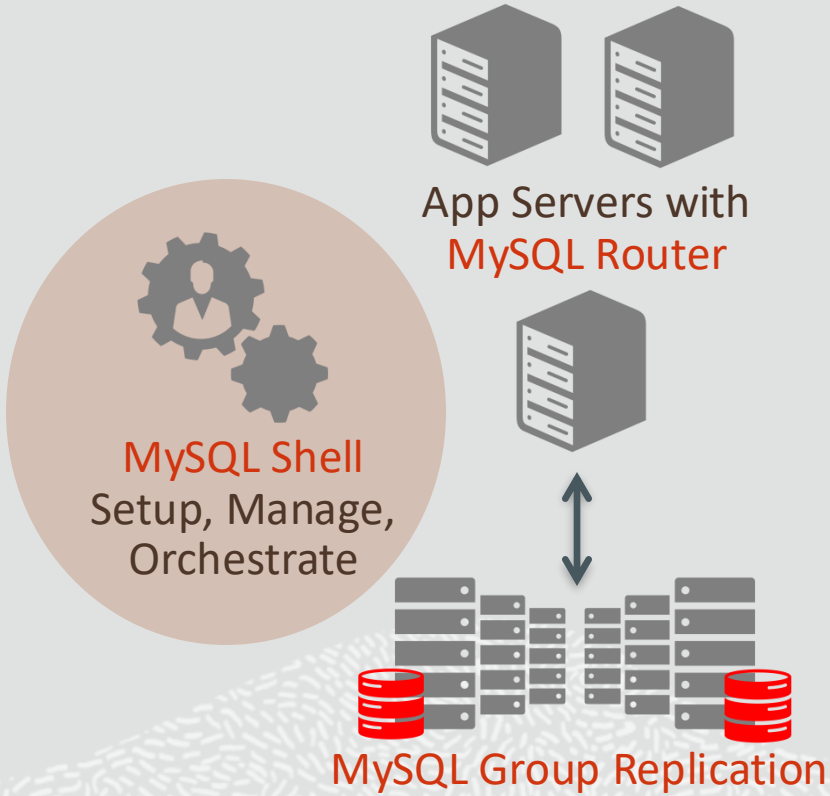
MySQL Group Replication: Database HA

- **"Single/Multi-master update anywhere replication plugin for MySQL"**
- **Part of MySQL InnoDB Cluster solutions**
  - configurable to overcome actual InnoDB Cluster limits
- **Provides virtually synchronous replication for MySQL 5.7.17+, 8.0.x**
  - Based on Paxos protocol
- **Automates operations**
  - Conflict detection and resolution
  - Failure detection, fail-over, recovery
  - Group membership management and reconfiguration
- **Supported on all MySQL platforms**
  - Linux, Windows, Solaris, OSX, FreeBSD



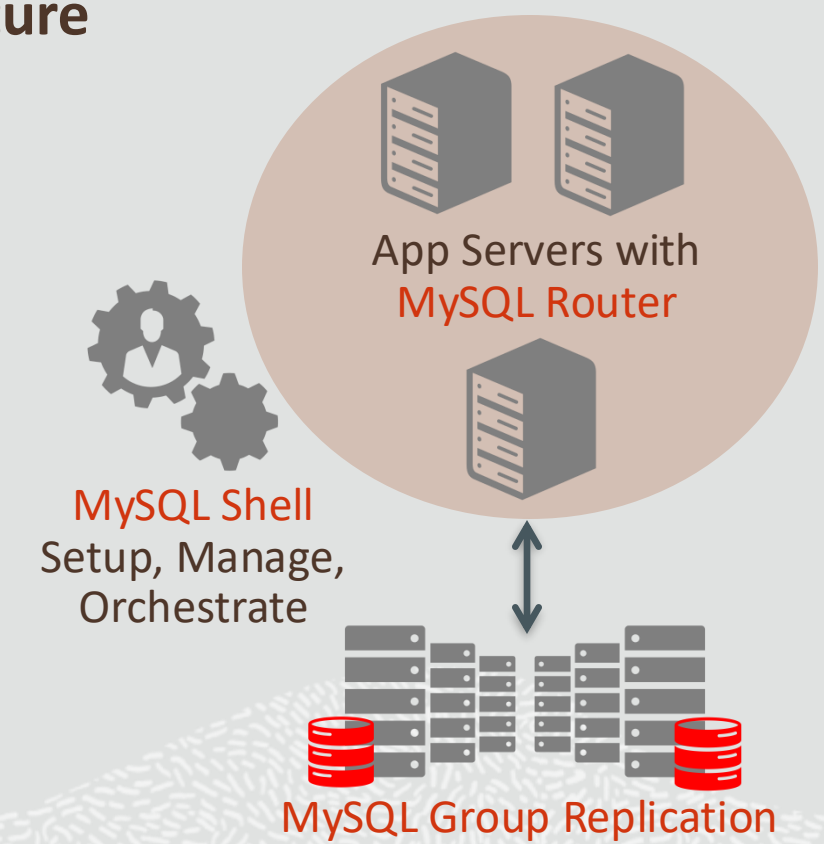
MySQL Shell – The DBA Interface

- **Multi-Language: JavaScript, Python, and SQL**
  - Naturally scriptable
- **Supports Document and Relational models**
- **Exposes full Development and Administration APIs**
  - InnoDB Cluster support
  - MySQL 8.0 clone plugin support for InnoDB Cluster
- **Classic MySQL protocol and X protocol**
- **Custom reports**



MySQL Router – Transparent Access to Database Architecture

- **Transparent client connection routing**
  - Load Balancing
  - Application connection failover
  - Little to no configuration needed
- **Stateless design offers easy HA client routing**
  - Router as part of the application stack
- **Integration into InnoDB Cluster & InnoDB ReplicaSet**
  - Understands Group Replication & Replication



MySQL InnoDB Cluster : Where to use?

Consistency is critical (RPO=0)

If primary fails

Split brain prevention

Automatic failover

New Primary automatically elected

Automatic Network Partition handling

Read Scale Out

- Fluid replication infrastructure
- Replication Lag handling
- Configurable consistency levels

Write Anywhere

Write to many nodes at same time

Orders WRITES within a group (XCOM)

Consistency guaranteed

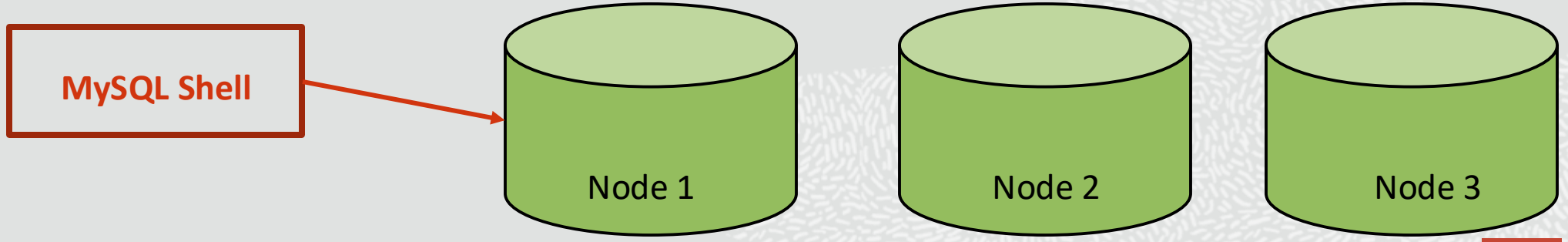
MySQL Shell : Setup InnoDB Cluster easily

- Connect to one server and create cluster with this member

```
MySQL  JS > \connect admin@10.0.0.75
MySQL  10.0.0.75:33060+ ssl  JS > Cluster = dba.createCluster("testcluster")
```

- Add the other two members to the cluster

```
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.76:3306")
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.77:3306")
```



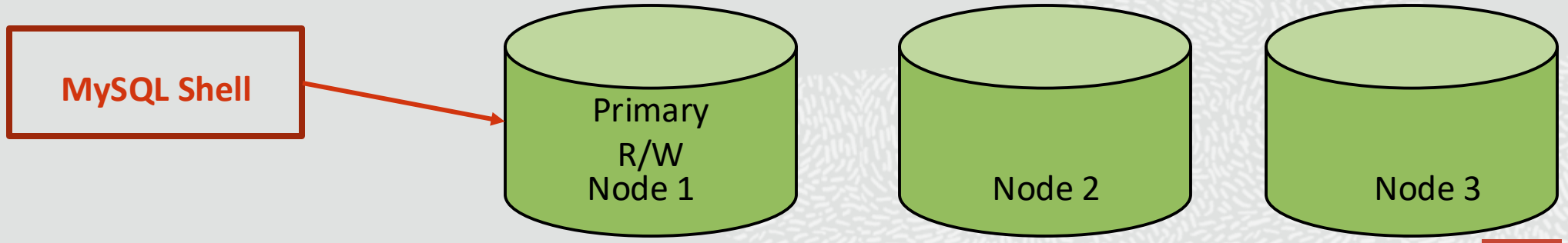
MySQL Shell : Setup InnoDB Cluster easily

- Connect to one server and create cluster with this member

```
MySQL  JS > \connect admin@10.0.0.75
MySQL  10.0.0.75:33060+ ssl  JS > Cluster = dba.createCluster("testcluster")
```

- Add the other two members to the cluster

```
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.76:3306")
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.77:3306")
```



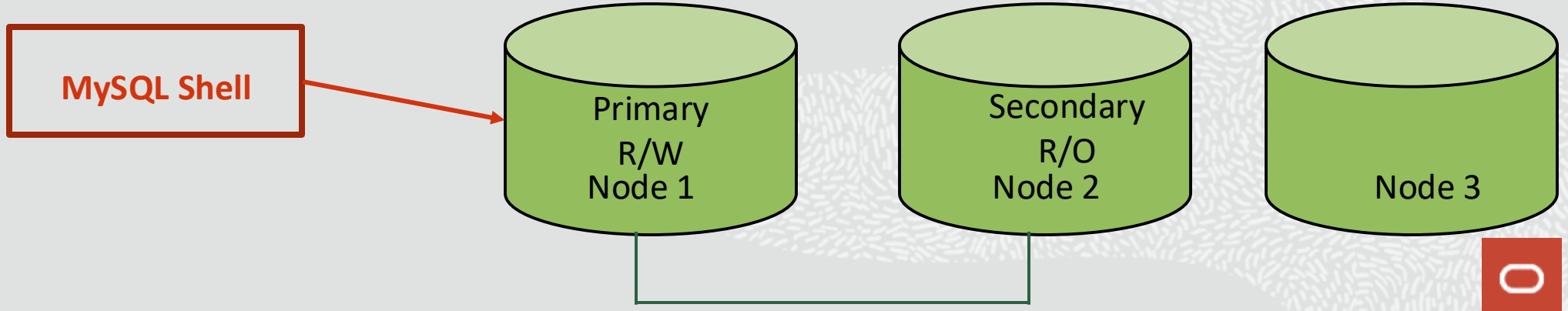
MySQL Shell : Setup InnoDB Cluster easily

- Connect to one server and create cluster with this member

```
MySQL  JS > \connect admin@10.0.0.75
MySQL  10.0.0.75:33060+ ssl  JS > Cluster = dba.createCluster("testcluster")
```

- Add the other two members to the cluster

```
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.76:3306")
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.77:3306")
```



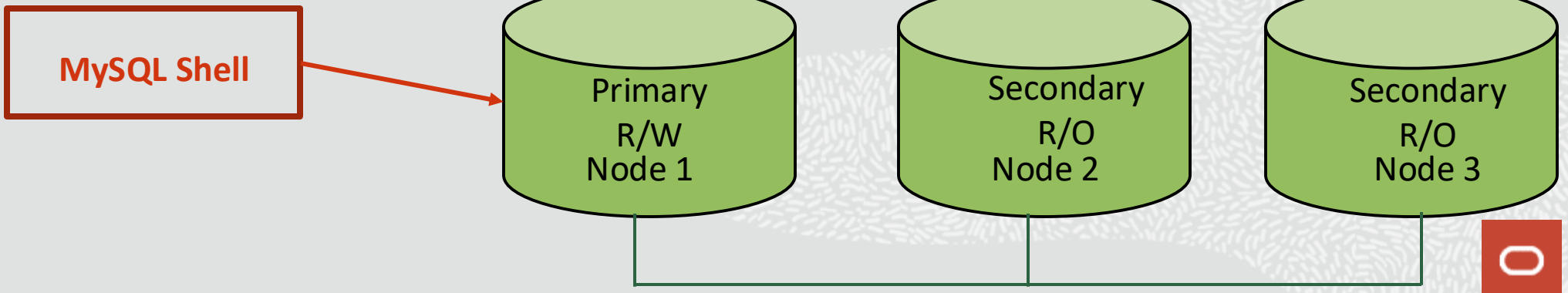
MySQL Shell : Setup InnoDB Cluster easily

- Connect to one server and create cluster with this member

```
MySQL  JS > \connect admin@10.0.0.75
MySQL  10.0.0.75:33060+ ssl  JS > Cluster = dba.createCluster("testcluster")
```

- Add the other two members to the cluster

```
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.76:3306")
MySQL  10.0.0.75:33060+ ssl  JS > Cluster.addInstance("admin@10.0.0.77:3306")
```



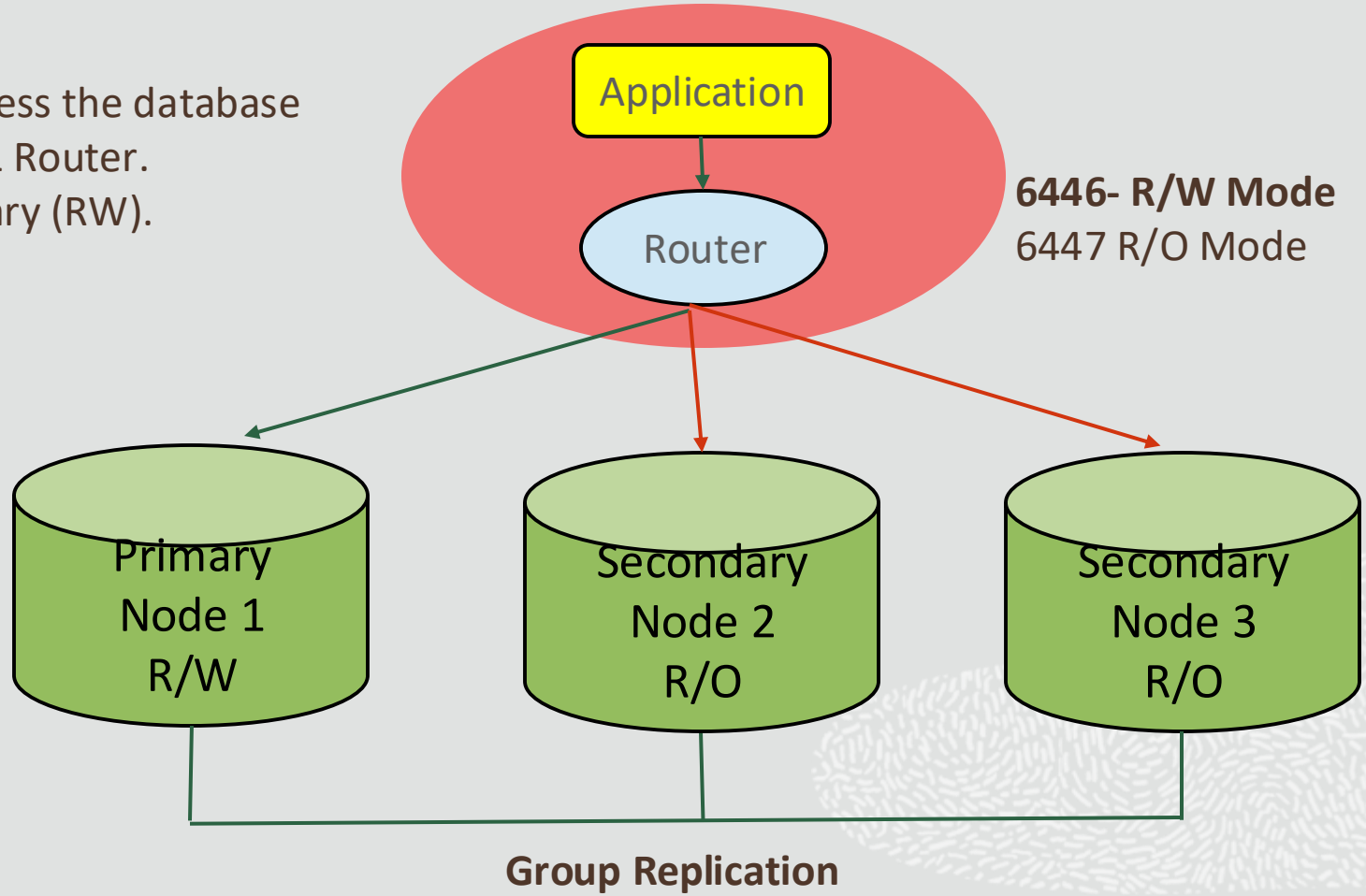
Check the status of the cluster

```
MySQL 10.0.0.75:33060+ ssl JS > Cluster.status()
```


```
"clusterName": "testCluster",
"defaultReplicaSet": {
    "name": "default",
    "primary": "student102-serverb:3307",
    "ssl": "REQUIRED",
    "status": "OK",
    "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
    "topology": {
        "student102-servera:3307": {
            "address": "student102-servera:3307",
            "mode": "R/O",
            "readReplicas": {},
            "replicationLag": null,
            "role": "HA",
            "status": "ONLINE",
            "version": "8.0.18"
        },
        "student102-serverb:3307": {
            "address": "student102-serverb:3307",
            "mode": "R/W",
            "readReplicas": {},
            "replicationLag": null,
            "role": "HA",
            "status": "ONLINE",
            "version": "8.0.18"
        },
        "student102-serverb:3317": {
            "address": "student102-serverb:3317",
            "mode": "R/O",
            "readReplicas": {},
            "replicationLag": null,
            "role": "HA",
            "status": "ONLINE",
            "version": "8.0.18"
        }
    },
    "topologyMode": "Single-Primary"
}
```

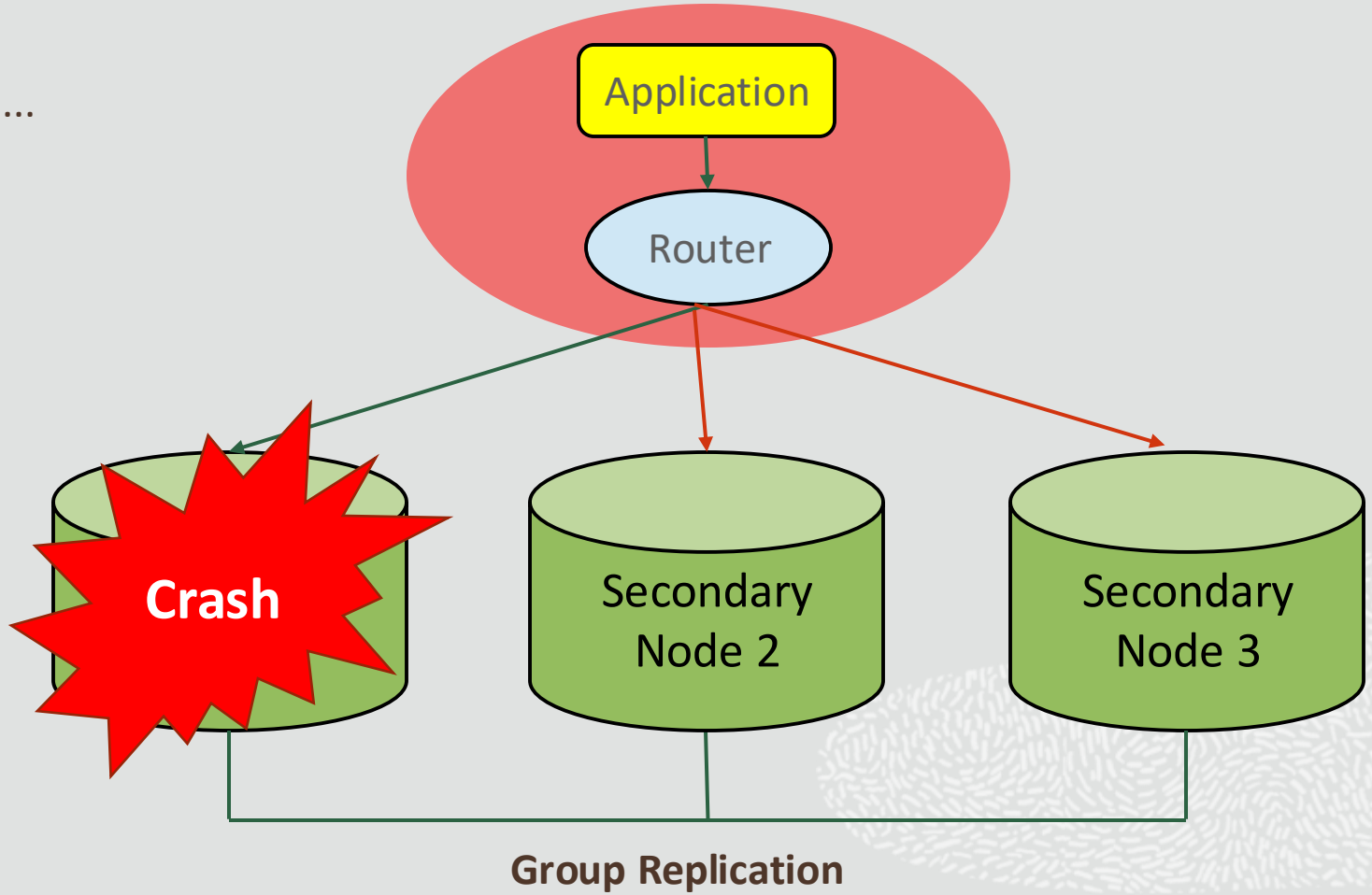
InnoDB Cluster: Application access

Application access the database through MySQL Router.
Node 1 is Primary (RW).



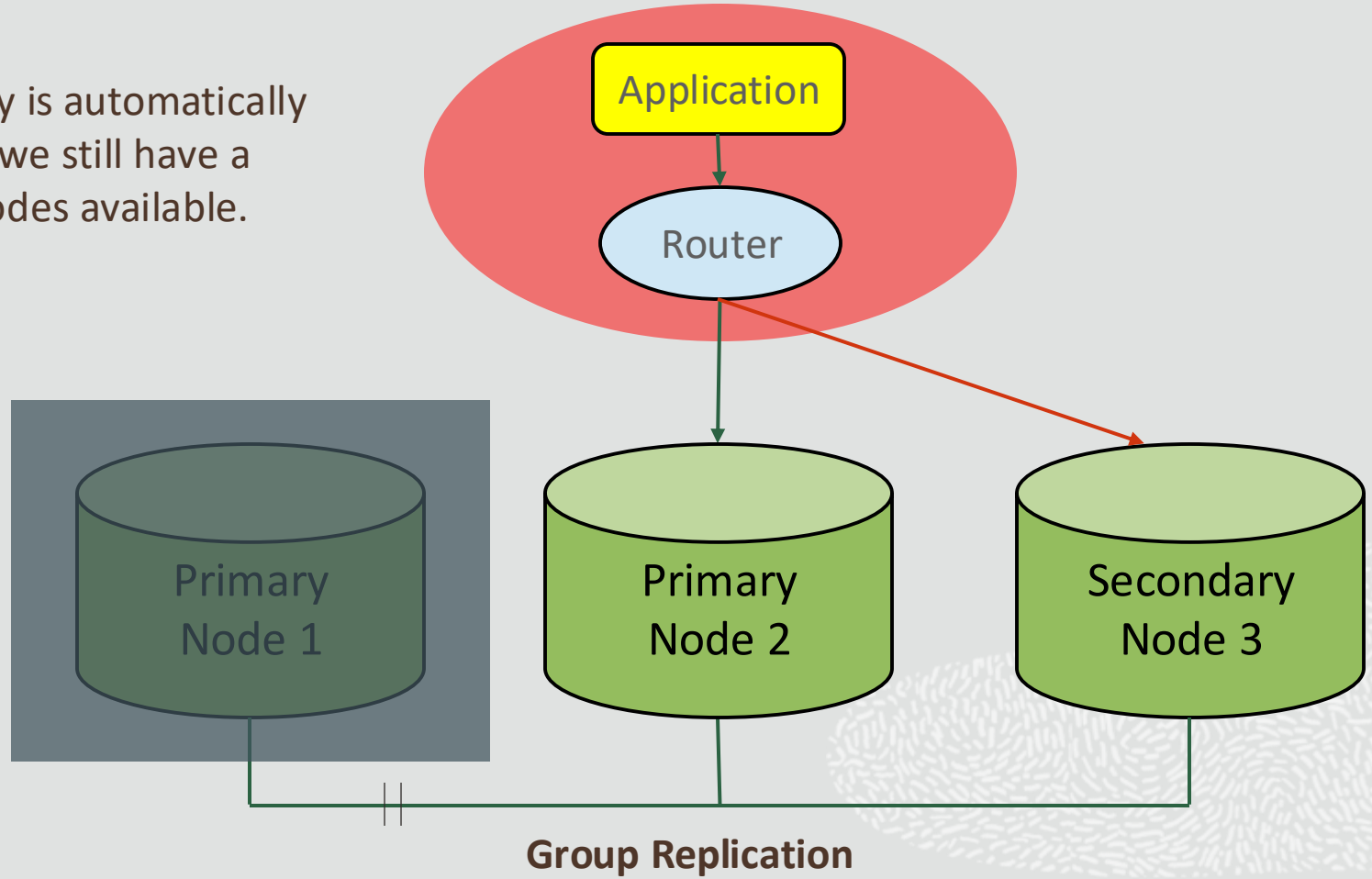
InnoDB Cluster: Failure detection

Crash happens ...



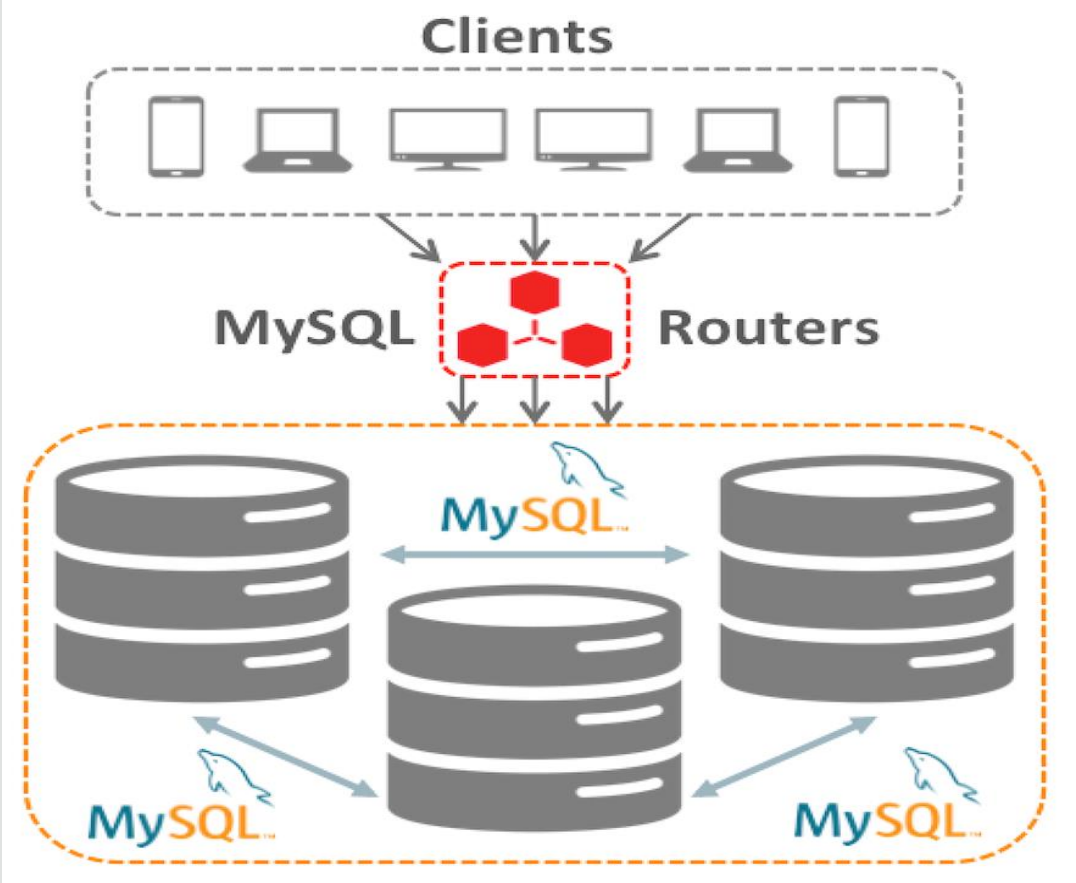
InnoDB Cluster: Failover PRIMARY Role

A new primary is automatically elected since we still have a majority of nodes available.



MySQL InnoDB Cluster

- Automate Failover of MySQL and Client Connections
- Reduce Downtime with a Speedy Failover
- Less Risk for Data Inconsistencies with virtually synchronous replication



- Developed and Maintained by MySQL
- Enterprise tools & support to improve manageability
- Works on all MySQL Supported Platforms

MySQL 8.0: Alter Table - Instant Add Column

Contribution from Tencent

- Only a metadata change
- No copying of data
- Smaller final data size
- Forward compatibility with old data file

ALTER TABLE … ADD COLUMN c, ALGORITHM = INSTANT

Supports DYNAMIC/COMPACT/REDUNDANT row formats

MySQL Instant Add Column- Study & Findings

To test the various ALTER algorithm COPY, INSTANT, INPLACE. I am going to demonstrate table called *tbl_ITRequesthistory* ,which holds 15 millions of **real time customer data.**

MySQL 8.0.13 ALTER TABLE with **COPY** Algorithm.

Command to ADD column in table:mysql> alter table tbl_ITRequestHistory add column col1 varchar(20) , **algorithm=copy;**

```
mysql> alter table tbl_ITRequestHistory add column col1 varchar(20) , algorithm=copy;
Query OK, 1573415 rows affected (50.17 sec)
Records: 1573415  Duplicates: 0  Warnings: 0

mysql>
```

MySQL 8.0.13 ALTER TABLE with **INPLACE** Algorithm

Command to ADD column in table: mysql> alter table tbl_ITRequestHistory add column col10 varchar(20) , **algorithm=inplace;**

```
mysql> alter table tbl_ITRequestHistory add column col10 varchar(20) , algorithm=inplace;
Query OK, 0 rows affected (20.59 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

MySQL 8.0.13 ALTER TABLE with **INSTANT** Algorithm.

Command to ADD column in table : mysql> alter table tbl_ITRequestHistory add column col15 varchar(20) , **algorithm=instant;**

```
mysql> alter table tbl_ITRequestHistory add column col16 varchar(20) , algorithm=instant;
Query OK, 0 rows affected (0.07 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> _
```

MySQL Instant Add Column - Results

MySQL 8.0.13 ALTER MODE	COPY Algorithm	INPLACE Algorithm	INSTANT Algorithm
Time	50.17 sec	20.59 sec	0.07 sec

MySQL 8.0 - FUNCTIONAL INDEXES

- Index over an expression

```
CREATE TABLE t1 (col1 INT, col2 INT);
CREATE INDEX idx1 ON t1 ((col1 + col2), (col1 - col2), col1) ;
```

- Document content, e.g. JSON array

```
CREATE TABLE lottery (data JSON);
CREATE INDEX ticket_idx ON lottery
        ((CAST(data->'$.lottery_tickets' AS UNSIGNED INT ARRAY))) ;
```

MySQL 8.0 – INVISIBLE INDEXES

- Indexes are “hidden” to the MySQL Optimizer
  - Not the same as “disabled indexes”
  - Contents are fully up to date and maintained by DML
- Two use cases:
  - Soft Delete: What will happen if I delete this index?
  - Staged Rollout: I will create this index over night and make it visible when I am at work tomorrow

```
ALTER TABLE Country ADD INDEX c (Continent) INVISIBLE;
<<SQL Statements>
# after some time
ALTER TABLE Country ALTER INDEX c VISIBLE;
```

MySQL 8.0 – INVISIBLE COLUMN

- Prior to MySQL 8.0.23, all columns are visible.
- At least one column has to be visible.
- An invisible column is normally hidden to queries, but can be accessed if explicitly referenced.
- Useful:-
- *SELECT * queries to access a table, and must continue to work without modification even if the table is altered to add a new column that the application does not expect to be there.*

- *In a SELECT * query, the * evaluates to all table columns, except those that are invisible, so the solution is to add the new column as an invisible column. The column remains "hidden" from SELECT * queries, and the application continues to work as previously.*

mysql> create table employee(empid INT ,empname Varchar(20));

mysql> Insert into employee values(100,'Ram');

mysql> alter table employee add column deptname varchar(20);

mysql> Insert into employee values(200,'Shayam');
ERROR 1136 (21S01): Column count doesn't match value count at row 1

After Adding INVISIBLE COLUMN

mysql> alter table employee modify deptname varchar(20) **INVISIBLE**;
Query OK, 0 rows affected (0.04 sec)

mysql> mysql> Insert into employee values(200,'Shayam');
Query OK, 1 row affected (0.01 sec)
mysql> select * from employee;

```
+-------+---------+
| empid | empname |
+-------+---------+
|   100 | Ram     |
|   200 | Shayam  |
+-------+---------+--
```

2 rows in set (0.00 sec)
mysql> alter table employee modify deptname varchar(20) **VISIBLE**;
Query OK, 0 rows affected (0.04 sec)

5 minute break

MySQL 8 - CHECK CONSTRAINT

- Standard SQL Syntax

[CONSTRAINT [symbol]] CHECK (condition) [[NOT] ENFORCED]

- Example

```
CREATE TABLE t1 (c1 INTEGER CONSTRAINT c1_chk CHECK (c1 > 0) ,
                 c2 INTEGER CONSTRAINT c2_chk CHECK (c2 > 0) ,
                 CONSTRAINT c1_c2_chk CHECK (c1 + c2 < 9999) );
```

CHECK CONSTRAINTS

We already saw that 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)
```

MySQL 8 - NOWAIT and SKIP LOCKED

```
SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
NOWAIT;
```

Error immediately if a row is already locked

```
SELECT * FROM tickets
WHERE id IN (1,2,3,4)
AND order_id IS NULL
FOR UPDATE
SKIP LOCKED;
```

Non deterministically skip over locked rows

Let us take an Example

```
mysql> SELECT * FROM Employee;
+-------+---------+----------+--------+
| EmpId | Empname | DeptName | Salary |
+-------+---------+----------+--------+
|     1 | Neeraj  | .Net     |  45010 |
|     2 | Ankit   | Java     |   5010 |
|     3 | Akshay  | Java     |   6010 |
|     4 | Ramesh  | .Net     |   7610 |
|     5 | Vikas   | Java     |   4010 |
|     6 | Neha    | Php      |   8500 |
|     7 | Shivika | Php      |   4500 |
|     8 | Tarun   | .Net     |   9500 |
+-------+---------+----------+--------+
8 rows in set (0.00 sec)
```

```
SESSION 1>START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

SESSION 1>SELECT * FROM Employee WHERE EmpId IN (1,2,3,4,5) FOR UPDATE;
+-------+---------+----------+--------+
| EmpId | Empname | DeptName | Salary |
+-------+---------+----------+--------+
|     1 | Neeraj  | .Net     |  45010 |
|     2 | Ankit   | Java     |   5010 |
|     3 | Akshay  | Java     |   6010 |
|     4 | Ramesh  | .Net     |   7610 |
|     5 | Vikas   | Java     |   4010 |
+-------+---------+----------+--------+
5 rows in set (0.00 sec)
```

```
SESSION 2>SELECT * FROM Employee WHERE EmpId IN (4,5,6,7,8,9) FOR UPDATE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
SESSION 2>
```

waited for innodb_lock_wait_timeout ~50s default

why lock on record 6,7,8,9 ??

```
SESSION 2>SELECT * FROM Employee WHERE EmpId IN (4,5,6,7,8,9) FOR UPDATE SKIP LOCKED;
+-------+---------+----------+--------+
| EmpId | Empname | DeptName | Salary |
+-------+---------+----------+--------+
|     6 | Neha    | Php      |   8500 |
|     7 | Shivika | Php      |   4500 |
|     8 | Tarun   | .Net     |   9500 |
+-------+---------+----------+--------+
3 rows in set (0.00 sec)
```

Let us take an Example

```
mysql> SELECT * FROM Employee;
+-------+---------+----------+--------+
| EmpId | Empname | DeptName | Salary |
+-------+---------+----------+--------+
|     1 | Neeraj  | .Net     |  45010 |
|     2 | Ankit   | Java     |   5010 |
|     3 | Akshay  | Java     |   6010 |
|     4 | Ramesh  | .Net     |   7610 |
|     5 | Vikas   | Java     |   4010 |
|     6 | Neha    | Php      |   8500 |
|     7 | Shivika | Php      |   4500 |
|     8 | Tarun   | .Net     |   9500 |
+-------+---------+----------+--------+
8 rows in set (0.00 sec)
```

```
SESSION 1>START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

SESSION 1>SELECT * FROM Employee WHERE EmpId IN (1,2,3,4,5) FOR UPDATE;
+-------+---------+----------+--------+
| EmpId | Empname | DeptName | Salary |
+-------+---------+----------+--------+
|     1 | Neeraj  | .Net     |  45010 |
|     2 | Ankit   | Java     |   5010 |
|     3 | Akshay  | Java     |   6010 |
|     4 | Ramesh  | .Net     |   7610 |
|     5 | Vikas   | Java     |   4010 |
+-------+---------+----------+--------+
5 rows in set (0.00 sec)
```

```
SESSION 2>SELECT * FROM Employee WHERE EmpId IN (4,5,6,7,8,9) FOR UPDATE;
ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction
SESSION 2>
```

waited for innodb_lock_wait_timeout ~50s default

Why to wait till timeout???

```
SESSION 2>SELECT * FROM Employee WHERE EmpId IN (4,5,6,7,8,9) FOR UPDATE NOWAIT;
ERROR 3572 (HY000): Statement aborted because lock(s) could not be acquired immediately and NOWAIT is set.
SESSION 2>SELECT * FROM Employee WHERE EmpId IN (4,5) FOR UPDATE NOWAIT;
ERROR 3572 (HY000): Statement aborted because lock(s) could not be acquired immediately and NOWAIT is set.
SESSION 2>
```

Account Locking After Failed logins

- The Ability to Lock an account on a MySQL Instance after two many failed logins.
- Introduced in 8.01.9

Syntax

- CREATE USER 'demo'@'localhost'
  IDENTIFIED by '****'
  FAILED_LOGIN_ATTEMPTS 3
  PASSWORD_LOCK_TIME 2;
- You can also use

PASSWORD_LOCK_TIME **UNBOUNDED**; to keep the account locked.

The range for the lock time can range from 0 (which disables locking) to 32767 (32767 days is 89.7 years!).

To Unlock the accounts.

ALTER USER 'demo'@'localhost' ACCOUNT UNLOCK;

Example

```
CREATE USER 'demo'@'localhost' IDENTIFIED by 'MySQL8.0'
FAILED_LOGIN_ATTEMPTS 3 PASSWORD_LOCK_TIME 2;
C:\>mysql -udemo -p
Enter password: ******
ERROR 1045 (28000): Access denied for user 'demo'@'localhost' (using password:
YES)

C:\>mysql -udemo -p
Enter password: *******
ERROR 1045 (28000): Access denied for user 'demo'@'localhost' (using password:
YES)

C:\>mysql -udemo -p
Enter password: ********
ERROR 3955 (HY000): Access denied for user 'demo'@'localhost'. Account is blocked
for 2 day(s) (2 day(s) remaining) due to 3 consecutive failed logins.
C:\>
```

Dual Password Support

- User accounts can have a primary password and a secondary password.
- Introduced in 8.0.14

- Dual-password capability makes it possible to seamlessly perform credential changes in scenarios like this:
- A system has a large number of MySQL servers, possibly involving replication.
- Multiple applications connect to different MySQL servers.
- Periodic credential changes must be made to the account or accounts used by the applications to connect to the servers.

Example

ALTER USER 'demo'@'localhost' IDENTIFIED BY 'MySQL8.1' RETAIN CURRENT PASSWORD;

C:\>mysql -udemo -p
Enter password: ******
ERROR 1045 (28000): Access denied for user 'demo'@'localhost' (using password: YES)

C:\>mysql -udemo –p
Enter password: *******

And How Do I Get Rid Of A Secondary Password ?

ALTER USER 'demo'@'localhost' DISCARD OLD PASSWORD;

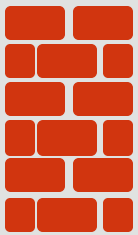
Break for 10 Minute
12 PM will start

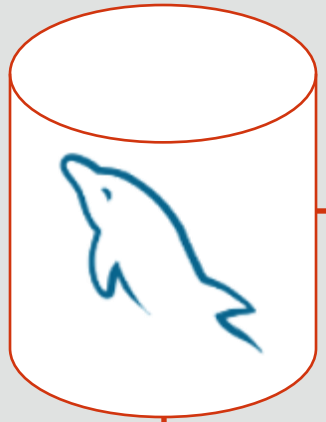
MySQL Enterprise Security

Centralized user account management
Active directory, LDAP, 2FA, Kerberos, Sasl …

Database Firewall
Allow/Deny lists
Block SQL injection attacks

Audit Compliance
Granular database object and user audit
Compatible with the major SIEM tools





- Oracle Audit Vault
- Splunk
- IBM Guardian
- ArcSight
- Others

#####

- KMIP v1.2 (SafeNet, VaultCore, Townsend, Vormetric, etc.)
- AWS
- Hashicorp

Data Masking and De-Identification
Protect confidential information

Transparent Data Encryption
Protect data against theft

MySQL Enterprise Edition

Advanced Features

- Scalability
- High Availability
- Security
- Encryption + TDE
- Data Masking
- Firewall

Management Tools

- Monitoring
- Backup
- Development
- Administration
- Migration

Support

- Technical Support
- Consultative Support
- Oracle Certifications

MySQL 8 - The complete list of new features

https://mysqlserverteam.com/the-complete-list-of-new-features-in-mysql-8-0/

References

- Changes in MySQL 8.0.24
  - https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-24.html
- Top Blog Posts about MySQL 8.0.24
  - https://mysqlserverteam.com/the-mysql-8-0-24-maintenance-release-is-generally-available/

  https://planet.mysql.com/

MySQL on Social Media

https://www.facebook.com/mysql

https://twitter.com/mysql

https://www.linkedin.com/company/mysql

ORACLE