MySQL

ORACLE

MySQL Sharding:

Tools and Best Practices for Horizontal Scaling

Mats Kindahl (mats.kindahl@oracle.com)
Alfranio Correia (alfranio.correia@oracle.com)
Narayanan Venkateswaran (v.narayanan@oracle.com)

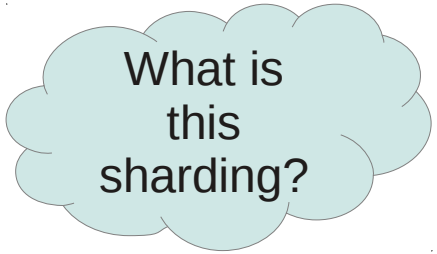
2013-09-21

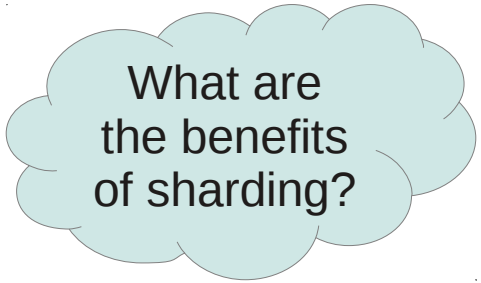
Safe Harbor Statement

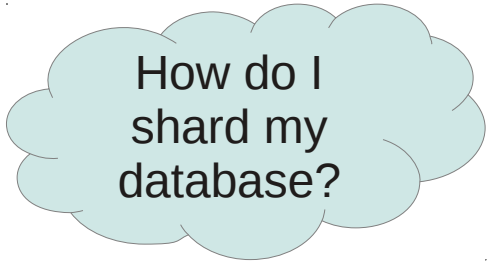
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decision. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Program Agenda

- Handling Scaling
- What is sharding?
- Managing a Sharded Database
- Working with a Sharded Database



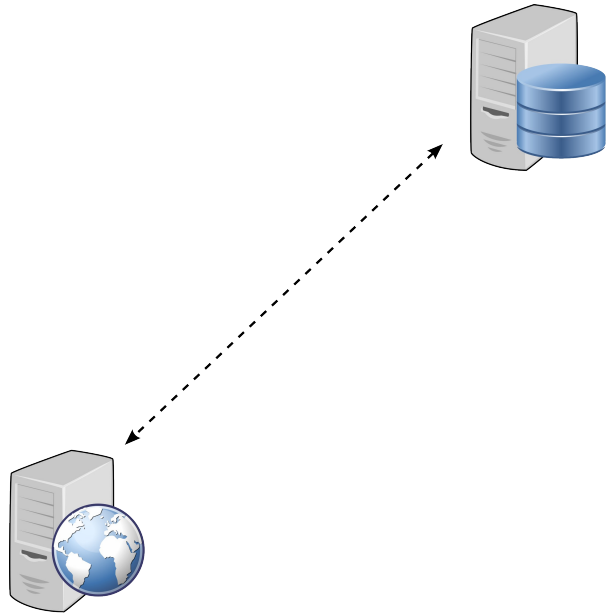




It's all about scaling...

The Growing Enterprise

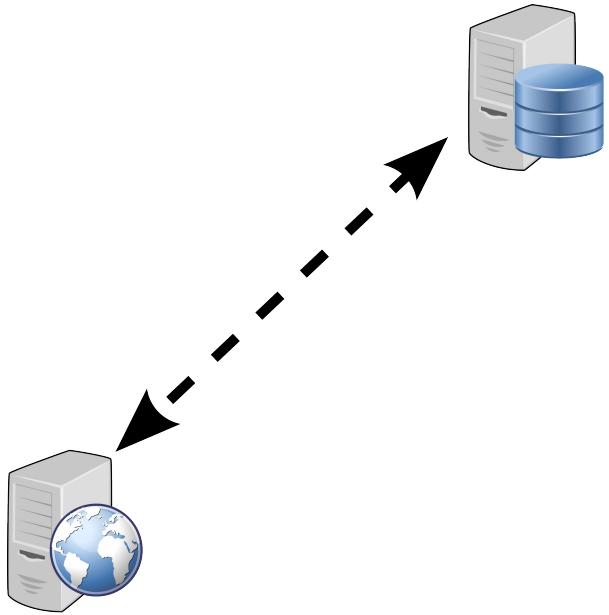
- Start with a single server



It's all about scaling...

The Growing Enterprise

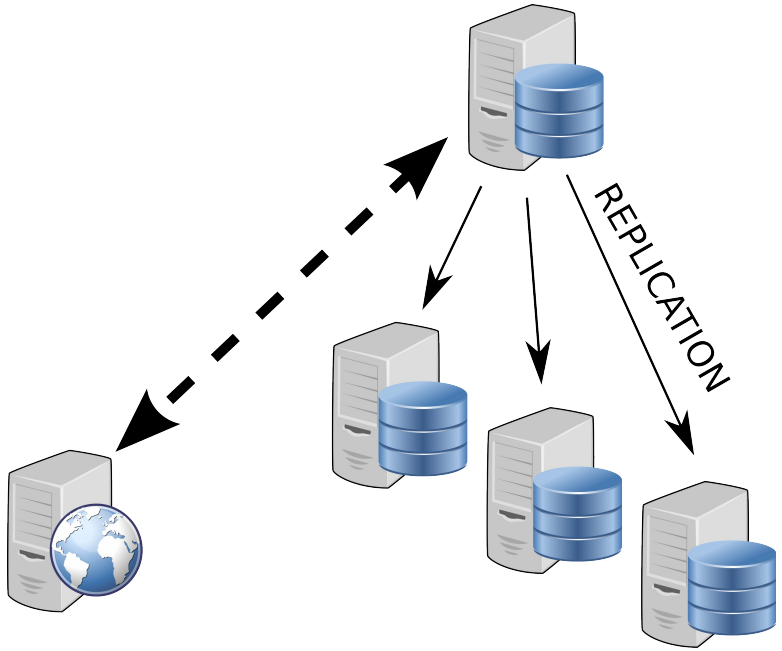
- Start with a single server
- More and more page requests
  - ...more and more reads
- What to do?
  - Scale out!



It's all about scaling...

The Growing Enterprise

- Start with a single server
- More and more page requests
  - ...more and more reads
- What to do?
  - Scale out!
- Replicate to read servers

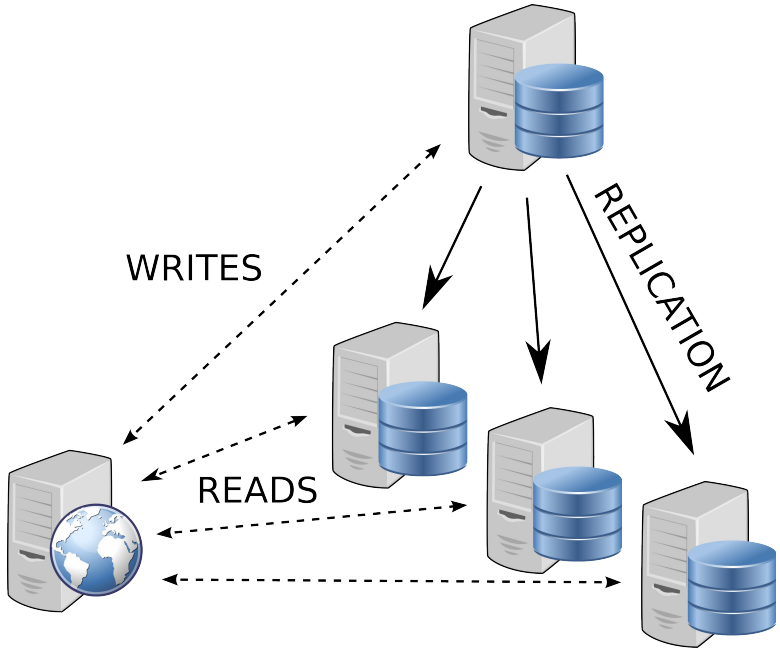


 |

It's all about scaling...

The Growing Enterprise

- Start with a single server
- More and more page requests
  - ...more and more reads
- What to do?
  - Scale out!
- Replicate to read servers
- Perform a *read-write split*
  - Writes go to master
  - Reads go to read servers

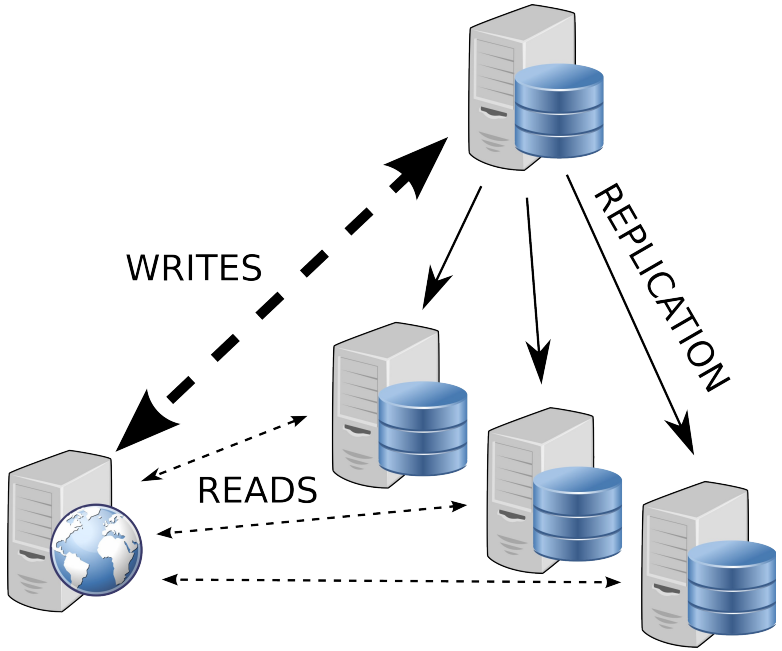


 |

It's all about scaling...

The Growing Enterprise

- More and more updates
  - Write load increases
- What now?

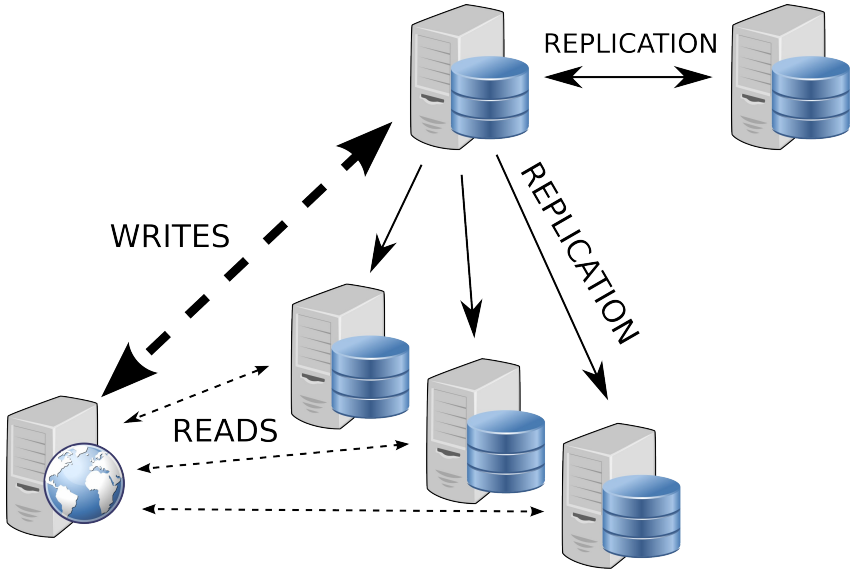


 |

It's all about scaling...

The Growing Enterprise

- More and more updates
  - Write load increases
- What now?
  - Add another master?

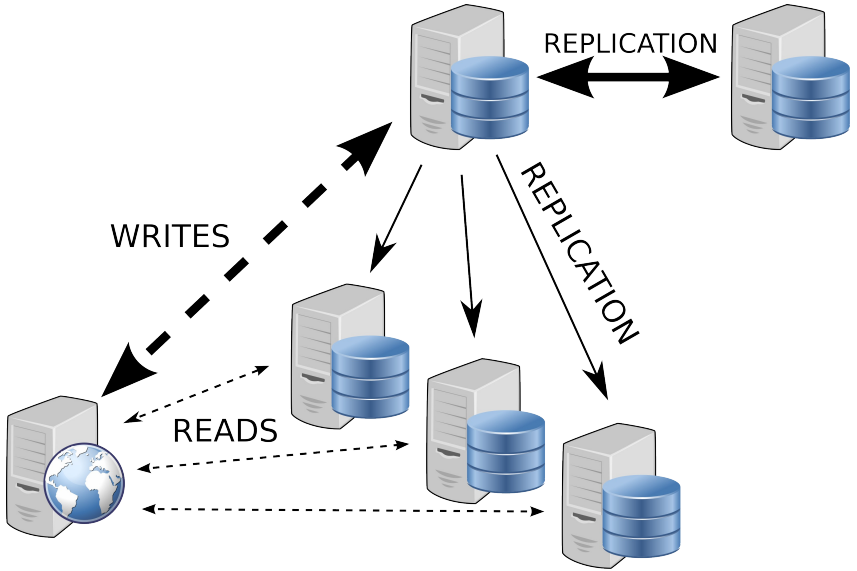


 |

It's all about scaling...

The Growing Enterprise

- More and more updates
  - Write load increases
- What now?
  - Add another master?
- Doesn't work...
  - Write load is replicated

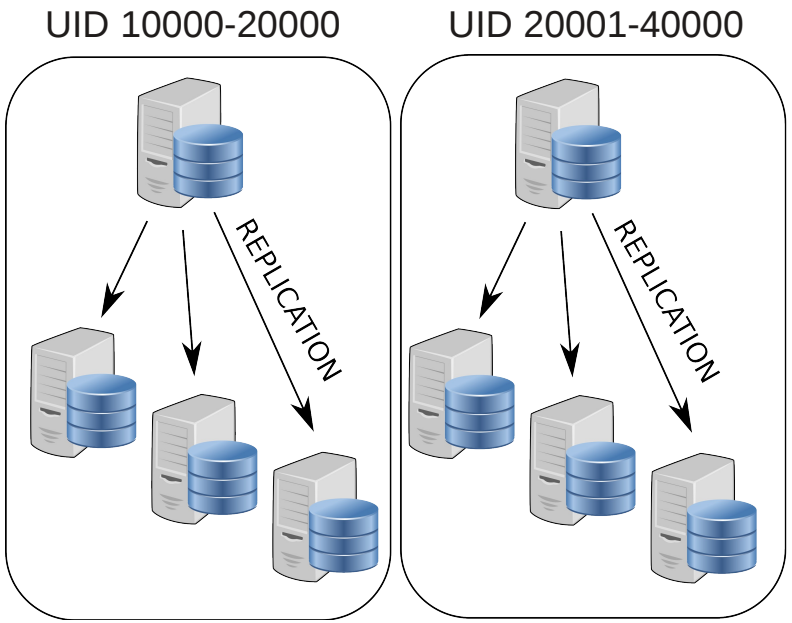


 |

It's all about scaling...

The Growing Enterprise

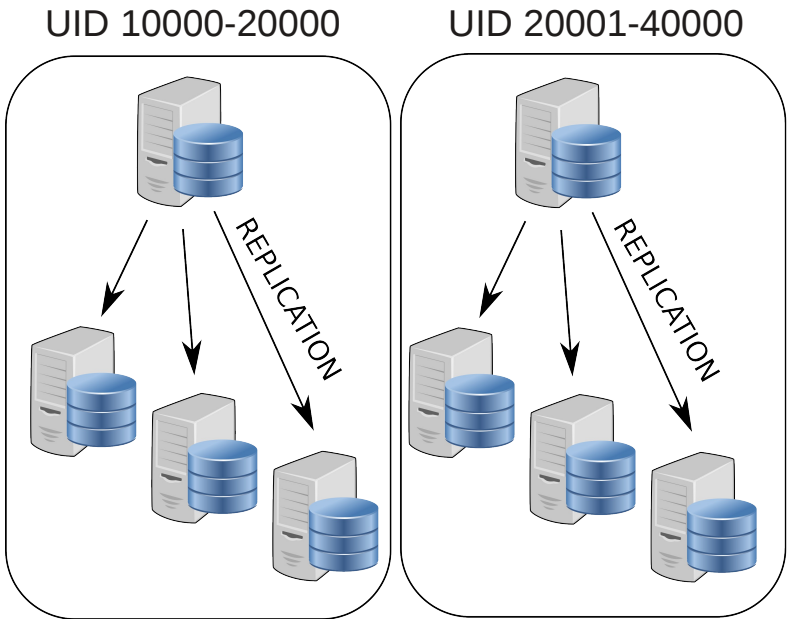
- More and more updates
  - Write load increases
- What now?
  - Add another master?
- Doesn't work...
  - Write load is replicated
- Partition database
  - Distribute writes
  - Called *sharding*



 |

Benefits of Sharding

- Write scalability
  - Can handle more writes
- Large data set
  - Database too large
  - Does not fit on single server
- Improved performance
  - Smaller index size
  - Smaller working set
  - Improve performance



Architecture of a Sharding Solution

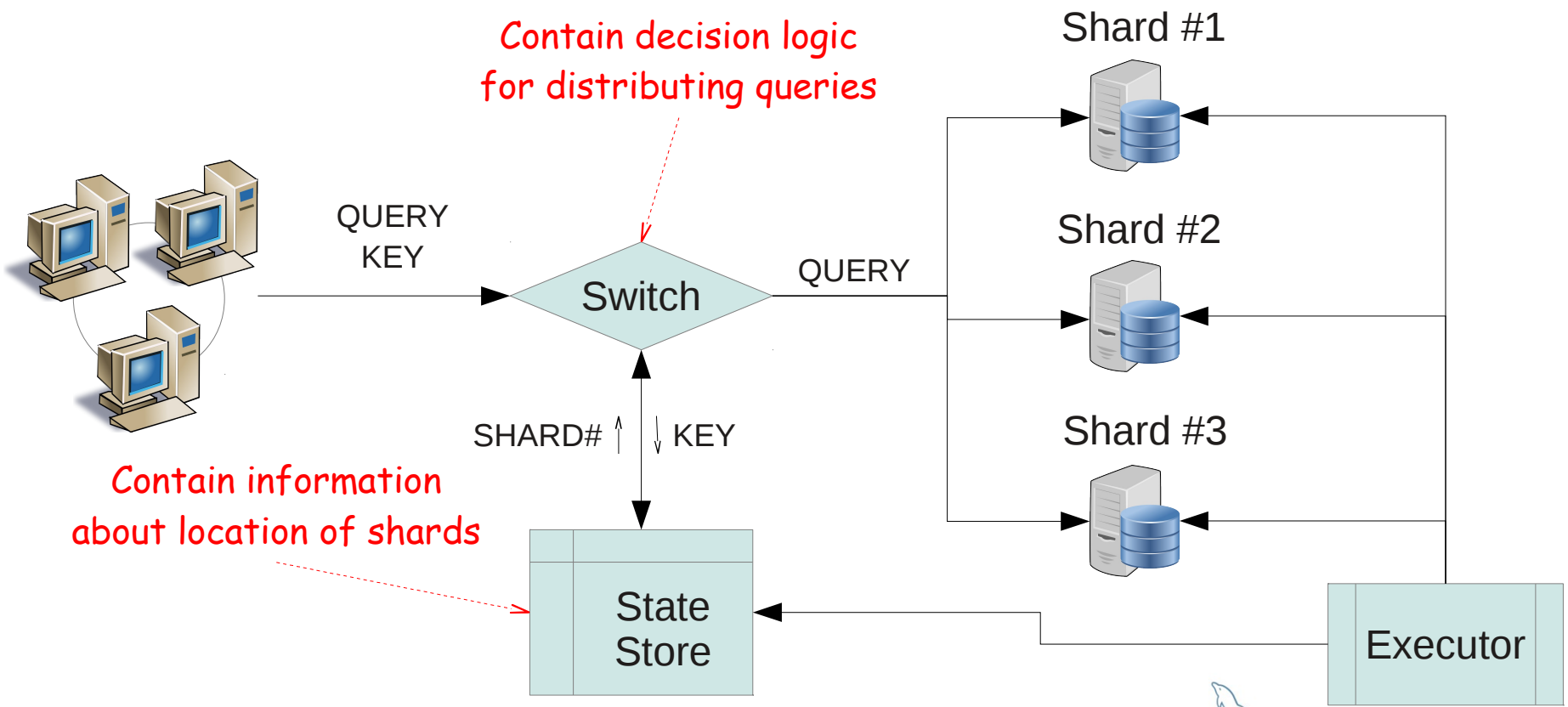
What do you need to consider?

- High level architecture
- Transaction and sharding key handling
- Granularity of sharding

High-level Architecture

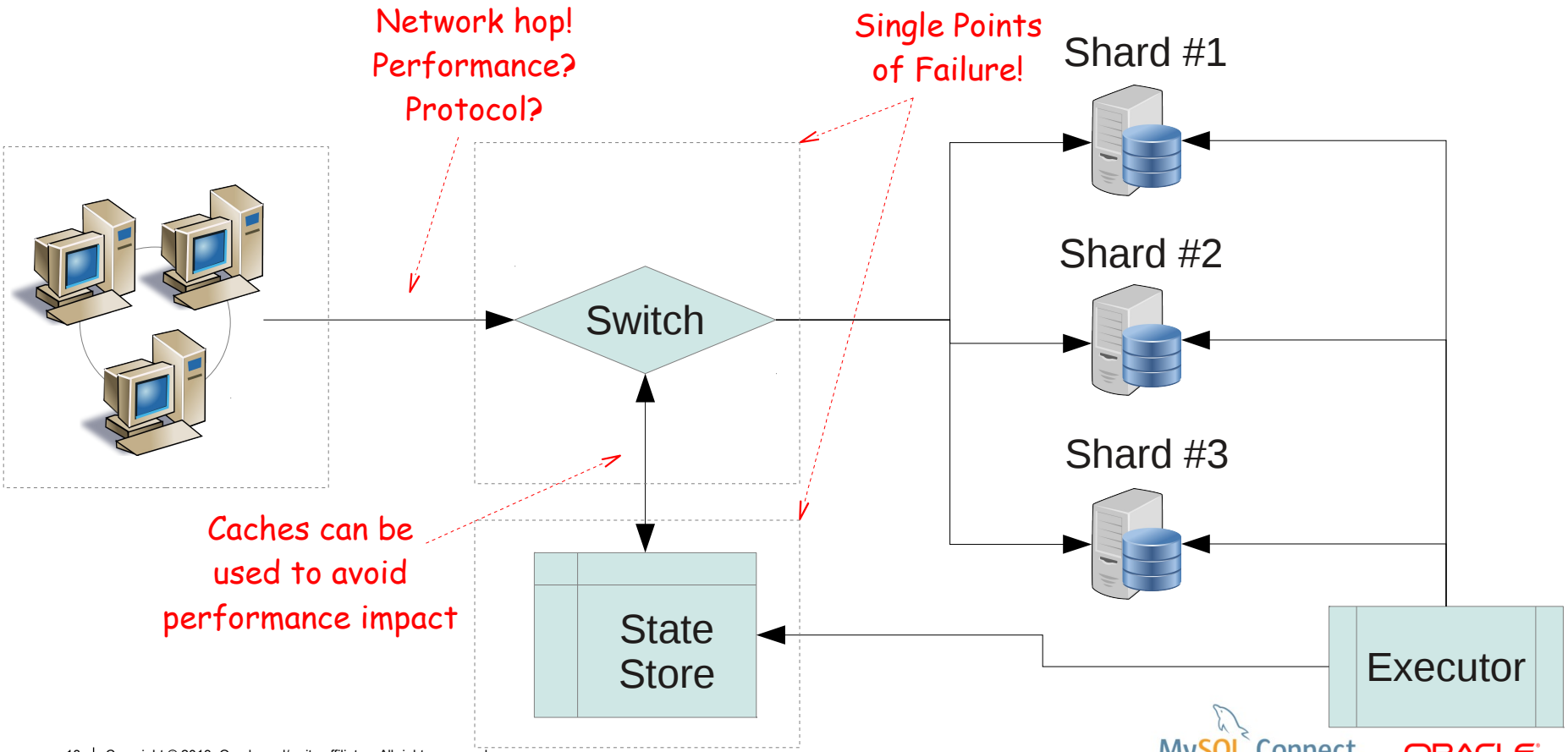
- What components do we need?
- How are they deployed?

Components for a Sharding Solution



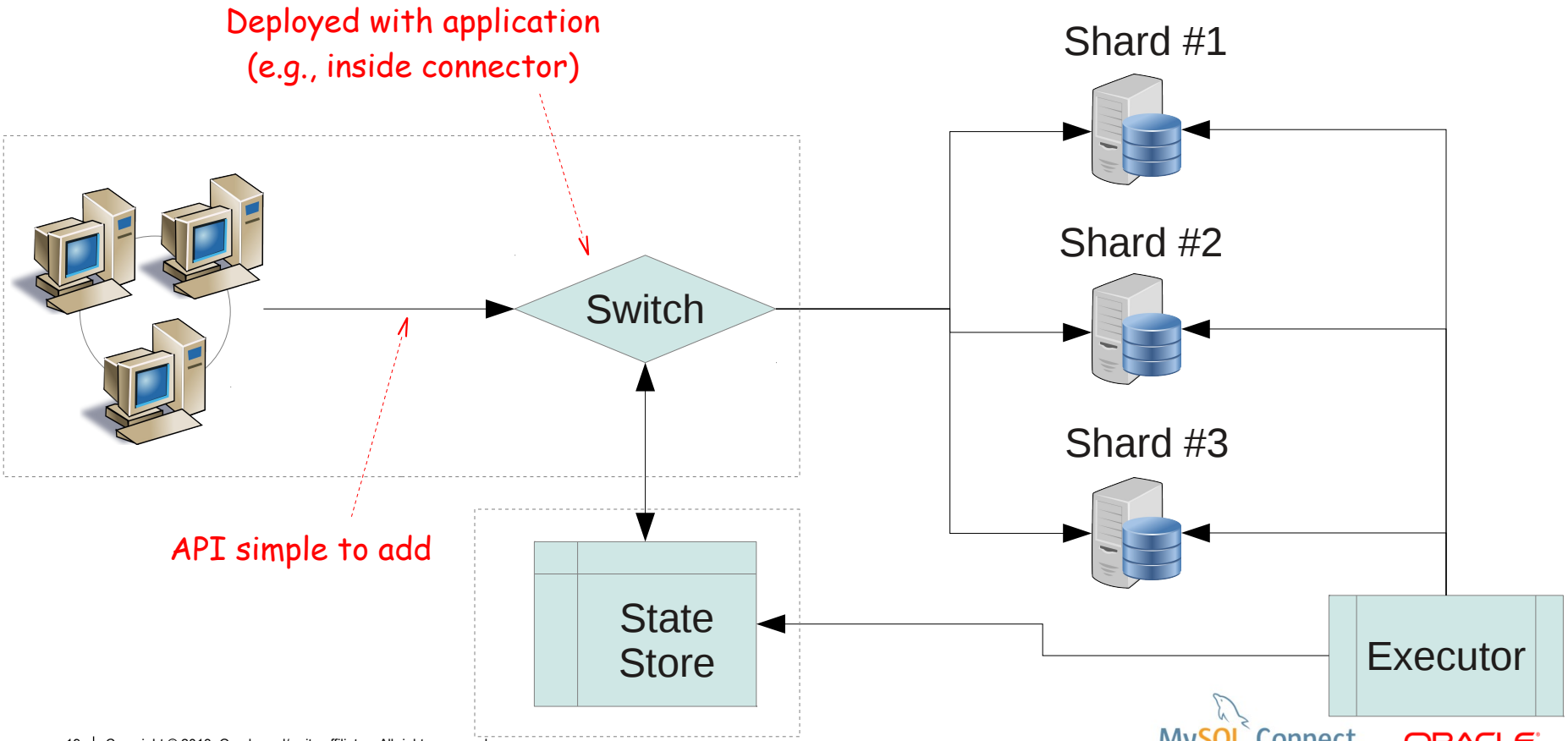
 |

Deployment of the Sharding Solution



 |

Deployment of the Sharding Solution

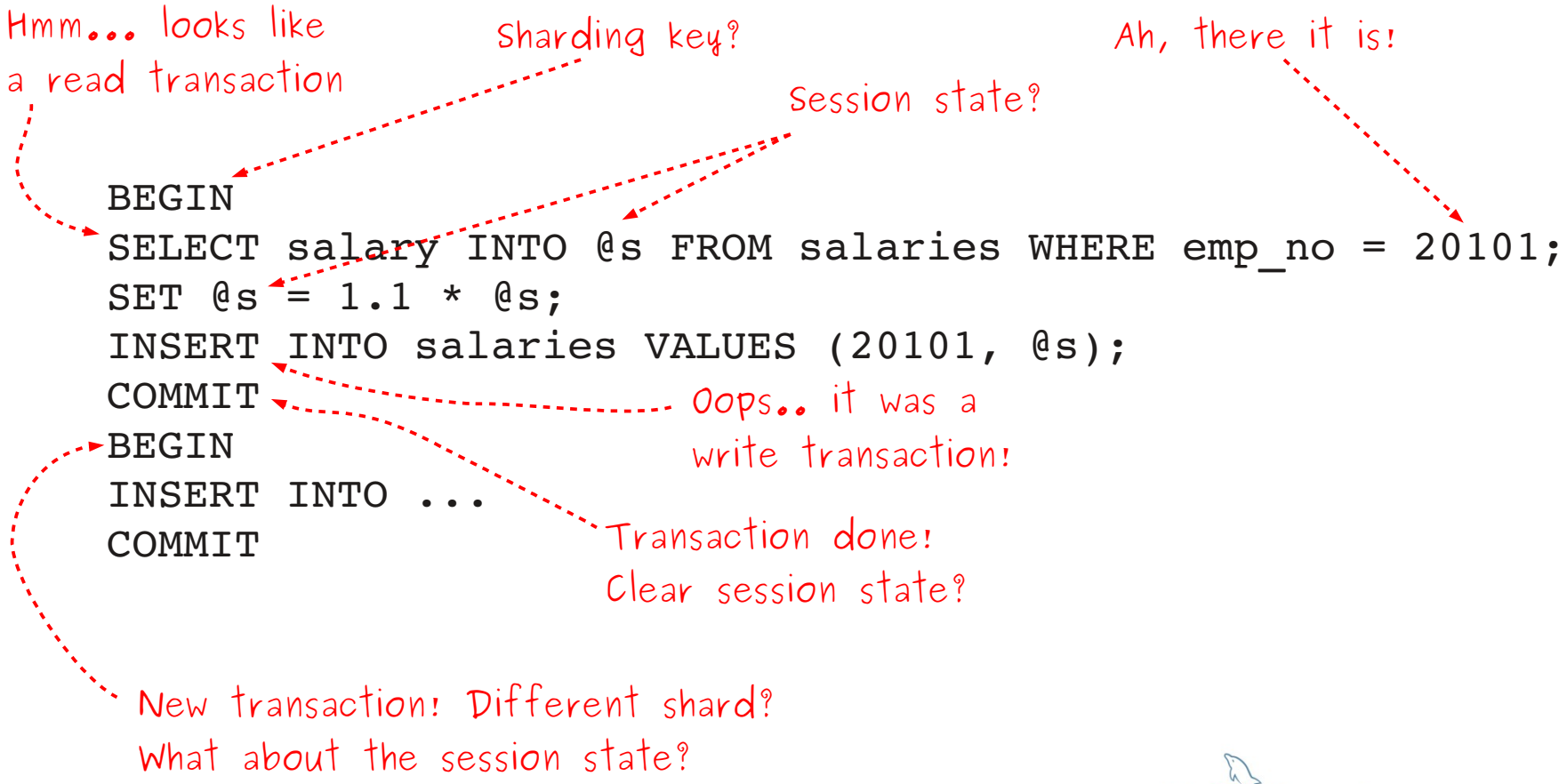


 |

Transaction and Shard Key Handling

- How are the transactions handled?
- How do get the shard key for a transaction?
- How to compute shard from key?

Transaction Handling



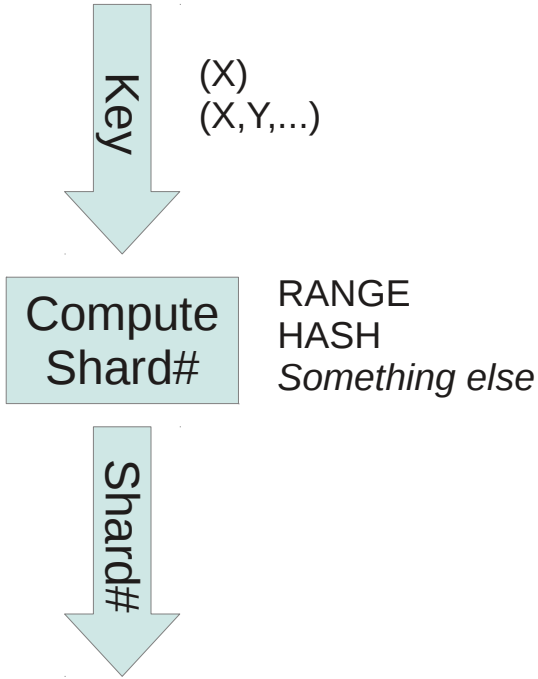
 |

Transaction Handling

- Detecting transaction boundaries
- Managing session state
  - Move session state between servers
    - Easy to use
    - Expensive and error prone
  - Reset state after each transaction
    - Transactions start with default session state

Mapping the Sharding Key

- What is a sharding key?
  - Single column
  - Multi column
    - Same table?
    - Different tables?
- How is the key transformed?
  - Hash
  - Range
  - User-defined



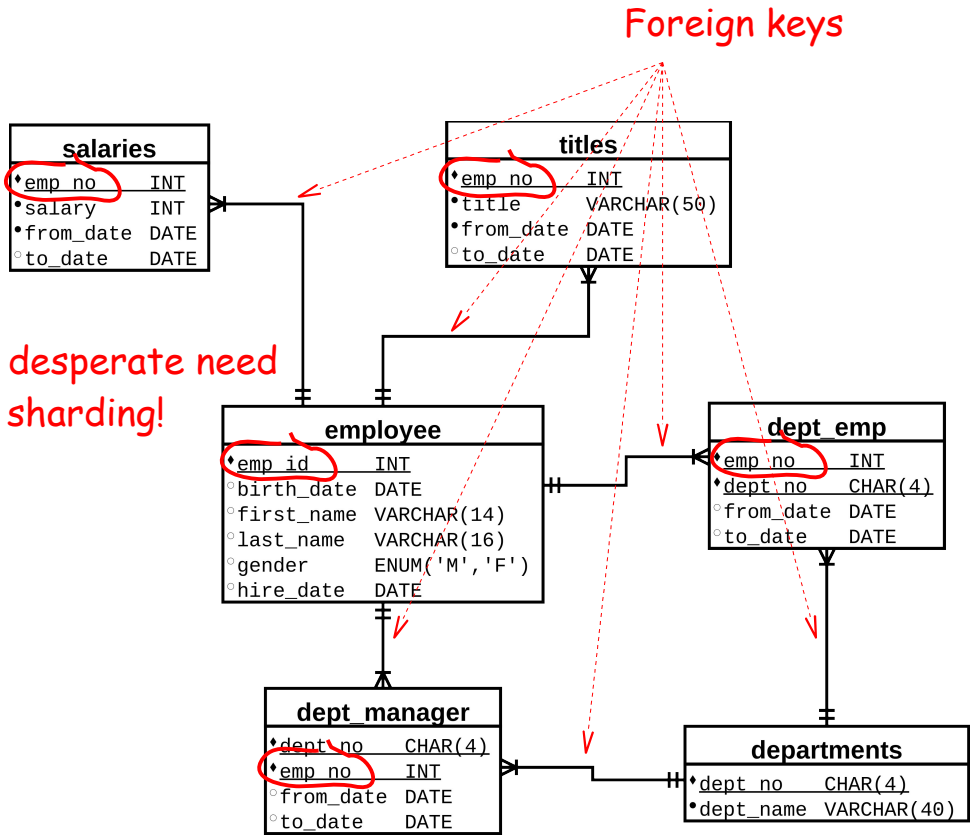
Granularity of Sharding

- Can multiple tables be sharded with the same key?
- Can we shard different tables different ways?
- Do we allow global tables?
- Do we allow cross-database queries?

Sharded Tables

Table	Rows
salaries	284 404 700
titles	44 330 800
employees	30 002 400
dept_emp	33 160 300
dept_manager	2 400
departments	900

In desperate need of sharding!



 |

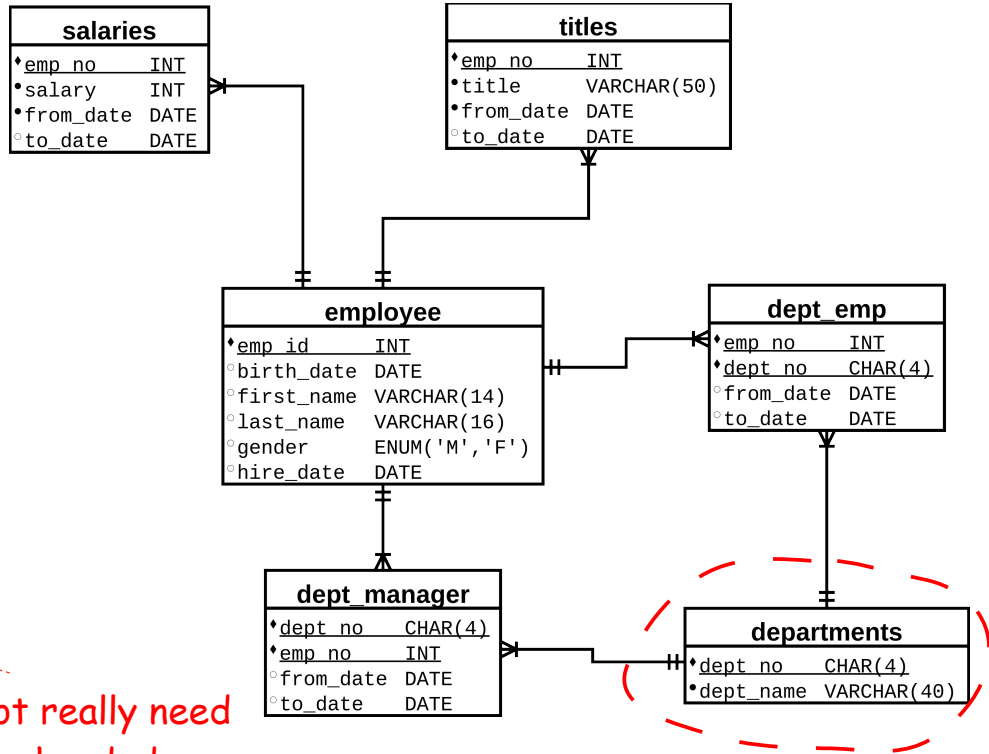
Multi-table Query with Sharded Tables

```
SELECT first_name, last_name, salary
FROM salaries JOIN employees USING (emp_no)
WHERE emp_no = 21012
    AND CURRENT_DATE BETWEEN from_date AND to_date;
```

- Referential Integrity Constraint
  - Example query joining salaries and employees
  - Same key, same shard: co-locate rows for same user
- JOIN normally based on equality
  - Using non-equality defeats purpose of foreign key

Global Tables

Table	Rows
salaries	284 404 700
titles	44 330 800
employees	30 002 400
dept_emp	33 160 300
dept_manager	2 400
departments	900



Do not really need to be sharded

 |

Multi-table Query with Global Tables

```
SELECT first_name, last_name, GROUP_CONCAT(dept_name)
  FROM employees JOIN dept_emp USING (emp_no)
                 JOIN departments USING (dept_no)
WHERE emp_no = 21012 GROUP BY emp_no;
```

- JOIN with **departments** table
  - Has no employee number, hence no sharding key
  - Table need to be present on all shards
- How do we update global tables?

Managing a Sharded Database

MySQL Fabric

An extensible and easy-to-use framework for managing a farm of MySQL server supporting high-availability and sharding

MySQL Fabric: What is it?

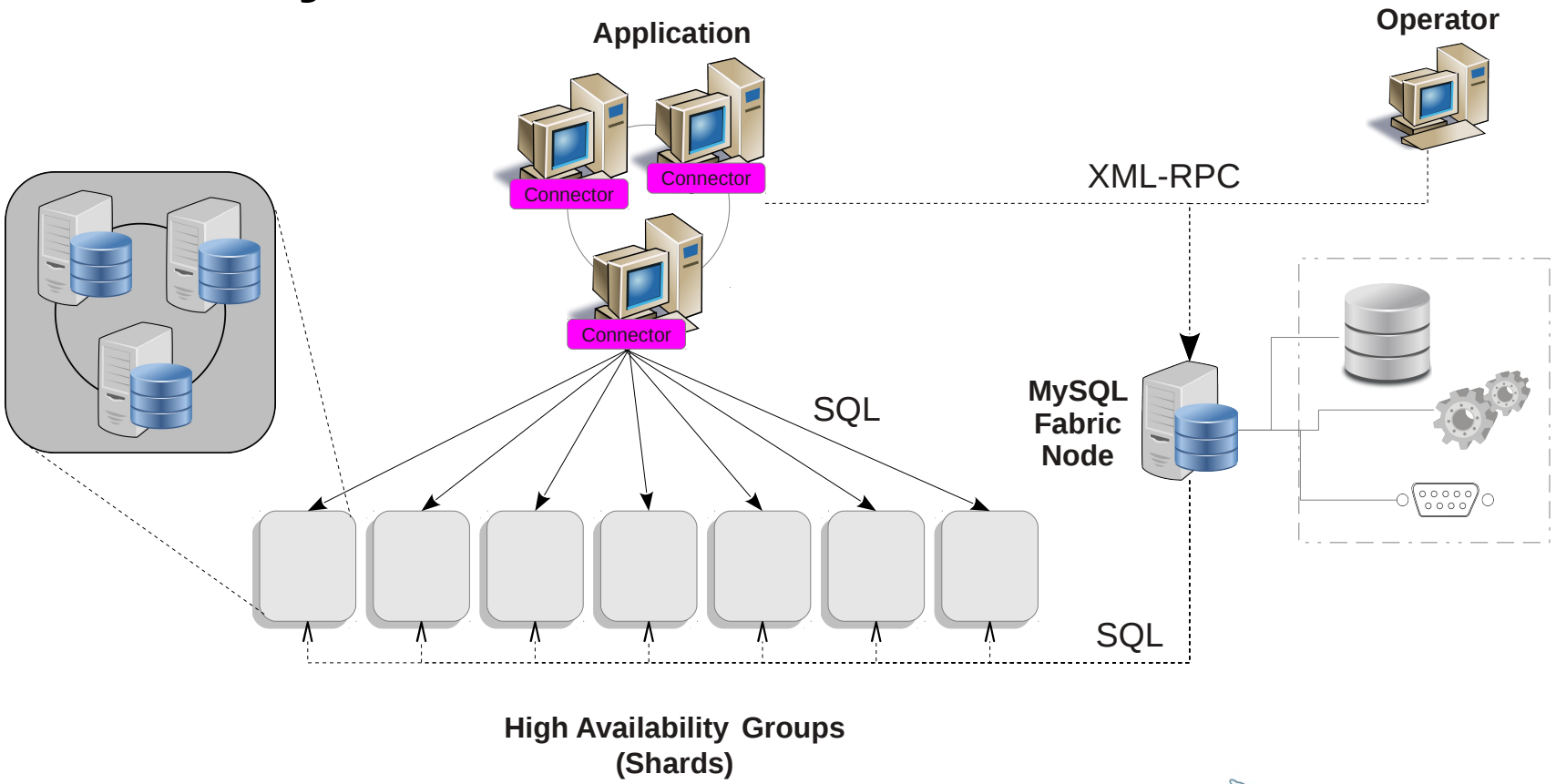
- "Farm" Management System
  - High-Availability
  - Sharding
- Distributed
- Procedure Execution
- Extensible

- Written in Python
- Early alpha
  - Long road ahead
- Open Source
  - You can participate
  - Suggest features
  - Report bugs
  - Contribute patches
- MySQL 5.6 is focus

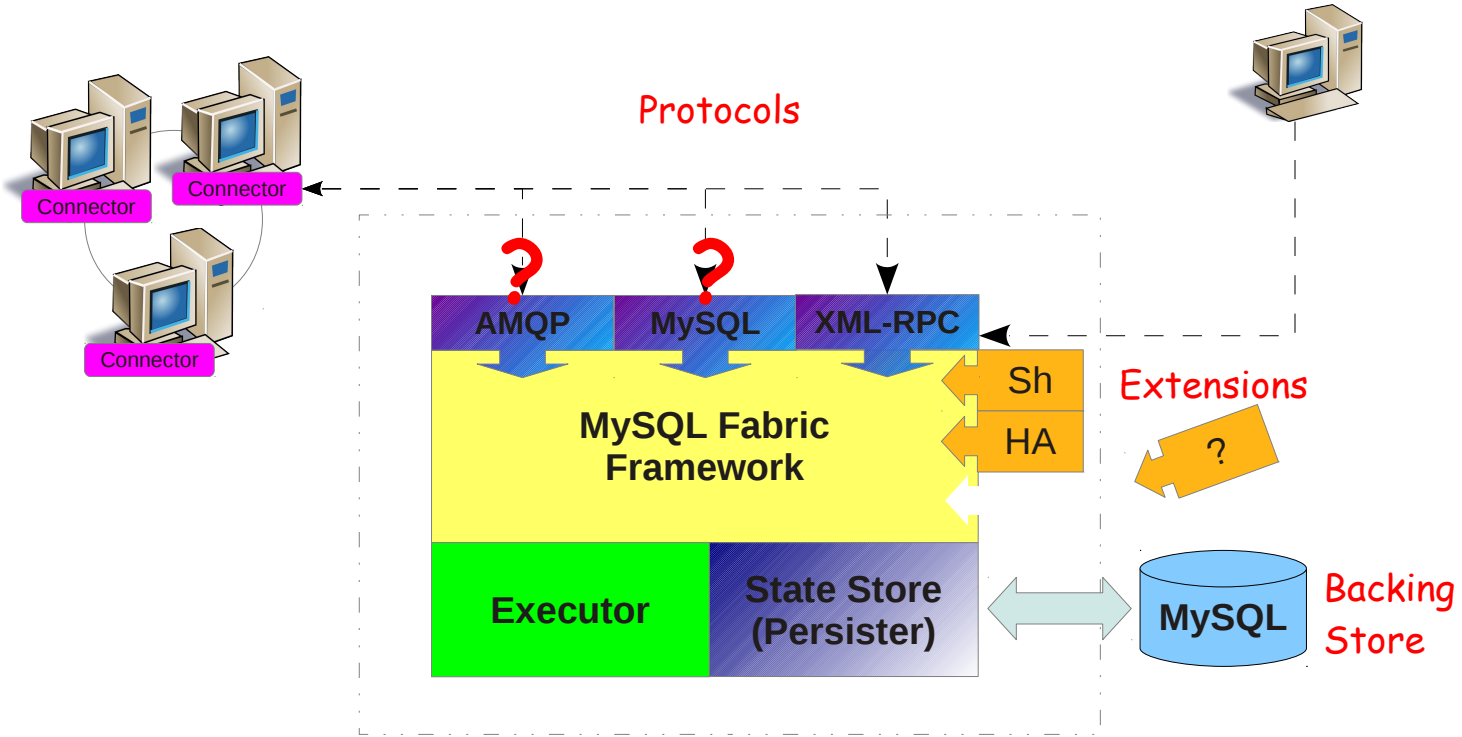
MySQL Fabric: Features

- Decision logic in connector
  - Reducing network load
- Support Transactions
  - API to provide sharding key
- Global Updates
  - Global Tables
  - Schema updates
- Procedure Executor
- Shard Multiple Tables
  - Using same key
- Sharding Functions
  - Range
  - (Consistent) Hash
- Shard Operations
  - Using built-in executor
  - Shard move
  - Shard split

Birds-eye View of a Sharded Database



MySQL Fabric Node Architecture



 |

MySQL Fabric: Prerequisites

- MySQL Servers (version 5.6.10 or later)
  - Server for meta-data backing store
  - Servers being managed
- Python 2.6 or 2.7
  - No support for 3.x yet
- MySQL Utilities 1.4.0
  - Available at http://labs.mysql.com/

MySQL Fabric: Configuration

- Backing Store
  - MySQL server
  - Persistent storage for state
  - Storage engine-agnostic
- Protocol
  - Address where node will be
  - Currently only XML-RPC
- Logging
  - Chatty: **INFO** (default)
  - Moderate: **WARNING**
  - URL for rotating log

```
[storage]
address = localhost:3306
user = fabric
password =
database = fabric
connection_timeout = 6

[protocol.xmlrpc]
address = localhost:8080
threads = 5

[logging]
level = INFO
url = file:///var/log/fabric.log
```


MySQL Fabric: Setup and Teardown

- Create the necessary tables in backing store

```
mysqlfabric manage setup
```

- Remove the tables from backing store

```
mysqlfabric manage teardown
```

- Connect to database server in “storage” section
  - Ensure that you have the necessary users and privileges

MySQL Fabric: Starting and Stopping

- Start MySQL Fabric node in foreground – print log to terminal

```
mysqlfabric manage start
```

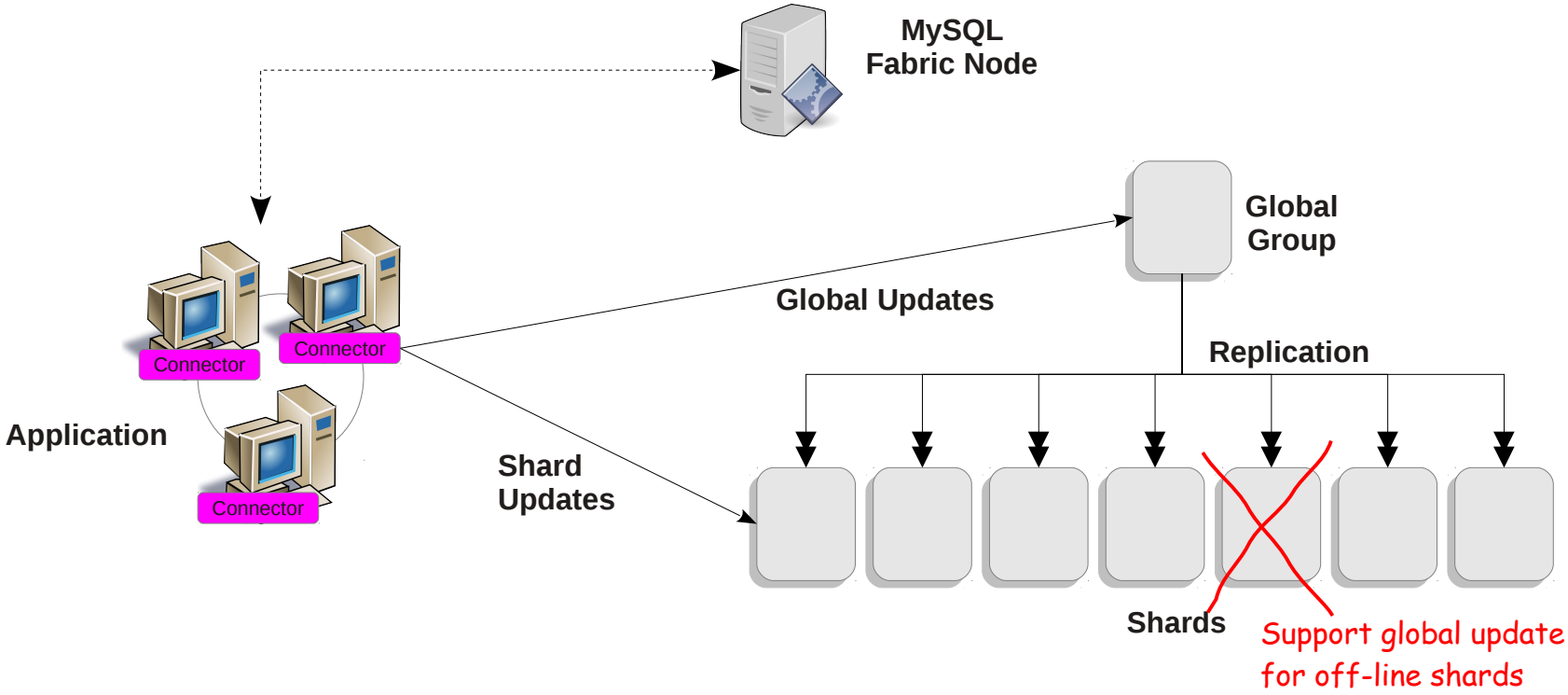
- Start MySQL Fabric node in background – print log to file

```
mysqlfabric manage start --daemonize
```

- Stop MySQL Fabric node

```
mysqlfabric manage stop
```

Sharding Architecture



 |

MySQL Fabric: Sharding Setup

- Set up some groups
  - `my_global` – for global updates
  - `my_group.*` – for the shards
  - Add servers to the groups
- Create a shard mapping
  - A “distributed database”
  - Mapping keys to shards
  - Give information on what tables are sharded
- Add shards

MySQL Fabric: Create Groups and add Servers

- Define a group

```
mysqlfabric group create my_global
```

- Add servers to group

```
mysqlfabric group add my_global global.example.com \
  mats xyzzy
mysqlfabric group add my_global …
```

User + Password
(Likely to go away)

MySQL Fabric: Create Groups and add Servers

- Promote one server to be primary

```
mysqlfabric group promote my_global
```

- Tell failure detector to monitor group

```
mysqlfabric group activate my_global
```

MySQL Fabric: Set up Shard Mapping

- Define shard mapping

```
mysqlfabric sharding define hash my_global
```

Will show the shard map identifier (a number)

- Add tables that should be sharded

```
mysqlfabric sharding add_mapping 1 \
  employees.employees emp_no
mysqlfabric sharding add_mapping 1 \
  employees.salaries emp_no
```

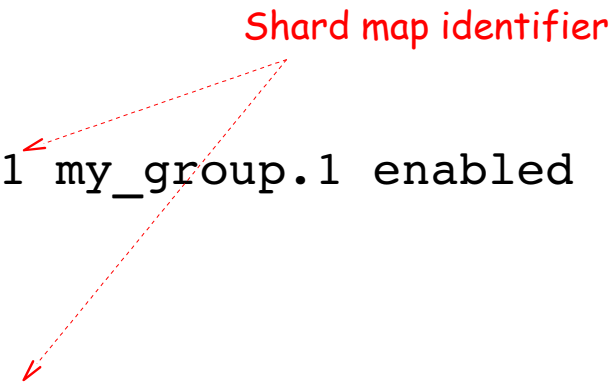
Shard map identifier

- Tables not added are global

MySQL Fabric: Add Shards

- Add shards to shard mapping

```
mysqlfabric sharding add_shard 1 my_group.1 enabled
    .
    .
    .
mysqlfabric sharding add_shard 1 my_group.N enabled
```



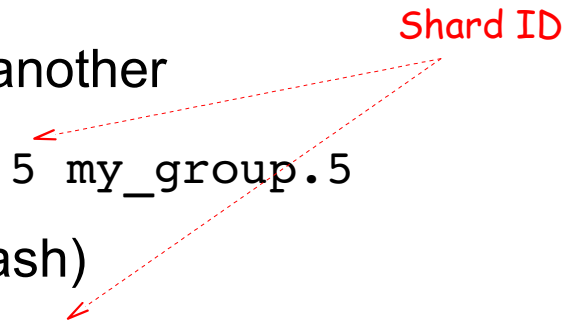
MySQL Fabric: Moving and Splitting Shards

- Moving a shard from one group to another

```
mysqlfabric sharding move 5 my_group.5
```

- Splitting a shard into two pieces (hash)

```
mysqlfabric sharding split 5 my_group.6
```



Working with a Sharded Database

Fabric-aware Connector API

- Support Transactions
  - Sharding key out of band

- Fabric-aware Connectors
  - Connector/J
  - Connector/Python
  - Connector/PHP
- Fabric-aware Frameworks
  - Doctrine
  - Hibernate
- Focus on Connector/Python

Fabric-aware Connector API

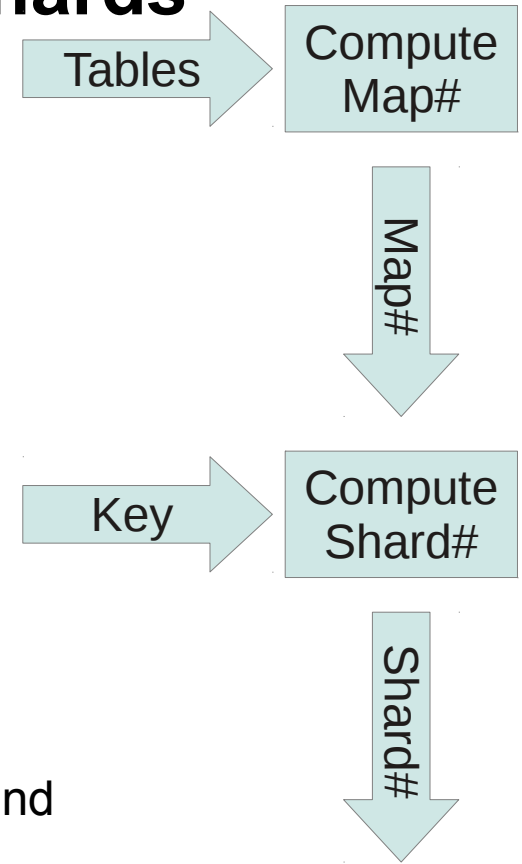
- Establish a “virtual” connection
  - Real server connection established lazily
- Provide connection information for the *Fabric node*
  - Fabric node will provide information about real servers

```
import mysql.connector.fabric as connector

conn = connector.MySQLFabricConnection(
    fabric={"host": "fabric.example.com", "port" : 8080},
    user='mats', database="employees")
```


Digression: Computing Shards

- Multiple Mappings
  - Which mapping to use?
  - Application don't care
    … but know tables in transaction
  - Currently only one mapping
- Computing shard requires
  - Tables + sharding key
- Extended Connector API
  - Extra properties passed out-of-band



Connector API: Shard Specific Query

- Provide tables in query
  - **Property:** `tables`
  - Fabric will compute map
- Provide sharding key
  - **Property:** `key`
  - Fabric will compute shard

```
conn.set_property(tables=['employees.employees', 'employees.titles'],
                  key=emp_no)
cur = conn.cursor()
cur.execute("INSERT INTO employees VALUES (%s,%s,%s)",
            (emp_no, first_name, last_name))
cur.execute("INSERT INTO titles(emp_no,title,from_date)"
            " VALUES (%s,%s,CURDATE())",
            (emp_no, 'Intern'));
conn.commit()
```

Transactions work fine!

Connector API: Shard Specific Query

- Provide tables in query
  - **Property:** `tables`
  - Fabric will compute map
- Provide sharding key
  - **Property:** `key`
  - Fabric will compute shard

```
conn.set_property(tables=['employees.employees', 'employees.titles'],
                  key=emp_no)
cur = conn.cursor()
cur.execute(
    "SELECT first_name, last_name, title"
    "  FROM employees JOIN titles USING (emp_no)"
    " WHERE emp_no = %d", (emp_no,))
for row in cur:
    print row[0], row[1], ",", row[2]
```

Join queries are sent to correct shard and executed there

Connector API: Global Update

- Provide tables in query
  - **Property:** `tables`
  - Fabric will compute map
- Set global scope
  - **Property:** `scope`
  - Query goes to global group

```
conn.set_property(tables=['employees.titles'], scope='GLOBAL')
cur = conn.cursor()
cur.execute("ALTER TABLE employees.titles ADD nickname VARCHAR(64)")
```


Closing Remarks

Thoughts for the Future

- Connector multi-cast
  - Scatter-gather
- Internal interfaces
  - Improve extension support
  - Improve procedures support
- Command-line interface
  - Improving usability
  - Focus on ease-of-use
- More protocols
  - MySQL-RPC Protocol?
  - AMQP?
- More frameworks?
- More HA group types
  - DRBD
  - MySQL Cluster
- Fabric-unaware connectors?

Thoughts for the Future

- "More transparent" sharding
  - Single-query transactions
  - Cross-shard joins is a problem
- Multiple shard mappings
  - Independent tables
- Multi-way shard split
  - Efficient initial sharding
  - Better use of resources
- High-availability executor
  - Node failure stop execution
  - Replicated State Machine
  - Fail over to other Fabric node
- Distributed failure detector
  - Connectors report failures
  - Custom failure detectors

MySQL Connect Sessions

- MySQL High Availability: Managing Farms of Distributed Servers
  - September 22, 5:30pm-6:30pm in Imperial Ballroom B
- Scaling PHP Applications
  - September 22, 10:00am-11:00am in Union Square Room ¾
- MySQL Sharding, Replication, and HA
  - September 21, 5:30-6:30pm in Imperial Ballroom B

References

- MySQL Forum: *Fabric, Sharding, HA, Utilities*
  - http://forums.mysql.com/list.php?144
- A Brief Introduction to MySQL Fabric
  - http://mysqlmusings.blogspot.com/2013/09/brief-introduction-to-mysql-fabric.html
- MySQL Fabric – Sharding – Introduction
  - http://vnwrites.blogspot.com/2013/09/mysqlfabric-sharding-introduction.html
- Migrating From an Unsharded to a Sharded Setup
  - http://vnwrites.blogspot.com/2013/09/mysqlfabric-sharding-migration.html

Keeping in Touch

Mats Kindahl
- **Twitter:** @mkindahl
- **Blog:** http://mysqlmusings.blogspot.com

Alfranio Correia
- **Twitter:** @alfranio
- **Blog:** http://alfranio-distributed.blogspot.com

Narayanan Venkateswaran
- **Twitter:** @vn_tweets
- **Blog:** http://vnwrites.blogspot.com

Thank you!