
PlanetScale

Vitess for Newbies: Scaling MySQL the Youtube way

Igor Donchovski

Enterprise Customer Engineer, PlanetScale

About me

👷 Enterprise Support Engineer - Planetscale

🎓 Education: MSc in Software Engineering

📜 Certifications

- MongoDB Certified DBA
- Oracle Professional MySQL 5.7
- Terraform Associate Certified
- GCP Professional Architect

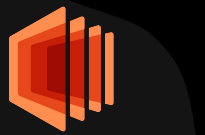
💻 Expertise: 20+ years in IT, 15+ years with MySQL, 10+ years MongoDB, few years with PostgreSQL

🌟 Personal: Husband and Father, Avid Traveler, Speaker

🧩 Playing Linkedin puzzle games on daily basis

Agenda

- MySQL Limits
- Online DDL
- Query Management
- High Availability
- Disaster Recovery
- Vertical Scaling
- Horizontal Scaling
- Vitess



MySQL - Limits

Physical Limits (32TB RAM, 144+ cores per socket (8 sockets CPU)

Tablespace size (64TB with 16k page size)

Database size (no actual limit)

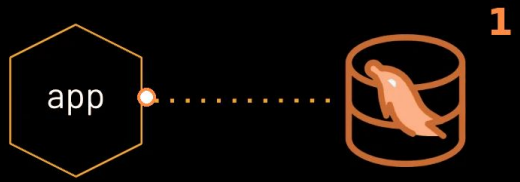
Number of rows (18 quintillion for bigint)

Max connections (100k)

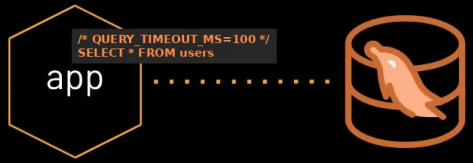
Max Threads Running (< 100k)

MySQL - Query Management

Query Timeout



Comment directives

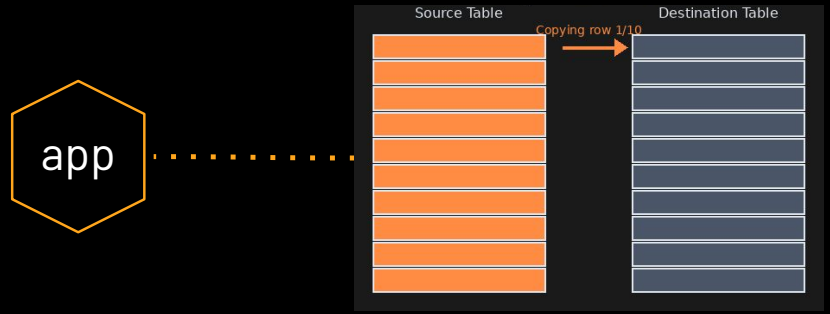


Limit rows returned

Query Consolidation

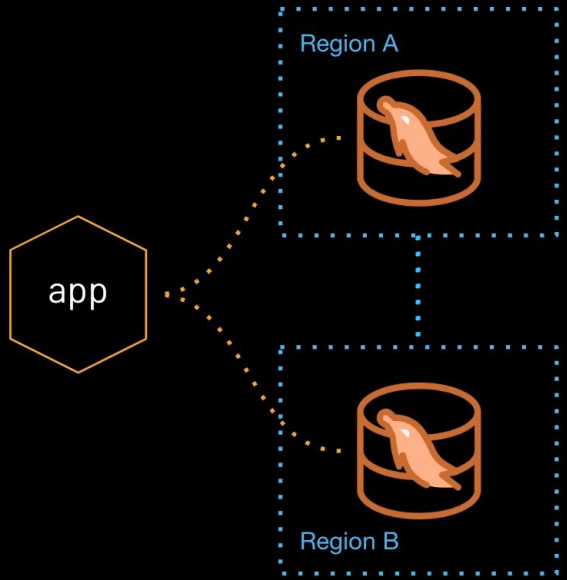
MySQL - DDL

- MySQL instant ddl
- pt-online-schema-change
- gh-ost
- spirit
- Resuming on failures
- Reverting the change
- Resource Greedy



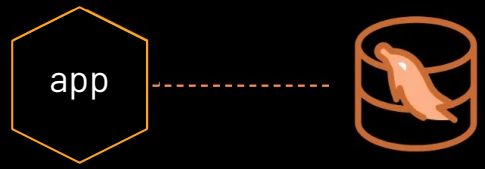
MySQL - HA

- Zero downtime migration
- Move/Copy a subset of the tables
- Checksum the data
- Switch reads from A to B
- Switch writes from A to B
- Revert reads/writes from B to A



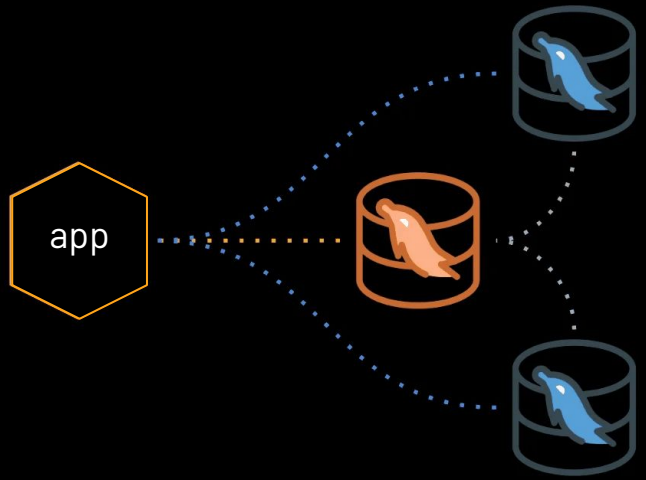
MySQL - HA

- Only use the Primary
- Always read from a Replica
- Skip reading from replica if lag > $X
- Min serving replicas
- Disaster recovery (A,B,C)



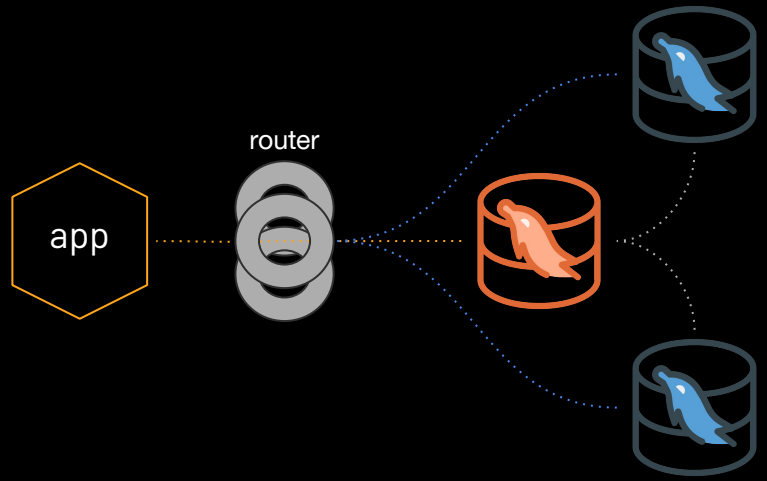
MySQL - HA

- Primary failure
- Lost transactions
- Replica failure
- App errors
- Replica clone and warm up

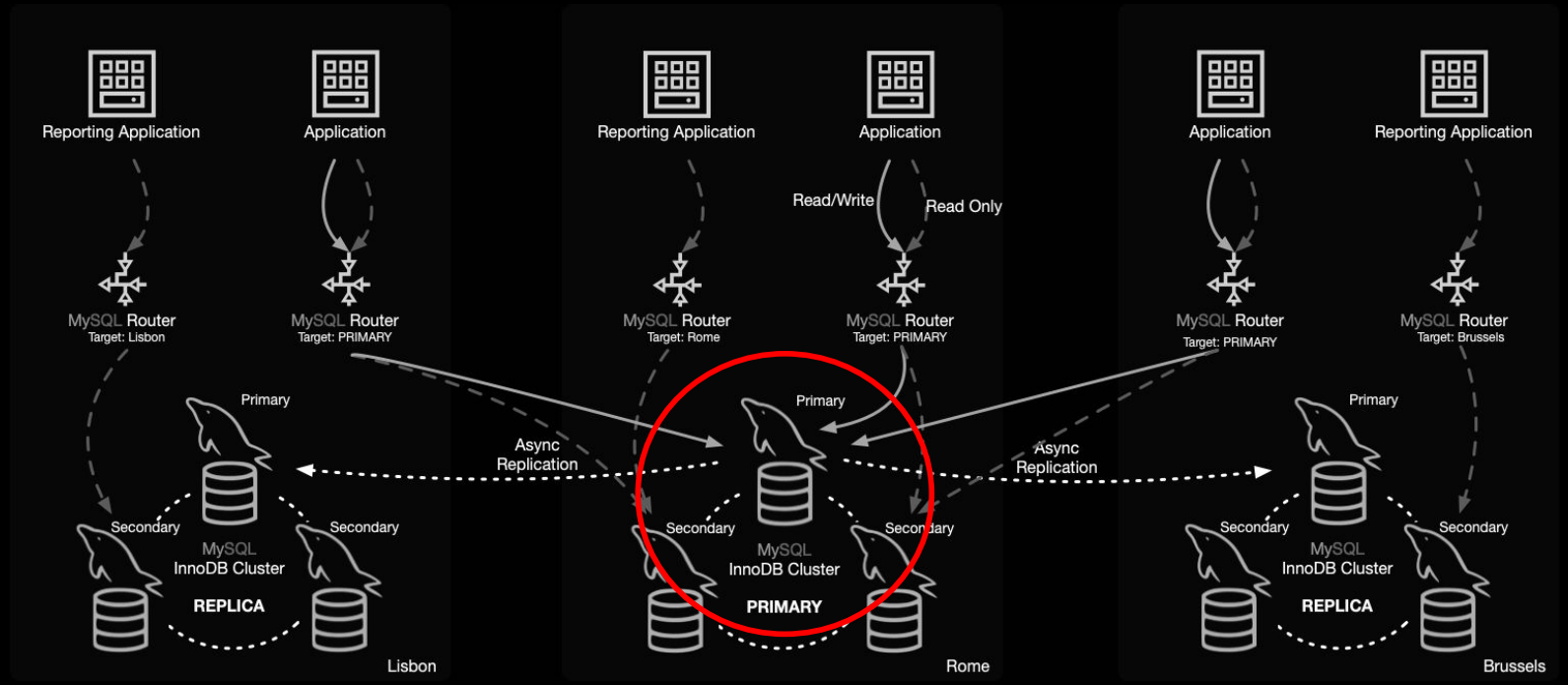


MySQL - HA

- Remove the complexity from the app
- Router as a middle layer
- Semi sync replication
- Synchronous replication
- Everything as HA

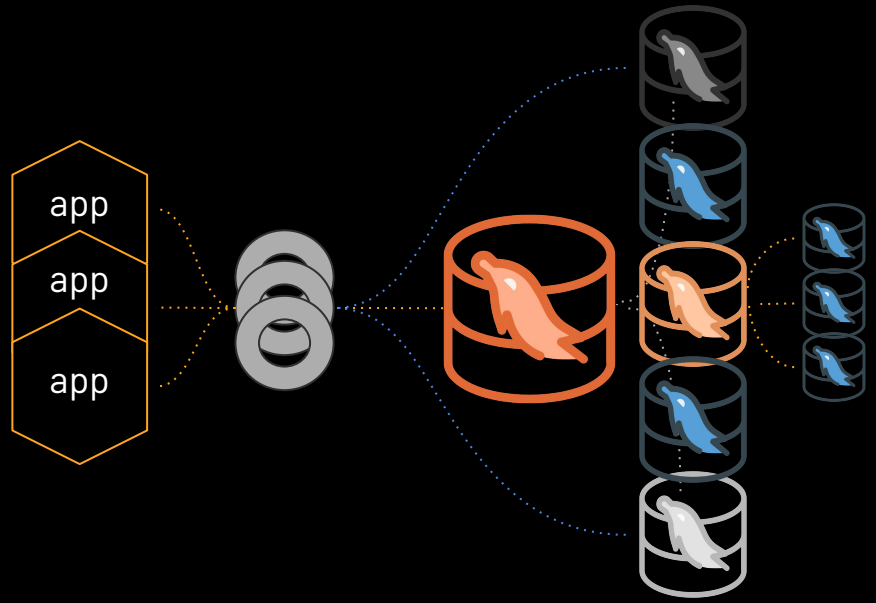


MySQL - HA and DR



MySQL - Vertical Scaling

- Increased workload
- Reporting replicas
- Dedicated backup replica
- Delayed replica
- Intermediary primary/replica



MySQL - Vertical Scaling

Gym Tracker App

users — 200,000 rows

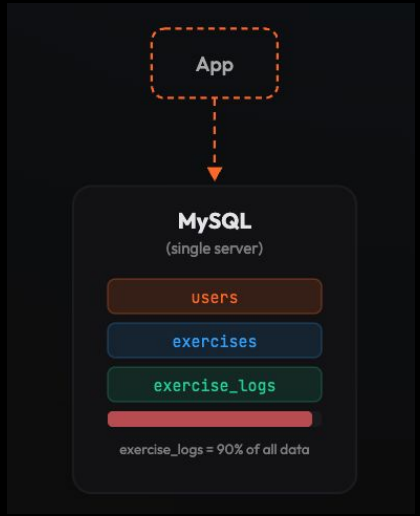
COLUMN	TYPE	
id	BIGINT	PK
username	VARCHAR(100)	
email	VARCHAR(255)	
created_at	DATETIME	

exercises — 200 rows

COLUMN	TYPE	
id	BIGINT	PK
name	VARCHAR(200)	
muscle_group	VARCHAR(100)	

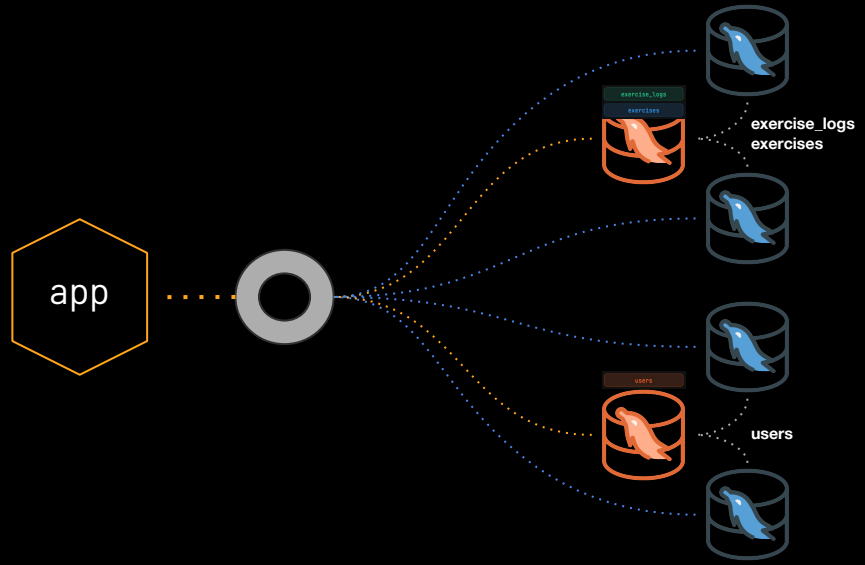
exercise_logs — 50,000,000+ rows

COLUMN	TYPE	
id	BIGINT	PK, auto_inc
user_id	BIGINT UNSIGNED	FK → users
exercise_id	BIGINT UNSIGNED	FK → exercises



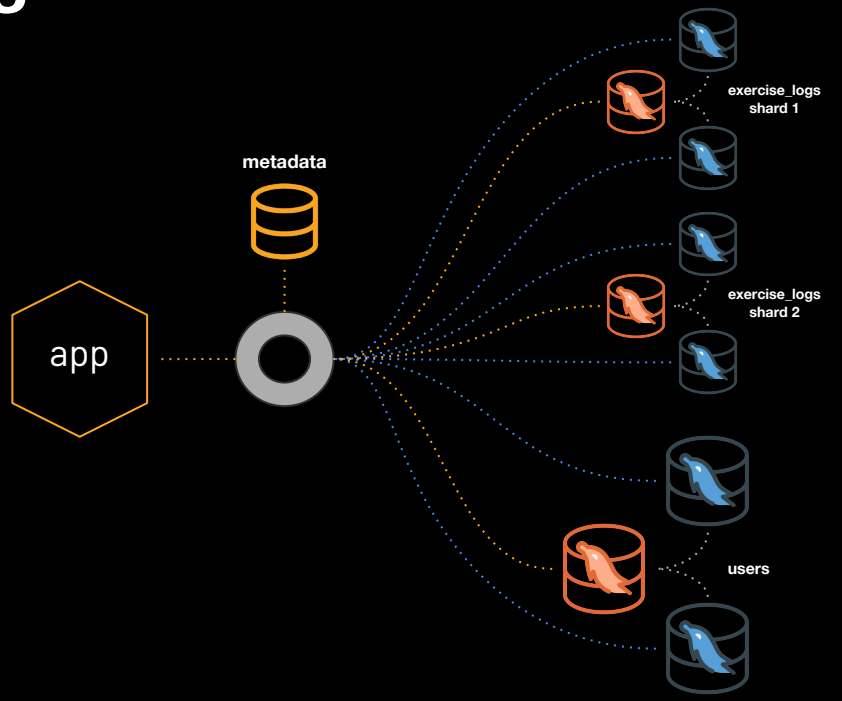
MySQL - Vertical Scaling

- Divide and conquer
- Scale to largest machines
- Scale read capacity
- Normalization
- Referential integrity



MySQL - Horizontal Scaling

- App logic where to send the query
- Primary failure
- Limiting the result sets
- Resharding
- Updating the metadata
- Restoring a node



MySQL - Horizontal Scaling

users

ID	USERNAME
1	alice
7	grace

exercises REF

All 200 rows (full copy)

exercise_logs

ID	USER_ID	EX_ID	SETS	REPS
1001	1	5	3	10
1002	1	12	4	8
1008	7	3	3	12

users

ID	USERNAME
2	bob
5	eve

exercises REF

All 200 rows (full copy)

exercise_logs

ID	USER_ID	EX_ID	SETS	REPS
1003	2	1	5	5
1004	2	8	3	10
1009	5	15	4	6

users

ID	USERNAME
3	carol
8	heidi

exercises REF

All 200 rows (full copy)

exercise_logs

ID	USER_ID	EX_ID	SETS	REPS
1005	3	5	4	10
1010	8	1	3	15
1011	8	12	5	5

users

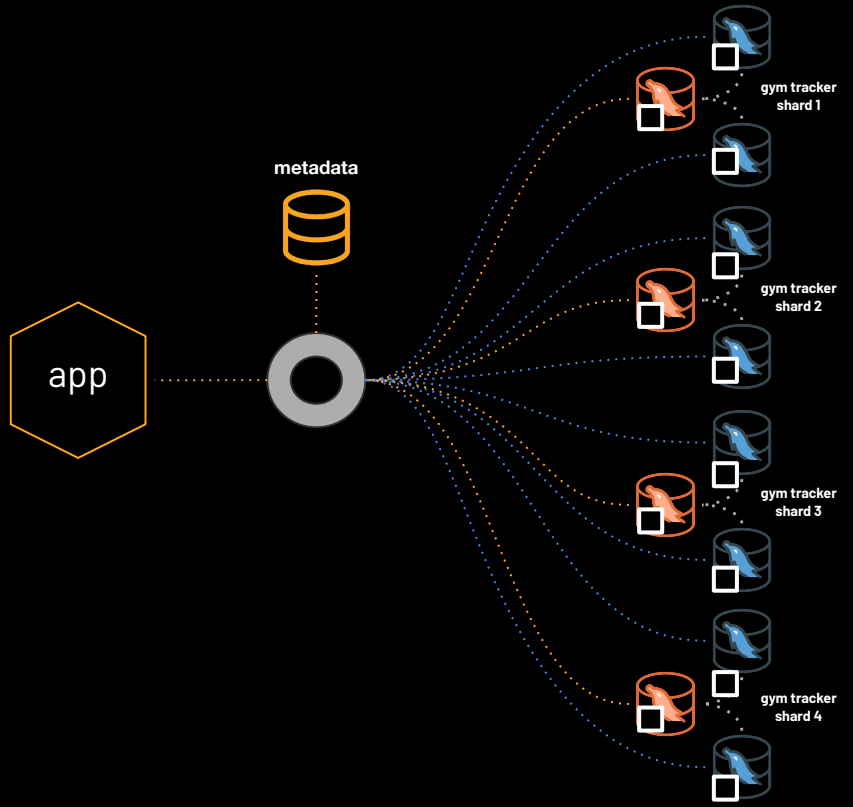
ID	USERNAME
4	diana
6	frank

exercises REF

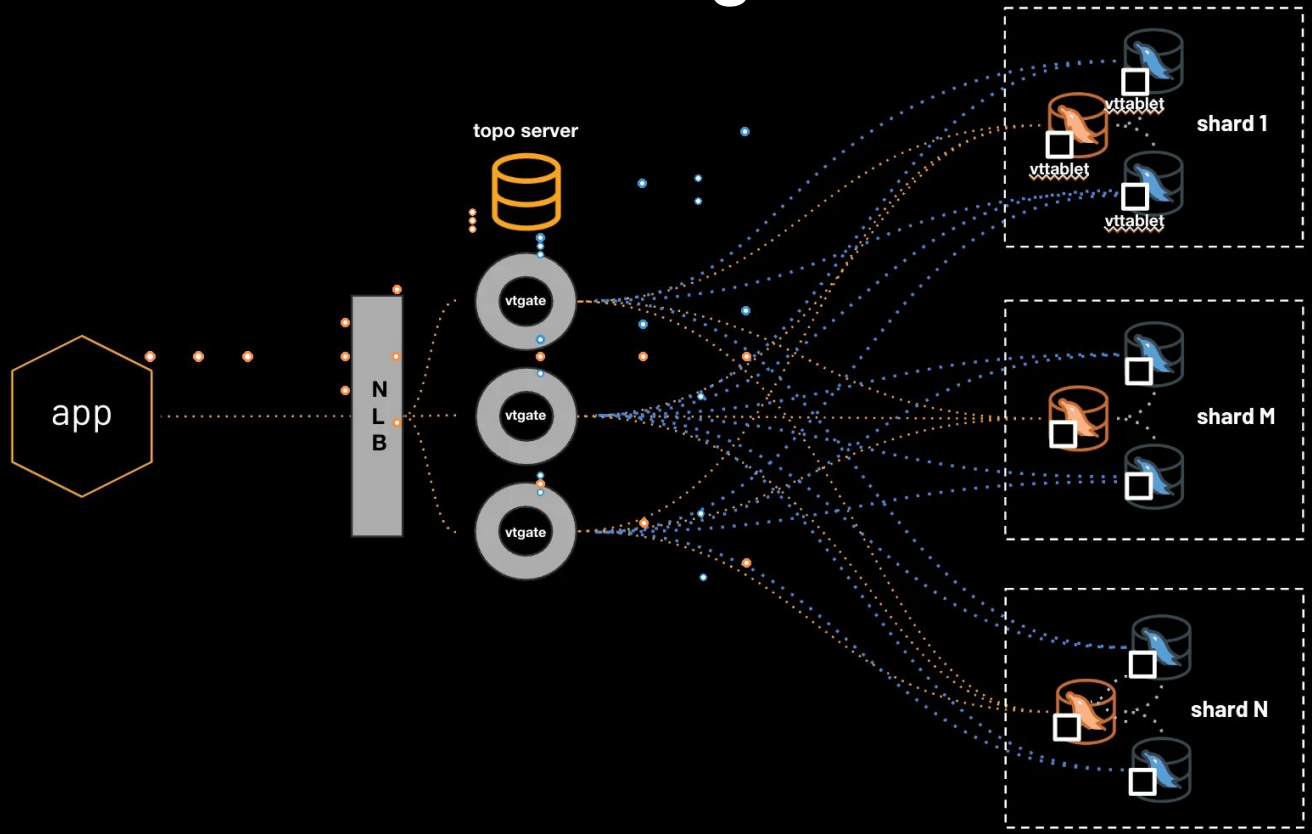
All 200 rows (full copy)

exercise_logs

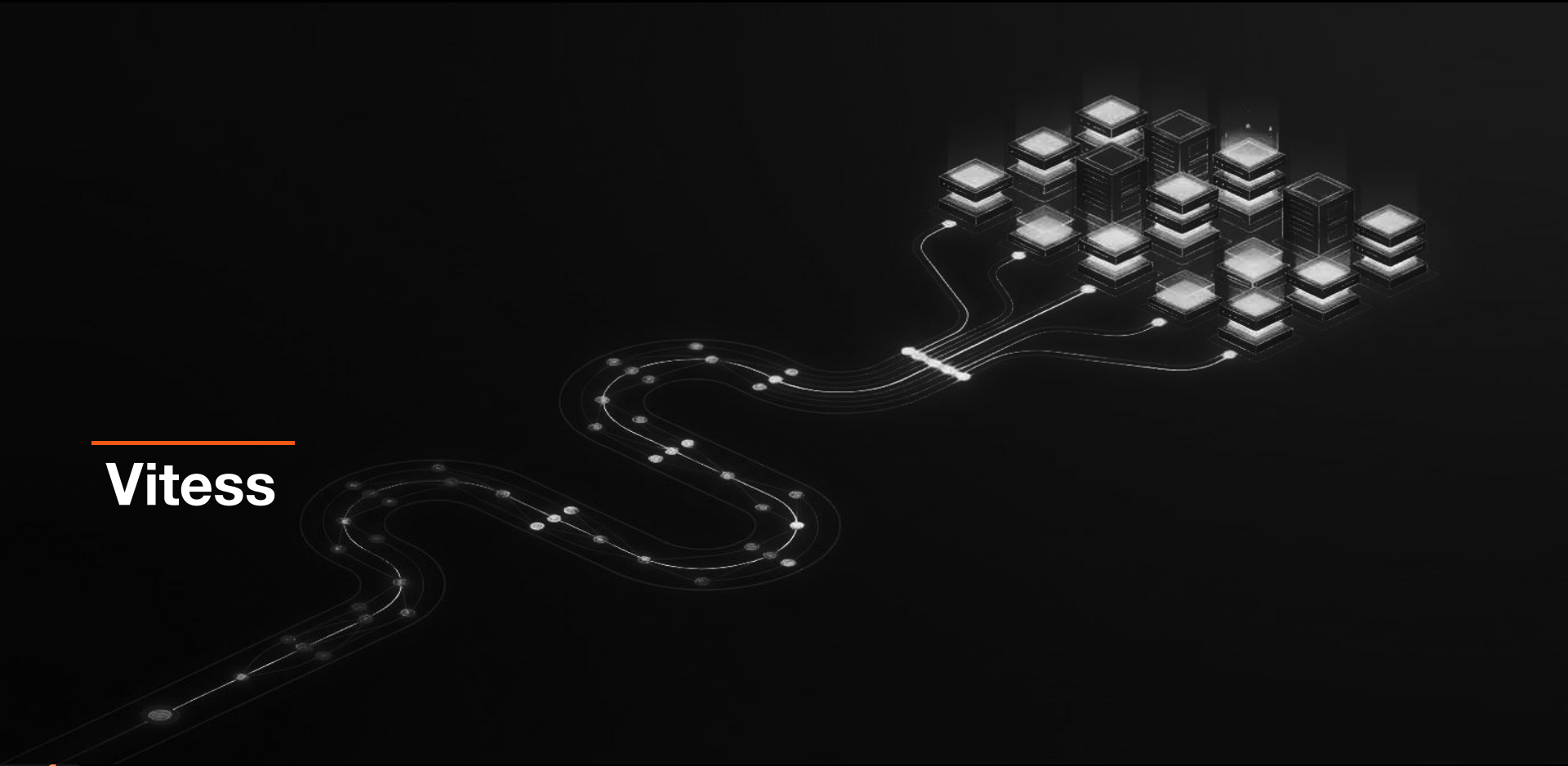
ID	USER_ID	EX_ID	SETS	REPS
1006	4	3	4	12
1007	4	8	3	8
1012	6	5	5	10



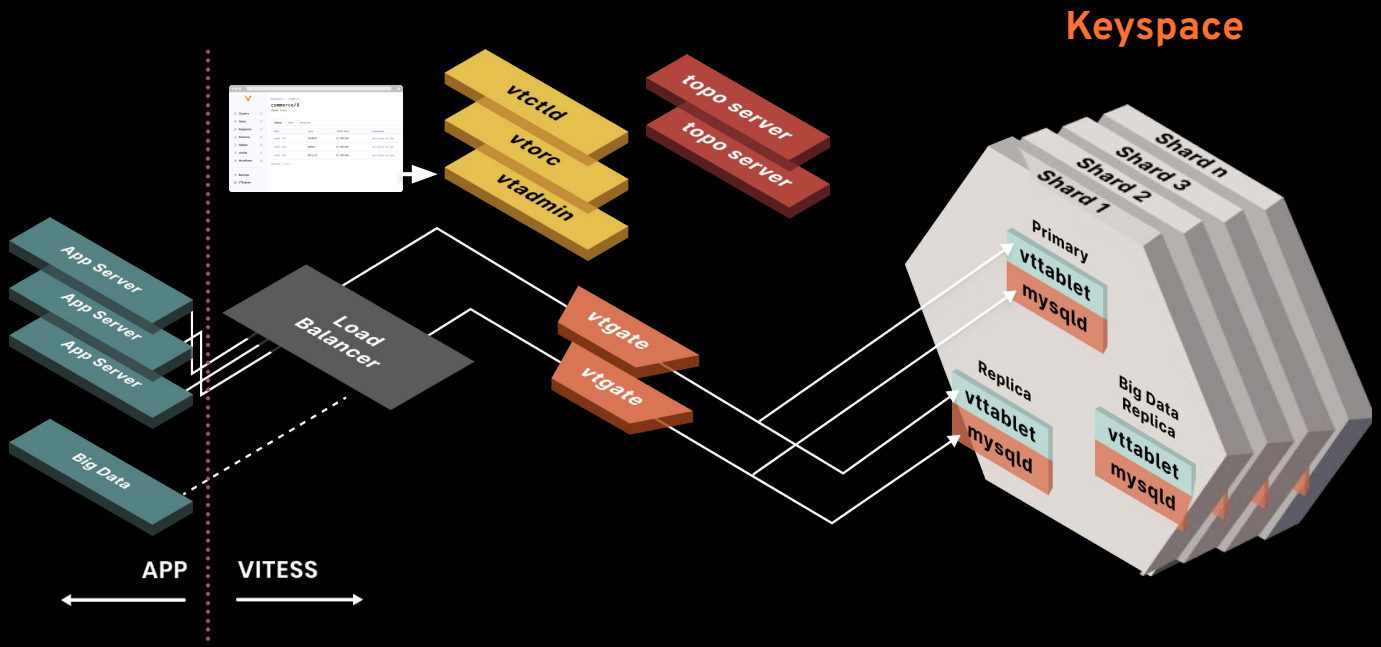
MySQL - Horizontal Scaling



Vitess



Vitess Architecture



Vitess serves millions of QPS in production

slack
New Relic
Square
Flipkart
HubSpot
peak
Pinterest
shopify
nozzle
weave
GitHub
JD.COM 京东
Quiz of Kings
stitchlabs
PlanetScale

Concepts

Keyspace

Shard

VSchema

VIndex

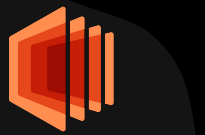
Vitess needs to deal with

Database Frameworks

ORMs

Legacy Code

Third Party Apps

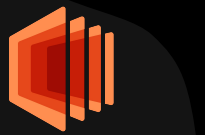


Vitess provides an illusion

A single database (server)

MySQL 8.0 and above

A single, dedicated connection



VReplication

Use cases

- Migration
  - Import data into Vitess
- (Re)sharding
  - Move data around
- Materialisation
  - Improve query performance
  - Materialise unsharded data on each shard

How it works?

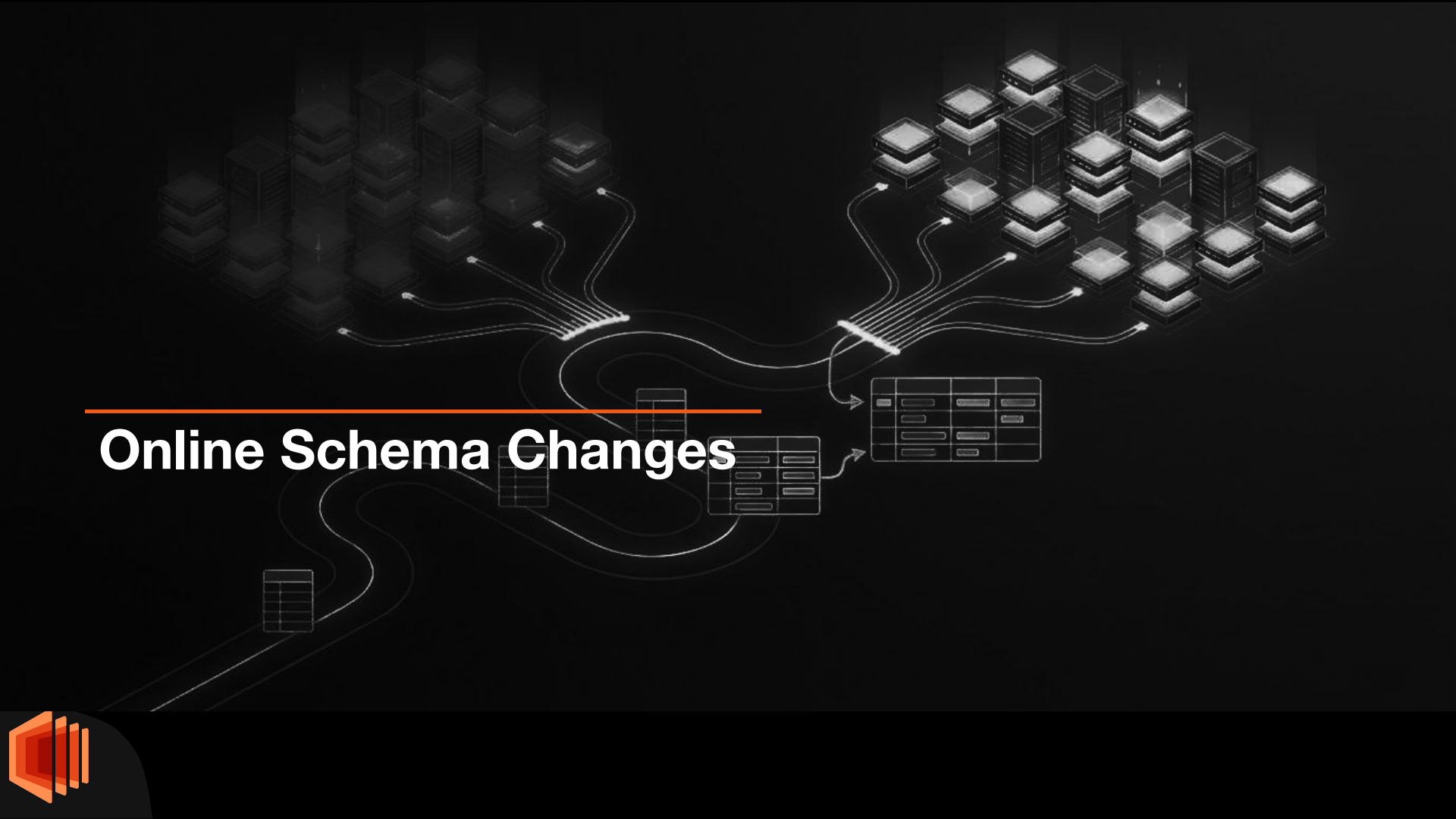
- Copy phase: `SELECT * FROM table [WHERE id > ?]`
  - Streams rows in PK order
  - Keeps track of last copied PK
  - Default duration 1h (configurable)
- Replication phase
  - Catch up on binary logs
  - Keeps track of replication position (GTID preferred)

Import workflow

- Create an “external” keyspace with a vttablet pointing to an existing MySQL server.
- Create a Vitess keyspace
- Use the MoveTables command to create a VReplication Workflow to move data into Vitess
- Confirm that all data was migrated (VDiff)
- Switch read traffic to go to Vitess
- Switch write traffic to go to Vitess
- Full support for rollback in case of issues

VReplication demo

https://vitess-import-flow.vercel.app/



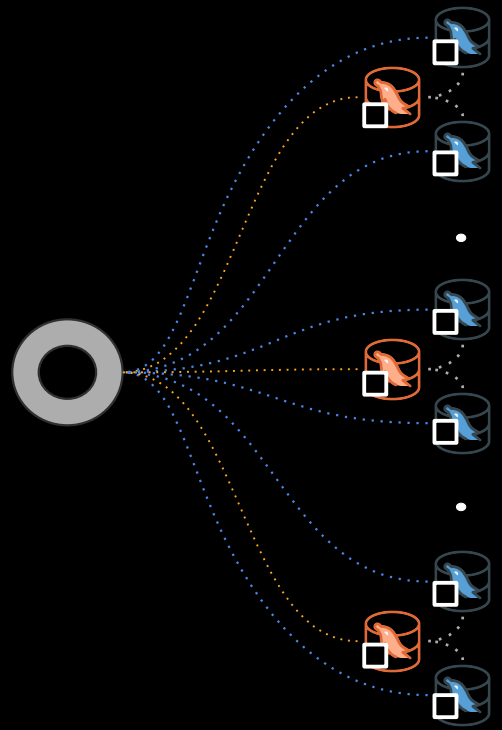
Online Schema Changes

Based on VReplication

```
mysql> SET @@ddl_strategy = 'online';
Query OK, 0 rows affected (0.00 sec)

mysql> ALTER TABLE table1 CHANGE id id bigint unsigned;
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| 4d8246f2_9801_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.02 sec)
```

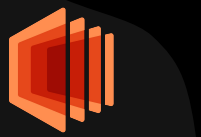
Grab a coffee!



Based on VReplication

```
mysql> SHOW vitess_migrations LIKE '4d8246f2_9801_11ed_a6ae_c87f5403e983'\G

*************************** 1. row ***************************
...
     migration_uuid: 4d8246f2_9801_11ed_a6ae_c87f5403e983
    added_timestamp: 2023-01-19 14:58:08
completed_timestamp: 2023-01-19 14:58:18
   migration_status: complete
migration_statement: alter table table1 change column id id bigint unsigned
           strategy: online
...
1 row in set (0.00 sec)
```



But there is more…

- Migrations become fully reversible
- Because the VReplication stream could easily be reversed, the old table is kept up-to-date.
- Now a revert operation is as simple as

```
mysql> REVERT vitess_migration '4d8246f2_9801_11ed_a6ae_c87f5403e983';
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| 20f5337f_9826_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.04 sec)
```

Declarative migrations

No need to write **ALTER** statements anymore, just resubmit the **CREATE TABLE** statement and Vitess will figure out the difference and run the migration…

```
mysql> SET @@ddl_strategy = 'online -declarative';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE TABLE `table1` (
       `id` bigint unsigned NOT NULL,
       `data` varchar(512) DEFAULT NULL,
       PRIMARY KEY (`id`)
     ) ENGINE=InnoDB;
+--------------------------------------+
| uuid                                 |
+--------------------------------------+
| c423f39b_982c_11ed_a6ae_c87f5403e983 |
+--------------------------------------+
1 row in set (0.02 sec)
```

Conclusion

- Schema changes are being put back into the hands of the developers!
- Easy to run
- Easy to revert
- Due to the robustness of VReplication, it can even survive a primary failover or other type of outage

Summary

MySQL Has Limits

Physical, operational, and scaling ceilings that every growing application eventually hits

Horizontal Scaling

Shard transparently, apps see one database while Vitess manages many underneath

Online DDL Made Easy

Vitess Online Schema Changes with VReplication reversible, resumable, across all shards

VReplication Powers It All

Migration, resharding, and materialisation through a unified replication framework

Built-in HA & DR

Automated failover, topology-aware routing, and multi-region disaster recovery

Zero App Changes

Vitess provides the illusion of a single MySQL server, works with ORMs, frameworks, legacy code


Thank you!