ORACLE

Secure Your Data Workshop

MySQL Replication for High Availability

Dale Dasker
Manager MySQL Solution Engineering
dale.dasker@oracle.com
September 29, 2022

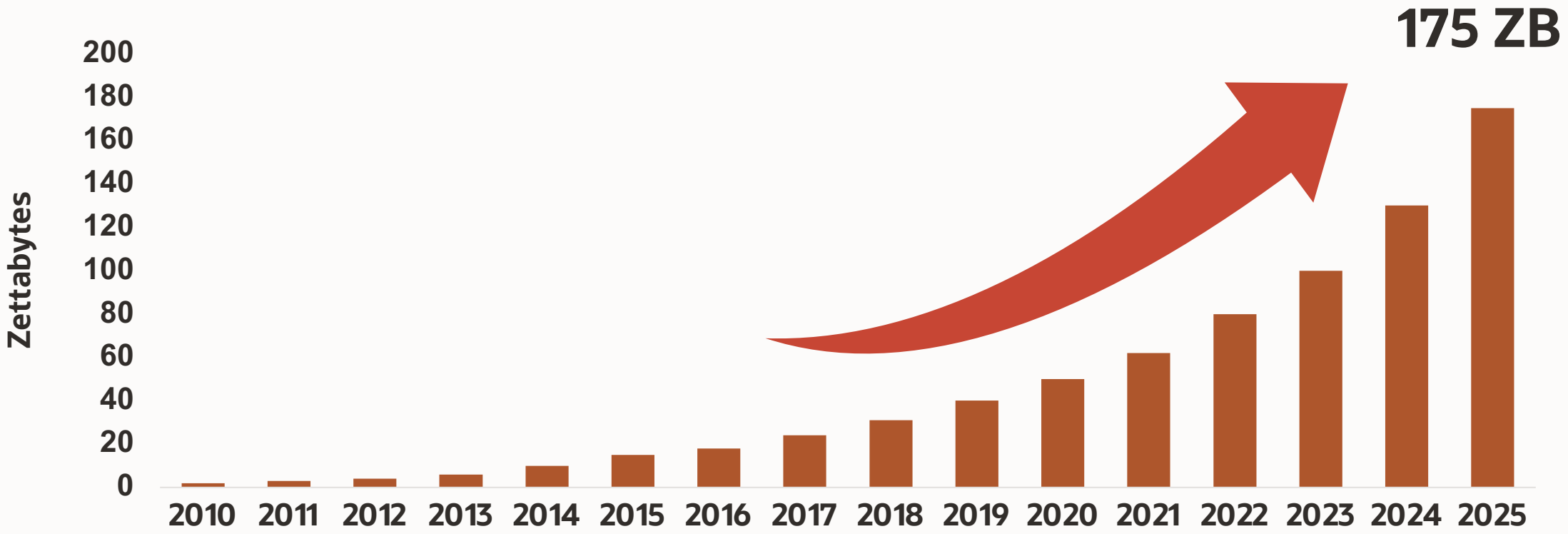
Agenda

Achieve High Availability with MySQL Replication Protocols

- MySQL Replication Overview
- Workshop Overview
- Setup and Installation of:
  - MySQL ReplicaSets
  - MySQL InnoDB Clusters
  - MySQL InnoDB ClusterSets

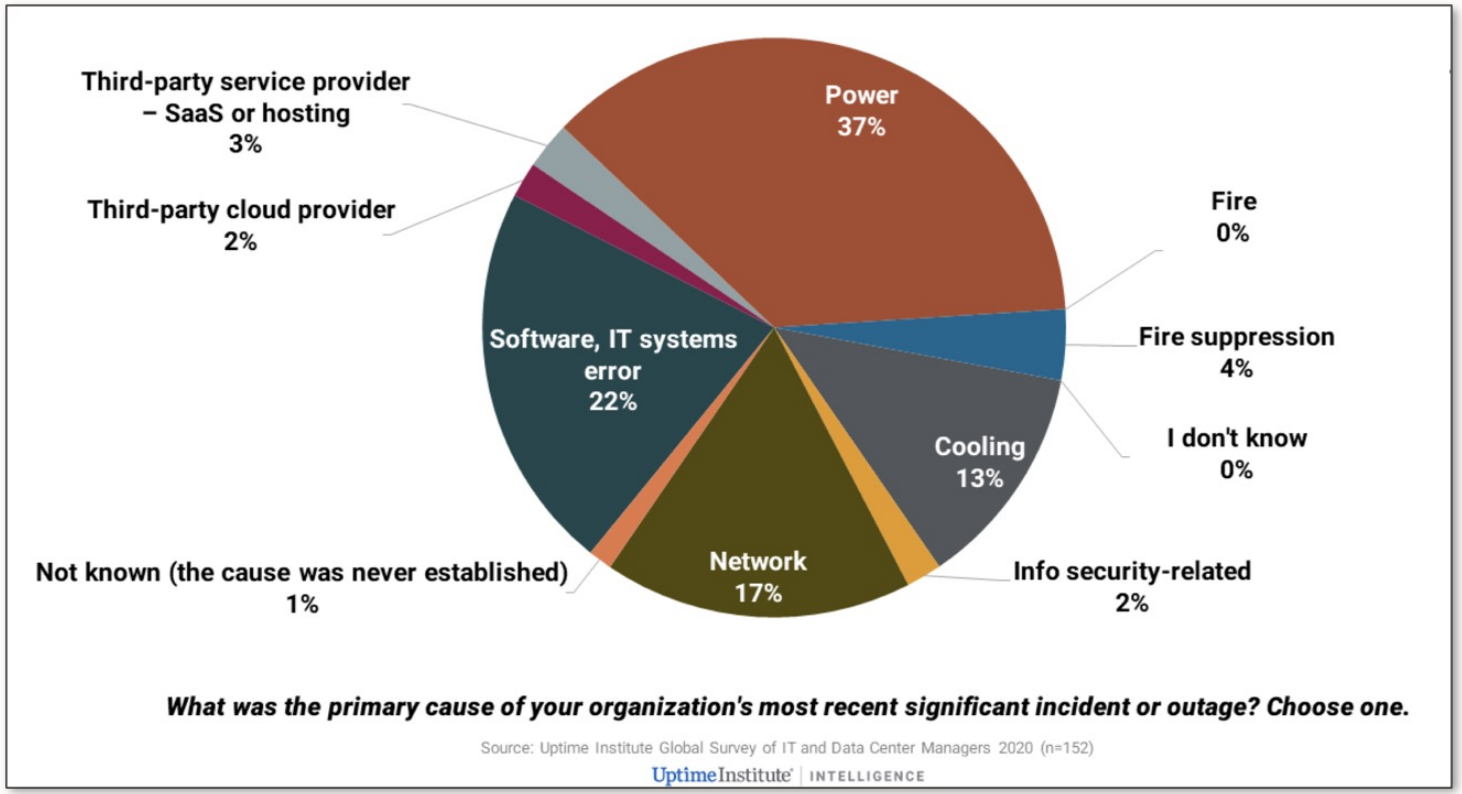
Why?

Global Datasphere



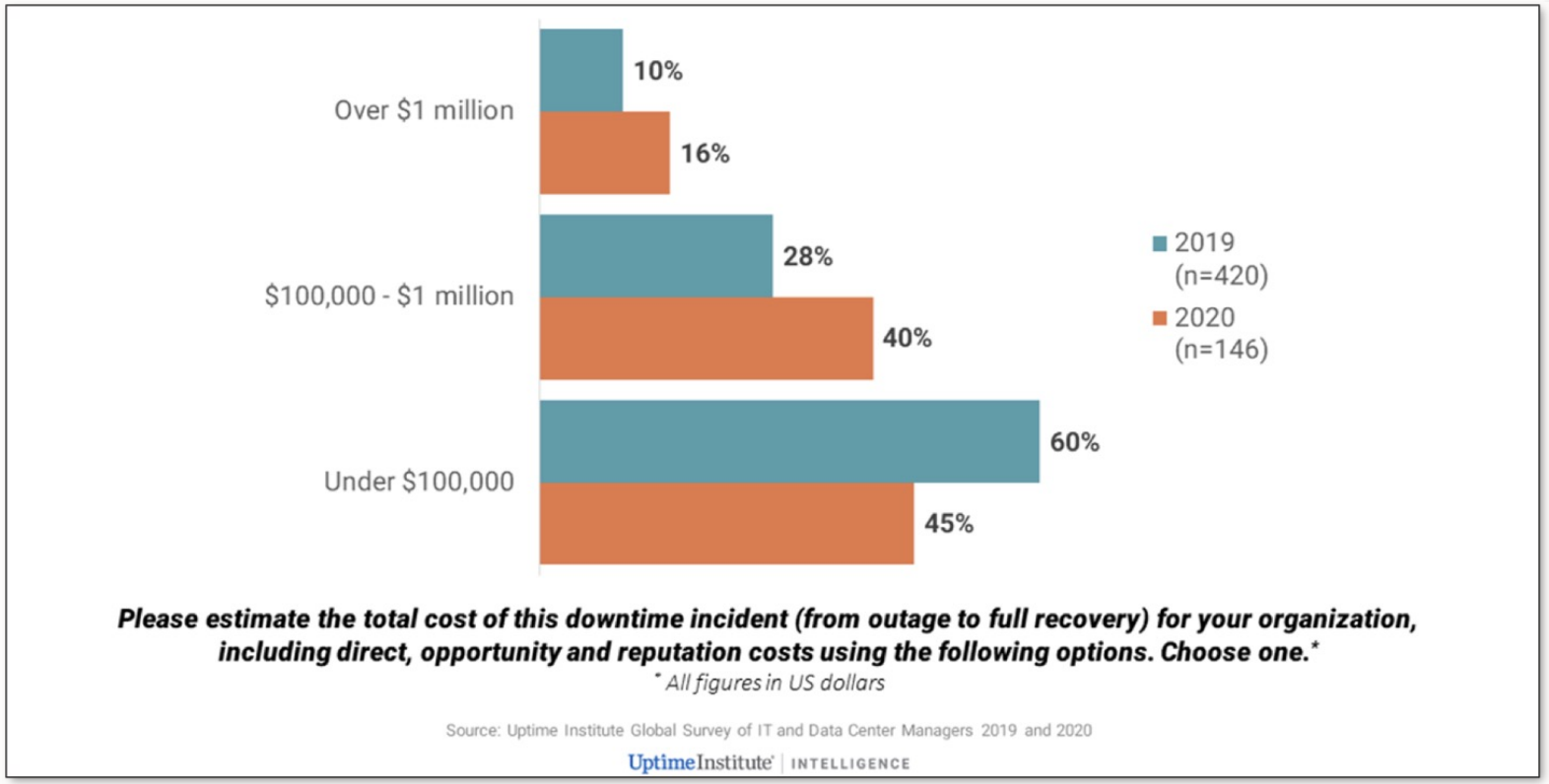
Data: Your Most Valuable Asset

IT Disasters & Outages: Primary Causes



On-site power failure is the biggest cause of signiicant outages

IT Disasters & Outages: Costs are Rising



Over half who had experienced an outage costing more than $100,000.

IT Disasters and Outages: Examples

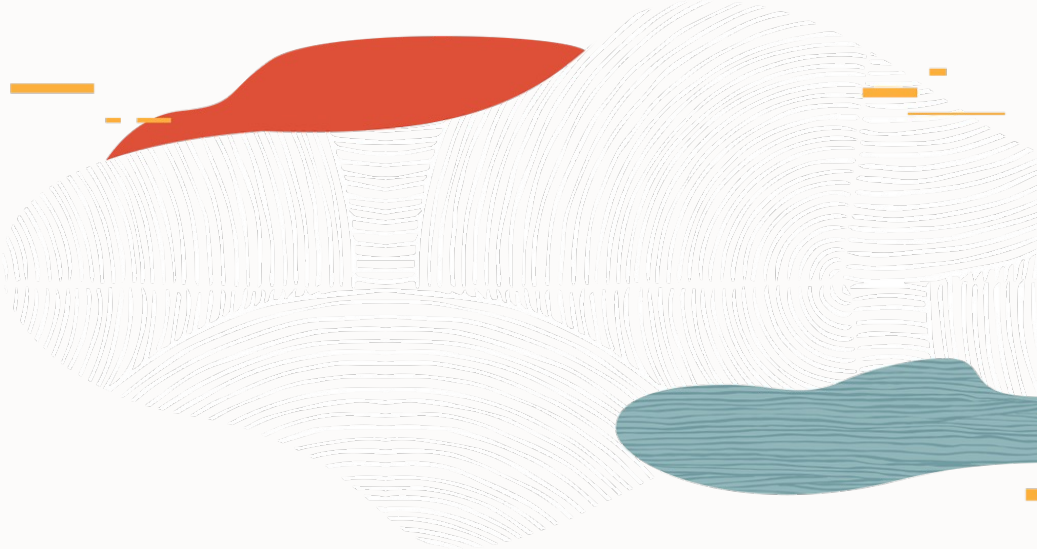
5-hour computer outage cost us $150 million. The airline eventually canceled about 1,000 flights on the day of the outage and ground an additional 1,000 flights over the following two days.

Millions of websites offline after ire at French cloud services irm. The ire is expected to cost the company more than €105 million.

Tens of thousands of passengers were stranded in cities around the world due to cancellation of about 130 flights and the delay of 200.

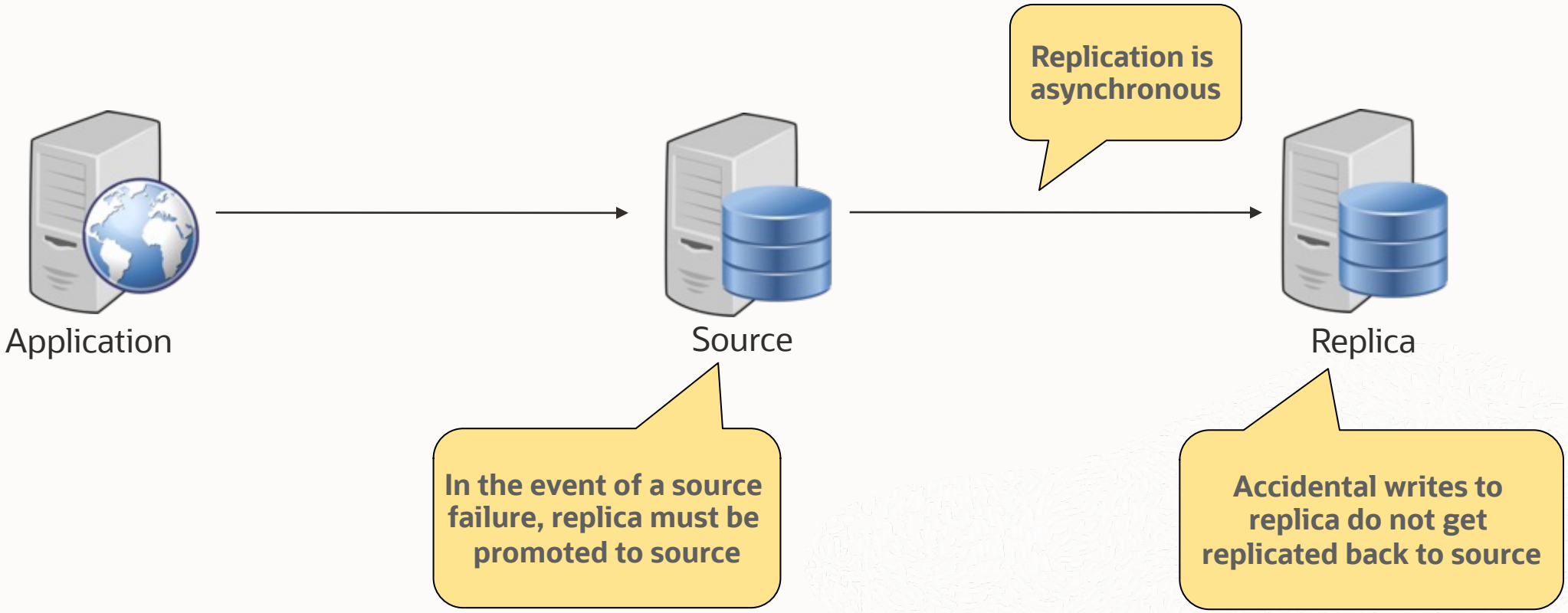
Millions of bank customers were unable to access online accounts. The bank took almost 2 days to recover and get back to normal functioning.

ORACLE



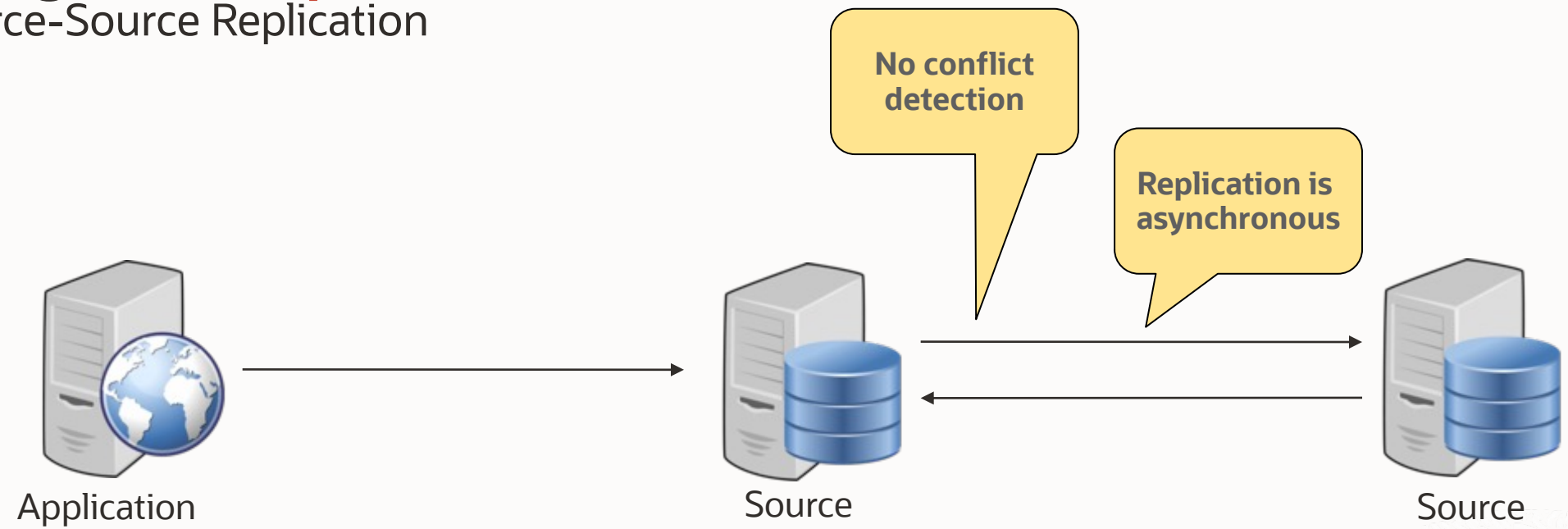
MySQL Replication Overview

Background: Popular Solutions
Source-Replica Replication



Background: Popular Solutions

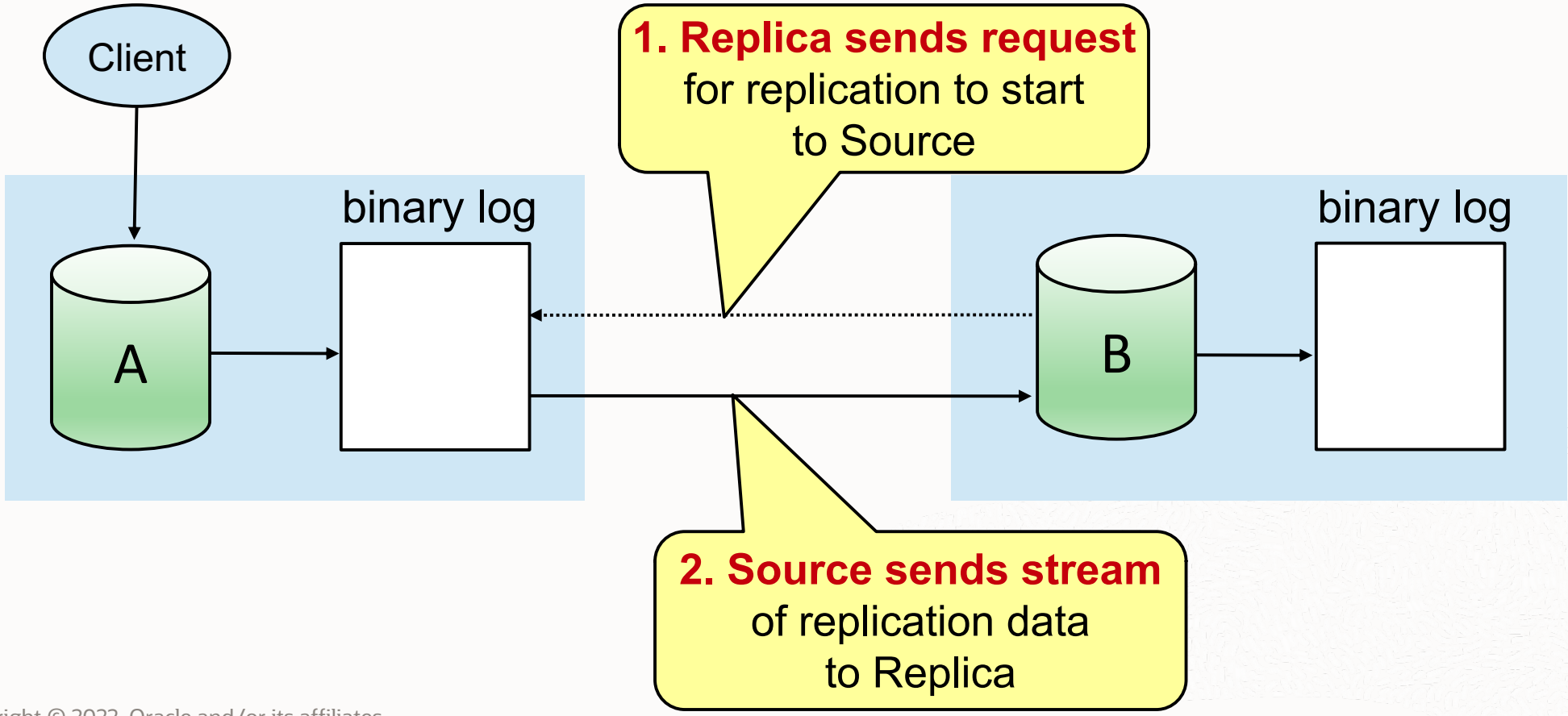
Source-Source Replication



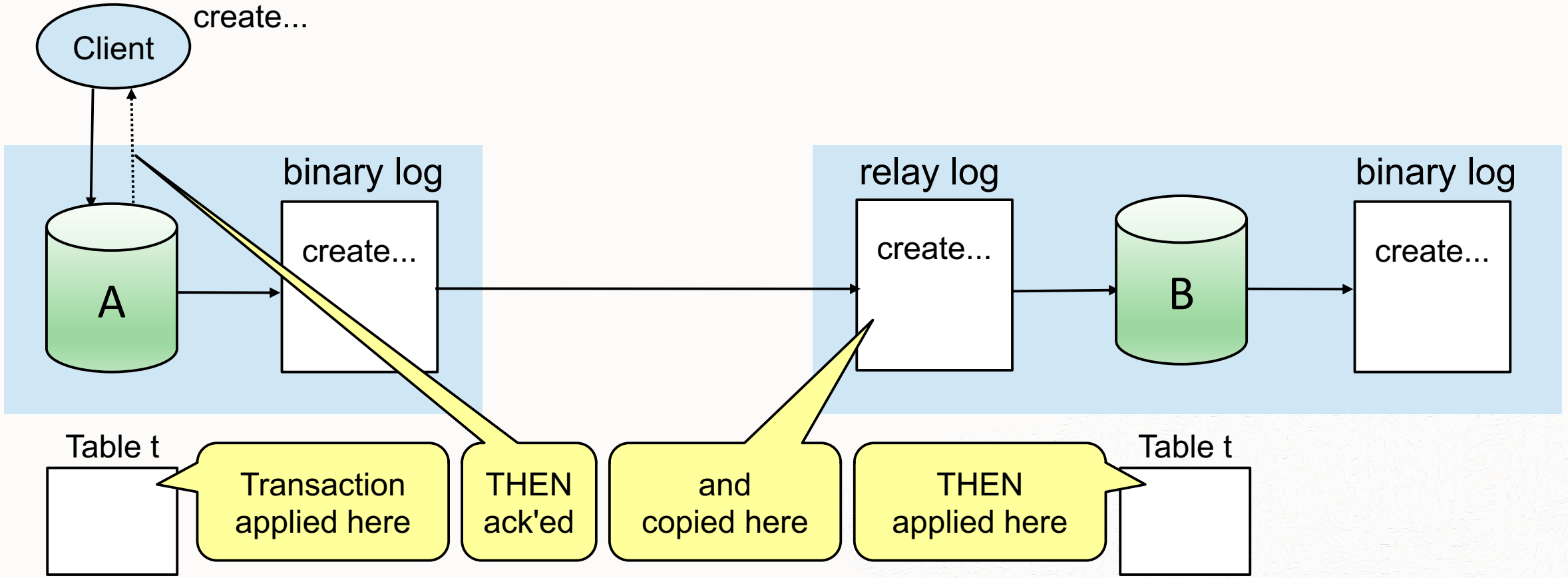
ORACLE

How Does Replication Work?

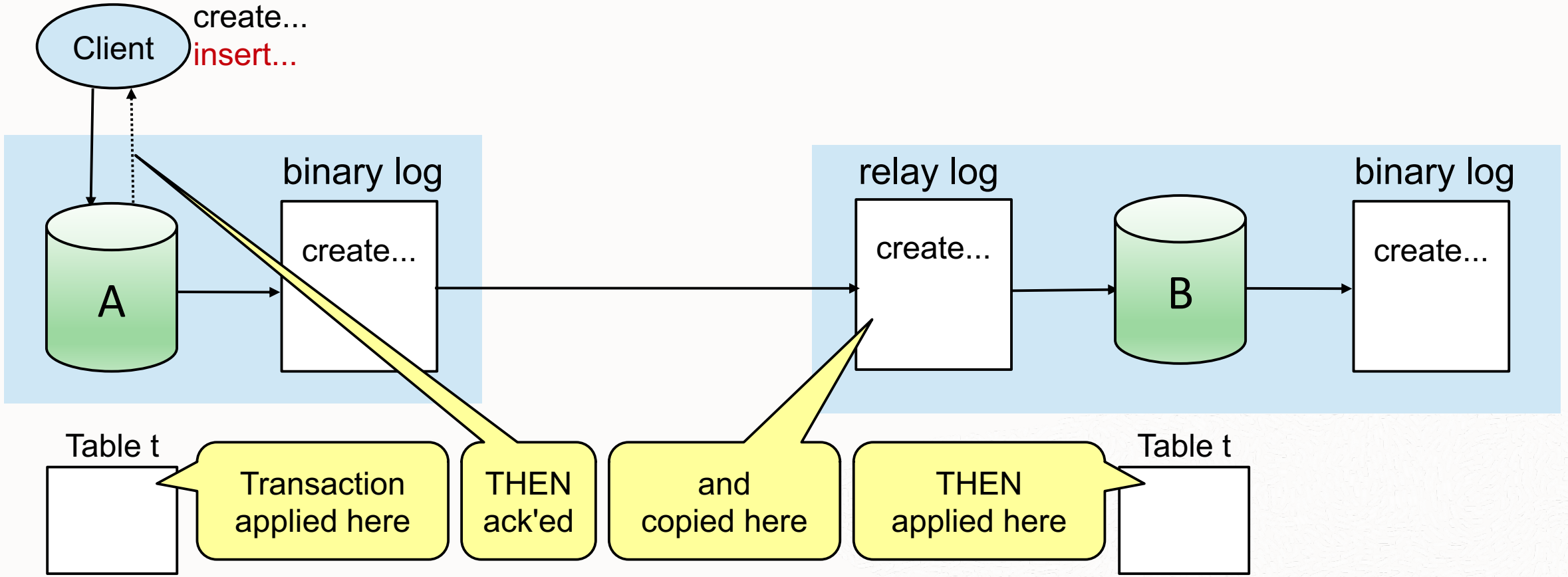
Replica Initiates Replication



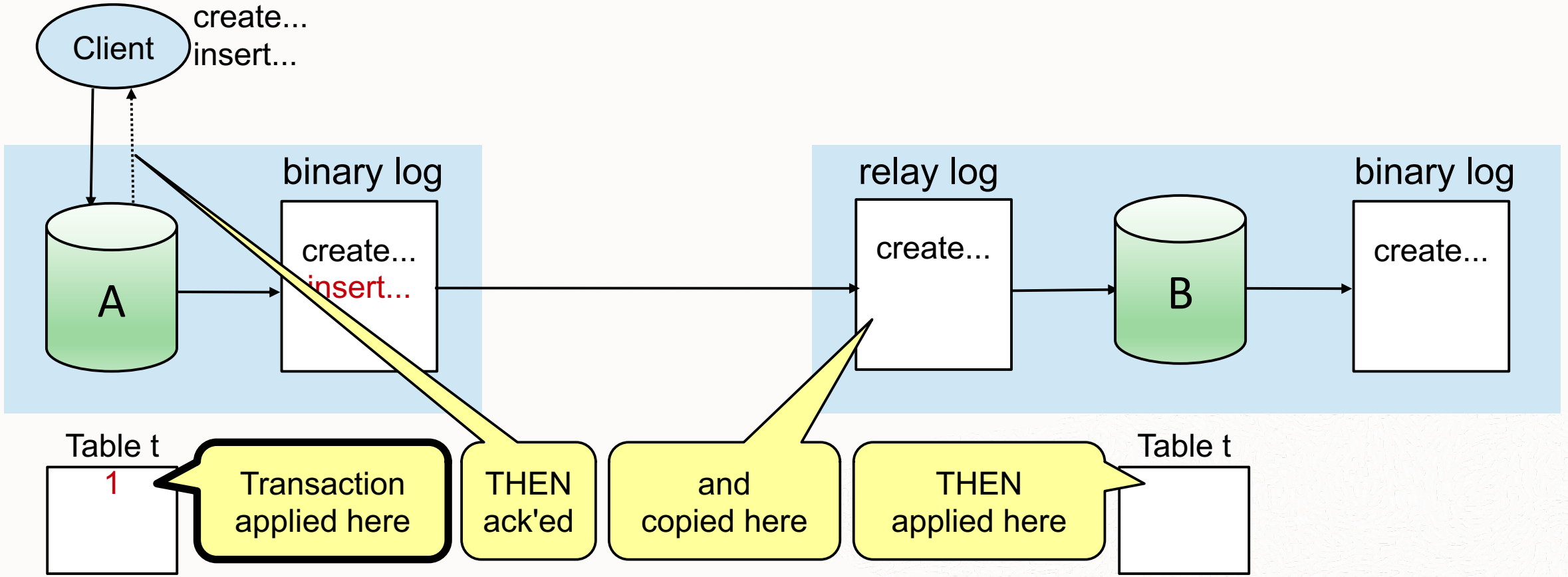
MySQL Replication
Asynchronous



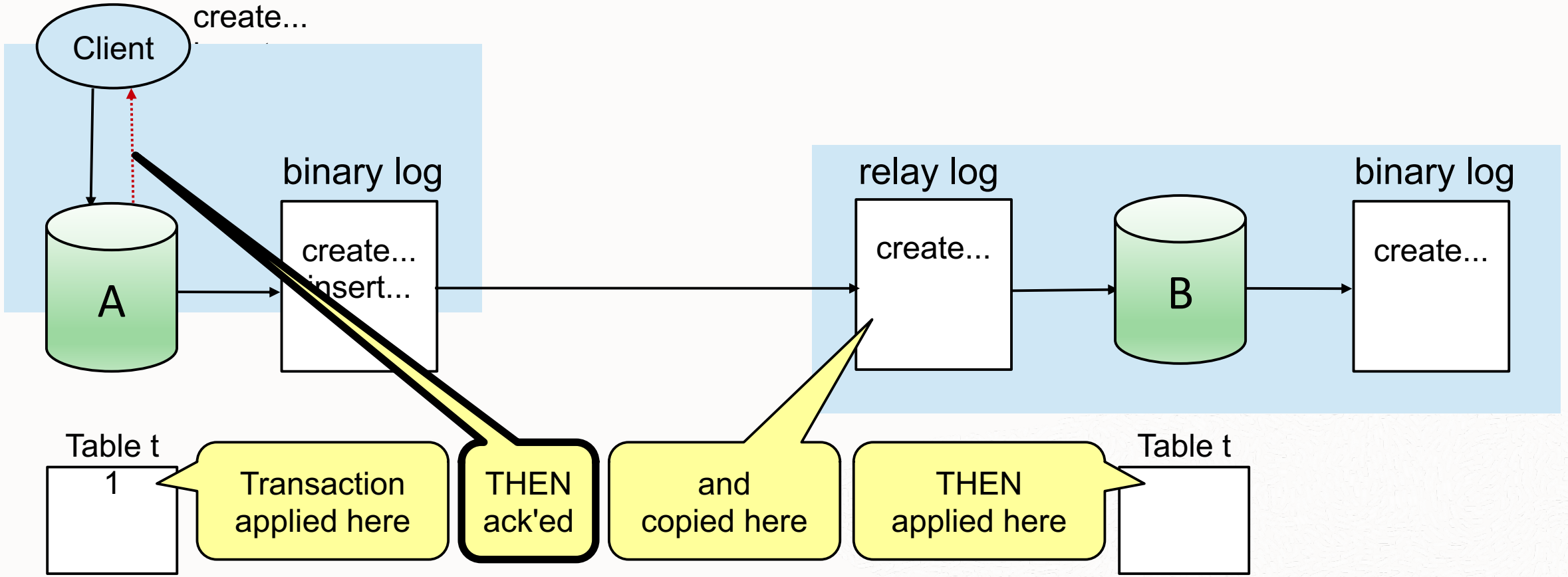
MySQL Replication
Asynchronous



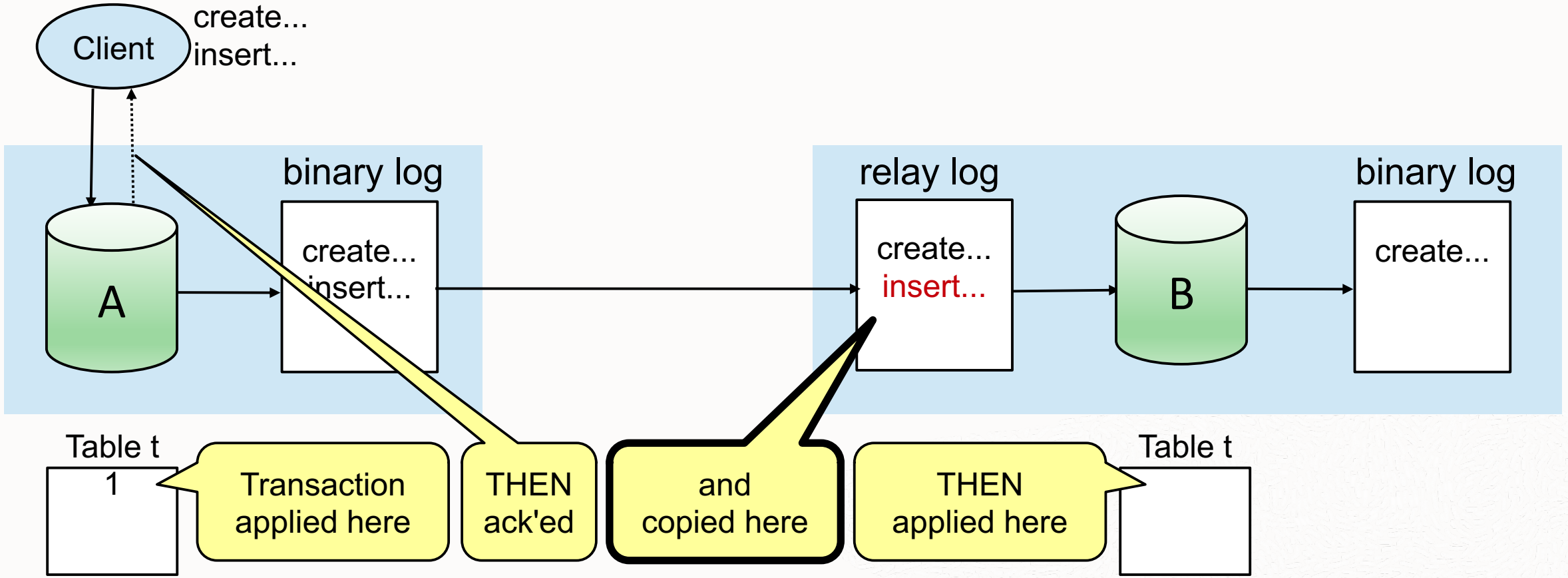
MySQL Replication
Asynchronous



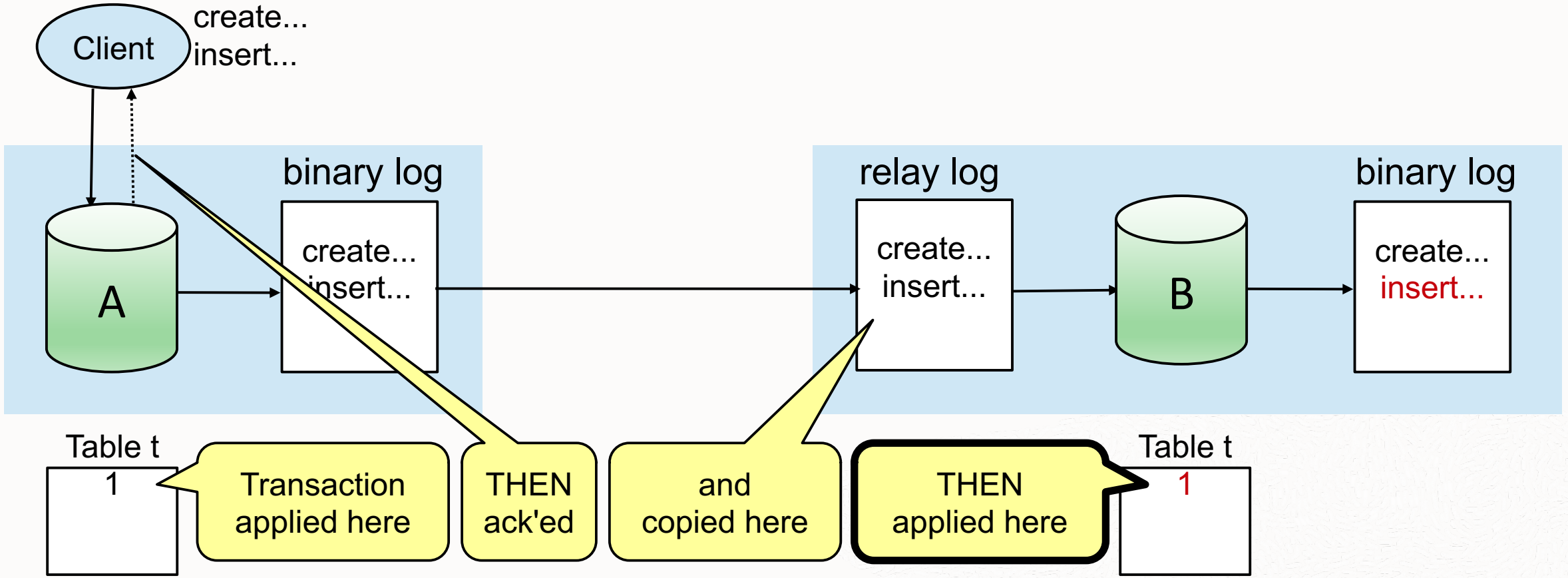
MySQL Replication
Asynchronous



MySQL Replication
Asynchronous



MySQL Replication
Asynchronous



Change Propagation

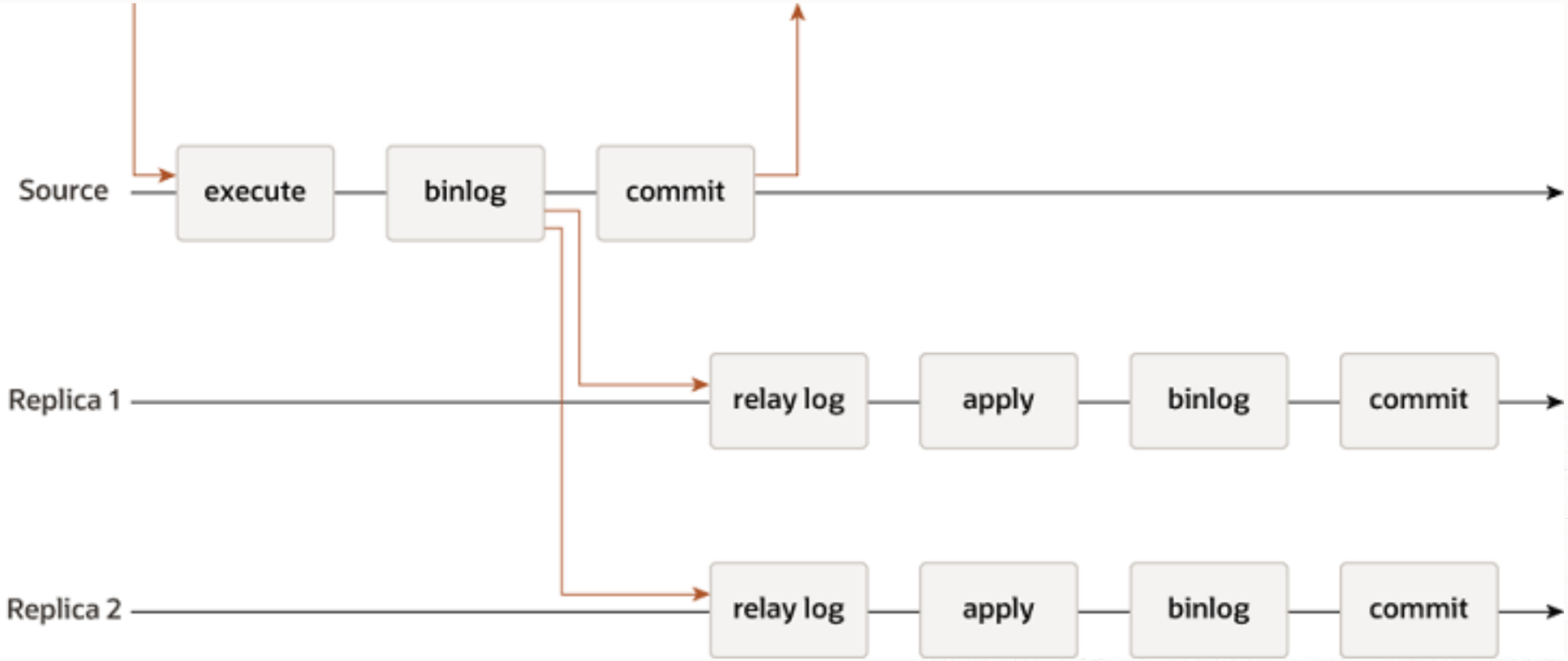
Asynchronous Replication

- Transactions committed immediately
- Events are propagated after the commit operation
- Faster, but vulnerable to lost updates on server crashes and inconsistency
- Native in MySQL Server

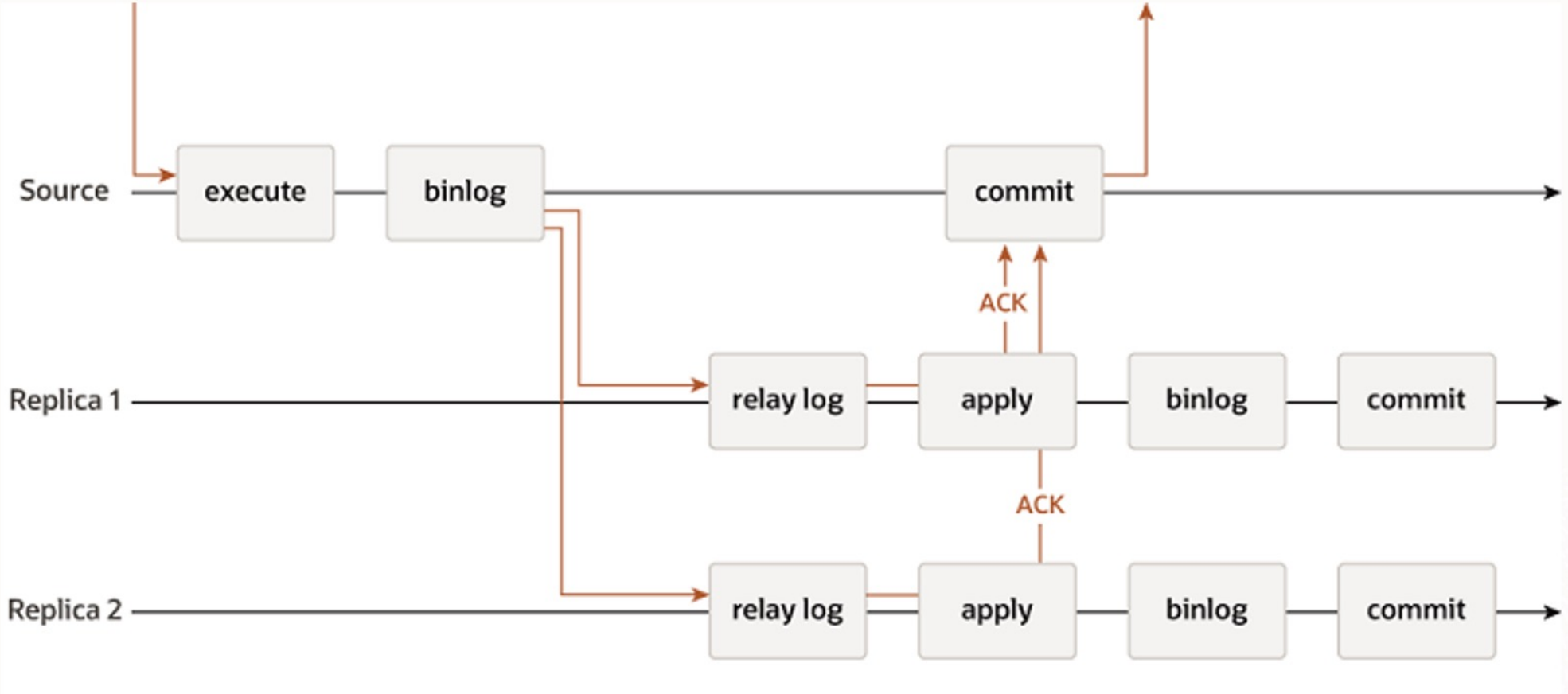
Semi-synchronous Replication

- Similar to asynchronous but commits are not acknowledged until N replicas have received and stored the correspondent events
- Introduced in MySQL 5.5

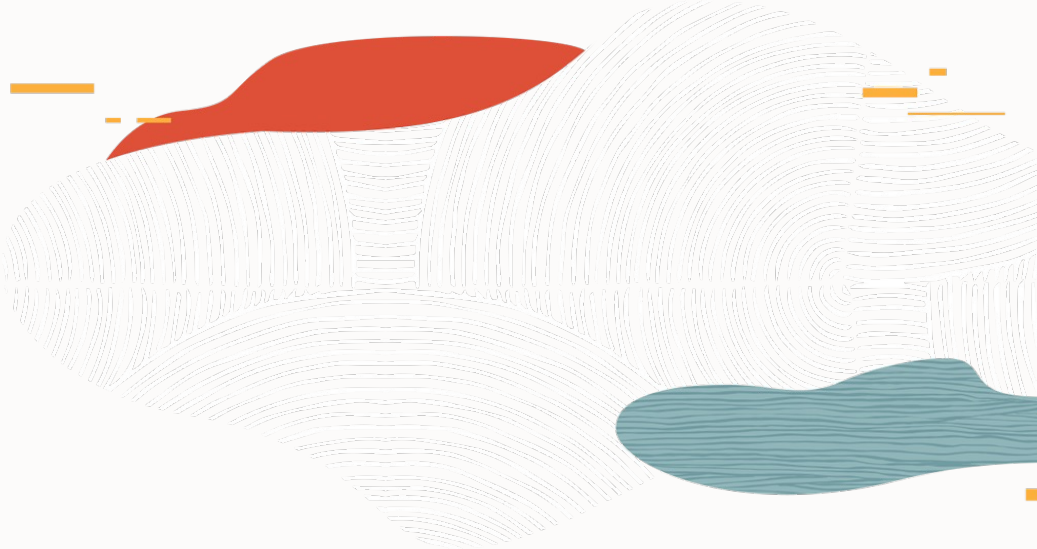
Replication Technologies: Native Replication



Replication Technologies: Semi-Synchronous Replication



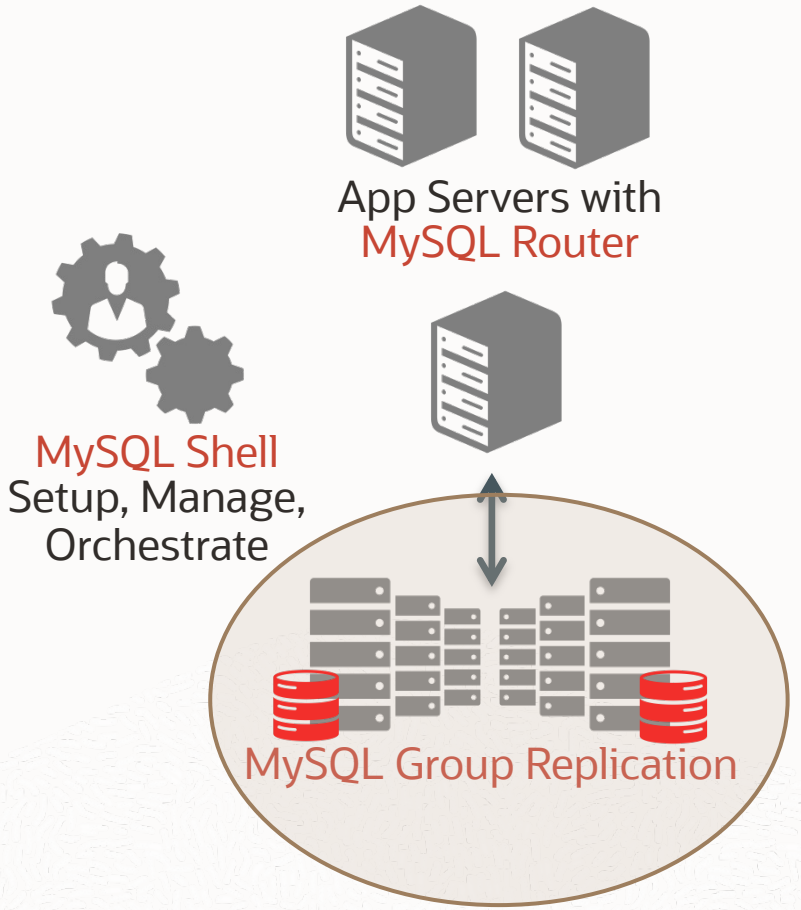
ORACLE



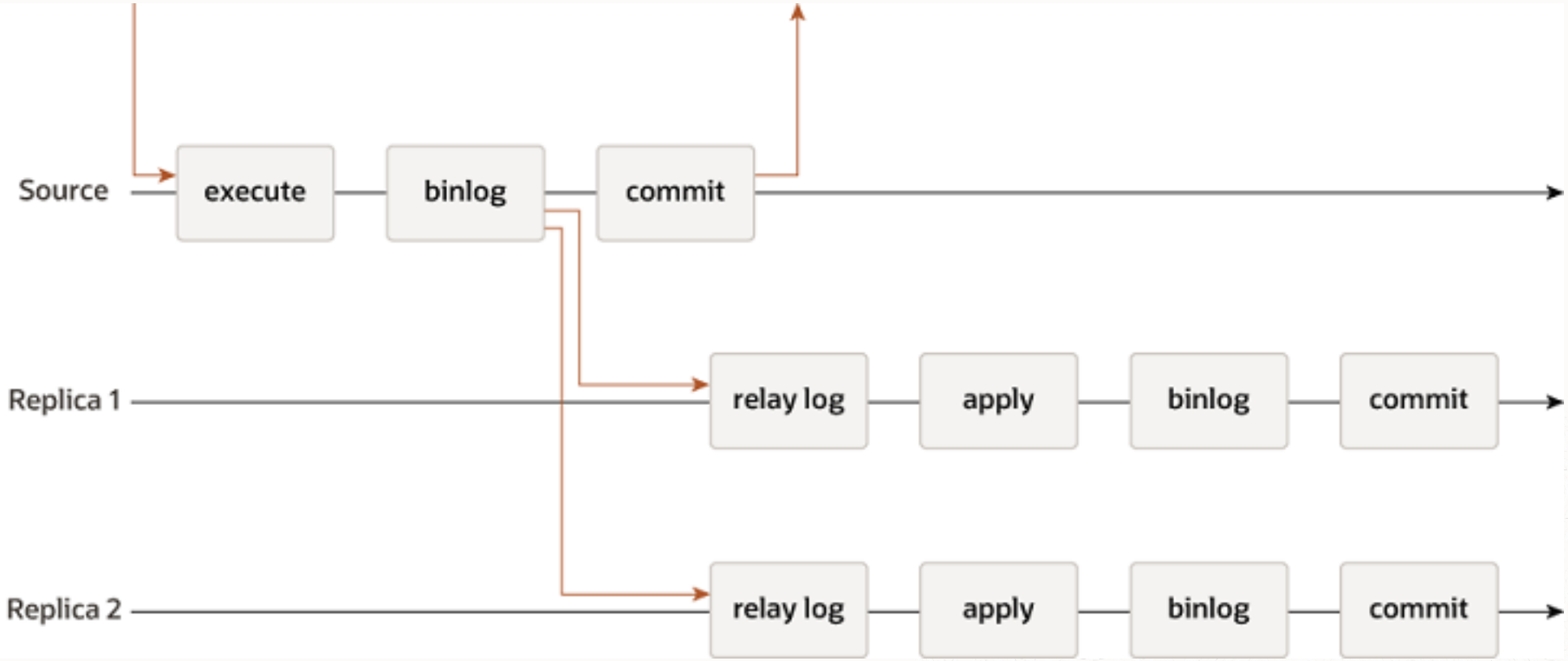
Group Replication

MySQL Group Replication

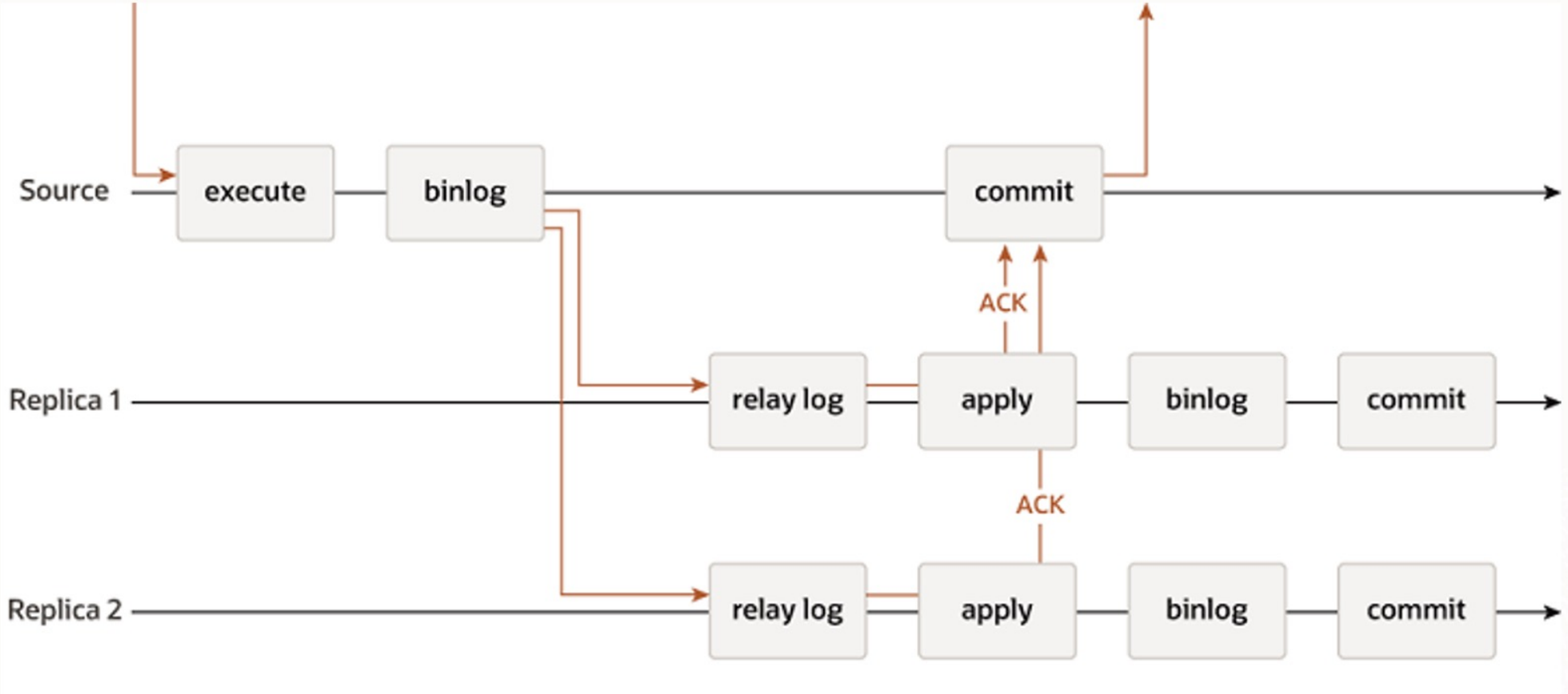
- **Implementation of Replicated Database State Machine**
  - Total Order – Writes
  - XCOM - Paxos implementation
- **Configurable Consistency Guarantees**
  - eventual consistency
  - 8.0+: per session & global read/write consistency
- **Using MySQL replication framework by design**
  - Binary & Relay logs
  - GTIDs: Global Trans+ Transaction IDs Generally
- **Available since MySQL 5.7**
- **Supported on all platforms:** Linux, Windows, Solaris, OSX, FreeBSD



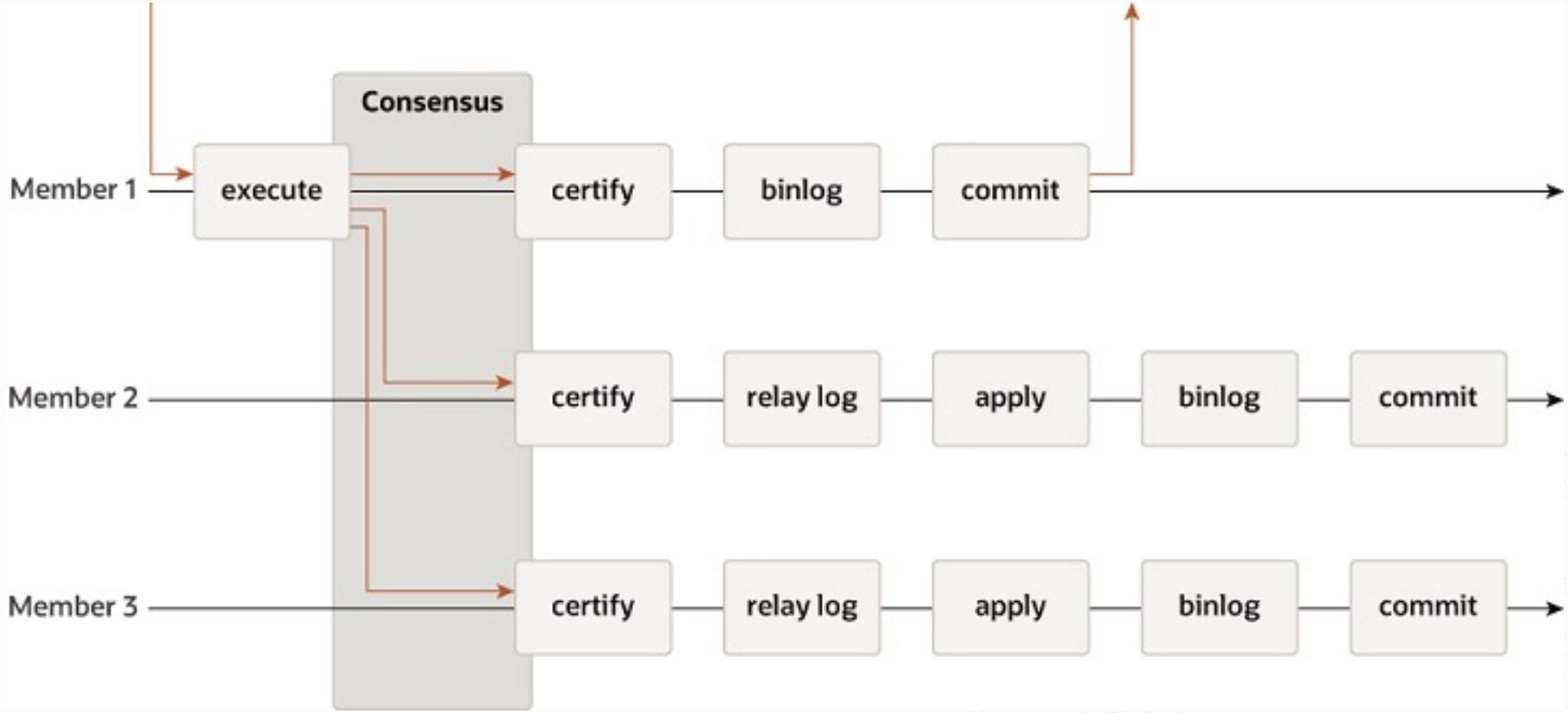
Replication Technologies: Native Replication



Replication Technologies: Semi-Synchronous Replication



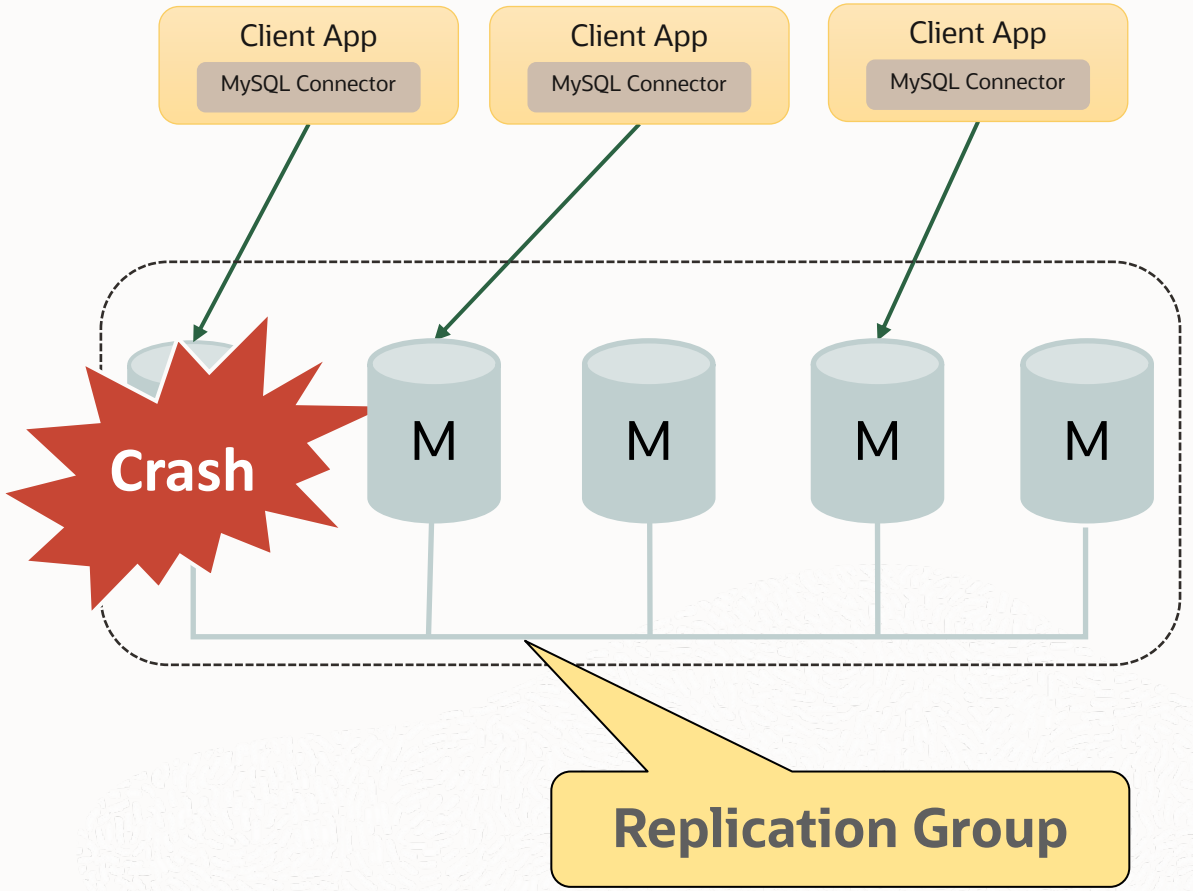
Replication Technologies: Group Replication



Understand Group Replication High Availability

Simpler Failover

No need to choose a new primary
No need to configure the new primary
No need to switch slaves to new primary



Understand Group Replication High Availability

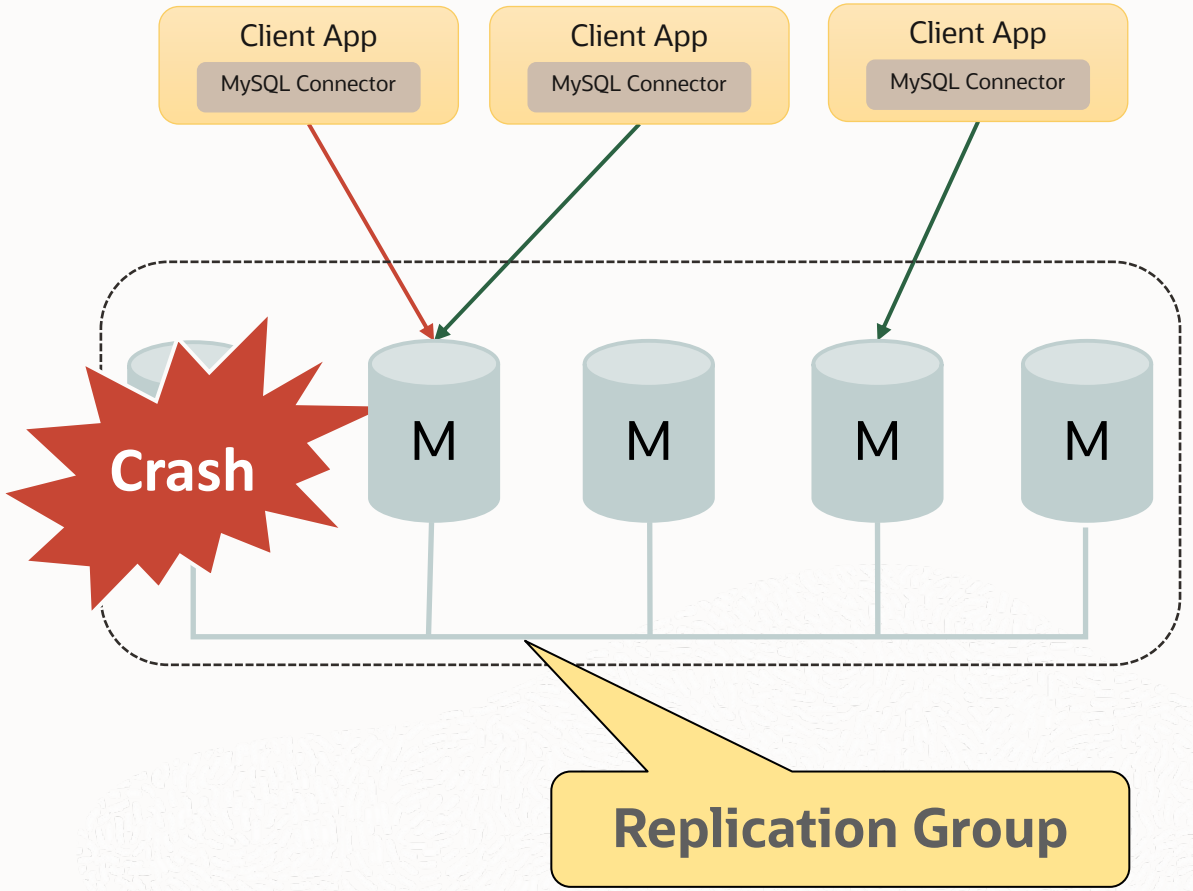
Simpler Failover

No need to choose a new primary

No need to configure the new primary

No need to switch slaves to new primary

Only need to switch crashed server's connections to other members.

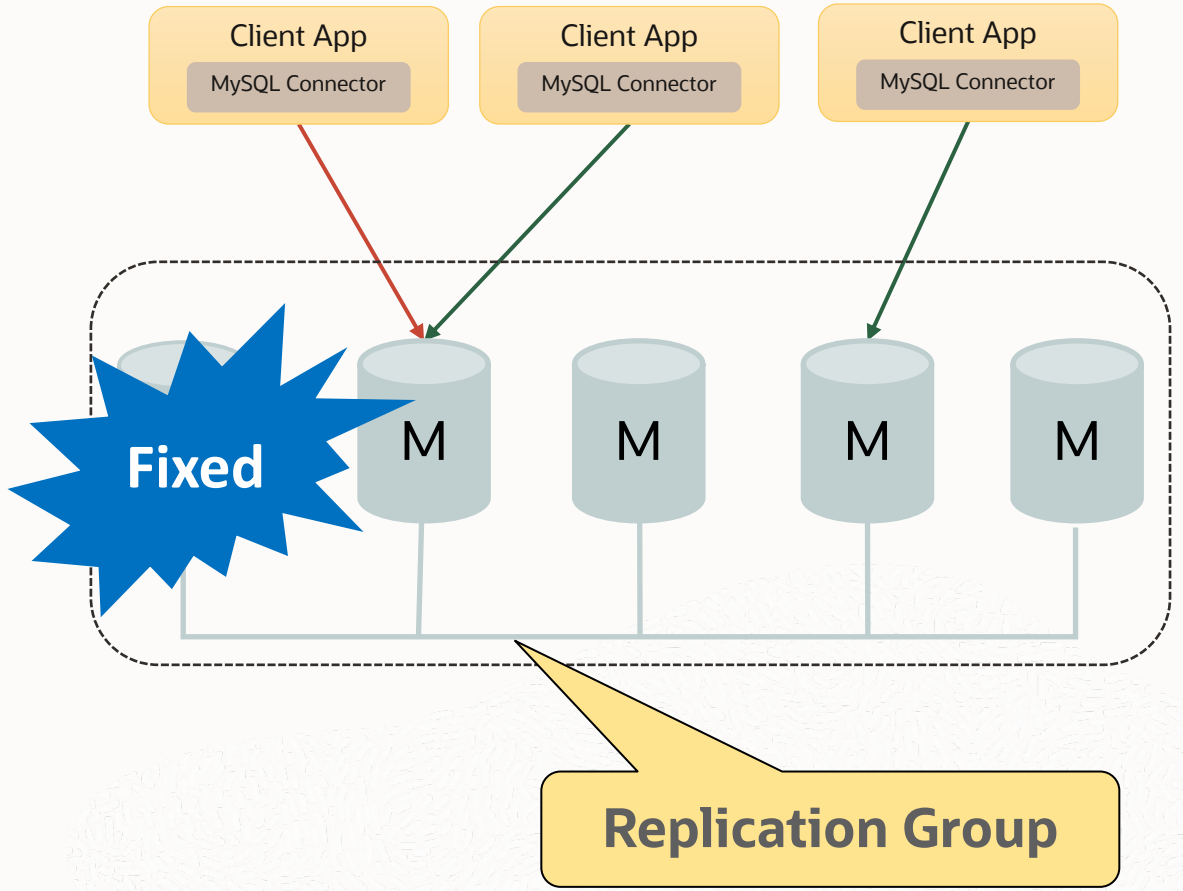


Understand Group Replication High Availability

Automatic Recovery

No need to check and truncate binlog events which are not replicated

No need to switch to new primary



Understand Group Replication High Availability

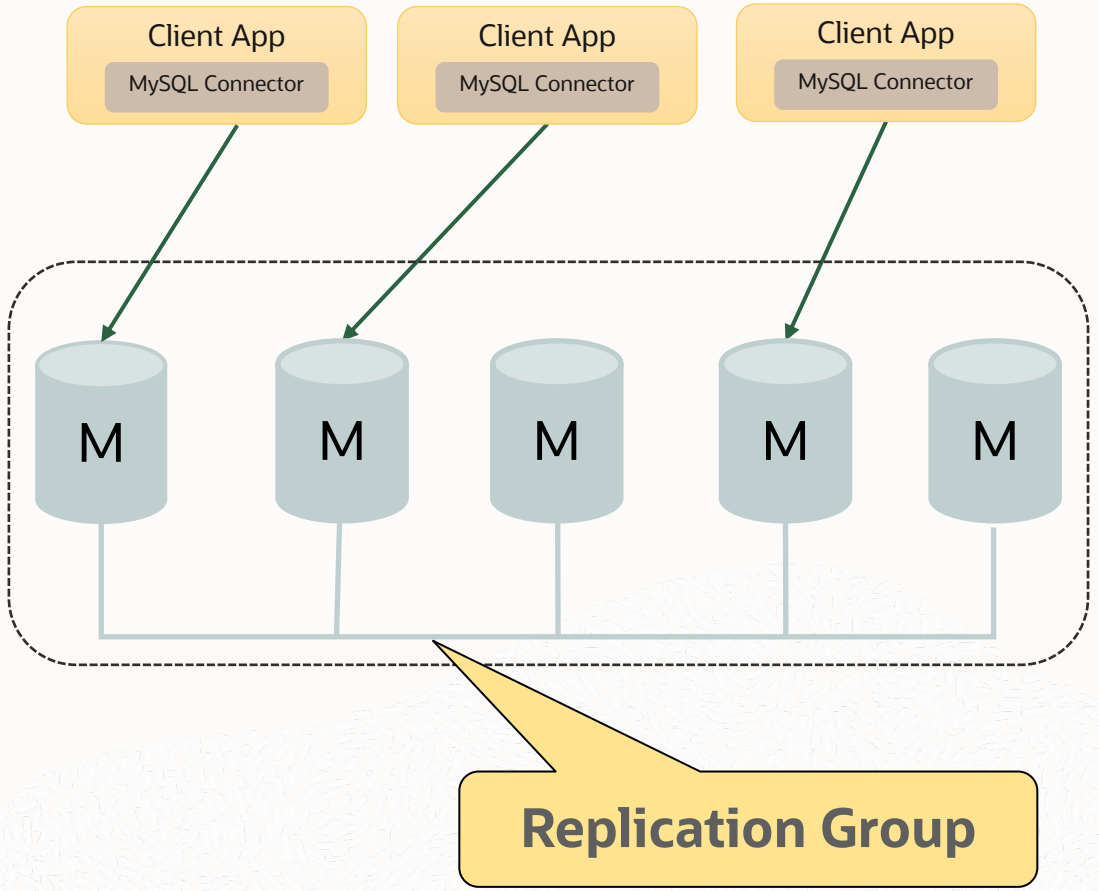
Automatic Recovery

No need to check and truncate binlog events which are not replicated

No need to switch to new primary

Just need to rejoin the group

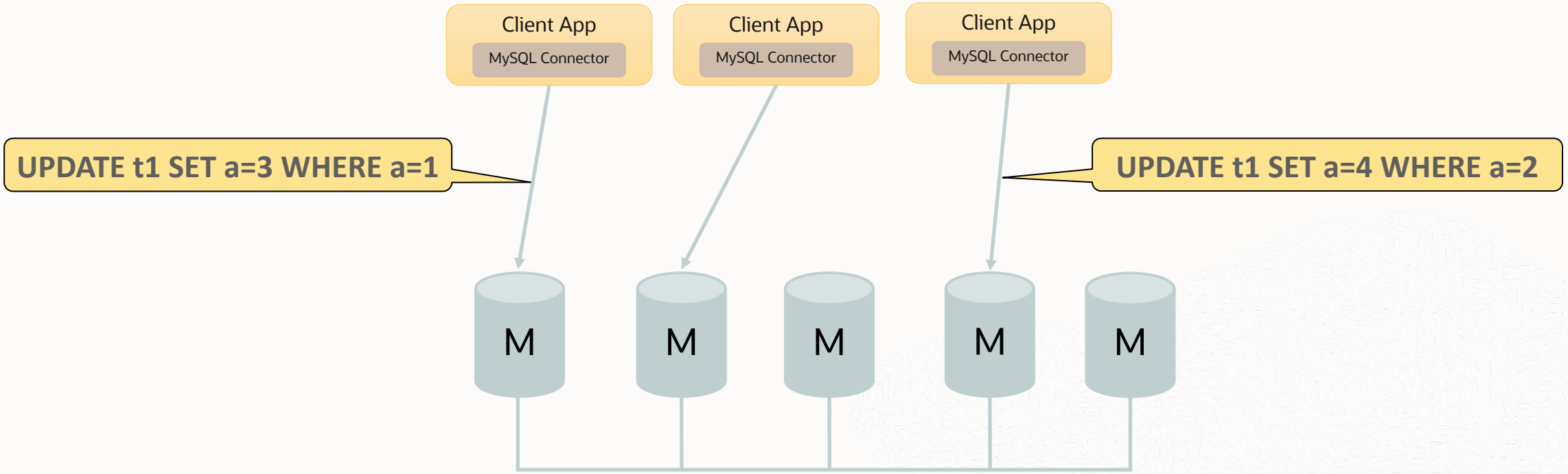
START GROUP_REPLICATION



Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.
Conflicts will be detected and dealt with.

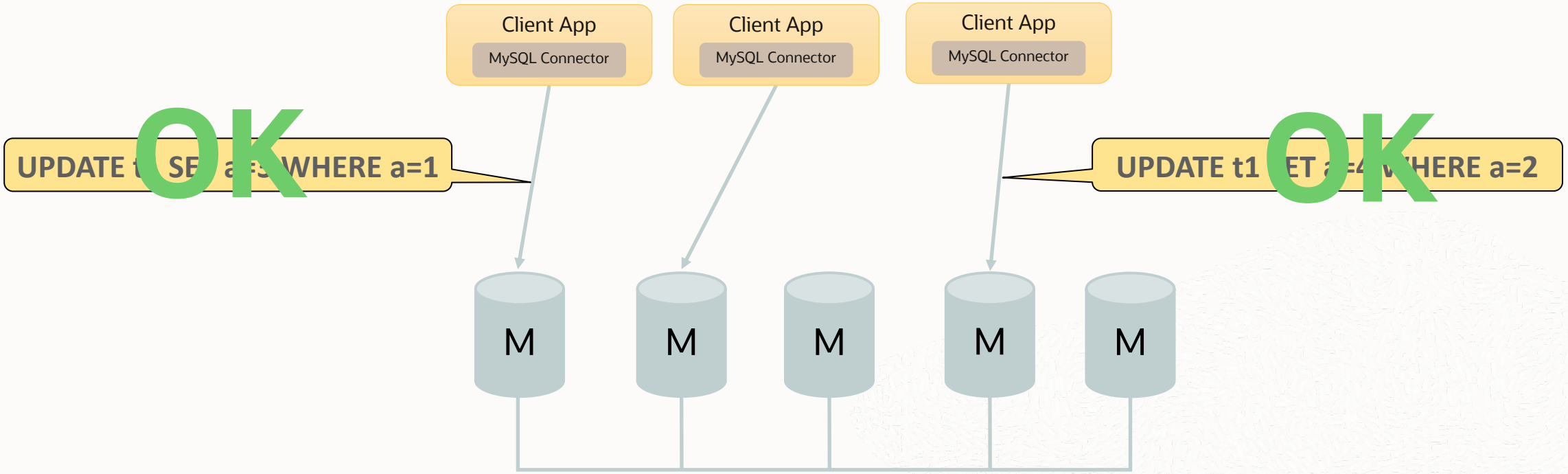
First committer wins rule.



Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.
Conflicts will be detected and dealt with.

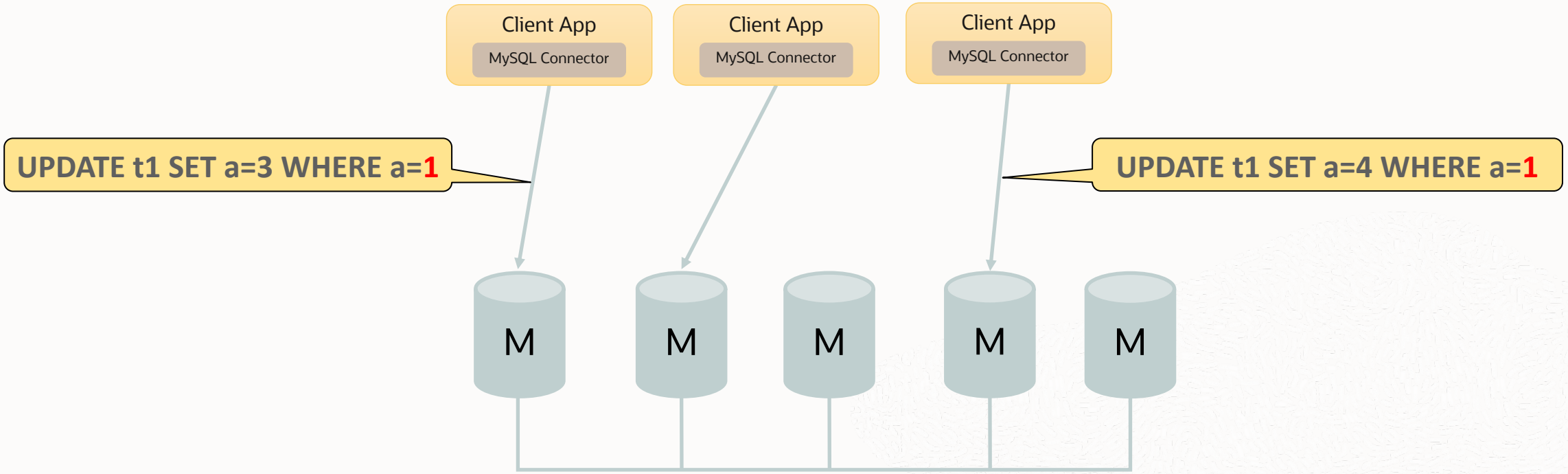
First committer wins rule.



Multi-Master update everywhere!

Any two transactions on different servers can write to the same tuple.
Conflicts will be detected and dealt with.

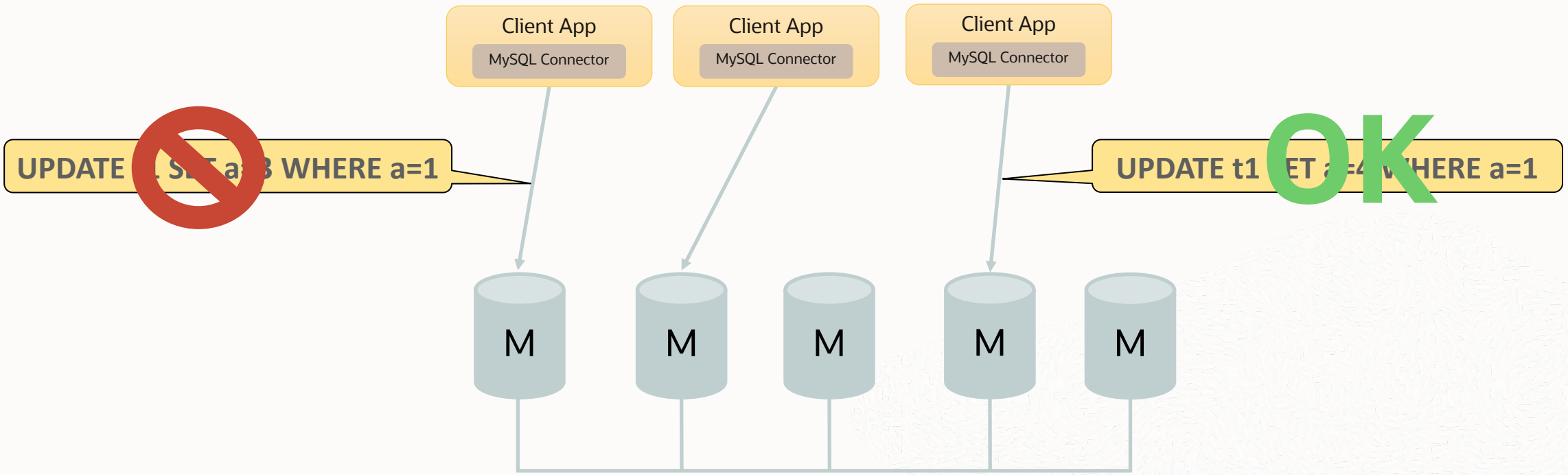
First committer wins rule.



Multi-Master update everywhere!

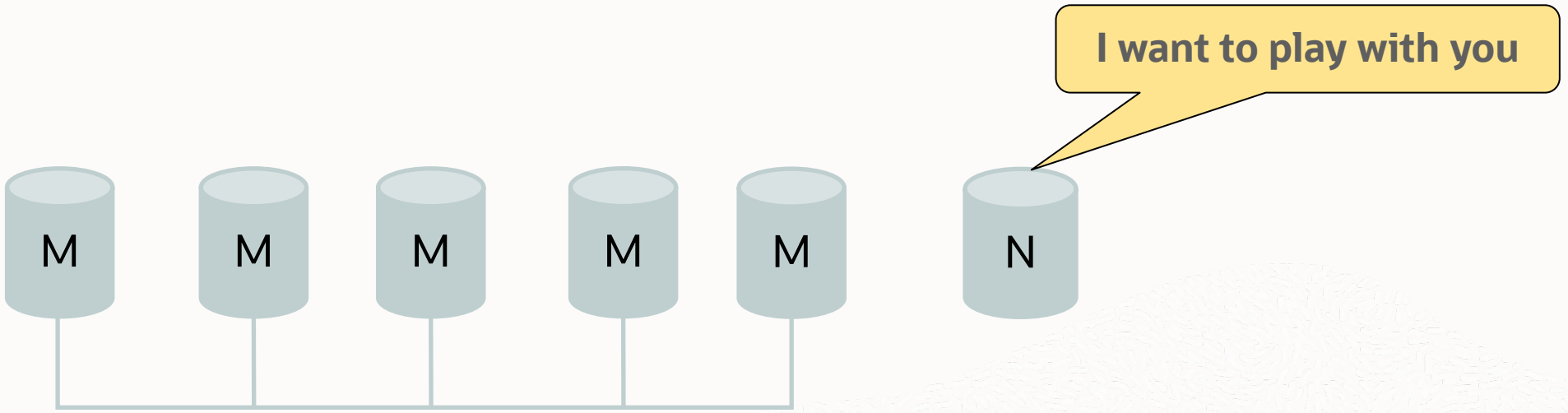
Any two transactions on different servers can write to the same tuple.
Conflicts will be detected and dealt with.

First committer wins rule.



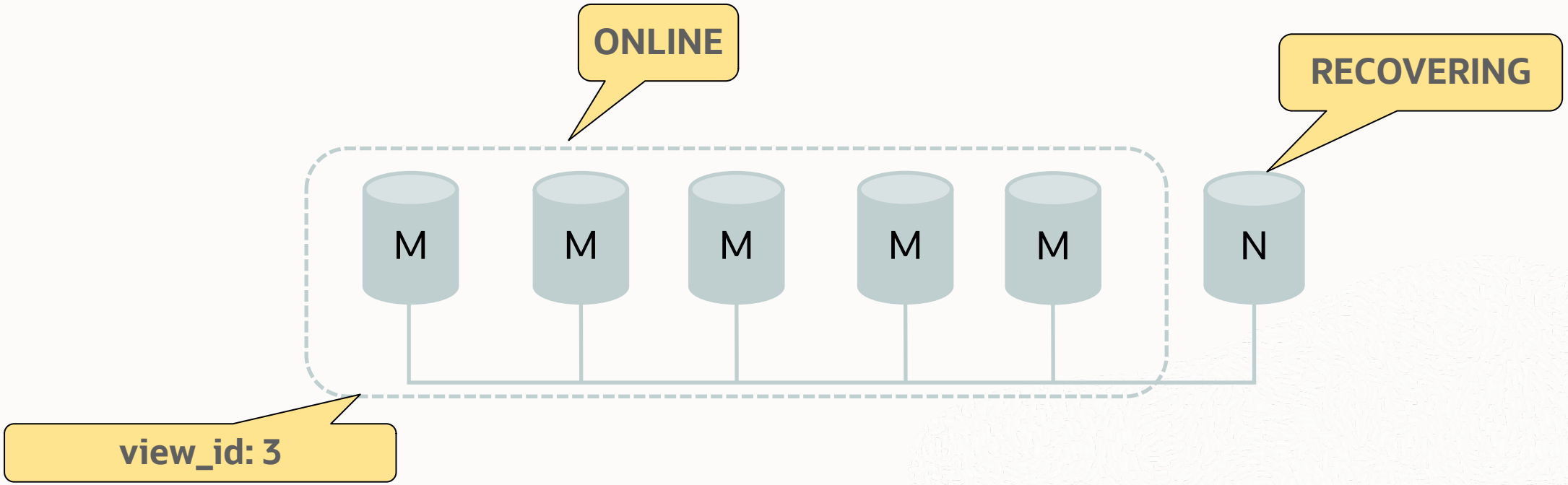
Automatic distributed server recovery!

Server that joins the group will automatically synchronize with the others.



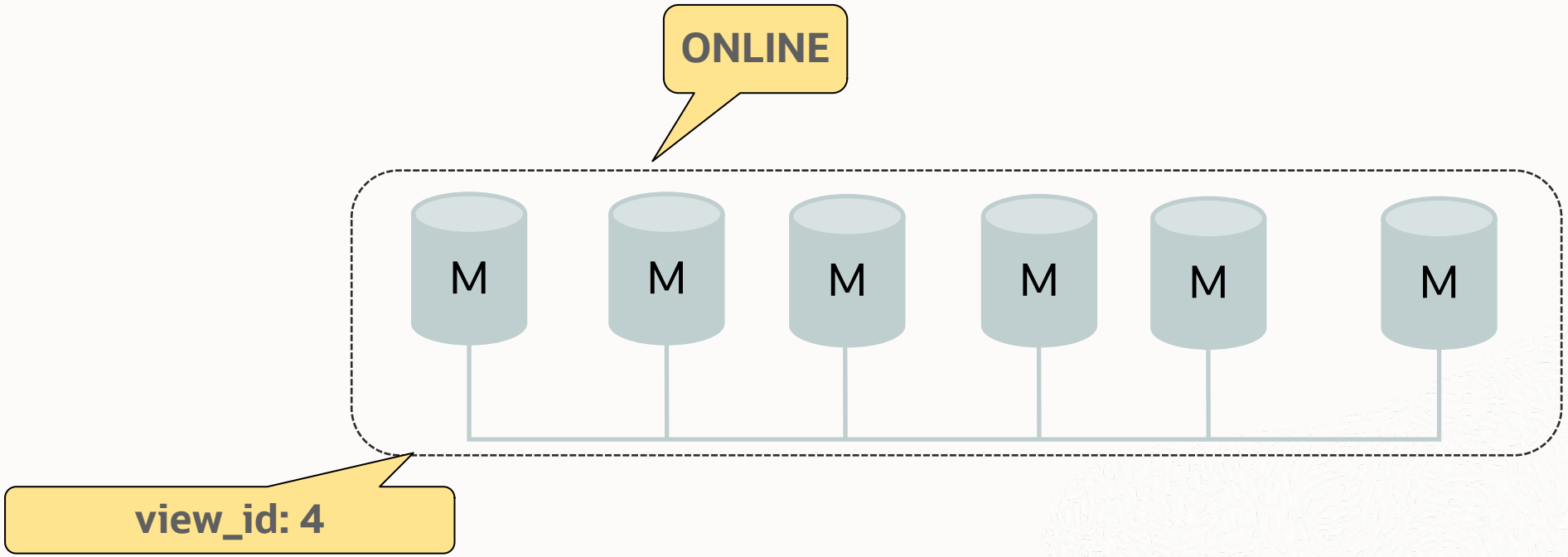
Automatic distributed server recovery!

Server that joins the group will automatically synchronize with the others.



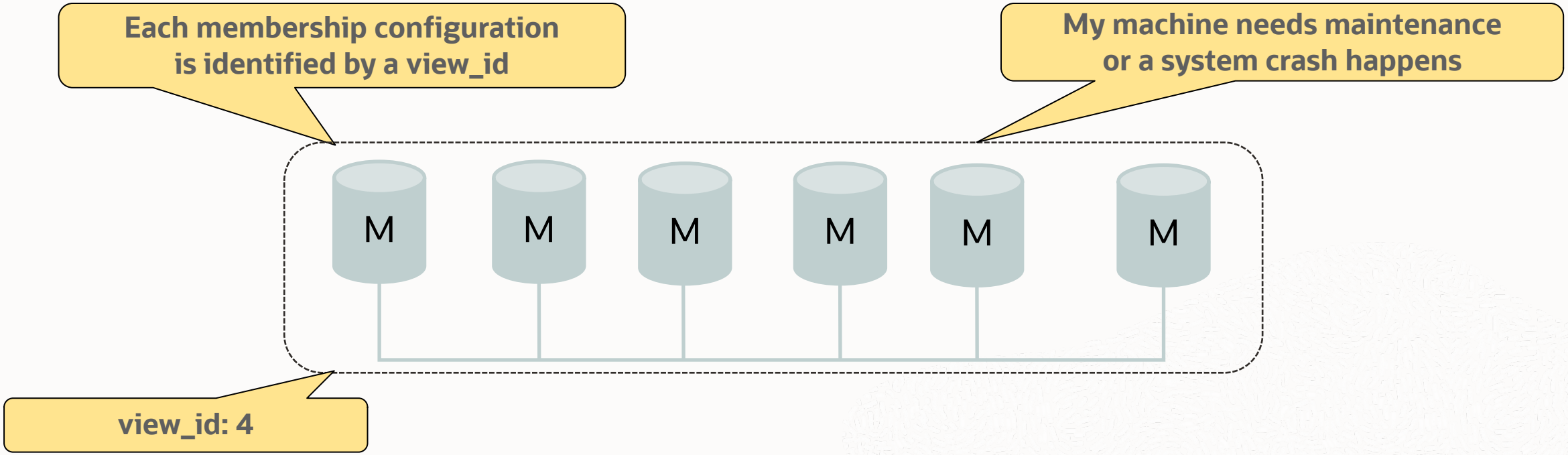
Automatic distributed server recovery!

Server that joins the group will automatically synchronize with the others.



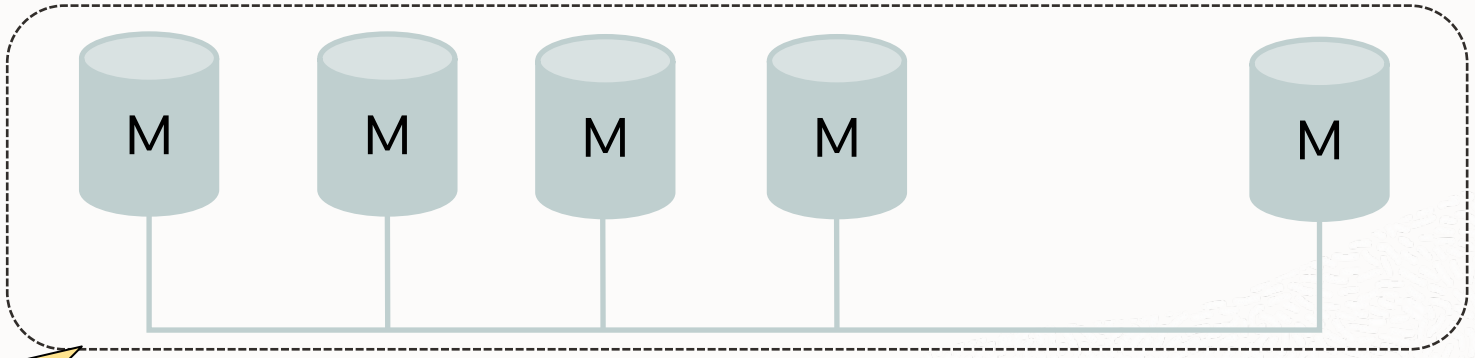
Automatic distributed server recovery!

If a server leaves the group, the others will automatically be informed.



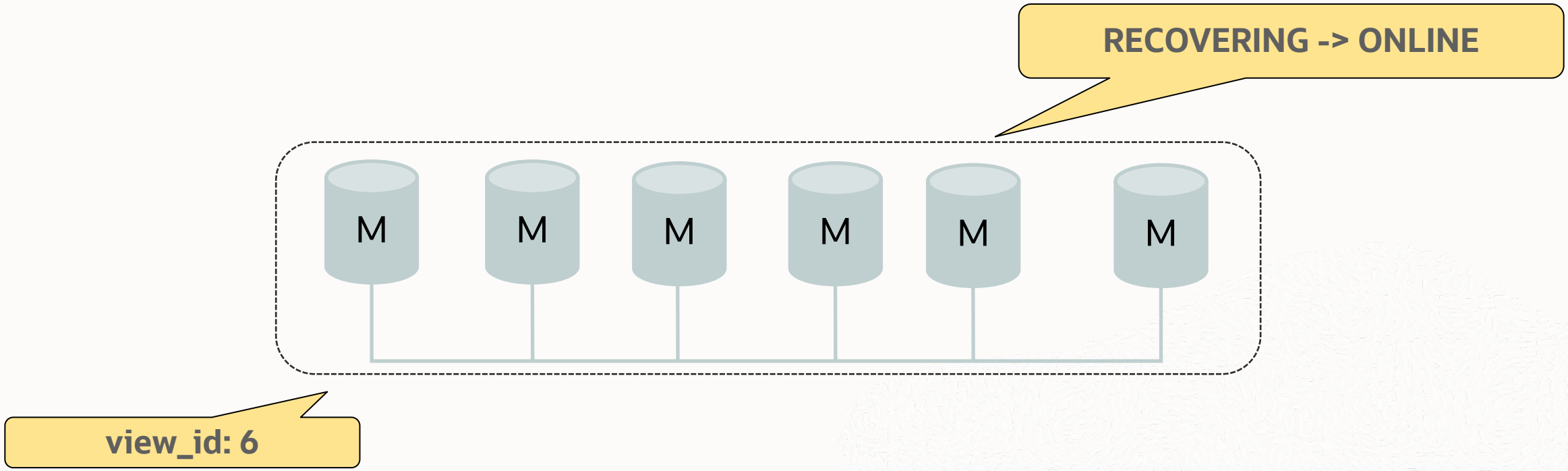
Automatic distributed server recovery!

If a server leaves the group, the others will automatically be informed.



Automatic distributed server recovery!

Server that (re)joins the group will automatically synchronize with the others.



Consistency Levels

Eventual Consistency (default)

- Transaction does not wait at all.
- Executes on the current snapshot of the data on that member.

Before Consistency (Synchronize on Reads)

- Transaction waits for all preceding transactions to complete.
- Executes on the most up to date snapshot of the data in the group.

After Consistency (Synchronize on Writes)

- Transaction waits until all members have executed it.
- Executes on the current snapshot of the data on that member.

Consistency Levels

Before and After (Yes, you can **combine** both)

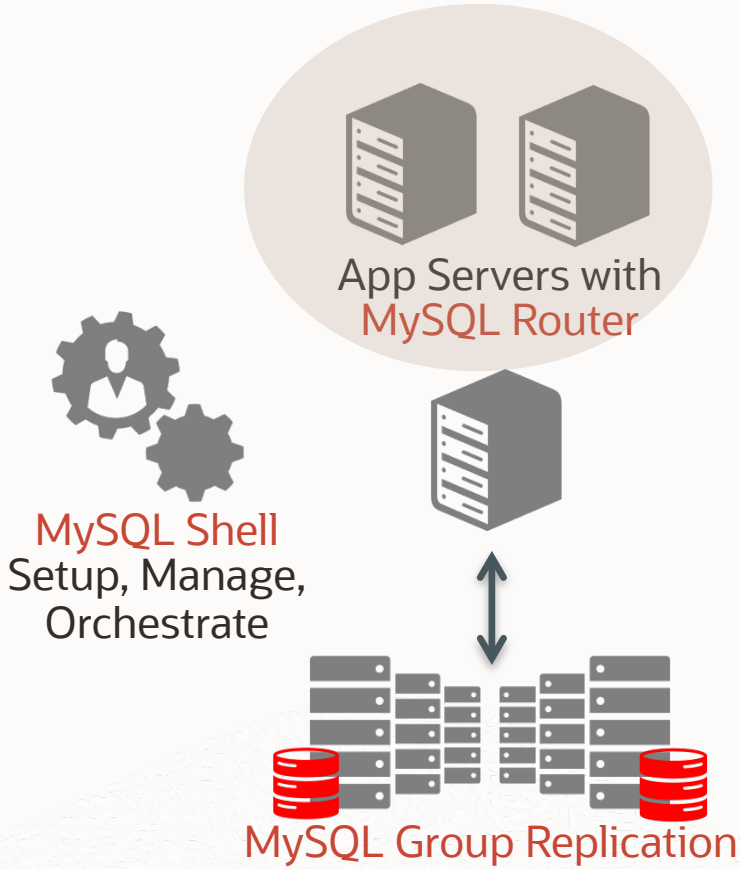
- Transaction waits for all preceding transactions and for all members to execute it.
- Executes on the most up to date snapshot of the data in the group and updates everywhere before returning to the application.

Before On Primary Fail-over

- Transaction waits for all transactions in the new primary's replication backlog to be executed.
- Executes on the snapshot of the data that the old primary was in when it stepped down (or crashed).

MySQL Router

- **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 topology
- **Currently TCP Port each for PRIMARY and NON-PRIMARY traffic**

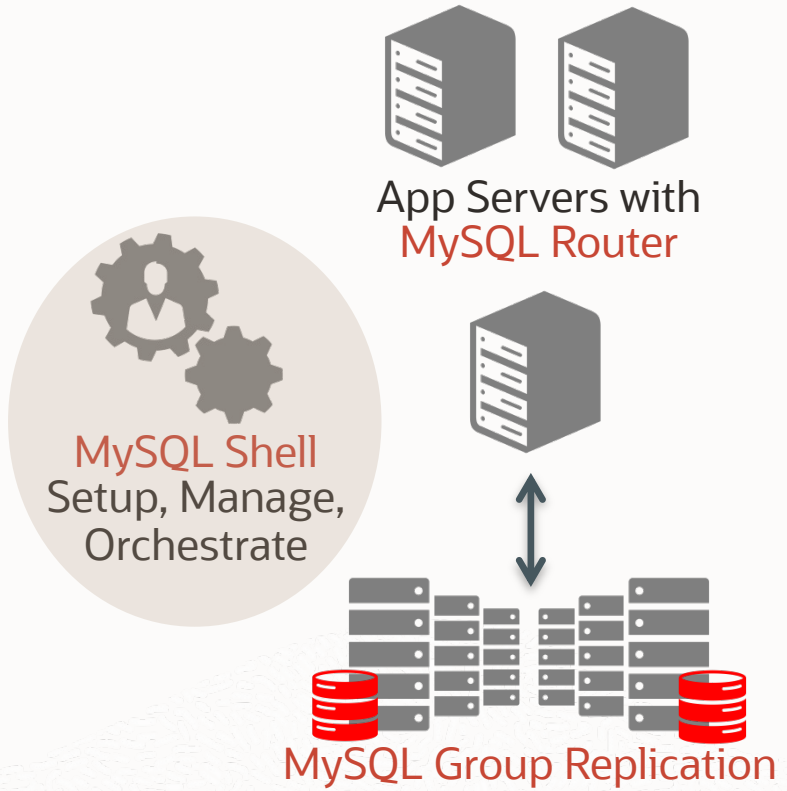


MySQL Shell

Database Administration Interface

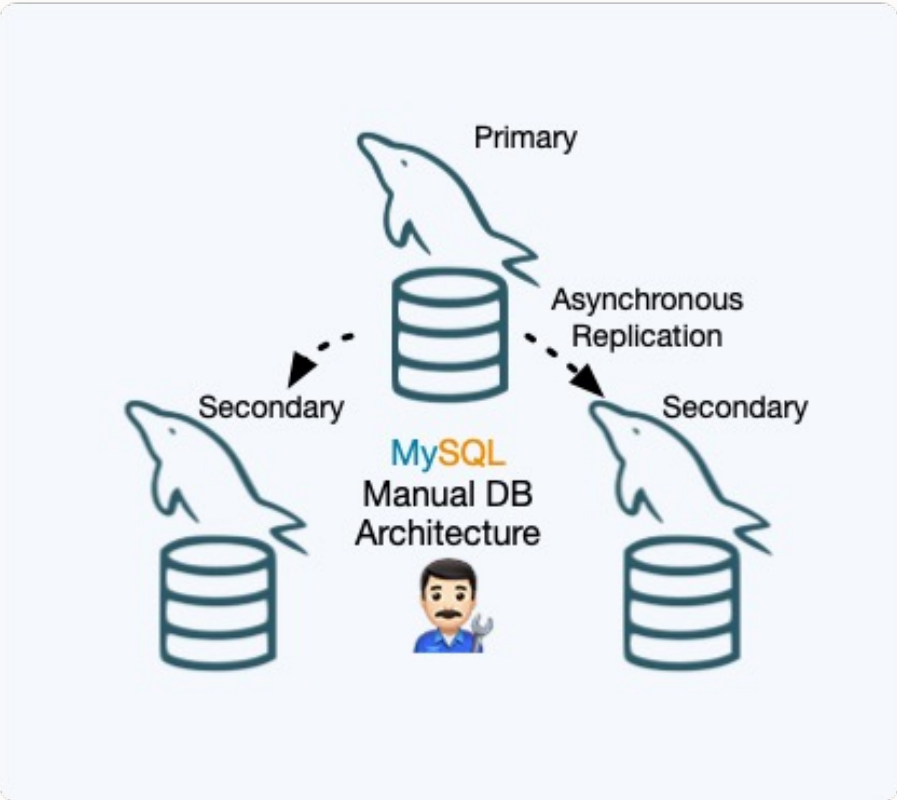
- Multi-LangMulti-Language: JavaScript, Python, and SQL
- Naturally scriptable
- Supports Document and Relational models
- Exposes full Development and Admin API
- Classic MySQL protocol and X protocol

" MySQL Shell provides the developer and DBA with a single intuitive, flexible, and powerful interface for all MySQL related tasks!"



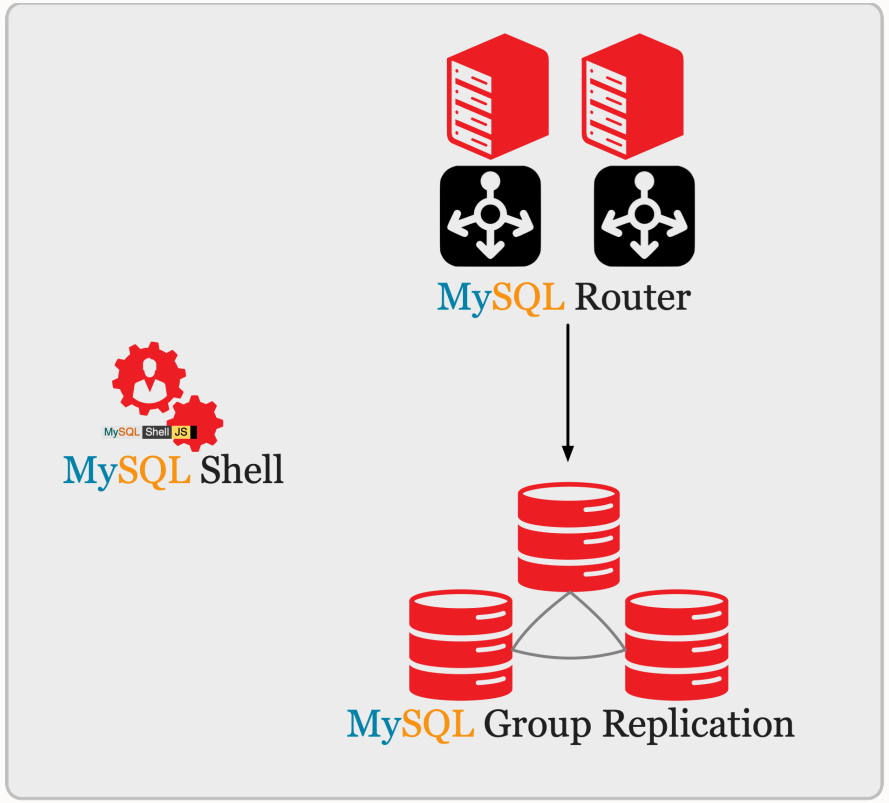
Past, Present & Future

'Past' - Manual



- Setting up Replication topology was usually done manually, taking many steps
  - including user management, restoring backups, configuring replication...
- MySQL only offered the technical pieces, leaving it up to the user to setup an (always customized) architecture
- Even required other software ... bringing lot's of work for DBA's and experts, who spent their time automating and integrating their customized architecture

Present - Solutions!



RPO = 0

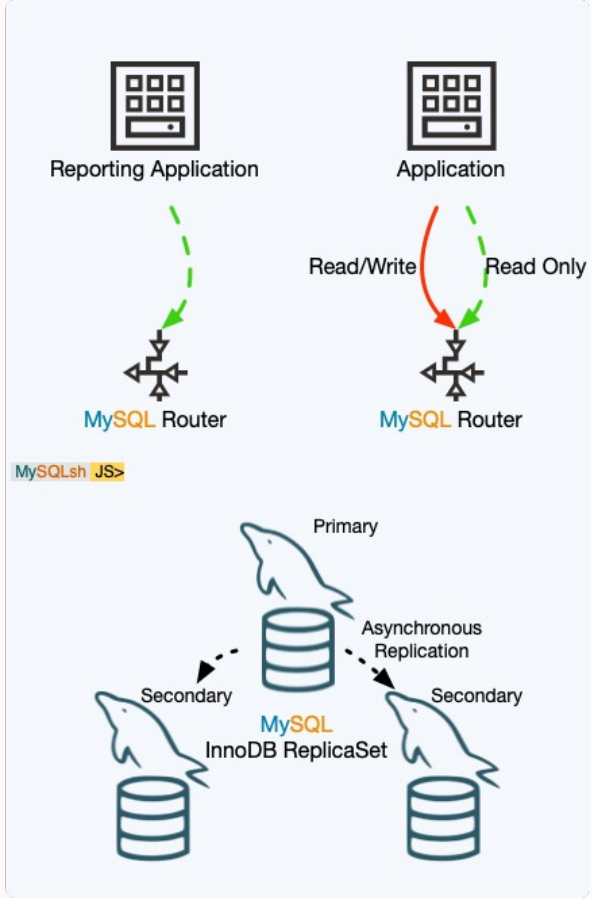
RTO = seconds (automatic failover)

2016 - MySQL InnoDB Cluster

- MySQL Group Replication: Automatic membership changes, network partition handling, consistency…
- MySQL Shell to provide a powerful interface that helps in automating and integrating all components
- InnoDB `CLONE` to automatically provision members, fully integrated in InnoDB
- MySQL Router
- MySQL Server

"A single product — MySQL — with high availability and scaling features baked in; providing an integrated end-to-end solution that is easy to use."

Present - Solutions!



RPO != 0

RTO = minutes (manual failover)

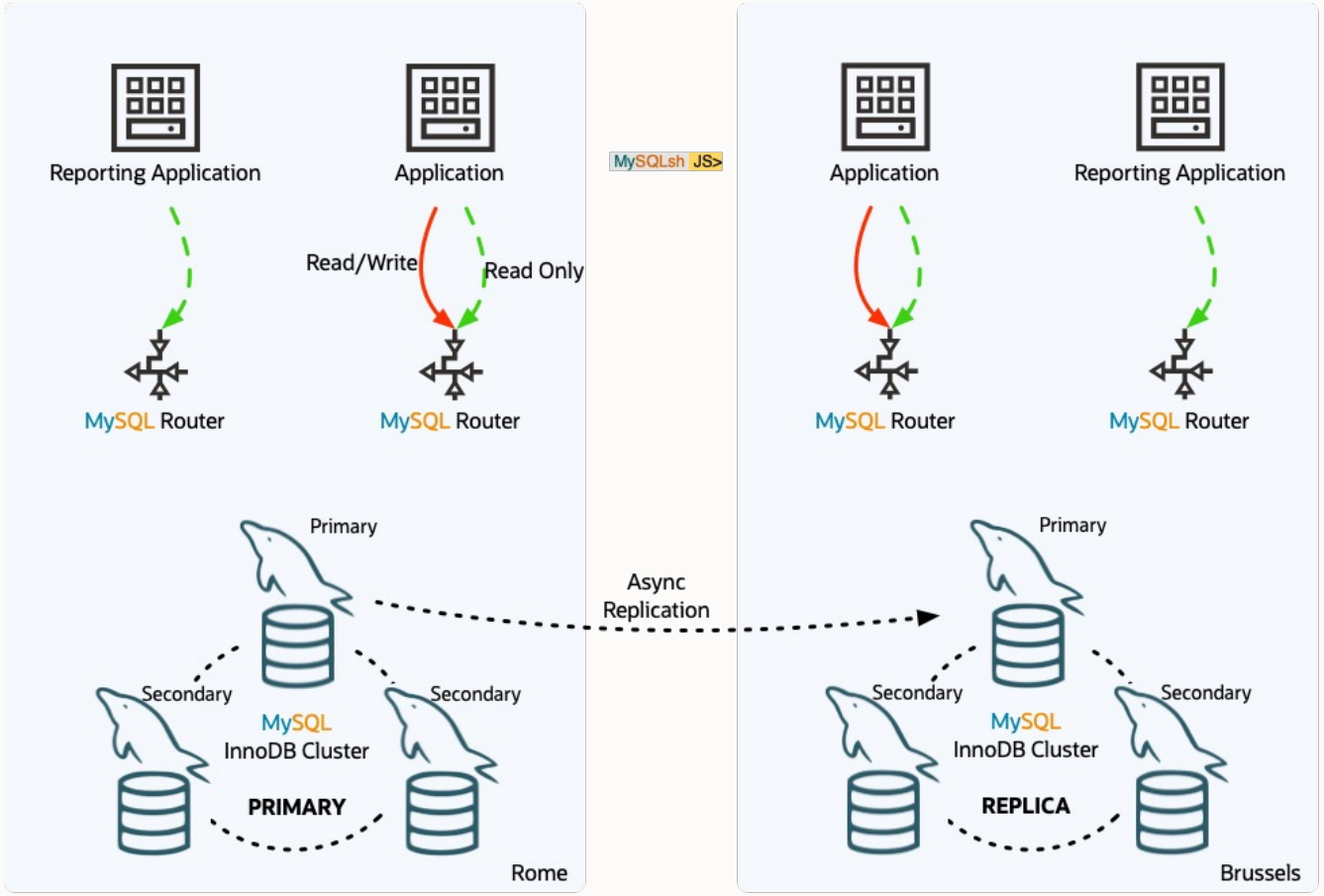
2020 - MySQL InnoDB Replicaset

- 'classic', 'asynchronous' Replication based Solution, fully integrated
- MySQL Shell
- MySQL Router
- MySQL Server

MySQL InnoDB ClusterSet

MySQL InnoDB ClusterSet

One or more REPLICA MySQL InnoDB Clusters attached to a PRIMARY MySQL InnoDB Cluster



High Availability (Failure Within a Region)

- RPO = 0
- RTO = seconds (automatic failover)

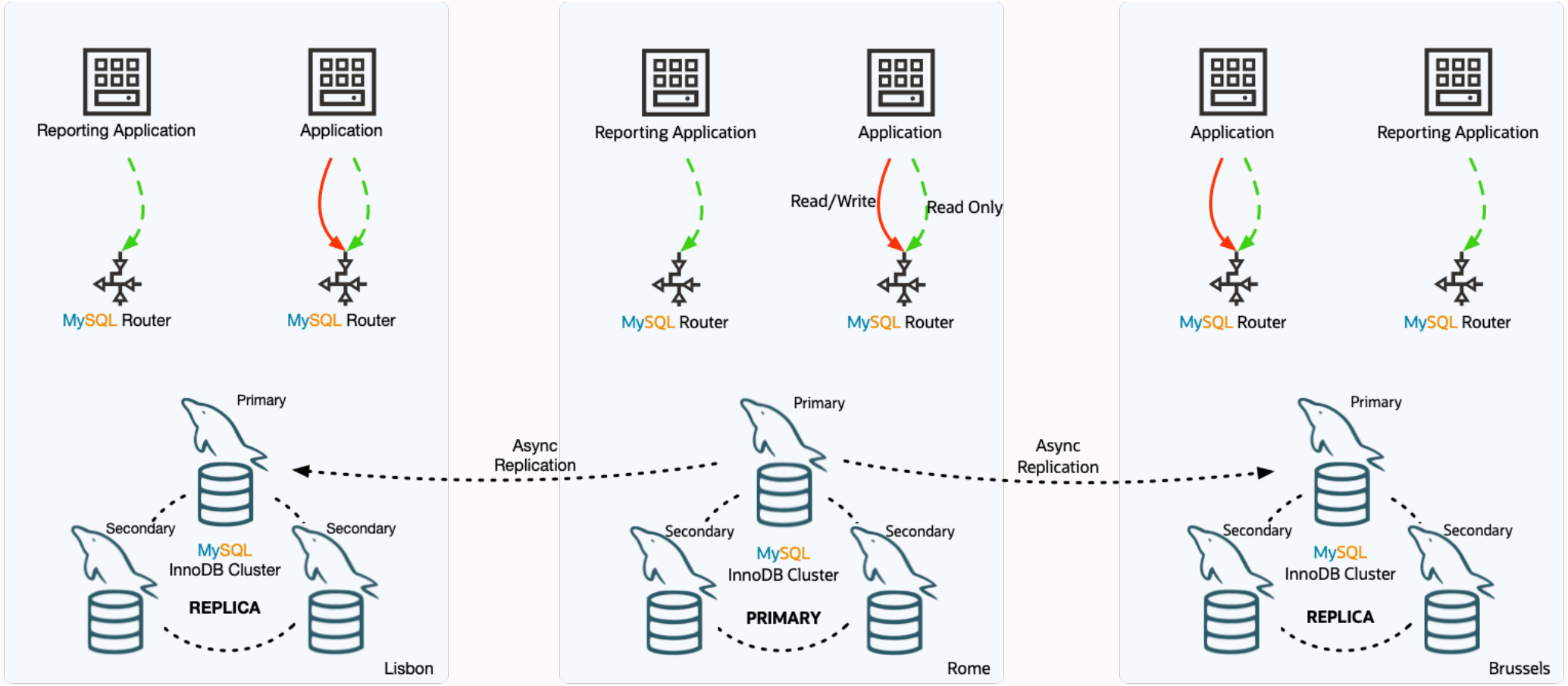
Disaster Recovery (Region Failure)

- RPO != 0
- RTO = minutes or more (manual failover)
- No write performance impact

Features

- Easy to use
- Familiar interface and usability
  `mysqlsh`, `CLONE`, ...
- Add/remove nodes/clusters online
- Router integration, no need to reconigure application if the topology changes

MySQL InnoDB ClusterSet - 3 Datacenters



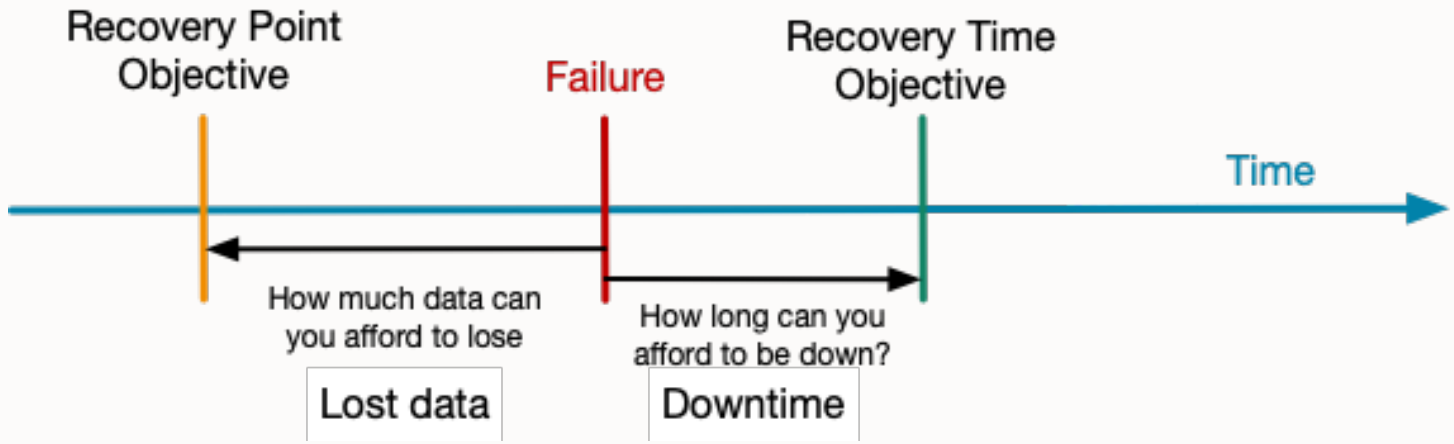
Business Requirements

Business Requirements
Concepts - RTO & RPO

- RTO: Recovery Time Objective
  - How long does it take to recover from a single failure
- RPO: Recovery Point Objective
  - How much data can be lost when a failure occurs

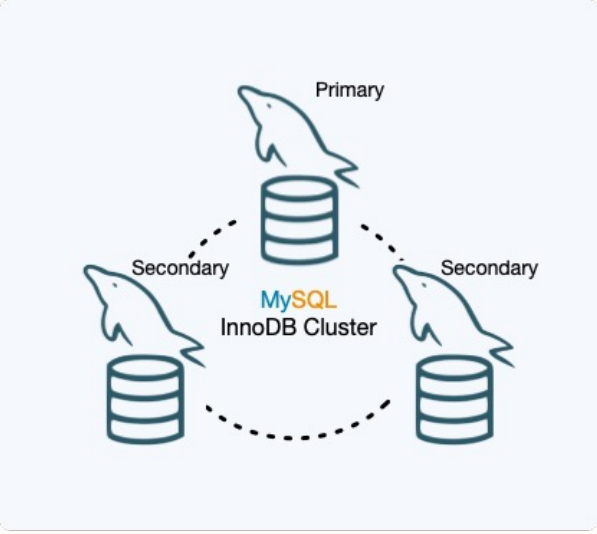
Types of Failure:

High Availability: Single Server Failure, Network Partition
Disaster Recovery: Full Region/Network Failure
Human Error: Little Bobby Tables



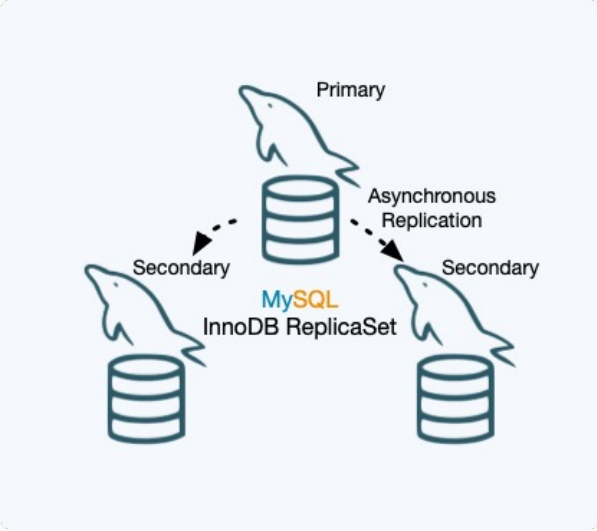
High Availability - Single Region

MySQL InnoDB Cluster



- RPO = 0
- RTO = Seconds

MySQL InnoDB ReplicaSet



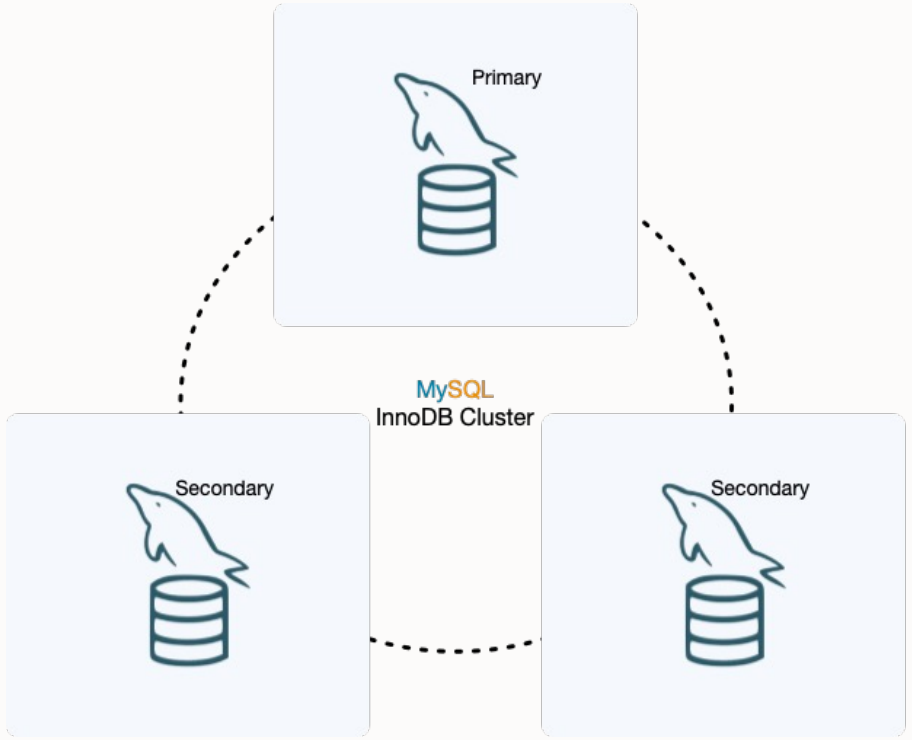
- RPO != 0
- RTO = Minutes+ (manual failover)
- Best write performance
- Manual failover

Disaster Recovery - Multi Region

MySQL InnoDB Cluster

- RPO = 0
- RTO = Seconds

- Multi-Region Multi-Primary
- 3 DC
- Requires very stable WAN
- Write performance affected by latency between dc's



Disaster Recovery - Multi Region

MySQL InnoDB ClusterSet

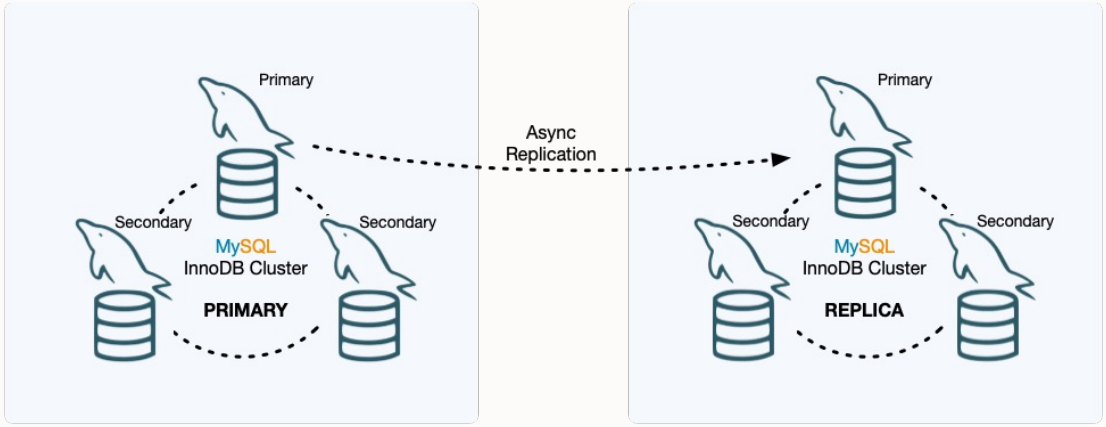
- RPO != 0
- RTO = Minutes+ (manual failover)

- RPO = 0 & RTO = seconds within Region (HA)
- Write performance (no sync to other region required)

  Higher RTO: Manual failover

  RPO != 0 when region fails

MySQL InnoDB ClusterSet



Workshop Overview

Workshop Overview

- Goals
  1. Create a OCI Compute server for hosting MySQL Enterprise Edition
  2. Install MySQL Enterprise Edition
  3. Overview & Setup
     1. InnoDB ReplicaSets
     2. InnoDB Cluster
     3. InnoDB ClusterSets.
- What this Workshop is not:
  - In-depth tutorial on Oracle Cloud Infrastructure
  - MySQL Training Class

- Lab:
  - https://bit.ly/HA_Workshop

Thank you