TRANSACTION

Lê Hồng Hải
UET-VNUH

Overview

1. Introduction
2. Commit-Rollback
3. Isolation Levels
4. Deadlock

Transaction example

- A bank customer transfers money from his savings account to his current account
- A transaction to add new sales order:
  1. Insert a new sales order into the *orders table* for a given customer.
  2. Insert new sales order items into the *orderdetails* table

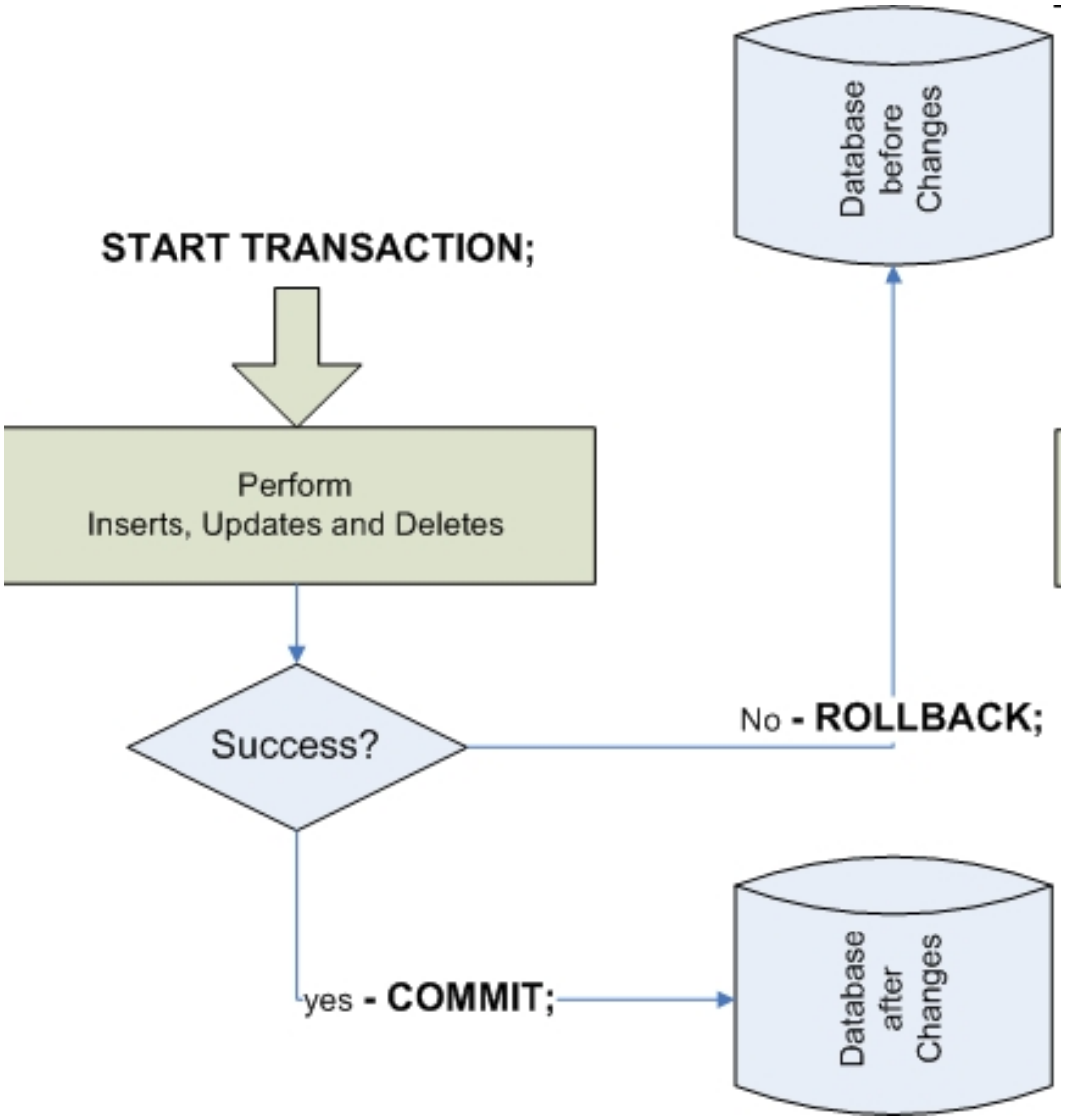
Transaction example

- Now, imagine what would happen *if one or more steps above fail* due to some reasons

Transaction

- Transaction allows you to execute a set of operations to ensure that the database never contains the result of partial operations
- If one of them fails, the rollback occurs to restore the database to its original state
- If no error occurs, the entire set of statements is committed to the database

Transaction-Commit-Rollback



Script that performs the above steps:

```
-- 1. start a new transaction
START TRANSACTION;
-- 2. Get the latest order number
SELECT
    @orderNumber:=MAX(orderNUmber)+1
FROM
    orders;
-- 3. insert a new order for customer 145
INSERT INTO orders(orderNumber,
                   orderDate,
                   requiredDate,
                   shippedDate,
                   status,
                   customerNumber)
VALUES(@orderNumber,
       '2005-05-31',
       '2005-06-10',
       '2005-06-11',
       'In Process',
        145);
```

Script that performs the above steps (ctn..)

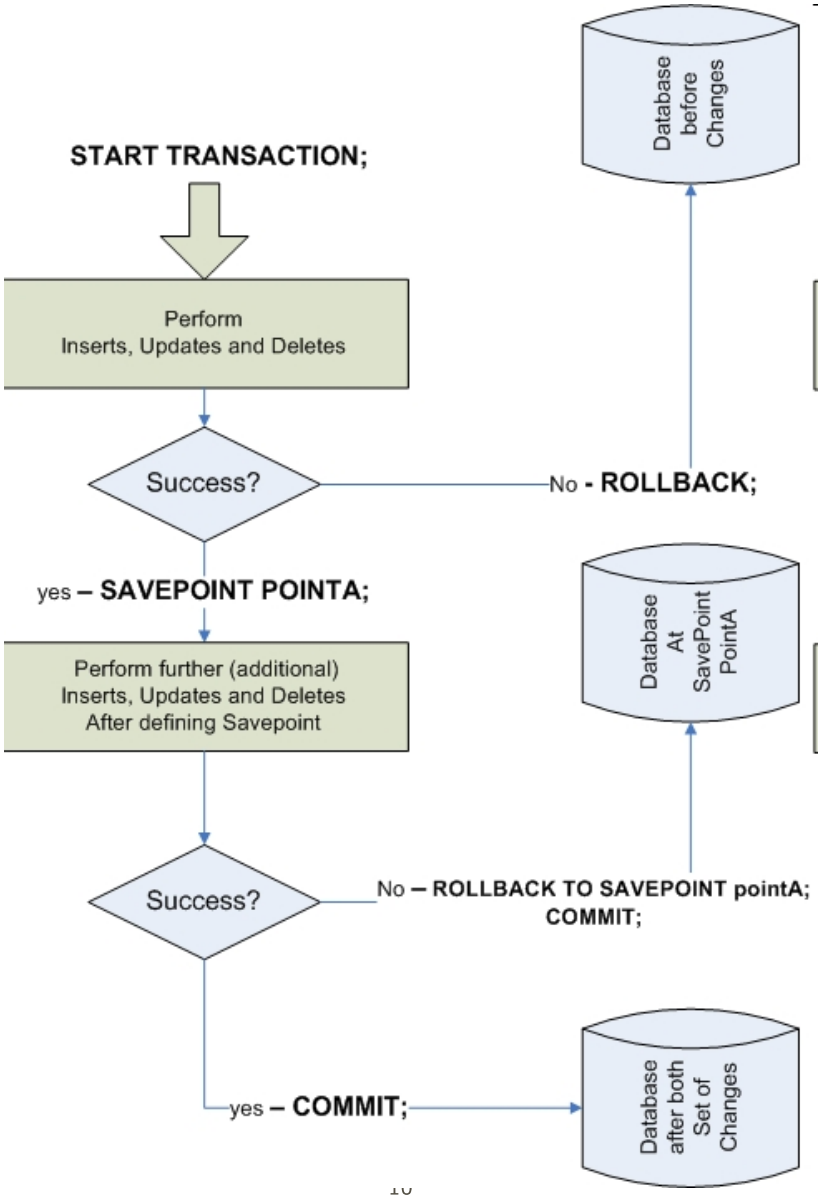
```
-- 4. Insert order line items
INSERT INTO orderdetails(orderNumber,
                         productCode,
                         quantityOrdered,
                         priceEach,
                         orderLineNumber)
VALUES(@orderNumber,'S18_1749', 30, '136', 1),
      (@orderNumber,'S18_2248', 50, '55.09', 2);

-- 5. commit changes
COMMIT;
```

More on Transaction

- The SAVEPOINT command defines a marker in a transaction
- The ROLLBACK TO SAVEPOINT command allows rolling back to a previous marker

Transaction Savepoint



MySQL Transaction

- MySQL **InnoDB storage engine** supports transactions
- MyISAM does not support transactions

Transaction Properties

- *ACID* (Atomic, Consistent, Isolated, Durable)

https://www.ibm.com/docs/en/cics-ts/5.4?topic=processing-acid-properties-transactions

Transaction in programming languages

- http://www.mysqltutorial.org/mysql-transaction.aspx
- http://www.mysqltutorial.org/mysql-jdbc-transaction/

Transaction in Distributed System

- The two-phase commit protocol provides atomicity for distributed transactions to ensure that each participant in the transaction agrees on whether the transaction should be committed or not

Today's Overview

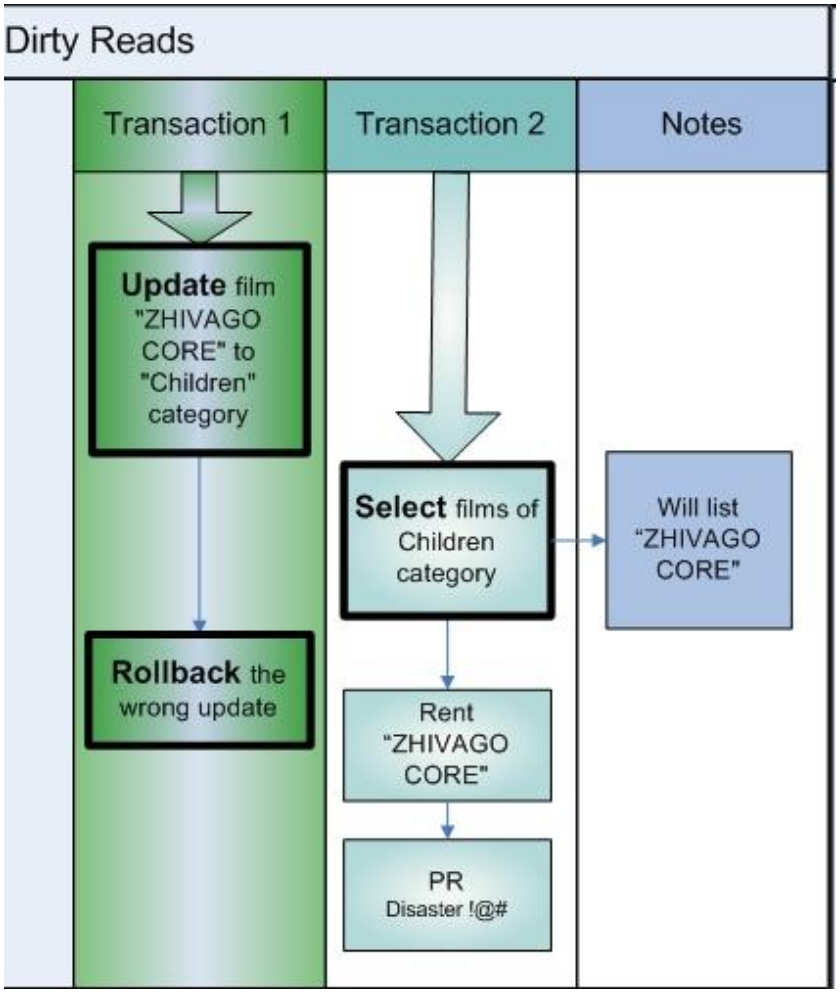
1	• Introduction
2	• Commit-Rollback
3	• Isolation Levels
4	• MVCC

Transaction isolation levels

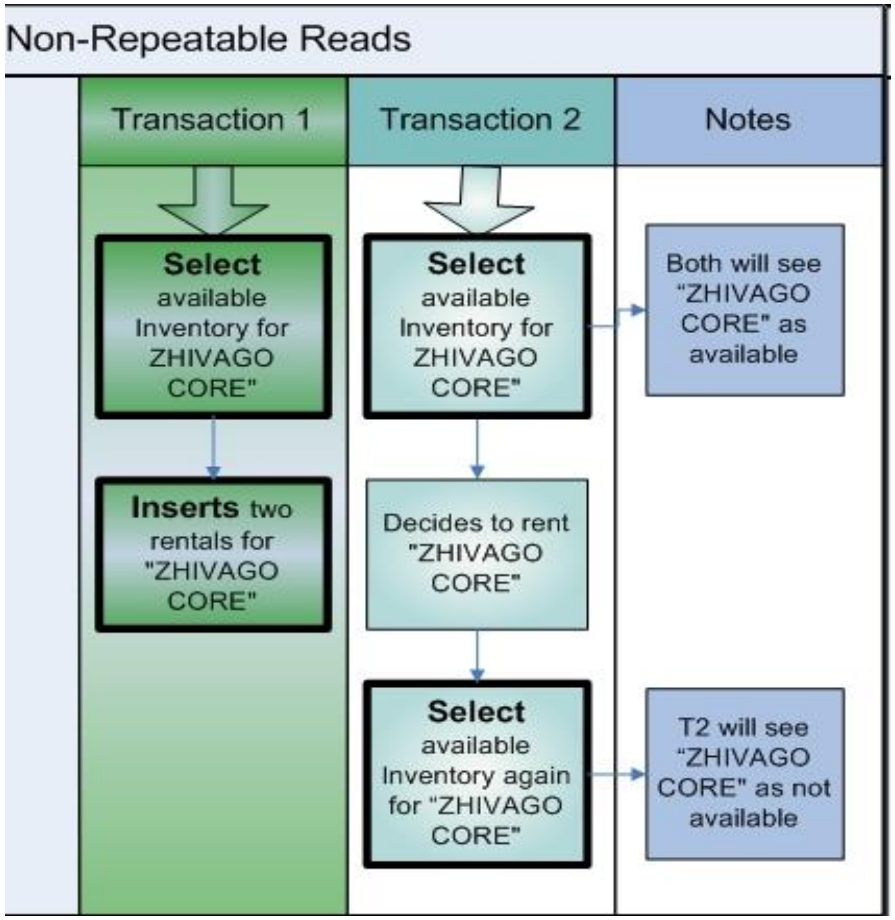
- In databases, multiple users can view and edit data simultaneously
- Concurrent operations can result in inconsistent and inaccurate data

Dirty Reads

- Dirty read occurs when a transaction is allowed to read data being updated by another uncommitted transaction



Nonrepeatable read



Phantom Reads

- Occurs when within a transaction, two identical queries return different sets of rows when executed

MySQL Isolation Levels

Isolation Levels	Usage	Dirty reads	Nonrepeatable reads	Phantom reads
READ UNCOMMITTED	Use in situations where accuracy is not so important	Yes	Yes	Yes
READ COMMITTED	Prevent *dirty reads*	No	Yes	Yes
REPEATABLE-READ	Default Isolation level in MySQL	No	No	Yes
SERIALIZABLE	Transactions are completely isolated from each other and are processed sequentially	No	No	No

Choose Isolation Level

- To decide the isolation level to use, it is necessary to balance between the required level of accuracy of the retrieved data and the processing performance
- *The higher the isolation level, the more impact it has on performance*

Setting Isolation level in MySQL

- SET SESSION *tx_isolation*='READ-COMMITTED';
- SELECT *@@global.tx_isolation*, *@@session.tx_isolation*

Mysql8 has renamed tx_isolation to transaction_isolation.

Multi-versioned concurrency control (MVCC)

- MySQL uses MVCC in transaction management
- MVCC provides each connection to the database with a snapshot of the data
- Any changes will not be visible to other users until the transaction is committed

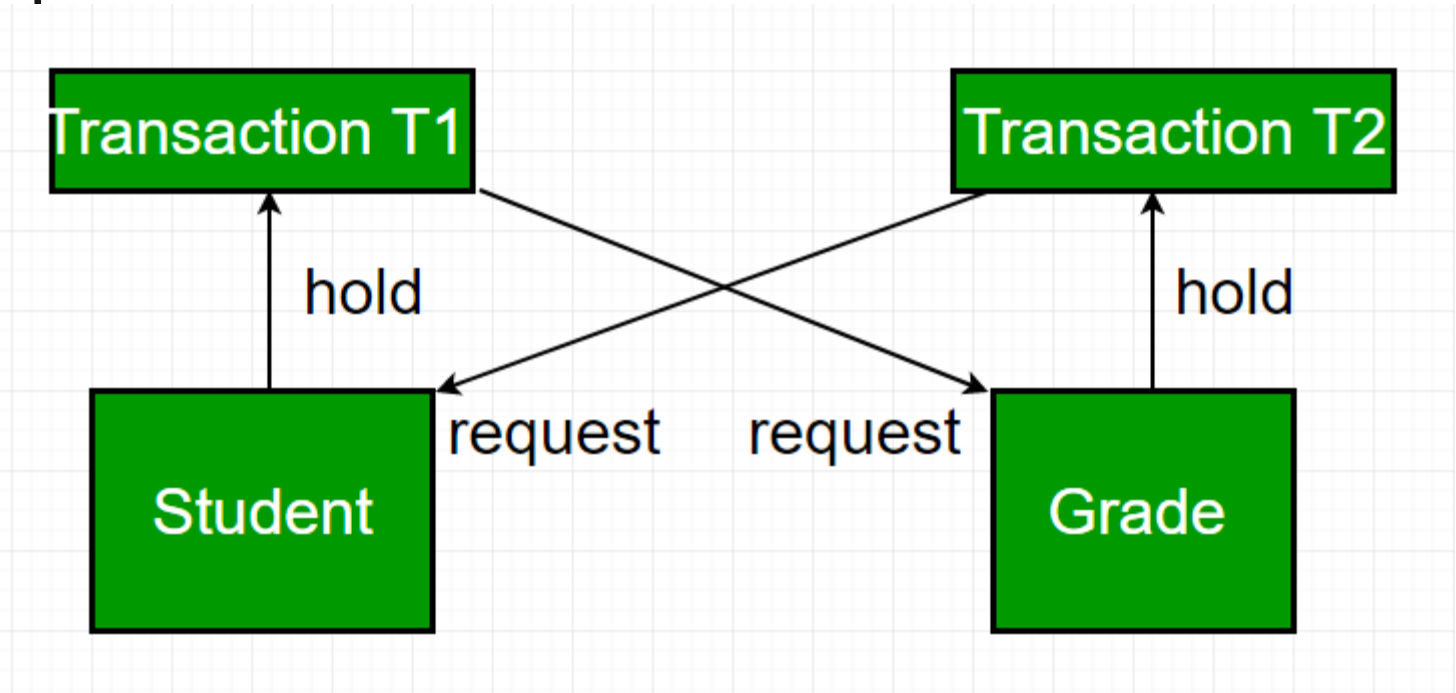
http://en.wikipedia.org/wiki/Multiversion_concurrency_control

Overview

1	• Introduction
2	• Commit-Rollback
3	• Isolation Levels
4	• Deadlock

Deadlock i

- A deadlock happens when two or more transactions are mutually holding and requesting locks on the same resources, creating a cycle of dependencies



Deadlock

- To solve this problem, database systems implement various forms of deadlock detection and timeout
- The InnoDB storage engine will notice circular dependencies and return an error instantly

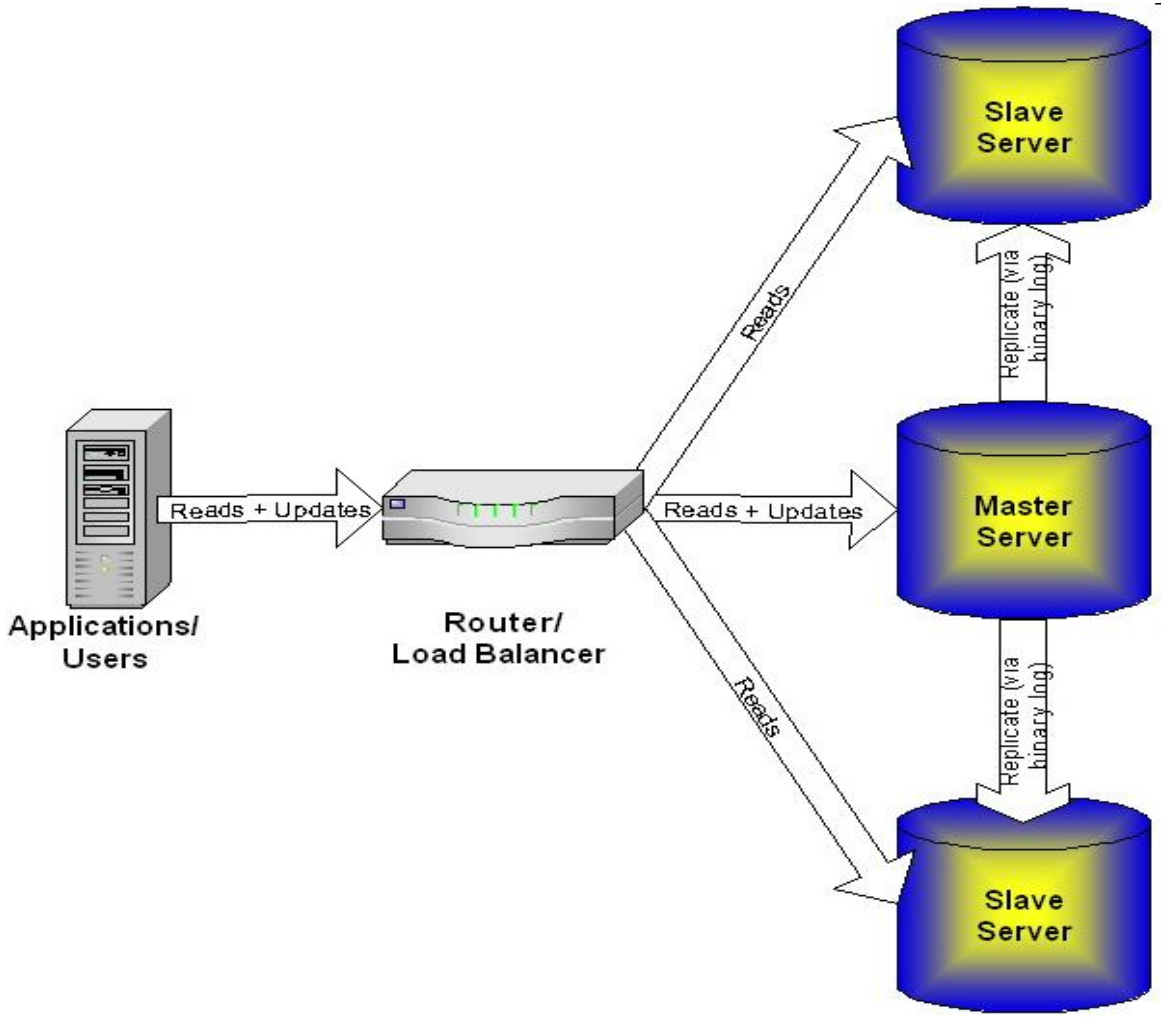
*SELECT * from performance_schema.data_locks;*

Non Transactional engine (MyISAM)

- To lock tables that do not support transactions, use the LOCK TABLES statement
- Once you have completed updating the tables, you need to use the UNLOCK TABLES statement to release the tables

Replication

Database Replication



Database Replication

- Updates to one database are automatically replicated to other replicas
- Updates to the master server (*primary*)
- Queries that read data can be assigned to the master server or slave servers (*replicas*)

Replication Advantages

- Availability: Replicas can be used as "hot" backups; if the master database is not available, the replicas can take over as the master until the error is resolved
- Backups: Replicas can be used as backups, which can be used to perform long backups without locking the master
- Load Balancing: Read queries can be distributed to different replicas

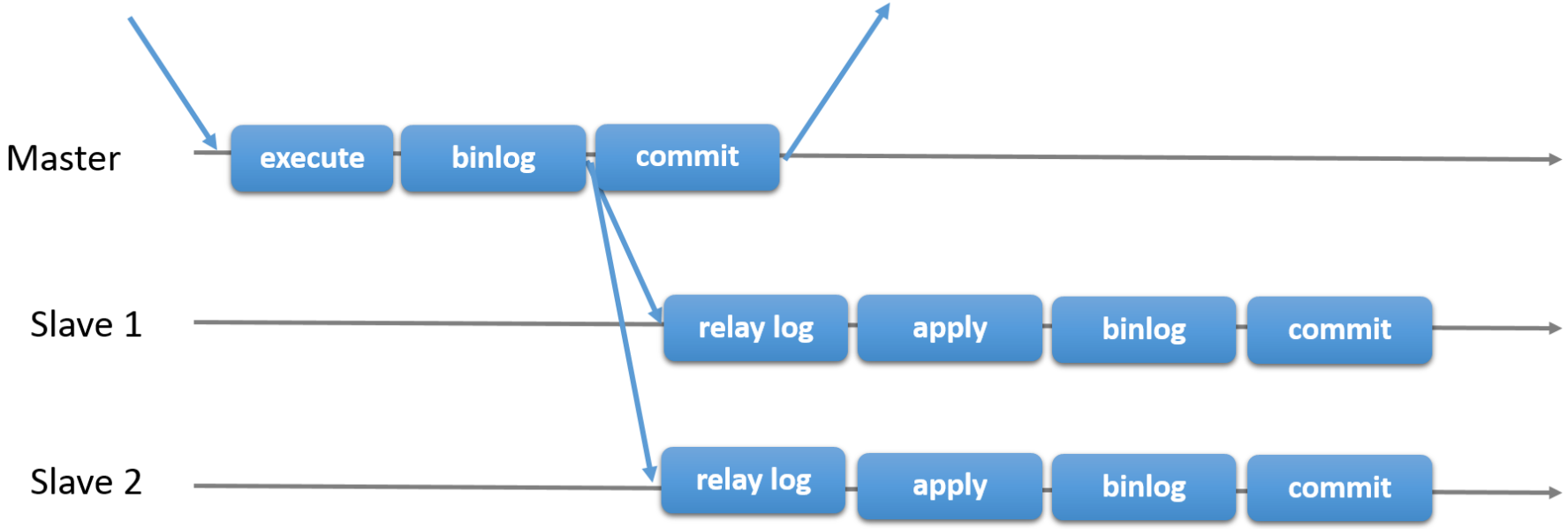
Asynchronous Replication in MySQL

- In MySQL, replication is a one-way, asynchronous process
- The master database will store all updates in a binary log file. Updates in the log file are then used to synchronize the database on the slave server
- Slave servers connect to the master server to read log files and update changes

LOG Formats

- **Statement-based binary logging**: the master writes SQL statements to the binary log
- **Row-based logging**: the master writes events to the binary log that indicate how individual table rows are changed
- Depending on certain statements, and also the storage engine being used, the log is automatically switched to row-based in particular cases

MySQL Asynchronous Replication



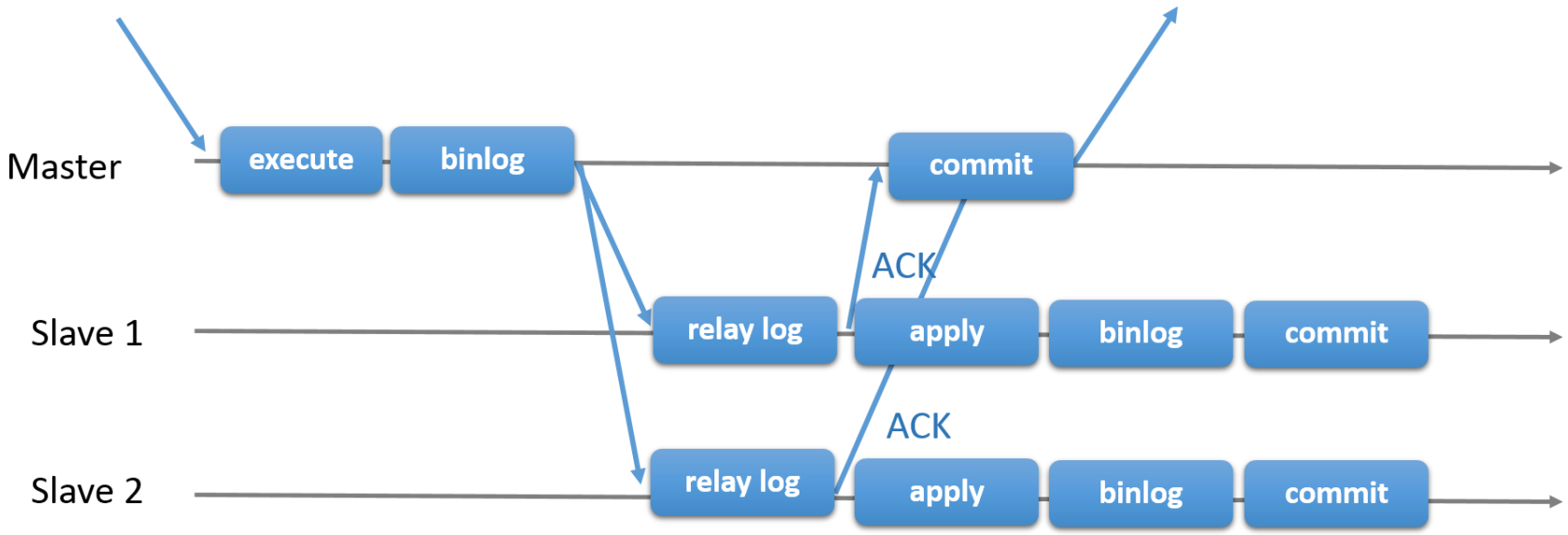
Config Slave server

```
CHANGE MASTER TO MASTER_HOST='192.168.0.100',
MASTER_USER='slave_user',
MASTER_PASSWORD='<some_password>',
MASTER_LOG_FILE='mysql-bin.006',
MASTER_LOG_POS=183;
```

Config Slave server

- *START SLAVE;*
- SHOW SLAVE STATUS

MySQL Semisynchronous Replication

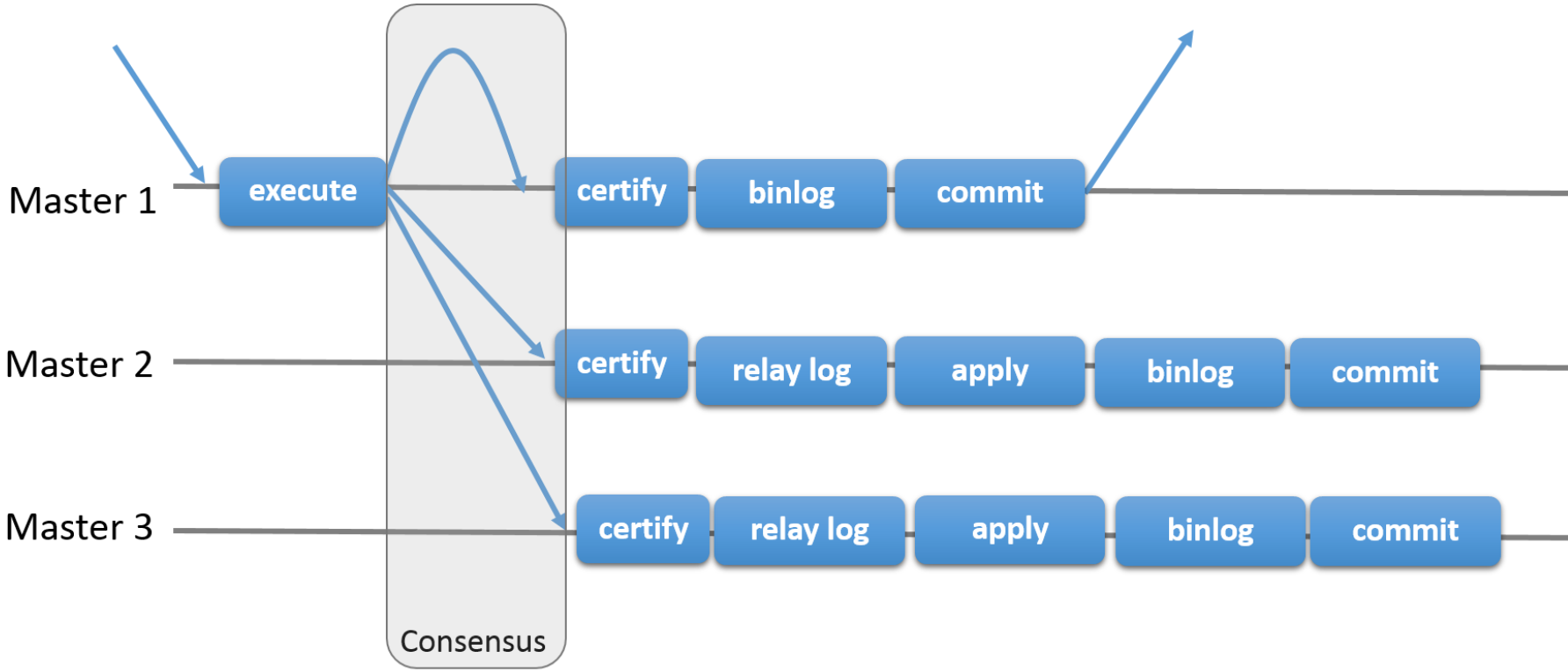


Semisynchronous Replication

- While the master is blocking (waiting for acknowledgment from a slave), it does not return to the session that performed the transaction
- When the block ends, the master returns to the session, which then can proceed to execute other statements

MySQL Group Replication

- Implements a multi-master update everywhere replication protocol.



Some Multi-Master solutions for MySQL

PERCONA
XtraDB Cluster

GALERA CLUSTER

MySQL Group Replication

THANKS YOU