Новые возможности репликации MySQL 5.6

Konstantin Osipov

Moscow
MySQL User Group

А также:
Luís Soares
Sven Sandberg

Содержание

- История репликации в MySQL
- Причины введения GTID
- Устройство и использование GTID
- Практика повышения отказоустойчивости

Введение

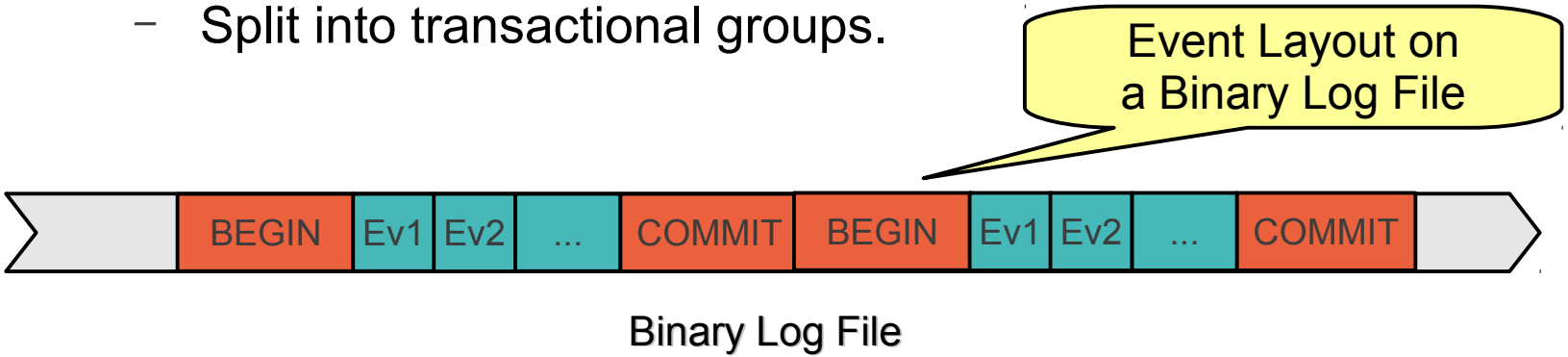
Компоненты репликации в MySQL

- MySQL **Master** сервер
  - Изменяет данные
  - Логирует изменения (**Events**) в файл (**Binary Log**)
- MySQL **Slave** сервер
  - Получает изменения с мастера
  - Проигрывает изменения на данных slave server'a

Введение

MySQL Replication Components: Binary Log

- The **Binary Log**
  - File based log that records the changes on the master.
  - Statement or Row based format (may be intermixed).
  - Split into transactional groups.



Введение

MySQL Replication Components: Binary Log

Under the Hood

Binary log files: mysql-bin.000001, mysql-bin.000002, …

- данные, **events**.

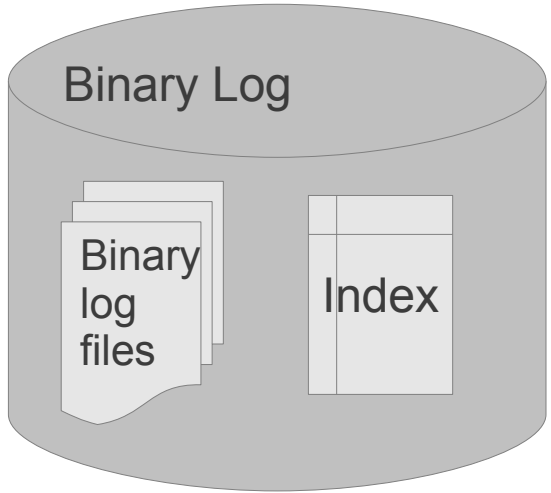
Index: mysql-bin.index

- Индекс по бинлогу – для быстрого поиска по

Log coordinate:

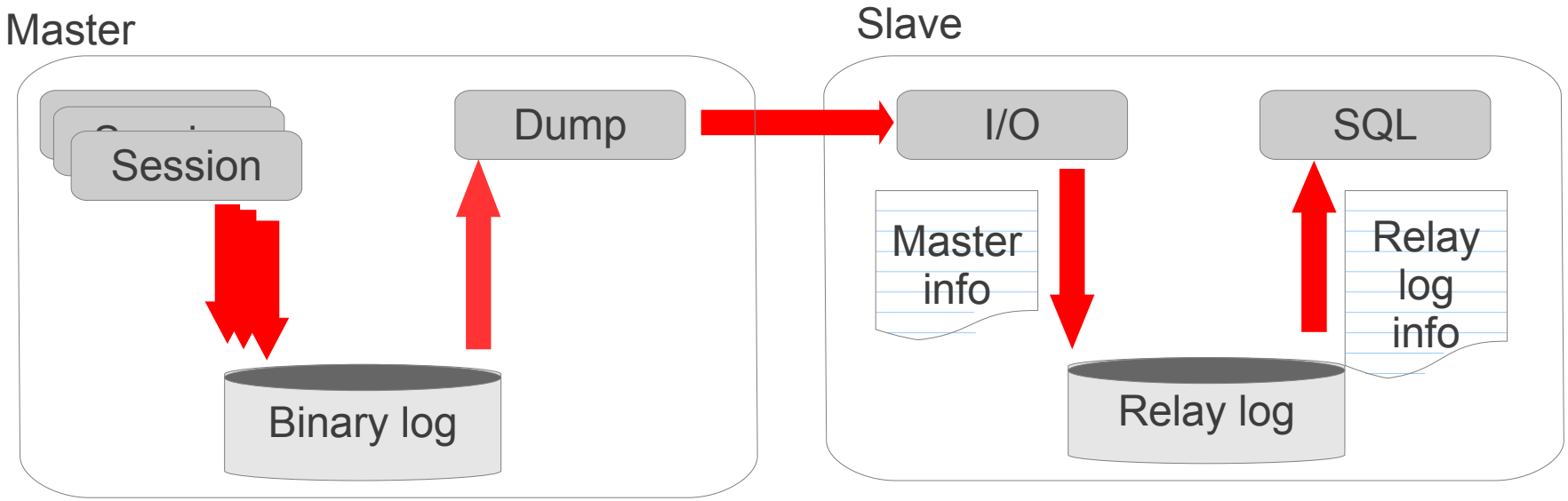
- binlog file name + event offset in the file (3.23.15+)

- Global Transaction Identifiers (5.6+)



Архитектура репликации в MySQL

MySQL Replication Architecture



- I/O и SQL Thread Replication Metadata хранится в файлах.
- **В MySQL 5.6 метаданные могут храниться в InnoDB.**

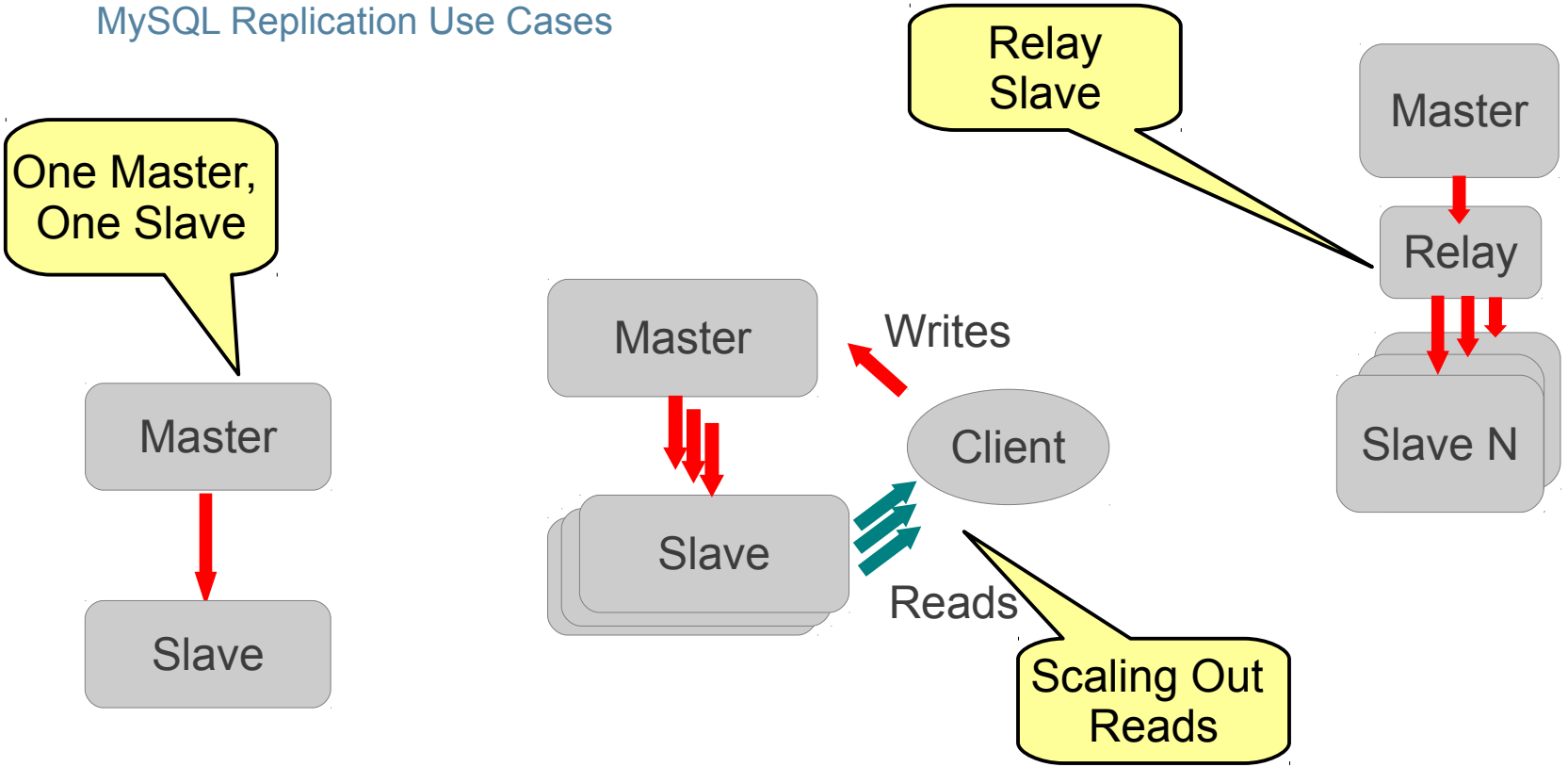
Background

Changes Propagation

- **Asynchronous Replication (MySQL 3.23.15+)**
  - Transactions are committed and externalized without interaction with the replication layer.
  - Events are propagated after the commit operation is acknowledged.
  - Faster but vulnerable to lost updates on server crashes and inconsistency.
  - Built into the server.
- **Semi-synchronous Replication (MySQL 5.5+)**
  - Master commits transaction but waits until one slave acknowledges having received and stored the event before replying to the client.

Введение

MySQL Replication Use Cases



GTID: мотивация

Motivation: Seamless Failover

- Сервера иногда крэшатся (сбой оборудования, баг, метеорит...)
- Slave должен быть “повышен” в мастера (promotion)
- Способ индексации событий в бинлоге является КЛЮЧЕВЫМ!
  - MySQL 3.23+: FILENAME+OFFSET
    - Локален для сервера, абсолютен
  - MySQL 5.6+: TRANSACTION IDENTIFIERS
    - Глобальный, логический id, генерируется автоматически

GTID: внутреннее устройство

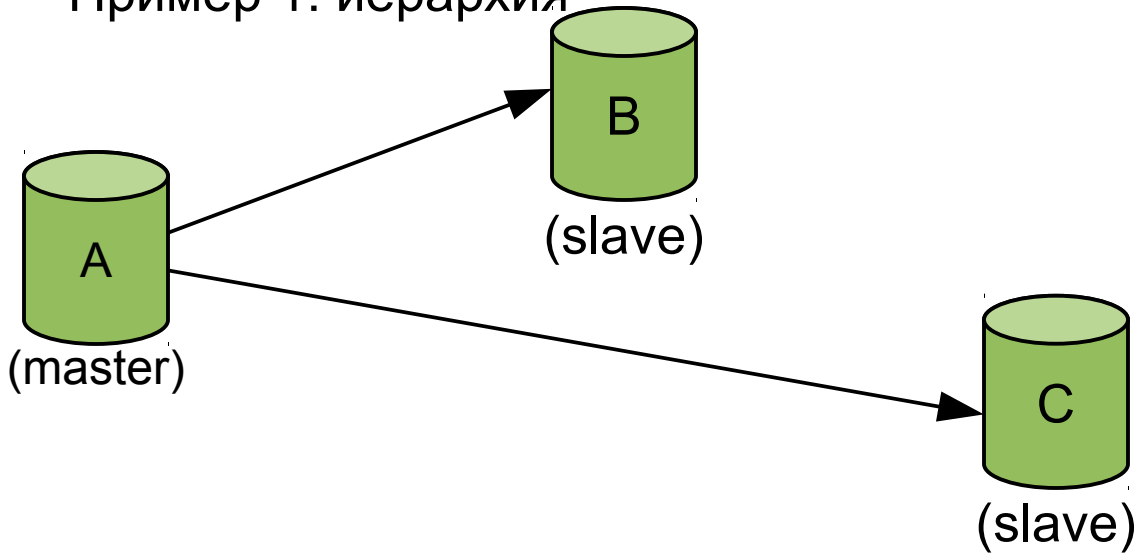
Global Transaction Identifiers

- Generate a global transaction identifier on commit:
  - server_uuid:number
    `a61678ba-4889-4279-9e58-45ba840af334:1`
  - server_uuid identifies the server; globally unique
  - number is incremented by 1 for each transaction on this server

GTID: автоматический failover

Motivation: Seamless Failover

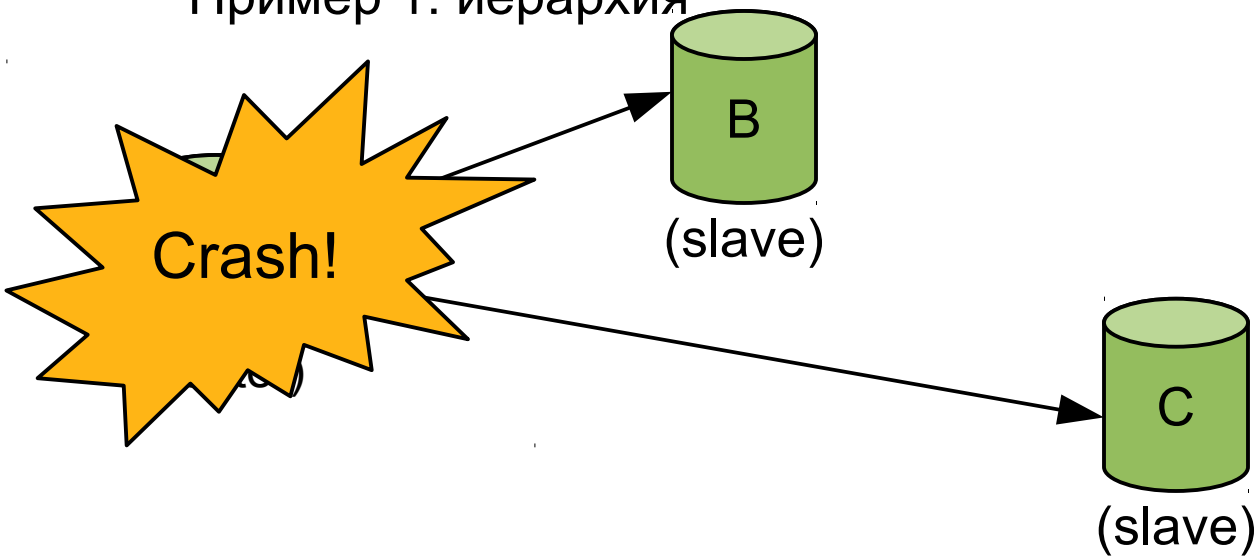
- Процедура failover при выходе из строя мастера
- Пример 1: иерархия



GTID: автоматический failover

Motivation: Seamless Failover

- Enable fail-over if master crashes
- Пример 1: иерархия

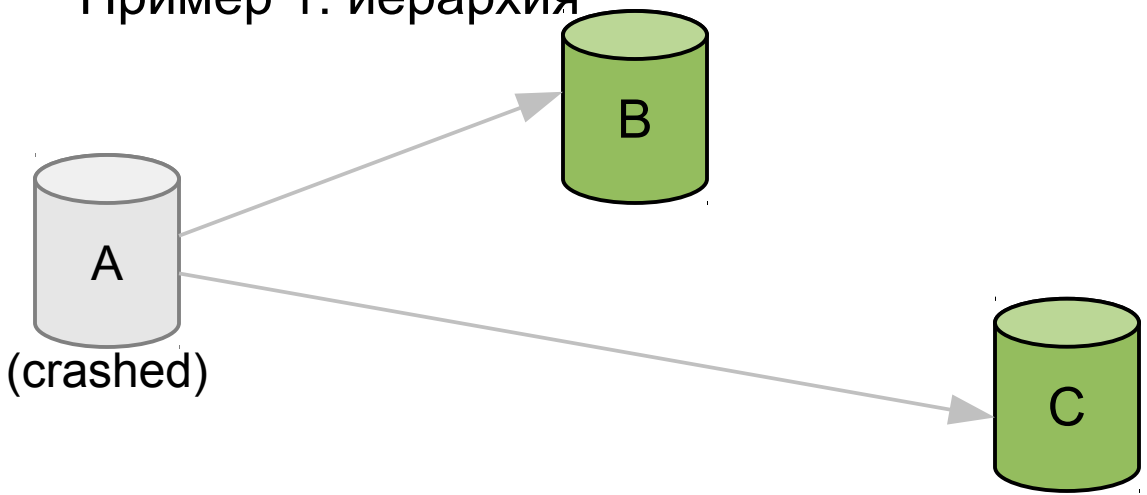


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия

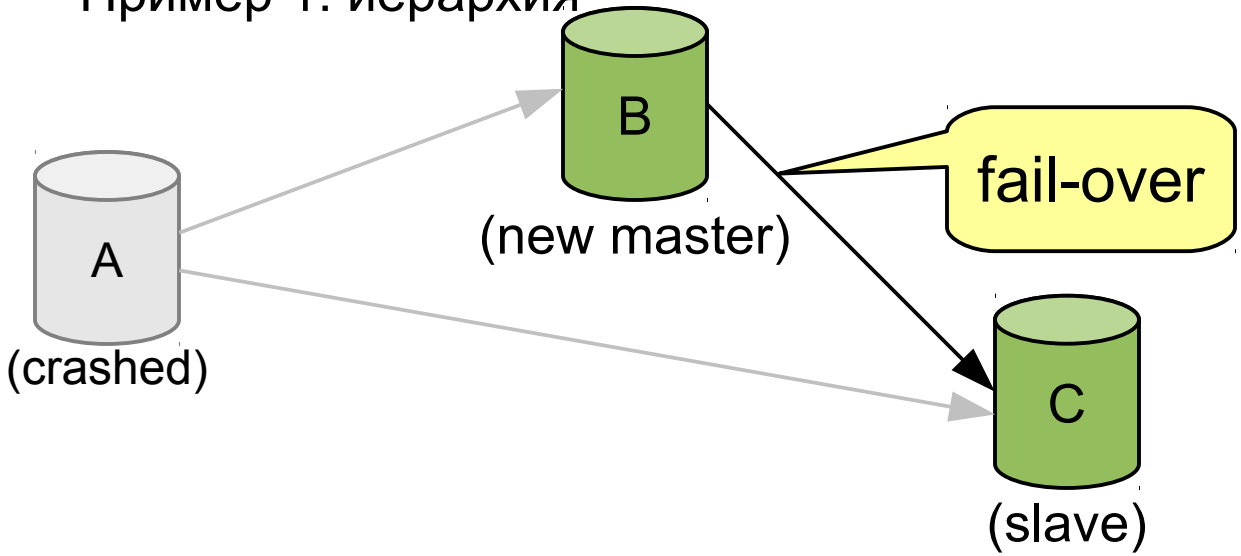


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия

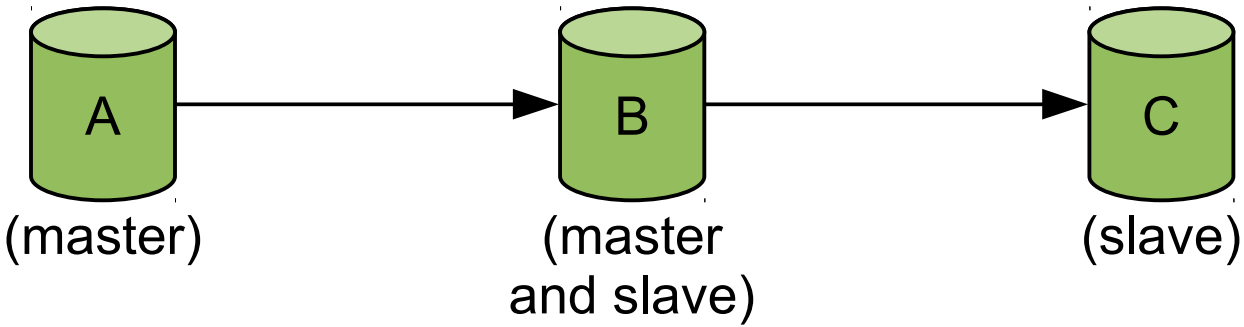


GTID: автоматический fail-over

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия
- Пример 2: цепочка



GTID: автоматический failover

Motivation: Seamless Failover

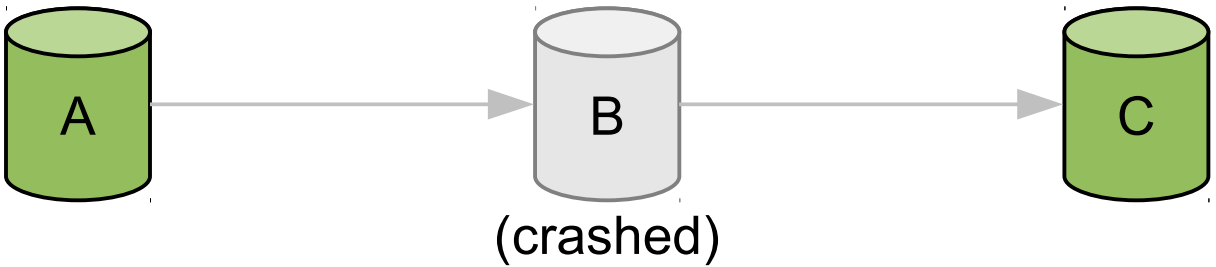
- Enable fail-over if master crashes
- Пример 1: иерархия
- Пример 2: цепочка

GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия
- Пример 2: цепочка

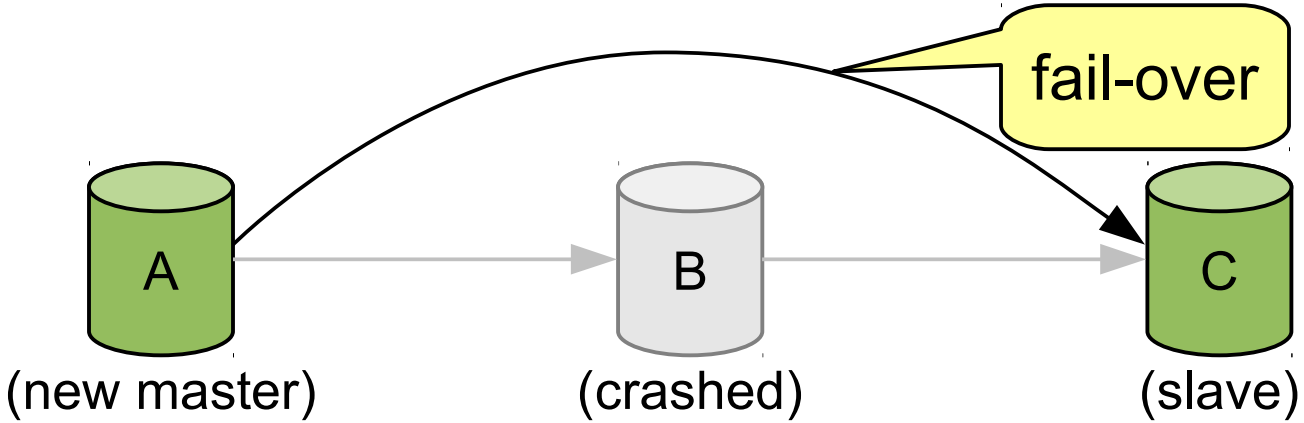


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия
- Пример 2: цепочка

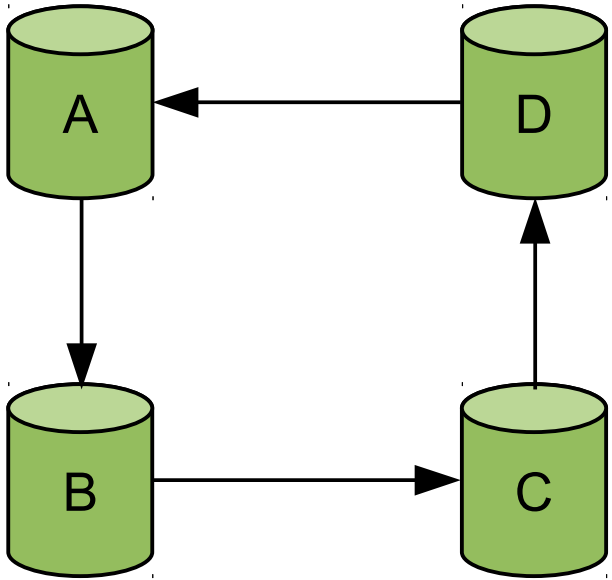


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия
- Пример 2: цепочка
- Пример 3: кольцо

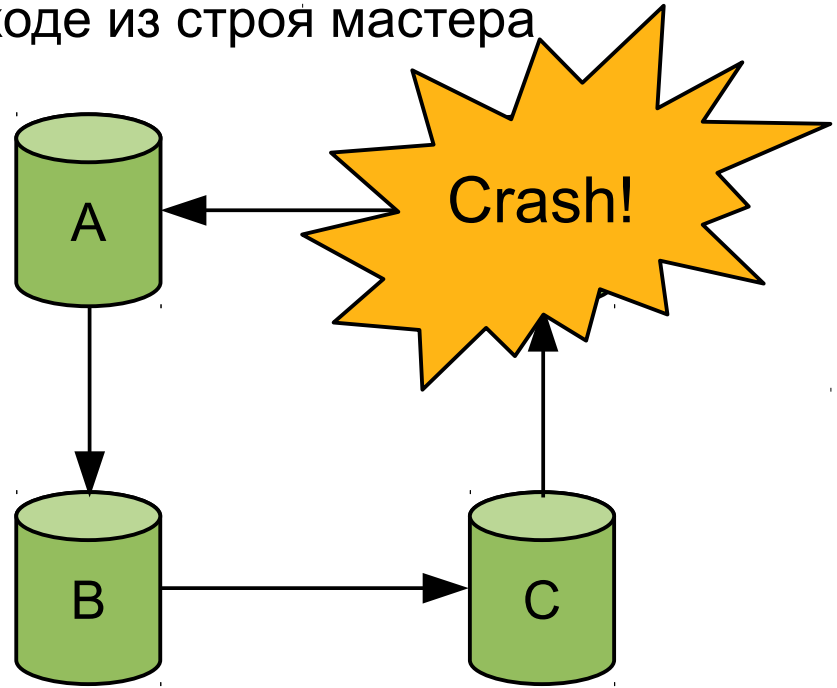


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия
- Пример 2: цепочка
- Пример 3: кольцо

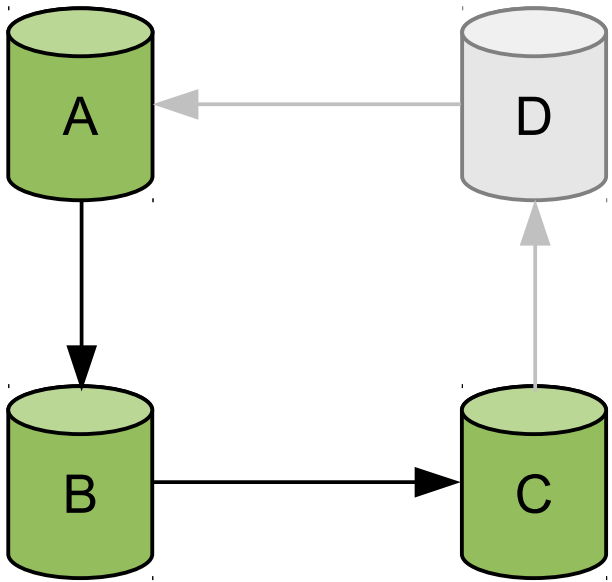


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

- Пример 1: иерархия
- Пример 2: цепочка
- Пример 3: кольцо

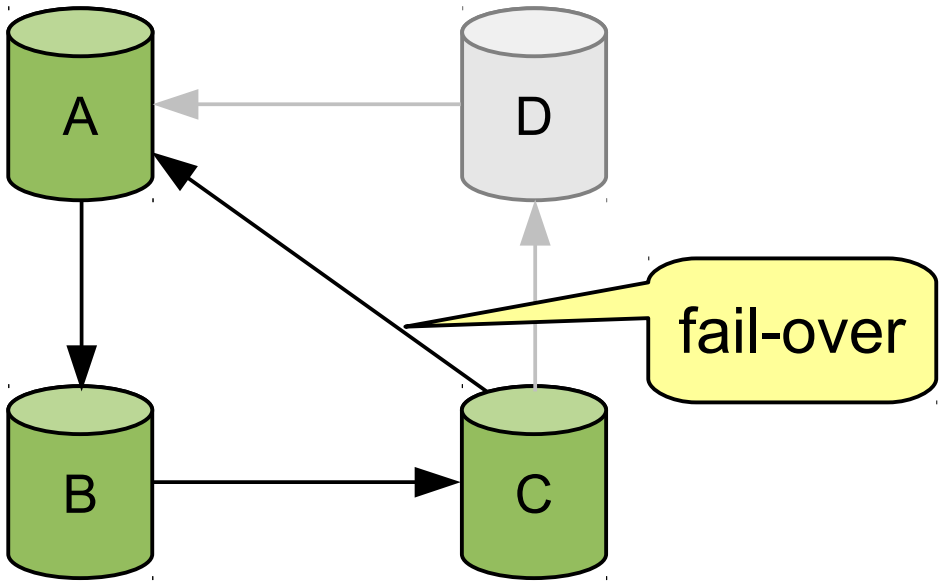


GTID: автоматический failover

Motivation: Seamless Failover

Процедура failover при выходе из строя мастера

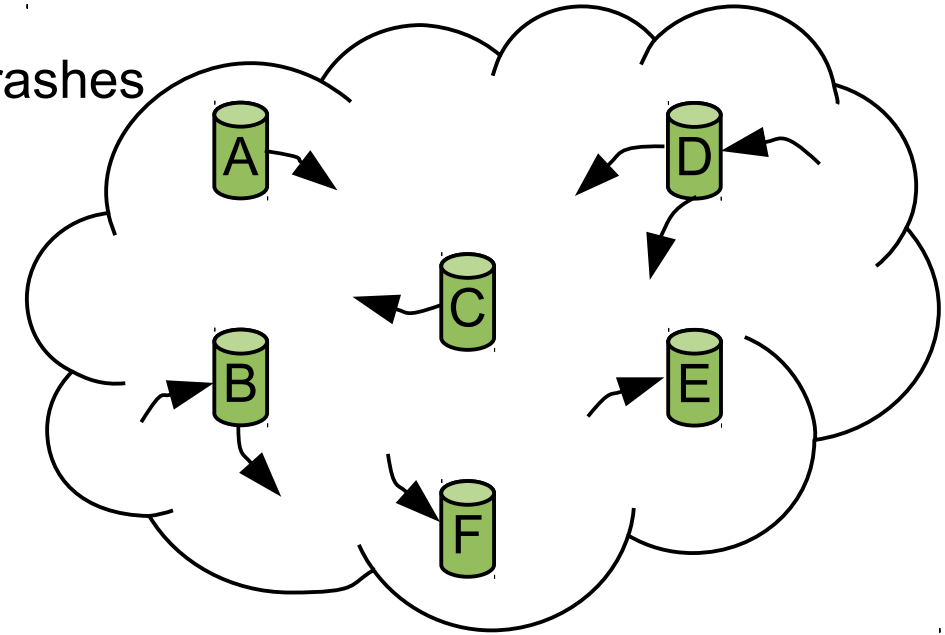
- Пример 1: иерархия
- Пример 2: цепочка
- Пример 3: кольцо



GTID: автоматический failover

Motivation: Seamless Failover

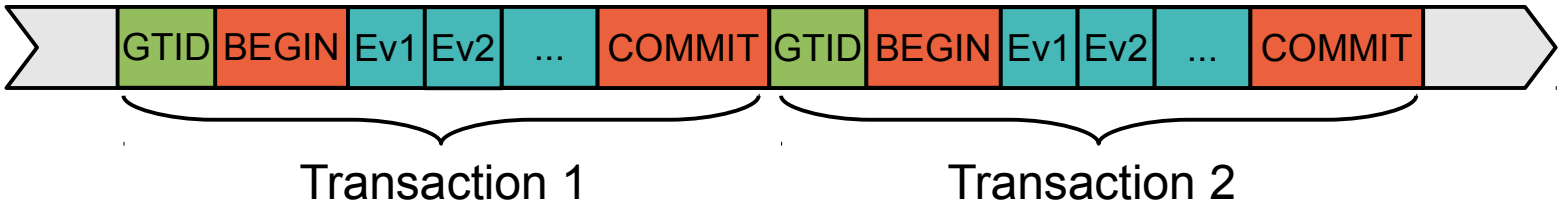
- Enable fail-over if master crashes
- Пример 1: иерархия
- Пример 2: цепочка
- Пример 3: кольцо
- Пример 4: произвольная топология



GTID и binlog

Global Transaction Identifiers

- global transaction identifier создаётся в момент commita:
  - server_uuid:number
    `a61678ba-4889-4279-9e58-45ba840af334:1`
  - server_uuid identifies the server; globally unique
  - number is incremented by 1 for each transaction on this server
- Write GTID to binary log



Global Transaction Identifiers

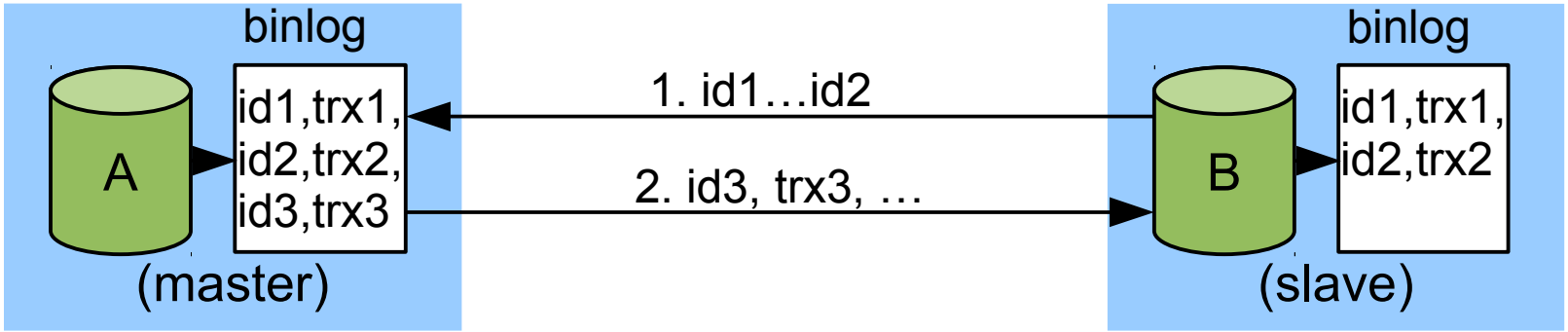
Global Transaction Identifiers

- Generate a global transaction identifier on commit:
  - server_uuid:number
    `a61678ba-4889-4279-9e58-45ba840af334:1`
  - server_uuid identifies the server; globally unique
  - number is incremented by 1 for each transaction on this server
- Write GTID to binary log
- Preserve GTID when slave re-executes transaction

GTID: новый протокол репликации

New protocol

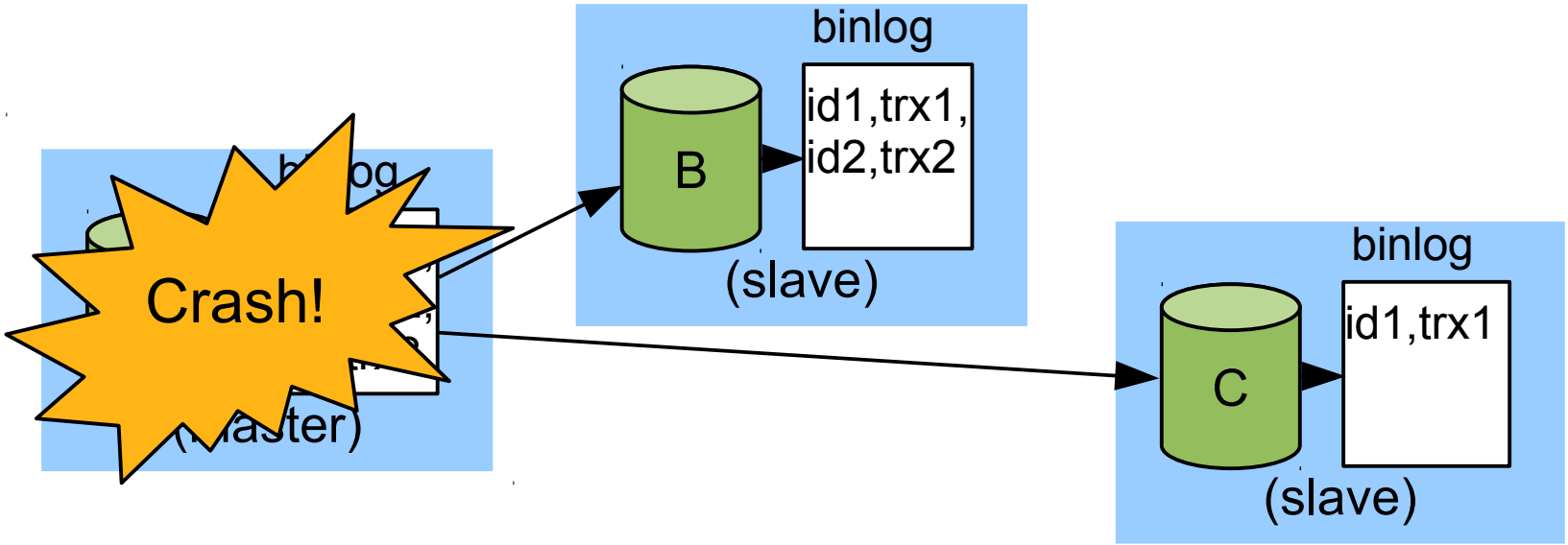
- Реплика посылает мастеру:
  диапазон транзакций, которые были выполнены на реплике
- Мастер посылает реплике все остальные транзакции



GTID: новый протокол репликации

New protocol used in failover

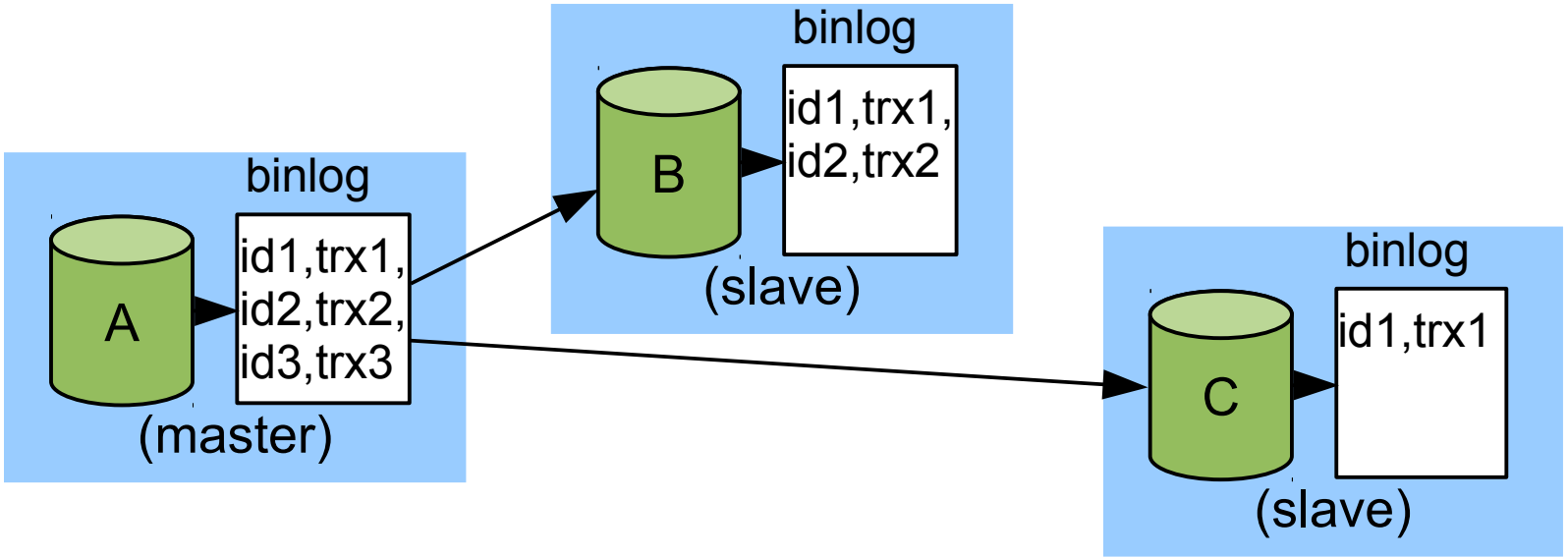
- Пример 1: иерархия



GTID: новый протокол репликации

New protocol used in failover

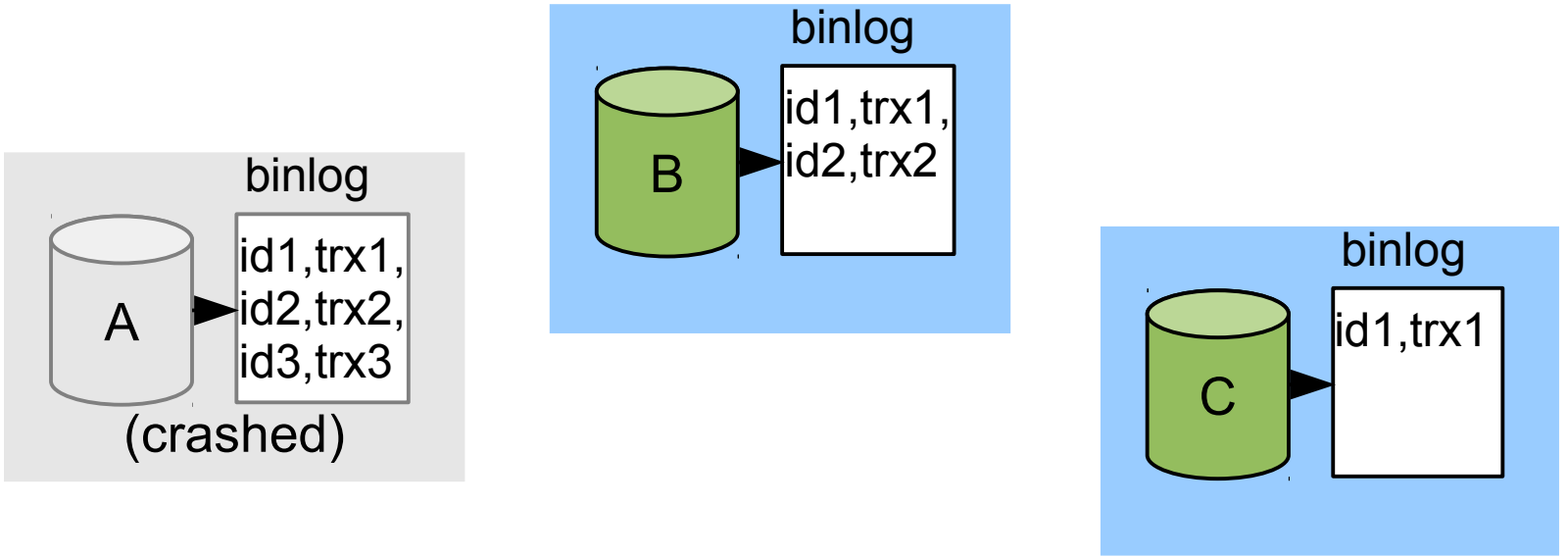
- Пример 1: иерархия



GTID: новый протокол репликации

New protocol used in failover

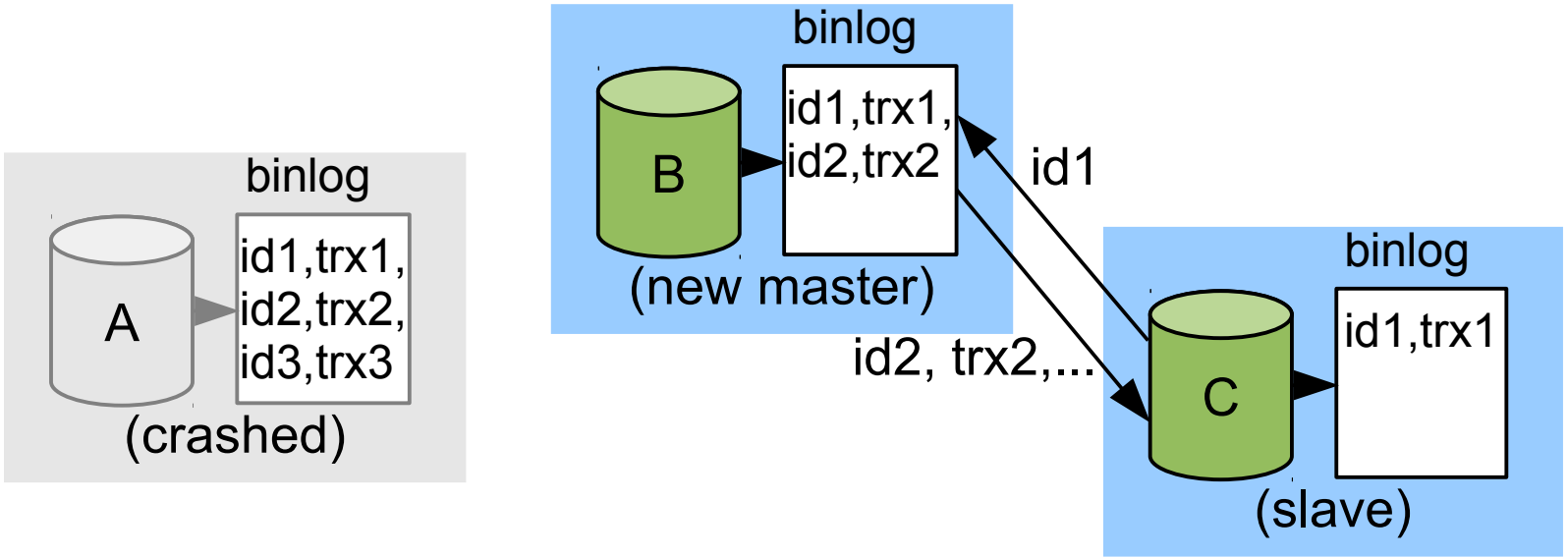
- Пример 1: иерархия



GTID: новый протокол репликации

New protocol used in failover

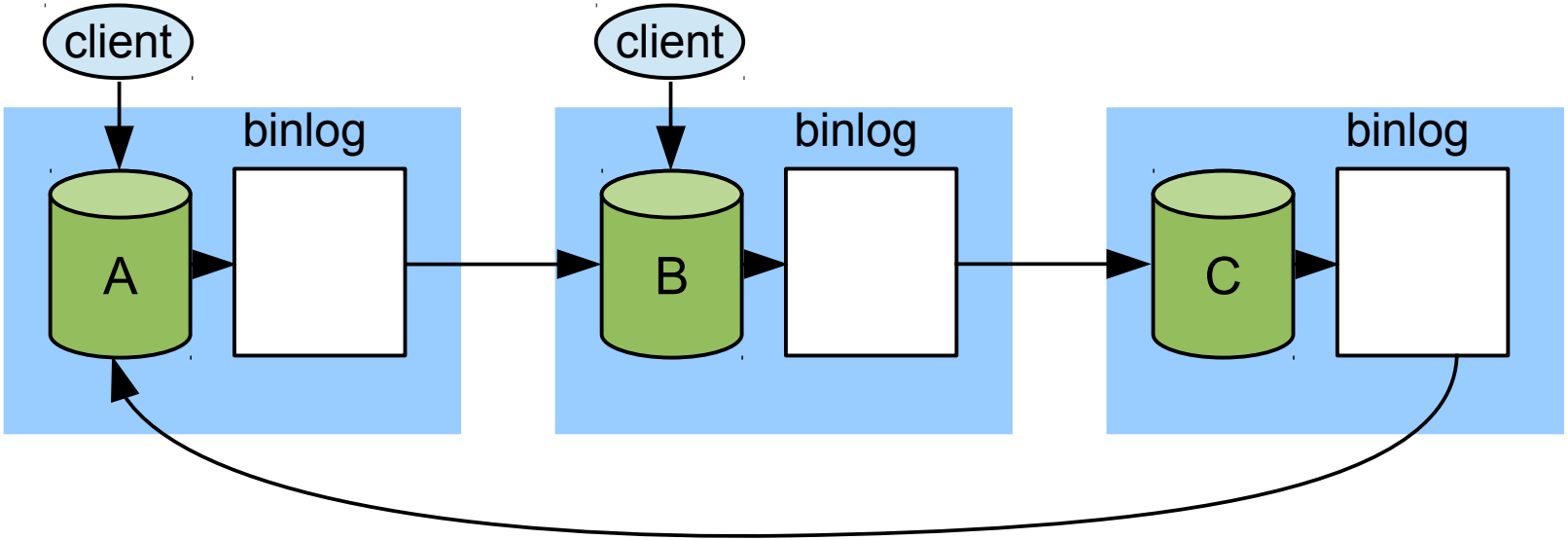
- Пример 1: иерархия



GTID: новый протокол репликации

New protocol

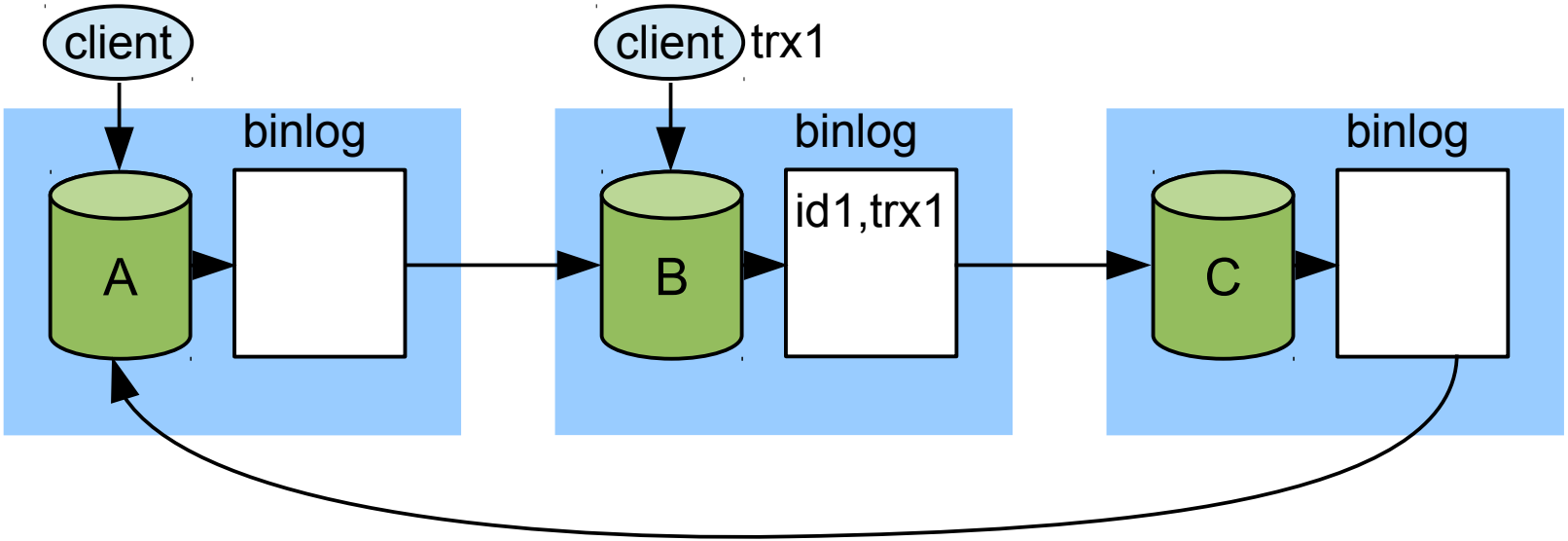
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый протокол репликации

New protocol

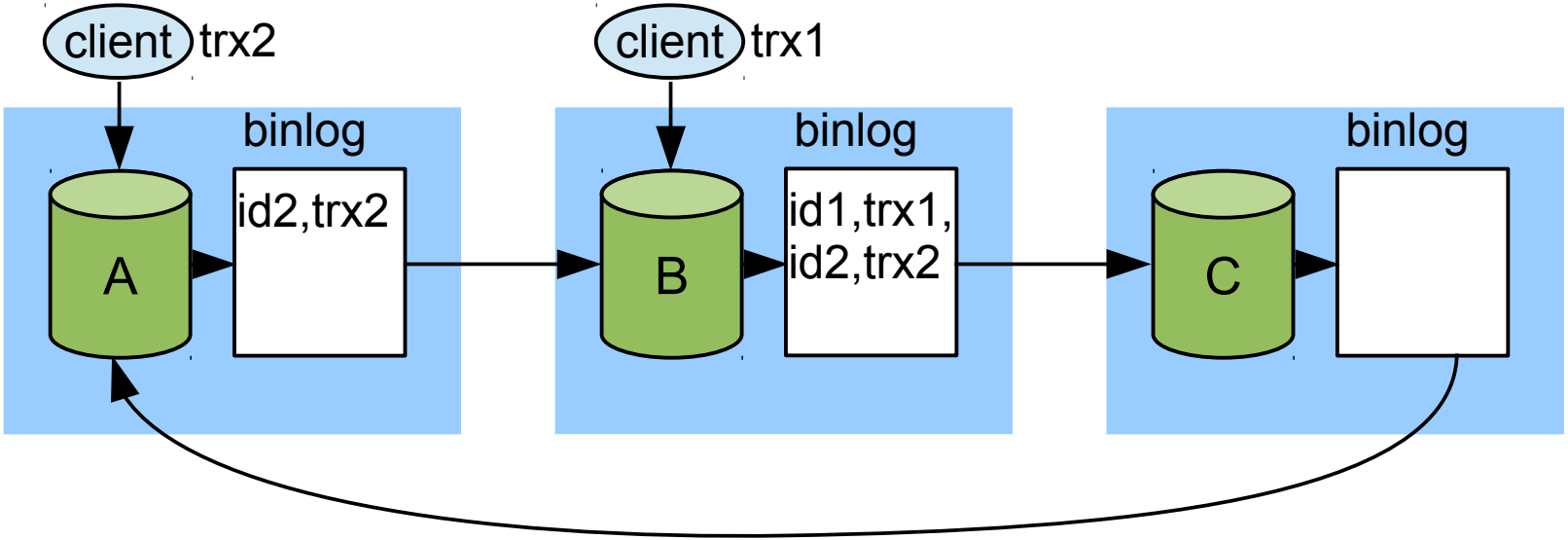
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый протокол репликации

New protocol

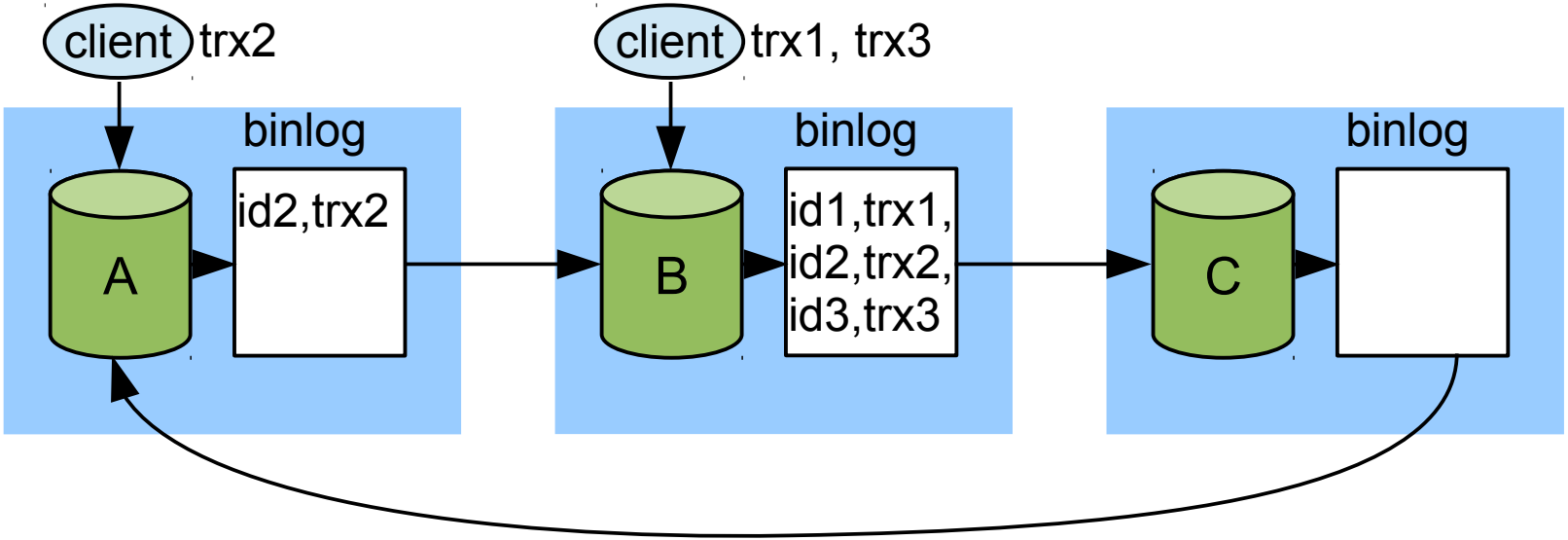
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый протокол репликации

New protocol

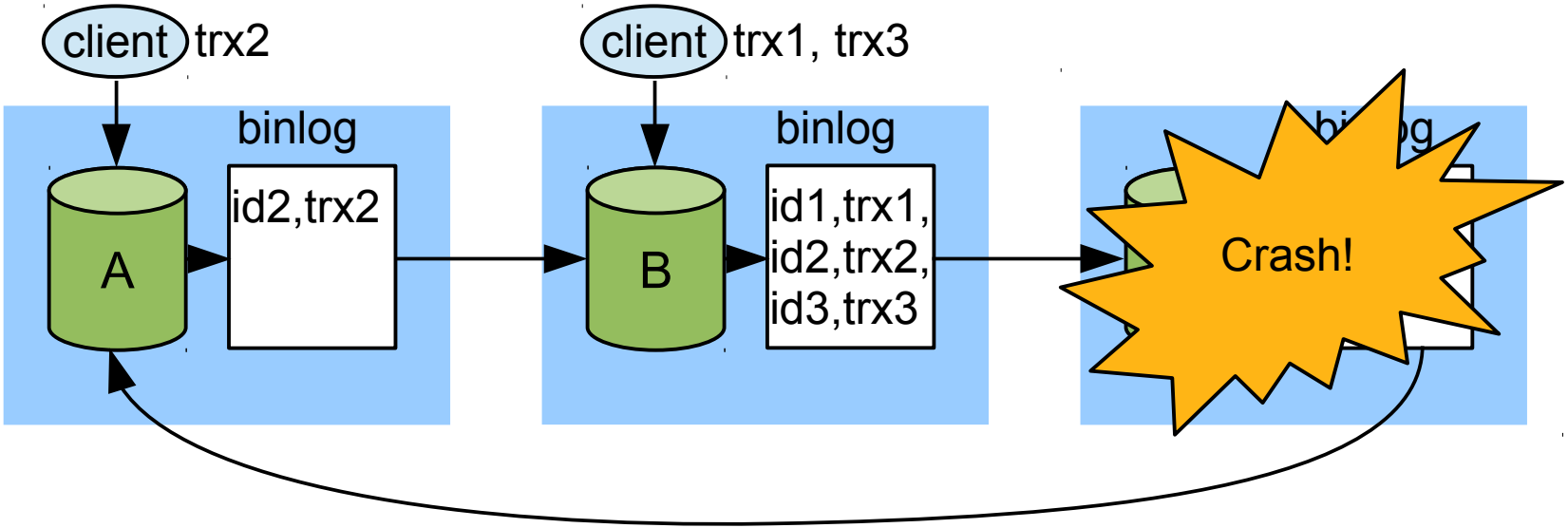
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый протокол репликации

New protocol

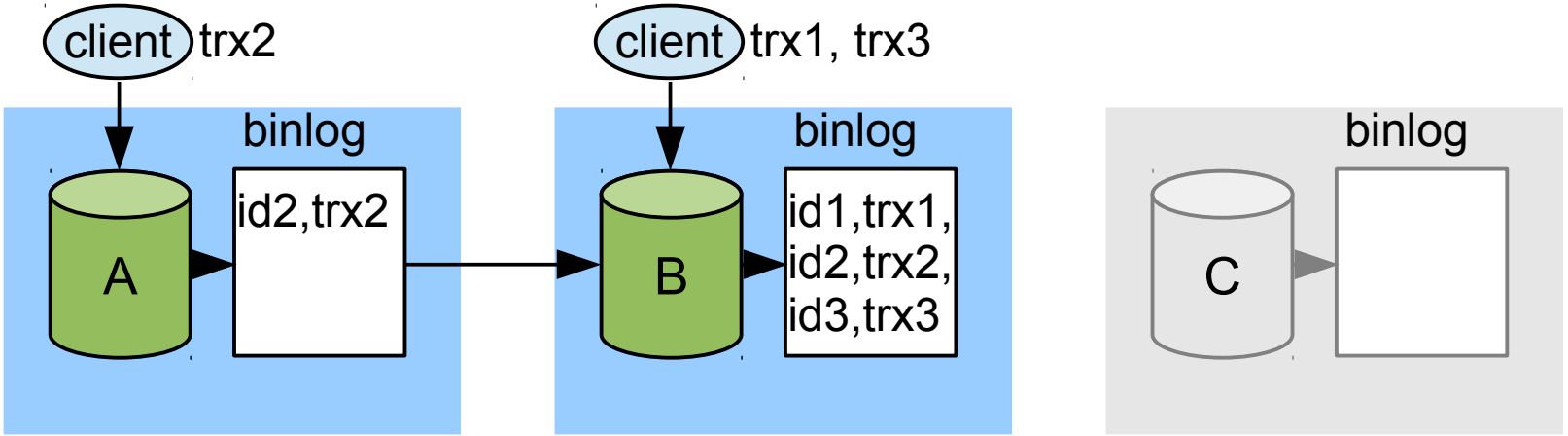
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый протокол репликации

New protocol

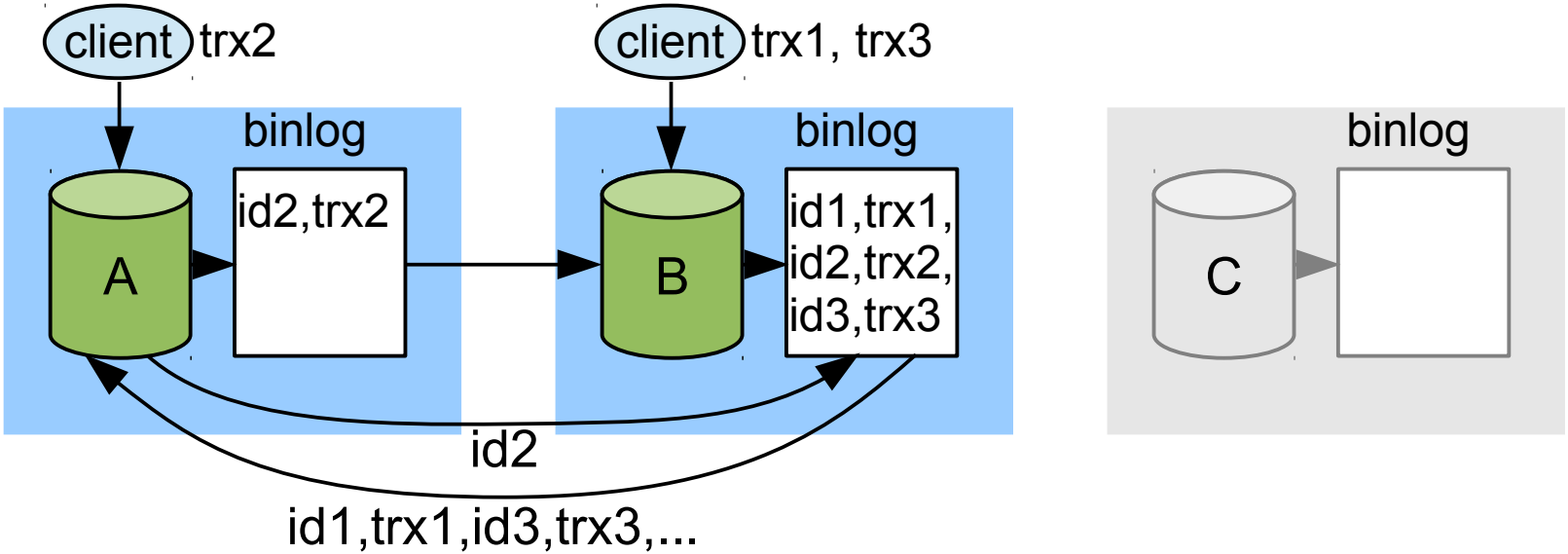
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый протокол репликации

New protocol

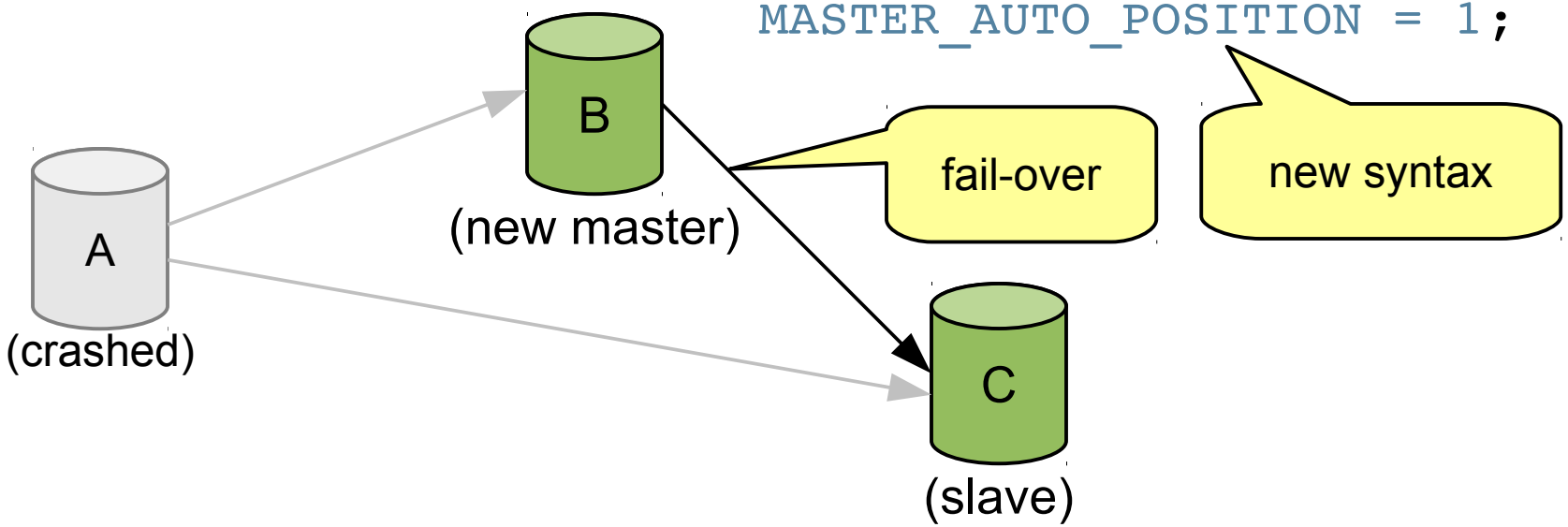
- Пример 1: иерархия
- Пример 2: кольцо



GTID: новый синтаксис CHANGE MASTER

How to use: New syntax

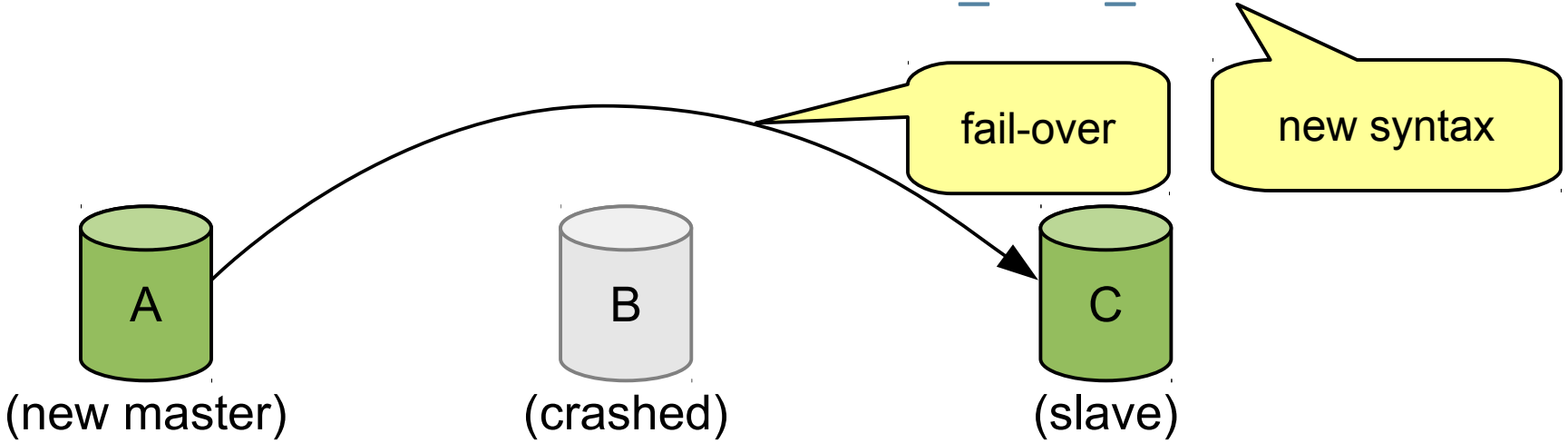
```
server_C> CHANGE MASTER TO MASTER_HOST = 'server_B',
          MASTER_AUTO_POSITION = 1;
```



GTID: новый синтаксис CHANGE MASTER

How to use: New syntax

```
server_C> CHANGE MASTER TO MASTER_HOST = 'server_A',
                           MASTER_AUTO_POSITION = 1;
```



GTID: мониторинг

Monitoring: gtid_done

```
master> CREATE TABLE t1 (a INT);
```

GTID: мониторинг

Monitoring: gtid_done

```
master> CREATE TABLE t1 (a INT);
master> SELECT @@global.gtid_done;
a61678ba-4889-4279-9e58-45ba840af334:1
```

GTID: мониторинг

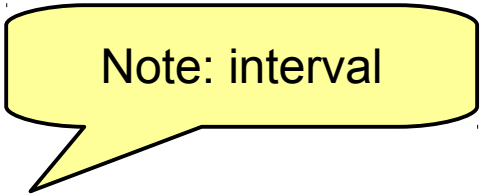
Monitoring: gtid_done

```
master> CREATE TABLE t1 (a INT);
master> SELECT @@global.gtid_done;
a61678ba-4889-4279-9e58-45ba840af334:1
master> INSERT INTO t1 VALUES (1);
master> INSERT INTO t1 VALUES (2);
```

GTID: мониторинг

Monitoring: gtid_done

```
master> CREATE TABLE t1 (a INT);
master> SELECT @@global.gtid_done;
a61678ba-4889-4279-9e58-45ba840af334:1
master> INSERT INTO t1 VALUES (1);
master> INSERT INTO t1 VALUES (2);
master> SELECT @@global.gtid_done;
a61678ba-4889-4279-9e58-45ba840af334:1-3
```



GTID: мониторинг

Monitoring: gtid_done

```
master> SELECT @@global.gtid_done;
a61678ba-4889-4279-9e58-45ba840af334:1-10000
 slave> SELECT @@global.gtid_done;
a61678ba-4889-4279-9e58-45ba840af334:1-9999
```

GTID: итоги

Summary

- Purpose: Fail-over (or change topology – crash not required)
- Basic usage is simple
  - `CHANGE MASTER TO MASTER_AUTO_POSITION = 1`
  - Failover has very small admin overhead
- Basic monitoring is simple
  - `SELECT @@global.gtid_done`

Автоматический failover

MySQL Utilities

- A collection of Python utilities for managing MySQL databases
- MySQL Workbench Plugin
- Available under the GPLv2 license
- Library to grow solutions for common administrative problems
- Download MySQL Workbench from:
  - **http://www.mysql.com/downloads/workbench/**
- You can also download the latest development source code иерархия for the MySQL Workbench Utilities from:
  - **http://launchpad/net/mysql-utilities**

Automatic Fail-Over

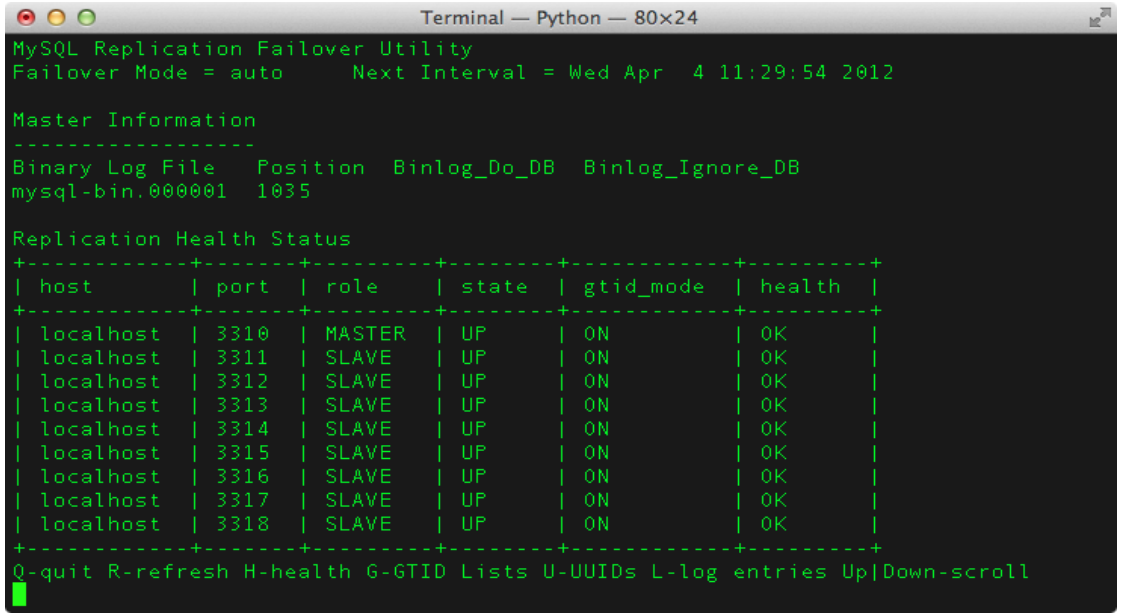
MySQL Utilities

- Easily administer MySQL servers from the command цепочка
  - mysqldbcompare – compare databases
  - mysqldbcopy – copy databases between servers
  - **mysqlfailover** – Automatic fail-over
  - **mysqlrpladmin –** General replication administration utility
  - **mysqlrplshow** – show a graph of your topology
  - **mysqlreplicate** – setup replication
  - mysqlrplcheck – check replication configuration
  - ...
- Build your own tools on top of the core of the library, e.g., automate timeshare multi-source replication setup or failing-over to a slave

Automatic Fail-Over

mysqlfailover

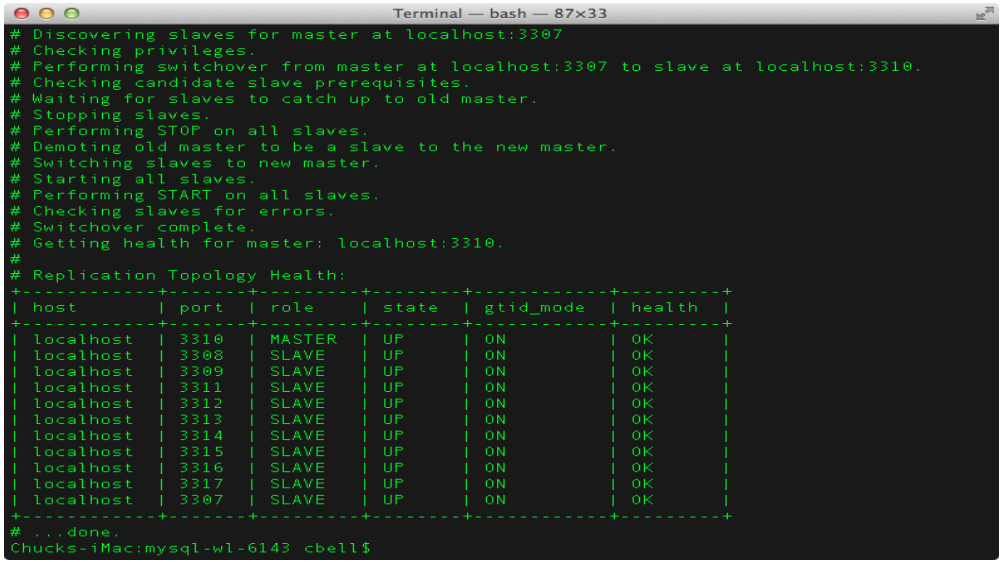
- Check and report health at specific intervals in seconds.
- Automatic fail-over.



Automatic Fail-Over

mysqlrpladmin

- General replication administration utility: start, stop, topology health, elect, failover, switchover, gtid.
- On-demand failover or switchover.



Enhanced Reliability

User Interface for Controlling CRC.

- **--binlog-checksum** = CRC32 | NONE
  - Turns on/off generation of CRCs on the master.
  - SET @@global.binlog_checksum
- **--master-verify-checksum**=0 | 1
  - Dump thread and user sessions verify checksums.
  - SET @@global.master_verify_checksum
- **--slave-sql-verify-checksum**=0 | 1
  - SQL thread verifies checksums.
  - SET @@global.slave_sql_verify_checksums

Self-healing Slaves

Hands-on

- Backward compatible: one can still use files if one wants:
  - --master-info-repository= **FILE** | **TABLE**
  - --relay-log-info-repository= **FILE** | **TABLE**
- Tables are located in the **mysql** schema and are **InnoDB** by default.
  - mysql.slave_relay_log_info
  - mysql.slave_master_info
- Tables engine can be **altered**, e.g.:
  - ALTER TABLE mysql.slave_master_info ENGINE = ...;

Self-healing Slaves

Hands-on

- **slave_relay_log_info** is updated after each relay log rotation, when the SQL thread is stopped, after a commit or rollback, after a statement executed without transactional context.
  - DDLs in mysql are not transactional, thus the table is updated after the event is processed.
  - **--sync-relay-log-info** has no effect on tables.

Self-healing Slaves

Hands-on

- **slave_master_info** is updated when the relay log is rotated, CHANGE MASTER is executed, on STOP|START SLAVE IO_THREAD and when sync-master-info period has elapsed.
  - **--sync-master-info** = 0
    - updates only on the cases mentioned.
    - together with **–relay-log-recovery=1** should provide sufficient fault-tolerance.
  - **--sync-master-info** = N (N >0)
    - comes with a penalty, but table is updated more often.

Summary

- MySQL 5.6 is the Foundation for building rock solid highly available services infrastructures.
- High Availability features: Global Transaction Identifiers, Crash-safe Slaves, Crash-safe Binary logs.
- Flexibility and usability enhancements.
- Reduced administration overhead.
- MySQL Utilities already provide automatic fail-over capabilities, off-loading the DBA.