ORACLE

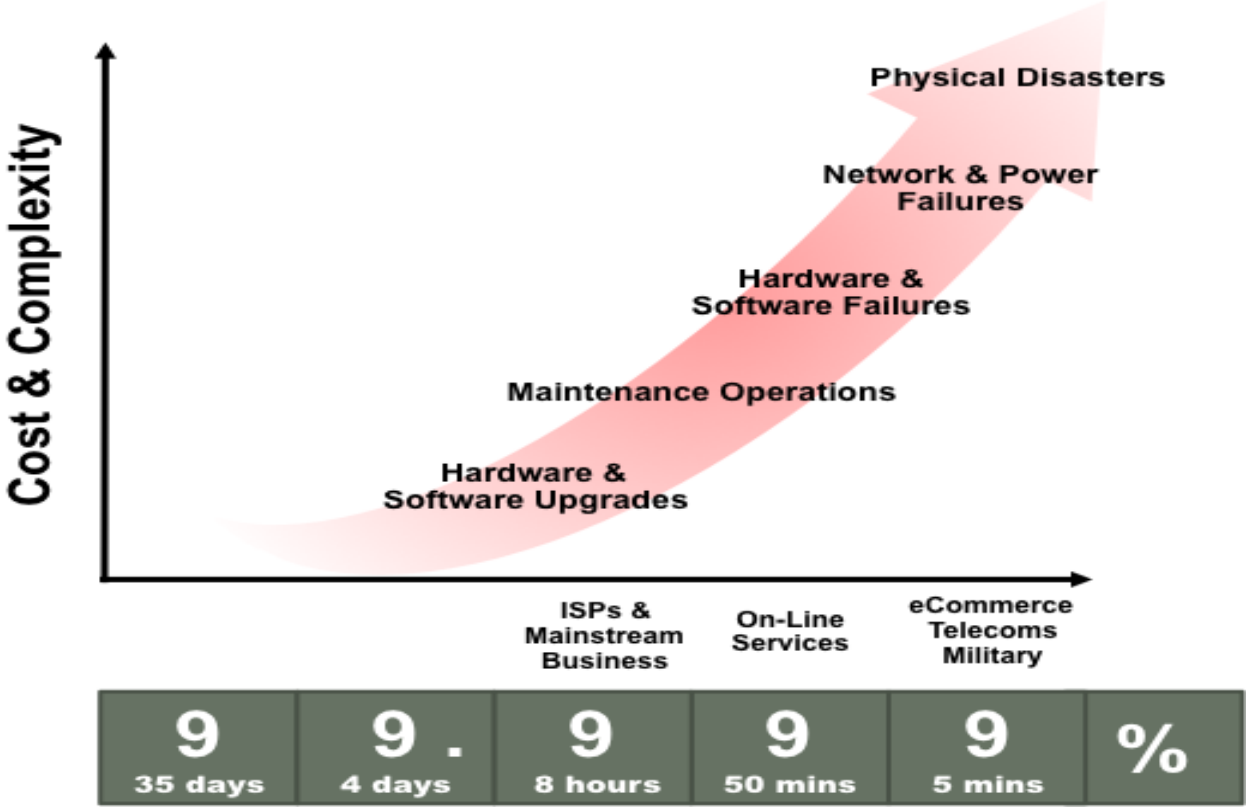
MySQLの高可用性構成

日本オラクル株式会社
山崎 由章 / MySQL Senior Sales Consultant,
Asia Pacific and Japan

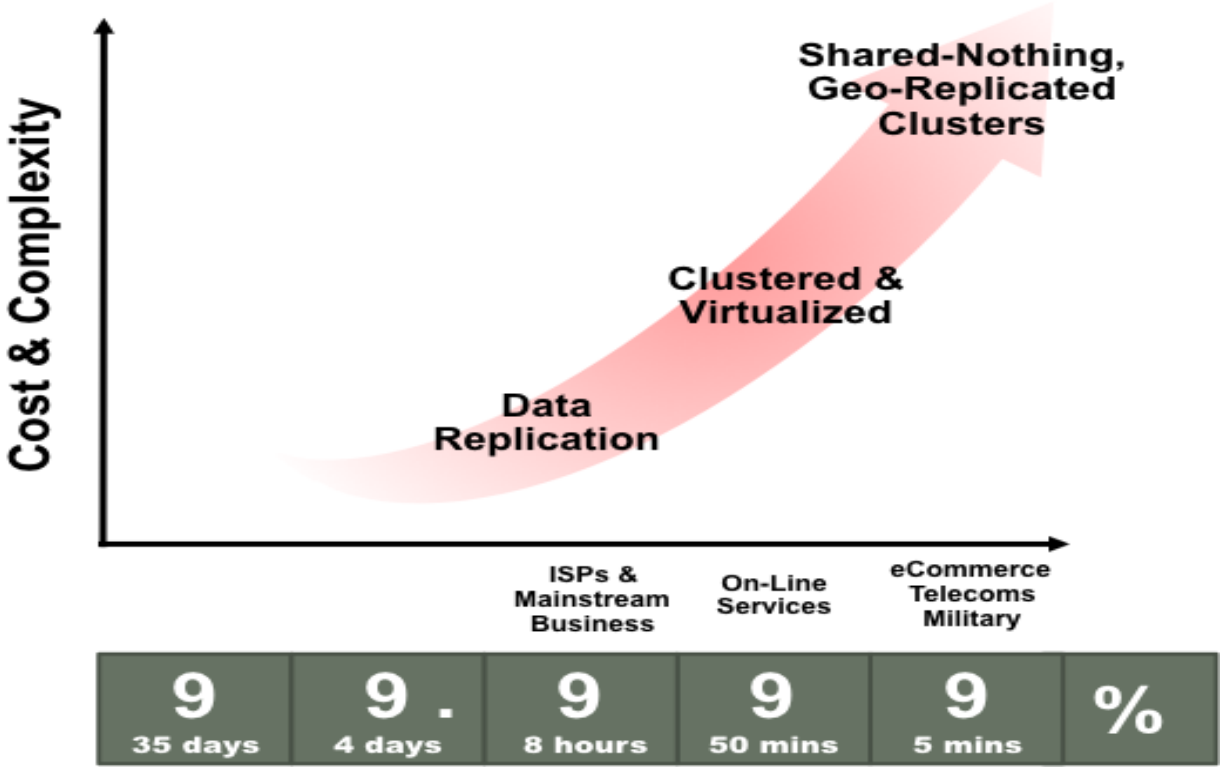
MySQL

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

Designing for High Availability

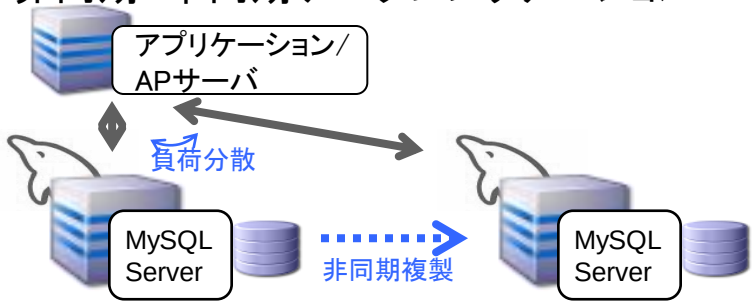


Selecting the Right HA Architecture

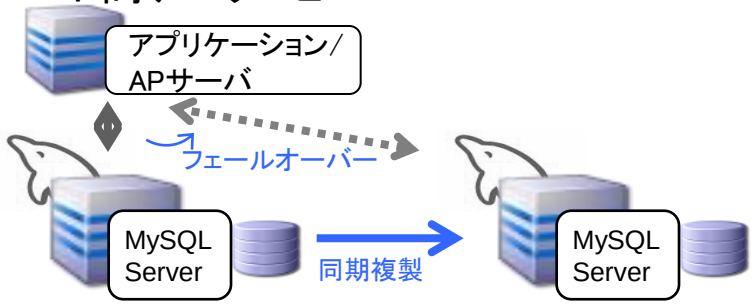


MySQLの高可用性構成

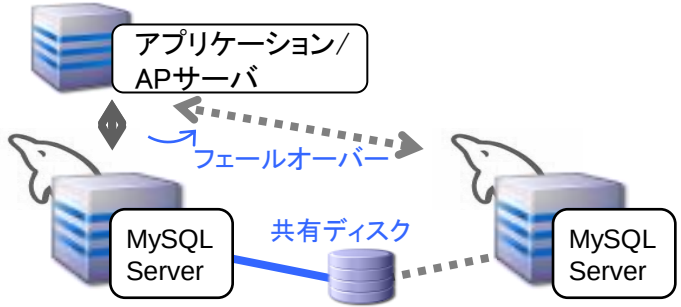
- レプリケーション（標準機能）
  非同期&準同期データレプリケーション



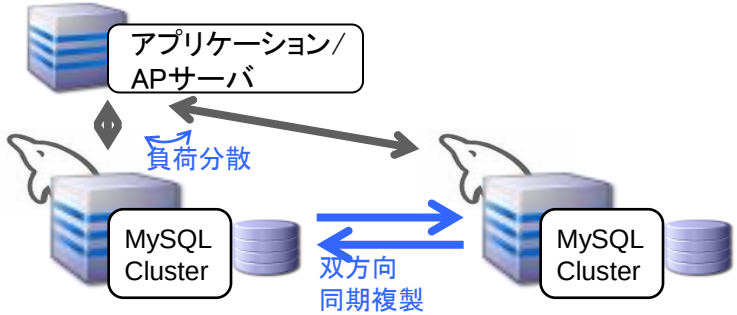
- MySQL+DRBD
  ノード間データコピー



- 3rdベンダ製HAソフト利用
  共有ディスクにデータを格納

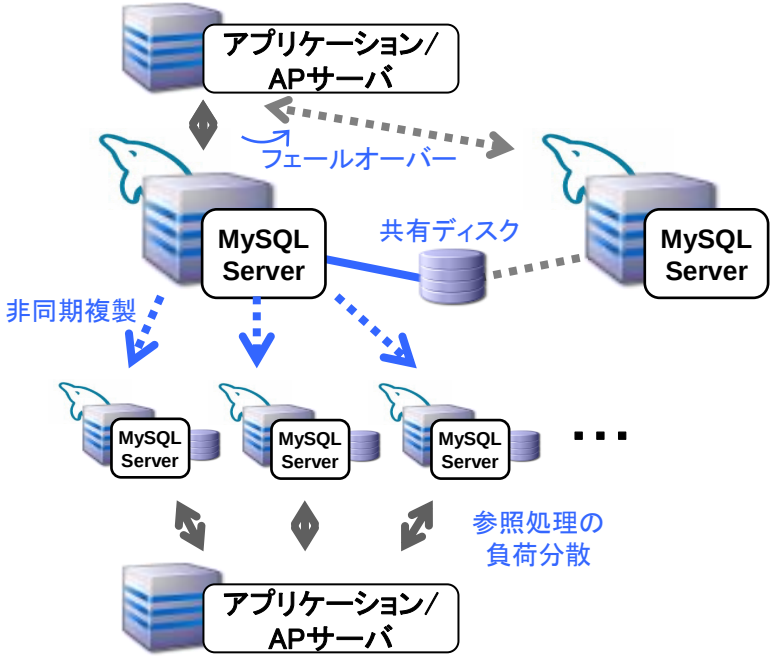


- MySQL Cluster
  シェアードナッシング型高性能クラスタ

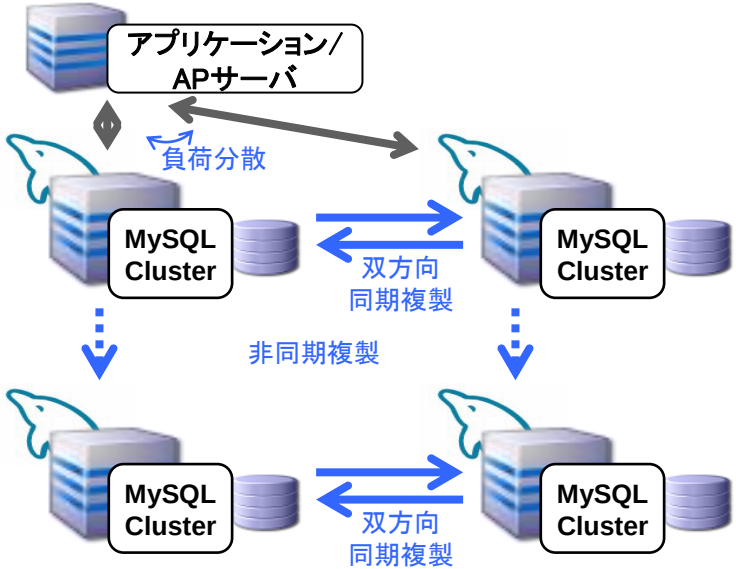


複合型の高可用性構成例

- 共有ディスク型構成
  ＋レプリケーション



- MySQL Cluster
  ＋レプリケーション



MySQL レプリケーション

レプリケーションとは？

データの変更点を1つ以上の場所に複製すること

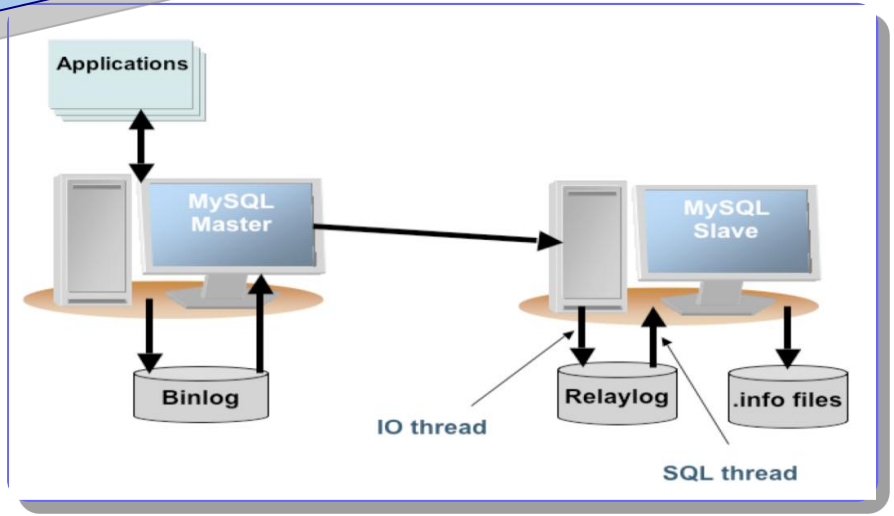
非同期レプリケーション - Asynchronous Replication

同期レプリケーション - Synchronous Replication

レプリケーション

- MySQLの標準機能
  - シンプルな設定
  - マスタ→スレーブ
  - 多数Webでの実績
- 非同期型 or 準同期
- 特徴
  - 参照性能を向上させる構成
  - バックアップ用途での利用も
  - 基本は一方向でのデータコピー
    だが、双方向や循環型での利用も可能
    （データの更新には注意が必要）
  - 更新ログ(bin-log)を利用

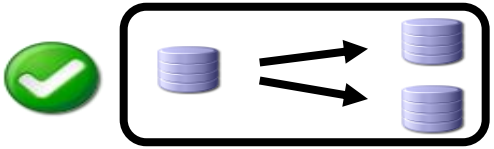
**Webアプリケーションでは参照が95%、
更新が5%というケースも (Digg.com)
> シンプルなスケールアウト構成によって
簡単に20倍以上の性能向上が図れる**



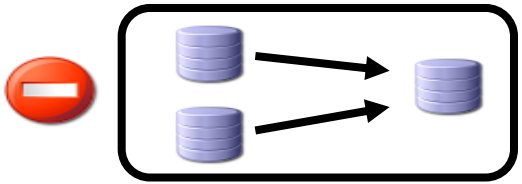
MySQL レプリケーションの構成パターン

マスタ > スレーブ

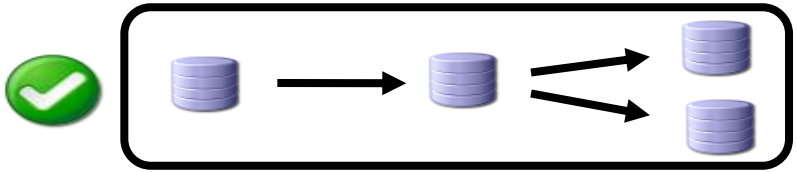
マスタ > マルチスレーブ



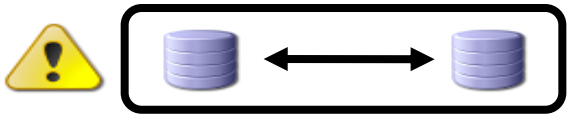
マルチマスタ > スレーブ (マルチソース)



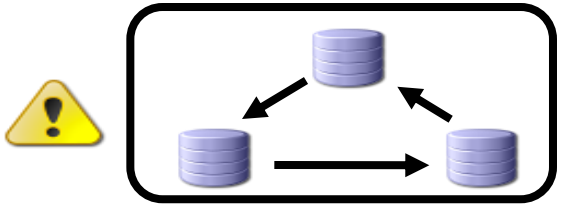
マスタ > スレーブ > マルチスレーブ



マスタ < > マスタ (マルチマスタ)



循環型 (マルチマスタ)



MySQLレプリケーションのセットアップ

- [Master]下記オプションを設定して起動;
  - server-id
  - log-bin
  - datadir *
- [Slave]下記オプションを設定して起動;
  - server-id
  - datadir *
  - port *
  - socket * (if in Unix like OS)
- [Master]レプリケーション用のユーザを作成
  - "REPLICATION SLAVE"権限を付与

```
CREATE USER 'sluser'@'localhost' IDENTIFIED BY 'slpass';
GRANT REPLICATION SLAVE ON *.* TO 'sluser'@'localhost';
```

MySQLレプリケーションのセットアップ

- [Slave] `CHANGE MASTER TO`コマンドを実行;
  - MASTER_HOST
  - MASTER_USER
  - MASTER_PASSWORD

  または、これらのオプションを設定ファイルに記述
- [Slave] `START SLAVE`コマンドを実行

```
CHANGE MASTER TO
  MASTER_HOST='localhost',
  MASTER_USER='sluser',
  MASTER_PASSWORD='slpass';
START SLAVE;
```

※MySQL 5.6の場合、セキュリティ向上のためにCHANGE MASTER TO時にMASTER_USER、MASTER_PASSWORDを指定せずに、START SLAVE時に指定することも可能。(master.info内にユーザ名/パスワードが保存されることを防ぐ)

実際の運用時に検討すべき事項

- 最初にデータをスレーブにコピーする際、バイナリログのポジション(トランザクション番号)を記録
- マスターとスレーブ間の通信をSSLで暗号化
  - 圧縮機能とあわせてディザスタリカバリ用途での利用時に重要
- MySQLレプリケーション単独では用意されていない機能
  - 高可用性構成としての利用時にフェールオーバーさせる仕組み
    =>MySQL5.6にて、自動フェールオーバーできるスクリプトを提供
  - 更新と参照の処理を分散させる仕組み
  - スレーブ間でのロードバランスの仕組み

http://dev.mysql.com/doc/refman/5.6/en/replication-howto.html

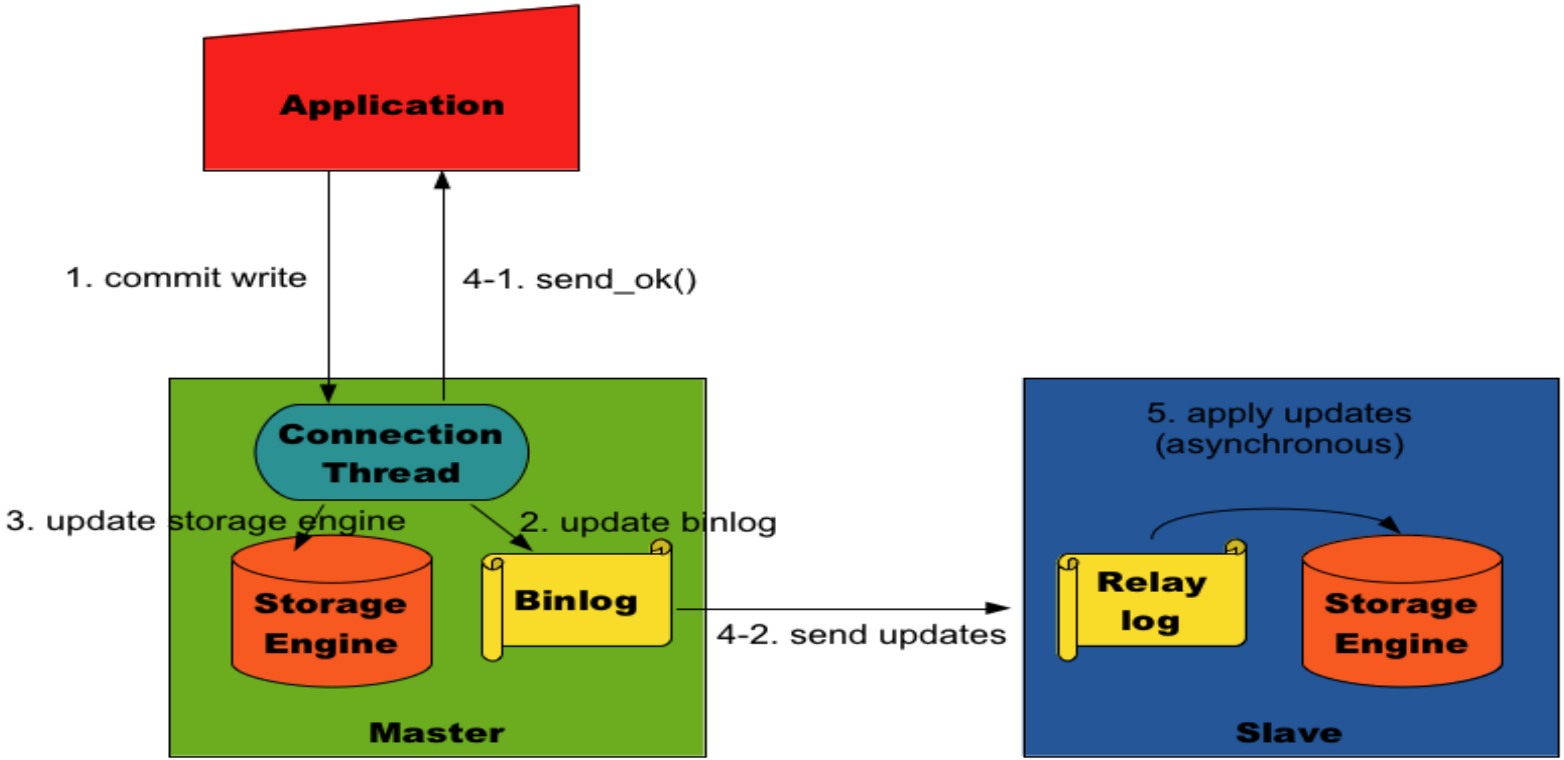
MySQLレプリケーションでの「交通整理」

- MySQL Proxyを利用
  - Master : proxy-backend-addresses
  - Slaves : proxy-read-only-backend-addresses

  http://dev.mysql.com/doc/refman/5.6/en/mysql-proxy-configuration.html
- MySQLのJDBC Driver "MySQL Connector/J"を利用

  http://dev.mysql.com/doc/refman/5.6/en/connector-j-reference-configuration-properties.html

非同期レプリケーション



バイナリログフォーマットの種類

フォーマット	説明	サイズ	Non-deterministic	Trigger
SBR	ステートメント（SQL文）がそのままバイナリログに記録される。	小	×	○
RBR	更新されたデータそのものが記録される。	大	○	×
MBR	SBRとRBRを状況に応じて切り換える。	小	○	△

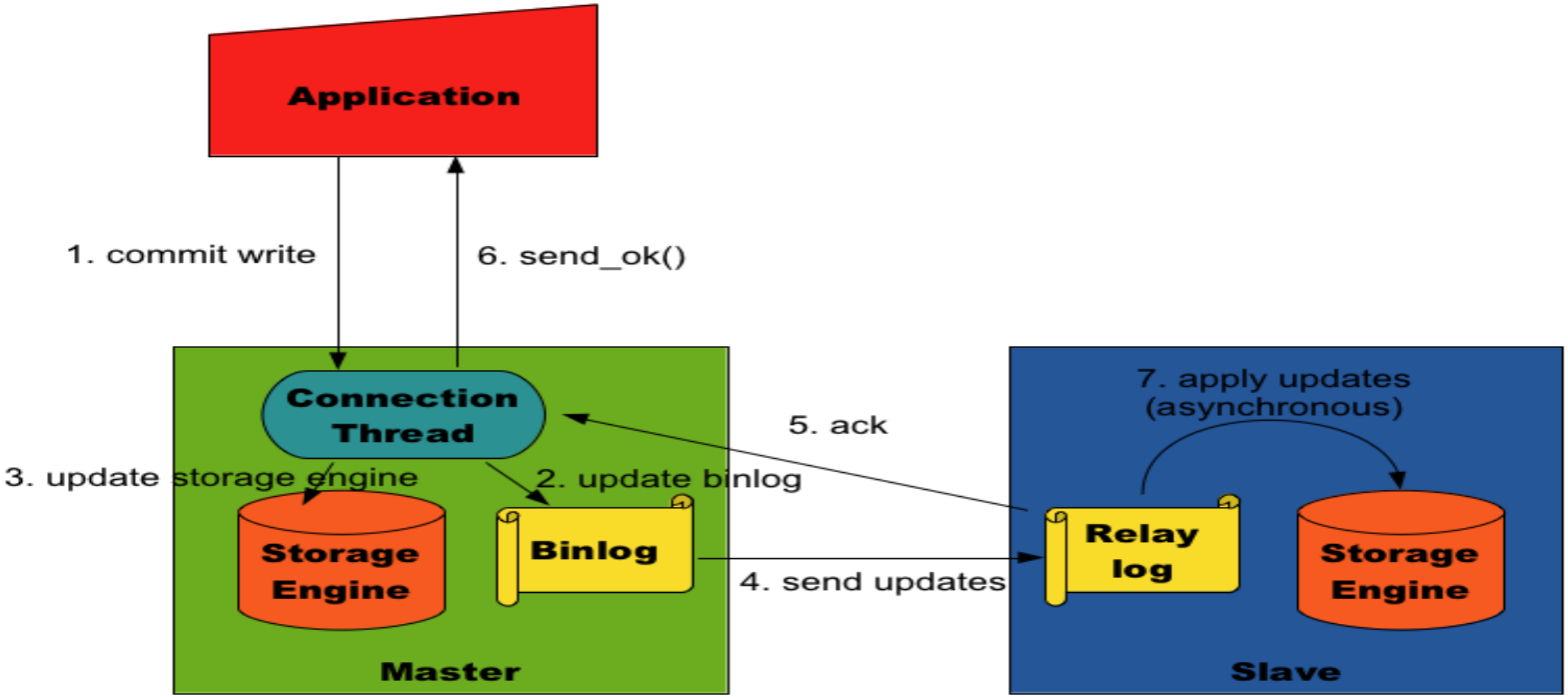
Non-deterministicって？

- 非決定性なSQL文
  ＝実行するたびに結果が変わる可能性がある。
  - UUID()、UUID_SHORT()
  - USER()
  - FOUND_ROWS()
  - LOAD_FILE()
  - SYSDATE()
  - GET_LOCK()、RELEASE_LOCK()
  - IS_FREE_LOCK()、IS_USED_LOCK()
  - MASTER_POS_WAIT()
  - SLEEP()
  - VERSION()
  - ソートなしのLIMIT句
  - UDF、非決定性のストアドプロシージャ/ファンクション
  - INFORMATION_SCHEMAの参照
  - READ-COMMITTED/READ-UNCOMMITTED

バイナリログの管理

- 有効化: --log-bin=binlog
- 一覧表示: SHOW BINARY LOGS;
- 内容の確認:
  - SHOW BINLOG EVENTS IN 'binlog.000001'
  - mysqlbinlog binlog.000001
- 削除: PURGE BINARY LOGS TO 'binlog.000002'
- 自動削除: --expire-logs-days=30

準同期レプリケーション



Semi-synchronous Replication

- **On master**
  - INSTALL PLUGIN 'rpl_semi_sync_master' SONAME 'semisync_master.so';
  - SET rpl_semi_sync_master_enabled=1;
  - SET rpl_semi_sync_master_timeout=1000; (1s, default 10s)
- **On slave**
  - INSTALL PLUGIN 'rpl_semi_sync_slave' SONAME 'semisync_slave.so';
  - SET rpl_semi_sync_slave_enabled=1;
  - START SLAVE;

Semi-synchronous Replication

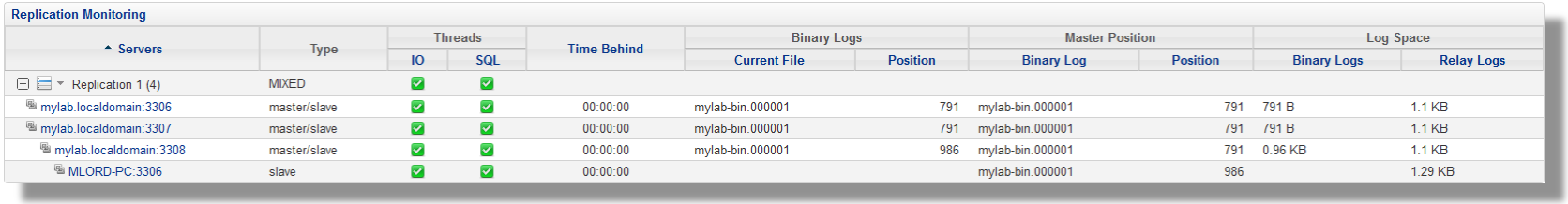
Checking the state

- **On master**
  - Rpl_semi_sync_master_status
  - Rpl_semi_sync_master_clients
  - Rpl_semi_sync_master_yes_tx
  - Rpl_semi_sync_master_no_tx
- **On Slave**
  - Rpl_semi_sync_slave_status

レプリケーション監視の自動化
MySQL Enterprise Monitor

- 自動でレプリケーション構成、マスタ/スレーブを検出
- リアルタイムでレプリケーションの稼働状況を収集
- 同期に問題があれば通知

監視作業負荷の軽減：レプリケーションの監視と稼働統計をコマンド無しで



Replication Monitoring

Servers	Type	Threads		Time Behind	Binary Logs		Master Position		Log Space	
		IO	SQL		Current File	Position	Binary Log	Position	Binary Logs	Relay Logs
Replication 1 (4)	MIXED	✓	✓							
mylab.localdomain:3306	master/slave	✓	✓	00:00:00	mylab-bin.000001	791	mylab-bin.000001	791	791 B	1.1 KB
mylab.localdomain:3307	master/slave	✓	✓	00:00:00	mylab-bin.000001	791	mylab-bin.000001	791	791 B	1.1 KB
mylab.localdomain:3308	master/slave	✓	✓	00:00:00	mylab-bin.000001	986	mylab-bin.000001	791	0.96 KB	1.1 KB
MLORD-PC:3306	slave	✓	✓	00:00:00			mylab-bin.000001	986		1.29 KB

MySQL 5.6での
レプリケーションの改善点

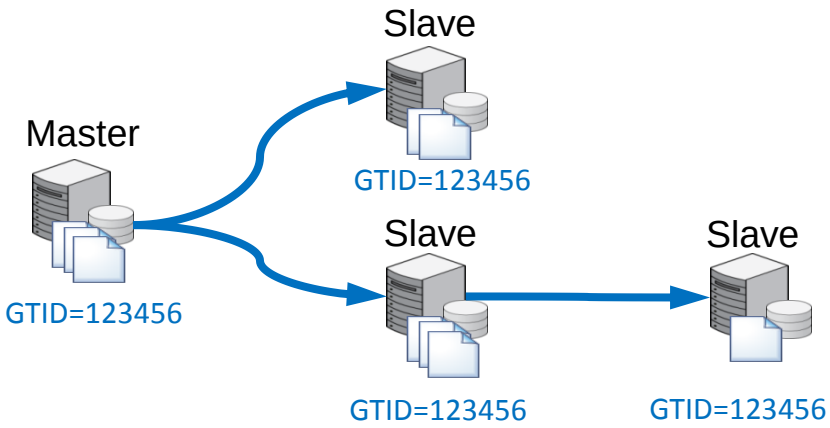
MySQL 5.6での強化ポイント

- パフォーマンス
  - マルチスレッドスレーブ
  - スレーブの行検出アルゴリズムの追加
  - バイナリログのサイズ削減
  - バイナリログ書込み時のロック削減
  - バイナリログのグループコミット
- フェイルオーバー＆リカバリ
  - グローバルトランザクションID(GTID)
  - mysqlfailover、mysqlrepladmin ユーティリティ
  - クラッシュ後の不完全なバイナリログを自動修復
  - クラッシュセーフなスレーブ
- 信頼性向上
  - チェックサムの追加
- 開発生産性、運用性向上
  - MySQL レプリケーション ユーティリティ
  - サーバUUIDの生成
  - バイナリログへの追加情報出力
  - スレーブが使用するNICの指定
  - バイナリログのリモートバックアップ
  - 遅延レプリケーション

フェイルオーバー＆リカバリ

グローバルトランザクションID(GTID)

- 複数台のレプリケーション環境でも容易にトランザクションの追跡/比較が可能
  - トランザクションを一意に識別できる識別子をバイナリログに記録
- フェイルオーバーのために、最も最新のスレーブを自動認識
- 多段構成のレプリケーションが容易に



グローバルトランザクションID(GTID)

- コミット時にGTIDが生成される
  - server_uuid:number
    a61678ba-4889-4279-9e58-45ba840af334:1
  - server_uuid によって、サーバを一意に識別できる
  - number は、各サーバ単位でトランザクション毎に1ずつ増加する
- GTIDはバイナリログに記録される

- GTIDによって、スレーブによるトランザクションの再実行を防ぐ

グローバルトランザクションID(GTID)

- フェイルオーバー、トポロジ変更時の動作(例：マスターを変更)
  - スレーブは新しいマスターに実行が完了しているGTID を伝える
  - マスターはスレーブが実行していないトランザクションだけを
    スレーブに送る

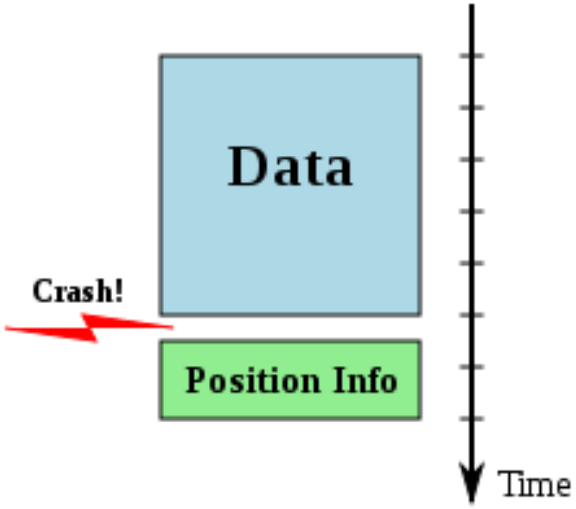
- @@GTID_EXECUTED で実行が完了したGTID を確認できる
  - 確認例）SELECT @@global.gtid_EXECUTED;

- マスターの変更時、ポジションは自動的に認識される
  - 管理の手間削減
  - CHANGE MASTER TO MASTER_AUTO_POSITION = 1

クラッシュセーフなスレーブ

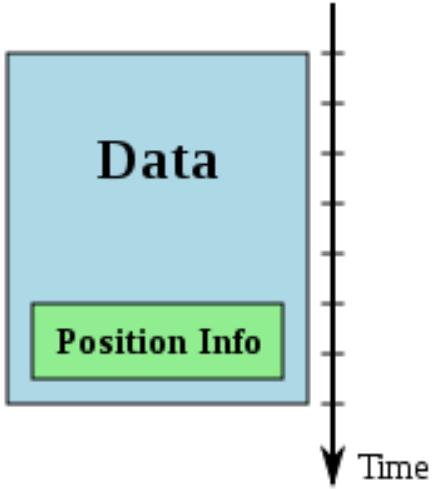
- ポジションの情報はファイルに格納されている
  - master.info、relay-log.info

- スレーブサーバにおいて、Commit と fsyncの間でクラッシュが起きたら？
  - ポジションとデータのミスマッチが発生
  - リカバリ時に、間違ったポジションから ログを適用してしまう
  - 手動でポジションを修正するか、 スレーブの再セットアップが必要



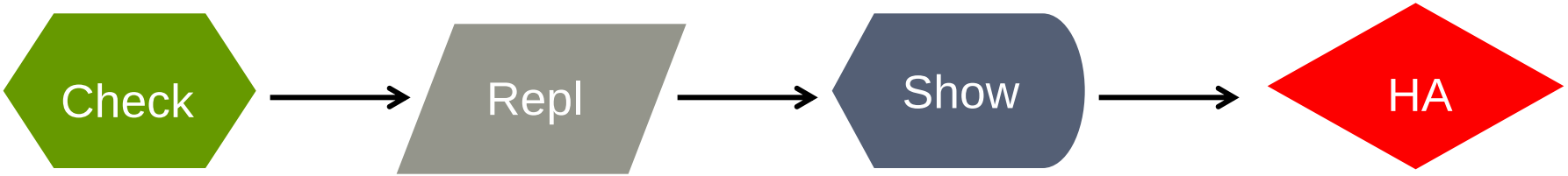
クラッシュセーフなスレーブ

- master.info、relay-log.info の情報をInnoDB上のテーブルに格納可能
  - データとポジションの情報をトランザクショナルに書込み可能
  - master_info_repository、relay_log_info_repository で設定
- スレーブサーバがクラッシュしても、マスターとの同期が崩れない
  - クラッシュセーフにするためには、relay_log_info_repository=TABLE と relay_log_recovery=ON を設定
- より強靭なレプリケーション環境が構築可



開発生産性、運用性向上

MySQL レプリケーション ユーティリティ



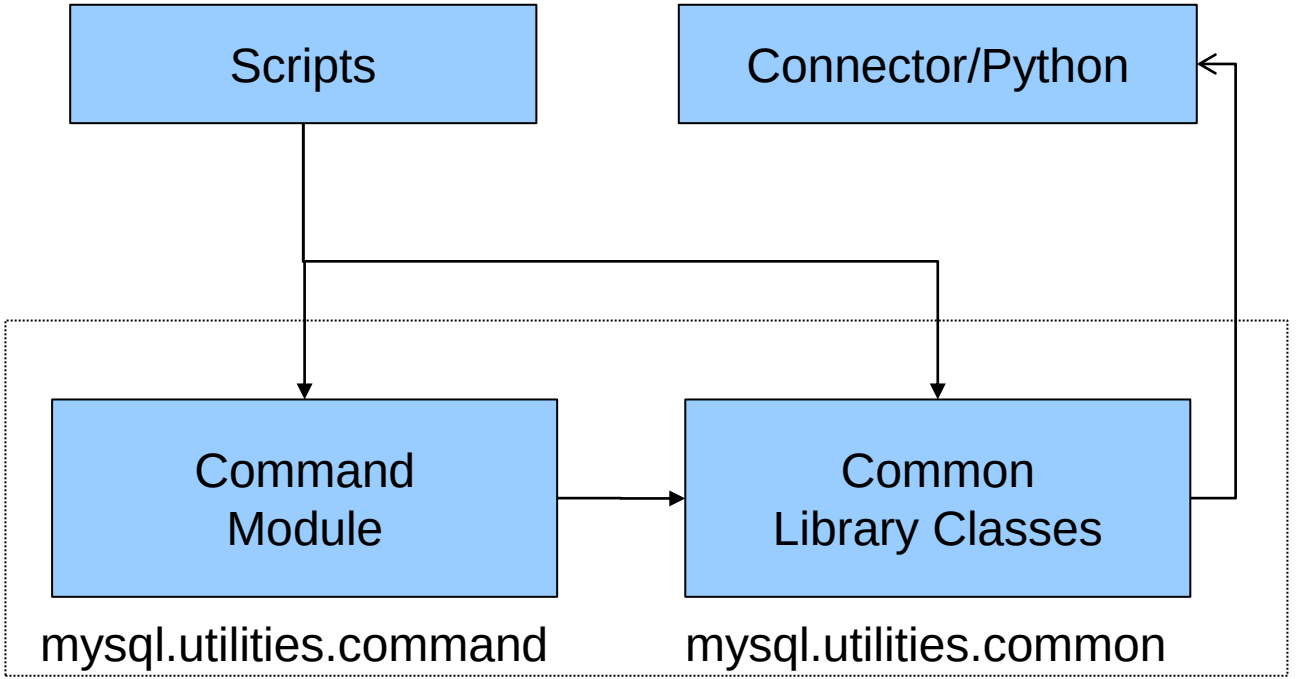
- Pythonスクリプトで実装された各種ユーティリティを提供
- スクリプトをカスタマイズ可能
  - ダウンロード先
    http://dev.mysql.com/downloads/tools/utilities/

MySQL Utilities

MySQL Utilitiesとは?

- MySQLを管理するためのPythonスクリプト集
- 最新バージョンは**1.3.5**
- ライセンスはGPLv2
- コードライブラリを用意しているので拡張が容易
- さまざまな運用管理タスクをカバーできるツール作りが目標
- ダウンロード先
  http://dev.mysql.com/downloads/tools/utilities/

アーキテクチャ



MySQL Utilities Library

データベース管理

- mysqldbcompare – データや定義を比較
- mysqldbcopy – 別のサーバにデータベースをコピー
- mysqldbexport – データとメタデータをエクスポート
- mysqldbimport – データとメタデータをインスポート
- mysqldiff – サーバ間のテーブルなどオブジェクトの定義を比較

General Utilities

- mysqldiskusage – デーブルおよびデータファイルのサイズを表示
- mysqlindexcheck – インデックスの重複をチェック
- mysqlmetagrep – テーブル定義のメタデータをgrep (正規表現利用可)
- mysqlprocgrep – プロセス情報をgrep (正規表現利用可)
- mysqluserclone – 別のサーバにユーザアカウントををコピー
- mysqluc – コマンドライン環境

High Availability

- mysqlfailover – レプリケーションの自動フェールオーバー
- mysqlreplicate – レプリケーションを設定
- mysqlrpladmin – レプリケーションの各種管理
- mysqlrplcheck – レプリケーションが正しく設定されているかの確認
- mysqlrplshow – レプリケーショントポロジ(親子関係)を図示

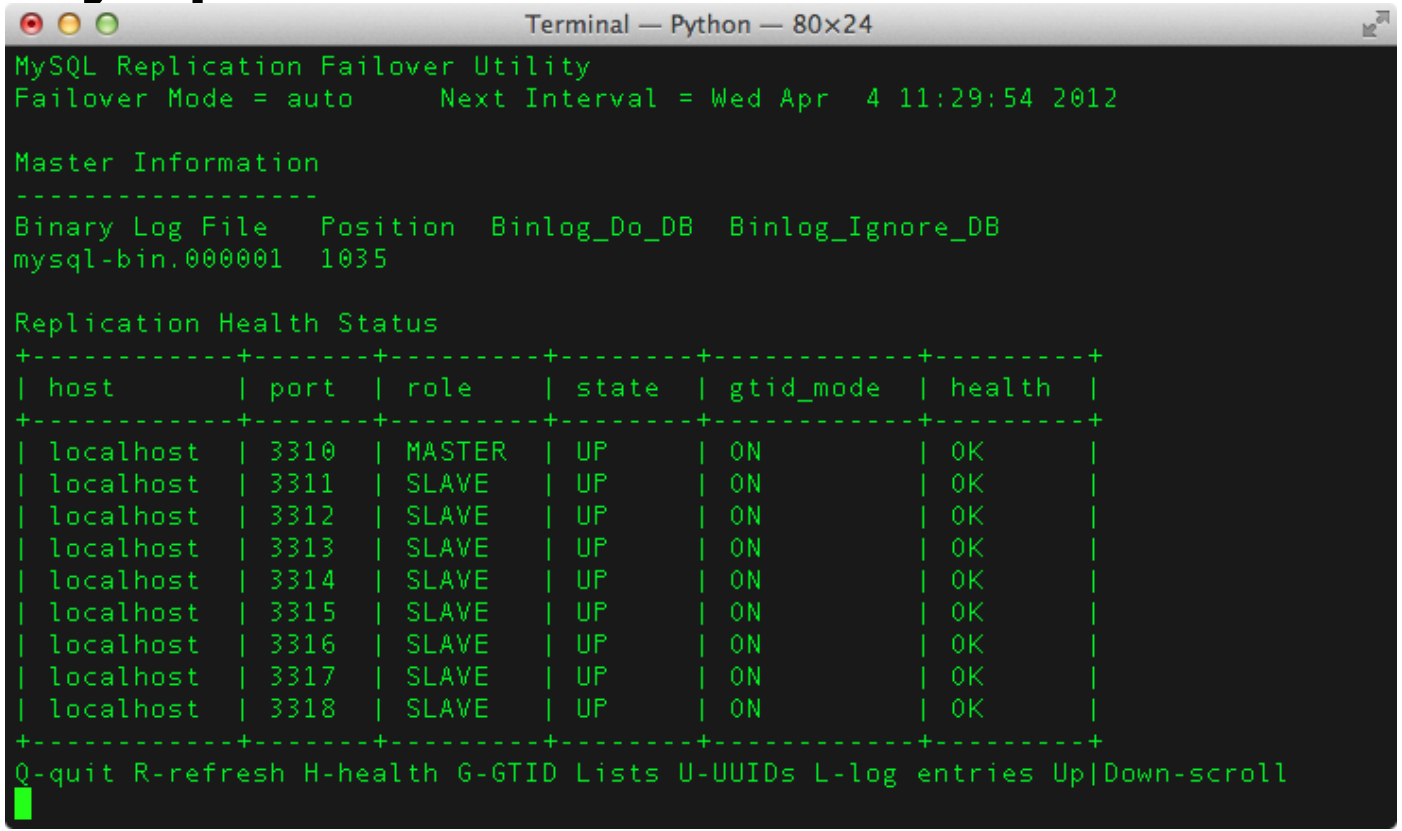
Server Operations

- mysqlserverclone – 既存のMySQLサーバのコピーを作成
- mysqlserverinfo – サーバの稼働状況を表示

mysqlfailoverによる自動フェールオーバー

- GTIDが有効になったMySQL 5.6の自動フェールオーバーが可能
- レプリケーションの稼働状態をチェック
- 1.3.4より、デーモンとしての稼働も可能に

mysqlfailoverによる自動フェールオーバー



mysqlfailover

- フェールオーバーのモード
  - **Auto** – まず候補リストのスレーブ、その後それ以外のスレーブにフェールオーバー
  - **Elect** – 候補リストのスレーブのみにフェールオーバー
  - **Fail** – マスター障害時にエラーとする

mysqlfailover

- 拡張のポイント
  - **exec-fail-check** - アプリケーション特有の問題を検知して、フェールオーバーが必要か判断するスクリプトを実行
  - **exec-before** - フェールオーバーが起こる前に実行されるスクリプト
  - **exec-after** - 新しいマスターにフェールオーバーした直後に実行されるスクリプト
  - **exec-post-fail** - フェールオーバーが完了し、全てのスレーブが新しいマスターを参照した後で実行されるスクリプト

MySQL 5.6のGITDによるレプリケーション運用

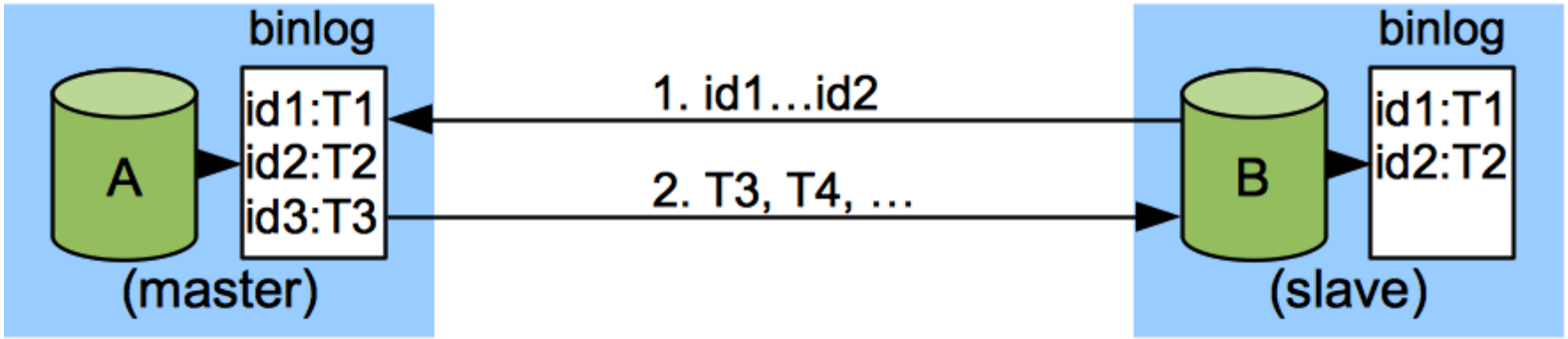
グローバルトランザクションID(GTID)

- コミット時にGTIDが生成される
  - server_uuid:number
    a61678ba-4889-4279-9e58-45ba840af334:1
  - server_uuid によって、サーバを一意に識別できる
  - number は、各サーバ単位でトランザクション毎に1ずつ増加する
- GTIDはバイナリログに記録される

- GTIDによって、スレーブによるトランザクションの再実行を防ぐ

Global Transaction Identifiers

- 新しいレプリケーションのプロトコル:
  - スレーブは *master:range* を表すIDをマスターに送る
  - マスタは *range* **以外**の全てのトランザクションをスレーブに送る



Global Transaction Identifiers

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

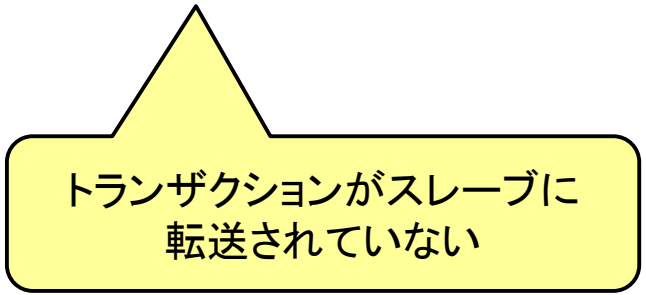
新しい変数:
gtid_executed

右側はIDの範囲

Global Transaction Identifiers

master> SELECT @@global.gtid_executed;
a61678ba-4889-4279-9e58-45ba840af334:1-10000

slave> SELECT @@global.gtid_executed;
a61678ba-4889-4279-9e58-45ba840af334:1-9999



GTIDによるレプリケーション

- **必要事項:**
  - トランザクションをサポートしたストレージエンジン(InnoDB)のテーブルとサポートしないストレージエンジン(MyISAMなど)のテーブルを、同一のトランザクション内やSQL文で変更しない (5.6.9以降)
  - CREATE TABLE … SELECT を使用しない
  - CREATE TEMPORARY TABLE や DROP TEMPORARY TABLE をトランザクション内部で実行しない

GTIDによるレプリケーション

- サーバのフェオールオーバーの準備
  1. データの同期を取り、全てのサーバを一旦停止
  2. 全てのmy.cnfに下記を追加:

```
gtid-mode=on
enforce-gtid-consistency=on
log-bin
log-slave-updates
```

  3. 全てのサーバを起動
  4. 実行:

```
> CHANGE MASTER TO MASTER_AUTO_POSITION = 1
```

GTIDによるレプリケーション

- フェールオーバー

slaveにて、新しいマスターを指定:

```
> CHANGE MASTER TO MASTER_HOST = '<host>',
                   MASTER_PORT = <port number>,
                   MASTER_USER = '<user name>'
                   MASTER_PASSWORD = 'secret';
```

- 「バイナリログポジション」の指定は不要

GTID_NEXT

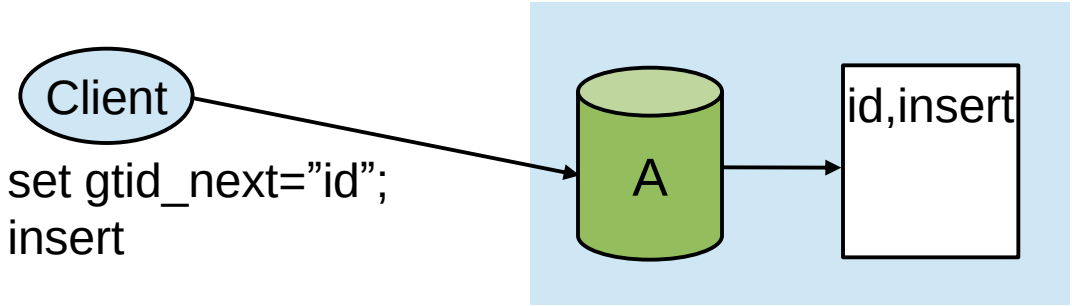
- **GTID_NEXT** – セッションごとのシステム変数
- デフォルト: **“AUTOMATIC”**

  → サーバが次のトランザクション用に **GTIDを生成**
- スレーブのスレッドが **“UUID:NUMBER”** を設定

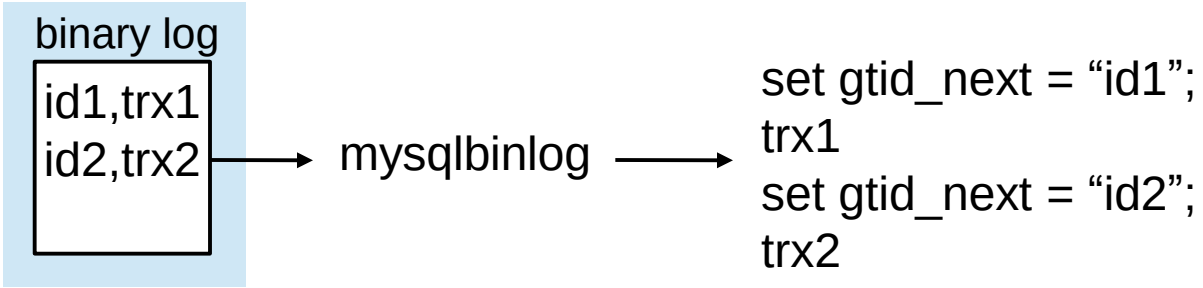
  → サーバは **指定されたGTID** を次のトランザクションで使用

GTID_NEXT

- クライアントもGTID_NEXT を取得可能:

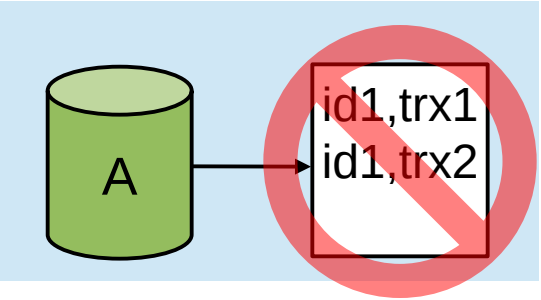


- **mysqlbinlog** が SET GTID_NEXT 文を出力:



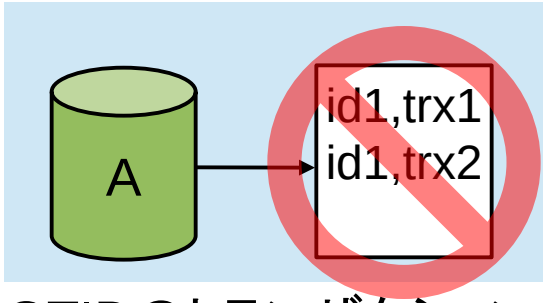
GTIDs Must Be Unique

- 同じGTIDのトランザクションは、2度実行できない:

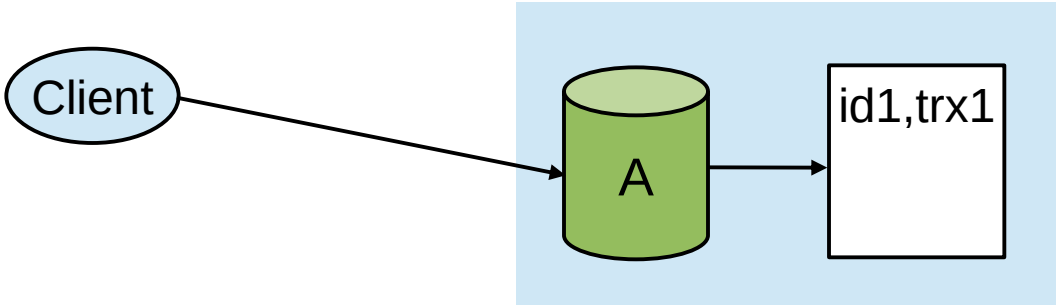


GTIDs Must Be Unique

- 同じGTIDのトランザクションは、2度実行できない:

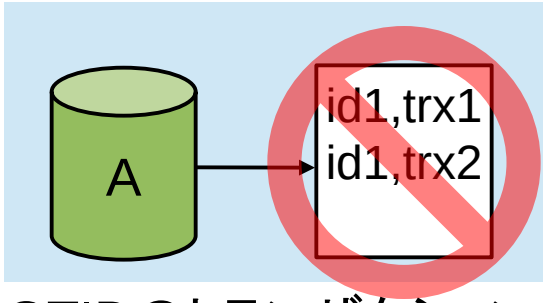


- 同じGTIDのトランザクションを再度実行した場合 → **トランザクションはスキップされる**

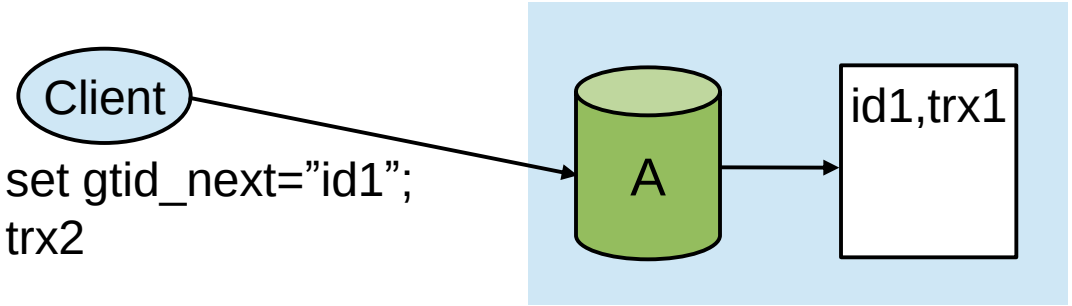


GTIDs Must Be Unique

- 同じGTIDのトランザクションは、2度実行できない:

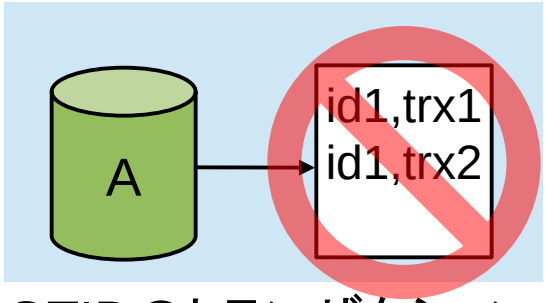


- 同じGTIDのトランザクションを再度実行した場合 → **トランザクションはスキップされる**

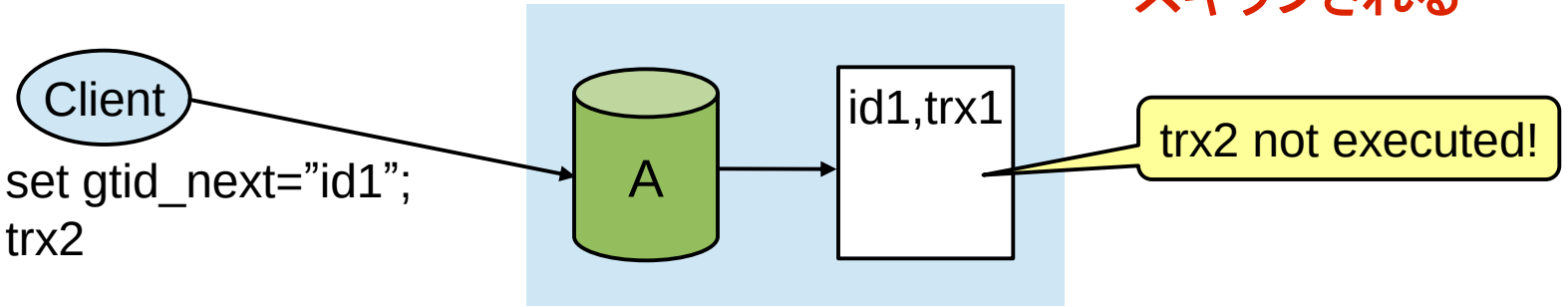


GTIDs Must Be Unique

- 同じGTIDのトランザクションは、2度実行できない:

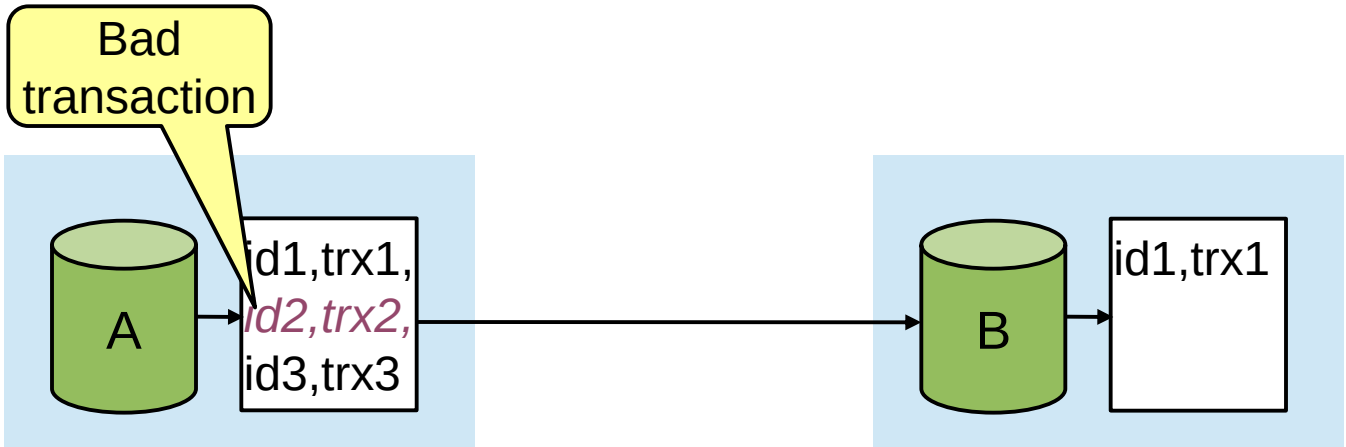


- 同じGTIDのトランザクションを再度実行した場合 → **トランザクションはスキップされる**



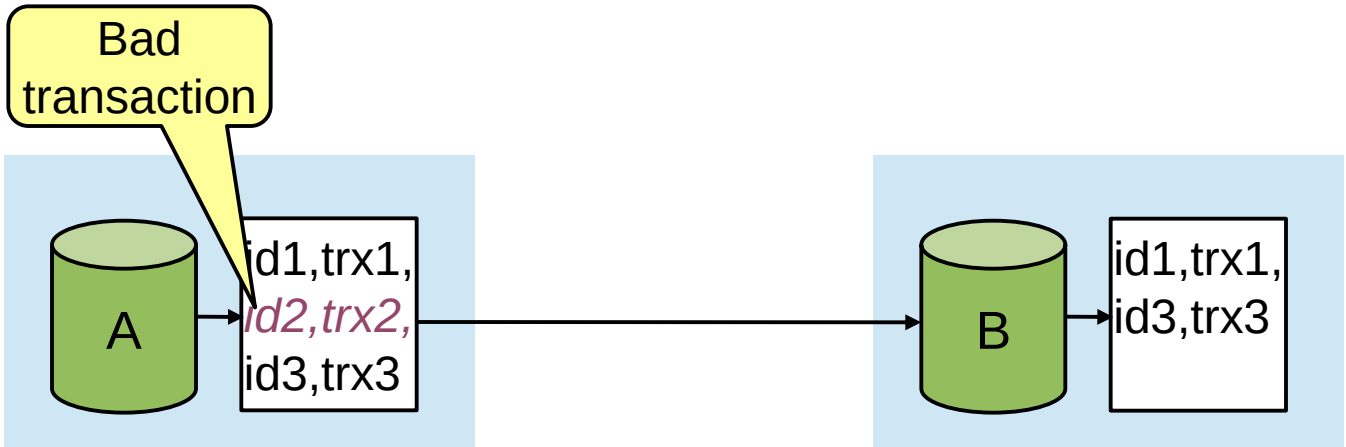
Skipping a Transaction

- GTID_NEXTの使用例: **不要なトランザクションをスキップする**
- トランザクションは再実行されない
- レプリケーションのポジションを移動させるだけでは不十分



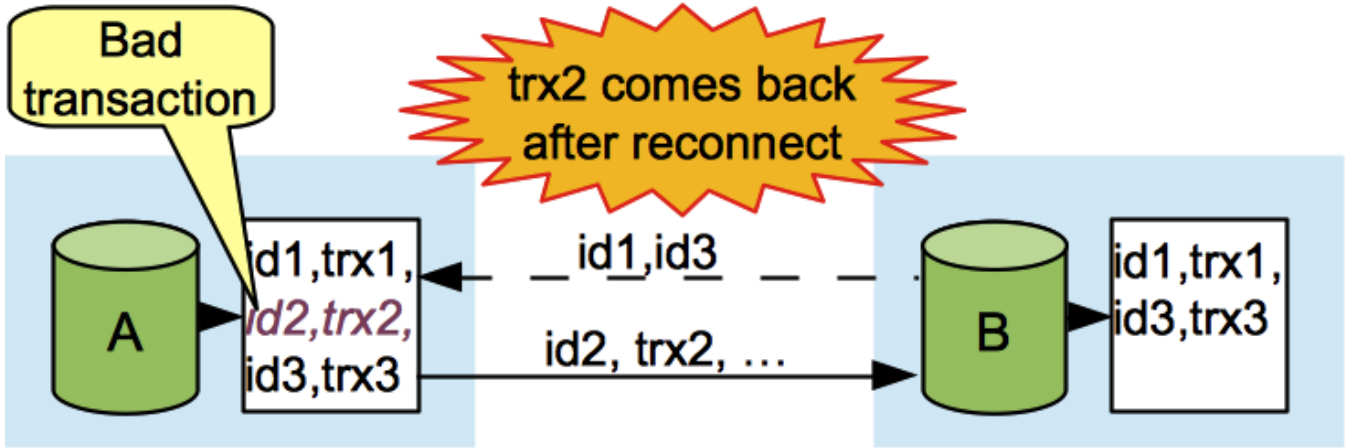
Skipping a Transaction

- GTID_NEXTの使用例: **不要なトランザクションをスキップする**
- トランザクションは再実行されない
- レプリケーションのポジションを移動させるだけでは不十分



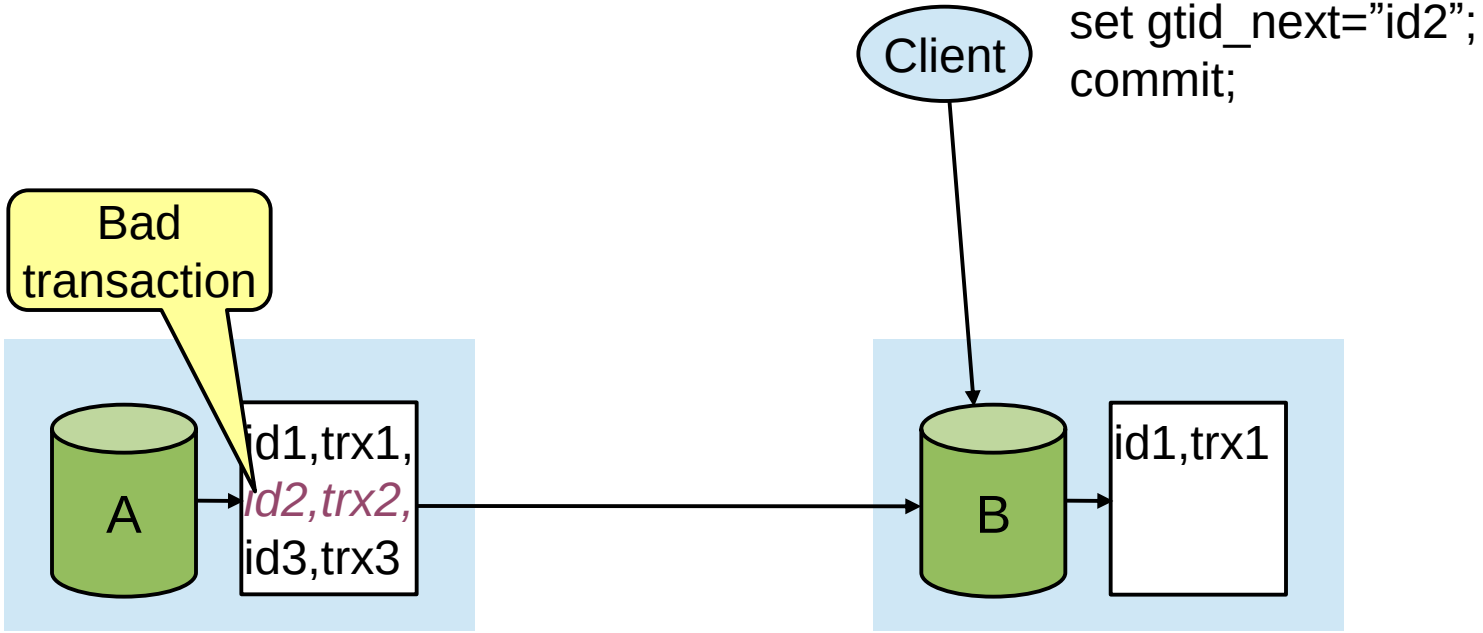
Skipping a Transaction

- GTID_NEXTの使用例: **不要なトランザクションをスキップする**
- トランザクションは再実行されない
- レプリケーションのポジションを移動させるだけでは不十分



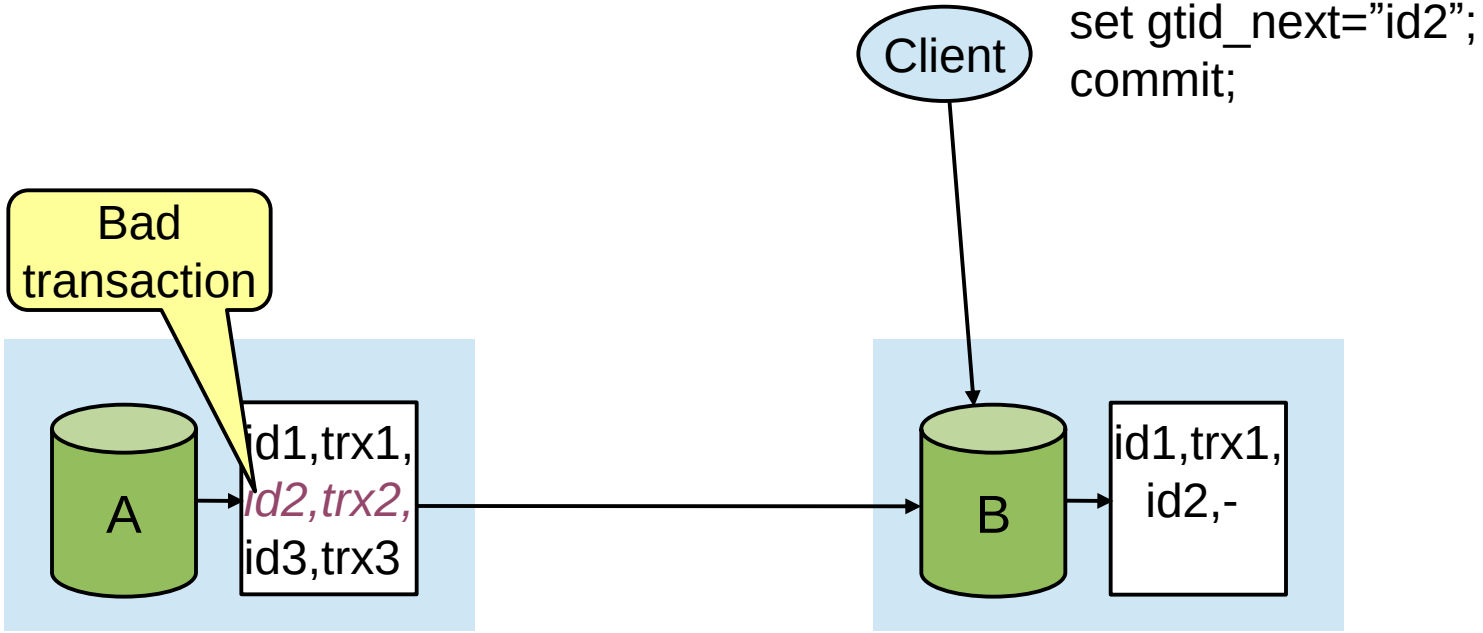
Skipping a Transaction

- トランザクションをスキップする方法



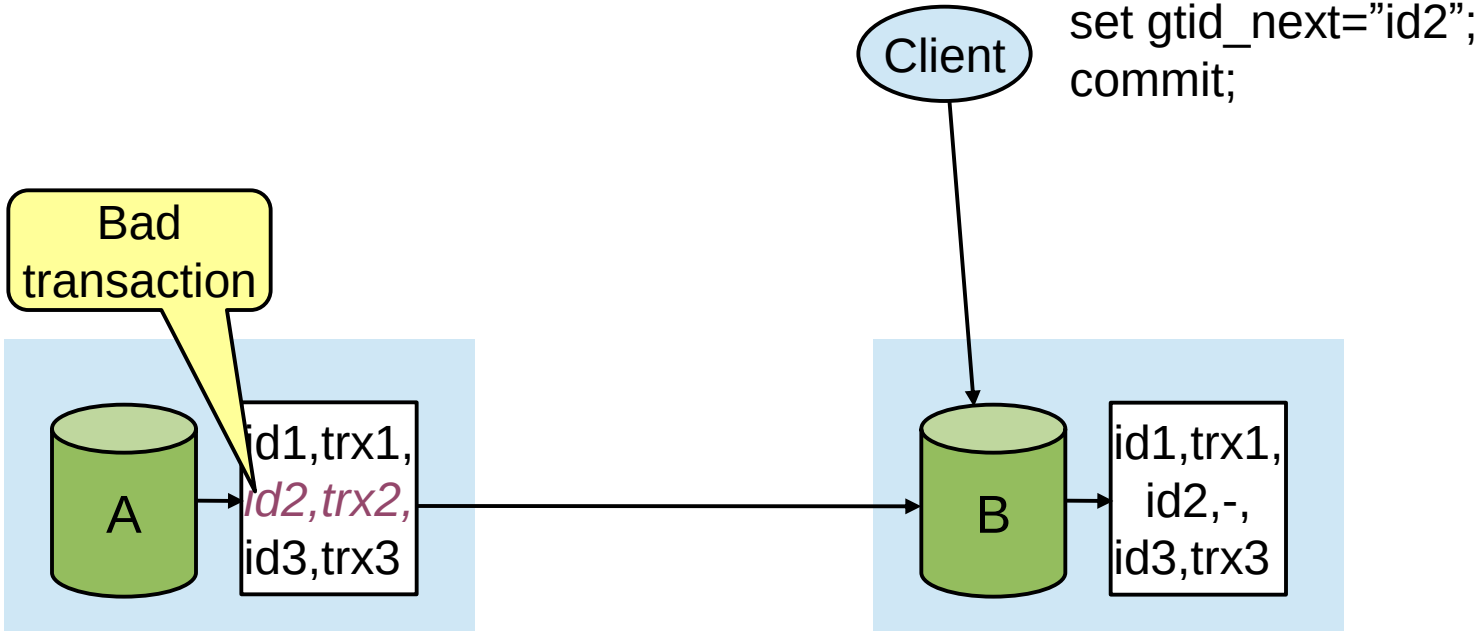
Skipping a Transaction

- トランザクションをスキップする方法



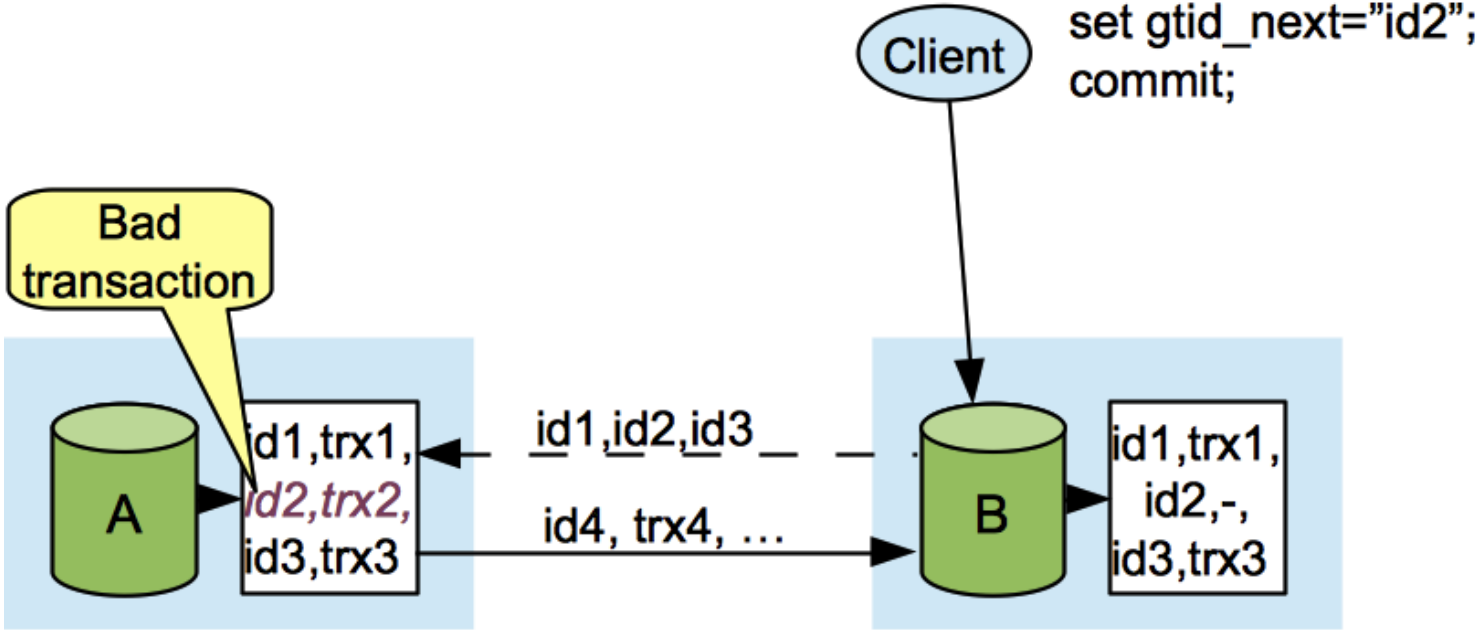
Skipping a Transaction

- トランザクションをスキップする方法



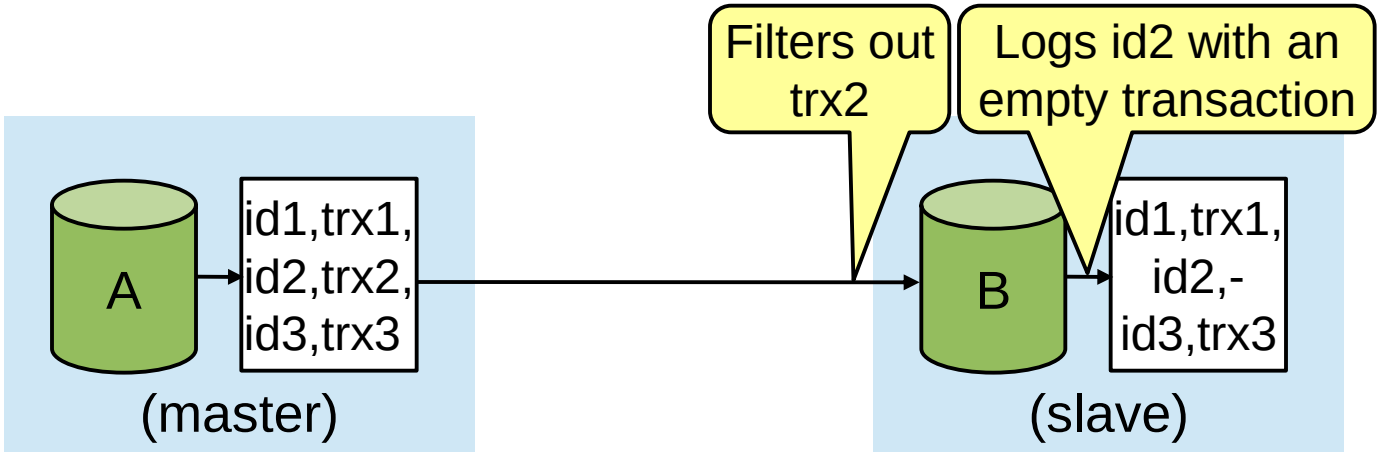
Skipping a Transaction

- トランザクションをスキップする方法



Filters

- --replicate-ignore-table *etc*
- トランザクションは無視される
- スレーブは空のトランザクションを実行する



まとめ

- ポジションをAからBに進める方法:
  - A~Bのトランザクションに対応するGTIDで、空のトランザクションを実行する
- ポジションを過去のポジションに戻す方法:
  - できない
- N個のトランザクションをスキップする方法:
  - GTIDを調べて、それぞれのGTIDに対して空のトランザクションをコミットする
- レプリケーションフィルターは、自動的に空のトランザクションをコミットする

Hardware and Software
Engineered to Work Together


ORACLE®