# MySQL入門(レプリケーション編)

Yoshiaki Yamasaki / 山﨑 由章
MySQL Senior Sales Consultant, Asia Pacific and Japan

# SAFE HARBOR STATEMENT

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するものではない為、
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。

# Program Agenda

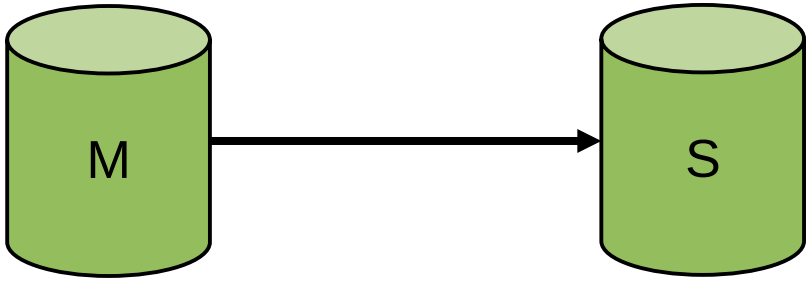
1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

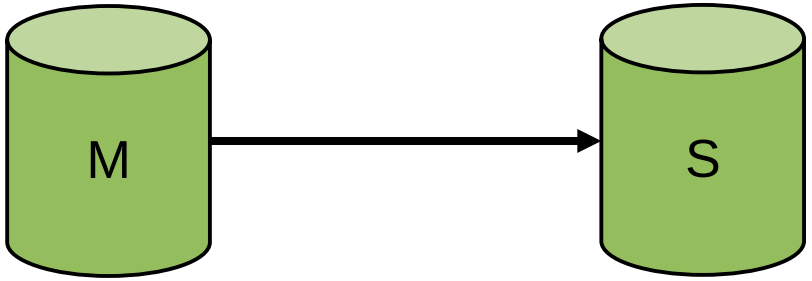
# レプリケーションとは？

- データの複製(レプリカ)を別のサーバーに持てる機能
- MySQLの標準機能で、多数のWebサイト等で利用されている
  - シンプルな設定で利用可能
  - マスター→スレーブ 構成

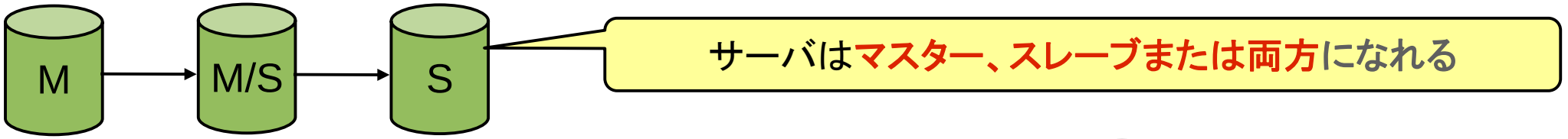


# レプリケーションとは？

- マスターサーバー
  - データを変更
  - 変更内容をスレーブに転送
- スレーブサーバー
  - マスターでの変更内容を受け取る
  - 変更内容をデーターベースに反映

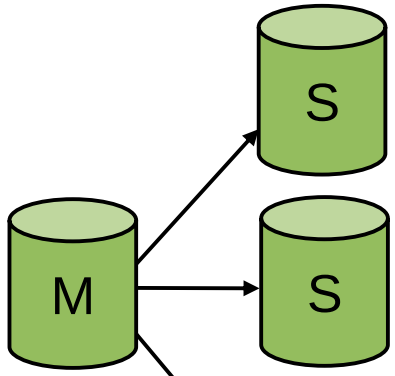


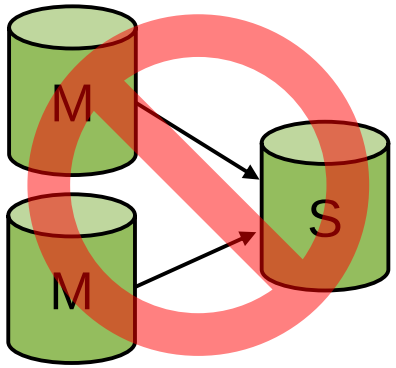
# 補足




サーバは**マスター、スレーブまたは両方**になれる

マスターは**複数のスレーブ**を持てる

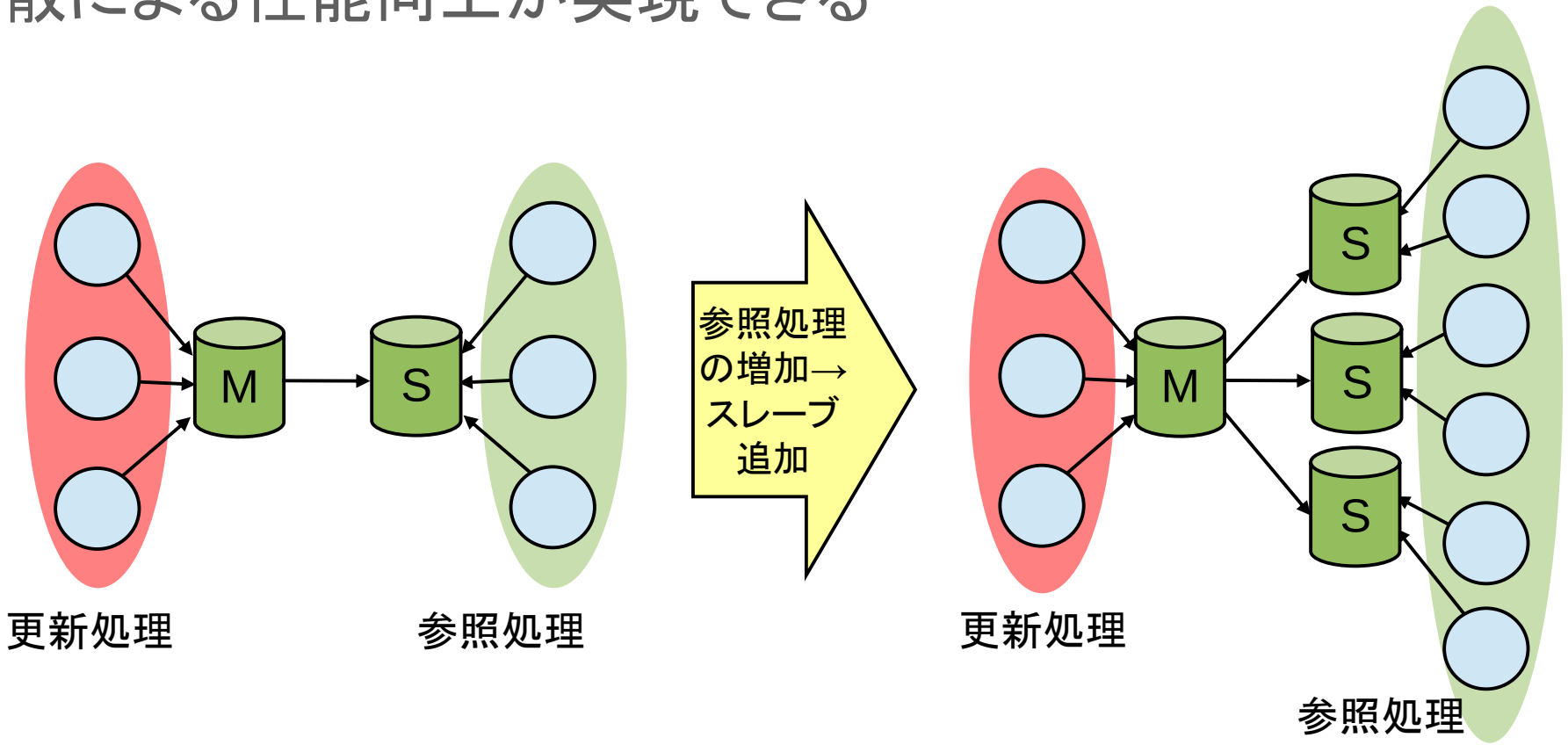



スレーブは**1つのマスターのみ**を持てる

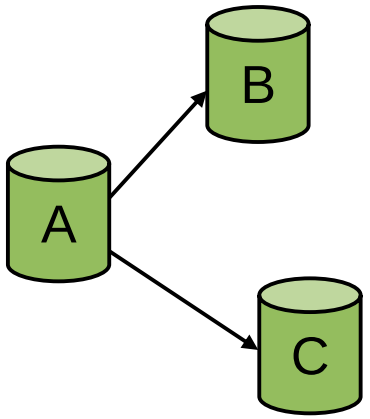
# レプリケーションの利点：参照性能の向上

- 参照処理の負荷が高い場合は、スレーブサーバーを追加することで、負荷分散による性能向上が実現できる



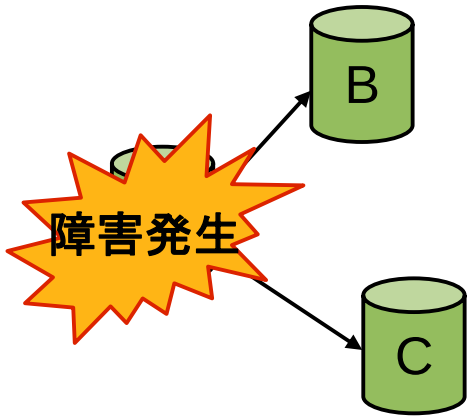
# レプリケーションの利点：高可用性構成の実現

- マスターの障害時に、スレーブをマスターに昇格することで高可用性を実現可能



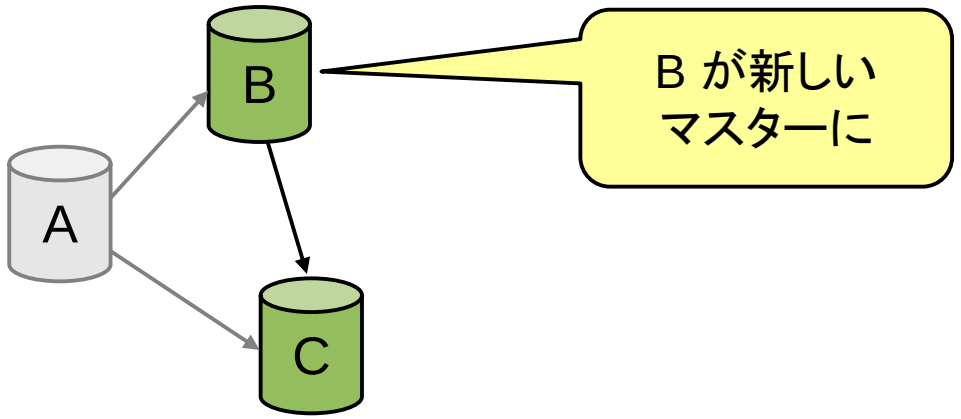
# レプリケーションの利点：高可用性構成の実現

- マスターの障害時に、スレーブをマスターに昇格することで高可用性を実現可能



# レプリケーションの利点：高可用性構成の実現

- マスターの障害時に、スレーブをマスターに昇格することで高可用性を実現可能



# レプリケーションの利点：地理的冗長性の実現

- 地理的に離れた場所に、災害対策サイトを構築可能

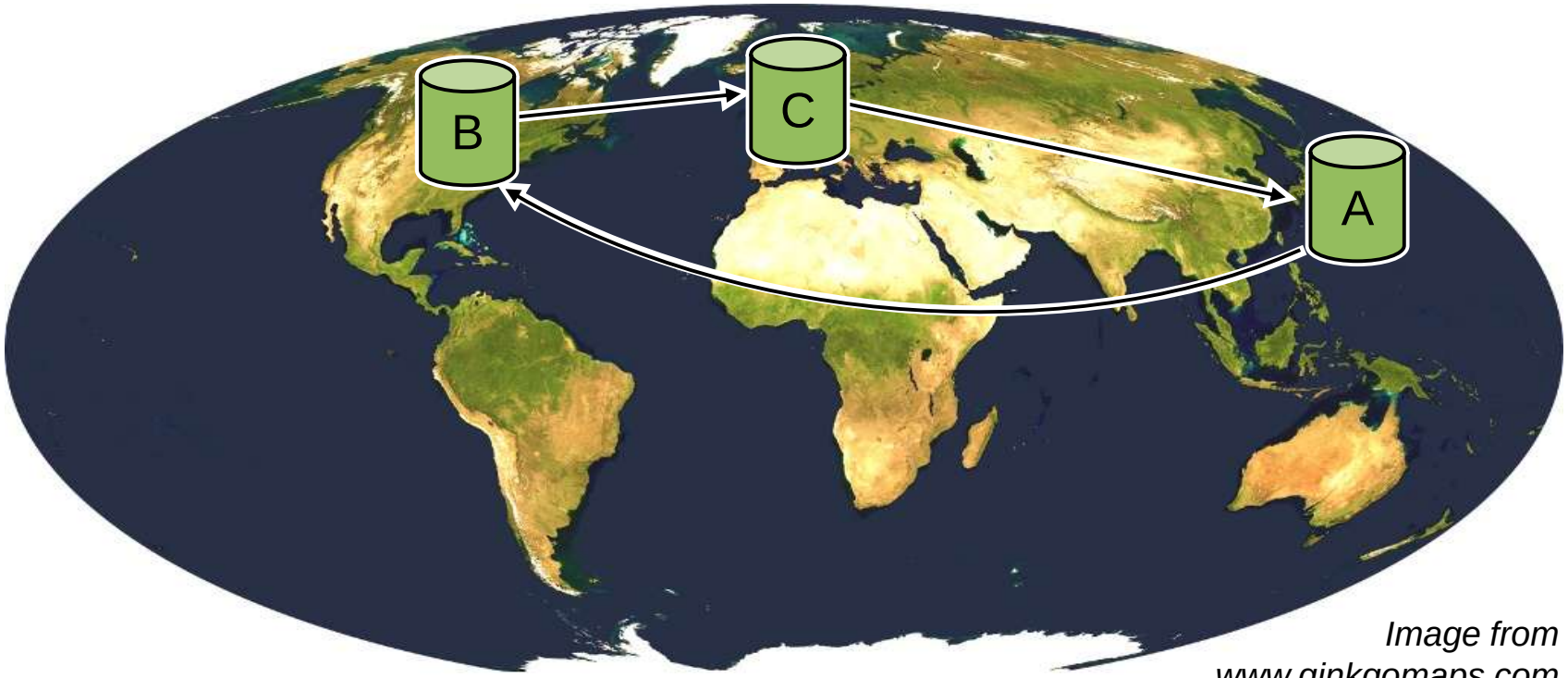
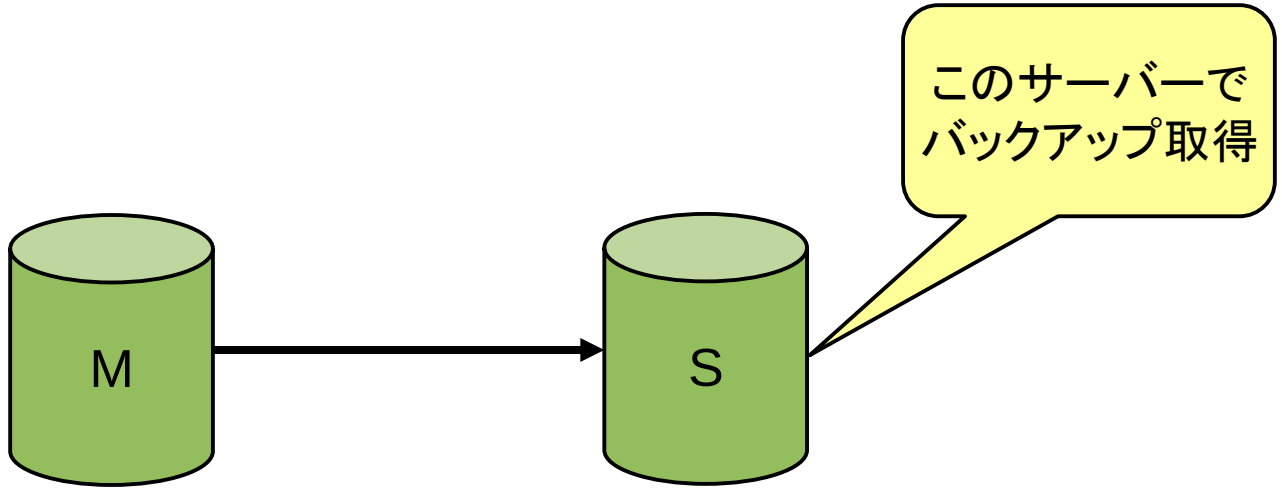



*Image from www.ginkgomaps.com*

# レプリケーションの利点：バックアップサーバーとしての利用

- スレーブサーバーでバックアップを取得することで、
  マスターサーバーに影響を与えずにバックアップ取得可能
  - 例）マスターサーバーは常時稼働させつつ、スレーブサーバーでDBを停止して
    コールドバックアップを取得する

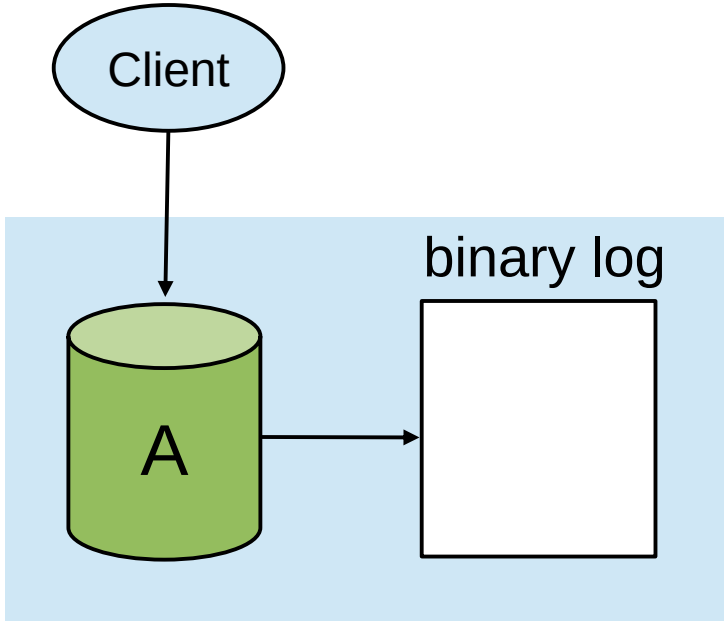


# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

# レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する

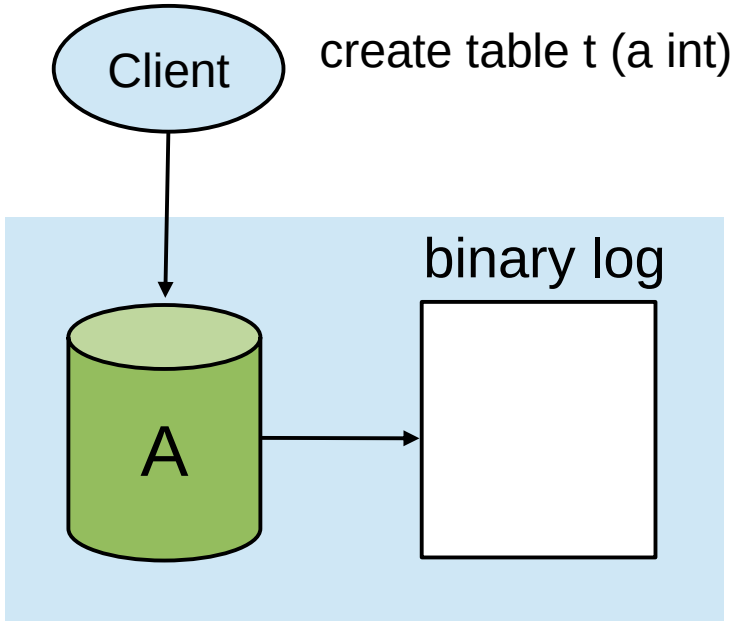


# バイナリログ

- 発行されたクエリのうち、更新系のSQL文のみを記録しているログファイル
  - クエリ実行日時などのメタデータも記録
  - トランザクションのコミット時に同期的に記録 (**sync_binlog=1**)
- バイナリ形式で記録
  - **mysqlbinlog**コマンドにてテキスト化が可能
- 起動オプションを指定して、出力する
  - **--log-bin[=file_name]**
  - 通常の運用時には利用することを推奨
  - データディレクトリとは別のディスクに出力することを推奨
- ログファイル名の拡張子に通し番号を記録
  例）**file_name-bin.001, file_name-bin.002**, etc.
  - 現在利用中のログ番号はインデックスファイルに記録 (**file_name.index**)

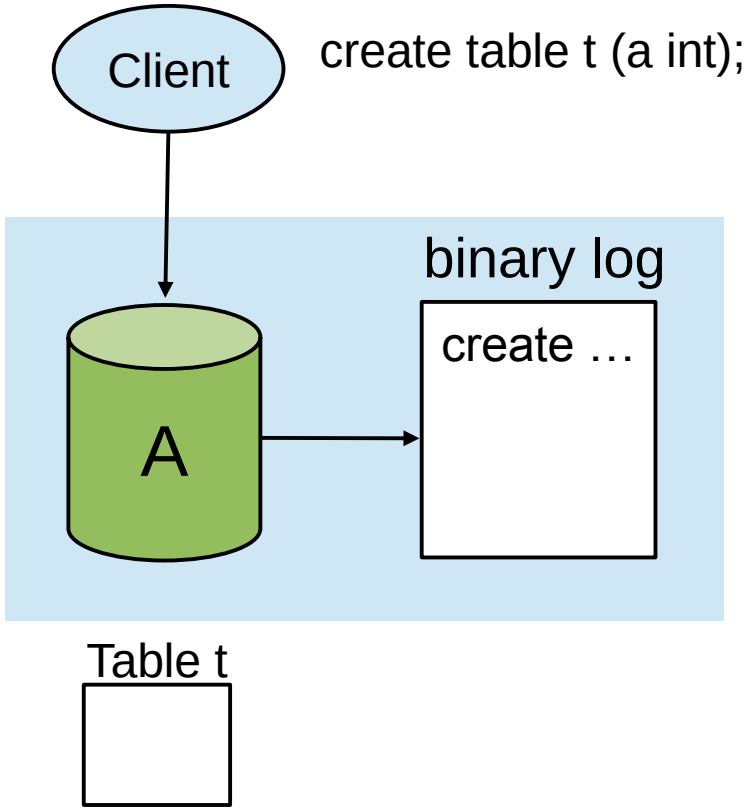
# レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



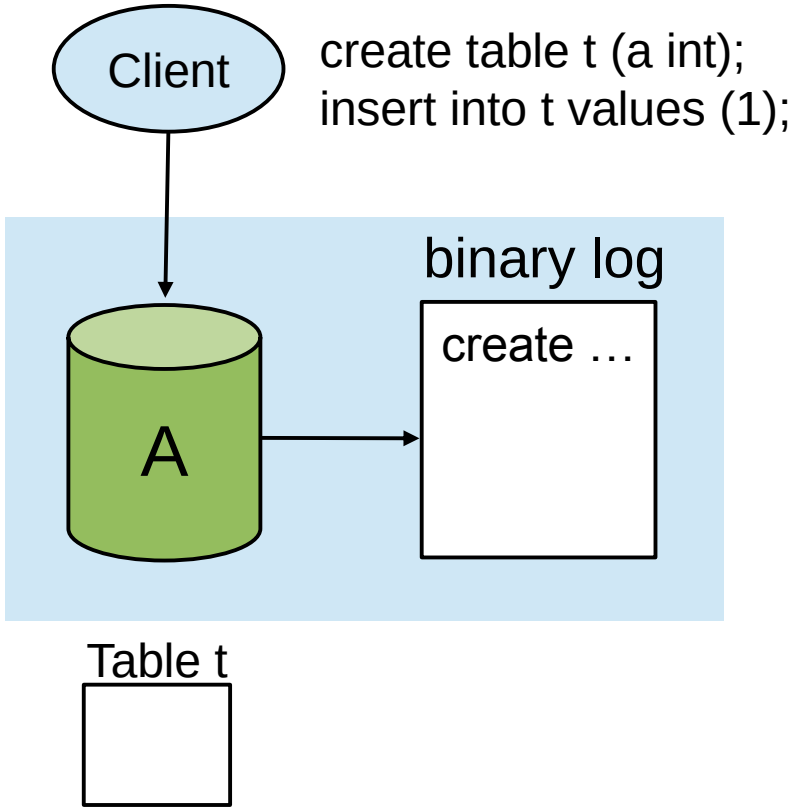
# レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



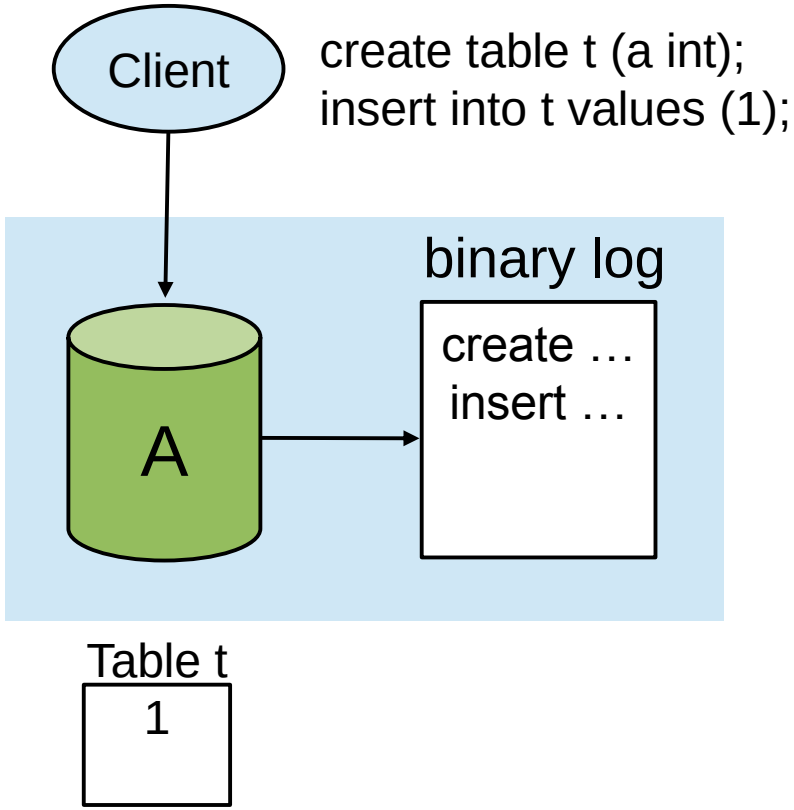
# レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



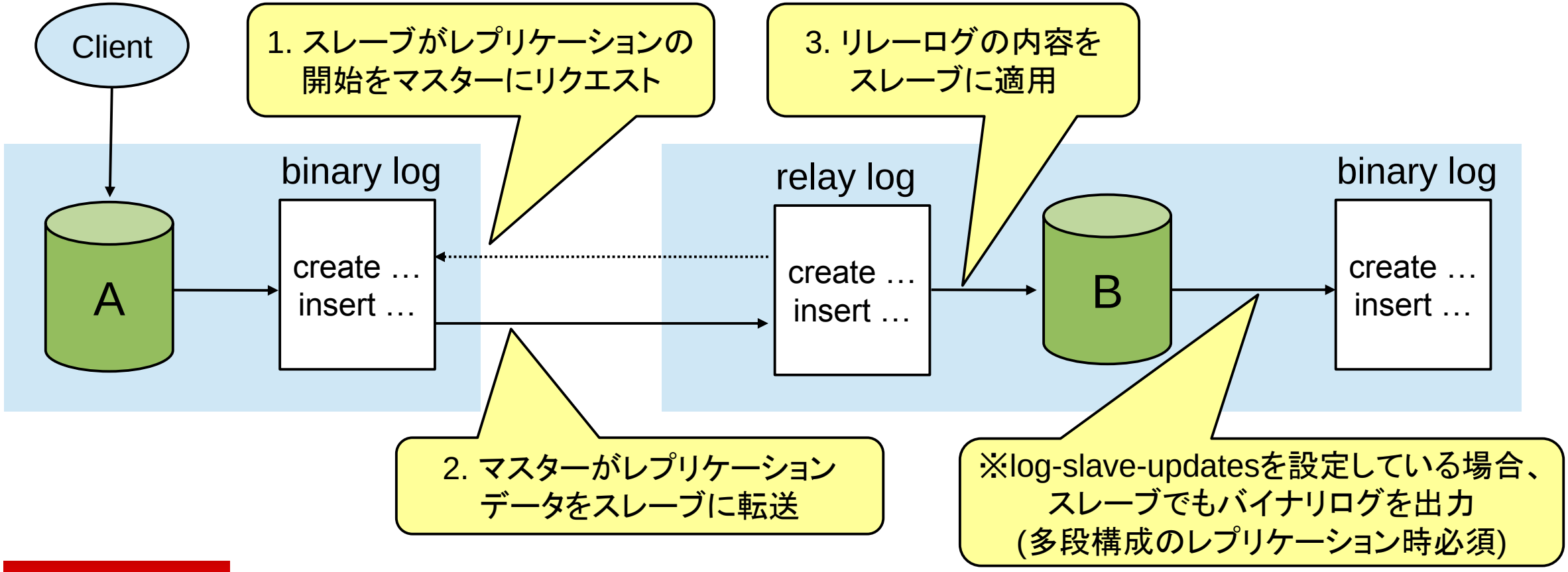
# レプリケーションの仕組み

- マスターサーバーでの全ての変更点をバイナリログに記録する



# レプリケーションの仕組み

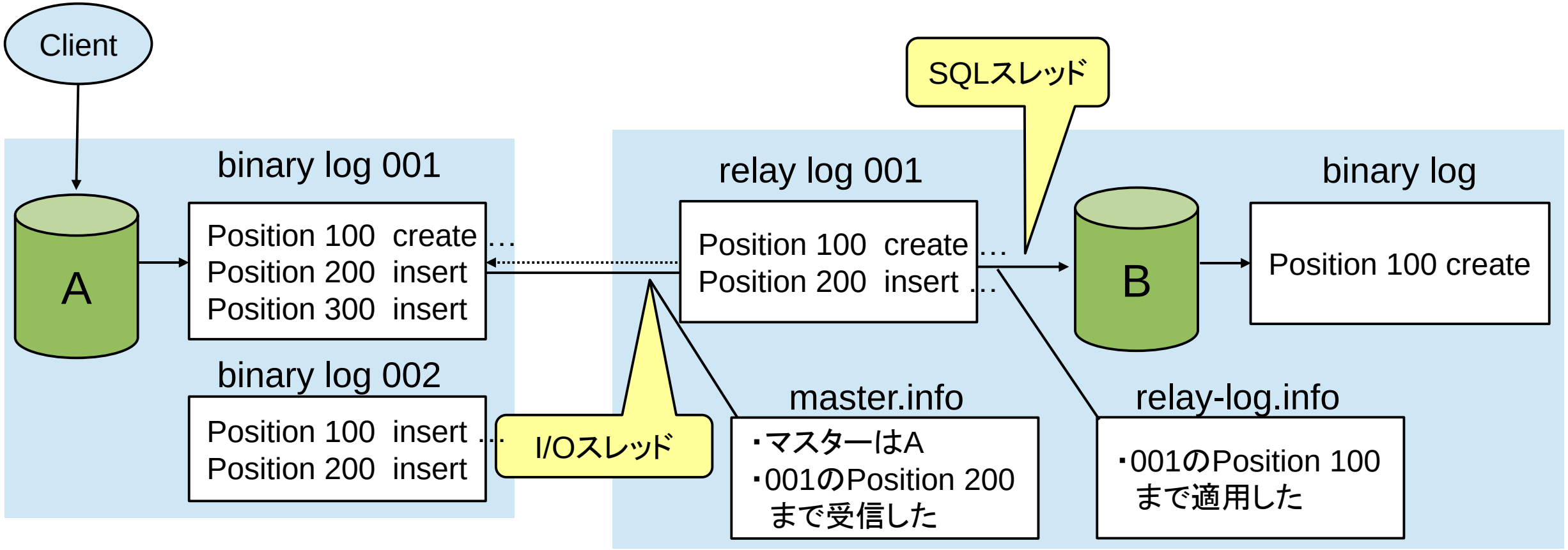
- スレーブからレプリケーション開始
- バイナリログの内容をスレーブに転送し、実行



# スレーブ上に存在するファイル、スレッド

- ファイル
  - リレーログファイル：マスターから受信した変更点を記録したファイル
  - バイナリログファイル：スレーブで実行した変更点を記録したファイル
    (log-slave-updatesを有効にしている場合のみ出力)
  - master.info：マスターへの接続に必要な情報や、読み取りを開始する
    バイナリログの位置情報(バイナリログファイル名とポジション)が
    記録されているOS上のファイル。(MySQL 5.6からテーブル内に格納可能)
  - relay-log.info：リレーログをどこまで適用したかを記録しているOS上のファイル。
    (MySQL 5.6からテーブル内に格納可能)
- スレッド
  - I/Oスレッド：マスターから受信したバイナリログをリレーログファイルとして保存
  - SQLスレッド：リレーログファイル内の更新内容をDBへ反映する

# イメージ図



# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

# レプリケーションの種類

- バイナリログの記録方式による違い
  - STATEMENT：文(SQL文)
  - ROW：行
  - MIXED：文と行が混在
- 同期方式による違い
  - 非同期：変更点を非同期で転送
  - 準同期：変更点を同期で転送し、非同期でDBに反映
- GTIDを使用する/しない による違い
  - GTIDを使用しない：従来からあるレプリケーションモード
  - GTIDを使用する：MySQL 5.6で追加されたレプリケーションモード

# バイナリログの記録方式のよる違い

| フォーマット | 説明 | バイナリログのサイズ | Non-deterministic | スレーブでトリガーが動作するか？ |
|---|---|---|---|---|
| SBR (Statement Based Replication) | SQL文がそのままバイナリログに記録される | 小 | × | 動作する |
| RBR (Row Based Replication) | 更新されたデータそのものが記録される | 大 | ○ | 動作しない (トリガーにより変更された行の変更は伝搬される) |
| MBR (Mixed Based Replication) | SBRとRBRを状況に応じて切り換える | 小 | ○ | SBRの場合動作する RBRの場合動作しない |

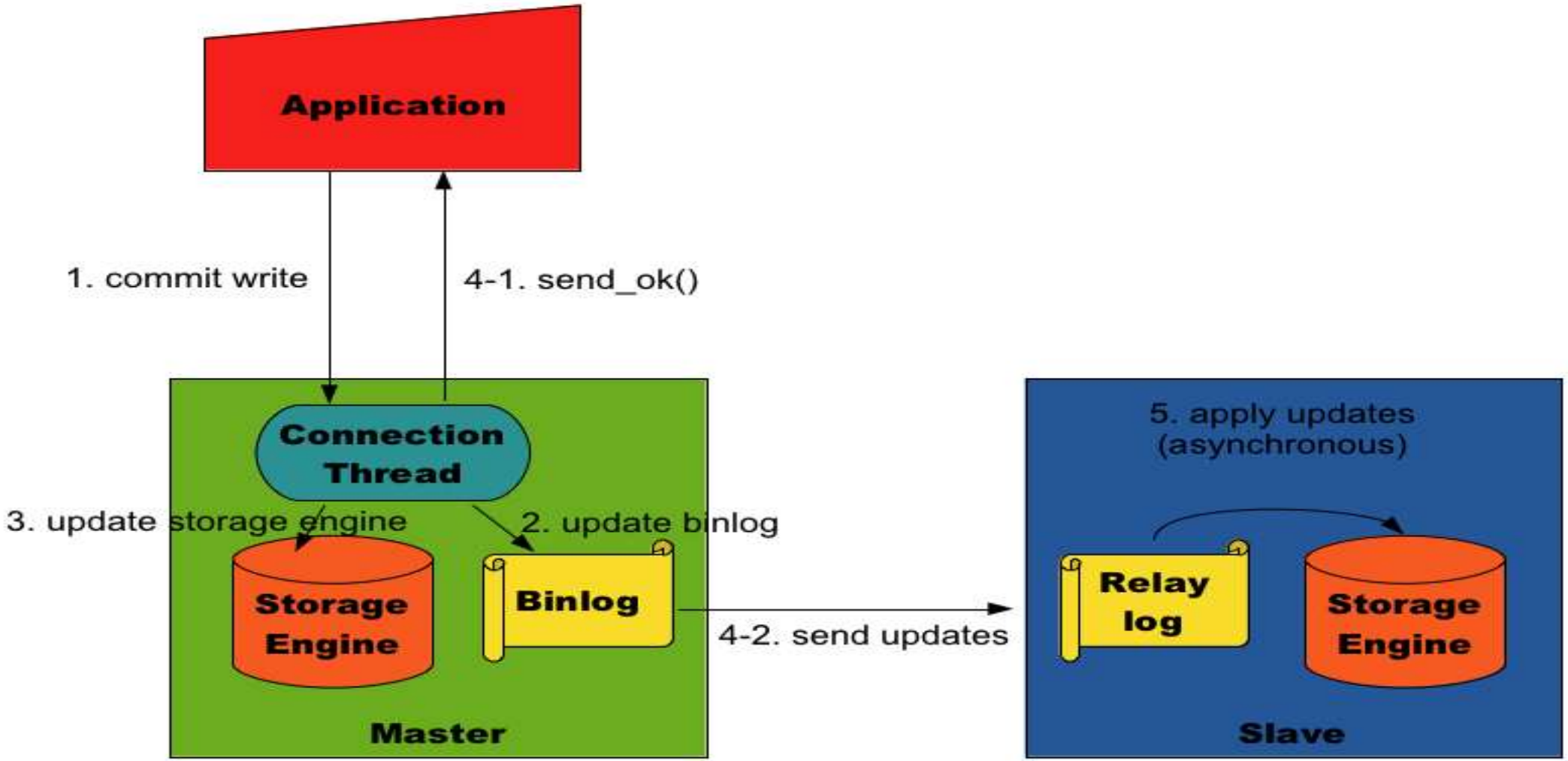
# Non-deterministicとは？

- 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SQL文
  - 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

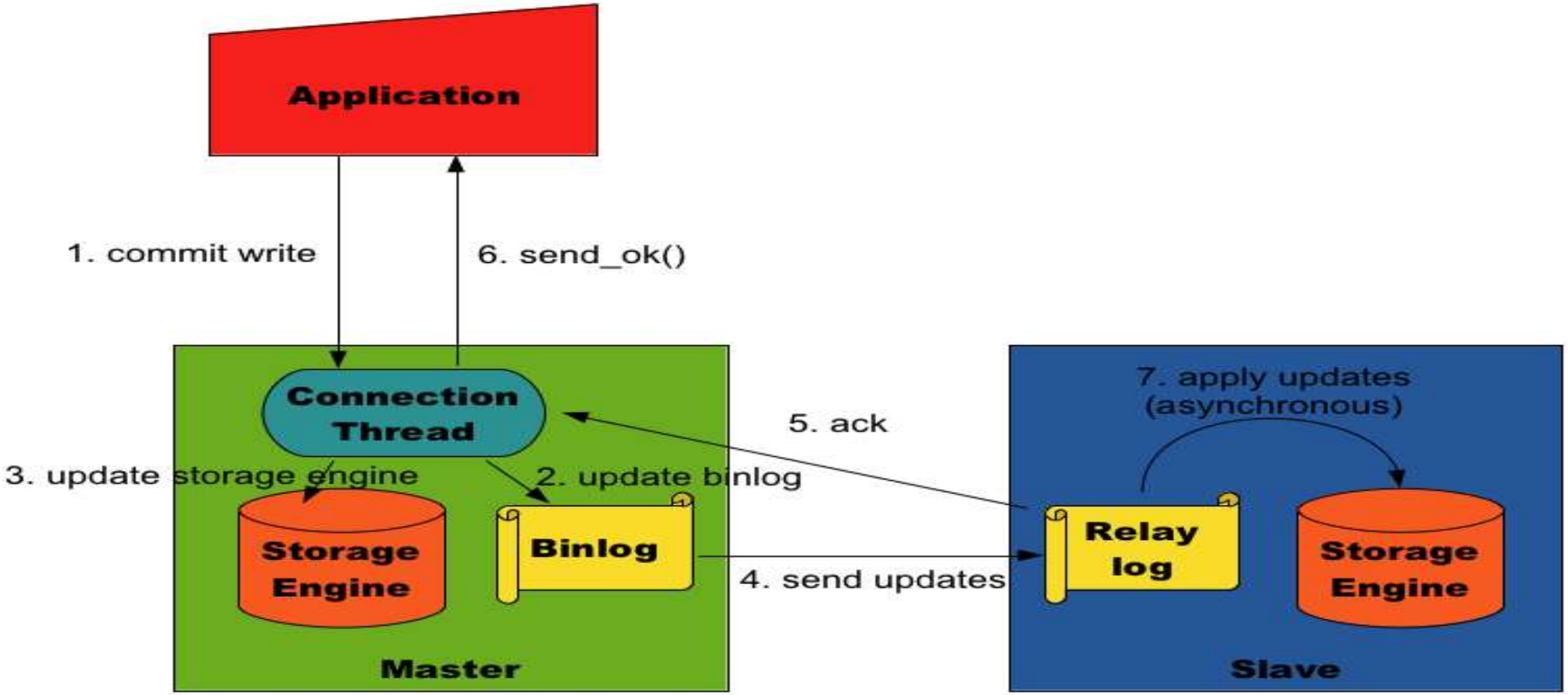
# 同期方式による違い

- 非同期(デフォルト)
  - 変更点を非同期で転送
  - メリット:準同期よりもマスターサーバーの更新処理のレスポンスタイムがいい
  - デメリット:マスターサーバーに障害が発生した場合、障害発生直前の更新内容がスレーブに伝搬されていない可能性がある
  - 特に、負荷分散目的に向く
    (障害発生直前の更新データを保護する必要がある場合は、別途アプリケーション側での対応が必要)
- 準同期(MySQL 5.5から追加された機能)
  - 変更点を同期で転送し、非同期でDBに反映
  - メリット:マスターサーバーに障害が発生した場合、障害発生直前の更新内容もスレーブに伝搬されている
  - デメリット:非同期よりもマスターサーバーの更新処理のレスポンスタイムが悪い
  - 特に、高可用性目的に向く(障害発生直前の更新データもDB側で保護する必要がある場合)

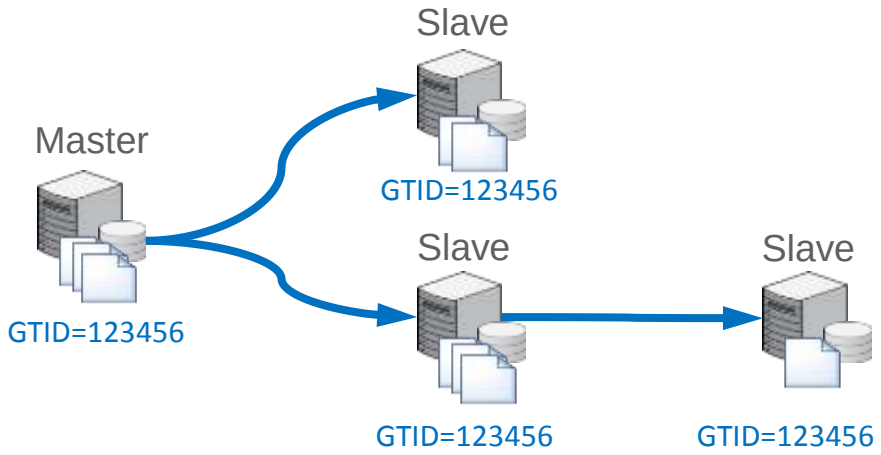
# 非同期レプリケーション



# 準同期レプリケーション



# GTIDを使用する/しないによる違い




- GTID(グローバルトランザクションID)
  - MySQL 5.6で追加された機能
  - 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グローバルで(レプリケーションを構成するMySQLサーバー全てにおいて)一意に識別できる識別子をバイナリログに記録
  - 使用する場合は、レプリケーションの運用方法が従来の方式とは変わる
    - レプリケーション開始時にポジションを自動認識 (master_auto_position=1)
    - フェイルオーバーの為に、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

各サーバーの128bit Server ID　　トランザクションID

```
SET @@SESSION.GTID_NEXT= ' 8560c2ac-e1dc-11e4-88ff-0800275399c1 : 6170 '/*!*/;
```

# GTIDのメリット

- バイナリログのポジションを自動認識してくれるため、指定する必要が無い(マスターに障害が発生してフェイルオーバーする場合など、障害発生時のポジションを確認せずにフェイルオーバー処理を実現できる)
- MySQL Utilities内のmysqlfailoverを使用して自動フェイルオーバーを実現できるなど、以下のGTIDに依存した機能が使える
  - mysqlfailover：レプリケーション環境の自動フェイルオーバー
  - mysqlrpladmin：レプリケーション環境の管理(一部機能のみGTIDに依存)
  - mysqlrplms：ラウンドロビン接続によるマルチソースレプリケーション
  - mysqlrplsync：レプリケーションの同期状況を確認
  - mysqlslavetrx：スレーブでトランザクションをスキップ(※)

※現在開発中のMySQL Utilities 1.6にて追加予定

# GTIDのデメリット

- 以下の制限事項がある
  - マスター/スレーブ共に、InnoDB以外のストレージエンジンは使えない
    (トランザクションに対応したストレージエンジン以外は使えない)
  - “CREATE TABLE ... SELECT“ が使用できない
  - "CREATE TEMPORARY TABLE" および "DROP TEMPORARY TABLE" をトランザクションの中で使用できない(トランザクションを使わない場合は、使用可能)
  - sql_slave_skip_counter はサポートされない
    (トランザクションをスキップしたい場合は、gtid_executed 変数を利用する必要がある)
  - 全スレーブでバイナリログを出力する必要がある

# GTIDのデメリット

- レプリケーションのフィルタリングと併用することが困難である
  - フィルタリングすると、GTIDに欠番が出来るため運用が非常に複雑になる
- GTIDを有効にするためには、全サーバーを一度停止してからGTIDモードに移行する必要があるため、既存環境をGTIDモードに移行し難い(1サーバーずつGTIDモードを有効にできない)
  - 現在開発中のMySQL 5.7では、1サーバーずつGTIDモードを有効にできる

# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. **レプリケーションの設定方法**
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

# レプリケーションの設定方法(GTID無効の場合)

1. レプリケーション用のパラメータを設定
2. マスターサーバーにレプリケーション用ユーザーを作成
3. マスターサーバーのバックアップを取得して、
   スレーブサーバーにリストア
   - バックアップ取得時のバイナリログファイルのファイル名とポジションを記録しておく
4. スレーブサーバーで CHANGE MASTER TO コマンドを実行
5. スレーブサーバーで START SLAVE コマンドを実行

# 1. レプリケーション用のパラメータ設定(GTID無効)

- マスター : 下記オプションを設定して起動
  - server-id
  - log-bin
  - datadir *
- スレーブ : 下記オプションを設定して起動
  - server-id
  - datadir *
  - port *
  - socket *　(Lunix系OSの場合)
  - read_only　(必須ではないが、設定を推奨)

* は、テスト目的で1台のサーバー内でマスター、スレーブを作成する場合に必要な設定

# 2. マスターサーバーにレプリケーション用ユーザーを作成 (GTID無効)

- "REPLICATION SLAVE"権限を付与してユーザーを作成
  - 例

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

# 3. バックアップを取得してスレーブサーバーへリストア (GTID無効)

- コールドバックアップを取得してリストアする
- mysqldumpでバックアップを取得してリストアする
  - **バックアップ取得例**

```
$ mysqldump --user=root --password=root --master-data=2 ¥
    --socket=/usr/local/mysql/data/mysql.sock ¥
    --hex-blob --default-character-set=utf8 --all-databases ¥
    --single-transaction > mysql_bkup_dump.sql
```

※バックアップ取得時のバイナリログファイルのファイル名とポジションを記録しておく

# 補足:mysqldumpのオプション

- **--master-data=2**
  - バックアップ取得のバイナリファイル名とバイナリファイル内の位置(**Position**)を
    コメントとしてバックアップファイルに記録
- **--hex-blob**
  - バイナリ型(**BINARY**、**VARBINARY**、**BLOG**) と**BIT**型のデータを16進数表記で出力
- **--default-character-set**
  - **mysqldump**がデフォルトで利用するキャラクタセットを指定。
    通常はMySQLサーバのシステム変数**default-character-set**と同じものを指定すれば良い
- **--all-databases**
  - 全てのデータベースをバックアップ
- **--lock-all-tables**
  - 全てのテーブルをロックしてバックアップを取得する
- **--single-transaction**
  - InnoDBがサポートしているトランザクションの仕組みを利用して、
    InnoDBテーブルに限り一貫性のとれたバックアップを取得する

# 注意事項：mysqldumpによるバックアップ

- データの整合性を保つために、バックアップ取得中は、テーブルに関するDDL文（※）を実行しないこと

  ※ALTER TABLE, CREATE TABLE, DROP TABLE, RENAME TABLE, TRUNCATE TABLE

- マニュアルの“--single-transaction”オプションの説明部分より引用
  - 「While a --single-transaction dump is in process, to ensure a valid dump file (correct table contents and binary log coordinates), no other connection should use the following statements: ALTER TABLE, CREATE TABLE, DROP TABLE, RENAME TABLE, TRUNCATE TABLE.」

# 4. スレーブサーバーで CHANGE MASTER TO コマンドを実行
# 5. スレーブサーバーで START SLAVE コマンドを実行 (GTID無効)

- CHANGE MASTER TO コマンドを実行
- START SLAVE コマンドを実行
  - 例

```
CHANGE MASTER TO MASTER_HOST='localhost',
 -> MASTER_USER='repl',
 -> MASTER_PASSWORD='repl',
 -> MASTER_LOG_FILE='bin.000001',
 -> MASTER_LOG_POS=1790;
START SLAVE;
```

※青字部分は、バックアップ取得時に記録したファイル名とポジションを指定

※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)

# レプリケーションの設定方法(GTID有効)

1. レプリケーション用のパラメータを設定
2. マスターサーバーにレプリケーション用ユーザーを作成
3. マスターサーバーのバックアップを取得して、
   スレーブサーバーにリストア
   - バイナリログファイルのファイル名とポジションを記録する必要無し
4. スレーブサーバーで CHANGE MASTER TO コマンドを実行
5. スレーブサーバーで START SLAVE コマンドを実行

# 1. レプリケーション用のパラメータ設定(GTID有効)

- マスター：下記オプションを設定して起動
  - server-id
  - log-bin
  - datadir *
  - gtid-mode=on
  - enforce-gtid-consistency=on
  - log-slave-updates

* は、テスト目的で1台のサーバー内でマスター、スレーブを作成する場合に必要な設定
※GTIDモードにする場合は、赤字のパラメータをマスター/スレーブの両方で指定する

# 1. レプリケーション用のパラメータ設定(GTID有効)

- スレーブ : 下記オプションを設定して起動
  - server-id
  - log-bin
  - datadir *
  - port *
  - socket *　(Lunix系OSの場合)
  - read_only　(必須ではないが、設定を推奨)
  - gtid-mode=on
  - enforce-gtid-consistency=on
  - log-slave-updates

# 2. マスターサーバーにレプリケーション用ユーザーを作成 (GTID有効)

- "REPLICATION SLAVE"権限を付与してユーザーを作成
  - 例

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

# 3. バックアップを取得してスレーブサーバーへリストア (GTID有効)

- コールドバックアップを取得してリストアする
  - datadir配下のauto.cnfを削除しておく
    (マスターとスレーブでserver-uuidを一意にするため(※))

    ※GTIDのフォーマットにはserver-uuidが含まれているため、server-uuidを一意にしておく必要あり
- mysqldumpでバックアップを取得してリストアする
  - **バックアップ取得例**

```
$ mysqldump --user=root --password=root –master-data=2 ¥
    --socket=/usr/local/mysql/data/mysql.sock ¥
    --hex-blob --default-character-set=utf8 --all-databases ¥
    --single-transaction --triggers --routines --events > mysql_bkup_dump.sql
```

※Warning発生を防ぐために”--triggers --routines --events”も指定
※GTIDモードで取得したmysqldumpには、”SET @@GLOBAL.GTID_PURGED=‘XXX’;” が含まれる

# 4. スレーブサーバーで CHANGE MASTER TO コマンドを実行
# 5. スレーブサーバーで START SLAVE コマンドを実行 (GTID有効)

- CHANGE MASTER TO コマンドを実行
- START SLAVE コマンドを実行
  - 例

```
CHANGE MASTER TO MASTER_HOST='localhost',
 -> MASTER_USER='repl',
 -> MASTER_PASSWORD='repl',
 -> MASTER_AUTO_POSITION=1;
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)

# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

# バイナリログの管理

- **SHOW MASTER STATUS** コマンドで現在使用中の
  バイナリログファイル名とポジションを確認
- **SHOW MASTER LOGS** コマンドで全てのバイナリログファイル名を列挙
- **FLUSH [BINARY] LOGS** コマンドまたはMySQL
  サーバの再起動でログファイルのローテーション
- **PURGE MASTER** コマンドで特定の時点までの
  バイナリログを削除
- **RESET MASTER** コマンドで全てのバイナリログを削除

※バイナリログは溜まり続けるファイルであるため、運用の中で定期的に削除が必要
　(expire_logs_daysを設定して、指定した日数を超えたものを自動削除することも可能)
※リレーログは、自動的に削除される(デフォルトで"relay_log_purge=1"になっている)

# バイナリログの管理

```
mysql> SHOW MASTER STATUS;
+--------------+----------+--------------+------------------+
| File         | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+--------------+----------+--------------+------------------+
| MySQL.000007 |      107 |              |                  |
+--------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql>
mysql> SHOW MASTER LOGS;
+--------------+-----------+
| Log_name     | File_size |
+--------------+-----------+
| MySQL.000001 |      1110 |
| MySQL.000002 |      2797 |
...
| MySQL.000007 |       107 |
+--------------+-----------+
7 rows in set (0.00 sec)
```

# バイナリログの管理

```
mysql> FLUSH BINARY LOGS;
Query OK, 0 rows affected (0.42 sec)

mysql> SHOW MASTER STATUS;
+---------------+----------+--------------+------------------+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+---------------+----------+--------------+------------------+
| MySQL.000008 |      107 |              |                  |
+---------------+----------+--------------+------------------+
1 row in set (0.00 sec)

mysql> SHOW MASTER LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| MySQL.000001 |      1110 |
...
| MySQL.000007 |       146 |
| MySQL.000008 |       107 |
+---------------+-----------+
8 rows in set (0.00 sec)
```

# バイナリログの管理

```
mysql> PURGE MASTER LOGS TO 'MySQL.000003';
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW MASTER LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| MySQL.000003 |      2315 |
| MySQL.000004 |       628 |
| MySQL.000005 |      1090 |
| MySQL.000006 |       126 |
| MySQL.000007 |       146 |
| MySQL.000008 |       107 |
+---------------+-----------+
6 rows in set (0.00 sec)
```

# バイナリログの管理

```
mysql> RESET MASTER;
Query OK, 0 rows affected (0.06 sec)

mysql> SHOW MASTER LOGS;
+---------------+-----------+
| Log_name      | File_size |
+---------------+-----------+
| MySQL.000001 |       107 |
+---------------+-----------+
1 row in set (0.00 sec)
```

# バイナリログの管理

- **SHOW BINLOG EVENTS** コマンドでバイナリログファイルの中身を確認

```
mysql>  SHOW BINLOG EVENTS IN 'MySQL.000005';
+--------------+-----+-------------+-----------+-------------+-----------------------------------------------------------------------+
| Log_name     | Pos | Event_type  | Server_id | End_log_pos | Info                                                                  |
+--------------+-----+-------------+-----------+-------------+-----------------------------------------------------------------------+
| MySQL.000005 |   4 | Format_desc |         1 |         120 | Server ver: 5.6.24-enterprise-commercial-advanced-log, Binlog ver: 4 |
| MySQL.000005 | 120 | Query       |         1 |         195 | BEGIN                                                                 |
| MySQL.000005 | 195 | Query       |         1 |         310 | use `world`; insert into world.country(code) values('XXX')            |
| MySQL.000005 | 310 | Query       |         1 |         425 | use `world`; insert into world.country(code) values('YYY')            |
| MySQL.000005 | 425 | Query       |         1 |         540 | use `world`; insert into world.country(code) values('ZZZ')            |
| MySQL.000005 | 540 | Xid         |         1 |         571 | COMMIT /* xid=37 */                                                   |
+--------------+-----+-------------+-----------+-------------+-----------------------------------------------------------------------+
6 rows in set (0.01 sec)
```

```
root@localhost [PrivDB]> SHOW BINLOG EVENTS IN 'mysql.000143';
+--------------+-----+---------------+-----------+-------------+-----------------------------------------------------------------------------------------+
| Log_name     | Pos | Event_type    | Server_id | End_log_pos | Info                                                                                    |
+--------------+-----+---------------+-----------+-------------+-----------------------------------------------------------------------------------------+
| mysql.000143 |   4 | Format_desc   |         1 |         120 | Server ver: 5.6.24-enterprise-commercial-advanced-log, Binlog ver: 4                    |
| mysql.000143 | 120 | Previous_gtids |         1 |         151 |                                                                                         |
| mysql.000143 | 151 | Previous_gtids |         1 |         222 | 8560c2ac-e1dc-11e4-88ff-0800275399c1:1-6216                                             |
| mysql.000143 | 222 | Gtid          |         1 |         270 | SET @@SESSION.GTID_NEXT= '8560c2ac-e1dc-11e4-88ff-0800275399c1:6217'                    |
| mysql.000143 | 270 | Query         |         1 |         370 | use `test`; DELETE FROM `test`.`T_Work_mem01`                                           |
| mysql.000143 | 370 | Query         |         1 |         470 | use `test`; DELETE FROM `test`.`T_Work_mem02`                                           |
| mysql.000143 | 470 | Gtid          |         1 |         518 | SET @@SESSION.GTID_NEXT= '8560c2ac-e1dc-11e4-88ff-0800275399c1:6218'                    |
| mysql.000143 | 518 | Query         |         1 |         642 | use `PrivDB`; alter table Show_Priv_Row_MyISAM engine=MyISAM                            |
| mysql.000143 | 642 | Gtid          |         1 |         690 | SET @@SESSION.GTID_NEXT= '8560c2ac-e1dc-11e4-88ff-0800275399c1:6219'                    |
| mysql.000143 | 690 | Query         |         1 |         773 | BEGIN                                                                                   |
| mysql.000143 | 773 | Intvar        |         1 |         805 | INSERT_ID=12                                                                            |
| mysql.000143 | 805 | Query         |         1 |         958 | use `PrivDB`;insert into Show_Priv_Row_MyISAM(name,description) values('D','説明列')    |
| mysql.000143 | 958 | Query         |         1 |        1042 | COMMIT                                                                                  |
+--------------+-----+---------------+-----------+-------------+-----------------------------------------------------------------------------------------+
13 rows in set (0.00 sec)
```

# レプリケーション管理のためのコマンド(スレーブ側)

- **START SLAVE [*SLAVE_TYPE*]** コマンドでスレーブ起動
- **STOP SLAVE [*SLAVE_TYPE*]** コマンドでスレーブ停止
- **SHOW SLAVE STATUS**コマンドでスレーブの状態を確認
  - I/Oスレッドによってバイナリログファイルを何処まで転送出来ているか
  - SQLスレッドによってリレーログ内のSQLを何処まで実行したか
- **STOP SLAVE; SET GLOBAL SQL_SLAVE_SKIP_COUNTER=1; STAT SLAVE;** で次のイベント(トランザクション)をスキップ
  - 運用中に何らしかの原因でレプリケーションエラーが発生した場合、状況を確認後、特定のトランザクションをスキップすることで回避する場合、などに使用
  - GTIDモードの場合は、この方法は使用できない

【GTIDモード場合】stop slave; SET GTID_NEXT = “128bitサーバーID:トランザクションID";
begin;commit;SET GTID_NEXT = AUTOMATIC; start slave;

# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報

# その他の考慮事項

- 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
    =>MySQL 5.6にて、自動フェールオーバーできるスクリプトを提供(MySQL Utilities)
  - 更新と参照の処理を振り分ける仕組み、スレーブ間でのロードバランスの仕組み
    =>Connector/J(Java)やmysqlnd_ms(PHP)などで制御可能
    =>MySQL Fabricでも制御可能

# Connector/J(MySQLのJDBCドライバ)のロードバランス/フェイルオーバー機能

- 接続URLを以下の形式で指定することで、各種機能を利用可能
- jdbc:mysql://primary,failover-1,failover-2...
  - 通常利用するサーバが停止すると、他のサーバにフェールオーバーする
- jdbc:mysql:replication://master,slave1,slave2…
  - レプリケーション構成において、更新処理はマスタにて実行され、参照処理はスレーブ間で分散する
- jdbc:mysql:loadbalance://server1,server2…
  - MySQL Cluster(NDB)やマルチマスタレプリケーションの構成の場合、参照更新処理を全てのノードに分散する

**マニュアル：MySQL Connector/J Developer Guide :: 8 Multi-Host Connections**
**http://dev.mysql.com/doc/connector-j/en/connector-j-multi-host-connections.html**

# Connector/J(MySQLのJDBCドライバ)のロードバランス/フェイルオーバー機能

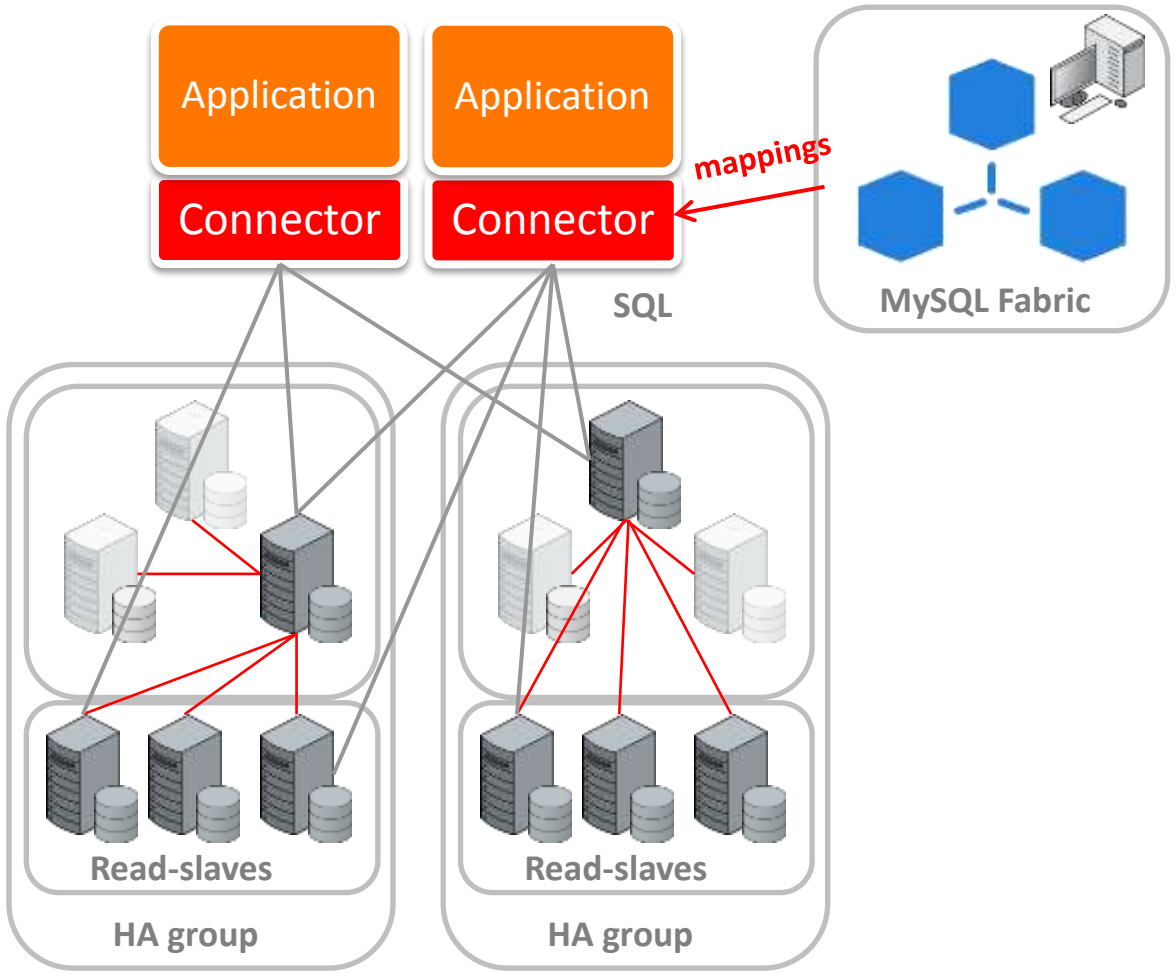
- プロパティの補足
- autoReconnect=true
  - 接続が切れた時に再接続を行う
    (接続が切れた状態でクエリ実行した場合、クエリは実行されずに
    SQLExceptionがスローされる)
- failOverReadOnly=false
  - 別のノードに接続先を変更した場合に、読み取り専用モードにするか否かを制御する
- roundRobinLoadBalance=true
  - 障害発生時(フェイルオーバー時)に次のサーバーから接続を試みる
    (前頁の例では、primaryとの接続が切れた場合に、再度primaryへ接続することなく、
    failover-1へ接続を試みる)

# mysqlnd (MySQL native driver for PHP)の拡張機能

- mysqlnd_ms(Master Slave)
  - mysqlndのプラグインで、ロードバランスやマスター/スレーブの振り分け、フェイルオーバーに対応可能

- 参考
  - mysqlnd レプリケーションおよびロードバランシング用プラグイン
    http://php.net/manual/ja/book.mysqlnd-ms.php
  - Yakst - mysqlnd_msによるシンプルなMySQLのマスタHA
    http://yakst.com/ja/posts/654

# MySQL Fabric 1.5: 高可用性 & シャーディング




- OpenStack との統合
- 高可用性
  - サーバの監視; スレーブの自動昇格と透過的なレプリケーション切り替え
- シャーディングによる拡張性
  - アプリケーションがシャードのキーを提供
    - 整数型、日付型、文字列型
  - レンジまたはハッシュ
  - シャード再構成可能
- Fabric対応コネクタ利用: Python, Java, PHP, .NET, C (labs)
  - プロキシを使わないので低レイテンシ、ボトルネック無し

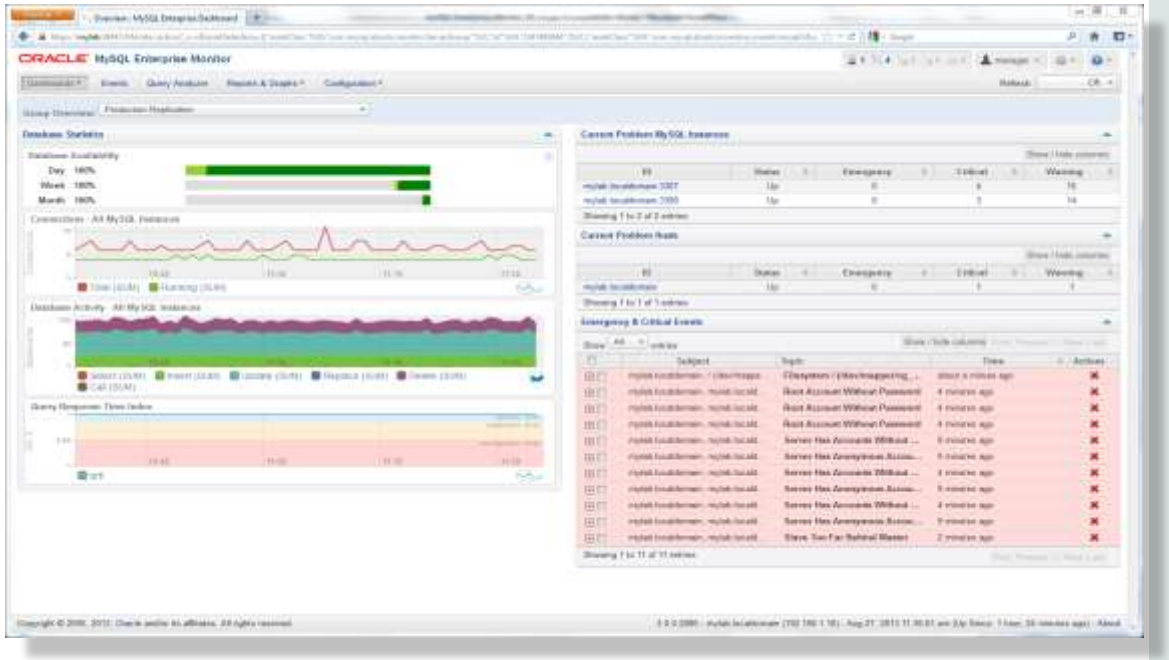
# その他の考慮事項

- 一度に大量の更新処理を実行しない(トランザクションを細かく分割する)
  - スレーブの遅延を防ぐための工夫
  - マスターのトランザクションがコミットされてから、その内容がスレーブに転送されるため、トランザクション実行に時間がかかる場合は、その分スレーブへの反映も遅くなる
- レプリケーションが正しく運用できているか監視する
  ⇒ MySQL Enterprise Monitorで監視可能

# MySQL Enterprise Monitor

- 複数のMySQLサーバを一括監視可能なダッシュボード
- システム中のMySQLサーバやレプリケーション構成を自動的に検出し監視対象に追加
- ルールに基づく監視と警告
- **問題が発生する前に通知**
- 問題のあるSQL文の検出、統計情報の分析が可能なQuery Analyzer

参照： MySQL Enterprise Monitor




*“バーチャルなMySQL DBA”アシスタント*

# Replication Monitor

- レプリケーショントポロジーの自動検知
- マスター/スレーブのパフォーマンス監視
- レプリケーションアドバイザーによるサポート
- レプリケーションのベストプラクティスを提示

*"The MySQL Enterprise Monitor is an absolute must for any DBA who takes his work seriously."*

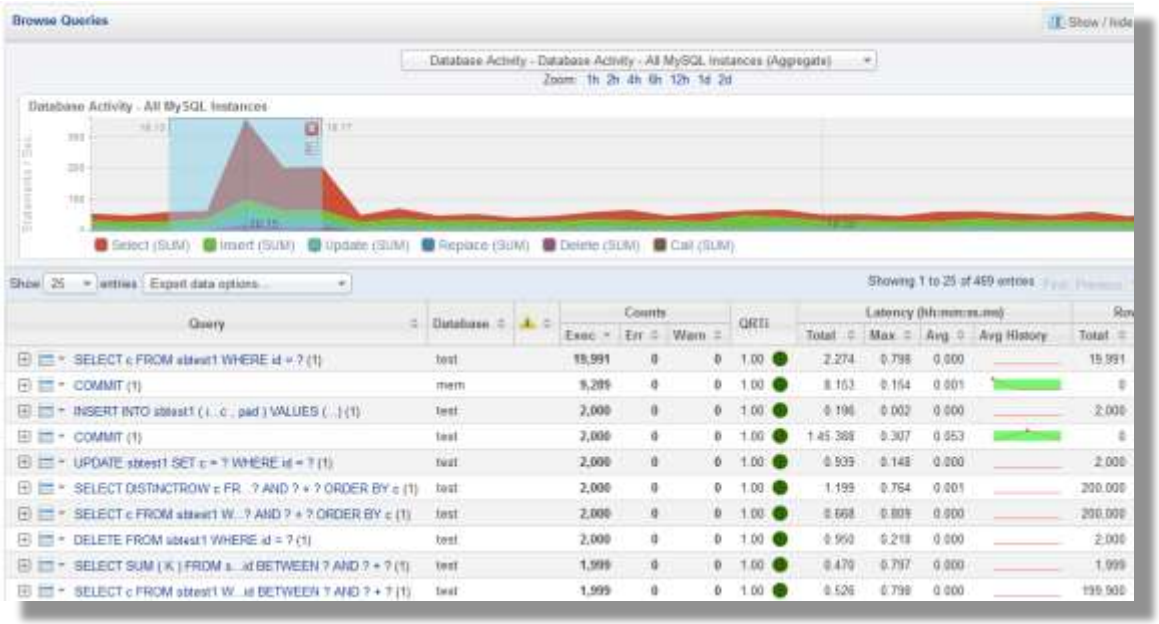
- Adrian Baumann, System Specialist
Federal Office of Information Technology & Telecommunications

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 |

# クエリ解析機能 - MySQL Query Analyzer

- 全てのMySQLサーバの
  全てのSQL文を一括監視
- vmstatなどのOSコマンドやMySQLの
  SHOWコマンドの実行、
  ログファイルの個別の監視は不要
- クエリの実行回数、エラー回数、
  実行時間、転送データ量などを一覧表示
- チューニングのための解析作業を省力化




*"With the MySQL Query Analyzer, we were able to identify and analyze problematic SQL code, and triple our database performance. More importantly, we were able to accomplish this in three days, rather than taking weeks."*

Keith Souhrada
Software Development Engineer
Big Fish Games

| | MySQL Editions | | |
|---|---|---|---|
| | Standard Edition | Enterprise Edition | Cluster CGE |
| 機能概要 | | | |
| MySQL Database | ✔ | ✔ | ✔ |
| MySQL Connectors | ✔ | ✔ | ✔ |
| MySQL Replication | ✔ | ✔ | ✔ |
| MySQL Fablic | | ✔ | ✔ |
| MySQL Partitioning | | ✔ | ✔ |
| MySQL Utilities | | ✔ | ✔ |
| Storage Engine: MyISAM, InnoDB | ✔ | ✔ | ✔ |
| Storage Engine: NDB (ndbcluster) | | | ✔ |
| MySQL Workbench SE/EE* | ✔ | ✔ | ✔ |
| MySQL Enterprise Monitor* | | ✔ | ✔ |
| MySQL Enterprise Backup* | | ✔ | ✔ |
| MySQL Enterprise Authentication (外部認証サポート) * | | ✔ | ✔ |
| MySQL Enterprise Audit (ポリシーベース監査機能) * | | ✔ | ✔ |
| MySQL Enterprise Encryption（非対称暗号化）* | | ✔ | ✔ |
| MySQL Enterprise Firewall（SQLインジェクション対策）* | | ✔ | ✔ |
| MySQL Enterprise Scalability (スレッドプール) * | | ✔ | ✔ |
| MySQL Enterprise High Availability (HAサポート) * | | ✔ | ✔ |
| Oracle Enterprise Manager for MySQL* | | ✔ | ✔ |
| MySQL Cluster Manager (MySQL Cluster管理) * | | | ✔ |
| MySQL Cluster Geo-Replication | | | ✔ |

*商用版のみで利用可能な追加機能

| | MySQL Editions | | |
|---|---|---|---|
| | Standard SE | Enterprise EE | Cluster CGE |
| **Oracle Premium Support** | | | |
| 24時間365日サポート | ✔ | ✔ | ✔ |
| インシデント数無制限 | ✔ | ✔ | ✔ |
| ナレッジベース | ✔ | ✔ | ✔ |
| バグ修正&パッチ提供 | ✔ | ✔ | ✔ |
| コンサルティングサポート | ✔ | ✔ | ✔ |
| **オラクル製品との動作保証** | | | |
| Oracle Linux | ✔ | ✔ | ✔ |
| Oracle VM | ✔ | ✔ | ✔ |
| Oracle Solaris | ✔ | ✔ | ✔ |
| Oracle Enterprise Manager | | ✔ | ✔ |
| Oracle GoldenGate | | ✔ | ✔ |
| Oracle Data Integrator | | ✔ | ✔ |
| Oracle Fusion Middleware | | ✔ | ✔ |
| Oracle Secure Backup | | ✔ | ✔ |
| Oracle Audit Vault and Database Firewall | | ✔ | ✔ |

※最新の対比表は、MySQL Editionsのサイトを参照下さい。

# MySQL Enterprise Edition管理ツールと拡張機能概要

| MySQL Enterprise Edition | |
|---|---|
| MySQL Enterprise Monitor | 複数サーバの一括管理、クエリ性能分析 |
| MySQL Enterprise Backup | **高速**なオンラインバックアップ、ポイントインタイムリカバリ |
| MySQL Enterprise Scalability | Thread Poolプラグインによる**性能拡張性**の向上 |
| MySQL Enterprise Authentication | LDAPやWindows Active Directoryとの**外部認証と統合管理** |
| MySQL Enterprise Audit | ユーザ処理の**監査**、Oracle DBと同じツールで管理可能 |
| MySQL Enterprise Encryption | 非対称暗号化(公開鍵暗号)の業界標準機能を提供 |
| MySQL Enterprise Firewall | **SQLインジェクション対策** |
| Oracle Enterprise Manager for MySQL | Oracle Enterprise ManagerからMySQLを統合管理可能 |
| Oracle Premier Support | 24x7, **インシデント無制限、コンサルティングサポート** |

# 補足:レプリケーション監視(MySQL Enterprise Monitor未使用)

- SHOW SLAVE STATUSの結果から、主に以下の点を監視
  - I/Oスレッド、SQLスレッドが稼働しているか？
    - I/Oスレッド：Slave_IO_Running
    - SQLスレッド：Slave_SQL_Running
  - レプリケーション遅延が起きていないか？また、起きている場合の詳細状況
    - レプリケーション遅延の有無：Seconds_Behind_Master
    - バイナリログ、リレーログを何処まで転送/実行しているか
      - バイナリログの転送状況：Master_Log_File、Read_Master_Log_Pos
      - リレーログの実行状況：Relay_Master_Log_File、Exec_Master_Log_Pos
    - ネットワーク遅延を確認するためには、マスターでのSHOW MASTER STATUS 実行結果も確認する必要がある。(SHOW SLAVE STATUS の Master_Log_File、Read_Master_Log_Pos と比較)

# 例：SHOW SLAVE STATUSの結果(次ページに続く)

```
root@localhost [test]> show slave status¥G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 192.168.56.201
                  Master_User: repl_user
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000142
          Read_Master_Log_Pos: 838
               Relay_Log_File: GA02-relay-bin.000103
                Relay_Log_Pos: 1064
        Relay_Master_Log_File: mysql-bin.000142
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table: test.T_Work_mem02,test.T_Work_mem01
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 838
              Relay_Log_Space: 1812
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
```

ログポジション

スレッドの状態

フィルタリング

# 例：SHOW SLAVE STATUSの結果(続き)

```
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 1
                  Master_UUID: 8560c2ac-e1dc-11e4-88ff-0800275399c1
             Master_Info_File: mysql.slave_master_info
                    SQL_Delay: 0
          SQL_Remaining_Delay: NULL
      Slave_SQL_Running_State: Slave has read all relay log; waiting for the slave I/O thread to update it
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set: 8560c2ac-e1dc-11e4-88ff-0800275399c1:381-6216
            Executed_Gtid_Set: 1-20:6057-6165:6167-6169:6171-6182:6184-6214:6216
                Auto_Position: 1
1 row in set (0.00 sec)
```

マスターとの遅延(秒)

GTIDの状態

# 補足:レプリケーション監視(MySQL Enterprise Monitor未使用)

- スロークエリーログの監視
  - 遅延の原因となり得る長時間実行されるクエリを確認
- マスター、スレーブのディスク空き容量
  - 特にスレーブのディスク空き容量が減っている場合は、リレーログが削除されていない(レプリケーションが停止している)危険性あり
- マスター、スレーブのサーバーリソース使用状況
  (CPU、メモリ、I/O量、ネットワークトラフィック)

※MySQL 5.6以降で実装されているマルチスレッドスレーブを使わない場合、
スレーブでの更新処理はシングルスレッドで実行されることにも注意

# Program Agenda

1. レプリケーションとは？
2. レプリケーションの仕組み
3. レプリケーションの種類
4. レプリケーションの設定方法
5. バイナリログの管理方法
6. その他の考慮事項
7. 参考情報、補足

# ホワイトペーパー

- MySQLレプリケーション - MySQL 5.5によるスケーラビリティと可用性の強化
  http://www.mysql.com/why-mysql/white-papers/wp-mysql-5-5-replication-ja/

- MySQL 5.6 レプリケーション : 概要
  http://www.mysql.com/why-mysql/white-papers/mysql-replication-ja/

- MySQL 5.6 レプリケーション : チュートリアル - スケーラビリティと可用性の強化
  http://www.mysql.com/why-mysql/white-papers/mysql-replication-tutorial-ja/

# MySQL5.6～ 主なレプリケーションパラメータ オプション

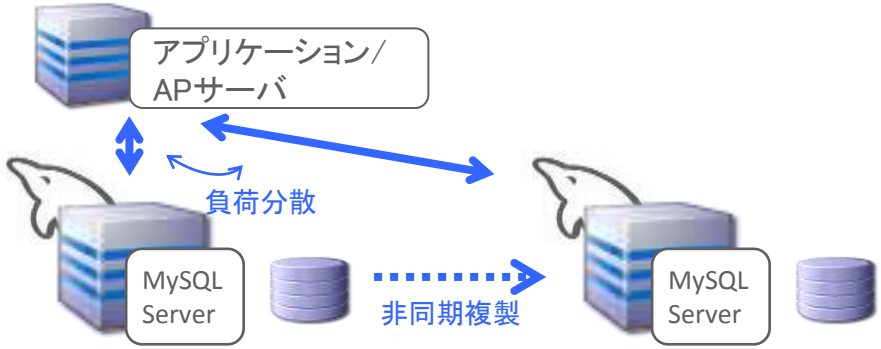
| レプリケーション関連設定 | 概要 |
|---|---|
| binlog_row_image=minimal | 行イメージを全て保持するのではなく、最低限のカラムの情報だけ保持するように、バイナリログのフォーマットを変更可能です。full, minimal, noblob |
| slave_parallel_workers=n | スレーブ側での処理をマルチスレッド化できるため、スレーブの遅延を改善出来る可能性があります。（スキーマ単位） |
| relay_log_info_repository=TABLE | ポジションの情報をInnoDB上のテーブルに記録する為、クラッシュセーフになりました。(“relay_log_recovery=ON”も合わせて設定) |
| relay_log_recovery=ON | リレーログに問題があった時に、マスターから自動的に読み直します。 |
| binlog_checksum=NONE | バイナリーログチェックサム CRC32, NONE (デフォルト値：CRC32) |
| binlog_rows_query_log_events=ON | SQL文に関する情報をバイナリログに追加できます。レプリケーションの追跡や問題発生時のデバッグに役に立ちます。(mysqlbinlog –vv ） |
| MASTER_DELAY=N | CHANGE MASTER実行時に指定。遅延させたい時間を秒単位で指定 |
| MASTER_BIND | CHANGE MASTER実行時に指定。スレーブサーバーが複数のNICを持っている場合、マスターとの接続に使用するNICを明示的に指定できるようになりました。 |

# 補足:レプリケーションのフィルタリング

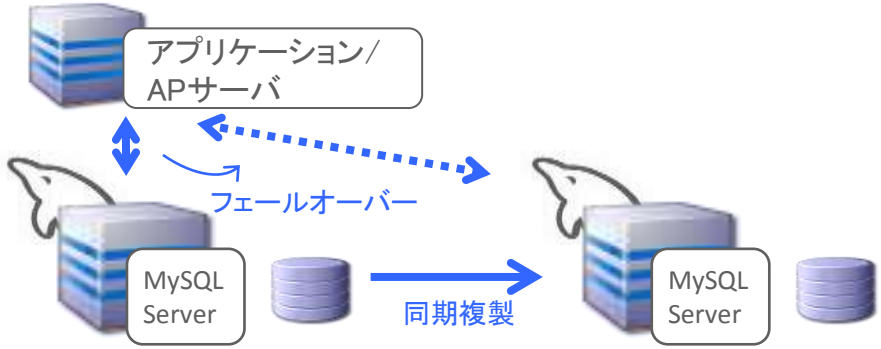
- スレーブ側で受取ったバイナリログから適用する内容を設定できる(my.cnfに記入)
  - replicate-do-db=*db_name*
    - 指定したデータベース(スキーマ)だけをレプリケーション対象にする
  - replicate-do-table=*db_name.tbl_name*
    - 指定したテーブルだけをレプリケーション対象にする
  - replicate-ignore-db=*db_name*
    - 指定したデータベース(スキーマ)をレプリケーション対象から除く
  - replicate-ignore-table=*db_name.tbl_name*
    - 指定したテーブルをレプリケーション対象から除く

# 補足：MySQLの高可用性構成のパターン

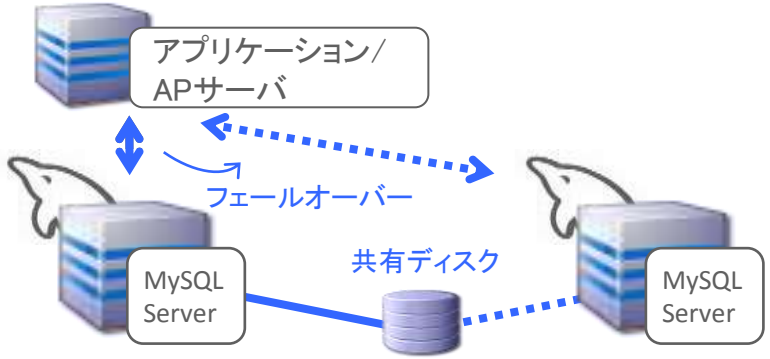
- レプリケーション（標準機能）
  非同期&準同期データレプリケーション



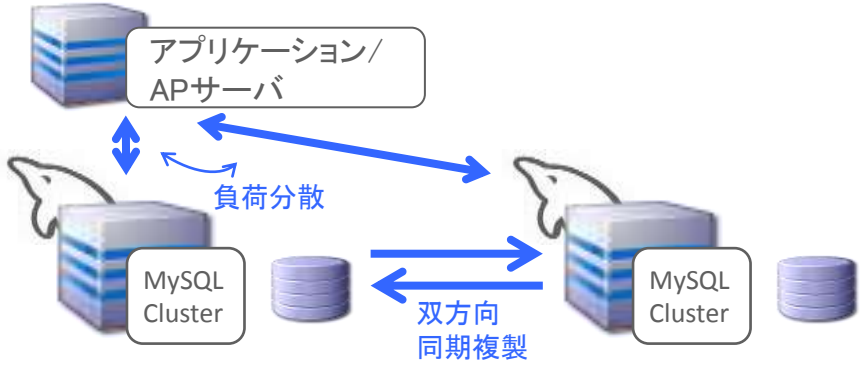

- MySQL+DRBD
  Linux用のノード間データコピー




- Oracle Clusterwareなど
  共有ディスクにデータを格納

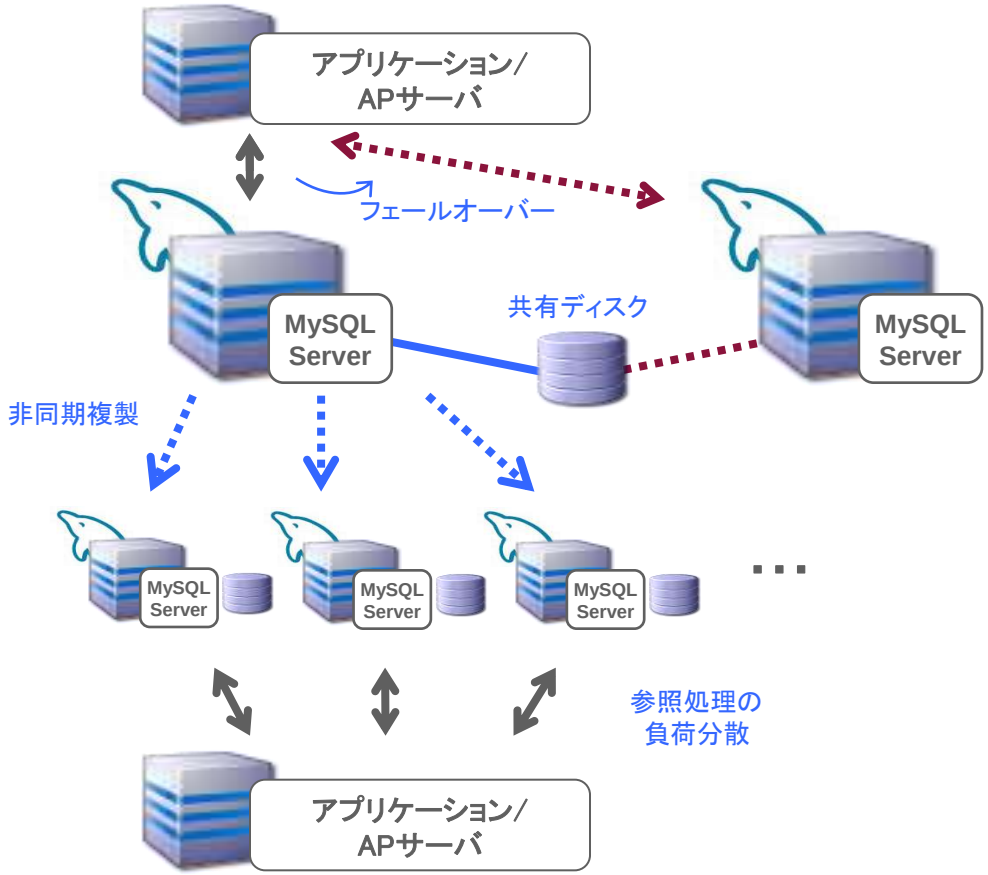



- MySQL Cluster
  シェアードナッシング型高性能クラスタ

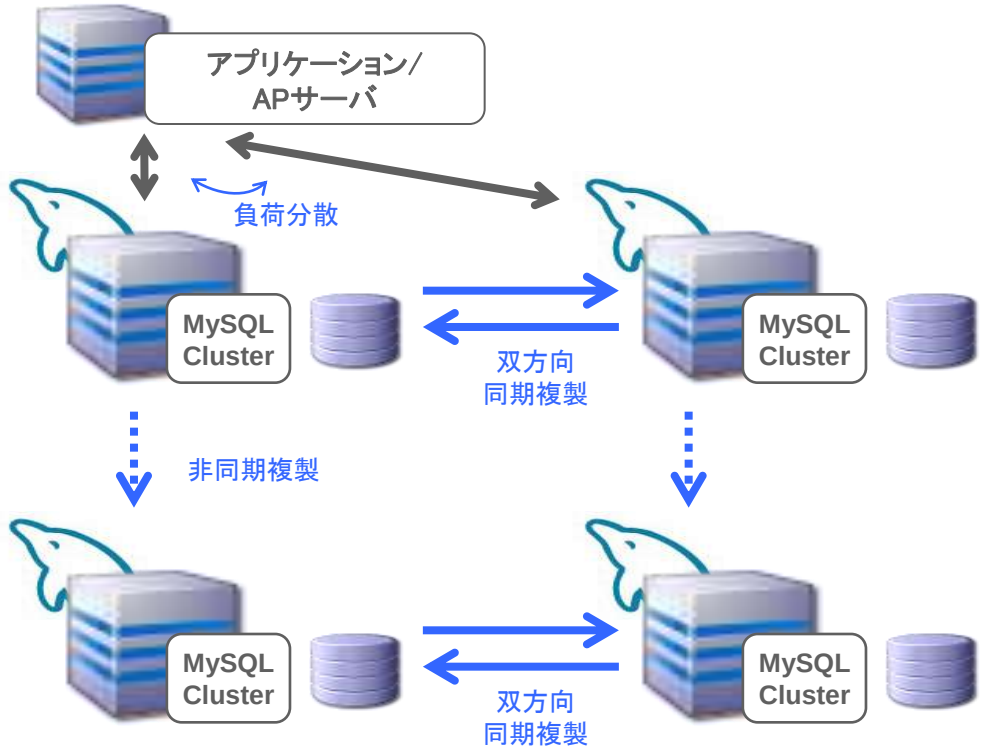


# 補足：複合型の高可用性構成例

- 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



# 補足:レプリケーションによる高可用性構成

- メリット
  - MySQLの標準機能だけで実現でき、共有ディスクや特別なソフトウェアも不要であるため、共有ディスクを使った方式に比べH/Wコスト、ソフトウェアコストを低く抑えられる
  - 参照処理の負荷分散と高可用性構成を同じ仕組みで実現できる(スレーブを参照処理でも活用すれば、H/Wリソースの有効活用にもつながる)
- デメリット
  - フェイルオーバー処理を別途実現する必要があるなど、運用時の考慮事項が多い
    - 障害発生時に、どのようにしてフェイルオーバー処理を実行するか？
    - フェイルオーバーによってMySQLサーバーの構成が変わった場合、アプリケーションからMySQLサーバーへの接続先を切り替える必要がある
    - (MySQL 5.5以前の場合) スレーブがクラッシュセーフでないため、スレーブに障害が発生するとスレーブを再構築しないといけない場合がある

# 補足:レプリケーションによる高可用性構成

- デメリットに対する改善機能
  - フェイルオーバー処理の自動化
    - GTIDモードでレプリケーションを構成すると、MySQL Utilities内のmysqlfailoverを使用することで、自動フェイルオーバーが可能になる(mysqlfailoverの実体はPythonスクリプト)
  - アプリケーションからの接続先切り換えの必要性
    - MySQL Utilities内のMySQL Fabricを使用することで、フェイルオーバー処理を自動化でき、フェイルオーバー後にもアプリケーションからの接続先変更が不要になる(内部的に、GTIDモードによるレプリケーションを使用)
  - (MySQL 5.5以前の場合) スレーブがクラッシュセーフでないため、スレーブに障害が発生するとスレーブを再構築しないといけない場合がある
    - MySQL 5.6で以下のパラメータを設定することで、クラッシュセーフなスレーブが実現できる(※)
      - relay_log_recovery = ON
      - relay_log_info_repository = TABLE

※マスター/スレーブ共に、InnoDBを使用する必要あり

# 補足：MySQL+DRBDによる高可用性構成

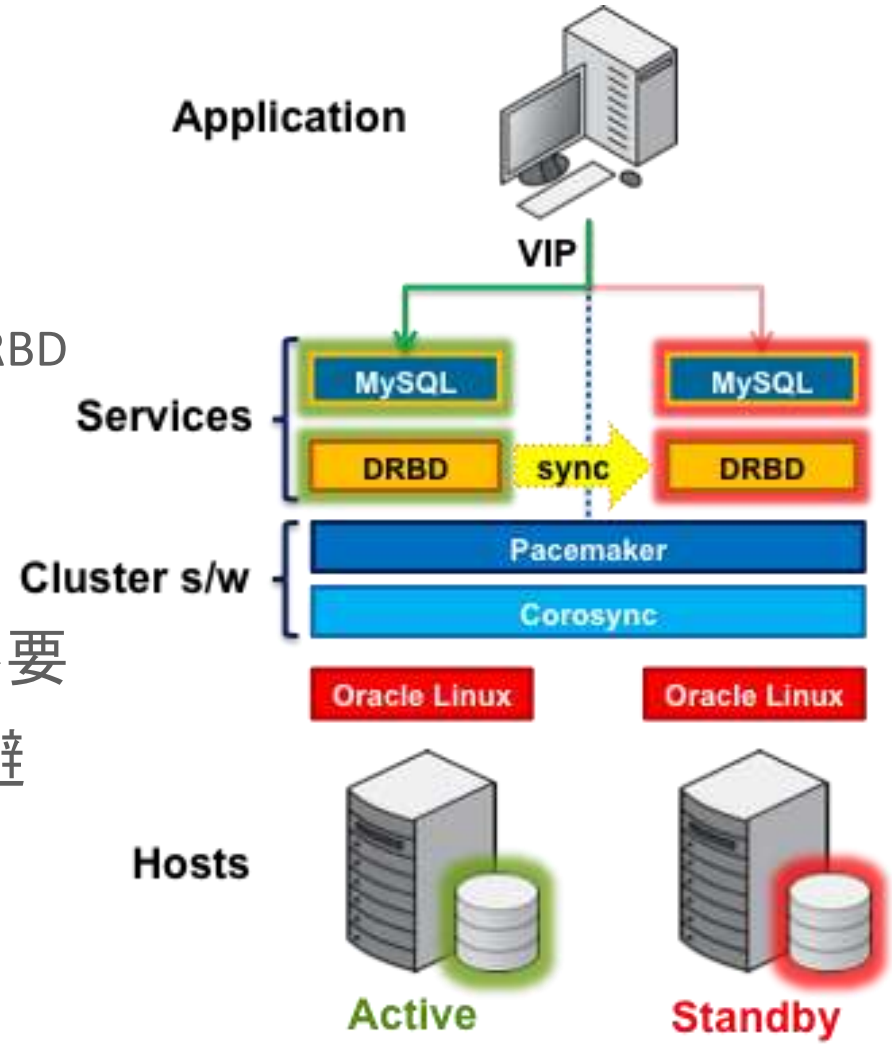
- メリット
  - 共有ディスクが不要であり、共有ディスクを使用した高可用性構成に比べてH/Wコストを低く抑えられる
- デメリット
  - フェイルオーバー処理を別途実現する必要があるなど、運用時の考慮事項が多い
    - 障害発生時に、どのようにしてフェイルオーバー処理を実行するか？
      - DRBDで同期しているディスク領域と同期していないディスク領域が混在することにも注意が必要
    - プライマリ/スタンバイで同期が取れなくなった場合の対応
      - MySQL以外に、DRDBについても知識が必要
  - スタンバイ機は完全なスタンバイ機となる(レプリケーションでスレーブを参照処理に利用する場合に比べ、H/Wリソースを有効活用できない)

# High Availability with Oracle Linux

## Oracle Linux ＋ DRBD Stack

- 認定構成だからこそ実現できる、
  Oracleによるフルスタックサポート
  - Oracle Linux Unbreakable Enterprise Kernel R2 に統合されたDRBD
  - Oracle Linux 6.2以上で使用可能
  - クラスタリングとフェイルオーバーのために、
    PacemakerとCorosyncを使用
- 分散ストレージを利用するため、共有ディスクやSAN不要
- 同期レプリケーションによってデータを失うリスクを回避
- オープンソースで実績の多いソリューション

※ホワイトペーパー：DRBD - Configuration and Deployment Guide
http://www.mysql.com/why-mysql/white-papers/mysql_wp_drbd.php



# 補足：共有ディスク＋クラスタウェア(Oracle Clusterwareなど)を使った高可用性構成

- メリット
  - フェイルオーバー処理をクラスタウェアで自動制御できるため、運用負荷が低い
  - 共有ディスクにデータがあるため、プライマリ/スタンバイでデータの不整合が起きない
- デメリット
  - 共有ディスクやクラスタウェアを用意する必要があり、その分H/Wコスト、ソフトウェアコストがかかる
    ⇒Oracle Clusterwareを使用することで、ソフトウェアコストを削減可能
  - スタンバイ機は完全なスタンバイ機となる(レプリケーションでスレーブを参照処理に利用する場合に比べ、H/Wリソースを有効活用できない)

# High Availability with Oracle Linux

## Oracle Clusterware ＋ MySQL Enterprise Edition

- Oracle Clusterware 12cに、MySQL対応のエージェントが追加
  - MySQL対応エージェントを使用するためには、MySQL EEが必要
  - Oracle Linuxのサポート契約があれば、Oracle Clusterwareについてもサポートを受けることが可能
    ⇒Oracle Clusterwareを使った高可用性構成が、安価に構築可能

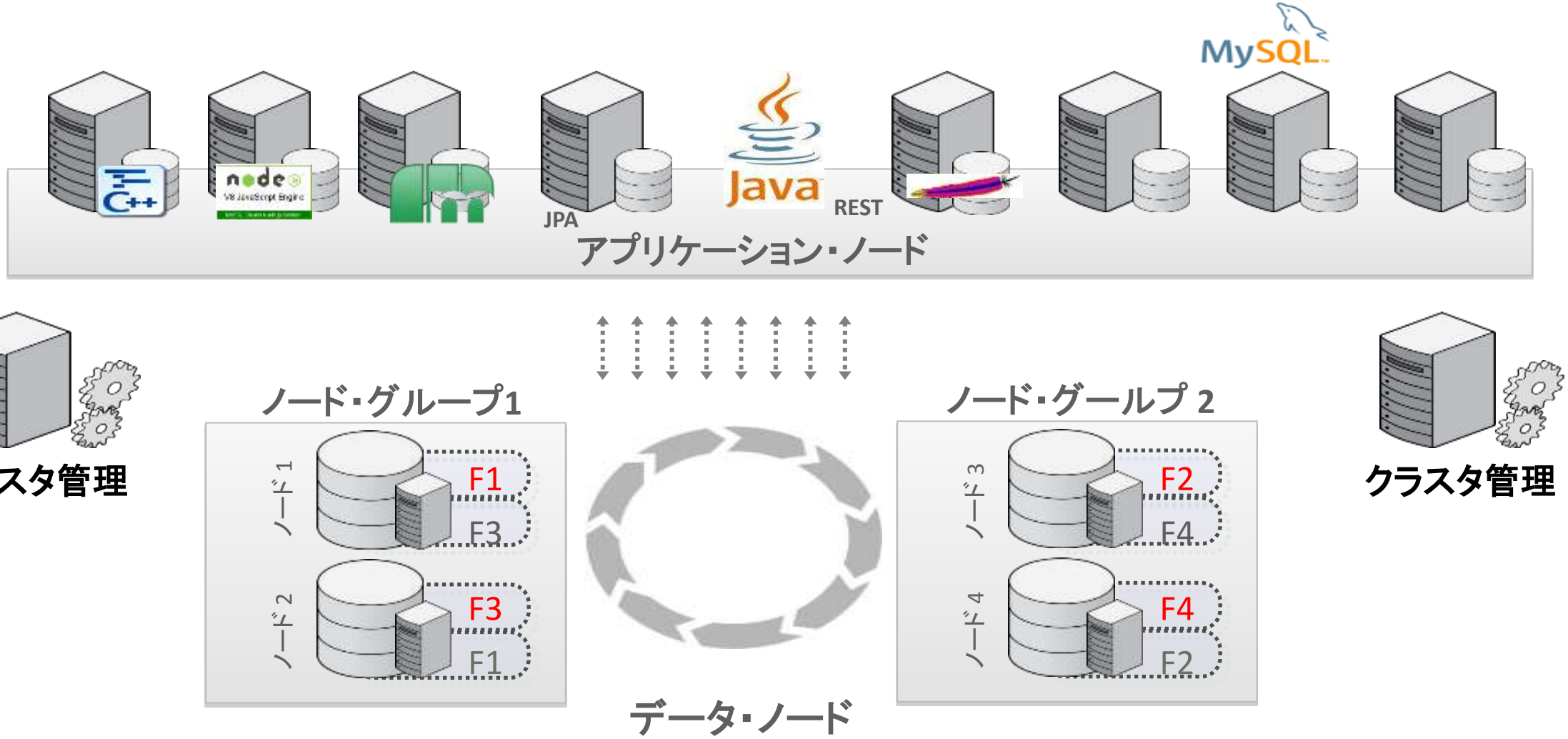
http://www.oracle.com/technetwork/database/database-technologies/clusterware/overview/index.html
http://www.oracle.com/technetwork/database/database-technologies/clusterware/downloads/ogiba-2189738.pdf

# 補足：MySQL Clusterによる高可用性構成

- メリット
  - 単一障害点が無く、フェイルオーバー時間も極めて短いため、可用性が非常に高い
  - アクティブ-アクティブ構成のため、スタンバイ機を用意する場合に比べH/Wリソースを有効活用できる
  - 参照処理だけでなく、更新処理についても負荷分散できる。
  - SQLだけでなく、豊富なNoSQLインターフェースも持っている(両方の利点を活かせる)(※)
- デメリット
  - 通常のMySQLサーバーとはアーキテクチャが異なる(データの持ち方が異なる)ため、アプリケーションの処理内容によって、向き/不向きがある
    - 主キーやユニークインデックスベースの処理が得意
    - スキャン系の処理は、主キー/ユニークインデックスベースの処理に比べるとオーバーヘッドが大きい
  - 運用方法が、通常のMySQLサーバーと異なる(MySQL Clusterの知識が別途必要)
  - 必要なサーバー台数が多くなる(最小構成で3台から)

※MySQL Clusterは、元々はNoSQLのデータストアでした。

# MySQL Clusterのアーキテクチャ概要



# 1,000億ドル以上の取引を守るMySQL Cluster

PayPal™

**アプリケーション**

世界最大級のオンライン決済サービス。Paypalの口座間やクレジットカードでの送金や入金が可能。アクティブアカウント1億以上、20以上の通貨に対応し、203の国と地域で利用可能。年率30%の成長。

**MySQL導入の効果**

MySQL ClusterをAWSの5拠点に導入し、全世界で1/3秒未満のレイテンシを実現。リアルタイムでの不正検知が可能に。

**MySQL導入の理由**

“NoSQLの特徴である迅速な開発とSQLモデルの信頼性の両方のメリットを実装してるため”
Daniel Austin, Chief Architect, PayPal

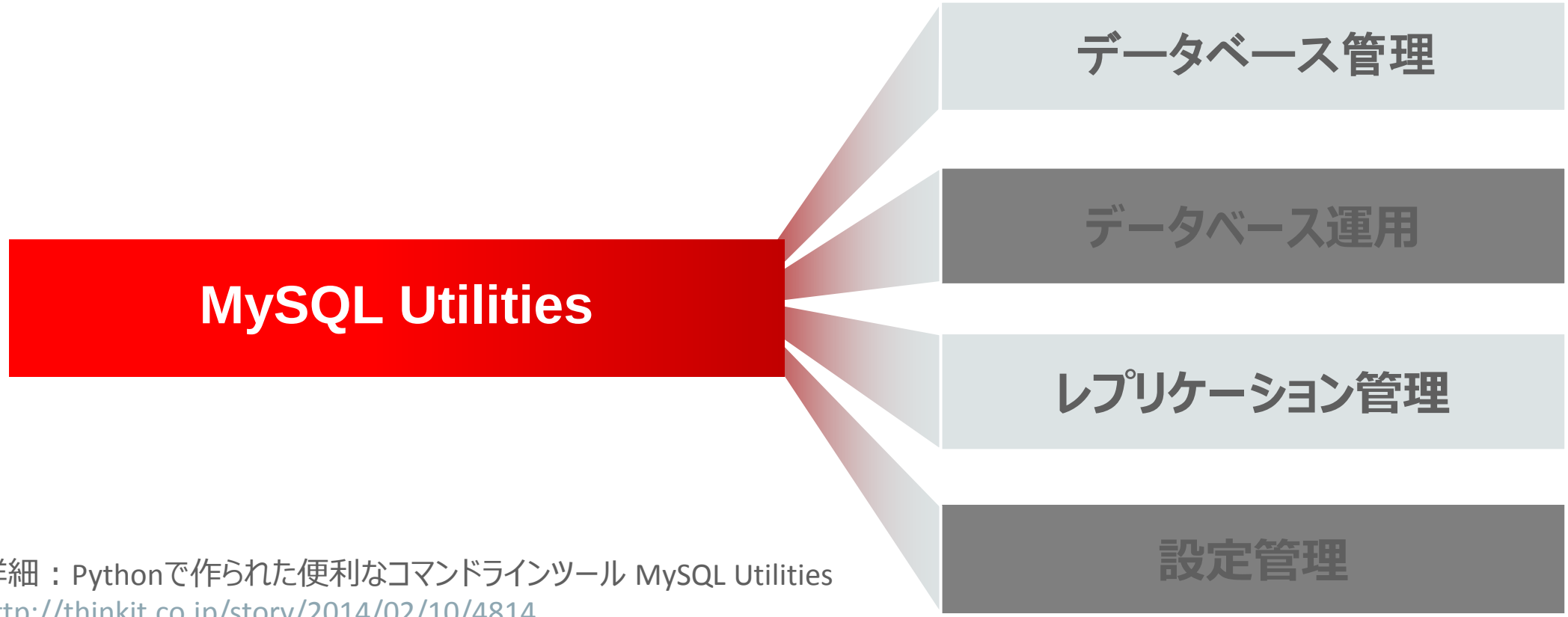
ORACLE

# Who's Using MySQL Cluster?

# Appendix：MySQL Utilities

MySQL Utilities
Provide Support by
MySQL Enterprise Edition

運用管理に関するPythonスクリプト




詳細：Pythonで作られた便利なコマンドラインツール MySQL Utilities
http://thinkit.co.jp/story/2014/02/10/4814

ORACLE

| Utility Category | Command | 概要 |
|---|---|---|
| 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 |
| データベース運用 | mysqluc | コマンドライン環境 |
| | mysqldiskusage | デーブルおよびデータファイルのサイズを表示 |
| | mysqlfrm | .frmファイルを読取り、CREATE TABLE文を作成 |
| | 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コピー |
| 設定管理 | 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 |

| Utility Category | Command | 概要 |
|---|---|---|
| レプリケーション管理 | mysqlfailover | レプリケーションの自動フェールオーバー (GTID) |
| | mysqlreplicate | レプリケーションを設定 |
| | mysqlrplms | ラウンドロビンによるマルチソースレプリケーション (GTID) |
| | mysqlrpladmin | レプリケーションの各種管理 (GTID) |
| | 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 |
| | mysqlrplsync | マスタ-スレーブ間、スレーブ-スレーブ間でのデータ同期状況を確認 (GTID) |

| | | |
|---|---|---|
| **Specialized Operations** | mysqlauditadmin | 監査ログのメンテナンス |
| | mysqlauditgrep | 監査ログの検索 |

※監査ログ取得は、MySQL Enterprise Edition(商用版)のみの機能です。
MySQL Enterprise Audit
http://www-jp.mysql.com/products/enterprise/audit.html

# Show slave attached to master
## mysqlrplshow

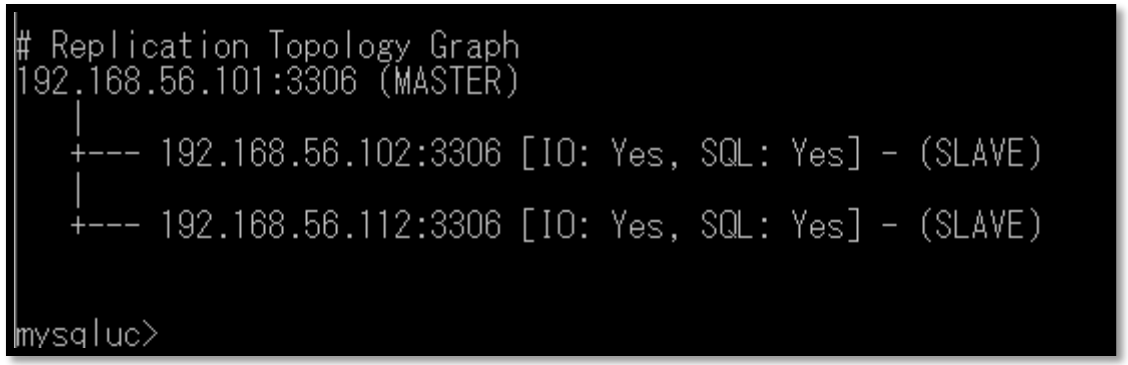
MySQL Utilities

- マスターに接続されているスレーブを表示してレプリケーショントポロジー確認可能

```
mysqluc> mysqlrplshow  --master=admin:pass@'192.168.56.101' --discover-slaves-login=admin:pass --verbose
WARNING: Using a password on the command line interface can be insecure.
# master on 192.168.56.101: ... connected.
# Finding slaves for master: 192.168.56.101:3306
WARNING: IP lookup by address failed for 192.168.56.101,reason: host not found

# Replication Topology Graph
192.168.56.101:3306 (MASTER)
   |
   +--- 192.168.56.102:3306 [IO: Yes, SQL: Yes] - (SLAVE)
   |
   +--- 192.168.56.112:3306 [IO: Yes, SQL: Yes] - (SLAVE)

mysqluc>
```




ORACLE

# Replication Synchronization Checker

## mysqlrplsync

MySQL Server versions 5.6.14 ～

MySQL Utilities

- レプリケーション構成で稼働中のサーバー間でデータの一貫性を確認可能
  - 不足しているデータベース、テーブル,テーブル毎の差分データ
  - サーバー間でデータの違いを見つける同期アルゴリズム
- 特定のサーバー群を比較する事が可能 **(--slaves=)**
- 特定のデータのチェックを制限する事が可能 **(--exclude)**
- マスターに接続されているスレーブを見つける事が可能 **(--discover-slaves-login)**
- パフォーマンスへの影響を制御することができます **(--checksum-timeout=)**

```
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","SHOW CREATE TABLE `sakila`.`city`"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","LOCK TABLES `sakila`.`city` READ"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","COMMIT"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","SELECT @@GLOBAL.GTID_EXECUTED"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","SHOW VARIABLES LIKE 'server_uuid'"
"2015-01-08 14:24:12","[admin] @  [192.168.56.1]",184,1,"Connect","admin@192.168.56.1 on "
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",184,1,"Query","SET NAMES 'utf8' COLLATE 'utf8_general_ci'"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",184,1,"Query","SET @@session.autocommit = OFF"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","CHECKSUM TABLE `sakila`.`city`"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",184,1,"Quit",""
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","UNLOCK TABLES"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","COMMIT"
"2015-01-08 14:24:12","admin[admin] @  [192.168.56.1]",168,1,"Query","SELECT SCHEMA_NAME FROM INFORMATION_SCHEMA.SCHEMATA\
          WHERE SCHEMA_NAME = 'sakila'"
```

ORACLE

# Replication Synchronization Checker

## mysqlrplsync

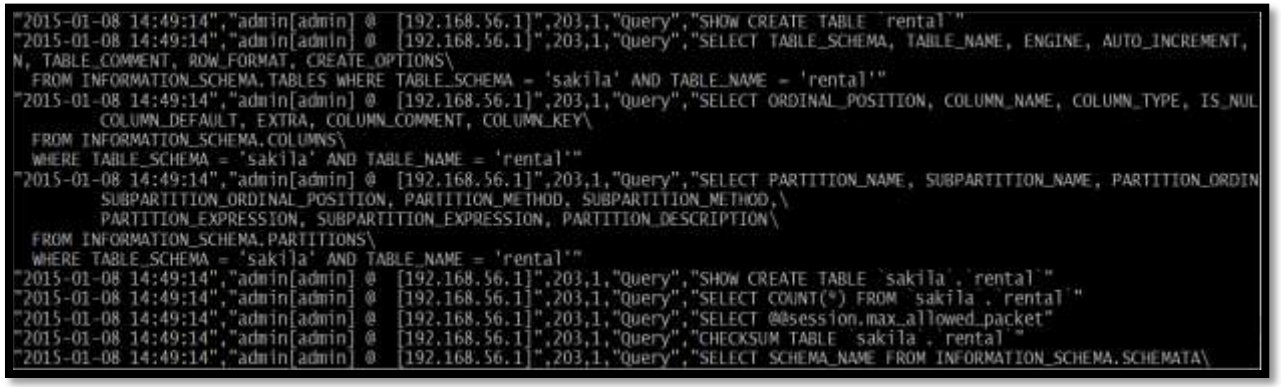

```
shell> mysqlrplsync --master=user:pass@localhost:3310 ¥
       --slaves=rpl:pass@localhost:3311,rpl:pass@localhost:3312

# GTID differences between Master and Slaves:
# - Slave 'localhost@3311' is 15 transactions behind Master.
# - Slave 'localhost@3312' is 12 transactions behind Master.
#
# Checking data consistency.
#
# Using Master 'localhost@3310' as base server for comparison.
# Checking 'test_rplsync_db' database...
# - Checking 't0' table data...
#   [OK] `test_rplsync_db`.`t0` checksum for server 'localhost@3311'.
#   [OK] `test_rplsync_db`.`t0` checksum for server 'localhost@3312'.
# - Checking 't1' table data...
#   [OK] `test_rplsync_db`.`t1` checksum for server 'localhost@3311'.
#   [OK] `test_rplsync_db`.`t1` checksum for server 'localhost@3312'.
# Checking 'test_db' database...
# - Checking 't0' table data...
#   [OK] `test_db`.`t0` checksum for server 'localhost@3311'.
#   [OK] `test_db`.`t0` checksum for server 'localhost@3312'.
# - Checking 't1' table data...
#   [OK] `test_db`.`t1` checksum for server 'localhost@3311'.
#   [OK] `test_db`.`t1` checksum for server 'localhost@3312'.
#
#...done.
# SUMMARY: No data consistency issue found.
```

# Compare Databases
## mysqldbcompare

MySQL Utilities

- データベース間における差を検知
  - データベースから欠落しているオブジェクトを発見
  - 定義が異なるオブジェクト
  - テーブル間のデータの違い
- 結果からオブジェクトとデータを同期する事が可能　(--changes-for=server1 –a)
- ユースケース
  - マスターとスレーブの一貫性の確認
  - 本番環境、ステージング環境、開発環境の一貫性の確認
  - 新しくデータと古いデータの差分レポート作成
  - バックアップとの差異を比較




ORACLE

# Compare Databases
## mysqldbcompare


```
$ mysqldbcompare --server1=root@localhost --server2=root@backup_host:3310 ¥
> inventory1:inventory2 --run-all-tests
# server1 on localhost: ... connected.
# server2 on backup_host: ... connected.
# Checking databases inventory1 on server1 and inventory2 on server2
WARNING: Objects in server1:inventory but not in server2:inventory:
      VIEW: finishing_up
      VIEW: cleaning
                                Defn    Row     Data
Type     Object Name                    Diff    Count   Check
------------------------------------------------------------------------
TABLE    supplier                       pass    FAIL    FAIL
Row counts are not the same among inventory1.supplier and inventory2.supplier.
Data differences found among rows:
--- inventory1.supplier
+++ inventory1.supplier
@@ -1,2 +1,2 @@
 code,name
-2,Never Enough Inc.
+2,Wesayso Corporation
```

# Hardware and Software
## Engineered to Work Together


ORACLE®