MySQL学习笔记（Day043：replication_4-GTID）

MySQL学习

一. Flashback

之前我们如果要撤销某个操作，可能需要使用一个 `全备 + 增量备份 + binlog` 的方式（前滚的方式）。
但是很可能我们 `仅仅需要撤销的是几分钟前的操作` ，采用前滚的方式恢复是比较耗时间的。
处理这种问题我们可以使用 `Flashback（闪回）` 技术。

MySQL官方版本中并 `不提供Flashback` （闪回）的功能，所以这里我们使用姜老师的 `innosql` 版本中的 `mysqlbinlog -B` 的功能，来实现Flashback（ `与原版mysqlbinlog兼容` ）。

```
[root@MyServer ~]> ./mysqlbinlog_innosql.5.6  -V  # 目前（20160405）暂时支持到MySQL5.6，后续演示也使用MySQL5.6
./mysqlbinlog_innosql.5.6 Ver 3.4 for Linux at x86_64

[root@MyServer ~]> ./mysqlbinlog_innosql.5.6  --help | grep flashback
  -B, --flashback     Flashback data to start_postition or start_datetime. # -B 参数，可以做到flashback
flashback                         FALSE
```

1.1. 演示闪回功能

1. 准备数据

```
mysql> select version();
+------------+
| version()  |
+------------+
| 5.6.27-log |
+------------+
1 row in set (0.00 sec)

mysql> show master status\G
*************************** 1. row ***************************
             File: bin.000014  -- 当前的binlog是 bin.000014
         Position: 191
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: 714b5c24-98f7-11e5-b2fe-5254a03976fb:1-20
1 row in set (0.00 sec)

mysql> use burn_test;
Database changed

mysql> select * from test_purge;
+------+
| a    |
+------+
|   10 |
|   30 |
|   40 |
+------+
3 rows in set (0.00 sec)

mysql> insert into test_purge values(1024);
Query OK, 1 row affected (0.03 sec)

mysql> select * from test_purge;
+------+
| a    |
+------+
|   10 |
|   30 |
|   40 |
| 1024 |
+------+
4 rows in set (0.00 sec)
```

通过 `mysqlbinlog` 解析，可以有如下结果

```
[root@MyServer 5.6]> /root/mysqlbinlog_innosql.5.6  bin.000014 -vv
## ------------------省略其他输出-----------------
BEGIN
/*!*/;
# at 316
#160405 22:04:55 server id 5627  end_log_pos 374 CRC32 0x0e2a9bff       Table_map: `burn_test`.`test_purge` mapped to number 73
# at 374
#160405 22:04:55 server id 5627  end_log_pos 414 CRC32 0x9076fc41       Write_rows: table id 73 flags: STMT_END_F
BINLOG '
B8YDVxP7FQAAOgAAAHYBAAAAAEkAAAAAAAEACWJ1cm5fdGVzdAAKdGVzdF9wdXJnZQABAwAA/5sq
Dg==
B8YDVx77FQAAKAAAAJ4BAAAAAEkAAAAAAAEAAgAB//4ABAAAQfx2kA==
'/*!*/;
### INSERT INTO `burn_test`.`test_purge`     ## 这个是insert的注释
### SET
###   @1=1024 /* INT meta=0 nullable=0 is_null=0 */
# at 414
#160405 22:04:55 server id 5627  end_log_pos 445 CRC32 0x7be28ced       Xid = 18
COMMIT/*!*/;
## ------------------省略其他输出-----------------
```

通过 Flashback 功能就可以把 `insert改成delete` （ *同理 update 仅需要交换前后项即可* ）

```
[root@MyServer 5.6]> /root/mysqlbinlog_innosql.5.6  bin.000014 -B -vv  # 使用-B参数
## ------------------省略其他输出-----------------
BINLOG '
B8YDVxP7FQAAOgAAAHYBAAAAAEkAAAAAAAEACWJ1cm5fdGVzdAAKdGVzdF9wdXJnZQABAwAA/5sq
Dg==
B8YDVyD7FQAAKAAAAJ4BAAAAAEkAAAAAAAEAAgAB//4ABAAAQfx2kA==
'/*!*/;
### DELETE FROM `burn_test`.`test_purge`    ## 从原来的insert变成了delete
### WHERE
###   @1=1024 /* INT meta=0 nullable=0 is_null=0 */
DELIMITER ;
## ------------------省略其他输出-----------------
```

将Flashback生成的binlog应用到MySQL

```
[root@MyServer 5.6]> /root/mysqlbinlog_innosql.5.6 -vv  -B bin.000014   --start-position=316 | mysql -uroot -p -S /tmp/mysql.sock_56  # 使用管道的方式
Enter password:
```

检查 `test_purge` 表中的内容，发现之前插入的记录 `a=1024` 已经被删除了（ `通过将insert改成delete，逻辑闪回` ）

```
mysql> select * from test_purge;
+----+
| a  |
+----+
| 10 |
| 30 |
| 40 |
+----+
3 rows in set (0.00 sec)
```

> Oracle的 `闪回` 是使用的 `UNDO` 信息（ `物理的` ），而我们这里使用的是 `binlog的逆操作` （ `逻辑的` ）；
> `mysqlbinlog -B` 只能对 `DML` 语句进行闪回；而Oracle是可以闪回 `DDL` 操作的（ *MySQL如果要支持闪回DDL，需要修改源码* ）
>
> *innosql中的闪回DDL操作是通过 `Recycle Bin Tablespace` 的方式实现的（可以简单理解为回收站）*

1.2. 关于Master Flashback的位置

假设现在 Master 宕机了，然后切到了Slave，此时 Master 恢复后，需要将部分数据Flashback掉（ *可能是宕机前最后一部分没有传过去的binlog* ），那 Flashback 掉的位置就很重要了，这个位置一般 `以Slave上SQL线程最终回放完的位置为准`

```
mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.14.70
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000022
          Read_Master_Log_Pos: 194
               Relay_Log_File: relay.000005
                Relay_Log_Pos: 355
        Relay_Master_Log_File: bin.000022  -- 回放到的对应的文件
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 194         --  回放到的文件对应的位置
              Relay_Log_Space: 592
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           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: 100
                  Master_UUID: c241b625-e932-11e5-bb11-5254f035dabc
             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 more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set:
            Executed_Gtid_Set: c241b625-e932-11e5-bb11-5254f035dabc:1-11559
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)
```

在等待Slave 回放完 以后，通过 Relay_Master_Log_File 和 Exec_Master_Log_Pos 的值，就 可以知道Master上需要Flashback掉的位置

二. slave_rows_search_algorithms

> 官方文档

```
mysql> show variables like "%algorithms%";
+-------------------------------+-----------------------+
| Variable_name                 | Value                 |
+-------------------------------+-----------------------+
| slave_rows_search_algorithms | TABLE_SCAN,INDEX_SCAN |
+-------------------------------+-----------------------+
1 row in set (0.00 sec)
```

上面这个参数默认 不要去更改 。这个参数表明了怎样来 提高复制的性能 。

之前强调过的 每张表一定要有主键 的要求，除了是符合范式的要求，也可以 提高主从复制的性能 。
因为Slave进行回放的时候，是 根据索引（包括主键）进行回放 的，具体信息如下：

Index used / option value	INDEX_SCAN, HASH_SCAN or INDEX_SCAN, TABLE_SCAN, HASH_SCAN	INDEX_SCAN, TABLE_SCAN	TABLE_SCAN, HASH_SCAN
Primary key or unique key	Index scan	index scan	Index hash
(Other) Key	Index hash	Index scan	Index hash
No index	Table hash	Table scan	Table hash

Slave回放时有如下过程：

1. 先找主键
2. 没有主键则找唯一索引
3. 没有唯一索引则找普通索引
4. 没有普通索引则 全表扫描

> delete from table_a where a in (1, 3);
> 针对上述语句，假设 table_a不存在任何主键或索引 ，则在 Master 上操作的时候只要 扫描一遍 该表即可，但是 复制是基于行的（ROW格式） ，在 Slave 上就要 扫描两次 ，一次扫描 =1 的，一次扫描 =3 的。
>
> **全表扫描时：Master 扫描一次，复制时，Slave上扫 N 次**

新增的 Hash Scan 方式，会先增加一个哈希表，这样就只扫描一次了，但是创建哈希表的代价很大。所以 默认没有启用 的。

所以强烈建议：每张表上都要有一个主键

三. 半同步复制的状态

> 官方文档 – 服务器状态

```
--
-- master端
--
mysql> show global status like "%rpl%";
+--------------------------------------------+-------+
| Variable_name                              | Value |
+--------------------------------------------+-------+
| Rpl_semi_sync_master_clients               | 1     | -- 半同步复制的client数量
| Rpl_semi_sync_master_net_avg_wait_time     | 0     | -- master平均等待slave的时间，mysql-5.7.8 后被废弃
| Rpl_semi_sync_master_net_wait_time         | 0     | -- master总的等待slave的时间，mysql-5.7.8 后被废弃
| Rpl_semi_sync_master_net_waits             | 0     | -- master总的等待slave的次数
| Rpl_semi_sync_master_no_times              | 0     | -- 切成异步的次数（no = number of）
| Rpl_semi_sync_master_no_tx                 | 0     | -- 切成异步后提交的事物数
| Rpl_semi_sync_master_status                | ON    | -- 半同步复制的状态
| Rpl_semi_sync_master_timefunc_failures     | 0     | -- master调用gettimeofday()函数失败的此时
| Rpl_semi_sync_master_tx_avg_wait_time      | 0     | -- master等待事物的平均时间
| Rpl_semi_sync_master_tx_wait_time          | 0     | -- master等待事物的总的时间
| Rpl_semi_sync_master_tx_waits              | 0     | -- master等待事物的次数
| Rpl_semi_sync_master_wait_pos_backtraverse | 0     |
| Rpl_semi_sync_master_wait_sessions         | 0     |
| Rpl_semi_sync_master_yes_tx                | 0     |
| Rpl_semi_sync_slave_status                 | OFF   |
+--------------------------------------------+-------+
15 rows in set (0.01 sec)
```

模拟停止IO线程，然后查看状态

```
--
-- slave 端
--
mysql> stop slave io_thread;  -- 只停掉IO线程
Query OK, 0 rows affected (0.01 sec)
```

```
--
-- master 端
--
mysql> show global status like "%rpl%";
+-----------------------------------------+-------+
| Variable_name                           | Value |
+-----------------------------------------+-------+
| Rpl_semi_sync_master_clients            | 1     |
| Rpl_semi_sync_master_net_avg_wait_time  | 0     |
| Rpl_semi_sync_master_net_wait_time      | 0     |
| Rpl_semi_sync_master_net_waits          | 0     |
| Rpl_semi_sync_master_no_times           | 0     |
| Rpl_semi_sync_master_no_tx              | 0     |
| Rpl_semi_sync_master_status             | ON    | -- 还是ON
| Rpl_semi_sync_master_timefunc_failures  | 0     |
| Rpl_semi_sync_master_tx_avg_wait_time   | 0     |
| Rpl_semi_sync_master_tx_wait_time       | 0     |
| Rpl_semi_sync_master_tx_waits           | 0     |
| Rpl_semi_sync_master_wait_pos_backtraverse | 0     |
| Rpl_semi_sync_master_wait_sessions      | 0     |
| Rpl_semi_sync_master_yes_tx             | 0     |
| Rpl_semi_sync_slave_status              | OFF   |
+-----------------------------------------+-------+
15 rows in set (0.00 sec)

mysql> insert into test_rpl(b,c,d) values(16,17,18);
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (5.03 sec)  -- 等待5秒后，切成异步

mysql> show global status like "%rpl%";
+-----------------------------------------+-------+
| Variable_name                           | Value |
+-----------------------------------------+-------+
| Rpl_semi_sync_master_clients            | 0     | -- client数变为0
| Rpl_semi_sync_master_net_avg_wait_time  | 0     |
| Rpl_semi_sync_master_net_wait_time      | 0     |
| Rpl_semi_sync_master_net_waits          | 0     |
| Rpl_semi_sync_master_no_times           | 1     | -- 切成异步的次数
| Rpl_semi_sync_master_no_tx              | 1     | -- 切成异步后的事物数
| Rpl_semi_sync_master_status             | OFF   |
| Rpl_semi_sync_master_timefunc_failures  | 0     |
| Rpl_semi_sync_master_tx_avg_wait_time   | 0     |
| Rpl_semi_sync_master_tx_wait_time       | 0     |
| Rpl_semi_sync_master_tx_waits           | 0     |
| Rpl_semi_sync_master_wait_pos_backtraverse | 0     |
| Rpl_semi_sync_master_wait_sessions      | 0     |
| Rpl_semi_sync_master_yes_tx             | 0     |
| Rpl_semi_sync_slave_status              | OFF   |
+-----------------------------------------+-------+
15 rows in set (0.00 sec)
```

四. mysqlrplshow

在Master端执行 `show slave hosts` 可以看到连接到Master的Slave节点。

```
--
-- Master 端
--
mysql> show slave hosts;
+-----------+------+------+-----------+--------------------------------------+
| Server_id | Host | Port | Master_id | Slave_UUID                           |
+-----------+------+------+-----------+--------------------------------------+
|       101 |      | 3306 |       100 | e313c232-e932-11e5-b79e-5254f03466c1 |
+-----------+------+------+-----------+--------------------------------------+
1 row in set (0.00 sec)
```

我们发现上面的 `Host` 一栏是空的，同时，如果我们执行 `mysqlrplshow` 命令去查看复制关系，也是会报错的

```
#
# Slave 端
#
[root@Slave1 ~]> mysqlrplshow --master=root:123@172.18.14.70:3306 --discover-slaves-login=root:123 --verbose
WARNING: Using a password on the command line interface can be insecure.
# master on 172.18.14.70: ... connected.
# Finding slaves for master: 172.18.14.70:3306
WARNING: There are slaves that have not been registered with --report-host or --report-port:
 - unknown host:3306

# Replication Topology Graph
No slaves found.
```

需要在 `Slave` 的 `/etc/my.cnf` 中增加 `report-host` 配置，然后重启MySQL实例

```
#
# Slave端
#
[mysqld]
# Slave和Master通信的IP
report-host=172.18.14.71
```

```
[root@Slave1 ~]> service mysqld restart
Shutting down MySQL... SUCCESS!
Starting MySQL. SUCCESS!


--
-- Master 端
--
mysql> show slave hosts;
+-----------+--------------+------+-----------+--------------------------------------+
| Server_id | Host         | Port | Master_id | Slave_UUID                           |
+-----------+--------------+------+-----------+--------------------------------------+
|       101 | 172.18.14.71 | 3306 |       100 | e313c232-e932-11e5-b79e-5254f03466c1 |
+-----------+--------------+------+-----------+--------------------------------------+
1 row in set (0.00 sec)


#
# 任意一个可以访问Master端的服务器
#
[root@Slave1 ~]> mysqlrplshow --master=root:123@172.18.14.70:3306 --discover-slaves-login=root:123 --verbose
WARNING: Using a password on the command line interface can be insecure.
# master on 172.18.14.70: ... connected.
# Finding slaves for master: 172.18.14.70:3306
WARNING: Unable to find aliases for hostname 'Slave1' reason: [Errno -2] Name or service not known
WARNING: IP lookup by address failed for 172.18.14.70,reason: Unknown host
# 上面的两个Warning只是没法做name解析，可以忽略

# Replication Topology Graph
172.18.14.70:3306 (MASTER)
   |
   +--- 172.18.14.71:3306 [IO: Yes, SQL: Yes] - (SLAVE)  # 当前服务器状态OK
```

五. GTID

5.1. GTID的介绍

1. `G` lobal `T` ransaction `Id` entifier – 全局事物ID
2. `GTID = Server_UUID + Transaction_ID`
    - `Server_UUID` 是全局唯一的
    - `Transaction_ID` 是自增的
3. `GTID` 的作用是替代 `Filename + Position`

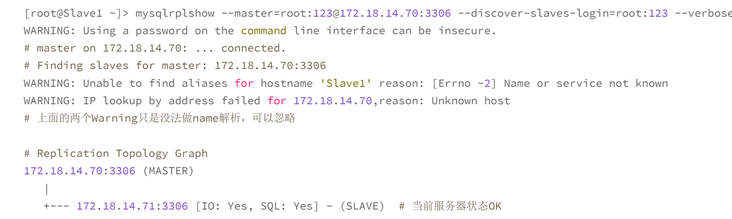
```
mysql> show variables like "server_uuid";
+---------------+--------------------------------------+
| Variable_name | Value                                |
+---------------+--------------------------------------+
| server_uuid   | c241b625-e932-11e5-bb11-5254f035dabc |
+---------------+--------------------------------------+
1 row in set (0.00 sec)
```

在MySQL中看到的 `UUID` ，实际是保存在 `$DATADIR/auto.cnf` 中的，且该文件是服务器初始化的时候自动生成的。

```
[root@Master ~]> cat /data/mysql_data/5.7.11/auto.cnf
[auto]
server-uuid=c241b625-e932-11e5-bb11-5254f035dabc
```

> **通过 冷备 做备份，拷贝 `$DATADIR` 时，记得要把备份中的 `auto.cnf` 给删除**

5.2. GTID的意义



- 假设现在 没有GTID
    - 当 `Master宕机` 后，一个 `Slave` 被 `提升为New Master` ，如果需要继续维持复制关系，就需要把另外两个Slave的 `CHANGE MASTER` 指向 `New Master` ；
    - 那问题来了，原来Slave是指向 `Master` 的 `Filename_M + Position_M` 的位置，现在要指向 `New Master` 上新的 `Filename_N + Position_N` 的位置，这 `两个位置是比较难对应起来的` ；
    - 此时两个Slave要继续复制（*CHANGE MASTER*）会比较麻烦。
- 使用GTID
    - 和上面一样的场景，两个Slave需要重新指向 `New Master` ，由于 `使用了GTID` ，目前 `Slave-A` 获取到的日志对应的 `GTID=G_A` ， `Slave-B` 获取到的日志对应的 `GTID=G_B` ；
    - 此时 `New Master` 上是存在 `G_A 和 G_B` （*`通过选举出来的，获取的日志应该是最多的`*），那两个Slave就可以直接使用 `G_A和G_B` 这两个GTID，通过指向 `New Master` 接着继续复制；

5.3. GTID的配置

```
[mysqld]
log_bin = bin.log
gtid_mode = ON
log_slave_updates = 1
enforce_gtid_consistency = 1
```

1. MySQL 5.6 必须开启参数 `log_slave_updates` （ *5.6版本的限制* ）
    - `log_slave_updates` 详见 级联复制
2. MySQL 5.6 升级到gtid模式需要停机重启
3. MySQL 5.7 版本开始可以不开启 `log_slave_updates`
4. MySQL 5.7.6 版本开始可以在线升级成gtid模式

5.4. 基于GTID的复制

```
      GTID rpl    +--------+    GTID rpl
          +-------+ Master +-------+
          |       +--------+       |
          |                        |
     +----V----+              +----V----+
     | Slave 1 |              | Slave 2 |
     +---------+              +---------+
```

`Master` 备份数据库的步骤略过，这里使用 `mydumper` 备份，且 `Slave1` 和 `Slave2` 已经通过 `myloader` 进行恢复数据。

> 在开启GTID后，使用 `mysqldump` 备份单个数据库时，会有 `Warning` ，大致意思为你只备份了部分数据库，但是启用GTID后包含了所有的事物。
> 可以忽略该警告。

当然我们还是看一下备份后的 `metadata`

```
[root@Slave2 backup]> cat metadata
Started dump at: 2016-04-07 17:11:56
SHOW MASTER STATUS:
        Log: bin.000020
        Pos: 194
        GTID:c241b625-e932-11e5-bb11-5254f035dabc:1-11558
# 这里除了我们之前用过的 Filename 和 POS ，还有GTID

Finished dump at: 2016-04-07 17:11:57
```

注意：这个 `metadata` 中的 `GTID:c241b625-e932-11e5-bb11-5254f035dabc:1-11558` 的意思 等同于 通过 `mysqldump` 备份得到的文件中的 `SET @@GLOBAL.GTID_PURGED='c241b625-e932-11e5-bb11-5254f035dabc:1-11558';` ，均表示为：这部分的GTIDs对应的 事物 已经在这个备份中了，Slave在还原备份后，进行复制时， 要跳过这些GTIDs（对应的事物）。

5.4.1. 跳过GTIDs

如果备份使用 `mysqldump` 进行备份， 请跳过该步骤 ，因为在备份中已经存在下面这个语句：

```
--
-- GTID state at the beginning of the backup
--

SET @@GLOBAL.GTID_PURGED='c241b625-e932-11e5-bb11-5254f035dabc:1-11558';
```

因为我们使用 `mydumper` 进行备份和还原， `myloader` 并不会帮我们执行上述语句，所以我们要手工执行该语句，让 `Slave` 知道这些GTIDs需要跳过：

```
--
-- Slave 1 端
--
mysql> SET @@GLOBAL.GTID_PURGED='c241b625-e932-11e5-bb11-5254f035dabc:1-11558';
ERROR 1840 (HY000): @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.
```

这里报出一个错误，意思为如果要设置 `@@GLOBAL.GTID_PURGED` ，则 `@@GLOBAL.GTID_EXECUTED` 必须为空。

结合参考链接 和 官方文档（ 感谢 **M062-北京-艾度纽斯-郭释** 给出的链接 ），我们需要使用 `reset master;` 命令，将 `@@GLOBAL.GTID_EXECUTED` 清空。

```
--
-- Slave 1 端
--
mysql> reset master;
Query OK, 0 rows affected (0.07 sec)

mysql> SET @@GLOBAL.GTID_PURGED='c241b625-e932-11e5-bb11-5254f035dabc:1-11558';
Query OK, 0 rows affected (0.02 sec)
```

> 至此，我们已经跳过了备份中存在的GTIDs所对应的事物

5.4.2. CHANGE MASTER

```
--
-- Slave 1 端
--
mysql> change master to master_host="172.18.14.70", master_port=3306, master_user='rpl', master_password='123', master_auto_position=1;
Query OK, 0 rows affected, 2 warnings (0.12 sec)
-- 再也不用输入 log_file 和 log_pos 了


mysql> start slave;
Query OK, 0 rows affected (0.06 sec)

mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.14.70
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000021
          Read_Master_Log_Pos: 194
               Relay_Log_File: relay.000002
                Relay_Log_Pos: 355
        Relay_Master_Log_File: bin.000021
             Slave_IO_Running: Yes  -- IO 线程运行正常
            Slave_SQL_Running: Yes  -- SQL 线程运行正常
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 194
              Relay_Log_Space: 552
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           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: 100
                  Master_UUID: c241b625-e932-11e5-bb11-5254f035dabc
             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 more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set:
            Executed_Gtid_Set: c241b625-e932-11e5-bb11-5254f035dabc:1-11558  -- 自动认为这些已经执行过了
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)
```

> `Slave 2` 的操作过程与上述一样，这里省略

5.4.3. 复制完成

```
--
-- Master 端
--
mysql> show slave hosts;
+-----------+-------------+------+-----------+--------------------------------------+
| Server_id | Host        | Port | Master_id | Slave_UUID                           |
+-----------+-------------+------+-----------+--------------------------------------+
|       101 | 172.18.14.71 | 3306 |       100 | 4144fe65-fcfc-11e5-9c82-5254f03466c1 |
|       102 | 172.18.14.72 | 3306 |       100 | 4729a5ec-fcfc-11e5-8c97-5254f0d46611 |
+-----------+-------------+------+-----------+--------------------------------------+
2 rows in set (0.00 sec)


[root@Slave1 mysql]> mysqlrplshow --master=root:123@172.18.14.70:3306 --discover-slaves-login=root:123 --verbose
WARNING: Using a password on the command line interface can be insecure.
# master on 172.18.14.70: ... connected.
# Finding slaves for master: 172.18.14.70:3306
WARNING: Unable to find aliases for hostname 'Slave1' reason: [Errno -2] Name or service not known
WARNING: IP lookup by address failed for 172.18.14.70,reason: Unknown host

# Replication Topology Graph
172.18.14.70:3306 (MASTER)
   |
   +--- 172.18.14.71:3306 [IO: Yes, SQL: Yes] - (SLAVE)
   |
   +--- 172.18.14.72:3306 [IO: Yes, SQL: Yes] - (SLAVE)
```

```
--
-- Master 端
--
mysql> insert into test_rpl(b,c,d) values(25,26,27);
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.02 sec)


--
-- Slave 1 端
--
mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave1     |
+------------+
1 row in set (0.00 sec)

mysql> select * from burn_test.test_rpl;
+---+------+------+------+
| a | b    | c    | d    |
+---+------+------+------+
| 1 |    1 |    2 |    3 |
| 2 |    4 |    5 |    6 |
| 3 |    7 |    8 |    9 |
| 4 |   10 |   11 |   12 |
| 5 |   13 |   14 |   15 |
| 6 |   16 |   17 |   18 |
| 7 |   19 |   20 |   21 |
| 8 |   22 |   23 |   24 |
| 9 |   25 |   26 |   27 |  -- 新插入的25，26，27
+---+------+------+------+
9 rows in set (0.00 sec)


--
-- Slave 2 端
--
mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
1 row in set (0.00 sec)

mysql> select * from burn_test.test_rpl;
+---+------+------+------+
| a | b    | c    | d    |
+---+------+------+------+
| 1 |    1 |    2 |    3 |
| 2 |    4 |    5 |    6 |
| 3 |    7 |    8 |    9 |
| 4 |   10 |   11 |   12 |
| 5 |   13 |   14 |   15 |
| 6 |   16 |   17 |   18 |
| 7 |   19 |   20 |   21 |
| 8 |   22 |   23 |   24 |
| 9 |   25 |   26 |   27 | -- 新插入的25，26，27
+---+------+------+------+
9 rows in set (0.00 sec)
```

> 至此，基于GTID的复制就搭建完成了

5.5. GTID 与 Filename-Pos的对应

在 binlog 中，多了一个 `GTID 的 event` ，如下所示

```
mysql> show binlog events in 'bin.000021';
+------------+-----+----------------+-----------+-------------+-------------------------------------------------------------------+
| Log_name   | Pos | Event_type     | Server_id | End_log_pos | Info                                                              |
+------------+-----+----------------+-----------+-------------+-------------------------------------------------------------------+
| bin.000021 |   4 | Format_desc    |       100 |         123 | Server ver: 5.7.11-log, Binlog ver: 4                             |
| bin.000021 | 123 | Previous_gtids |       100 |         194 | c241b625-e932-11e5-bb11-5254f035dabc:1-11558                      |
| bin.000021 | 194 | Gtid           |       100 |         259 | SET @@SESSION.GTID_NEXT= 'c241b625-e932-11e5-bb11-5254f035dabc:11559' |  -- 一个GTID 的 event
| bin.000021 | 259 | Query          |       100 |         336 | BEGIN                                                             |
| bin.000021 | 336 | Table_map      |       100 |         395 | table_id: 108 (burn_test.test_rpl)                                |
| bin.000021 | 395 | Write_rows     |       100 |         447 | table_id: 108 flags: STMT_END_F                                   |
| bin.000021 | 447 | Xid            |       100 |         478 | COMMIT /* xid=184 */                                              |
| bin.000021 | 478 | Stop           |       100 |         501 |                                                                   |
+------------+-----+----------------+-----------+-------------+-------------------------------------------------------------------+
8 rows in set (0.00 sec)
```

通过 扫描 binlog中的GTID值，就可以知道 GTID与Filename-Pos对应的关系 ，但是如果binlog非常大，扫描的量也是会很大的，所以在binlog开头部分有一个 `Previous_gtids` 的event，如下所示：

```
mysql> show binlog events in 'bin.000021'\G
*************************** 1. row ***************************
   Log_name: bin.000021
        Pos: 4
 Event_type: Format_desc
  Server_id: 100
End_log_pos: 123
       Info: Server ver: 5.7.11-log, Binlog ver: 4
*************************** 2. row ***************************
   Log_name: bin.000021
        Pos: 123
 Event_type: Previous_gtids  -- 表示在次之前，GTID运行到的范围是哪里
  Server_id: 100
End_log_pos: 194
       Info: c241b625-e932-11e5-bb11-5254f035dabc:1-11558
```

> 如果我要的GTID比 `Previous_gtids` 的大，就扫描当前文件，反之则扫描之前的文件，依次类推。
> 因为binlog在rotate（*rotate events*）的时候，是知道当前最大的GTID的，可以将该值写入到下一个新的binlog的开头，即 `Previous_gtids`

六. 级联复制

6.1. 级联复制的介绍

```
Master      Replication    Slave
+---------+ ------------->+---------+              +---------+
| MySQL A |               | MySQL B |              | MySQL C |
+---------+               +---------+--------------->---------+
                            Master     Replication     Slave
```

`MySQL B` 从 `MySQL A` 上复制， `MySQL C` 从 `MySQL B` 上复制，此时 `MySQL B` 上就要开启 `log_slave_updates`

如果 `MySQL B` 上 不启用`log_slave_updates` ，则 不会产生`binlog` ，没有 `binlog` 则无法提供复制；

> `log_bin` 参数是当有直接对数据库进行操作的时候，产生binlog， 对复制产生的操作不会产生`binlog`（*仅存relay-log*）

6.2. 级联复制的场景

1. 跨机房的复制

```
                                    +
          Area Shanghai             |    Area Bejing
      M                     S       |
+---------+   rpl     +---------+   |      +---------+
| MySQL A +-----------> MySQL B +----------> MySQL C |
+----+----+           +---------+   |rpl   +---------+
     |                     M        |           S
     |rpl                            |
     |                               |
+----v----+                          |
| MySQL D |                          |
+---------+                          |
     S                               |
                                    +
```

在跨机房搭建复制时，如果 `MySQL A` 挂了， `MySQL B` 提升为 `New Master` ，此时 `MySQL C` 是不需要去做 `CHANGE MASTER` 操作的。
缺点是复制的延迟会更大（*跨机房的延迟本来就很难避免*）。

2. 库的拆分

```
                                          MySQL C
                                        +-----------+
                                        | +-------+ |
   MySQL A             MySQL B       +------> | DB_A  | |
+-----------+       +-----------+     |      | +-------+ |
| +-------+ |       | +-------+ |     |      +-----------+
| | DB_A  | |       | | DB_A  +------+   replicate-do-db=DB_A
| +-------+ |  rpl  | +-------+ |
| | DB_B  | +-------> | DB_B  | |
| +-------+ |       | +-------+ |
| | DB_C  | |       | | DB_C  | |
| +-------+ |       | +-------+ |
+-----------+       +-----------+
     M                   S
```

当 `MySQL A` 压力很大的时候，需要把 `DB_A` 拆分出去，就可以使用 级联复制 ，让 `DB_A` 形成单独的库。

6.3. 级联复制测试

```
     M                   S
+--------+    rpl     +---------+          +---------+
| Master +------------> Slave 1 +----------> Slave 3 |
+----+---+            +---------+    rpl   +---------+
     |                     M                   S
     |rpl
     |
+----V----+
| Slave 2 |
+---------+
     S
```

`Master --> Slave1 --> Slave2` 就是级联复制， `Master到Slave1` 和 `Master到Slave2` 的部分之前做完了，现在做 `Slave1到Slave3` 的 级联复制 。

1. 在Master端新建一个DB名为 `test_dodb` ，用于在 `Slave3` 上测试参数 `replicate_do_db` 。

```
--
-- Master端
--
mysql> create database test_dodb;
Query OK, 1 row affected (0.02 sec)

mysql> commit;
Query OK, 0 rows affected (0.00 sec)

--
-- Slave1 端
--
mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave1     |
+------------+
1 row in set (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| burn_test          |
| mysql              |
| performance_schema |
| sys                |
| test_dodb          |  -- 复制过来了
+--------------------+
6 rows in set (0.00 sec)

--
-- Slave2 端
--
mysql> select @@hostname;
+------------+
| @@hostname |
+------------+
| Slave2     |
+------------+
1 row in set (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| burn_test          |
| mysql              |
| performance_schema |
| sys                |
| test_dodb          |  -- 复制过来了
+--------------------+
6 rows in set (0.00 sec
```

2. 在Slave3上增加如下参数

```
[mysqld]
report-host=172.18.14.73
# 只复制test_dodb的库
replicate_do_db=test_dodb
```

3. change master

```
--
-- Slave3 端
--
mysql> change master to master_host="172.18.14.71", master_port=3306, master_user='rpl', master_password='123', master_auto_position=1;  -- 请先确保Slave1上有'rpl'用户
Query OK, 0 rows affected, 2 warnings (0.12 sec)  -- 还是密码明文的warnings，忽略

mysql> start slave;
Query OK, 0 rows affected (0.06 sec)

mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State:
                  Master_Host: 172.18.14.71
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File:
          Read_Master_Log_Pos: 4
               Relay_Log_File: relay.000001
                Relay_Log_Pos: 4
        Relay_Master_Log_File:
             Slave_IO_Running: No   -- IO线程没有启动
            Slave_SQL_Running: Yes
              Replicate_Do_DB: test_dodb
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 0
              Relay_Log_Space: 154
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           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: 1236
                Last_IO_Error: Got fatal error 1236 from master when reading data from binary log: 'The slave is connecting using CHANGE MASTER TO MASTER_AUTO_POSITION = 1, but the master has purged binary logs containing GTIDs that the slave requires.'  -- 又报了binlog被purge的错
               Last_SQL_Errno: 0
               Last_SQL_Error:
  Replicate_Ignore_Server_Ids:
             Master_Server_Id: 101
                  Master_UUID: 4144fe65-fcfc-11e5-9c82-5254f03466c1
             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 more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp: 160409 22:09:08
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set:
            Executed_Gtid_Set:
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)
```

> 复制关系是这样的： `Master-->Slave1-->Slave3` ，且我们这里使用了基于GTID的复制；
> 也就是说，尽管 `Slave3` 指向了 `Slave1` ，但是传递给 `Slave3` 的GTID（ `uuid+xid` ）还是原来 `Master` 上产生 的，这些 `GTID` 的值 本身 不会 因为多了一个Slave1中转，而 发生任何改变 。

由于复制到 `Slave1` 时，采用了 备份恢复 (*myloader*)的方式，且Master上部分的binlog被purge了（ *人为删除* ），也就不难理解上面的报错了。
因为我们的 `test_dodb` 创建完成后，没有做任何删除binlog的动作，且我们的 `Slave3` 只需要同步 `test_dodb` ，所以我们这里可以放心的 跳过 被purge的GTID

```
--
-- Slave 1 端
--
mysql> show variables like "%gtid%";
+---------------------------------+-------------------------------------------+
| Variable_name                   | Value                                     |
+---------------------------------+-------------------------------------------+
| binlog_gtid_simple_recovery     | ON                                        |
| enforce_gtid_consistency        | ON                                        |
| gtid_executed                   |                                           |
| gtid_executed_compression_period | 1000                                      |
| gtid_mode                       | ON                                        |
| gtid_next                       | AUTOMATIC                                 |
| gtid_owned                      |                                           |
| gtid_purged                     | c241b625-e932-11e5-bb11-5254f035dabc:1-11558 |  -- 被purge的部分
| session_track_gtids             | OFF                                       |
+---------------------------------+-------------------------------------------+
9 rows in set (0.00 sec)


--
-- Slave 3 端
--
mysql> set @@global.gtid_purged='c241b625-e932-11e5-bb11-5254f035dabc:1-11558';   -- 跳过该部分
Query OK, 0 rows affected (0.01 sec)

mysql> stop slave;
Query OK, 0 rows affected (0.01 sec)

mysql> start slave;
Query OK, 0 rows affected (0.07 sec)

mysql> show slave status\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.18.14.71
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000003
          Read_Master_Log_Pos: 1032
               Relay_Log_File: relay.000004
                Relay_Log_Pos: 1233
        Relay_Master_Log_File: bin.000003
             Slave_IO_Running: Yes    -- IO线程正常
            Slave_SQL_Running: Yes    -- SQL线程正常
              Replicate_Do_DB: test_dodb  -- 只复制test_dodb的库
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 1032
              Relay_Log_Space: 1671
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           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: 101
                  Master_UUID: 4144fe65-fcfc-11e5-9c82-5254f03466c1
             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 more updates
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp:
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set: 4144fe65-fcfc-11e5-9c82-5254f03466c1:1-2,
c241b625-e932-11e5-bb11-5254f035dabc:11559-11561
            Executed_Gtid_Set: 4144fe65-fcfc-11e5-9c82-5254f03466c1:1-2,
c241b625-e932-11e5-bb11-5254f035dabc:1-11561
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| test_dodb          | -- 复制过来了
+--------------------+
5 rows in set (0.00 sec)
```