MySQL学习笔记（Day044：replication_4-GTID）

MySQL学习

一. GTID复制出错的处理

注意：按照之前给出的 `/etc/my.cnf` 的配置，且在Slave上配置 `read_only=1;` ，复制是 不会出错 的。

这里仅仅演示一下如果人为的在从机上误操作导致的复制失败，如何恢复。

1.1. 在从机上插入一条记录（模拟误操作）

现在 `Slave2` 上插入一条记录（*现实中如果配置了readonly，在app中是无法插入的，app不会给root权限*）

```
--
-- Slave2 端
--
mysql> insert into t_a(b) values(10);
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)

mysql> select * from t_a;
+---+------+
| a | b    |
+---+------+
| 1 |   10 |  -- a是主键
+---+------+
1 row in set (0.01 sec)
```

1.2. 在主机上插入同样的记录

```
--
-- Master 端
--
mysql> insert into t_a(b) values(10);
Query OK, 1 row affected (0.00 sec)

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

查看 `Slave2` 上的状态

```
--
-- Slave2 端
--
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.000023
          Read_Master_Log_Pos: 844
               Relay_Log_File: relay.000011
                Relay_Log_Pos: 603
        Relay_Master_Log_File: bin.000023
             Slave_IO_Running: Yes
            Slave_SQL_Running: No
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table:
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 1062  -- 记录冲突了
                   Last_Error: Coordinator stopped because there were error(s) in the worker(s). The most recent failure being: Worker 0 failed executing transaction 'c241b625-e932-11e5-bb11-5254f035dabc:11562' at master log bin.000023, end_log_pos 813. See error log and/or performance_schema.replication_applier_status_by_worker table for more details
about this failure or others, if any.
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 575
              Relay_Log_Space: 1109
              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: NULL
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 1062   -- 错误1062，记录重复了
               Last_SQL_Error: Coordinator stopped because there were error(s) in the worker(s). The most recent failure being: Worker 0 failed executing transaction 'c241b625-e932-11e5-bb11-5254f035dabc:11562' at master log bin.000023, end_log_pos 813. See error log and/or performance_schema.replication_applier_status_by_worker table for more details
about this failure or others, if any.
  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:
           Master_Retry_Count: 86400
                  Master_Bind:
      Last_IO_Error_Timestamp:
     Last_SQL_Error_Timestamp: 160410 01:08:08
               Master_SSL_Crl:
           Master_SSL_Crlpath:
           Retrieved_Gtid_Set: c241b625-e932-11e5-bb11-5254f035dabc:11561-11562
            Executed_Gtid_Set: 4729a5ec-fcfc-11e5-8c97-5254f0d46611:1-4,
c241b625-e932-11e5-bb11-5254f035dabc:1-11561  -- 只执行到了11561:
                                              -- 11562这个事物报错了。
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)
```

> 其实从数据一致性看，目前主从数据是一致的，只是复制过来的数据在回放时，发现了自己已经有了该部分数据（*Error:1062*），从而引发了复制异常（*SQL回放线程停止*）。
>
> 我们只需要告诉MySQL，“跳过”这部分一样的GTID，继续复制，即可。

1.3. 处理复制错误

这里的 跳过 的方法很巧妙，步骤如下

1. **将Slave上的 `gtid_next` 指向 执行失败 的那个 `gtid`**
   - 这里执行失败的 `gtid` 报错信息中已经给出： `'c241b625-e932-11e5-bb11-5254f035dabc:11562'`
   - 如果不看报错信息，可以看 `Retrieved_Gtid_Set` 和 `Executed_Gtid_Set` 的对比结果
2. **执行一个空的事物，即 `begin;commit;`**
   - 这样就把 失败的gtid 对应到了一个 空的事物 上，这个步骤即为 “跳过” 的意思
3. **将 `gtid_next` 设置（*还原*）为 `automatic`**

```
--
-- Slave 2 端
--
mysql> select @@gtid_next;  -- 当前为默认值， AUTOMATIC
+-------------+
| @@gtid_next |
+-------------+
| AUTOMATIC   |
+-------------+
1 row in set (0.00 sec)

-- -- 步骤1 ： 设置 gtid_next 为回放失败的gtid
mysql> set gtid_next='c241b625-e932-11e5-bb11-5254f035dabc:11562'; -- 设置为之前失败的那个GTID的值
Query OK, 0 rows affected (0.00 sec)

-- -- 步骤2 : 执行一个空的事物，让回放失败的gtid对应到这个空的事物
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

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

-- -- 步骤3 : 还原gtid_next为automatic
mysql> set gtid_next="automatic";
Query OK, 0 rows affected (0.00 sec)
```

> 至此，回放失败的gtid被我们映射到了一个空的事物上，且此时 数据层面上其实是一致 的（*至少这个gtid之前的事物是一致的，或者现在的Slave是落后Master的*），我们再 重启复制 ，让Slave去追Master即可。

```
--
-- Slave 2端
--
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: 172.18.14.70
                  Master_User: rpl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: bin.000023
          Read_Master_Log_Pos: 844
               Relay_Log_File: relay.000013
                Relay_Log_Pos: 436
        Relay_Master_Log_File: bin.000023
             Slave_IO_Running: Yes  -- 恢复正常
            Slave_SQL_Running: Yes  -- 恢复正常
              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: 844
              Relay_Log_Space: 915
              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: c241b625-e932-11e5-bb11-5254f035dabc:11561-11562
            Executed_Gtid_Set: 4729a5ec-fcfc-11e5-8c97-5254f0d46611:1-4,
c241b625-e932-11e5-bb11-5254f035dabc:1-11562  -- 执行到了11562
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)
```

1.4. 测试复制

1. Master端插入一个测试数据

```
--
-- Master 端
--
mysql> insert into t_a(b) values(20);
Query OK, 1 row affected (0.00 sec)

mysql> commit;
Query OK, 0 rows affected (0.03 sec)
```

2. 查看Slave2上的数据是否同步

```
--
-- Slave 2 端
--
mysql> select * from t_a;
+---+------+
| a | b    |
+---+------+
| 1 |   10 |
| 2 |   20 |  -- 立即同步过来的数据
+---+------+
2 rows in set (0.00 sec)
```

至此，GTID复制出错的处理就完成了。
注意：这里仅仅是 跳过错误 ，和原来的 `sql_slave_skip_counter` （ *该功能在GTID下失效* ）功能类似，无法保证主从数据是一致的（ *需要人工介入进行确认，比如仅仅主键一样，其他列不一样* ）

> 如果出现了很多的GTID的错误，可能是从机上有大量的操作，建议 重新搭建主从复制
> **但还是要在源头上避免此类情况的发生，确保在从机上开启 `read_only=1` ，并且避免人工的误操作。**

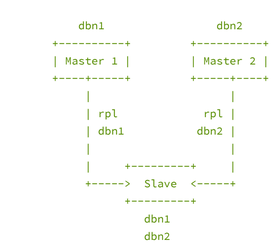
最后强调： `GTID` 是 基于事物 的复制，一致性要求很高， 强烈建议 在 `Slave` 上开启 `read_only=1`

二. 多源复制

2.1. 多源复制的介绍

> 官方文档 – replication-channels

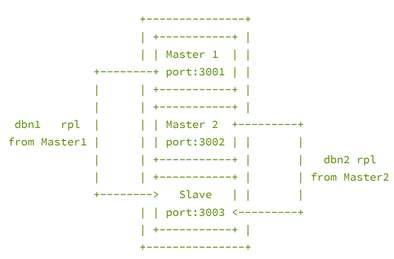
多源复制 是指一个从机，多个主机的复制，从 MySQL5.7.6 才有的功能，如下图所示：



在语法层面上，只是在 原来的change master 的基础上，增加了 `for channel 'channel_name'`

2.2. 多源复制的演示

这里为了简化拓扑，在一台主机上使用多实例方式来模拟多主机，如下图所示：



2.2.1. 准备数据

```
--
-- Master 1 端
--
mysql> select @@port;
+--------+
| @@port |
+--------+
|   3001 |
+--------+
1 row in set (0.00 sec)

mysql> grant replication slave on *.* to 'rpl'@'%' identified by '123';  -- 建议先创建用户，这里简化步骤
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> create database dbn1;
Query OK, 1 row affected (0.01 sec)

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

mysql> use dbn1;
Database changed

mysql> create table t_a(a int auto_increment primary key);
Query OK, 0 rows affected (0.16 sec)

mysql> insert into t_a values(NULL);
Query OK, 1 row affected (0.04 sec)

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

mysql> select * from t_a;
+---+
| a |
+---+
| 1 |
+---+
1 row in set (0.00 sec)

mysql> show master status\G
*************************** 1. row ***************************
             File: bin.000003
         Position: 1602
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: 1d0117c6-fe69-11e5-90cb-5254a03976fb:1-139    -- 执行到139
1 row in set (0.00 sec)
```

```
##
## Master 1 端
##
# 备份dbn1库，届时还原到Slave上
[root@MyServer ~]> mysqldump -uroot -p123 -h 127.0.0.1 -P 3301 -B dbn1 > dbn1.sql

[root@MyServer ~]> cat dbn1.sql
# ------------省略其他输出------------
# 在恢复备份的时候，会自动执行该语句，就不需要我们手工跳过了，如果是mydumper之类的，需要手工跳过
SET @@GLOBAL.GTID_PURGED='1d0117c6-fe69-11e5-90cb-5254a03976fb:1-139';
# ------------省略其他输出------------
```

```
--
-- Master 2 端
--
mysql> select @@port;
+--------+
| @@port |
+--------+
|   3002 |
+--------+
1 row in set (0.00 sec)

mysql> grant replication slave on *.* to 'rpl'@'%' identified by '123';
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> create database dbn2;
Query OK, 1 row affected (0.02 sec)

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

mysql> use dbn2;
Database changed

mysql> create table b(a int auto_increment primary key);
Query OK, 0 rows affected (0.15 sec)

mysql> insert into b values(NULL);
Query OK, 1 row affected (0.05 sec)

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

mysql> select * from b;
+---+
| a |
+---+
| 1 |
+---+
1 row in set (0.00 sec)

mysql> show master status\G
*************************** 1. row ***************************
             File: bin.000003
         Position: 1598
     Binlog_Do_DB:
 Binlog_Ignore_DB:
Executed_Gtid_Set: a64ea37e-fe69-11e5-9867-5254a03976fb:1-139  - 执行到139
1 row in set (0.00 sec)
```

```
##
## Master 2 端
##
# 备份dbn2库，届时还原到Slave上
[root@MyServer ~]> mysqldump -uroot -p123 -h 127.0.0.1 -P 3002 -B dbn2 > dbn2.sql

[root@MyServer ~]> cat dbn2.sql
# ------------省略其他输出------------
# 在恢复备份的时候，会自动执行该语句，就不需要我们手工跳过了，如果是mydumper之类的，需要手工跳过
SET @@GLOBAL.GTID_PURGED='a64ea37e-fe69-11e5-9867-5254a03976fb:1-139';
# ------------省略其他输出------------
```

2.2.2. 还原Slave

```
##
## Slave 端
##
[root@MyServer ~]> mysql -u root -p  -S /tmp/mysql.sock_1003 < dbn1.sql
Enter password:
ERROR 1840 (HY000) at line 24: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.
```

还是之前的问题，需要在Slave上，先操作一下 `reset master` ，以清空 `@@GLOBAL.GTID_EXECUTED` 。

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


[root@MyServer ~]> mysql -u root -p  -S /tmp/mysql.sock_1003 < dbn1.sql
Enter password:
# -- 执行成功
[root@MyServer ~]> mysql -u root -p  -S /tmp/mysql.sock_1003 < dbn2.sql
Enter password:
ERROR 1840 (HY000) at line 24: @@GLOBAL.GTID_PURGED can only be set when @@GLOBAL.GTID_EXECUTED is empty.
## -- 再次执行 reset master 后，再恢复dbn2.sql
[root@MyServer ~]> mysql -u root -p  -S /tmp/mysql.sock_1003 < dbn2.sql
Enter password:
# -- 执行成功


mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| dbn1               |
| dbn2               |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
6 rows in set (0.00 sec)

mysql> select * from dbn1.t_a;
+---+
| a |
+---+
| 1 |
+---+
1 row in set (0.00 sec)

mysql> select * from dbn2.b;
+---+
| a |
+---+
| 1 |
+---+
1 row in set (0.00 sec)

-- 此时dbn1和dbn2的数据就恢复完成了
```

2.2.3. change master

先在Slave上设置 需要复制的库 **，如果不设置的话，默认会同步系统的库（mysql），这样可能会复制出错（** *因为master1和master2上的mysql库中可能有相同的记录* **）。**

```
##
## Slave 端
##
[mysqld3003]
# 只复制dbn1 和 dbn2，这个步骤很重要，且有多个时，必须分多行写
replicate_do_db=dbn1
replicate_do_db=dbn2



--
-- Slave 端
--
mysql> select @@port;
+--------+
| @@port |
+--------+
|   3003 |
+--------+
1 row in set (0.00 sec)

--
-- 复制 Master1 上的数据
--
mysql> change master to master_host='127.0.0.1', master_port=3001, master_user='rpl', master_password='123', master_auto_position=1 for channel 'ch1';  -- 多源复制 channel : ch1
Query OK, 0 rows affected, 2 warnings (0.17 sec)

mysql> start slave for channel 'ch1';  -- 启动ch1
Query OK, 0 rows affected (0.03 sec)

mysql> show slave status for channel 'ch1'\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 127.0.0.1
                  Master_User: rpl
                  Master_Port: 3001
                Connect_Retry: 60
              Master_Log_File: bin.000003
          Read_Master_Log_Pos: 1602
               Relay_Log_File: relay-ch1.000004
                Relay_Log_Pos: 1819
        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: dbn1,dbn2
          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: 1602
              Relay_Log_Space: 2277
              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: 1001
                  Master_UUID: 1d0117c6-fe69-11e5-90cb-5254a03976fb
             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: 1d0117c6-fe69-11e5-90cb-5254a03976fb:1:31-139
            Executed_Gtid_Set: 1d0117c6-fe69-11e5-90cb-5254a03976fb:1-139,
a64ea37e-fe69-11e5-9867-5254a03976fb:1-139
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name: ch1   -- channel为ch1
           Master_TLS_Version:
1 row in set (0.00 sec)

--
-- 复制 Master2 上的数据
--
mysql> change master to master_host='127.0.0.1', master_port=3002, master_user='rpl', master_password='123', master_auto_position=1 for channel 'ch2';  -- 多源复制 channel : ch2
Query OK, 0 rows affected, 2 warnings (0.20 sec)

mysql> start slave for channel 'ch2';  -- 启动ch2
Query OK, 0 rows affected (0.04 sec)

mysql> show slave status for channel 'ch2'\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 127.0.0.1
                  Master_User: rpl
                  Master_Port: 3002
                Connect_Retry: 60
              Master_Log_File: bin.000003
          Read_Master_Log_Pos: 1598
               Relay_Log_File: relay-ch2.000002
                Relay_Log_Pos: 396
        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: dbn1,dbn2
          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: 1598
              Relay_Log_Space: 597
              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: 1002
                  Master_UUID: a64ea37e-fe69-11e5-9867-5254a03976fb
             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: 1d0117c6-fe69-11e5-90cb-5254a03976fb:1-139,
a64ea37e-fe69-11e5-9867-5254a03976fb:1-139
                Auto_Position: 1
         Replicate_Rewrite_DB:
                 Channel_Name: ch2   -- channel为ch2
           Master_TLS_Version:
1 row in set (0.00 sec)
```

```
--
-- Master 1 端
--
mysql> insert into t_a values(NULL);
Query OK, 1 row affected (0.03 sec)

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

--
-- Master 2 端
--
mysql> insert into b values(NULL);
Query OK, 1 row affected (0.05 sec)

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

--
-- Slave 端
--
mysql> select * from dbn1.t_a;
+---+
| a |
+---+
| 1 |
| 2 |
+---+
2 rows in set (0.00 sec)

mysql> select * from dbn2.b;
+---+
| a |
+---+
| 1 |
| 2 |
+---+
2 rows in set (0.00 sec)
```

至此，多源复制的主从搭建就完成了。

2.2.4. 使用场景

如果 `Master2` 上也有一个 `dbn1` 的库，会有问题么？

> 1. 如果不做额外的配置，是 会有错误 的；
> 2. 如果配置了 `slave_skip_errors = ddl_exist_errors` ，且 没有重复数据 话，复制关系还是正常的。

这种操作可以起到 数据聚合 的效果。将分库分表后的数据聚合在一起，以供其他应用进行分析（**前提是数据不能有重复**）。
但是最合适的场景还是将 不同的库 进行复制。

三. 其他注意事项

1. 中间件的unique key
   中间件可以保证分区键是唯一的（比如order_id），但是对于其他唯一索引来说，需要业务层去保证。
2. reset slave all
   使用该命令时 不会清空数据 ，仅仅是清空 `show slave status\G` 里面的信息，所以在使用该命令之前，请 先记录（备份） `show slave status\G` 的信息。
3. GTID
   在开启GTID后，不能在一个事物中使用创建临时表的语句，需要使得 `autocommit=1;` 才可以。
   在开启GTID后，不能使用 `create table select ...` 的语法来创建表了，因为这其实是多个事物了，GTID没法对应。