MySQL 主从复制及读写分离

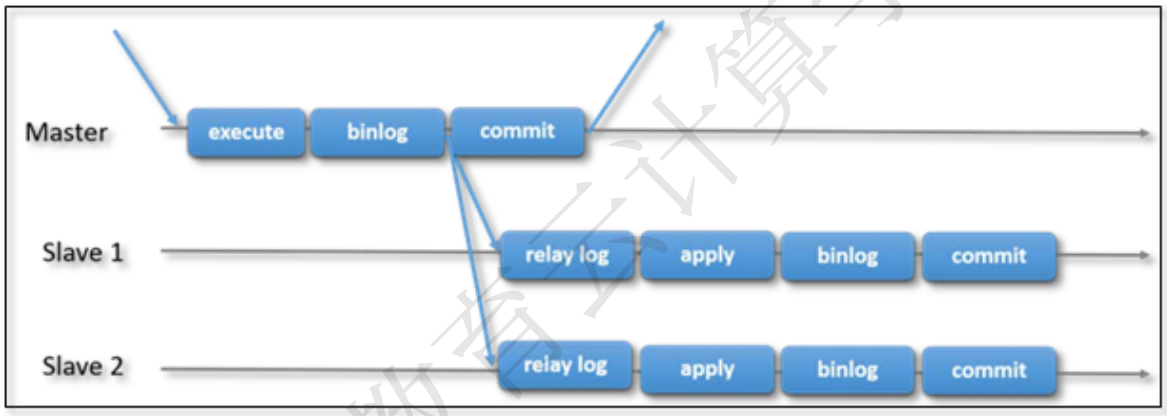
一、MySQL 复制的应用常见场景

- 读写分离，提高查询访问性能，有效减少主数据库访问压力。
- 实时灾备，主数据库出现故障时，可快速切换到从数据库。
- 数据汇总，可将多个主数据库同步汇总到一个从数据库中，方便数据统计分析。

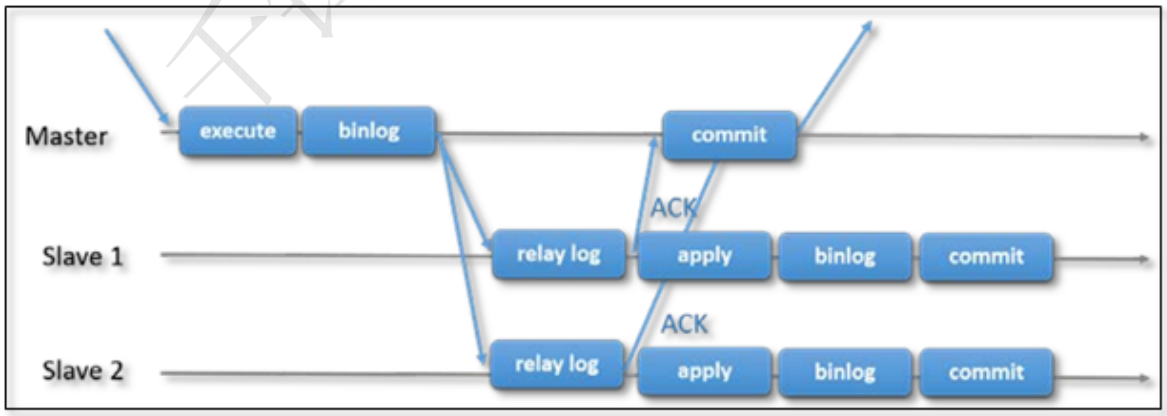
二、MySQL 主从复制原理介绍

1、MySQL 异步和半同步复制

传统的 MySQL 复制提供了一种简单的主–从复制方法。 有一个主，以及一个或多个从。 主节点执行和提交事务，然后将它们（异步地）发送到从节点，以重新执行（在基于语句的复制中）或应用（在基于行的复制中）。 这是一个 shared-nothing 的系统，默认情况下所有 server 成员都有一个完整的数据副本



还有一个半同步复制，它在协议中添加了一个同步步骤。 这意味着主节点在提交时需要等待从节点确认它已经接收到事务。只有这样，主节点才能继续提交操作。



在上面的两个图片中，可以看到传统异步 MySQL 复制协议（以及半同步）的图形展示。 蓝色箭头表示在不同 server 之间或者 server 与 client 应用之间的信息交互。

2、MySQL 主从复制过程

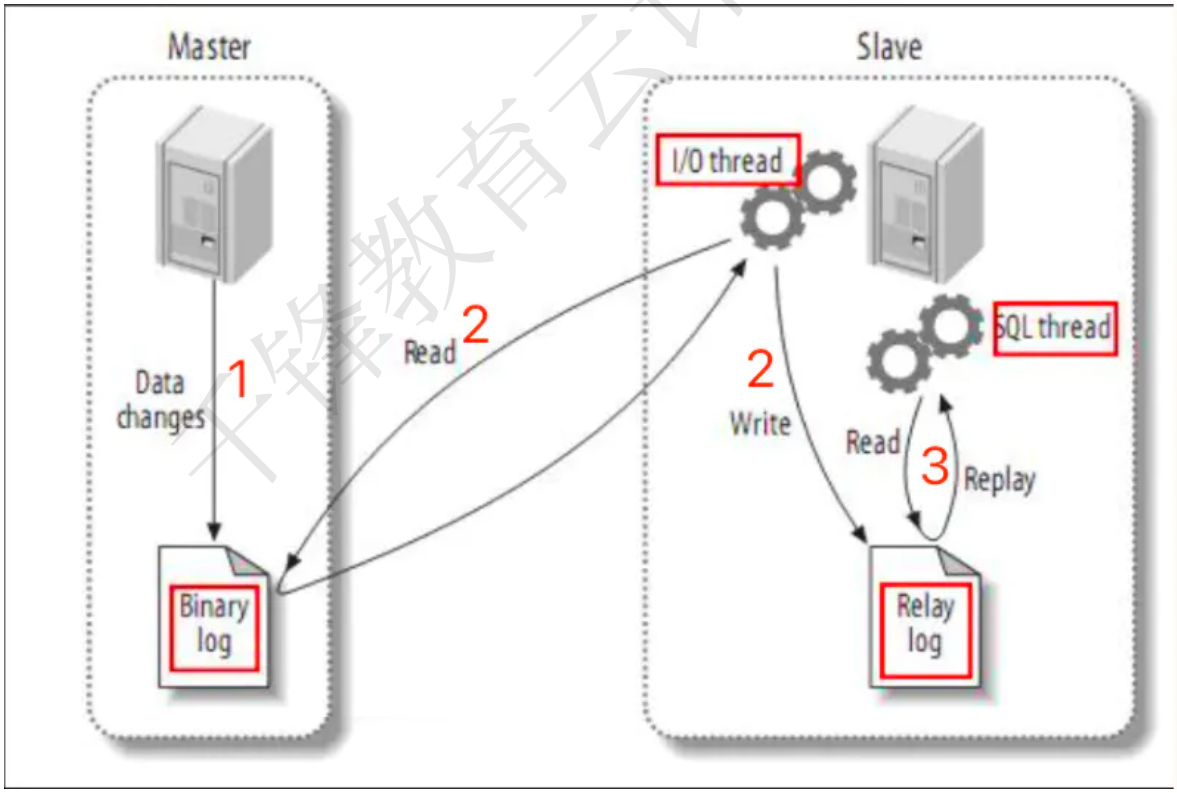
- 开启binlog日志，通过把主库的 binlog 传送到从库，从新解析应用到从库。
- 复制需要3个线程（dump、io、sql）完成。

- 复制是异步的过程。主从复制是异步的逻辑的SQL语句级的复制。

3、MySQL 主从复制前提

- 主服务器一定要打开二进制日志
- 必须两台服务器（或者是多个实例）
- 从服务器需要一次数据初始化
- 如果主从服务器都是新搭建的话，可以不做初始化
- 如果主服务器已经运行了很长时间了，可以通过备份将主库数据恢复到从库。
- 主库必须要有对从库复制请求的用户。
- 从库需要有relay-log设置，存放从主库传送过来的二进制日志 show variables like '%relay%';
- 在第一次的时候，从库需要change master to 去连接主库。
- change master信息需要存放到 master.info 中 show variables like '%master_info%';
- 从库怎么知道，主库发生了新的变化?通过relay-log.info记录的已经应用过的relay-log信息。
- 在复制过程中涉及到的线程
  - 从库会开启一个IO thread(线程)，负责连接主库，请求binlog，接收binlog并写入relay-log。
  - 从库会开启一个SQL thread(线程)，负责执行relay-log中的事件。
  - 主库会开启一个dump thrad(线程)，负责响应从IO thread的请求。

4、MySQL 主从复制实现



- 通过二进制日志
- 至少两台（主、从）
- 主服务器的二进制日志“拿”到从服务器上再运行一遍。
- 通过网络连接两台机器，一般都会出现延迟的状态。也可以说是异步的。
- 从库通过手工执行change master to 语句连接主库，提供了连接的用户一切条件 user、password、port、ip
- 并且让从库知道，二进制日志的起点位置（file名 position号）

- 启动从库同步服务 *start slave*
- 从库的IO和主库的dump线程建立连接
- 从库根据change master to 语句提供的file名和position号，IO线程向主库发起binlog的请求
- 主库dump线程根据从库的请求，将本地binlog以events的方式发给从库IO线程
- 从库IO线程接收binlog evnets，并存放到本地relay-log中，传送过来的信息，会记录到master.info中。
- 从库SQL线程应用relay-log，并且把应用过的记录到relay-log.info,默认情况下，已经应用过的relay会自动被清理purge。

2、MySQL复制有三种核心格式

- 复制的工作原理是数据库修改记录到bin log日志并传递到slave，然后slave在本地还原的过程。而时间记录到bin log的格式会有所不同。

1、基于语句的复制（statement based replication）

- 基于主库将SQL语句写入到bin log中完成复制。

2、基于行数据的复制（row based replication）：

- 基于主库将每一行数据变化的信息作为时间写入到bin log中完成日志。默认就是基于行级别的复制，因为它相对语句复制逻辑更为严谨。

3、混合复制（mixed based replication）：

- 上述两者的结合。默认情况下优先使用基于语句的复制，只有当部分语句如果基于语句复制不完全的情况下才会自动切换为基于行数据的复制。

```
[root@master.qfedu.com ~]# mysql -uroot -p'Qfedu.123com'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> SHOW VARIABLES LIKE '%BINLOG_FORMAT%';              # 默认就是基于行复制的
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| binlog_format | ROW   |
+---------------+-------+
row in set (0.00 sec)

mysql>
mysql> SET binlog_format='STATEMENT';                      # 修改成基于语句复制
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> SHOW VARIABLES LIKE '%BINLOG_FORMAT%';
```

```
+---------------+-----------+
| Variable_name | Value     |
+---------------+-----------+
| binlog_format | STATEMENT |
+---------------+-----------+
row in set (0.00 sec)

mysql>
mysql> quit
Bye
```

- 只作为修改默认的复制模式格式案例（推荐大家使用默认的基于ROW的复制方式）

三、MySQL 传统主从同步配置

1、MySQL 主从部署环境

- Master： 192.168.172.110
- Slave： 192.168.172.111
- 端口： 3306
- Master, Slave 按照以下步骤安装mysql数据库

2、MySQL yum包下载

```
[root@master.qfedu.com ~]# wget http://repo.mysql.com/mysql57-community-release-el7-10.noarch.rpm
```

3、MySQL 软件源安转

```
[root@master.qfedu.com ~]# yum -y install mysql57-community-release-el7-10.noarch.rpm
```

4、MySQL 服务安装

```
[root@master.qfedu.com ~]# yum install  -y  mysql-community-server mysql
```

5、MySQL 服务启动

```
[root@master.qfedu.com ~]# systemctl start mysqld.service
```

6、MySQL 服务运行状态检查

```
[root@master.qfedu.com ~]# systemctl status mysqld.service
```

7、MySQL 修改临时密码

1、获取MySQL的临时密码

- 为了加强安全性，MySQL5.7为root用户随机生成了一个密码，在error log中，关于error log的位置，如果安装的是RPM包，则默认是/var/log/mysqld.log。
- 只有启动过一次mysql才可以查看临时密码

```
[root@master.qfedu.com ~]# grep 'temporary password' /var/log/mysqld.log
```

2、登陆并修改密码

1、使用默认的密码登陆

```
[root@master.qfedu.com ~]# mysql -uroot -p
```

2、修改密码

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Qfedu.123com';
```

3、解决不符合密码复杂性要求

```
ERROR 1819 (HY000): Your password does not satisfy the current policy
requirements
```

1、修改 密码策略

```
mysql> set global validate_password_policy=0;
```

2、修改密码的长度

```
mysql> set global validate_password_length=1;
```

3、执行修改密码成功

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'root123';
```

8、MySQL 授权远程主机登录

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'slave'@'192.168.%.%' IDENTIFIED BY
'mypassword' WITH GRANT OPTION;
mysql> FLUSH  PRIVILEGES;
```

9、MySQL 编辑配置文件

1、Master 配置文件

```
[root@master.qfedu.com ~]# cat /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
default-storage-engine=INNODB
symbolic-links=0
server_id=6
log_bin=/var/log/mysql/mysql-bin
```

2、Slave 配置文件

```
[root@slave.qfedu.com ~]# cat /etc/my.cnf
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
default-storage-engine=INNODB
symbolic-links=0
server_id=8
log_bin=/var/log/mysql/mysql-bin
```

10、MySQL 创建主从同步账号

- 在主库创建一个专门用来复制的数据库用户，所有从库都用这个用户来连接主库，确保这个用户只有复制的权限

```
[root@master.qfedu.com ~]# mysql -uroot -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> CREATE USER 'slave'@'192.168.%.%' IDENTIFIED BY 'Qfedu.123com';
Query OK, 0 rows affected (0.01 sec)

mysql>
mysql> GRANT REPLICATION SLAVE ON *.* TO 'slave'@'192.168.%.%';
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> quit
Bye
```

11、MySQL 测试账号远程登录

```
[root@slave.qfedu.com ~]# mysql -uslave -p'Qfedu.123com' -P 3306 -h
master.qfedu.com
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 20
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql>
mysql> QUIT
Bye
```

12、MySQL 获取主库的日志信息并生成主库数据镜像

```
mysql> FLUSH TABLES WITH READ LOCK;                    # 对主库上所有表加锁，停止修改，即在从库复制的过程中主库不能执行UPDATA，DELETE，INSERT语句!
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> SHOW MASTER STATUS;                             # 获取主库的日志信息，file 表示当前日志文件名称，position 表示当前日志的位置
+---------------------------------+----------+--------------+------------------+-------------------+
| File                            | Position | Binlog_Do_DB | Binlog_Ignore_DB | Executed_Gtid_Set |
+---------------------------------+----------+--------------+------------------+-------------------+
| /var/log/mysql/mysql-bin.000002 |     4095 |              |                  |                   |
+---------------------------------+----------+--------------+------------------+-------------------+
1 row in set (0.00 sec)

mysql>
```

13、MySQL 备份主库数据

```
[root@master.qfedu.com ~]#  mysqldump  -u root -p'Qfedu.123com' --master-data --all-databases > qfedu.com-master.sql
mysqldump: [Warning] Using a password on the command line interface can be insecure.

[root@master.qfedu.com ~]# ll
total 776
-rw-r--r-- 1 root root 791962 Jun 10 19:24 qfedu.com-master.sql
```

- 主库数据备份完毕后，释放主库锁，需要注意，在上锁这一段期间，无法对数据库进行写操作，比如UPDATA，DELETE，INSERT

```
mysql> UNLOCK TABLES;
Query OK, 0 rows affected (0.00 sec)
```

14、MySQL 从库还原数据

- 将主库的镜像拷贝当从库中，将数据还原到从库

```
[root@master.qfedu.com ~]# ll
total 776
-rw-r--r-- 1 root root 791962 Jun 10 19:24 qfedu.com-master.sql

[root@master.qfedu.com ~]# scp qfedu.com-master.sql slave.qfedu.com:/root/
The authenticity of host 'slave.qfedu.com (192.168.172.111)' can't be
established.
ECDSA key fingerprint is SHA256:BkN6bKO2q5zMgvremE/3rOIsCaq9eTPudgfUOlhbqGo.
ECDSA key fingerprint is MD5:75:bd:cb:be:35:f8:45:a2:ea:74:bc:aa:29:74:4d:0d.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'slave.qfedu.com,192.168.172.107' (ECDSA) to the list
of known hosts.
root@slave.qfedu.com's password:
qfedu.com-master.sql                                      100%  773KB  60.7MB/s
  00:00
```

- 从库数据还原

```
[root@slave.qfedu.com ~]# pwd
/root

[root@slave.qfedu.com ~]# ll
total 776
-rw-r--r-- 1 root root 791962 Jun 10 19:37 qfedu.com-master.sql

[root@slave.qfedu.com ~]# mysql -uroot -p'Qfedu.123com'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> source qfedu.com-master.sql;
Query OK, 0 rows affected (0.00 sec)
Query OK, 0 rows affected (0.00 sec)
.......
Query OK, 0 rows affected (0.00 sec)
mysql>
```

15、MySQL 从库配置同步

- 在从库上建立复制关系，即从库指定主库的日志信息和链接信息

```
mysql> CHANGE MASTER TO
    -> MASTER_HOST='master.qfedu.com',
    -> MASTER_PORT=3306,
    -> MASTER_USER='slave',
    -> MASTER_PASSWORD='Qfedu.123com',
    -> MASTER_LOG_FILE='/var/log/mysql/mysql-bin.000002',
    -> MASTER_LOG_POS=4095;
Query OK, 0 rows affected, 2 warnings (0.01 sec)
mysql>
```

16、MySQL 从库启动复制进程

```
mysql> START SLAVE;
Query OK, 0 rows affected (0.00 sec)

mysql>
mysql> SHOW SLAVE STATUS\G
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: master.qfedu.com
                  Master_User: copy
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: /var/log/mysql/mysql-bin.000002
          Read_Master_Log_Pos: 4095
               Relay_Log_File: node107-relay-bin.000002
                Relay_Log_Pos: 332
        Relay_Master_Log_File: /var/log/mysql/mysql-bin.000002
             Slave_IO_Running: Yes    # 观察IO进程是否为yes，如果为yes说明正常，如果长时间处于"Connecting"状态就得检查你的从库指定的主库的链接信息是否正确
            Slave_SQL_Running: Yes    # 观察SQL进程是否为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: 4095
              Relay_Log_Space: 541
              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      #该参数表示从库和主库有多少秒的延迟，再过多少秒数据和主库保持一致，如果为0表示当前从库和主库的数据是一致的，如果该数较大的话你得考虑它的合理性。需要注意下该参数的值。
```

```
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: 106
                  Master_UUID: 74227047-8b60-11e9-8cba-000c29985293
             Master_Info_File: /var/lib/mysql/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:
                Auto_Position: 0
         Replicate_Rewrite_DB:
                 Channel_Name:
           Master_TLS_Version:
1 row in set (0.00 sec)

mysql>
```

17、MySQL 验证主从数据同步

1、主库中创建 qfedu 数据库

```
[root@master.qfedu.com ~]# mysql -uroot -p'Qfedu.123com'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 22
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> CREATE DATABASE qfedu DEFAULT CHARACTER SET = utf8;
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> GRANT ALL PRIVILEGES ON qfedu.* TO 'qfedu'@'192.168.172.%' IDENTIFIED BY
'Qfedu.123com' WITH GRANT OPTION;
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql>
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| qfedu                |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> QUIT
Bye
```

2、从库的服务器观察数据是否同步

```
[root@slave.qfedu.com ~]# mysql -uroot -p'Qfedu.123com'
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 5.7.25-log MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| qfedu              |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

mysql> QUIT
Bye
[root@slave.qfedu.com ~]#
```

3、测试完成后删除 qfedu 库，

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| qfedu              |
```

```
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)


mysql> drop database qfedu;
Query OK, 0 rows affected (0.00 sec)

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

mysql>
```

四、MySQL 基于 GTID 的主从复制

1、GTID 概念介绍

- GTID即全局事务ID (global transaction identifier), 其保证为每一个在主上提交的事务在复制集群中可以生成一个唯一的ID。
- GTID最初由google实现，官方MySQL在5.6才加入该功能。mysql主从结构在一主一从情况下对于GTID来说就没有优势了，而对于2台主以上的结构优势异常明显，可以在数据不丢失的情况下切换新主。
- 使用GTID需要注意: 在构建主从复制之前，在一台将成为主的实例上进行一些操作（如数据清理等），通过GTID复制，这些在主从成立之前的操作也会被复制到从服务器上，引起复制失败。也就是说通过GTID复制都是从最先开始的事务日志开始，即使这些操作在复制之前执行。比如在server1上执行一些drop、delete的清理操作，接着在server2上执行change的操作，会使得server2也进行server1的清理操作。
- GTID实际上是由UUID+TID (即transactionId)组成的。其中UUID(即server_uuid) 产生于auto.conf文件(cat /data/mysql/data/auto.cnf)，是一个MySQL实例的唯一标识。TID代表了该实例上已经提交的事务数量，并且随着事务提交单调递增，所以GTID能够保证每个MySQL实例事务的执行（不会重复执行同一个事务，并且会补全没有执行的事务）。GTID在一组复制中，全局唯一。 下面是一个GTID的具体形式 :

2、GTID 的工作流程为

- master更新数据时，会在事务前产生GTID，一同记录到 binlog 日志中。
- slave 端的 i/o 线程将变更的 binlog，写入到本地的 relay log 中。
- sql 线程从 relay log 中获取 GTID，然后对比 slave 端的 binlog 是否有记录。
- 如果有记录，说明该 GTID 的事务已经执行，slave 会忽略。
- 如果没有记录，slave 就会从 relay log 中执行该 GTID 的事务，并记录到 binlog。
- 在解析过程中会判断是否有主键，如果没有就用二级索引，如果没有就用全部扫描。

3、GTID 的优点

- 一个事务对应一个唯一ID，一个GTID在一个服务器上只会执行一次;
- GTID是用来代替传统复制的方法，GTID复制与普通复制模式的最大不同就是不需要指定二进制文件名和位置;
- 减少手工干预和降低服务故障时间，当主机挂了之后通过软件从众多的备机中提升一台备机为主机;

4、GTID 复制同步过程

- 主从架构：ServerC <-----ServerA ----> ServerB 即一个主数据库ServerA，两个从数据库ServerB和ServerC
- 当主机ServerA 挂了之后 ，此时ServerB执行完了所有从ServerA 传过来的事务，ServerC 延时一点。这个时候需要把 ServerB 提升为主机 ，Server C 继续为备机；当ServerC 链接ServerB 之后,首先在自己的二进制文件中找到从ServerA 传过来的最新的GTID，然后将这个GTID 发送到ServerB ,ServerB 获得这个GTID之后,就开始从这个GTID的下一个GTID开始发送事务给ServerC。这种自我寻找复制位置的模式减少事务丢失的可能性以及故障恢复的时间。

5、GTID 的缺点(限制)

- 不支持非事务引擎;
- 不支持create table ... select 语句复制(主库直接报错);(原理: 会生成两个sql, 一个是DDL创建表SQL, 一个是insert into 插入数据的sql; 由于DDL会导致自动提交, 所以这个sql至少需要两个GTID, 但是GTID模式下, 只能给这个sql生成一个GTID)
- 不允许一个SQL同时更新一个事务引擎表和非事务引擎表;
- 在一个复制组中，必须要求统一开启GTID或者是关闭GTID;
- 开启GTID需要重启 (mysql5.7除外);
- 开启GTID后，就不再使用原来的传统复制方式;
- 对于create temporary table 和 drop temporary table语句不支持;
- 不支持sql_slave_skip_counter;

6、GTID 与 binlog对应关系

- 假设有4个binlog: bin.001,bin.002,bin.003,bin.004

```
bin.001 : Previous-GTIDs=empty; binlog_event有：1-40
bin.002 : Previous-GTIDs=1-40;  binlog_event有：41-80
bin.003 : Previous-GTIDs=1-80;  binlog_event有：81-120
bin.004 : Previous-GTIDs=1-120;  binlog_event有：121-160
```

- 假设现在我们要找GTID=$A，那么MySQL的扫描顺序为： 从最后一个binlog开始扫描（即：bin.004）
- bin.004的Previous-GTIDs=1-120，如果$A=140 > Previous-GTIDs,那么肯定在bin.004中
- bin.004的Previous-GTIDs=1-120，如果$A=88 包含在Previous-GTIDs中,那么继续对比上一个binlog文件 bin.003,然后再循环前面2个步骤，直到找到为止

7、GTID 相关参数

参数	comment
gtid_executed	执行过的所有GTID
gtid_purged	丢弃掉的GTID
gtid_mode	gtid模式
gtid_next	session级别的变量，下一个gtid
gtid_owned	正在运行的gtid
enforce_gtid_consistency	保证GTID安全的参数

8、GTID 开启必备的条件

1、MySQL 5.6

```
gtid_mode=ON(必选)
log_bin=ON(必选)
log-slave-updates=ON(必选)
enforce-gtid-consistency(必选)
```

2、MySQL 5.7

- MySQL5.7.13 or higher

```
gtid_mode=ON(必选)
enforce-gtid-consistency（必选）
log_bin=ON（可选）--高可用切换，最好设置ON
log-slave-updates=ON（可选）--高可用切换，最好设置ON
```

9、 GTID 开启时需要注意的问题

- slave不能执行任何sql,包括超级用户;
- read_only=on, slave必须要开启这个，避免业务执行sql;
- 保证当前slave的事务id为1;
- 当slave同步出现问题时，手动跳过，需要考虑的问题
- 执行的sql，不能记录事务id，否则slave切换为master时，会导致从同步失败，因为binglog早已删除。
- SET @MYSQLDUMP_TEMP_LOG_BIN = @@SESSION.SQL_LOG_BIN;
- SET @@SESSION.SQL_LOG_BIN= 0;

10、MySQL 配置基于GTID的主从复制

- 传统的基于binlog position复制的方式有个严重的缺点：如果slave连接master时指定的binlog文件错误或者position错误，会造成遗漏或者重复， 很多时候前后数据是有依赖性的，这样就会出错而导致数据不一致。

1、环境

角色	IP地址	Server_id	数据库状态	操作系统
mysql-master	172.16.60.211	1	全新安装mysql 5.7	CentOS7.5
mysql-slave	192.168.152.*	2	全新安装mysql 5.7	CentOS7.5

角色	IP地址	Server_id	数据库状态	操作系统
mysql-slave	192.168.152.%	2	全新安装mysql 5.7	CentOS7.5

2、系统配置

```
[root@mysql-master ~]# cat /etc/redhat-release
CentOS Linux release 7.5.1804 (Core)

# 为了方便实验，关闭所有节点的防火墙
[root@mysql-master ~]# systemctl stop firewalld

[root@mysql-master ~]# cat /etc/sysconfig/selinux |grep "SELINUX=disabled"
SELINUX=disabled
[root@mysql-master ~]# setenforce 0
setenforce: SELinux is disabled
[root@mysql-master ~]# getenforce
Disabled
```

3、安装 Mysql

1、MySQL yum包下载

```
[root@master.qfedu.com ~]# wget http://repo.mysql.com/mysql57-community-release-el7-10.noarch.rpm
```

2、MySQL 软件源安转

```
[root@master.qfedu.com ~]# yum -y install mysql57-community-release-el7-10.noarch.rpm
```

3、MySQL 服务安装

```
[root@master.qfedu.com ~]# yum install  -y  mysql-community-server mysql
```

4、MySQL 服务启动

```
[root@master.qfedu.com ~]# systemctl start mysqld.service
```

5、MySQL 服务运行状态检查

```
[root@master.qfedu.com ~]# systemctl status mysqld.service
```

6、MySQL 修改临时密码

1、获取MySQL的临时密码

- 为了加强安全性，MySQL5.7为root用户随机生成了一个密码，在error log中，关于error log的位置，如果安装的是RPM包，则默认是/var/log/mysqld.log。
- 只有启动过一次mysql才可以查看临时密码

```
[root@master.qfedu.com ~]# grep 'temporary password' /var/log/mysqld.log
```

2、登陆并修改密码

1、使用默认的密码登陆

```
[root@master.qfedu.com ~]# mysql -uroot -p
```

2、修改密码

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'Qfedu.123com';
```

3、解决不符合密码复杂性要求

```
ERROR 1819 (HY000): Your password does not satisfy the current policy
requirements
```

1、修改 密码策略

```
mysql> set global validate_password_policy=0;
```

2、修改密码的长度

```
mysql> set global validate_password_length=1;
```

3、执行修改密码成功

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY 'root123';
```

4、MySQL 授权远程主机登录

```
mysql> GRANT ALL PRIVILEGES ON *.* TO 'root'@'%' IDENTIFIED BY 'mypassword' WITH
GRANT OPTION;
mysql> FLUSH  PRIVILEGES;
```

5、主数据库上的操作

1、 在my.cnf文件中配置GTID主从复制

```
[root@mysql-master ~]# cp /etc/my.cnf /etc/my.cnf.bak
[root@mysql-master ~]# >/etc/my.cnf
[root@mysql-master ~]# cat /etc/my.cnf
[mysqld]
datadir = /var/lib/mysql
socket = /var/lib/mysql/mysql.sock

symbolic-links = 0

log-error = /var/log/mysqld.log
pid-file = /var/run/mysqld/mysqld.pid

#GTID:
server_id = 1
gtid_mode = on
enforce_gtid_consistency = on

#binlog
log_bin = mysql-bin
log-slave-updates = 1
```

```
binlog_format = row
sync-master-info = 1
sync_binlog = 1

#relay log
skip_slave_start = 1
```

2、重启mysql服务

```
[root@mysql-master ~]# systemctl restart mysqld
```

3、登录mysql并查看master状态

- 发现多了一项"Executed_Gtid_Set "

```
[root@mysql-master ~]# mysql -uroot -p'Qfedu.123com'
mysql> show master status;
+------------------+----------+--------------+------------------+-------------------
-----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
+------------------+----------+--------------+------------------+-------------------
-----+
| mysql-bin.000001 |      154 |              |                  |
    |
+------------------+----------+--------------+------------------+-------------------
-----+
1 row in set (0.00 sec)

mysql> show global variables like '%uuid%';
+---------------+--------------------------------------+
| Variable_name | Value                                |
+---------------+--------------------------------------+
| server_uuid   | 317e2aad-1565-11e9-9c2e-005056ac6820 |
+---------------+--------------------------------------+
1 row in set (0.00 sec)
```

4、查看确认 gtid 功能打开

```
mysql> show global 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_owned                      |       |
| gtid_purged                     |       |
| session_track_gtids             | OFF   |
+---------------------------------+-------+
8 rows in set (0.00 sec)
```

5、查看确认binlog日志功能打开

```
mysql> show variables like 'log_bin';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| log_bin       | ON    |
+---------------+-------+
1 row in set (0.00 sec)
```

6、授权slave复制用户并刷新权限

```
mysql> grant replication slave,replication client on *.* to
slave@'192.168.152.%' identified by "Qfedu.123com";
Query OK, 0 rows affected, 1 warning (0.03 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.04 sec)

mysql> show grants for slave@'192.168.152.%';
+-----------------------------------------------------------------------------
+
| Grants for slave@192.168.152.%
|
+-----------------------------------------------------------------------------
+
| GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'192.168.152.%'
|
+-----------------------------------------------------------------------------
+
1 row in set (0.00 sec)
```

7、再次查看master状态

```
mysql> show master status;
+------------------+----------+--------------+------------------+--------------
---------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set                         |
+------------------+----------+--------------+------------------+--------------
---------------------------+
| mysql-bin.000001 |      622 |              |                  | 317e2aad-1565-
11e9-9c2e-005056ac6820:1-2 |
+------------------+----------+--------------+------------------+--------------
---------------------------+
1 row in set (0.00 sec)
```

- 启动配置之前，同样需要对从服务器进行初始化。对从服务器初始化的方法基本和基于日志点是相同的，只不过在启动了GTID模式后，在备份中所记录的就不是备份时的二进制日志文件名和偏移量了，而是记录的是备份时最后的GTID值。
- 需要先在主数据库机器上把目标库备份一下，假设这里目标库是qfedu（为了测试效果，下面手动创建）

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
```

```
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> CREATE DATABASE qfedu CHARACTER SET utf8 COLLATE utf8_general_ci;
Query OK, 1 row affected (0.02 sec)

mysql> use qfedu;
Database changed
mysql> create table if not exists demo (id int(10) PRIMARY KEY
AUTO_INCREMENT,name varchar(50) NOT NULL);
Query OK, 0 rows affected (0.27 sec)

mysql> insert into qfedu.demo values(1,"congcong"),(2,"huihui"),(3,"grace");
Query OK, 3 rows affected (0.06 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from qfedu.demo;
+----+----------+
| id | name     |
+----+----------+
|  1 | congcong |
|  2 | huihui   |
|  3 | grace    |
+----+----------+
3 rows in set (0.00 sec)
```

9、备份qfedu库

```
[root@mysql-master ~]# mysqldump --single-transaction --master-data=2 --triggers
--routines --databases qfedu -uroot -p'Qfedu.123com' > /root/qfedu.sql
```

- mysql5.6使用mysqldump备份时，指定备份的具体库，使用--database
- mysql5.7使用mysqldump备份时，指定备份的具体库，使用--databases

```
[root@mysql-master ~]# ls /root/qfedu.sql
/root/qfedu.sql
[root@mysql-master ~]# cat /root/qfedu.sql
-- MySQL dump 10.13  Distrib 5.7.24, for Linux (x86_64)
--
-- Host: localhost    Database: qfedu
-- ------------------------------------------------------
-- Server version       5.7.24-log
..............
..............
--
-- GTID state at the beginning of the backup
--

SET @@GLOBAL.GTID_PURGED='317e2aad-1565-11e9-9c2e-005056ac6820:1-5';
```

10、同步备份到从库

把备份的/root/qfedu.sql文件拷贝到mysql-slave从数据库服务器上

```
[root@mysql-master ~]# rsync -e "ssh -p22" -avpgolr /root/qfedu.sql
root@192.168.152.166:/root/
```

11、从数据库上的操作

- 在my.cnf文件中配置GTID主从复制
- 与主服务器配置除了server_id不一致外，从服务器还可以在配置文件里面添加："read_only = on"，
- 使从服务器只能进行读取操作，此参数对超级用户无效，并且不会影响从服务器的复制；

```
[root@mysql-slave ~]# cp /etc/my.cnf /etc/my.cnf.bak
[root@mysql-slave ~]# >/etc/my.cnf
[root@mysql-slave ~]# vim /etc/my.cnf
[mysqld]
datadir = /var/lib/mysql
socket = /var/lib/mysql/mysql.sock

symbolic-links = 0

log-error = /var/log/mysqld.log
pid-file = /var/run/mysqld/mysqld.pid

#GTID:
server_id = 2
gtid_mode = on
enforce_gtid_consistency = on

#binlog
log_bin = mysql-bin
log-slave-updates = 1
binlog_format = row
sync-master-info = 1
sync_binlog = 1

#relay log
skip_slave_start = 1
read_only = on
```

12、重启mysql服务

```
[root@mysql-slave ~]# systemctl restart mysqld
```

13、从库导入数据

接着将主数据库目标库的备份数据qfedu.sql导入到从数据库里

```
[root@mysql-slave ~]# ls /root/qfedu.sql
/root/qfedu.sql
[root@mysql-slave ~]# mysql -p'Qfedu.123com'
.........
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
```

```
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
4 rows in set (0.00 sec)

mysql> source /root/qfedu.sql;

mysql> select * from qfedu.demo;
+----+----------+
| id | name     |
+----+----------+
|  1 | congcong |
|  2 | huihui   |
|  3 | grace    |
+----+----------+
3 rows in set (0.00 sec)
```

14、从库配置同步

在从数据库里，使用change master 配置主从复制

```
mysql> stop slave;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> change master to
master_host='172.16.60.211',master_user='slave',master_password='Qfedu.123com',m
aster_auto_position=1;
Query OK, 0 rows affected, 2 warnings (0.26 sec)

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

mysql> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 172.16.60.211
                  Master_User: slave
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000001
          Read_Master_Log_Pos: 1357
               Relay_Log_File: mysql-slave1-relay-bin.000002
                Relay_Log_Pos: 417
        Relay_Master_Log_File: mysql-bin.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
..............
..............
            Executed_Gtid_Set: 317e2aad-1565-11e9-9c2e-005056ac6820:1-5
                Auto_Position: 1
```

- 由IO/SQL线程为 yes可知，mysql-slave节点已经和 mysql-master节点配置了主从同步关系

15、检测主从同步

- mysql-master主数据库上进行状态查看和测试测试插入

```
mysql> show master status;
+------------------+----------+--------------+------------------+-------------
--------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set                        |
+------------------+----------+--------------+------------------+-------------
--------------------------+
| mysql-bin.000001 |     1357 |              |                  | 317e2aad-1565-
11e9-9c2e-005056ac6820:1-5 |
+------------------+----------+--------------+------------------+-------------
--------------------------+
1 row in set (0.00 sec)

mysql> show slave hosts;
+-----------+------+------+-----------+--------------------------------------+
| Server_id | Host | Port | Master_id | Slave_UUID                           |
+-----------+------+------+-----------+--------------------------------------+
|         2 |      | 3306 |         1 | 2c1efc46-1565-11e9-ab8e-00505688047c |
+-----------+------+------+-----------+--------------------------------------+
1 row in set (0.00 sec)

mysql> insert into qfedu.demo values(4,"beijing"),(5,"hefei"),(10,"xihu");
Query OK, 3 rows affected (0.06 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> delete from qfedu.demo where id<4;
Query OK, 3 rows affected (0.10 sec)

mysql> select * from qfedu.demo;
+----+---------+
| id | name    |
+----+---------+
|  4 | beijing |
|  5 | hefei   |
| 10 | xihu    |
+----+---------+
3 rows in set (0.00 sec)
```

- mysql-slave从数据库上查看

```
mysql> select * from qfedu.demo;
+----+---------+
| id | name    |
+----+---------+
|  4 | beijing |
|  5 | hefei   |
| 10 | xihu    |
+----+---------+
3 rows in set (0.00 sec)
```

- 发现 mysql-slave从数据库已经将新插入的数据同步完成