MySQL权限控制及日志管理

一、MySQL 的用户管理和权限管理

1、DCL（Data Control Language 数据库控制语言）

- 数据库授权、角色控制等操作
- GRANT 用户授权，为用户赋予访问权限
- REVOKE 取消授权，撤回授权权限

2、MySQL 权限表

1、mysql.user

- 用户字段：Host、User、Password
- 权限字段：_Priv结尾的字段
- 安全字段：ssl x509字段
- 资源控制字段：max_开头的字段

2、mysql.db

- 用户字段：Host、User、Password
- 权限字段：剩下的_Priv结尾的字段

3、mysql.tables_priv，mysql.columns_priv、procs_priv

- 表、列、存储过程的授权表

4、授权级别排列

- mysql.user #全局授权
- mysql.db #数据库级别授权
- 其他 #表级，列级授权

5、数据库和表格式

```
数据库名.*                 数据库中的所有
数据库名.表名              指定数据库中的某张表
数据库名.存储过程          指定数据库中的存储过程
*.*                        所有数据库
```

6、用户和 IP格式

```
用户名@IP地址              用户只能在改IP下才能访问
用户名@192.168.1.%         用户只能在改IP段下才能访问(通配符%表示任意)
用户名@%.qfedu.com
用户名@%                   用户可以再任意IP下访问(默认IP地址为%)
```

3、MySQL 用户管理

1、创建用户

1、CREATE USER语句创建

```
CREATE USER '用户名'@'IP地址' [ IDENTIFIED BY '密码' ];
```

2、GRANT语句创建

```
GRANT SELECT ON *.* TO '用户名'@’IP地址’  IDENTIFIED BY "密码";
```

3、创建实例

```
CREATE USER 'qfedu'@'localhost' IDENTIFIED BY '123456';
CREATE USER 'qfedu'@'192.168.1.101' IDENDIFIED BY '123456';
CREATE USER 'qfedu'@'192.168.1.%' IDENDIFIED BY '123456';
CREATE USER 'qfedu'@'%' IDENTIFIED BY '123456';
GRANT ALL ON *.* TO 'user3'@’localhost’ IDENTIFIED BY ‘123456’;
```

2、删除用户

1、DROP USER 删除

```
DROP USER '用户名'@'IP地址';
```

- 实例

```
DROP USER 'user1'@’localhost’;
```

2、DELETE 语句删除

```
DELETE FROM mysql.user WHERE user='用户名' AND host='IP地址'
```

- 实例

```
DELETE FROM mysql.user WHERE user=’user2’ AND host=’localhost’;
```

3、修改用户

```
RENAME USER '旧用户名'@'IP地址' TO '新用户名'@'IP地址' ;
```

实例

```
RENAME USER 'old_user'@‘localhost’ TO 'new_user'@'localhost';
```

4、修改密码

注意： 修改完密码必须刷新权限

```
FLUSH PRIVILEGES;
```

1、root 用户修改自己密码

```
方法一：
    mysqladmin -uroot -p123 password 'new_password'    # 123为旧密码

方法二：
    alter user 'root'@'localhost' identified by 'new_pssword';

方法三：
    SET PASSWORD=password(‘new_password’);
```

2、root 修改其他用户密码

```
方法一：
    alter user 'qfedu'@'localhost' identified by 'Qfedu.1234com';

方法三：
    GRANT SELECT ON *.* TO 用户名@’ip地址’ IDENTIFIED BY ‘yuan’;
```

3、普通用户修改自己密码

```
SET password=password(‘new_password’);
```

5、找回 root 密码

1、修改 MySQL 配置文件

- 在[mysqld]下面加上 skip-grant-tables

```
[root@qfedu.com ~]# vim /etc/my.cnf
[mysqld]
...
#设置免密登录
skip-grant-tables
```

2、重启 MySQL

```
[root@qfedu.com ~]# systemctl restart mysqld
```

3、修改密码

1、终端输入 mysql 直接登录 MySQL数据库

```
[root@qfedu.com ~]# mysql
```

2、切换到 MySQL 系统库 mysql

```
mysql> use mysql;
```

3、设置密码

```
mysql> update user set authentication_string=password('密码') where user='root';
```

4、注释掉免密登录

```
[root@qfedu.com ~]# vim /etc/my.cnf
[mysqld]
...
#设置免密登录
#skip-grant-tables
```

5、重启 MySQL 然后登录

```
[root@qfedu.com ~]# systemctl restart mysqld
[root@qfedu.com ~]# mysql -uroot -p
```

6、密码复杂度

1、安装密码插件

- MySQL 默认启用了密码复杂度设置，插件名字叫做 validate_password

```
mysql> INSTALL PLUGIN validate_password SONAME 'validate_password.so';
```

2、修改配置文件

1、修改配置

```
[root@qfedu.com ~]#  vim /etc/my.cnf
[mysqld]
plugin-load=validate_password.so
validate_password_policy=0
validate-password=FORCE_PLUS_PERMANENT
```

2、重启MySQL生效

```
[root@qfedu.com ~]# systemctl restart mysqld
```

2、查看错误日志

3、登陆数据库查看

```
mysql>  show variables like 'validate%';
+--------------------------------------+--------+
| Variable_name                        | Value  |
+--------------------------------------+--------+
| validate_password_check_user_name    | OFF    |
| validate_password_dictionary_file    |        |
| validate_password_length             | 8      |
| validate_password_mixed_case_count   | 1      |
| validate_password_number_count       | 1      |
| validate_password_policy             | MEDIUM |
| validate_password_special_char_count | 1      |
+--------------------------------------+--------+
7 rows in set (0.04 sec)
```

1、validate_password_policy

- 代表的密码策略，可配置的值有以下：默认是MEDIUM
- 0 or LOW 仅需需符合密码长度（由参数validate_password_length指定）

- 1 or MEDIUM 满足LOW策略，同时还需满足至少有1个数字，小写字母，大写字母和特殊字符
- 2 or STRONG 满足MEDIUM策略，同时密码不能存在字典文件（dictionary file）中

2、validate_password_dictionary_file

- 用于配置密码的字典文件，当validate_password_policy设置为STRONG时可以配置密码字典文件，字典文件中存在的密码不得使用。

3、validate_password_length

- 用来设置密码的最小长度，默认值是8最小是0

4、validate_password_mixed_case_count

- 当validate_password_policy设置为MEDIUM或者STRONG时，密码中至少同时拥有的小写和大写字母的数量，默认是1最小是0；默认是至少拥有一个小写和一个大写字母。

5、validate_password_number_count

- 当validate_password_policy设置为MEDIUM或者STRONG时，密码中至少拥有的数字的个数，默认1最小是0

6、validate_password_special_char_count

- 当validate_password_policy设置为MEDIUM或者STRONG时，密码中至少拥有的特殊字符的个数，默认1最小是0

7、密码不符合复杂性

```
mysql>  GRANT ALL ON *.* TO admin1@'%' IDENTIFIED BY '123';
ERROR 1819 (HY000): Your password does not satisfy the current policy
requirements
```

- 处理方法：

1、查看密码策略

```
mysql> select @@validate_password_policy;           # 查看密码复杂性策略
mysql> select @@validate_password_length;           # 查看密码复杂性要求密码最低长度大
小
```

2、更换符合复杂性要求的密码

```
mysql> set global validate_password_length=1;       # 设置密码复杂性要求密码最低长度为
1
```

3、关闭复杂性策略

```
mysql> set global validate_password_policy=0;       # 关闭密码复杂性策略
```

4、MySQL 登录

```
mysql -u用户名 -p密码 [ -h主机 ] [ -P端口 ];
mysql -u用户名 -p密码 [ -h主机 ] [ -P端口 ] [ -e"SQL语句" ];
```

- 实例

```
[root@qfedu.com ~]# mysql -uroot -hlocalhost -p'qfedu.123com' -P3306
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 8
Server version: 5.7.29 MySQL Community Server (GPL)

Copyright (c) 2000, 2020, 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>
```

5、MySQL 权限管理

1、查看权限

```
SHOW GRANTS FOR  '用户'@'IP地址';
```

- 可以不指定用户，则显示当前用户权限

实例

```
mysql> SHOW GRANTS FOR 'root'@'localhost';
```

2、授权及设置密码

1、授权及设置密码

```
GRANT 权限 [,权限...[,权限]] NO 数据库.数据表 TO '用户'@'IP地址';
GRANT 权限 [,权限...[,权限]] NO 数据库.数据表 TO '用户'@'IP地址' IDENTIFIED BY '密码';
```

- 可以简化多次授权，并用逗号分隔
- GRANT需要明确一下信息
  - 要授予的权限
  - 被授予权限的数据库或表
  - 用户名

2、权限简介

权限	权限级别	权限说明
CREATE	数据库、表或索引	创建数据库、表或索引权限
DROP	数据库或表	删除数据库或表权限
GRANT OPTION	数据库、表或保存的程序	赋予权限选项
REFERENCES	数据库或表	
ALTER	表	更改表，比如添加字段、索引等
DELETE	表	删除数据权限
INDEX	表	索引权限
INSERT	表	插入权限
SELECT	表	查询权限
UPDATE	表	更新权限
CREATE VIEW	视图	创建视图权限
SHOW VIEW	视图	查看视图权限
ALTER ROUTINE	存储过程	更改存储过程权限
CREATE ROUTINE	存储过程	创建存储过程权限
EXECUTE	存储过程	执行存储过程权限
FILE	服务器主机上的文件访问	文件访问权限
CREATE TEMPORARY TABLE	服务器管理	创建临时表权限
LOCK TABLES	服务器管理	锁表权限
CREATE USER	服务器管理	创建用户权限
RELOAD	服务器管理	执行flush-hosts, flush-logs, flush-privileges, flush-status, flush-tables, flush-threads, refresh, reload等命令的权限
PROCESS	服务器管理	查看进程权限
REPLICATION CLIENT	服务器管理	复制权限

权限	权限级别	权限说明
REPLICATION SLAVE	服务器管理	复制权限
SHOW DATABASES	服务器管理	查看数据库权限
SHUTDOWN	服务器管理	关闭数据库权限
SUPER	服务器管理	执行kill线程权限

权限分布	可能的设置的权限
表权限	'Select', 'Insert', 'Update', 'Delete', 'Create', 'Drop', 'Grant', 'References', 'Index', 'Alter'
列权限	'Select', 'Insert', 'Update', 'References'
过程权限	'Execute', 'Alter Routine', 'Grant'

3、授权设置密码实例

```
mysql> GRANT ALL ON *.* TO admin1@'%' IDENTIFIED BY '(Qfedu.123com)';
mysql> GRANT ALL ON *.* TO admin2@'%' IDENTIFIED BY '(Qfedu.123com)' WITH GRANT
OPTION;
mysql> GRANT ALL ON qfedu.* TO admin3@'%' IDENTIFIED BY '(Qfedu.123com)';
mysql> GRANT ALL ON qfedu.* TO admin3@'192.168.122.220' IDENTIFIED BY
'(Qfedu.123com)';
mysql> GRANT ALL ON qfedu.user TO admin4@'%' IDENTIFIED BY '(Qfedu.123com)';
mysql> GRANT SELECT(col1),INSERT(col2,col3) ON qfedu.user TO admin5@'%'
IDENTIFIED BY '(Qfedu.123com)';
```

4、MySQL 8.x 授权方式

1、创建新的用户

```
CREATE USER '用户名'@'localhost' IDENTIFIED BY '密码';
```

2、数据库授权

```
GRANT ALL PRIVILEGES ON 数据库名.* TO '用户名'@'IP地址';
```

3、回收权限

```
REVOKE 权限 ON 数据库.数据表 FROM '用户'@'IP地址';
```

- 被回收的权限必须存在，否则会出错
- 整个服务器，使用 GRANT ALL 和 REVOKE ALL；
- 整个数据库，使用 ON datebase.*；
- 特定的表：使用 ON datebase.table；
- 实例

```
mysql> REVOKE DELETE ON *.*  FROM admin1@'%';
   # 回收指定权限
mysql> REVOKE ALL PRIVILEGES  ON *.*  FROM admin2@'%';
   # 回收所有权限
mysql> REVOKE ALL PRIVILEGES,GRANT OPTION  ON *.* FROM 'admin2'@'%';
# 回收多个指定权限
```

4、刷新权限

```
mysql> FLUSH PRIVILEGES;
```

- flush privileges 命令本质上的作用是将当前user和privilige表中的用户信息/权限设置从mysql库(MySQL数据库的内置库)中提取到内存里。
- MySQL用户数据和权限有修改后，搜索希望在"不重启MySQL服务"的情况下直接生效，那么就需要执行这个命令

二、MySQL 日志管理

1、MySQL 错误日志

1、错误日志作用

- 记录MySQL启动及工作过程中，状态、报错、警告。

2、设置错误日志

1、修改配置文件，并重启MySQL

1、配置错误日志

```
[root@qfedu.com ~]# vim /etc/my.cnf
log_error=/data/3306/data/mysql.log   #这里的路径和文件名称可以随便定义
```

2、重启MySQL生效

```
[root@qfedu.com ~]# systemctl restart mysqld
```

2、查看错误日志

```
mysql> select @@log_error;
+---------------------------+
| @@log_error               |
+---------------------------+
| /data/3306/data/mysql.log |
+---------------------------+
1 row in set (0.00 sec)
```

注意： 查看错误日志关注[ERROR]的上下文.

2、MySQL 二进制日志

1、二进制日志作用

- 数据恢复必备的日志。

- 主从复制依赖的日志。

2、MySQL 二进制日志设置

1、修改配置文件

```
[root@qfedu.com ~]# vim /etc/my.cnf
server_id=6
log_bin=/data/3306/binlog/mysql-bin
```

- 配置说明
  - server_id 是5.7之后开二进制日志必加的参数
  - log_bin= 打开二进制功能
  - /data/3306/binlog/ 指定存放路径
  - mysql-bin 文件名前缀

2、创建目录并授权

```
[root@qfedu.com 3306]# mkdir -p /data/3306/binlog/
[root@qfedu.com 3306]# chown -R mysql.mysql /data/3306/*
```

3、重启数据库

```
[root@qfedu.com 3306]# systemctl restart mysqld

[root@qfedu.com binlog]# ll binlog/
total 8
-rw-r----- 1 mysql mysql 768 Aug 14 20:02 mysql-bin.000001
-rw-r----- 1 mysql mysql  35 Aug 14 18:18 mysql-bin.index
```

- 配置说明
  - mysql-bin 是在配置文件配置的前缀
  - 000001 MySQL每次重启，重新生成新的

3、二进制日志内容

- 除了查询类的语句，都会记录，即**所有数据库变更类的语句。**

1、记录语句的种类

- DDL（数据定义语言）：create、drop
- DCL（数据控制语言）：grant 、revoke
- DML（数据操作语言）：insert、update、delete

2、不同语句的记录格式说明

- DDL、DCL直接以语句（statement）方式记录 .
- DML 语句有三种模式：SBR、RBR、MBR

```
mysql> select @@binlog_format;
+-----------------+
| @@binlog_format |
+-----------------+
| ROW             |
+-----------------+
1 row in set (0.00 sec)
```

- 配置说明
  - statement---->SBR：做什么记录什么，即SQL语句
  - row---------->RBR：记录数据行的变化（默认模式，推荐）
  - mixed-------->MBR：自动判断记录模式
- SBR和RBR的区别

区别项	SBR	RBR（默认、推荐）
记录内容	SQL语句	记录数据行的变化
可读性	较强	差
日志量	小	大
日志记录准确性	数据误差	没有误差

4、二进制日志工作模式

1、配置二进制日志工作模式

1、修改配置文件

```
[root@qfedu.com ~]# vim /etc/my.cnf
[mysqld]
binlog_format='ROW'
```

2、重启数据库

```
[root@qfedu.com ~]# systemctl restart mysqld
```

2、查看二进制日志工作模式

```
mysql> show  variables like "binlog%";
+-----------------------------------------+--------------+
| Variable_name                           | Value        |
+-----------------------------------------+--------------+
| binlog_cache_size                       | 32768        |
| binlog_checksum                         | CRC32        |
| binlog_direct_non_transactional_updates | OFF          |
| binlog_error_action                     | ABORT_SERVER |
| binlog_format                           | ROW          |
| binlog_group_commit_sync_delay          | 0            |
| binlog_group_commit_sync_no_delay_count | 0            |
| binlog_gtid_simple_recovery             | ON           |
| binlog_max_flush_queue_time             | 0            |
| binlog_order_commits                    | ON           |
| binlog_row_image                        | FULL         |
```

```
| binlog_rows_query_log_events                | OFF          |
| binlog_stmt_cache_size                      | 32768        |
| binlog_transaction_dependency_history_size | 25000        |
| binlog_transaction_dependency_tracking      | COMMIT_ORDER |
+---------------------------------------------+--------------+
15 rows in set (0.00 sec)
```

3、二进制日志三种模式的区别

1、ROW: 基于行的复制

优点：所有的语句都可以复制，不记录执行的sql语句的上下文相关的信息，仅需要记录那一条记录被修改成什么了

缺点：binlog 大了很多，复杂的回滚时 binlog 中会包含大量的数据；主服务器上执行update语句时，所有发生变化的记录都会写到 binlog 中；比如有这样一条update语句：update product set owner_member_id='d' where owner_member_id='a',执行之后，日志中记录的不是这条update语句所对应的事件(mysql是以事件的形式来记录bin-log日志)，而是这条语句所更新的每一条记录的变化情况，这样就记录成很多条记录被更新的很多事件。自然，bin-log日志的量会很大。

2、Statement: 基于sql语句的复制

优点：不需要记录每一行的变化，减少了binlog日志量，节约了IO，提高性能

缺点：由于它是记录的执行语句，所以为了让这些语句在slave端也能正确执行，那么他还必须记录每条语句在执行的时候的一些相关信息，也就是上下文信息，以保证所有语句在slave端被执行的时候能够得到和在master端执行时候相同的结果。另外就是,由于mysql现在发展比较快，很多的新功能加入，使mysql的复制遇到了不小的挑战,自然复制的时候涉及到越复杂的内容，bug也就越容易出现。在statement level下，目前已经发现的就有不少情况会造成mysql的复制问题，主要是修改数据的时候使用了某些特定的函数或者功能的时候会出现，比如sleep()在有些版本就不能正确复制。

3、mixed模式： row 与 statement 结合

实际上就是前两种模式的结合，在mixed模式下，mysql会根据执行的每一条具体的sql语句来区分对待记录的日志形式，也就是在statement和row之间选一种。新版本中的statement level还是和以前一样，仅仅记录执行的语句。而新版本的mysql中对row level模式被做了优化，并不是所有的修改都会以row level来记录，像遇到表结构变更的时候就会以statement模式来记录，如果sql语句确实就是update或者delete 等修改数据的语句，那么还是会记录所有行的变更。

4、二进制日志事件

1、 二进制日志事件简介

- 二进制日志内容以事件（binlog events）为最小记录单元。
- 对于DDL和DCL，一个语句就是一个事件。
- 对于DML（标准的事务语句），只记录已提交的事务的DML语句

```
begin ;     事件1
a           事件2
b           事件3
commit;     事件4
```

2、事件的构成（为了截取日志）

```
[root@qfedu.com binlog]# mysqlbinlog mysql-bin.000001
# at 219                事件开始的位置(position)
end_log_pos 319         事件结束的位置(position)
#200219 14:28:12        事件发生的时间
create database qfedu   事件内容
```

3、二进制日志的基础查看

1、查看二进制日志的配置信息

```
mysql> show variables like '%log_bin%';
+---------------------------------+-------------------------------+
| Variable_name                   | Value                         |
+---------------------------------+-------------------------------+
| log_bin                         | ON                            |
| log_bin_basename                | /var/log/mysql/mysql-bin      |
| log_bin_index                   | /var/log/mysql/mysql-bin.index |
| log_bin_trust_function_creators | OFF                           |
| log_bin_use_v1_row_events       | OFF                           |
| sql_log_bin                     | ON                            |
+---------------------------------+-------------------------------+
6 rows in set (0.00 sec)
```

- 字段说明
  - log_bin 开启二进制日志的开关
  - log_bin_basename 位置
  - sql_log_bin 临时开启或关闭二进制日志的小开关

2、查看二进制日志的基本信息

1、打印出当前MySQL的所有二进制日志，并且显示最后使用到的 position

```
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |       316 |
+------------------+-----------+
1 row in set (0.00 sec)
```

2、查看当前正在使用的二进制日志

```
mysql> show binary logs;
+------------------+-----------+
| Log_name         | File_size |
+------------------+-----------+
| mysql-bin.000001 |       316 |
+------------------+-----------+
1 row in set (0.00 sec)

mysql> show master status;（常用）
+------------------+----------+--------------+------------------+---------------
----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
```

```
+------------------+----------+--------------+------------------+--------------
----+
| mysql-bin.000001 |      316 |              |                  |
    |
+------------------+----------+--------------+------------------+--------------
----+
1 row in set (0.00 sec)
```

3、查看二进制日志的事件信息

```
mysql> show master status;
+------------------+----------+--------------+------------------+--------------
----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
+------------------+----------+--------------+------------------+--------------
----+
| mysql-bin.000001 |      316 |              |                  |
    |
+------------------+----------+--------------+------------------+--------------
----+
1 row in set (0.00 sec)

mysql> show binlog events in 'mysql-bin.000001';
+------------------+-----+----------------+-----------+-------------+----------
---------------------------+
| Log_name         | Pos | Event_type     | Server_id | End_log_pos | Info
                           |
+------------------+-----+----------------+-----------+-------------+----------
---------------------------+
| mysql-bin.000001 |   4 | Format_desc    |         6 |         123 | Server
ver: 5.7.29-log, Binlog ver: 4 |
| mysql-bin.000001 | 123 | Previous_gtids |         6 |         154 |
                            |
| mysql-bin.000001 | 154 | Anonymous_Gtid |         6 |         219 | SET
@@SESSION.GTID_NEXT= 'ANONYMOUS'  |
| mysql-bin.000001 | 219 | Query          |         6 |         316 | create
database qfedu                  |
+------------------+-----+----------------+-----------+-------------+----------
---------------------------+
4 rows in set (0.00 sec)
```

4、二进制日志内容的查看和截取

1、内容查看命令

```
[root@qfedu.com ~]# mysqlbinlog  /data/3306/binlog/mysql-bin.000001
[root@qfedu.com ~]# mysqlbinlog --base64-output=decode-rows -vvv
/data/3306/binlog/mysql-bin.000001
```

- 记不住参数可以去mysqlbinlog --help中查看

2、日志的截取

截取语法:

- --start-position 开始截取 pod 点

- --stop-position 结束截取 pod 点

```
[root@qfedu.com ~]# mysqlbinlog --start-position=xxx  --stop-position=xxx
/data/3306/binlog/mysql-bin.000001 >/data/bin.sql
```

3、数据恢复实例

1、准备数据

```
mysql>  create database binlog charset utf8mb4;
Query OK, 1 row affected (0.00 sec)

mysql>  use binlog;
Database changed
mysql>  create table t1(id int) engine=innodb charset=utf8mb4;
Query OK, 0 rows affected (0.02 sec)

mysql>
mysql>  insert into t1 values(1),(2),(3);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql>  insert into t1 values(11),(12),(13);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

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

mysql>

mysql>  update t1 set id=10 where id>10;

Query OK, 3 rows affected (0.00 sec)
Rows matched: 3  Changed: 3  Warnings: 0

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

mysql>

mysql>  select * from t1;
+------+
| id   |
+------+
|    1 |
|    2 |
|    3 |
|   10 |
|   10 |
|   10 |
+------+
6 rows in set (0.00 sec)
```

2、删除数据

```
mysql>  drop database binlog;
Query OK, 1 row affected (0.00 sec)
```

3、 数据恢复

1、确认起点和终点

```
mysql>  show master status;
+------------------+----------+--------------+------------------+---------------
----+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set |
+------------------+----------+--------------+------------------+---------------
----+
| mysql-bin.000001 |     1610 |              |                  |
    |
+------------------+----------+--------------+------------------+---------------
----+
1 row in set (0.00 sec)


mysql>  show binlog events in 'mysql-bin.000001';

起点：
| mysql-bin.000001 |  381 | Query          |         6 |         497 | create
database binlog charset utf8mb4                                |

终点：
| mysql-bin.000001 | 1575 | Query          |         6 |        1673 | drop
database binlog
```

2、截取日志

```
[root@qfedu.com ~]# mysqlbinlog --start-position=381  --stop-position=1575
/data/3306/binlog/mysql-bin.000003>/data/bin.sql
```

3、恢复日志

```
mysql>  set sql_log_bin=0;   #  临时关闭当前会话的binlog记录
mysql>  source /data/bin.sql;
mysql>  set sql_log_bin=1;   #  打开当前会话的binlog记录
```

5、基于 gtid 的二进制日志管理

1、gtid（Global Transaction ID）简介

- 全局唯一的事务编号。
- 幂等性。
- GtID包括两部分：
  - Server_uuid：
  - Tx_id：

2、gtid 配置

1、查看 gtid

```
mysql>  show variables like '%gtid%';
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| binlog_gtid_simple_recovery     | ON        |
| enforce_gtid_consistency        | OFF       |
| gtid_executed_compression_period | 1000      |
| gtid_mode                       | OFF       |
| gtid_next                       | AUTOMATIC |
| gtid_owned                      |           |
| gtid_purged                     |           |
| session_track_gtids             | OFF       |
+---------------------------------+-----------+
8 rows in set (0.00 sec)
```

2、修改配置

```
[root@qfedu.com ~]# vim /etc/my.cnf
[mysqld]
gtid_mode=on                    # 开启 gtid
enforce_gtid_consistency=true   # 强制GTID一致性
log_slave_updates=1             # 主从复制中从库记录 binlog，并统一GTID信息
```

3、重启数据库

```
[root@qfedu.com ~]# systemcat restart mysqld
```

3、基于 gtid 截取日志

- 对于 DDL和 DCL 一个操作就是一个 GTID。
- 对于 DML，一个完整的事务就是已给 GTID。

```
mysql> show variables like '%gtid%';
+---------------------------------+-----------+
| Variable_name                   | Value     |
+---------------------------------+-----------+
| binlog_gtid_simple_recovery     | ON        |
| enforce_gtid_consistency        | ON        |
| gtid_executed_compression_period | 1000      |
| gtid_mode                       | ON        |
| gtid_next                       | AUTOMATIC |
| gtid_owned                      |           |
| gtid_purged                     |           |
| session_track_gtids             | OFF       |
+---------------------------------+-----------+
8 rows in set (0.00 sec)


mysql> show master status;
+------------------+----------+--------------+------------------+---------------
---------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set                          |
```

```
+------------------+----------+--------------+------------------+--------------
--------------------------+
| mysql-bin.000002 |     1359 |              |                  | 827ddb16-4ec8-
11ea-b734-000c293df1f0:1-5 |
+------------------+----------+--------------+------------------+--------------
--------------------------+
1 row in set (0.00 sec)

mysql>  show binlog events in 'mysql-bin.000002';
+------------------+------+---------------+-----------+-------------+----------
-------------------------------------------------------------+
| Log_name         | Pos  | Event_type    | Server_id | End_log_pos | Info
                                                             |
+------------------+------+---------------+-----------+-------------+----------
-------------------------------------------------------------+
| mysql-bin.000002 |    4 | Format_desc   |         6 |         123 | Server
ver: 5.7.29-log, Binlog ver: 4                                |
| mysql-bin.000002 |  123 | Previous_gtids |         6 |         154 |
                                                            |
| mysql-bin.000002 |  154 | Gtid          |         6 |         219 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:1'    |
| mysql-bin.000002 |  219 | Query         |         6 |         338 | create
database binlog1 charset utf8mb4                              |
| mysql-bin.000002 |  338 | Gtid          |         6 |         403 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:2'    |
| mysql-bin.000002 |  403 | Query         |         6 |         536 | use
`binlog1`; create table t1(id int) engine=innodb charset=utf8mb4 |
| mysql-bin.000002 |  536 | Gtid          |         6 |         601 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:3'    |
| mysql-bin.000002 |  601 | Query         |         6 |         676 | BEGIN
                                                            |
| mysql-bin.000002 |  676 | Table_map     |         6 |         724 | table_id:
108 (binlog1.t1)                                            |
| mysql-bin.000002 |  724 | Write_rows    |         6 |         774 | table_id:
108 flags: STMT_END_F                                       |
| mysql-bin.000002 |  774 | Xid           |         6 |         805 | COMMIT /*
xid=11 */                                                   |
| mysql-bin.000002 |  805 | Gtid          |         6 |         870 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:4'    |
| mysql-bin.000002 |  870 | Query         |         6 |         945 | BEGIN
                                                            |
| mysql-bin.000002 |  945 | Table_map     |         6 |         993 | table_id:
108 (binlog1.t1)                                            |
| mysql-bin.000002 |  993 | Write_rows    |         6 |        1043 | table_id:
108 flags: STMT_END_F                                       |
| mysql-bin.000002 | 1043 | Xid           |         6 |        1074 | COMMIT /*
xid=12 */                                                   |
| mysql-bin.000002 | 1074 | Gtid          |         6 |        1139 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:5'    |
| mysql-bin.000002 | 1139 | Query         |         6 |        1214 | BEGIN
                                                            |
| mysql-bin.000002 | 1214 | Table_map     |         6 |        1262 | table_id:
108 (binlog1.t1)                                            |
| mysql-bin.000002 | 1262 | Update_rows   |         6 |        1328 | table_id:
108 flags: STMT_END_F                                       |
| mysql-bin.000002 | 1328 | Xid           |         6 |        1359 | COMMIT /*
xid=15 */                                                   |
```

```
| mysql-bin.000002 | 1359 | Gtid           |         6 |        1424 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:6'    |
+------------------+------+----------------+-----------+-------------+----------
-------------------------------------------------------------+
37 rows in set (0.00 sec)
```

1、基于 gtid 截取日志

- --include-gtids= 包含
- --exclude-gtids= 排除
- --skip-gtids= 跳过
- 截取1-3号事务

```
[root@qfedu.com ~]# mysqlbinlog --include-gtids=' 827ddb16-4ec8-11ea-b734-
000c293df1f0:1-3' /data/binlog/mysql-bin.000001>/data/gtid.sql
```

- 截取 1-10 gtid事务,跳过6号和8号事务

```
[root@qfedu.com ~]# mysqlbinlog --include-gtids=' 827ddb16-4ec8-11ea-b734-
000c293df1f0:1-10 --exclude-gtids='545fd699-be48-11e9-8f0a-
000c2980e248:6,545fd699-be48-11e9-8f0a-000c2980e248:8'    /data/binlog/mysql-
bin.000001>/data/gtid.sql
```

2、gtid 截取日志实例

1、准备环境

```
mysql>  create database gtid charset utf8mb4;
Query OK, 1 row affected (0.00 sec)

mysql>  use gtid;
Database changed

mysql>  create table t1(id int) engine=innodb charset=utf8mb4;
Query OK, 0 rows affected (0.02 sec)

mysql>  insert into t1 values(1),(2),(3);
Query OK, 3 rows affected (0.06 sec)
Records: 3  Duplicates: 0  Warnings: 0

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

mysql>  insert into t1 values(11),(12),(13);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

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

mysql>  select * from t1;
+------+
| id   |
+------+
|    1 |
|    2 |
```

```
|    3 |
|   11 |
|   12 |
|   13 |
+------+
6 rows in set (0.00 sec)
```

2、删除数据

```
mysql>  drop database gtid;
Query OK, 1 row affected (0.01 sec)

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

3、找起点和终端(gtid)

```
mysql>  show master status;
+------------------+----------+--------------+------------------+---------------
-----------------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
Executed_Gtid_Set                          |
+------------------+----------+--------------+------------------+---------------
-----------------------------+
| mysql-bin.000002 |     2409 |              |                  | 827ddb16-4ec8-
11ea-b734-000c293df1f0:1-10 |
+------------------+----------+--------------+------------------+---------------
-----------------------------+
1 row in set (0.00 sec)


mysql>  show binlog events in 'mysql-bin.000002';
| mysql-bin.000002 | 1359 | Gtid           |         6 |        1424 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:6'    |
| mysql-bin.000002 | 1424 | Query          |         6 |        1534 | create
database gtid charset utf8mb4                                  |

| mysql-bin.000002 | 2252 | Gtid           |         6 |        2317 | SET
@@SESSION.GTID_NEXT= '827ddb16-4ec8-11ea-b734-000c293df1f0:10'   |
| mysql-bin.000002 | 2317 | Query          |         6 |        2409 | drop
database gtid                                                   |
```

4、截取日志（仅供参考）

```
[root@qfedu.com ~]# mysqlbinlog --skip-gtids --include-gtids='827ddb16-4ec8-
11ea-b734-000c293df1f0:6-9' /data/3306/binlog/mysql-bin.000002 > /data/gtid.sql
```

5、恢复数据

```
mysql>  set sql_log_bin=0;
Query OK, 0 rows affected (0.00 sec)

mysql>  source /data/gtid.sql

mysql>  set sql_log_bin=1;

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

6、二进制日志其他操作

1、自动清理日志

1、查看自动清理周期

```
mysql> show variables like '%expire%';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| disconnect_on_expired_password | ON    |
| expire_logs_days               | 0     |
+--------------------------------+-------+
2 rows in set (0.00 sec)
```

2、零时设置自动清理周期

```
mysql> set global expire_logs_days=8;
mysql> show variables like '%expire%';
+--------------------------------+-------+
| Variable_name                  | Value |
+--------------------------------+-------+
| disconnect_on_expired_password | ON    |
| expire_logs_days               | 8     |
+--------------------------------+-------+
2 rows in set (0.00 sec)
```

3、永久生效

1、修改配置文件

```
[root@qfedu.com ~]# vim /etc/my.cnf
[mysqld]
expire_logs_days=15;
```

- 企业建议,至少保留两个全备周期+1的binlog

2、重启数据库

```
[root@qfedu.com ~]# systemcat restart mysqld
```

2、手工清理

```
PURGE BINARY LOGS BEFORE now() - INTERVAL 3 day;
PURGE BINARY LOGS TO 'mysql-bin.000009';
```

- 注意: 不要手工 rm binlog文件
- 主从关系中，主库执行此操作 reset master;，主从环境必崩

3、二进制日志的滚动

```
mysql> flush logs;
mysql> select @@max_binlog_size;
```

3、MySQL 慢日志

1、慢日志简介

- 记录运行较慢的语句记录slowlog中。
- 功能是辅助优化的工具日志。
- 应激性的慢可以通过show processlist进行监控
- 一段时间的慢可以进行slow记录、统计

2、慢日志配置

1、查看慢日志

1、查看是否开启

```
mysql>  show variables like '%slow_query%';
+---------------------+----------------------------------+
| Variable_name       | Value                            |
+---------------------+----------------------------------+
| slow_query_log      | OFF                              |
| slow_query_log_file | /var/lib/mysql/localhost-slow.log |
+---------------------+----------------------------------+
2 rows in set (0.00 sec)
```

- 重连或新开一个会话才能看到修改值

2、查看阈值

```
mysql> select @@long_query_time;
+-------------------+
| @@long_query_time |
+-------------------+
|         10.000000 |
+-------------------+
1 row in set (0.00 sec)
```

```
mysql> SHOW GLOBAL VARIABLES LIKE 'long_query_time%';
+-----------------+-----------+
| Variable_name   | Value     |
+-----------------+-----------+
| long_query_time | 10.000000 |
+-----------------+-----------+
1 row in set (0.00 sec)

mysql> SHOW VARIABLES LIKE 'long_query_time%';
+-----------------+-----------+
| Variable_name   | Value     |
+-----------------+-----------+
| long_query_time | 10.000000 |
+-----------------+-----------+
1 row in set (0.00 sec)


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

2、配置慢日志

1、零时设置开启

```
SET GLOBAL slow_query_log = 1;        #默认未开启，开启会影响性能，mysql重启会失效
```

2、设置阈值：

```
SET GLOBAL long_query_time=3;
```

3、永久生效

1、修改配置文件

```
[root@qfedu.com ~]# vim /etc/my.cnf
[mysqld]
slow_query_log=1
slow_query_log_file=/data/3306/data/qfedu-slow.log
long_query_time=0.1   默认配置10秒钟
log_queries_not_using_indexes=1
```

2、重启数据库

```
[root@qfedu.com ~]# systemcat restart mysqld
```

3、慢语句模拟

```
mysql> set sql_log_bin=0;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> select sleep(4);
+----------+
| sleep(4) |
+----------+
|        0 |
+----------+
1 row in set (4.00 sec)

mysql> SHOW GLOBAL STATUS LIKE '%Slow_queries%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Slow_queries  | 1     |
+---------------+-------+
1 row in set (0.00 sec)


mysql> set sql_log_bin=1;
Query OK, 0 rows affected (0.00 sec)
```

4、慢日志分析工具

```
[root@qfedu.com ~]# mysqldumpslow -s r -t 10 /data/3306/data/qfedu-slow.log
# 得到返回记录集最多的10个SQL
[root@qfedu.com ~]# mysqldumpslow -s c -t 10 /data/3306/data/qfedu-slow.log
# 得到访问次数最多的10个SQL
[root@qfedu.com ~]# mysqldumpslow -s t -t 10 -g "LEFT JOIN"
/data/3306/data/qfedu-slow.log   # 得到按照时间排序的前10条里面含有左连接的查询语句
[root@qfedu.com ~]# mysqldumpslow -s r -t 10 /data/3306/data/qfedu-slow.log |
more           # 结合| more使用，防止爆屏情况

s：表示按何种方式排序
c：访问次数
l：锁定时间
r：返回记录
t：查询时间
al：平均锁定时间
ar：平均返回记录数
at：平均查询时间
t：返回前面多少条的数据
g：后边搭配一个正则匹配模式，大小写不敏感
```