

# MySQL 数据库日志审计实验

## 一【实验目标】

- 熟练管理和使用 MySQL 二进制日志
- 熟练使用日志文件恢复数据

## 二【实验环境】

- Windows 10
- MySQL 5.5

## 三【实验原理】

my.ini 配置信息的 log-bin 没有指定文件扩展名，这是因为即使你指定上扩展名它也不使用。当 mysql 创建二进制日志文件时，首先创建一个以“mysql\_log\_bin”为名称，以“.index”为后缀的文件；再创建一个以“mysql\_log\_bin”为名称，以“.000001”为后缀的文件。当 mysql 服务重新启动一次以“.000001”为后缀的文件会增加一个，并且后缀名加 1 递增；如果日志长度超过了 max\_binlog\_size 的上限(默认是 1G)也会创建一个新的日志文件；使用 flushlogs(mysql 命令符)或者执行 mysqladmin - u - pflush-logs(Windows 命令提示符)也会创建一个新的日志文件。

MySQL 的二进制日志文件是 MySQL 安全系统里的非常重要的文件，虽然打开日志，所有非 SELECT 的操作都会记录下来，但这在某些场合显得很冗余，比如还原数据库，或者导入数据等，日志文件会一条不漏的记录下所有的命令，这会使日志文件迅速变大。但正是因为这样，让数据库的安全性变得更加可靠。下面将通过日志文件恢复误操作，找回丢失的数据。

#### 四【实验步骤】

1、输入密码【Admin123456】，登录系统。如图 1 所示。

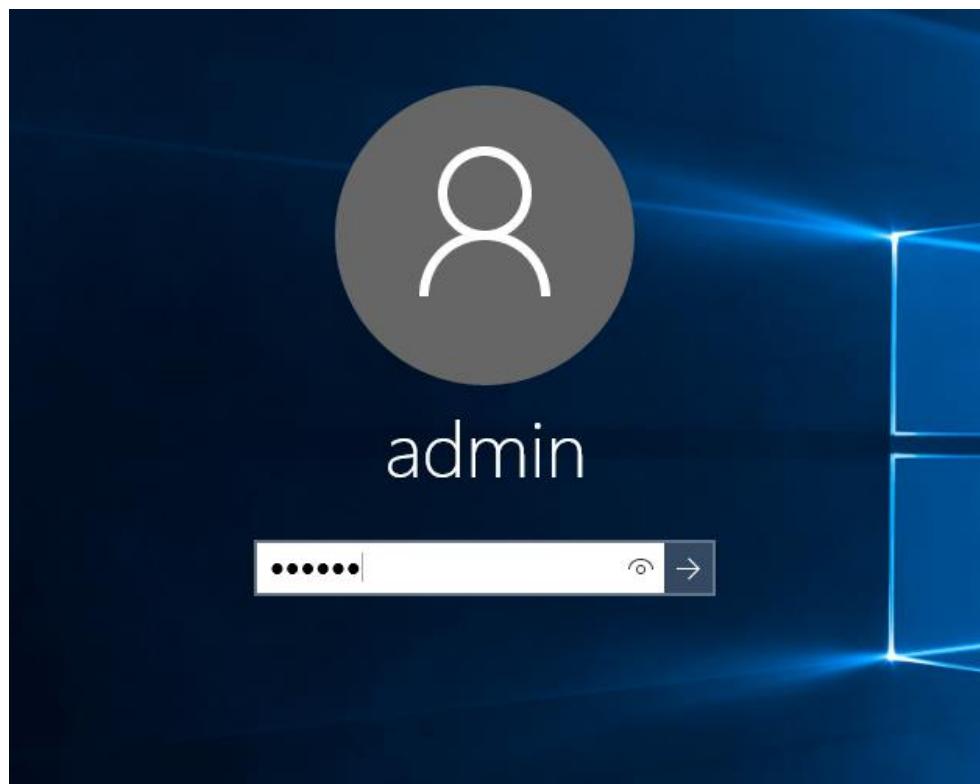


图 1

2、打开 MySQL 安装目录 C:\Tools\mysql\data。如图 2 所示。

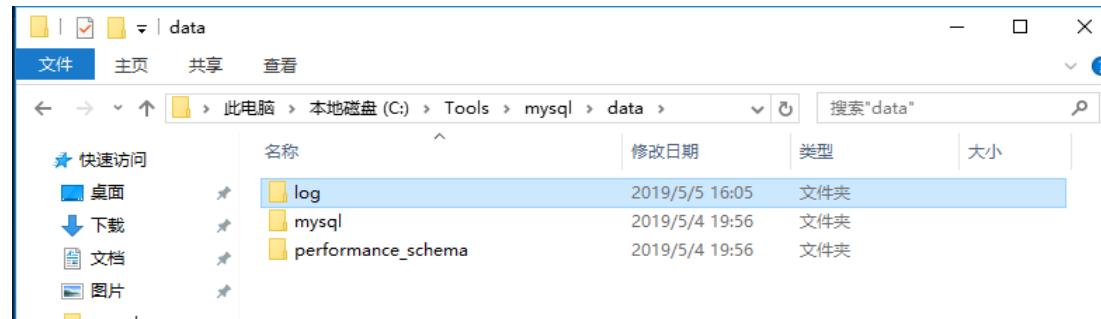


图 2

3、复制当前路径，如图 3 所示。

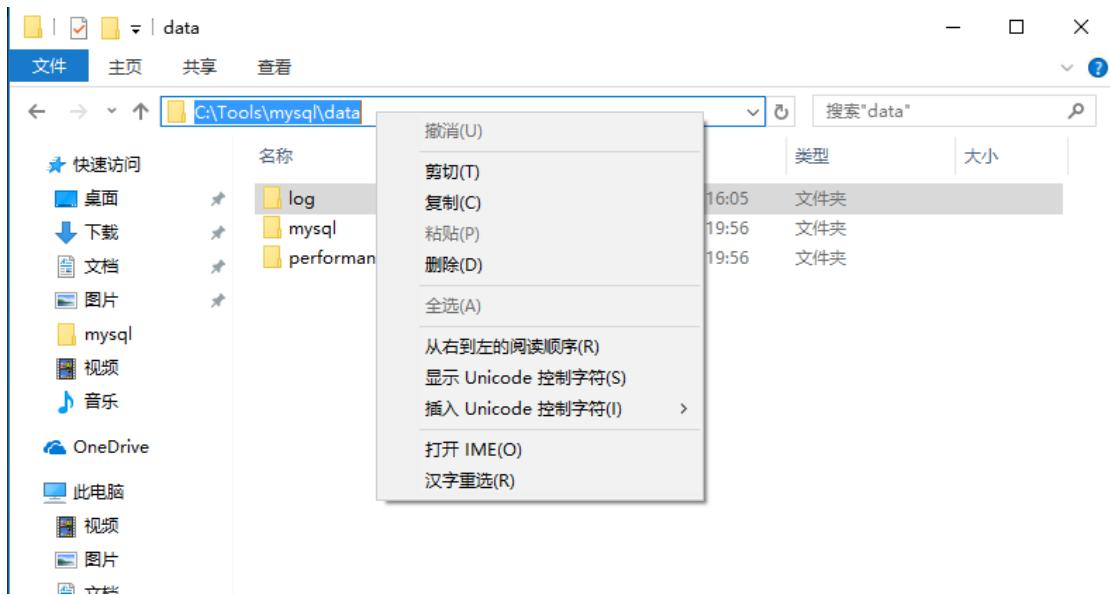


图 3

4、找到 MySQL 配置文件 my.ini。如图 4 所示。

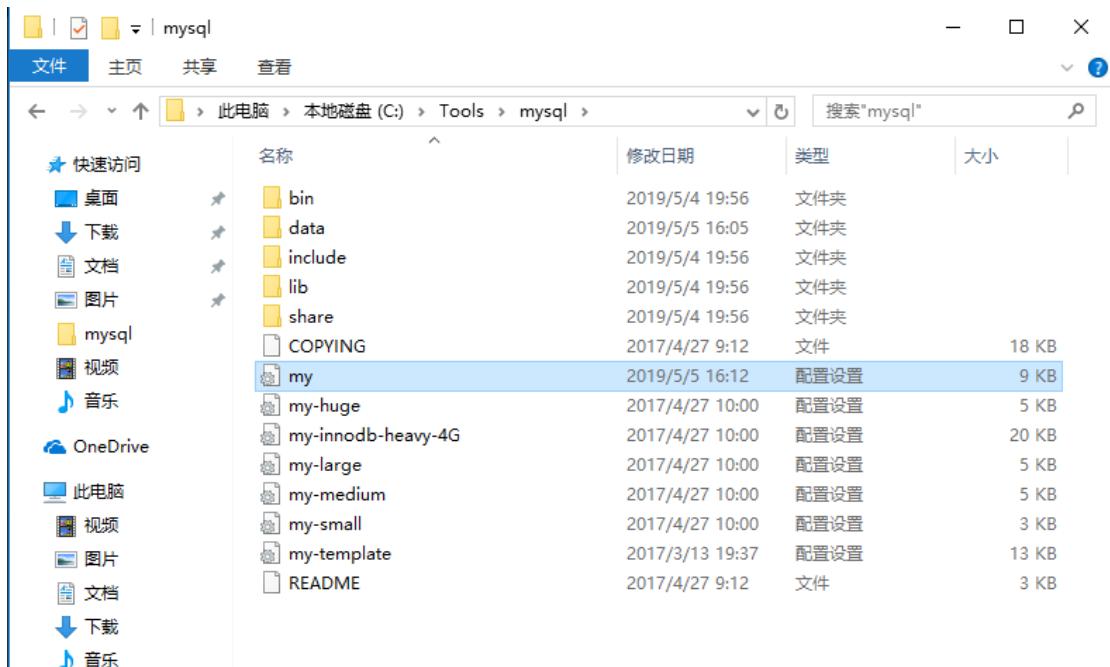


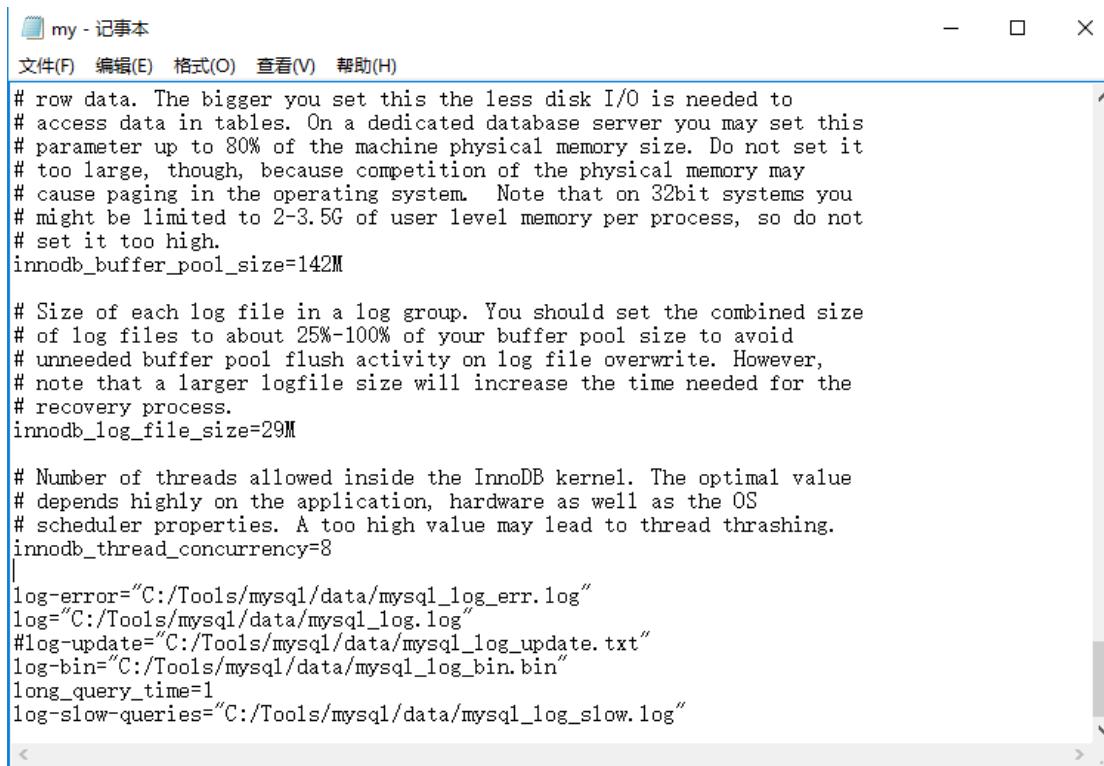
图 4

5、打开配置文件，在文件末尾添加内容

```
log-error="C:/Tools/mysql/data/mysql_log_err.log"  
log="C:/Tools/mysql/data/mysql_log.log"  
#log-update="C:/Tools/mysql/data/mysql_log_update.txt"  
log-bin="C:/Tools/mysql/data/mysql_log_bin.bin"  
long_query_time=1
```

log-slow-queries="C:/Tools/mysql/data/mysql\_log\_slow.log"

如图 5 所示。



```
# row data. The bigger you set this the less disk I/O is needed to
# access data in tables. On a dedicated database server you may set this
# parameter up to 80% of the machine physical memory size. Do not set it
# too large, though, because competition of the physical memory may
# cause paging in the operating system. Note that on 32bit systems you
# might be limited to 2-3.5G of user level memory per process, so do not
# set it too high.
innodb_buffer_pool_size=142M

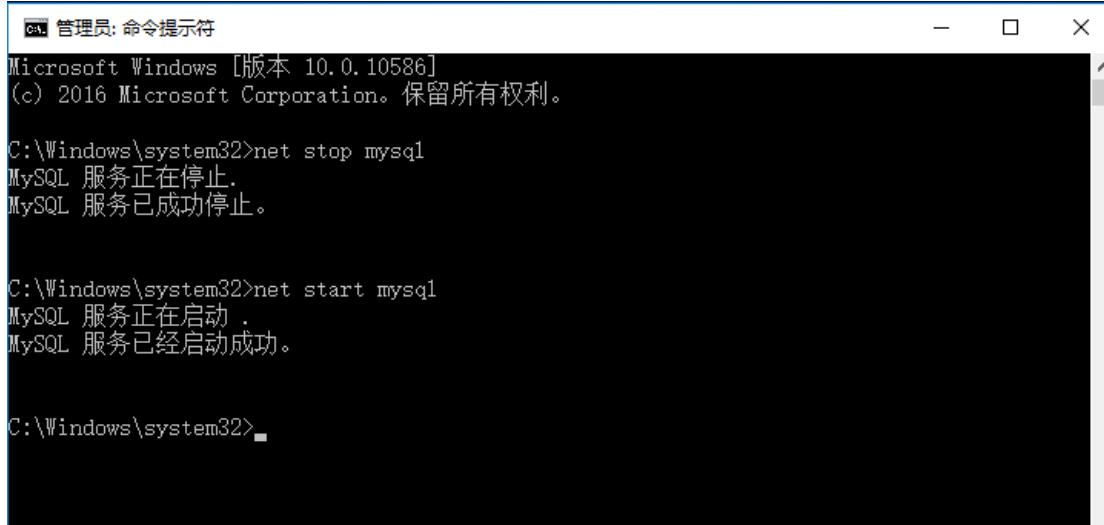
# Size of each log file in a log group. You should set the combined size
# of log files to about 25%-100% of your buffer pool size to avoid
# unneeded buffer pool flush activity on log file overwrite. However,
# note that a larger logfile size will increase the time needed for the
# recovery process.
innodb_log_file_size=29M

# Number of threads allowed inside the InnoDB kernel. The optimal value
# depends highly on the application, hardware as well as the OS
# scheduler properties. A too high value may lead to thread thrashing.
innodb_thread_concurrency=8

log-error="C:/Tools/mysql/data/mysql_log_err.log"
log="C:/Tools/mysql/data/mysql_log.log"
#log-update="C:/Tools/mysql/data/mysql_log_update.txt"
log-bin="C:/Tools/mysql/data/mysql_log_bin.bin"
long_query_time=1
log-slow-queries="C:/Tools/mysql/data/mysql_log_slow.log"
```

图 5

6、重启 MySQL 服务。如图 6 所示。



```
Administrator: 命令提示符
Microsoft Windows [版本 10.0.10586]
(c) 2016 Microsoft Corporation. 保留所有权利。

C:\Windows\system32>net stop mysql
MySQL 服务正在停止。
MySQL 服务已成功停止。

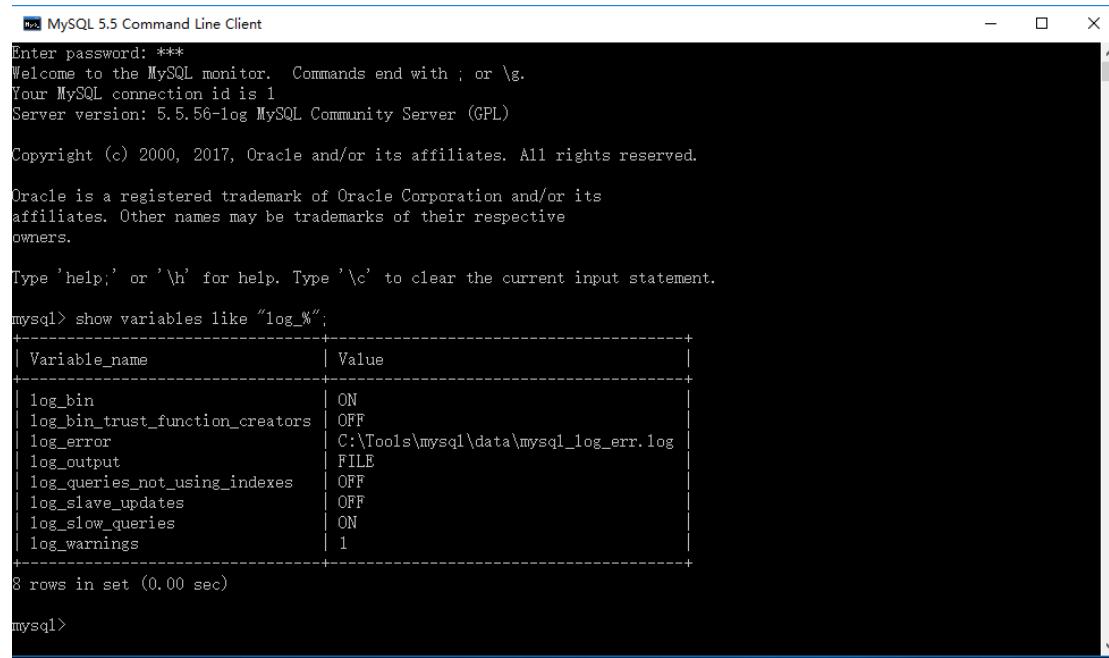
C:\Windows\system32>net start mysql
MySQL 服务正在启动 .
MySQL 服务已经启动成功。

C:\Windows\system32>
```

图 6

7、输入命令 show variables like 'log\_%';，查看 MySQL 日志的关闭与开启。

如图 7 所示。



MySQL 5.5 Command Line Client

```
Enter password: ***
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.5.56-log MySQL Community Server (GPL)

Copyright (c) 2000, 2017, 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.

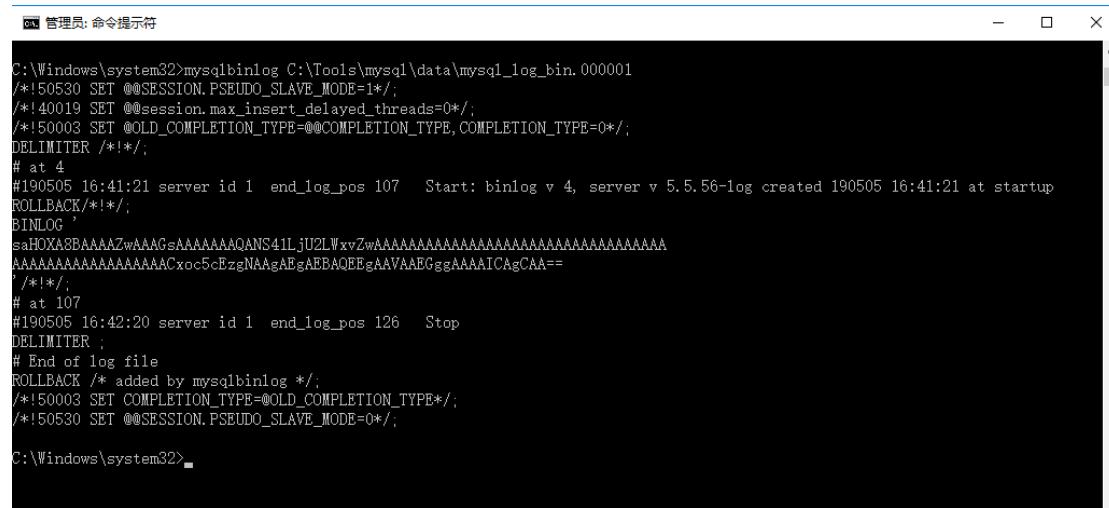
Type 'help,' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show variables like "log_%";
+-----+-----+
| Variable_name      | Value   |
+-----+-----+
| log_bin            | ON      |
| log_bin_trust_function_creators | OFF    |
| log_error          | C:\Tools\mysql\data\mysql_log_err.log |
| log_output          | FILE   |
| log_queries_not_using_indexes | OFF    |
| log_slave_updates  | OFF    |
| log_slow_queries    | ON     |
| log_warnings        | 1      |
+-----+-----+
8 rows in set (0.00 sec)

mysql>
```

图 7

8、输入命令：mysqlbinlog C:\Tools\mysql\data\mysql\_log\_bin.000001 查看日志。如图 8 所示。



管理员: 命令提示符

```
C:\Windows\system32>mysqlbinlog C:\Tools\mysql\data\mysql_log_bin.000001
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=1*/;
/*!40019 SET @@session.max_insert_delayed_threads=0*/;
/*!50003 SET @OLD_COMPLETION_TYPE=@@COMPLETION_TYPE,COMPLETION_TYPE=0*/;
DELIMITER /*!*/;
# at 4
#190505 16:41:21 server id 1  end_log_pos 107  Start: binlog v 4, server v 5.5.56-log created 190505 16:41:21 at startup
ROLLBACK/*!*/;
BINLOG '
saHOXA8BAAA2wAAAGsAAAAAAQANS41LJU2LWxvZwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAcxc5cEzgNAAgAEgAEPQEEgAAVAAEGggAAAAICAgCAA==
/*!*/;
# at 107
#190505 16:42:20 server id 1  end_log_pos 126  Stop
DELIMITER ;
# End of log file
ROLLBACK /* added by mysqlbinlog */;
/*!50003 SET COMPLETION_TYPE=@OLD_COMPLETION_TYPE*/;
/*!50530 SET @@SESSION.PSEUDO_SLAVE_MODE=0*/;

C:\Windows\system32>
```

图 8

9、输入命令：show master logs; 查看二进制日志数目。如图 9 所示。

```
MySQL 5.5 Command Line Client

mysql> show master logs;
+-----+-----+
| Log_name        | File_size |
+-----+-----+
| mysql_log_bin.000001 |      126 |
| mysql_log_bin.000002 |      126 |
| mysql_log_bin.000003 |      107 |
+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

图 9

10、输入命令：show master status；查看当前二进制日志。如图 10 所示。

```
mysql> show master status;
+-----+-----+-----+-----+
| File          | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+-----+-----+-----+-----+
| mysql_log_bin.000003 |      107 |           |           |
+-----+-----+-----+-----+
1 row in set (0.00 sec)

mysql>
```

图 10

11、输入命令：mysqlbinlog

C:\Tools\mysql\data\mysql\_log\_bin.000001 >aabb.sql 将二进制的日志文件转换成.sql 的文件，并输入命令 dir 查看。如图 11 所示。

```
管理员: 命令提示符

C:\Windows\System32>mysqlbinlog C:\Tools\mysql\data\mysql_log_bin.000001 >aabb.sql
C:\Windows\System32>dir
驱动器 C 中的卷没有标签。
卷的序列号是 6A02-1E50

C:\Windows\System32 的目录

2019/05/05 17:06 <DIR> .
2019/05/05 17:06 <DIR> ..
2015/10/31 00:09 <DIR> 0409
2015/10/30 15:17 760 @edptcoastimage.png
2015/10/30 15:18 600 @language_notification_icon.png
2015/10/30 15:18 600 @optionalfeatures.png
2015/10/30 15:18 120 @TileEmptyxIIImage.png
2015/10/30 15:17 15,106 @WiFiNotificationIcon.png
2019/05/05 17:06 738 aabb.sql
```

图 11

12、打开数据库，执行“use test；”切换到 test 数据库下，执行“show tables；”命令查看该数据库中的表，执行命令 “create table student(id int, name char(50), age int)；” 创建新表 student。如图 12 所示。

```
MySQL 5.5 Command Line Client

mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)

mysql> create table student(id int, name char(50), age int);
Query OK, 0 rows affected (0.02 sec)

mysql>
```

图 12

13、向 student 表中插入数据，并查询 student 表，如图 13 所示。

```
MySQL 5.5 Command Line Client

mysql> insert into student values(1,'name1',22),(2,'name2',21),(3,'name3',24);
Query OK, 3 rows affected (0.00 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from student;
+----+----+----+
| id | name | age |
+----+----+----+
| 1  | name1 | 22 |
| 2  | name2 | 21 |
| 3  | name3 | 24 |
+----+----+----+
3 rows in set (0.00 sec)
```

图 13

14、删除 student 表中数据，删除表，如图 14 所示。

```
mysql> delete from student;
Query OK, 3 rows affected (0.00 sec)

mysql> drop table student;
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
Empty set (0.00 sec)
```

图 14

15、查看二进制日志，后缀编号同第九步一致。如图 15 和图 16 所示。

```
管理员: 命令提示符
C:\Windows\system32>mysqlbinlog C:\Tools\mysql\data\mysql_log_bin.000003
```

图 15

```

# at 107
#190505 17:21:01 server id 1  end_log_pos 220  Query      thread_id=1      exec_time=0      error_code=0
use `test`/*!*/;
SET TIMESTAMP=1557048061/*!*/;
SET @@session.pseudo_thread_id=1/*!*/;
SET @@session.foreign_key_checks=1, @@session.sql_auto_is_null=0, @@session.unique_checks=1, @@session.autocommit=1/*!*/;
SET @@session.sql_mode=1344274432/*!*/;
SET @@session.auto_increment_increment=1, @@session.auto_increment_offset=1/*!*/;
/*!\\C utf8 */!*/
SET @@session.character_set_client=33,@@session.collation_connection=33,@@session.collation_server=33/*!*/;
SET @@session.lc_time_names=0/*!*/;
SET @@session.collation_database=DEFAULT/*!*/;
create table student(id int, name char(50), age int)
/*!*/;
# at 220
#190505 17:22:42 server id 1  end_log_pos 288  Query      thread_id=1      exec_time=0      error_code=0
SET TIMESTAMP=1557048162/*!*/;
BEGIN
/*!*/;
# at 288
#190505 17:22:42 server id 1  end_log_pos 421  Query      thread_id=1      exec_time=0      error_code=0
insert into student values(1,'name1',22),(2,'name2',21),(3,'name3',24)
/*!*/;
# at 421
#190505 17:22:42 server id 1  end_log_pos 448  Xid = 10
COMMIT/*!*/;
# at 448
#190505 17:23:38 server id 1  end_log_pos 516  Query      thread_id=1      exec_time=0      error_code=0
SET TIMESTAMP=1557048218/*!*/;
BEGIN

```

图 16

16、截取日志的第 107 行至 448 行，其中 107 为第九步中 position 的值 448 为删除操作开始前的值，存入 107448.sql 文件，如图 17 所示。

```
C:\Windows\System32>mysqlbinlog C:\Tools\mysql\data\mysql_log_bin.000003 --start-position=107 --stop-position=448 -r 107448.sql
```

图 17

17、把提取出来的日志文件段导入到数据库中，如图 18 所示。

```

MySQL 5.5 Command Line Client

mysql> source C:\Windows\SysWOW64\107448.sql
Query OK, 0 rows affected (0.00 sec)

Database changed
Query OK, 0 rows affected (0.00 sec)

Charset changed
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

```

图 18

18、查看恢复的结果，如图 19 所示。

```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| student       |
+-----+
1 row in set (0.00 sec)

mysql> select * from student;
+----+----+----+
| id | name | age |
+----+----+----+
| 1  | name1 | 22 |
| 2  | name2 | 21 |
| 3  | name3 | 24 |
+----+----+----+
3 rows in set (0.00 sec)
```

图 19

## 五【实验思考】

- 通过日志恢复数据与通过备份恢复数据各有何优劣？