目录

[第16章 mysql系统库之日志记录表 1](#_Toc16202)

[16.1　日志信息概述 1](#_Toc8359)

[16.2 日志表详解 3](#_Toc32674)

[16.2.1　general\_log 3](#_Toc19245)

[16.2.2　slow\_log 4](#_Toc6659)

[17.3　查看SQL日志信息 7](#_Toc6225)

**第16章 mysql系统库之日志记录表**

**16.1　日志信息概述**

#MySQL的日志系统包含：普通查询日志、慢查询日志、错误日志（记录服务器启动时、运行中、停止时的错误信息）、二进制日志（记录服务器运行过程中数据变更的逻辑日志）、中继日志（记录从库I/O线程从主库获取的主库数据变更日志）、DDL日志（记录DDL语句执行时的元数据变更信息。

#在MySQL 5.7中，只有普通查询日志、慢查询日志**支持写入表**中（**也支持写入文件**中），其他日志类型在MySQL 5.7中**只支持写入文件**中。**(注：所以不要把日志和日志表搞混了，日志的形式既有表也有文件，日志表的好处是供程序查询，但日志表本身不可改动)**

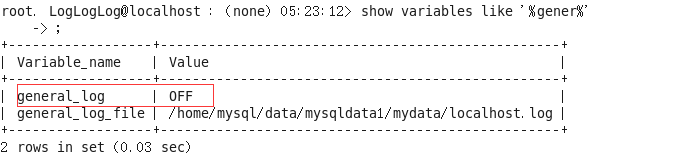
#默认情况下，所有日志均被写在datadir目录下

**注：datadir是指mysql存放数据库数据的地方**

**比如在mysql库里存放了各类日志表(其他地方也有)：**

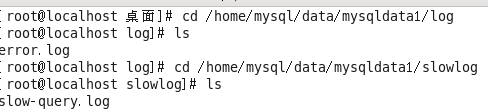
#**普通查询日志**：general\_log\_file=/home/mysql/data/mysqldata1/mydata/localhost.log

**注：为避免开销过大此日志不一定会开启，因此也未必有此日志文件**



#**错误日志**：log\_error=/home/mysql/data/mysqldata1/log/error.log

#**慢查询日志**：slow\_query\_log\_file=/home/mysql/data/mysqldata1/slowlog/slow-query.log

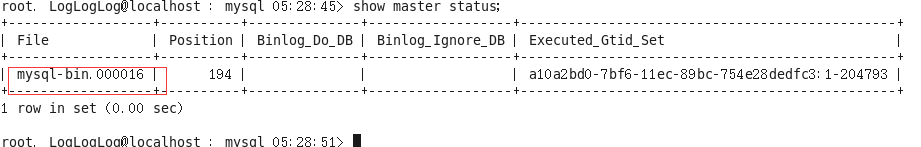


#**二进制日志**：log\_bin\_basename=/home/mysql/data/mysqldata1/binlog/mysql-bin/log\_bin\_index=/home/mysql/data/mysqldata1/binlog/mysql-bin.index

文本

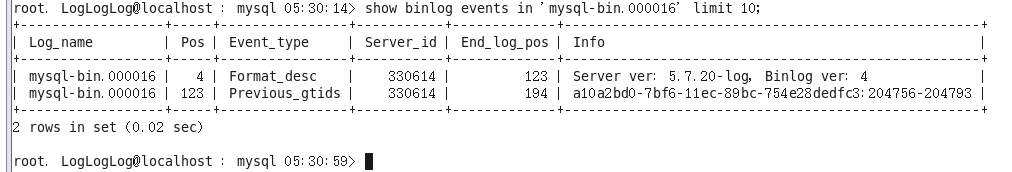
描述已自动生成

**注：可以知道现在用的是哪个日志。**



show binlog events in 'mysql-bin.000019' limit 10;

**然后查询日志信息如下：**



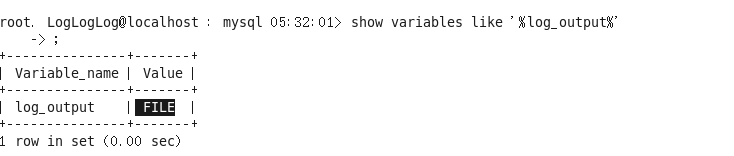
**16.2 日志表详解**

#普通查询日志(general\_log)和慢查询日志(slow\_log)可以通过log\_output=TABLE设置保存到mysql.general\_log表和mysql.slow\_log表中

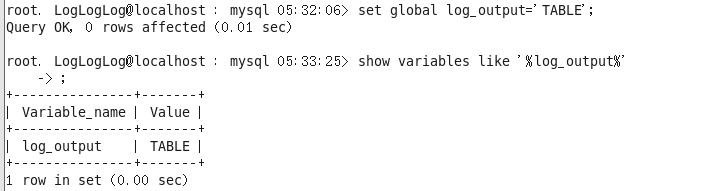
**16.2.1　general\_log**

#general\_log**表**提供查询普通SQL语句的执行记录信息，用于查看客户端到底在服务器上执行了什么SQL语句。

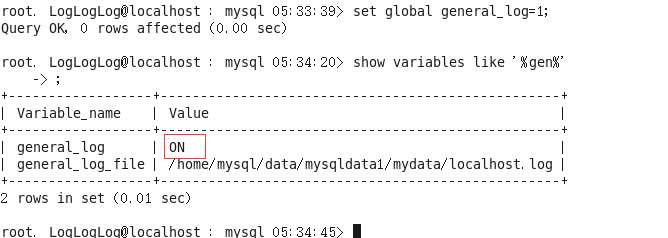
**注：默认普通查询日志保存在日志文件中，可以改为输出到general\_log表。这样做的好处是**



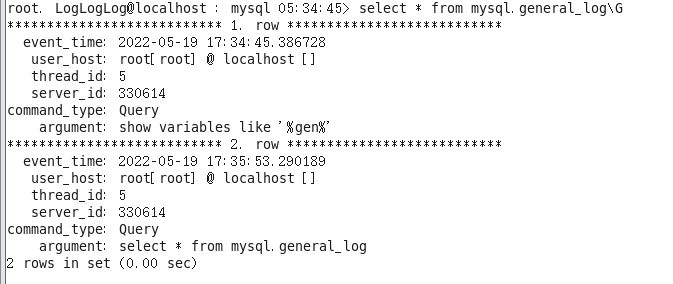
mysql> set global log\_output='TABLE';



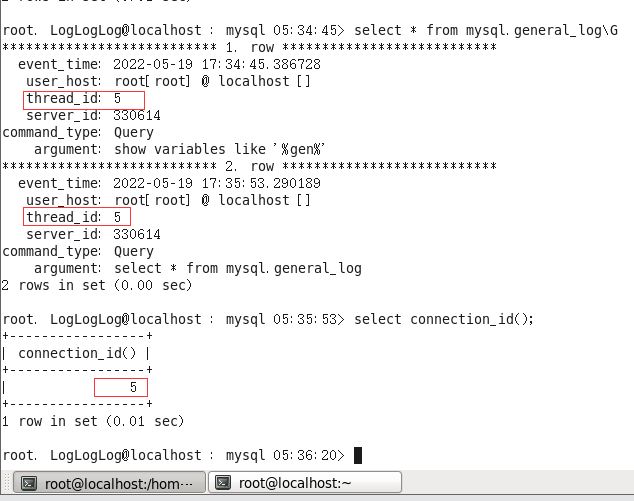
mysql> set global general\_log=1;



mysql> select \* from mysql.general\_log\G



mysql> select connection\_id();

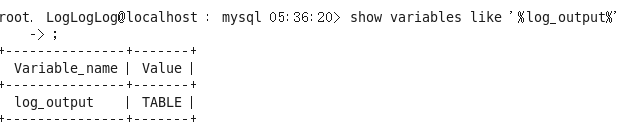


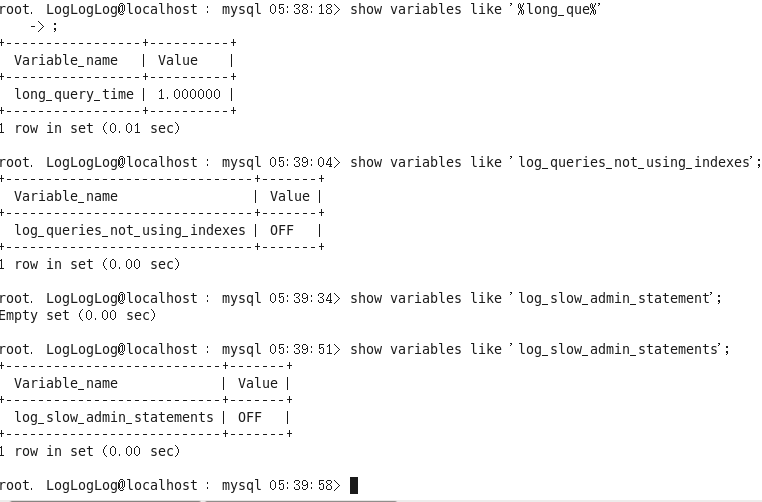
**注：以上信息主要含义就是何时（event\_time）、何用户(user\_host)通过何种渠道 (thread\_id和 process id)在何处(server\_id)执行了何种查询(argument)。mysqld按照接收请求的顺序将语句写入查询日志中。**

**16.2.2　slow\_log**

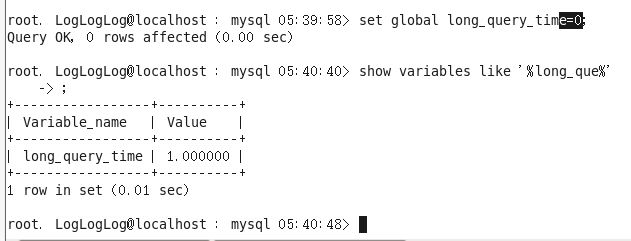
# slow\_log表提供查询执行时间**超过long\_query\_time设置值**的SQL语句、未使用索引的语句（需要开启参数log\_queries\_not\_using\_indexes=ON）或者管理语句（需要开启参数log\_slow\_admin\_statements=ON）。

**注：注意检查是否把日志输出到数据表而不是文件当中：**



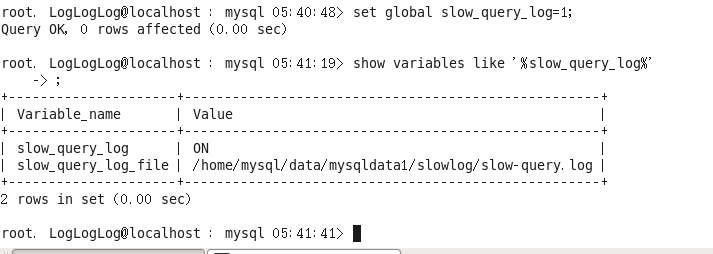


mysql> set global long\_query\_time=0;

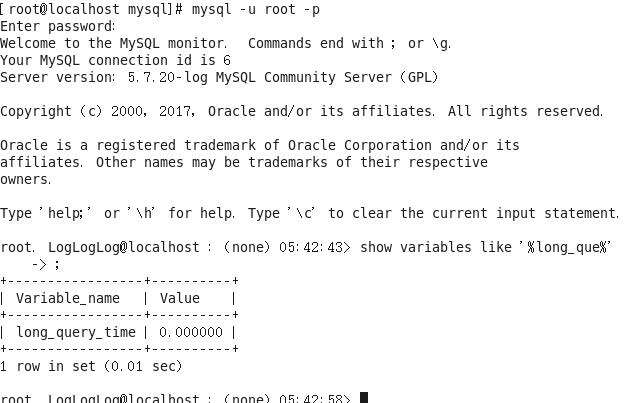


#开启慢查询，默认不开启

mysql> set global slow\_query\_log=1;



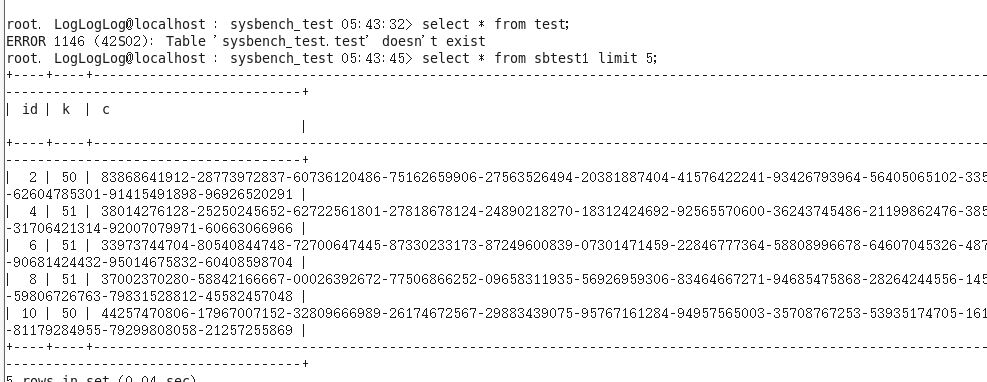
**注：这里为方便实验，把慢查询的时间标准(long\_query\_time)将为0(原来为1秒)。修改后变量值不会马上改变(还是1)。需要结束会话并重新开始：**



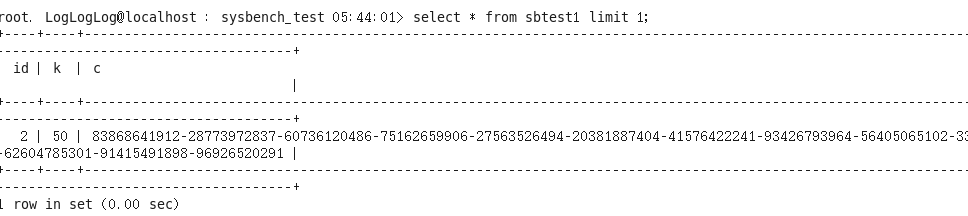
mysql> use test

mysql> show tables;

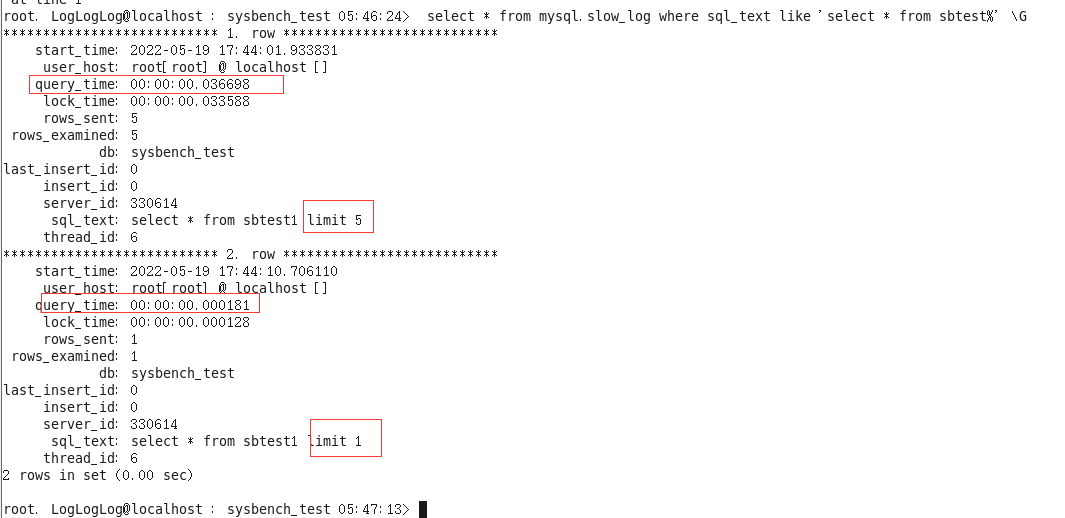
mysql> select \* from test;



**注：因为test库已不再采用，这里用了sysbench\_test。另一个查询只要求返回一个结果：**



mysql> select \* from mysql.slow\_log where sql\_text like 'select \* from sbtest%' \G



**注：对比一下上面这两个查询的query\_time。另外，字段rows\_sent告诉我们这两个查询各自返回的数据行数，lock\_time记录了慢查询语句持有锁的时间。**

**总结：请按如下流程图(当然要结合书本介绍)，对一条实际语句进行分析，有实际操作配合说明者更佳。**

#管理语句是指：ALTER TABLE、ANALYZE TABLE、CHECK TABLE、CREATE INDEX、DROP INDEX、OPTIMIZE TABLE和REPAIR TABLE。设置log\_slow\_admin\_statements=1 则日志将记录慢管理语句。

执行语句：

insert into test\_stat values(1,1,1);=>

（log\_slow\_admin\_statement=ON? No）=>

（=long\_query\_time?YES）=>

（min\_examined\_row\_limit=0?YES）=>

记录slog.log

图示

描述已自动生成

## 17.3　查看SQL日志信息

注：这个示例讲授了与慢查询日志有关的信息查看操作。mysql把超过一定执行时间的SQL记录到慢查询日志中。有一系列变量控制该日志的记录行为，这里参考了文章（给出了更易读的总结）：

<https://www.cnblogs.com/saneri/p/6656161.html>

图形用户界面, 文本, 应用程序, Word

描述已自动生成

#首先开启**第一个会话（会话1）**，并修改log\_output变量值为’TABLE'，修改慢查询日志记录时间为“0”，启用查询日志和慢查询日志记录功能。

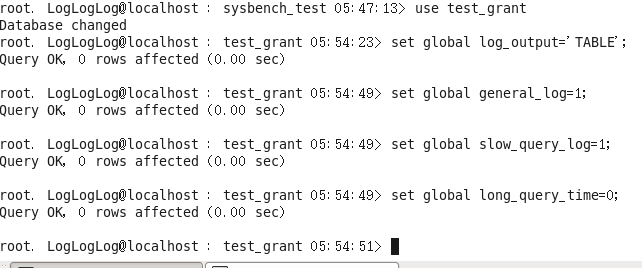
mysql>

set global log\_output='TABLE';

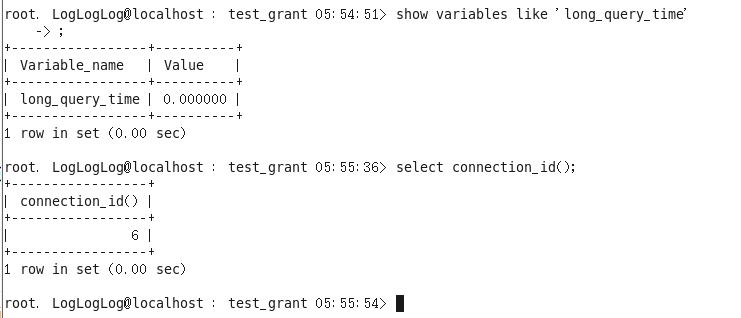
set global general\_log=1;

set global slow\_query\_log=1;

set global long\_query\_time=0;



注：long\_query\_time设置了不会立即生效，要退出会话然后重新登录才可。但这只是其中一个可能解决方法。总之要确认是否更改生效：

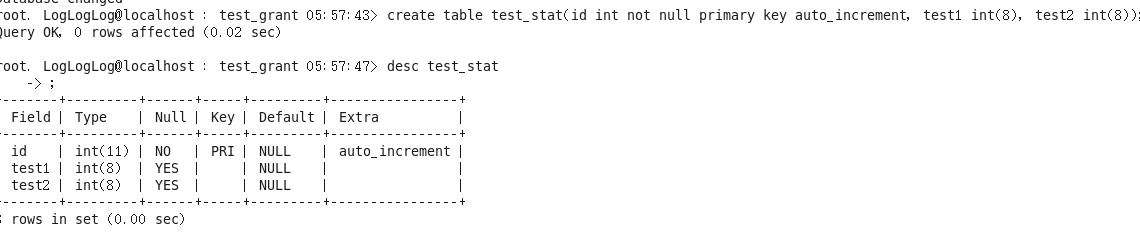


#开启**第二个会话（会话2）**，这里基于**17.2.1节**创建的表做进一步拓展，在会话2中对表加锁。

注：这里同样用上次实训用到的test\_grant替代。

注：需先额外新增数据表：

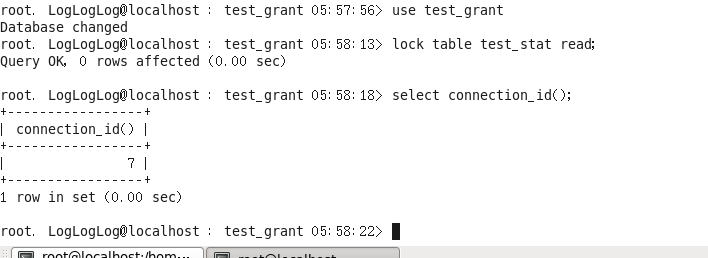
mysql> create table test\_stat(id int not null primary key auto\_increment, test1 int(8), test2 int(8));



mysql> use test\_grant

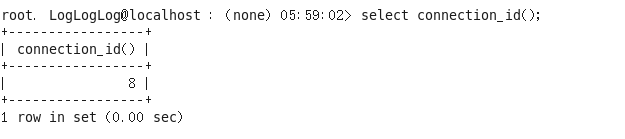
mysql> lock table test\_stat read;

mysql> select connection\_id();



#开启**第三个会话（会话3）**，在会话3中对test\_stat表插入一行数据。

mysql> select connection\_id();



mysql> use test\_grant

# 显式开启一个事务

mysql> begin;

# 被阻塞

mysql> insert into test\_stat values(1,1,1);

文本

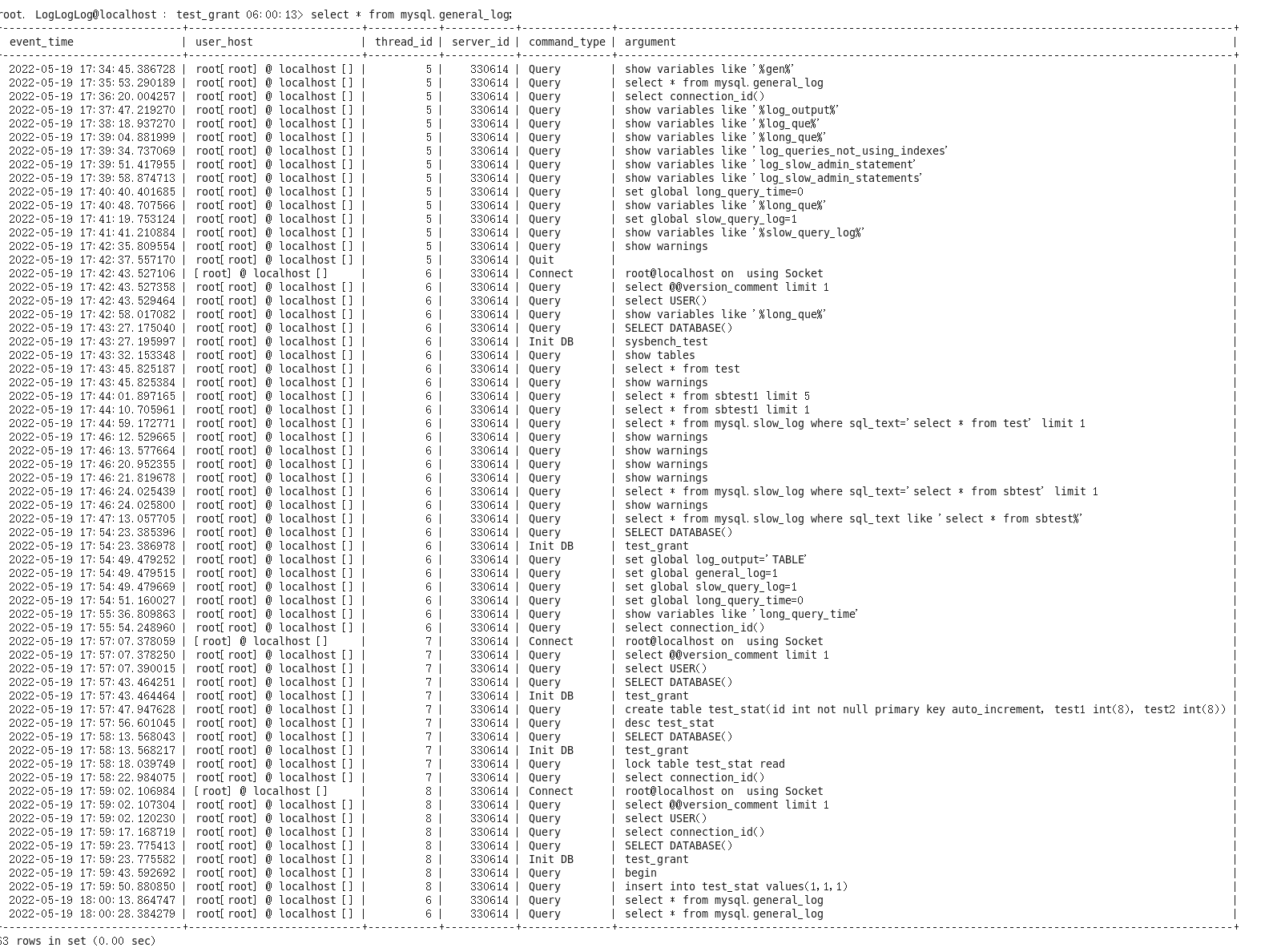
描述已自动生成

# 现在**回到会话1中**，查询mysql.general\_log表。

mysql> select \* from mysql.general\_log;

注：按照时间一直阅读下来，可以发现**general\_log表记录了这个实验的每次操作(当然也包括了一些误操作)**

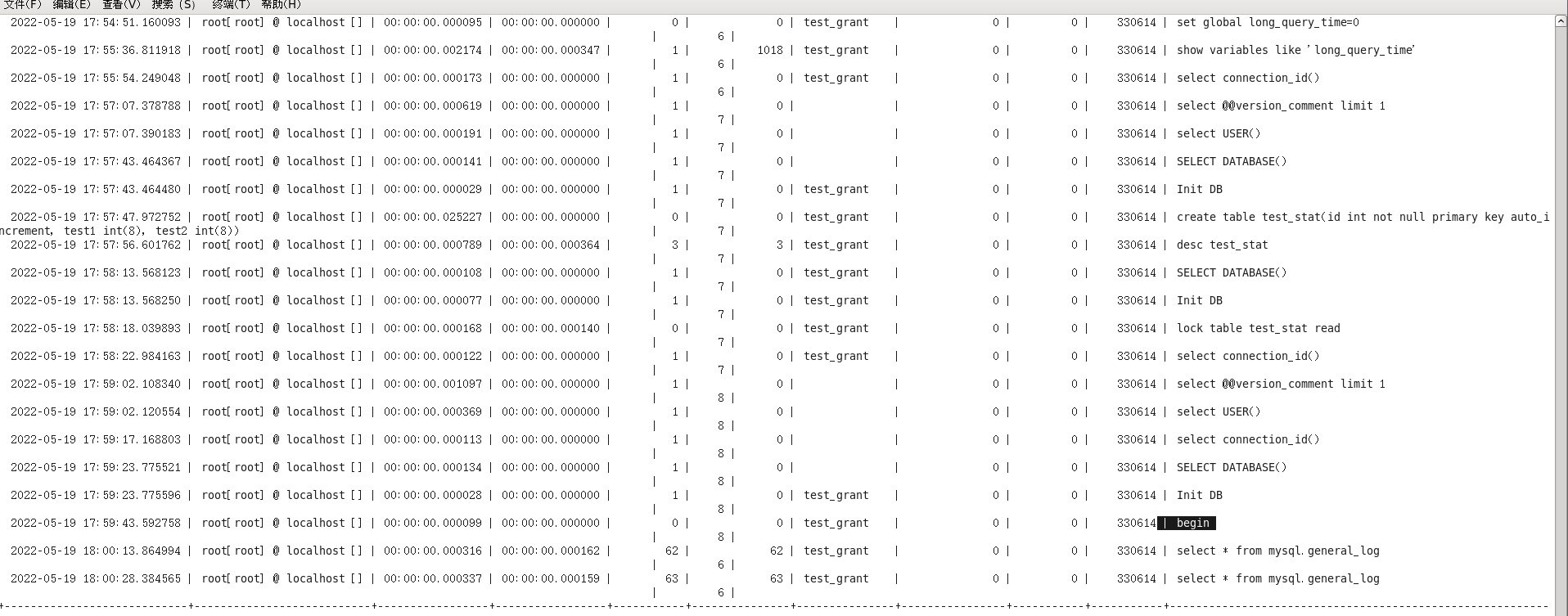
**会话123**



# 以上数据表明，general\_log表中的数据是在MySQL服务器接收到语句之后一开始执行就**立即记录**的。现在我们查询slow\_log表

mysql> select \* from mysql.slow\_log;

# 此时发现slow\_log表中**没有INSERT语句**的慢查询日志记录



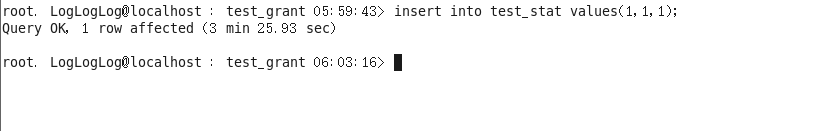
#**回到会话2**中进行解锁

mysql> unlock tables;



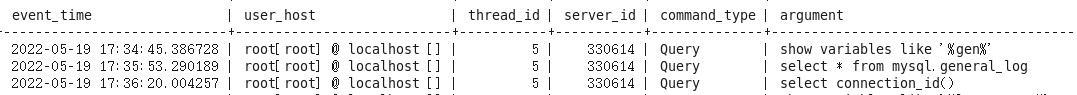
#**回到会话3**中，查看INSERT事务的执行结果。

mysql> insert into test\_stat values(1,1,1);



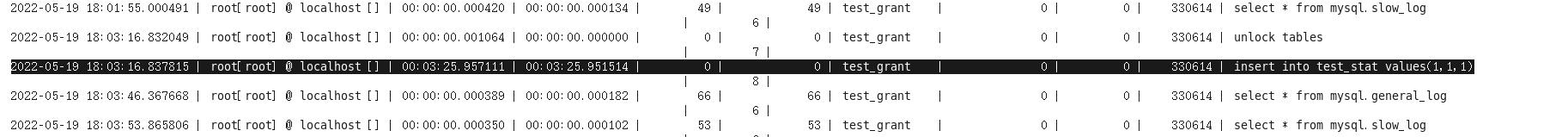
# **回到会话1**中，查询general\_log表和slow\_log表

mysql> select \* from mysql.general\_log;



# **会话1中**查询slow\_log表，注意**慢查询日志中的记录总是包含锁等待时间**

mysql> select \* from mysql.slow\_log;



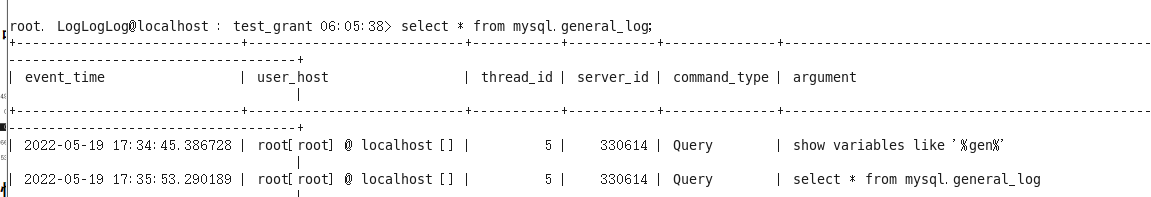
# 注意：此时会话3的事务仍未提交，所以在**慢查询日志表中记录的语句与事务无关**，只需要语句执行完成即会进行记录

#回到**会话3**中，对事务进行提交。

mysql> commit;

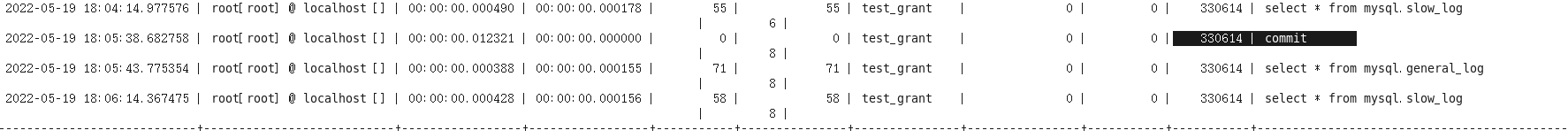
# general\_log表

mysql> select \* from mysql.general\_log;



# slow\_log表

mysql> select \* from mysql.slow\_log;

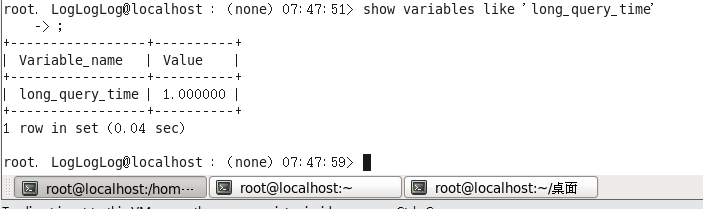


#综上所述，查询日志会在语句一开始执行时就进行记录，而慢查询语句会在语句执行完成并释放完所有的锁之后进行记录，两者的记录语句都与事务是否提交无关。

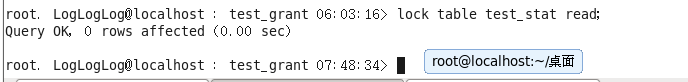
注：书中最后有一个扩展讨论，重新对以上实验修改如下：

会话1，把long\_query\_time变量值设置为1。记得退出会话重新登录并检查设置是否生效：

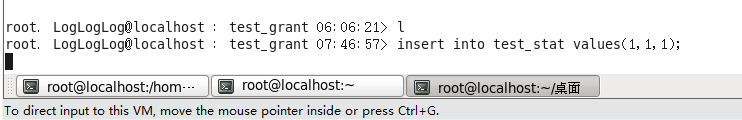




会话2，上锁



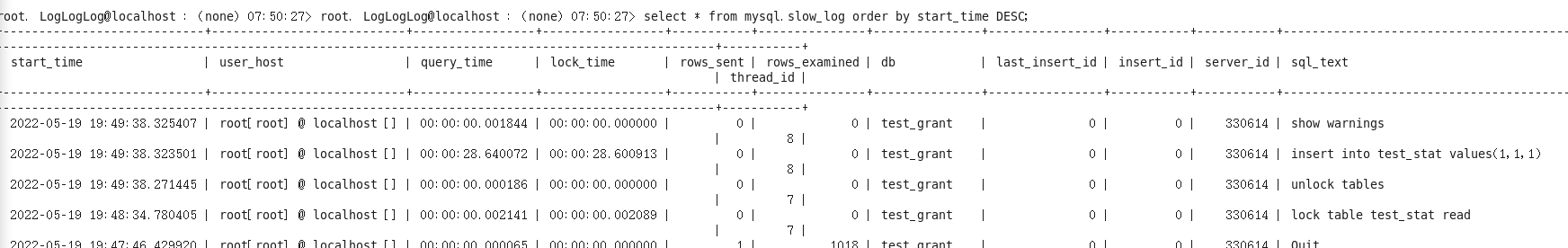
会话3，插入数据行被阻塞



然后会话2解锁：



回到会话1，查询slow\_log



可以发现并没有最近当前时间(19:51分左右)的insert操作被记录。