# 19 **为什么我只查一行的语句，也执行这么慢**

创建一个表，表里面有两个字段id和c,并且向表里插入10万行记录

mysql> CREATE TABLE `t` (

`id` int(11) NOT NULL,

`c` int(11) DEFAULT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB;

delimiter ;;

create procedure idata()

begin

declare i int;

set i=1;

while(i<=100000) do

insert into t values(i,i);

set i=i+1;

end while;

end;;

delimiter ;

call idata();

**第一类 查询长时间不返回**

**mysql> select \* from t where id=1;**

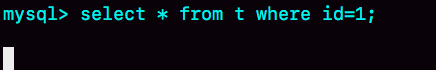


图 1 查询长时间不返回

一般碰到这种情况，大概率是表t被锁住了，首先执行一下show processlist命令，看看但是出于什么状态。

图2 就是show processlist命令查看 wait for table metadata lock的示意图

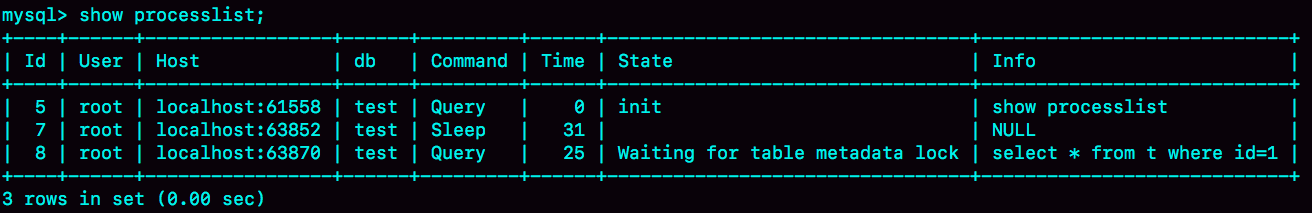


图2 wait for table metadata lock状态示意图

出现**这个状态表示的是，现在有一个线程正在表T上请求或持有MDL写锁，把select语句堵住了**

**在第6篇文章中，有一种复现方法，那个复现过程是MySQL5.6版本，而MySQL5.7版本修改了MDL加锁策略，所有就不能复现这个过程了**

**不过**，在MySQL5.7版本下复现这个过程，如图3所示，



图3，MySQL 5.7 MySQL中wait for table metadata lock的复现步骤

Session A 通过lock table 命令持有表t的MDL写锁，而session B的查询需要获取MDL读锁。所以，sessionB进入等待状态。

这类问题的处理方式，就是找到谁持有MDL写锁，然后把它kill掉

但是，由于在show processlist 的结果里面，session A的Command 列是 sleep ，导致查找起来不方便，不过有了performance\_shcema 和sys 系统库后，（MySQL启动时需要设置performance\_schema=on,相比设置为off会有10%左右的性能损耗）

通过插叙，sys.schema\_table\_lock\_wait这张表，我们可以找出直接造成阻塞的process id,

把这个用kill命令断开即可。

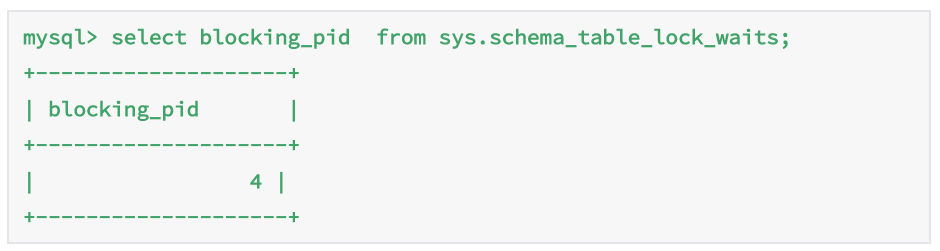


图 3 查获 加锁表的线程id

**等 flush**

接下来，另一种查询被堵住的情况

mysql> select \* from information\_schema.processlist where id=1;

可以看下图5 ，查出来这个线程的状态是 Wait for table flush,可以设想一下这是什么原因

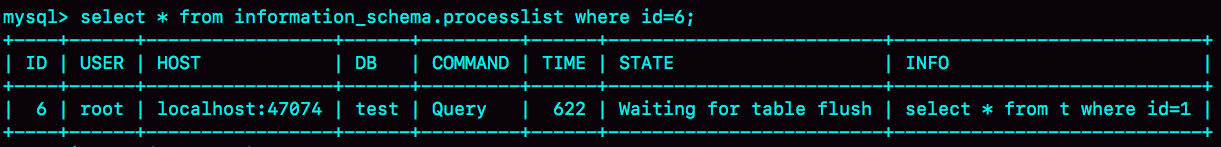


图 5 wait for table flush状态示意图

这个状态表示的是，现在有一个线程证要对表t做flush操作，MySQL里面对表做flush 操作的用法，一般有一下两个

flush tables t with read lock;

flush tables with read lock;

这两个flush 操作，如果指定表t的话，代表只是关闭表t；如果没有指定具体的表，则表示关闭MySQL里所有打开的表。

但是正常这两个语句执行起来都很快，除非他们也被别的线程堵住了。

所以，出现wait for table flush的状态可能情况是：有一个flush tables 命令被别的语句堵住了，然后有堵住了select 语句

复现步骤如下



图6， wait for table flush的复现步骤

Session A中故意没调用一次sleep(1),这样这个语句默认要执行10秒，在这期间表t一直是被session A“打开”着，然后，session B 的flush tables t 命令再要去关闭表t，就需要等session A的查询结束，这样，session c 要再次查询的话，就会被flush 命令堵住。

图7 是这个复现步骤的show processlist结果，

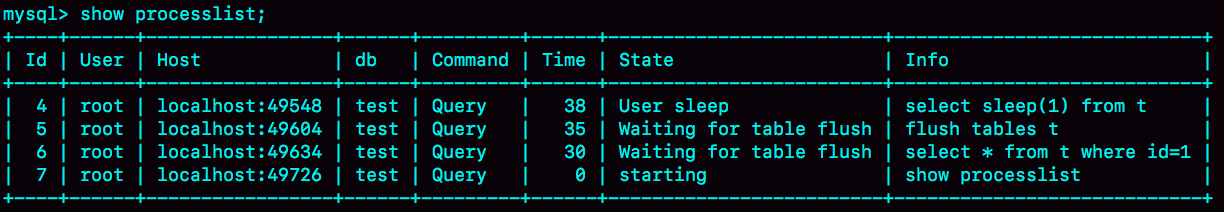


图 7 Wait for table flush 的show processlist结果

**等行锁**

现在，经过了表级锁的考验，select 语句终于来到引擎

mysql> select \* from t where id=1 lock in share mode;

由于访问 id=1 这个记录是要加读锁，如果这时候，已经有了一个事务在这行记录上持有一个写锁，select 语句会被堵住。

复现步骤和现场如下



图 8 行锁复现

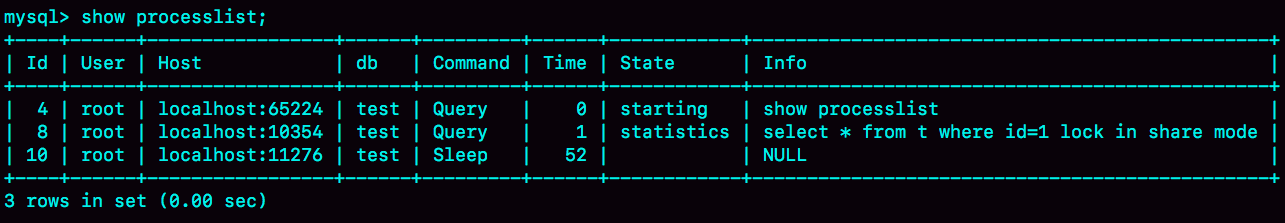


图9 show processlist 现场

显然，session A启动了事务，占有了写锁，还没有提交，导致sessionB堵住

这个问题不难分析，问题是怎么查出谁占用了这个写锁，如果用到的是MySQL5.7 ,可以通过sys.innodb\_lock\_waits表查到。

查询方式

mysql> select \* from t sys.innodb\_lock\_waits where locked\_table=`'test'.'t'`\G

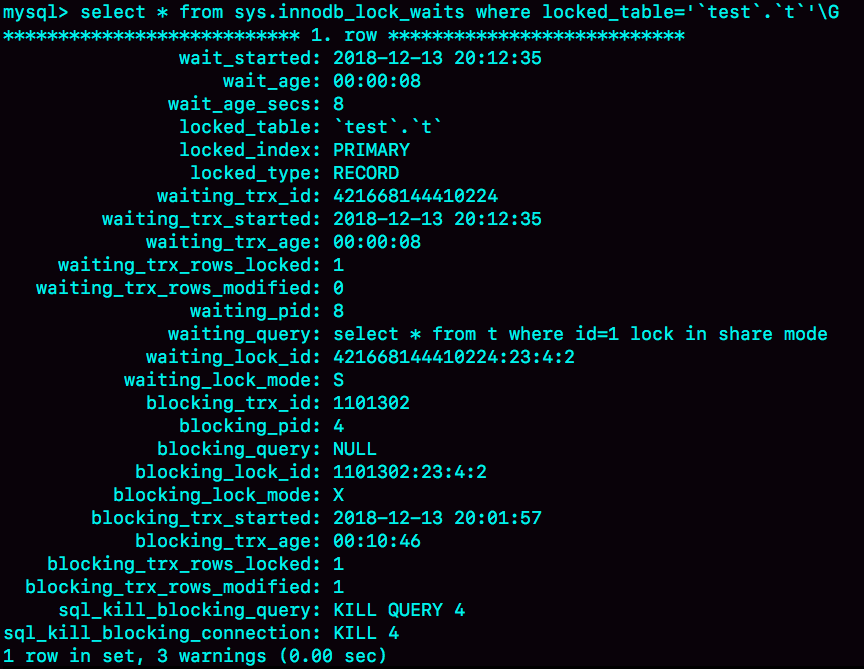


图 10 通过sys.innodb\_lock\_waits 查行锁

可以看到，4号线程造成阻塞，干掉这个线程就是kill query 4 或 kill 4

这里不该显示KILL QUERY 4 .这个命令表示停止4号线程当前正在执行的语句，而这个方法其实是没用的。因为占用行锁的是update语句，这个语句已经是之前执行完成了，现在执行Kill QUERY,无法让这个事务去掉id=1上的锁。

KILL 4 才有效，也就是直接断开这个连接，这里隐含的一个逻辑是，连接被断开的时候，会自动回滚这个链接里的正在执行的线程，也就释放了id=1上的行锁。

**第二类 查询慢**

经过了“锁”看看一些慢查询的例子

mysql> select \* from t where c=50000 limit 1;

由于字段c上没有索引，这个语句只能走id主键顺序扫描，因此需要扫描5万行。

查看慢日志 在slow log里，设置 set long\_query\_time=0.

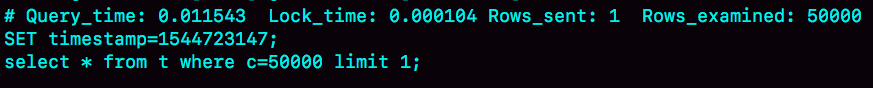


图11 全表扫描5万行的slow log

Rows\_examined显示扫描5万行，一般配置超过1秒才算慢查询，但是要记住：**坏查询不应定是慢查询**。例子中只有10万行数据，如果数据量大的话，执行时间线性上涨。

扫描越多，所以执行越慢，这个好理解

再看只扫描一行，但是执行很慢的语句

如图12所示，

**mysql> select \* from t where id=1；**

虽然扫描行数是1，但执行时间是800毫秒。

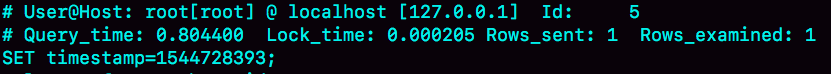


图12 扫描一行却执行的很慢

如果吧这个slow\_log的截图在往下一点，可以看到下面一个语句select \*from t where id= 1 lock in share mode,执行扫描行数也是1，执行时间是0.2毫秒。

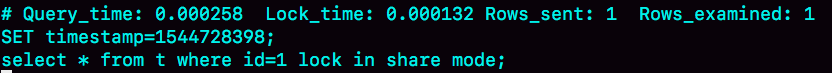


图13 加上 lock in share mode 的slow log

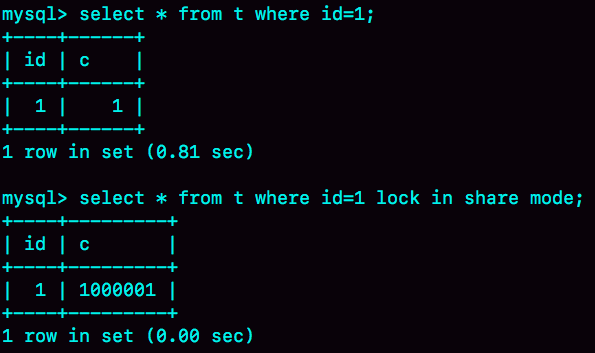


图14 两个语句的输出结果

第一个语句的查询结果是c=1,带lock in share mode的语句返回的是c=1000001,不明白原因看下复现步骤



图 15 复现步骤

SessionA 用start transaction with consistent snapshot开启了一个事务，之后sessionB才开始执行。

Session B执行完100次update语句，id=1这一行处于什么状态，

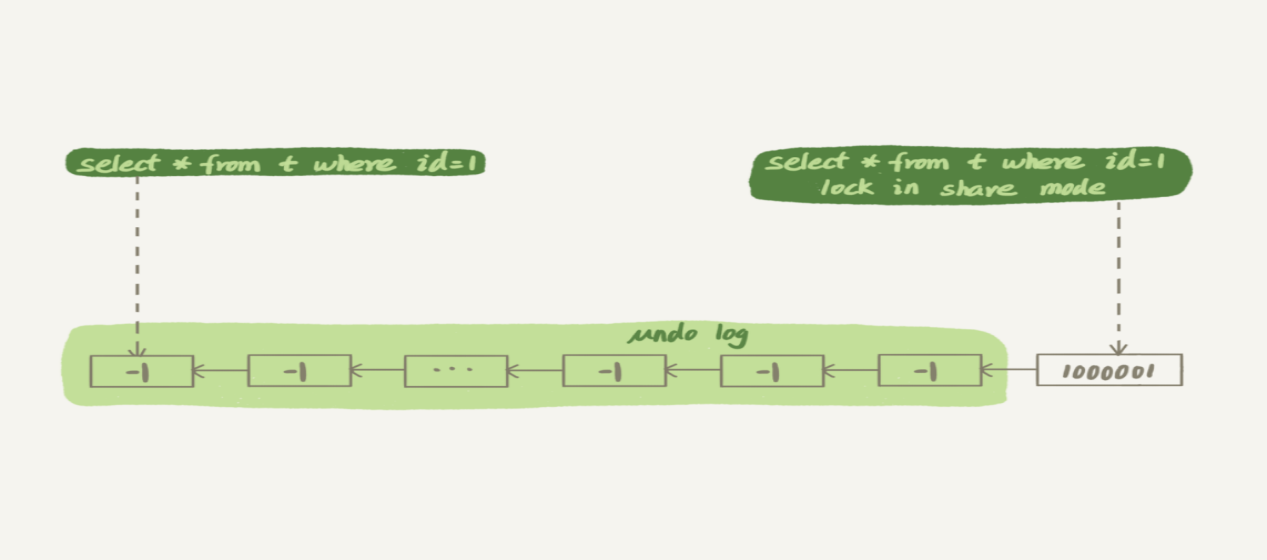


图16 id = 1的数据状态

SessionB 更新完100万次，生成100万个回滚日志（undo log）

带lock in share mode的SQL语句，是当前读，因此会直接读到1000001这个结果，所以速度很快，而select \* from t where id =1 ，这个语句，是一致性读，因此需要从1000001开始依次执行undo log，执行100万次以后，才将1这个结果返回

注意 undo log 里记录的器是吧“2”变成“1”，把”3”变成“2”这样的操作逻辑，化成简图的目的是方便看

**思考题**

举例加锁的时候，用的这个语句，select \* from where id = 1 lock in share mode由于id上有索引，所以可以直接定位id=1这一行，因此读锁也只是加在这一行上。

但是如下SQL语句

begin;

select \* from t where c=5 for update;

commit;

这个语句序列是怎么加锁的呢？加的锁又是什么时候释放的呢？