32 | 为什么还有kill不掉的语句

MySQL中两个kill命令

1. Kill query + 线程id, 表示终止这个线程正在执行的语句
2. Kill connection + 线程id ,connection可省略，表示断开这个线程的连接

如果这个线程有语句正在执行，也是要先停止正在执行的语句

使用kill命令，却没能断开这个链接。在show processlist命令，看到这条语句的Command列显示的是killed

大多数情况下，kill query/connection 命令是有效的，比如一个查询的过程中，发现执行的时间太久，要放弃查询，使用kill query命令，终止这条语句，

还有另外一种情况是，语句处于**锁等待**的时候，直接使用kill命令也是有效的

看下这个例子

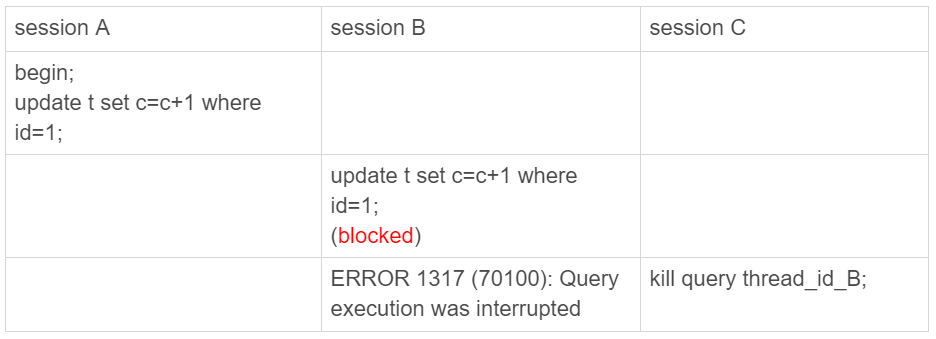


图 1 kill query成功的例子

Session c 执行kill query以后，session B几乎同时就提示语句被中断，符合预期

**收到kill以后，线程做什么？**

Session B是直接终止掉线程，什么都不管就直接退出吗？-- 显然，不行的

第6篇讲过，当对一个表锁增删改查操作时，会在表上加MDL读锁，所以，sessionB虽然处于blocked状态，还是拿着一个MDL读锁的，如果线程被kill的时，就直接终止，之后MDL锁就没有机会被释放了。

这样看来，kill并不是马上停止的意思，而是告诉执行线程，这条语句已经不需要继续执行了，可以开始“执行停止逻辑了”。

注意 跟linux的kill命令类似，kill-N pid 并不是让进程直接停止，而是给进程一个信号，然后进程处理这个信号，进入终止逻辑，只是对于MySQL的kill命令来说，不需要传信号参数，就只有“停止”这个命令

**实现上，当用户执行kill query thread\_id\_B时，MySQL里处理kill命令的线程做了两件事；**

1. 把sessionB的运行状态改成THE::KILL\_QUERY(将变量killed赋值为THD::KILL\_QUERY)
2. 给sessionB的执行线程发一个信号。

为什么要发信号？

因为想图1的例子里，sessionB处于锁等待状态，如果只是想把sessionB的线程状态设置THD::KILL\_QUERY,线程B并不知道这个状态变化，还是会继续执行。发一个信号的目的是，让sessionB退出等待，来处理这个THD::KILL\_QUERY状态。

上面的分析，隐含三个意思

1. 一个语句执行过程中有多出“埋点”，在这些“埋点”的地方判断线程状态，如果发现线程状态是THD::KILL\_QUERY,才开始进入语句终止逻辑
2. 如果处于等待状态，必须是一个可以被唤醒的等待，否则根本不会执行到“埋点”处
3. 语句从开始进入终止逻辑，到终止逻辑完成，是一个过程的

**看一个kill不掉的例子，**也29篇中提到的innodb\_thread\_concurrency不够用的例子

首先，执行set global innodb\_thread\_concurrency=2，将innoDB的并发线程上限数设置为2，然后执行下面的序列

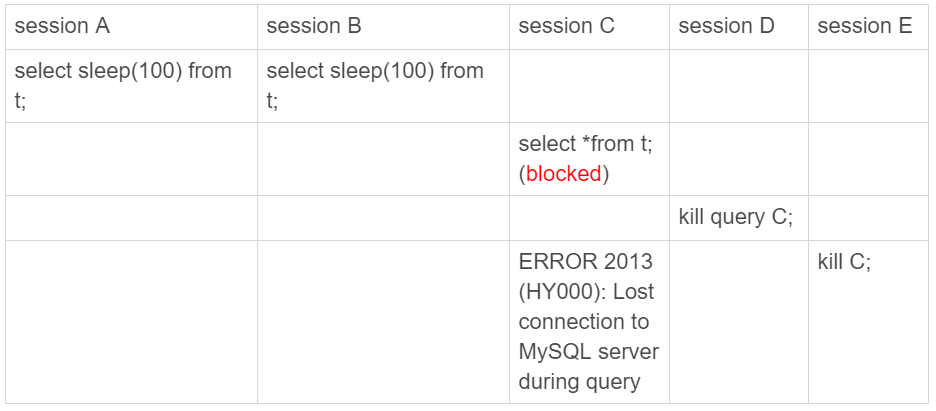


图 2 kill query无效的例子

1 session C执行的时候被堵住了

2 但是seesionD 执行的kill query C命令没什么效果

3 知道sessionE执行了 kill connection命令，才断开了session C的连接，提示“Lost connection to MySQL server during query”

4 此时，如果在sessionE 中执行show processlist,看看这个图

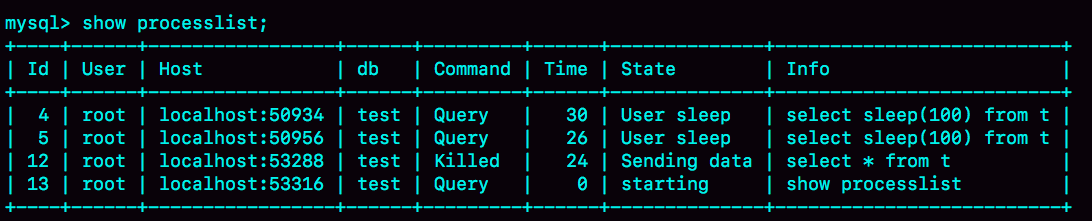


图3 kill connection 之后的效果

此时，id=12这个线程的Command列显示的Killed，也就是说，客户端虽然断开了连接，但实际上服务端这条语句还在执行过程中。

**为什么在执行kill query命令时，这条语句不像第一个例子的update语句一样退出？**

在实现上，等行锁是，使用的是pthread\_cond\_timewait函数，这个等待状态可以被唤醒。但是，在这个例子里，12号线程等待逻辑是这样的：每10毫秒判断一下是否可以进入InnoDB执行，如果不行，就调用nanosleep函数进入sleep状态。

也就是说，虽然12号线程的状态已经被设置为KILL\_QUERY,但是在这个等待进入InnoDB的循环过程中，并没有去判断线程的状态，因此根本不会进入终止逻辑阶段。

而当sessionE执行kill connection命令时，是这么做的饿

1. 把12号线程状态设置为KILL\_CONNECTION
2. 关掉12号线程的网络连接，因为这个操作，此时sessionC收到了断开连接的提示

为什么执行show processlist时，会看到Command显示为killed?这是因为在执行 show processlist的时候，有个特殊的逻辑

如果一个线程的状态是KILL\_CONNECTION，就把Command列显示成Killed。

即使客户端退出，这个线程的状态仍然是等待中，这个线程什么时候会退出？

答案是，只有等到满足进入InnoDB的条件后，sessionC的查询语句继续执行，然后才有可能判断到线程状态已经变为了KIll\_QUERY或者KILL\_CONNECTION,再进入终止逻辑阶段

**这个例子是kill无效的第一类情况，即：线程没有执行判断线程状态的逻辑。**跟这种情况类相同的，还有IO压力大，读写IO的函数一直无法返回，导致不能及时判断线程的状态的

**另一类，终止逻辑耗时较长，**此时，从show processlist结果看Command=Killed，需要等到终止逻辑完成，语句才算真正完成，这类情况，比较常见的场景如下

1. 超大事务执行期间被kill,此时，回滚操作需要对事务执行期间生成的所有新数据版本做回收操作，耗时很长
2. 大查询回滚，如果查询过程中生成了比较大的临时文件，加上此时文件系统压力大，删除临时文件可能需要等待IO资源，导致耗时较长
3. DDL命令执行到最后阶段，如果被Kill,需要删除中间过程的临时文件，也可能受IO资源影响耗时较久。

如果直接在客户端通过Ctrl+C 命令，是不是就可以直接终止线程呢？

答案：不可以

这里有一个误解，其实在客户端的操作只能操作客户端的线程，客户端和服务端只能通过网络交互，是不可能直接操作服务端线程的。

由于MySQL是停等协议，所以这个线程执行的语句还没有返回的时候，再往这个连接里继续发命令也是没有用的，实际上，执行ctrl+ c的时候，是MySQL客户端另外启动一个连接，然后发送一个kill query命令

**另外两个关于客户端的误解**

**第一个误解是：如果库里面的表特别多，连接就会很慢**

有些线上的库，会包含很多表，此时，每次用客户端连接都会卡在下面的界面上

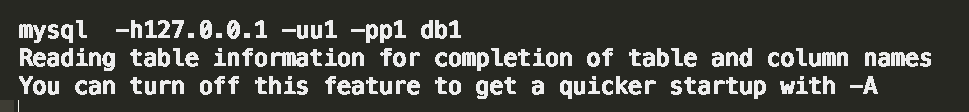


图 4 连接等待

如果db1这个库表很少的话，连接起来就很快，可以很快的进入命令的状态，因此会认为表的性能影响连接性能

第一篇讲过，每个客户端和服务端建立连接的时候，需要做的事就是TCP握手，用户校验，获取权限，这几个操纵，跟数据库里面的表的数量无关。

实际上，正如图中的文字提示所说的，当使用默认参数连接的时候，MySQL客户端会提供一个本地库名和表名补全的功能，为了实现这个功能，客户端连接成功后，需要多做一些操作

1. 执行show processlist
2. 切到db1库，这是show tables;
3. 把两个命令的结果用于构建一个本地的哈希表

在这些操作中，醉花时间的是第三步在本地构建哈希表的操作。所以，当一个库中的表的个数非常多，这一步就会花很长时间。

**我们感知到连接过程慢，其实并不是连接慢，也不是服务端慢，而是客户端慢**

图中的提示说，如果在连接命令中加入-A,就可以关掉这个自动补全的功能，然后客户端就可以快速返回了。

这里自动补全的效果就是，在输入库名或者表明时，输入前缀，可以使用tab键自动补全表名或显示提示。

实际使用中，如果自动补全功能用得并不多，建议每次使用默认加-A

提示里没有说，除了加-A以外，加-quick(或简写为-q)参数，也可以跳过这个阶段，但是，这个**-quick是一个更容易引起误会的参数，也是关于客户端常见的一个误解**

是不是觉得这个参数应该是一个让服务端加速的参数？但恰恰相反，设置了这个参数可能会降低服务端的性能，为什么这么说呢？

MySQL客户端发送请求后，接收服务端返回的结果的方式有两种

1. 一种是本地缓存，也就是在本地开一片内存，先把结果存起来。如果用API开发，对应的是mysql\_store\_result方法

2.另一种是不缓存，对一个处理一个，如果用API开发，对应的是mysql\_use\_result方法

MySQL客户端默认采用第一种方式，而如果加上-quick参数，就会使用第二种不缓存的方式

采用不缓存的方式，如果本地处理的慢，就会导致服务端发送结果被阻塞，因此会让服务端变慢，关于服务端的具体行为，以后介绍

为什么去这个参数取名叫quick呢？这是因为使用这个参数可达到以下三点效果

·第一点，就是前面提到的，跳过表名自动补全功能

·第二点，mysql\_store\_result,需要申请本地内存来缓存查询结果，如果结果太大，会耗费较多的本地内存，可能会影响客户端本地机器的性能

·第三点，是不会执行命令记录到本地的命令历史文件

所以 -quick参数的意思，是让客户端变得更快

**思考题**

如果碰到一个被killed的事务一直处于回滚状态，是应该直接把MySQL进程强行重启，还是应该让他自己自行完成呢？为什么？