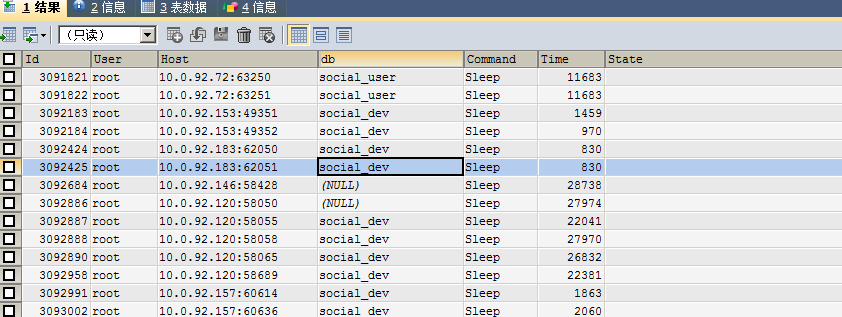
现象：执行sql语句一直卡顿。Waiting for table metadata lock

（我当时是执行，一条ddl语句 ALTER TABLE p\_sku ADD INDEX INDEX\_THIRDSKUID\_INVOICETYPE(third\_sku\_id, invoice\_type) COMMENT '三方skuid开票类型组合索引';）

解决过程：

展示执行的操作

SHOW PROCESSLIST ;



找出要终止执行的id，特别是出现Waiting for table metadata lock 然后kill掉。例如：KILL 3098535

Waiting for table metadata lock 是锁表情形。

SELECT \* FROM information\_schema.innodb\_trx 查看有未提交的事务

找到未提交的事务 事务线程id，然后kill掉trx\_mysql\_thread\_id

SELECT \* FROM performance\_schema.events\_statements\_current

找到对应的thread\_id 然后kill掉session

参考文档：<https://www.cnblogs.com/digdeep/p/4892953.html>

MySQL在进行alter table等DDL操作时，有时会出现Waiting for table metadata lock的等待场景。而且，一旦alter table TableA的操作停滞在Waiting for table metadata lock的状态，后续对TableA的任何操作（包括读）都无法进行，因为他们也会在Opening tables的阶段进入到Waiting for table metadata lock的锁等待队列。如果是产品环境的核心表出现了这样的锁等待队列，就会造成灾难性的后果。

造成alter table产生Waiting for table metadata lock的原因其实很简单，一般是以下几个简单的场景：

**场景一：长事物运行，阻塞DDL，继而阻塞所有同表的后续操作**

通过show processlist可以看到TableA上有正在进行的操作（包括读），此时alter table语句无法获取到metadata 独占锁，会进行等待。

这是最基本的一种情形，这个和mysql 5.6中的online ddl并不冲突。一般alter table的操作过程中（见下图），在after create步骤会获取metadata 独占锁，当进行到altering table的过程时（通常是最花时间的步骤），对该表的读写都可以正常进行，这就是online ddl的表现，并不会像之前在整个alter table过程中阻塞写入。（当然，也并不是所有类型的alter操作都能online的，具体可以参见官方手册：<http://dev.mysql.com/doc/refman/5.6/en/innodb-create-index-overview.html>）  
**处理方法：** kill 掉 DDL所在的session.

**场景二：未提交事物，阻塞DDL，继而阻塞所有同表的后续操作**

通过show processlist看不到TableA上有任何操作，但实际上存在有未提交的事务，可以在 **information\_schema.innodb\_trx**中查看到。在事务没有完成之前，TableA上的锁不会释放，alter table同样获取不到metadata的独占锁。

处理方法：通过 select \* from **information\_schema.innodb\_trx**\G, 找到未提交事物的sid, 然后 kill 掉，让其回滚。

**场景三：**

通过show processlist看不到TableA上有任何操作，在information\_schema.innodb\_trx中也没有任何进行中的事务。这很可能是因为在一个显式的事务中，对TableA进行了一个失败的操作（比如查询了一个不存在的字段），这时事务没有开始，但是失败语句获取到的锁依然有效，没有释放。从**performance\_schema.events\_statements\_current**表中可以查到失败的语句。

官方手册上对此的说明如下：

If the server acquires metadata locks for a statement that is syntactically valid but fails during execution, it does not release the locks early. Lock release is still deferred to the end of the transaction because the failed statement is written to the binary log and the locks protect log consistency.

也就是说除了语法错误，其他错误语句获取到的锁在这个事务提交或回滚之前，仍然不会释放掉。because the failed statement is written to the binary log and the locks protect log consistency 但是解释这一行为的原因很难理解，因为错误的语句根本不会被记录到二进制日志。

**处理方法：**通过**performance\_schema.events\_statements\_current**找到其sid, kill 掉该session. 也可以 kill 掉DDL所在的session.

总之，alter table的语句是很危险的(其实他的危险其实是未提交事物或者长事务导致的)，在操作之前最好确认对要操作的表没有任何进行中的操作、没有未提交事务、也没有显式事务中的报错语句。如果有alter table的维护任务，在无人监管的时候运行，最好通过**lock\_wait\_timeout**设置好超时时间，避免长时间的metedata锁等待。