

事务持续执行之谜：怎样找出对行记录上锁的 SQL？

原创 何文超 爱可生开源社区 2025年04月16日 22:30 上海

作者：何文超，分享 MySQL 和 OceanBase 相关技术博文。个人博客【CSDN | 雅俗数据库】

爱可生开源社区出品，原创内容未经授权不得随意使用，转载请联系小编并注明来源。

本文约 1500 字，预计阅读需要 3 分钟。



1. 故障背景

在数据库运行过程中，部分事务语句无法正常提交，相应的会话状态会在很长时间内保持活跃。然而，当我们使用 `show processlist` 命令进行检查时，往往难以获取到导致事务无法提交的异常会话 SQL 语句，这给故障排查和处理带来了极大的困难。

2. 故障复现

2.1 模拟两个会话操作

以下是两个会话的操作示例，用于模拟事务锁等待的情况：

```
-- 会话 1
mysql> begin;
mysql> delete from db02.order_info where id in (12,13);

-- 会话 2
mysql> begin;
```

```
mysql> update db02.order_info set create_time='2025-02-10 10:00:00' whereid=12;
-- 执行完处于奔住状态，超过 innodb_lock_wait_timeout 参数设定值，会超时回滚。
-- ERROR 1205 (HY000): Lock wait timeout exceeded; try restarting transaction

-- 设置会话级别锁等待超时参数，便于测试
mysql> set session innodb_lock_wait_timeout=3600;
mysql> update db02.order_info set create_time='2025-02-10 10:00:00' whereid=12;
```

2.2 检索 show processlist

使用以下 SQL 语句查询正在执行的 SQL 语句：

```
mysql> select * from information_schema.processlist where COMMAND <> 'Sleep';
```

ID	USER	HOST	DB	COMMAND	TIME	STATE
57	repl	10.186.63.118:36624	NULL	Binlog Dump GTID	2862216	Master
5	event_scheduler	localhost	NULL	Daemon	3011932	Waiting
376285	root	localhost	NULL	Query	67	updating
376271	root	localhost	NULL	Query	0	executing

4 rows in set (0.01 sec)

从查询结果中，我们并未找到导致 UPDATE 操作等待的事务语句。

3. 排查思路

3.1 查看未提交的事务

```
mysql> SELECT trx_id, trx_state, trx_started, trx_tables_locked, trx_rows_locked FROM informa
***** 1. row *****
      trx_id: 3600172 -- 刚刚运行的第二个语句事务 ID
      trx_state: LOCKWAIT -- 处于锁等待状态
      trx_started: 2025-04-02 14:20:34
      trx_tables_locked: 1 -- 锁了 1 张表
      trx_rows_locked: 1 -- 锁了 1 行
***** 2. row *****
      trx_id: 3600069 -- 刚刚运行的第一个语句事务 ID
      trx_state: RUNNING -- 获得锁的状态
      trx_started: 2025-04-02 14:18:18
      trx_tables_locked: 1
```

```
trx_rows_locked: 2
2 rows in set (0.00 sec)
```

3.2 查看等待锁的事务信息

```
mysql> SELECT wait_started, locked_table, waiting_trx_id, blocking_trx_id, sql_kill_block
***** 1. row *****
      wait_started: 2025-04-02 14:20:34 -- 等待锁开始的时间
      locked_table: `db02`.`order_info` -- 被锁定的表名（格式为数据库名.表名）
      waiting_trx_id: 3600172 -- 正在等待锁的事务 ID
      blocking_trx_id: 3600069 -- 持有锁从而阻塞其他事务的事务 ID
      sql_kill_blocking_connection: KILL 376283 -- 用于终止持有锁的连接的 SQL 语句
1 row in set (0.01 sec)
```

3.3 查询持有锁连接对应的 SQL 语句

```
mysql> SELECT a.thread_id,a.sql_text FROM performance_schema.events_statements_history a
+-----+-----+-----+
| thread_id | sql_text |
+-----+-----+-----+
| 376455 | select @@version_comment limit 1 |
| 376455 | begin |
| 376455 | delete from db02.order_info where id in(12,13) |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

通过以上查询，我们找到了 `delete from db02.order_info where id in(12,13)` 语句。在与业务侧确认该语句是否合理后，如果没问题可以使用 `KILL` 命令终止相应的连接。

4. 解决方案

杀掉锁源 SQL 对应的连接线程。

```
KILL 376283
```

5. 总结

5.1 锁相关表

在排查数据库事务锁等待问题时，主要涉及以下几个关键系统表：

表名	作用
information_schema.processlist	查看当前数据库中正在执行的 SQL 语句和会话状态信息
information_schema.innodb_trx	存储 InnoDB 存储引擎中未提交事务的详细信息，如事务 ID、事务状态、事务开始时间、锁定的表数量和行数量等
sys.innodb_lock_waits	记录等待锁的事务信息，包括等待开始时间、被锁定的表名、等待锁的事务 ID、持有锁的事务 ID 以及用于终止持有锁连接的 SQL 语句
performance_schema.events_statements_current	记录当前正在执行的 SQL 语句的相关信息
performance_schema.events_statements_history	存储线程执行的 SQL 语句历史记录
performance_schema.threads	包含线程的相关信息，可用于关联 PROCESSLIST_ID 和 THREAD_ID

5.2 排查 SQL

考虑到以上排查步骤较为繁琐，在生产故障紧急情况下，我们可以使用以下 SQL 语句进行快速排查：

```
SELECT
    a.THREAD_ID,
    a.SQL_TEXT,
    b.PROCESSLIST_ID ,
    DATE_FORMAT(c.trx_started, '%Y-%m-%d %H:%i:%s') AS transaction_start_time
FROM
    performance_schema.events_statements_history a
JOIN
    performance_schema.threads b ON a.THREAD_ID = b.THREAD_ID
JOIN
    information_schema.innodb_trx c ON b.PROCESSLIST_ID = c.trx_mysql_thread_id;
```

执行该 SQL 语句后，得到如下结果：

+	-----+	-----+	-----+
	THREAD_ID		SQL_TEXT

```
+-----+-----+
| 376457 | select @@version_comment limit 1 |
| 376457 | begin |
| 376457 | update db02.order_info set create_time='2025-02-10 10:00:00' where id=12 |
| 376457 | set session innodb_lock_wait_timeout=3600 |
| 376455 | select @@version_comment limit 1 |
| 376455 | begin |
| 376455 | delete from db02.order_info where id in(12,13) |
+-----+-----+
7 rows in set (0.00 sec)
```

结果集中各字段含义如下：

- **THREAD_ID**：MySQL 数据库内线程 ID。
- **SQL_TEXT**：当前线程正在执行的 SQL 语句的文本内容。
- **PROCESSLIST_ID**：数据库会话 ID，主要用于客户端连接的线程管理，例如可以使用 **KILL** 命令结合 **PROCESSLIST_ID** 终止某个客户端连接。
- **transaction_start_time**：事务开始时间。

5.3 相关参数

针对 InnoDB 存储引擎，可以通过调整 **innodb_lock_wait_timeout** 参数来处理锁等待超时报错问题，从而提升数据库并发性能。以下是 **lock_wait_timeout** 和 **innodb_lock_wait_timeout** 两个参数的详细对比：

对比项	lock_wait_timeout	innodb_lock_wait_timeout
适用范围	对所有存储引擎都适用，主要用于表级锁等待	仅针对 InnoDB 存储引擎内部锁等待
默认值	31536000 秒（即 1 年）	通常为 50 秒
作用机制	非 InnoDB 存储引擎操作请求锁被占用时进入等待，超设定值操作终止并报错	InnoDB 事务获取锁被占用时进入等待，超设定值事务自动回滚、释放部分锁资源并报错

本文关键字：#MySQL# #InnoDB# #锁#



推荐阅读

故障分析 | MySQL 8.0 中多字段虚拟列引发的宕机

故障分析 | 如何解决由触发器导致 MySQL 内存溢出？

故障分析 | 查询 ps.data_locks 导致 MySQL hang 住

故障分析 | TCP 缓存超负荷导致的 MySQL 连接中断

SQLC 是一款全方位的 SQL 质量管理平台，
覆盖开发至生产环境的 SQL 审核和管理。



支持主流开源、商业、国产数据库。
为开发者、DBA、运维人员提供流程自动化
能力，提升上线效率，提高数据质量。



请添加小助手加入
SQLC 技术交流群


SQLC 企业版
商务咨询/预约演示



✨ Github : <https://github.com/actiontech/sqlc>

📖 文档 : <https://actiontech.github.io/sqlc-docs/>

🌐 官网 : <https://opensource.actionsky.com/sqlc/>

 微信群：请添加小助手加入 ActionOpenSource

 商业支持：<https://www.actionsky.com/sql>



MySQL 231 InnoDB 62 锁 36

MySQL · 目录

上一篇

MySQL 如何实现安全连接？

下一篇

实现一个 MySQL 配置对比脚本需要考虑哪些
细节？