

告别排队，Trae订阅上线，首月Pro仅\$3，开发无忧

立即体验

稀土掘金 首页 ▾

探索稀土掘金



登录

# 如何迅速并识别处理MDL锁阻塞问题

华为云开发者联盟 2025-01-22 78 阅读9分钟

关注

告别排队，Trae订阅上线，首月Pro仅\$3，开发无忧

摘要：TaurusDB推出MDL锁视图功能，帮助用户迅速识别并处理MDL锁阻塞问题，从而有效减少对业务的负面影响，提升数据库管理效率。

本文分享自华为云社区 [《【华为云MySQL技术专栏】TaurusDB MDL锁视图》](#)，作者：GaussDB 数据库。



## 一、背景

数据库中的元数据锁（MDL，Metadata Lock），用来保护表元数据信息的一致性。用户对表进行读写操作或结构变更时，系统会添加不同类型的MDL锁。当客户业务设计不合理，有部分事务长时间持有MDL锁时，可能会阻塞其他会话获取相应的MDL锁。此时，用户使用执行‘SHOW PROCESSLIST’命令，通常会看到多个会话处于“Waiting for metadata lock”状态。但由于无法明确各个会话ID之间的关联，往往无法快速找到导致大量MDL锁等待的根源，使得用户不得不盲目地Kill大量可疑的会话，甚至直接重启实例来快速恢复业务，这种做法无疑增加了解决问题的成本，对业务产生较大影响。

因此，自社区MySQL 5.7版本之后，在PERFORMANCE\_SCHEMA库中新增了METADATA\_LOCKS表，用于记录系统中MDL锁的状态信息，但是需要客户启用Performance

Schema 性能分析监控插件。启用后，Performance Schema 会收集大量的性能数据，包括SQL语句执行情况和实例内的锁状态信息等，这会对 MySQL 实例的性能产生一定的负担，尤其在高并发的生产环境中，性能开销更为明显。

鉴于此，TaurusDB推出MDL锁视图功能，帮助用户迅速识别并处理MDL锁阻塞问题，从而有效减少对业务的负面影响，提升数据库管理效率。

## 二、MDL锁阻塞场景分析

我们以表1中的MDL锁阻塞案例来介绍MDL锁视图的使用场景

时间\会话ID	SESSION 2	SESSION 3	SESSION 4	SESSION 5
T1	BEGIN; SELECT * FROM t1;			
T2		BEGIN; SELECT * FROM t2;		
T3			TRUNCATE TABLE t2; (blocked)	
T4				SELECT * FROM t2; (blocked)

表1 MDL锁阻塞案例

由于SESSION 3存在一个长事务未提交，会一直持有t2表的MDL\_SHARED\_READ（SR）类型锁。当SESSION 4对表t2执行TRUNCATE操作时，需要获取MDL-EXCLUSIVE（X）锁，但由于MDL锁类型和SR锁不兼容而被阻塞。

随后，SESSION 5的DML操作在添加SR类型的锁时，发现MDL锁等待队列中有比SR类型的锁优先级更高的X锁在等待，所以SESSION 5的SR锁请求也会处于等待状态，详细原因可参考《[【华为云MySQL技术专栏】TaurusDB新特性解读：非阻塞DDL](#)》。

用户发现DDL和DML操作都被阻塞后，执行SHOW PROCESSLIST查看原因。在SHOW PROCESSLIST的信息中，只能看到如图1中的结果：

Id	User	Host	db	Command	Time	State	Info
2	root	localhost	test	Sleep	73		NULL
3	root	localhost	test	Sleep	63		NULL
4	root	localhost	NULL	Query	35	Waiting for table metadata lock	truncate table test.t2
5	root	localhost	test	Query	17	Waiting for table metadata lock	select *from test.t2
6	root	localhost	test	Query	0	starting	

图 1 SHOW PROCESSLIST结果

- SESSION 4执行TRUNCATE操作时，被其他SESSION持有的table metadata lock阻塞；
- SESSION 5执行SELECT操作时，也同样被阻塞；
- 无法确定哪个会话（2或3? ）阻塞了SESSION 4和SESSION 5；

此时，如果业务盲目地去kill其他会话（2或3），可能会影响其他不相关的业务，从而加大问题处理的成本。在实际的生产业务中，可能有更多的会话，用户从成百上千的会话信息中几乎无法找到导致MDL锁等待的根源，只能盲目地Kill大量的会话或者重启实例来快速恢复。而且用户在事后也无法定位到根因，从源头杜绝此类问题的再次发生。刚好，TaurusDB的MDL锁视图功能在这个时候就可以发挥作用。

### 三、MDL锁视图介绍

---

TaurusDB的MDL锁视图以系统表的形式呈现，该表位于INFORMATION\_SCHEMA库下，表名为METADATA\_LOCK\_INFO。其中每一行的信息表示一个会话持有或正在等待的MDL锁信息。

每个字段的具体含义，如表2所示：

序号	字段名	字段说明
1	THREAD_ID	会话ID，与show processlist结果中的ID列对应
2	LOCK_STATUS	MDL锁的两种状态。 1) PENDING：表示会话正在等待该MDL锁。 2) GRANTED：表示会话已获得该MDL锁。
3	LOCK_MODE	加锁的模式，共有10种类型的锁，如MDL_SHARED、MDL_EXCLUSIVE、MDL_SHARED_READ、MDL_SHARED_WRITE等。
4	LOCK_TYPE	MDL锁的类型，如Table metadata lock、Schema metadata lock、Global read lock等。
5	LOCK_DURATION	MDL锁范围，取值如下： <b>MDL_STATEMENT</b> ：表示语句级别，在语句结束后即可释放； <b>MDL_TRANSACTION</b> ：表示事务级别，在事务提交后才能释放； <b>MDL_EXPLICIT</b> ：表示GLOBAL级别，在执行FLUSH TABLE WITH READ LOCK等SQL时会添加，在显示执行UNLOCK TABLES时释放。
6	TABLE_SCHEMA	数据库名，对于Global read lock类型的MDL锁，该值为空。
7	TABLE_NAME	表名，对于Global read lock或Schema metadata lock类型的MDL锁，该值为空。

掘金技术社区 @ 华为云开发者联盟

表2 MDL锁视图表字段含义

此表的查询结果中，同一个会话可能持有多行MDL锁的相关信息。主要有以下几方面的原因：

1) 当执行涉及多张表的连表查询时，会给每一个表添加MDL\_SHARED\_READ模式的MDL锁。

2) 事务级别的MDL锁，只有在事务结束时才会释放。因此，当一个事务涉及多张表的DML操作时，这个会话会同时持有多个MDL锁，直到事务结束。

3) 在DDL语句的执行过程中，需要添加多种类型的MDL锁。例如，在添加列的DDL语句中，可能会添加Backup lock,Global read lock,Schema metadata lock,Table metadata lock，并且在不同阶段对锁的模式进行升/降级。



## 四、MDL锁视图使用方法

针对表格1中的MDL锁阻塞场景，用户可以执行以下SQL语句：SELECT \* FROM INFORMATION\_SCHEMA.METADATA\_LOCK\_INFO;

再结合SHOW PROCESSLIST的输出结果，快速定位到问题根因。

在图2元数据锁视图的结果信息中，我们应该从PENDING状态的会话开始入手。

THREAD_ID	LOCK_STATUS	LOCK_MODE	LOCK_TYPE	LOCK_DURATION	TABLE_SCHEMA	TABLE_NAME
2	GRANTED	MDL_SHARED_READ	Table metadata lock	MDL_TRANSACTION	test	t1
3	GRANTED	MDL_SHARED_READ	Table metadata lock	MDL_TRANSACTION	test	t2
4	GRANTED	MDL_INTENTION_EXCLUSIVE	Global read lock	MDL_STATEMENT		
4	GRANTED	MDL_INTENTION_EXCLUSIVE	Schema metadata lock	MDL_TRANSACTION	test	
4	PENDING	MDL_EXCLUSIVE	Table metadata lock		test	t2
5	PENDING	MDL_SHARED_READ	Table metadata lock		test	t2

图 2 元数据锁视图结果

根据MDL锁等待的TABLE\_SCHEMA和TABLE\_NAME信息，找到其他THREAD ID下，具有相同库名和表名且状态是GRANTED的MDL锁信息。这个THREAD ID即是造成锁等待的会话。

基于以上原理，我们可以看出：

- 会话4在等待获取test库t2表的MDL\_EXCLUSIVE模式的元数据锁；
- 会话5在等待获取test库t2表的MDL\_SHARED\_READ模式的元数据锁；
- 会话3持有test库t2表t2的MDL锁，该MDL锁为事务级别，只要session 3 的事务不提交，session 4和5便会一直阻塞。

所以，通过MDL锁视图，我们只需要在会话3中执行Rollback或者Commit，便可以让业务继续运行。虽然MDL锁视图可以帮助定位到导致大量MDL锁等待的根源，但是当会话较多时，表中很多不相关的MDL锁信息查看起来也会耗费大量时间，这里我们提供一个可以快速查找到阻塞会话的SQL。在发生问题时，只要执行一下这个语句，就可以迅速找到需要kill的会话。

▼ vbnet

体验AI代码助手

复制代码

```
1 SELECT f.processlist_id, p.Info AS sql_info
2 FROM (
3     SELECT DISTINCT c.blocking_processlist_id AS processlist_id
4     FROM (
5         SELECT DISTINCT b.THREAD_ID AS blocking_processlist_id
6         FROM information_schema.metadata_lock_info a
```

```

7      JOIN information_schema.metadata_lock_info b
8      ON a.TABLE_SCHEMA = b.TABLE_SCHEMA
9      AND a.TABLE_NAME = b.TABLE_NAME
10     AND a.lock_status = 'PENDING'
11     AND b.lock_status = 'GRANTED'
12     AND a.THREAD_ID <> b.THREAD_ID
13 ) c
14 WHERE c.blocking_processlist_id NOT IN (
15     SELECT DISTINCT d.THREAD_ID AS blocked_processlist_id
16     FROM information_schema.metadata_lock_info d
17     JOIN information_schema.metadata_lock_info e
18     ON d.TABLE_SCHEMA = e.TABLE_SCHEMA
19     AND d.TABLE_NAME = e.TABLE_NAME
20     AND d.lock_status = 'PENDING'
21     AND e.lock_status = 'GRANTED'
22     AND d.THREAD_ID <> e.THREAD_ID
23 )
24 ) f
25 JOIN information_schema.processlist p ON processlist_id = p.Id;

```

## 五、原理解析

基于对《[【华为云MySQL技术专栏】TaurusDB MDL实现机制解析](#)》中MDL锁相关数据结构和MDL锁添加、释放流程的分析，TaurusDB在INFORMATION\_SCHEMA库下添加了MDL锁视图。

在TaurusDB的内部架构中，每一个用户连接有一个THD（Thread Handler，线程处理器）对象，这些THD对象统一由Global\_THD\_manager结构体进行管理。如图3所示，每个THD对象关联了一个MDL\_context实例，这个实例提供了线程级的MDL锁操作接口，包括申请、释放MDL锁以及锁的升降级。

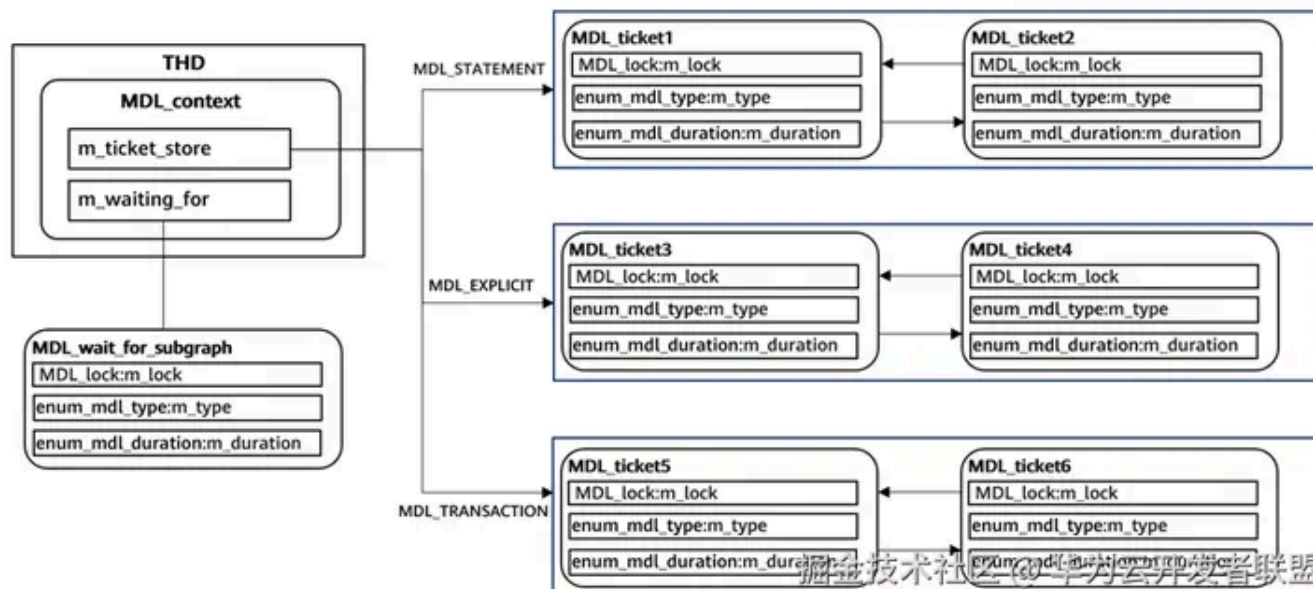


图 3. MDL锁基本概念图

其中，THD中的MDL\_context也有两个变量m\_ticket\_store和m\_waiting\_for用于维护会话持有和等待的MDL锁的信息。

- 1. m\_ticket\_store: 用来存储当前线程获取的所有MDL\_ticket。为了提升搜索效率，根据MDL锁的持续时间（语句执行时间段，事务执行时间段和显示指定时间段）将其划分为三个链表，在需要获取MDL锁前，会先在这些链表内查询是否已经获取到了相同的或这是更强类型的MDL锁，如果搜索不到继续获取MDL锁。
- 2. m\_waiting\_for: 用来存储当前线程正在等待的MDL锁，一个线程同一时刻只能等待一种类型的MDL锁。

每一个链表中的MDL\_ticket对象是每个线程已经获取到的MDL锁或者是要请求的MDL锁的详细信息，其结构体内包含MDL锁的模式（enum\_mdl\_type）、MDL锁的持续时间（enum\_mdl\_duration）和MDL锁对象（MDL\_lock）。其中，MDL\_lock由MDL\_key唯一标识。MDL\_key是一个三元组，由命名空间（enum\_mdl\_namespace）、库名和对象名组成。

因此，在用户查询MDL锁视图时，实现流程如图4所示，只需要遍历所有会话THD的MDL\_context对象，根据其m\_ticket\_store链表中的每一个MDL\_ticket对象构造出处于GRANTED状态的MDL锁信息。同理，通过m\_waiting\_for对象获取到处于PENDING状态的MDL锁信息。最后，将结果集返回给客户端展示即可。

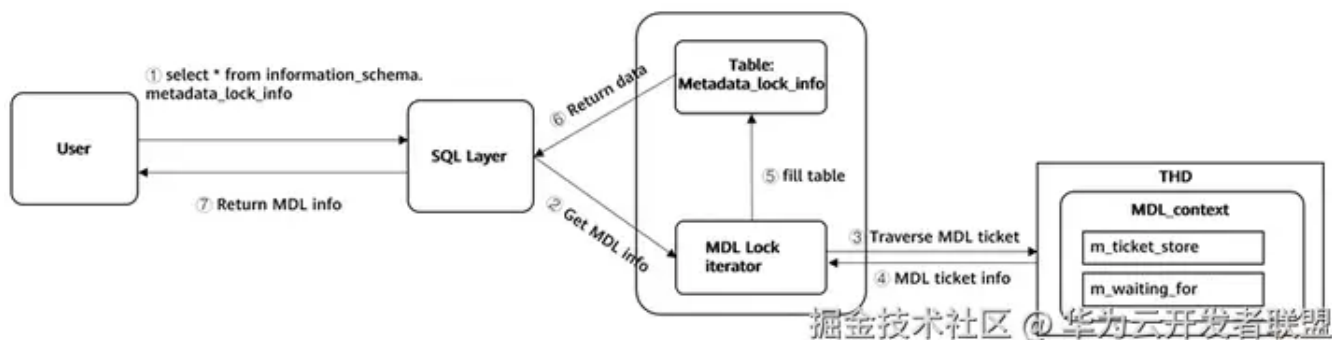


图 4 MDL锁视图实现流程

其中，i\_s\_metadata\_lock\_info\_fill\_table和List\_THD\_MDl\_tickets为核心函数，用来实现遍历Global\_THD\_manager中的THD并从其MDL\_context中构造当前会话的持有和等待的MDL锁信息。

```
1  i_s_metadata_lock_info_fill_table() {
2      //对系统中每一个THD执行List_THD_MDL_tickets函数找到持有和等待的MDL锁
3      Global_THD_manager::get_instance()->do_for_all_thd_copy(List_THD_MDL_tickets)
4  }
5
6  List_THD_MDL_tickets() {
7      // 获取当前THD的MDL_context:
8      MDL_context &mdl_ctx = inspect_thd->mdl_context;
9      // 获取当前THD持有的MDL锁
10     const MDL_ticket_store &m_ticket_store = mdl_ctx.get_mdl_ticket_store();
11     // 遍历每个m_ticket_store的三个作用范围内的MDL_ticket
12     for (int i = 0; i < MDL_DURATION_END; i++) {
13         MDL_ticket_store::List_iterator it = m_ticket_store.list_iterator(duration);
14         lock_extras.duration = duration;
15         while ((ticket = it++)) {
16             enum_mdl_duration duration = (enum_mdl_duration)(i);
17             // 根据MDL_ticket中的信息填充到MDL锁视图中
18             fill_row_callback(ticket, &lock_extras, args);
19         }
20     }
21     // 获取当前THD等待的MDL锁
22     ticket = dynamic_cast<const MDL_ticket *>(mdl_ctx.get_m_waiting_for());
23
24     if (ticket != nullptr) {
25         // 填充MDL锁的额外信息，PENDING状态和作用范围
26         lock_extras.lock_status = MDL_ticket::PENDING;
27         lock_extras.duration = ticket->get_duration();
28         // 根据MDL_ticket中的信息填充到MDL锁视图中
29         fill_row_callback(ticket, &lock_extras, args);
30     }
31 }
```

## 六、总结

TaurusDB的MDL锁视图INFORMATION\_SCHEMA.METADATA\_LOCK\_INFO，可以在不开启Performance Schema性能监控插件时，获取到实例系统中所有MDL锁的持有和等待状态。熟练的使用MDL锁视图，可以帮助用户快速地定位和分析导致大量 MDL 锁等待的根本原因，还能够根据分析结果进行迅速有效的处理，解决MDL长时间锁等待问题，并且不会因为依赖Performance Schema性能监控插件而对系统性能产生任何影响。

关注“GaussDB数据库”公众号，了解更多动态

[点击关注，第一时间了解华为云新鲜技术~](#)



标签： 数据库

本文收录于以下专栏

◀

1 / 2

▶



程序员之家

专栏目录

技术资讯分享，欢迎投稿、交流~

78 订阅 · 3.0k 篇文章

订阅

评论 0



登录 / 注册 即可发布评论!

暂无评论数据

目录

收起 ^

- 一、背景
- 二、MDL锁阻塞场景分析

### 三、MDL锁视图介绍

### 四、MDL锁视图使用方法

### 五、原理解析

### 六、总结

## 相关推荐

史上最全MySQL各种锁详解

30k阅读 · 97点赞

深入分析MySQL：什么是锁？如何解决幻读问题？

254阅读 · 4点赞

mysql创建索引导致死锁，数据库崩溃，mysql的表级锁之【元数据锁（meta data lock，MDL）】全解


716阅读 · 1点赞

什么情况下 MySQL 连查询都能被阻塞？

2.4k阅读 · 1点赞

## 为你推荐

### 实例带你了解GaussDB数据库的LOCK TABLE

华为云开发者联盟 1年前  360  1  评论

数据库 后端 SQL

### 一次性全讲透GaussDB（DWS）锁的问题

华为云开发者联盟 1年前  297  点赞  评论

数据库 后端 华为

### 关于加索引把数据库搞崩这件事

Maskvvv 1年前  645  5  评论

MySQL

### MYSQL原理、设计与应用（四）

张子栋 4月前  97  2  评论

后端 数据库

### MySQL锁总结

菜狗大作战 3年前  269  点赞  评论




MySQL 后端

### MySQL那些“锁”事，你听烦了吗？

半亩方塘立身 1年前  1.7k  10  评论

MySQL 数据库 后端

## mysql---MySQL\_MetaData Lock

后端程序员Aska 1月前  1.4k  17  1

后端

## Mysql读写锁保姆级图文教程

华为云开发者联盟 3年前  1.8k  2  评论

数据库

## 几个常见而严重的 MySQL 问题分析

编程学习网 4年前  1.2k  5  评论

MySQL

## 几个常见而严重的 MySQL 问题分析

编程学习网 4年前  1.1k  1  评论

MySQL

## 海山数据库(He3DB)源码解读：海山PG 死锁处理实现

Xiaye\_DB 10月前  43  点赞  评论

数据库

## MySQL 数据库事务隔离性的实现

华为云开发者联盟 4年前  942  9  1

数据库 后端

## INSERT...SELECT语句对查询的表加锁吗

GreatSQL 1年前  522  点赞  评论

数据库

## MySQL (/八) MySQL锁

IT\_wtz 3年前  323  1  评论

MySQL 后端

## 十一，MySQL锁机制详解

normaling 1月前  46  点赞  评论

MySQL