1. 现象描述

最近在某项目中发现有运行慢的报表在查询过程中，对查询中的某张表创建索引时，出现等待现象。观察processlist表中看到看到创建索引时，申请metadata-lock锁而处理等待中。下面的测试完全模拟预发布真实情况进行测试。

1. 环境描述

数据库版本：MySQL5.7.17 RDS环境

1. 准备测试数据

|  |
| --- |
| creata database test;  use test;  DROP TABLE xs;  DROP TABLE class;  CREATE TABLE xs(xh INT AUTO\_INCREMENT PRIMARY KEY,NAME VARCHAR(20),age INT,class INT);  INSERT INTO xs(NAME,age,class) VALUES('zhang.san',23,1);  INSERT INTO xs(NAME,age,class) VALUES('li.shi',28,2);  INSERT INTO xs(NAME,age,class) VALUES('wang.wu',38,1);  INSERT INTO xs(NAME,age,class) VALUES('zhao.liu',41,2);  CREATE TABLE class(id INT AUTO\_INCREMENT PRIMARY KEY,NAME VARCHAR(20));  INSERT INTO class(NAME) VALUES('A');  INSERT INTO class(NAME) VALUES('B');  DELIMITER $$  USE `test`$$  DROP PROCEDURE IF EXISTS `proc\_xs\_sele`$$  CREATE DEFINER=`root`@`192.168.88.39` PROCEDURE `proc\_xs\_sele`(p\_times INT )  BEGIN  DECLARE n\_times INT;  DECLARE n\_rec INT DEFAULT 0;  SET n\_times=p\_times;  WHILE n\_times>0 DO  SELECT COUNT(0) INTO n\_rec FROM xs a,class b WHERE a.`class`=b.`id`;  SET n\_times=n\_times-1;  END WHILE;  END$$  DELIMITER ; |

1. 模拟查询阻塞DDL

会话1：模拟一直在查询

|  |
| --- |
| SET SESSION TRANSACTION ISOLATION LEVEL READ UNCOMMITTED;  START TRANSACTION;  CALL proc\_xs\_sele(100000000);  COMMIT; |

会话2：模拟DDL操作

|  |
| --- |
| SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ ;  START TRANSACTION;  CREATE INDEX idx\_xs\_u1 ON xs(NAME); |

会话3：查询事务及进程状态

|  |
| --- |
| SELECT \* FROM information\_schema.INNODB\_TRX;  SELECT \* FROM information\_schema.INNODB\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  SELECT \* FROM information\_schema.processlist WHERE db='test'; |

说明：在会话2创建索引期间，通过会话3查询进程及事务状态如下图



会话2在等待获取表xs的元数据锁而处理等待中：“Waiting for table metadata lock”

需要注意，此处会话1查询同时出现在innodb\_trx，processlist表中。

1. 模拟查询不阻塞DDL

会话1：模拟一直在查询，首先回滚或提交之前的事务，然后运行查询，该查询不以事务形式工作。在操作前先回滚之前的事务。

|  |
| --- |
| ROLLBACK;  CALL proc\_xs\_sele(100000000); |

会话2：模拟DDL操作

|  |
| --- |
| SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ ;  START TRANSACTION;  DROP INDEX idx\_xs\_u1 ON xs;  CREATE INDEX idx\_xs\_u1 ON xs(NAME); |

会话3：查询事务及进程状态

|  |
| --- |
| SELECT \* FROM information\_schema.INNODB\_TRX;  SELECT \* FROM information\_schema.INNODB\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  SELECT \* FROM information\_schema.processlist WHERE db='test'; |

说明：会话2可以正常创建索引。在这个示例中，会话1中一直在运行的查询，在innodb\_trx表中看不到有事务信息。

1. 原因分析

当一个查询放在start transaction;或begin;中，会对事务中的查询语句中的所有参数查询的表上加表级别“table metadata lock”锁。这样对查询中的任何一张表进行DDL操作均会阻塞，因为所有表的DDL操作只有申请成功“table metadata lock”锁之后才会继续进行操作。

通过测试了MySQL中三种事务隔离级别均是这种工作方式，只要查询在事务中，会就会上元数据锁。同时也测试了不同事务隔离级别混用的情况也是一样的。

另外，发现了一个问题，在会话1获取了元数据锁时，在MySQL的数据字典表innodb\_locks中没有记录，同时在会话2处理等待时，也没有在视图innodb\_lock\_waits中写入数据。只在processlist中能够看到一些有锁的痕迹。如下图：



我个人理解是，在事务中的查询增加元数据锁为了确保事务运行过程中不会因表结构或索引变动影响当前的语句运行而增加的。但是这样做对并发的影响非常大。这也是MySQL数据库锁机器不完善的一个地方。相信以后版本会不断完善的。

1. 解决办法

（1）在发起查询时，不要开启事务，这样查询中所有的表不会增加元数据锁，也不会阻塞DDL操作。

（2）建议所有会话都统一使用MySQL的已提交读（READ COMMITTED）事务隔离级别，不要混用不同事务隔离级别，避免不同事务隔离级别混用产生的问题。