MySQL5.7事务隔离级别对并发影响测试

编制人：马飞

日期：2018-6-11~12

# 一、概述

本文争对目前益海嘉里项目线上更新阻塞比较多的情况进行原因排查。下面分别模拟一张大表，分两种隔离级别进行测试，每种场景分别对表上的查询条件有索引和无索引的情况进行测试。

# 二、测试环境

|  |  |
| --- | --- |
| 操作系统 | CentOS release 6.8 (Final) |
| 数据库 | Server version: 5.7.18 MySQL |

# 三、准备数据

|  |
| --- |
| creata database test;  use test;  DROP TABLE xs;  CREATE TABLE xs(xh INT not null,NAME VARCHAR(20),age INT,class varchar(1));  DELIMITER $$  USE `test`$$  DROP PROCEDURE IF EXISTS `proc\_xs\_ins`$$  CREATE PROCEDURE `proc\_xs\_ins`(p\_times INT )  BEGIN  DECLARE n\_times INT;  DECLARE n\_rec INT DEFAULT 0;  SET n\_times=p\_times;  truncate table xs;  WHILE n\_times>0 DO  INSERT INTO xs(xh,NAME,age,class) VALUE(n\_times,CONCAT('zhang',n\_times), FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  SET n\_times=n\_times-1;  END WHILE;  END$$  DELIMITER ; |

# 四、测试场景

## 4.1 可重复性读

### 4.1.1 无索引插入对DML影响

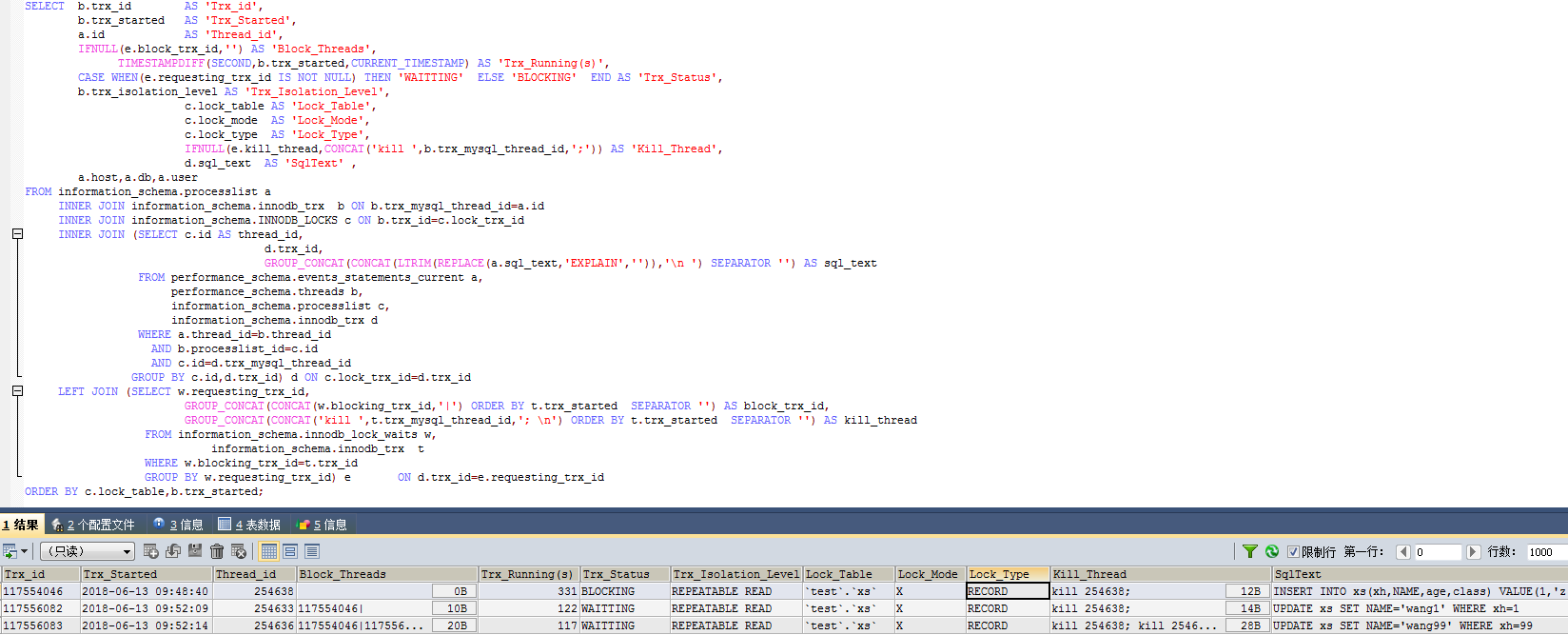
#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  truncate table xs;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：插入一条相同XH的数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条不同XH的数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(2,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2，会话3，在事务表中可同时查询出三条事务信息，没有锁定及阻塞信息。  2.在可重复性读隔离级别下当XH无索引时，插入操作不会阻塞其它插入操作。 |

#### （2）对更新影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：更新与“会话1”XH相同的数据**，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话3：更新与“会话1”XH不同的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang99' WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有3条记录，锁定表3条记录，锁等待3行记录  2.在可重复性读隔离级别下,当XH无索引时，插入操作会阻塞其会话对该表的更新操作。  3.在可重复性读隔离级别下,当XH无索引时，对整张表的所有记录加了X锁。 |

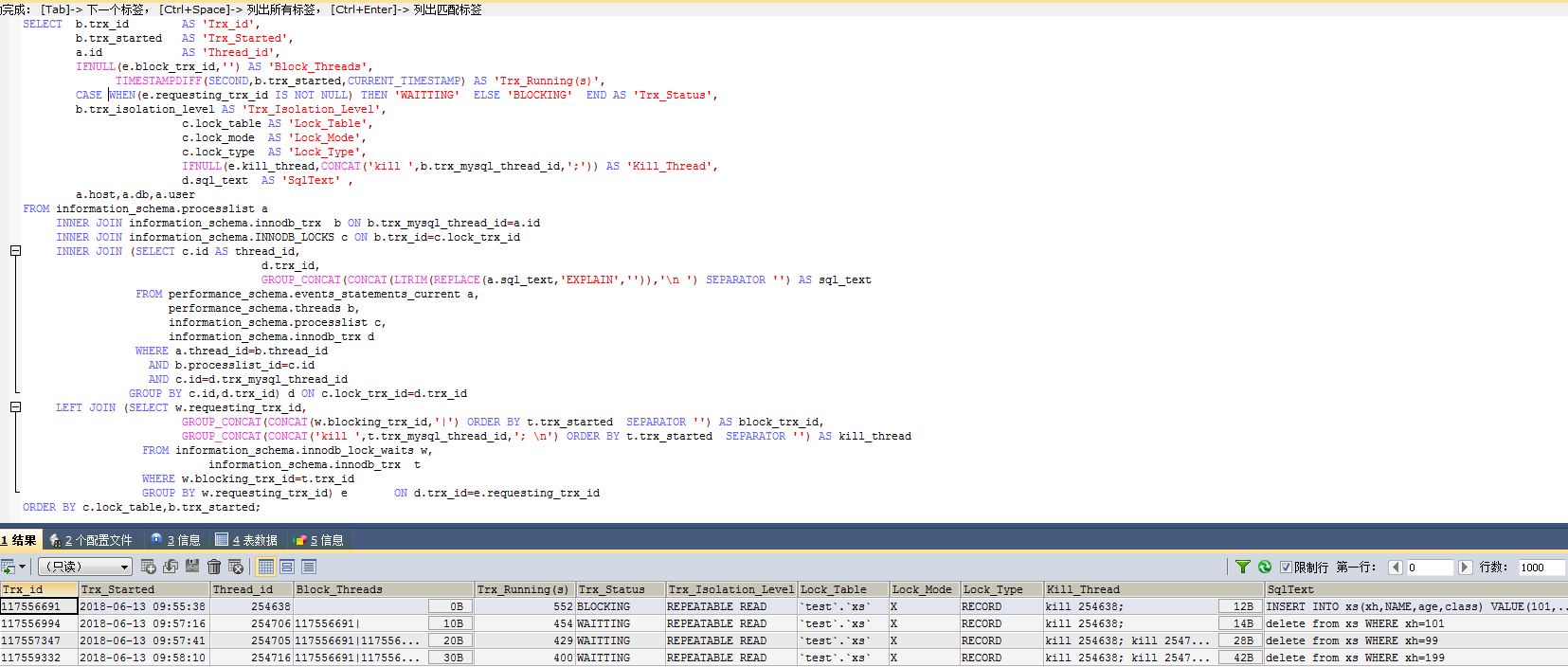
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  CALL proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：删除一条还没有提交的记录**，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=101;  *#会话3：删除一条已存在数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=99;  *#会话4：删除一条不存在数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=199;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,会话4，在事务表中有4条记录，锁定表4条记录，锁等待6行记录  2.在可重复性读隔离级别下，当XH无索引时，插入操作会阻塞其会话对该表的删除操作。  3.在可重复性读隔离级别下，当XH无索引时，对整张表的所有记录加了X锁。 |

锁等待查询：

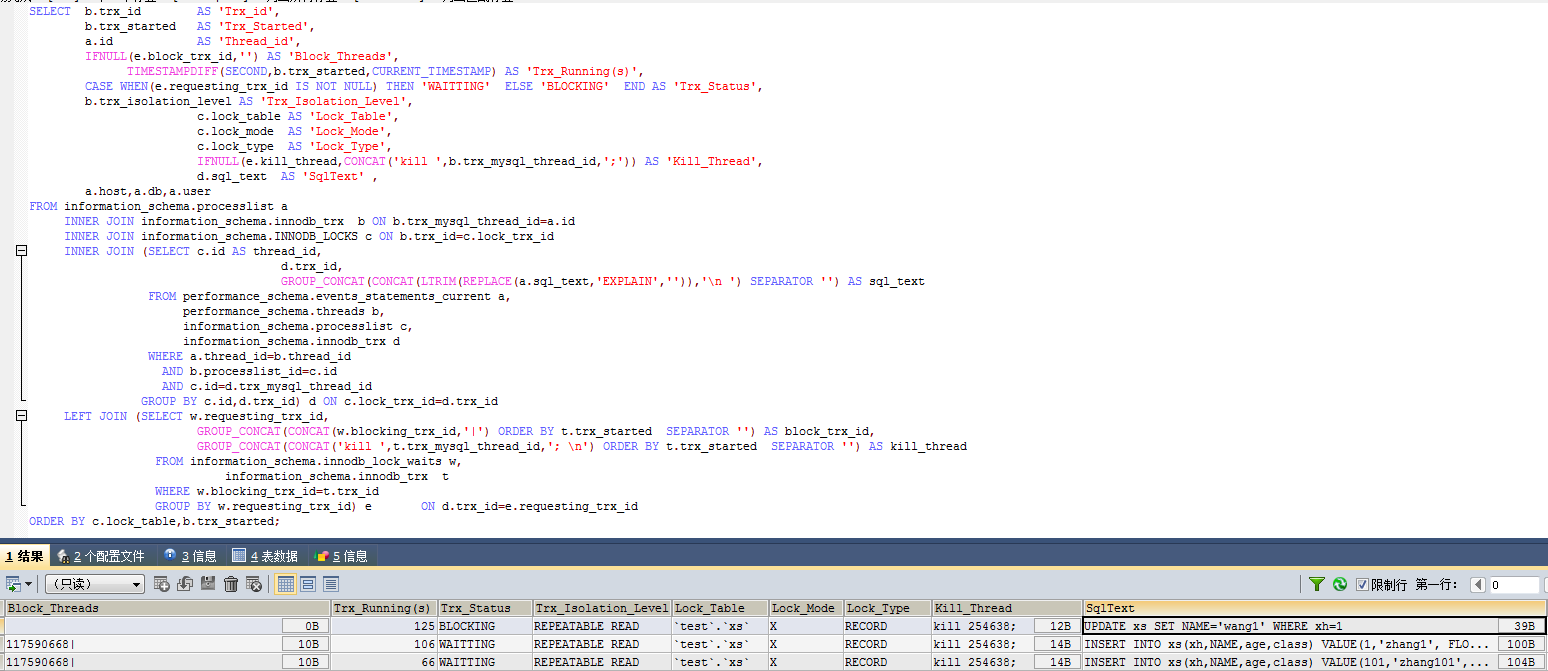


### 4.1.2 无索引更新对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条XH相同的数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有3条记录，锁定表3条记录，锁等待2行记录  2.在可重复性读隔离级别下，当XH无索引时，更新操作会阻塞其会话对该表的插入操作。  3.在可重复性读隔离级别下，当XH无索引时，对整张表了X锁，新旧值均无法插入。 |

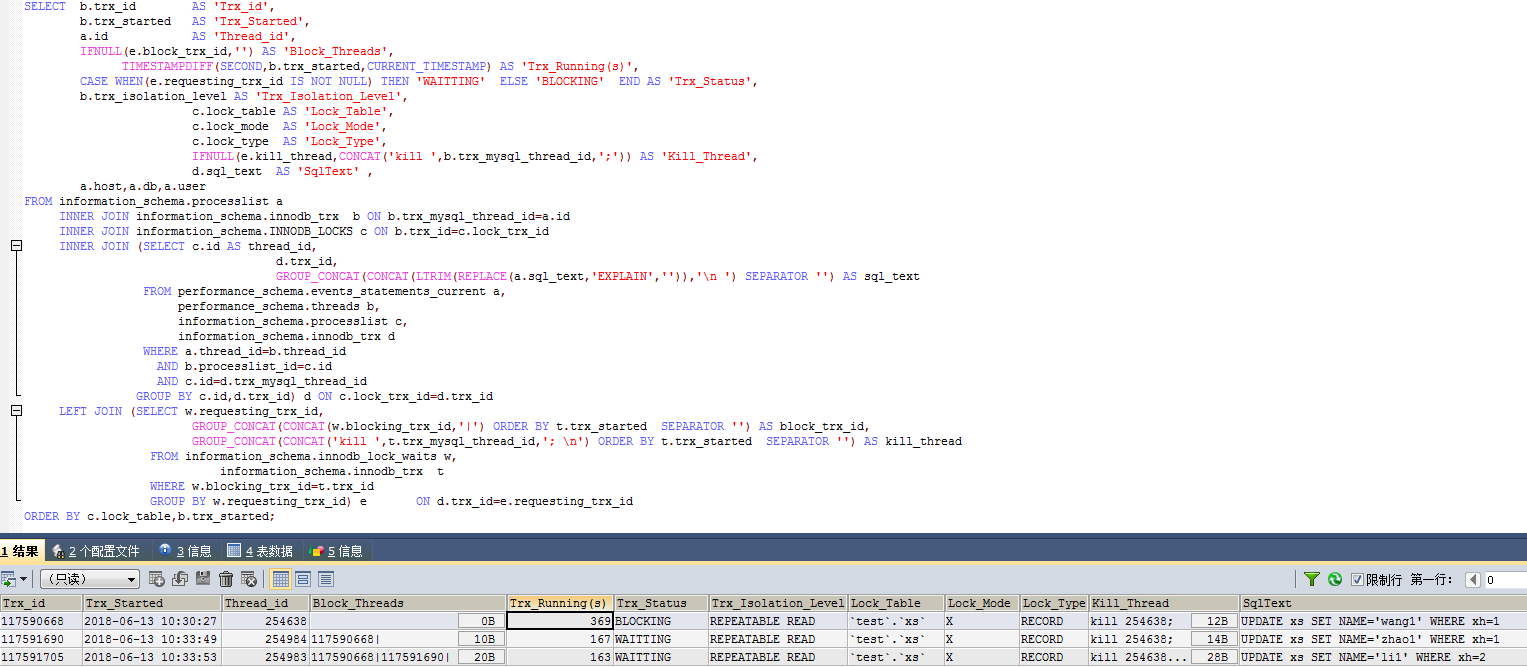
锁等待查询：



#### （2）对更新影响测试

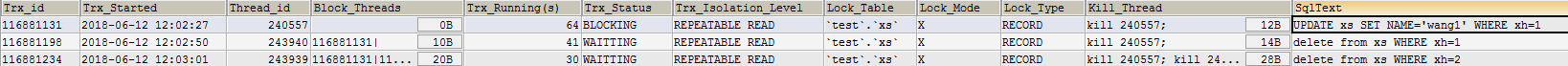
|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有3条记录，锁定表3条记录，锁等待3行记录  2.在可重复性读隔离级别下，当XH无索引时，更新操作会阻塞其会话对该表的更新操作。  3.在可重复性读隔离级别下，当XH无索引时，对整张表了X锁，其它会话无法更新该表。 |

锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有三条记录，锁定表4条记录，锁等待3行记录  2.在可重复性读隔离级别下当XH无索引时，更新操作会阻塞其会话对该表的删除操作。  3.在可重复性读隔离级别下为了防止发生幻读，对整张表了X锁，其它会话无法删除该表记录。 |

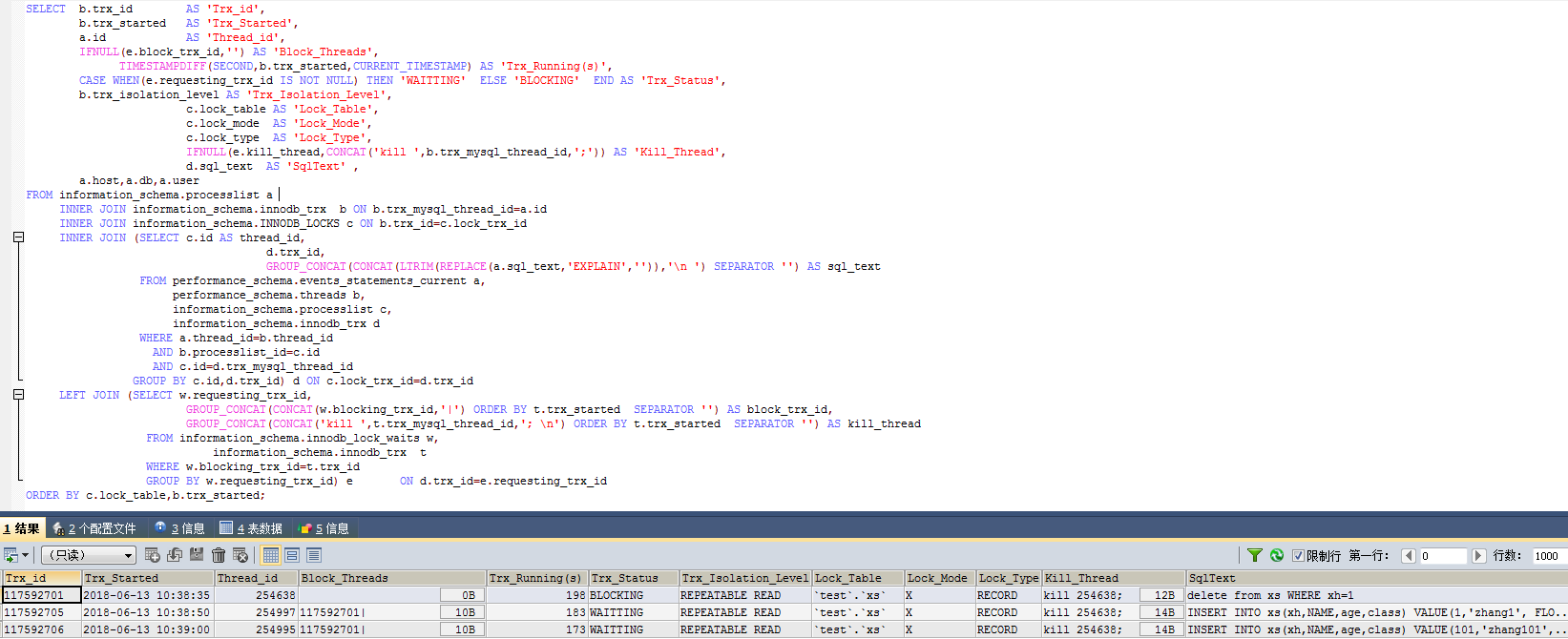


### 4.1.3 无索引删除对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有3条记录，锁定表3条记录，锁等待2行记录  2.在可重复性读隔离级别下，当ID无索引时，删除操作会阻塞其会话对该表的插入操作。  3.在可重复性读隔离级别下，当ID无索引时，对整张表了X锁，新旧值均无法插入。 |

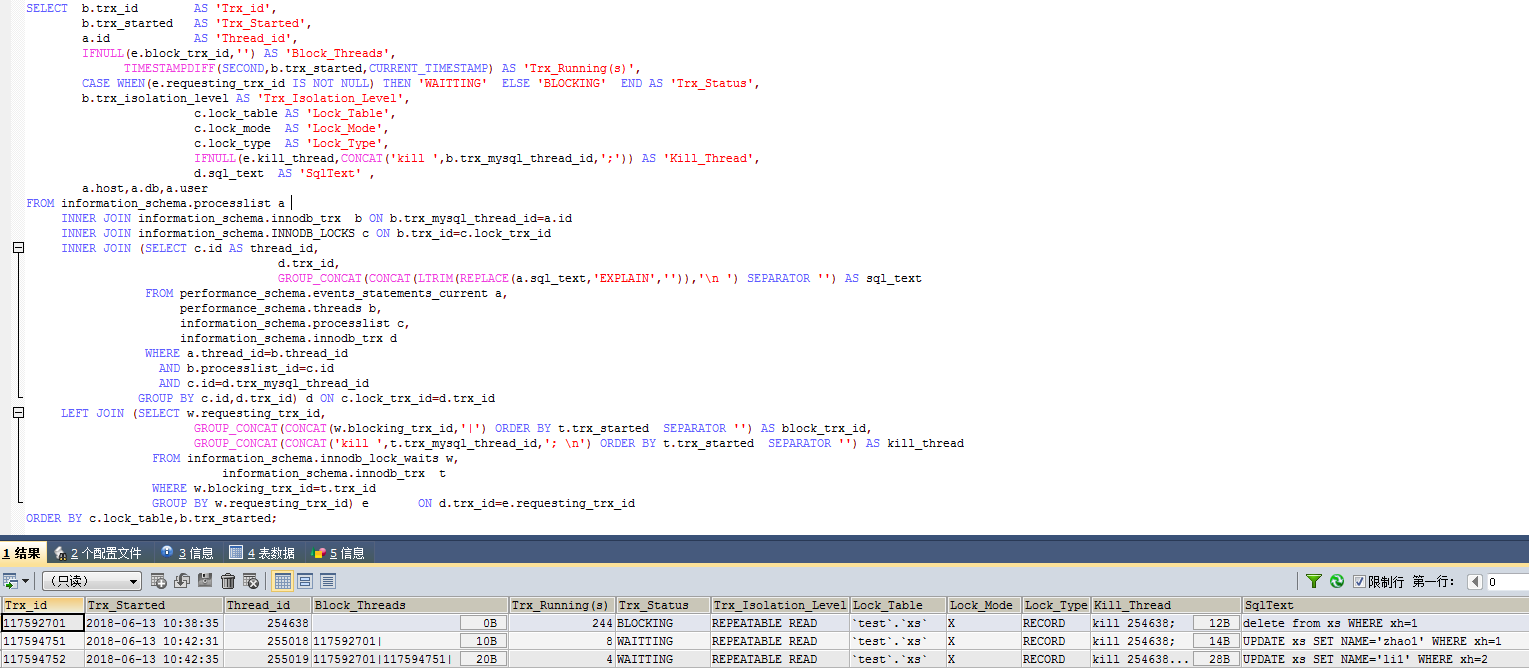
锁等待查询：



#### （2）对更新影响测试

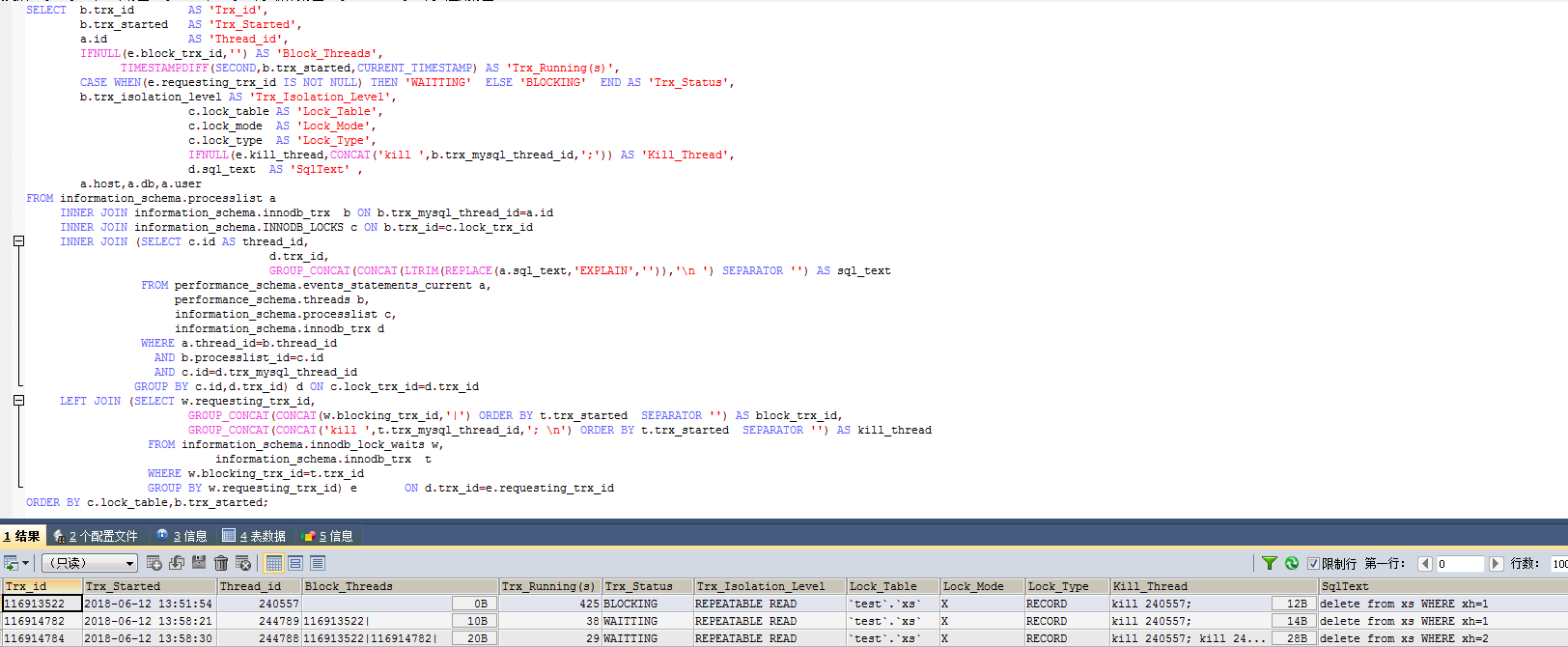
|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有3条记录，锁定表3条记录，锁等待3行记录  2.在可重复性读隔离级别下，当ID无索引时，删除操作会阻塞其会话对该表的更新操作。  3.在可重复性读隔离级别下，当ID无索引时，对整张表了X锁，其它会话无法更新该表。 |

锁等待查询：



#### （3）对删除影响测试

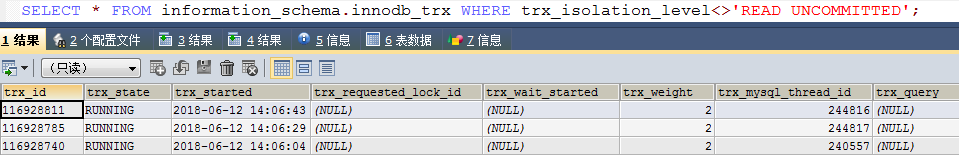
|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有三条记录，锁定表3条记录，锁等待3行记录  2.在可重复性读隔离级别下当ID无索引时，删除操作会阻塞其会话对该表的删除操作。  3.在可重复性读隔离级别下为了防止发生幻读，对整张表了X锁，其它会话无法删除该表记录。 |



### 4.1.4 有索引插入对DML影响

#### （1）对插入影响测试

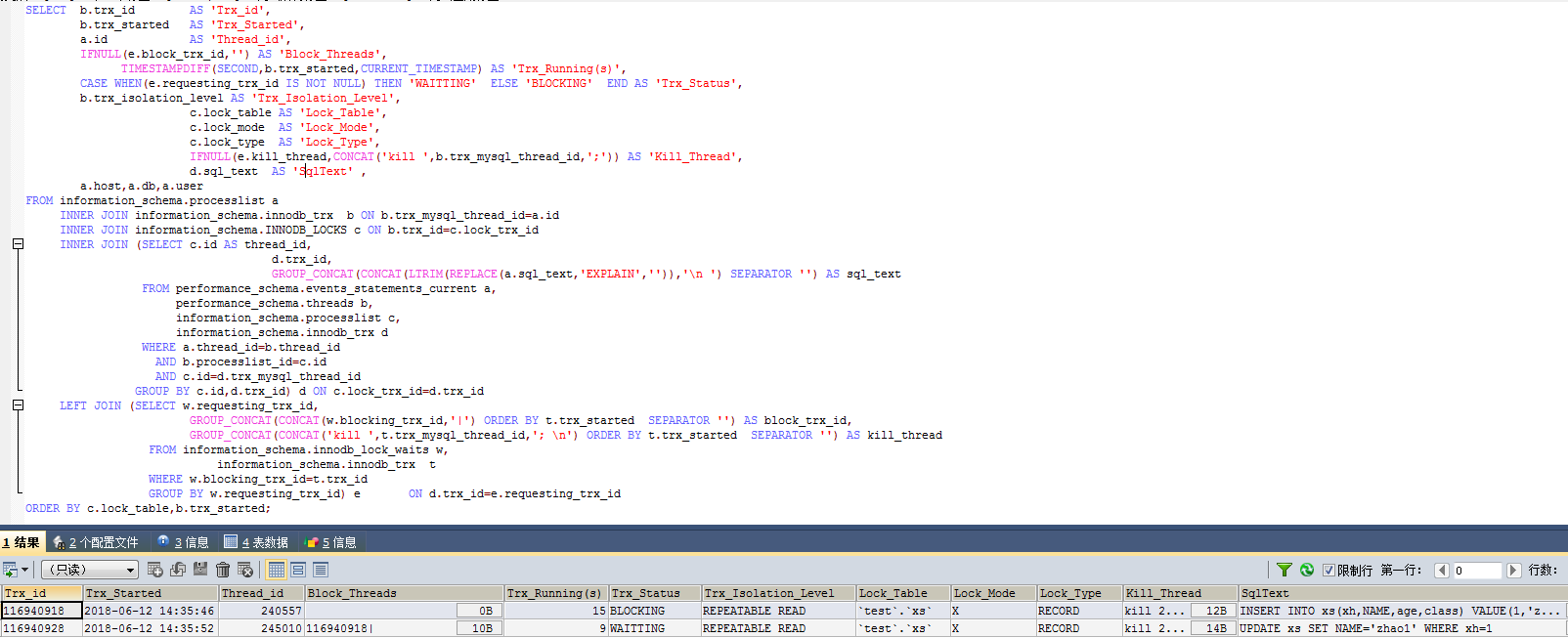
|  |
| --- |
| *#会话1：插入一条数据*  USE test;  TRUNCATE TABLE xs;  CREATE INDEX idx\_xs\_xh ON xs(xh);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：插入一条相同XH数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条不同XH数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(2,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3，在事务表中可同时查询出3条事务信息，没有锁定及阻塞信息。  2.在可重复性读隔离级别下,当XH有索引时，插入操作不会阻塞其它插入操作。 |



#### （2）对更新影响测试

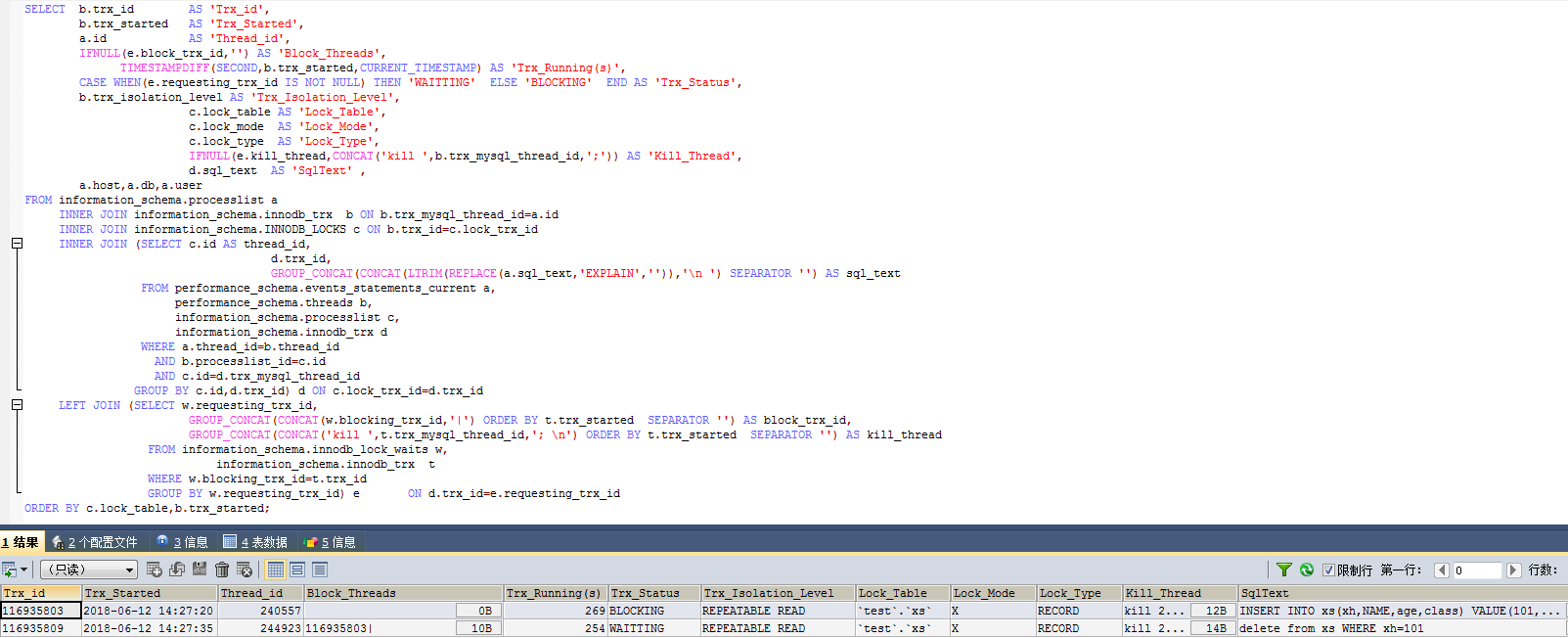
|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  call proc\_xs\_ins(100);  create index idx\_xs\_xh on xs(xh);  *#开始事务*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhao1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：更新与“会话1”XH相同的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：更新与“会话1”XH不同的数据，可以继续*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang99' WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2，不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在可重复性读隔离级别下，当ID有索引时，插入操作会阻塞相同条件的会话，不会阻塞不同条件的会话对该表的更新操作。  3.在可重复性读隔离级别下，当ID有索引时，则只在满足条件的记录上加X锁。 |

锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  CALL proc\_xs\_ins(100);  create index idx\_xs\_xh on xs(xh);  *#开始事务*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：删除一条不存在数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=101;  *#会话3：删除一条已存在数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2，不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在可重复性读隔离级别下，当ID有索引时，插入操作会阻塞相同条件的会话进行删除操作，不会阻塞不同条件的会话对该表的删除操作。  3.在可重复性读隔离级别下，如果XH上有索引，则只在满足条件的记录上加X锁。 |

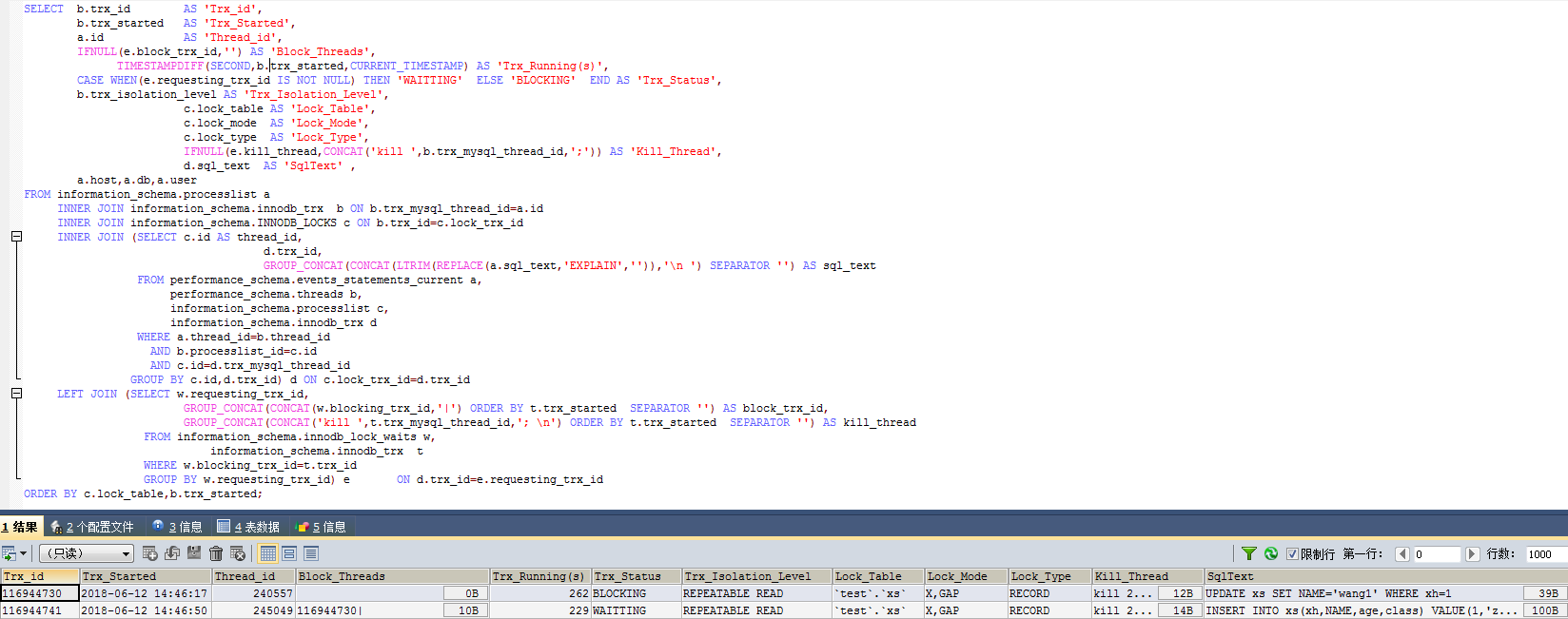


### 4.1.5 有索引更新对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在可重复性读隔离级别下，当XH有索引时，更新操作不会阻塞相同条件的会话。  3.在可重复性读隔离级别下，当XH有索引时，只对满足条件的记录加行级X锁，不满足条件的行可以进行插入操作。 |

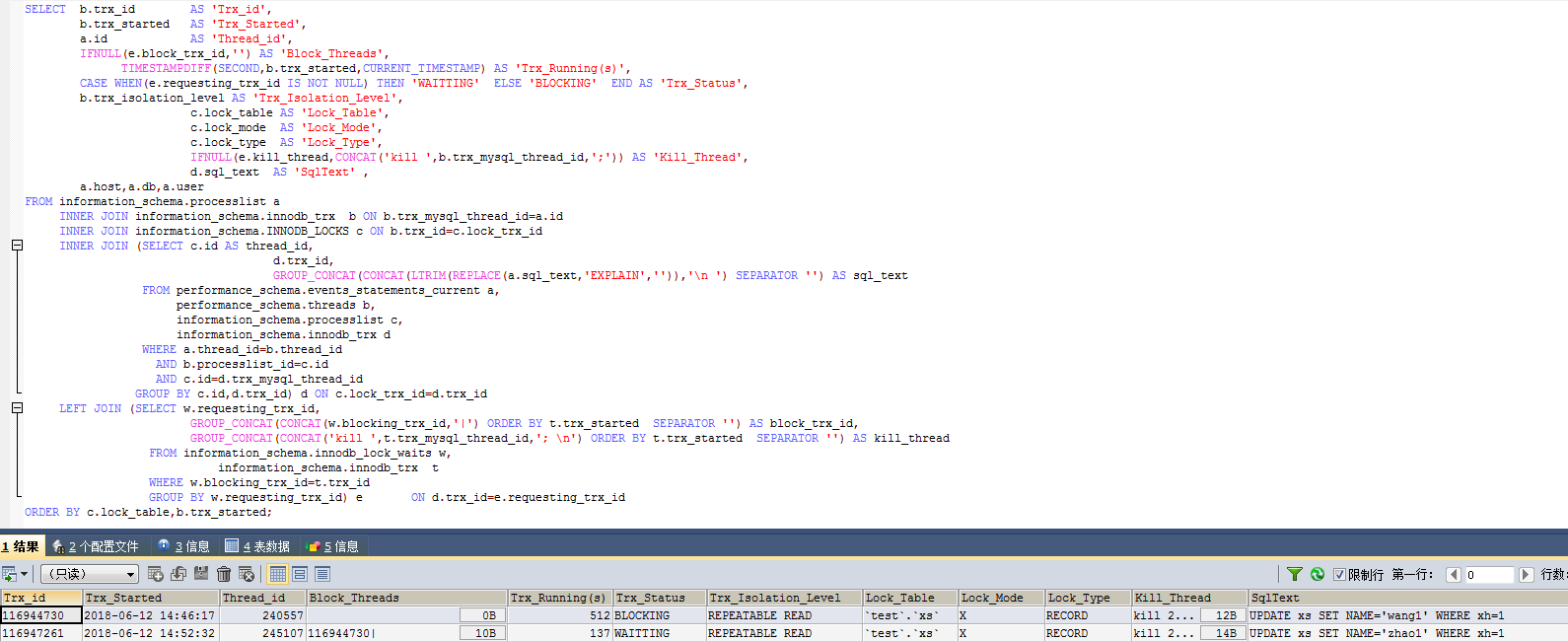
锁等待查询：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2，不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在可重复性读隔离级别下，当XH有索引时，更新操作会阻塞相同条件的更新操作，不同条件的更新操作不受影响。  3.在可重复性读隔离级别下，当XH有索引时，只对满足条件的记录加行级X锁，不满足条件的行可以进行更新操作。 |

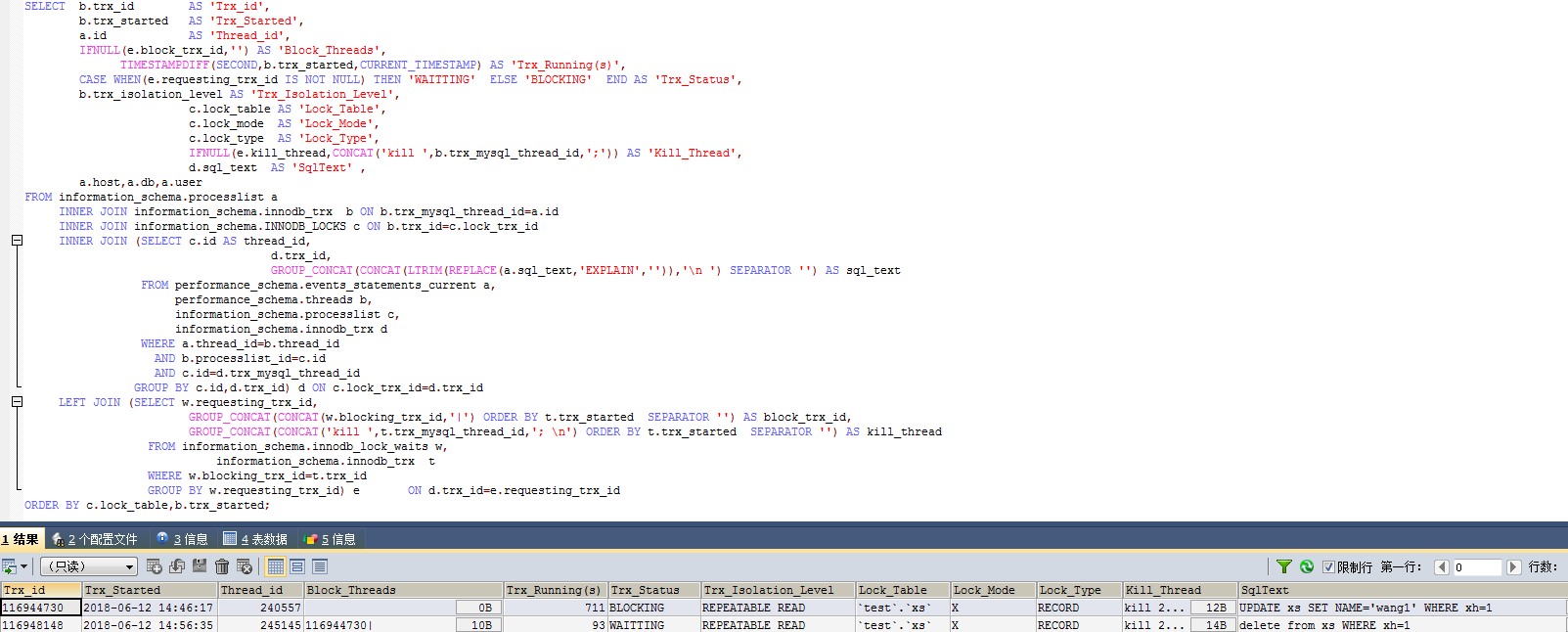
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2，不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在可重复性读隔离级别下,当XH有索引时，更新操作会阻塞其会话该表的相同条件的删除操作，不同条件的删除操作不受影响。  3.在可重复性读隔离级别下,如果条件列上有索引，只对满足条件的记录加行级X锁，不满足条件的行可以进行删除操作。 |

锁等待查询：

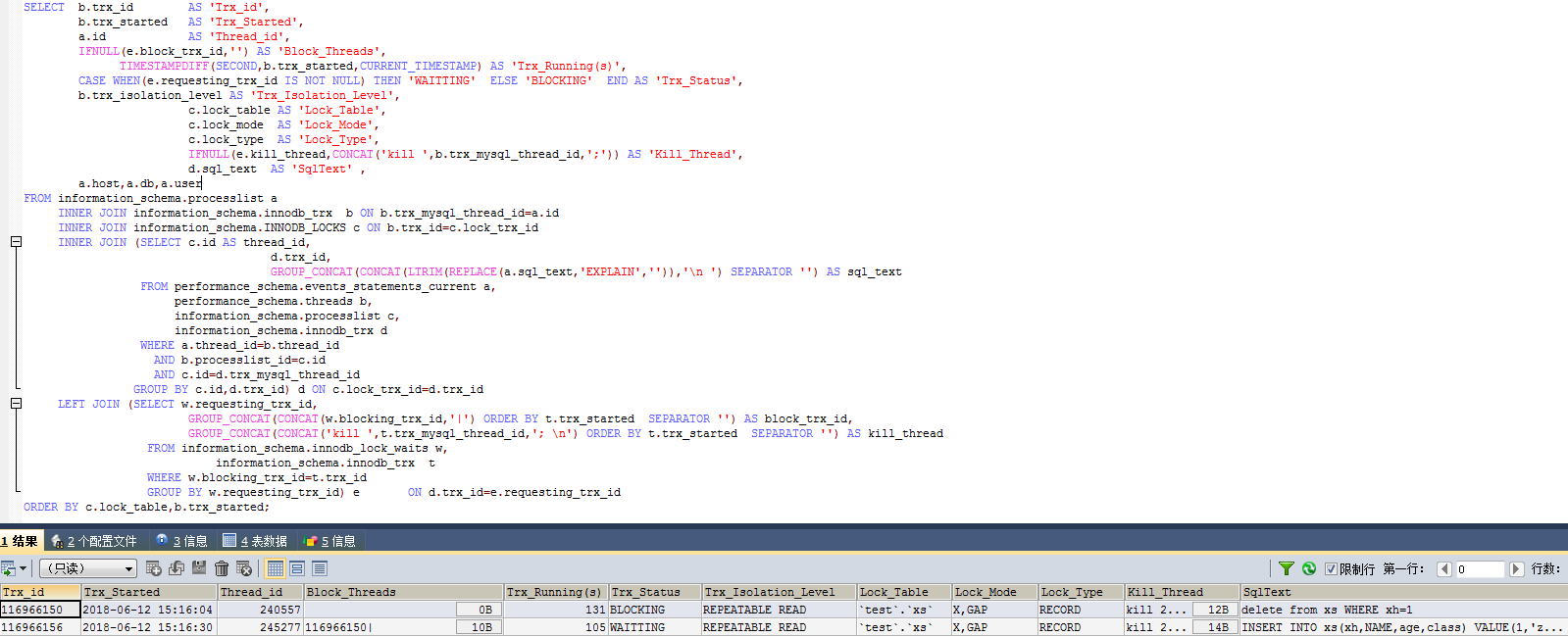


### 4.1.6 有索引删除对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在可重复性读隔离级别下,当XH有索引时，删除操作会阻塞相同条件的其会话对该表的插入操作。  3.在可重复性读隔离级别下,当XH有索引时，删除操作只对满足条件的记录加行级X锁，不满足条件的插动操作可以继续。 |

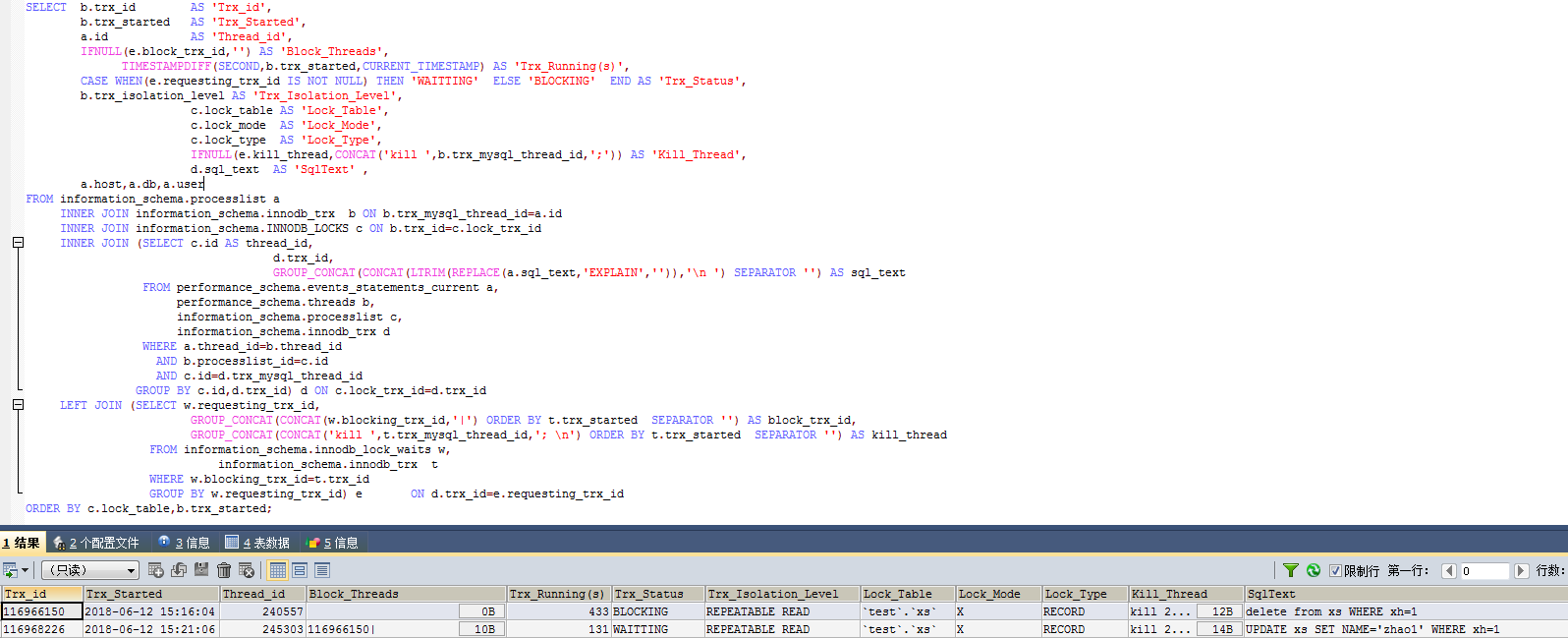
锁等待查询：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在可重复性读隔离级别下，当XH有索引时，删除操作会阻塞相同条件的其会话对该表的更新操作。  3.在可重复性读隔离级别下，当XH列有索引时，只对满足条件的记录加行级X锁，不满足条件的插动操作可以继续。 |

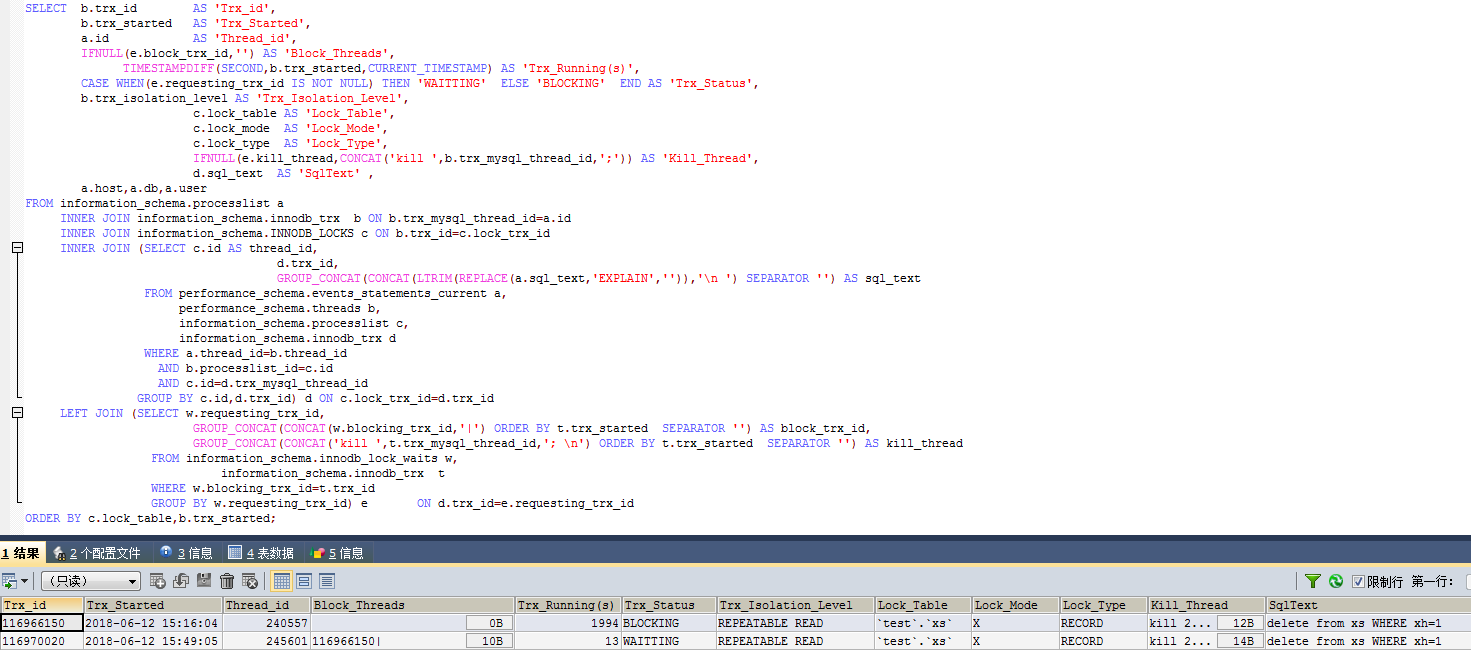
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在可重复性读隔离级别下，当XH有索引时，删除操作会阻塞相同条件的其会话对该表的删除操作。  3.在可重复性读隔离级别下，当XH有索引时，只对满足条件的记录加行级X锁，不满足条件的删除操作可以继续。 |

锁等待查询：



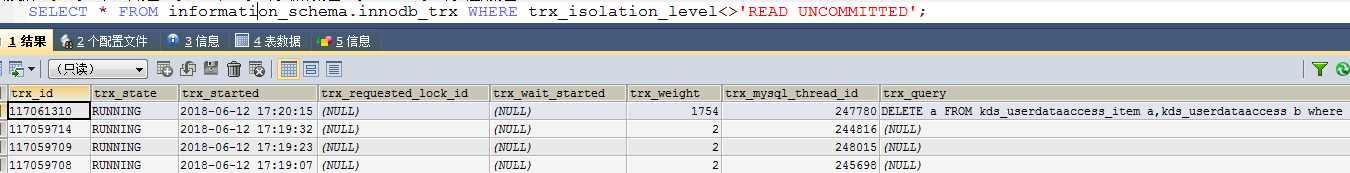
## 4.2 已提交读

### 4.2.1 无索引插入对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  drop index idx\_xs\_xh on xs;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  truncate table xs;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：插入一条相同XH数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条不同XH数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(2,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3，在事务表中可同时查询出3条事务信息，没有锁定及阻塞信息。  2.在已提交读隔离级别下，当XH无索引时，插入操作不会阻塞其它插入操作。  3.在已提交读隔离级别下，当XH无索引时，插入操作不会对表和记录加X锁。 |

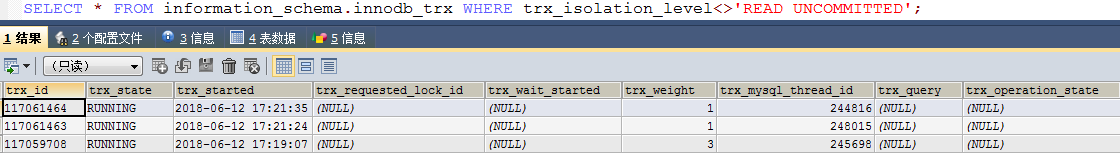
事务信息：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  *#开始事务*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhao1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：更新与“会话1”XH相同的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：更新与“会话1”XH不同的数据，可以继续*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang99' WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3，在事务表中可同时查询出3条事务信息，没有锁定及阻塞信息。  2.在可重复性读隔离级别下，当XH无索引时，插入操作不会阻塞其它更新操作。 |

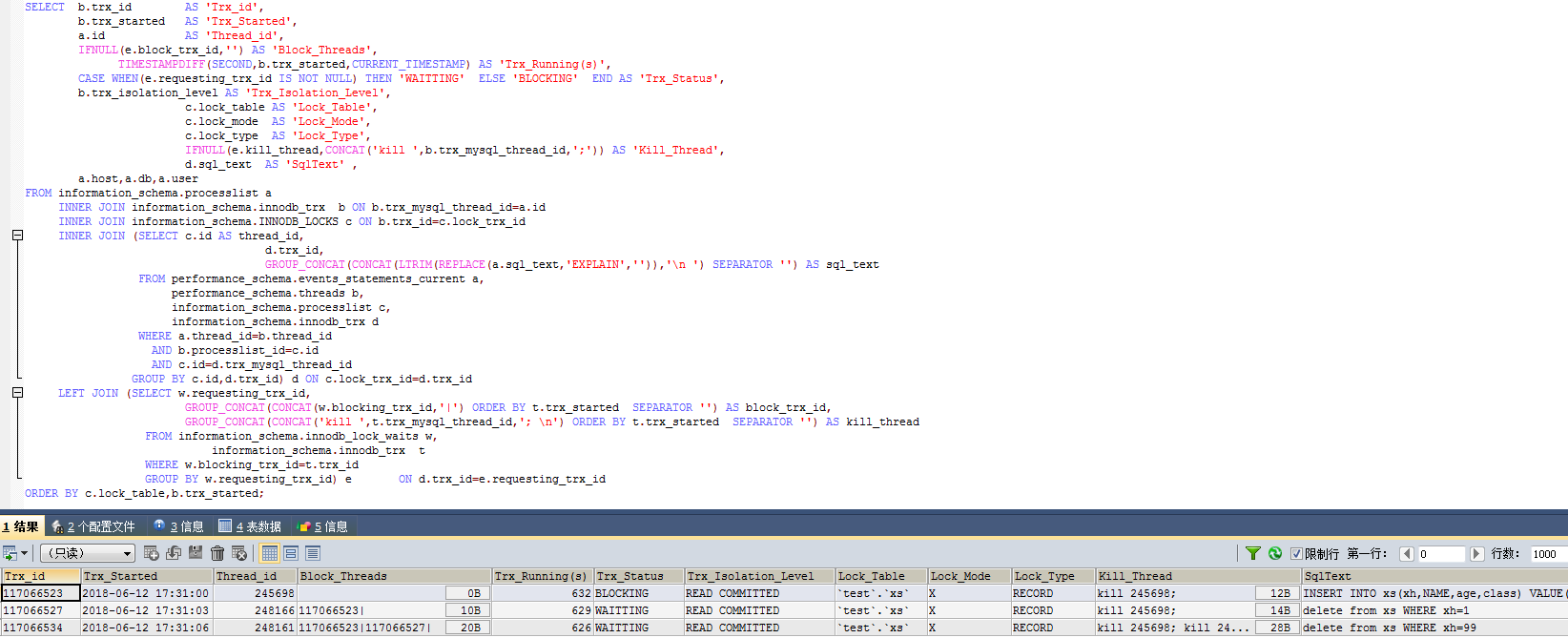
事务信息，没有锁定和等待：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  drop index idx\_xs\_xh on xs;  CALL proc\_xs\_ins(100);  *#开始事务*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：删除一条相同XH的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：删除一条不同XH的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2和会话3,在事务表中有3条记录，锁定表4条记录，锁等待2行记录  2.在已提交读隔离级别下，当ID无索引时，插入操作会在表级别增加独占锁，其它相同条件或不同条件的会话的删除操作将被阻塞。 |

锁阻塞情况：

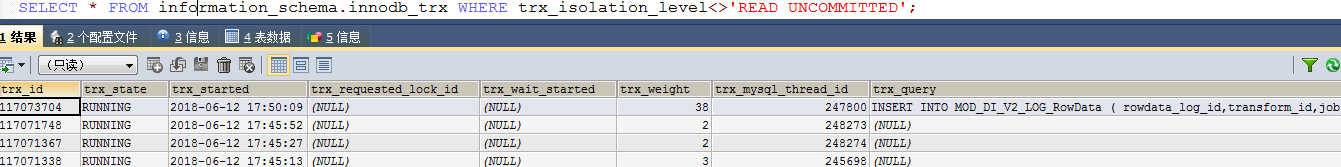


### 4.2.2 无索引更新对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3，在事务表中可同时查询出3条事务信息，没有锁定及阻塞信息。  2.在已提交读隔离级别下，当XH无索引时，更新操作只在满足条件的记录上加X锁, 不会阻塞其它插入操作。 |

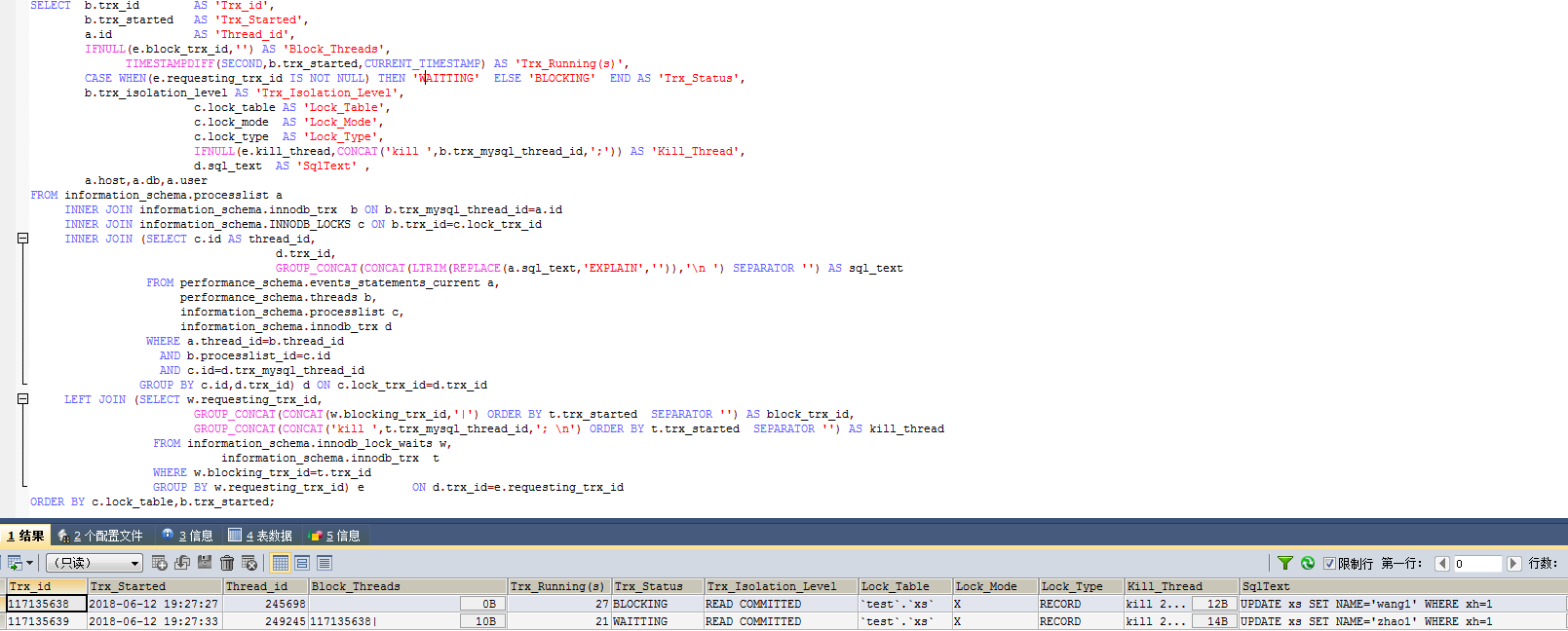
无锁等待,以下是事务查询：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2，不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在已提交读隔离级别下，当XH无索引时，更新操作会阻塞其会话对该表的相同条件的更新操作，不同条件的更新操作不受影响。  3.在已提交读隔离级别下，当XH无索引时，只对满足条件的记录加行级X锁，不满足条件的行可以进行更新操作。 |

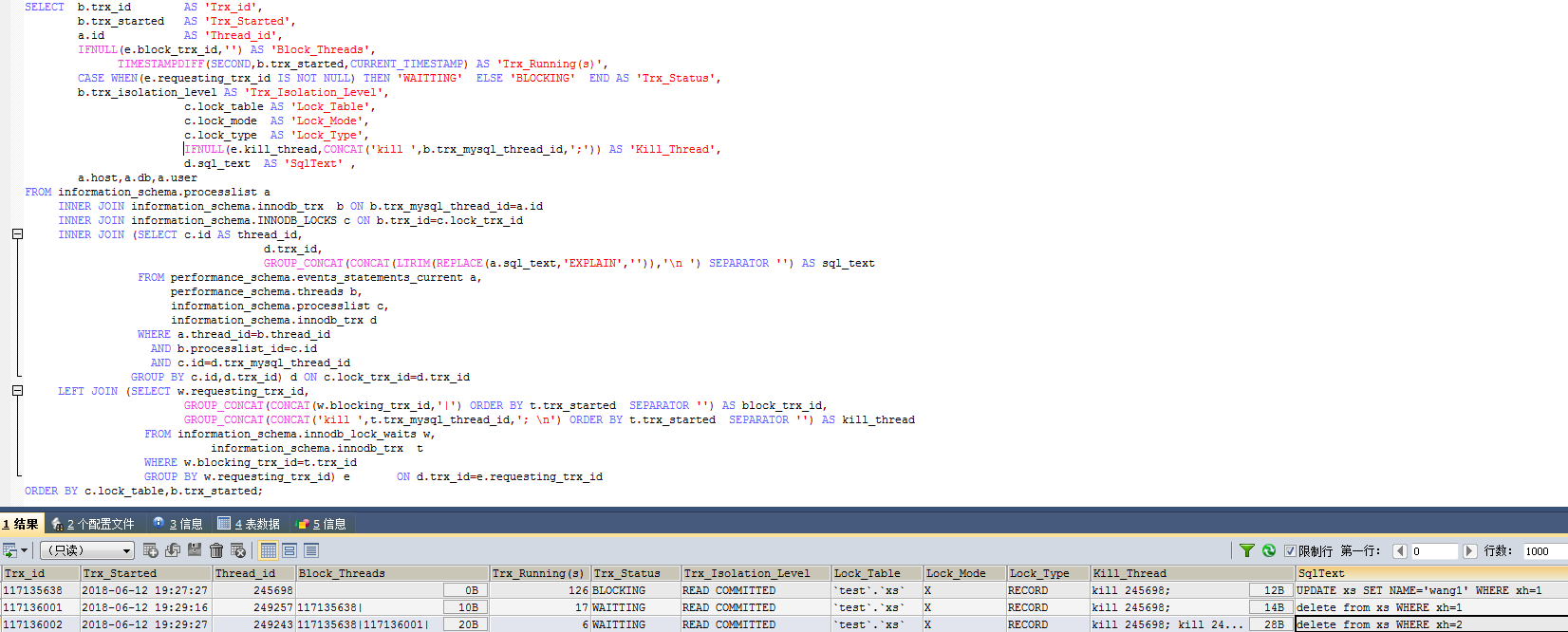
锁等待查询：

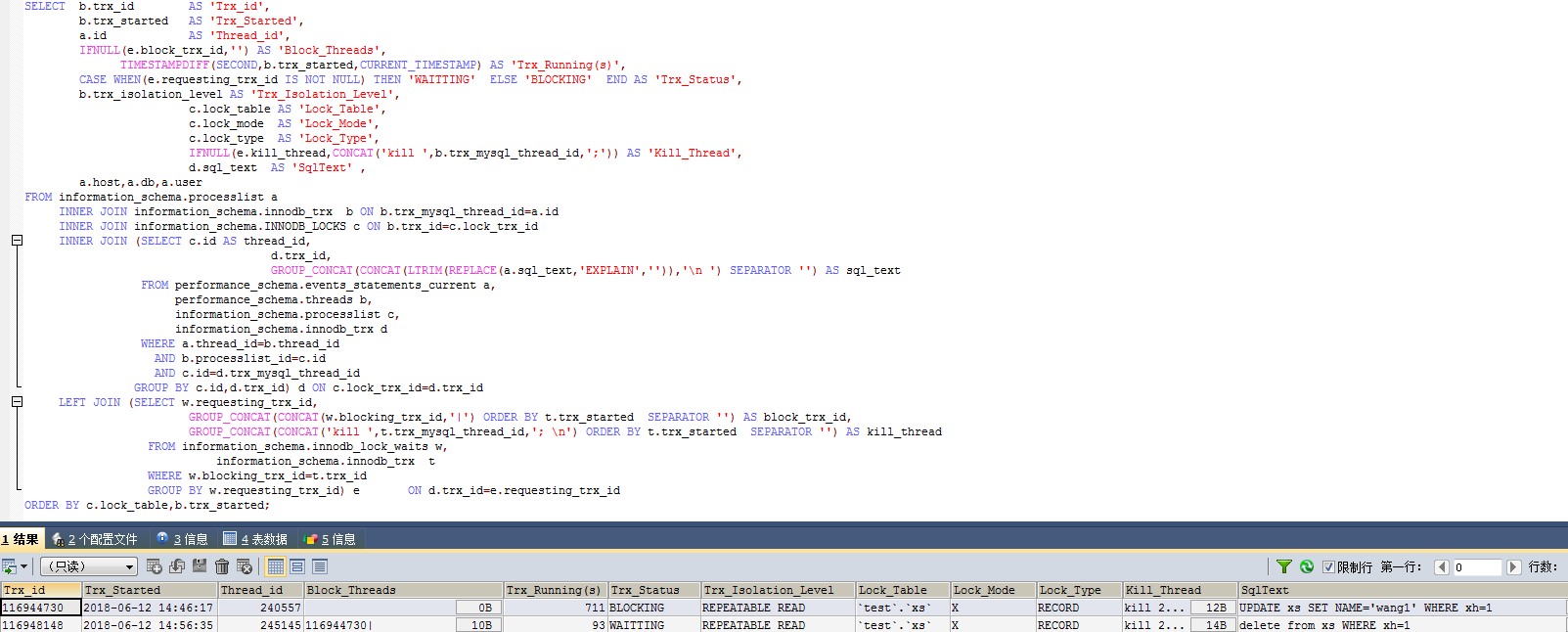


#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、阻塞会话3,在事务表中有3条记录，锁定表3条记录，锁等待3行记录。  2.已提交读隔离级别下，当XH无索引时，更新操作会阻塞其会话对该表的的删除操作。  3.已提交读隔离级别下，当XH无索引时，对表加X锁，所有对表的删除操作均被阻塞。 |

锁等待查询：



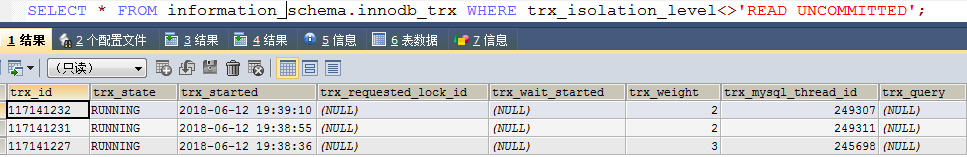


### 4.2.3 无索引删除对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3,在事务表中有3条记录，无锁定记录，无锁等待记录。  2.已提交读隔离级别下,当XH无索引时，删除操作不会阻塞其会话对该表的插入操作。 |

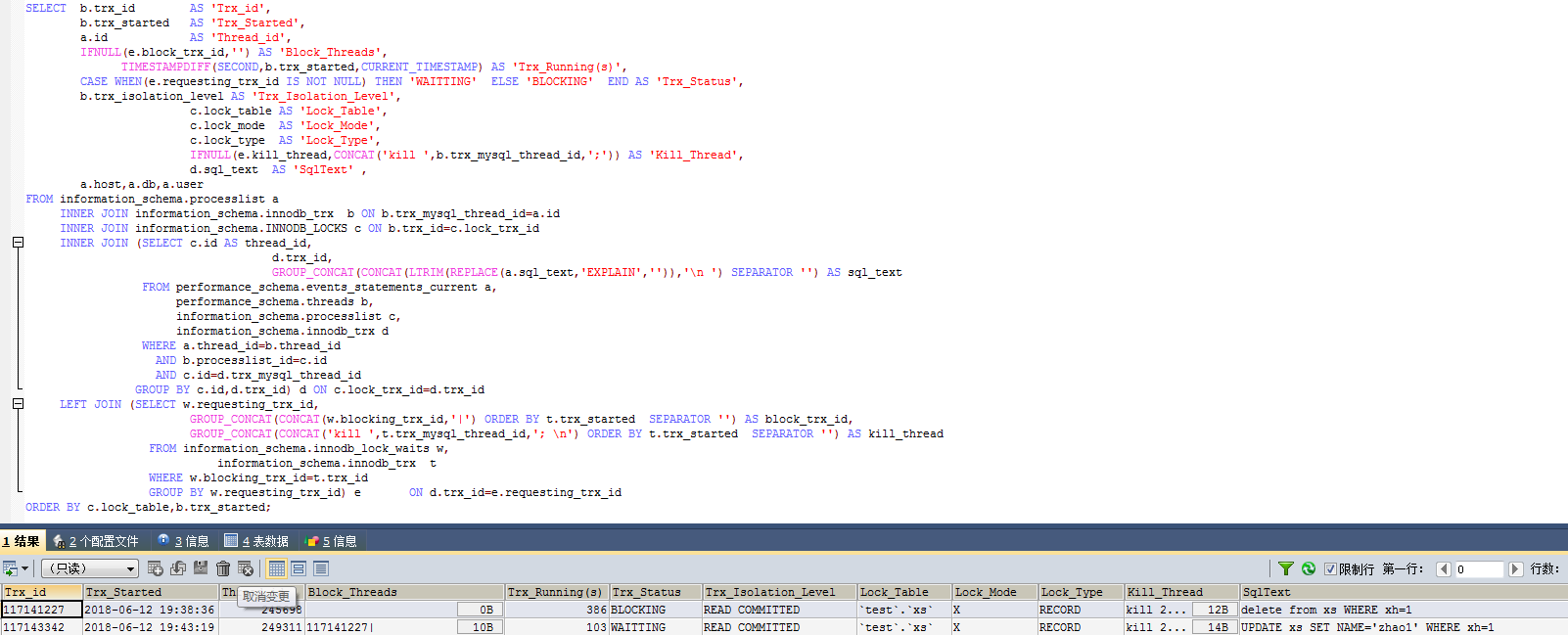
无锁等待,只有三个事务：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不会阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在已提交读隔离级别下,当XH无索引时，删除操作会阻塞相同条件的其会话对该表的更新操作。  3.在已提交读隔离级别下,当XH无索引时，只对满足条件的记录加行级X锁，不满足条件的更新操作可以继续。 |

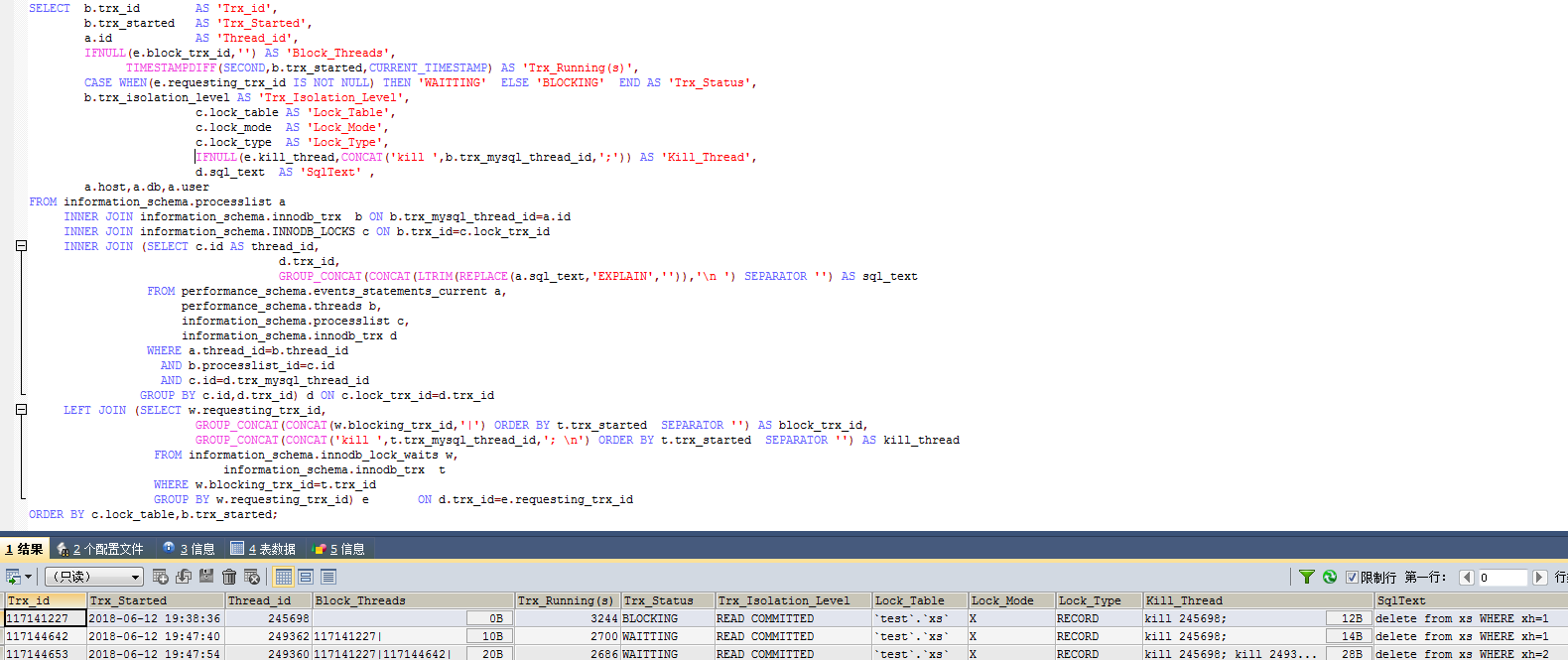
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test  truncate table xs;  drop index idx\_xs\_xh on xs;  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、会话3,在事务表中有3条记录，锁定表3条记录，锁等待3行记录。  2.在已提交读隔离级别下，当XH无索引时，删除操作会阻塞其会话对该表的删除操作。  3.在已提交读隔离级别下，当XH无索引时，只对表加X锁，所有对该表的删除操作均被阻塞。 |

锁等待查询：



### 4.2.4 有索引插入对DML影响

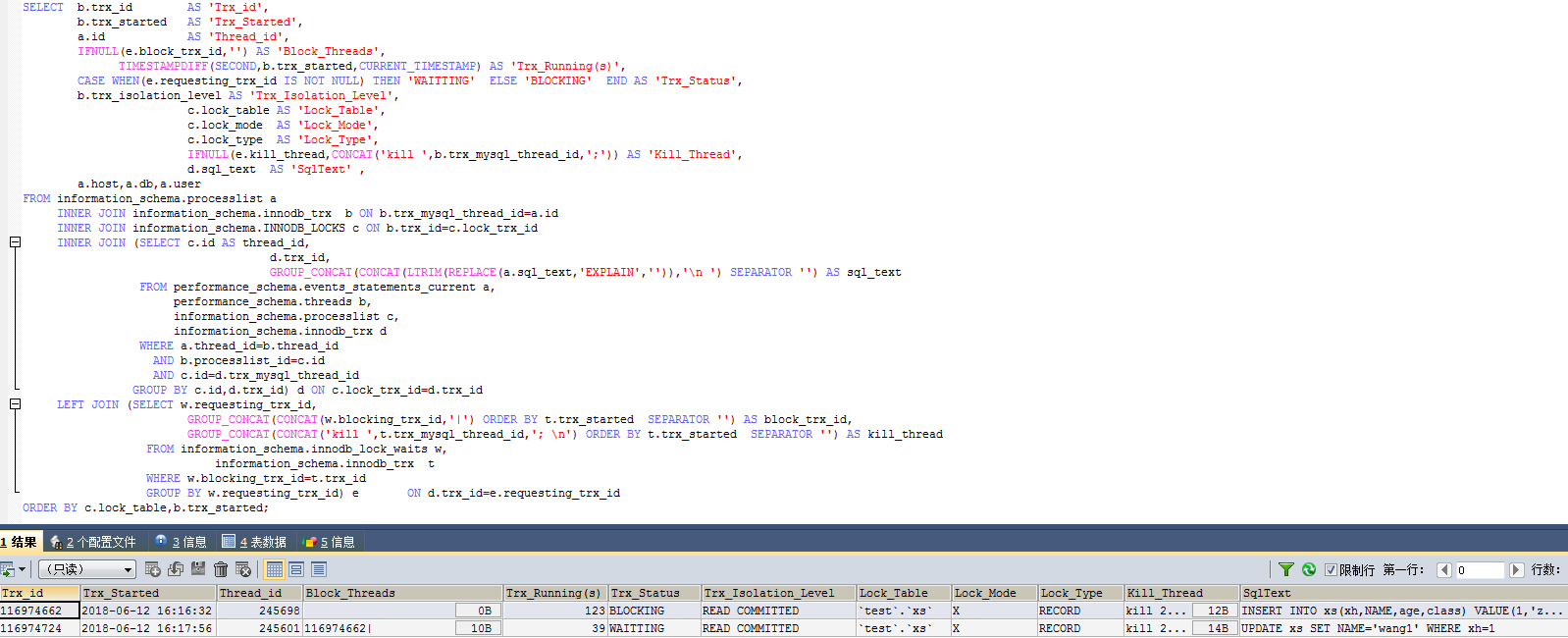
#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：插入一条数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(2,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3，在事务表中可同时查询出3条事务信息，没有锁定及阻塞信息。  2.在已提交读隔离级别下，当XH有索引时，插入操作不会阻塞其它插入操作。 |

#### （2）对更新影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  *#开始事务*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：更新与“会话1”XH相同的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话3：更新与“会话1”XH不同的数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang99' WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2，不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在已提交读隔离级别下，当XH有索引时，插入操作会阻塞其会话对该表的更新操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中满足条件的行记录加X锁。 |

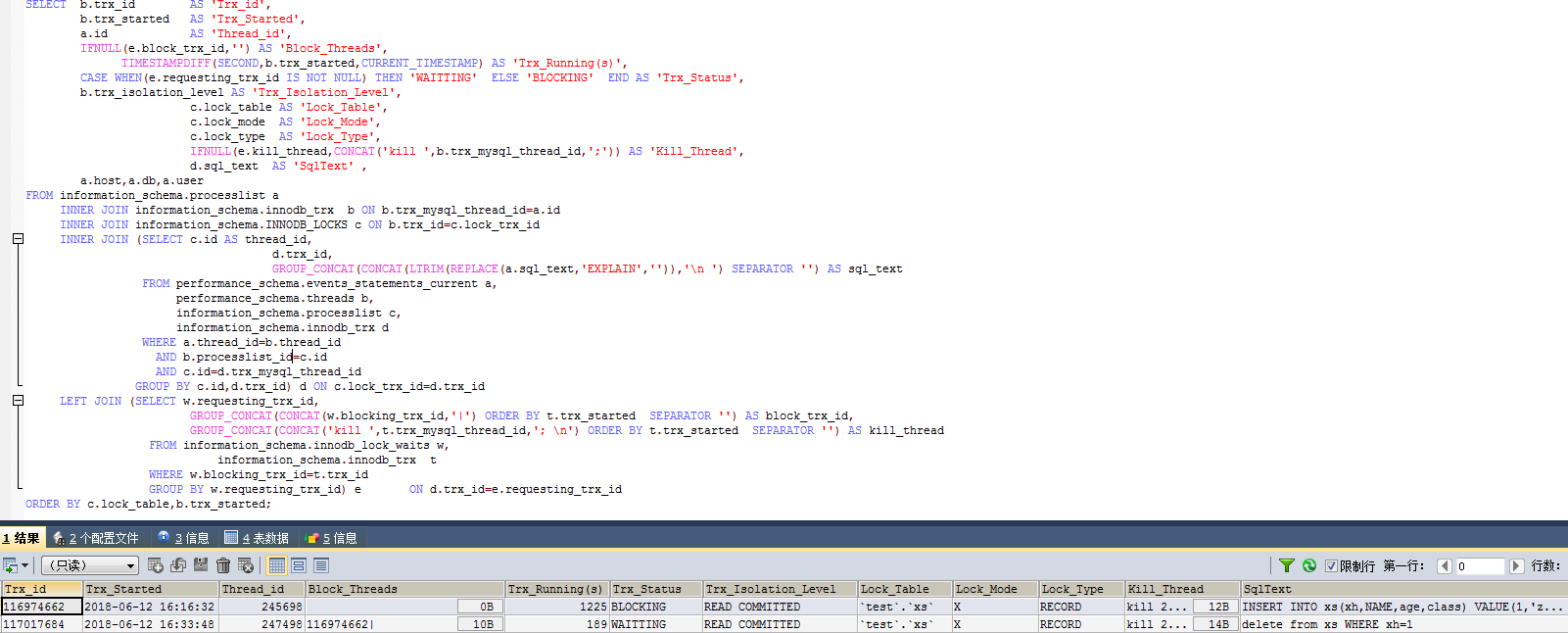
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：准备数据，XH=1~100*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  CALL proc\_xs\_ins(100);  *#开始事务*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话2：删除一条不存在数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：删除一条已存在数据，处理等待中*  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=99;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在已提交读隔离级别下，当XH有索引时，插入操作会阻塞相同条件的会话对该表的删除操作，不同条件的删除操作可以继续。  3.在已提交读隔离级别下，当XH有索引时，只对表中满足条件的记录加X锁。 |

锁等待查询：

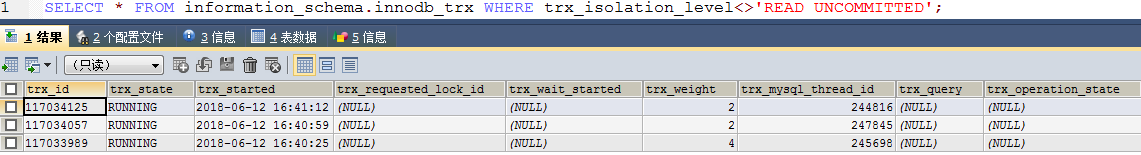


### 4.2.5 有索引更新对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3,在事务表中有三条记录，没有锁定及阻塞信息。  2.在已提交读隔离级别下，当XH有索引时，更新操作只会锁定数据库中表中满足条件的行的记录，不会阻塞相同条件或不同条件的其会话对该表的插入操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中已有数据且满足条件的记录加X锁，对于插入操作不论插入是否相同XH的记录都是可以的，因为没有对新数据加锁。 |

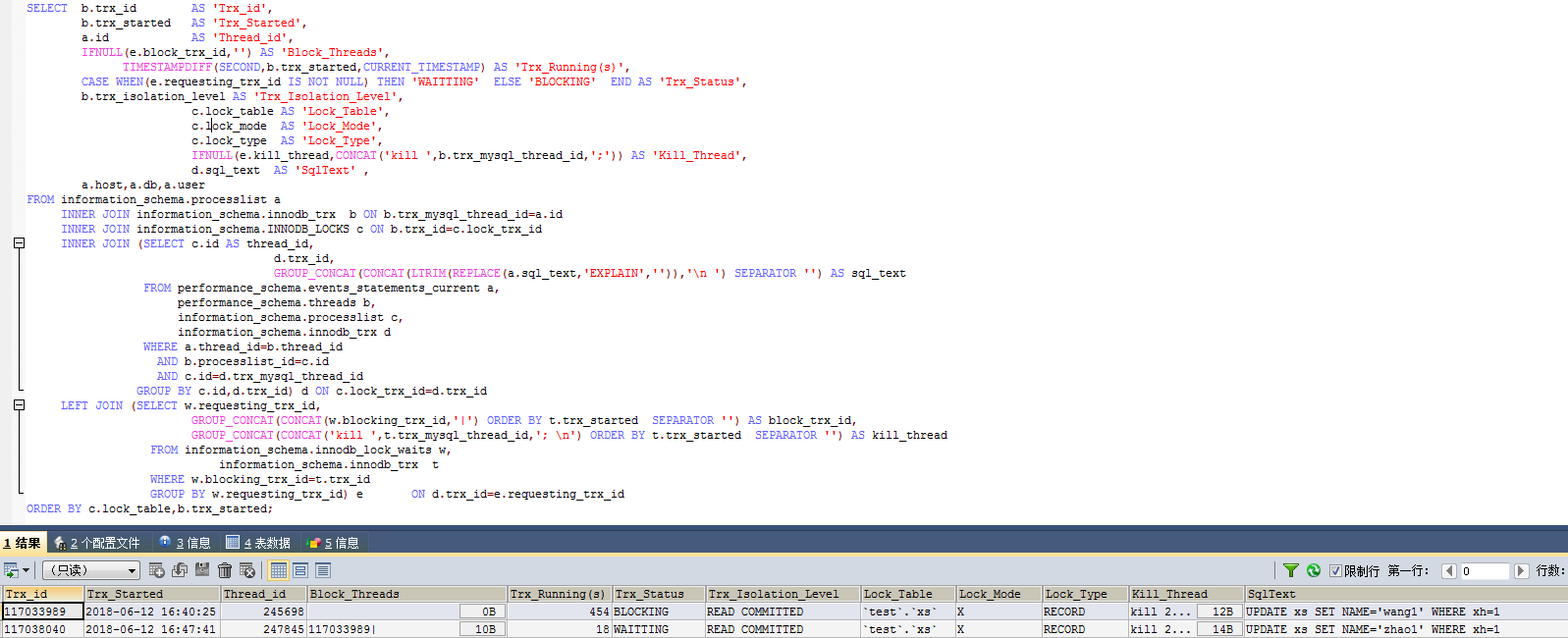
锁等待查询：无，只有三条事务信息，如下图：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞“会话2”，不会阻塞“会话3”,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在已提交读隔离级别下，当XH有索引时，更新操作会阻塞相同条件的其会话对该表的更新操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中已有数据且满足条件的记录加X锁，对于更新操作会阻寒相同条件的会话，不同条件的更新操作可以继续。 |

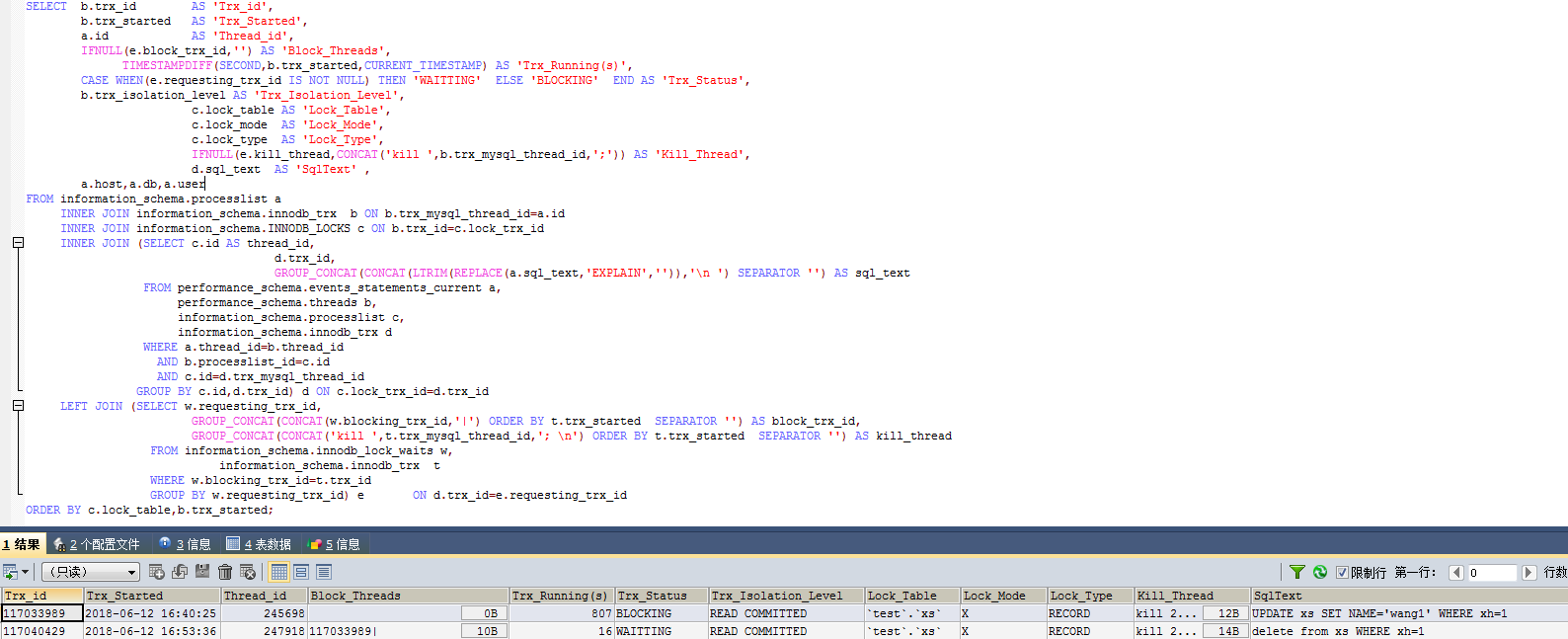
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='wang1' WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2、不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在已提交读隔离级别下，当XH有索引时，更新操作会阻塞相同条件会话对该表的删除操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中已有数据的满足条件的行加X锁，其它会话可以删除不满足条件的记录。 |

锁等待查询：

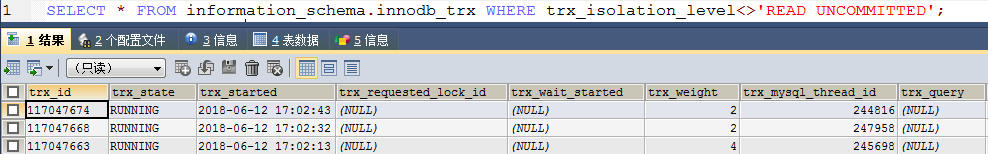


### 4.2.6 有索引删除对DML影响

#### （1）对插入影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(1,'zhang1', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  INSERT INTO xs(xh,NAME,age,class)  VALUE(101,'zhang101', FLOOR(RAND()\*100), CHAR(65+FLOOR(RAND()\*25)));  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1不阻塞会话2、会话3,在事务表中有三条记录，没有锁定及阻塞信息。  2.在已提交读隔离级别下，当XH有索引时，删除操作只会锁定数据库中表中满足条件的行的记录，不会阻塞相同条件或不同条件的其会话对该表的插入操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中已有数据且满足条件的记录加X锁，对于插入操作不论插入是否相同XH的记录都是可以的，因为没有对新数据加锁。 |

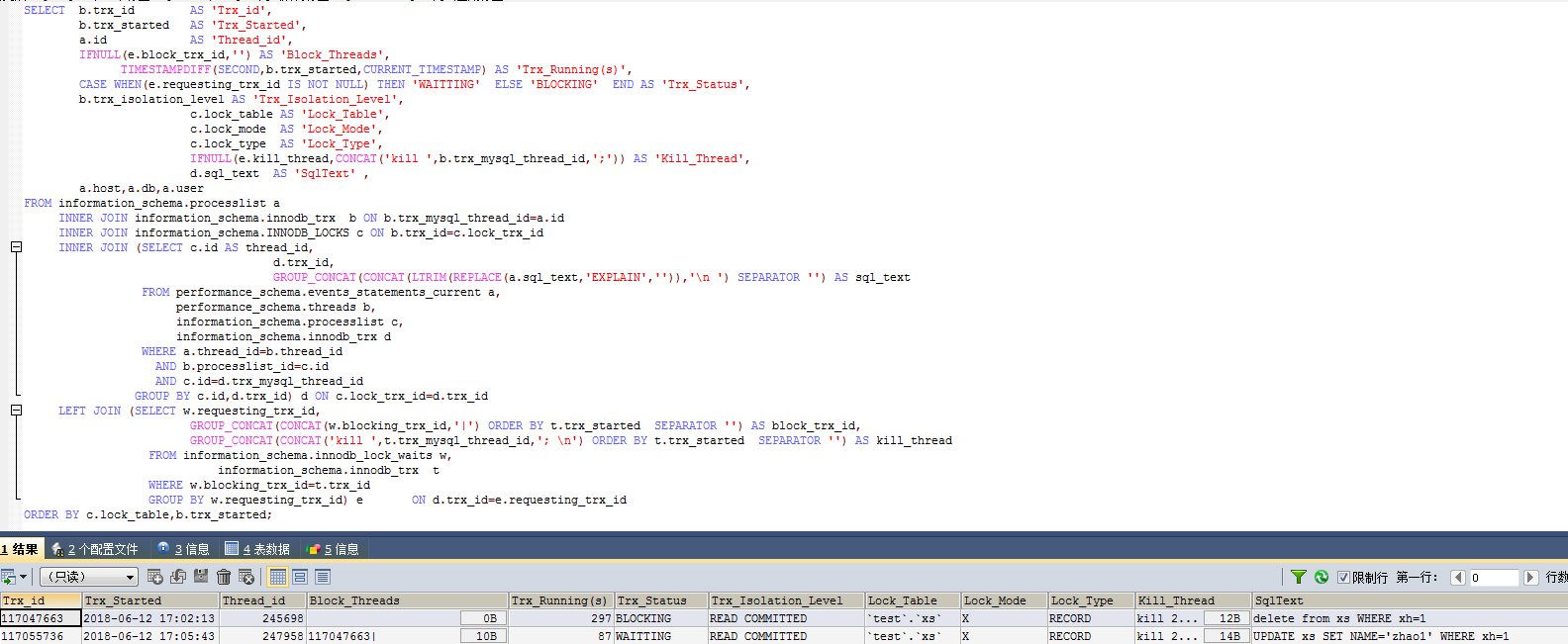
锁等待查询：无，只有三条事务信息，如下图：



#### （2）对更新影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='zhao1' WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  UPDATE xs SET NAME='li1' WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞“会话2”，不会阻塞“会话3”,在事务表中有3条记录，锁定表2条记录，锁等待1行记录。  2.在已提交读隔离级别下，当XH有索引时，删除操作会阻塞相同条件的其会话对该表的更新操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中已有数据且满足条件的记录加X锁，对于更新操作会阻塞相同条件的会话，不同条件的更新操作可以继续。 |

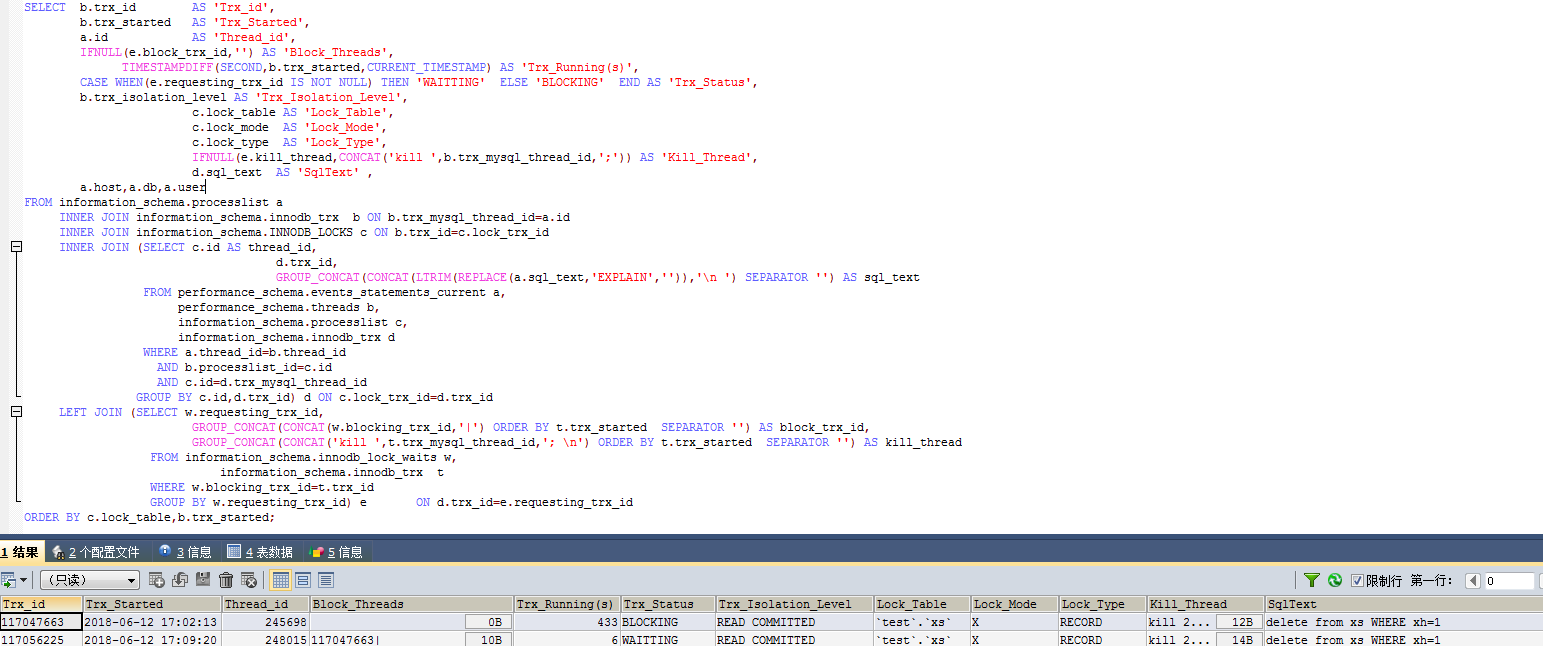
锁等待查询：



#### （3）对删除影响测试

|  |
| --- |
| *#会话1：插入一条数据*  use test;  truncate table xs;  create index idx\_xs\_xh on xs(xh);  call proc\_xs\_ins(100);  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话2：插入一条数据,XH与更新的序号一样*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=1;  *#会话3：插入一条数据，一个新的XH*  use test;  SET SESSION TRANSACTION ISOLATION LEVEL READ COMMITTED;  START TRANSACTION;  delete from xs WHERE xh=2;  *#事务及锁定查询*  SELECT \* FROM information\_schema.innodb\_trx;  SELECT \* FROM information\_schema.innodb\_locks;  SELECT \* FROM information\_schema.innodb\_lock\_waits;  *#结论*  1.会话1阻塞会话2,不阻塞会话3,在事务表中有3条记录，锁定表2条记录，锁等待1行记录  2.在已提交读隔离级别下，当XH有索引时，删除操作会阻塞相同条件会话对该表的删除操作。  3.在已提交读隔离级别下，当XH有索引时，只对表中已有数据的满足条件的行加X锁，其它会话可以删除不满足条件的记录。 |

锁等待查询：



# 五、总结

通过以上对可重复性读和已提交读分36种情况进行的测试表明，已提交读隔离级别比可重复性读隔离级别能够支持更大程序的并发，减少不同会话的锁等待和阻塞。

# 六、建议

先在测试环境中修改MySQL数据库默认事务隔离级别为已提交读，进行一段时间使用后，再在RDS预发布环境中修改，最后在生产环境中修改。

# 七、附件

## 7.1 锁等待查询

|  |
| --- |
| SELECT b.trx\_id AS 'Trx\_id',  b.trx\_started AS 'Trx\_Started',  a.id AS 'Thread\_id',  IFNULL(e.block\_trx\_id,'') AS 'Block\_Threads',  TIMESTAMPDIFF(SECOND,b.trx\_started,CURRENT\_TIMESTAMP) AS 'Trx\_Running(s)',  CASE WHEN(e.requesting\_trx\_id IS NOT NULL) THEN 'WAITTING' ELSE 'BLOCKING' END AS 'Trx\_Status',  b.trx\_isolation\_level AS 'Trx\_Isolation\_Level',  c.lock\_table AS 'Lock\_Table',  c.lock\_mode AS 'Lock\_Mode',  c.lock\_type AS 'Lock\_Type',  IFNULL(e.kill\_thread,CONCAT('kill ',b.trx\_mysql\_thread\_id,';')) AS 'Kill\_Thread',  d.sql\_text AS 'SqlText' ,  a.host,a.db,a.user  FROM information\_schema.processlist a  INNER JOIN information\_schema.innodb\_trx b ON b.trx\_mysql\_thread\_id=a.id  INNER JOIN information\_schema.INNODB\_LOCKS c ON b.trx\_id=c.lock\_trx\_id  INNER JOIN (SELECT c.id AS thread\_id, d.trx\_id,  GROUP\_CONCAT(CONCAT(LTRIM(REPLACE(a.sql\_text,'EXPLAIN','')),'\n ') SEPARATOR '') AS sql\_text  FROM performance\_schema.events\_statements\_current a,  performance\_schema.threads b,  information\_schema.processlist c,  information\_schema.innodb\_trx d  WHERE a.thread\_id=b.thread\_id  AND b.processlist\_id=c.id  AND c.id=d.trx\_mysql\_thread\_id  GROUP BY c.id,d.trx\_id) d ON c.lock\_trx\_id=d.trx\_id  LEFT JOIN (SELECT w.requesting\_trx\_id, GROUP\_CONCAT(CONCAT(w.blocking\_trx\_id,'|')  ORDER BY t.trx\_started SEPARATOR '') AS block\_trx\_id,  GROUP\_CONCAT(CONCAT('kill ',t.trx\_mysql\_thread\_id,'; \n')  ORDER BY t.trx\_started SEPARATOR '') AS kill\_thread  FROM information\_schema.innodb\_lock\_waits w,  information\_schema.innodb\_trx t  WHERE w.blocking\_trx\_id=t.trx\_id  GROUP BY w.requesting\_trx\_id) e ON d.trx\_id=e.requesting\_trx\_id  ORDER BY c.lock\_table,b.trx\_started; |

## 7.2 锁资源等待图

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 可重复性读 | | | | | | 已提交读 | | | | | |
| 无索引 | | | 有索引 | | | 无索引 | | | 有索引 | | |
| I | U | D | I | U | D | I | U | D | I | U | D |
| INSERT | √ | × | × | √ | △ | △ | √ | √ | √ | √ | √ | √ |
| UPDATE | × | × | × | △ | △ | △ | √ | △ | △ | △ | △ | △ |
| DELETE | × | × | × | △ | △ | △ | △ | △ | × | △ | △ | △ |

说明：√：表示对相同或不同条件的记录均不阻塞

×：表示对相同或不同条件的记录均阻塞

△：表示对相同条件记录阻塞，不同条件的记录不阻塞