

1. 数据库

PRODUCTTABLE :

表	ID	PID	PNAME	Count	AID
accounttable					
buyproductrecordtable	3	1	Apple	1000	0
producttable					
registerrecordtable					

buyproductrecordtable

RecordId	AccountId	ProductId	Count
1001	247337	1	1
1002	246630	1	1
1003	245331	1	1
1004	247728	1	1
1005	246113	1	1
1006	247527	1	1
1007	247504	1	1
1008	245710	1	1
1009	247010	1	1
1010	248032	1	1
1011	247124	1	1
1012	247515	1	1
1013	245193	1	1
1014	247515	1	1
1015	245900	1	1
1016	247527	1	1
1017	246596	1	1
1018	244992	1	1
1019	246596	1	1
1020	246998	1	1
1021	247895	1	1
1022	247303	1	1
1023	247716	1	1
1024	245319	1	1
1025	248602	1	1
1026	248423	1	1
1027	245205	1	1
1028	246216	1	1

报错信息：



The screenshot shows a Java IDE with a debugger window open. The window title is "file:///E:/Project/RequestRobotProject/RequestRobot/RequestRobot/bin/Debug/RequestRobot.exe". The content of the window is a stack trace of a deadlock error. The error message is: "Deadlock found when trying to get lock; try restarting transaction". The stack trace shows the error occurred in the "com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException" class. The error message is repeated three times, indicating a recursive deadlock. The stack trace also shows the error occurred while setting parameters for the SQL statement: "UPDATE ProductTable SET Count=Count-1 WHERE PID=?". The stack trace ends with "成功!" (Success!).

```
Deadlock found when trying to get lock; try restarting transaction
; 1; Deadlock found when trying to get lock; try restarting transaction; nested
exception is com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException:
Deadlock found when trying to get lock; try restarting transaction

### Error updating database. Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransa
ctionRollbackException: Deadlock found when trying to get lock; try restarting t
ransaction
### The error may involve com.example.demo.BookInterface.IProductBook.UpdateCoun
t-Inline
### The error occurred while setting parameters
### SQL: UPDATE ProductTable SET Count=Count-1 WHERE PID=?
### Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException: De
adlock found when trying to get lock; try restarting transaction
; 1; Deadlock found when trying to get lock; try restarting transaction; nested
exception is com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException:
Deadlock found when trying to get lock; try restarting transaction

### Error updating database. Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransa
ctionRollbackException: Deadlock found when trying to get lock; try restarting t
ransaction
### The error may involve com.example.demo.BookInterface.IProductBook.UpdateCoun
t-Inline
### The error occurred while setting parameters
### SQL: UPDATE ProductTable SET Count=Count-1 WHERE PID=?
### Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException: De
adlock found when trying to get lock; try restarting transaction
; 1; Deadlock found when trying to get lock; try restarting transaction; nested
exception is com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException:
Deadlock found when trying to get lock; try restarting transaction

成功!
成功!

### Error updating database. Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransa
ctionRollbackException: Deadlock found when trying to get lock; try restarting t
ransaction
### The error may involve com.example.demo.BookInterface.IProductBook.UpdateCoun
t-Inline
### The error occurred while setting parameters
### SQL: UPDATE ProductTable SET Count=Count-1 WHERE PID=?
### Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException: De
adlock found when trying to get lock; try restarting transaction
; 1; Deadlock found when trying to get lock; try restarting transaction; nested
exception is com.mysql.jdbc.exceptions.jdbc4.MySQLTransactionRollbackException:
Deadlock found when trying to get lock; try restarting transaction

成功!

### Error updating database. Cause: com.mysql.jdbc.exceptions.jdbc4.MySQLTransa
ctionRollbackException: Deadlock found when trying to get lock; try restarting t
ransaction
### The error may involve com.example.demo.BookInterface.IProductBook.UpdateCoun
```

购买商品整个事务流程（商品数量 > 0 的情况下）（事务隔离级别为序列化的级别下会出现上述错误。）

①：SELECT * FROM PRODUCT WHERE ID=3 查出商品信息。（对该记录 添加 共享锁）。

②：UPDATE PRODUCT SET COUNT =COUNT -1 WHERE PID=3 (对该记录添加排它锁)。

③：INSERT INTO buyproductrecordtable new Record (加锁吗？)

购买商品整个事务流程（商品数量 <= 0 的情况下）

①：SELECT * FROM PRODUCT WHERE ID=3 查出商品信息。（对该记录 添加 共享锁）。

参考资料：

<https://blog.csdn.net/trusause/article/details/79487807>

问题1：

参考资料部分内容：

```
-----
T1:
begin tran
select * from table (holdlock) (holdlock意思是加共享锁，直到事物结束才释放)
update table set column1='hello'

T2:
begin tran
select * from table(holdlock)
update table set column1='world'
```

假设T1和T2同时达到select，T1对table加共享锁，T2也对加共享锁，当T1的select执行完，准备执行update时，根据锁机制，T1的共享锁需要升级到排他锁才能执行接下来的update.在升级排他锁前，必须等table上的其它共享锁释放，但因为holdlock这样的共享锁只有等事务结束后才释放，所以因为T2的共享锁不释放而导致T1等（等T2释放共享锁，自己好升级成排他锁），同理，也因为T1的共享锁不释放而导致T2等。死锁产生了。

升级成排它锁的条件是什么，什么情况下，共享锁会升级成排它锁？升级以后，锁定的粒度和范围会变化吗？

现在的共享锁是针对整个table表的，如果 update table set cloum1='hello' 再加上 WHERE ID =3 呢？

照样会升级吗？如果升级了，那么排它锁的范围不就改变了吗。。。

首先，共享锁和排他锁是不兼容的，所以，仅仅在T1中，SELECT 时，给table加了共享锁，holdlock的特性，事务结束之后才释放共享锁，那么，这样不也是死锁了么，共享锁一直不释放，排它锁一直获取不到。（除非上面说的共享 -升级-> 为 排它锁成立，才有可能导致T1本身不死锁。）

问题2：

参考资料部分内容：

死锁怎么解决呢？一种办法是，如下：

例6：

```
-----  
T1:  
begin tran  
select * from table(xlock) (xlock意思是直接对表加排他锁)  
update table set column1='hello'
```

```
T2:  
begin tran  
select * from table(xlock)  
update table set column1='world'
```

这样，当T1的select 执行时，直接对表加上了排他锁，T2在执行select时，就需要等T1事物完全执行完才能执行。排除了死锁发生。

但当第三个user过来想执行一个查询语句时，也因为排他锁的存在而不得不等待，第四个、第五个user也会因此而等待。在大并发情况下，让大家等待显得性能就太友好了，所以，这里引入了更新锁。

xlock有延迟释放锁直到事务结束的效果吗？？？

问题3：

可见事务隔离级别是通过改变锁来实现的。

3 何时加锁？

如何加锁，何时加锁，加什么锁，你可以通过hint手工强行指定，但大多是数据库系统自动决定的。这就是为什么我们可以不懂锁也可以高高兴兴的写SQL。

例15:

```
-----  
T1:  begin tran  
      update table set column1='hello' where id=1  
T2:  SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED -- 事物隔离级别为允许脏读  
      go  
      select * from table where id=1
```

这里，T2的select可以查出结果。如果事物隔离级别不设为脏读，则T2会等T1事物执行完才能读出结果。

数据库如何自动加锁的？

- 1) T1执行，数据库自动加排他锁
- 2) T2执行，数据库发现事物隔离级别允许脏读，便不加共享锁。不加共享锁，则不会与已有的排他锁冲突，所以可以脏读。

问题4：

如果事务1和事务2都对ID为3的同一记录进行了修改，并且都没有提交，那么事务3去读ID=3的这条记录时，读的是谁的值？

事务1：

Begin tran

UPDATE TABLE SET CONT =100 WHERE ID =1；

commit;

事务2：

Begin tran

UPDATE TABLE SET CONT =200 WHERE ID =1；

commit;

事务3：SELECT COUNT FROM TABLE WHERE ID =1;