Atitit db deadlock prblm cause and solu 数据库死锁原因与解决

　 在数据库中有两种基本的锁类型：排它锁（Exclusive Locks，即X锁）和共享锁（Share Locks，即S锁）。当数据对象被加上排它锁时，其他的事务不能对它读取和修改。加了共享锁的数据对象可以被其他事务读取，但不能修改。数据库利用这两种基本的锁类型来对数据库的事务进行并发控制。

目录

[1. 下面总结下这两种锁造成的常见的死锁情况与解决方案: 1](#_Toc30584)

[1.1. 一. 事务之间对资源访问顺序的交替 1](#_Toc10584)

[1.2. 二. 并发修改同一记录 2](#_Toc27535)

[2. Solu 2](#_Toc26712)

[2.1. 减少修改 思路尽可能减少修改 Replkace 换成insert on dulip update .. 2](#_Toc1763)

[2.2. 减少修改 触发器建立约束 if not ,refuse up 2](#_Toc3840)

[2.3. -解决死锁( Kill conn 3](#_Toc9246)

[2.4. -解决死锁( Conn timeout机制，连接池模式 3](#_Toc4694)

[2.5. 更快引擎的数据库 mem 3](#_Toc23847)

[2.6. 加锁法 单进程 sync同步 3](#_Toc8628)

[2.7. 加锁法 多进程 数据库枷锁 3](#_Toc29885)

[3. Other 3](#_Toc10885)

[4. merge\_sysnc 4](#_Toc2487)

[4.1. opensession a new sess 4](#_Toc752)

[5. multi session or multi fac 4](#_Toc18827)

[6. Ref 4](#_Toc15063)

# 下面总结下这两种锁造成的常见的死锁情况与解决方案:

## 一. 事务之间对资源访问顺序的交替

出现原因：

一个用户A 访问表A（锁住了表A），然后又访问表B；另一个用户B 访问表B（锁住了表B），然后企图访问表A；这时用户A由于用户B已经锁住表B，它必须等待用户B释放表B才能继续，同样用户B要等用户A释放表A才能继续，这就死锁就产生了。

解决方法：

这种死锁比较常见，是由于程序的BUG产生的，除了调整的程序的逻辑没有其它的办法。仔细分析程序的逻辑，对于数据库的多表操作时，尽量按照相同的顺序进行处理，尽量避免同时锁定两个资源，如操作A和B两张表时，总是按先A后B的顺序处理， 必须同时锁定两个资源时，要保证在任何时刻都应该按照相同的顺序来锁定资源。

## 二. 并发修改同一记录

出现原因：

　用户A查询一条纪录，然后修改该条纪录；这时用户B修改该条纪录，这时用户A的事务里锁的性质由查询的共享锁企图上升到独占锁，而用户B里的独占锁由于A有共享锁存在所以必须等A释放掉共享锁，而A由于B的独占锁而无法上升的独占锁也就不可能释放共享锁，于是出现了死锁。这种死锁由于比较隐蔽，但在稍大点的项目中经常发生。

　一般更新模式由一个事务组成，此事务读取记录，获取资源（页或行）的共享 (S) 锁，然后修改行，此操作要求锁转换为排它 (X) 锁。如果两个事务获得了资源上的共享模式锁，然后试图同时更新数据，则一个事务尝试将锁转换为排它 (X) 锁。共享模式到排它锁的转换必须等待一段时间，因为一个事务的排它锁与其它事务的共享模式锁不兼容；发生锁等待。第二个事务试图获取排它 (X) 锁以进行更新。由于两个事务都要转换为排它 (X) 锁，并且每个事务都等待另一个事务释放共享模式锁，因此发生死锁。

解决方法：

————————————————

版权声明：本文为CSDN博主「zxcodestudy」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/qq\_16681169/java/article/details/74784193

# Solu

## 减少修改 思路尽可能减少修改 Replkace 换成insert on dulip update ..

这样减少了一半的修改，one up one ce

## 减少修改 触发器建立约束 if not ,refuse up

CREATE TRIGGER `upchk` BEFORE UPDATE ON `football\_match\_t` FOR EACH ROW begin

if new.tee\_time=old.tee\_time and new.match\_status=old.match\_status then

SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = 'same stt n tee time';

end if;

## -解决死锁( Kill conn

#------解决死锁(补充法)

代码泄漏在所难免,,必须两手抓...

客户端,自动关闭timeout conn ,好像对timeout 事务不生效..只好自己写gc outtime...

服务器,设置 conn timeout,,而且lock timeout...

看门狗必须的..查询lock 或者过长时间sql ,kill....

## -解决死锁( Conn timeout机制，连接池模式

如果不能exe,timeout ,,close conn..auto..

## 更快引擎的数据库 mem

## 加锁法 单进程 sync同步

## 加锁法 多进程 数据库枷锁

先在加个锁lock表格(现成id，库表名称，记录号

在触发器里面判断枷锁，如果非加锁现成修改记录，直接抛出异常，拒绝修改

## 简化隔离级别到最低

## 增加异步future超时模式

如果此记录死锁跳过

# Other

atitit.减少数据库死锁总结o91 fx

# merge\_sysnc

## opensession a new sess

# multi session or multi fac

# Ref

数据库常见死锁原因及处理\_zhoxing-CSDN博客