**数据库中的锁**

**锁的概念和分类**

**-> 什么是锁**

1. 数据库默认为表添加的锁的权限是 表 的级别的，也可以更改权限
2. 锁的使用能够减少并发情况的发生，但是特殊情况下也可能出现 死锁

**-> 锁的优化方式**

1. 优化原则：尽早地释放掉锁的资源

**-> X锁 排他锁**

1. 当一个用户对数据库中的表进行 增、删、改 的操作的时候，数据库就会为表添加上一个X锁
2. 当一个表中存在X锁的时候，只能是当前用户能对这张表进行访问，其他用户不能够对这张表进行访问
3. X锁只能添加在一个没有锁的表中，当一个表中已经存在了锁，那么就不能再为这张表添加X锁

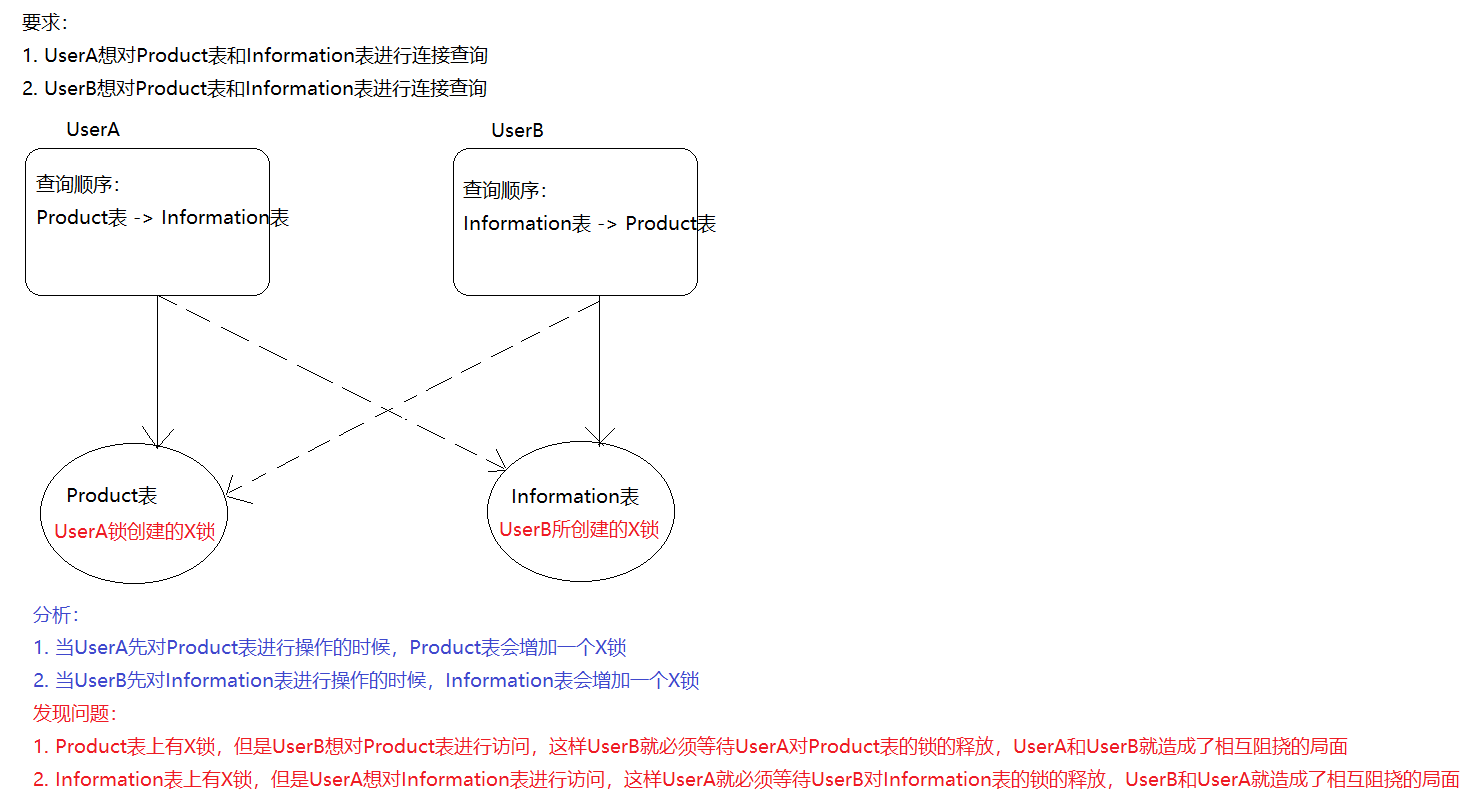
**-> S锁 共享锁**

1. 当一个用户对数据库中的表进行 查找 的操作的时候，数据库就为会表添加上一个S锁
2. 当一个表中存在S锁的时候，其他用户只能对这张表进行 查找 的操作：因为当我们进行 增、删、改 的操作的时候，数据库就会为表添加一个X锁，但是X锁只能够添加在没有锁的表中
3. 一个表能够同时存在多个共享锁：当一个表中已经有一个共享锁，而其他用户又对这张表进行 查找 的操作的时候，这时候表又会添加一个对应这个用户的S锁

**一些死锁的场景**

**-> 第一种类型**

当多个进程同时访问数据库中的数据，并且进程之间操作表的顺序不一致，相互等待对方释放资源、相互阻挠的时候就很有可能发生死锁的情况，如下图：



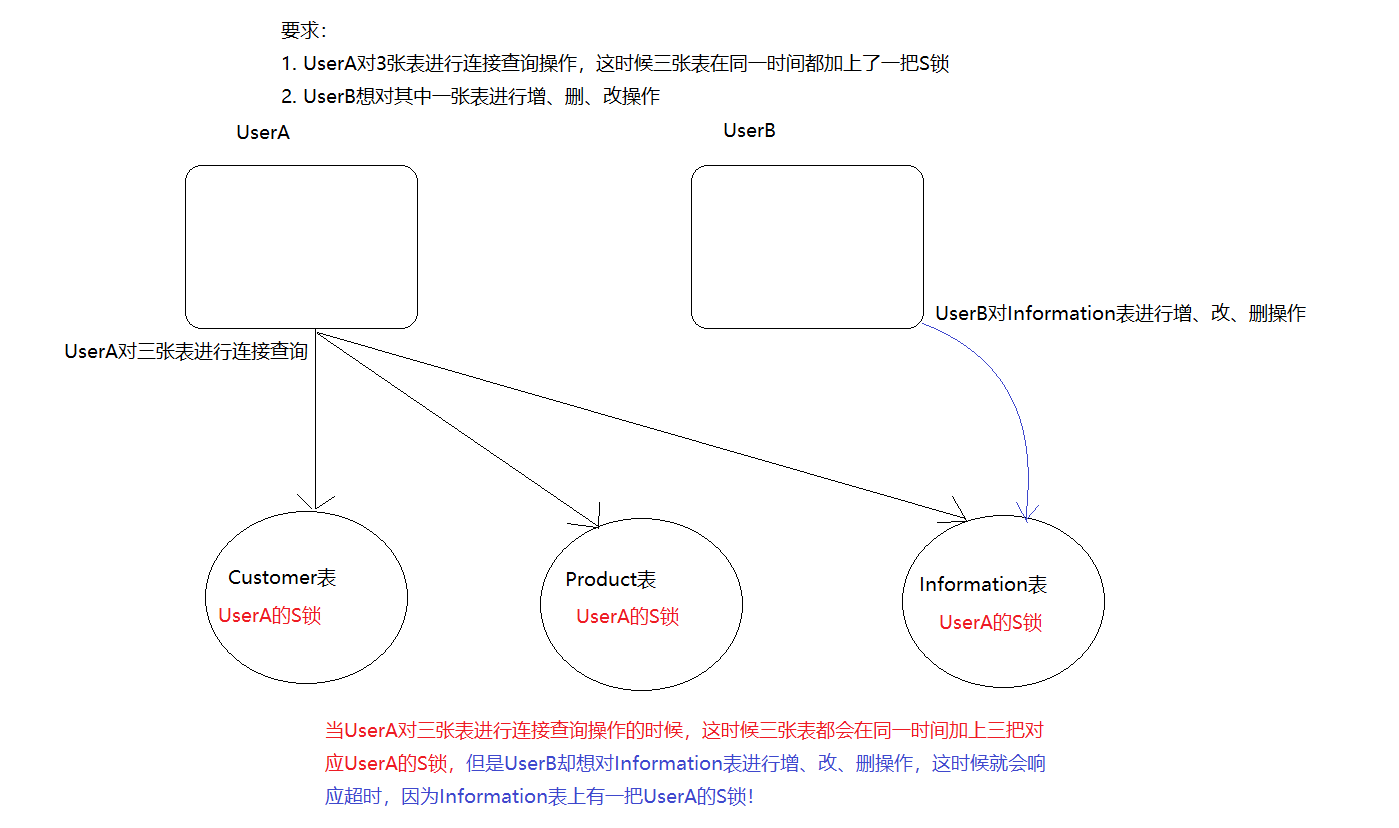
UserA和UserB都想对Product表和Information表进行连接查询，但是顺序就不一致，UserA先操作的Product表，这样Product表就会有一个UserA的会话所创建的X锁，UserB先操作Information表，这样Information表就会有一个UserB的会话所创建的X锁，但是这时候UserA对Product表的操作完了、UserB对Information表的操作完了，这时候UserA想去操作Information表、UserB想去操作Product表，但是这两个表中都有锁了，并且两个client（UserA、UserB）的会话还没有结束，这样两者之间就相互阻挠、互相等待对方的资源释放，这样就造成了死锁

**解决方法**

让两个client（UserA、UserB）的操作顺序一致，如上图，我们让UserA、UserB的操作顺序都是先操作Product表再操作Information表，这样就不会相互阻挠，UserA先操作的Product表，UserB等待UserA对Product表操作的结束再去操作Product表，然后UserA操作Information表，当UserB操作Product表结束后又去操作Information表，并且等待UserA操作的结束，通过让两者间的操作顺序保持一致，就能很大概率的减少死锁情况的发生

**-> 第二种类型**

当一个用户对多张表进行连接查询操作的时候，那么这些表在同一时间都会增加一个S锁，如果这时候还有其他用户对这些表进行增、删、改的操作的时候，就会造成连接超时的情况（因为S锁的存在，其他用户不允许对该表进行增、删、改的操作），也就是死锁，如下图：



当一个用户对原始表进行操作并且占用且加锁的时候，其余用户再对原始表进行操作，就有可能发生死锁的情况

**解决方法**

我们可以建立如上图所示的三张表的本地临时表，然后让UserA再对三张本地临时表进行连接查询操作，然后让UserB再对原始表进行增、改、查的操作的时候，就不用等待UserA对原始表的锁的释放了，这样就能够把两者分隔开了

**关于悲观锁和乐观锁的使用**

**上面所说的锁是关于数据库设计上所定义的锁，还有两种锁为程序员锁定义的锁，它们就是悲观锁和乐观锁**

1. 悲观锁

-> 假定会发生并发冲突，屏蔽一切可能违反数据完整性的操作

-> 悲观锁的场景是假定其他用户企图访问、更改你正在访问、更改的对象的概率很高的环境，因此在悲观锁的环境中，在你开始改变此对象之前就将该对象锁住，并且直到你提交了所作的更改之后才释放锁，这也铸造了悲观锁的缺陷，不论是页锁还是行锁，加锁的时间可能会很长，这样可能会长时间的锁定一个对象，限制其他用户访问，也就是说悲观锁的并发访问性不好

-> 像平时的 【lock】/【Monitor】 等锁定方式都是属于悲观锁的范畴

-> 如何使用悲观锁

1. 乐观锁