数据库并发控制

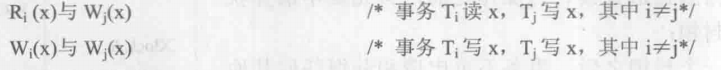
1. 数据不一致性：并发操作破坏了事务的隔离性
2. 丢失修改：T1和T2读入同一个数据并修改，T2提交的结果破坏了T1提交的结果，导致T1的修改丢失
3. 不可重复读（相对于之前的读取）：T1读取数据后，T2执行更新（修改，删除，插入）操作，使T1无法再现前一次读取结果
4. 读脏数据（相对于数据库中的数据）：T1修改某一数据并将其写回磁盘，T2读取后，T1由于某种原因被撤销，则此时T2读取的数据和数据库中的数据不一致。



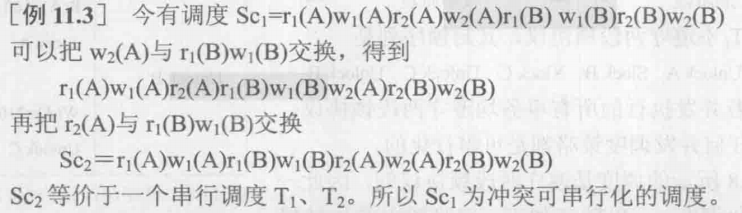
1. 并发控制技术：封锁、时间戳、乐观控制法、多版本并发控制（MVCC）
2. 封锁
3. 封锁类型：排他锁（X锁，写锁），共享锁（s锁，读锁）
4. 排他锁：T对数据A加上X锁后，T可以对A进行读取和修改，但是其他事物不可以对A添加任何类型的锁，直到T释放A上的X锁。
5. 共享锁：T对A加上S锁后，T可以读取A，但是不可以修改A。其他事务可以添加S锁，但是不能添加X锁。
6. 封锁协议：不同的封锁协议能达到的一致性级别不同
7. 一级封锁协议：事物T在修改数据R之前必须先添加X锁，直到事务结束（commit， rollback）才释放——解决了丢失修改问题。
8. 二级封锁协议：在一级封锁协议的基础上增加事务T在读取数据R之前必须先对其加S锁，读取完即可释放S锁——解决了丢失修改和读脏数据
9. 三级封锁协议：在一级封锁协议的基础上增加事务T在读取R之前必须先对其加S锁，直到事务结束才释放——可解决以上三种问题。



1. 活锁和死锁
2. 活锁：（一种“饥饿”状态）T1封锁了数据R，T2请求封锁R，只能先等着，T3，T4,T5也请求R。T1解锁之后，系统把锁给了T3，T4，T5… T2一直得不到锁。
   1. 简单解决方法：先到先服务
3. 死锁：T1封锁R1，请求R2,，T2封锁R2，请求R1。
   1. 死锁的预防：（死锁产生的条件：互斥条件，不可剥夺条件，请求和保持，循环等待）
      1. 一次封锁法（破坏“请求与保持”）：要求每个事务必须一次将所有要使用的数据全部加锁，否则就不能继续执行。
      2. 顺序封锁法（破坏“循环等待”）：预先对数据对象规定一个封锁顺序，所有事务都按照这个顺序封锁资源
   2. 死锁的诊断与解除
      1. 超时法：如果一个事务的等待时间超过了规定时限，就认为发生了死锁。
      2. 等待图法
4. 并发调度的可串行性
5. 可串行化调度：多个事务的并行执行是正确的，当且仅当其结果与按某一次序串行地执行这些事务的结果相同
6. 冲突操作：指不同事务对同一个数据的读写操作和写写操作。不同事务的冲突操作和同一事务的两个操作是不能交换（swap）的。



1. 一个调度SC在保证冲突操作的次序不变的情况下，通过交换两个事务不冲突操作的次序得到另一个调度sc，如果sc是串行地，称SC是冲突可串行化的调度（这是一个充分不必要条件）



1. 两段锁协议：为了保证并发调度的正确性，保证并发调度是可串行化的（前面的封锁协议是为了保证数据库的一致性）
   1. 两段锁协议：事务必须分两个阶段对数据项加锁和解锁。第一段获得封锁，事务可以获得任何数据项上的任何类型的锁，但是不能释放任何锁；第二阶段释放封锁，事务可以释放任何数据项的任何类型的封锁，但是不能在申请任何锁。
      1. 在对任何数据进行读写操作之前，首先要申请并获得对该数据的封锁；
      2. 在释放一个封锁之后，事务不再申请和获得任何其他封锁。
2. 其他并发控制方法
   1. 时间戳：给每一个事务盖上一个时标，即任务开始执行的时间。每个事务具有唯一的时间戳。如果发生冲突操作，就回滚具有较早时间戳的事务，以保证其他事务的正常执行，被回滚的事务被赋予新的时间戳并从头开始执行。
   2. 乐观控制法：让事务自由执行，事务提交之前进行正确性检查，如果检查后发现事务冲出现过冲突并影响了可串行性，则拒绝提交并回滚。
   3. 多版本控制法（MVCC）：

版本：指数据库中数据对象的一个快照，记录了数据对象某个时刻的状态

多版本并发控制利用物理存储上的多版本来维护数据的一致性。当一个事务检索数据库时，看到的是一个数据的一段时间之前的快照，而不管正在处理的数据当前的状态。和分所机制相比，主要的好处是消除了数据库中数据对象读和写操作的冲突，有效的提高了系统的性能。