并发控制概述

- 多事务执行方式
  - 事务串行执行
  - 交叉并发执行
  - 同时并发执行
- 事务并发带来的问题
  - 可能会存取和储存不正确的数据，破坏事务的一致性和数据库的一致性
  - 数据库管理系统必须提供并发控制机制
  - 并发控制机制是衡量一个数据库管理系统性能的重要标志之一
- 并发控制的任务
  - 保证事务的隔离性和一致性
  - 对并发操作进行正确调度
- 并发控制的主要技术
  - 封锁，时间戳，乐观控制法，多版本并发控制等
- 并发操作带来数据的不一致性
  - 丢失数据
  - 不可重复性
  - 读脏数据

并发操作可能带来什么问题？

- 两段锁协议（简称2PL）
  - 为了保证并发调度的正确性
  - 所谓两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁
    - 在对任何数据进行读写操作之前，首先申请并获得该数据的封锁
      - 第一阶段是获得封锁，也称为扩展阶段
    - 在释放一个封锁之后，事务不在申请和获得任何其他的封锁
      - 第二阶段是释放封锁，也称为收缩阶段
- 并发调度的可串行性
  - 串行调度一定是正确的
  - 可串行化调度　　多个事务的并发执行是正确的，当且仅当其结果与按一次序串行地执行这些事务时的结果相同，称这种调度策略为客串行调度
  - 可串行性　　并发事务调度的准则
  - 冲突可串行化调度　　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外一个调度SC'，如果SC'是串行的，称调度SC为冲突可串行化的调度
- 封锁的粒度
  - 封锁对象的大小称为封锁力度
  - 封锁粒度与系统开发度和并发控制的开销密切相关
    - 封锁的粒度越大，数据库所能封锁的数据单元就越少，并发度就越小，系统开销也越小
    - 数据库的粒度越小，并发度较高，但系统开销也越大
  - 多粒度封锁　　对一个节点加锁意味着这个节点的所有后裔节点也被加以同样类型的锁
  - 意向锁　　如果对一个节点加意向锁，则说明该节点的下层节点正在被加锁
- 封锁
  - 什么是封锁
    - 封锁时实现并发控制的一个非常重要的技术
    - 事务T在对某个数据对象（如表，记录等）操作之前，先向系统发送请求，对其加锁
    - 加锁后事务T就对该数据对象有了一定的控制，在事务T释放它的锁之前，其他对象不能更行此数据对象
  - 基本类型
    - 排他锁（exclusive locks）
      - 排他锁又称为写锁
      - 若事务T对数据对象A加了X锁，则只允许T读写和修改A，其他任何事务都不能再对A加任何类型的锁，直到T释放A上的锁为至
    - 共享锁（share locks）
      - 共享锁又称为读锁
      - 若事务T对数据对象A加了S锁，则事务T可以读A但不能修改A，其他事务只能再对A加S锁，而不能加X锁，直到T释放A上的S锁为止
    - 相容矩阵

T1 \ T2	X	S	-
X	N	N	Y
S	N	Y	
-	Y	Y	Y

子主题

两种锁，以及他们的特点

  - 封锁协议
    - 一级封锁协议
      - 事务T在修改数据R之前必须先对其加X锁，直到事务结束才释放
      - 防止丢失修改，并保证事务T是可恢复的
    - 二级封锁协议
      - 在一级封锁协议基础上增加事务T在读取数据R之前必须先对其加S锁，读完后即可释放S锁
      - 防止丢失修改，进一步防止读脏数据
    - 三级封锁协议
      - 在一级封锁协议的基础上增加事务T在读取数据R之前必须先对其加S锁，知道事务结束后才释放
      - 防止丢失修改和读脏数据，进一步可以防止不可重复读
  - 封锁带来的问题
    - 活锁
      - 线程并没有阻塞，所以是活的状态，但是在做无用功。
      - 避免方法　　采用先来先服务的策略
    - 死锁
      - 两个线程都处于阻塞状态，在等待其他进程释放锁
      - 死锁的预防
        - 一次封锁法
        - 顺序封锁法
      - 死锁的诊断
        - 超时法
        - 等待图法
      - 死锁的解除　　选择一个处理死锁代价最小的事务，将其撤销，释放此事务持有的所有的锁，使其他事务得以继续运行下去

如何预防死锁