第十一章 并发控制

- 前言
  - 数据库的重要特征是它能为多个用户提供数据共享。DBMS允许共享的用户数目是DBMS重要标志之一。
  - DBMS必须提供**并发控制机制**来*协调并发用户的并发操作*以保证并发事务的**隔离性**和**一致性**，**保证数据库的一致性**
- 并发控制是什么
  - 多用户数据库系统
    - 数据库是共享资源，应该允许多个用户使用
    - 允许多个用户同时使用的数据库系统称为：多用户数据库系统
  - 事务的并行和串行
    - 一个一个地串行执行，即每个时刻只有一个用户程序执行对数据库的存取（许多系统资源将处于空闲状态）
    - 为了充分发挥数据库共享资源的特点，提高系统效率，应允许多个用户程序并行地存取数据库，这样就会产生多个用户程序同时存取同一数据的情况
      - 交叉并发
      - 同时并发
  - 并发的问题
    - 若对并发操作不加控制就可能会存取和存储不正确的数据，破坏数据库的完整性
- 并发控制概述
  - 并发操作带来的数据不一致性包括三类：
    - 丢失修改

T1	T2
Read(A)=16 A=A-1 Write(A)=15	Read(A)=16 A=A-1 Write(A)=15

    - 不可重复读
      - 不可重复读是指事务T1读取数据后，事务T2执行更新操作，使T1无法再现前一次读取结果。具体地讲，不可重复读包括两种情况：
        - 事务T1读取某一数据后，事务T2对其做了修改，当事务T1再次读该数据时，得到与前一次不同的值。

T1	T2
Read(A)=50 Read(B)=100 Read(B)=200	 Read(B)=100 B=B*2 Write(B)=200

- 幻象读

T1	T2
Read(T表元组数)=10 Read(T表元组数)=11	Insert into t values(...)

  - 事务T1按一定条件从数据库中读取了某些数据记录后，事务T2删除了其中部分记录，当T1再次按相同条件读取数据时，发现某些记录消失了。
  - 事务T1按一定条件从数据库中读取某些数据记录后，事务T2插入了一些记录，当T1再次按相同条件读取数据时，发现多了一些记录

- 读“脏”数据

T1	T2
Read(C)=100 C=C*2 Write(C)=200 Rollback	 Read(C)=200

  - 读“脏”数据是指事务T1修改某一数据，并将其写回磁盘，事务T2读取同一数据后，T1由于某种原因被撤消，这时T1已修改过的数据恢复原值，T2读到的数据就与数据库中的数据不一致，则T2读到的数据就为“脏”数据，即不正确的数据。

- 封锁
  - 封锁是实现并发控制的一个非常重要的技术。利用封锁机制，可以解决前面所讲的三个问题
  - **封锁的定义**
    - 就是事务T在对某个数据对象例如表、记录等操作之前，先向系统发出**加锁请求**
    - 在获得封锁后，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，其他的事务不能更新此数据对象
  - 封锁的基本类型

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

Y:相容的请求

N:不相容的请求

    - **X 排它锁（写锁，Exclusive Locks，简称X锁）**
    - **S 共享锁（读锁，Share Locks，简称S锁）**
- 封锁协议
  - **封锁协议（Locking Protocol）**
    - 对数据对象加锁时，还需要约定一些规则
    - 例如*应何时申请X锁或S锁、持锁时间、何时释放*等。我们称这些规则为封锁协议（Locking Protocol）

- 1级封锁协议
  - 1级封锁协议是：
    - 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*。事务结束包括**正常结束（COMMIT）**和**非正常结束（ROLLBACK）**。
  - 1级封锁协议**可防止丢失修改**，并**保证事务T是可恢复的**。
  - 在1级封锁协议中，如果仅仅是读数据不对其进行修改，是不需要加锁的，所以它**不能保证可重复读**和**不读“脏”数据**。
- 2级封锁协议
  - 2级封锁协议是：
    - 1级封锁协议加上事务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。
  - 2级封锁协议除防止了丢失修改，还可**进一步防止读“脏”数据**
- 3级封锁协议
  - 3级封锁协议是：
    - **1级封锁协议**加上事务T在读取数据R之前必须先对其加S锁，直到事务结束才释放。
  - 3级封锁协议除防止了丢失修改和不读‘脏’数据外，还**进一步防止了不可重复读**
- 补充：隔离级别与不一致
  - 实践中采用事务隔离级别来解决并发不一致问题

隔离级别	读“脏”	不可重复读	幻象读
未提交读	Y	Y	Y
提交读	N	Y	Y
可重复读	N	N	Y
可串行读	N	N	N

  - 丢失修改问题
    - 当两个事务检索相同的行，然后基于原检索的值对行进行更新时，会发生丢失更新。如果要避免此情况，必须将隔离级别设置为“可重复读”或更高。
    - 如果两个事务使用一个 UPDATE 语句更新行，并且不基于以前检索的值进行更新，则在提交读隔离级别也不会发生丢失更新
  - **可串行读**是事务隔离的最高级别，事务之间完全隔离，不会出现并发事务的任何不一致冲突
- 活锁和死锁（封锁的方法可能引起活锁和死锁）
  - 活锁
    - 解决办法：先来先服务
  - 死锁
    - 解决：防、治两种
      - 死锁的预防（用的少）

- 产生死锁的原因是
  - 两个或多个事务都已封锁了一些数据对象，然后又都请求对已为其他事务封锁的数据对象加锁，从而出现死等待。
  - 防止死锁的发生其实就是要破坏产生死锁的条件。预防死锁通常有两种方法：
    - 一次封锁法
      - 一次封锁法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。
      - 可以有效地防止死锁的发生
      - 也存在问题
        - 一次就将以后要用到的全部数据加锁，势必扩大了封锁的范围，从而降低了系统的并发度
    - 顺序封锁法
      - 顺序封锁法是预先对数据对象规定一个封锁顺序，所有事务都按这个顺序实行封锁
      - 顺序封锁法可以有效地防止死锁
      - 同样存在问题
        - 事务的封锁请求可以随着事务的执行而动态地决定，很难事先确定每一个事务要封锁哪些对象，因此也就很难按规定的顺序去施加封锁
- 死锁的诊断与解除
  - 超时法
    - 如果一个事务的等待时间超过了规定的时限，就认为发生了死锁。
    - 超时法实现简单，但其不足也很明显。
      - 一是有可能误判死锁，事务因为其他原因使等待时间超过时限，系统会误认为发生了死锁。
      - 二是时限若设置得太长，死锁发生后不能及时发现
  - 等待图法
    - 事务等待图是一个有向图G=(T,U)。
      - T为结点的集合，每个结点表示正运行的事务；
      - U为边的集合，每条边表示事务等待的情况。
    - 若T1等待T2,则T1、T2之间划一条有向边，从T1指向T2。
    - 事务等待图动态地反映了所有事务的等待情况。*并发控制子系统周期性地（比如每隔1分钟）检测事务等待图*，如果**发现图中存在回路**，则**表示系统中出现了死锁**
- 并发调度的可串行性

- 计算机系统对并发事务中并发操作的调度是随机的，而不同的调度可能会产生不同的结果，那么哪个结果是正确的，哪个是不正确的呢?
- 事务串行执行的结果是正确的(事务间互相没有干扰)。
- **可串行化（Serializable）的调度**
- 冲突可串行化调度
  - 判断可串行化调度的充分条件
  - **操作序列**
  - **冲突操作**
    - 不同的事务对同一个数据的读写操作和写写操作
  - **冲突等价的调度**
    - 如果调度*S*只交换一对**非冲突指令**的次序，就变成调度*S'*，则二者称为 冲突等价
      - 不同事务之间的冲突操作不能改变前后顺序
      - 同一事务的任何两个操作不能改变前后顺序
  - **冲突可串行化调度**
    - **冲突可串行化是可串行化调度的充分条件，不是必要条件**
    - 如调度S冲突等价于串行调度S'，则称为该调度为冲突可串行化的调度
    - 可串行性是并发事务正确性的准则。
      - 按这个准则规定，一个给定的并发调度，当且仅当它是可串行化的，才认为是正确调度
      - DBMS普遍采用封锁方法实现并发操作调度的可串行性，从而保证调度的正确性
      - **两段锁协议**就是保证并发调度可串行性的封锁协议
- 两段锁协议（2PL）
  - 实现并发控制的主要方法
  - 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
      - 1. 扩展阶段：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，而且
      - 2. 收缩阶段：在释放一个封锁之后，事务不再申请和获得任何其他封锁
    - **并发执行的所有事务均遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的**
    - 一次封锁法遵守两段锁协议
    - 遵守两段锁协议的事务可能发生**死锁**
- 封锁的粒度
  - 封锁粒度
    - **封锁对象的大小**

- 封锁对象可以是
  - 逻辑单元：
    - · 属性值，属性值的集合
    - · 元组，关系
    - · 索引项，整个索引
    - · 整个数据库
  - 也可以是物理单元：
    - · 页（数据页或索引页）
    - · 块等
- 多粒度封锁
  - 允许多粒度树中每个节点被独立加锁

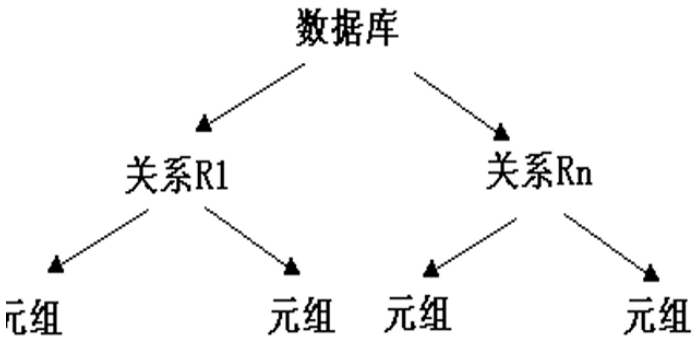

图8.8　三级粒度树。

    - 多粒度树
    - 对一个结点加锁，意味着这个结点的所有后裔结点也被假意同类型的锁
  - 显式封锁
  - 隐式封锁
- 意向锁
  - 意向锁的含义
  - **1. IS锁**
  - **2. IX锁**
  - **3. SIX锁**
  - 相容性矩阵

	IS	IX	S	S IX	X
IS	✓	✓	✓	✓	×
IX	✓	✓	×	×	×
S	✓	×	✓	×	×
S IX	✓	×	×	×	×
X	×	×	×	×	×

- MVCC（多版本并发控制）
  - MVCC维持一个数据的多个版本
    - 数据元素X上的每一个写操作产生X的一个新版本
    - 为X的每一个读操作选择一个版本
  - 读与写之间没有冲突
  - 数据库写只等待正在对同一行数据进行更新的写

以上内容整理于 幕布文档