第十五章 并发控制

一、锁协议

1.分类

- 共享锁(shared):只能读，记为S
- 排他锁(exclusive):可以读和写，记为X

2.地位

事务只有在并发控制管理器授予所需锁后才能继续其操作

3.锁相容

	S	X
S	true	false
X	false	false

4.指令

一个事务通过执行$\boldsymbol{lock\text{-}S}(Q)$指令来申请数据项Q上的共享锁

一个事务通过执行$\boldsymbol{lock\text{-}X}(Q)$指令来申请数据项Q上的排他锁

一个事务通过$\boldsymbol{unlock}(Q)$指令来释放数据项Q上的锁

二、两阶段锁协议

1.作用:

用来保证冲突可串行化的方式

2.要求

- 第一阶段：增长阶段
  - 只能对事务进行加锁的操作，不能进行释放锁的操作
- 第二阶段：缩减阶段
  - 只能对事务进行释放锁的操作，不能进行加锁的操作

三、严格两阶段锁协议

1.作用:

用来保证冲突可串行化的方式

2.要求

- 第一阶段：增长阶段
  - 只能对事务进行加锁的操作，不能进行释放锁的操作
- 第二阶段：缩减阶段
  - 只能对事务进行释放锁的操作，不能进行加锁的操作
- 在当前事务提交或回滚之前，不能够释放排他锁

四、强两阶段锁协议

1.作用:

用来保证冲突可串行化的方式

2.要求

- 第一阶段：增长阶段
  - 只能对事务进行加锁的操作，不能进行释放锁的操作
- 第二阶段：缩减阶段
  - 只能对事务进行释放锁的操作，不能进行加锁的操作
- 在当前事务提交或回滚之前，不能释放任何锁

五、锁转换

1.第一阶段：增长阶段

- 可以申请共享锁
- 可以申请排他锁
- 可以将共享锁转换为排他锁（升级）

2.第二阶段：缩减阶段

- 可以释放共享锁
- 可以释放排他锁
- 可以将排他锁转换为共享锁（降级）

六、死锁处理

1.方法：

- 死锁预防协议->不让系统进入死锁状态
  - 如果系统进入死锁状态的概率相对较高，通常使用死锁预防机制
- 死锁检测+死锁恢复->允许系统进入死锁状态

2.死锁预防

- 使用抢占和事务回滚
  - **wait-die(非抢占):**当事务T_i申请的数据项当前被T_j持有，仅当T_i的时间戳小于T_j的时间戳，允许T_i等待。否则，T_i回滚。（新人不抢老人的活）
  - **wound-wait(抢占):**当事务T_i申请的数据项当前被T_j持有，仅当T_i的时间戳大于T_j的时间戳，允许T_i等待。否则，T_i回滚。（新人抢老人的活）
- 锁超时
  - 申请锁的事务至多等待一段给定的时间。若在此期间未授予该事务锁，则称该事务超时，此时事务自己重启并回滚。

六、死锁处理

3.死锁的检测与恢复

- 死锁的检测
  - 画等待图，没有环路就没有死锁
    - $T_i \rightarrow T_j$，意为事务T_i在等待事务T_j释放所需的数据项
    - 当事务T_j不再持有事务T_i所需数据项时，这条边才从等待图中删除
- 从死锁中恢复
  - 解除死锁最通常的做法是回滚一个或多个事务，步骤如下
    - 选择牺牲者
      - “最小代价”
    - 回滚
      - 分为彻底回滚和部分回滚
    - 饿死
      - 为了避免饿死，我们需要规定每个事务被当作牺牲者次数的上限（可以在代价因素中包含回滚的次数）