第10章

并发操作带来的数据不一致

1.丢失修改
2.不可重复读
3.读"脏"数据

丢失修改

事务 1 与事务 2 从数据库中读入同一数据并修改，事务 2 的提交结果破坏了事务 1 提交的结果,导致事务 1 的修改被丢失。

不可重复读

不可重复读是指事务 1 读取数据后,事务2执行更新操作,使事务 1 无法再现前一次读取结果。分为以下三类：
１．事务 2 对其做了修改,当事务 1 再次读该数据时,得到与前一次不同的值。
２．事务2删除了其中部分记录,事务1再次读取数据时,发现某些记录消失了。
３．事务2插入了一些记录,当事务1再次按相同条件读取数据时,多了一些记录。
后两种不可重复读有时也称为幻影现象

读“脏”数据

事务1修改某一数据,并将其写回磁盘,事务2读取同一数据后事务1由于某种原因被撤消,这时事务1已修改过的数据恢复原值事务2读到的数据就与数据库中的数据不一致,是不正确的数据,又称为“脏”数据。

数据不一致性

由于并发操作破坏了事务的隔离性
并发控制就是要用正确的方式调度并发操作，使一个用户事务的执行不受其他事务的干扰，从而避免造成数据的不一致性

并发控制的主要技术

封锁(商用的一般采用)
时间戳
乐观控制法

封锁

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,其它的事务不能更新

此数据对象。封锁是实现并发控制的一个非常重要的技术

分类

排它锁（X 锁）
共享锁（S 锁）

X 锁

若事务T对数据对象A加上X锁,则只允许T读取和修改A,其它任何事务都不能再对A加任何类型的锁,直到T释放A上的锁保证其他事务在T释放A上的锁之前不能再读取和修改 A

S 锁

若事务T对数据对象A加上S锁,则其它事务只能再对A加S锁,而不能加X锁,直到T释放A上的S锁保证其他事务可以读A,但在T释放A上的S锁之前不能对A做任何修改

	X	**S**	**-**
X	N	N	Y
S	N	Y	Y
-	Y	Y	Y

封锁协议

1 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
正常结束 (COMMIT)
非正常结束 (ROLLBACK)
1 级封锁协议可防止丢失修改
在 1 级封锁协议中,如果是读数据,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

1 级封锁协议 + 事务 T 在读取数据 R 前必须先加 S 锁,读完后即可释放 S 锁
2 级封锁协议可以防止丢失修改和读“脏”数据。
在 2 级封锁协议中,由于读完数据后即可释放 S 锁,所以它不能保证可重复读 。

3级封锁协议

1级封锁协议+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 级封锁协议可防止丢失修改、读脏数据和不可重复读。

	X锁		S锁		一致性保证		
	操作结束释放	事务结束释放	操作结束释放	事务结束释放	不丢失修改	不读"脏"数据	可重复读
1级封锁协议		√			√		
2级封锁协议		√	√		√	√	
3级封锁协议		√		√	√	√	√

活锁和死锁

活锁

指的是任务或者执行者没有被阻塞，由于某些条件没有满足，导致一直重复尝试—失败—尝试—失败的过程。
解决办法：按请求封锁的先后次序对这些事务排队，该数据对象上的锁一旦释放，首先批准申请队列中的第一个事务获得锁

死锁

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。
解决办法：
１．一次封锁法：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
存在问题：降低了并发度，扩大了封锁范围，难以事先精确确定封锁对象
２．顺序封锁法：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。
存在问题：维护成本高；难于实现；
DBMS 在解决死锁的问题上更普遍采用的是诊断并解除死锁的方法
１．允许死锁发生
２．解除死锁（由 DBMS 的并发控制子系统定期检测系统中是否存在死锁，一旦检测到死锁,就要设法解除）

检测死锁

１．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
优点：实现简单
缺点：时限太短可能误判死锁，太长可能死锁不能及时发现
２．等待图法：用事务等待图动态反映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 min)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。（ＰＳ：简单点说就是循环扫描，看图中是否存在环）

解除死锁

选择一个处理死锁代价最小的事务,将其撤消,释放此事务持有的所有的锁,使其它事务能继续运行下去。

正确的调度

将所有事务串行起来的调度策略一定是正确的调度策略（串行一定正确）
几个事务并行执行是正确的当且仅当其结果和按某一次串行的执行他们时的结果相同，这种并行调度策略称为可串行化的调度

可串行化调度的充分条件

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' 是串行的,称调度 Sc 为冲突可串行化的调度，一个调度是冲突可串行化,一定是可串行化的调度

冲突操作

指不同的事务对同一个数据的读写操作和写写操作，其他操作是不冲突操作，不同事务的冲突操作和同一事务的两个操作不能交换，冲突可串行化调度是可串行化调度的充分条件,不是必要条件。还有不满足冲突可串行化条件的可串行化调度

两段锁协议

指所有事务必须分两个阶段对数据项加锁和解锁
1. 在对任何数据进行读、写操作之前,事务首先要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。
并发执行的所有事务均遵守两段锁协议,则对这些事务的所有并发调度策略都是可串行化的。

封锁粒度

封锁对象的大小称为封锁粒度
在关系数据库中,封锁对象:
1，逻辑单元 : 属性值、属性值集合、元组、关系、索引项、整个索引、整个数据库等
2．物理单元:页(数据页或索引页)、物理记录等

描述	程度
封锁粒度	大
系统被封锁的对象	少
并发度	小
系统开销	小

需要处理多个关系的大量元组的用户事务:以数据库为封锁单位;
需要处理大量元组的用户事务:以关系为封锁单元;
只处理少量元组的用户事务:以元组为封锁单位

多粒度封锁

多粒度树，以树形结构来表示多级封锁粒度，根结点是整个数据库,表示最大的数据粒度，叶结点表示最小的数据粒度
允许多粒度树中的每个结点被独立地加锁
对一个结点加锁意味着这个结点的所有后裔结点也被加以同样类型的锁
在多粒度封锁中一个数据对象可能以两种方式封锁:显式封锁和隐式封锁
(类似与二叉树上的lazy标记，查讯某个节点加没加锁就类似与pushdown操作)

意向锁

目的：提高对某个数据对象加锁时系统的检查效率
对任一结点加基本锁,必须先对它的上层结点加意向锁
如果对一个结点加意向锁,则说明该结点的下层结点正在被加锁
(类比于flag标记这个子树是否都可选择)
１．意向共享锁（IS锁）
如果对一个数据对象加 IS 锁,表示它的后裔结点拟(意向)加 S 锁。
２．意向排它锁（IX锁）
如果对一个数据对象加 IX 锁,表示它的后裔结点拟(意向)加 X 锁。
３．共享意向排它锁（SIX锁）
如果对一个数据对象加 SIX 锁,表示对它加　S 锁,再加 IX 锁,即 SIX = S + IX 。

	S	**X**	**IS**	**IX**	**SIX**	**-**
S	Y	N	Y	N	N	Y
X	N	N	N	N	N	Y
IS	Y	N	Y	Y	Y	Y
IX	N	N	Y	Y	N	Y
SIX	N	N	Y	N	N	Y
-	Y	Y	Y	Y	Y	Y

锁的强度

是指它对其他锁的排斥程度
一个事务在申请封锁时以强锁代替弱锁是安全的,反之则不然
具有意向锁的多粒度封锁方法
１．申请封锁时应该按自上而下的次序进行
２．释放封锁时则应该按自下而上的次序进行
功能：
１．提高了系统的并发度
２．减少了加锁和解锁的开销
３．在实际的数据库管理系统产品中得到广泛应用