《数据库系统》作业-第十一章

姓名：任宇 学号：33920212204567

1. 在数据库中为什么要并发控制？并发控制技术能保证事务的哪些特性？
   答：数据库是一个共享资源，可以供多个用户使用。当多个事务并发地存取数据库时就会产生同时读取和/或修改统一数据的情况。如果对并发操作不加控制就可能会存储和存取不正常的数据，破坏数据的一致性。所以数据库必须提供并发控制机制。
   并发控制机制可以保证事务的一致性和隔离性。

2. 并发操作可能会产生哪几类数据不一致？用什么方法能避免各种不一致的情况？
   答：产生的数据不一致：
   （1）丢失操作：两个事务 T1 和 T2 读入统一数据并修改，T2 提交的结果覆盖了 T1 提交的结果，导致 T1 的修改丢失。
   （2）不可重复读：T1 读取某一数据后，T2 对其执行更新操作，导致 T1 无法再现前一次的读取结果。
   （3）读“脏”数据：事务 T1 修改某一数据，并将其写入磁盘，事务 T2 读取统一数据后，T1 由于某种原因被撤销，这是 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，T2 读到的数据就成为“脏”数据，即不正确的数据。
   避免方法：
   采用并发控制，常用的并发控制技术包括封锁方法、时间戳方法、乐观控制方法和多版本并发控制方法等。

3. 什么是封锁？基本的封锁类型有哪几种？试述它们的含义。
   答：封锁就是事务 T 在对某个数据对象例如表、记录等操作之前，现象系统发出请求，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，其他的事务不能更新或者读取此数据对象。
   基本的封锁类型有两种：排他锁和共享锁。
   排他锁又称为写锁。若事务 T 对数据对象 A 加上 X 所，则只允许 T 读取和修改 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。这就保证了其他事物在 T 释放 A 上的锁之前不能再读取和修改 A。
   共享锁又称为读锁。若事务 T 对数据对象 A 加上 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。这就保证了其他事务可以读 A,但在 T 释放 A 上的 S 锁之前不能对 A 做任何修改。

4. 如何用封锁机制保证数据的一致性?
   答：
   DBMS 在对数据进行读写操作之前首先对该数据执行封锁操作，例如事务 T1 在对 A 进行修改之前先对 A 执行 Xlock(A)即对 A 加 X 锁。这样，当 T2 请求

对 A 加 X 锁时就被拒绝，T2 只能等待 T1 释放 A 上的锁后才能获得对 A 的 X 锁，这时它读到的 A 是 T1 更新后的值，再按此新的 A 值进行运算。这样就不会丢失 T1 的更新。DBMS 按照一定的封锁协议对并发操作进行控制，使得多个并发操作有序地执行，就可以避免丢失修改、不可重复读和读“脏”数据等数据不一致性。

5. 什么是活锁?试述活锁的产生原因和解决方法。
答：如果事务 T1 封锁了数据 R,事务 T2 又请求封锁 R,于是 T2 等待。T3 也请求封锁 R,当 T1 释放了 R 上的封锁之后系统首先批准了 T3 的请求,T2 仍然等待。然后 T4 又请求封锁 R,当 T3 释放了 R 上的封锁之后系统又批准了 T4 的请求……T2 有可能永远等待，这就是活锁的情形。活锁的含义是该等待事务等待时间太长，似乎被锁住了，实际上可能被激活。
活锁产生的原因：当一系列封锁不能按照其先后顺序执行时，就可能导致一些事务无限期等待某个封锁，从而导致活锁。
避免活锁的简单方法是采用先来先服务的策略。当多个事务请求封锁同一数据对象时，封锁子系统按请求封锁的先后次序对事务排队，数据对象上的锁一旦释放就批准申请队列中第一个事务获得锁。

6. 什么是死锁?请给出预防死锁的若干方法。
答：如果事务 T1 封锁了数据 R1,T2 封锁了数据 R2,然后 T1 又请求封锁 R2,因 T2 已封锁了 R2,于是 T1 等待 T2 释放 R2 上的锁。接着 T2 又申请封锁 R1 因 T1 已封锁了 R1,T2 也只能等待 T1 释放 R1 上的锁。这样就出现了 T1 在等待 T2,而 T2 又在等待 T1 的局面，T,和 T2 两个事务永远不能结束，形成死锁。
预防死锁通常有两种方法：
（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加锁，否则就不能继续执行。
（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。

7. 请给出检测死锁发生的一种方法？当发生死锁后如何解除死锁？
答：DBMS 中诊断死锁的方法与操作系统类似，一般使用超时法或者事物等待图法。超时法是指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。DBMS 并发控制子系统检测到死锁后，就要设法解除。通常采用的方法是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锁，使其他事务得以继续运行下去。

8. 什么样的并发调度才是正确的并发调度？
答：可串行化的调度是正确的调度。可串行化的调度的定义：多个事务的并发执行是正确的，当且仅当其结果与按某一次序串行地执行它们时的结果相同，称这种调度策略为可串行化的调度。

9. 设 T1、T2、T3 是如下的三个事务，设 A 的初值为 0。
T1：A：= A+2；
T2：A：= A*2；

T3：A：= A*A；（A<-A^2）

1) 若这三个事务允许并发执行，则有多少种可能的正确结果？请一一列举出来。
2) 请给出一个可串行化的调度，并给出执行结果。
3) 请给出一个非串行化的调度，并给出执行结果。
4) 若这三个事务都遵守两阶段封锁协议，请给出一个不产生死锁的可串行化调度。
5) 若这三个事务都遵守两阶段封锁协议，请给出一个产生死锁的调度。

答：

1) 可能有 4 种正确的结果。若 T1 T2 T3，则结果为 16；若 T1 T3 T2，则结果为 8；若 T2 T1 T3，则结果为 4；若 T2 T3 T1，则结果为 2；若 T3 T1 T2，则结果为 4；若 T3 T2 T1，则结果为 2。
2) 调度如下，结果为 16

T1	T2	T3
Slock A Y=A=0 Unlock A XLock A A=Y+2 写回 A Unlock A	 Slock A 等待 等待 等待 Y=A=2 Unlock A Xlock A A=Y*2 写回 A Unlock A	 Slock A 等待 等待 等待 Y=A Unlock A Xlock A A=Y*Y 写回 A Unlock A

3) 调度如下，结果为 0

T1	T2	T3
Slock A Y=A Unlock A Xlock A 等待 A=Y+2	 Slock A Y=A Unlock A	

写回 A Unlock A		Slock A 等待 Y=A Unlock A Xlock A
	Xlock A 等待 等待 等待 A=Y*2 写回 A Unlock A	Y=Y*Y 写回 A Unlock A

4) 调度如下

T1	T2	T3
Slock A Y=A Xlock A A=Y+2 写回 A Unlock A Unlock A	Slock A 等待 等待 Y=A Xlock A 等待 A=Y*2 写回 A Unlock A Unlock A	Slock A 等待 等待 等待 Y=A Xlock A A=Y*Y 写回 A Unlock A Unlock A

5) 调度如下

T1	T2	T3
Slock A Y=A Xlock A 等待	 Slock A Y=A Xlock A	

	等待	Slock A Y=A Xlock A 等待

10. 今有三个事务的一个调度 r3（B）r1（A）w3（B）r2（B）r2（A）w2（B）r1（B）w1（A），该调度是冲突可串行化的调度吗？为什么？

答：是可串行化的调度，可以变为 r3（B）w3（B）r2（B）r2（A）w2（B）r1（A）r1（B）w1（A），这是串行且基于不冲突操作。

12. 举例说明对并发事务的一个调度是可串行化的，而这些并发事务不一定遵守两段锁原则。

答：

假设有三个事务 T1、T2 和 T3，它们对数据项 X 和 Y 进行读写操作：

T1：W1(Y) W1(X)

T2：W2(Y) W2(X)

T3：W3(X)

现在有一个调度 S 如下：

S = W1(Y) W2(Y) W2(X) W1(X) W3(X)

这个调度不满足冲突可串行化的条件，因为它交换了 T1 和 T2 对 X 的写操作顺序。也就是说，这个调度不能通过交换不冲突操作得到一个串行调度。

但这个调度是可串行化的，因为它满足目标可串行化的条件。例如，这个调度与下面的串行调度等价：

S1 = W1(Y) W1(X) W2(Y) W2(X) W3(X)

因为它们对 X 和 Y 的最终值都相同（分别为 T3 和 T2 的值）。

可以看出，这个调度 S 不遵守两段锁原则，因为 T1 和 T2 在释放锁之后又申请了新的锁，没有将获得锁和释放锁分为两个阶段。

13. 考虑如下的调度，说明这些调度集合之间的包含关系。

1） 正确的调度

2） 可串行化的调度

3） 遵循两阶段封锁（2PL）的调度

4） 串行调度

答：遵循两阶段封锁（2PL）的调度 $\subset$ 正确的调度=可串行化的调度

串行调度 $\subset$ 正确的调度

14. 考虑 T1 和 T2 两个事务。

T1：R（A）；R（B）；B=A+B；W（B）

T2：R（B）；R（A）；A=A+B；W（A）

1） 改写 T1 和 T2，增加加锁操作和解锁操作，遵循两阶段封锁协议。

2） 说明 T1 和 T2 的执行是否会引起死锁，给出 T1 和 T2 的一个调度说明之。

答：

T1	T2
Slock A R（A） Xlock B R（B） B=A+B W（B） Unlock A Unlock B	Slock B R（B） Xlock A R（A） A=A+B W（A） Unlock B Unlock A

可能会产生死锁，如下：

T1	T2
Slock A R（A） Xlock B	 Slock B R（B） Xlock A

15. 为什么要引进意向锁？意向锁的含义是什么？

答：引进意向锁是为什么提高封锁子系统的效率。在多粒度封锁方法中，一个数据对象可能以两种方式加锁——显式封锁和隐式封锁。因此系统在对某一数据对象加锁时，不仅要检查该数据对象上有无(显式和隐式)封锁与之冲突，还要检查其所有上级结点和所有下级结点，看申请的封锁是否与这些结点上的(显式和隐式)封锁冲突。显然，这样的检查方法效率很低。为此引进了意向锁。

意向锁的含义是：对任一结点加锁时，必须先对它的上层结点加意向锁。引进意向锁后，系统对某一数据对象加锁时不必逐个检查与下一级结点的封锁冲突。

16. 试述常用的意向锁：IS 锁、IX 锁、SIX 锁，给出这些锁的相容矩阵。

答：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。例如，要对某个元组加 S 锁，则要首先对关系和数据库加 IS 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。例如，要对某个元组加 X 锁，则要首先对关系和数据库加 IX 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。

相容矩阵：

T1\T2	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