|  |  |
| --- | --- |
| 第12章作业 | |
| 姓名 | 潘腾凯 |
| 学号 | 37220232203786 |
| 班级 | 01班 |

**1. 在数据库中为什么要采用并发控制？并发控制技术能保证事务的哪些特性？**

答：

（1）采用并发控制的原因：

提高系统效率：多个用户或事务同时对数据库进行操作，能充分利用数据库系统的资源，提升系统的吞吐量和响应速度，比如在一个电商平台中，多个用户同时查询商品、下单等操作，若串行执行效率极低。

满足多用户需求：数据库通常是多用户共享的，要支持多个用户同时操作，并发控制可保障多用户环境下数据库操作的正常进行。

（2）保证事务的特性：主要保证事务的原子性、一致性、隔离性和持久性，即 ACID 特性。通过并发控制，可避免多个事务相互干扰，确保每个事务好像在独立执行，从而保障事务执行前后数据库状态的一致性等 。

**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？**

（1）类型主要有：

丢失修改：两个或多个事务同时对同一数据进行修改，一个事务的修改结果会覆盖另一个事务的修改结果，导致先做的修改丢失。

不可重复读：一个事务内多次读取同一数据，在读取过程中，其他事务对该数据进行了修改并提交，导致该事务前后读取结果不一致。

脏读：一个事务读取了另一个事务尚未提交的修改数据，之后若那个事务回滚，那么该事务读取的数据就是无效的 “脏” 数据。

幻读：也称幻影现象，是指事务T1读取数据后，事务T2执行插入或删除操作，使T1无法再现前一次读取的结果。

（2）避免方法：采用并发控制技术，常用的是封锁机制。比如X 锁和S 锁，通过合理给数据对象加锁，控制不同事务对数据的操作权限，避免上述不一致情况；也可采用时间戳排序、乐观并发控制等其他并发控制策略 。

**3. 事务的隔离级别都有哪些，事务隔离级别与数据一致性的关系是什么？**

（1）事务隔离级别：

读未提交：事务能读取到其他事务未提交的修改数据，隔离级别最低，会产生读 “脏” 数据、不可重复读、丢失修改等问题。

读已提交：事务只能读取到其他事务已提交的修改数据，可避免读 “脏” 数据，但仍可能出现不可重复读、丢失修改。

可重复读：保证一个事务内多次读取同一数据时，其值保持一致，可避免读 “脏” 数据和不可重复读，但在一些数据库中可能存在幻读问题。

串行化：最高隔离级别，强制事务串行执行，可避免所有并发不一致问题，但会极大降低系统并发性能，因事务要排队依次执行。

（2）与数据一致性的关系：隔离级别越高，对并发操作的限制越严格，数据一致性保障越好，但系统并发性能越低；隔离级别越低，系统并发性能越高，但数据不一致的风险越大。实际应用中需在数据一致性需求和系统性能之间做权衡，选择合适的隔离级别，比如一般业务系统常采用读已提交或可重复读级别 。

**4.什么是封锁？基本的封锁类型有几种？试述它们的含义。**

答：（1）封锁：是数据库并发控制的一种机制，通过对数据对象加锁，限制事务对数据的操作，实现并发控制。

（2）基本封锁类型：

共享锁（S 锁）：事务对数据加 S 锁后，其他事务可加 S 锁共享读取，不能加排他锁修改 。

排他锁（X 锁）：事务对数据加 X 锁后，其他事务既不能加 S 锁读，也不能加 X 锁修改 。

**5.如何用封锁机制保证数据的一致性？**

答：通过合理运用封锁机制，遵循封锁协议（如三级封锁协议 ），对数据操作加锁，限制并发操作干扰，保证事务执行时数据的一致性，避免丢失修改、读 “脏” 数据、不可重复读等问题。

**6.什么是活锁？试述活锁的产生原因和解决方法。**

答：（1）活锁：事务因调度策略等，长期得不到执行机会，处于等待。

（2）产生原因：调度中优先选择某些事务，导致其他事务长期等待。

解决方法：采用 “先来先服务” 等公平调度策略，按请求顺序分配资源执行事务，类似队列的数据结构。

**7.什么是死锁？请给出预防死锁的若干方法。**

答：（1）死锁：多个事务循环等待对方占有的资源，都无法继续执行。

（2）预防方法：

一次封锁法，事务一次性申请所需全部锁。

顺序封锁法，给数据对象规定固定封锁顺序，事务按序申请锁 。

**8.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？**

检测方法：超时法，即事务等待超设定时间判定死锁 ；事务等待图法，即构建图，若有环则加死锁 。

解除方法：选择一个或几个事务撤销，释放资源，使其他事务继续执行。

**9.什么样的并发调度是正确的调度？**

并发调度结果与某一串行调度结果相同，则该并发调度是正确的调度，即可串行化调度。

**10.设T1、T2、T3是如下的三个事务（A的初值为0）。**

**T1：A := A + 2；**

**T2：A := A \* 2；**

**T3：A := A \*\* 2；（即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段锁协议，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段锁协议，请给出一个产生死锁的调度。**

① 3 个事务串行执行有 6 种顺序（T1T2T3、T1T3T2、T2T1T3、T2T3T1、T3T1T2、T3T2T1 ），计算每种顺序结果：

T1T2T3：A先执行T1变为 2，T2变为 4，T3变为 16 。

T1T3T2：T1后A = 2，T3后A = 4，T2后A = 8 。

T2T1T3：T2后A = 0（原0乘2），T1后A = 2，T3后A = 4 。

T2T3T1：T2后A = 0，T3后A = 0，T1后A = 2 。

T3T1T2：T3后A = 0（0 平方），T1后A = 2，T2后A = 4 。

T3T2T1：T3后A = 0，T2后A = 0，T1后A = 2 。正确结果对应这 6 种串行结果。

② 可串行化调度如 T1 执行完（A = 2），再 T2（A = 4 ），再 T3（A = 16 ），结果 16 。

③ 非串行化调度示例：T1 读 A = 0 ，T2 读 A = 0 ，T1 执行 A = 2 提交；T2 执行 A = 0 \* 2 = 0 提交；T3 读 A = 0 执行 A = 0 。结果与串行不同（如串行 T2T1T3 结果是 4，此调度结果 0）。

④ 两段锁协议下，如 T1 加锁（先 S 锁读，再 X锁改 ）完成，释放锁；T2 加锁操作，再 T3 ，按 T1T2T3 顺序，不会死锁，结果 16 。

⑤ 两段锁协议下死锁调度：T1 加 X 锁改 A（占资源 ），需等 T2 资源；T2 加 X 锁改 A（占资源 ），需等 T1 资源，相互等待形成死锁，如 T1 先部分操作占锁，T2 也占锁并等 T1 ，T1 等 T2 。

**11.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：要判断该调度是否为冲突可串行化调度，需找出冲突操作，构建优先图。

冲突操作判断：读写、写读、写写操作针对同一数据项时为冲突操作。调度中各事务操作及冲突情况如下（R读、W写，括号内为数据项 ）：

T1与T2：R1(A)与W2(A)（冲突，写读 ）、W1(A)与W2(A)（冲突，写写 ）；R1(B)与W2(B)（冲突，写读 ）等。

T1与T3：W3(B)与R1(B)（冲突，读写 ）、W3(B)与W1(B)（冲突，写写 ）等。

T2与T3：W3(B)与R2(B)（冲突，读写 ）、W3(B)与W2(B)（冲突，写写 ）等。

构建优先图：根据冲突操作中事务执行先后确定边（若Ti的冲突操作先于Tj执行，则画Ti → Tj的边 ）。经分析，优先图存在环（需详细梳理操作顺序推导，此处省略复杂过程 ），所以该调度不是冲突可串行化调度。

1. **试证明若并发事务遵守两段锁协议，则对这些事务的并发调度是可串行化的。**

答：采用反证法 + 两段锁协议特性证明： 假设存在遵守两段锁协议的事务并发调度不可串行化，即存在冲突操作导致优先图有环（T1 → T2 → ...... → T1 ）。

两段锁协议要求事务分两个阶段：① 加锁阶段（只能加锁，不能解锁 ）；② 解锁阶段（只能解锁，不能加锁 ）。

若有环，说明存在事务Ti的加锁操作在Tj加锁操作之后，且Tj加锁在Ti之后，这与两段锁 “先集中加锁，后集中解锁” 的阶段特性矛盾。 故假设不成立，遵守两段锁协议的并发事务调度一定可串行化。

**13.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协议。**

答：

示例：

事务T1：R(A)；W(B)

事务T2：R(B)；W(A)

调度：R1(A) → R2(B) → W1(B) → W2(A)

分析：

可串行化判断：该调度结果与串行调度（如T1全执行完再T2，或T2全执行完再T1 ）结果一致（需验证数据读写结果，此处省略计算 ），所以是可串行化调度。

两段锁协议判断：T1先读A（可加 S 锁 ），再写B（需加 X 锁 ），但读A后未到解锁阶段就写B加锁，不满足 “先加锁阶段、后解锁阶段” 的两段锁要求；同理T2也不满足。 因此，存在可串行化调度对应的事务不遵守两段锁协议，两段锁是可串行化的充分非必要条件 。

**14.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 遵循两段锁的调度。 ④ 串行调度。**

答： 遵循两段锁的调度 真包含于 正确的调度 = 可串行化的调度

串行调度 真包含于 正确的调度

**15.考虑如下的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，并要求遵循两段锁协议。**

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

答：

① 改写事务

两段锁协议要求事务分 “加锁阶段”（只能加锁，不能解锁 ）和 “解锁阶段”（只能解锁，不能加锁 ）。对读写操作加锁，读操作加共享锁（S 锁 ），写操作加排他锁（X 锁 ），全部加锁后统一解锁。

T1改写：LockS(A); R(A); LockS(B); R(B); LockX(B); B = A + B; W(B); Unlock(A); Unlock(B); （先加 S 锁读A、B，因要写B，再加 X 锁写B；加锁阶段完成后，解锁阶段统一解锁 ）

T2改写：LockS(B); R(B); LockS(A); R(A); LockX(A); A = A + B; W(A); Unlock(B); Unlock(A); （先加 S 锁读B、A，因要写A，再加 X 锁写A；加锁阶段完成后，解锁阶段统一解锁 ）

② 是否死锁：可能死锁，取决于加锁顺序。

调度示例 1（死锁场景 ）：T1: LockS(A) → R(A); LockS(B)（等待，因T2已加S锁 ）T2: LockS(B) → R(B); LockS(A)（等待，因T1已加S锁 ） 此时T1等T2释放B的 S 锁，T2等T1释放A的 S 锁，循环等待，产生死锁。

调度示例 2（无死锁场景 ）：T1完整执行：LockS(A) → R(A) → LockS(B) → R(B) → LockX(B) → W(B) → Unlock(A) → Unlock(B); 再执行T2：LockS(B) → R(B) → LockS(A) → R(A) → LockX(A) → W(A) → Unlock(B) → Unlock(A); 此调度按串行顺序执行，无死锁，因T1解锁后T2才申请锁，资源不冲突。

**16 为什么要引进意向锁？意向锁的含义是什么？**

答：引进意向锁原因：在多粒度封锁（表、页、行等不同粒度）中，判断父节点（如页对表 ）锁与子节点（行对页 ）锁的冲突，避免逐行检查子节点锁，提升锁判断效率。

意向锁含义：表明事务对某一数据对象加锁的意向，同时记录父节点（更高粒度 ）锁的意向，用于多粒度封锁的快速冲突判断 。

**17.试述常用的意向锁（IS、IX、SIX），给出这些锁的相容矩阵。**

答：常用意向锁及含义：

IS 锁（意向共享锁）：事务意图在数据对象的某些子对象上加 S 锁，先在父对象上加 IS 锁 。

IX 锁（意向排他锁）：事务意图在数据对象的某些子对象上加 X 锁，先在父对象上加 IX 锁 。

SIX 锁（共享意向排他锁）：事务先在数据对象上加 S 锁，又意图在某些子对象上加 X 锁，在父对象上加 SIX 锁 。

相容矩阵（行表示现有锁，列表示申请锁，“Y” 相容，“N” 不相容 ）：

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

