|  |  |
| --- | --- |
| 第11章作业 | |
| 姓名 | 潘腾凯 |
| 学号 | 37220232203786 |
| 班级 | 01班 |

1. **试述事务的概念及事务的4个特性。数据库恢复技术能保证事务的哪些特性?**

答：（1）事务的概念：是数据库管理系统执行过程中的一个逻辑单元，是由一组操作构成的不可分割的整体。这些操作要么全部执行成功，要么全部失败回滚，确保数据库从一个一致状态转换到另一个一致状态。

（2）四个特性分别是ACID，即原子性、一致性、隔离性、持续性。

原子性：事务是数据库的逻辑工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性：事务执行的结果必须是使数据库从一个一致性状态转为另一个一致性状态。

隔离性：一个事务的执行不能被其他事务干扰，即一个事务的内部操作及使用的数据对其他并发事务是隔离的，并发执行的各个事务之间不能互相干扰。

持续性：指一个事务一旦提交，对数据库中数据的改变就应该是永久性的，接下来其他操作或故障不应该对其执行结果有任何影响。

（3）数据库恢复可以保证事务的原子性和持续性。

**2.为什么事务非正常结束时会影响数据库数据的正确性?请举例说明之。**

答：非正常结束破坏了事务的ACID特性，可能导致数据库中数据处于不一致状态，破坏数据的正确性。以下从三个方面结合实例说明原因：

（1）原子性缺失导致数据不完整

原子性要求事务中的操作必须全部成功或全部回滚。若事务执行到一半时中断且未回滚，会残留部分操作结果，导致数据逻辑矛盾。  
举例：银行转账事务（从账户 A 转 100 元到账户 B）包含两步操作：

扣除 A 账户 100 元（A 余额 = A - 100）；

增加 B 账户 100 元（B 余额 = B + 100）。

非正常结束场景：

执行完第一步后，系统突然崩溃，事务未提交且未回滚。

此时数据库中 A 账户已扣款，但 B 账户未到账，导致A+B 的总余额减少 100 元，违反 “转账前后总金额不变” 的业务规则，数据失去一致性。

（2）持久性未保障导致已提交数据丢失

持久性要求已提交事务的修改必须永久保存。若事务提交后，修改尚未写入磁盘时发生硬件故障（如磁盘损坏），可能导致数据丢失，影响正确性。  
举例：电商订单系统中，用户下单并支付成功（事务已提交），系统需执行：

创建订单记录（状态为 “已支付”）；

扣除库存（商品库存 -1）。

非正常结束场景：

事务提交后，订单记录已写入内存但未同步到磁盘，此时磁盘突然损坏。数据库恢复时，因未持久化存储，订单记录和库存扣除操作丢失，导致：用户端显示订单存在，但数据库中无记录；库存未实际扣除，可能引发超卖问题。

（3）其他潜在原因：脏数据与隔离性间接影响

在并发场景下，事务非正常结束可能通过隔离性漏洞间接导致数据错误。例如：

事务 T1 修改数据后未提交（处于中间状态），事务 T2 读取了该未提交数据（脏读）。

若 T1 随后因异常回滚，T2 基于脏数据的计算结果将完全错误。

举例：库存查询与扣减场景：

T1 查询商品库存为 100 件，准备扣减 5 件（未提交）；

T2 此时查询库存（读取 T1 的中间结果 100-5=95 件），并基于 95 件生成销售计划；

T1 因异常回滚，库存恢复为 100 件，而 T2 的销售计划基于错误的 95 件，导致后续库存超卖。

**3.登记日志文件时为什么必须先写日志文件，后写数据库?**

答：这是由事务的原子性和持久性决定的。先写日志的本质目的是确保在数据库修改前，其对应的日志已持久化存储。这样即使在操作过程中发生系统崩溃或断电，也能通过日志恢复数据，避免丢失已提交事务的修改或残留未提交的修改。在持久性方面，确保已提交事务的修改可通过 Redo 日志永久恢复；在原子性方面，确保未提交事务的修改可通过 Undo 日志彻底回滚。此外，通过日志的顺序写入和异步数据刷新，可以在保证数据安全的同时提升读写性能。

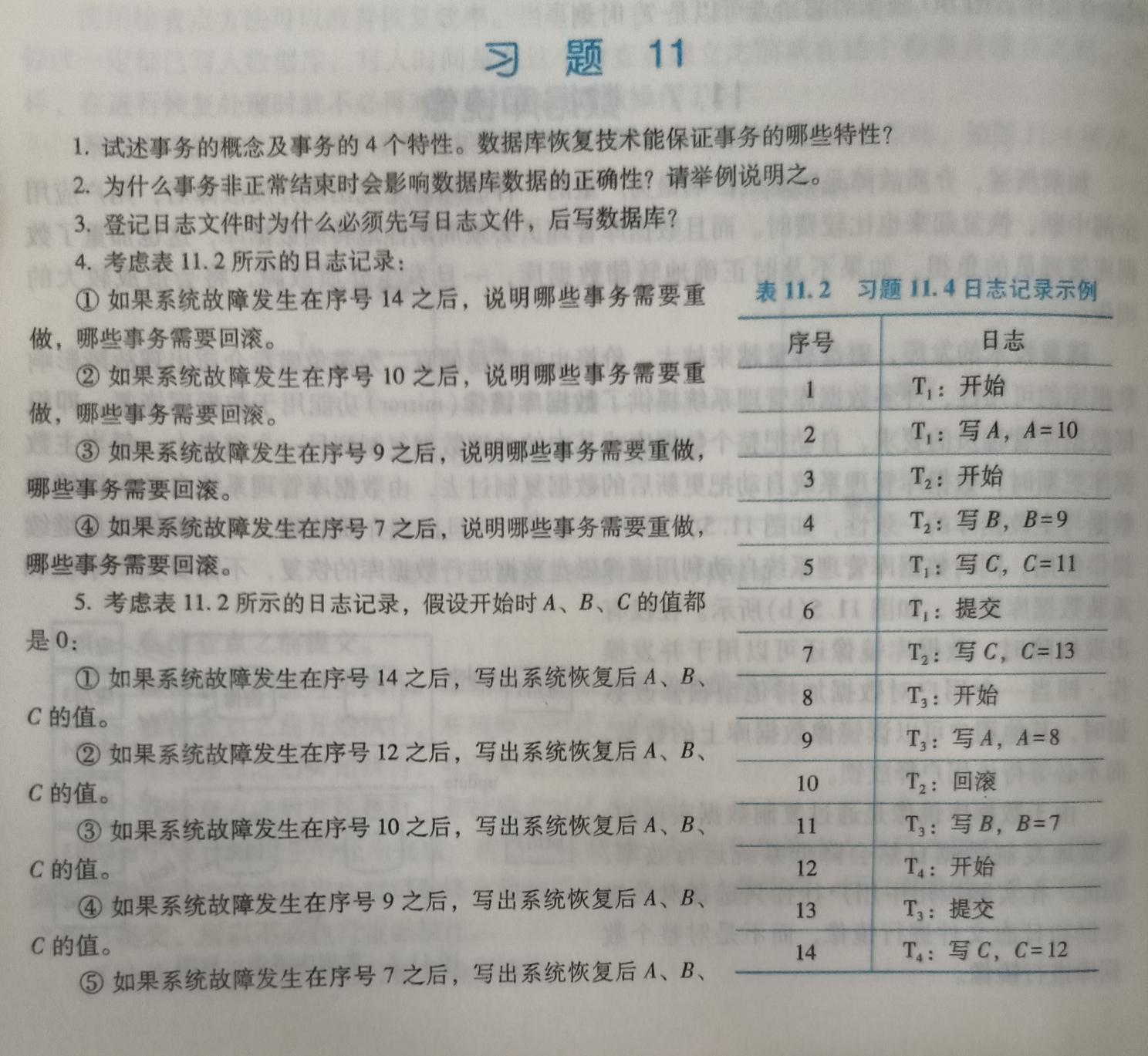
**4.考虑表11.2所示的日志记录:**

**①如果系统故障发生在序号14之后，说明哪些事务需要重做，哪些事务需要回滚。**

**②如果系统故障发生在序号10之后，说明哪些事务需要重做，哪些事务需要回滚。**

**③如果系统故障发生在序号9之后，说明哪些事务需要重做，哪些事务需要回滚。**

**④如果系统故障发生在序号7之后，说明哪些事务需要重做，哪些事务需要回滚。**

****

答：①重做：T1、T3；回滚T2、T4

因为T1和T3事务已经提交，故重做；T2回滚，T4执行中，没有提交，所以要回滚。

②重做：T1；回滚：T2、T3

故障前事务T1已经提交，故重做；T2回滚，T3执行中，没有提交，所以要回滚。T4还未开始。

③重做：T1；回滚：T2、T3

T1已提交；T2、T3执行中；T4未开始。

④重做：T1；回滚：T2

T1已经提交，重做；T2执行中，回滚；T3、T4未开始。

**5.考虑表 11.2所示的日志记录，假设开始时 A、B、C的值都是0:**

**①如果系统故障发生在序号14之后，写出系统恢复后A、B.C的值。**

**②如果系统故障发生在序号12之后，写出系统恢复后 A、B、C的值。**

**③如果系统故障发生在序号10之后，写出系统恢复后A、B、C的值。**

**④如果系统故障发生在序号9之后，写出系统恢复后A、B、C的值。**

**⑤如果系统故障发生在序号7之后，写出系统恢复后A、B、C的值。**

**⑥如果系统故障发生在序号5之后，写出系统恢复后 A、B、C的值。**

答：①A=8，B=7，C=11

②A=10，B=0，C=11

③A=10，B=0，C=11

④A=10，B=0，C=11

⑤A=10，B=0，C=11

⑥A=0，B=0，C=0

**6.针对不同的故障类型(事务故障、系统故障、介质故障)，试给出恢复的策略和方法。**

答：（1）事务故障的恢复：

从日志文件末尾开始反向扫描，查找该事务的更新操。

按日志记录的逆序（从后往前）执行Undo 操作：对于UPDATE操作，用日志中记录的旧值（Before Image）恢复数据；对于INSERT操作，执行删除操作；对于DELETE操作，执行插入旧值操作。

直至遇到事务的开始标记，完成故障的恢复。

（2）系统故障的恢复：

正向扫描日志文件，找出在故障发生前已经提交的事务，将其事务标识记入重做队列REDO

-LIST；同时找出故障发生时尚未完成的人物，将其事务标识记入撤销队列UNDO-LIST。

对撤销队列中的各个事务进行UNDO处理。

对重做队列中的各个事务进行REDO处理。

（3）介质故障的恢复：

装入最新的数据库后备副本，使数据库恢复到最近一次转储时的一致性状态。

装入转储结束时刻的日志文件副本。

启动系统恢复命令，由DBMS完成恢复功能，重做已完成的事务。

**7.什么是检查点记录?检查点记录包括哪些内容?**

答：（1）检查点记录（Checkpoint Record）是数据库恢复技术中的关键机制，用于定期将内存中已修改的数据页（脏页）写入磁盘，并在日志文件中记录检查点时刻的数据库状态，通常记为Tc。

（2）检查点记录通常包含以下信息，用于标识检查点时刻的数据库状态和恢复所需的关键元数据：

1） 检查点开始的时间戳

记录检查点操作的起始时间，用于后续恢复时定位日志范围。

2）当前活跃事务列表

活跃事务：在检查点时刻尚未提交或回滚的事务（即处于BEGIN TRANSACTION但未执行COMMIT/ROLLBACK的事务）。

列表内容包括：

事务 ID（如T1, T2）；

每个事务的最近日志记录地址（用于恢复时快速定位该事务的日志起始位置）。

3） 脏页列表

脏页：在内存中被修改但尚未写入磁盘的数据页（对应磁盘上的物理块编号）。

列表记录这些脏页的物理地址（如数据文件路径 + 页号），确保在系统故障时能通过日志重做或回滚这些页的修改。

4.）日志文件的当前位置

记录检查点完成时日志文件的最后一条记录的地址（如日志序列号 LSN），用于恢复时确定从何处开始扫描日志。

5）检查点结束标识

标记检查点操作完成，确保日志的完整性（如记录END CHECKPOINT标记）。

**8.具有检查点的恢复技术有什么优点？试举一个具体例子加以说明。**

答：（1）其核心优点如下：

1）大幅减少恢复时的日志扫描范围

检查点优化后只需从最近一个检查点开始扫描日志，跳过检查点之前已确认稳定的日志记录，减少 I/O 操作和处理时间。

2） 降低内存数据丢失风险

检查点会强制将内存中已修改的数据页（脏页）写入磁盘，确保内存数据与磁盘数据定期同步。即使系统故障导致内存数据丢失，也能通过最近的检查点恢复大部分数据，减少需要重做或回滚的操作量。

3） 快速定位活跃事务和脏页

检查点记录中包含活跃事务列表和脏页列表，恢复时可直接根据这些信息对未提交的活跃事务执行回滚，或对已提交但脏页未完全写入磁盘的事务执行重做。避免了传统恢复中遍历全量日志来识别事务状态的复杂过程。

4） 支持在线恢复和增量备份

检查点可作为恢复的基准点，配合增量备份（仅备份检查点之后的变化），进一步提升备份与恢复效率。

（2）举例如下：

假设数据库运行过程中按以下时间线执行操作，并在T5时刻触发检查点CP1：

T1: 事务A开始（修改账户表数据页P1）

T2: 事务B开始（修改账户表数据页P2）

T3: 检查点CP1触发，刷新脏页P1、P2到磁盘，记录活跃事务[A, B]

T4: 事务A提交

T5: 事务C开始（修改数据页P3）

T6: 系统故障（内存数据丢失，磁盘数据停留在CP1后的状态）

传统恢复方式（无检查点）

需扫描全部日志：从第一条日志开始处理，依次判断每个事务的状态。

处理步骤：

发现事务 A 已提交，重做对 P1 的修改（但 P1 在 CP1 时已写入磁盘，重做是冗余的）；

发现事务 B 未提交，回滚对 P2 的修改；

发现事务 C 未提交，回滚对 P3 的修改。

缺点：浪费时间重做已持久化的数据（P1、P2），且扫描范围大。

带检查点的恢复方式

仅扫描 CP1 之后的日志：从检查点 CP1 开始处理，跳过 CP1 之前已确认写入磁盘的操作。

处理步骤：

分析阶段：读取 CP1 记录，获取活跃事务列表 [A, B] 和脏页列表 [P1, P2]。

重做阶段：

事务 A 在 CP1 后提交，需重做对 P1 的修改（但 P1 已在 CP1 时写入磁盘，实际无需操作）；

事务 C 在 CP1 后开始并修改 P3（脏页未写入磁盘），需重做 P3 的修改（若事务 C 已提交）或回滚（若未提交）。

回滚阶段：事务 B 未提交，回滚对 P2 的修改（P2 在 CP1 时已写入磁盘，回滚需撤销其修改）。

优点：跳过 CP1 之前的日志扫描，直接从 CP1 开始，减少约 50% 的日志处理量；明确区分已持久化和未持久化的脏页，避免冗余操作。

**9.试述使用检查点方法进行恢复的步骤。**

答：在重新开始文件中，找到最后一个检查点记录在日志文件中的地址，由该地址在日志文件中找到最后一个检查点记录。

由该检查点记录得到检查点建立时刻所有正在执行的事务清单ACTIVE-LIST，并建立两个十五队列：UNDO-LIST和REDO-LIST。把ACTIVE-LIST暂时放入UNDO-LIST队列，REDO-LIST队列暂为空。

从检查点开始正向扫描日志文件。如有新开始的事务Ti，把Ti暂时放入UNDO-LIST队列。如有已提交的事务Tj，把Tj从UNDO-LIST队列迁移到REDO-LIST队列，直到日志文件结束。

对UNDO-LIST中的每个事务执行UNDO操作，对REDO-LIST中的每个事务执行REDO操作。

**10.什么是数据库镜象?它有什么用途?**

答：数据库镜像是一种数据库高可用性和灾备技术，通过实时复制主数据库（主体数据库）的数据到一个或多个辅助数据库（镜像数据库），使主库与镜像库保持实时同步。

用途有：

用于数据库恢复。当出现介质故障时，镜像磁盘可继续提供使用，同时DBMS自动利用镜像磁盘数据进行数据库的恢复，不必关闭系统和重装数据库副本。

提高数据库的可用性。在没有出现故障时，当一个用户对数据加排他型锁进行修改时，其他用户可以读镜像数据库上的数据，而不必等待该用户释放锁。

