作业 7

SA20225085 朱志儒

1. 目前许多 DBMS 例如 MySQL 都默认不支持嵌套事务（即在一个事务内部又开始另一个事务），请分析一下：如果 DBMS 支持嵌套事务，将面临哪些问题（至少写出 2 点以上并且要给出自己的分析）？

   解：如果 DBMS 支持嵌套事务，将面临如下问题：

   问题 1：如果回滚最外部事务，那么将回滚所有内部事务，无论内部事务是否已经提交过，这将大大降低数据库的执行效率。

   问题 2：如果内部事务出现回滚，那么最外部事务将被提前终止并回滚，而这也将极大地降低数据库的执行效率。

2. 下面是一个数据库系统开始运行后的日志记录，该数据库系统支持检查点。

   1) <T1, Begin Transaction>

   2) <T1, A, 10, 40>

   3) <T2, Begin Transaction>

   4) <T1, B, 20, 60>

   5) <T1, A, 40, 75>

   6) <T2, C, 30, 50>

   7) <T2, D, 40, 80>

   8) <T1, Commit Transaction>

   9) <T3, Begin Transaction>

   10) <T3, E, 50, 90>

   ---------- ①

   11) <T2, D, 80, 65>

   12) <T2, C, 50, 75>

   13) <T2, Commit Transaction>

   ---------- ②

   14) <T3, Commit Transaction>

   15) **<CHECKPOINT>**

16) <T4, Begin Transaction>

17) <T4, F, 60, 120>

18) <T4, G, 70, 140>

---------- ③

19) <T4, F, 120, 240>

20) <T4, Commit Transaction>

设日志修改记录的格式为<Tid, Variable, Old value, New value>，请给出对于题中所示①、②、③三种故障情形下，数据库系统恢复的过程以及数据元素 A, B, C, D, E, F 和 G 在执行了恢复过程后的值。

解：故障①：

从最近的<CHECKPOINT>开始，正向扫描日志，将<commit>的事务放入 Redo 列表中，将没有结束的事务放入 Undo 列表，则 Redo 列表：{T1}，Undo 列表：{T2，T3}。

反向扫描日志，对 Undo 列表中的事务执行 Undo：

T3: E = 50

T2: D = 40

T2: C = 30

正向扫描日志，对 Redo 列表中的事务执行 Redo：

T1: A = 40

T1: B = 60

T1: A = 75

最后，在日志中写入<abort，T2>，<abort，T3>。

故，数据元素 A = 75, B = 60, C = 30, D = 40, E = 50。

故障②：

从最近的<CHECKPOINT>开始，正向扫描日志，将<commit>的事务放入 Redo 列表中，将没有结束的事务放入 Undo 列表，则 Redo 列表：{T1，T2}，Undo 列表：{T3}。

反向扫描日志，对 Undo 列表中的事务执行 Undo：

T3: E = 50

正向扫描日志，对 Redo 列表中的事务执行 Redo：

T1: A = 40

T1: B = 60

T1: A = 75

T2: C = 50

T2: D = 80

T2: D = 65

T2: C = 75

最后，在日志中写入<abort，T3>。

故，数据元素 A = 75, B = 60, C = 75, D = 65, E = 50。

故障③：

从最近的<CHECKPOINT>开始，正向扫描日志，将<commit>的事务放入 Redo 列表中，将没有结束的事务放入 Undo 列表，则 Redo 列表：{ }，Undo 列表：{T4}。

反向扫描日志，对 Undo 列表中的事务执行 Undo：

T4: G = 70

T4: F = 60

正向扫描日志，对 Redo 列表中的事务执行 Redo：

Redo 列表为空

最后，在日志中写入<abort，T4>。

故，数据元素 A = 75, B = 60, C = 75, D = 65, E = 90, F = 60, G = 70。

3. 采用了两阶段锁协议的事务是否一定不会出现脏读问题？如果不会，请解释理由；如果会，请给出一个例子。

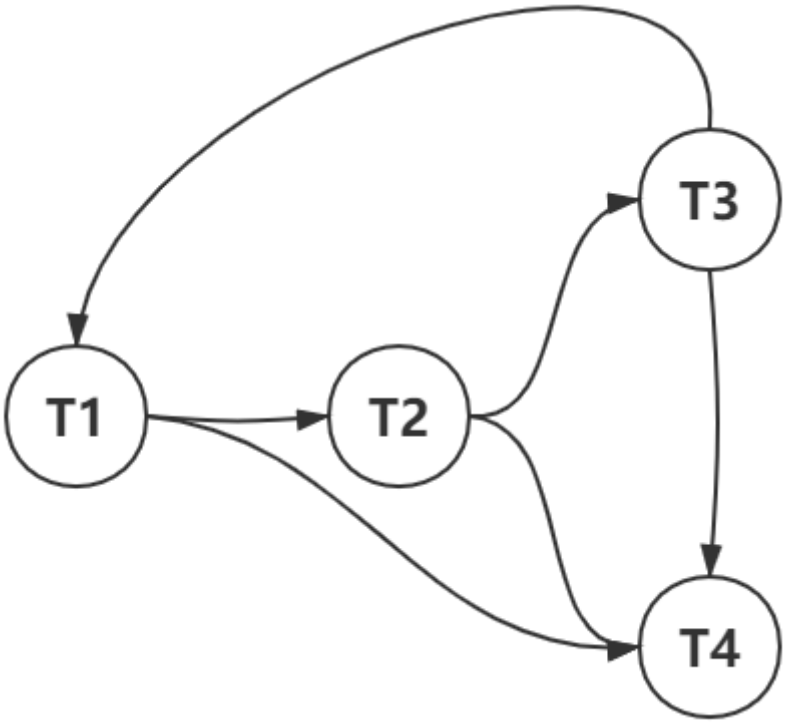
解：采用了两阶段锁协议的事务一定不会出现脏读问题，因为事务 A 在访问数据 R 前会对数据 R 加 S 锁，其他事务 B 如果要修改数据 R 前需要加 X 锁，而 S 锁与 X 锁不相容，事务 B 加 X 锁会被拒绝，事务 B 进入等待，如此事务 A 不会脏读。

如果在事务 A 对数据 R 加 S 锁前，事务 B 已经对数据 R 加 X 锁，则事务 A 加 S 锁会被拒绝，等到事务 B 修改完 commit 后释放 X 锁，事务 A 就可对数据 R 加 S 锁，进而读取已 commit 的数据 R，如此事务 A 不会脏读。

4. 判断下面的并发调度是否冲突可串？如果是，请给出冲突等价的串行调度事务顺序；如果不是，请解释理由。

w3(D); r1(A); w2(A); r4(A); r1(C); w2(B); r3(B); r3(A); w1(D); w3(B); r4(B); r4(C); w4(C); w4(B)

解：上面的并发调度对应的优先图如下：



显然，优先图中存在圈：T1→T2→T3→T1，故该并发调度不是冲突可串的。