**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。

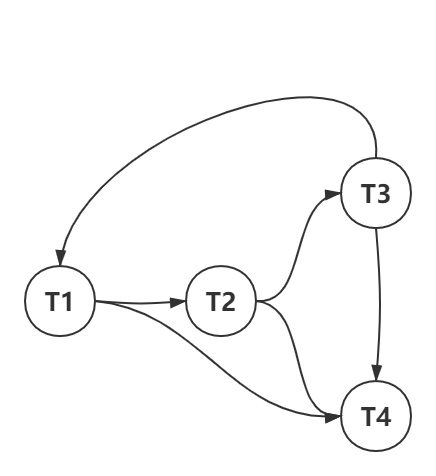
**解：**采用了两阶段锁协议的事务一定不会出现脏读问题，因为事务A在访问数据R前会对数据R加S锁，其他事务B如果要修改数据R前需要加X锁，而S锁与X锁不相容，事务B加X锁会被拒绝，事务B进入等待，如此事务A不会脏读。

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。