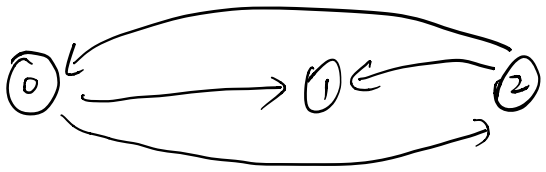
数据库第六次作业

1. 画出优先图如下：



由于0和2中存在环路，所以这不是一个冲突可串行化调度

2. (1) 三个事务都遵守两段锁协议，给出一个不产生死锁的可串行化调度

T1	T2	T3
	Lock-X(A) READ(A) A = A×3 WRITE(A) Unlock(A)	
		Lock-X(A) READ(A) $A = A^2$ WRITE(A) Unlock(A)
Lock-X(A) READ(A) A = A+4 WRITE(A) Unlock(A)		

最终 A=40

(2)

T1	T2	T3
SlockA Y=A=2		
	SlockA Y=A=2	
XlockA 等待		
	XlockA 等待	
		SlockA Y=A=2 XlockA 等待

3.

T1	T2
	READ(B) B=B-50 WRITE(B)
READ(B)	
	READ(A) A=A+50 WRITE(A)
READ(A) Display(A+B)	
	Display(A+B)

4. 恢复算法过程：自底向上扫描，遇到 <checkpoint {T4, T5}>，
将活跃事务集 L = {T4, T5} 加入 Undo-List
随向从检查点往下扫描，遇到 <T5 commit> 故将 T5 从 Undo-list
转移入 Redo-List
又遇到 <T6 start>，故将 T6 加入 Undo-List
继续扫描，遇到 <T4 abort>，故将 T4 移出 Undo-List

Undo-List : {T6}

Redo-List : {T5}

数据库系统恢复后，T6 执行 Undo，T5 执行 Redo

最终结果：A = 700　　B = 1000　　C = 100

5.

a.

1	<T0, Start>
2	<T2, Start>
3	<T2, C, 35, 70>
4	<Start checkpoint T0, T2>
5	<end checkpoint>
6	<T2, commit>
7	<T0, A, 50, 70>
8	<T1, Start>
9	<T1, B, 30, 20>
10	<T1, commit>
11	<T3, Start>
12	<T3, D, 15, 30>

b. Undo-List : T_0, T_3 因为它们在故障前 start，并未 commit

Redo-List : T_1, T_2 因为它们在故障前 start，在 checkpoint 之后 commit

c. T_0, T_3 执行 Undo 需要添加 $<T_0, abort>$、$<T_3, abort>$ 表示它们已中止并回滚。

此外还会周期性地写入检查点记录。