CS 425 Homework 5

Mukai Yu

Wednesday, April 13, 2022

1. (a) T1: Commited.
   Because initially all accounts have balance 0.

   T2: Aborted.
   Because A doesn't have enough money for withdrawal of 70.

   T3: Aborted.
   Because C doesn't have enough money for withdrawal of 30.

   T4: Commited.
   Because A, B have enough money for withdrawal of 20 and 60, C can accept any amount of deposit.

   T5: Commited.
   Because it's just read operation without any other transactions being processed at the same time.

   (b) T1: Nothing.
   No $BALANCE$ called.

   T2: Balance of B is 30.

   T3: Balance of A is 10

   T4: Nothing.
   No $BALANCE$ called.

   T5: Balance of A is 30.
   Balance of B is 20.
   Balance of C is 40.

2. (a) T1.2 is **read B**, T2.4 is **write B**
T2.4 is **write B**, T1.5 is **write B**
T1.3 is **write A**, T2.3 is **read A**

(b) No. Because T1.3 **write A** and T2.3 **read A** is executed in the order of T1 → T2, But T2.4 **write B** and T1.5 **write B** is executed in the order of T2 → T1, which is conflicting the previous execution order.

(c) T1 → T2

T1	T2
read A	
read B	
write A	
	read D
	write C
read D	
write B	
T1 releases all locks	
	read A
	write B
	write E

(d) T2 → T1

T1	T2
	read D
	write C
	read A
read A	
	write B
	write E
	T2 releases all locks
read B	
write A	
read D	
write B	

(e) Deadlock:

T1	T2	Lock A	Lock B
	read D		
	write C		
	read A	read by T2	
read A		read by T1, T2	
read B		read by T1, T2	read by T1
	write B	read by T1, T2	read by T1; write waited by T2
write A		read by T1, T2; write waited by T1	read by T1;write waited by T2

T1 holds read lock of A and B, waits write lock of A.
T2 holds read lock of A, waits write lock of B.

(f) Serial equivalent to T1 → T2:

T1	T2
read A	
read B	
write A	
	read D
	write C
	read A
read D	
write B	
T1 releases all locks	
	write B
	write E

This is serial equivalent to T1 → T2, because **write A** of T1 → **read A** of T2, and **read B & write B** of T1 → **write B** of T2.
For strict two-phase locking, T1's release of any lock must happen at the commit point of T1, which is after **write B** of T1.
Thus under strict two-phase locking, it is impossible for **read A** of T2 to happen before **write B** of T1.

(g) Timestamped ordering T1 → T2 (omitting C, D, E's state):

T1	T2	A.commitedTS	A.RTS	A.TW	B.commitedTS	B.RTS	B.TW
read A			1				
read B			1			1	
write A			1	1		1	
	read D		1	1		1	
	write C		1	1		1	
read D			1	1		1	
write B			1	1		1	1
	read A	1	1, 2		1	1	
	write B	1	1, 2		1	1	2
	write E	1	1, 2		1	1	2
		1	1, 2		2	1	

(h) T1 gets aborted (omitting C, D, E's state):

T1	T2	A.commitedTS	A.RTS	A.TW	B.commitedTS	B.RTS	B.TW
read A			1				
	read D		1				
	write C		1				
	read A		1, 2				
	write B		1, 2				2
	write E		1, 2				2
read B			1, 2		2		

read B of T1 gets T1 aborted since B has already been committed by T2.

3. (a) Commit time:

   Server 1: t = 70 ms
   Server 2: t = 80 ms
   Server 3: t = 75 ms

   (b) Earliest time t = 70 ms