Yao Zhou
001586311

1. Given the following 2 different transactions. List all potential schedules for T1 and T2 and determine which schedules are conflict serializable and which are not. (10 POINTS)

Transaction 1	Transaction 2
READ(X)	READ(X)
X = X -N;	X = X +M
WRITE(X)	WRITE(X)
READ(Y)	
Y = Y + N	
WRITE(Y)	

Schedule 1: (**conflict serializable schedule**)

Time	Transaction1	Transaction2
T1	Begin transaction	
T2	READ(X)	
T3	X = X – N;	
T4	WRITE(X)	
T5	READ(Y)	
T6	Y = Y + N;	
T7	WRITE(Y)	
T8	Commit	
T9		Begin transaction
T10		READ(X)
T11		X = X + M;
T12		WRITE(X)
T13		Commit

Schedule 2: (**conflict serializable schedule**)

Time	Transaction1	Transaction2
T1		Begin transaction
T2		READ(X)
T3		X = X + M;
T4		WRITE(X)
T5		Commit
T6	Begin transaction	
T7	READ(X)	
T8	X = X – N;	
T9	WRITE(X)	
T10	READ(Y)	
T11	Y = Y + N;	
T12	WRITE(Y)	
T13	Commit	

Schedule 3: (**conflict serializable schedule**)

Time	Transaction1	Transaction2
T1	Begin transaction	
T2	READ(X)	
T3	X = X – N;	
T4	WRITE(X)	
T5		Begin transaction
T6		READ(X)
T7		X = X + M;
T8		WRITE(X)
T9		Commit
T10	READ(Y)	
T11	Y = Y + N;	
T12	WRITE(Y)	
T13	Commit	

Schedule 4: (**not conflict serializable schedule**)

Time	Transaction1	Transaction2
T1	Begin transaction	
T2	READ(X)	
T3	X = X – N;	
T4		Begin transaction
T5		READ(X)
T6		X = X + M;
T7		WRITE(X)
T8		Commit
T9	WRITE(X)	
T10	READ(Y)	
T11	Y = Y + N;	
T12	WRITE(Y)	
T13	Commit	

Schedule 5: (**not conflict serializable schedule**)

Time	Transaction1	Transaction2
T1	Begin transaction	
T2	READ(X)	
T3	X = X – N;	
T4		Begin transaction
T5		READ(X)
T6	WRITE(X)	
T7		X = X + M;
T8		WRITE(X)
T9		Commit
T10	READ(Y)	
T11	Y = Y + N;	
T12	WRITE(Y)	
T13	Commit	

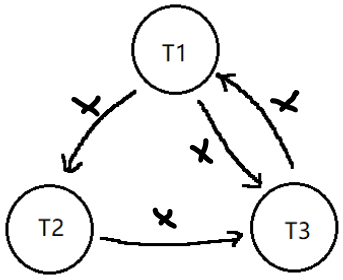
2. Which of the following schedules is conflict serializable? For each serializable schedule determine the equivalent serial schedule.(10 POINTS)

Schedule 1	Schedule 2	Schedule 3
T1 Read(X)	T1 Read(X)	T3 Read(X)
T3 Read(X)	T3 Read(X)	T2 Read(X)
T1 Write(X)	T3 Write(X)	T3 Write(X)
T2 Read(X)	T1 Write(X)	T1 Read(X)
T3 Write(X)	T2 Read(X)	T1 Write(X)

3. Draw the precedence graph for the 3 schedules in problem 2. (20 POINTS)

Schedule1:

Time	T1	T2	T3
1	READ(X)		
2			READ(X)
3	WRITE(X)		
4		READ(X)	
5			WRITE(X)

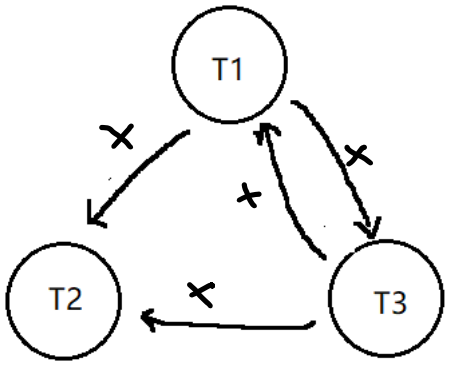


Because the precedence graph has a circle among T1,T2 and T3, the schedule is **not conflict**

serializable schedule.

Schedule2:

Time	T1	T2	T3
1	READ(X)		
2			READ(X)
3			WRITE(X)
4	WRITE(X)		
5		READ(X)	



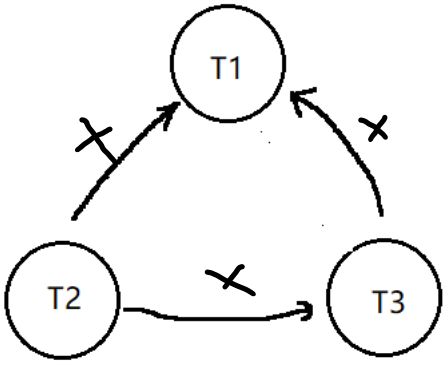
Because the precedence graph has a circle between T1 and T3, the schedule is **not conflict serializable schedule**.

Schedule3:

Time	T1	T2	T3
1		READ(X)	
2			READ(X)
3			WRITE(X)
4	READ(X)		
5	WRITE(X)		

Equivalent serial schedule with Schedule 3

Time	T1	T2	T3
1			READ(X)
2		READ(X)	
3			WRITE(X)
4	READ(X)		
5	WRITE(X)		



Because the precedence graph does not have a circle, the schedule is **conflict serializable schedule**.

4. Provide a schedule that exhibits the deadlock problem. Describe the issue. (10 POINTS)

Time	Transaction1	Transaction2
T1	Begin transaction	
T2	Write_lock(X)	Begin transaction
T3	Read(X)	Write_lock(Y)
T4	X = X - 10	Read(Y)
T5	Write(X)	Y = Y + 100
T6	Write_lock(Y)	Write(Y)
T7	WAIT	Write_lock(X)
T8	WAIT	WAIT
T9	WAIT	WAIT
T10	…	WAIT
T11	…	…
T12	…	…

Deadlock is an impasse that may result when two (or more) transactions are each waiting for locks held by the other to be released.[Reference from T6_L4_DeadLock.ppt] As we can see the above schedule example, T1 was writing a lock on X but not released and waiting for Y, However, T2 was also waiting for locking X and didn't release the lock on Y. In this situation, T1 and T2 are stuck in other's lock and can't move forward.

5. Apply the timestamping algorithm to the following schedule. State if it can be performed as is or what transactions will need to be restarted given the basic timestamping ordering algorithm. (20 POINTS)

TIME	Transaction A	Transaction B	Transaction C
1		READ(Z)	

2		READ(Y)	
3		WRITE(Y)	
4			READ(Y)
5			READ(Z)
6	READ(X)		
7	WRITE(X)		
8			WRITE(Y)
9			WRITE(Z)
10		READ(X)	
11	READ(Y)		
12	WRITE(Y)		
13		WRITE(X)	

RT = read_timestamp; WT = write_timestamp; TA = transaction A; TB = transaction B; TC = transaction C

Time	Operation	Timestamp modification
1	B READ(Z)	TS(TA); TS(TB) = 1; TS(TC) ; RT(X); RT(Y); RT(Z) = 1; WT(X); WT(Y); WT(Z)
2	B READ(Y)	TS(TA); TS(TB) = 1; TS(TC) ; RT(X); RT(Y) = 1; RT(Z) = 1; WT(X); WT(Y); WT(Z)
3	B WRITE(Y)	TS(TA); TS(TB) = 1; TS(TC) ; RT(X); RT(Y) = 1; RT(Z) = 1; WT(X); WT(Y) = 1; WT(Z)
4	C READ(Y)	TS(TA); TS(TB) = 1; TS(TC) = 4 ; RT(X); RT(Y) = 4; RT(Z) = 1; WT(X); WT(Y) = 1; WT(Z)
5	C READ(Z)	TS(TA); TS(TB) = 1; TS(TC) = 4 ; RT(X); RT(Y) = 4; RT(Z) = 4;

		WT(X); WT(Y) = 1; WT(Z)
6	A READ(X)	TS(TA) = 6; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 4; RT(Z) = 4; WT(X); WT(Y) = 1; WT(Z)
7	A WRITE(X)	TS(TA) = 6; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 4; RT(Z) = 4; WT(X) = 6; WT(Y) = 1; WT(Z)
8	C WRITE(Y)	TS(TA) = 6; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 4; RT(Z) = 4; WT(X) = 6; WT(Y) = 4; WT(Z)
9	C WRITE(Z)	TS(TA) = 6; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 4; RT(Z) = 4; WT(X) = 6; WT(Y) = 4; WT(Z) = 4
10	B READ(X)	TS(TA) =1 < RT(X) = 6, so TB needs to be rolled back and restarted
11	A READ(Y)	TS(TA) = 6; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 6; RT(Z) = 4; WT(X) = 6; WT(Y) = 4; WT(Z) = 4
12	A WRITE(Y)	TS(TA) = 6; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 4; RT(Z) = 4; WT(X) = 6; WT(Y) = 6; WT(Z) = 4
13	B READ(Z)	TS(TA) = 13; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 4; RT(Z) = 13; WT(X) = 6; WT(Y) = 6; WT(Z) = 4
14	B READ(Y)	TS(TA) = 13; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 13; RT(Z) = 13; WT(X) = 6; WT(Y) = 6; WT(Z) = 4
15	B WRITE(Y)	TS(TA) = 13; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 6; RT(Y) = 13; RT(Z) = 13; WT(X) = 6; WT(Y) = 13; WT(Z) = 4
16	B READ(X)	TS(TA) = 13; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 13; RT(Y) = 13; RT(Z) = 13; WT(X) = 6; WT(Y) = 13; WT(Z) = 4

17	B WRITE(X)	TS(TA) = 13; TS(TB) = 1; TS(TC) = 4 ; RT(X) = 13; RT(Y) = 13; RT(Z) = 13; WT(X) = 13; WT(Y) = 13; WT(Z) = 4

When time 10, B read(Z), the timestamp of transaction B is 1 which is less than the read_timestamp(A) = 6; thus transaction B needs to abort and restart at time 13 and got a new timestamp TS(TB) = 13;

6. Below is a log corresponding to a particular schedule at the point of a system crash for 4 transactions T1, T2, T3, and T4. Suppose that we use the immediate update protocol with checkpointing. Describe the recovery process from the system crash. Specify which transactions are rolled back, which operations in the log are redone and which operations in the log are undone. State whether any cascading rollbacks take place. (20 POINTS)

Start TRANSACTION 1
T1 READ(A)
T1 READ(D)
T1 WRITE(D, 20, 25)
COMMIT T1
CHECKPOINT
Start TRANSACTION T2
T2 READ(B)
T2 WRITE B 12, 18
Start TRANSACTION T4
T4 READ(D)
T4 WRITE (D, 25, 15)
START TRANSACTION T3
T3 WRITE(C,30,40)
T4 READ(A)
T4 WRITE(A, 30, 20)
T4 COMMIT

T2 READ(D)
T2 WRITE(D,15,25)
SYSTEM CRASH

Transaction 1 has already committed before CHECKPOINT time, thus **T1 has been written to secondary storage**.
Transaction 4 committed after CHECKPOINT time and before time SYSTEM CRASH, so the **T4 needs to redo**.
Transaction 2 and 3 didn't commit before SYSTEM CRASH time, so **T2 AND T3 need to undo**.
There is no cascade rollback in this case

7. Describe what a cascading rollback is. (10 points)

Cascading rollback is when a single transaction failure leads to a series of transaction rollbacks. (Reference from T6_L2_Safe_Schedules.ppt) Which means, for example, when T3 abort and roll back, and T2 which depend on T3, also needs to be rolled back. While T1 depend on T2, also roll back along with T2, this situation calls cascading rollback.