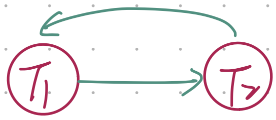
1. Given the following 2 different transactions. List all potential schedules for T1 and T2 and determine which schedules are conflict serializable and which are not. Assume at each time interval a command is issued by one of the transactions. Remember a schedule will vary from another schedule if by chance a transaction accesses a data object in a different order than the other schedule. Also note, different schedules may generate the same precedence graph. (15 POINTS)

Transaction 1	Transaction 2
Start transaction	Start transaction
READ(X)	READ(X)
X = X -N	X = X +M
WRITE(X)	WRITE(X)
Commit	Commit

S_1: $T_1(r)$ $T_2(r)$ $T_1(w)$ $T_2(w)$

not conflict-serializable

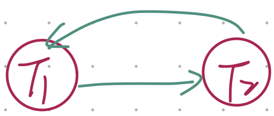
T_1	T_2
r	
	r
w	
	w



S_2: $T_2(r)$ $T_1(r)$ $T_2(w)$ $T_1(w)$

not conflict-serializable

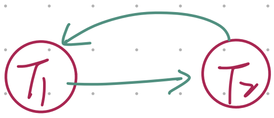
T_1	T_2
	r
r	
	w
w	



S_3: $T_1(r)$ $T_2(r)$ $T_2(w)$ $T_1(w)$

not conflict-serializable

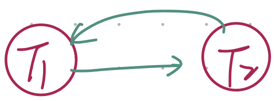
T_1	T_2
r	
	r
	w
w	



S_4: $T_2(r)$ $T_1(r)$ $T_1(w)$ $T_2(w)$

not conflict-serializable

T_1	T_2
	r
r	
w	
	w



S_5: $T_1(r)$ $T_1(w)$ $T_2(r)$ $T_2(w)$

conflict-serializable

T_1	T_2
r	
w	
	r
	w

S_6: $T_2(r)$ $T_2(w)$ $T_1(r)$ $T_1(w)$

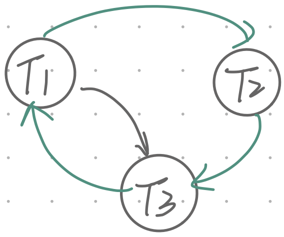
conflict-serializable

T_1	T_2
	r
	w
r	
w	

2.

①

T_1	T_2	T_3
r(x)		
		r(x)
w(x)		
	r(x)	
		w(x)

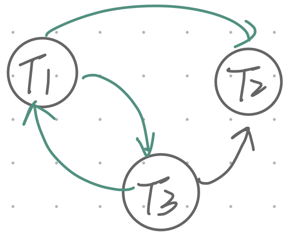


Cycle

not conflict serializable

②

T_1	T_2	T_3
r(x)		
		r(x)
		w(x)
w(x)		
	r(x)	

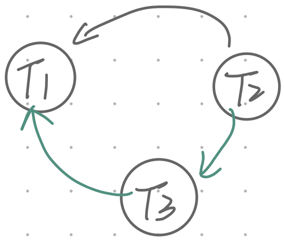


cycle

not conflict serializable

③

T_1	T_2	T_3
		r(x)
	r(x)	
		w(x)
r(x)		
w(x)		



Ascyclic

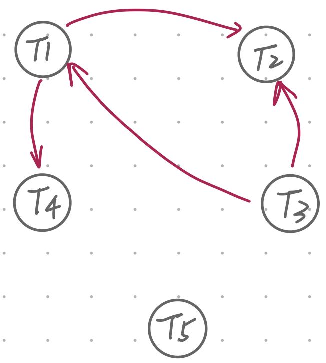
conflict serializable ✓

3. third one ; $T_2 \to T_3 \to T_1$

4

4. Create a waits-for graph for the following lock table. (10 points)

Transaction	Items locked	Items Waiting on
T1	X	Y, Z
T2	Y	
T3		X,Y
T4	Z	
T5		

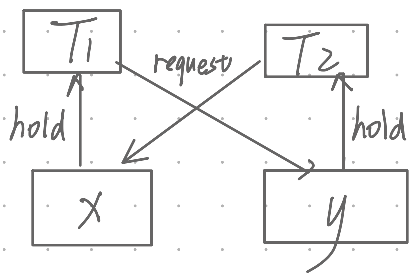


T_1 waits for T_2, T_4

T_3 waits for T_1, T_2

5. T_1 hold x require y

T_2 hold y require x



Transaction T_1 holds a lock on x and need to update y. At same time, transaction T_2 holds a lock on y and need x. T_1 will wait for T_2 to give up the lock, and T_2 also will wait for T_1 to give up the lock. → stuck in a deadlock.

6.

Thomas' Write Rule:

It is an optimization of the basic timestamping algorithm that reduces the number of transaction rollbacks by ignoring certain write operations rather than rejecting them.
if a more recent transaction has already written the value of an object, then a less recent transaction does not need to perform its write since the more recent one will eventually overwrite it.
It provides an optimization for dealing with stale updates of a data object. TW identifies when stale updates can be safely ignored and not completed，- TS(T) < WTS(X), instead of rolling back T, it simply ignores the write operation and lets T continue - Otherwise, the write operation proceeds as in the basic time stamping algorithm

Basic Timestamping Algorithm: :

Each transaction is assigned a unique timestamp when it enters the system. Similarly, each data item in the database has a read timestamp (RTS) and a write timestamp (WTS) that represent the largest timestamps of any transaction that successfully read or wrote the item.
It ensures that the timestamp for the transactions determines the order that the conflicting operations are completed，The rules for this algorithm are:
If a transaction T issues a read(X) operation:
- If TS(T) < WTS(X), the read is rejected because it would cause a read from the future, and T is rolled back
- Otherwise, the read is executed, and RTS(X) is updated if TS(T) > RTS(X)
If a transaction T issues a write(X) operation:
- If TS(T) < RTS(X) OR TS(T) < WTS(X), the write is rejected because it would cause the write to the past overwriting more recent data, and T is rolled back
- Otherwise, the write is executed, and WTS(X) is updated to TS(T)

Diff:

The second rule in the timestamp ordering protocol states that Ti is rolled back if write (K) is issued by Ti and TS[Ti]<W-ts(K). This is the one difference between thomas write and timestamp protocol. If TS(Ti) = R timestamp(K), the Thomas writes rule allows the write operation to be disregarded.

7. Apply the basic timestamping algorithm to the following schedule. Assume that the transaction id issued to the latest transaction is 20 (transaction 20 is the latest transaction before transactions A, B, AND C). Determine if the schedule can be performed as is or if transactions will need to be restarted. If so, what transactions will need to be restarted given the basic timestamping ordering algorithm. (10 POINTS) → B

TIME	Transaction A	Transaction B	Transaction C
11		READ(Z) √	
12		READ(Y) √	
13		WRITE(Y) √	
14			READ(Y) √
15			READ(Z) √
16	READ(X) √		
17	WRITE(X) √		
18			WRITE(Y) √
19			WRITE(Z) √
20		READ(X) X rej	
21	READ(Y) √		
22	WRITE(Y) √		
23		WRITE(X)	

- Time=11: Ts(B)=21, read(z)→21

	X	Y	Z
R			21
W			

	A	B	C
Ts		21	

- Time=12: Ts(B)=21, read(y)→21

	X	Y	Z
R		21	21
W		21	

	A	B	C
Ts		21	

- Time=13: Ts(B)=21, write(y)→21

- Time=14: Ts(C)=22, read(y)→22

	X	Y	Z
R		22	21
W		21	

	A	B	C
Ts		21	22

- Time 15
Ts(c)=22. read(z)=22

	X	Y	Z
R		22	22
W		21	

	A	B	C
Ts		21	22

- Time 16
Ts(A)=23 read(x)=23

	X	Y	Z
R	23	22	22
W		21	

	A	B	C
Ts	23	21	22

- Time 17
Ts(A)=23. write(x)=23

	X	Y	Z
R	23	22	22
W	23	21	

	A	B	C
Ts	23	21	22

- Time 18
Ts(C)=22. write(y) was 21<22 → 22

	X	Y	Z
R	23	22	22
W	23	22	

	A	B	C
Ts	23	21	22

- Time 19
Ts(c)=22. write(z)=22

	X	Y	Z
R	23	22	22
W	23	22	22

	A	B	C
Ts	23	21	22

- Time 20
Ts(B)=21 read(x) was 23 > Ts(B)
rejected. roll back

- Time 21
Ts(A)=23 read(y) was 22 < Ts(A)

	X	Y	Z
R	23	23	22
W	23	22	22

	A	B	C
Ts	23	21	22

- Time 22
Ts(A)=23 write(y) was 22 < Ts(A)

	X	Y	Z
R	23	23	22
W	23	23	22

	A	B	C
Ts	23	21	22

- Time 23
B was rolled back.

8. 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. (10 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

needs to be undone

Rollback : Any transaction that had not committed before the crash with need to roll back, so, T2 and T3 (didn't commit before crash)

Redone : Any transaction that had committed after the most recent checkpoint need to be redone, so, T4

Undone : Any transaction that was active at the time of crash and had not committed need to be undone, so any WRITE operations by T2 and T3 must be undone

Cascading Rollbacks:
The immediate update protocol is in place with checkpointing, the database system ensures that transactions write only to data items that have been committed. Uncommitted transactions should not depend on other uncommitted data, so no cascading rollbacks should occur.

9. Describe what a cascading rollback is. Provide an example of a schedule with a cascade rollrock. (10 points)

Cascading rollback occurs when a transaction not yet committed but is rolled back, forcing other transactions that read uncommitted data written by it also be rolled back.

Time	Transaction T_1	Transaction T_2	Transaction T_3
1	R(A)		
2	W(A)		
3		R(A)	
4		W(A)	
5			R(A)
6			W(A)
7	Rollback		

T_1 rollback, all of its writes be undone,

→ T_2 read the data T_1 write, T_2 also need rollback

→ T_3 read data from T_2, T_3 also rollback.

10. Given the schedule below classified the schedule as recoverable or not recoverable. Please support your answer with a valid description. (5 points)

Time	Transaction 1	Transaction 2	Transaction 3
t_1	Start transaction 1		
t_2	Read(X) 2		Start transaction 3
t_3			Read(X) 4
t_4	Write(X) 5		
t_5	Commit 6		
t_6		Start transaction 7	Commit 8
t_7		Read(X) 9	
t_8		Commit	

Ans:

In database systems, a schedule is classified as recoverable if no transaction commits until all transactions from which it has read uncommitted data either commit or abort.

in step 9, T_2 reads X written by T_1 when T_1 has comitted, this conforms to the recoverable schedule requirments.

→ Schedule is recoverable.

recoverable schedule ensures that if a transaction T1 reads data written by another transaction T2, then T1 can only commit after T2 has committed.