Database Development and Design (CPT201)

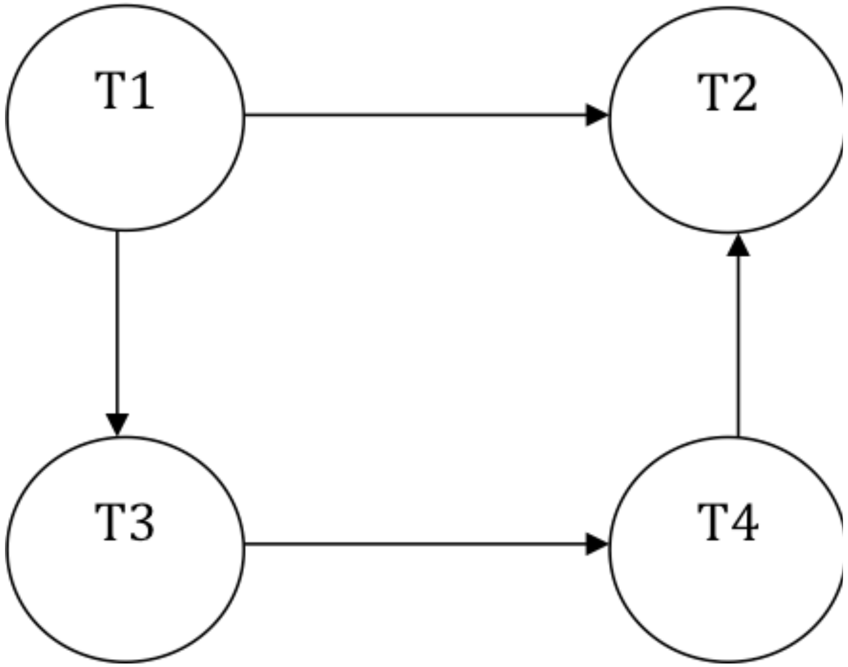
Tutorial 6

Dr. Wei Wang
Department of Computing

Q1

- Draw the precedence diagram for the schedule below and determine if it is conflict serialisable, and if yes, to which serial schedule it is equivalent to.
- Schedule: T1:read(X); T1:write(X); T1:read(Y); T2:read(X); T3:write(Y); T2:read(Z); T4:read(Y); T4:read(W); T4:write(W); T2:read(W);
- **NOTE**: you should draw the instructions of the schedule in a table first.

Q1 Solution



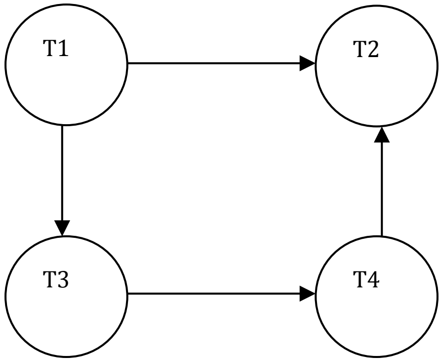
- No cycle, so it is conflict serialisable.

Q2

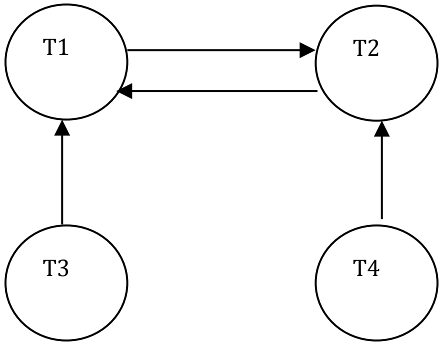
- Determine if the following two schedules are conflict serialisable by drawing the precedence graph. If so, identify an equivalent serial schedule. When drawing the precedence graph show all the edges – there is no requirement to show redundant edges.
  - S1: r1(X); w1(X); r1(Y); r2(X); w3(Y); r2(Z); r4(Y); r4(W); w4(W); r2(W);
  - S2: r1(X); r2(X); r1(Y); r2(Y); r3(X); r4(Y); w1(X); w2(Y);

Q2 Solution

- S1: This is a directed acycle graph. Therefore S1 is conflict serialisable and it is equivalent to a serial schedule of T1; T3; T4: T2



- S2: There is a cycle in the graph. Therefore S1 is not conflict serialisable.



Q3

- Is the following schedule recoverable? Justify your answer.
- T1:write(X); T1:write(Y); T2:read(X); T2:write(Y); T2:abort; T1:write(Z); T1:commit; T3:read(Y); T3:write(Z); T3:commit.

Q3 Solution

- The above schedule is recoverable.
- X is written by T1 and read by T2 and T2 is aborted;
- Y is written by T1 and read by T3 and T3 is committed after T1.

Q4

- Any problem with the following schedule using 2PL? Justify your answer.

T1	T2
lock-X(B);	
read(B);	
B := B − 50;	
write(B);	
	lock-S(A);
	read(A);
	lock-S(B);
lock-X(A);	
read(A);	
A := A + 50;	
write(A);	
unlock(B);	
unlock(A).	
	read(B);
	display(A + B);
	unlock(A);
	unlock(B).

Q4 Solution

- The deadlock problem will arise.
- Justification: T2 requests the shared lock on B, however, B is locked by T1 using the exclusive lock. T1 requests the exclusive lock on A, but A is locked by T2 using the shared lock.
- So neither of the transactions can proceed.

Q5

The assignments happen within the local memory space of the transactions and the effects of these assignments are not reflected in the database until the **write** operation. If the database failure happens immediately after time=7, (1) which transactions need to be redone and which need to be undone? Justify your answer (Describe how the recovery algorithm works). (2) what logs need to be added?

Time	T1	T2	T3
0			start
1			read(Y)
2			Y=Y+1
3	start		
4	read(X)		
5	X=X+1		
6			write(Y)
7			commit
8		start	
9		read(X)	
10		read(Y)	
11		Y=Y+X	
12		write(Y)	
13		commit	
14	read(Y)		
15	Y=Y+X		
16	write(X)		
17	----------------------	---Checkpoint ----	-------------------
18	commit		

Q5 Solution

- (1)
  - The recovery algorithm will scan from the last checkpoint, which is not shown in the above schedule. From this partial schedule, T1 (be more precise, part of the instructions in T1) and T3 need to be redone.
  - As T2 started after the failure, so it is not relevant.
  - At the end of the redo pass, as T3 already committed before the failure so it is removed from the undo list. So this pass will produce an undo list which only contains T1.
  - In the undo pass, T1 needs to be undone.
- (2)
  - <T1 abort>

Q6

- Examine the schedule given below. There are three transactions, T1, T2, and T3. Initially, the value of X = 1 and Y = 2.
- The assignments happen within the local memory space of the transactions and the effects of these assignments are not reflected in the database until the **write** operation.
- Assume that the operations happen instantaneously, so there is no delay on the execution of an operation in the schedule.

Q6 cont’d

time	T1	T2	T3
0			start
1			read(Y)
2			Y=Y+1
3	start		
4	read(X)		
5	X=X+1		
6			write(Y)
7			commit
8		start	
9		read(X)	
10		read(Y)	
11		Y=Y+X	
12		write(Y)	
13		commit	
14	read(Y)		
15	Y=Y+X		
16	write(X)		
17	----------------------	---Checkpoint ----	-------------------
18	commit		

Q6 cont'd

- (a) Show the log entries that would be generated by this execution. Use for the log the following notation, similar to the one used in the lecture.
- (b) Assume that the undo/redo algorithm with checkpoints is used. The database crashes immediately after statement 7. Assume that the log records up to now are on disk.
  - What transactions would have to be rolled back?
  - What transactions would need to be redone?
- (c) Whilst we maintain the assumption that the undo/redo algorithm with checkpoints is being used, let us now assume that the database crashes just after statement 17. Assume that all the log records until this point are on disk.
  - What transactions would have to be rolled back?
  - What transactions would have to be redone?
  - What logs need to be added after the successful recovery?

Q6 Solution

- (a)
  - <T3 start>
  - <T1 start>
  - <T3, Y, 2, 3>
  - <T3 commit>
  - <T2 start>
  - <T2, Y, 3, 4>
  - <T2 commit>
  - <T1, X, 1, 2>
  - <checkpoint {T1}>
  - <T1 commit>

Q6 Solution cont'd

- (b)
  - T1 has to roll back (however, part of the instructions in T1 need to be redone)
  - T3 has to be redone
- (c)
  - T1 has to roll back
  - No transaction needs to be redone
  - <T1, X, 1>
    <T1 abort>