Assignment 5
Class: SE Comp
Year: 2020-21
Performed by: Danyl Fernandes, 72

Q1. Consider the two transactions T1 and T2 and two schedules S1 and S2 as given below:

S1: R1(A) R1(B) R2(A) R2(C) W1(B) R3(B) R3(C) W3(B) W2(A) W2(C)

S2: R1(X) R3(Z) W3(Z) R2(Y) R1(Y) W2(Y) W3(X) W2(Z) W1(X)

Here R1(A) denotes read operation on A by transaction T1 and W1(A) denotes write operation on A by transaction T1

Which of the above schedules is conflict serializable?

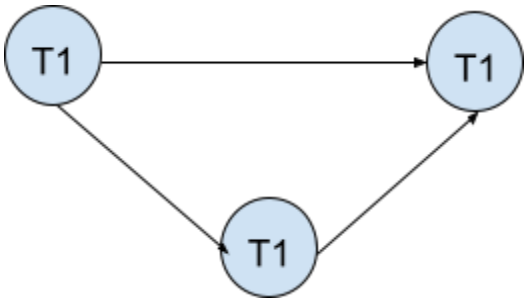
Schedule 1:

R1(A) R1(B) R2(A) R2(C) W1(B) R3(B) R3(C) W3(B) W2(A) W2(C)

T1	T2	T3
R1(A)		
R1(B)		
	R2(A)	
	R2(C)	
W1(B)		
		R3(B)
		R3(C)
		W3(B)
	W2(A)	
	W2(C)	

T1	T2	T3
R1(A)		
R1(B)		
W1(B)		R3(B)
		R3(C)
		W3(B)
	R2(A)	
	R2(A)	
	R2(C)	
	W2(A)	
	W2(C)	

- Shifting W1(B) above and shifting R2(A) and R2(C) below.
- Shifting R3(B), R3(C) and W3(B) above.
- Shifting R2(A) and R2(C).



Hence the schedule is conflict serializable since no cycle formed.

Order:

- T1:
  - Indegree = 0
  - Outdegree = 2
- T2:
  - Indegree = 2
  - Outdegree = 0
- T3:
  - Indegree = 1
  - Outdegree = 1

Indegree(T1) = 0.
Hence, the first transaction is T1.
Remove node T1 from precedence graph.

- T2:
  - Indegree = 1
  - Outdegree = 0
- T3:
  - Indegree = 0
  - Outdegree = 1

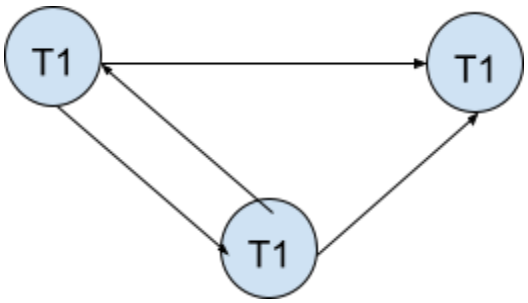
Indegree(T3) = 0.
Hence, the second transaction is T3.
Remove node T3 from precedence graph.

T2:
Indegree = 0
Outdegree = 0

Indegree(T2) = 0.
Hence, the third transaction is T2.
Remove node T2 from precedence graph.
Hence, order is T1 => T3 => T2

Schedule 2:

R1(X) R3(Z) W3(Z) R2(Y) R1(Y) W2(Y) W3(X) W2(Z) W1(X)



Hence the schedule is not conflict serializable since a cycle formed.

Q2. Explain lock based concurrency control technique and its disadvantage

Lock Based Protocols:

A lock is a variable associated with a data item that describes a status of a data item with respect to possible operation that can be applied to it. They synchronize the access by concurrent transactions to the database items. It is required in this protocol that all the data items must be accessed in a mutually exclusive manner. Let me introduce you to two common locks which are used and some terminology followed in this protocol.

- **Shared Lock (S):**
  - also known as Read-only lock. As the name suggests it can be shared between transactions because while holding this lock the transaction does not have the permission to update data on the data item. S-lock is requested using lock-S instruction.
- **Exclusive Lock (X):**
  - Data items can be both read as well as written.This is Exclusive and cannot be held simultaneously on the same data item. X-lock is requested using lock-X instruction.A transaction may be granted a lock on an item if the requested lock is compatible with locks already held on the item by other transactions.
  - Any number of transactions can hold shared locks on an item, but if any transaction holds an exclusive(X) on the item no other transaction may hold any lock on the item.
  - If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. Then the lock is granted.
- **Upgrade / Downgrade locks:**
  - A transaction that holds a lock on an item A is allowed under certain conditions to change the lock state from one state to another.
  - **Upgrade:** A S(A) can be upgraded to X(A) if Ti is the only transaction holding the S-lock on element A.
  - **Downgrade:** We may downgrade X(A) to S(A) when we feel that we no longer want to write on data-item A. As we were holding X-lock on A, we need not check any conditions.
- **Lock Compatibility Matrix:**
  - Any number of transactions can hold shared locks on an item, but if any transaction holds an exclusive lock on the item no other transaction may hold any lock on the item.

	S	**X**
S	True	False
X	False	False

- If a lock cannot be granted, the requesting transaction is made to wait till all incompatible locks held by other transactions have been released. The lock is then granted.

Lock Conversions:

- Two-phase locking with lock conversions:
  - First Phase:
    - can acquire a lock-S on item
    - can acquire a lock-X on item
    - can convert a lock-S to a lock-X (upgrade)
  - Second Phase:
    - can release a lock-S
    - can release a lock-X
    - can convert a lock-X to a lock-S (downgrade)
- This protocol ensures serializability.
- But still relies on the programmer to insert the various locking instructions.

Disadvantages:

Deadlock: consider the above execution phase. Now, T1 holds an Exclusive lock over B, and T2 holds a Shared lock over A. Consider Statement 7, T2 requests a lock on B, while in Statement 8 T1 requests lock on A. This as you may notice imposes a Deadlock as none can proceed with their execution.

Starvation: is also possible if concurrency control manager is badly designed. For example: A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request and are granted an S-lock on the same item. This may be avoided if the concurrency control manager is properly designed.

Exclusivity: Only one thread can enter the critical section, if there are multiple readers and they can read at the same time, it is suboptimal in this case. Two-phase locking does not ensure freedom from deadlocks. In addition to deadlocks, there is a possibility of starvation. Starvation occurs if the concurrency control manager is badly designed.

For example:

- A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request and are granted an S-lock on the same item.
- The same transaction is repeatedly rolled back due to deadlocks.

Concurrency control manager can be designed to prevent starvation