COURSE : DATABASE MANAGEMENT SYSTEMS
TOPIC : CONCURRENCY CONTROL-2

COURSE CODE: 23AD2102R

Session - 5

AIM OF THE SESSION

To familiarize students with the Basic concepts about Concurrency Techinques and Locking Protocols

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

1. Discuss about Terminology of Concurrency and Locking Protocols.
2. Describe the Locking Protocols.
3. Describe Two phase locking with suitable example.
4. Discuss about Deadlock prevention in locking .

LEARNING OUTCOMES

At the end of this session, students should be able to:

1. Review of Cascading Rollback & Cascade less Schedule
2. Concurrency Control: Lock based Protocols

SESSION OBJECTIVE

At the end of the session the students will understand

- ***Graph based Protocols***
- ***Deadlock Handling and Prevention***
- ***Deadlock Detection and Recovery***

DEADLOCK PROBLEM IN LOCK BASED PROTOCOLS

- A deadlock occurs during two-phase locking when a transaction is waiting for an item that is locked exclusively by another.
- Suppose Transaction 1 has locked item X and then attempts to lock item Y, but item Y has already been locked by Transaction 2. Under two-phase locking rules, Transaction 1 cannot proceed until it acquires the lock for item Y, so it is forced to wait.
- Meanwhile Transaction 2 has a lock on item Y, but needs item X to proceed with its operation. Transaction 2 cannot proceed either until it acquires the lock for item X. Neither transaction can proceed until the other releases the lock, but neither can release the lock until its operation is completed.

DEADLOCK PROBLEM IN 2-PHASE LOCKING

Time	T1	T2
t1	begin-trans	
t2	read-lock(A) → shared lock	
t3	read(A)	
t4	write-lock(B) →exclusive lock	
t5	read(B)	begin-trans
t6		write-lock(B) →not-granted
t7		wait
t8		wait
t9		wait
t10		

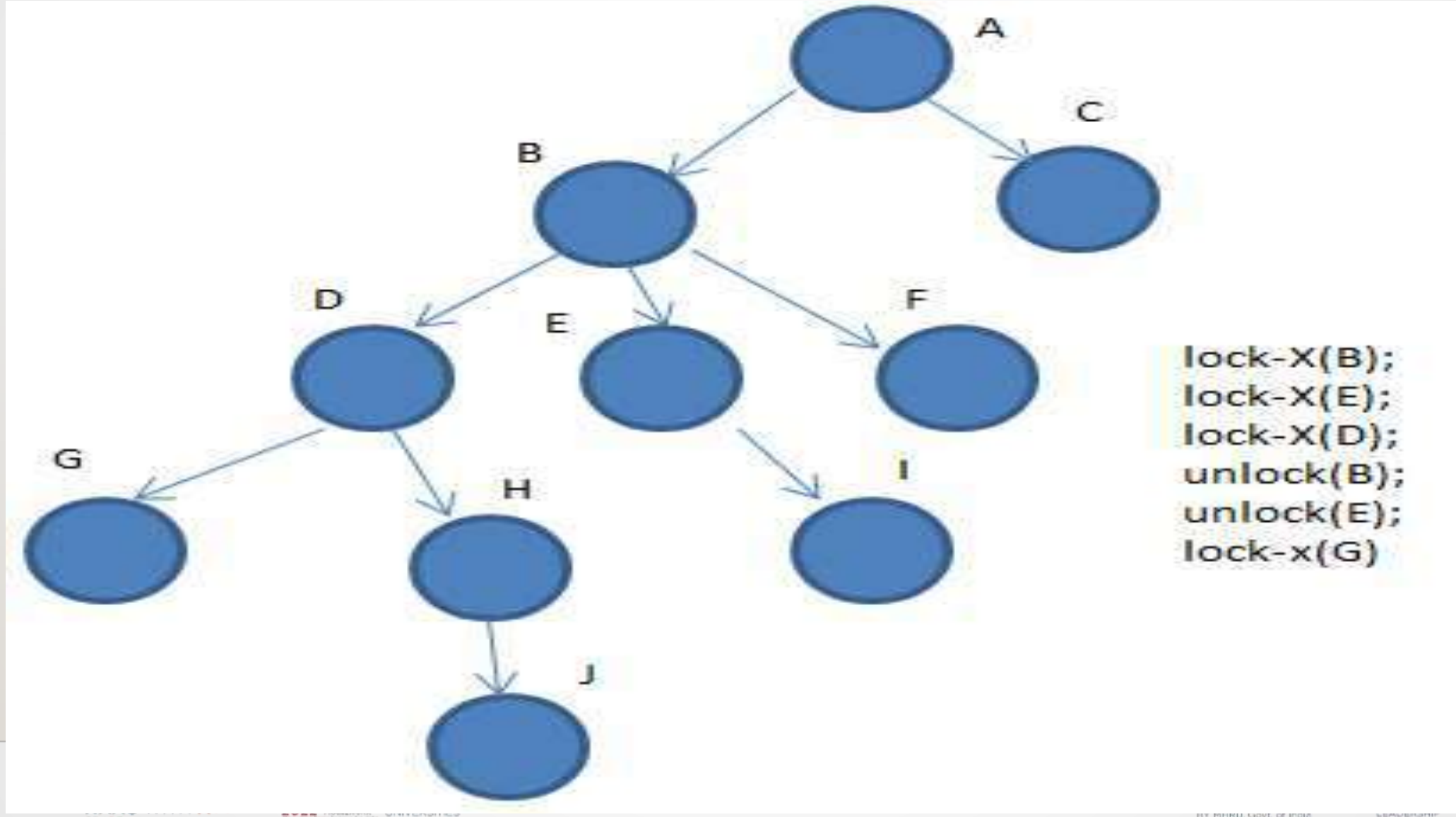
In Transaction 1 there are two locks being implemented. During time t2, read-lock is being used on value A because value A is needed to read only and write-lock is being used on value B because value B is needed for modifying or writing data. Meanwhile Transaction 2 begins and write-lock is being requested and this is when dead-lock happens which the write-lock in Transaction 2 Is not granted because value A is being write-lock in Transaction T1. As we know in growing phase only locks are allowed until all the operations is done in that following Transaction only, release lock will be performed which is known as during shrinking phase.

GRAPH BASED PROTOCOLS

- Also called as Tree locking Protocols
- Only exclusive locks are allowed
- The first lock by T_i may be on any data item. Subsequently a data item Q can be locked by T_i only if the parent of Q is currently locked by T_i
- Data items may be unlocked at any time.
- A data item that has been locked and unlocked by T_i cannot subsequently be relocked by T_i
- Unlocking may occur earlier in the tree locking protocol than in the two phase locking protocol

- **The tree protocol ensures conflict serializability as well as freedom from deadlock(does not ensure recoverability).**

GRAPH BASED PROTOCOLS



ADVANTAGES &DISADVANTAGES OF GRAPH BASED PROTOCOLS

Advantages :

1. Shorter waiting times, and increases in concurrency.
2. Protocol is deadlock free, no rollbacks are required.
3. Freedom from deadlocks

Disadvantages:

1.Protocol does not guarantee recoverability or cascade freedom

(need to introduce commit dependiencies to ensure recoverability)

2. In tree-locking protocol, a transaction that needs to access data item A and J in the database graph, must lock not only A and J, but also data items B,D,H ,this additional locking results in increased locking overhead and the possibility of additional waiting time and potential decrease in concurrency.

DEADLOCK HANDLING

- A system is in a deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set.
- More precisely, there exists a set of waiting transactions {T0, T1, . . . , Tn} such that T0 is waiting for a data item that T1 holds, and T1 is waiting for a data item that T2 holds, and . . . , and Tn−1 is waiting for a data item that Tn holds, and Tn is waiting for a data item that T0 holds.
- None of the transactions can make progress in such a situation.
- The only remedy to this undesirable situation is for the system to invoke some drastic action, such as rolling back some of the transactions involved in the deadlock.
- Rollback of a transaction may be partial: That is, a transaction may be rolled back to the point where it obtained a lock whose release resolves the deadlock.

TWO METHODS TO DEAL DEADLOCK PROBLEM

- *Deadlock Prevention*
- *Deadlock Detection and Recovery*

DEADLOCK PREVENTION

There are two approaches to deadlock prevention.

- One approach ensures that no cyclic waits can occur by ordering the requests for locks, or requiring all locks to be acquired together.

- The other approach is closer to deadlock recovery, and performs transaction rollback instead of waiting for a lock, whenever the wait could potentially result in a deadlock.

DEADLOCK PREVENTION:APPRAOCH-1

- Ensuring that no cyclic waits can occur by ordering the requests for locks, or requiring all locks to be acquired together.
- It requires that each transaction locks all its data items before it begins execution. Moreover, either all are locked in one step or none are locked.

DEADLOCK PREVENTION:APPRAOCH-2

- The second approach for preventing deadlocks is to use preemption and transaction rollbacks.
- In preemption, when a transaction T_j requests a lock that transaction T_i holds, the lock granted to T_i may be **preempted** by rolling back of T_i, and granting of the lock to T_j.
- To control the preemption, we assign a unique timestamp, based on a counter or on the system clock, to each transaction when it begins.
- The system uses these timestamps only to decide whether a transaction should wait or roll back

DEADLOCK DETECTION

- Deadlocks can be described precisely in terms of a directed graph called a **wait-for graph**.

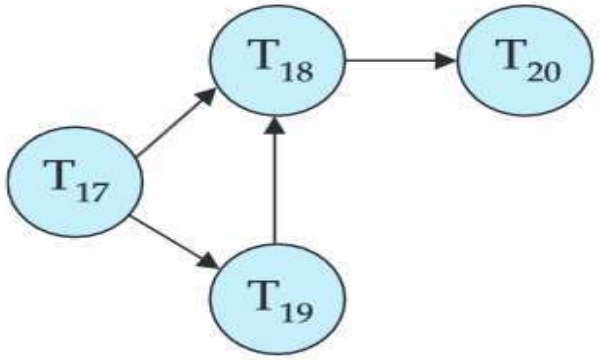


Figure 15.13 Wait-for graph with no cycle.

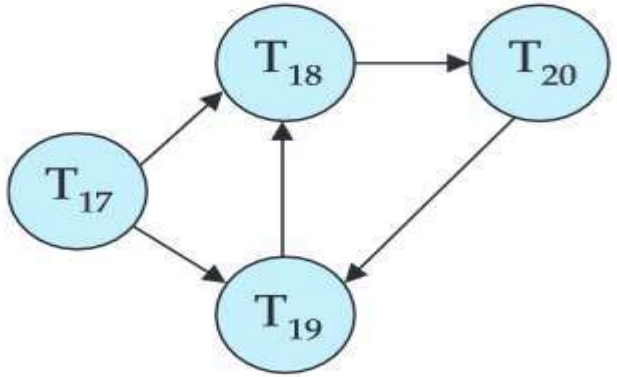


Figure 15.14 Wait-for graph with a cycle.

- set of vertices and E is a set of edges.
- The set of vertices consists of all the transactions in the system.
- Each element in the set *E* of edges is an ordered pair $T_i \rightarrow T_j$.
- If $T_i \rightarrow T_j$ is in *E*, then there is a directed edge from transaction Ti to Tj , implying that transaction T is waiting for transaction T to release a data item that it needs.

DEADLOCK DETECTION

When transaction T_i requests a data item currently being held by transaction T_j, then the edge $T_i \rightarrow T_j$ is inserted in the wait-for graph. This edge is removed only when transaction T_j is no longer holding a data item needed by transaction T_i.

A deadlock exists in the system if and only if the wait-for graph contains a cycle. Each transaction involved in the cycle is said to be deadlocked. To detect deadlocks, the system needs to maintain the wait-for graph, and periodically to invoke an algorithm that searches for a cycle in the graph.

RECOVERY FROM DEADLOCK

- When a detection algorithm determines that a deadlock exists, the system must recover from the deadlock.
- The most common solution is to roll back one or more transactions to break the deadlock. Three actions need to be taken:

- ***Selection of Victim***
- ***Rollback***
- ***Starvation***

SELECTION OF VICTIM

- Given a set of deadlocked transactions, we must determine which transaction (or transactions) to roll back to break the deadlock.

We should roll back those transactions that will incur the minimum cost.

Unfortunately, the term minimum cost is not a precise one.

Many factors may determine the cost of a rollback, including:

a. How long the transaction has computed, and how much longer the transaction will compute before it completes its designated task.

b. How many data items the transaction has used.

c. How many more data items the transaction needs for it to complete.

d. How many transactions will be involved in the rollback.

ROLLBACK

- Once we have decided that a particular transaction must be rolled back, we must determine how far this transaction should be rolled back.
- The simplest solution is a ***total rollback***: Abort the transaction and then

restart it.

- However, it is more effective to roll back the transaction only as far as necessary to break the deadlock. ***Such partial rollback*** requires the system to maintain additional information about the state of all the running transactions.
- Specifically, the sequence of lock requests/grants and updates performed by the transaction needs to be recorded.
- The deadlock detection mechanism should decide which locks the selected transaction needs to release in order to break the deadlock.
- The selected transaction must be rolled back to the point where it obtained the first of these locks, undoing all actions it took after that point.

ROLLBACK & STARVATION

- The recovery mechanism must be capable of performing such partial rollbacks.
- Furthermore, the transactions must be capable of resuming execution after a partial rollback.
- ***Starvation***: In a system where the selection of victims is based primarily on cost factors, it may happen that the same transaction is always picked as a victim.
- As a result, this transaction never completes its designated task, thus there is starvation. We must ensure that a transaction can be picked as a victim only a (small) finite number of times.
- The most common solution is to include the number of rollbacks in the cost factor.

SUMMARY OF THE SESSION

- Graph-based locking protocols impose restrictions on the order in which items are accessed, and can thereby ensure serializability without requiring the use of two-phase locking, and can additionally ensure deadlock freedom.
- Various locking protocols do not guard against deadlocks. One way to prevent deadlock is to use an ordering of data items, and to request locks in a sequence consistent with the ordering.
- Another way to prevent deadlock is to use preemption and transaction rollbacks. To control the preemption, we assign a unique timestamp to each transaction. The system uses these timestamps to decide whether a transaction should wait or roll back.
- If a transaction is rolled back, it retains its old time stamp when restarted.

SUMMARY OF THE SESSION

- If deadlocks are not prevented, the system must deal with them by using a deadlock detection and recovery scheme.

To do so, the system constructs a wait-for graph.

- A system is in a deadlock state if and only if the wait-for graph contains a cycle.
- When the deadlock detection algorithm determines that a deadlock exists, the system must recover from the deadlock.
- It does so by rolling back one or more transactions to break the deadlock.

SELF-ASSESSMENT QUESTIONS

A transaction is made to wait until all _____ locks held on the item are released

(a) Compatible
(b) Equivalent
(c) Compatible
(d) Executable

2. The protocol that indicates when a transaction may lock and unlock each of the data items is called as

a) Locking protocol
b) Unlocking protocol
c) Granting protocol
d) Conflict protocol

TERMINAL QUESTIONS

- 1. Explain different types of Locking Protocols

2. Describe the advantages and disadvantages of Two phase locking protocol

3. What do you mean by concurrency control techniques?

4. Explain the working of locking technique in concurrency control. What benefits does Rigorous two-phase locking provide? How does it compare with other forms of two-phase locking?