LOCK CONVERSIONS

Let's revisit

1. Let's go back and understand Different types of locks
2. And also what Two Phase Locking protocol is..

Okay!! Let us learn something new..

LOCK CONVERSIONS!!!

Lock conversions

1. Refinement of two-phase locking protocol
2. Provide a mechanism for upgrading a shared lock to an exclusive lock, and downgrading an exclusive lock to a shared lock.
3. We denote conversion from shared to exclusive modes by **upgrade**, and from exclusive to shared by **downgrade**.
4. **Upgrading can take place in only the growing phase**
5. **Downgrading can take place in only the shrinking phase.**

Why??

Concurrent execution of t1 and t2 is not possible as shared lock cannot be given above an exclusion lock.

T1	T2
X(A)	
R(A)	S(A)
X(B)	Waiting...
R(B)	
W(A)	

Solution

T1 could initially lock A in shared mode and before the write operation on A, to upgrade it to exclusive lock on A.

Concurrent execution is possible here

Shared(A) —Upgrading→ **Exclusive(A)**

Exclusive A) —Downgrading→ **Shared(A)**

T1	T2
S(A)	
R(A)	S(A)
X(B)	R(A)
R(B)	
Upgrade(A)	
W(A)	

Implementation of Locking - Lock Manager

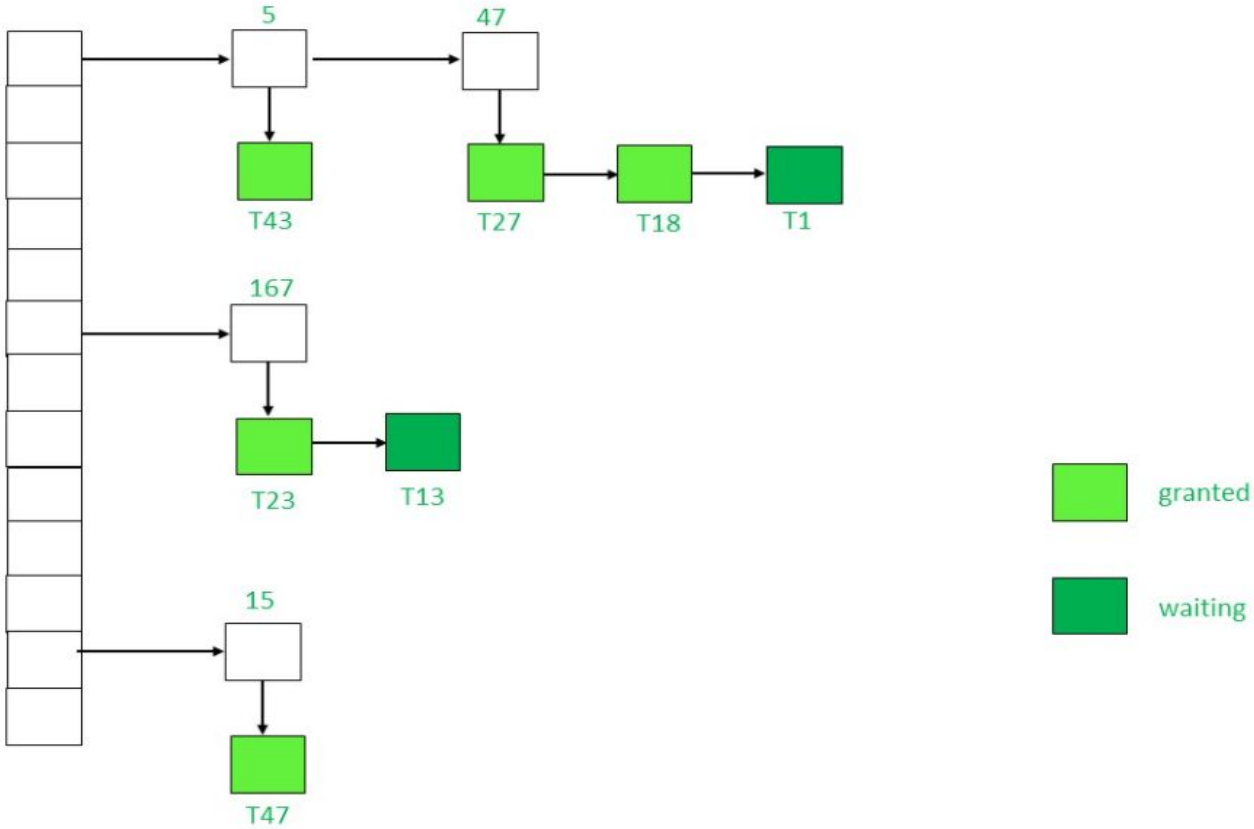
1. Locking protocols are used in database management systems as a means of concurrency control.
2. Multiple transactions may request a lock on a data item simultaneously. Hence, we require a mechanism to manage the locking requests made by transactions. Such a mechanism is called as **Lock Manager**.
3. It relies on the process of message passing where transactions and lock manager exchange messages to handle the locking and unlocking of data items.

Implementation of Locking - Lock Manager (cont.)

Data structure used in Lock Manager –

1. The data structure required for implementation of locking is called as **Lock table.**
   a. It is a hash table where name of data items are used as hashing index.
   b. Each locked data item has a linked list associated with it.
   c. Every node in the linked list represents the transaction which requested for lock, mode of lock requested (mutual/exclusive) and current status of the request (granted/waiting).
   d. Every new lock request for the data item will be added in the end of linked list as a new node.
   e. Collisions in hash table are handled by technique of separate chaining.

Implementation of Locking - Lock Manager (cont.)



Implementation of Locking - Lock Manager (cont.)

Working of Lock Manager –

1. Initially the lock table is empty as no data item is locked.
2. Whenever lock manager receives a lock request from a transaction Ti on a particular data item Qi following cases may arise:
   a. If Qi is not already locked, a linked list will be created and lock will be granted to the requesting transaction Ti.
   b. If the data item is already locked, a new node will be added at the end of its linked list containing the information about request made by Ti.
3. If the lock mode requested by Ti is compatible with lock mode of transaction currently having the lock, Ti will acquire the lock too and status will be changed to 'granted'. Else, status of Ti's lock will be 'waiting'.
4. If a transaction Ti wants to unlock the data item it is currently holding, it will send an unlock request to the lock manager. The lock manager will delete Ti's node from this linked list. Lock will be granted to the next transaction in the list.
5. Sometimes transaction Ti may have to be aborted. In such a case all the waiting request made by Ti will be deleted from the linked lists present in lock table. Once abortion is complete, locks held by Ti will also be released.

AMRITA VISHWA VIDYAPEETHAM

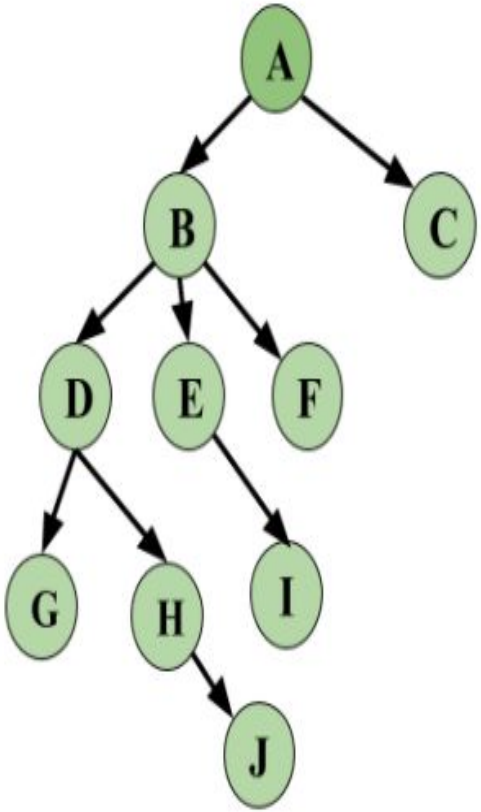
Graph Based Protocol

1. Graph Based Protocols are yet another way of implementing Lock Based Protocols.
2. As we know the prime problems with Lock Based Protocol has been avoiding Deadlocks and ensuring a Strict Schedule.
3. We've seen that Strict Schedules are possible with following Strict or Rigorous 2-PL. We've even seen that Deadlocks can be avoided if we follow Conservative 2-PL but the problem with this protocol is it cannot be used practically.
4. Graph Based Protocols are used as an alternative to 2-PL. Tree Based Protocols is a simple implementation of Graph Based Protocol.
5. A prerequisite of this protocol is that we know the order to access a Database Item.
    a. For this we implement a Partial Ordering on a set of the Database Items (D) {d1, d2, d3, ..., dn} . The protocol following the implementation of Partial Ordering is stated as-
        i. If di –> dj then any transaction accessing both di and dj must access di before accessing dj.
        ii. Implies that the set D may now be viewed as a directed acyclic graph (DAG), called a database graph.

Tree Based Protocol

1. Partial Order on Database items determines a tree like structure.
2. Only Exclusive Locks are allowed.
3. The first lock by Ti may be on any data item. Subsequently, a data Q can be locked by Ti only if the parent of Q is currently locked by Ti.
4. Data items can be unlocked at any time.

Following the Tree based Protocol ensures Conflict Serializability and Deadlock Free schedule. We need not wait for unlocking a Data item as we did in 2-PL protocol, thus increasing the concurrency.



T1	T2	T3	T1 (CONT)	T2(CONT)	T3(CONT)
Lock-X(A)					Lock-X(E)
Lock-X(B)				Unlock-X(H)	
	Lock-X(D)		Lock-X(B)		
	Lock-X(H)		Lock-X(G)		
	Unlock-X(D)		Unlock-X(D)		
Lock-X(E)					Unlock-X(E)
Lock-X(D)					Unlock-X(B)
Unlock-X(B)			Unlock-X(G)		
Unlock-X(E)					
		Lock-X(B)			

Tree Based Protocol

Advantages

1. Ensures Conflict Serializable Schedule.
2. Ensures Deadlock Free Schedule
3. Unlocking can be done anytime

Disadvantages –

1. Unnecessary locking overheads may happen sometimes, like if we want both D and E, then at least we have to lock B to follow the protocol.
2. Cascading Rollbacks is still a problem. We don't follow a rule of when Unlock operation may occur so this problem persists for this protocol.

Thank you !!