DBMS: Concurrency Control

Akhilesh Arya

Concurrency Control

- Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database
- In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database

- In a database transaction, the two main operations are
  - **READ**
  - **WRITE**
- So, there is a need to manage these two operations
- following problems occur with the Concurrent Execution of the operations:
  - Lost Update Problems (**W - W Conflict**)
  - Dirty Read Problems (**W-R Conflict**)
  - Unrepeatable Read Problem (**R-W-R Conflict**)

Lost Update Problems (W - W Conflict)

- *When two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e., concurrent execution)*

Time	T_x	T_y
t_1	READ (A)	—
t_2	A = A - 50	
t_3	—	READ (A)
t_4	—	A = A + 100
t_5	—	—
t_6	WRITE (A)	—
t_7		WRITE (A)

LOST UPDATE PROBLEM

Dirty Read Problems (W-R Conflict)

- *When one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction*

Cont..

Time	T_x	T_y
t_1	READ (A)	—
t_2	A = A + 50	—
t_3	WRITE (A)	—
t_4	—	READ (A)
t_5	SERVER DOWN ROLLBACK	—

DIRTY READ PROBLEM

Unrepeatable Read Problem (W-R Conflict)

- *Also known as Inconsistent Retrievals Problem that occurs when in a transaction, two different values are read for the same database item*

Time	T_x	T_y
t_1	READ (A)	—
t_2	—	READ (A)
t_3	—	A = A + 100
t_4	—	WRITE (A)
t_5	READ (A)	—

UNREPEATABLE READ PROBLEM

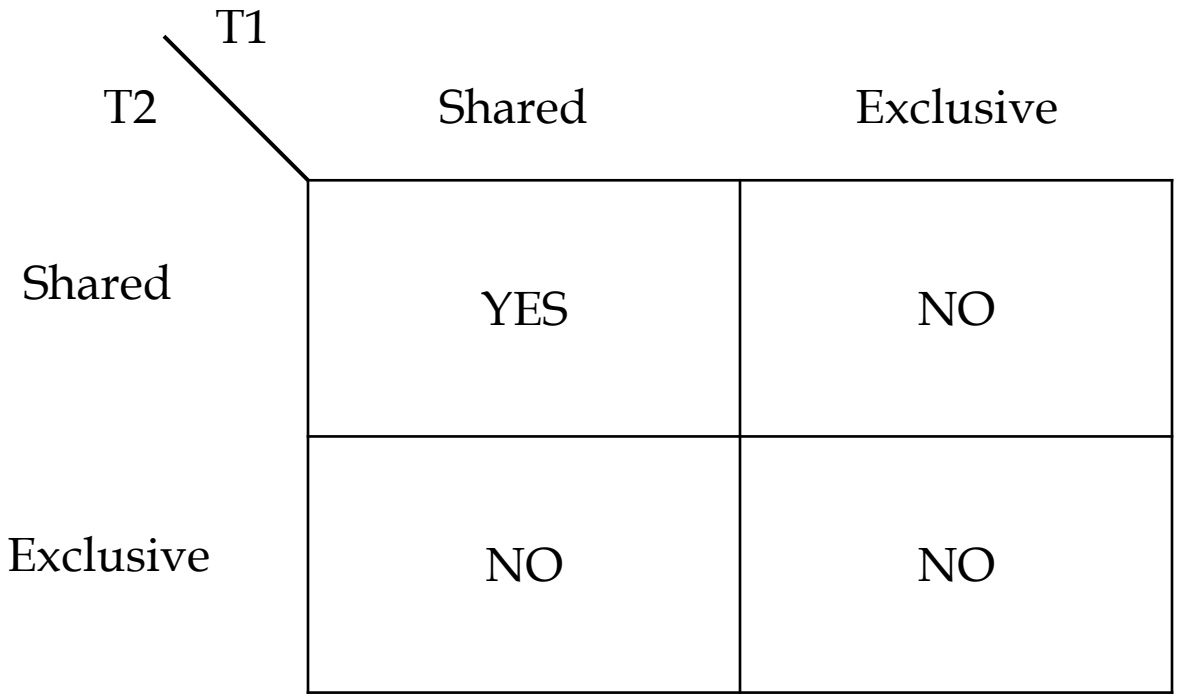
Concurrency Control Protocols

- The concurrency control protocols ensure the *atomicity, consistency, isolation, durability* and *serializability* of the concurrent execution of the database transactions
- These protocols are categorized as:
  - Lock Based Concurrency Control Protocol
  - Time Stamp Concurrency Control Protocol
  - Validation Based Concurrency Control Protocol

Lock Based Protocol

- In this type of protocol, any transaction cannot read or write data until it acquires an appropriate lock on it
- There are two types of lock:
  1. **Shared lock:**
     - It is also known as a Read-only lock. In a shared lock, the data item can only read by the transaction
     - It can be shared between the transactions because when the transaction holds a lock, then it can't update the data on the data item
  2. **Exclusive lock:**
     - In the exclusive lock, the data item can be both reads as well as written by the transaction.
     - This lock is exclusive, and in this lock, multiple transactions do not modify the same data simultaneously

Compatibility Table of Locks



- There are 4 types of locking protocols available as follows:
  - Simplistic lock protocol
  - Pre- claiming lock protocol
  - Two Phase locking protocol (2PL)
  - Strict Two Phase locking protocol (Strict- 2PL)

Simplistic Lock Protocol

- Simplistic lock-based protocols allow all the transactions to get the lock on the data before insert or delete or update on it
- It will unlock the data item after completing the transaction

Example

Can we ensure:
Serializablity
Cascadless
And recoverable schedule with this approach?

T1	T2
Lock-X(A)	
R(A)	
W(A)	
Unlock(A)	
	Lock-S(B)
	R(B)
	Unlock(B)
Lock-X(B)	
R(B)	
W(B)	
Unlock(B)	
	Lock-S(A)
	R(A)
	Unlock(A)

Pre- claiming Lock Protocol

- Pre-claiming Lock Protocols evaluate the transaction to list all the data items on which they need locks
- Before initiating an execution of the transaction, it requests DBMS for all the lock on all those data items
- If all the locks are granted then this protocol allows the transaction to begin
- When the transaction is completed then it releases all the lock.
- If all the locks are not granted then this protocol allows the transaction to rolls back and waits until all the locks are granted

Two Phase locking protocol (2PL)

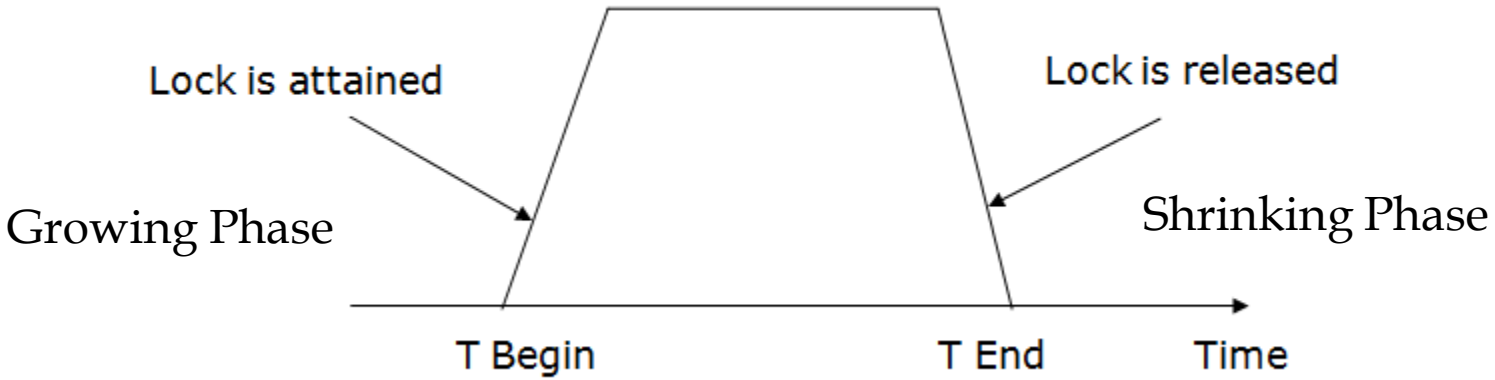
- The two-phase locking protocol divides the execution phase of the transaction into three parts:
  - In the first part, when the execution of the transaction starts, it **seeks permission** for the lock it requires
  - In the second part, the transaction **acquires all the locks**
  - The third phase is started as soon as the transaction **releases its first lock**

Note: ***Transaction cannot demand any new locks. It only releases the acquired locks. In the third phase***

Cont..

- There are two phases of 2PL:
  - **Growing phase:** In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released
  - **Shrinking phase:** In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired

- If lock conversion is allowed then the following phase can happen:
  - Upgrading of lock (from S(a) to X (a)) is allowed in growing phase.
  - Downgrading of lock (from X(a) to S(a)) must be done in shrinking phase.

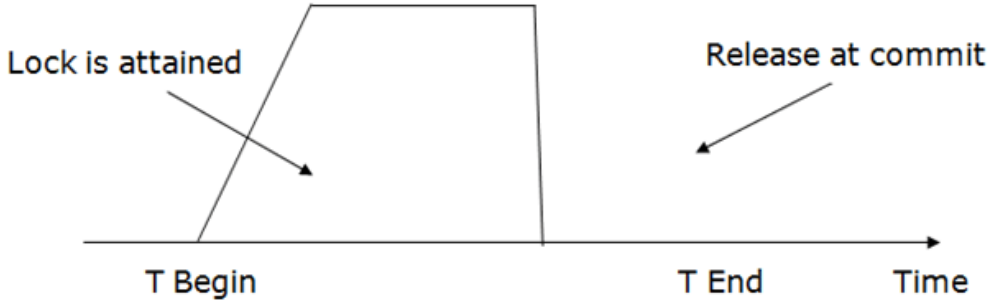


- Find the growing shrinking and lock point of the following transactions (T1 & T2)

	T1	T2
0	LOCK-S(A)	
1		LOCK-S(A)
2	LOCK-X(B)	
3	——	——
4	UNLOCK(A)	
5		LOCK-X(C)
6	UNLOCK(B)	
7		UNLOCK(A)
8		UNLOCK(C)
9	——	——

Strict Two-phase locking (Strict-2PL)

- The only difference between 2PL and strict 2PL is that Strict-2PL does not release a lock after using it
- Strict-2PL waits until the whole transaction to commit, and then it releases all the locks at a time
- Strict-2PL protocol does not have shrinking phase of lock release



Time Stamp Based Protocol

- The Timestamp Ordering Protocol is used to order the transactions based on their **Timestamps**
- The order of transaction is nothing but the **ascending order** of the transaction creation
- The priority of the **older transaction is higher** that's why it executes first
- To determine the timestamp of the transaction, this protocol uses system time or logical counter
- The timestamp ordering protocol also maintains the timestamp of last **'read' and 'write'** operation on a data

- Check the following condition whenever a transaction Ti issues a **Read (X)** operation:
  - If W_TS(X) >TS(Ti) then the operation is rejected
  - If W_TS(X) <= TS(Ti) then the operation is executed
  - Timestamps of all the data items are updated
- Check the following condition whenever a transaction Ti issues a **Write(X)** operation:
  - If TS(Ti) < R_TS(X) then the operation is rejected.
  - If TS(Ti) < W_TS(X) then the operation is rejected and Ti is rolled back

Validation based Protocol

- Validation phase is also known as **optimistic concurrency control technique**.
- In the validation based protocol, the transaction is executed in the following three phases:
  - **Read phase:** In this phase, the transaction T is read and executed. It is used to read the value of various data items and stores them in temporary local variables. It can perform all the write operations on temporary variables without an update to the actual database

Cont..

- **Validation phase:** In this phase, the temporary variable value will be validated against the actual data to see if it violates the serializability
- **Write phase:** If the validation of the transaction is validated, then the temporary results are written to the database or system otherwise the transaction is rolled back