CONCURRENCY CONTROL

Definition

- Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.
- But before knowing about concurrency control, we should know about concurrent execution.

Concurrent Execution

- 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.
- It means that the same database is accessed simultaneously on a multi-user system by different users.
- While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.

Concurrent Execution

- The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database.
- Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

Problems with concurrent execution

Some problems may occur during concurrent execution. Such as:

1. Lost Update Problem(W-W Conflict)
2. Dirty Read Problem(W-R Conflict)
3. Unrepeatable Read Problem(R-W Conflict)

Lost Update Problem

The problem occurs *when two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e., concurrent execution) that makes the values of the items incorrect hence making the database inconsistent.*

Lost Update Problem

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 Problem

The dirty read problem occurs *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. There comes the Read-Write Conflict between both transactions.*

Dirty Read Problem

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

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

- Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistencies in the database.
- Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

Concurrency Control Protocols

The concurrency control protocols ensure the *atomicity, consistency, isolation, durability* and *serializability* of the concurrent execution of the database transactions. The protocols are categorized as:

- Lock Based Concurrency Control Protocol
- Time Stamp Concurrency Control Protocol
- Validation Based Concurrency Control Protocol

Lock Based Protocols

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:

a) Shared Lock
b) Exclusive Lock

Shared Locks

- 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.

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.

Types of Locking Protocols

There are four types of locking protocols available. Such as:

a) Simplistic Lock Protocol
b) Two-Phase Locking Protocol
c) Strict Two-Phase Locking Protocol
d) Rigorous Locking

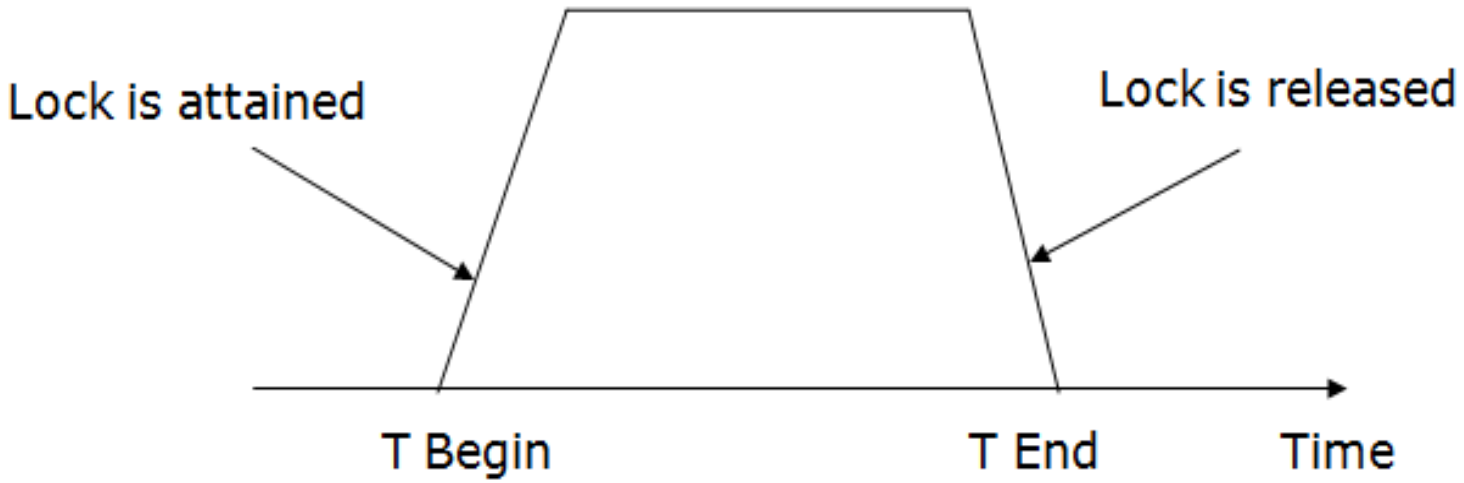
Simplistic Lock Protocol

- It is the simplest way of locking the data while transaction.
- 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.

Two Phase Locking Protocol

- 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.
- In the third phase, the transaction cannot demand any new locks. It only releases the acquired locks.

Two Phase Locking Protocol



Two Phase Locking Protocol

There are two phases of 2PL:

a) Growing phase: In the growing phase, a new lock on the data item may be acquired by the transaction, but none can be released.

b) Shrinking phase: In the shrinking phase, existing lock held by the transaction may be released, but no new locks can be acquired.

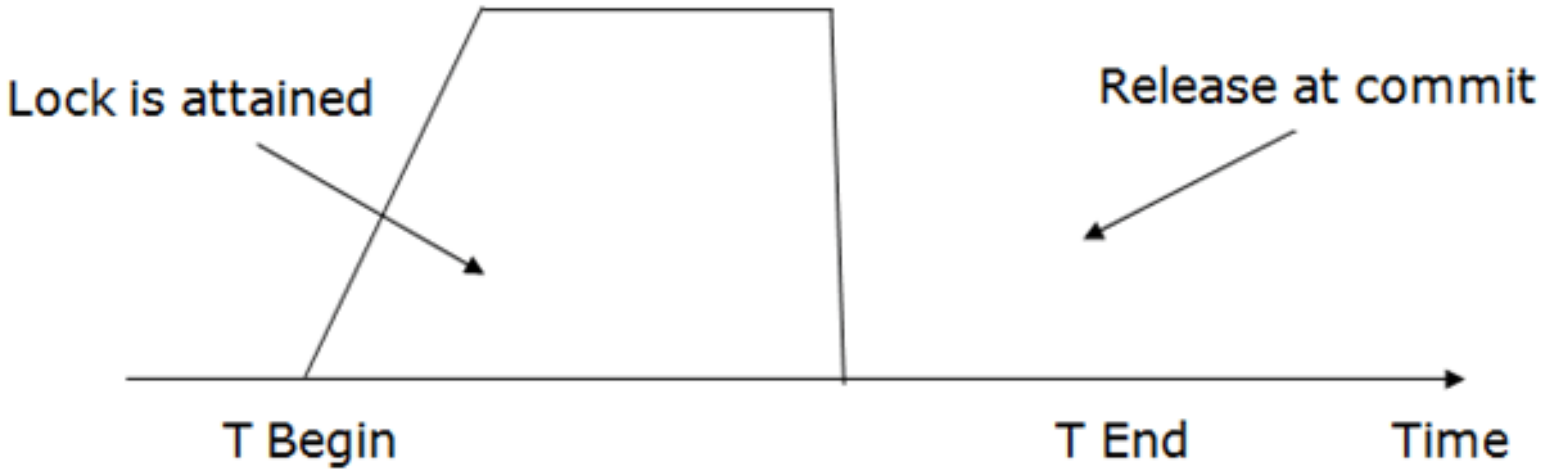
Growing and Shrinking Phase

	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

- The first phase of Strict-2PL is similar to 2PL. In the first phase, after acquiring all the locks, the transaction continues to execute normally.
- 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.

Strict Two-Phase Locking



Timestamp Concurrency Control

- Timestamp is a unique identifier created by DBMS to identify a transaction.
- Timestamp values are assigned in the order of creation of transaction.
- Timestamp of a transaction T is denoted as TS(T).
- This method of concurrency control never uses locks.

Timestamp Concurrency Control

- 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 lock-based protocol is used to manage the order between conflicting pairs among transactions at the execution time. But Timestamp based protocols start working as soon as a transaction is created.

Working Principle of TSO

1. 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.

2. 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 otherwise the operation is executed.

Here

TS(TI) denotes the timestamp of the transaction Ti.
R_TS(X) denotes the Read time-stamp of data-item X.
W_TS(X) denotes the Write time-stamp of data-item X.

Advantages of Timestamp Ordering

- It ensures serializability
- TS protocol ensures freedom from deadlock that means no transaction ever waits.

Disadvantage of Timestamp Ordering

- The schedule may not be recoverable and may not even be cascade- free.

Thank You