Concurrency Control Protocols

Concurrency control protocols ensure the atomicity, serializability and isolation of the concurrent transactions. The Concurrency control protocols can be broadly classified into the following categories:

1. Lock Based Protocol
2. Timestamp protocol

Lock Based Protocol

In this protocol, any transaction in the database cannot read or write any data item until it acquires an appropriate lock on it. This means that in the database, any transaction cannot retrieve, insert, delete, or update the data until it gets access to that specific data item. Any transaction proceeds only when the lock request is granted. Following are there two different types of protocol:

1. Shared Lock
2. Exclusive lock

Shared Lock(S)

This type of lock is also called a read-only lock. With the shared lock, the data item can only be read by the transaction. In the shared lock, the data item can be shared between the transactions because when a transaction holds this type of lock, then it cannot update the data item.

If a transaction Ti has obtained a shared lock which is denoted by S on an item Q, then Ti transaction can read, but cannot write, Q.

For example: When two transactions are reading the account balance of a person. Then, the database will let them read by placing the shared lock. But at the same time, if another transaction wants to update the account balance by placing, the shared lock does not allow it until the reading process is finished.

Exclusive Lock(X)

In this type of lock, a data item can be read as well as written by the transaction. Once this lock is placed on the data item, no other lock (either shared or Exclusive) can be placed on that data item until Exclusive lock is released. This lock is exclusive, and in this lock, multiple transactions in the database do not modify the same data item simultaneously.

If a transaction Ti has obtained an exclusive lock which is denoted by X on an item Q, then Ti transaction can both read and write Q.

For example: when a transaction wants to update the account balance of a person, then in database let it does by placing the X-lock on it. But if a second transaction wants to read the data, the shared lock does not allow it. And if another transaction wants to write the data, the exclusive lock doesn't allow that.

Lock Compatibility Matrix

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

There are four types of lock protocol available:

1. Simplistic lock protocol

This is the simplest way of locking the data items during transactions. This protocol allows all transactions to get the lock (shared or exclusive) on the data before insert, delete, or update operations performed on it. And this protocol will unlock the data item after completing the operations in the transaction.

2. Pre-claiming Lock protocol

This protocol evaluates those transactions which contain the data items that need to be locked. Before the transaction begins its execution, this protocol requests DBMS for all locks on all the data items.

When all the locks are granted, then it allow the transaction to start its execution and release all the locks when it is completed. If all the locks are not assigned to all the data items, then this protocol rolls back the transaction and wait until the locks are granted.

3. Two-Phase Locking Protocol (2PL)

This type of locking protocol (2PL) divides the execution phase of the transaction into three parts.

In the first part, when the transaction starts its execution, it seeks permission for the locks it requires. In the second part, any transaction in the database acquire all the locks. As soon as a transaction in database releases its first lock, the third phase starts. In this part, a transaction cannot demand any new lock, and here it only released all the acquired locks.

1. The two-phase locking protocol consists of two phases:
2. **a) Growing Phase:** The phase where the transaction acquires all the locks on the data items, but none can be released known as a growing phase.
3. **Shrinking Phase:** The phase where the existing locks held by the transaction may be released, but no new lock on the data item can be acquired. This type of phase is known as the shrinking phase.

Note: Lock Point: The point in a schedule where the transaction has obtained its final lock (end of growing phase) is called a lock point.

Note: If the conversion of lock is allowed, then in growing phase upgrading of a lock (from S(a) to X(a)) is allowed. And in shrinking phase downgrading of a lock (from X(a) to S(a)) must be done.

Example:

Time	Transaction T1	Transaction T2
t1	Lock-X(B)	Lock-S(A)
t2	Read(B)	Read(A)
t3	B=B-50	Lock-S(B)
t4	Write(B)	Read(B)
t5	Lock-X(B)	unlock(A)
t6	Lock-X(A)	unlock(B)
t7	Read(A)	
t8	A=A-50	
t9	Write(A)	
t10	unlock(B)	
t11	unlock(A)	

4. Strict two-phase locking protocol

The first phase of this protocol is similar to the two-phase locking protocol (2PL). In the first phase of this protocol, when the transaction acquired all the locks, it continues to execute normally.

The only difference between the two-phase locking protocol and strict two-phase locking protocol is that this protocol does not have a shrinking phase for releasing the locks. It waits till the whole transaction to finish and commit, and then it releases all the lock held by the transaction at a time.

Timestamp Based Protocol

The Timestamp-based protocol is the most commonly used concurrency control protocol and is used to order the execution of the concurrent transaction based on their Timestamp. This protocol uses the **logical counter** or **system time** to determine the timestamp of the transaction. It also maintains the timestamp of last **'read'** and **'write'** operations on the data.

The lock-based protocol acquires locks at the time of execution. But in this protocol, as soon as the transaction is created, it assigns the order of the concurrent transactions. The priority of an older transaction is higher; that's why it starts its execution first.

Let's suppose there are two transactions T1 and T2. Suppose the T1 transaction has entered the system at 005 times and transaction T2 has entered the system at 008 times. T1 transaction has the higher priority that's why it executes first as it is entered the system first.

Two types of Timestamp:
1. W_Timestamp(X): This represents the largest timestamp of any transaction that executes **write(X)** successfully.
2. R_Timestamp(X): This represents the largest timestamp of any transaction that executes **Read(X)** successfully.
Timestamp ordering protocol
The timestamp ordering protocol ensures that any conflicting read and write operation are executed in timestamp order.
Following are the two cases which describe the work of basic timestamp ordering protocol:
1. Suppose a transaction Ti read an item (X), check the following condition:

- If Timestamp(Ti) < W_Timestamp(X), then the read operation is rejected and Ti is rolled back.
- If Timestamp(Ti) > W_Timestamp(X), then the operation is executed.

2. Suppose a transaction Ti write an item (X), check the following condition:

- If Timestamp(Ti) < R_Timestamp(X) then the operation is rejected.
- If Timestamp(Ti) < W_Timestamp(X) then the write operation is rejected and Ti transaction is rolled back otherwise, other operations are executed.