CONCURRENCY CONTROL PROTOCOL

In the multi-user system , we all know that multiple transactions run in parallel, thus trying to access the same data and suppose if one transaction already has the access to the data item and now another transaction tries to modify the data then it leads to error in database. There are issues with the concurrent execution of transactions such as conflicting operation where in simultaneously both transactions might try to access the data item which leads to the problem in database. But there are many advantages of concurrent execution, So the possible solution to this is to learn methods to control the concurrency execution where we will know to manage transactions to ensure concurrent transaction with no problems at all.

Concurrency control can be simply defined as the process of managing the simultaneous execution of transactions in a shared database thus ensuring the serialization of transactions.

Concurrency controls aims towards isolation (transactions do not interfere with each other), to preserve the database consistency and to resolve the conflicting operations such as read-write and write-write.

Lets study the protocols available -

LOCK BASED PROTOCOL:

It provides a guaranteed access to the data item in database to only a current transaction. Lock is acquired when access to the data item has to be made and lock is released when the transaction is completed and then another transaction can have access to it. The only requirement in this protocol is all the data items have to be accessed in mutually exclusive manner which means if one transaction has the access to the data item, then the other transaction should not be allowed.

The locks are of two types

1. Binary Locks : The word binary itself implies it can have two states - locked and unlocked.

It is as simple as saying that you can either access the data item or you cannot.

2. Shared/ Exclusive: Acquires the lock based on its usage.

a) Shared lock (lock-S) : It is used when data item value just has to be read.

b) Exclusive lock (lock-X) : It is used when data item value has to updated, could be read or write.

	read	write
read	S	X
write	X	X

A concept called compatibility between lock modes comes into picture when dealing with the locks. Look at the table shown to understand compatibility of the modes.

- If one transaction has acquired shared lock and one more wants to acquire it, then it is surely allowed, that simply means if one transaction is reading a value other transaction can also read it. Lets recall the point that shared data is used for only reading the data item(not write).
- Shared and exclusive lock are not compatible with each other which means that when one transaction acquires the exclusive lock, the other transaction cannot have access to it. Hence in the table, it is shown that read-read can have shared lock represented by 'S' and other operations such as read-write, write-write cannot have shared lock and can have only exclusive lock.

Lets take an example of how lock protocol works to understand in much better way. Consider two transactions T1 and T2. say the data item is A. Initially lets assume that T1 wants to have access to A, therefore T1 acquires exclusive lock on data item, it reads the value and writes as well then releases the lock, say T2 just wants to read the value then it acquires the shared lock and reads the value and then releases lock.

There are 4 types of lock protocols available-

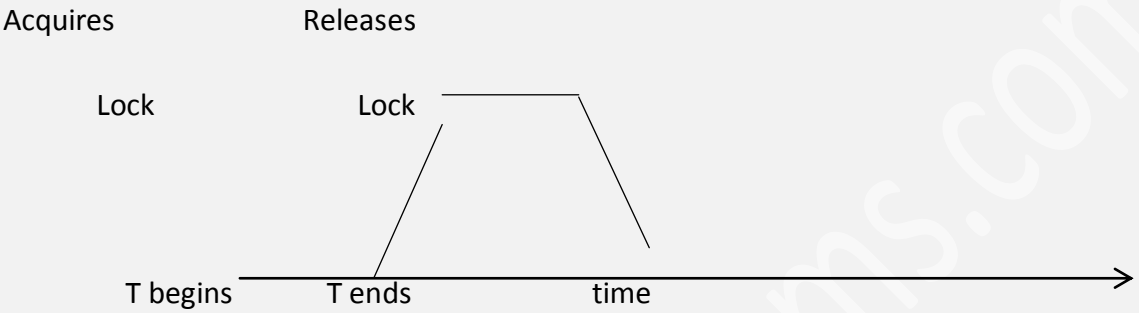
1. Simple Lock Protocol - As the name itself implies it is the simple way of locking and unlocking as and when required. This protocol supports the transaction to acquire the lock on the data in order to modify or update and after the transaction is finished, the protocol will unlock it.

2. Pre- Claiming Protocol - This protocol works on requests to the system made by the transaction. Initially a list of data items which have to be accessed is created, all the data items in that particular list thus needs the lock on them. The transaction makes a request to the system for the locks. Transaction proceeds if the locks have been created else it just has the option to get back and wait and for the locks.

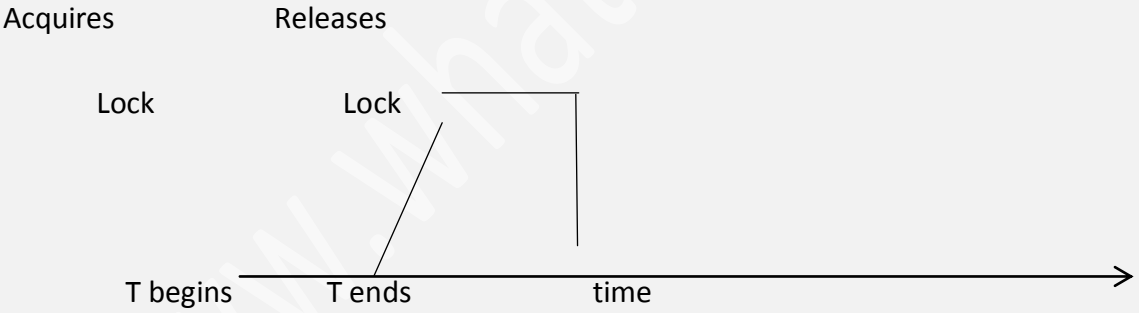
3. Two Phase Locking Protocol (2PL) - This protocol is all about growing and shrinking of locks. The phases are called as growing and shrinking as all the locks are being acquired and released respectively.

The execution takes place in 3 steps:

- ✓ As the transaction takes place, it keeps requesting or seeking the permission for the lock it needs.
- ✓ Transaction acquires all the locks.
- ✓ Transaction releases the locks , this phase starts when it releases its first lock.



4. Strict Two Phase Protocol - It is similar to two phase locking protocol till the acquires all locks, it differs in the fact it does not release the locks at a time, and holds the locks until reaches the commit point.



TIME STAMP BASED PROTOCOL:

This protocol is used very commonly. Time stamp is used to link time with some event or in more particular say transaction. To ensure serializability, we associate transaction with the time called as time stamp. In simple words we order the transaction based on the time of arrival and there is no deadlock.

The algorithm that we implement here should make sure that the conflicting data item or operations are in the order such that they do not cause the violation.

For each data item, two time stamp are maintained.

1. Read time stamp - time stamp of youngest transaction which has performed operation read on the data item.
2. Write time stamp - time stamp of youngest transaction which has performed operation write on the data item.

Let the transaction T's time-stamp be denoted by TS(T), Read time-stamp of data-item be denoted by R-timestamp(X), and Write time-stamp of data-item be denoted by W-timestamp(X).

The protocol works as follows-

If a transaction issues read operation

- If Ts(T) < W-timestamp(X) then

  read request is rejected

  else execute the transaction and update the time-stamp.

If a transaction operates write operation

- If Ts(T) < R-timestamp(X) or If TS(T) <W-timestamp(X) then

  write request is rejected

  else transaction gets executed and update the time-stamp.

So these were the protocols to control concurrency in DBMS. Hope you understood it well.