



SNS COLLEGE OF TECHNOLOGY

(An Autonomous Institution)

Approved by AICTE, New Delhi, Affiliated to Anna University, Chennai

Accredited by NAAC-UGC with 'A++' Grade (Cycle III) &

Accredited by NBA (B.E - CSE, EEE, ECE, Mech & B.Tech.IT)

COIMBATORE-641 035, TAMIL NADU



UNIT IV - Transaction

Transaction Concepts – ACID Properties – Schedules – Serializability – Concurrency Control – Need for Concurrency – **Locking Protocols** – Two Phase Locking – Deadlock – Transaction Recovery – Save Points – Isolation Levels – SQL Facilities for Concurrency and Recovery.

Locking Protocols

What is a Lock?

A lock is a variable associated with a data item that indicates whether it is currently in use or available for other operations. Locks are essential for managing access to data during concurrent transactions. When one transaction is accessing or modifying a data item, a lock ensures that other transactions cannot interfere with it, maintaining data integrity and preventing conflicts. This process, known as locking, is a widely used method to ensure smooth and consistent operation in database systems.

Lock Based Protocols

Lock-Based Protocols in DBMS ensure that a transaction cannot read or write data until it gets the necessary lock. Here's how they work:

- These protocols prevent concurrency issues by allowing only one transaction to access a specific data item at a time.
- Locks help multiple transactions work together smoothly by managing access to the database items.
- Locking is a common method used to maintain the serializability of transactions.
- A transaction must acquire a read lock or write lock on a data item before performing any read or write operations on it.

Types of Lock

1. Shared Lock (S): Shared Lock is also known as Read-only lock. As the name suggests it can be shared between transactions because while holding this lock the transaction does not have the permission to update data on the data item. S-lock is requested using lock-S instruction.

2. Exclusive Lock (X): Data item can be both read as well as written. This is Exclusive and cannot be held simultaneously on the same data item. X-lock is requested using lock-X instruction.

Rules of Locking

The basic rules for Locking are given below:

Read Lock (or) Shared Lock(S)

- If a Transaction has a Read lock on a data item, it can read the item but not update it.
- If a transaction has a Read lock on the data item, other transaction can obtain Read Lock on the data item but no Write Locks.
- So, the Read Lock is also called a Shared Lock.

Write Lock (or) Exclusive Lock (X)

- If a transaction has a write Lock on a data item, it can both read and update the data item.
- If a transaction has a write Lock on the data item, then other transactions cannot obtain either a Read lock or write lock on the data item.
- So, the Write Lock is also known as Exclusive Lock.

Lock Compatibility Matrix

- A transaction can acquire a lock on a data item only if the requested lock is compatible with existing locks held by other transactions.
- Shared Locks (S): Multiple transactions can hold shared locks on the same data item simultaneously.
- Exclusive Lock (X): If a transaction holds an exclusive lock on a data item, no other transaction can hold any type of lock on that item.
- If a requested lock is not compatible, the requesting transaction must wait until all incompatible locks are released by other transactions.
- Once the incompatible locks are released, the requested lock is granted.

	S	X
S	✓	✗
X	✗	✗

Concurrency Control Protocols

Concurrency Control Protocols are the methods used to manage multiple transactions happening at the same time. They ensure that transactions are executed safely without interfering with each other, maintaining the accuracy and consistency of the database. These protocols prevent issues like data conflicts, lost updates or inconsistent data by controlling how transactions access and modify data.

Types of Lock-Based Protocols

1. Simplistic Lock Protocol

It is the simplest method for locking data during a transaction. Simple lock-based protocols enable all transactions to obtain a lock on the data before inserting, deleting, or updating it. It will unlock the data item once the transaction is completed.

Example:

Consider a database with a single data item $X = 10$.

Transactions:

- T1: Wants to read and update X.
- T2: Wants to read X.

Steps:

1. T1 requests an exclusive lock on X to update its value. The lock is granted.
 - T1 reads $X = 10$ and updates it to $X = 20$.
2. T2 requests a shared lock on X to read its value. Since T1 is holding an exclusive lock, T2 must wait.
3. T1 completes its operation and releases the lock.
4. T2 now gets the shared lock and reads the updated value $X = 20$.

This example shows how simplistic lock protocols handle concurrency but do not prevent problems like deadlocks or limits concurrency.

2. Pre-Claiming Lock Protocol

The Pre-Claiming Lock Protocol evaluates a transaction to identify all the data items that require locks. Before the transaction begins, it requests the database management system to grant locks on all necessary data elements. If all the requested locks are successfully acquired, the transaction proceeds. Once the transaction is completed, all locks are released. However, if any of the locks are unavailable, the transaction rolls back and waits until all required locks are granted before restarting.

Example:

Consider two transactions T1 and T2 and two data items, X and Y:

1. Transaction T1 declares that it needs:

- A write lock on X.
- A read lock on Y.

Since both locks are available, the system grants them. T1 starts execution:

- It updates X.
- It reads the value of Y.

2. While T1 is executing, Transaction T2 declares that it needs:

- A read lock on X.

However, since T1 already holds a write lock on X, T2's request is denied. T2 must wait until T1 completes its operations and releases the locks.

3. Once T1 finishes, it releases the locks on X and Y. The system now grants the read lock on X to T2, allowing it to proceed.

This method is simple but may lead to inefficiency in systems with a high number of transactions.

3. Two-phase locking (2PL)

A transaction is said to follow the Two-Phase Locking protocol if Locking and Unlocking can be done in two phases :

- Growing Phase: New locks on data items may be acquired but none can be released.
- Shrinking Phase: Existing locks may be released but no new locks can be acquired.

4. Strict Two-Phase Locking Protocol

Strict Two-Phase Locking requires that in addition to the 2-PL all Exclusive(X) locks held by the transaction be released until after the Transaction Commits.

Problem With Simple Locking: Partial Schedule

S.No	T1	T2
1	lock-X(B)	
2	read(B)	
3	B:=B-50	
4	write(B)	
5		lock-S(A)

S.No	T1	T2
6		read(A)
7		lock-S(B)
8	lock-X(A)	
9

1. Deadlock

In the given execution scenario, T1 holds an exclusive lock on B, while T2 holds a shared lock on A. At Statement 7, T2 requests a lock on B, and at Statement 8, T1 requests a lock on A. This situation creates a deadlock, as both transactions are waiting for resources held by the other, preventing either from proceeding with their execution.

2. Starvation

Starvation is also possible if concurrency control manager is badly designed. For example: A transaction may be waiting for an X-lock on an item, while a sequence of other transactions request and are granted an S-lock on the same item. This may be avoided if the concurrency control manager is properly designed.