SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

COURSE NAME : 19SB504 DATABASE MANAGEMENT SYSTEMS

III YEAR / V SEMESTER

Unit IV- **TRANSACTIONS MANAGEMENT**

Topic : **SERIALIZABILITY & CONCURRENCY CONTROL, LOCKING PROTOCOLS – TWO PHASE LOCKING**

SERIALIZABILITY

Serializability in a Database Management System (DBMS) is a **critical concept** related to the **execution of multiple transactions in a concurrent environment.**

It ensures that **concurrent execution of transactions** does not lead to data **anomalies or inconsistencies** in the database.

A schedule is considered serializable if its execution is equivalent to a serial execution of the same transactions, meaning that the final state of the database **remains consistent.**

SERIALIZABILITY

Serializability guarantees two key properties

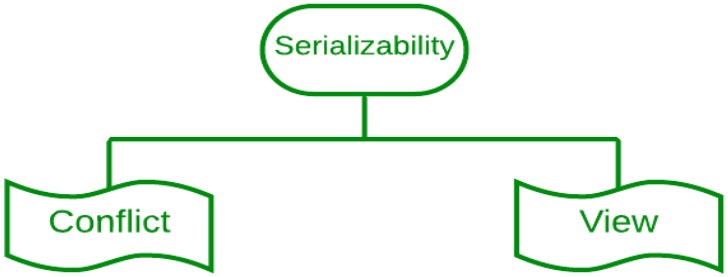
Isolation: Each transaction appears to execute in isolation, as if **no other transactions were running concurrently.**

Consistency: The database **starts in a consistent state**, and it **ends in a consistent state after the execution** of the transactions.

- ✓ To ensure serializability, DBMSs **use concurrency control mechanisms**, such as **locking, timestamp-based methods, and two-phase locking.**
- ✓ These mechanisms prevent conflicts between transactions by controlling access to data items.

SERIALIZABILITY

There are two commonly used techniques for ensuring serializability:



Conflict Serializability:

- ✓ This technique **examines whether there are any conflicts between transactions.**
- ✓ A conflict occurs when two transactions access the same data item, and at least one of them is a write operation (update or delete).

SERIALIZABILITY

View Serializability:

- ✓ View serializability is a broader concept that considers **the overall effect of transactions on the database.**
- ✓ A schedule is view-serializable if it can be **transformed into a serial schedule without changing the read-write relationships.**
- ✓ Both techniques aim to **guarantee serializability** and maintain the **integrity and consistency** of the database in a **multi-user, concurrent environment**.

CONCURRENCY CONTROL

CONCURRENCY CONTROL

Concurrent Execution in DBMS

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 executed simultaneously on a multi-user system by different users.

CONCURRENCY CONTROL

Problems with Concurrent Execution

The two main operations are **READ and WRITE** operations.

So, there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, and the data may become inconsistent.

So, the following problems occur with the Concurrent Execution of the operations:

CONCURRENCY CONTROL

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

Example

Consider the below diagram where two transactions TX and TY, are performed on the same account A where the balance of account A is $300.

CONCURRENCY CONTROL

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

CONCURRENCY CONTROL

- ✓ At time t1, transaction TX reads the value of account A, i.e., $300 (only read).
- ✓ At time t2, transaction TX deducts $50 from account A that becomes $250 (only deducted and not updated/write).
- ✓ Alternately, at time t3, transaction TY reads the value of account A that will be $300 only because TX didn't update the value yet.
- ✓ At time t4, transaction TY adds $100 to account A that becomes $400 (only added but not updated/write).

CONCURRENCY CONTROL

- ✓ At time t6, transaction TX writes the value of account A that will be updated as $250 only, as TY didn't update the value yet.
- ✓ Similarly, at time t7, transaction TY writes the values of account A, so it will write as done at time t4 that will be $400. It means the value written by TX is lost, i.e., $250 is lost.
- ✓ Hence data becomes incorrect, and database sets to inconsistent.

CONCURRENCY CONTROL

Concurrently control

- ✓ Concurrently control is a very important concept of DBMS which **ensures the simultaneous execution** or manipulation of data by several processes or user without resulting in data inconsistency.

- ✓ Concurrency Control **deals with interleaved execution of more than one transaction.**

- ✓ **maintaining the concurrency of the database, we have the concurrency control protocols.**

CONCURRENCY CONTROL

Concurrency Control Protocols

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

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

Lock Based Concurrency Control Protocol

Lock-based concurrency control protocols are commonly used in database management systems (DBMS) to **manage concurrent access to data**.

These protocols use locks to **ensure that transactions can access and modify data** in a **controlled and consistent** manner.

There are two primary types of locks:

1. **read locks** (shared locks)
2. **write locks** (exclusive locks).

Read Lock (Shared Lock): When a transaction acquires a read lock on a data item, it indicates that it intends to read the data item. Multiple transactions can acquire read locks on the same data item simultaneously, **allowing for concurrent reading but preventing concurrent writing.**

Example: Library Book Reading

Write Lock (Exclusive Lock): When a transaction acquires a write lock on a data item, it indicates that it intends to modify (write) the data item. Only one transaction can have a write lock on a data item at a time. Write locks are exclusive and block other transactions from acquiring read or write locks on the same data item.

Example: Conference Room Reservation System

Two Phase Locking

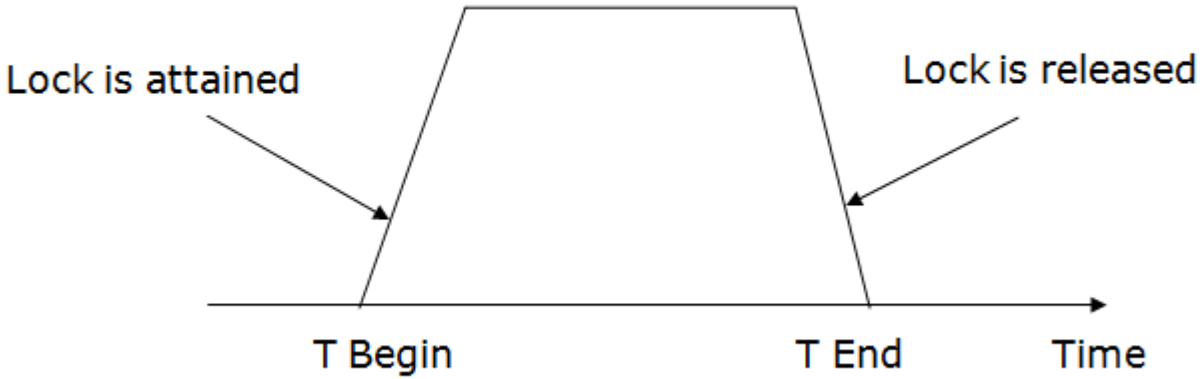
The primary goal of 2PL is to **prevent conflicts and anomalies** that can arise from concurrent access to shared data while ensuring that the transactions are executed in a manner that is equivalent to some serial (sequential) order.

- ✓ 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**.



Example

Consider an online shopping application **where multiple customers** can place orders, and the application needs to maintain consistent inventory levels.

Customer A's Order:

- Customer A selects three items to purchase: Item X, Item Y, and Item Z.
- The application's database uses the **Two-Phase Locking protocol.**
- Customer A's transaction begins.

- **Growing Phase:**
  - Customer A's transaction acquires a lock on Item X, indicating that it wants to buy this item.
  - Customer A's transaction acquires a lock on Item Y.
  - Customer A's transaction acquires a lock on Item Z.
- **Shrinking Phase:**
  - Customer A's transaction releases the lock on Item Y, indicating that the purchase of Item Y is complete.
  - Customer A's transaction cannot acquire any more locks (it cannot add more items to the cart).

Customer B's Order:

Customer B is shopping simultaneously with Customer A.

Growing Phase:

- ✓ Customer B's transaction wants to buy Item Y (the same item Customer A already has in their cart).
- ✓ Customer B's transaction requests a lock on Item Y, but it is denied since Customer A currently holds the lock for Item Y.
- ✓ Customer B's transaction is blocked until Customer A completes the purchase of Item Y and releases the lock.

Customer A Completes the Order:

Customer A proceeds to checkout and completes the purchase of all items (X, Y, and Z).

Customer A's transaction releases the locks on Items X, Y, and Z.

Customer B's Order Resumes:

Now that Customer A has released the lock on Item Y, Customer B's transaction can acquire the lock on Item Y.

Customer B can proceed with their purchase.

Thank you .....