Unit – 6

Transaction Processing, Concurrency Control and Recovery Techniques

Transaction in DBMS

- Transactions refer to a set of operations that are used for performing a set of logical work possibly update various data items.
- A transaction includes one or more database access operations – these can include insertion, deletion, modification or retrieval operations.
- Usually, a transaction means the data present in the DB has changed.

Transaction Management

Transaction Management refers to the tasks of processing multiple transactions issued by various clients of a database server in such a way that the ACID contract can be fulfilled, that is, the properties of atomicity, consistency preservation, isolation, and durability of each individual transaction can be guaranteed.

Transaction Prcoessing

- Transaction processing is the computerized system that handles and records transactions, ensuring they are completed reliably and accurately.

Transaction Processing System

- A transaction processing system allows application programmers to concentrate on writing code that supports the business, by shielding application programs from the details of transaction management.
- It manages the concurrent processing of transactions.
- It enables the sharing of data. It ensures the integrity of data.

Simple Transaction Example

1. Read your account balance
2. Deduct the amount from your balance .
3. Write the remaining balance to your account
4. Read your friend’s account balance
5. Add the amount to his account balance
6. Write the new updated balance to his account .

This whole set of operations can be called a transaction.

In DBMS, we write the above 6 steps transaction like this:
Let's say your account is A and your friend's account is B, you are transferring 10000 from A to B, the steps of the transaction are:

1. **R(A);**
2. **A = A - 10000;**
3. **W(A);**
4. **R(B);**
5. **B = B + 10000;**
6. **W(B);**

In the above transaction R refers to the Read operation and W refers to the write operation.

Desirable Properties of Transaction : ACID

- To ensure the integrity of data during a transaction, the database system should possess several properties, often called ACID properties.
- ACID stands for Atomicity Consistency Isolation and Durability.

Atomicity:

- A transaction is an atomic unit of processing. It is either performed in its entirely or not performed at all.
- It is the responsibility of the transaction recovery subsystem of a DBMS.
- If a transaction fails to complete for some reason, such as system crash in the midst of transaction execution, the recovery technique must undo any effects of the transaction on the database.

ACID Properties:

Consistency preservation:

- A correct execution of transaction must take the database from one consistent state to another.
- The preservation of consistency is generally considered to be the responsibility of the programmers who write the database programs or of the DBMS module that enforces integrity constraints.
- A consistent state of the database satisfies the constraints specified in the schema as well as any constraints on the database that should hold.

ACID Properties:

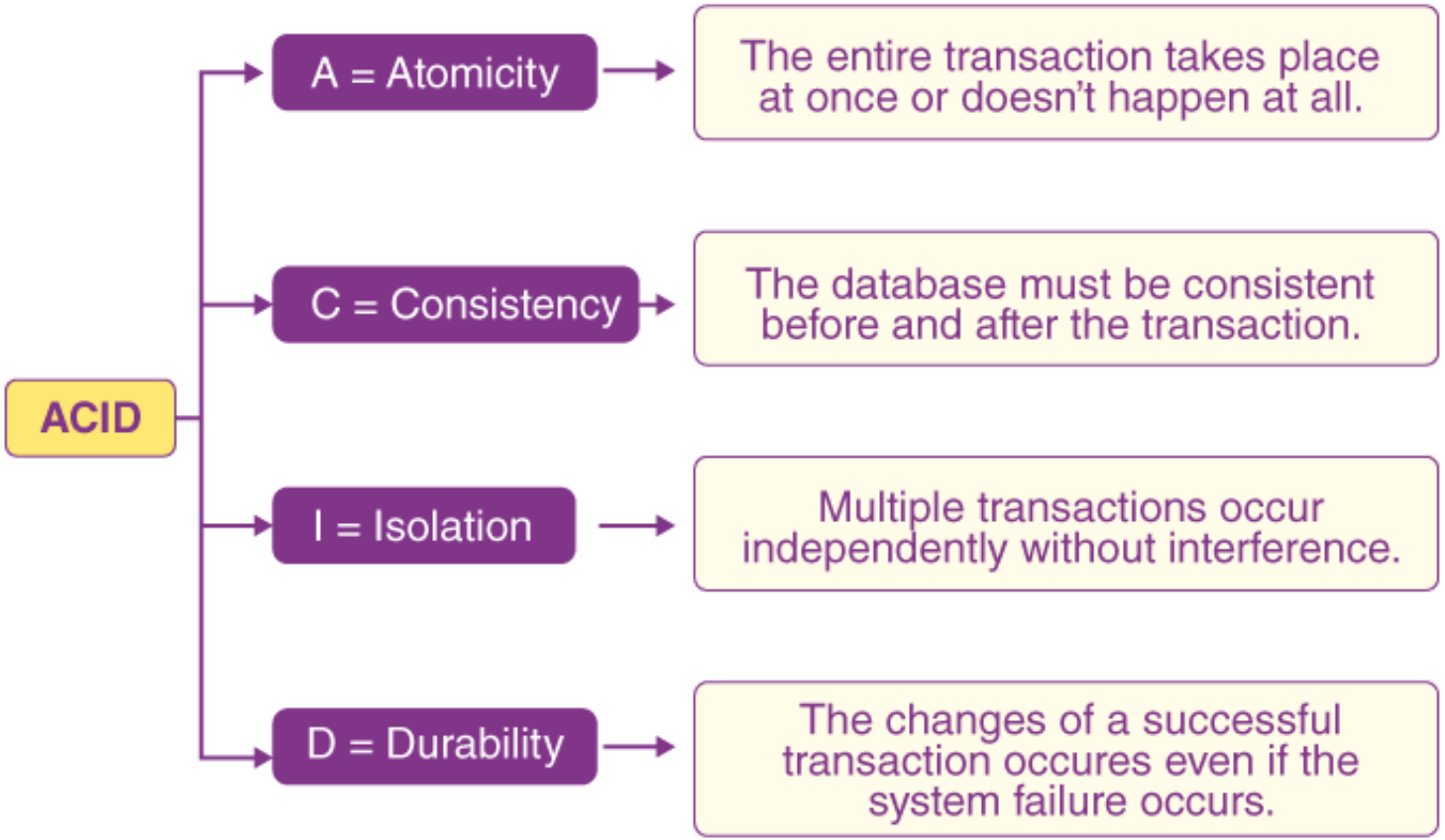
Isolation:

- A transaction should not make its update visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transaction unnecessary.
- It is forced by the concurrency control sub-system of the DBMS.

Durability or permanency:

- Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.
- It is the responsibility of the recovery sub-system of the DBMS.

ACID Properties in DBMS



Implementation of Atomticity and Durability

- The simplest way to enforce **Atomicity** is for the DBMS to refuse to start any transaction until the previous one has committed. Unfortunately, this can be too restrictive, especially if the transaction needs to interact with a user. While one user is dithering, several other users could be served.

- **Durability** can be achieved by flushing the transaction's log records to non-volatile storage before acknowledging commitment. In distributed transactions, all participating servers must coordinate before commit can be acknowledged. This is usually done by a two-phase commit protocol.

Transaction States:

- States through which a transaction goes during its lifetime.
- These are the states which tell about the current state of the Transaction and also tell how we will further do the processing in the transactions. These states govern the rules which decide the fate of the transaction whether it will commit or abort.
- They also use **Transaction log.**
- Transaction log is a file maintain by recovery management component to record all the activities of the transaction. After commit is done transaction log file is removed.

These are different types of Transaction States :

1. Active State
When the instructions of the transaction are running then the transaction is in active state. If all the ‘read and write’ operations are performed without any error then it goes to the “partially committed state”; if any instruction fails, it goes to the “failed state”.

2. Partially Committed
After completion of all the read and write operation the changes are made in main memory or local buffer. If the changes are made permanent on the database, then the state will change to “committed state” and in case of failure it will go to the “failed state”.

3. Failed State
When any instruction of the transaction fails, it goes to the “failed state” or if failure occurs in making a permanent change of data on database.

4. Aborted State

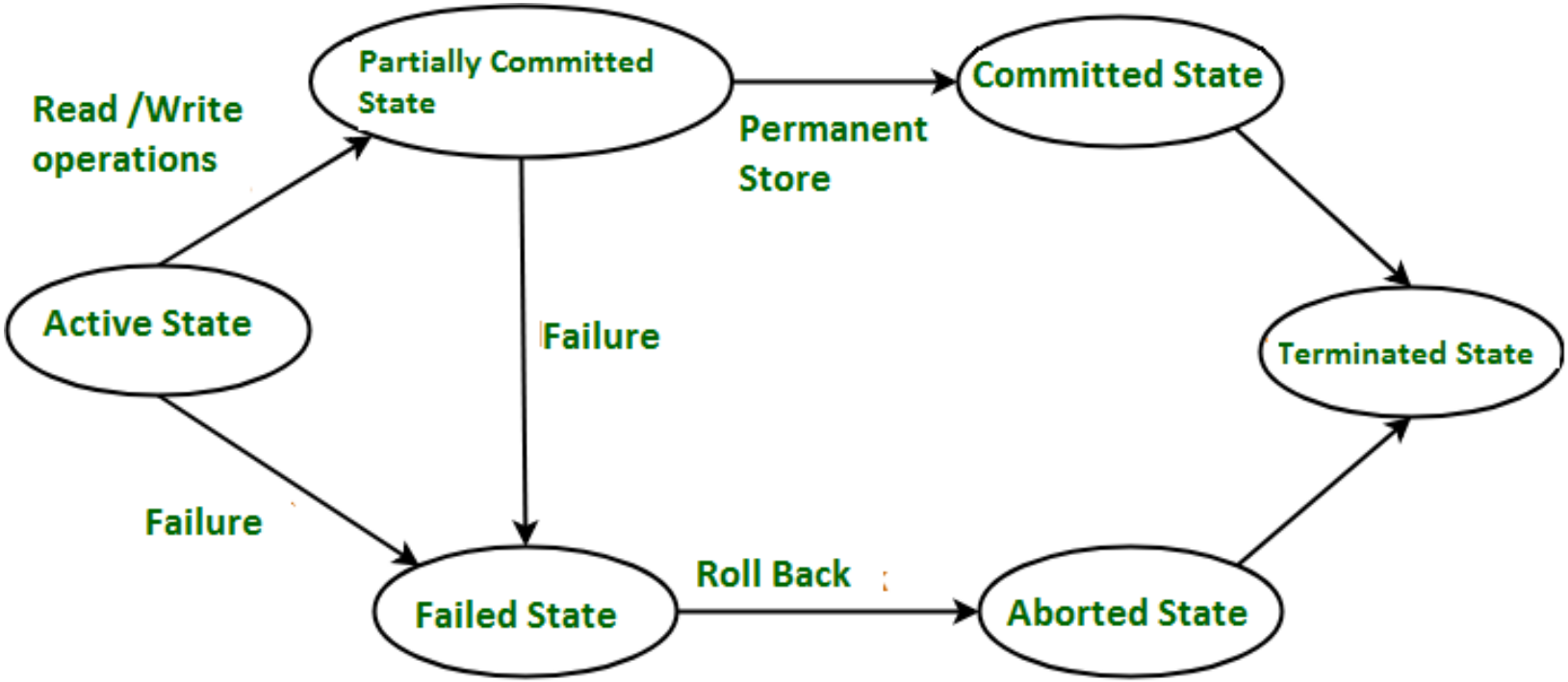
After having any type of failure, the transaction goes from “failed state” to “aborted state” and since in previous states, the changes are only made to local buffer or main memory and hence these changes are deleted or rolled-back.

5. Committed State

It is the state when the changes are made permanent on the Data Base and the transaction is complete and therefore terminated in the “terminated state”.

6. Terminated State

If there isn’t any roll-back or the transaction comes from the “committed state”, then the system is consistent and ready for new transaction and the old transaction is terminated.

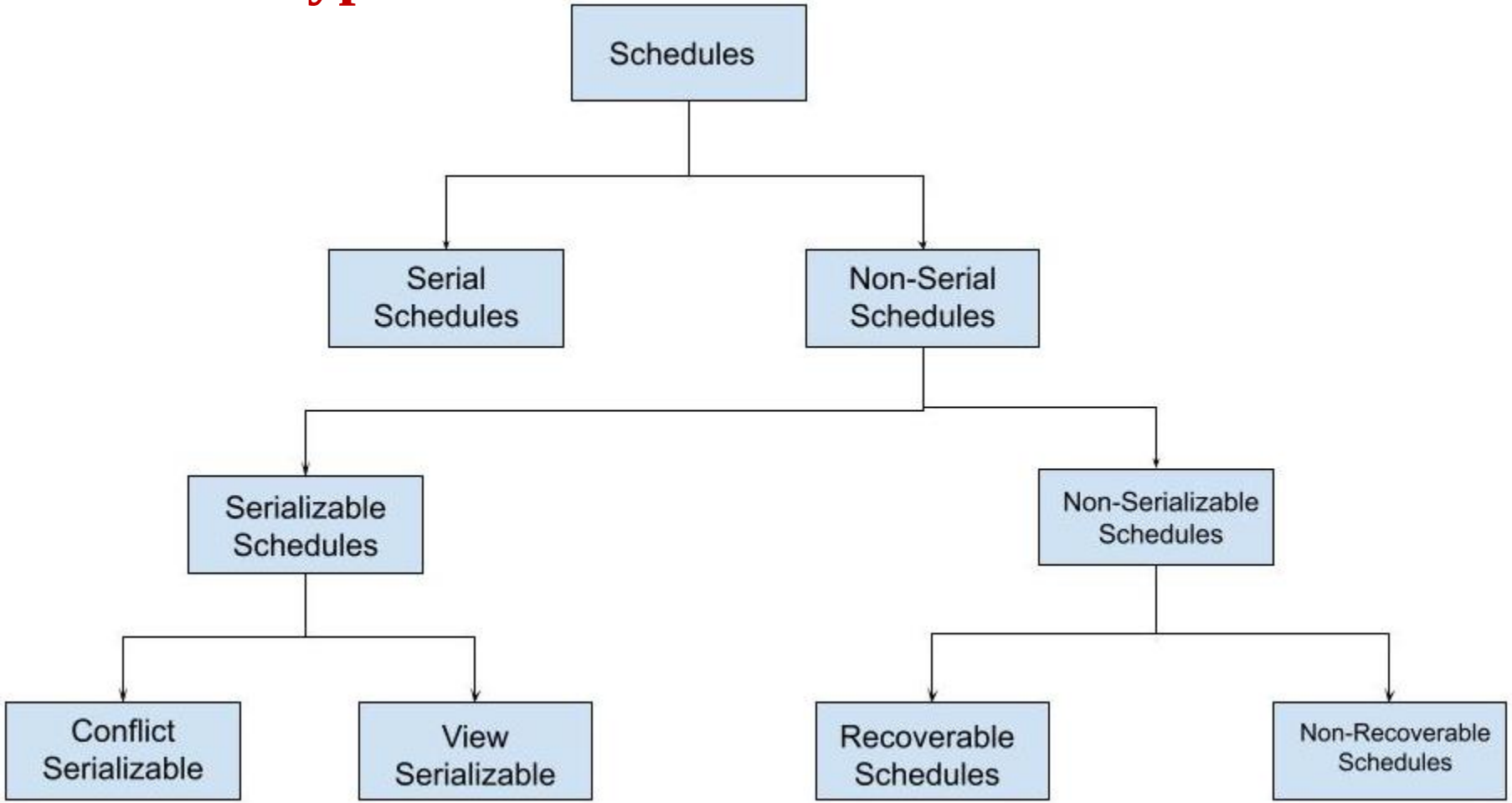


Transaction States in DBMS

Schedule

- Schedule is a process of lining the transactions and executing them one by one.
- When there are multiple transactions that are running in a concurrent manner and the order of operation is needed to be set so that the operations do not overlap each other, Scheduling is brought into play and the transactions are timed accordingly.

Schedule Types



Serializability

- Serializability is the concept in a transaction that helps to identify which non-serial schedule is correct and will maintain the database consistency. It relates to the isolation property of transaction in the database.
- Serializability is the concurrency scheme where the execution of concurrent transactions is equivalent to the transactions which execute serially.
- It refers to the sequence of actions such as read, write, abort, commit are performed in a serial manner.

Serializable Schedule

- Serializable schedule is a sequence of database actions (read and write operations) that does not violate the serializability property.
- Serializability property ensures that each transaction appears to execute atomically and is isolated from other transactions' effects.

Basic Concept of Concurrency Control

- Concurrency Control:
  - Manages simultaneous access to a database.
  - It prevents two users from editing the same record at the same time and also serializes transactions for backup and recovery.

What is Concurrency Control?

- Data concurrency is the ability to allow multiple users to affect multiple transaction within a database.
- **Concurrency Control** in Database Management System is a procedure of managing simultaneous operations without conflicting with each other.
- Concurrent access is quite easy if all users are just reading data. There is no way they can interfere with one another.
- Though for any practical Database, it would have a mix of READ and WRITE operations and hence the concurrency is a challenge.

Need for Concurrency Control

- It ensures that Database transactions are performed concurrently and accurately to produce correct results without violating data integrity of the respective Database especially where two or more database transactions are executed simultaneously, which require access to the same data..
- To apply Isolation through mutual exclusion between conflicting transactions
- To resolve read-write and write-write conflict issues
- To preserve database consistency through constantly preserving execution obstructions.
- Concurrency control helps to ensure serializability
- The system needs to control the interaction among the concurrent transactions. This control is achieved using concurrent-control schemes.

Concurrency Control Protocols

Different concurrency control protocols offer different benefits between the amount of concurrency they allow and the amount of overhead that they impose. Following are the most common Concurrency Control techniques in DBMS:

- **Lock-Based Protocols**
- **Two Phase Locking Protocol**
- **Timestamp-Based Protocols**

Locking Protocols

- Lock Based Protocols in DBMS is a mechanism in which a transaction cannot Read or Write the data until it acquires an appropriate lock.
- This helps in eliminating the concurrency problem by locking a particular transaction to a particular user.

Transaction T1	
LOCK S (A)	// Shared Lock on data "A"
R (A)	// Read Operation on data "A"
Unlock (A)	// Unlock data "A"

	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	——	——

Time Stamp Based Protocol

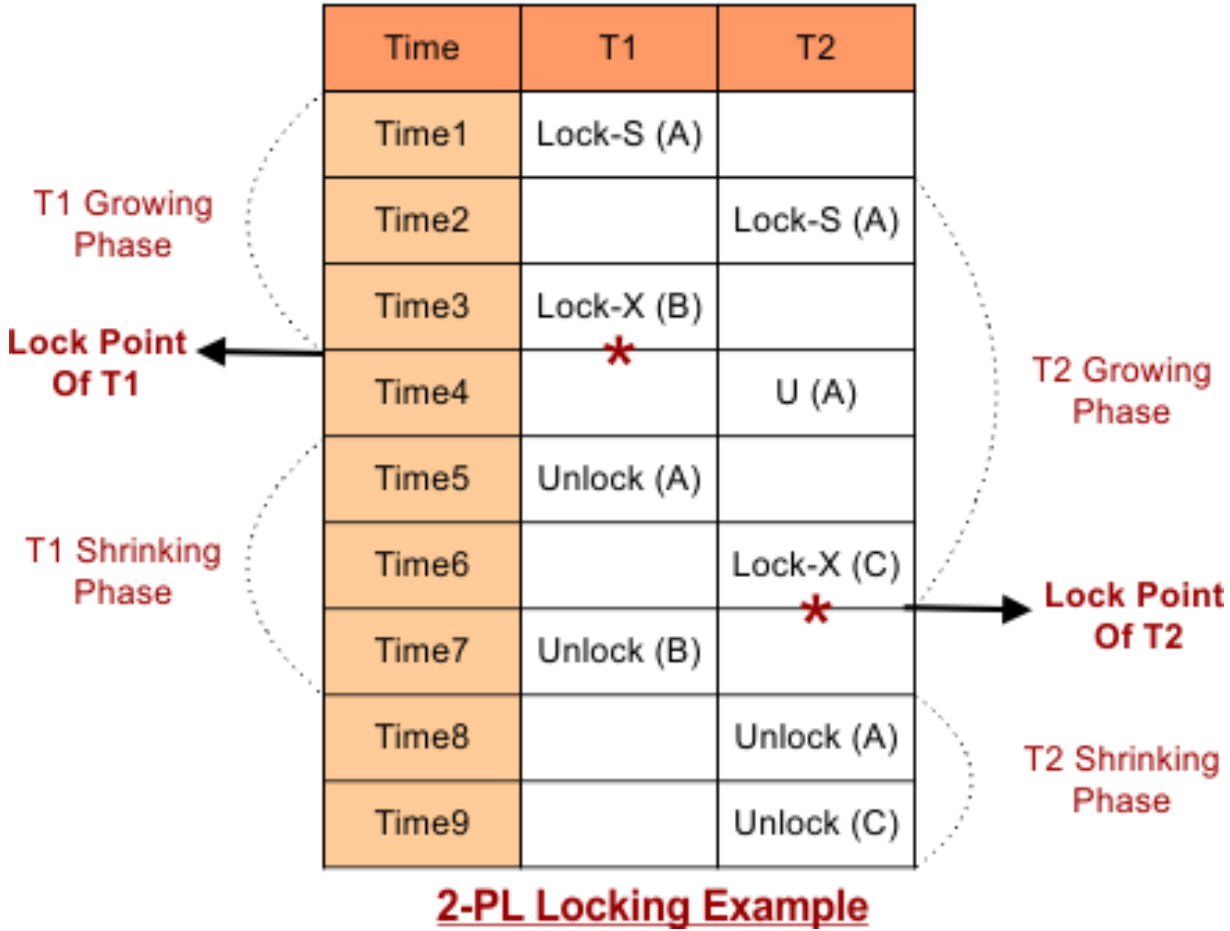
- Timestamp based Protocol in DBMS is an algorithm which uses the System Time or Logical Counter as a timestamp to serialize the execution of concurrent transactions.
- The Timestamp-based protocol ensures that every conflicting read and write operations are executed in a timestamp order.
- The order of transaction is nothing but the ascending order of the transaction creation.
- The priority of the older transaction is higher that's why it executes first.

Two Phase Locking Protocol

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

  1. Growing Phase − All the locks are issued in this phase. No locks are released, after all changes to data-items are committed and then the second phase (shrinking phase) starts.

  2. Shrinking phase − No locks are issued in this phase, all the changes to data-items are noted (stored) and then locks are released.
- **Lock Point :** In the growing phase transaction reaches a point where all the locks it may need has been acquired. This point is called LOCK POINT. After the lock point has been reached, the transaction enters a shrinking phase.



2-PL Locking Example

Types of failures

- **Catastrophic failure**
  - Catastrophic failure is a type of database failure that is severe and often results in the loss of a significant amount of data.
  - In these cases, it may not be possible to recover the database using traditional methods, such as restoring from a backup or repairing the database.
  - The only guaranteed method of recovering from catastrophic failure is through database backup
  - Examples:
    - ✓ Complete system outage
    - ✓ Natural disasters
    - ✓ Security breaches
    - ✓ Nuclear accidents

Types of failures

- **Non-Catastrophic**
  - It refers to a failure or error in a system that does not lead to a complete and irreversible loss of functionality or data.
  - Non-catastrophic failures are typically less severe and may be recoverable with minimal impact on the overall system.
  - **Examples:**
    - ✓ Temporary network interruptions or slowdowns.
    - ✓ Software bugs that cause certain features to malfunction but do not crash the entire system.
    - ✓ Disk errors that result in the corruption of some data.
    - ✓ Hardware malfunctions.
    - ✓ Resource exhaustion issues, such as running out of memory or disk space, which can impact performance but are not necessarily catastrophic.

Recovery Concepts

- Recovery refers to the set of techniques and processes used to restore a database to a correct, consistent state after a failure.
- Failures can range from system crashes, power outages, hardware malfunctions, to software errors, and even human-induced errors.
- The goal of recovery is to ensure data integrity and consistency by undoing the effects of incomplete transactions and bringing back the database to a state that existed before the failure occurred.
- The aim is to minimize data loss, correct errors, and ensure that the database can be used normally after an unexpected incident.
- Some methods by which we can recover database from failure state are:
- Log Based Recovery
- Caching (Buffering) o Disk blocks
- Write-Ahead Logging
- Checkpointing

NO-UNDO/ REDO Recovery Based on Deferred Update

- Deferred Upadte is a technique for the maintenance of the transaction log files of the DBMS.
- It is also called NO-UNDO/REDO technique.
- It is used for the recovery of transaction failures that occur due to power, memory, or OS failures.
- The log file contains all the changes that are to be applied to the database.
- The changes are not applied immediately to the database.
- In this method of recovery, firstly the changes carried out by a transaction on the data are done in the log file and then applied to the database on commit.

Recovery Technique Based on Immediate Update

- It is a technique in which whenever any transaction is executed, the updates are made directly to the database and the log file is also maintained which contains both old and new values. Once the commit is done, all the changes get stored permanently in the database, and records in the log file are thus discarded.
- The log file contains both old as well as new values.
- Concept of shadow paging is used in immediate update method.

Shadow Paging

- Shadow Paging is a technique which involves maintaining two page tables during the transaction process: the current page table and the shadow page table.
- The shadow page table remains unchanged throughout the transaction and acts as a backup, while the current page table is updated.
- If a transaction fails, the system can revert to the shadow page table, ensuring data consistency.

Database Backup from Catastrophic Failures.

Database backup is a crucial aspect of disaster recovery planning. The following methods can help for database backup :

- **Regular Backups:** Implement a regular backup schedule to ensure that critical data is backed up at frequent intervals.
- **Offsite Storage:** Store backup copies of the database in offsite locations or on geographically distributed servers.
- **Encryption:** Encrypt backup data to protect it from unauthorized access or tampering.
- **Versioning:** Maintain multiple versions of backup copies to manage the lifecycle of backup data.

Database Recovery from Catastrophic Failures.

Recovering a database from catastrophic failures involves restoring the database to a consistent and usable state.
the steps involved in database recovery from catastrophic failures:

- **Assessment of Damage**: After a catastrophic failure, identify the cause of the failure and any potential points of failure in the system.
- **Identify Recovery Point**: Determine the recovery point, which is the point in time to which the database needs to be restored.
- **Restore from Backup**: If backups are available, restore the database from the most recent backup to the desired recovery point.
- **Data Validation:** Validate the restored data to ensure its accuracy and completeness.

End!