Unit-IV

Transaction Management:

Transaction refers to a collection of operations that form a single logical unit of work.

Transaction is represented with the symbol - T_x, where x represents the TransactionId.

T_x	SQL
—	SQL> Savepoint S_1;
—	— —
—	— — —
—	— — —
—	— — —
End of T_x	SQL> Commit;
	(OR)
	Rollback to S_1;

Example: If a bank has to transfer Rs.1,000 from account A to account B, then the following steps are required before the transfer is successful.

(i) Reduce the balance of the account A by the amount of Rs.1,000.

(ii) Increase the balance of the account B by the amount of Rs.1,000.

If any one / both of the statements fail, the transaction would not be successful.

Properties of Transaction: **Imp**

Every database system has 4 Transaction properties

1) Atomicity
2) Consistency
3) Isolation
4) Durability

} ACID properties

Atomicity: This property states that a transaction must be treated as an atomic unit, i.e, either all of its operations are executed or none.

There are no transactions which are left partially completed.

Consistency: Database should be consistent before and after the execution of the transaction. No transaction should have any adverse effect on the data residing in the database.

Isolation: Multiple transactions execute concurrently. Database system must provide a mechanism to isolate transactions from the effects of other concurrently executing transactions.

Durability: Once the transaction is successful, the changes made to the database persist (store permanently), even if there are system failures.

States of Transaction:

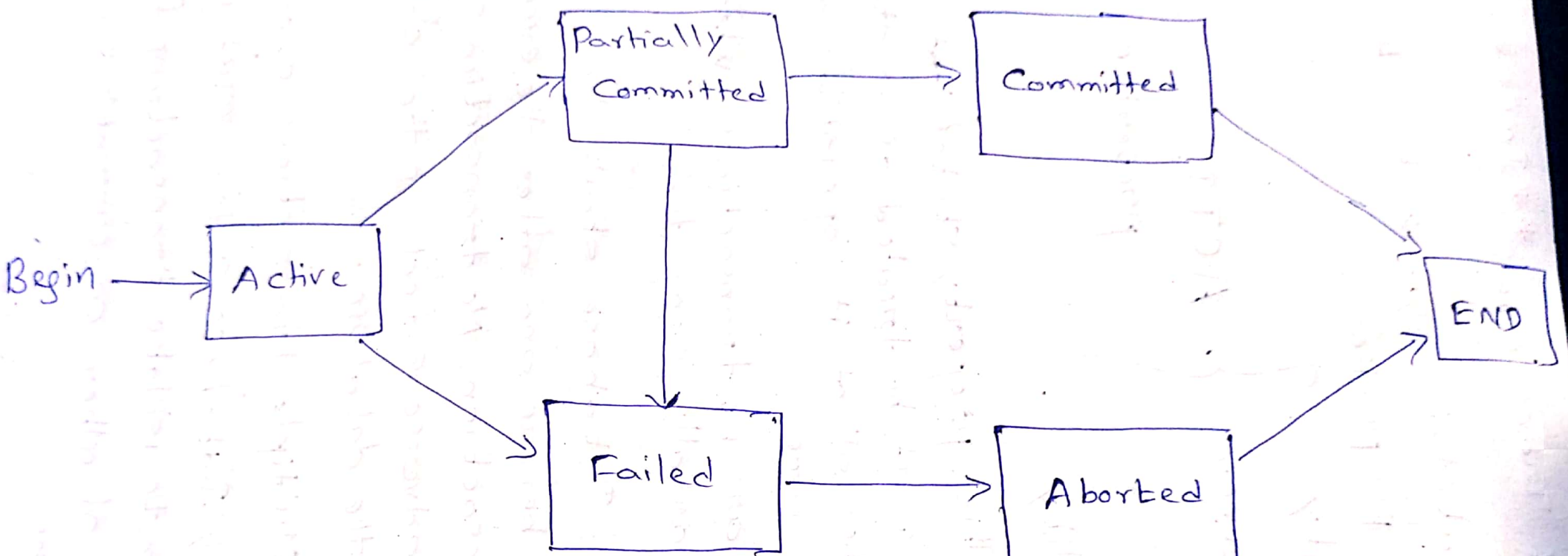
Committed: A transaction that completes its execution successfully is called Committed Transaction

Roll back: A transaction which is not completed successfully must be Rolled back (undone the changes, if any)

Abort: A transaction may not always complete its execution successfully. This state is called Aborted.

Failed: A transaction is said to be in failed state after the system determining that the transaction can no longer proceed with its normal execution because of some hardware or software error. Such Transaction must be rolled back.

Partially Committed: In a partially committed state, a transaction executes its final



Concurrency Control:

Concurrency is the process of allowing multiple transactions to run on the same database. There may be several complications with consistency of the data while executing concurrent transactions. The DB must control the interaction among the concurrent transactions to prevent them from destroying the consistency of the DB.

Concurrency control is the process of regulating access to the same data by multiple transactions operating in the same DB environment.

Need for Concurrency Control:

Execution of concurrent transactions over a shared database can create several data integrity and consistency problems like

- Lost Updates
- Uncommitted Data
- Inconsistent retrievals

Concurrency Control Schemes:

1) **Lock Based Protocol:** A lock is a mechanism that tells DBMS whether data item is being used by any transaction for read/write purpose.

Read operation performed by different transactions on the same data item value which can read by any number of transactions at any given time.

Write operation performed by a transaction makes the data value remains in an inconsistent state till the completion of transaction. any other transaction with read/write will get inconsistent data during the execution of above said write operation.

Types of Locks: Locks are of two types:

(i) **Shared Lock:** A transaction which acquires shared lock on a data item allows data value to perform read operation. Minimum two transactions on same data item are required to get Shared lock.

(ii) **Exclusive Lock:** A transaction may acquire exclusive lock on a data item in order to perform both read/write operations. This lock is exclusive for one transaction on a data item, i.e, no other transactions are allowed on the same data item.

Every locking mechanism requires Binary locks i.e, Lock and Unlock.

- Locked objects (data items) are unavailable to other objects.
- Unlocked objects are open to any transaction.

Ex:

Shared Lock

```
Lock-S(T1);
Read T1;
Unlock(T1);
```

T_2 is also allowed

Exclusive Lock

```
Lock-X(T1);
Read T1;
Update T1;
Unlock(T1)
```

write — only T_1 is allowed

```
Lock-X(T2)
Read T2;
Update T2;
Unlock(T2)
```

only T_2 is allowed.

11 Two-Phase Locking (2PL): 2PL

A transaction is said to be Two-phase locked if

- Before reading x, it sets a read lock on T
- Before writing x, it sets a write lock on T
- It holds each lock until after it executes the corresponding operation
- After its first unlock operation, it requests no new locks.

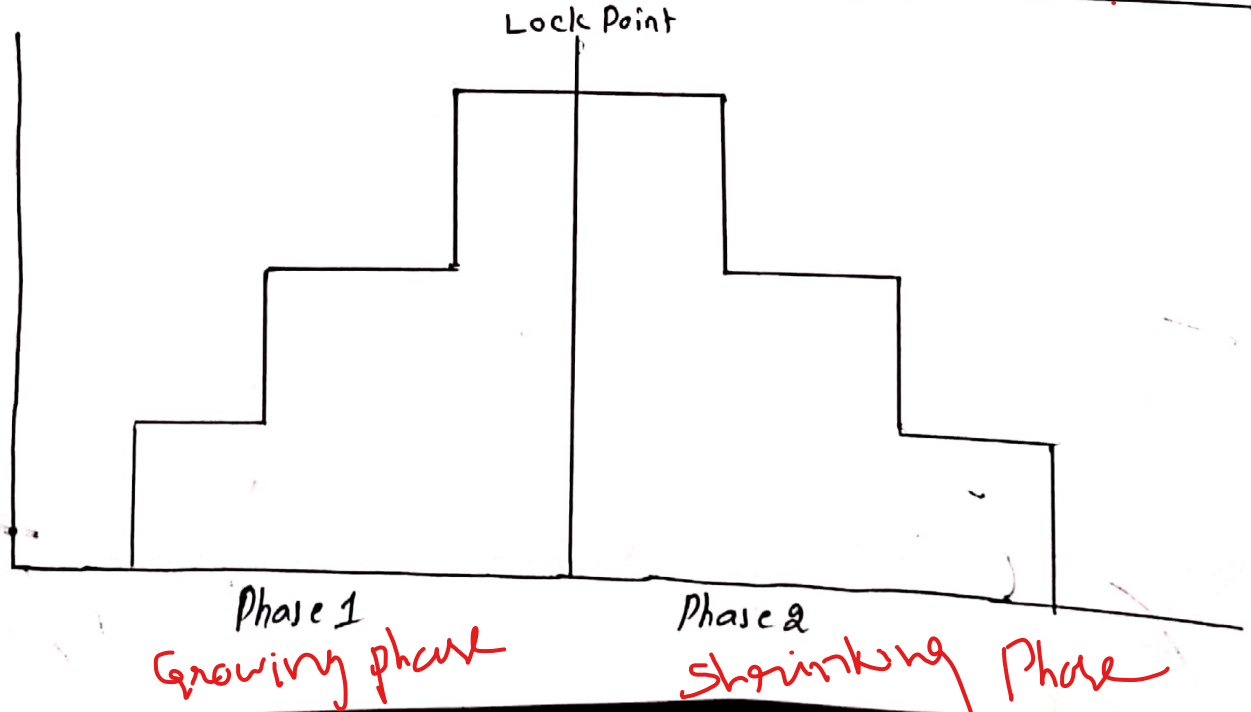
Phase 1: (Growing Phase) ✓

In this phase, we put read or write lock based on need of the data.

In this phase, we don't release any lock.

Phase 2: (Shrinking Phase): ✓

This phase is just reverse of growing phase. In this phase, we release read and write locks, but doesn't put any lock on data.



10:00
9:59
9:58
0:00

(ii) Timestamp Based Protocol:

Timestamp is a system clock value which is assigned to a transaction. The implementation concepts are: Object Created timestamp, ✓
Lastmodified timestamp, ✓
object session out timestamp etc.

A timestamp can be implemented in two ways.

(i) Write-Stamp(T): The last system clock value when there was a data ~~updation~~ & updation/write operation

(ii) Read-Stamp(T): The last system clock value when there was a data retrieval/write/select operation

iv. Optimistic Approach: ✓

Majority of the Databases implement optimistic Approach by default (i.e, There is no requirement of writing an algorithm to implement this concept). This approach neither requires Locking system or Time-stamping to implement.

Every DB optimistic approach requires three concepts

a) Read: Transactions read the data, execute the statements ✓
b) write: Transactions modifies the data and makes them available to other transaction to perform read/write. ✓
c) Validate: Transaction validates the data, (i.e, DB changes shouldn't affect Integrity and consistency of data). ✓

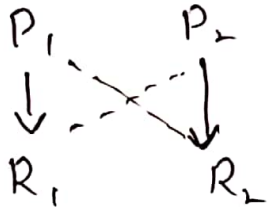
Deadlock

A system is in a deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set.

Deadlock refers to a particular situation where two or more processes are each waiting for another to release a resource or more than two processes are waiting for resources in a circular chain.

Deadlock is a common problem in multiprocessing environment.

Ex:



P_1 is holding Resource R_1 and waiting for R_2 and P_2 is holding Resource R_2 and waiting for R_1. One of them (either P_1 or P_2) has to release the resource, otherwise the situation is "Deadlock".

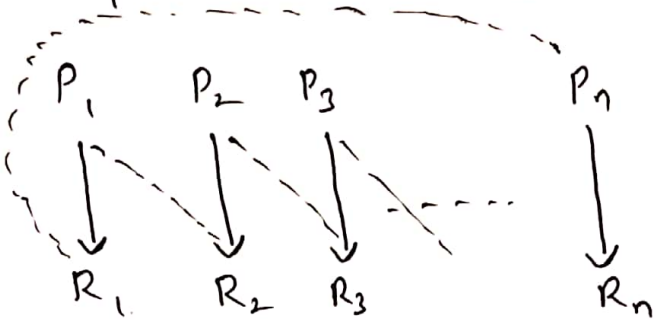
There are four conditions of Deadlock

(i) Mutual Exclusion: There should be atleast one resource which is not used or one resource shared by more than one process at a time.

(ii) Hold and Wait: A process 'P_1' holding a resource and waiting for another resource held by another process P_2.

(iii) No-Preemption: A resource cannot be forcibly taken from a process. Only the process can release a resource that is being held by it.

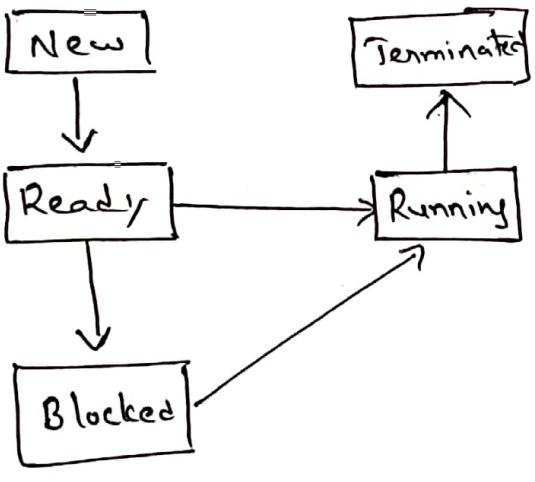
(iv) Circular wait: A condition where Process P_1 in waiting for the resource held by P_2 and P_2 is waiting for the resource held by P_3 and so on and the last process is waiting for the first process.



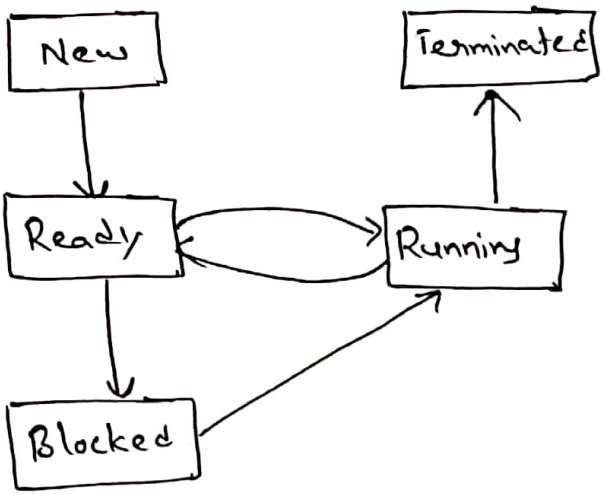
To avoid a deadlock, the above mentioned necessary conditions shouldn't occur.

Note: The DBMS inspects the operations and analyzes if they can create a deadlock situation. If it finds that a deadlock situation might occur, then that transaction is never allowed to be executed.

Non-Preemption



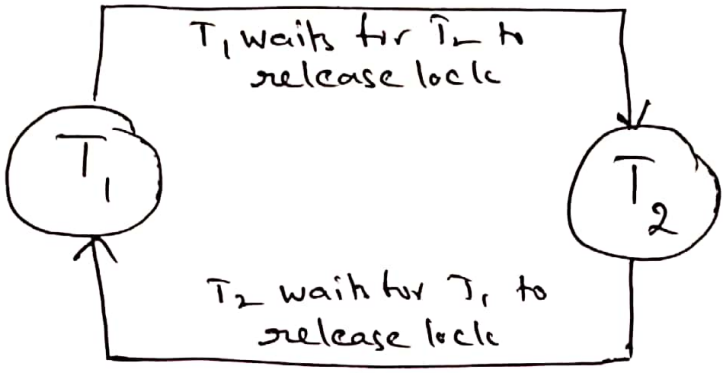
Preemption



A process going from Running State to Ready state is Preemption.

Deadlock Detection and Recovery:

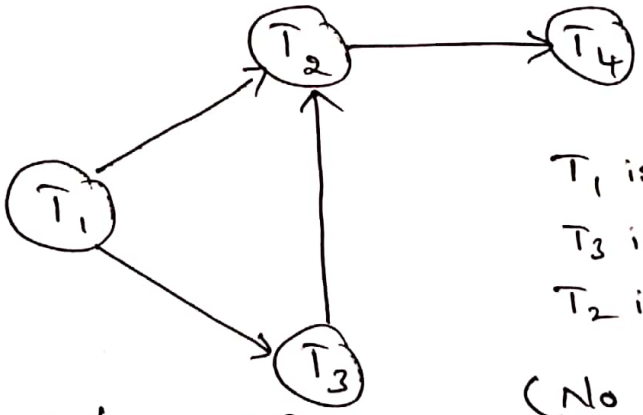
Wait-for-Graph is the best method for detecting the deadlock situation.



(Follows Directed Graph)

A deadlock occurs in a system if there exists a cycle in the wait for graph and each transaction involved in the cycle is said to be in a deadlock state.

Example: (1)

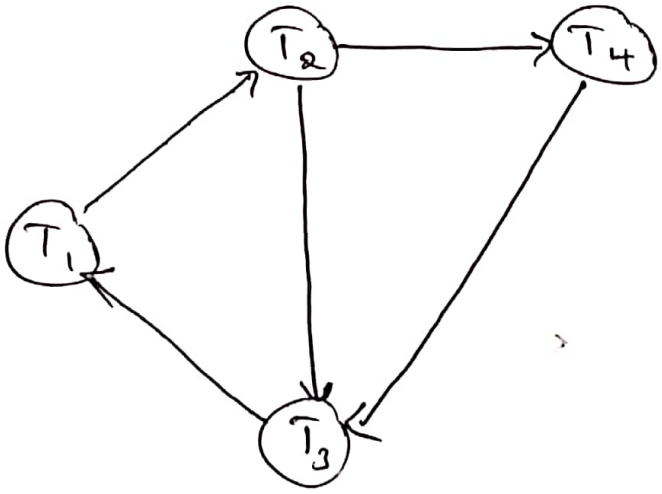


T_1 is waiting for T_2 & T_3
T_3 is waiting for T_2
T_2 is waiting for T_4

(No Deadlock)

There is no cycle in Ex(1), Hence No Deadlock occurs.

(2)



$T_1 \rightarrow T_2 \rightarrow T_4 \rightarrow T_3 \rightarrow T_1$

$T_1 \rightarrow T_2 \rightarrow T_3$

(Deadlock occurs)

There is a cycle in Ex(2), Hence Deadlock occurs.

Recovery: Rollback one or more transactions until the system shows no more deadlock situation. Rollback the transactions which will incur minimum cost.

There are three steps to Recover

(i) Selection of victim:

The transaction which has to be rolled back is called Victim.

(ii) Rollback: Simple solution is Total Rollback. The another solution is Partial Rollback which rollbacks only specified transaction.

(iii) Starvation: Starvation is the situation when a transaction has to wait for a indefinite period of time to acquire a lock.

Starvation is the situation caused because of lack of resources.

Database Recovery

Introduction: A computer system is an electromechanical device subject to failures of various types. The reliability of the DB system is linked to the reliability of the computer system on which it runs.

The Types of failures that the computer system is likely to be subjected to include failures of components or subsystems, software failures, power outage, accidents, unforeseen situations and natural or manmade disasters. DB recovery techniques are the methods of making the DB fault tolerant. The aim of the recovery scheme is to allow database operations to be resumed after a failure with minimum loss of information at an economically justifiable cost.

1. **Need for Recovery:** The process of Recovery is needed to handle the problems of DB failures.

The following are some of the failures occur in a DB system:

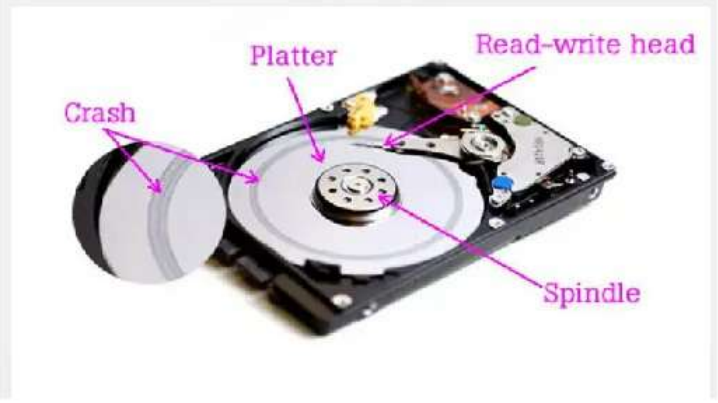
(i) **Transaction Failure:** There are two types of errors that may cause a transaction to fail:

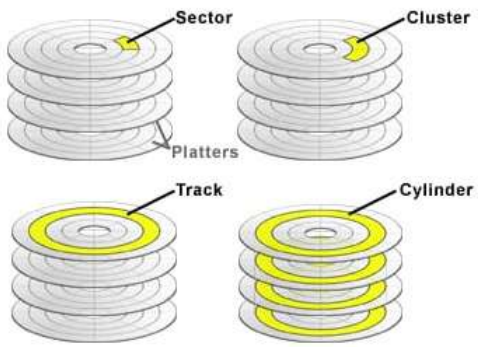
a) **Logical Error:** The transaction can no longer continue with its normal execution because of some internal condition such as incorrect input, Division by zero, integer overflow etc. Certain exceptional conditions that are not programmed correctly may also result in transaction cancellation.

b) System Error: The system has entered an undesirable state (Eg. Deadlock). As a result of this undesirable state, the transaction cannot continue with its normal execution. This type of transaction can be re-executed at a later time.

(ii) System Crash: Any hardware malfunction or a bug in the database software or the operating system that causes the loss of the content of volatile storage (RAM) and brings transaction processing to a halt. The contents of non-volatile storage (Hard disk) is not corrupted.

(iii) Disk Failure: A disk block loses its content as a result of either a head crash or failure during a data transfer operation. These errors may occur during read/write operations. Copies of the data on other disks or archival backups on storage media.





(iv) Physical Problems and Natural Disasters:

Problems like Data theft, Fire accident, overwriting of secodary storage comes under Physical Problems where as Problems like Earthquake, Floods, Tsunami comes under Natural Disasters. Both may result in the Loss of Data.

4) Transactions and Recovery: A transaction is the basic unit of recovery in DB system. The software which deals with DB recovery is Recovery Manager. The recovery manager has to ensure that on recovery from failure, all the effects of a given transaction are permanently recorded in the DB or none of them are recorded.

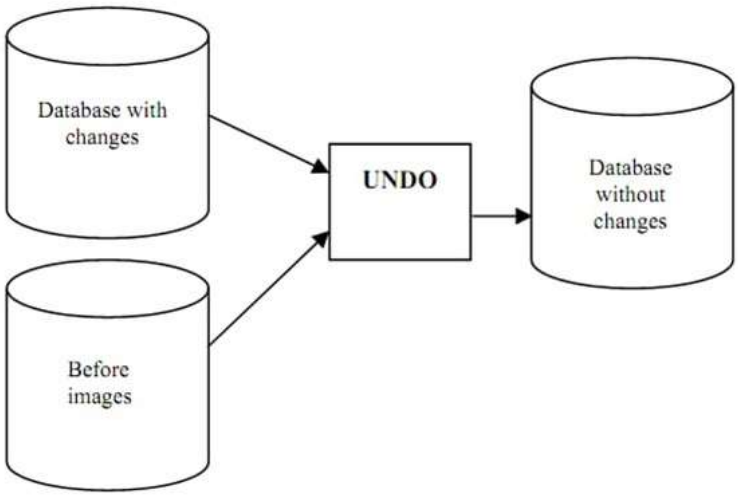
A transaction always begin with a successful execution of BEGIN TRANSACTION statement and it ends with successful execution of COMMIT statement.

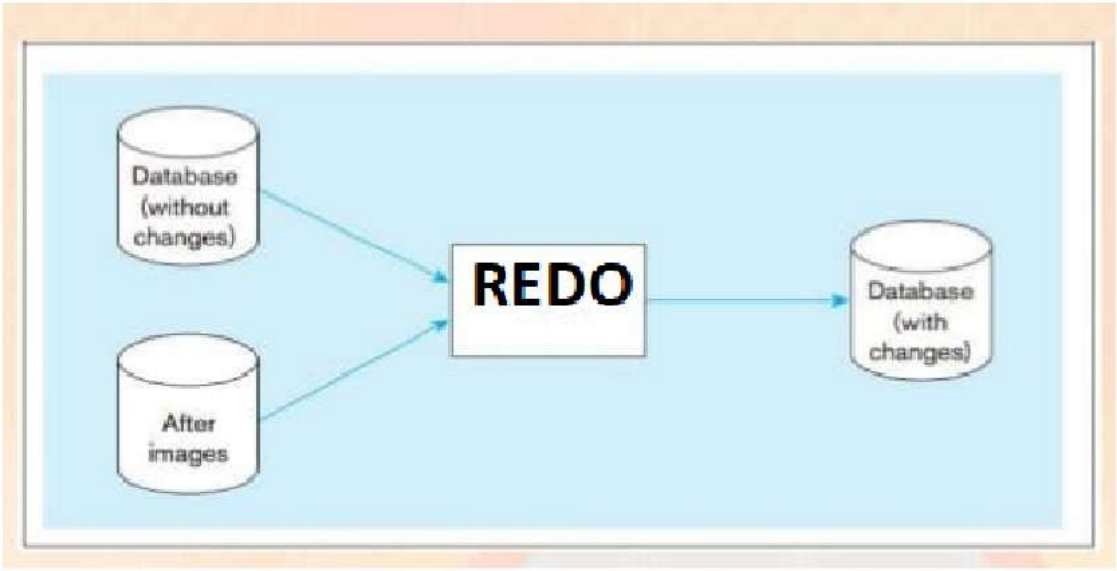
There are two types of Transaction Recovery

(i) Forward Recovery
(ii) Backward Recovery

Forward Recovery (REDO): Forward Recovery is the recovery procedure which is used in case of physical damage (Eg. Failure of Secondary storage while writing of data to DB).

Backward Recovery (UNDO): Backward Recovery is a recovery procedure which is used in the case of an error occurred in the middle of normal execution of a transaction. The recovery manager must undo the transaction when an error occurred in the middle of execution of a transaction.





III. DBMS Recovery Facilities:

A DBMS should provide the following facilities:

(i) Backup facilities: Every DBMS must backup periodic copies of the DB.

(ii) Logging facilities: It helps in keeping track of the current state of Transaction and DB changes

(iii) Checkpoint facilities: It enables updates to the database that are in progress to be made permanent.

(iv) Recovery Manager: It allows the system to restore the DB to a consistent state in case of failure.

IV. Recovery Techniques:

Recovery Techniques can be categorized into three types:

(i) Log-Based Recovery: The most widely used structure for recording database modifications is the Log. The Log is a sequence of Log Records. (Log Record maintains all the update activities in the DB).

An update log contains the following fields:

a) Transaction identifier: It is the unique identifier of the transaction that performed the write operation

b) Data-item identifier: It is the unique identifier of the data item written. Typically, It is the location on disk of the data item.

(iii) old value: It is the value of the data item prior to the write

(iv) New value: It is the value of the data item will have after the write.

→ The log based recovery technique is further classified into Two types:

1. Deferred update: The deferred update ensures transaction atomicity by recording all database modifications in the log, but deferring the execution of all write operations of a transaction until the transaction is successfully completed. Data changes recorded in the log file are applied to the database on commit. If a transaction fails before reaching its commit point, there is no need to undo any operation because the transaction has not affected the database on disk in any way.

(2) Immediate Update: The immediate update allows database modifications to be output to the database while the transaction is still in the active state. i.e, when a transaction issues an update command the database can be updated immediately without any need to wait for the transaction to reach its commit point. On commit, all changes made to the DB are made permanent and the records in the log file are discarded. On rollback, old values are restored into the database using the old values stored in the log file and all the changes made by the transaction to the DB are discarded.

(ii) Checkpoints:

This technique is used to solve the problem by upcoming the difficulties in Log-Based Recovery technique.

The two major difficulties with Log-Based approach:

a) The search process is Time Consuming.

b) Most of the transactions need to be redone have already written their updates into the DB. Although redoing them will cause no harm, it will nevertheless cause recovery to take longer.

A Checkpoint is a snapshot or copy of DB at a particular moment in time. Typically, a checkpoint involves recording a certain amount of information so that if a failure occurs, the database server can restart at that established point.

A checkpoint operation is performed periodically by the system as follows:

. A start of checkpoint record giving the identification, that is, checkpoint along with the date and time of the checkpoint.

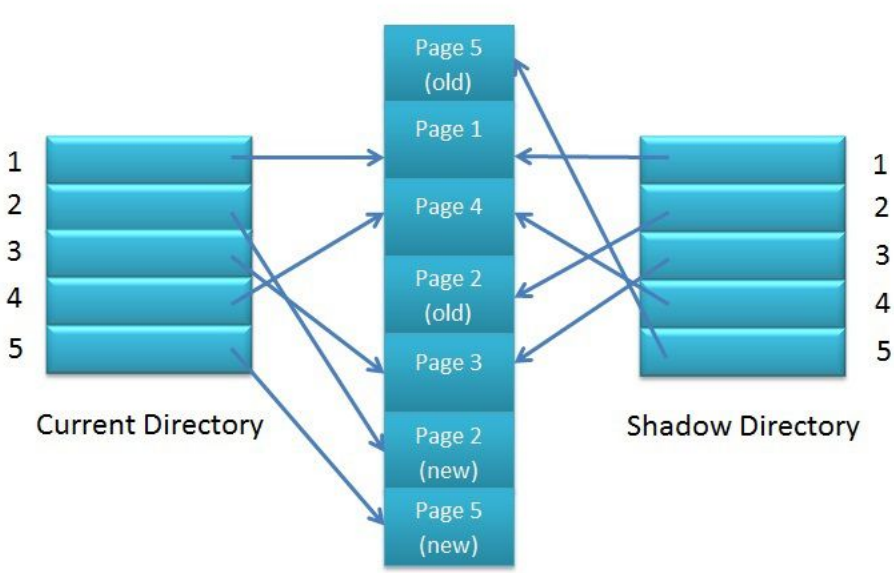
. All log information from the buffers in the volatile storage is copied to the log on the stable storage.

. All database updates from the buffers in the volabile storage are written to the disk.

(iii) **Shadow Paging:** The shadow paging technique is different from the log-based recovery techniques in a way that it doesn't require the use of log in an environment where only one transaction is active at a time. The log is mandatory where multiple concurrent transactions are executing.

Shadow paging is based on making copies of the database. In this recovery scheme, the database is partitioned into fixed-length blocks referred as PAGES. These pages are mapped into physical blocks of storage with the help of page table called directory. These pages do not need to be stored in any particular order on disk. The main idea behind this technique is to maintain two page tables:

(i) Current page table or Current Directory
(ii) Shadow page table or Shadow Directory.



Unit IV

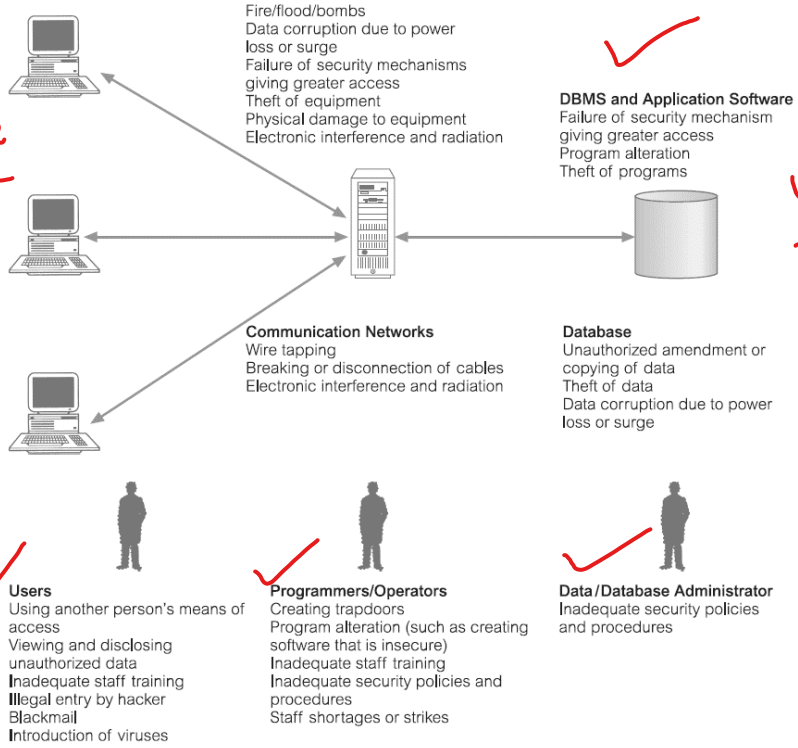
Database Security

Introduction: The mechanism that protect the database against intentional or accidental threats is called Database Security.

Database security is applied in the following situations:

- Theft and fraud - affect not only the database environment but also the entire organization.
- Loss of confidentiality - refers to the need to maintain secrecy over data.
- Loss of privacy - the need to protect data about individuals.
- Loss of integrity - results in invalid or corrupted data
- Loss of availability - 24/7 availability (that is, 24 hours a day, 7 days a week) is required.

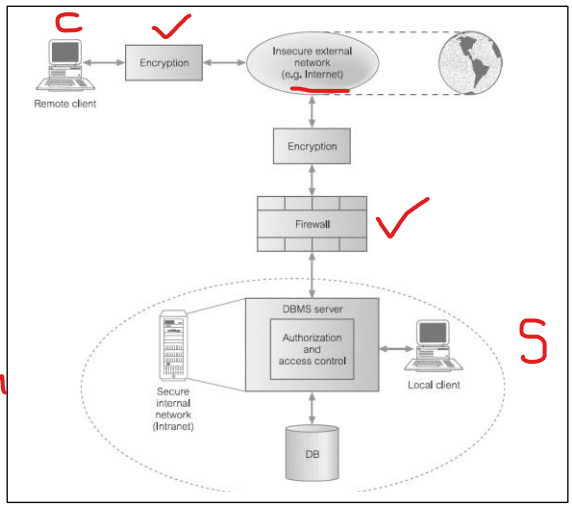
Threat: Any situation or event, whether intentional or accidental, that may adversely affect a system and consequently the organization. A threat may be caused by a situation or event involving a person, action, or circumstance that is likely to bring harm to an organization. The problem facing any organization is to identify all possible threats. Therefore, as a minimum an organization should invest time and effort in identifying the most serious threats. Intentional threats involve people and may be perpetrated by both authorized users and unauthorized users, some of whom may be external to the organization. A summary of the potential threats to computer systems is represented in the following diagram.



Computer Based Controls:

Computer-based security controls for a multi-user environment (Diagram) are as follows:

- Authorization
- Access Controls
- Views
- Backup And Recovery
- Integrity
- Encryption
- Raid Technology



1. **Authorization and Authentication:**

The granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object is called **Authorization**. Authorization controls can be built into the software, and govern not only what system or object a specified user can access, but also what the user may do with it. The process of authorization involves authentication of user/program requesting access to Database objects

A mechanism that determines whether a user (he/she) claims to be part of Database system is called **Authentication**. A system administrator is usually responsible for allowing users to have access to a computer system by creating individual user accounts. Each user is given a unique identifier, which is used by the operating system to determine who they are. Associated with each identifier is a password, chosen by the user and known to the operating system, which must be supplied to enable the operating system to verify (or authenticate) who the user claims to be.

2. **Access Controls:**

Access Control is a privilege which allows a user to create or access (i.e., read, write, or modify) some database object (such as a relation, view, or index) or to run certain DBMS utilities. As excessive granting of unnecessary privileges can compromise security. A privilege should only be granted to a user if that user cannot accomplish his or her work without that privilege. The DBMS subsequently keeps track of how these privileges are granted to other users, and possibly revoked, and ensures that at all times only users with necessary privileges can access an object.

Some of the Access Controls are:

Discretionary Access Control (DAC): Most commercial DBMSs provide an approach to managing privileges that uses SQL called Discretionary Access Control. The SQL standard supports DAC through the GRANT and REVOKE commands. The GRANT command gives privileges to users, and the REVOKE command takes away privileges.

Mandatory Access Control (**MAC**): It is based on system-wide policies that cannot be changed by individual users. In this approach each database object is assigned a *security class* and each user is assigned a *clearance* for a security class, and *rules* are imposed on reading and writing of database objects by users. The DBMS determines whether a given user can read or write a given object based on certain rules that involve the security level of the object and the clearance of the user. These rules seek to ensure that sensitive data can never be passed on to another user without the necessary clearance. The SQL standard does not include support for MAC.

3. **Views:**

A view is the dynamic result of one or more relational operations operating on the base relations to produce another relation. A view is a *virtual relation/table* that does not actually exist in the database, but is produced upon request by a particular user, at the time of request. The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users. The user is not aware of the existence of any attributes or rows that are missing from the view. A view can be defined over several relations with a user being granted the appropriate privilege to use it, but not to use the base relations

4. **Backup and Recovery:**

The process of periodically taking a copy of the database and log files on to offline storage media is called **Backup**. DBMS should provide backup facilities to assist with the recovery of a database following failure. It is always advisable to make backup copies of the database and log files at regular intervals and to ensure that the copies are in a secure location. In the event of a failure that renders the database unusable, the backup copy and the details captured in the log file are used to restore the database to the latest possible consistent state.

The process of keeping and maintaining a log file (or journal) of all changes made to the database to enable recovery to be undertaken effectively in the event of a failure is called **Journaling**. A DBMS should provide logging facilities, sometimes referred to as journaling, which keep track of the current state of transactions and database changes, to provide support for recovery procedures. The advantage of journaling is that, in the event of a failure, the database can be recovered to its last known consistent state using a backup copy of the database and the information contained in the log file. If no journaling is enabled on a failed system, the only means of recovery is to restore the database using the latest backup version of the database.

5. **Integrity:**

Integrity constraints also contribute to maintaining a secure database system by preventing data from becoming invalid, and hence giving misleading or incorrect results. The two principal rules for the relational model are known as entity integrity (Primary key) and referential integrity (Foreign Key).

6. **Encryption and Decryption:**

The encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key is called **Encryption**. The conversion of encrypted data into its original form is called **Decryption**.

To transmit data securely over insecure networks requires the use of a **cryptosystem**, which includes:

- An *encryption key* to encrypt the data (plaintext).
- An *encryption algorithm* that, with the encryption key, transforms the plaintext into *cipher-text*.
- A *decryption key* to decrypt the cipher-text.
- A *decryption algorithm* that, with the decryption key, transforms the cipher-text back into plaintext.

7. RAID Tools:

RAID (Redundant Array of Independent Disks) works on having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance.

Performance is increased through *data* striping (the data is segmented into equal-size partitions) which are transparently distributed across multiple disks. Reliability is improved through storing redundant information across the disks using a *parity* scheme or an *error-correcting* scheme.

There are a number of different disk configurations with RAID, termed RAID *levels*:

- **RAID 0 – Non-redundant**: This level maintains no redundant data and so has the best write performance since updates do not have to be replicated. Data striping is performed at the level of blocks.
- **RAID 1 – Mirrored**: This level maintains (*mirrors*) two identical copies of the data across different disks. To maintain consistency in the presence of disk failure, writes may not be performed simultaneously. This is the most expensive storage solution.
- **RAID 2 – Memory-Style Error-Correcting Codes**: With this level, the striping unit is a single bit and Hamming codes are used as the redundancy scheme.
- **RAID 3 – Bit-Interleaved Parity**: This level provides redundancy by storing parity information on a single disk in the array. This parity information can be used to recover the data on other disks should they fail. This level uses less storage space than RAID 1 but the parity disk can become a bottleneck.
- **RAID 4 – Block-Interleaved Parity**: With this level, the striping unit is a disk block – a parity block is maintained on a separate disk for corresponding blocks from a number of other disks. If one of the disks fails, the parity block can be used with the corresponding blocks from the other disks to restore the blocks of the failed disk.
- **RAID 5 – Block-Interleaved Distributed Parity**: This level uses parity data for redundancy in a similar way to RAID 3 but stripes the parity data across all the disks, similar to the way in which the source data is striped.
- **RAID 6 – Dual Distributed Parity**: This level is similar to RAID 5 but additional redundant data is maintained to protect against multiple disk failures. Error-correcting codes are used instead of using parity.