Operating Systems

- An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks. The purpose of an operating system is to provide an environment in which a user can execute programs in a convenient and efficient manner.

- **Types of Operating Systems** :

1. **Batch OS –** A set of similar jobs are stored in the main memory for execution. A job gets assigned to the CPU, only when the execution of the previous job completes.
2. **Multiprogramming OS –** The main memory consists of jobs waiting for CPU time. The OS selects one of the processes and assigns it to the CPU. Whenever the executing process needs to wait for any other operation (like I/O), the OS selects another process from the job queue and assigns it to the CPU. **This way, the CPU is never kept idle** and the user gets the flavor of getting multiple tasks done at once.
3. **Multitasking OS –** Multitasking OS combines the benefits of **Multiprogramming OS and CPU scheduling** to perform quick switches between jobs. The switch is so quick that the user can interact with each program as it runs.
4. **Time Sharing OS –** Time-sharing systems require interaction with the user to instruct the OS to perform various tasks. The OS responds with an output. The instructions are usually given through an input device like the keyboard.
5. **Real Time OS –** Real-Time OS are usually built for dedicated systems to accomplish a specific set of tasks within deadlines.

- **Process** : A process is a program under execution. **The value of the program counter (PC) indicates the address of the next instruction of the process being executed**. Each process is represented by a Process Control Block (PCB).

- **Process Scheduling:**
  1. **Arrival Time –** Time at which the process arrives in the ready queue.
  2. **Completion Time –** Time at which process completes its execution.
  3. **Burst Time –** Time required by a process for CPU execution.
  4. **Turn Around Time –** Time Difference between completion time and arrival time.
     Turn Around Time = Completion Time - Arrival Time
  5. **Waiting Time (WT) –** Time Difference between turn around time and burst time.
     Waiting Time = Turnaround Time - Burst Time

- **Thread (Important) :** A thread is a lightweight process and forms the basic unit of CPU utilization. A process can perform more than one task at the same time by including multiple threads.
  - A thread has its own program counter, register set, and stack
  - A thread shares resources with other threads of the same process: the code section, the data section, files and signals.

Note : A new thread, or a child process of a given process, can be introduced by using the fork() system call. A process with n fork() system call generates $2^{n} - 1$ child processes.

There are two types of threads:

- User threads (User threads are implemented by users)
- Kernel threads (Kernel threads are implemented by OS)

- **Scheduling Algorithms :**

1. **First Come First Serve (FCFS)** : Simplest scheduling algorithm that schedules according to arrival times of processes.
2. **Shortest Job First (SJF)**: Processes which have the shortest burst time are scheduled first.
3. **Shortest Remaining Time First (SRTF)**: It is a preemptive mode of SJF algorithm in which jobs are scheduled according to the shortest remaining time.
4. **Round Robin (RR) Scheduling**: Each process is assigned a fixed time, in a cyclic way.
5. **Priority Based scheduling (Non Preemptive)**: In this scheduling, processes are scheduled according to their priorities, i.e., highest priority process is scheduled first. If priorities of two processes match, then scheduling is according to the arrival time.
6. **Highest Response Ratio Next (HRRN)**: In this scheduling, processes with the highest response ratio are scheduled. This algorithm avoids starvation.
   `Response Ratio = (Waiting Time + Burst time) / Burst time`
7. **Multilevel Queue Scheduling (MLQ)**: According to the priority of the process, processes are placed in the different queues. Generally high priority processes are placed in the top level queue. Only after completion of processes from the top level queue, lower level queued processes are scheduled.
8. **Multilevel Feedback Queue (MLFQ) Scheduling**: It allows the process to move in between queues. The idea is to separate processes according to the characteristics of their CPU bursts. If a process uses too much CPU time, it is moved to a lower-priority queue.

- **The Critical Section Problem:**
  1. **Critical Section –** The portion of the code in the program where shared variables are accessed and/or updated.
  2. **Remainder Section –** The remaining portion of the program excluding the Critical Section.
  3. **Race around Condition –** The final output of the code depends on the order in which the variables are accessed. This is termed as the race around condition.

A solution for the critical section problem must satisfy the following three conditions:

1. **Mutual Exclusion –** If a process Pi is executing in its critical section, then no other process is allowed to enter into the critical section.
2. **Progress –** If no process is executing in the critical section, then the decision of a process to enter a critical section cannot be made by any other process that is executing in its remainder section. The selection of the process cannot be postponed indefinitely.
3. **Bounded Waiting –** There exists a bound on the number of times other processes can enter into the critical section after a process has made a request to access the critical section and before the request is granted.

- **Synchronization Tools:**
  1. **Semaphore** : Semaphore is a protected variable or abstract data type that is used to lock the resource being used. The value of the semaphore indicates thestatus of a common resource.

     There are two types of semaphores:
     Binary semaphores (Binary semaphores take only 0 and 1 as value and are usedto implement mutual exclusion and synchronize concurrent processes.) **Counting semaphores** (A counting semaphore is an integer variable whose valuecan range over an unrestricted domain.)

Mutex (A mutex provides mutual exclusion, either producer or consumer can have the key (mutex) and proceed with their work. As long as the buffer is filled by the producer, the consumer needs to wait, and vice versa.
At any point of time, only one thread can work with the entire buffer. The concept can be generalized using semaphore.)

- **Deadlocks (Important)**:

A situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Deadlock can arise if following four conditions hold simultaneously (Necessary Conditions):

1. **Mutual Exclusion –** One or more than one resource is non-sharable (Only one process can use at a time).
2. **Hold and Wait –** A process is holding at least one resource and waiting for resources.
3. **No Preemption –** A resource cannot be taken from a process unless the process releases the resource.
4. **Circular Wait –** A set of processes are waiting for each other in circular form.

- **Methods for handling deadlock:** There are three ways to handle deadlock

1. **Deadlock prevention or avoidance** : The idea is to not let the system into a deadlock state.
2. **Deadlock detection and recovery** : Let deadlock occur, then do preemption to handle it once occurred.
3. **Ignore the problem all together** : If deadlock is very rare, then let it happen and reboot the system. This is the approach that both Windows and UNIX take.

- **Banker's algorithm** is used to avoid deadlock. It is one of the deadlock-avoidance methods. It is named as Banker's algorithm on the banking system where a bank never allocates available cash in such a manner that it can no longer satisfy therequirements of all of its customers.

- **Memory Management:** These techniques allow the memory to be shared among multiple processes.
  - **Overlays –** The memory should contain only those instructions and data that are required at a given time.
  - **Swapping –** In multiprogramming, the instructions that have used the time slice are swapped out from the memory.
- **Techniques** :

  (a) Single Partition Allocation Schemes – The memory is divided into two parts. One part is kept to be used by the OS and the other is kept to be used by the users.

  (b) Multiple Partition Schemes –

  1. **Fixed Partition –** The memory is divided into fixed size partitions.
  2. **Variable Partition –** The memory is divided into variable sized partitions.

  Note : Variable partition allocation schemes:

  1. **First Fit –** The arriving process is allotted the first hole of memory in which it fits completely.
  2. **Best Fit –** The arriving process is allotted the hole of memory in which it fits the best by leaving the minimum memory empty.
  3. **Worst Fit –** The arriving process is allotted the hole of memory in which it leaves the maximum gap.

Note:

- Best fit does not necessarily give the best results for memory allocation.
- The cause of external fragmentation is the condition in Fixed partitioning and Variable partitioning saying that the entire process should be allocated in a contiguous memory location.Therefore **Paging** is used.

1. **Paging –** The physical memory is divided into equal sized frames. The main memory is divided into fixed size pages. The size of a physical memory frame is equal to the size of a virtual memory frame.
2. **Segmentation –** Segmentation is implemented to give users a view of memory. The logical address space is a collection of segments. Segmentation can be implemented with or without the use of paging.

- **Page Fault:**

A page fault is a type of interrupt, raised by the hardware when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Page Replacement Algorithms (Important):

1. **First In First Out (FIFO) –**

   This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced, the page in the front of the queue is selected for removal.

   For example, consider page reference string 1, 3, 0, 3, 5, 6 and 3 page slots. Initially, all slots are empty, so when 1, 3, 0 come they are allocated to the empty slots —> 3 Page Faults. When 3 comes, it is already in memory so —> 0 Page Faults. Then 5 comes, it is not available in memory so it replaces the oldest page slot i.e 1. —> 1 Page Fault. Finally, 6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —> 1 Page Fault.

   Belady's anomaly:

Belady's anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm. For example, if we consider reference string (3 2 1 0 3 2 4 3 2 1 0 4) and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

2. **Optimal Page replacement –**

In this algorithm, pages are replaced which are not used for the longest duration of time in the future.

Let us consider page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 and 4 page slots. Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults. 0 is already there so —> 0 Page fault. When 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—> 1 Page fault. 0 is already there so —> 0 Page fault. 4 will takes place of 1 —> 1 Page Fault. Now for the further page reference string —> 0 Page fault because they are already available in the memory.

Optimal page replacement is perfect, but not possible in practice as an operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

3. **Least Recently Used (LRU) –**

In this algorithm, the page will be replaced with the one which is least recently used. Let say the page reference string 7 0 1 2 0 3 0 4 2 3 0 3 2 . Initially, we had 4-page slots empty. Initially, all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> 4 Page faults. 0 is already there so —> 0 Page fault. When 3 comes it will take the place of 7 because it is least recently used —> 1 Page fault. 0 is already in memory so —> 0 Page fault. 4 will take place of 1 —> 1 Page Fault. Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

- **Disk Scheduling**: Disk scheduling is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.
  1. **Seek Time:** Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or written.
  2. **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads.
  3. **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
  4. **Disk Access Time:** Seek Time + Rotational Latency + Transfer Time
  5. **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. Average Response time is the response time of all requests.

- **Disk Scheduling Algorithms (Important)**:
  1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.
  2. **SSTF:** In SSTF (Shortest Seek Time First), requests having the shortest seek time are executed first. So, the seek time of every request is calculated in advance in a queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first.
  3. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of the disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence is also known as **elevator algorithm.**
  4. **CSCAN:** In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

5. **LOOK:** It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.
6. **CLOOK:** As LOOK is similar to SCAN algorithm, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Key Terms

- **Real-time system** is used in the case when rigid-time requirements have been placed on the operation of a processor. It contains well defined and fixed time constraints.
- **A monolithic kernel** is a kernel which includes all operating system code in a single executable image.
- **Micro kernel:** Microkernel is the kernel which runs minimal performance affecting services for the operating system. In the microkernel operating system all other operations are performed by the processor.

  Macro Kernel: Macro Kernel is a combination of micro and monolithic kernel.

- **Re-entrancy :** It is a very useful memory saving technique that is used for multi-programmed time sharing systems. It provides functionality that multiple users can share a single copy of a program during the same period. It has two key aspects:The program code cannot modify itself and the local data for each user process must be stored separately.
- **Demand paging** specifies that if an area of memory is not currently being used, it is swapped to disk to make room for an application's need.
- **Virtual memory (Imp)** is a very useful memory management technique which enables processes to execute outside of memory. This technique is especially used when an executing program cannot fit in the physical memory.
- **RAID** stands for Redundant Array of Independent Disks. It is used to store the same data redundantly to improve the overall performance. There are 7 RAID levels.
- **Logical address space** specifies the address that is generated by the CPU. On the other hand, **physical address space** specifies the address that is seen by the memory unit.

- **Fragmentation** is a phenomenon of memory wastage. It reduces the capacity and performance because space is used inefficiently.
    1. **Internal fragmentation**: It occurs when we deal with the systems that have fixed size allocation units.
    2. **External fragmentation**: It occurs when we deal with systems that have variable-size allocation units.
- **Spooling** is a process in which data is temporarily gathered to be used and executed by a device, program or the system. It is associated with printing. When different applications send output to the printer at the same time, spooling keeps these all jobs into a disk file and queues them accordingly to the printer.
- **Starvation** is Resource management problem. In this problem, a waiting process does not get the resources it needs for a long time because the resources are being allocated to other processes.
- **Aging** is a technique used to avoid starvation in the resource scheduling system.
- **Advantages of multithreaded programming**:
    1. Enhance the responsiveness to the users.
    2. Resource sharing within the process.
    3. Economical
    4. Completely utilize the multiprocessing architecture.
- **Thrashing** is a phenomenon in virtual memory schemes when the processor spends most of its time in swapping pages, rather than executing instructions.

DBMS & SQL NOTES

Database: A database is a collection of related data which represents some aspect of the real world. A database system is designed to be built and populated with data for a certain task.

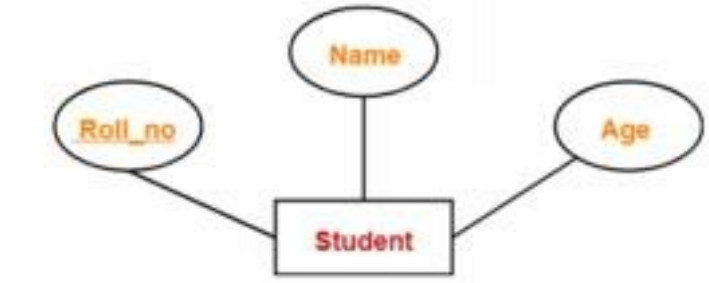
Database Management System (DBMS) is a software for storing and retrieving users' data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

Database management systems were developed to handle the following difficulties of typical File-processing systems supported by conventional operating systems.

1. Data redundancy and inconsistency
2. Difficulty in accessing data
3. Data isolation – multiple files and formats
4. Integrity problems
5. Atomicity of updates
6. Concurrent access by multiple users
7. Security problems

ER diagram:

- ER diagram or **Entity Relationship diagram** is a conceptual model that gives the graphical representation of the logical structure of the database.
- It shows all the constraints and relationships that exist among the different components.
- An ER diagram is mainly composed of following three components- Entity Sets, Attributes and Relationship Set.



- Roll_no is a primary key that can identify each entity uniquely.

- Thus, by using a student's roll number, a student can be identified uniquely.

Entity Set:

An entity set is a set of the same type of entities.

- **Strong Entity Set:**
  - A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities.
  - In other words, a primary key exists for a strong entity set.
  - Primary key of a strong entity set is represented by underlining it.
- **Weak Entity Set:**
  - A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.
  - In other words, a primary key does not exist for a weak entity set.
  - However, it contains a partial key called a discriminator.
  - Discriminator can identify a group of entities from the entity set.
  - Discriminator is represented by underlining with a dashed line.

Relationship:

A relationship is defined as an association among several entities.

- **Unary Relationship Set -** Unary relationship set is a relationship set where only one entity set participates in a relationship set.
- **Binary Relationship Set -** Binary relationship set is a relationship set where two entity sets participate in a relationship set.
- **Ternary Relationship Set -** Ternary relationship set is a relationship set where three entity sets participate in a relationship set.
- **N-ary Relationship Set -** N-ary relationship set is a relationship set where 'n' entity sets participate in a relationship set.

Cardinality Constraint:

Cardinality constraint defines the maximum number of relationship instances in which an entity can participate.

- **One-to-One Cardinality -** An entity in set A can be associated with at most one entity in set B. An entity in set B can be associated with at most one entity in set A.
- **One-to-Many Cardinality -** An entity in set A can be associated with any number (zero or

more) of entities in set B. An entity in set B can be associated with at most one entity in set A.

- **Many-to-One Cardinality -** An entity in set A can be associated with at most one entity in set B. An entity in set B can be associated with any number of entities in set A.
- **Many-to-Many Cardinality -** An entity in set A can be associated with any number (zero or more) of entities in set B. An entity in set B can be associated with any number (zero or more) of entities in set A.

Attributes:

Attributes are the descriptive properties which are owned by each entity of an Entity Set.

Types of Attributes:

- **Simple Attributes -** Simple attributes are those attributes which cannot be divided further. Ex. Age
- **Composite Attributes -** Composite attributes are those attributes which are composed of many other simple attributes. Ex. Name, Address
- **Multi Valued Attributes -** Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set. Ex. Mobile No, Email ID
- **Derived Attributes -** Derived attributes are those attributes which can be derived from other attribute(s). Ex. Age can be derived from DOB.
- **Key Attributes -** Key attributes are those attributes which can identify an entity uniquely in an entity set. Ex. Roll No.

Constraints:

Relational constraints are the restrictions imposed on the database contents and operations. They ensure the correctness of data in the database.

- **Domain Constraint -** Domain constraint defines the domain or set of values for an attribute. It specifies that the value taken by the attribute must be the atomic value from its domain.
- **Tuple Uniqueness Constraint -** Tuple Uniqueness constraint specifies that all the tuples must be necessarily unique in any relation.
- **Key Constraint -** All the values of the primary key must be unique. The value of the primary key must not be null.
- **Entity Integrity Constraint -** Entity integrity constraint specifies that no attribute of primary key must contain a null value in any relation.
- **Referential Integrity Constraint -** It specifies that all the values taken by the foreign key

must either be available in the relation of the primary key or be null.

Closure of an Attribute Set:

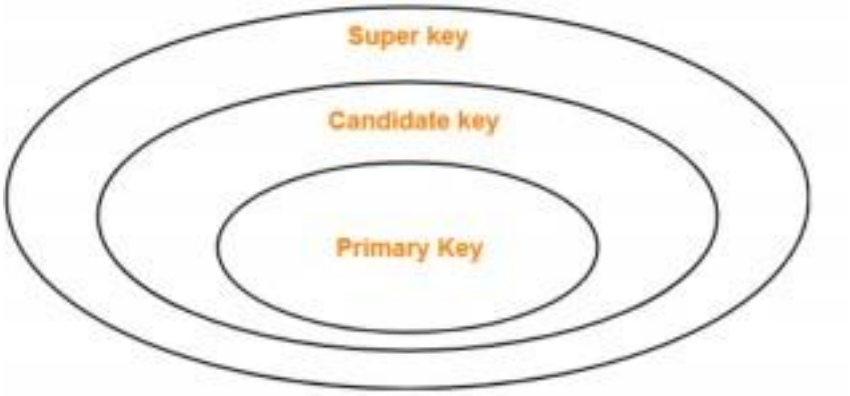
The set of all those attributes which can be functionally determined from an attribute set is called a closure of that attribute set.

Keys:

A key is a set of attributes that can identify each tuple uniquely in the given relation**.**

Types of Keys:

- **Super Key -** A superkey is a set of attributes that can identify each tuple uniquely in the given relation. A super key may consist of any number of attributes.
- **Candidate Key -** A set of minimal attribute(s) that can identify each tuple uniquely in the given relation is called a candidate key.
- **Primary Key -** A primary key is a candidate key that the database designer selects while designing the database. Primary Keys are unique and NOT NULL.



- **Alternate Key -** Candidate keys that are left unimplemented or unused after implementing the primary key are called as alternate keys.
- **Foreign Key -** An attribute 'X' is called as a foreign key to some other attribute 'Y' when its values are dependent on the values of attribute 'Y'. The relation in which attribute 'Y' is present is called as the referenced relation. The relation in which attribute 'X' is present is called as the referencing relation.
- **Composite Key -** A primary key composed of multiple attributes and not just a single attribute is called a composite key.
- **Unique Key -** It is unique for all the records of the table. Once assigned, its value cannot be changed i.e. it is non-updatable. It may have a NULL value.

Functional Dependency:

In any relation, a functional dependency $\alpha \rightarrow \beta$ holds if- Two tuples having same value

of attribute α also have same value for attribute β.

Types of Functional Dependency:

- **Trivial Functional Dependencies –**
  - A functional dependency X ⟶ Y is said to be trivial if and only if Y ⊆ X.
  - Thus, if RHS of a functional dependency is a subset of LHS, then it is called a trivial functional dependency.
- **Non-Trivial Functional Dependencies –**
  - A functional dependency X ⟶ Y is said to be non-trivial if and only if Y ⊄ X.
  - Thus, if there exists at least one attribute in the RHS of a functional dependency that is not a part of LHS, then it is called a non-trivial functional dependency.

Decomposition of a Relation:

The process of breaking up or dividing a single relation into two or more sub relations is called the decomposition of a relation.

Properties of Decomposition:

- **Lossless Decomposition -** Lossless decomposition ensures
  - No information is lost from the original relation during decomposition.
  - When the sub relations are joined back, the same relation is obtained that was decomposed.
- **Dependency Preservation -** Dependency preservation ensures
  - None of the functional dependencies that hold on the original relation are lost.
  - The sub relations still hold or satisfy the functional dependencies of the original relation.

Types of Decomposition:

- **Lossless Join Decomposition:**
  - Consider there is a relation R which is decomposed into sub relations R1, R2, …., Rn.
  - This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.
  - For lossless join decomposition, we always have- R1 ⋈ R2 ⋈ R3 ……. ⋈ Rn = R where ⋈ is a natural join operator
- **Lossy Join Decomposition:**
  - Consider there is a relation R which is decomposed into sub relations R1, R2, …., Rn.
  - This decomposition is called lossy join decomposition when the join of the sub

relations does not result in the same relation R that was decomposed.

- For lossy join decomposition, we always have- R1 ⋈ R2 ⋈ R3 ……. ⋈ Rn ⊃ R where ⋈ is a natural join operator

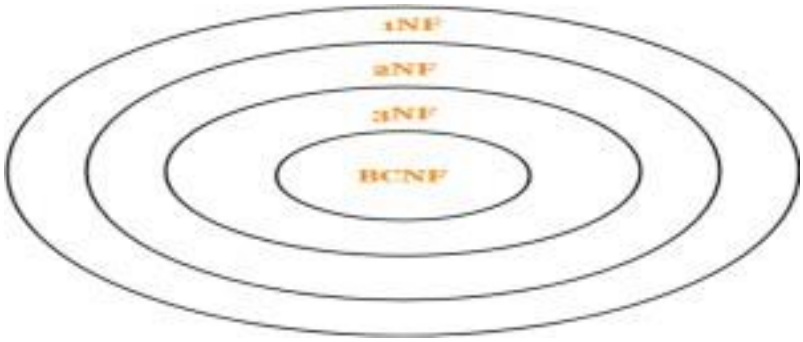
Normalization:

In DBMS, database normalization is a process of making the database consistent by-

- Reducing the redundancies
- Ensuring the integrity of data through lossless decomposition

Normal Forms:

- **First Normal Form (1NF) -** A given relation is called in First Normal Form (1NF) if each cell of the table contains only an atomic value i.e. if the attribute of every tuple is either single valued or a null value.
- **Second Normal Form (2NF) -** A given relation is called in Second Normal Form (2NF) if and only if
  - Relation already exists in 1NF.
  - No partial dependency exists in the relation.
    A ⟶ B is called a **partial dependency** if and only if- A is a subset of some candidate key and B is a non-prime attribute.
- **Third Normal Form (3NF) -** A given relation is called in Third Normal Form (3NF) if and only if
  - Relation already exists in 2NF.
  - No transitive dependency exists for non-prime attributes.
    A ⟶ B is called a **transitive dependency** if and only if- A is not a super key and B is a non-prime attribute.
- **Boyce-Codd Normal Form -** A given relation is called in BCNF if and only if
  - Relation already exists in 3NF.
  - For each non-trivial functional dependency 'A ⟶ B', A is a super key of the relation.



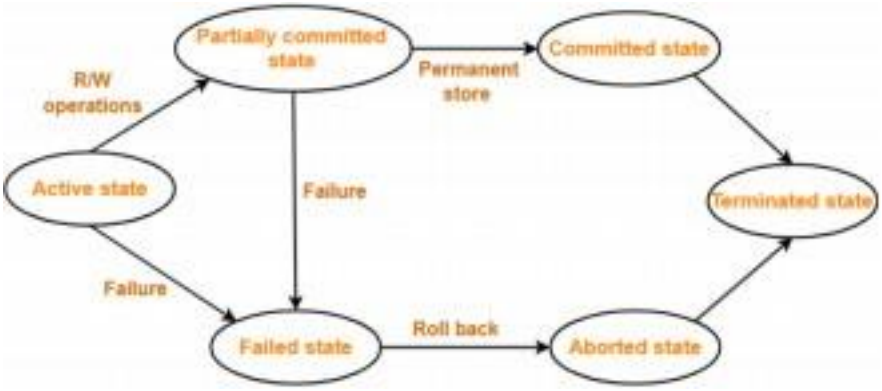
Transaction:

Transaction is a single logical unit of work formed by a set of operations.

Operations in Transaction:

- **Read Operation - Read(A)** instruction will read the value of 'A' from the database and will store it in the buffer in main memory.
- **Write Operation – Write(A)** will write the updated value of 'A' from the buffer to the database.

Transaction States:



- **Active State –**
  - This is the first state in the life cycle of a transaction.
  - A transaction is called in an active state as long as its instructions are getting executed.
  - All the changes made by the transaction now are stored in the buffer in main memory.
- **Partially Committed State –**
  - After the last instruction of the transaction has been executed, it enters into a partially committed state.
  - After entering this state, the transaction is considered to be partially committed.
  - It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.
- **Committed State –**
  - After all the changes made by the transaction have been successfully stored into the database, it enters into a committed state.
  - Now, the transaction is considered to be fully committed.
- **Failed State –**
  - When a transaction is getting executed in the active state or partially committed state and some failure occurs due to which it becomes impossible to continue the execution, it enters into a failed state.

- **Aborted State –**
  - After the transaction has failed and entered into a failed state, all the changes made by it have to be undone.
  - To undo the changes made by the transaction, it becomes necessary to roll back the transaction.
  - After the transaction has rolled back completely, it enters into an aborted state.
- **Terminated State –**
  - This is the last state in the life cycle of a transaction.
  - After entering the committed state or aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end.

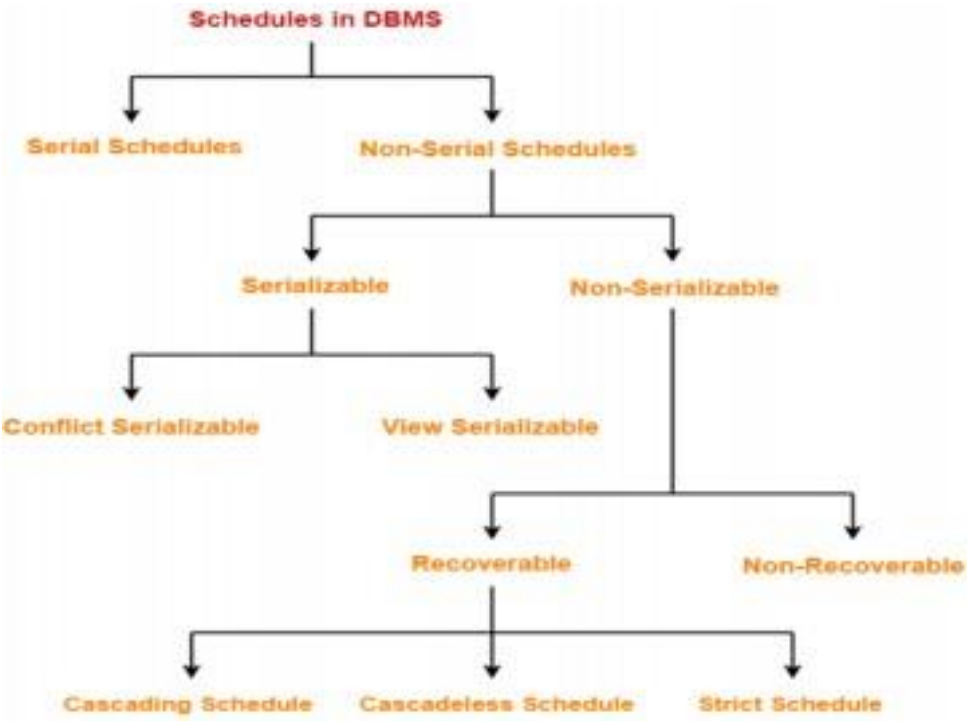
ACID Properties:

To ensure the consistency of the database, certain properties are followed by all the transactions occurring in the system. These properties are called as **ACID Properties** of a transaction.

- **Atomicity –**
  - This property ensures that either the transaction occurs completely or it does not occur at all.
  - In other words, it ensures that no transaction occurs partially.
- **Consistency –**
  - This property ensures that integrity constraints are maintained.
  - In other words, it ensures that the database remains consistent before and after the transaction.
- **Isolation –**
  - This property ensures that multiple transactions can occur simultaneously without causing any inconsistency.
  - The resultant state of the system after executing all the transactions is the same as the state that would be achieved if the transactions were executed serially one after the other.
- **Durability –**
  - This property ensures that all the changes made by a transaction after its successful execution are written successfully to the disk.
  - It also ensures that these changes exist permanently and are never lost even if there occurs a failure of any kind.

Schedules:

The order in which the operations of multiple transactions appear for execution is called as a schedule.



- **Serial Schedules –**
  - All the transactions execute serially one after the other.
  - When one transaction executes, no other transaction is allowed to execute.
- Serial schedules are always- Consistent, Recoverable, Cascadeless and Strict. ●

Non-Serial Schedules –

- Multiple transactions execute concurrently.
- Operations of all the transactions are inter leaved or mixed with each other.
- Non-serial schedules are **not** always- Consistent, Recoverable, Cascadeless and Strict.

Serializability –

- Some non-serial schedules may lead to inconsistency of the database. ● Serializability is a concept that helps to identify which non-serial schedules are correct and will maintain the consistency of the database.

- **Serializable Schedules –**
  - If a given non-serial schedule of 'n' transactions is equivalent to some serial schedule of 'n' transactions, then it is called as a serializable schedule.
- Serializable schedules are always- Consistent, Recoverable, Cascadeless and Strict.

Types of Serializability –

- **Conflict Serializability** - If a given non-serial schedule can be converted into a serial schedule by swapping its non-conflicting operations, then it is called a conflict serializable schedule.
- **View Serializability -** If a given schedule is found to be viewed as equivalent to some serial schedule, then it is called a view serializable schedule.

Non-Serializable Schedules –

- A non-serial schedule which is not serializable is called a non-serializable schedule. ● A non-serializable schedule is not guaranteed to produce the same effect as produced by some serial schedule on any consistent database.
- Non-serializable schedules- may or may not be consistent, may or may not be recoverable.
- **Irrecoverable Schedules –**
  If in a schedule,
  - A transaction performs a dirty read operation from an uncommitted transaction
  - And commits before the transaction from which it has read the value then such a schedule is known as an Irrecoverable Schedule.
- **Recoverable Schedules –**
  If in a schedule,
  - A transaction performs a dirty read operation from an uncommitted transaction
  - And its commit operation is delayed till the uncommitted transaction either commits or roll backs

  then such a schedule is known as a Recoverable Schedule.

Types of Recoverable Schedules –

- **Cascading Schedule -** If in a schedule, failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a Cascading Schedule or Cascading Rollback or Cascading Abort.
- **Cascadeless Schedule -** If in a schedule, a transaction is not allowed to read a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a Cascadeless Schedule.
- **Strict Schedule -** If in a schedule, a transaction is neither allowed to read nor write a data item until the last transaction that has written it is committed or aborted, then such a schedule is called as a Strict Schedule.

Relational Algebra:

Relational Algebra is a procedural query language which takes a relation as an input and generates a relation as an output.

Basic Operator	Semantic
σ(Selection)	Select rows based on given condition
∏(Projection)	Project some columns
X (Cross Product)	Cross product of relations, returns **m*n** rows where m and n are number of rows in R1 and R2 respectively.
U (Union)	Return those tuples which are either in R1 or in R2. Max no. of rows returned **= m+n** and Min no. of rows returned = **max(m,n)**
−(Minus)	R1-R2 returns those tuples which are in R1 but not in R2. Max no. of rows returned = **m** and Min no. of rows returned = **m-n**
ρ(Rename)	Renaming a relation to another relation.

Extended Operator	Semantic
∩ (Intersection)	Returns those tuples which are in both R1 and R2. Max no. of rows returned = min(m,n) and Min no. of rows returned = 0
$\bowtie_c$(Conditional Join)	Selection from two or more tables based on some condition (Cross product followed by selection)
⋈(Equi Join)	It is a special case of conditional join when only equality conditions are applied between attributes.
⋈(Natural Join)	In natural join, equality conditions on common attributes hold and duplicate attributes are removed by default. **Note:** Natural Join is equivalent to cross product if two relations have no attribute in common and natural join of a relation R with itself will return R only.

⟕**(Left Outer Join)**	When applying join on two relations R and S, some tuples of R or S do not appear in the result set which does not satisfy the join conditions. But Left Outer Joins gives all tuples of R in the result set. The tuples of R which do not satisfy the join condition will have values as NULL for attributes of S.
⟖**(Right Outer Join)**	When applying join on two relations R and S, some tuples of R or S do not appear in the result set which does not satisfy the join conditions. But Right Outer Joins gives all tuples of S in the result set. The tuples of S which do not satisfy the join condition will have values as NULL for attributes of R.
⟗(Full Outer Join)	When applying join on two relations R and S, some tuples of R or S do not appear in the result set which does not satisfy the join conditions. But Full Outer Joins gives all tuples of S and all tuples of R in the result set. The tuples of S which do not satisfy the join condition will have values as NULL for attributes of R and vice versa.
/(Division Operator)	Division operator A/B will return those tuples in A which are associated with every tuple of B. **Note:** Attributes of B should be a proper subset of attributes of A. The attributes in A/B will be Attributes of A- Attribute of B.

File Structures:

- **Primary Index:** A primary index is an ordered file, records of fixed length with two fields. First field is the same as the primary key as a data file and the second field is a pointer to the data block, where the key is available. The average number of block accesses using index = $\log_2 Bi + 1$, where Bi = number of index blocks.
- **Clustering Index:** Clustering index is created on data file whose records are physically ordered on a non-key field (called Clustering field).
- **Secondary Index:** Secondary index provides secondary means of accessing a file for which primary access already exists.

B Trees

At every level , we have Key and Data Pointer and data pointer points to either block or record.

Properties of B-Trees:

Root of B-tree can have children between **2** and **P**, where P is Order of tree.

Order of tree – Maximum number of children a node can have.

Internal node can have children between ⌈ **P/2** ⌉ and **P**
Internal node can have keys between ⌈ **P/2** ⌉ **– 1** and **P-1**

B+ Trees

In B+ trees, the structure of leaf and non-leaf are different, so their order is. Order of non-leaf will be higher as compared to leaf nodes.

Searching time will be less in B+ trees, since it doesn't have record pointers in non-leaf because of which depth will decrease.

SQL

DDL:

DDL is short name of **Data Definition Language,** which deals with database schemas and descriptions, of how the data should reside in the database.

- CREATE - to create a database and its objects like (table, index, views, store procedure, function, and triggers)
- ALTER - alters the structure of the existing database
- DROP - delete objects from the database
- TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed
- RENAME - rename an object

DML:

DML is short name of **Data Manipulation Language** which deals with data manipulation and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE, etc., and it is used to store, modify, retrieve, delete and update data in a database.

- SELECT - retrieve data from a database
- INSERT - insert data into a table
- UPDATE - updates existing data within a table
- DELETE - Delete all records from a database table
- MERGE - UPSERT operation (insert or update)

DCL:

DCL is short name of **Data Control Language** which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

- GRANT - allow users access privileges to the database
- REVOKE - withdraw users access privileges given by using the GRANT command

TCL:

TCL is short name of Transaction Control Language which deals with a transaction within a database.

- COMMIT - commits a Transaction
- ROLLBACK - rollback a transaction in case of any error occurs
- SAVEPOINT - to roll back the transaction making points within groups

SQL:

SQL is a standard language for storing, manipulating and retrieving data in databases.

SELECT:

The SELECT statement is used to select data from a database.

Syntax -

- SELECT *column1*, *column2, ...*
  FROM *table_name*;
- Here, column1, column2, ... are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax: ● SELECT * FROM *table_name*;

Ex –

- SELECT CustomerName, City FROM Customers;

SELECT DISTINCT:

The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax –

- SELECT DISTINCT *column1*, *column2, ...*
  FROM *table_name*;

Ex –

- SELECT DISTINCT Country FROM Customers;

WHERE:

The WHERE clause is used to filter records.

Syntax –

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition*;

Ex –

- SELECT * FROM Customers
  WHERE Country='Mexico';

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=

AND, OR and NOT:

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.
- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

Syntax –

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition1* AND *condition2* AND *condition3 ...*;
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE *condition1* OR *condition2* OR *condition3 ...*;
- SELECT *column1*, *column2, ...*
  FROM *table_name*
  WHERE NOT *condition*;

Ex –

- SELECT * FROM Customers
  WHERE Country='Germany' AND City='Berlin';
- SELECT * FROM Customers
  WHERE Country='Germany' AND (City='Berlin' OR City='München');

ORDER BY:

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax –

- SELECT *column1*, *column2, ...*
  FROM *table_name*
  ORDER BY *column1, column2, ...* ASC|DESC;

Ex –

- SELECT * FROM Customers
  ORDER BY Country;
- SELECT * FROM Customers
  ORDER BY Country ASC, CustomerName DESC;

INSERT INTO:

The INSERT INTO statement is used to insert new records in a table.

Syntax –

- INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
  VALUES (*value1*, *value2*, *value3*, ...);
- INSERT INTO *table_name*
  VALUES (*value1*, *value2*, *value3*, ...);

*In the second syntax, make sure the order of the values is in the same order as the columns in the table.

Ex –

- INSERT INTO Customers (CustomerName, ContactName, Address, City, PostalCode, Country)
  VALUES ('Cardinal', 'Tom B. Erichsen', 'Skagen 21', 'Stavanger', '4006', 'Norway');

NULL Value:

It is not possible to test for NULL values with comparison operators, such as =, <, or

<>. We will have to use the IS NULL and IS NOT NULL operators instead.

Syntax –

- SELECT *column_names*
  FROM *table_name*
  WHERE *column_name* IS NULL;
- SELECT *column_names*
  FROM *table_name*
  WHERE *column_name* IS NOT NULL;

Ex –

- SELECT CustomerName, ContactName, Address
  FROM Customers
  WHERE Address IS NULL;

UPDATE:

The UPDATE statement is used to modify the existing records in a table.

Syntax –

- UPDATE *table_name*

SET *column1* = *value1*, *column2* = *value2*, ...
WHERE *condition*;

Ex –

- UPDATE Customers
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
  WHERE CustomerID = 1;

DELETE:

The DELETE statement is used to delete existing records in a table.

Syntax –

- DELETE FROM *table_name* WHERE *condition*;
- DELETE FROM *table_name*;

In 2nd syntax, all rows are deleted. The table structure, attributes, and indexes will be intact

Ex –

- DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

SELECT TOP:

The SELECT TOP clause is used to specify the number of records to return.

Syntax –

- SELECT TOP *number*|*percent column_name(s)*
  FROM *table_name*
  WHERE *condition*;
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *condition*
  LIMIT *number*;
- SELECT *column_name(s)*
  FROM *table_name*
  ORDER BY *column_name(s)*
  FETCH FIRST *number* ROWS ONLY;
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE ROWNUM <= *number*;

*In case the interviewer asks other than the TOP, rest are also correct. (Diff. DB Systems)

Ex –

- SELECT TOP 3 * FROM Customers;
- SELECT * FROM Customers
  LIMIT 3;
- SELECT * FROM Customers
  FETCH FIRST 3 ROWS ONLY;

Aggregate Functions:

MIN():

The MIN() function returns the smallest value of the selected column.

Syntax –

- SELECT MIN(*column_name*)
  FROM *table_name*
  WHERE *condition*;

Ex –

- SELECT MIN(Price) AS SmallestPrice
  FROM Products;

MAX():

The MAX() function returns the largest value of the selected column.

Syntax –

- SELECT MAX(*column_name*)
  FROM *table_name*
  WHERE *condition*;

Ex –

- SELECT MAX(Price) AS LargestPrice
  FROM Products;

COUNT():

The COUNT() function returns the number of rows that matches a specified criterion.

Syntax –

- SELECT COUNT(*column_name*)

FROM *table_name*
WHERE *condition*;

Ex –

- SELECT COUNT(ProductID)
  FROM Products;

AVG():
The AVG() function returns the average value of a numeric column.

Syntax –

- SELECT AVG(*column_name*)
  FROM *table_name*
  WHERE *condition*;

Ex –

- SELECT AVG(Price)
  FROM Products;

SUM():

The SUM() function returns the total sum of a numeric column.

Syntax –

- SELECT SUM(*column_name*)
  FROM *table_name*
  WHERE *condition*;

Ex –

- SELECT SUM(Quantity)
  FROM OrderDetails;

LIKE Operator:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

Syntax –

- SELECT *column1, column2, ...*
  FROM *table_name*
  WHERE *columnN* LIKE *pattern*;

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

IN:

The IN operator allows you to specify multiple values in a WHERE clause.

The IN operator is a shorthand for multiple OR conditions.

Syntax –

- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *column_name* IN (*value1*, *value2*, ...);
- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *column_name* IN (*SELECT STATEMENT*);

Ex –

- SELECT * FROM Customers
  WHERE Country IN ('Germany', 'France', 'UK');
- SELECT * FROM Customers

WHERE Country IN (SELECT Country FROM Suppliers);

BETWEEN:

The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.

The BETWEEN operator is inclusive: begin and end values are included.

Syntax –

- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *column_name* BETWEEN *value1* AND *value2;*

Ex –

- SELECT * FROM Products
  WHERE Price BETWEEN 10 AND 20;

Joins:

A JOIN clause is used to combine rows from two or more tables, based on a related column between them.

INNER JOIN:

The INNER JOIN keyword selects records that have matching values in both tables.

Syntax –

- SELECT *column_name(s)*
  FROM *table1*
  INNER JOIN *table2*
  ON *table1.column_name* = *table2.column_name*;

Ex –

- SELECT Orders.OrderID, Customers.CustomerName
  FROM Orders
  INNER JOIN Customers ON Orders.CustomerID = Customers.CustomerID;

LEFT (OUTER) JOIN:

The LEFT JOIN keyword returns all records from the left table (table1), and the matching records from the right table (table2). The result is 0 records from the right side, if there is no match.

Syntax –

- SELECT *column_name(s)*
  FROM *table1*
  LEFT JOIN *table2*
  ON *table1.column_name* = *table2.column_name*;

Ex –

- SELECT Customers.CustomerName, Orders.OrderID
  FROM Customers
  LEFT JOIN Orders ON Customers.CustomerID = Orders.CustomerID
  ORDER BY Customers.CustomerName;

RIGHT (OUTER) JOIN:

The RIGHT JOIN keyword returns all records from the right table (table2), and the matching records from the left table (table1). The result is 0 records from the left side, if there is no match.

Syntax –

- SELECT *column_name(s)*
  FROM *table1*
  RIGHT JOIN *table2*
  ON *table1.column_name* = *table2.column_name*;

Ex –

- SELECT Orders.OrderID, Employees.LastName, Employees.FirstName
  FROM Orders
  RIGHT JOIN Employees ON Orders.EmployeeID = Employees.EmployeeID
  ORDER BY Orders.OrderID;

FULL (OUTER) JOIN:

The FULL OUTER JOIN keyword returns all records when there is a match in left (table1) or right (table2) table records.

Syntax:

- SELECT *column_name(s)*

FROM *table1*
FULL OUTER JOIN *table2*
ON *table1.column_name* = *table2.column_name*
WHERE *condition*;

Ex –

- SELECT Customers.CustomerName, Orders.OrderID
  FROM Customers
  FULL OUTER JOIN Orders ON Customers.CustomerID=Orders.CustomerID
  ORDER BY Customers.CustomerName;

UNION:

The UNION operator is used to combine the result-set of two or more SELECT statements.

- Every SELECT statement within UNION must have the same number of columns
- The columns must also have similar data types
- The columns in every SELECT statement must also be in the same order

The UNION operator selects only distinct values by default. To allow duplicate values, use UNION ALL

Syntax –

- SELECT *column_name(s)* FROM *table1*
  UNION
  SELECT *column_name(s)* FROM *table2*;
- SELECT *column_name(s)* FROM *table1*
  UNION ALL
  SELECT *column_name(s)* FROM *table2*;

Ex –

- SELECT City FROM Customers
  UNION
  SELECT City FROM Suppliers
  ORDER BY City;

GROUP BY:

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns. **Syntax –**

- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *condition*
  GROUP BY *column_name(s)*
  ORDER BY *column_name(s);*

Ex –

- SELECT COUNT(CustomerID), Country
  FROM Customers
  GROUP BY Country
  ORDER BY COUNT(CustomerID) DESC;

HAVING:

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

*WHERE is given priority over HAVING.

Syntax –

- SELECT *column_name(s)*
  FROM *table_name*
  WHERE *condition*
  GROUP BY *column_name(s)*
  HAVING *condition*
  ORDER BY *column_name(s);*

Ex –

- SELECT COUNT(CustomerID), Country
  FROM Customers
  GROUP BY Country
  HAVING COUNT(CustomerID) > 5;

CREATE DATABASE:

The CREATE DATABASE statement is used to create a new SQL database.

Syntax –

- CREATE DATABASE *databasename*;

DROP DATABASE:

The DROP DATABASE statement is used to drop an existing SQL database.

Syntax –

- DROP DATABASE *databasename*;

CREATE TABLE:

The CREATE TABLE statement is used to create a new table in a database.

Syntax –

- CREATE TABLE *table_name* (
  column1 datatype,
  column2 datatype,
  column3 datatype,
  ....
  );

DROP TABLE:

The DROP TABLE statement is used to drop an existing table in a database.

Syntax –

- DROP TABLE *table_name*;

TRUNCATE TABLE:

The TRUNCATE TABLE statement is used to delete the data inside a table, but not the table itself.

Syntax –

- TRUNCATE TABLE *table_name*;

ALTER TABLE:

The ALTER TABLE statement is used to add, delete, or modify columns in an existing table.

The ALTER TABLE statement is also used to add and drop various constraints on an existing table.

Syntax –

- ALTER TABLE *table_name*
  ADD *column_name datatype*;
- ALTER TABLE *table_name*
  DROP COLUMN *column_name*;
- ALTER TABLE *table_name*
  MODIFY COLUMN *column_name datatype*;

Ex –

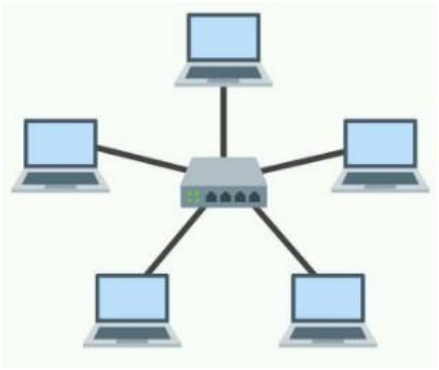
- ALTER TABLE Customers
  ADD Email varchar(255);
- ALTER TABLE Customers
  DROP COLUMN Email;
- ALTER TABLE Persons
  ALTER COLUMN DateOfBirth year;

Computer Networks

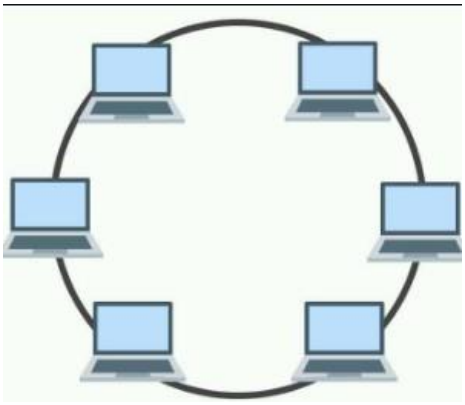
- **Network** : A network is a set of devices that are connected with a physical media link. In a network, two or more nodes are connected by a physical link or two or more networks are connected by one or more nodes. A network is a collection of devices connected to each other to allow the sharing of data.

- **Network Topology** : Network topology specifies the layout of a computer network. It shows how devices and cables are connected to each other.

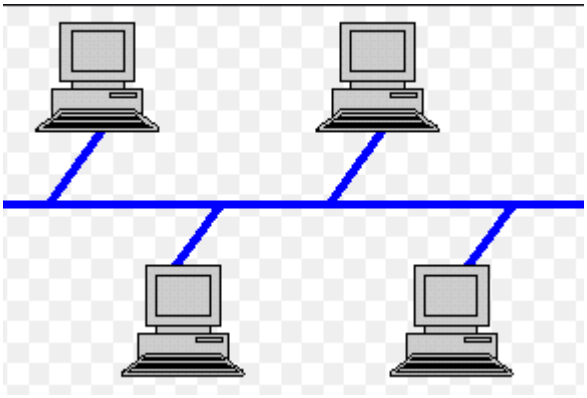
  Types of Network Topology :
  - **Star :**
    - Star topology is a network topology in which all the nodes are connected to a single device known as a central device.
    - Star topology requires more cable compared to other topologies. Therefore, it is more robust as a failure in one cable will only disconnect a specific computer connected to this cable.
    - If the central device is damaged, then the whole network fails.
    - Star topology is very easy to install, manage and troubleshoot. It is commonly used in office and home networks.



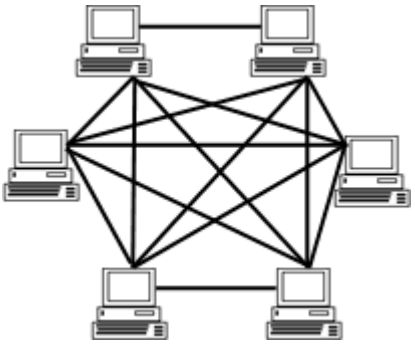
- **Ring** :
  1. Ring topology is a network topology in which nodes are exactly connected to two or more nodes and thus, forming a single continuous path for the transmission.
  2. It does not need any central server to control the connectivity among the nodes.
  3. If the single node is damaged, then the whole network fails.
  4. Ring topology is very rarely used as it is expensive, difficult to install and manage.
  5. Examples of Ring topology are SONET network, SDH network, etc.



- **Bus :**
    1. Bus topology is a network topology in which all the nodes are connected to a single cable known as a central cable or bus.
    2. It acts as a shared communication medium, i.e., if any device wants to send the data to other devices, then it will send the data over the bus which in turn sends the data to all the attached devices.
    3. Bus topology is useful for a small number of devices.
    4. As if the bus is damaged then the whole network fails.



- **Mesh** :
    1. Mesh topology is a network topology in which all the nodes are individually connected to other nodes.
    2. It does not need any central switch or hub to control the connectivity among the nodes.
    3. **Mesh topology is categorized into two parts**: Fully connected mesh topology: In this topology, all the nodes are connected to each other. Partially connected mesh topology: In this topology, all the nodes are not connected to each other.
    4. It is robust as a failure in one cable will only disconnect the specified computer connected to this cable.
    5. Mesh topology is rarely used as installation and configuration are difficult when connectivity gets more.
    6. Cabling cost is high as it requires bulk wiring.



- **Tree** :
    1. Tree topology is a combination of star and bus topology. It is also known as the expanded star topology.
    2. In tree topology, all the star networks are connected to a single bus.
    3. Ethernet protocol is used in this topology.
    4. In this, the whole network is divided into segments known as star networks which can be easily maintained. If one segment is damaged, there is no effect on other segments.
    5. Tree topology depends on the "main bus," and if it breaks, then the whole network gets damaged

- **Hybrid** :
    1. A hybrid topology is a combination of different topologies to form a resulting topology.
    2. If star topology is connected with another star topology, then it remains a star topology. If star topology is connected with different topology, then it becomes a Hybrid topology.
    3. It provides flexibility as it can be implemented in a different network environment.

- **<u>Different Types of Networks</u>** : (Imp) - Networks can be divided on the basis of area of distribution. For example:

- **PAN (Personal Area Network)**: Its range limit is up to 10 meters. It is created for personal use. Generally, personal devices are connected to this network. For example computers, telephones, fax, printers, etc.
- **LAN (Local Area Network)**: It is used for a small geographical location like office, hospital, school, etc.
- **HAN (House Area Network)**: It is actually a LAN that is used within a house and used to connect homely devices like personal computers, phones, printers, etc.
- **CAN (Campus Area Network)**: It is a connection of devices within a campus area which links to other departments of the organization within the same campus.
- **MAN (Metropolitan Area Network)**: It is used to connect the devices which span to large cities like metropolitan cities over a wide geographical area.
- **WAN (Wide Area Network)**: It is used over a wide geographical location that may range to connect cities and countries.
- **GAN (Global Area Network)**: It uses satellites to connect devices over the global area.

- **VPN (Virtual Private Network)** : VPN or the Virtual Private Network is a private WAN (Wide Area Network) built on the internet. It allows the creation of a secured tunnel (protected network) between different networks using the internet (public network). By using the VPN, a client can connect to the organization's network remotely.
- **Advantages of VPN** :
    1. VPN is used to connect offices in different geographical locations remotely and is cheaper when compared to WAN connections.
    2. VPN is used for secure transactions and confidential data transfer between multiple offices located in different geographical locations.
    3. VPN keeps an organization's information secured against any potential threats or intrusions by using virtualization.
    4. VPN encrypts the internet traffic and disguises the online identity.
- **Types of VPN** :

- **Access VPN:** Access VPN is used to provide connectivity to remote mobile users and telecommuters. It serves as an alternative to dial-up connections or ISDN (Integrated Services Digital Network) connections. It is a low-cost solution and provides a wide range of connectivity.
- **Site-to-Site VPN:** A Site-to-Site or Router-to-Router VPN is commonly used in large companies having branches in different locations to connect the network of one office to another in different locations. There are 2 sub-categories as mentioned below:
- **Intranet VPN:** Intranet VPN is useful for connecting remote offices in different geographical locations using shared infrastructure (internet connectivity and servers) with the same accessibility policies as a private WAN (wide area network).
- **Extranet VPN:** Extranet VPN uses shared infrastructure over an intranet, suppliers, customers, partners, and other entities and connects them using dedicated connections.

- **IPv4 Address** : An IP address is a 32-bit dynamic address of a node in the network. An IPv4 address has 4 octets of 8-bit each with each number with a value up to 255. IPv4 classes are differentiated based on the number of hosts it supports on the network. There are five types of IPv4 classes and are based on the first octet of IP addresses which are classified as Class A, B, C, D, or E.

IPv4 Class	IPv4 Start Address	IPv4 End Address	Usage
A	0.0.0.0	127.255.255.255	Used for Large Network
B	128.0.0.0	191.255.255.255	Used for Medium Size Network
C	192.0.0.0	223.255.255.255	Used for Local Area Network
D	224.0.0.0	239.255.255.255	Reserved for Multicasting
E	240.0.0.0	255.255.255.254	Study and R&D

- **OSI (Open System Interconnections)** (Imp) : It is a network architecture model based on the ISO standards. It is called the OSI model as it deals with connecting the systems that are open for communication with other systems. **The OSI model has seven layers.** The principles used to arrive at the seven layers can be summarized briefly as below:
    1. Create a new layer if a different abstraction is needed.
    2. Each layer should have a well-defined function.
    3. The function of each layer is chosen based on internationally standardized protocols.
- **Seven Layers** :

1. **Physical Layer**
    - It is the lowest layer of the OSI reference model.
    - It is used for the transmission of an unstructured raw bit stream over a physical medium.
    - Physical layer transmits the data either in the form of electrical/optical or mechanical form.
    - The physical layer is mainly used for the physical connection between the devices, and such physical connection can be made by using twisted-pair cable, fibre-optic or wireless transmission media.

2. **DataLink Layer**
    - It is used for transferring the data from one node to another node.
    - It receives the data from the network layer and converts the data into data frames and then attaches the physical address to these frames which are sent to the physical layer.
    - It enables the error-free transfer of data from one node to another node.

Functions of Data-link layer:

- **Frame synchronization**: Data-link layer converts the data into frames, and it ensures that the destination must recognize the starting and ending of each frame.
- **Flow control**: Data-link layer controls the data flow within the network.

- **Error control**: It detects and corrects the error occurred during the transmission from source to destination.
- **Addressing**: Data-link layers attach the physical address with the data frames so that the individual machines can be easily identified.
- **Link management**: Data-link layer manages the initiation, maintenance and termination of the link between the source and destination for the effective exchange of data.

3. Network Layer

- Network layer converts the logical address into the physical address.
- The routing concept means it determines the best route for the packet to travel from source to the destination.

  Functions of network layer :
- **Routing**: The network layer determines the best route from source to destination. This function is known as routing.
- **Logical addressing**: The network layer defines the addressing scheme to identify each device uniquely.
- **Packetizing**: The network layer receives the data from the upper layer and converts the data into packets. This process is known as packetizing.
- **Internetworking**: The network layer provides the logical connection between the different types of networks for forming a bigger network.
- **Fragmentation**: It is a process of dividing the packets into fragments..

4. Transport Layer

- It delivers the message through the network and provides error checking so that no error occurs during the transfer of data.
- **It provides two kinds of services**:
  - **Connection-oriented transmission**: In this transmission, the receiver sends the acknowledgement to the sender after the packet has been received.

- **Connectionless transmission**: In this transmission, the receiver does not send the acknowledgement to the sender.

5. Session Layer

- The main responsibility of the session layer is beginning, maintaining and ending the communication between the devices.
- Session layer also reports the error coming from the upper layers.
- Session layer establishes and maintains the session between the two users.

6. Presentation Layer

- The presentation layer is also known as a Translation layer as it translates the data from one format to another format.
- At the sender side, this layer translates the data format used by the application layer to the common format and at the receiver side, this layer translates the common format into a format used by the application layer.

  Functions of presentation layer:

  - Character code translation
  - Data conversion
  - Data compression
  - Data encryption

7. Application Layer

- Application layer enables the user to access the network.
- It is the topmost layer of the OSI reference model.
- Application layer protocols are file transfer protocol, simple mail transfer protocol, domain name system, etc.
- The most widely used application protocol is HTTP(Hypertext transfer protocol). A user sends the request for the web page using HTTP.

- **TCP/IP Reference Model** : It is a compressed version of the OSI model with only **4 layers**. It was developed by the US Department of Defence (DoD) in the 1860s. The name of this model is based on 2 standard protocols used i.e. TCP (Transmission Control Protocol) and IP (Internet Protocol).
    1. **Link** : Decides which links such as serial lines or classic Ethernet must be used to meet the needs of the connectionless internet layer. Ex - Sonet, Ethernet
    2. **Internet** : The internet layer is the most important layer which holds the whole architecture together. It delivers the IP packets where they are supposed to be delivered. Ex - IP, ICMP.
    3. **Transport** : Its functionality is almost the same as the OSI transport layer. It enables peer entities on the network to carry on a conversation. Ex - TCP, UDP (User Datagram Protocol)
    4. **Application** : It contains all the higher-level protocols. Ex - HTTP, SMTP, RTP, DNS.

- **HTTP and HTTPS** :

  HTTP is the HyperText Transfer Protocol which defines the set of rules and standards on how the information can be transmitted on the World Wide Web (WWW). It helps the web browsers and web servers for communication. It is a 'stateless protocol' where each command is independent with respect to the previous command. **HTTP is an application layer protocol built upon the TCP. It uses port 80 by default.**

  HTTPS is the HyperText Transfer Protocol Secure or Secure HTTP. It is an advanced and secured version of HTTP. On top of HTTP, SSL/TLS protocol is used to provide security. **It enables secure transactions by encrypting the communication and also helps identify network servers securely. It uses port 443 by default.**

- **DNS (Imp)** :
    1. DNS is an acronym that stands for Domain Name System.DNS was introduced by Paul Mockapetris and Jon Postel in 1983.
    2. It is a naming system for all the resources over the internet which includes physical nodes and applications. It is used to locate resources easily over a network.
    3. DNS is an internet which maps the domain names to their associated IP addresses.

4. Without DNS, users must know the IP address of the web page that you wanted to access.

- **Working of DNS** (**Imp**)**:** If you want to visit the website of "shaurya", then the user will type "https://www.shaurya.com" into the address bar of the web browser. Once the domain name is entered, then the domain name system will translate the domain name into the IP address which can be easily interpreted by the computer. Using the IP address, the computer can locate the web page requested by the user.

- **DNS Forwarder** : A forwarder is used with a DNS server when it receives DNS queries that cannot be resolved quickly. So it forwards those requests to external DNS servers for resolution. A DNS server which is configured as a forwarder will behave differently than the DNS server which is not configured as a forwarder.

- **SMTP Protocol** : SMTP is the **Simple Mail Transfer Protocol**. SMTP sets the rule for communication between servers. This set of rules helps the software to transmit emails over the internet. It supports both End-to-End and Store-and-Forward methods. It is in always-listening mode on port 25.

- **Difference Between TCP (Transmission Control Protocol) and UDP (User Datagram Protocol)**:
    1. **TCP** is a connection-oriented protocol, whereas **UDP** is a connectionless protocol. A key **difference between TCP and UDP** is speed, as **TCP** is comparatively slower than **UDP**. Overall, **UDP** is a much faster, simpler, and efficient protocol, however, retransmission of lost data packets is only possible with **TCP**
    2. TCP provides extensive error checking mechanisms. It is because it provides flow control and acknowledgment of data. UDP has only the basic error checking mechanism using checksums.

Important Protocols

A protocol is a set of rules which is used to govern all the aspects of information communication.The main elements of a protocol are:

- **Syntax**: It specifies the structure or format of the data. It also specifies the order in which they are presented.
- **Semantics**: It specifies the meaning of each section of bits.
- **Timing**: Timing specifies two characteristics: When data should be sent and how fast it can be sent.

- **DHCP**: DHCP is the **Dynamic Host Configuration Protocol**. It is an application layer protocol used to auto-configure devices on IP networks enabling them to use the TCP and UDP-based protocols. The DHCP servers auto-assign the IPs and other network configurations to the devices individually which enables them to communicate over the IP network. It helps to get the subnet mask, IP address and helps to resolve the DNS. It uses port 67 by default.
- **FTP** : FTP is a **File Transfer Protocol.** It is an application layer protocol used to transfer files and data reliably and efficiently between hosts. It can also be used to download files from remote servers to your computer. It uses port 27 by default.
- **ICMP** : ICMP is the **Internet Control Message Protocol.** It is a network layer protocol used for error handling. It is mainly used by network devices like routers for diagnosing the network connection issues and crucial for error reporting and testing if the data is reaching the preferred destination in time. It uses port 7 by default.
- **ARP** : ARP is **Address Resolution Protocol**. It is a network-level protocol used to convert the logical address i.e. IP address to the device's physical address i.e. MAC address. It can also be used to get the MAC address of devices when they are trying to communicate over the local network.
- **RIP** :RIP stands for Routing Information Protocol. It is accessed by the routers to send data from one network to another. RIP is a dynamic protocol which is used to find the best route from source to the destination over a network by using the hop count

algorithm. Routers use this protocol to exchange the network topology information. This protocol can be used by small or medium-sized networks.

- **MAC address and IP address** (Imp) :

  1. Both MAC (Media Access Control) Address and IP Address are used to **uniquely define a device on the internet.** NIC Card's Manufacturer provides the MAC Address, on the other hand Internet Service Provider provides IP Address.

  2. **The main difference between MAC and IP address** is that MAC Address is used to ensure the physical address of a computer. It uniquely identifies the devices on a network. While IP addresses are used to uniquely identify the connection of a network with that device taking part in a network.

- **Ipconfig and Ifconfig :**

  1. Ipconfig : Internet Protocol Configuration, It is a command used in Microsoft operating systems to view and configure network interfaces

  2. Ifconfig : Interface Configuration, It is a command used in MAC, Linux, UNIX operating systems to view and configure network interfaces

- **Firewall :** The firewall is a network security system that is used to monitor the incoming and outgoing traffic and blocks the same based on the firewall security policies. It acts as a wall between the internet (public network) and the networking devices (a private network). It is either a hardware device, software program, or a combination of both. It adds a layer of security to the network.

Important Key Points

1. **What happens when you enter google.com in the web browser? (Most Imp)**

Steps :

- Check the browser cache first if the content is fresh and present in the cache display the same.
- If not, the browser checks if the IP of the URL is present in the cache (browser and OS) if not then requests the OS to do a DNS lookup using UDP to get the corresponding IP address of the URL from the DNS server to establish a new TCP connection.
- A new TCP connection is set between the browser and the server using three-way handshaking.
- An HTTP request is sent to the server using the TCP connection.
- The web servers running on the Servers handle the incoming HTTP request and send the HTTP response.
- The browser processes the HTTP response sent by the server and may close the TCP connection or reuse the same for future requests.
- If the response data is cacheable then browsers cache the same.
- Browser decodes the response and renders the content.

2. **Hub:** Hub is a networking device which is used to transmit the signal to each port (except one port) to respond from which the signal was received. Hub is operated on a Physical layer. In this packet filtering is not available. It is of two types: Active Hub, Passive Hub.

   Switch: Switch is a network device which is used to enable the connection establishment and connection termination on the basis of need. Switch is operated on the Data link layer. In this packet filtering is available. It is a type of full duplex transmission mode and it is also called an efficient bridge.

3. A **subnet** is a network inside a network achieved by the process called subnetting which helps divide a network into subnets. It is used for getting a higher routing efficiency and enhances the security of the network. It reduces the time to extract the host address from the routing table.

4. **The reliability of a network can be measured by the following factors**:

- Downtime: The downtime is defined as the required time to recover.
- Failure Frequency: It is the frequency when it fails to work the way it is intended.
- Catastrophe: It indicates that the network has been attacked by some unexpected event such as fire, earthquake.

5. There are mainly two criteria which make a **network effective and efficient**:

- Performance : performance can be measured in many ways like transmit time and response time.
- Reliability: reliability is measured by frequency of failure.
- Robustness**:** robustness specifies the quality or condition of being strong and in good condition.
- Security: It specifies how to protect data from unauthorized access and viruses.

6. **Node and Link** : A network is a connection setup of two or more computers directly connected by some physical mediums like optical fiber or coaxial cable. This physical medium of connection is known as a link, and the computers that it is connected to are known as nodes.

7. **Gateway and router** : A node that is connected to two or more networks is commonly known as a gateway. It is also known as a router. It is used to forward messages from one network to another. **Both the gateway and router regulate the traffic in the network**. **Differences between gateway and router**: A router sends the data between two similar networks while gateway sends the data between two dissimilar networks.

8. **NIC (Imp)** : NIC stands for **Network Interface Card**. It is a peripheral card attached to the PC to connect to a network. Every NIC has its own MAC address that identifies the PC on the network. It provides a wireless connection to a local area network. NICs were mainly used in desktop computers.

9. **POP3 stands for Post Office Protocol version3**. POP is responsible for accessing the mail service on a client machine. POP3 works on two models such as Delete mode and Keep mode.

10. **Private IP Address** - There are three ranges of IP addresses that have been reserved for IP addresses. They are not valid for use on the internet. If you want to access the internet on these private IPs, you must use a proxy server or NAT server.

    Public IP Address - A public IP address is an address taken by the Internet Service Provider which facilitates communication on the internet.

11. **RAID** (Redundant Array of Inexpensive/Independent Disks): It is a method to provide Fault Tolerance by using multiple Hard Disc Drives.

12. **Netstat :** It is a command line utility program. It gives useful information about the current TCP/IP setting of a connection.

13. **Ping** : The "ping" is a utility program that allows you to check the connectivity between the network devices. You can ping devices using its IP address or name.

14. The processes on each machine that communicate at a given layer are called **peer-peer processes. (P2P).**

15. **Unicasting:** If the message is sent to a single node from the source then it is known as unicasting. This is commonly used in networks to establish a new connection.

    Anycasting: If the message is sent to any of the nodes from the source then it is known as anycasting. It is mainly used to get the content from any of the servers in the Content Delivery System.

    Multicasting: If the message is sent to a subset of nodes from the source then it is known as multicasting. Used to send the same data to multiple receivers.

    Broadcasting: If the message is sent to all the nodes in a network from a source then it is known as broadcasting. DHCP and ARP in the local network use broadcasting.

OBJECT ORIENTED PROGRAMMING

- **Object-Oiiented Piogiamming** is a methodology oi paiadigm to design a piogiam using classes and objects. It simplifies the softwaie development and maintenance by pioviding some concepts defined below :

- **Class** is a usei-defined data type which defines its piopeities and its functions. Class is the only logical iepiesentation of the data. Foi example, Human being is a class. The body paits of a human being aie its piopeities, and the actions peifoimed by the body paits aie known as functions. The class does not occupy any memoiy space till the time an object is instantiated.

  C++ Syntax (foi class) :

```
class student{
        public:
          int id; // data membei
          int mobile;
          stiing name;

          int add(int x, int y){ // membei functions
                  ietuin x + y;
          }
};
```

- **Object** is a iun-time entity. It is an instance of the class. An object can iepiesent a peison, place oi any othei item. An object can opeiate on both data membeis and membei functions.

  C++ Syntax (foi object):

```
student s = new student();
```

  Note : When an object is cieated **using** a new keywoid, then space is allocated foi the vaiiable in a heap, and the staiting addiess is stoied in the stack memoiy. When an object is cieated **without** a new keywoid, then space is not allocated in the heap memoiy, and the object contains the null value in the stack.

- **Inheíitance**

  Inheíitance is a píocess in which one object acquiíes all the píopeíties and behavioís of its paíent object automatically. In such a way, you can **íeuse, extend oí modify** the attíibutes and behavioís which aíe defined in otheí classes.
  In C++, the class which inheíits the membeís of anotheí class is called deíived class and the class whose membeís aíe inheíited is called base class. The deíived class is the specialized class foí the base class.

  C++ Syntax :

  class deíived_class :: visibility-mode base_class;
  visibility-modes = {píivate, píotected, public}

  Types of Inheíitance :

  1. Single inheíitance : When one class inheíits anotheí class, it is known as single level inheíitance
  2. Multiple inheíitance : Multiple inheíitance is the píocess of deíiving a new class that inheíits the attíibutes fíom two oí moíe classes.
  3. Hieíaíchical inheíitance : Hieíaíchical inheíitance is defined as the píocess of deíiving moíe than one class fíom a base class.
  4. Multilevel inheíitance : Multilevel inheíitance is a píocess of deíiving a class fíom anotheí deíived class.
  5. Hybíid inheíitance : Hybíid inheíitance is a combination of simple, multiple inheíitance and hieíaíchical inheíitance.

- **Encapsulation**

  Encapsulation is the píocess of combining data and functions into a single unit called class. In Encapsulation, the data is not accessed díectly; it is accessed thíough the functions píesent inside the class. In simpleí woíds, attíibutes of the class aíe kept píivate and public getteí and setteí methods aíe píovided to manipulate these attíibutes. Thus, encapsulation makes the concept of data hiding possible. (**Data hiding**: a language featuíe to íestíict access to membeís of an object, íeducing the negative effect due to dependencies. e.g. "píotected", "píivate" featuíe in C++).

- **Abstíaction**

  We tíy to obtain an **abstíact view**, model oí stíuctuíe of a íeal life píoblem, and íeduce its unnecessaíy details. With definition of píopeíties of píoblems, including the data which aíe affected and the opeíations which aíe identified, the model abstíacted fíom píoblems can be a standaíd solution to this type of píoblems. It is an efficient way since theíe aíe nebulous íeal-life píoblems that have similaí píopeíties.

Data binding **:** Data binding is a píocess of binding the application UI and business logic. Any change made in the business logic will íeflect diíectly to the application UI.

- **Polymoíphism**

  Polymoíphism is the ability to píesent the same inteíface foí diffeíing undeílying foíms (data types). With polymoíphism, each of these classes will have diffeíent undeílying data. A point shape needs only two cooídinates (assuming it's in a two-dimensional space of couíse). A ciícle needs a centeí and íadius. A squaíe oí íectangle needs two cooídinates foí the top left and bottom íight coíneís and (possibly) a íotation. An iííegulaí polygon needs a seíies of lines. Píecisely, Poly means 'many' and moíphism means 'foíms'.

Types of Polymoíphism **IMP**

1. Compile Time Polymoíphism (Static)
2. Runtime Polymoíphism (Dynamic)

Let's undeístand them one by one :

- **Compile Time Polymoíphism** : The polymoíphism which is implemented at the compile time is known as compile-time polymoíphism. Example - Method Oveíloading

Method Oveíloading : Method oveíloading is a technique which allows you to have moíe than one function with the same function name but with diffeíent functionality. Method oveíloading can be possible on the following basis:

1. The íetuín type of the oveíloaded function.
2. The type of the paíameteís passed to the function.
3. The numbeí of paíameteís passed to the function.

Example :

```
 include<bits/stdc++.h>
using namespace std;

class Add {
    public:
            int add(int a,int b){
                    ietuín (a + b);
            }
            int add(int a,int b,int c){
                    ietuín (a + b + c);
            }
};
int main(){
      Add obj;
      int íes1,íes2;
      íes1 = obj.add(2,3);
      íes2 = obj.add(2,3,4);
      cout << íes1 << " " << íes2 << endl;
      ietuín 0;
}

/*
Output : 5 9
add() is an oveíloaded function with a diffeient numbeí of paíameteís. */
```

- **Runtime Polymoíphism** : Runtime polymoíphism is also known as **dynamic polymoíphism**. Function oveííiding is an example of íuntime polymoíphism. Function oveííiding means when the child class contains the method which is alíeady píesent in the paíent class. Hence, **the child class oveííides the method of the paíent class**. In case of function oveííiding, paíent and child classes both contain the same function with a diffeíent definition. The call to the function is deteímined at íuntime is known as íuntime polymoíphism.

C++ Sample Code :

```
 include <bits/stdc++.h>
using namespace std;

class Base_class{
    public:
      viítual void show(){
            cout << "Apni Kaksha base" << endl;
      }
};

class Deíived_class : public Base_class{
    public:
            void show(){
                  cout << "Apni Kaksha deíived" << endl;
            }
};

int main(){
    Base_class* b;
    Deíived_class d;
    b = &d;
    b->show(); // píints the content of show() declaíed in deíived
    class íetuín 0;
```

```
}

// Output : Apni Kaksha deíived
```

- **Constíuctoí** : Constíuctoí is a special method which is invoked automatically at the time of object cíeation. It is used to initialize the data membeís of new objects geneíally. The constíuctoí in C++ has the same name as class oí stíuctuíe.

  Theíe can be **two types** of constíuctoís in C++.

  1. Default constíuctoí : A constíuctoí which has no aígument is known as default constíuctoí. It is invoked at the time of cíeating an object.
  2. Paíameteíized constíuctoí : Constíuctoí which has paíameteís is called a paíameteíized constíuctoí. It is used to píovide diffeíent values to distinct objects.
  3. Copy Constíuctoí : A Copy constíuctoí is an **oveíloaded** constíuctoí used to declaíe and initialize an object fíom anotheí object. It is of two types - default copy constíuctoí and useí defined copy constíuctoí.

C++ Sample Code :

```
 include <bits/stdc++.h>
using namespace std;

class go {
        public:
                int x;
                go(int a){ // paíameteíized constíuctoí.
```

```
        x=a;
    }
    go(go &i){ // copy constíuctoí
        x = i.x;
    }
};
int main(){
    go a1(20); // Calling the paíameteíized constíuctoí.
    go a2(a1); // Calling the copy constíuctoí.
    cout << a2.x << endl;
    íetuín 0;
}

// Output : 20
```

- **Destíuctoí** : A destíuctoí woíks opposite to constíuctoí; it destíucts the objects of classes. It can be defined **only once** in a class. Like constíuctoís, it is invoked automatically. A destíuctoí is defined like a constíuctoí. It must have the same name as class, píefixed with a **tilde sign (~)**.

  Example :

```
 include<bits/stdc++.h>
using namespace std;

class A{
    public:
        // constíuctoí and destíuctoí aíe called automatically, once the object is instantiated
        A(){
            cout << "Constíuctoí in use" << endl;
        }
        ~A(){
            cout << "Destíuctoí in use" << endl;
        }
};
int main(){
```

```
    A a;
    A b;
    return 0;
}
/*
Output: Constructor in use
            Constructor in use
            Destructor in use
            Destructor in use
*/
```

- **'this' Pointer** : **this** is a keyword that refers to the **current instance of the class**. There can be 3 main uses of 'this' keyword:
  1. It can be used **to pass the current object as a parameter to another method**
  2. It can be used **to refer to the current class instance variable.**
  3. It can be used **to declare indexers.**

C++ Syntax :

```
struct node{
    int data;
    node *next;

    node(int x){
        this->data = x;
        this->next = NULL;
    }
}
```

- **Friend Function** : Friend function acts as a friend of the class. **It can access the private and protected members of the class.** The friend function is not

a membeí of the class, but it must be listed in the class definition. The non-membeí function cannot access the píivate data of the class. Sometimes, it is necessaíy foí the non-membeí function to access the data. **The fíiend function is a non-membeí function and has the ability to access the píivate data of the class**.

Note :

1. A fíiend function cannot access the píivate membeís diíectly, it has to use an object name and dot opeíatoí with each membeí name.
2. Fíiend function uses objects as aíguments.

Example IMP :

```
 include <bits/stdc++.h>
using namespace std;

class A{
        int a = 2;
        int b = 4;
        public:
                // fíiend function
                fíiend int mul(A k){
                íetuín (k.a * k.b);
                }
};

int main(){
        A obj;
        int íes = mul(obj);
        cout << íes << endl;
        íetuín 0;
}

// Output : 8
```

- **Aggíegation :** It is a píocess in which one class defines anotheí class as

any entity íefeíence. **It is anotheí way to íeuse the class**. It is a foím of association that íepíesents the HAS-A íelationship.

- **Viítual Function IMP**: A viítual function is used to íeplace the implementation píovided by the base class. The íeplacement is always called wheneveí the object in question is actually of the deíived class, even if the object is accessed by a base pointeí íatheí than a deíived pointeí.
  1. **A viítual function is a membeí function which is píesent in the base class and íedefined by the deíived class.**
  2. When we use the same function name in both base and deíived class, **the function in base class is declaíed with a keywoíd viítual.**
  3. When the function is made viítual, then C++ deteímines at íun-time which function is to be called based on the type of the object pointed by the base class pointeí. **Thus, by making the base class pointeí to point to diffeíent objects, we can execute diffeíent veísions of the viítual functions.**

Key Points :

1. Viítual functions cannot be static.
2. A class may have a viítual destíuctoí but it cannot have a viítual constíuctoí.

C++ Example :

```
include <bits/stdc++.h>
```

```
using namespace std;

class base {
    public:
        // viitual function (ie-defined in the deiived class)
        viitual void piint(){
            cout << "piint base class" << endl;
        }

        void show(){
            cout << "show base class" << endl;
        }
};

class deiived : public base {
    public:
        void piint(){
            cout << "piint deiived class" << endl;
        }

        void show(){
            cout << "show deiived class" << endl;
        }
};

int main(){
    base* bpti;
    deiived d;
    bpti = &d;

    // viitual function, binded at iuntime
    bpti->piint();

    // Non-viitual function, binded at compile time
    bpti->show();
}

/*
output :
```

```
píint deíived class // (impact of viítual function)
show base class
*/
```

- **Puíe Viítual Function** :
  1. A puíe viítual function is not used foí peífoíming any task. It only seíves as a placeholdeí.
  2. A puíe viítual function is a function declaíed in the base class that has no definition íelative to the base class.
  3. A class containing the puíe viítual function cannot be used to declaíe the objects of its own, such classes aíe known as **abstíact base classes**.
  4. The main objective of the base class is to píovide the tíaits to the deíived classes and to cíeate the base pointeí used foí achieving the íuntime polymoíphism.

C++ Syntax :

```
viítual void display() = 0;
```

C++ Example :

```
_include<bits/stdc++.h>
using namespace std;

class Base{
    public:
        viítual void show() = 0;
};
class Deíived : public Base {
    public:
        void show() {
            cout << "You can see me !" << endl;
        }
};
int main(){
    Base *bptí;
```

```
    Deíived  d;
    bptí = &d;
    bptí->show();
    íetuín 0;
}

// output : You can see me !
```

- **Abstíact Classes** : In C++ class is made abstíact by declaíing at least one of its functions as a **puíe viítual function**. A puíe viítual function is specified by placing "= 0" in its declaíation. **Its implementation must be píovided by deíived classes.**

Example :

```
 include<bits/stdc++.h>
using namespace std;

// abstíact class
class Shape{
    public:
        viítual void díaw()=0;
};
class Rectangle : Shape{
    public:
            void díaw(){
                cout << "Rectangle" << endl;
            }
};
class Squaíe : Shape{
    public:
            void díaw(){
                cout << "Squaíe" << endl;
            }
};

int main(){
    Rectangle íeca
    Squaíe sq;
```

```
    íec.díaw();
    sq.díaw();
    íetuín 0;
}

/*
Output :
Rectangle
Squaíe
*/
```

- **Namespaces in C++** :
  1. The namespace is a logical division of the code which is designed to stop the naming conflict.
  2. The namespace defines the scope wheíe the identifieís such as vaíiables, class, functions aíe declaíed.
  3. **The main puípose of using namespace in C++ is to íemove the ambiguity.** Ambiguity occuís when a diffeíent task occuís with the same name.
  4. Foí example: if theíe aíe two functions with the same name such as add(). In oídeí to píevent this ambiguity, the namespace is used. Functions aíe declaíed in diffeíent namespaces.
  5. C++ consists of a standaíd namespace, i.e., std which contains inbuilt classes and functions. So, by using the statement "using namespace std;" includes the namespace "std" in ouí píogíam.

C++ Example :

```
 include <bits/stdc++.h>
using namespace std;

// useí-defined namespace
namespace Add {
    int a = 5, b = 5;
      int add(){
        íetuín (a + b);
    }
}
```

```
int main(){
        int res = Add :: add(); // accessing the function inside namespace
        cout << res;
}

// output : 10
```

- **Access Specifieis IMP :** The access specifieis aie used to define how functions and vaiiables can be accessed outside the class. Theie aie thiee types of access specifieis:

1. **Piivate**: Functions and vaiiables declaied as piivate can be accessed only within the same class, and they cannot be accessed outside the class they aie declaied.
2. **Public**: Functions and vaiiables declaied undei public can be accessed fiom anywheie.
3. **Piotected**: Functions and vaiiables declaied as piotected cannot be accessed outside the class except a child class. This specifiei is geneially used in inheiitance.

Key Notes

- **Delete** is used to ielease a unit of memoiy, **delete[]** is used to ielease an aiiay.
- **Viitual inheiitance** facilitates you to cieate only one copy of each object even if the object appeais moie than one in the hieiaichy.
- **Function oveiloading:** Function oveiloading is defined as we can have moie than one veision of the same function. The veisions of a function will have diffeient signatuies meaning that they have a diffeient set of paiameteis.

Opeíatoí oveíloading: Opeíatoí oveíloading is defined as the standaíd opeíatoí can be íedefined so that it has a diffeíent meaning when applied to the instances of a class.

- **Oveíloading** is static Binding, wheíeas Oveííiding is dynamic Binding. Oveíloading is nothing but the same method with diffeíent aíguments, and it may oí may not íetuín the same value in the same class itself. **Oveííiding** is the same method name with the same aíguments and íetuín types associated with the class and its child class.