UNIT-III

Operating Systems: Functions of operating systems, Types of operating systems, Device & resource management.
Database Management Systems: Data models, RDBMS, SQL, Database transactions, Data centers, cloud services.

Operating system:

An operating system is an interface between the computer's hardware and its user. The operating system is responsible for processing data, managing files, running applications and controlling peripheral devices such as input, output, discs and printers.

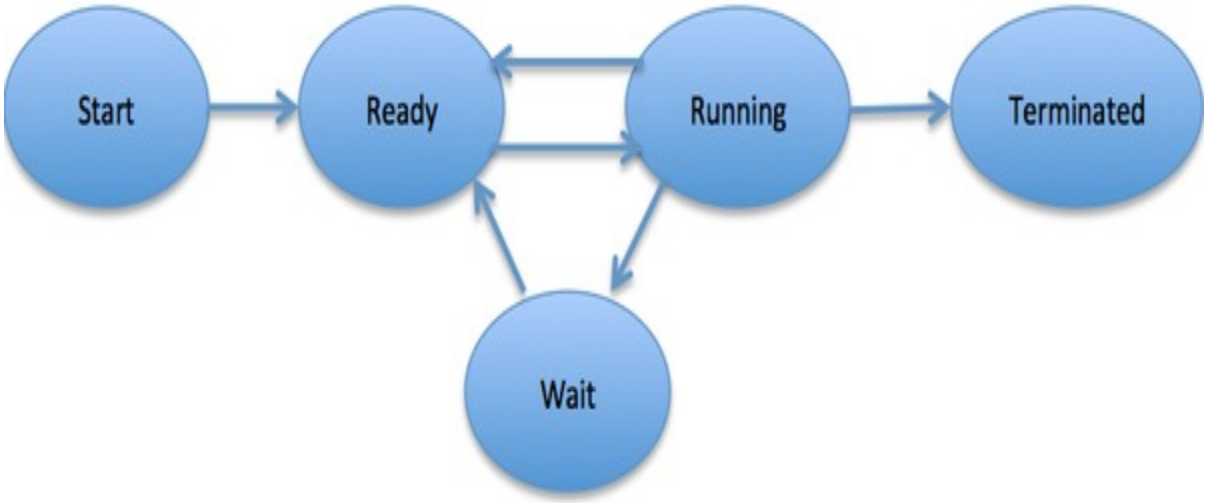
When we power on a computer, the switch sends electricity to the motherboard. If the power supply is adequate, the BIOS(Basic Input/Output System) chip takes over. It checks for the proper functioning of basic input and output devices and the system is booted. The process of loading the operating system into main memory is called booting.

Process:

When a process executes, it passes through different states. These stages may differ in different operating systems, and the names of these states are also not standardized.
In general, a process can have one of the following five states at a time.

S.N.	State & Description
1	Start This is the initial state when a process is first started/created.

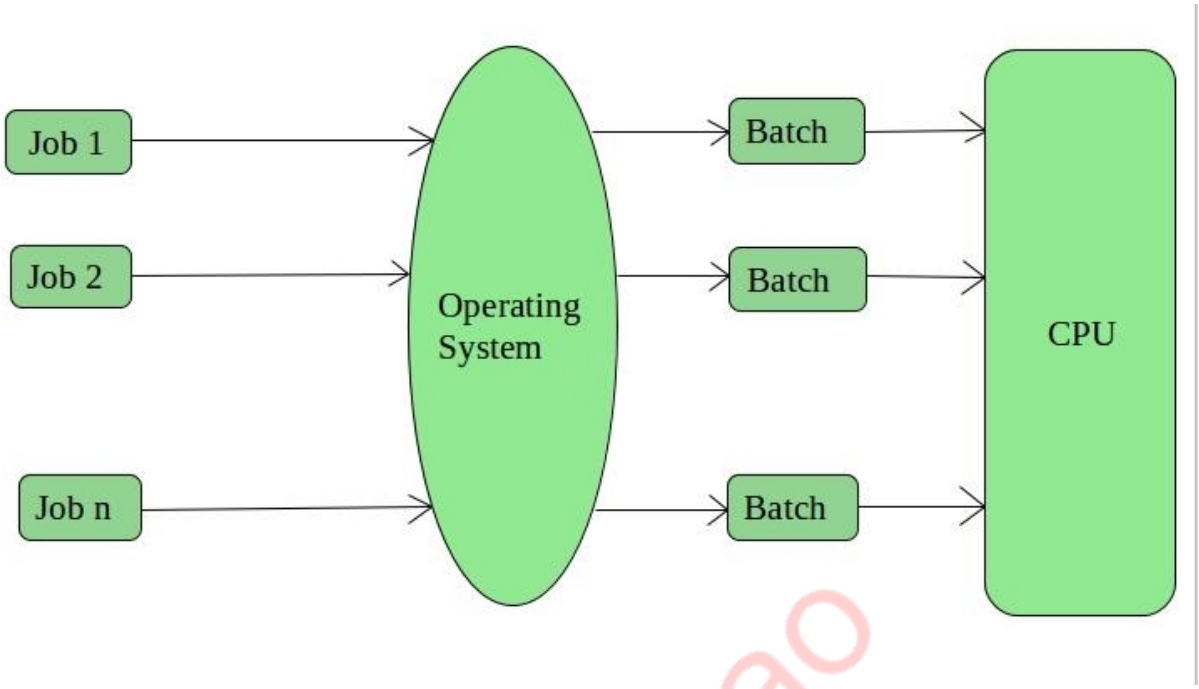
2	Ready The process is waiting to be assigned to a processor. Ready processes are waiting to have the processor allocated to them by the operating system so that they can run. Process may come into this state after Start state or while running it by but interrupted by the scheduler to assign CPU to some other process.
3	Running Once the process has been assigned to a processor by the OS scheduler, the process state is set to running and the processor executes its instructions.
4	Waiting Process moves into the waiting state if it needs to wait for a resource, such as waiting for user input, or waiting for a file to become available.
5	Terminated or Exit Once the process finishes its execution, or it is terminated by the operating system, it is moved to the terminated state where it waits to be removed from main memory.



TYPES OF OPERATING SYSTEMS:

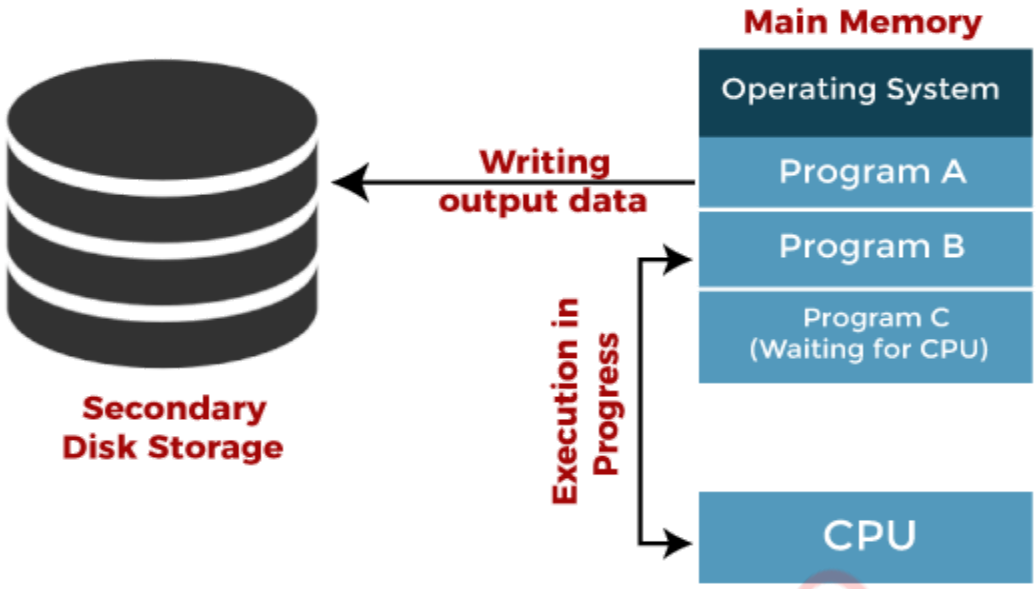
Batch Operating System:

In the 1970s, Batch processing was very popular. In this technique, similar types of jobs were batched together and executed in time. There is an operator which takes similar jobs having the same requirement and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs.



Multiprogramming Operating System:

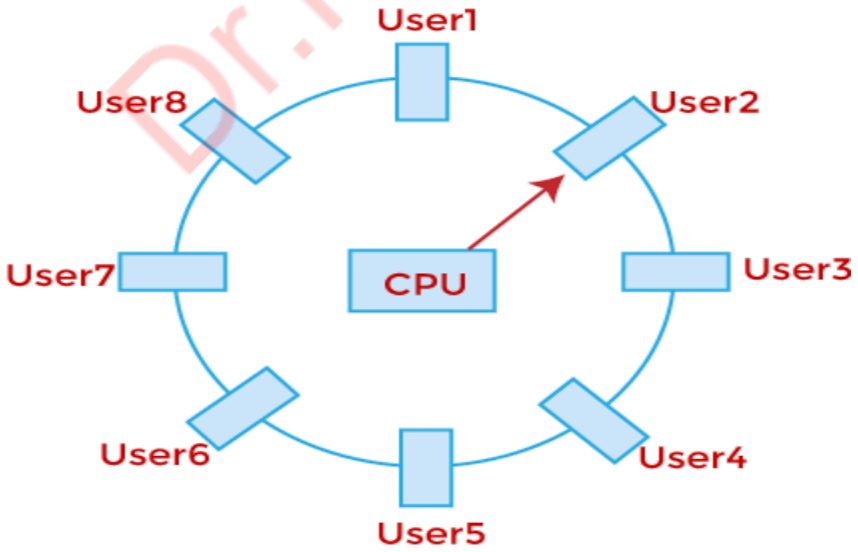
Multiprogramming is an extension to batch processing where the CPU is always kept busy. Each process needs two types of system time: CPU time and IO time.In a multiprogramming environment, when a process does its I/O, The CPU can start the execution of other processes. Therefore, multiprogramming improves the efficiency of the system.



Jobs in multiprogramming system

Time-Sharing Operating System

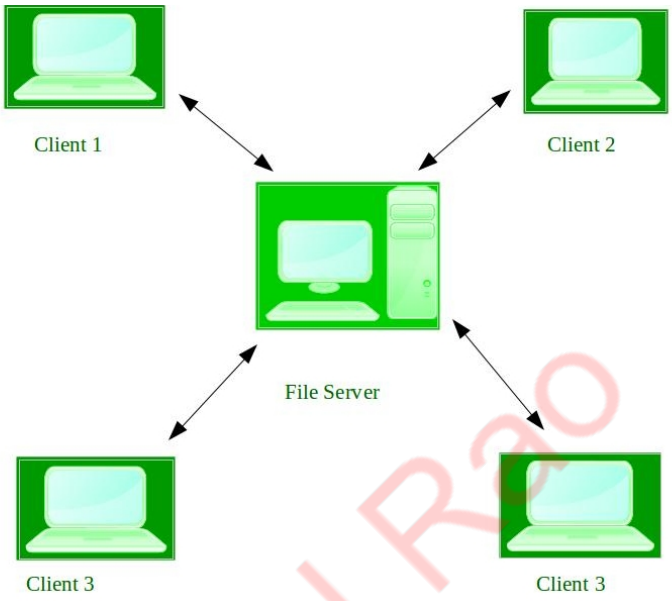
It is a logical extension of multiprogramming.In time-sharing, the CPU is switched among multiple programs given by different users on a scheduled basis.Thus it helps to provide a large number of user's direct access to the main computer.



Timesharing in case of 8 users

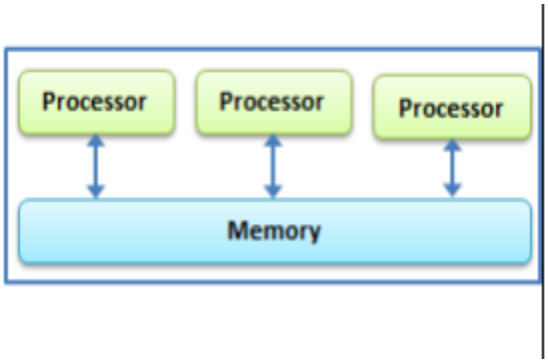
Network Operating System

An Operating system, which includes software and associated protocols to communicate with other computers via a network conveniently and cost-effectively, is called Network Operating System.



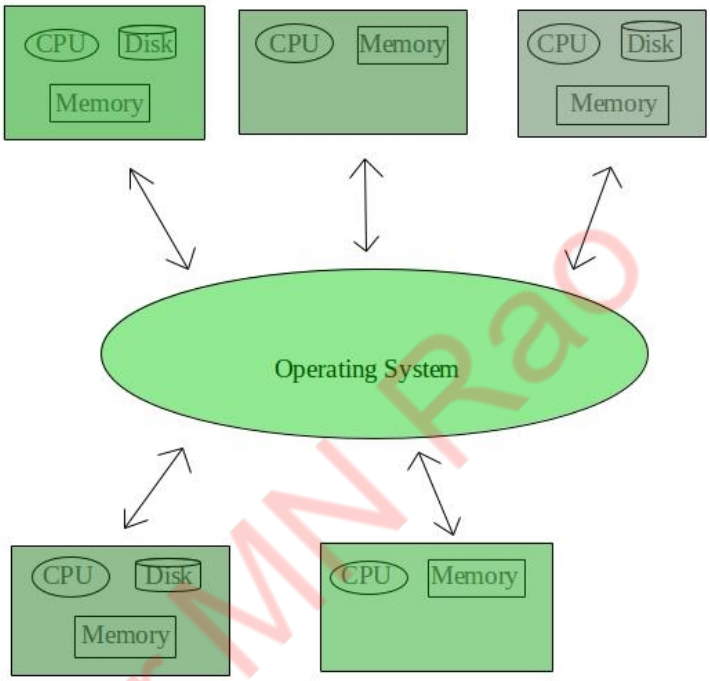
Parallel Operating System

Parallel operating system is one with more than one CPU connected closely which carries out the necessary activities in parallel. These multiple processors share memory, computer buses, and some peripheral devices; communication usually takes place through the shared memory, which can be accessed by numerous programs. It provides a schematic representation of a parallel system. These systems are also called tightly coupled systems because the processors share a common memory for communication.



Distributed Operating System

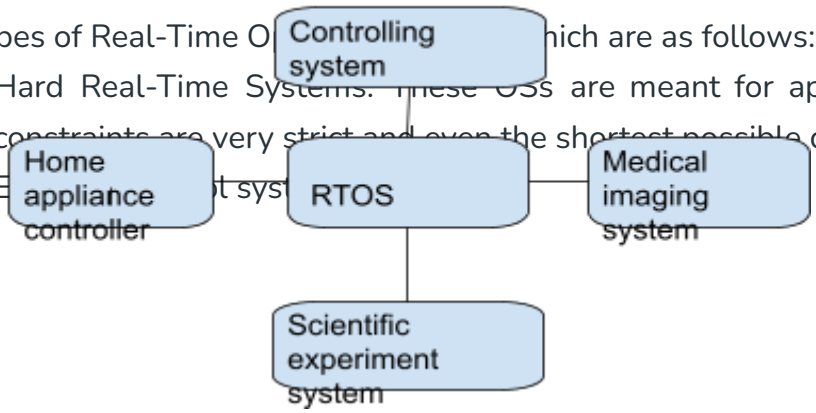
The Distributed Operating system is not installed on a single machine, it is divided into parts, and these parts are loaded on different machines. A part of the distributed Operating system is installed on each machine to make their communication possible. Distributed Operating systems are much more complex, large, and sophisticated than Network operating systems because they also have to take care of varying networking protocols.



Real Time Operating System

In Real-Time Systems, each job carries a certain deadline within which the job is supposed to be completed, otherwise, the huge loss will be there, or even if the result is produced, it will be completely useless.

Two types of Real-Time O hich are as follows:



- Hard Real-Time Systems: These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. E t sys

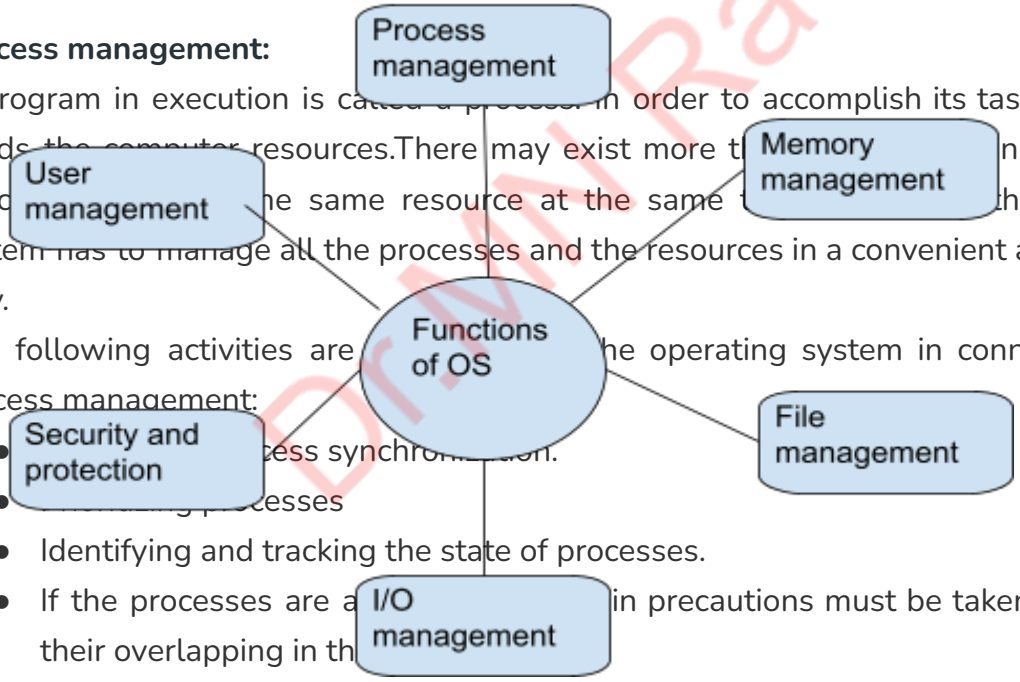
- Soft Real-Time Systems: These OSs are for applications where time-constraint is less strict. Ex: telephone switches.

Handheld systems:

Handheld operating systems are found in personal digital assistants(PDAs), smartphones and tablets connected to the internet. Android and iOS are the most popular handheld operating systems.

Functions of the Operating System:

The main task of the operating system is to provide services and allocate the resources required for the execution of a program. The various functions of operating systems are:



Process management:

A program in execution is called a process. In order to accomplish its task, a process needs the computer resources.There may exist more than one process in the system which may require the same resource at the same time. Therefore the operating system has to manage all the processes and the resources in a convenient and efficient way.

The following activities are carried out by the operating system in connection with process management:

- Process synchronization.
- Creating processes
- Identifying and tracking the state of processes.
- If the processes are accessing I/O devices, precautions must be taken to prevent their overlapping in the

Memory management:

Memory management is the functionality of an operating system which handles or manages primary memory and moves processes back and forth between main memory and disk during execution. Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It decides which process will

get memory at what time. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

File management:

The file management system is also known as the file system. It is responsible for file management in any system. The various functions involved in file management are as follows:

- It is responsible for creating new files in the computer system and placing them in specific locations.
- It is responsible for locating the existing files in the computer system.
- It facilitates keeping the files in separate folders known as directories. These directories allow users to quickly search for files or organize files based on their types of uses.
- It enables users to change the data of files or the name of files in directories.

Device and resource management:

Device management in an operating system means controlling the Input/Output devices like disk, microphone, keyboard, printer, magnetic tape, USB ports, camcorder, scanner, other accessories, and supporting units like supporting units control channels. A process may require various resources, including main memory, file access, and access to disk drives, and others. If resources are available, they could be allocated, and control returned to the CPU. Otherwise, the procedure would have to be postponed until adequate resources become available. The system has multiple devices, and in order to handle these physical or virtual devices, the operating system requires a separate program known as an ad device controller. It also determines whether the requested device is available.

Security and protection:

The operating system employs password protection and other similar mechanisms to safeguard user data.(Username/ Password, User Key/ User Card. User Attribute

Identification) By offering files and directories authorization of access rights, it also prevents unwanted access to applications and user data. Malware protection software or anti-virus software must be installed for external operation.

User interface:

The user and operating system are connected with each other with the help of interface, so interface is used to connect the user and OS. A user interface regulates how information is entered, how instructions are displayed on the screen, and how data is entered.

User interface comes in two ways:

- **Command-line interface:** The user enters the command or strikes the shortcut key on the keyboard in a command line interface
- **Graphical user interface:** GUI allows the user to interact with the computer through graphical icons and interactive menus.

DATABASE MANAGEMENT SYSTEMS

Syllabus: Data models, RDBMS, SQL, Database transactions, Data centers, cloud services.

DATA MODELS:

Data models define how the logical structure of a database is modeled.

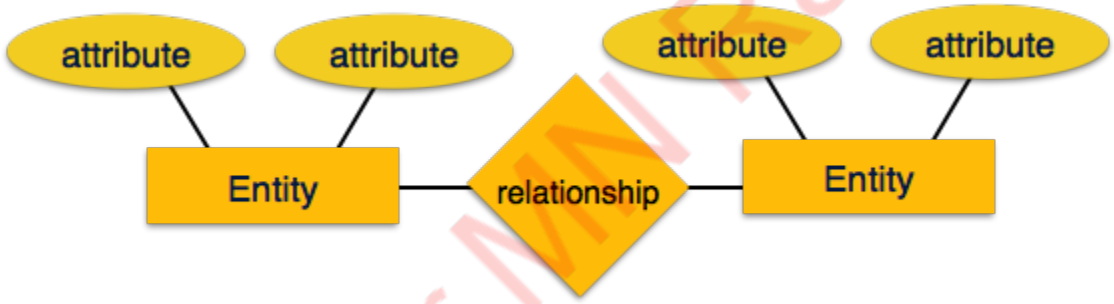
Entity-Relationship Model:

Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenarios into the database model, the ER Model creates entity set, relationship set, general attributes and constraints. ER Model is best used for the conceptual design of a database.

ER Model is based on –

- Entities and their *attributes*.
- Relationships among entities.

These concepts are explained below.



Entity – An entity in an ER Model is a real-world entity having properties called attributes. Every attribute is defined by its set of values called domain. For example, in a school database, a student is considered as an entity. Students have various attributes like name, age, class, etc.

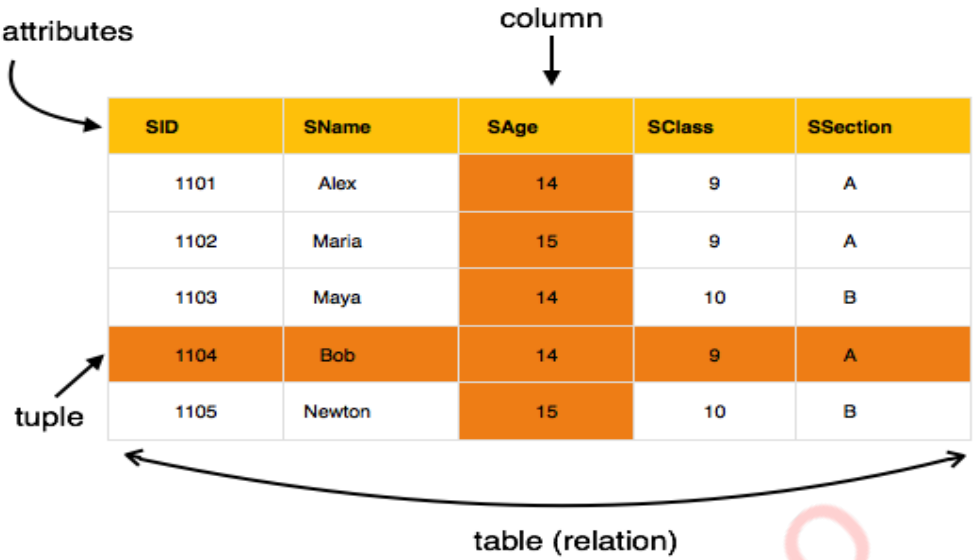
Relationship – The logical association among entities is called *relationship*. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of associations between two entities.

Mapping cardinalities –

- one to one
- one to many
- many to one
- many to many

Relational Model:

The most popular data model in DBMS is the Relational Model.This was introduced by EF Codd. This model represents the data in the form of tables and their relationships. Each table is described with its set of attributes.



.

The advantages of this model are -

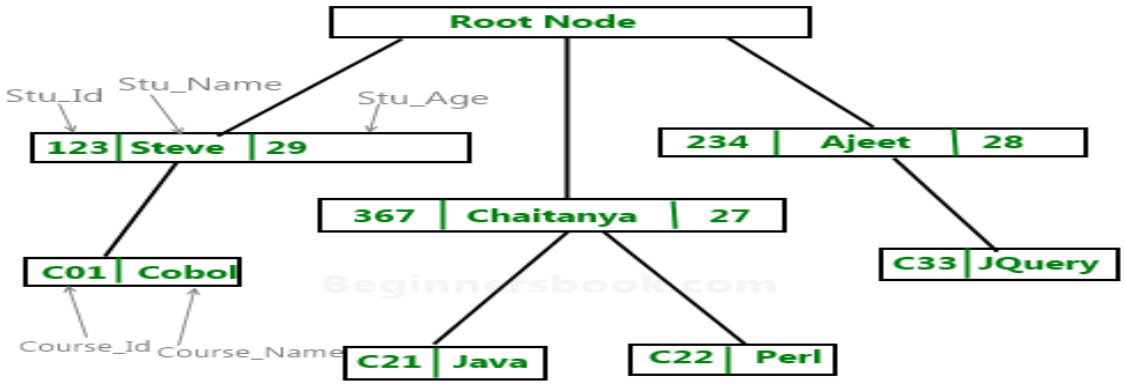
- The data available in a relational data model can be easily categorized into groups.
- The redundant data can be easily identified and removed.
- Access and manipulation of data can be simple and flexible.
- The relations among the table ensure that the data can be accessed by multiple users at the same time.

Hierarchical model

In a hierarchical **model**, data is organized into a tree-like structure with each record having one parent record and many children. The main drawback of this model is that it can have only one to many relationships between nodes.

Advantages:

- Fast addition , deletion, update and insertion operations.
- Since the data model has many relations with 1 to n cardinality, it is efficient.
- Top level data can be accessed easily.



Network Model

Network Model is the same as hierarchical model except that it has graph-like structure rather than a tree-based structure. Unlike hierarchical models, this model allows each record to have more than one parent record.

ADVANTAGES:

- They can represent complex data relationships better than hierarchical models.
- They can handle many relations
- Data access is optimized
- The database design is more structured

RDBMS:

RDBMS stands for Relational Database Management System. RDBMS is a program used to maintain a relational database. RDBMS is the basis for all modern database systems such as MySQL, Microsoft SQL Server, Oracle, and Microsoft Access. RDBMS uses SQL queries to access the data in the database.

A relational database defines database relationships in the form of tables. The tables are related to each other - based on data common to each.

Look at the following three tables "Customers", "Orders", and "Shippers" from the Northwind database:

Customers Table

CustomerID	CustomerName	ContactName	Address
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312
4	Around the Horn	Thomas Hardy	120 Hanover Sq.
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8

The relationship between the "Customers" table and the "Orders" table is the CustomerID column:

Orders Table

OrderID	CustomerID	EmployeeID	OrderDate	ShipperID
10278	5	8	1996-08-12	2
10280	5	2	1996-08-14	1
10308	2	7	1996-09-18	3
10355	4	6	1996-11-15	1
10365	3	3	1996-11-27	2
10383	4	8	1996-12-16	3
10384	5	3	1996-12-16	3

The relationship between the "Orders" table and the "Shippers" table is the ShipperID column:

Shippers Table

ShipperID	ShipperName	Phone
1	Speedy Express	(503) 555-9831
2	United Package	(503) 555-3199
3	Federal Shipping	(503) 555-9931

Advantages:

- Each table can be altered independently without disturbing the others.
- Data can be updated at a single location without making changes to several files.
- It allows for client-side architecture that stores numerous users together.
- It supports SQL language.
- It makes it easier to store and retrieve massive amounts of data.

SQL:

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database.

SQL is the standard language for Relational Database Systems. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Postgres and SQL Server use SQL as their standard database language.

SQL Process

When you are executing an SQL command for any RDBMS, the system determines the best way to carry out your request and SQL engine figures out how to interpret the task.

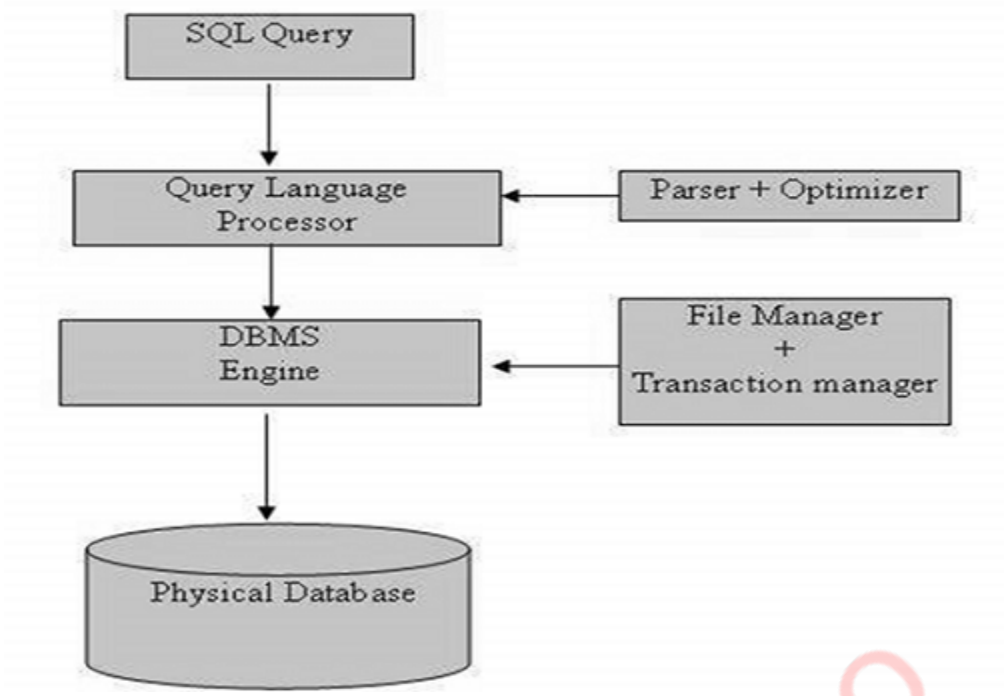
There are various components included in this process.

These components are –

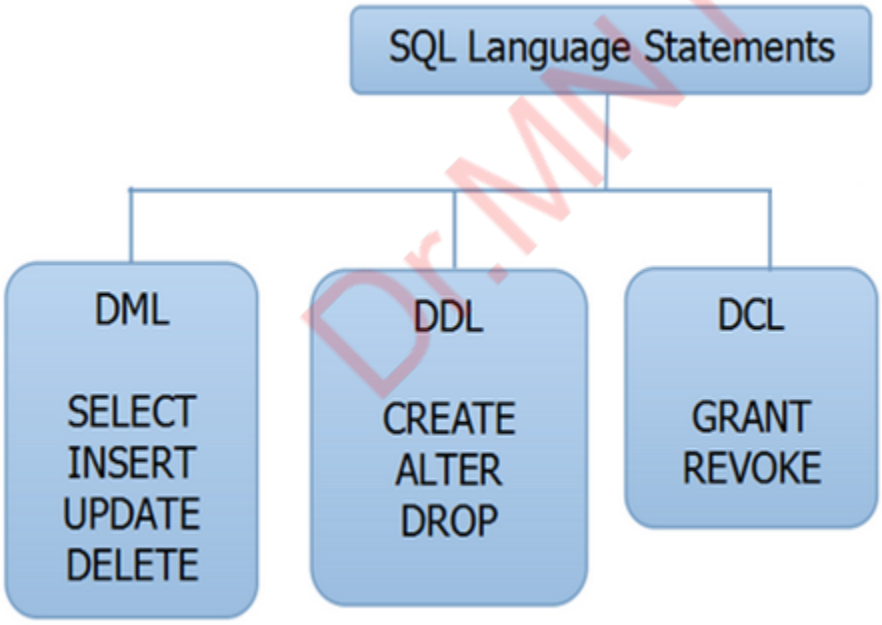
- Query Dispatcher
- Optimization Engines
- Classic Query Engine
- SQL Query Engine, etc.

A classic query engine handles all the non-SQL queries, but a SQL query engine won't handle logical files.

Following is a simple diagram showing the SQL Architecture –



SQL Commands



DDL

In Data Definition Language (DDL), we have three different SQL statements.

CREATE

CREATE statement is used to create a new table in an existing database. CREATE statement is also used to create other database object such as a stored procedure, function, etc.

Example

- Create Table Student (Rank Int,StudentName varchar(50),Mark Float)

ALTER

Alter statements can add a column, modify a column, drop a column, rename a column or rename a table.

Example

- Alter Table Student Add (StudentAddress varchar (100))

DROP

SQL DROP TABLE statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for the table.

Example

- Drop Student

DML(Data Manipulation Language):

we have four different SQL statements, Select, Insert, Update, and Delete.

SELECT Statement

The SELECT statement is used to select records from the table, with or without a condition.

Example

- select * from student- Gets all records of the student table.
- Select * from student where rank>5- Gets records with the condition where students' rank is greater than 5.

INSERT

INSERT statement is used to insert a set of values into a database table. Insert statement it used with Values.

Example- Insert Into Student (Rank, StudentName, Mark) Values(1,'Kumar',450)

UPDATE

The UPDATE statement is used to update existing values in a table, which is based on some condition.

Example

- update student set StudentName='Manoj' where StudentName='Kumar'

The query given above will update the studentName from Manoj to Kumar where student Name Kumar.

DELETE

Delete statement is used to delete the existing record in the table, which is based on some condition.

Example

- Delete from Student where StudentName='Manoj'

The query given above will delete records which has StudentName as Manoj.

DCL

In Data Control Language(DCL), it defines the control over the data in the database. We have two different commands, which are

GRANT

Grant is allowed to do the specified user to the specified tasks.

Syntax:

GRANT privilege_name
ON object_name
TO {user_name |PUBLIC |role_name}
[WITH GRANT OPTION];

REVOKE

It is used to cancel previously granted or denied permissions.

Syntax:

REVOKE privilege_name
ON object_name
FROM {user_name |PUBLIC |role_name}

Database transactions:

A transaction can be defined as a group of tasks. A single task is the minimum processing unit which cannot be divided further.

To explain the concept of a database transaction, let us use a typical example of transferring money between Account A and Account B. Let's say you want to move 5 dollars from Account A to Account B. This action can be broken down into the following simple operations:

1. Create a record to transfer 5 dollars from Account A to Account B. This is typically called the *begin* of a database transaction.
2. Read the balance from Account A.
3. Subtract 5 dollars from the balance of Account A.

4. Read the balance from Account B.
5. Add 5 dollars credit to Account B.

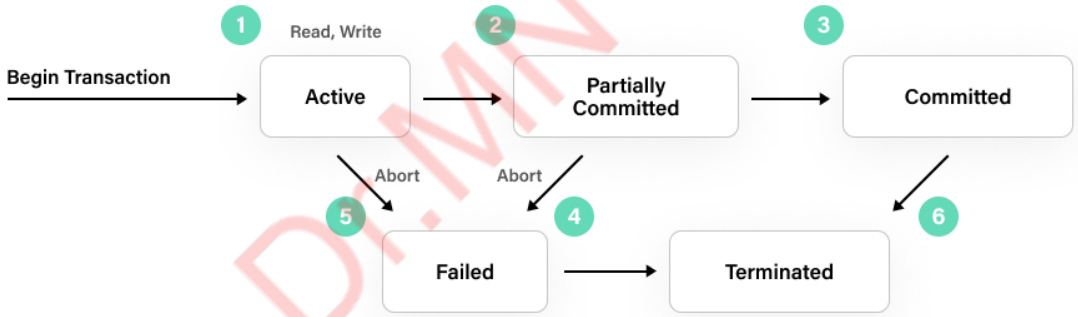
Need for database transactions:

Before we learn about how database transactions work, let's explore why they are needed in the first place.

1. System failures are inevitable, and in these cases, a transaction provides a way to ensure that the outcome is reliable and consistent. This means that the state of the database reflects all transactional changes committed before the point of failure and that transactions that were in-flight at the failure point are cleanly rolled back.
2. When multiple concurrent requests are hitting the database server, changing the same underlying data simultaneously, the transaction must isolate requests from each other to avoid conflicts.

Transaction life cycle:

A transaction in a database can be in one of the following states -



During its lifecycle, a database transaction goes through multiple states. These states are called transaction states and are typically one of the following:

1. **Active states:** It is the first state during the execution of a transaction. A transaction is *active* as long as its instructions (read or write operations) are performed.
2. **Partially committed:** A change has been executed in this state, but the database has not yet committed the change on disk. In this state, data is stored in the memory buffer, and the buffer is not yet written to disk.
3. **Committed:** In this state, all the transaction updates are permanently stored in the database. Therefore, it is not possible to rollback the transaction after this point.

4. **Failed:** If a transaction fails or has been aborted in the active state or partially committed state, it enters into a *failed* state.
5. **Terminated state:** This is the last and final transaction state after a committed or aborted state. This marks the end of the database transaction life cycle.

ACID properties

In relational databases, transactions must be atomic, consistent, isolated and durable. These properties are commonly abbreviated as ACID. ACID properties ensure that a database transaction is processed reliably.

Atomicity

Atomicity in terms of a transaction means *all or nothing*. When a transaction is committed, the database either completes the transaction successfully or rolls it back so that the database returns to its original state.

Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**.

Before: X : 500	Y: 200
Transaction T	
T1	**T2**
Read (X) X: = X − 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

Consistency

One of the key advantages of using a transaction is maintaining data integrity, regardless of whether it succeeds or fails. Transactions can only alter affected data in a way that is authorized by the database engine, ensuring that a consistent view of the data is maintained at all times. For example, when users deposit money in an online banking app, they want to see the result of this deposit reflected immediately when they view their balance. To ensure their money has not been lost. With strong transactional consistency, there should never appear to be more or less money in aggregate in the bank than there is.

It refers to the correctness of a database. Referring to the example above,

The total amount before and after the transaction must be maintained.

Total **before T** occurs = **500 + 200 = 700**.

Total **after T occurs** = **400 + 300 = 700**.

Therefore, the database is **consistent**. Inconsistency occurs in case **T1** completes but **T2** fails. As a result, T is incomplete.

Isolation

With multiple concurrent transactions running at the same time, each transaction should be kept independent without affecting other transactions executing simultaneously. For most database systems, the order of the transactions is not known in advance. Transactions are instead run in parallel, and some form of database locking is utilized to ensure that the result of one transaction does not impact that of another. Typically, databases offer several isolation levels to control the degree of transactional integrity.

Let **X**= 500, **Y** = 500.

Consider two transactions **T** and **T".**

T	T"
Read (X) X: = X*100 Write (X) Read (Y) Y: = Y – 50 Write (Y)	Read (X) Read (Y) Z: = X + Y Write (Z)

Suppose **T** has been executed till **Read (Y)** and then **T"** starts. As a result, interleaving of operations takes place due to which **T"** reads the correct value of **X** but the incorrect value of **Y** and sum computed by

T": (X+Y = 50, 000+500=50, 500)

is thus not consistent with the sum at end of the transaction:

T: (X+Y = 50, 000 + 450 = 50, 450).

This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

Durability

Durability means that a successful transaction commit will survive permanently. To accomplish this, an entry is added to the database transaction log for each successful transaction.

Data Centers

Data centers host large quantities of sensitive information, both for their own purposes and the needs of their customers.Data centers are made up of three primary types of components: compute, storage, and network. However, these components are only the top of the iceberg in a modern DC.

Data centers vary in size, from a small server room all the way up to groups of geographically distributed buildings, but they all share one thing in common: they are a critical business asset where companies often invest in and deploy the latest advancements in data center networking, compute and storage technologies.

Types of data centers:

The modern data center has evolved from a facility containing an on-premises infrastructure to one that connects on-premises systems with cloud infrastructures where networks, applications and workloads are virtualized in multiple private and public clouds.

- Enterprise data centers are typically constructed and used by a single organization for their own internal purposes. These are common among tech giants.
- Colocation data centers function as a kind of rental property where the space and resources of a data center are made available to the people willing to rent it.
- Managed service data centers offer aspects such as data storage, computing, and other services as a third party, serving customers directly.
- Cloud data centers are distributed and are sometimes offered to customers with the help of a third-party managed service provider.

Applications of data centers:

- Storage , backup, recovery and maintenance of databases.
- Integrate IoT, big data, AI and machine learning applications.
- Expand the use of utilities such as emails.
- Provide a platform for e-commerce applications.
- Ensure scalability.
- Provide flexibility of data access and ensure security of data.
- Reduce the operational overhead and maintenance of data.

Architecture of a data center:

Data center architectures and requirements can differ significantly. For example, a data center built for a cloud service provider like Amazon satisfies facility, infrastructure and security requirements that significantly differ from a completely private data center, such as one built for a government facility that is dedicated to securing classified data.

The primary elements of a data center break down as follows:

- **Facility** – the usable space available for IT equipment. Providing round-the-clock access to information makes data centers some of the world's most energy-consuming facilities. Design to optimize space and environmental

control to keep equipment within specific temperature/humidity ranges are both emphasized.

- **Core components** – equipment and software for IT operations and storage of data and applications. These may include storage systems; servers; network infrastructure, such as switches and routers; and various information security elements, such as firewalls.
- **Support infrastructure** – equipment contributing to securely sustaining the highest availability possible. The Uptime Institute has defined four tiers of data centers, with availability ranging from 99.671% to 99.995%. Some components for supporting infrastructure include:
    - **Uninterruptible Power Sources (UPS)** – battery banks, generators and redundant power sources.
    - **Environmental control** – computer room air conditioners (CRAC); heating, ventilation and air conditioning (HVAC) systems; and exhaust systems.
    - **Physical security systems** – biometrics and video surveillance systems.
- **Operations staff** – personnel available to monitor operations and maintain IT and infrastructure equipment around the clock.

Cloud services:

The growth of data has stimulated the need for huge storage mediums. Configuring such huge storage would mean increased operational costs and maintenance overheads. The invention of cloud services has provided a solution through unlimited data storage that is free or has minimal operational costs.

A cloud database is a database service built and accessed through a cloud platform. It serves many of the same functions as a traditional database with the added flexibility of cloud computing. Users install software on a Cloud infrastructure to implement the database.

Advantages

- This is a service provided through a cloud platform which can be accessed anywhere.
- Since they are offered as a service, they can be configured and maintained by the user without any providers
- They offer support to traditional relational databases such as MySQL, PostgreSQL and NoSQL.
- They can be accessed through web interfaces of APIS provided by various vendors like AWS, Azure and Google Cloud.

- Users can access cloud databases virtually, from anywhere through web interfaces.
- Since they are provided through cloud infrastructure, these databases can expand their storage capacity exponentially.
- When the databases crash due to hardware or software malfunction, they can be quickly re-configured.
- The databases are highly secured through cloud security mechanisms.