INTRODUCTION

File System

A file system is a process that manages how and where data on a storage disk, typically a hard disk drive (HDD), is stored, accessed, and managed. It is a logical disk component that manages a disk's internal operations as it relates to a computer and is abstract to a human user. The file system enables you to view a file in the current directory as files are often managed in a hierarchy.

Example: File Allocation Table (FAT), Global File System (GFS), Hierarchical File System (HFS), New Technology File System (NTFS), Universal Disk Format (UDF), etc.

Drawbacks of File System

1. Storage
2. Attributes
3. Concurrency
4. Security (role based access)
5. Duplication of data
6. Dependency on application programs

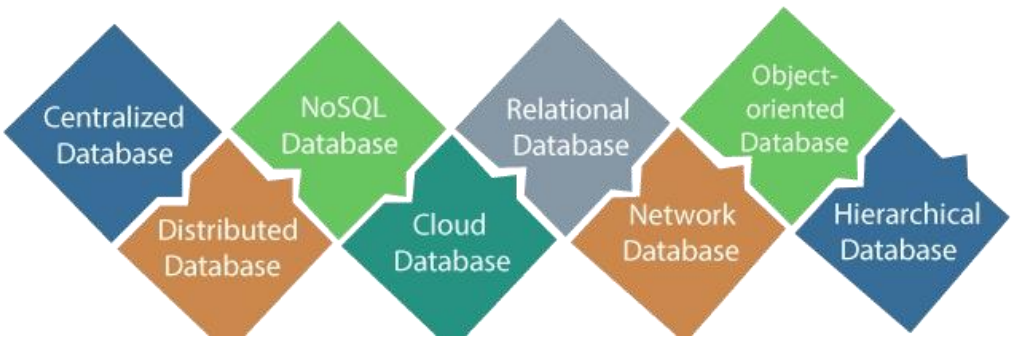
Database Management System (DBMS)

DBMS stands for database management system. We can break it like this DBMS = database + management system. Database is a collection of related data and management system is a set of programs (collection of operation) to store and retrieve those data. Based on this we can define DBMS like this:

DBMS is a collection of inter-related data and set of programs to store and access those data in an easy and effective manner.

For example: the college database organizes the data about the admin, staff, students and faculty etc.

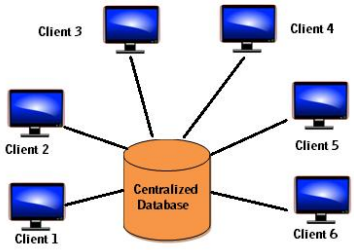
Database Types



1. Centralized database

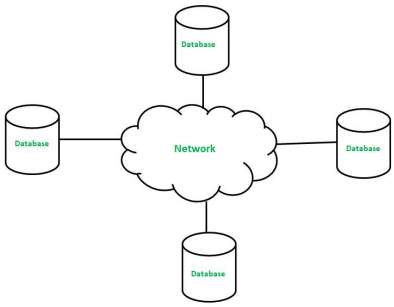
A centralized database stores all data in a single central location, such as a main computer or server. Users from different departments or locations access the same database through a network. This setup ensures data consistency, easier management, and simpler backup processes, as all information is maintained in one place.

However, it has a major drawback—if the central system fails, the entire database becomes inaccessible. Centralized databases are commonly used in banks, schools, and government organizations.



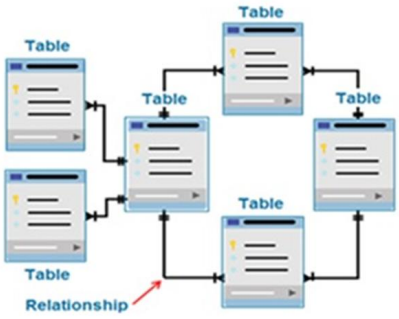
2. Distributed database

A distributed database stores data across multiple interconnected computers or servers located in different places. Even though data is physically distributed, it behaves as a single unified database to users. Each site can process queries locally and share data when needed. This improves performance, reliability, and availability, but managing synchronization and consistency can be complex. Distributed databases are ideal for large organizations and online systems like e-commerce platforms.



3. Relational Database (RDBMS)

A relational database organizes data into tables (rows and columns) where relationships are established through common fields or keys. It uses Structured Query Language (SQL) for data manipulation and ensures data integrity, accuracy, and reduced redundancy. This model is easy to understand and widely used for structured data. Examples include MySQL, Oracle, PostgreSQL, and SQL Server, which are used in banking, HR, and business applications.



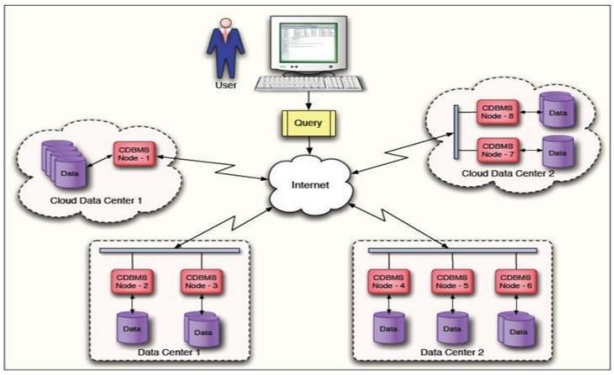
4. NoSQL Database (Non-Relational)

NoSQL databases are designed to handle unstructured and semi-structured data that doesn't fit well into traditional tables. They store data as documents, key-value pairs, graphs, or wide columns, making them

suitable for Big Data and real-time applications. NoSQL systems provide high scalability, flexibility, and fast data processing. Popular examples include MongoDB, Cassandra, Redis, and Neo4j, used in social media, IoT, and recommendation systems.

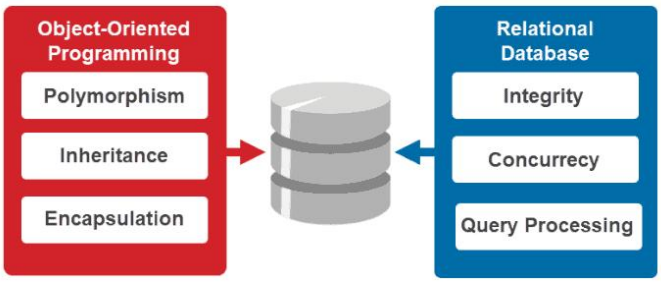
5. Cloud Database

A cloud database is hosted on cloud computing platforms instead of local servers. It provides remote accessibility, automatic backups, and high scalability, allowing users to access data anytime from anywhere via the internet. Cloud service providers handle maintenance, updates, and security. This makes cloud databases cost-effective and convenient for organizations of all sizes. Examples include Amazon RDS, Google Cloud SQL, and Microsoft Azure SQL Database.



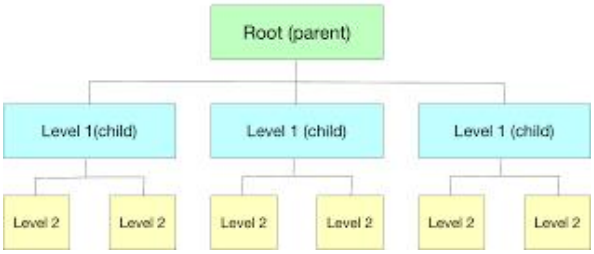
6. Object-Oriented Database

An object-oriented database stores data in the form of objects, similar to how it is represented in object-oriented programming languages like Java or C++. Each object contains both data (attributes) and functions (methods), allowing more natural data modeling. This type of database supports inheritance, encapsulation, and polymorphism, making it suitable for complex applications like CAD, multimedia, and scientific research. Examples include ObjectDB and db4o.



7. Hierarchical Database

A hierarchical database organizes data in a tree-like structure where each child record has only one parent. Data is represented as a hierarchy of records connected through links, supporting one-to-many relationships. This model provides fast data access for structured and predictable relationships but is difficult to modify if the structure changes. It is mainly used in banking, telecom, and legacy systems. An example is IBM's Information Management System (IMS).



8. Network Database

A network database represents data using a graph structure, where records are connected by links or pointers. Unlike hierarchical databases, it allows many-to-many relationships, offering more flexibility in data access. This model improves dat a efficiency and relationship representation, though it is more complex to design and maintain. Network databases were widely used in early business and engineering applications, with examples such as IDS and TurboIMAGE.

Application of DBMS

A Database Management System (DBMS) is used in almost every sector to efficiently store, manage, and retrieve large volumes of data. Some common applications of DBMS are as follows:

1. Banking: DBMS is used to manage customer information, accounts, loans, and daily banking transactions. It helps maintain transaction records, ATM operations, fund transfers, and account balances accurately.

2. Universities / Educational Institutions: Educational institutions use DBMS to handle student registration, course enrollment, grades, attendance, and examination results, ensuring efficient academic and administrative management.

***3. Airlines*:** DBMS is used for managing flight reservations, ticket bookings, passenger details, and schedule information. It ensures real-time data availability for both passengers and airline staff.

4. Library Management System: Libraries use DBMS to maintain records of book issues, return dates, book titles, authors, and availability. It simplifies tracking of borrowed books and managing library inventories.

5. Telecommunications: DBMS manages call records, billing information, and prepaid account balances. It helps generate monthly bills and monitor customer usage efficiently.

6. Sales and Marketing: DBMS is used to store and manage customer data, product details, purchase history, and sales transactions. It also supports inventory tracking and customer relationship management.

7. Finance: Financial institutions use DBMS to maintain data related to holdings, sales, purchases, and market transactions of financial instruments like stocks, bonds, and mutual funds, ensuring accuracy and security.

8. Human Resource Management (HRM): DBMS stores detailed information about employees, salaries, payroll, taxes, benefits, and attendance, helping in payroll processing and workforce analysis.

9. Manufacturing: DBMS assists in managing supply chains, production tracking, inventory management, and order processing, ensuring smooth factory operations and resource planning.

10. Medical Sector: Hospitals and clinics use DBMS to maintain patient records, appointment schedules, treatment history, prescriptions, and billing information, improving healthcare efficiency and patient care.

11. Military and Defense: DBMS is used to manage personnel records, equipment inventories, mission data, and logistics, ensuring the security, accuracy, and confidentiality of sensitive military information.

Advantage of DBMS over File System

1. Reduced Data Redundancy and Improved Consistency: In file systems, the same data is often stored in multiple places, causing duplication and inconsistency. A DBMS maintains a centralized database, avoiding redundancy and ensuring that data remains consistent and up to date throughout the system.

2. Better Data Sharing and Security: DBMS allows multiple users and applications to access data simultaneously in a controlled manner. Through authorization and access controls, the Database Administrator (DBA) ensures that only authorized users can view or modify sensitive information, enhancing overall data security.

3. Data Integrity and Independence: A DBMS enforces integrity constraints to maintain the accuracy and validity of stored data. It also provides data independence, meaning changes in the data structure do not affect application programs. This separation of data and applications increases system flexibility and maintainability.

4. Efficient Data Access and Management: DBMS uses optimized algorithms and query languages (like SQL) to store, retrieve, and process data efficiently. This results in faster performance and easier data management compared to file systems.

5. Backup, Recovery, and Reliability: DBMS includes built-in backup and recovery mechanisms to protect data from loss due to system failures. It ensures data reliability and quick restoration, which is difficult to achieve in file-based systems.

Disadvantages of DBMS

1. *Increased Complexity:* A DBMS is more complex to design, implement, and manage than a simple file-based system. It requires proper planning, configuration, and understanding of database concepts, which makes it more challenging to maintain.

2. *Need for Specialized Manpower:* Operating and managing a DBMS requires trained database administrators (DBAs), designers, and developers with specialized skills. Recruiting and training such personnel increase operational costs.

*3. Large System Size***:** DBMS software occupies a significant amount of memory and storage space. Along with databases, supporting tools and utilities can make the system heavy and resource-demanding.

4. High Implementation and Maintenance Cost: The initial setup cost of a DBMS, including hardware, software licenses, and skilled staff, is much higher than that of a traditional file system. Continuous updates, backups, and maintenance also add to the expense.

5. Performance Issues: Because DBMSs are designed to handle a wide range of applications, they may not be optimized for specific tasks. This generality can reduce performance for smaller or specialized applications compared to a tailor-made file system.

DBMS vs File System

Aspect	File System	Database Management System (DBMS)
Definition	Used to manage and organize files stored on a computer's hard disk.	A software system used to store, manage, and retrieve user data efficiently.
Data Redundancy	Redundant (duplicate) data is often present.	Redundancy is minimized through centralized control.
Query Processing	Query processing is not efficient and requires manual coding.	Query processing is fast and efficient using SQL.
Data Consistency	Data consistency is low due to duplication and scattered storage.	High data consistency is achieved through normalization and integrity constraints.
Complexity and Transactions	Simple structure; does not support complex transactions.	More complex but supports complicated transactions easily.
Security	Provides limited data security.	Offers strong security mechanisms and access controls.
Cost	Less expensive to implement and maintain.	More expensive due to software, hardware, and skilled personnel costs.
Crash Recovery	Does not support recovery after system failure.	Provides robust backup and crash recovery mechanisms.

Database Architecture

Database architecture represents the logical framework that defines the design, structure, and organization of a database system.It specifies how data is stored, managed, retrieved, and utilized by users and applications, serving as a blueprint for the functioning of a database within a computer system.

The architecture of a DBMS breaks down the database system into separate components, allowing them to be modified, updated, or replaced independently without affecting the entire system.This layered approach also provides a clear understanding of the individual components and their roles within the database.

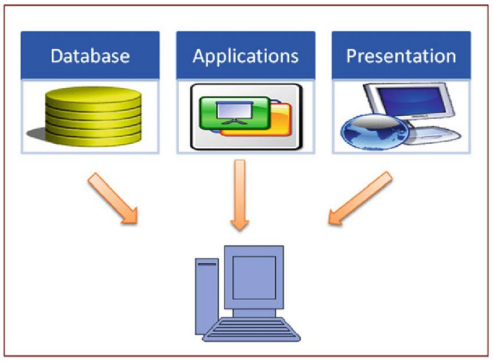
Types of DBMS Architecture

There are mainly three types of DBMS architecture:

1. One tier architecture or single tier architecture
2. Two-tier architecture
3. Three tier architecture

One Tier Architecture or Single Tier Architecture

1 tier architecture in DBMS is the simplest architecture of database in which the client, server, and database all reside on the same machine. In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.The 1-tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

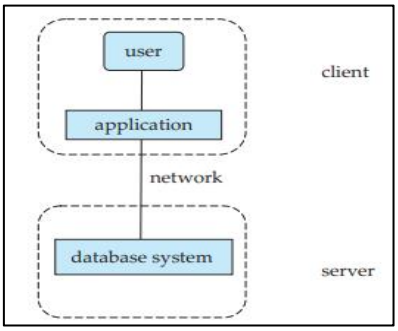


Two Tier Architecture

A Two-Tier Architecture in DBMS is a database architecture in which the presentation layer (user interface and application logic) runs on the client (such as a PC, mobile device, or tablet), while the data is stored and managed on a server, known as the second tier.

This model follows the basic client-server architecture, where the client application directly communicates with the database system on the server using APIs such as ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity). In this architecture:

- The client side is responsible for the user interface and application logic.
- The server side handles core database functions, including query processing, transaction management, and data storage.
- Communication between the client and the server occurs through a network connection, enabling the client application to send queries and receive results directly from the database system.



Advantages: Simple and easy to maintain,Fast communication,High performance for small systems,Cost-effective.

Disadvantages: Limited scalability,Security issues,Poor flexibility,Performance bottlenecks

Three Tier Architecture

A Three-Tier Architecture in DBMS is a database design model that introduces an additional application layer between the client and the server. Unlike the two-tier model, in this architecture, the client cannot directly communicate with the database server. Instead, the client interacts with an application server, which processes the request and then communicates with the database system.

The end user has no direct knowledge of the database beyond the application server, and similarly, the database has no awareness of the individual users—it only communicates with the application server. This architecture is commonly used in large-scale web applications where high performance, security, and scalability are essential. Layers of Three-Tier Architecture:

1. *Presentation Layer (Client Tier)*

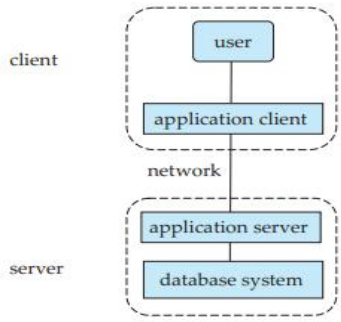
This is the topmost layer that interacts directly with the user through interfaces like web browsers or mobile apps. It handles user input and output but does not process or store data.

2. *Application Layer (Middle Tier)*

Also called the business logic layer, it serves as a bridge between the user interface and the database. It processes client requests, applies business rules, performs computations, and sends the required queries to the database server. Examples include web servers and API frameworks such as Java EE, .NET, or Python Flask.

3. *Database Layer (Data Tier)*

This is the bottom layer responsible for data storage, retrieval, and management. It consists of the DBMS and the database server, handling query processing, transaction management, and data integrity.



Advantages: High scalability and flexibility,Enhanced security.

Disadvantages: Complex implementation,Higher cost

Database Schema

A Database Schema is the logical structure or blueprint of a database that defines how data is logically organized, stored, and related within a Database Management System (DBMS). It represents the overall design of the database, showing how data is arranged and how different entities are connected. The schema acts as a framework for data storage but does not contain the actual data itself. Instead, it defines the structure and rules that guide how data is stored and accessed.It includes schema objects such as tables,

fields, views, indexes, relationships, primary keys, and foreign keys, along with definitions of attributes, data types, and constraints. Acting as the blueprint of the database, it ensures data consistency, integrity, and efficient management by defining how information is logically structured and interrelated throughout the system.

Types of Database Schema

1. Physical Schema

Describes how data is physically stored on storage devices (e.g., files, blocks, indexes). It includes information on data storage paths, file organization methods, access mechanisms, and indexing techniques. This schema focuses on performance optimization and storage efficiency.
Example: How the student table is stored on disk and how indexes are created for faster searching.

2. Logical Schema

Defines the logical design of the database structure, independent of physical storage. It specifies tables, columns, relationships, constraints, and keys. This schema is created by database designers and is the most important part for users and developers.
Example: A student database with tables for Student, Course, and Enrollment along with their relationships.

3. View Schema (External Schema)

Describes how individual users or applications view data. It defines subsets of the database relevant to specific users, providing customized access and security. Multiple view schemas can exist for a single logical schema.
Example: A teacher may only view student names and grades, while an admin can access all student details.