

# DBMS

## Introduction

A **Database Management System (DBMS)** is a software system that is designed to manage and organize **data** in a structured manner within a **database**. It allows users to **create, modify, and query a database**, as well as manage the security and access controls for that database.

## Key Features of DBMS

1. **Data modeling:** A DBMS provides tools for creating and modifying data models, which define the structure and relationships of the data.
2. **Data storage and retrieval:** A DBMS is responsible for storing and retrieving data from the database and can provide various methods for searching and querying the data.
3. **Concurrency control:** A DBMS provides mechanisms for controlling concurrent access to the database, to ensure that multiple users can access the data without conflicting with each other.
4. **Data integrity and security:** A DBMS provides tools for enforcing data integrity and security constraints, such as constraints on the values of data and access controls that restrict who can access the data.
5. **Backup and recovery:** A DBMS provides mechanisms for backing up and recovering the data in the event of a system failure.

## Classification of DBMS

### *Relational Database Management System (RDBMS)*

- Organizes data in tables with rows and columns.
- Establishes relationships between tables using primary and foreign keys.

### **Non-Relational Database Management System (NoSQL)**

- Organizes data in diverse structures like key-value pairs, documents, graphs, or column-based formats.
- Tailored to handle large-scale, high-performance scenarios efficiently.

## History of DBMS

- In the **1950s and early 1960s**, data storage and processing were primarily handled using **magnetic tapes and punched card decks**. Data processing tasks, such as payroll, involved sequentially reading data from tapes or card decks, making the process time-consuming and rigid. Accessing data directly was not possible due to the sequential nature of tapes.
- During the early 1960s, **Charles Bachman** developed the **Integrated Data Store (IDS)**, one of the first database management systems based on the network data model. This significant advancement earned him the **Turing Award**. IDS allowed more complex data relationships and laid the groundwork for future DBMS developments.
- In the **late 1960s**, **IBM** introduced the **Integrated Management System**, based on the **hierarchical data model**. This period also saw the introduction of hard disks, allowing direct data access and enabling more flexible and efficient data management, marking a technological shift.
- **The 1970s** marked a revolutionary change with **Edgar Codd's** introduction of the **relational database model**. This model simplified data management and querying by organizing data into tables with rows and columns. Its simplicity and efficiency quickly made it the standard for database systems.
- During the **1980s**, **IBM** developed the **Structured Query Language (SQL)** as part of the System R project, which became the standard language for managing and querying relational databases. This decade saw the emergence of commercial relational databases like IBM DB2, Oracle, and Ingres, which eventually replaced older network and hierarchical models.
- In the early **1990s**, the focus shifted to **decision support and querying applications**. Tools for analyzing large datasets became increasingly important, and parallel database products were introduced to handle the growing data volumes. Databases also began incorporating object-relational support to enhance functionality.

- The **1990s** witnessed the explosive growth of the World Wide Web, significantly **impacting database deployment**. Databases needed to support high transaction-processing rates, reliability, and 24/7 availability. Web interfaces to databases became essential to meet the demands of web applications.
- In the **2000s**, **XML** and **XQuery** **emerged** for complex data types and data exchange, while relational databases remained core for large-scale applications. Open-source databases like **PostgreSQL** and **MySQL** saw significant growth. Specialized databases such as column-stores for data analysis and highly parallel systems were developed. Distributed data-storage systems for large web platforms like Amazon and Google also emerged, along with advancements in streaming data management and data-mining techniques.

## **Different Types of DBMS**

There are **various types of database management systems** based on database structures. We can arrange data in various formats for a variety of use cases. Let's see these types of DBMS one by one:

### **Centralized DBMS**

In a centralized database, a single central database is used to serve data to multiple devices. Each user can access the database after authentication and be able to work with it.

### **Decentralized DBMS**

In the decentralized database, all the data is collectively stored in multiple databases. All these databases are connected with the help of networking. To the end-user, this entire system appears like a single coherent system.

### **Relational DBMS**

The relational database management system is also known as RDBMS. It is one of the types of DBMS which is widely used for commercial applications. It contains tables in which data is stored in the form of rows and columns like

an Excel sheet. Some of the tables possess the relationship among them and the data is retrieved with the help of join operation. This join operation helps us to get data from 2 or more tables with the help of logical queries.

## **NoSQL DBMS**

NoSQL or non-relational databases are the most popular databases due to their high scalability and availability. In this type of database, the data is stored in collections, and it doesn't contain tables like relational databases.

Collection is simply the group of documents in which we have data with similar meanings and similar purposes.

In NoSQL, we can store data in key-value pairs as well as with the help of graphs. It increases productivity by a significant amount and comparatively, it is easy to work with them.

## **Hierarchical DBMS**

In hierarchical databases, data is arranged in a tree-like format where we have a parent-child relationship between nodes. The parent can have many children, but children contain only one parent.

## **Network DBMS**

The network database model has various nodes, and these nodes are connected with each other. These models are complex in nature. This model allows multiple parents for a single child node so we can create more complicated structures with it.

## **Object-Oriented DBMS**

The Object-oriented database management system is one of the types of DBMS, in which data is stored in the objects. These objects are created from the classes. Classes are nothing but the description of an object. It is like object-oriented programming languages.

## **Some Popular Database Management Systems**

There are several popular **Database Management Systems (DBMS)** that cater to different needs and preferences. Some widely used **DBMS** are:

### **1. MySQL:**

- An **open-source relational database management system (RDBMS)**.
- Known for its reliability, ease of use, and strong community support.
- Frequently used for web applications and small to medium-sized databases.

### **2. PostgreSQL:**

- An **open-source object-relational database system**.
- Emphasizes extensibility and standards compliance.
- Suitable for complex applications and large-scale databases.

### **3. Microsoft SQL Server:**

- A **relational database management system** developed by Microsoft.
- Offers a comprehensive suite of features and tools for enterprise-level applications.
- Commonly used in conjunction with Microsoft's .NET framework.

### **4. Oracle Database:**

- A powerful and widely used **relational database management system**.
- Known for its scalability, security features, and support for complex transactions.
- Popular in large enterprises and critical business applications.

### **5. MongoDB:**

- A leading **NoSQL database management system**.
- Stores data in flexible, JSON-like documents in a schema-less fashion.
- Ideal for handling large amounts of unstructured or semi-structured data.

## 6. SQLite:

- A self-contained, serverless, and zero-configuration **relational database engine**.
- Lightweight and suitable for embedded systems, mobile applications, and small-scale deployments.

## 7. Redis:

- An **in-memory data structure store** that is often used as a cache or message broker.
- Provides high-performance data storage and retrieval for key-value pairs. Commonly used in real-time applications as a caching mechanism.

## Advantages of DBMS

1. **Security and Reliability:** DBMS ensures secure data storage through authentication and user authorization.
2. **Data Redundancy Reduction:** Normalization techniques help minimize and remove data redundancy.
3. **Multiple Data Views:** Provides different data views tailored for different users' needs.
4. **Backup and Recovery:** Facilitates data backup and recovery to prevent data loss.
5. **Integration with Programming Languages:** Can be integrated with languages like Python and Java to enhance database functionalities.

## Disadvantages of DBMS

1. **Complexity:** DBMS systems can be complex to work with and manage.
2. **Cost of Hardware:** Involves significant cost for purchasing necessary hardware for data storage.
3. **Setup Time:** Setting up a DBMS can be time-consuming.
4. **Licensing Costs:** Many commercial DBMS products require paid licenses.
5. **Need for Skilled Staff:** Requires skilled technical staff, adding to the operational costs.

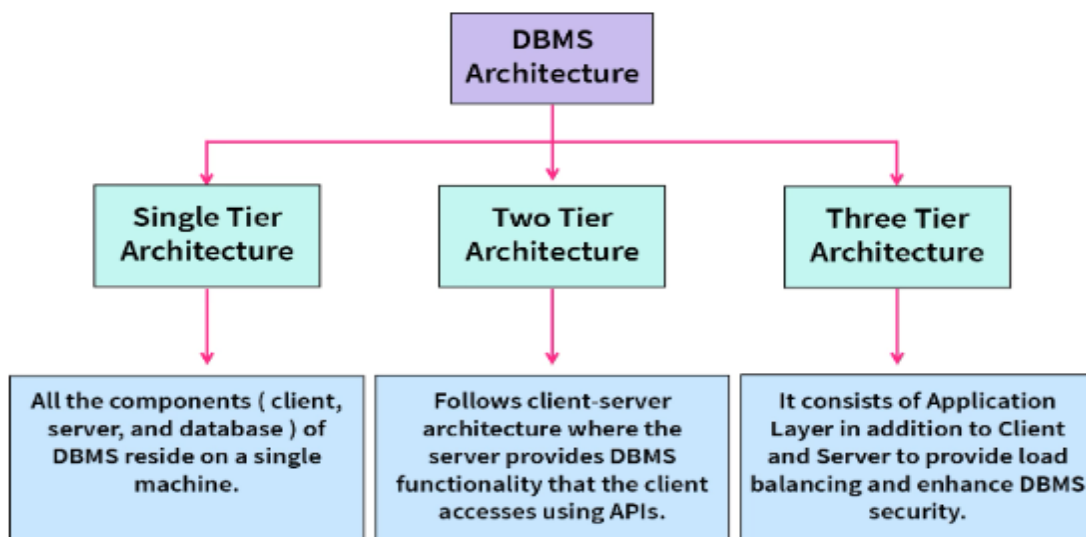
## DBMS Architecture Types

Database management systems (DBMS) are organized into **multiple levels of abstraction** to ensure proper functioning. These layers describe both the design and the operations of the DBMS, facilitating a structured approach to database management. A DBMS is **not always directly accessible** by users or applications; instead, **various architectures** are employed based on how users connect to the database. These architectures are classified into **tiers**, defining the number of layers in the DBMS structure.

An **n-tier DBMS architecture** divides the entire DBMS into related but independent **layers**. For example, a **one-tier architecture** has a **single layer**, a **two-tier architecture** has **two layers**, and a **three-tier architecture** has **three layers**. As the number of layers increases, so does the level of abstraction, enhancing both the security and complexity of the DBMS. Importantly, these layers are independent of each other, meaning changes in one layer do not impact others, allowing for **modular** and **flexible** system management.

Now, let's look at the most common DBMS architectures:

- *Single Tier Architecture (One-Tier Architecture)*
- *Two-Tier Architecture*
- *Three-Tier Architecture*



## **1-Tier Architecture:**

**Definition:** In a 1-tier architecture, the database is directly accessible by the user or application without any intermediary layers.

**Explanation:** The user interacts directly with the database. All the data, business logic, and presentation logic are handled within a single layer, often on a single machine.

**Example:** When learning SQL, you set up an SQL server and a database on your local system. This setup allows you to interact directly with the relational database and execute operations without a network connection. This direct interaction on a single machine is an example of 1-Tier DBMS architecture.

## **2-Tier Architecture:**

**Definition:** A 2-tier architecture consists of two layers: the client (user interface) and the server (database).

**Explanation:** In this architecture, the client communicates directly with the database server. The business logic and database management are handled on the server side, while the user interface runs on the client side.

**Example:** When you visit a bank to withdraw cash, the banker enters your withdrawal amount and account details into a system. The client-side application then communicates with the server-side database to check your account balance. This interaction between the client application and the server database is an example of 2-Tier DBMS architecture.

## **3-Tier Architecture**

**Definition:** A 3-tier architecture includes three layers: the presentation layer (user interface), the application layer (business logic), and the database layer.

**Explanation:** The user interface interacts with the application server, which in turn communicates with the database server. This separation allows for more



scalable and manageable systems, where the business logic is handled separately from data storage and user interaction.

**Example:** A web application, such as an online shopping site. The user's browser (presentation layer) interacts with a web server (application layer) that processes request and communicates with a database server (database layer) to fetch and store data.