**DBMS**

**What is DBMS?**

Database Management System is a software or technology used to manage data from a **#**database. Some popular databases are MySQL, Oracle, MongoDB, etc.

***# database is a collection of related data***

DBMS provides many operations e.g. creating a database, storing in the database, updating an existing database, delete from the database. DBMS is a system that enables you to store, modify and retrieve data in an organized way. It also provides security to the database.

**Types of Data Bases**

1. **Structured database**: can be stored in a particular structure – e.g. RDBMS (Relational DBMS) -> used in Indian Railways, School / College management Systems

Structured data is stored in the forms of relation. [relation -> technical term fort tables ]

1. **Unstructured database:** there is no predefined way for storing the data. Most of the data stored nowadays is unstructured data. **Used in Big Data.**

**Advantages of DBMS**

*#* ***Note:*** *File system stores the data in the hierarchical format, but we use client-server data at present, for which the file system is not efficient*

1. **Faster, efficient Searching:** The Database management system (DBMS) helps to produce quick answers to database queries thus making data access faster and more accurate.
2. **Better memory utilisation.**
3. **Ease of searching**: Users do not need to know the address of the stored data to access the data. No metadata is required for searching of data (which is mandatory in the case of a file sytem).
4. **Concurrency:** Multiple users search data at the same time (concurrently), this will be problematic in the case of file systems, but there are protocols present in DBMS which solve those problems.
5. **Security:** Supports ***role-based access to data***, that is specific data is visible to the viewer according to their authority.
6. **Prevents data redundancy:** When working with a structured database, DBMS provides the feature to prevent the input of duplicate items in the database. e.g. – If there are two same students in different rows, then one of the duplicate data will be deleted.
7. **Scalability and flexibility**: DBMS is highly scalable and can easily accommodate changes in data volumes and user requirements. It provides flexibility in data storage, retrieval, and manipulation, allowing users to easily modify the structure and content of the database as needed.

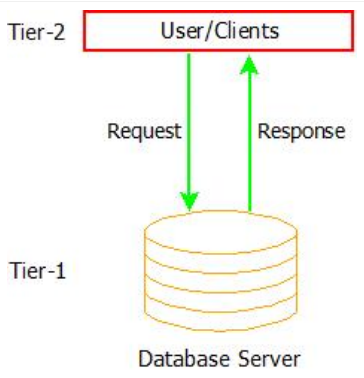
**Disadvantages of DBMS**

1. **Cost of hardware is software:** For DBMS, it is mandatory to have a high-speed processor and also a large memory size.
2. **Requires trained staff** to use DBMS, hence cost of staff training involved before implementation.
3. **Cost of data conversion:** We need to convert our data into a database management system, there is a requirement of a lot of money as it adds to the cost of the database management system.

**Types of architecture in DBMS**

*# note : tier -> layer*

1. **2-Tier Architecture:** A client-server architecture where the user interface and the application logic are separated into two separate components. The client component is typically the user interface and the server component is responsible for handling the data and business logic.



**2-Tier Architecture**

e.g: Bank management sytem, railway management system.

**Advantage(s):**

Maintainence is easy as user is limited

**Disadvantage(s):**

**Security:** Client directly interacts with the database, this can cause problems.

**Scalability:** When number of user and amount of data being accessed concurrently, this architecture fails(lead s to bottle neck problem)

1. **3-Tier Architecture:** The 3-tier architecture divides an application's components into three tiers or layers. Each layer has its own set of responsibilities. Here the client does not directly communicate with the server. Instead, it interacts with an application server which further communicates with the database system and then the query processing and transaction management takes place.

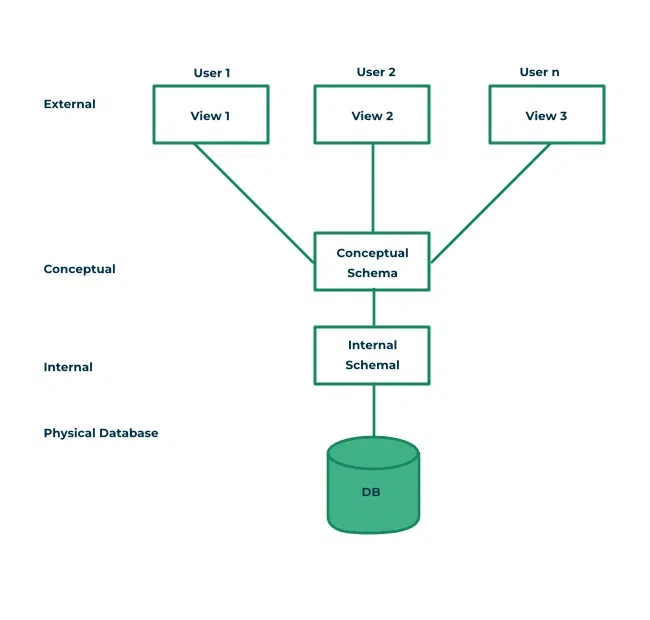
**Layers of 3-Tier Architecture:**

**View Level->Presentation Tier:** The presentation tier is the user interface or client layer of the application. It is responsible ***for presenting data to the user and receiving input from the user***. This tier can be a web browser, mobile app, or desktop application.

**Logical Level->Application Tier:** The application tier is the middle layer of the 3-tier architecture. It is responsible for ***processing and managing******the business logic of the application***. This tier communicates with the presentation tier to receive user input and communicates with the data management tier to retrieve or store data. This tier may include application servers, web servers, or APIs.

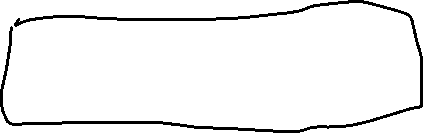
**Physical Level->Data Management Tier:** The data management tier is the bottom layer of the 3-tier architecture. It is responsible for ***managing and storing data*.** This tier can include databases, data warehouses, or data lakes. The data management tier communicates with the application tier to receive or store data.

**\*\* internal \*\*schema is also called physical \*\*schema \*\***

****

**Tier-1**

**View Level (Presentation Layer/ Presentation Tier)**



**Tier-2**

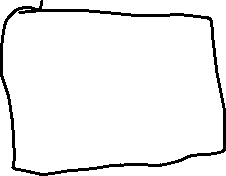
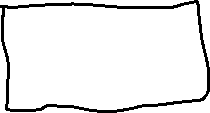
**Logical Level (Application Layer/ Tier)**

**Tier-3**

**Physical Level (Data Management Tier**

Logical Data independence

Physical Data independence

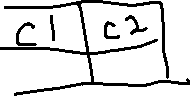


**3-Tier Architecture**

**#Logical Level independence: *The ability to modify the logical \*\*schema without causing application programs to be rewritten.*** Changes done at the conceptual level will not affect the application layer, it will not affect the actual logical structure. This is done using views [***view:*** *virtual table -> portion of a table selected, being shown to the user as a whole table].*

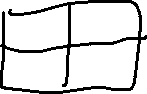
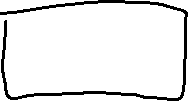
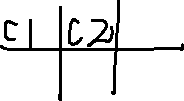
Let us consider the following table being shown to user1 and user2.

Now if user1 adds another column **C3** in the table. It will not affect what is visible to user2, since the logical structure remains the same. This way logical level independence is achieved.



**Initial View for**

**user1, user2**



**New View for user2 New view for user1**

**#Physical data independence:** Data changed through 1) Storage Structure 2) Data Structure change 3) Index value change, will not affect the conceptual \*\*schema. This is known as physical data independence.

Application of this can be seen in case of large companies, e.g. Google have changed many storages according to their needs, but that did not affect the overall data in the conceptual \*\*schema, i.e. applications produced will remain intact.

**Advantages:**

1. **Improved scalability:** The application servers can be deployed on many machines. Also, the database does not make longer connections with every client – it only requires connections from a smaller number of application servers.
2. **Improves data integrity:** All the updated information goes through the second tier. The second tier can ensure that only important information is allowed to be updated in the database and the risk of unreliable client applications corrupting information is removed.
3. **Security:** The client does not have direct access to the database; hence it is more difficult for a client to obtain unauthorized data.
4. High-performance, lightweight.
5. Better to re-use.
6. **Scalability:** Each item can scale horizontally.

**Disadvantages**

1. It is more complex than the 2-tier client-server computing model; because it is more difficult to build a 3-tier application compared to a 2-tier application. The points of communication are doubled.
2. The client does not maintain a persistent database connection.
3. A separate proxy server may be required.
4. Network traffic will be increased if a separate proxy server is used.
5. The physical separation of application servers containing business logic functions and database servers containing databases may be something that affects performance.

**\*\*Schema:**

It is the logical representation of a database.

In DBMS data is stored in the memory, but while presenting to the user, it is represented in the form of tables.

**Integrity Constraints:**

**Entity Relationship Diagram/Model (ER Diagram/ER Model):**

This is used to conceptually view the table structure. This helps to create a basic idea of the structure of the table. This is done in order to identify the requirements, before creation of the actual table.

Entity + Relation



i.e. ERD gives us the relationship

Object Relationship b/w objects in a schema

e.g.

Student Study Course

**Types of Attributes**

There are **six** such types of attributes: **Simple, Composite, Single-valued, Multi-valued, and Derived attribute**. One more attribute is their, i.e. **Complex Attribute, this is the rarely used attribute**.

**Simple attribute:** An attribute that cannot be further subdivided into components is a simple attribute.

**e.g.** The roll number of a student, the id number of an employee.

**Composite attribute**: An attribute that can be split into components is a composite attribute.

**e.g.** The address can be further split into house number, street number, city, state, country, and pin code, the name can also be split into first name middle name, and last name.

**Single-valued attribute**: The attribute which takes up only a single value for each entity instance is a single-valued attribute.

**e.g.** The age of a student.

**Multi-valued attribute:** The attribute which takes up more than a single value for each entity instance is a multi-valued attribute.

**e.g.** Phone number of a student: Landline and mobile.

**Derived attribute:** An attribute that can be derived from other attributes is derived attributes.

**e.g.** Total and average marks of a student.

**Complex attribute:** Those attributes, which can be formed by the nesting of composite and multi-valued attributes, are called “Complex Attributes“. These attributes are rarely used in DBMS (DataBase Management System). That’s why they are not so popular.

**Stored attribute:**  The stored attribute are those attribute which doesn’t require any type of further update since they are stored in the database.

**e.g.** DOB (Date of birth) is the stored attribute.

**Key attribute:** Key attributes are those attributes that can uniquely identify the entity in the entity set.

**e.g.** Roll-No is the key attribute because it can uniquely identify the student.

