Database Management System

A **database management system (DBMS)** is a collection of programs that enables users to create and maintain a database. The DBMS is hence a general-purpose software system that facilitates the processes of defining, constructing, and manipulating databases for various applications. Defining a database involves specifying the data types, structures, and constraints for the data to be stored in the database. Constructing the database is the process of storing the data itself on some storage medium that is controlled by the DBMS. Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data. **"Mini world" refers to the real-world system or domain that the database is designed to represent and manage.** It is not necessary to use general-purpose DBMS software to implement a computerized database. We could write our own set of programs to create and maintain the database, in effect creating our own special-purpose DBMS software. In either case—whether we use a general-purpose DBMS or not—we usually have to employ a considerable amount of software to manipulate the database. We will call the database and DBMS software together a database system.

An Example Let us consider an example that most readers may be familiar with: a UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment. The database is organized as five files, each of which stores data records of the same type. The STUDENT file stores data on each student; the COURSE file stores data on each course; the SECTION file stores data on each section of a course; the GRADE_REPORT file stores the grades that students receive in the various sections they have completed; and the PREREQUISITE file stores the prerequisites of each course.

DataBase:

A database is a collection of related data. By data, we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers, and addresses of the people you know.

Evolution of DB & DBMS:

Flat files (1960s – 1980s)

Flat file database is a database that stores information in a single file or table. In a text file, every line contains one record where fields either have fixed length or they are separated by commas, whitespaces, tabs or any other character. In a flat file database, there is no structural relationship among the records and they cannot contain multiple tables as well.

Advantages:

- Flat file database is best for small databases.
- It is easy to understand and implement. Fewer skills are required to handle a flat file database.
- Less hardware and software skills are required to maintain a flat file database.

Disadvantages:

- A flat file may contain fields which duplicate the data as there is no automation in flat files.
- If one record is to be deleted from the flat file database, then all the relevant information in different fields has to be deleted manually making the data manipulation inefficient.
- Flat file database waste the computer space by requiring it to keep the information on items that are logically cannot be available.

- Information retrieving is very time consuming in a large database.

Implementation of a flat file database

Flat file database is implemented in:

- Berkeley DB
- SQLite
- Mimesis
- TheIntegrationEngineer etc.

Hierarchical database (1970s – 1990s)

As the name indicates, hierarchical database contains data in a hierarchically-arranged data. More perceptively it can be visualized as a family tree where there is a parent and a child relationship. Each parent can have many children but one child can only have one parent i.e.; one-to-many relationship. Its hierarchical structure contains levels or segments which are equivalent to the file system's record type. All attributes of a specific record are listed under the entity type.

In hierarchical database, the entity type is the main table, rows of a table represent the records and columns represent the attributes.

Advantages:

In a hierarchical database pace of accessing the information is speedy due to the predefined paths. This increases the performance of a database.

The relationships among different entities are easy to understand.

Disadvantages:

Hierarchical database model lacks flexibility. If a new relationship is to be established between two entities, then a new and possibly a redundant database structure has to be built.

Maintenance and of data is inefficient in a hierarchical model. Any change in the relationships may require manual reorganization of the data.

This model is also inefficient for non-hierarchical accesses.

Network database (1970s – 1990s)

The inventor of network model is Charles Bachmann. Unlike hierarchical database model, network database allows multiple parent and child relationships i.e., it maintains many-to-many relationship. Network database is basically a graph structure. The network database model was created to achieve three main objectives:

- To represent complex data relationships more effectively.
- To improve the performance of the database.
- To implement a database standard.

In a network database a relationship is referred to as a set. Each set comprises of two types of records, an owner record which is same as parent type in hierarchical and a member record which is similar to the child type record in hierarchical database model.

Advantages:

- The network database model makes the data access quite easy and proficient as an application can access the owner record and all the member records within a set.
- This model is conceptually easy to design.
- This model ensures data integrity because no member can exist without an owner. So the user must make an owner entry and then the member records.
- The network model also ensures the data independence because the application works independently of the data.

Disadvantages:

- The model lacks structural independence which means that to bring any change in the database structure; the application program must also be modified before accessing the data.
- A user-friendly database management system cannot be established via network model.

Implementation of network database

Network database is implemented in:

- Digital Equipment Corporation DBMS-10
- Digital Equipment Corporation DBMS-20
- RDM Embedded
- Turbo IMAGE
- Univac DMS-1100 etc.

Relational database (1980s – present)

Relational database model was proposed by E.F. Codd. After the hierarchical and network model, the birth of this model was huge step ahead. It allows the entities to be related through a common attribute. So, in order to relate two tables (entities),

they simply need to have a common attribute. In the tables there are primary keys and alternative keys. Primary keys form a relation with the alternative keys. This property makes this model extremely flexible.

Thus, using relational database sample information can be stored using small tables. The accessing of data is also very efficient. The user only has to enter a query, and the application provides the user with the asked information.

Relational databases are established using a computer language, Structured Query Language (SQL). This language forms the basis of all the database applications available today, from Access to Oracle.

Advantages:

- Relational database supports mathematical set of operations like union, intersection, difference and Cartesian product. It also supports select, project, relational join and division operations.
- Relational database uses normalization structure which helps to achieve data independence more easily.
- Security control can also be implemented more effectively by imposing an authorization control on the sensitive attributes present in a table.
- Relational database uses a language which is easy and human readable.

Disadvantages:

- The response to a query becomes time-consuming and inefficient if the number of tables between which the relationships are established increases.

Implementation of Relational Database:

- Oracle
- Microsoft
- IBM
- My SQL
- PostgreSQL
- SQLite

Object-oriented database (1990s – present)

Object oriented database management system is that database system in which the data or information is presented in the form of objects, much like in object-oriented programming language. Furthermore, object-oriented DBMS also facilitate the user by offering transaction support, language for various queries, and indexing options. Also, these database systems have the ability to handle data efficiently over multiple servers.

Unlike relational database, object-oriented database works in the framework of real programming languages like JAVA or C++.

Advantages:

- If there are complex (many-to-many) relationships between the entities, the object-oriented database handles them much faster than any of the above discussed database models.
- Navigation through the data is much easier.
- Objects do not require assembly or disassembly hence saving the coding and execution time.

Disadvantages:

- Lower efficiency level when data or relationships are simple.
- Data can be accessible via specific language using a particular API which is not the case in relational databases.

Object-relational database (1990s – present)

Defined in simple terms, an object relational database management system displays a modified object-oriented user-display over the already implemented relational database management system. When various software interacts with this modified-database management system, they will customarily operate in a manner such that the data is assumed to be saved as objects.

The basic working of this database management system is that it translates the useful data into organized tables, distributed in rows and columns, and from then onwards, it manages data the same way as done in a relational database system. Similarly, when the data is to be accessed by the user, it is again translated from processed to complex form.

Advantages:

- Data remains encapsulated in object-relational database.
- Concept of inheritance and polymorphism can also be implemented in this database.

Disadvantages:

- Object relational database is complex.
- Proponents of relational approach believe simplicity and purity of relational model are lost.
- It is costly as well.
- Web enabled database (1990s – present):

- Web enabled database simply put a database with a web-based interface.

This implies that there can be a separation of concerns; namely, the web designer does not need to know the details about the DB's underlying design. Similarly, the DB designer needs to concern himself with the DB's web interface.

A web enabled database uses three layers to function: a presentation layer, a middle layer and the database layer.

Advantages:

- A web-enabled database allows users to get the information they need from a central repository on demand.
- The database is easy and simple to use.
- The data accessibility is easy via web-enabled database.

Disadvantages:

- Main disadvantage is that it can be hacked easily.
- Web enabled databases support the full range of DB operations, but in order to make them easy to use, they must be "dumped down".

Characteristics of DB System:

- **Manages Information**

A database always takes care of its information because information is always helpful for whatever work we do. It manages all the information that is required to us. Managing information by using a database, we become more deliberated user of our data.

- **Easy Operation Implementation**

All the operations like insert, delete, update, search etc. are carried out in a flexible and easy way. Database makes it very simple to implement these operations. A user with little knowledge can perform these operations. This characteristic of database makes it more powerful.

- **Multiple Views of Database**

Basically, a view is a **subset of the database**. A view is defined and devoted for a particular user of the system. Different users of the system may have different views of the same system. Every view contains only the data of interest to a user or a group of users. It is the responsibility of users to be aware of how and where the data of their interest is stored.

- **Data For Specific Purpose**

A database is designed for data of specific purpose. **For example**, a database of student management system is designed to maintain the record of student's marks, fees and attendance etc. This data has a specific purpose of maintaining student record.

- **It has Users of Specific Interest**

A database always has some indented group of users and applications in which these user groups are interested.

For example, in a library system, there are three users, official administration of the college, the librarian, and the students.

- **Represent Some Aspects of Real-World Applications**

A database represents some features of real-world applications. Any change in the real world is reflected in the database. If we have some changes in our real applications like railway reservation system then it will be reflected in database too.

For example, let us take an example of railway reservation system; we have in our mind some certain applications of maintaining records of attendance, waiting list, train arrival and departure time, certain day etc. related to each train.

- **Logical Relationship Between Records and Data**

A database gives a logical relationship between its records and data. So, a user can access various records depending upon the logical conditions by a single query from the database.

Advantages and Disadvantages of DBMS

Advantage:

- Reduction in data redundancy
- Reduction in inconsistency
- Sharing of Data
- Improvement in Data Security
- Maintenance of Data Integrity
- Efficient system
- Better interaction with User.

Disadvantage:

- Problem Association with Centralization
- Cost of software
- Cost of hardware
- Complexity of Backup & Recovery

DBMS Users

- Naive Users/ End user

- Application Programmers
- Sophisticated Users
- Specialized Users/DBA

End user:

Naïve user/end user as the name suggests that the Users have no knowledge about the database, but have to work very frequently with the system. But you must be wondering how is this possible.

Ok, so let's take some examples:

A user went to an ATM machine to withdraw some money, now it is not important to know about the SQL or the database for the same. that simply means it is not important to have the technical knowledge in this case, but the user can frequently use the system. So, Naive Users are categorised as Unsophisticated users who don't have technical knowledge, anyways they don't need to know technical knowledge for using the system. All they need to know is how to use the system.

Application Programmers:

Application Programmers are computer professionals who are specialised in the development of the software. This software is built by writing codes. Application Programmers know some programming language which is used to build software or applications. Also, they use some of the application development tools for the same. Their main job is to develop user interfaces which are easy to access by the end users. Applications or software should be user-friendly as it will also be used by Naive Users.

Sophisticated Users:

Naive users were in the category of Unsophisticated users, and now come the Sophisticated users who know about all the corners of the system. Sophisticated users know how to interact with the system. An example of a Sophisticated user is the Database Administrator like he/she knows the SQL language, which is used to write complex queries and fetch data from the database. Sophisticated users also know how to work with data analysis software as it helps in visualizing the data.

Specialized Users/DBA:
The name itself tells that these are the users with some special knowledge or access. Like these users specialized in DB Administrator. They don't only know how to write the SQL query for accessing the data, but they are also

aware of Computer-aided design systems. They have a strong knowledge base and are experts in running the systems.

They not only access the database, but they are the ones who create the database, they are the ones who create the tables, and they are the one who defines the rules for logging the data in the database.

They not only access the text i.e., normal data, but also plays with the multimedia data. Multimedia data is all about like video data, audio data, text, animation, graphics and images, etc.

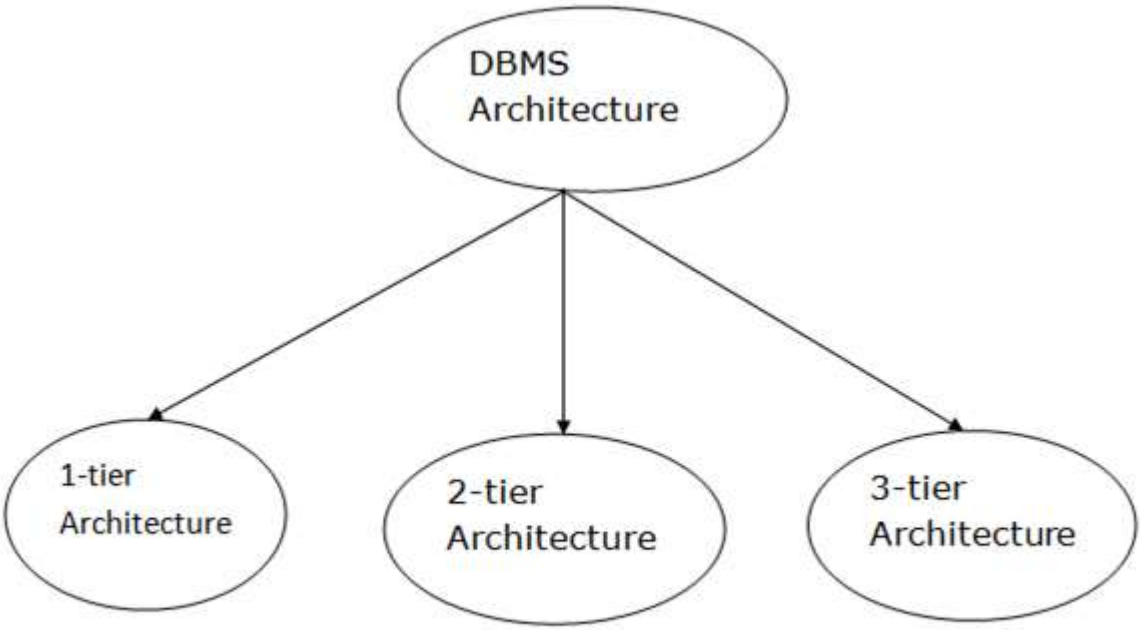
Specialized users mainly work on the projects in real-time which is responsible to log the data in a well-defined manner into the database. These projects will be deployed and also will be used until the next generation and so on.

The main one is the database administrator. Database Administrators are also known as DBA. Any organization or company dealing with any kind of data or having a database will also have a database administrator. DBA is one of the most important categories under the Classification of DBMS Users. Data administrators are the ones who are responsible for any activities being done on the database. Data Administrator has complete privilege over the entire database. He/she has full control of the central system.

DBMS Architecture:

The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks. The client/server architecture consists of many PCs and a workstation which are connected via the network.DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture



Database architecture can be seen as a single tier or multi-tier. But logically, database architecture is of two types like: **2-tier architecture** and **3-tier architecture**.

1-Tier Architecture

- 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.

2-Tier Architecture

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.

- To communicate with the DBMS, client-side application establishes a connection with the server side.

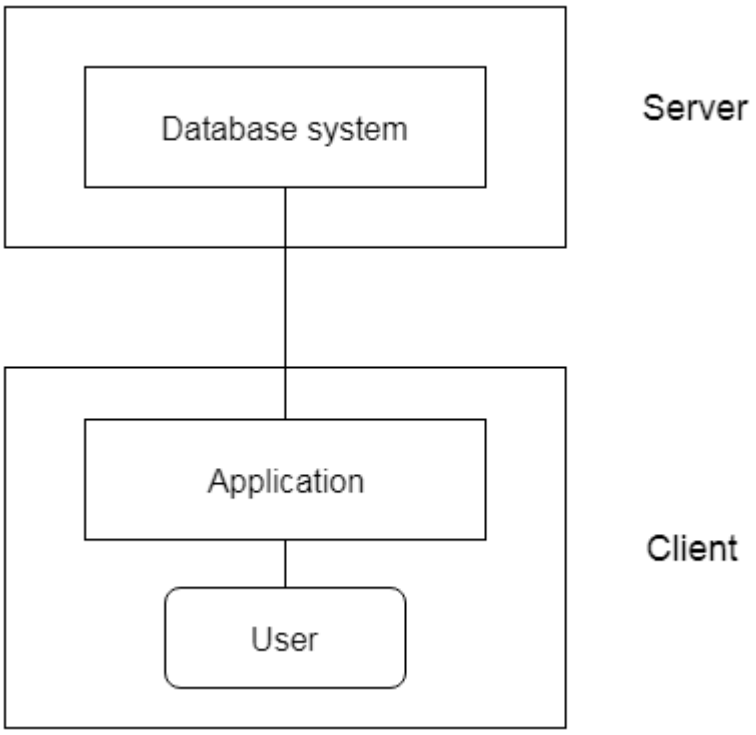
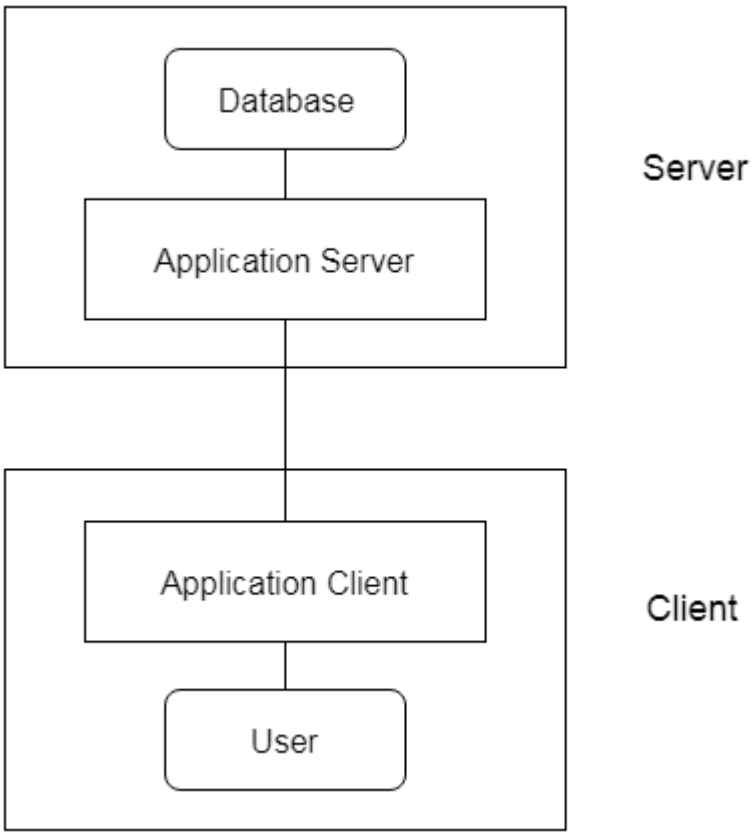


Fig: 2-tier Architecture

3-Tier Architecture

- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.
- The 3-Tier architecture is used in case of large web application.



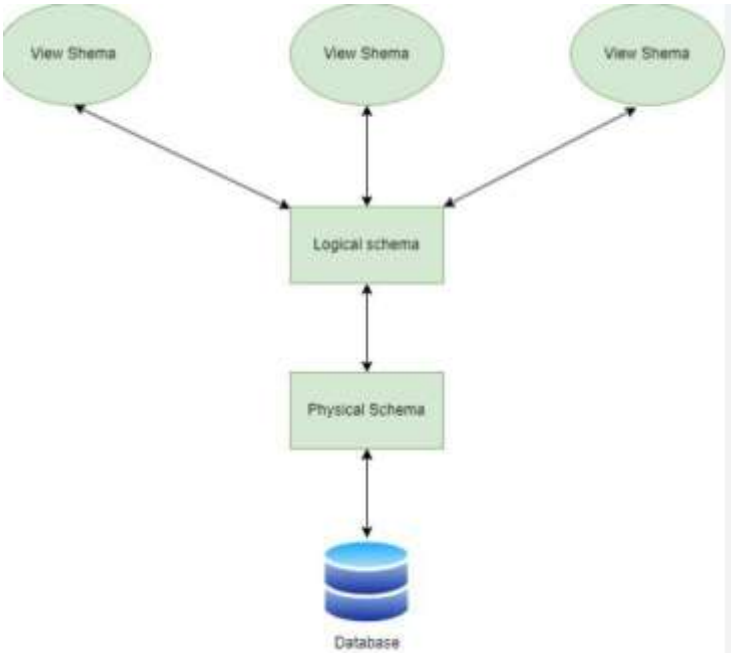
Database Schemas and Instances:

DBMS Schema

Definition of schema: Design of a database is called the schema. For example: An **employee** table in database exists with the following attributes:

```
EMP_NAME EMP_ID   EMP_ADDRESS EMP_CONTACT
-------- ------   ----------- -----------
```

This is the schema of the **employee** table. Schema defines the attributes of tables in the database. **Schema is of three types: Physical schema, logical schema and view schema.**



- **Physical Database Schema:**
    - A Physical schema defines, how the data or information is stored physically in the storage systems in the form of files & indices. This is the actual code or syntax needed to create the structure of a database, we can say that when we design a database at a physical level, it's called physical schema.
    - The Database administrator chooses where and how to store the data in the different blocks of storage.
- **Logical Database Schema:**
    - A logical database schema defines all the logical constraints that need to be applied to the stored data, and also describes tables, views, entity relationships, and integrity constraints.
    - The Logical schema describes how the data is stored in the form of tables & how the attributes of a table are connected.
    - Using **ER modelling** the relationship between the components of the data is maintained.
    - In logical schema different integrity constraints are defined in order to maintain the quality of insertion and update the data.

- **View Database Schema:**
    - It is a view level design which is able to define the interaction between end-user and database.
    - User is able to interact with the database with the help of the interface without knowing much about the stored mechanism of data in database.

Instances:

The Collection of information stored in a database at a particular moment is called as an instance of the database.

Database language:

"Database language" typically refers to a programming language or query language that is specifically designed for interacting with and managing databases.

1. **Data Definition Language (DDL):** DDL is used for defining the structure and schema of the database. It includes commands for creating, modifying, and deleting database objects like tables, indexes, and constraints. Common DDL commands include:
    - **`CREATE TABLE`**: Defines a new table with its columns and data types.
    - **`ALTER TABLE`**: Modifies an existing table's structure.
    - **`DROP TABLE`**: Deletes a table and its data.

DDL Example:

```
CREATE TABLE Employees (
```

```
    EmployeeID INT PRIMARY KEY,
    FirstName VARCHAR (50),
    LastName VARCHAR (50),
    Department VARCHAR (50)
);
```

2. **Data Manipulation Language (DML):** DML is used for interacting with the data stored in the database. It includes commands for querying, inserting, updating, and deleting data. Common DML commands include:
    - **`SELECT`**: Retrieves data from one or more tables.
    - **`INSERT INTO`**: Adds new data into a table.
    - **`UPDATE`**: Modifies existing data in a table.
    - **`DELETE FROM`**: Removes data from a table.

DML example:

```
-- Inserting data
INSERT INTO Employees (EmployeeID, FirstName, LastName, Department)
VALUES (1, 'akanksha', 'singh', 'assistant professor');

-- Updating data
UPDATE Employees
SET Department = 'Data Analyst'
WHERE EmployeeID = 1;

-- Querying data
SELECT FirstName, LastName, Department
FROM Employees
WHERE Department = 'Data Analyst';

-- Deleting data
```

```
DELETE FROM Employees
WHERE EmployeeID = 1;
```

Data Models

A Database model defines the logical design and structure of a database. It defines how data will be stored, accessed, and updated in a database management system. The database model sets the rule, relationships, constraints, etc. to define how data is stored in the database. There are different types of Database models and each one has its own set of features.

Types of Data Models:

Relational Data Model:

- In this model, data is organized in two-dimensional **tables** and the relationship is maintained by storing a common field.
- This model was introduced by **E.F Codd** in 1970, and since then it has been the most widely used database model.
- The basic structure of data in the relational model is **tables**. All the information related to a particular type is stored in **rows** of that table.
- Hence, tables are also known as **relations** in the relational model.
- Some of the most popular databases are based on this database model. For example, **Oracle**, **MySQL**, etc.

Advantages of the Relational Model

1. It's simple and easy to implement.
2. Poplar database software is available for this database model.
3. It supports SQL using which you can easily query the data.

Object-oriented Model:

- In this model, data is stored in the form of objects.
- The behaviour of the object-oriented database model is just like object-oriented programming.

- A very popular example of an Object Database management system or **ODBMS** is **MongoDB** which is also a NoSQL database.
- This database model is not mature enough as compared to the relational database model.

Advantages of the Object-oriented Model

1. It can easily support complex data structures, with relationships.
2. It also supports features like Inheritance, Encapsulation, etc.

Entity-relationship Model:

- In this database model, relationships are created by dividing objects of interest into entities and their characteristics into attributes.
- Different entities are related using **relationships**.
- ER Models are defined to represent the relationships in pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in a relational model.
- Let's take an **example**, if we have to design a School Database, then the **student** will be an **entity** with **attributes** *name*, *age*, *address,* etc. As an **Address** is generally complex, it can be another **entity** with **attributes** *street*, *Pin code*, *city,* etc, and there will be a relationship between them.

Advantages of the ER Model

1. It is easy to understand and design.
2. Using the ER model, we can represent data structures easily.
3. As the ER model cannot be directly implemented into a database model, it is just a step toward designing the relational database model.

Hierarchical Model:

- The hierarchical database model organizes data into a **tree-like structure**, with a **single root**, to which all the other data is linked.
- The hierarchy starts from the **Root** data, and expands like a tree, adding **child** nodes to the **parent** nodes.

- In this model, a child node will only have a **single parent node**.
- This model efficiently describes many real-world relationships like the *index of a book*, etc.
- IBM's Information Management System (IMS) is based on this model.
- Data is organized into a tree-like structure with a **one-to-many relationship** between two different types of data, for example, one **department** can have many **courses**, many **teachers**, and of course many **students**

advantages/Disadvantages of the Hierarchical Model

Here are a few points to mark the advantages and disadvantages of the Hierarchical database model:

1. Because it has one-to-many relationships between different types of data so it is easier and fast to fetch the data.
2. But the Hierarchical model is less flexible.
3. And it doesn't support many-to-many relationships.

Network Model:

- The Network Model is an extension of the Hierarchical model.
- In this model, data is organized more like a **graph**, and allowed to have *more than one parent node*.
- This database model uses many-to-many data relationships.
- Integrated Data Store (IDS) is based on this database model.
- This was the most widely used database model before Relational Model was introduced.
- The implementation of the Network model is complex, and it's very difficult to maintain it.
- The Network model is difficult to modify also.

Advantages of the Network Model

1. It supports complex relationships
2. It allows more flexibility