**UNIT- I(PART-I)**

**What is DBMS?**

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

**(or)**

A **database-management system** (DBMS) is a collection of interrelated data and a set of programs to access those data.

**The primary goal of a DBMS** is to provide a way to store and retrieve database information that is both convenient and efficient.

**What is Database?**

**Database:**Collection of related data (logically coherent)

Database systems are designed to manage large bodies of information.

To find out what database is, we have to start from data, which is the basic building block of any DBMS.

**Data**: Facts, figures, statistics etc. having no particular meaning (e.g. 1, ABC, 19 etc).

**Record**: Collection of related data items, e.g. in the above example the three data items had no meaning. But

|  |  |  |
| --- | --- | --- |
| **Roll** | **Name** | **age** |
| 1 | ABC | 19 |

if we organize them in the following way, then they collectively represent meaningful information.

**Table** or **Relation**: Collection of related records.

|  |  |  |
| --- | --- | --- |
| **Roll** | **Name** | **Age** |
| 1 | ABC | 19 |
| 2 | DEF | 20 |
| 3 | GHI | 22 |

**Key Features of DBMS**

* **Data modeling:** A DBMS provides tools for creating and modifying data models, which define the structure and relationships of the data in a database.
* **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.
* **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.
* **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.
* **Backup and recovery:** A DBMS provides mechanisms for backing up and recovering the data in the event of a system failure.
* **DBMS can be classified into two types:** Relational Database Management System (RDBMS) and Non-Relational Database Management System (NoSQL or Non-SQL)
* **RDBMS:**Data is organized in the form of tables and each table has a set of rows and columns. The data are related to each other through primary and foreign keys.
* **NoSQL:**Data is organized in the form of key-value pairs, documents, graphs, or column-based. These are designed to handle large-scale, high-performance scenarios.

**Types of DBMS**

1. **Relational Database Management System (RDBMS):** Data is organized into tables (relations) with rows and columns, and the relationships between the data are managed through primary and foreign keys. SQL (Structured Query Language) is used to query and manipulate the data.
2. **NoSQL DBMS:** Designed for high-performance scenarios and large-scale data, NoSQL databases store data in various non-relational formats such as key-value pairs, documents, graphs, or columns.
3. **Object-Oriented DBMS (OODBMS):** Stores data as objects, similar to those used in object-oriented programming, allowing for complex data representations and relationships

**Characteristics of DBMS**

Some well-known characteristics are present in the DBMS (Database Management System). These are explained below.

1. **Real World Entity**

* The Database can store information such as the cost of vegetables, milk, bread, etc. In DBMS (Database Management System), the entities look like real-world entities.
* For example, if we want to create a student database, we need some entity. Any student stores their data.
* In the Database, then, it should be the real-world entity. The most commonly used properties in the student database are name, age, gender, roll number, etc.

2. **Self-explaining nature**

* In DBMS (Database Management System), the Database contains another database, and another database also contains metadata.
* Here the term metadata means data about data.
* For example, in a school database, the total number of rows and the table's name are examples of metadata.
* So the self-explaining nature means the Database explains all the information automatically itself. This is because, in the Database, all the data are stored in a structured format.

3. **Atomicity of Operations (Transactions)**

* Here, atomicity means either the operation should be performed or not performed. i.e., it should complete the operation on 0% or 100%.
* For example, every bank has its own Database, and the Database contains all the information about its customers. Let transaction is the most common atomic operation of the bank. If Sona wants to transfer 1000 rupees to the Archita account, it is possible with the help of the atomicity feature of the Database. If there is a problem in the Archita account, if there is a problem in the atomicity of the Database, then the money will be deducted from the Sona account but not credited to the Archita account.
* The Database has the feature of atomicity then; such transactions have not occurred at all, and if the transaction fails, then the money will automatically return to the sender account.
* Basically, for a successful transaction, the total operation depends on the Database. If the Database works perfectly, the transaction will be successful, and if the Database fails, the whole banking server will be down.

4. **Concurrent Access without Anomalies**

* Here the term anomalies mean multiuser can access the Database and fetch the information without any problem.
* For a better understanding, let's take the example of a bank again. Let Sonu give his ATM card to his sister Archita and tell her to withdraw 5000 from the ATM. At the same time, Sonu transferred 2000 rupees to his brother Monu. At the same time, both operations perform successfully. Initially, Sonu had 10000 rupees in his bank account. After both transactions, i.e., transfer and withdraw, when Sonu checks his bank balance, it shows 3000 rupees. This error-free updation of bank balance is possible with the help of the concurrent feature of the Database.
* Thus here we see that concurrent is a great feature of the Database.

5. **Stores Any Kind of Structured Data**

* The Database has the ability to store the data in a structured format.
* DBMS has the ability to store any type of data that exists in the real world, and these data are structured way. It is another type of very important characteristic of DBMS.

6. **Integrity**

* Here the term integrity means the data should be correct and consistent in nature. Let's understand this by taking an example.
* Let's say there is a bank named ABC bank, and ABC bank has its own Database for the storage of its customer data. If we try to enter the account details of ABC bank and the account details are not available in the bank, then the Database gives the incorrect output. However, if a customer changes their address but the new address is not updated in the Database, it is called data inconsistency.
* So the data available in the Database should be correct as well as consistent.

7. **Ease of Access (The DBMS Queries)**

* The file and folder system was used to store the data before the DBMS came to the market.
* Searching for the student's name was a very difficult task at that time. This is because **every search operation is done manually in the file and folder system**. But when DBMS comes into the market, it is very easy to access the Database.
* In DBMS, we can search any kind of stored data by applying a simple search operation query. It is so much faster than manual searching.
* In DBMS, there is a CRUD operation ( here CRUD means Create, Read, Update & Delete) by which we can implement all the types of query in the Database.

8. **SQL and No-SQL Databases**

* There are two types of databases (not DBMS): SQL and No-SQL.
* The SQL databases store the data in the form of Tables, i.e., rows and columns. The No-SQL databases can store data in any form other than a table. For instance: the very popular MongoDB stores the data in the form of JSON (JavaScript Object Notation).
* The availability of SQL and No-SQL databases allows us to choose the method of storing the data as well.

9. **ACID Properties**

* The DBMS follows certain properties to maintain consistency in the Database. These properties are usually termed ACID Properties.
* ACID stands for Atomicity, Consistency, Isolation, and Durability.
* We have already talked about atomicity and consistency. **Atomicity** means the transaction should either be 0% or 100% completed, and **consistency** means that the change in data should be reflected everywhere in a database.
* **Isolation** means that multiple transactions can occur independently without the interference of some other transactions.
* **Durability** means that the chances of a successful atomic transaction, i.e., a transaction that has been 100% completed, should reflect in the Database.

10. **Security**

* The Database should be accessible to the users in a limited way.
* Unauthorized users should not be allowed to access the Database.
* Authentication: The DBMS has authentication for various users that directly refers to the limit to which the user can access the Database. Authentication means the process of laughing in of the user only with the rights that he/she has been authorized to. For instance, in any organization, the admin has access to make changes to the Database of the organization as some new employee might have joined the organization or someone might have left it. However, the employees have access only to their personal profiles and can make changes to them only. They cannot access the Database of any other employee or the organization as a whole.

****Difference between File System and DBMS(or)** Database Systems versus File Systems**

Keeping organizational information in a **file-processing system** has a number of **major disadvantages:**

*•* **Data redundancy and inconsistency**. Since different programmers create the files and application programs over a long period, the various files are likely to have different formats and the programs may be written in several programming languages. Moreover, the same information may be duplicated in several places (files). For example, the address and telephone number of a particular customer may appear in a file that consists of savings-account records

and in a file that consists of checking-account records. This redundancy leads to higher storage and access cost. In addition, it may lead to **data inconsistency**; that is, the various copies of the same data may no longer agree. For example, a changed customer address may be reflected in savings-account records but not elsewhere in the system.

*•* **Difficulty in accessing data**. Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer asks the data-processing department to generate such a list. Because the designers of the original system did not anticipate this request, there is no application program on hand to meet it. There is, however, an application program to generate the list of *all* customers. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of $10,000 or more. As expected, a program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory. The point here is that con nventional file-processing environments do not allow needed data to be retrieved in a convenient and efficient manner. More responsive data-retrieval systems are required for general use.

*•* **Data isolation**. Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.

*•* **Integrity problems**. The data values stored in the database must satisfy certain types of **consistency constraints**. For example, the balance of a bank account may never fall below a prescribed amount (say, $25). Developers enforce these constraints in the system by adding appropriate code in the various application programs. However, when new constraints are added, it is difficult to change the programs to enforce them. The problem is compounded when constraints involve several data items from different files.

*•* **Atomicity problems**. A computer system, like any other mechanical or electrical device, is subject to failure. In many applications, it is crucial that, if afailure occurs, the data be restored to the consistent state that existed prior to the failure. Consider a program to transfer $50 from account *A* to account *B*. If a system failure occurs during the execution of the program, it is possible that the $50 was removed from account *A* but was not credited to account *B*, resulting in an inconsistent database state. Clearly, it is essential to database consistency that either both the credit and debit occur, or that neither occur. That is, the funds transfer must be *atomic*—it must happen in its entirety or not at all. It is difficult to ensure atomicity in a conventional file-processing system.

*•* **Concurrent-access anomalies**. For the sake of overall performance of the system and faster response, many systems allow multiple users to update the data simultaneously. In such an environment, interaction of concurrent updates may result in inconsistent data. Consider bank account *A*, containing $500. If two customers withdraw funds (say $50 and $100 respectively) from account *A* at about the same time, the result of the concurrent executions may leave the account in an incorrect (or inconsistent) state. Suppose that the programs executing on behalf of each withdrawal read the old balance, reduce that value by the amount being withdrawn, and write the result back. If the two programs run concurrently, they may both read the value $500, and write back $450 and $400, respectively. Depending on which one writes the values last, the account may contain either $450 or $400, rather than the correct value of $350. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult to provide because data may be accessed by many different application programs that have not been coordinated previously.

*•* **Security problems**. Not every user of the database system should be able to access all the data. For example, in a banking system, payroll personnel need to see only that part of the database that has information about the various bank employees. They do not need access to information about customer accounts. But, since application programs are added to the system in an ad hoc

manner, enforcing such security constraints is difficult

| **Basics** | **File System** | **DBMS** |
| --- | --- | --- |
| **Structure** | The file system is a way of arranging the files in a storage medium within a computer. | DBMS is software for managing the database. |
| **Data Redundancy** | Redundant data can be present in a file system. | In DBMS there is no redundant data. |
| **Backup and Recovery** | It doesn’t provide Inbuilt mechanism for backup and recovery of data if it is lost. | It provides in house tools for backup and recovery of data even if it is lost. |
| **Query processing** | There is no efficient query processing in the file system. | Efficient query processing is there in DBMS. |
| **Consistency** | There is less data consistency in the file system. | There is more data consistency because of the process of [normalization](https://www.geeksforgeeks.org/normal-forms-in-dbms/" \t "_blank). |
| **Complexity** | It is less complex as compared to DBMS. | It has more complexity in handling as compared to the file system. |
| **Security Constraints** | File systems provide less security in comparison to DBMS. | DBMS has more security mechanisms as compared to file systems. |
| **Cost** | It is less expensive than DBMS. | It has a comparatively higher cost than a file system. |
| **Data Independence** | There is no data independence. | In DBMS [data independence](https://www.geeksforgeeks.org/what-is-data-independence-in-dbms/" \t "_blank) exists, mainly of two types:  1) [Logical Data Independence](https://www.geeksforgeeks.org/physical-and-logical-data-independence/" \t "_blank).  2)Physical Data Independence. |
| **User Access** | Only one user can access data at a time. | Multiple users can access data at a time. |
| **Meaning** | The users are not required to write procedures. | The user has to write procedures for managing databases |
| **Sharing** | Data is distributed in many files. So, it is not easy to share data. | Due to centralized nature data sharing is easy |
| **Data Abstraction** | It give details of storage and representation of data | It hides the internal details of [Database](https://www.geeksforgeeks.org/what-is-database/" \t "_blank) |
| **Integrity Constraints** | Integrity Constraints are difficult to implement | Integrity constraints are easy to implement |
| **Attribute*s*** | To access data in a file , user requires attributes such as file name, file location. | No such attributes are required. |
| **Example** | [Cobol](https://www.geeksforgeeks.org/how-to-install-cobol-on-macos/" \t "_blank), [C++](https://www.geeksforgeeks.org/c-plus-plus/" \t "_blank) | [Oracle](https://www.geeksforgeeks.org/oracle-interview-experience-8/" \t "_blank), [SQL Server](https://www.geeksforgeeks.org/sql-tutorial/" \t "_blank) |

# **Database Users**

A Database User is defined as a person who interacts with data daily, updating, reading, and modifying the given data. Database users can access and retrieve data from the database through the Database Management System (DBMS) applications and interfaces.

**Types of Database Users**

Database users are categorized based on their interaction with the database. There are seven types of database users in DBMS. Below mentioned are the types of database users:

**1. Database Administrator (DBA)**

A Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of the database. The DBA will then create a new account ID and password for the user if he/she needs to access the database. DBA is also responsible for providing security to the database and he allows only authorized users to access/modify the database. DBA is responsible for problems such as security breaches and poor system response time.

* DBA also monitors the recovery and backup and provides technical support.
* The DBA has a DBA account in the DBMS which is called a system or superuser account.
* DBA repairs damage caused due to hardware and/or software failures.
* DBA is the one having privileges to perform [DCL (Data Control Language)](https://www.geeksforgeeks.org/dcl-full-form/" \t "_blank)operations such as [GRANT and REVOKE](https://www.geeksforgeeks.org/difference-between-grant-and-revoke/" \t "_blank), to allow/restrict a particular user from accessing the database.

**2. Naive / Parametric End Users**

Parametric End Users are the unsophisticated who don’t have any DBMS knowledge but they frequently use the database applications in their daily life to get the desired results. For example, Railway’s ticket booking users are naive users. Clerks in any bank is a naive user because they don’t have any DBMS knowledge but they still use the database and perform their given task.

**3. A System Analyst**

A system Analyst is a user who analyzes the requirements of parametric end users. They check whether all the requirements of end users are satisfied.

**4. Sophisticated Users**

Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database. They can develop their own database applications according to their requirement. They don’t write the program code but they interact the database by writing SQL queries directly through the query processor.

**5. Database Designers**

Data Base Designers are the users who design the structure of database which includes tables, indexes, views, [triggers](https://www.geeksforgeeks.org/sql-triggers/" \t "_blank), stored procedures and constraints which are usually enforced before the database is created or populated with data. He/she controls what data must be stored and how the data items to be related. It is the responsibility of Database Designers to understand the requirements of different user groups and then create a design which satisfies the need of all the user groups.

**6. Application Programmers**

Application Programmers also referred as System Analysts or simply Software Engineers, are the back-end programmers who writes the code for the application programs. They are the computer professionals. These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc. Application programmers design, debug, test, and maintain set of programs called “canned transactions” for the Naive (parametric) users in order to interact with database.

### ****7. Casual Users / Temporary Users****

Casual Users are the users who occasionally use/access the database but each time when they access the database they require the new information, for example, Middle or higher level manager.

### ****8. Specialized users****

Specialized users are sophisticated users who write specialized database application that does not fit into the traditional data-processing framework. Among these applications are computer aided-design systems, knowledge-base and expert systems etc.

### Advantages of DBMS

The advantages of the DBMS are explained below −

* **Redundancy problem can be solved.**

In the File System, duplicate data is created in many places because all the programs have their own files which create data redundancy resulting in wastage of memory. In DBMS, all the files are integrated in a single database. So there is no chance of duplicate data.

For example: A student record in a library or examination can contain duplicate values, but when they are converted into a single database, all the duplicate values are removed.

* **Has a very high security level.**

Data security level is high by protecting your precious data from unauthorized access. Only authorized users should have the grant to access the database with the help of credentials.

* **Presence of Data integrity.**

Data integrity makes unification of so many files into a single file. DBMS allows data integrity which makes it easy to decrease data duplicity Data integration and reduces redundancy as well as data inconsistency.

* **Support multiple users.**

DBMS allows multiple users to access the same database at a time without any conflicts.

* **Avoidance of inconsistency.**

DBMS controls data redundancy and also controls data consistency. Data consistency is nothing but if you want to update data in any files then all the files should not be updated again.

In DBMS, data is stored in a single database so data becomes more consistent in comparison to file processing systems.

* **Shared data**

Data can be shared between authorized users of the database in DBMS. All the users have their own right to access the database. Admin has complete access to the database. He has a right to assign users to access the database.

* **Enforcement of standards**

As DBMS have central control of the database. So, a DBA can ensure that all the applications follow some standards such as format of data, document standards etc. These standards help in data migrations or in interchanging the data.

* **Any unauthorized access is restricted**

Unauthorized persons are not allowed to access the database because of security credentials.

* **Provide backup of data**

Data loss is a big problem for all the organizations. In the file system users have to back up the files in regular intervals which lead to waste of time and resources.

DBMS solves this problem of taking backup automatically and recovery of the database.

### **Disadvantages of DBMS:**

The disadvantages of DBMS are as follows:

* **Complexity**

The provision of the functionality that is expected of a good DBMS makes the DBMS an extremely complex piece of software. Database designers, developers, database administrators and end-users must understand this functionality to take full advantage of it.

Failure to understand the system can lead to bad design decisions, which leads to a serious consequence for an organization.

* **Size**

The functionality of DBMS makes use of a large piece of software which occupies megabytes of disk space.

* **Performance**

Performance may not run as fast as desired.

* **Higher impact of a failure**

The centralization of resources increases the vulnerability of the system because all users and applications rely on the availability of DBMS, the failure of any component can bring operation to halt.

* **Cost of DBMS**

The cost of DBMS varies significantly depending on the environment and functionality provided. There is also the recurrent annual maintenance cost.

# **Database Applications**

Nowadays, any business that has small or large amounts of data needs a database to store and manage the information. The database is an easy, reliable, secure, and efficient way to maintain business information. There are many applications where databases are used.

Databases are widely used. Here are some representative applications:

• **Banking:** For customer information, accounts, and loans, and banking transactions.

• **Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner—terminals situated around the world accessed the central database system through phone lines and other data networks.

• **Universities:** For student information, course registrations, and grades.

• **Credit card transactions:** For purchases on credit cards and generation of monthly statements.

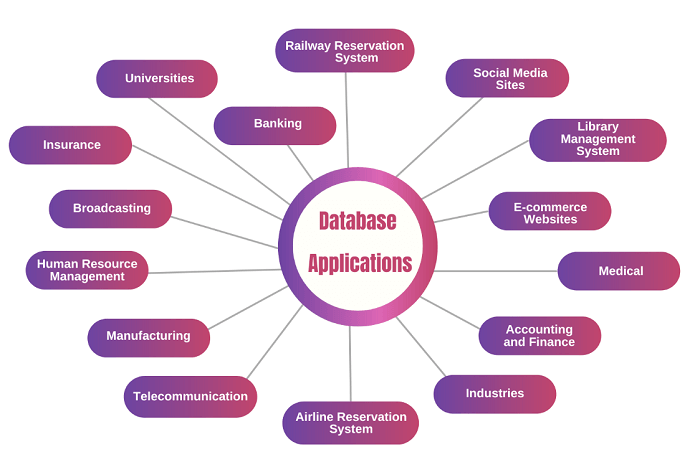
• **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about he communication networks.

• **Finance:** For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds.

• **Sales:** For customer, product, and purchase information.

• **Manufacturing:** For management of supply chain and for tracking production of items in factories, inventories of items in warehouses/stores, and orders for items.

• **Human resources:** For information about employees, salaries, payroll taxes and benefits, and for generation of paychecks.



# **Data Models in DBMS**

A **data model** is a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

A Data Model in Database Management System (DBMS)  is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

**Types of Relational Models**

It is basically classified into 3 types:-

1. Conceptual Data Model (or) ER-Mdel
2. Representational Data Model(or) Relational model
3. Physical Data Model

**1. Conceptual Data Model**

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the [entity/relationship model (ER model)](https://www.geeksforgeeks.org/introduction-of-er-model/). The E/R model specializes in entities, relationships, and even attributes that are used by database designers. In terms of this concept, a discussion can be made even with non-computer science(non-technical) users and stakeholders, and their requirements can be understood.

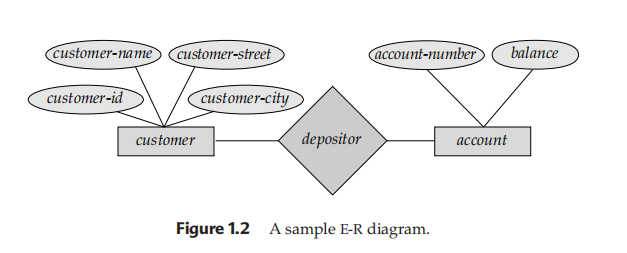
**Entity-Relationship Model( ER Model):** It is a high-level data model which is used to define the data and the relationships between them. It is basically a conceptual design of any database which is easy to design the view of data.

**The entity-relationship (E-R) data mode**l is based on a perception of a real world that consists of a collection of basic objects, called entities, and of relationships among these objects. **An entity** is a “thing” or “object” in the real world that is distinguishable from other objects. For example, each person is an entity, and bank accounts can be considered as entities.

**Entities** are described in a database by a set of **attributes**. For example, the attributes account-number and balance may describe one particular account in a bank, and they form attributes of the account entity set. Similarly, attributes customer-name, customer-street address and customer-city may describe a customer entity

Components of ER Model:

1. **[Entity:](https://www.geeksforgeeks.org/difference-between-entity-entity-set-and-entity-type/)** An entity is referred to as a real-world object. It can be a name, place, object, class, etc. These are represented by a rectangle in an ER Diagram.
2. **[Attributes:](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/)**An attribute can be defined as the description of the entity. These are represented by Ellipse in an ER Diagram. It can be Age, Roll Number, or Marks for a Student.
3. **[Relationship:](https://www.geeksforgeeks.org/relationships-in-er-model/)**Relationships are used to define relations among different entities. Diamonds and Rhombus are used to show Relationships.



**2. Representational Data Model(or) Relational Model**

**The relational model** uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name.

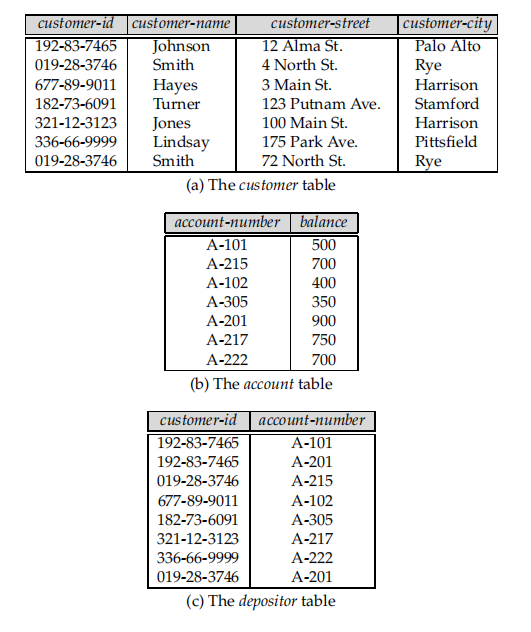
**The relational model** is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type

**The relational data model** is the most widely used data model, and a vast majority of current database systems are based on the relational model

**The relational model is** at a lower level of abstraction than the E-R model. Database designs are often carried out in the E-R model, and then translated to the relational model;

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a [Relational model](https://www.geeksforgeeks.org/relational-model-in-dbms/). The relational Model consists of [Relational Algebra](https://www.geeksforgeeks.org/introduction-of-relational-algebra-in-dbms/) and [Relational Calculus](https://www.geeksforgeeks.org/tuple-relational-calculus-trc-in-dbms/). In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.

The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model



**3. Physical Data Model**

 The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other. Here, we basically save tables in memory so they can be accessed efficiently. In order to come up with a good physical model, we have to work on the relational model in a better way. [Structured Query Language (SQL)](https://www.geeksforgeeks.org/structured-query-language/) is used to practically implement Relational Algebra.

This Data Model describes **HOW** the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.

**Characteristics of a physical data model:**

* The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
* Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
* Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
* Columns should have exact datatypes, lengths assigned and default values.
* Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined

**Some Other Data Models**

**1. Hierarchical Model**

The [hierarchical Model](https://www.geeksforgeeks.org/hierarchical-model-in-dbms/) is one of the oldest models in the data model which was developed by IBM, in the 1950s. In a hierarchical model, data are viewed as a collection of tables, or we can say segments that form a hierarchical relation. In this, the data is organized into a tree-like structure where each record consists of one parent record and many children. Even if the segments are connected as a chain-like structure by logical associations, then the instant structure can be a fan structure with multiple branches. We call the illogical associations as directional associations.

**2. Network Model**

The [Network Model](https://www.geeksforgeeks.org/network-model-in-dbms/) was formalized by the Database Task group in the 1960s. This model is the generalization of the hierarchical model. This model can consist of multiple parent segments and these segments are grouped as levels but there exists a logical association between the segments belonging to any level. Mostly, there exists a many-to-many logical association between any of the two segments.

1. **Object-Oriented Data Model**

The object-oriented model can be seen as extending the E-R model with notionsns), and object identity. The **object-relational data model** combines features of the object-oriented datamodel and relational data model.

In the [Object-Oriented Data Model](https://www.geeksforgeeks.org/basic-object-oriented-data-model/), data and their relationships are contained in a single structure which is referred to as an object in this data model. In this, real-world problems are represented as objects with different attributes. All objects have multiple relationships between them. Basically, it is a combination of Object Oriented programming and a Relational Database Model.

**4. Float Data Model**

The float data model basically consists of a two-dimensional array of data models that do not contain any duplicate elements in the array. This data model has one drawback it cannot store a large amount of data that is the tables can not be of large size.

**5. Context Data Model**

The Context data model is simply a data model which consists of more than one data model. For example, the Context data model consists of ER Model, Object-Oriented Data Model, etc. This model allows users to do more than one thing which each individual data model can do.

**6. Semi-Structured Data Model**

Semi-Structured data models deal with the data in a flexible way. Some entities may have extra attributes and some entities may have some missing attributes. Basically, you can represent data here in a flexible way.

**Advantages of Data Models**

1. Data Models help us in representing data accurately.
2. It helps us in finding the missing data and also in minimizing Data Redundancy.
3. Data Model provides data security in a better way.
4. The data model should be detailed enough to be used for building the physical database.
5. The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.

**Disadvantages of Data Models**

1. In the case of a vast database, sometimes it becomes difficult to understand the data model.
2. You must have the proper knowledge of [SQL](https://www.geeksforgeeks.org/structured-query-language/) to use physical models.
3. Even smaller change made in structure require modification in the entire application.
4. There is no set data manipulation language in DBMS.
5. To develop Data model one should know physical data stored characteristics.

**Database Schema**

* A database schema is a **logical representation of data** that shows how the data in a database should be stored logically. It shows how the data is organized and the relationship between the tables.
* Database schema contains table, field, views and relation between different keys like [primary key](https://www.geeksforgeeks.org/primary-key-in-dbms/), [foreign key](https://www.geeksforgeeks.org/foreign-key-constraint-in-sql/).
* Data are stored in the form of files which is unstructured in nature which makes accessing the data difficult. Thus to resolve the issue the data are organized in structured way with the help of database schema.
* Database schema provides the organization of data and the relationship between the stored data.
* Database schema defines a set of guidelines that control the database along with that it provides information about the way of accessing and modifying the data.

**Types of Database Schemas:**

**There are 3 types of database schema**

**Physical Dtabase Schema**

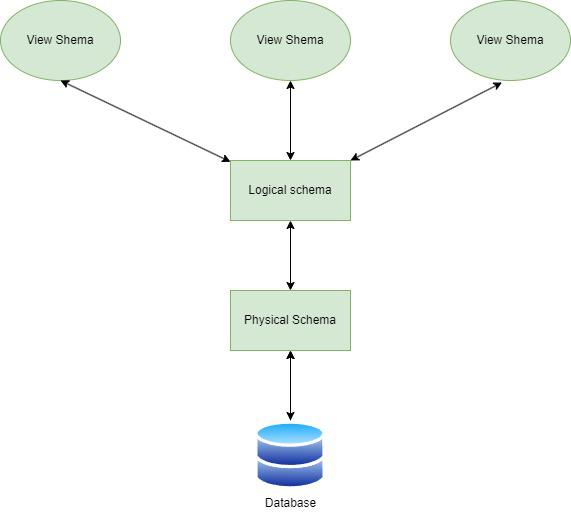
* A Physical schema defines, **how** the data or information is stored physically in the storage systems in the form of files & indices. This is the actual code or syntax needed to create the structure of a database, we can say that when we design a database at a physical level, it’s called physical 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.



*Fig:Three Layer Schema Design*

## DBMS Instance?

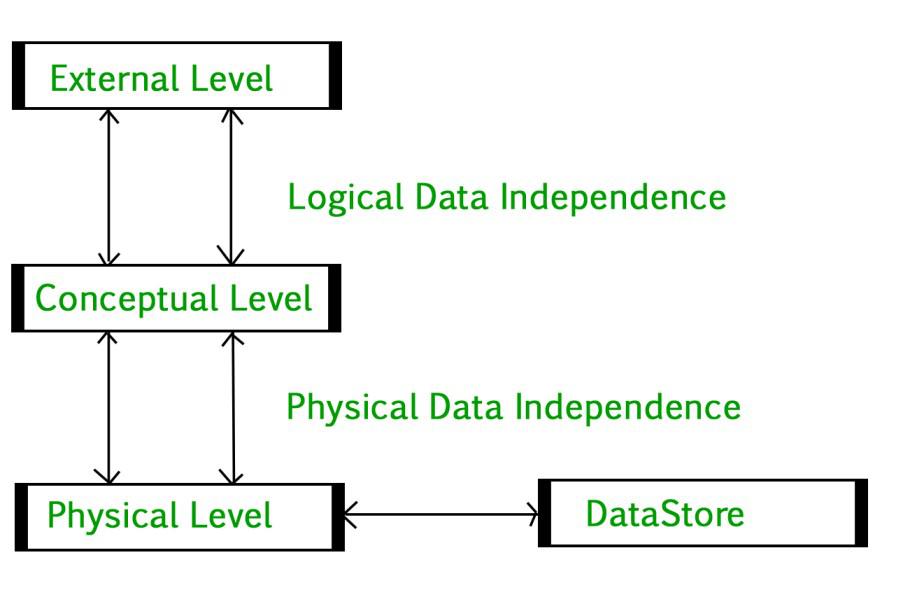
In DBMS, the data is stored for a particular amount of time and is called an instance of the database. The database schema defines the attributes of the database in the particular DBMS. The value of the particular attribute at a particular moment in time is known as an instance of the DBMS.

For example, in the above example, we have taken the example of the attribute of the schema. In this example, each table contains two rows or two records. In the above schema of the table, the employee table has some instances because all the data stored by the table have some instances.

**Three-tier schema Architecture**

The 3-tier architecture is a commonly used architectural approach in Database Management Systems (DBMSs) for the design and development of applications that work with databases. The 3-tier architecture divides an application’s components into three tiers or layers. Each layer has its own set of responsibilities.

DBMS 3-Tier architecture divides the complete system into three inter-related but independent modules as shown below:



*DBMS 3-tier architecture*

* **Physical Level:** At the physical level, the information about the location of database objects in the data store is kept. Various users of DBMS are unaware of the locations of these objects.In simple terms,physical level of a database describes how the data is being stored in secondary storage devices like disks and tapes and also gives insights on additional storage details.
* **Conceptual Level:**At conceptual level, data is represented in the form of various database tables. For Example, STUDENT database may contain STUDENT and COURSE tables which will be visible to users but users are unaware of their storage.Also referred as logical schema,it describes what kind of data is to be stored in the database.
* **External Level:** An external level specifies a view of the data in terms of conceptual level tables.  Each external level view is used to cater to the needs of a particular category of users. For Example, FACULTY of a university is interested in looking course details of students, STUDENTS are interested in looking at all details related to academics, accounts, courses and hostel details as well. So, different views can be generated for different users. The main focus of external level is data abstraction.

**Data Independence**

Data independence means a change of data at one level should not affect another level. Two types of data independence are present in this architecture:

* **Physical Data Independence:** Any change in the physical location of tables and indexes should not affect the conceptual level or external view of data. This data independence is easy to achieve and implemented by most of the DBMS.
* **Conceptual Data Independence:** The data at conceptual level schema and external level schema must be independent. This means a change in conceptual schema should not affect external schema. e.g.; Adding or deleting attributes of a table should not affect the user’s view of the table. But this type of independence is difficult to achieve as compared to physical data independence because the changes in conceptual schema are reflected in the user’s view.

# **Structure of Database Management System**

A Database Management System (DBMS) is software that allows users to define, store, maintain, and manage data in a structured and efficient manner. It acts as an intermediary between data and users, allowing disparate data from different applications to be managed. A DBMS simplifies the complexity of data processing by providing tools to organize data, ensure its integrity, and prevent unauthorized access or loss of data.

A Database Management System (DBMS) is software that allows users to define, store, maintain, and manage data in a structured and efficient manner. It acts as an intermediary between data and users, allowing disparate data from different applications to be managed. DBMS simplifies the complexity of data processing by providing tools to organize data, ensure its integrity, and prevent unauthorized access or loss of data.

**Data theft can be carried out by, among others:**

* **Hacking and exploiting:** Attackers can use DBMS security gaps to access unauthorized sensitive data.
* **Insider threats:** Employees or contractors compromise privileged access to information.
* **Phishing and social engineering:**These are techniques that will trick the authorized user into revealing the login credentials to enable intrusion.
* **Malware and ransomware attacks:** These are malware that make database security vulnerable to attack, thus giving access to attackers to steal data or lock down data until some amount of ransom is paid.

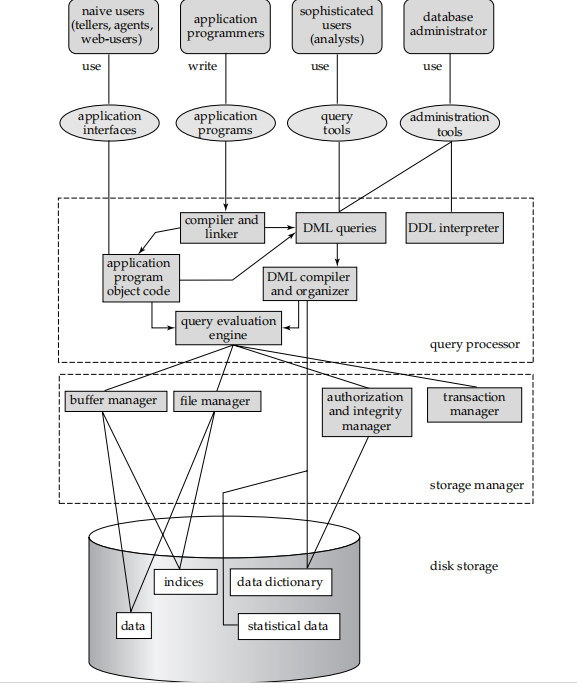
Data theft prevention is not only an issue in sensitive information matters but also for building trust between businesses and clients. Controls over access, periodic audits, real-time monitoring of activities done through the database are effective measures one could consider to reduce the risk. Also, following cyber security protocols and periodic inundation of database systems will reduce most of the vulnerabilities.

**Database Architecture vs. Tier Architecture**

Structure of  Database Management System is also referred to as  Overall System Structure or Database Architecture but it is different from the tier architecture of Database.

**Components of a Database System**

Query Processor, Storage Manager, and Disk Storage. These are explained as following below.



**Storage Manager**

A *storage manager* is a program module that provides the interface between the lowlevel data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for the interaction with the file manager. The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system. The storage manager translates the various DML statements into low-level file-system commands. Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

*•* **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.

*•* **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.

*•* **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.

*•* **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory. The storage manager implements several data structures as part of the physical system implementation:

*•* **Data files**, which store the database itself.

*•* **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.

*•* **Indices**, which provide fast access to data items that hold particular values

**The Query Processor**

The query processor components include

*•* **DDL interpreter**, which interprets DDL statements and records the definitions

in the data dictionary.

*•* **DML** compiler, which translates DML statements in a query language into an

evaluation plan consisting of low-level instructions that the query evaluation

engine understands.

A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives.

*•* **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

Figure shows these components and the connections among them.

### ****3. Disk Storage:****

It contains the following components:

* **Data Files:**It stores the data.
* **Data Dictionary:**It contains the information about the structure of any database object. It is the repository of information that governs the metadata.
* **Indices:**It provides faster retrieval of data item.

## Levels of DBMS Architecture

The structure of a Database Management System (DBMS) can be divided into three main components: the Internal Level, the Conceptual Level, and the External Level.

### Internal Level:

This level represents the physical storage of data in the database. It is responsible for storing and retrieving data from the storage devices, such as hard drives or solid-state drives. It deals with low-level implementation details such as data compression, indexing, and storage allocation.

### Conceptual Level:

This level represents the logical view of the database. It deals with the overall organization of data in the database and the relationships between them. It defines the data schema, which includes tables, attributes, and their relationships. The conceptual level is independent of any specific DBMS and can be implemented using different DBMSs.

### External Level:

This level represents the user’s view of the database. It deals with how users access the data in the database. It allows users to view data in a way that makes sense to them, without worrying about the underlying implementation details. The external level provides a set of views or interfaces to the database, which are tailored to meet the needs of specific user groups

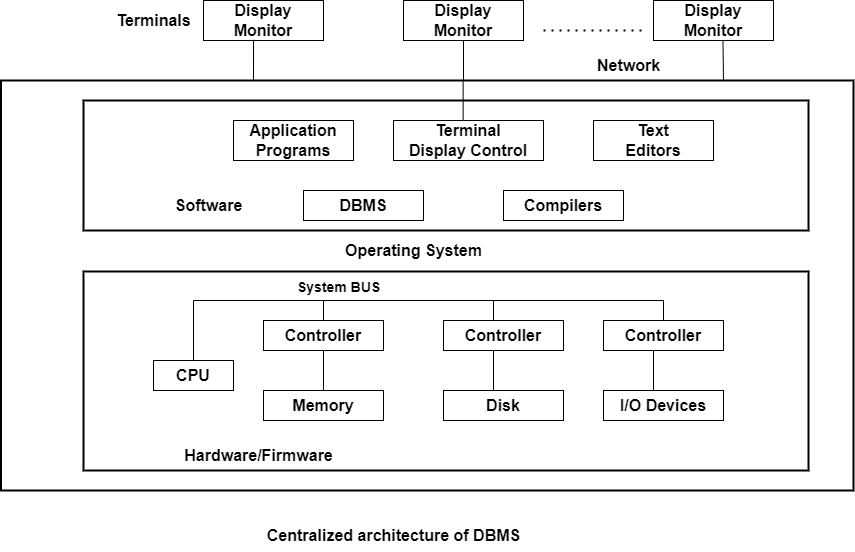
# **Centralized and Client Server Architecture for DBMS**

## Centralized Architecture of DBMS:

Architectures for DBMSs have generally followed trends seen in architectures for larger computer systems. The primary processing for all system functions, including user application programs, user interface programs, and all DBMS capabilities, was handled by mainframe computers in earlier systems.

The primary cause of this was that the majority of users accessed such systems using computer terminals with limited processing power and merely display capabilities. Only display data and controls were delivered from the computer system to the display terminals, which were connected to the central node by a variety of communications networks, while all processing was done remotely on the computer system.

The majority of users switched from terminals to PCs and workstations as hardware prices decreased. Initially, Database Systems operated on these computers in a manner akin to how they had operated display terminals. As a result, the DBMS itself continued to operate as a centralized DBMS, where all DBMS functionality, application program execution, and UI processing were done on a single computer. The physical elements of a centralized architecture Client/server DBMS designs emerged as DBMS systems gradually began to take advantage of the user side's computing capability.



## Client-server Architecture of DBMS:

We first talk about client/server architecture in general, and then we look at how DBMSs use it. In order to handle computing settings with a high number of PCs, workstations, file servers, printers, database servers, etc., the client/server architecture was designed.

A network connects various pieces of software and hardware, including email and web server software. To define specialized servers with a particular functionality is the aim. For instance, it is feasible to link a number of PCs or compact workstations to a file server that manages the client machines' files as clients. By having connections to numerous printers, different devices can be designated as a printer server; all print requests from clients are then directed to this machine. The category of specialized servers also includes web servers and email servers. Many client machines can utilize the resources offered by specialized servers. The user is given the proper user interfaces for these servers as well as local processing power to run local applications on the client devices. This idea can be applied to various types of software, where specialist applications, like a CAD (computer-aided design) package, are kept on particular server computers and made available to a variety of clients. Some devices (such as workstations or PCs with discs that only have client software installed) would only be client sites.

The idea of client/server architecture presupposes an underpinning structure made up of several PCs and workstations as well as fewer mainframe computers connected via LANs as well as other types of computer networks. In this system, a client is often a user machine that offers local processing and user interface capabilities. When a client needs access to extra features-like database access-that are not available on that system, it connects to a server that offers those features. A server is a computer system that includes both hardware and software that can offer client computer services like file access, printing, archiving, or database access. Generally speaking, some workstations install both client and server software, while others just install client software. Client and server software, however, typically run on separate workstations, which is more typical. On this underlying client/server framework, **Two-tier** and **Three-tier** fundamental DBMS architectures were developed.

### Two-Tier Client Server Architecture:

Here, the term "two-tier" refers to our architecture's two layers-the Client layer and the Data layer. There are a number of client computers in the client layer that can contact the database server. The API on the client computer will use **JDBC** or some other method to link the computer to the database server. This is due to the possibility of various physical locations for clients and database servers.

### Three-Tier Client-Server Architecture:

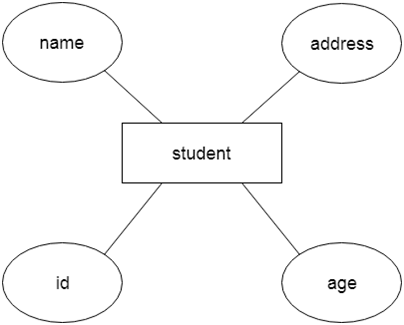
The Business Logic Layer is an additional layer that serves as a link between the Client layer and the Data layer in this instance. The layer where the application programs are processed is the business logic layer, unlike a Two-tier architecture, where queries are performed in the database server. Here, the application programs are processed in the application server itself.

# **Unit I-II**

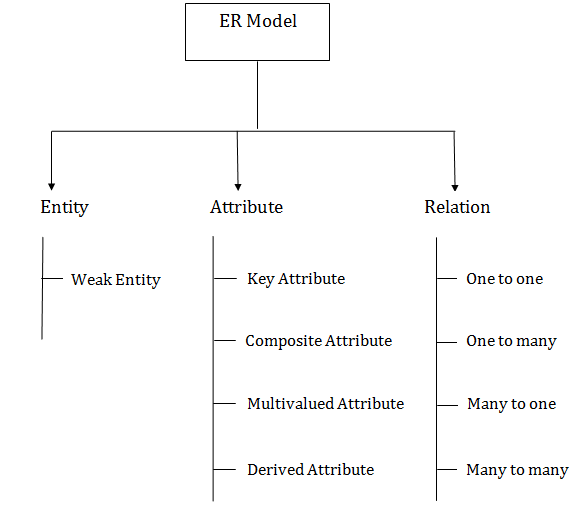
# Introduction of ER Model

* ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
* In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

**For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



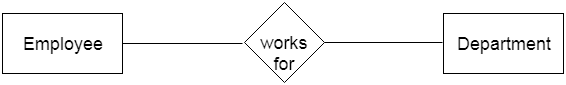
**Component of ER Diagram**



**1. Entity:**

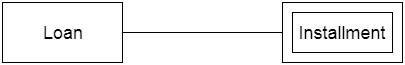
An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



**1.Weak Entity**

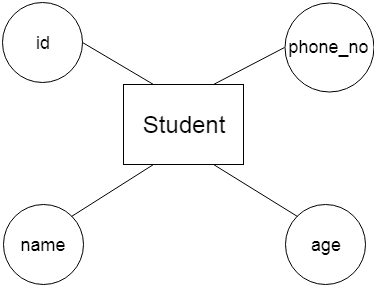
An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



**2. Attribute**

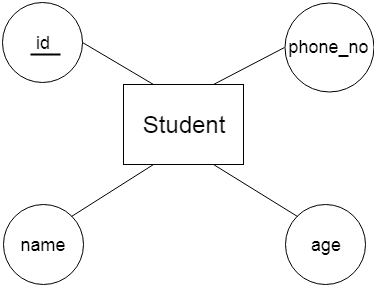
The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

**For example,** id, age, contact number, name, etc. can be attributes of a student.



**a. Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



**b. Composite Attribute**

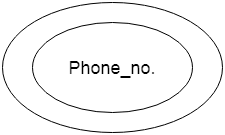
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



**c. Multivalued Attribute**

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

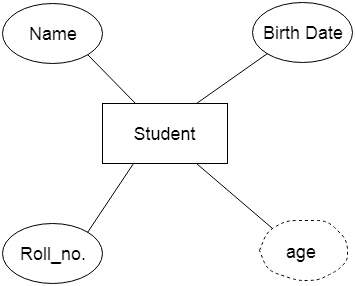
**For example,** a student can have more than one phone number.



**d. Derived Attribute**

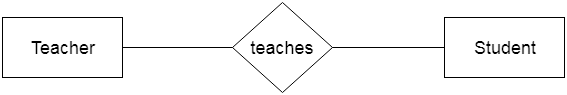
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

**For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



**3. Relationship**

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.

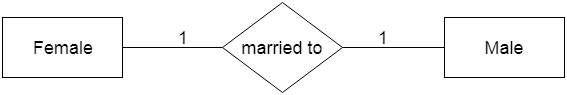


Types of relationship are as follows:

**a. One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

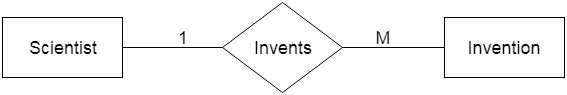
**For example,** A female can marry to one male, and a male can marry to one female.



**b. One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

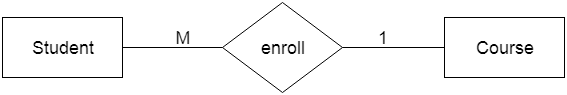
**For example,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



**c. Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

**For example,** Student enrolls for only one course, but a course can have many students.



**d. Many-to-many relationship**

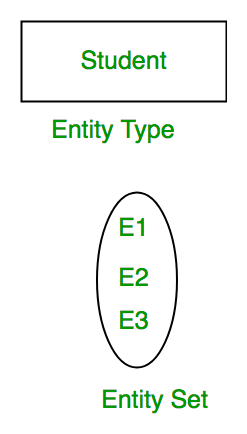
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

**For example,** Employee can assign by many projects and project can have many employees.



**Entity Set**

An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



*Entity Set*

We can represent the entity set in ER Diagram but can’t represent entity in ER Diagram because entity is row and column in the relation and ER Diagram is graphical representation of data.

**Types of Entity**

There are two types of entity:

**1. Strong Entity**

A [Strong Entity](https://www.geeksforgeeks.org/difference-between-strong-and-weak-entity/)is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

**2. Weak Entity**

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can’t be defined. These are called [Weak Entity types](https://www.geeksforgeeks.org/weak-entity-set-in-er-diagrams/).

**For Example,**A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can’t exist without the employee. So Dependent will be a **Weak Entity Type**and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type**.

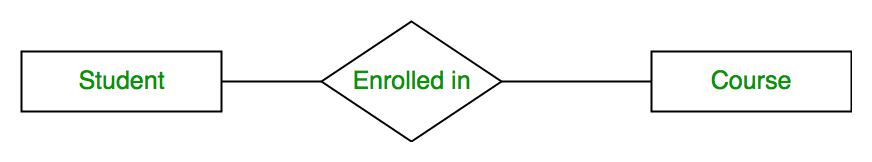
A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



*Strong Entity and Weak Entity*

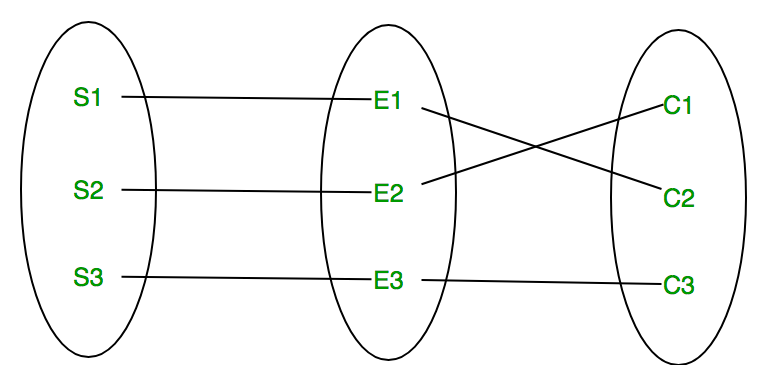
**Relationship Type and Relationship Set**

A Relationship Type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



*Entity-Relationship Set*

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

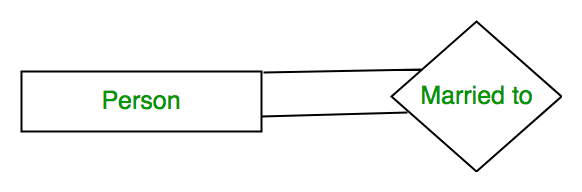


*Relationship Set*

**Degree of a Relationship Set**

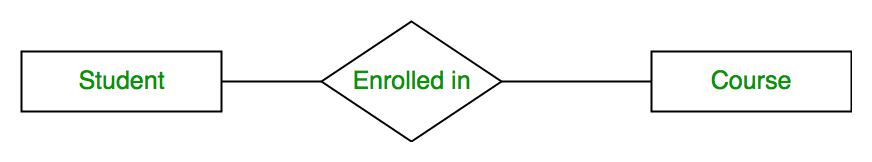
The number of different entity sets participating in a relationship set is called the [degree of a relationship set.](https://www.geeksforgeeks.org/degree-of-relations-in-dbms/)

**1. Unary Relationship:**When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



*Unary Relationship*

**2. Binary Relationship:**When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



*Binary Relationship*

**3. Ternary Relationship:**When there are three entity sets participating in a relationship, the relationship is called a ternary relationship.

**4. N-ary Relationship:**When there are n entities set participating in a relationship, the relationship is called an n-ary relationship.

# **Constraints on ER model in DBMS?**

**Constraints** are used for modeling limitations on the relations between entities.

There are two types of constraints on the Entity Relationship (ER) model −

* Mapping cardinality or cardinality ratio.
* Participation constraints.

## Mapping Cardinality

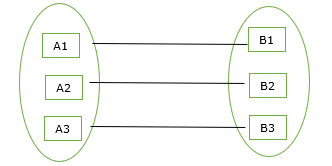
It is expressed as the number of entities to which another entity can be associated via a relationship set.

For the binary relationship set there are entity set A and B then the mapping cardinality can be one of the following −

* One-to-one
* One-to-many
* Many-to-one
* Many-to-many

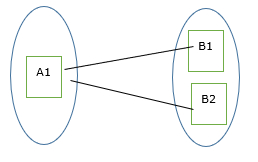
### One-to-one relationship

An entity set A is associated with at most one entity in B and an entity in B is associated with at most one entity in A.



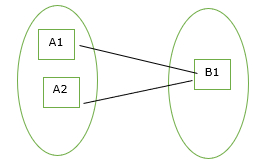
### One-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with at most one entity in A.



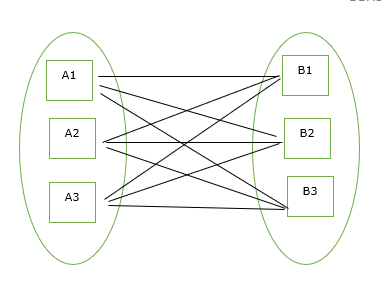
### Many-to-one relationship

An entity set A is associated with at most one entity in B and an entity set in B can be associated with any number of entities in A with a possibility of zero.



### Many-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with any number of entities in A with a possibility of zero.

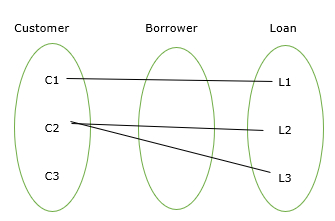


## Participation Constraints

Participate constraints are two types as mentioned below

* Total participation
* Partial Participation

The participation constraints are explained in the diagram below



Here, the customer to Loan is partial participation and the loan to the customer is total participation.

### Total participation

The participation of an entity set E in a relationship set R is said to be total if every entity in E Participates in at least one relationship in R.

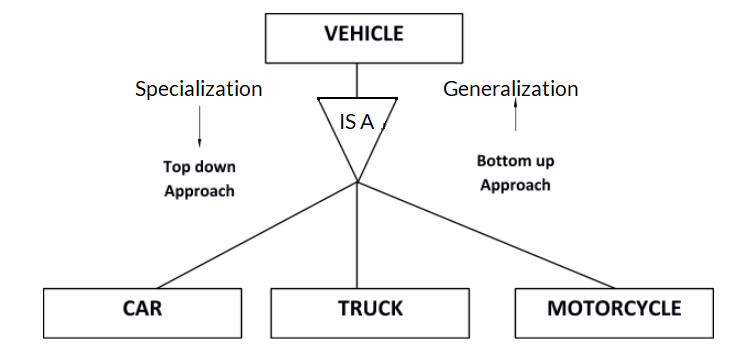
For Example − Participation of loan in the relationship borrower is total participation.

### Partial Participation

If only some of the entities in E participate in relationship R, then the participation of E in R is said to be partial participation.

## Subclasses

A subclass is a class derived from the superclass. It inherits the properties of the superclass and also contains attributes of its own. An example is:



Car, Truck and Motorcycle are all subclasses of the superclass Vehicle. They all inherit common attributes from vehicle such as speed, colour etc. while they have different attributes also i.e Number of wheels in Car is 4 while in Motorcycle is 2.

## Super classes

A superclass is the class from which many subclasses can be created. The subclasses inherit the characteristics of a superclass. The superclass is also known as the parent class or base class.

In the above example, Vehicle is the Superclass and its subclasses are Car, Truck and Motorcycle.

## Inheritance

Inheritance is basically the process of basing a class on another class i.e to build a class on a existing  class. The new class contains all the features and functionalities of the old class in addition to its own.

The class which is newly created is known as the subclass or child class and the original class is the parent class or the superclass.

**Generalization**

Generalization is the process of extracting common properties from a set of entities and creating a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher-level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher-level entity called PERSON as shown in Figure 1. In this case, common attributes like P\_NAME, and P\_ADD become part of a higher [entity](https://www.geeksforgeeks.org/difference-between-entity-and-object/) (PERSON), and specialized [attributes](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/) like S\_FEE become part of a specialized entity (STUDENT).

Generalization is also called as ‘ Bottom-up approach”.

**Specialization**

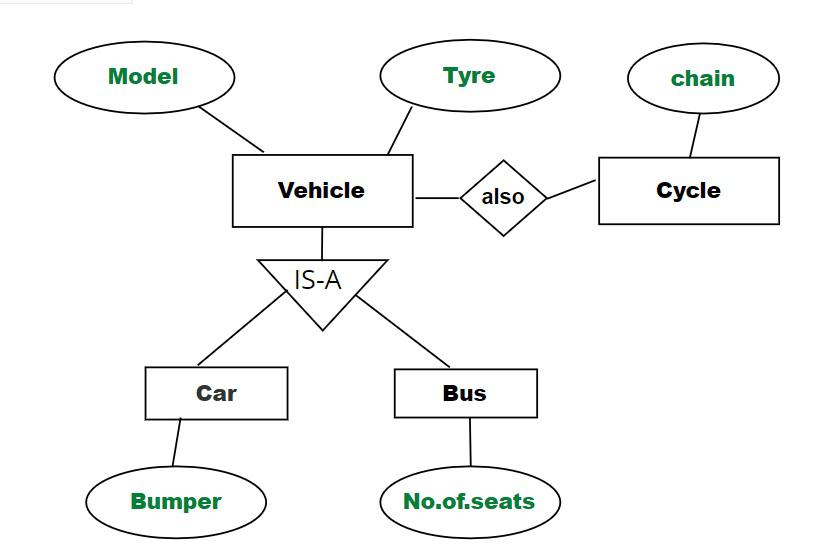
In specialization, an entity is divided into sub-entities based on its characteristics. It is a top-down approach where the higher-level entity is specialized into two or more lower-level [entities](https://www.geeksforgeeks.org/difference-between-entity-entity-set-and-entity-type/). For Example, an EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER, etc. as shown in Figure 2. In this case, common attributes like E\_NAME, E\_SAL, etc. become part of a higher entity (EMPLOYEE), and specialized attributes like TES\_TYPE become part of a specialized entity (TESTER).

Specialization is also called as ” Top-Down approch”.

**Inheritance:**

It is an important feature of generalization and specialization

* **Attribute inheritance**: allows lower level entities to inherit the attributes of higher level entities and vice versa.
* in diagram: **Car** entity is an inheritance of **Vehicle**entity ,So Car can acquire attributes of **Vehicle** example:car can acquire **Model** attribute of **Vehicle**.
* **Participation inheritance**: In participation inheritance, relationships involving higher level entity set also inherited by lower level entity and vice versa.
* in diagram: Vehicle entity has an relationship with Cycle entity ,So **Cycle entity** can acquire attributes of lower level entities i.e**Car** and **Bus** since it is inheritance of **Vehicle.**



*example of Relation*

