

**UNIT-I: Introduction:** Database system, Characteristics (Database Vs File System), Database Users, Advantages of Database systems, Database applications. Brief introduction of different Data Models; Concepts of Schema, Instance and data independence; Three tier schema architecture for data independence; Database system structure, environment, Centralized and Client Server architecture for the database.

**Entity Relationship Model:** Introduction, Representation of entities, attributes, entity set, relationship, relationship set, constraints, sub classes, super class, inheritance, specialization, generalization using ER Diagrams.

## DATA BASE SYSTEMS

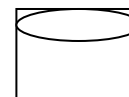
**DATA:** Data is a collection of raw facts. Data constitute the building blocks of Information. For example, roll no, sname, group, address, phone no, etc., are data items.

**INFORMATION:** Information is the result of processed data to reveal its meaning. Timely and useful information requires accurate data.

For example, in a student database file having the information of (1, Srinivas, CSE, Tadepalligudem). It is the information of particular student having a group of data items.

**DATABASE:** It is a collection of interrelated data that contains information about one particular organization or enterprise. It consists of collection database files.

SYMBOL of database:



**DBMS:** A database management system (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database. It is a software layer.

### **Basic File Terminology Definitions:**

**Field:** A character or group of characters that has a specific meaning. A field is used to define and store data. For example in a EMP file eno,ename,sal etc are fields.

**Record:** A logical collection of one or more fields that describes a person, place or thing. For Example the fields that constitute a record for employee named Rama might consists of Rama's name,Idno,sal etc.,

**File:** File is a collection of related records. For example a PRODUCT file contains all records of product file.

Ex: The records of STUDENT file are given below

Student No.	SName	Marks1	Marks2	Marks3
S01	Saritha	80	60	70
S02	Kavitha	70	80	60

S03	Krishna	60	70	90
-----	---------	----	----	----

Aspect	Database	File System
<b>Data Structure</b>	Structured or semi-structured	Unstructured
<b>Organization</b>	Tables (relational) or collections (NoSQL)	Hierarchical directories and files
<b>Query Capabilities</b>	Advanced querying (e.g., SQL, APIs)	Limited to file paths and content searching
<b>Relationships</b>	Supports data relationships (e.g., foreign keys)	No inherent relationship management
<b>Metadata</b>	Rich metadata (e.g., schema, constraints)	Basic metadata (e.g., file size, timestamps)
<b>Concurrency</b>	High, with transaction control and locks	Basic, relies on OS-level file locking
<b>Performance</b>	Optimized for relational and complex queries	Optimized for sequential file access
<b>Scalability</b>	Vertical and horizontal scaling	Vertical scaling only
<b>Backup &amp; Recovery</b>	Built-in tools for consistency	Relies on external tools or manual processes
<b>Access Control</b>	Fine-grained (row, column, or user-level)	Coarse-grained (file or folder-level)
<b>ACID Compliance</b>	Usually ACID-compliant	Not applicable
<b>Data Integrity</b>	Enforced through constraints and rules	No built-in data integrity features
<b>Use Case</b>	Structured data storage (e.g., CRM, ERP)	Storing unstructured files (e.g., images)
<b>Data Volume</b>	Efficient for large structured datasets	Suitable for large binary or flat files
<b>Ease of Management</b>	Requires setup and maintenance (e.g., database software)	Simple to set up and use
<b>Cost</b>	Higher cost (e.g., software, hardware)	Lower cost (basic file system is OS-integrated)

**Database Users :** Database users are individuals or applications that interact with a database to perform various operations, such as creating, reading, updating, and deleting data. Here are some types of database users:

1. **End Users :** End users are individuals who interact with the database through a user interface, such as a web application or a mobile app. They typically perform CRUD (Create, Read, Update, Delete) operations.
2. **Application Developers :** Application developers are responsible for designing and developing applications that interact with the database. They write code to perform database operations and integrate the database with the application.

3. **Database Administrators (DBAs)** : DBAs are responsible for managing and maintaining the database. They perform tasks such as database design, implementation, security, backup, and recovery.
4. **Data Analysts** : Data analysts are responsible for analyzing data in the database to gain insights and make informed decisions. They use various tools and techniques, such as SQL, data visualization, and data mining.
5. **Data Scientists** : Data scientists are responsible for extracting insights and knowledge from large datasets in the database. They use various techniques, such as machine learning, predictive analytics, and data visualization.
6. **System Administrators** : System administrators are responsible for managing the underlying infrastructure that supports the database, such as servers, storage, and networking.
7. **Security Administrators** : Security administrators are responsible for ensuring the security and integrity of the database. They perform tasks such as access control, authentication, and encryption.
8. **Business Intelligence Developers** : Business intelligence developers are responsible for designing and developing reports, dashboards, and other business intelligence solutions that interact with the database.
9. **Data Engineers** : Data engineers are responsible for designing, building, and maintaining large-scale data systems, including databases, data warehouses, and data lakes.
10. **Researchers** : Researchers use databases to store and analyze large datasets, often in academic or scientific contexts.

### Advantages of Database Management System

The advantages of database management systems are:

1. **Data Security:** As the number of users increases, the data transferring or data sharing rate also increases thus increasing the risk of data security. It is widely used in the corporate world where companies invest large amounts of money, time, and effort to ensure data is secure and used properly. A Database Management System (DBMS) provides a better platform for data privacy and security policies thus, helping companies to improve Data Security.
2. **Data integration:** Due to the Database Management System we have access to well-managed and synchronized forms of data thus it makes data handling very easy and gives an integrated view of how a particular organization is working and also helps to keep track of how one segment of the company affects another segment.
3. **Data abstraction:** The major purpose of a [database](#) system is to provide users with an abstract view of the data. Since many complex algorithms are used by the developers to increase the efficiency

of databases that are being hidden by the users through various data abstraction levels to allow users to easily interact with the system.

4. **Reduction in data Redundancy:** When working with a structured database, DBMS provides the feature to prevent the input of duplicate items in the database. for e.g. – If there are two same students in different rows, then one of the duplicate data will be deleted.
5. **Data sharing:** A DBMS provides a platform for sharing data across multiple applications and users, which can increase productivity and collaboration.
6. **Data consistency and accuracy:** DBMS ensures that data is consistent and accurate by enforcing data integrity constraints and preventing data duplication. This helps to eliminate data discrepancies and errors that can occur when data is stored and managed manually.
7. **Data organization:** A DBMS provides a systematic approach to organizing data in a structured way, which makes it easier to retrieve and manage data efficiently.
8. **Efficient data access and retrieval:** DBMS allows for efficient data access and retrieval by providing indexing and query optimization techniques that speed up data retrieval. This reduces the time required to process large volumes of data and increases the overall performance of the system.
9. **Concurrency and maintained Atomicity :** That means, if some operation is performed on one particular table of the database, then the change must be reflected for the entire database. The DBMS allows concurrent access to multiple users by using the synchronization technique.
10. **Scalability and flexibility:** DBMS is highly scalable and can easily accommodate changes in data volumes and user requirements. DBMS can easily handle large volumes of data, and can scale up or down depending on the needs of the organization.

### **DATABASE APPLICATIONS :**

Databases are widely used across various industries and domains due to their ability to efficiently store, manage, and retrieve data. Here are key applications of databases:

#### **1. Business and Enterprise Applications**

- **Customer Relationship Management (CRM):** Managing customer data, tracking interactions, and analyzing customer behavior.
- **Enterprise Resource Planning (ERP):** Storing and managing data related to supply chain, finance, and human resources.
- **E-commerce Systems:** Managing product catalogs, user accounts, inventory, and transaction details.

#### **2. Banking and Finance**

- **Transaction Management:** Tracking deposits, withdrawals, and other account activities.
- **Risk Management:** Analyzing financial risk using historical data.
- **Fraud Detection:** Monitoring unusual patterns in transactions to identify potential fraud.

#### **3. Education**

- **Student Information Systems:** Tracking student enrollment, grades, and attendance.
- **Learning Management Systems (LMS):** Storing course content, user progress, and assessments.

- **Research Databases:** Managing large datasets for academic research.

#### 4. Government

- **Public Administration:** Managing citizen records like voter registration, tax records, and licenses.
- **Law Enforcement:** Tracking criminal records, investigations, and case histories.
- **Data Analytics:** Analyzing data to shape public policies and optimize services.

#### 5. Social Media

- **User Profiles:** Managing personal information, posts, and connections.
- **Content Management:** Organizing posts, comments, and multimedia.
- **Analytics:** Tracking user engagement and trends.

#### 6. Travel and Tourism

- **Reservation Systems:** Managing hotel bookings, flights, and car rentals.
- **Customer Feedback:** Collecting and analyzing reviews and ratings.
- **Tour Planning:** Storing itineraries and package details.

**Data Model :** Data models describe how a database's logical structure is represented. In a database management system, data models are essential for introducing abstraction. Data models specify how data is linked to one another, as well as how it is handled and stored within the system.

#### Types of Data Models in DBMS

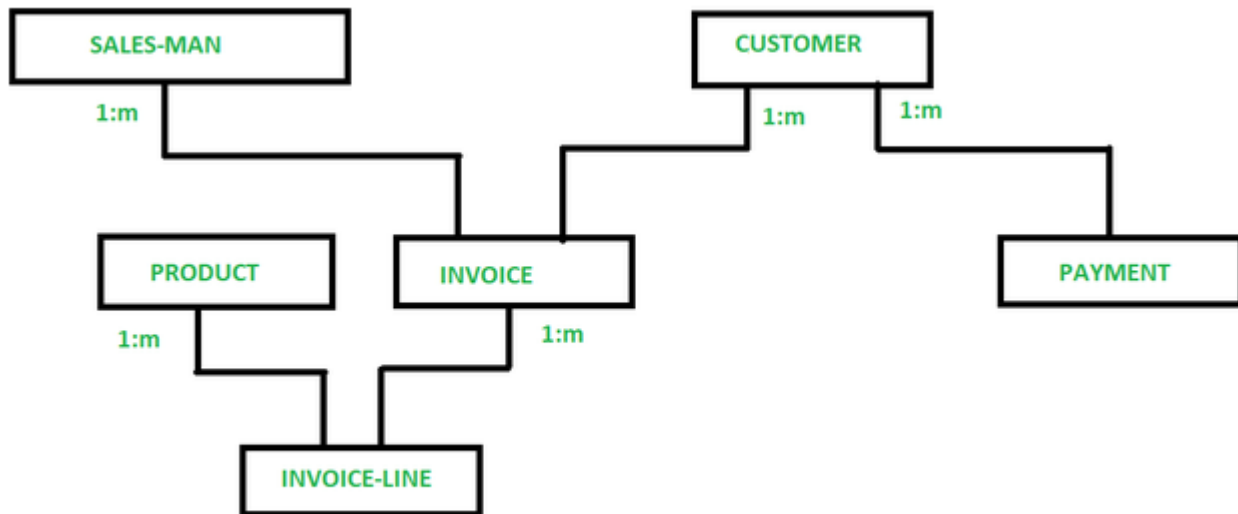
- Hierarchical Model. This concept uses a hierarchical tree structure to organize the data.
- Network Model.
- Entity-Relationship Model.
- Relational Model.
- Object-Oriented Data Model.

**Hierarchical Model :** The [hierarchical Model](#) is one of the oldest data model which was developed by IBM, in the 1950s. In a hierarchical model, data are viewed as a collection of 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.

**Network Data Model** This model was formalized by the Database Task group (DBTG) 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. We called **graphs** the logical associations between the segments. There will be more connections among different nodes. It can have M: N relations i.e, many-to-many which allows a record to have more than one parent segment.

Here, a relationship is called a set, and each set is made up of at least 2 types of record which are given below:

- An owner record that is the same as of parent in the hierarchical model.
- A member record that is the same as of child in the hierarchical model.



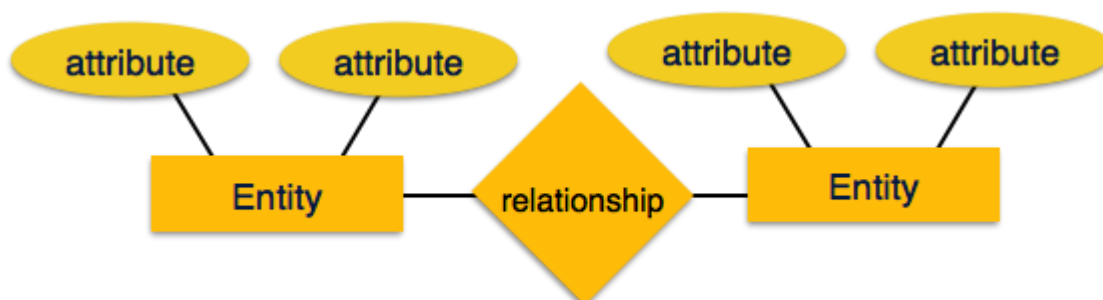
**Entity – Relationship Model :** Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

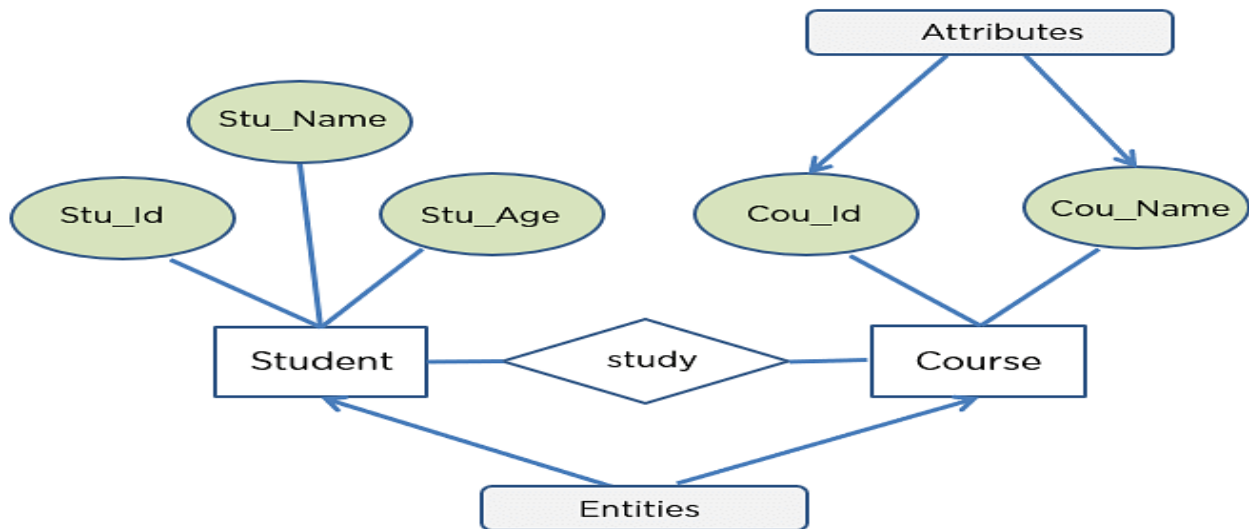
ER Model is based on –

- **Entities** and their *attributes*.
- **Relationships** among entities.

These concepts are explained below.



- **Entity** – An entity is a thing, place, or person in the real-world having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Relationship** – The logical association among entities is called **relationship**. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities. Mapping cardinalities are one to one, one to many, many to one and many to many.



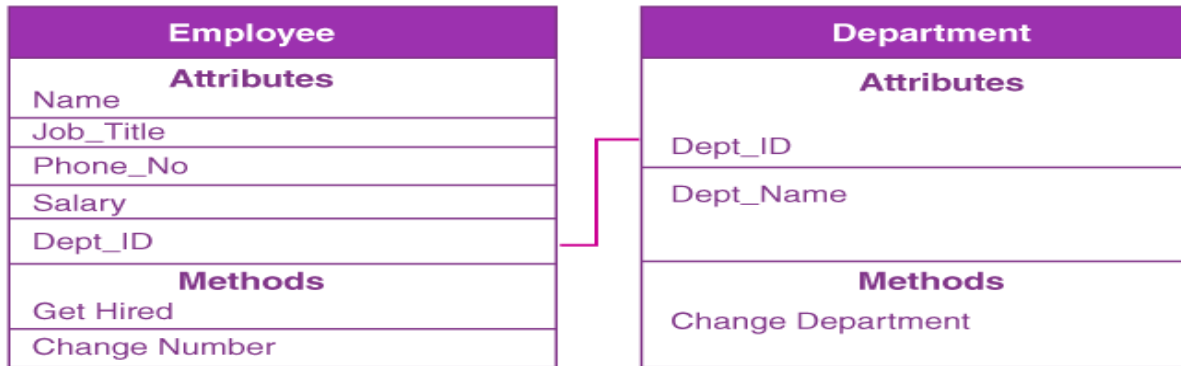
**Relational Data Model :** The relational model for database management is an approach to logically represent and manage the data stored in a database. In this model, the data is organized into a collection of two-dimensional inter-related tables, also known as relations. Each relation is a collection of columns and rows, where the column represents the attributes of an entity and the rows (or tuples) represents the records.

**Properties of Relations :**

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- Tuple has no duplicate value
- Order of tuples can have different sequence

**Object Oriented Data Model :** The OODM is a better representation of real-world challenges. Both the data and the relationship are contained into a single structure that's known as an object in this model. We can now store audios, pictures, videos, and other types of data in databases, which was previously impossible with the relational approach





Object Oriented Model

### What is Schema?

- The Skeleton of the database is created by the attributes and this skeleton is named Schema.
- Schema mentions the logical constraints like table, primary key, etc.
- The schema does not represent the data type of the attributes.

### 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](#), [foreign key](#).
- Data is organized in structured way with the help of database schema.

**Types of Database Schemas :** There are 3 types of database 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.

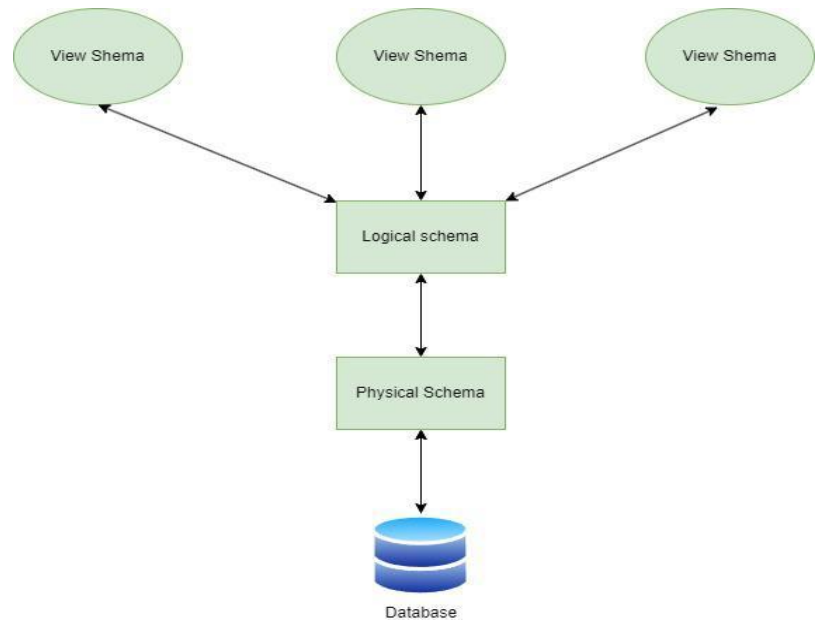
#### Logical Database Schema

- 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.



**Database Instance :** A **database instance** is a type of snapshot of an actual database as it existed at an instance of time. Hence it varies or can be changed as per the time. In contrast, the database schema is static and very complex to change the structure of a database.

### Data Independence :

Data independence refers to the ability to modify the schema definition at one level of the database system without altering the schema definition at the next higher level. There are two types of data independence:

#### 1. Logical Data Independence

- Logical data independence refers to the ability to change the conceptual schema without having to change the external schema.
- Logical data independence is used to separate the external level from the conceptual view.
- If we do any changes in the conceptual view of the data, then the user view of the data would not be affected.

#### 2. Physical Data Independence

- Physical data independence can be defined as the capacity to change the internal schema without having to change the conceptual schema.
- If we do any changes in the storage size of the database system server, then the Conceptual structure of the database will not be affected.
- Physical data independence is used to separate conceptual levels from the internal levels.

### 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

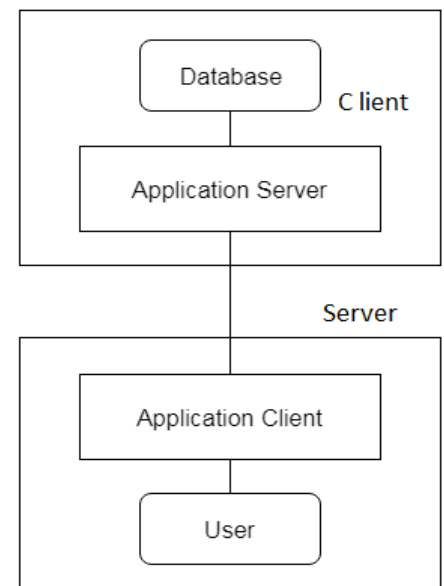
- In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of 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.

#### 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.



**Structure of DBMS :** [Database Management System \(DBMS\)](#) is software that allows access to data stored in a database and provides an easy and effective method of –

- Defining the information.
- Storing the information.

- Manipulating the information.
- Protecting the information from system crashes or data theft.
- Differentiating access permissions for different users.

Three Parts that make up the Database System are:

- Query Processor
- Storage Manager
- Disk Storage

**1. Query Processor :** The query processing is handled by the query processor, as the name implies. It executes the user's query, to put it simply. In this way, the query processor aids the database system in making data access simple and easy. The query processor's primary duty is to successfully execute the query. The Query Processor transforms (or interprets) the user's application program-provided requests into instructions that a computer can understand.

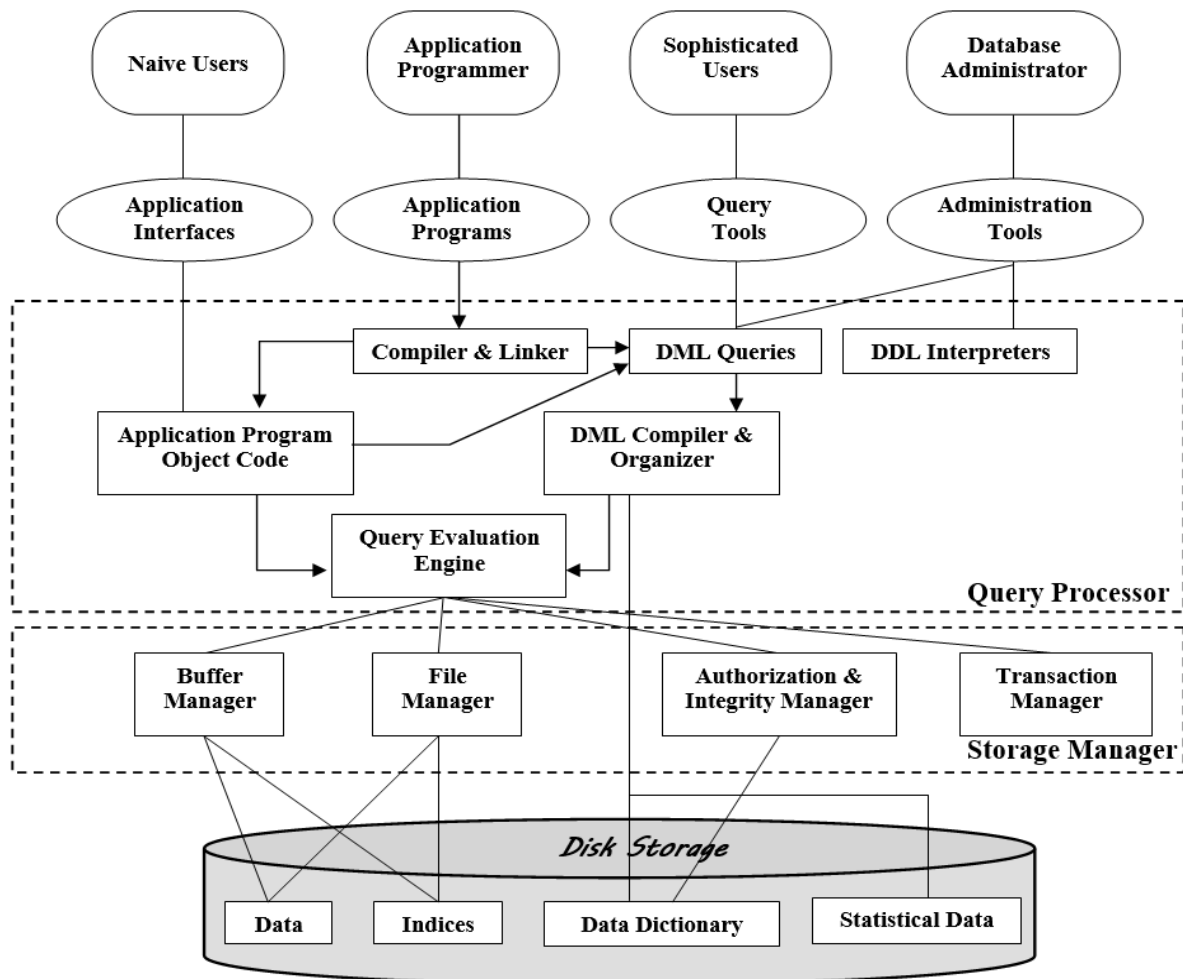


Figure: System Architecture

**Components of the Query Processor:**

- **DDL Interpreter:**

Data Definition Language is what DDL stands for. As implied by the name, the DDL Interpreter interprets DDL statements like those used in schema definitions (such as create, drop, etc.). This interpretation yields a set of tables that include the meta-data (data of data) that is kept in the data dictionary. Metadata may be stored in a data dictionary. In essence, it is a part of the disc storage.

- **DML Compiler:**

Compiler for DML Data Manipulation Language is what DML stands for. In keeping with its name, the DML Compiler converts DML statements like select, update, and delete into low-level instructions or simply machine-readable object code, to enable execution. The optimization of queries is another function of the DML compiler. Since a single question can typically be translated into a number of evaluation plans. As a result, some optimization is needed to select the evaluation plan with the lowest cost out of all the options. This process, known as query optimization, is exclusively carried out by the DML compiler. Simply put, query optimization determines the most effective technique to carry out a query.

- **Query Optimizer:**

It starts by taking the evaluation plan for the question, runs it, and then returns the result. Simply said, the query evaluation engine evaluates the SQL commands used to access the database's contents before returning the result of the query. In a nutshell, it is in charge of analyzing the queries and running the object code that the DML Compiler produces. Apache Drill, Presto, and other Query Evaluation Engines are a few examples.

## **2. Storage Manager:**

An application called Storage Manager acts as a conduit between the queries made and the data kept in the database. Another name for it is Database Control System. By applying the restrictions and running the DCL instructions, it keeps the database's consistency and integrity. It is in charge of retrieving, storing, updating, and removing data from the database.

### **Components of Storage Manager**

Following are the components of Storage Manager:

- **Integrity Manager:**

Whenever there is any change in the database, the Integrity manager will manage the integrity constraints.

- **Authorization Manager:**

Authorization manager verifies the user that he is valid and authenticated for the specific query or request.

- **File Manager:**

All the files and data structure of the database are managed by this component.

- **Transaction Manager:**

It is responsible for making the database consistent before and after the transactions. Concurrent processes are generally controlled by this component.

- **Buffer Manager:**

The transfer of data between primary and secondary memory and managing the cache memory is done by the buffer manager.

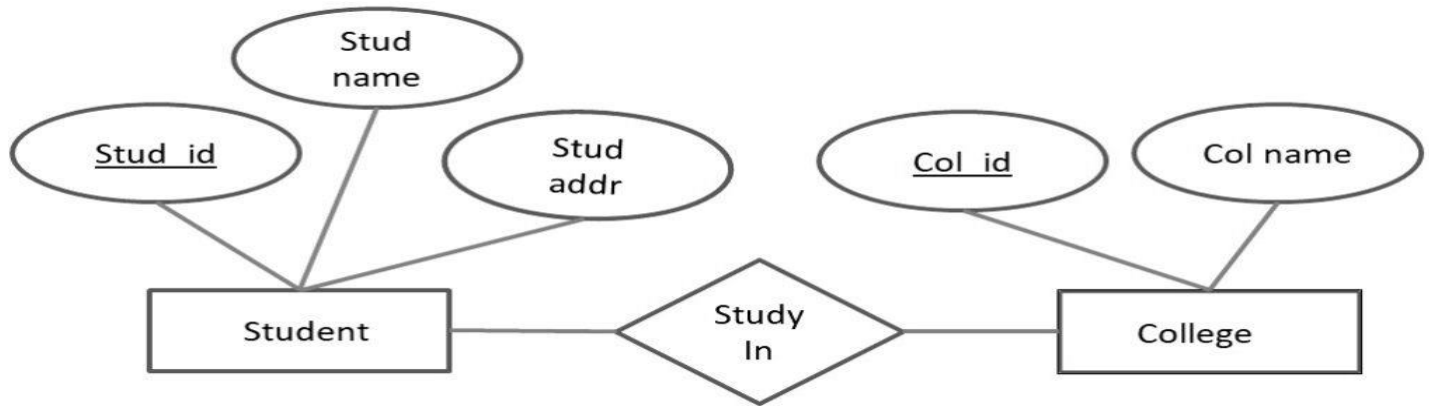
## **3. Disk Storage**

A DBMS can use various kinds of Data Structures as a part of physical system implementation in the form of disk storage.

**Components of Disk Storage : Following are the components of Disk Manager:**







- **Data Dictionary:** It contains the metadata (data of data), which means each object of the database has some information about its structure. So, it creates a repository which contains the details about the structure of the database object.
- **Data Files:** This component stores the data in the files.
- **Indices:** These indices are used to access and retrieve the data in a very fast and efficient way.

**Entity – Relationship model** is the high-level data model. It stands for the Entity-relationship model and is used to represent a logical view of the system from a data perspective. Creating an ER Model in DBMS is considered a best practice before implementing your database because it makes it easier for



the developers to understand the database system just by looking at the ER model. It develops a conceptual design for a database that provides a very simple and straightforward view of the data. ER model makes use of ER diagrams, which are the diagrams sketched to design a database. ER diagrams are built on three basic concepts: entities, attributes, and relationships between them. An [ER diagram in DBMS](#) defines entities, associated attributes, and relationships between entities. This helps visualize the logical structure of the database. The ER diagram was proposed by Peter Chen in 1971 as a visual tool to represent the ER model. He wanted to use the ER model as a conceptual modeling approach for DBMS. Let's understand this with the help of a simple example. Suppose you need to design a database for a school. In this database, a student is an entity with attributes such as an address, name, id, age, grades, etc.

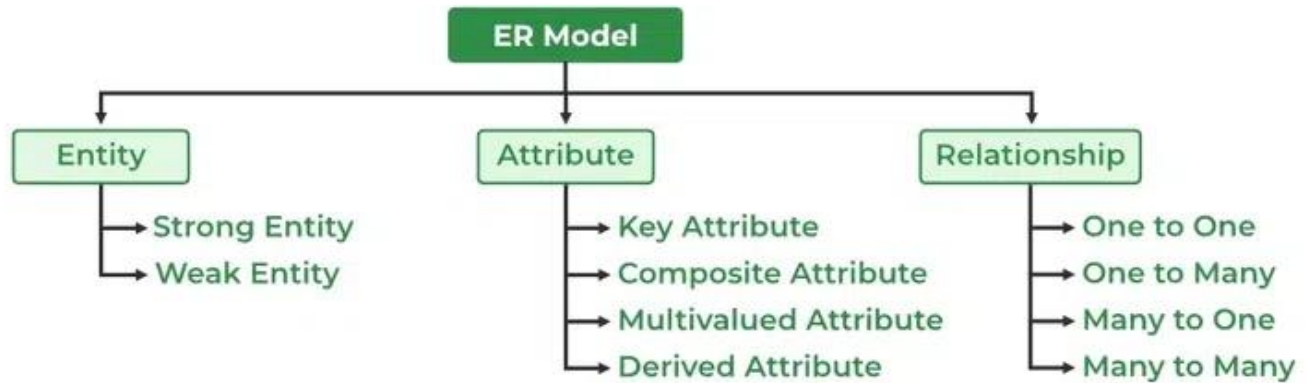
The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

Symbol Name	Symbol	Represents
Rectangles		Represents Entity
Ellipses		Represents Attribute
Diamonds		Represents Relationship
Lines		Links Attribute(s) to entity set(s) or Entity set(s) to Relationship set(s)
Double Ellipses		Represents Multivalued Attributes
Primary key		Represents Key Attributes / Single Valued Attributes

**Component of ER Diagram:** ER diagram consists of three basic concepts :

- Entities
- Attributes
- Relationships





## Entity :

An entity is an object in the real world that is distinguishable from other objects. Examples such as Employee, Department, Students etc.. Each entity consists of several characteristics or attributes that describe that entity. For example, if a Student is an entity, its attributes or characteristics are age, name, programme, brach, DOB, address and so on.

STUDENT

**Entity set:** is a group of entities of similar kinds. It can contain entities with attributes that share similar values. It's collectively a group of entities of a similar type. For example, a car entity set, a bank account entity set, and so on.

### Entities are of two types:

1. **Strong Entity** – A strong entity is an entity type that has a key attribute. It doesn't depend on other entities in the [schema](#). A strong entity always has a primary key, and it is represented by a single rectangle in the ER diagram.

Example – roll\_number identifies each student of the organization uniquely and hence, we can say that the student is a strong entity type.

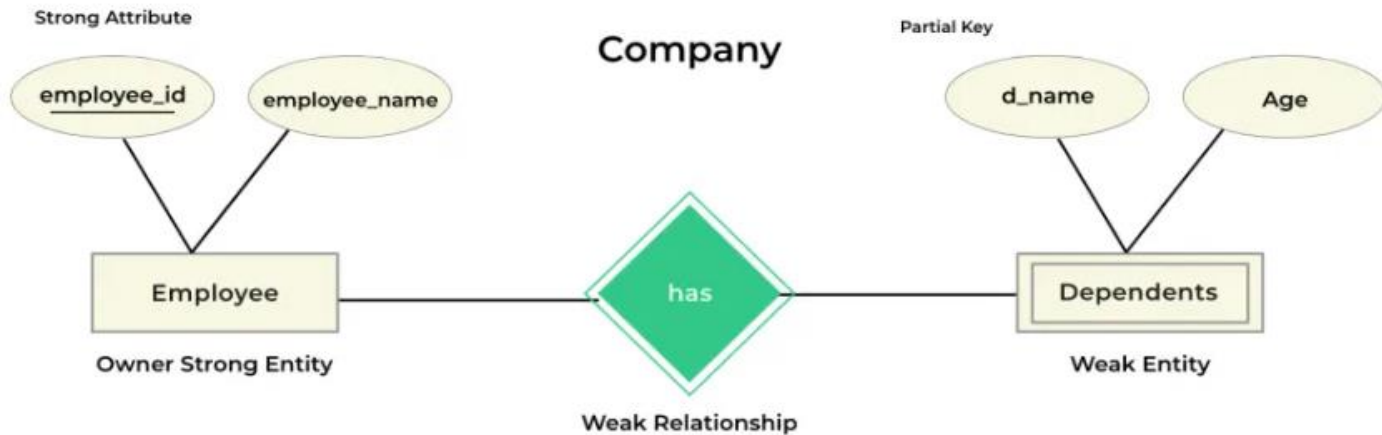
2. **Weak Entity** – [Weak entity](#) type doesn't have a key attribute and so we cannot uniquely identify them by their attributes alone. Therefore, a foreign key must be used in combination with its attributes to create a primary key. They are called Weak entity types because they can't be identified on their own. A weak entity is represented by a double-outlined rectangle in ER diagrams. An entity that depends on another entity called a weak entity.

Example for weak entity:

- In the ER diagram, we have **two entities Employee and Dependents**.
- **Employee is a strong entity** because it has a **primary key** attribute called Employee number (Employee\_No) which is capable of uniquely identifying all the employee.
- Unlike Employee, **Dependents is weak entity** because it **does not have any primary key**.



- D\_Name along with the Employee\_No can uniquely identify the records of Dependents. So here the D\_Name (Depends Name) is partial key.



The relationship between a weak entity type and a strong entity type is shown with a double-outlined diamond instead of a single-outlined diamond. This representation can be seen in the image given below.

Strong entity	Weak entity
Strong entity always has a primary key.	It will not have a primary key but it has partial key.
It is not dependent on any other entity.	Weak entity is dependent on the strong entity.
Represented by a single rectangle.	Represented by double rectangle.
Relationship between two strong entities is represented by a single diamond.	Relationship between a strong entity and the weak entity is represented by double Diamond.
A strong entity may or may not have total participation.	It has always total participation.

### Attribute

Attributes are the characteristics or properties which define the entity type. In ER diagram, the attribute is represented by an oval.

For example, here id, Name, Age, and Mobile\_No are the attributes that define the entity type Student.

### There are seven types of attributes:

**1. Simple attribute:** Attributes that cannot be further decomposed into sub-attributes are called simple attributes. It's an atomic value and is also known as the key attribute. The simple attributes are represented by an oval shape in ER diagrams with the attribute name underlined. For example, the roll number of a student, or the student's contact number are examples of simple attributes. As you can see from the above image, the attributes id is represented in the shape of an ellipse along with having the attribute name underlined.

2. **Composite attribute:** An attribute that can be decomposed into many other attributes is known as composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.

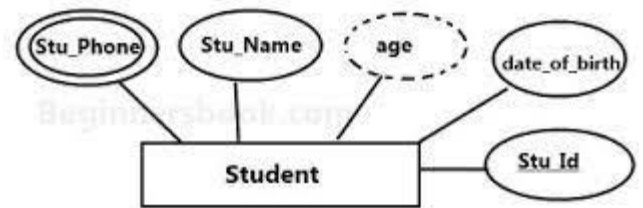
3. **Single-Valued Attributes :** Single valued attributes are those attributes that consist of a single value for each entity instance and can't store more than one value. The value of these single-valued attributes always remains the same, just like the name of a student.

4. **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.

5. **Derived Attributes:** are those attributes whose values can be derived from the values of other attributes. They are always dependent upon other attributes for their value.

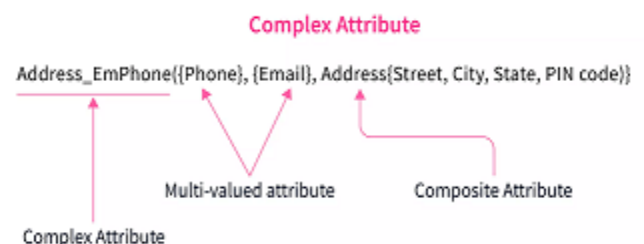
**For example,** As we were discussing above, DOB is a single-valued attribute and remains constant for an entity instance. From DOB, we can derive the Age attribute, which changes every year, and can easily calculate the age of a person from his/her date of birth value. Hence, the Age attribute here is derived attribute from the DOB single-valued attribute.



6. **Key Attributes :** Key attributes are special types of attributes that act as the primary key for an entity and they can uniquely identify an entity from an entity set. The values that key attributes store must be unique and non-repeating.

7. **Complex Attributes :** Complex attributes are rarely used in DBMS. They are formed by the combination of multi-valued and composite attributes. These attributes always have many sub-sections in their values.

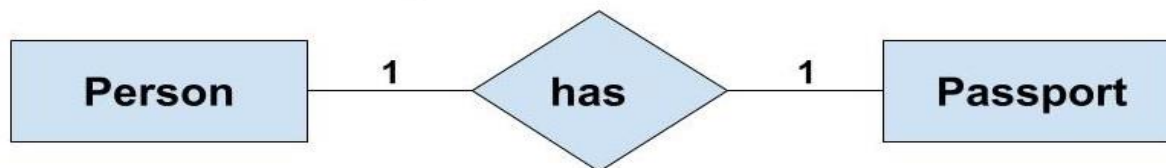
For example, Address\_EmPhone (which represents Address, Email, and Phone number altogether) is a complex attribute. Email and Phone number are multi-valued attributes while Address is a composite attribute which is further subdivided as House number, Street, City & State. This combination of multi-valued and composite attributes altogether forms a complex attribute.



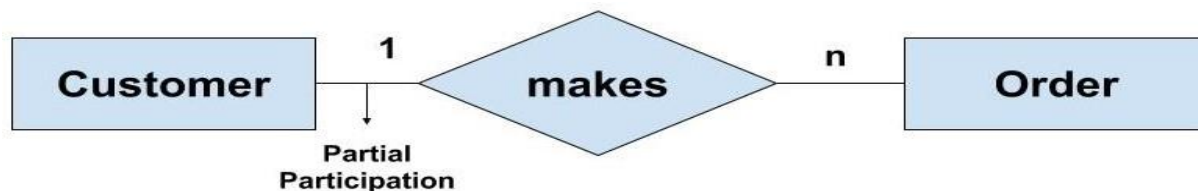
**Relationship:** The concept of [relationship in DBMS](#) is used to describe the association between different entities. This is denoted by the diamond. For **example**, the teacher entity type is related to the student entity type and their relation is represented by the diamond shape.

There are four types of relationships:

1. **One-to-One Relationships:** When only one instance of an entity is associated with the relationship to one instance of another entity, then it is known as one to one relationship. For example, let us assume that a person has one passport and a passport can be owned by one person. Therefore the relation is one-to-one.



2. **One-to-Many Relationships:** If only one instance of the entity on the left side of the relationship is linked to multiple instances of the entity on the right side, then this is considered a given one-to-many relationship. For example, a customer can place many orders. But any order is placed by only one customer.



3. **Many-to-One Relationships:** If only one instance of the entity on the left side of the relationship is linked to multiple instances of the entity on the right side, then this is considered a given one-to-many relationship. For example, a Student enrolls for only one course, but a course can have many students.
4. **Many to Many Relationships:** If multiple instances of the entity on the left are linked by relationships to multiple instances of the entity on the right, this is considered a many-to-one-relationship means relationship. For example, a customer can buy many products and a product is purchased by many customers.



## Relationship Sets :

A relationship set is a set of relationships of same type. A relationship set may be a unary relationship set or binary relationship set or ternary relationship set or n-ary relationship set.

### Unary Relationship :

Unary relationship set is a relationship set where only one entity set participates in a relationship set. Example-. One person is married to only one person. The degree of relationship is 1.



### Binary Relationship :

A binary relationship is relationship set where two entities participate in a relationship set. Example – A teacher teaches subjects. Here Teacher and Subject are two entities. The degree of relationship is 2.



### Ternary Relationship :

A Ternary relations is a relationship set where three entities participate in a relationship. Example – Doctor prescribes Medicines and Patient use these medicines prescribed by Doctor. The degree of relationship is 3.



### Recursive Relationship :

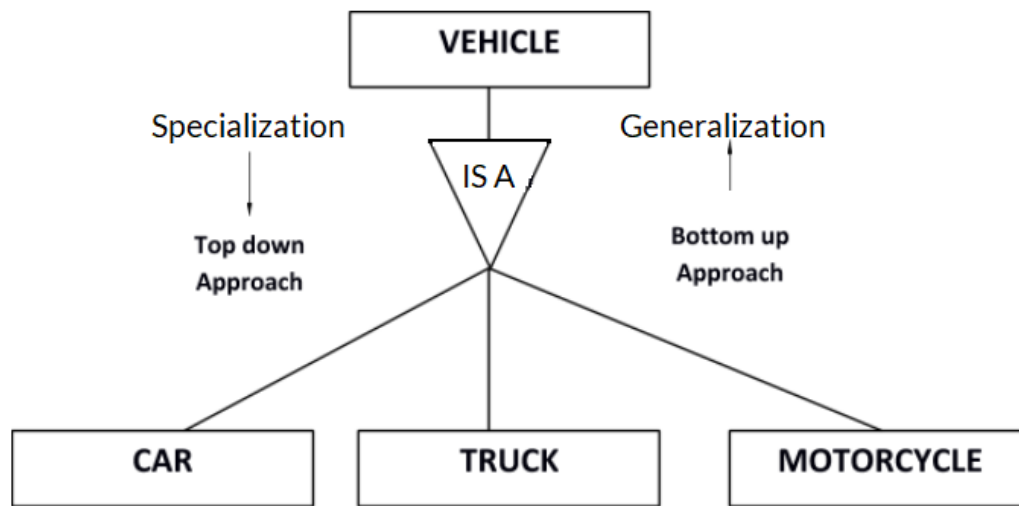
When there is a relationship between two entities of the same type, it is known as a recursive relationship. This means that the relationship is between different instances of the same entity type. An employee can supervise multiple employees. Hence, this is a recursive relationship of entity employee with itself.

### Constraints in ER model:

In the Entity-Relationship (ER) model, there are several types of constraints that help define the relationships and rules governing the data. The main types of constraints include:

1. **Key Constraints:** Ensures that each entity in an entity set is unique. A primary key uniquely identifies each record in a table.
2. **Domain Constraints:** Specifies that the values of an attribute must fall within a defined domain. For example, an age attribute must be a positive integer.
3. **Entity Integrity Constraints:** Ensures that the primary key of an entity cannot be null. This maintains the uniqueness of each entity.
4. **Referential Integrity Constraints:** Maintains the consistency of relationships between entities. For instance, a foreign key in one entity must match a primary key in another entity or be null.
5. **Cardinality Constraints:** Defines the number of instances of one entity that can or must be associated with instances of another entity. This includes one-to-one, one-to-many, and many-to-many relationships.
6. **Participation Constraints:** Indicates whether all or only some entity instances participate in a relationship. This can be total (mandatory participation) or partial (optional 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:



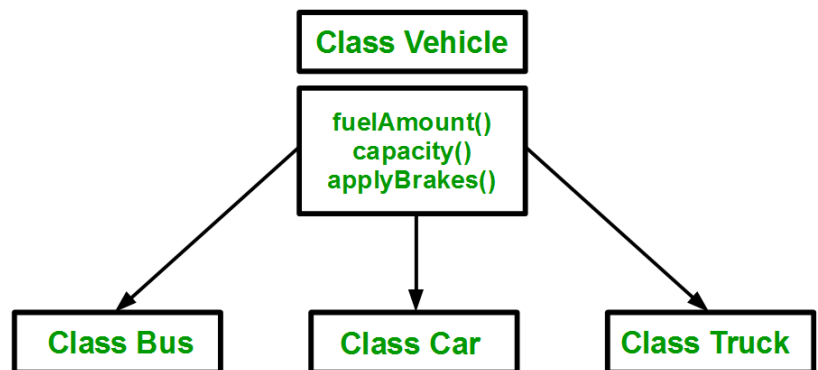
Car, Truck and Motorcycle are all subclasses of the superclass Vehicle. They all inherit common attributes from vehicle such as speed, colour etc. while they have different attributes also i.e Number of wheels in Car is 4 while in Motorcycle is 2.

**Superclasses :** 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 a feature or a process in which new classes are created from the existing classes. The **new class created is called “derived class” or “child class”** and **the existing class is known as the “base class” or “parent class”**. The derived class now is said to be inherited from the base class.

When we say **derived class inherits the base class**, it means that the derived class inherits all the properties of the base class, without changing the properties of base class and may add new features to its own.

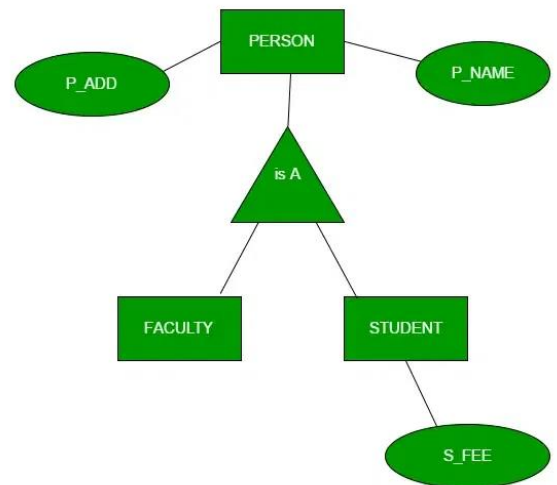


## Generalization

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 (PERSON), and specialized attributes like S\_FEE become part of a specialized entity (STUDENT).

Generalization is also called as ' Bottom-up approach".

Generalization

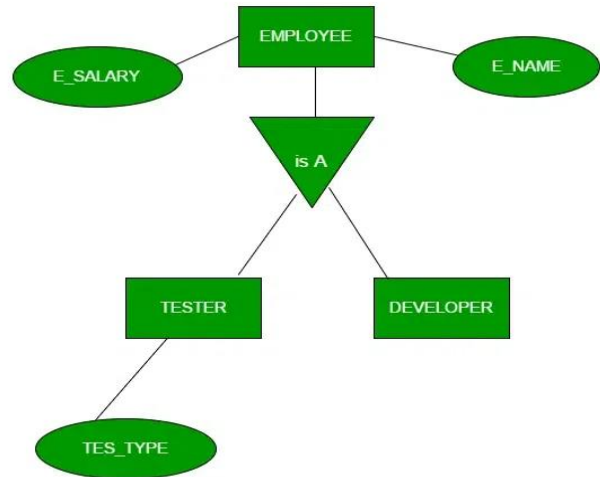


## Specialization

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. For Example, an EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER, etc. as shown in Figure. 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 approach".

Specialization



Specialization