**UNIT - I**

**Database System Applications: A Historical Perspective, File Systems versus a DBMS, the Data Model, Levels of Abstraction in a DBMS, Data Independence, Structure of a DBMS Introduction to Database Design: Database Design and ER Diagrams, Entities, Attributes, and Entity Sets, Relationships and Relationship Sets, Additional Features of the ER Model, Conceptual Design With the ER Model**

**A Historical Perspective**

Database Management Systems (DBMS) have been around for several decades, and their history can be traced back to the early 1960s. In the early days, computer systems were designed to manage data in a hierarchical or navigational manner, where data was stored in a tree-like structure. This method of storing data was inefficient and difficult to use, as it required a lot of manual effort to access and manage the data.

In the late 1960s, The first general-purpose DBMS, designed by **Charles Bachman**, was called the **Integrated Data Store (IDS)** which was based on network data model for which he was received the **Turing Award** (The most prestigious award which is equivalent to Nobel prize in the field of Computer Science.).

In the late 1970s, **Mr Edgar Codd** proposed a new data representation framework called the **Relational Database Model**. Mr Edgar Codd won the 1981 Turing Award for his seminal work. This model was based on the concept of a table, with rows representing individual records and columns representing individual fields within those records. The relational model allowed for more efficient storage and retrieval of data and was easier to use than the hierarchical or navigational models.

In the late 1980s IBM developed the **Structured Query Language (SQL)** for relational databases, as a part of R project. This system was designed to manage large amounts of data and was used primarily in corporate and government applications. SQL was adopted by the American National Standards Institute (ANSI) and International Organization for Standardization (ISO).

In the 1980s, several new DBMS products were introduced, including Oracle, Sybase, and Microsoft SQL Server. These systems were designed to be more user-friendly and to support more advanced data modeling and query languages.

In the 1990s, **object-oriented DBMS (OODBMS)** emerged, which were designed to store and manage complex data structures, such as multimedia and other types of non-traditional data. These systems were initially popular in research and academic environments, but their adoption was limited in the commercial sector.

In the 1991, **Microsoft ships MS access**, a personal DBMS and that displaces all other personal DBMS products.

In the 1997, **XML** applied to database processing. Many vendors begin to integrate XML into DBMS products.

In the 2000s, web-based applications and cloud computing became more popular, and DBMS systems began to adapt to these new technologies. New DBMS systems were developed to support distributed and web-based applications, including NoSQL databases such as MongoDB and Cassandra.

Today, DBMS systems continue to evolve, with an emphasis on scalability, performance, and support for cloud-based applications. Some of the most popular DBMS systems in use today include Oracle, Microsoft SQL Server, MySQL, PostgreSQL, and MongoDB.

* 1960 – Charles Bachman designed the first DBMS system
* 1970 – Codd introduced IBM’S [Information Management System (IMS)](https://www.ilearnlot.com/role-of-the-management-information-system-mis/54777/)
* 1976 – Peter Chen coined and defined the Entity-relationship model also known as the ER model
* 1980 – Relational Model becomes a widely accepted database component
* 1985 – Object-oriented DBMS develops.
* 1990 – Incorporation of object-orientation in relational DBMS.
* 1991 – Microsoft ships MS access, a personal DBMS and that displaces all other personal DBMS products.
* 1995 – First Internet database applications
* 1997 – XML applied to database processing. Many vendors begin to integrate XML into DBMS products.

# File Systems v/s DBMS

# What is a File system?

A file system is a technique of arranging the files in a storage devices like a hard disk, pen drive, DVD, etc. It helps you to organizes the data and allows easy retrieval of files when they are required. A file system enables you to handle the way of reading and writing data to the storage medium. It is directly installed into the computer with the Operating systems such as Windows and Linux.

# What is DBMS?

Database Management System (DBMS) is a software for storing and retrieving user’s data while considering appropriate security measures. It consists of a group of programs that manipulate the database. The DBMS accepts the request for data from an application and instructs the DBMS engine to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

# Difference between File System and Database systems

| **File System** | **Database systems** |
| --- | --- |
| A file system is a software that manages and organizes the files in a storage medium. It controls how data is stored and retrieved. | DBMS or Database Management System is a software application. It is used for accessing, creating, and managing databases. |
| The file system provides the details of data representation and storage of data. | DBMS gives an abstract view of data that hides the details. |
| Storing and retrieving of data can’t be done efficiently in a file system. | DBMS is efficient to use as there are a wide variety of methods to store and retrieve data. |
| Data redundancy is more. | Data redundancy is less. |
| Conventional file systems are used where there is less demand for security constraints(Security is very low). | Database systems are used when security constraints are high(Security is high). |
| File systems define the data in un-structured manner. Data is usually in isolated form. | Database systems define the data in a structured manner. Also there is well defined co-relation among the data. |
| Data inconsistency is more in file systems. | Data inconsistency is less in database systems. |
| User locates the physical address of file to access the data in conventional file systems. | User is unknown to the physical address of the data used in database systems. |
| There is no ability to concurrently access the data using conventional file system. | There is ability to access the data concurrently using database systems. |
| Not provide support for complicated transactions. | Easy to implement complicated transactions. |
| It doesn’t offer backup and recovery of data if it is lost. | DBMS system provides backup and recovery of data even if it is lost. |

Disadvantages of File system

* Each application has its data file so, the same data may have to be recorded and stored many times.
* Data dependence in the file processing system are data-dependent, but, the problem is incompatible with file format.
* Limited data sharing.
* The problem with security.
* Time-consuming.
* It allows you to maintain the record of the big firm having a large number of items.
* Required lots of labor work to do.
* Advantages of DBMS system
* DBMS offers a variety of techniques to store & retrieve data
* Uniform administration procedures for data
* Application programmers never exposed to details of data representation and Storage.
* A DBMS uses various powerful functions to store and retrieve data efficiently.
* Offers Data Integrity and Security.
* The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
* Reduced Application Development Time
* Consume lesser space.
* Reduction of redundancy.
* Data independence.

# Data Models

A data model is a collection of high-level data description constructs that hide many low-level storage details. A DBMS allows a user to define the data to be stored in terms of a data model.

The data models can be classified into different categories:

1. **Hierarchical Model**
2. **Network Model**
3. **Entity-Relationship Model**
4. **Relational Model**
5. **Object-Based Data Model**
6. **Semi-structured Data model**

# 1. Hierarchical Model

Hierarchical Model was the first DBMS model. This model organises the data in the hierarchical tree structure.

The hierarchy starts from the root which has root data and then it expands in the form of a tree adding child node to the parent node. This model easily represents some of the real-world relationships like food recipes, sitemap of a website etc.

Depicts a set of one-to-many (1:M) relationships

## Disadvantages of the hierarchical model:

* easy to design at the same time it is quite complex to implement.
* This model also lacks flexibility as the changes in the new tables or segments often yield very complex system management tasks. Here, a deletion of one segment can lead to the involuntary deletion of all segments under it.

# 2. Network Model

This model is an extension of the hierarchical model, the only difference is that a record can have more than one parent. It replaces the hierarchical tree with a graph.

The network model was created to represent complex data relationships more effectively when compared to hierarchical models, to improve database performance and standards.

Depicts both one-to-many (1:M) and many-to-many (M:N) relationships.

## Disadvantages of the Network Model:

* Database contains a complex array of pointers.
* System complexity limits efficiency.
* Structural changes require changes in all application programs.
* Navigation systems yield complex implementation and management.
* Keep heavy pressure on programmers due to the complex structure.
* Any change like updating, deletion, insertion is very complex.

# 3. Entity-Relationship Model

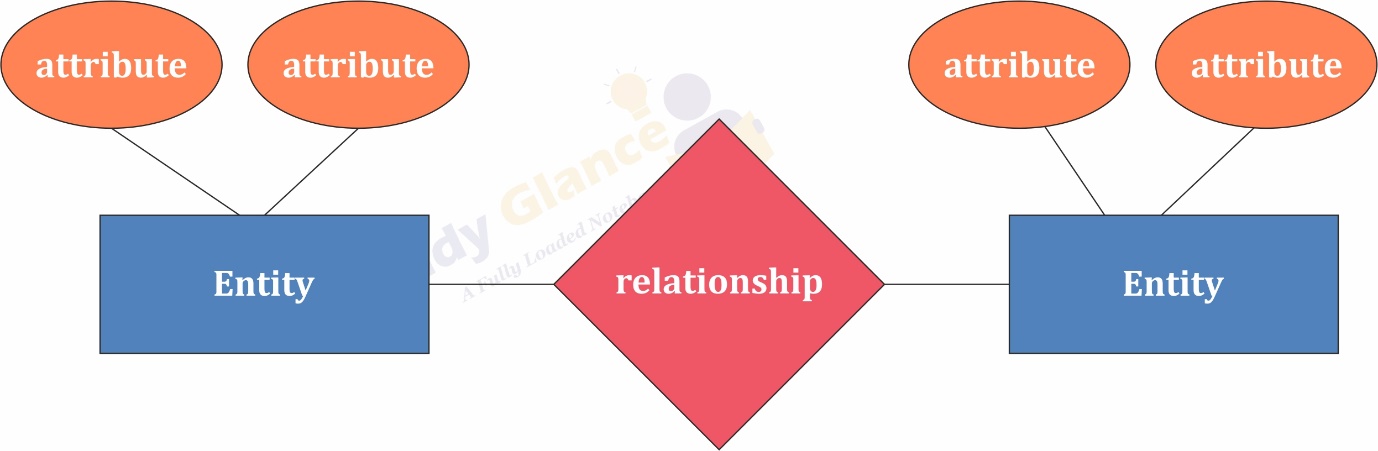
An ER model is the logical representation of data as objects and relationships among them. These objects are known as entities, and relationship is an association among these entities.

## Entity-Relationship Model Components

ER diagram basically having three components:

1. **Entities** − It is a real-world thing which can be a person, place, or even a concept. For Example: Department, Admin, Courses, Teachers, Students, Building, etc are some of the entities of a School Management System.
2. **Attributes** − An entity which contains a real-world property called an attribute. For Example: The entity employee has the property like employee id, salary, age, etc.
3. **Relationship** − Relationship tells how two attributes are related. For Example: Employee works for a department.

An entity has a real-world property called attribute and these attributes are defined by a set of values called domain.



## Advantages of Entity-Relationship Model

* The ER model is easy to build.
* This model is widely used by database designers for communicating their ideas.
* This model can easily convert to any other model like network model, hierarchical model etc.
* It is integrated with the dominant relational model.

## Disadvantages of Entity-Relationship Model

* There is no industry standard for developing an ER model.
* Information might be lost or hidden in the ER model.
* There is no Data Manipulation
* There is limited relationship representation.

# 4. Relational Model

The relational model uses a collection of tables to represent both data and the relationships. Tables are also known as relations. Each table has multiple columns represent as attributes, Attributes are the properties which define a relation. Each row of the table represents as Tuple, Tuple is one piece of information.

## Terminologies used in Realation Model

* **Tables:** relations are saved in the table format. A table has two properties rows and columns
* **Attribute:** columns represent as attributes
* **Tuple:** A Row represent as Tuple
* **Relation Schema:** A relation schema represents the name of the relation with its attributes.
* **Degree:** The total number of attributes which in the relation is called the degree of the relation.
* **Cardinality:** Total number of rows present in the Table.
* **Column:** The column represents the set of values for a specific attribute.
* **Relation instance:** The set of tuples of a relation at a particular instance of time is called as relation instance.

## Advantages of Realation Model

* **Structural Independence**: Structural independence is an ability that allows us to make changes in one database structure without affecting other. The relational model have structural independence. Hence making required changes in the database is convenient in relational database model.
* **Conceptual Simplicity**: The relational model allows the designer to simply focus on logical design and not on physical design. Hence relational models are conceptually simple to understand.
* **Query Capability**: Using simple query language (such as SQL) user can get information from the database or designer can manipulate the database structure.
* **Easy design,maintenance and usage**: The relational models can be designed logically hence they are easy to maintain and use.

## Disadvantages of Realation Model

* Relational model requires powerful hardware and large data storage devices.
* May lead to slower processing time.
* Poorly designed systems lead to poor implementation of database systems.

# 5. Object-Based Data Model

The complex real world problems are represented as objects with different attributes. In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred as object. All objects have multiple relationships between them. Basically, it is combination of Object Oriented programming and Relational Database Model.

## Components of Object Oriented Data Model

1. **Objects:** An object is an abstraction of a real world entity or we can say it is an instance of class. Objects encapsulates data and code into a single unit which provide data abstraction by hiding the implementation details from the user.
2. **Attribute:** An attribute describes the properties of object.
3. **Methods:** Method represents the behavior of an object, it represents the real-world action
4. **Class:** A class is a collection of similar objects with shared structure i.e. attributes and behavior.
5. **Inheritance:** new classes are created from the existing classes

## Advantages of Object-Based Data Model

* Reduced Maintenance
* Real-World Modeling
* Improved Reliability and Flexibility
* High Code Reusability

## Disadvantages of Object-Based Data Model

* It is a complex navigational system.
* Slow development of standards.
* High system overheads.
* Slow transactions.

# 6. Semi-structured Data model

The semi-structured data model permits the specification of data where individual data items of same type may have different sets of attributes. The Extensible Markup Language (XML) is widely used to represent semistructured data model.

## Advantages of Semi-structured Data Model

* Data is not constrained by fixed schema.
* It is flexible.
* It is portable.

## Disadvantage of Semi-structured Data Model

* Queries are less efficient than other types of data model.

# Levels of Abstraction in DBMS

Data Abstraction is a process of hiding unwanted or irrelevant details from the end user. It provides a different view and helps in achieving data independence which is used to enhance the security of data.

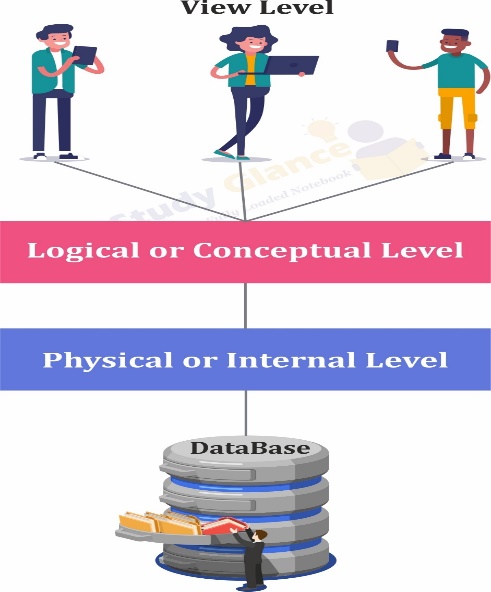
The database systems consist of complicated data structures and relations. For users to access the data easily, these complications are kept hidden, and only the relevant part of the database is made accessible to the users through data abstraction.

# Levels of abstraction for DBMS

Database systems include complex data-structures. In terms of retrieval of data, reduce complexity in terms of usability of users and in order to make the system efficient, developers use levels of abstraction that hide irrelevant details from the users. Levels of abstraction simplify database design.

Mainly there are three levels of abstraction for DBMS

1. **Physical or Internal Level**
2. **Logical or Conceptual Level**
3. **View or External Level**



# 1. Physical or Internal Level

The internal level has an internal schema which describes the physical storage structure of the database.

The internal schema is also known as a physical schema.

It uses the physical data model. It is used to define that how the data will be stored in a block.

The physical level is used to describe complex low-level data structures in detail.

## Facts about Internal schema

* The internal schema is the lowest level of data abstraction
* It helps you to keeps information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records
* The internal view tells us what data is stored in the database and how

# 2. Logical or Conceptual Level

The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.

The conceptual schema describes the structure of the whole database.

The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.

In the conceptual level, internal details such as an implementation of the data structure are hidden.

Programmers and database administrators work at this level.

## Facts about Conceptual schema

* Defines all database entities, their attributes, and their relationships
* Security and integrity information
* In the conceptual level, the data available to a user must be contained in or derivable from the physical level.

# 3. View or External Level

It hides the unrelated details of the database from the user. There may be “n” number of external views for each database.

Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.

View level can be used by all users (all levels' users). This level is the least complex and easy to understand.

## Facts about External schema

* An external level is only related to the data which is viewed by specific end users.
* This level includes some external schemas.
* External schema level is nearest to the user
* An external schema is also known as view schema.
* Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
* The view schema describes the end user interaction with database systems.

# Data Independence

* Data independence can be explained using the three-schema architecture.
* Data independence refers characteristic of being able to modify the schema at one level of the database system without altering the schema at the next higher level.

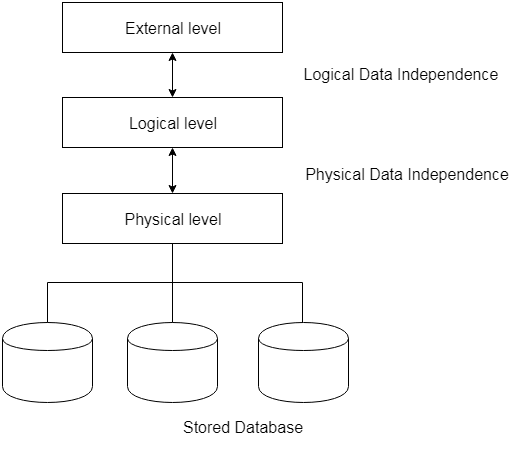
There are two types of data independence:

## **1. Logical Data Independence**

* Logical data independence refers characteristic of being able 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.
* Logical data independence occurs at the user interface level.

## **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.
* Physical data independence occurs at the logical interface level.



**Fig: Data Independence**

# structure of DBMS

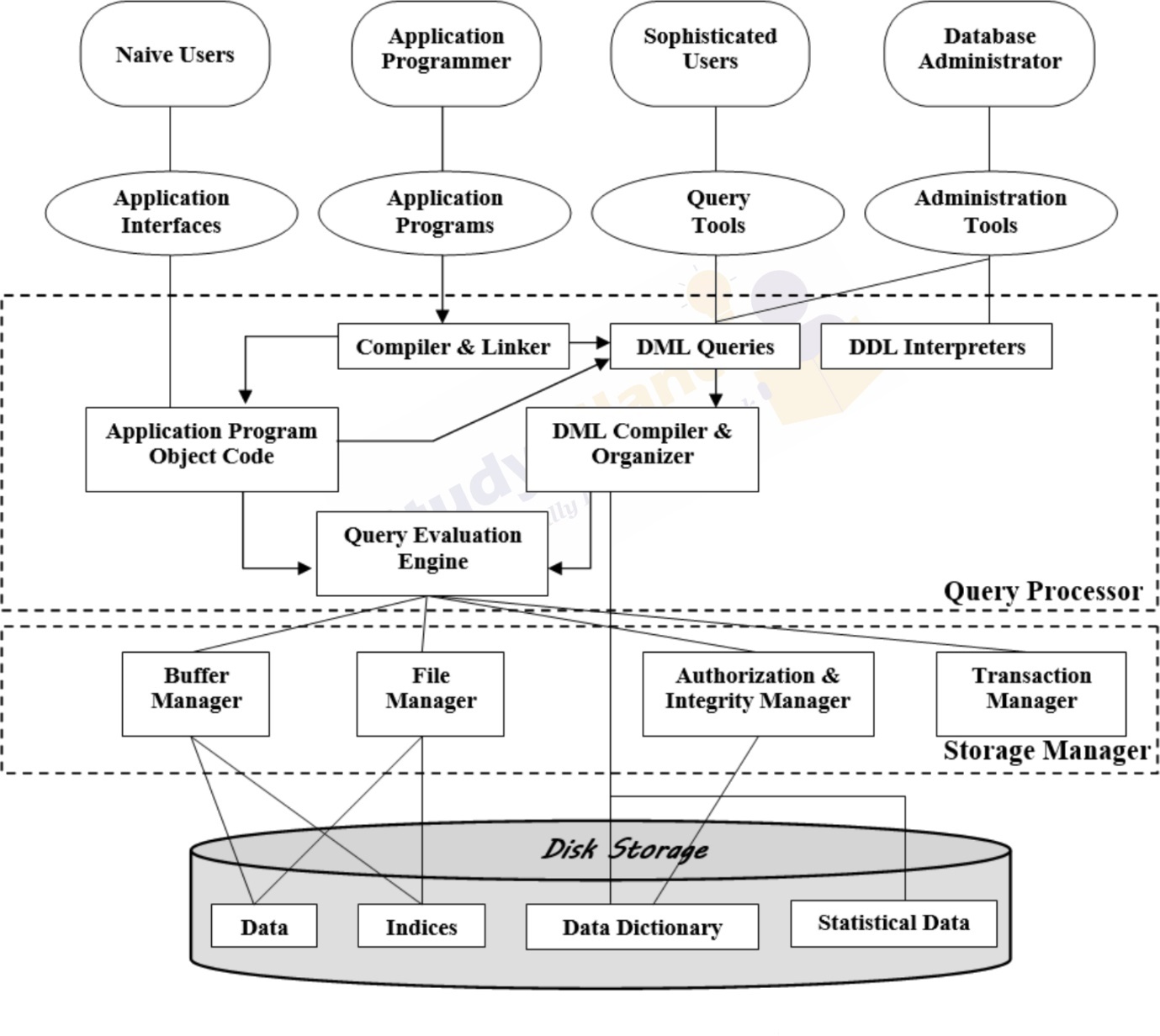
Database System Architecture

The typical structure of DBMS is based on Relational data model.

The top part of the architecture shows application interfaces used by naive users, application programs created by application programmers, query tools used by sophisticated users and administration tools used by database administrator.

The lowest part of the architecture is for disk storage.

The Middle two parts(**Query processor and storage manager**) are important components of database architecture.



## Query processor:

The interactive query processor helps the database system to simplify and facilitate access to data. It consists of DDL(**Data Definition Language**) interpreter, DML(**Data Manipulation Language**) compiler and query evaluation engine.

The following are various functionalities and components of query processor

* **DDL interpreter**: This is basically a translator which interprets the DDL statements in data dictionaries.
* **DML compiler**: It translates DML statements query language into an evaluation plan. This plan consists of the instructions which query evaluation engine understands.
* **Query evaluation engine**: It executes the low-level instructions generated by the DML compiler.

When a user issues a query, the parsed query is presented to a query optimizer, which uses information about how the data is stored to produce an efficient execution plan for evaluating the query. An execution plan is a blueprint for evaluating a query. It is evaluated by query evaluation engine.

## Storage manager:

Storage manager is the component of database system that provides interface between the low level data stored in the database and the application programs and queries submitted to the system.

The storage manager is responsible for storing, retrieving, and updating data in the database. The storage manager components include

* **Authorization and integrity manager**: Validates the users who want to access the data and tests for integrity constraints.
* **Transaction manager**: Ensures that the database remains in consistent despite of system failures and concurrent transaction execution proceeds without conflicting.
* **File manager**: Manages allocation of space on disk storage and representation of the information on disk.
* **Buffer manager**: Manages the fetching of data from disk storage into main memory. The buffer manager also decides what data to cache in main memory. Buffer manager is a crucial part of database system.

Storage manager implements several data structures such as

* **Data files**: Used for storing database itself.
* **Data dictionary**: Used for storing metadata, particularly schema of database.
* **Indices**: Indices are used to provide fast access to data items present in the database

**Database Design in DBMS**

Database design is the organization of data according to a database model. Properly designed databases are easu to maintain, improves data consistency.

The database design process can be divided into six steps. The **ER model**(Entity Relationship model) is most relevant to the first three steps.

1. **Requirement analysis**
2. **Conceptual database design**
3. **Logical database design**
4. **Schema refinement**
5. **Physical database design**
6. **Application and security design**

**1. Requirement analysis**

* It is necessary to understand **what data need to be stored in the database**, what applications must be built, what are all those operations that are frequently used by the system.
* The requirement analysis is an informal process and it requires proper communication with user groups.
* There are several methods for organizing and presenting information gathered in this step. Some automated tools can also be used for this purpose.

**2. Conceptual database design**

* The information gathered, is used to develop a high-level description of the data to be stored in the database
* This is a steps in which E-R Model i.e. Entity Relationship model is built.
* The goal of this design is to create a simple description of data that matches with the requirements of users.

**3. Logical database design**

* This is a step in which ER model in converted to relational database schema, sometimes called as the logical schema in the relational data model.

**4. Schema refinement**

* In this step, relational database schema is analyzed to identify the potential problems and to refine it.
* The schema refinement can be done with the help of normalizing and restructuring the relations.

**5. Physical database design**

* The design of database is refined further.
* This step may simply involve building indexes on tables and clustering tables, redesign of parts of the database schema obtained from the earlier design steps.

**6. Application and security design**

* Using design methodologies like UML(Unified Modeling Language) try to address the complete software design of the database can be accomplished.
* The role of each entity in every process must be reflected in the application task.
* For each role, there must be the provision for accessing and prohibiting some part of database.
* Thus some access rules must be enforced on the application(which is accessing the database) to protect the security features.

# ER Diagrams in DBMS

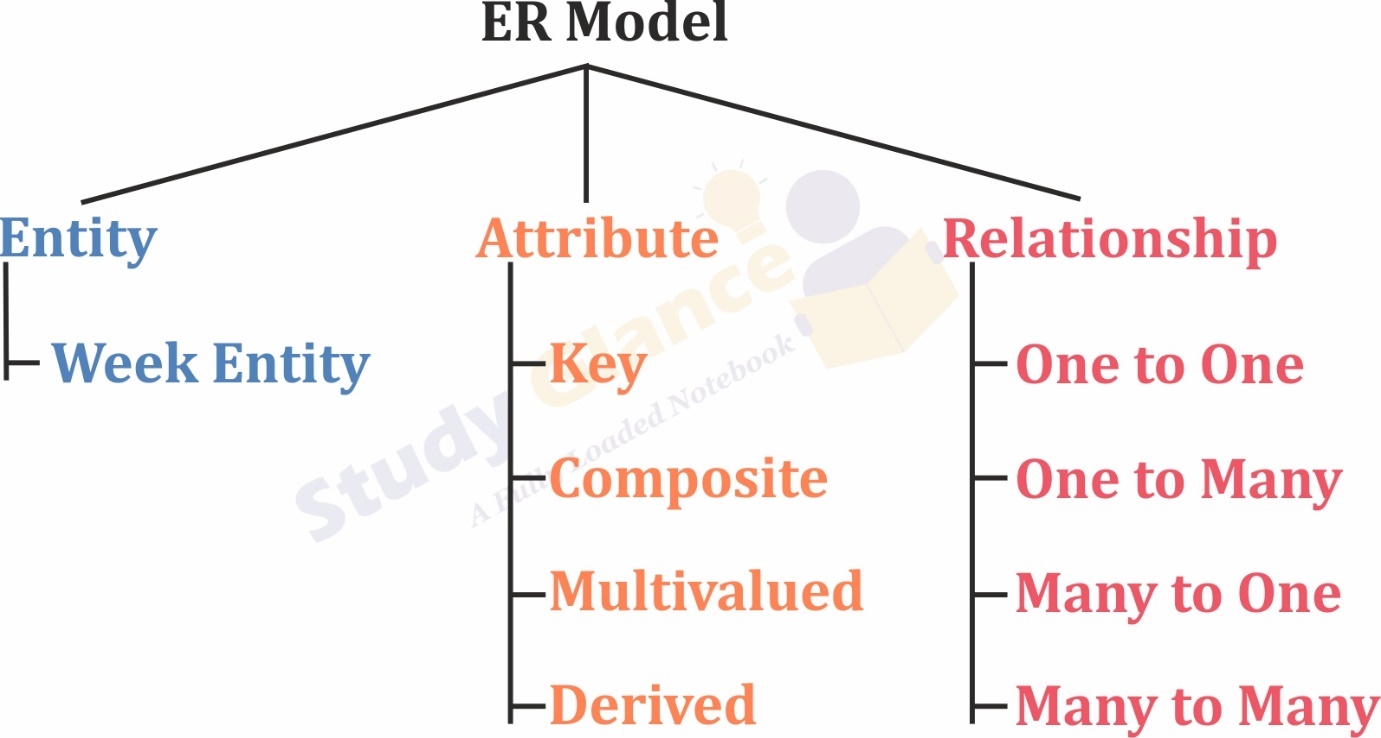
ER model in DBMS is the high-level data model. It stands for the Entity-relationship model and is used to represent a logical view of the system from a data perspective. In simple words, the entity relationship diagram is a blueprint that can used to create a database. E-R diagrams are used to model real-world objects like a person, a car, a company and the relation between these real-world objects.

## Features of ER model

* E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).
* E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.
* E-R diagrams require no technical knowledge and no hardware support.
* These diagrams are very easy to understand and easy to create even by a naive user.
* It gives a standard solution of visualizing the data logically.

ER diagram basically having three components:

1. **Entities** − It is a real-world thing which can be a person, place, or even a concept. For Example: Department, Admin, Courses, Teachers, Students, Building, etc are some of the entities of a School Management System.
2. **Attributes** − An entity which contains a real-world property called an attribute. For Example: The entity employee has the property like employee id, salary, age, etc.
3. **Relationship** − Relationship tells how two attributes are related. For Example: Employee works for a department.



| **Component** | **Symbol** |
| --- | --- |
| Entity | entity |
| Weak Entity | weak entity |
| Attribute | Attribute |
| Key Attribute | Key Attribute |
| Composite Attribute | Composite Attribute |
| Multivalued Attribute | Multivalued Attribute |
| Derived Attribute | Derived Attribute |
| Relationship |  |
| Weak Relationship |  |
| Participation Constraints |  |

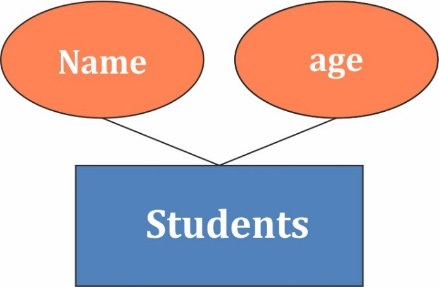
# Attributes in DBMS

Attributes define the properties of a data object of entity. For example if student is an entity, his ID, name, address, date of birth, class are its attributes.

# Types of Attributes in DBMS

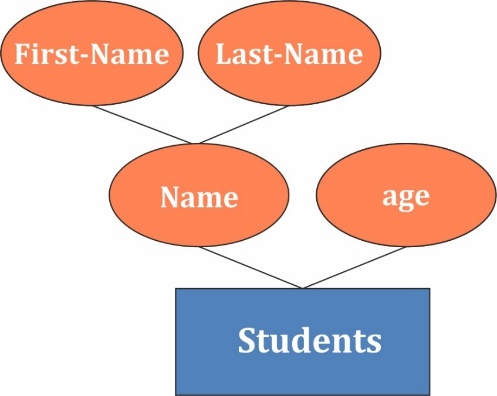
## 1. Simple Attributes:

Simple attributes are attributes that are drawn from the atomic value domains, which cannot be divided further.



## 2. Composite attribute:

Composite attributes are those attributes which are composed of many other simple attributes. Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first-name and last-name.

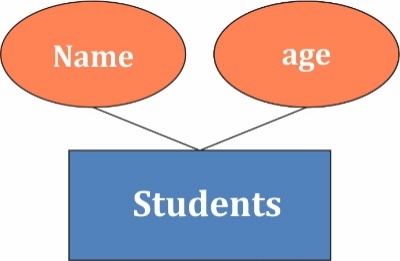


## Simple Vs. Composite Attributes

| **Simple Attribute** | **Composite Attribute** |
| --- | --- |
| The attribute which cannot further split into its components is a simple attribute. | The file system provides the details of data representation and storage of data. |
| Example: The marks of a student, the age of an employee etc. | Example: Name of the student can split into first, middle and last name. |

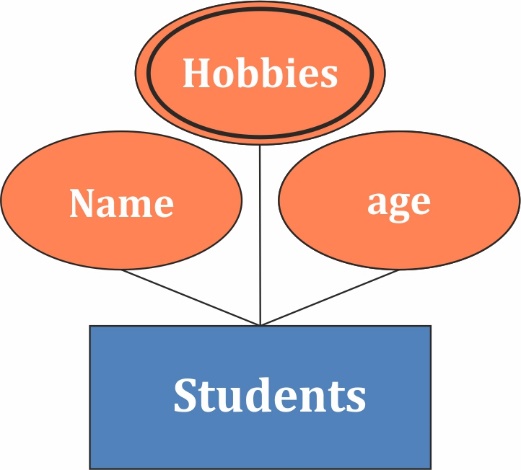
## 3. Single Valued Attributes:

Single valued attributes are those attributes which can take only one value for a given entity from an entity set. Single-value attributes contain single value. For example − age, gender etc.,.



## 4. Multi Valued Attributes:

Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set. Multi-value attributes may contain more than one values. For example, a person know more than one Languages, hobbies etc.



| **Simple Attribute** | **Composite Attribute** |
| --- | --- |
| The attribute which has a single value for each entity instance is known as single-valued attribute. There is no alternative of this value. | The attribute which takes up more than one value for each entity instance is known as multi valued attribute. |
| Example: The RollNo, DOB, Gender of a student will always be a single value. | Example: A person know more than one Languages, and there hobbies |

## Single-Valued Vs. Multi-Valued Attributes

## 5. Derived Attributes:

Derived attributes are those attributes which can be derived from other attribute(s).

Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, age of the student should not be saved directly in the database, instead it can be derived from Date of Birth.

# Entity in DBMS

**Entity:** An entity is anything in the real world, such as an object, class, person, or place. Objects that physically exist and are logically constructed in the real world are called entities. An entity is distinguishable from other entity.

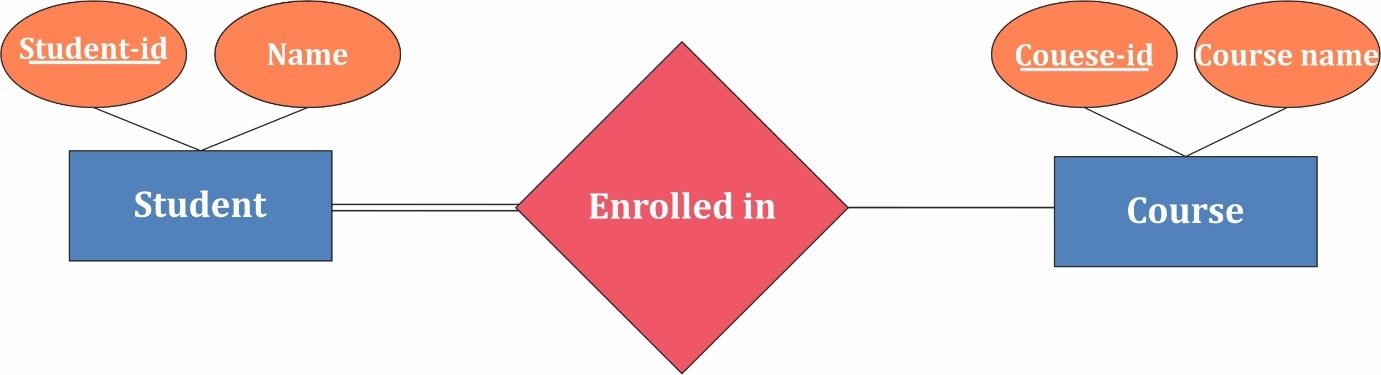
**Entity type:** The entity type is a collection of the entity having similar attributes.

**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. The entity set need not be disjoint.

In summary, an Entity is an object of a Type Entity and the set of all entities is called an entity set.

## Entities are of two types:

**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.



**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. It relies on another powerful entity for its unique identity. A weak entity is represented by a double-outlined rectangle in ER diagrams.



| **Strong Entity** | **Weak Entity** |
| --- | --- |
| Strong entity always has a primary key. | While a weak entity has a partial discriminator key. |
| Strong entity is not dependent on any other entity. | Weak entity depends on strong entity. |
| Strong entity is represented by a single rectangle. | Weak entity is represented by a double rectangle. |
| Two strong entity’s relationship is represented by a single diamond. | While the relation between one strong and one weak entity is represented by a double diamond. |
| Strong entities have either total participation or not. | While weak entity always has total participation. |

## Strong Entity v/s Weak Entity

# Relationships in DBMS

Definition

A relationship is defined as an association among several entities.

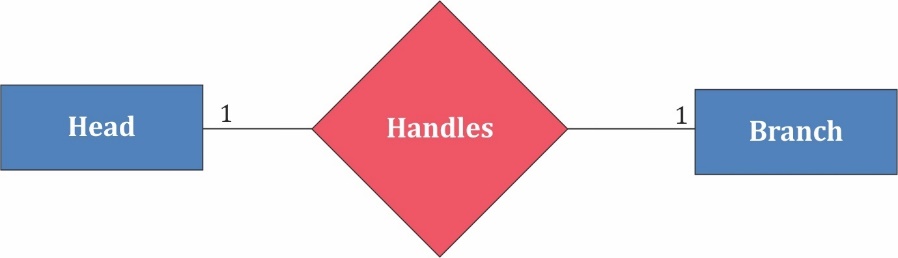
# Relationship Set

A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

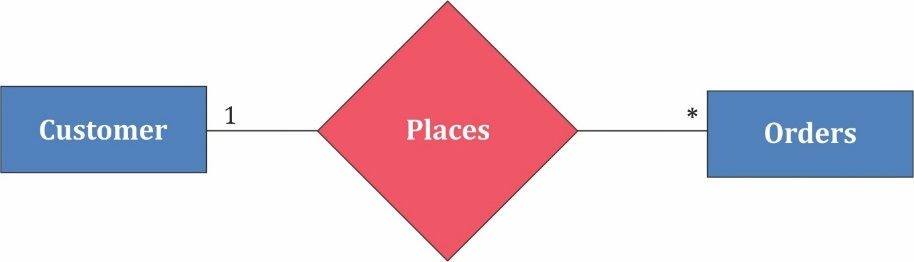
## Mapping Cardinalities:

express the number of entities to which another entity can be associated via a relationship. For binary relationship sets between entity sets A and B, the mapping cardinality must be one of:

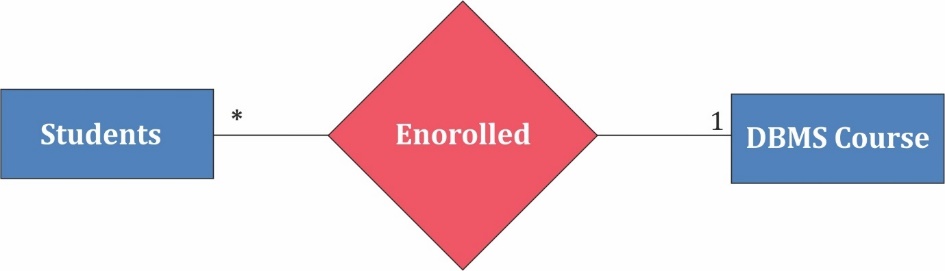
**One-to-one:** An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.



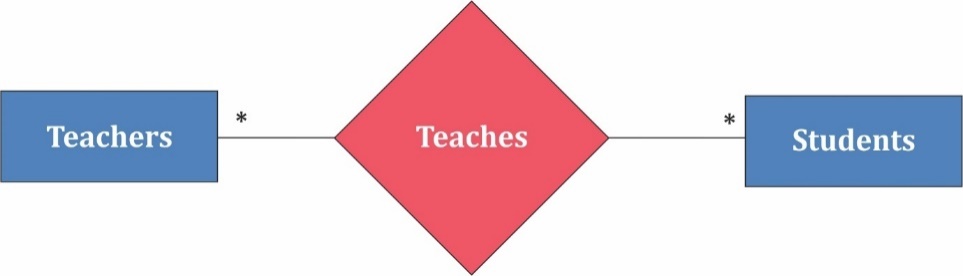
**One-to-many:** An entity in A is associated with any number in B. An entity in B is associated with at most one entity in A.



**Many-to-one:** An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.



**Many-to-many:** Entities in A and B are associated with any number from each other.



The appropriate mapping cardinality for a particular relationship set depends on the real world being modeled.

## Degree of a Relationship Set

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

Degree

Degree of a relationship set = Number of entity sets participating in a relationship set

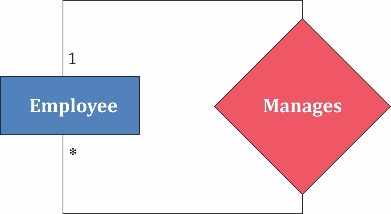
# Types of Relationship Sets

On the basis of degree of a relationship set, a relationship set can be classified into the following types-

* **Unary relationship set**
* **Binary relationship set**
* **Ternary relationship set**

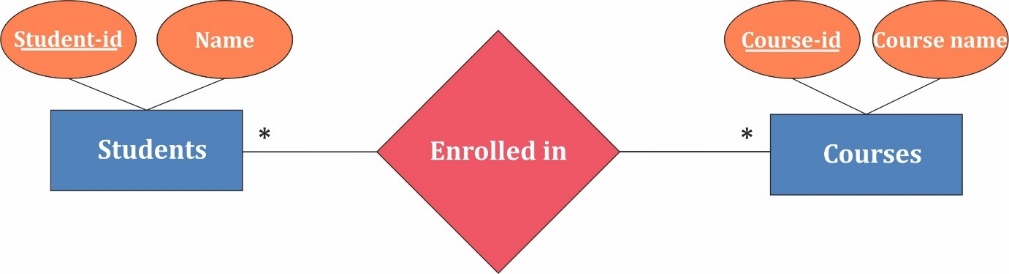
# Unary Relationship Set

Unary relationship set is a relationship set where only one entity set participates in a relationship set.



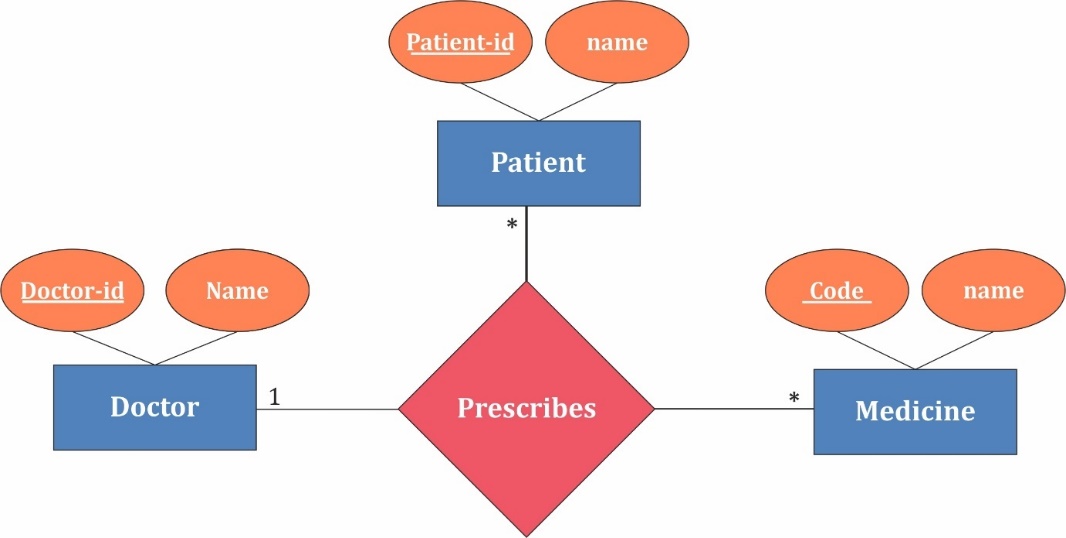
# Binary Relationship Set

Binary relationship set is a relationship set where two entity sets participate in a relationship set.



# Ternary Relationship Set

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.



# Participation Constraints

## Total Participation of an Entity set

Total participation of an entity set represents that each entity in entity set must have at least one relationship in a relationship set. It is also called mandatory participation. Total participation is represented using a double line between the entity set and relationship set.

## Partial participation of an Entity Set

Partial participation of an entity set represents that each entity in the entity set may or may not participate in the relationship instance in that relationship set. It is also called as optional participation. Partial participation is represented using a single line between the entity set and relationship set.

**Additional Features of the ER Model**

As the complexity of data increased, it became more and more difficult to use the traditional ER Model for database modelling. Hence some Additional Features were made to the existing ER Model to make it able to handle the complex applications better.

Hence, Three new concepts were added to the existing ER Model, they were:

* **Generalization**
* **Specialization**
* **Aggregration**

Some entities have relationships that form hierarchies. For instance, Employee can be an hourly employee or contracted employee.

In this relationship hierarchies, some entities can act as superclass and some other entities can act as subclass.

**Superclass**: An entity type that represents a general concept at a high level, is called superclass.

**Subclass**: An entity type that represents a specific concept at lower levels, is called subclass.

The subclass is said to inherit from superclass. When a subclass inherits from one or more superclasses, it inherits all their attributes. In addition to the inherited attributes, a subclass can also define its own specific attributes.

The symbol used for specialization/ Generalization is



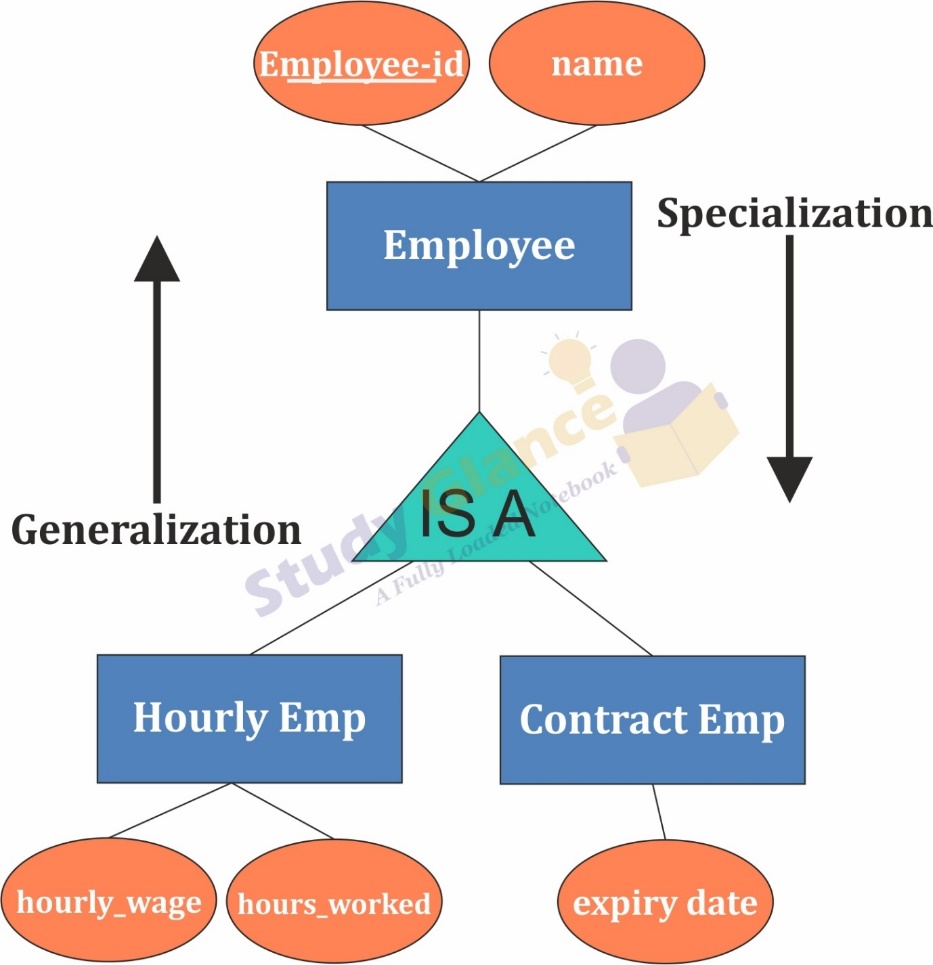
**Generalization**

Generalization is a process of extracting common properties from a set of entities and creating a generalized entity from it. It is a bottom-up approach, and it helps to reduce the size and complexity of the schema.

Example: Let us take two low-level entities as Car and Bus, and these two will have many common attributes and some specific attributes. And We will generalize and link the common attributes to the newly formed high-level entity named Vehicle.

**Specialization**

Specialization is opposite to Generalization. In this, entity is divided into subentities bases on their charactertics(distingvishing features). It breaks an entity into multiple entities from higher level to lower level. It is a top down approach.



**Aggregration**

Aggregation refers to the process by which entities are combined to form a single meaningful entity. The specific entities are combined because they do not make sense on their own. To establish a single entity, aggregation creates a relationship that combines these entities. The resulting entity makes sense because it enables the system to function well.

# Introduction to the Relational Model

The **Relational Model(RM)** 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.

## Terminologies used in Realation Model

* **Tables:** relations are saved in the table format. A table has two properties rows and columns
* **Attribute:** columns represent as attributes
* **Tuple:** A Row represent as Tuple
* **Relation Schema:** A relation schema represents the name of the relation with its attributes.
* **Degree:** The total number of attributes which in the relation is called the degree of the relation.
* **Cardinality:** Total number of rows present in the Table.
* **Column:** The column represents the set of values for a specific attribute.
* **Relation instance:** The set of tuples of a relation at a particular instance of time is called as relation instance.

## Advantages of Realation Model

* **Structural Independence**: Structural independence is an ability that allows us to make changes in one database structure without affecting other. The relational model have structural independence. Hence making required changes in the database is convenient in relational database model.
* **Conceptual Simplicity**: The relational model allows the designer to simply focus on logical design and not on physical design. Hence relational models are conceptually simple to understand.
* **Query Capability**: Using simple query language (such as SQL) user can get information from the database or designer can manipulate the database structure.
* **Easy design,maintenance and usage**: The relational models can be designed logically hence they are easy to maintain and use.

## Disadvantages of Realation Model

* Relational model requires powerful hardware and large data storage devices.
* May lead to slower processing time.
* Poorly designed systems lead to poor implementation of database systems.