Database System versus File System

DBMS	File Processing System
Minimal data redundancy problem in DBMS	Data Redundancy problem exits
Data Inconsistency does not exist	Data Inconsistency exist here
Accessing database is easier	Accessing is comparatively difficult
The problem of data isolation is not found in database	Data is scattered in various files and files may be of different format, so data isolation problem exists
Transactions like insert, delete, view, updating, etc are possible in database	In file system, transactions are not possible
Concurrent access and recovery is possible in database	Concurrent access and recovery is not possible
Security of data	Security of data is not good
A database manager (administrator) stores the relationship in form of structural tables	A file manager is used to store all relationships in directories in file systems.

Advantages of DBMS

1. Controlling of Redundancy:

Data redundancy refers to the duplication of data (i.e storing same data multiple times). In a database system, by having a centralized database and centralized control of data by the DBA the unnecessary duplication of data is avoided. It also eliminates the extra time for processing the large volume of data. It results in saving the storage space.

2. Improved Data Sharing:

DBMS allows a user to share the data in any number of application programs.

3. Data Integrity:

Integrity means that the data in the database is accurate. Centralized control of the data helps in permitting the administrator to define integrity constraints to the data in the database. For example: in customer database we can can enforce an integrity that it must accept the customer only from Noida and Meerut city.

4. Security:

Having complete authority over the operational data, enables the DBA in ensuring that the only mean of access to the database is through proper channels. The DBA can define authorization checks to be carried out whenever access to sensitive data is attempted.

5. Data Consistency:

By eliminating data redundancy, we greatly reduce the opportunities for inconsistency. For example: is a customer address is stored only once, we cannot have disagreement on the stored values. Also updating data values is greatly simplified when each value is stored in one place only. Finally, we avoid the wasted storage that results from redundant data storage.

6. Efficient Data Access:

In a database system, the data is managed by the DBMS and all access to the data is through the DBMS providing a key to effective data processing

7. Enforcements of Standards:

With the centralized of data, DBA can establish and enforce the data standards which may include the naming conventions, data quality standards etc.

8. Data Independence:

Ina database system, the database management system provides the interface between the application programs and the data. When changes are made to the data representation, the metadata obtained by the DBMS is changed but the DBMS is continues to provide the data to application program in the previously used way. The DBMs handles the task of transformation of data wherever necessary.

9. Reduced Application Development and Maintenance Time:

DBMS supports many important functions that are common to many applications, accessing data stored in the DBMS, which facilitates the quick development of application.

Disadvantages of DBMS

A. Increased Complexity

B. Requirement of New and Specialized Men power

C. Large Size of DBMS

Describing & Storing Data in DBMS

The user of a DBMS is ultimately concerned with some real-world enterprise, and the data to be stored describes various aspects of this enterprise. For example, there are students, faculty, and courses in a university, and the data in the university database describes these entities and their relationships.

There are three types of describing & storing data in DBMS:

1. **Data Models**
2. **Level of Abstraction**

3. **Data Independency**

1. **Data Models**

A database model or database schema is the structure or format of a database, described in a formal language supported by the database management system. Schemas are generally stored in a data dictionary. Although a schema is defined in text database language, the term is often used to refer to a graphical depiction of the database structure.

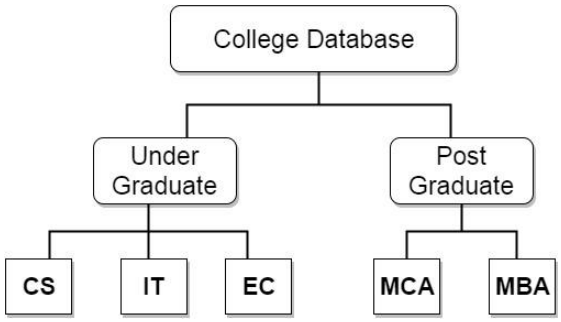
A. Hierarchical Database Model
B. Network Database Model
C. Relational Database Model
D. Object-Oriented Database Model

A. Hierarchical Database Model

This is the oldest form of database. This data model organizes the data in the tree structure i.e each child node can have only one parent node and at the top of the structure, there is a single parenthesis node. In this model a database record is a tree that consists of one or more groupings of fields called segments, which makeup the individual nodes of the tree. This model use one-to-many relationship

Advantage: Data access is quite predictable in structure and retrieval and updates can be highly optimized by a DBMS.

Disadvantage: The link is permanently established and cannot be modified which makes this model rigid.

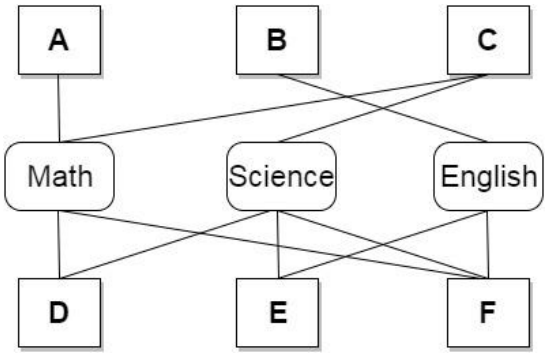


B. Network Database Model

The Network database model was developed as an alternative to the hierarchical database. This model expands on the hierarchical model by providing multiple paths among segments i.e more than one parent-child relationship. Hence this model allows one-to-one, one-to-many and many-to-many relationships

Supporting multiple paths in the data structure eliminates some of the drawbacks of the hierarchical model, the network model is not very practical.

Disadvantage: It can be quite complicated to maintain all the links.



C. Relational Database Model

The key differences between previous database models and relational database model is in terms of flexibility. A relational database represents all data in the database as simple two-dimensional tables called relations. Each row of a relational table, called tuple, represents a data entity with columns of the table representing attributes (fields). The allowable values for these attributes are called the domain. Each row in a relational table must have a unique primary key and also has some secondary keys which correspond with primary keys in other tables.

Advantage: Provides flexibility that allows changes to the database structure to be easily accommodated. It facilitates multiple views of the same database for different users.

For example: COLLEGE table has Batch_Year as primary key and has secondary keys Student_ID and Course_ID, these keys serve as primary keys for STUDENT and COURSE tables.

Student Table

Student_ID	Student_Name
101	Shubham
102	Rajat

Course Table

Course_ID	Course_Name
14	Java
16	Android

College Table

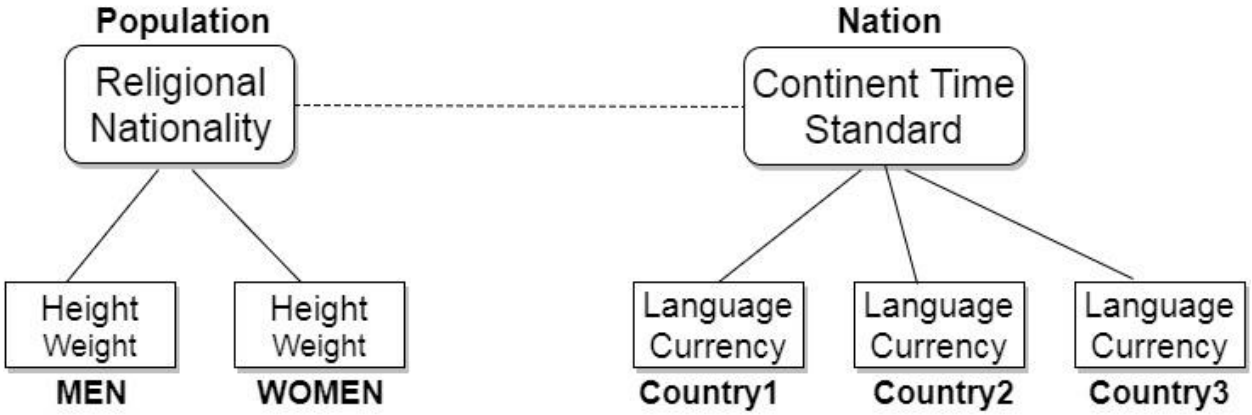
Batch_Year	Student_ID	Course_ID	Teacher_Name	Teacher_Number

2012-16	101	14	Jack	9876543
2013-17	102	16	Tom	9823451

D. Object-Oriented Database Model

The relational database model has a widely variety of applications. However it does not easily support the distribution of one database across a number of servers. Due to this, object-oriented database management system was developed. In these databases, the users can define own data access methods, the representation of data and the method of manipulating it. An object-oriented database stores and maintains objects.

Example: The class population is the root of class hierarchy, which includes the Nation class. The Population class is also the root of two sub-class, men and women. The Nation class is the root of other sub-classes country1, country2, country3. Note that each class has its own set of attributes apart from the root class's attributes.



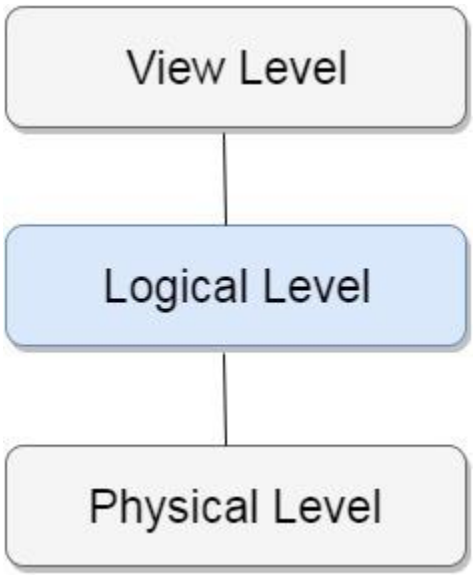
2. Level of Abstraction

For the system to be usable, it must retrieve data efficiently. The need for efficiency has led designers to use complex data structures to represent data in the database. Developers hide the complexity from users through several levels of abstraction to simplify user's interactions with the system.

Physical Level: The lowest level of abstraction describes how the data are actually stored.

Logical Level: The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data.

View Level: The highest level of abstraction describes only part of the entire database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many view for the same database.



3. Data Independence

Data Independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
Data Independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed.

There are two types of data independence:

A. **Logical Data Independence**
B. **Physical Data Independence**

A. Logical Data Independence

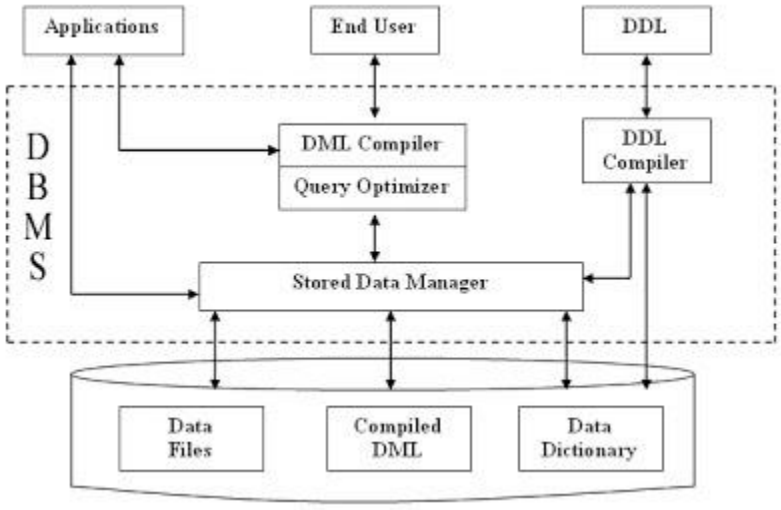
It is the capacity to change the conceptual schema without having to change external schemas or application program. We may change the conceptual schema to expand the database, to change constraints, or to reduce the database.

B. Physical Data Independence

It is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schema need not to be changed as well. Changes to the internal schema may be needed because some physical files had to be reorganized.

STRUCTURE OF DBMS

DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations (insert, delete, update and retrieval) on the database. The components of DBMS perform these requested operations on the database and provide necessary data to the users. The various components of DBMS are shown below:



1. **DDL Compiler** - Data Description Language compiler processes schema definitions specified in the DDL. It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.

2. **DML Compiler and Query optimizer** - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access. The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.

3. **Data Manager** - The Data Manager is the central software component of the DBMS also knows as Database Control System.
The Main Functions Of Data Manager Are: –

   a. Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.
   b. Controls DBMS information access that is stored on disk.
   c. It also controls handling buffers in main memory.

d. It also enforces constraints to maintain consistency and integrity of the data.
e. It also synchronizes the simultaneous operations performed by the concurrent users.
f. It also controls the backup and recovery operations.

4. Data Dictionary - Data Dictionary is a repository of description of data in the database. It contains information about

a. Data - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
b. Relationships between database transactions and data items referenced by them which is useful in determining which transactions are affected when certain data definitions are changed.
c. Constraints on data i.e. range of values permitted.
d. Detailed information on physical database design such as storage structure, access paths, files and record sizes.
e. Access Authorization - is the Description of database users their responsibilities and their access rights.
f. Usage statistics such as frequency of query and transactions.

g. Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS.
h. Importance of Data Dictionary -
i. Data Dictionary is necessary in the databases due to following reasons:
j. It improves the control of DBA over the information system and user's understanding of use of the system.
k. It helps in documentating the database design process by storing documentation of the result of every design phase and design decisions.
l. It helps in searching the views on the database definitions of those views.
m. It provides great assistance in producing a report of which data elements (i.e. data values) are used in all the programs.
n. It promotes data independence i.e. by addition or modifications of structures in the database application program are not affected.

5. Data Files - It contains the data portion of the database.

6. Compiled DML - The DML complier converts the high level Queries into low level file access commands known as compiled DML.

7. End Users - Actual user of database.

Entity Relationship Model: -

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Entity Set

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Entity- Represented by Rectangles

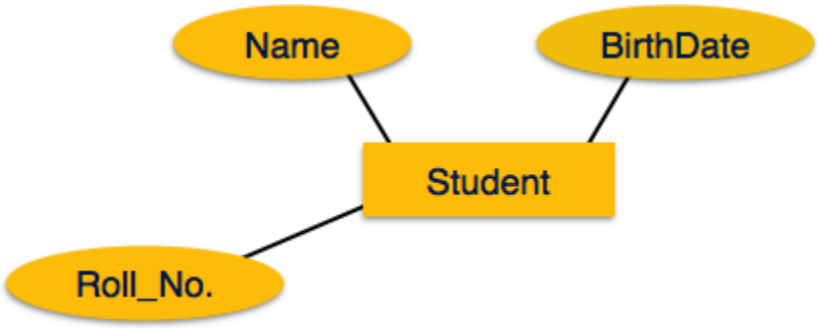
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes (Represented by Ellipses)

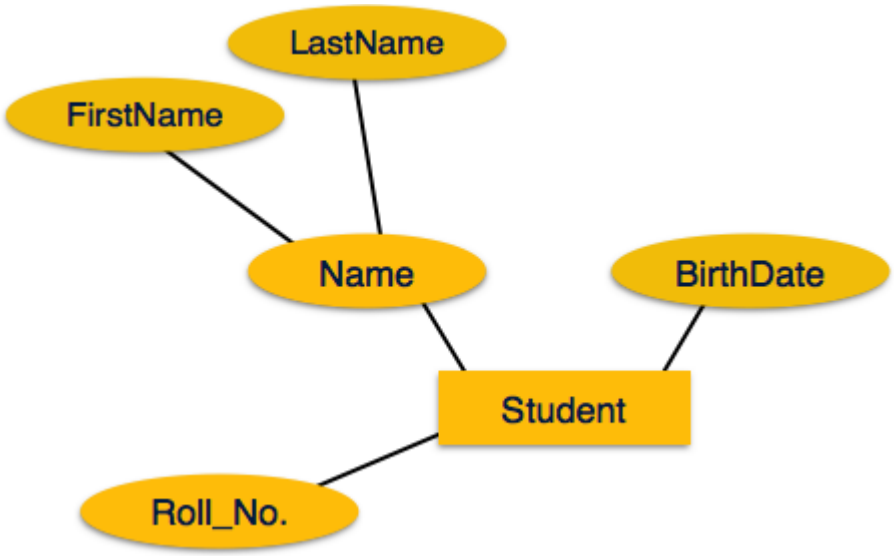
Entities are represented by means of their properties, called **attributes**. All attributes have values. For example, a student entity may have name, class, and age as attributes.

Types of Attributes

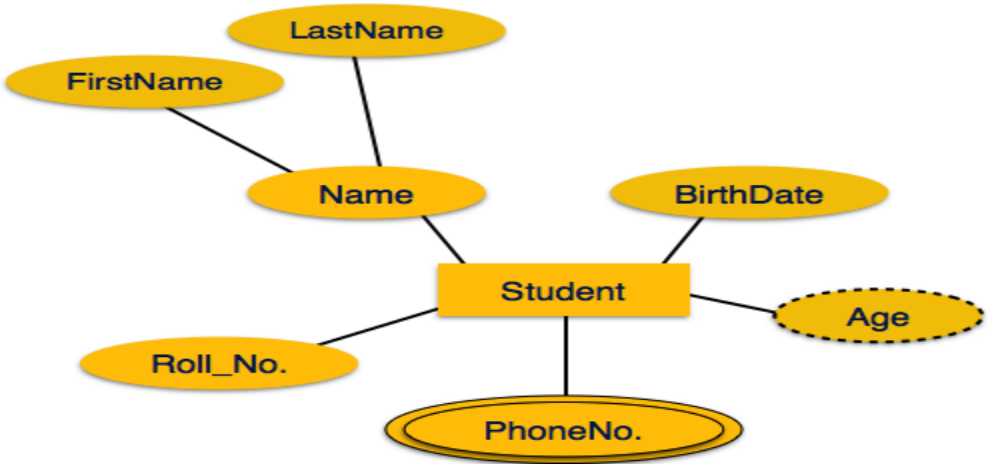
- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



- **Composite attribute** − Composite attributes are made of more than one simple attribute. **For example**, a student's complete name may have **first_name and last_name**.
- If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



- **Derived attribute (dashed ellipse)** − 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, average_salary** in a department should not be saved directly in the database, instead it can be derived. For another example, **age** can be derived from data_of_birth**.**



- **Single-value attribute** − Single-value attributes contain single value. For example − Social_Security_Number.
- **Multi-value attributes (Double Ellipses)** − Multi-value attributes may contain more than one value. For example, a person can have more than one **phone number**, email_address, etc.

Entity-Set and Keys

Key is **an attribute** or **collection of attributes** that uniquely identifies an entity among entity set.

For example, the **roll_number** of a student makes him/her identifiable among students.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship

The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, **Works_at** and **Enrolls** are called relationships.

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**.

Degree of Relationship

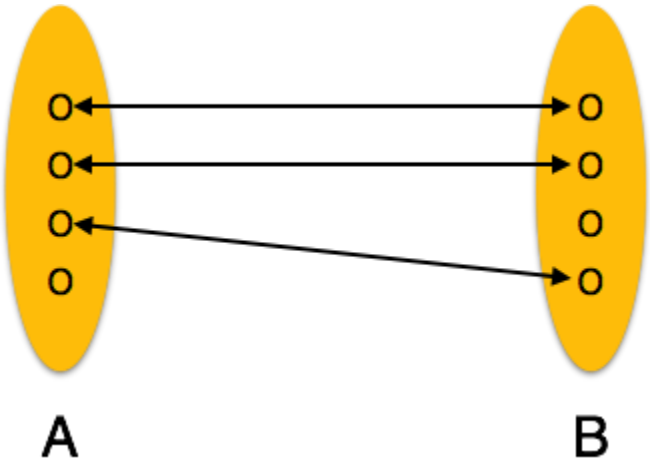
The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

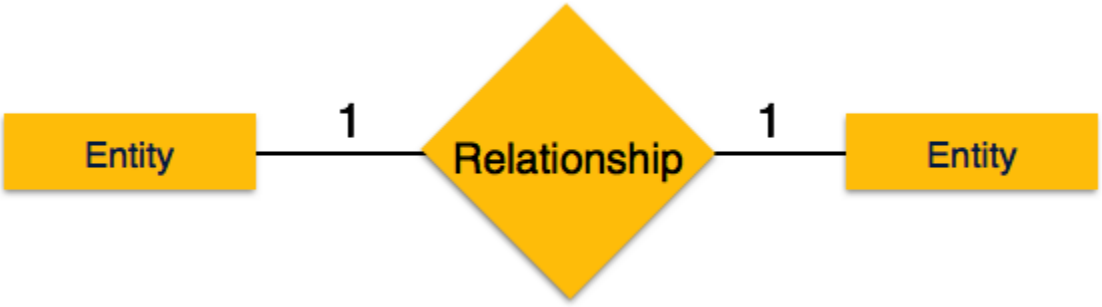
Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set. Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

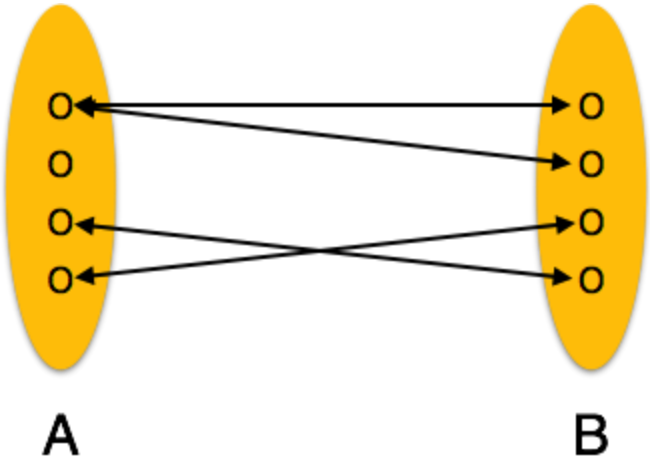
- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



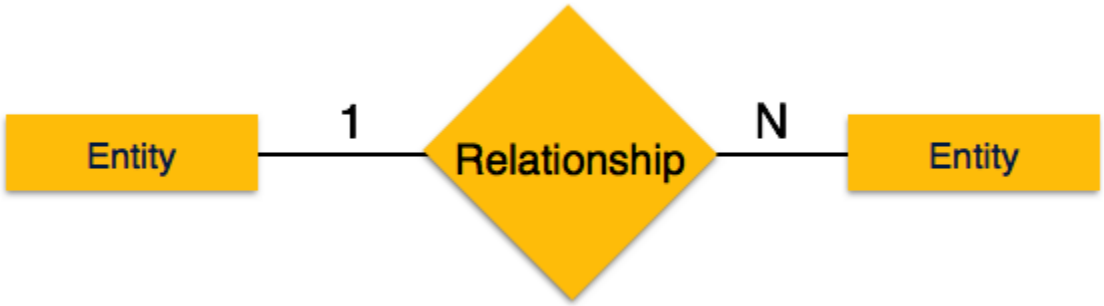
- **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



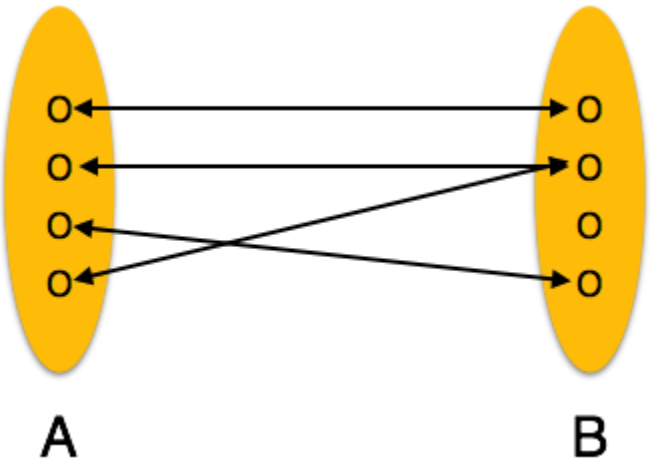
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.

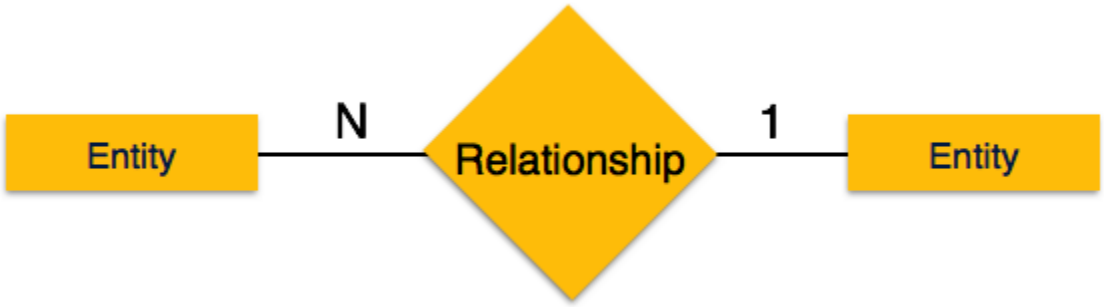


- **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.

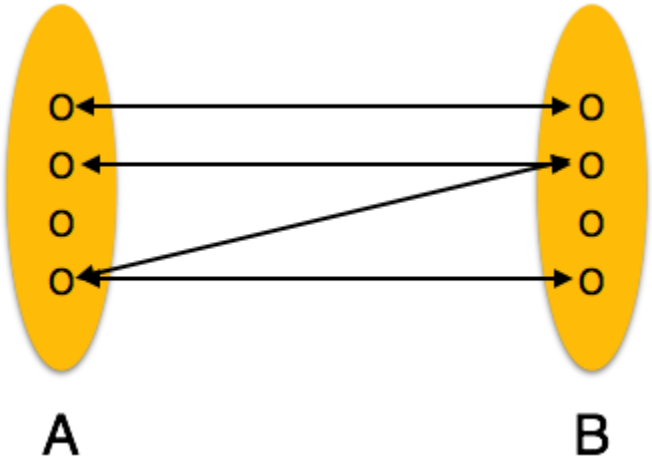


- **Many-to-one** − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one

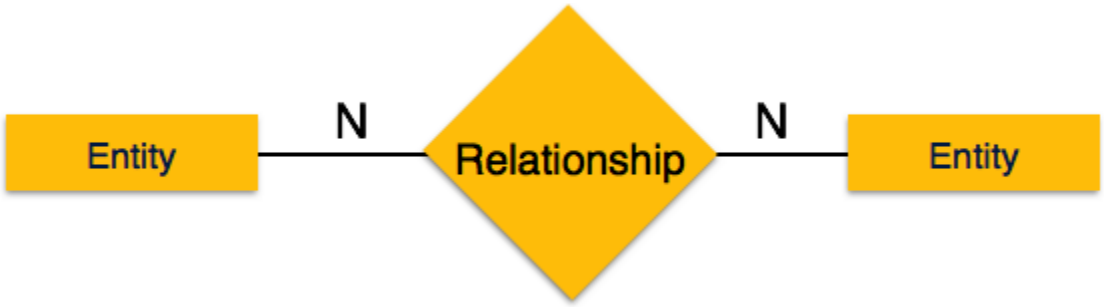
instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.



- **Many-to-many** − One entity from A can be associated with more than one entity from B and vice versa.

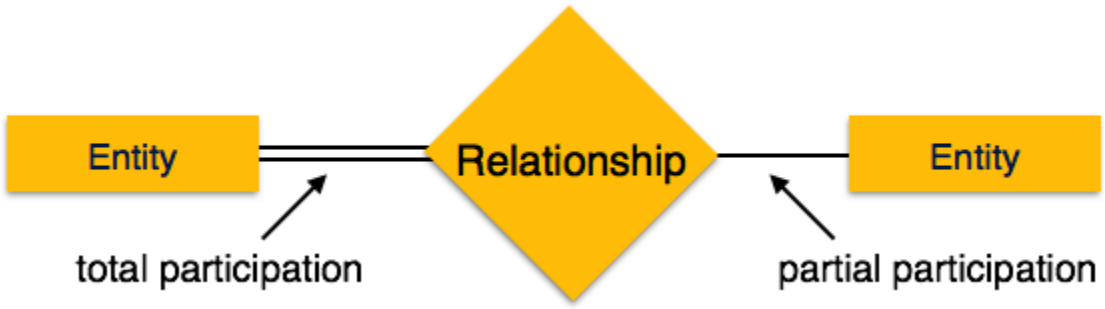


- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



Participation Constraints

- **Total Participation** − each entity is involved in the relationship. Total participation is represented by double lines.
- **Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.



Types of DBMS Entities

The following are the types of entities in DBMS:

Strong Entity

The strong entity has a primary key. Weak entities are dependent on strong entity. Its existence is not dependent on any other entity.

Strong Entity is represented by a single rectangle:

Weak Entity

The weak entity in DBMS do not have a primary key and are dependent on the parent entity. It mainly depends on other entities.

Weak Entity is represented by double rectangle:

- **Class hierarchy**

The ER Model has the power of expressing database entities in a conceptual hierarchical manner. As the hierarchy goes up, it generalizes the view of entities, and as we go deep in the hierarchy, it gives us the detail of every entity included.

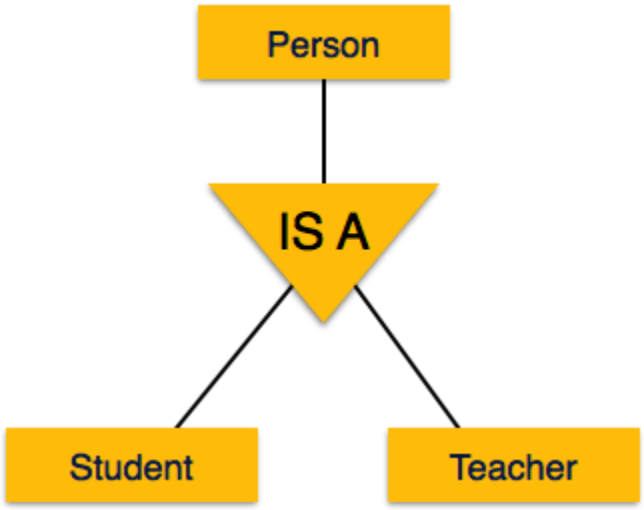
Going up in this structure is called **generalization**, where entities are clubbed together to represent a more generalized view. For example, a particular student named Mira can be

generalized along with all the students. The entity shall be a student, and further, the student is a person. The reverse is called **specialization** where a person is a student, and that student is Mira.

Class hierarchy can be viewed one of two ways

1. **Specialization (Top Down Approach)**

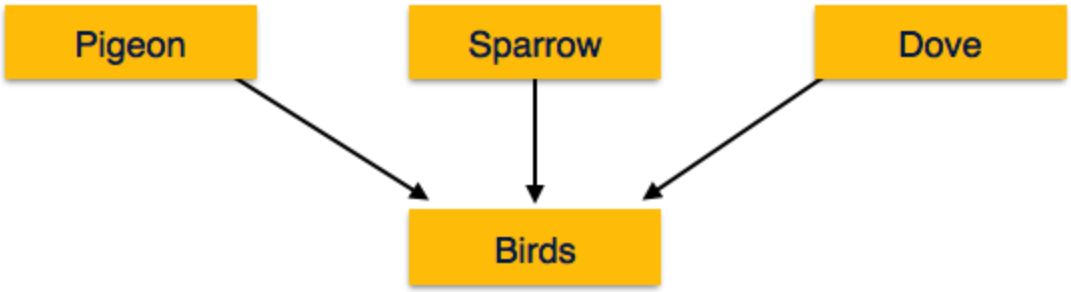
Specialization is the opposite of generalization. In specialization, a group of entities is divided into sub-groups based on their characteristics. Take a group 'Person' for example. A person has name, date of birth, gender, etc. These properties are common in all persons, human beings. But in a company, persons can be identified as employee, employer, customer, or vendor, based on what role they play in the company.



Similarly, in a school database, persons can be specialized as teacher, student, or a staff, based on what role they play in school as entities.

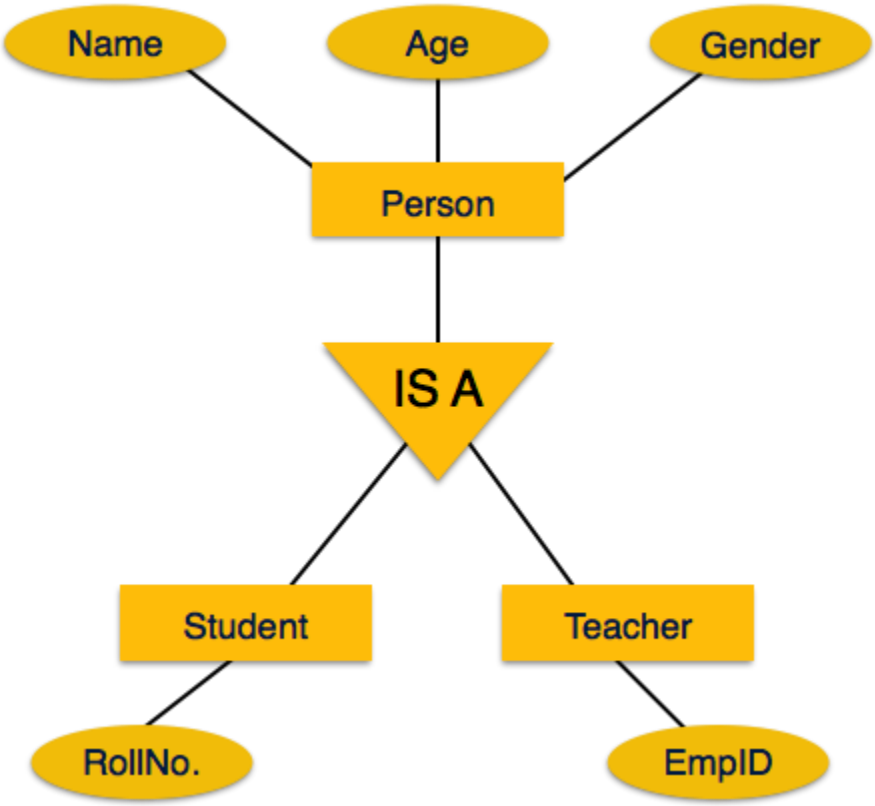
2. **Generalization (Bottom Up Approach)**

As mentioned above, the process of generalizing entities, where the generalized entities contain the properties of all the generalized entities is called generalization. In generalization, a number of entities are brought together into one generalized entity based on their similar characteristics. For example, pigeon, house sparrow, crow and dove can all be generalized as Birds.



Inheritance

We use all the above features of ER-Model in order to create classes of objects in object-oriented programming. The details of entities are generally hidden from the user; this process known as **abstraction**. Inheritance is an important feature of Generalization and Specialization. It allows lower-level entities to inherit the attributes of higher-level entities.



For example, the attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

E-R Diagram of University System