What is Database?

- A database is a systematic collection of data. They support electronic storage and manipulation of data. Databases make data management easy.
- A database is usually controlled by database management system(DBMS).
- **Example:** Face book, It needs to store, manipulate, and present data related to members, their friends, member activities, messages, advertisements, and a lot more,

3.1 Database characteristics:

A database management system has following characteristics:

1. **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationships between tables which makes the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.
2. **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows **Normalisation** which divides the data in such a way that repetition is minimum.
3. **Data Consistency:** On Live data, i.e. data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
4. **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it(update, insert, delete data) at the same time and still manages to maintain the data consistency.
5. **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. **Security:** The DBMS also takes care of the security of data, protecting the data from un-authorised access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.

7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

➢ 3.1.1 Data Independence (Logical and Physical)

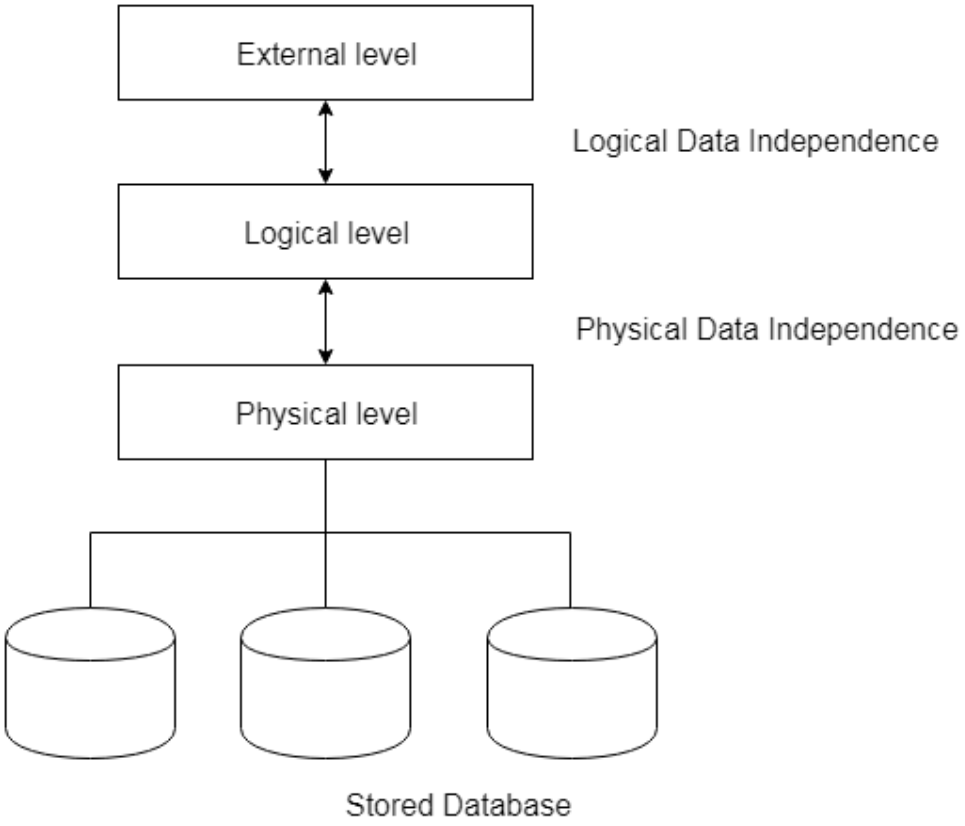
- A database system normally contains a lot of data in addition to user's data.
- For example, it stores data about data, known **as metadata**, to locate and retrieve data easily.
- It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users.
- If the entire data is dependent, it would become a tedious and highly complex job.
- Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.
- There are two types of data independence:

1. Logical Data Independence

- Logical data is data about database, that is, it stores information about **how data is managed inside.**
- For example, a table (relation) stored in the database and all its constraints, applied on that relation.
- Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk.
- If we do some changes on table format, it should not change the data residing on the disk.
- Logical data independence occurs at **the user interface level**.

2. Physical Data Independence

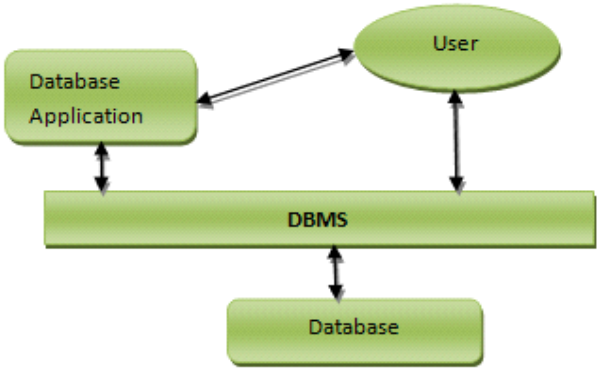
- All the schemas are logical, and the actual data is stored in bit format on the disk.
- Physical data independence is **the power to change the physical data without impacting the schema or logical data.**
- For example, in case we want to change or upgrade the storage system itself – suppose we want to **replace hard-disks with SSD** – it should not have any impact on the logical data or schemas.
- 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.



➢ 3.1.2 Components of Database (User, Application , DBMS, Database)

- **User: –**
- Users are the one who really uses the database. Users can be administrators, developers, or end-users.
- **Data or Database: –**
- Data is one of the important factors of the database.
- A very huge amount of data will be stored in the database and it forms the main source for all other components to interact with each other.
- There are two types of data.
- One is user data. It contains the data which is responsible for the database, i.e.; based on the requirement, the data will be stored in the various tables of the database in the form of rows and columns.
- Another data is Metadata. It is known as 'data about data', i.e.; it stores the information like how many tables, their names, how many columns and their names, primary keys, foreign keys, etc. basically these metadata will have information about each table and their constraints in the database.
- **DBMS: –**

- This is the software that helps the user to interact with the database. It allows the users to insert, delete, update, or retrieve the data. All these operations are handled by query languages like MySQL, Oracle, etc.
- **Database Application: –**
- It the application program which helps the users to interact with the database by means of query languages. The database application will not have any idea about the underlying DBMS.



3.1.3 Database Architecture (1-tier, 2-tier, 3-tier)

- A Database Management system is not always directly available for users and applications to access and store data in it.
- A Database Management system can be **centralised**(all the data stored at one location), **decentralised**(multiple copies of database at different locations) or **hierarchical**, depending upon its architecture.

1-tier DBMS architecture

- When the database is directly available to the user for using it to store data. Generally such a setup is used for local application development, where programmers communicate directly with the database for quick response.
- **Advantages**

1. Easy to implement and optimize performance.
2. Do not have compatibility or Context switching issues.

3. The cost of deployment is less eg - development and management cost.

➢ **Disadvantages**

1. Do not support remote/ distributed access for data resources.
2. higher maintenance.
3. The cost of the central mainframe is high.

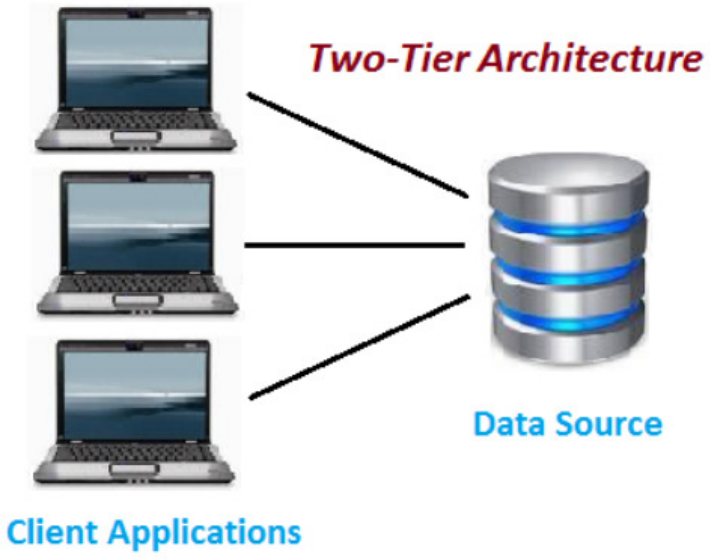
➢ Database Architecture is logically of two types:

1. 2-tier DBMS architecture(High level application)
2. 3-tier DBMS architecture (Web based application)

❖ 2-tier DBMS Architecture

➢ The two-tier is based on **Client Server architecture**. The two-tier architecture is like client server application.

➢ The direct communication takes place between client and server. There is no intermediate between client and server. Because of tight coupling a 2 tiered application will run faster.



- The above figure shows the architecture of two-tier. Here the direct communication between client and server, there is no intermediate between client and server.
- real life example of Railway Reservation two-tier architecture:
- In this application the Staff user is an end user who is using Railway reservation application software. He gives inputs to the application software and it sends requests to Server. So here both Database and Server are incorporated with each other, so this technology is called as "Client-Server Technology".
- The Two-tier architecture is divided into two parts:

  1) Client Application (Client Tier)
  2) Database (Data Tier)
- On client application side the code is written for saving the data in the SQL server database. Client sends the request to server and it process the request & send back with data. The main problem of two tier architecture is the server cannot respond multiple request same time, as a result it cause a data integrity issue.
- **Advantages:**

1. Easy to maintain and modification is bit easy
2. Communication is faster

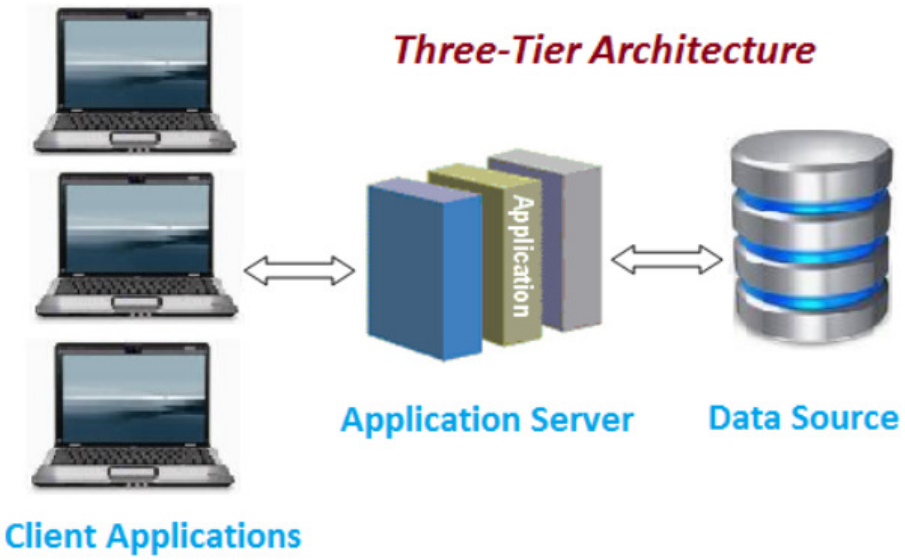
- **Disadvantages:**

1. In two tier architecture application performance will be degrade upon increasing the users.
2. Cost-ineffective

❖ 3-tier DBMS Architecture

- **Three-tier architecture** Three layers in the three tier architecture are as follows:

- **1) Client layer**
  2) Business layer
  3) Data layer



1) Client layer:

- It is also called as Presentation layer which contains UI part of our application.
- This layer is used for the design purpose where data is presented to the user or input is taken from the user.
- For example designing registration form which contains text box, label, button etc.

2) Business layer:

- In this layer all business logic written like **validation of data, calculations, data insertion** etc.
- This acts as a interface **between Client layer and Data Access Layer**.
- This layer is also called the intermediary layer helps to make communication faster between client and data layer.

3) Data layer:

- In this layer actual database is comes in the picture.
- Data Access Layer contains **methods to connect with database** and to perform insert, update, delete, get data from database based on our input data.
- **Advantages**

1. High performance, lightweight persistent objects
2. Scalability – Each tier can scale horizontally
3. Better Re-use
4. Improve Data Integrity
5. Improved Security – Client is not direct access to database.
6. Easy to maintain and modification is bit easy

- **Disadvantages**

1. Increase Complexity/Effort

➢ comparison between 1 Tire,2-Tire, 3-Tire

1. Client -> Which requests the data

2. Server -> Which serves the data

3. Database -> The place where data is kept

- Now let us come to the different architecture of DBMS :
- **1 Tier** => The Client, Server and Database resides on the same machine.
- **2 Tier** => The client on one machine and the server and database on one machine, i.e. two machines.
- **3 Tier** => We have three different machines one for each client, server and a separate machine dedicated to database.

➢ 3.2 Database Models (Hierarchical, Network, E/R, Relational)

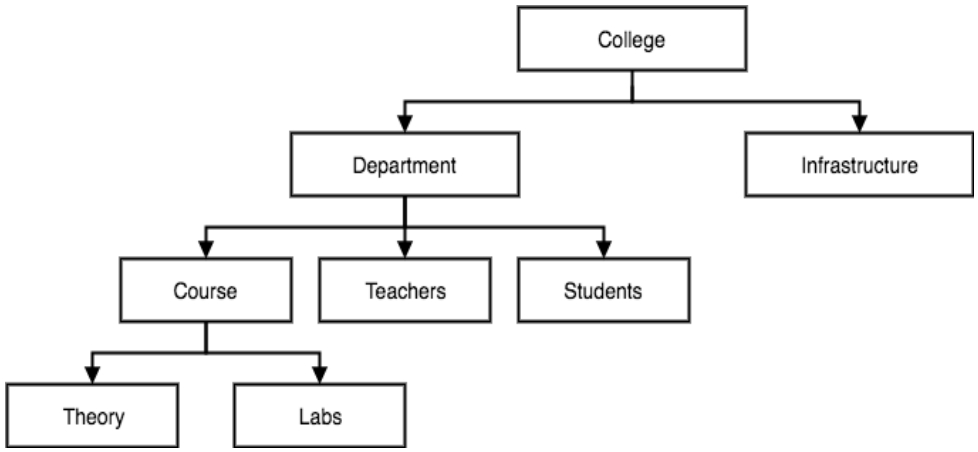
❖ Data Models

- A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system.
- While the Relational Model is the most widely used database model, there are other models too:

1. Hierarchical Model
2. Network Model
3. Entity-Relational Model
4. Relational Model

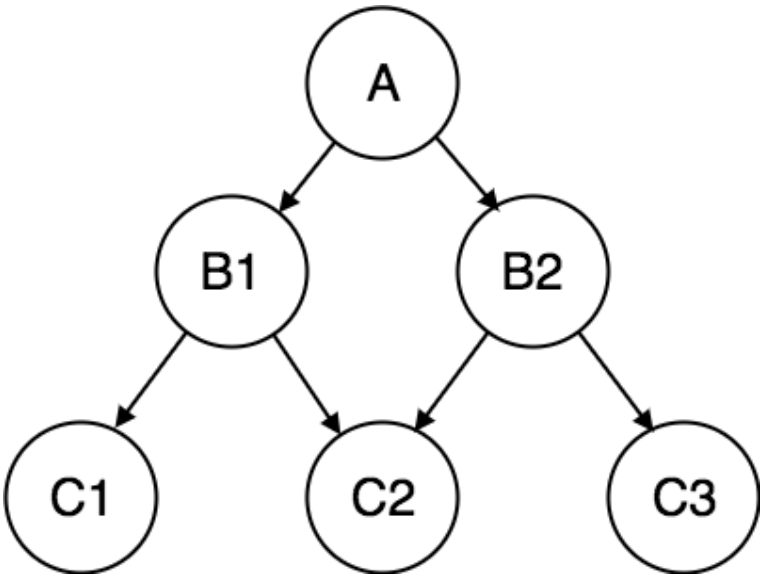
1. Hierarchical Model

- This database model organizes data into a tree-like-structure, with a single root, to which all the other data is linked.
- The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- In this model, a child node will only have a single parent node.
- This model efficiently describes many real-world relationships like index of a book, recipes etc.
- In hierarchical model, data is organized into tree-like structure with one **one-to-many relationship between two different types of data**.
- For example, one department can have many courses, many professors and of-course many students.



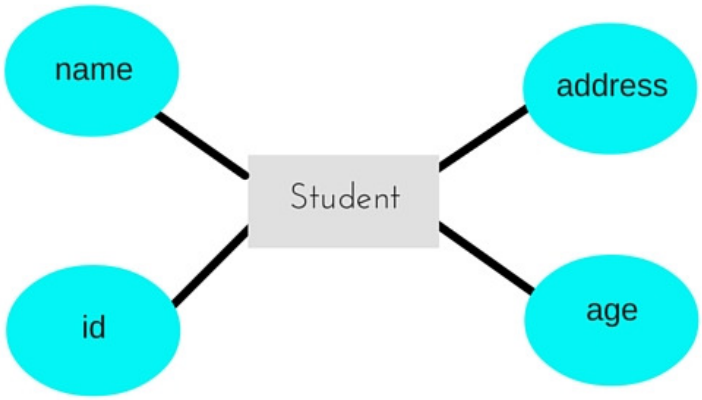
2. Network Model

- This is an extension of the Hierarchical model.
- In this model data is organised more like a graph, and are allowed to have more than one parent node.
- In this database model data is more related as more relationships are established in this database model. Also, as the data is more related, hence accessing the data is also easier and fast.
- This database model was used to map many-to-many data relationships.
- This was the most widely used database model, before Relational Model was introduced.



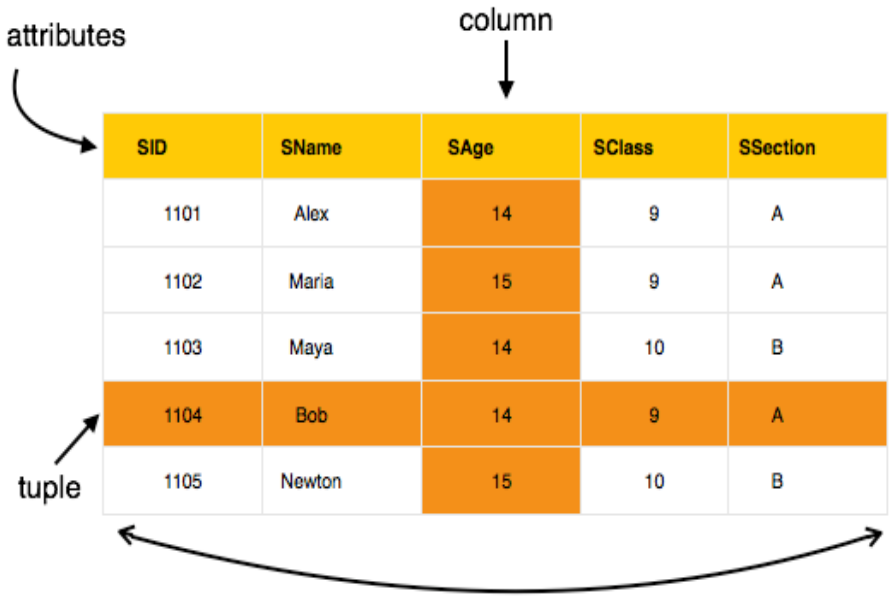
3. Entity-Relationship Model

- In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.
- Different entities are related using relationships.
- E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.
- This model is good to design a database, which can then be turned into tables in relational model(explained below).
- Let's take an example, If we have to design a School Database, then Student will be an entity with attributes name, age, address etc. As Address is generally complex, it can be another entity with attributes street name, pincode, city etc, and there will be a relationship between them.
- Relationships can also be of different types. To learn about E-R Diagrams in details, click on the link.



4. Relational Model

- In relational data model, data is handled on a conceptual rather than physical basis. In this we can process entire file with single statement written in query language(SQL).
- In this model, data is organised in two-dimensional tables and the relationship is maintained by storing a common field.
- This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, we can say the only database model used around the world.
- The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table. A table consists of unique attributes(columns) and tuples(rows).
- Hence, tables are also known as relations in relational model.
- Sometimes the value to be inserted into a particular cell will be unknown, or it will not have any value. This is represented by a null. Please note that null is not the same as zero, blank or an empty string.
- A database whose logical organization is based on relational data model is a Relational Database.
- A DBMS that manages the relational database is known as RDBMS



SID	SName	SAge	SClass	SSection
1101	Alex	14	9	A
1102	Maria	15	9	A
1103	Maya	14	10	B
1104	Bob	14	9	A
1105	Newton	15	10	B

student_Id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_Id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_Id	subject_Id	marks
1	1	98
1	2	78
2	1	76
3	2	88

➢ 3.2.1 E/R model : Entity, Relationship, Attribute

An ER diagram is mainly composed of following three components-

1. Entity Sets
2. Attributes
3. Relationship Set

1.Entity Sets-

- **Entity**: A real world object distinguishable from other objects like employee, student, car, house. etc.
- An entity is described using a set of attributes.
- Each entity has specific attribute that is the characteristics of entity. (i.e **student entity** has **rollno, name,age** these are the **attributes** of student entity
- **Entity Set :**An entity set is a set of same type of entities.
- example : all student
- An entity refers to any object having-
- A **physical existence** such as a particular person, office, house or car.
- A **conceptual existence** such as a school or a company.
- All entities in an entity set have the **same set of attributes**

Example-

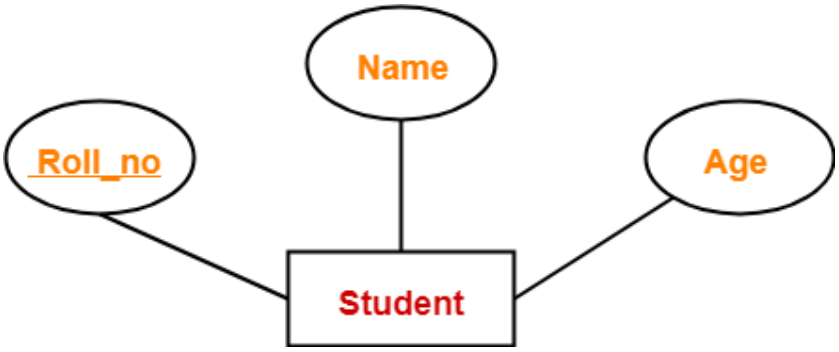
Consider the following Student table-

Roll_no	Name	Age
1	Akshay	20
2	Rahul	19
3	Pooja	20
4	Aarti	19

This complete table is referred to as "**Student Entity Set**" and **each row represents** an "**Entity**".

Representation as ER Diagram-

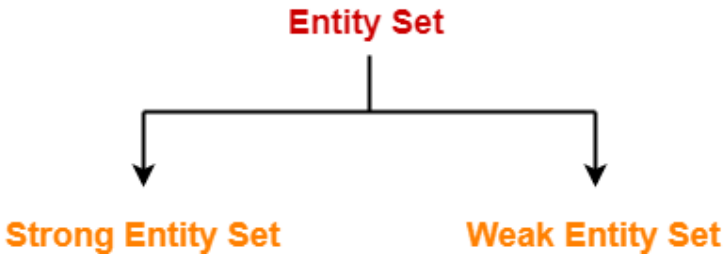
The above table may be represented as ER diagram as-



Here,

- **Roll_no** is a primary key that can identify each entity uniquely.
- Thus, by using student's roll number, a student can be identified uniquely.

Entity set types-

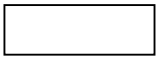
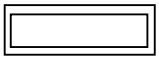


- **Symbol:-**

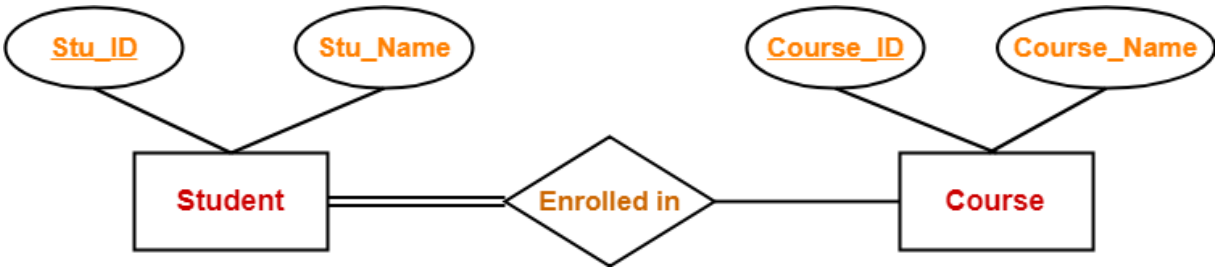
1. Strong Entity Set-

- A strong entity set is an entity set that contains sufficient attributes **to uniquely identify all** its entities.
- In other words, a **primary key** exists for a strong entity set.
- Primary key of a strong entity set is represented by **underlining it.**

Symbols Used-

	A **single rectangle** is used for representing a strong entity set.
	A **diamond symbol** is used for representing the relationship that exists between two strong entity sets.
	A double rectangle is used for representing a weak entity set.
	A single line is used for representing the connection of the strong entity set with the relationship set.
	A double line is used for representing the total participation of an entity set with the relationship set.

Example-



In this ER diagram,

- Two strong entity sets "**Student**" and "**Course**" are related to each other.
- Student ID and Student name are the attributes of entity set "Student".
- Student ID is the primary key using which any student can be identified uniquely.
- Course ID and Course name are the attributes of entity set "Course".
- Course ID is the primary key using which any course can be identified uniquely.
- Double line between Student and relationship set signifies total participation.
- It suggests that each student must be enrolled in at least one course.
- Single line between Course and relationship set signifies partial participation.
- It suggests that there might exist some courses for which no enrolments are made.

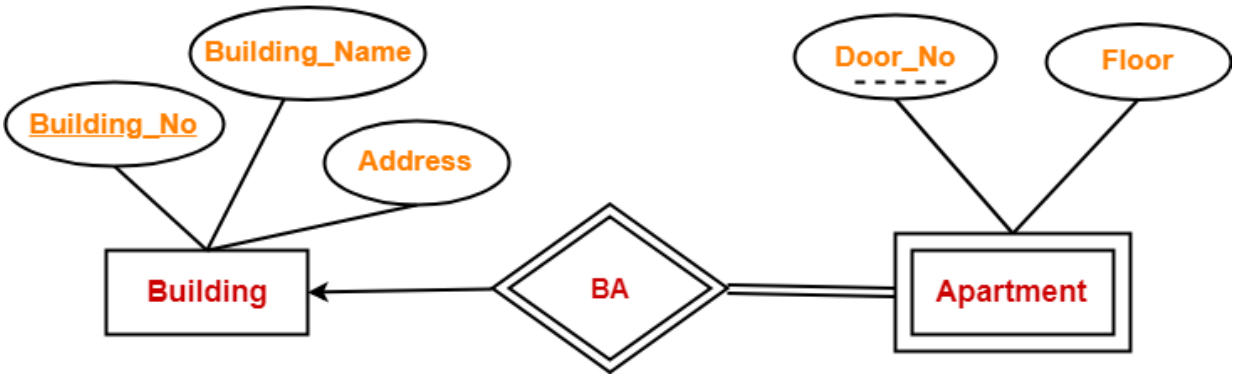
2. Weak Entity Set-

- A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.
- In other words, a primary key does not exist for a weak entity set.
- However, it contains a partial key called as a **discriminator.**
- Discriminator can identify a group of entities from the entity set.
- **Discriminator** is represented by **underlining with a dashed line.**
- weak entity is represented using a **double rectangle.**

Symbols Used-

- A **double rectangle** is used for representing a weak entity set.
- A **double diamond symbol** is used for representing the relationship that exists between the strong and weak entity sets and this relationship is known as **identifying relationship**.
- A **double line** is used for representing the **connection of the weak entity** set with the relationship set.
- **Total participation always exists in the identifying relationship**.

Example-



In this ER diagram,

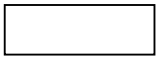
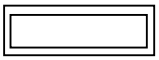
- One **strong entity** set "**Building**" and one **weak entity** set "**Apartment**" are related to each other.
- Strong entity set "Building" has "**building_no**" as its **primary key**.
- "**Door_no**" is the **discriminator** of the weak entity set "**Apartment**".

- This is **because** "**door_no**" alone cannot identify an apartment uniquely as there may be **several other buildings having the same door number**.
- **Double line** between Apartment and relationship set signifies **total participation**.
- It suggests that **each apartment** must be present in at **least one building**.
- **Single line** between Building and relationship set signifies **partial participation**.
- It suggests that there might exist **some buildings which has no apartment**.

To uniquely identify any apartment,

- First, building_no is required to identify the particular building.
- Secondly, door_no of apartment is required to uniquely identify the apartment.
- Thus,

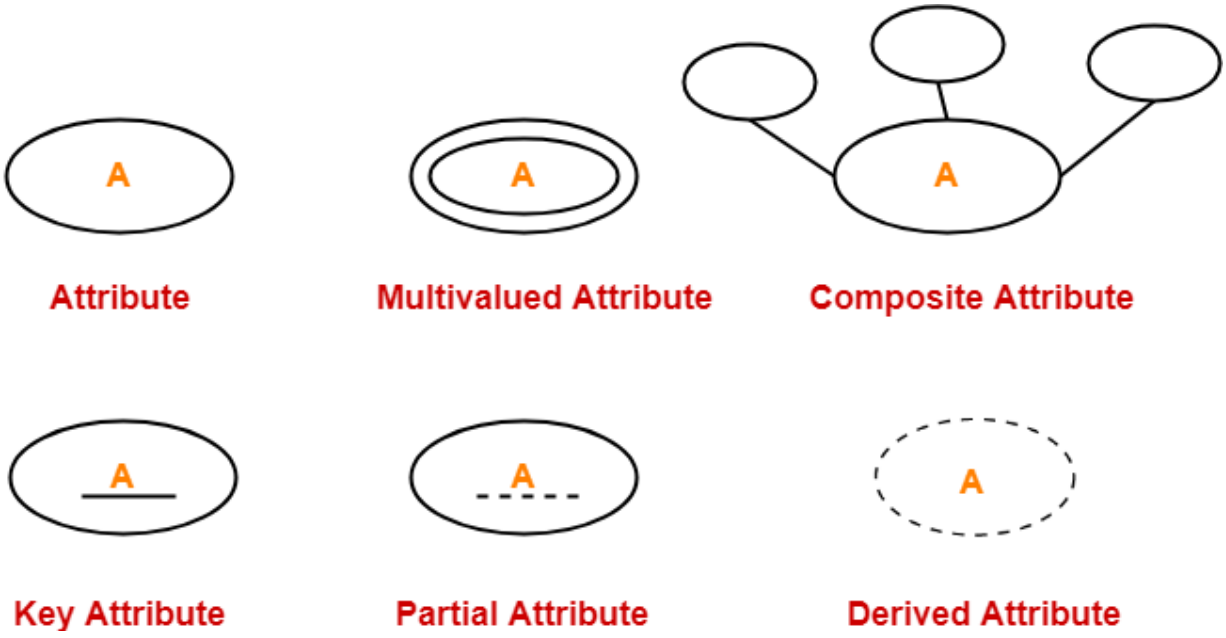
Differences between Strong entity set and Weak entity set

Strong entity set	Weak entity set
Symbol : 	Symbol : 
It contains sufficient attributes to form its primary key. A **diamond symbol** is used for the representation of the relationship that **exists between the two strong entity sets.**	It does not contain sufficient attributes to form its primary key. A **double diamond symbol** is used for the representation of the identifying relationship that **exists between the strong and weak entity set.**
A **single line** is used for the representation of the connection between the **strong entity set and the relationship.**	A **double line** is used for the representation of the **connection between the weak entity set and the relationship set.**
Total participation may or may not exist in the relationship.	**Total participation always exists** in the identifying relationship.

2. Attributes-

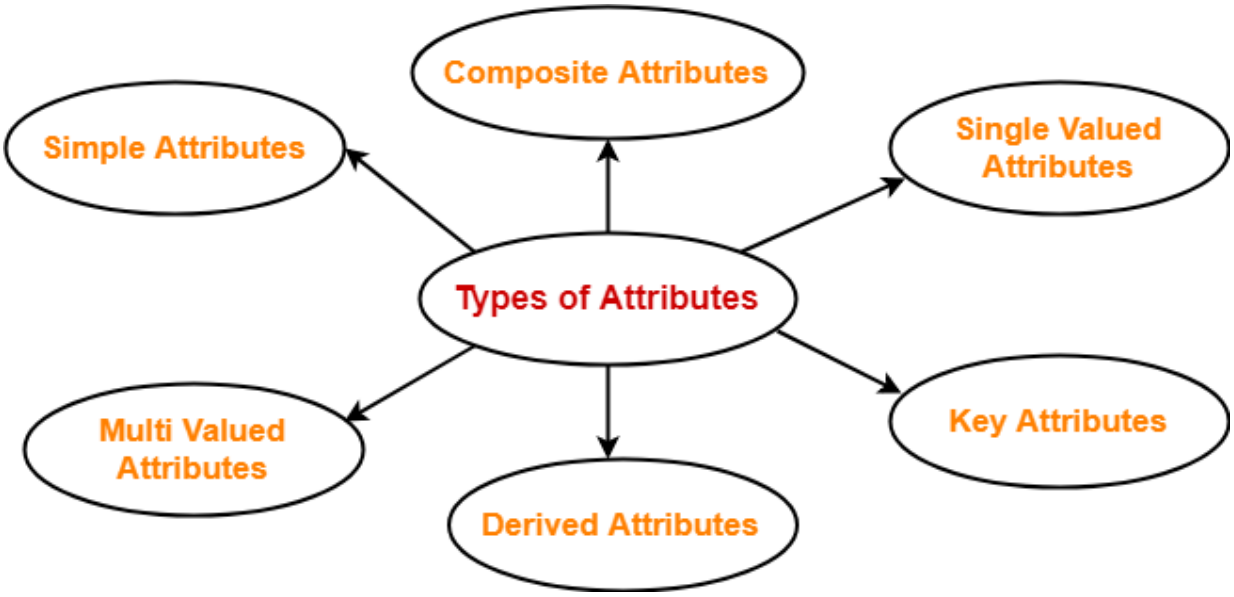
- Attributes are the **properties** which describes the entities of an entity set.
- There are several types of attributes.

1. Simple attributes
2. Composite attributes
3. Single valued attributes
4. Key attributes
5. Multi valued attributes
6. Derived attributes



- Attributes are the descriptive properties which are owned by each entity of an **Entity Set**.
- Each attribute exist a **specific domain** or set of values for each attribute from where the attribute can take its values.

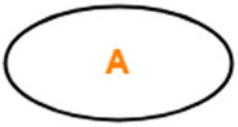
Types of Attributes-



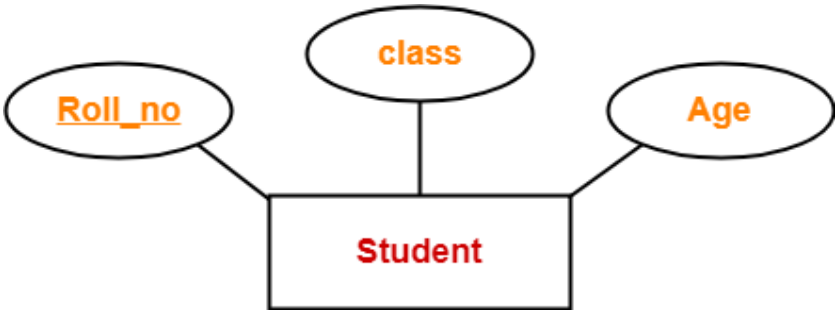
1. Simple Attributes-

Simple attributes are those attributes which **cannot be divided further**.

Symbol:



Example-

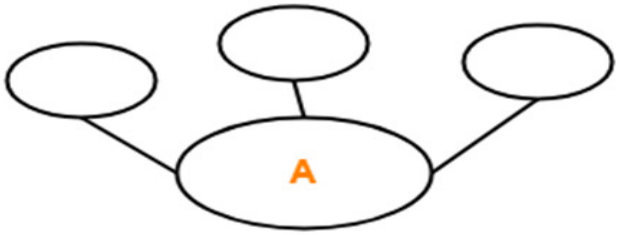


Here, all the attributes are simple attributes as they cannot be divided further.

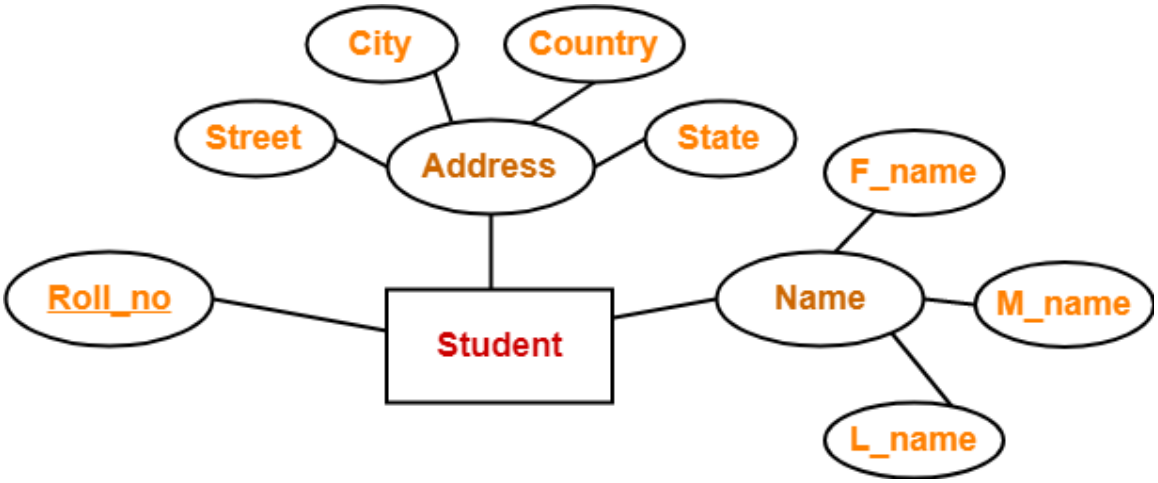
2. Composite Attributes-

- Composite attributes are those attributes which are **composed of many other simple attributes**.
- which can be **divided into subparts** to represent more basic attributes with its independent meaning.

Symbol-



Example-

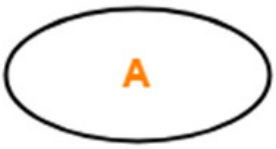


Here, the attributes **"Name" and "Address"** are composite attributes as they are composed of many other simple attributes.

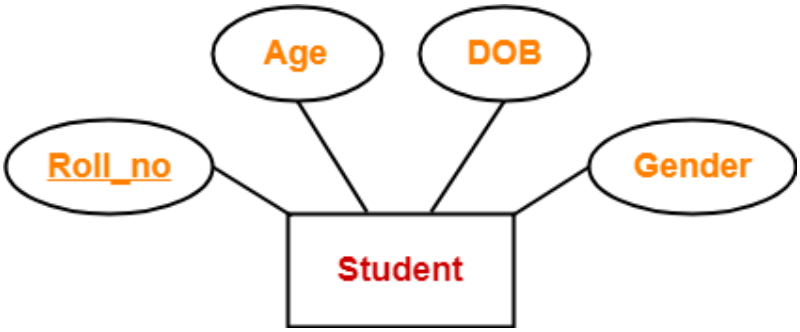
3. Single Valued Attributes-

Single valued attributes are those attributes which can take **only one value** for a given entity from an entity set.

symbol :



Example-



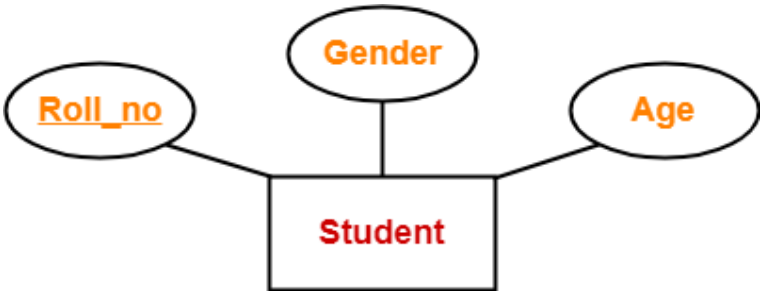
Here, all the attributes are single valued attributes as they can take **only one specific value** for each entity.

4. Key Attributes-

- Key attributes are those attributes which can **identify an entity uniquely in an entity set.**
- key attributes **values are distinct** for each individual entity in the entity set.
- It represented using name **underline inside** the oval.

Symbol-

Example-

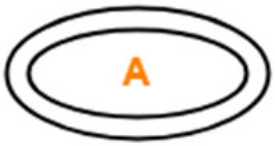


Here, the attribute "**Roll_no**" is a key attribute as it can identify any student uniquely.

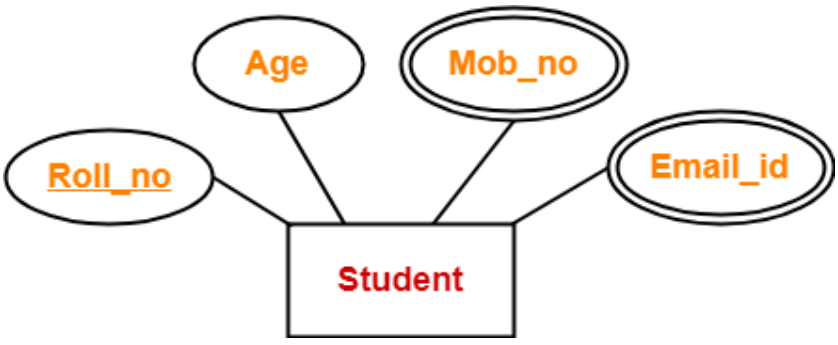
5. Multi Valued Attributes-

Multi valued attributes are those attributes which **can take more than one value** for a given entity from an entity set.

Symbol -



Example-



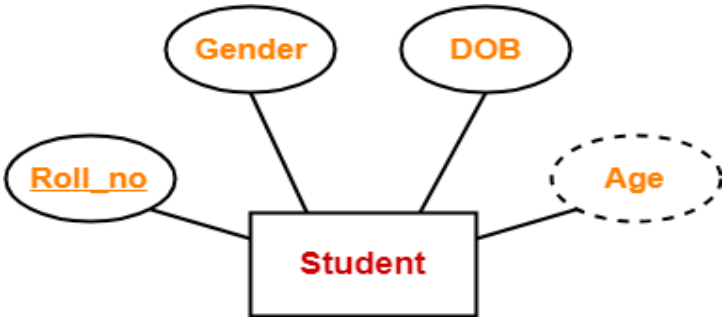
Here, the attributes "Mob_no" and "Email_id" are multi valued attributes as they can take more than one values for a given entity.

6. Derived Attributes-

Derived attributes are those attributes which can be **derived from other attribute**(s) or **calculated from stored attributes.**

Symbol:

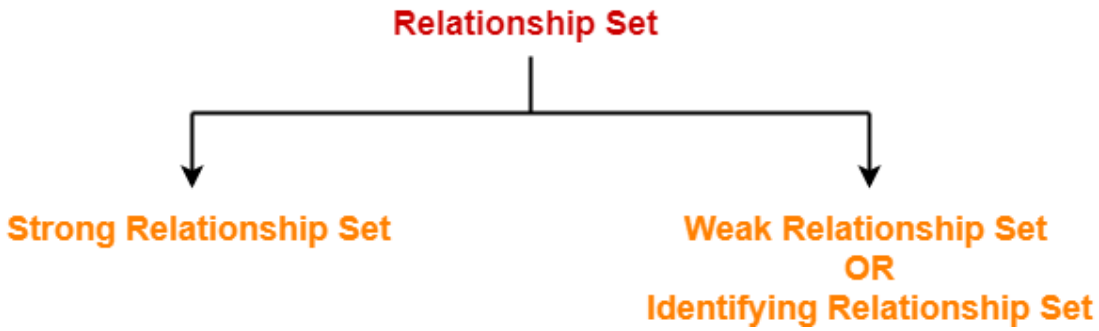
Example-



Here, the attribute "**Age**" is a **derived attribute** as it can be derived from the attribute "**DOB**" which is **stored attribute**.

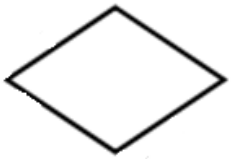
3. Relationship Sets-

- Relationship defines an association among several entities.
- A relationship set is a set of same type of relationships.
- A relationship set may be of the following two types-



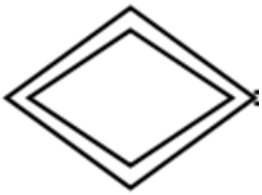
1. Strong Relationship Set-

- A strong relationship exists between two strong entity sets.
- It is represented using a diamond symbol.



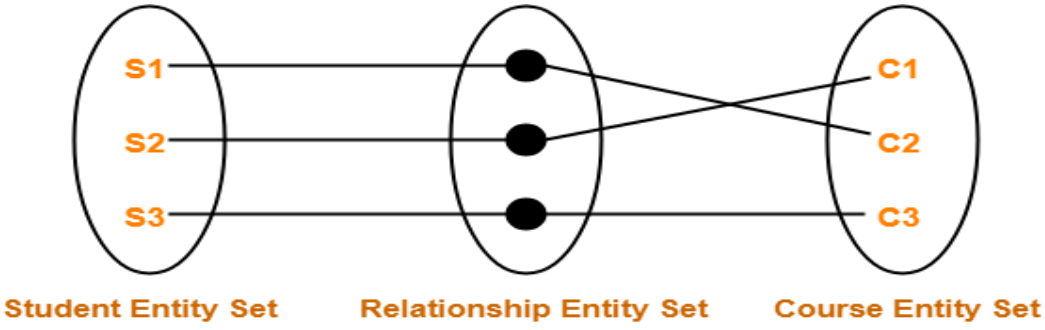
2. Weak Relationship Set-

- A weak or identifying relationship exists between the strong and weak entity set.
- It is represented using a double diamond symbol.



Example-

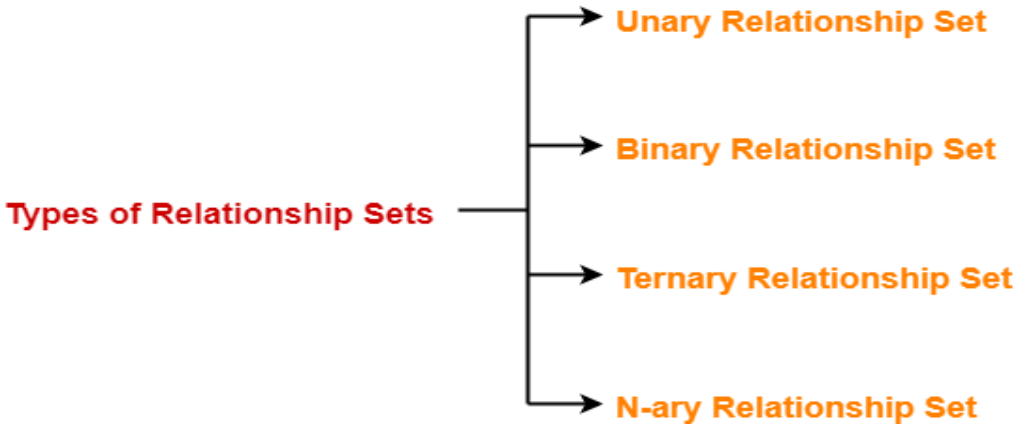
Set representation of above ER diagram is-



Set Representation of ER Diagram

❖ Types of Relationship Sets-

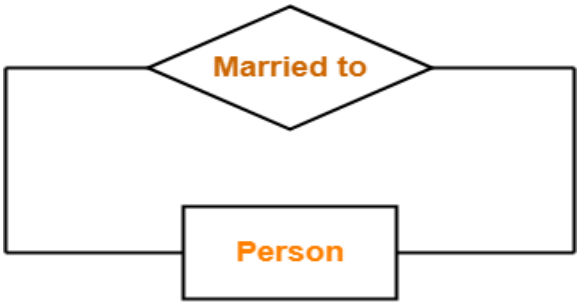
On the basis of degree of a relationship set, a relationship set can be classified into the following types-



1. Unary Relationship Set-

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

Example-



Unary Relationship Set

2. Binary Relationship Set-

- Binary relationship set is a relationship set where **two entity sets participate in** a relationship set.
- Student is enrolled in a Course

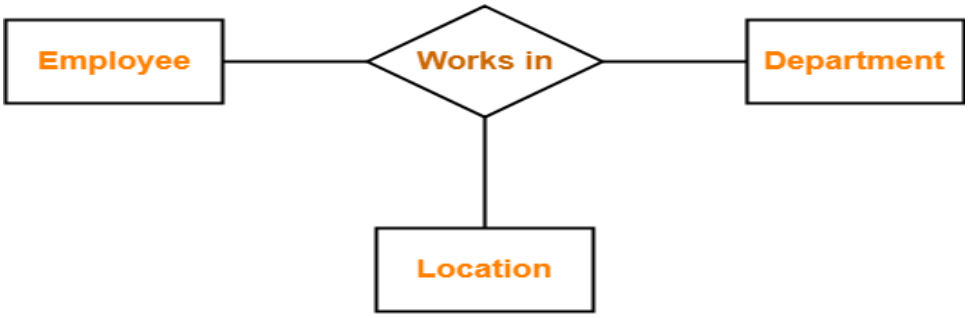
Example:

Binary Relationship Set

3. Ternary Relationship Set-

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

Example-



Ternary Relationship Set

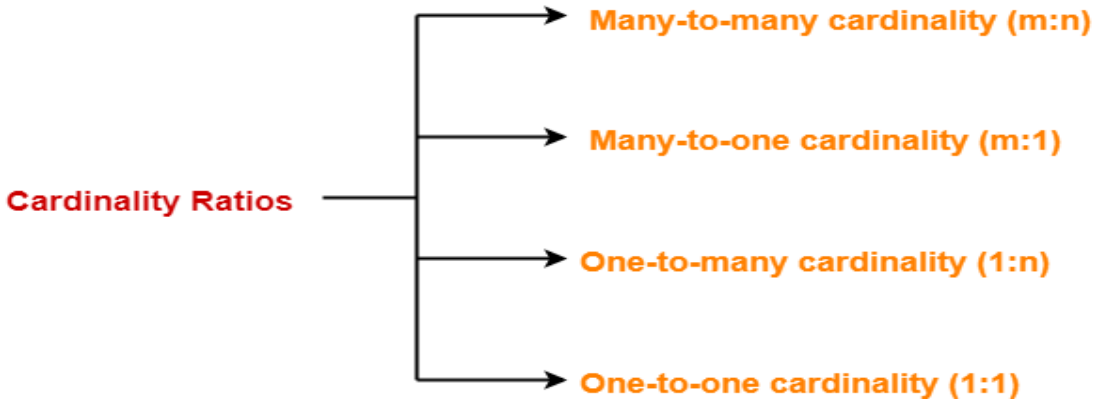
4. N-ary Relationship Set-

N-ary relationship set is a relationship set where **'n' entity sets participate** in a relationship set.

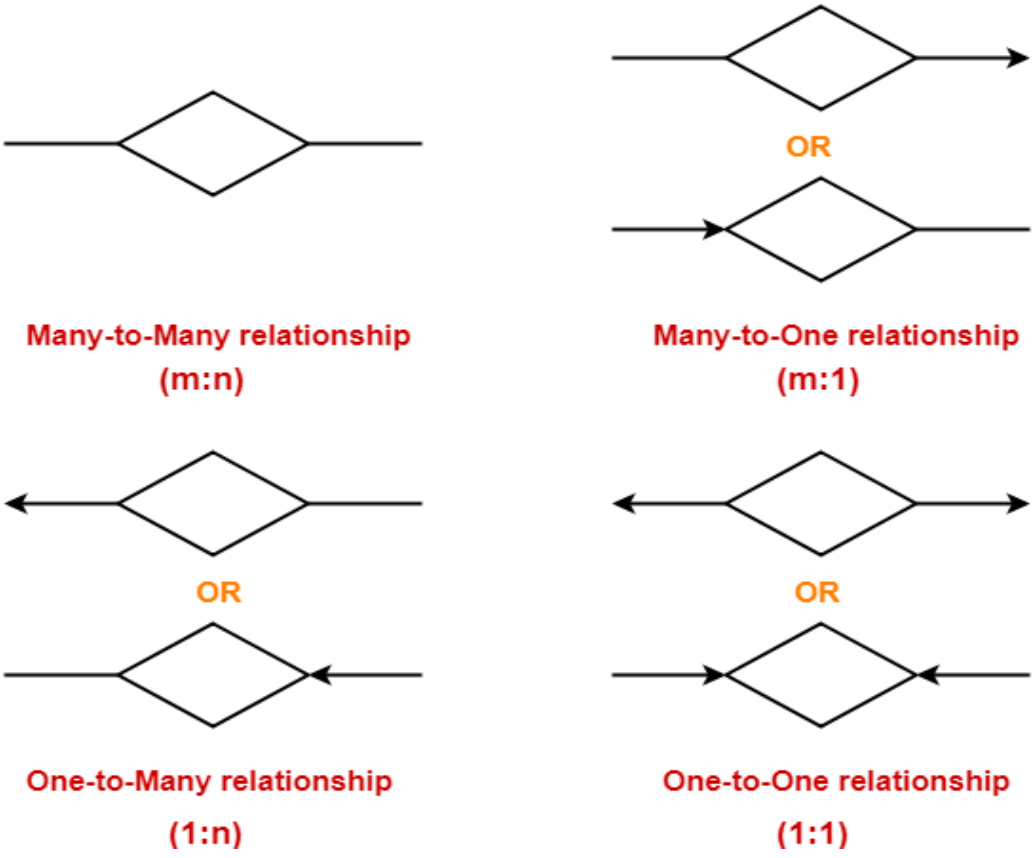
- **E/R Diagram : One to one, one to many , many to one, many to many**

Cardinality Constraint-

There are 4 types of cardinality ratios-



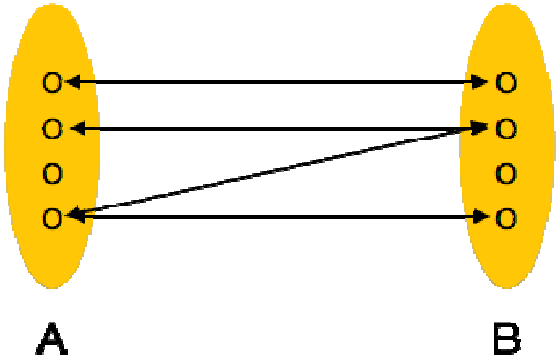
Symbol



1. Many-to-Many Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.



Symbol Used-

Cardinality Ratio = m : n

Example-

Consider the following ER diagram-

Many to Many Relationship

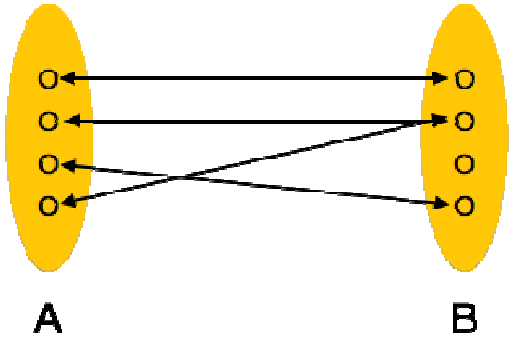
Here,

- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by any number (zero or more) of students.

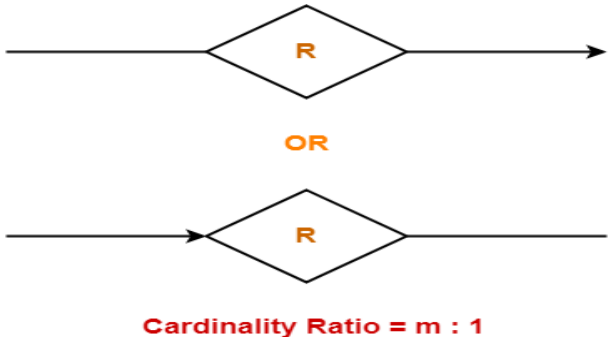
2. Many-to-One Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.



Symbol Used-



Example-

Consider the following ER diagram-

Many to One Relationship

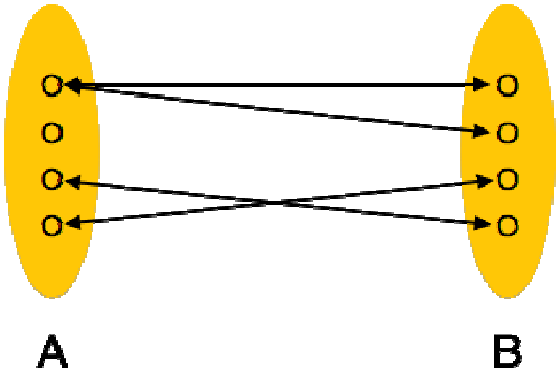
Here,

- One student can enroll in at most one course.
- One course can be enrolled by any number (zero or more) of students.

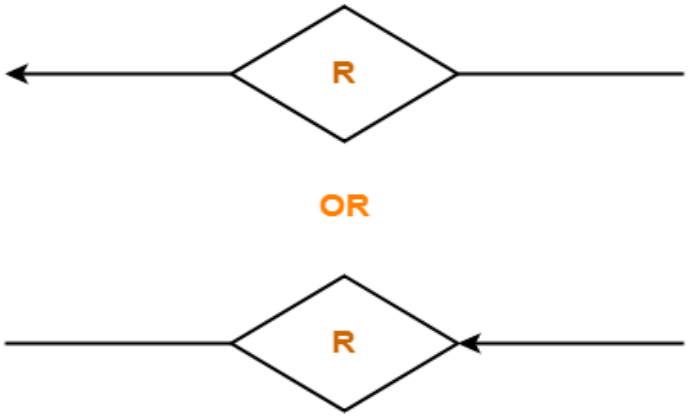
3. One-to-Many Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with at most one entity in set A.



Symbol Used-



Example-

Consider the following ER diagram-

One to Many Relationship

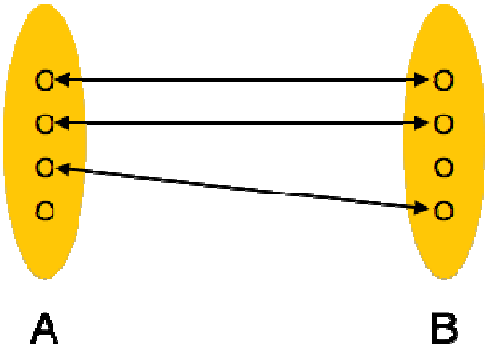
Here,

- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by at most one student.

4. One-to-One Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with at most one entity in set A.



Symbol Used-

OR

Cardinality Ratio = 1 : 1

Example-

Consider the following ER diagram-

One to One Relationship

Here,

- One student can enroll in at most one course.
- One course can be enrolled by at most one student.

❖3.3 Types of keys :

❖Database Keys

- Keys are very important part of **Relational database model.**
- They are used to **establish and identify relationships between tables** and also to **uniquely identify any record or row of data inside a table.**
- A Key can be a **single attribute or a group of attributes,** where the combination may act as a key.
- **Why we need a Key?**
- In real world applications, number of tables required for storing the data is huge, and the different tables are related to each other as well.
- Also, tables store a lot of data in them. **Tables generally extends** to thousands of records stored in them, **unsorted and unorganised.**
- Now to **fetch any particular record from such dataset**, you will have to apply some conditions, but what if there is duplicate data present and every time you try to fetch some data by applying certain condition, you get the **wrong data.**
- To avoid all this, Keys are defined to easily identify any row of data in a table.

3.3.1 Super key, candidate key, Primary key, Composite key, Foreign key, Unique key.

- **Types of keys:**

1. super key
2. candidate key
3. primary key

4. composite key
5. foreign key
6. unique key

Let's try to understand about all the keys using a simple example.

student_id	Name	phone	age
1	Akon	9876723452	17
2	Akon	9991165674	19
3	Bkon	7898756543	18
4	Ckon	8987867898	19
5	Dkon	9990080080	17

Let's take a simple **Student** table, with fields student_id, name, phone and age.

1. Super Key

- **Super Key** is defined as a set of attributes within a table that can **uniquely identify each record within a table.**
- Super Key is a **superset of Candidate key.**
- A super key is not restricted to have any specific number of attributes.
- Thus, a super key may consist of any number of attributes.
- **Example :**

Student (student_id , name , phone, age)
In the table defined above super key would include

- student_id
- (student_id, name)
- phone

- **So they all are super keys.**

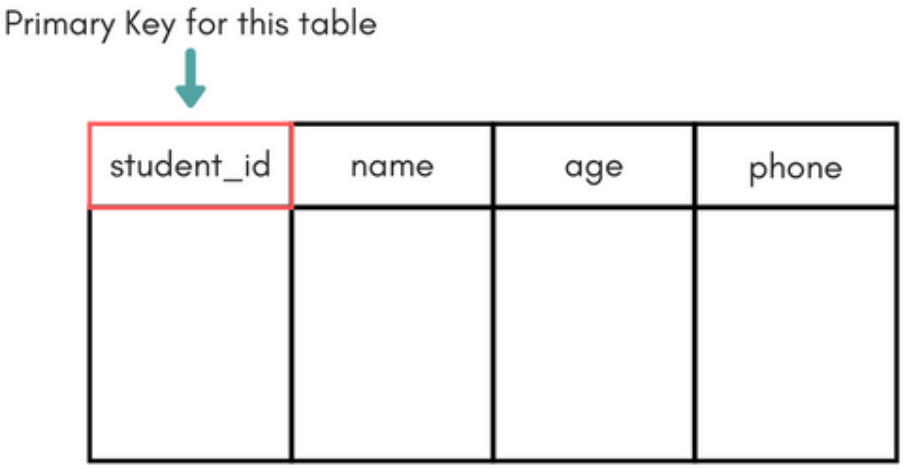
2. Candidate Key

- **A minimal super key** is called as a candidate key.
- A set of minimal attribute(s) that can identify each tuple/record uniquely in the given relation is called as a candidate key.

- It is an **attribute or a set of attributes that can act as a Primary Key** for a table to uniquely identify each record in that table. There can be **more than one candidate key.**
- In our example, student_id and phone both are **candidate keys** for table **Student**.
- A candidate key **can never be NULL or empty**. And its value should **be unique.**
- A candidate key can be a **combination of more than one columns(attributes).like (student_id,age)**
- Those attributes which appears in some candidate key are called as **prime attributes.**

3. Primary Key

- Primary key is a candidate key that is most appropriate to become the **main key for any table**.
- It is a key that can **uniquely** identify each record in a table.

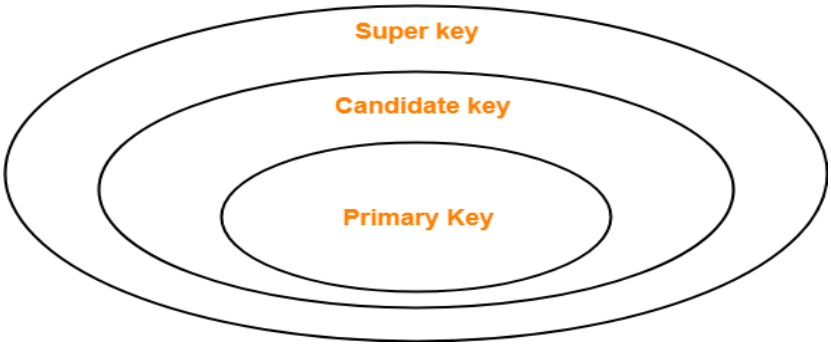


- For the table **Student** we can make the student_id column as the primary key.

NOTES-

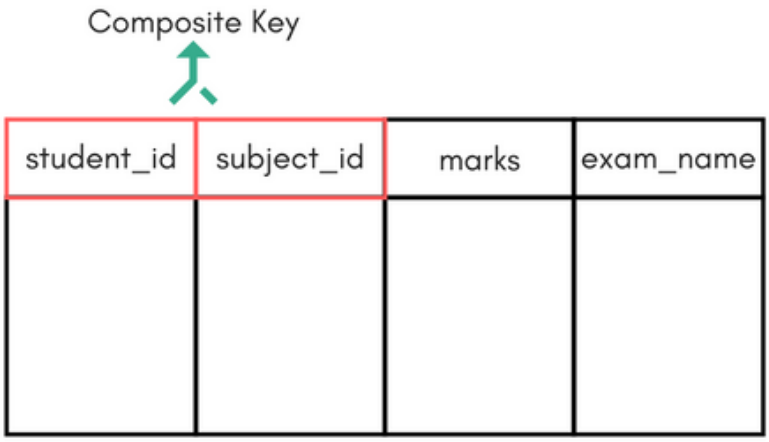
- The value of primary key can never be NULL.
- The value of primary key must always be unique.
- The values of primary key can never be changed i.e. no updation is possible.
- The value of primary key must be assigned when inserting a record.

- A relation is allowed to have only one primary key.



4. Composite Key

- A primary key comprising of multiple attributes and not just a single attribute is called as a composite key.
- Key that consists of **two or more attributes that uniquely identify** any record in a table is called **Composite key**. But the attributes which together form the **Composite key** are not a key independently or individually.

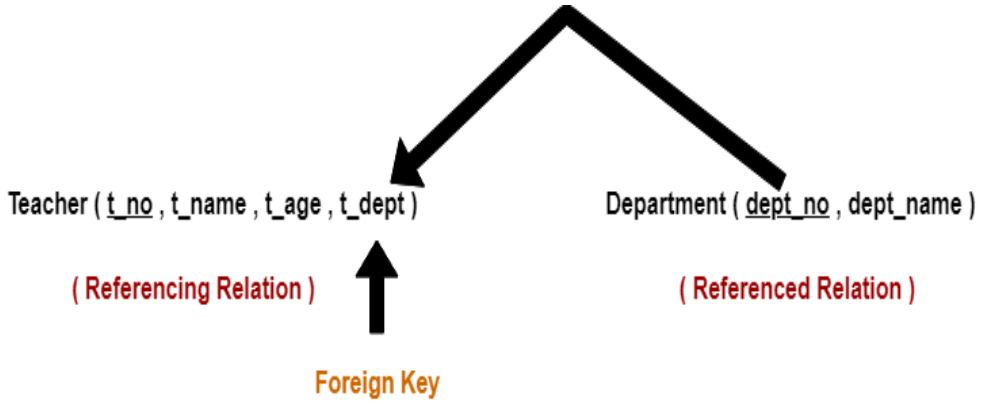


Score Table - To save scores of the student for various subjects.

- In the above picture we have a **Score** table which stores the marks scored by a student in a particular subject.
- In this table student_id and subject_id together will form the primary key, hence it is a composite key.

5. Foreign Key-

- An attribute 'X' is called as a foreign key to some other attribute 'Y' when its values are dependent on the values of attribute 'Y'.
- The attribute 'X' can assume only those values which are assumed by the attribute 'Y'.
- Here, the relation in which attribute 'Y' is present is called as the **referenced relation**.
- The relation in which attribute 'X' is present is called as the **referencing relation**.
- The attribute 'Y' might be present in the same table or in some other table.
- **Example-**
- Consider the following two schemas-



- Here, t_dept can take only those values which are present in dept_no in Department table since only those departments actually exist.

NOTES-

- Foreign key **references the primary key** of the table.
- Foreign key can take only those values which are present in the primary key of the referenced relation.
- Foreign key may have a name other than that of a primary key.
- Foreign key can take the NULL value.
- There is no restriction on a foreign key to be unique. In fact, foreign key is not unique most of the time.
- Referenced relation may also be called as the master table or primary table.

- Referencing relation may also be called as the foreign table.

6. Unique Key-

- Unique key is unique for all the records of the table.
- Once assigned, its value cannot be changed i.e. it is non-updatable.
- It may have a NULL value.

Example-

- The best example of unique key is **Adhaar Card Numbers.**
- The Adhaar Card Number is unique for all the citizens (tuples) of India (table).
- If it gets lost and another duplicate copy is issued, then the duplicate copy always has the same number as before.
- Thus, it is non-updatable.
- Few citizens may not have got their Adhaar cards, so for them its value is NULL.