CT-2:

Unit 2 topics:

a) What is the need for ER Model?

ER Diagram:

- Entity Relational model is a model for identifying entities to be represented in the database and representation of how those entities are related.
- The ER data model specifies enterprise schema that represents the overall logical structure of a database graphically.
- 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

i) E-R diagrams are used to represent E-R model in a database, which makes them easy to be converted into relations (tables).

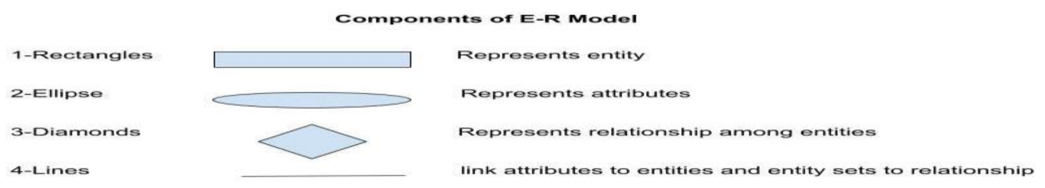
ii) E-R diagrams provide the purpose of real-world modeling of objects which makes them intently useful.

iii) E-R diagrams require no technical knowledge and no hardware support.

iv) These diagrams are very easy to understand and easy to create even by a naive user.

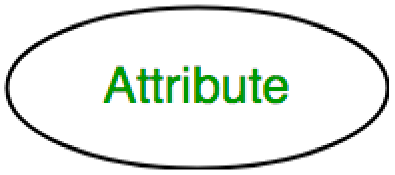
v) It gives a standard solution of visualizing the data logically.

ER Model is used to model the logical view of the system from a data perspective which consists of these components:

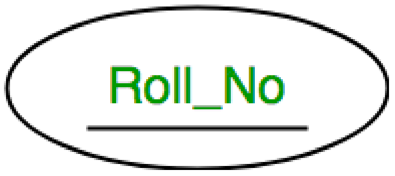


Keys, Attributes and Constraints:

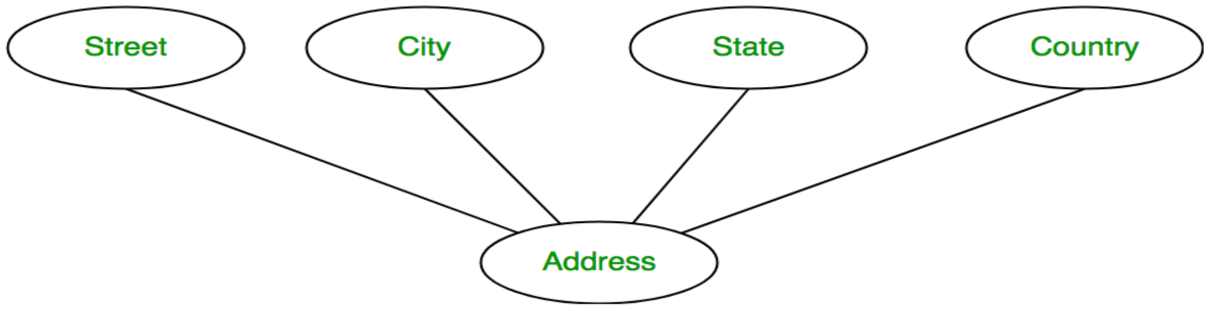
Attribute(s):
Attributes are the **properties that define the entity type**. For example, Roll No, Name, DOB, Age, Address, Mobile No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



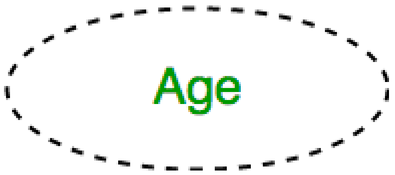
1. **Key Attribute –**
   The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.



2. **Composite Attribute –**
   An attribute **composed of many other attributes** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



3. **Derived Attribute –**
   An attribute that can be **derived from other attributes** of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval.

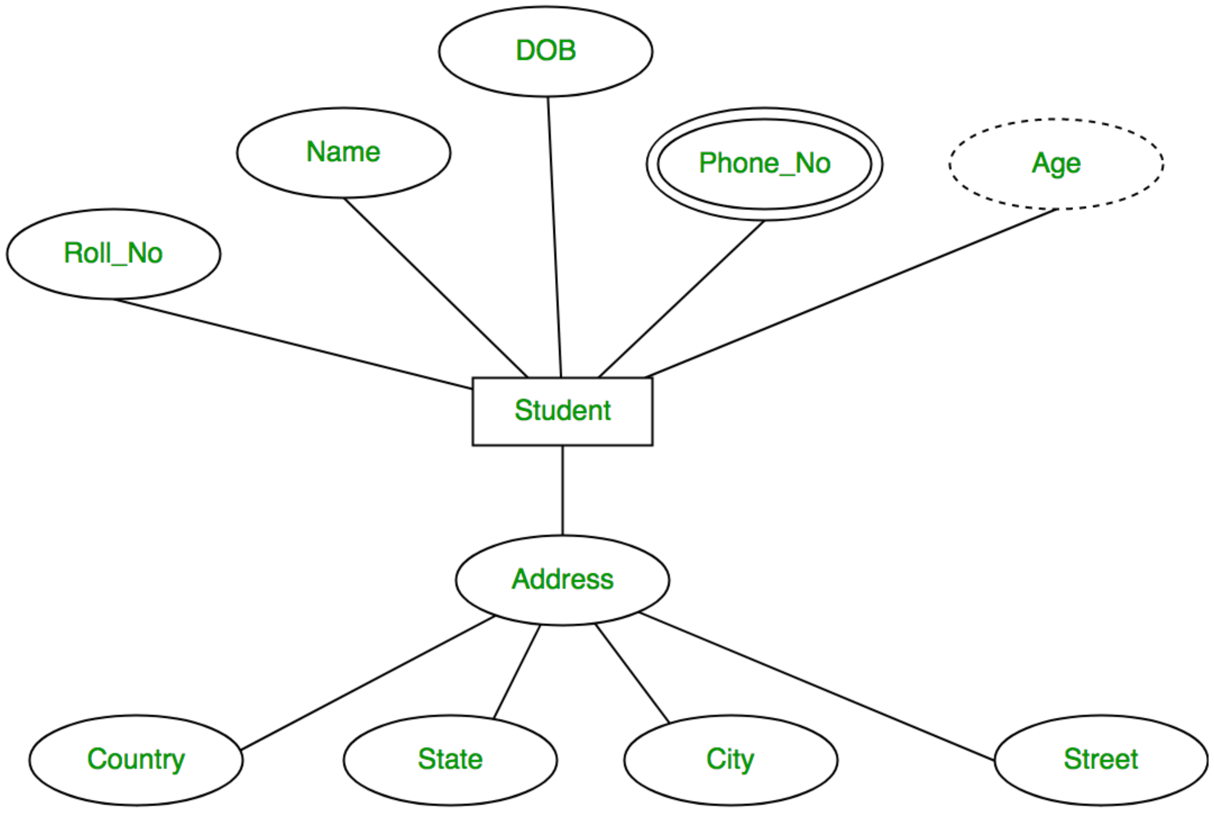


4.Multi-valued attribute Definition

A multivalued attribute is capable of storing more than one value in a single attribute i.e nothing but it can hold multiple values for the single attribute.

Example 1

In a Real-world scenario, a person can have more than one phone number hence "phone number attribute" will become a multivalued attribute.The complete entity type **Student** with its attributes can be represented as:



Difference between Strong and Weak Entity:

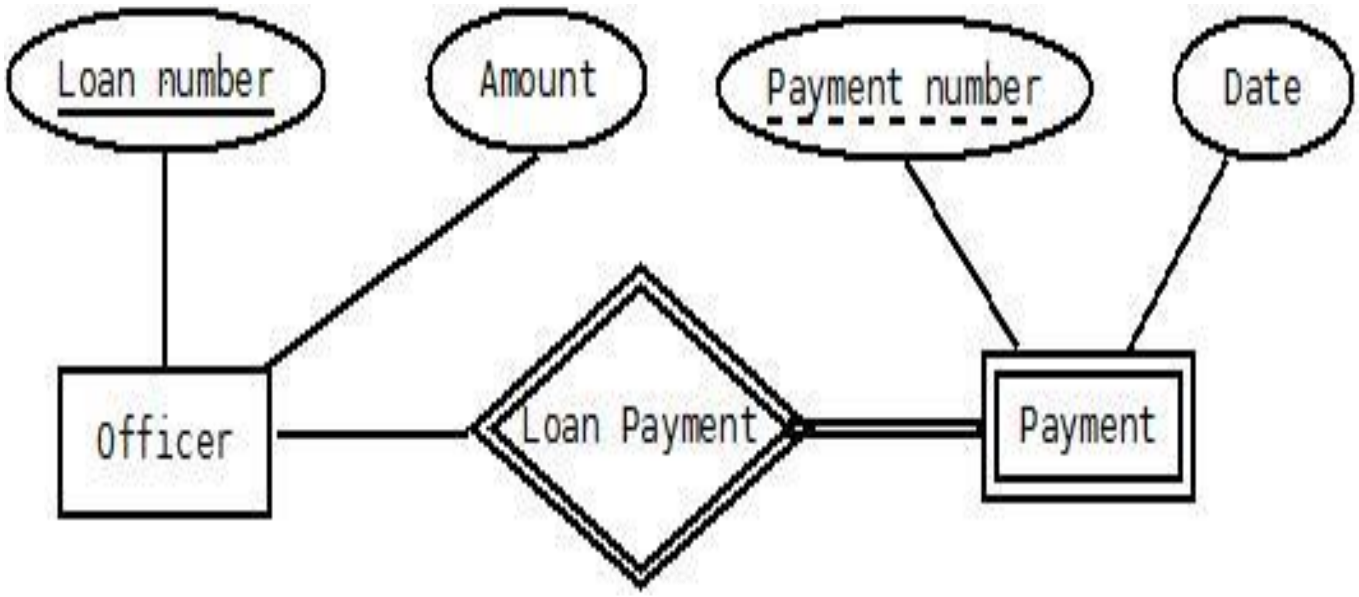
S.NO	Strong Entity	Weak Entity
1.	Strong entity always has a primary key.	While a weak entity has a partial discriminator key.
2.	Strong entity is not dependent on any other entity.	Weak entity depends on strong entity.
3.	Strong entity is represented by a single rectangle.	Weak entity is represented by a double rectangle.
4.	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.
5.	Strong entities have either total participation or not.	While weak entity always has total participation.

Draw an ER diagram to Illustrate weak entity set?

The entity sets which do not have sufficient attributes to form a primary key are known as **weak entity sets** and the entity sets which have a primary key are known as strong entity sets.

As the weak entities do not have any primary key, they cannot be identified on their own, so they depend on some other entity (known as owner entity). The weak entities have total participation constraint (existence dependency) in its identifying relationship with owner identity. Weak entity types have partial keys. Partial Keys are set of attributes with the help of which the tuples of the weak entities can be distinguished and identified. **Weak entities** are represented with **double rectangular** box in the ER Diagram.

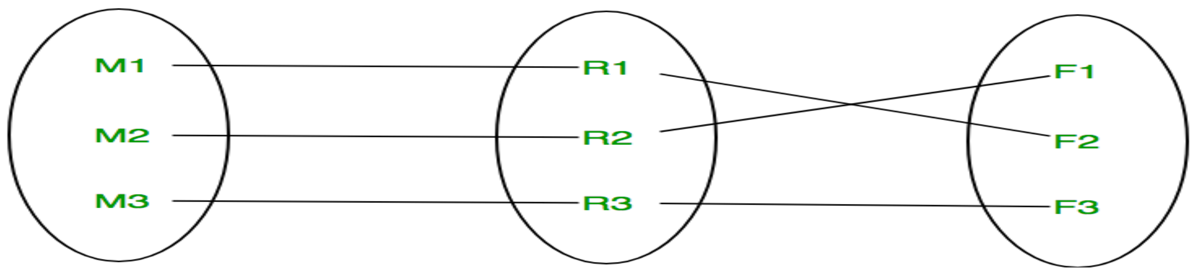
Example-1:
In the below ER Diagram, ‘Payment’ is the weak entity. ‘Loan Payment’ is the identifying relationship and ‘Payment Number’ is the partial key. Primary Key of the Loan along with the partial key would be used to identify the records.



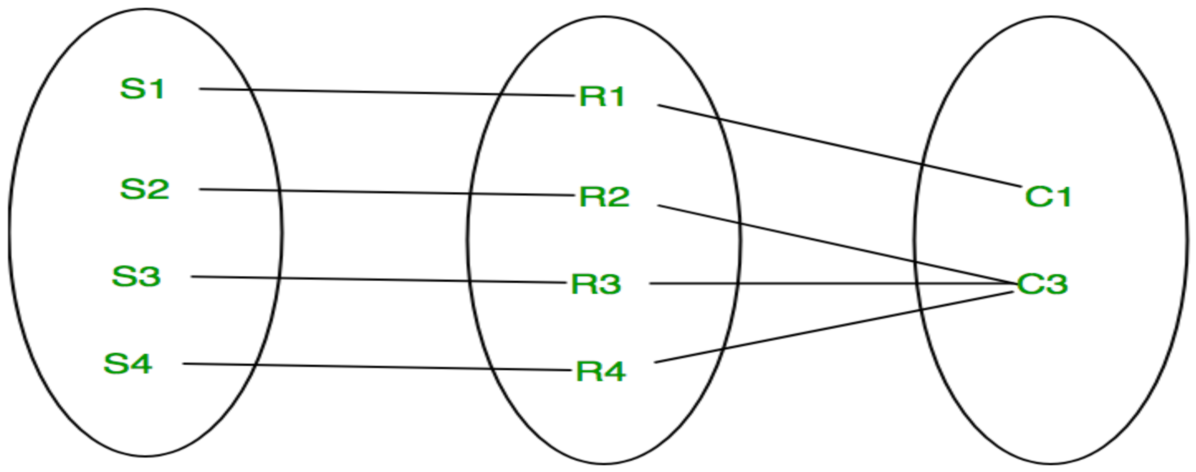
What is Mapping Cardinalities | ER Diagrams

Mapping cardinality is the maximum number of relationship instances in which an entity can participate. The **number of times an entity of an entity set participates in a relationship** set is known as cardinality. Cardinality can be of different types:

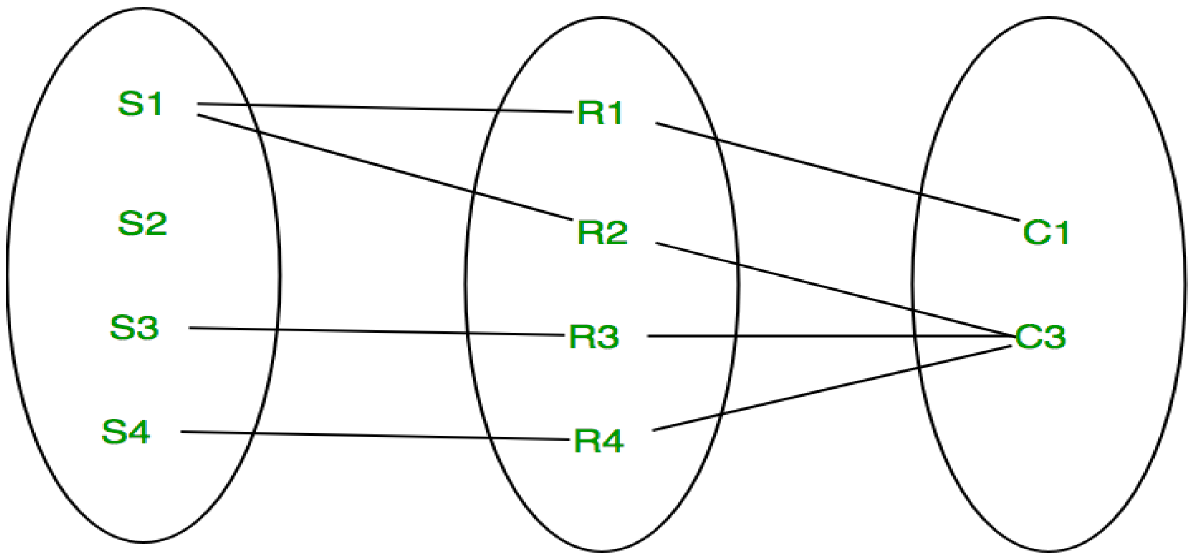
1. **One-to-one –** When each entity in each entity set can take part **only once in the relationship**, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So, the relationship will be one-to-one. the total number of tables that can be used in this is **2**.



2. Many to one – When entities in one entity set **can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set,** cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by many students. So the cardinality will be n to 1. It means that for one course there can be n students but for one student, there will be only one course.
The total number of tables that can be used in this is **3**



3. Many to many – When entities in all entity sets can **take part more than once in the relationship** cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.
the total number of tables that can be used in this is **3**
.



ER Design Issues

1) Use of Entity Set vs Attributes

The use of an entity set, or attribute depends on the structure of the real-world enterprise that is being modelled and the semantics associated with its attributes. It leads to a mistake when the user uses the primary key of an entity set as an attribute of another entity set. Instead, he should use the relationship to do so. Also, the primary key attributes are implicit in the relationship set, but we designate it in the relationship sets.

2) Use of Entity Set vs. Relationship Sets

It is difficult to examine if an object can be best expressed by an entity set or relationship set. To understand and determine the right use, the user need to designate a relationship set for describing an action that occurs in-between the entities. If there is a requirement of representing the object as a relationship set, then its better not to mix it with the entity set.

3) Use of Binary vs n-ary Relationship Sets

Generally, the relationships described in the databases are binary relationships. However, non-binary relationships can be represented by several binary relationships. For example, we can create and represent a ternary relationship 'parent' that may relate to a child, his father, as well as his mother. Such relationship can also be represented by two binary relationships i.e., mother and father, that may relate to their child. Thus, it is possible to represent a non-binary relationship by a set of distinct binary relationships.

4) Placing Relationship Attributes

The cardinality ratios can become an affective measure in the placement of the relationship attributes. So, it is better to associate the attributes of one-to-one or one-to-many relationship sets with any participating entity sets, instead of any relationship set. The decision of placing the specified attribute as a relationship or entity attribute should possess the characteristics of the real world enterprise that is being modelled.

Relational Model in DBMS

The relational model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation STUDENT with attributes ROLL_NO, NAME, ADDRESS, PHONE, and AGE shown in Table 1.

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI		18

Advantages:

- Simple model
- It is Flexible
- It is Secure
- Data accuracy
- Data integrity
- Operations can be applied easily

Disadvantage:

- Not good for large database
- Relation between tables become difficult some time

Characteristics of Relational Model:

- Data is represented into rows and columns called as relation.

- Data is stored in tables having relationship between them called Relational model.
- Relational model supports the operations like Data definition, Data manipulation, Transaction management.
- Each column have distinct name and they are representing attribute.
- Each row represents the single entity.

UNIT 3:

SQL | DDL, DQL, DML, DCL and TCL Commands

DQL (Data Query Language):

DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

It uses only one command:

SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

For example:

1. SELECT emp_name
2. FROM employee
3. WHERE age > 20.

TCL (Transaction Control Language):

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure

a. **Commit:** Commit command is used to save all the transactions to the database.

Example:

```
DELETE FROM CUSTOMERS
WHERE AGE = 25;
 COMMIT;
```

b. Rollback: Rollback command is used to undo transactions that have not already been saved to the database.

Example:

1. DELETE FROM CUSTOMERS
2. WHERE AGE = 25;
3. ROLLBACK;

c. SAVEPOINT: It is used to roll the transaction back to a certain point without rolling back the entire transaction.

Syntax:

1. SAVEPOINT SAVEPOINT_NAME;

Defining Constraints-Primary Key, Foreign Key, Unique, not null, check, IN operator

Constraints are the rules that we can apply on the type of data in a table. That is, we can specify the limit on the type of data that can be stored in a particular column in a table using constraints.

The available constraints in SQL are:

- **NOT NULL**: This constraint tells that we cannot store a null value in a column. That is, if a column is specified as NOT NULL then we will not be able to store null in this particular column any more.
- **UNIQUE**: This constraint when specified with a column, tells that all the values in the column must be unique. That is, the values in any row of a column must not be repeated.
- **PRIMARY KEY**: A primary key is a field which can uniquely identify each row in a table. And this constraint is used to specify a field in a table as primary key.

- **FOREIGN KEY**: A Foreign key is a field which can uniquely identify each row in a another table. And this constraint is used to specify a field as Foreign key.
- **CHECK**: This constraint helps to validate the values of a column to meet a particular condition. That is, it helps to ensure that the value stored in a column meets a specific condition.
- **DEFAULT**: This constraint specifies a default value for the column when no value is specified by the user.

Nested Queries in SQL

In nested queries, a query is written inside a query. The result of inner query is used in execution of outer query. We will use **STUDENT, COURSE, STUDENT_COURSE** tables for understanding nested queries.

Independent Nested Queries: In independent nested queries, query execution starts from innermost query to outermost queries. The execution of inner query is independent of outer query, but the result of inner query is used in execution of outer query. Various operators like IN, NOT IN, ANY, ALL etc are used in writing independent nested queries

Co-related Nested Queries: In co-related nested queries, the output of inner query depends on the row which is being currently executed in outer query.

- **Correlated Query –**
  In Correlated Query, Outer query executes first and for every Outer query row Inner query is executed. Hence, Inner query uses values from Outer query.
  Example –
- `Orders (OrderID, CustomerID, OrderDate);`

  `Customers (CustomerID, CustomerName, ContactName, Country);`

  Find details of customers who have ordered.

```
SELECT * FROM Customers where

EXISTS (SELECT CustomerID FROM Orders

WHERE Orders.CustomerID=Customers.CustomerID);
```

Subquery

When a query is included inside another query, the Outer query is known as Main Query, and Inner query is known as Subquery.

- To display NAME, LOCATION, PHONE_NUMBER of the students from DATABASE table whose section is A

```
Select NAME, LOCATION, PHONE_NUMBER from DATABASE

WHERE ROLL_NO IN

(SELECT ROLL_NO from STUDENT where SECTION=’A’);
```

NAME	ROLL_NO	LOCATION	PHONE_NUMBER
Ravi	104	Salem	8989898989
Raj	102	Coimbatore	8877665544

SQL | Triggers

Trigger is a statement that a system executes automatically when there is any modification to the database. In a trigger, we first specify when the trigger is to be executed and then the action to be performed when the trigger executes. Triggers are used to specify certain integrity constraints and referential constraints that cannot be specified using the constraint mechanism of SQL.

Example –

Suppose, we are adding a tuple to the ‘Donors’ table that is some person has donated blood. So, we can design a trigger that will automatically add the value of donated blood to the ‘Blood record’ table.

Types of Triggers –

We can define 6 types of triggers for each table:

1. **AFTER INSERT** activated after data is inserted into the table.

2. **AFTER UPDATE:** activated after data in the table is modified.

3. **AFTER DELETE:** activated after data is deleted/removed from the table.

4. **BEFORE INSERT:** activated before data is inserted into the table.

5. **BEFORE UPDATE:** activated before data in the table is modified.

6. **BEFORE DELETE:** activated before data is deleted/removed from the table.

What is Stored Procedures in SQL ?

Stored Procedures are created to perform one or more DML operations on Database. It is nothing but the group of SQL statements that accepts some input in the form of parameters and performs some task and may or may not returns a value.

Syntax: Creating a Procedure

```
CREATE or REPLACE PROCEDURE name(parameters)

IS

variables;

BEGIN

//statements;

END;
```

PL/SQL print largest number

```
DECLARE

        a NUMBER := 46;

        b NUMBER := 67;

        c NUMBER := 21;

BEGIN

        --block start

        --If condition start

        IF a > b

        AND a > c THEN
        --if a is greater then print a
        dbms_output.Put_line('Greatest number is '||a);
        ELSIF b > a
                AND b > c THEN
        --if b is greater then print b
        dbms_output.Put_line('Greatest number is '||b);
        ELSE
        --if c is greater then print c
        dbms_output.Put_line('Greatest number is '||c);
        END IF;
--end if condition
END;
--End program
```

what is entity type and entity set

Entity:

An entity is referred to as an object or thing that exists in the real world. For example, customer, car, pen, etc.

Entity Type:

A collection of entities with general characteristics is known as an entity type.

For example, a database of a corporate company has entity types such as employees, departments, etc. In DBMS, every entity type contains a set of attributes that explain the entity.

The Employee entity type can have attributes such as name, age, address, phone number, and salary.

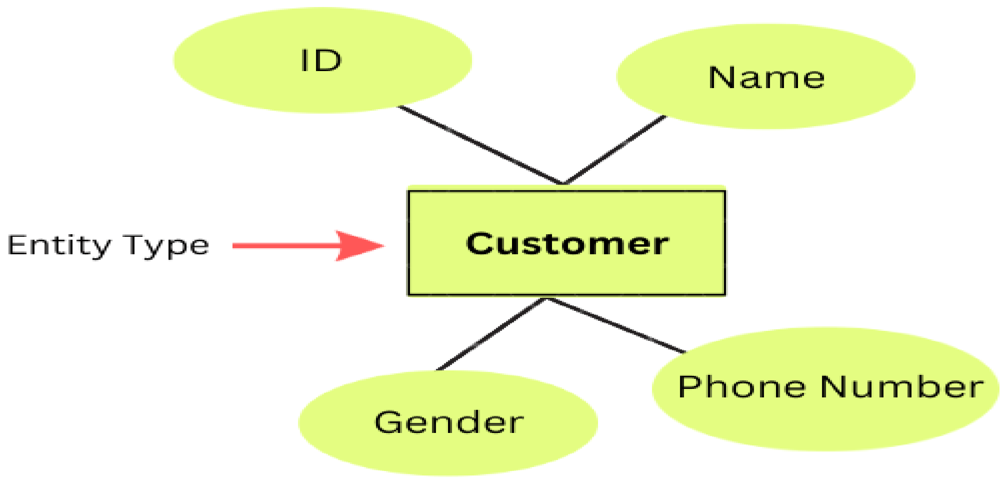
The Department entity type can have attributes such as name, number, and location in the department.

Kinds of Entity Type

There are two kinds of entity type, which are as follows:

1. Strong Entity Type: It is an entity that has its own existence and is independent.

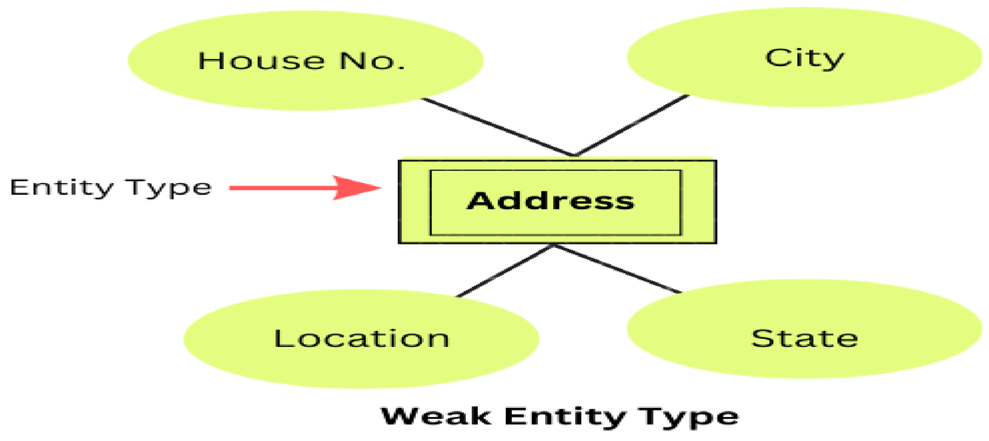
The entity relationship diagram represents a strong entity type with the help of a single rectangle. Below is the ERD of the strong entity type:



Strong Entity Type

2. Weak Entity Type: It is an entity that does not have its own existence and relies on a strong entity for its existence.

The Entity Relationship Diagram represents the weak entity type using double rectangles. Below is the ERD of the weak entity type:



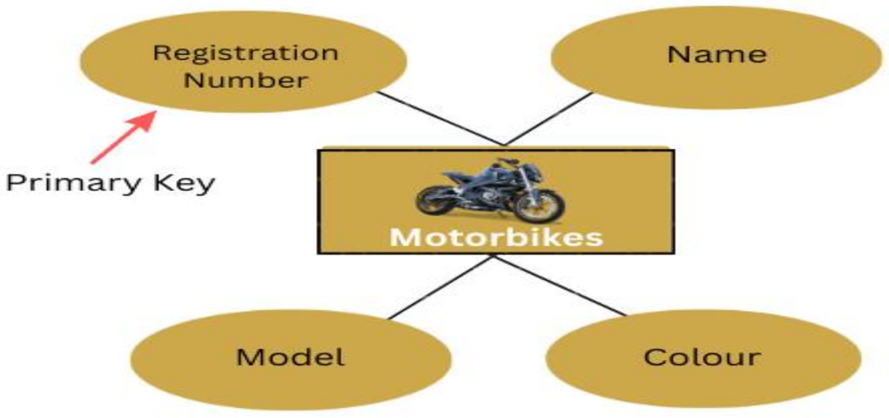
Entity Set

An entity set is a group of entities of the same entity type.

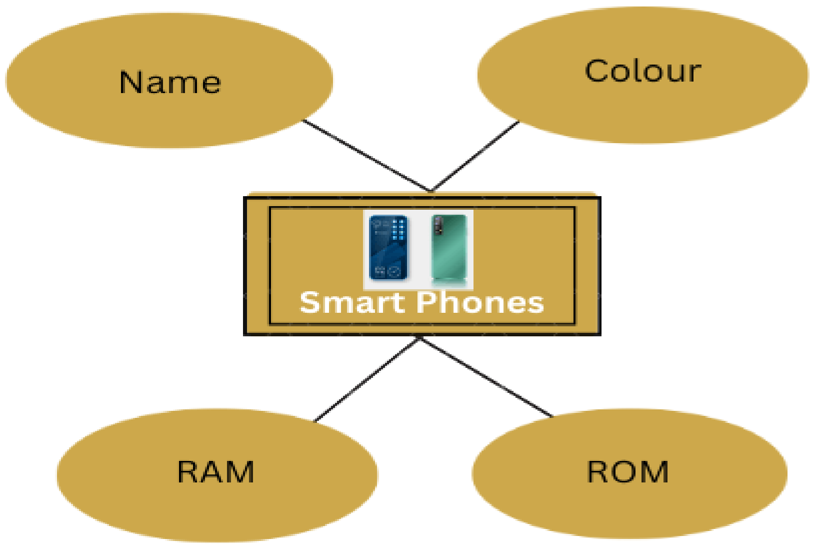
For example, an entity set of students, an entity set of motorbikes, an entity of smartphones, an entity of customers, etc.

Entity sets can be classified into two types:

The entity sets which do not have sufficient attributes to form a primary key are known as **weak entity sets** and the entity sets which have a primary key are known as strong entity sets.



ER Diagram of Strong Entity Set



ER Diagram of Weak Entity Set

Difference between Schema and Database

Let’s see the difference between Schema and Database:

S.NO	Schema	Database
1.	Schema may be a structural read of a info or database.	The info or database may be a assortment of reticulate knowledge.
2.	Schema once declared mustn’t be changed often.	Knowledge during a info or database keeps on change all time, therefore database or info modifies often.
3.	In schema, Tables name, fields name, its sorts as well as constraints are included.	Database or info includes such schema, records, constraints for the information.
4.	For a info, Schema is specified by DDL.	In a database, The operations such as updates and adds are done using DML.

Aggregate functions in SQL

In database management an aggregate function is a function where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning.

Various Aggregate Functions

```
1) Count()
2) Sum()
3) Avg()
4) Min()
5) Max()
```

Count():

Count(*): Returns total number of records .i.e 6.
Count(salary): Return number of Non Null values over the column salary. i.e 5.
Count(Distinct Salary): Return number of distinct Non Null values over the column salary .i.e 4

Sum():

sum(salary): Sum all Non Null values of Column salary i.e., 310
sum(Distinct salary): Sum of all distinct Non-Null values i.e., 250.

Avg():

Avg(salary) = Sum(salary) / count(salary) = 310/5
Avg(Distinct salary) = sum(Distinct salary) / Count(Distinct Salary) = 250/4

Min():

Min(salary): Minimum value in the salary column except NULL i.e., 40.
Max(salary): Maximum value in the salary i.e., 80.

Difference between Generalization and Specialization :

GENERALIZATION	**SPECIALIZATION**
Generalization works in Bottom-Up approach.	Specialization works in top-down approach.
In Generalization, size of schema gets reduced.	In Specialization, size of schema gets increased.
Generalization is normally applied to group of entities.	We can apply Specialization to a single entity.
Generalization can be defined as a process of creating groupings from various entity sets	Specialization can be defined as process of creating subgrouping within an entity set

GENERALIZATION	**SPECIALIZATION**
In Generalization process, what actually happens is that it takes the union of two or more lower-level entity sets to produce a higher-level entity sets.	Specialization is reverse of Generalization. Specialization is a process of taking a subset of a higher level entity set to form a lower-level entity set.
Generalization process starts with the number of entity sets and it creates high-level entity with the help of some common features.	Specialization process starts from a single entity set and it creates a different entity set by using some different features.
In Generalization, the difference and similarities between lower entities are ignored to form a higher entity.	In Specialization, a higher entity is split to form lower entities.
There is no inheritance in Generalization.	There is inheritance in Specialization.