**3. Database Design and E-R Model**

**3.1 Overview of Database Design**

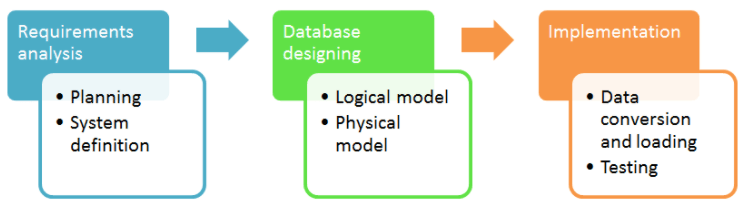
- Database design can be generally defined as a collection of tasks or processes that enhance the designing, development, implementation, and maintenance of enterprise data management system.

- Designing a proper database reduces the maintenance cost thereby improving data consistency and the cost-effective measures are greatly influenced in terms of disk storage space.

**Design Phases**

**Importance of Database Design**

1. Database designs provide the blueprints of how the data is going to be stored in a system.
2. A proper design of a database highly affects the overall performance of any application.
3. The designing principles defined for a database give a clear idea of the behaviour of any application and how the requests are processed.

****

**Requirement Analysis:**

**Planning** - This stage is concerned with planning the entire DDLC (Database Development Life Cycle).

**System definition** - This stage covers the boundaries and scopes of the proper database after planning.

### Database Designing:

**Physical Model** - The physical model is concerned with the practices and implementations of the logical model.

**Logical Model** - This stage is primarily concerned with developing a model based on the proposed requirements. The entire model is designed on paper without any implementation or adopting DBMS considerations.

### Implementation:

**Data conversion and loading** - This section is used to import and convert data from the old to the new system.

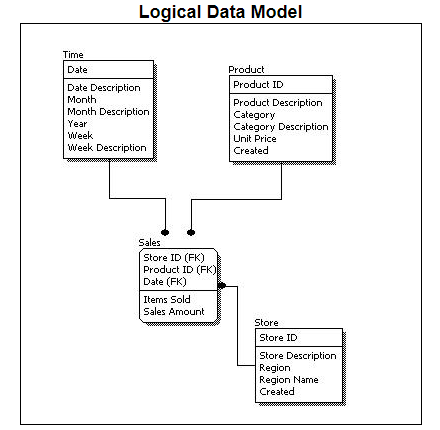
**Testing** - This stage is concerned with error identification in the newly implemented system. Testing is a crucial step because it checks the database directly and compares the requirement specifications.

### Logical

A logical data model generally describes the data in as many details as possible, without having to be concerned about the physical implementations in the database.

Features of logical data model might include:

1. All the entities and relationships amongst them.
2. Each entity has well-specified attributes.
3. The primary key for each entity is specified.
4. Foreign keys which are used to identify a relationship between different entities are specified.
5. Normalization occurs at this level.



### Physical

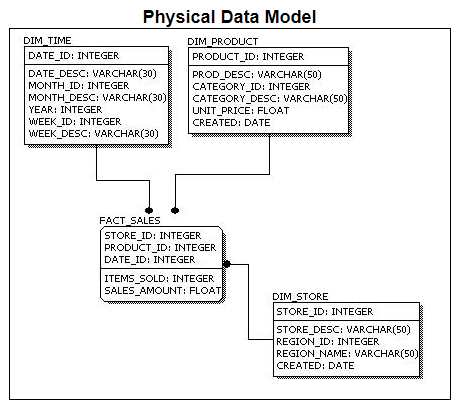
* A Physical data mode generally represents how the approach or concept of designing the database.
* The main purpose of the physical data model is to show all the **structures** of the table including the **column name, column data type, constraints, keys (primary and foreign)**, and the relationship among tables.

The following are the features of a physical data model:

1. Specifies all the columns and tables.
2. Specifies foreign keys that usually define the relationship between tables.
3. Based on user requirements, de-normalization might occur.
4. Since the physical consideration is taken into account so there will straightforward reasons for difference than a logical model.
5. Physical models might be different for different RDBMS. For example, the data type column may be different in MySQL and SQL Server.

While designing a physical data model, the following points should be taken into consideration:

1. Convert the entities into tables.
2. Convert the defined relationships into foreign keys.
3. Convert the data attributes into columns.
4. Modify the data model constraints based on physical requirements.



**3.2 The Entity-Relationship Model**

**a. Entity Sets –**

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

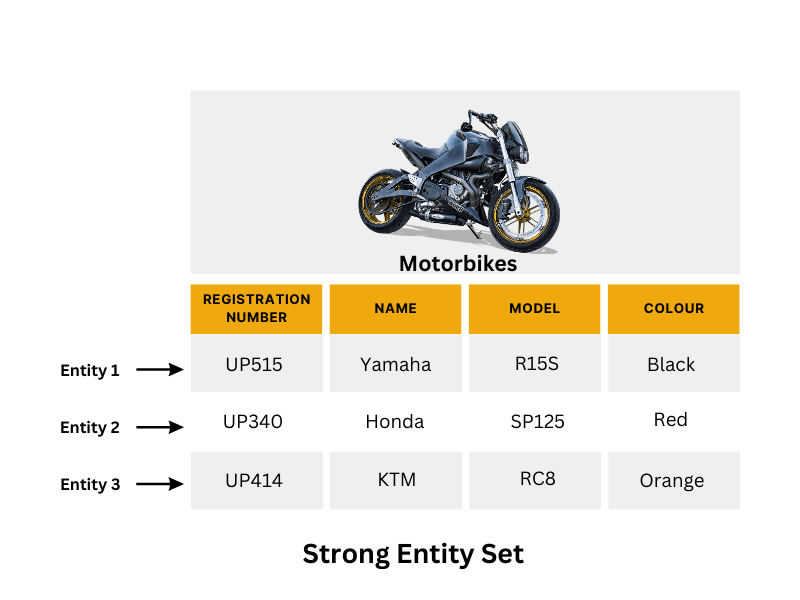
For example, an entity set of students, an entity set of motorbikes, an entity of smartphones, an entity of customers, etc. Entity sets can be classified into two types:

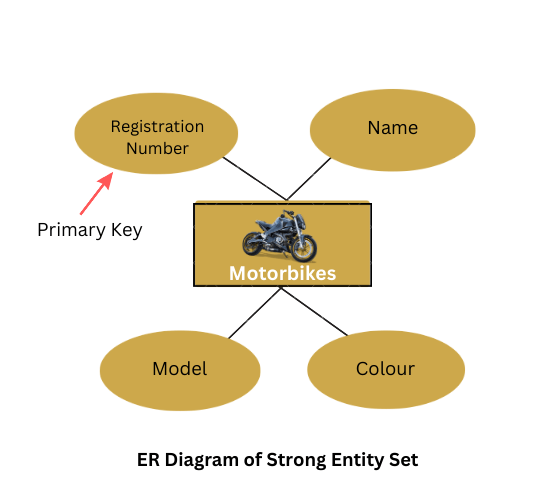
**1. Strong Entity Set:**

In a DBMS, a strong entity set consists of a primary key.

For example, an entity of motorbikes with the attributes, motorbike's registration number, motorbike's name, motorbike's model, and motorbike's colour.

Below is the representation of a strong entity set in tabular form:



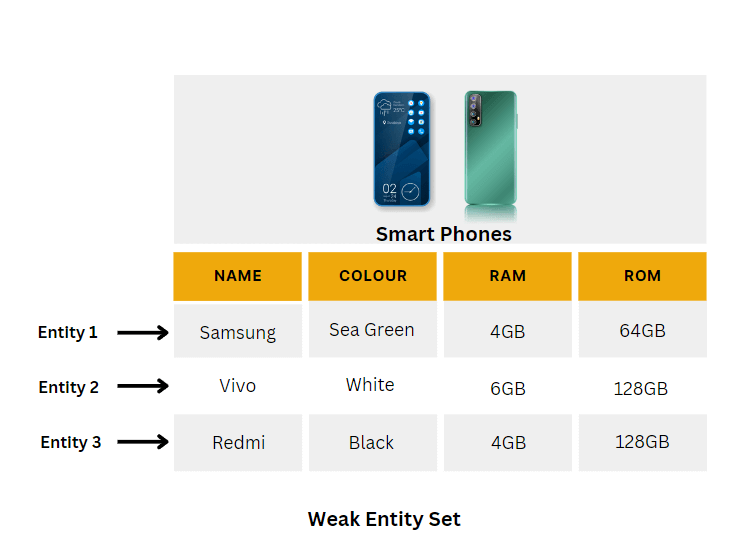


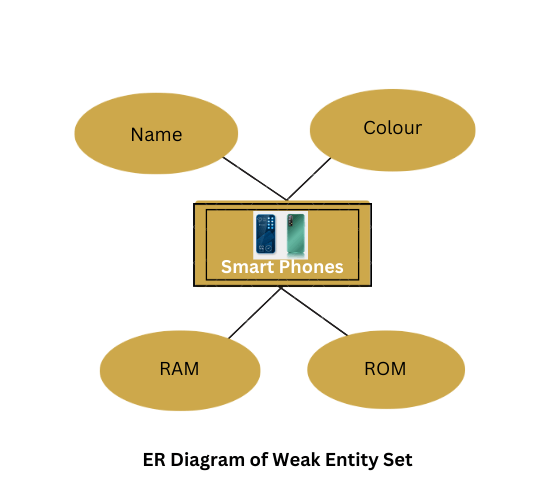
**2. Weak Entity Set:**

In a DBMS, a weak entity set does not contain a primary key.

For example, An entity of smartphones with its attributes, phone's name, phone's colour, and phone's RAM.

Below is the representation of a weak entity set in tabular form:





**b. Relationship Sets:**

|  |
| --- |
| A relationship is defined as an association among several entities. |

## **Example -**

‘Enrolled in’ is a relationship that exists between entities **Student** and **Course**.

A relationship set is a set of relationships of same type.

Set representation of above ER diagram is-

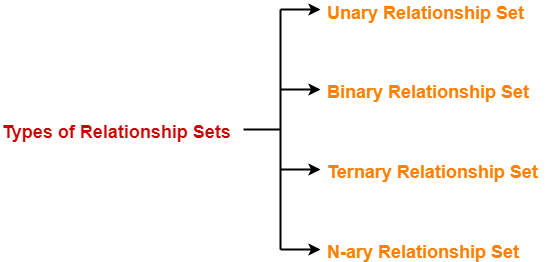
## **Degree of a Relationship Set-**

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

|  |
| --- |
| **Degree of a relationship set = Number of entity sets participating in a relationship set** |

## **Types of Relationship Sets -**

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



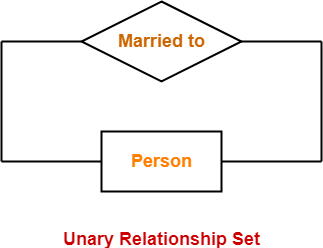
1. Unary relationship set
2. Binary relationship set
3. Ternary relationship set
4. N-ary relationship set

## **1. Unary Relationship Set-**

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

### ****Example-****

One person is married to only one person

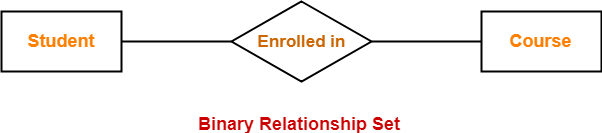


## **2. Binary Relationship Set-**

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

### **Example-**

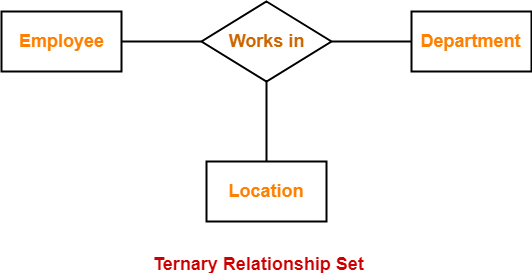
Student is enrolled in a Course



## **3. Ternary Relationship Set-**

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

### ****Example-****



## **4. N-ary Relationship Set-**

N-ary relationship set is a relationship set where ‘n’ entity sets participate in a relationship set.

**c. Attributes:**

* An **attribute** is a property or characteristic of an entity. An entity may contain any number of attributes.
* One of the attributes is considered as the primary key.
* In an Entity-Relation model, attributes are represented in an elliptical shape.

### ****Single-valued attribute:****

The attribute which takes up only a single value for each entity instance is a single-valued attribute.

**Example:** The age of a student.

### ****Multi-valued attribute:****

The attribute which takes up more than a single value for each entity instance is a multi-valued attribute.

**Example:**Phone number of a student: Landline and mobile.

### ****Derived attribute:****

An attribute that can be derived from other attributes is derived attributes.

**Example:** Total and average marks of a student.

### ****Complex attribute:****

Those attributes, which can be formed by the nesting of composite and multi-valued attributes, are called “***Complex Attributes***“. These attributes are rarely used in DBMS(DataBase Management System). That’s why they are not so popular.

### Stored attribute:

 The stored attribute are those attribute which doesn’t require any type of further update since they are stored in the database.

**Example:**DOB(Date of birth) is the stored attribute.

### Key attribute:

Key attributes are those attributes that can uniquely identify the entity in the entity set.

**Example:** Roll-No is the key attribute because it can uniquely identify the student.

Complex attributes are the nesting of two or more composite and multi-valued attributes. Therefore, these multi-valued and composite attributes are called ‘Components’ of complex attributes.

            These components are grouped between parentheses ‘( )’ and multi-valued attributes between curly braces ‘{ }’, Components are separated by commas ‘, ‘.

For example**:** let us consider a person having multiple phone numbers, emails, and an address.

Here, phone number and email are examples of multi-valued attributes and address is an example of the composite attribute, because it can be divided into house number, street, city, and state.

### Complex attributes

# **Constraints in SQL**

Constraints in SQL means we are applying certain conditions or restrictions on the database.

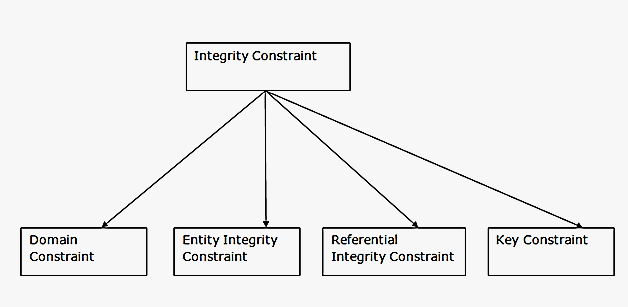
### Types:

1. **Column Level Constraint:**  
   Column Level Constraint is used to apply a constraint on a single column.
2. **Table Level Constraint:**  
   Table Level Constraint is used to apply a constraint on multiple columns.

# **Integrity Constraints**

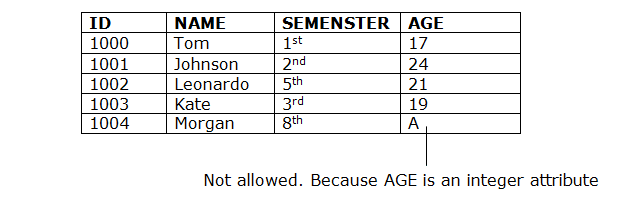
* Integrity constraints are a set of rules. It is used to maintain the quality of information.
* Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
* Thus, integrity constraint is used to guard against accidental damage to the database.

## **Types of Integrity Constraint**



### 1. Domain constraints

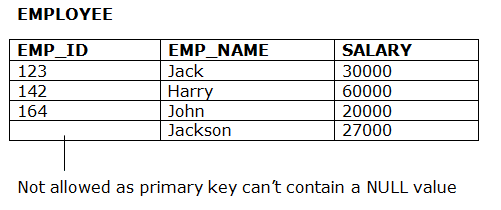
* Domain constraints can be defined as the definition of a valid set of values for an attribute.
* The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

**Example:** 

### 2. Entity integrity constraints

* The entity integrity constraint states that primary key value can't be null.
* This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
* A table can contain a null value other than the primary key field.

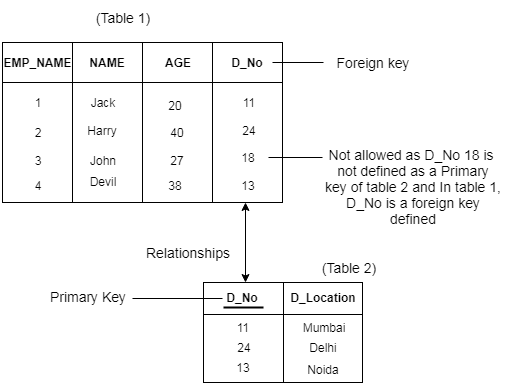
**Example:**



### 3. Referential Integrity Constraints

* A referential integrity constraint is specified between two tables.
* In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

**Example:**



### 4. Key constraints

* Keys are the entity set that is used to identify an entity within its entity set uniquely.
* An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

**Example:**

### DBMS Integrity Constraints

### Constraints available in SQL are:

1. NOT NULL
2. UNIQUE
3. PRIMARY KEY
4. FOREIGN KEY
5. CHECK

### Cardinalities:

### In a database, the mapping cardinality or cardinality ratio means to denote the number of entities to which another entity can be linked through a certain relation set.

### Mapping cardinality is most useful in describing binary relation sets, although they can contribute to the description of relation sets containing more than two entity sets. H

### ere, we will focus only on binary relation sets means we will find the relation between entity sets A and B for the set R. So we can map any one of following the cardinality:

### 1. One-to-one: In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say that a unit or item in B is connected to at most one unit or item in A.

### https://media.geeksforgeeks.org/wp-content/uploads/20220131223352/121.jpg

### Example: In a particular hospital, the surgeon department has one head of department. They both serve one-to-one relationships.

### https://media.geeksforgeeks.org/wp-content/uploads/20220208104357/onetoone.jpg

### 2. One-to-many: In this type of cardinality mapping, an entity in A is associated with any number of entities in B. Or we can say that one unit or item in B can be connected to at most one unit or item in A.

### https://media.geeksforgeeks.org/wp-content/uploads/20220131231234/12n.jpg

### Example: In a particular hospital, the surgeon department has multiple doctors. They serve one-to-many relationships.

### https://media.geeksforgeeks.org/wp-content/uploads/20220208104356/onetomany.jpg

### 3. Many-to-one: In this type of cardinality mapping, an entity in A is connected to at most one entity in B. Or we can say a unit or item in B can be associated with any number (zero or more) of entities or items in A.

### https://media.geeksforgeeks.org/wp-content/uploads/20220131232355/m21.jpg

### Example:In a particular hospital, multiple surgeries are done by a single surgeon. Such a type of relationship is known as a many-to-one relationship.

### https://media.geeksforgeeks.org/wp-content/uploads/20220208104355/manytoone.jpg

### 4. Many-to-many:  In this type of cardinality mapping, an entity in A is associated with any number of entities in B, and an entity in B is associated with any number of entities in A.

### https://media.geeksforgeeks.org/wp-content/uploads/20220208105038/manytomantrelationship.jpg

### In a particular company, multiple people work on multiple projects. They serve many-to-many relationships.

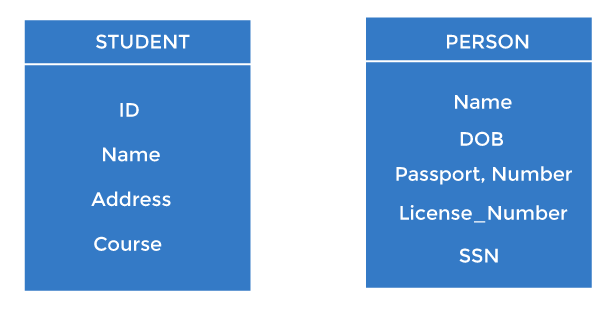
### https://media.geeksforgeeks.org/wp-content/uploads/20220208104354/manytomany.jpg

### Keys:

# **Keys**

* Keys play an important role in the relational database.
* It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.

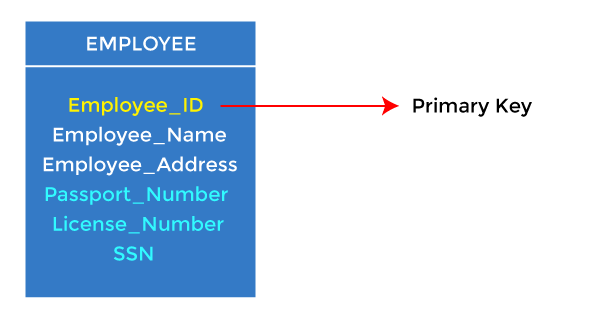
**For example,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport\_number, license\_number, SSN are keys since they are unique for each person.



## **Types of keys:**

### 1. Primary key

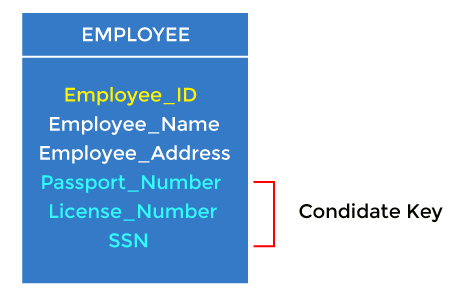
* It is the first key used to identify one and only one record uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
* In the EMPLOYEE table, ID can be the primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License\_Number and Passport\_Number as primary keys since they are also unique.
* For each entity, the primary key selection is based on requirements and developers.



### 2. Candidate key

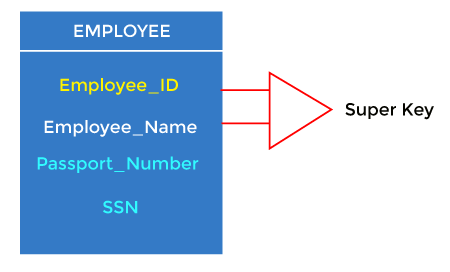
* A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
* Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

**For example:** In the EMPLOYEE table, id is best suited for the primary key. The rest of the attributes, like SSN, Passport\_Number, License\_Number, etc., are considered a candidate key.



### 3. Super Key

Super key is an attribute set that can uniquely identify a tuple. A super key is a superset of a candidate key.

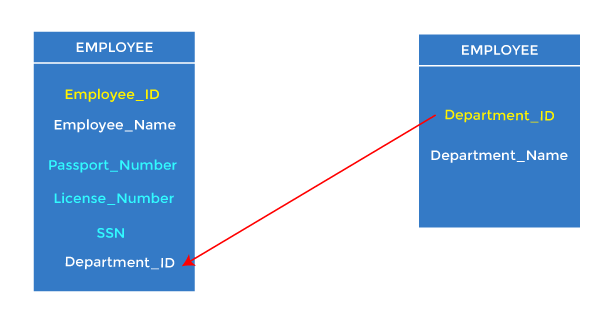


**For example:** In the above EMPLOYEE table, for(EMPLOEE\_ID, EMPLOYEE\_NAME), the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

The super key would be EMPLOYEE-ID (EMPLOYEE\_ID, EMPLOYEE-NAME), etc.

### 4. Foreign key

* Foreign keys are the column of the table used to point to the primary key of another table.
* Every employee works in a specific department in a company, and employee and department are two different entities. So we can't store the department's information in the employee table. That's why we link these two tables through the primary key of one table.
* We add the primary key of the DEPARTMENT table, Department\_Id, as a new attribute in the EMPLOYEE table.
* In the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.



### Entity Sets:

### Relationship Sets:

### Participation Constraints:

* [**Prepare**](https://prepinsta.com/dbms/participation-constraints/)
* [**Prime Video**](https://prepinsta.com/online-classes/)
* [**OffCampus**](https://prepinsta.com/offcampus-updates/?utm_source=top_menu_prepinsta&utm_medium=Offcampus_top_menu&utm_campaign=Top_menu_traffic)
* [**Internship**](https://prepinsta.com/internship-program/?utm_source=top+menu&utm_medium=prepinsta)
* [**Placement Stats**](https://prepinsta.com/prepinsta-prime-placement-report/?utm_source=top_menu&utm_medium=nav_bar&utm_campaign=placement_report_top_menu)

# **Participation Constraints in DBMS**

# **I**n a Relationship, Participation constraint specifies the presence of an entity when it is related to another entity in a relationship type. It is also called the minimum cardinality constraint.

This constraint **specifies the number of instances of an entity that are participating in the relationship type.**

There are two types of Participation constraint:

* + Total participation
  + Partial participation

### ****Total participation constraint****

* It specifies that**each entity present in the entity set must mandatorily participate in at least one relationship instance of that relationship set**, for this reason, it is also called as mandatory participation
* It is**represented using a double line** between the entity set and relationship set

### ****Partial participation****

* It specifies that **each entity in the entity set may or may not participate in the relationship instance of the relationship set**, is also called as optional participation
* It is represented using a**single line between the entity set and relationship set** in the ER diagram

## **Participation constraints in DBMS**

On this page we will learn about Participation constraints in DBMS.