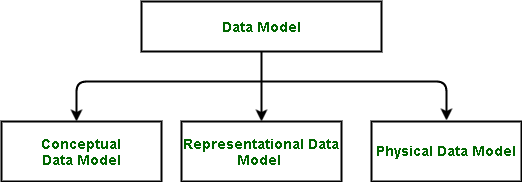
**Data model:**

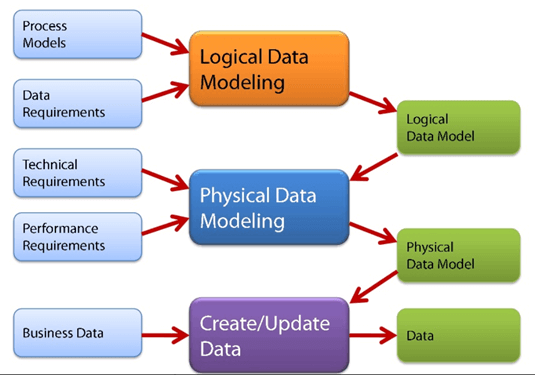
A Data Model in Database Management System (DBMS)  is the concept of tools that are developed to summarize the description of the database. Data Models provide us with a transparent picture of data which helps us in creating an actual database. It shows us from the design of the data to its proper implementation of data.

**Types of Relational Models**

1. Conceptual Data Model
2. Representational Data Model
3. Physical Data Model

It is basically classified into 3 types:-





**1. Conceptual Data Model**

The conceptual data model describes the database at a very high level and is useful to understand the needs or requirements of the database. It is this model, that is used in the requirement-gathering process i.e. before the Database Designers start making a particular database. One such popular model is the entity/relationship model (ER model). The E/R model specializes in entities, relationships, and even attributes that are used by database designers. In terms of this concept, a discussion can be made even with non-computer science(non-technical) users and stakeholders, and their requirements can be understood.

**Entity-Relationship Model( ER Model):** It is a high-level data model which is used to define the data and the relationships between them. It is basically a conceptual design of any database which is easy to design the view of data.

Components of ER Model:

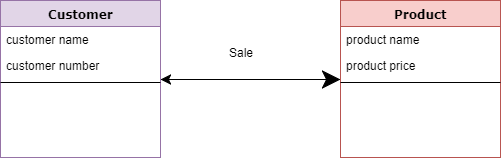
1. [**Entity:**](https://www.geeksforgeeks.org/difference-between-entity-entity-set-and-entity-type/) An entity is referred to as a real-world object. It can be a name, place, object, class, etc. These are represented by a rectangle in an ER Diagram.
2. [**Attributes:**](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/)An attribute can be defined as the description of the entity. These are represented by Eclipse in an ER Diagram. It can be Age, Roll Number, or Marks for a Student.
3. [**Relationship:**](https://www.geeksforgeeks.org/relationships-in-er-model/)Relationships are used to define relations among different entities. Diamonds and Rhombus are used to show Relationships.

The 3 basic tenants of Conceptual Data Model are

* **Entity**: A real-world thing
* **Attribute**: Characteristics or properties of an entity
* **Relationship**: Dependency or association between two entities

Data model example:

* Customer and Product are two entities. Customer number and name are attributes of the Customer entity
* Product name and price are attributes of product entity
* Sale is the relationship between the customer and product

Conceptual Data Model

**Characteristics of a conceptual data model**

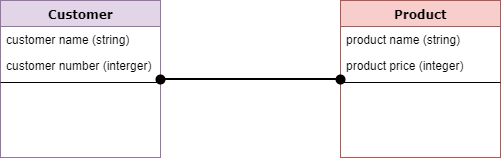
* Offers Organization-wide coverage of the business concepts.
* This type of Data Models are designed and developed for a business audience.
* The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the “real world.”

Conceptual data models known as Domain models create a common vocabulary for all stakeholders by establishing basic concepts and scope

**2. Representational Data Model**

This type of data model is used to represent only the logical part of the database and does not represent the physical structure of the database. The representational data model allows us to focus primarily, on the design part of the database. A popular representational model is a [Relational model](https://www.geeksforgeeks.org/relational-model-in-dbms/). The relational Model consists of [Relational Algebra](https://www.geeksforgeeks.org/introduction-of-relational-algebra-in-dbms/) and [Relational Calculus](https://www.geeksforgeeks.org/tuple-relational-calculus-trc-in-dbms/). In the Relational Model, we basically use tables to represent our data and the relationships between them. It is a theoretical concept whose practical implementation is done in Physical Data Model.

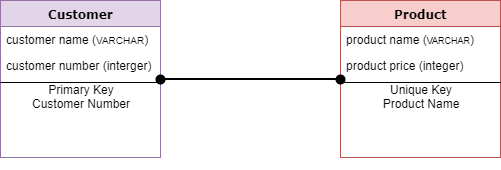
The advantage of using a Representational data model is to provide a foundation to form the base for the Physical model

Logical Data Model

**3. Physical Data Model**

 The physical Data Model is used to practically implement Relational Data Model. Ultimately, all data in a database is stored physically on a secondary storage device such as discs and tapes. This is stored in the form of files, records, and certain other data structures. It has all the information on the format in which the files are present and the structure of the databases, the presence of external data structures, and their relation to each other. Here, we basically save tables in memory so they can be accessed efficiently. In order to come up with a good physical model, we have to work on the relational model in a better way. [Structured Query Language (SQL)](https://www.geeksforgeeks.org/structured-query-language/) is used to practically implement Relational Algebra.

This Data Model describes **HOW** the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database.



**Characteristics of a physical data model:**

* The physical data model describes data need for a single project or application though it maybe integrated with other physical data models based on project scope.
* Data Model contains relationships between tables that which addresses cardinality and nullability of the relationships.
* Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
* Columns should have exact datatypes, lengths assigned and default values.
* Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined

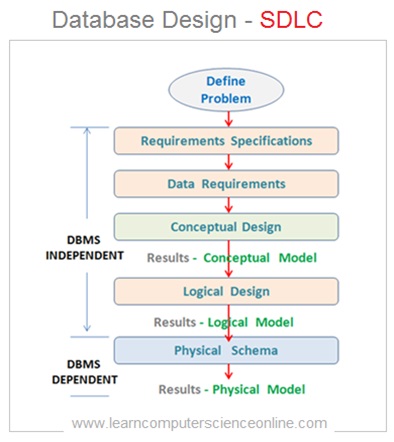
**Advantages of Data Models**

1. Data Models help us in representing data accurately.
2. It helps us in finding the missing data and also in minimizing Data Redundancy.
3. Data Model provides data security in a better way.
4. The data model should be detailed enough to be used for building the physical database.
5. The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.

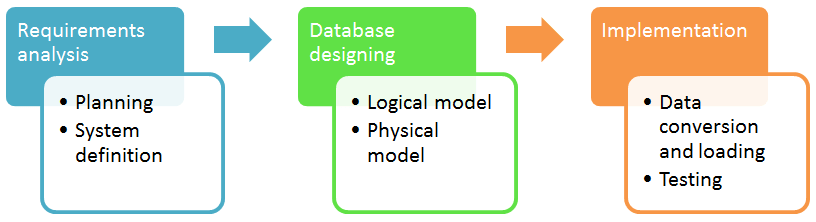
**Disadvantages of Data Models**

1. In the case of a vast database, sometimes it becomes difficult to understand the data model.
2. You must have the proper knowledge of [SQL](https://www.geeksforgeeks.org/structured-query-language/) to use physical models.
3. Even smaller change made in structure require modification in the entire application.
4. There is no set data manipulation language in DBMS.
5. To develop Data model one should know physical data stored characteristics.

**Overview of database design:**



**Database development life cycle:**

[](https://www.guru99.com/images/DatabaseDesignProcess(1).png)

The database development life cycle has a number of stages that are followed when developing database systems.

The steps in the development life cycle do not necessarily have to be followed religiously in a sequential manner.

On small database systems, the process of database design is usually very simple and does not involve a lot of steps.

In order to fully appreciate the above diagram, let’s look at the individual components listed in each step for overview of design process in DBMS.

**Requirements analysis**

* **Planning** – This stages of database design concepts are concerned with planning of entire Database Development Life Cycle. It takes into consideration the Information Systems strategy of the organization.
* **System definition** – This stage defines the scope and boundaries of the proposed database system.

**Database designing**

* **Logical model** – This stage is concerned with developing a database model based on requirements. The entire design is on paper without any physical implementations or specific DBMS considerations.
* **Physical model** – This stage implements the logical model of the database taking into account the DBMS and physical implementation factors.

**Implementation**

* **Data conversion and loading** – this stage of relational databases design is concerned with importing and converting data from the old system into the new database.
* **Testing** – this stage is concerned with the identification of errors in the newly implemented system. It checks the database against requirement specifications.

**Two Types of Database Techniques**

1. **Normalization**
2. **ER Modeling**

**Conceptual design:**

# Introduction of ER Model

The Entity Relational Model is a model for identifying entities to be represented in the [database](https://www.geeksforgeeks.org/what-is-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.

The Entity Relationship Diagram explains the relationship among the entities present in the database. ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects. In short, ER Diagram is the structural format of the database.

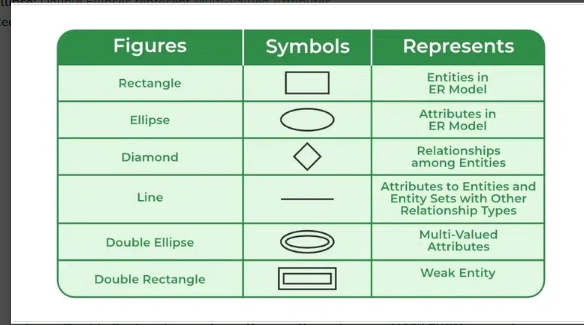
## Why Use ER Diagrams In DBMS?

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

## Symbols Used in ER Model

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

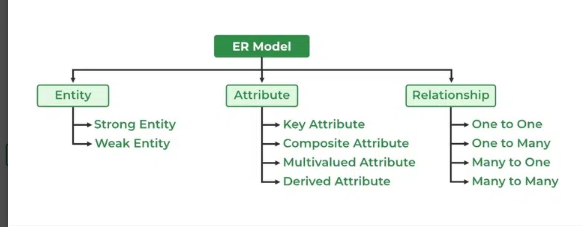
* **Rectangles:**Rectangles represent Entities in ER Model.
* **Ellipses:**Ellipses represent Attributes in ER Model.
* **Diamond:**Diamonds represent Relationships among Entities.
* **Lines:**Lines represent attributes to entities and entity sets with other relationship types.
* **Double Ellipse:**Double Ellipses represent [Multi-Valued Attributes](https://iotap.geeksforgeeks.org/problems/what-is-the-difference-between-single-valued-and-multi-valued-attributes).
* **Double Rectangle:**Double Rectangle represents a Weak Entity.



*Symbols used in ER Diagram*

## Components of ER Diagram

ER Model consists of Entities, Attributes, and Relationships among Entities in a Database System.

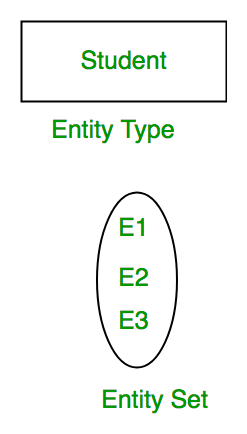


*Components of ER Diagram*

### Entity

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

**Entity Set:** An Entity is an object of Entity Type and a set of all entities is called an entity set. For Example, E1 is an entity having Entity Type Student and the set of all students is called Entity Set. In ER diagram, Entity Type is represented as:



*Entity Set*

#### 1. Strong Entity

A [Strong Entity](https://www.geeksforgeeks.org/difference-between-strong-and-weak-entity/) is a type of entity that has a key Attribute. Strong Entity does not depend on other Entity in the Schema. It has a primary key, that helps in identifying it uniquely, and it is represented by a rectangle. These are called Strong Entity Types.

#### 2. Weak Entity

An Entity type has a key attribute that uniquely identifies each entity in the entity set. But some entity type exists for which key attributes can’t be defined. These are called [Weak Entity types](https://www.geeksforgeeks.org/weak-entity-set-in-er-diagrams/).

**For Example,** A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents don’t have existed without the employee. So Dependent will be a **Weak Entity Type** and Employee will be Identifying Entity type for Dependent, which means it is **Strong Entity Type**.

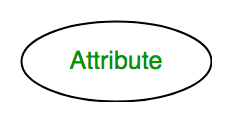
A weak entity type is represented by a Double Rectangle. The participation of weak entity types is always total. The relationship between the weak entity type and its identifying strong entity type is called identifying relationship and it is represented by a double diamond.



*Strong Entity and Weak Entity*

### ****Attributes****

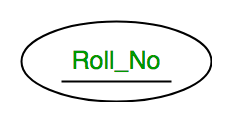
[Attributes](https://www.geeksforgeeks.org/types-of-attributes-in-er-model/) are the properties that define the entity type. For example, Roll\_No, Name, DOB, Age, Address, and Mobile\_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.



*Attribute*

#### ****1. Key Attribute****

The attribute which **uniquely identifies each entity** in the entity set is called the key attribute. For example, Roll\_No will be unique for each student. In ER diagram, the key attribute is represented by an oval with underlying lines.



*Key Attribute*

#### ****2. Composite Attribute****

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



*Composite Attribute*

#### ****3. Multivalued Attribute****

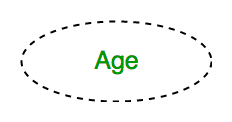
An attribute consisting of more than one value for a given entity. For example, Phone\_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.



*Multivalued Attribute*

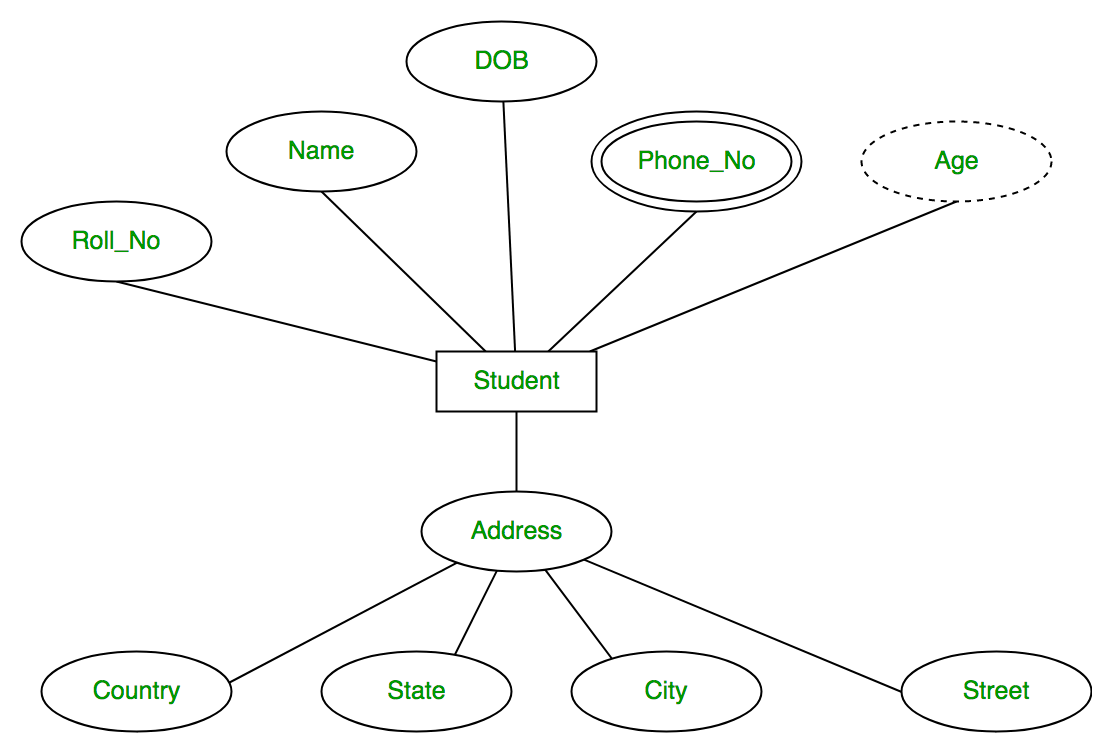
#### ****4. Derived Attribute****

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.



*Derived Attribute*

The Complete Entity Type Student with its Attributes can be represented as:



*Entity and Attributes*

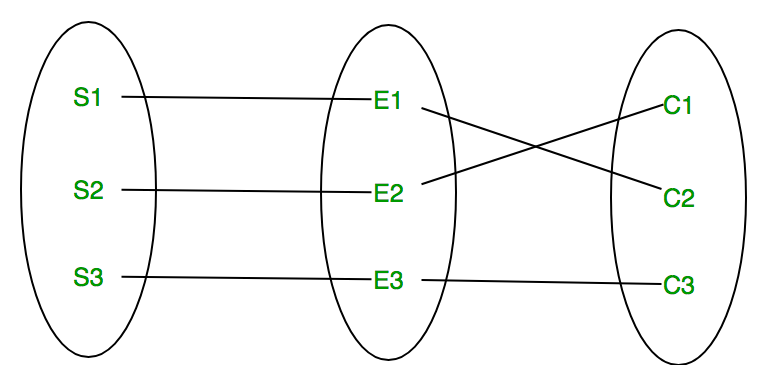
### ****Relationship Type and Relationship Set****

A Relationship Type represents the association between entity types. For example, ‘Enrolled in’ is a relationship type that exists between entity type Student and Course. In ER diagram, the relationship type is represented by a diamond and connecting the entities with lines.



*Entity-Relationship Set*

A set of relationships of the same type is known as a relationship set. The following relationship set depicts S1 as enrolled in C2, S2 as enrolled in C1, and S3 as registered in C3.

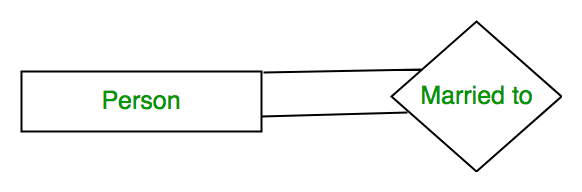


*Relationship Set*

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

The number of different entity sets participating in a relationship set is called the [degree of a relationship set.](https://www.geeksforgeeks.org/degree-of-relations-in-dbms/)

**1. Unary Relationship:**When there is only ONE entity set participating in a relation, the relationship is called a unary relationship. For example, one person is married to only one person.



*Unary Relationship*

**2. Binary Relationship:**When there are TWO entities set participating in a relationship, the relationship is called a binary relationship. For example, a Student is enrolled in a Course.



*Binary Relationship*

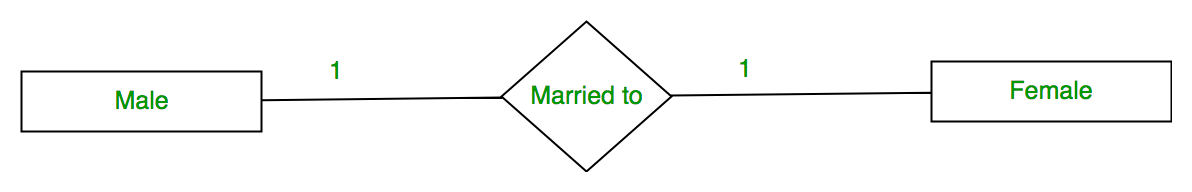
**3. n-ary Relationship:**When there are n entities set participating in a relation, the relationship is called an n-ary relationship.

#### ****Cardinality****

The number of times an entity of an entity set participates in a relationship set is known as [cardinality](https://www.geeksforgeeks.org/cardinality-in-dbms/). Cardinality can be of different types:

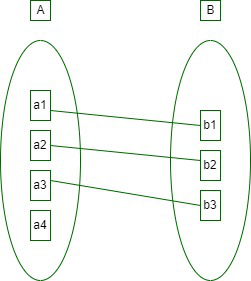
**1. One-to-One:** When each entity in each entity set can take part only once in the relationship, the cardinality is one-to-one. Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.

the total number of tables that can be used in this is 2.



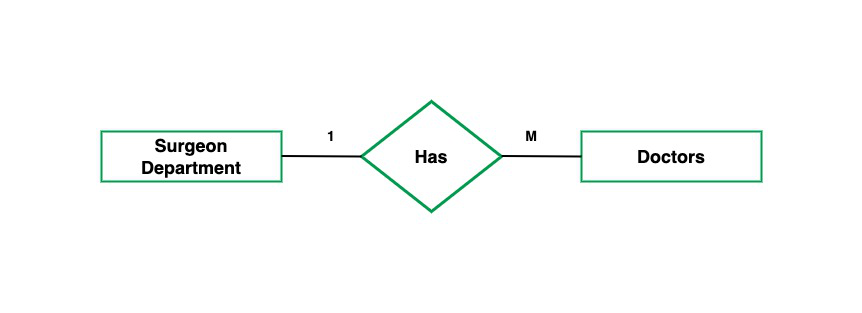
*One-to-One Cardinality*

Using Sets, it can be represented as:



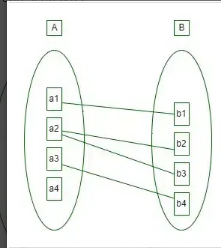
*Set Representation of One-to-One*

**2. One-to-Many:** In one-to-many mapping as well where each entity can be related to more than one relationship and the total number of tables that can be used in this is 2.



*One to Many*

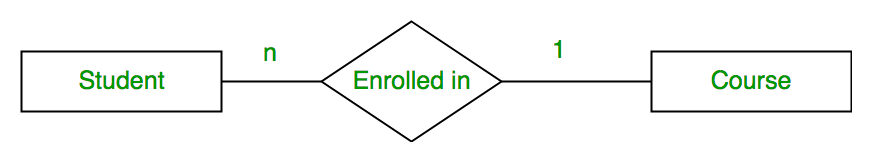
Using sets, one-to-many cardinality can be represented as:



*Set Representation of One-to-Many*

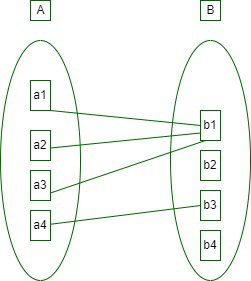
**3. Many-to-One:** When entities in one entity set can take part only once in the relationship set and entities in other entity sets can take part more than once in the relationship set, cardinality is many to one. Let us assume that a student can take only one course but one course can be taken by 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.



*Many-to-One Relationship*

Using Sets, it can be represented as:

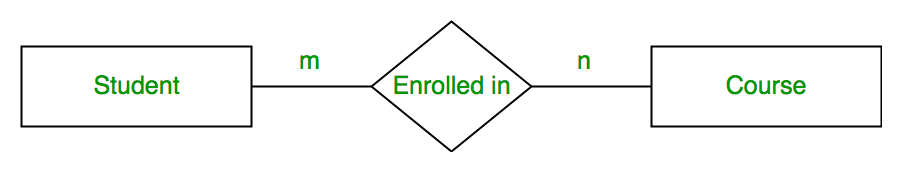


*Set Representation of Many-to-One*

In this case, each student is taking only 1 course but 1 course has been taken by many students.

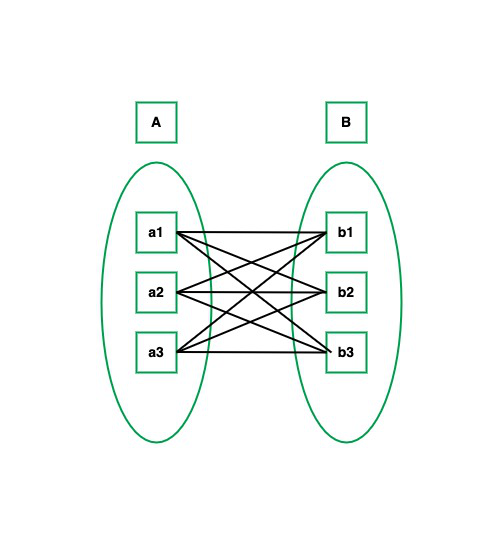
**4. Many-to-Many:**When entities in all entity sets can take part more than once in the relationship cardinality is many to many. Let us assume that a student can take more than one course and one course can be taken by many students. So the relationship will be many to many.

the total number of tables that can be used in this is 3.



*Many-to-Many*

Using Sets, it can be represented as:



*Many-to-Many Set Representation*

In this example, student S1 is enrolled in C1 and C3 and Course C3 is enrolled by S1, S3, and S4. So it is many-to-many relationships.

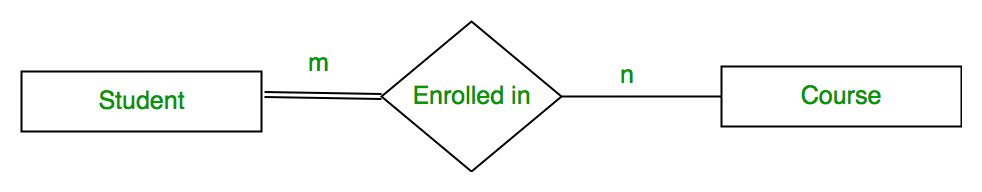
### ****Participation Constraint****

[Participation Constraint](https://www.geeksforgeeks.org/structural-constraints-of-relationships-in-er-model/) is applied to the entity participating in the relationship set.

**1. Total Participation –** Each entity in the entity setmust participate in the relationship. If each student must enroll in a course, the participation of students will be total. Total participation is shown by a double line in the ER diagram.

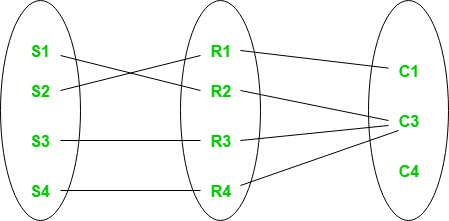
**2. Partial Participation –** The entity in the entity set may or may NOT participate in the relationship. If some courses are not enrolled by any of the students, the participation in the course will be partial.

The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.



*Total Participation and Partial Participation*

Using Set, it can be represented as,



*Set representation of Total Participation and Partial Participation*

Every student in the Student Entity set participates in a relationship but there exists a course C4 that is not taking part in the relationship.

# Constraints in DBMS:

Constraints enforce limits to the data or type of data that can be inserted/updated/deleted from a table. The whole purpose of constraints is to maintain the **data integrity**during an update/delete/insert into a table. In this tutorial we will learn several types of constraints that can be created in RDBMS.

**Types of constraints**

* NOT NULL
* UNIQUE
* DEFAULT
* CHECK
* Key Constraints – PRIMARY KEY, FOREIGN KEY
* Domain constraints
* Mapping constraints

#### NOT NULL:

NOT NULL constraint makes sure that a column does not hold NULL value. When we don’t provide value for a particular column while inserting a record into a table, it takes NULL value by default. By specifying NULL constraint, we can be sure that a particular column(s) cannot have NULL values.

Example:

CREATE TABLE STUDENT(

ROLL\_NO INT **NOT NULL**,

STU\_NAME VARCHAR (35) **NOT NULL**,

STU\_AGE INT **NOT NULL**,

STU\_ADDRESS VARCHAR (235),

PRIMARY KEY (ROLL\_NO)

);

Read more about [this constraint here](https://beginnersbook.com/2014/05/not-null-constraint-in-sql/).

#### UNIQUE:

UNIQUE Constraint enforces a column or set of columns to have unique values. If a column has a unique constraint, it means that particular column cannot have duplicate values in a table.

CREATE TABLE STUDENT(

ROLL\_NO INT NOT NULL,

STU\_NAME VARCHAR (35) NOT NULL **UNIQUE**,

STU\_AGE INT NOT NULL,

STU\_ADDRESS VARCHAR (35) **UNIQUE**,

PRIMARY KEY (ROLL\_NO)

);

## Key constraints:

#### PRIMARY KEY:

[Primary key](https://beginnersbook.com/2015/04/primary-key-in-dbms/) uniquely identifies each record in a table. It must have unique values and cannot contain nulls. In the below example the ROLL\_NO field is marked as primary key, that means the ROLL\_NO field cannot have duplicate and null values.

CREATE TABLE STUDENT(

ROLL\_NO   INT  NOT NULL,

STU\_NAME VARCHAR (35)  NOT NULL UNIQUE,

STU\_AGE INT NOT NULL,

STU\_ADDRESS VARCHAR (35) UNIQUE,

**PRIMARY KEY** (ROLL\_NO)

);

# Foreign key in DBMS

**Definition**: Foreign keys are the columns of a table that points to the [primary key](https://beginnersbook.com/2015/04/primary-key-in-dbms/) of another table. They act as a cross-reference between tables.

**For example**:  
In the below example the Stu\_Id column in Course\_enrollment table is a foreign key as it points to the primary key of the Student table.

**Course\_enrollment table:**

|  |  |
| --- | --- |
| Course\_Id | Stu\_Id |
| C01 | 101 |
| C02 | 102 |
| C03 | 101 |
| C05 | 102 |
| C06 | 103 |
| C07 | 102 |

**Student table:**

|  |  |  |
| --- | --- | --- |
| Stu\_Id | Stu\_Name | Stu\_Age |
| 101 | Chaitanya | 22 |
| 102 | Arya | 26 |
| 103 | Bran | 25 |
| 104 | Jon | 21 |

**Note**: Practically, the foreign key has nothing to do with the primary key tag of another table, if it points to a unique column (not necessarily a primary key) of another table then too, it would be a foreign key. So, a correct definition of foreign key would be: Foreign keys are the columns of a table that points to the [candidate key](https://beginnersbook.com/2015/04/candidate-key-in-dbms/) of another table.

**Domain Constraints**

In a database table, domain constraints are guidelines that specify the acceptable values for a certain property or field. These restrictions guarantee data consistency and aid in preventing the entry of inaccurate or inconsistent data into the database. The following are some instances of domain restrictions in a DBMS −

* **Data type constraints** − These limitations define the kinds of data that can be kept in a column. A column created as VARCHAR can take string values, but a column specified as INTEGER can only accept integer values.
* **Length Constraints** − These limitations define the largest amount of data that may be put in a column. For instance, a column with the definition VARCHAR(10) may only take strings that are up to 10 characters long.
* **Range constraints** − The allowed range of values for a column is specified by range restrictions. A column designated as DECIMAL(5,2), for example, may only take decimal values up to 5 digits long, including 2 decimal places.
* **Nullability constraints** − Constraints on a column's capacity to accept NULL values are known as nullability constraints. For instance, a column that has the NOT NULL definition cannot take NULL values.
* **Unique constraints** − Constraints that require the presence of unique values in a column or group of columns are known as unique constraints. For instance, duplicate values are not allowed in a column with the UNIQUE definition.
* **Check constraints** − Constraints for checking data: These constraints outline a requirement that must hold for any data placed into the column. For instance, a column with the definition CHECK (age > 0) can only accept ages that are greater than zero.
* **Default constraints** − Constraints by default: Default constraints automatically assign a value to a column in case no value is provided. For example, a column with a DEFAULT value of 0 will have 0 as its value if no other value is specified.