**Database Modeling**

A data model is simply a diagram that displays a set of tables and the relationship between them. This helps us in understanding the purpose of the tables as well as their dependencies.

Whenever there is data there is always a requirement to store data in the database. The data model is used to represent the data objects in the database, the association between the data objects, and also rules. The data model defines rules to store the data in the database.

A data model is applicable to a software development that involves creation of database objects to store and manipulate data. This includes transaction systems as well as data warehouse.

**Uses of data models**

* It is used to represent all the data objects in the database accurately. If any data is omitted it can create problems while performing database operations.
* The data model helps to represent the data in logical and physical levels. ‘
* The relational tables, foreign and primary keys are all defined by the data models.
* The data redundancy can be eliminated from the database as the data model can easily identify the redundant data. The data models can also be used to identify the missing data and can take appropriate actions according to that.
* As the data models help to represent the data virtually the developers use this virtual representation to create an actual database from that.
* Using the data models while creating the database helps to maintain the database and helps to upgrade the database with fewer efforts.

**Stages of Data Modelling**

* Conceptual Data model
* Logical Data Model
* Physical Data Model

#### 1. Conceptual

This type of data model is used to define what the system actually contains. The conceptual data model is mostly used by data architects and business stakeholders. The business concepts and the business rules are managed by using the conceptual data model.

#### 2. Logical

This type of data model is used to define how the system will actually implement without knowing the database management system. The logical data model is developed by business analysts and data architects. The data structures and implementation rules are defined in this model.

#### 3. Physical

This type of data model is used to define how the system will actually implement without knowing the database management system. The physical data model is developed by database developers and database administrators. The physical data model is used to actually implement the database.

**ER MODEL**

* ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
* It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
* In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

ER Model is used to model the logical view of the system from data perspective.

Component of ER Diagram



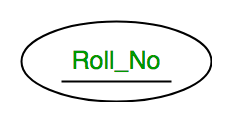
**Entity, Entity Type, Entity Set –**

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.

An Entity is an object of Entity Type and set of all entities is called as entity set.

**Attribute(s):**  
Attributes are the **properties which define the entity type**. For example, Roll\_No, Name, DOB, Age, Address, Mobile\_No are the attributes which defines entity type Student. In ER diagram, attribute is represented by an oval.

**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.



**Composite Attribute –**

An attribute **composed of many other attribute** 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.



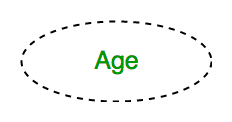
**Multivalued Attribute –**

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

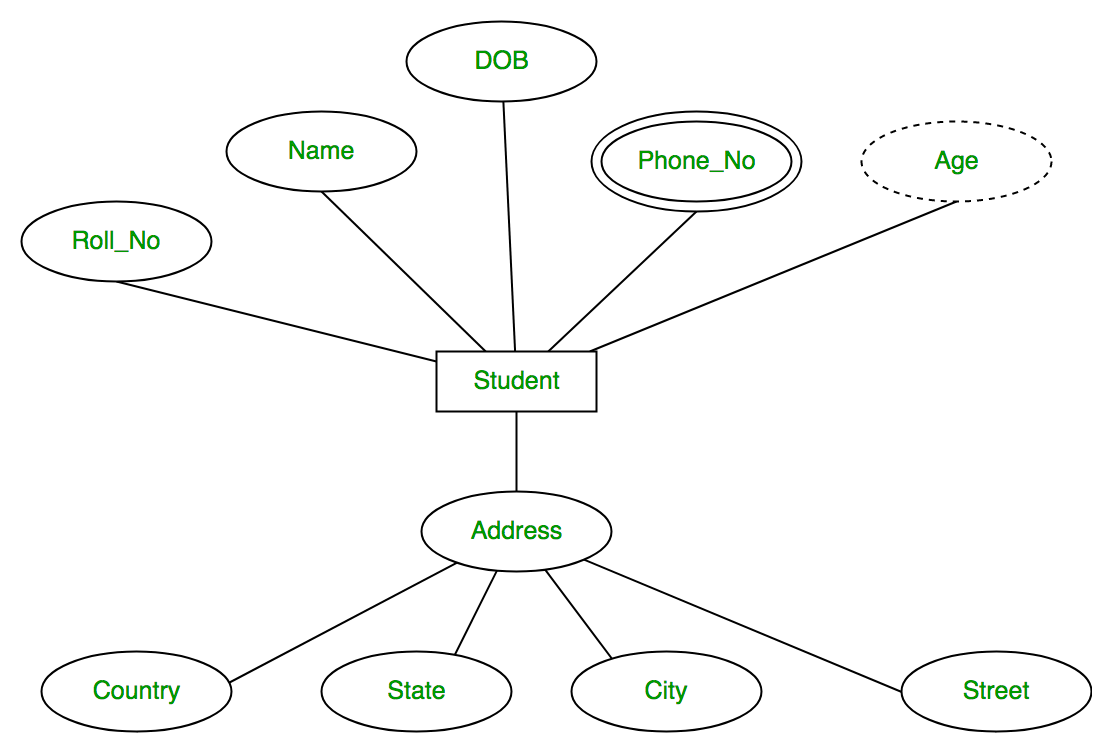


**Derived Attribute –**

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



The complete entity type**Student** with its attributes can be represented as:

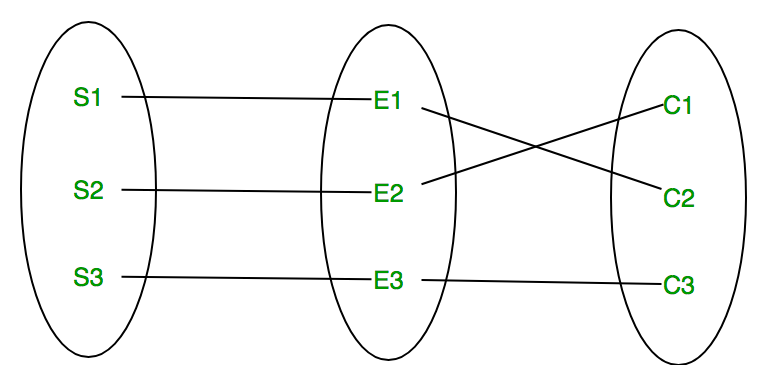


**Relationship Type and Relationship Set:**

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, relationship type is represented by a diamond and connecting the entities with lines.



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

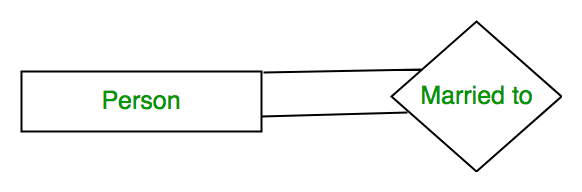


**Degree of a relationship set:**

The number of different entity sets **participating in a relationship** set is called as degree of a relationship set.

1. **Unary Relationship –**

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



**Binary Relationship –**

When there are **TWO entities set participating in a relation**, the relationship is called as binary relationship. For example, Student is enrolled in Course.

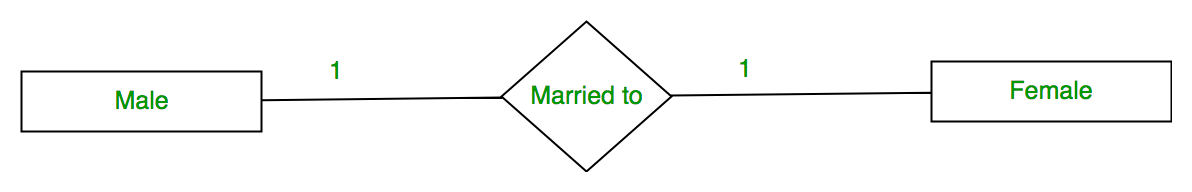


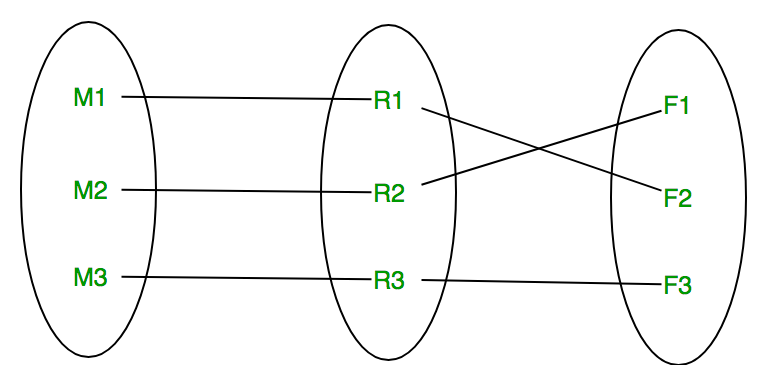
1. **n-ary Relationship –**

When there are n entities set participating in a relation, the relationship is called as n-ary relationship.

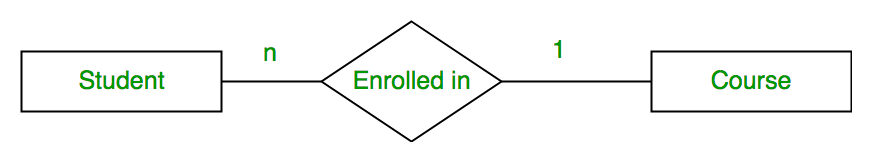
**Cardinality:**  
The **number of times an entity of an entity set participates in a relationship** set is known as cardinality. Cardinality can be of different types:

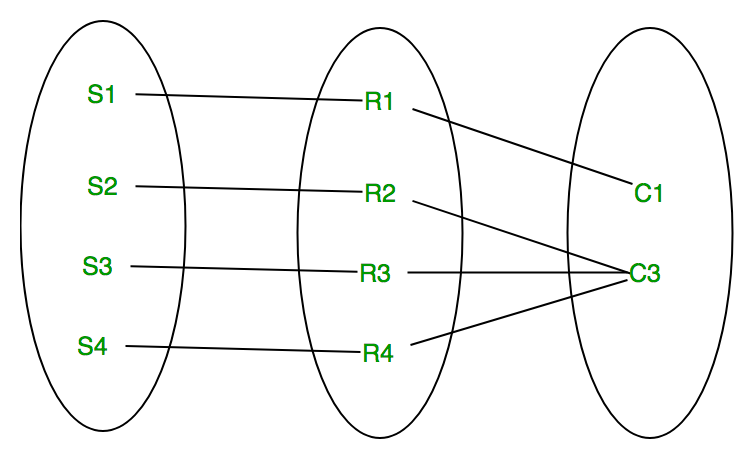
1. **One to one –** When each entity in each entity set can take part **only once in the relationship**, the cardinality is one to one. Let us assume that a male can marry to one female and a female can marry to one male. So the relationship will be one to one.

Using Sets, it can be represented as:



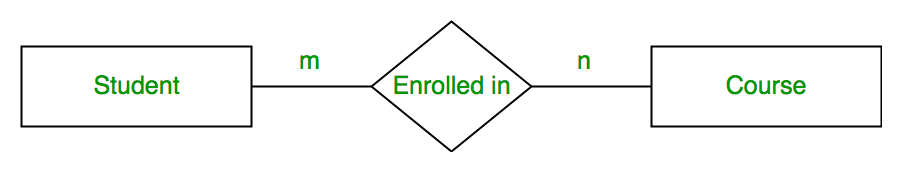
1. **Many to one –** When entities in one entity set **can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set,** cardinality is many to one. Let us assume that a student can take only one course but one course can be taken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

Using Sets, it can be represented as:

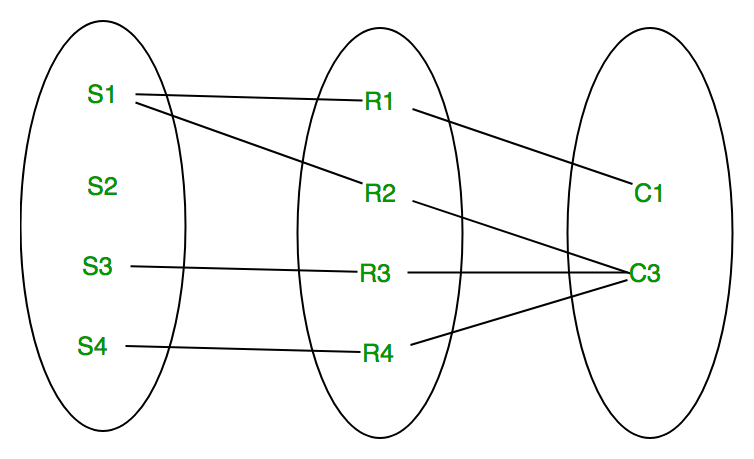


In this case, each student is taking only 1 course but 1 course has been taken by many students.

1. **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.



Using sets, it can be represented as:



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 is applied on the entity participating in the relationship set.

1. **Total Participation –** Each entity in the entity set**must participate** in the relationship. If each student must enroll in a course, the participation of student will be total. Total participation is shown by double line in ER diagram.
2. **Partial Participation –** The entity in the entity set **may or may NOT participat**e in the relationship. If some courses are not enrolled by any of the student, the participation of course will be partial.

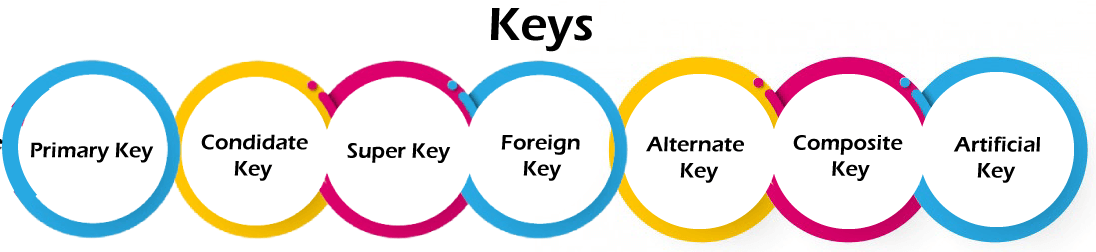
The diagram depicts the ‘Enrolled in’ relationship set with Student Entity set having total participation and Course Entity set having partial participation.

# **DBMS 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.

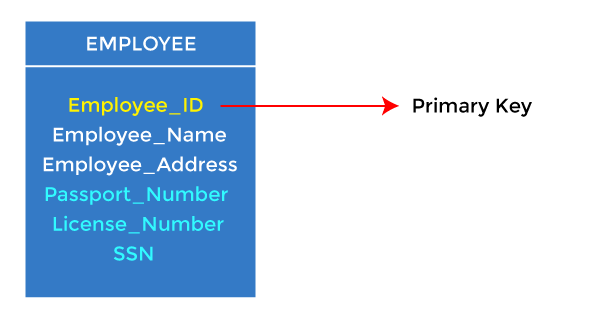
**For example,** ID is used as a key in the Student table because it is unique for each student. In the PERSON table, passport\_number, license\_number, SSN are keys since they are unique for each person.

## Types of keys:



### **1. Primary key**

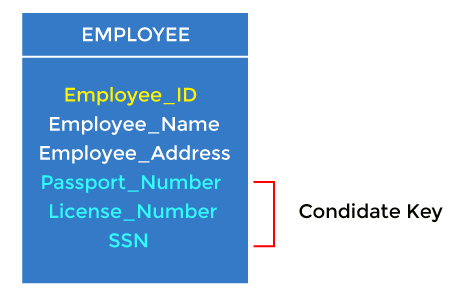
* It is the first key used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys, as we saw in the PERSON table. The key which is most suitable from those lists becomes a primary key.
* In the EMPLOYEE table, ID can be the primary key since it is unique for 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**

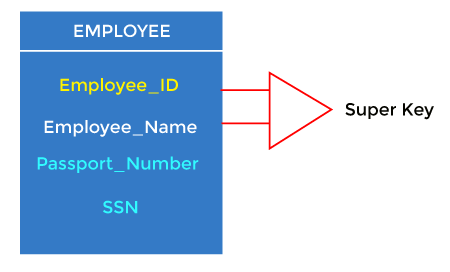
* A candidate key is an attribute or set of attributes that can uniquely identify a tuple.
* Except for the primary key, the remaining attributes are considered a candidate key. The candidate keys are as strong as the primary key.

**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.

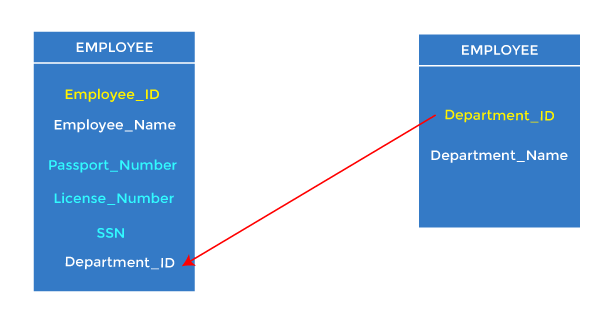


**For example:** In the above EMPLOYEE table, for(EMPLOEE\_ID, EMPLOYEE\_NAME), the name of two employees can be the same, but their EMPLYEE\_ID can't be the same. Hence, this combination can also be a key.

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.



### **5. Alternate key**

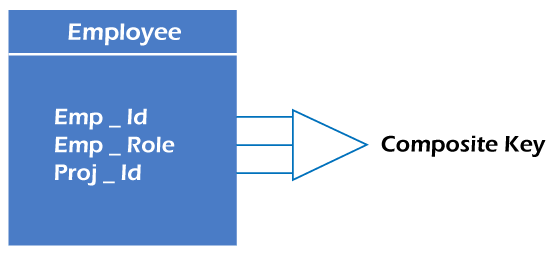
There may be one or more attributes or a combination of attributes that uniquely identify each tuple in a relation. These attributes or combinations of the attributes are called the candidate keys. One key is chosen as the primary key from these candidate keys, and the remaining candidate key, if it exists, is termed the alternate key. **In other words,** the total number of the alternate keys is the total number of candidate keys minus the primary key. The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.

**For example,** employee relation has two attributes, Employee\_Id and PAN\_No, that act as candidate keys. In this relation, Employee\_Id is chosen as the primary key, so the other candidate key, PAN\_No, acts as the Alternate key.



### **6. Composite key**

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.



**For example,** in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp\_ID, Emp\_role, and Proj\_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.

### **7. Artificial key**

The key created using arbitrarily assigned data are known as artificial keys. These keys are created when a primary key is large and complex and has no relationship with many other relations. The data values of the artificial keys are usually numbered in a serial order.

**For example,** the primary key, which is composed of Emp\_ID, Emp\_role, and Proj\_ID, is large in employee relations. So it would be better to add a new virtual attribute to identify each tuple in the relation uniquely.

