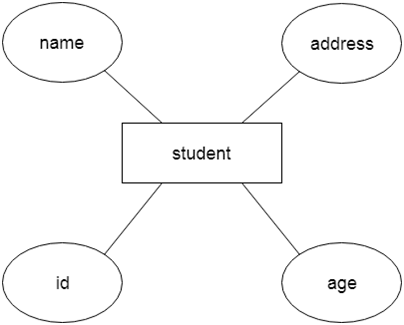
ER model

* 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 represented as a diagram called an entity-relationship diagram.

**For example,** Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



## Component of ER Diagram



### 1.Entity:

An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.

Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



**Weak Entity**

An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



### 2. Attribute

The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.



1. **Key Attribute**

The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



**b. Composite Attribute**

An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



1. **Multivalued Attribute**

An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

For example, a student can have more than one phone number.



**d. Derived Attribute**

An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



### 3. Relationship

A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



Types of relationship are as follows:

**a. One-to-One Relationship**

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

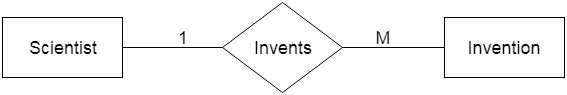
For example, A female can marry to one male, and a male can marry to one female.



**b. One-to-many relationship**

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

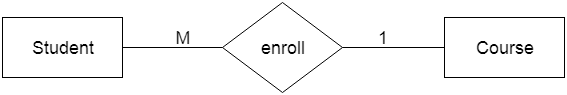
For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



**c. Many-to-one relationship**

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

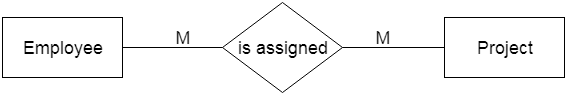
For example, Student enrolls for only one course, but a course can have many students.



**d. Many-to-many relationship**

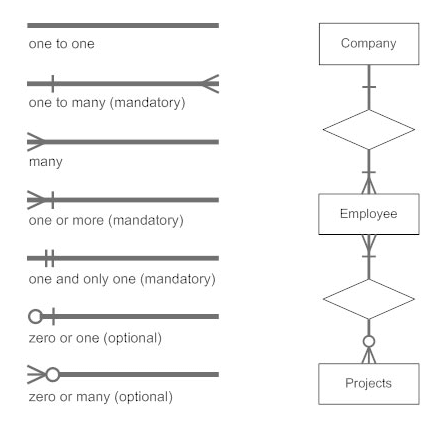
When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



# Notation of ER diagram

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:



# Mapping Constraints

* mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.
* It is most useful in describing the relationship sets that involve more than two entity sets.
* For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:

1.One to one (1:1)

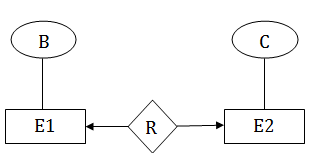
2.One to many (1:M)

3.Many to one (M:1)

4.Many to many (M:M)

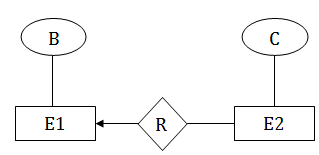
## One-to-one

In one-to-one mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with at most one entity in E1.



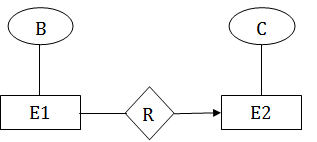
## One-to-many

In one-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with at most one entity in E1.



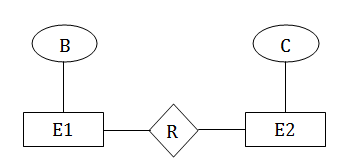
## Many-to-one

In one-to-many mapping, an entity in E1 is associated with at most one entity in E2, and an entity in E2 is associated with any number of entities in E1.



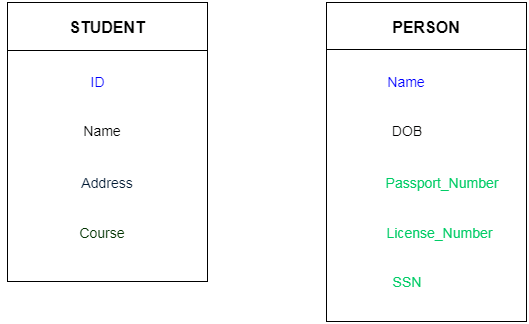
## Many-to-many

In many-to-many mapping, an entity in E1 is associated with any number of entities in E2, and an entity in E2 is associated with any number of entities in E1.

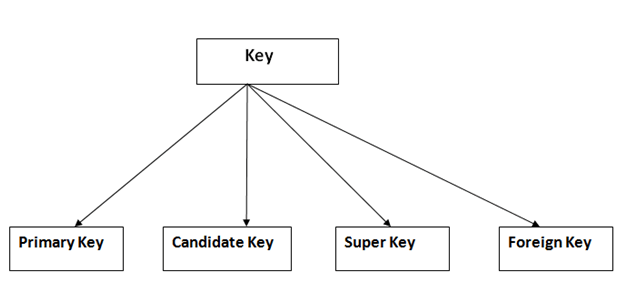


# Keys

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

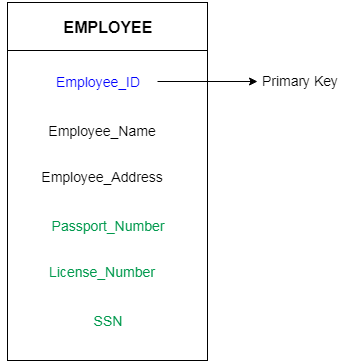


## Types of key:



### 1. Primary key

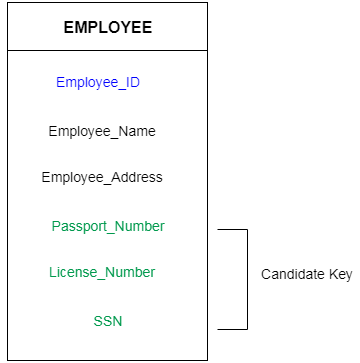
* It is the first key which is used to identify one and only one instance of an entity uniquely. An entity can contain multiple keys as we saw in PERSON table. The key which is most suitable from those lists become a primary key.
* In the EMPLOYEE table, ID can be primary key since it is unique for each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



### 2. Candidate key

* A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
* The remaining attributes except for primary key are considered as 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. Rest of the attributes like SSN, Passport\_Number, and License\_Number, etc. are considered as a candidate key.



### 3. Super Key

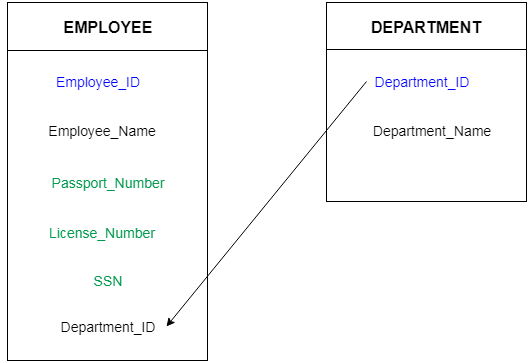
Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.

**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

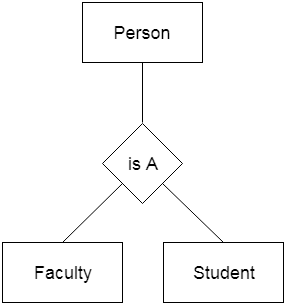
* Foreign keys are the column of the table which is used to point to the primary key of another table.
* In a company, every employee works in a specific department, and employee and department are two different entities. So we can't store the information of the department in the employee table. That's 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.
* Now in the EMPLOYEE table, Department\_Id is the foreign key, and both the tables are related.



# Generalization

* Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
* In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
* Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
* In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.

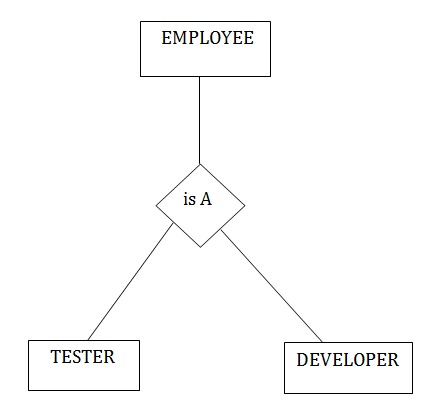
For example, Faculty and Student entities can be generalized and create a higher level entity Person.



# Specialization

* Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
* Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
* Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.

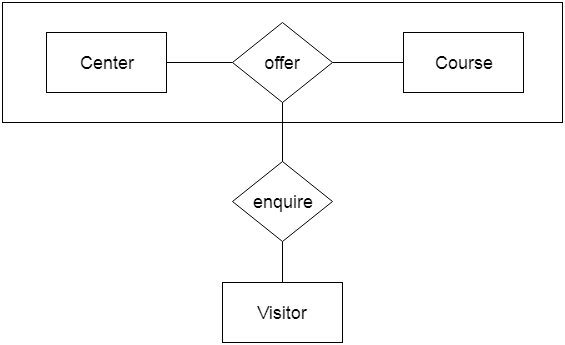
For example: In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.



# Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

**For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.

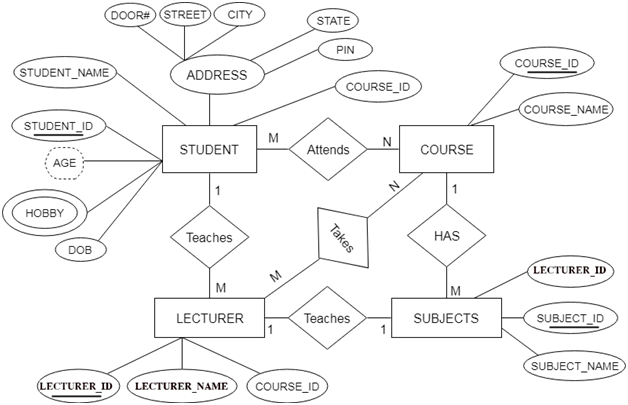


# Reduction of ER diagram to Table

The database can be represented using the notations, and these notations can be reduced to a collection of tables.

In the database, every entity set or relationship set can be represented in tabular form.

The ER diagram is given below:



There are some points for converting the ER diagram to the table:

* Entity type becomes a table.

In the given ER diagram, LECTURE, STUDENT, SUBJECT and COURSE forms individual tables.

* All single-valued attribute becomes a column for the table.

In the STUDENT entity, STUDENT\_NAME and STUDENT\_ID form the column of STUDENT table. Similarly, COURSE\_NAME and COURSE\_ID form the column of COURSE table and so on.

* A key attribute of the entity type represented by the primary key.

In the given ER diagram, COURSE\_ID, STUDENT\_ID, SUBJECT\_ID, and LECTURE\_ID are the key attribute of the entity.

* The multivalued attribute is represented by a separate table.

In the student table, a hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD\_HOBBY with column name STUDENT\_ID and HOBBY. Using both the column, we create a composite key.

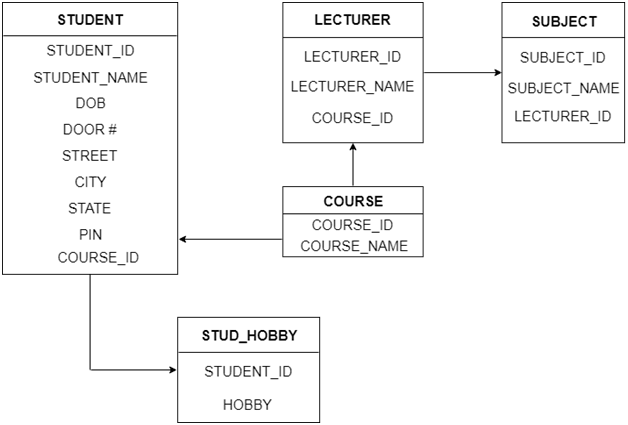
* Composite attribute represented by components.

In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

* Derived attributes are not considered in the table.

In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

Using these rules, you can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:



**Table structure**

# Relationship of higher degree

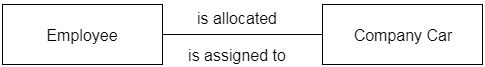
The degree of relationship can be defined as the number of occurrences in one entity that is associated with the number of occurrences in another entity.

There is the three degree of relationship:

1. One-to-one (1:1)
2. One-to-many (1:M)
3. Many-to-many (M:N)

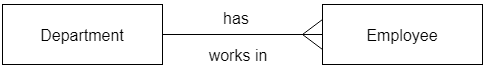
### 1. One-to-one

* In a one-to-one relationship, one occurrence of an entity relates to only one occurrence in another entity.
* A one-to-one relationship rarely exists in practice.
* **For example:** if an employee is allocated a company car then that car can only be driven by that employee.
* Therefore, employee and company car have a one-to-one relationship.



### 2. One-to-many

* In a one-to-many relationship, one occurrence in an entity relates to many occurrences in another entity.
* **For example:** An employee works in one department, but a department has many employees.
* Therefore, department and employee have a one-to-many relationship.



### 3. Many-to-many

* In a many-to-many relationship, many occurrences in an entity relate to many occurrences in another entity.
* Same as a one-to-one relationship, the many-to-many relationship rarely exists in practice.
* **For example:** At the same time, an employee can work on several projects, and a project has a team of many employees.
* Therefore, employee and project have a many-to-many relationship.

