**UNIT II**

**Entity-Relationship Model**: Introduction, The Building blocks of an Entity Relationship Diagram, classification of entity sets, attribute classification, relationship degree, relationship classification, reducing ER diagram to tables, enhanced entity-relationship model (EER model), generalization and specialization, ISA relationship and attribute inheritance, multiple inheritance, constraints on specialization and generalization, aggregation and composition, entity clusters, connection traps, advantages of ER modelling.

---

## 2.1 Introduction:

Peter Chen first proposed modelling databases using a graphical technique that humans can relate to easily. Humans can easily perceive entities and their characteristics in the real world and represent any relationship with one another. Entity–Relationship (ER) model gives the conceptual model of the world to be represented in the database. The main motivation for defining the ER model is to provide a high level model for conceptual database design, which acts as an intermediate stage prior to mapping the enterprise being modelled onto a conceptual level.

The ER model achieves a high degree of data independence which means that the database designer do not have to worry about the physical structure of the database. A database schema in ER model can be pictorially represented by Entity–Relationship diagram.

## 2.2 The Building Blocks of an Entity–Relationship Diagram

The basic building blocks of Entity- Relationship diagram are Entity, Attribute and Relationship.

### Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities.

### Entity Type

An entity type or entity set is a collection of similar entities. Some examples of entity types are:

- All students in PSG, say STUDENT.
- All courses in PSG, say COURSE.
- All departments in PSG, say DEPARTMENT.

### Relationship

The association among entities is called a relationship. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

### Attributes

Attributes are properties of entity types. In other words, entities are described in a database by a set of attributes. The following are example of attributes:

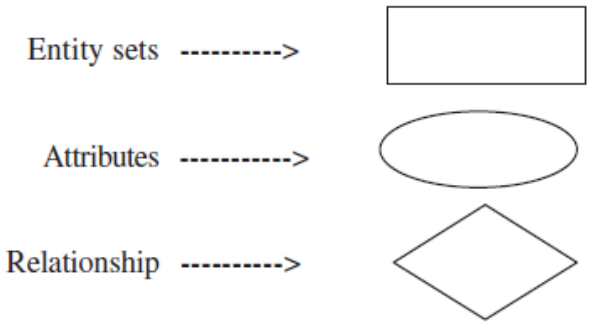
- Brand, cost, and weight are the attributes of CELLPHONE.
- Roll number, name, and grade are the attributes of STUDENT.

### ER Diagram

The ER diagram is used to represent database schema. In ER diagram:

- A rectangle represents an entity set.
- An ellipse represents an attribute.
- A diamond represents a relationship.

– Lines represent linking of attributes to entity sets and of entity sets to relationship sets.




**Example of ER diagram**

In the ER diagram the two entities are STUDENT and CLASS. Two simple attributes which are associated with the STUDENT are Roll number and the name. The attributes associated with the entity CLASS are Subject Name and Hall Number. The relationship between the two entities STUDENT and CLASS is Attends.

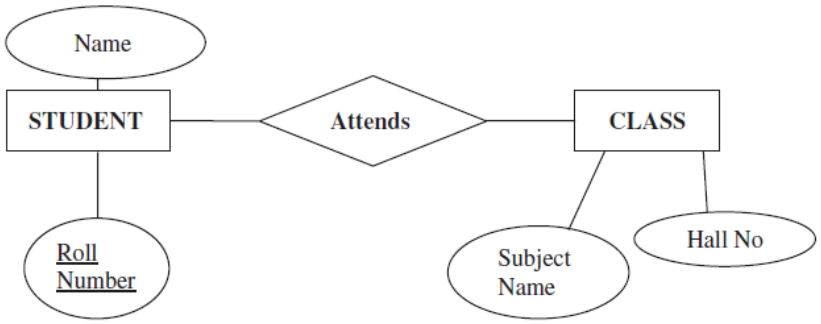
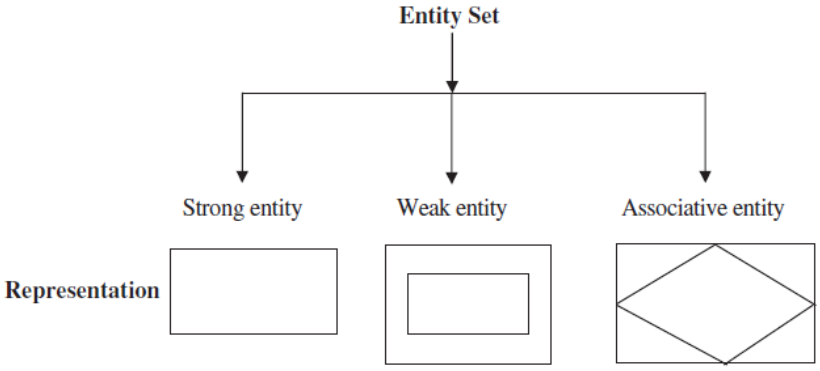



Fig. 2.1. ER diagram

**2.3 Classification of Entity Sets**

Entity sets can be broadly classified into:

1. Strong entity.
2. Weak entity.
3. Associative entity.




**Strong Entity**

Strong entity is one whose existence does not depend on other entity.

**Example**

Consider the example, student takes course. Here student is a strong entity.


In this example, course is considered as weak entity because, if there are no students to take a particular course, then that course cannot be offered. The COURSE entity depends on the STUDENT entity.

**Weak Entity**

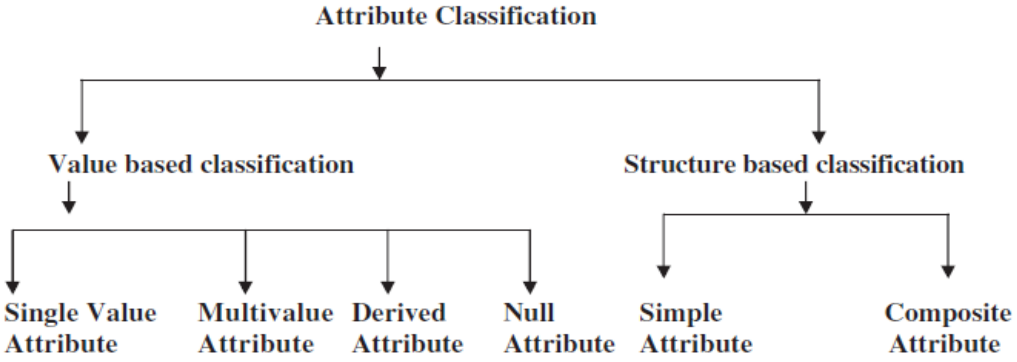
Weak entity is one whose existence depends on other entity.

**Example**

Consider the example, customer borrows loan. Here loan is a weak entity. For every loan, there should be at least one customer. Here the entity loan depends on the entity customer hence loan is a weak entity.

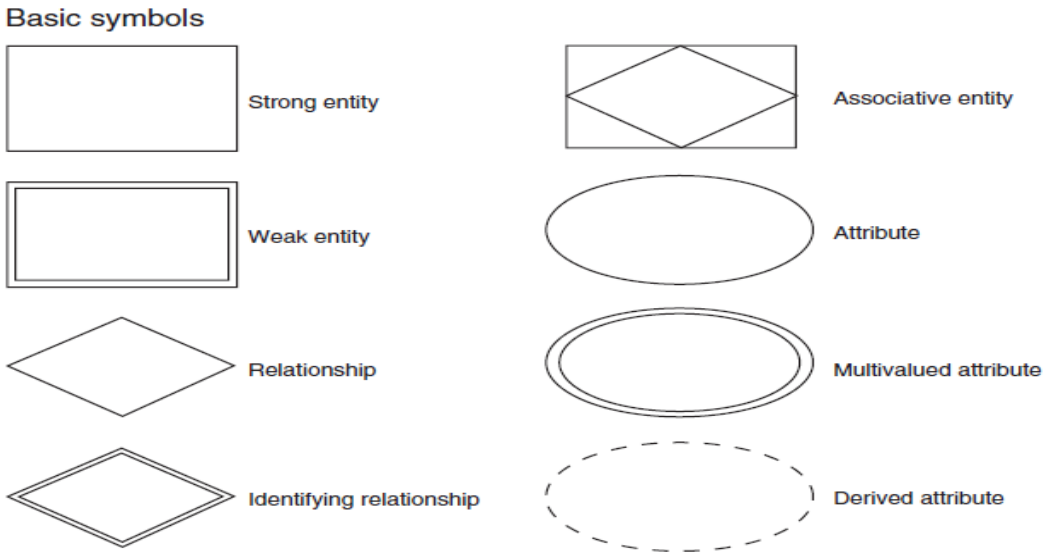

## 2.4 Attribute Classification

Attribute is used to describe the properties of the entity. This attribute can be broadly classified based on value and structure. Based on value the attribute can be classified into single value, multivalue, derived, and null value attribute. Based on structure, the attribute can be classified as simple and composite attribute.




**Symbols Used in ER Diagram**

The elements in ER diagram are Entity, Attribute, and Relationship. The different types of entities like strong, weak, and associative entity, different types of attributes like multivalued and derived attributes and identifying relationship and their corresponding symbols are shown later.




**Single Value Attribute**

Single value attribute means, there is only one value associated with that attribute.

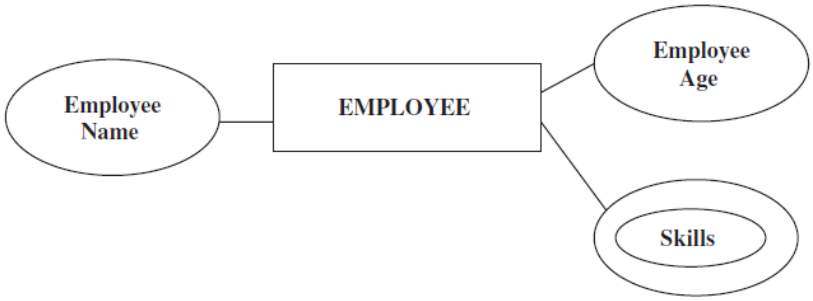
**Example**

The examples of single value attribute are age of a person, Roll number of the student, Registration number of a car, etc.

**Multivalued Attribute**

In the case of multivalue attribute, more than one value will be associated with that attribute. For example, a person can have more than one phone number, email_address, etc.

**Examples of Multivalued Attribute**

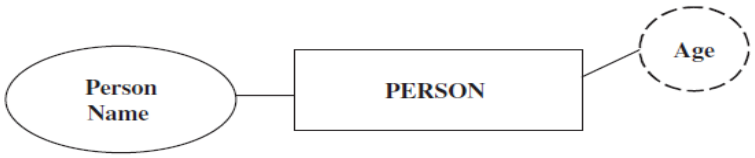



**Derived Attribute**

Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.

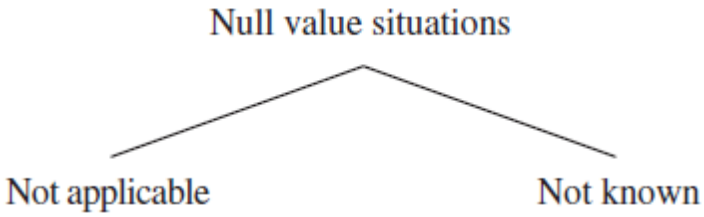
**Example of Derived Attribute**

1. Age of a person can be derived from the date of birth of the person. In this example, age is the derived attribute.




**Null Value Attribute**

In some cases, a particular entity may not have any applicable value for an attribute. For such situation, a special value called null value is created.

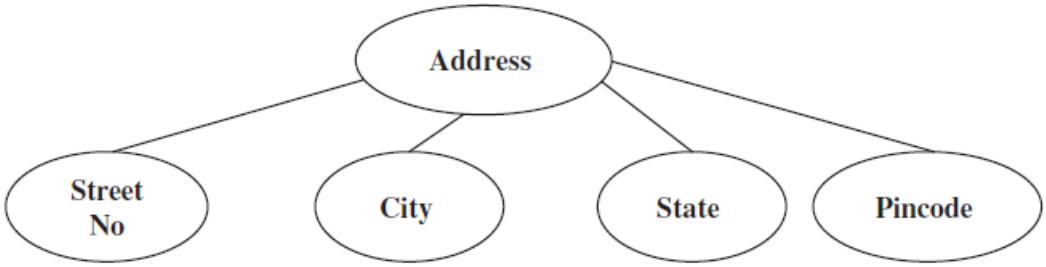



**Example**
In application forms, there is one column called phone no. if a person do not have phone then a null value is entered in that column.

**Simple attribute** – Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits

**Composite attribute** – Composite attributes are made of more than one simple ttribute. For example, a student's complete name may have first_name and last_name.

Consider the attribute "address" which can be further subdivided into Street name, City, and State.

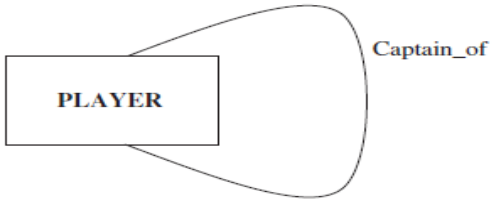



## 2.5 Relationship Degree

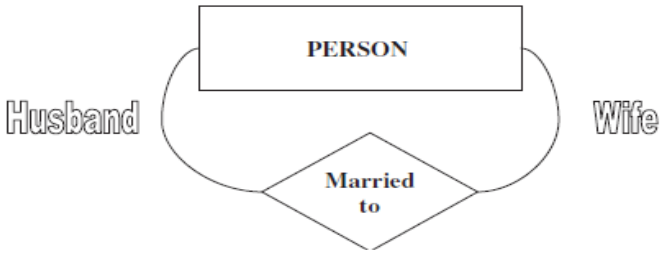
Relationship degree refers to the number of associated entities. The relationship degree can be broadly classified into unary, binary, and ternary relationship.

### 2.5.1 Unary Relationship

The unary relationship is otherwise known as recursive relationship. In the unary relationship the number of associated entity is one. An entity related to itself is known as recursive relationship.

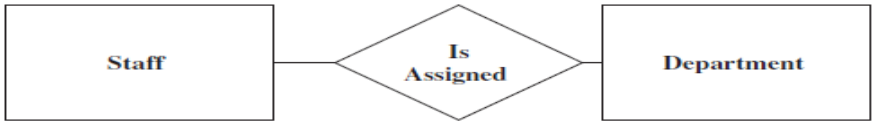



**Roles and Recursive Relation**
When an entity sets appear in more than one relationship, it is useful to add labels to connecting lines. These labels are called as roles.
**Example**
In this example, Husband and wife are referred as roles.




### 2.5.2 Binary Relationship

In a binary relationship, two entities are involved. Consider the example; each staff will be assigned to a particular department. Here the two entities are STAFF and DEPARTMENT.

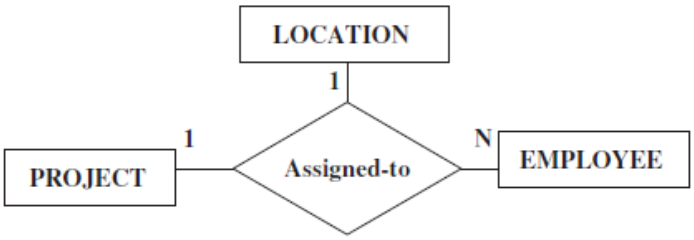



### 2.5.3 Ternary Relationship

In a ternary relationship, three entities are simultaneously involved. Ternary relationships are required when binary relationships are not sufficient to accurately describe the semantics of an association among three entities.

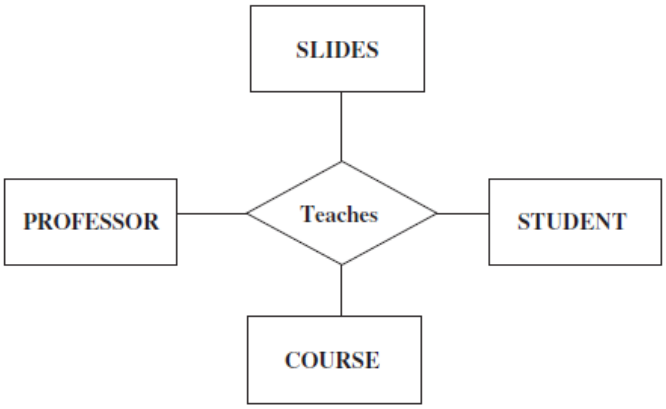
**Example**

Consider the example of employee assigned a project. Here we are considering three entities EMPLOYEE, PROJECT, and LOCATION. The relationship is "assigned-to." Many employees will be assigned to one project hence it is an example of one-to-many relationship.




### 2.5.4 Quaternary Relationships

Quaternary relationships involve four entities. The example of quaternary relationship is "A professor teaches a course to students using slides." Here the four entities are PROFESSOR, SLIDES, COURSE, and STUDENT. The relationships between the entities are "Teaches."




## 2.6 Relationship Classification

Relationship is an association among one or more entities. This relationship can be broadly classified into one-to-one relation, one-to-many relation, many-to- many relation and recursive relation.

### 2.6.1 One-to-Many Relationship Type

The relationship that associates one entity to more than one entity is called one-to-many relationship. Example of one-to-many relationship is Country having states. For one country there can be more than one state hence it is an example of one-to-many relationship. Another example of one-to-many relationship is parent–child relationship. For one parent there can be more than one child. Hence it is an example of one-to-many relationship.

### 2.6.2 One-to-One Relationship Type

One-to-one relationship is a special case of one-to-many relationship. True one-to-one relationship is rare. The relationship between the President and the country is an example of one-to-one relationship. For a particular country there will be only one President. In general, a country will not have more than one President hence the relationship between the country and the President is an example of one-to-one relationship. Another example of one-to-one relationship is House to Location. A house is obviously in only one location.

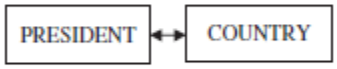
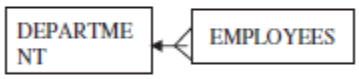
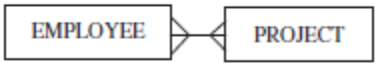
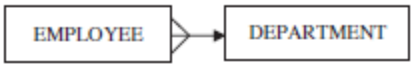
### 2.6.3 Many-to-Many Relationship Type

The relationship between EMPLOYEE entity and PROJECT entity is an example of many-to-many relationship. Many employees will be working in many projects hence the relationship between employee and project is many-to- many relationship.

### 2.6.4 Many-to-One Relationship Type

The relationship between EMPLOYEE and DEPARTMENT is an example of many-to-one relationship. There may be many EMPLOYEES working in one DEPARTMENT. Hence relationship between EMPLOYEE and DEPARTMENT is many-to-one relationship. The four relationship types are summarized and shown in Table 2.1.

**Table 2.1.** Relationship types

| Relationship type | Representation | Example |
|---|---|---|
| One-to-one |  |   |
| One-to-many |  |   |
| Many-to-many |  |   |
| Many-to-one |  |   |

## 2.7 Reducing ER Diagram to Tables

To implement the database, it is necessary to use the relational model. There is a simple way of mapping from ER model to the relational model. There is almost one-to-one correspondence between ER constructs and the relational ones.

### 2.7.1 Mapping Algorithm

The mapping algorithm gives the procedure to map ER diagram to tables.

The rules in mapping algorithm are given as:

While determining the minimum number of tables required for binary relationships with given cardinality ratios, keep these thumb rules in your mind-

- For binary relationship with cardinality ration m : n , separate and individual tables will be drawn for each entity set and relationship.
- For binary relationship with cardinality ratio either m : 1 or 1 : n , always remember "many side will consume the relationship" i.e. a combined table will be drawn for many side entity set and relationship set.
- For binary relationship with cardinality ratio 1 : 1 , two tables will be required. You can combine the relationship set with any one of the entity sets.

**Regular Entity**

Regular entities are entities that have an independent existence and generally represent real-world objects such as persons and products. Regular entities are represented by rectangles with a single line.

### 2.7.2 Mapping Regular Entities

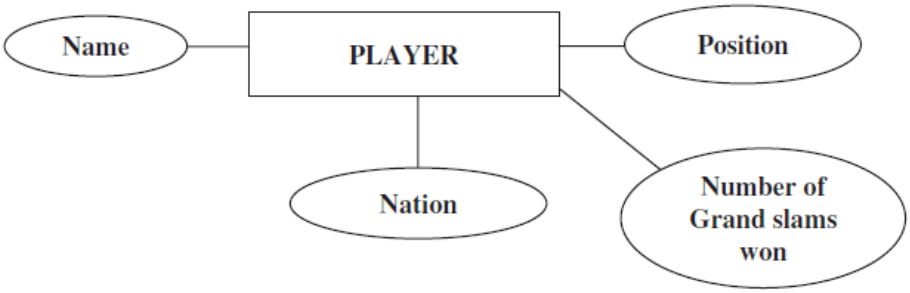
– Each regular entity type in an ER diagram is transformed into a relation. The name given to the relation is generally the same as the entity type.

– Each simple attribute of the entity type becomes an attribute of the relation.

– The identifier of the entity type becomes the primary key of the corresponding relation.

**Example 1**

Mapping regular entity type tennis player




This diagram is converted into corresponding table as

| Player Name | Nation | Position | Number of Grand slams won |
|---|---|---|---|
| Roger Federer | Switzerland | 1 | 5 |
| Roddick | USA | 2 | 4 |

Here,

– **Entity name = Name of the relation or table**.

In our example, the entity name is PLAYER which is the name of the table

– **Attributes of ER diagram=Column name of the table**.

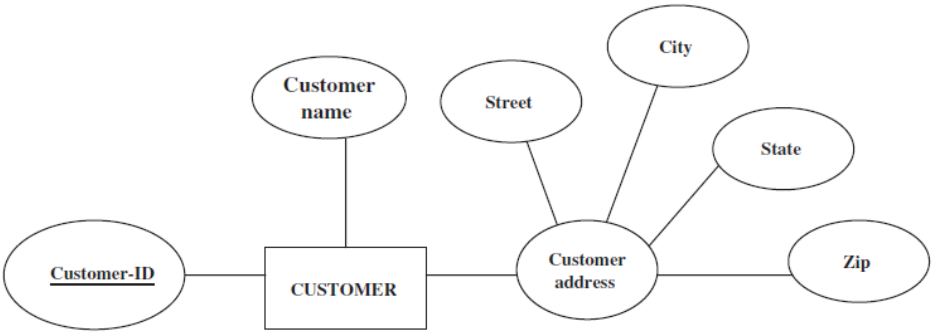
In our example the Name, Nation, Position, and Number of Grand slams won which forms the column of the table.

### 2.7.3 Converting Composite Attribute in an ER Diagram to Tables

When a regular entity type has a composite attribute, only the simple component attributes of the composite attribute are included in the relation.

**Example**

In this example the composite attribute is the Customer address, which consists of Street, City, State, and Zip.




CUSTOMER

| Customer-ID | Customer name | Street | City | State | Zip |
|---|---|---|---|---|---|

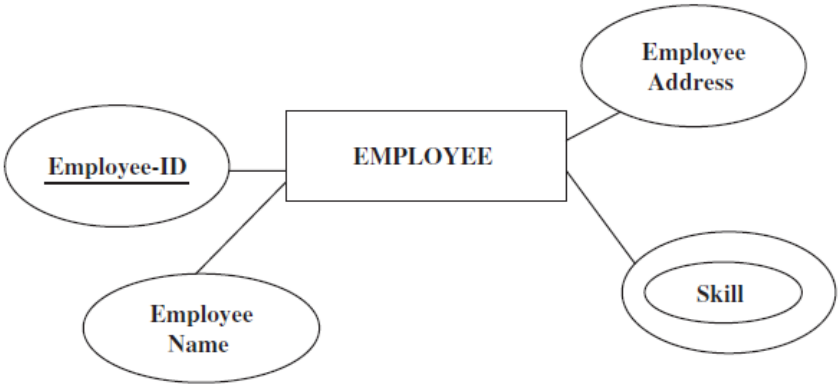
When the regular entity type contains a multi-valued attribute, two new relations are created.
The first relation contains all of the attributes of the entity type except the multi-valued attribute.
The second relation contains two attributes that form the primary key of the second relation. The first of these attributes is the primary key from the first relation, which becomes a foreign key in the second relation. The second is the multi-valued attribute.

### 2.7.4 Mapping Multi-valued Attributes in ER Diagram to Tables

A multi-valued attribute is having more than one value. One way to map a multi-valued attribute is to create two tables.

**Example**

In this example, the skill associated with the EMPLOYEE is a multi-valued attribute, since an EMPLOYEE can have more than one skill as fitter, electrician, turner, etc.




EMPLOYEE

| Employee-ID | Employee-Name | Employee-Address |
|---|---|---|

EMPLOYEE-SKILL

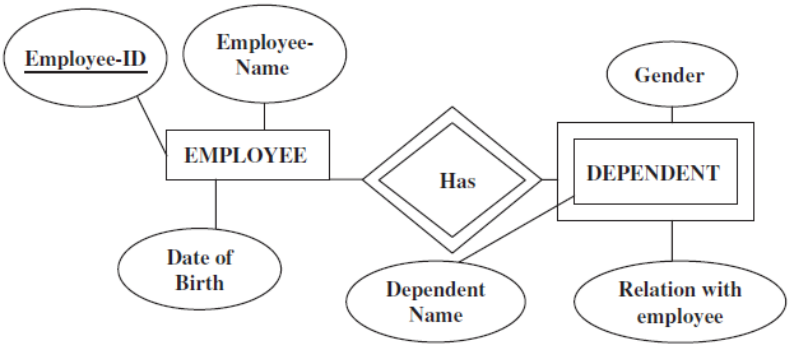
| EMPLOYEE-ID | Skill |
|---|---|

### 2.7.5 Converting "Weak Entities" in ER Diagram to Tables

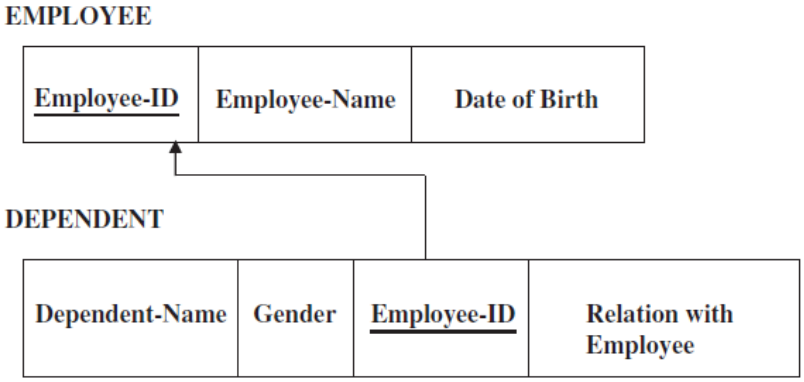
Weak entity type does not have an independent existence and it exists only through an identifying relationship with another entity type called the owner.

For each weak entity type, create a new relation and include all of the simple attributes as attributes of the relation. Then include the primary key of the identifying relation as a foreign key attribute to this new relation.

The primary key of the new relation is the combination of the primary key of the identifying and the partial identifier of the weak entity type. In this example DEPENDENT is weak entity.




The corresponding table is given by




### 2.7.6 Converting Binary Relationship to Table

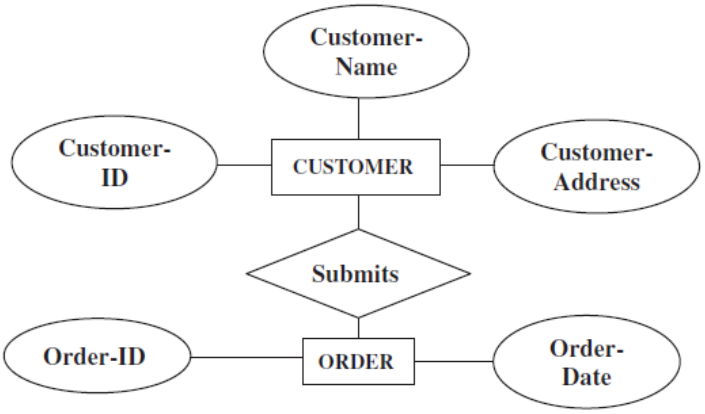
A relationship which involves two entities can be termed as binary relationship. This binary relationship can be one-to-one, one-to-many, many-to-one, and many-to-many.

**Mapping one-to-Many Relationship**

For each 1–M relationship, first create a relation for each of the two entity type's participation in the relationship.

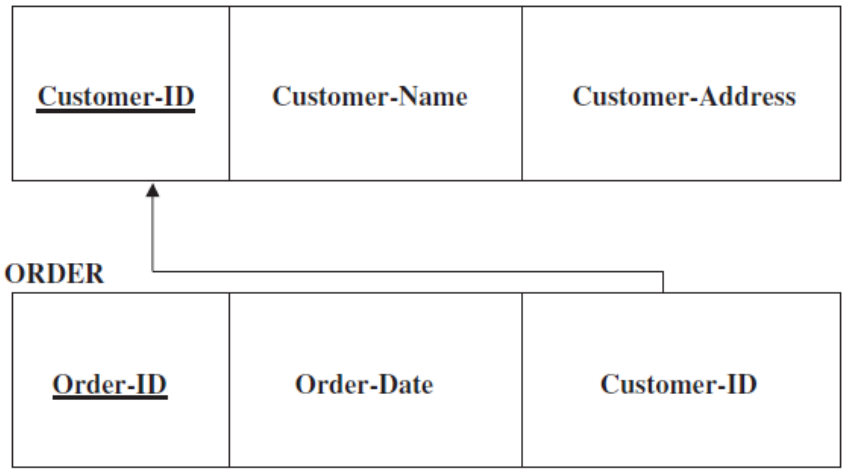
**Example**

One customer can give many orders. Hence the relationship between the two entities CUSTOMER and ORDER is one-to-many relationship. In one-to-many relationship, include the primary key attribute of the entity on the one-side of the relationship as a foreign key in the relation that is on the many side of the relationship.




Here we have two entities CUSTOMER and ORDER. The relationship between CUSTOMER and ORDER is one-to-many. For two entities CUSTOMER and ORDER, two tables namely CUSTOMER and ORDER are created as shown later. The primary key CUSTOMER ID in the CUSTOMER relation becomes the foreign key in the ORDER relation.

**CUSTOMER**




Binary one-to-one relationship can be viewed as a special case of one-to-many relationships.

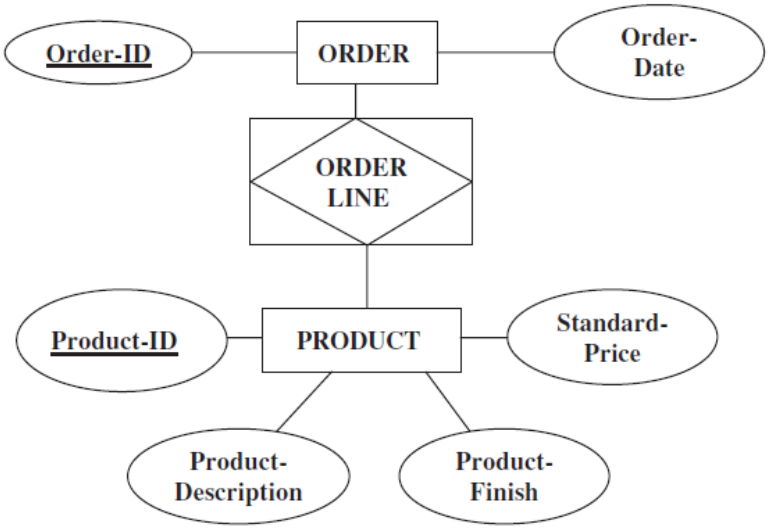
The process of mapping one-to-one relationship requires two steps. First, two relations are created, one for each of the participating entity types. Second, the primary key of one of the relations is included as a foreign key in the other relation.

### 2.7.7 Mapping Associative Entity to Tables

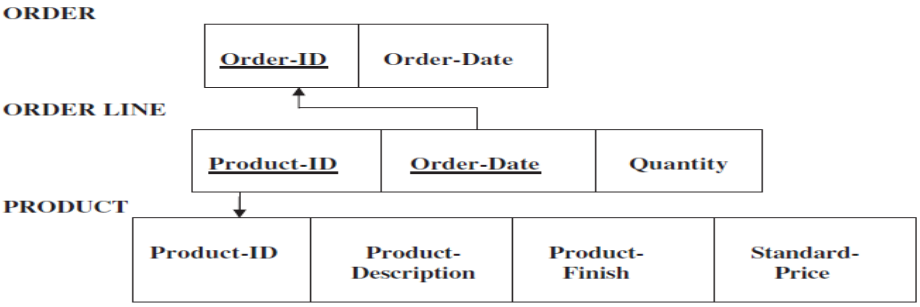
Many-to-many relationship can be modeled as an associative entity in the ER diagram.

**Example 1. (Without Identifier)**

Here the associative entity is ORDERLINE, which is without an identifier. That is the associative entity ORDERLINE is without any key attribute.



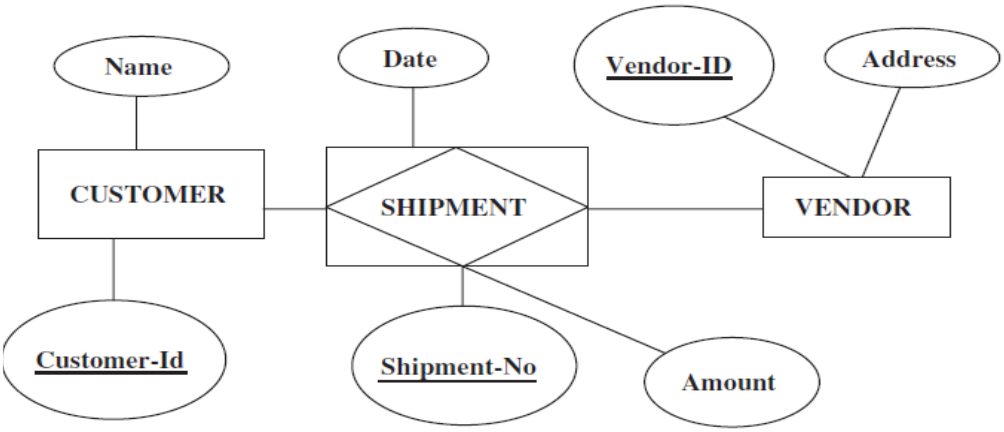

The first step is to create three relations, one for each of the two participating entity types and the third for the associative entity. The relation formed from the associative entity is associative relation.




**Example 2. (With Identifier)**

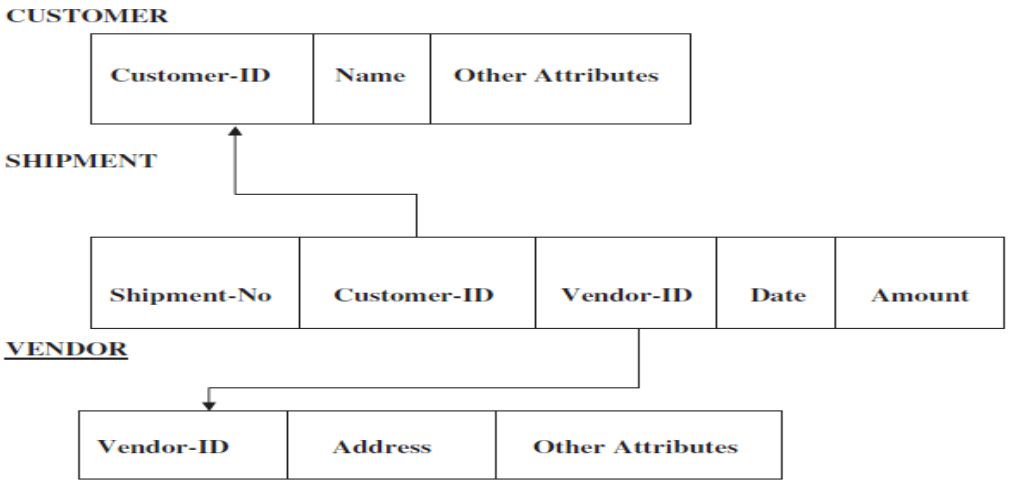
Sometimes data models will assign an identifier (surrogate identifier) to the associative entity type on the ER diagram. There are two reasons to motivate this approach:

1. The associative entity type has a natural identifier that is familiar to end user.
2. The default identifier may not uniquely identify instances of the associative entity.




(a) Shipment-No is a natural identifier to end user.

(b) The default identifier consisting of the combination of Customer-ID and Vendor-ID does not uniquely identify the instances of SHIPMENT.

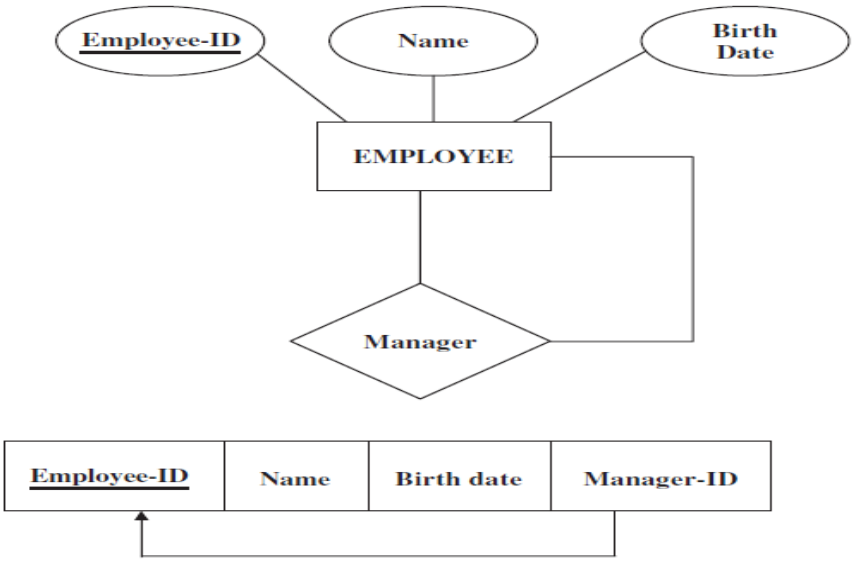



### 2.7.8 Converting Unary Relationship to Tables

Unary relationships are also called recursive relationships. The two most important cases of unary relationship are one-to-many and many-to-many.

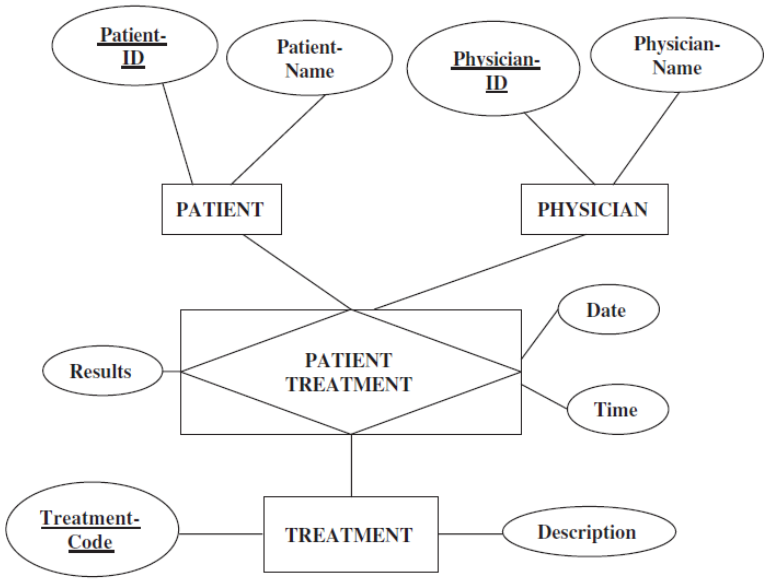
**One-to-many Unary Relationship**

Each employee has exactly one manager. A given employee may manage zero to many employees. The foreign key in the relation is named Manager-ID. This attribute has the same domain as the primary key Employee-ID.




### 2.7.9 Converting Ternary Relationship to Tables

A ternary relationship is a relationship among three entity types. The three entities given in this example are PATIENT, PHYSICIAN, and TREATMENT. The PATIENT–TREATMENT is an associative entity.




The primary key attributes – Patient ID, Physician ID, and Treatment Code – become foreign keys in PATIENT TREATMENT. These attributes are components of the primary key of PATIENT TREATMENT.

**PATIENT TREATMENT**

| Patient-ID | Patient-Name |
|---|---|

**PHYSICIAN**

| Physician-ID | Physician-Name |
|---|---|

**PATIENT TREATMENT**

| Patient-ID | Physician-ID | Treatment-Code | Date | Time | Results |
|---|---|---|---|---|---|

**TREATMENT**

| Treatment-Code | Description |
|---|---|