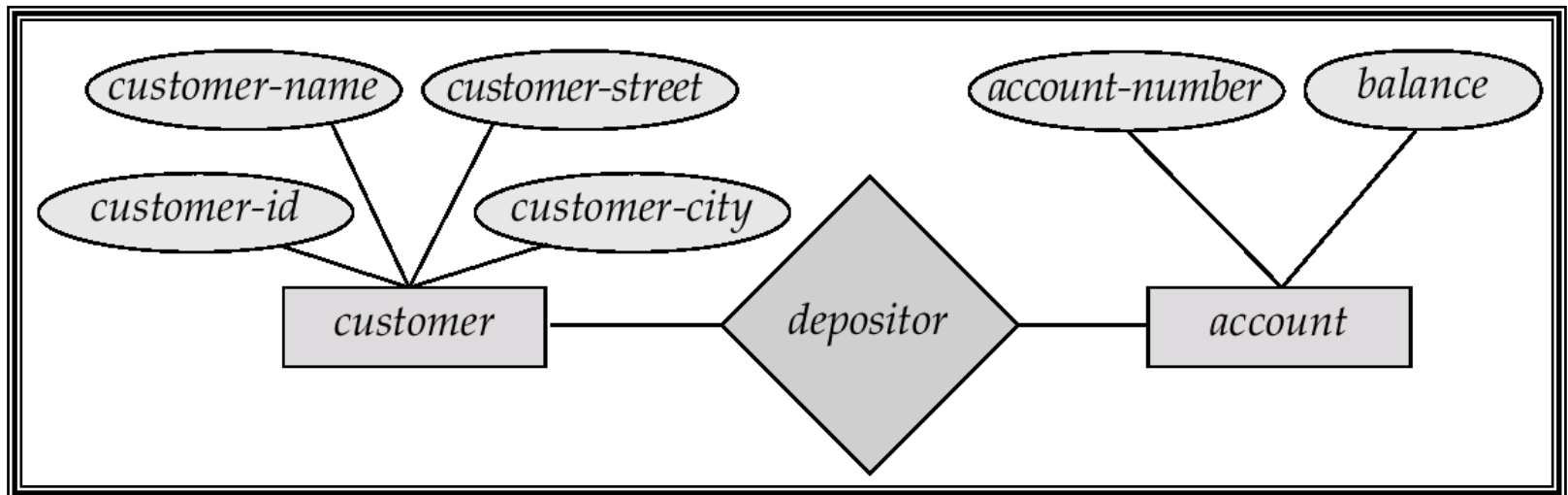


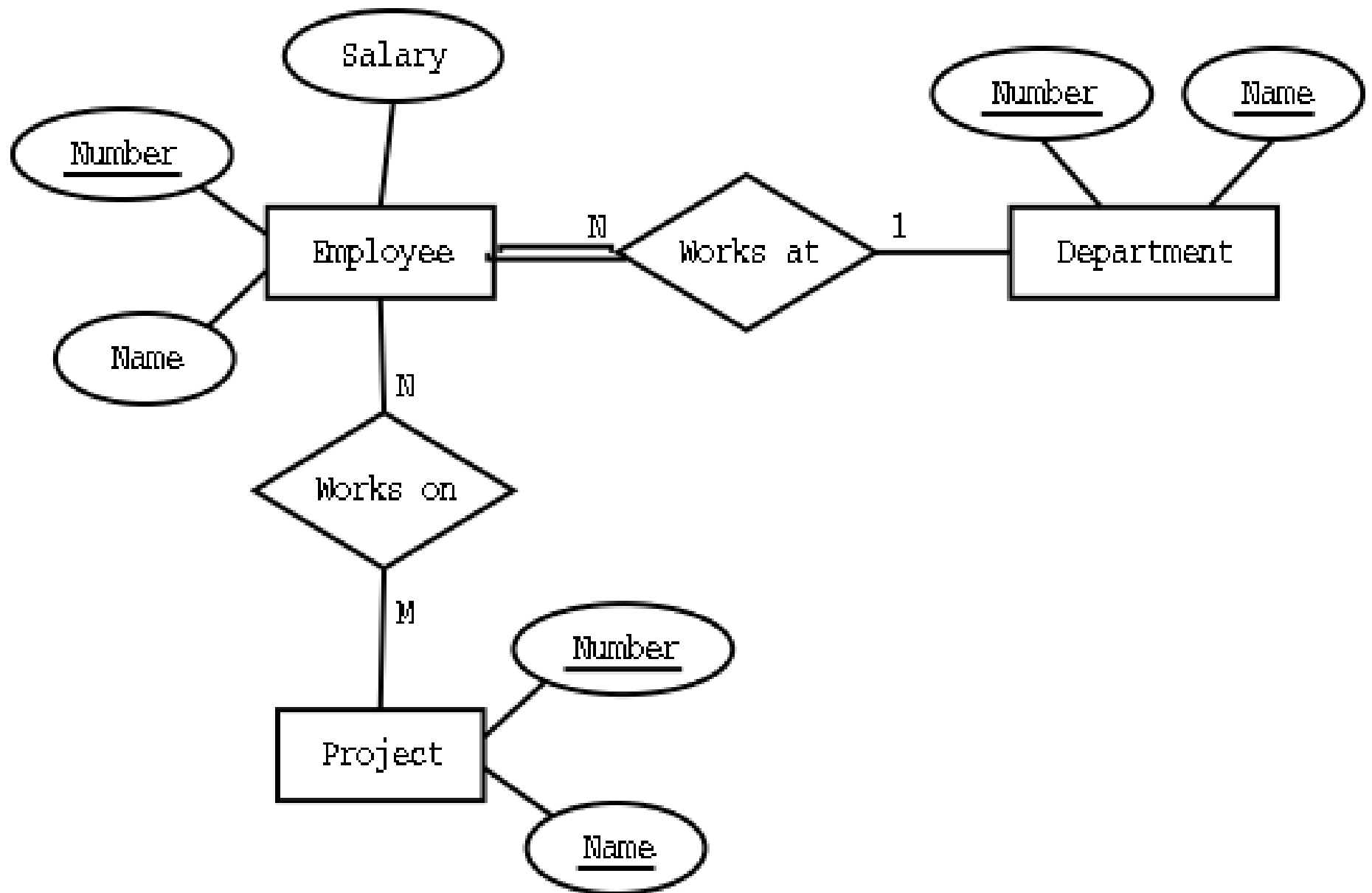
# E-R Model

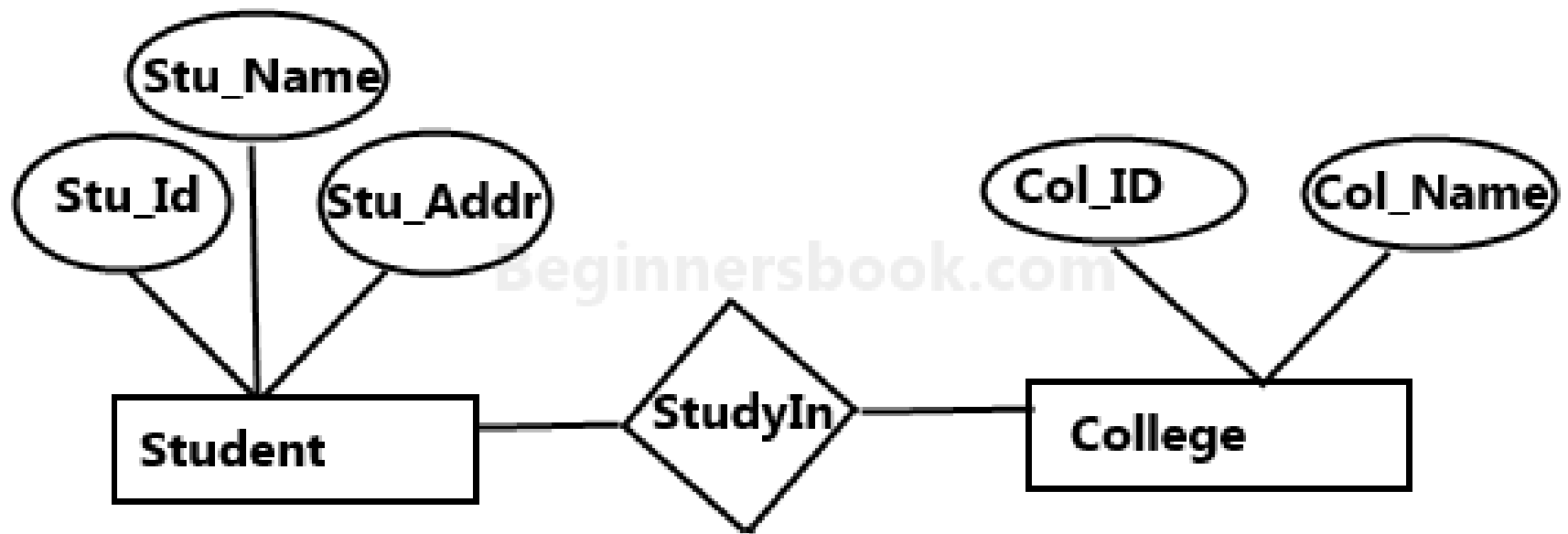
## *Definition -*

- The entity-relationship model (or ER model) is a way of graphically representing the logical relationships of entities (or objects) in order to create a database. The ER model was first proposed by Peter Pin-Shan Chen of Massachusetts Institute of Technology (MIT) in the 1970s

# Entity-Relationship Model







**Sample E-R Diagram**

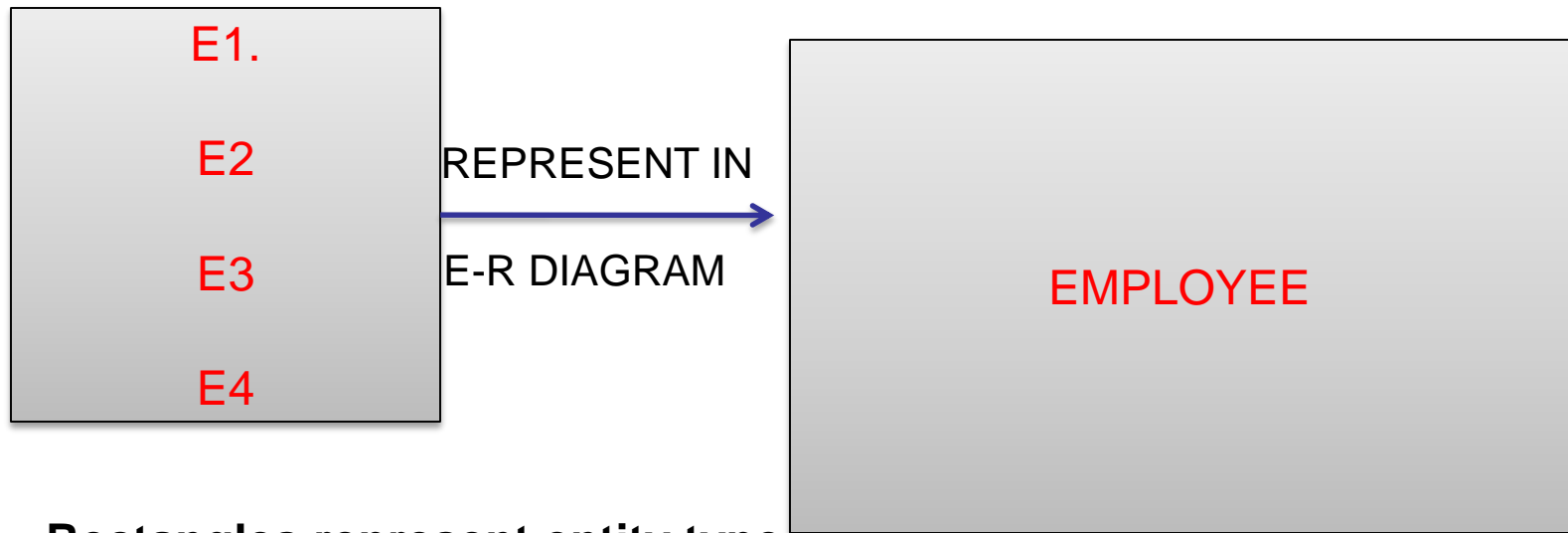
# Features

- It has high degree of data independence
- The ER model is a top-down approach in system design.
- It can be used as a basis for unification views of data such as network, relational modeling.
- It was developed after the relational database when the industry shifted attention to transaction processing.

# Entity

- An entity is an object that exists and is
- distinguishable from other objects.
- Might be
- Object with physical existence like Lect, student, car.
- Object with conceptual or logical existence like course, job, position.

- An **ENTITY SET** is the collection of all entities of a particular entity type in the database at any point of time.
- □ Example:, companies, trees, Employee etc



Rectangles represent entity type

**Entity Type:-** a collection of similar entities

- A set of entities that have the same attributes is called an entity type. Each entity type in the database is described by a name and a list of attributes. For example an entity employee is an entity type that has Name, Age and Salary attributes.
- The individual entities of a particular entity type are grouped into a collection or entity set, which is also called the extension of the entity type. An entity is a thing in the real world. It may be an object with a physical existence or an object with a conceptual existence. A set of these entities having same attributes is entity type and collection of individual entity type is an entity set.



# ENTITY SET corresponding to the ENTITY TYPE CAR

CAR

Registration(RegistrationNumber, State), VehicleID, Make, Model, Year, (Color)

$car_1$

((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 1999, (red, black))

$car_2$

((ABC 123, NEW YORK), WP9872, Nissan 300ZX, 2-door, 2002, (blue))

$car_3$

((VSY 720, TEXAS), TD729, Buick LeSabre, 4-door, 2003, (white, blue))

.

.

.

we can also understand this by an analogy .

entity type is like fruit which is a class .we haven't seen any "fruit"yet though we have seen instance of fruit like "apple ,banana,mango etc.hence..

fruit=entity type=EMPLOYEE

apple=entity=e1 or e2 or e3

entity set= bucket of apple,banana ,mango etc={e1,e2.....}

# Attributes

- **Attributes** are properties used to describe an entity. For example an EMPLOYEE entity may have a Name, SSN, Address, Sex, BirthDate.
- **Attribute Domain**: The set of allowable values for one or more attributes.
- Attributes can be classified as being: *simple* or *composite*; *single-valued* or *multi-valued*; or *derived*.

# Types of Attributes (1)

- **Simple**

- Each entity has a single atomic value for the attribute. For example, SSN or Sex.

- **Composite**

- The attribute may be composed of several components. For example, Address (Apt#, House#, Street, City, State, ZipCode, Country) or Name (FirstName, MiddleName, LastName). Composition may form a hierarchy where some components are themselves composite.

- **Multi-valued**

- An entity may have multiple values for that attribute. For example, Color of a CAR or mobile no.

# Types of Attributes (2)

- **Single values:-** attribute that have only one value for each entity e.g. name, age for employee.
- **Derived:-** attribute contain values that are derived from other attributes e.g. age can be derived from DOB & current date
- **Null Value :-** unknown value e.g. PF no. of employee

**STORED ATTRIBUTE:-** Attributes that are directly stored in the database and can not derived from other one are called stored attributes .

For example, Birth Date attribute of a STUDENT entity

- **STRONG ENTITY SETS**

An entity set containing a key attribute are called strong entity types or regular entity types.

For example, The STUDENT entity has a key attribute Roll No which uniquely identifies it, hence is a strong entity set.

- **WEAK ENTITY SETS**

An entity set may not have sufficient attribute to form a primary key. Entity types that do not contain any key attributes, and hence can not be identified independently are called weak entity sets.

- A weak entity can be identified uniquely only by considering some of its attributes in conjunction with the primary key attribute of another entity, which is called the identifying owner entity

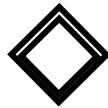
# NOTATION FOR ER SCHEMAS

ENTITY TYPE 




WEAK ENTITY TYPE

RELATIONSHIP TYPE 



IDENTIFYING RELATIONSHIP TYPE

ATTRIBUTE 



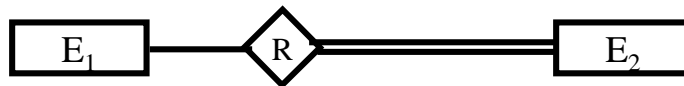
KEY ATTRIBUTE

MULTIVALUED ATTRIBUTE 

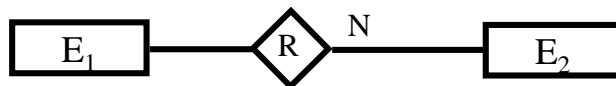


COMPOSITE ATTRIBUTE

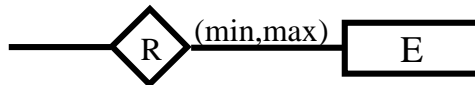
DERIVED ATTRIBUTE 



TOTAL PARTICIPATION OF  $E_2$  IN R



CARDINALITY RATIO 1:N FOR  $E_1:E_2$  IN R



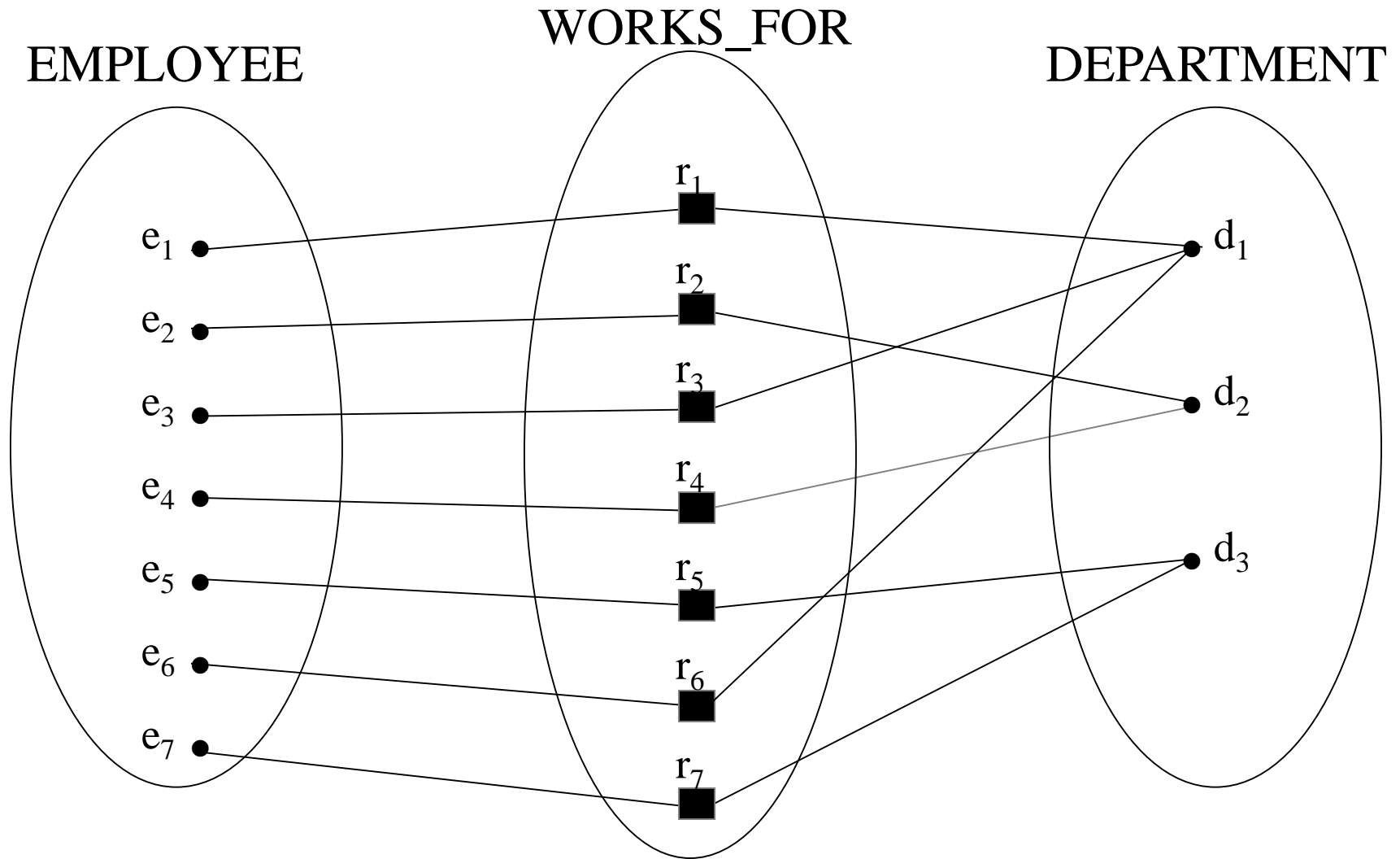
STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

# Relationships and Relationship Types (1)

- A relationship relates two or more distinct entities with a specific meaning. For example, EMPLOYEE John Smith works on the ProductX PROJECT or EMPLOYEE Franklin Wong manages the Research DEPARTMENT.



# Example relationship instances of the WORKS\_FOR relationship between EMPLOYEE and DEPARTMENT



# Degree of a Relationship Type

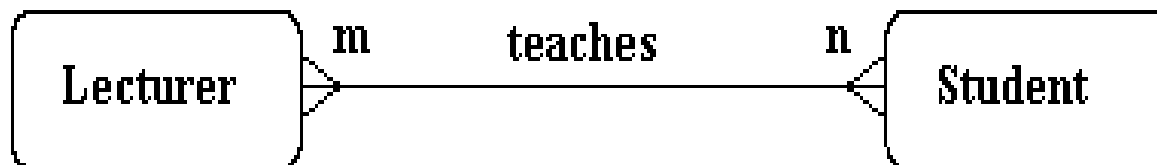
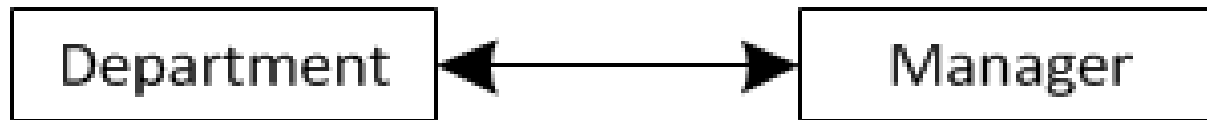
- The degree of a relationship type is the number of participating entity types.
- unary(recursive) relationship type :- that involves only one entity.
- Binary relationship type :-has 2 entity type link together.
- Ternary:- if there are 3 entity type link together.

# Mapping Cardinalities

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many



- **A one to one relationship** -For example, There is only one manager that manages one department, so it is a one to one (1:1) relationship
- **A one to many relationship** - one manager manages many employees, but each employee only has one manager, so it is a one to many (1:n) relationship
- **A many to one relationship** - many students study one course. They do not study more than one course, so it is a many to one (m:1) relationship
- **A many to many relationship** - One lecturer teaches many students and a student is taught by many lecturers, so it is a many to many (m:n) relationship



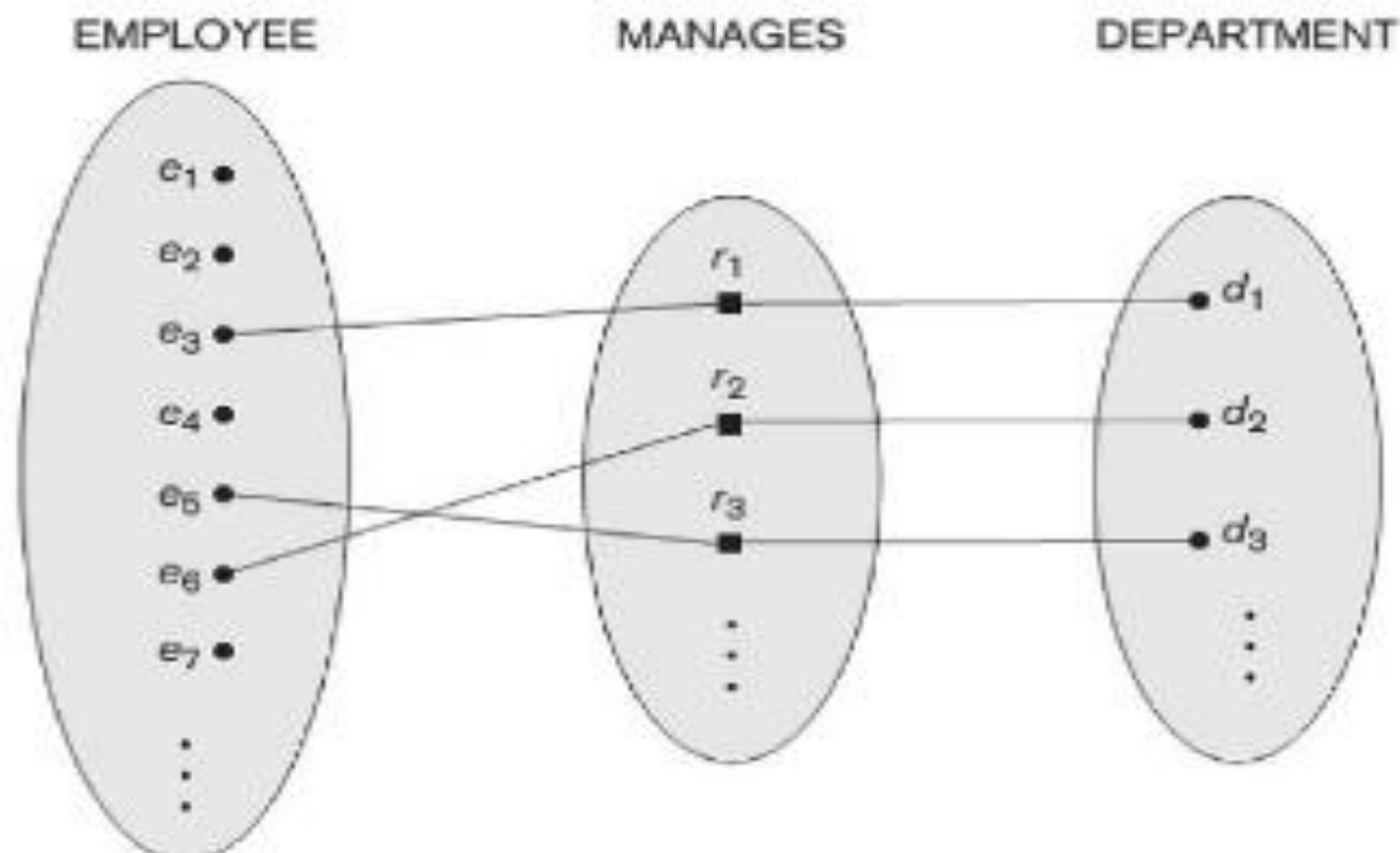
# Constraints on Relationships

- Constraints on Relationship Types  
(Also known as ratio constraints)
  - **Cardinality Ratio (specifies *maximum* participation)**
    - **One-to-one (1:1)**
    - **One-to-many (1:N) or Many-to-one (N:1)**
    - **Many-to-many (M:N)**
  - **Existence Dependency Constraint (specifies *minimum* participation) (also called participation constraint)**
    - **zero (optional participation, not existence-dependent)**
    - **one or more (mandatory participation, existence-dependent)**

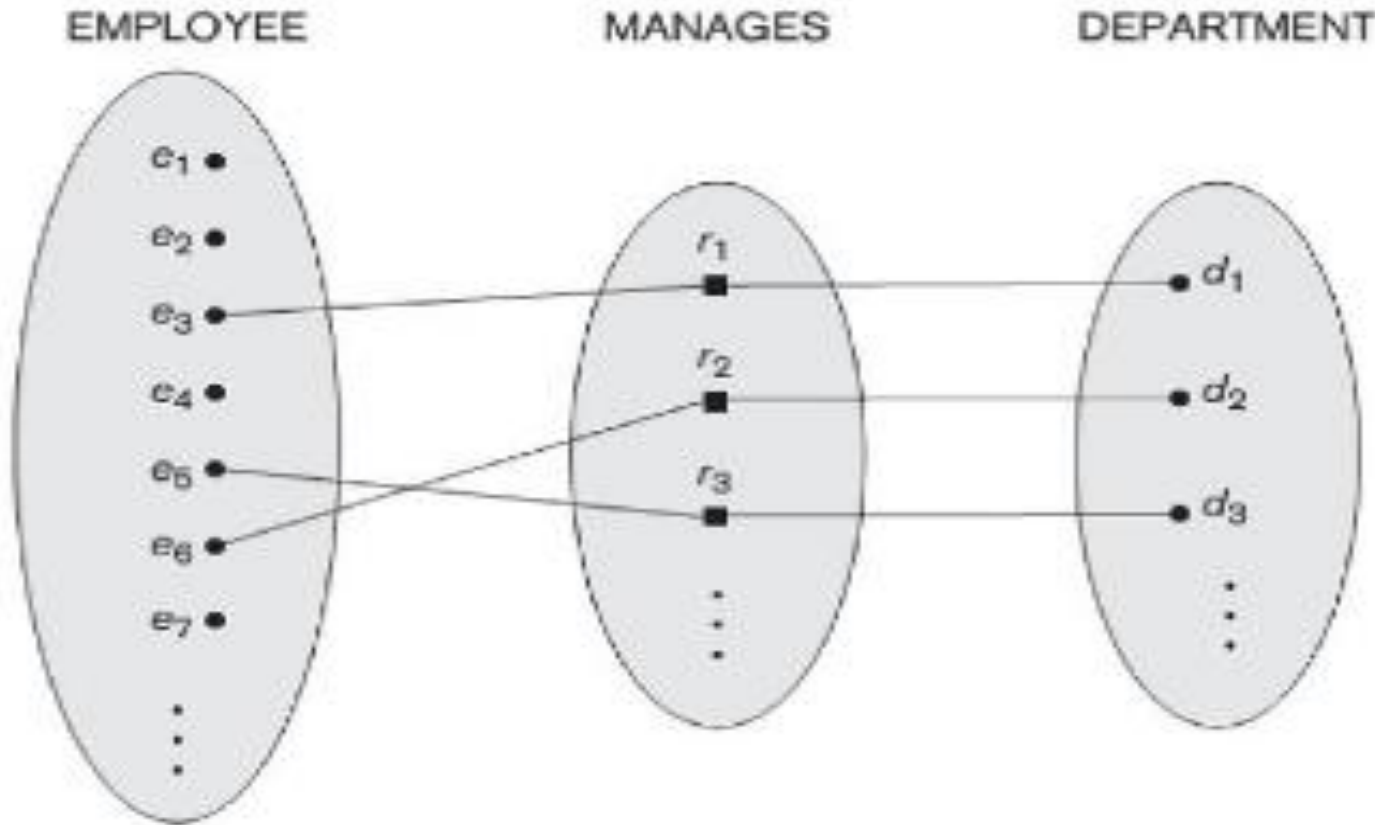
# Example of 1:1 relationship

## Miniworld rules

- Employee can manage one department only
- Department can have one manager only



# Example of 1:1 Relationship (with one Partial and one total Partition)



**Cardinality: 1**

**1**

**Participation: 0**

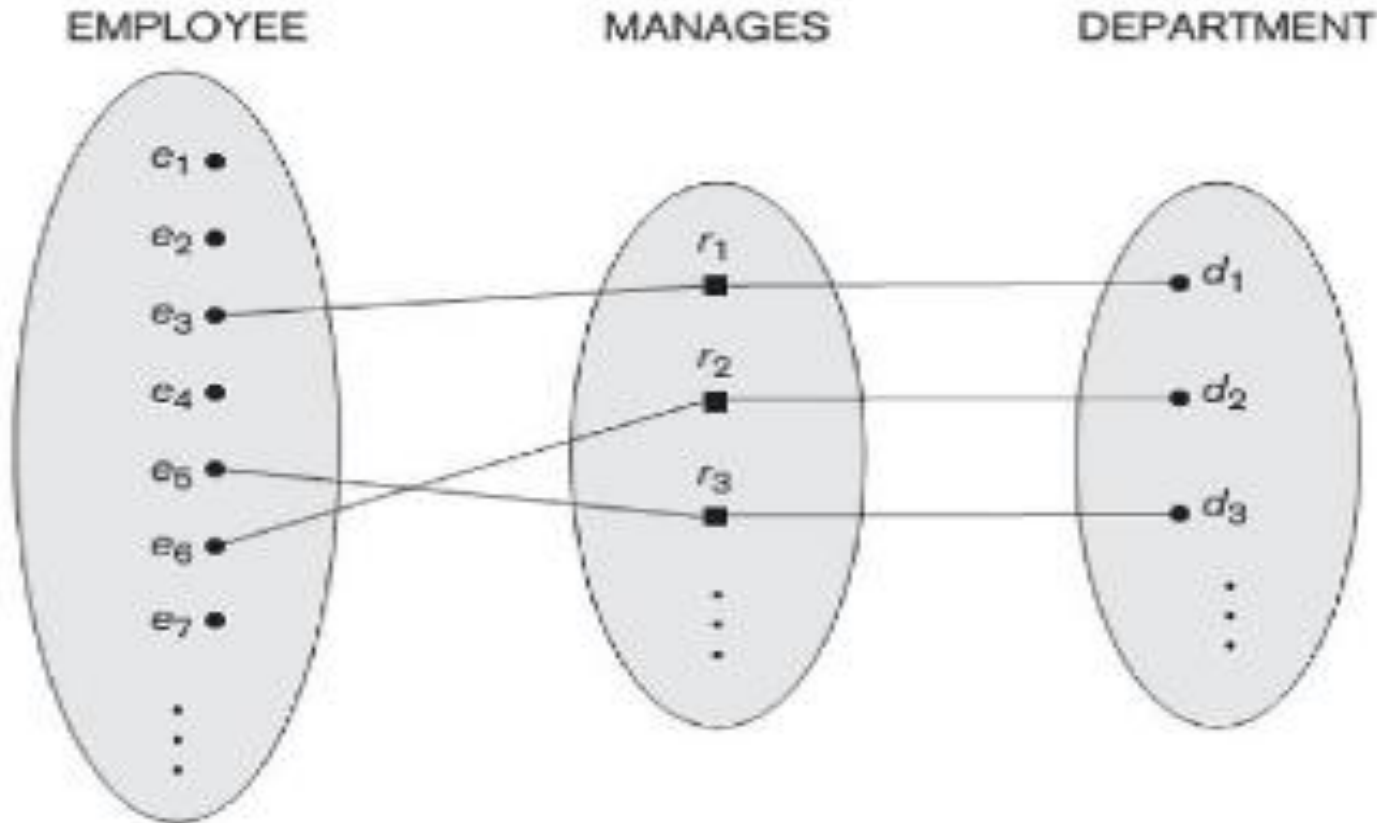
**1**

**This is represented as:**





# Example of 1:1 Relationship with (Min, Max) Representation



**Cardinality: 1**

**1**

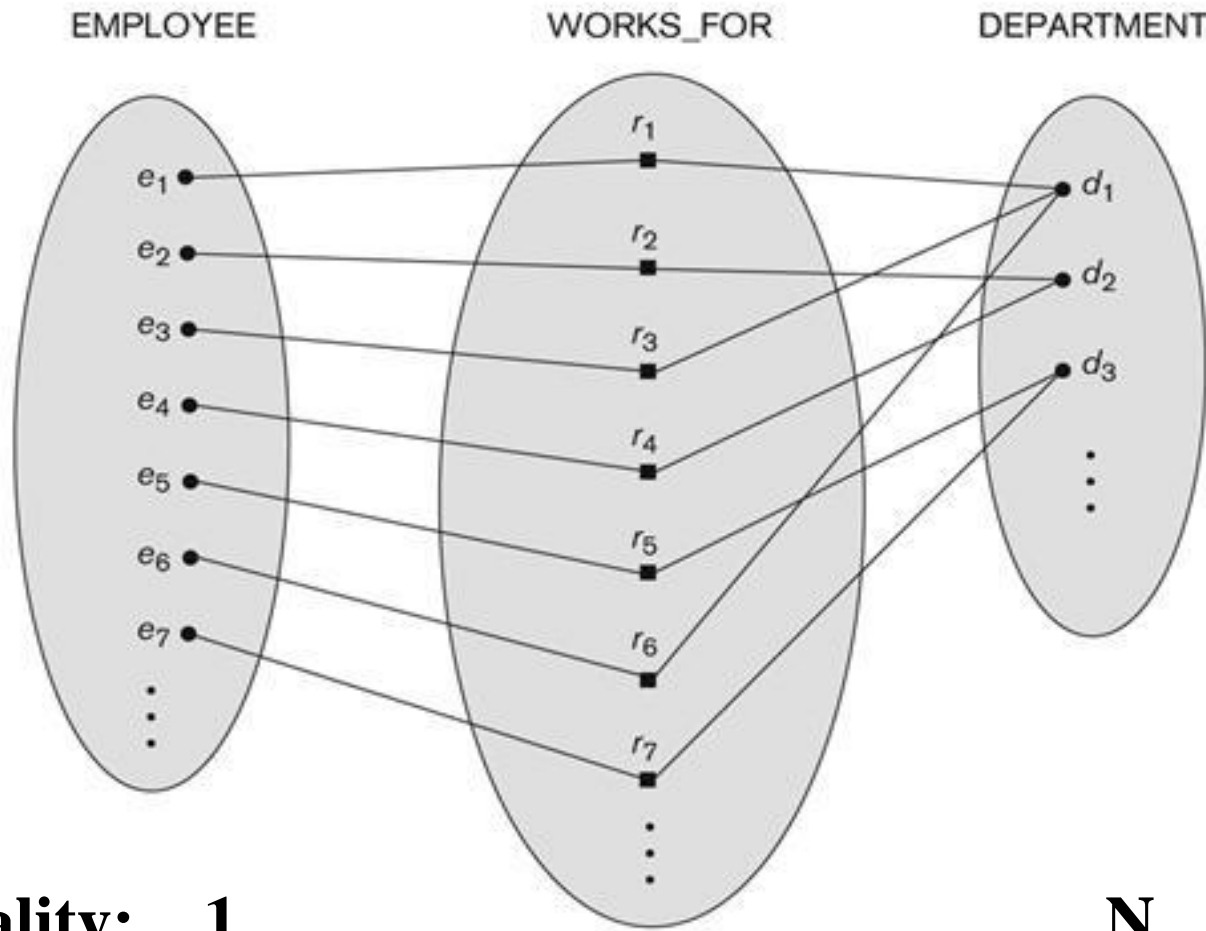
**Participation: 0**

**1**

**This is represented as:**



# Many to 1 Relationship (N:1) or 1 to Many Relationship (1:N)

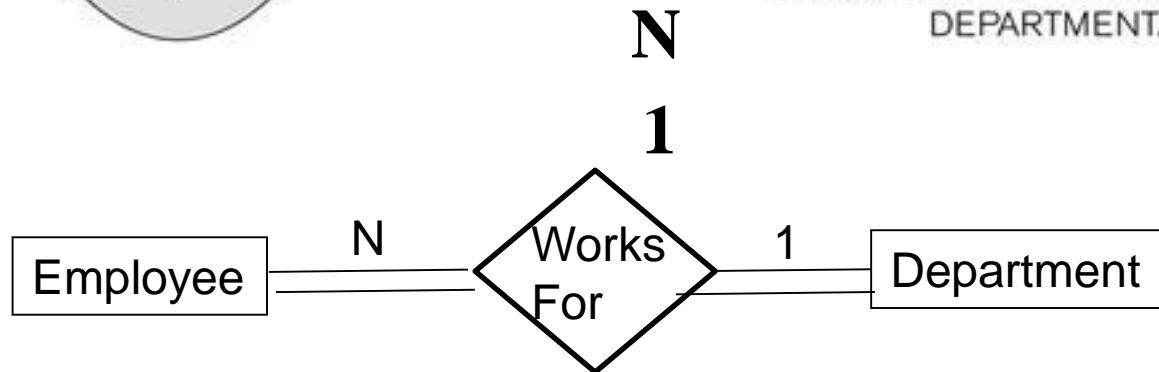


**Figure 3.9**  
Some instances in the WORKS\_FOR relationship set, which represents a relationship type WORKS\_FOR between EMPLOYEE and DEPARTMENT.

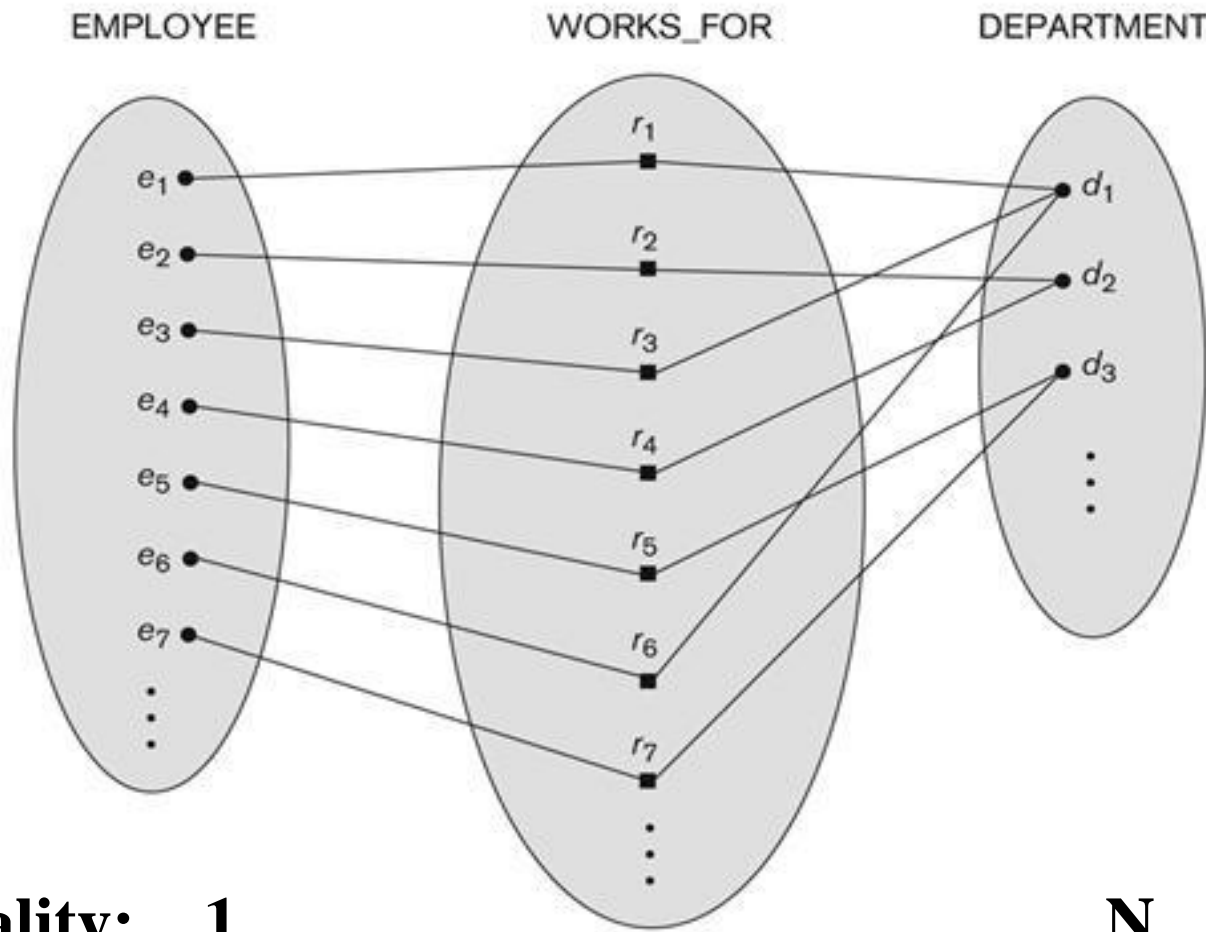
**Cardinality: 1**

**Participation: 1**

**This is represented as:**



# Many to 1 Relationship (N:1) or 1 to Many Relationship (1:N)



with (Min, Max)  
Representation

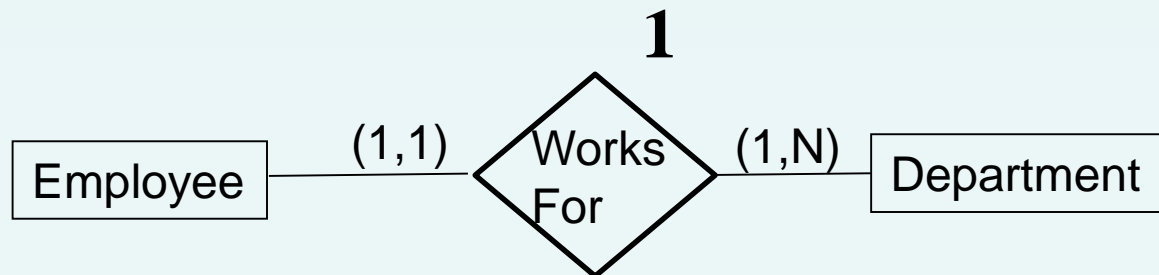
**Figure 3.9**

Some instances in the WORKS\_FOR relationship set, which represents a relationship type WORKS\_FOR between EMPLOYEE and DEPARTMENT.

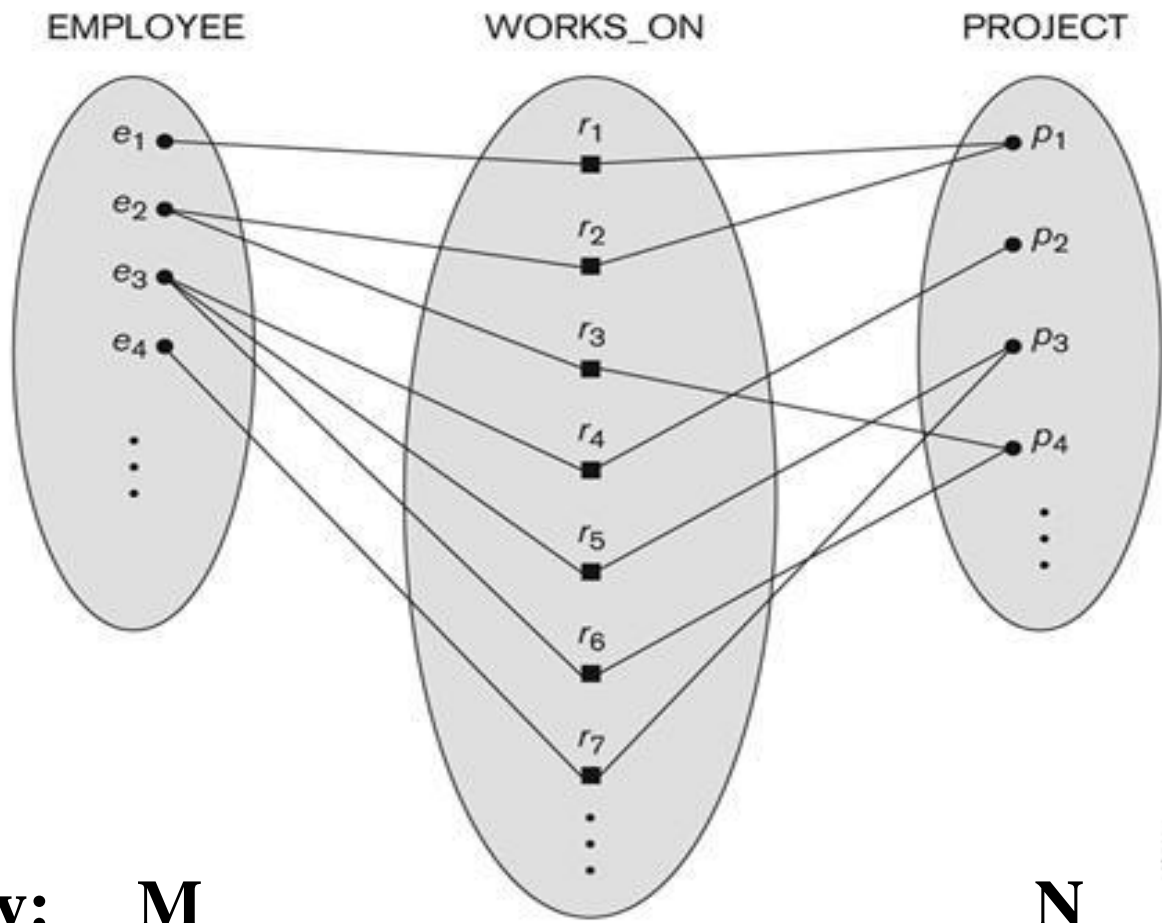
**Cardinality: 1**

**Participation: 1**

**This is represented as:**



# Many to Many Relationship (M:N)

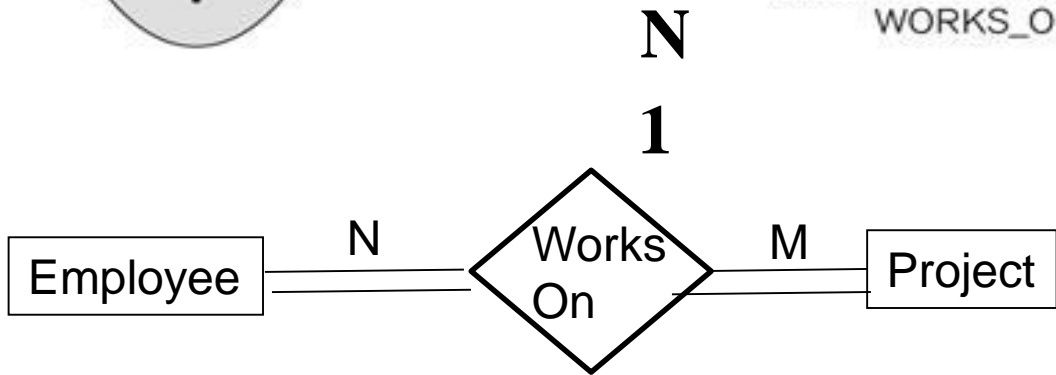


**Figure 3.13**  
An M:N relationship,  
WORKS\_ON.

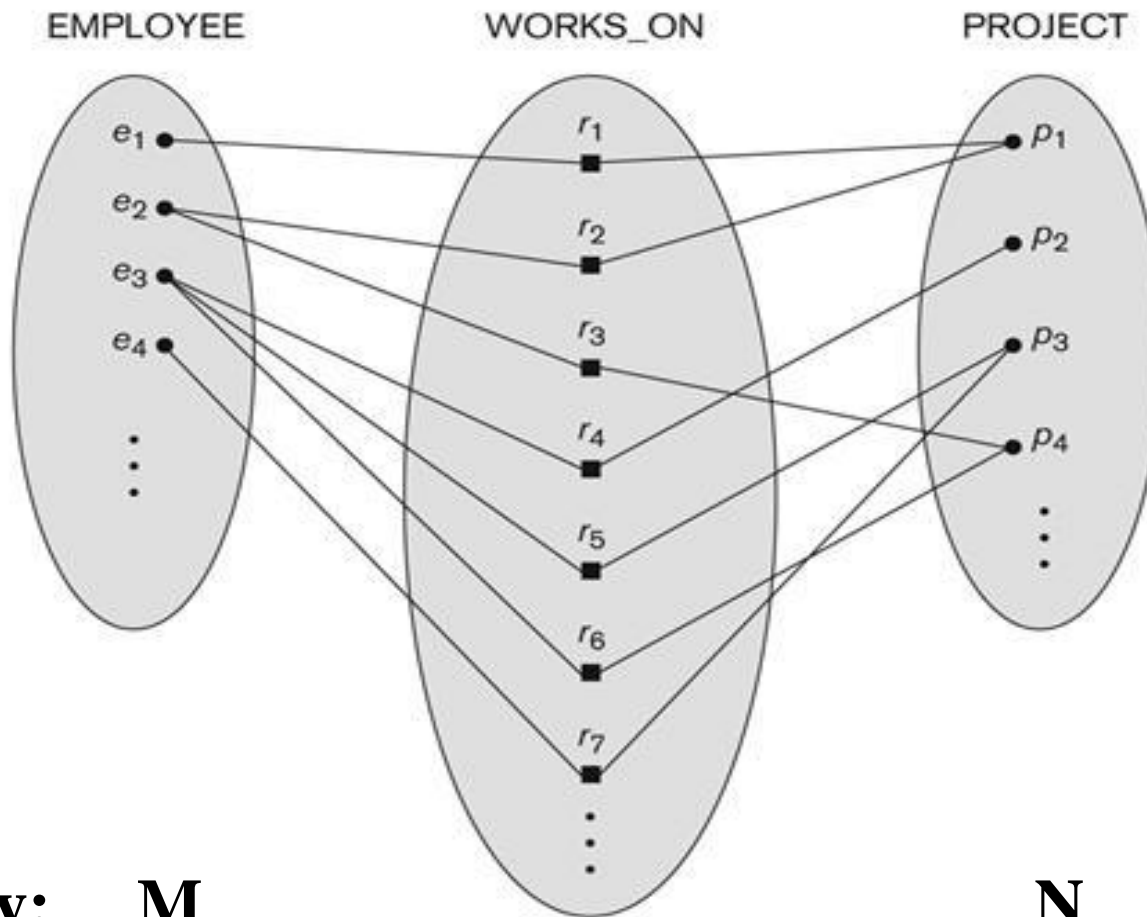
**Cardinality: M**

**Participation: 1**

**This is represented as:**



# Many to Many Relationship (M:N) with (Min, Max) Representation



**Figure 3.13**  
An M:N relationship,  
WORKS\_ON.

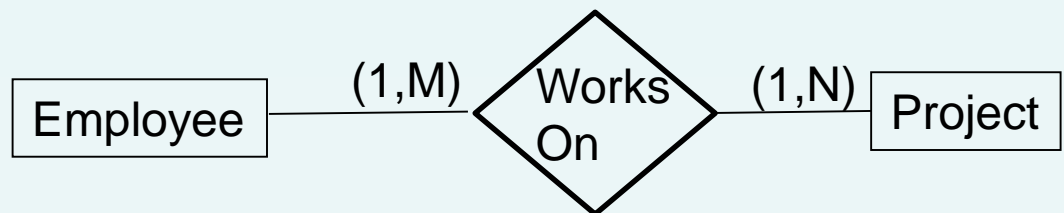
**Cardinality:** M

N

**Participation:** 1

1

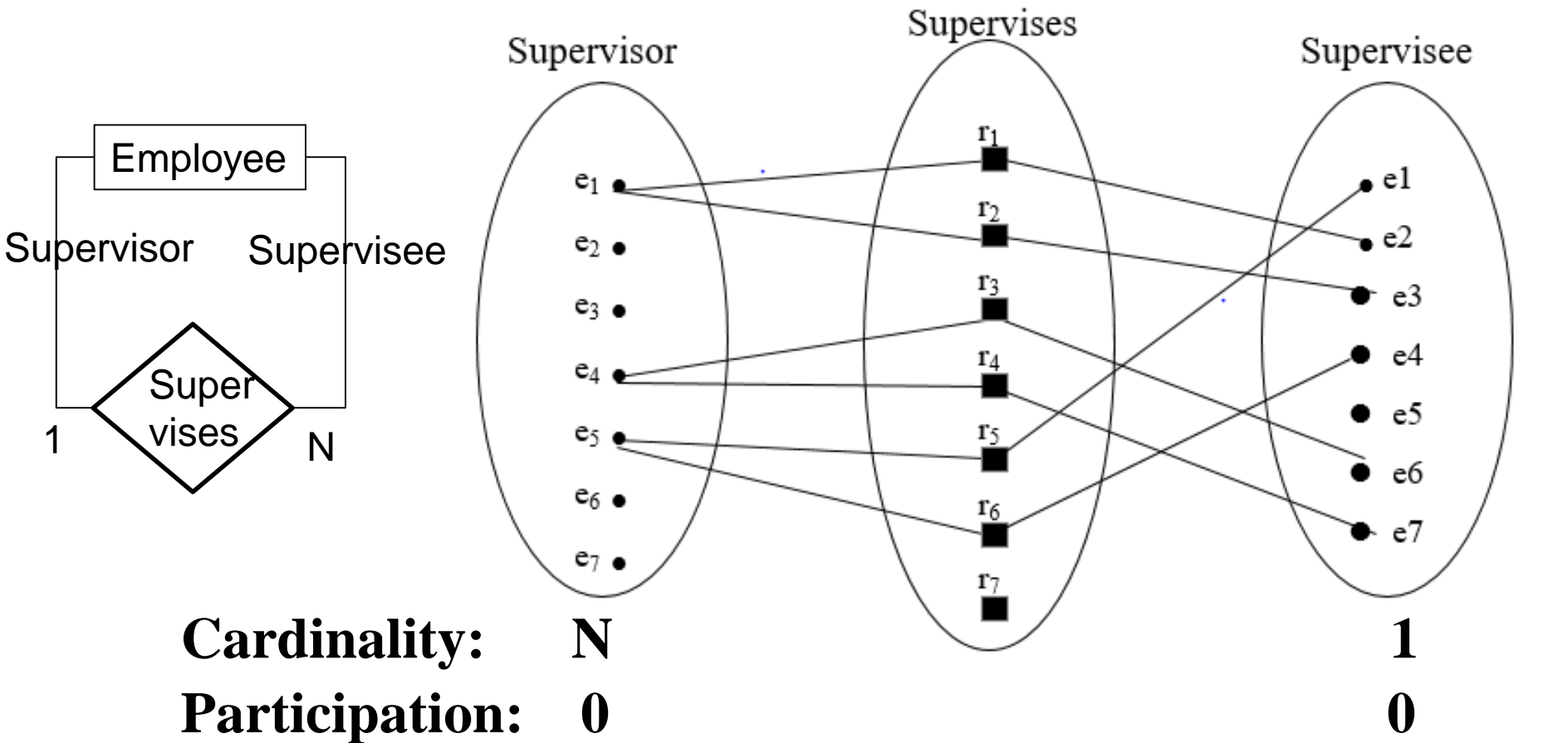
**This is represented as:**



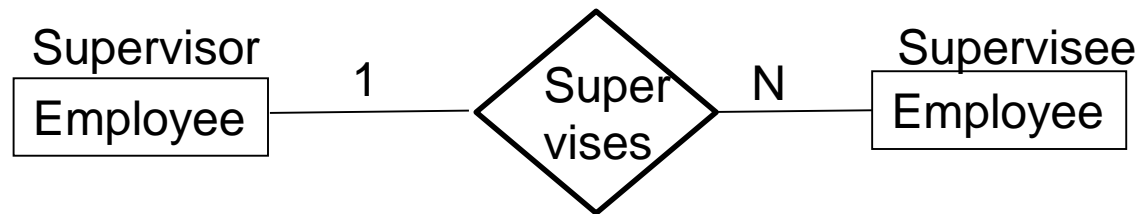
# Displaying a recursive relationship

- In a recursive relationship type.
  - Both participations are same entity type in different roles.
  - For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker).
- In following figure, first role participation labeled with 1 and second role participation labeled with 2.
- In ER diagram, need to display role names to distinguish participations.

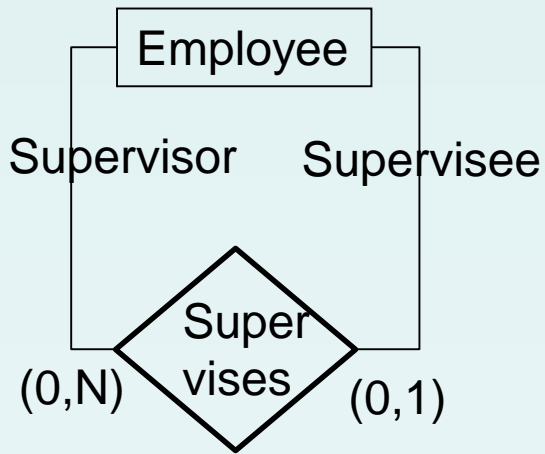
# Example relationship instances of the WORKS\_FOR relationship between EMPLOYEE and DEPARTMENT



**This is represented as:**

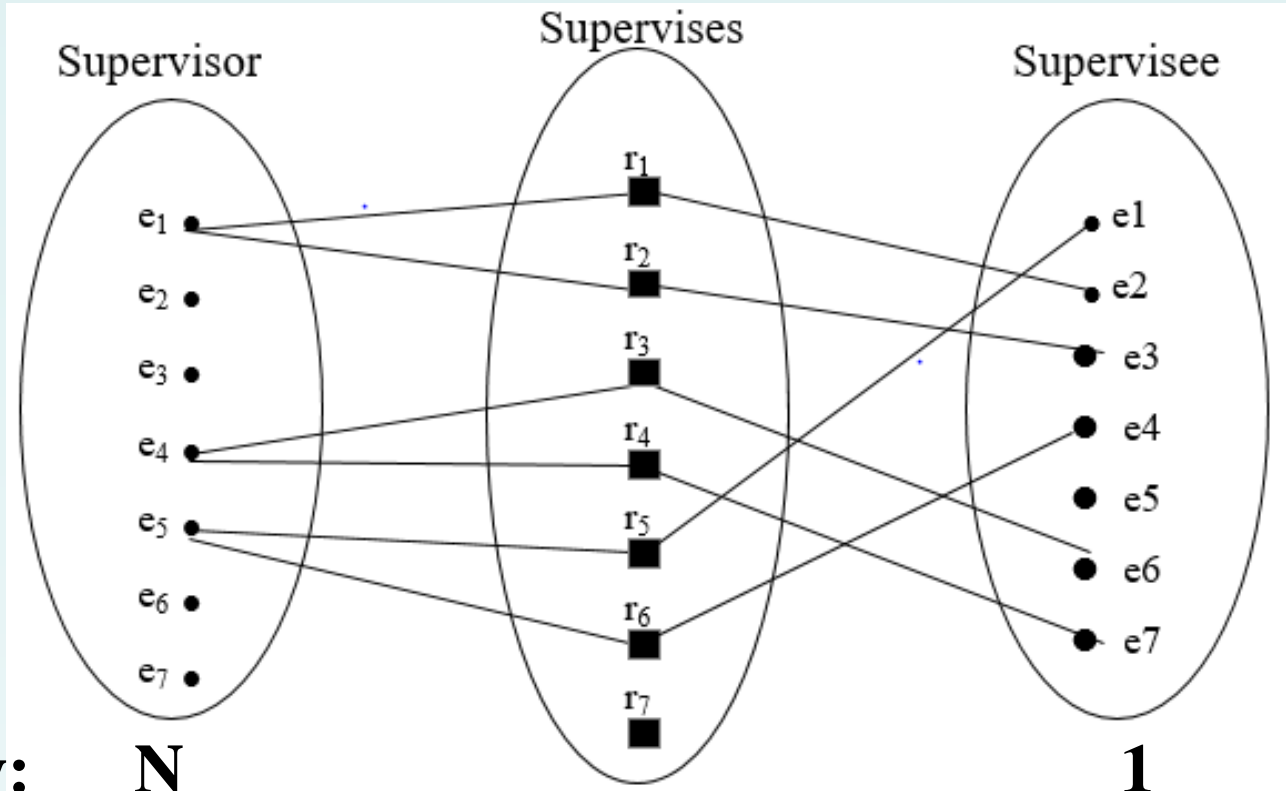


# Example relationship instances of the WORKS\_FOR relationship between EMPLOYEE and DEPARTMENT with (Min, Max) Representation



**Cardinality: N**

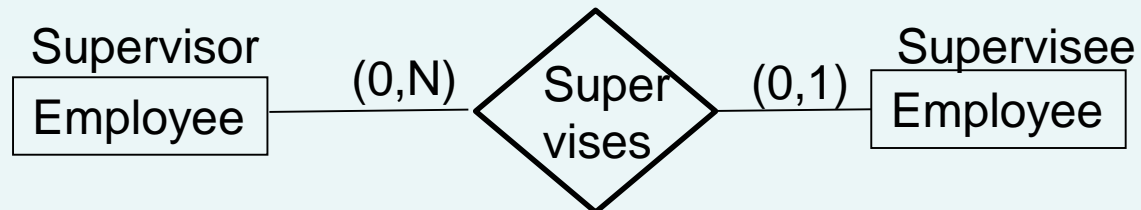
**Participation: 0**



**1**

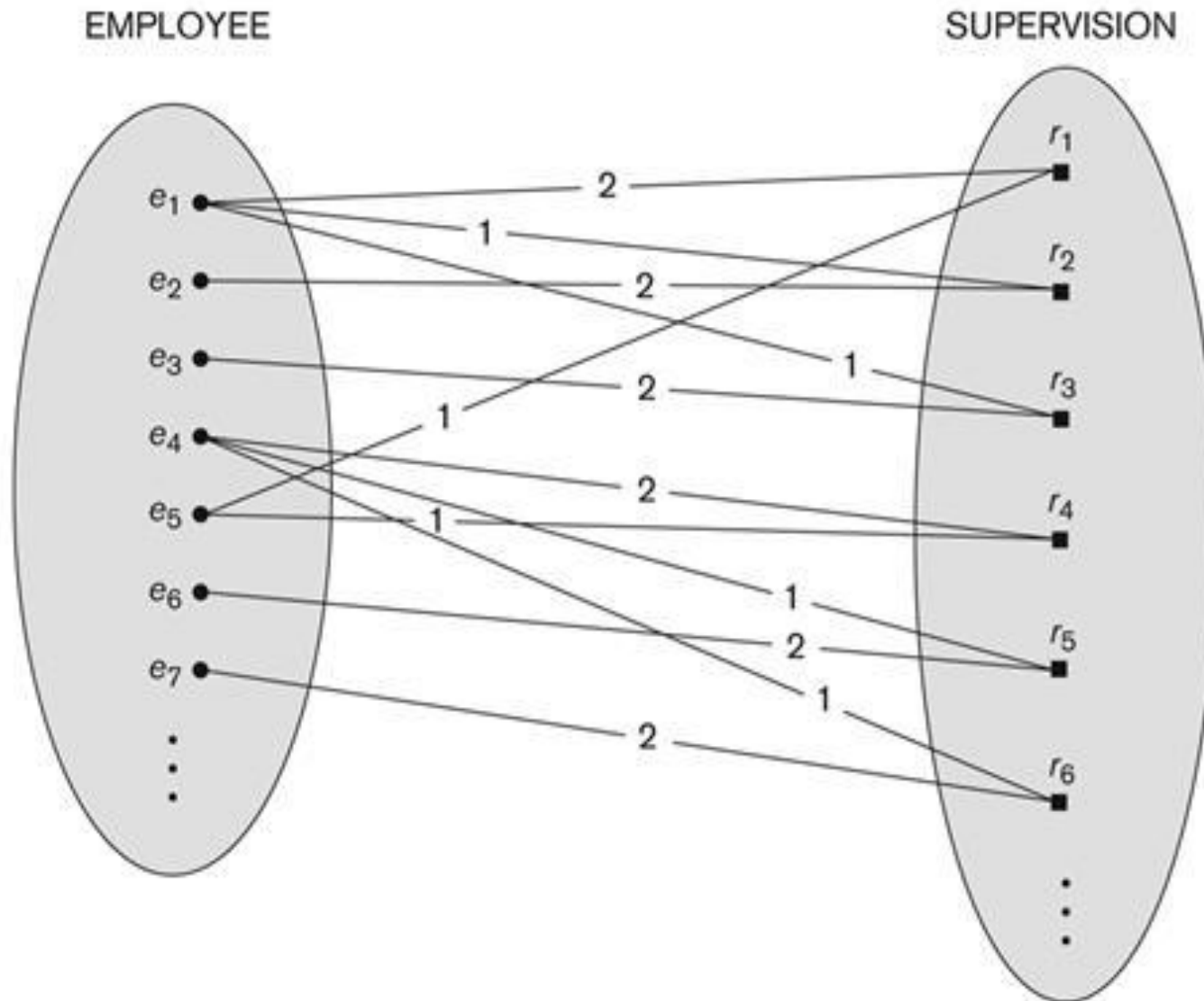
**0**

**This is represented as:**





## Recursive Relationship

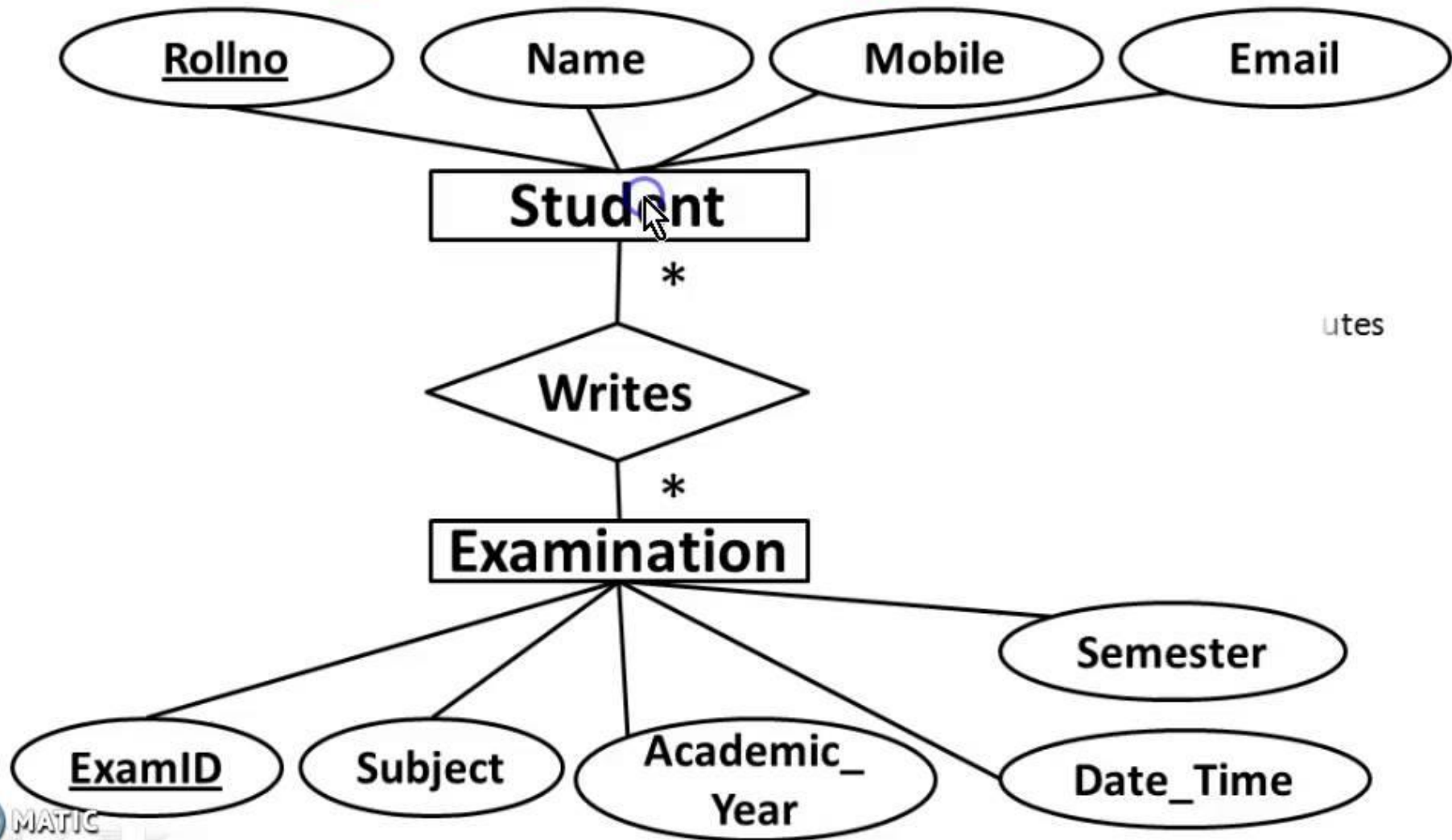


**Figure 3.11**

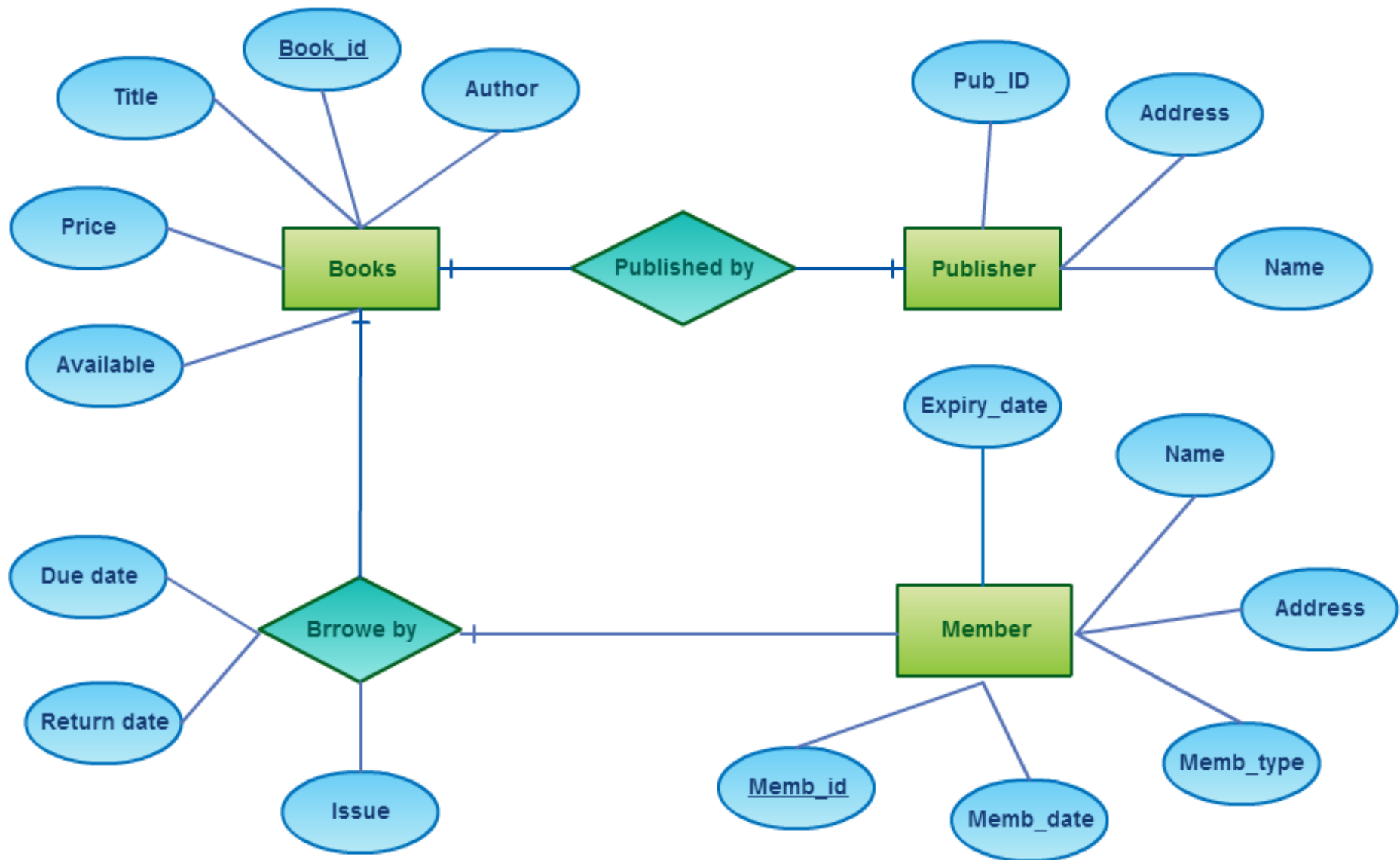
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

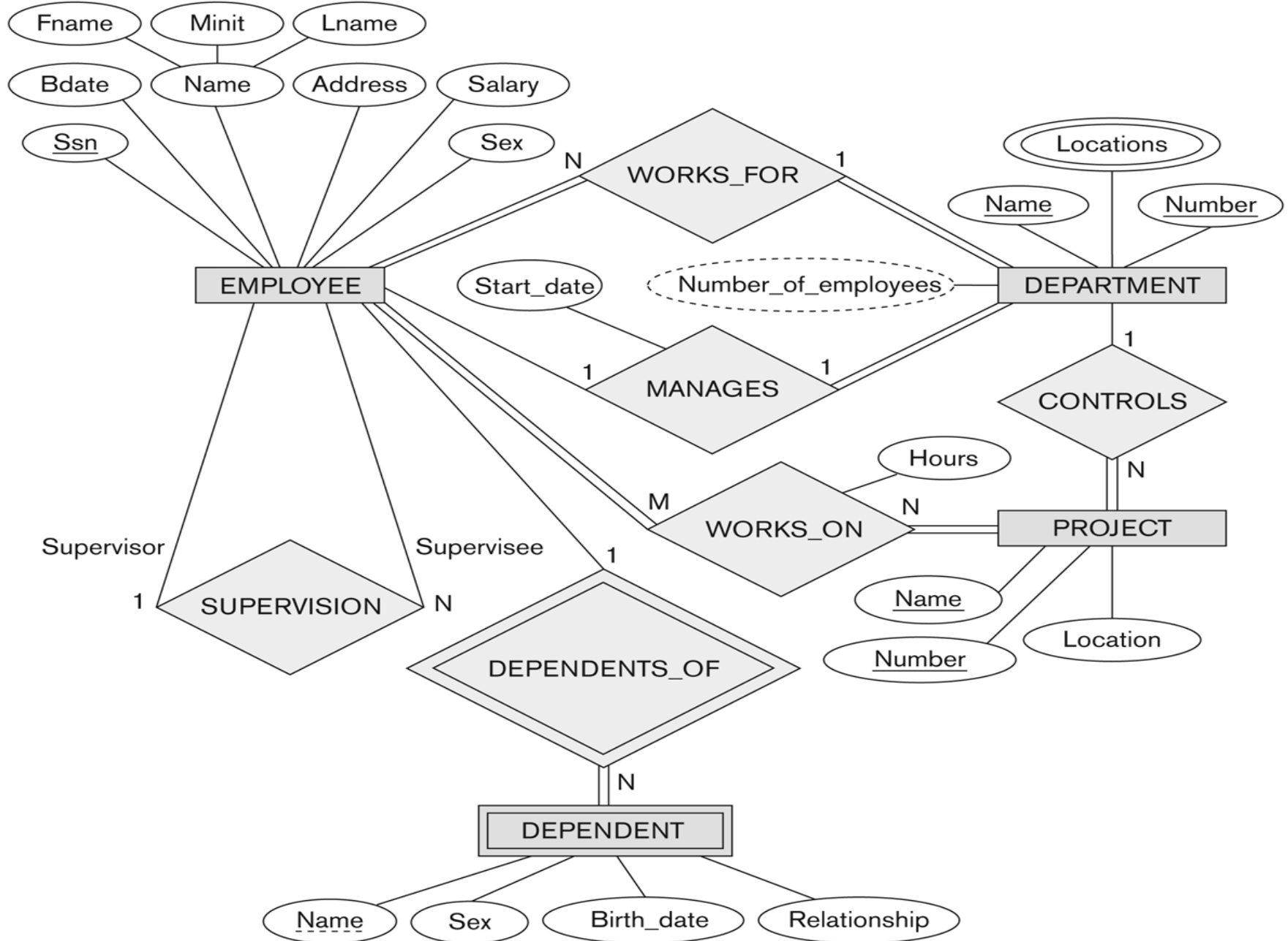
# Example 1

- Students write examinations



# E-R Diagram of Library Management System



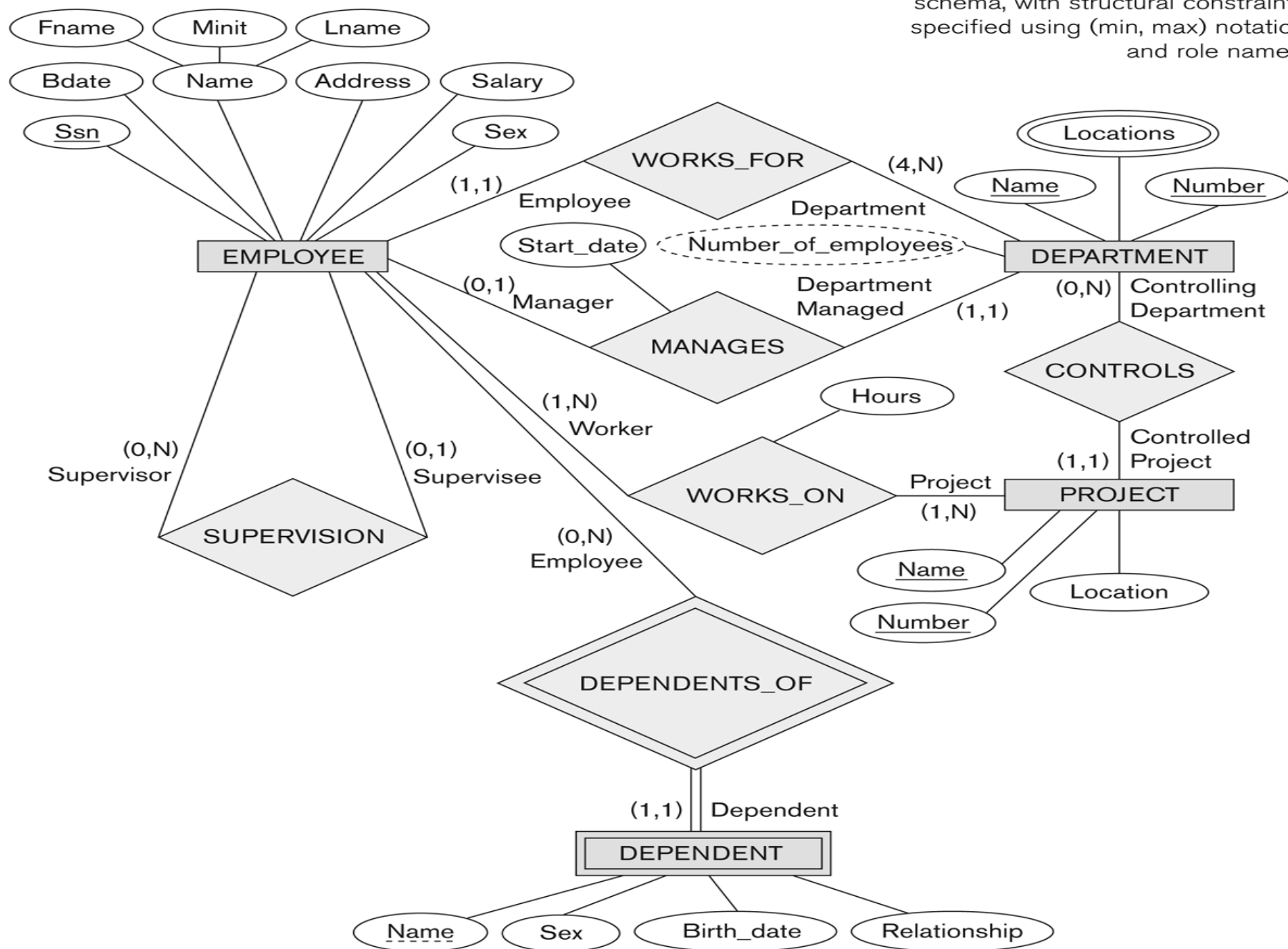


**Figure 3.2**

An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

**Figure 3.15**

ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

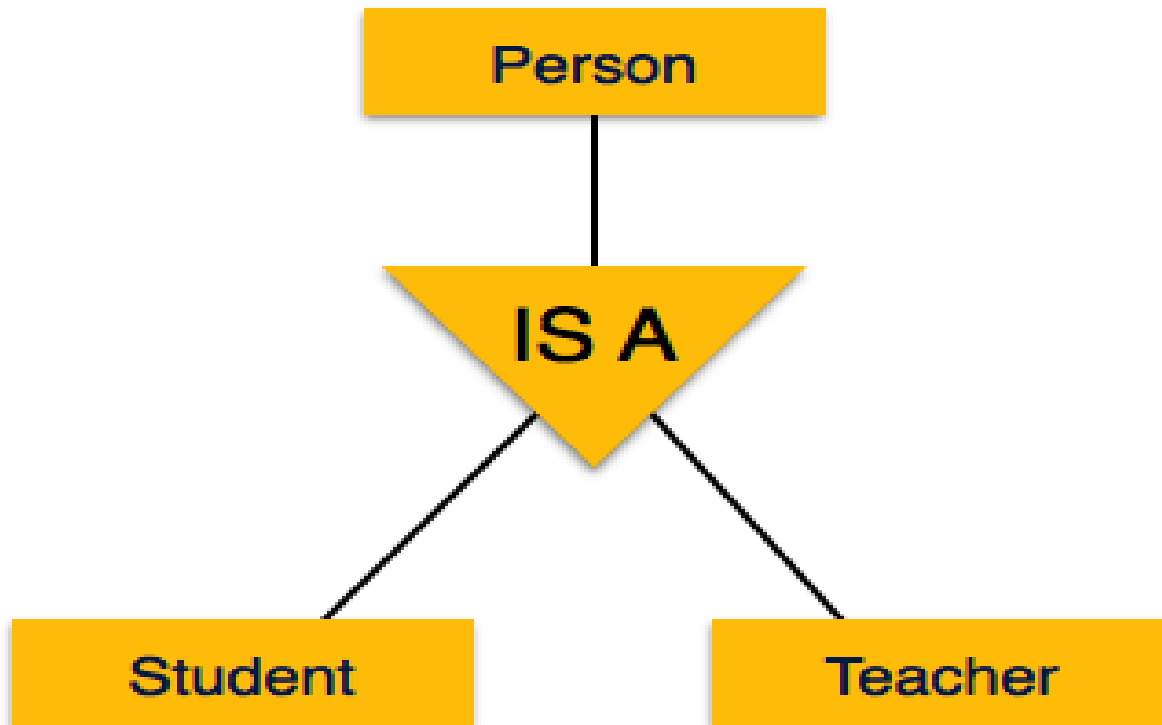


# Extended(Enhanced ) ER Model:

- The ER modeling concepts are sufficient for representing traditional database application. For more complex database application such as telecommunications, CAD/CAM, GIS etc, we need more complex requirements than traditional applications.
- In late **1970's database designers have tried to design more accurate ER model, which reflects the data properties and constraints more accurately.** So extended (Enhanced) ER model have some enhanced features than normal ER model. **It uses the concepts of Specialization, Generalization and Aggregation.**

# Specialization:

- Specialization is the process through which we can relate one entity to more than one entity. In other words specialization is the process to defining a set of subclasses of an entity type.
- **This entity is called superclass. For ex. An entity “employee” has the sub entity, ‘faculty’, ‘Staff’. So employee has the relation with both the sub entity. This relation name is “IS A”. as shown in fig. So Specialization follow the process of one to many relationship.**

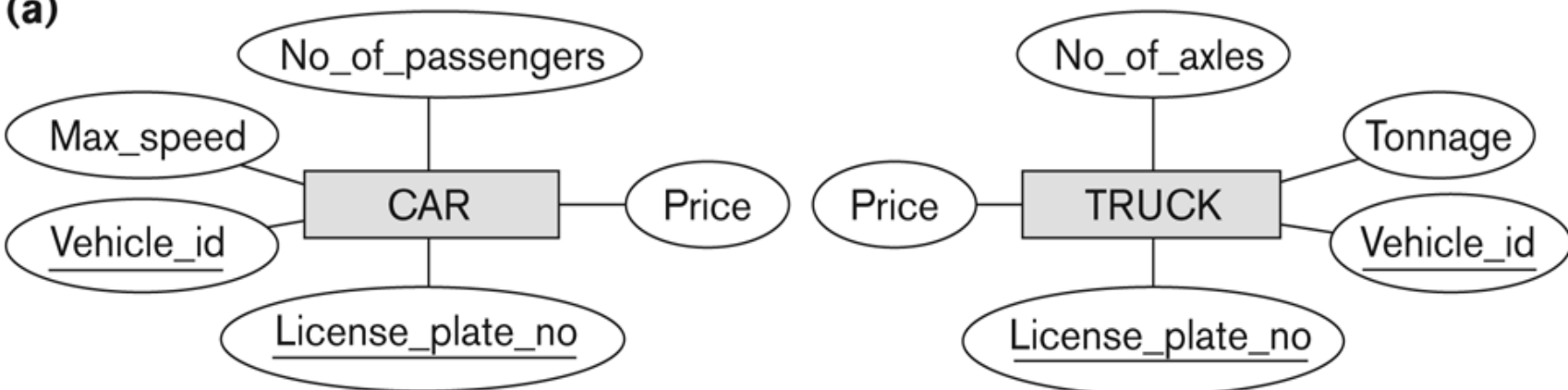




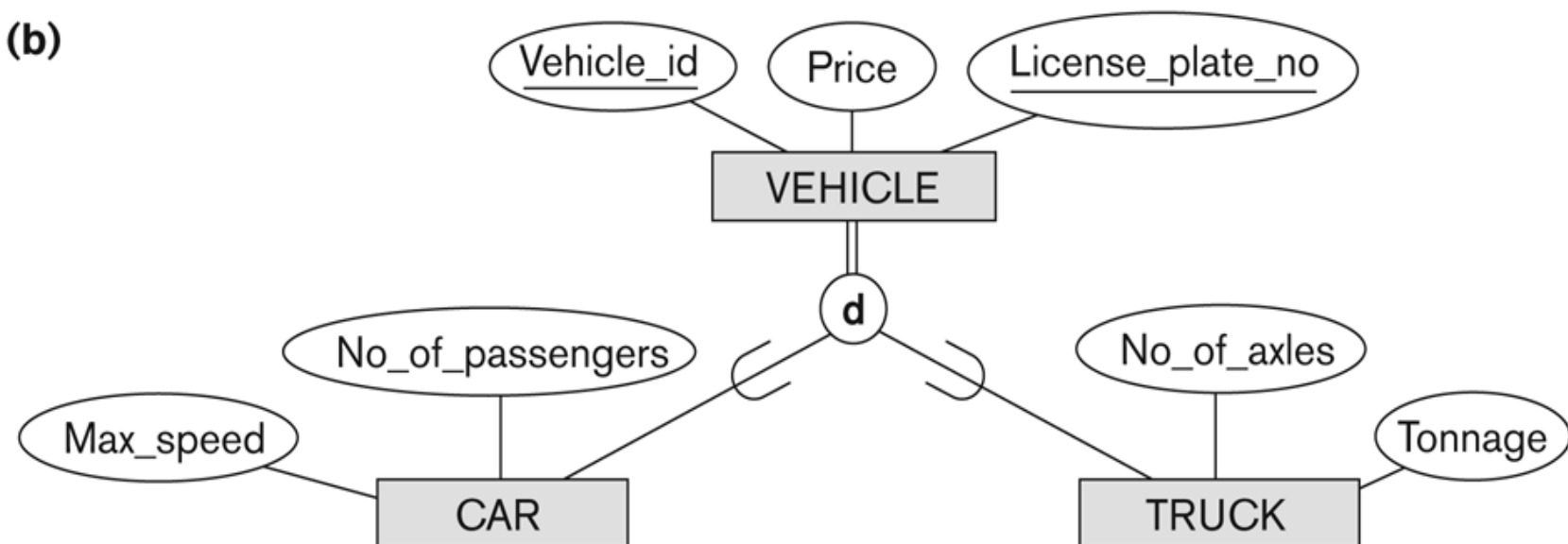
# Generalization:

- Generalization is just reverse of Specialization. Generalization is the process to define a generalized entity type from the given entity type.
- For ex. Consider the two entity CAR and TRUCK. Because both have some common attributes, they can be combined to make a super entity called VEHICLE.
- So it is the process to identify the common features (attributes) from two or more entities and generalized them into a super entity.
- Several classes with common features are generalized into a superclass and original classes become its subclasses

(a)



(b)



**Figure 4.3**

Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Animal

Bottom Up Approach



Tiger

Lion

Elephant

Account

Bottom Up Approach



Saving

Current

Name

Age

Gender

Person



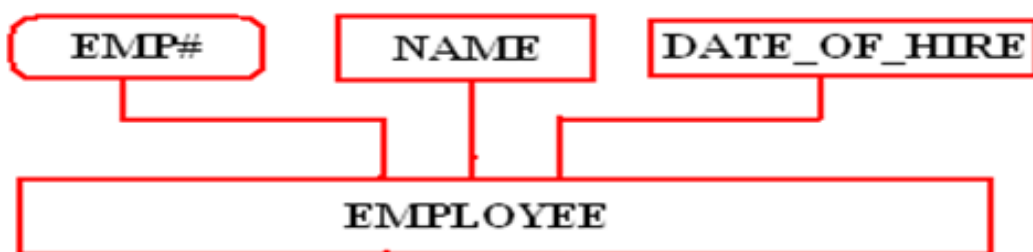
Student

Teacher

RollNo.

EmpID

GENERALIZATION



IS\_A

SPECIALIZATION



SALARY

FULL TIME\_  
EMPLOYEE

TYPE

PART TIME\_  
EMPLOYEE

IS\_A

IS\_A

FACULTY

STAFF

TEACHING

CASUAL

DEGREE

SUBJECT\_OF\_  
INTEREST

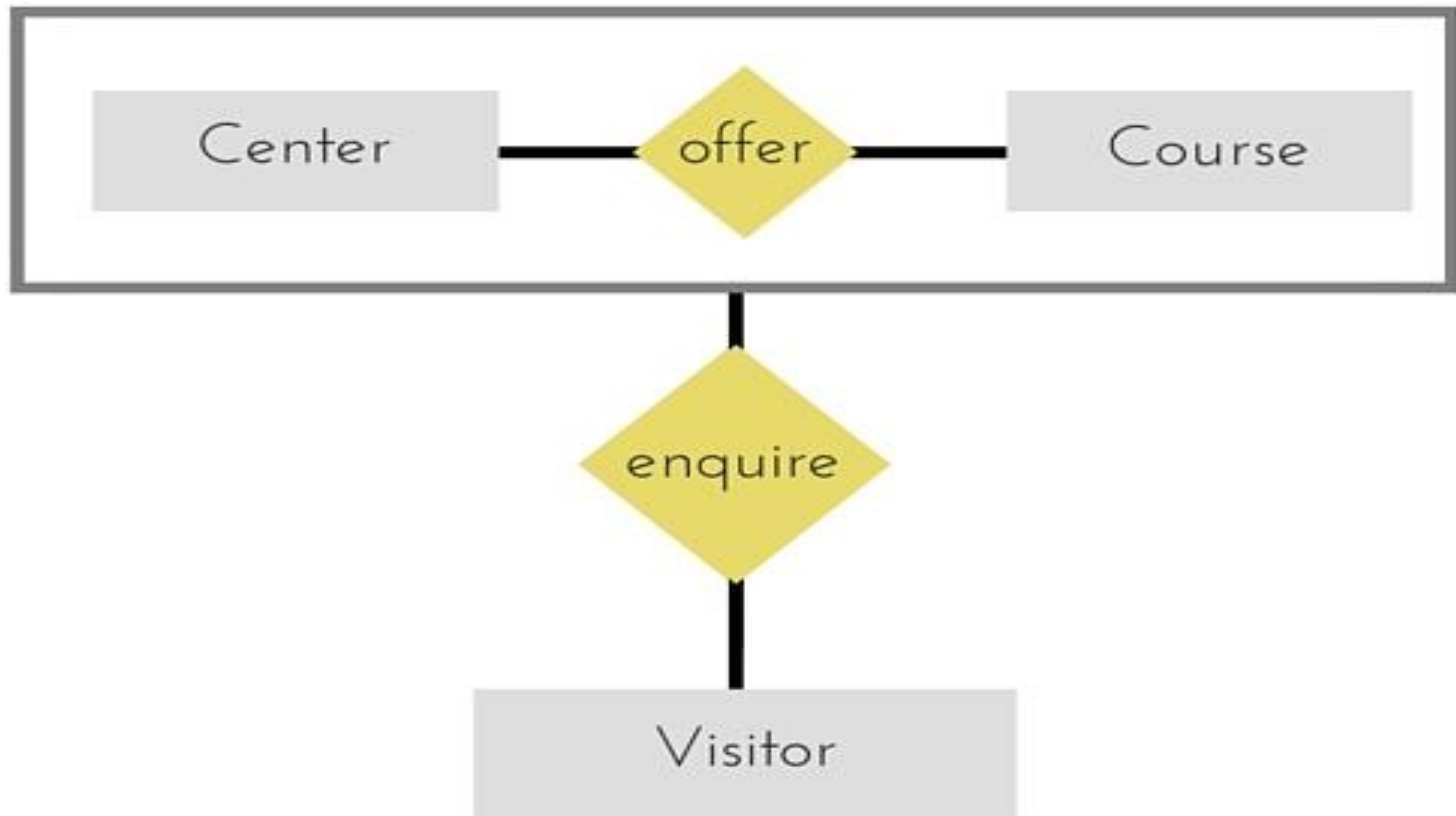
CLASSIFICATION

STIPEND

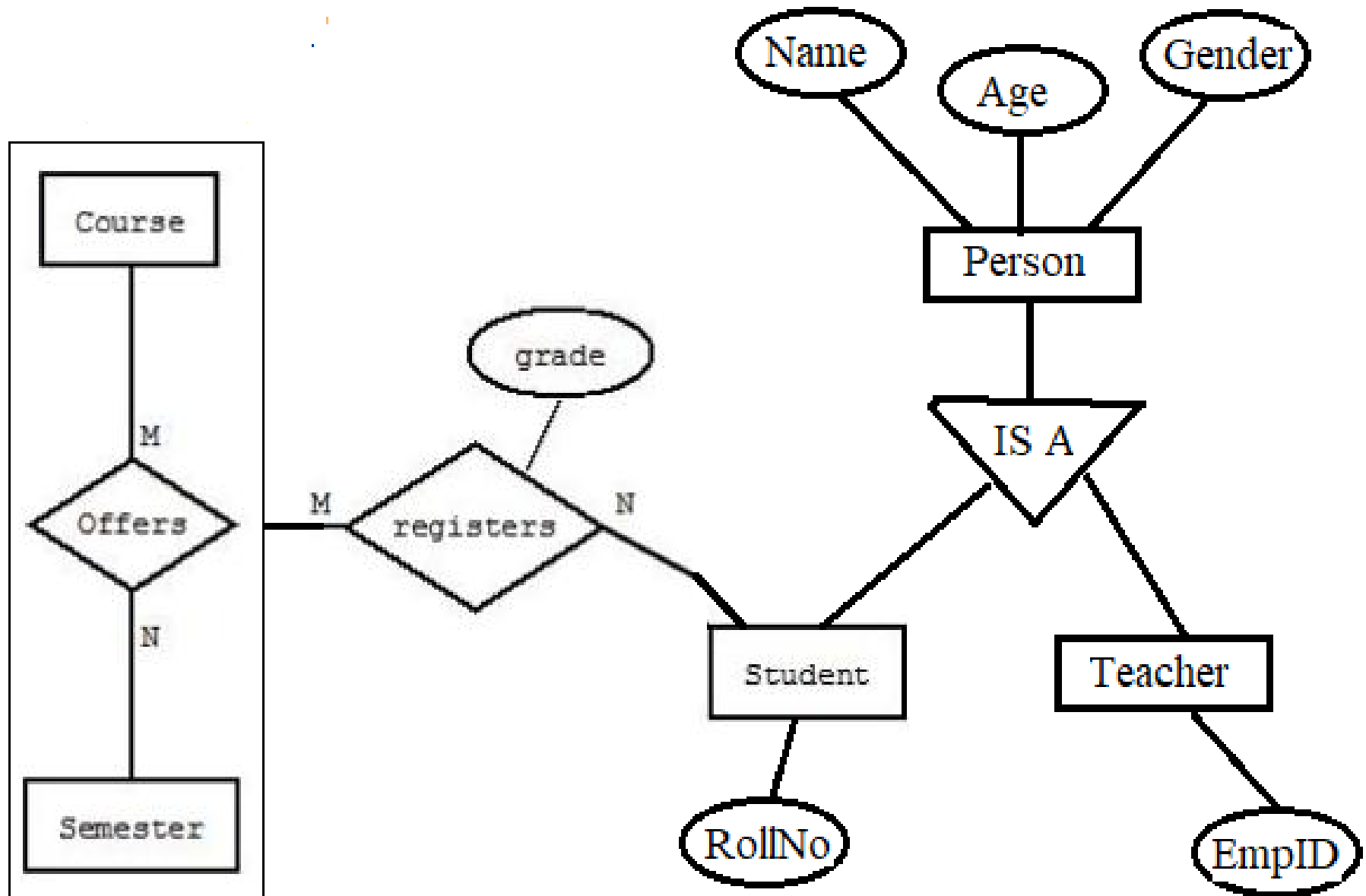
HOUR RATE

# Aggregation:

- Aggregation is a process when relation between two entity is treated as a single entity. Here the relation between Center and Course, is acting as an Entity in relation with Visitor.



## Extended ER Diagram Example



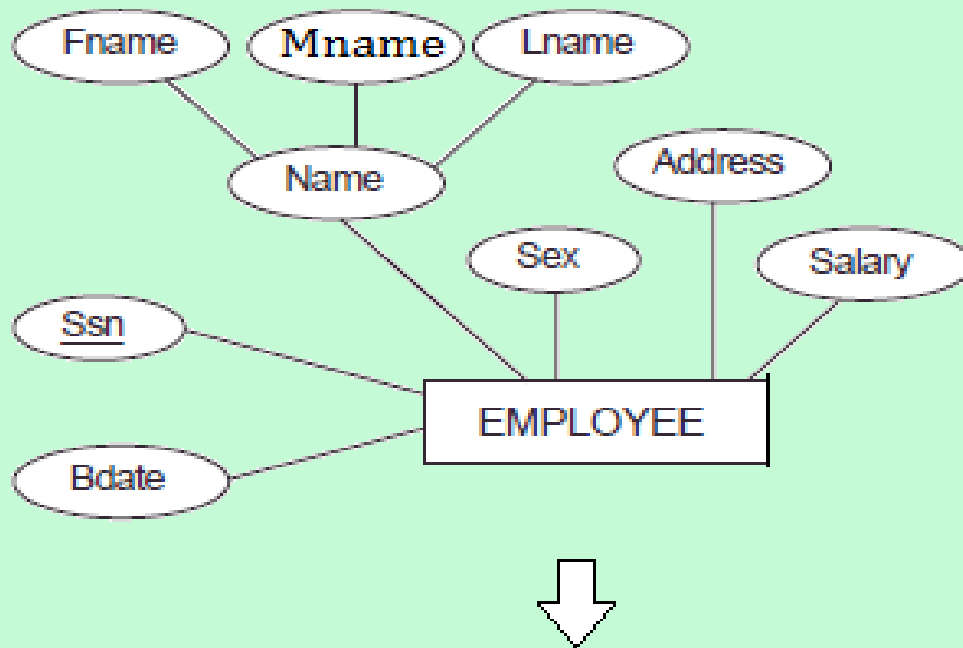
# **ER Model to Relational Model – Conversion**

To convert the ER model to relational model there are 7 Steps to be followed, which are:

- 1. Conversion of Strong Entities**
- 2. Conversion of Weak Entities**
- 3. Conversion of one to one Relationships**
- 4. Conversion of One to Many Relationships**
- 5. Conversion of Many to Many Relationships**
- 6. Conversion of n-ary Relationships**
- 7. Conversion of Multivalued Attribute**

# 1. Conversion of Strong Entities –

- For each strong entity in ERD, create a separate table with the same name.
- Create all simple Attributes
- Break the Composite attributes into simple attributes & create them.
- Choose a Primary Key for the table.

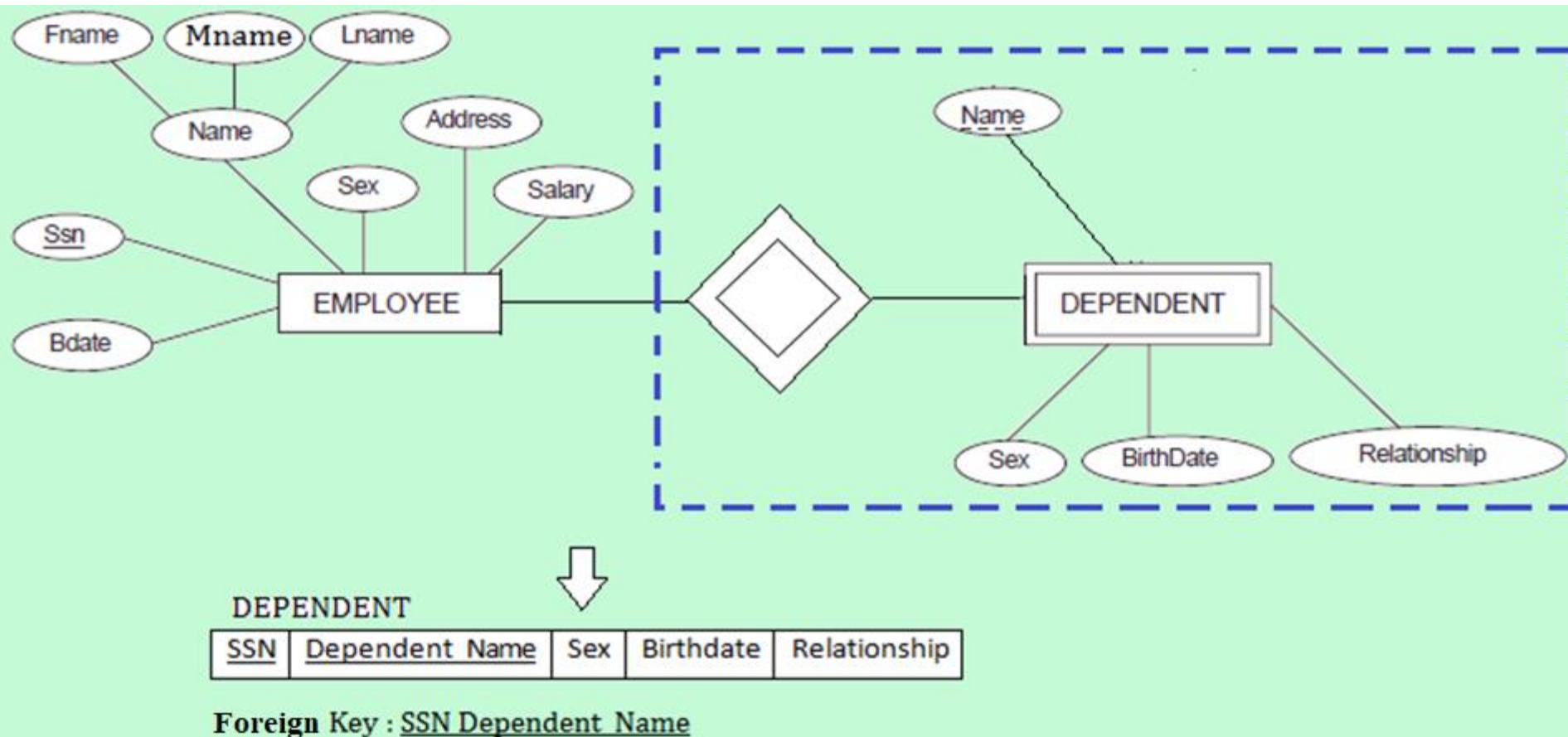


<u>SSN</u>	Fname	Mname	Lname	Sex	Bdate	Address	Salary
------------	-------	-------	-------	-----	-------	---------	--------



## 2. Conversion of Weak Entities

- For each weak entity, create a separate table with the same name.
- Include Primary Key of the strong entity as a foreign key in the table.
- Select the Primary Key attributes of strong entity and the partial Key attribute of the weak entity, and declare them as primary key.



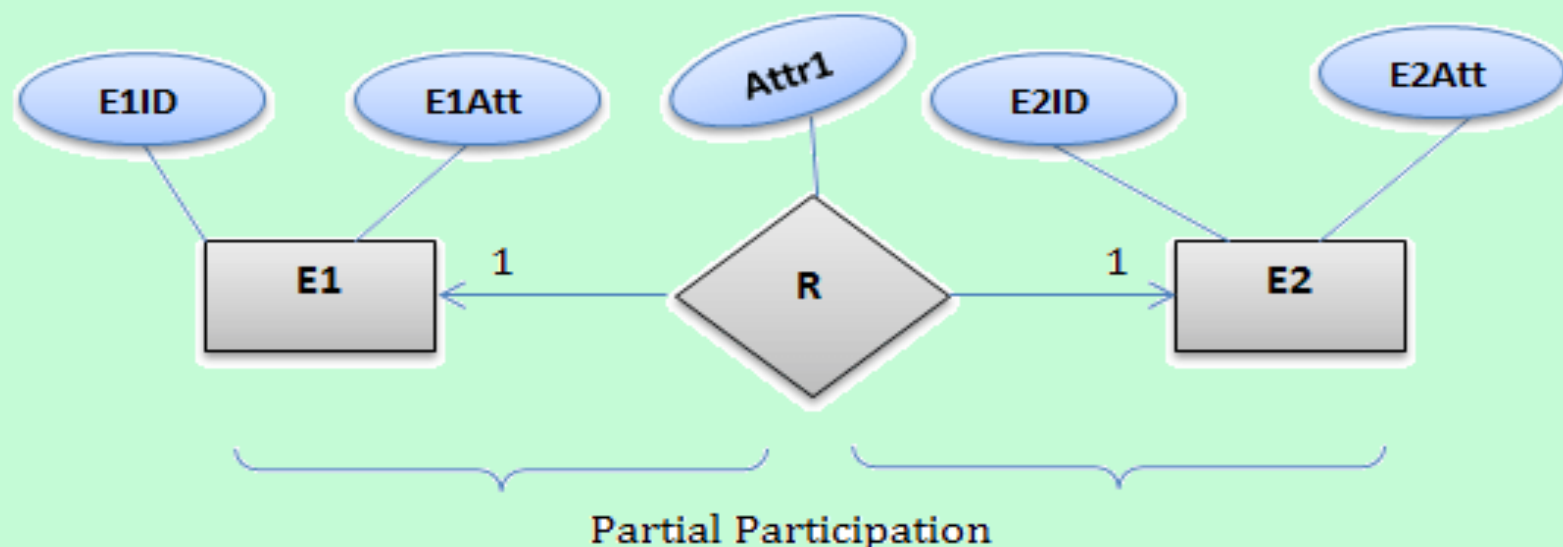
### 3. Conversion of One to One Relationships

There are two possible approaches on the basis of Participation Constraints –

#### 1. Partial Participation on Both Sides –

- For each One to One Cardinality between E1 and E2 with partial participation on both sides, modify either E1 or E2 to include the primary key of other table as a foreign key. So, 1:1 cardinality with partial participation on both sides can be minimized into two relations only.
- If we try to minimize the above ERD in a single table, i.e. E1RE2, then it contains too many NULL values, and therefore, we are not be able to select a primary key.

## One to One Conversion with Partial Participation on Both Sides -



<u>E1ID</u>	E1Attr
1	x
2	x
3	y

E1ID	E2ID
1	1003
2	1001

<u>E2ID</u>	E2Attr
1001	p
1002	q
1003	q

PK : Primary Key  
FK : Foreign Key

Modification Can be done as -

<u>E1ID</u>	E1Attr	E2ID
1	X	q
2	X	p
3	Y	Null

PK : E1ID  
FK : E2id

<u>E2ID</u>	E2Attr
1001	p
1002	q
1003	q

PK : E2ID

(OR)

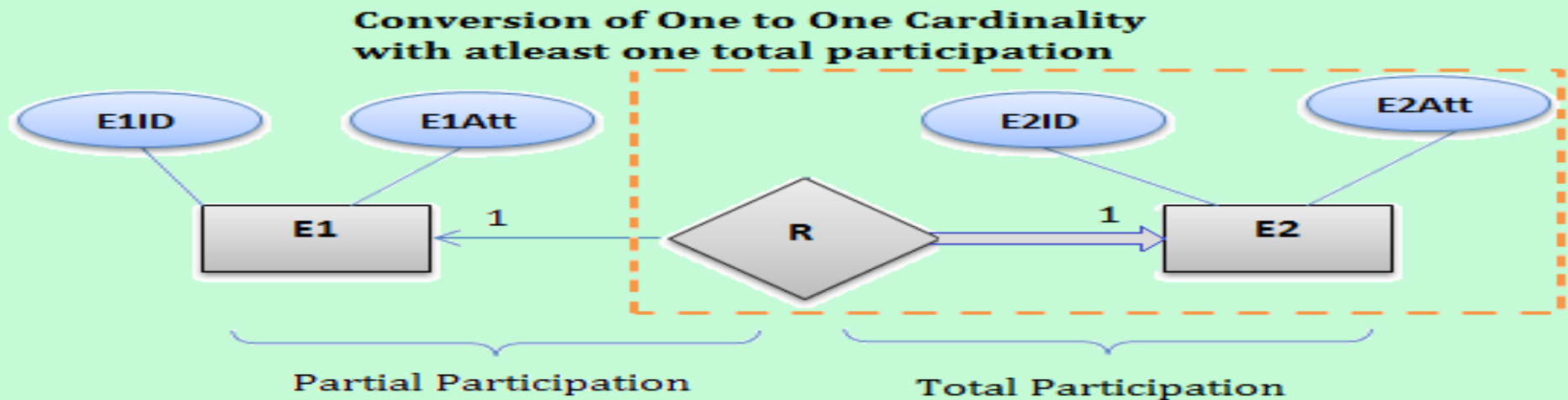
<u>E2ID</u>	E2Attr	E1ID
1001	p	2
1002	q	Null
1003	q	1

PK : E2ID  
FK : E1ID

<u>E1ID</u>	E1Attr
1	x
2	x
3	y

PK : E1ID

**2. Cardinality with atleast one Total Participation** – For each One to One Cardinality between E1 and E2 with atleast one total participation, modification is done only on total participation side. So, One to One Cardinality with atleast one Total Participation can be minimized into a single relation.



E1ID	E1Attr
1	x
2	x
3	y

E1ID	E2ID
3	1001
2	1002

E2ID	E2Attr
1001	p
1002	q

Modification Can be done as -

E1ID	E1Attr
1	x
2	x
3	y

Primary Key : E2ID

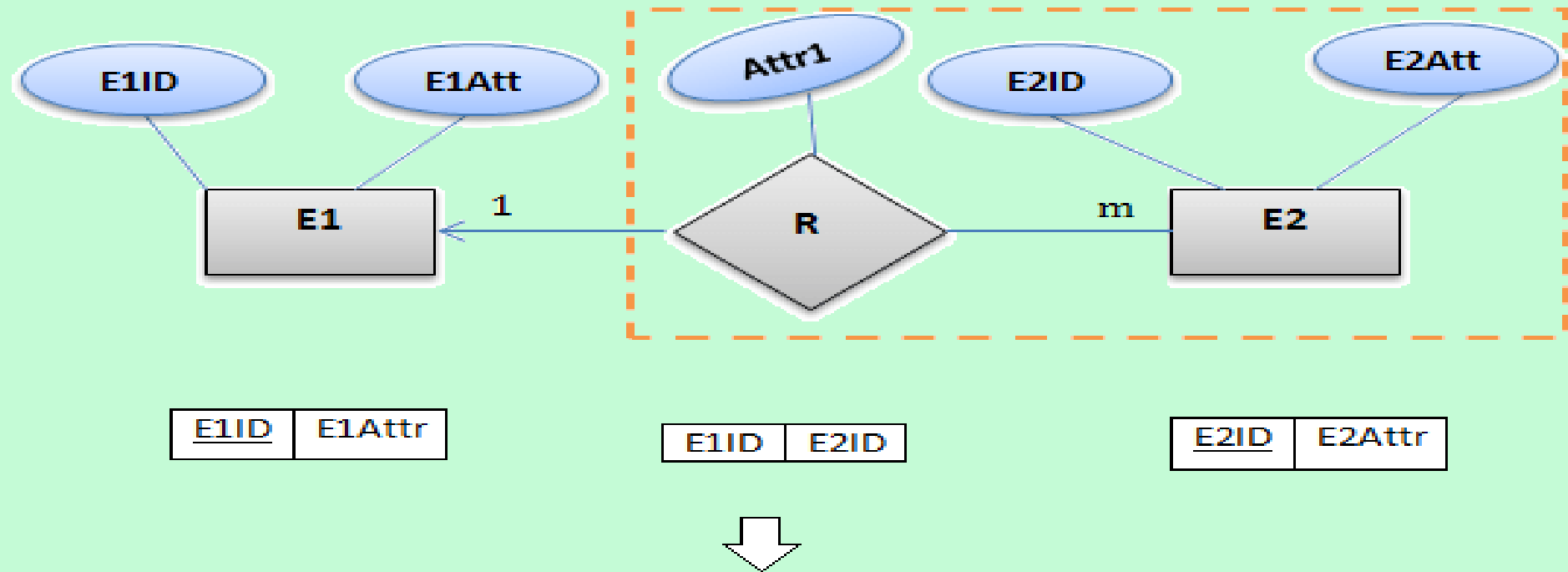
E2ID	E1ID	E2Attr
1001	3	P
1002	2	q

Primary Key : E2ID  
Foreign Key : E1ID

## 4. Conversion of One to Many or Many to One Relationship –

For each one to many relationship between E1 and E2, modify many side relation to include from one side as a Foreign Key.

### One to Many or Many to one Conversion -



Modificaion can be done as -

<u>E1ID</u>	E1Attr
-------------	--------

Primary Key : E1ID

<u>E2ID</u>	E1ID	E2Attr	Attr1
-------------	------	--------	-------

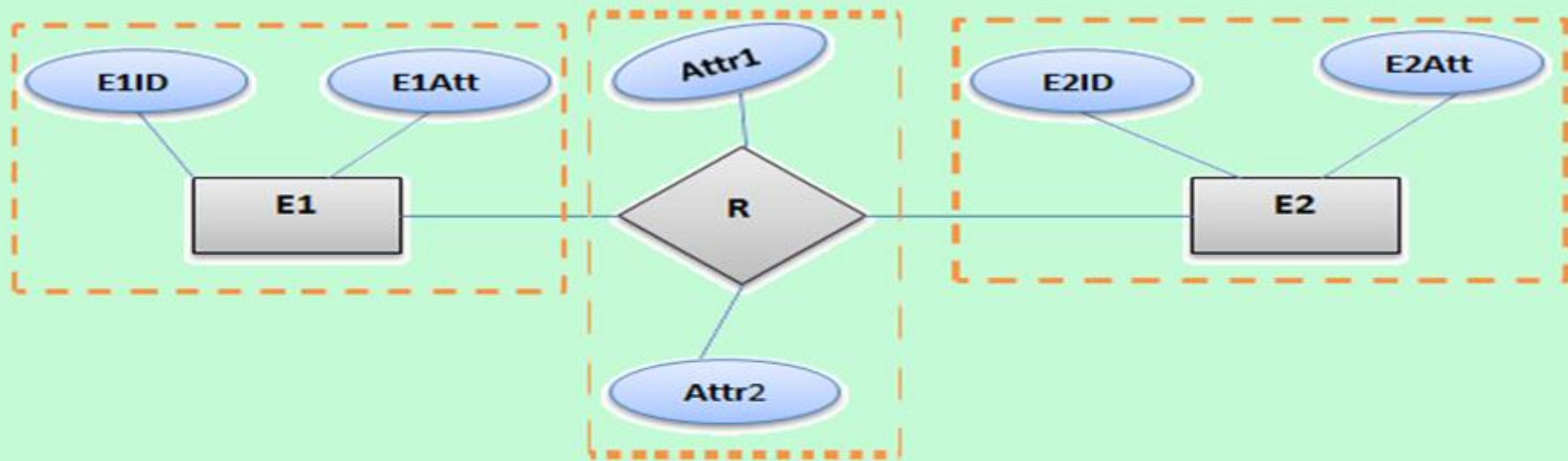
Primary Key : E2ID  
Foreign Key : E1ID

## 5. Conversion of Many to Many Relationship –

For each one to many relationship between E1 and E2, create a separate table and include primary key of both the tables as a Foreign Key.

If relationship is having one or more attributes, then these must also be included in the table.

Many To Many Relationship Conversion -



<u>E1ID</u>	E1Attr
-------------	--------

E1ID	E2ID	Attr1	Attr2
------	------	-------	-------

<u>E2ID</u>	E2Attr
-------------	--------



Modification can be Done as -

<u>E1ID</u>	E1Attr
-------------	--------

PK : E1ID

<u>E1ID</u>	<u>E2ID</u>	Attr1	Attr2
-------------	-------------	-------	-------

FK : E1ID E2ID

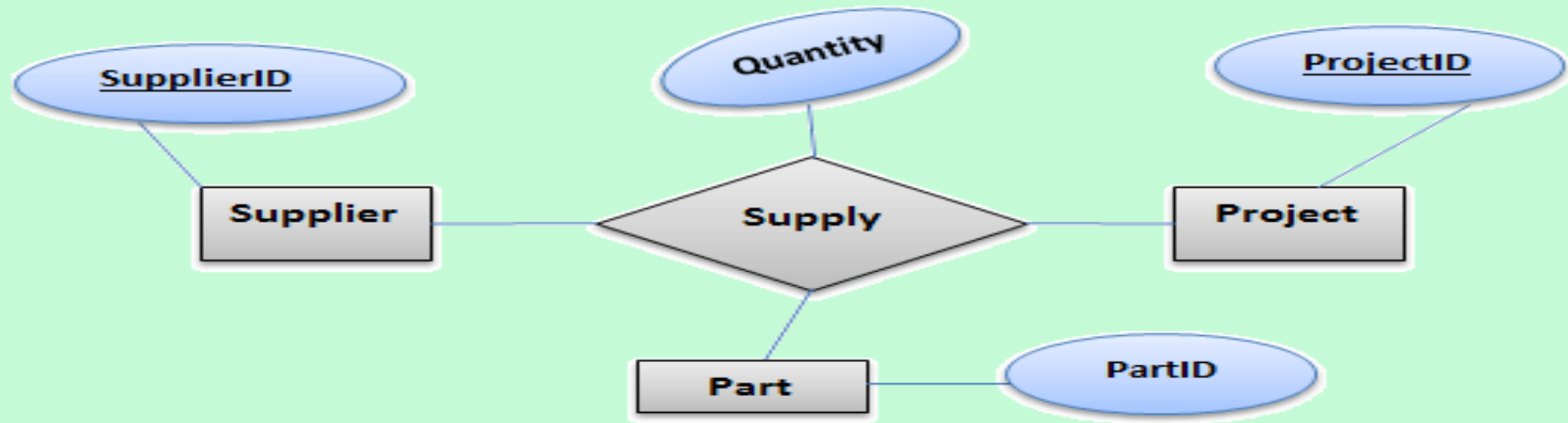
<u>E2ID</u>	E2Attr
-------------	--------

PK : E2ID

## 6. Conversion of n-ary Relationship –

- For each n-ary relationship, Create a separate table and include primary keys of all other entities as a foreign key.
- If the relationships has some attributes, then these must also be included in the table.

Conversion of n-ary Relationships -



<u>SupplierID</u>	.....
-------------------	-------

PK : SupplierID

<u>PartID</u>	.....
---------------	-------

PK : PartID

<u>ProjectID</u>	.....
------------------	-------

PK : ProjectID

<u>SupplierID</u>	<u>PartID</u>	<u>ProjectID</u>	Quantity
-------------------	---------------	------------------	----------

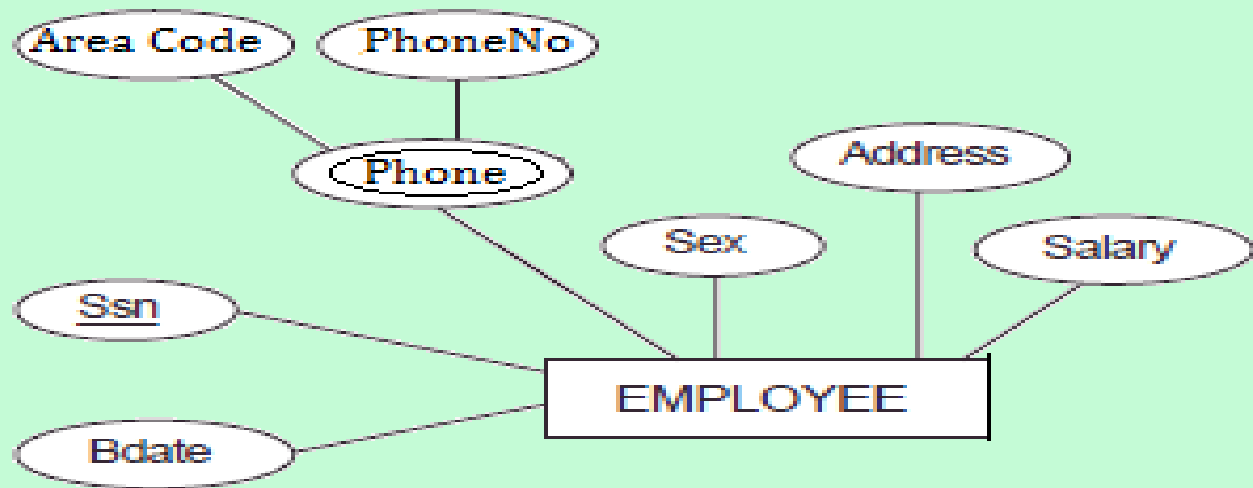
PK : SupplierID ProjectID Quantity

## 7. Conversion of multivalued Attributes –

.For each multivalued attributes, create a separate table, then include all of its simple attributes.

.Include the primary key of the original table as a foreign key.

### Conversion of Multivalued Attribute -



EMPLOYEE

<u>SSN</u>	Bdate	Sex	Address	Salary
------------	-------	-----	---------	--------

Phone

<u>SSN</u>	Area_Code	Phone_No.
------------	-----------	-----------