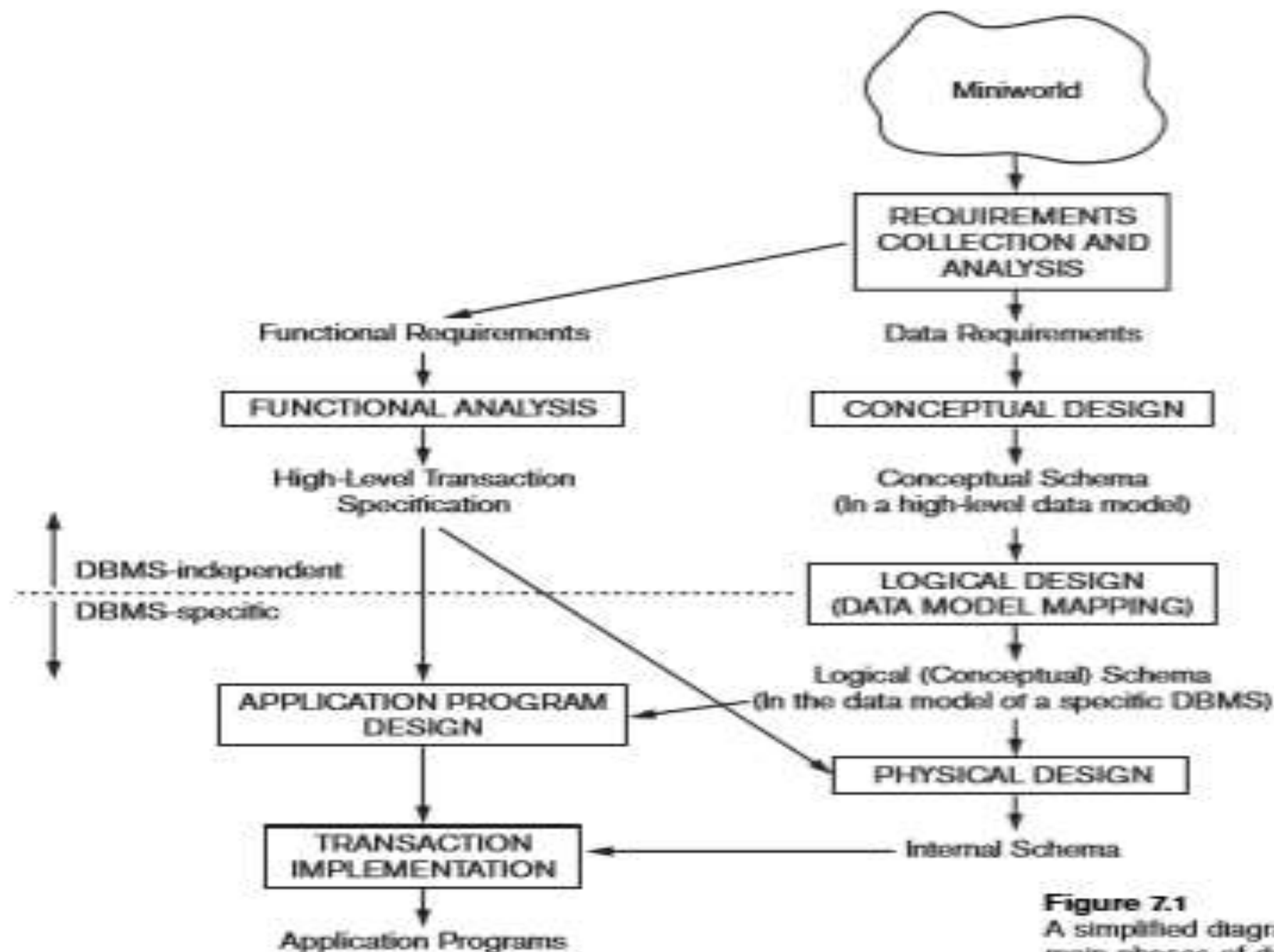


# Conceptual Data Modeling Using Entities and Relationships



**Figure 7.1**  
A simplified diagram to illustrate the main phases of database design.

# ER(Entity\_Relationship) Model

- ER model is a popular high-level conceptual data model.
- The diagrammatic notation associated with the ER model is known as ER diagrams.
- The ER model uses the concepts such as **entities** ,**relationships** and **attributes** to describe the data stored in the database.

- **Entity:**

An entity may be an object with a physical existence (Ex.: a particular person, car, house, employee) or it may be an object with a conceptual existence (Ex.: a job, a university course).

- **Attribute:**

The particular properties that describe an entity are called as attributes of an entity.

Ex. : Name and SSN of an EMPLOYEE.

- **Relationships:**

These are the associations of entities. Ex. : Employee **Works\_for** the Department. Here Works\_for is the association between Employee and Department.

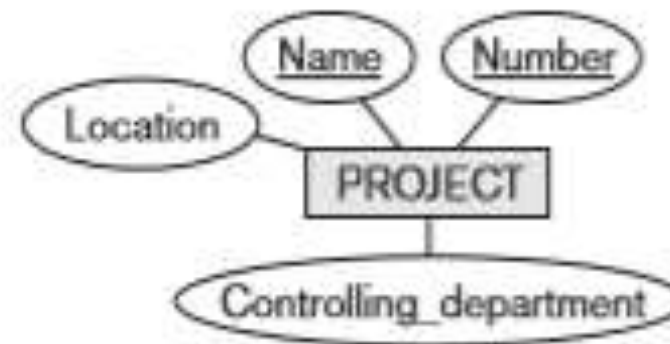
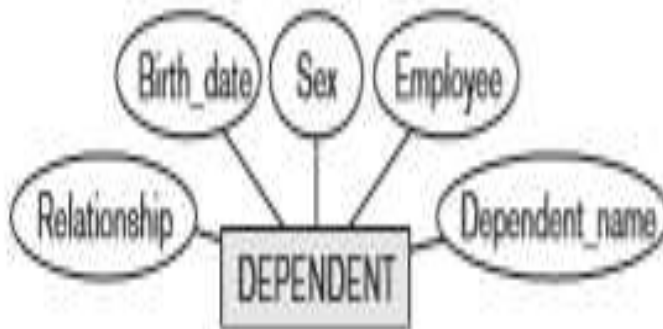
# Different types of Attributes

1. Simple versus Composite
2. Single valued versus Multivalued
3. Stored versus Derived
4. Complex

EX. {Address\_phone( {Phone(Area\_code,Phone\_number)},Address(Street\_address  
(Number,Street,Apartment\_number),City,State,Zip) )}

# Conceptual Design of COMPANY Database

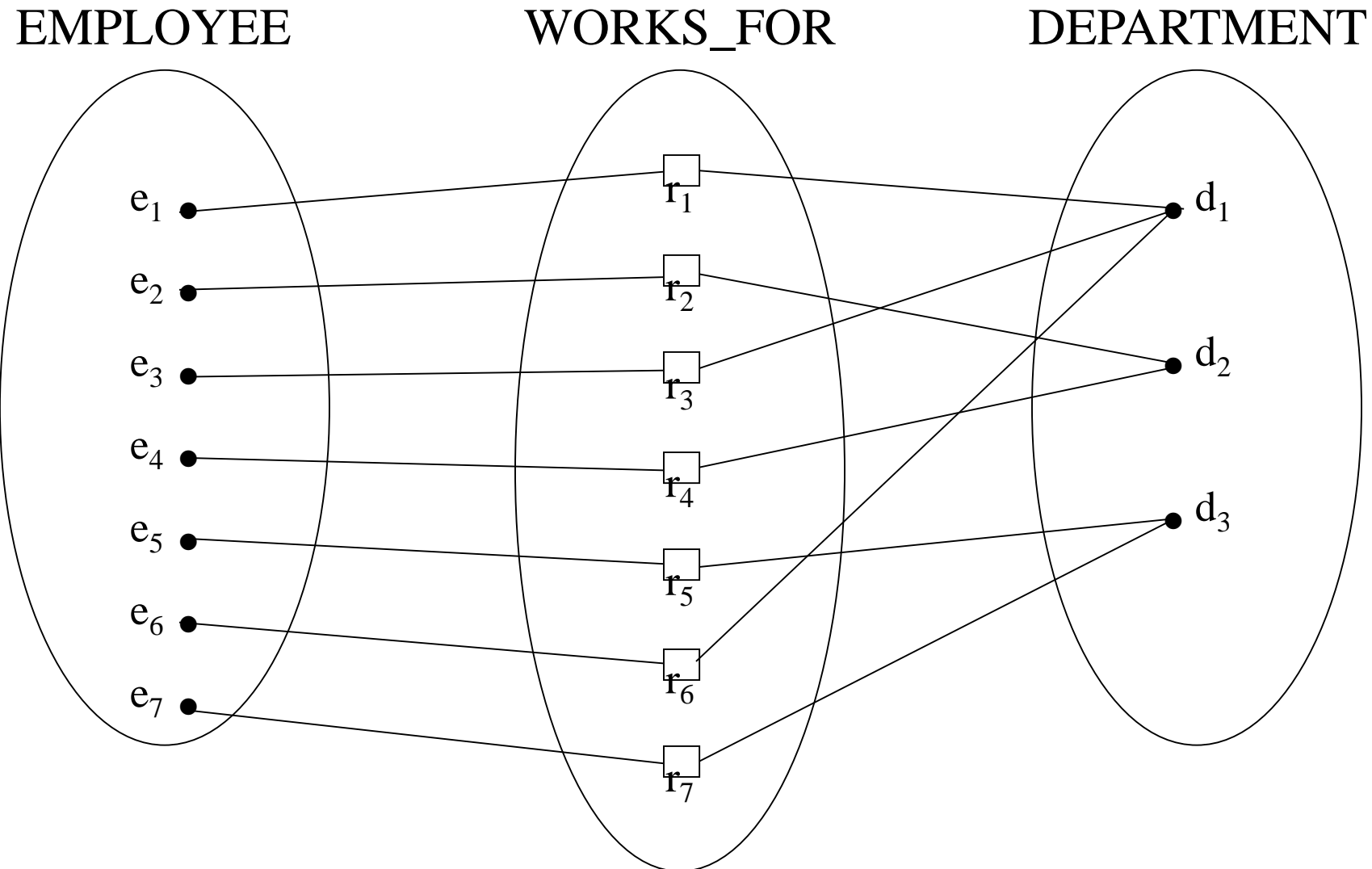
Initial Conceptual Design of the COMPANY Database:



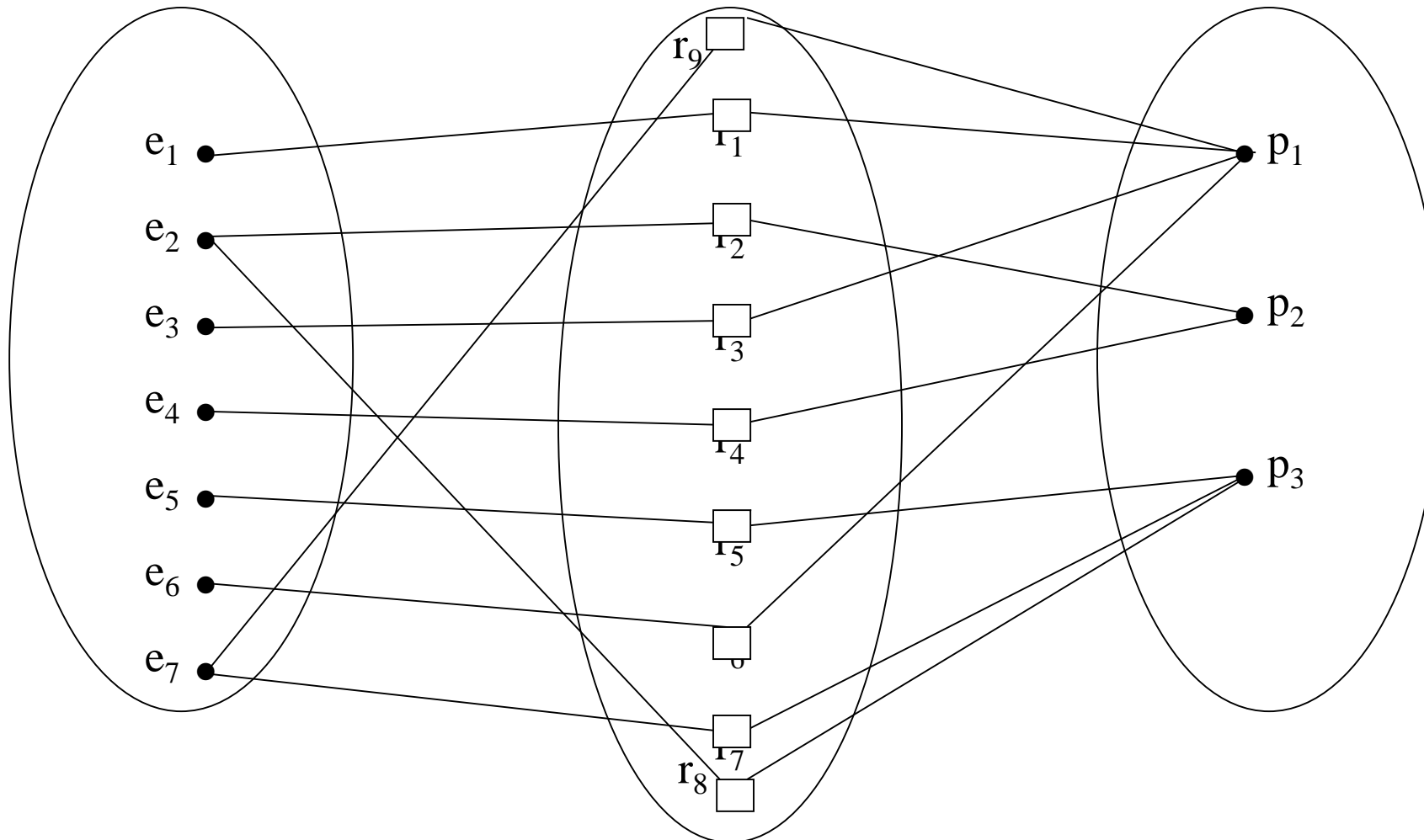
# Relationship Types, Sets, and Instances

- A relationship type **R** among  $n$  entity types **E1, E2, ..., En** defines a set of associations — or a relationship set
- Mathematically, the relationship set **R** is a set of relationship instances **ri**, where each **ri** associates  $n$  individual entities (**e1,e2,...,en**)
- 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.
- Relationships of the same type are grouped into a relationship type.  
For example, the WORKS\_ON relationship type in which EMPLOYEEs and PROJECTs participate, or the MANAGES relationship type in which EMPLOYEEs and DEPARTMENTs participate.

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



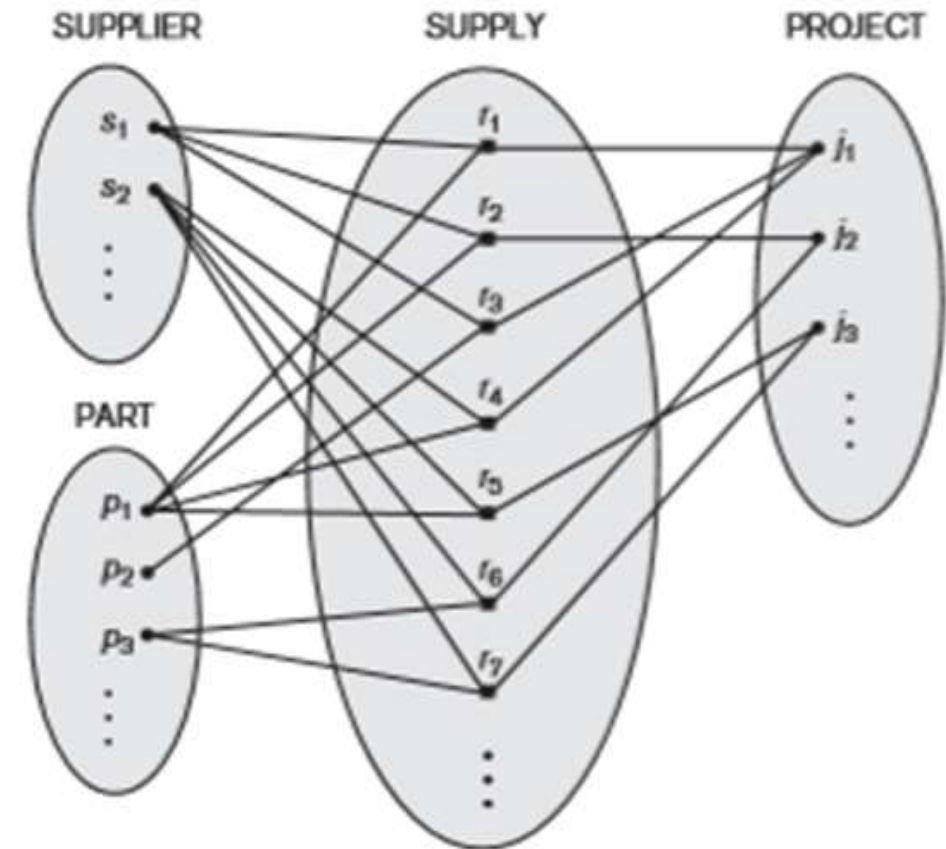
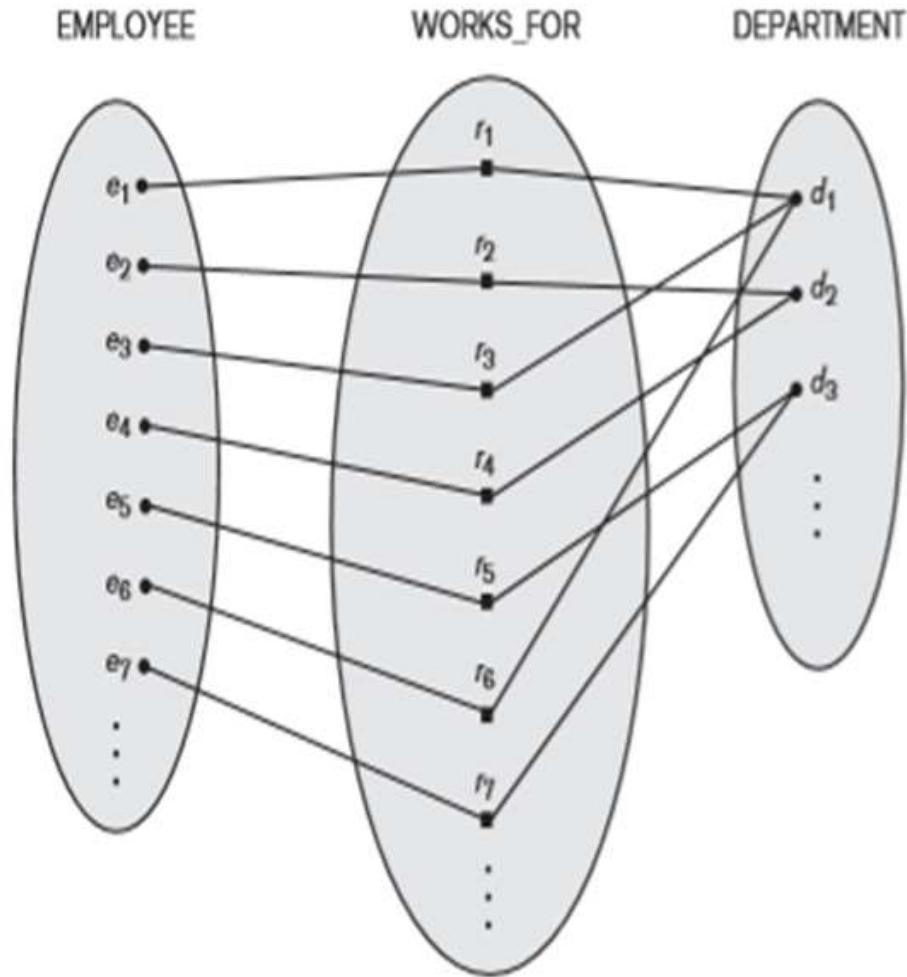
## Example relationship instances of the WORKS\_ON relationship between EMPLOYEE and PROJECT





The degree of a relationship type is the number of participating entity types.

A relationship type of degree two is called binary, and one of degree three is called ternary

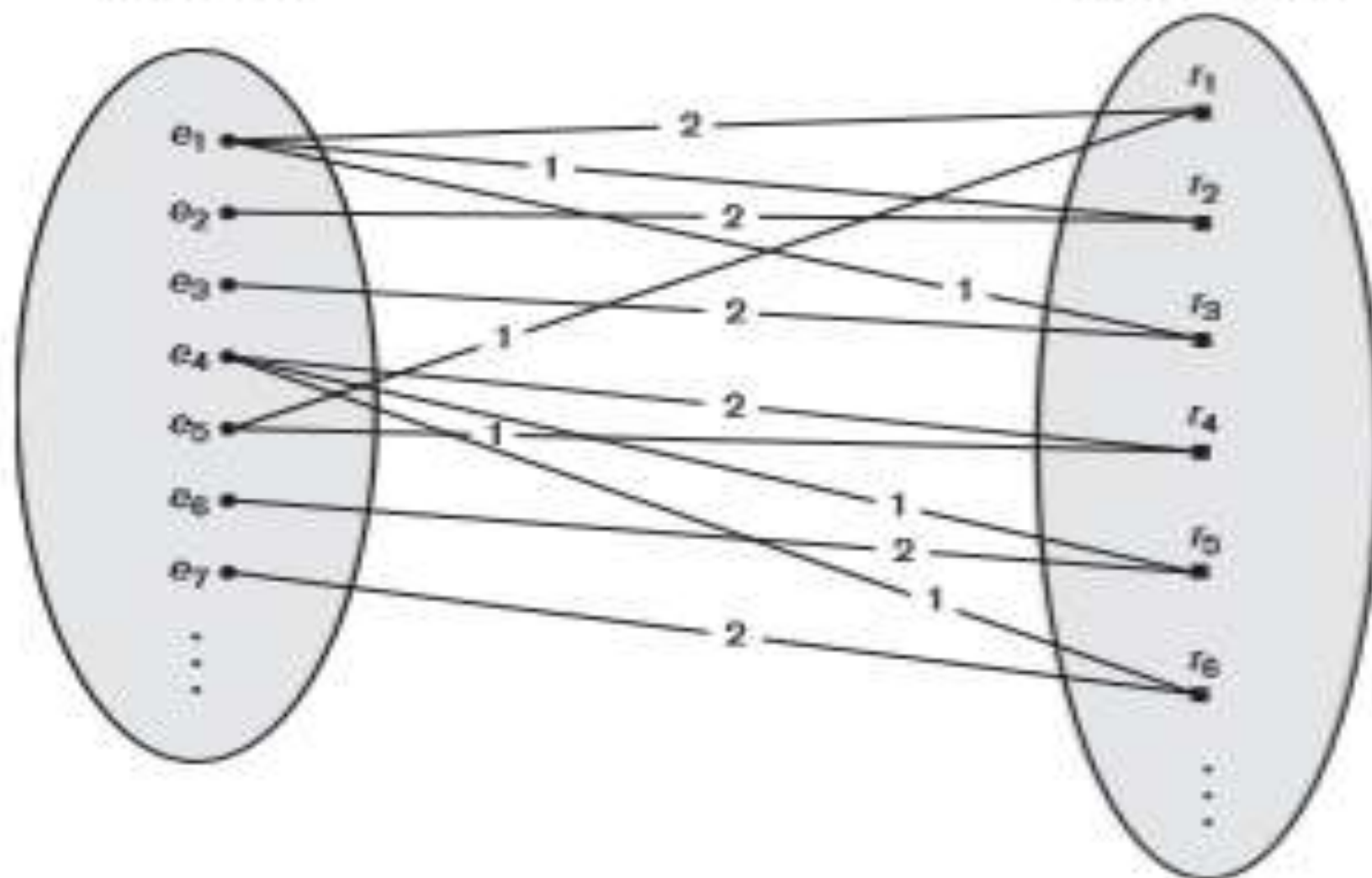


# Role Names and Recursive Relationships

- Each entity type that participates in a relationship type plays a particular role in the relationship.
- The role name signifies the role that a participating entity from the entity type plays in each relationship instance, and helps to explain what the relationship means.
- For example, in the WORKS\_FOR relationship type, EMPLOYEE plays the role of employee or worker and DEPARTMENT plays the role of department or employer.
- Role names are not technically necessary in relationship types where all the participating entity types are distinct, since each participating entity type name can be used as the role name.
- However, in some cases the same entity type participates more than once in a relationship type in different roles. In such cases the role name becomes essential for distinguishing the meaning of the role that each participating entity plays. Such relationship types are called **recursive relationships**.

EMPLOYEE

SUPERVISION



# Constraints on Binary Relationship Types

- Relationship types usually have certain constraints that limit the possible combinations of entities that may participate in the corresponding relationship set.
- These constraints are determined from the miniworld situation that the relationships represent.
- **Ex.** The company has a rule that each employee must work for exactly one department. This is the constraint of the relationship “WORKS\_FOR”. And this constraint must be described in the schema.
- There are two main types of binary relationship constraints: 1) **Cardinality ratio**  
2) **Participation.**

# Cardinality Ratios for Binary Relationships

- The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in.
- For example, in the WORKS\_FOR binary relationship type, DEPARTMENT:EMPLOYEE is of cardinality ratio 1:N, meaning that each department can be related to (that is, employs) any number of employees, but an employee can be related to (work for) only one department.
- This means that for this particular relationship WORKS\_FOR, a particular department entity can be related to any number of employees (N indicates there is no maximum number).
- On the other hand, an employee can be related to a maximum of one department.
- The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

- An example of a 1:1 binary relationship is MANAGES which relates a department entity to the employee who manages that department.
- This represents the miniworld constraints that—at any point in time—an employee can manage one department only and a department can have one manager only.
- The relationship type WORKS\_ON is of cardinality ratio M:N, because the miniworld rule is that an employee can work on several projects and a project can have several employees.
- Cardinality ratios for binary relationships are represented on ER diagrams by displaying 1, M, and N on the diamonds.
- Notice that in this notation, we can either specify no maximum (N) or a maximum of one (1) on participation.
- An alternative notation allows the designer to specify a specific maximum number on participation, such as 4 or 5.

# Participation Constraints and Existence Dependencies.

- The participation constraint specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- This constraint specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the minimum cardinality constraint.
- There are two types of participation constraints—**total** and **partial**.

## ➤ **Total participation:**













- ✓ If a company policy states that every employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS\_FOR relationship instance.
- ✓ Thus, the participation of EMPLOYEE in WORKS\_FOR is called **total participation**, meaning that every entity in the total set of employee entities must be related to a department entity via WORKS\_FOR relationship.
- ✓ Total participation is also called **existence dependency**.

## ➤ **Partial participation:**


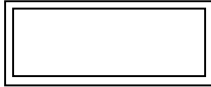





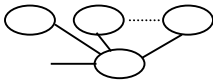

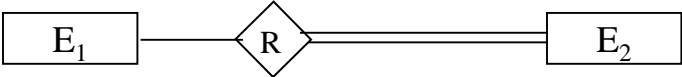
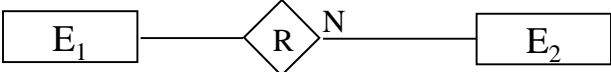
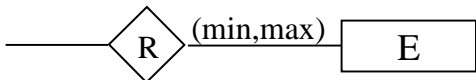
- ✓ We do not expect every employee to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is **partial**, meaning that some or part of the set of employee entities are related to some department entity via MANAGES, but not necessarily all.
- ✓ In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship, whereas partial participation is represented by a single line .
- ✓ Notice that in this notation, we can either specify no minimum (partial participation) or a minimum of one (total participation). The alternative notation allows the designer to specify a specific minimum number on participation in the relationship, such as 4 or 5.



# Summary of the notation for ER diagrams.

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	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 $R$

# Summary of the notation for ER diagrams.

Symbol	Meaning
	ENTITY TYPE
	WEAK ENTITY TYPE
	RELATIONSHIP TYPE
	IDENTIFYING RELATIONSHIP TYPE
	ATTRIBUTE
	KEY ATTRIBUTE
	MULTIVALUED ATTRIBUTE
	COMPOSITE ATTRIBUTE
	DERIVED ATTRIBUTE
	TOTAL PARTICIPATION OF E <sub>2</sub> IN R
	CARDINALITY RATIO 1:N FOR E <sub>1</sub> :E <sub>2</sub> IN R
	STRUCTURAL CONSTRAINT (min, max) ON PARTICIPATION OF E IN R

## Alternative (min, max) notation for relationship structural constraints:

- Specified on *each participation* of an entity type E in a relationship type R
- Specifies that each entity e in E participates in *at least* min and *at most* max relationship instances in R
- Default(no constraint): min=0, max=n
- Must have  $\text{min} \leq \text{max}$ ,  $\text{min} \geq 0$ ,  $\text{max} \geq 1$
- Derived from the knowledge of mini-world constraints

### Examples:

- A department has *exactly one* manager and an employee can manage *at most one* department.

Specify (0,1) for participation of EMPLOYEE in MANAGES

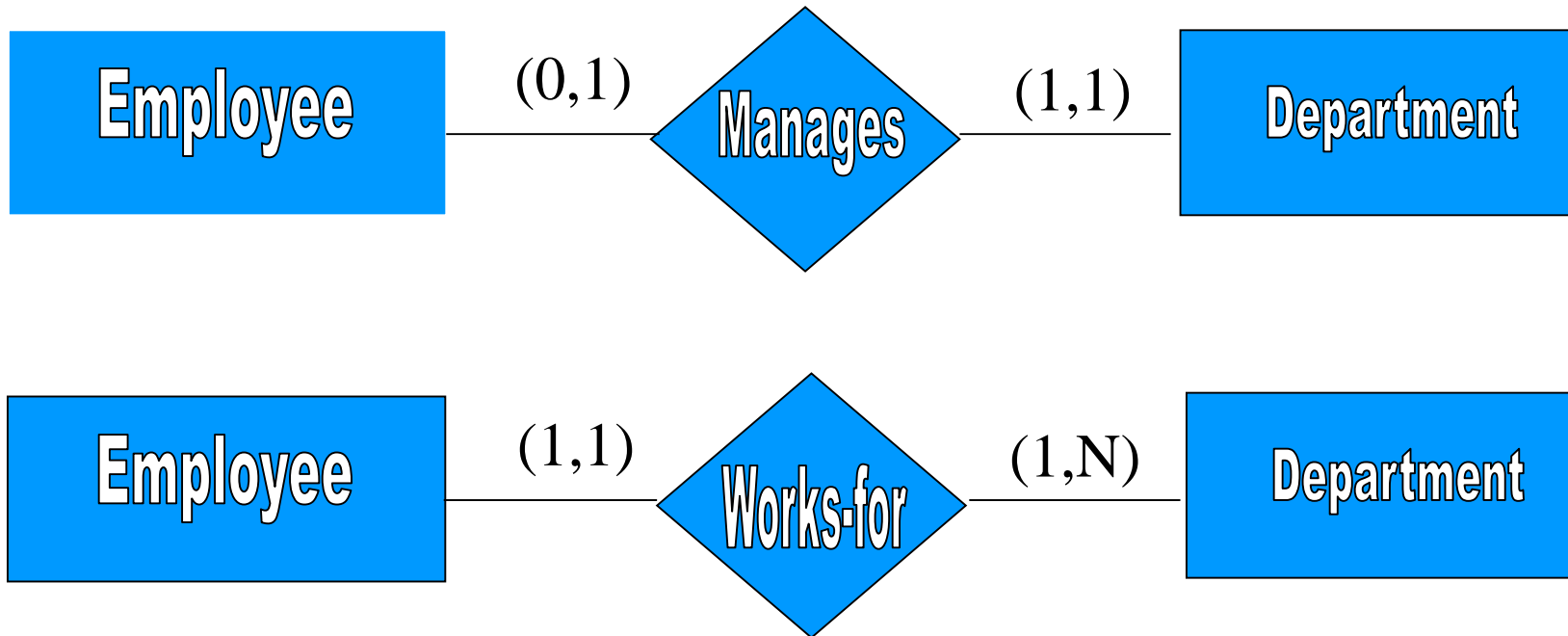
Specify (1,1) for participation of DEPARTMENT in MANAGES

- An employee can work for *exactly one* department but a department can have *any number of employees*.

Specify (1,1) for participation of EMPLOYEE in WORKS\_FOR

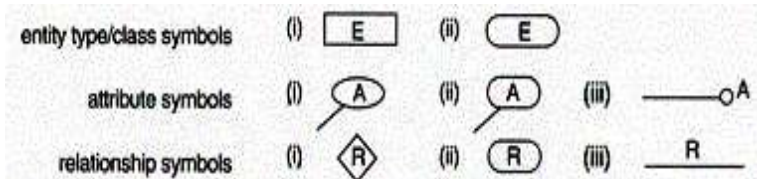
Specify (0,n) for participation of DEPARTMENT in WORKS\_FOR

# The (min,max) notation relationship constraints

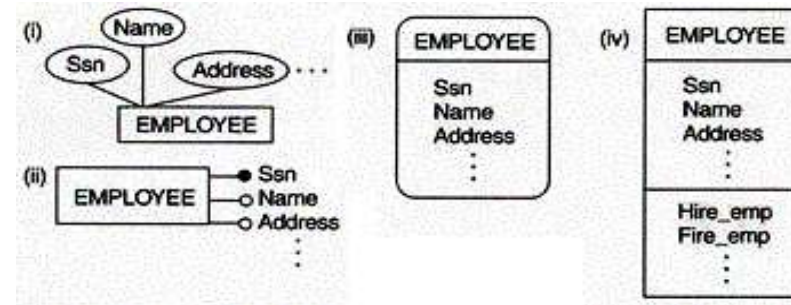


# Alternative Diagrammatic Notations

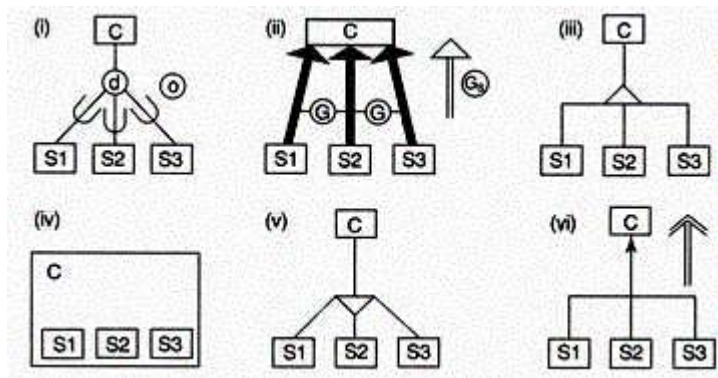
Symbols for entity type / class, attribute and relationship



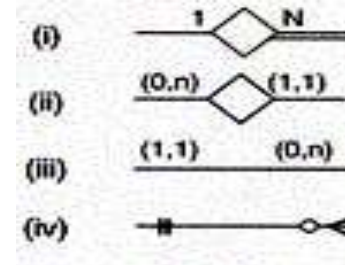
Displaying attributes



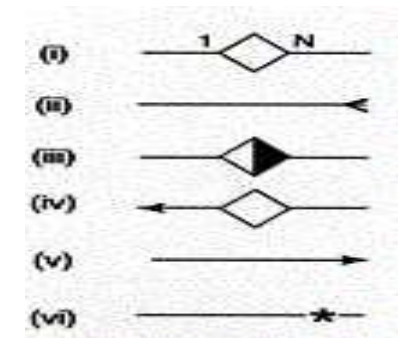
Notations for displaying specialization / generalization



Various (min, max) notations



Displaying cardinality ratios



# Weak Entity Types

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- Entities are identified by the combination of:
  - A partial key of the weak entity type
  - The particular entity they are related to in the identifying entity type

## Example:

Suppose that a **DEPENDENT** entity is identified by the dependent's first name and birthdate, *and* the specific **EMPLOYEE** that the dependent is related to. **DEPENDENT** is a weak entity type with **EMPLOYEE** as its identifying entity type via the identifying relationship type **DEPENDENT\_OF**

# Attributes of Relationship types

- A relationship type can have attributes; for example, **Hours PerWeek** of **WORKS\_ON**; its value for each relationship instance describes the number of hours per week that an **EMPLOYEE** works on a **PROJECT**.



# Refining the ER Design for the COMPANY Database

- We can now refine the database design in **slide 6** by changing the attributes that represent relationships into relationship types.
- A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type. It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship.
- It is important to note that in our notation, once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.
- The cardinality ratio and participation constraint of each relationship type are determined from the requirements.
- Similarly, an attribute that exists in several entity types may be elevated or promoted to an independent entity type. For example, suppose that several entity types in a UNIVERSITY database, such as STUDENT, INSTRUCTOR, and COURSE, each has an attribute **Department** in the initial design; the designer may then choose to create an entity type DEPARTMENT with a single attribute **Dept\_name** and relate it to the three entity types (STUDENT, INSTRUCTOR, and COURSE) via appropriate relationships. Other attributes/relationships of DEPARTMENT may be discovered later.



The binary relationship names are choosed in such a way that the ER diagram of the schema must be readable from left to right and from top to bottom.

In our company database example, we specify the following six relationship types:

1) **MANAGES**: It is a 1:1 relationship type between EMPLOYEE and DEPARTMENT.

EMPLOYEE participation is **partial**. DEPARTMENT participation is **total**, because the in the requirements, user mentioned that, an employee may manage or not a department, but a department must have a manager at all times. The attribute **Start\_date** is assigned to this relationship type.

2) **WORKS\_FOR**: It is a 1:N relationship type between DEPARTMENT and EMPLOYEE.

Both participations are total.

3) **CONTROLS**: It is a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial, after consultation with the users indicates that some departments may control no projects.

**4) SUPERVISION:** It is a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role).

Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.

**5) WORKS\_ON:** It is determined to be an M:N relationship type with attribute Hours, after the users indicate that a project can have several employees working on it.

Both participations are determined to be total.

**6) DEPENDENTS\_OF:** It is a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT.

The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

After specifying the above six relationship types, we remove from the entity types in slide6, all attributes that have been refined into relationships.

These include :

**Manager** and **Manager\_start\_date** from DEPARTMENT;

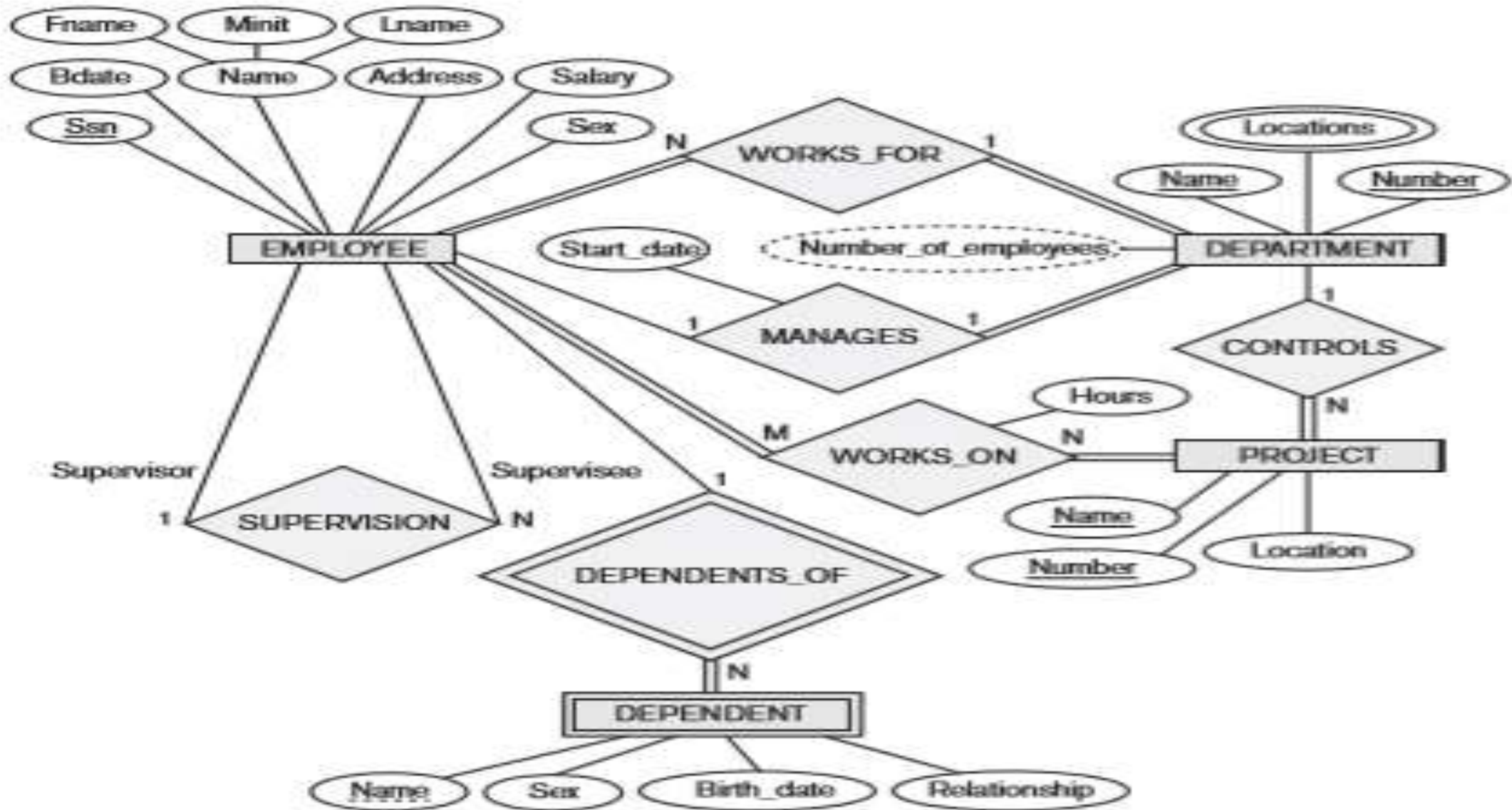
**Controlling\_department** from PROJECT;

**Department**, **Supervisor** and **Works\_on** from EMPLOYEE;

**Employee** from DEPENDENT.

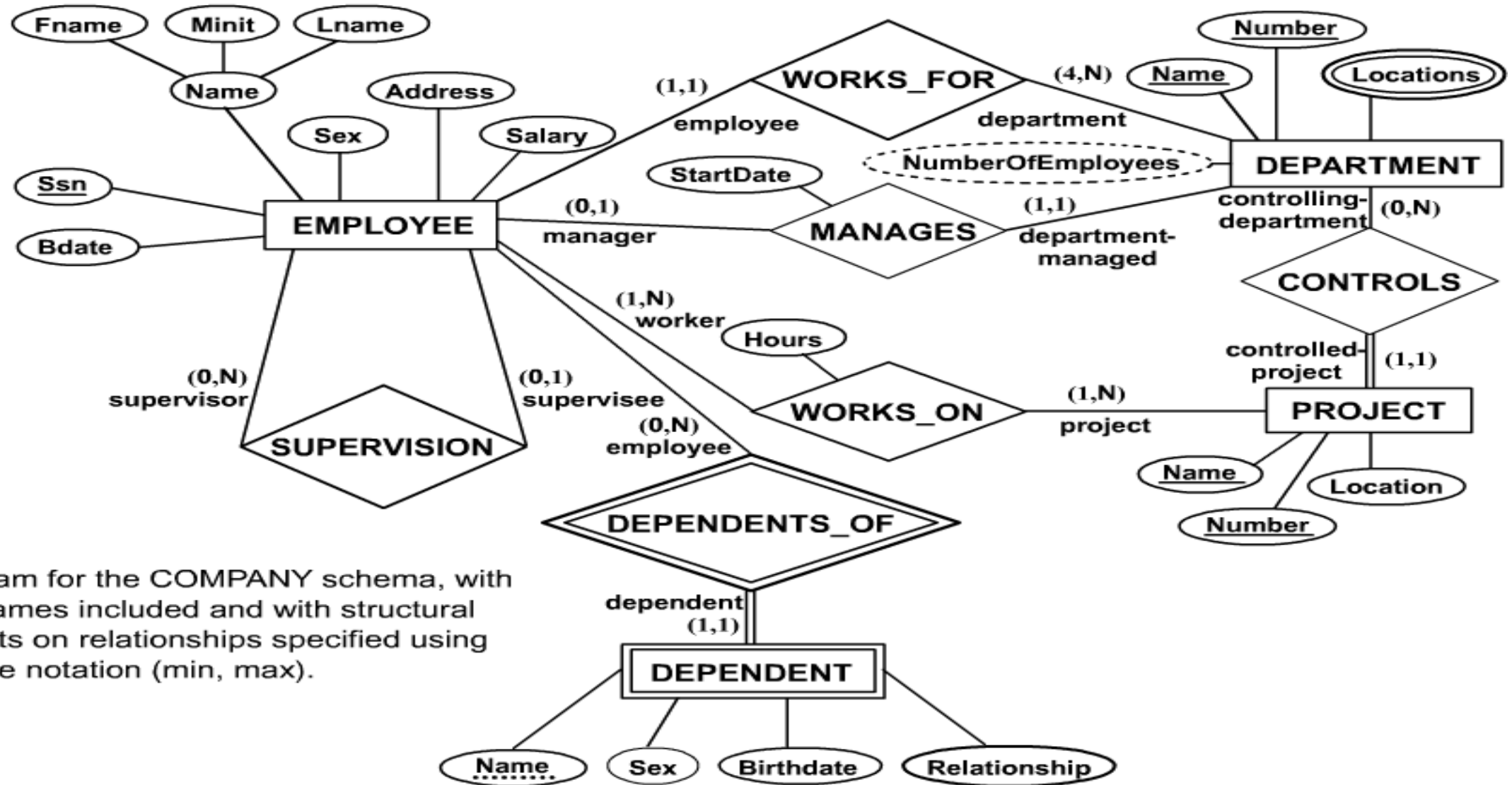
It is important to have the least possible redundancy when we design the conceptual schema of a database.

# An ER diagram for the COMPANY database



# COMPANY ER Schema Diagram using (min, max) notation

## Alternative ER Notations



ER diagram for the COMPANY schema, with all role names included and with structural constraints on relationships specified using alternative notation (min, max).



# Relational Database Design using ER-to-Relational Mapping

## **ER-to-Relational Mapping algorithm:**

### **Step 1: Mapping of Regular Entity Types:**

- ✓ For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
- ✓ Include only the simple component attributes of a composite attribute. Choose one of the key attributes of E as the primary key for R. If the chosen key of E is a composite, then the set of simple attributes that form it will together form the primary key of R.
- ✓ The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps.

### **Step 2: Mapping of Weak Entity Types:**

- ✓ For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- ✓ In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W.
- ✓ The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

### Step 3: Mapping of Binary 1:1 Relationship Types:

- ✓ For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- ✓ Choose one of the relations—S, say—and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
- ✓ Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.
- ✓ In our example(*company database*), we map the 1:1 relationship type MANAGES (*Refer ER Diagram of Company database*) by choosing the participating entity type DEPARTMENT to serve in the role of S because its participation in the MANAGES relationship type is total (every department has a manager).
- ✓ We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it Mgr\_ssn.
- ✓ We also include the simple attribute Start\_date of the MANAGES relationship type in the DEPARTMENT relation and rename it Mgr\_start\_date

## Step 4: Mapping of Binary 1:N Relationship Types:

- ✓ For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type.
- ✓ Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; we do this because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type.
- ✓ Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S.
- ✓ In our example, we now map the 1:N relationship types WORKS\_FOR, CONTROLS, and SUPERVISION (*Refer ER diagram of company database*).
- ✓ For WORKS\_FOR we include the primary key Dnumber of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it Dno.
- ✓ For SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself—because the relationship is recursive—and call it Super\_ssn.
- ✓ The CONTROLS relationship is mapped to the foreign key attribute Dnum of PROJECT, which references the primary key Dnumber of the DEPARTMENT relation.



## Step 5: Mapping of Binary M:N Relationship Types:

- ✓ For each binary M:N relationship type R, create a new relation S to represent R.
- ✓ Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.
- ✓ Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.
- ✓ In our example, we map the M:N relationship type WORKS\_ON ( *Refer ER diagram of company database*) by creating the relation WORKS\_ON
- ✓ We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS\_ON and rename them Pno and Essn, respectively.
- ✓ We also include an attribute Hours in WORKS\_ON to represent the Hours attribute of the relationship type. The primary key of the WORKS\_ON relation is the combination of the foreign key attributes {Essn, Pno}.

## Step 6: Mapping of Multivalued Attributes:

- ✓ For each multivalued attribute A, create a new relation R.
- ✓ This relation R will include an attribute corresponding to A, plus the primary key attribute K—as a foreign key in R—of the relation that represents the entity type or relationship type that has A as a multivalued attribute.
- ✓ The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.
- ✓ In our example, we create a relation DEPT\_LOCATIONS (*Company database*).
- ✓ The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation.
- ✓ The primary key of DEPT\_LOCATIONS is the combination of {Dnumber, Dlocation}.

## Step 7: Mapping of N-ary Relationship Types:

- ✓ For each  $n$ -ary relationship type  $R$ , where  $n > 2$ , create a new relation  $S$  to represent  $R$ .
- ✓ Include as foreign key attributes in  $S$  the primary keys of the relations that represent the participating entity types.
- ✓ Also include any simple attributes of the  $n$ -ary relationship type (or simple components of composite attributes) as attributes of  $S$ .
- ✓ The primary key of  $S$  is usually a combination of all the foreign keys that reference the relations representing the participating entity types.
- ✓ Ex. Mapping the  $n$ -ary relationship type SUPPLY

SUPPLIER

<u>SNAME</u>	...
--------------	-----

PROJECT

<u>PROJNAME</u>	...
-----------------	-----

PART

<u>PARTNO</u>	...
---------------	-----

SUPPLY

<u>SNAME</u>	<u>PROJNAME</u>	<u>PARTNO</u>	QUANTITY
--------------	-----------------	---------------	----------

## Result of mapping the COMPANY ER schema into a relational schema.

