

# CHAPTER 3

## Data Modeling Using the Entity-Relationship (ER) Model

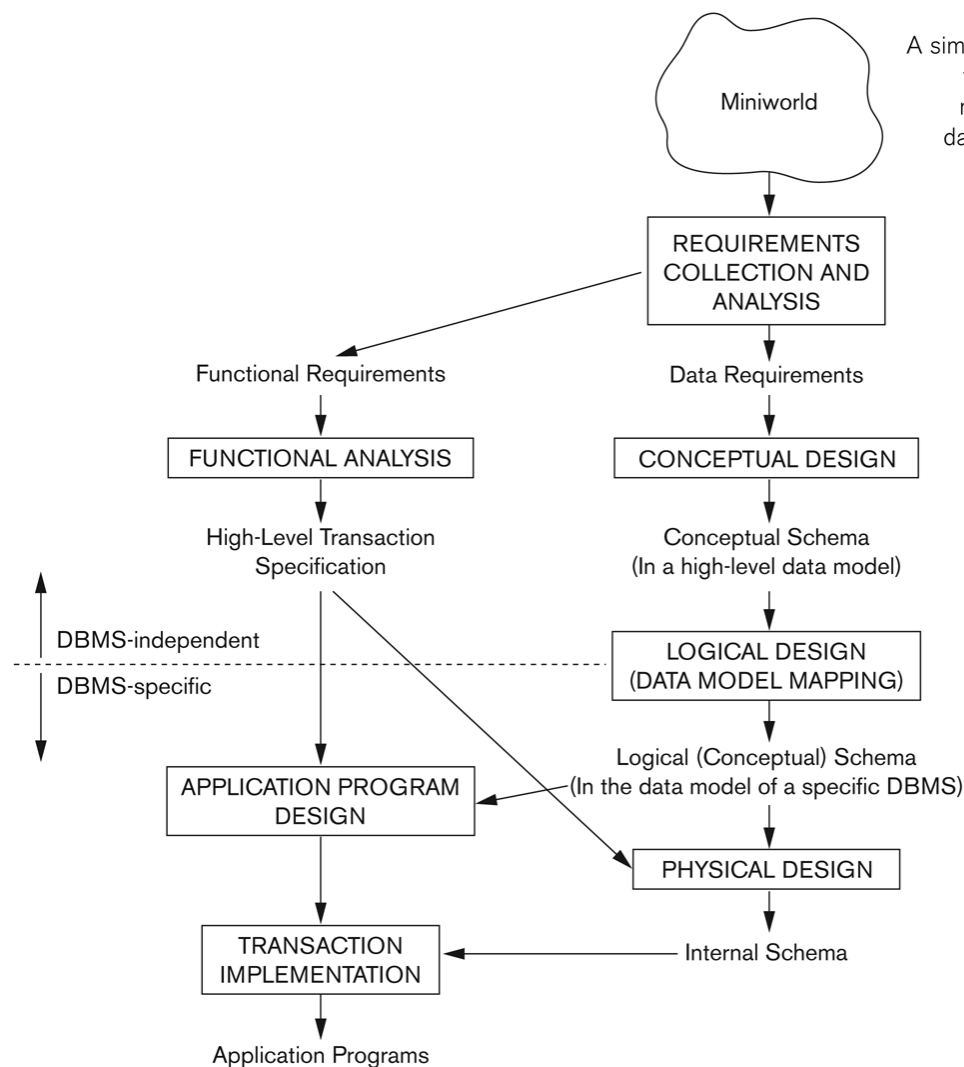
# Chapter Outline

- Overview of Database Design Process
- Example Database Application (COMPANY)
- ER Model Concepts
  - Entities and Attributes
  - Entity Types, Value Sets, and Key Attributes
  - Relationships and Relationship Types
  - Weak Entity Types
  - Roles and Attributes in Relationship Types
- ER Diagrams - Notation
- ER Diagram for COMPANY Schema
- Alternative Notations – UML class diagrams, others
- Relationships of Higher Degree

# Overview of Database Design Process

- Two main activities:
  - Database design
  - Applications design
- Focus in this chapter on conceptual database design
  - To design the conceptual schema for a database application
- Applications design focuses on the programs and interfaces that access the database
  - Generally considered part of software engineering

# Overview of Database Design Process



**Figure 3.1**  
A simplified diagram  
to illustrate the  
main phases of  
database design.

# Methodologies for Conceptual Design

- Entity Relationship (ER) Diagrams (This Chapter)
- Enhanced Entity Relationship (EER) Diagrams (Chapter 4)
- Use of Design Tools in industry for designing and documenting large scale designs
- The UML (Unified Modeling Language) Class Diagrams are popular in industry to document conceptual database designs

# Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
  - The company is organized into DEPARTMENTS. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
  - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

# Example COMPANY Database (Continued)

- The database will store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - The DB will keep track of the number of hours per week that an employee currently works on each project.
  - It is required to keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTS.
  - For each dependent, the DB keeps a record of name, sex, birthdate, and relationship to the employee.

# ER Model Concepts

## ■ Entities and Attributes

- Entity is a basic concept for the ER model. Entities are specific things or objects in the mini-world that are represented in the database.
  - For example the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
- Attributes are properties used to describe an entity.
  - For example an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate
- A specific entity will have a value for each of its attributes.
  - For example a specific employee entity may have Name='John Smith', SSN='123456789', Address ='731, Fondren, Houston, TX', Sex='M', BirthDate='09-JAN-55'
- Each attribute has a *value set* (or data type) associated with it – e.g. integer, string, date, enumerated type, ...



# 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.

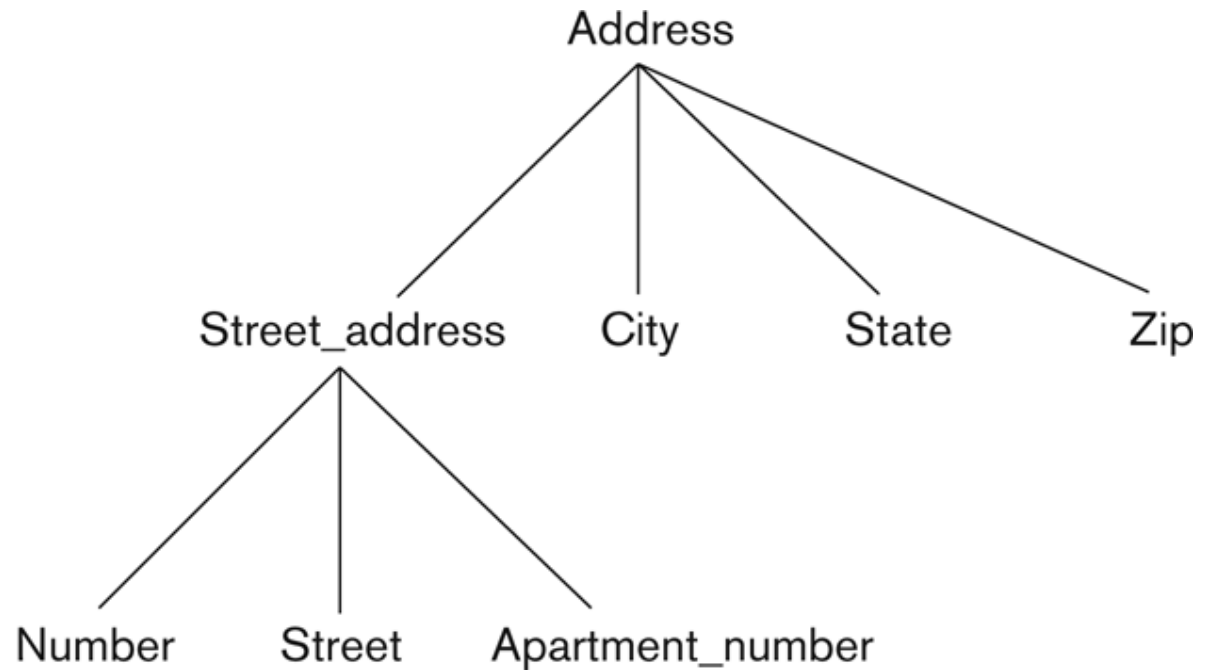
## ■ Multi-valued

- An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
  - Denoted as {Color} or {PreviousDegrees}.

# Types of Attributes (2)

- In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.
  - For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
  - Multiple PreviousDegrees values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field

# Example of a composite attribute



**Figure 3.4**

A hierarchy of composite attributes.

# Entity Types and Key Attributes (1)

- Entities with the same basic attributes are grouped or typed into an entity type.
  - For example, the entity type EMPLOYEE and PROJECT.
- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
  - For example, SSN of EMPLOYEE.

# Entity Types and Key Attributes (2)

- A key attribute may be composite.
  - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), aka license plate number.
- Each key is underlined (Note: this is different from the relational schema where only one “primary key is underlined).

# Entity Set

- Each entity type will have a collection of entities stored in the database
  - Called the **entity set** or sometimes **entity collection**
- Previous slide shows three CAR entity instances in the entity set for CAR
- Same name (CAR) used to refer to both the entity type and the entity set
- However, entity type and entity set may be given different names
- Entity set is the current *state* of the entities of that type that are stored in the database

# Value Sets (Domains) of Attributes

- Each simple attribute is associated with a value set
  - E.g., Lastname has a value which is a character string of upto 15 characters, say
  - Date has a value consisting of MM-DD-YYYY where each letter is an integer
- A **value set** specifies the set of values associated with an attribute

# Attributes and Value Sets

- Value sets are similar to data types in most programming languages – e.g., integer, character (n), real, bit
- Mathematically, an attribute  $A$  for an entity type  $E$  whose value set is  $V$  is defined as a function

$$A : E \rightarrow P(V)$$

Where  $P(V)$  indicates a power set (which means all possible subsets) of  $V$ . The above definition covers simple and multivalued attributes.

- We refer to the value of attribute  $A$  for entity  $e$  as  $A(e)$ .








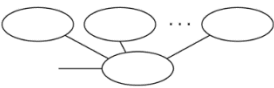

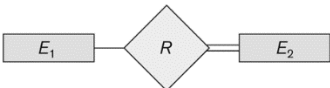




# Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
- Attributes are displayed in ovals
  - Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is underlined
  - Multivalued attributes displayed in double ovals
- See the full ER notation in advance on the next slide

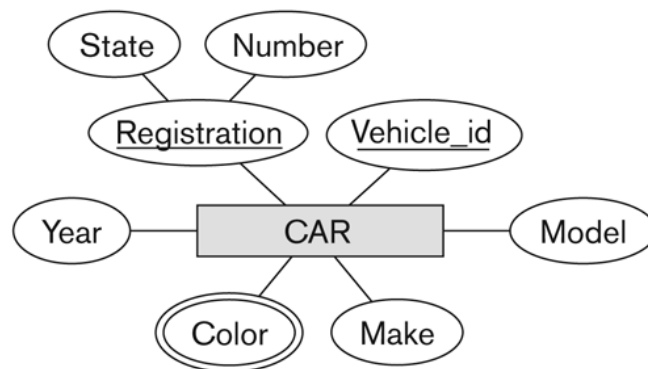
# NOTATION for ER diagrams

**Figure 3.14**  
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 $R$

# Entity Type CAR with two keys and a corresponding Entity Set

(a)



**Figure 3.7**

The CAR entity type with two key attributes, Registration and Vehicle\_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

CAR  
Registration (Number, State), Vehicle\_id, Make, Model, Year, {Color}

CAR<sub>1</sub>  
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR<sub>2</sub>  
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

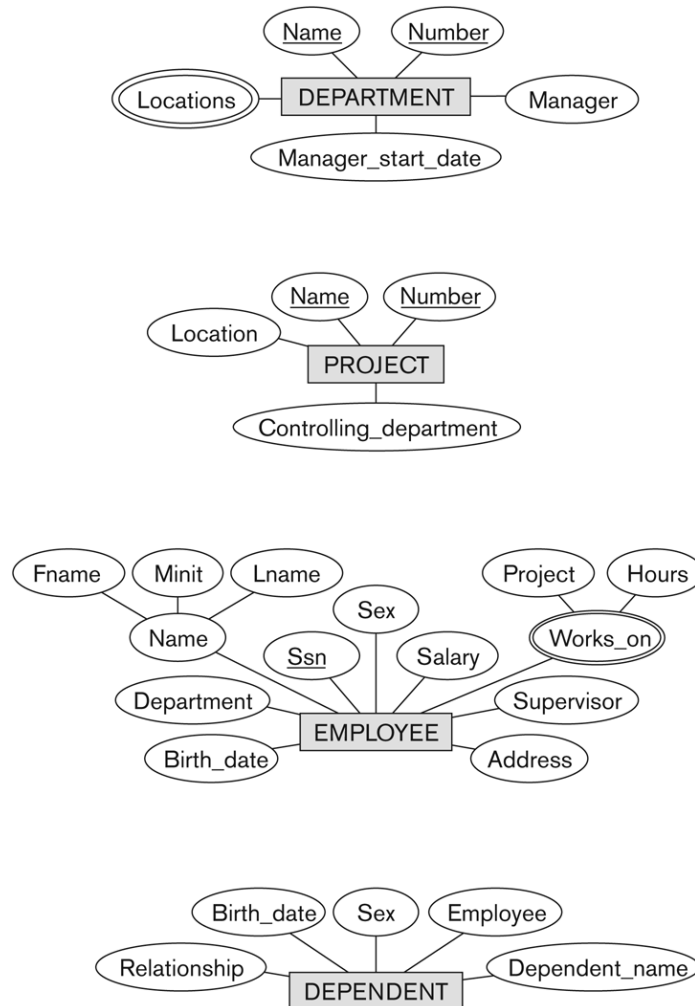
CAR<sub>3</sub>  
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

# Initial Conceptual Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - DEPARTMENT
  - PROJECT
  - EMPLOYEE
  - DEPENDENT
- Their initial conceptual design is shown on the following slide
- The initial attributes shown are derived from the requirements description

# Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT



**Figure 3.8**

Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

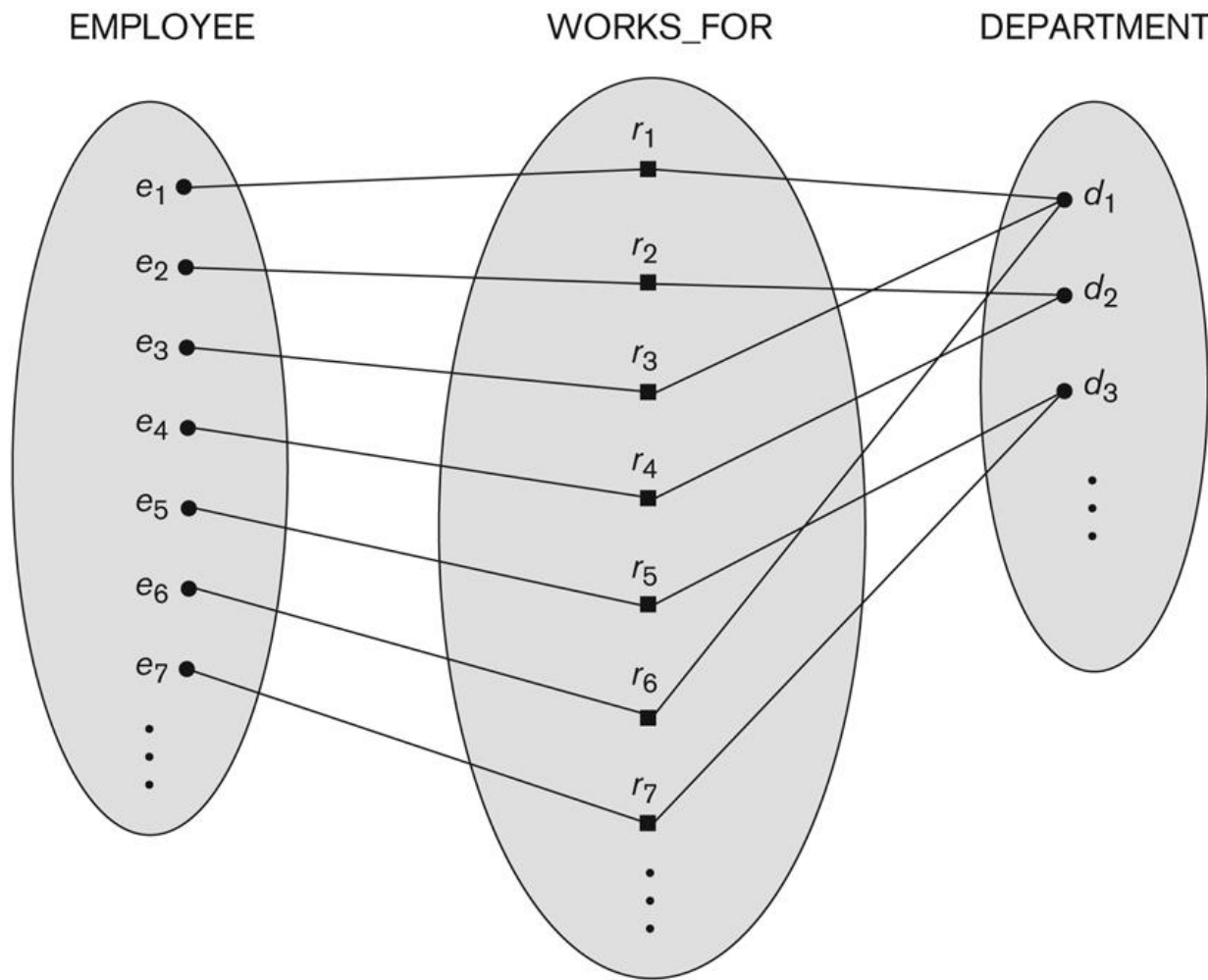
# Refining the initial design by introducing **relationships**

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - Entities (and their entity types and entity sets)
  - Attributes (simple, composite, multivalued)
  - Relationships (and their relationship types and relationship sets)
- We introduce relationship concepts next

# 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.
- Relationships of the same type are grouped or typed 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.
- The degree of a relationship type is the number of participating entity types.
  - Both MANAGES and WORKS\_ON are *binary* relationships.

# Relationship instances of the WORKS\_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

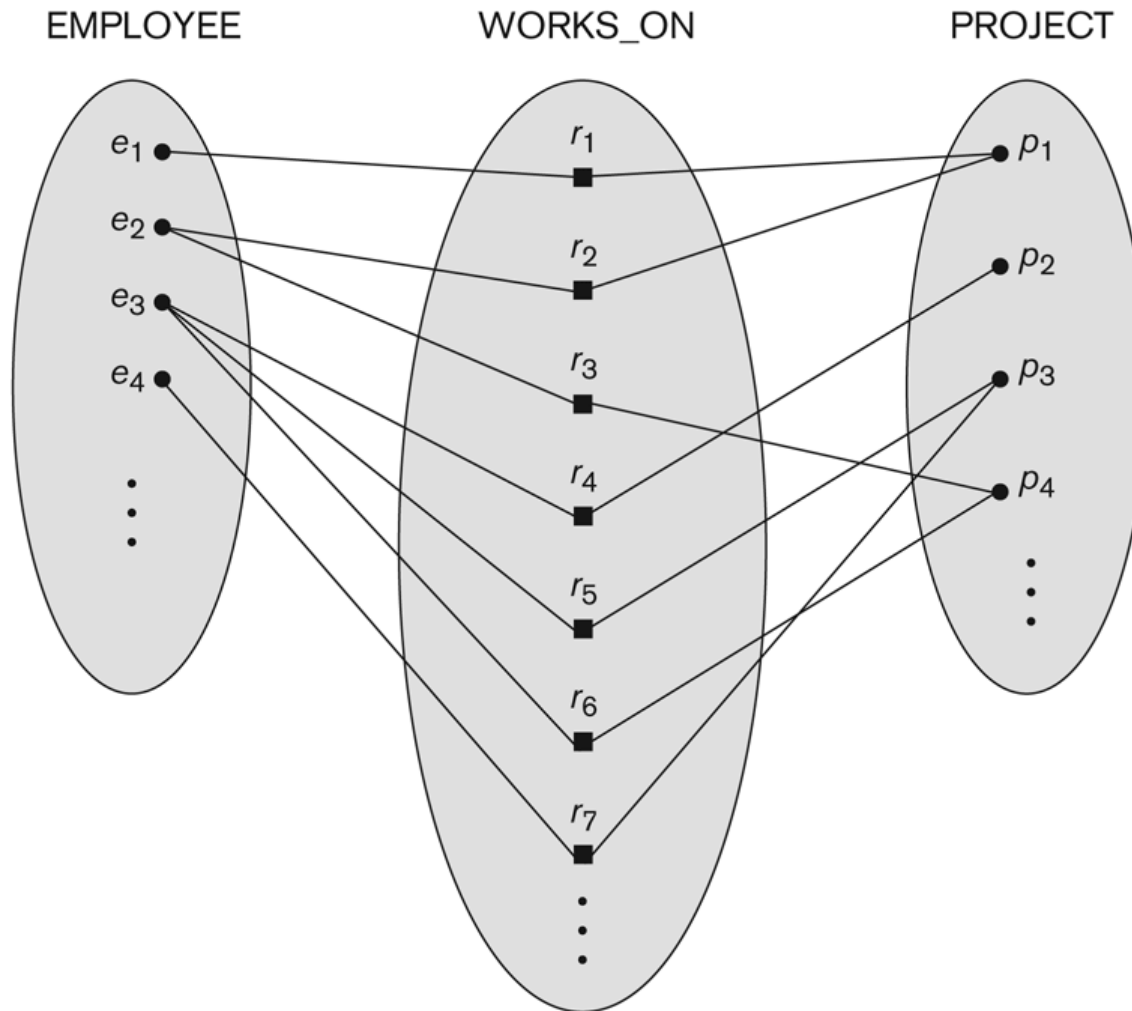


**Figure 3.9**

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



# Relationship instances of the M:N WORKS\_ON relationship between EMPLOYEE and PROJECT



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

# Relationship type vs. relationship set (1)

## ■ Relationship Type:

- Is the schema description of a relationship
- Identifies the relationship name and the participating entity types
- Also identifies certain relationship constraints

## ■ Relationship Set:

- The current set of relationship instances represented in the database
- The current *state* of a relationship type

# Relationship type vs. relationship set (2)

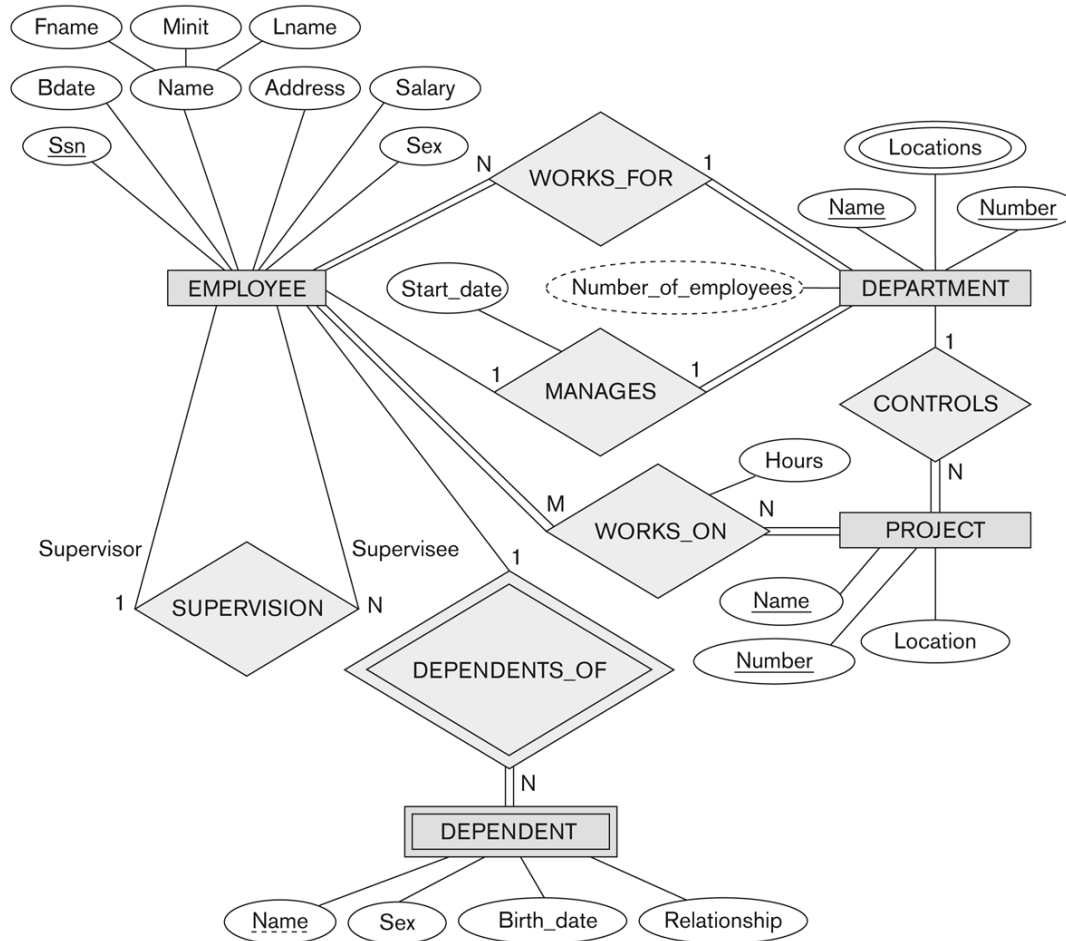
- Previous figures displayed the relationship sets
- Each instance in the set relates individual participating entities – one from each participating entity type
- In ER diagrams, we represent the *relationship type* as follows:
  - Diamond-shaped box is used to display a relationship type
  - Connected to the participating entity types via straight lines
  - Note that the relationship type is not shown with an arrow. The name should be typically be readable from left to right and top to bottom.

# Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships( degree 2)
- Listed below with their participating entity types:
  - WORKS\_FOR (between EMPLOYEE, DEPARTMENT)
  - MANAGES (also between EMPLOYEE, DEPARTMENT)
  - CONTROLS (between DEPARTMENT, PROJECT)
  - WORKS\_ON (between EMPLOYEE, PROJECT)
  - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
  - DEPENDENTS\_OF (between EMPLOYEE, DEPENDENT)

# ER DIAGRAM – Relationship Types are:

WORKS\_FOR, MANAGES, WORKS\_ON, CONTROLS, SUPERVISION, DEPENDENTS\_OF



**Figure 3.2**

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

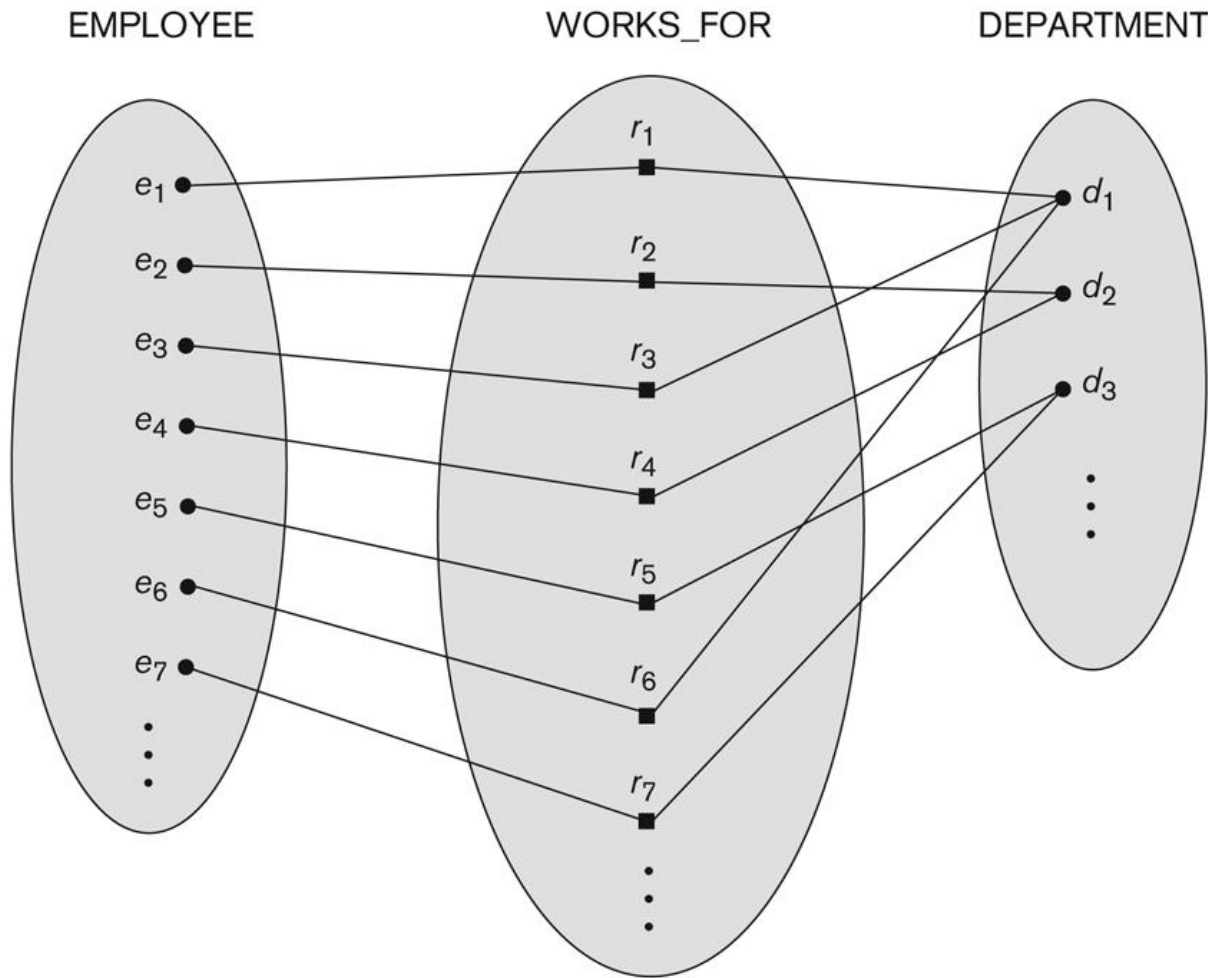
# Discussion on Relationship Types

- In the refined design, some attributes from the initial entity types are refined into relationships:
  - Manager of DEPARTMENT -> MANAGES
  - Works\_on of EMPLOYEE -> WORKS\_ON
  - Department of EMPLOYEE -> WORKS\_FOR
  - etc
- In general, more than one relationship type can exist between the same participating entity types
  - MANAGES and WORKS\_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
  - Different meanings and different relationship instances.

# 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)

# Many-to-one (N:1) Relationship

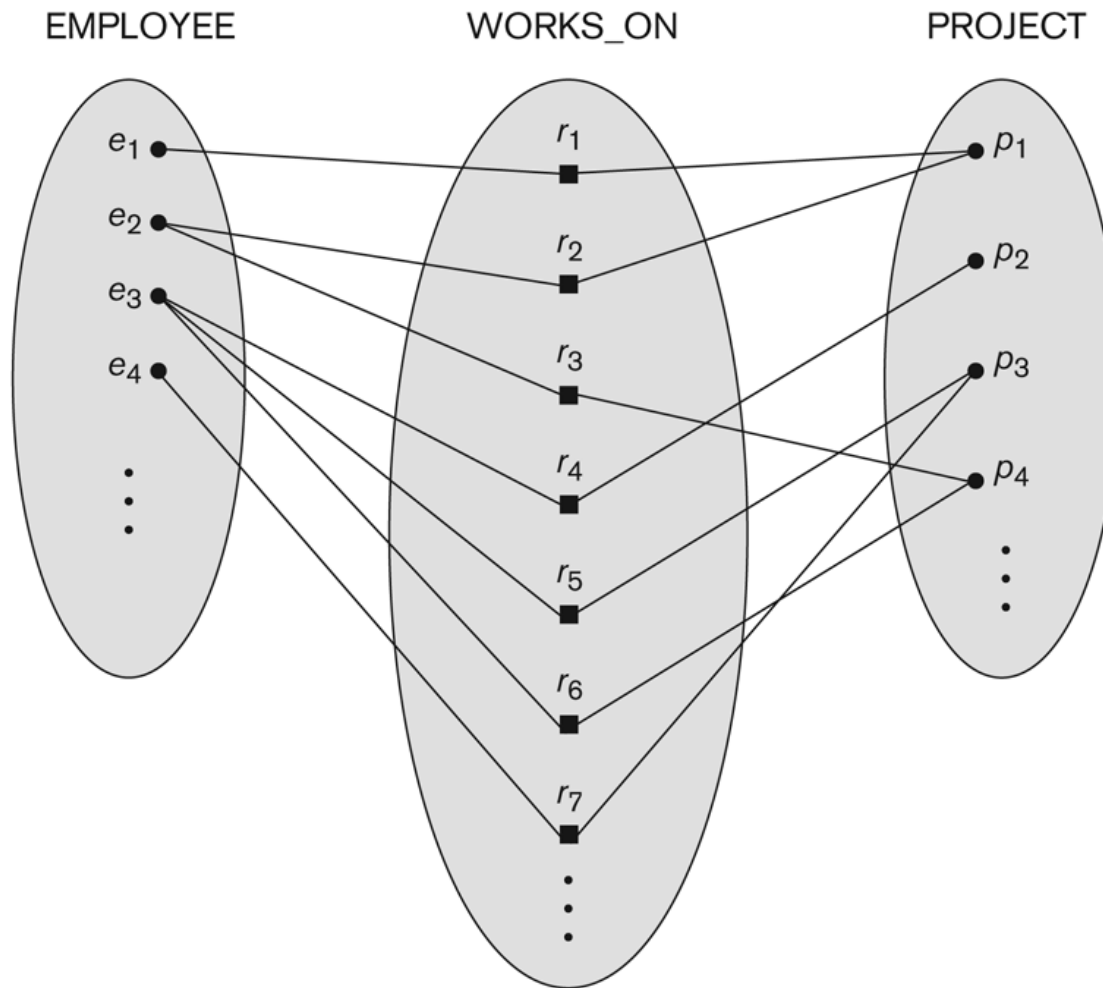


**Figure 3.9**

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



# Many-to-many (M:N) Relationship



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

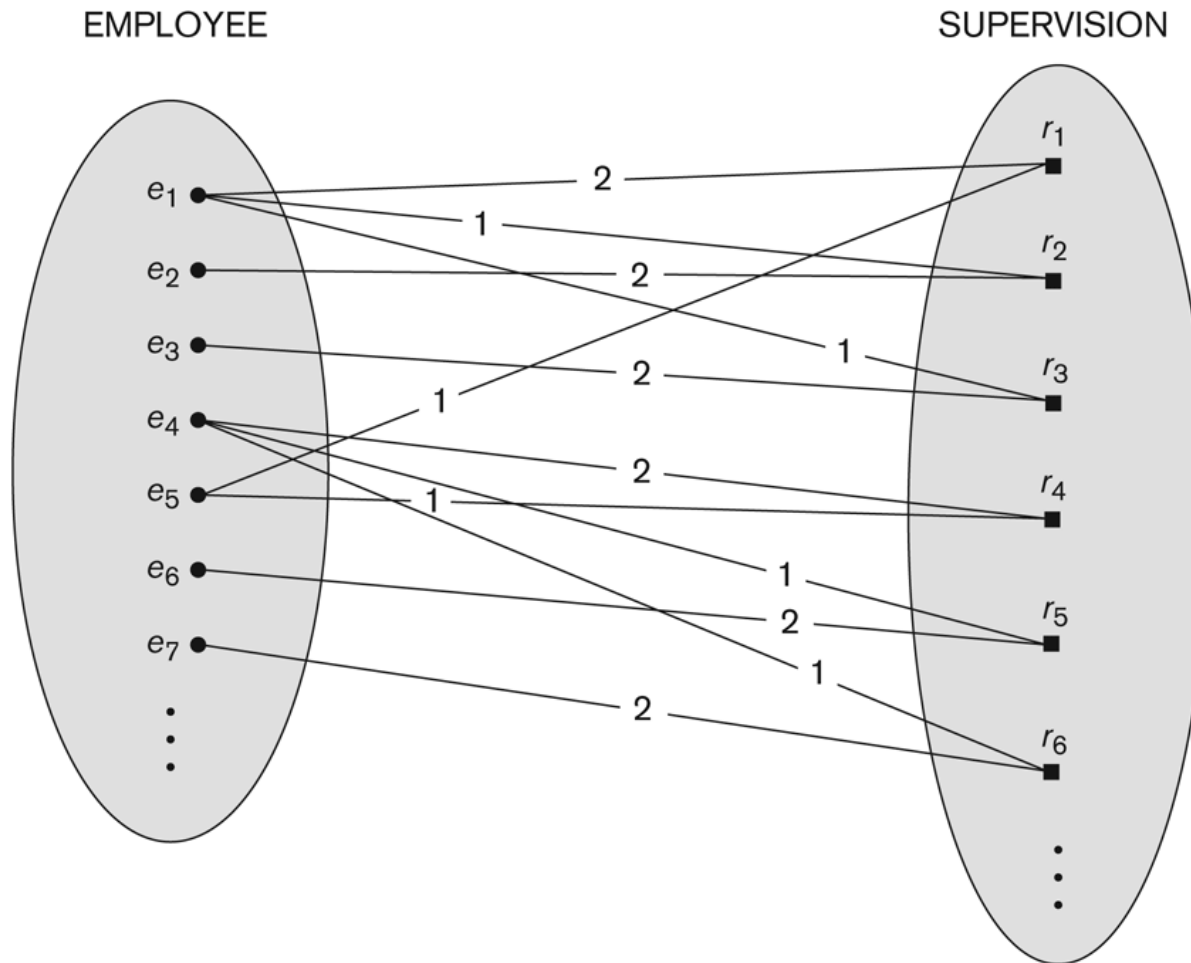
# Recursive Relationship Type

- A relationship type between the same participating entity type in **distinct roles**
- Also called a **self-referencing** relationship type.
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
  - One employee in *supervisor* role
  - One employee in *supervisee* role

# 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.

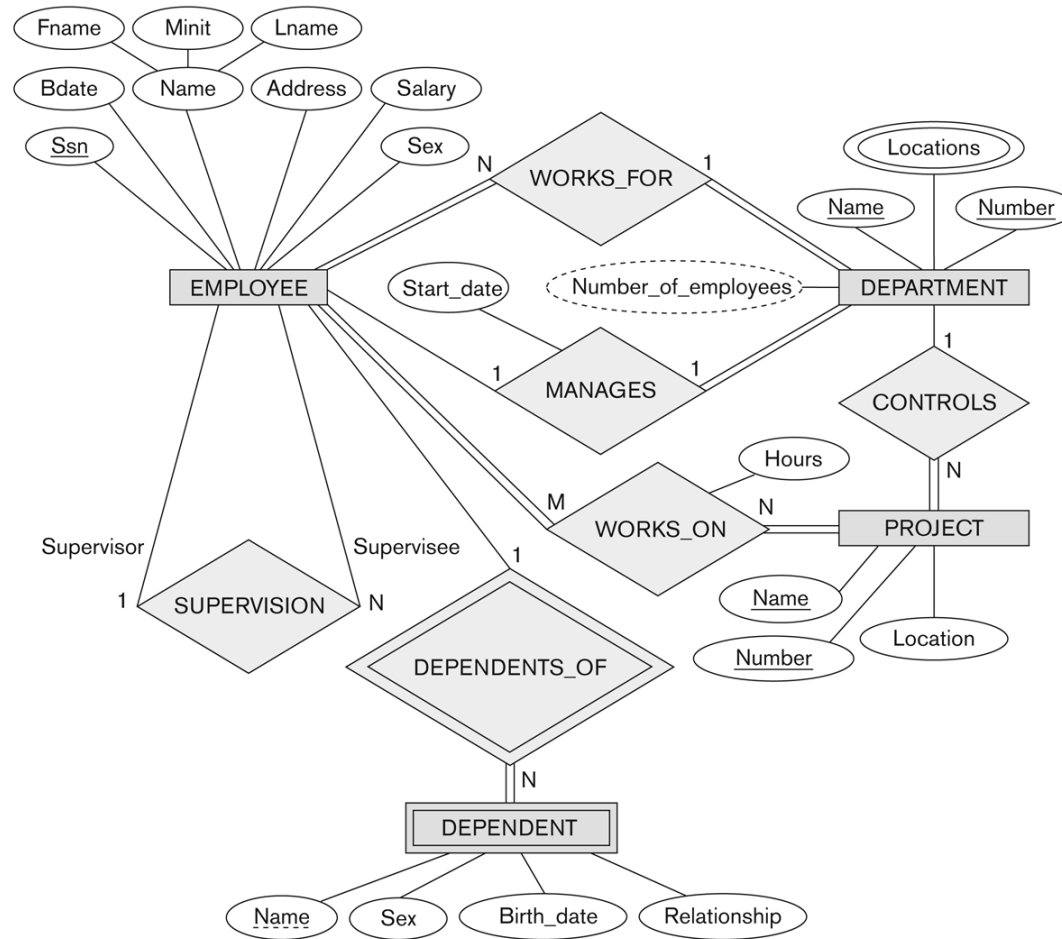
# A Recursive Relationship Supervision`



**Figure 3.11**

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

# Recursive Relationship Type is: SUPERVISION (participation role names are shown)



**Figure 3.2**

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

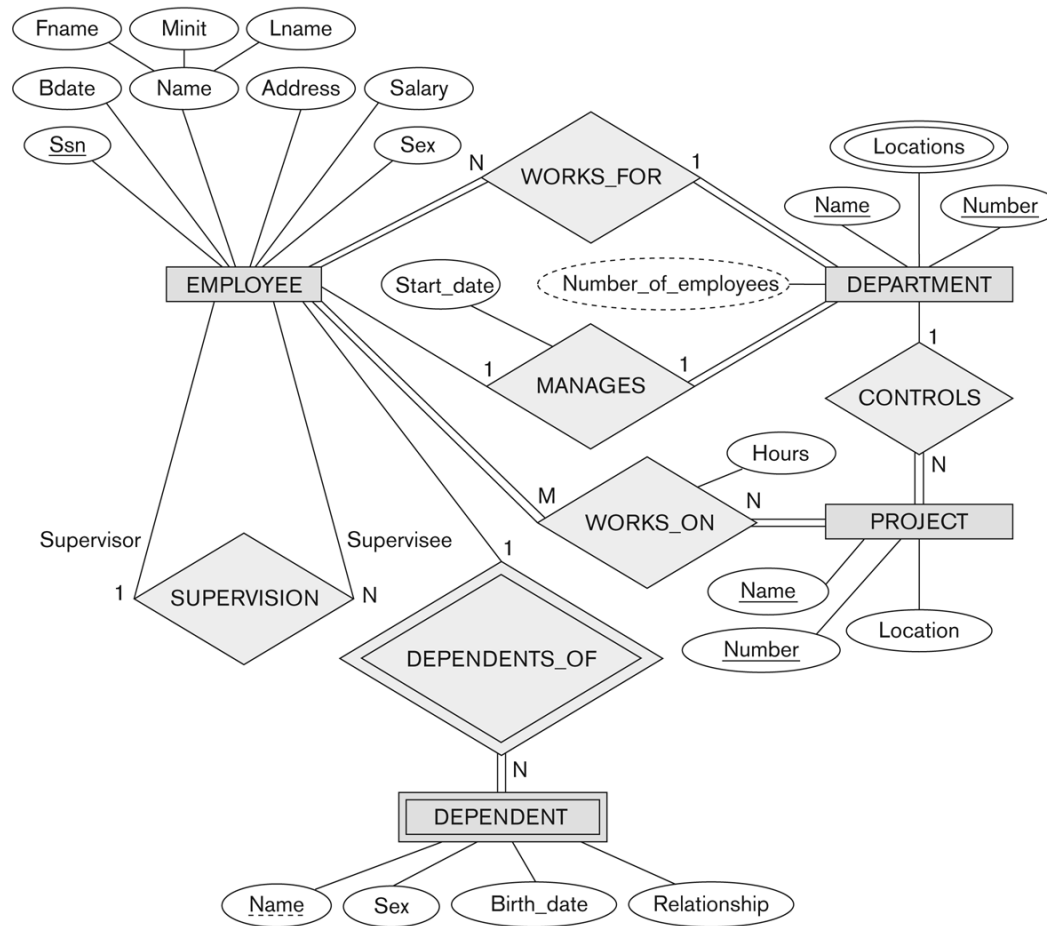
# Weak Entity Types

- An entity that does not have a key attribute and that is identification-dependent on another entity type.
- 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 relationship type
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT\_OF

# Attributes of Relationship types

- A relationship type can have attributes:
  - For example, HoursPerWeek of WORKS\_ON
  - Its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
    - A value of HoursPerWeek depends on a particular (employee, project) combination
- Most relationship attributes are used with M:N relationships
  - In 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

# Example Attribute of a Relationship Type: Hours of WORKS\_ON



**Figure 3.2**

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



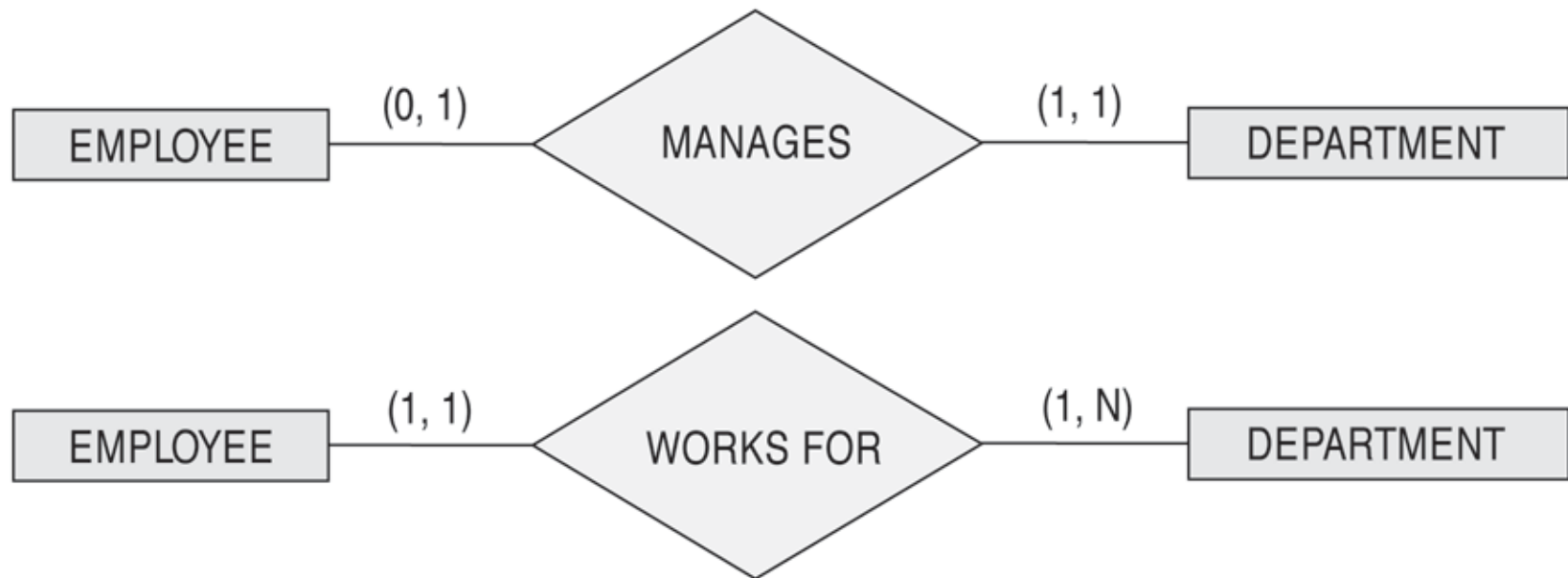
# Notation for Constraints on Relationships

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Shown by placing appropriate numbers on the relationship edges.
- Participation constraint (on each participating entity type): total (called existence dependency) or partial.
  - Total shown by double line, partial by single line.
- NOTE: These are easy to specify for Binary Relationship Types.

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

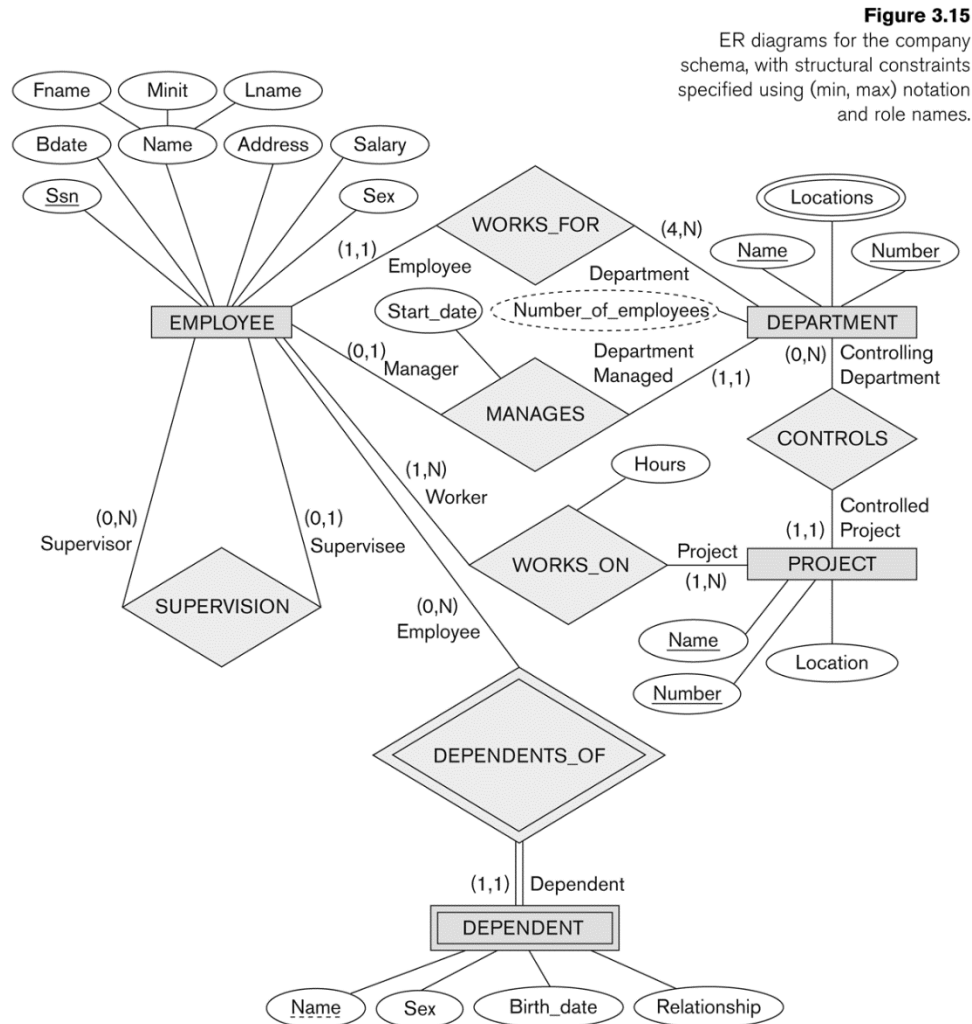
- Specified on each participation of an entity type *E* in a relationship type *R*
- Specifies that each entity *e* in *E* participates in at least *min* and at most *max* relationship instances in *R*
- Default(no constraint):  $\text{min}=0$ ,  $\text{max}=\infty$  (signifying no limit)
- Must have  $\text{min} \leq \text{max}$ ,  $\text{min} \geq 0$ ,  $\text{max} \geq 1$
- 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, $\infty$ ) for participation of DEPARTMENT in WORKS\_FOR

# The (min,max) notation for relationship constraints



Read the min,max numbers next to the entity type and looking **away from** the entity type

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


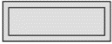





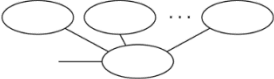

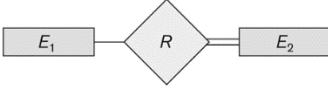
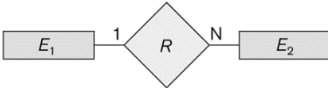
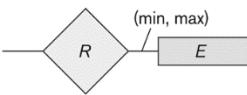


# Alternative diagrammatic notation

- ER diagrams is one popular example for displaying database schemas
- Many other notations exist in the literature and in various database design and modeling tools
- Appendix A illustrates some of the alternative notations that have been used
- UML class diagrams is representative of another way of displaying ER concepts that is used in several commercial design tools

# Summary of notation for ER diagrams

**Figure 3.14**  
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 $R$

# Relationships of Higher Degree

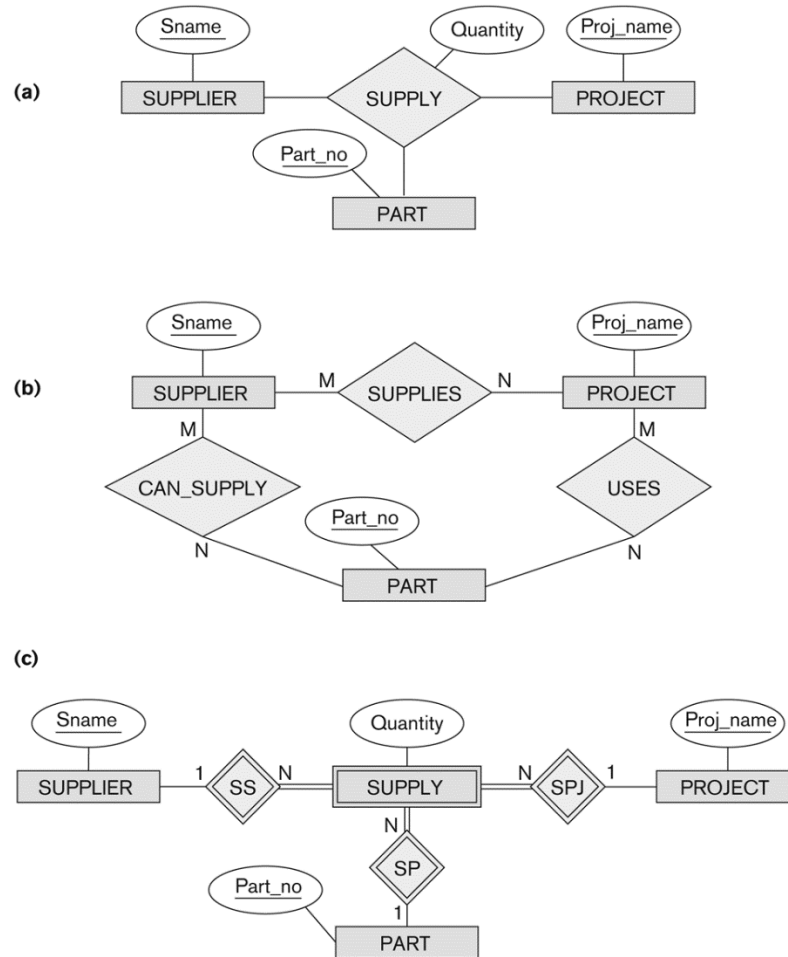
- Relationship types of degree 2 are called binary
- Relationship types of degree 3 are called ternary and of degree  $n$  are called  $n$ -ary
- In general, an  $n$ -ary relationship is not equivalent to  $n$  binary relationships
- Constraints are harder to specify for higher-degree relationships ( $n > 2$ ) than for binary relationships

# Discussion of n-ary relationships ( $n > 2$ )

- In general, 3 binary relationships can represent different information than a single ternary relationship (see Figure 3.17a and b on next slide)
- If needed, the binary and n-ary relationships can all be included in the schema design (see Figure 3.17a and b, where all relationships convey different meanings)
- In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (see Figure 3.17c)



# Example of a ternary relationship



**Figure 3.17**

Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

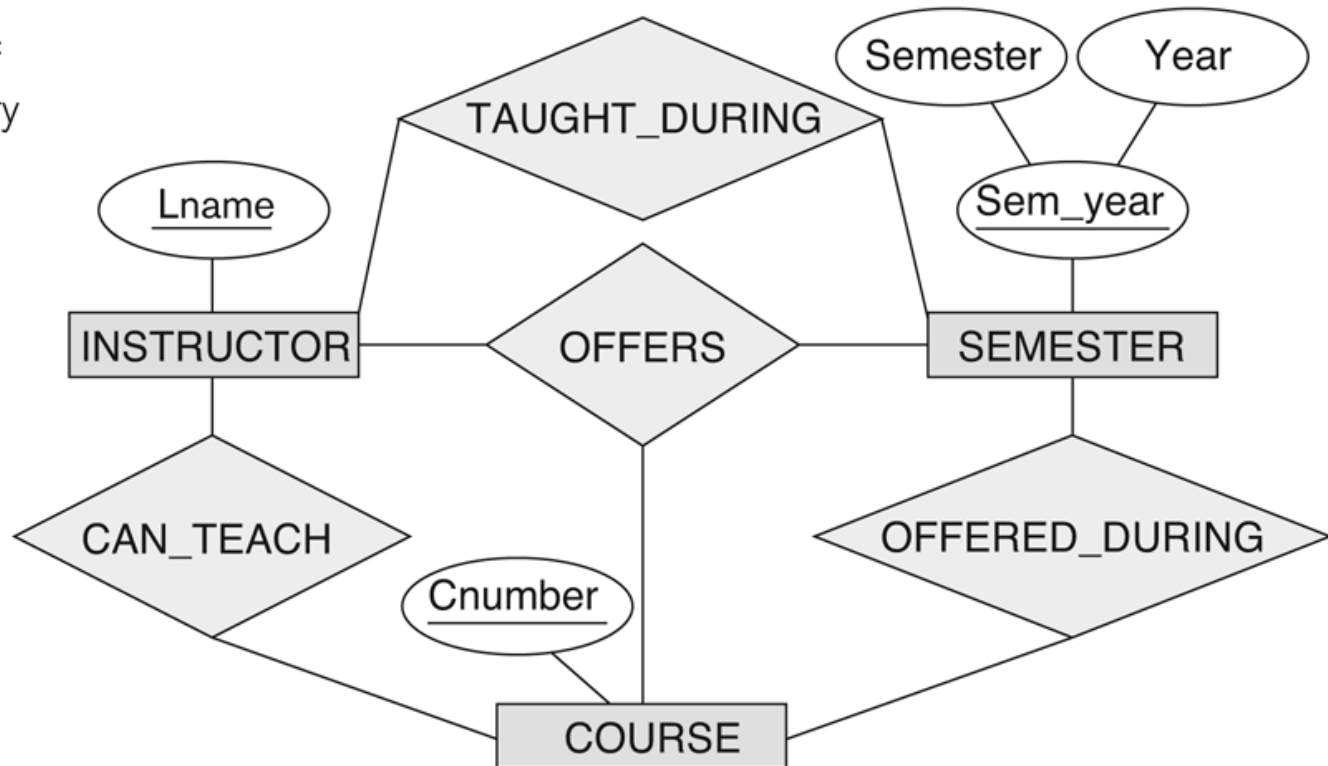
## Discussion of n-ary relationships ( $n > 2$ )

- If a particular binary relationship can be derived from a higher-degree relationship at all times, then it is redundant
- For example, the TAUGHT\_DURING binary relationship in Figure 3.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships)

# Another example of a ternary relationship

**Figure 3.18**

Another example of ternary versus binary relationship types.



# Displaying constraints on higher-degree relationships

- The (min, max) constraints can be displayed on the edges – however, they do not fully describe the constraints
- Displaying a 1, M, or N indicates additional constraints
  - An M or N indicates no constraint
  - A 1 indicates that an entity can participate in at most one relationship instance *that has a particular combination of the other participating entities*
- In general, both (min, max) and 1, M, or N are needed to describe fully the constraints
- Overall, the constraint specification is difficult and possibly ambiguous when we consider relationships of a degree higher than two.

# Another Example: A UNIVERSITY Database

- To keep track of the enrollments in classes and student grades, another database is to be designed.
- It keeps track of the COLLEGES, DEPARTMENTS within each college, the COURSEs offered by departments, and SECTIONS of courses, INSTRUCTORS who teach the sections etc.
- These entity types and the relationships among these entity types are shown on the next slide in Figure 3.20.

# UNIVERSITY database conceptual schema

