# Database Systems

**Chapter # 3**

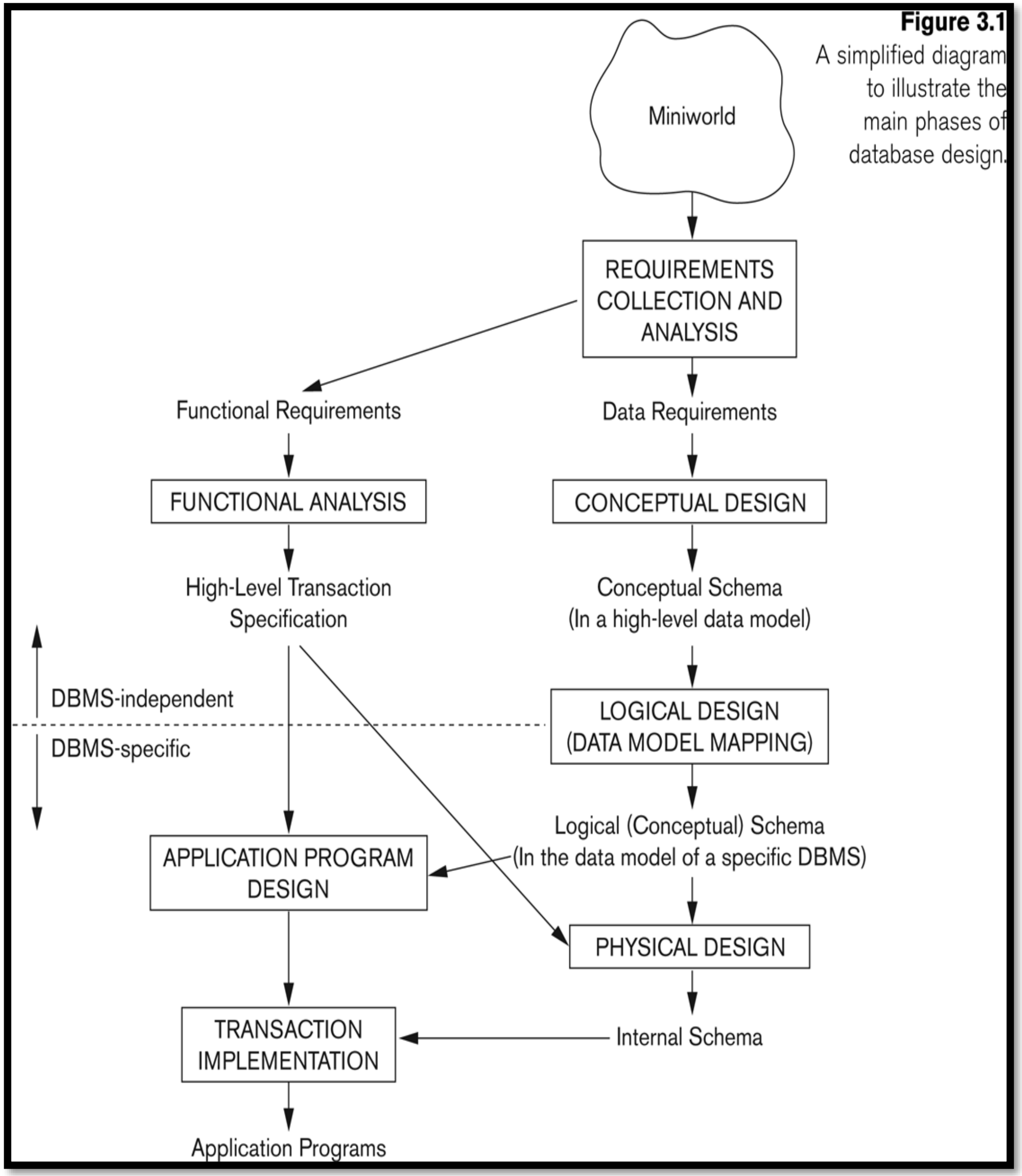
**Data Modeling Using the Entity-Relationship (ER) Model**

# Chapter Outlines

- **Using High-Level Conceptual Data Models for Database Design**
- **A Sample Database Application**
- **Entity Types**
- **Entity Sets Attributes and Keys**
- **Relationship Types**
- **Relationship Sets**
- **Roles, and Structural Constraints**
- **Weak Entity Types**
- **Refining the ER Design for the COMPANY Database**
- **ER Diagrams Naming Conventions, and Design Issues**
- **Relationship Types of Degree Higher than Two**

# Entity–relationship (ER) model

- Is a popular high-level conceptual data model.
- Frequently used for the conceptual design of database applications



Miniworld

REQUIREMENTS COLLECTION AND ANALYSIS

Functional Requirements

Data Requirements

FUNCTIONAL ANALYSIS

CONCEPTUAL DESIGN

High-Level Transaction Specification

Conceptual Schema (In a high-level data model)

DBMS-independent

DBMS-specific

LOGICAL DESIGN (DATA MODEL MAPPING)

Logical (Conceptual) Schema (In the data model of a specific DBMS)

APPLICATION PROGRAM DESIGN

PHYSICAL DESIGN

TRANSACTION IMPLEMENTATION

Internal Schema

Application Programs

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

# ER concepts

- The ER model describes data as entities, relationships, and attributes
- **Entities and Their Attributes:**
  - **Entity:** which is a thing or object in the real world with an independent existence.
  - Can be an object with a physical existence: i.e. a particular person, car, house, or employee)
  - Can be an object with a conceptual existence: i.e. a company, a job, or a university course)

# ER concepts

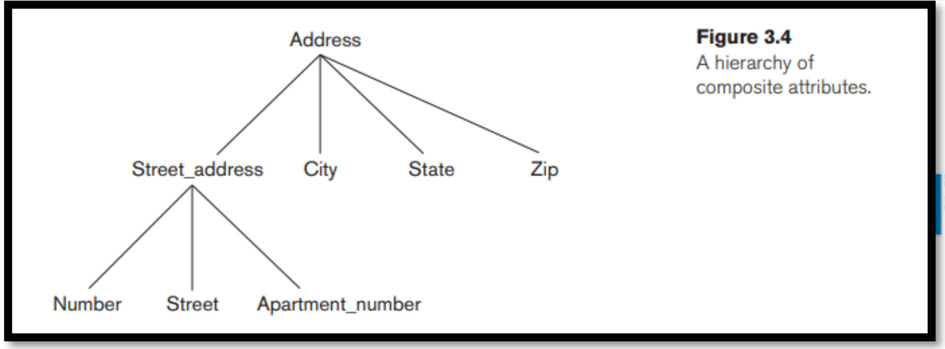


**Figure 3.4**
A hierarchy of composite attributes.

- **Attributes:** Characteristics or Properties of Entity
  - i.e. Employee’s First name, Last name, address etc.
- Simple Attributes: are atomic
- Composite attributes: divided into smaller subparts
  - form a hierarchy
  - i.e. the Address attribute of the EMPLOYEE(Street_address, City, State, and Zip)
  - First name and last name of employee.

# ER concepts

- **Single-Valued Attributes:** have a single value for a particular entity.
  - i.e. Age is a single-valued attribute of a person.
- **Multivalued Attributes:** have a set of values for the same entity.
  - i.e. Car can have more than two colors
  - i.e. Person can have one or more degrees in college
  - Have lower and upper bounds to constrain the number of values allowed for each individual entity. i.e. car can have two colors at most.

# Summary of Notation for ER Diagrams

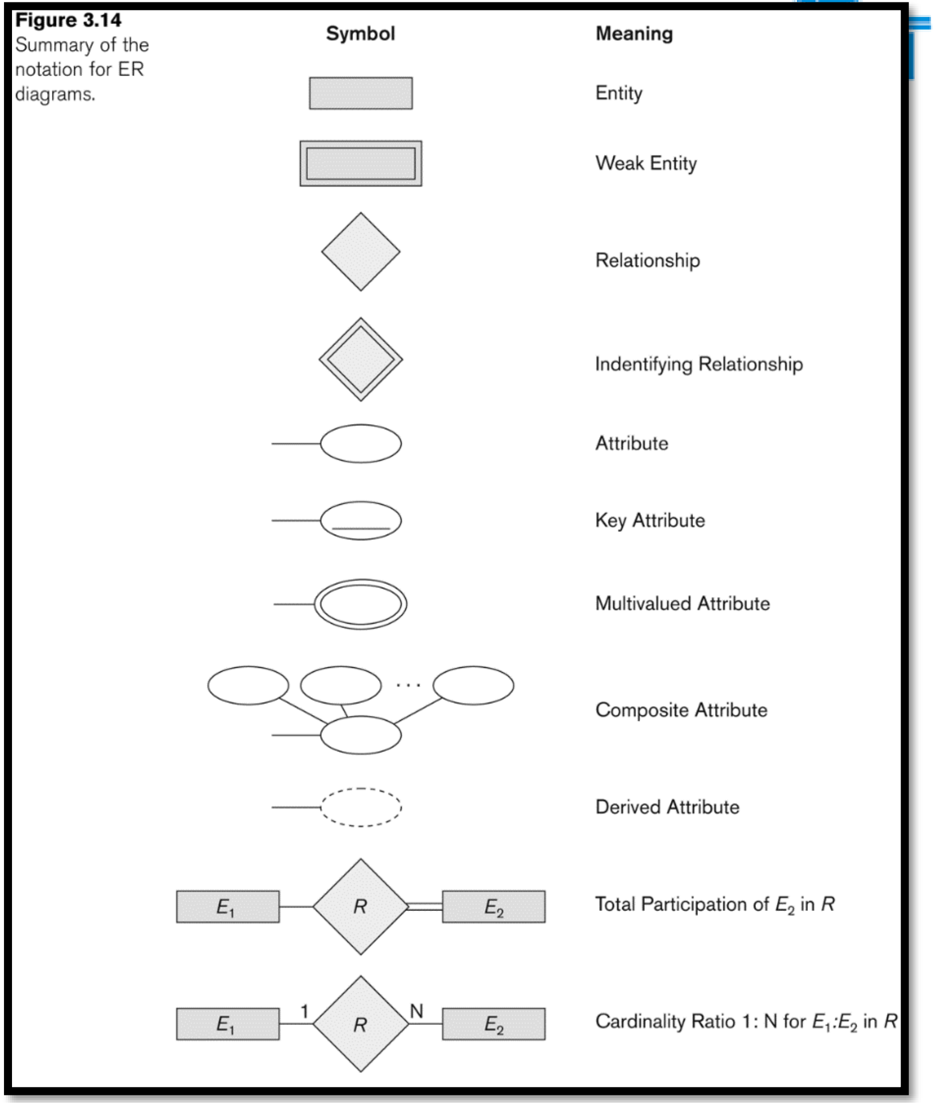
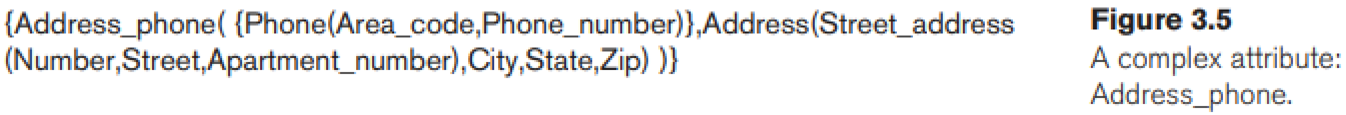


**Figure 3.14**
Summary of the notation for ER diagrams.

| Symbol | Meaning |
|---|---|
| | Entity |
| | Weak Entity |
| | Relationship |
| | Indentifying Relationship |
| | Attribute |
| | Key Attribute |
| | Multivalued Attribute |
| . . . | Composite Attribute |
| | Derived Attribute |
| $E_1$ $R$ $E_2$ | Total Participation of $E_2$ in $R$ |
| $E_1$ 1 $R$ N $E_2$ | Cardinality Ratio 1: N for $E_1$:$E_2$ in $R$ |

# ER concepts

- **Derived Attributes:** two (or more) attribute values are related.
  - i.e. Age can be derived from birthdate
  - So, age is a derived attribute.
- **Stored Attribute:** that are already specified but not derived
- **Complex Attributes:**
  - i.e. a person can have more than one residence and each residence can have a single address and multiple phones



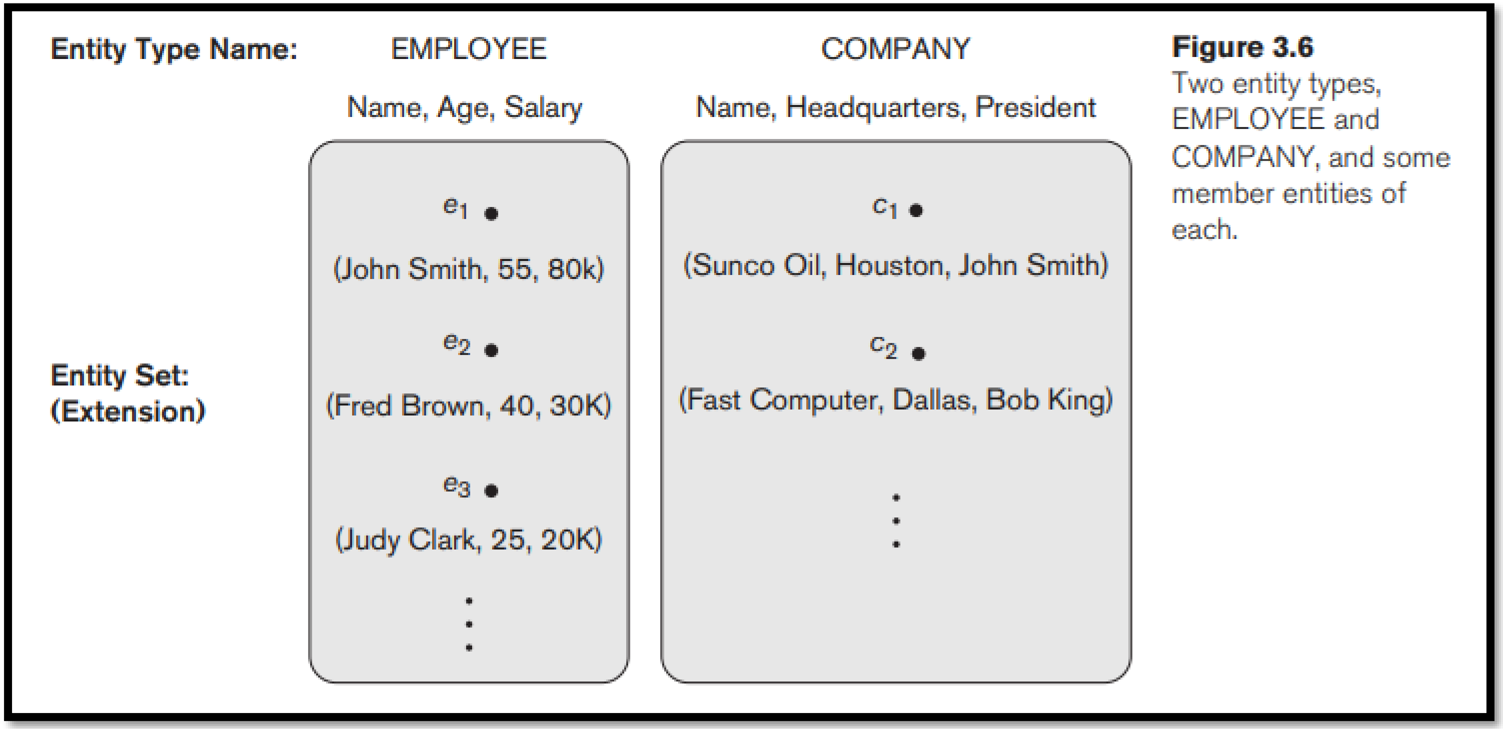
{Address_phone( {Phone(Area_code,Phone_number)},Address(Street_address (Number,Street,Apartment_number),City,State,Zip) )}

**Figure 3.5**
A complex attribute: Address_phone.

# ER concepts

- **Entity Types:** Different Entities
  - Specifies Schema or intension
  - i.e. EMPLOYEE and COMPANY
- **Entity Set:** The collection of all entities of a particular entity type in the database at any point in time is called an entity set or entity collection
  - Specifies Collection of entities: Extension
  - i.e. a company employing hundreds of employees
  - These employee entities share the same attributes, but each entity has its own value(s) for each attribute.

# ER concepts



**Entity Type Name:** EMPLOYEE COMPANY

Name, Age, Salary

Name, Headquarters, President

**Entity Set:**
**(Extension)**

$e_1$ ●
(John Smith, 55, 80k)

$e_2$ ●
(Fred Brown, 40, 30K)

$e_3$ ●
(Judy Clark, 25, 20K)

⋮

$c_1$ ●
(Sunco Oil, Houston, John Smith)

$c_2$ ●
(Fast Computer, Dallas, Bob King)

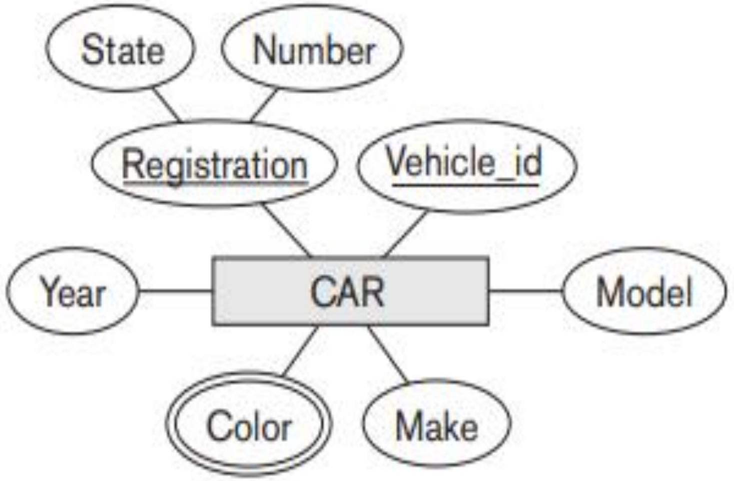
⋮

**Figure 3.6**
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

# ER Concepts

- **Key Attributes of an Entity Type:** distinct for each individual entity in the entity set. and its values can be used to identify each entity uniquely
- **For example,** the Name attribute is a key of the COMPANY entity type because no two companies are allowed to have the same name.
- A key attribute may be composite.
  - **i.e.,** Registration is a key of the CAR entity type with components (Number, State).
- **An entity type may also have no key, in which case it is called a weak entity type.**

(a)



(b)

CAR

Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR$_1$

((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR$_2$

((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR$_3$

((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

**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.

# ER Concepts

- **Value Sets (Domains) of Attributes:**
  - Values (or domain of values) represent the data from the current state
  - Value sets are not typically displayed in basic ER diagrams
  - similar to the basic data types available in most programming languages, such as integer, string, Boolean, float, subrange, and so on.

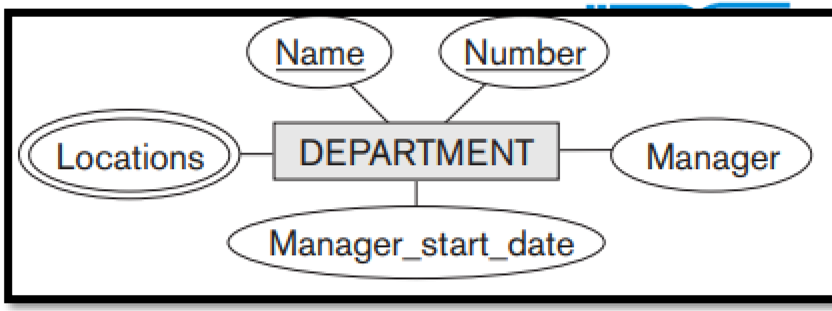
# A Sample Database Application

- The COMPANY database keeps track of a company’s employees, departments, and projects.
- The company is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.

# A Sample Database Application

- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- The database will store each employee’s name, a unique Social Security number, address, salary, sex (gender), and birth date.
- An employee is assigned to one department, but may work on several projects, which are not necessarily controlled by the same department. It is required to keep track of the current number of hours per week that an employee works on each project, as well as the direct supervisor of each employee (who is another employee).
- The database will keep track of the dependents of each employee for insurance purposes, including each dependent’s first name, sex, birth date, and relationship to the employee.

# Initial Conceptual Design of the COMPANY Database



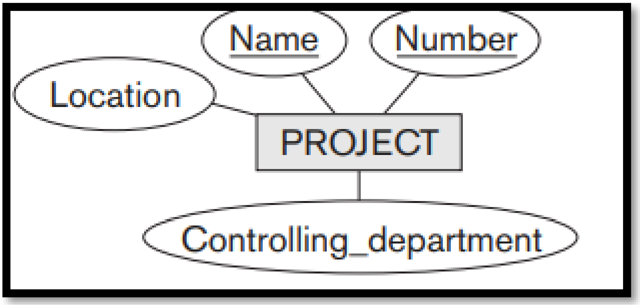
- Four entity types:

1. Entity type: DEPARTMENT
   - Attributes: Name, Number, Locations, Manager, and Manager_start_date.
   - Locations is the only multivalued attribute.
   - key attributes: Name and Number because each was specified to be unique.

2. Entity type: PROJECT

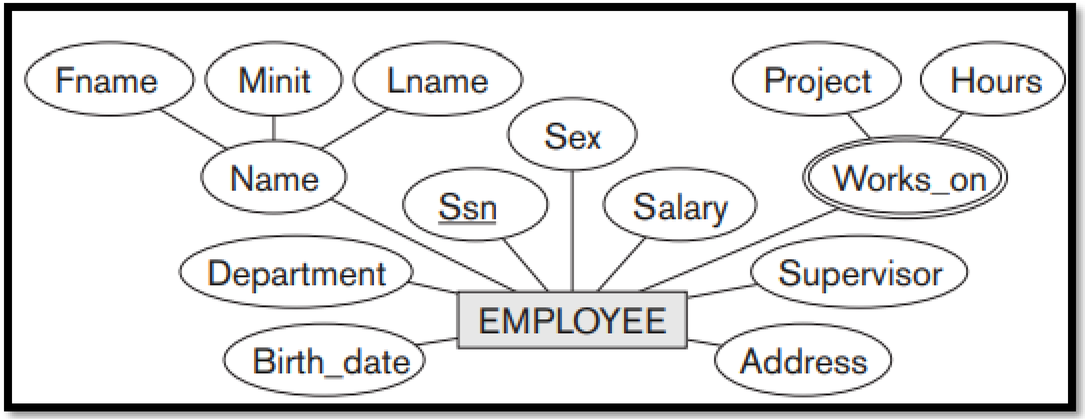
Attributes: Name, Number, Location, and Controlling_department.

Key Attributes: Both Name and Number are (separate) key attributes.



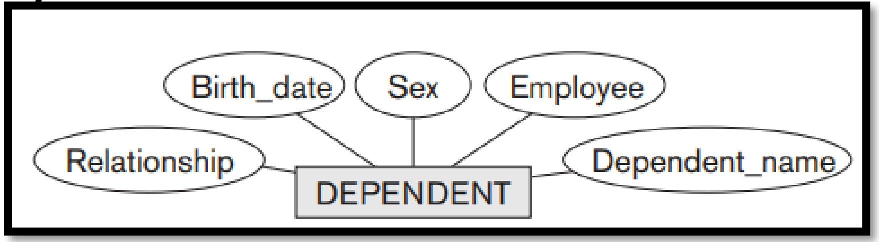
# Initial Conceptual Design of the COMPANY Database

- **Entity type: EMPLOYEE**
  - Attributes: Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor
  - Composite Attributes: Name and Address may be composite attributes; however, this was not specified in the requirements.



# Initial Conceptual Design of the COMPANY Database

- **Entity type: DEPENDENT**
- Attributes: Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).
- Multivalued composite attribute of EMPLOYEE: Works_on (Project, Hours)
- or alternatively



- Multivalued composite attribute of PROJECT: Workers (Employee, Hours).
- We choose the first alternative in Figure 3.8;

# Relationship Types

- **Relationship Type:**
  - Is the schema description of a relationship
  - Identifies the relationship name and the participating entity types
  - Also identifies certain relationship constraints.
  - For example, the attribute Manager of DEPARTMENT refers to an employee who manages the department;
  - the attribute Controlling_department of PROJECT refers to the department that controls the project;
  - the attribute Supervisor of EMPLOYEE refers to another employee (the one who supervises this employee);
  - the attribute Department of EMPLOYEE refers to the department for which the employee works; and so on.
- **Relationship Set:**
  - The current set of relationship instances represented in the database

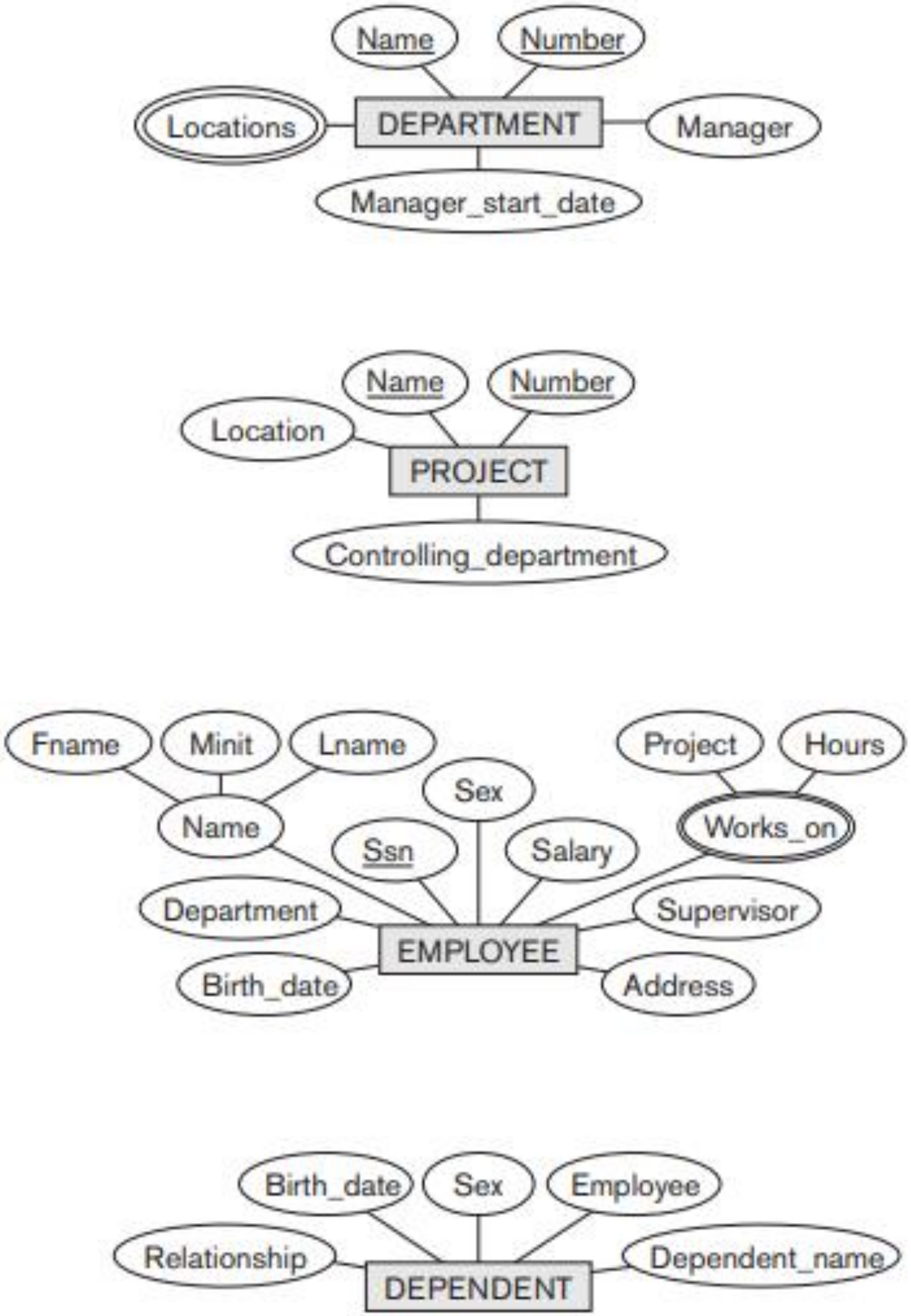
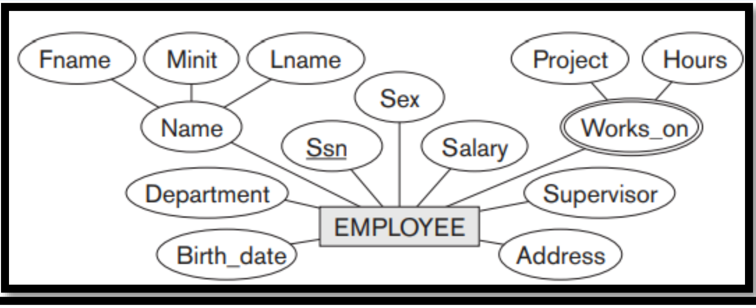


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.



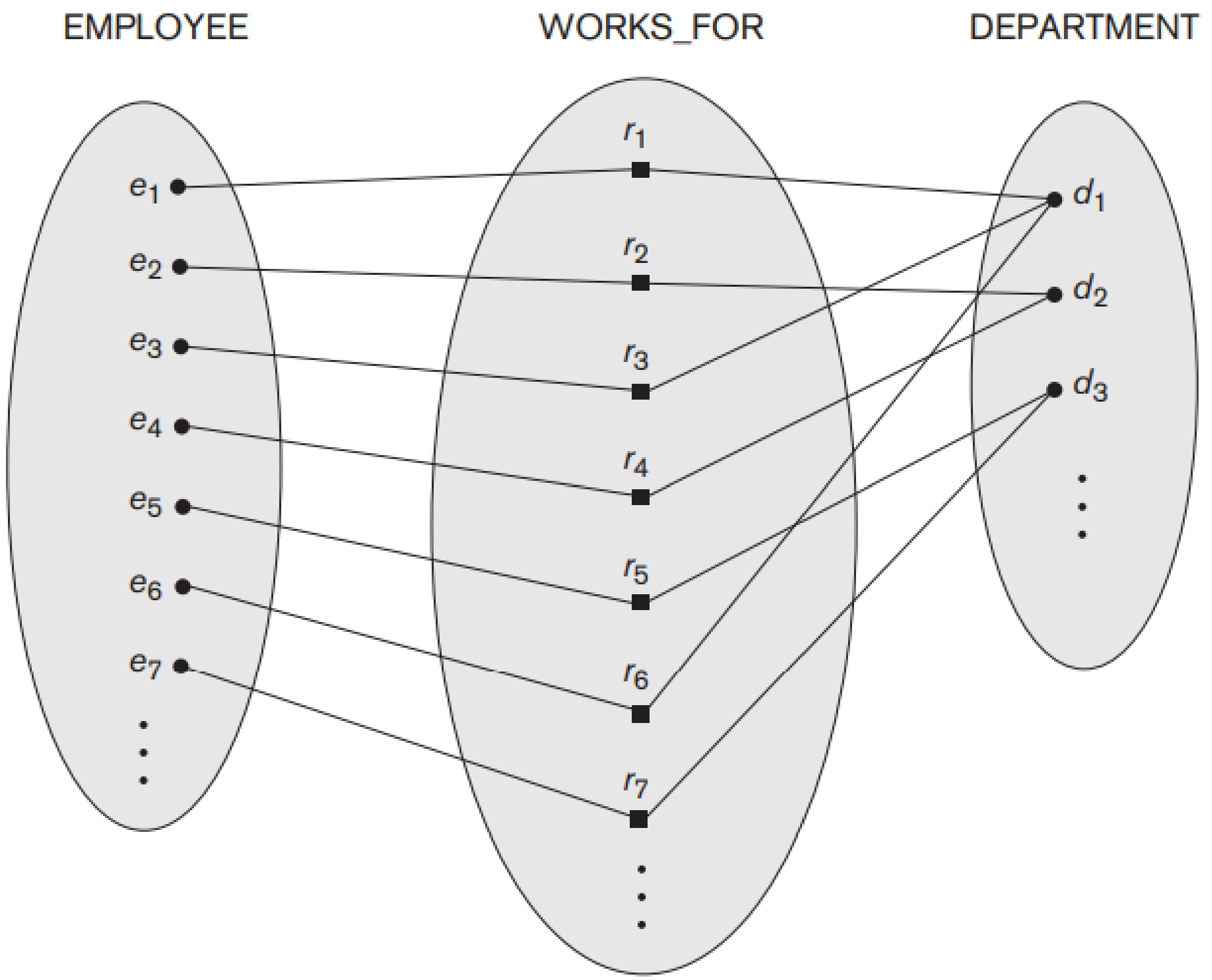


**Figure 3.9**
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

# Relationship Types

- In ER diagrams, relationship types are displayed as diamond-shaped boxes which are connected by straight lines to the rectangular boxes representing the participating entity types.
- The relationship name is displayed in the diamond-shaped box

# Relationship Degree

- **Degree of a Relationship Type:** number of participating entity types.
  - i.e. WORKS_FOR relationship is of degree two (binary).
  - Binary degree of relationship is most common.
- Recursive relationships or self-referencing relationships: same entity type participates more than once in a relationship type in different roles

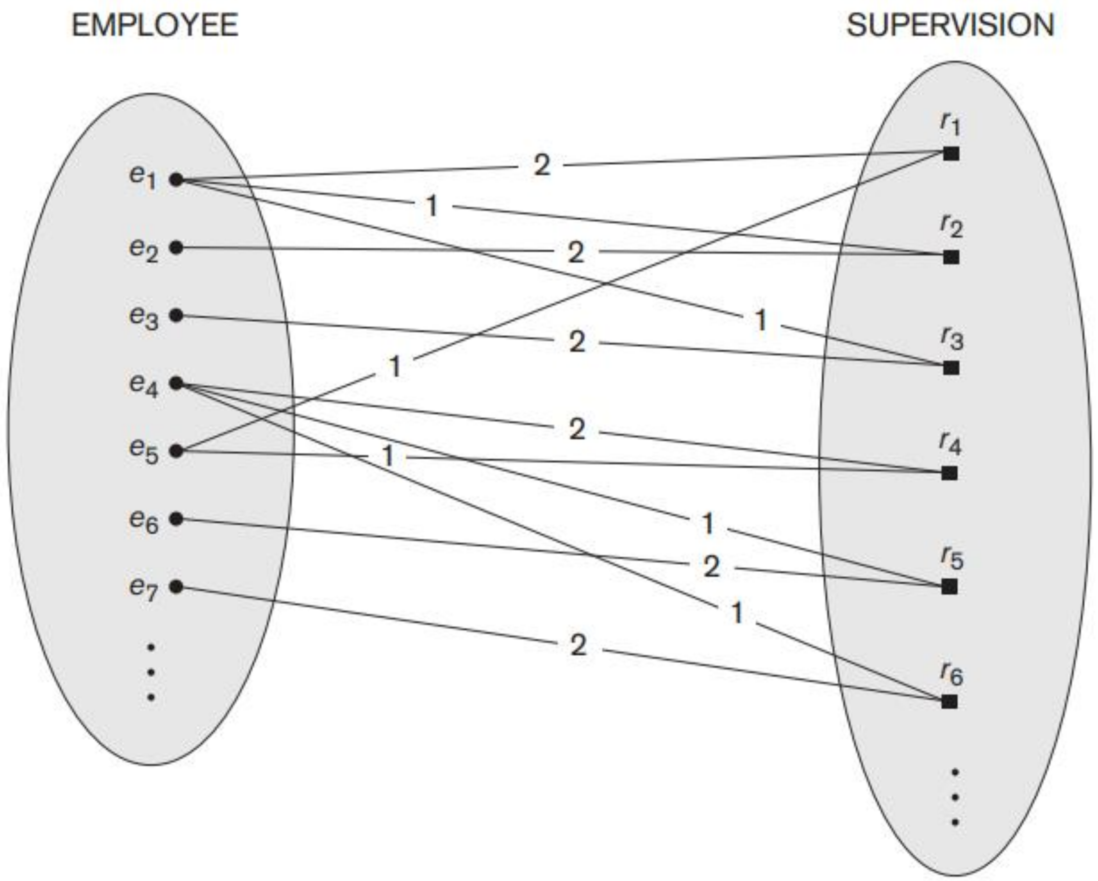


**Figure 3.11**
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

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

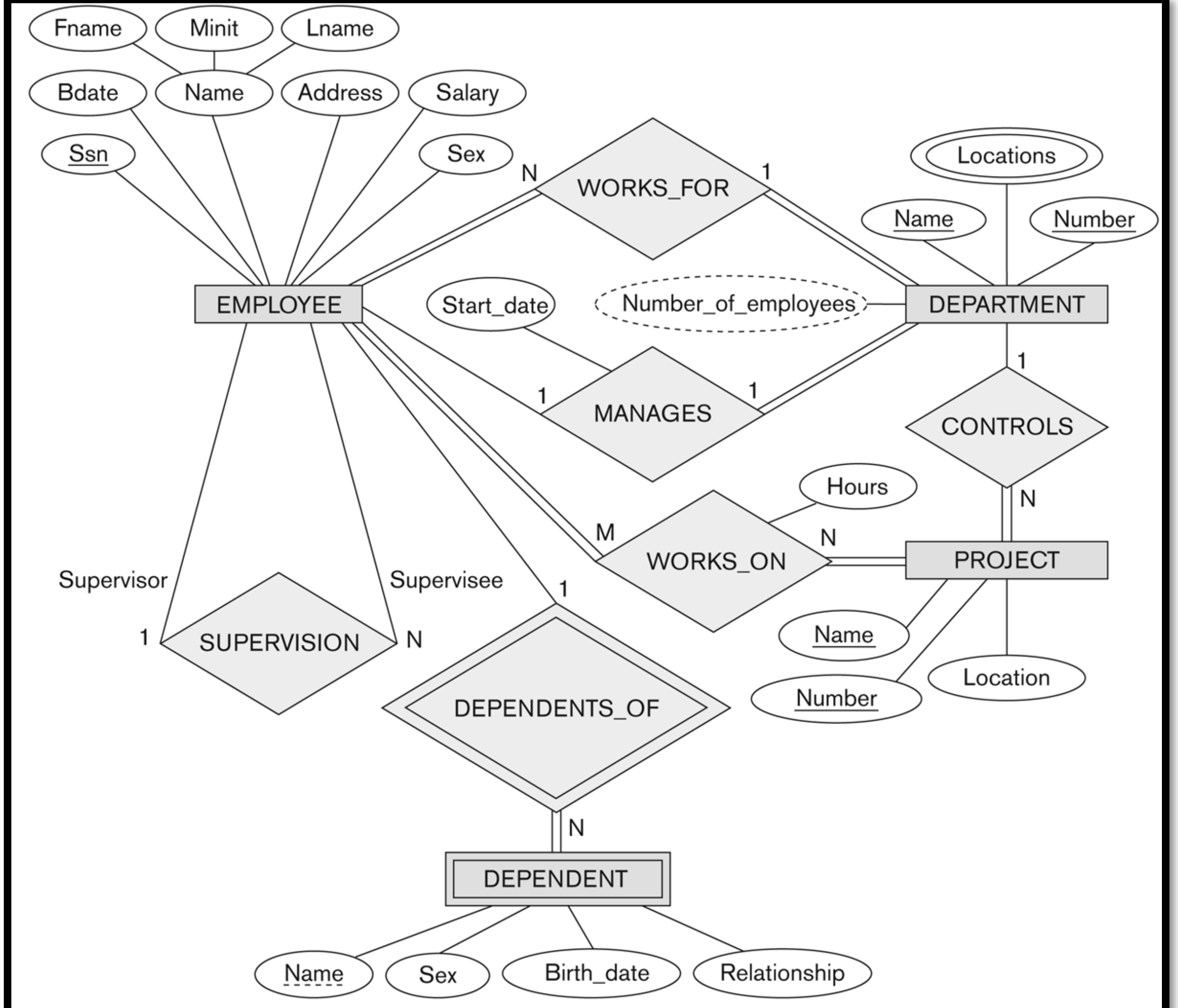
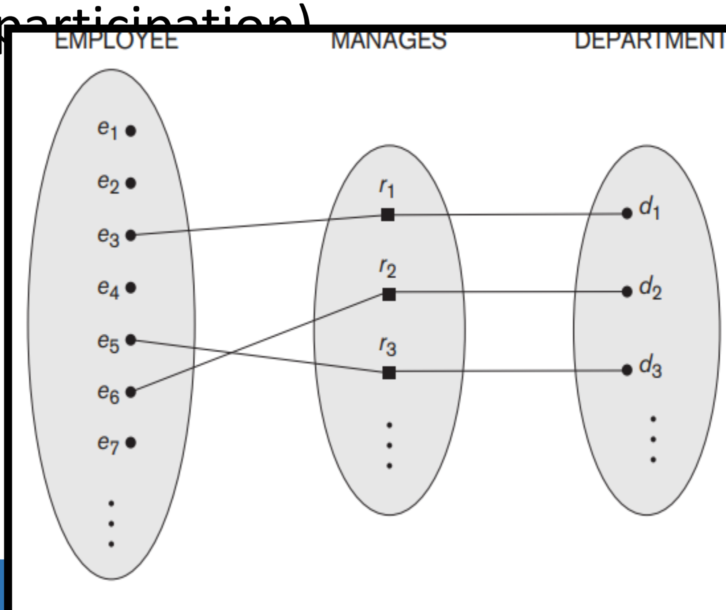


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Constraints on Binary Relationship Types

- **Constraint:** each employee must work for exactly one department.
- **Cardinality Ratios for Binary Relationships.**
  - Maximum number of relationship instances that an entity can participate in.
  - 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)



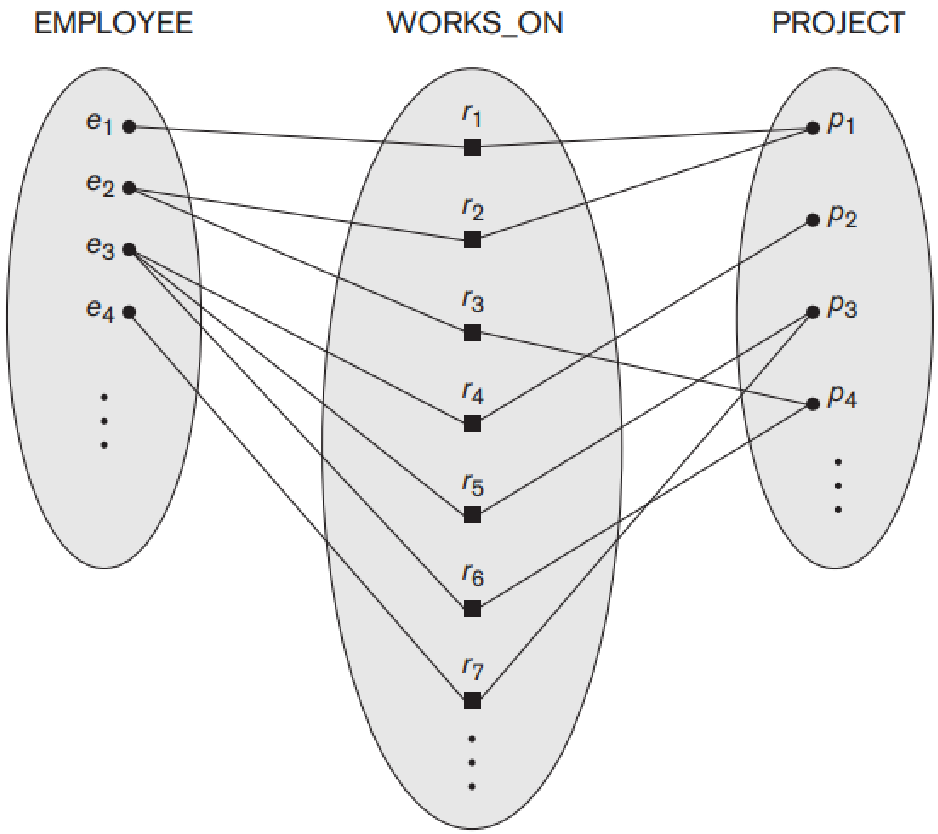


**Figure 3.13**
An M:N relationship,
WORKS_ON.

# Participation Constraints and Existence Dependencies

- **Participation constraint:** specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- There are two types of participation constraints
  - Total Participation/ Existence: Employee must work for department otherwise the employee entity does not exist.
  - Partial Participation: department must have a single manager to manage the department. Manager instances comes from the employee instance set.
- Structural Constraints: cardinality ratio and participation constraints.

# Attributes of Relationship Types

- Relationship types can also have attributes.

1:1 (attribute can move to any of the two entities)
1:N or N:1 (attribute will move to the entity with N)
M:N (new relation will contain the attributes of both entities)

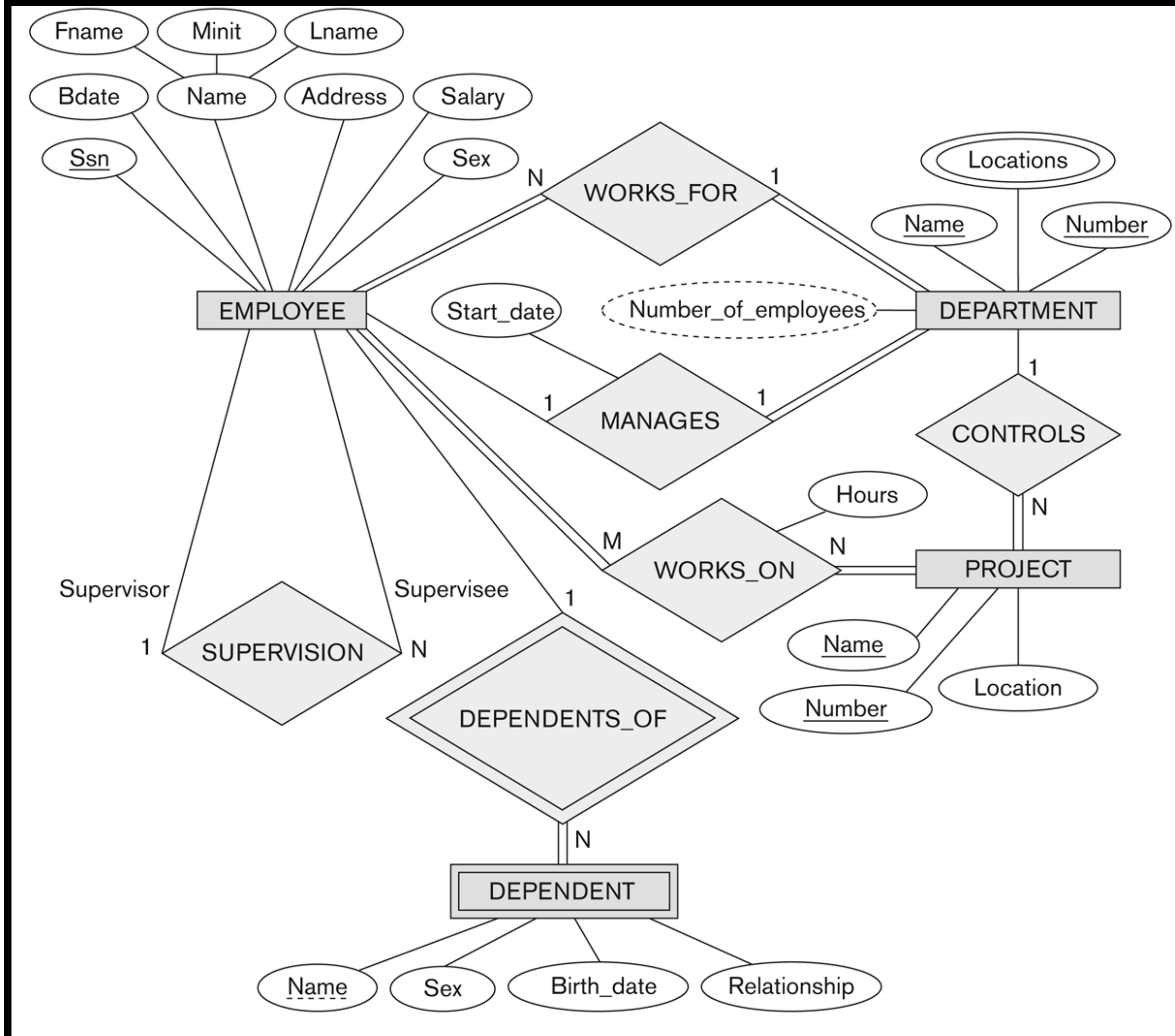


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Attributes of Relationship Types

- **Attributes of 1:1 or 1:N** relationship types can be migrated to one of the participating entity types.
- For a 1:N relationship type, a relationship attribute can be migrated only to the entity type on the N-side of the relationship.
- For M:N (many-to-many) relationship types,
    - Some attributes may be determined by the combination of participating entities, not by any single entity.
    - Such attributes must be specified as relationship attributes. An example is the Hours attribute of the M:N relationship WORKS_ON

# Attributes of Relationship Types

- The number of hours per week an employee currently works on a project is determined by an employee-project combination and not separately by either entity.

# Weak Entity Types

- **Weak entity types:** Entity types that do not have key attributes of their own and their identification is dependent on another entity type..
- **Strong entity types:** that do have a key attribute.
- Always has a total participation constraint (existence dependency) with respect to its identifying relationship:
  - A weak entity cannot be identified without an owner entity.

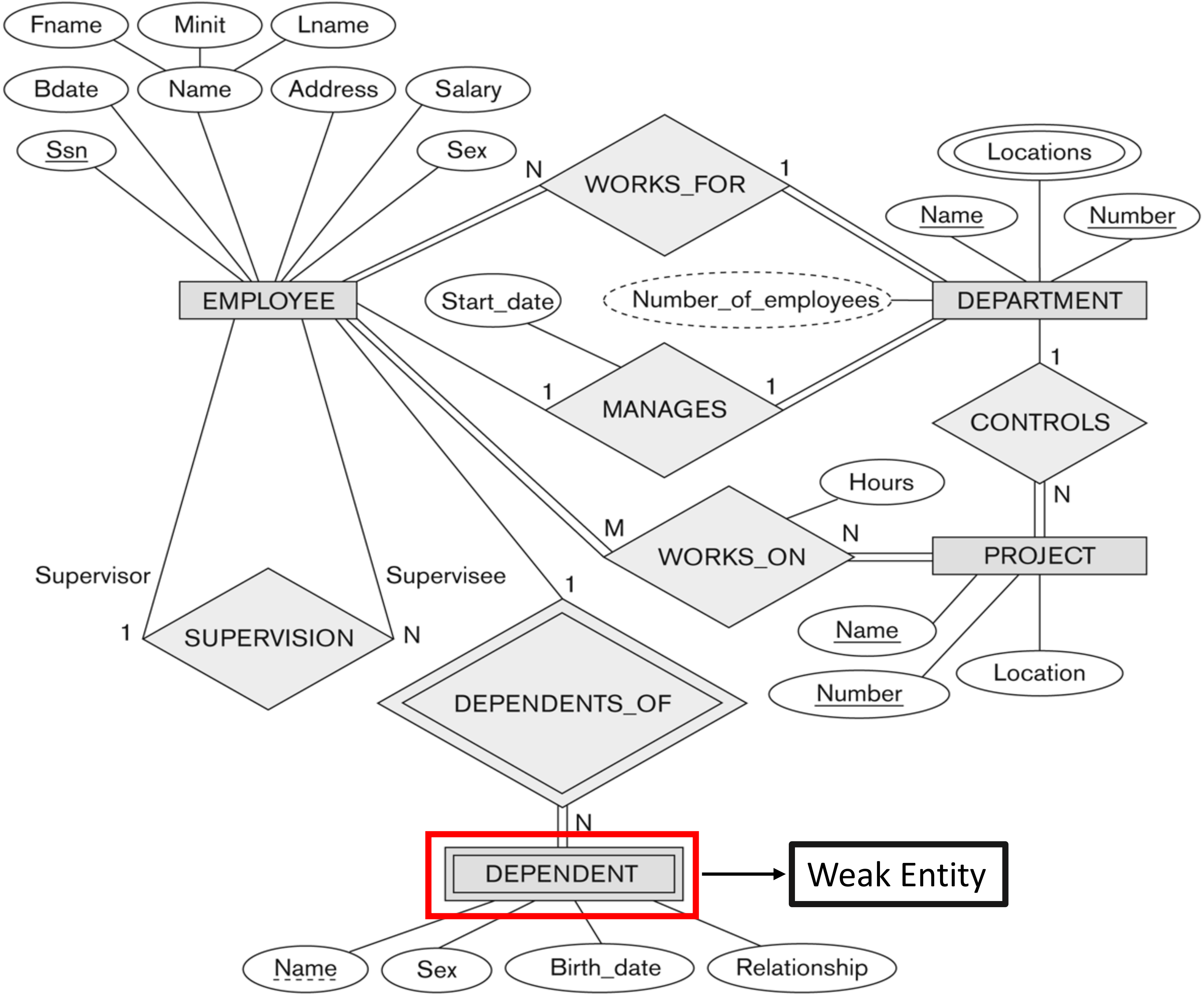


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Weak Entity Types

- A weak entity type normally has a partial key, which is the attribute that can uniquely identify weak entities. i.e., Name, Birthdate etc.
- **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

# Refining the ER Design for the COMPANY Database

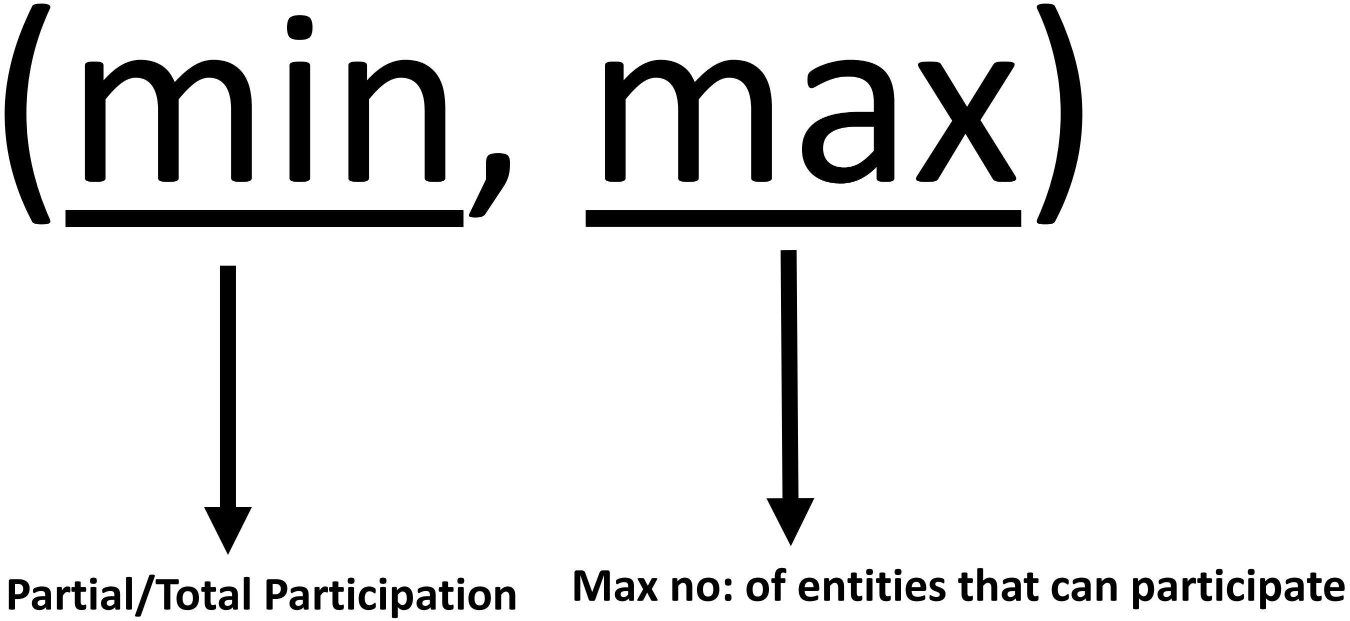
- **WORKS_FOR**, a 1:N (one-to-many) relationship type between DEPARTMENT and EMPLOYEE.
- **CONTROLS,** a 1:N relationship type between DEPARTMENT and PROJECT.
- **SUPERVISION**, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role)
- **WORKS_ON,** determined to be an M:N (many-to-many) relationship type with attribute Hours.
- **DEPENDENTS_OF,** a 1:N relationship type between EMPLOYEE and DEPENDENT

# 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 (signifying no limit)
- Must have min≤max, min≥0, max ≥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

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

- The numbers mean that for each entity e in E, e must participate in at least min and at most max relationship instances in R at any point in time.
- In this method, min = 0 implies partial participation, whereas min > 0 implies total participation.

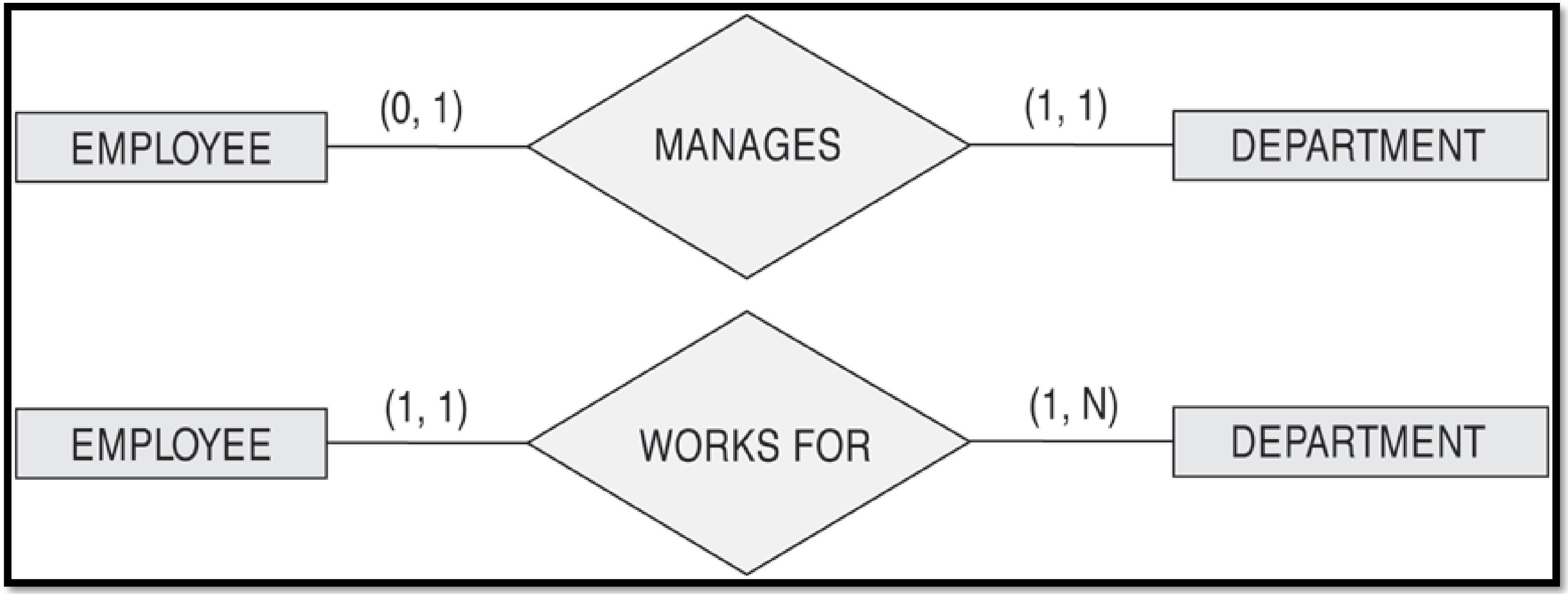


(min, max)

Partial/Total Participation

Max no: of entities that can participate

# 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

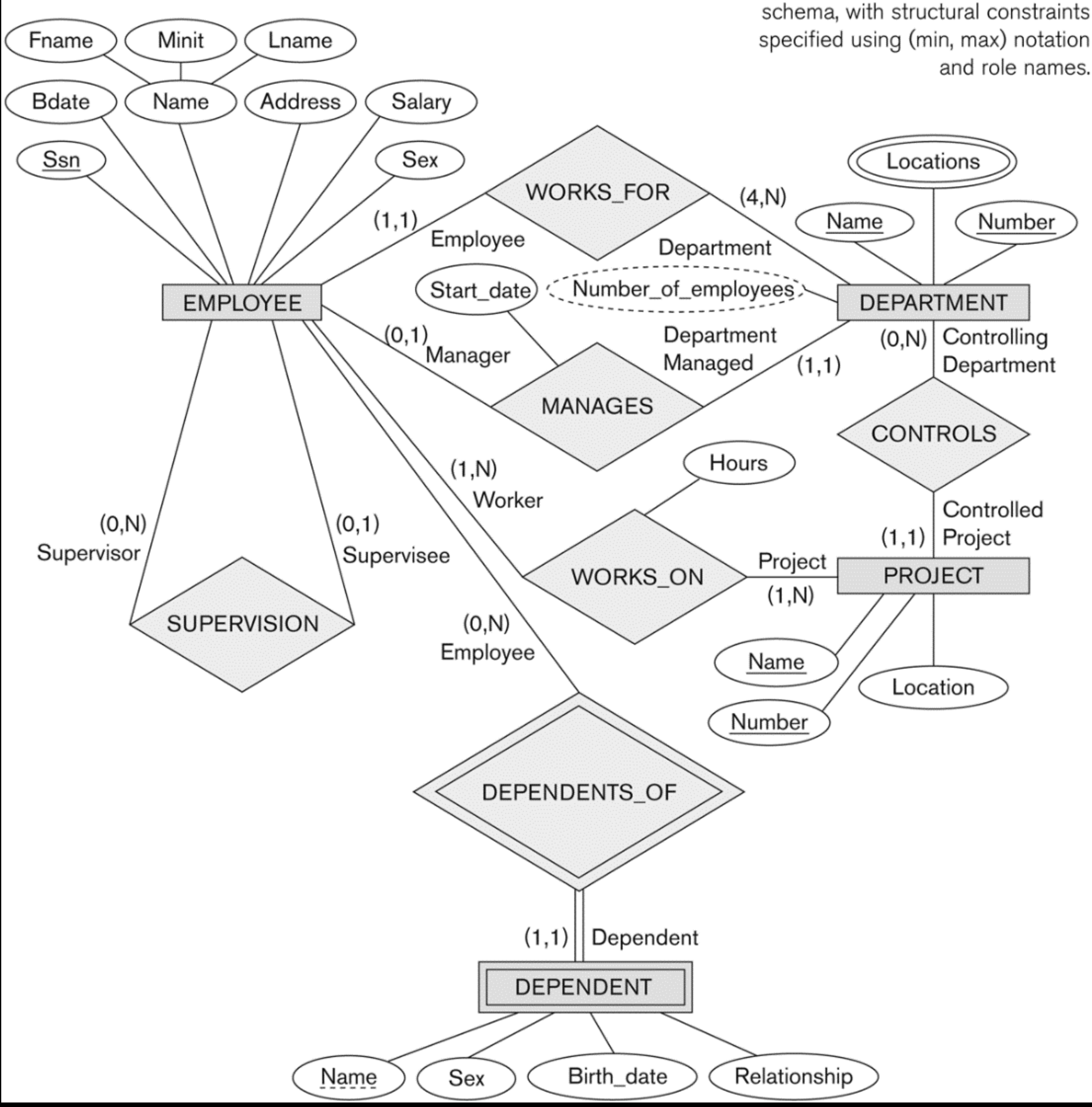


**Figure 3.15**
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

# 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

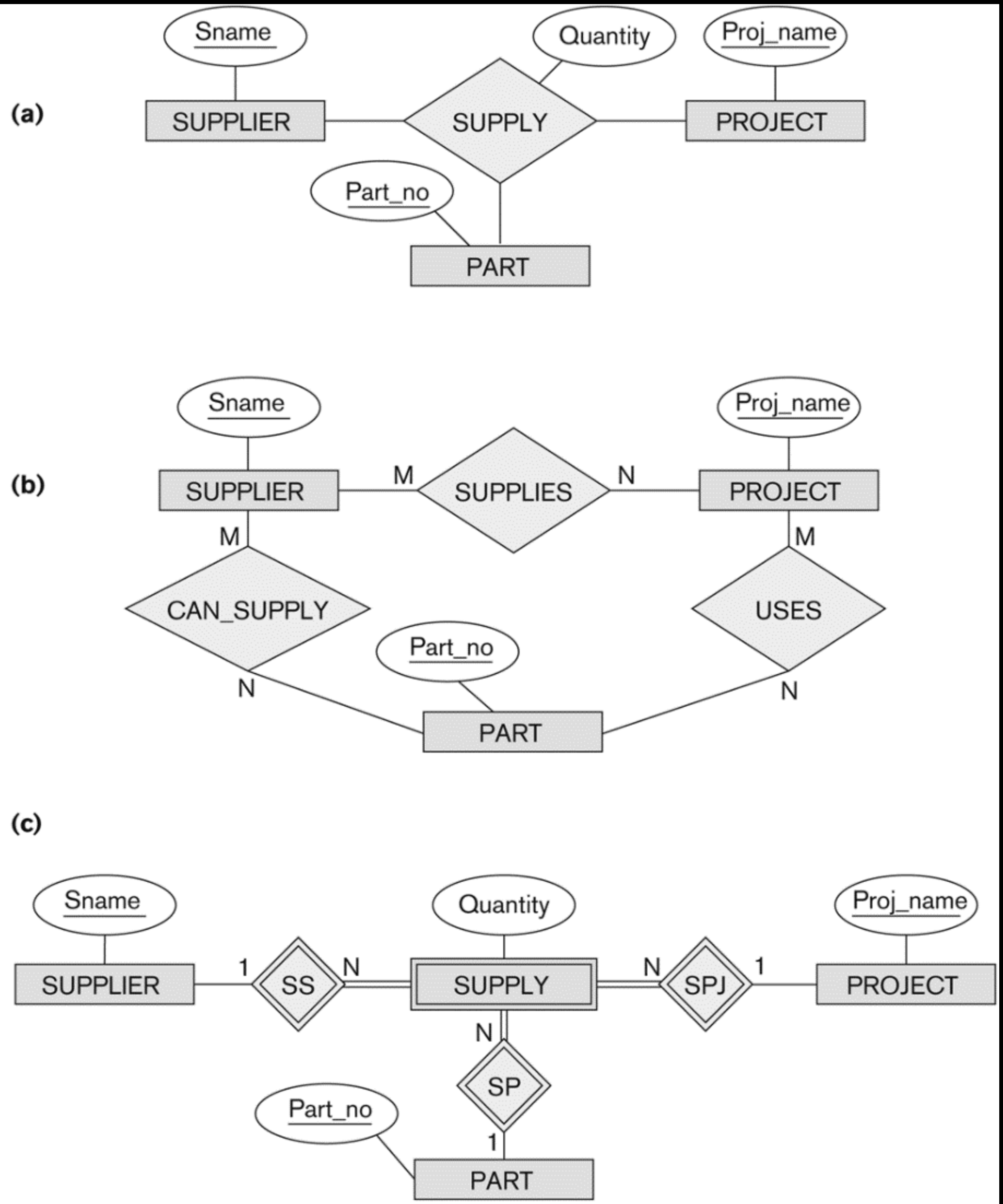


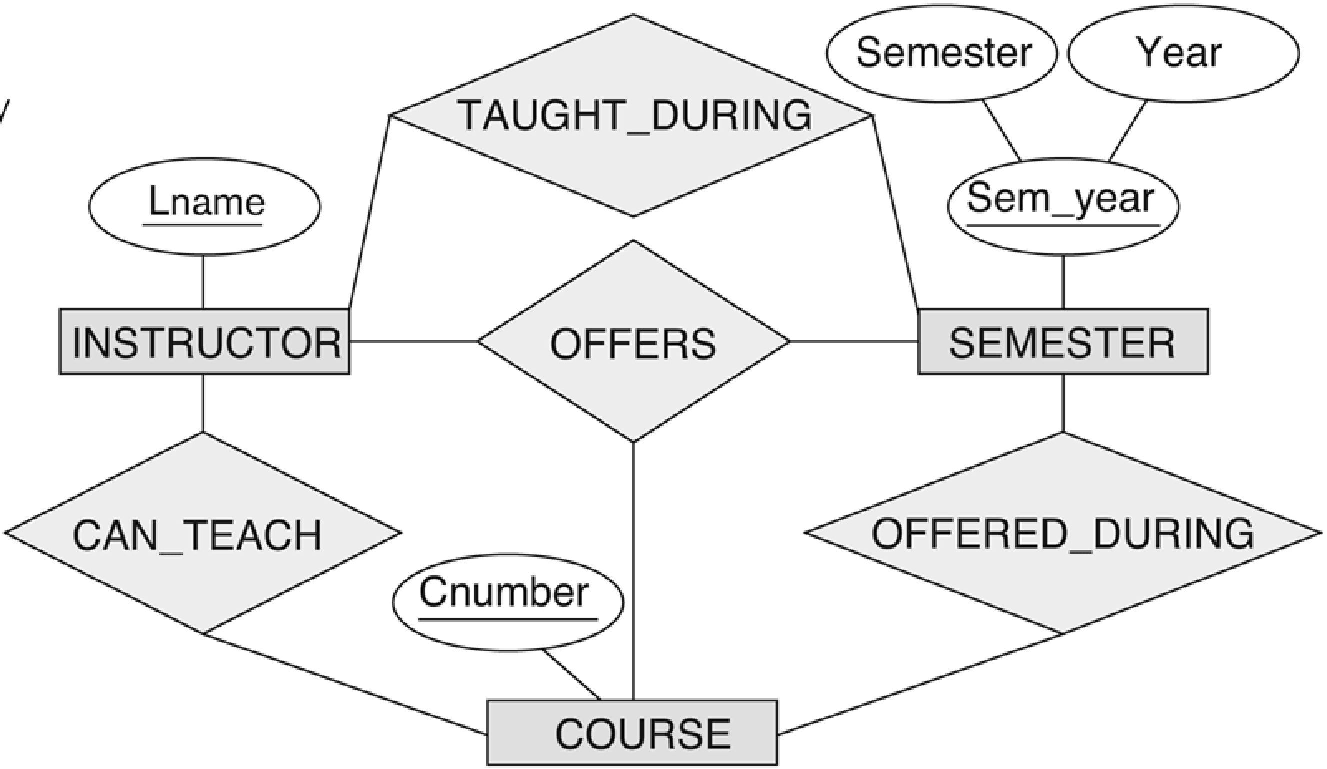
**Figure 3.17**
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

# 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.

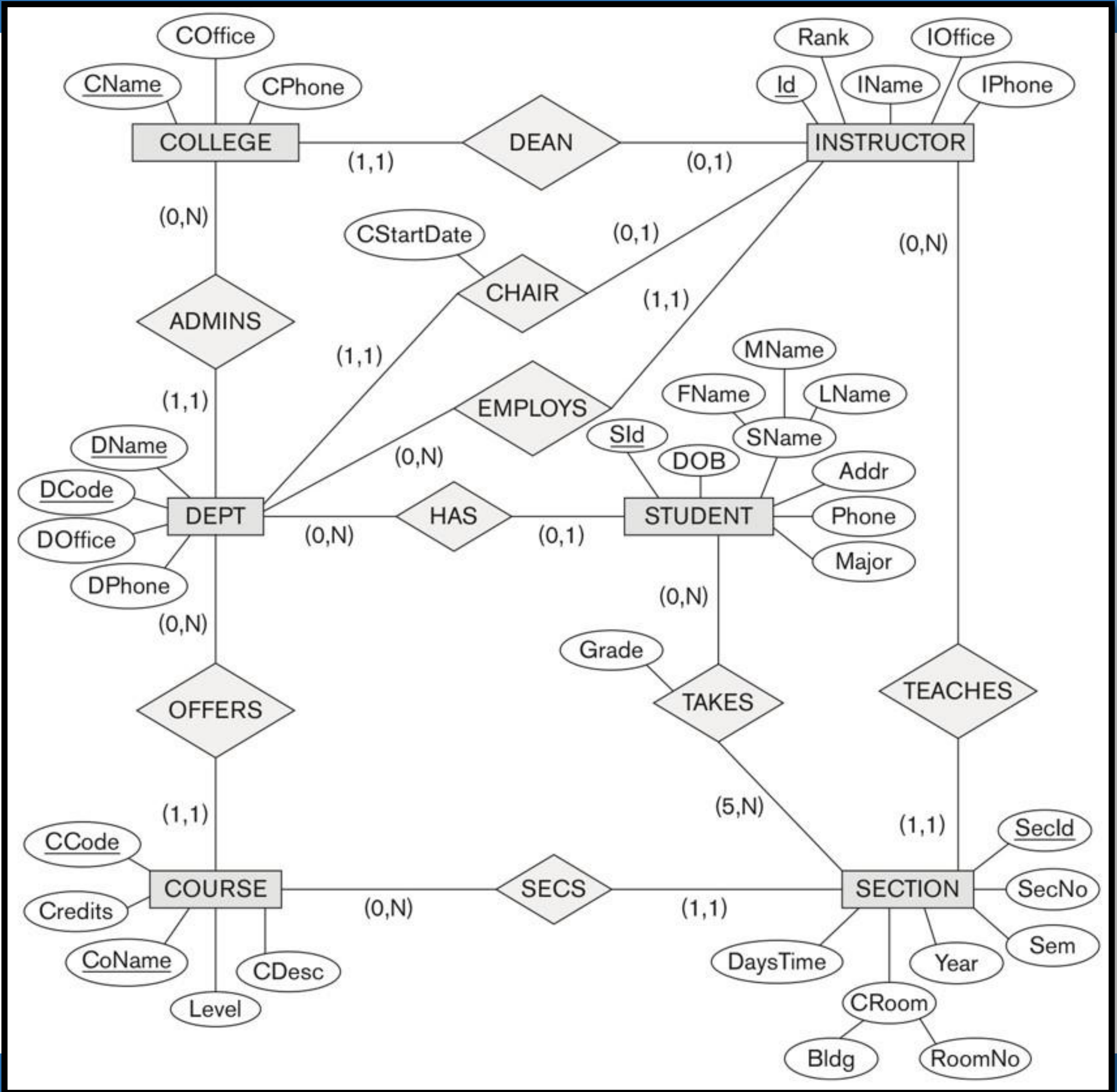


# Displaying constraints on higher-degree relationships

- 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.



UNIVERSITY database conceptual schema

# Some of the Automated Database Design Tools (Note: Not all may be on the market now)

| COMPANY | TOOL | FUNCTIONALITY |
|---|---|---|
| Embarcadero Technologies | ER Studio | Database Modeling in ER and IDEF1X |
| | DB Artisan | Database administration, space and security management |
| Oracle | Developer 2000/Designer 2000 | Database modeling, application development |
| Popkin Software | System Architect 2001 | Data modeling, object modeling, process modeling, structured analysis/design |
| Platinum (Computer Associates) | Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus | Data, process, and business component modeling |
| Persistence Inc. | Pwertier | Mapping from O-O to relational model |
| Rational (IBM) | Rational Rose | UML Modeling & application generation in C++/JAVA |
| Resolution Ltd. | Xcase | Conceptual modeling up to code maintenance |
| Sybase | Enterprise Application Suite | Data modeling, business logic modeling |
| Visio | Visio Enterprise | Data modeling, design/reengineering Visual Basic/C++ |