# 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

# Overview of Database Design Process

- Database application includes Two main activities:
  - Database design
  - Applications design
- Focus in this chapter on 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

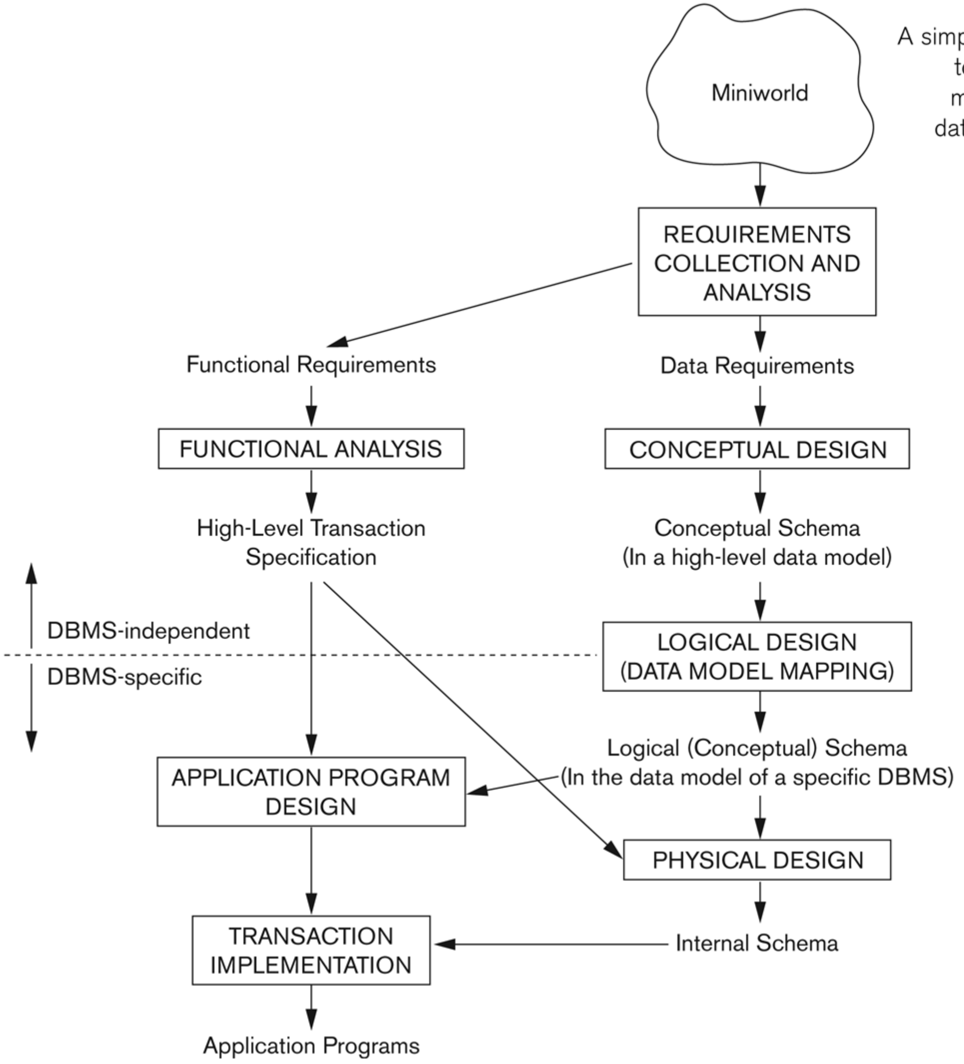

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (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.

# 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 (Contd.)

- We store each EMPLOYEE's social security number, address, salary, sex, and birth date.
  - Each employee *works for* one department but may *work on* several projects.
  - We keep track of the number of hours per week that an employee currently works on each project.
  - We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee for insurance purposes.

# ER DIAGRAM for the COMPANY datbase

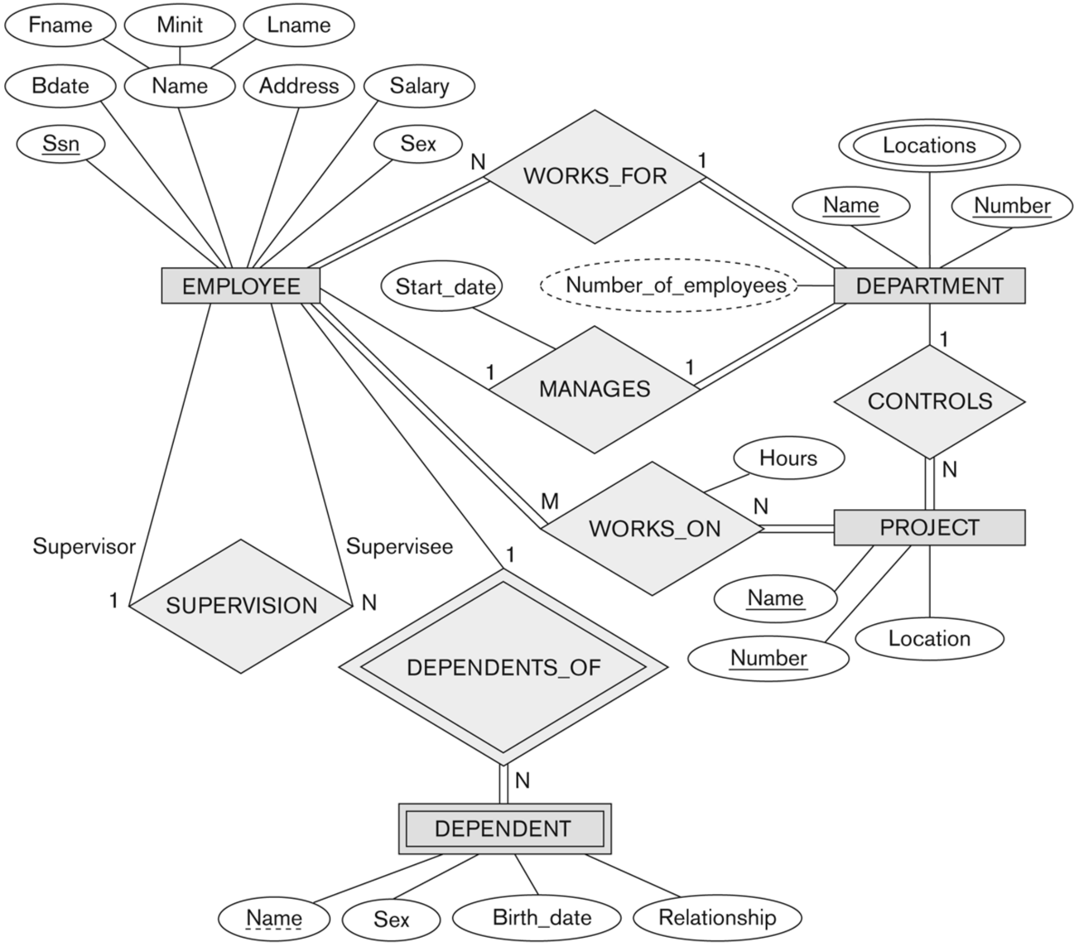


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# ER Model Concepts

- **Entities and Attributes**
  - **Entities** are **specific objects or things** in the mini-world that are represented in the database. An entity may be an object with a physical existence (for example, a particular person, car, or with a conceptual existence (for instance, a company, a job, or a university course)
    - 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, enumerated type, …

# Initial Design of Entity Types:

## EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

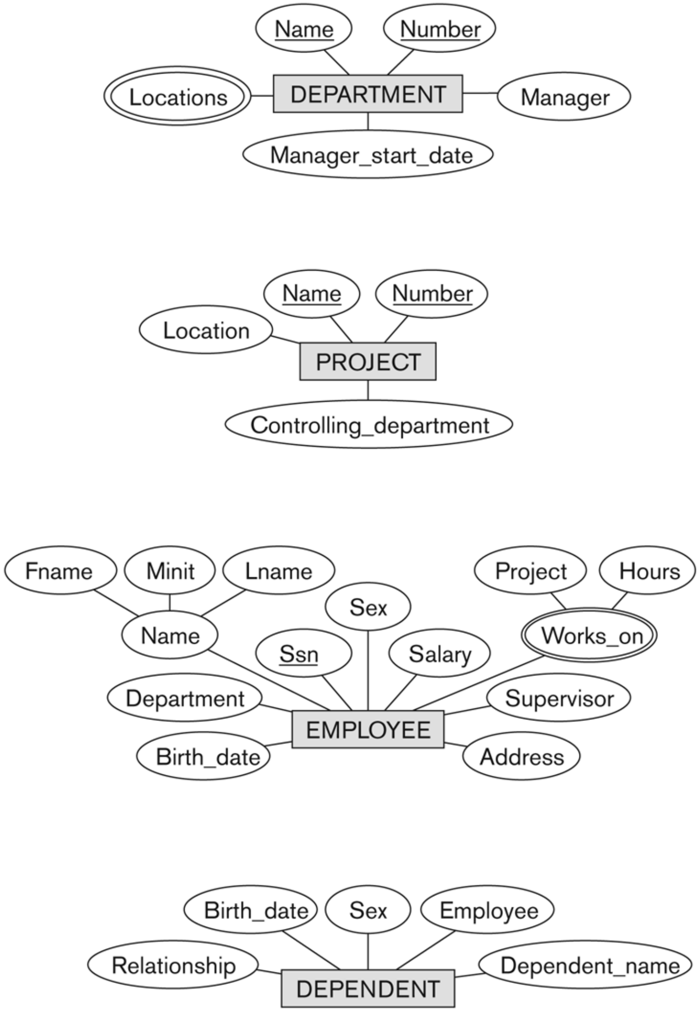


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# Types of Attributes

- *Simple versus composite*
- *Single-valued* versus *multi-valued*
- *Stored versus derived*

# Types of Attributes

- 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.

# Example of a composite attribute

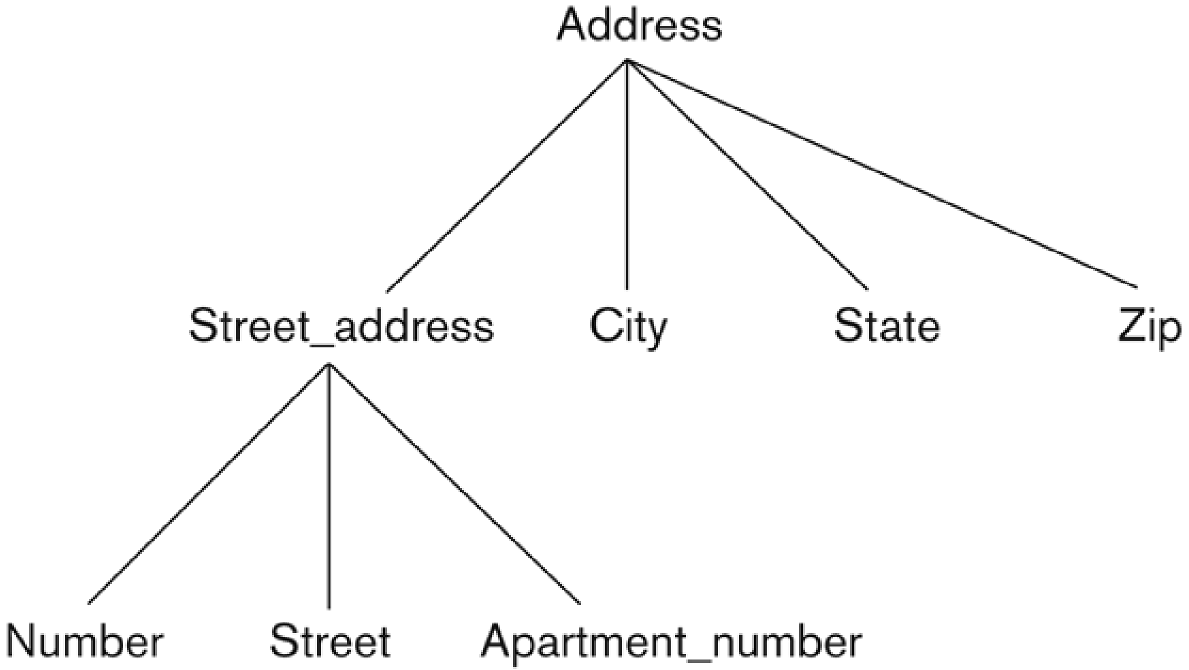


**Figure 3.4**
A hierarchy of composite attributes.

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

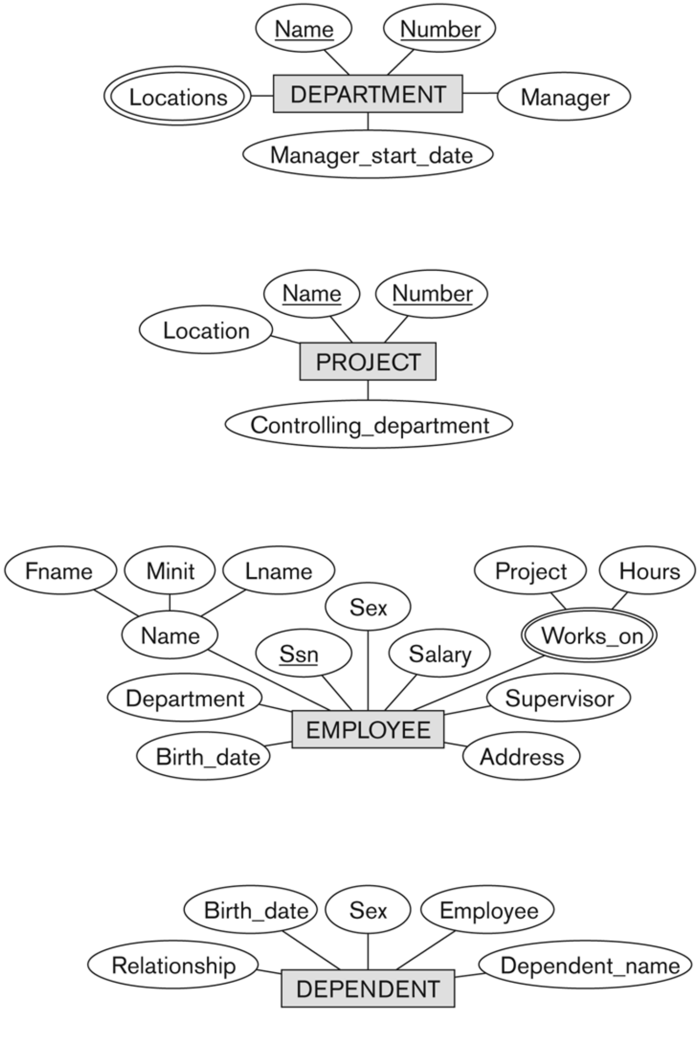


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# Types of Attributes

- Age is a single-valued attribute of a person.
- Multi-valued {}
  - An entity may have **multiple values** for that attribute. For example, Color of a CAR or College_degree of a PERSON.
    - Denoted as {Color} or {College_degree}.
  - It may have upper bounds to constrain the *number of values* allowed for each individual entity (Ex: a car can have three colors at most)
- **Complex Attributes:** In general, composite and multi-valued attributes may be **nested arbitrarily to any number of levels**, although this is rare.
  - For example, College_degree of a STUDENT is a composite multi-valued attribute denoted by {College_degree(College, Year, Degree, Field)}
  - Multiple College_degree values can exist
  - Each has four subcomponent attributes:
    - College, Year, Degree, Field

# Initial Design of Entity Types:

## EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

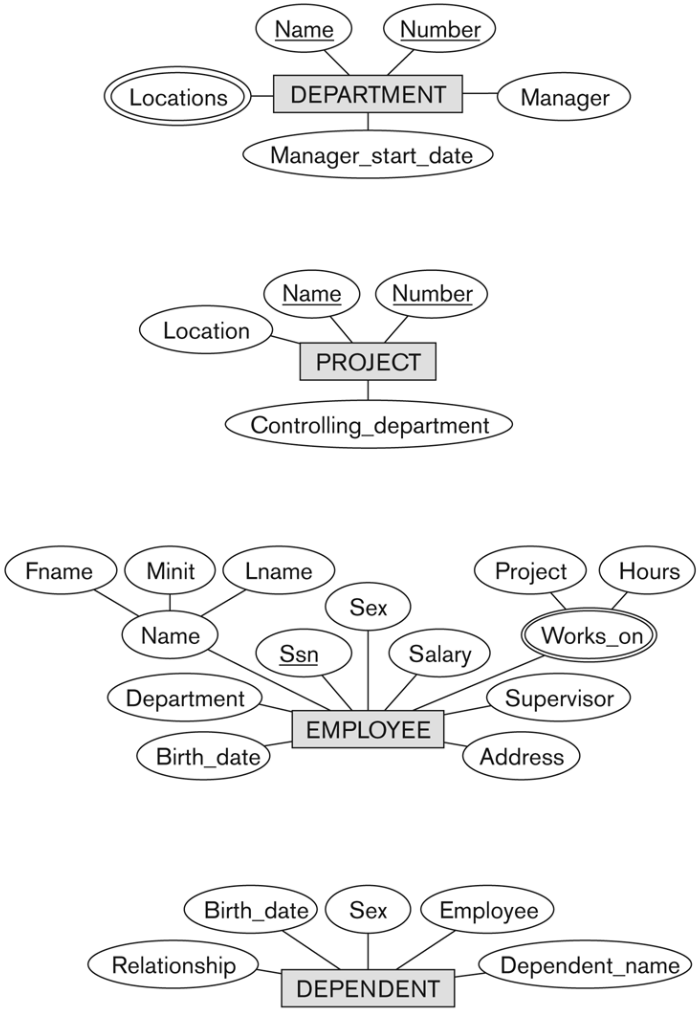


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# Types of Attributes

- **Derived Attributes:**
  - For example, the Age and Birth_date attributes of a person.
  - The Age attribute is called a **derived attribute and is said to be derivable from the Birth_date attribute,** which is called a **stored attribute.**
  - **Some attribute values can be derived from** *related entities; for example, an attribute Number_of_employees of a DEPARTMENT* entity can be derived by counting the number of employees related to (working for) that department.

# ER DIAGRAM for the COMPANY datbase

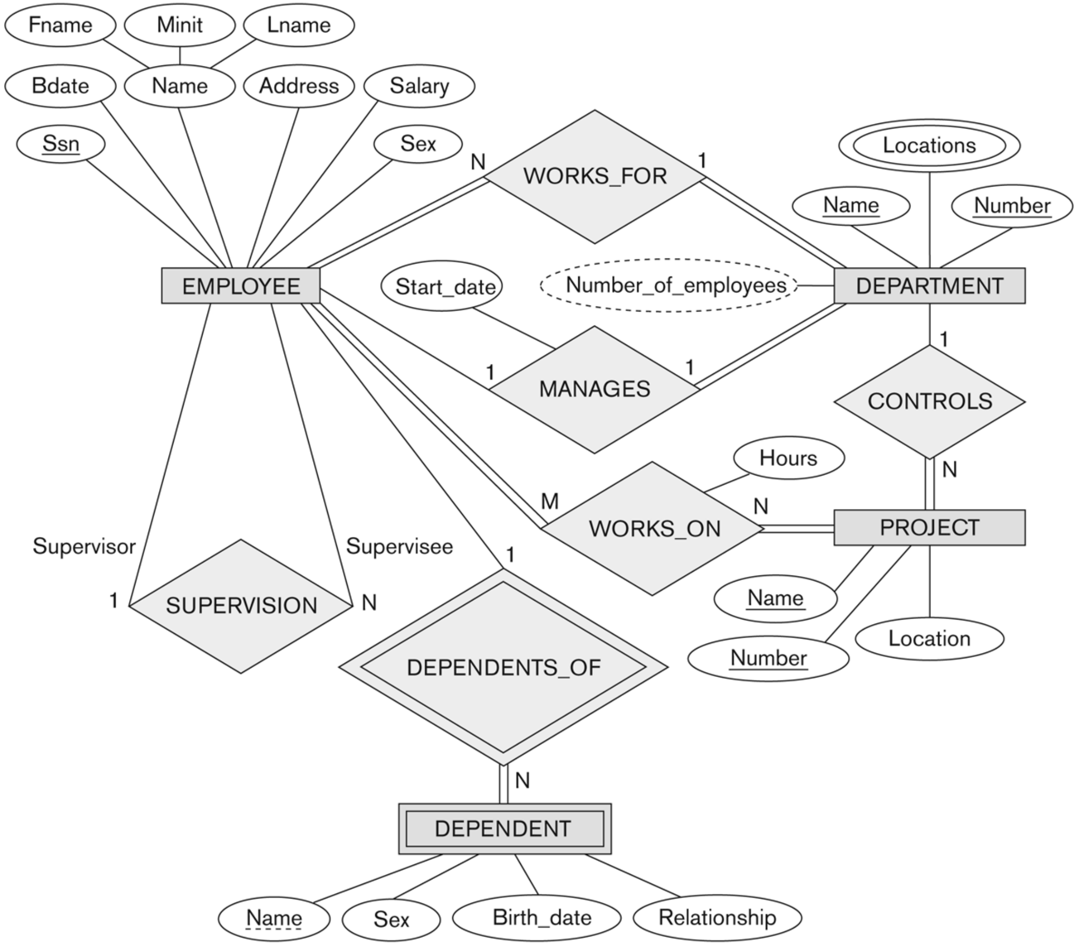


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Null – 3 cases

- **Case 1 (Not applicable): a particular entity may not have an applicable value** for an attribute.
  - For example, the Apartment_number and a College_degrees attribute
- **Case 2 (Unknown):** NULL can also be used if we **do not know** the value of an attribute for a particular entity—for example, if we do not know the home phone number of 'John Smith' i*.*
  - Exists but is *missing—for instance, if the Height attribute of a* person is listed as NULL.
  - *Not known whether the attribute value exists*—for example, if the Home_phone attribute of a person is NULL.

# Entity Types and Key Attributes

- 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

- 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 (candidate key)**.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN)
    - VehicleTagNumber (Number, State), plate number.
- Each key is <u>underlined</u>
- <u>An entity type may also have *no key, in which case it is called a weak entity type*</u>

# Initial Design of Entity Types:

## EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

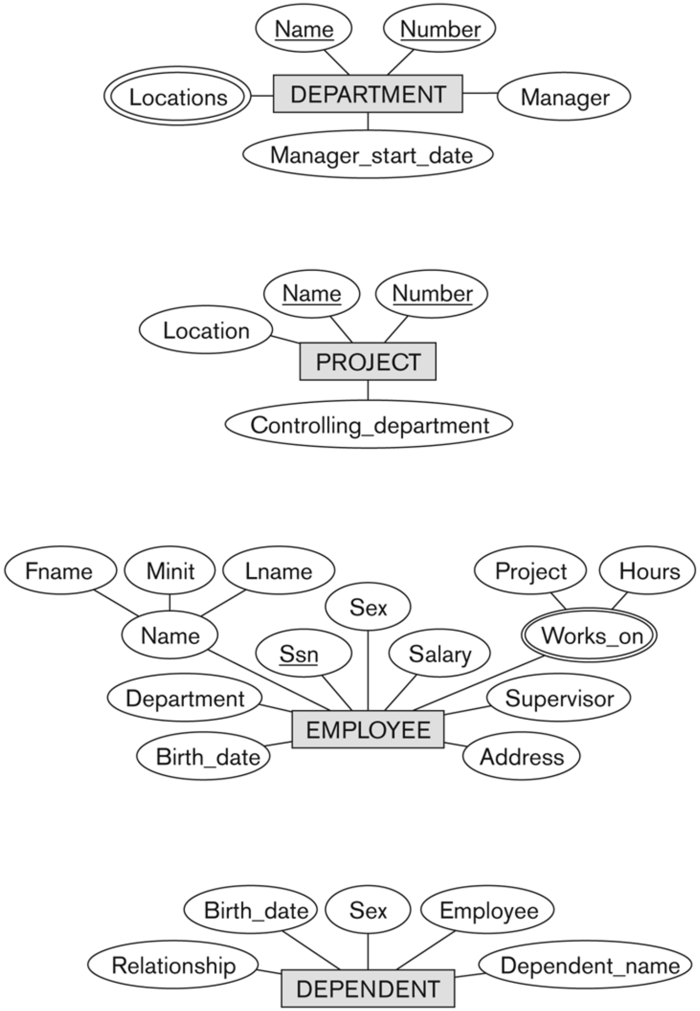


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# 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 CAR example on next slide

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

(a)

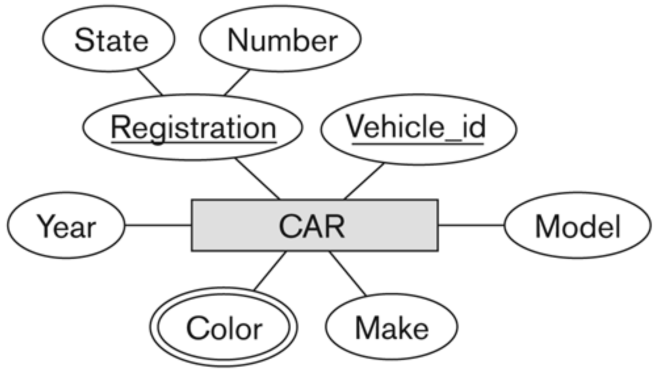


**Figure 3.7**
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

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

CAR$_1$
((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})

⋮

# Entity Set

- Each entity type will have a collection of entities stored in the database
  - Called the **entity set**
- 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
- Entity set is the current *state* of the entities of that type that are stored in the database

# Initial 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 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, DEPENDENT

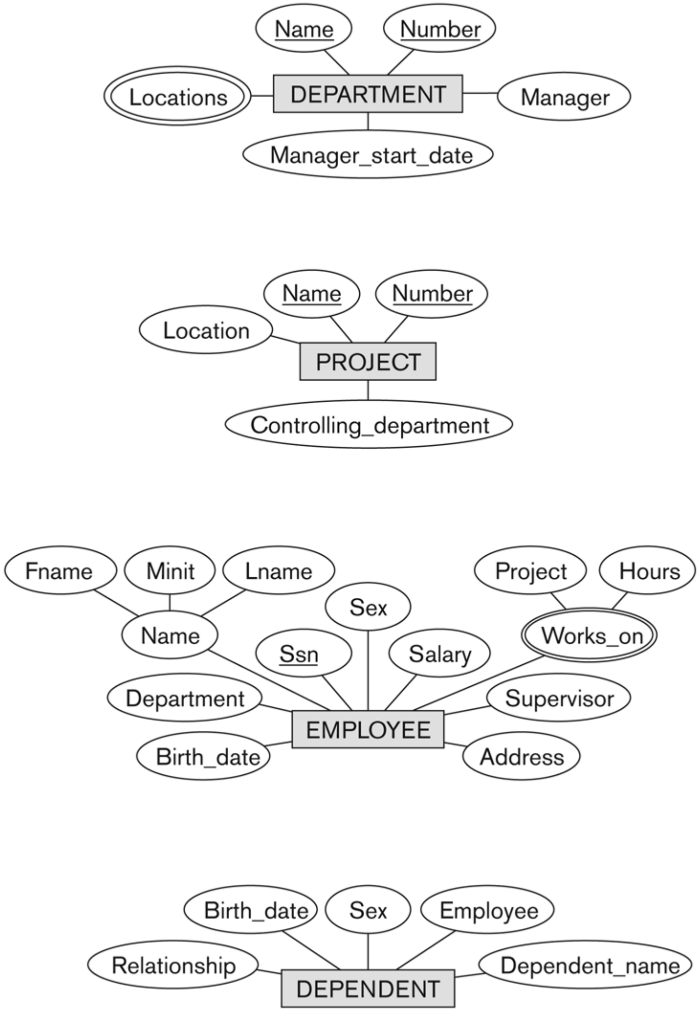


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# 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

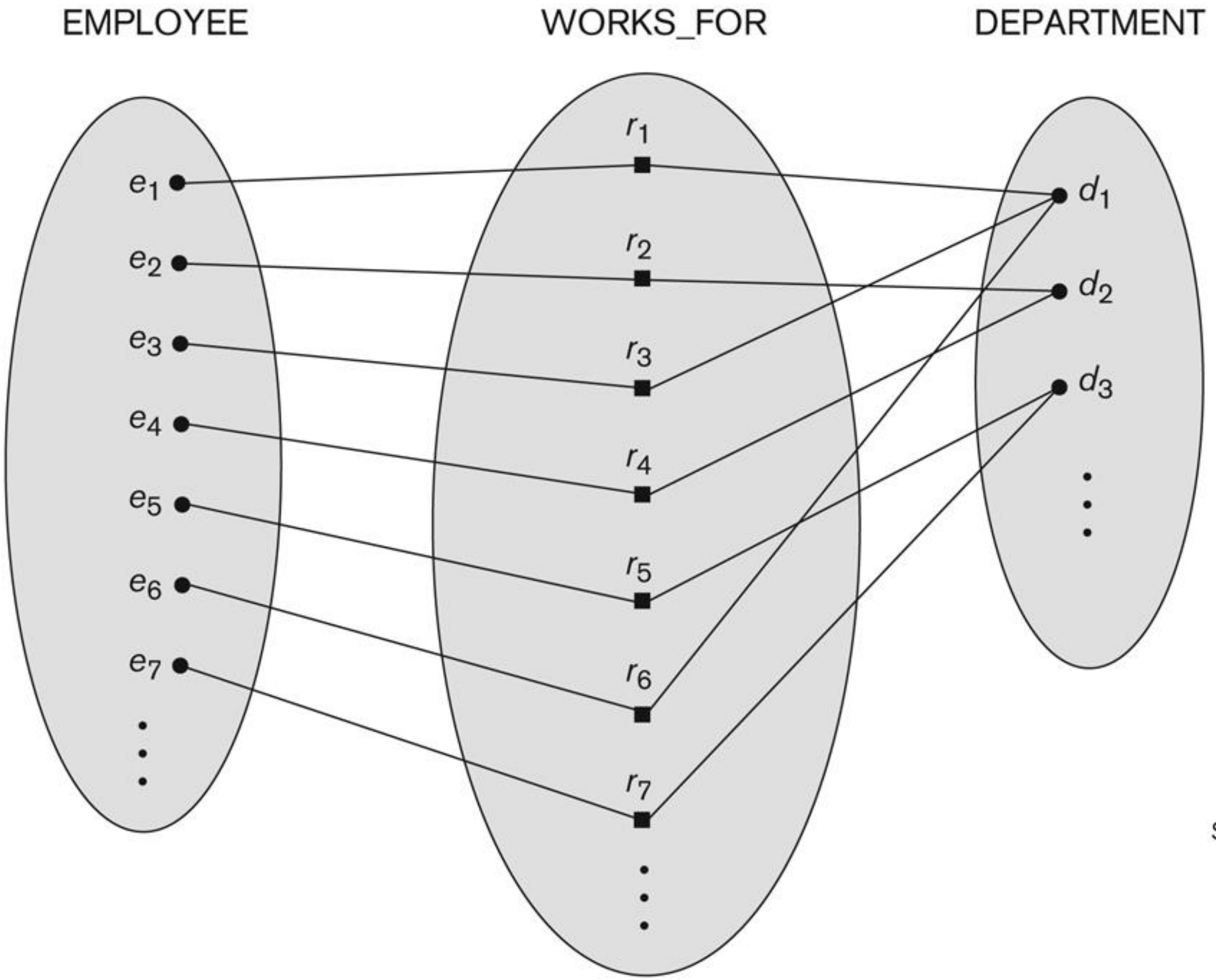


**Figure 3.9**
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

# Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

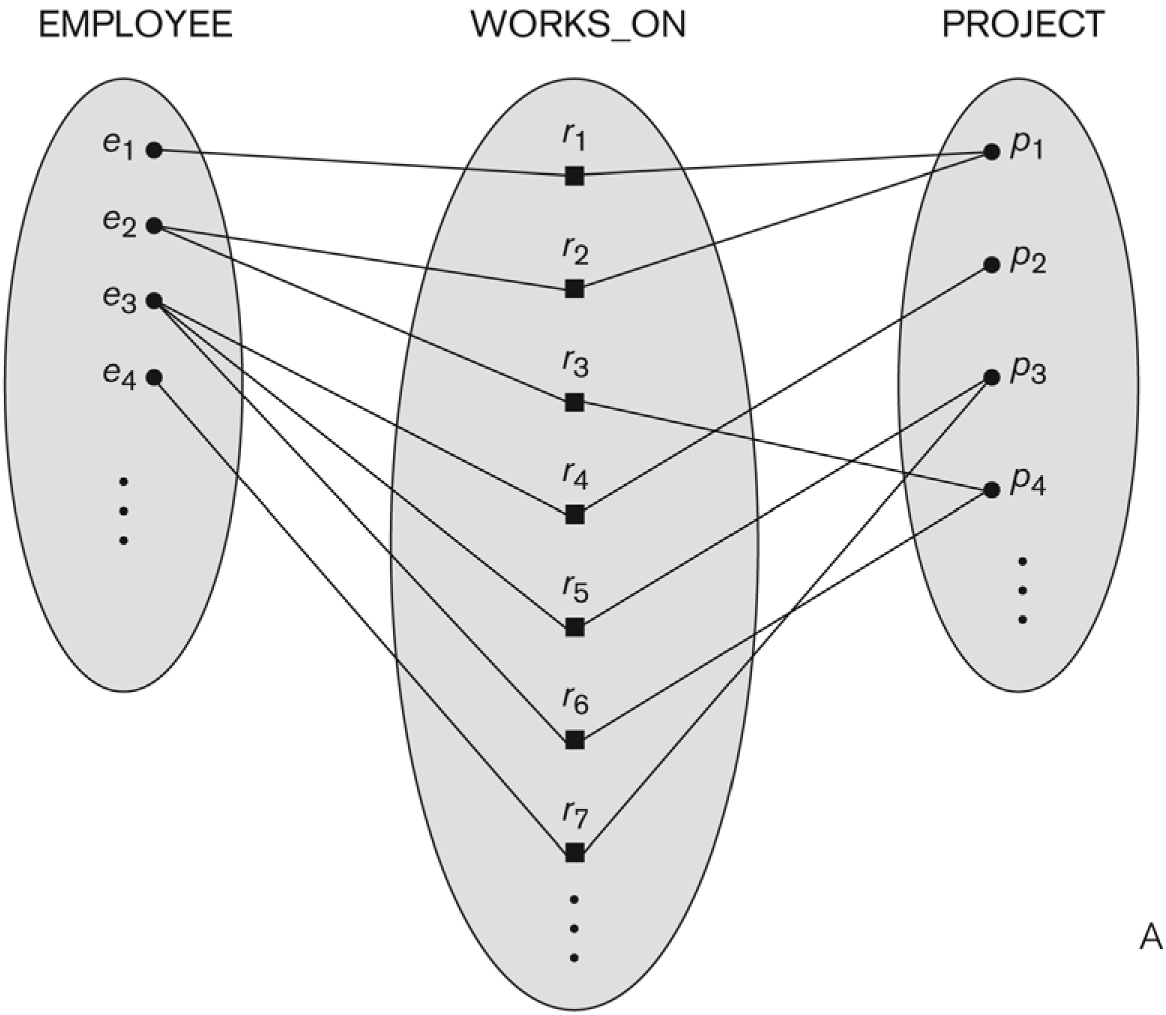


**Figure 3.13**
An M:N relationship, WORKS_ON.

# 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

# 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**

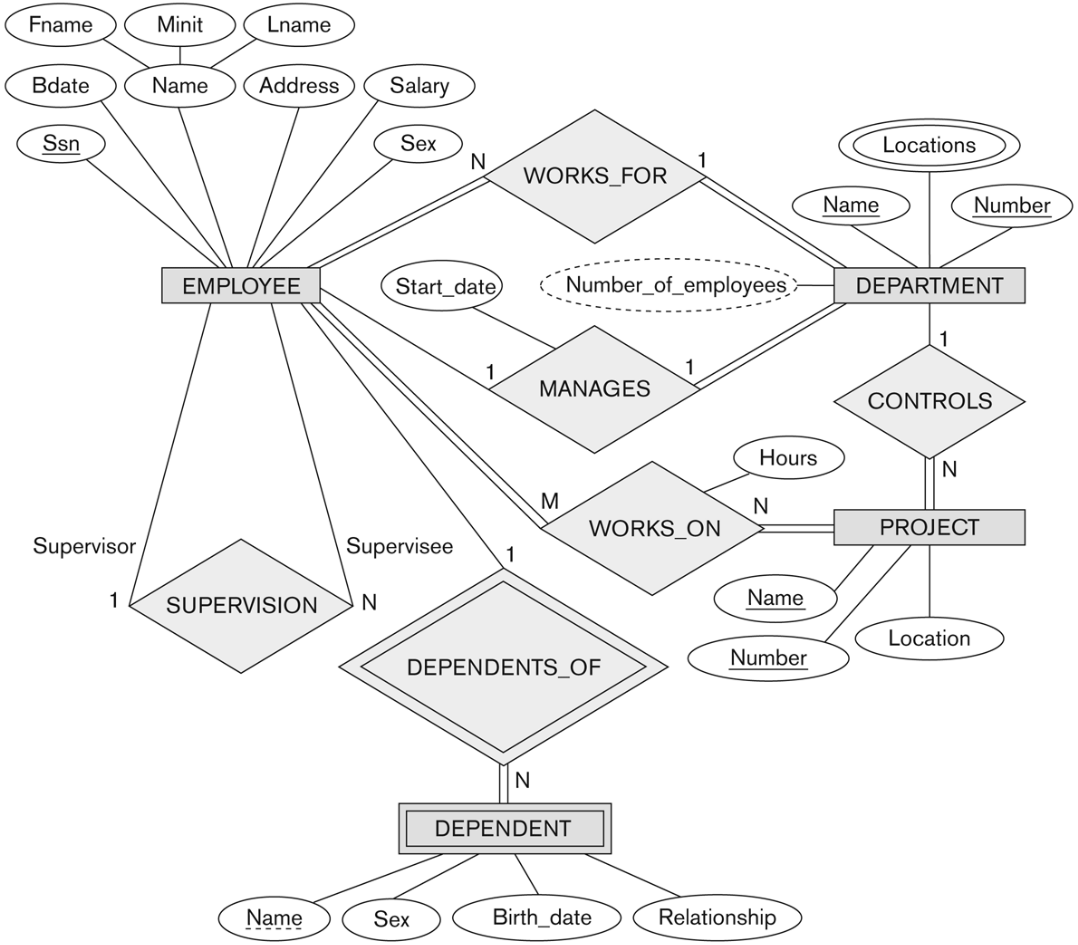


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# 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.

# Initial Design of Entity Types:

## EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

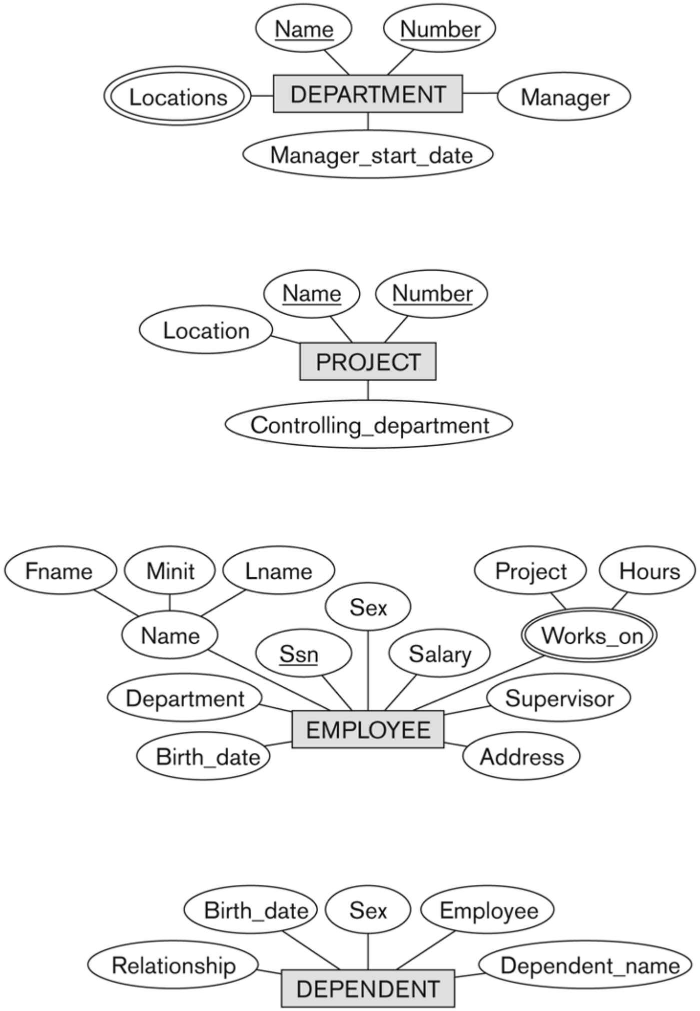


**Figure 3.8**
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

# ER DIAGRAM – Relationship Types are:

**WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF**

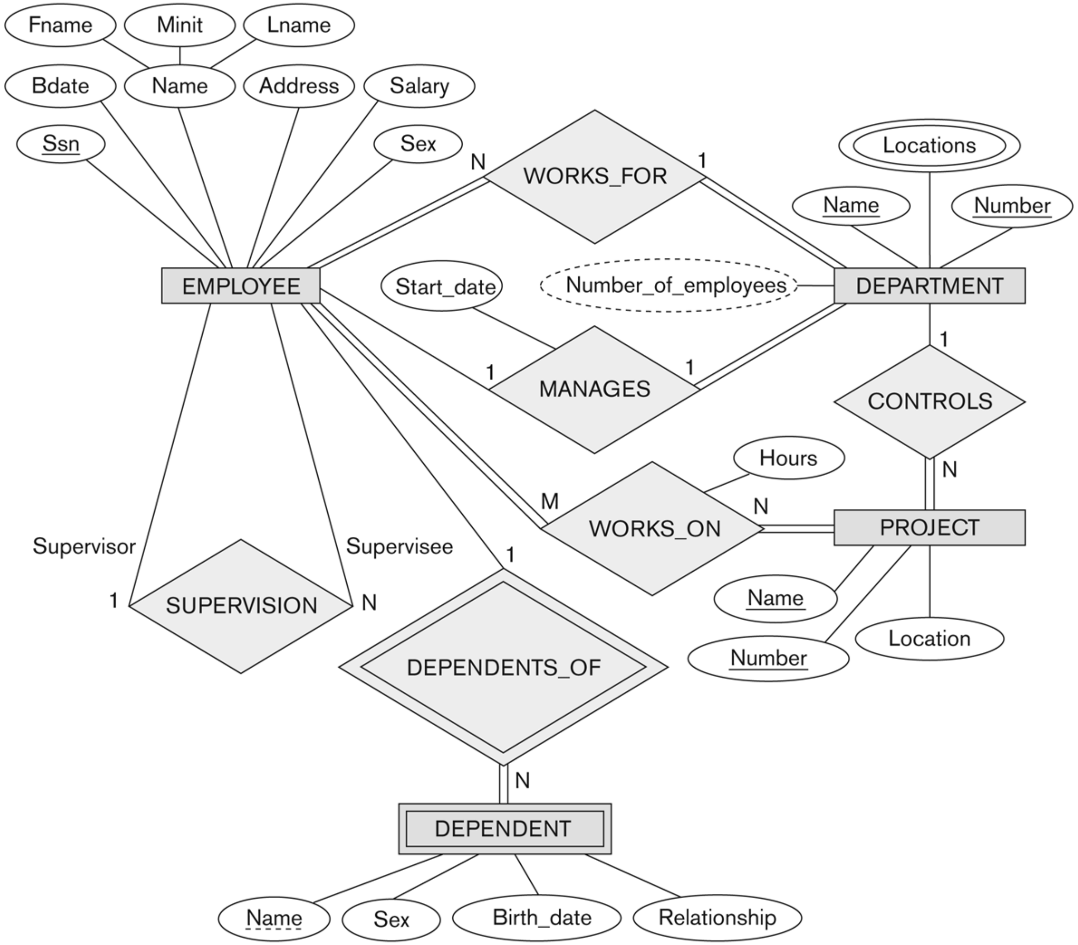


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Role Names and Recursive Relationships

- An relationship type whose with the same participating entity type in **distinct roles**
- 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

# ER DIAGRAM – Relationship Types are:

**WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF**

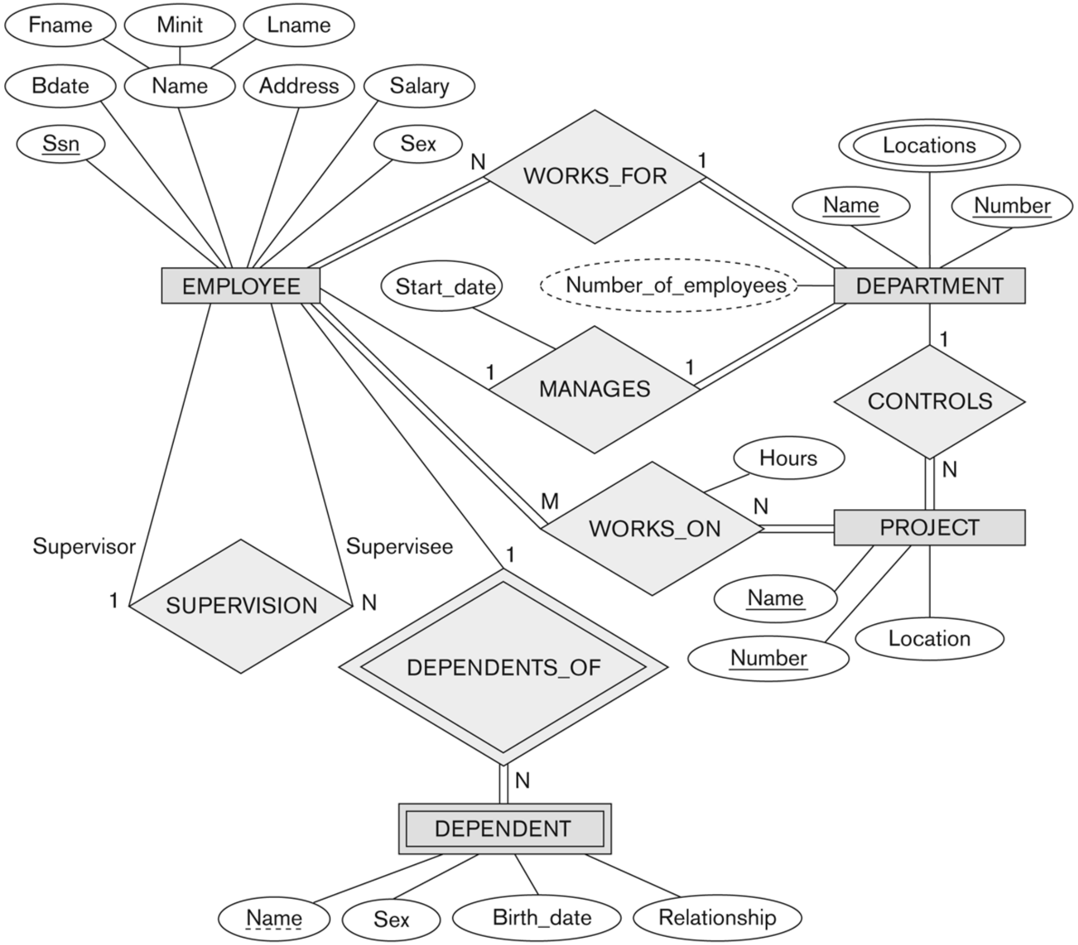


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# 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 *(Dots)*
  - The particular entity they are related to in the identifying entity 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

# ER DIAGRAM – Relationship Types are:

**WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF**

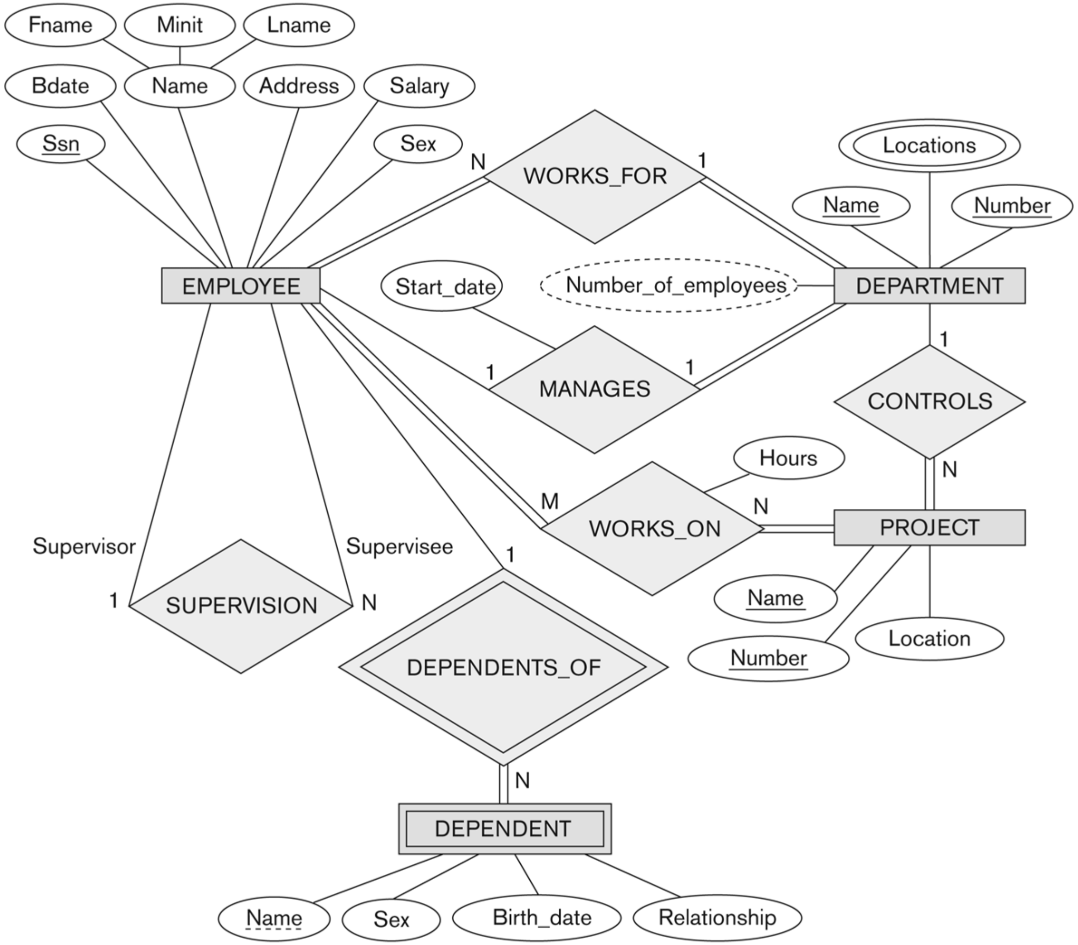


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# Weak Entity Types

- A weak entity type always has a *total participation constraint (existence* dependency) with respect to its identifying relationship.
  - Not every existence dependency results in a weak entity type For example, a DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity.
- Weak entity types can sometimes be represented as complex (composite, multivalued) attributes.
  - We could specify a multivalued attribute Dependents for EMPLOYEE, which is a composite attribute with component attributes Name, Birth_date, Sex, and Relationship.
- The choice of which representation to use is made by the database designer.
- If the weak entity participates independently in relationship types other than its identifying relationship type, then it should not be modeled as a complex attribute.

# 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)

# 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.

# ER DIAGRAM – Relationship Types are:

**WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF**

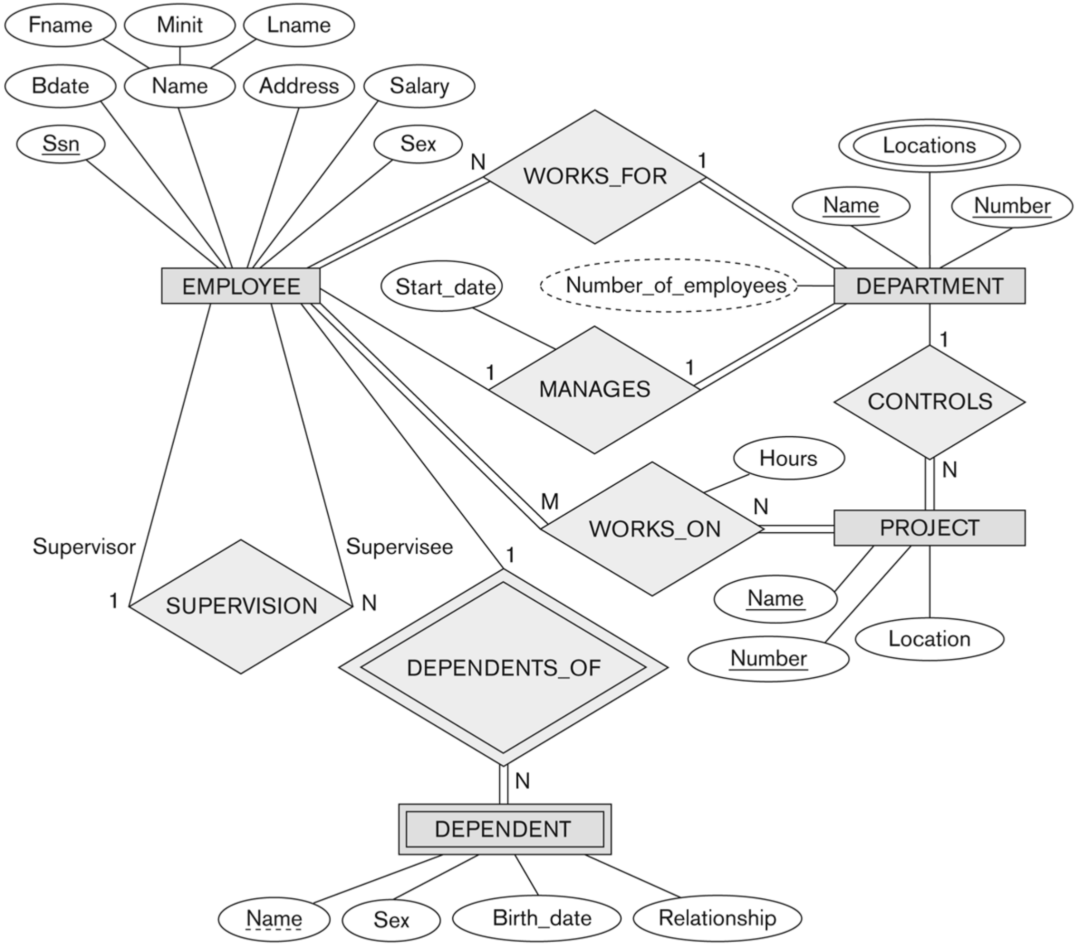


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

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

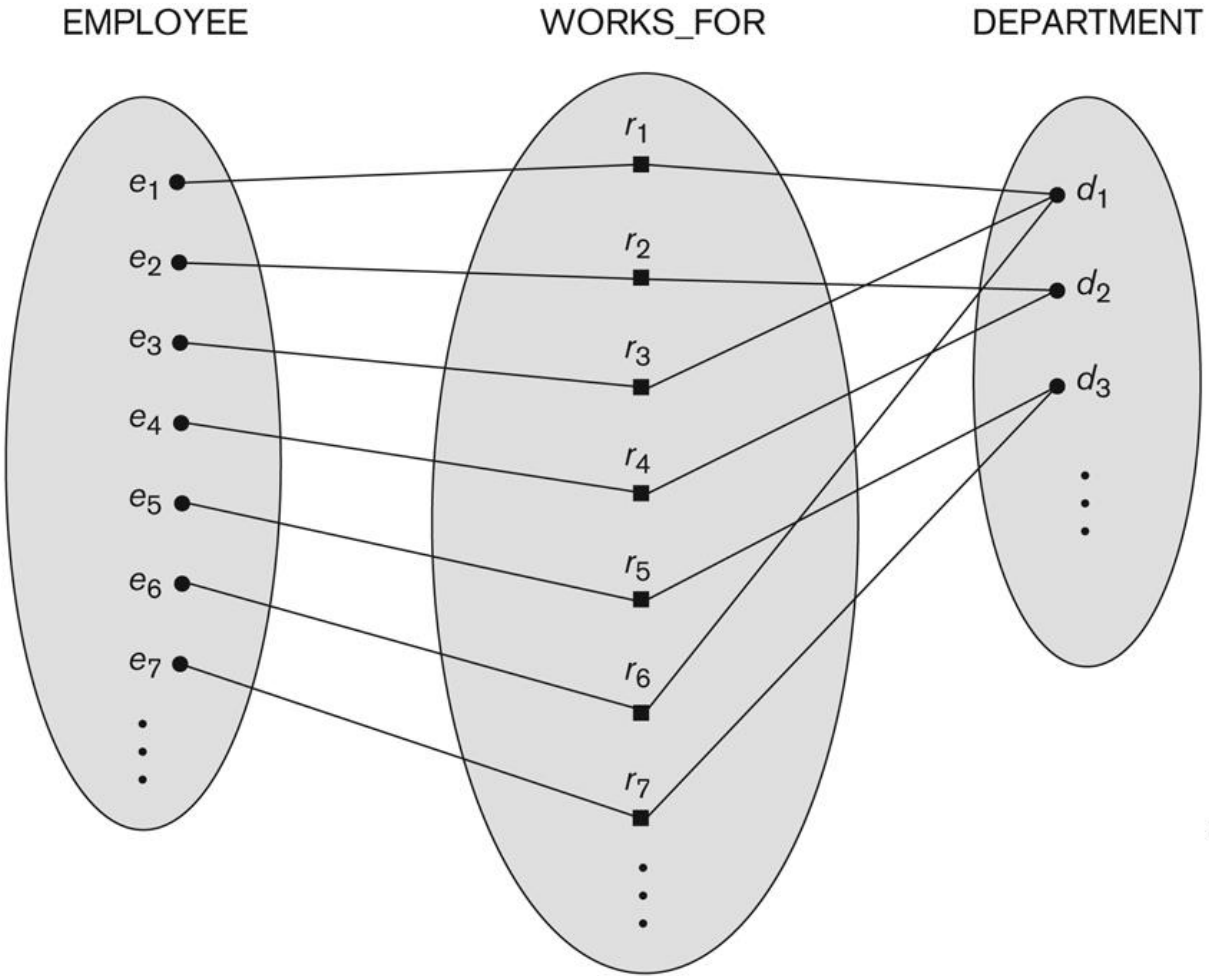


**Figure 3.9**
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

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

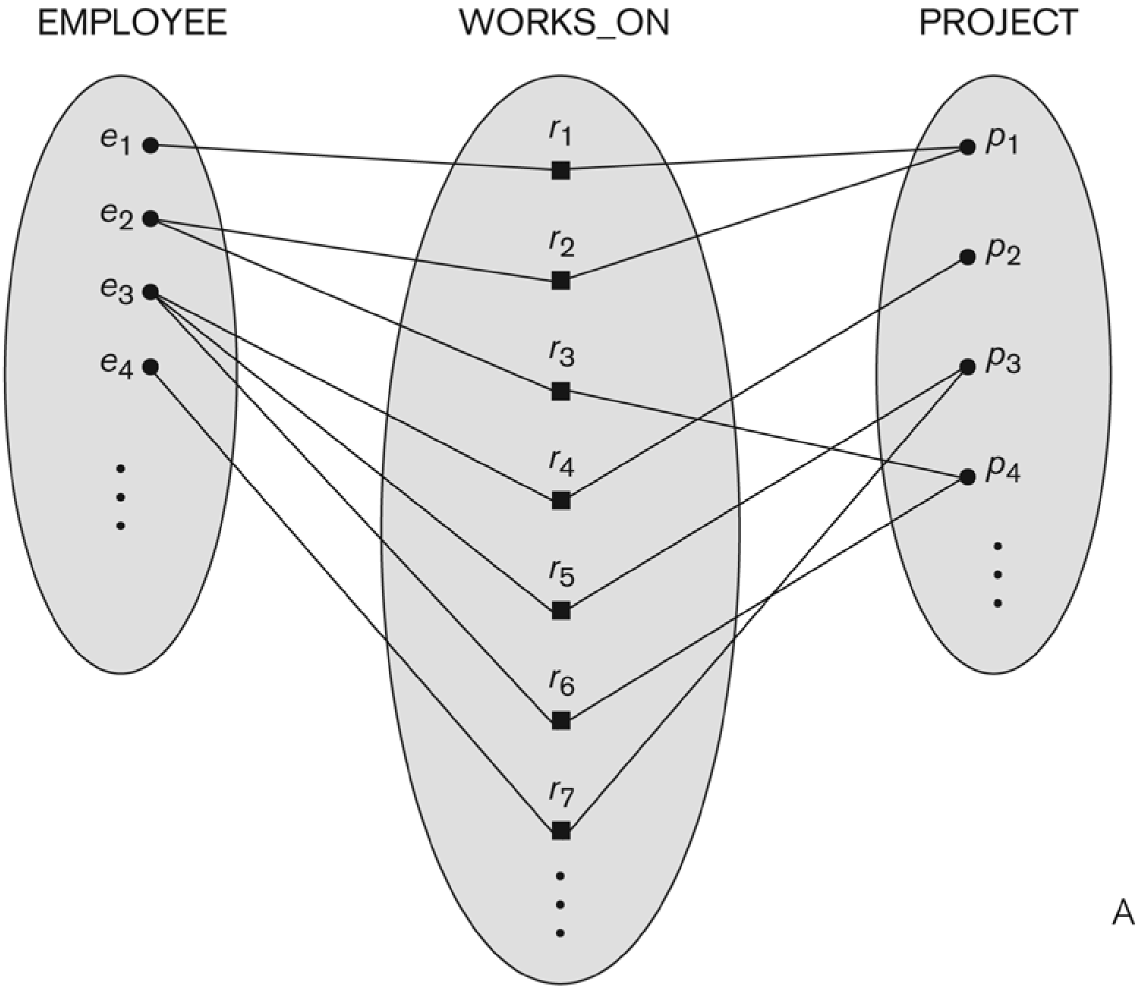


**Figure 3.13**
An M:N relationship,
WORKS_ON.

# 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`

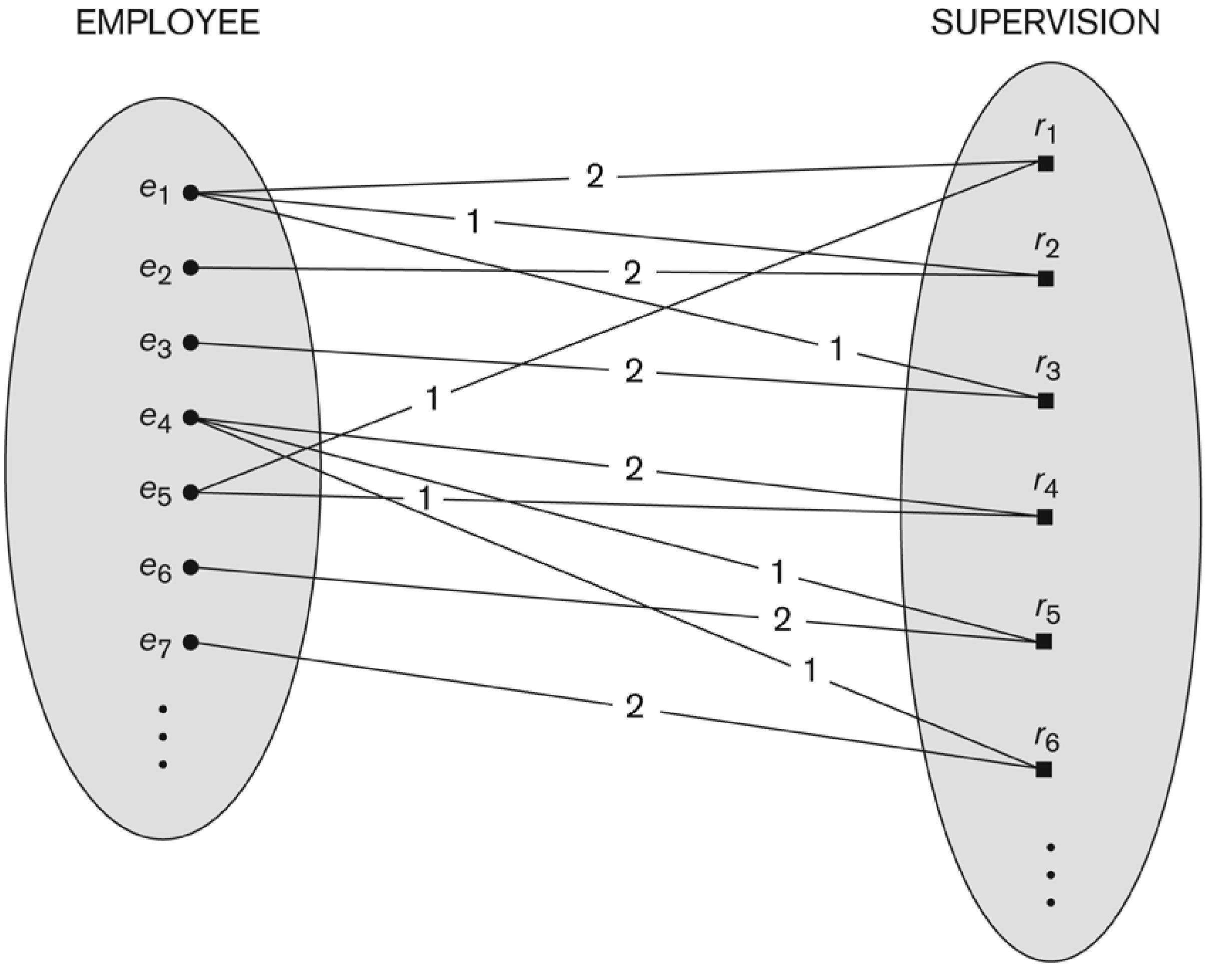


**Figure 3.11**
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

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

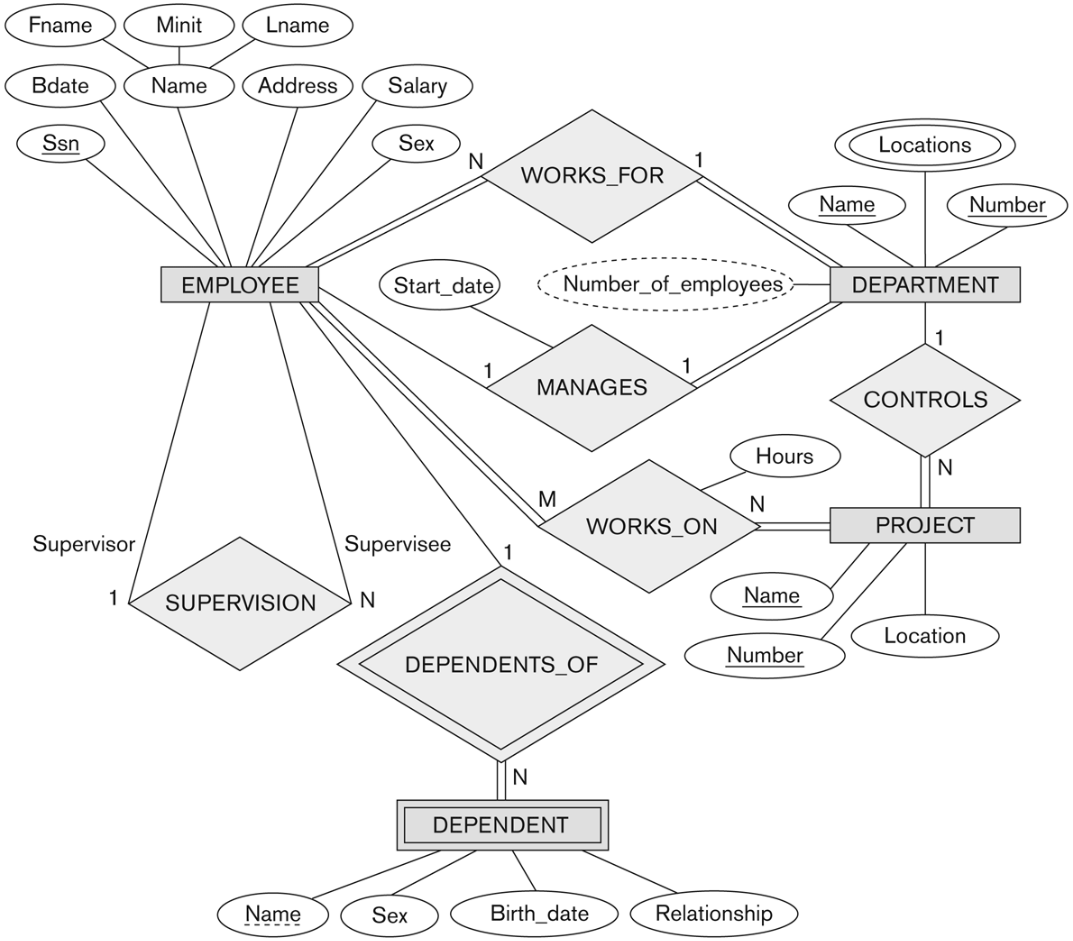


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# 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**
    - For example, if the WORKS_FOR relationship also has an attribute Start_date

# Example Attribute of a Relationship Type: Hours of WORKS_ON

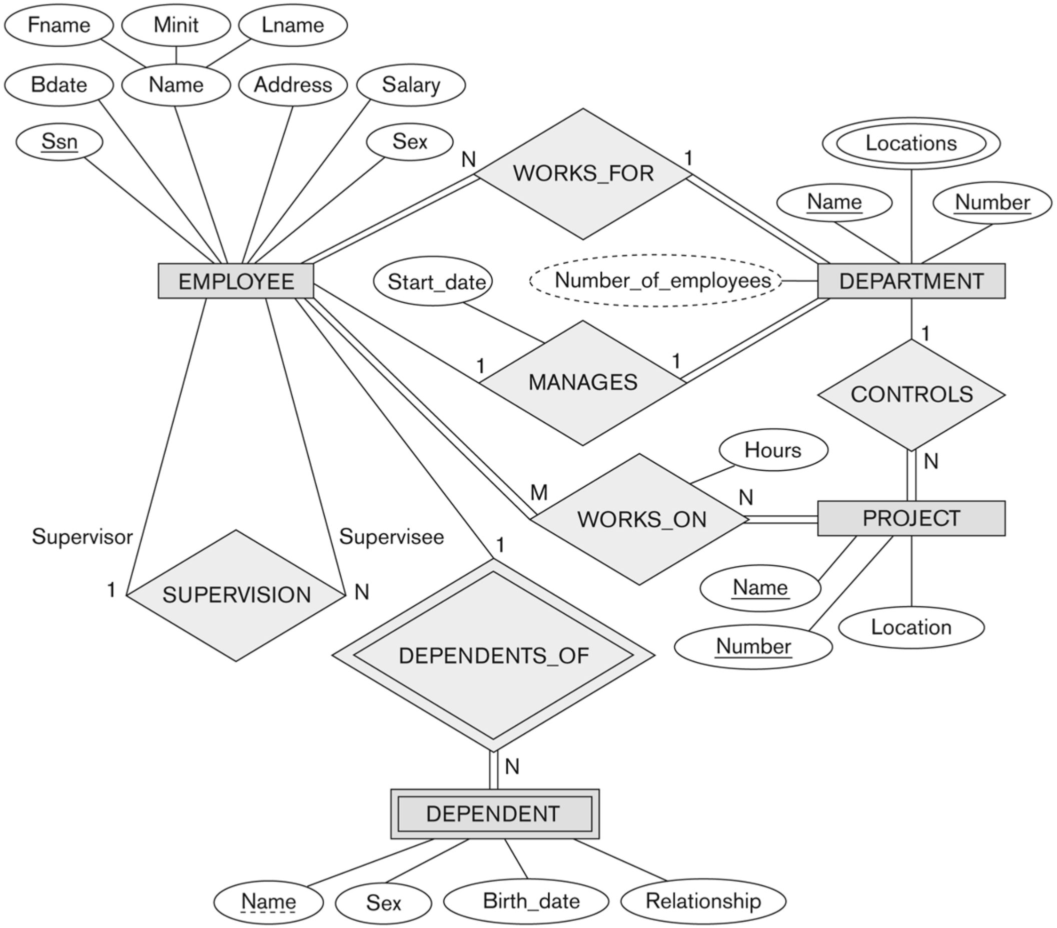


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# ER DIAGRAM – Relationship Types are:

**WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF**

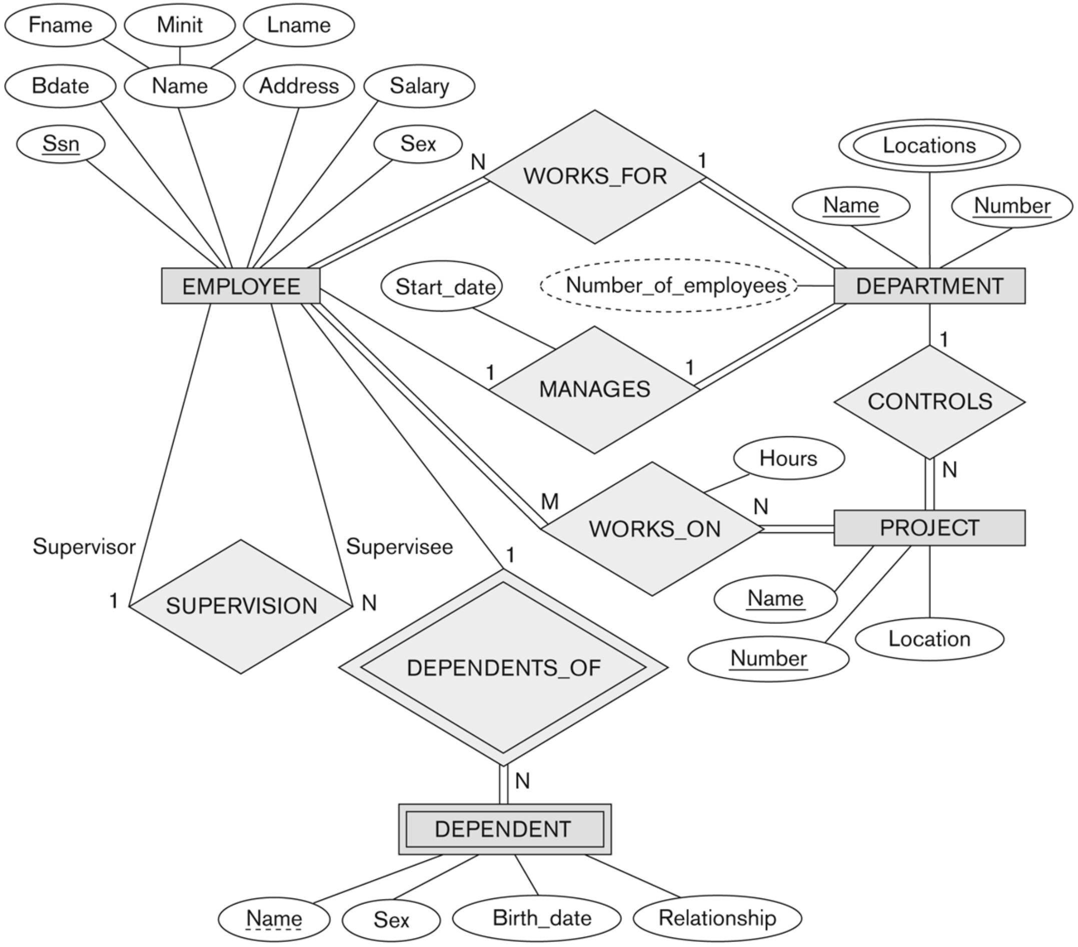
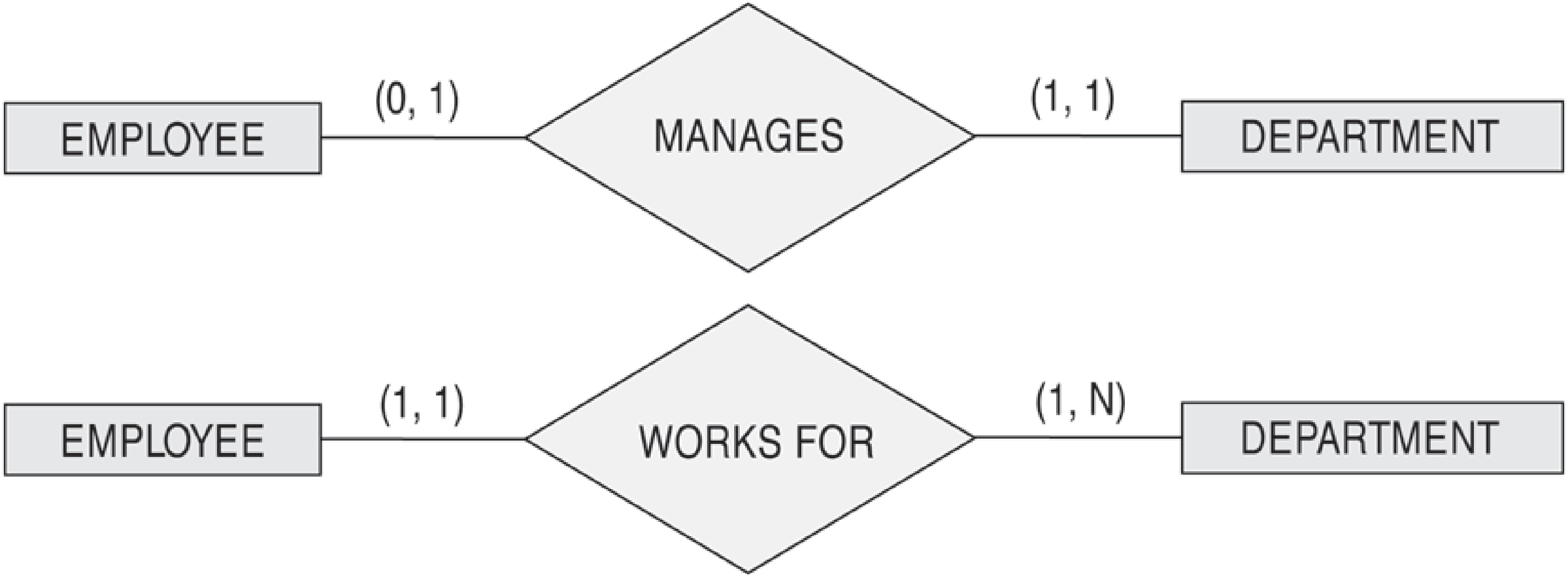


**Figure 3.2**
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

# 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

# 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

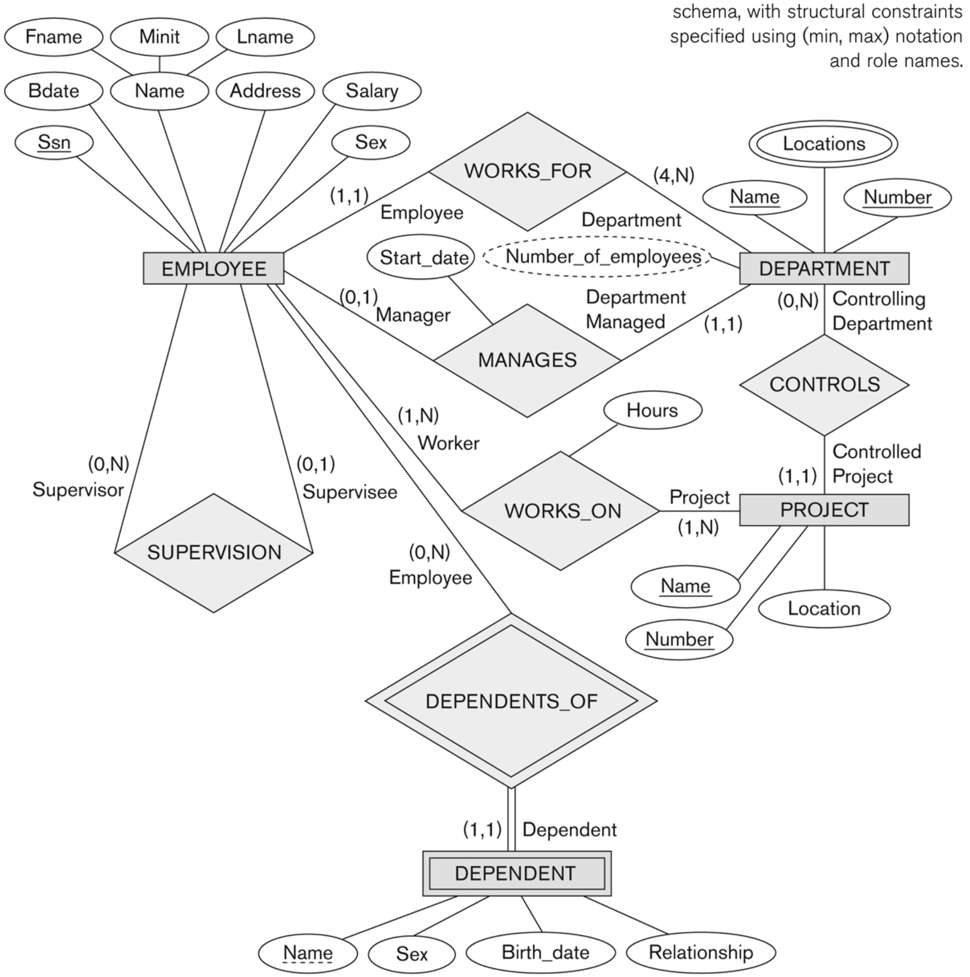
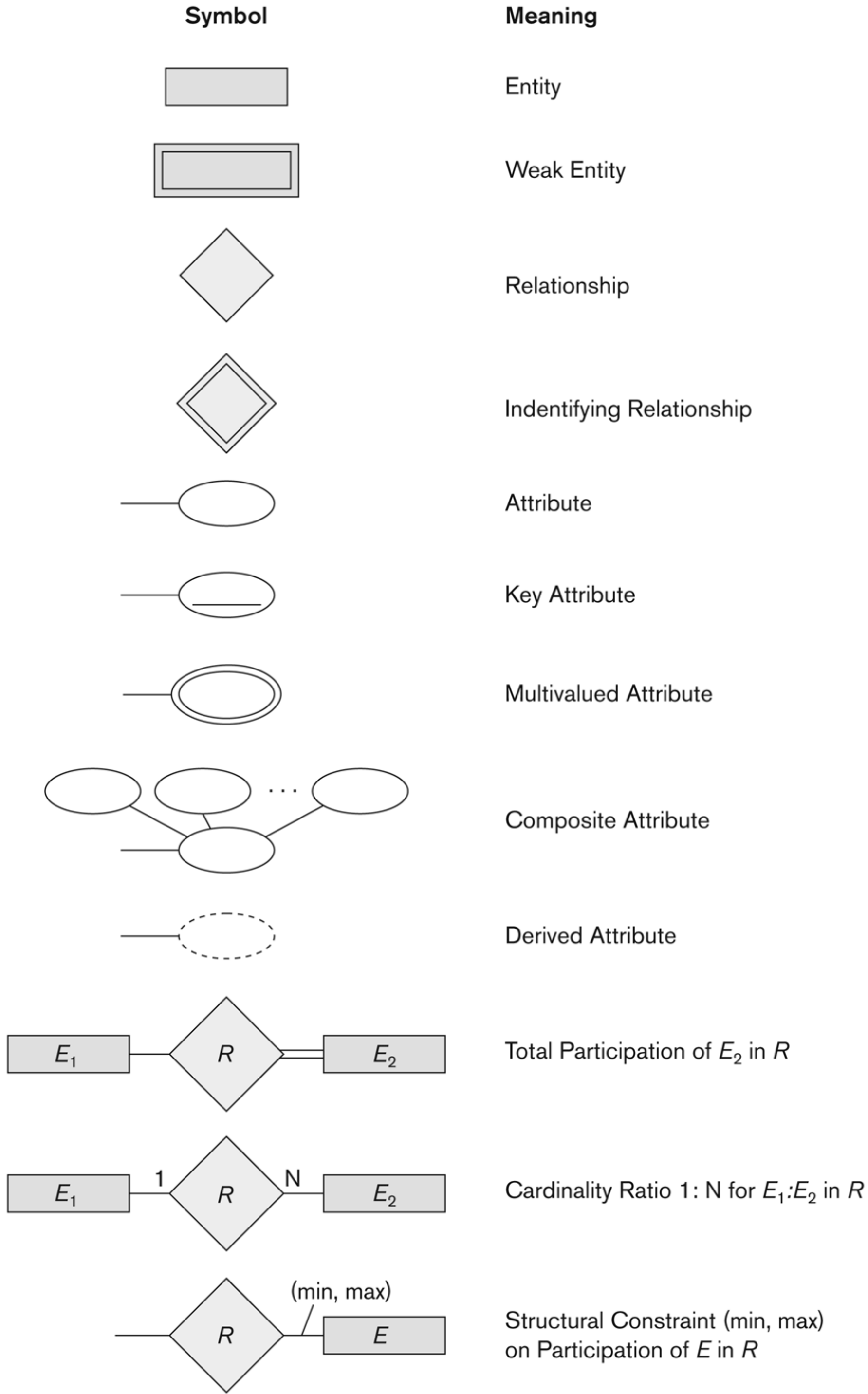


**Figure 3.15**
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

# Summary of notation for ER diagrams

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



Entity

Weak Entity

Relationship

Indentifying 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 $R$