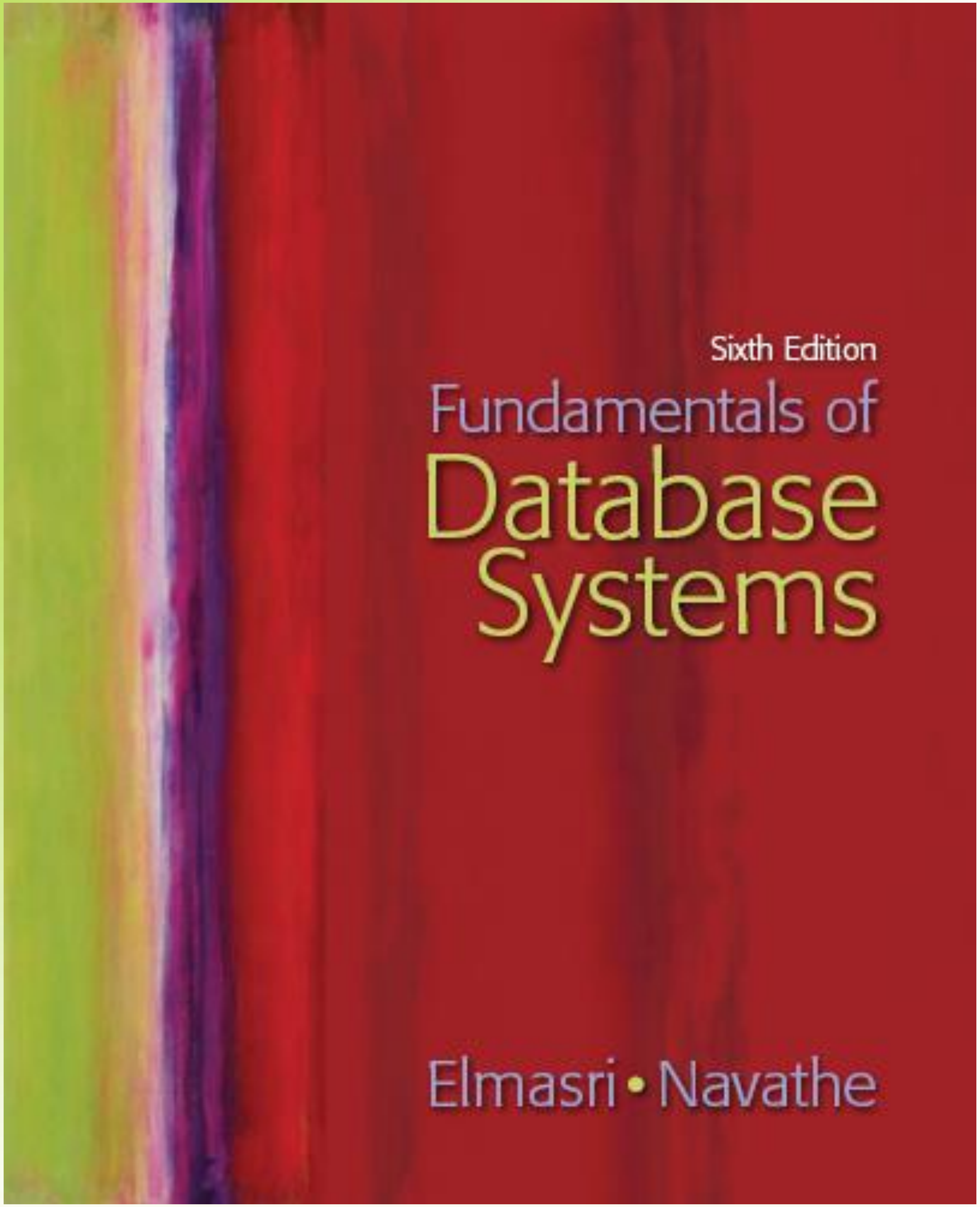
Chapter 7

Conceptual Modeling and Database Design



Sixth Edition

Fundamentals of Database Systems

Elmasri • Navathe

Addison-Wesley is an imprint of PEARSON

Chapter 7

Data Modeling Using the Entity-Relationship Model

Addison-Wesley
is an imprint of
PEARSON

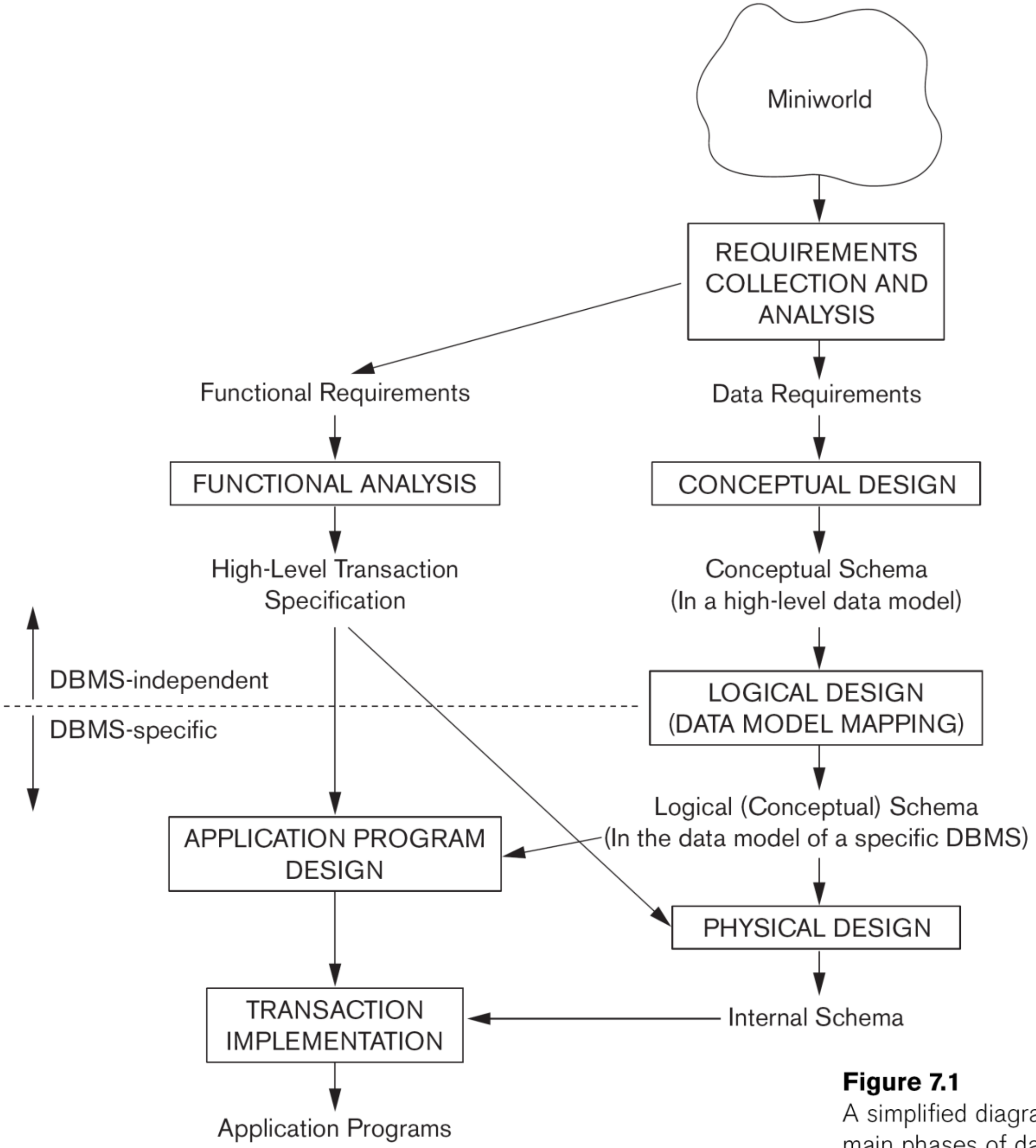
Chapter Outline

- Overview of the 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 Model Notations, Naming Conventions, Design Issues
- ER Diagram for COMPANY Schema
- Alternative Notations – UML class diagrams
- n-ary Relationships with degree n > 2

Addison-Wesley
is an imprint of
PEARSON

Overview of the Database Design Process

- Two main activities (see Figure 7.1, next slide):
  - Database schema design
  - Application programs design
- Focus in this chapter on database schema design
  - Given the database requirements, design the conceptual schema for a database
- Application programs design focuses on the programs and interfaces that access and update the database (considered part of *software engineering* discipline)



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

The Database Design Process

- *Requirements Analysis and Specification* is in the realm of Systems Analysis and Design
  - In this chapter, we assume it is already completed
  - Briefly discussed in Chapter 10
- This and next chapter focus on *Conceptual Design* (see Figure 7.1)
  - *Physical Design* discussed in Chapter 20, after presenting file structures and indexing (Chapters 17, 18)
  - *Logical Design* presented in Chapter 9

The Entity-Relationship (ER) Model

- ER model is a *conceptual data model* for database design
  - Has an associated notation (**ER schema diagrams**) for drawing/displaying the database schema
  - Many variations of ER model exists
  - Also, many extensions (see EER model, Chapter 8)
- Next slide (Figure 7.2) shows a complete ER schema diagram for a COMPANY database
  - We will explain gradually how this design is created
  - First we introduce the requirements for the COMPANY database
  - Then we present ER model concepts and diagrammatic notation gradually, and design the schema step-by-step

Addison-Wesley is an imprint of PEARSON

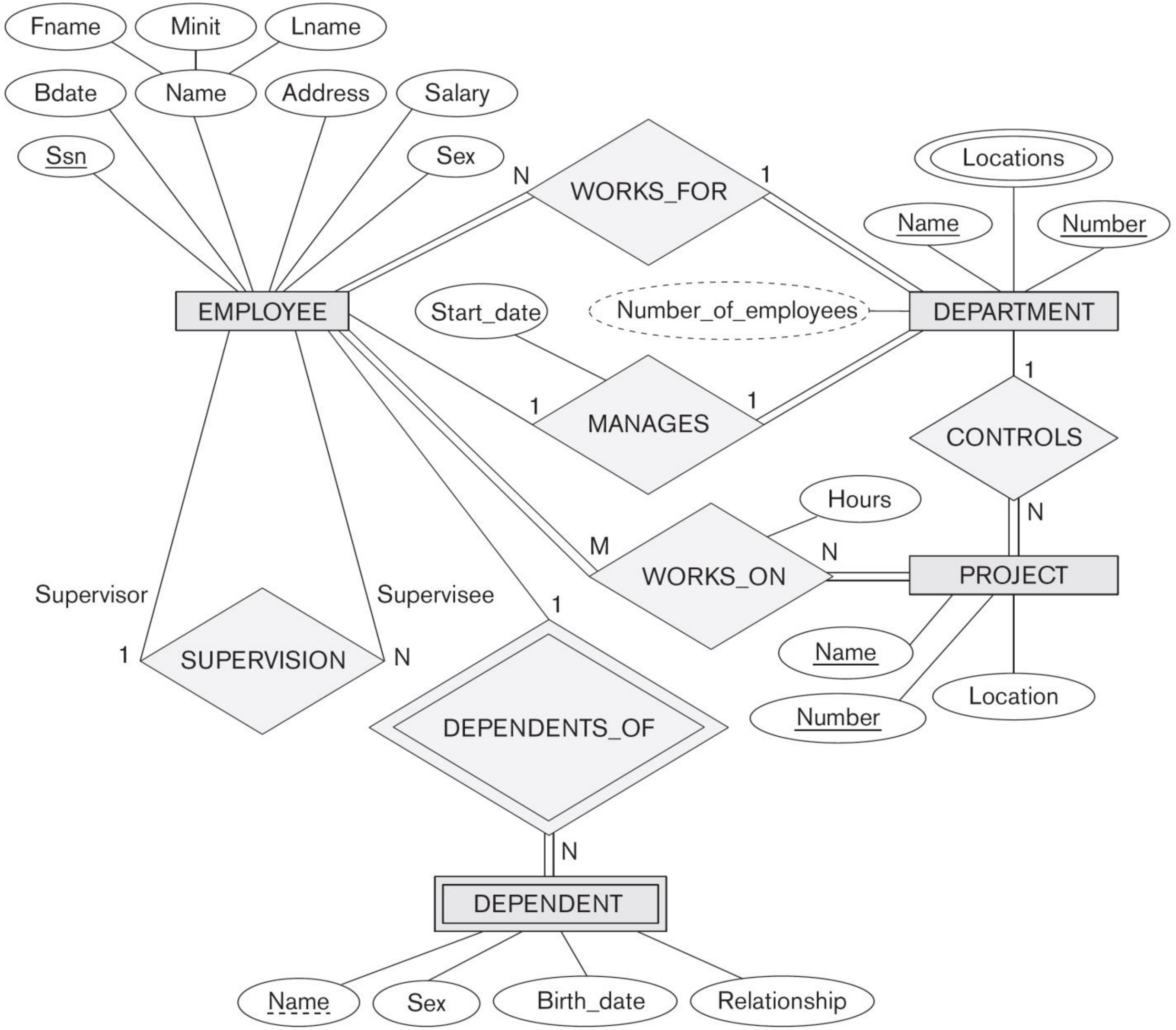


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Addison-Wesley
is an imprint of

PEARSON

Example: COMPANY Database

- Create a database schema design based on the following (simplified) **requirements** for a 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.

COMPANY Database (cont.)

- The database will store each EMPLOYEE’s name, social security number (unique for each employee), address, salary, sex, and birthdate.
  - 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.
- An employee can *have* DEPENDENTs.
  - For each dependent, the database keeps track of their first name, sex, birthdate, and their relationship to the employee (child, spouse, etc.).

ER Model Concepts

- Entities and Attributes
  - **Entities**: Specific objects or things in the mini-world that are represented in the database.
    - Examples: the EMPLOYEE John Smith, the Research DEPARTMENT, the ProductX PROJECT
  - **Attributes**: Properties used to describe an entity.
    - Examples: an EMPLOYEE entity may have the attributes Name, SSN, Address, Sex, BirthDate
  - Data **values**: A specific entity has a value for each of its attributes.
    - Example: An 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, subrange, enumerated type, …

Types of Attributes

- **Simple attribute** (sometimes called *atomic*):
  - Each entity has a single value for the attribute. For example, the SSN or Sex of an employee.
  - Not divisible
- **Composite attribute** (also called *compound)*:
  - 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 (Figure 7.4, next slide).
- **Multi-valued attribute** (also called *repeating group* or *collection*):
  - An single entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
    - Denoted as {Color} or {PreviousDegrees}.

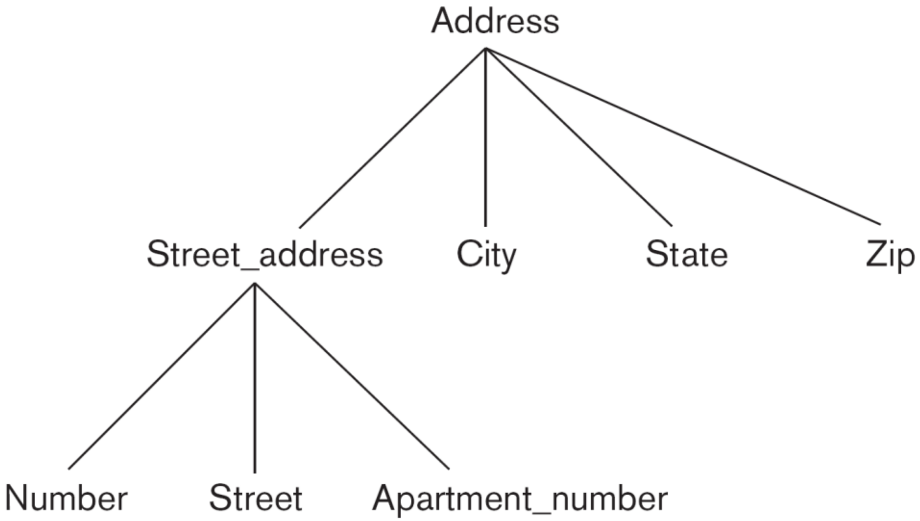


Figure 7.4
A hierarchy of composite attributes.

[3] Zip Code is the name used in the United States for a five-digit postal code, such as 76019, which can be extended to nine digits, such as 76019-0015. We use the five-digit Zip in our examples.

Types of Attributes (cont.)

- Composite and multi-valued attributes may be nested (to any number of levels).
  - – Example: PreviousDegrees of a STUDENT can be a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
  - – Multiple PreviousDegrees values can exist for a particular student
  - – Each has four subcomponent attributes:
    - College, Year, Degree, Field
- Stored vs Derived attribute
  - Age & Birth_date

Entity Types and Key Attributes

- Entities with the same basic attributes are grouped (or typed) into an **entity type**.
  - Examples: EMPLOYEE or PROJECT.
- **Key attribute:** an attribute of an entity type for which each entity must have a unique (distinct) value.
  - Example: SSN of EMPLOYEE, or PNUMBER of PROJECT, or PNAME of PROJECT.

Addison-Wesley is an imprint of PEARSON

Entity Types and Key Attributes (cont.)

- A key attribute may be *composite*.
  - Example: VehicleTagNumber (also known as LicensePlateNo) of a CAR is a key with two components (LicNumber, State).
- An entity type may have *more than one key*.
  - The CAR entity type may have two keys:
    - VehicleIdentificationNumber (popularly called VIN, unique number stamped on each new car)
    - VehicleTagNumber (Number, State)
- Each key is <u>underlined</u> in ER diagrams (see next slides)

Addison-Wesley is an imprint of

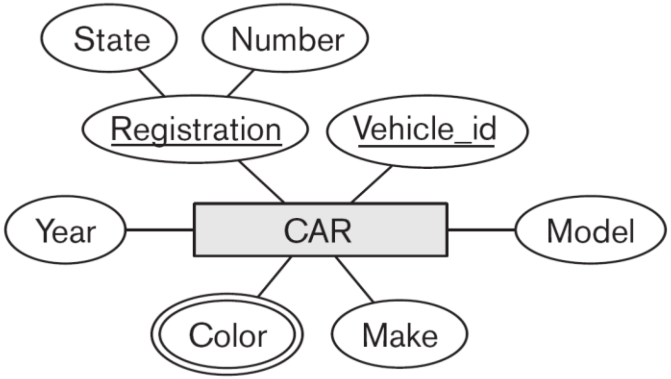
PEARSON

Displaying an Entity type

- In ER diagrams, the entity type name 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 (Figure 7.7(a)) on next slide

Figure 7.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(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})

⋮

[5] We use a notation for ER diagrams that is close to the original proposed notation (Chen 1976). Many other notations are in use; we illustrate some of them later in this chapter when we present UML class diagrams and in Appendix A.

Entity Set

- Each entity type will have a *collection of individual entities* stored in the database
  - Called the **entity set**
  - Previous slide (Figure 7.7(b) show three CAR entities in the entity set for CAR
  - Same name (CAR) refers to both entity type and entity set
  - Object models (see Chapter 11) give *different names* to the entity type and the entity set
  - Entity set *changes over time* as entities are created and deleted – represents *current state* of database

Addison-Wesley is an imprint of PEARSON

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
- Initial design (Figure 7.8) on following slide, will be refined into final design
- Initial attributes shown are derived from the requirements description

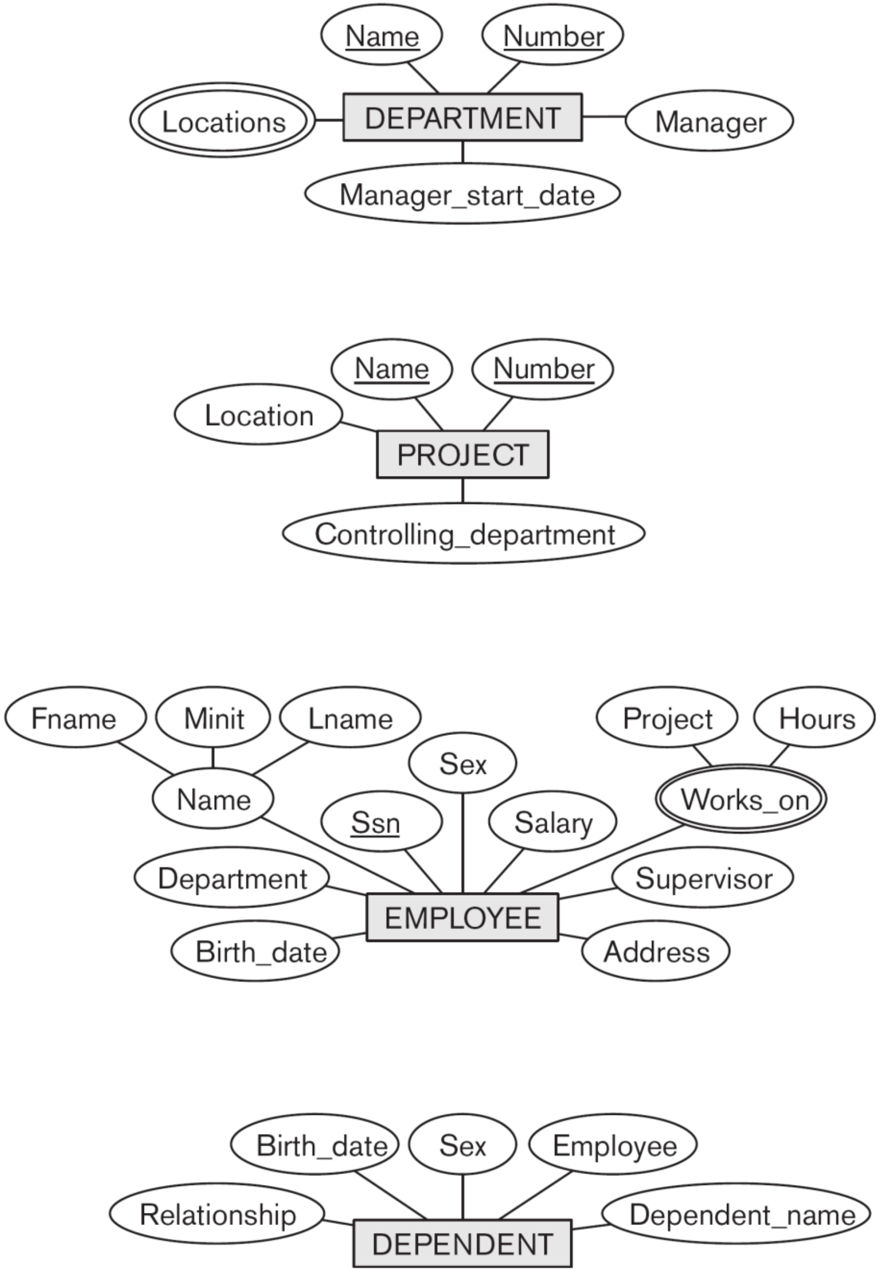


Figure 7.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

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

- 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 Type vs. Relationship Set

- 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

Addison-Wesley
is an imprint of

PEARSON

Relationship Set

- A set of *associations* (or *relationship instances*) between individual entities from the participating entity sets:
  - – Example: Figure 7.9 (next slide) shows a relationship set for WORKS_FOR
  - – {r1, r2, r3, r4, r5, r6, r7, ...}
  - – Relationship instance r1=(e1, d1) means EMPLOYEE e1 *WORKS_FOR* DEPARTMENT d1
  - – Associates e1 with d1

Addison-Wesley
is an imprint of
PEARSON

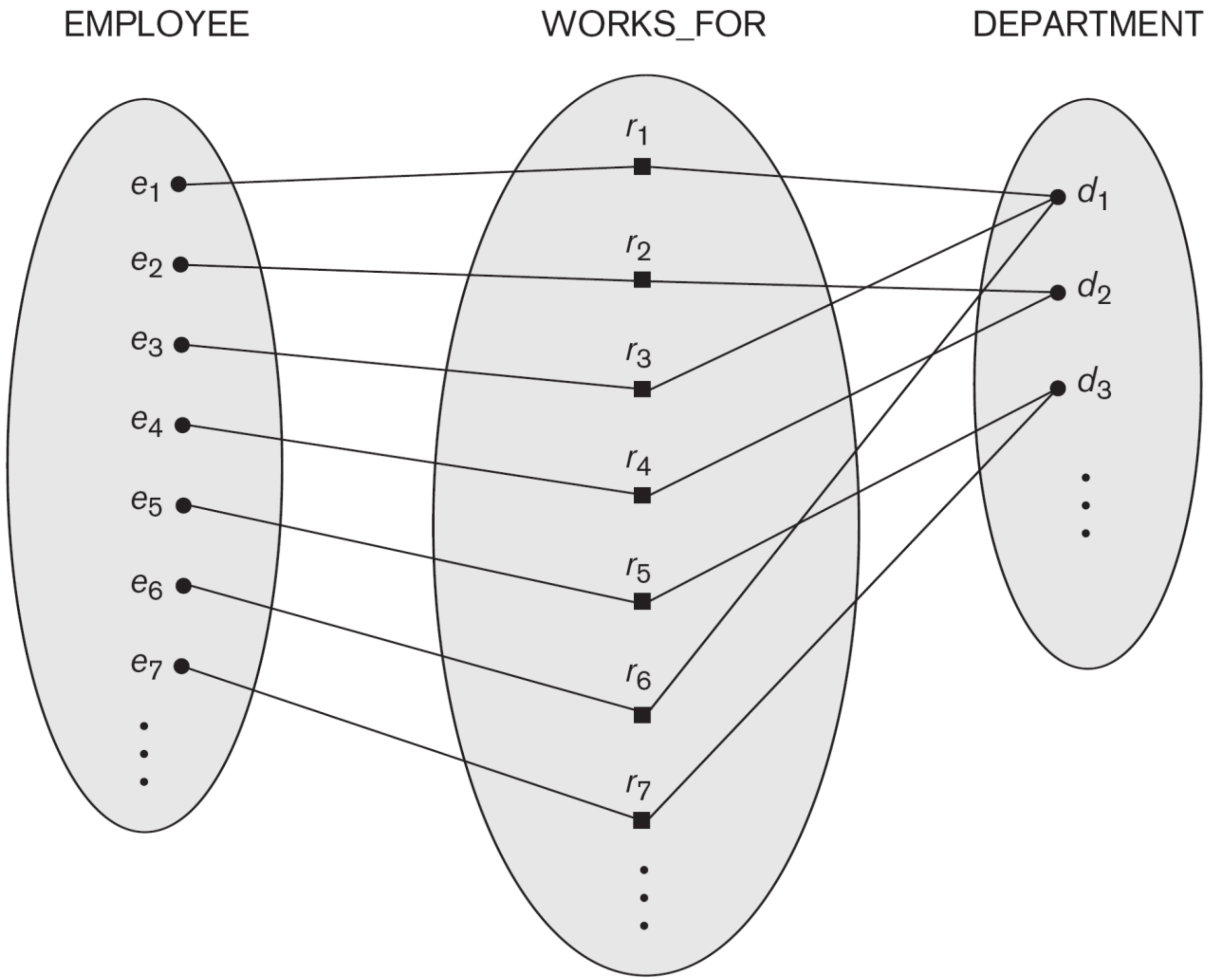


Figure 7.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Addison-Wesley is an imprint of
PEARSON

Relationship Type

- Previous figure displayed the relationship set
- 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
  - **Degree** of a relationship type is the *number of participating entity types*

Refining the COMPANY Initial Design by Including Relationships

- By examining the requirements, attributes in the initial design that *refer to other entities* are *converted into* relationships (and removed from the entity types)
- Some of these *relationship attributes* (Figure 7.8, repeated on next slide) are:
  - The Department attribute of EMPLOYEE refers to the DEPARTMENT entity that the employee WORKS_FOR
  - The Manager attribute of DEPARTMENT refers to the EMPLOYEE entity who MANAGES the DEPARTMENT
  - The Supervisor attribute of EMPLOYEE refers to *another* EMPLOYEE entity (this is called a *recursive relationship*)
  - Several other similar attributes are converted into relationships – can you identify those in next slide?

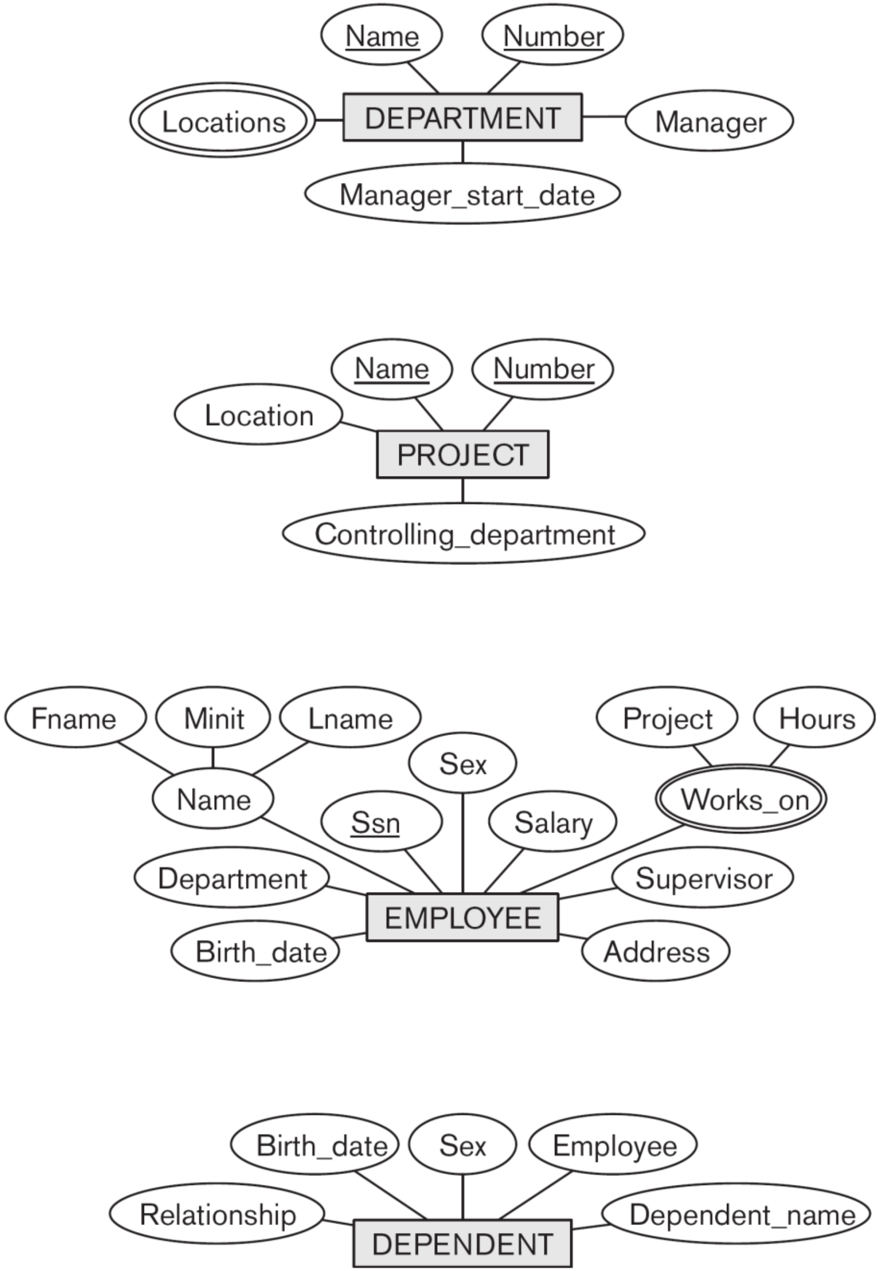


Figure 7.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Refining the COMPANY Initial Design by Including Relationships (cont.)

- Six relationship types are identified for the COMPANY database schema (see Figure 7.2, repeated next slide)
- 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)

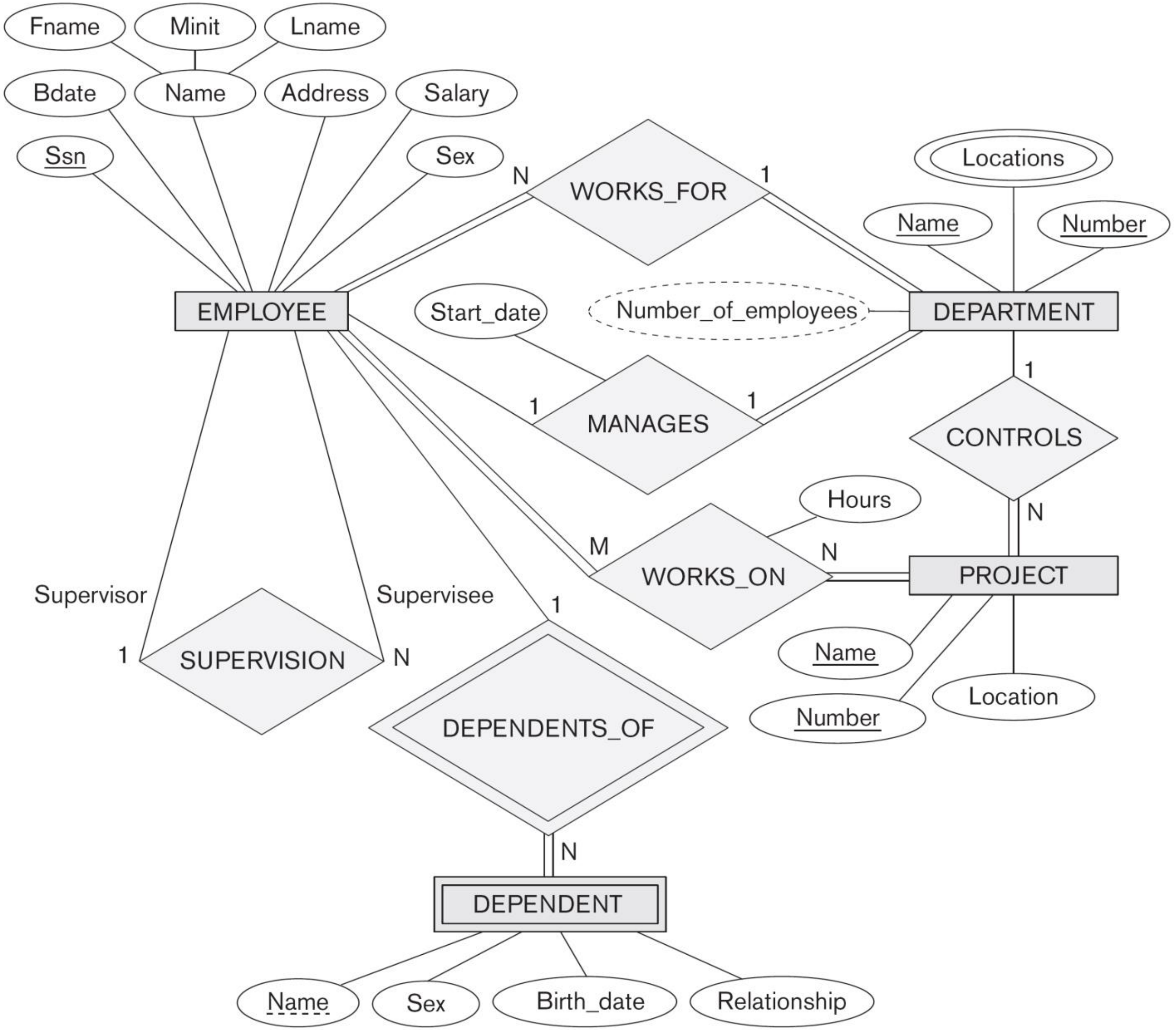


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Addison-Wesley
is an imprint of
PEARSON

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.

Addison-Wesley
is an imprint of
PEARSON

Recursive Relationship Type

- A relationship type with the same entity type participating twice in **two distinct roles**
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
  - supervisor (or boss) role
  - supervisee (or subordinate) role
  - must distinguish the roles in a relationship instance
- Each relationship instance ri relates two distinct EMPLOYEE entities (see Figure 7.11, next slide):
  - One employee in *supervisor* role (labeled 1 in Fig. 7.11)
  - One employee in *supervisee* role (labeled 2 in Fig. 7.11)

Addison-Wesley
is an imprint of
PEARSON

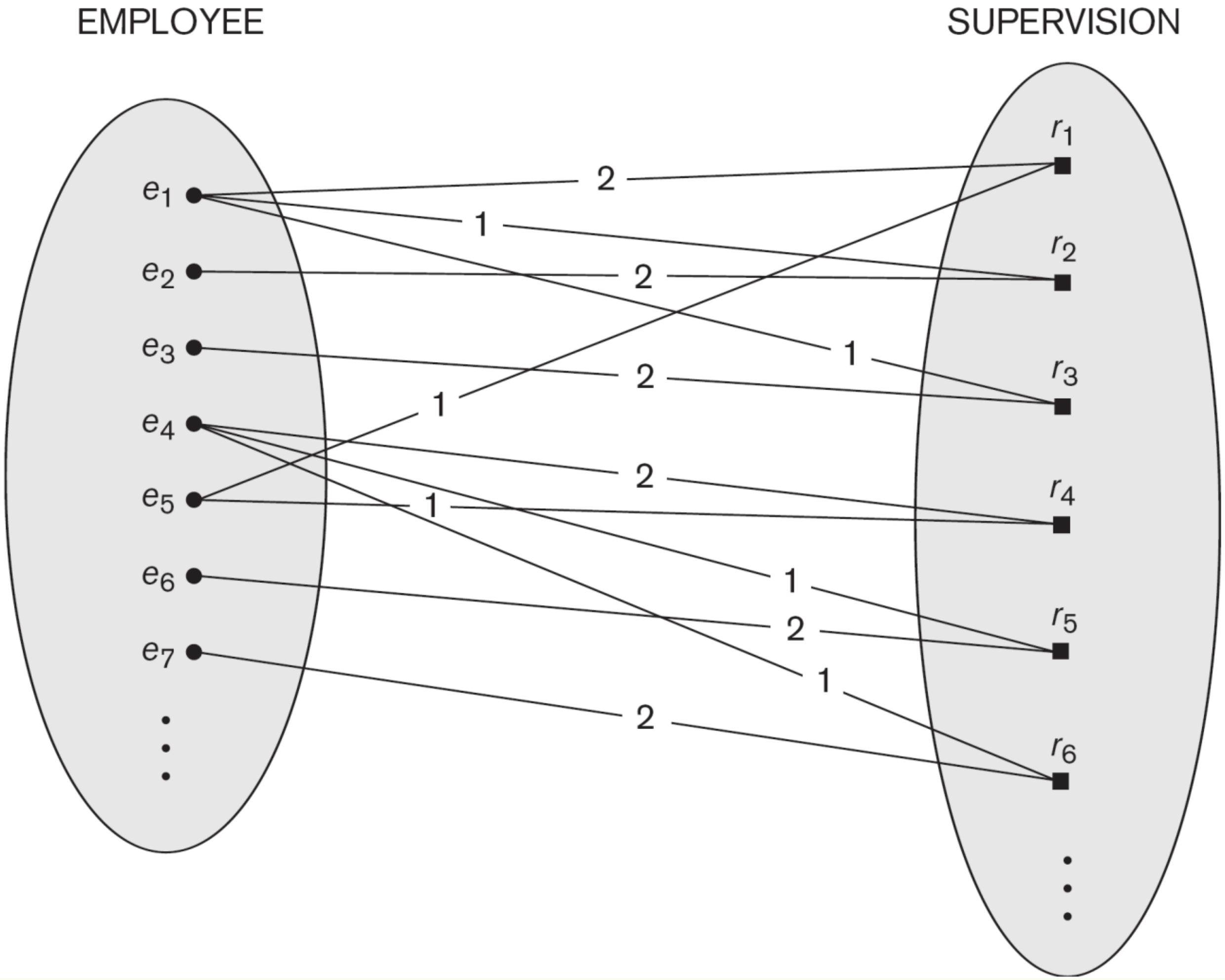


Figure 7.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Weak Entity Types

- An entity type that does not have a key attribute on its own
- A weak entity must participate in an *identifying relationship type* with an *owner* (or identifying) entity type
- Individual 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** (see Figure 7.2)**:**
  - 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 (owner) entity type via the identifying relationship type DEPENDENT_OF

Constraints on Relationships

- Constraints on Relationship Types
  - Two main types of constraints on binary relationships
  - **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)

Addison-Wesley
is an imprint of
PEARSON

Constraints on Relationships (cont.)

- Cardinality ratio specified by labeling 1, M, or N to relationship lines in ER diagrams.
- See Figure 7.2, repeated in next slide
- Total participation specified by **double** line, partial participation by **single** line.
- These constraints are derived from the *real-world meaning and characteristics* of each relationship type
- In some ER diagrammatic notation, it is common to specify cardinality ration and participation constraint jointly using (min, max) notation
  - Called **(min, max) constraints** or **multiplicities**

Addison-Wesley
is an imprint of

PEARSON

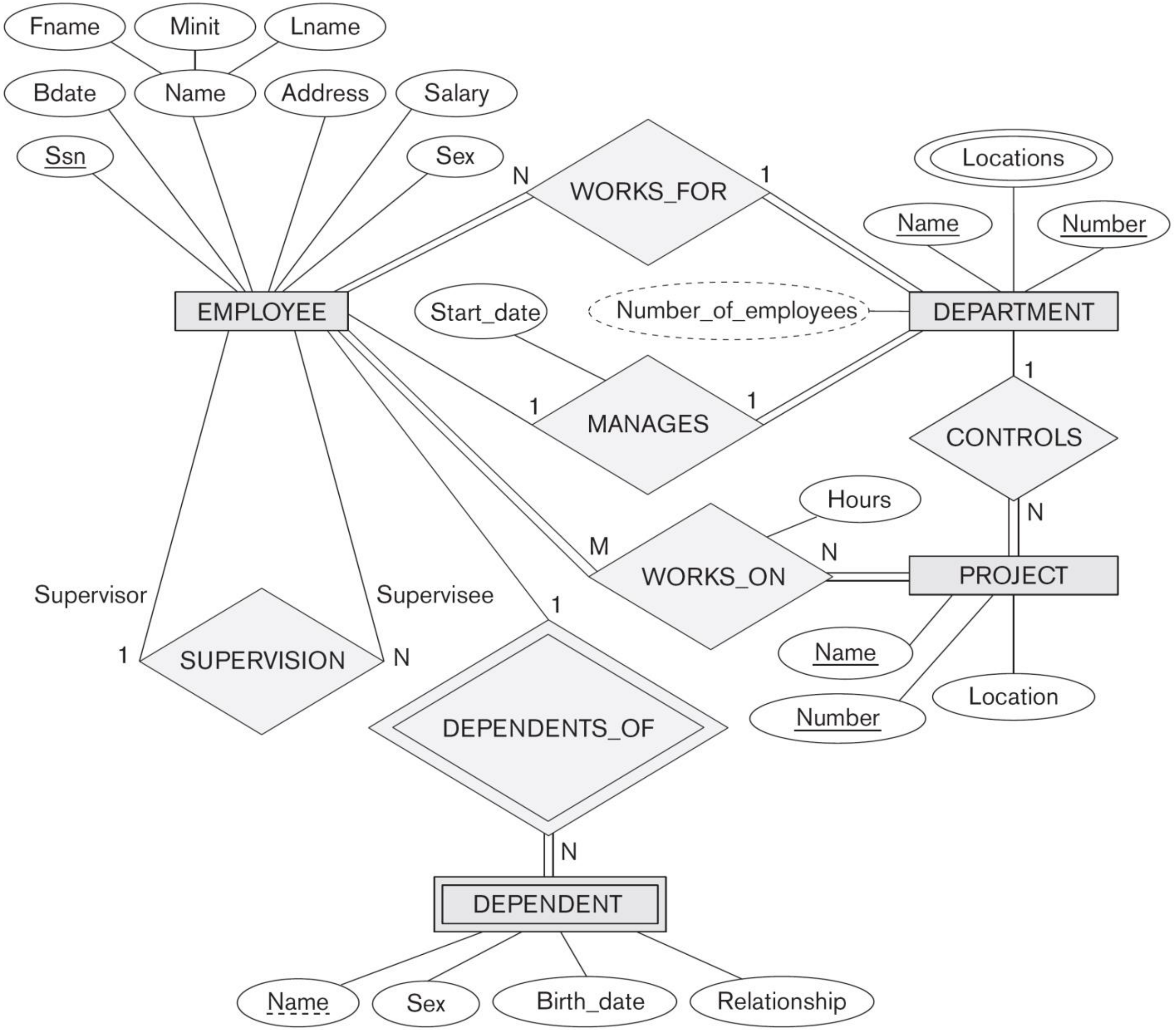


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Addison-Wesley
is an imprint of

PEARSON

Displaying a Recursive Relationship Type in ER Diagrams

- 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 ER diagram, need to display role names to distinguish participations (see Figure 7.2).
- Role names can also be optionally displayed for other relationship types

Addison-Wesley
is an imprint of
PEARSON

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

Addison-Wesley is an imprint of PEARSON

Alternative (min, max) notation

- Alternative way to specify relationship constraints; pecified 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 limits)
- 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

Addison-Wesley
is an imprint of
PEARSON

Alternative (min, max) notation (cont.)

- Figure 7.15 (next slide) shows the complete COMPANY ER schema diagram with the **(min, max) notation**
- Also shows all the (optional) **role names**
- Important: In some popular diagrammatic notations, the placement of (min, max) are reversed (placed on the other side of the binary relationship) – for example, in UML class diagrams (see later in this chapter)

Addison-Wesley
is an imprint of
PEARSON

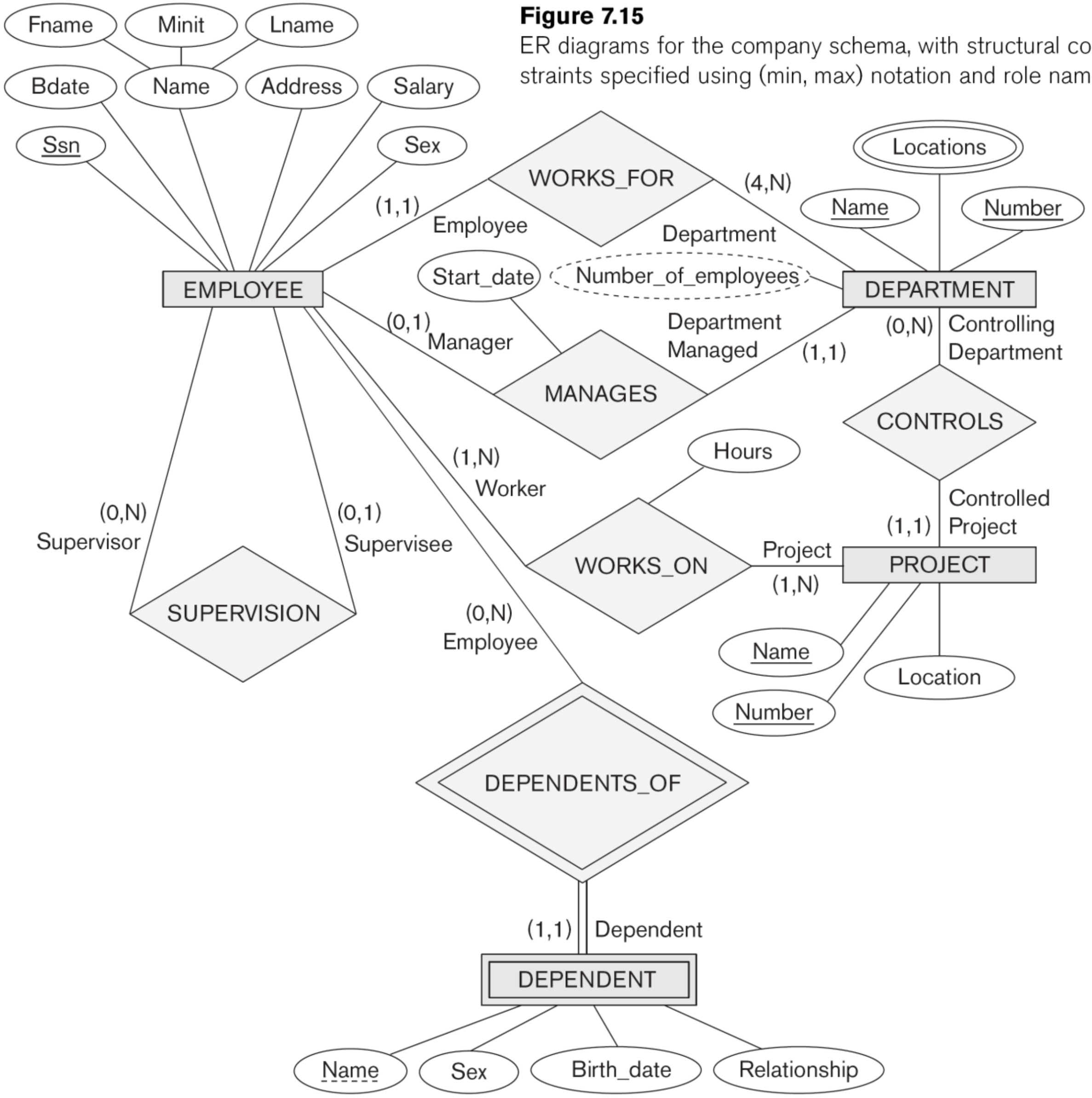


Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

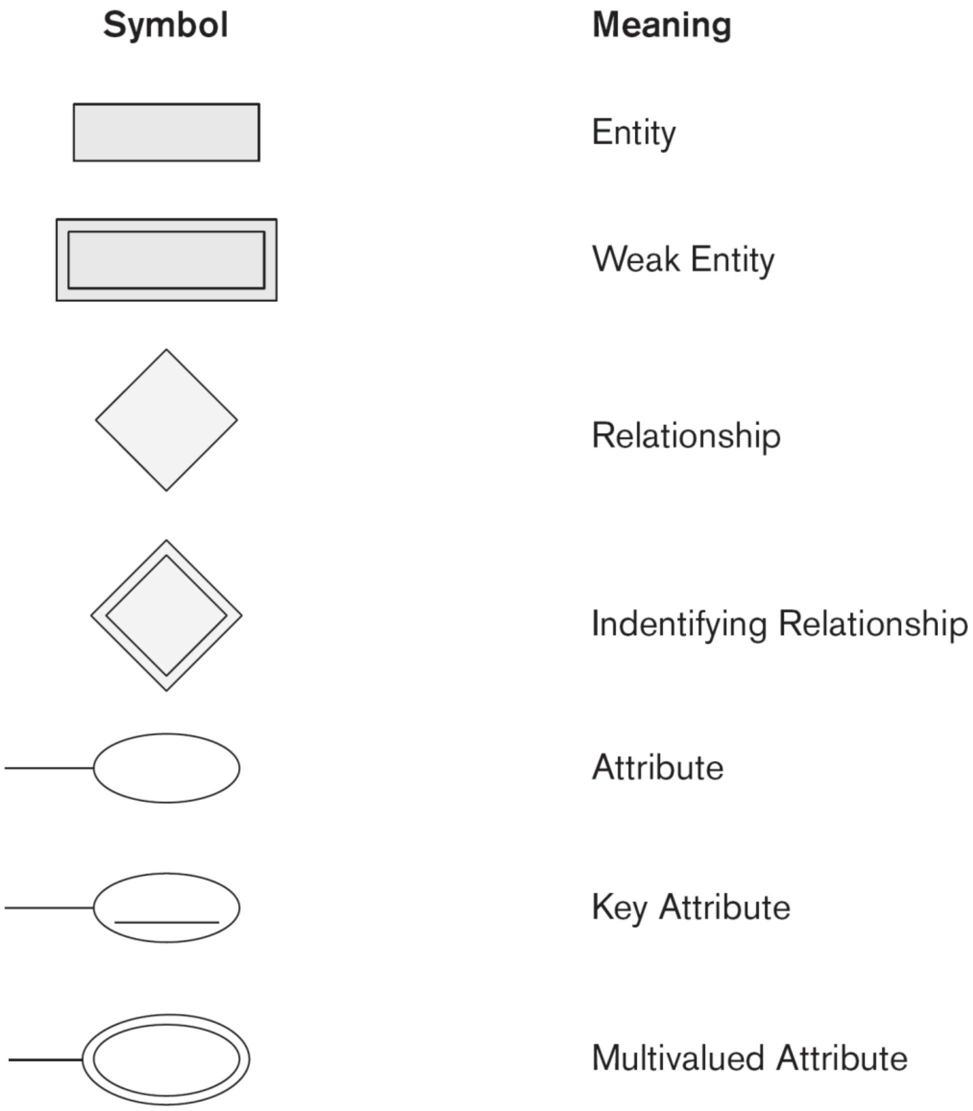
[14] In some notations, particularly those used in object modeling methodologies such as UML, the (min, max) is placed on the *opposite sides* to the ones we have shown. For example, for the WORKS_FOR relationship in Figure 7.15, the (1,1) would be on the DEPARTMENT side, and the (4,N) would be on the EMPLOYEE side. Here we used the original notation from Abrial (1974).

Summary of ER Diagrams

- Next two slides (Figure 7.14) summarizes the ER Diagrammatic notations described so far

Addison-Wesley
is an imprint of

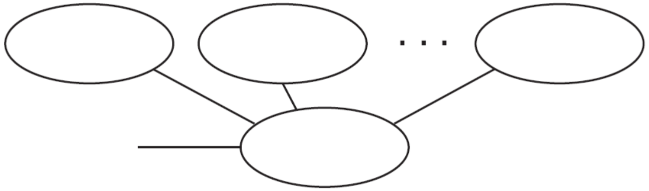
PEARSON



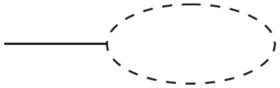
Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute

Figure 7.14
Summary of the notation for ER diagrams.

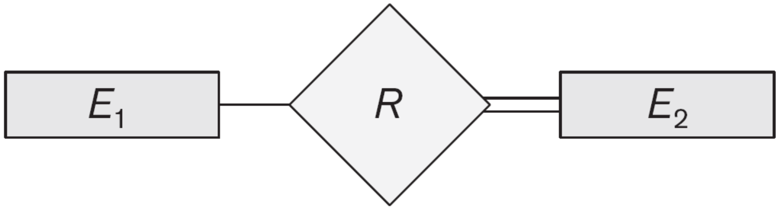
Continued next page…



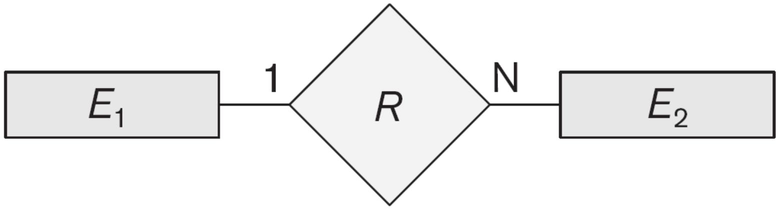
Composite Attribute



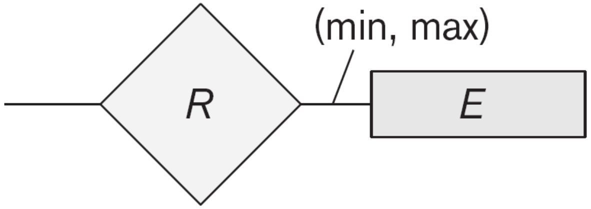
Derived Attribute



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

Alternative diagrammatic notation

- ER diagrams (as described here) is one popular method for displaying database schemas
- Many other diagrammatic 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 an alternative way of displaying ER concepts that is used in several automated design tools

UML Class Diagrams

- UML (Universal Modeling Language) is a popular language/methodology for object-oriented software design
- Part of software design is specifying **classes** using class diagrams – this is somewhat similar to ER design
- Classes (similar to entity types) as displayed as large rounded boxes with three sections:
  - Top section includes entity type (class) name
  - Second section includes attributes
  - Third section includes class operations (operations are not in basic ER model)
- Relationships (called associations) represented as lines connecting the classes
  - Other UML terminology also differs from ER terminology
- UML has many other types of diagrams for software design (see Chapter 10)

Addison-Wesley
is an imprint of
PEARSON

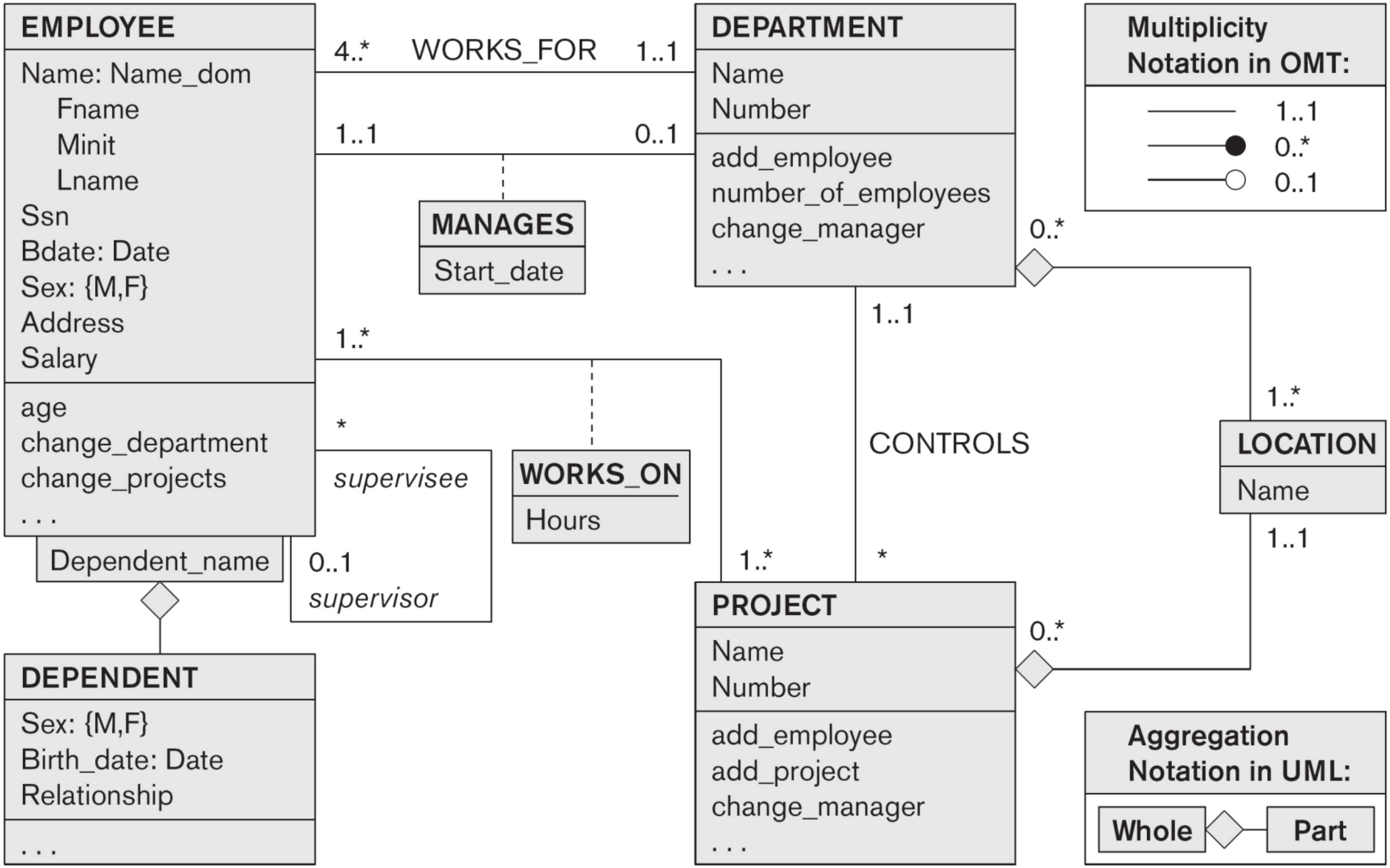
UML Class Diagrams (cont.)

- Next slide (Figure 7.16) shows example of UML class diagrams for the COMPANY database schema
- **Multiplicities** (similar to (min, max) constraints) placed on opposite end when compared to our previous notation:
  - Displayed as min..max
  - * represents no maximum limit on participation (like N)
- Two kinds of relationships
  - **Association**: Relationship between two independent objects; displayed as lines
  - **Aggregation**: Relationship between object and its parts; displayed as lines with small diamond at object end
- Weak entity can be represented using concept of **qualified association/aggregation (discriminator** similar to partial key)
- Relationship names are optional; relationship instances called **links;** relationship attributes called **link attributes**

Addison-Wesley is an imprint of

PEARSON

Figure 7.16
The COMPANY conceptual schema
in UML class diagram notation.



Addison-Wesley
is an imprint of

PEARSON

Relationships of Higher Degree

- Recall that **degree** of a relationship type is the *number of participating entities* in each instance
- Relationship types of degree 2 are called **binary**, degree 3 are **ternary**, and degree n are **n-ary**
- **Example:** A relationship instance in SUPPLY (Figure 7.10 , next slide) relates three entities (s, p, j) where s is a SUPPLIER, p a PART, j a PROJECT such that s *currently supplies* part p *to project* j (with Quantity items per month)
- In general, an n-ary relationship (where n > 2) is *not equivalent* to n binary relationships
- Constraints are *harder to specify* for higher-degree relationships (n > 2) than for binary relationships

Addison-Wesley
is an imprint of
PEARSON

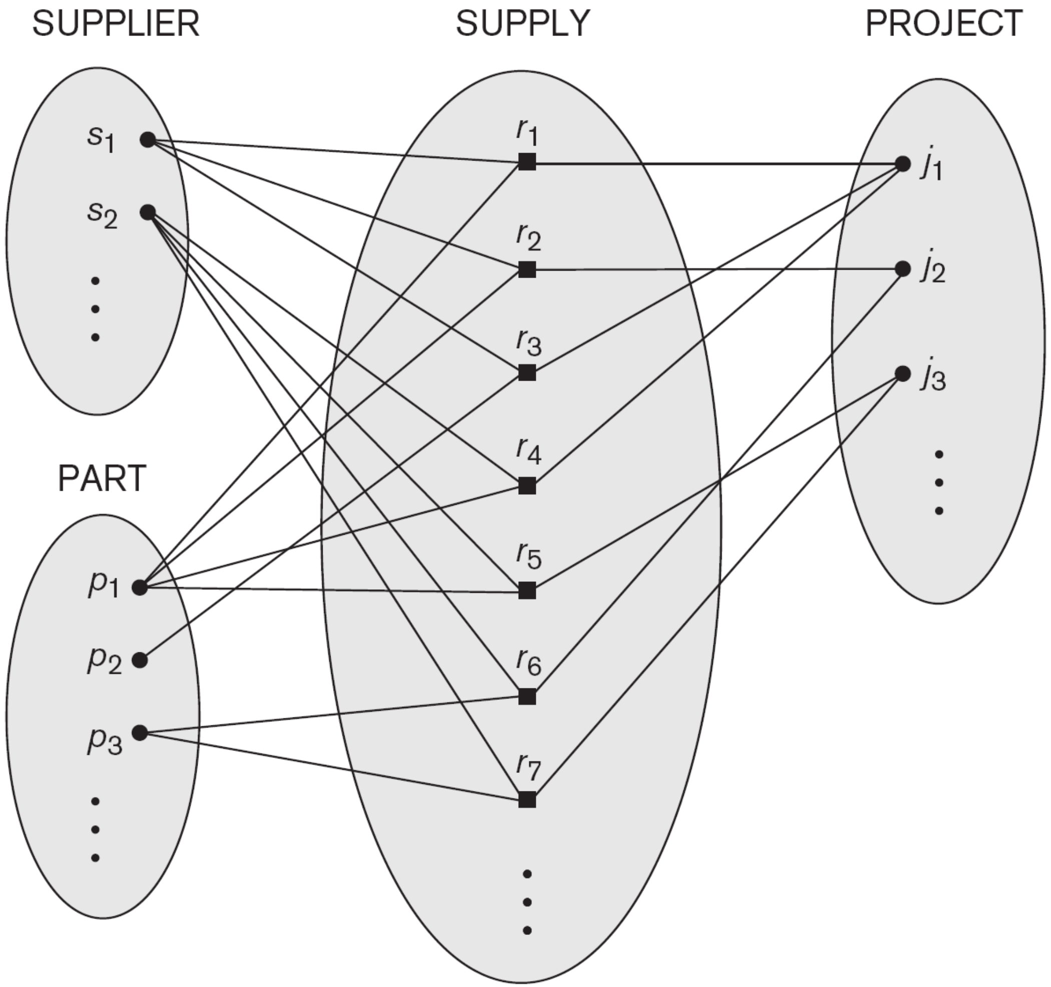


Figure 7.10
Some relationship instances in the SUPPLY ternary relationship set.

Addison-Wesley is an imprint of PEARSON

Discussion of n-ary relationships

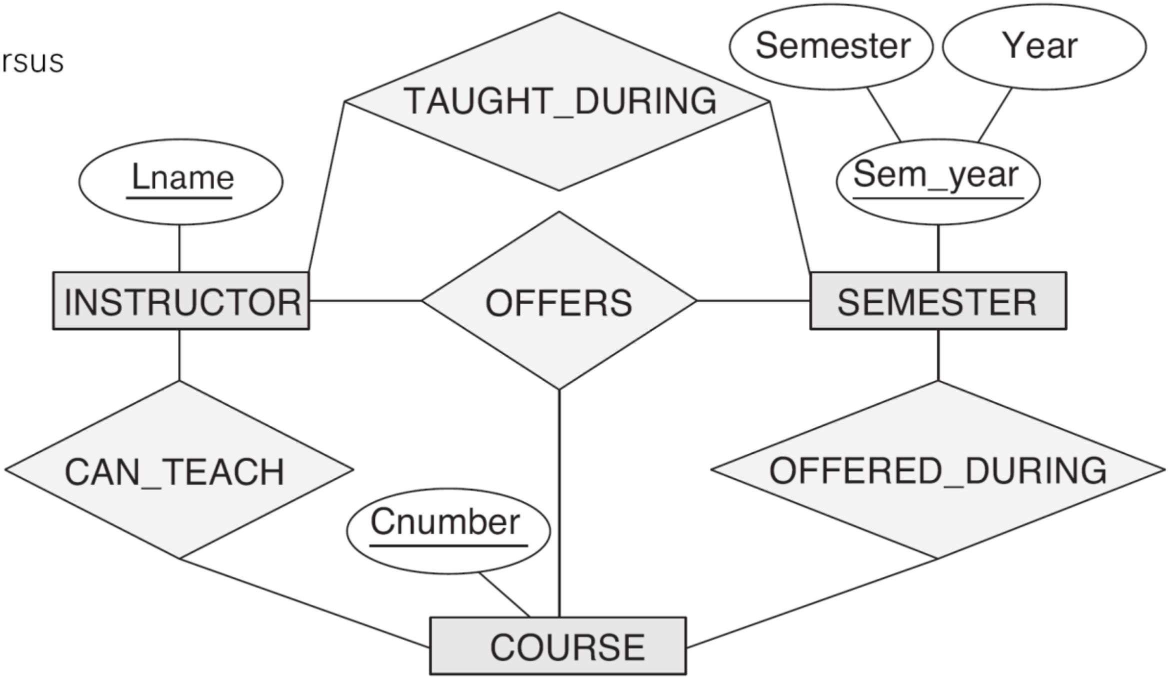
- In general, 3 binary relationships can represent *different information* than a single ternary relationship (see Figure 7.17a and b on next slide)
- If needed, the binary and n-ary relationships can all be included in the schema design
- In some cases, a ternary relationship can be represented as a weak entity type if the data model allows multiple identifying relationships (and hence multiple owner entity types) (see Figure 7.17c)

Addison-Wesley
is an imprint of
PEARSON

Discussion of n-ary relationships (cont.)

- If a particular binary relationship *can be derived from* a higher-degree relationship at all times, then it is redundant
- For example, TAUGHT_DURING binary relationship in Figure 7.18 (see next slide) can be derived from the ternary relationship OFFERS (based on the meaning of the relationships)
- It all depends on the meaning of the relationships in the real world

Figure 7.18
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

Extended Entity-Relationship (EER) Model (in next chapter)

- The basic ER model described so far does not support specialization and generalization abstractions
- Next chapter illustrates how the ER model can be extended with
  - – Type-subtype and set-subset relationships
  - – Specialization/Generalization Hierarchies
  - – Notation to display them in EER diagrams

Chapter 7 Summary

- ER Model Concepts: Entities, attributes, relationships
- Constraints in the ER model
- Using ER in step-by-step conceptual schema design for the COMPANY database
- ER Diagrams - Notation
- Alternative Notations – UML class diagrams, others

Additional Examples

- Next few slides are additional figures from the chapter, and figures from the Chapter 7 exercises

Addison-Wesley
is an imprint of

PEARSON

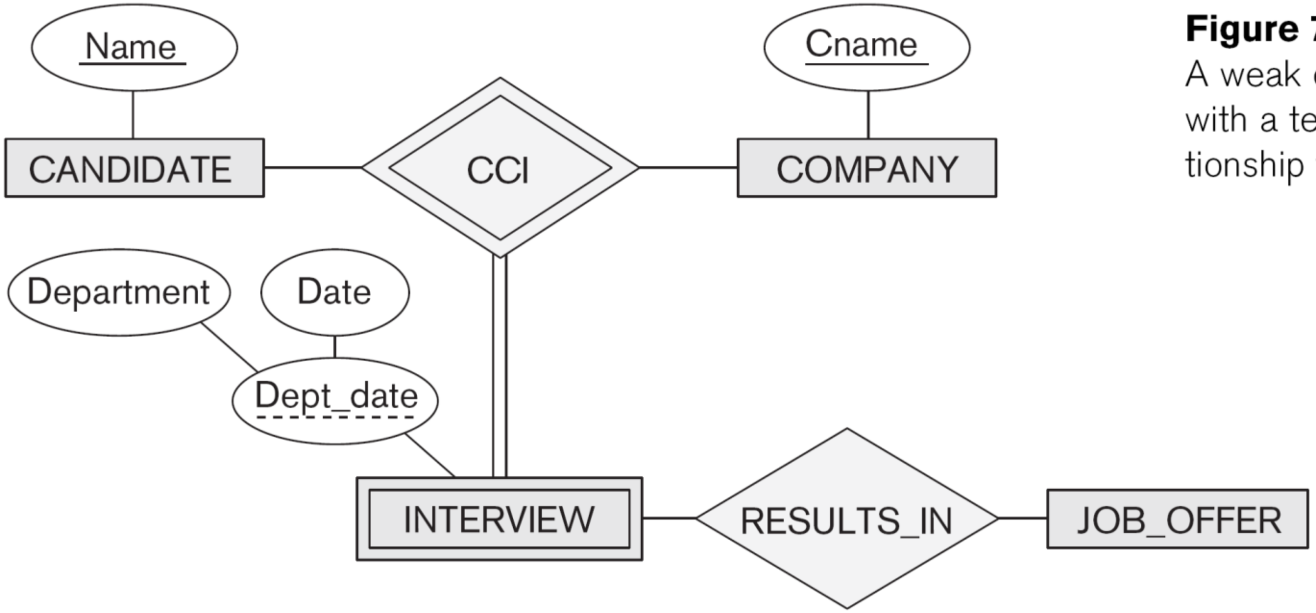
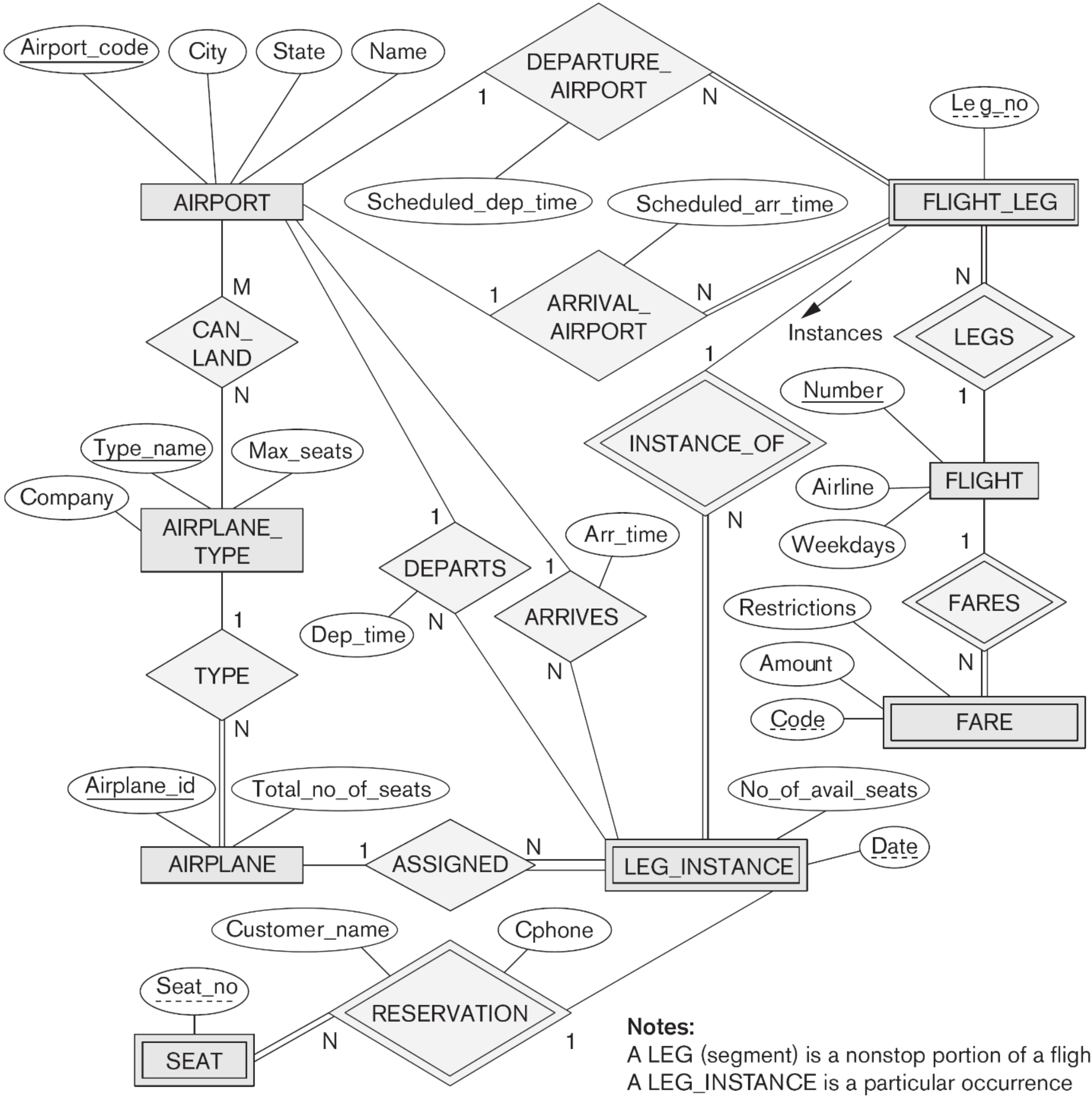


Figure 7.19
A weak entity type INTERVIEW with a ternary identifying relationship type.

Addison-Wesley
is an imprint of
PEARSON

Figure 7.20
An ER diagram for an AIRLINE database schema.



Notes:
A LEG (segment) is a nonstop portion of a flight.
A LEG_INSTANCE is a particular occurrence
of a LEG on a particular date.

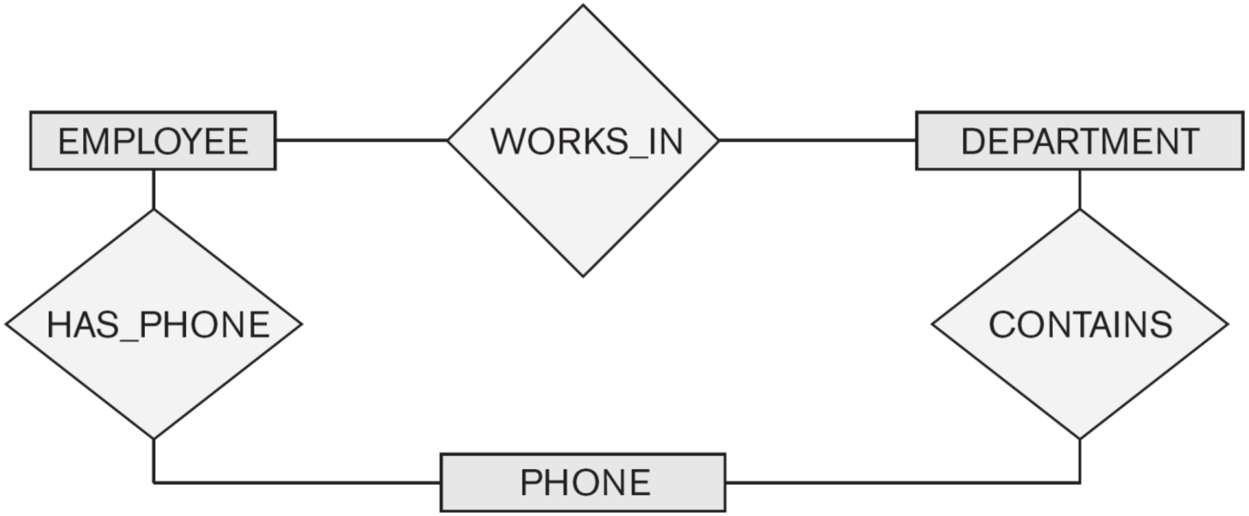


Figure 7.22
Part of an ER diagram for a COMPANY database.

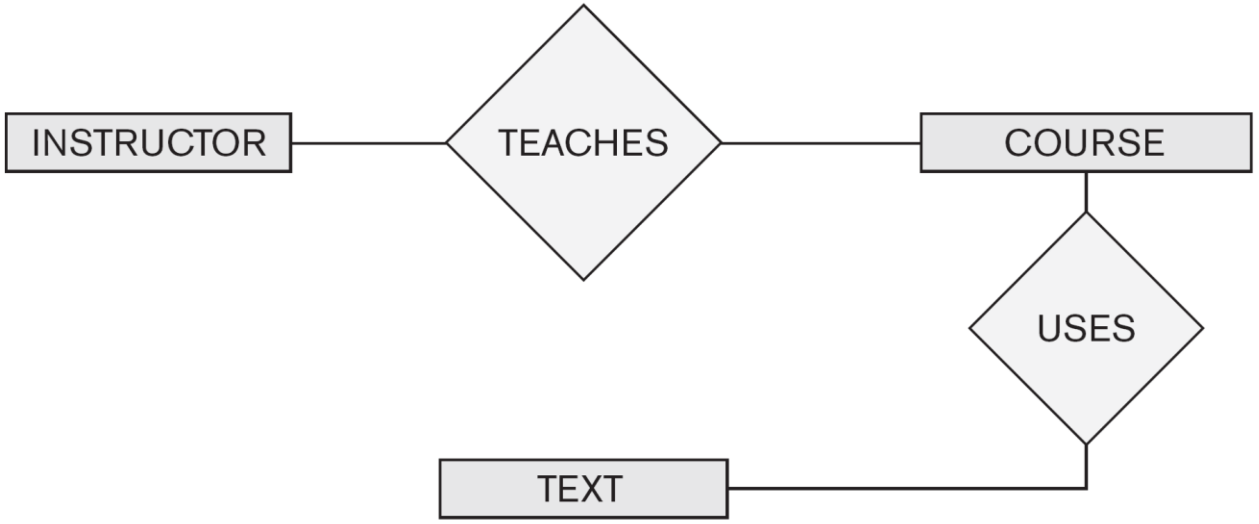


Figure 7.23
Part of an ER diagram for a COURSES database.