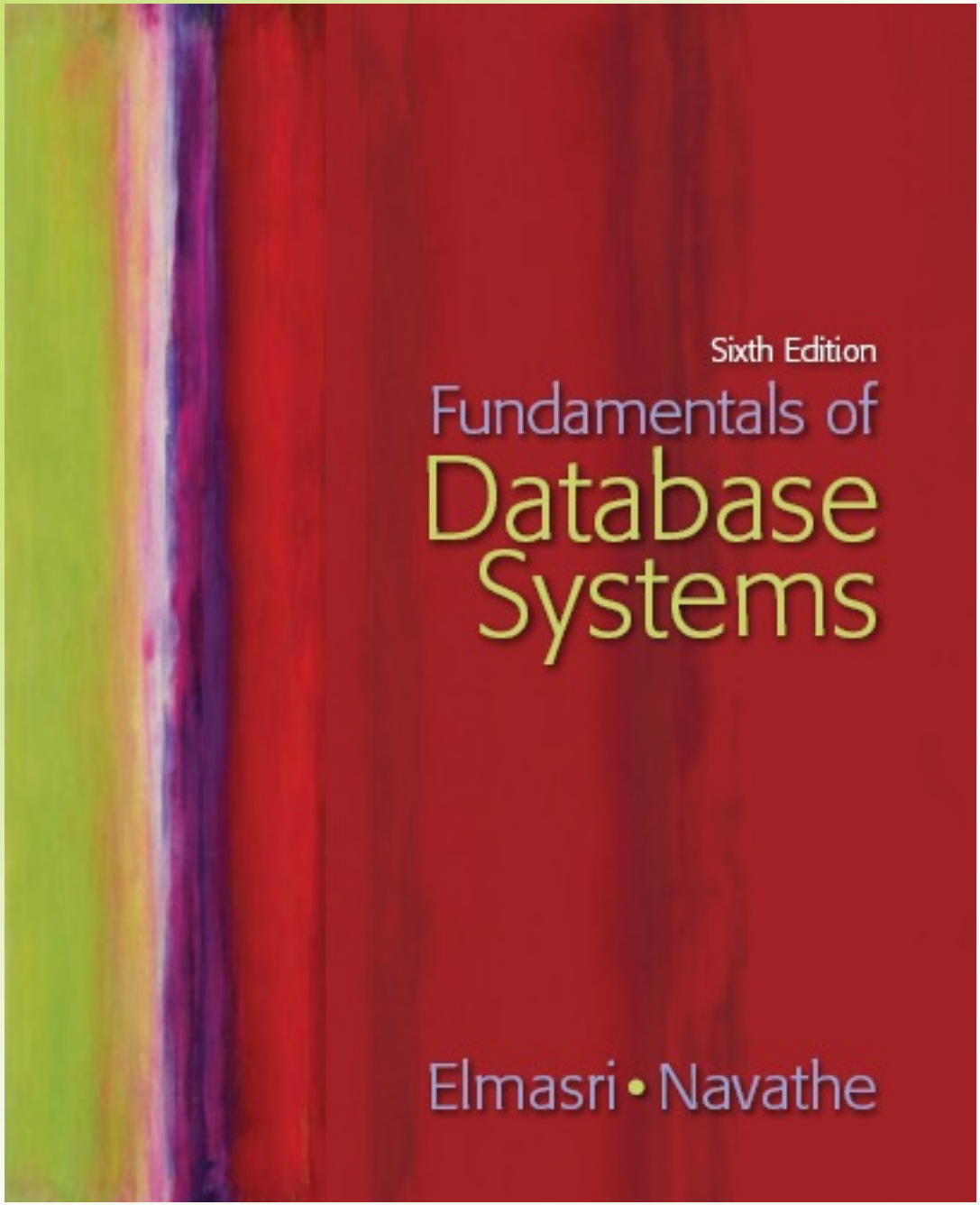
Chapter 7

(Chapter 3 in 7th edition)

Data Modeling Using the Entity-Relationship (ER) Model



Addison-Wesley
is an imprint of
PEARSON

Chapter 7 Outline

- 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

Chapter 7 Outline (cont'd.)

- Refining the ER Design for the COMPANY Database
- ER Diagrams, Naming Conventions, and Design Issues
- Example of Other Notation: UML Class Diagrams
- Relationship Types of Degree Higher than Two

Data Modeling Using the Entity-Relationship (ER) Model

- **Entity-Relationship (ER) model**
  - Popular high-level conceptual data model
- **ER diagrams**
  - Diagrammatic notation associated with the ER model
- **Unified Modeling Language (UML)**

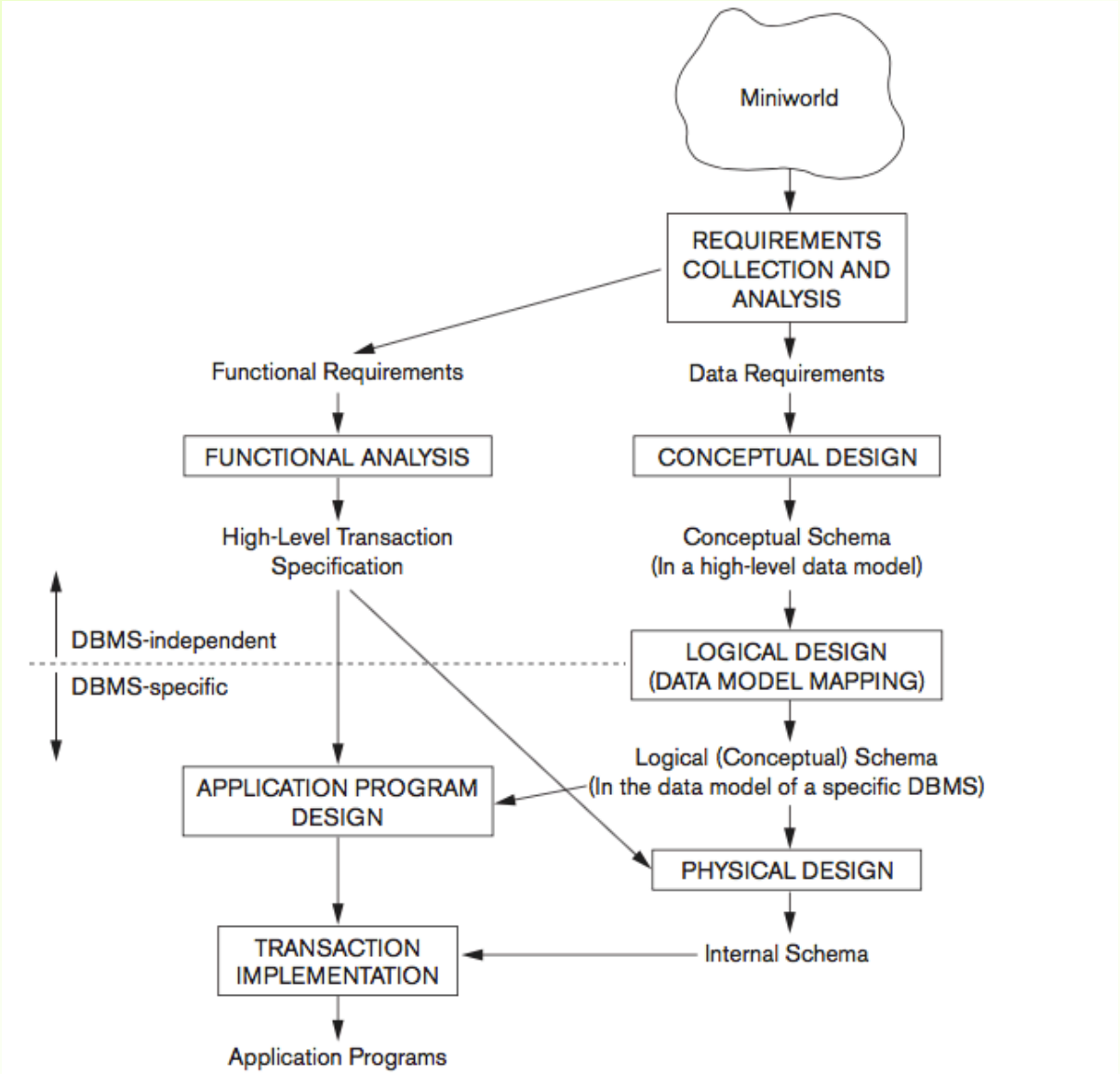


Figure 3.1
A simplified diagram to illustrate the main phases of database design.

Using High-Level Conceptual Data Models for Database Design

- **Requirements collection and analysis**
  - Database designers interview prospective database users to understand and document data requirements
  - Result: **data requirements**
  - **Functional requirements** of the application

Using High-Level Conceptual Data Models (cont'd.)

- **Conceptual schema**
  - Conceptual design
  - Description of data requirements
  - Includes detailed descriptions of the entity types, relationships, and constraints
  - Transformed from high-level data model into implementation data model

Using High-Level Conceptual Data Models (cont'd.)

- **Logical design** or **data model mapping**
  - Result is a database schema in implementation data model of DBMS
- **Physical design phase**
  - Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified

Entity Types, Entity Sets, Attributes, and Keys

- ER model describes data as:
  - Entities
  - Relationships
  - Attributes

Entities and Attributes

- **Entity**
  - Thing in real world with independent existence
- **Attributes**
  - Particular properties that describe entity
  - Types of attributes:
    - *Composite* versus *simple* (atomic) *attributes*
    - **Single-valued** versus **multivalued** attributes
    - **Stored** versus **derived** attributes
    - **NULL** values
    - **Complex** attributes

Entities and Attributes (cont'd.)

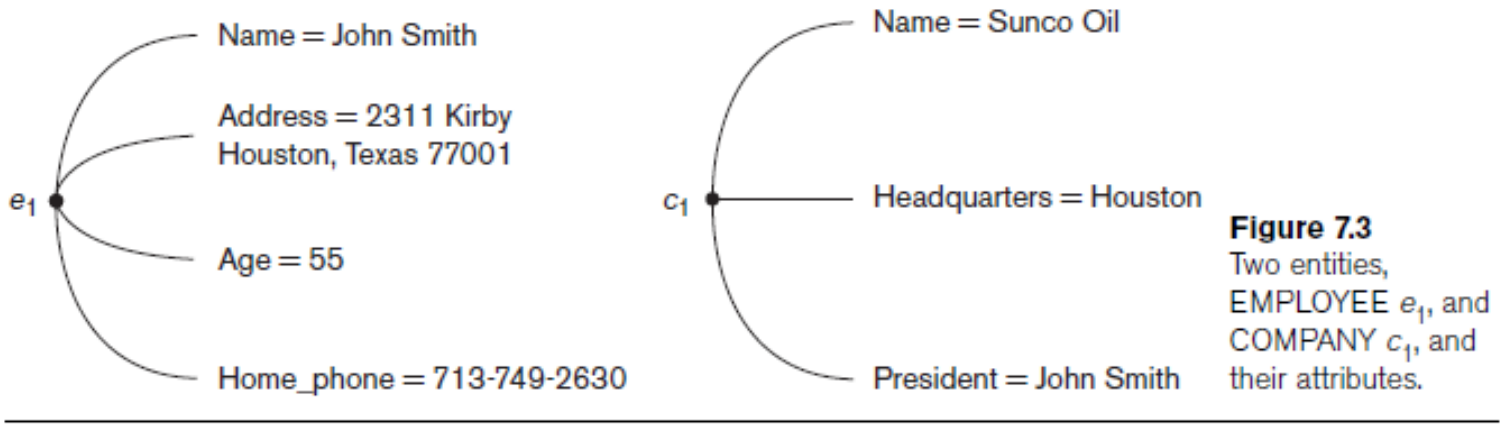


Figure 7.3
Two entities, EMPLOYEE e_1, and COMPANY c_1, and their attributes.

Composite Attributes

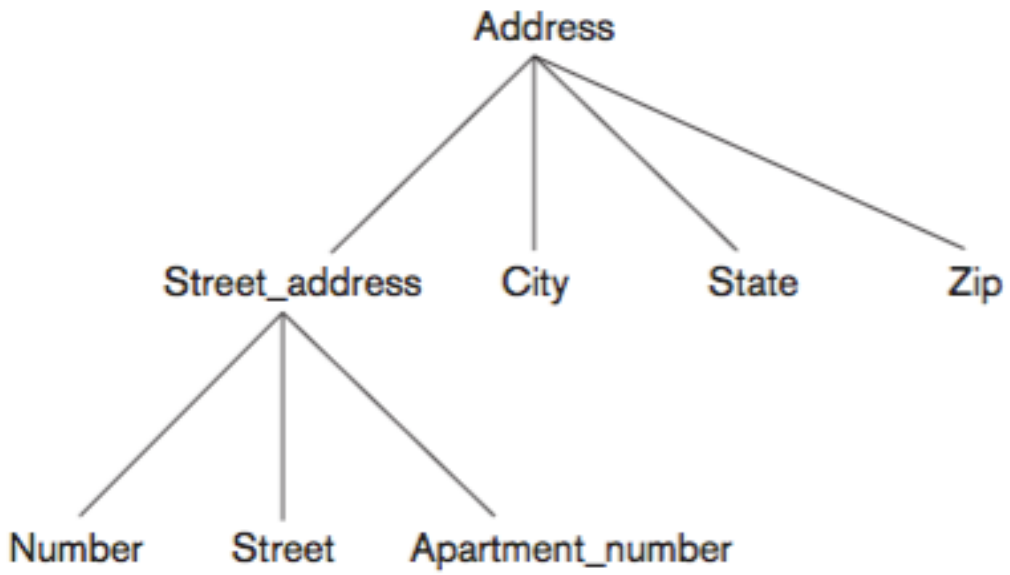


Figure 3.4
A hierarchy of composite attributes.

Single-Valued versus Multivalued Attributes

- Most attributes have a single value for a particular entity; such attributes are called **single-valued**.
  - Age, Height, Birth_date
- In some cases an attribute can have a set of values for the same entity; such attributes are called **multivalued**.
  - Phone_number, College_degree
- A multivalued attribute may have lower and upper bounds to constrain the *number of values* allowed.

Stored versus Derived Attributes

- When two (or more) attribute values are related, sometimes the value of one can be determined from the value of another one.
  - **The derived attribute** is then said to be **derivable from** the **stored attribute**.
    - Age can be derived from Birth_date
  - Some attribute values can be derived from *related entities.*
    - Number_of_employees by counting the number of employees related to (working for) that department.

NULL Values

- In some cases, a particular entity may not have an applicable value for an attribute.
  - Apartment_number attribute of an address applies only to addresses that are in apartment buildings .
  - College_degrees attribute applies only to people with college degrees.
- A special value called NULL is created.
- NULL can also be used if we do not know the value of an attribute for a particular entity

Complex Attributes

- Composite and multivalued attri- butes can be nested arbitrarily.
- Use parentheses ( ) to group components of a composite attribute and separate the components with commas.
- Use braces { } to display multivalued attributes
  - {Address_phone({Phone(Area_code,Phone_number)},Address(Street_address(Number,Street,Apartment_number),City,State,Zip))}

Entity Types, Entity Sets, Keys, and Value Sets

- **Entity type**
  - Collection (or set) of entities that have the same attributes

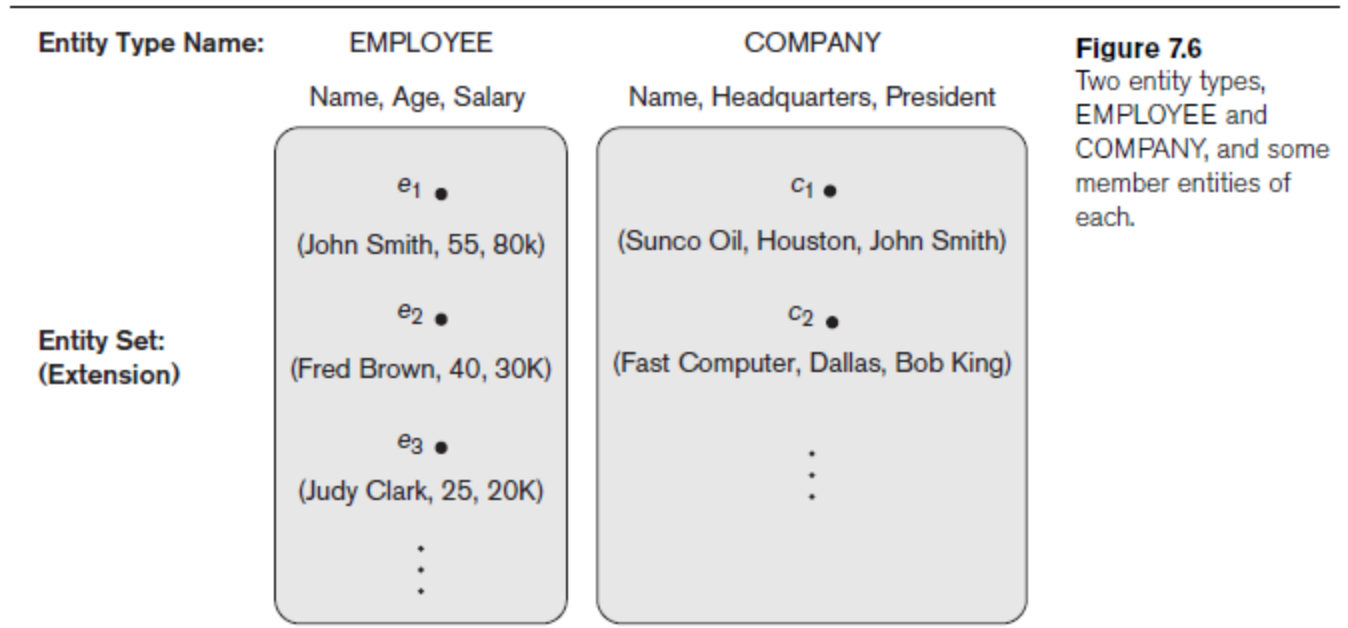
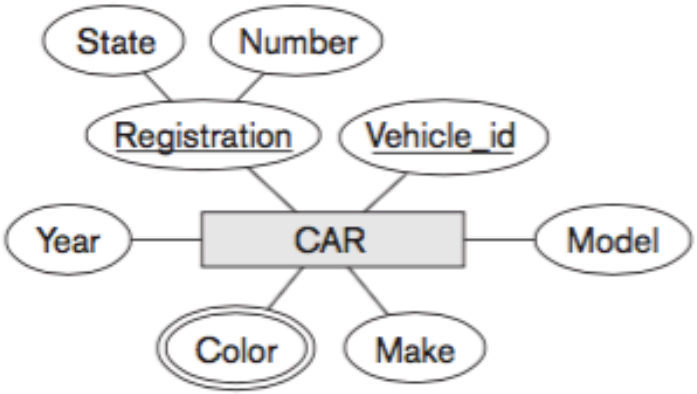


Figure 7.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

Entity Types, Entity Sets, Keys, and Value Sets (cont'd.)

- An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.
- Attribute names are enclosed in ovals and are attached to their entity type by straight lines.
- Composite attributes are attached to their component attributes by straight lines.
- Multivalued attributes are displayed in double ovals.
- Derived attributes are displayed in dashed ovals.

(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.

Key Attributes

- **Key** or **uniqueness constraint**
  - Attributes whose values are distinct for each individual entity in entity set
- **Key attribute**
  - Can be used to identify each entity uniquely
  - Sometimes several attributes together form a key, the *combination* of the attribute values must be distinct for each entity
    - Define a *composite attribute* and designate it as a key attribute of the entity type
    - Composite key must be *minimal*
  - Key attributes have their names **underlined** inside the oval
  - Uniqueness property must hold for every entity set of the entity type
  - Entity types may have *more than one* key attribute (candidate keys)
    - One must be chosen as the primary key when mapping ER model to relational DB

Value Sets

- **Value sets** (or **domain of values**)
  - Specifies set of values that may be assigned to that attribute for each individual entity
    - If the range of ages allowed for employees is between 16 and 70, we can specify the value set of the Age attribute of EMPLOYEE to be the set of integer numbers between 16 and 70
  - Value sets are not typically displayed in basic ER diagrams

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 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, Social Security number,2 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 cur- rent 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.

A Sample Database Application

- COMPANY
  - Employees, departments, and projects
  - Company is organized into departments
  - Department controls a number of projects
  - Employee: store each employee's name, Social Security number, address, salary, sex (gender), and birth date
  - Keep track of the dependents of each employee

Initial Conceptual Design of the COMPANY Database

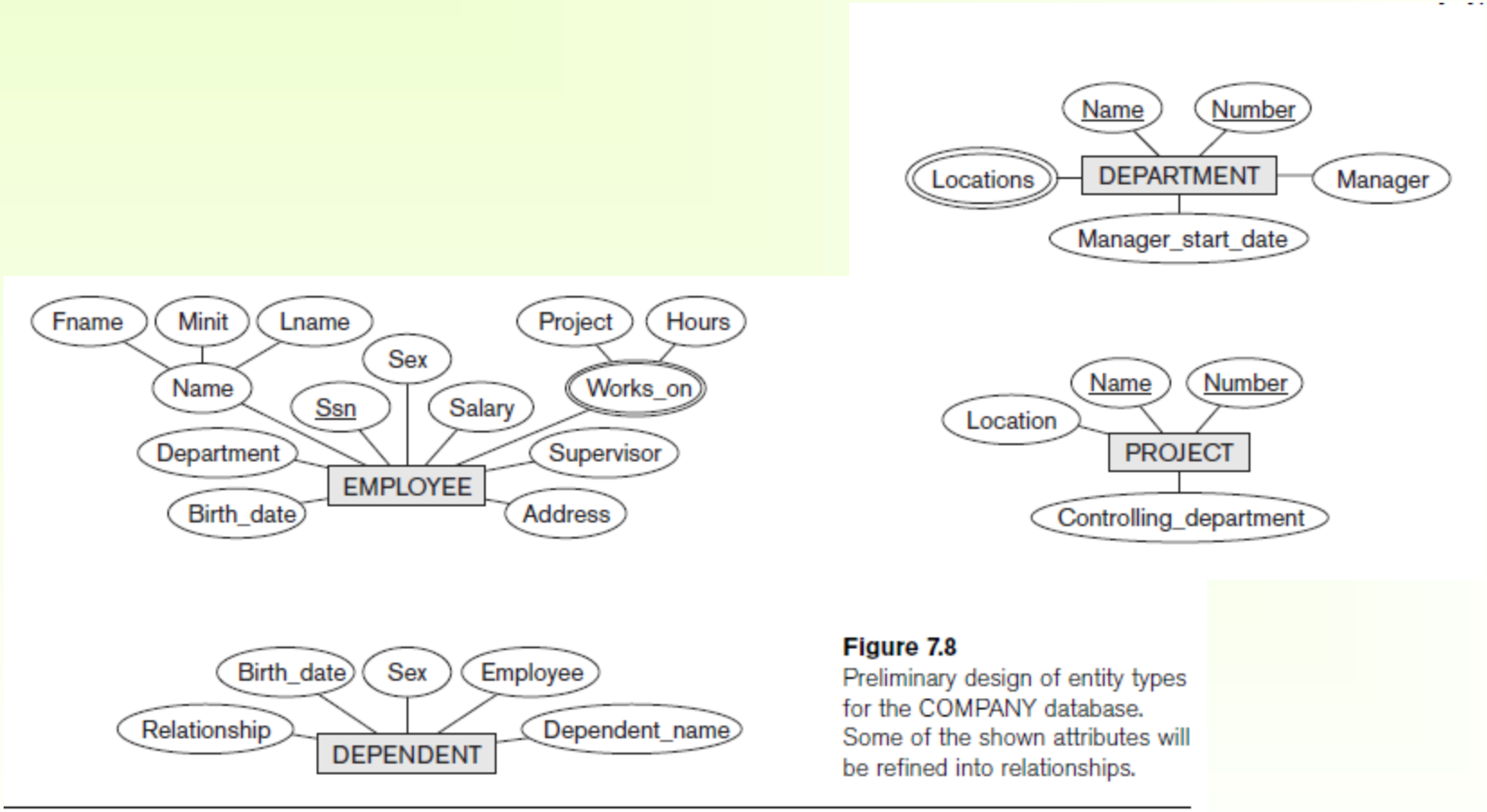


Figure 7.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Relationship Types, Relationship Sets, Roles, and Structural Constraints

- **Relationship**
  - When an attribute of one entity type refers to another entity type
  - Represent references as relationships not attributes

Relationship Types, Sets, and Instances

- **Relationship type** R among n entity types E_1, E_2, ..., E_n
  - Defines a set of associations among entities from these entity types
- **Relationship instances** r_i
  - Each r_i associates n individual entities (e_1, e_2, ..., e_n)
  - Each entity e_j in r_i is a member of entity set E_j
- 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.

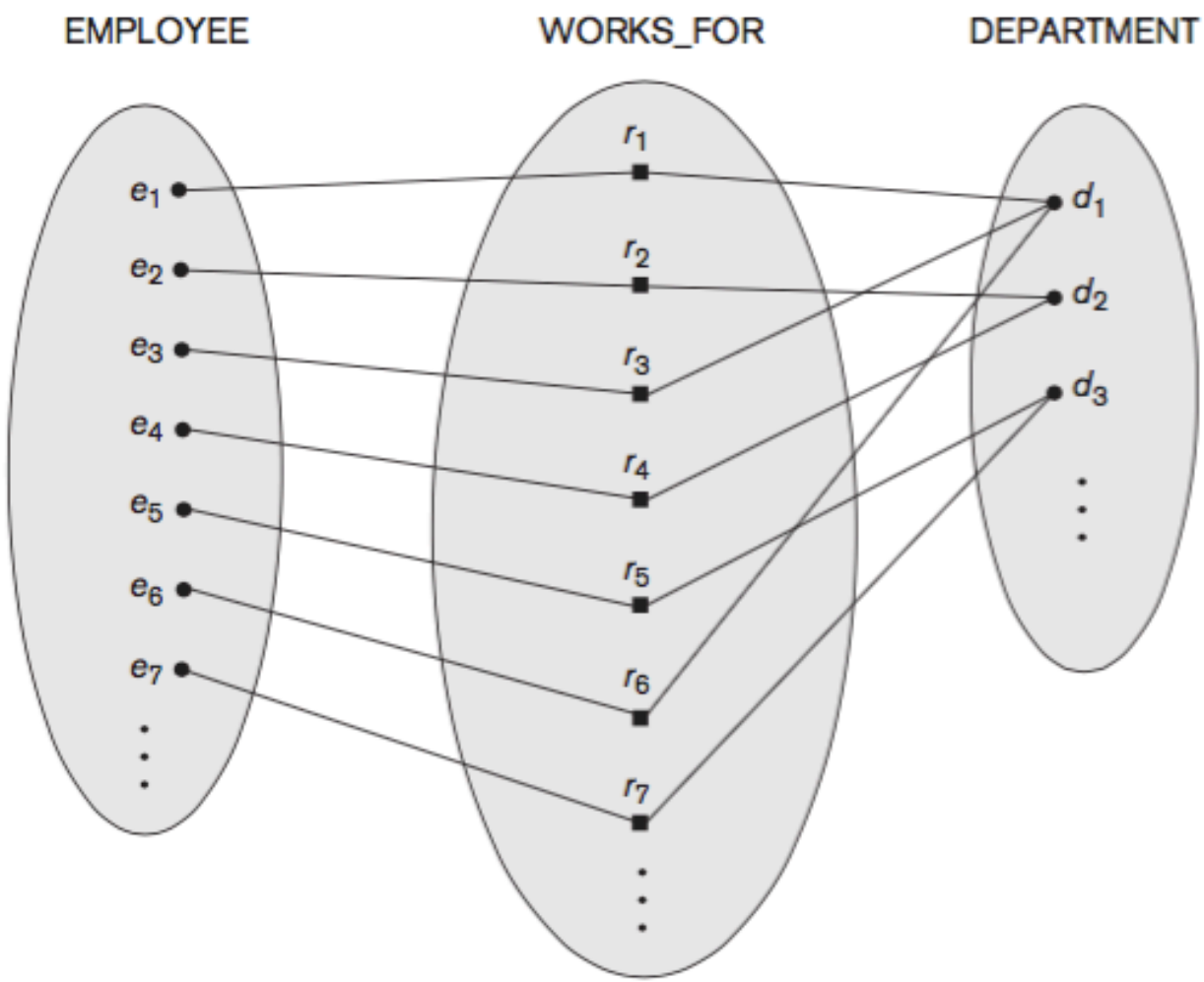


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship Degree

- **Degree** of a relationship type
  - Number of participating entity types
  - **Binary**, **ternary**
- Relationships as attributes
  - Think of a binary relationship type in terms of attributes

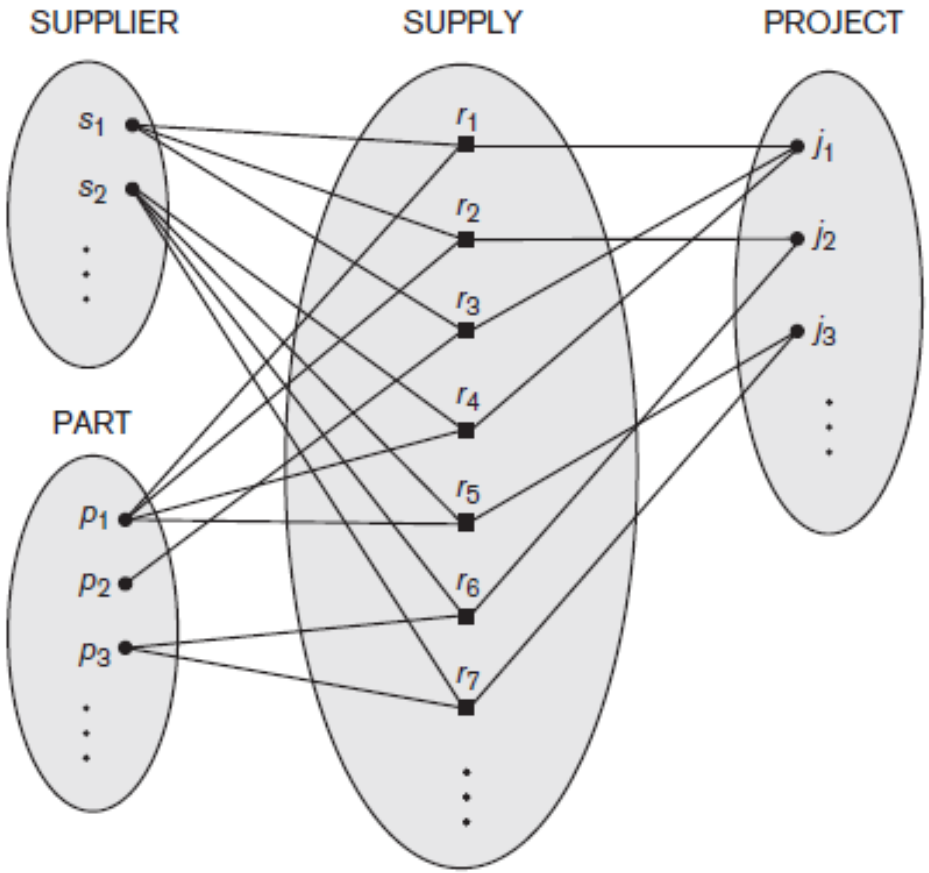


Figure 7.10
Some relationship instances in the SUPPLY ternary relationship set.

Role Names and Recursive Relationships

- **Role names** and recursive relationships
  - Role name signifies role that a participating entity plays in each relationship instance
- **Recursive** relationships
  - Same entity type participates more than once in a relationship type in different roles
  - Must specify role name

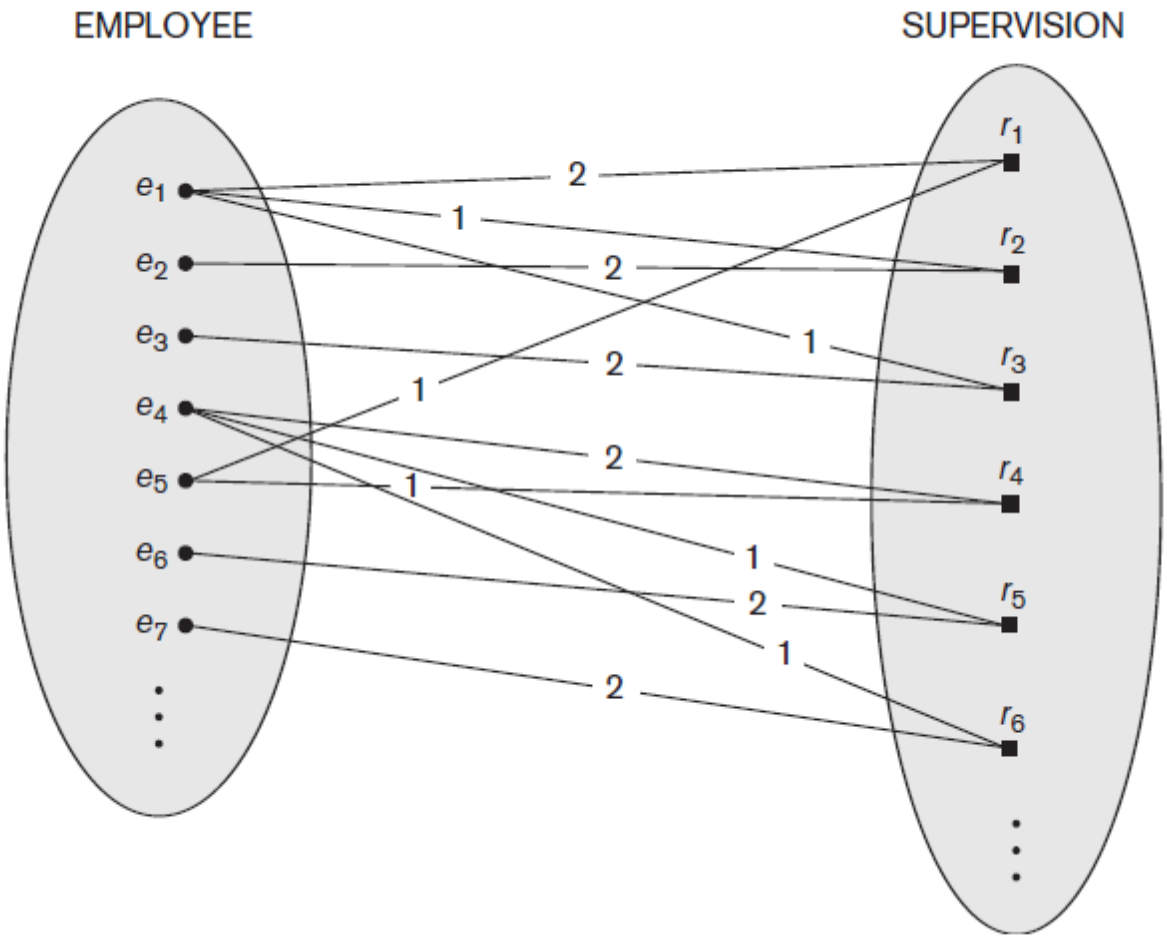
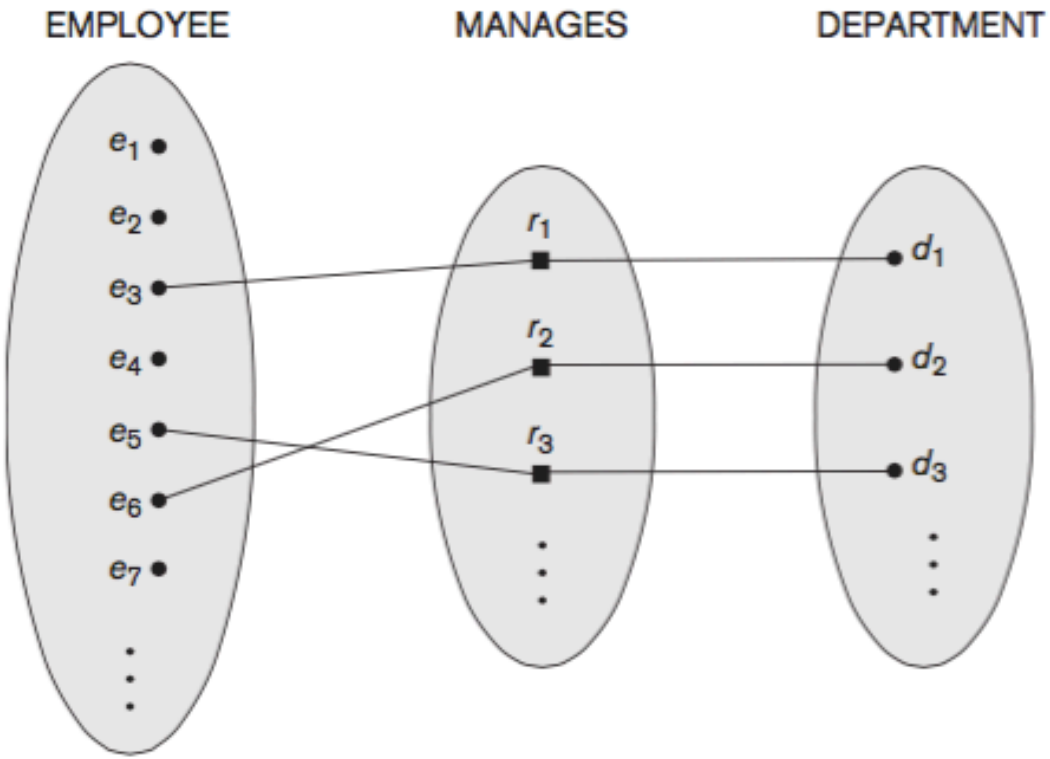


Figure 7.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Constraints on Binary Relationship Types

- **Cardinality ratio** for a binary relationship
  - Specifies maximum number of relationship instances that entity can participate in
  - Cardinality ratios for binary relationships are represented on ER diagrams by displaying 1, M, and N on the diamonds

Figure 3.12
A 1:1 relationship, MANAGES.



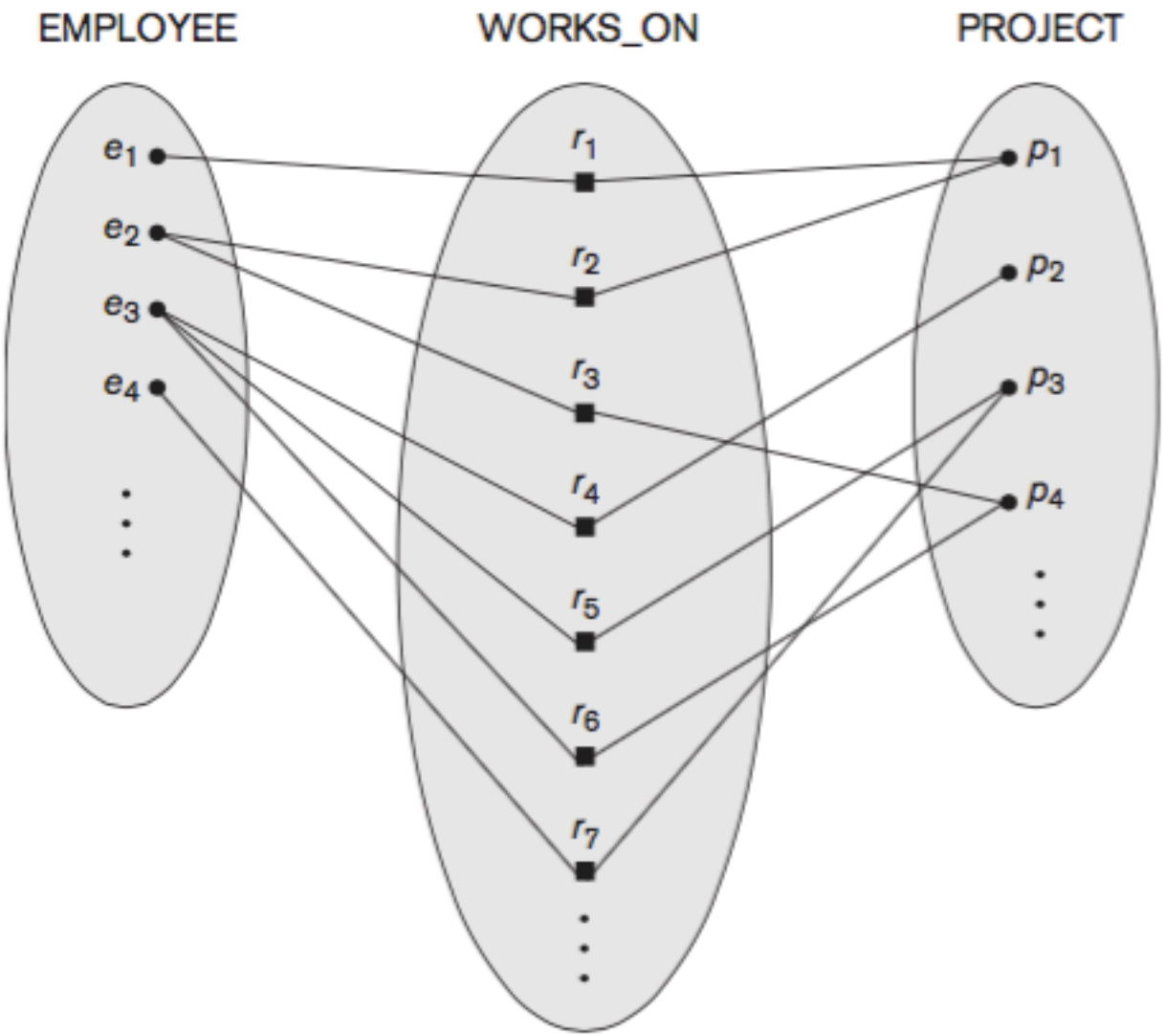


Figure 3.13
An M:N relationship, WORKS_ON.

Constraints on Binary Relationship Types

- **Participation constraint**
  - Specifies whether existence of entity depends on its being related to another entity
  - Types: **total** and **partial**
    - If a company policy states that *every* employee must work for a department, then an employee entity can exist only if it participates in at least one WORKS_FOR relationship instance. Thus, the participation of EMPLOYEE in WORKS_FOR is called **total participation**, meaning that every entity in *the total set* of employee entities must be related to a department entity via WORKS_FOR. Total participation is also called **existence dependency**
    - If we do not expect every employee to manage a department, the participation of EMPLOYEE in the MANAGES relationship type is **partial**, meaning that *some* or *part of the set of* employee entities are related to some department entity via MANAGES, but not necessarily all
  - In ER diagrams, total participation is displayed as a *double line* connecting the participating entity type to the relationship, partial participation is represented by a *single line*

Attributes of Relationship Types

- Relationship types can also have attributes, similar to those of entity types.
- Attributes of 1:1 or 1:N relationship types can be migrated to one entity type
- For a 1:1 relationship type
  - Relationship attribute can be migrated only either entity type
- For a 1:N relationship type
  - Relationship attribute can be migrated only to entity type on N-side of relationship
- For M:N relationship types
  - Some attributes may be determined by combination of participating entities
  - Must be specified as relationship attributes

Weak Entity Types

- Do not have key attributes of their own
  - Identified by being related to specific entities from another entity type
  - Entity type DEPENDENT, related to EMPLOYEE, the attributes of DEPENDENT are Name, Birth_date, Sex, and Relationship (to the employee). Two dependents of *two distinct employees* may have the same values for Name, Birth_date, Sex, and Relationship, but they are still distinct entities. They are identified as distinct entities only after determining the *particular employee entity* to which each dependent is related. Each employee entity is said to *own* the dependent entities that are related to it
- **Identifying relationship**
  - Relates a weak entity type to its owner
- Always has a total participation constraint
- A weak entity type normally has a **partial key**, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity*
- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines
- The partial key attribute is underlined with a dashed or dotted line

Refining the ER Design for the COMPANY Database

- Change attributes that represent relationships into relationship types
- Determine cardinality ratio and participation constraint of each relationship type

Refining the ER Design for the COMPANY Database

- MANAGES, which is a 1:1(one-to-one) relationship type between EMPLOYEE and DEPARTMENT. EMPLOYEE participation is partial. DEPARTMENT participation is not clear from the requirements. We question the users, who say that a department must have a manager at all times, which implies total participation.13 The attribute Start_date is assigned to this relationship type.
- WORKS_FOR, a 1:N (one-to-many) relationship type between DEPARTMENT and EMPLOYEE. Both participations are total.
- CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT. The participation of PROJECT is total, whereas that of DEPARTMENT is deter- mined to be partial, after consultation with the users indicates that some departments may control no projects.
- SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervi- sor role) and EMPLOYEE (in the supervisee role). Both participations are determined to be partial, after the users indicate that not every employee is a supervisor and not every employee has a supervisor.
- WORKS_ON, determined to be an M:N (many-to-many) relationship type with attribute Hours, after the users indicate that a project can have several employees working on it. Both participations are determined to be total.
- DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT. The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

ER Diagrams, Naming Conventions, and Design Issues

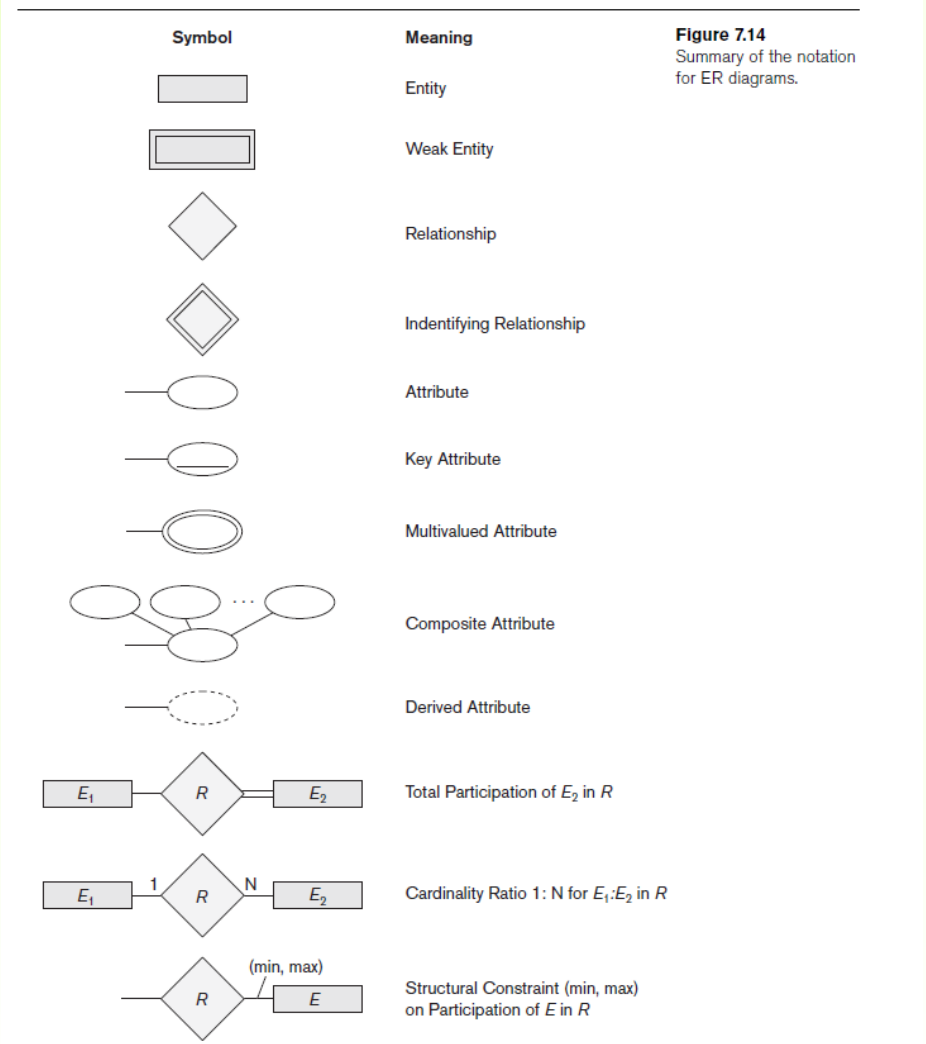


Figure 7.14
Summary of the notation for ER diagrams.

Proper Naming of Schema Constructs

- Choose names that convey meanings attached to different constructs in schema
- Nouns give rise to entity type names
- Verbs indicate names of relationship types
- Choose binary relationship names to make ER diagram readable from left to right and from top to bottom

Design Choices for ER Conceptual Design

- Model concept first as an attribute
  - Refined into a relationship if attribute is a reference to another entity type
- Attribute that exists in several entity types may be elevated to an independent entity type
  - Can also be applied in the inverse

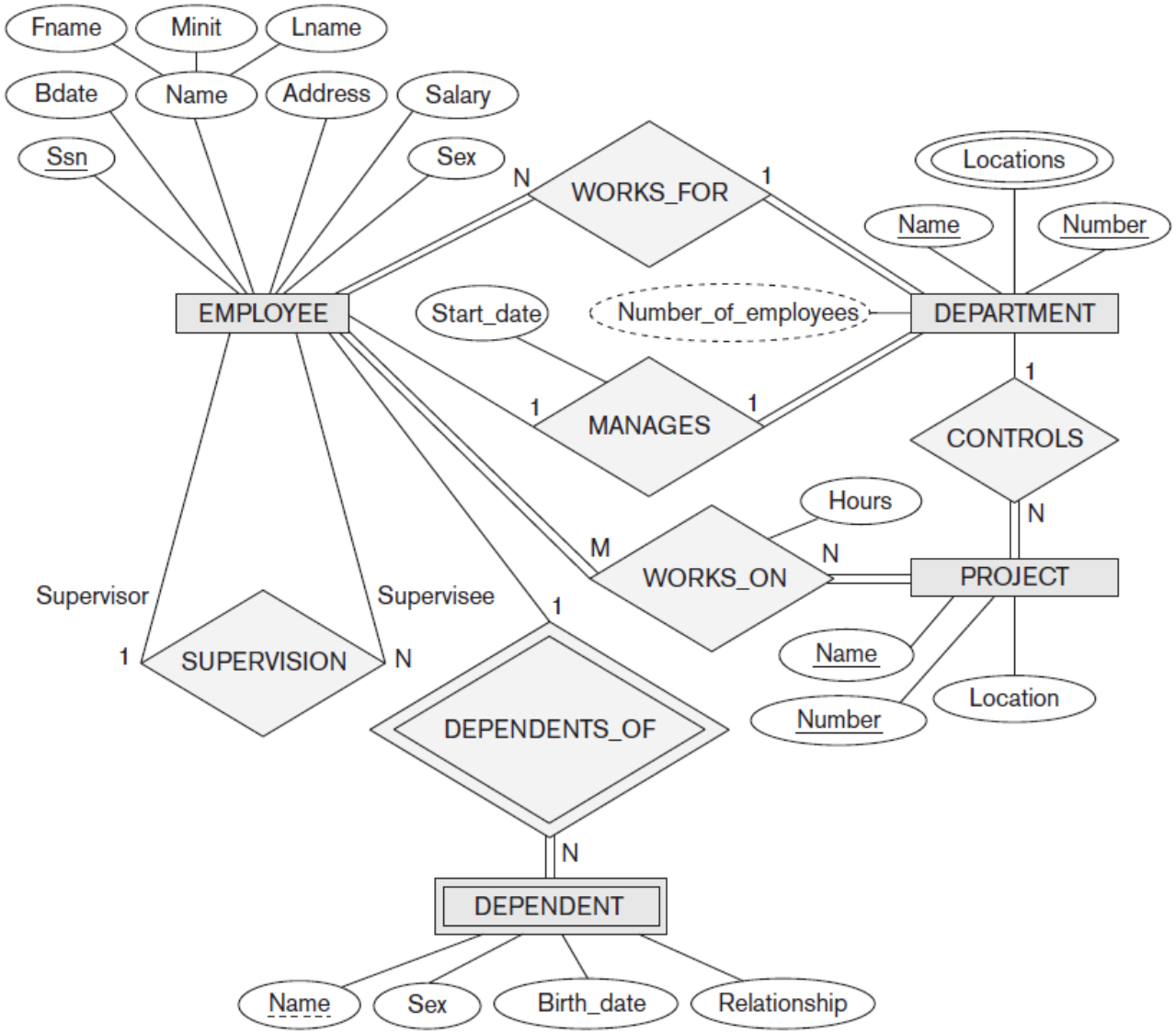


Figure 7.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter and is summarized in Figure 7.14.

Alternative Notations for ER Diagrams

- Specify structural constraints on relationships
  - Replaces cardinality ratio (1:1, 1:N, M:N) and single/double line notation for participation constraints
  - Associate a pair of integer numbers (min, max) with each participation of an entity type *E* in a relationship type *R*, where $0 \leq \text{min} \leq \text{max}$ and $\text{max} \geq 1$

Relationship Types of Degree Higher than Two

- **Degree** of a relationship type
  - Number of participating entity types
- *Binary*
  - Relationship type of degree two
- *Ternary*
  - Relationship type of degree three

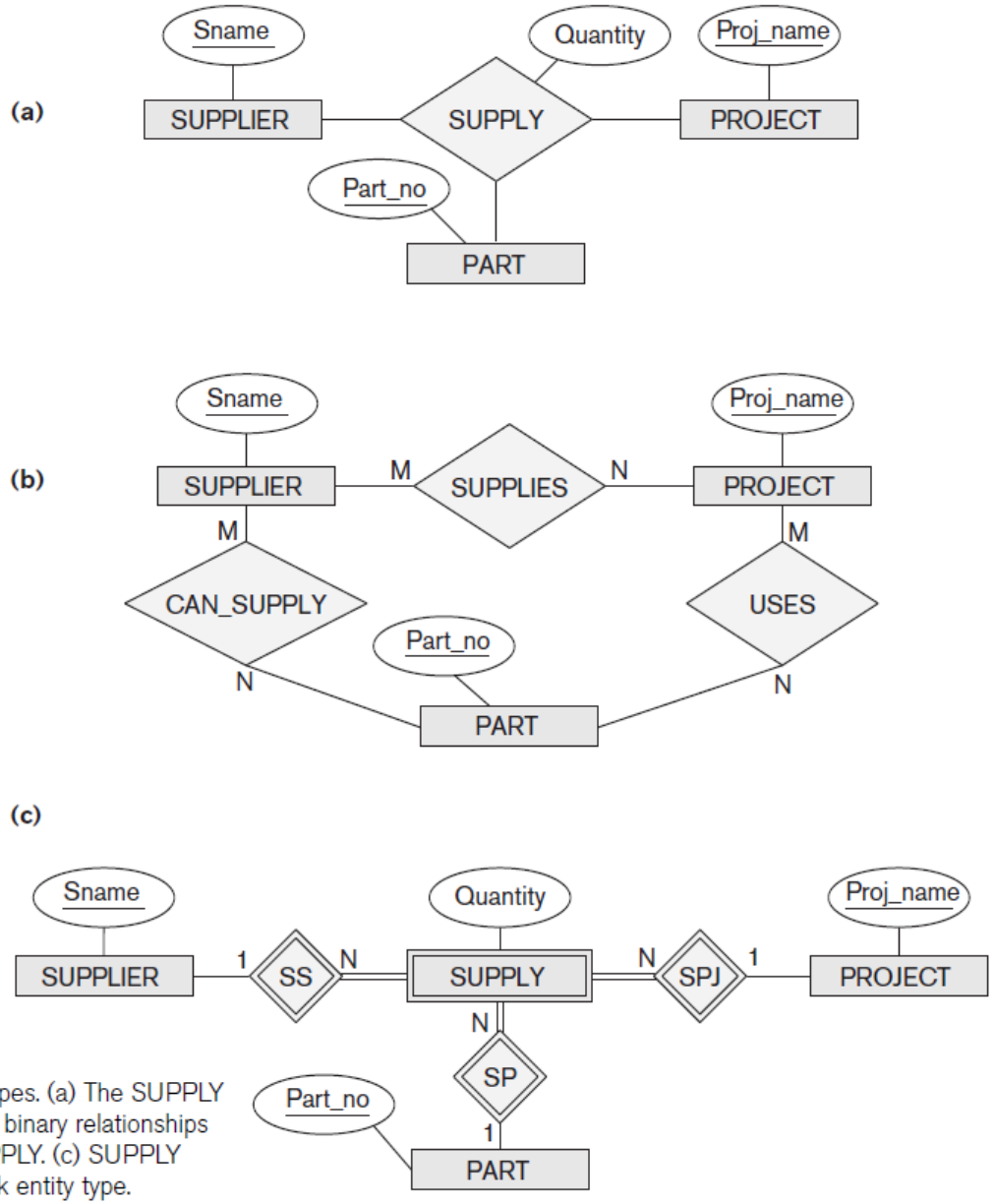
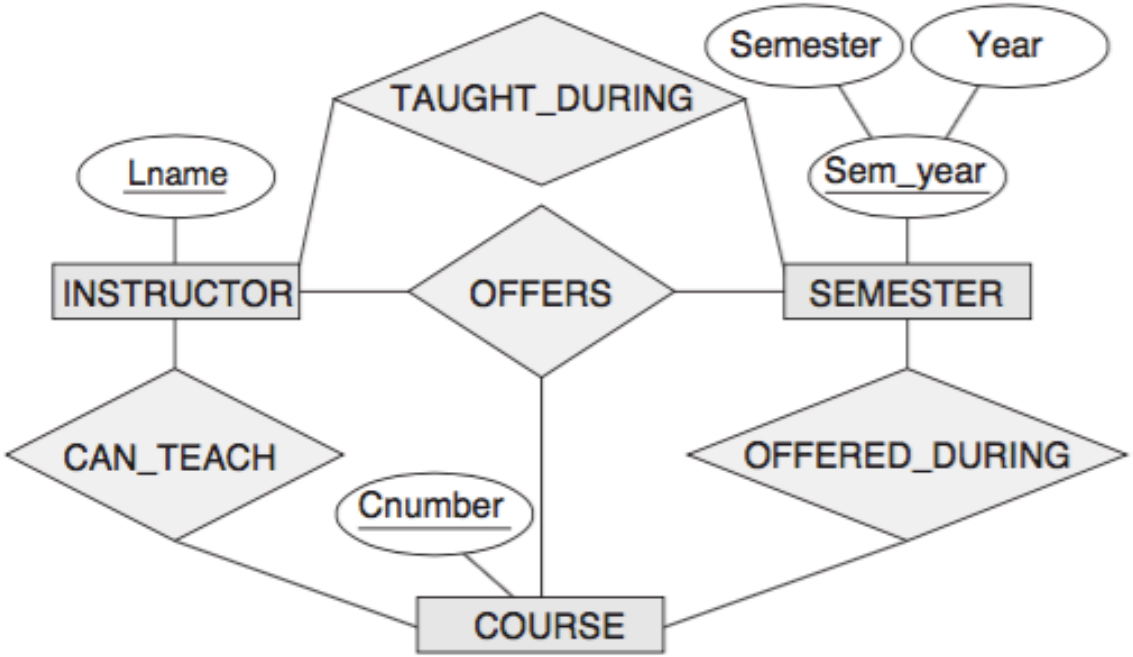


Figure 7.17
Ternary relationship types. (a) The SUPPLY relationship. (b) Three binary relationships not equivalent to SUPPLY. (c) SUPPLY represented as a weak entity type.

Figure 3.18
Another example of ternary versus binary relationship types.



Choosing between Binary and Ternary (or Higher-Degree) Relationships

- Some database design tools permit only binary relationships
  - Ternary relationship must be represented as a weak entity type
  - No partial key and three identifying relationships
- Represent ternary relationship as a regular entity type
  - By introducing an artificial or surrogate key

Constraints on Ternary (or Higher-Degree) Relationships

- Notations for specifying structural constraints on *n*-ary relationships
  - Should both be used if it is important to fully specify structural constraints

Example of Other Notation: UML Class Diagrams

- UML methodology
  - Used extensively in software design
  - Many types of diagrams for various software design purposes
- UML class diagrams
  - Entity in ER corresponds to an object in UML

Example of Other Notation: UML Class Diagrams (cont'd.)

- **Class** includes three sections:
  - Top section gives the class name
  - Middle section includes the attributes;
  - Last section includes operations that can be applied to individual objects

Example of Other Notation: UML Class Diagrams (cont'd.)

- **Associations**: relationship types
- **Relationship instances**: links
- Binary association
  - Represented as a line connecting participating classes
  - May optionally have a name
- Link attribute
  - Placed in a box connected to the association's line by a dashed line

Example of Other Notation: UML Class Diagrams (cont'd.)

- **Multiplicities**: min..max, asterisk (*) indicates no maximum limit on participation
- Types of relationships: **association** and **aggregation**
- Distinguish between **unidirectional** and **bidirectional** associations
- Model weak entities using **qualified association**

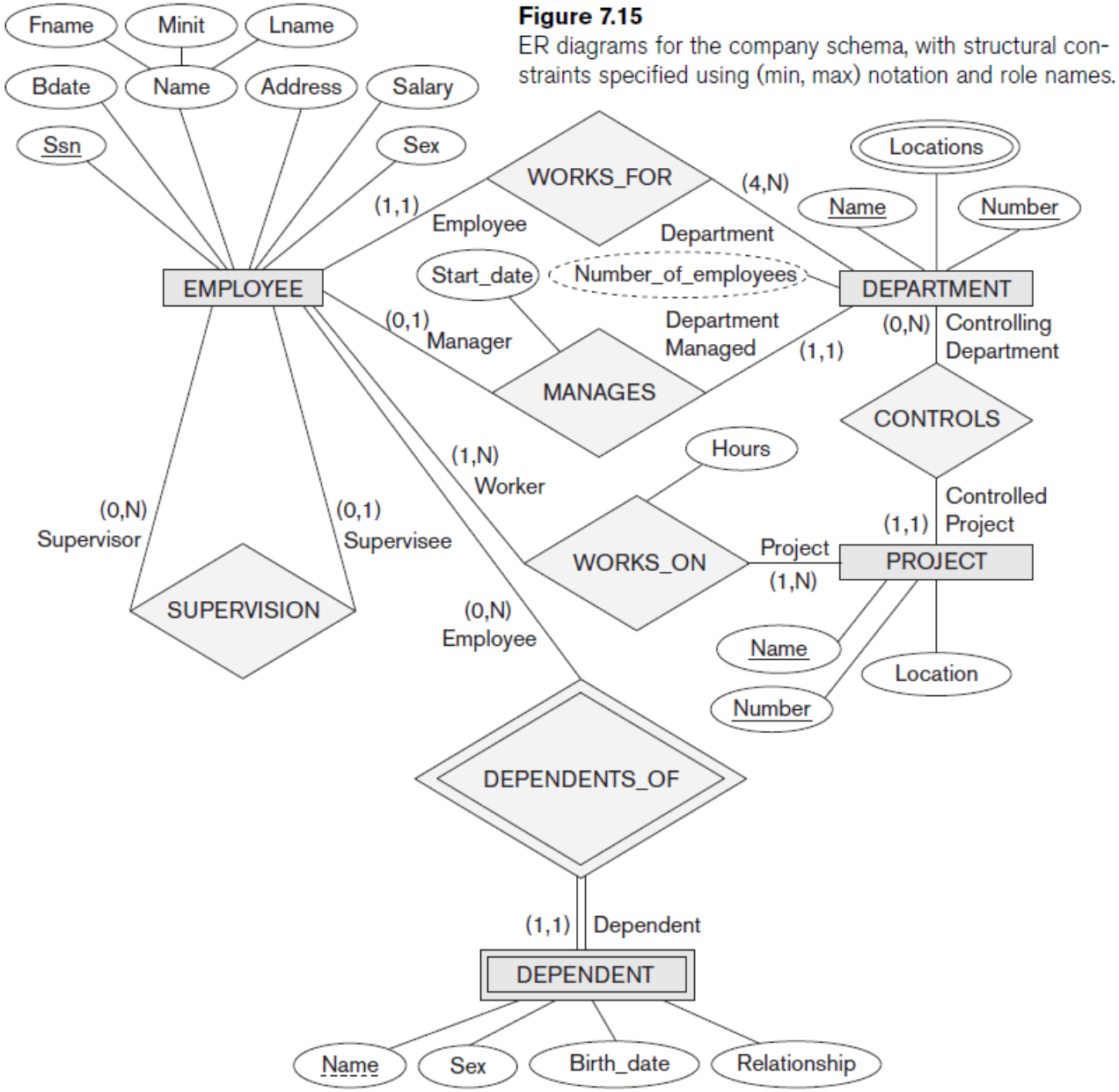
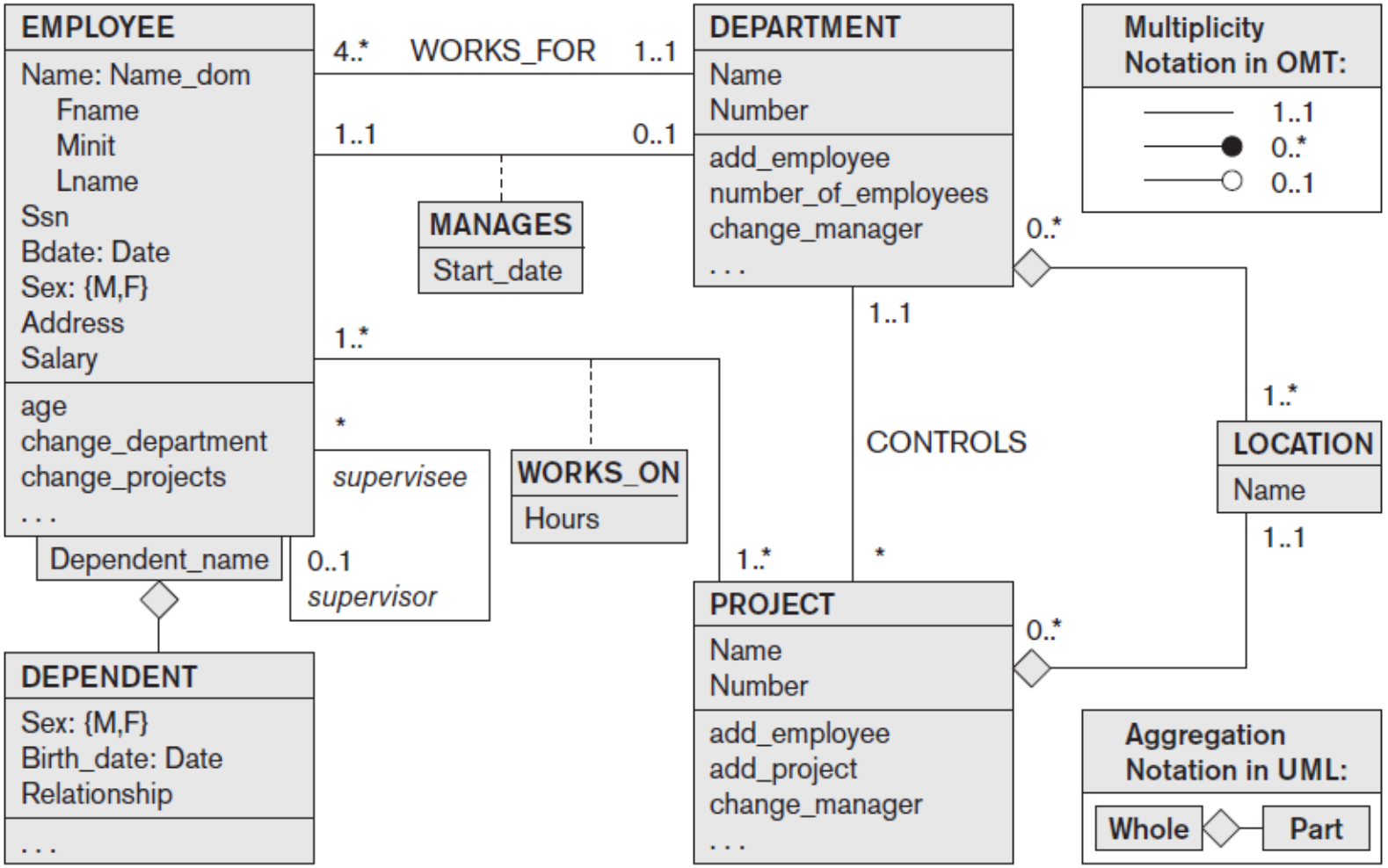


Figure 7.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

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



Summary

- Basic ER model concepts of entities and their attributes
  - Different types of attributes
  - Structural constraints on relationships
- ER diagrams represent E-R schemas
- UML class diagrams relate to ER modeling concepts