DATA MODELING

Aims:

> At the end of this group of four lectures you should be able to understand the concepts and importance of data modeling in general. You should also be able to design database schemas according to specified requirements using the ER and EER data models.

Reading: Elmasri & Navathe, Chapters 3 & 4

OVERVIEW

1. Phases of database design
2. Entity-Relationship Model (ER)
3. Enhanced Entity-Relationship Model (EER)
4. Why model?
5. Data modeling tools

PHASES OF DATABASE DESIGN

- Requirements collection and analysis
- Conceptual design
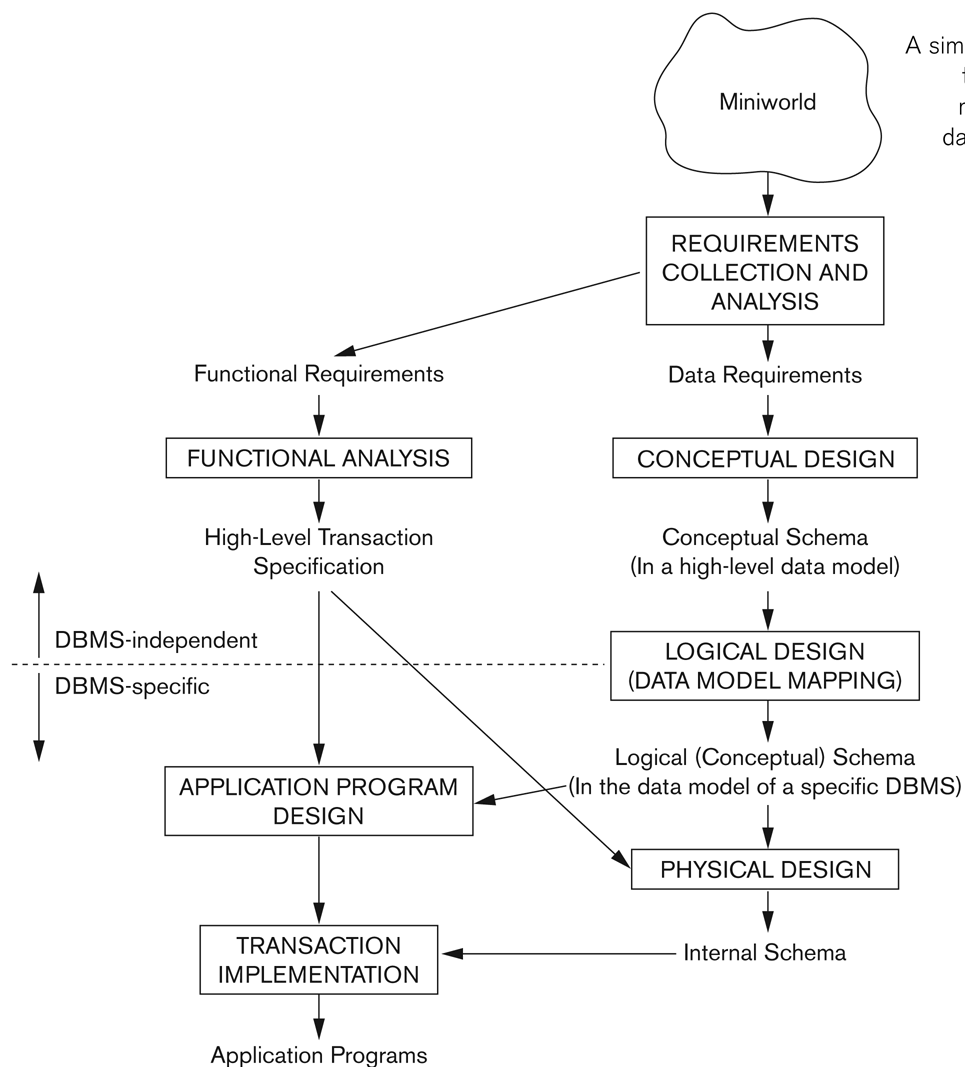- Logical design
- Physical design

PHASES OF DATABASE DESIGN

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

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

METHODOLOGIES FOR CONCEPTUAL DESIGN

- Entity Relationship (ER) Diagrams (Chapter 3)
- Enhanced Entity Relationship (EER) Diagrams (Chapter 4)
- Use of Design Tools in industry for designing and documenting large scale designs
- The UML (Unified Modeling Language) Class Diagrams are popular in industry to document conceptual database designs

OVERVIEW

1. **Phases of database design**
2. Entity-Relationship Model (ER)
   - **Structural component**
     - Entities and attributes
     - Relationships
     - Weak entities
   - **Integrity component**
     - Key attributes
     - Cardinality ratio
     - Participation constraint
   - **Graphical notation and naming conventions**
3. Enhanced Entity-Relationship Model (EER)
4. Why model?
5. Data modeling tools

REQUIREMENTS FOR 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.

- Each department controls a number of projects. Each project has a name, number and is located at a single location.

REQUIREMENTS FOR THE COMPANY DATABASE

- We store each employee's social security number, 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.

- Each employee may have a number of dependents. For each dependent, we keep track of their name, sex, birthdate, and relationship to employee.

STRUCTURAL COMPONENT OF ER

- **Entities and Attributes**
  - **Entities** are specific objects or things in the mini-world that are represented in the database
  - **Attributes** are properties used to describe an entity
  - A specific entity will have a value for each of its attributes. For example: Employee
    - Name='John Smith'
    - SSN='123456789'
    - Address ='731, Fondren, Houston, TX'
    - Gender='M'
    - BirthDate='09-JAN-85'
  - Each attribute has a ***domain*** (value set)

Types of Attributes

- **Simple (atomic) or Composite**

  Gender is a simple attribute

  Name (FirstName, MiddleName, LastName)

  Composition may form a hierarchy where some components are themselves composite.

- **Single/Multi-valued**

  PreviousDegrees of a STUDENT

  Denoted as {PreviousDegrees}

- **Stored or Derived**

EXAMPLE OF A COMPOSITE ATTRIBUTE
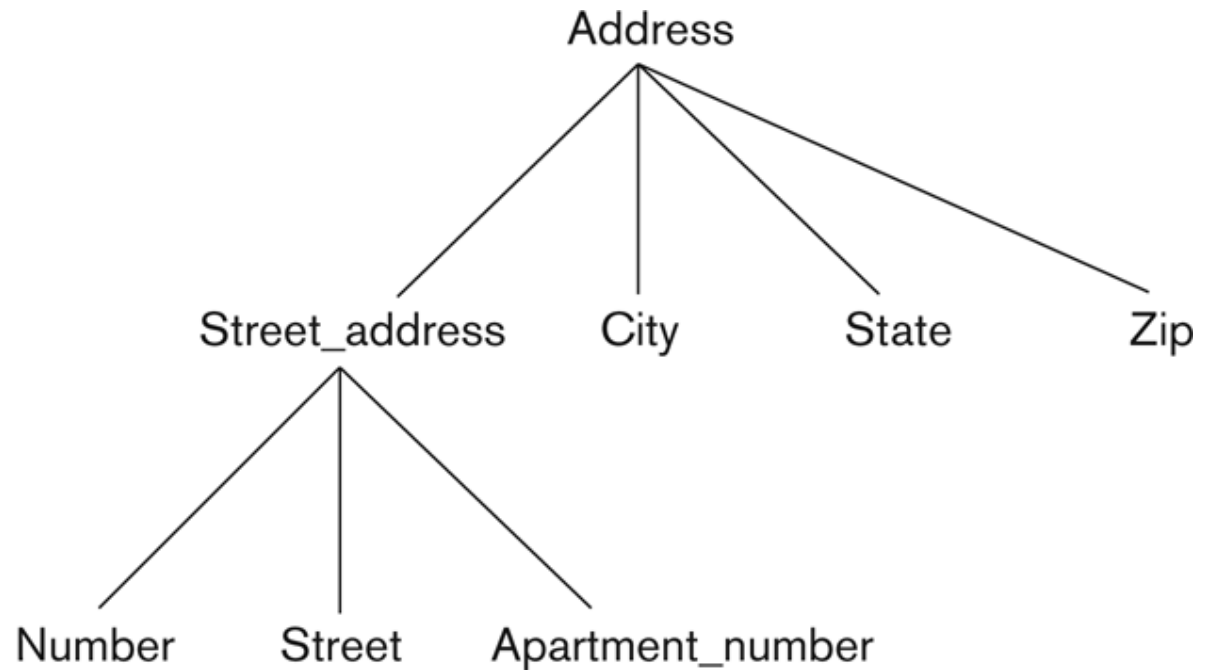
Figure 3.4
A hierarchy of composite attributes.

TYPES OF ATTRIBUTES (CONT.)

Composite and multi-valued attributes may be nested.

Degrees of a STUDENT is a composite multi-valued attribute denoted by:

{Degrees (University, Year, Degree, Field)}

THE NULL VALUE

Three interpretations

- Not known
- Not applicable
- Missing

ENTITY TYPES AND KEY ATTRIBUTES

- **Entity type**: entities with the same attributes
- An attribute of an entity type for which each entity must have a unique value is called a **key attribute**
- A key attribute may be composite

  CourseCode is a key of the COURSE entity type with components (DisciplineCode, CourseNumber).

- An entity type may have more than one key

  EMPLOYEE entity type may have two keys:
  - EmployeeNo
  - IRD_No

Slide
2-14

ENTITY TYPE CAR WITH TWO KEYS AND A CORRESPONDING ENTITY SET

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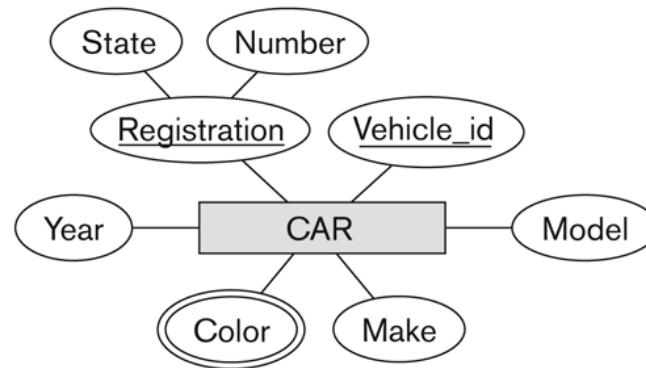
Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

State   Number

Registration   Vehicle_id

Year   CAR   Model

Color   Make

(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 types for the COMPANY database

- EMPLOYEE
- DEPARTMENT
- PROJECT
- DEPENDENT

INITIAL DESIGN OF ENTITY TYPES:
EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT
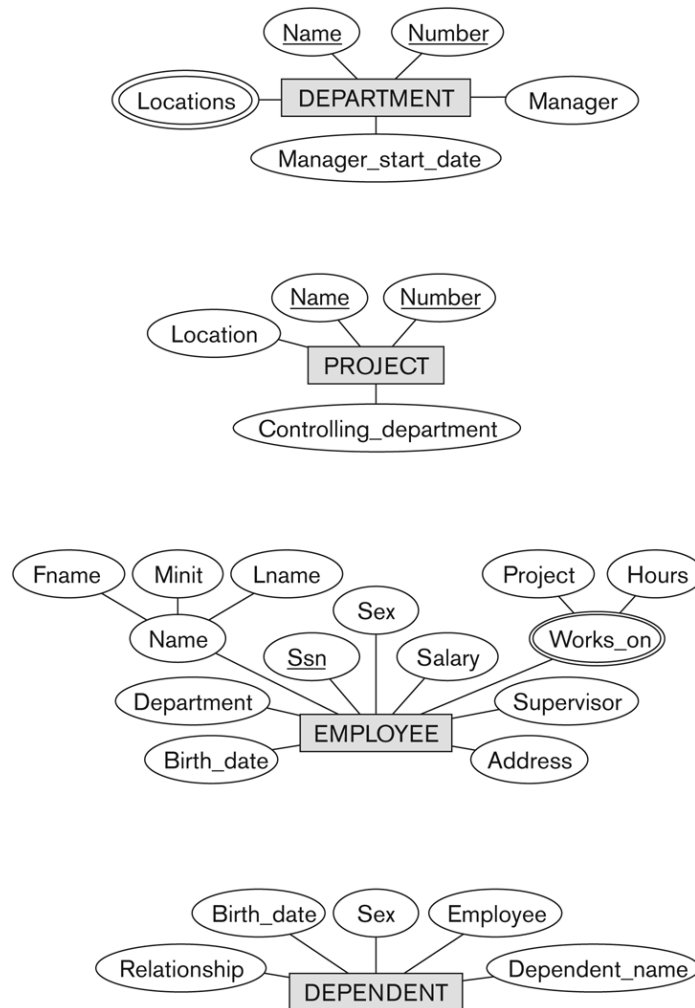
Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

RELATIONSHIPS

- A relationship relates two or more distinct entities with a specific meaning

  John Smith works on ProductX

  Franklin Wong manages the Research department

- Relationships of the same type are grouped or typed into a relationship type

  WORKS_ON: EMPLOYEE and PROJECT participate

  MANAGES: EMPLOYEE and DEPARTMENT participate

- The d**egree of a relationship type** is the number of participating entity types

RELATIONSHIP INSTANCES

EMPLOYEE   WORKS_FOR   DEPARTMENT
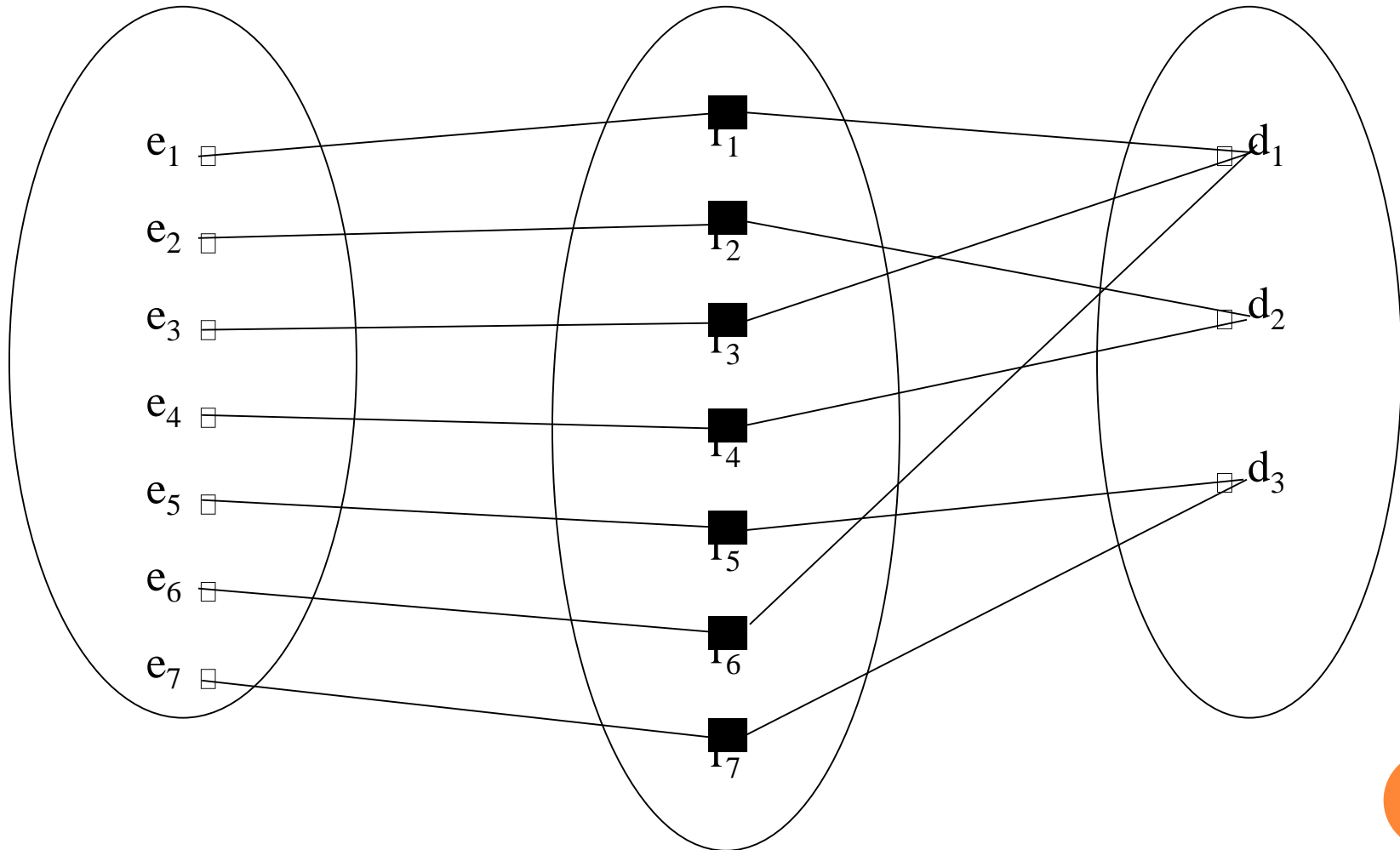
EXAMPLE RELATIONSHIP INSTANCES OF WORKS_ON
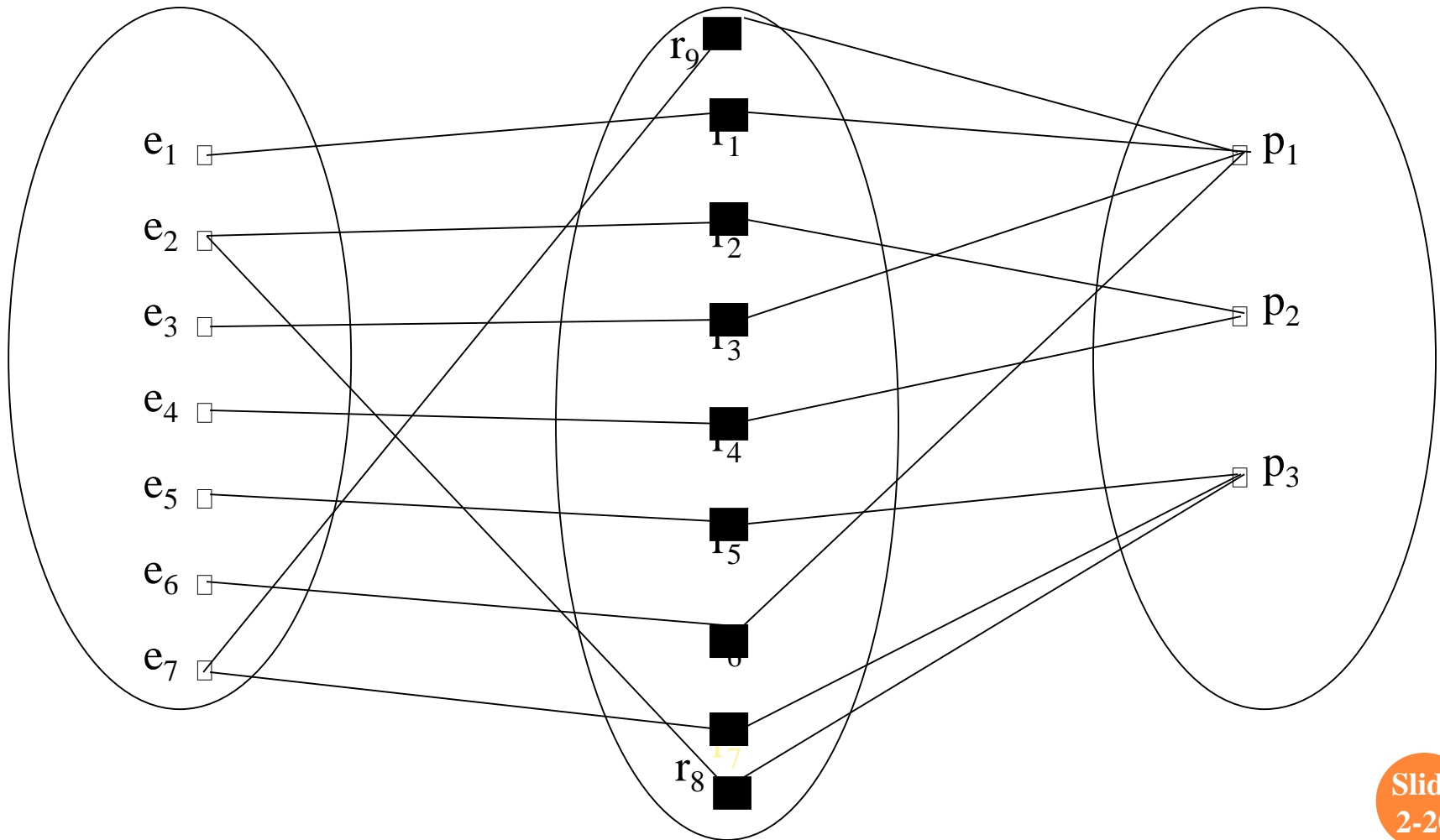
RELATIONSHIP TYPE VS. RELATIONSHIP SET

- 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

Slide 2-21

RELATIONSHIP TYPES (CONT.)

- More than one relationship type can exist with the same participating entity types.

- MANAGES and WORKS_FOR are distinct relationships between EMPLOYEE and DEPARTMENT, but with different meanings and different relationship instances.

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.

Relationships Types for the COMPANY database

- WORKS_FOR
- MANAGES
- CONTROLS
- WORKS_ON
- SUPERVISION
- DEPENDENTS_OF

INCOMPLETE COMPANY SCHEMA (1)
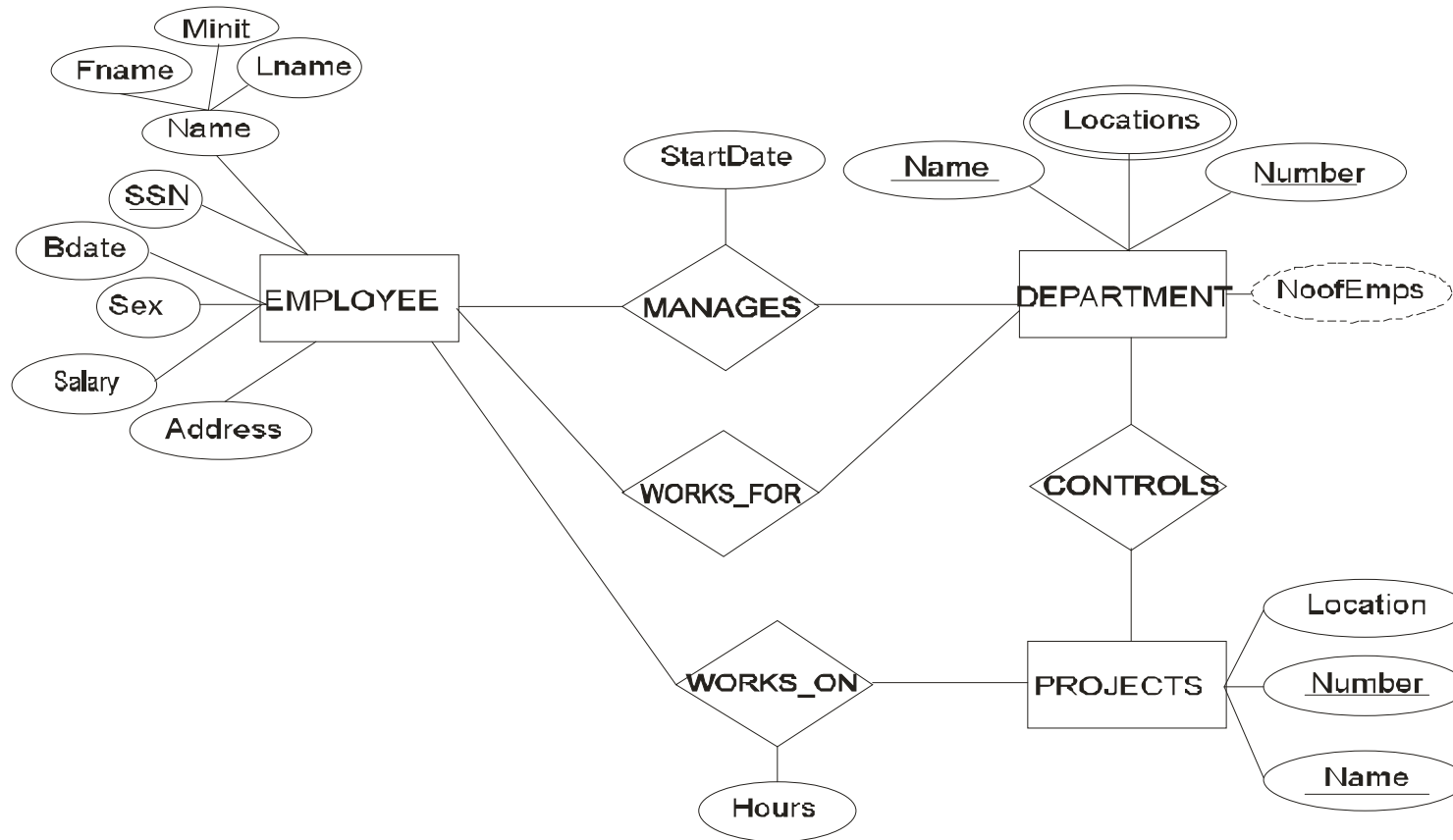
RECURSIVE RELATIONSHIP TYPES

o Both participations are same entity type in different roles

o For example, SUPERVISION relationships between EMPLOYEE (in role of supervisor or boss) and (another) EMPLOYEE (in role of subordinate or worker)

o In following figure, first role participation labeled with 1 and second role participation labeled with 2

o In ER diagram, need to display role names to distinguish participations.

A RECURSIVE RELATIONSHIP

EMPLOYEE   SUPERVISION

Slide
2-27
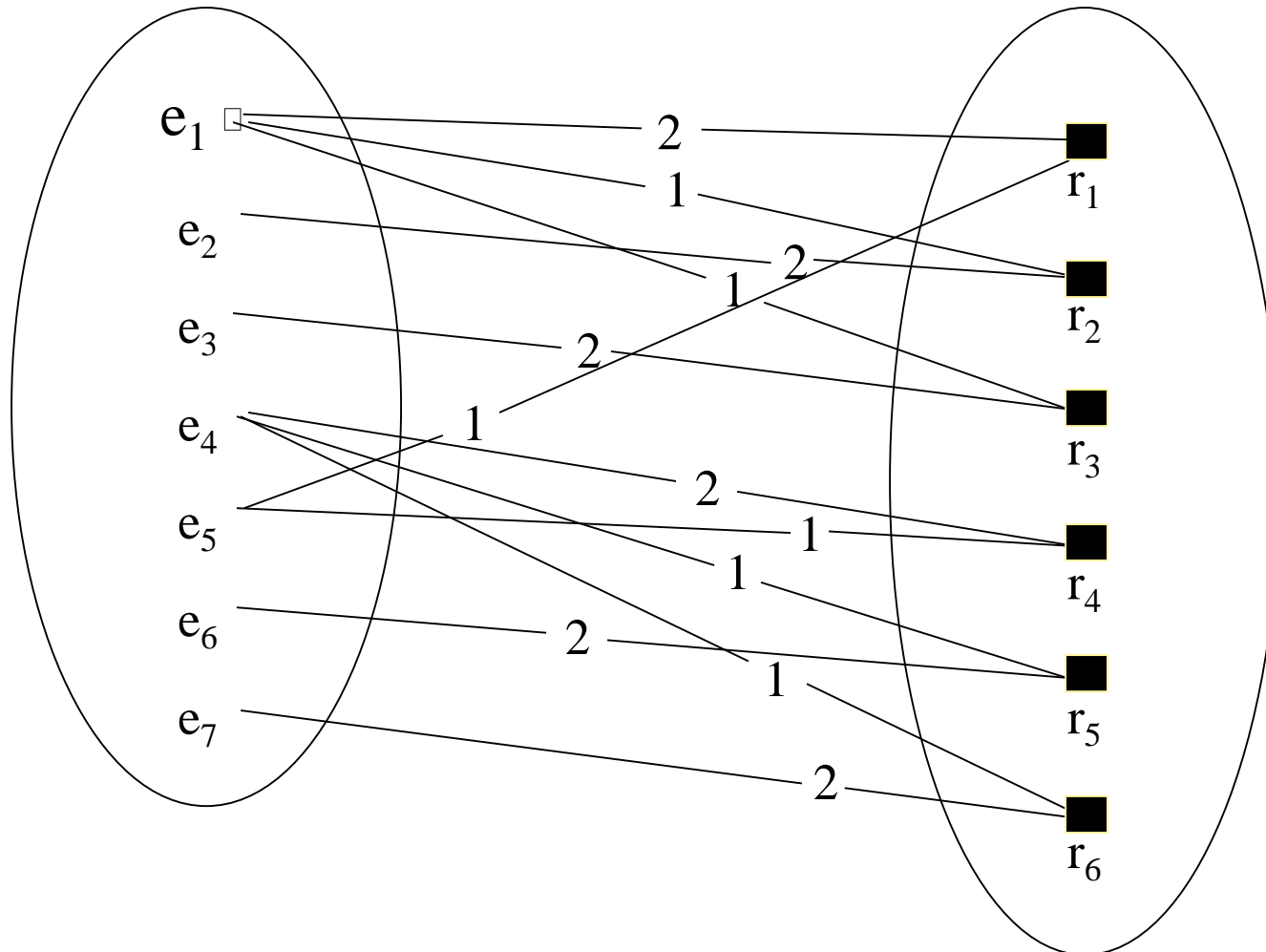
INCOMPLETE COMPANY SCHEMA (2)
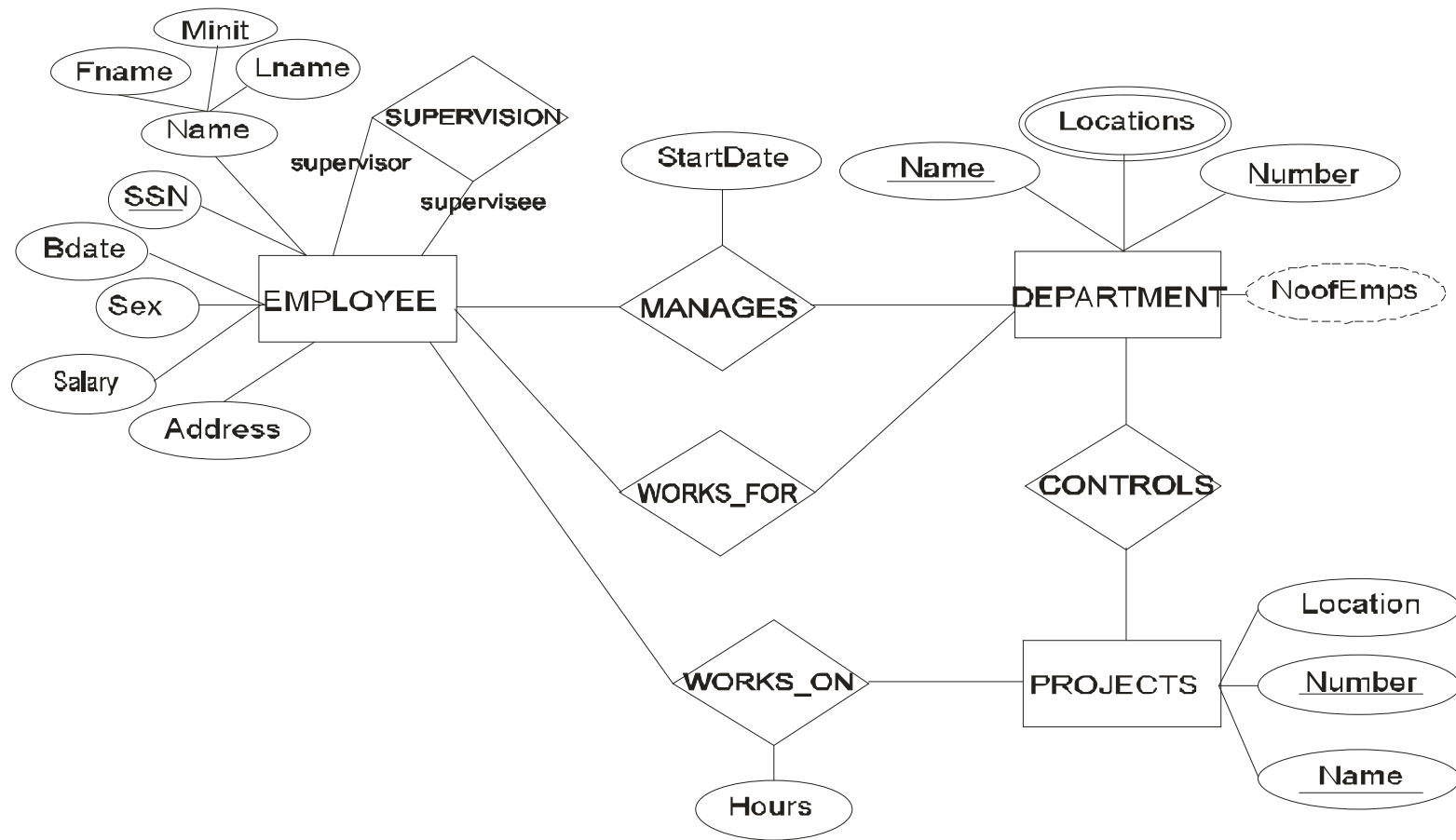
Weak Entity Type

- An entity type that does not have a key attribute
- A weak entity must participate in an **identifying relationship** type with an **owner** or identifying entity typeEntities 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:

Suppose that a DEPENDENT entity is identified by the dependent's first name *and* the specific EMPLOYEE that the dependent is related to.

DEPENDENT
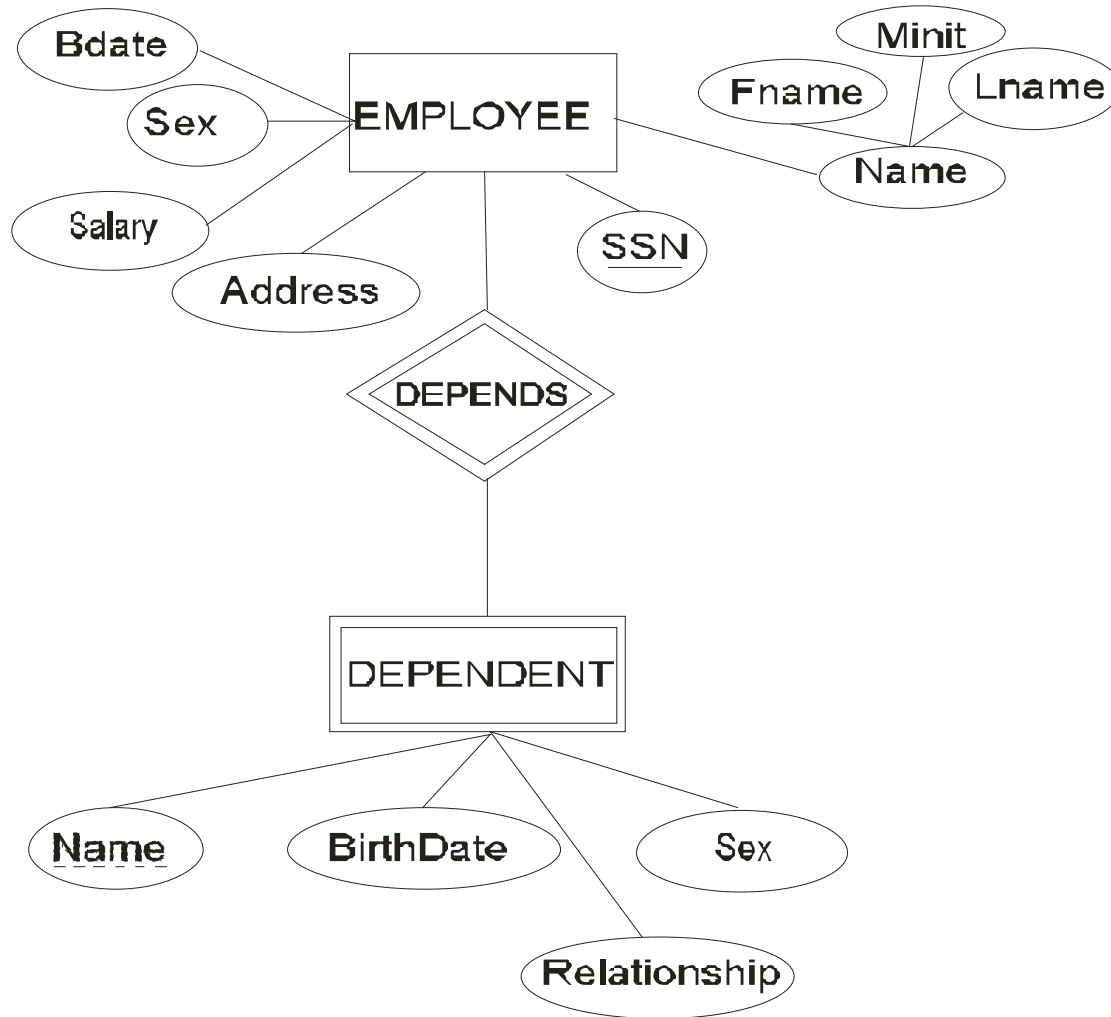
INTEGRITY COMPONENT OF ER

- Key attribute

- Cardinality ratio

- Participation constraint

CARDINALITY RATIO

Maximum Cardinality

- One-to-one (1:1)
- One-to-many (1:N) or Many-to-one (N:1)
- Many-to-many (N:M)

EMPLOYEE   WORKS_FOR   DEPARTMENT

e_1 r_1 d_1

e_2 r_2

e_3 r_3 d_2

e_4 r_4

e_5 r_5 d_3

e_6 r_6

e_7 r_7
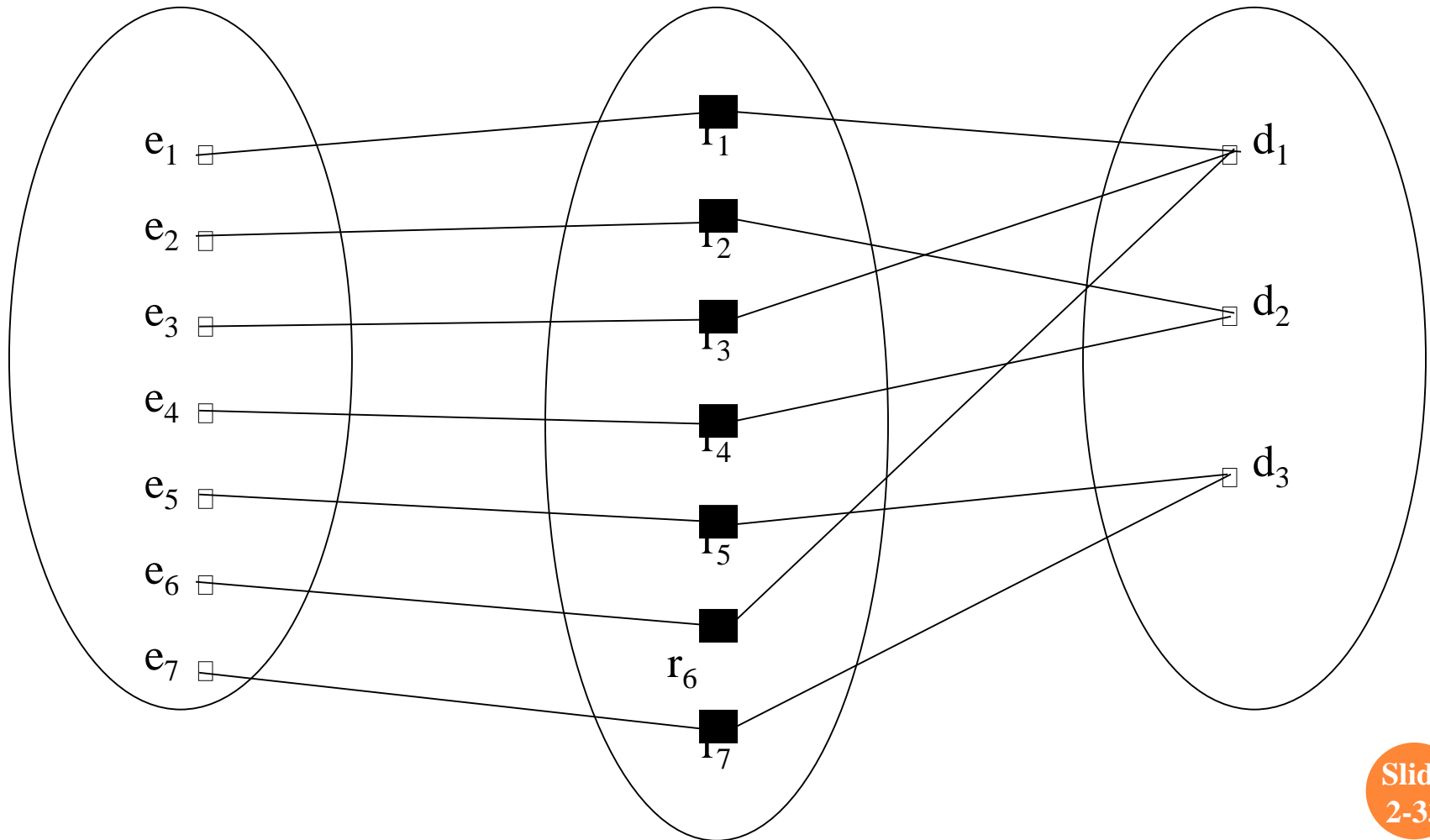
MANY-TO-MANY (M:N) RELATIONSHIP
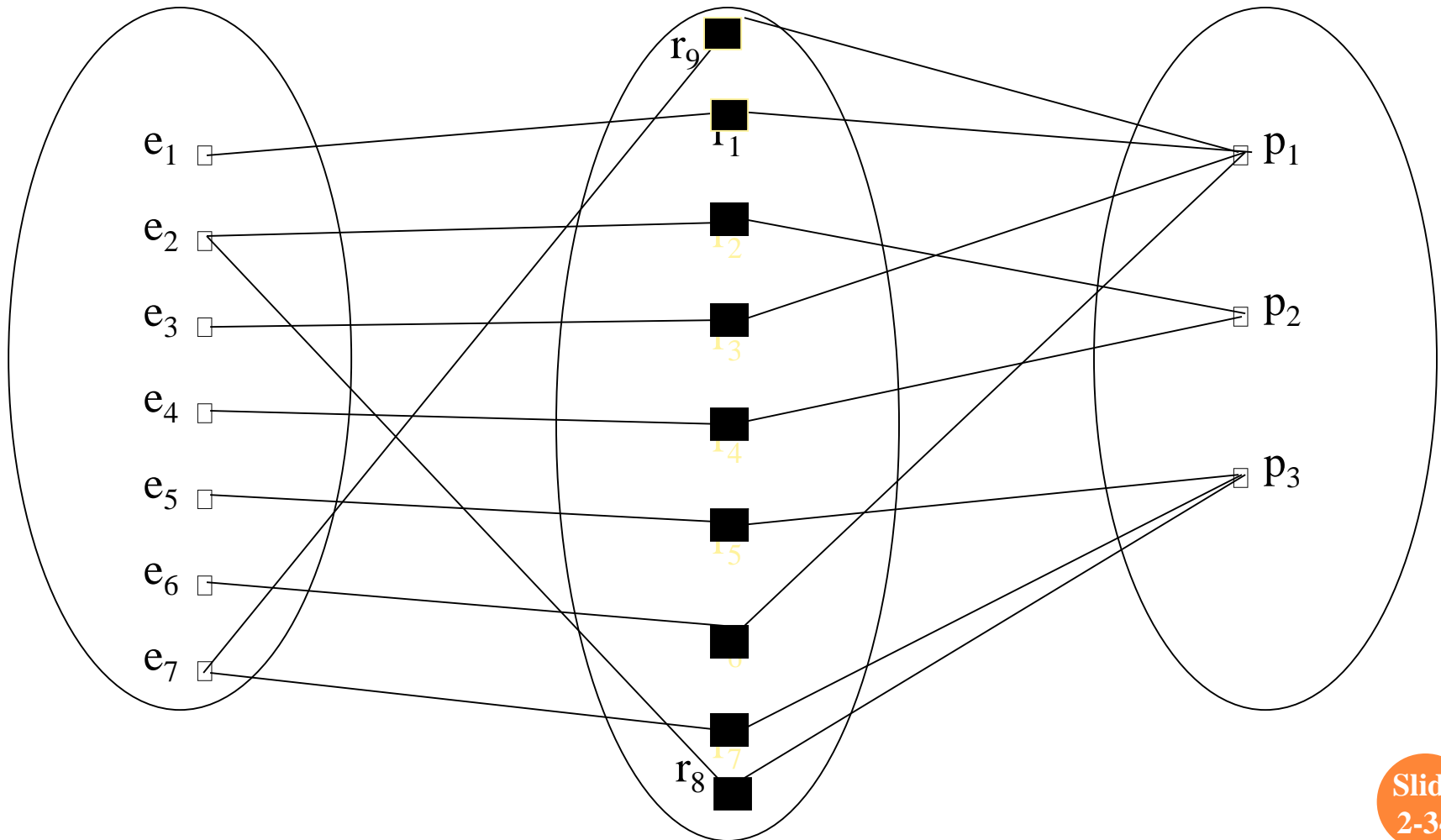
Participation constraint

- Total (existential)
- Partial

Minimum cardinality
  (also called existence dependency constraint)

Slide
2-35

RELATIONSHIP CONSTRAINTS

❑ **Cardinality ratio** (of a binary relationship):

$$1:1, \ 1:N, \ N:1, \ or \ M:N$$

❑ **Participation constraint** (on each participating entity type): total (called *existence dependency*) or partial.

❑ Easy to specify for Binary Relationship Types

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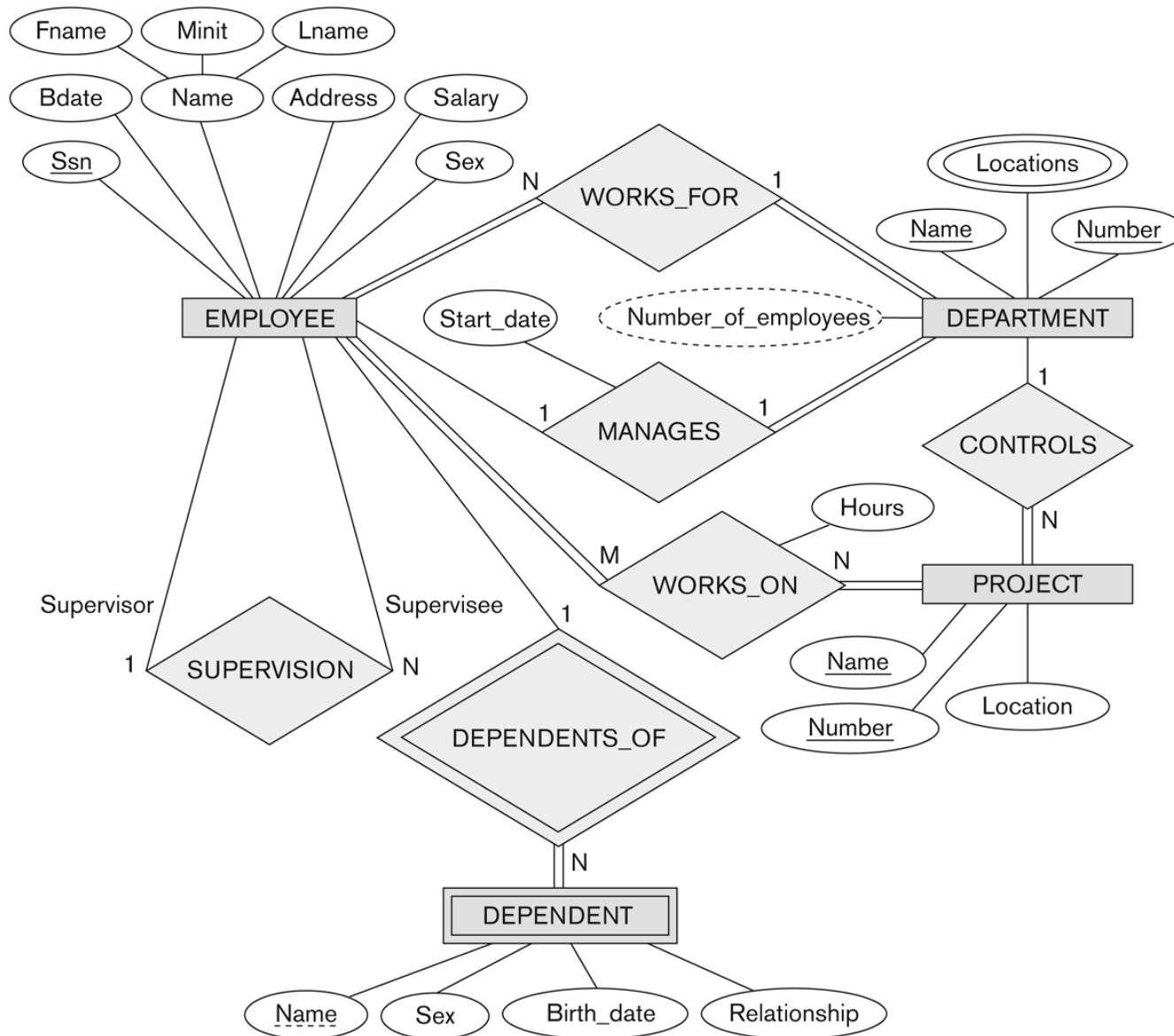
Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

EXAMPLE 2: THE MOVIES DATABASE

You are to design a database for a video club. The following requirements are given:

- Every movie has a unique number associated with it. Additional information about each movie include: the title of the movie, the year it was produced, the type of the movie, the critics rating (number of stars); the number of Academy Award nominations received and won. A movie is directed by one director only.

- Each director is represented by his/her name, unique number, year of birth and year of death (if appropriate).

- Movie stars are assigned unique numbers also. The star's name, birthplace, year of birth and year of death (if applicable) are known, as well as the role(s) the star played in various movies.

THE MOVIE DATABASE (CONT.)

○ Every customer has a unique number. Names and addresses of customers are stored in the database, as well as the number of rentals the customer has made, the number of bonus units the customer is entitled to (one for every 10 rentals) and the date of joining the club.

○ There may be several copies of a movie in the video club. Every DVD has its code (unique). It is known which movie a DVD contains, when it was purchased, how many times it was rented and which customer currently has it (if appropriate).

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
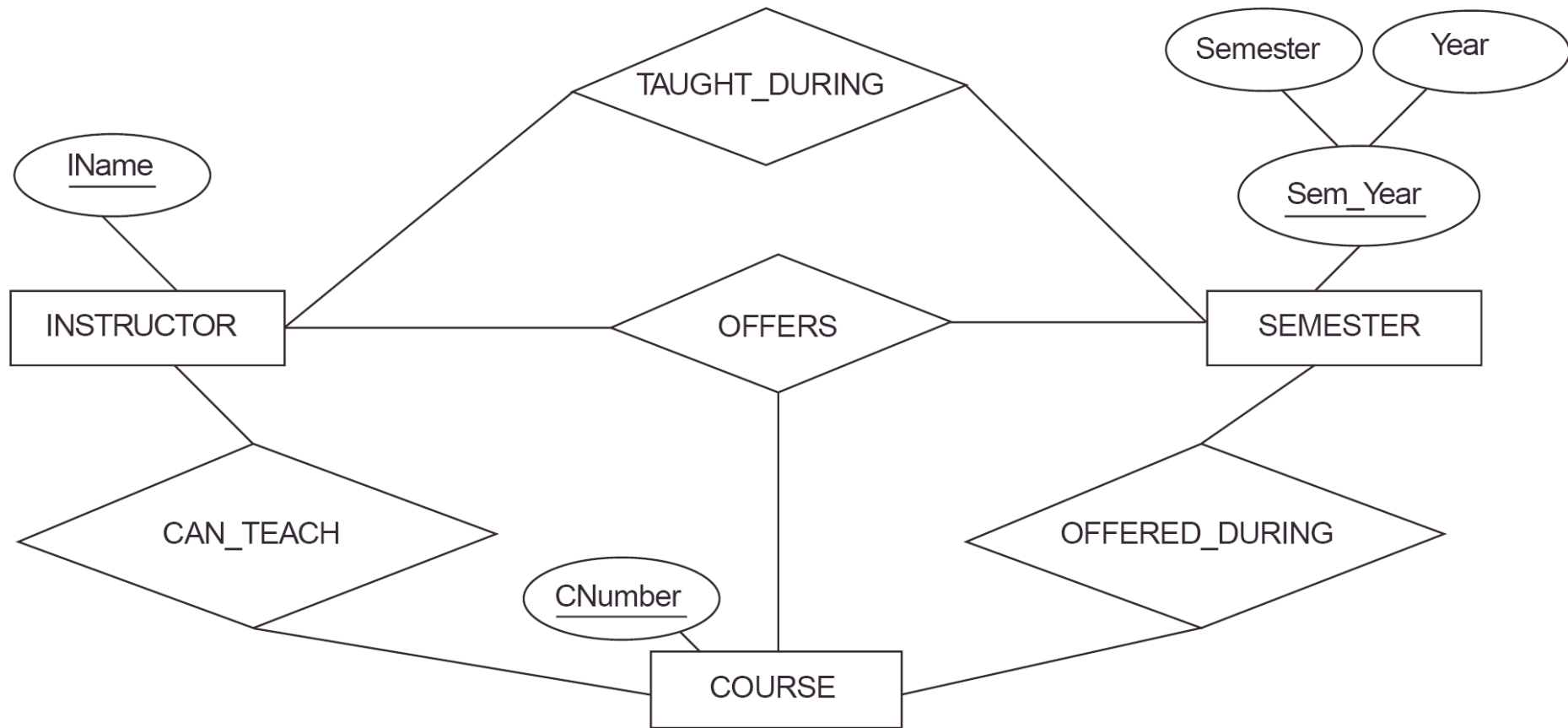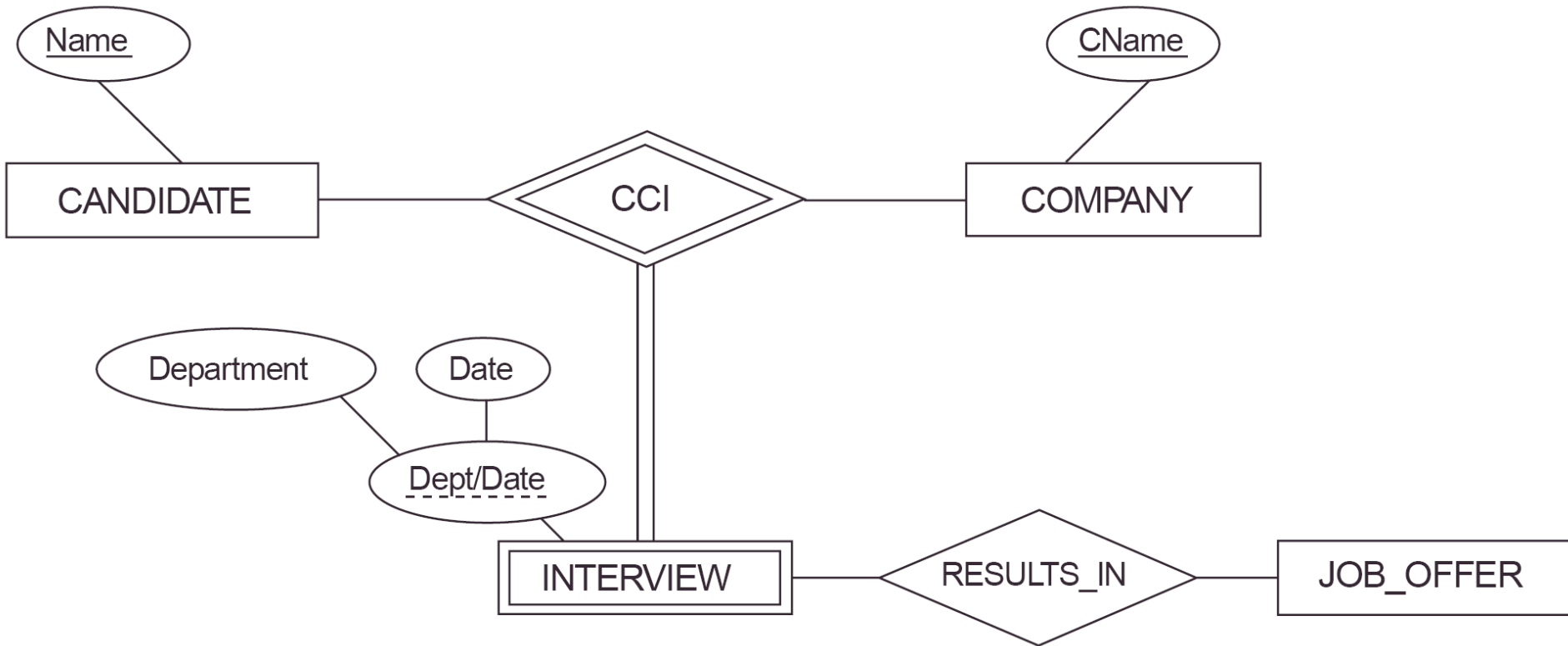
Ternary relationship

(a)

SName

Quantity

ProjName

SUPPLIER

SUPPLY

PROJECT

PartNo

PART

(b)

SName

ProjName

SUPPLIER

SUPPLIES

PROJECT

CAN_SUPPLY

USES

PartNo
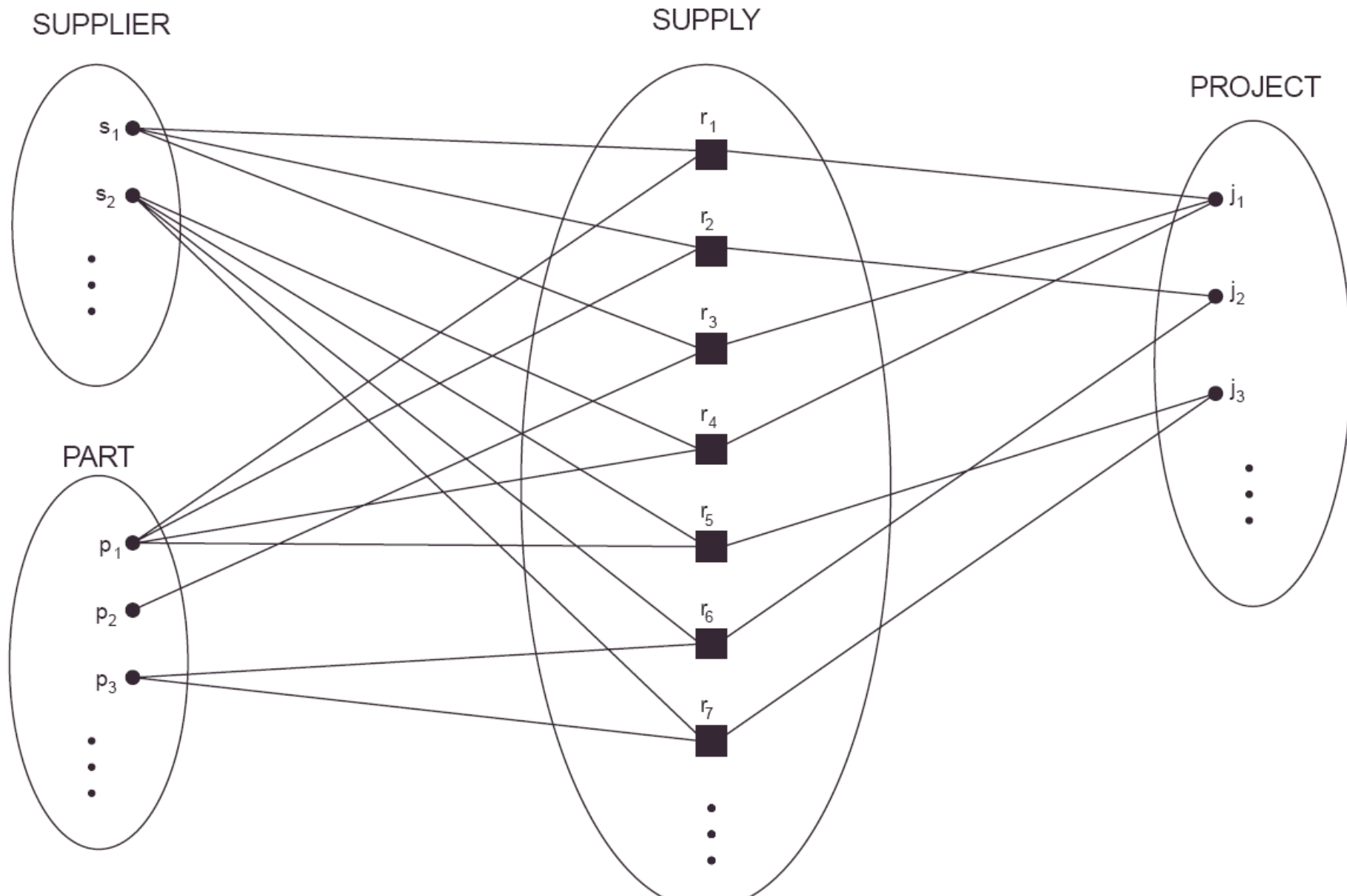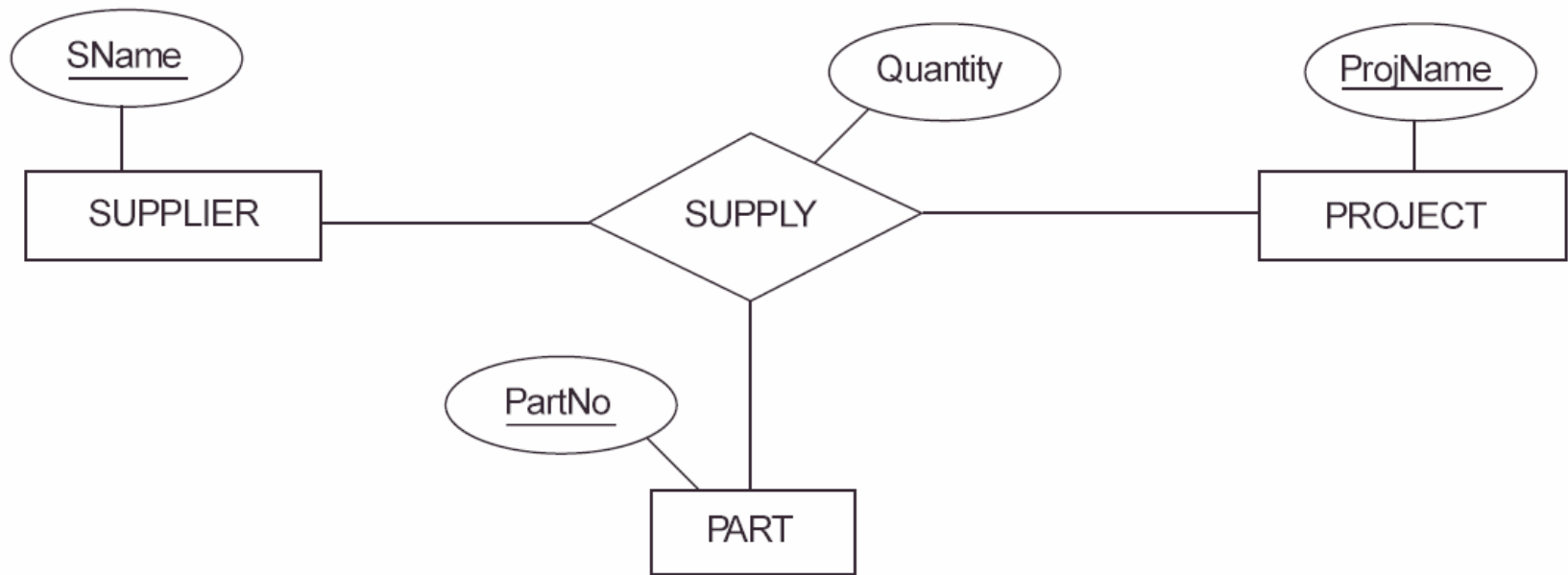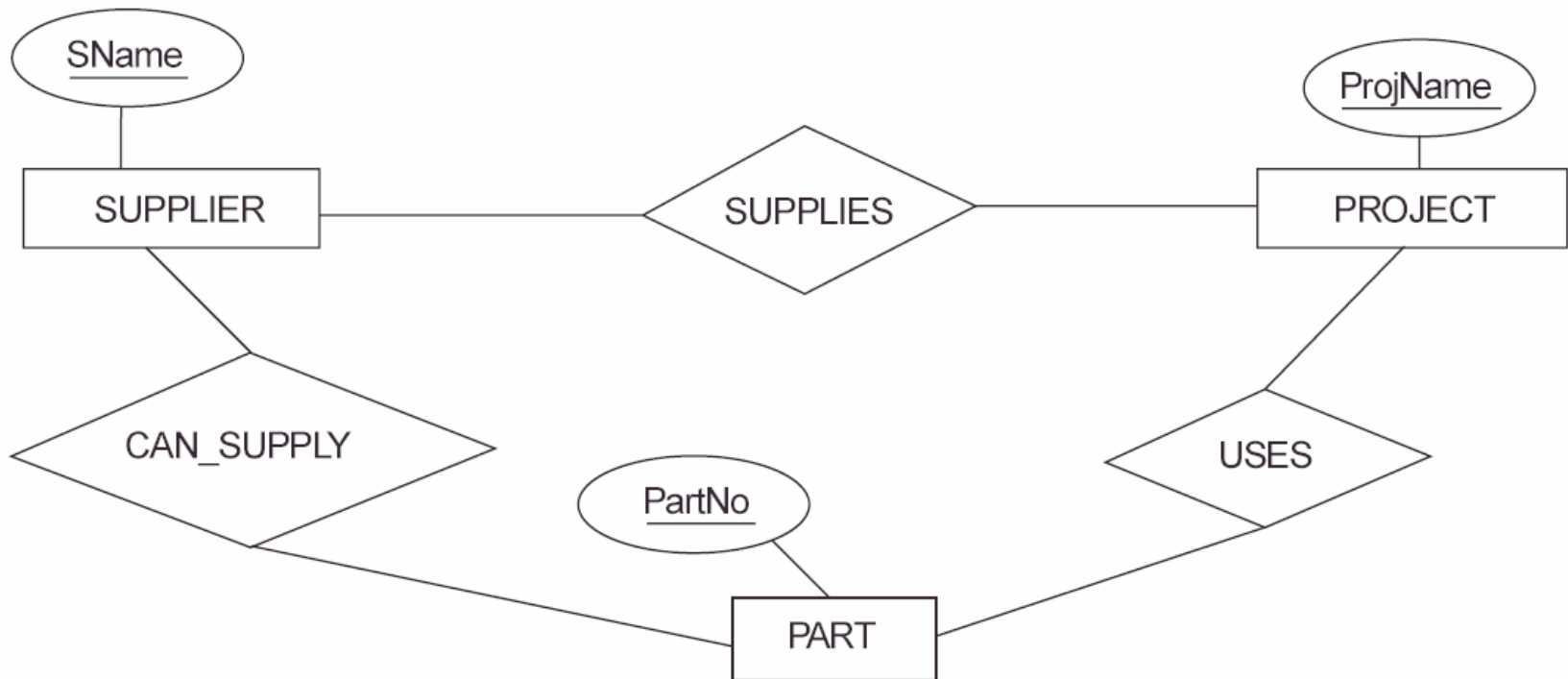
PART

TERNARY RELATIONSHIP (FIG. 3.18)

TERNARY IDENTIFYING RELATIONSHIP

Structural constraints

- (min,max)
- 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
- Must have min≤max, min≥0, max ≥1
- Derived from the knowledge of mini-world constraints

STRUCTURAL CONSTRAINTS

❑ 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
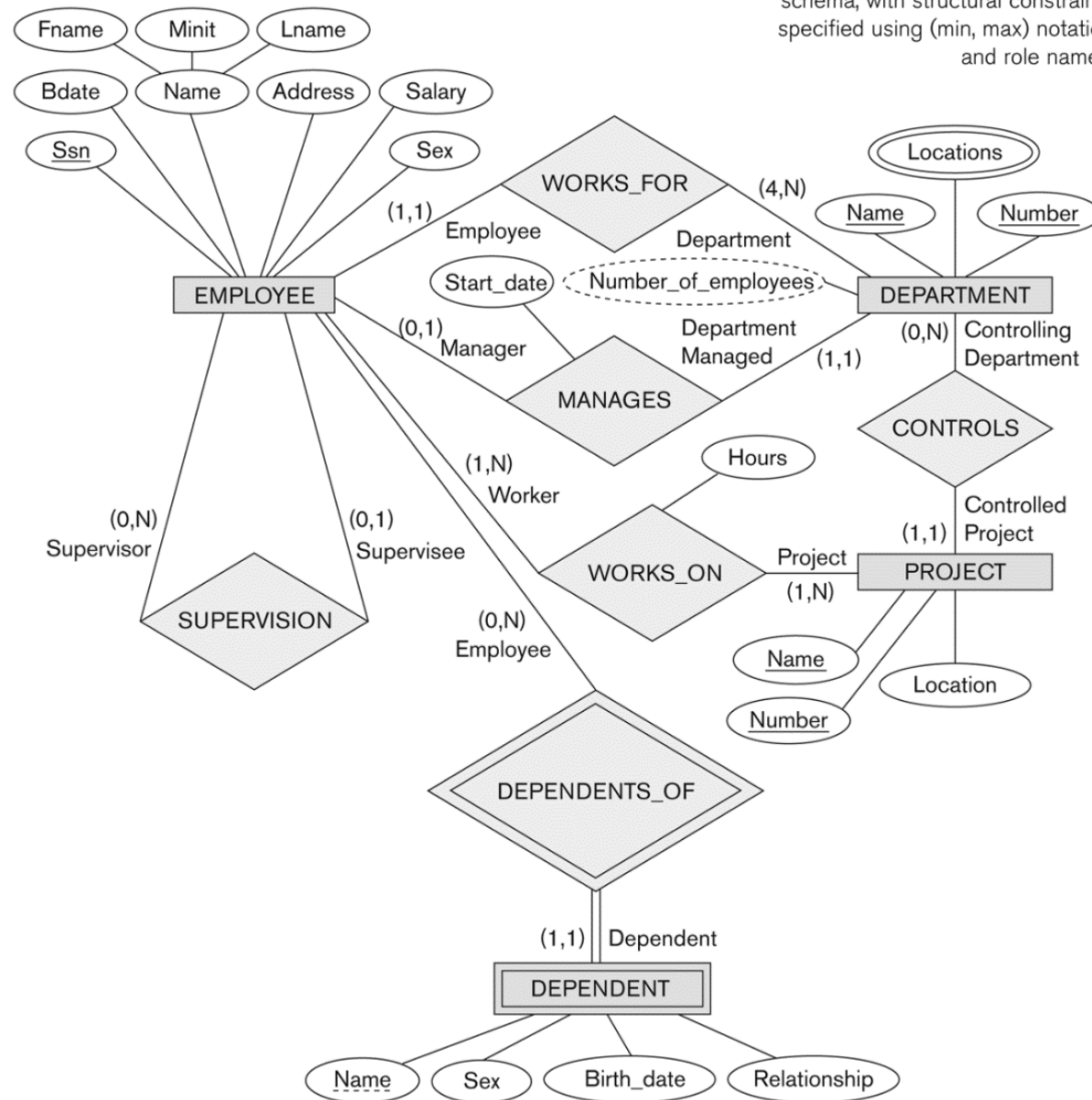- Specify (1,n) for participation of DEPARTMENT in WORKS_FOR

STRUCTURAL CONSTRAINTS

EMPLOYEE —(0,1)— MANAGES —(1,1)— DEPARTMENT

EMPLOYEE —(1,1)— WORKS_IN —(1,N)— DEPARTMENT
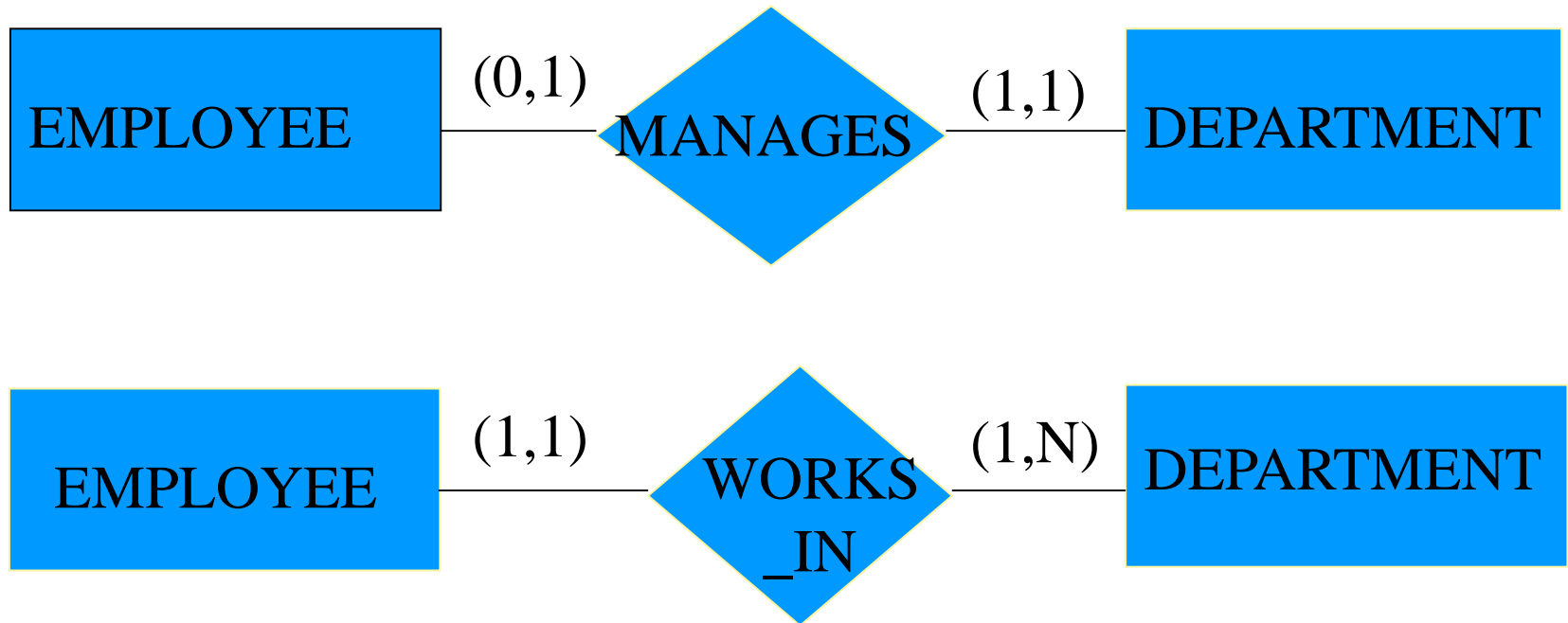
COMPANY ER Schema Diagram using (min, max) notation
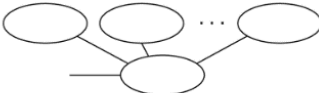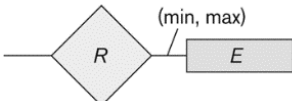
Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

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
(connected ovals)	Composite Attribute
⟋⬭ (dashed)	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
R — (min, max) — E	Structural Constraint (min, max) on Participation of E in R

Naming Conventions

- Choose a name from the standard vocabulary for the domain
- Use a singular noun (adjective + noun) for entity types
- Use verbs for relationship types
- Attribute names should be capitalized

NAMING CONVENTIONS (CONT.)

- All entity/relationship type names should be unique
- All attributes within a single entity types should have unique names
- Role names are shown in lowercase letters
- ER diagram should be readable left to right, and from top to bottom

Phases of ER modeling

1. Identify entity types and their attributes
2. Identify relationship types and their attributes
3. Examine the design for potential problems and reiterate 1-3

IDENTIFY ENTITY TYPES AND ATTRIBUTES

Look for

1. Locations
2. Events
3. Other systems
4. Roles played
5. Organizational units structures

IDENTIFY ENTITY TYPES AND ATTRIBUTES

The attribute test:

If attribute has parts, are any of these parts likely to appear in a query?

Avoid storing unnecessary attributes

# IDENTIFY RELATIONSHIP TYPES AND THEIR ATTRIBUTES

Heuristics:

- Look for different roles of the same entity type to see how many relationship exist
- Look for associations between instances of the same entity type to identify recursive relationship types

Slide
2-55

POTENTIAL DESIGN PROBLEMS

- An entity type having only one attribute

- An entity type not participating in any relationship type (missing relationship?)

- The appearance of an attribute of one entity type that refers to some other entity type (missing relationship)

- Two relationship types

  E1 RT1 E2   and   E2 RT2 E1

  EMPLOYEE WORKS-ON PROJECT

  PROJECT REQUIRES EMPLOYEE

AN EXAM QUESTION

[8 marks] Examine the ER schema given below and identify any deficiencies in it.
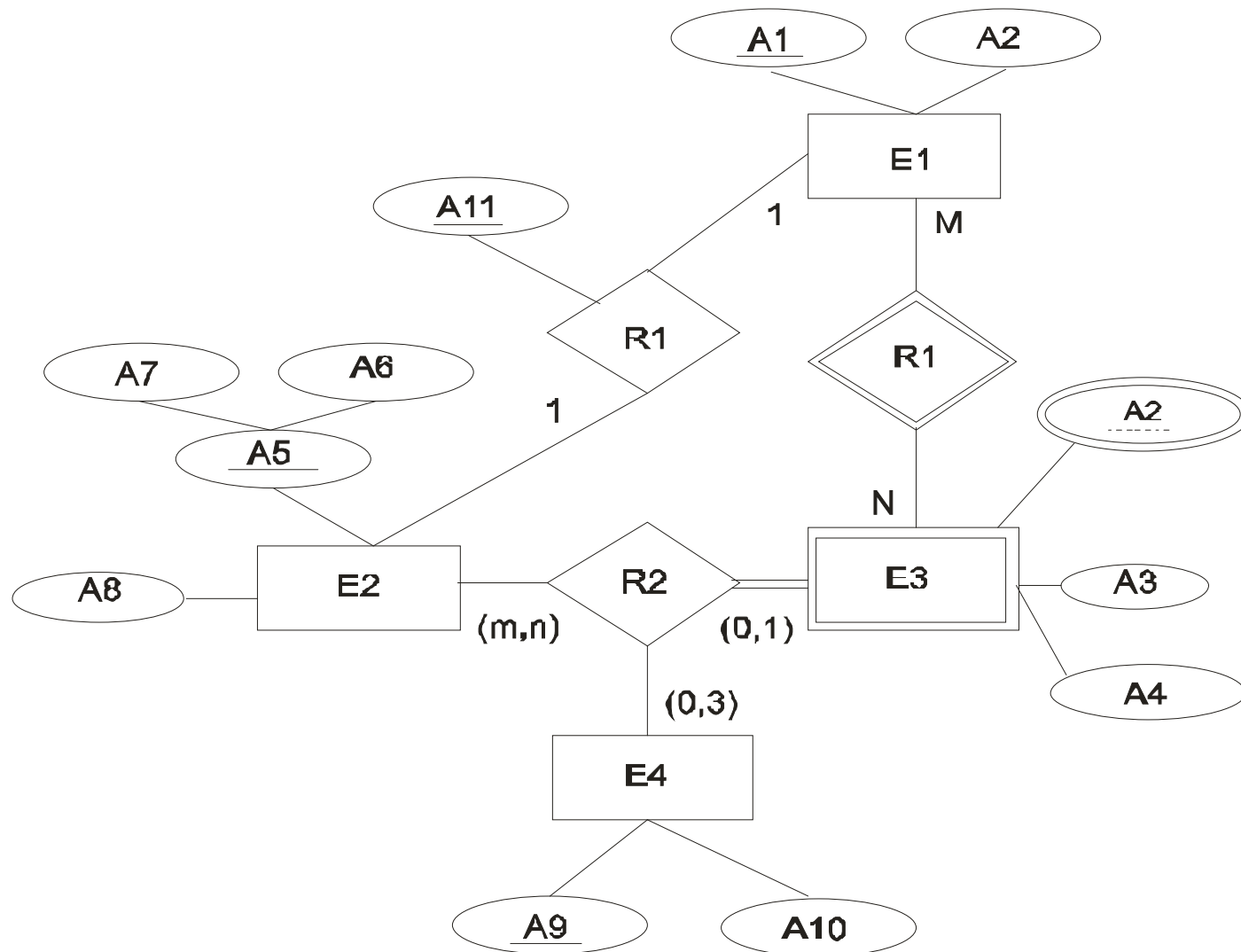
EQUIVALENT DIAGRAMS?
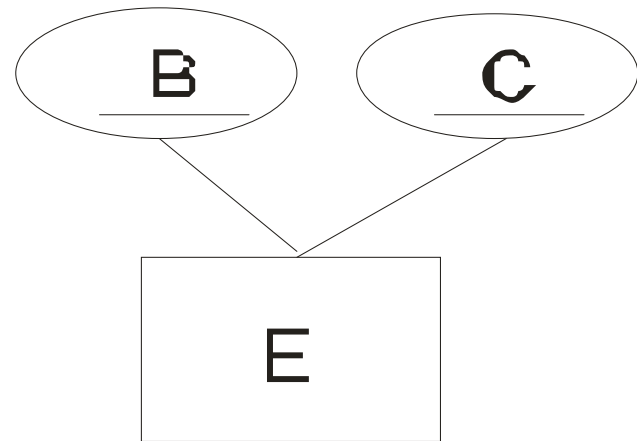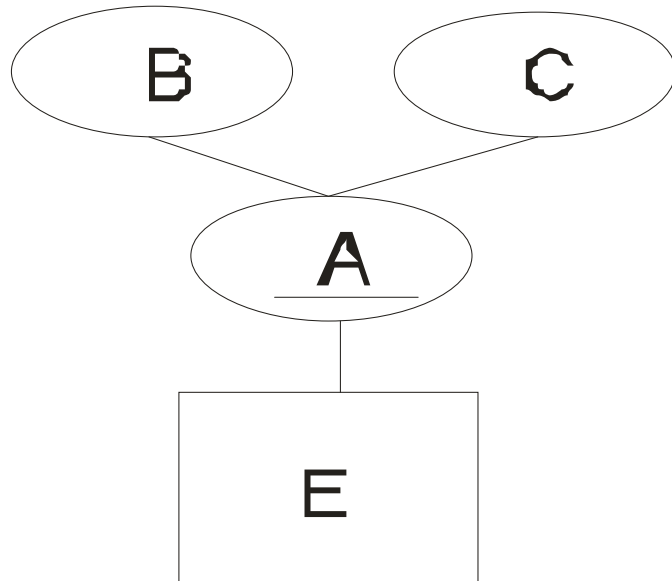
OVERVIEW

1. Phases of database design
2. Entity-Relationship Model (ER)
3. **Enhanced Entity-Relationship Model (EER)**
   - The need for more expressive models
   - New concepts
     - Superclass/subclass relationship
     - Generalization and specialization
     - Attribute inheritance
     - Category
4. Why model?
5. Data modeling tools

THE NEED FOR MORE EXPRESSIVE MODELS

- ER model does not support abstraction (generalization and specialization)
- New areas
  - CAD/CAM systems
  - Multimedia databases
  - GIS databases
  - AI Knowledge bases
- New requirements
  - Complex objects, multiple versions
  - Special data types for unstructured data

AN EXAMPLE: THE *PERFECT HOME* ASSIGNMENT

- You are chosen to design a database for the *Perfect Home* agency which manages properties for rent on behalf of their owners.

- …

- Each staff member has a (unique) number, name and address, phone, sex, date of birth, IRD number, job title, salary and the date of joining the organization and the office he/she is working for. The staff work in groups that are supervised by a supervisor and supported by a secretary. For secretaries, the agency stores typing speed. For each manager, the agency needs to know the starting date and an annual car allowance.

PERFECT HOME DATABASE (CONT.)
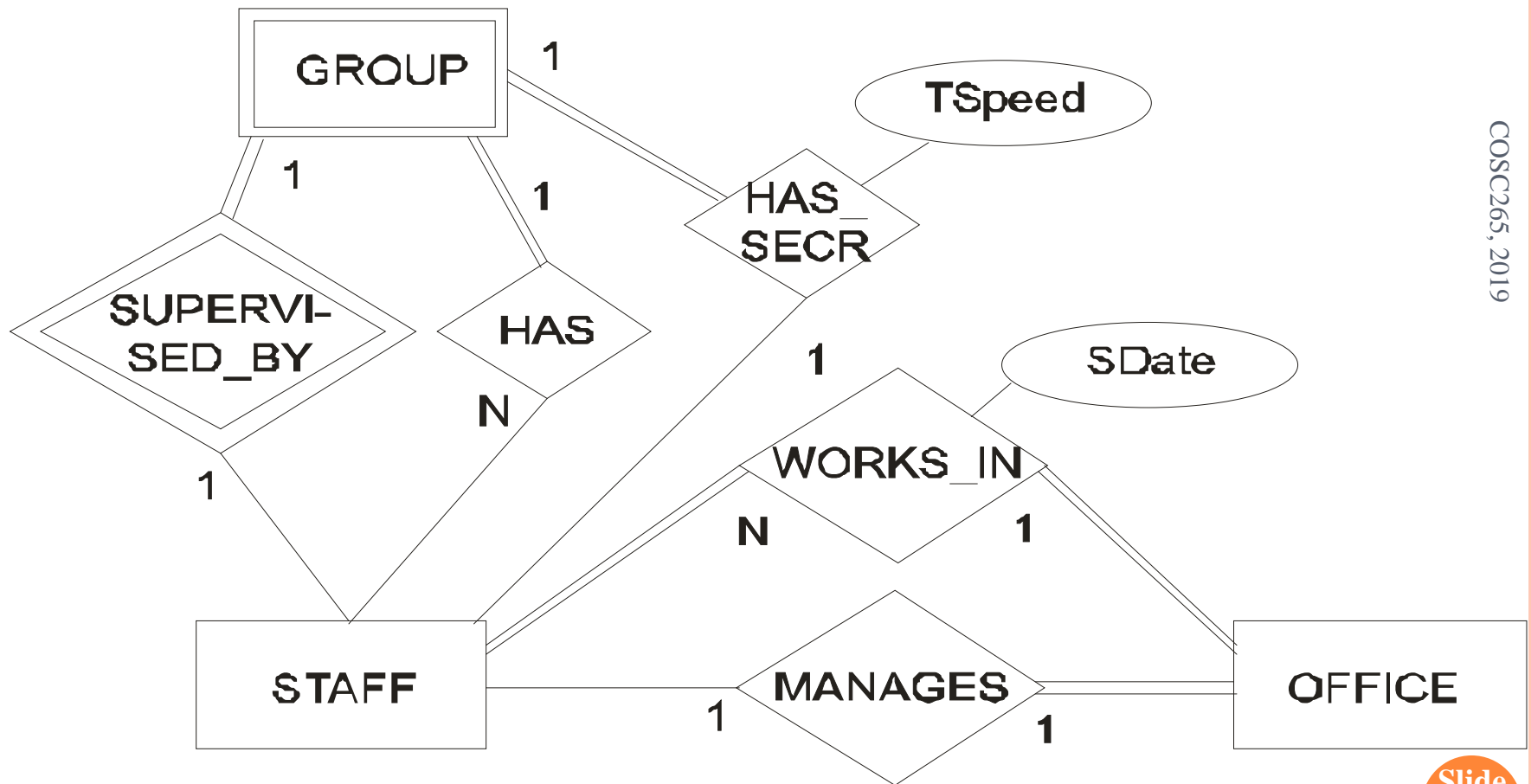
PERFECT HOME DATABASE (CONT.)

- The agency manages property for private or business owners.
- Each owner (private or business) has a unique number and owns at least one property.
- Additional information on private owners includes the owner's name, address and phone number.
- The details of business owners include the name of the business, type of business, address, phone number and contact name.

# *PERFECT HOME* DATABASE (CONT.)
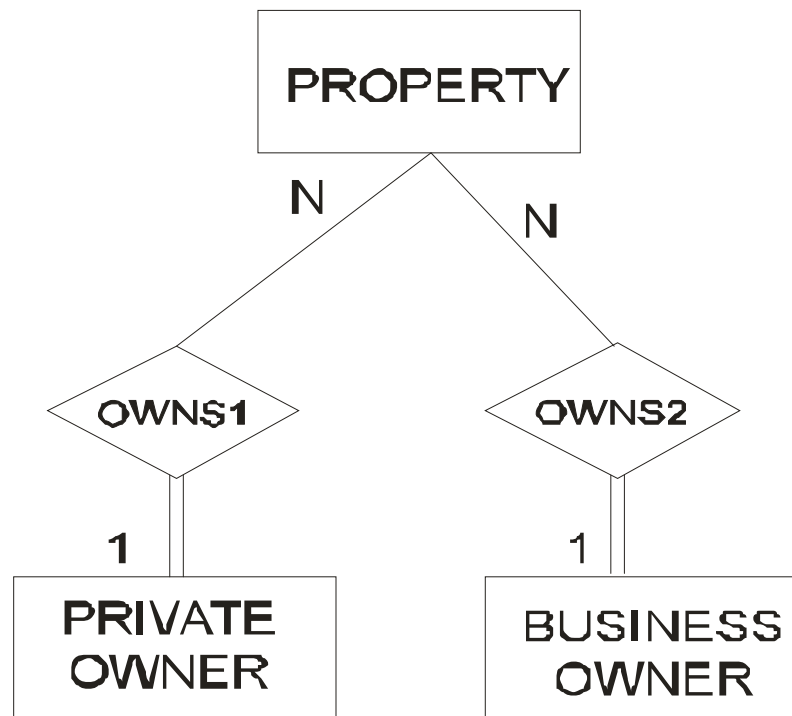
Enhanced-ER (EER) Model Concepts

- Includes all modeling concepts of basic ER
- Additional concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
- The resulting model is called the enhanced-ER or Extended ER (E2R or EER) model

SUBCLASSES AND SUPERCLASSES

- An entity type may have meaningful subgroupings of its entities
- Example: EMPLOYEE may be grouped into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE,...
  - Each of these groupings is a subset of EMPLOYEE entities
  - Each is called a subclass of EMPLOYEE
  - EMPLOYEE is the superclass for each of these subclasses
- Superclass/subclass (*is-a*) relationships
- An entity that is member of a subclass represents the same real-world entity as some member of the superclass
- Not every entity in a superclass will be a member of some subclass

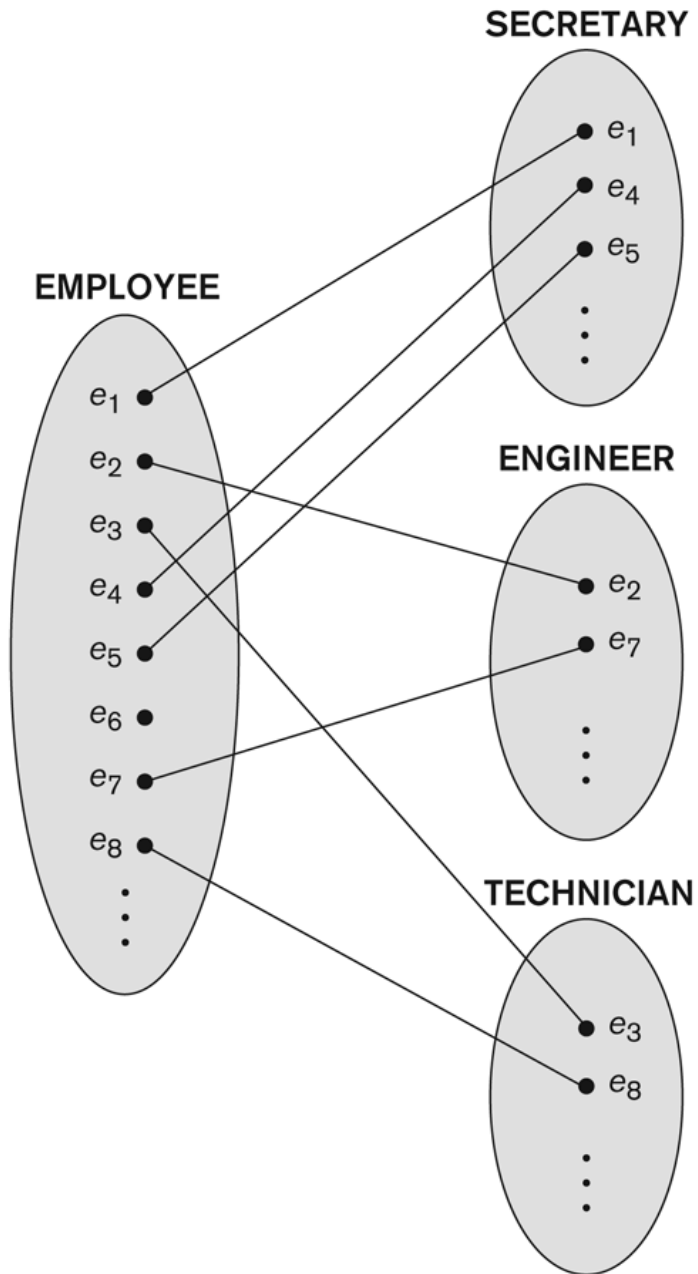# EXAMPLE OF A SPECIALIZATION
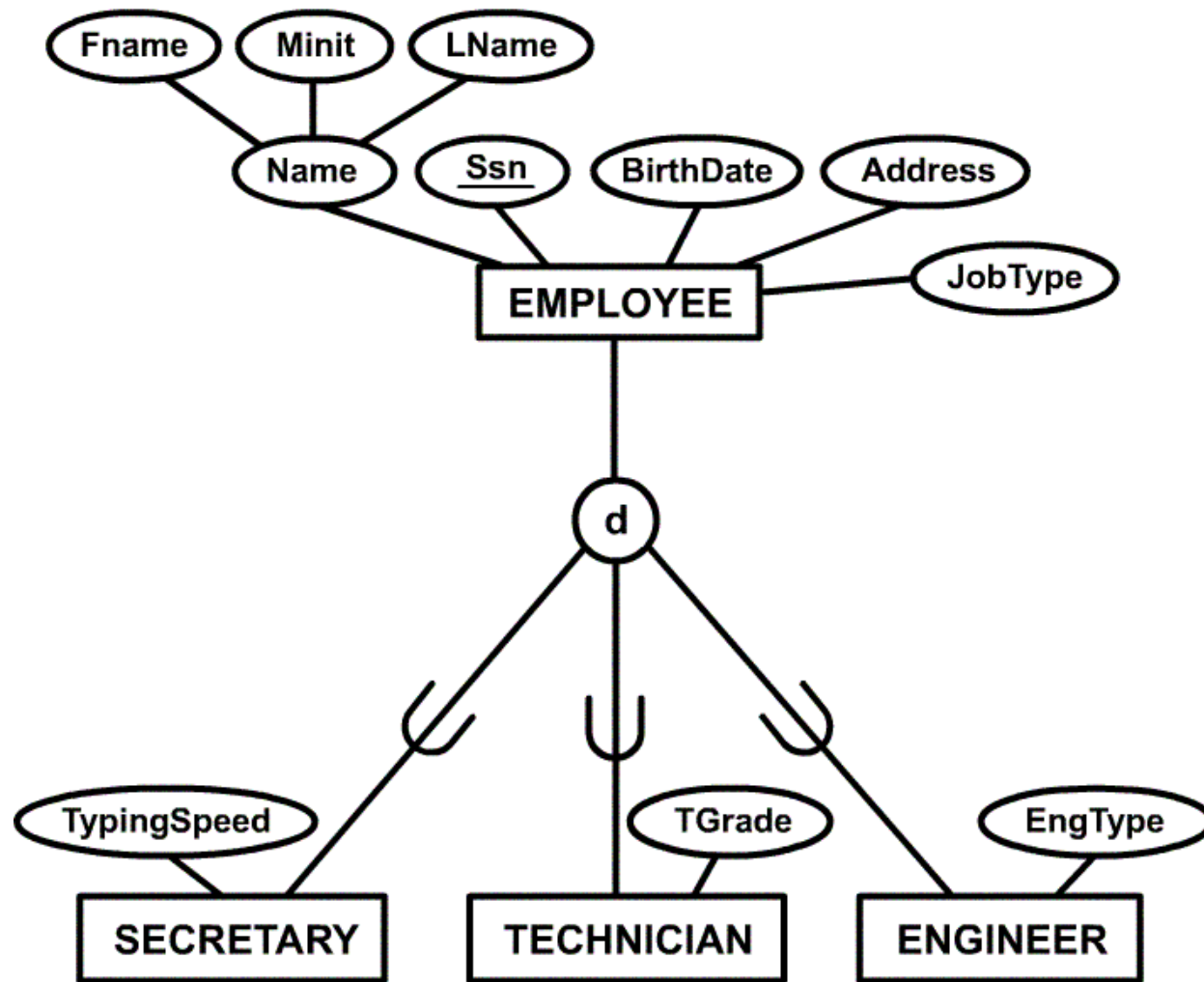
Figure 7.20
Instances of a specialization.

ATTRIBUTE INHERITANCE

- An entity that is member of a subclass *inherits* all attributes of the entity as a member of the superclass
- It also inherits all relationships

SPECIALIZATION

- The process of defining a set of subclasses of a superclass

- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass

- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type*.
  - May have several specializations of the same superclass

- Example: Another specialization of EMPLOYEE based in *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
  - Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams
  - Attributes of a subclass are called specific attributes. For example, TypingSpeed of SECRETARY
  - The subclass can participate in specific relationship types. For example, BELONGS_TO of HOURLY_EMPLOYEE

Figure 4.1
EER diagram notation to represent
subclasses and specialization.
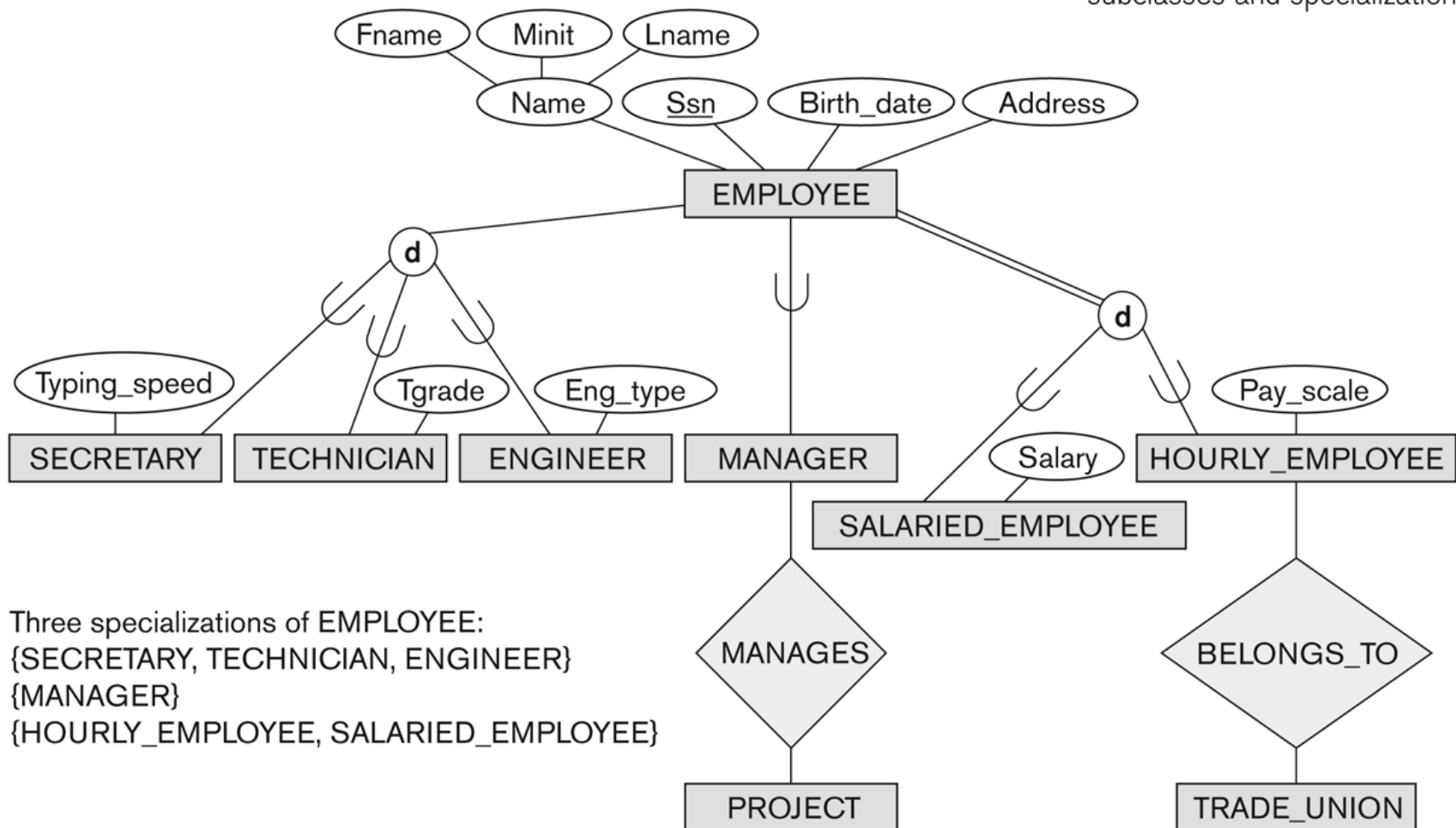
Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

GENERALIZATION

- The reverse of the specialization process

- Several classes with common features are generalized into a superclass; original classes become its subclasses

- Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

(a) CAR entity type with attributes: No_of_passengers, Max_speed, Vehicle_id, License_plate_no, Price. TRUCK entity type with attributes: No_of_axles, Tonnage, Vehicle_id, License_plate_no, Price.

(b) VEHICLE superclass with attributes: Vehicle_id, Price, License_plate_no. Generalization (d) into subclasses CAR (No_of_passengers, Max_speed) and TRUCK (No_of_axles, Tonnage).
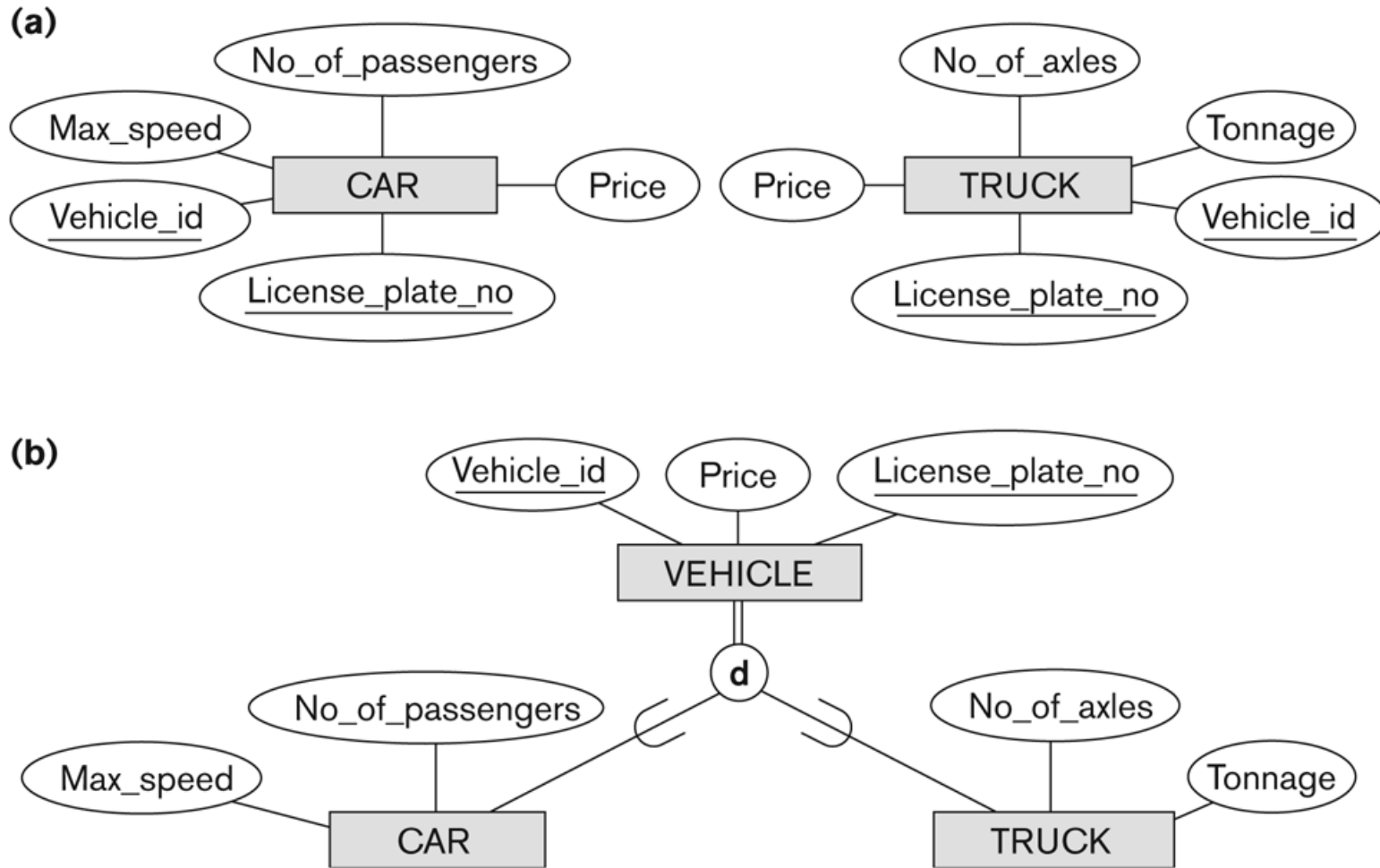
Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

GENERALIZATION AND SPECIALIZATION

- A superclass or subclass represents a set of entities

- Shown in rectangles in EER diagrams (as are entity types)

- Sometimes, all entity sets are simply called classes, whether they are entity types, superclasses, or subclasses

CONSTRAINTS ON SPECIALIZATION AND GENERALIZATION (1)

○ **Predicate defined specialization**: determined by a condition (predicate)

○ **Attribute defined-specialization**: defining attributes

- Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE

○ **User defined specialization**: no condition

Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass

CONSTRAINTS ON SPECIALIZATION AND GENERALIZATION (2)

- **Disjointness Constraint**:
  - **Disjoint**: an entity can be a member of at most one of the subclasses of the specialization (specified by **d)**
  - **Overlap**: the same entity may be a member of more than one subclass of the specialization (specified by **o)**

- **Completeness Constraint**:
  - **Total** specifies that every entity in the superclass must be a member of some subclass in the specialization/ generalization
  - Shown in EER diagrams by a double line
  - **Partial** allows an entity not to belong to any of the subclasses
  - Shown in EER diagrams by a single line

Constraints on Specialization and Generalization (3)

○ four types of specialization/generalization:
- Disjoint, total
- Disjoint, partial
- Overlapping, total
- Overlapping, partial

Figure 4.4
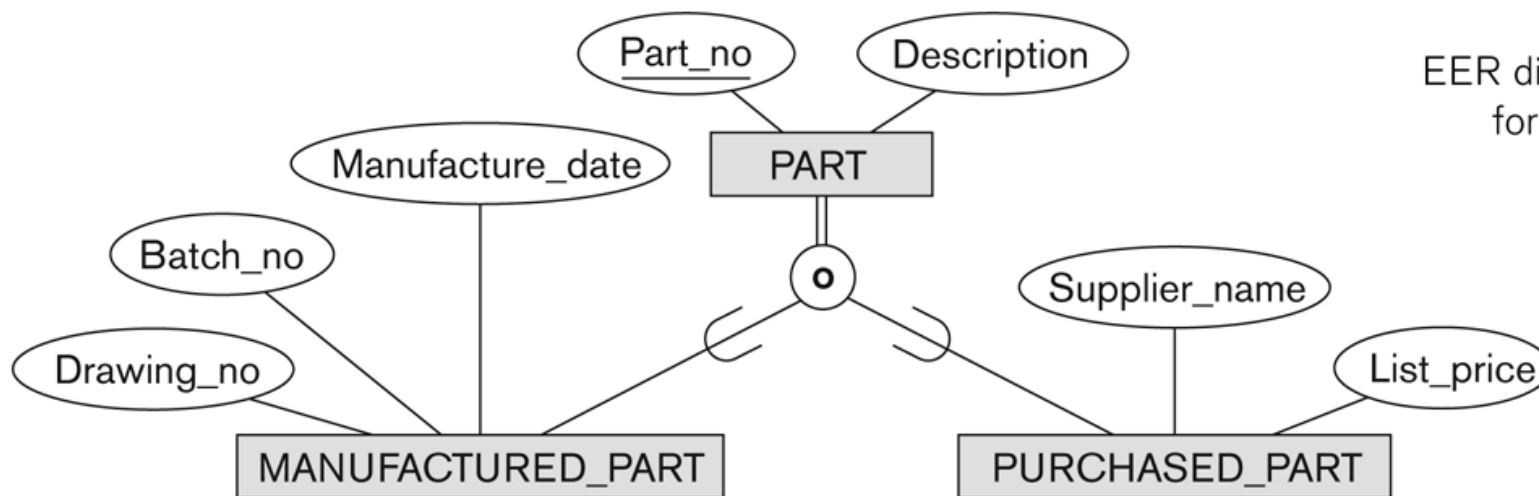EER diagram notation for an attribute-defined specialization on Job_type.
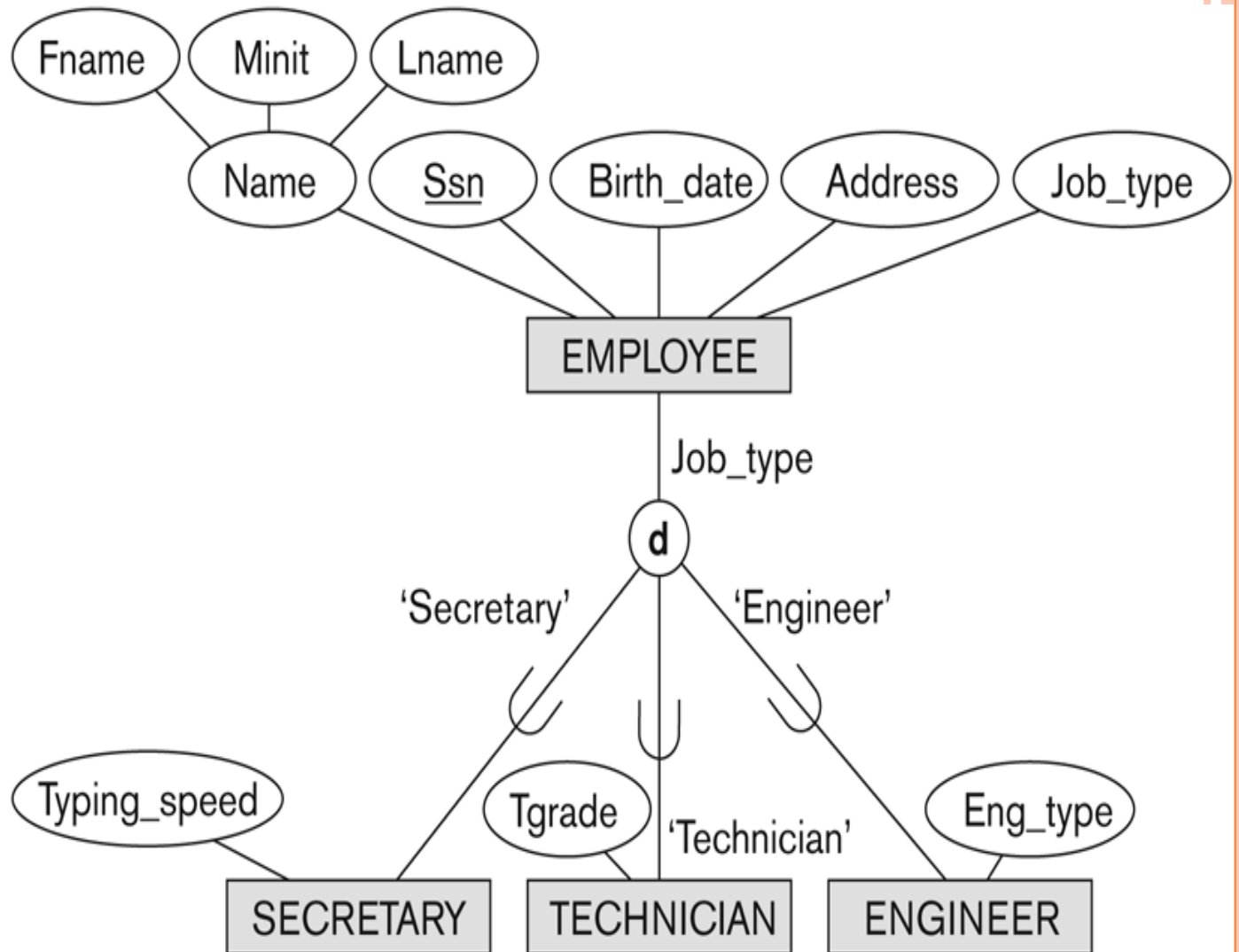
Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Specialization/Generalization Hierarchies

- A subclass may itself have further subclasses specified on it
- Forms a hierarchy
- Hierarchy has a constraint that every subclass has only one superclass (called *single inheritance*)
- A subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses
- In specialization, start with an entity type and then define subclasses of the entity type by successive specialization (top down conceptual refinement process)
- In generalization, start with many entity types and generalize those that have common properties (bottom up conceptual synthesis process)
- In practice, the combination of two processes is employed

SPECIALIZATION LATTICES AND SHARED SUBCLASS

- In a lattice, a subclass can be subclass of more than one superclass (called *multiple inheritance*)

- A subclass with more than one superclass is called a *shared subclass*

- A shared subclass is subclass in more than one distinct superclass/subclass relationships, where each relationship has a single superclass
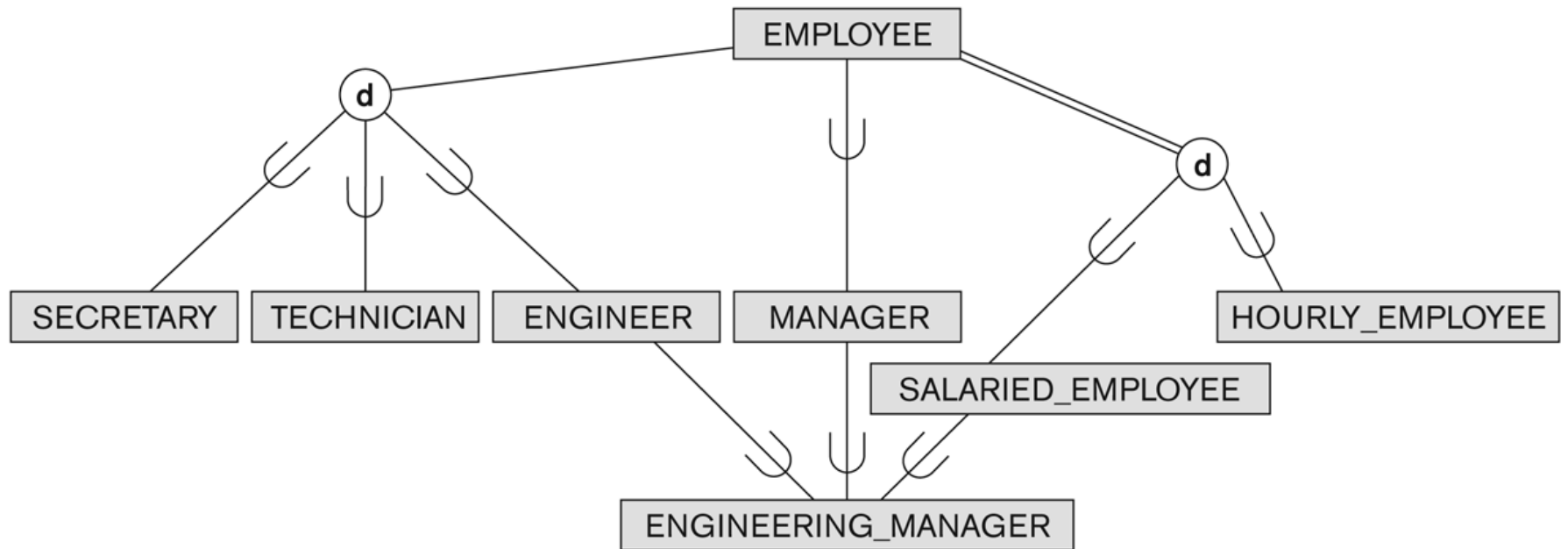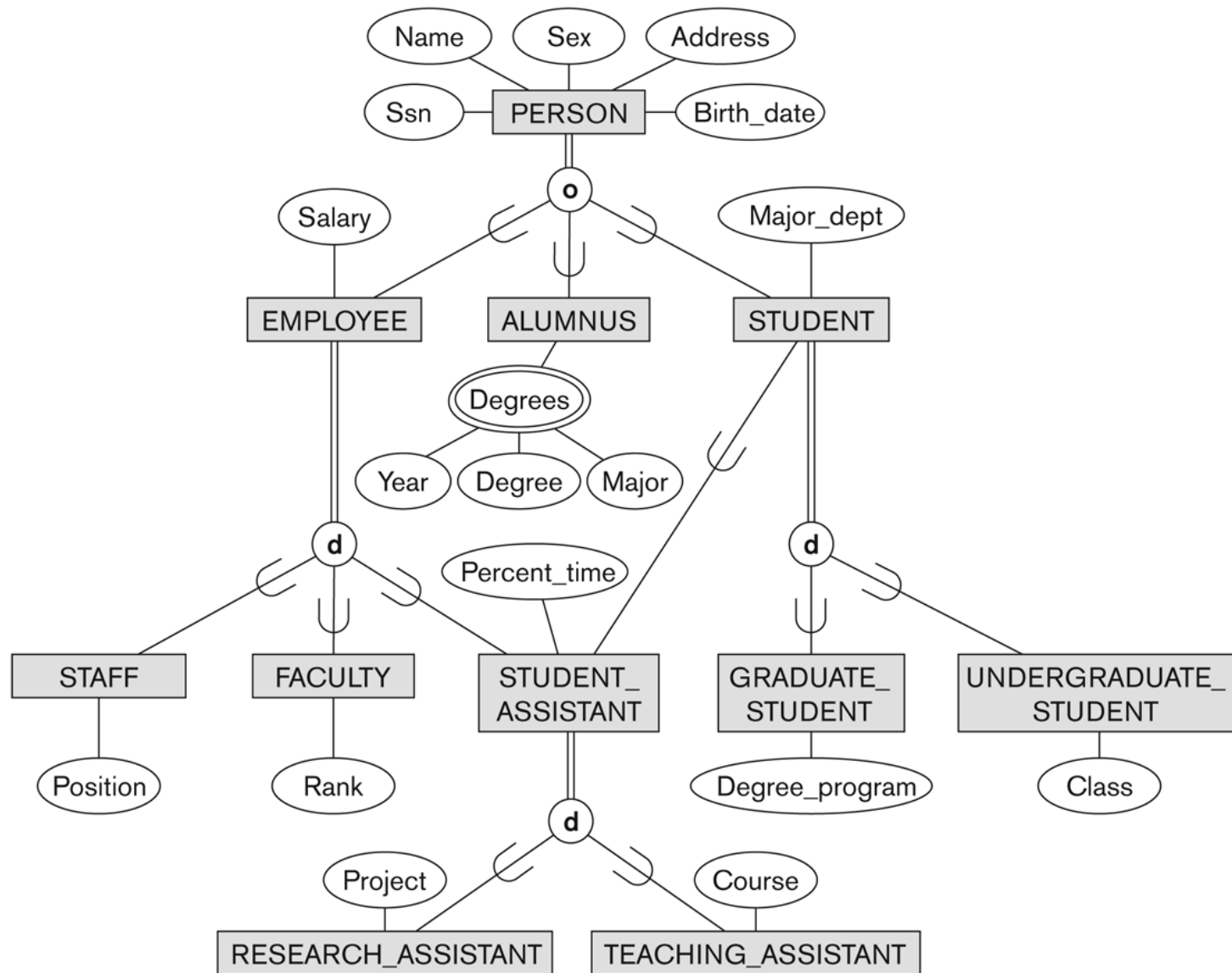
Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

CATEGORIES (UNION TYPES)

- In some cases, need to model a single superclass/subclass relationship with more than one superclass

- Superclasses represent different entity types

- Such a subclass is called a *category* or *union type*

- *Selective inheritance*

CATEGORY

Different from a shared subclass, which is subset of the intersection of its superclasses (shared subclass member must exist in all of its superclasses).
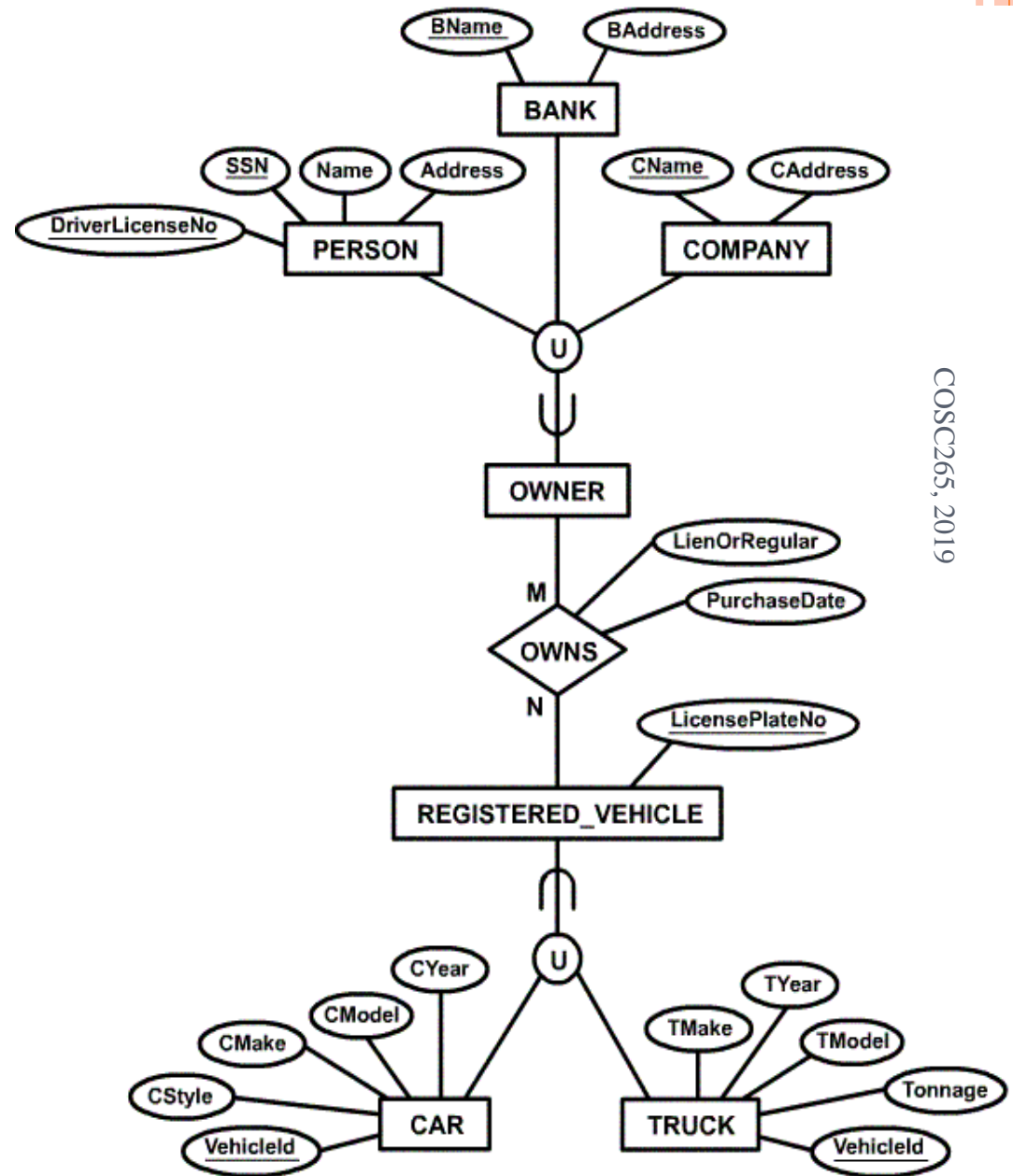
Heuristic rules for EER modeling

- Nouns are candidate for entity types
- Verbs are candidate for relationship types
- Different roles for entities of the same type: several relationships needed
- Phrases like *at most one, at least one, several* etc. indicate cardinality ratios
- Entity type has several groupings: specialization
- Two different entity types play the same role: category

WHY MODEL?

- To *understand* the problem better
- To identify the basic components of the solution
- To support reuse
- To develop a sharable database
- To enable integration of heterogeneous systems (reverse engineering)

DATA MODELING TOOLS

A number of popular tools that cover conceptual modeling and mapping into relational schema design.

POSITIVES: serves as documentation of application requirements, easy user interface - mostly graphics editor support

SOME DATABASE DESIGN TOOLS

- Oracle
  - Oracle Designer
  - Oracle SQL Developer Data Modeller
  - Visual Paradigm (ER and UML)
- MySQL Workbench
- Microsoft SQL Server Management Studio
- IBM Rational Rose XDE
- Others
  - Erwin data modeller (3rd party)
  - Enterprise Architect (UML, 3rd party)
  - …

PROBLEMS WITH MODELING TOOLS

- DIAGRAMMING
  - Poor conceptual meaningful notation
  - To avoid the problem of layout algorithms and aesthetics of diagrams, they prefer boxes and lines and do nothing more than represent relationships

- METHODOLOGY
  - lack of built-in methodology support
  - poor trade-off analysis or user-driven design preferences
  - poor design verification and suggestions for improvement