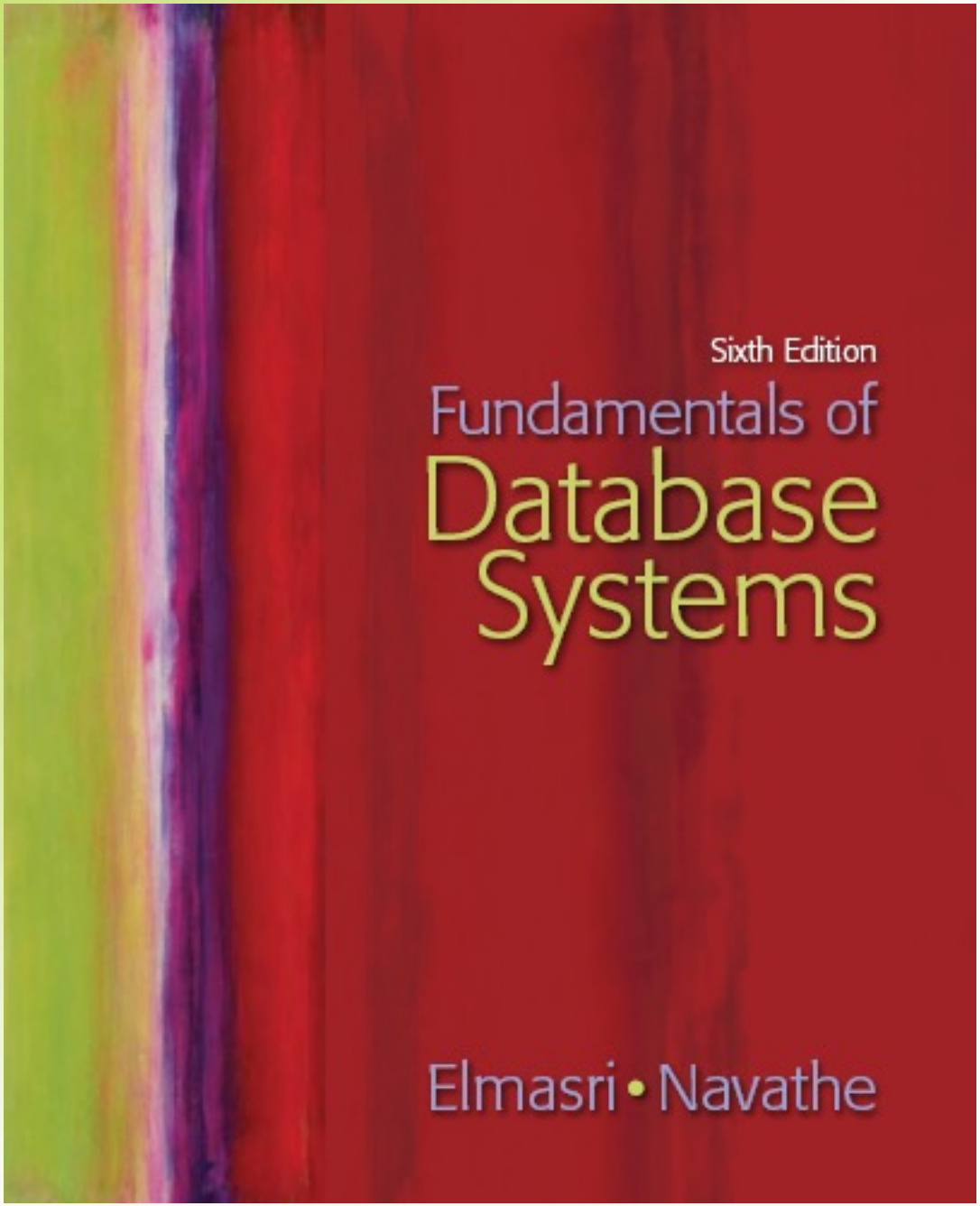
Chapter 8

(Ch. 4 in 11th Ed)

The Enhanced Entity-Relationship (EER) Model



Addison-Wesley
is an imprint of
PEARSON

Chapter 8 Outline

- Subclasses, Superclasses, and Inheritance
- Specialization and Generalization
- Constraints and Characteristics of Specialization and Generalization Hierarchies
- Modeling of UNION Types Using Categories

Chapter 8 Outline (cont'd.)

- A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions
- Example of Other Notation: Representing Specialization and Generalization in UML Class Diagrams
- Data Abstraction, Knowledge Representation, and Ontology Concepts

The Enhanced Entity-Relationship (EER) Model

- **Enhanced ER (EER) model**
  - Created to design more accurate database schemas
    - Reflect the data properties and constraints more precisely
  - More complex requirements than traditional applications

Subclasses, Superclasses, and Inheritance

- EER model includes all modeling concepts of the ER model
- In addition, EER includes:
  - **Subclasses** and **superclasses**
  - **Specialization** and **generalization**
  - **Category** or **union type**
  - **Attribute** and **relationship inheritance**

Subclasses, Superclasses, and Inheritance (cont'd.)

- **Enhanced ER** or **EER diagrams**
  - Diagrammatic technique for displaying these concepts in an EER schema
- **Subtype** or **subclass** of an entity type
  - Subgroupings of entities that are meaningful
  - Represented explicitly because of their significance to the database application

Subclasses, Superclasses, and Inheritance (cont'd.)

- Terms for relationship between a superclass and any one of its subclasses
  - **Superclass/subclass**
  - **Supertype/subtype**
  - **Class/subclass** relationship
- An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
- **Type inheritance**
  - Subclass entity inherits all attributes and relationships of superclass

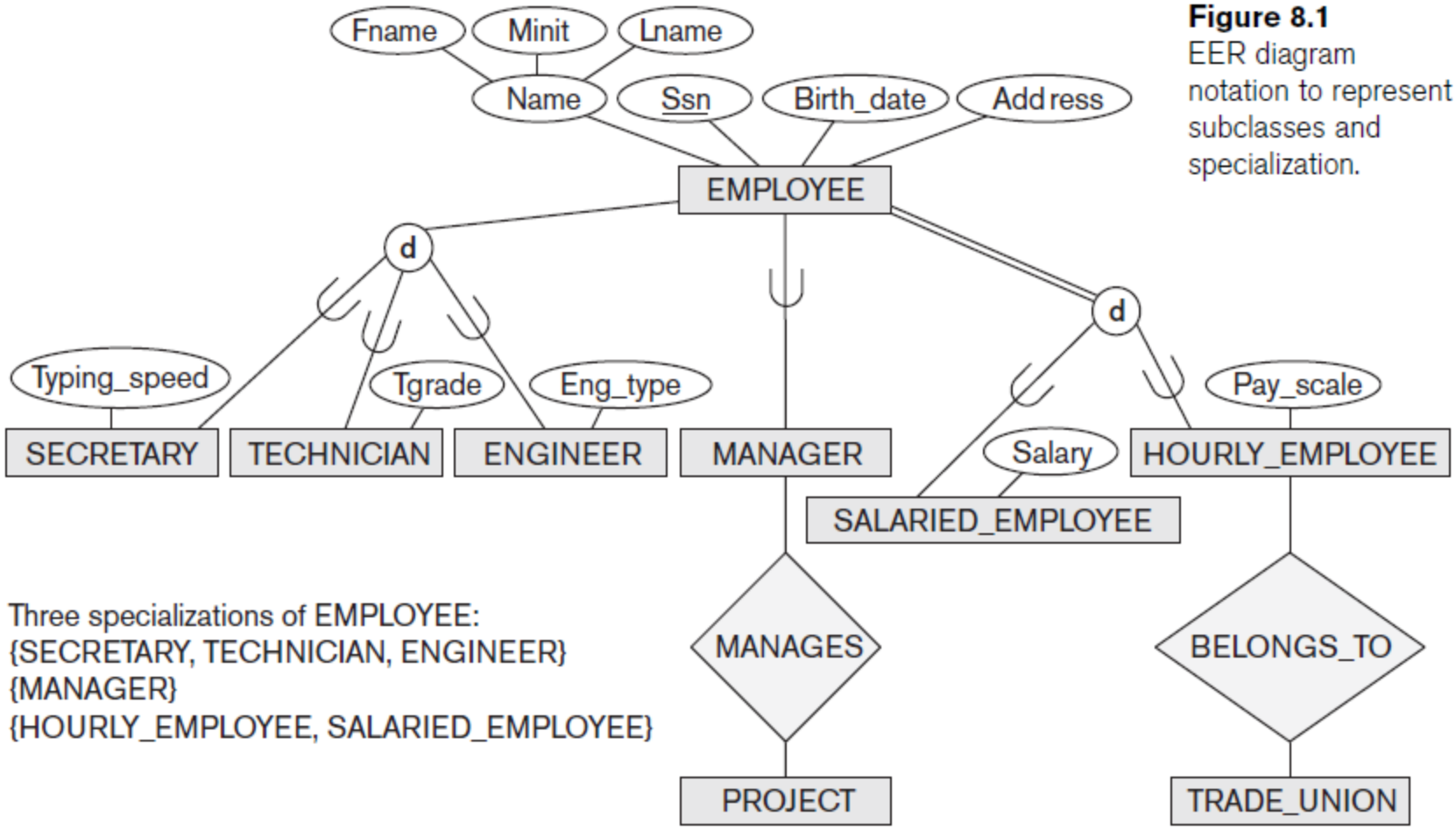


Figure 8.1
EER diagram notation to represent subclasses and specialization.

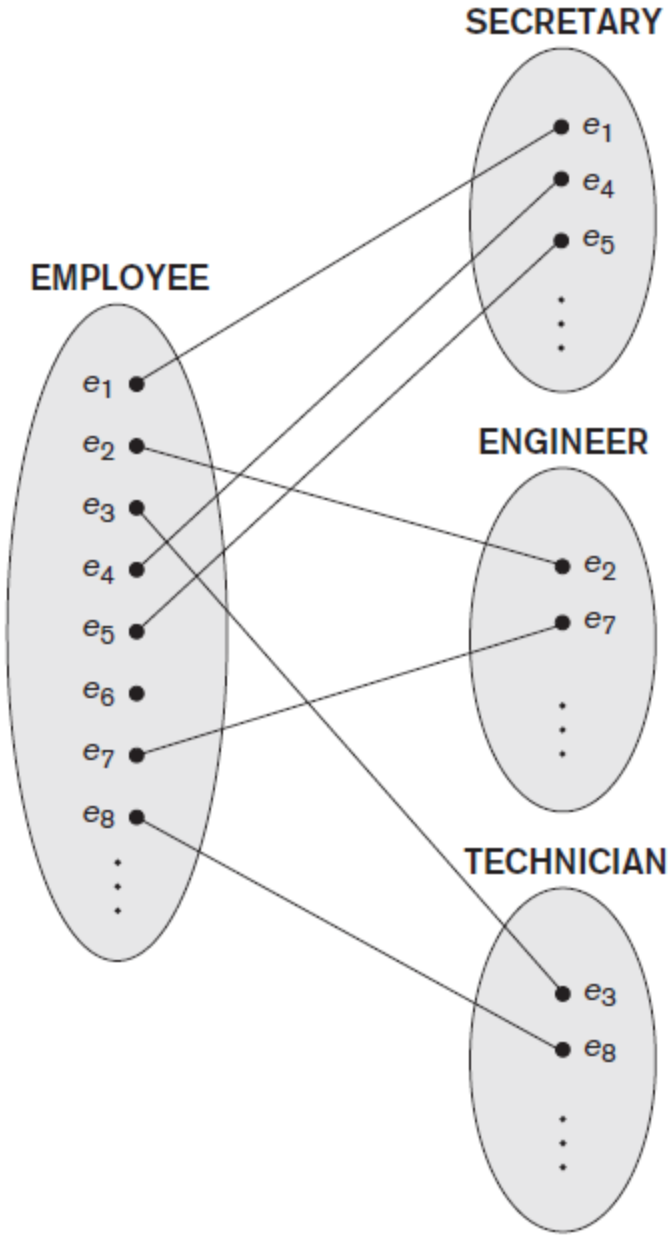


Figure 8.2
Instances of a specialization.

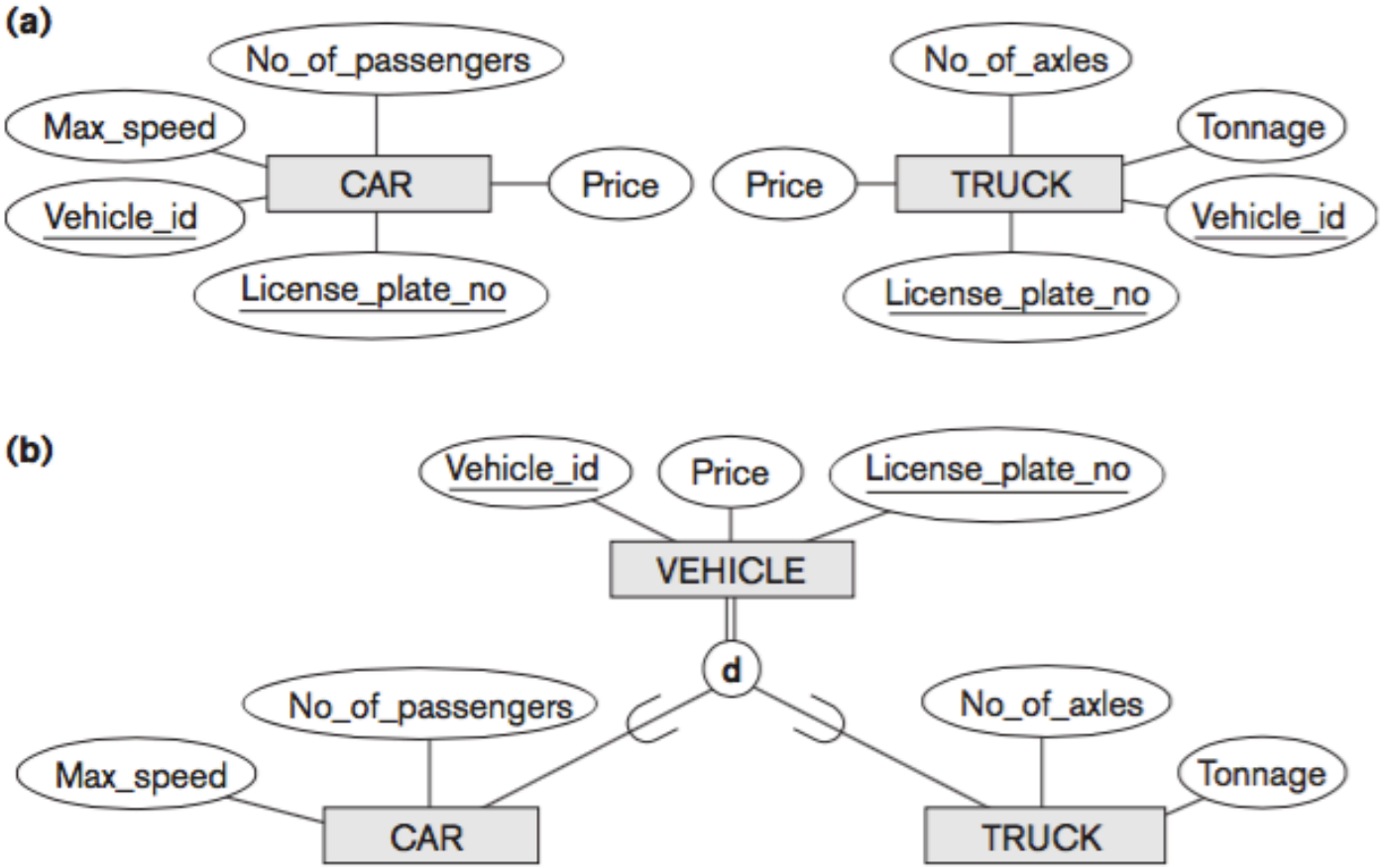
Specialization

- **Specialization**
  - Process of defining a set of subclasses of an entity type
  - Defined on the basis of some distinguishing characteristic of the entities in the superclass
  - Two main reasons for including specializations
    - Certain attributes may apply to some but not all entities
    - Some relationship types may be participated in only by entities that are members of the subclass
- Subclass can define:
  - **Specific (or local) attributes**
  - **Specific relationship types**

Generalization

- Reverse process of abstraction
- **Generalize** into a single **superclass**
  - Original entity types are special subclasses
- **Generalization**
  - Process of defining a generalized entity type from the given entity types

Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.



Constraints and Characteristics of Specialization and Generalization Hierarchies

- Constraints that apply to a single specialization or a single generalization
- Differences between specialization/ generalization lattices and hierarchies

Constraints on Specialization and Generalization

- Entities may have several specializations defined on the same entity type (or superclass)
  - Entities may belong to subclasses in each of the specializations
- Specializations may be several or one subclass
- Determine entity subtype:
  - **Predicate-defined** (or c**ondition-defined**) **subclasses**
  - **Attribute-defined specialization**
  - **User-defined**

Predicate-defined Specialization

- In some specializations we can determine exactly the entities that will become members of each subclass by placing a condition on the value of some attribute of the superclass
- For example, if the EMPLOYEE entity type has an attribute Job_type we can specify the condition of membership in the SECRETARY subclass by the condition (Job_type = 'Secretary'), which we call the **defining predicate** of the subclass
- We display a predicate-defined subclass by writing the predicate condition next to the line that connects the subclass to the specialization circle.

Attribute-defined Specialization

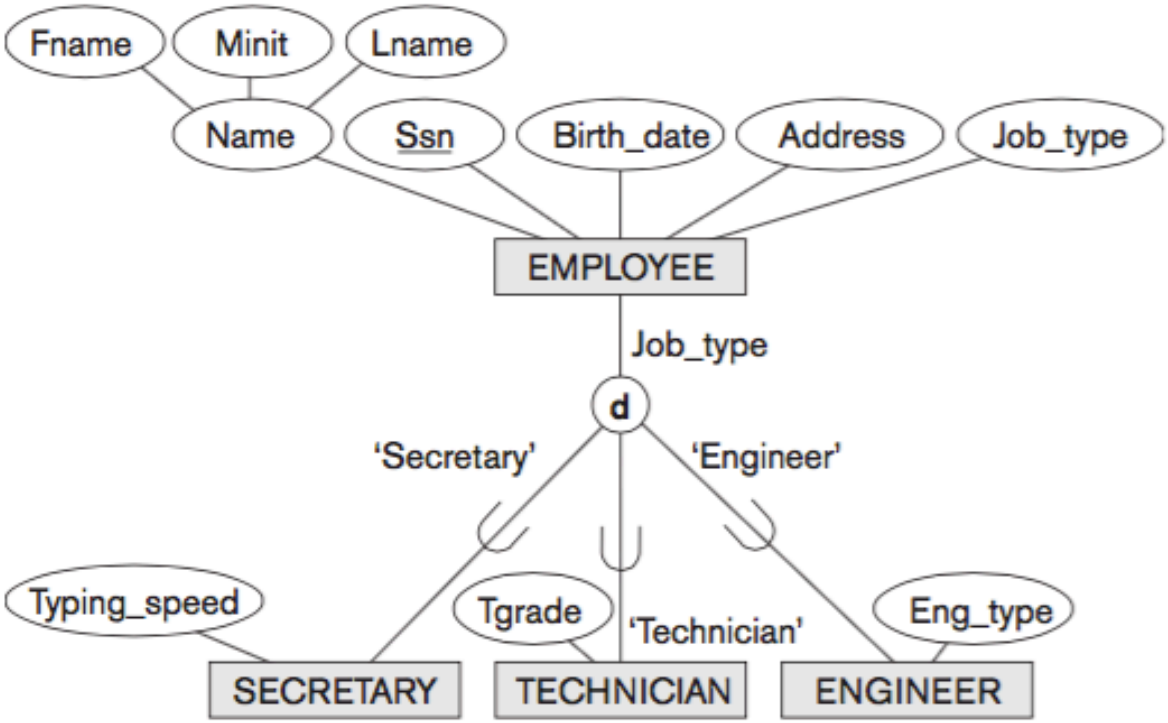


Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

User-defined Specialization

- When there is no condition for determining membership in a subclass
- Membership in such a subclass is determined by the database users when they apply the operation to add an entity to the subclass
- Membership is *specified individually for each entity by the user*, not by any condition that may be evaluated automatically.

Constraints on Specialization and Generalization (cont'd.)

- **Disjointness constraint**
  - If the subclasses of the specialization must be disjoint, their sets of entities must be **disjoint** (an entity can be a member of *at most* one of the subclasses of the specialization)
    - Indicated by a **d** inside the specialization circle
  - If the subclasses are not constrained to be disjoint, their sets of entities may be **overlapping** (the same entity may be a member of more than one subclass)
    - Indicated by an **o** inside the specialization circle

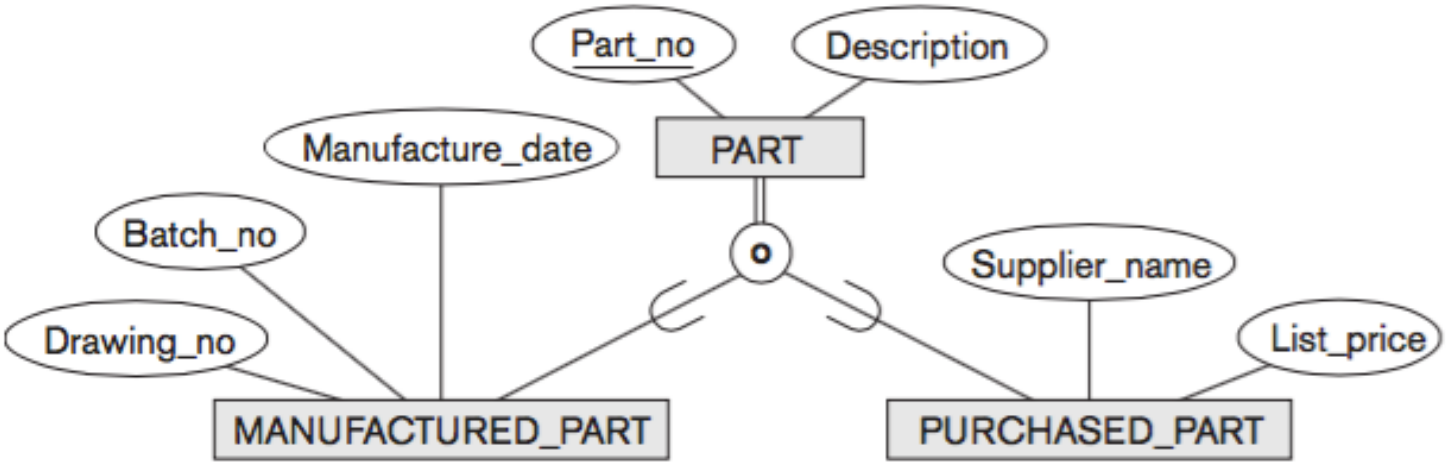


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Constraints on Specialization and Generalization (cont'd.)

- **Completeness** (or **totalness**) **constraint**
  - May be **total** or **partial**
  - **Total specialization** constraint specifies that *every* entity in the superclass must be a member of at least one subclass in the specialization.
    - Shown in EER diagrams by using a double line to connect the superclass to the circle
  - A **partial specialization** allows an entity not to belong to any of the subclasses
    - Shown in EER diagrams by using a single line to connect the superclass to the circle

Constraints on Specialization and Generalization (cont'd.)

- Disjointness and completeness constraints are independent
- Four possible constraints on a specialization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial
- The correct constraint is determined from the real-world meaning that applies to each specialization.
  - In general, a superclass that was identified through the *generalization* process usually is **total**

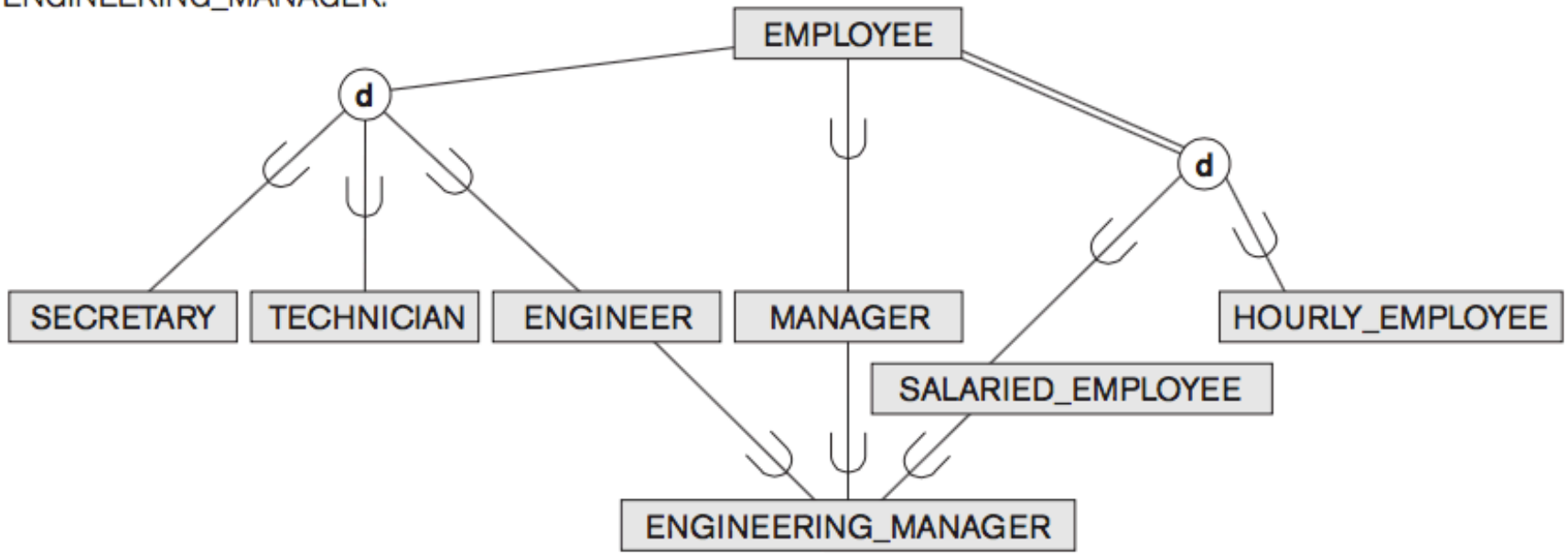
Constraints on Specialization and Generalization (cont'd.)

- Certain insertion and deletion rules apply to specialization (and generalization) as a consequence of the constraints specified earlier:
  - Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs.
  - Inserting an entity in a superclass implies that the entity is mandatorily inserted in all *predicate-defined* (or *attribute-defined*) subclasses for which the entity satisfies the defining predicate.
  - Inserting an entity in a superclass of a *total specialization* implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization.

Specialization and Generalization Hierarchies and Lattices

- A subclass may have further subclasses specified on it, forming a hierarchy or a lattice of specializations
- **Specialization hierarchy**
  - Every subclass participates as a subclass in only one class/subclass relationship
  - Results in a **tree structure** or **strict hierarchy**
- **Specialization lattice**
  - Subclass can be a subclass in more than one class/ subclass relationship

Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



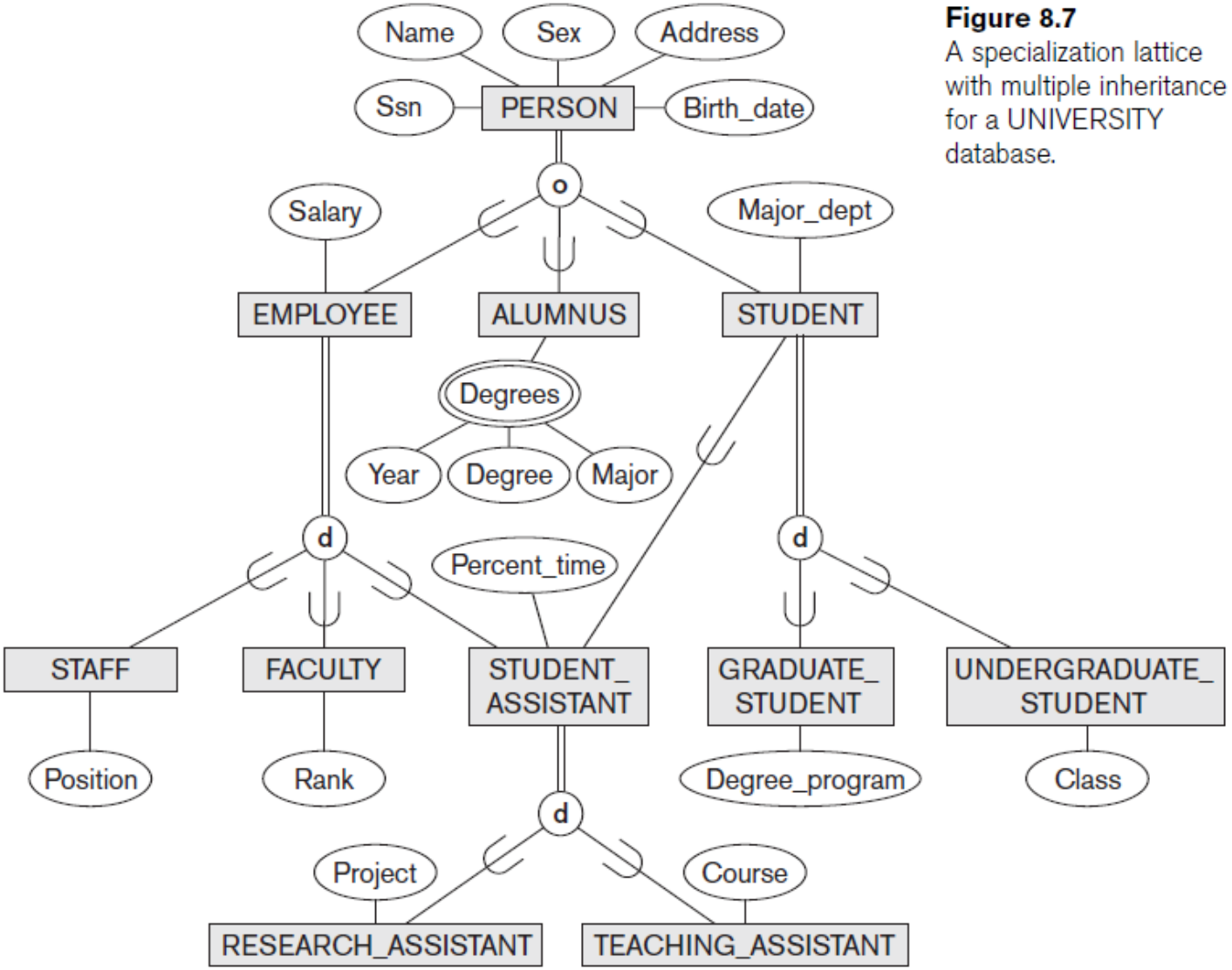


Figure 8.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

Specialization and Generalization Hierarchies and Lattices (cont'd.)

- **Multiple inheritance**
  - Subclass with more than one superclass
  - If attribute (or relationship) originating in the same superclass inherited more than once via different paths in lattice
    - Included only once in shared subclass
- **Single inheritance**
  - Some models and languages limited to single inheritance

Utilizing Specialization and Generalization in Refining Conceptual Schemas

- Specialization process
  - Start with entity type then define subclasses by successive specialization
  - **Top-down conceptual refinement process**
- **Bottom-up conceptual synthesis**
  - Involves generalization rather than specialization

Modeling of UNION Types Using Categories

- **Union type** or a **category**
  - Used when it's necessary to represent a collection of entities from different entity types
  - Represents a single superclass/subclass relationship with more than one superclass
  - Subclass represents a collection of objects that is a subset of the UNION of distinct entity types
  - Attribute inheritance works more selectively
  - Category can be **total** or **partial**
- Some modeling methodologies do not have union types

- Suppose that we have three entity types: PERSON, BANK, and COMPANY.
- In a database for motor vehicle registration, an owner of a vehicle can be a person, a bank, or a company.
- We need to create a class (collection of entities) that includes entities of all three types to play the role of *vehicle owner.*
- A category (union type) OWNER that is a *subclass of the UNION* of the three entity sets of COMPANY, BANK, and PERSON can be created for this purpose.
- A circle with the ∪ symbol stands for the *set union operation*

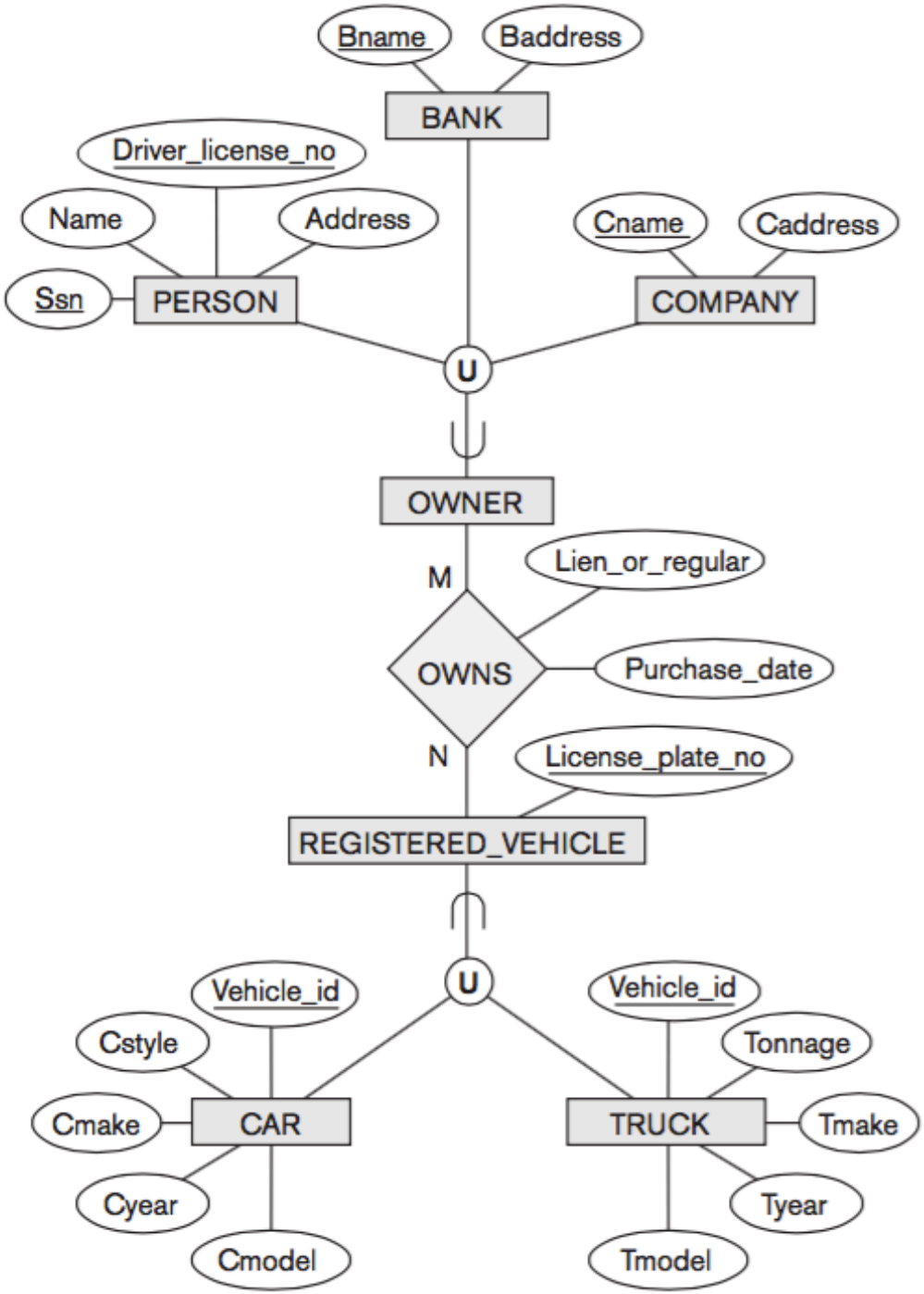


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Modeling of UNION Types Using Categories

- A category has two or more superclasses that may represent collections of entities from *distinct entity types,* whereas other superclass/subclass relationships always have a single superclass.
- Compare a category, such as OWNER with ENGINEERING_MANAGER.
  - E_M is a subclass of *each of* the three superclasses ENGINEER, MANAGER, and SALARIED_EMPLOYEE
    - An entity that is a member of E_M must exist in *all three collections.* An engineering manager must be an ENGINEER, a MANAGER, *and* a SALARIED_EMPLOYEE
  - A category is a subset of the *union* of its superclasses.
    - An entity that is a member of OWNER must exist in *only one* of the superclasses. An OWNER may be a COMPANY, a BANK, *or* a PERSON

Modeling of UNION Types Using Categories

- Attribute inheritance works more selectively in the case of categories.
  - For example, each OWNER entity inherits the attributes of a COMPANY, a PERSON, or a BANK, depending on the superclass to which the entity belongs.
  - On the other hand, a shared subclass such as ENGINEERING_MANAGER inherits *all* the attributes of its superclasses SALARIED_EMPLOYEE, ENGINEER, and MANAGER.

Modeling of UNION Types Using Categories

- A category can be **total** or **partial**.
  - A total category holds the *union* of all entities in its superclasses
    - Represented diagrammatically by a double line connecting the category and the circle
  - A partial category can hold a *subset of the union*
    - Indicated by a single line connecting the category and the circle
- If a category is total, it may be represented alternatively as a total specialization (or a total generalization).
  - The choice of which representation to use is subjective.
    - If the two classes represent the same type of entities and share numerous attributes, including the same key attributes, specialization/generalization is preferred
- Some modeling methodologies do not have union types

A Sample UNIVERSITY EER Schema, Design Choices, and Formal Definitions

- The UNIVERSITY Database Example
  - UNIVERSITY database
    - Students and their majors
    - Transcripts, and registration
    - University's course offerings
    - Sponsored research projects of faculty and graduate students

UNIVERSITY Schema

- For each person, the database maintains information on the person's Name [Name], Social Security number [Ssn], address [Address], sex [Sex], and birth date [Bdate].
- Two subclasses of the PERSON entity type are identified: FACULTY and STUDENT.
- Specific attributes of FACULTY are rank [Rank] (assistant, associate, adjunct, research, visiting, and so on), office [Foffice], office phone [Fphone], and salary [Salary]. All faculty members are related to the academic department(s) with which they are affiliated [BELONGS] (a faculty member can be associated with several departments, so the relationship is M:N).

UNIVERSITY Schema

- A specific attribute of STUDENT is [Class] (freshman = 1, sopho- more = 2, ... , MS student = 5, PhD student = 6). Each STUDENT is also related to his or her major and minor departments (if known) [MAJOR] and [MINOR], to the course sections he or she is currently attending [REGISTERED], and to the courses completed [TRANSCRIPT].
- Each TRANSCRIPT instance includes the grade the student received [Grade] in a section of a course.

UNIVERSITY Schema

- GRAD_STUDENT is a subclass of STUDENT, with the defining predicate (Class = 5 OR Class = 6). For each graduate student, we keep a list of previous degrees in a composite, multivalued attribute [Degrees]. We also relate the graduate student to a faculty advisor [ADVISOR] and to a thesis committee [COMMITTEE], if one exists.
- An academic department has the attributes name [Dname], telephone [Dphone], and office number [Office] and is related to the faculty member who is its chairperson [CHAIRS] and to the college to which it belongs [CD]. Each college has attributes college name [Cname], office number [Coffice], and the name of its dean [Dean].

UNIVERSITY Schema

- A course has attributes course number [C#], course name [Cname], and course description [Cdesc]. Several sections of each course are offered, with each section having the attributes section number [Sec#] and the year and quarter in which the section was offered ([Year] and [Qtr]).10 Section numbers uniquely identify each section. The sections being offered during the current quarter are in a subclass CURRENT_SECTION of SECTION, with the defining predicate Qtr = Current_qtr and Year = Current_year. Each section is related to the instructor who taught or is teach- ing it ([TEACH]), if that instructor is in the database.
- The category INSTRUCTOR_RESEARCHER is a subset of the union of FACULTY and GRAD_STUDENT and includes all faculty, as well as graduate students who are supported by teaching or research.

UNIVERSITY Schema

- Finally, the entity type GRANT keeps track of research grants and contracts awarded to the university. Each grant has attributes grant title [Title], grant number [No], the awarding agency [Agency], and the starting date [St_date]. A grant is related to one principal investigator [PI] and to all researchers it supports [SUPPORT].
- Each instance of support has as attributes the starting date of support [Start], the ending date of the support (if known) [End], and the percentage of time being spent on the project [Time] by the researcher being supported.

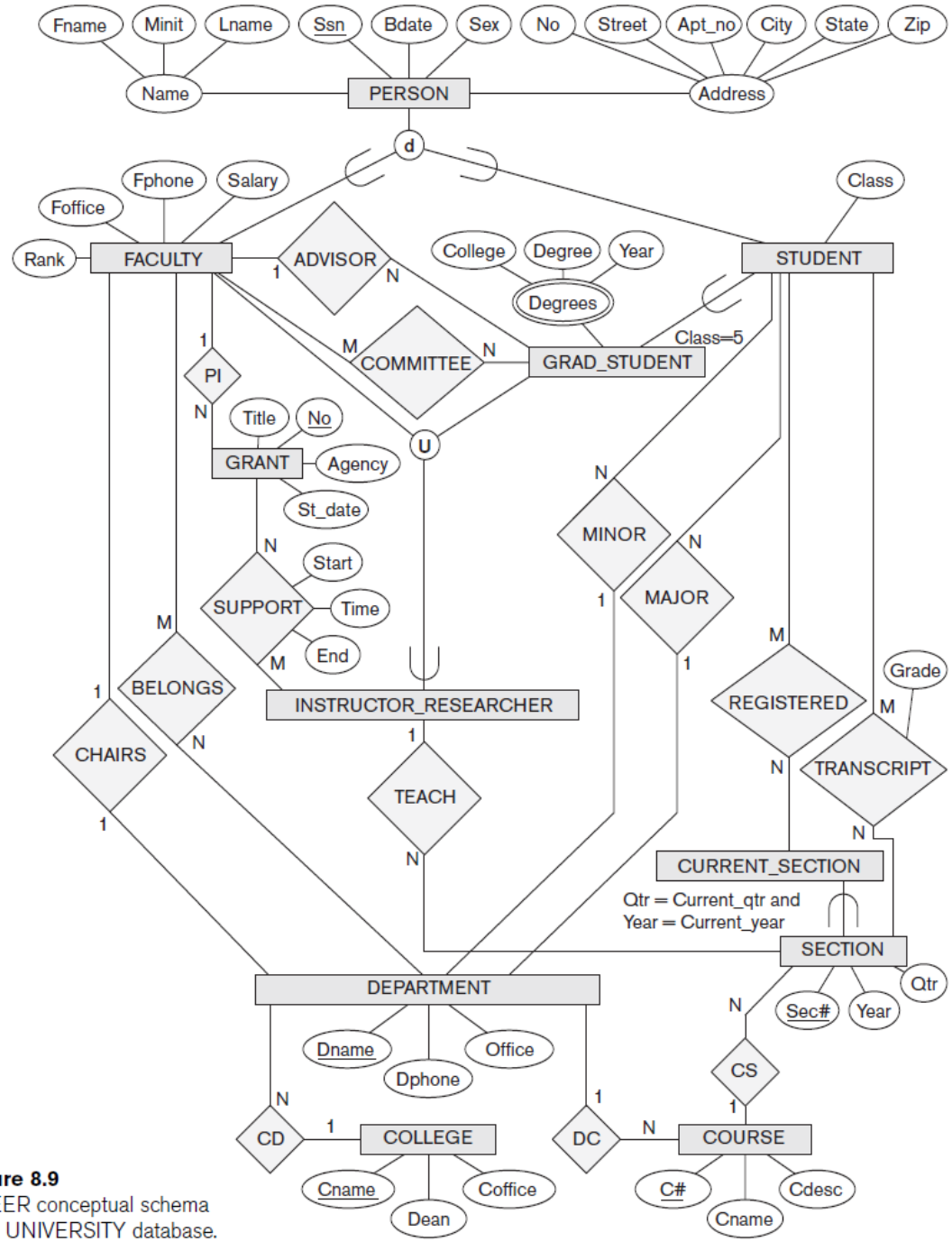


Figure 8.9
An EER conceptual schema for a UNIVERSITY database.

Design Choices for Specialization/Generalization

- Many specializations and subclasses can be defined to make the conceptual model accurate
  - The drawback is that the design becomes quite cluttered.
  - Represent only those subclasses that are deemed necessary
- If subclass has few specific attributes and no specific relationships
  - Can be merged into the superclass
  - The specific attributes would hold NULL values for entities that are not members of the subclass.
  - A *type* attribute could specify whether an entity is a member of the subclass.

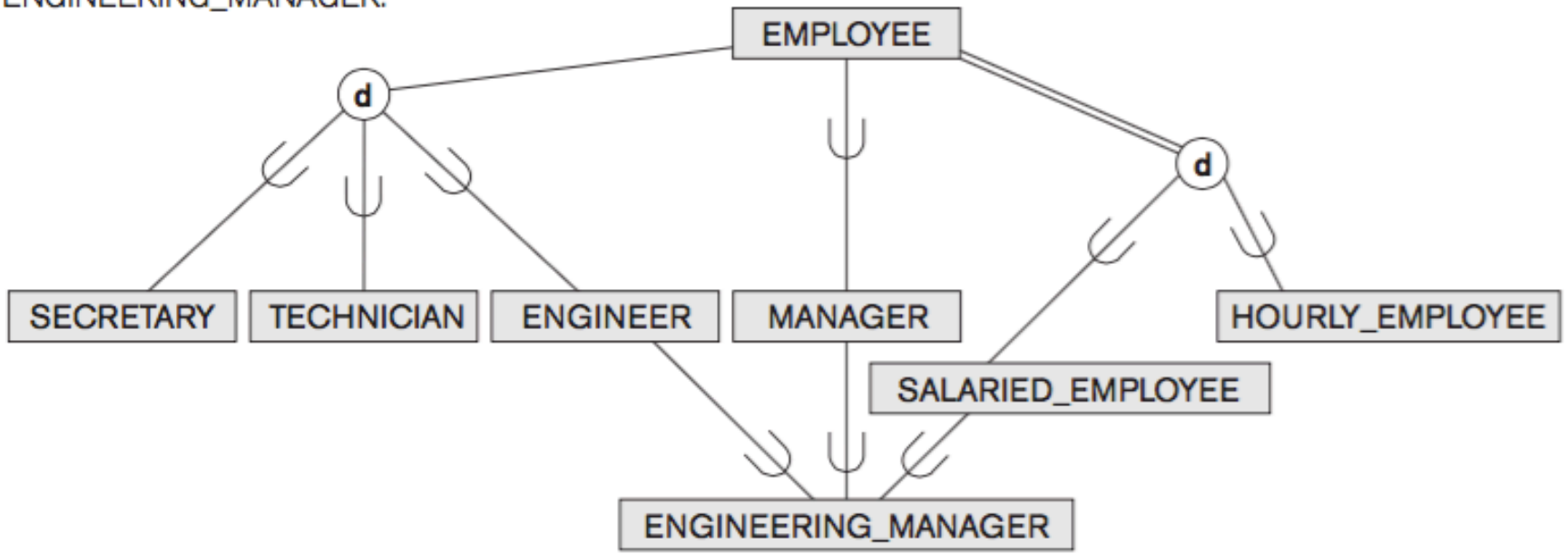
Design Choices for Specialization/Generalization (cont'd.)

- If all the subclasses of a specialization/ generalization have few specific attributes and no specific relationships
  - Can be merged into the superclass
  - Replace with one or more type attributes that specify the subclass or subclasses that each entity belongs to

Design Choices for Specialization/Generalization (cont'd.)

- Union types and categories should generally be avoided
- Choice of disjoint/overlapping and total/partial constraints on specialization/generalization
  - Driven by rules in miniworld being modeled
  - If the requirements do not indicate any particular constraints, the default would be overlapping and partial

Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



How Could We Apply These Guidelines?

- No specific (local) attributes are shown
- We could merge all the subclasses into the EMPLOYEE entity type and add the following attributes to EMPLOYEE:
  - An attribute Job_type whose value set {'Secretary', 'Engineer', 'Technician'} would indicate which subclass in the first specialization each employee belongs to.
  - An attribute Pay_method whose value set {'Salaried', 'Hourly'} would indicate which subclass in the second specialization each employee belongs to.
  - An attribute Is_a_manager whose value set {'Yes', 'No'} would indicate whether an individual employee entity is a manager or not.

Formal Definitions for the EER Model Concepts

- **Class**
  - Set or collection of entities
  - Includes any of the EER schema constructs of group entities
- **Subclass**
  - Class whose entities must always be a subset of the entities in another class
  - A **subclass** *S* is a class whose entities must always be a subset of the entities in another class, called the **superclass** *C* of the **superclass/subclass** (or **IS-A**) **relationship**
  - *C/S*

Formal Definitions for the EER Model Concepts

- **Specialization**
  - Set of subclasses that have same superclass
  - A **specialization** $Z = \{S1, S2, \ldots, Sn\}$ is a set of subclasses that have the same super-class G
  - G/Si is a superclass/subclass relationship for $i = 1, 2, \ldots, n$
  - Z is said to be **total** if: $\bigcup_{i=1}^{n} S_i = G$
    - Z is partial otherwise
  - Z is said to be **disjoint** if: $S_i \bigcap S_j = \varnothing$
    - Otherwise Z is overlapping

Formal Definitions for the EER Model Concepts (cont'd.)

- **Generalization**
  - Generalized entity type or superclass
- **Predicate-defined**
  - Predicate on the attributes of is used to specify which entities in *C* are members of *S*
- **User-defined**
  - Subclass that is not defined by a predicate

Formal Definitions for the EER Model Concepts (cont'd.)

- **Category**
  - Class that is a subset of the union of n defining superclasses
  - A **category** *T* is a class that is a subset of the union of *n* defining superclasses *D*1, *D*2, ... , *Dn*, *n* > 1
- **Relationship type**
  - Any class can participate in a relationship

Example of Other Notation

- Representing specialization and generalization in UML class diagrams
  - Basic notation
    - Connect the subclasses by vertical lines to a triangle connecting to the superclass.
    - A blank triangle indicates a specialization/generalization with the *disjoint* constraint
    - A filled triangle indicates an *overlapping* constraint
  - Base class
    - Root superclass
  - Leaf classes
    - Subclasses (leaf nodes)

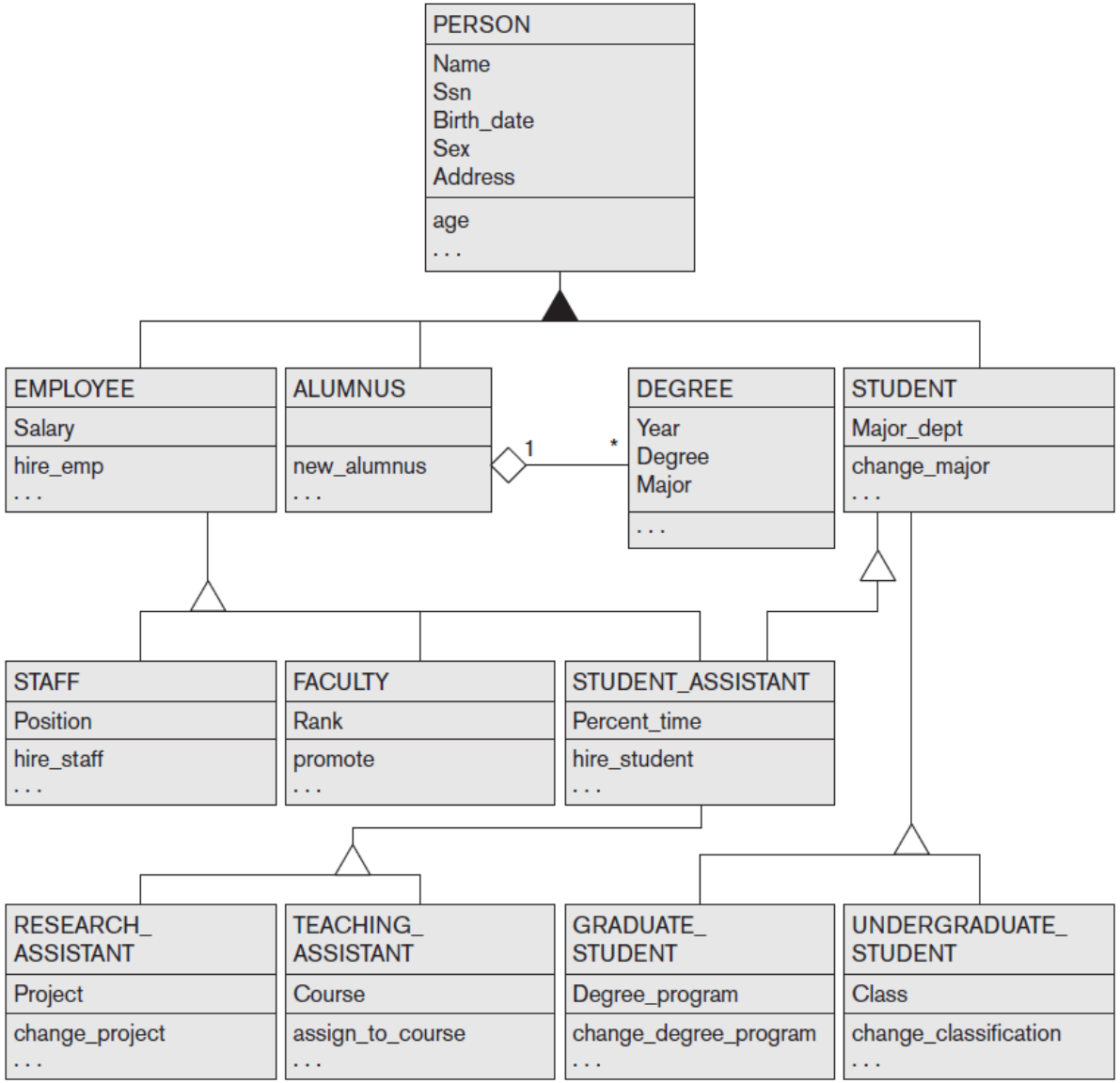


Figure 8.10
A UML class diagram corresponding to the EER diagram in Figure 8.7, illustrating UML notation for specialization/generalization.

Data Abstraction, Knowledge Representation, and Ontology Concepts

- Goal of **knowledge representation (KR)** techniques
  - Accurately model some **domain of knowledge**
  - Create an **ontology** that describes the concepts of the domain and how these concepts are interrelated
- Goals of KR are similar to those of semantic data models
  - Important similarities and differences

CD Vs. KR

- Both disciplines use an abstraction process to identify common properties and important aspects of objects in the miniworld (also known as *domain of discourse* in KR) while suppressing insignificant differences and unimportant details.
- Both disciplines provide concepts, relationships, constraints, operations, and languages for defining data and representing knowledge.
- KR is generally broader in scope than semantic data models. Different forms of knowledge, such as rules (used in inference, deduction, and search), incomplete and default knowledge, and temporal and spatial knowledge, are represented in KR schemes.

CD Vs. KR

- KR schemes include **reasoning mechanisms** that deduce additional facts from the facts stored in a database. Hence, whereas most current database systems are limited to answering direct queries, knowledge-based systems using KR schemes can answer queries that involve **inferences** over the stored data.
- Four **abstraction concepts** are used in semantic data models, such as the EER model, as well as in KR schemes: (1) classification and instantiation, (2) identification, (3) specialization and generalization, and (4) aggregation and association.

Classification and Instantiation

- **Classification**
  - Systematically assigning similar objects/entities to object classes/entity types
- **Instantiation**
  - Inverse of classification
  - Generation and specific examination of distinct objects of a class

Classification and Instantiation (cont'd.)

- **Exception objects**
  - Differ in some respects from other objects of class
  - KR schemes allow such **class properties**
- **Class properties**
  - Certain properties apply to the class as a whole and not to the individual objects
  - KR schemes allow such **class properties**.
- One class can be an instance of another class (called a meta-class)
  - Cannot be represented directly in EER model

Identification

- Abstraction process
- Classes and objects are made uniquely identifiable by means of some **identifier**
- Needed at two levels
  - To distinguish among database objects and classes
  - To identify database objects and to relate them to their real-world counterparts

Specialization and Generalization

- **Specialization**
  - Classify a class of objects into more specialized subclasses
- **Generalization**
  - Generalize several classes into a higher-level abstract class
  - Includes the objects in all these classes

Aggregation and Association

- **Aggregation**
  - Abstraction concept for building composite objects from their component objects
- **Association**
  - Associate objects from several independent classes
- Main structural distinction
  - When an association instance is deleted
    - Participating objects may continue to exist

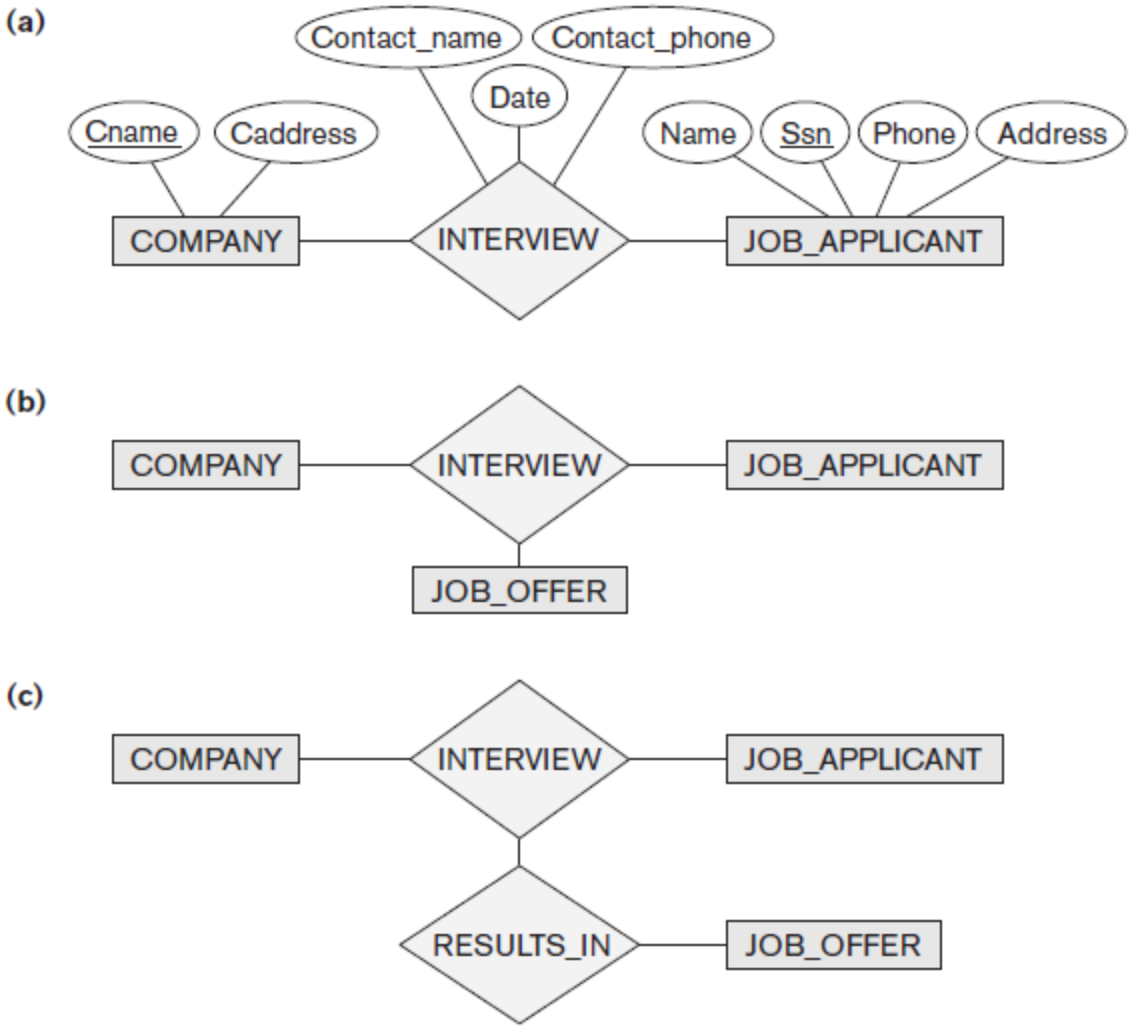


Figure 8.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER but allowed by some modeling tools). (e) Correct representation in ER.

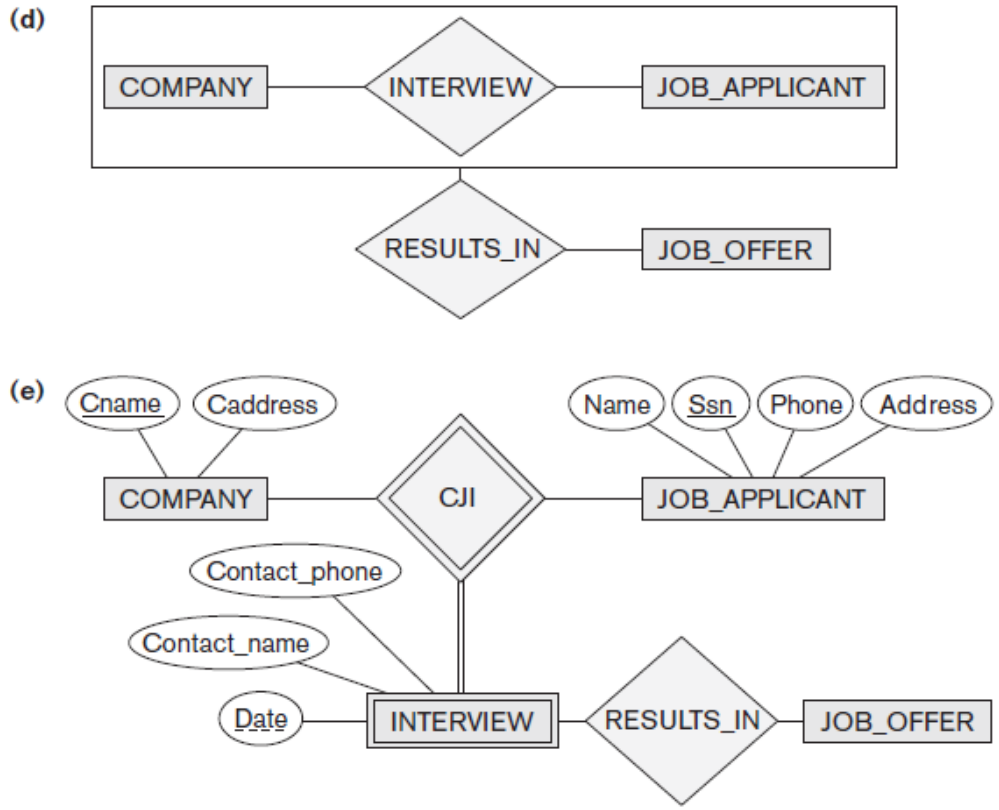


Figure 8.11
Aggregation. (a) The relationship type INTERVIEW. (b) Including JOB_OFFER in a ternary relationship type (incorrect). (c) Having the RESULTS_IN relationship participate in other relationships (not allowed in ER). (d) Using aggregation and a composite (molecular) object (generally not allowed in ER but allowed by some modeling tools). (e) Correct representation in ER.

Ontologies and the Semantic Web

- Documents contain less structure than database information does
- **Semantic Web**
  - Allow meaningful information exchange and search among machines
- **Ontology**
  - Specification of a **conceptualization**
- **Specification**
  - Language and vocabulary terms used to specify conceptualization

Summary

- Enhanced ER or EER model
  - Extensions to ER model that improve its representational capabilities
  - Subclass and its superclass
  - Category or union type
- Notation and terminology of UML for representing specialization and generalization