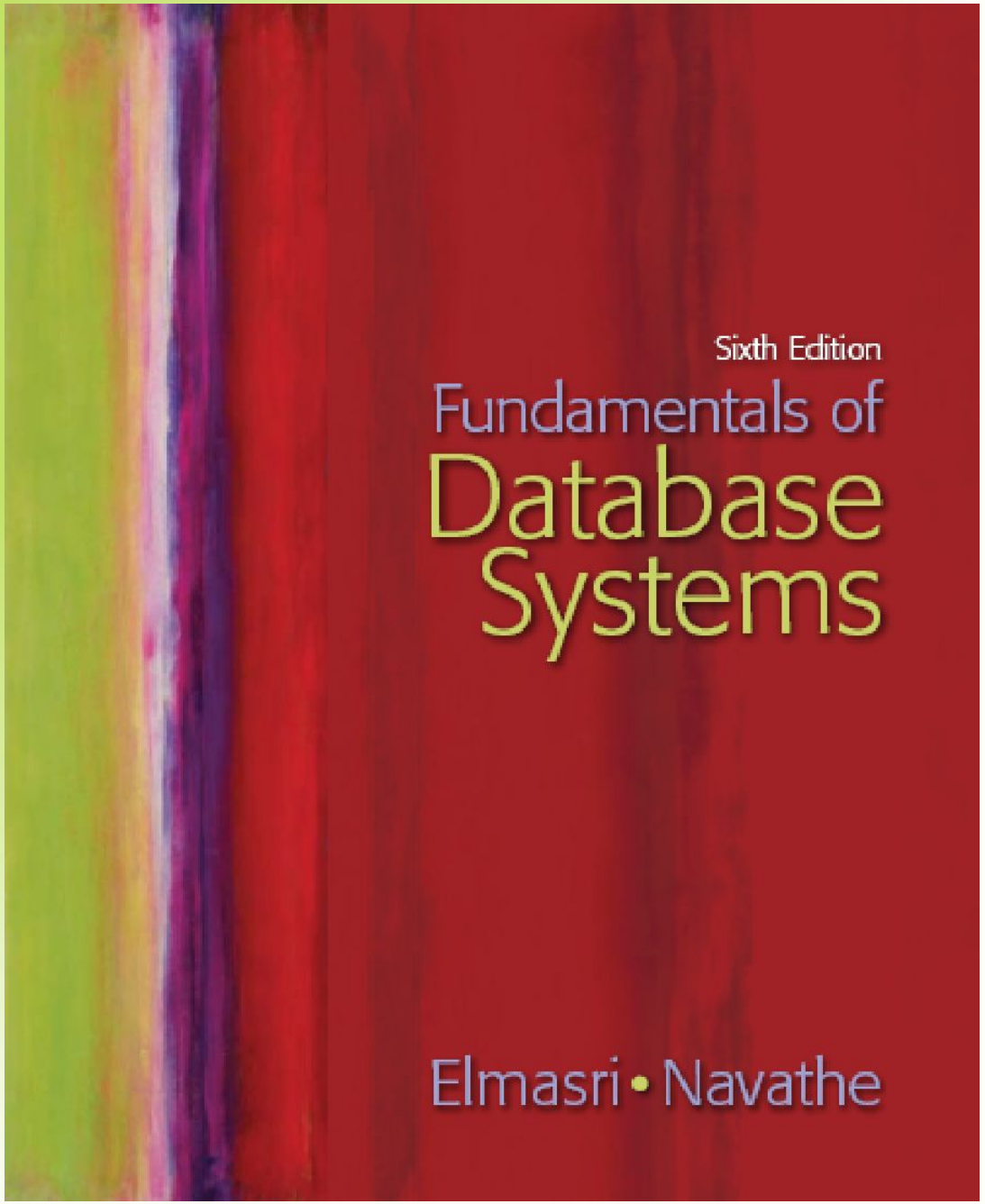
Chapter 8

The Enhanced Entry-Relationship (EER) Model



Addison-Wesley
is an imprint of
PEARSON

Chapter 8

The Enhanced Entity-Relationship (EER) Model

Addison-Wesley
is an imprint of
PEARSON

Chapter 8 Outline

- EER stands for Enhanced ER or Extended ER (sometimes referred to as E^2R)
- EER Model Concepts
  - Includes all modeling concepts of basic ER (Chapter 7)
  - Additional concepts:
    - subclasses/superclasses
    - specialization/generalization
    - categories (UNION types)
    - attribute and relationship inheritance
  - These are fundamental to conceptual modeling
- The additional EER concepts are used to model applications more completely and more accurately
  - EER includes some object-oriented concepts, such as inheritance
  - Also known as IS-A relationships in knowledge representation field of Artificial Intelligence

Subclasses and Superclasses

- An entity type may have additional meaningful subtypes (or specializations) of its entities
  - Example: EMPLOYEE may be further specialized into:
    - SECRETARY, ENGINEER, TECHNICIAN, …
      - Based on the EMPLOYEE's Job
    - MANAGER
      - EMPLOYEEs who are managers
    - SALARIED_EMPLOYEE, HOURLY_EMPLOYEE
      - Based on the EMPLOYEE's method of pay

Subclasses and Superclasses (cont.)

- EER diagrams extend ER diagrams to represent these additional subgroupings, called *subclasses* or *subtypes* (see Figure 8.1, next slide)
- Important Note: As with ER diagrams, there are many different diagrammatic notations for each concept
- We show some alternatives in Appendix A
- UML class diagrams notation for subclasses is presented later in this chapter

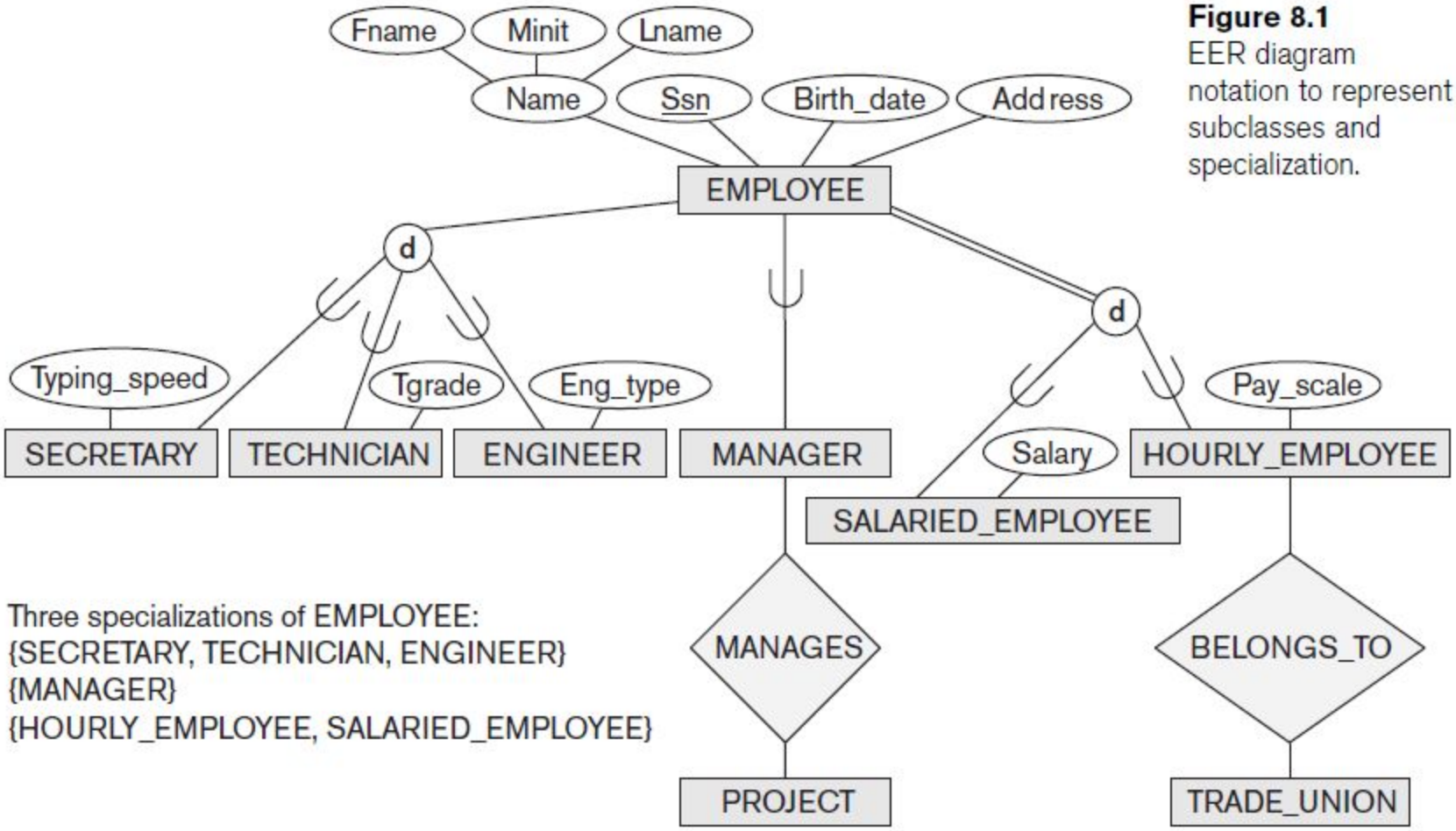


Figure 8.1
EER diagram notation to represent subclasses and specialization.

Subclasses and Superclasses (cont.)

- Each of these subgroupings (ENGINEER, TECHNICIAN, MANAGER, SALARIED_EMPLOYEE, etc.) will hold a subset of EMPLOYEE entities
- Each is called a **subclass** of EMPLOYEE
- EMPLOYEE is called the **superclass**
- The relationships are called superclass/subclass relationships:
  - EMPLOYEE/SECRETARY
  - EMPLOYEE/TECHNICIAN
  - EMPLOYEE/MANAGER
  - …

Subclasses and Superclasses (cont.)

- These are also called **IS-A relationships**
  - Based on Knowledge Representation terminology in Artificial Intelligence field
  - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ….
- Note: An entity that is member of a subclass represents *the same real-world entity* as some member of the superclass:
  - The subclass member is the same entity in a *distinct specific role*
  - Entity cannot exist in database merely by being a member of a subclass; it must also be a member of the superclass
  - A member of the superclass can be optionally included as a member of any number (zero or more) of its subclasses

Subclasses and Superclasses (cont.)

- Examples:
  - A salaried employee who is also an engineer belongs to the two subclasses:
    - ENGINEER, and
    - SALARIED_EMPLOYEE
  - A salaried employee who is also an engineering manager belongs to the three subclasses:
    - MANAGER,
    - ENGINEER, and
    - SALARIED_EMPLOYEE
- It is not necessary that every entity in a superclass be a member of some subclass

Attribute Inheritance in Superclass/ Subclass Relationships

- An entity that is member of a subclass *inherits*
  - All attributes of the entity as a member of the superclass
  - All relationships of the entity as a member of the superclass
- Example (Figure 8.1):
  - SECRETARY (as well as TECHNICIAN, MANAGER, ENGINEER, etc.) inherit the attributes Name, SSN, …, from EMPLOYEE
  - Every SECRETARY entity will have values for the inherited attributes

Specialization

- Is the process of defining a set of subclasses of a superclass – the set must be 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
    - {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE} is another specialization of EMPLOYEE based on *method of pay*

Specialization (cont.)

- Specialization can be diagrammatically represented in EER diagrams as (see Figure 8.1, repeated in next slide))
  - The subclasses are connected to a circle that represents the specialization (using lines *with the subset symbol*)
  - The circle is also connected to the superclass
  - Attributes of a subclass are called *specific* or *local* attributes.
    - For example, the attribute TypingSpeed of SECRETARY
  - The subclass can also participate in specific relationship types.
    - For example, a relationship BELONGS_TO of HOURLY_EMPLOYEE (see Figure 8.1)

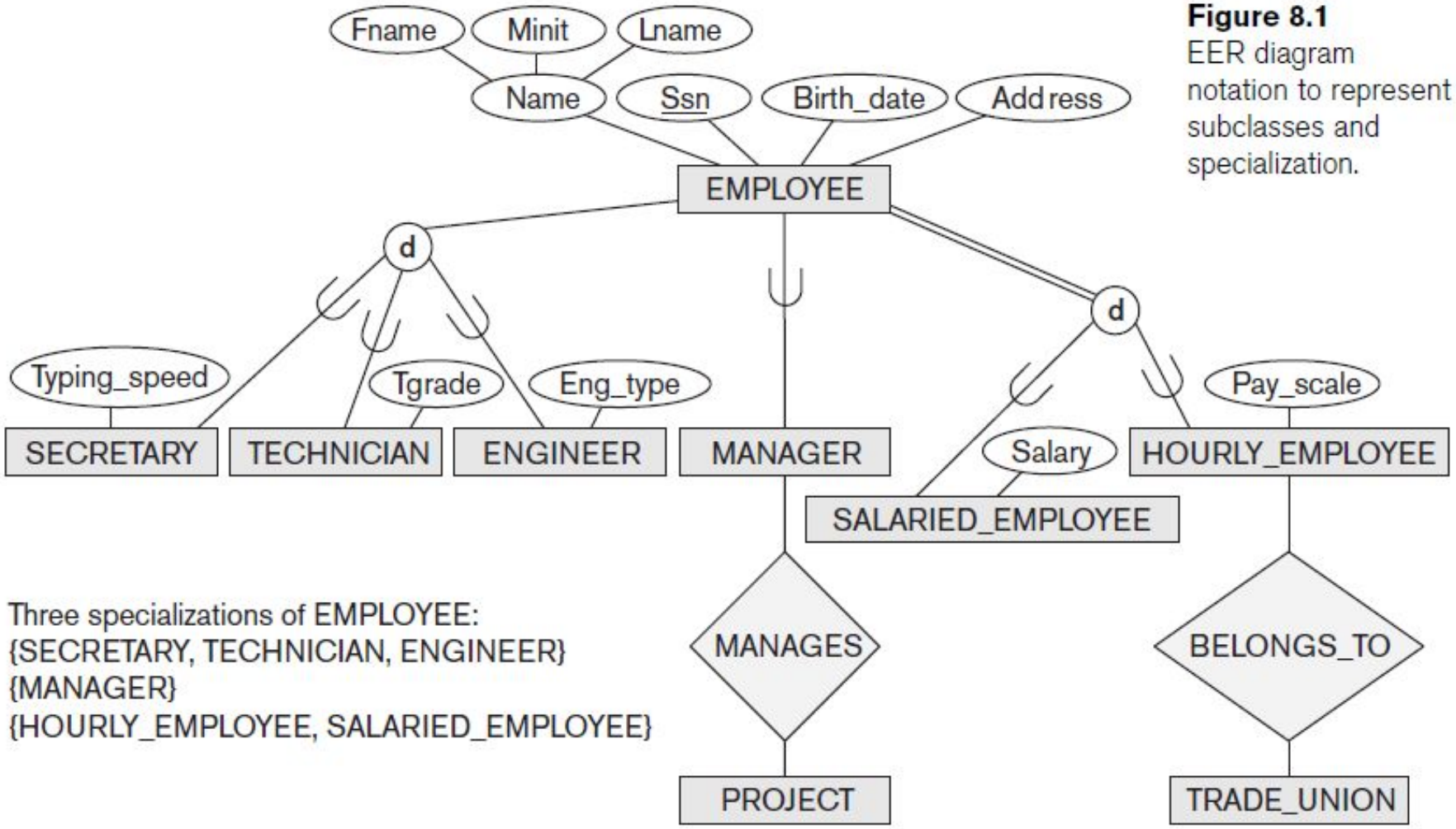


Figure 8.1
EER diagram notation to represent subclasses and specialization.

Generalization

- Generalization is the reverse of the specialization process
- Several classes with common features are **generalized** into a superclass;
  - original classes become its subclasses
- Example (Figure 8.3, next slide): CAR, TRUCK generalized into VEHICLE;
  - Both CAR, TRUCK become subclasses of the superclass VEHICLE because they have *several common attributes*.
  - VEHICLE includes the common attributes
  - Can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

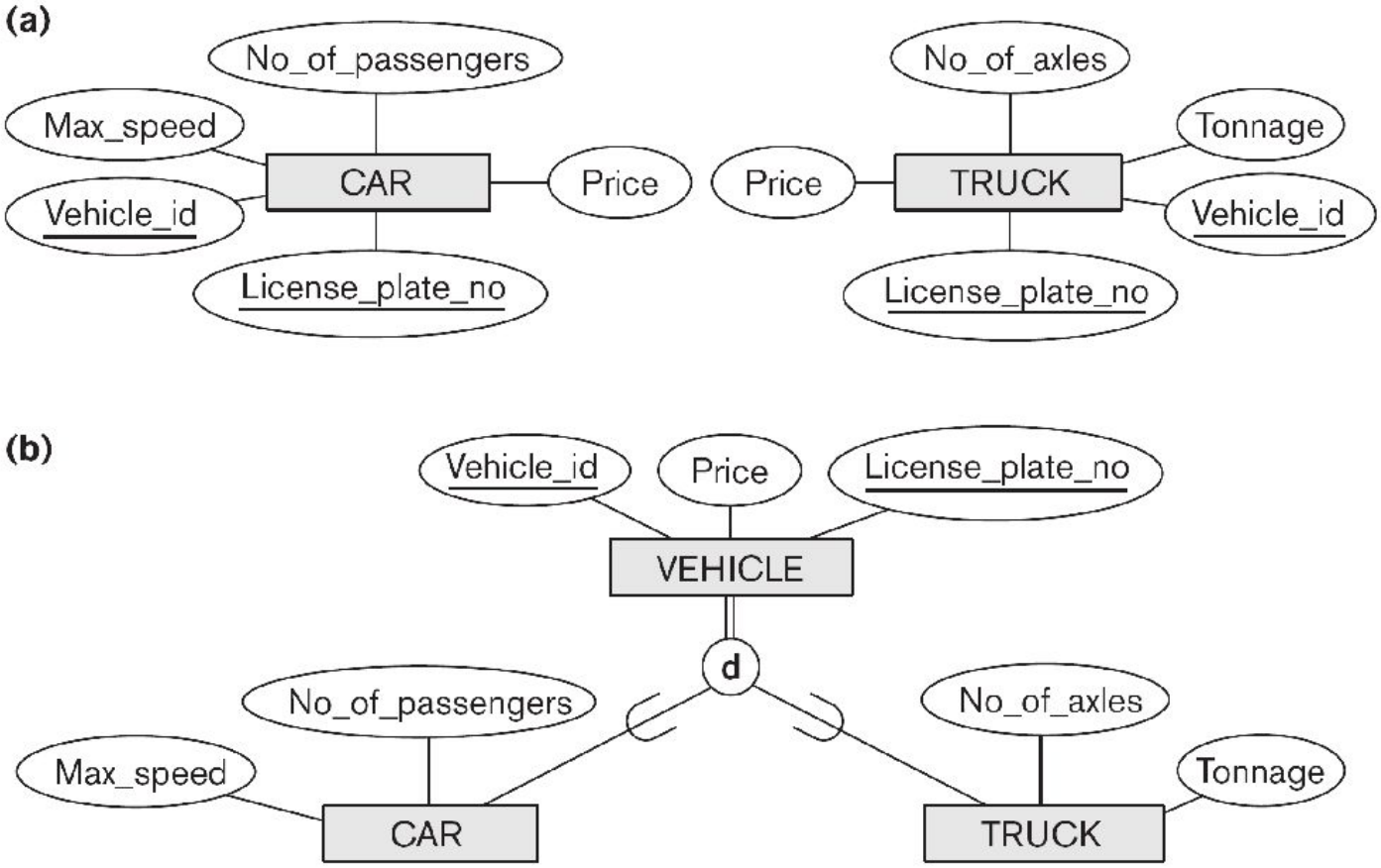


Figure 8.3
Generalization. (a) Two entity types, CAR and TRUCK. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Generalization vs. Specialization

- Diagrammatic notations sometimes distinguish generalization and specialization
  - Arrow pointing to the generalized superclass represents a generalization
  - Arrows pointing to the specialized subclasses represent a specialization
  - We *do not use* this notation because it is often unclear or subjective as to which process was used to reach the final design

Generalization vs. Specialization (cont.)

- Data Modeling with Specialization and Generalization
  - A superclass or subclass represents a *type of entity*, as well as the collection (or set or grouping) of entities of that type
  - Subclasses and superclasses are displayed in rectangles in EER diagrams (like entity types)
  - We can call all entity types ***classes***, whether they are entity types, superclasses, or subclasses (object-oriented terminology)

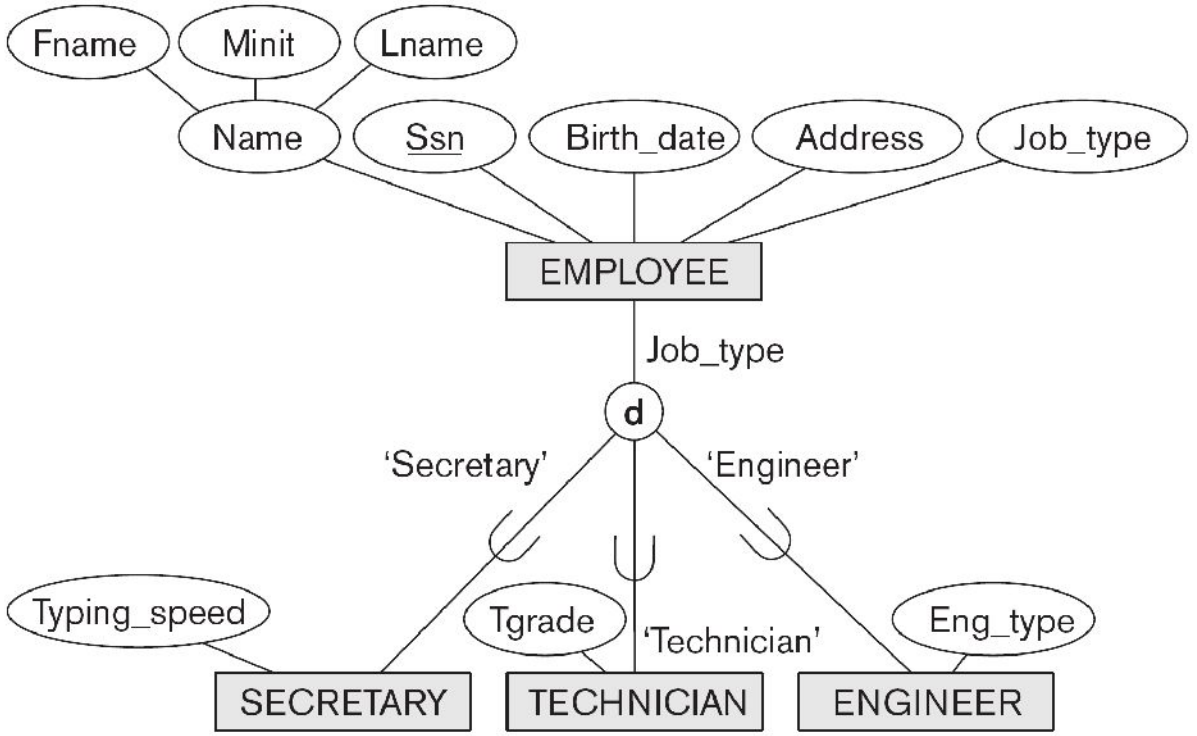
Constraints on Specialization and Generalization

- Two basic constraints can apply to a specialization/generalization:
  - Disjointness Constraint: **d** (disjoint) vs. **o** **(**overlapping)
  - Completeness Constraint: **total** (double line to superclass) vs. **partial** (single line)
  - Default is overlapping, partial
  - Decision on which constraint to choose is based on situation being modeled in mini-world

Constraints on Specialization and Generalization (cont.)

- Disjointness Constraint:
  - Specifies that the subclasses of the specialization must be *disjoint*:
    - an entity can be a member of *at most one* of the subclasses of the specialization
  - Specified by ***d*** in EER diagram (Figure 8.4)
  - If not disjoint, specialization is *overlapping*:
    - same entity may be a member of *more than one* subclass of the specialization
  - Specified by ***o*** in EER diagram (Figure 8.5)

Figure 8.4
EER diagram notation for an attribute-defined specialization on Job_type.



[6]Such an attribute is called a *discriminator* in UML terminology.

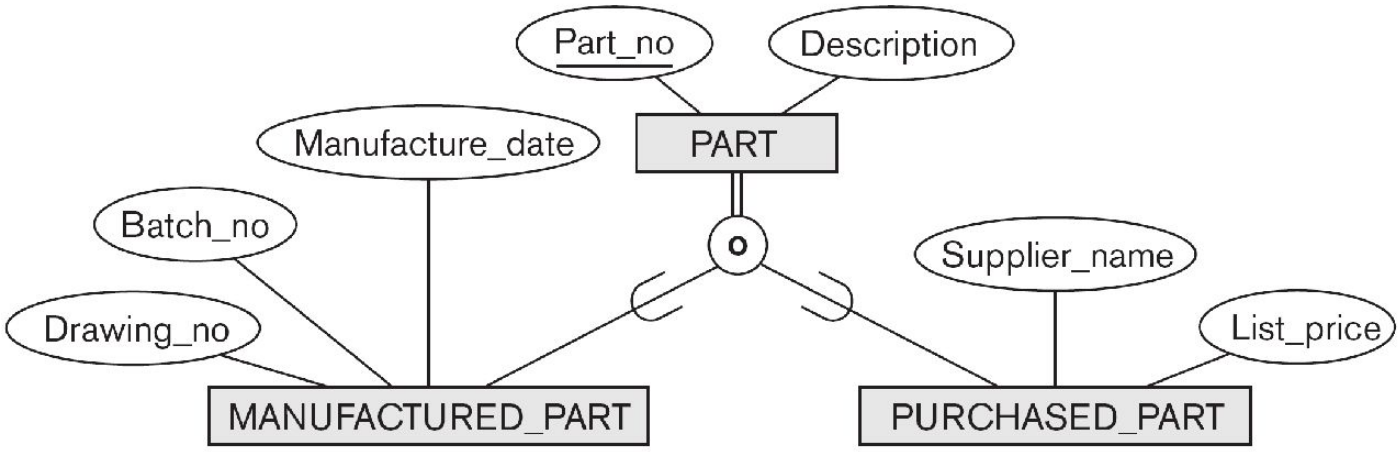


Figure 8.5
EER diagram notation for an overlapping (nondisjoint) specialization.

[7]The notation of using single or double lines is similar to that for partial or total participation of an entity type in a relationship type, as described in Chapter 7.

Constraints on Specialization and Generalization (cont.)

- Completeness Constraint:
  - ***Total*** specifies that every entity in the superclass *must be a member of some (at least one) subclass*
  - Shown in EER diagrams by a ***double line*** connected to the superclass (Figure 8.5)
  - ***Partial*** allows an entity not to belong to any of the subclasses
  - Shown by a ***single line*** (Figure 8.4)

Constraints on Specialization and Generalization (cont.)

- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial
- Note: Generalization *usually is total* because the superclass is derived from the subclasses.

Additional Constraints on Specialization and Generalization

- If a boolean condition (predicate) can determine exactly those entities that will become members of a subclass,it is called **predicate-defined** (or condition-defined) subclass:
  - Condition is like a *constraint* that determines subclass members
  - Can display the predicate condition next to the line attaching the subclass

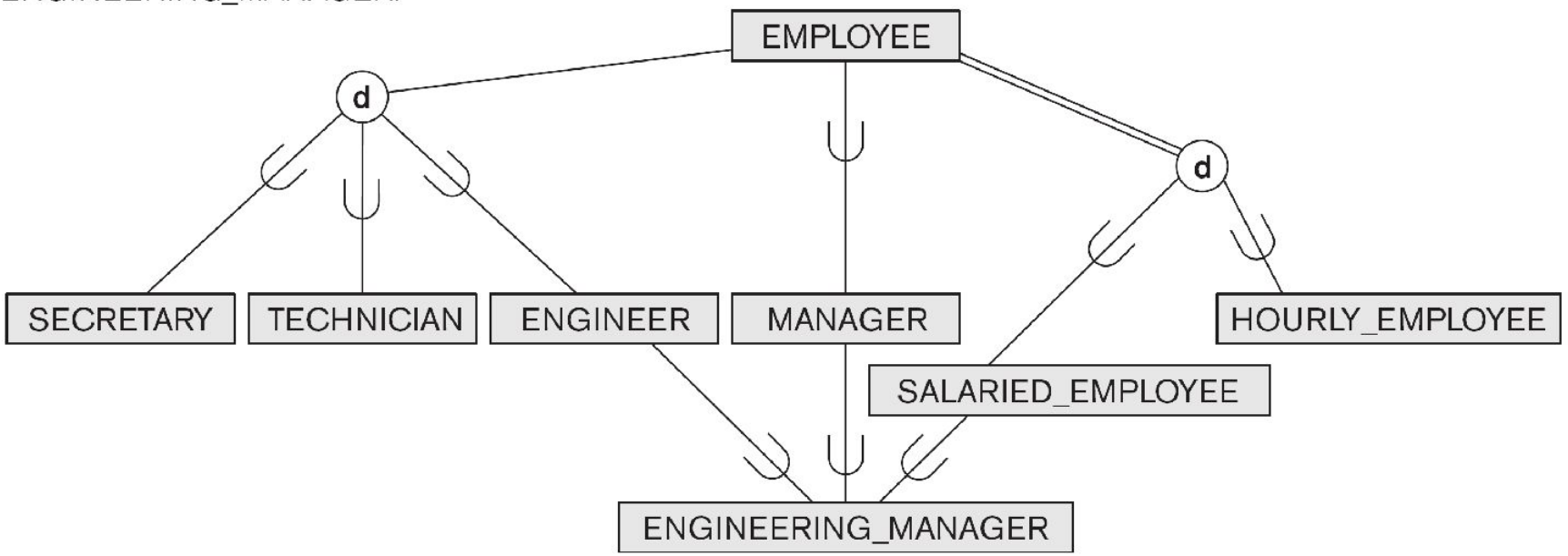
Additional Constraints on Specialization and Generalization (cont.)

- If one attribute defines conditions for all subclasses in a specialization, it is called **attribute-defined specialization**:
  - Attribute is called the *defining attribute* of the specialization
  - Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE (see Figure 8.4)
- If no condition determines membership, the subclass is called **user-defined specialization**:
  - Membership in a subclass is determined by the *database users* by explicitly adding an entity to a subclass

Specialization/Generalization Hierarchies, Lattices and Shared Subclasses

- A subclass may itself have further subclasses specified on it:
  - forms a hierarchy or a lattice
- ***Hierarchy*** has a constraint that *every subclass has only one superclass* (called ***single inheritance***); this is basically a ***tree structure***
- In a ***lattice***, a subclass can be subclass of more than one superclass (called ***multiple inheritance***) (see Figure 8.6, next slide)

Figure 8.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



Specialization/Generalization Hierarchies, Lattices and Shared Subclasses (cont.)

- In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its *ancestor superclasses*:
  - all the way to the root class
- Can have:
  - *specialization* hierarchies or lattices, or
  - *generalization* hierarchies or lattices,
  - depending on how they were *derived*
- In general, can just use the term *specialization* (to stand for the end result of either specialization or generalization)

Example of Specialization

- Figure 8.7 (next slide) shows an example specialization of different types of PERSONs in a UNIVERSITY database
  - STUDENT_ASSISTANT is the only shared subclass
  - Note: A shared subclass inherits attributes *only once* from a common ancestor; in Figure 8.7, STUDENT_ASSISTANT inherits PERSON attributes only once

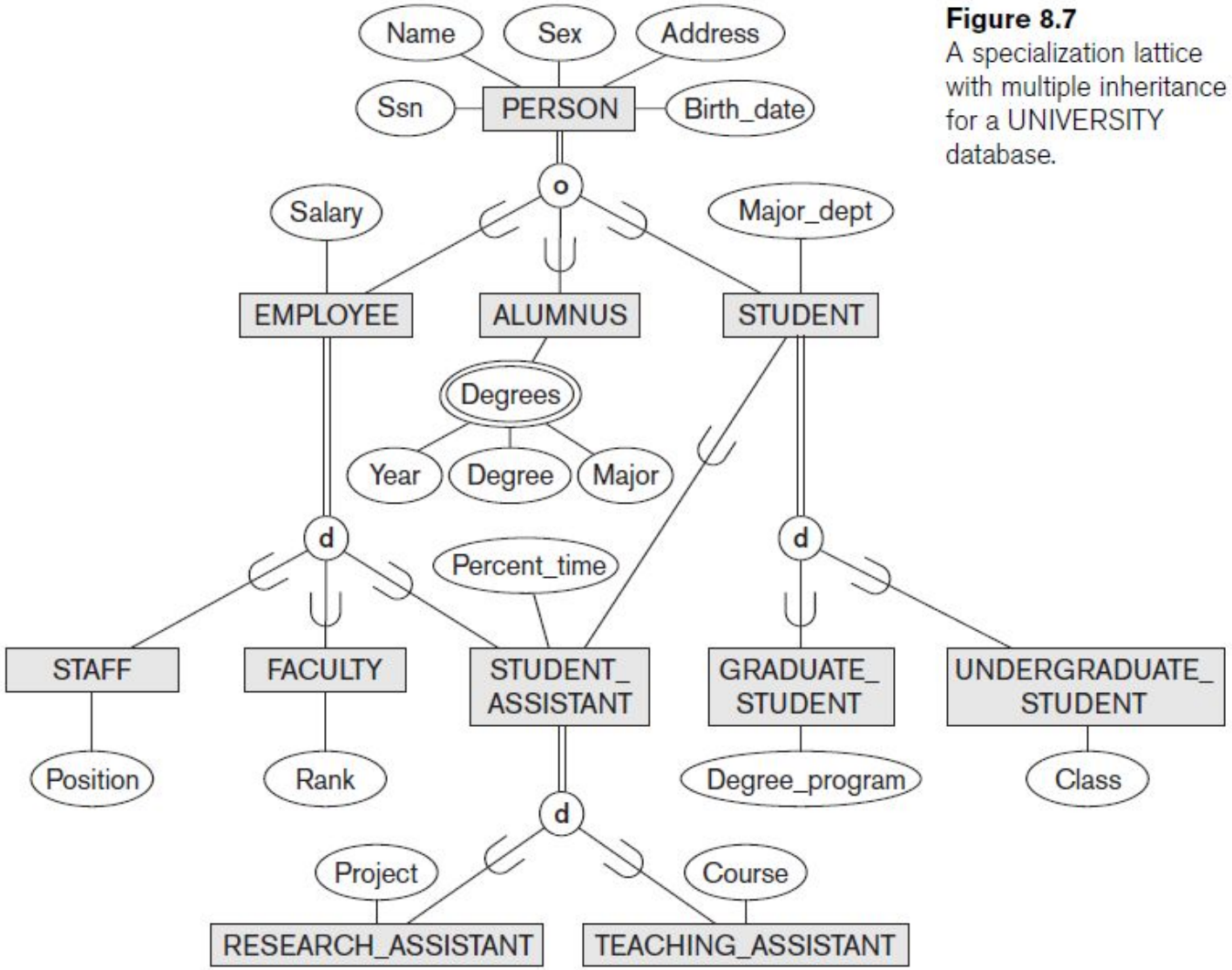


Figure 8.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

Specialization/Generalization Hierarchies, Lattices and Shared Subclasses (cont.)

- In *specialization*, start with an entity type and then define subclasses of the entity type by successive specialization
  - called a *top down* **conceptual refinement** process
- In *generalization*, start with many entity types and generalize those that have common properties (attributes and relationships)
  - Called a *bottom up* **conceptual synthesis** process
- In practice, a *combination of both processes* is usually employed

Categories (UNION TYPES)

- All of the *superclass/subclass* ***relationships*** we have seen thus far have a single superclass
- A shared subclass is a subclass in:
  - *more than one* distinct superclass/subclass relationships
  - each relationships has a *single superclass*
  - shared subclass leads to multiple inheritance
- In some cases, we need to model a *single superclass/subclass relationship* with *more than one* superclass
- The superclasses can represent different entity types
- Such a subclass is called a **category** or **UNION TYPE**

Categories (UNION TYPES) (cont.)

- Example: In a database for vehicle registration, a vehicle owner can be a PERSON, a BANK (holding a loan on a vehicle) or a COMPANY (see Figure 8.8, next slide)
  - A *category* (UNION type) called OWNER is created to represent a subset of the ***union*** of the three superclasses COMPANY, BANK, and PERSON
  - A category member must exist in ***at least one*** of its superclasses
- Difference from *shared subclass*, which is a:
  - subset of the ***intersection*** of its superclasses
  - shared subclass member must exist in ***all*** of its superclasses

Examples of EER Schema Diagrams

- Next two slides show examples of two EER schema diagrams
  - Figure 8.9, next slide, is a UNIVERSITY database
  - Figure 8.12, following slide, is a SMALL AIRPORT database

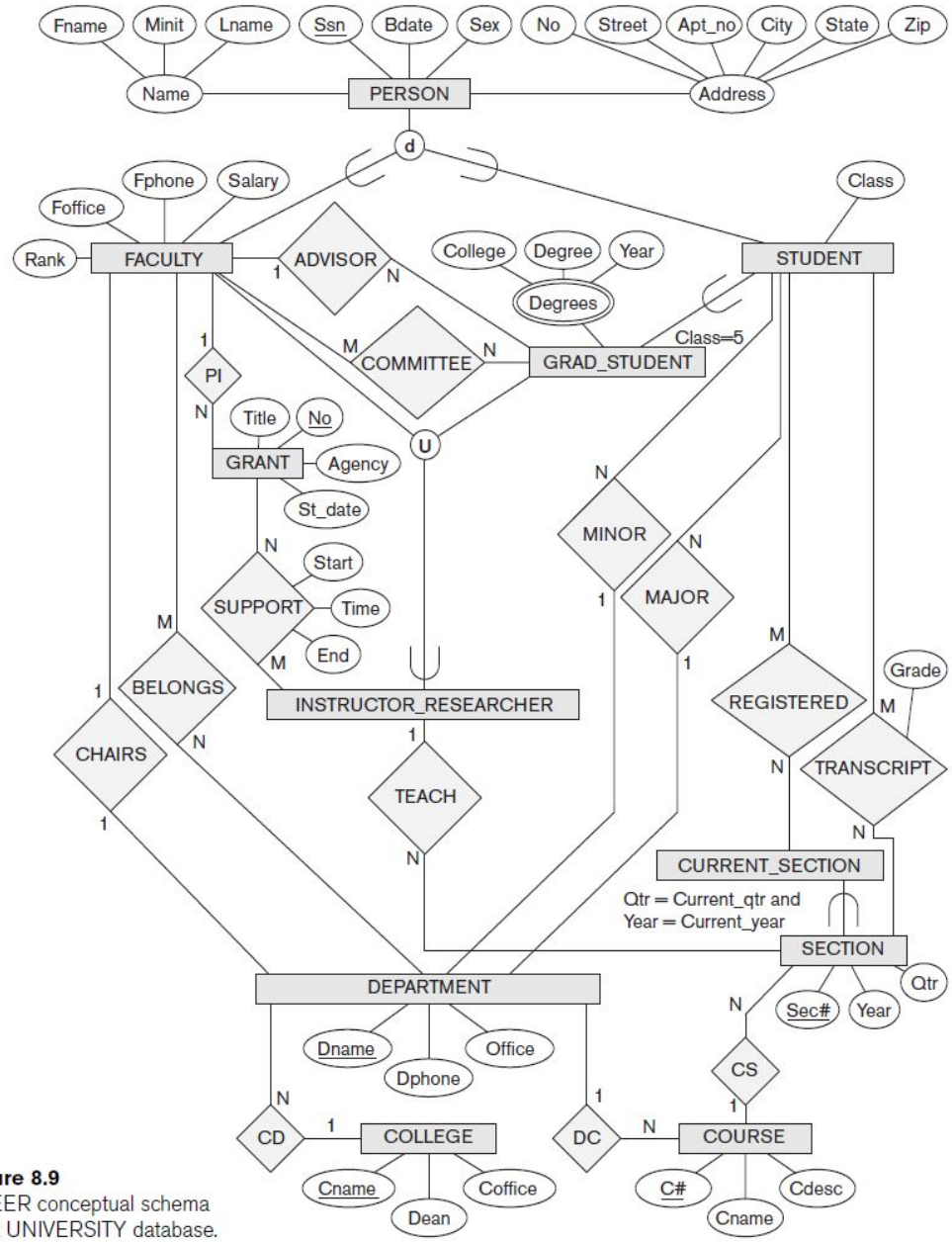


Figure 8.9
An EER conceptual schema for a UNIVERSITY database.

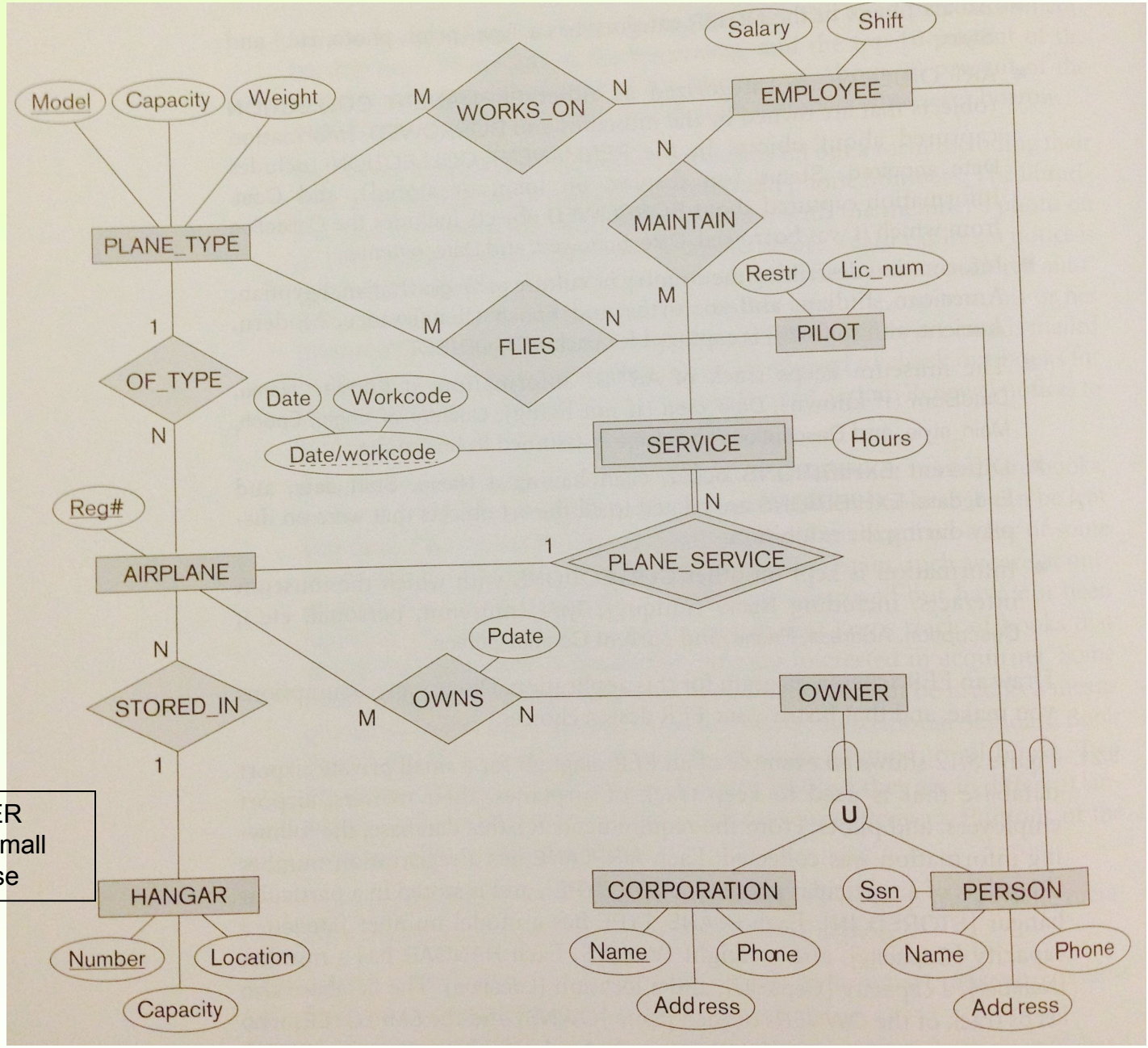


Figure 8.12 EER Schema for a small Airport Database

Alternative diagrammatic notations

- ER/EER diagrams are a specific notation for displaying the concepts of the enhanced model diagrammatically
- DB design tools use many alternative notations for the same or similar concepts
- One popular alternative notation uses *UML class diagrams*
- see next slide (Figure 8.10) for UML class diagrams notation

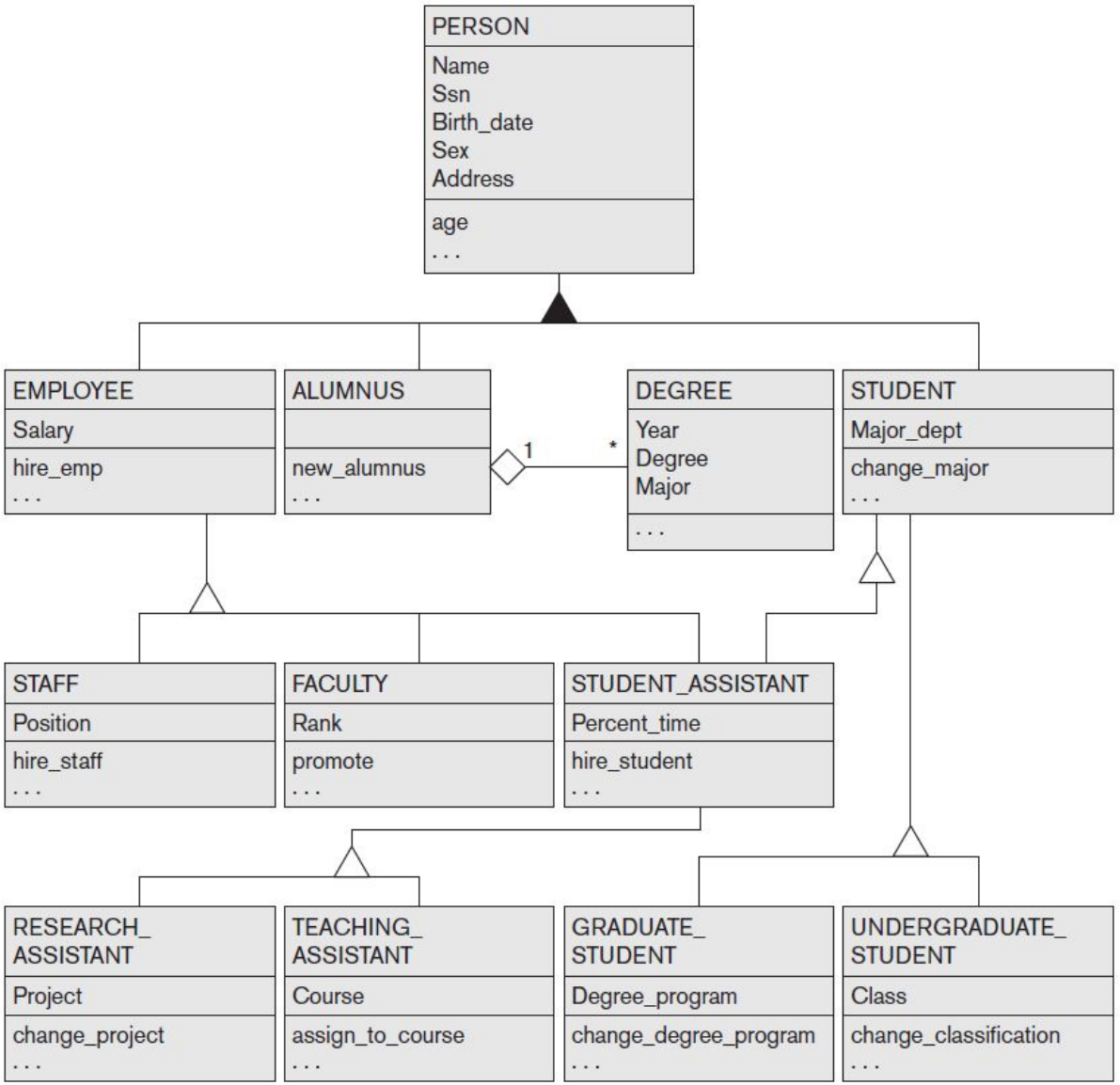


Figure 8.10
A UML class diagram corresponding to the EER diagram in Figure 8.7, illustrating UML notation for specialization/generalization.

General Conceptual Modeling Concepts

- GENERAL DATA ABSTRACTIONS
  - CLASSIFICATION and INSTANTIATION
  - AGGREGATION and ASSOCIATION (relationships)
  - GENERALIZATION and SPECIALIZATION
  - IDENTIFICATION
- CONSTRAINTS
  - CARDINALITY (Min and Max)
  - COVERAGE (Total vs. Partial, and Exclusive (disjoint) vs. Overlapping)

Ontologies

- Use conceptual modeling and other tools to develop “a specification of a conceptualization”
  - **Specification** refers to the language and vocabulary (data model concepts) used
  - **Conceptualization** refers to the description (schema) of the concepts of a particular field of knowledge and the relationships among these concepts
- Many medical, scientific, and engineering ontologies are being developed as a means of standardizing concepts and terminology

Ontologies (cont.)

- Example of ontology languages and tools:
  - **Protege:** an ontology editor that allows users to create and edit ontologies
  - **OWL (Ontology Web Language):** a language for specifying ontologies; accepts ontologies specified in XML (see chapter 12) and RDF (Resource Description Framework)
- Ontologies are considered essential for knowledge representation and information interchange in the “Semantic Web”

Summary, Formal Definitions of EER Model

- A class C:
  - An entity type (with a corresponding entity set):
    - could be entity type, subclass, superclass
- Note: The definition of *relationship type* in ER/EER should have 'entity type' replaced with 'class‘ to allow relationships among classes in general
- Subclass S is a class whose:
  - Type inherits all the attributes and relationship of superclass C
  - Set of entities must always be a subset of the set of entities of superclass C
    - $S \subseteq C$
  - A superclass/subclass relationship exists between S and C

Summary, Formal Definitions of EER Model (cont.)

- Specialization Z: Z = {S1, S2,…, Sn} is a set of subclasses with same superclass G; hence, G/Si is a superclass relationship for i = 1, …., n.
  - G is called a generalization of the subclasses {S1, S2,…, Sn}
  - Z is total if we always have:
    - S1 ∪ S2 ∪ … ∪ Sn = G;
    - Otherwise, Z is partial.
  - Z is disjoint if we always have:
    - ((Si ∩ S2) = empty-set) for i ≠ j;
    - Otherwise, Z is overlapping.

Summary, Formal Definitions of EER Model (cont.)

- Subclass S of C is predicate defined if predicate (condition) p on attributes of C is used to specify membership in S;
  - that is, S = C[p], where C[p] is the set of entities in C that satisfy condition p
- A subclass not defined by a predicate is called user-defined
- Attribute-defined specialization: if a predicate A = ci (where A is an attribute of G and ci is a constant value from the domain of A) is used to specify membership in each subclass Si in Z
  - Note: If ci ≠ cj for i ≠ j, and A is single-valued, then the attribute-defined specialization will be disjoint.

Summary, Formal Definitions of EER Model (cont.)

- Category (UNION type) T
  - A class that is a subset of the *union* of n defining superclasses D1, D2,…Dn, n>1:
    - $T \subseteq (D1 \cup D2 \cup \ldots \cup Dn)$
  - Can have a predicate pi on the attributes of Di to specify entities of Di that are members of T.
  - If a predicate is specified on every Di: T = $(D1[p1] \cup D2[p2] \cup \ldots \cup Dn[pn])$

Chapter 8 Summary

- Introduced the EER model concepts
  - Class/subclass relationships
  - Specialization and generalization
  - Inheritance
  - Categories/UNION Types
- These augment the basic ER model concepts introduced in Chapter 7
- EER diagrams and alternative notations were presented