Chap 4: The Enhanced Entity-Relationship (EER) Model

4.1 Subclasses, Superclasses, and Inheritance

EER model includes all the modeling concepts of the ER model

Associated with these concepts is the important mechanism of attribute and relationship inheritance.

- subclass and superclass
- specialization and generalization
- category or union type

Call the resulting schema diagrams enhanced ER or EER diagrams.

The name of an entity type is used to represent both a type of entity and the entity set or collection of entities of that type.

An entity type has numerous subgroupings or subtypes of its entities that are meaningful
and need to be represented explicitly because of their significance to the database
application.

The set or collection of entities in each of the latter groupings is a subset of the entities that
belong to the EMPLOYEE entity set, meaning that every entity that is a member of one of these subgroupings is also an employee. We call each of these subgrouping a subclass or subtype of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the superclass or super type for each of these sub classes

For example, the entities that are members of the EMPLOYEE entity type may be distinguished further into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE, and so on

We call the relationship between a superclass and any one of its subclasses a superclass/subclass or supertype/subtype or simply class/subclass relationship. Notice that a member entity of the subclass represents the same real-world entity as some member of the superclass.

An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass. Such an entity can be included optionally as a member of any number of subclasses.

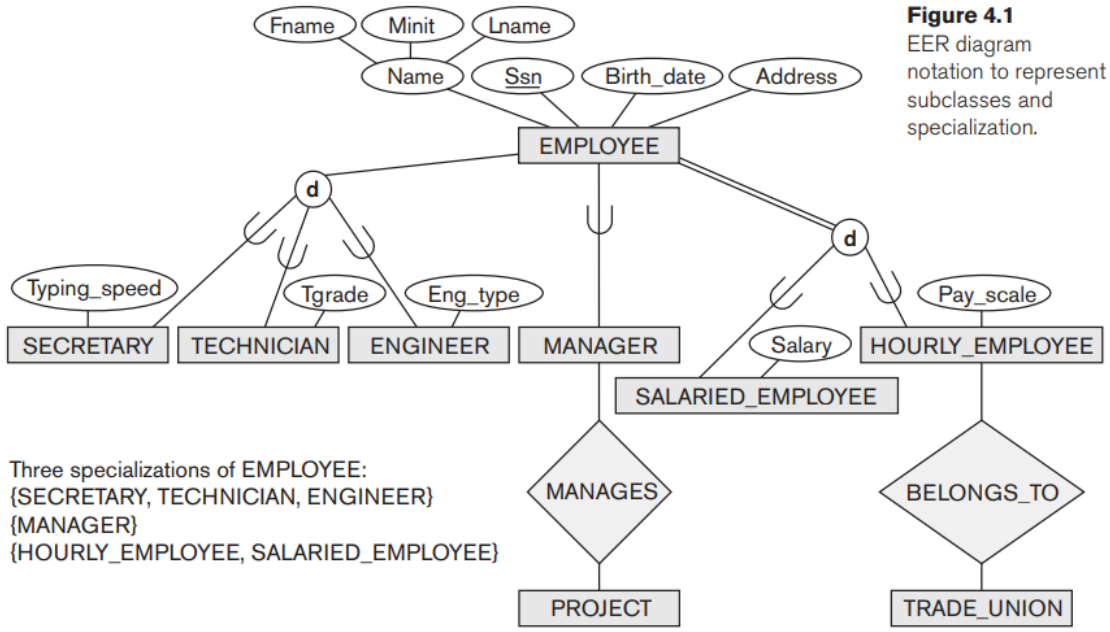


Figure 4.1
EER diagram notation to represent subclasses and specialization.

[3]A class/subclass relationship is often called an **IS-A** (or **IS-AN**) **relationship** because of the way we refer to the concept. We say a SECRETARY *is an* EMPLOYEE, a TECHNICIAN *is an* EMPLOYEE, and so on.

An important concept associated with subclasses (subtypes) is that of type inheritance. Recall that the type of an entity is defined by the attributes it possesses and the relationship types in which it participates. The entity inherits all the attributes and the relationships in which the superclass participates of the entity of the superclass.

A class/subclass relationship is often called an IS-A (or IS-AN) relationship because of the way we
refer to the concept.

4.2 Specialization and Generalization (chuyen mon hoa′ khai quat′ hoa)

4.2.1 Specilization

Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization

The set of subclasses that forms a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass

May have several specializations ò the same entity type based on different distinguishing characteristics

Attributes that apply only to entities of a particular subclass are called specific (or local) attributes of the subclass.

A subclass can participate in specific relationship types,

There are 2 main reasons for including class/subclass relationships and specializations

- The first is certain attributes may apply to some but no all entities of the superclass entity type. A subclass is defined in order to group the entities to which these attributes apply.
- The second reason for using subclasses is that some relationship types may be participated in only by entities that are members of the subclass.

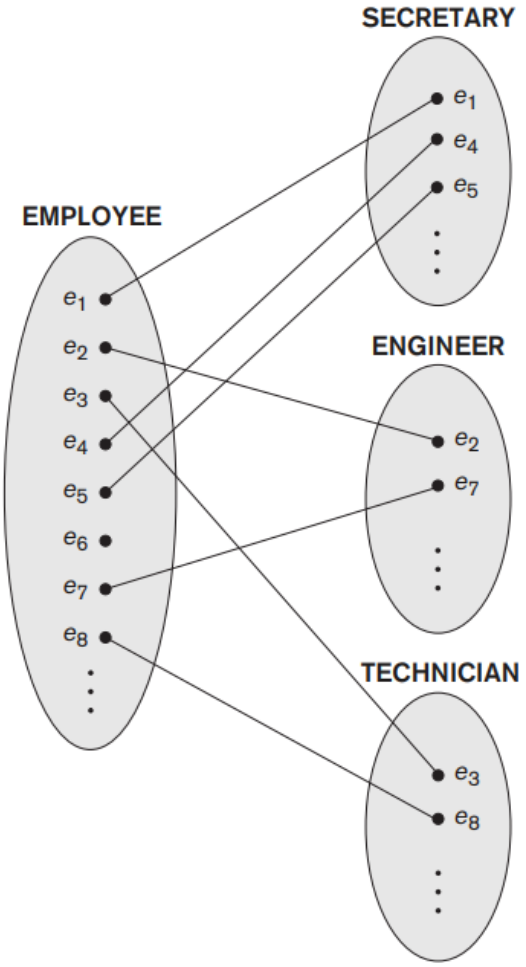
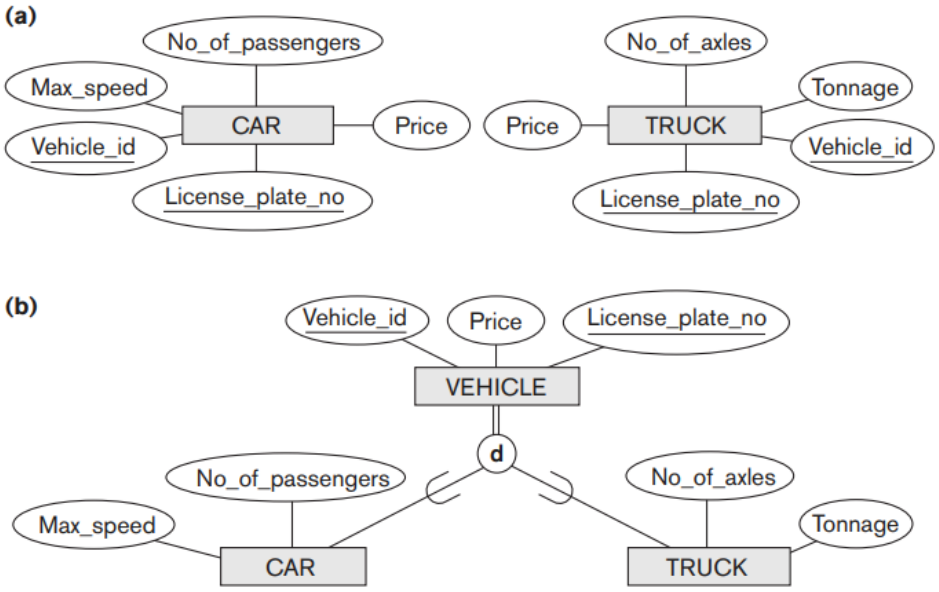


Figure 4.2
Instances of a speciali

4.2.2 Generalization

Reverse process of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single superclass of which the original entity types are special subclasses

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



The term generalization to refer to the process of defining a generalized entity type from the given entity types.

The generalization process can be viewed as being functionally the inverse of the specialization process

4.3 Constraints and Characteristics of Specialization and Generalization Hierarchies

4.3.1 Constraints on Specialization and Generalization

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. Such subclasses are called predicate-defined (or condition-defined) subclasses.

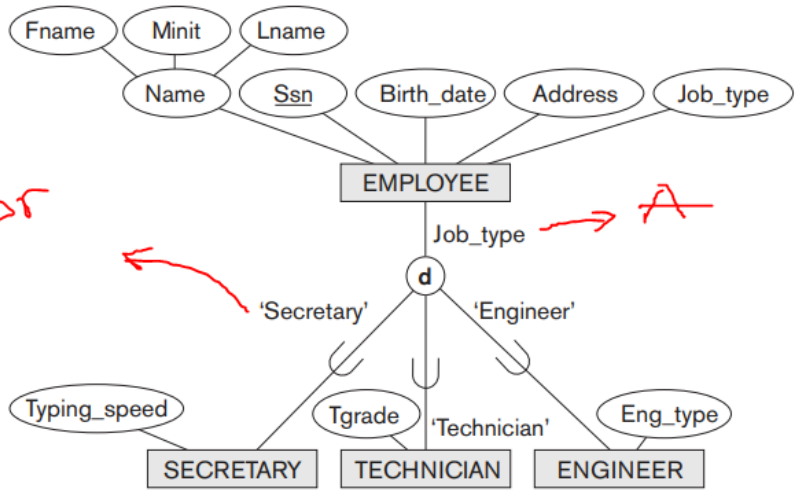


Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

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. Such subclasses are called predicate-defined (or condition-defined)
subclasses

If all subclasses in a specialization have their membership condition on the same attribute of the superclass, the specialization itself is called an attribute-defined specialization, and the attribute is called the defining attribute of the specialization.

When we do not have a condition for determining membership in a subclass, the subclass is called user-defined. Membership is specified individually for each entity by the user, not by any condition that may be evaluated automatically

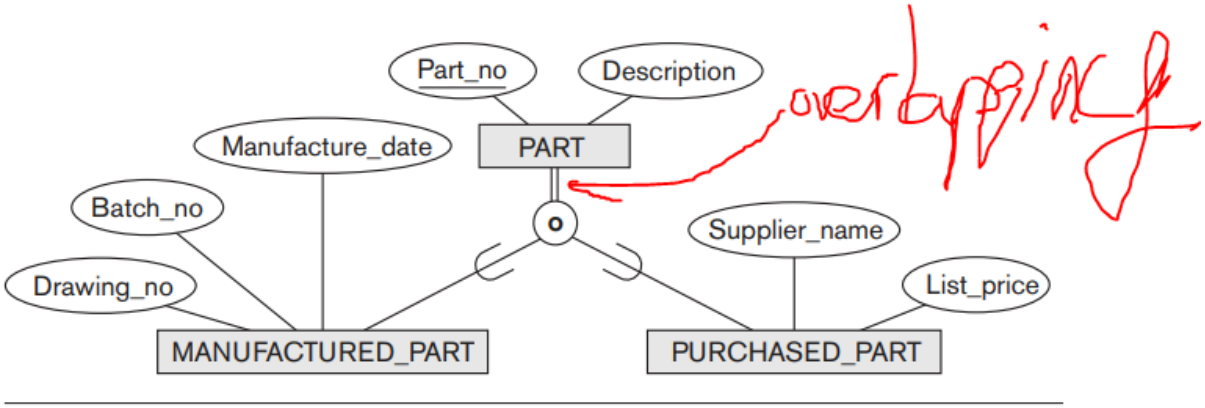
Two other constraints may apply to a specialization. The first is the disjointness constraint

(rang buoc roi rac)

An entity can be a member of at most one of the subclasses of the specialization

A specialization that is attribute-defined implies the disjointness constraint (if the attribute used to define the membership predicate is single valued).

The d notation also applies to user-defined subclasses of a specialization that must be
disjoint



The second constraint on specialization is called the completeness (or totalness) constraint, which may be total or partial

A total specialization constraint specifies that every entity in the superclass must be a member of at least one subclass in the specialization

partial specialization - which allows an entity not to belong to any of the subclasses

The 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, because the superclass is derived from the subclasses and hence contains only the entities that are in the subclasses

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.

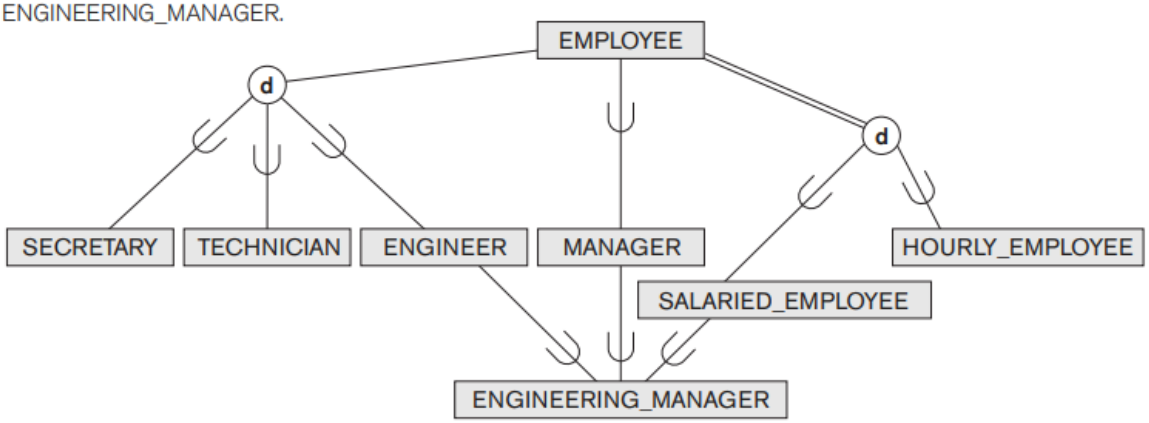
4.3.2 Specialization and Generalization Hierarchies and Lattices

A subclass itself may have further subclasses specified on it, forming a hierarchy or
a lattice of specializations

A specialization hierarchy has the constraint that every subclass participates as a subclass in only one class/subclass relationship, which results in a tree structure or strict hierarchy. (single inheritance)

A specialization lattice, a subclass can be a subclass in more than one class/subclass
relationship (multiple inheritance)

Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



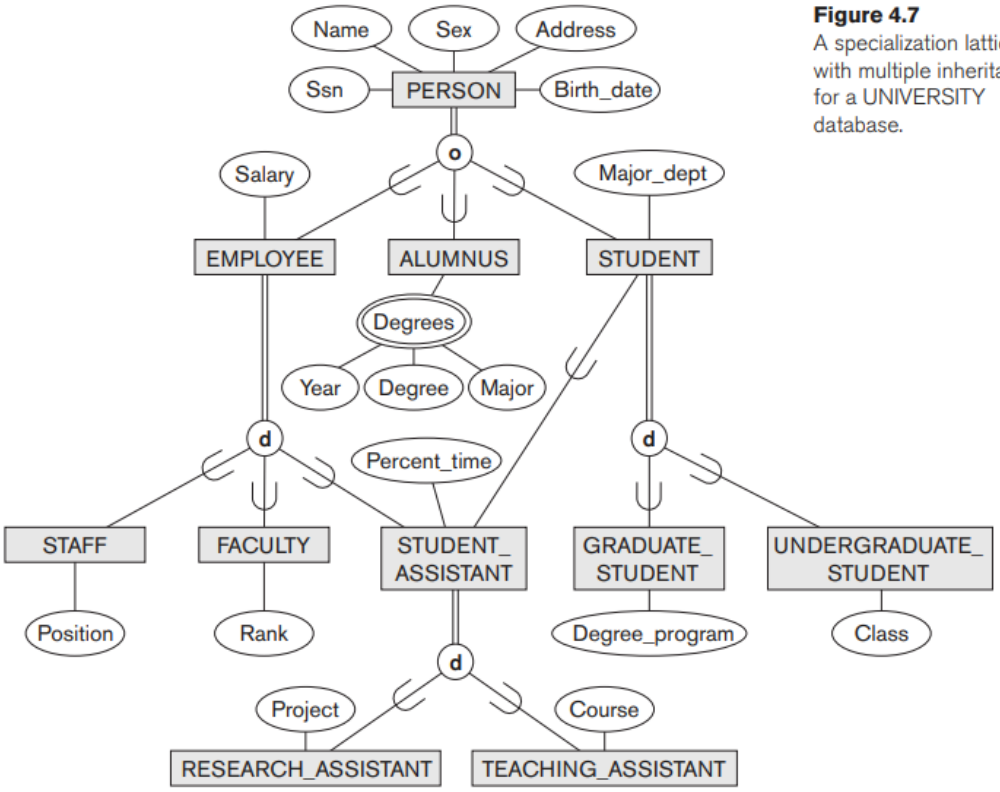


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

1. The database keeps track of three types of persons: employees, alumni, and students. A person can belong to one, two, or all three of these types. Each person has a name, SSN, sex, address, and birth date.
2. Every employee has a salary, and there are three types of employees: faculty, staff, and student assistants. Each employee belongs to exactly one of these types. For each alumnus, a record of the degree or degrees that he or she earned at the university is kept, including the name of the degree, the year granted, and the major department. Each student has a major department.
3. Each faculty has a rank, whereas each staff member has a staff position. Student assistants are classified further as either research assistants or teaching assistants, and the percent of time that they work is recorded in the database. Research assistants have their research project stored, whereas teaching assistants have the current course they work on

4. Students are further classified as either graduate or undergraduate, with the specific attributes degree program (M.S., Ph.D., M.B.A., and so on) for graduate students and class (freshman, sophomore, and so on) for undergraduates.

In such a specialization lattice or hierarchy, a subclass inherits the attributes not only of its direct superclass, but also of all its predecessor superclasses all the way to the root of the hierarchy or lattice if necessary.

Notice that an entity may exist in several leaf nodes of the hierarchy, where a leaf node is a class that has no subclasses of its own

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

Notice that the existence of at least one shared subclass leads to a lattice (and hence to multiple inheritance). if no shared subclasses existed, we would have a hierarchy rather than a lattice and only single inheritance would exist.

Can have specialization hierarchies or lattices, or generalization hierarchies or lattices.

4.3.3 Utilizing Specialization and Generalization in Refining Conceptual Schemas

In the specialization process, the database designers typically start with an entity type and then define subclasses of the entity type by successive specialization. They repeatedly define more specific groupings of the entity type, this process is called top-down conceptual refinement.

It is possible to arrive at the same hierarchy or lattice from the other direction. In such a case, the process involves generalization rather than specialization and corresponds to a bottom-up conceptual synthesis

4.4 Modeling of UNION Types Using Categories

a subclass will represent a collection of entities that is a subset of the UNION of entities from distinct entity types; we call such a subclass a union type or a category.

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.

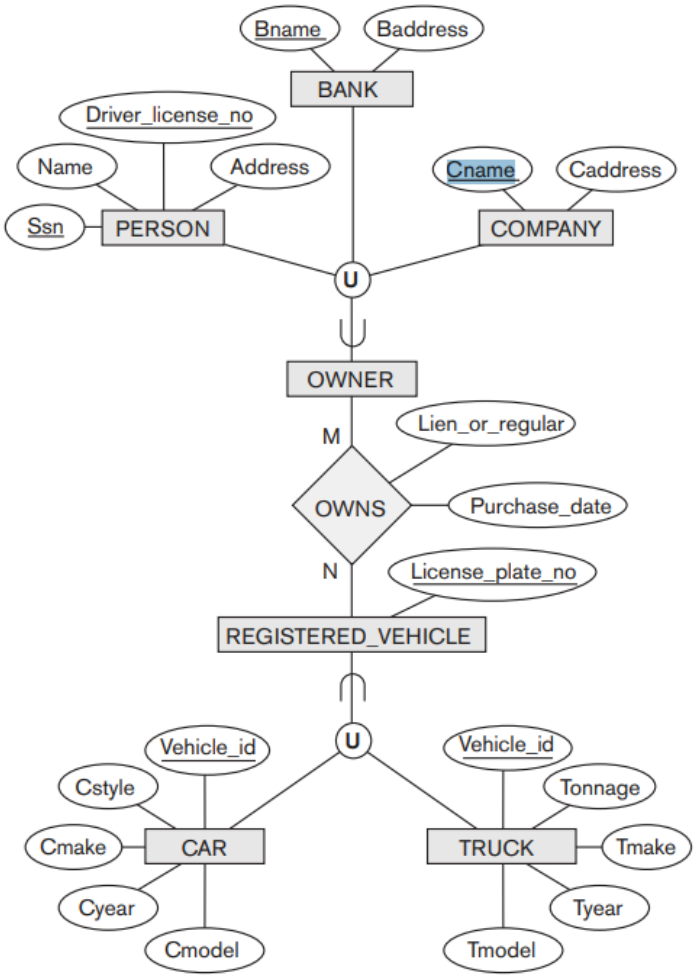


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

A category can be total or partial. A total category holds the union of all entities in its superclasses, whereas a partial category can hold a subset of the union.

The superclasses of a category may have different key attribute or may have the same attributes

Notice that if a category is total (not partial), it may be represented alternatively as a total specialization (or a total generalization)

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

4.5.1 A Different UNIVERSITY Database Example

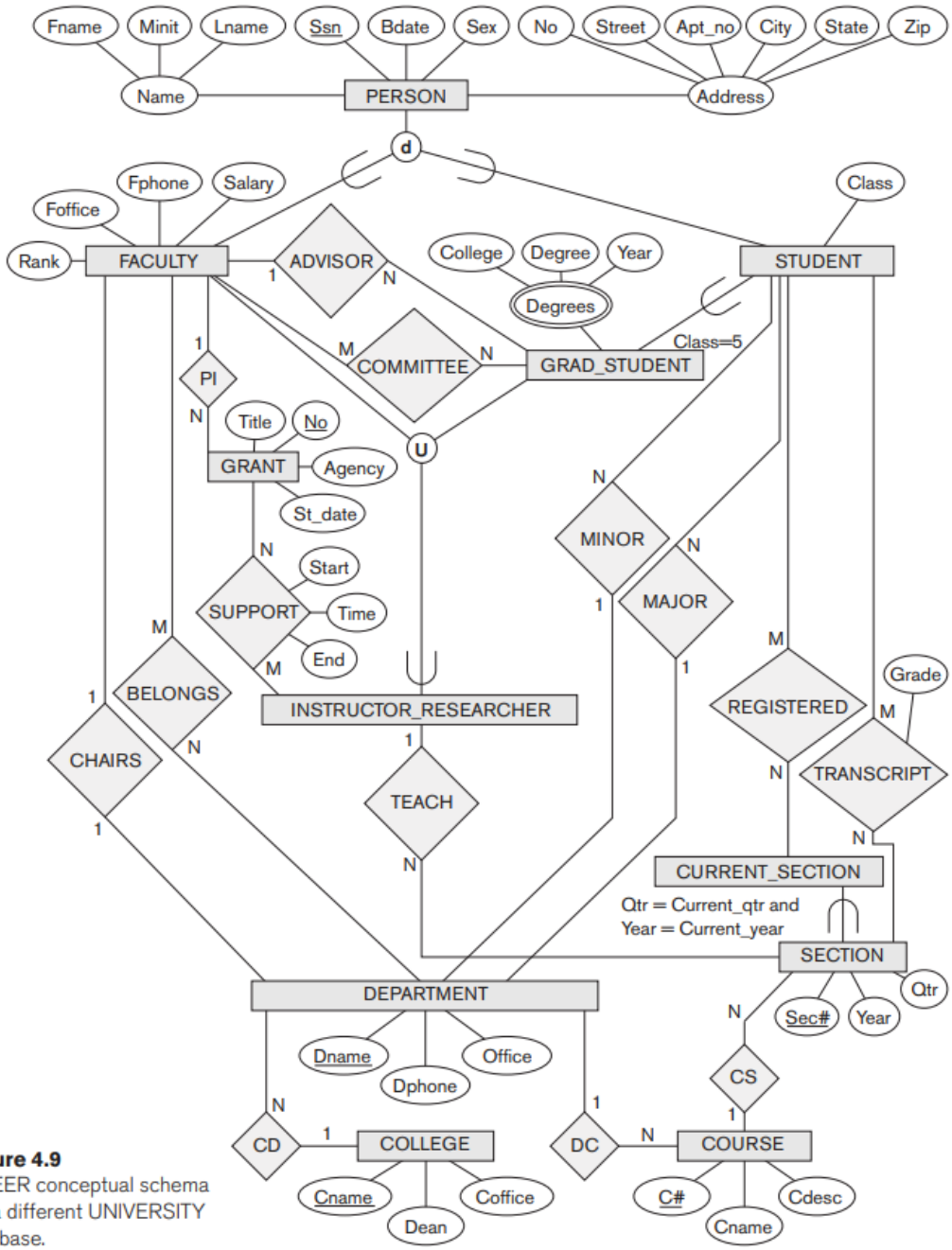


Figure 4.9
An EER conceptual schema for a different UNIVERSITY database.

4.5.2 Design Choices for Specialization/Generalization

The following guidelines can help to guide the design process for EER concepts:

- In general, many specializations and subclasses can be defined to make the conceptual model accurate. However, the drawback is that the design becomes quite cluttered. It is important to represent only those subclasses that are deemed necessary to avoid extreme cluttering of the conceptual schema
- In general, many specializations and subclasses can be defined to make the conceptual model accurate. However, the drawback is that the design becomes quite cluttered. It is important to represent only those subclasses that are deemed necessary to avoid extreme cluttering of the conceptual schema
- Similarly, if all the subclasses of a specialization/generalization have few specific attributes and no specific relationships, they can be merged into the superclass and replaced with one or more type attributes that specify the subclass or subclasses that each entity belongs to.
- Union types and categories should generally be avoided unless the situation definitely warrants this type of construct, which does occur in some practical situations.
- The choice of disjoint/overlapping and total/partial constraints on specialization/generalization is driven by the rules in the mini world being modeled. If the requirements do not indicate any particular constraints, the default would generally be overlapping and partial, since this does not specify any restrictions on subclass membership.

4.5.3 Formal Definitions for the EER Model Concepts

We now summarize the EER model concepts and give formal definitions. A **class**[11] defines a type of entity and represents a set or collection of entities of that type; this includes any of the EER schema constructs that correspond to collections of entities, such as entity types, subclasses, superclasses, and categories. 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**. We denote such a relationship by C/S. For such a superclass/subclass relationship, we must always have

$$S \subseteq C$$

A **specialization** $Z = \{S_1, S_2, \ldots, S_n\}$ is a set of subclasses that have the same superclass G; that is, G/S_i is a superclass/subclass relationship for $i = 1, 2, \ldots, n$. G is called a **generalized entity type** (or the **superclass** of the specialization, or a **generalization** of the subclasses $\{S_1, S_2, \ldots, S_n\}$). Z is said to be **total** if we always (at any point in time) have

$$\bigcup_{i=1}^{n} S_i = G$$

Otherwise, Z is said to be **partial**. Z is said to be **disjoint** if we always have

$$S_i \cap S_j = \emptyset \text{ (empty set) for } i \neq j$$

Otherwise, Z is said to be **overlapping**.

A subclass S of C is said to be **predicate-defined** if a predicate p on the attributes of C is used to specify which entities in C are members of S; that is, $S = C[p]$, where $C[p]$ is the set of entities in C that satisfy p. A subclass that is not defined by a predicate is called **user-defined**.

A specialization Z (or generalization G) is said to be **attribute-defined** if a predicate $(A = c_i)$, where A is an attribute of G and c_i is a constant value from the domain of A, is used to specify membership in each subclass S_i in Z. Notice that if $c_i \neq c_j$ for $i \neq j$, and A is a single-valued attribute, then the specialization will be disjoint.

A **category** T is a class that is a subset of the union of n defining superclasses $D_1, D_2, \ldots, D_n$, $n > 1$ and is formally specified as follows:

$$T \subseteq (D_1 \cup D_2 \ldots \cup D_n)$$

A predicate p_i on the attributes of D_i can be used to specify the members of each D_i that are members of T. If a predicate is specified on every D_i, we get

$$T = (D_1[p_1] \cup D_2[p_2] \ldots \cup D_n[p_n])$$

We should now extend the definition of **relationship type** given in Chapter 3 by allowing any class—not only any entity type—to participate in a relationship. Hence, we should replace the words *entity type* with *class* in that definition. The graphical notation of EER is consistent with ER because all classes are represented by rectangles.

4.6 Example of Other Notation: Representing Specialization and Generalization in UML Class Diagrams

The root superclass is called the base class, and the subclasses (leaf nodes) are called leaf classes.

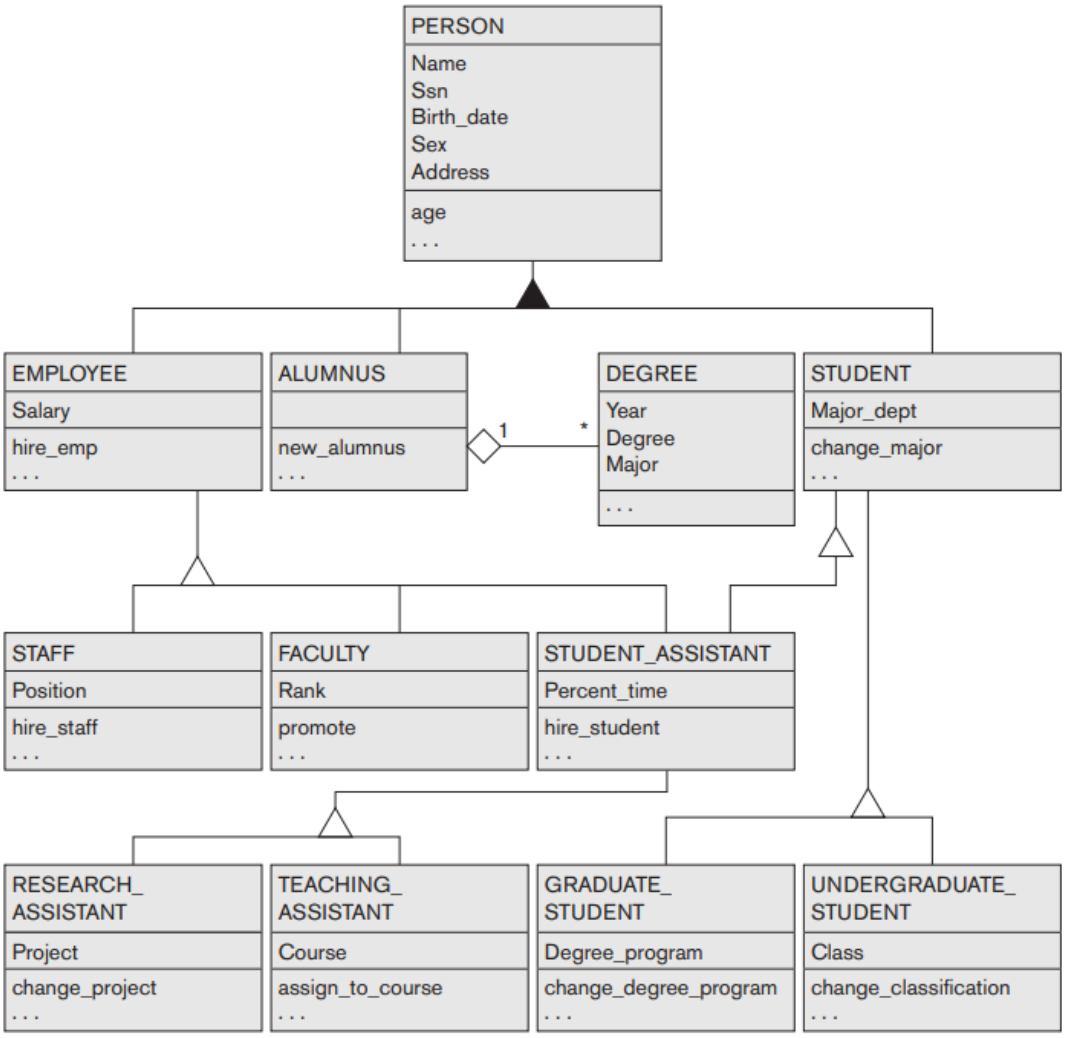


Figure 4.10
A UML class diagram corresponding to the EER diagram in Figure 4.7, illustrating UML notation for specialization/generalization.

4.7 Data Abstraction, Knowledge Representation, and Ontology Concepts

4.7.1 Classification and Instantiation

4.7.2 Identification

4.7.3 Specialization and Generalization

4.7.4 Aggregation and Association

4.7.5 Ontologies and the Semantic Web