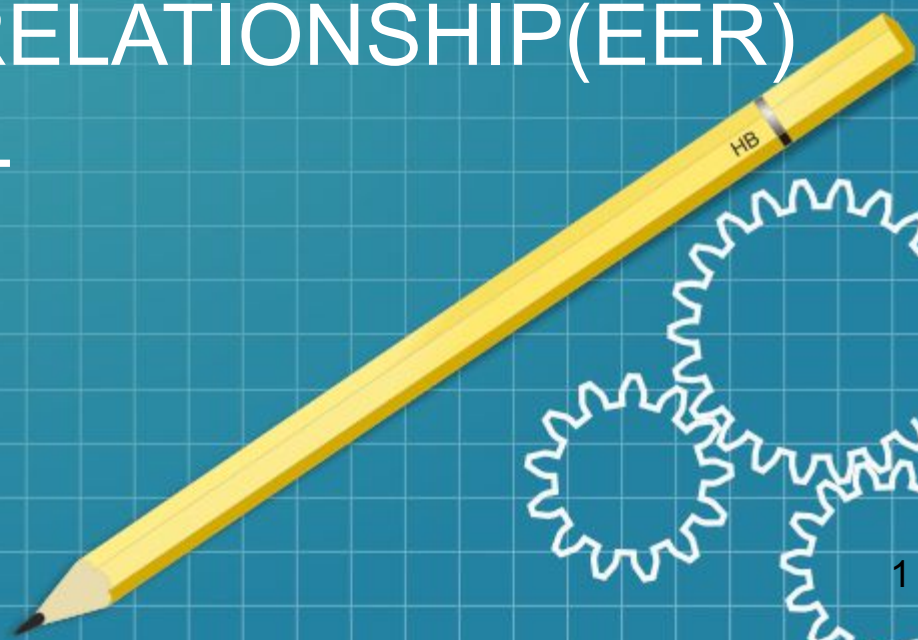
CHAPTER -8

THE ENHANCED ENTITY-RELATIONSHIP(EER) MODEL



-By Ranjana J

INDEX

-By Ranjana J

INTRODUCTION

- The ER modeling concepts discussed are sufficient for representing many database schemas for traditional database applications, which include many data-processing applications in business and industry.
- Since the late 1970s, however, designers of database applications have tried to design more accurate database schemas that reflect the data properties and constraints more precisely.

- These types of databases have more complex requirements than do the more traditional applications. This led to the development of additional **semantic data modeling** concepts that were incorporated into conceptual data models such as the ER model.
- Various semantic data models have been proposed in the literature.
- Many of these concepts were also developed independently in related areas of computer science, such as the *knowledge representation* area of artificial intelligence and the *object modeling* area in software engineering.

- In this chapter, we describe features that have been proposed for semantic data models, and show how the ER model can be enhanced to include these concepts, leading to the Enhanced ER (EER) model.

8.1 SUBCLASSES, SUPERCLASSES,AND INHERITANCE

- The EER model includes all the modeling concepts of the ER model.
- In addition, it includes the concepts of subclass and superclass and the related concepts of specialization and generalization.
- Another concept included in the EER model is that of a **category or union type** , which is used to represent a collection of objects (entities) that is the union of objects of different entity types. Associated with these concepts is the important mechanism of **attribute and relationship inheritance.**

SUBTYPE or SUBCLASS of an entity type

- **entity type :** an entity type is used to represent both a type of entity and the entity set or collection of entities of that type that exist in the database.

- **For example,** the entity type EMPLOYEE describes the type (that is, the attributes and relationships) of each employee entity, and also refers to the current set of EMPLOYEE entities in the COMPANY database.

- 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.
- The set 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 subgroupings a **subclass** or **subtype** of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the **superclass** or **supertype** for each of these subclasses.

- We call the relationship between a superclass and any one of its subclasses a **superclass/subclass** or **supertype/subtype** or simply **class/subclass** relationship.

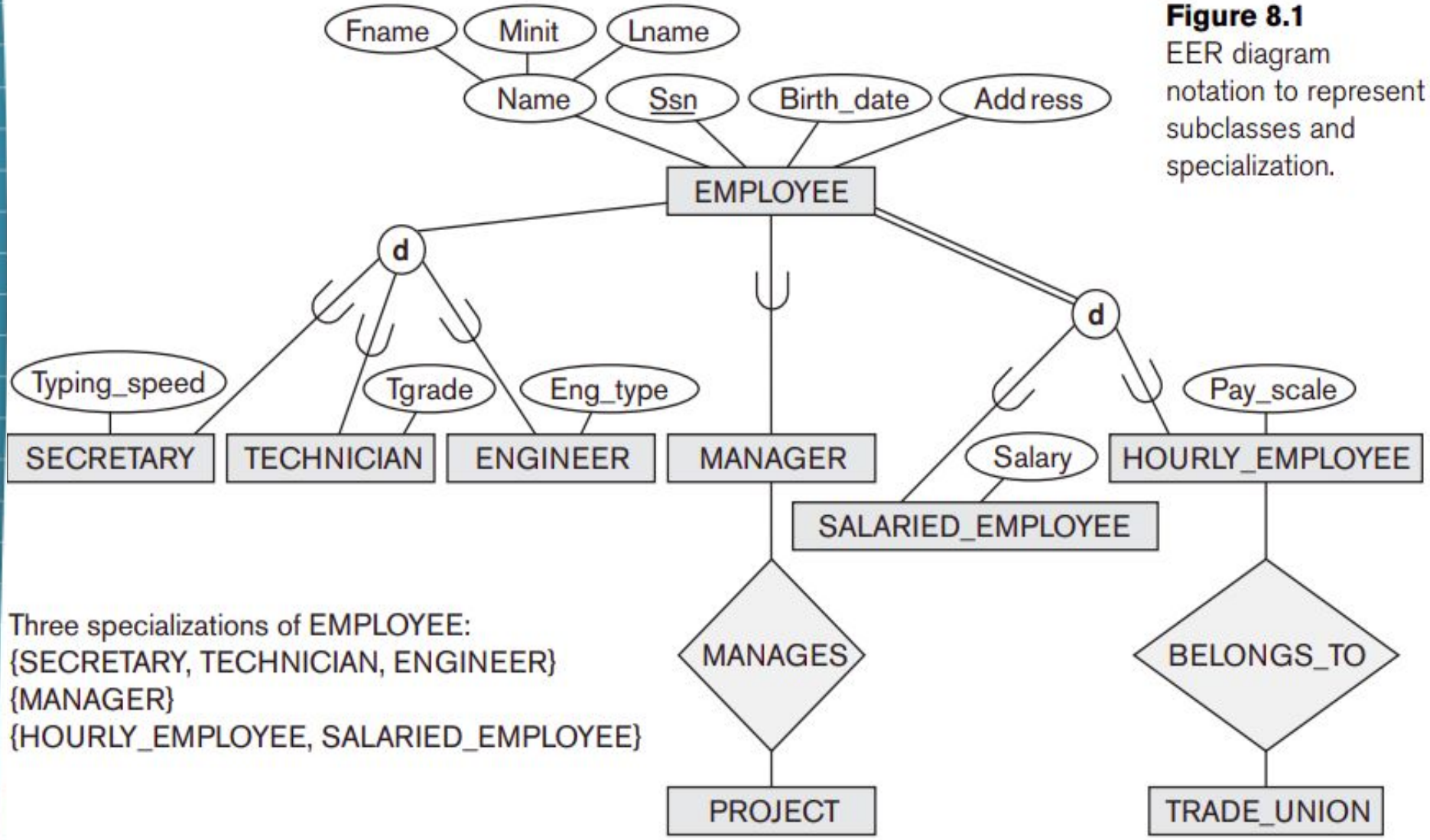


Figure 8.1
EER diagram notation to represent subclasses and specialization.

- Example, EMPLOYEE/SECRETARY and EMPLOYEE/TECHNICIAN are two class/subclass relationships in the previous figure.

- **Note**: 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.
  - For example, a salaried employee who is also an engineer belongs to the two subclasses ENGINEER and SALARIED_EMPLOYEE of the EMPLOYEE entity type. However, it is not necessary that every entity in a superclass is a member of some subclass.
- An important concept associated with subclasses (subtypes) is that of **type inheritance**.

8.2 **SPECIALIZATION AND GENERALIZATION**

8.2.1 Specialization

- Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the **superclass** of the specialization.
- We may have several specializations of the same entity type based on different distinguishing characteristics.
- For example, another specialization of the EMPLOYEE entity type may yield the set of subclasses {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}; this specialization distinguishes among employees based on the method of pay.
- The subclasses that define a specialization are attached by lines to a circle that represents the specialization, which is connected in turn to the superclass.

- Attributes that apply only to entities of a particular subclass—such as TypingSpeed of SECRETARY—are attached to the rectangle representing that subclass. These are called **specific attributes** (or local attributes) of the subclass.

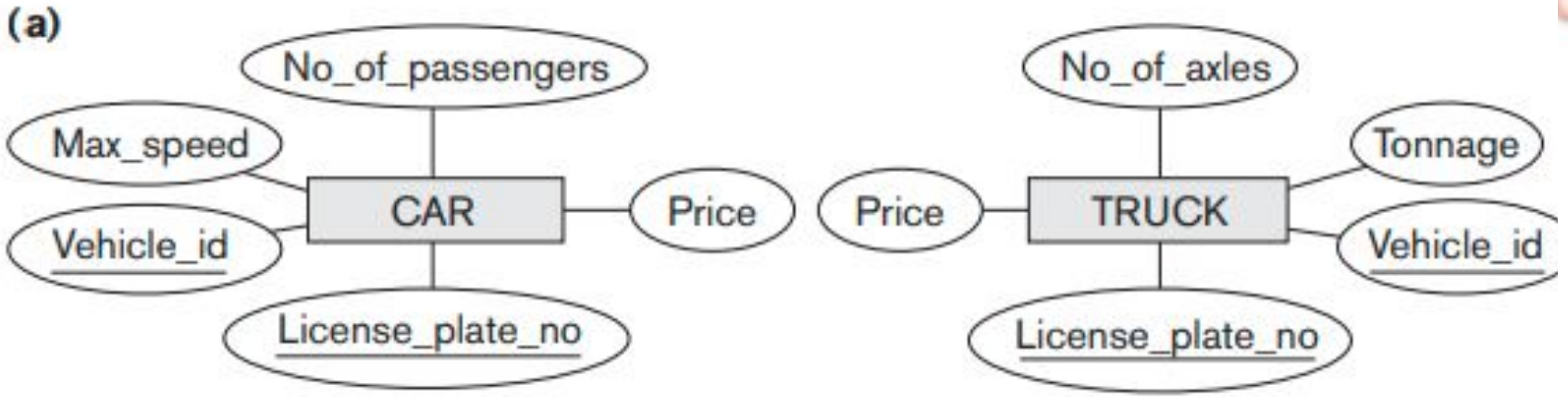
- NOTE : The main difference is that in a 1:1 relationship two distinct entities are related, whereas in a superclass/subclass relationship the entity in the subclass is the same real-world entity as the entity in the superclass but is playing a specialized role—for example, an EMPLOYEE specialized in the role of SECRETARY, or an EMPLOYEE specialized in the role of TECHNICIAN.

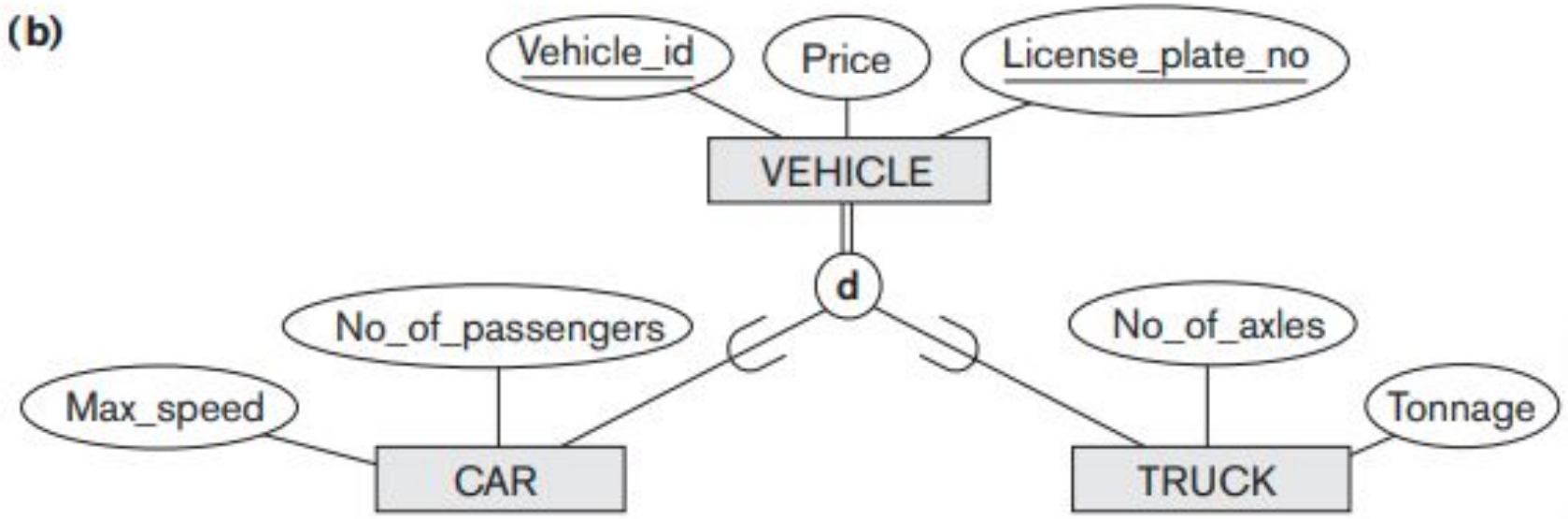
In summary, the specialization process allows us to do the following:

- Define a set of subclasses of an entity type
- Establish additional specific attributes with each subclass
- Establish additional specific relationship types between each subclass and other entity types or other subclasses

8.2.2 Generalization

- We can think of a 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.
- We use the term generalization to refer to the process of defining a generalized entity type from the given entity types.
- FOR EXAMPLE : we can view EMPLOYEE as a generalization of SECRETARY, TECHNICIAN, and ENGINEER.


(a)
No_of_passengers
Max_speed
CAR
Price
Vehicle_id
License_plate_no
No_of_axles
Tonnage
Price
TRUCK
Vehicle_id
License_plate_no


(b)
Vehicle_id
Price
License_plate_no
VEHICLE
d
No_of_passengers
Max_speed
CAR
No_of_axles
Tonnage
TRUCK

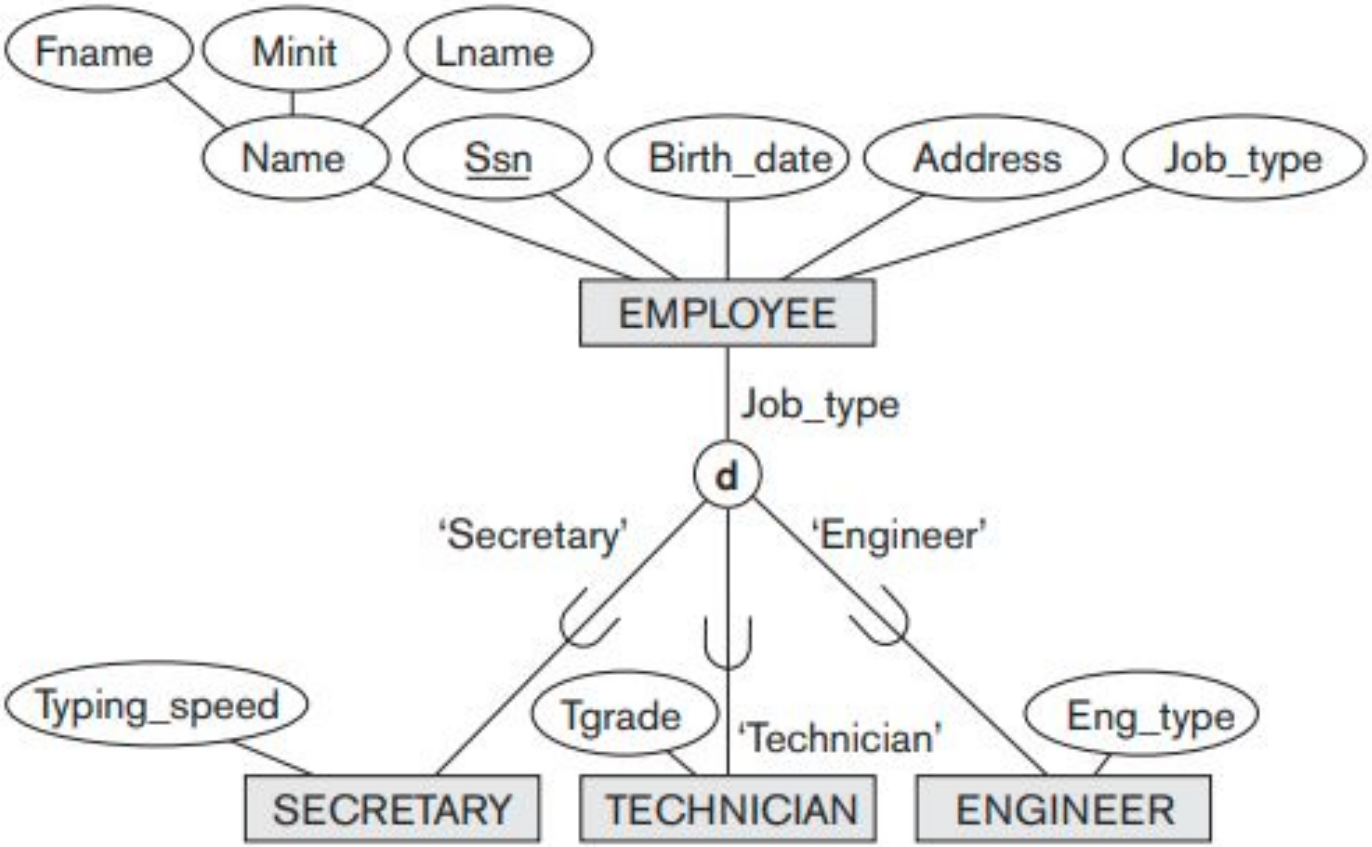
8.3 CONSTRAINTS AND CHARACTERISTICS OF SPECIALIZATION AND GENERALIZATION HIERARCHIES

8.3.1 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.
- For example, if the EMPLOYEE entity type has an attribute Job_type, as shown in Figure 8.4, 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.
- This condition is a constraint specifying that exactly those entities of the EMPLOYEE entity type whose attribute value for Job_type is ‘Secretary’ belong to 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.

- 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.**

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



Two other constraints may apply to a specialization :

1. Disjointness (or disjointedness) constraint
2. Completeness (or totalness) constraint

Disjointness (or disjointedness) constraint

- The first is the disjointness (or disjointedness) constraint, which specifies that the subclasses of the specialization must be disjoint.
- This means that 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, as illustrated by the specialization {HOURLY_EMPLOYEE, SALARIED_EMPLOYEE} in Figure 8.1.
- If the subclasses are not constrained to be disjoint, their sets of entities may be **overlapping**; that is, the same (real-world) entity may be a member of more than one subclass of the specialization. This case, which is the default, is displayed by placing an **o** in the circle.

Completeness (or totalness) constraint

- 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.
- A single line is used to display a **partial specialization**, which allows an entity not to belong to any of the subclasses.

- Notice that the disjointness and completeness constraints are independent. Hence, we have the following four possible constraints on specialization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial

SOME RULES

Certain insertion and deletion rules apply to specialization (and generalization) as a consequence of the constraints specified earlier. Some of these rules are as follows:

■ 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

8.3.2 Specialization and Generalization Hierarchies and Lattices

8.3.2 Specialization and Generalization Hierarchies and Lattices

- A subclass itself may have further subclasses specified on it, forming a hierarchy or a lattice of specializations. For example, in Figure 8.6 ENGINEER is a subclass of EMPLOYEE and is also a superclass of ENGINEERING_MANAGER; this represents the real-world constraint that every engineering manager is required to be an engineer.

- **A specialization hierarchy has the constraint that every subclass participates as a subclass in only one class/subclass relationship; that is, each subclass has only one parent, which results in a tree structure or strict hierarchy.**

- In contrast, for a specialization lattice, a subclass can be a subclass in more than one class/subclass relationship. Hence, Figure 8.6 is a lattice.

Figure 8.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

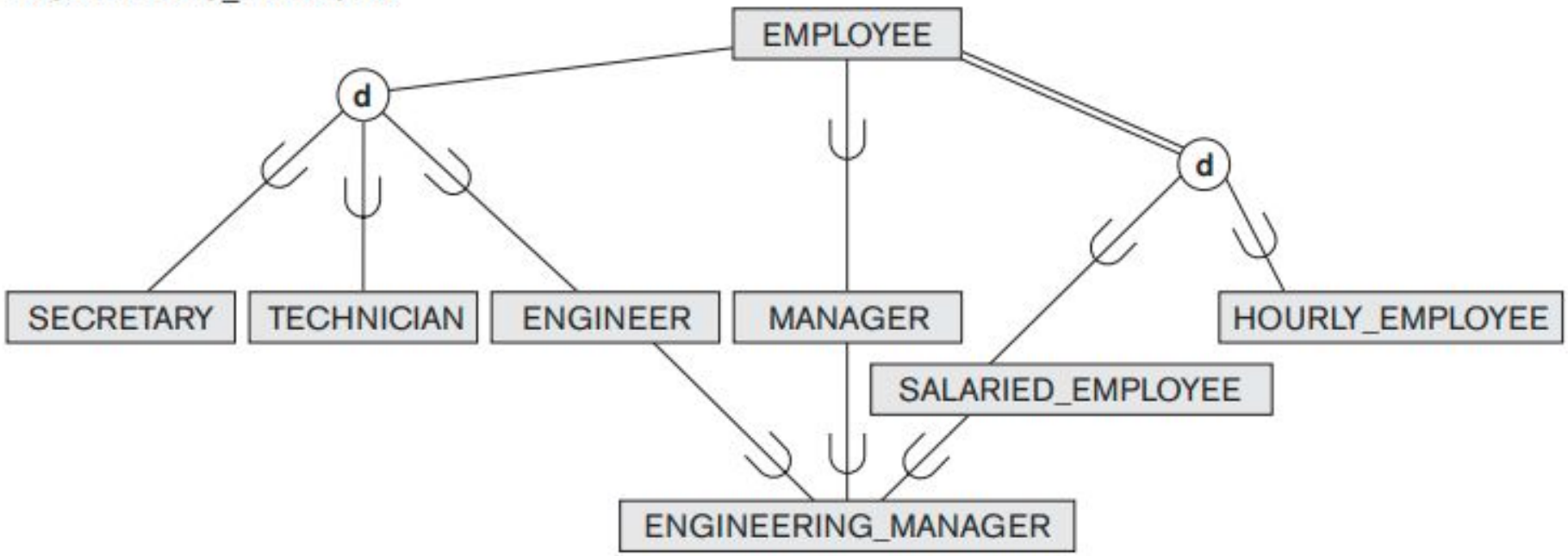


Figure 8.7 shows another specialization lattice of more than one level.

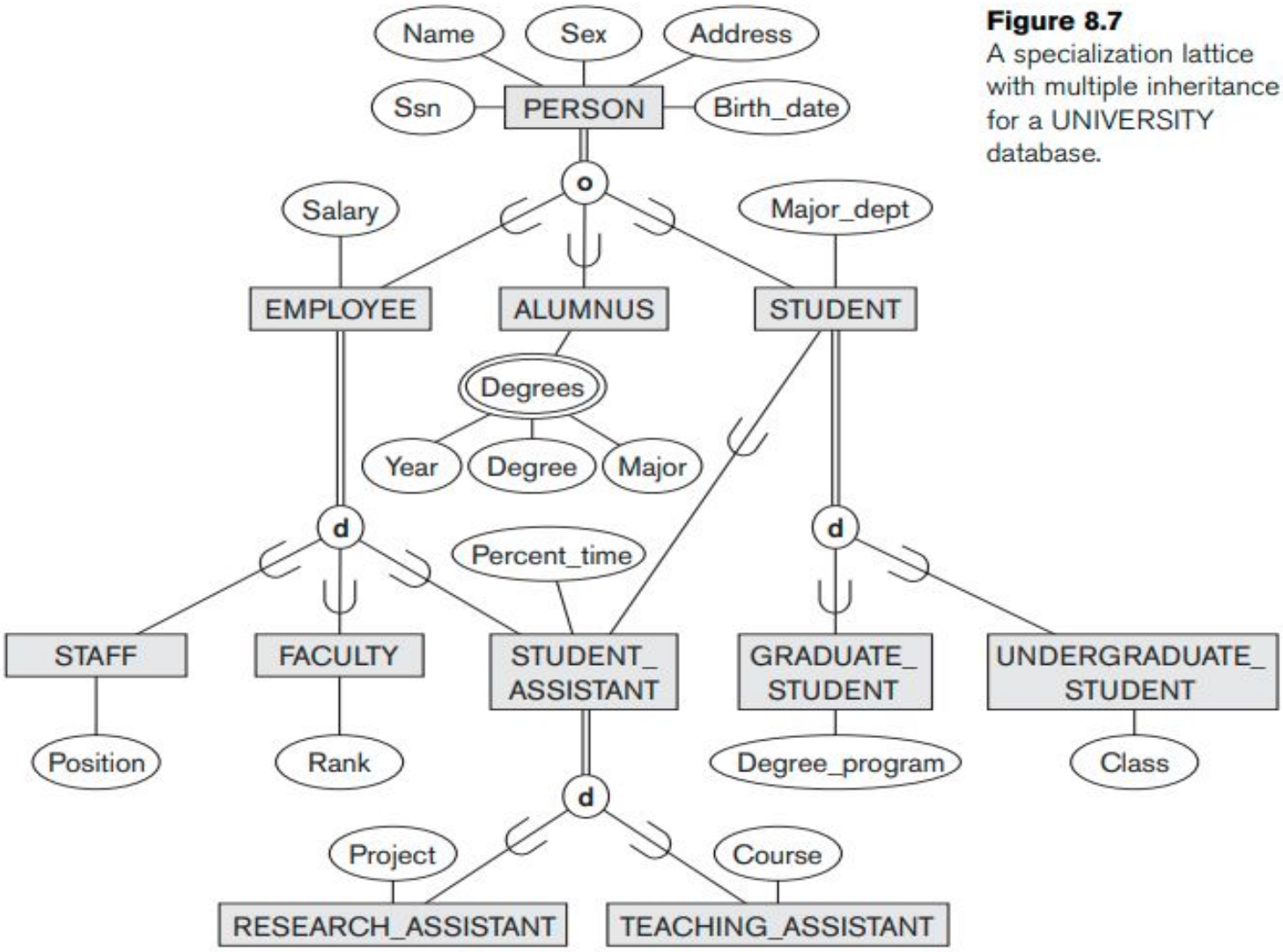


Figure 8.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

The requirements for the part of the UNIVERSITY database shown in Figure 8.7 are the following:

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 sheearned 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.

-By Ranjana J

In Figure 8.7, all person entities represented in the database are members of the PERSON entity type, which is specialized into the subclasses {EMPLOYEE, ALUMNUS, STUDENT}.

This specialization is overlapping; for example, an alumnus may also be an employee and may also be a student pursuing an advanced degree.

The subclass STUDENT is the superclass for the specialization {GRADUATE_STUDENT, UNDERGRADUATE_STUDENT}, while EMPLOYEE is the superclass for the specialization {STUDENT_ASSISTANT, FACULTY, STAFF}. Notice that STUDENT_ASSISTANT is also a subclass of STUDENT.

Finally, STUDENT_ASSISTANT is the superclass for the specialization into {RESEARCH_ASSISTANT, TEACHING_ASSISTANT}.

NOTE:

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. For example, a member of GRADUATE_STUDENT may also be a member of RESEARCH_ASSISTANT

A subclass with more than one superclass is called a shared subclass, such as ENGINEERING_MANAGER in Figure 8.6. This leads to the concept known as multiple inheritance, where the shared subclass ENGINEERING_MANAGER directly inherits attributes and relationships from multiple classes. 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.

8.3.3 Utilizing Specialization and Generalization in Refining Conceptual Schemas

8.3.3 Utilizing Specialization and Generalization in Refining Conceptual Schemas

- Now we elaborate on the differences between the specialization and generalization processes, and how they are used to refine conceptual schemas during conceptual database design.
- In the specialization process, we typically start with an entity type and then define subclasses of the entity type by successive specialization; that is, we repeatedly define more specific groupings of the entity type.
- For example, when designing the specialization lattice in Figure 8.7, we may first specify an entity type PERSON for a university database. Then we discover that three types of persons will be represented in the database: university employees, alumni, and students. We create the specialization {EMPLOYEE, ALUMNUS, STUDENT} for this purpose and choose the overlapping constraint, because a person may belong to more than one of the subclasses. We specialize EMPLOYEE further into {STAFF, FACULTY, STUDENT_ASSISTANT}, and specialize STUDENT into {GRADUATE_STUDENT, UNDERGRADUATE_STUDENT}. Finally, we specialize STUDENT_ASSISTANT into {RESEARCH_ASSISTANT, TEACHING_ASSISTANT}. This successive specialization corresponds to a top-down conceptual refinement process during conceptual schema design.So far, we have a hierarchy; then we realize that STUDENT_ASSISTANT is a shared subclass, since it is also a subclass of STUDENT, leading to the lattice.

- 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.

8.4 MODELING OF UNION TYPES USING CATEGORIES

- Union type or a Category
  - All of the superclass/subclass relationships we have seen thus far have a single superclass. A shared subclass such as ENGINEERING_MANAGER in the lattice in Figure 8.6 is the subclass in three distinct superclass/subclass relationships, where each of the three relationships has a single superclass. However, it is sometimes necessary to represent a single superclass/subclass relationship with more than one superclass, where the superclasses represent different entity types. In this case, the subclass will represent a collection of objects that is a subset of the UNION of distinct entity types; we call such a subclass a **union type or a category.**

FOR EXAMPLE

For example, 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 (holding a lien on a vehicle), 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. We display categories in an EER diagram as shown in Figure 8.8.
- The superclasses COMPANY, BANK, and PERSON are connected to the circle with the **U** symbol, which stands for the set union operation.
- An arc with the subset symbol connects the circle to the (subclass) OWNER category.

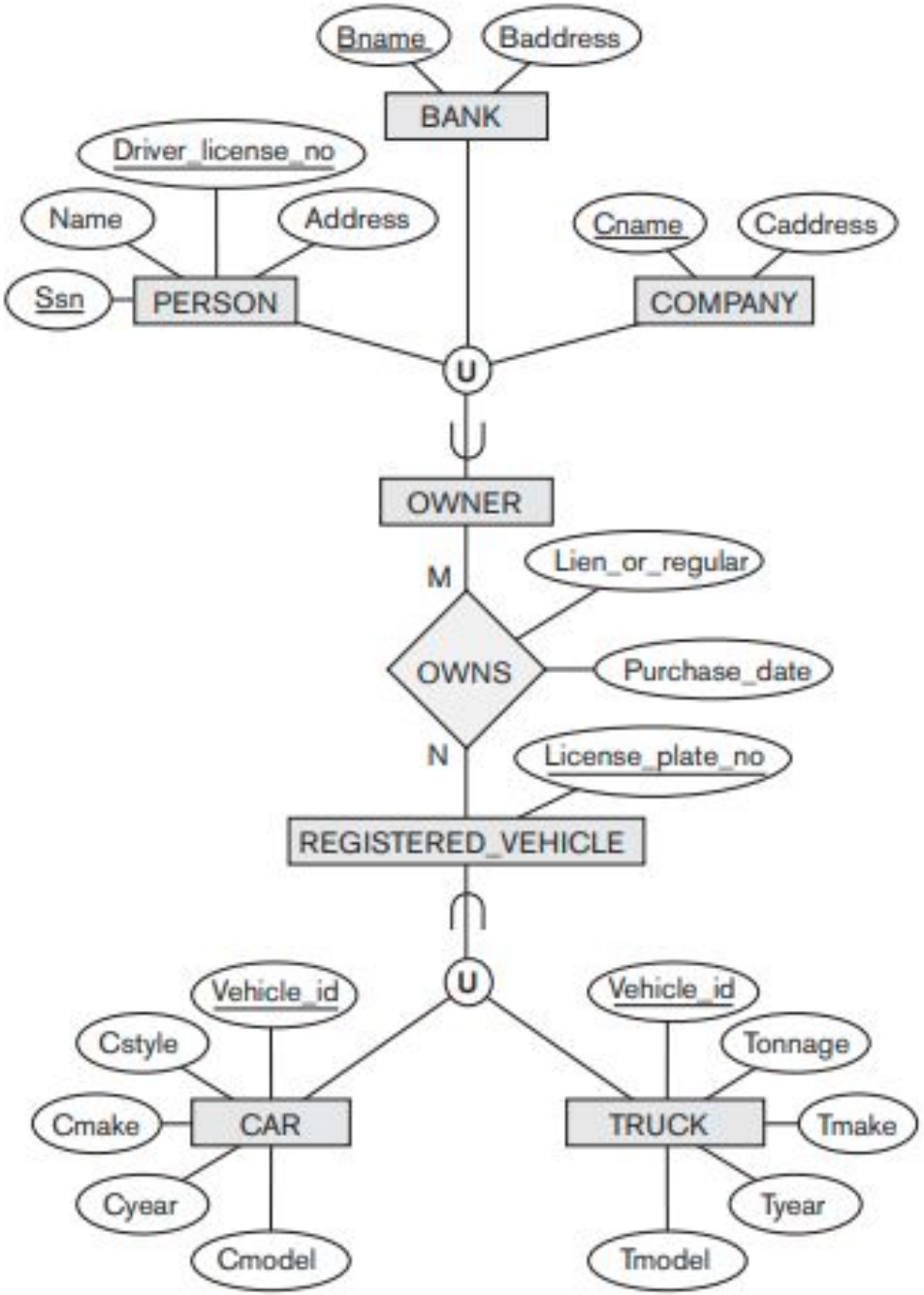


Figure 8.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

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. A total category is represented diagrammatically by a double line connecting the category and the circle, whereas a partial category is indicated by a single line.

REFERENCES:

1. FUNDAMENTALS OF Database Systems SIXTH EDITION, Ramez Elmasri,Shamkant B. Navathe.

NOTE : Refer to the references for detailed explanation(in case needed) of the topics mentioned in the content.