CHAPTER 4

Enhanced Entity-Relationship (EER) Modeling

Note: Slides, content, web links and end chapter questions are prepared from Pearson textbook (Elmasri & Navathe), and other Internet resources.

Topics of Discussion

A. Introduction to EER (Enhanced ER or Extended ER)
B. EER Model Concepts
    A. subclasses/superclasses
    B. attribute and relationship inheritance
    C. Specialization/generalization
    D. Constraints on Specialization/Generalization
    E. Specialization/Generalization Hierarchies, Lattices & Shared Subclasses
    F. categories (UNION types)
C. Knowledge Representation
D. Ontology and Semantic Web

EER - Introduction

- The ER modeling concepts discussed in Chapter 3 are sufficient for representing many database schemas for *traditional* database applications.
- The newer applications of database technology, such as databases for engineering design and manufacturing, telecommunications, complex software systems, and geographic information systems, require design more accurate schemas and constraints more precisely.
- This led to the development of additional semantic data modeling concepts. It related to the knowledge representation, area of artificial intelligence and the object modeling area in software engineering.
- ER model required to be enhanced to include these concepts, which leads to the enhanced ER (EER) model.

Subclasses and Superclasses

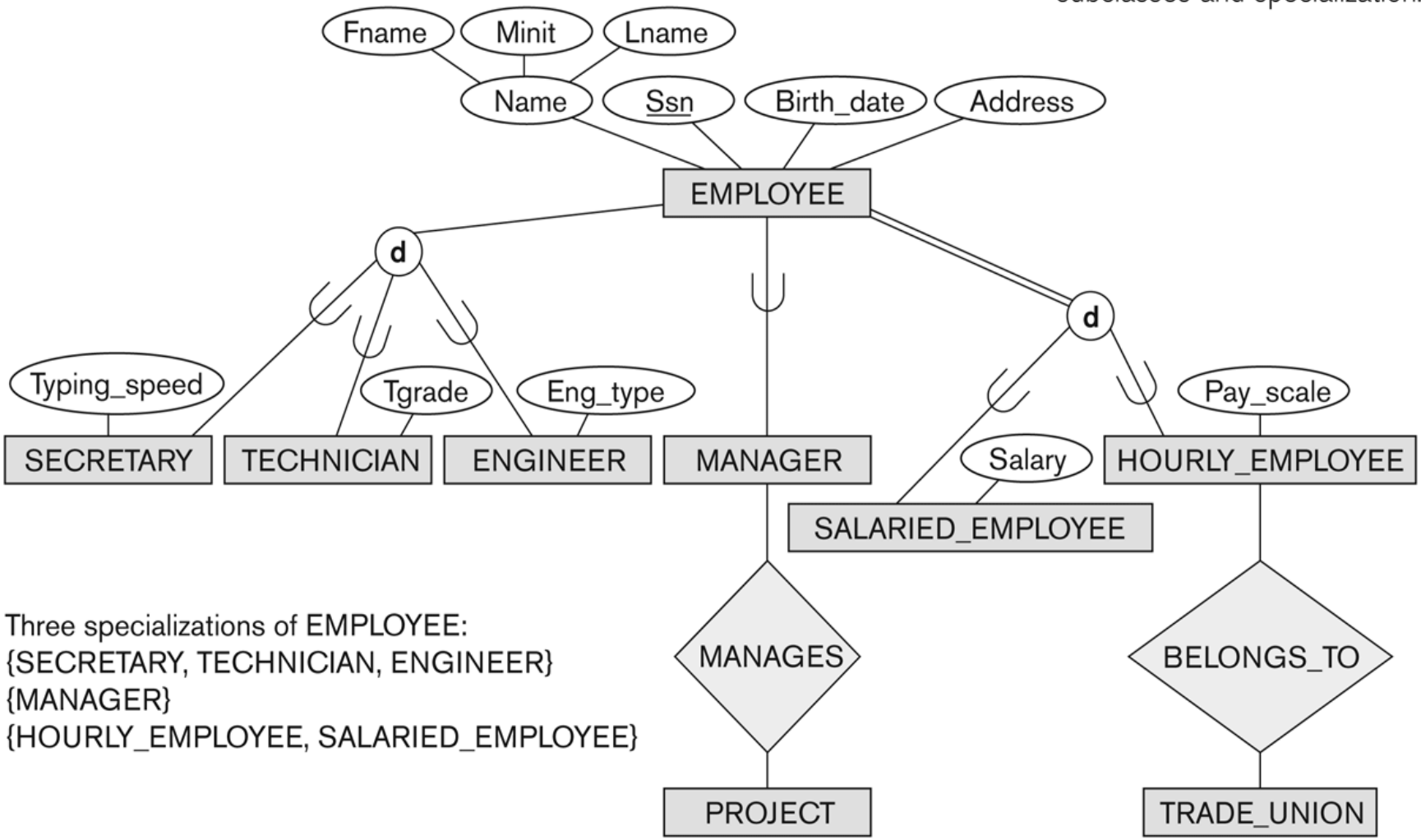
- An entity may have additional meaningful subgroupings of its entities
  - Example: EMPLOYEE may be further grouped into:
    - SECRETARY, ENGINEER, TECHNICIAN, …
      - Based on the EMPLOYEE's Job
    - MANAGER
      - EMPLOYEEs who are managers (the role they play)
    - SALARIED_EMPLOYEE, HOURLY_EMPLOYEE
      - Based on the EMPLOYEE's method of pay
- EER diagrams extend ER diagrams to represent these additional subgroupings, called *subclasses* or *subtypes*

Subclasses and Superclasses

- Each of these subgroupings is a subset of EMPLOYEE entities
- Each is called a subclass of EMPLOYEE
- EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass relationships:
  - EMPLOYEE/SECRETARY
  - EMPLOYEE/TECHNICIAN
  - EMPLOYEE/MANAGER
  - …

Subclasses and Superclasses

Figure 4.1
EER diagram notation to represent subclasses and specialization.



Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Subclasses and Superclasses

- These are also called IS-A relationships
  - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ….
- Note: An entity that is member of a subclass represents the same real-world entity as member of the superclass:
  - The subclass member is the same entity but in a *distinct specific role.*
  - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass.
  - Distinct records of subclasses (child) are related with super class (parent) with Key attribute.
  - It means **Ssn** will be a primary key in EMPOYEE entity and **Ssn** will be Foreign key in SECRETARY, TECHNICIAN and ENGINEER subclass entities.

Subclasses and Superclasses

- A member of the superclass can be optionally included as a member of any number of its subclasses. For example:
  - A salaried employee who is also an engineer belongs to the two subclasses:
    - ENGINEER, and
    - SALARIED_EMPLOYEE
  - A salaried employee who is also an engineer, and manager for any project 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

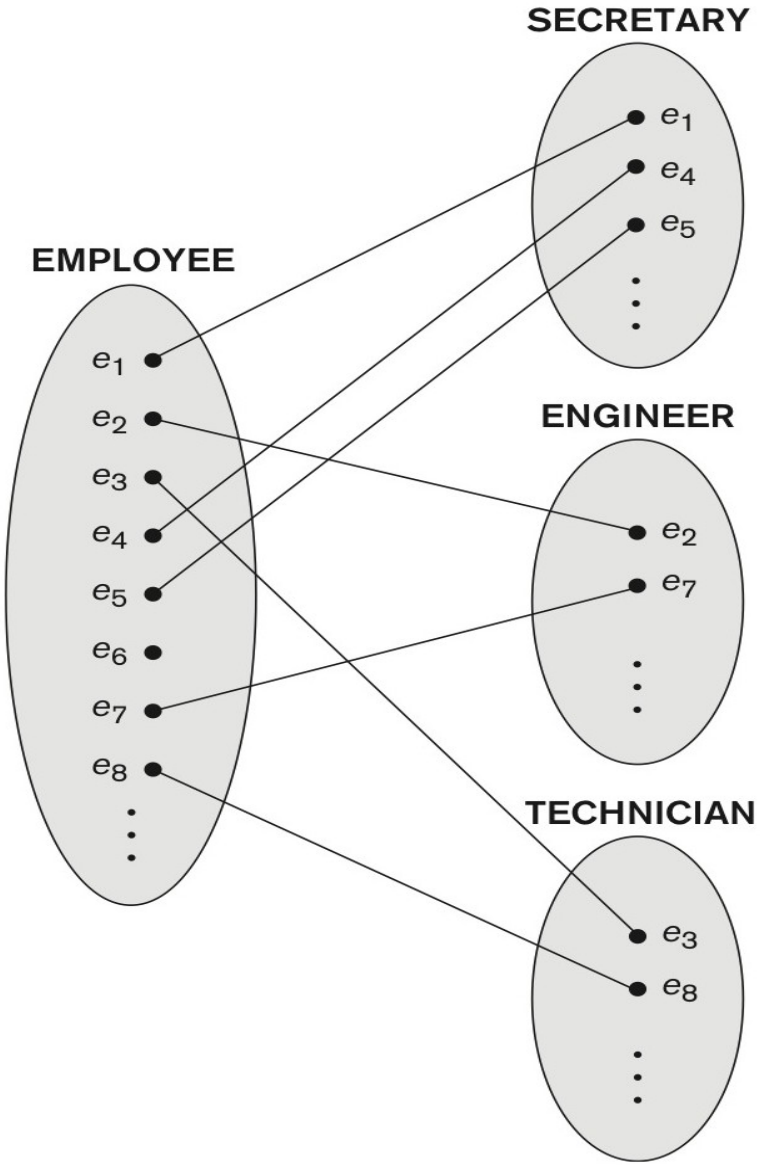
- An entity that is member of a subclass *inherits*
  - All attributes of the entity as a member of the superclass
  - All relationships of the entity as a member of the superclass
- Example:
  - In the previous slide, SECRETARY (as well as TECHNICIAN and ENGINEER) inherit the attributes Fname, Ssn, …, from EMPLOYEE.
  - Every SECRETARY entity will have values for the inherited attributes.

Specialization

- Specialization is 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.*
  - Example: MANAGER is a specialization of EMPLOYEE based on the role the employee plays
    - May have several specializations of the same superclass

Specialization

Figure 4.2 Instances of specialization.



Specialization

- Example: Another specialization of EMPLOYEE based on *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* 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.

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

Generalization

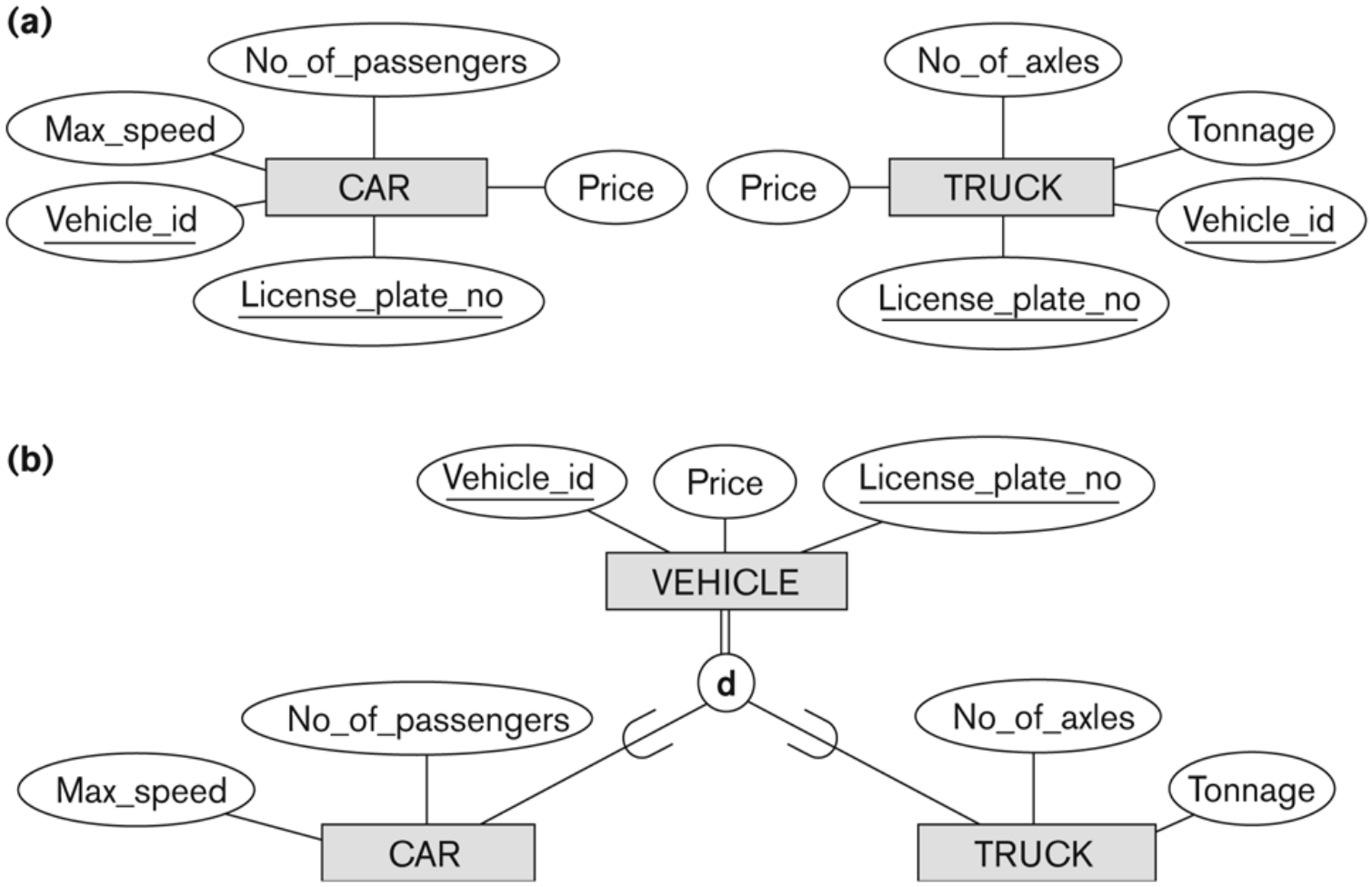


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Generalization and Specialization

- Diagrammatic notations are sometimes used to distinguish between generalization and specialization
  - Arrow pointing to the generalized superclass represents a generalization.
  - Arrows pointing to the specialized subclasses represent a specialization.
  - A superclass or subclass represents a collection (or set or grouping) of entities. It also represents a particular *type of entity*
  - Shown in rectangles in EER diagrams (as are entity types)
  - We can call all entity types (and their corresponding collections) ***classes***, whether they are entity types, superclasses, or subclasses

Constraints on Specialization and Generalization

- Let see constraints that apply to a single specialization or a single generalization.

  (For brevity, our discussion refers only to *specialization* even though it applies to *both* specialization and generalization.)

  1. Predicate-defined (or condition-defined): based on some predicate. E.g., based on value of an attribute, say, Age.
  2. Attribute-defined: based on the attribute. Like Job_type. (It shows next to the line drawn from the superclass toward the subclasses – in diagram).
  3. User-defined: membership is defined by the user on an entity-by-entity basis.

Constraints on Specialization and Generalization

1. Predicate-defined (or condition-defined)
   - If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called predicate-defined (or condition-defined) subclasses
   - Condition is a constraint that determines subclass members
   - Display a predicate-defined subclass by writing the predicate condition next to the line attaching the subclass to its superclass

Constraints on Specialization and Generalization

2. Attribute-defined:
   - If all subclasses in a specialization have membership condition on same attribute of the superclass, specialization is called an 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.
   - We display an attribute-defined specialization by placing the defining attribute name next to the arc from the circle to the superclass, as shown in figure 4.4

Constraints on Specialization and Generalization

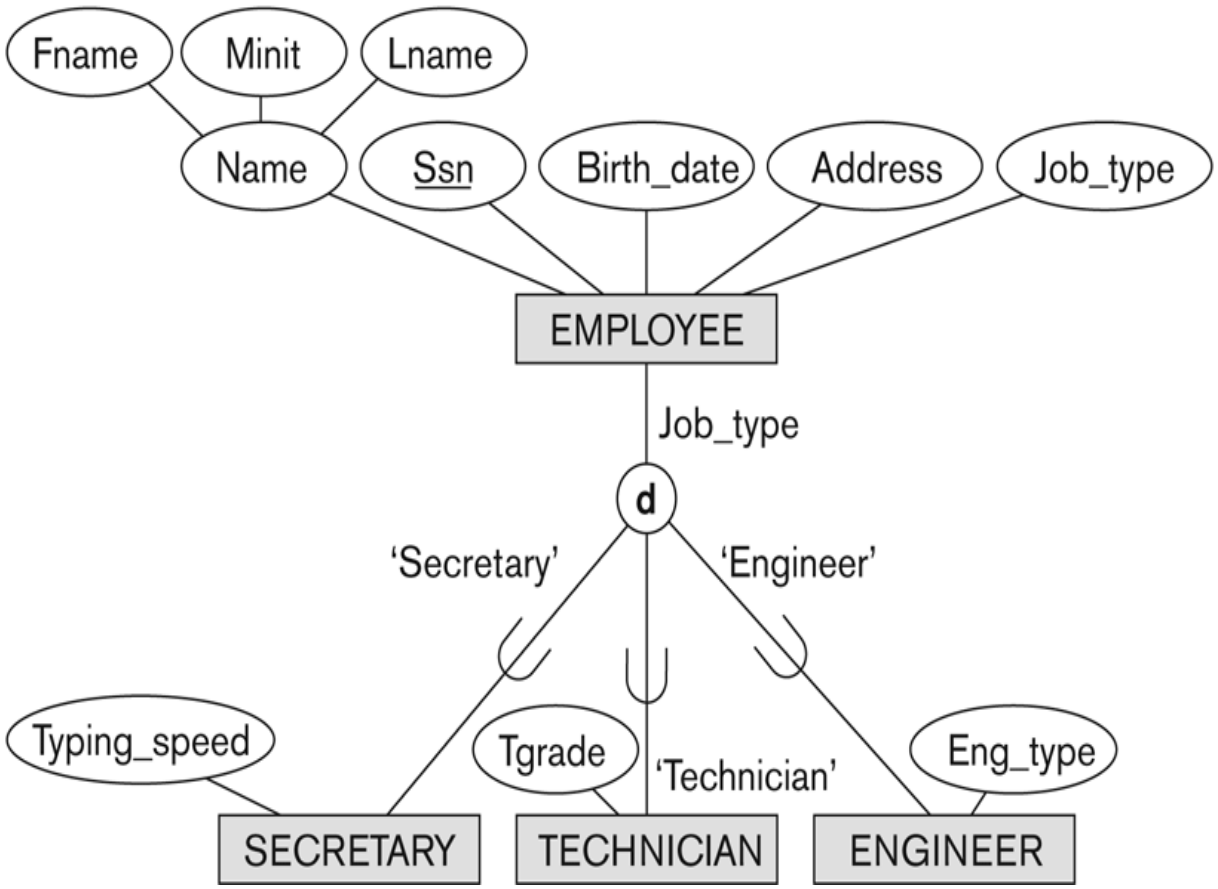
3. User-defined: If no condition determines membership, the subclass is called user-defined
   - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
   - Membership in the subclass is *specified individually for each entity* in the superclass by the user.

Constraints on Specialization and Generalization

- Basic constraints can apply to a specialization /generalization.
- 1) 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-4.4)

Displaying an attribute-defined specialization in EER diagrams

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



Constraints on Specialization and Generalization

- 2) Overlapping Constraint:
  - If not disjoint, specialization is *overlapping*:
    - that is the same (real –world) entity may be a member of more than one subclass of the specialization
    - Specified by ***o*** in EER diagram. (figure-4.5)

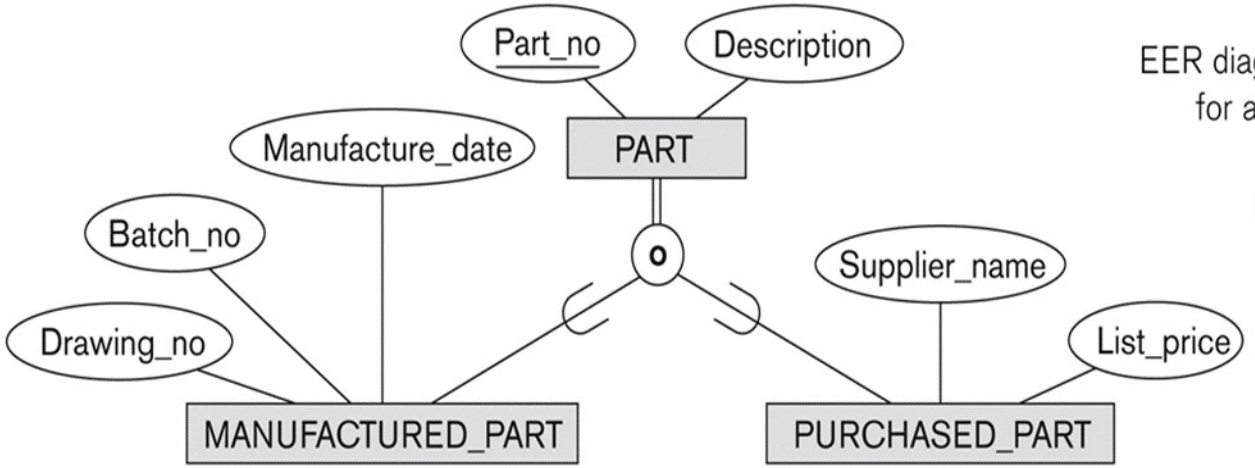


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Constraints on Specialization and Generalization

- 3) Completeness (Exhaustiveness) 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***
- 4) Partial Constraint:
  - *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

- Hence, we have four types of specialization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial

- Note: Generalization usually is total because the superclass is derived from the subclasses.

Specialization/Generalization Hierarchies, Lattices & 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***)

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses

- For example, in Figure 4.6 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.
  Shared Subclass “Engineering_Manager”

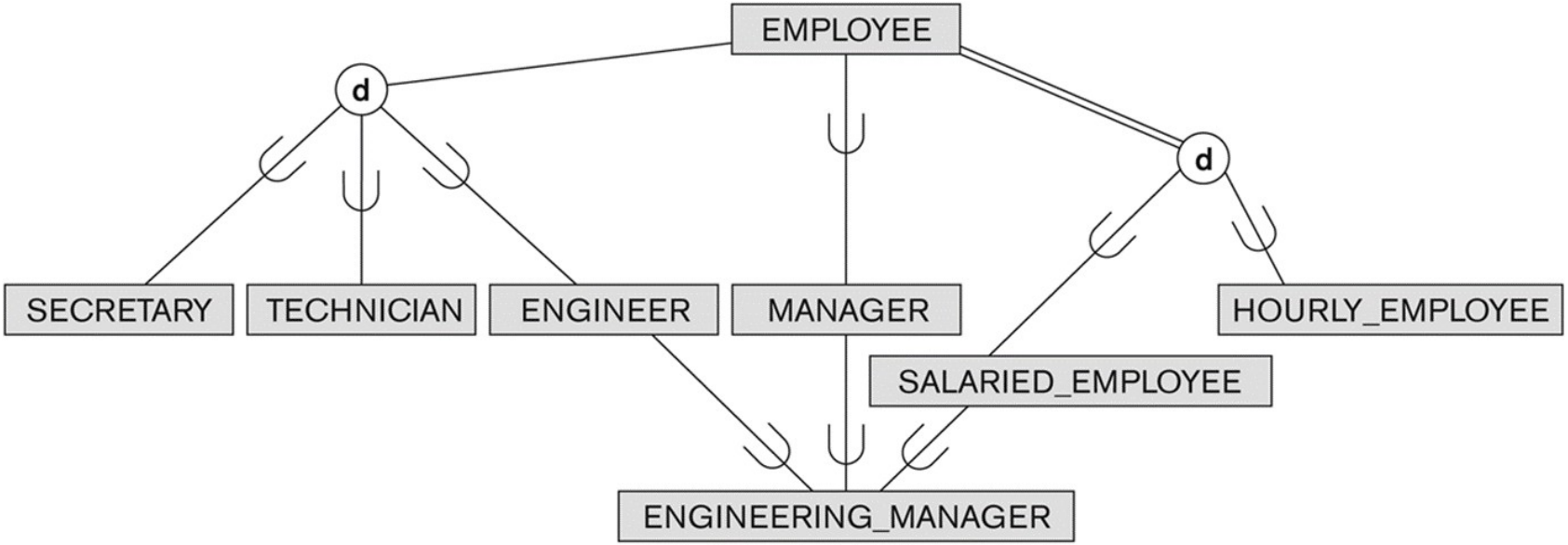


Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses

- In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses.
- A subclass with more than one superclass is called a shared subclass (multiple inheritance)
- Can have:
  - *specialization* hierarchies or lattices, or
  - *generalization* hierarchies or lattices,
  - depending on how they were *derived*

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses

- We just use *specialization* (to stand for the end result of either specialization or generalization)
- 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
  - called a *bottom up* conceptual synthesis process
- In practice, a *combination of both processes* is usually employed

One more example : Specialization / Generalization Lattice (UNIVERSITY)

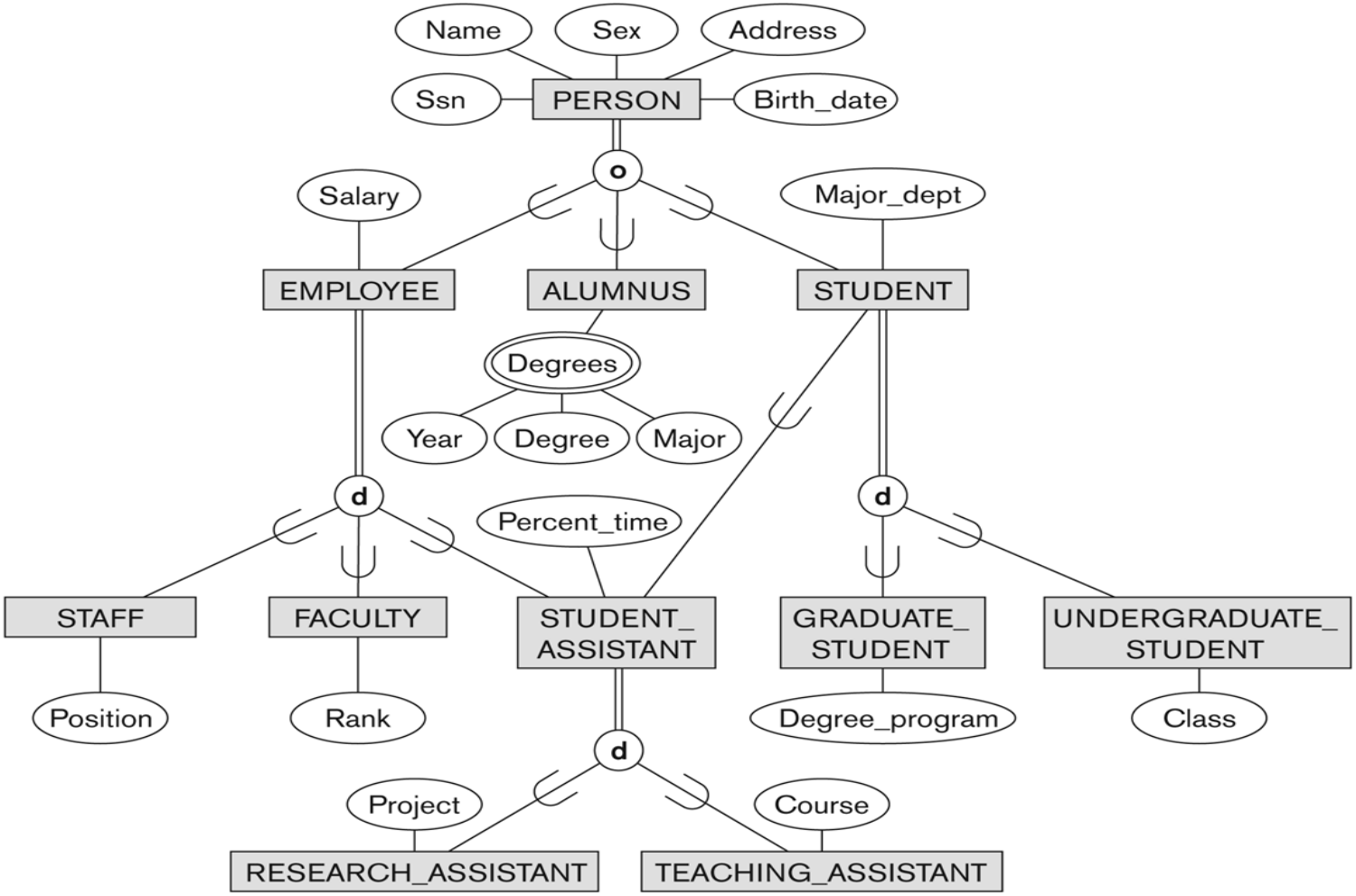


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

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
- Superclasses can represent different entity types
- Such a subclass is called a category or UNION TYPE

Two categories (UNION types): OWNER, REGISTERED_VEHICLE

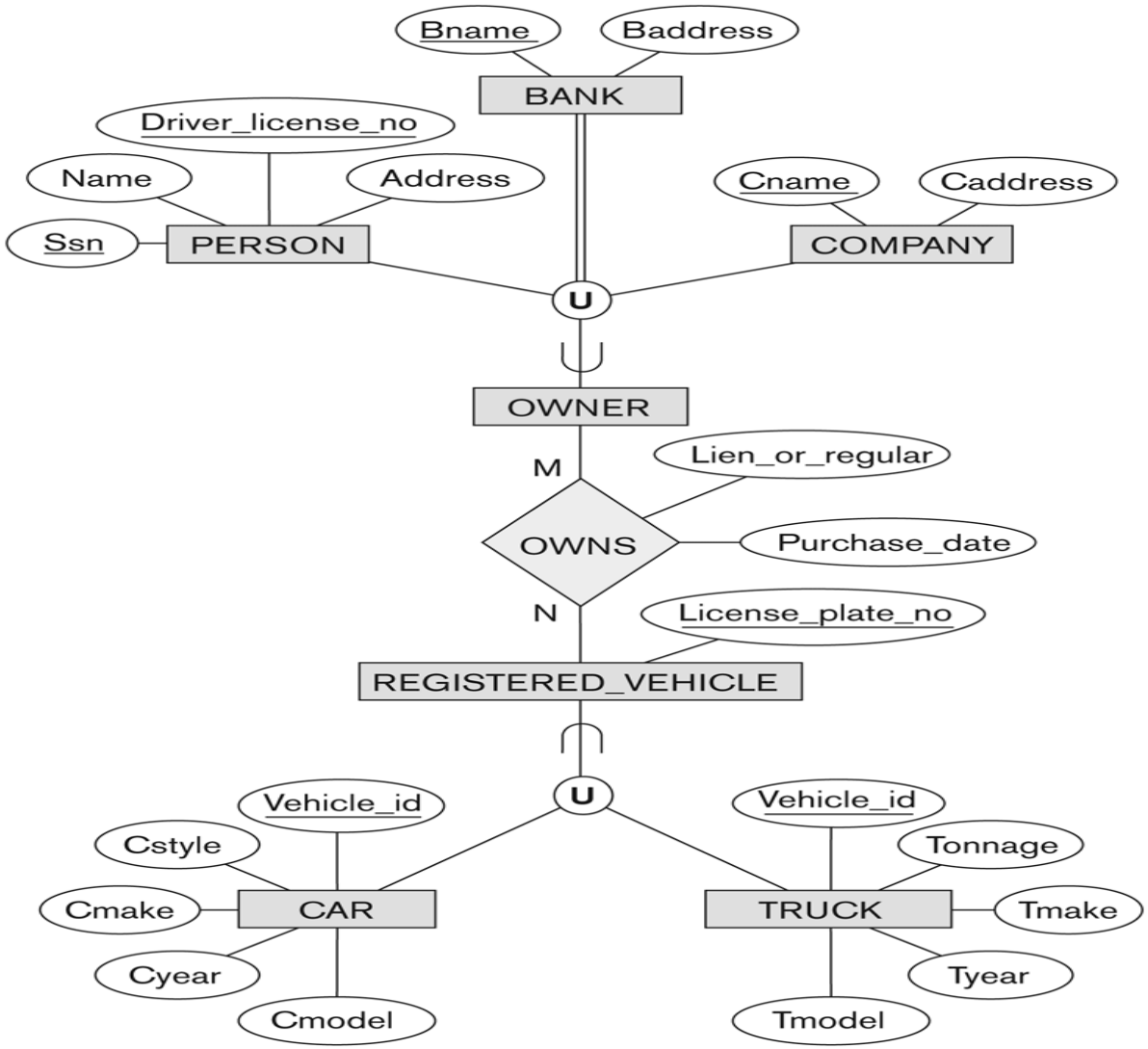


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Categories (UNION TYPES)

- Example: In a database for vehicle registration, a vehicle owner can be a PERSON, a BANK (holding a lien on a vehicle) or a COMPANY. (see figure 4.8)
  - 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 ***(****typically just 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

Knowledge Representation (KR)

- Deals with modeling and representing a certain domain of knowledge.
- Typically done by using some formal model of representation and by creating an Ontology.
- An ontology for a specific domain of interest describes a set of concepts and interrelationships among those concepts.
- An Ontology serves as a “schema” which enables interpretation of the knowledge in a “knowledge-base”.

Knowledge Representation (KR)

COMMON FEATURES (between KR and Data Models):

- Both use similar set of abstractions – classification, aggregation, generalization, and identification.
- Both provide concepts, relationships, constraints, operations and languages to represent knowledge and model data.

DIFFERENCES:

- KR has broader scope tries to deal with missing and incomplete knowledge, default and common-sense knowledge etc.
- KR schemes typically include rules and reasoning mechanisms for inferencing.

Knowledge Representation (KR)

DIFFERENCES (continued):

- Most KR techniques involve data and metadata. In data modeling, these are treated separately
- KR is used in conjunction with artificial intelligence systems to do decision support applications

(For more details on spatial, temporal and multimedia data modeling, will be discussed at Ch-26.)

Ontologies and Semantic web

- In recent years, the amount of computerized data and information available on the Web has spiraled out of control. Many different models and formats are used.
- In addition to the data models, (which we are discussing in this course) much information is stored in the form of documents, which have considerably less structure than database information does.
- One ongoing project that is attempting to allow information exchange among computers on the Web is called the Semantic Web, which attempts to create knowledge representation models that are quite general in order to allow meaningful information exchange and search among machines.
- The concept of ontology is considered to be the most promising basis for achieving the goals of the Semantic Web and is closely related to knowledge representation.

Ontologies and Semantic web

- Ontology originated in the fields of philosophy and metaphysics. The study of ontologies attempts to describe the concepts and relationships that are possible in reality through some common vocabulary; therefore, it can be considered as a way to describe the knowledge of a certain community about reality.
- One commonly used definition of ontology is 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.

- *(For details on use of Ontologies. You can refer see Sections 27.4.3 and 27.7.4. and several internet resources. That is not included this Fall 21 semester planning)*

End Chapter Questions

- 4.1. What is a subclass? When is a subclass needed in data modeling?
- 4.2. Define the following terms:
  superclass of a subclass, superclass/subclass relationship, IS-A relationship, specialization, generalization, category, specific (local) attributes, and specific relationships.
- 4.3. Discuss the mechanism of attribute/relationship inheritance. Why is it useful?
- 4.8. What is the difference between specialization and generalization? Why do we not display this difference in schema diagrams?