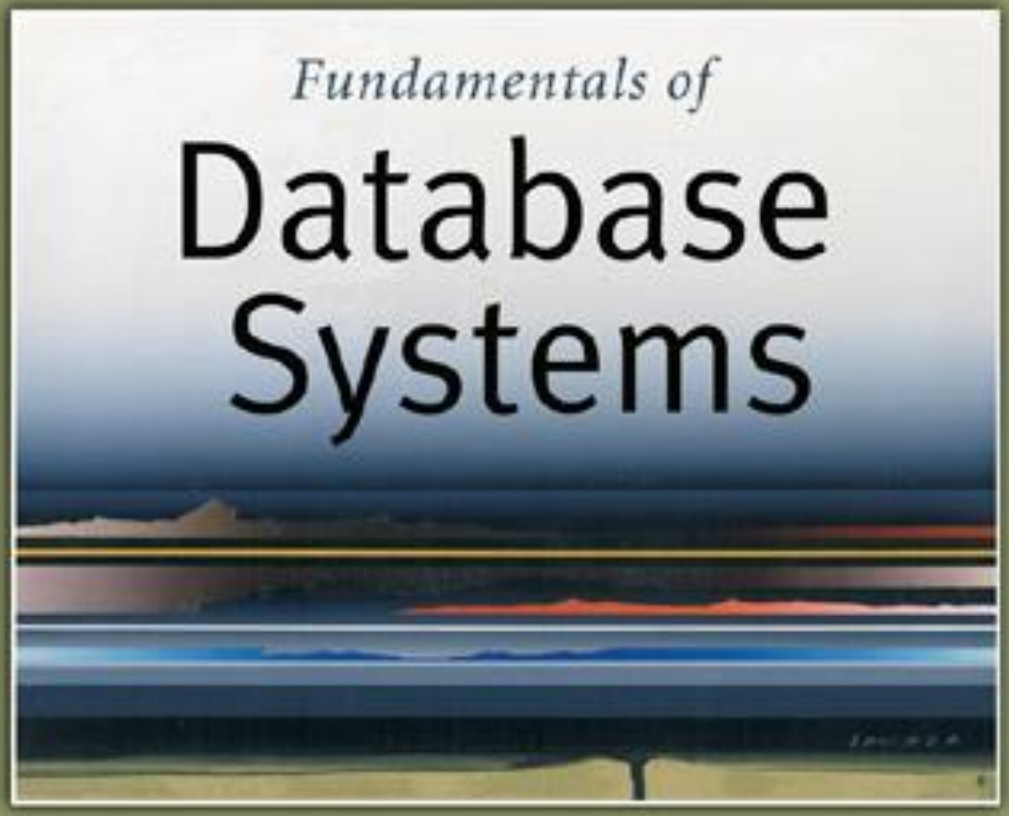
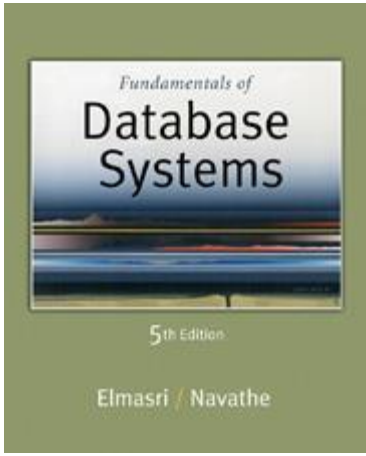


5th Edition

Elmasri / Navathe

Chapter 4

Enhanced Entity-Relationship (EER) Modeling



Chapter Outline

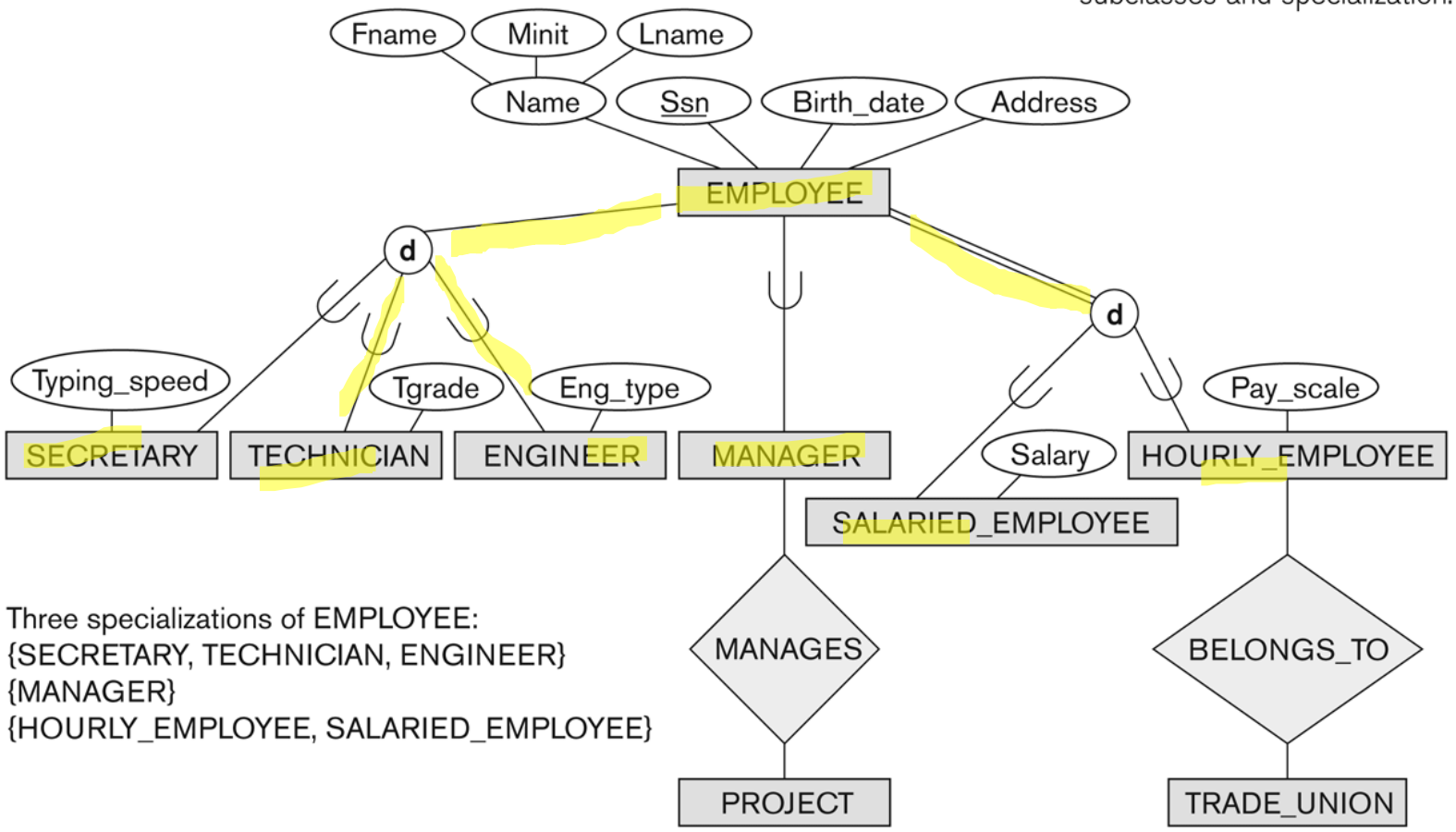
- EER stands for Enhanced ER or Extended ER
- EER Model Concepts
  - Includes all modeling concepts of basic ER
  - Additional concepts:
    - subclasses/superclasses
    - specialization/generalization
    - categories (UNION types)
    - attribute and relationship inheritance
  - These are fundamental 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

Subclasses and Superclasses (1)

- An entity type 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
    - 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

Figure 4.1
EER diagram notation to represent subclasses and specialization.



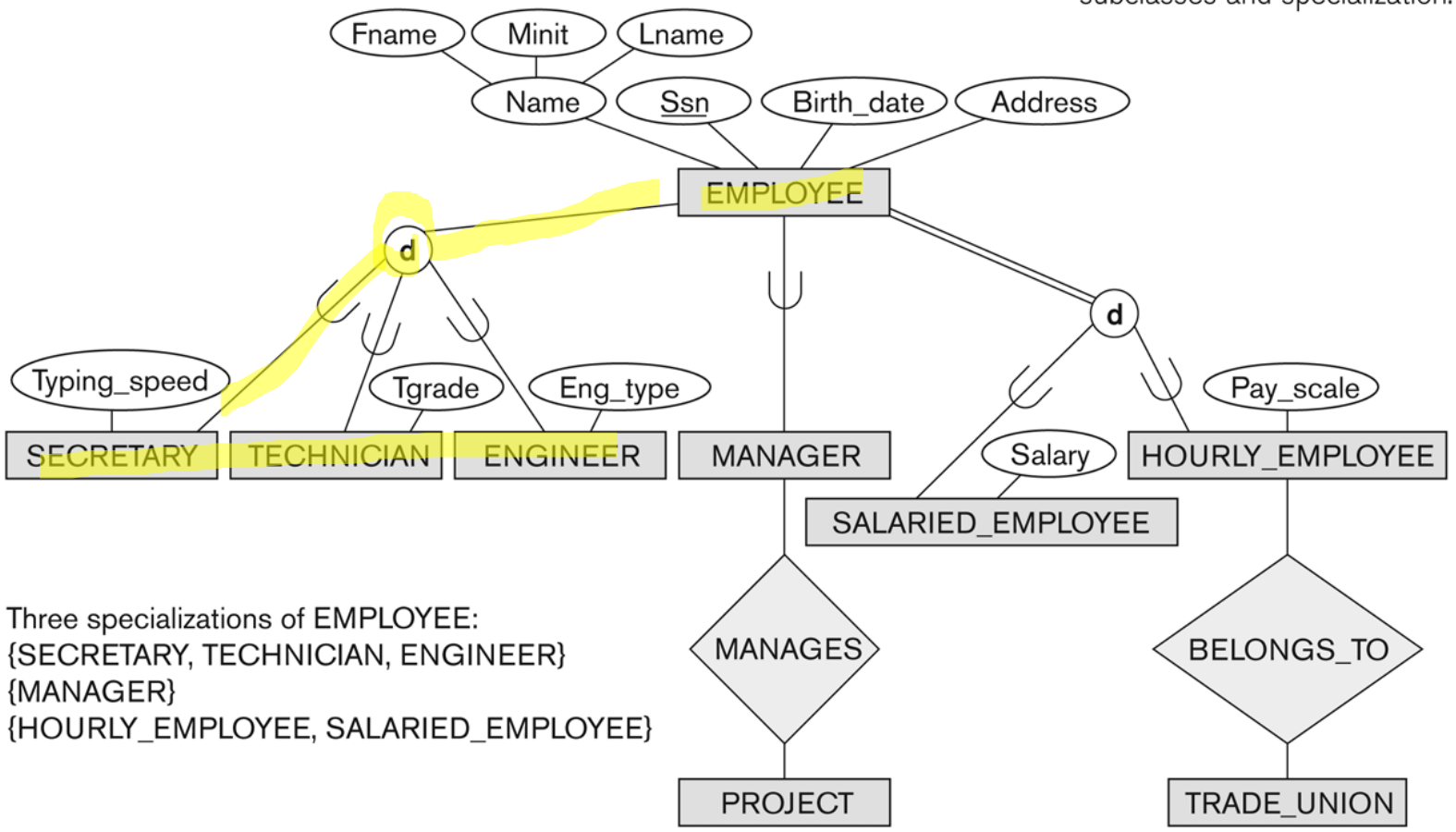
Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Subclasses and Superclasses (2)

- 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 (3)

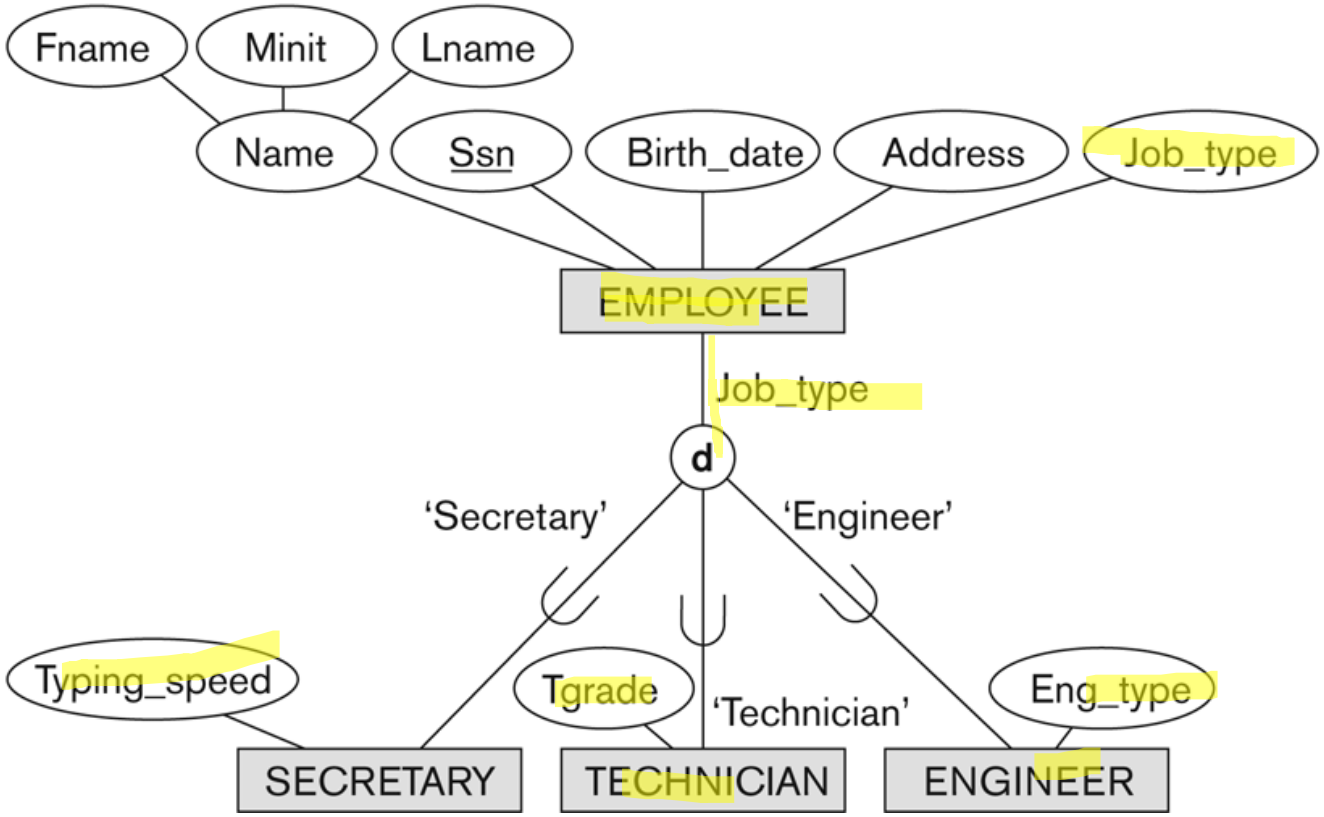
- These are also called **IS-A relationships**
  - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ….
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
  - The subclass member is the same entity in a *distinct specific role*
  - **An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass**
  - **A member of the superclass can be optionally included as a member of any number of its subclasses**

Subclasses and Superclasses (4)

- 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

Representing Specialization in EER Diagrams

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



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 Name, SSN, …, from EMPLOYEE
  - Every SECRETARY entity will have values for the inherited attributes

Specialization (1)

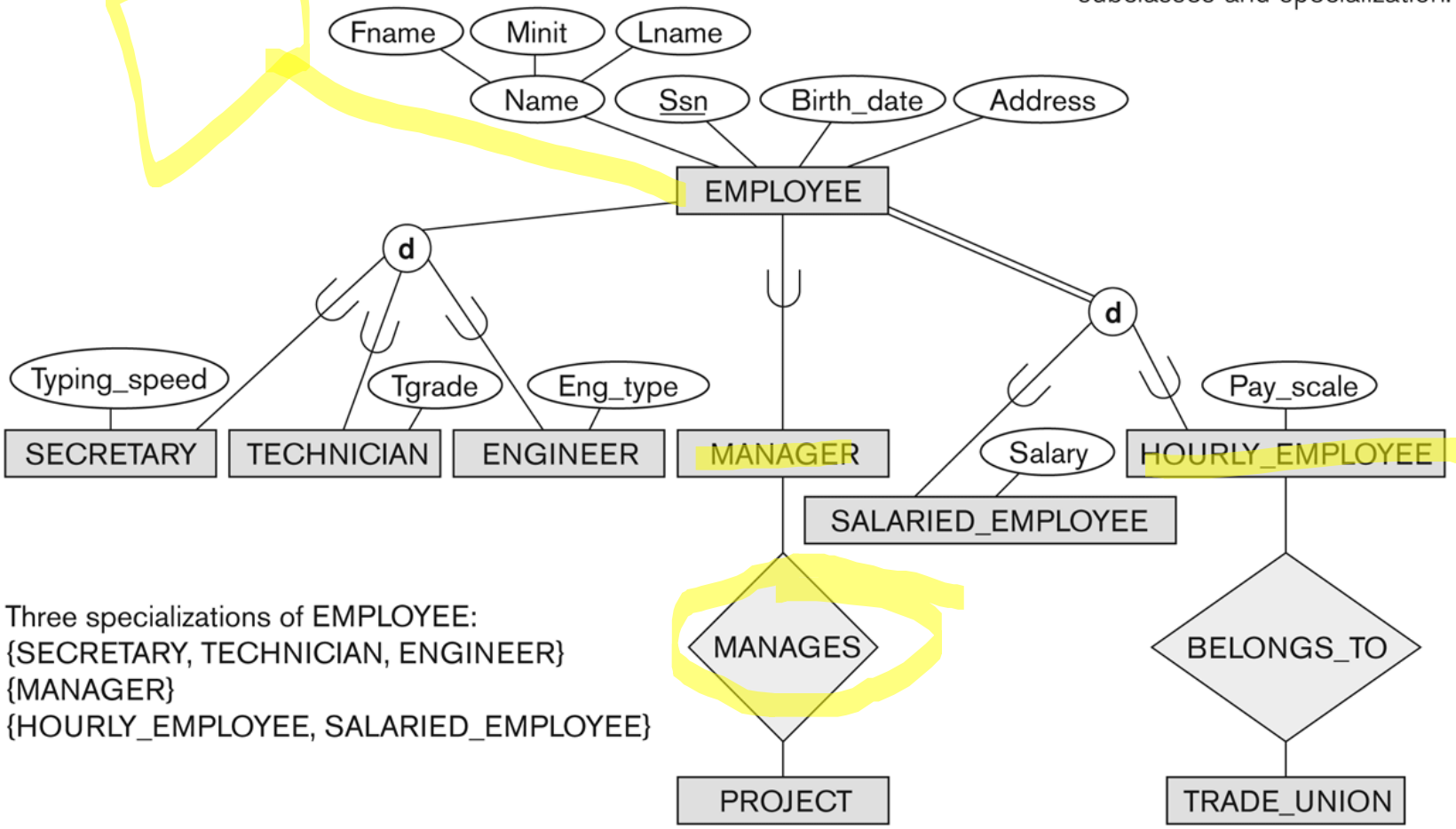
- Specialization is the process of **defining a set of subclasses of a superclass**
- The set of subclasses is **based upon some distinguishing characteristics** of the entities in the superclass
  - Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type.*
    - May have **several specializations of the same superclass**

Specialization (2)

- 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

Specialization (3)

Figure 4.1
EER diagram notation to represent subclasses and specialization.



Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_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, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

Generalization (2)

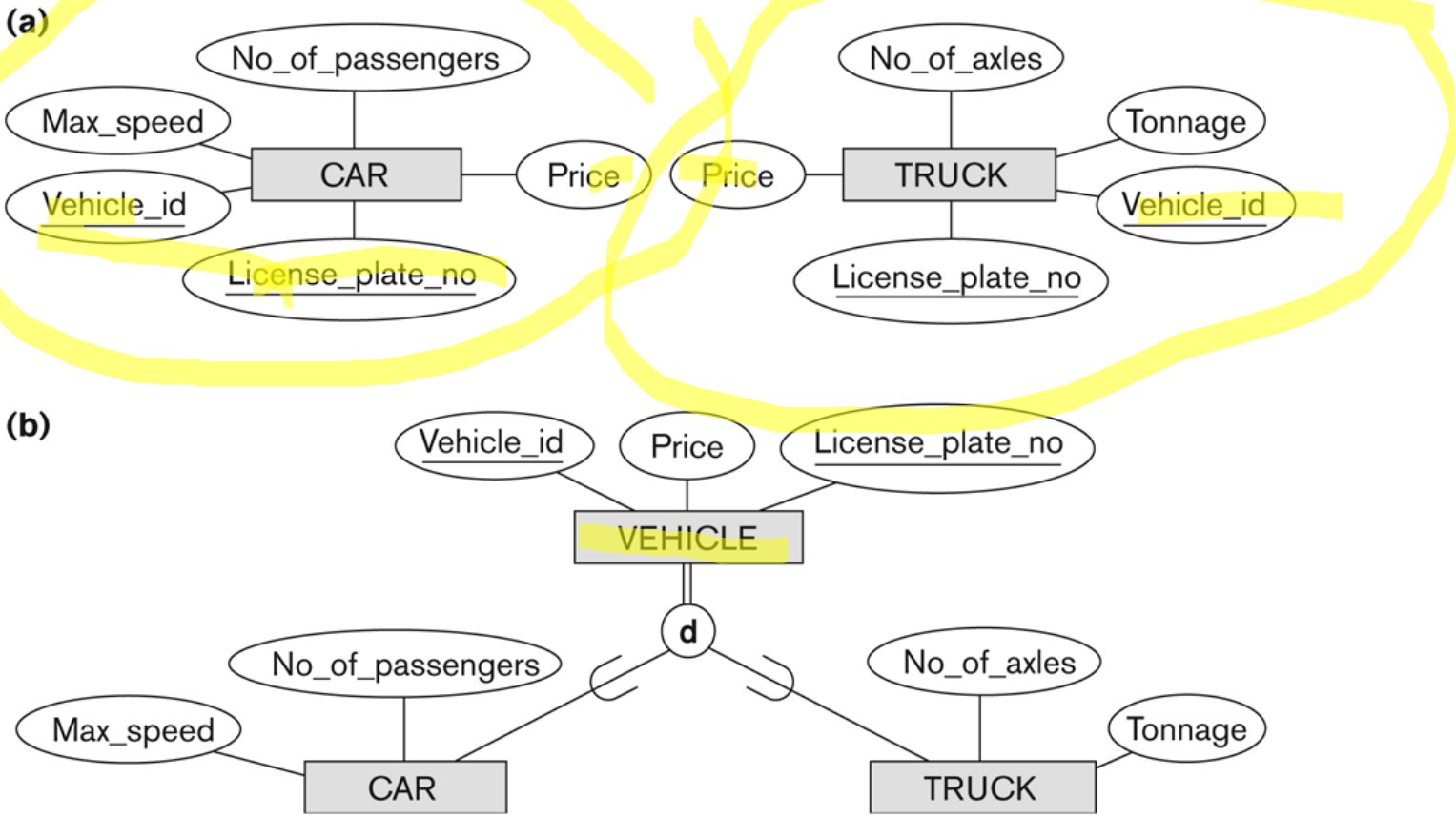


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Generalization and Specialization (1)

- Diagrammatic notation 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
  - We *do not use* this notation because it is often **subjective** as to which process is more appropriate for a particular situation
  - We advocate not drawing any arrows

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses (3)

- 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

Constraints on Specialization and Generalization (1)

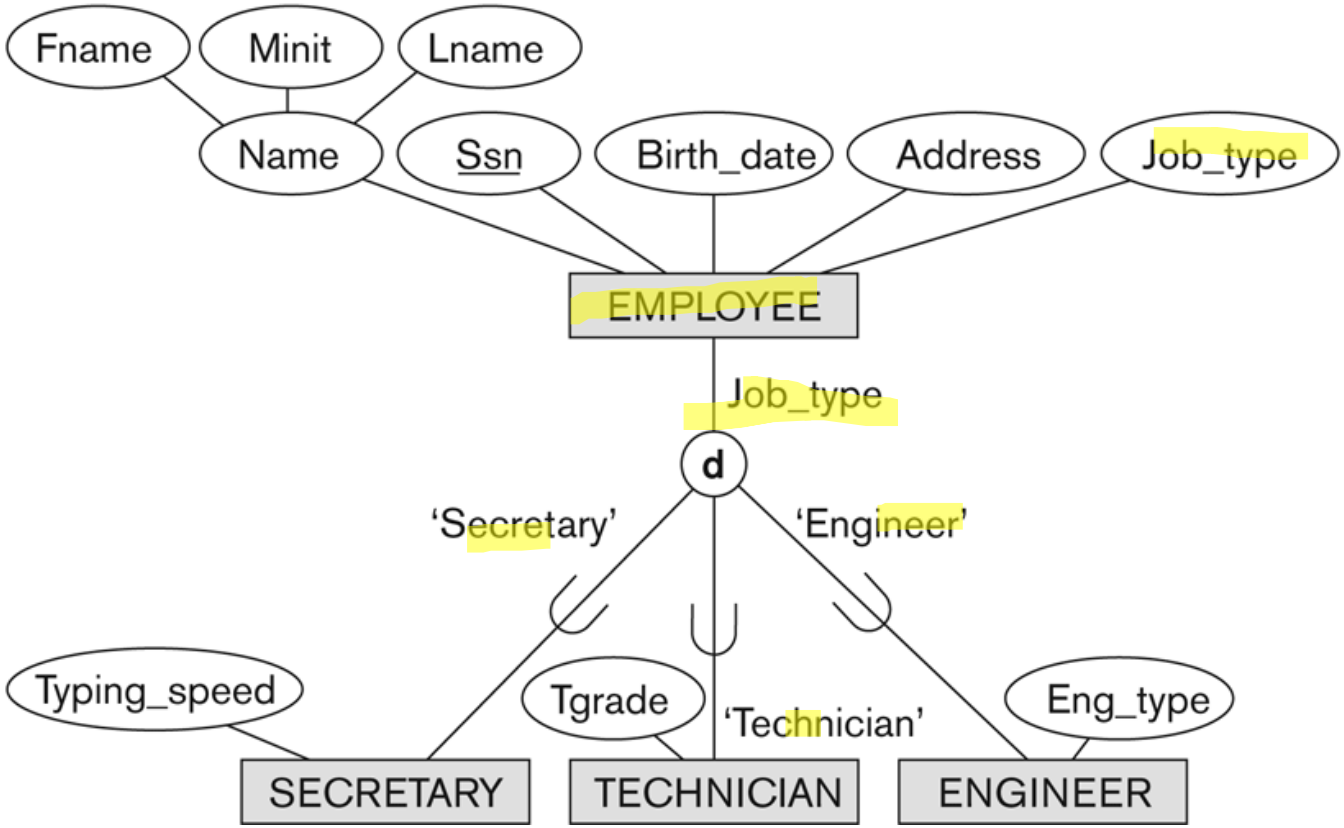
- If we can determine exactly those entities that will become members of each subclass **by a condition,** the subclasses are called **predicate-defined (or condition-defined) subclasses**
  - Condition is a constraint that determines subclass members
  - Display a predicate-defined subclass by writing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

Constraints on Specialization and Generalization (2)

- 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
- 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

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 (3)

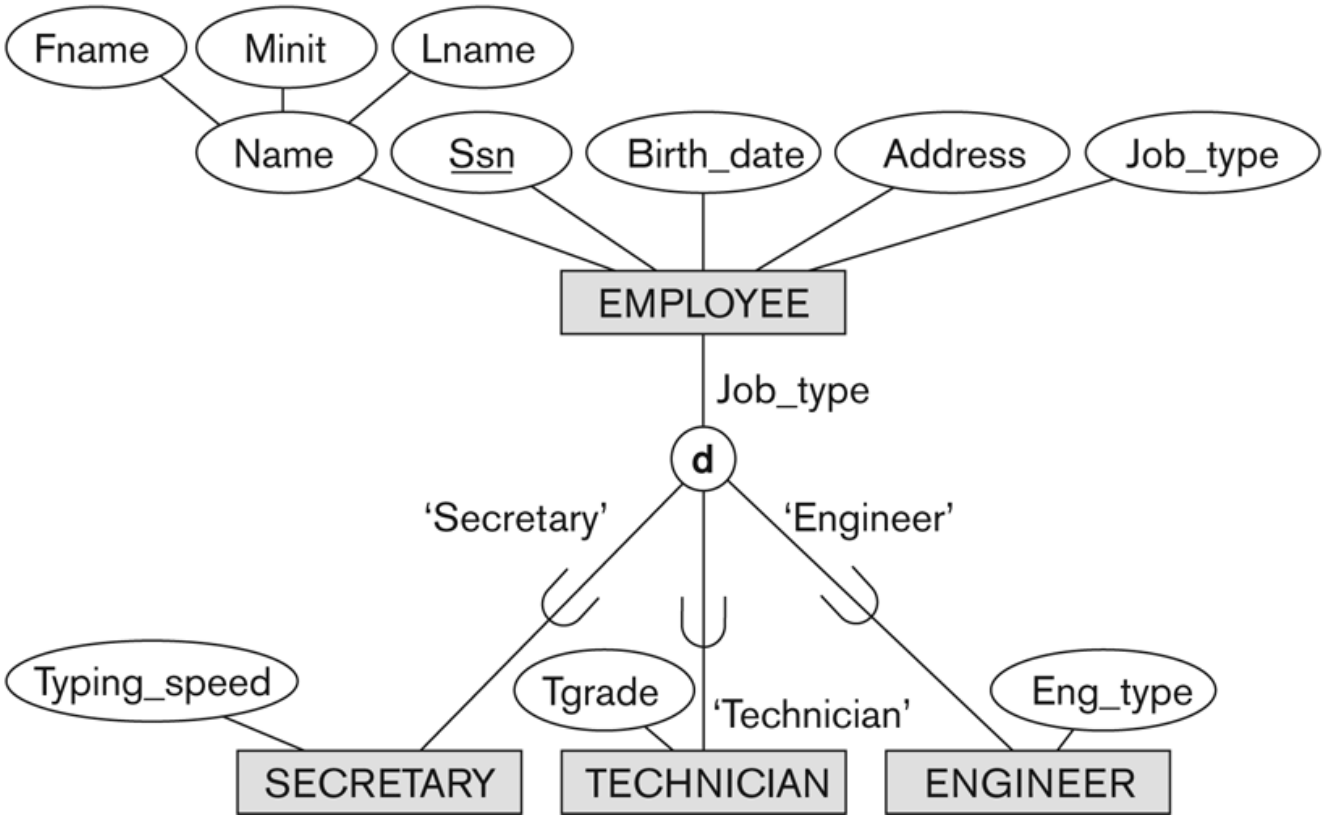
- Two basic constraints can apply to a specialization/generalization:
  - Disjointness Constraint:
  - Completeness Constraint:

Constraints on Specialization and Generalization (4)

- 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
  - If not disjoint, specialization is *overlapping*:
    - that is the same entity **may be a member of more than one** subclass of the specialization
  - Specified by ***o*** in EER diagram

Example of disjoint partial Specialization

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



Example of overlapping total Specialization

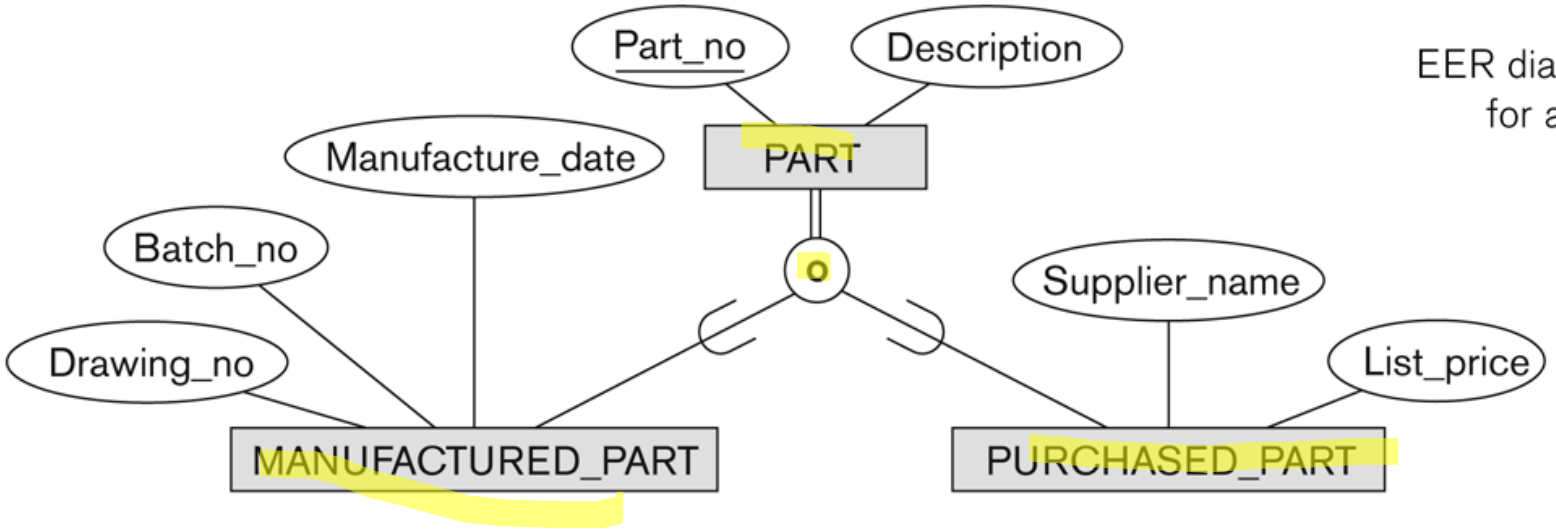


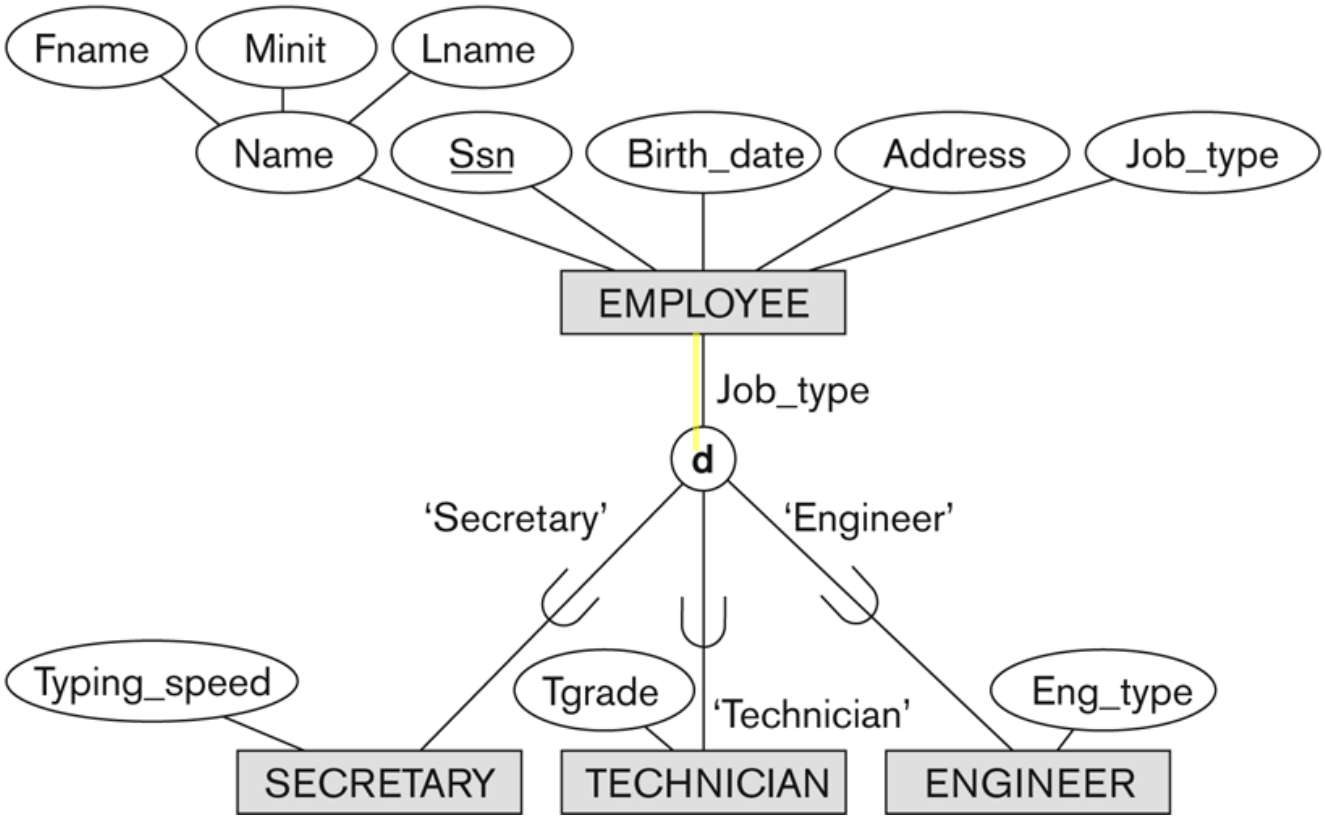
Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Constraints on Specialization and Generalization (5)

- Completeness 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***
  - *Partial* **allows an entity not to belong to any of the subclasses**
  - Shown in EER diagrams by a single line

Example of disjoint partial Specialization

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



Example of overlapping total Specialization

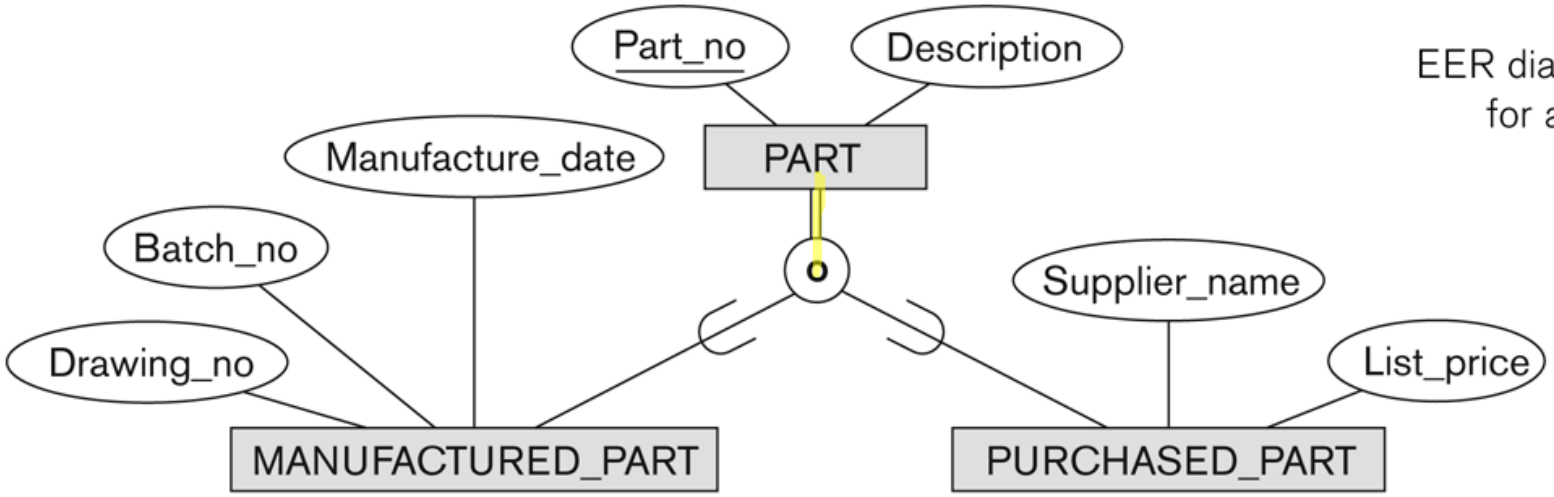


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Constraints on Specialization and Generalization (6)

- 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.

Generalization (2)

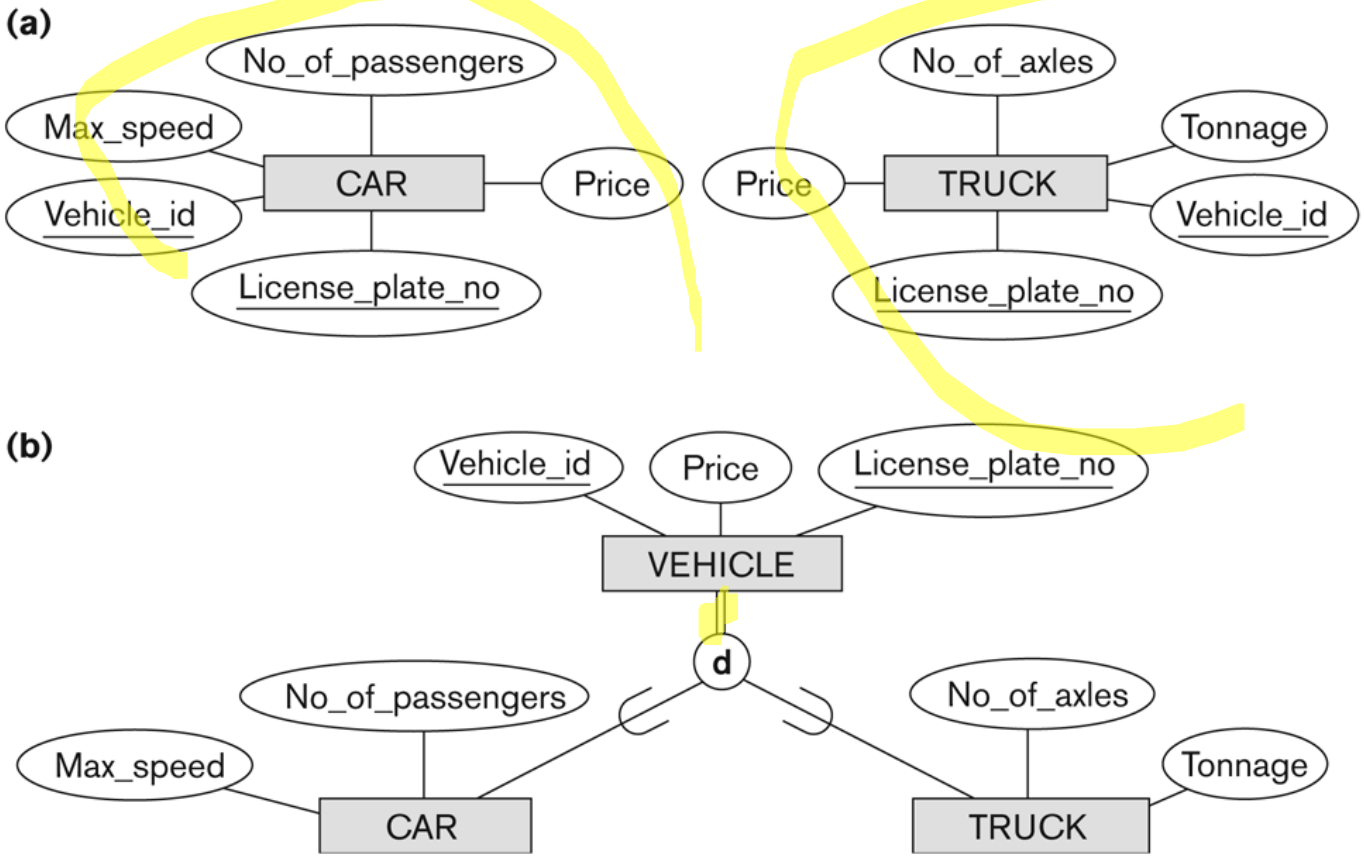


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses (1)

- 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***)

Shared Subclass “Engineering_Manager”

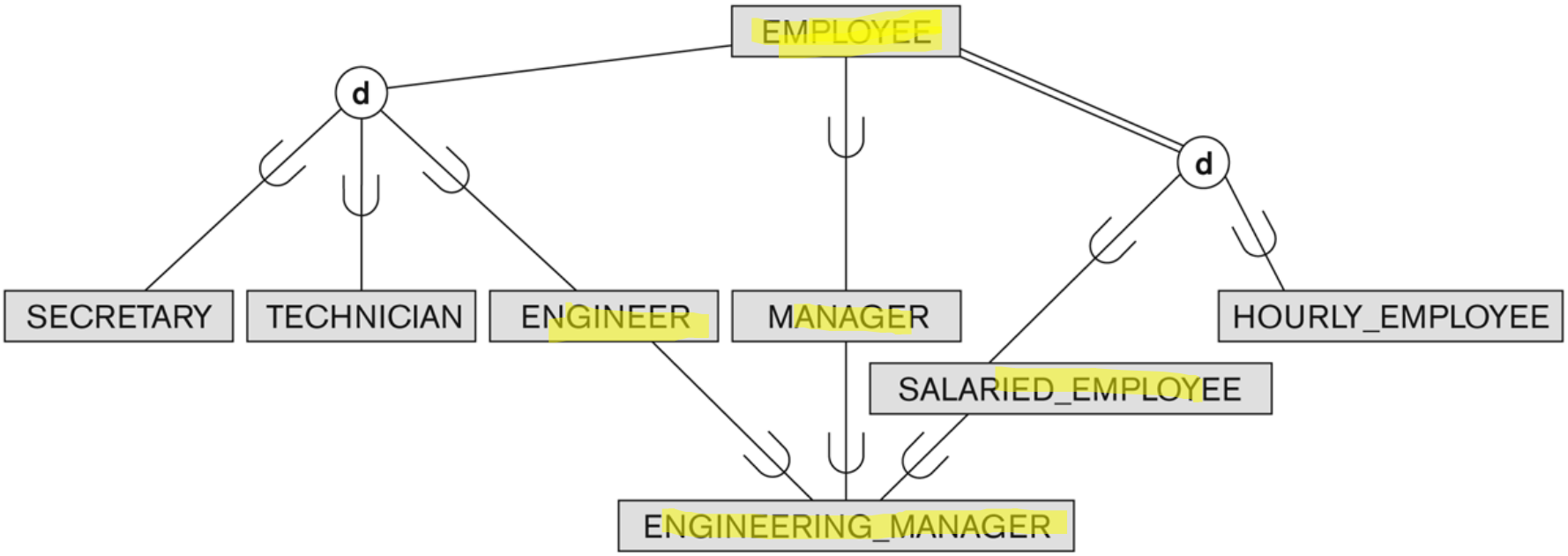


Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

Specialization/Generalization Hierarchies, Lattices & Shared Subclasses (2)

- 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*
- We just use *specialization* (to stand for the end result of either specialization or generalization)

Specialization / Generalization Lattice Example (UNIVERSITY)

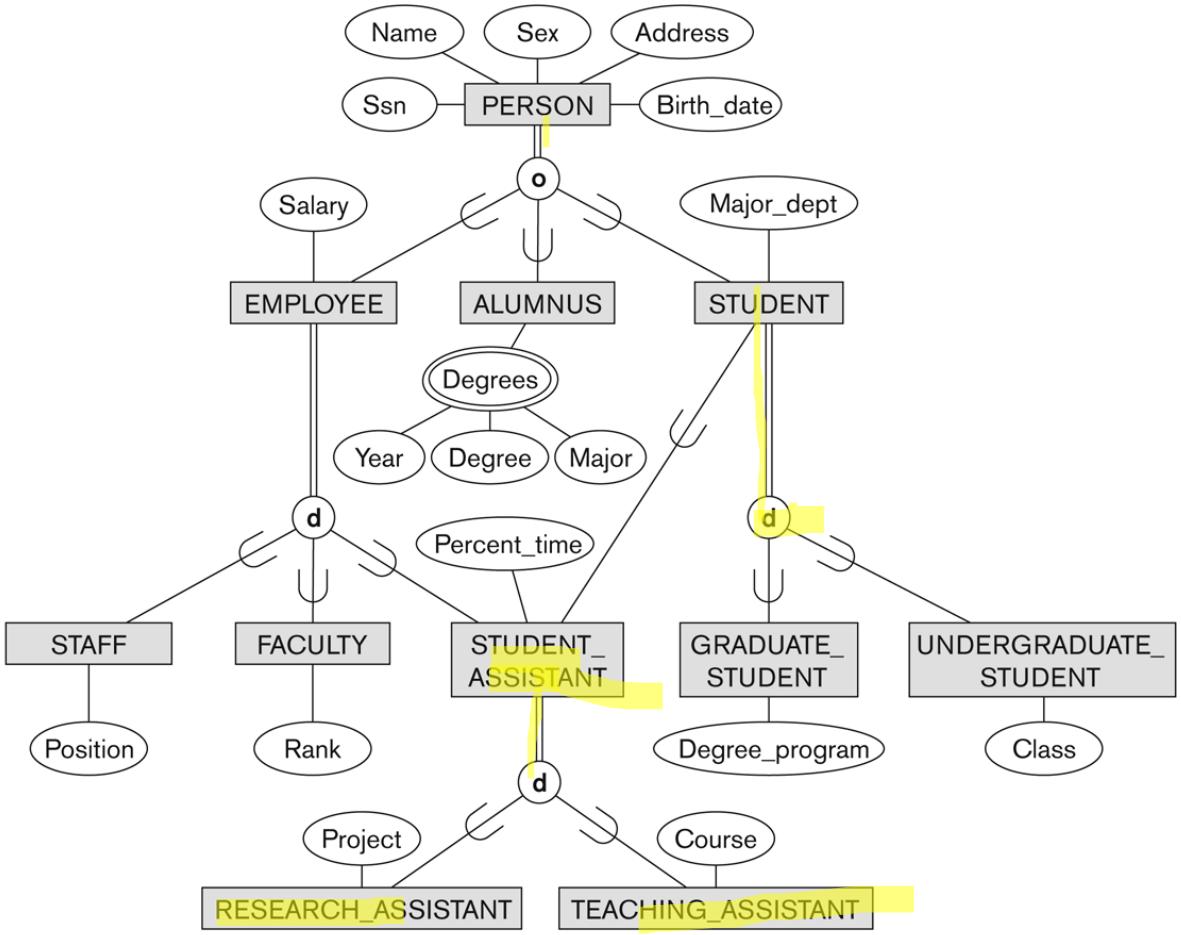


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

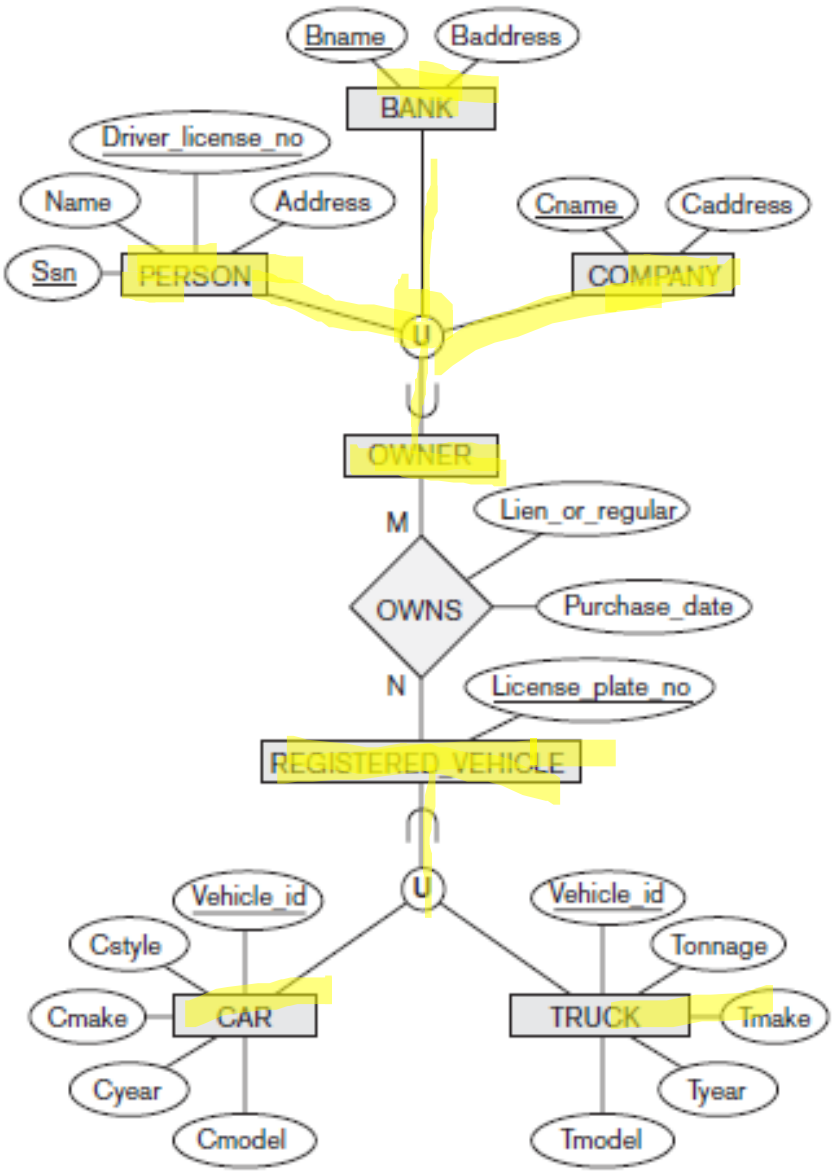
Categories (UNION TYPES) (1)

- 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**

Categories (UNION TYPES) (2)

- 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.
  - 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 ***one only*** of its superclasses
- Difference from *shared subclass*, which is a:
  - shared subclass subset of the *intersection* of its superclasses
  - shared subclass member must exist in ***all*** of its superclasses

Two categories (UNION types): OWNER, R CLE



- 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 and, a shared subclass such as ENGINEERING_MANAGER inherits *all* the attributes of its superclasses SALARIED_EMPLOYEE, ENGINEER, and MANAGER.

- It is interesting to note the difference between the category REGISTERED_VEHICLE and the generalized superclass VEHICLE.
- In the generalized superclass VEHICLE: Every car and every truck is a VEHICLE;
- but in the REGISTERED_VEHICLE category includes some cars and some trucks but not necessarily all of them (for example, some cars or trucks may not be registered).

- In general, a specialization or generalization, if it were *partial*, would not preclude VEHICLE from containing other types of entities, such as motorcycles.
- However, a category such as REGISTERED_VEHICLE implies that only cars and trucks, but not other types of entities, can be members of 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.

- Notice that if a category is total (not partial), it may be represented alternatively as a total specialization (or a total generalization). In this case, 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; otherwise, categorization (union type) is more appropriate

- **ER-to-Relational Mapping Algorithm**
  - Step 1: Mapping of Regular Entity Types
  - Step 2: Mapping of Weak Entity Types
  - Step 3: Mapping of Binary 1:1 Relation Types
  - Step 4: Mapping of Binary 1:N Relationship Types.
  - Step 5: Mapping of Binary M:N Relationship Types.
  - Step 6: Mapping of Multivalued attributes.
  - Step 7: Mapping of N-ary Relationship Types.

- **Mapping EER Model Constructs to Relations**
  - **Step 8: Options for Mapping Specialization or Generalization.**
  - **Step 9: Mapping of Union Types (Categories).**

Mapping EER Model Constructs to Relations

- **Step8: Options for Mapping Specialization or Generalization.**
  - Convert each specialization with m subclasses {S1, S2,….,Sm} and generalized superclass C, where the attributes of C are {k,a1,…an} and k is the (primary) key, into relational schemas using one of the four following options:
    - Option 8A: Multiple relations-Superclass and subclasses
    - Option 8B: Multiple relations-Subclass relations only
    - Option 8C: Single relation with one type attribute
    - Option 8D: Single relation with multiple type attributes

Mapping EER Model Constructs to Relations

- **Option 8A: Multiple relations-Superclass and subclasses**
  - Create a relation L for C with attributes Attrs(L) = {k,a1,…an} and PK(L) = k. **Create a relation Li for each subclass** Si, 1 < i < m, with the attributesAttrs(Li) = {k} U {attributes of Si} and PK(Li)=k. **This option works for any specialization (total or partial, disjoint of over-lapping).**

FIGURE 4.4
EER diagram notation for an attribute-defined specialization on JobType.

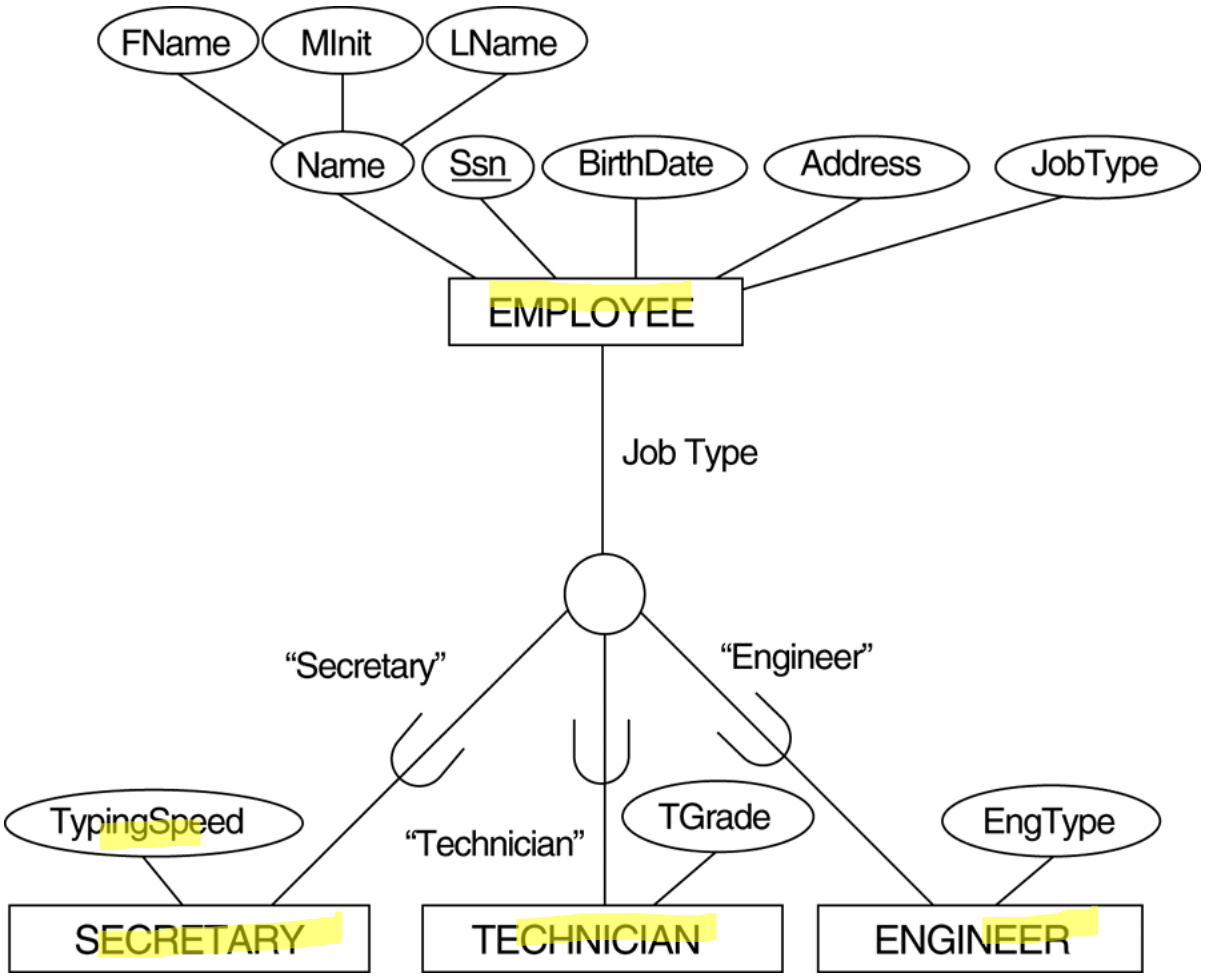
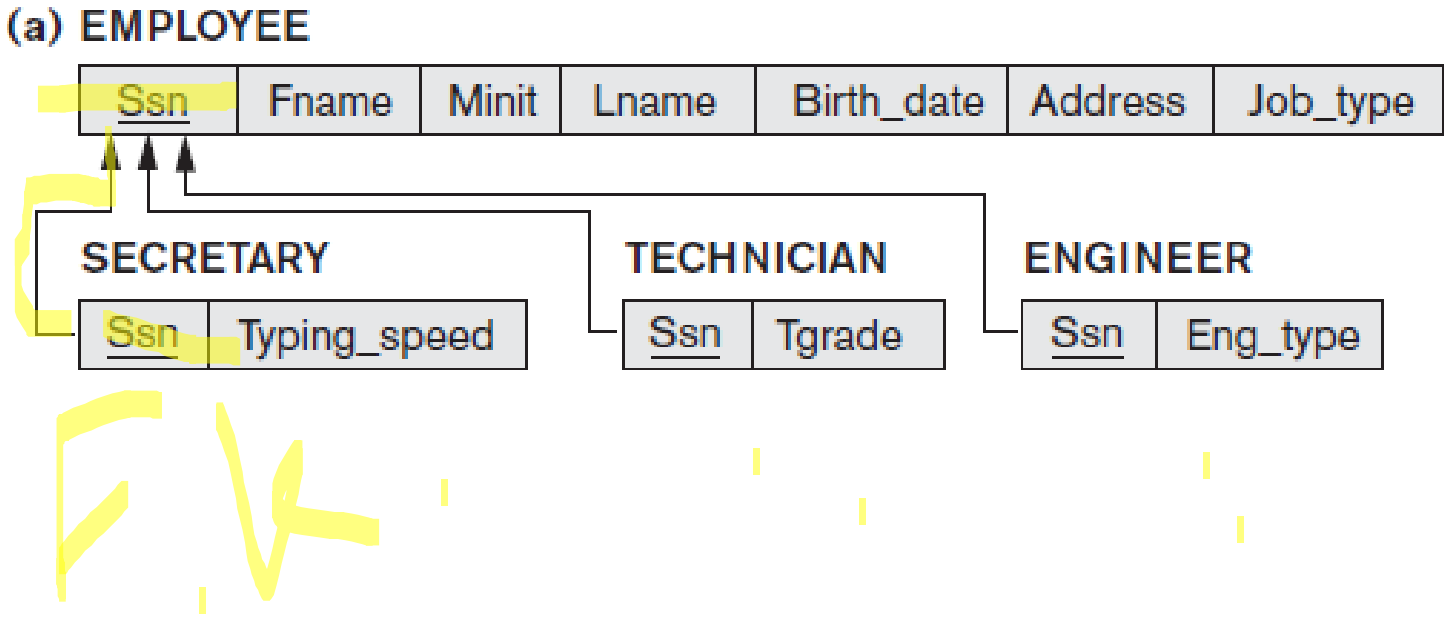


FIGURE 7.4
Options for mapping specialization or generalization.
(a) Mapping the EER schema in Figure 4.4 using option 8A.



Mapping EER Model Constructs to Relations

- **Option 8B: Multiple relations-Subclass relations only**
  - **Create a relation Li for each subclass** Si, 1 < i < m, with the attributes Attr(Li) = {attributes of Si} U {k,a1…,an} and PK(Li) = k. **This option only works for a specialization whose subclasses are total** (every entity in the superclass must belong to (at least) one of the subclasses).

FIGURE 4.3
Generalization. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.

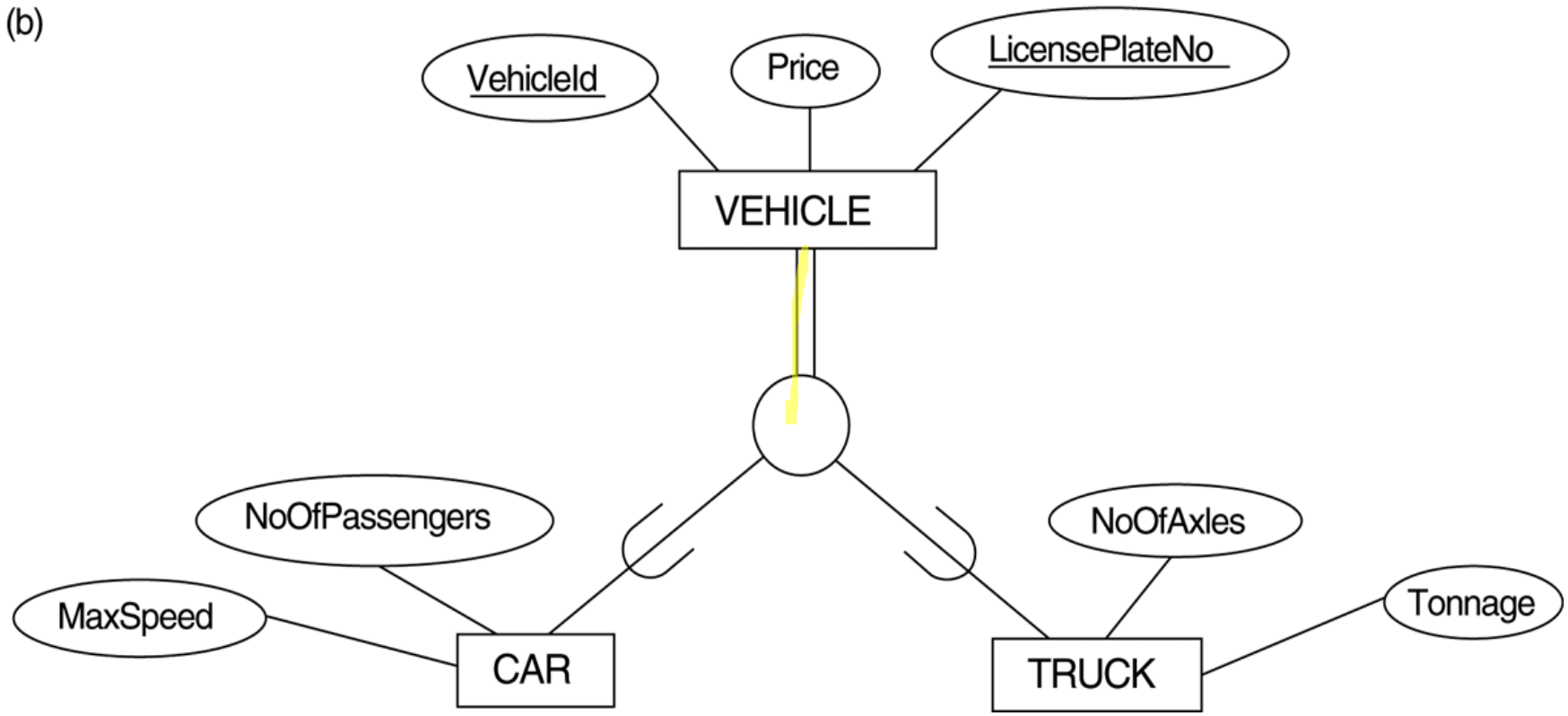
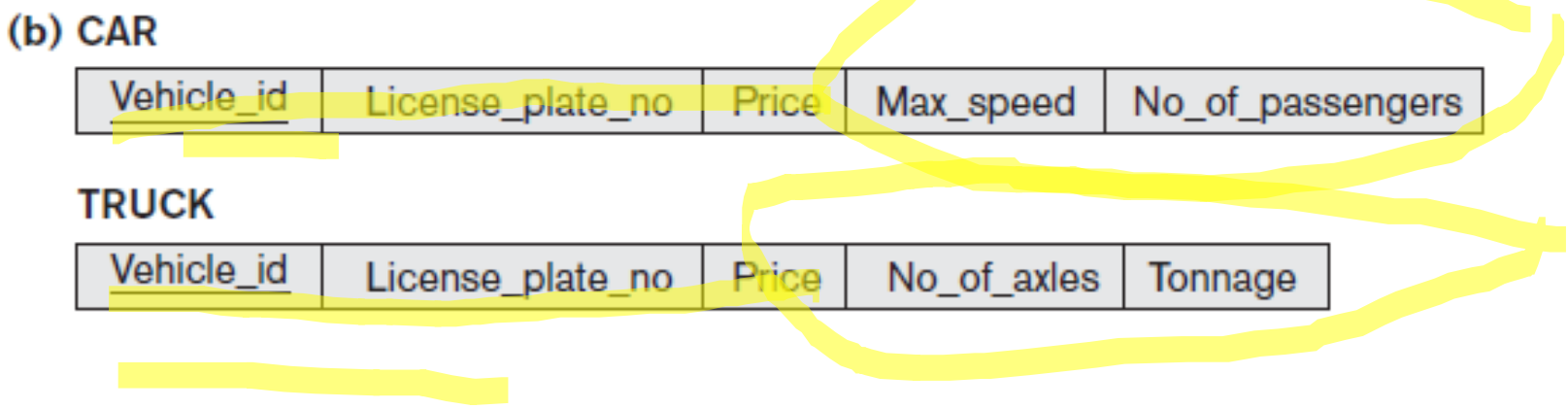


FIGURE 7.4
Options for mapping specialization or generalization.
(b) Mapping the EER schema in Figure 4.3b using option 8B.



Mapping EER Model Constructs to Relations (contd.)

- **Option 8C: Single relation with one type attribute**
  - Create **a single relation** L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t} and PK(L) = k. The attribute t is called a **type** (or **discriminating**) attribute that indicates the subclass to which each tuple belongs
  - **This option works only for a specialization whose subclasses are disjoint**, and has the potential for generating many NULL values if many specific attributes exist in the subclasses.

FIGURE 4.4
EER diagram notation for an attribute-defined specialization on JobType.

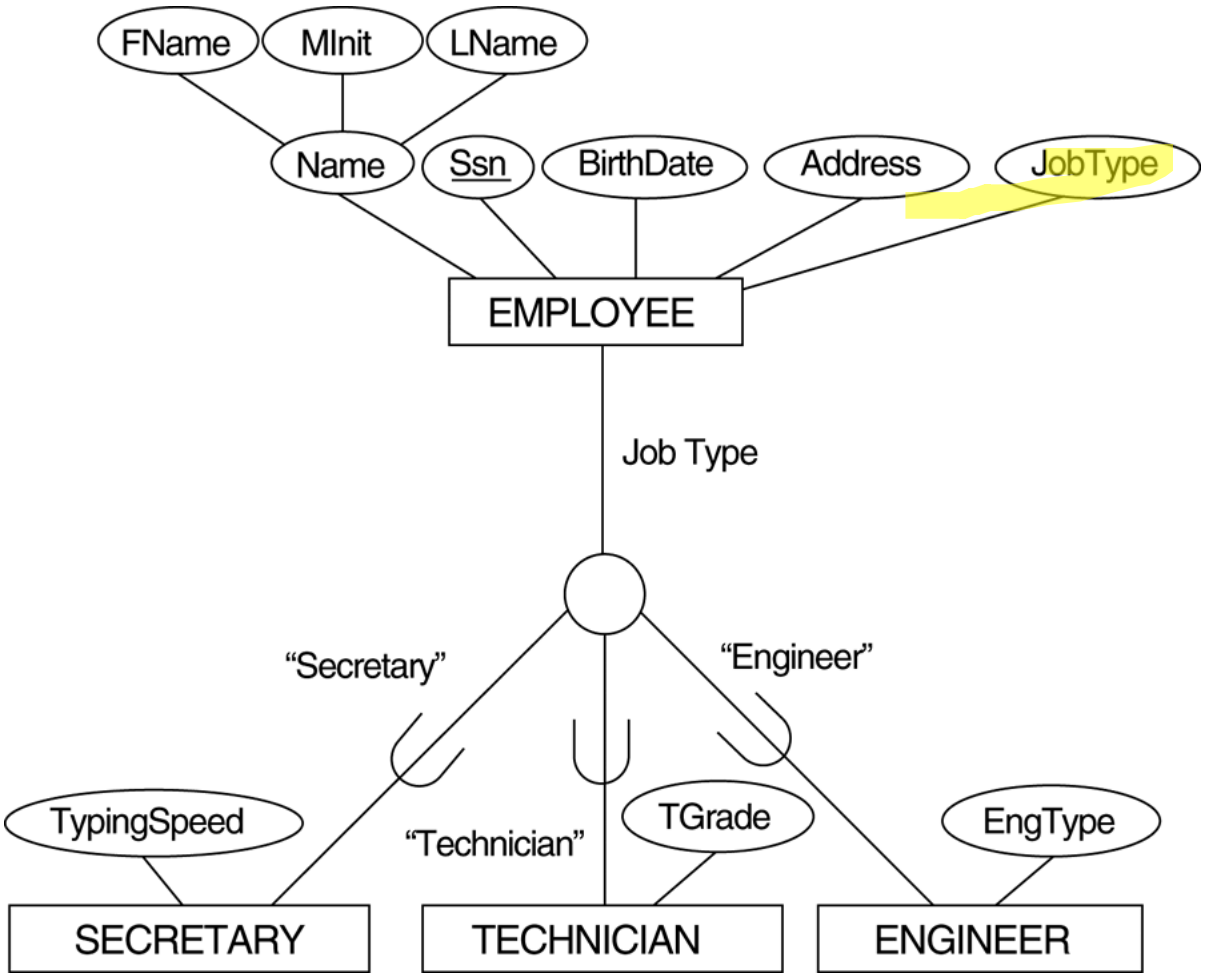


FIGURE 7.4
Options for mapping specialization or generalization.
(c) Mapping the EER schema in Figure 4.4 using option 8C.

(c) EMPLOYEE

Ssn	Fname	Minit	Lname	Birth_date	Address	Job_type	Typing_speed	Tgrade	Eng_type

Mapping EER Model Constructs to Relations (contd.)

- **Option 8D: Single relation with multiple type attributes**
  - Create a **single relation** schema L with attributes Attrs(L) = {k,a_1,…a_n} U {attributes of S_1} U…U {attributes of S_m} U {t_1, t_2,…,t_m} and PK(L) = k. Each t_i, 1 < I < m, is a **Boolean type** attribute indicating whether a tuple belongs to the subclass S_i.
  - **This option is used for a specialization whose subclasses are overlapping** (but will also work for a disjoint specialization).

FIGURE 4.5
EER diagram notation for an overlapping (non-disjoint) specialization.

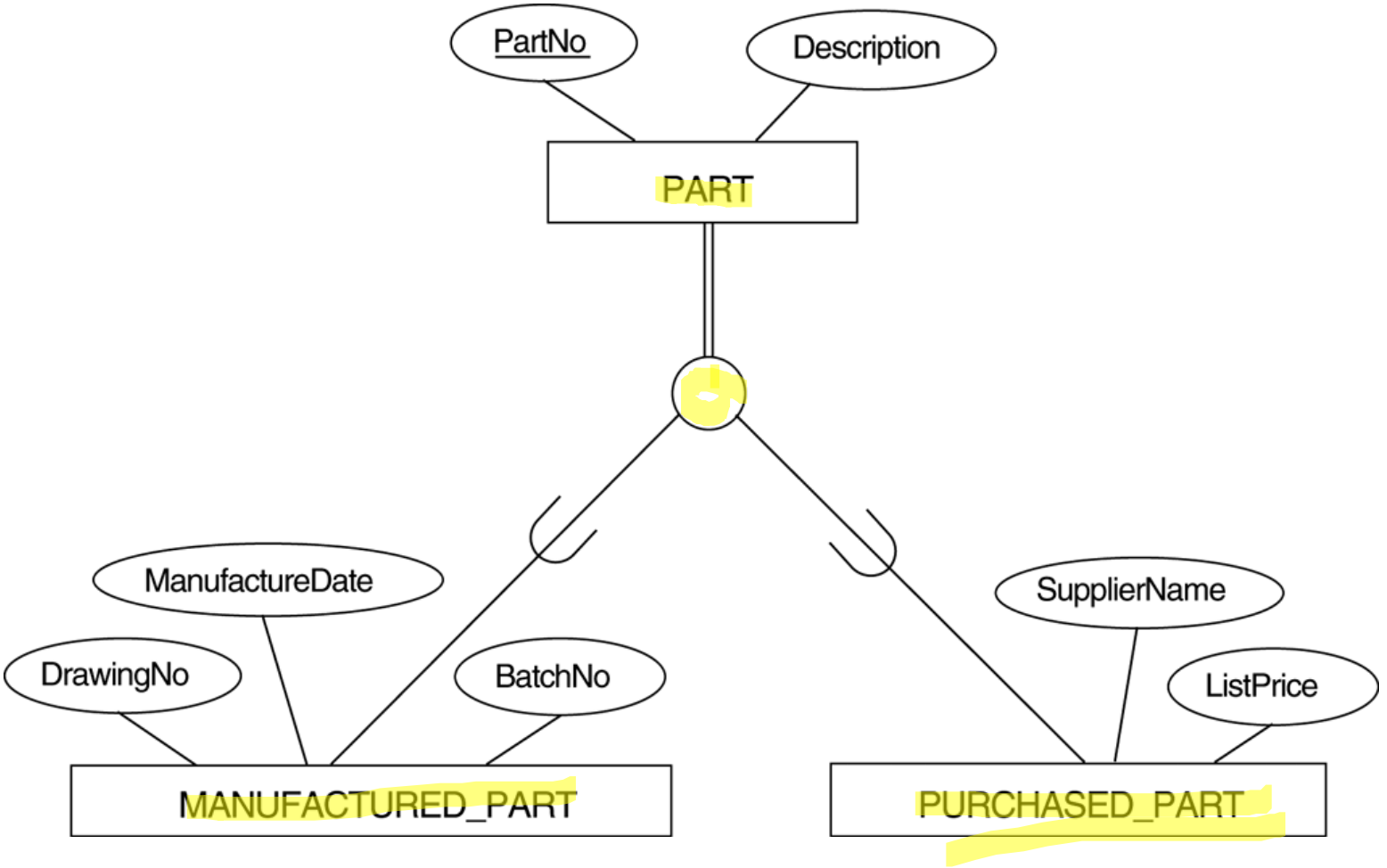
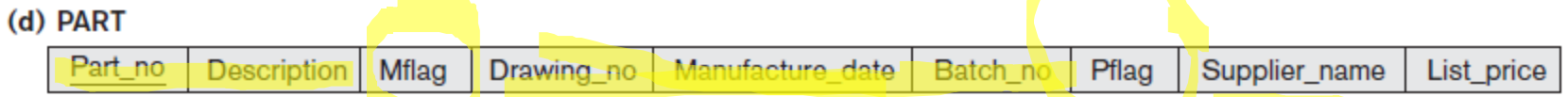


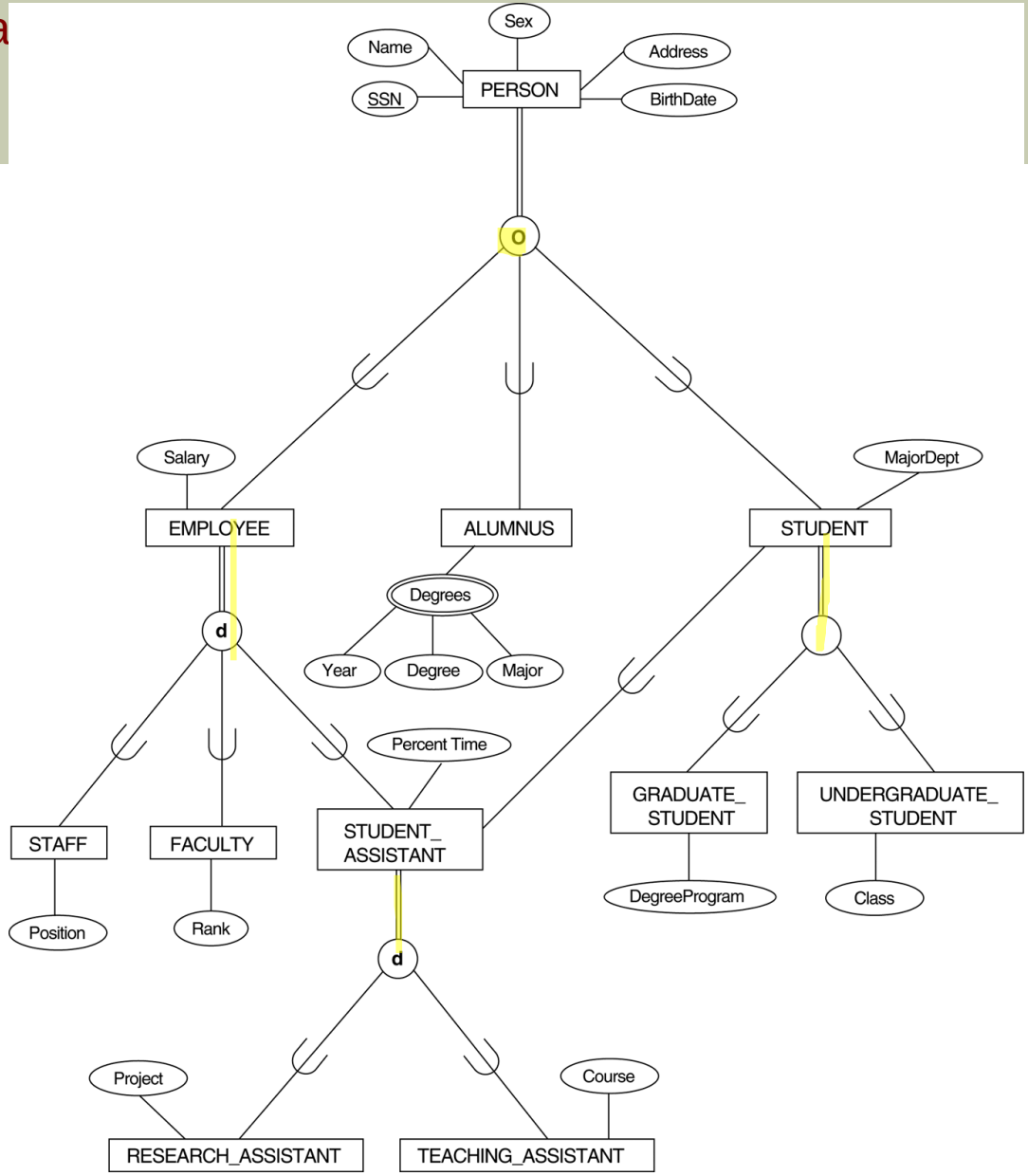
FIGURE 7.4
Options for mapping specialization or generalization. (d) Mapping Figure 4.5 using option 8D with Boolean type fields Mflag and Pflag.



Mapping EER Model Constructs to Relations (contd.)

- When we have a multilevel specialization (or generalization) hierarchy or lattice, **we do not have to follow the same mapping option for all the specializations**. Instead, we can use one mapping option for part of the hierarchy or lattice and other options for other part
- Mapping of Shared Subclasses (Multiple Inheritance)
  - A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance. **These classes must all have the same key attribute**; **otherwise, the shared subclass would be modeled as a category.**

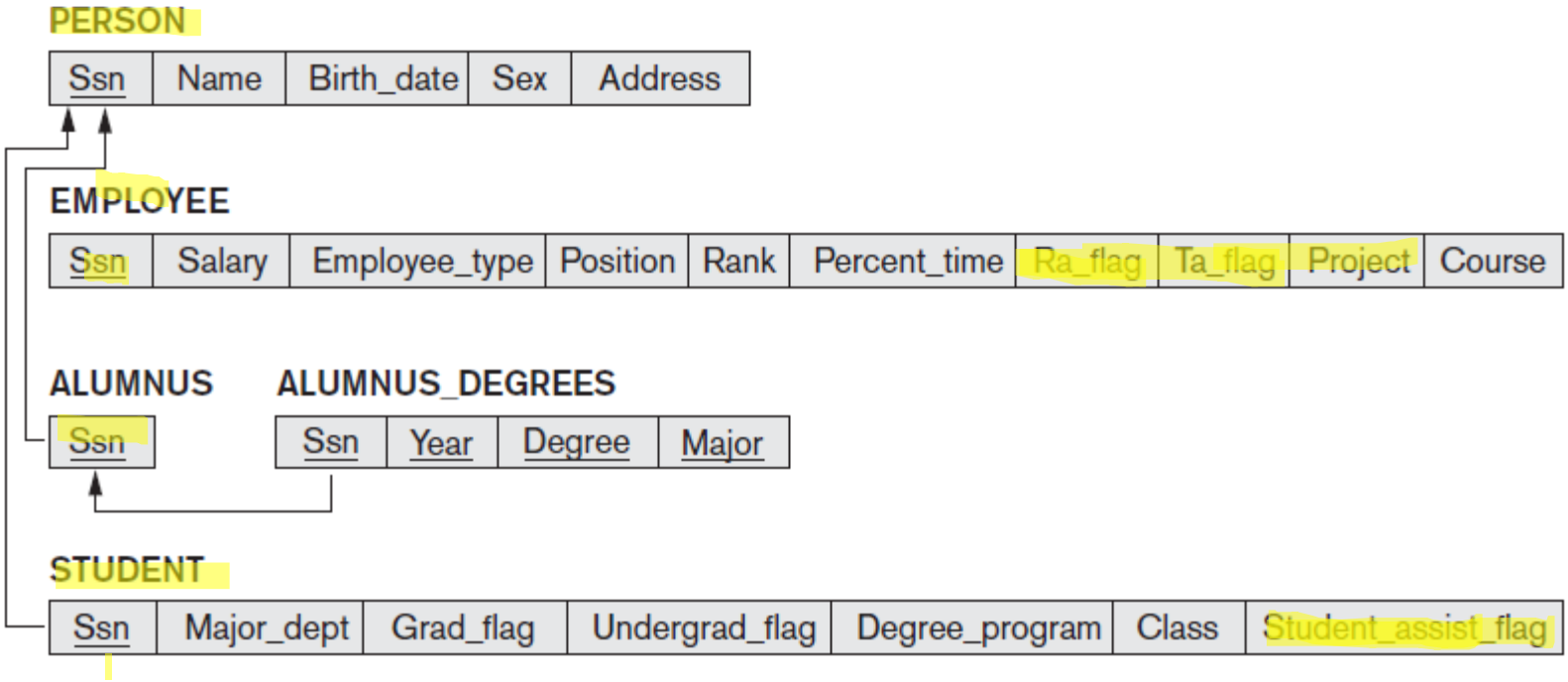
FIGURE 4.7
A specialization la
database.



- Here we used option 8A for **PERSON**/{EMPLOYEE, ALUMNUS, STUDENT},
- option 8C for **EMPLOYEE**/{STAFF, FACULTY, STUDENT_ASSISTANT} by including the type attribute Employee_type,
- and option 8D for **STUDENT_ASSISTANT**/{RESEARCH_ASSISTANT, TEACHING_ ASSISTANT} by including the type attributes Ta_flag and Ra_flag in EMPLOYEE, STUDENT/ STUDENT_ASSISTANT by including the type attributes Student_assist_flag in STUDENT, and STUDENT/{GRADUATE_STUDENT, UNDERGRADUATE_STUDENT} by including the type attributes Grad_flag and Undergrad_flag in STUDENT

FIGURE 7.5
Mapping the EER specialization lattice in Figure 4.6 using multiple options.

- options 8C and 8D are used for the shared subclass STUDENT_ASSISTANT. Option 8C is used in the EMPLOYEE relation (Employee_type attribute) and option 8D is used in the STUDENT relation



Mapping EER Model Constructs to Relations (contd.)

- **Step 9: Mapping of Union Types (Categories).**
  - For mapping a category whose defining superclass have **different keys**, it is customary to specify **a new key attribute**, called a **surrogate key**, when creating a relation to correspond to the category.
  - In the example below we can create a relation OWNER to correspond to the OWNER category and include any attributes of the category in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId.

- if a particular PERSON (or BANK or COMPANY) entity is not a member of OWNER, it would have a NULL value for its Owner_id attribute in its corresponding tuple in the PERSON (or BANK or COMPANY) relation, and it would not have a tuple in the OWNER relation.

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

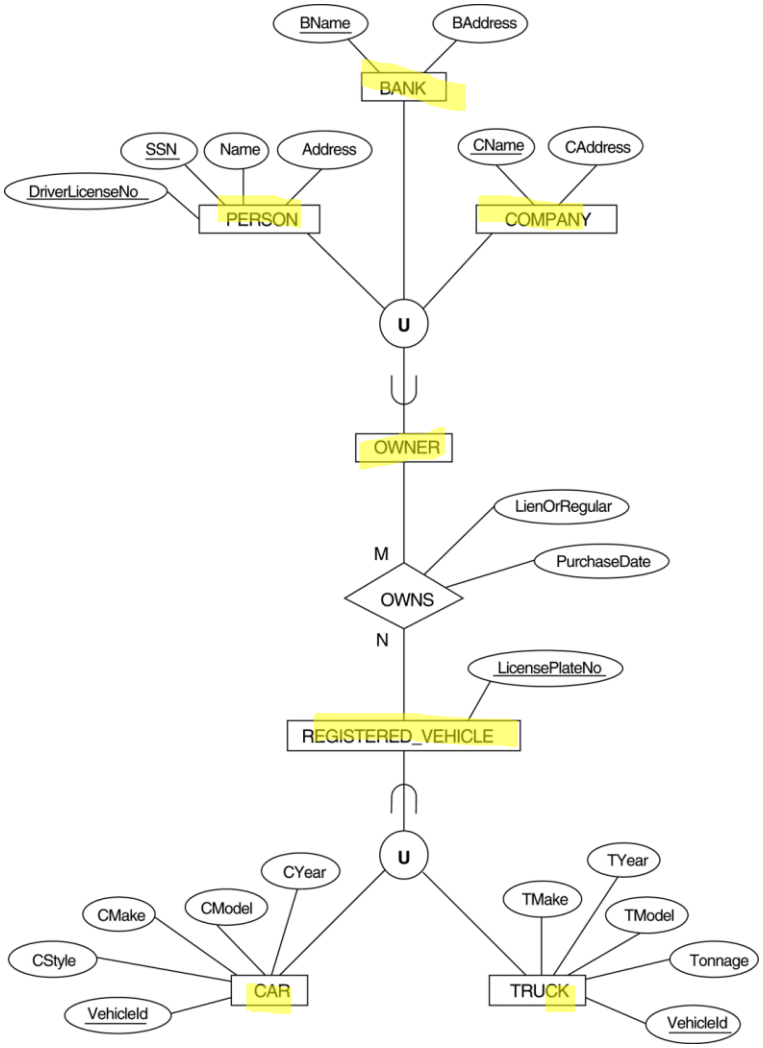
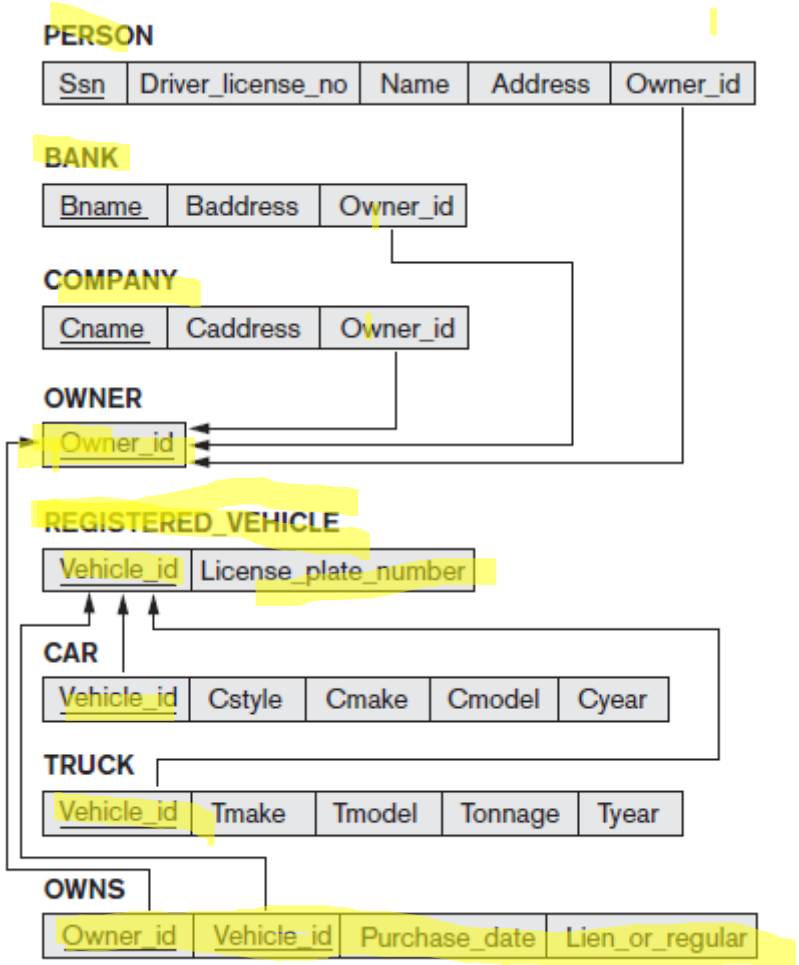


FIGURE 7.6
Mapping the EER categories (union types) in Figure 4.7 to relations.



Summary

- Introduced the EER model concepts
  - Class/subclass relationships
  - Specialization and generalization
  - Inheritance
- These augment the basic ER model concepts introduced in Chapter 3
- EER diagrams and alternative notations were presented