

# 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
  - ▶ Constraints on Specialization/Generalization

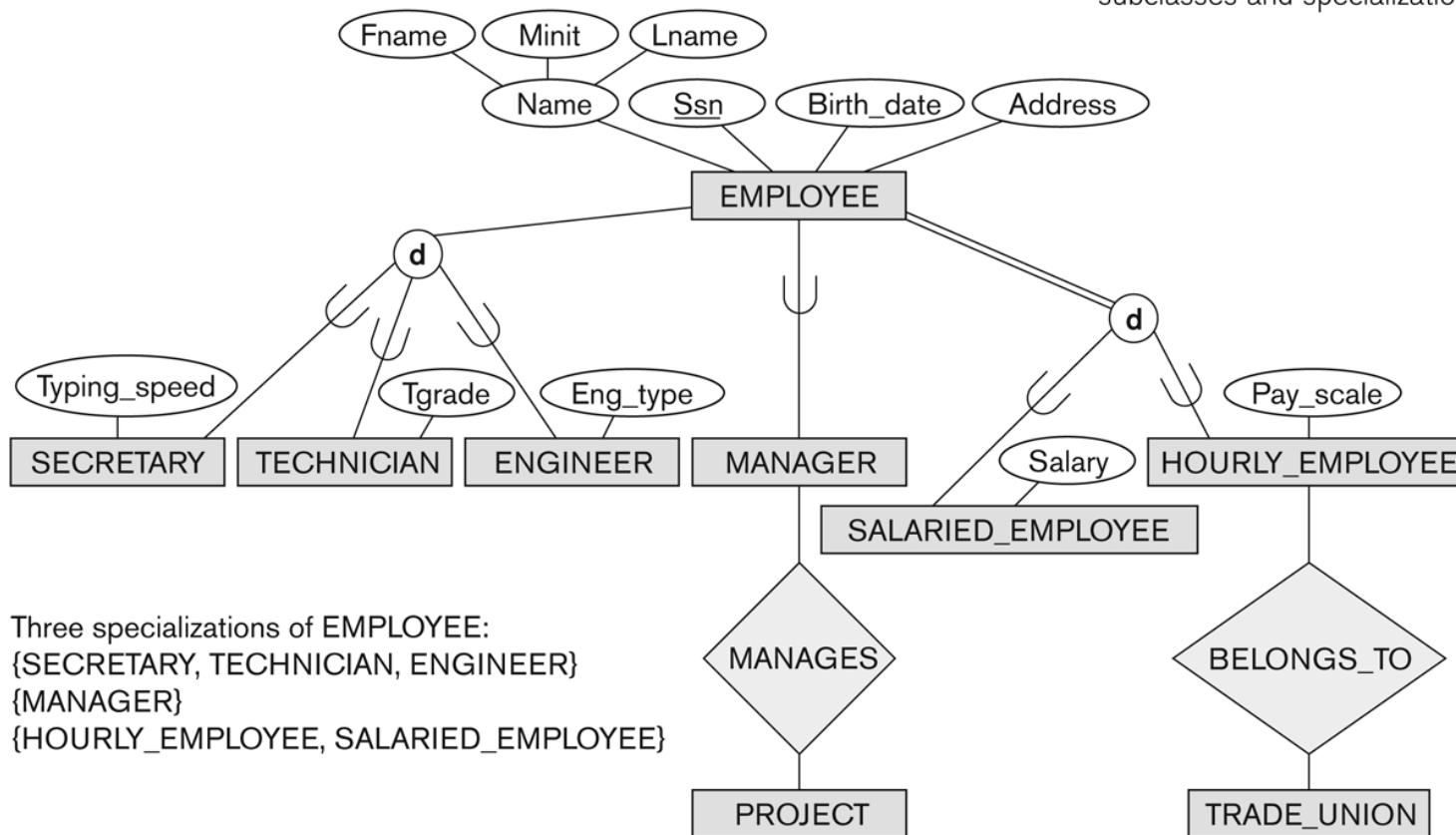
# Subclasses and Superclasses (1)

- ▶ 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 (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

**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 (3)

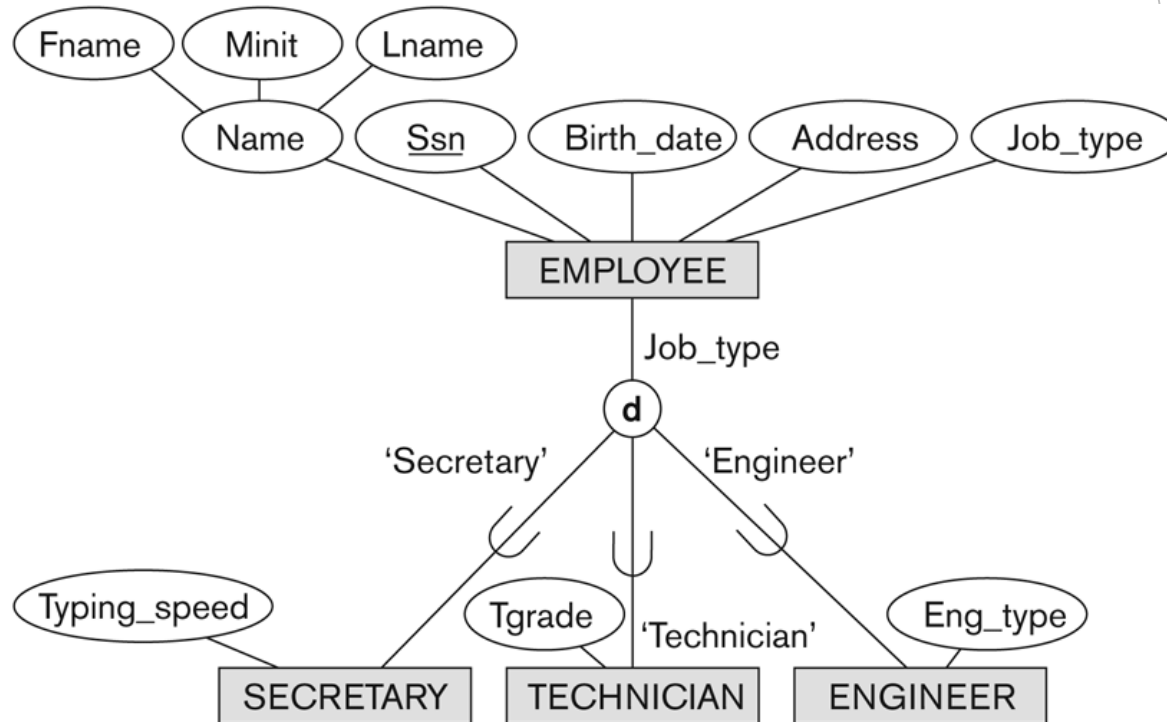
- ▶ These are also called IS-A relationships
  - ▶ SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ....
- ▶ Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
  - ▶ The subclass member is the same entity in a *distinct specific role*
  - ▶ 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*.
  - ▶ Example: *MANAGER is a specialization of EMPLOYEE based on the role the employee plays*
    - ▶ May have several specializations of the same superclass

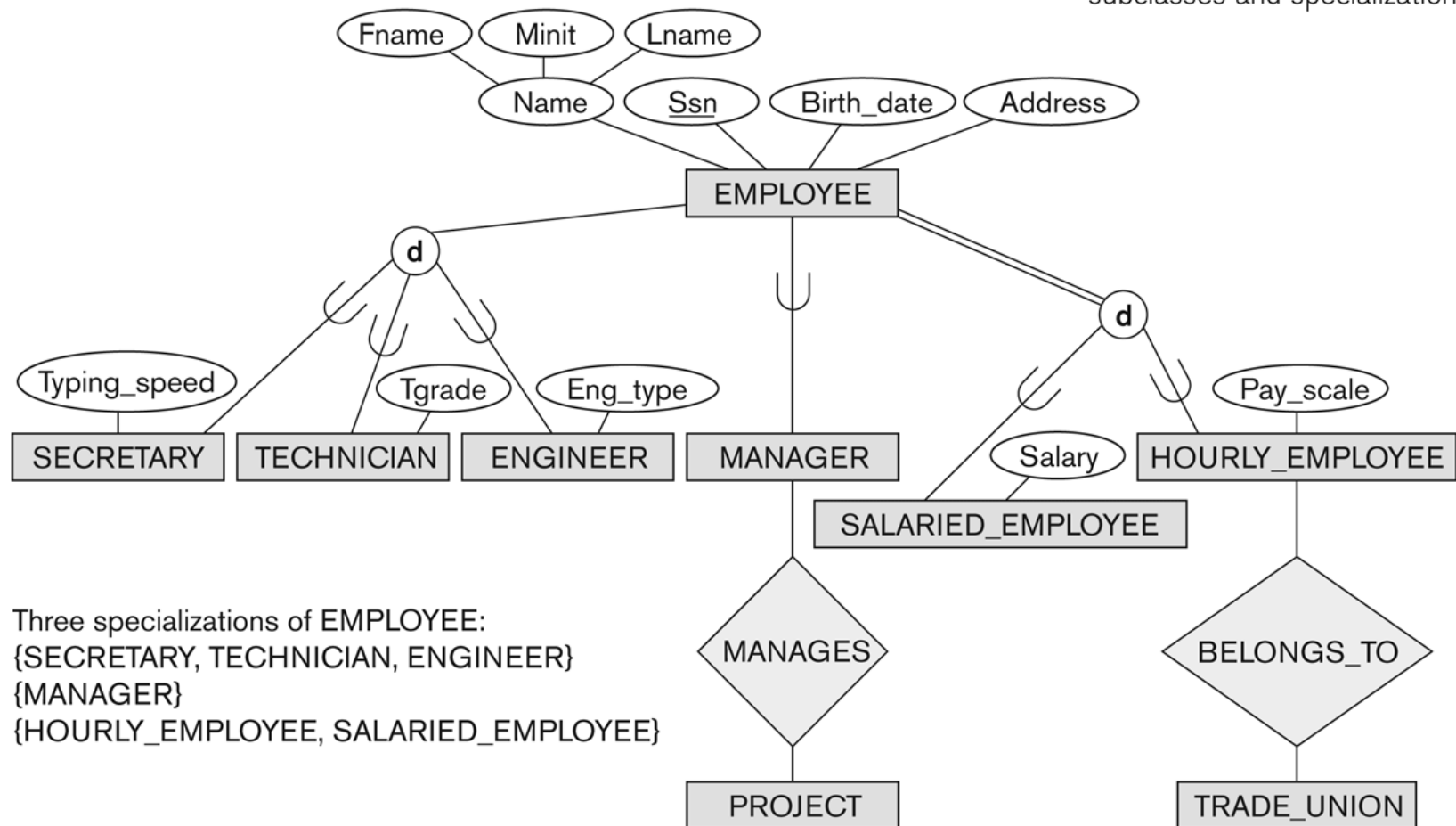
# Specialization (2)

- ▶ 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**

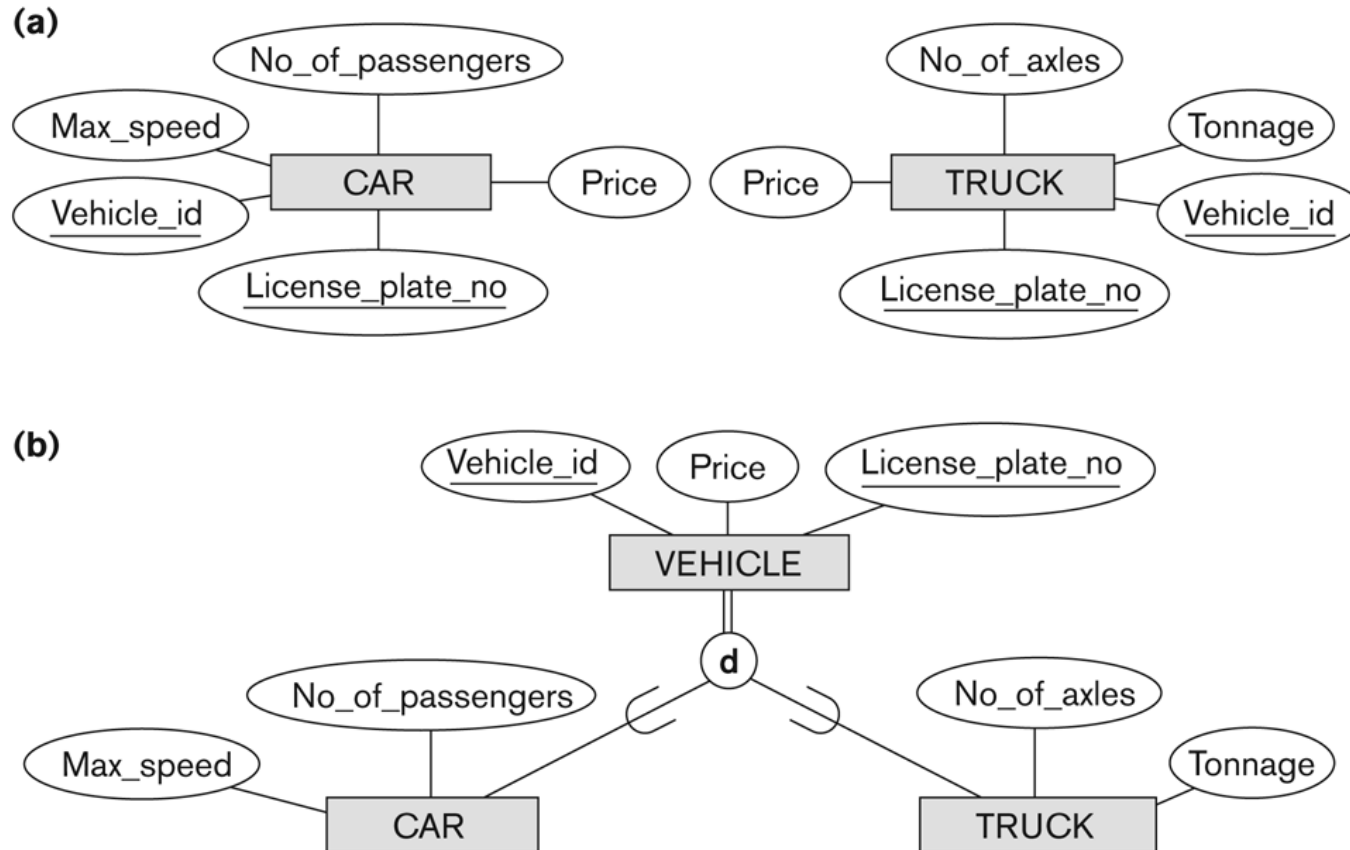
EER diagram notation to represent subclasses and specialization.



# Generalization (1)

- ▶ 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 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
  - ▶ 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

# Generalization and Specialization (2)

- ▶ Data Modeling with Specialization and Generalization
  - ▶ 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



# Types of Specialization

- ▶ Predicate-defined ( or condition-defined) : based on some predicate. E.g., based on value of an attribute, say, Job-type, or Age.
- ▶ Attribute-defined: shows the name of the attribute next to the line drawn from the superclass toward the subclasses (see Fig. 4.1)
- ▶ User-defined: membership is defined by the user on an entity by entity basis

# 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

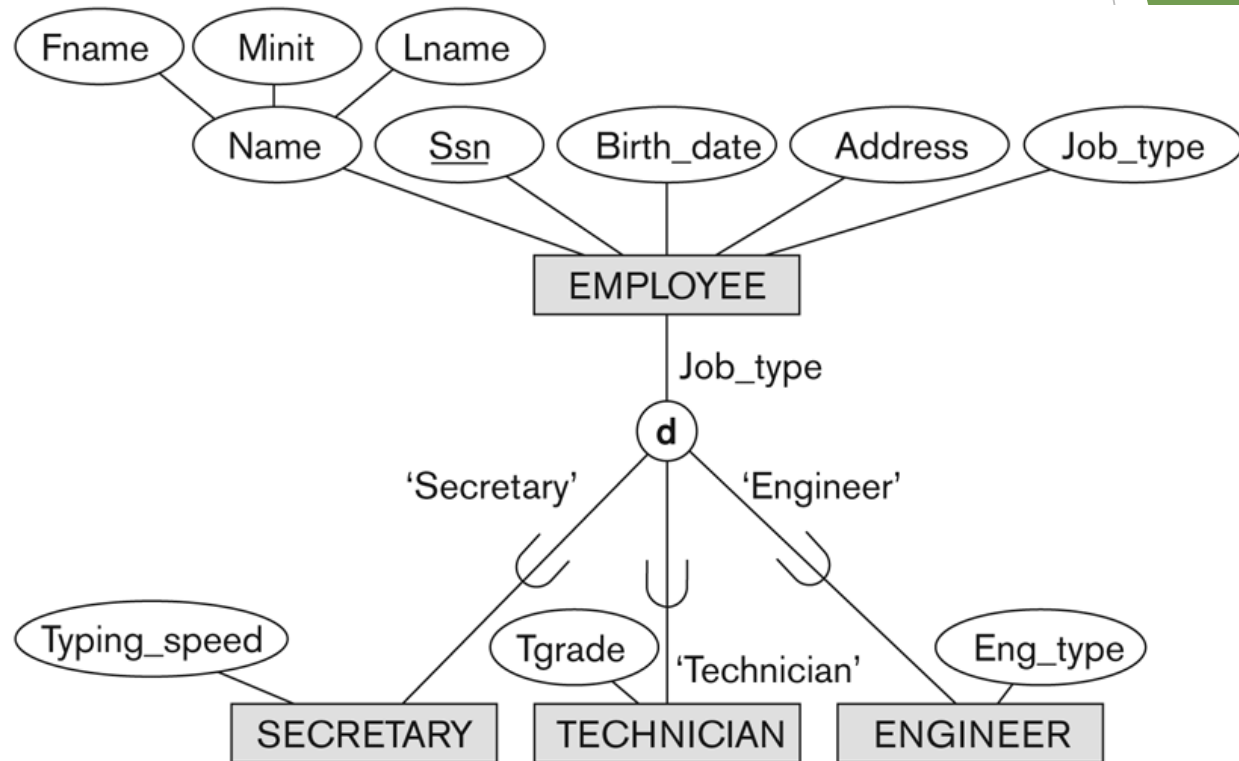
# Constraints on Specialization and Generalization (2)

- ▶ 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

# Constraints on Specialization and Generalization (5)

- ▶ 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*
  - ▶ *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 (6)

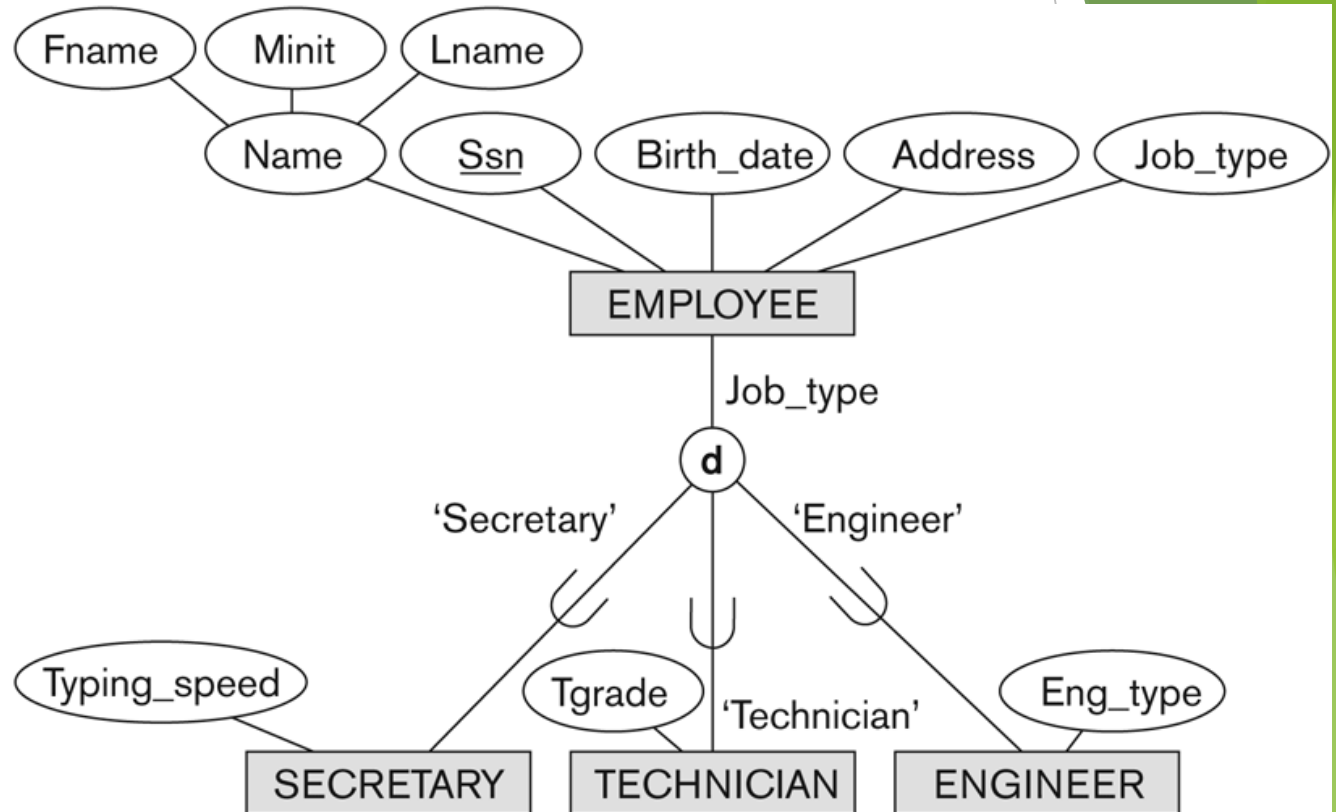
- ▶ Hence, we have four types of specialization/generalization:
  - ▶ Disjoint, total
  - ▶ Disjoint, partial
  - ▶ Overlapping, total
  - ▶ Overlapping, partial
- ▶ Note: Generalization usually is total because the superclass is derived from the subclasses.



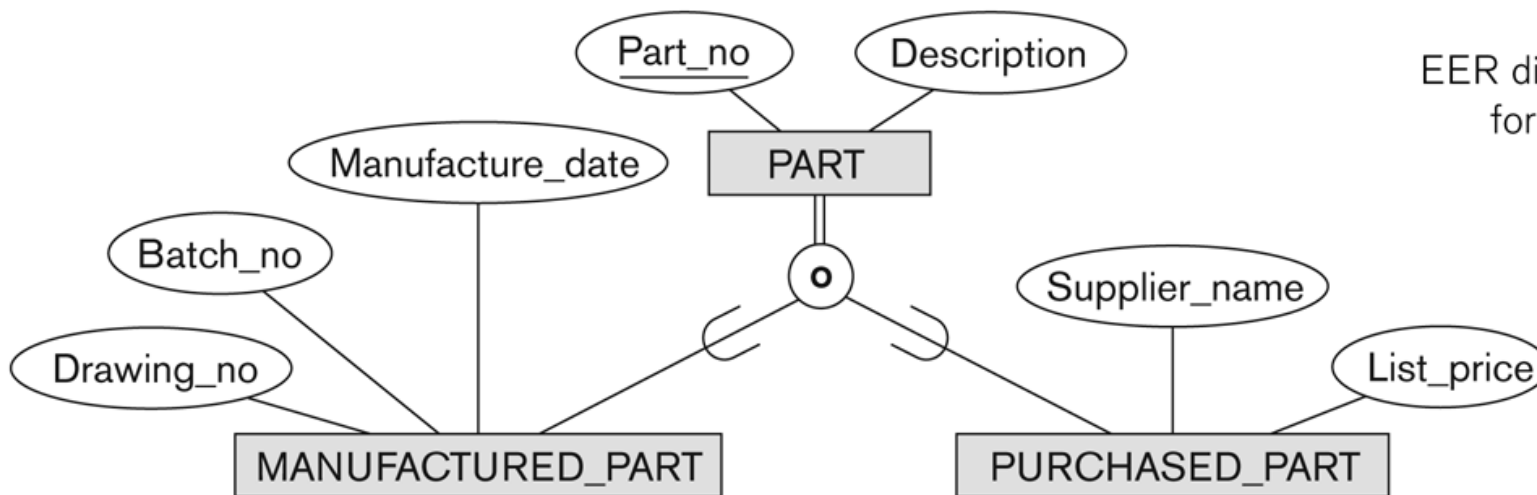
# Example of disjoint partial Specialization

**Figure 4.4**

EER diagram notation for an attribute-defined specialization on Job\_type.



# Example of overlapping total Specialization



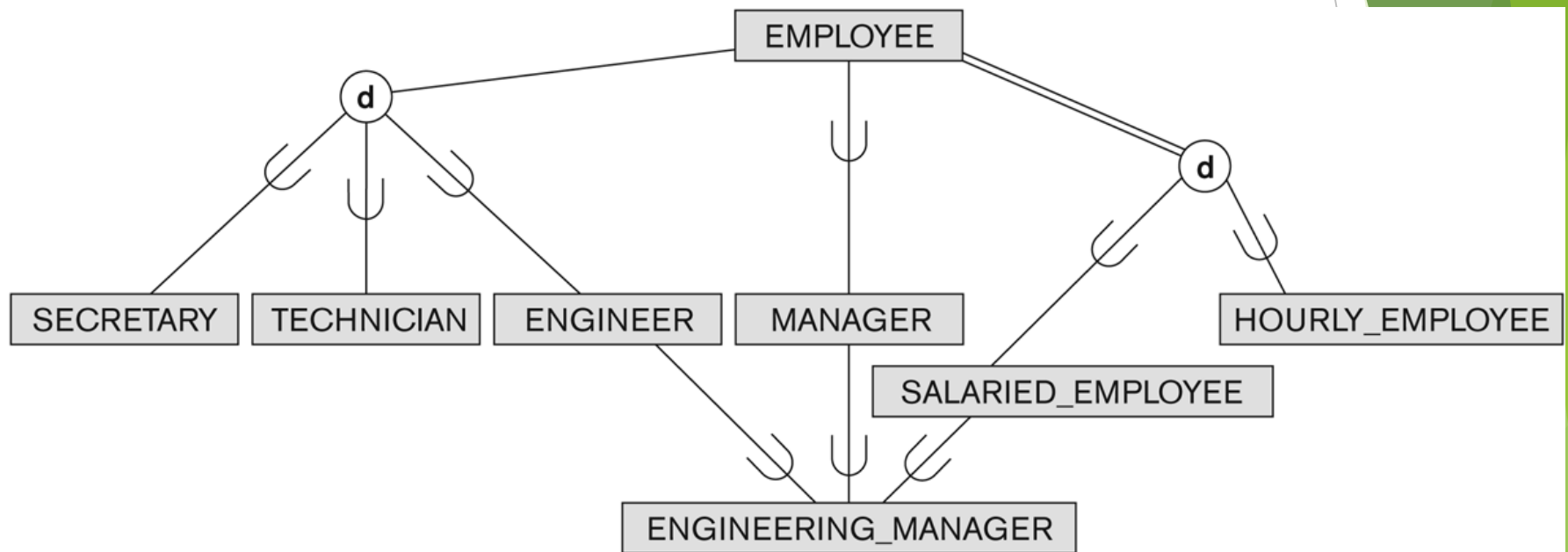
**Figure 4.5**

EER diagram notation for an overlapping (nondisjoint) specialization.

# 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.

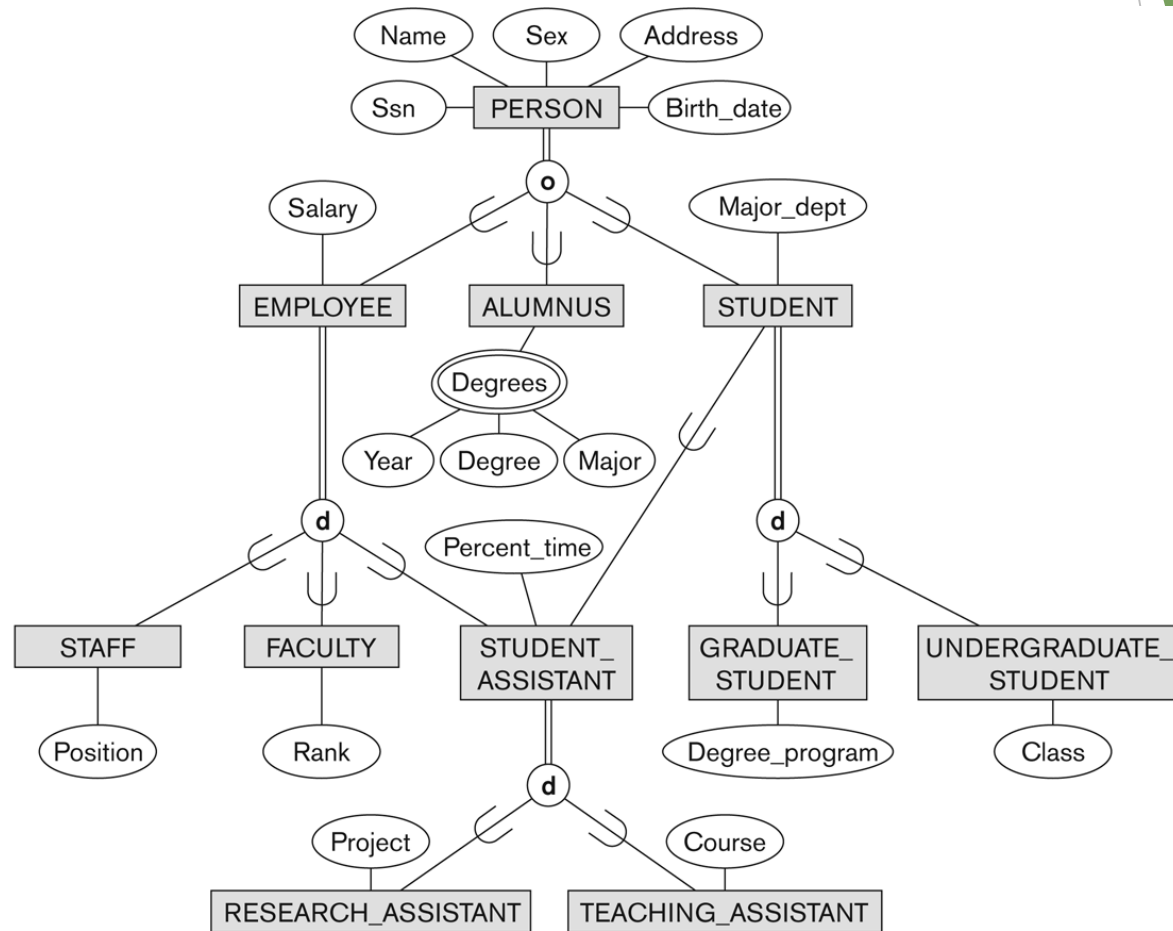
# Specialization/Generalization Hierarchies, Lattices & Shared Subclasses (2)

- ▶ 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 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

# Specialization / Generalization Lattice Example (UNIVERSITY)



**Figure 4.7**

A specialization lattice with multiple inheritance for a UNIVERSITY database.

# Categories (UNION TYPES) (1)

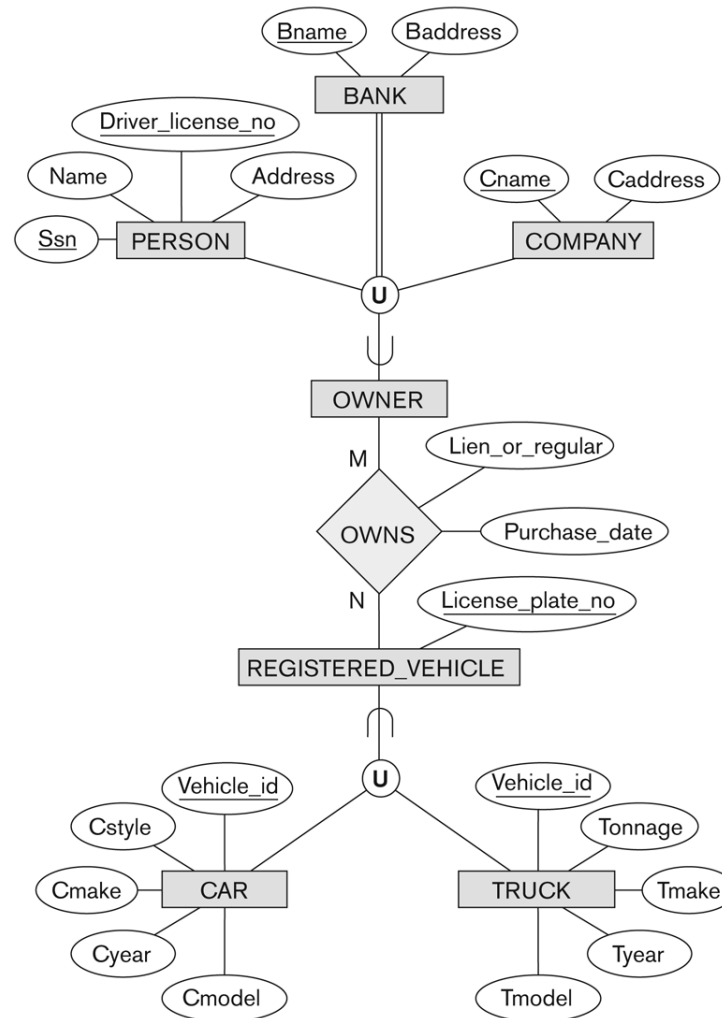
- ▶ All of the *superclass/subclass relationships* we have seen thus far have a single superclass
- ▶ A shared subclass is a subclass in:
  - ▶ *more than one* distinct superclass/subclass relationships
  - ▶ each relationships has a *single* superclass
  - ▶ shared subclass leads to multiple inheritance
- ▶ 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 ***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

# Two categories (UNION types): OWNER, REGISTERED\_VEHICLE



**Figure 4.8**  
Two categories (union types): OWNER and REGISTERED\_VEHICLE.

# Summary

- ▶ Introduced the EER model concepts
  - ▶ Class/subclass relationships
  - ▶ Specialization and generalization
  - ▶ Inheritance
- ▶ Constraints on EER schemas