COURSE NAME: DBMS
COURSE CODE:23AD2102A

Topic: **Entity-Relationship (ER) Modelling Extends**

Session - 5

AIM OF THE SESSION

To familiarize students with the basic concept of Designing ER diagrams, ER data Model, EER diagram in Entity Relationship (ER) Diagrams.

INSTRUCTIONAL OBJECTIVES

This Session is designed to:

- Understand the concepts of Designing ER diagrams, ER data Model used for Data Modelling.
- Compare and evalu chniques.

LEARNING OUTCOMES

At the end of this session, you should be able to:

- Identify the ER data Model and Designing ER diagrams.
- ER Data Modeling Diagram from Users Problem Description.

INTRODUCTION TO EXTENDED ER

- Extended Entity Relationship (EER) modeling is an extension of the original Entity Relationship (ER) model used in database design.

- The EER model includes additional concepts such as subclasses, superclasses, and inheritance relationships, which allow for more complex and flexible data modeling.

- In an EER model, entities can be grouped into classes or subclasses based on common attributes or behaviors. Each class can have its own attributes and relationships, and inherit attributes and relationships from its superclass.

- This allows for more efficient data modeling and better data organization, as well as greater flexibility in handling complex relationships between entities.

INTRODUCTION TO EXTENDED ER

- Other advanced concepts in EER modeling include multivalued attributes, derived attributes, and constraints such as participation constraints and disjoint constraints. These features help to ensure data integrity and consistency, and allow for more precise control over data relationships.

- In a PowerPoint presentation on EER modeling, it is important to provide a clear overview of the benefits and features of EER modeling, as well as examples of how it can be used in real-world database design scenarios.

- You may want to include diagrams and visual aids to help explain the concepts and illustrate how entities, classes, and relationships are organized in an EER model.

INTRODUCTION TO EXTENDED ER

- It is also important to highlight the potential challenges and limitations of EER modeling, such as the complexity of modeling and the need for specialized software tools.

- By providing a balanced and informative presentation on EER modeling, you can help your audience better understand the benefits and challenges of this advanced approach to database design.

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 (2)

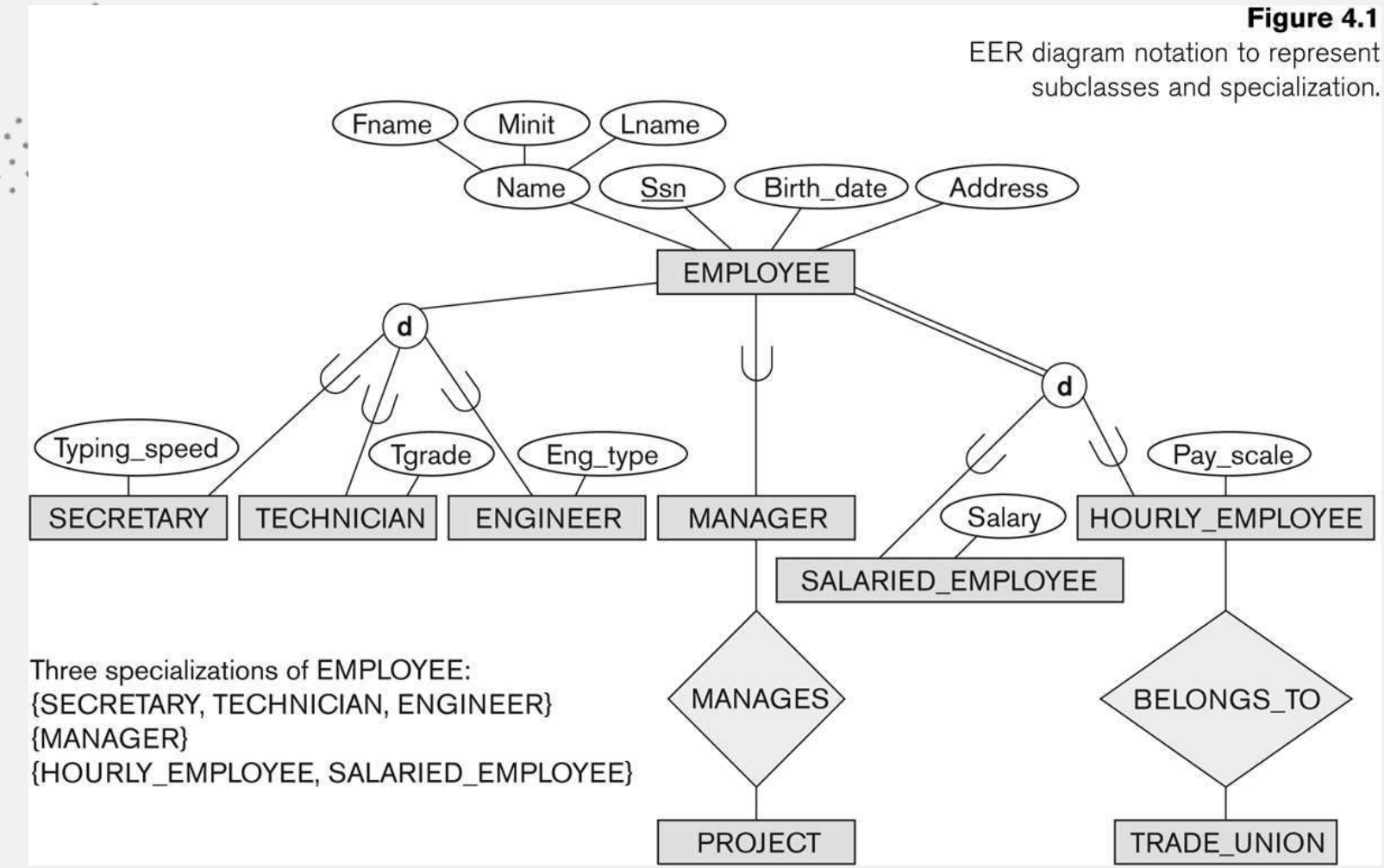


Figure 4.1
EER diagram notation to represent subclasses and specialization.

Subclasses and Superclasses (3)

- 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 (4)

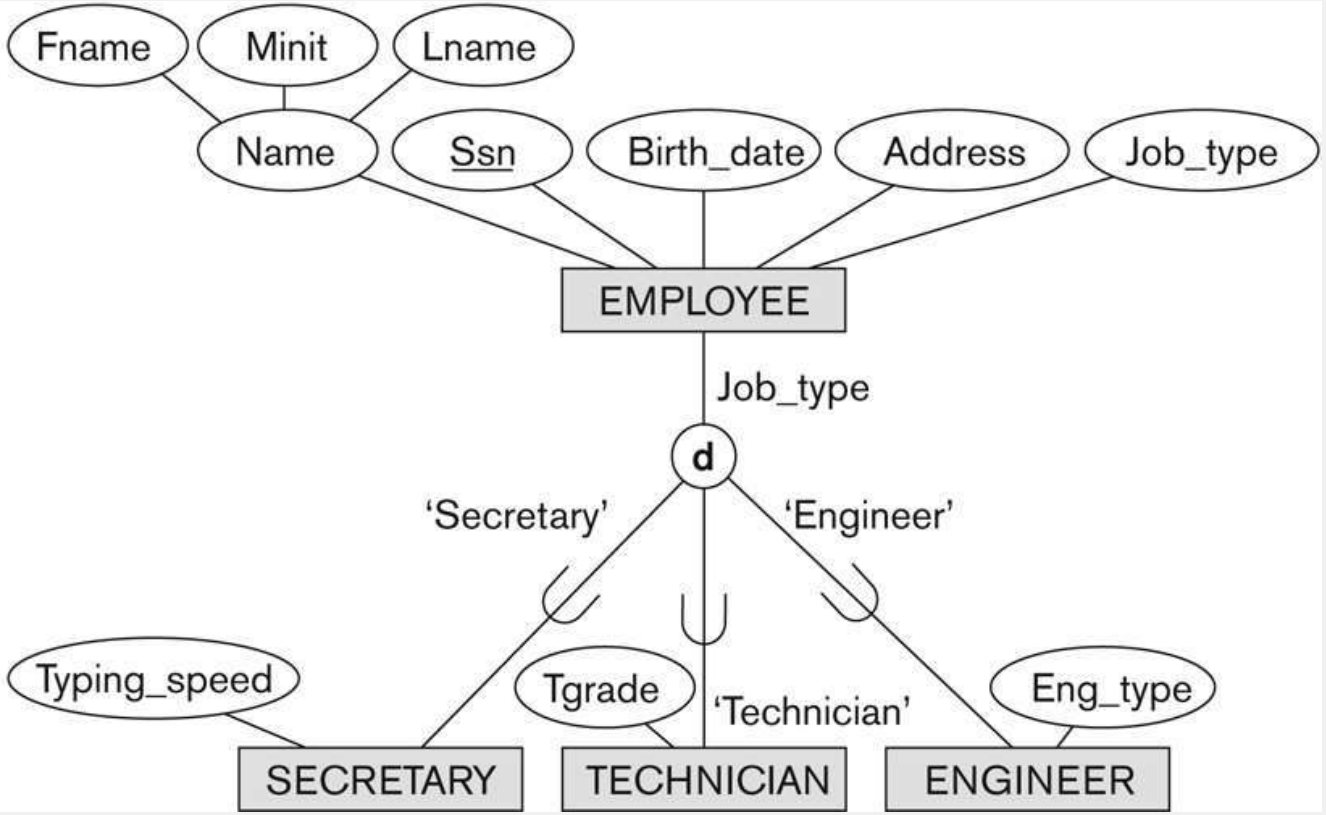
- These are also called IS-A relationships
  - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ….
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
  - The subclass member is the same entity in a *distinct specific role*
  - An entity 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 (5)

- 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)

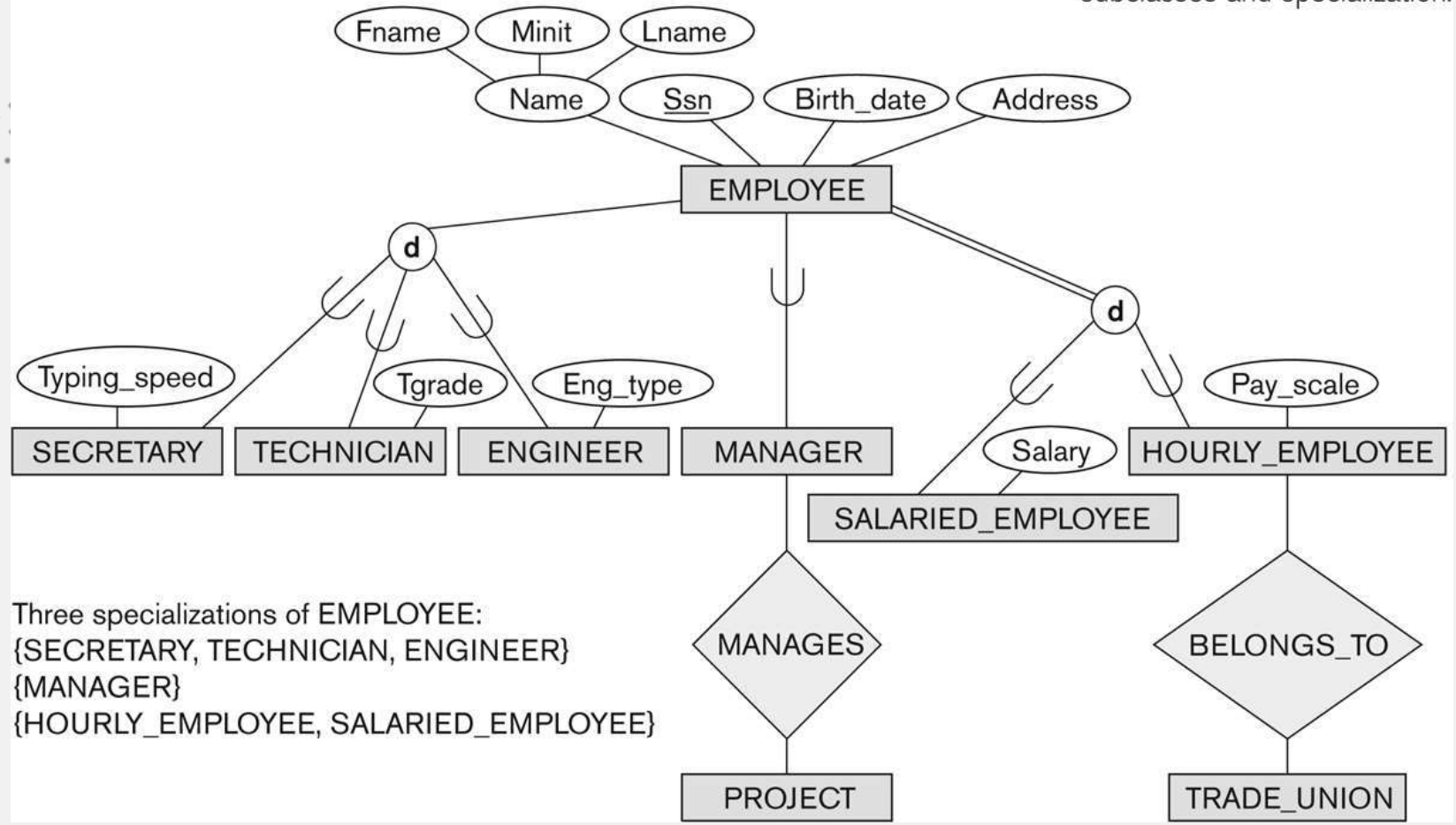
- Specialization is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
  - Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type.*
    - May have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



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)

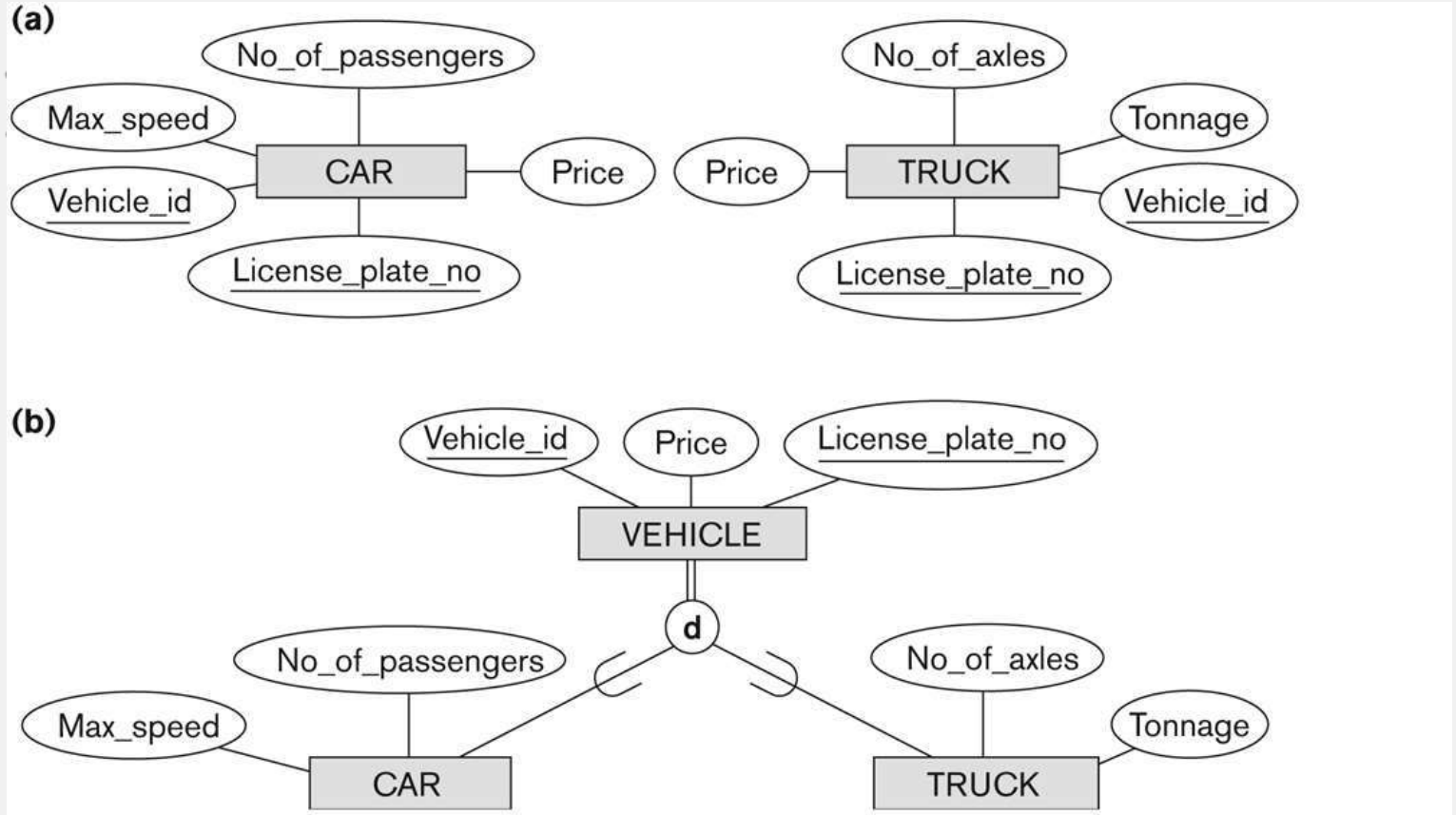


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Generalization and Specialization (1)

- Diagrammatic 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

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

Constraints on Specialization and Generalization (1)

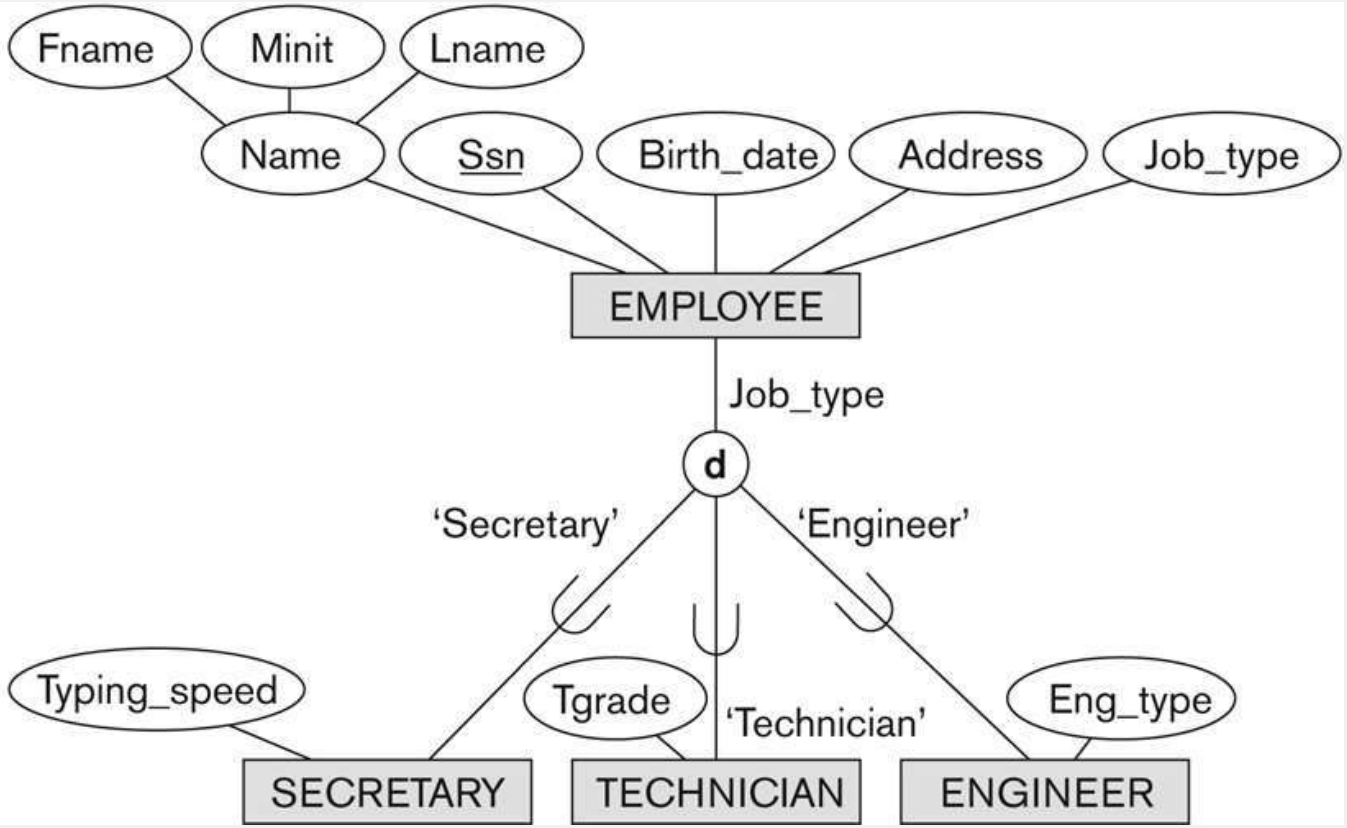
- If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called predicate-defined (or condition-defined) subclasses
  - Condition is a constraint that determines subclass members
  - Display a predicate-defined subclass by writing the predicate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

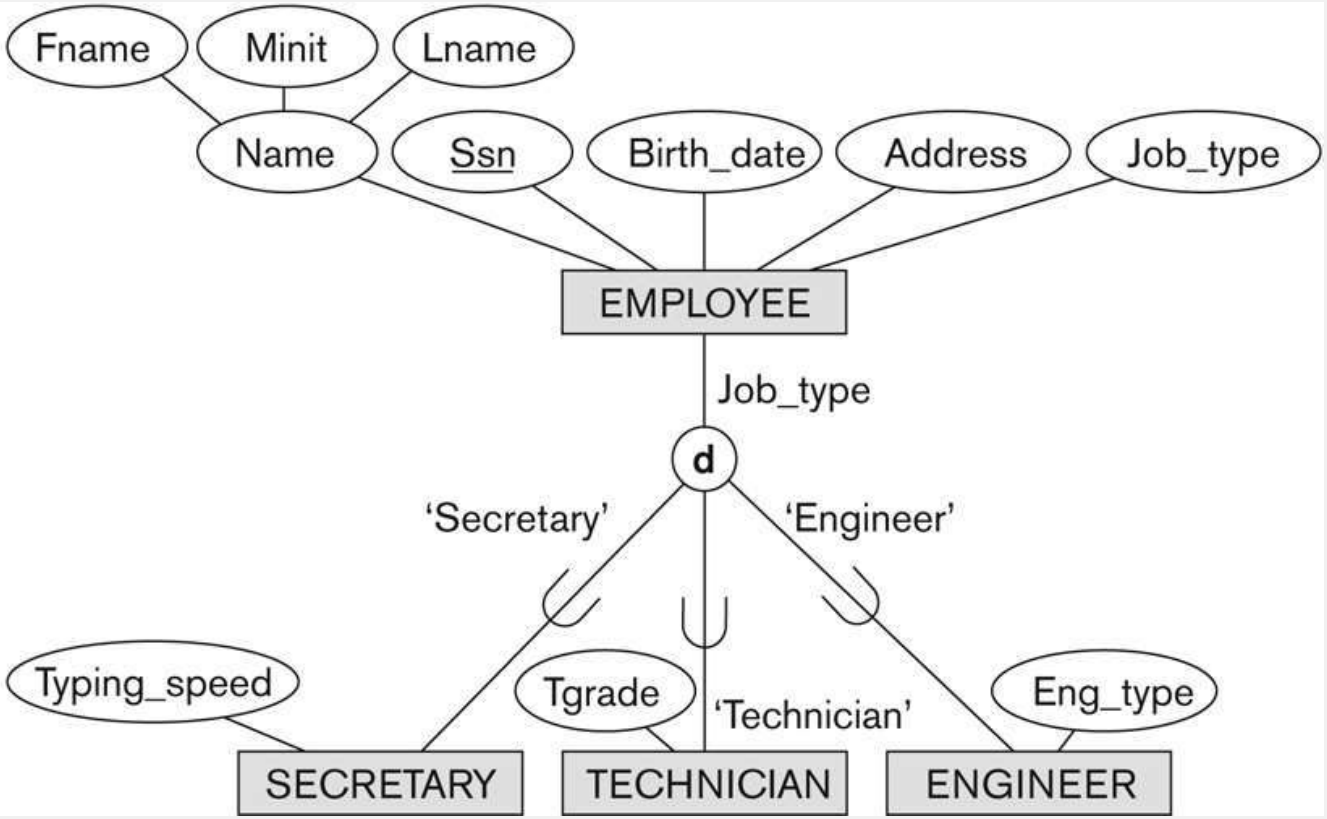
- Completeness Constraint:
  - *Total* specifies that every entity in the superclass must be a member of some subclass in the specialization/generalization
  - Shown in EER diagrams by a ***double line***
  - *Partial* allows an entity not to belong to any of the subclasses
  - Shown in EER diagrams 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

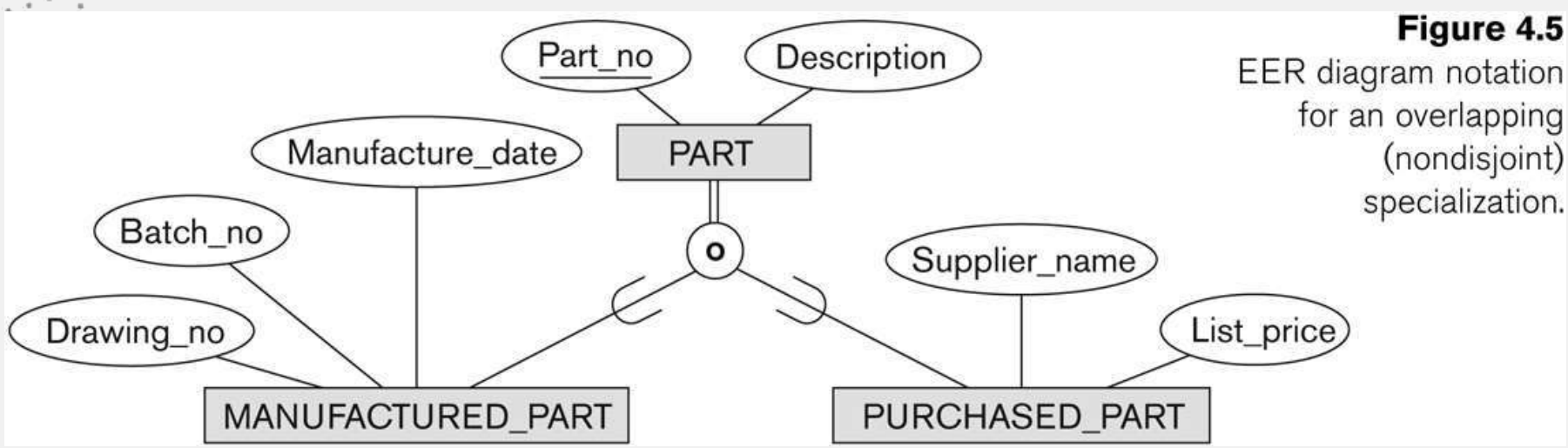


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

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***)

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

Shared Subclass “Engineering_Manager”

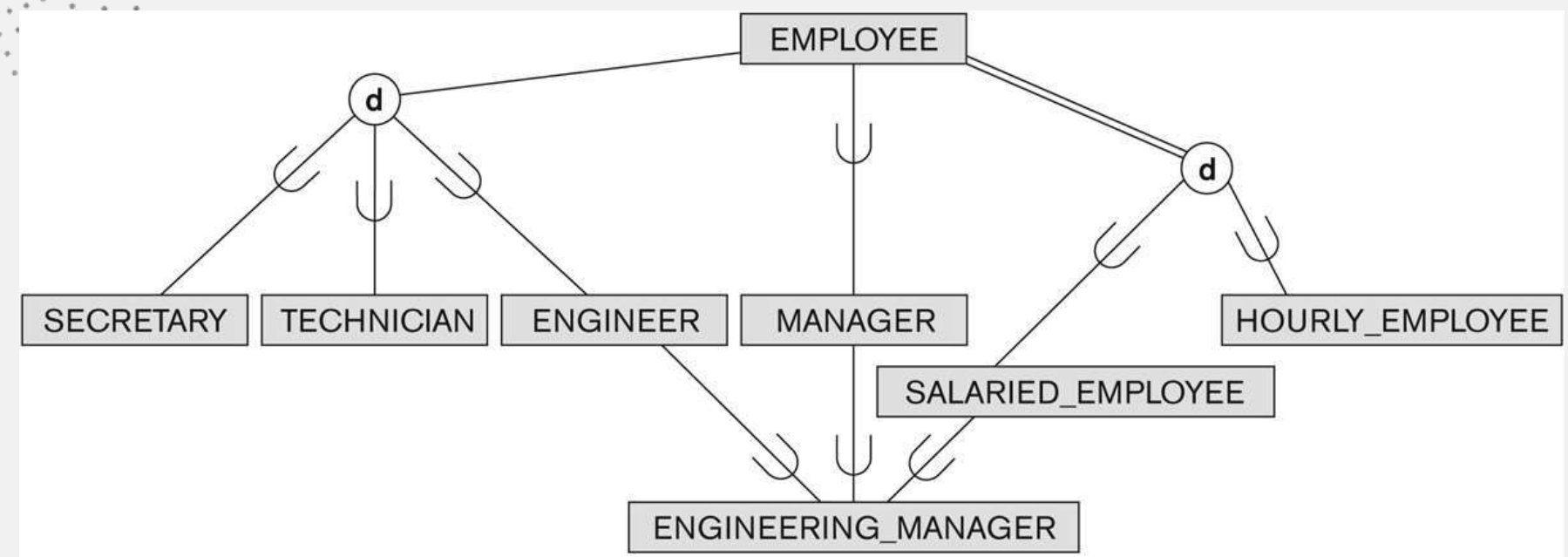


Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.

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)

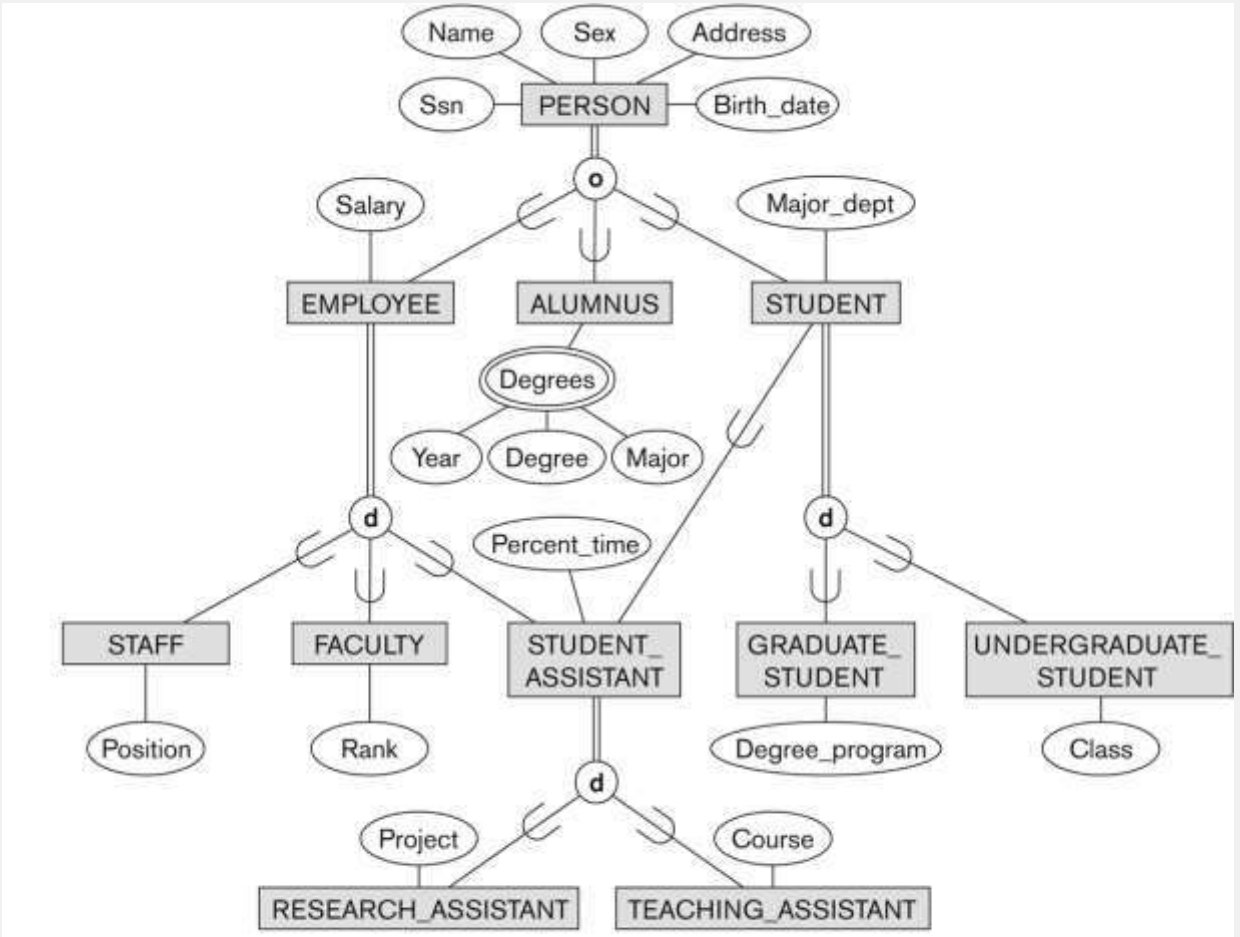


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

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*** 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

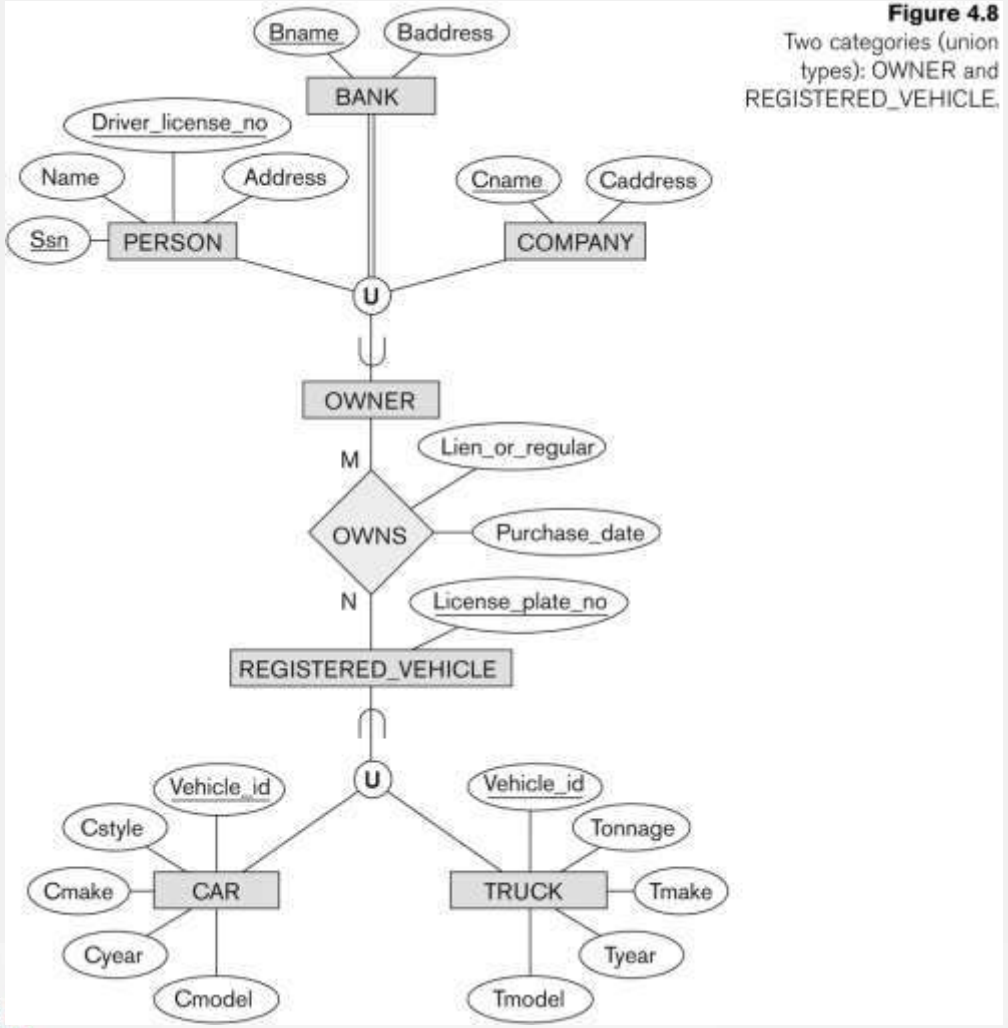


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Formal Definitions of EER Model (1)

- Class C:
  - A type of entity with a corresponding set of entities:
    - could be entity type, subclass, superclass, or category
- Note: The definition of *relationship type* in ER/EER should have 'entity type' replaced with 'class‘ to allow relationships among classes in general
- Subclass S is a class whose:
  - Type inherits all the attributes and relationship of a class C
  - Set of entities must always be a subset of the set of entities of the other class C
    - $S \subseteq C$
  - C is called the superclass of S
  - A superclass/subclass relationship exists between S and C

Formal Definitions of EER Model (2)

- Specialization Z: Z = {S1, S2,…, Sn} is a set of subclasses with same superclass G; hence, G/Si is a superclass relationship for i = 1, …., n.
  - G is called a generalization of the subclasses {S1, S2,…, Sn}
  - Z is total if we always have:
    - S1 ∪ S2 ∪ … ∪ Sn = G;
    - Otherwise, Z is partial.
  - Z is disjoint if we always have:
    - Si ∩ S2 empty-set for i ≠ j;
  - Otherwise, Z is overlapping.

Formal Definitions of EER Model (3)

- Subclass S of C is predicate defined if predicate (condition) p on attributes of C is used to specify membership in S;
  - that is, S = C[p], where C[p] is the set of entities in C that satisfy condition p
- A subclass not defined by a predicate is called user-defined
- Attribute-defined specialization: if a predicate A = ci (where A is an attribute of G and ci is a constant value from the domain of A) is used to specify membership in each subclass Si in Z
  - Note: If ci ≠ cj for i ≠ j, and A is single-valued, then the attribute-defined specialization will be disjoint.

Formal Definitions of EER Model (4)

- Category or UNION type T
  - A class that is a subset of the *union* of n defining superclasses D1, D2,…Dn, n>1:
    - $T \subseteq (D1 \cup D2 \cup \ldots \cup Dn)$
  - Can have a predicate pi on the attributes of Di to specify entities of Di that are members of T.
  - If a predicate is specified on every Di: $T = (D1[p1] \cup D2[p2] \cup \ldots \cup Dn[pn])$

SUMMARY

- Extended Entity Relationship (EER) modeling is an extension of the original Entity Relationship (ER) model used in database design.
- EER modeling includes additional concepts such as subclasses, superclasses, and inheritance relationships, which allow for more complex and flexible data modeling.
- Other advanced concepts in EER modeling include multivalued attributes, derived attributes, and constraints such as participation constraints and disjoint constraints.
- In a PowerPoint presentation on EER modeling, it is important to provide a clear overview of the benefits and features of EER modeling, as well as examples of how it can be used in real-world database design scenarios.

SUMMARY (Cont' d)

- Visual aids such as diagrams can help to explain the concepts and illustrate how entities, classes, and relationships are organized in an EER model.
- Additionally, it is important to highlight the potential challenges and limitations of EER modeling, such as the complexity of modeling and the need for specialized software tools.
- By providing a balanced and informative presentation on EER modeling, you can help your audience better understand the benefits and challenges of this advanced approach to database design.

SELF-ASSESSMENT QUESTIONS

1. What are participation constraints in EER modeling?

(a) Constraints that specify the minimum and maximum number of entities that must participate in a relationship Answer: a.

(b) Constraints that specify the type of relationship between entities

(c) Constraints that specify the uniqueness of values in an attribute

(d) Constraints that specify the order of values in an attribute

2. What is the purpose of disjoint constraints in EER modeling?

(a) To ensure that subclasses do not have overlapping sets of attributes and relationships… Answer: a

(b) To specify the minimum and maximum number of entities that must participate in a relationship

(c) To ensure that entities have unique values in a particular attribute

(d) To specify the type of relationship between entities

TERMINAL QUESTIONS

1. How does EER modeling extend the original ER model, and what benefits does it offer for database design?
2. What are subclasses and superclasses in EER modeling, and how do they help to organize and classify data?
3. Can you provide an example of how inheritance relationships can be used in EER modeling to simplify and streamline database design?
4. What are multivalued attributes, and how do they differ from other types of attributes in EER modeling?
5. What are participation constraints, and how do they ensure that relationships between entities are correctly defined and maintained?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. "Database System Concepts" by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan.

2. "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke.

3. "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant Navathe.

Sites and Web links:

1. Stanford Database Course: https://cs.stanford.edu/people/widom/cs145/

2. MIT OpenCourseWare - Database Systems: https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-830-database-systems-fall-2010/

3. Database Systems Concepts by Silberschatz, Korth and Sudarshan: http://www-db.cs.wisc.edu/courses/cs564-2009a/textbook/

THANK YOU

(DEEMED TO BE UNIVERSITY)

Team - DBMS