Extended Entity-Relationship (EER) Model

by

Dr. Pratik Roy

EER Model

- EER stands for **Enhanced ER** or **Extended ER**.
- Includes all modeling concepts of basic ER.
- Additional concepts:
  - subclasses/superclasses
  - **specialization**/**generalization**
  - **categories (UNION types)**
  - attribute and relationship inheritance
- Constraints on Specialization/Generalization
- The additional EER concepts are used to model applications more completely and more accurately.
- EER includes some object-oriented concepts, such as **inheritance**.

Subclass and Superclass

- 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.**
- 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.

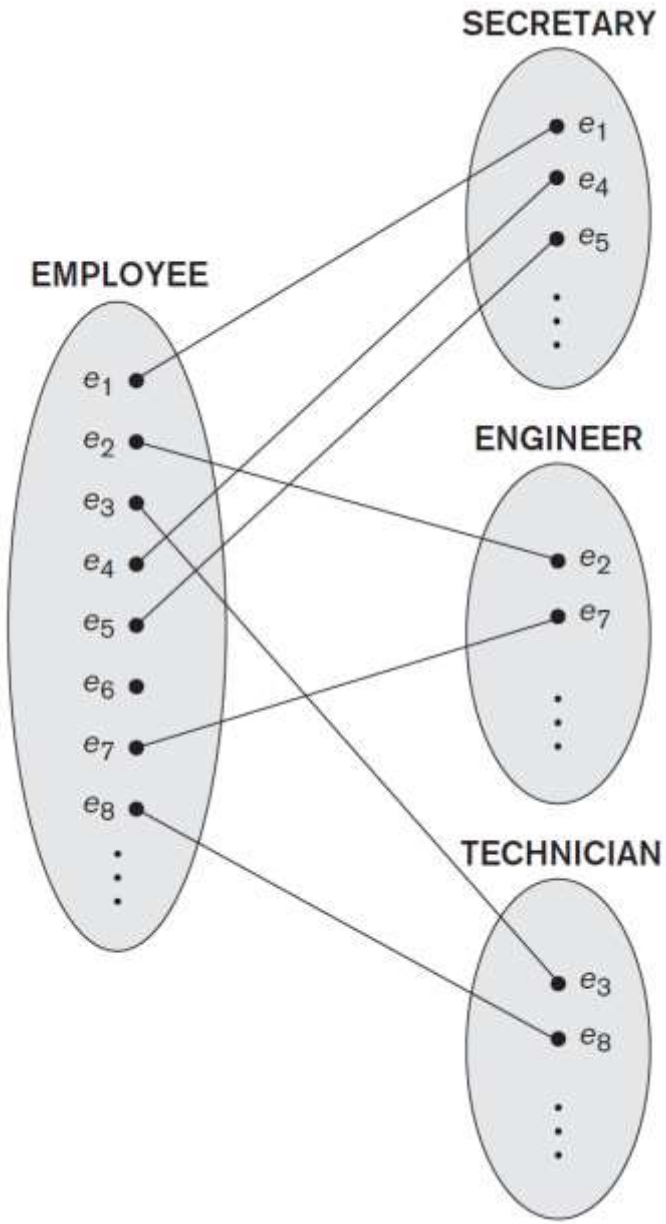
Subclass and Superclass

- Superclass/subclass relationships:
  - EMPLOYEE/SECRETARY
  - EMPLOYEE/TECHNICIAN
  - EMPLOYEE/MANAGER
- These are also called **IS-A** (or **IS-AN**) **relationships**
  - SECRETARY IS-AN EMPLOYEE, TECHNICIAN IS-AN EMPLOYEE
- 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 only 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.

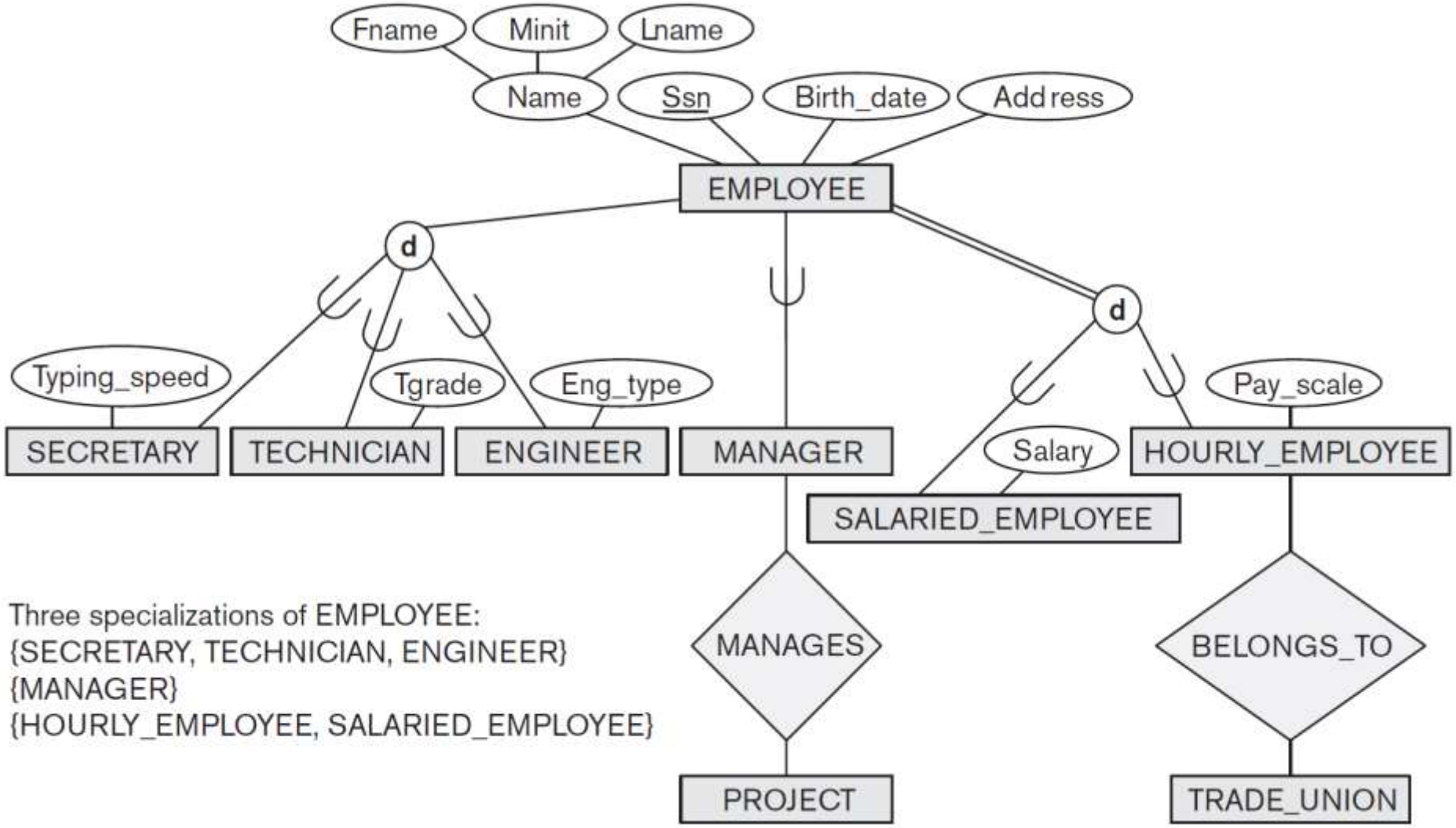
Subclass and Superclass

- **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

Subclass and Superclass



EER diagram notation to represent subclasses and specialization



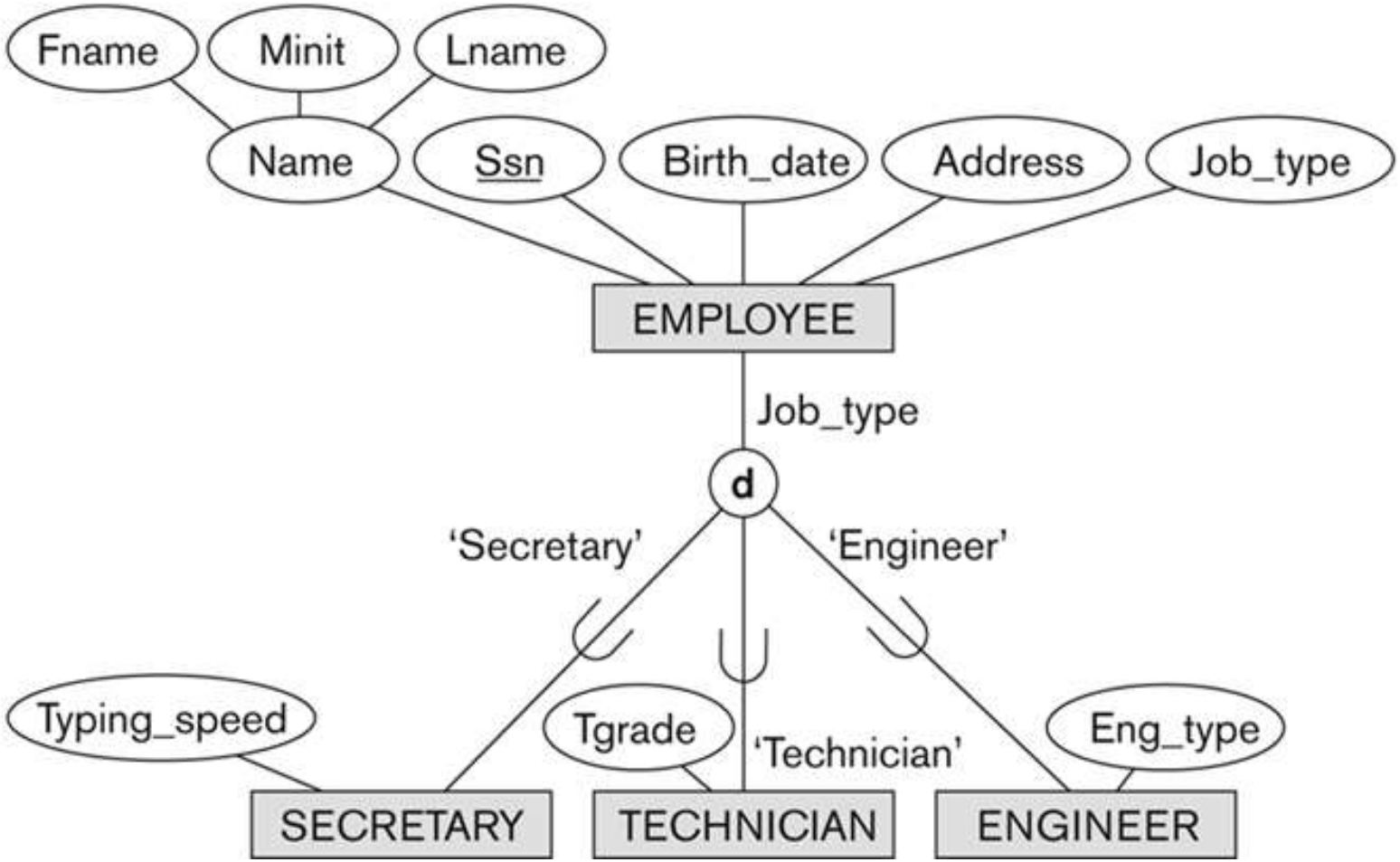
Inheritance

- 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**
  - 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

- The process of designating subgroupings within an entity set is called 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.*
  - MANAGER is a specialization of EMPLOYEE based on the *role the employee plays*.
  - {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE} is a specialization of EMPLOYEE based on *method of pay*.
- May have several specializations of the same superclass.
- Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams.

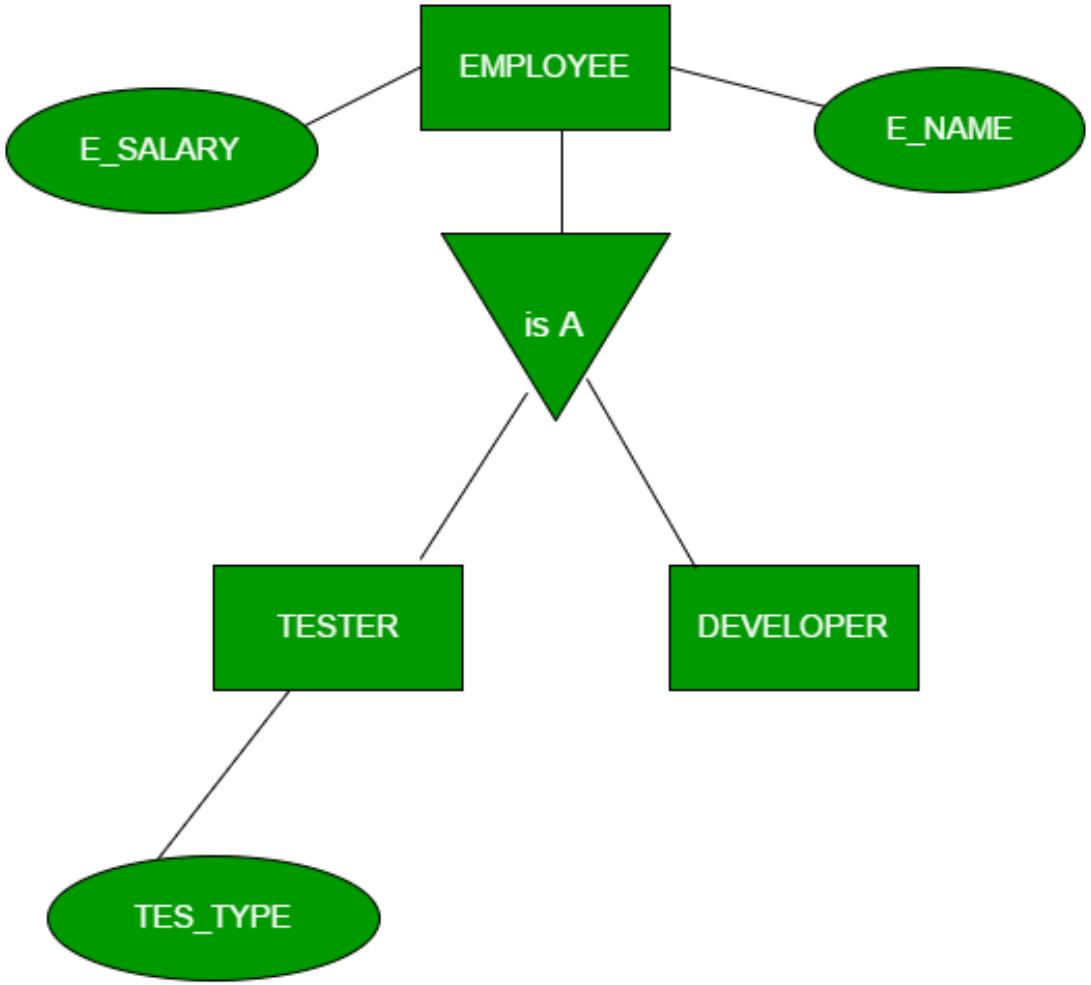
Specialization



Specialization

- 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
- **Top-down design process:** we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a triangle component labeled ISA.
- A lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

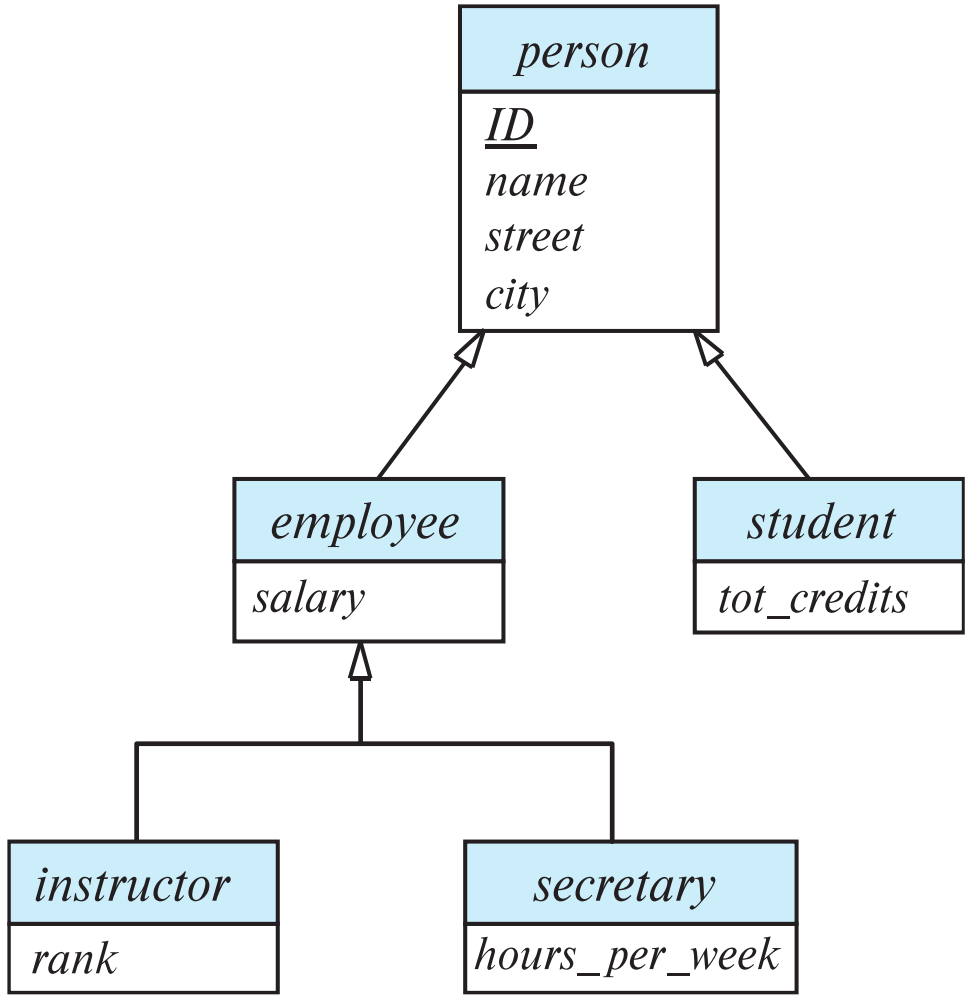
Specialization



Specialization

- **Overlapping specialization**
  - An entity may belong to multiple specialized entity sets
- **Disjoint specialization**
  - An entity must belong to at most one specialized entity set.
- For an overlapping specialization (as the case for MANAGER and ENGINEER as specializations of EMPLOYEE), two separate arrows are used.
- For a disjoint specialization (as the case for SECRETARY and MANAGER as specializations of EMPLOYEE), a single arrow is used.

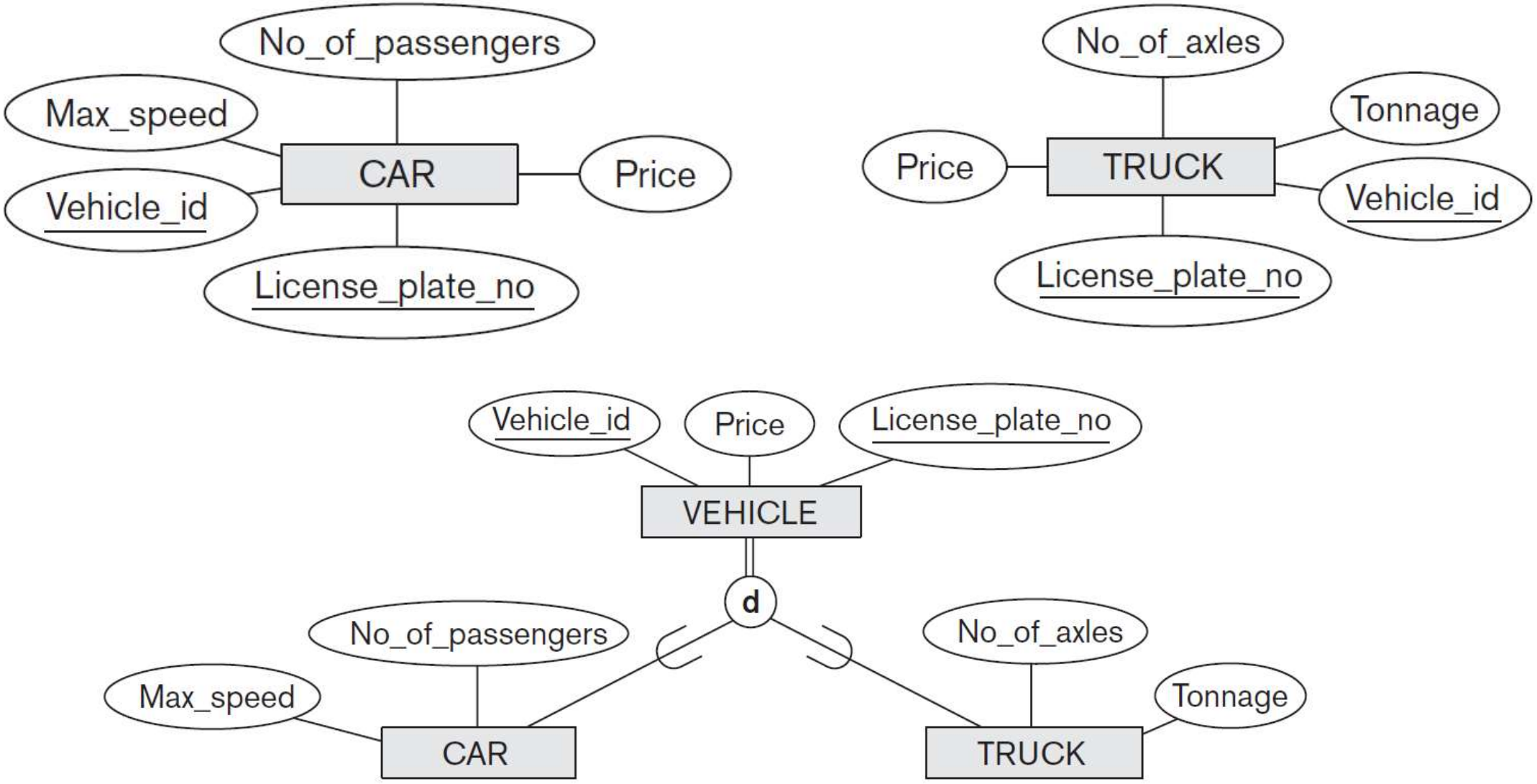
Specialization



Generalization

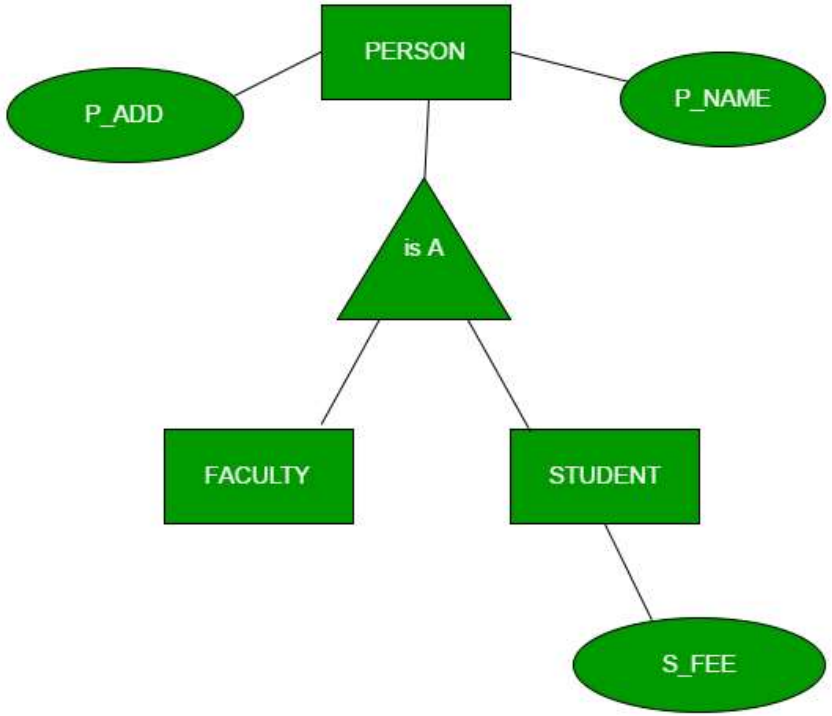
- Generalization is the reverse of the specialization process.
- Several classes with common features are generalized into a superclass.
  - Original classes become its subclasses
- **Example:** CAR, TRUCK generalized into VEHICLE
  - Both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE.
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK.
- Generalization refers to the process of defining a generalized entity type from the given entity types.
- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.

Generalization



Notations for 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



Generalization

- **Bottom-up design process** – combine a number of entity sets that share the same features (attributes) into a higher-level entity set.
- Specialization and generalization are inversions of each other.
- **Single inheritance** - In a hierarchy, a given entity set is involved as a lower-level entity set in only one ISA relationship.
- **Multiple inheritance -** If an entity set is a lower-level entity set in more than one ISA relationship, then the entity set has multiple inheritance.

Constraints

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

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*
  - same entity may be a member of more than one subclass of the specialization
- Specified by ***o*** in EER diagram

Constraints

Completeness Constraint

- **Total** specialization/generalization 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* to connect the superclass to the circle.
- **Partial** specialization/generalization allows an entity not to belong to any of the subclasses.
  - Shown in EER diagrams by a single line.

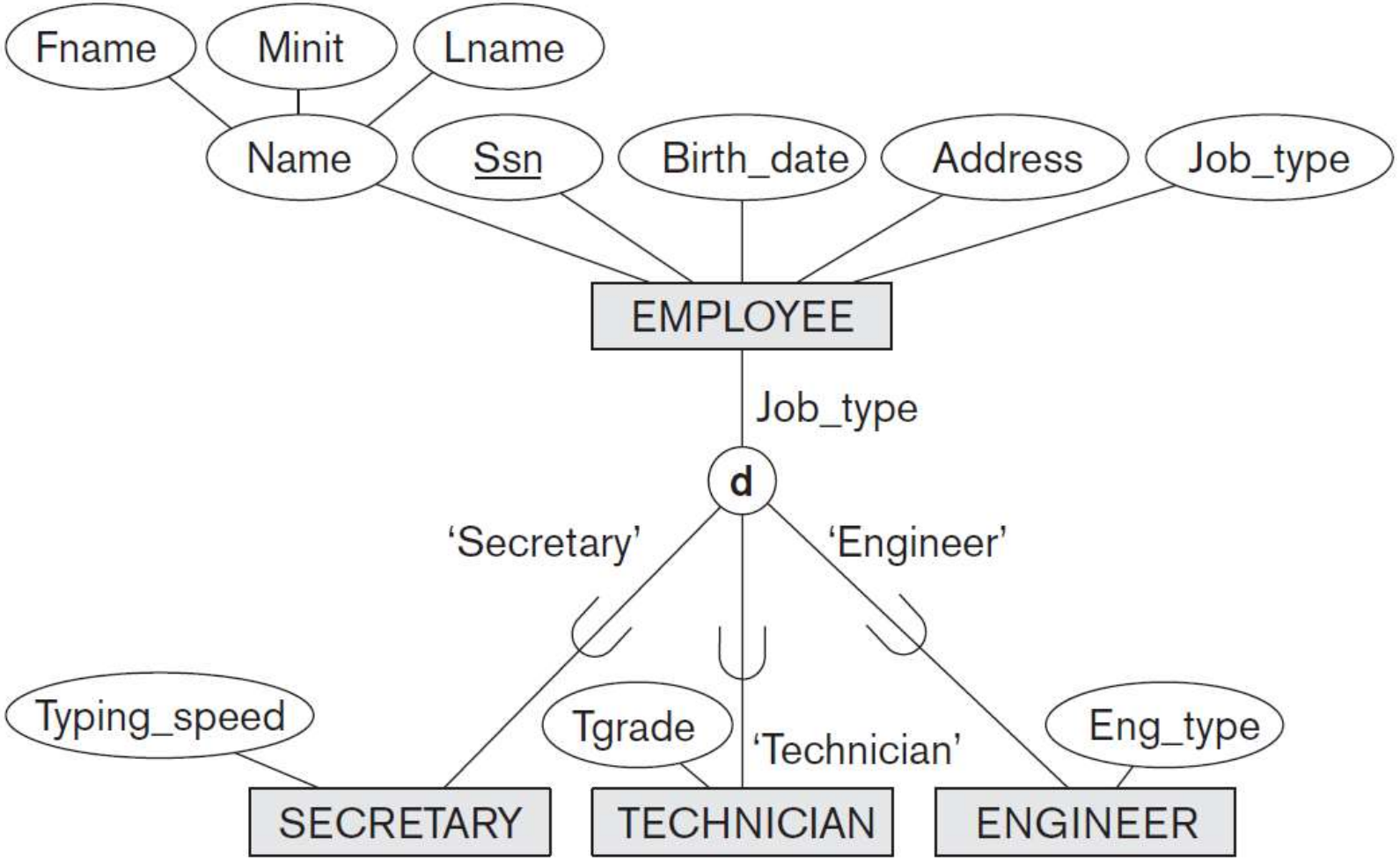
We have four types of specialization/generalization:

- Disjoint, total
- Disjoint, partial
- Overlapping, total
- Overlapping, partial

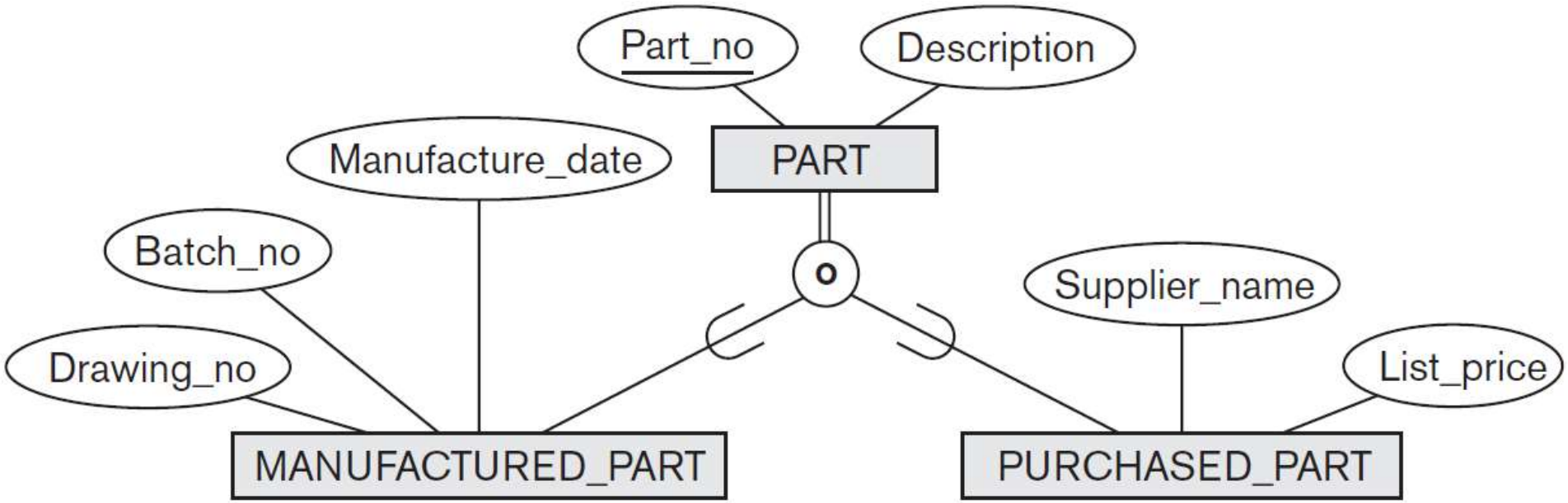
Generalization usually is total because the superclass is derived from the subclasses

- Contains only the entities that are in the subclasses

Example of disjoint partial Specialization



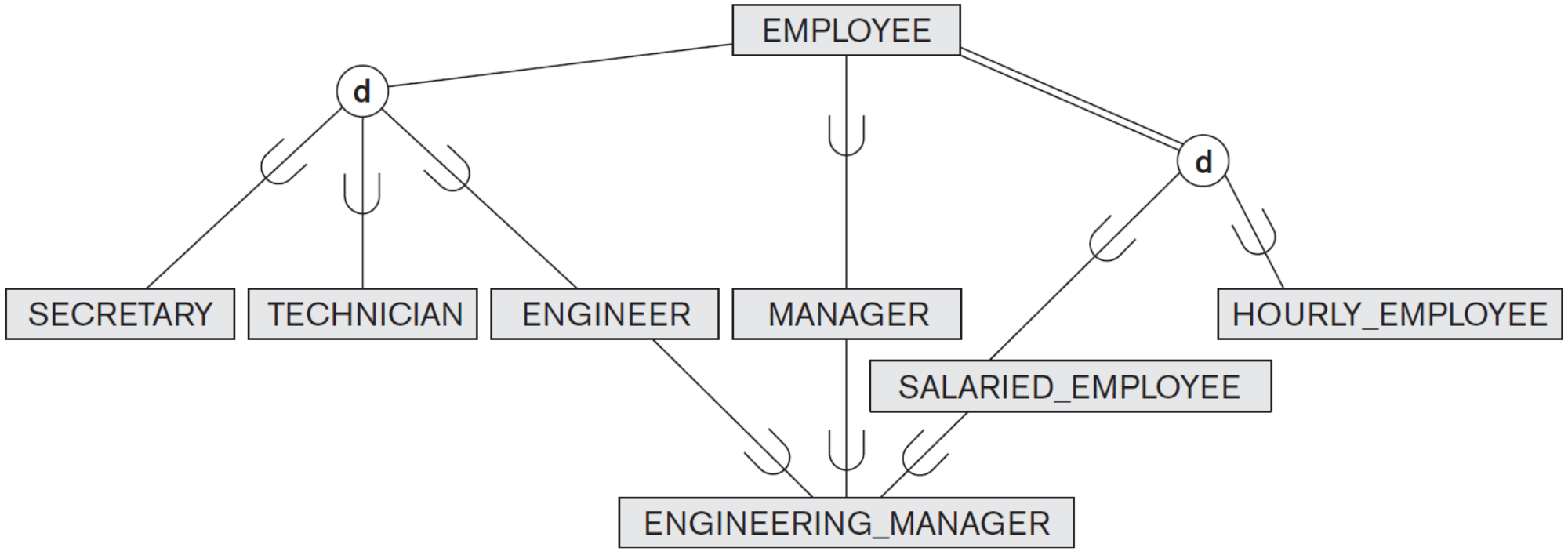
Example of overlapping total Specialization



Specialization/Generalization Hierarchies

- 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**
  - Basically a **tree structure**
- In a **lattice**, a subclass can be subclass of more than one superclass
  - Called **multiple inheritance**

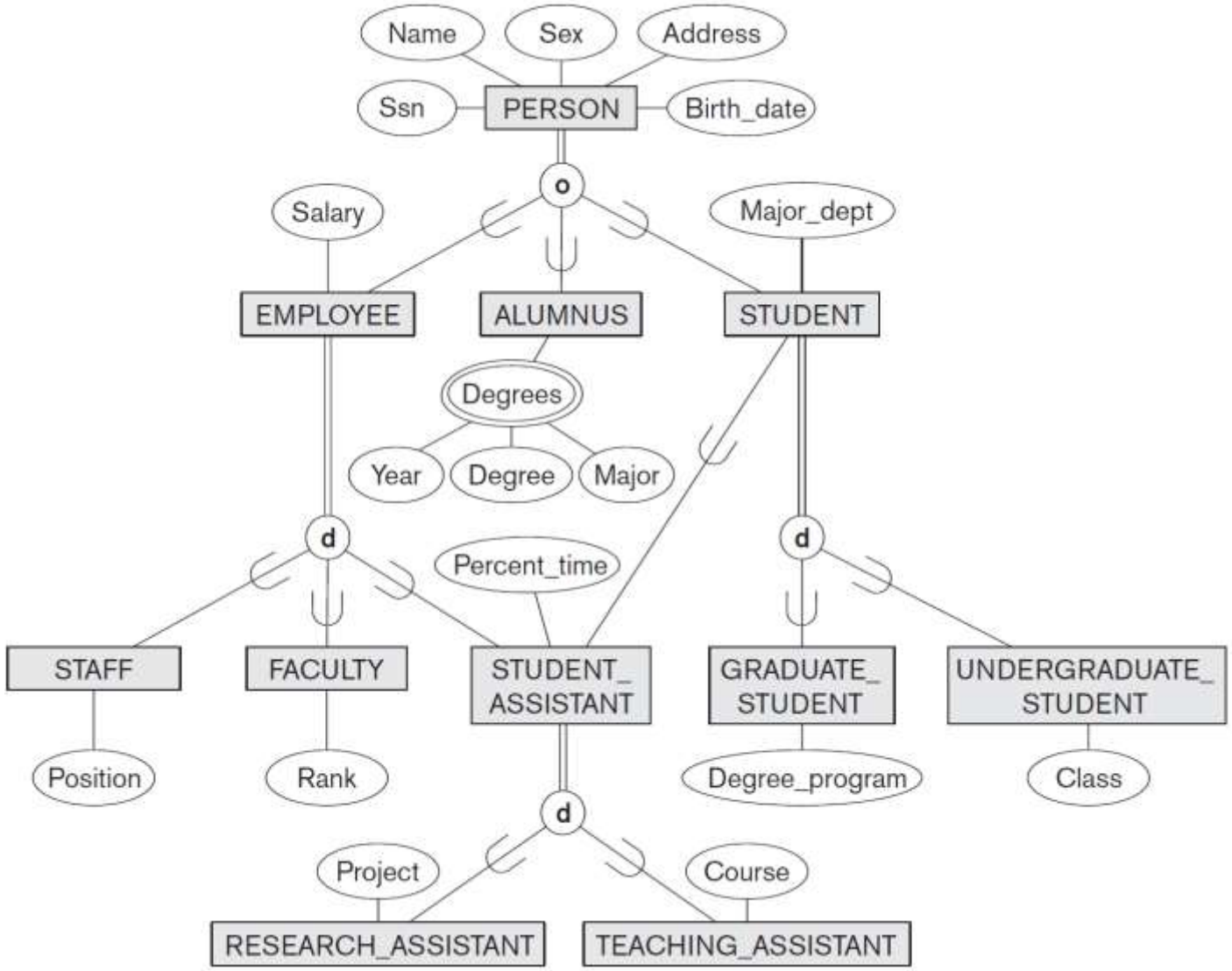
Specialization lattice



Specialization/Generalization Hierarchies and Lattices

- 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)
- 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)



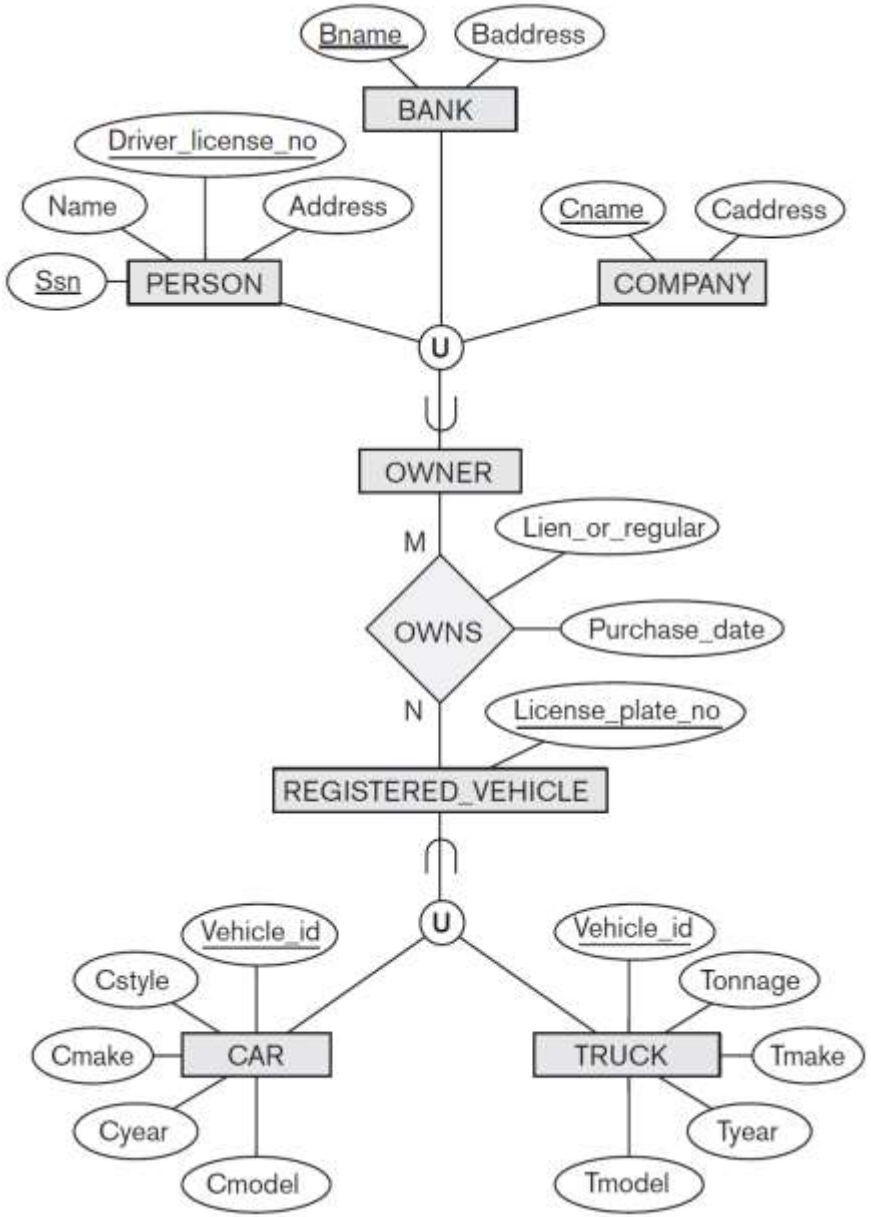
Categories (UNION TYPES)

- 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**.

Categories (UNION TYPES)

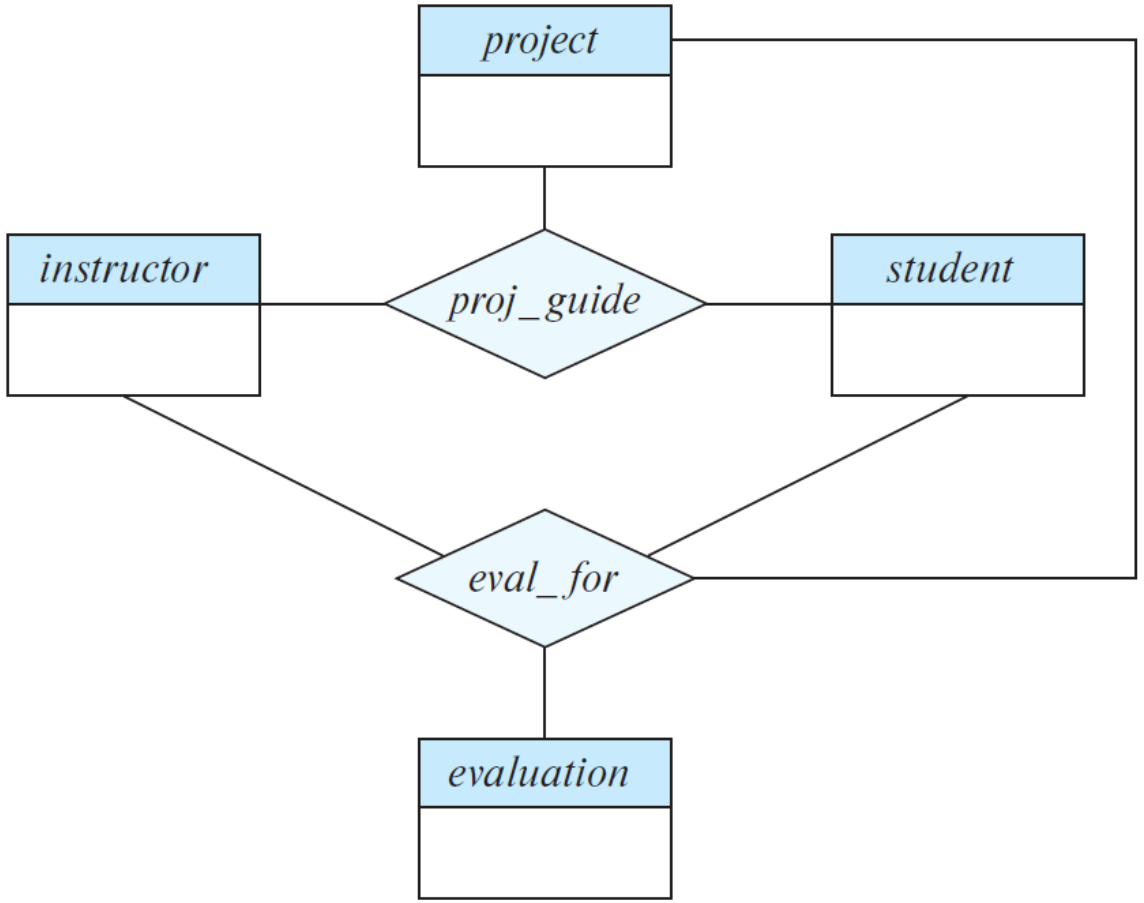
- **Example**
  - In a database for vehicle registration, a vehicle owner can be a PERSON, a BANK (holding a lien on a vehicle) or a COMPANY.
  - A *category* (UNION type) called OWNER is created to represent a subset of the *union* of the three superclasses COMPANY, BANK, and PERSON.
- A category member must exist in **at least one** of its superclasses.
- A category has two or more superclasses that may represent *distinct entity types*.
- Difference from shared subclass, which is a:
  - subset of the *intersection* of its superclasses
  - shared subclass member must exist in *all* of its superclasses
- Category is a subset of the *union* of its superclasses.
- A member of OWNER must exist in only one of the superclasses.

Categories (UNION TYPES)



- Category can be **total** or **partial**.
- Total category holds the *union* of all entities in its superclasses.
- Partial category can hold a *subset of the union*.

Aggregation

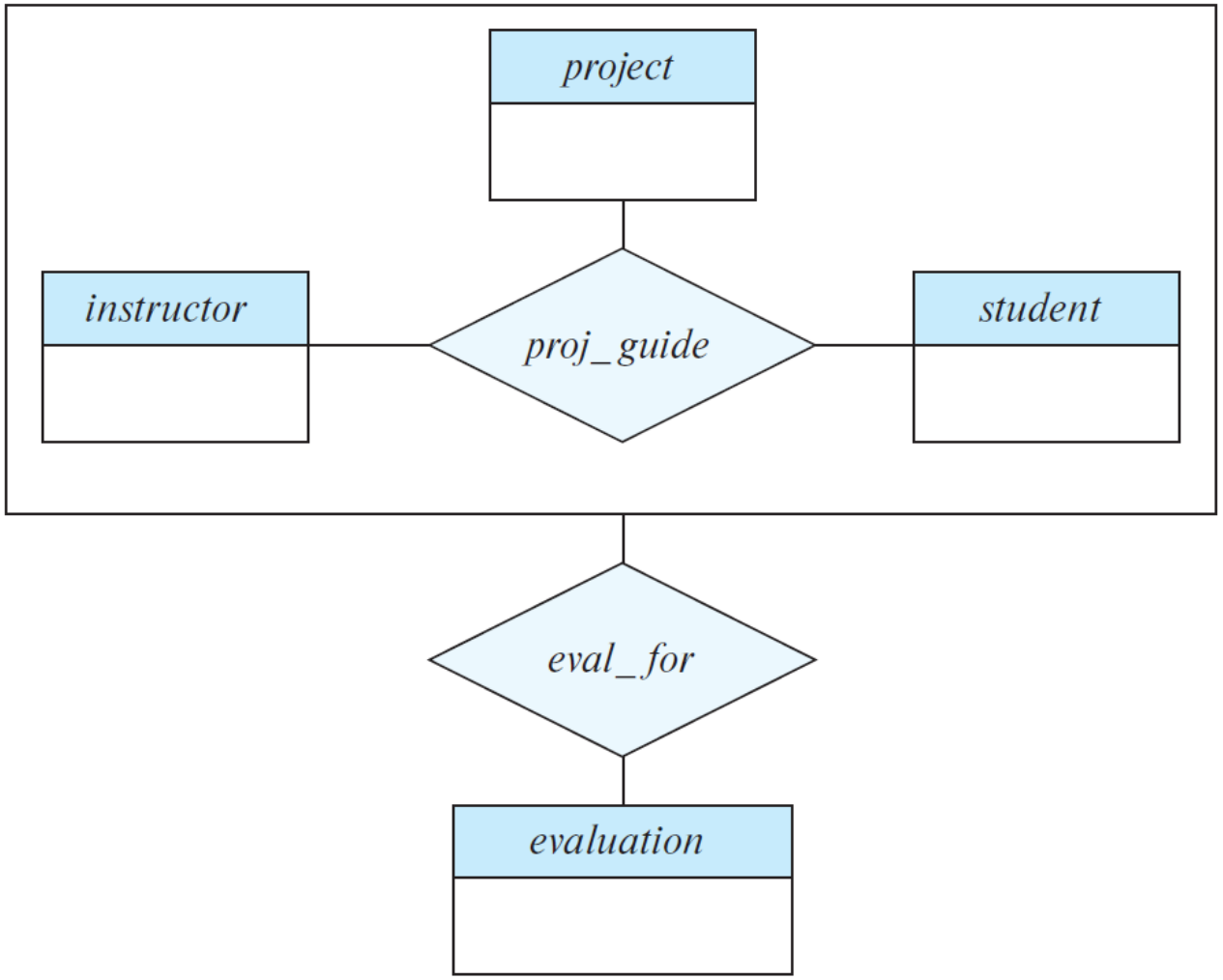


- Consider the ternary relationship *proj_guide*.
- Suppose we want to record evaluations of a student by a guide on a project.

Aggregation

- Relationship sets *eval_for* and *proj_guide* represent overlapping information.
- There is redundant information, since every instructor, student, project combination in *eval_for* must also be in *proj_guide*.
- Every *eval_for* relationship corresponds to a *proj_guide* relationship.
- Some *proj_guide* relationships may not correspond to any *eval_for* relationships.
- We can't discard the *proj_guide* relationship.
- Eliminate this redundancy via **aggregation**
  - Treat relationship as an abstract entity
  - Allows relationships between relationships
  - Abstraction of relationship into new entity

Aggregation

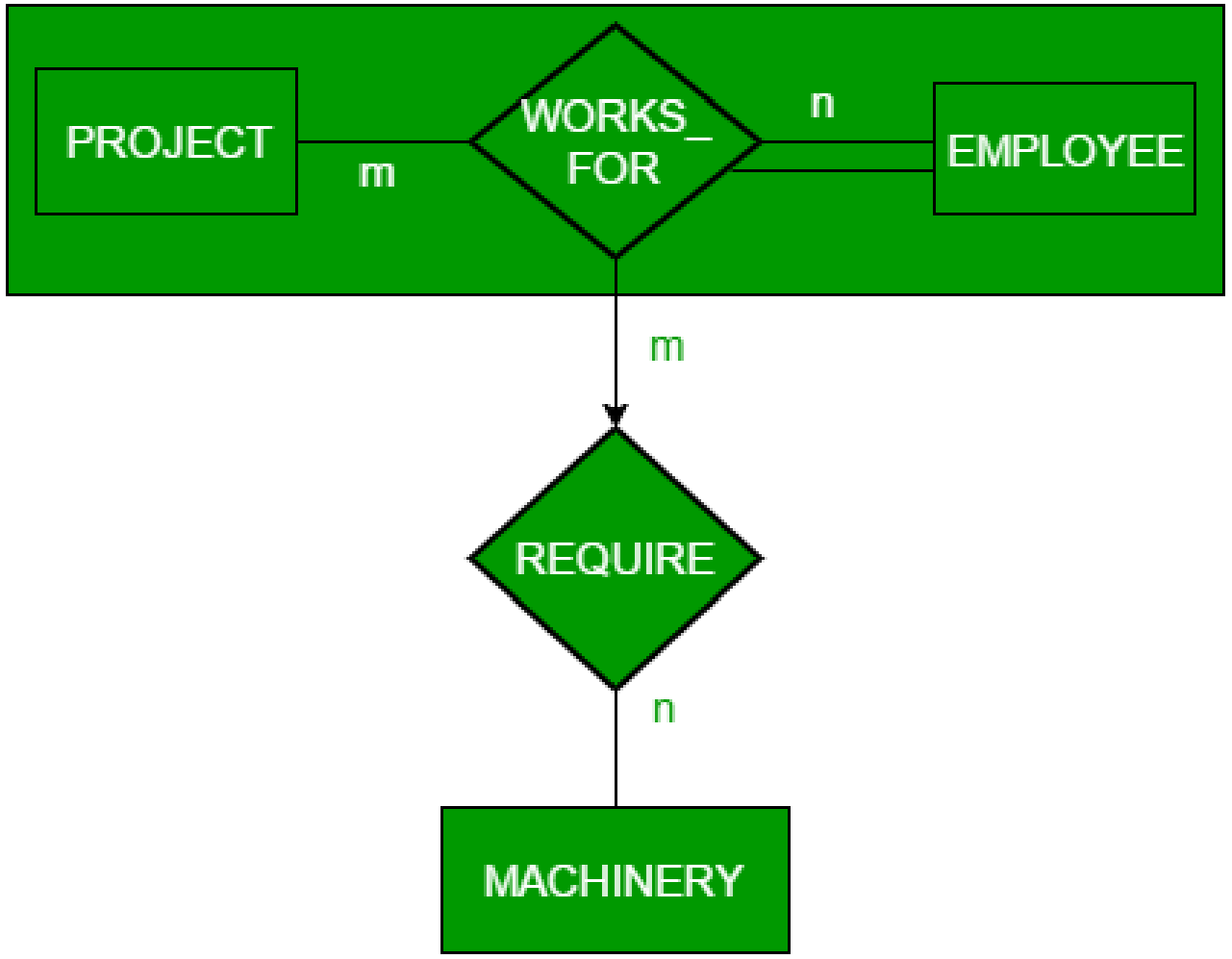


- Aggregation is an abstraction through which relationships are treated as higher-level entities.
- We regard relationship set *proj guide* (relating entity sets instructor, student, and project) as a higher-level entity set *proj_guide*.
- A student is guided by a particular instructor on a particular project.
- A student, instructor, project combination may have an associated evaluation

Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
- An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios.
- In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.
- Aggregation is an abstraction through which we can represent relationships as higher level entity sets.

Aggregation



- Employee working for a project may require some machinery.
- REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY.
- Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity.
- Relationship REQUIRE is created between

Formal Definitions for the EER Model Concepts

- A **class** is a set or collection of entities
  - could be entity type, subclass, superclass, or category
- **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.
- **Specialization** $Z = \{S_1, S_2, \ldots, S_n\}$ is a set of subclasses with same superclass G.
- G is called generalized entity type (or the superclass of the specialization, or **generalization** of the subclasses $\{S_1, S_2, \ldots, S_n\}$).

Formal Definitions for the EER Model Concepts

- Z is **total** if we always have:
  - $S_1 \cup S_2 \cup \ldots \cup S_n = G$
- Otherwise, Z is partial.
- Z is **disjoint** if we always have:
  - $S_i \cap S_j = \emptyset$ (empty-set) for $i \neq j$
- Otherwise, Z is **overlapping**.
- **Category** or **UNION type** T
  - A class that is a subset of the union of n different superclasses $D_1, D_2, \ldots, D_n$, $n>1$
  - $T \subseteq (D_1 \cup D_2 \cup \ldots \cup D_n)$