# Enhanced Entity-Relationship (EER) Model

## Database Design


Department of Computer Engineering

Sharif University of Technology

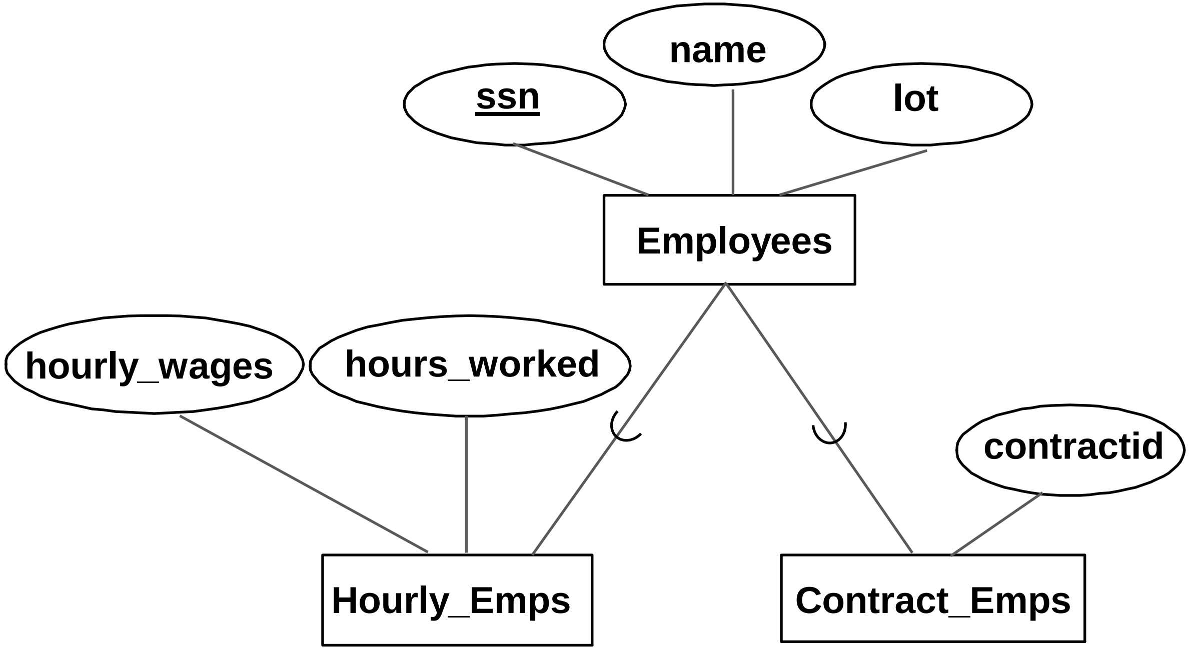
Maryam Ramezani maryam.ramezani@sharif.edu

# Subclasses, Superclasses, and Inheritance

- In many cases an entity type has numerous subgroupings or subtypes (which is called subclass or subtype) of its entities that are meaningful and need to be represented explicitly because of their significance to the database application.
  - Example: Employee is a entity type called superclass or supertype. SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE are subclassess or subtypes.
- An entity that is a member of a subclass inherits all the attributes of the entity as a member of the superclass.
- An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass.
- It is not necessary that every entity in a superclass is a member of some subclass.
- A class/subclass relationship is often called an IS-A (or IS-AN) relationship because of the way we refer to the concept. We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE, and so on.

# ISA (`is a' ) Hierarchies

- If we declare A ISA B, every A entity is also considered to be a B entity



# Specialization

- Specialization is the process of defining a set of subclasses of an entity type; this entity type is called the superclass of the specialization.
- Why we need specialization?
  - Certain attributes may apply to some but not all entities of the superclass entity type.
  - Some relationship types may be participated in only by entities that are members of the subclass

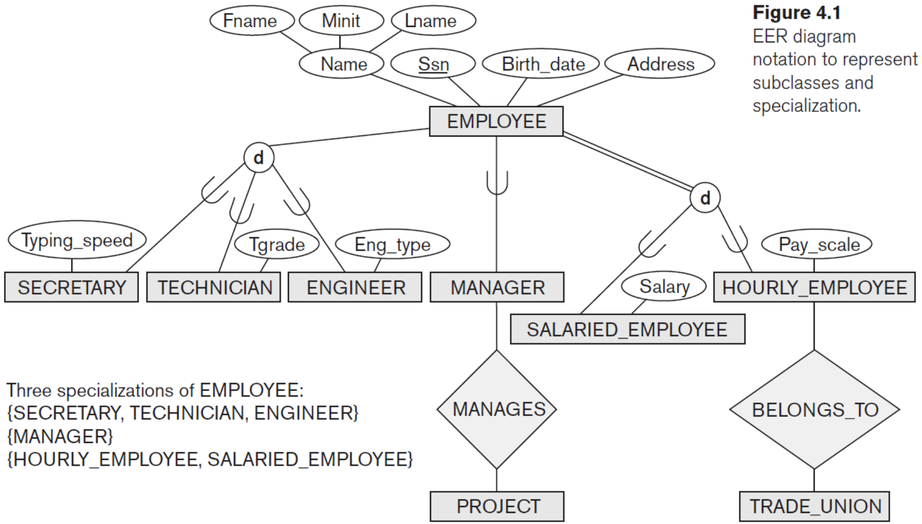
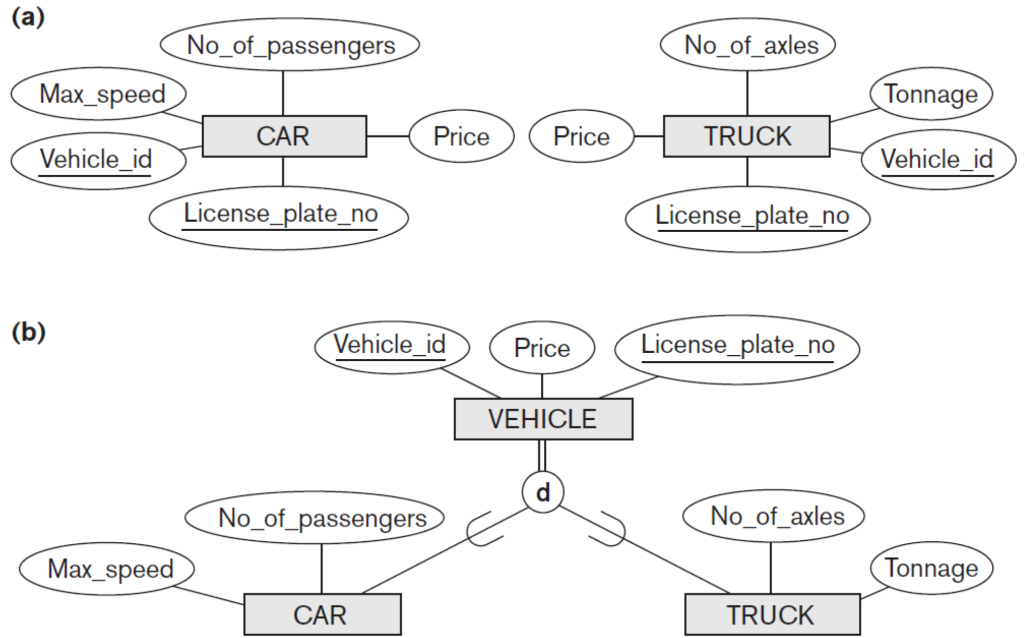



**Figure 4.1**
EER diagram notation to represent subclasses and specialization.

# Generalization

- We can think of a reverse process of abstraction in which we suppress the differences among several entity types, identify their common features, and generalize them into a single superclass of which the original entity types are special subclasses.

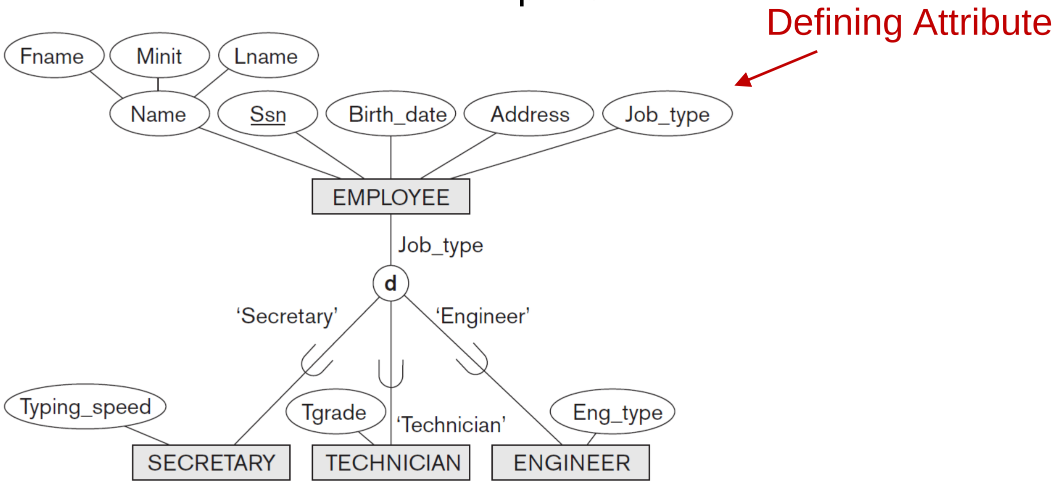


# Process for Finding SuperClass and SubClass

| Generalization | Specialization |
|---|---|
| Bottom Up Approach | Top Down Approach |
| Schema Size Reduce | Schema Size Increases |
| Combining Entities | Splitting an Entities |

# Constraints on Specialization and Generalization

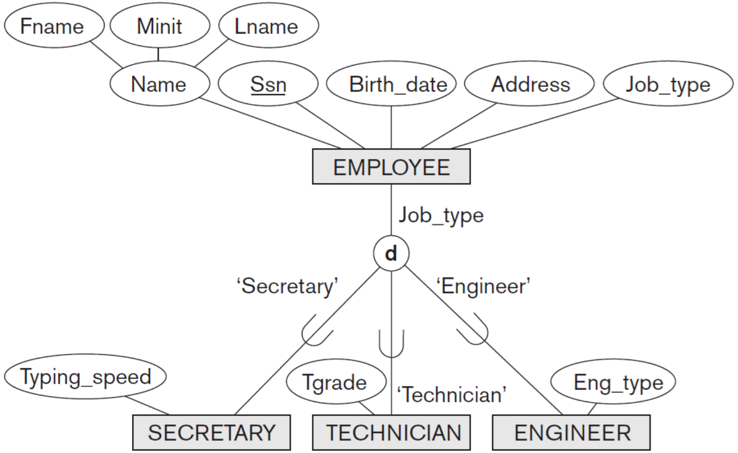
- Attribute–defined: If all subclasses in a specialization have their membership condition on the same attribute of the superclass.



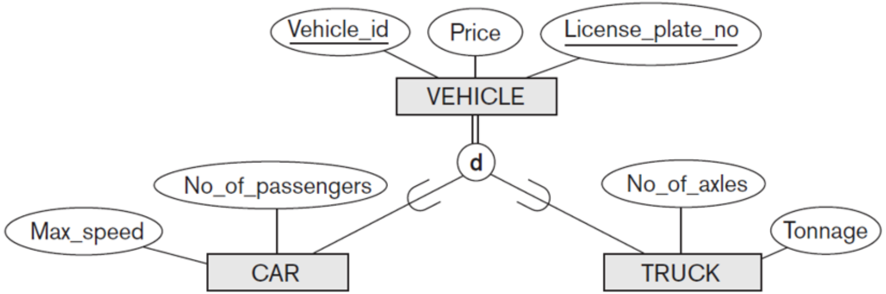

- User–defined: When we do not have a condition for determining membership in a subclass. Membership is specified individually for each entity by the user, not by any condition that may be evaluated automatically.

# Constraints on Specialization and Generalization

- Disjointness constraint: subclasses of the specialization must be disjoint sets: an entity can be a member of at most one of the subclasses.
  - EER: **d** in the circle stands for disjoint
  - Example:



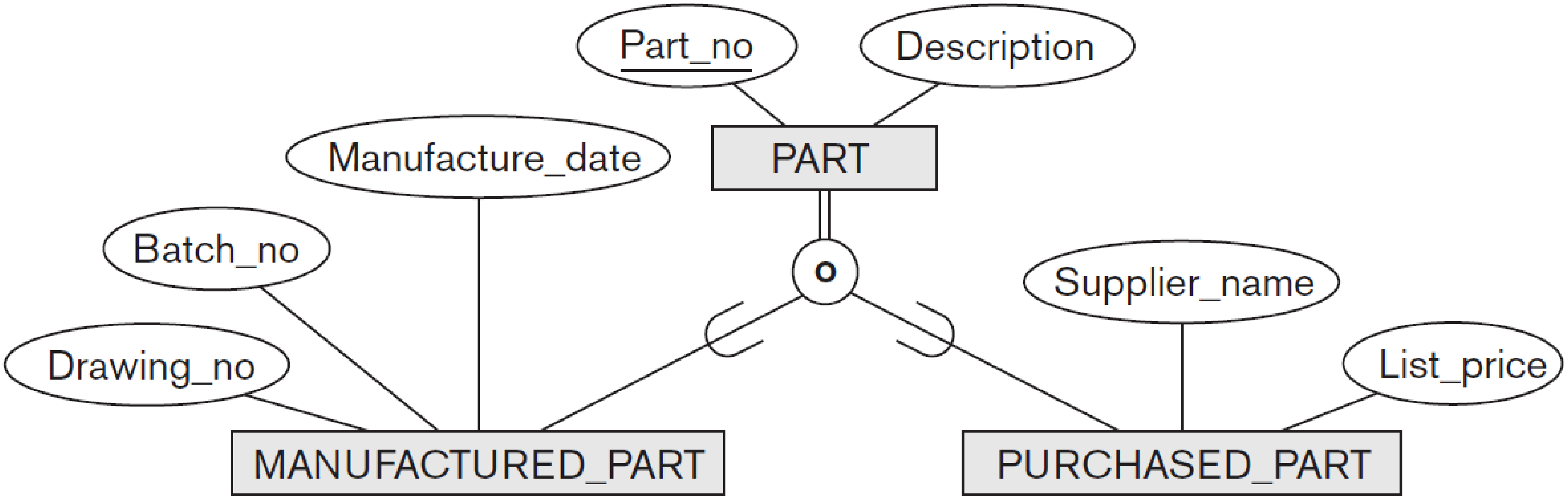

Attribute-defined




User-defined

# Constraints on Specialization and Generalization

- **Overlapping** constraint: If the subclasses are not constrained to be disjoint.
  - EER: **O** in the circle stands for disjoint
  - Example:



# Constraints on Specialization and Generalization

- Completeness (totalness): Every entity in the superclass must be a member of at least one subclass in the specialization.
  - EER: using a double line to connect the superclass to the circle.

- Partial: Allows an entity not to belong to any of the subclasses.
  - EER: using a single line to connect the superclass to the circle.

- Note: The notation of using single or double lines is similar to that for partial or total participation of an entity type in a relationship type,

# Constraints on Specialization and Generalization

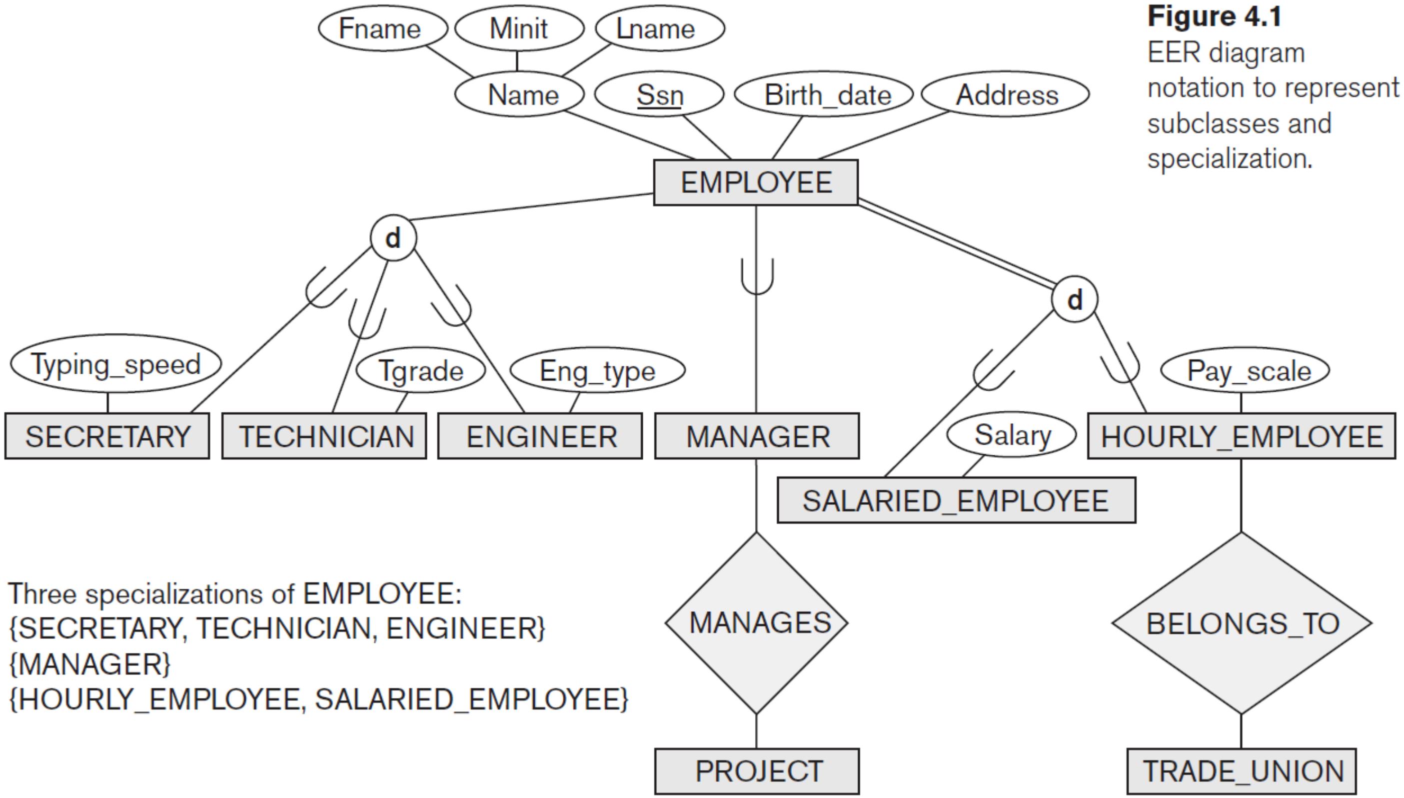


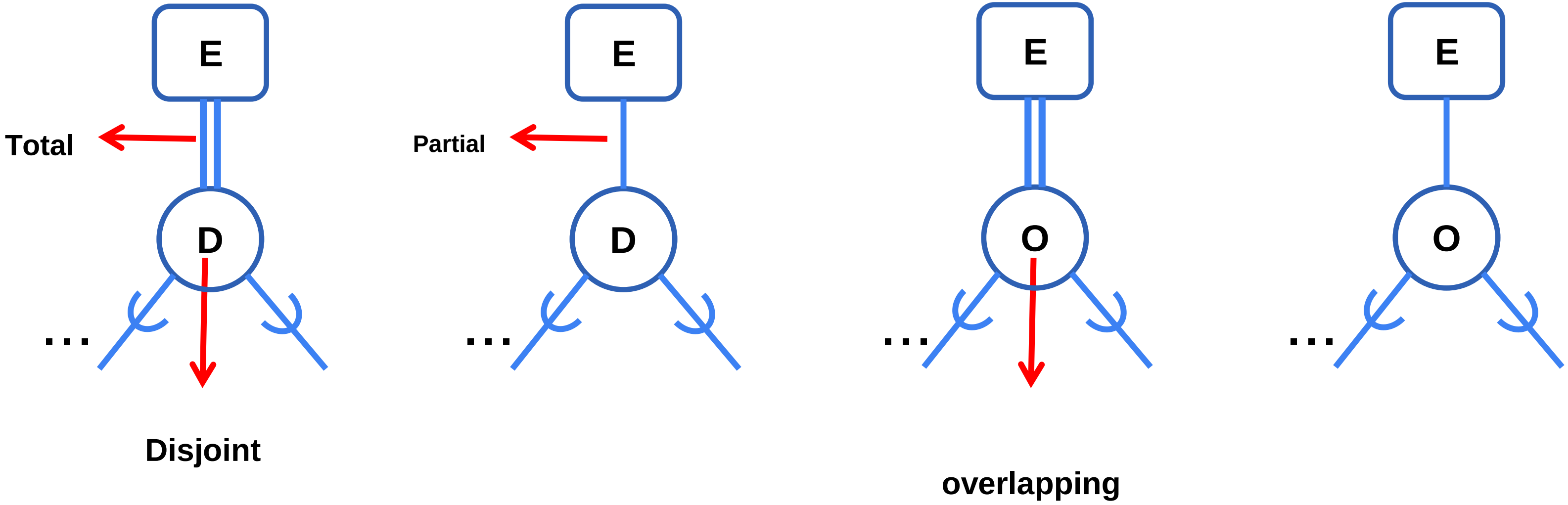

**Figure 4.1**
EER diagram notation to represent subclasses and specialization.

# Constraints on Specialization and Generalization

- In general, a superclass that was identified through the generalization process usually is total, because the superclass is derived from the subclasses and hence contains only the entities that are in the subclasses
- Conclusion

| | Disjointness | | Overlapping | |
|---|---|---|---|---|
| | Total | Partial | Total | Partial |
| **Attribute-Defined** | | | | |
| **User-Defined** | | | | |

# “ISA” Conclusion

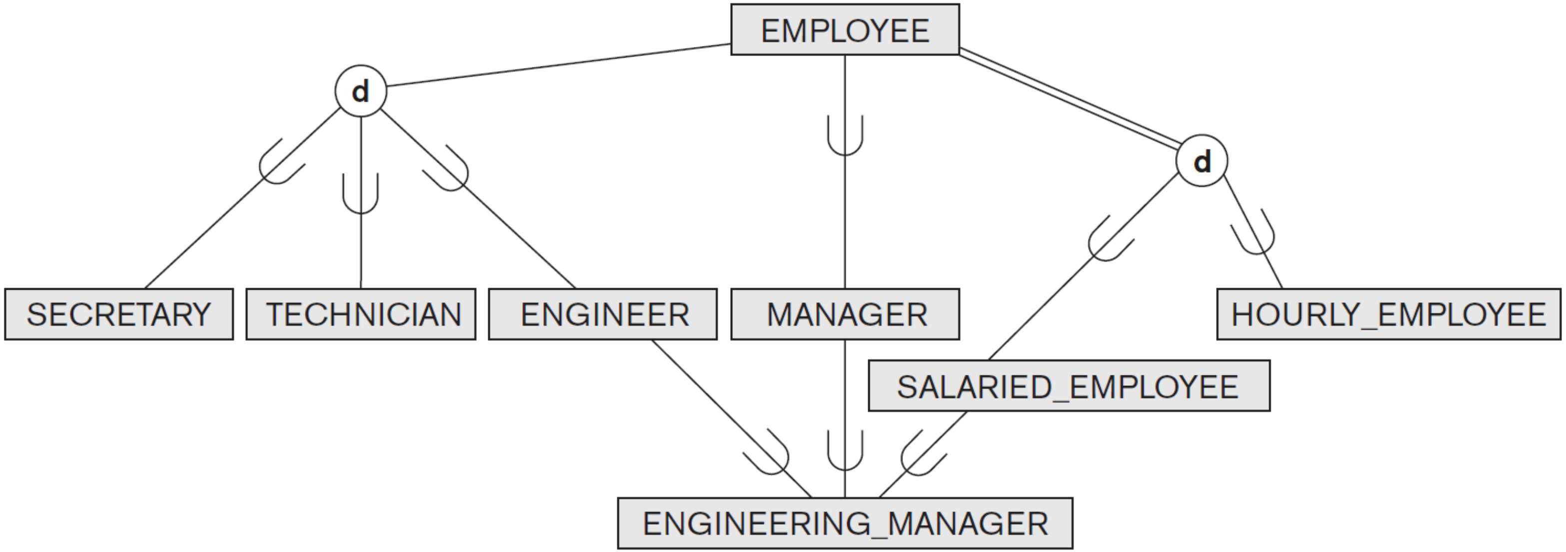


## Insertion and deletion rules apply to specialization (and generalization)

- Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs.
- Inserting an entity in a superclass implies that the entity is mandatorily inserted in all attribute-defined subclasses for which the entity satisfies the defining predicate.
- Inserting an entity in a superclass of a total specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization.

# Hierarchies & Lattices

- Tree structure or Strict hierarchy: each subclass has only one parent.
- Specialization lattice: A subclass can be a subclass in more than one class/subclass relationship.



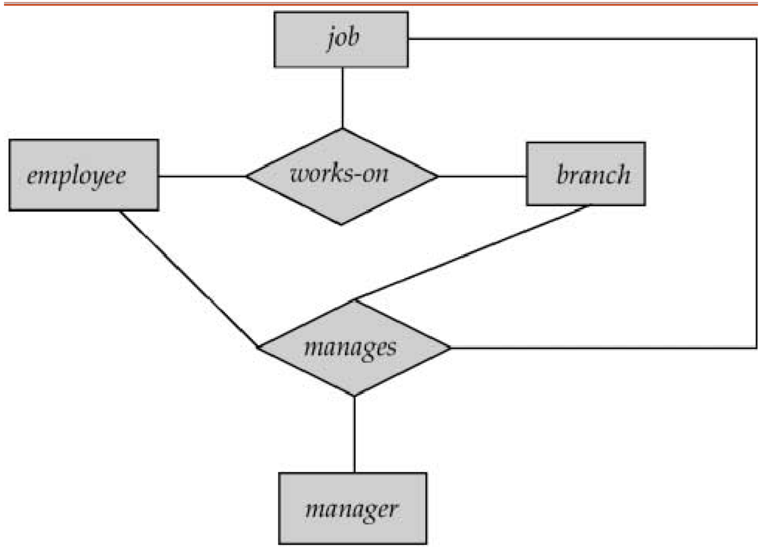
**The requirements for the part of the UNIVERSITY database:**

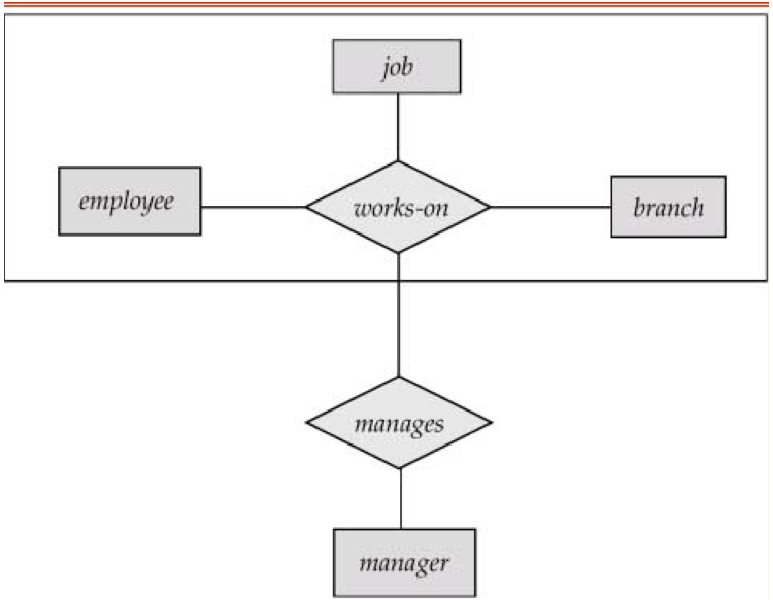
- The database keeps track of three types of persons: employees, alumni, and students. A person can belong to one, two, or all three of these types. Each person has a name, SSN, sex, address, and birth date.
- Every employee has a salary, and there are three types of employees: faculty, staff, and student assistants. Each employee belongs to exactly one of these types. For each alumnus, a record of the degree or degrees that he or she earned at the university is kept, including the name of the degree, the year granted, and the major department. Each student has a major department.
- Each faculty has a rank, whereas each staff member has a staff position. Student assistants are classified further as either research assistants or teaching assistants, and the percent of time that they work is recorded in the database. Research assistants have their research project stored, whereas teaching assistants have the current course they work on.
- Students are further classified as either graduate or undergraduate, with the specific attributes degree program (M.S., Ph.D., M.B.A., and so on) for graduate students and class (freshman, sophomore, and so on) for undergraduates.

# Modeling of UNION Types Using Categories

# Aggregation

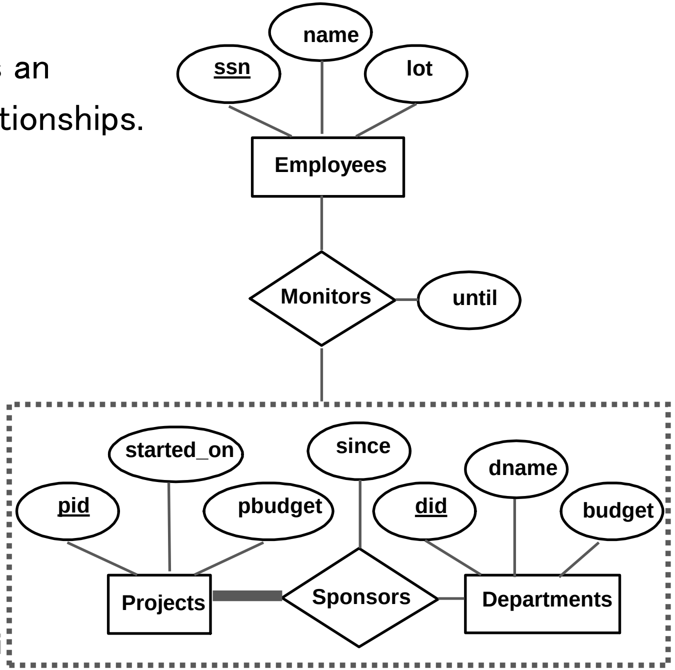
- Suppose we want to record managers for tasks performed by an employee at a branch



# Aggregation

- Represents a "has-a" or "is-part-of" relationship between entity types, where one represents the "whole" and the other the "part."
- Used when we have to model a relationship involving (entity sets and) a relationship set.
  - Aggregation allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.



# Summary of Conceptual Design

- Conceptual design follows requirements analysis,
  - Yields a high-level description of data to be stored
- ER model popular for conceptual design
  - Constructs are expressive, close to the way people think about their applications.
- Basic constructs: entities, relationships, and attributes (of entities and relationships).
- Some additional constructs: weak entities, ISA hierarchies, and aggregation.
- Note: There are many variations on ER model.

# Summary of ER (Contd.)

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies.
  - Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model. (e.g., z = x + y)
  - Constraints play an important role in determining the best database design for an enterprise.
- ER design is *subjective*. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, and whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.

# References

- Chapter 4 of FUNDAMENTALS OF Database Systems, SEVENTH EDITION
- Chapter 13 of Database Systems A Practical Approach to Design, Implementation, and Management, SIXth edition
- Chapter 6 of DATABASE SYSTEM CONCEPTS, S I XTH E D I T I ON.