COS221
L06 - Enhanced Entity-Relationship Modelling (Part 2)

(Chapter 8 in Edition 6 and Chapter 4 in Edition 7)

Linda Marshall

3 March 2023

Recap

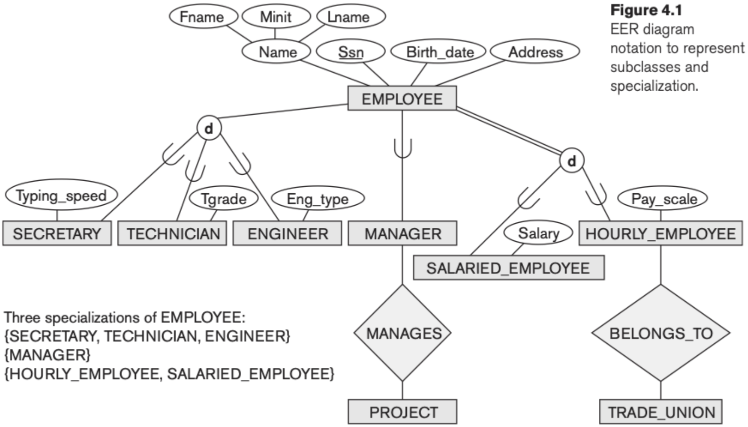


Figure 4.1
EER diagram notation to represent subclasses and specialization.

Definitions for EER concepts

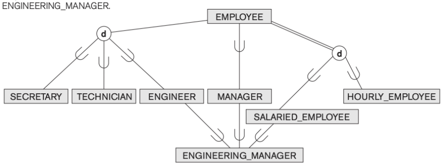
- A *class* is a set or collection of entities of a defined type.
- A *subclass* S is a class whose entities must always be a subset of the entities in another class, called the *superclass* C. S has an *is-a* relationship with C.
- A *specialisation* $Z = S_1, S_2, ..., S_n$ is a set of subclasses that have the same superclass G.
  - Z is said to be *total* if the superclass G is always equal to the union of all the subclasses. Otherwise it is *partial*.
  - Z is said to be *disjoint* if there is never an intersection between the subclasses. Otherwise it is *overlapping*.

Specialisation and Generalisation - Hierarchies and Lattices

A subclass may have further subclasses. This nesting will either result in a tree-like (hierarchical) or graph-based (lattice) structure.

- In a hierarchical structure, each subclass only has one parent.
- In a lattice structure, each subclass can participate in more than one class/subclass relationship.

Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



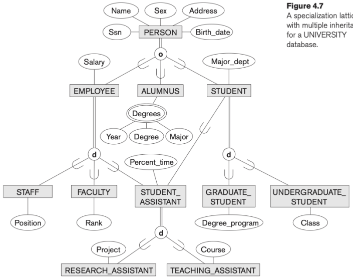


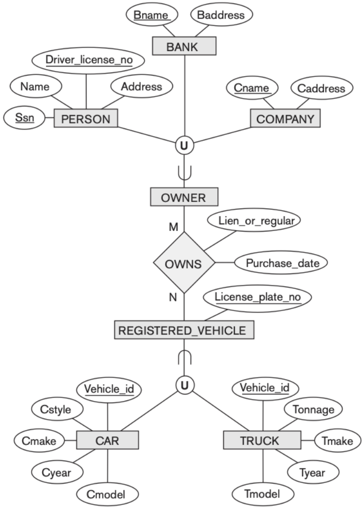
Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

Modelling UNION Types

- Up till now we have considered entities with a single superclass. For example the EMPLOYEE entity type.
- Sometimes a superclass/subclass relationship requires more than one superclass.
- Some tools cannot model UNION types and therefore a work-a-round needs to be defined. This will be discussed in a later chapter.

Modelling UNION Types

- The subclass represents a collection of objects formed by the UNION of distinct entity types.
- The subclass is referred to as the *union type* (or *category*).
- A UNION is denoted in EER-diagrams with by placing a **U** in the circle.



Modelling UNION Types

- OWNER is a union type (or category) of the PERSON or BANK or COMPANY entity types. REGISTERED_VEHICLE is a subclass of CAR or TRUCK.
- An entity that is a member of OWNER, must exist in only one of its superclasses. That is, an OWNER may be a PERSON, BANK or COMPANY.
- Attribute inheritance works selectively for union types. The subclass inherits attributes from one of its superclasses. This means that a superclass of a category may hold different keys. For example, OWNER with PERSON, BANK or COMPANY holds different keys.

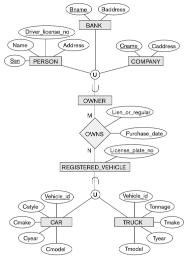


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

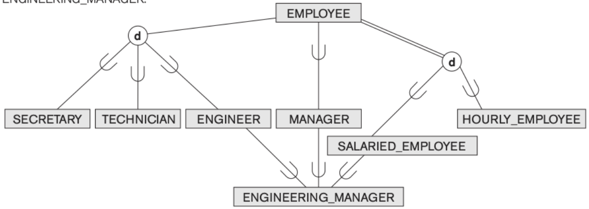
Modelling UNION Types

- A union type may be *partial* or *total*.
    - A *partial* category holds a subset of the union.
    - A *total* category holds the UNION of all entities in its superclass. A double line connecting the category and the circle is used to differentiate the total category.

Modelling UNION Types

Constrast a UNION with the engineering manager. An ENGINEERING_MANAGER must be an ENGINEER and a MANAGER and a SALARIED_EMPLOYEE.

Figure 4.6
A specialization lattice with shared subclass ENGINEERING_MANAGER.



EER Design Choices

- Specialisation and subclasses make the model more *accurate* but increase *clutter*.
- A subclass with **few local attributes** and **no relationship types** can be *merged into superclass*. Could then use ‘Type’ attribute and ‘NULL’ values for the non-members of the merged subclass.
- If **all subclasses** meet the requirements above, they can similarly be merged into their superclass.
- **Avoid UNION types** and categories as much as possible. Specialisation/generalisation usually sufficient.
- If the **requirements** do not indicate any particular constraints, the default would generally be *overlapping and partial*, since this does not specify any restrictions on subclass membership.

University Example - Requirements

1. 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.
2. 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.
3. 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.
4. 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.

University Example - Notations

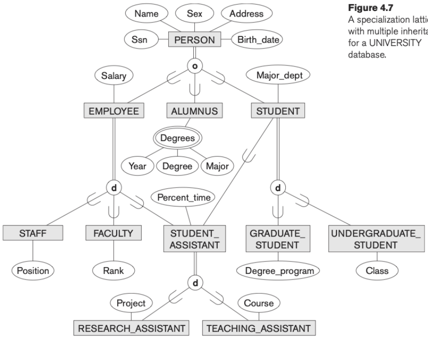


Figure 4.7
A specialization lattice with multiple inheritance for a UNIVERSITY database.

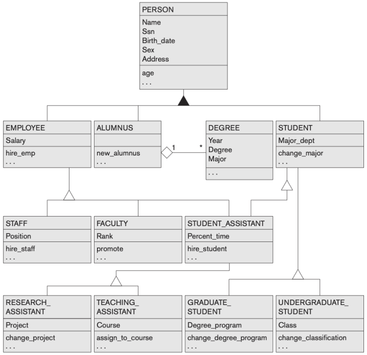


Figure 4.10
A UML class diagram corresponding to the EER diagram in Figure 4.7, illustrating UML notation for specialization/generalization.