

Enhanced Entity-Relationship Models

Enhanced Entity-Relationship Models

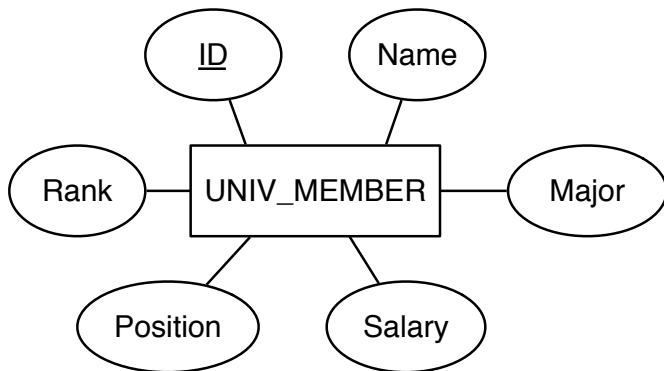
Enhanced ER models add inheritance to entity types – superclasses and subclasses. Subclasses inherit all the attributes of their superclass(es).

Subclasses can:

- ▶ be **disjoint** or **overlapping**;
- ▶ represent **total** or **partial** specialization.

University Member

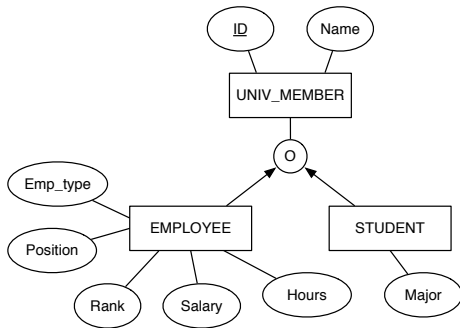
Consider this model of a university member:



- ▶ Do faculty members have a major?
- ▶ Do students have a rank or salary?

Employee and Student Subclasses

Here we break the UNIV_MEMBER into two subtypes:



Notice that the subclasses are overlapping, as denoted by the O in the circle under **UNIV_MEMBER**. That means that an entity can be both a **STUDENT** and an **EMPLOYEE**.

But there's more:

- ▶ What's that **Emp_type** attribute?
- ▶ Can an employee have a salary and a number of hours?

Faculty and Staff Subclasses

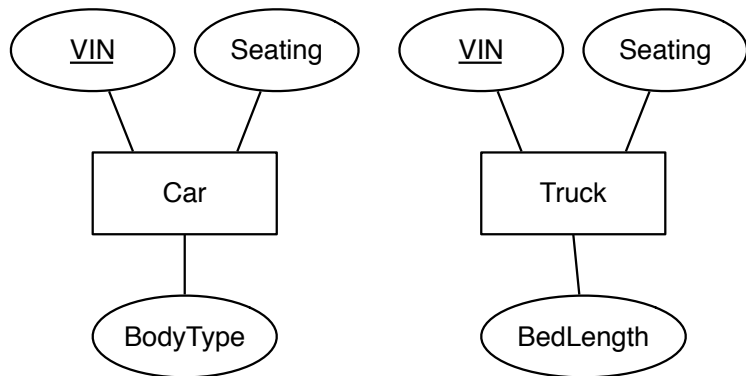
- ▶ The subclasses of EMPLOYEE are disjoint (D in the circle) – an employee is either SALARIED or HOURLY but not both.
- ▶ The double line from employee means **total specialization** – there are no EMPLOYEE instances, only SALARIED or HOURLY.

The process of going from an entity type that has attributes that don't apply to all entity instances to subclasses that are specialized but inherit the common attributes is called **specialization** (top-down).^a

^aWe're not quite done, but we'll return to this example.

Cars and Trucks

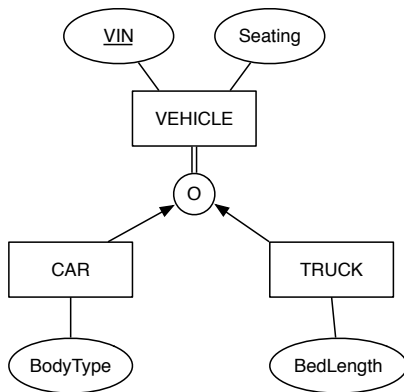
Let's consider another model.



Notice the common attributes. We can factor those into a superclass.

Vehicles

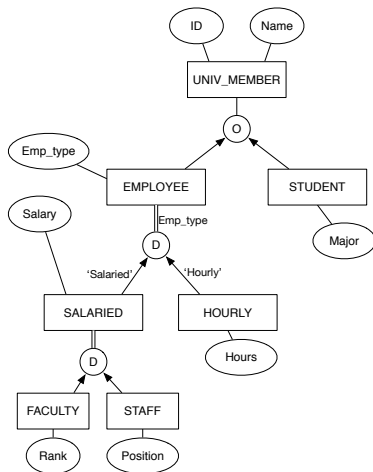
The common attributes are in the superclass and the specialized attributes are in the subclass.



The process of factoring the common attributes of two or more entity types and creating superclasses to hold the common attributes is called **generalization** (bottom-up).

Student Employees

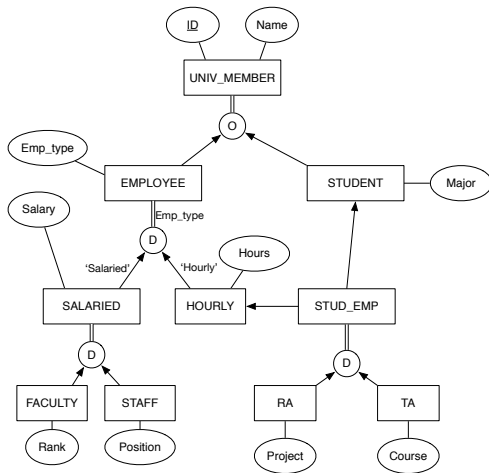
Remember our STUDENT - EMPLOYEE hierarchy:



What about STUDENTS who are also HOURLY employees, like TAs and research assistants?

Multiple Inheritance

Our previous EER models have been inheritance trees – each subclass had only one superclass. We can create lattices with multiple inheritance:



Subclasses provide for more granular and precise modeling of entity types. Use subclasses when:

- ▶ you want to represent commonality and differences between different entity types,
- ▶ you want to have different subsets of an entity type participate in different relationships (e.g., faculty participate in TEACHES relationships, HOURLY employees don't).