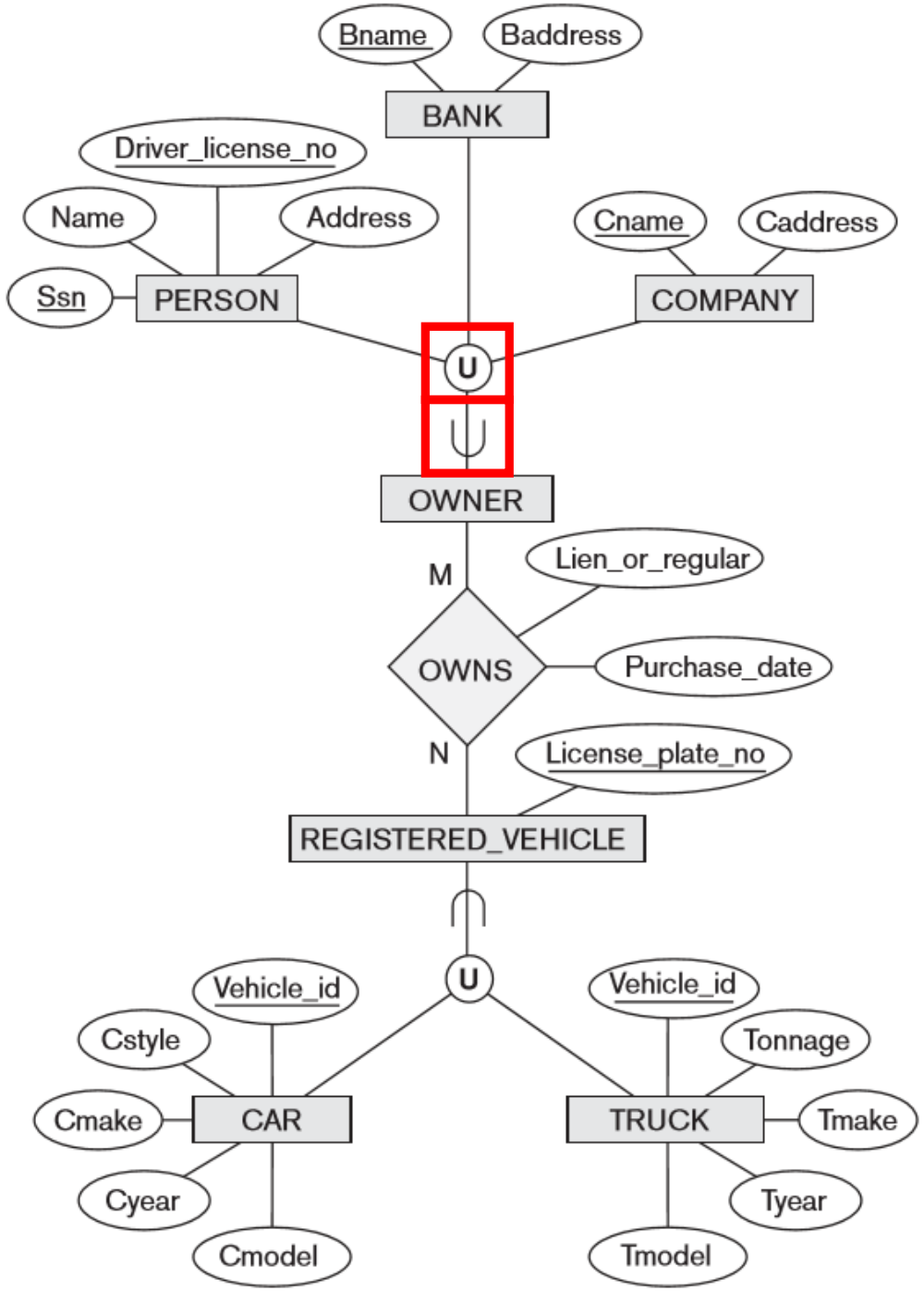
The Enhanced Entity-Relationship (EER) Model

Part 2

Modeling of UNION Types Using Categories

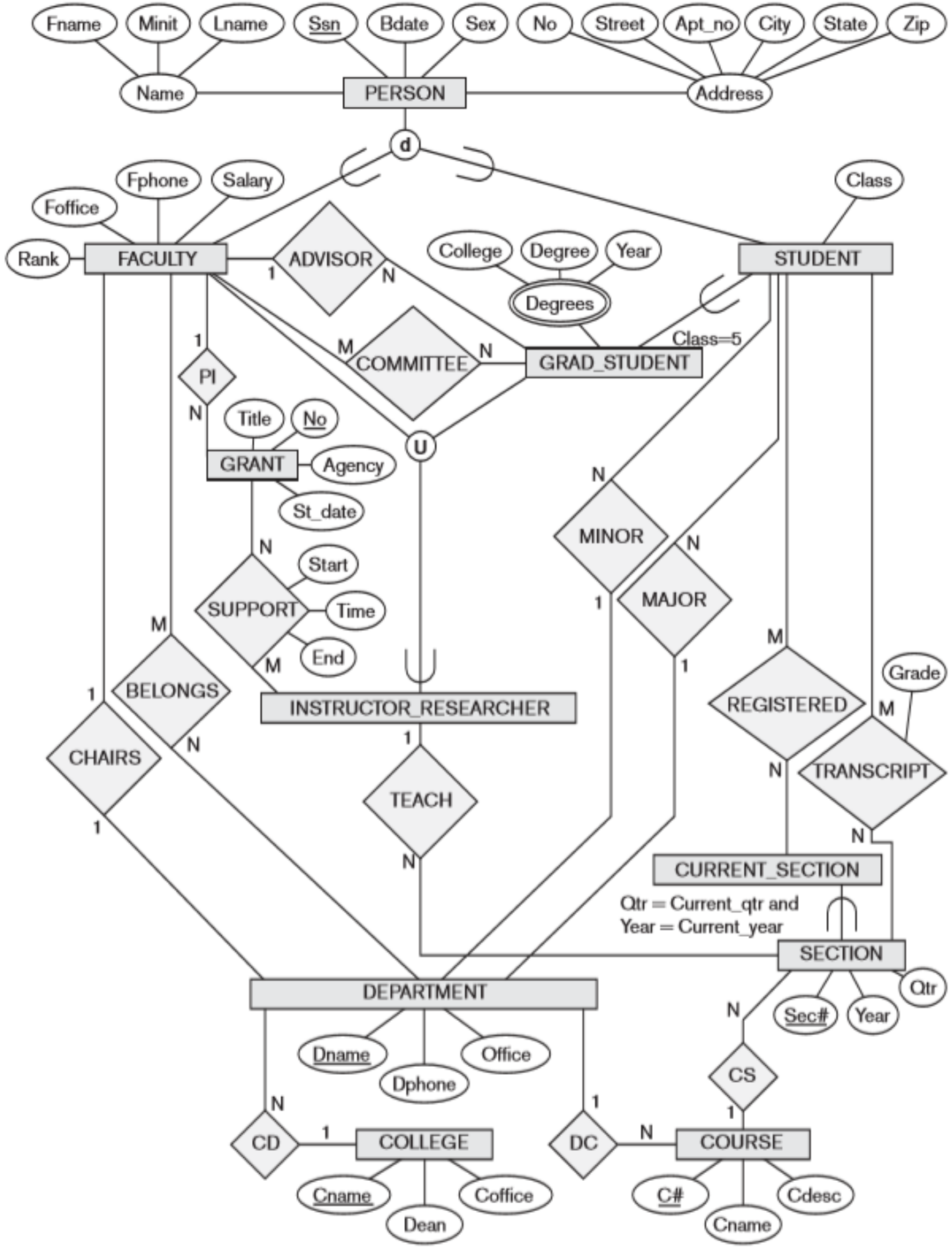
- All of the superclass/subclass relationships we have seen thus far have a *single superclass*.
- A shared subclass such as `ENGINEERING_MANAGER` is the subclass in three *distinct* superclass/subclass relationships, where each of the three relationships has a *single* superclass.
- However, it is sometimes necessary to represent a single superclass/subclass relationship with *more than one* superclass, where the superclasses represent different entity types.
- In this case, the subclass will represent a collection of objects that is a subset of the `UNION` of distinct entity types; we call such a subclass a **union type** or a **category**.


Bname
Baddress
BANK
Driver_license_no
Name
Address
Ssn
PERSON
Cname
Caddress
COMPANY
U
OWNER
M
Lien_or_regular
OWNS
Purchase_date
N
License_plate_no
REGISTERED_VEHICLE
U
Vehicle_id
Cstyle
Cmake
CAR
Cyear
Cmodel
Vehicle_id
Tonnage
TRUCK
Tmake
Tyear
Tmodel

- Attribute inheritance works more selectively in the case of categories.
- For example, each OWNER entity inherits the attributes of a COMPANY, a PERSON, or a BANK, depending on the superclass to which the entity belongs.
- On the other hand, a shared subclass such as ENGINEERING_MANAGER inherits all the attributes of its superclasses SALARIED_EMPLOYEE, ENGINEER, and MANAGER.

A Sample UNIVERSITY EER Schema and Design Choices

The UNIVERSITY Database Example


Fname
Minit
Lname
Ssn
Bdate
Sex
No
Street
Apt_no
City
State
Zip
Name
PERSON
Address
d
Fphone
Salary
Foffice
Rank
FACULTY
ADVISOR
College
Degree
Year
Degrees
STUDENT
Class
Class=5
GRAD_STUDENT
COMMITTEE
PI
Title
No
GRANT
Agency
St_date
U
MINOR
MAJOR
Start
SUPPORT
Time
End
BELONGS
CHAIRS
INSTRUCTOR_RESEARCHER
TEACH
REGISTERED
Grade
TRANSCRIPT
CURRENT_SECTION
Qtr = Current_qtr and
Year = Current_year
SECTION
Qtr
Sec#
Year
DEPARTMENT
Dname
Dphone
Office
CS
CD
COLLEGE
Cname
Coffice
Dean
DC
COURSE
C#
Cdesc
Cname

Design Choices for Specialization/Generalization

- In general, many specializations and subclasses can be defined to make the conceptual model accurate.
- However, the drawback is that the design becomes quite cluttered.
- It is important to represent only those subclasses that are deemed necessary to avoid extreme cluttering of the conceptual schema.

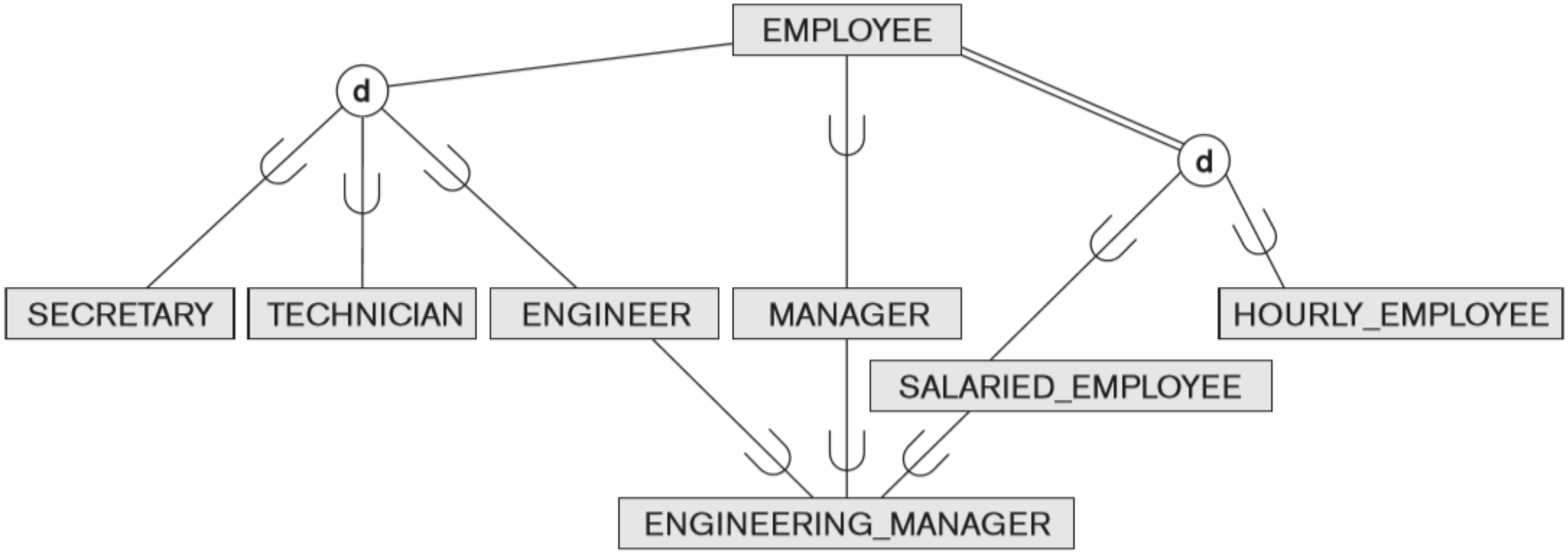
- If a subclass has few specific (local) attributes and no specific relationships, it can be merged into the superclass.
- The specific attributes would hold NULL values for entities that are not members of the subclass.
- A *type* attribute could specify whether an entity is a member of the subclass.

- Similarly, if all the subclasses of a specialization/generalization have few specific attributes and no specific relationships, they can be merged into the superclass and replaced with one or more *type* attributes that specify the subclass or subclasses that each entity belongs to.

- Union types and categories should generally be avoided unless the situation definitely warrants this type of construct, which does occur in some practical situations.

- The choice of disjoint/overlapping and total/partial constraints on specialization/generalization is driven by the rules in the miniworld being modeled.
- 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.

- As an example of applying these guidelines, consider the following figure, where no specific (local) attributes are shown.



- We could merge all the subclasses into the EMPLOYEE entity type, and add the following attributes to EMPLOYEE:
  - An attribute Job_type whose value set {'Secretary', 'Engineer', 'Technician'} would indicate which subclass in the first specialization each employee belongs to.
  - An attribute Pay_method whose value set {'Salaried', 'Hourly'} would indicate which subclass in the second specialization each employee belongs to.
  - An attribute Is_a_manager whose value set {'Yes', 'No'} would indicate whether an individual employee entity is a manager or not.