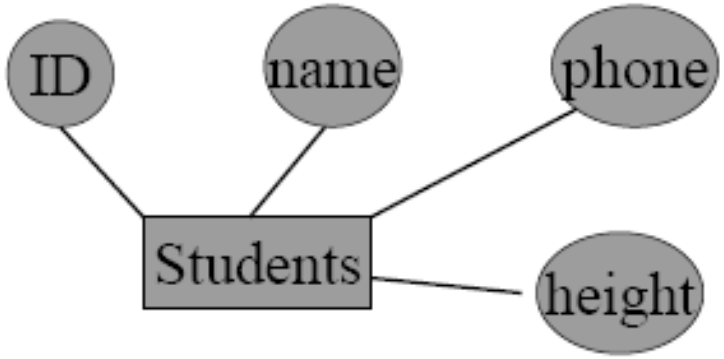
Lecture 3
Review of E-R Model & The Enhanced Entity-Relationship Model

Conceptual Design

- What are the *entities* and *relationships* in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the *constraints* that hold?
- Represent this information pictorially in *ER diagram*, then map ER diagram into a relational schema

Entities and Attributes

- Entity: real world “object”
- Entity set: set of “similar” entities/objects
- Attributes: properties typically used as column names, similar to fields of a struct



Relationships

- Relationship set: set of “similar” relationships
- Connect two or more entity sets
- Represented by diamonds

Structural Constraints

- Cardinality ratio (of a binary relationship): 1:1, 1:N, N:1, or M:N
  - Placing appropriate number/letter on the link
- Participation constraint
  - Total (existence dependency)
    - Double lining the link
  - Partial

Design Issues

- Setting: client has (possibly vague) ideas of what they want. You must design a database that represent these thoughts and only these thoughts
- Avoid redundancy
  - Redundancy = saying the same thing more than once
  - Wastes space and encourages inconsistency

In design phase, is it a good idea to keep the total number of entity sets and relationship sets as small as possible?

Yes

No

Total Results: 0

In design phase, is it a good idea to keep the total number of entity sets and relationship sets as small as possible?

Yes

No

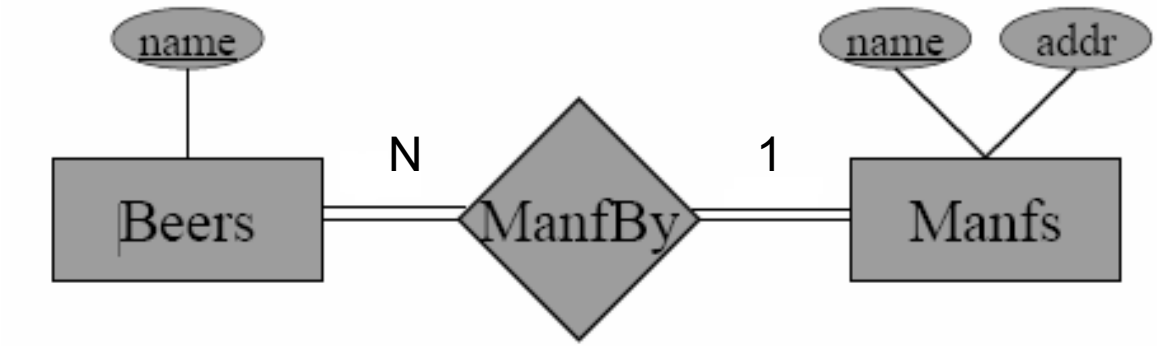
In design phase, is it a good idea to keep the total number of entity sets and relationship sets as small as possible?

Yes

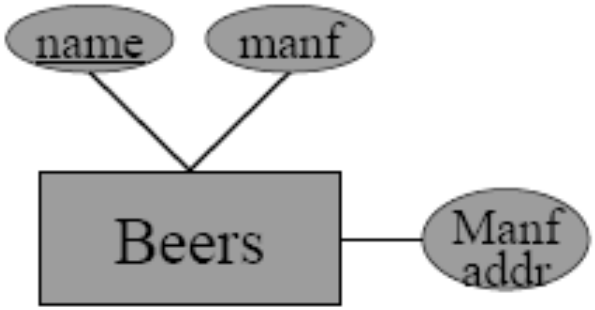
No

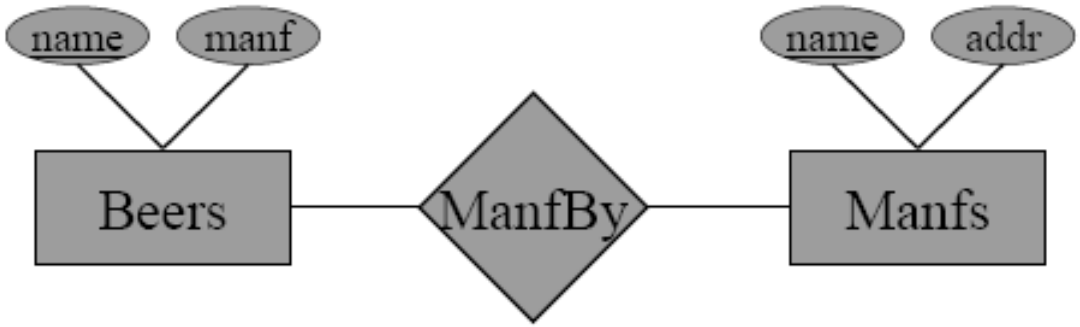
Example

- Good



- Bad



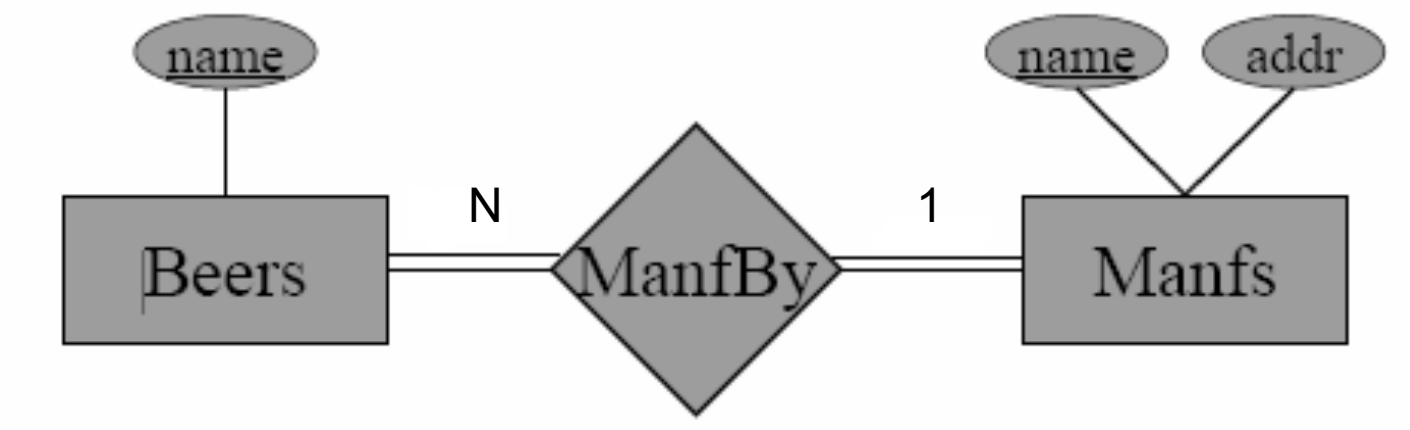


Intuitive Rules for Entity Sets vs. Attributes

- Make an entity set only if it is either:

1. More than a name of something; i.e., it has nonkey attributes or relationships with a number of different entity sets, or
2. The “many” in a many-one relationship

Example



- *Manfs* deserves to be an E.S. because we record *addr*, a nonkey attribute
- *Beers* deserves to be an E.S. because it is the “many” end

Don’t Overuse Weak Entity Sets

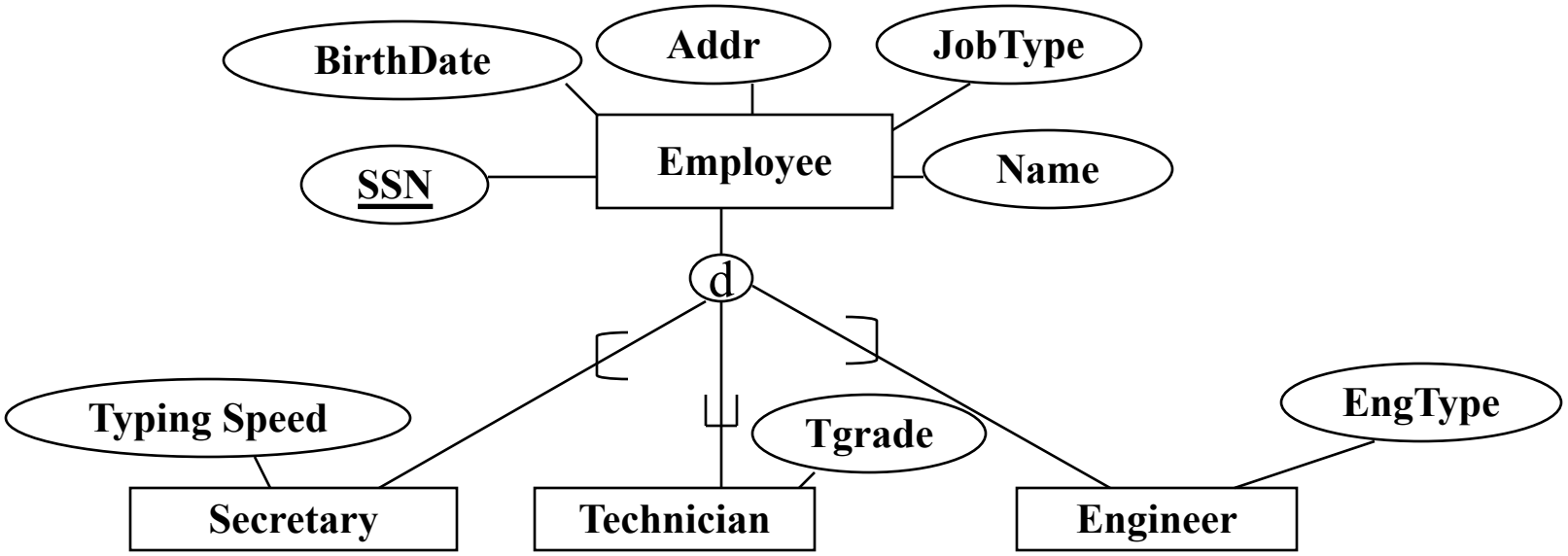
- There is a tendency to feel that no entity set has its entities uniquely determined without following some relationships
- However, in practice, we almost always create unique ID’s to compensate: social-security numbers, VIN’s, etc.

EER

- EER=ER + subclass/superclass (specialization/generalization)
- Specialization: Defines a set of subclasses of an entity set—superclass
- Generalization: Result of computing the union of two or more entity sets to produce a higher-level entity set

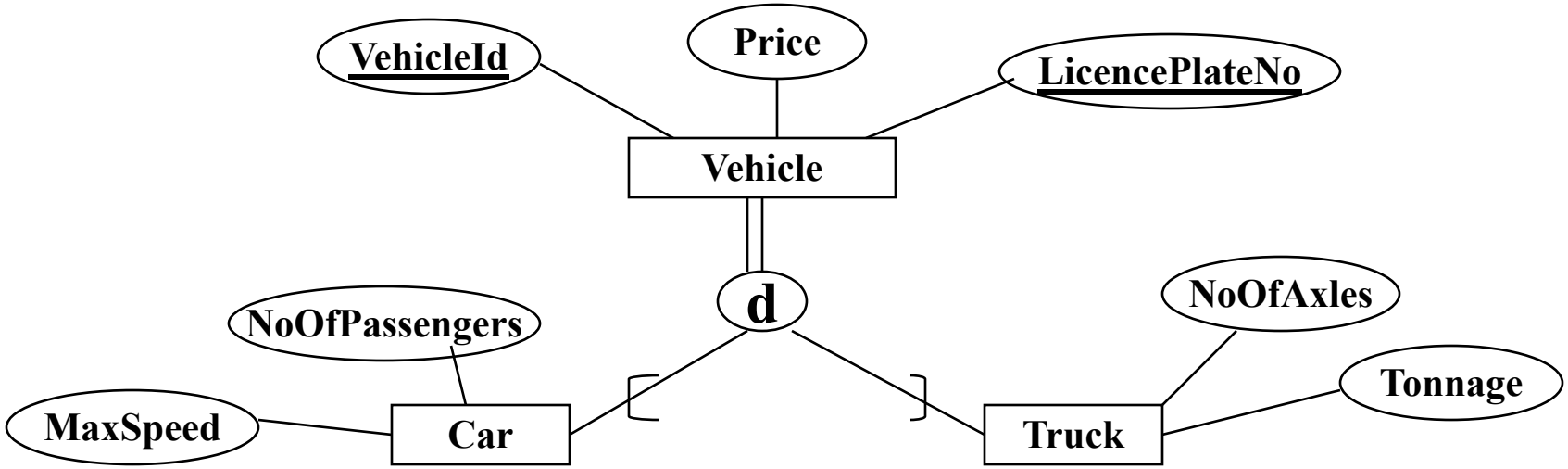
EER Example 1

- Employee (superclass) entity set can be grouped into secretary, engineer, technician,…(subclass)



EER Example 2

- Vehicle (superclass) entity set can be categorized into car, truck, … (subclass)



Inheritance

- An entity cannot merely exist by being a member of a subclass but no superclass; however, it is not necessary that every entity in a superclass be a member of some subclass
- Inheritance: An entity of subclass
  - Inherits all attributes of superclass
  - Inherits all relationships in which superclass participates
  - Plus its own specific attributes and relationships

Does an instance of a subclass automatically belong to its superclass?

- Yes and vice versa
- No
- Yes but not vice versa

Total Results: 0

Does an instance of a subclass automatically belong to its superclass?

- Yes and vice versa
- No
- Yes but not vice versa

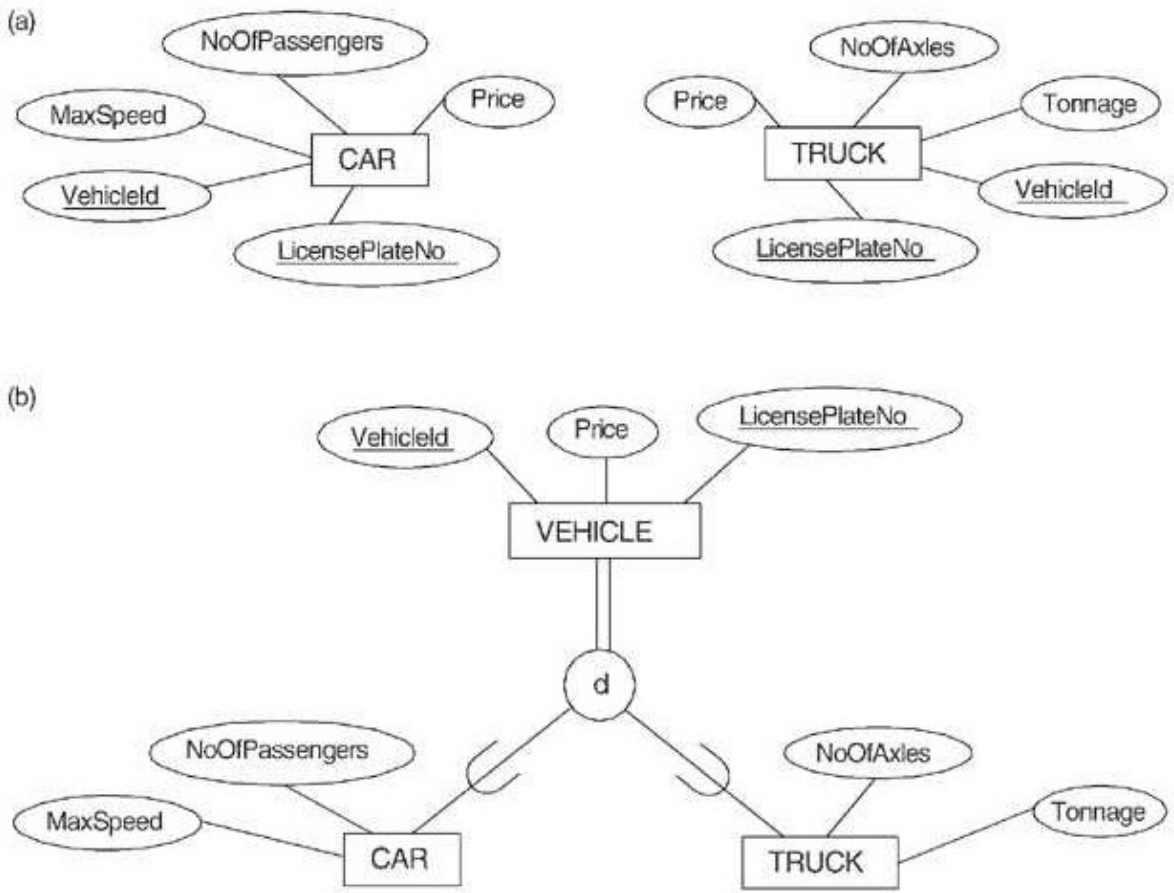
Does an instance of a subclass automatically belong to its superclass?

- Yes and vice versa
- No
- Yes but not vice versa

Specialization

- Why specialization
  - Define a set of subclasses of an entity set
  - Associate additional specific attributes with each subclass
  - Establish additional specific relationship sets between each subclass and other entity sets
- Types of specialization
  - Predicate-defined: Entities will become members of a subclass by satisfying explicit condition or predicate
    - Can automatically handled by the system
  - User defined

Generalization



Constraints of Specialization and Generalization

- Disjointness
  - Disjoint: An entity can be a member of at most one of the subclasses
  - Overlap: When subclasses are not disjoint
- Completeness
  - Total: Every entity in superclass must be a member of some subclass
  - Partial: An entity might not belong to any subclass
- 4 Possible constraints on specialization
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial

For attribute-defined specialization, if the defining attribute is single-valued, what does that imply?

- Disjoint
- Overlap

Total Results: 0

For attribute-defined specialization, if the defining attribute is single-valued, what does that imply?

Disjoint

Overlap

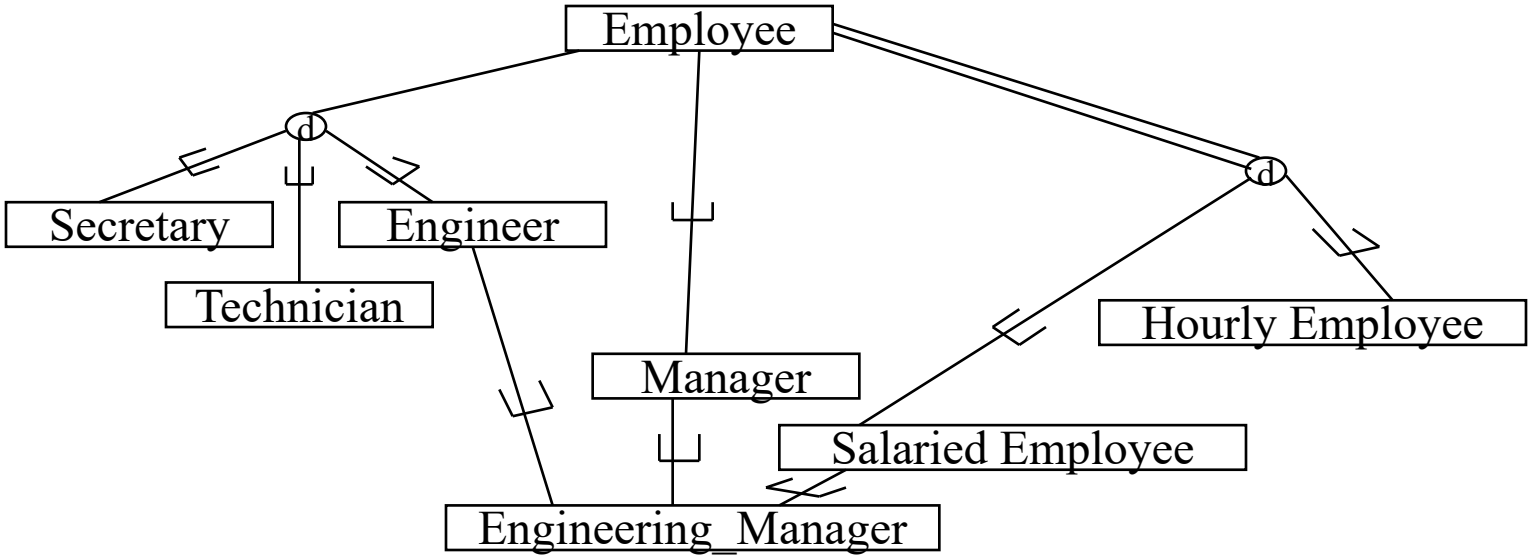
For attribute-defined specialization, if the defining attribute is single-valued, what does that imply?

Disjoint

Overlap

Hierarchies and Lattices

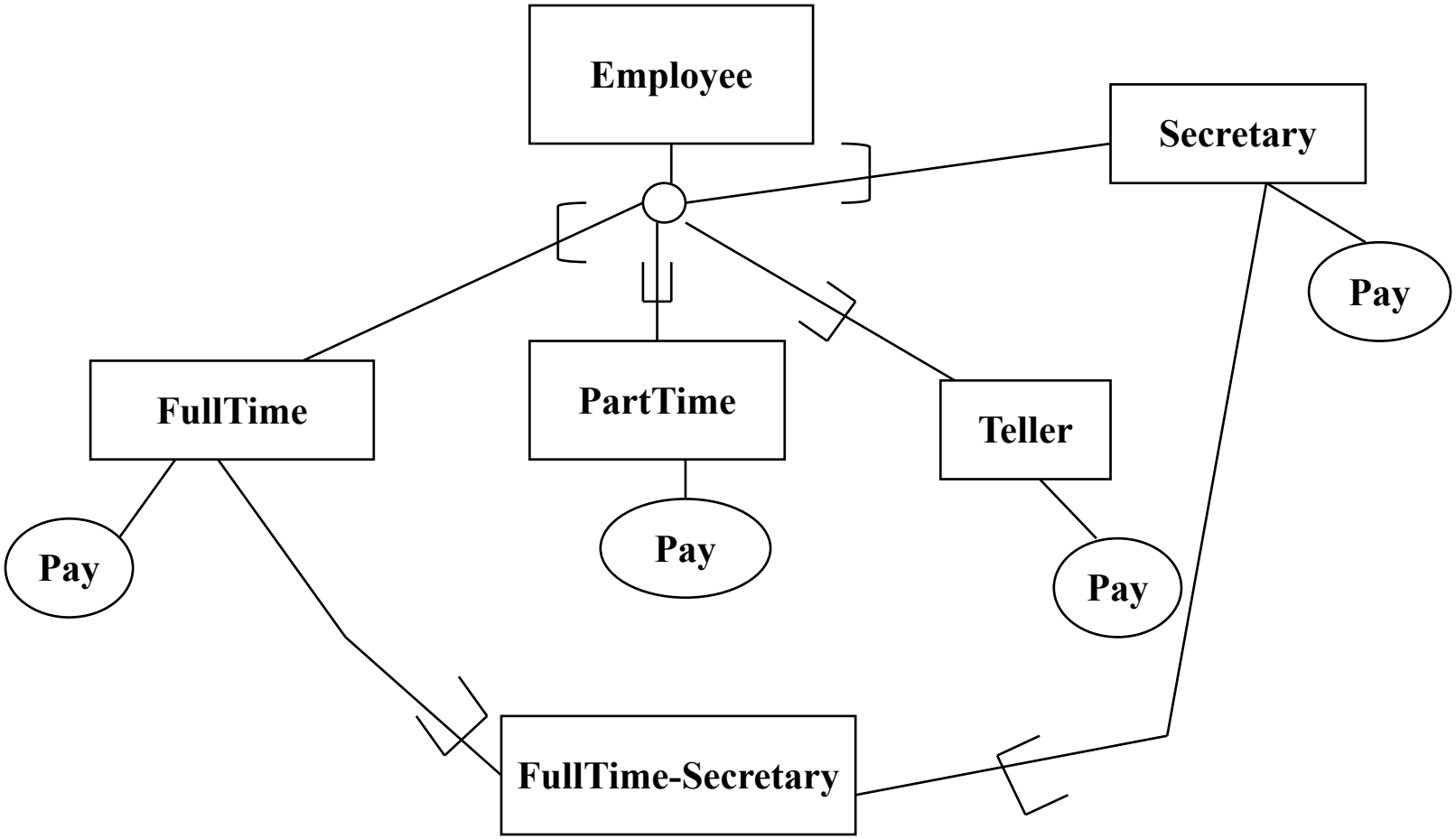
- Tree-structured (hierarchy): All subclasses have only one parent
- Graph-structured (lattice): A subclass may have multiple superclasses
  - Engineering_Manager inherits attributes and relationships from multiple superclasses



Multiple Inheritance

- Potential ambiguity: If same attributes can be inherited from more than one superclass
- Bank employee: Instead of defining attribute salary for superclass employee, we define attribute pay for each of full-time, part-time, teller, and secretary as follows:
  - Full-time: pay (0-100k)
  - Part-time: pay (0-30/hr)
  - Teller: pay (0-30k)
  - Secretary: pay (0-35k)

Bank Employee Example



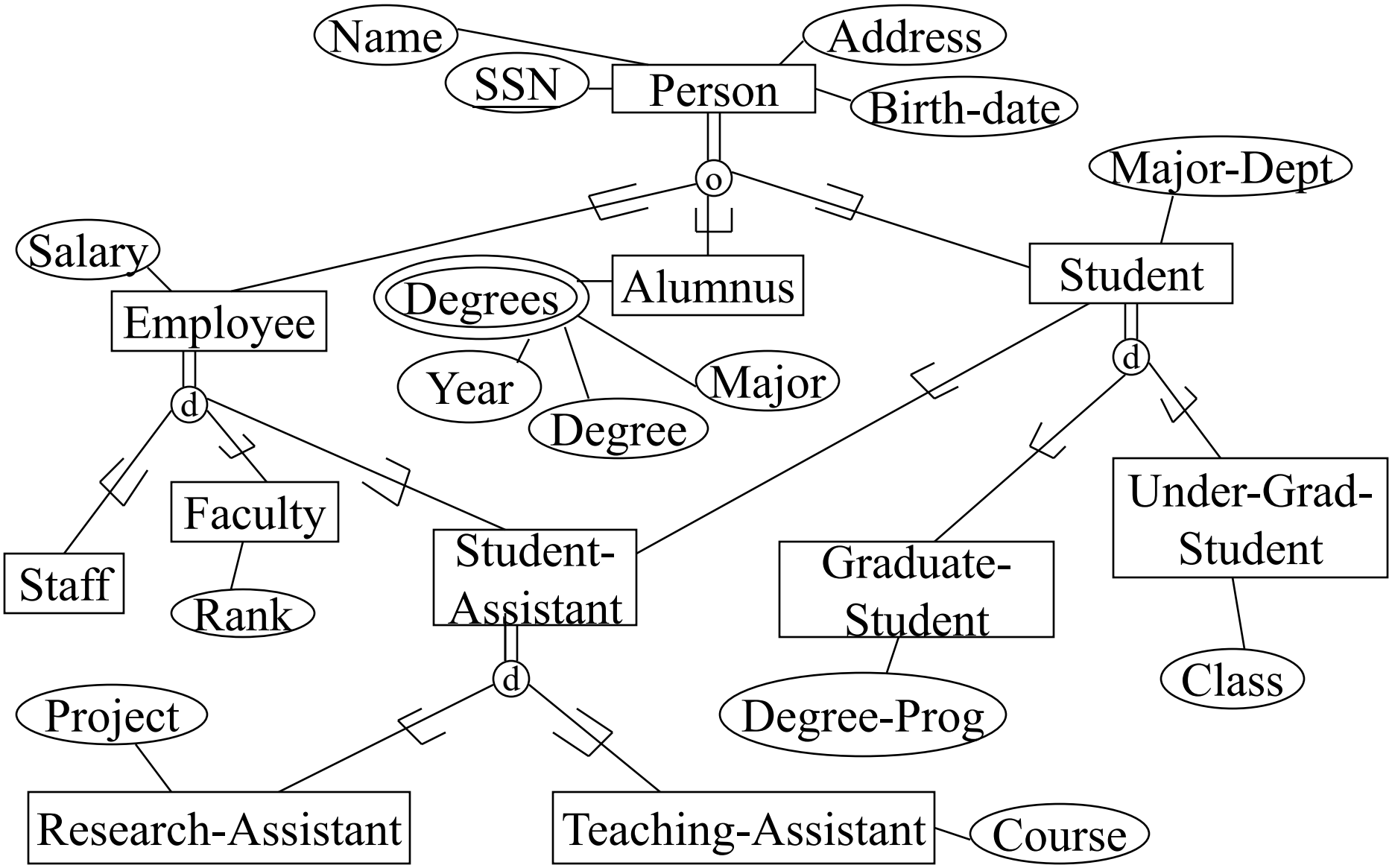
Solutions

- Keep all and use alias: fulltime.pay and secretary.pay
- Choose one based on implementation
- Force user to choose
- Error report

No Ambiguity

- If salary attribute is retained in only one superclass Employee – all subclasses share the same definition of salary
- An attribute originating in the same superclass is inherited more than once via different paths
  - The attribute should be included only once in subclasses

EER


Name
Address
SSN
Person
Birth-date
o
Major-Dept
Salary
Alumnus
Degrees
Student
Employee
d
Year
Major
Degree
d
Faculty
Under-Grad-
Student
Staff
Student-
Assistant
Graduate-
Student
Rank
Class
d
Project
Degree-Prog
Research-Assistant
Teaching-Assistant
Course

Exercise

Suppose it is necessary to keep track of

- different types of ACCOUNTS(SAVINGS_ACCTS, CHECKING_ACCTS,…)
- different types of LOANS(CAR_LOANS, HOME_LOANS,…)
- each account's TRANSACTIONS(deposits, withdrawals, checks,…)
- each loan's PAYMENTS
- include the amount, date, time,… for PAYMENTS and TRANSACTIONS

Modify the BANK schema on the right side