Entity-Relationship [ER] Model & Enhanced ER [EER]

Example COMPANY Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
    - The company is organized into DEPARTMENTs.
        - Each department has a name, number and an employee who *manages* the department.
        - We keep track of the start date of the department manager.
        - A department may have several locations.
    - Each department *controls* a number of PROJECTs.
        - Each project has a unique name, unique number and is located at a single location.

Example COMPANY Database

- We store each EMPLOYEE’s social security number, address, salary, sex, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
    - We keep track of the number of hours per week that an employee currently works on each project.
    - We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Initial Design – COMPANY

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
    - DEPARTMENT
    - PROJECT
    - EMPLOYEE
    - DEPENDENT

Initial Design – COMPANY

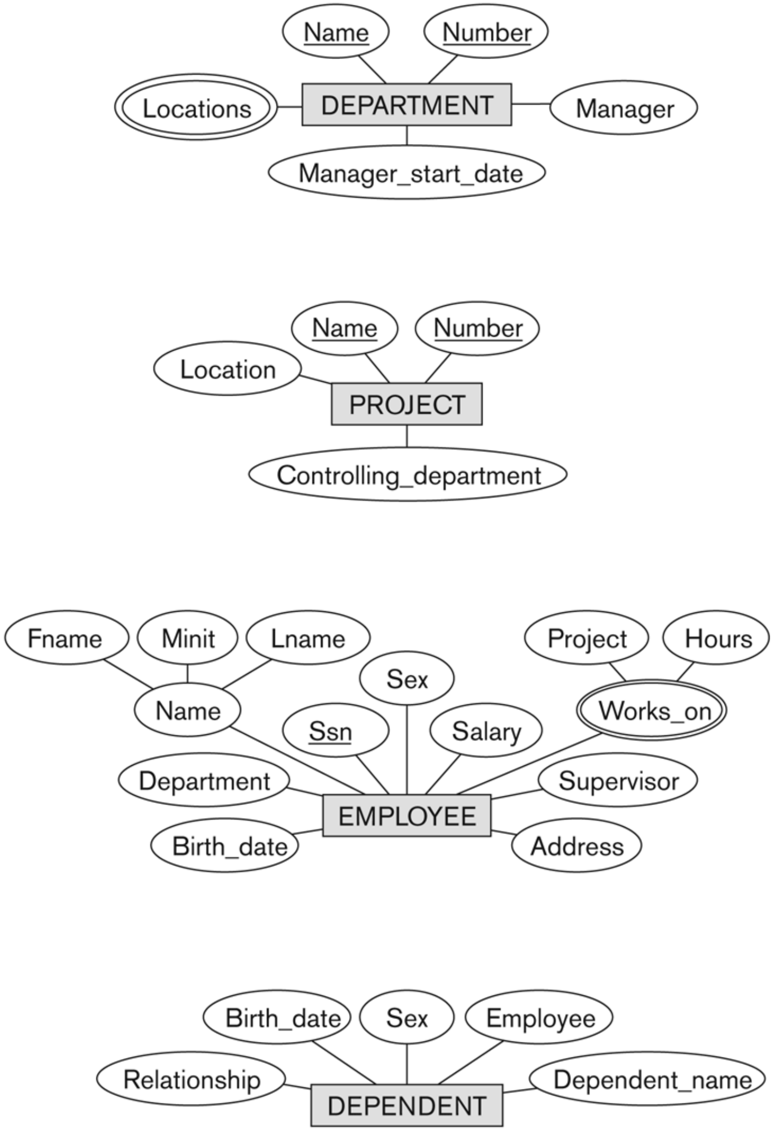


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Refining COMPANY – Relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships (degree 2)
- Listed below with their participating entity types:
  - WORKS_FOR (between EMPLOYEE, DEPARTMENT)
  - MANAGES (also between EMPLOYEE, DEPARTMENT)
  - CONTROLS (between DEPARTMENT, PROJECT)
  - WORKS_ON (between EMPLOYEE, PROJECT)
  - SUPERVISION (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
  - DEPENDENTS_OF (between EMPLOYEE, DEPENDENT)

Refining COMPANY – Relationships

- Draw the ER diagram by connecting the entity types with identified relationship types

Weak Entity Type

- An entity that does not have a key attribute
- A weak entity must participate in an identifying relationship type with an owner or identifying entity type
- **Example:**
  - A DEPENDENT entity is identified by the dependent's first name, *and* the specific EMPLOYEE with whom the dependent is related
  - Name of DEPENDENT is the *partial key*
  - DEPENDENT is a *weak entity type*
  - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Refining COMPANY – Weak Entity

- Weak entity types can be represented by as complex(composite, multi-valued) attributes
- Choose weak entity type if:
  - there are many attributes
  - the weak entity participates independently in relationship types other than its identifying relationship type
- Draw the ER diagram by connecting the weak entity type with owner entity type via the identifying relationship type

Constraints on Relationships

- Relationship instance – relates individual participating entities – one from each participating entity type
- The degree of a relationship type is the number of participating entity types
- Relationship type of degree two – binary, degree three – ternary, degree >3 – n-ary
- Constraints – cardinality ratio and participation constraint
- Cardinality ratio – 1:1, 1:N, N:1, M:N

Refining COMPANY – Cardinality Ratio

- Cardinality ratio shown by placing appropriate numbers on the relationship edges
- Identify the cardinality ratio of each relationship in the COMPANY database

Refining COMPANY – Participation

- Identify the participation constraint for each of the relationships in the COMPANY database
- Draw the Total participation by double line and Partial participation by single line
- If some cardinality ratio or dependency cannot be determined from the requirements, the users must be questioned further to determine these structural constraints

Recursive Relationship Type

- *Same* entity type participates more than once in a relationship type in *different roles*
  - Example: the SUPERVISION relationship
  - EMPLOYEE participates twice in two distinct roles:
    - supervisor (or boss) role
    - supervisee (or subordinate) role
  - Each relationship instance relates two distinct EMPLOYEE entities:
    - One employee in *supervisor* role
    - One employee in *supervisee* role

Recursive Relationship Type

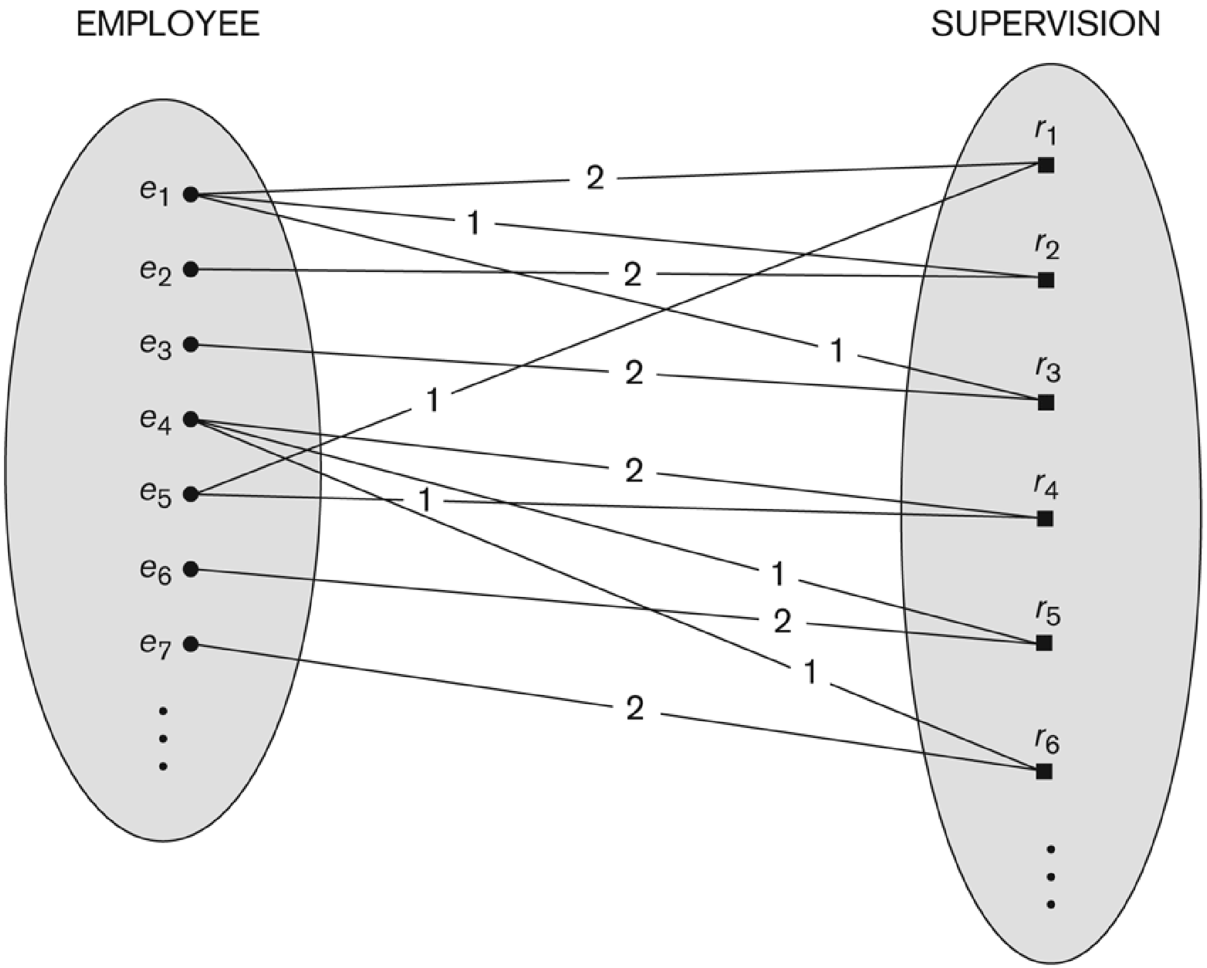
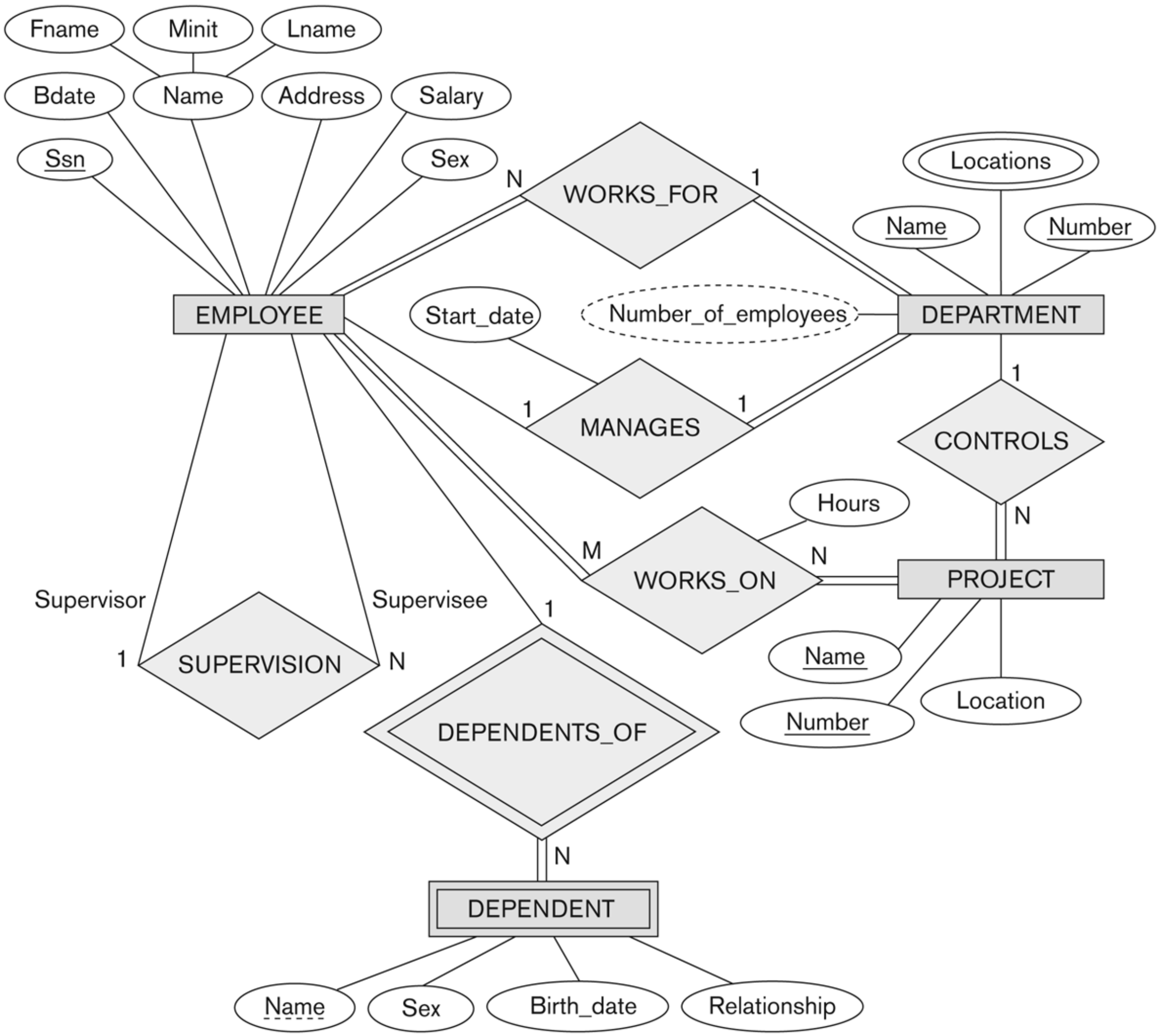


Figure 3.11
A recursive relationship SUPERVISION between EMPLOYEE in the *supervisor* role (1) and EMPLOYEE in the *subordinate* role (2).

Refining COMPANY – Recursive

- Mention the role of employee entity in recursive relationship type in COMPANY database

Refined COMPANY database



Relationship Attributes

- A relationship type can have atributes
- Attributes of 1:1 or 1:N relationship types can be migrated to one of the participating entity types
- start_date atrribute for MANAGES can be an attribute of either EMPLOYEE or DEPARTMENT
- For a 1:N, a relationship attribute can be migrated only to the N-side of the relationship
- For M:N, some attributes may be determined not by any single entity

Relationship Attributes

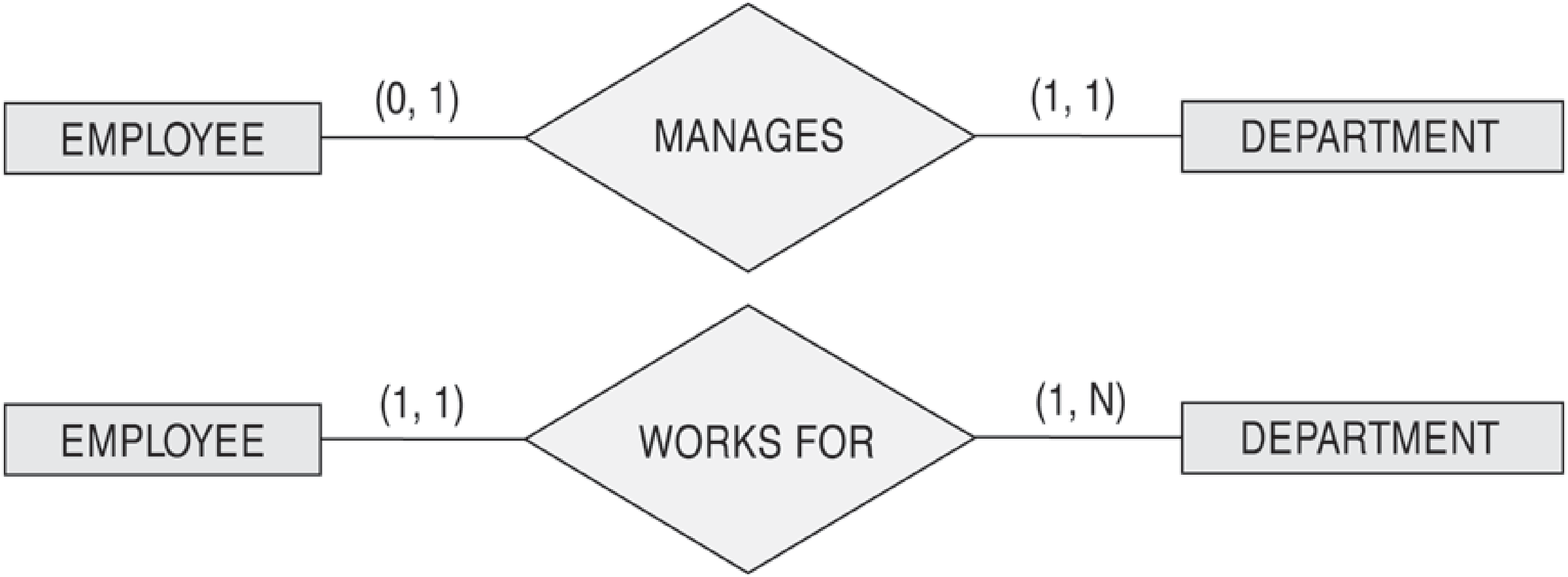
- Decision as to where a relationship attribute should be placed

  – is determined subjectively by the schema designer

(min,max) for Structural Constraints

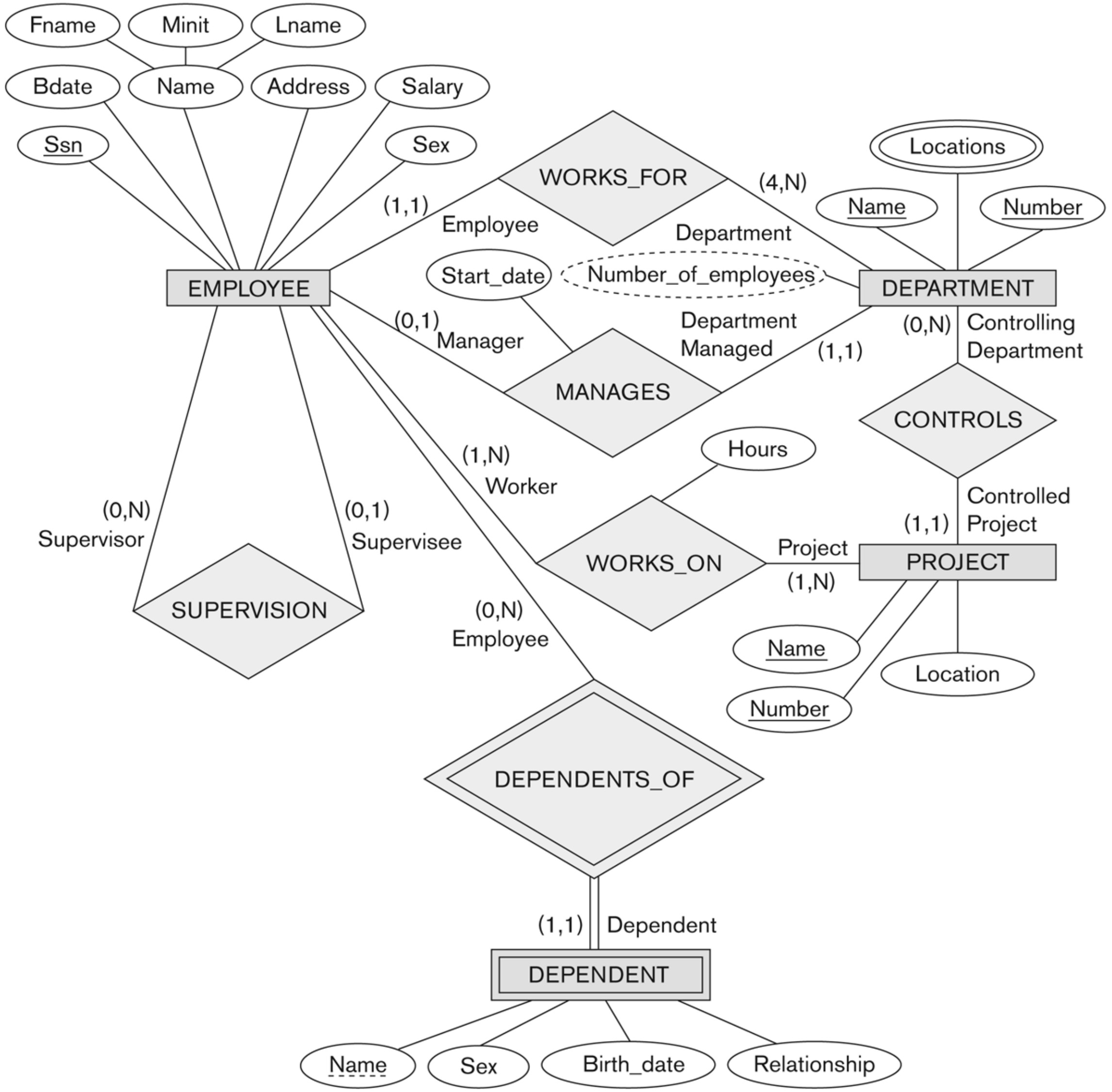
- Specifies that each entity e in E participates in at least *min* and at most *max* relationship instances in R
- Default(no constraint): min=0, max=n (signifying no limit)
- Must have min£max, min³0, max ³1
- Derived from the knowledge of mini-world constraints
- Examples:
  - A department has exactly one manager and an employee can manage at most one department.
    - Specify (1,1) for participation of DEPARTMENT in MANAGES
    - Specify (0,1) for participation of EMPLOYEE in MANAGES
  - An employee can work for exactly one department but a department can have any number of employees.

(min,max) for Structural Constraints



- Similary find the (min,max) for the *controls, works_on, dependents, supervision* relationships

Refined COMPANY – using (min,max)



Design Choices for ER

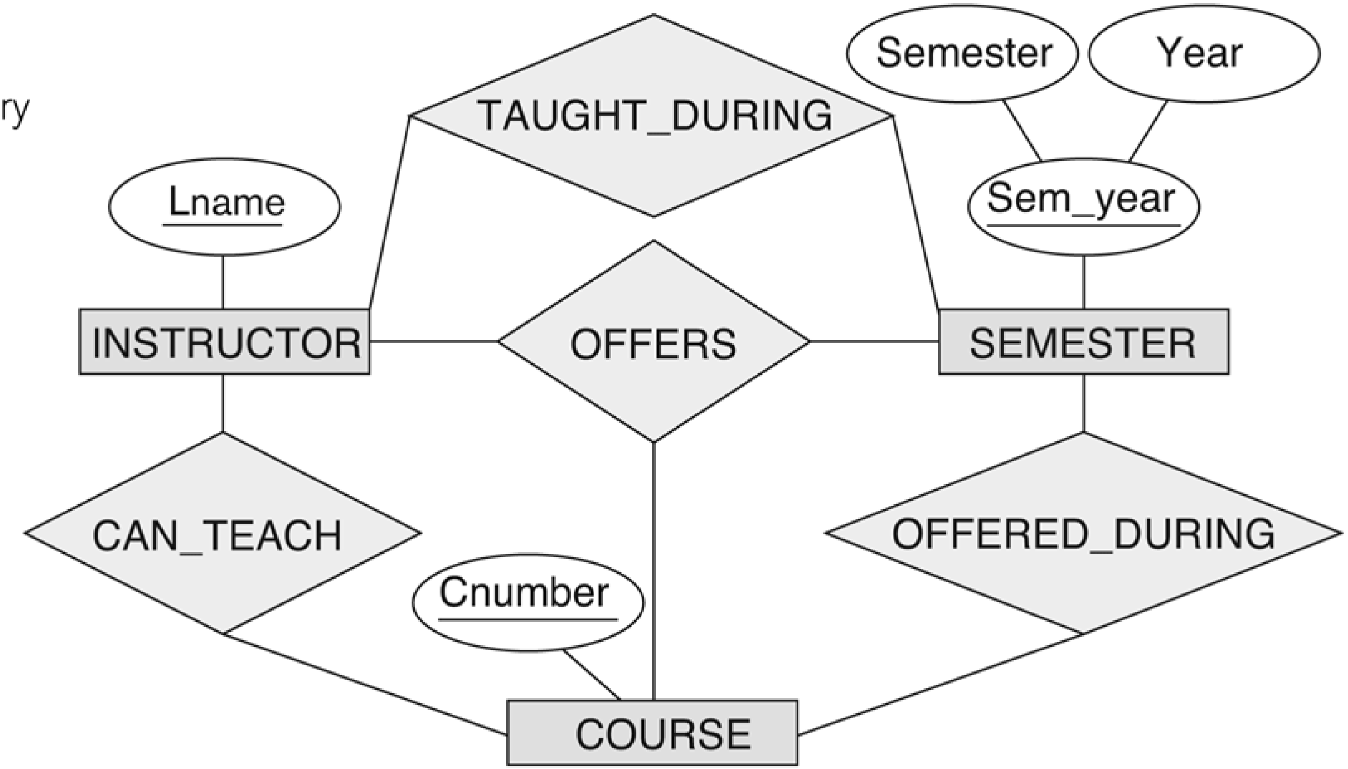
- Choose singular names for entity types
- Entity type and relationship type names are uppercase letters
- Attribute names are initial letter capitalized, and role names are lowercase letters
- A concept may be first modeled as an attribute and then refined into a relationship
- An attribute that exists in several entity types may be elevated to an independent entity type
- An entity type with a single attribute is related to only one other entity type, may be reduced to an attribute of related entity type

Discussion of n-ary relationships

- In general, an n-ary relationship is not equivalent to n binary relationships
- Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships
- In general, 3 binary relationships can represent different information than a single ternary relationship
- If needed, the binary and n-ary relationships can all be included in the schema design

Discussion of n-ary relationships

ternary versus binary
relationship types.



Enhanced ER [EER]

- ER model concepts are sufficient for representing many database schemas for traditional database applications
- Current applications such as CAD/CAM, tele-communications, GIS,... have more complex requirements
- Led to the development of *semantic data modeling* concepts
- ER model can enhanced to include semantic data model leading to Enhanced ER [EER] model

Enhanced ER [EER]

- Includes all modeling concepts of basic ER
- Additional concepts:
  - subclasses/superclasses
  - specialization/generalization
  - categories (UNION types)
  - attribute and relationship inheritance
- The additional EER concepts are used to model applications more completely and more accurately

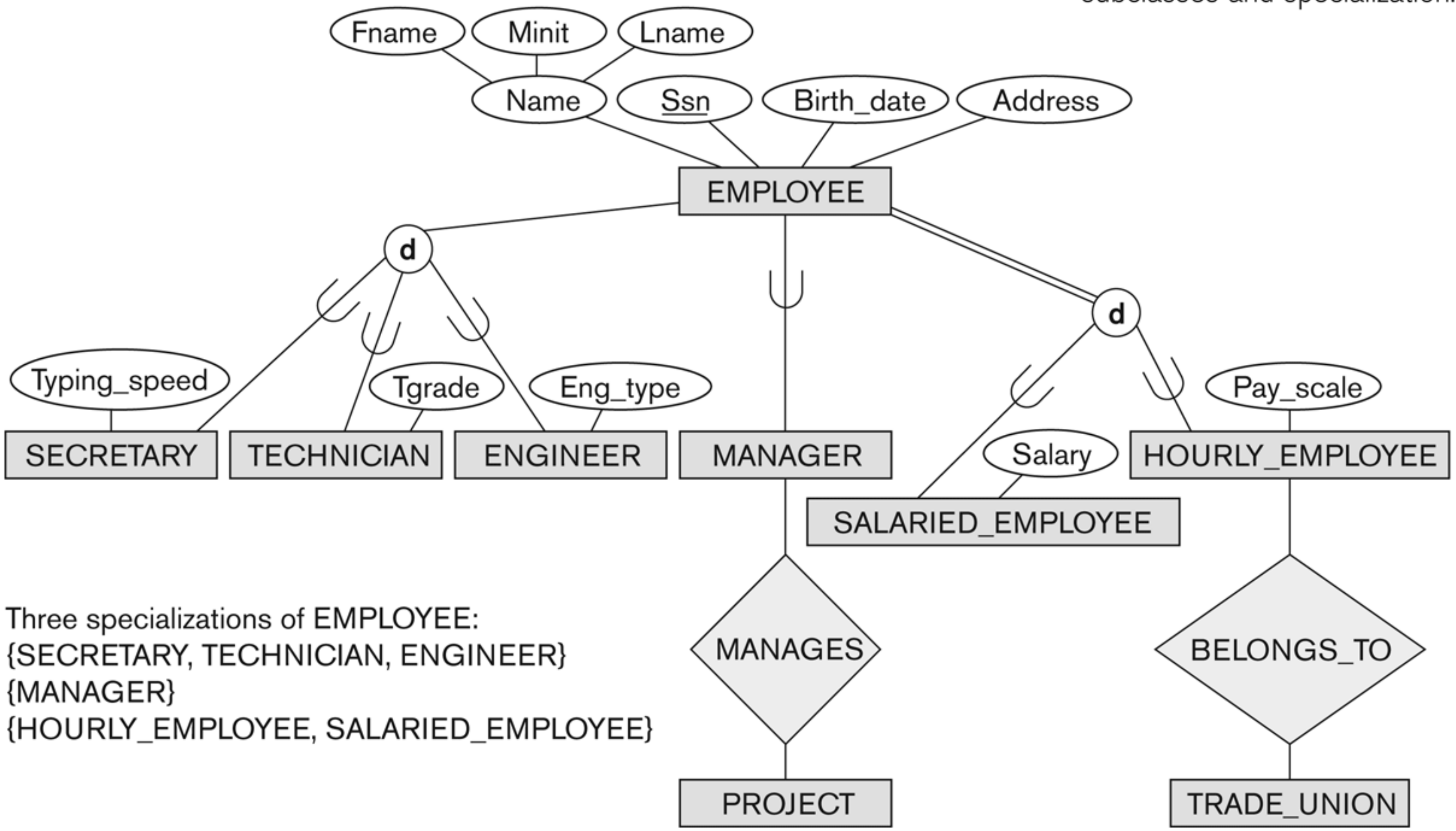
Subclasses and Superclasses

- Enhanced ER data model brings a number of new concepts:
  - ***Superclass*** / ***subclass*** relationship, called IS-A hierarchy, as well, together with specialization / generalization procedures, and
  - ***Category*** (a subset of the union of two different superclass entity sets)
  - ***Aggregate*** (as a representation of complex objects)

Subclasses and Superclasses

- An entity type may have additional meaningful subgroupings of its entities
  - Example: SECRETARY, ENGINEER, TECHNICIAN, MANAGER, etc.,
- Each of these subgroupings is a subset of EMPLOYEE entities
- Each is called a subclass of EMPLOYEE
- EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass [IS-A] relationships
- It is not necessary that every entity in a superclass be a member of some subclass

Subclasses and Superclasses



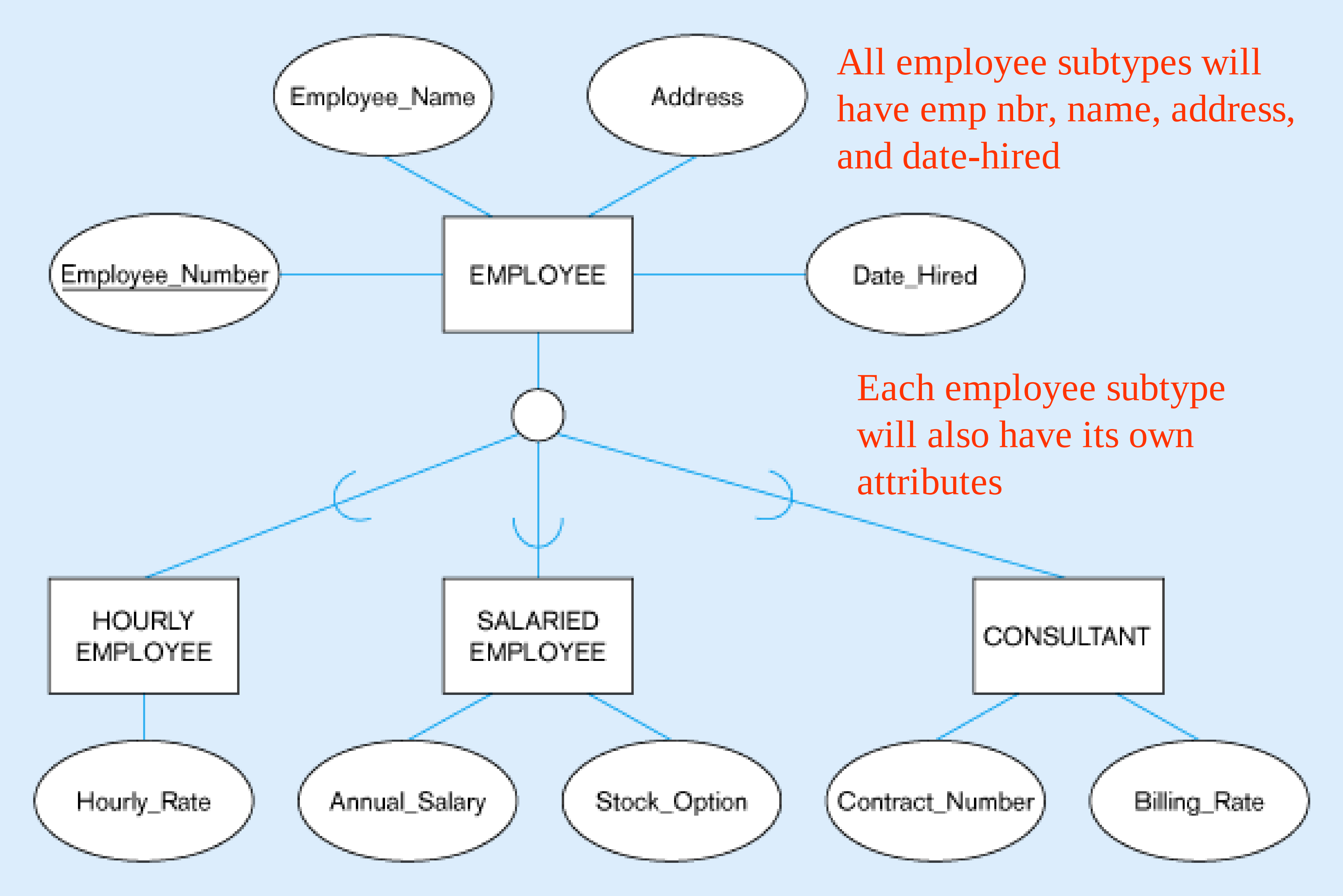
EER diagram notation to represent
subclasses and specialization.

Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Specialization

- An entity that is member of a subclass *inherits*
  - All attributes of the entity as a member of the superclass
  - All relationships of the entity as a member of the superclass
- Specialization is the process of defining a set of subclasses of a superclass
  - based upon some distinguishing characteristics of the entities in the superclass
  - may have several specializations of the same superclass
  - attributes of a subclass are called *specific* or *local* attributes
  - the subclass can also participate in specific relationship types

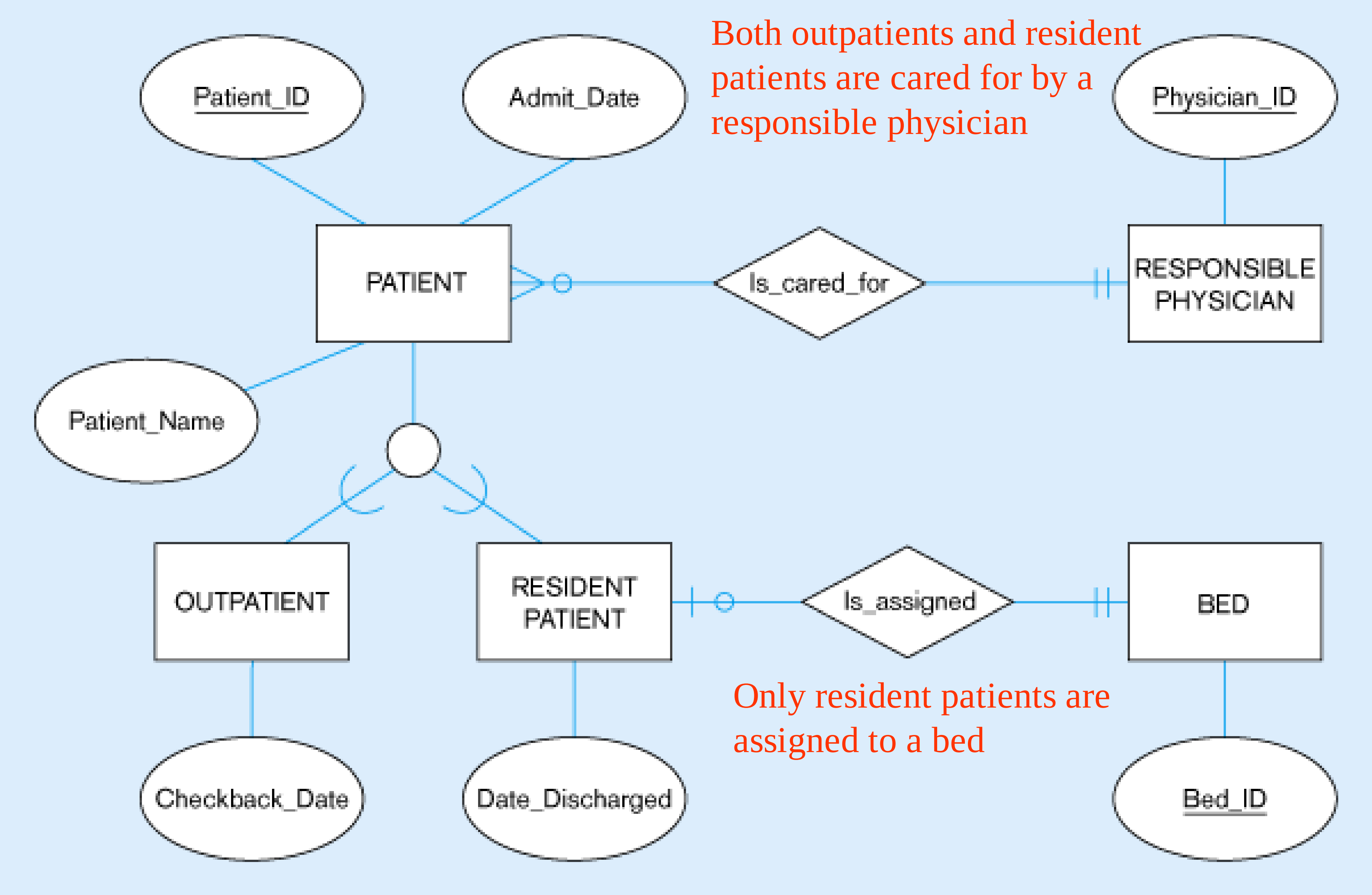
Specialization



All employee subtypes will have emp nbr, name, address, and date-hired

Each employee subtype will also have its own attributes

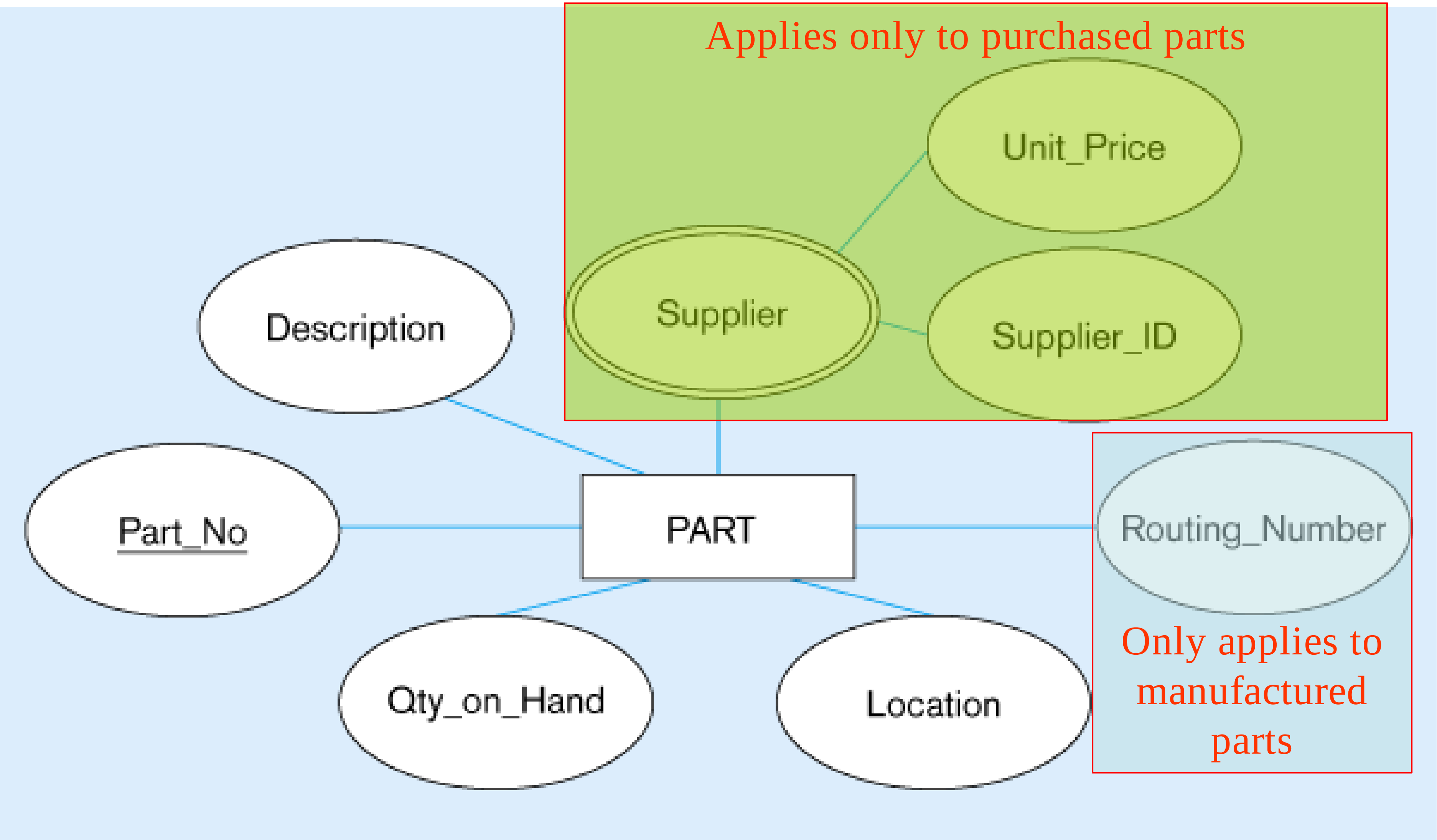
Specialization



Both outpatients and resident patients are cared for by a responsible physician

Only resident patients are assigned to a bed

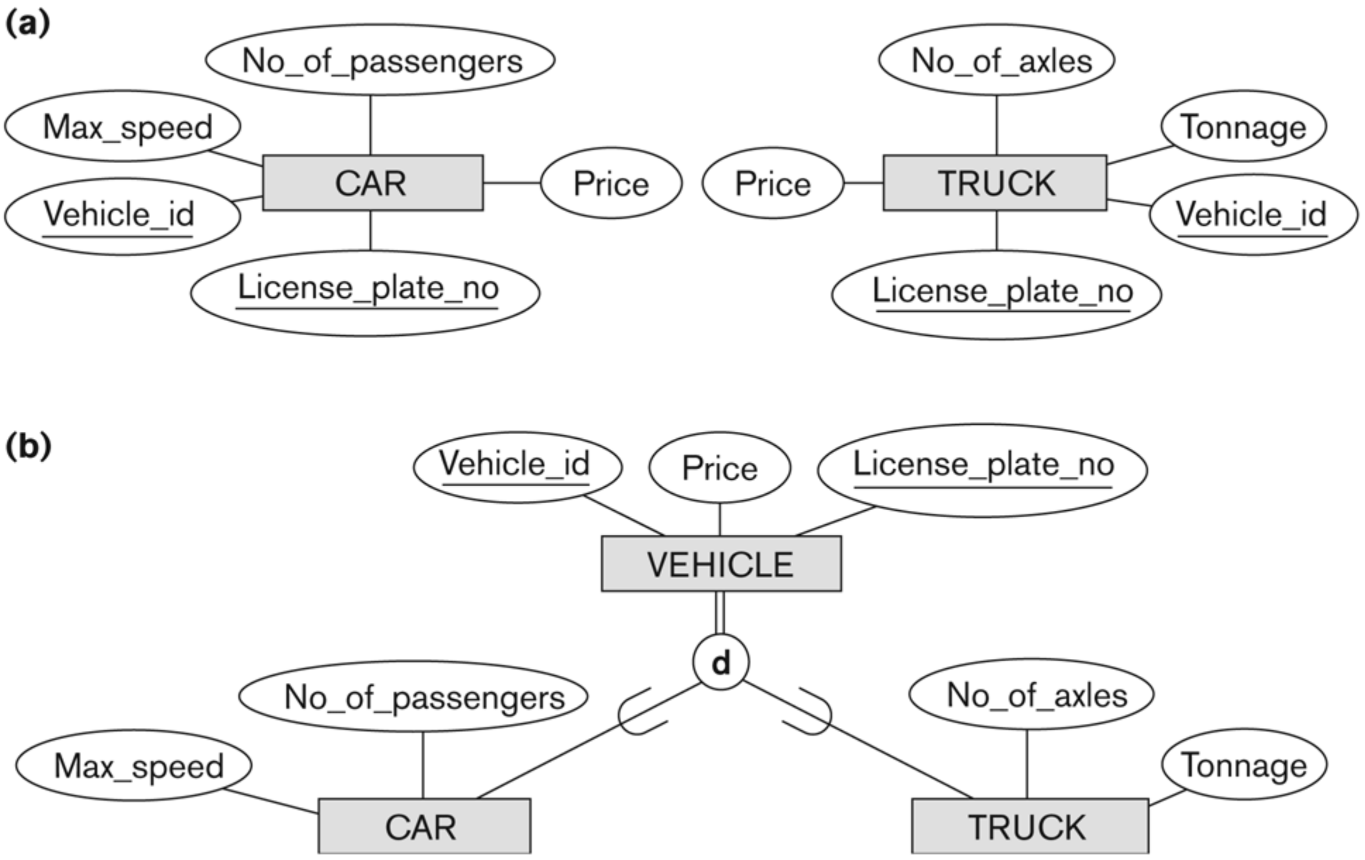
Example of Specialization



Generalization

- Generalization is the reverse of the specialization process
- Several classes with common features are generalized into a superclass
- Diagrammatic notation are sometimes used to distinguish between generalization and specialization – but it is subjective

Generalization



Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

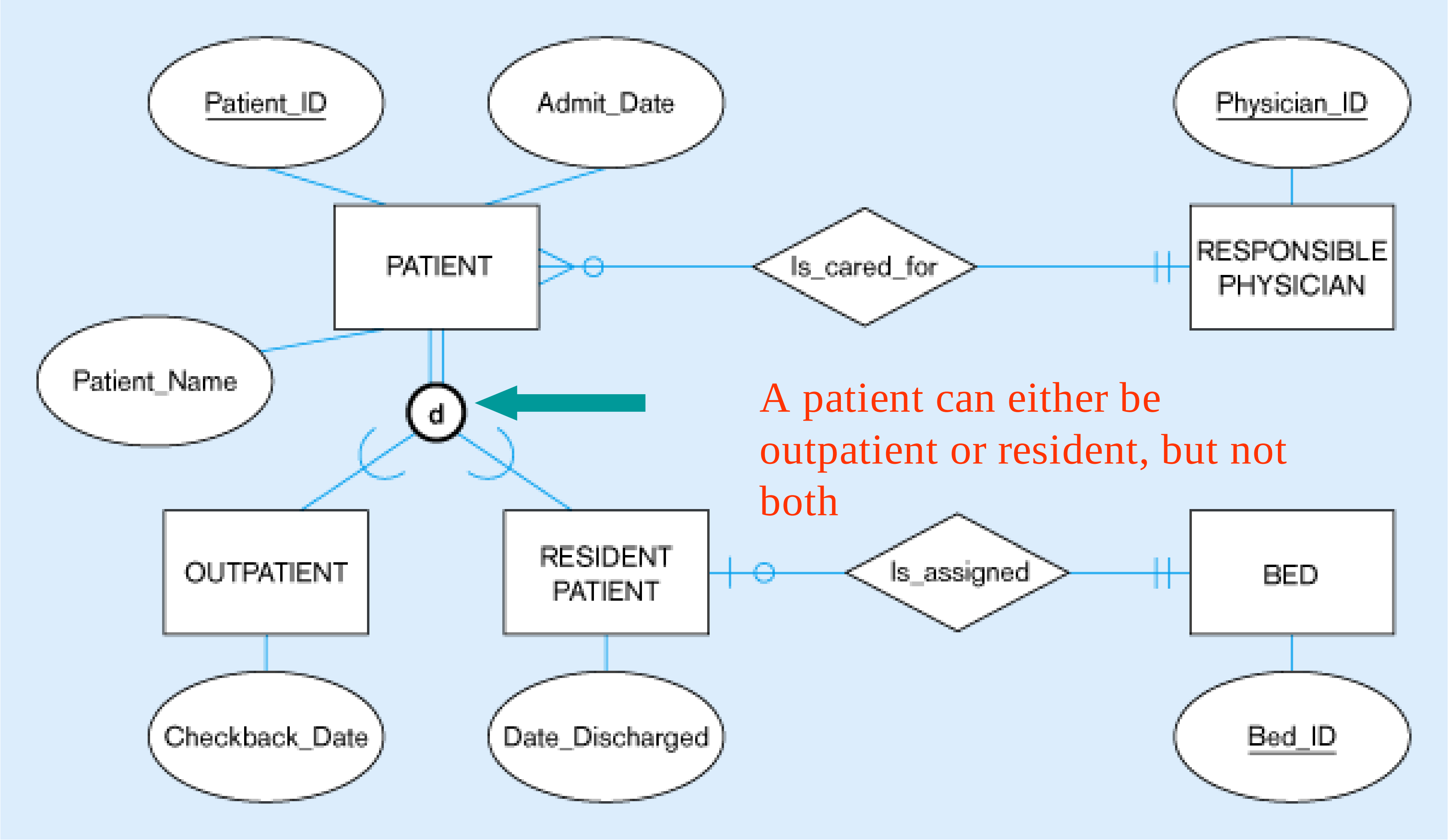
Constraints on Specialization/Generalization

- Two basic constraints can apply to a specialization/generalization:
  - Disjointness Constraint:
  - Completeness Constraint:

Disjointness Constraint

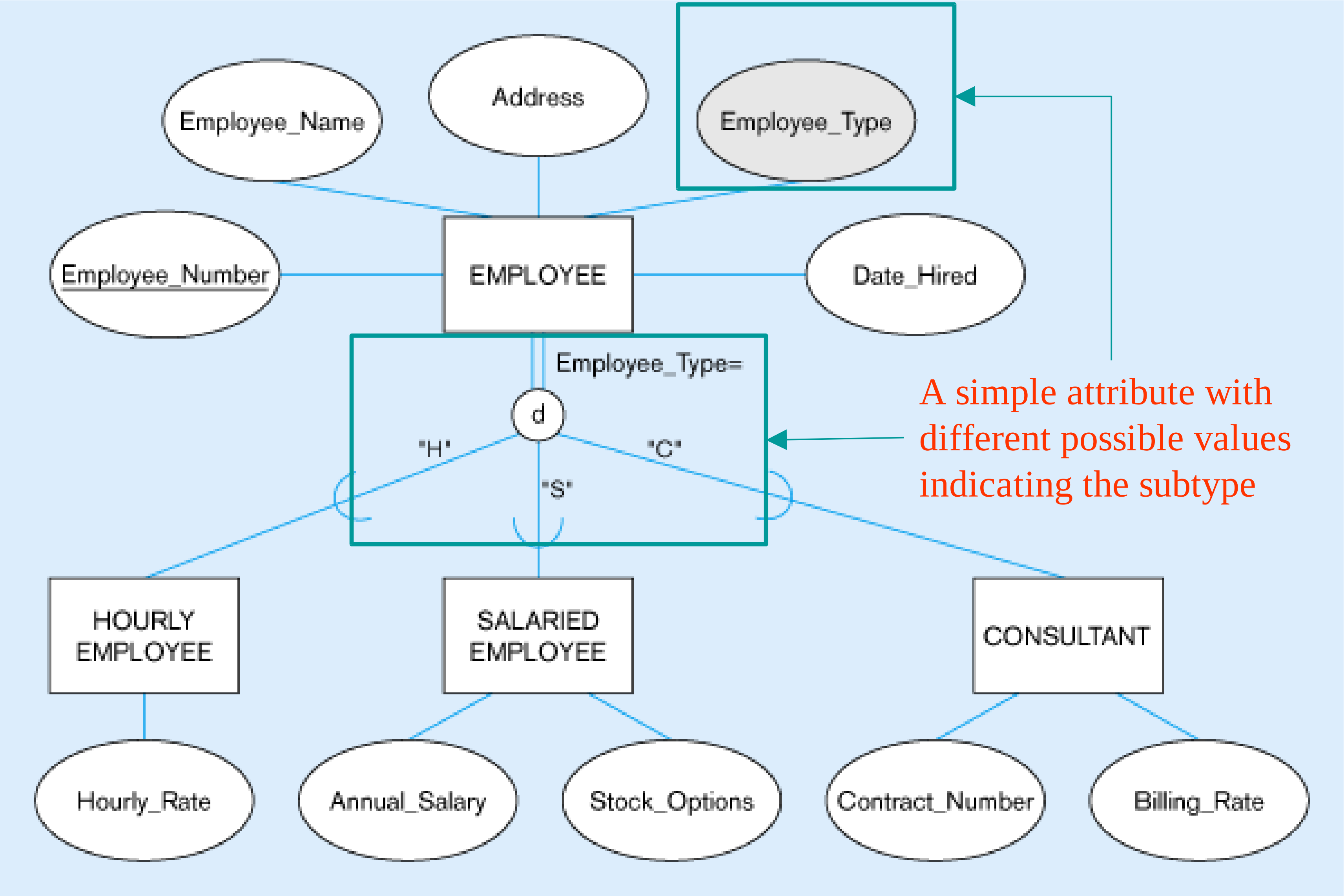
- Disjointness Constraint:
  - Specifies that the subclasses of the specialization must be *disjoint*:
    - an entity can be a member of at most one of the subclasses of the specialization
  - Specified by ***d*** in EER diagram
  - If not disjoint, specialization is *overlapping*:
    - that is the same entity may be a member of more than one subclass of the specialization
  - Specified by ***o*** in EER diagram

Disjoint



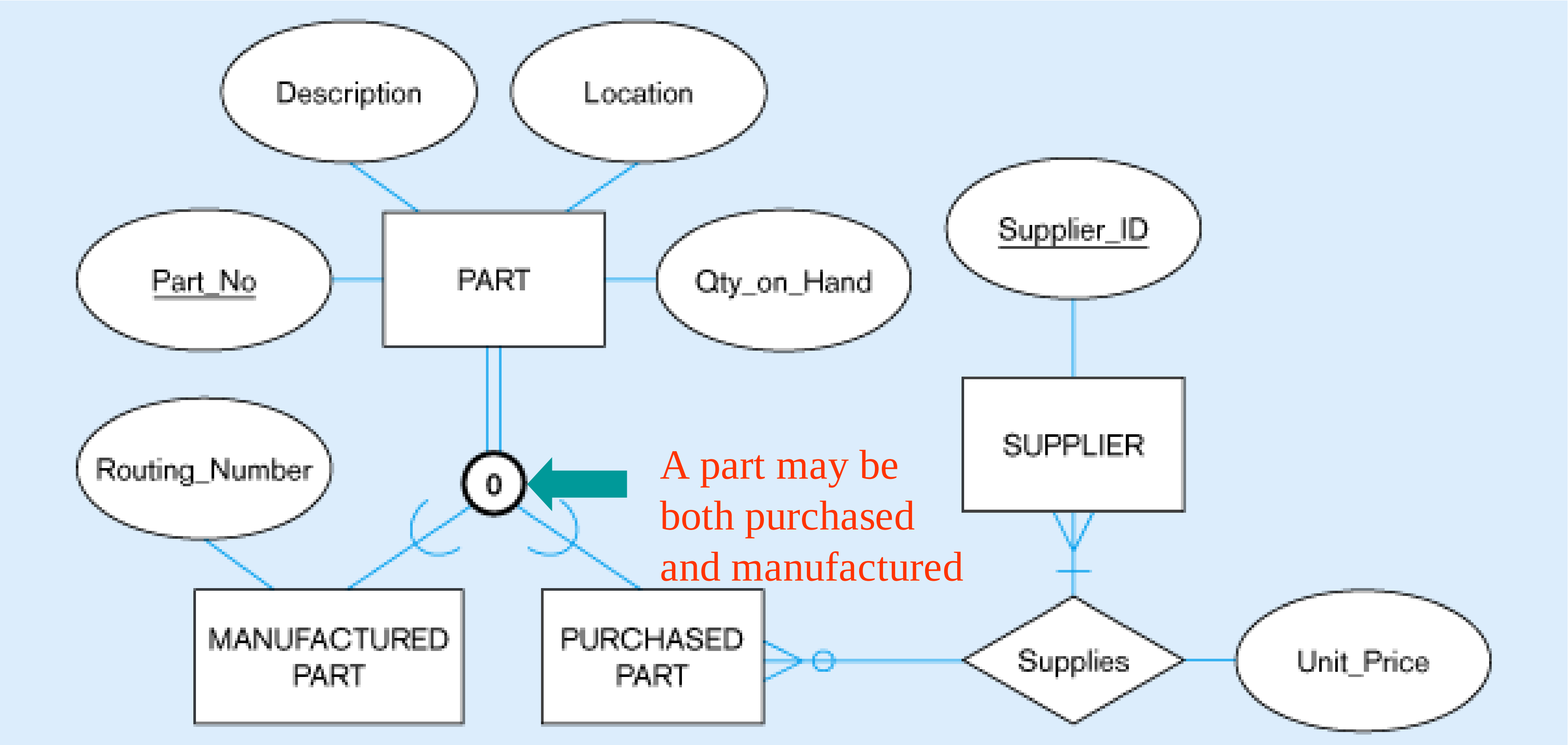
A patient can either be outpatient or resident, but not both

Subtype Discriminator



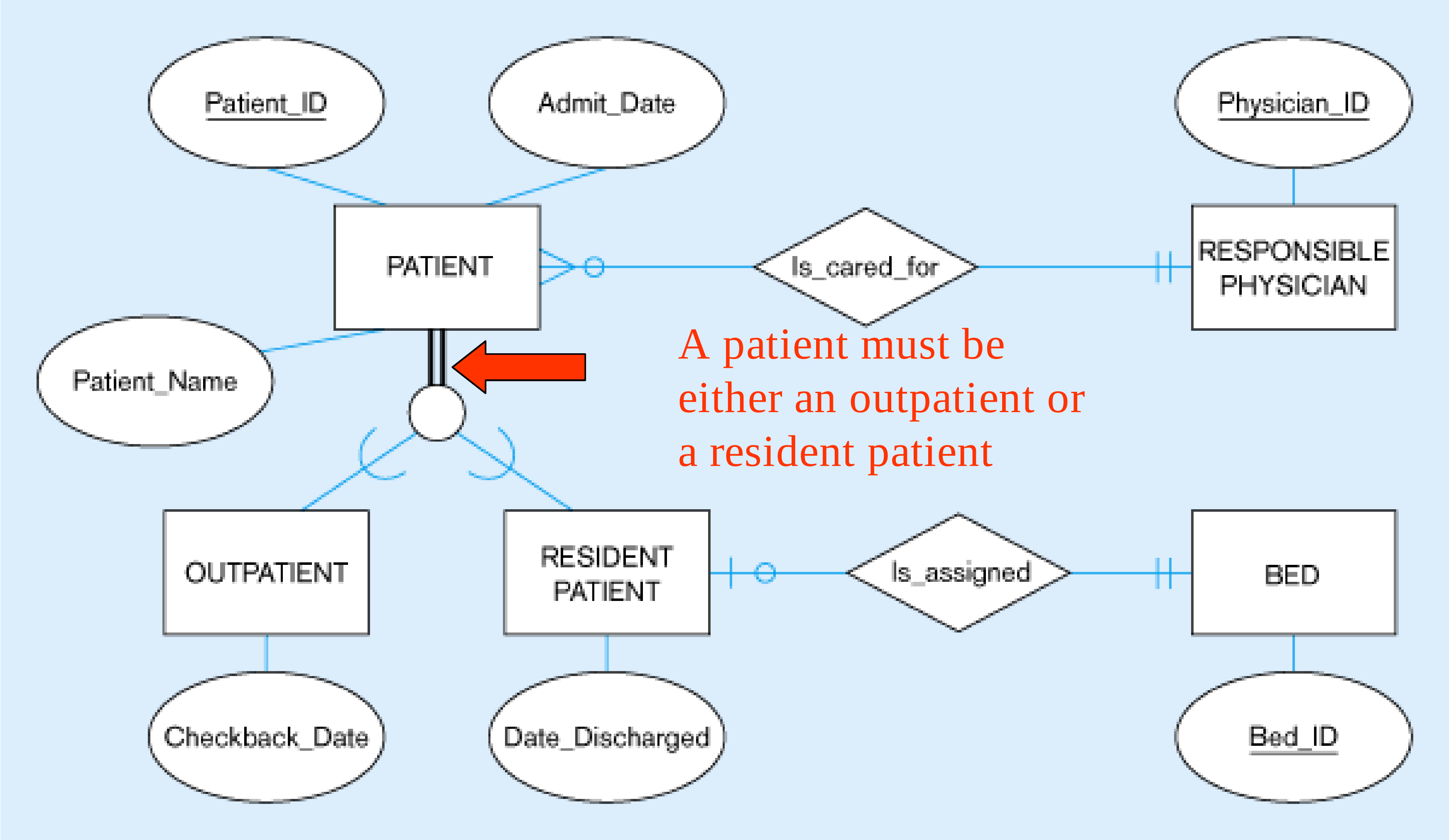
A simple attribute with different possible values indicating the subtype

Overlap



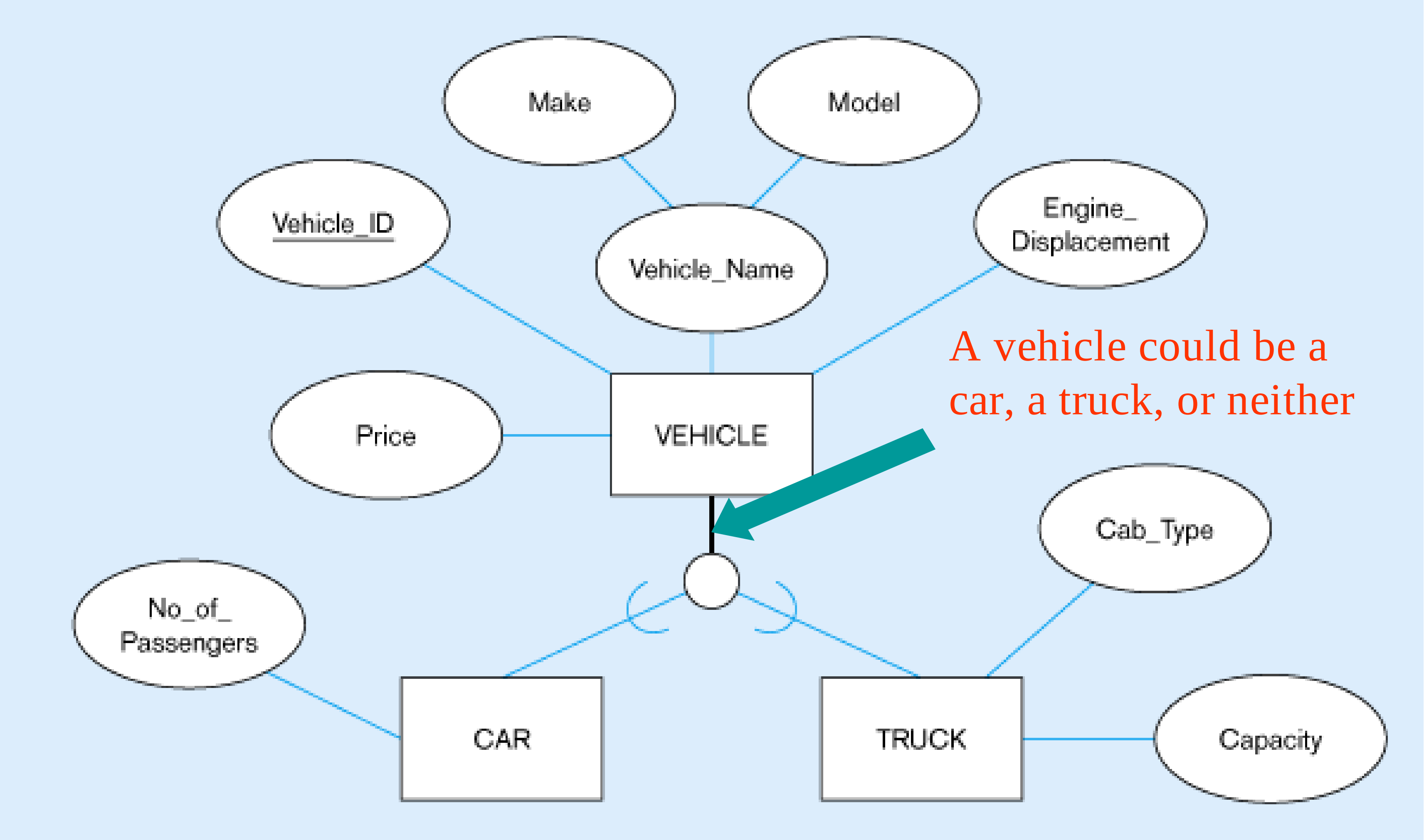
A part may be both purchased and manufactured

Completeness Constraint – Total



A patient must be either an outpatient or a resident patient

Completeness Constraint – Partial



A vehicle could be a car, a truck, or neither

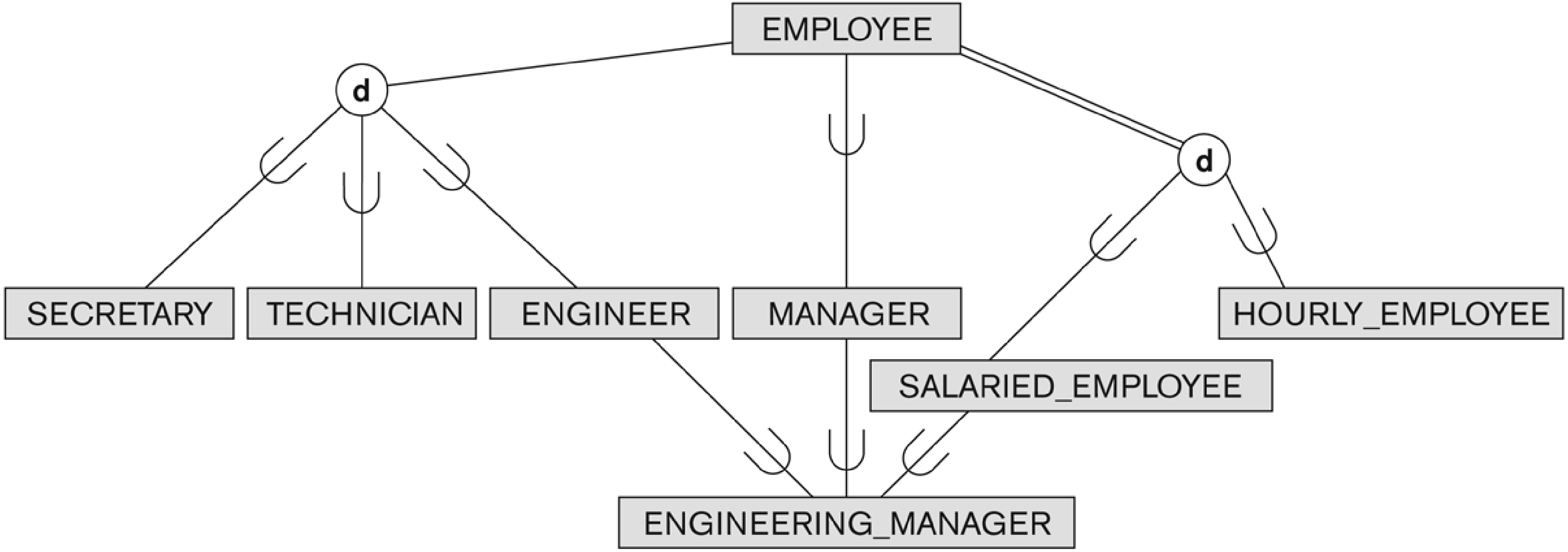
Constraints on Specialization/Generalization

- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial
- Note: Generalization usually is total because the superclass is derived from the subclasses.

Hierarchies, Lattices & Shared Subclasses

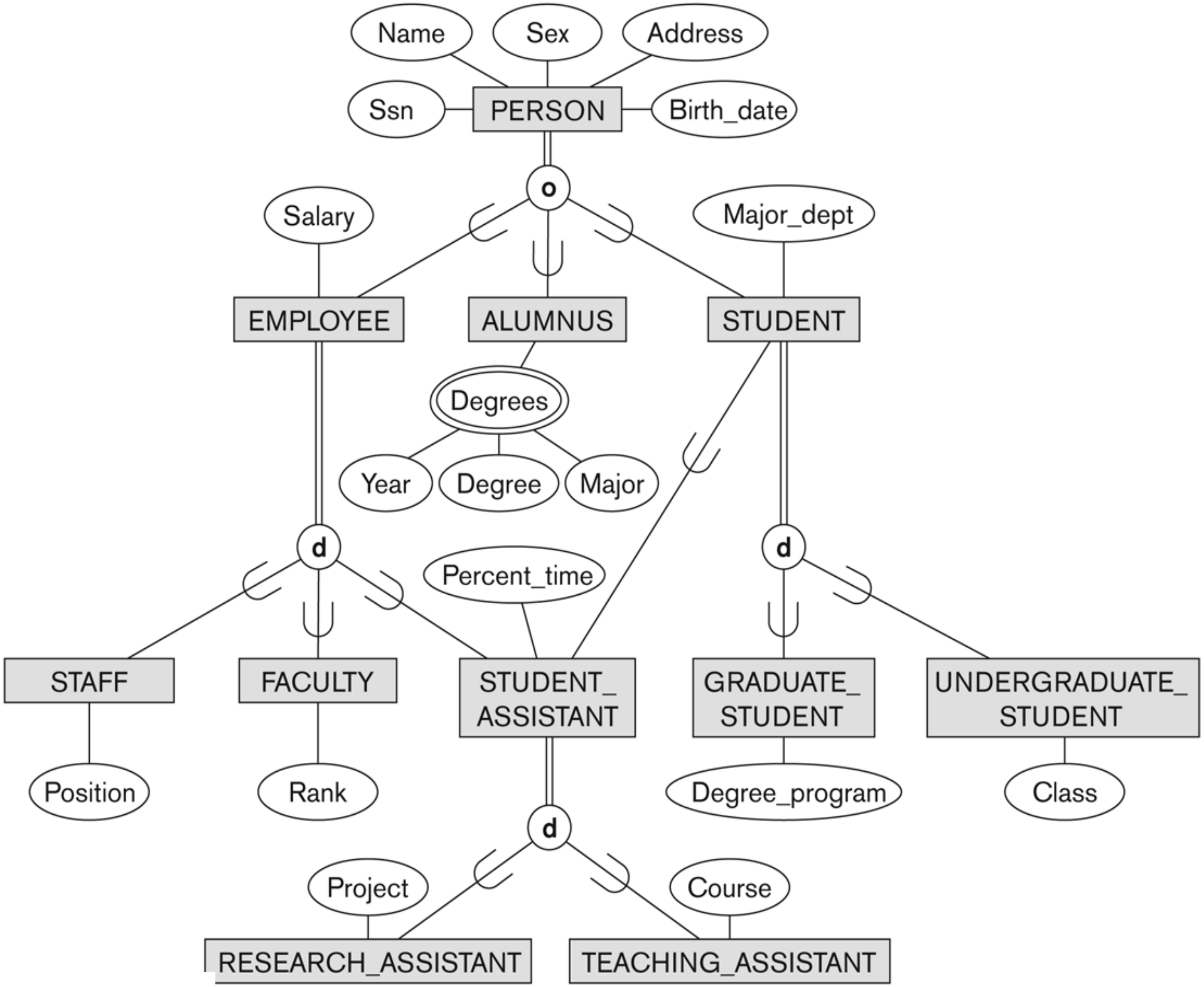
- A subclass may itself have further subclasses specified on it
  - forms a hierarchy or a lattice
- ***Hierarchy*** has a constraint that every subclass has only one superclass (called ***single inheritance***); this is basically a ***tree structure***
- In a ***lattice***, a subclass can be subclass of more than one superclass (called ***multiple inheritance***)
- In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses
- A subclass with more than one superclass is called a shared subclass (multiple inheritance)

Hierarchies, Lattices & Shared Subclasses



A specialization lattice with shared subclass ENGINEERING_MANAGER.

- Classwork EER



A specialization lattice with multiple inheritance for a UNIVERSITY database.

Categories – UNION Types

- All of the *superclass/subclass relationships* we have seen thus far have a single superclass
- A shared subclass is a subclass in:
  - *more than one* distinct superclass/subclass relationships
  - each relationships has a single superclass
  - shared subclass leads to multiple inheritance
- In some cases, we need to model a *single superclass/subclass relationship* with *more than one* superclass
- Superclasses can represent different entity types
- Such a subclass is called a category or UNION TYPE

Categories – UNION Types

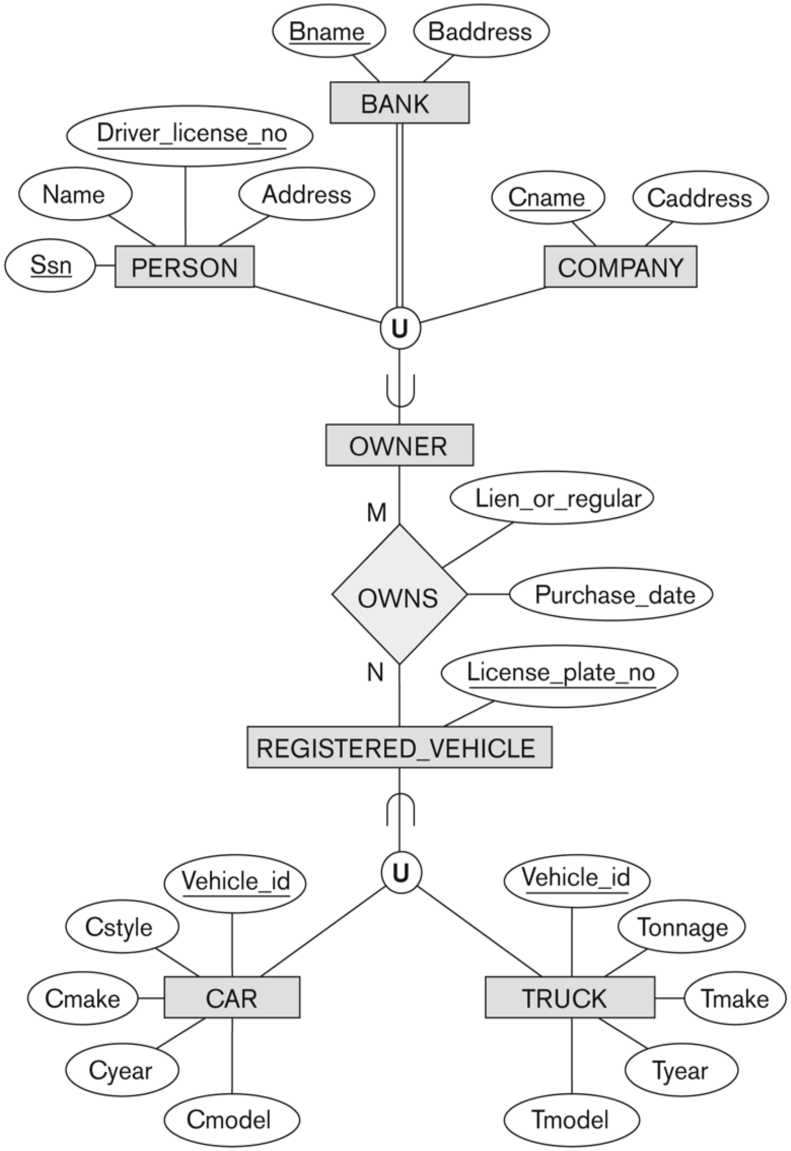


Figure 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

Summary

- ER Model Concepts: Entities, attributes, relationships
- Constraints in the ER model
- Using ER in step-by-step conceptual schema design for the COMPANY database
- Introduced the EER model concepts
  - Class/subclass relationships
  - Specialization and generalization
  - Inheritance

References

Fundamentals of Database Systems

Ramez Elmasri, Shamkant B.Navathe, 5th Edition

THANK
YOU!