ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
Vietnam - Korea University of Information and Communication Technology

Database Systems

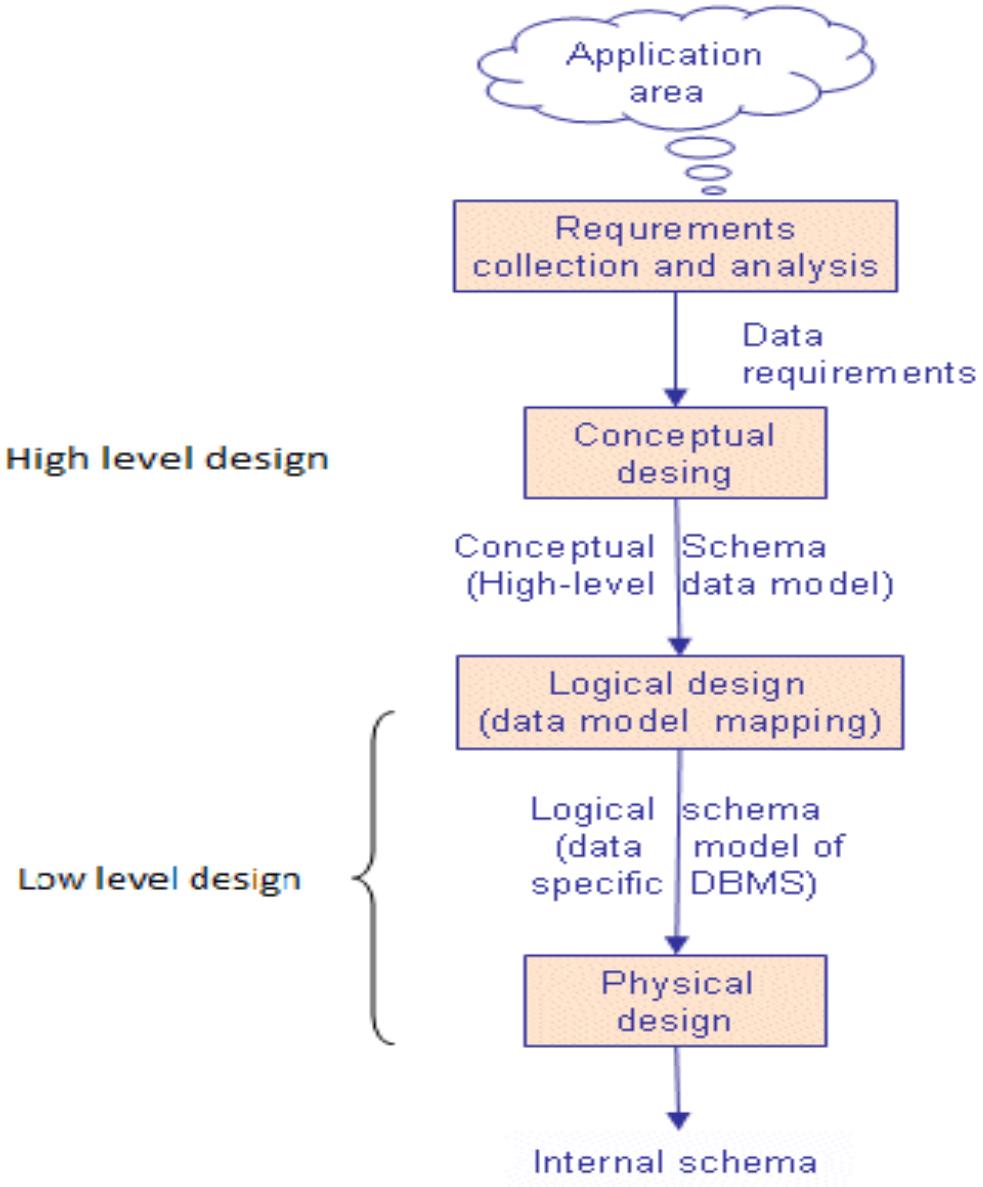
Chapter 2: Database Design

Session 1:
Entity Relationship Model

Outline

1. Database Design Process
2. Entity Relationship Model
3. Constructing an Entity Relationship Model
4. The Enhanced Entity Relationship Model

Database Design Process



Requirements Collection and Analysis

- Is a preliminary stage to database design.
- The process of collecting and analyzing information about the part of the organization that is to be supported by the database system, and using this information to identify the requirements for the new system.
- The outcome of this phase is a **user requirements specification**.

High level design

- **Conceptual database design**: is the process of constructing a data model from the information collected in the requirements analysis phase.
  - It is independent of any particular DBMS at high level design.
  - It construct an **entity relationship model** (ERM)
  - ERM includes **entities** that are represented in the database, **attributes** of the entities, the **relationships** among the entities, and **constraints** on the entities and relationships.

Low level design

- The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases:
  - Logical design
  - Physical design

Low level design

- Logical design:
  - We must **choose a DBMS** to implement our database design, and map the high level conceptual data model in the previous phase onto a logical data model of the chosen DBMS (relational, network, hierarchical, object-oriented)
  - We will only consider **relational DBMS**
  - **Convert an ER diagram from the conceptual model into a relational schemas in normal form using Normalization.**
    - Normalization is used to check the entity relationship model and help eliminate redundancy and other anomalies in the database.

Low level design

- Physical design:
  - Describes how the database is to be implemented:
    - Creating a set of relational tables and the constraints on these tables from the information presented in the logical data model;
    - Identifying the specific storage structures and access methods for the data to achieve an optimum performance for the database system;
    - Designing security protection for the system.

Entity Relationship Model

- ERM is a high level data model used for developing the conceptual design of the database.
- The ER model also has an associated diagrammatic representation, the **ER diagram**, which can express the overall logical structure of a database graphically
- Many various notations used with ERD—the original Chen notation and the newer Crow's Foot and UML notations.
  - The Chen notation favors conceptual modeling.
  - The Crow's Foot notation favors a more implementation-oriented approach
  - The UML notation can be used for both conceptual and implementation modeling.

Entity Relationship Model

- ERM has 3 basic concepts:
  - Entity
  - Attributes
  - Relationships

Entity

- An entity is a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data.
- An entity has a noun name.

  Example:
  - Person: Student, teacher, employee, author,..
  - Place: store, classroom, warehouse,…
  - Object: Movie, product, car, book,..
  - Event: Sale, Registration, Reservation
  - Concept: Account, Course

Entity

- An entity type (or entity set) is a collection of entities that share the common properties, or characteristics.
  - Example: the entity type **student** might represent the set of all students in the university.
- An **entity instance** is a single occurrence of an entity type.
  - For example, there is one EMPLOYEE entity type in most organizations, but there may be hundreds (or even thousands) of instances of this entity type stored in the database.

Entity

- Example: Entity type versus Entity instance

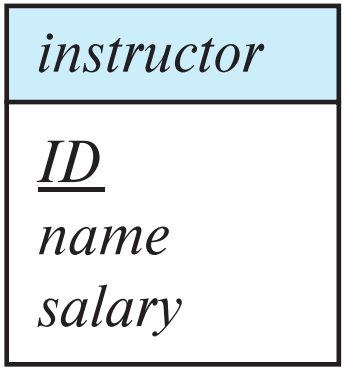
Entity type: EMPLOYEE

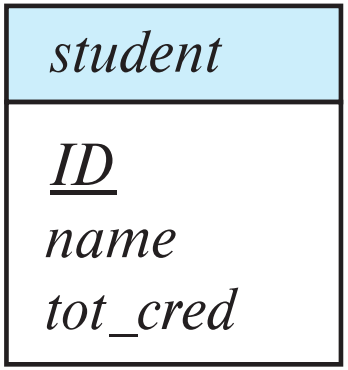
Attributes	Attribute Data Type	Example Instance	Example Instance
Employee Number	CHAR (10)	642-17-8360	534-10-1971
Name	CHAR (25)	Michelle Brady	David Johnson
Address	CHAR (30)	100 Pacific Avenue	450 Redwood Drive
City	CHAR (20)	San Francisco	Redwood City
State	CHAR (2)	CA	CA
Zip Code	CHAR (9)	98173	97142
Date Hired	DATE	03-21-1992	08-16-1994
Birth Date	DATE	06-19-1968	09-04-1975

An employee type Employee with 2 instances

Entity

- In an E-R diagram, the entity name is placed inside the box representing the entity type
- Primary key is underlined.



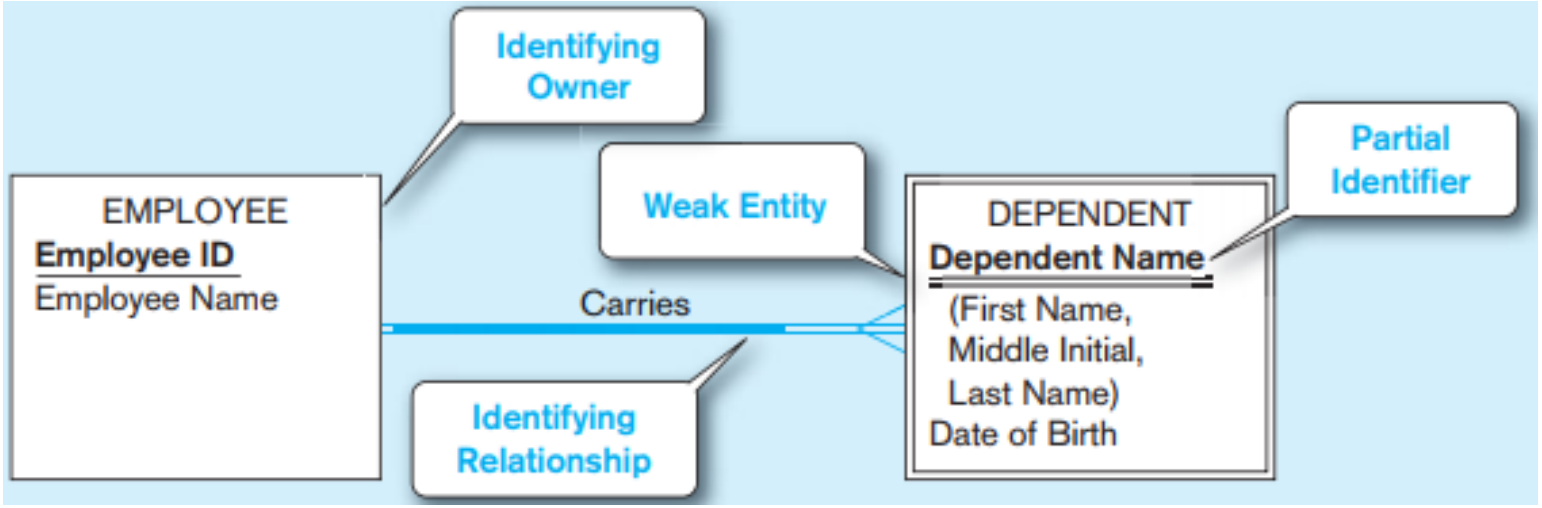


Entity

- There are two types of entities; weak and strong entity types.
  - A **strong entity type** is one that exists independently of other entity types.
    - Instances of a strong entity type always have a unique characteristic (called an *identifier*)—that is, an attribute or a combination of attributes that uniquely distinguish each occurrence of that entity.
  - A **weak entity type** is an entity type whose existence depends on some other entity type
    - The entity type on which the weak entity type depends is called the **identifying owner**
    - A weak entity type does not typically have its own identifier

Entity

- Example: Weak entity in ERD
  - Carries relationship is the identifying relationship (indicated by the double line)
  - The attribute Dependent Name serves as a *partial* identifier (indicated by double underline)



Attribute

- Attributes are properties or characteristics of an entity type.
- An attribute has a noun name
  - Example:
    - Student: Student ID, Student Name, Home Address, Phone Number, Major
    - Automobile: Vehicle ID, Color, Weight
    - Employee: Employee ID, Employee Name, Skill
- In E-R diagrams, we represent an attribute by placing its name in the entity it describes.
- Attributes may also associated with relationships.

Attribute

- Attributes have a **domain** which is the set of possible values for a given attribute, or domain is the set of permitted values for each attribute
  - Example:
    - for the gender attribute consists of only two possibilities: M or F
    - for grade is 0 to 10

Identifier Attribute

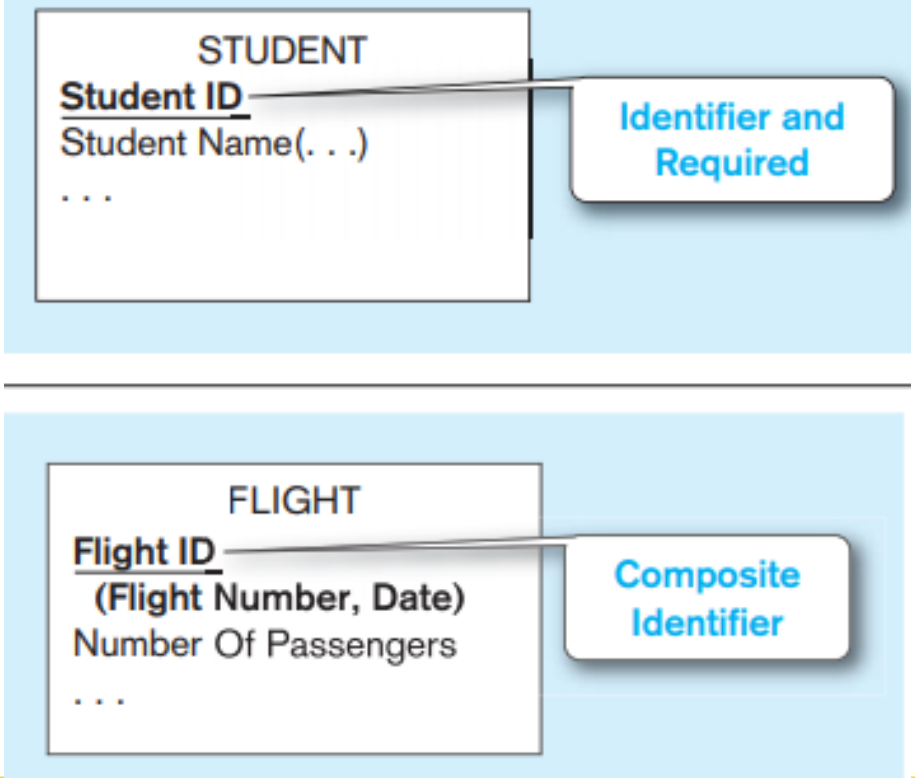
- Attribute types:
  - Identifier Attribute
  - Required and Optional attribute
  - Simple and Composite attribute
  - Single-valued and Multivalued attribute
  - Stored and Derived attribute

Identifier Attribute

- Identifier (Key)–an attribute (or combination of attributes) that uniquely identifies individual instances of an entity type. That is, no two instances of the entity type may have the same value for the identifier attribute.
  - Ex: Student ID is an identifier of Student entity type.
- Candidate Identifier–an attribute that could be a key…satisfies the requirements for being an identifier
  - Student Name is not a candidate identifier, because many students may potentially have the same name.

Identifier Attribute

- Simple versus Composite Identifier
  - A **composite identifier** is an identifier that consists of a composite attribute.



The identifier is boldfaced and underlined

Identifier Attribute

- Some entities may have more than one candidate identifier. The following criteria for selecting identifiers:
  - Will not change in value
  - Will not be null
  - No intelligent identifiers (e.g., containing locations or people that might change)
  - Substitute new, simple keys for long, composite keys

Null values

- In some cases, particular entity may not have an applicable value for an attribute. *Null* can also designate that an attribute value is **unknown**.
- Two cases:
  1. The attribute value is missing.
     - Ex: Height attribute is NULL.
  2. It is not known whether the attribute value exist.
     - Ex: College degree – some people may have it and some not

Simple & Composite Attribute

- Required attribute is an attribute that must have a value in it.
- An optional attribute that may not have a value in it and can be left blank.
- Example:

Entity type: STUDENT

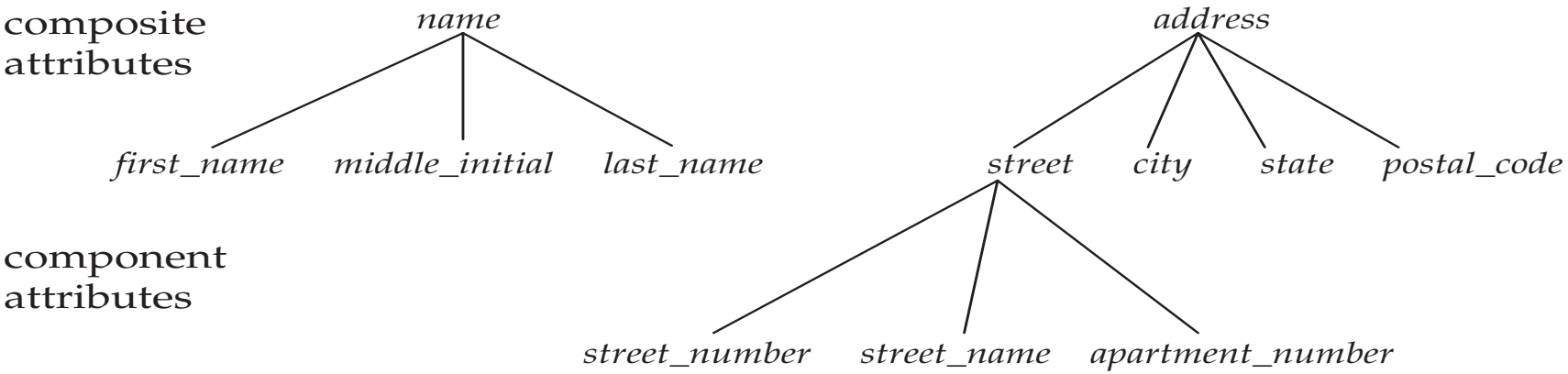
Attributes	Attribute Data Type	Required or Optional	Example Instance	Example Instance
Student ID	CHAR (10)	Required	876-24-8217	822-24-4456
Student Name	CHAR (40)	Required	Michael Grant	Melissa Kraft
Home Address	CHAR (30)	Required	314 Baker St.	1422 Heft Ave
Home City	CHAR (20)	Required	Centerville	Miami
Home State	CHAR (2)	Required	OH	FL
Home Zip Code	CHAR (9)	Required	45459	33321
Major	CHAR (3)	Optional	MIS	

Simple & Composite Attribute

- Simple Attribute
  - Attribute that have not been divided into subparts
  - Simple (Atomic) attributes
    - Ex: Age, City, Postal Code
- Composite Attributes
  - Can be divided into smaller subparts, which represent more basic attributes with independent meaning.
    - Ex: Name can broken down into component attributes: *first_name, middle_initial, last_name*

Simple & Composite Attributes

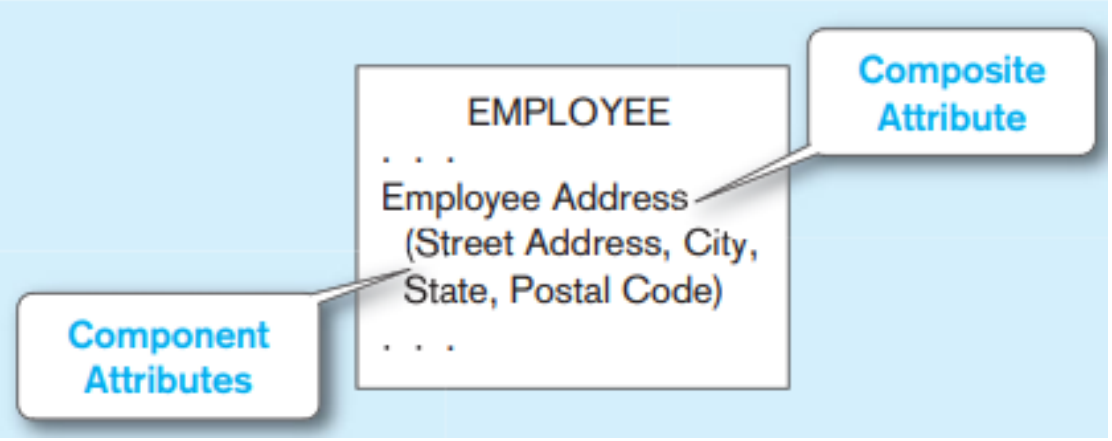
- Composite attributes
  - Example: name, address



- The decision about whether to subdivide an attribute into its component parts depends on whether users will need to refer to those individual components, and hence, they have organizational meaning.
- Most drawing tools do not have a notation for composite attributes, so you simply list all the component parts

Simple & Composite Attributes

- Composite attributes
  - Most drawing tools do not have a notation for composite attributes, so you simply list all the component parts



Single valued & Multivalued Attributes

- Single Valued Attributes
  - Can only have one value for the particular entity.
  - Ex: A student can have only one ID number
- Multivalued Attributes
  - Attributes that can have many values
  - Ex: A student can have more than one phone number, one skill
- Multivalued and composite are different concepts.
  - A multivalued attribute, may occur multiple times for each employee
  - Composite attributes, each of which occurs once for each employee, but which have component, more atomic attributes

Single valued & Multivalued Attributes

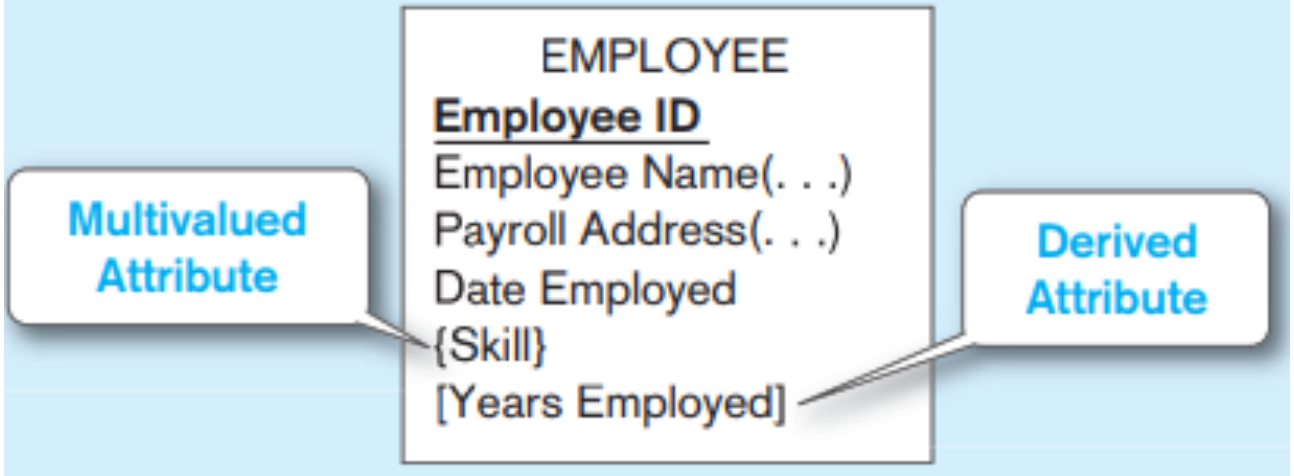
- Multivalued Attributes
  - Other E-R diagramming tools may use an asterisk (*) after the attribute name, or
    you may have to use supplemental documentation to specify a multivalued attribute

Stored & Derived Attributes

- Stored Attributes
  - The date_of_birth is the stored attribute
- Derived attributes
  - Can be computed from other attributes
  - Example: age is derived from date_of_birth.
  - Age is called derived attribute

Example

- E-R diagram with identifier, multivalued, and derived attributes



Relationship

- A **relationship** is an association among several entities
- A **relationship set** is a set of relationships of the same type.
- A relationship has a verb phrase name
- Two entities can have more than one type of relationship between them (multiple relationships)

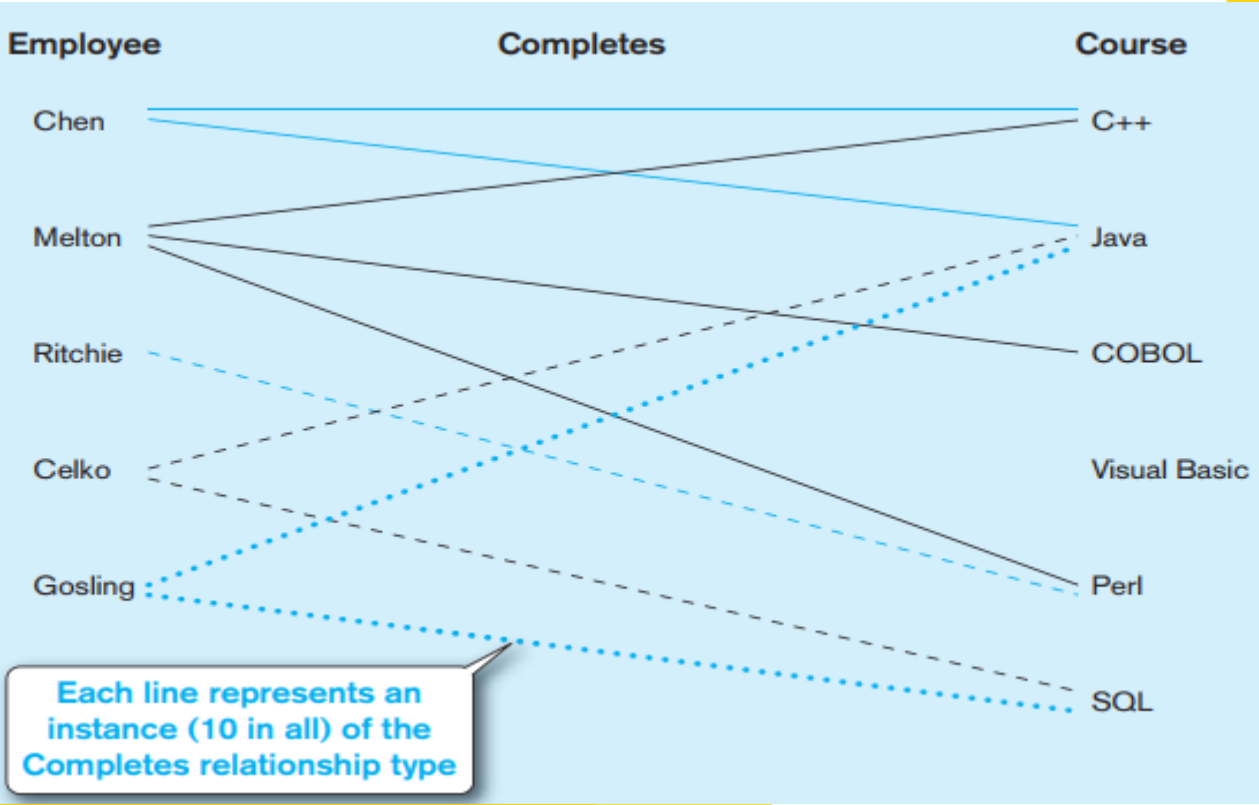
Relationship

- Relationship type and relationship instance
  - A **relationship type** is a meaningful association between (or among) entity types
    - The relationship type is modeled as lines between entity types
  - A relationship instance is an association between entity instances

Example Relationship type and Relationship instances

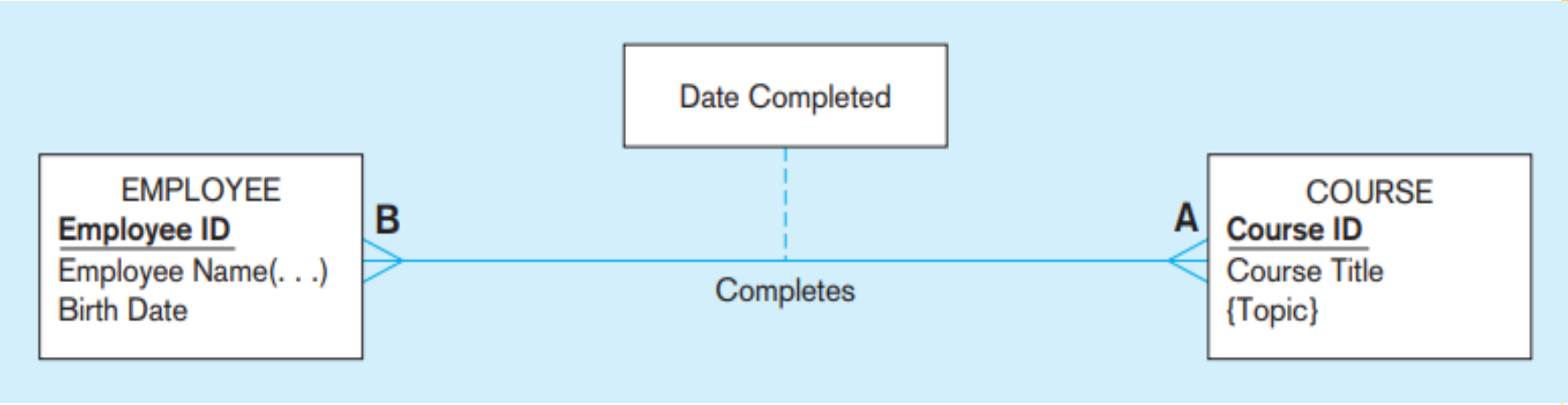
a) Relationship type

b) Relationship instances



Relationship

- Attributes on relationship
  - Relationships can have attributes which describe features pertaining to the association between the entities in the relationship
  - Example: Date Completed is a property of the relationship Completes, rather than a property of either Employee or Course entity.

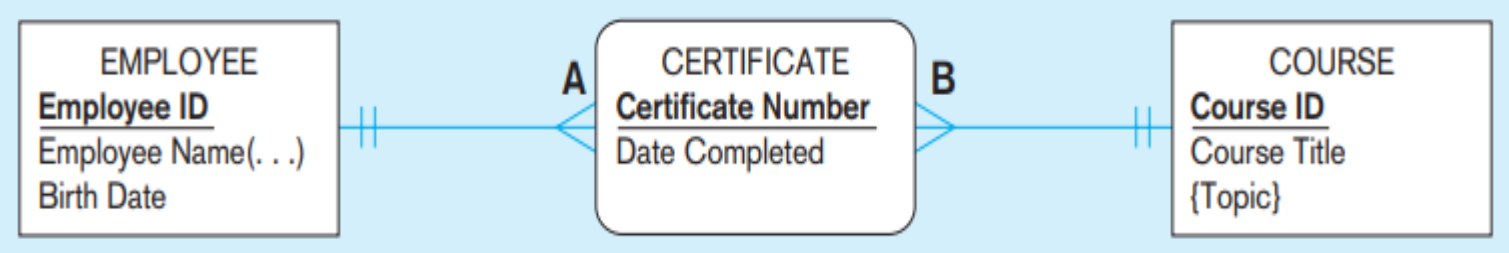


Relationship

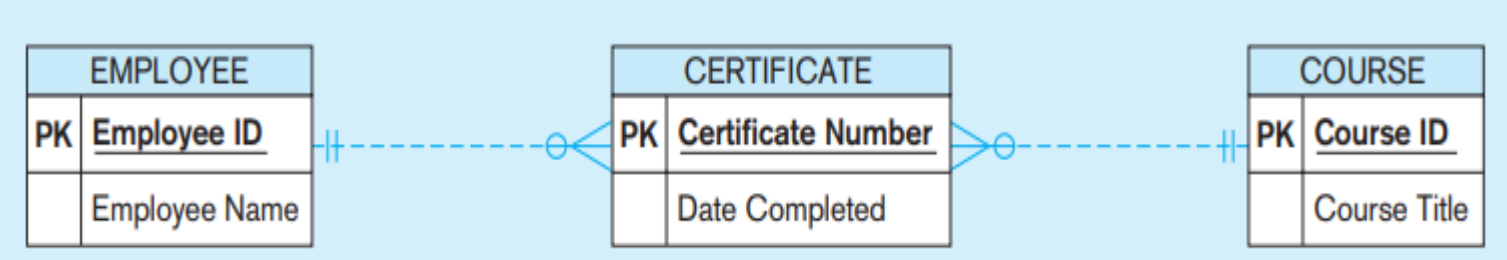
- Associative Entity
  - Associative entity is an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.
  - Associative entities are sometimes referred to as gerunds, because the relationship name (a verb) is usually converted to an entity name that is a noun.

Relationship

- Associative Entity
  - Example: **An associative entity Certificate**



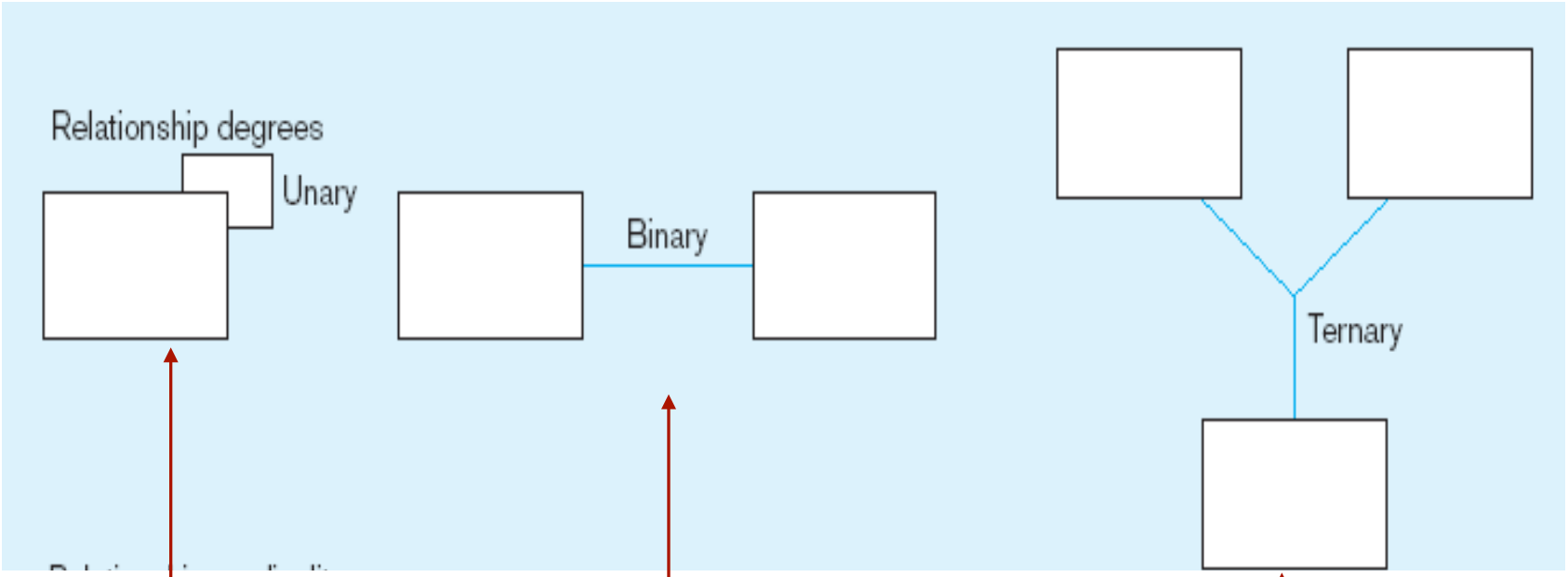
    - Note that there are no relationship names on the lines between an associative entity and a strong entity
  - An associative entity Certificate using Visio



Degree of a Relationship set

- The **degree of the relationship set** is the number of entity types that participate in a relationship.
  - **Unary relationship**, are also called Recursive relationship, which exists when association is maintained within a single entity (degree 1)
  - **Binary relationship**: exists when two entities are associated (degree 2)
    - Most relationship sets in a database system are binary.
  - **Ternary relationship**: exists when three entities are associated (degree 3)
- Relationships between more than three-entity types are rare.

Degree of a Relationship set



One entity related to another of the same entity type

Entities of two different types related to each other

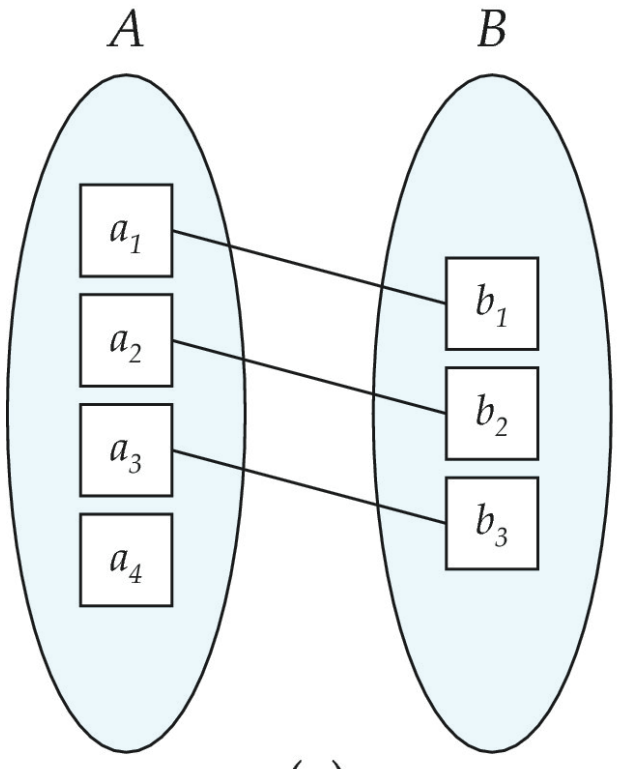
Entities of three different types related to each other

Cardinality constraints

- **Cardinality constraint**, or cardinality ratios, express the number of entities to which another entity can be associated via a relationship type.
- Mapping cardinalities are most useful in describing binary relationship set
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to many

Cardinality constraints

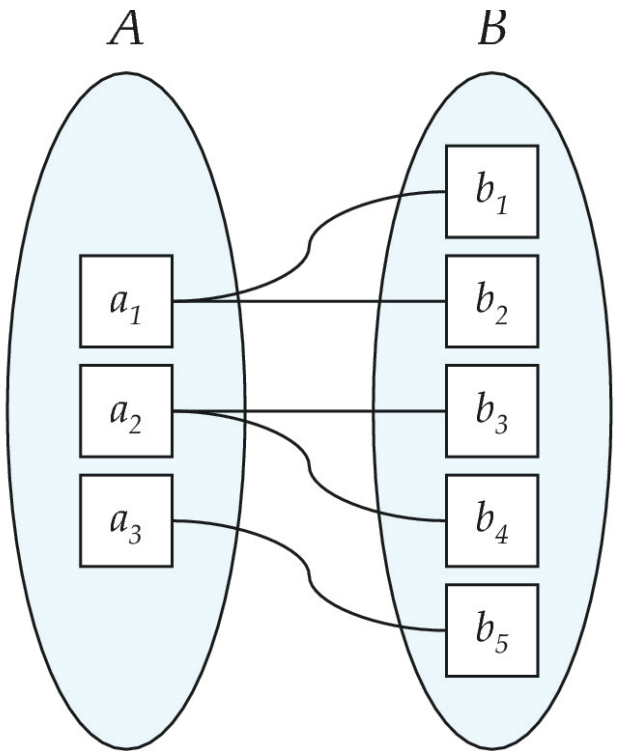
One to one



An entity in *A* is associated with *at most* one entity in *B*, and an entity in *B* is associated with *at most* one entity in *A*.

Cardinality constraints

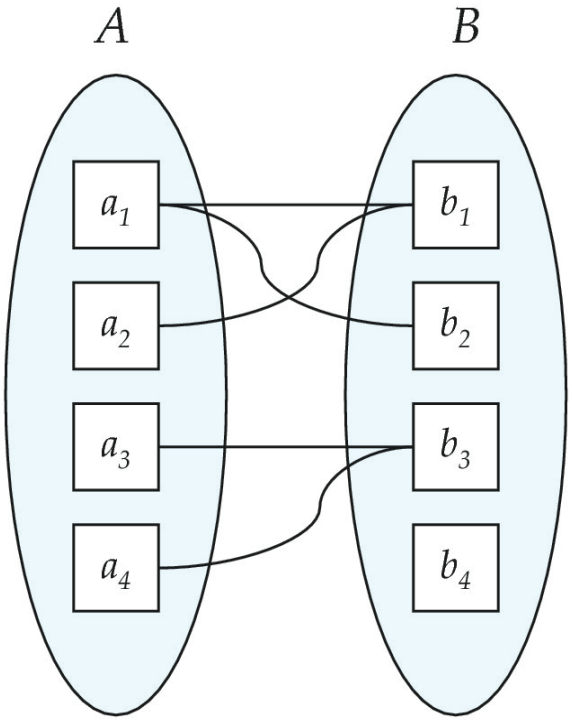
One to many



An entity in A is associated with any number (zero or more) of entities in B. An entity in B, however, can be associated with at most one entity in A.

Cardinality constraints

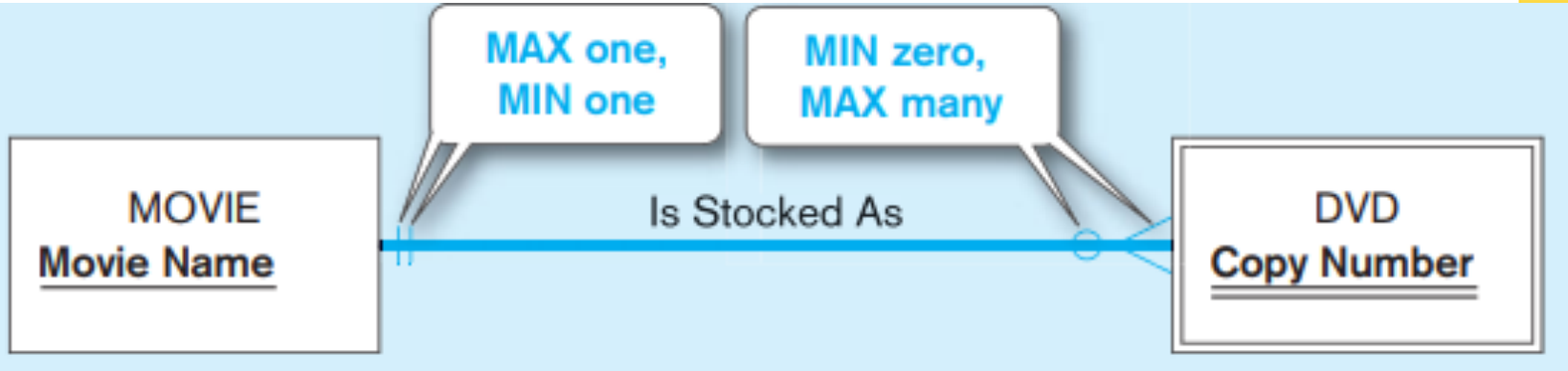
Many to many



An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

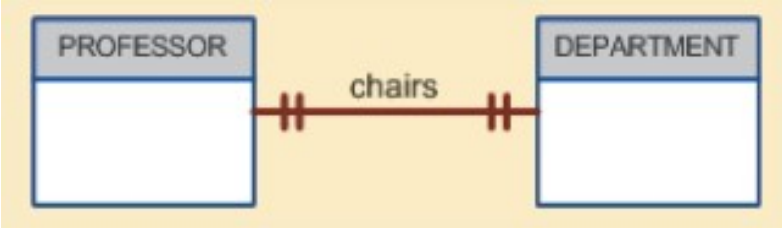
Cardinality constraints

- Minimum Cardinality
  - is the minimum number of instances of entity B that may be associated with each instance of entity A
- Maximum Cardinality
  - is the maximum number of instances of entity B that may be associated with each instance of entity A
- Ex:

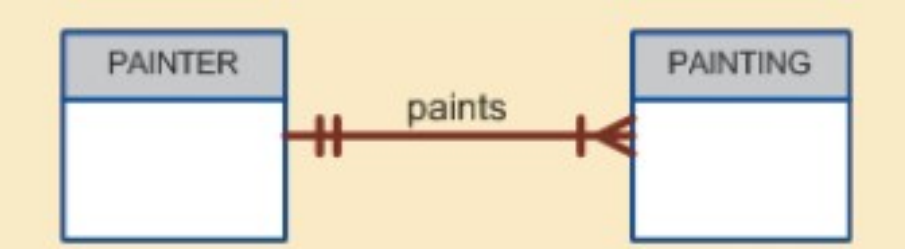


Example of Cardinality constraints

- One-to-One relationship
  - one department chair—a professor—can chair only one department, and one department can have only one department chair.

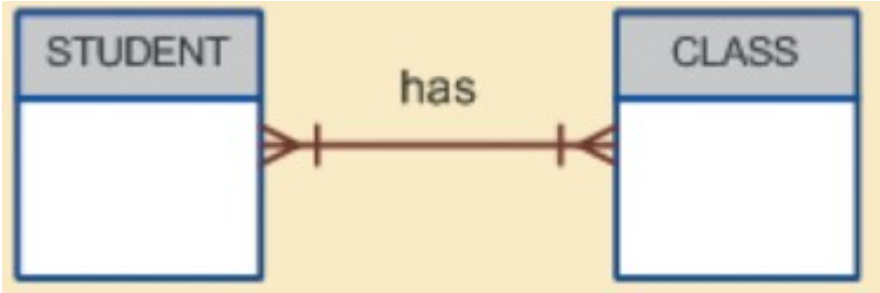


- One-to-Many relationship
  - Each painting is painted by one and only one painter, but each painter could have many paintings.



Example of Mapping Cardinalities

- Many-to-Many relationship
  - Each CLASS can have many STUDENTs, and each STUDENT can take many CLASSES.

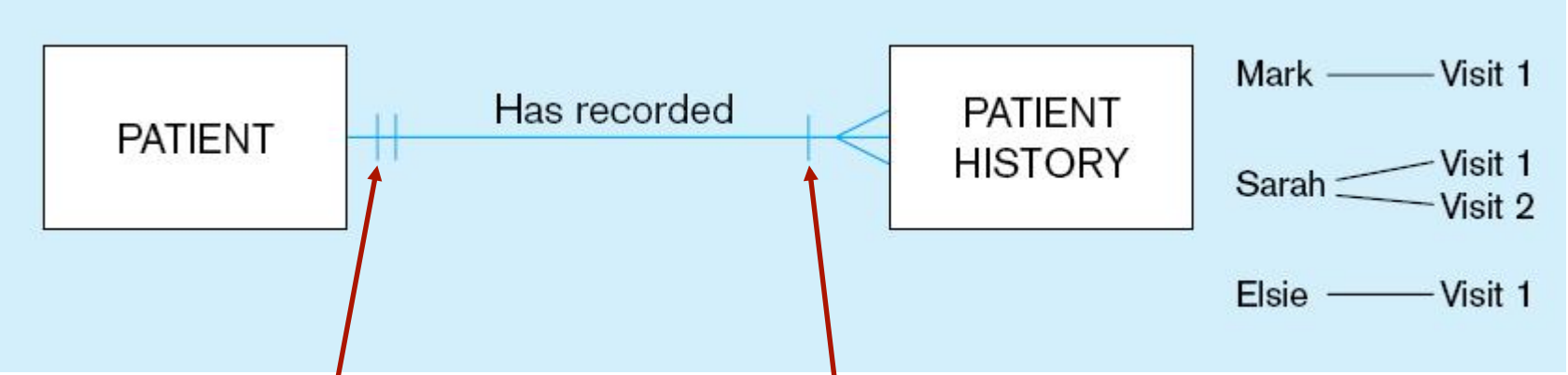


Optional and Mandatory Participation

- **Participation** determines whether all or only some entity occurrences participate in a relationship.
- Participation in a relationship can be **optional** (or **Partial**) or **mandatory** (or **Total**) participation.
  - Mandatory participation:
    - **all** entity occurrences are involved in a particular relationship.
    - *Minimum cardinality is one*
  - Optional participation:
    - **only some** entities may not participate in a particular relationship.
    - *Minimum cardinality is zero*

Optional and Mandatory Participation

- Ex1:

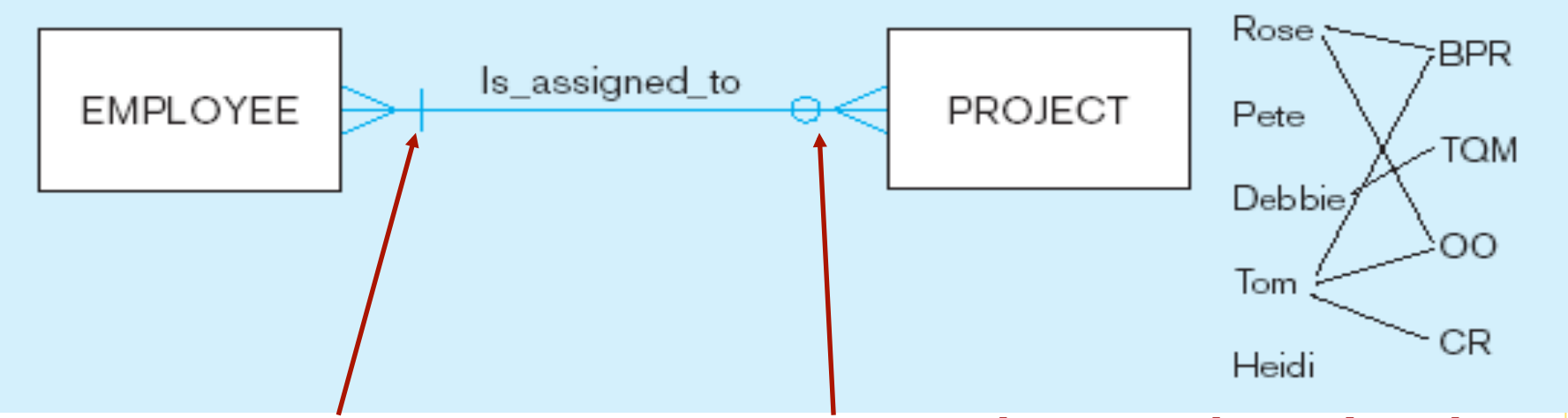


A patient history is recorded for one and only one patient

A patient must have recorded at least one history, and can have many

Optional and Mandatory Participation

- Ex2:



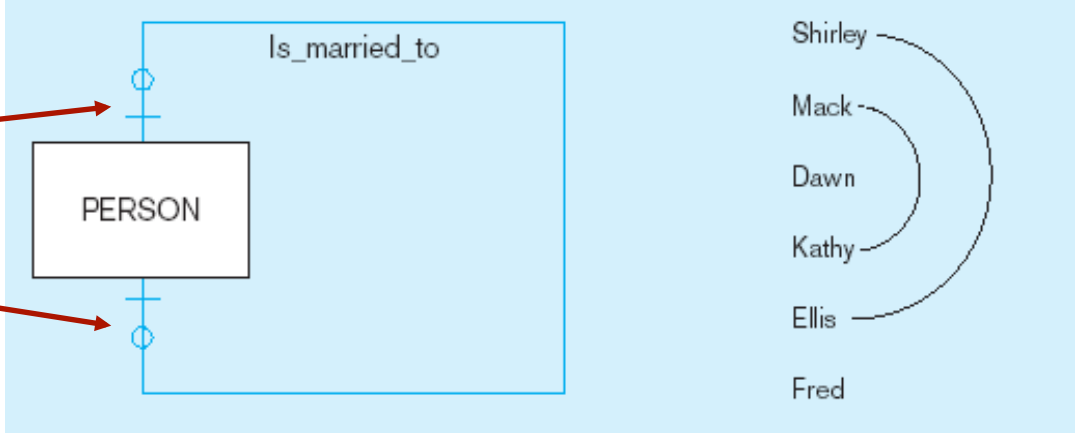
A project must be assigned to at least one employee, and may be assigned to many

An employee can be assigned to any number of projects, or may not be assigned to any at all

Optional and Mandatory Participation

- Ex2:

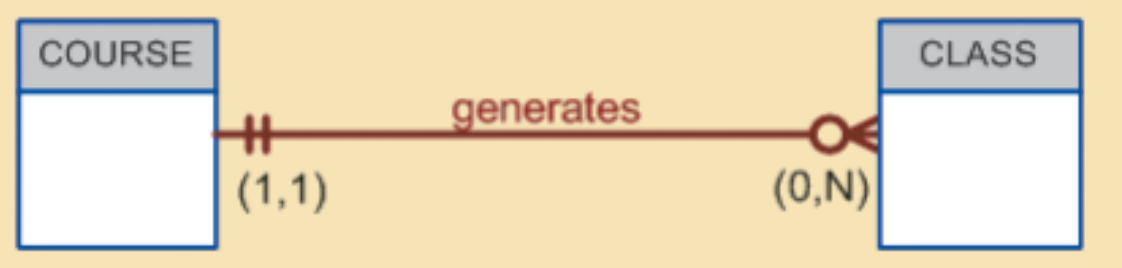
A person is married to at most one other person, or may not be married at all



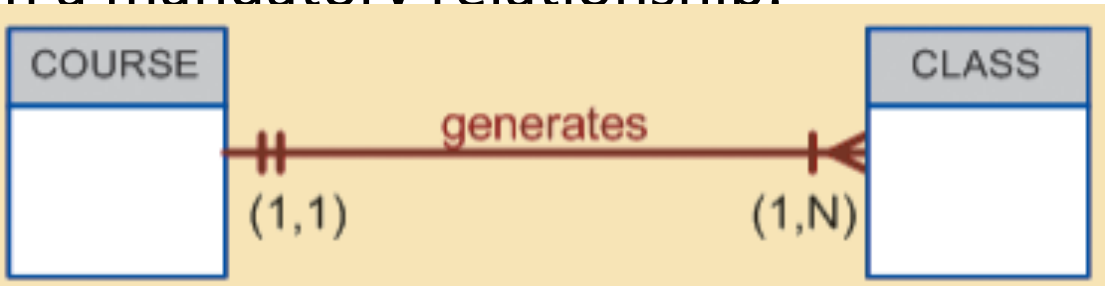
Optional and Mandatory Participation

- Another examples
  - An optional CLASS entity in the relationship "PROFESSOR teaches CLASS":

  - CLASS is optional to COURSE



  - COURSE and CLASS in a mandatory relationship:



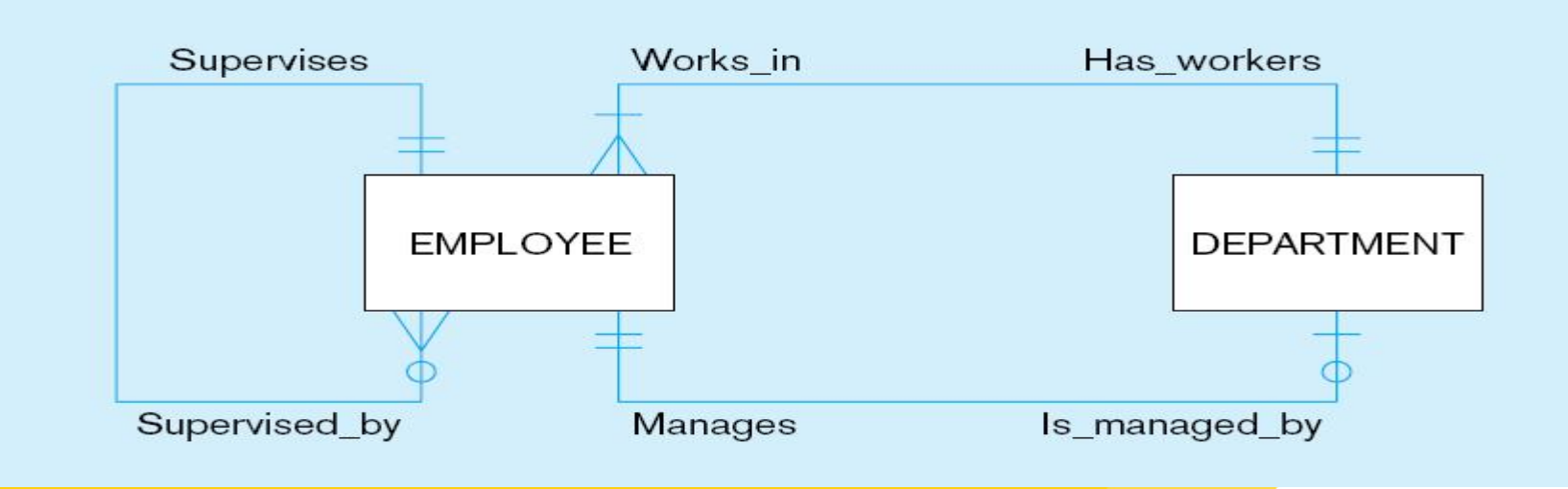
Optional and Mandatory Participation

- Example: Identifying relationship, cardinality between entities:

4. Customer and Order
5. Product and Category
6. Employee and Dependent
7. Employee and Department
8. Instructor and Class

Multiple Relationships Between Entity Types

- There may be more than one relationship between the same entity types in a given organization
- Ex: Employees and departments
  - One relationship associates employees with the department in which they work.
  - The second relationship associates each department with the employee who manages that department.

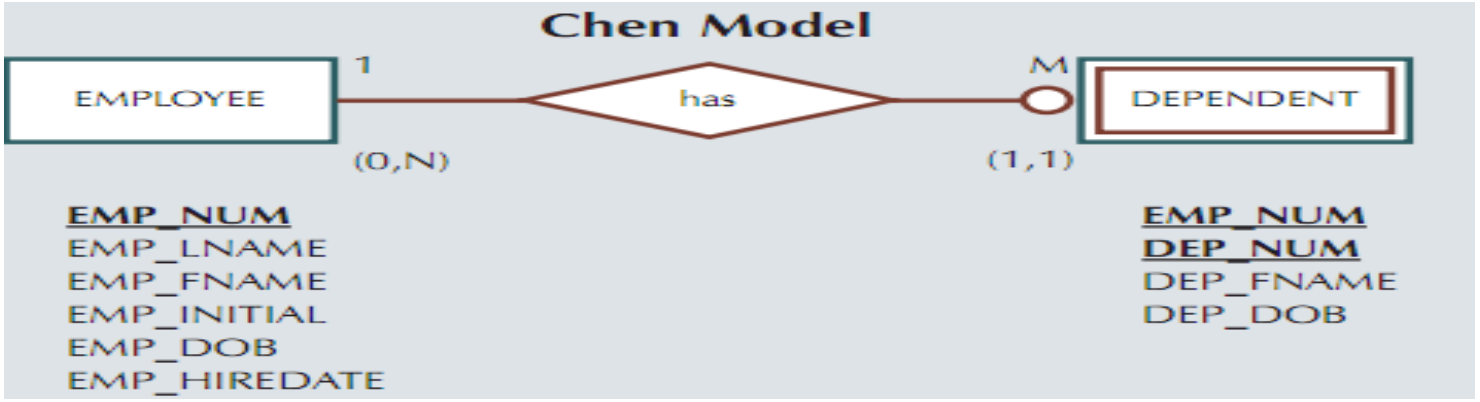


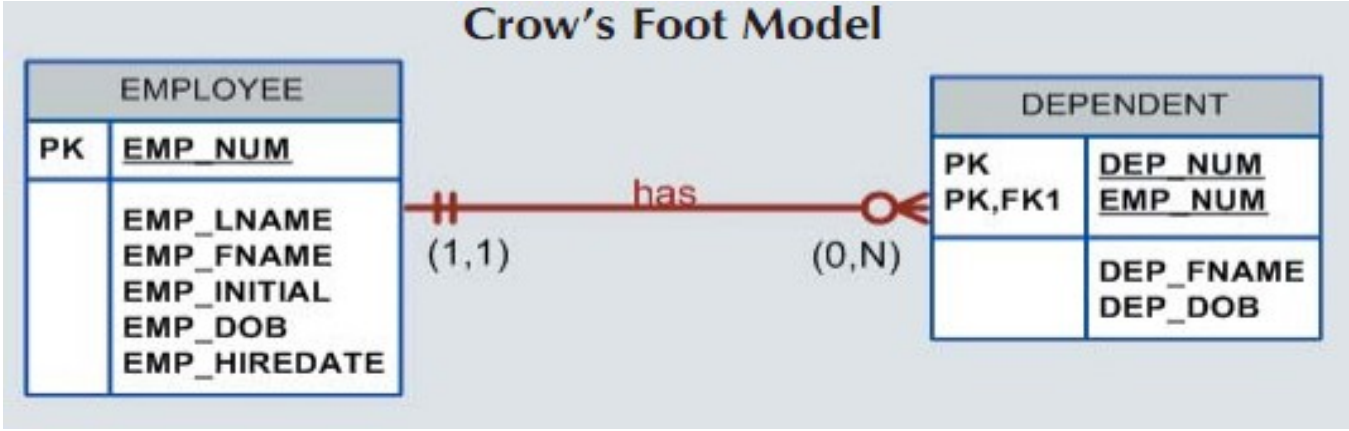
Weak Entity

- A **weak entity set** is an entity which is **existence-dependent;** that is, it cannot exist without the entity with which it has a relationship.
- An entity set that is not a weak entity set is termed a **strong entity set.**
- The weak entity has a primary key that is partially or totally derived from the parent entity (or strong entity) in the relationship.

Weak Entity

- Example





Constructing an ER model

Before beginning to draw the ER model, read the requirements specification carefully.

1. Identify entities
   - List all entity types (These are the object of interest in the system)
   - Remove duplicate entities (Also do not include the system as an entity type)
2. List the attributes of each entity
3. Mark the primary keys
   - Which attributes uniquely identify instances of that entity type?
   - This may not be possible for some weak entities.

Constructing an ER model

4. Define the relationships
   - Examine each entity type to see its relationship to the others.
5. Describe the cardinality and optionality of the relationships
   - Examine the constraints between participating entities.
6. Remove redundant relationships
   - Examine the ER model for redundant relationships.

❑ER modelling is iterative, so expect to draw several versions. Note that there is no one right answer to the problem, but some solutions are better than others!

Exercises

Ex1: Company organized into DEPARTMENT.

- An university has several departments. Each department employs many employees, but each employee works in one department only.
- An employee can supervise many other employees, but an employee may have only one supervisor.
- Each department offers many courses. A course can be a pre-requisite of many other courses, but a course may have only one pre-requisite.

Exercises

Ex2: Company organized into DEPARTMENT.

- Each department has unique name and a particular employee who manages the department. Start date for the manager is recorded. Department may have several locations.
- A department controls a number of PROJECT. Projects have a unique name, number and a single location.
- Company's EMPLOYEE name, ssno, address, salary, sex and birth date are recorded. An employee is assigned to one department, but may work for several projects (not necessarily controlled by her dept). Number of hours/week an employee works on each project is recorded; The immediate supervisor for the employee.
- Employee's DEPENDENT are tracked for health insurance purposes (dependent name, birthdate, relationship to employee).

The Enhanced Entity Relationship Model (EERM)

- The term **enhanced entity relationship (EER) model (or the Extended entity relationship model)** is used to identify the model that has resulted from extending the original E-R model with these new modeling constructs.
- A Diagram using this model is called an EER diagram (EERD)
- The most important modeling construct incorporated in the EER model is **supertype/subtype** relationships discriminator

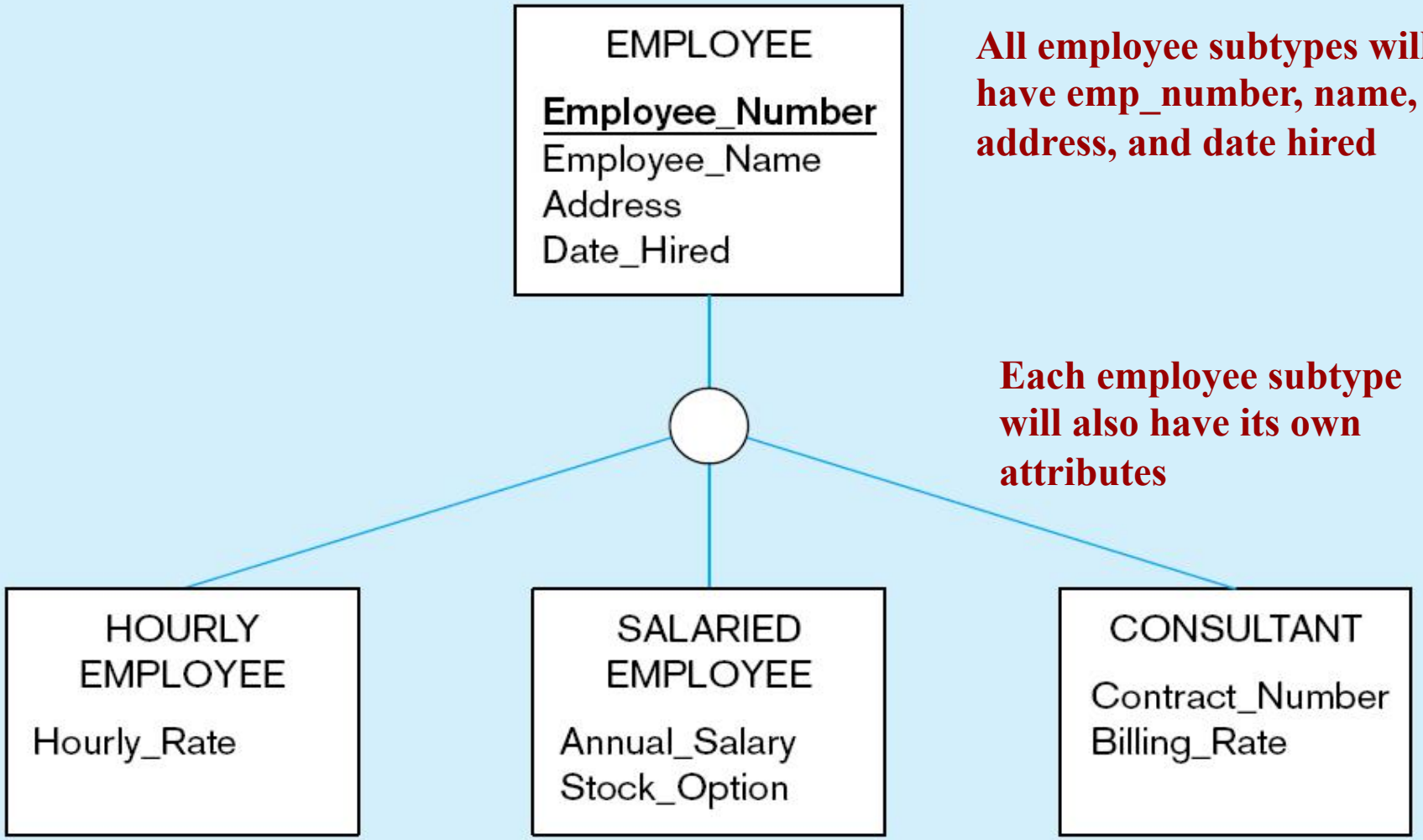
The Enhanced Entity Relationship Model (EERM)

- An entity supertype is a generic entity type that is related to one or more entity subtypes.
- The entity supertype contains common characteristics, and the entity subtypes each contain their own unique characteristics.

Supertype/Subtype relationship

- An example: Suppose that an organization has three basic types of employees: hourly employees, salaried employees, and contract consultants.
  - ***Hourly employees*** Employee Number, Employee Name, Address, Date Hired, Hourly Rate
  - ***Salaried employees*** Employee Number, Employee Name, Address, Date Hired, Annual Salary, Stock Option
  - ***Contract consultants*** Employee Number, Employee Name, Address, Date Hired, Contract Number, Billing Rate

Supertype/Subtype relationship



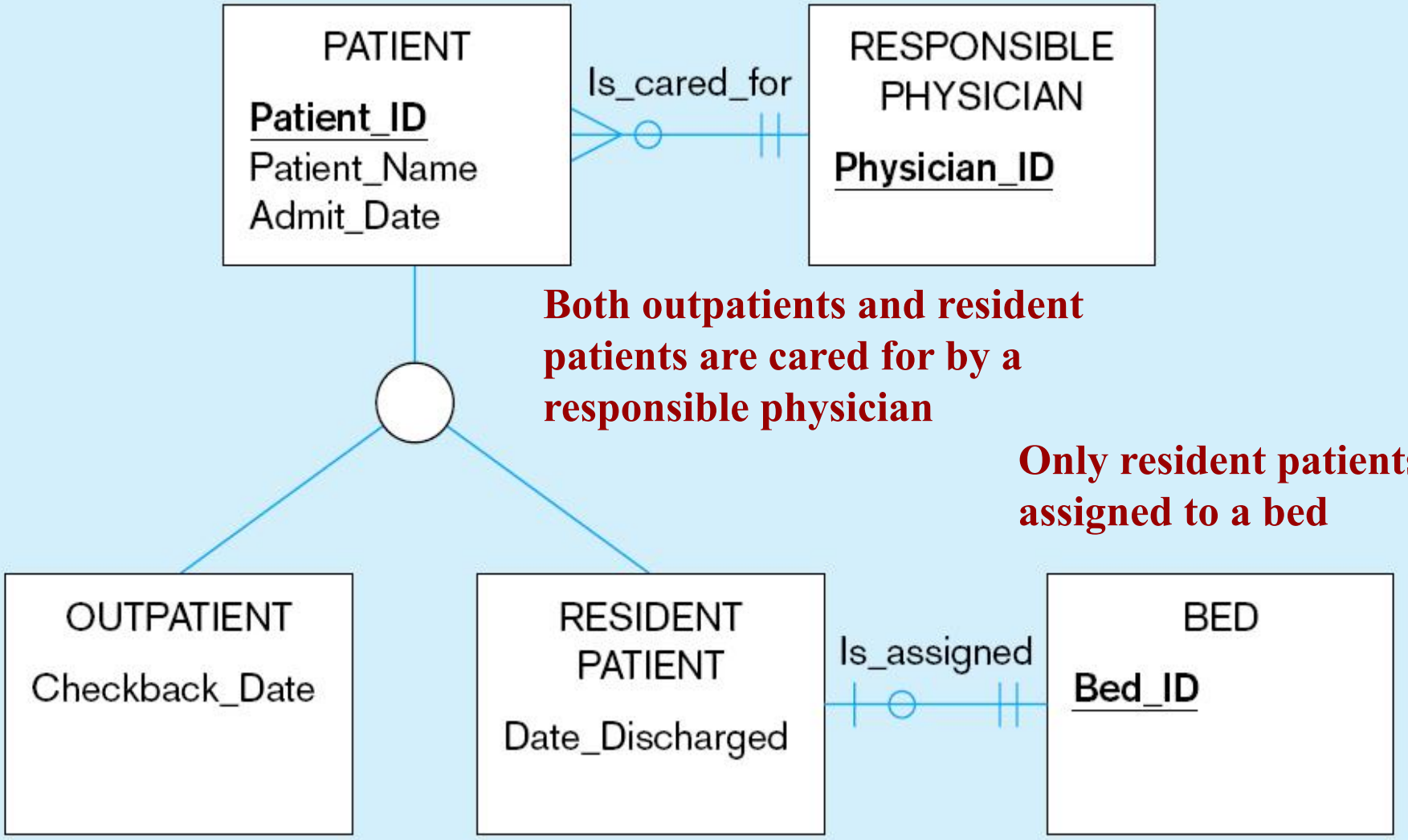
All employee subtypes will have emp_number, name, address, and date hired

Each employee subtype will also have its own attributes

Supertype/Subtype relationship

- The property of **inheritance** enables an entity subtype to inherit the attributes and relationships of the supertype.
- Relationships at the supertype level indicate that all subtypes will participate in the relationship.
- The instances of a subtype may participate in a relationship unique to that subtype. In this situation, the relationship is shown at the subtype level.

Example: Supertype/subtype relationships in a hospital

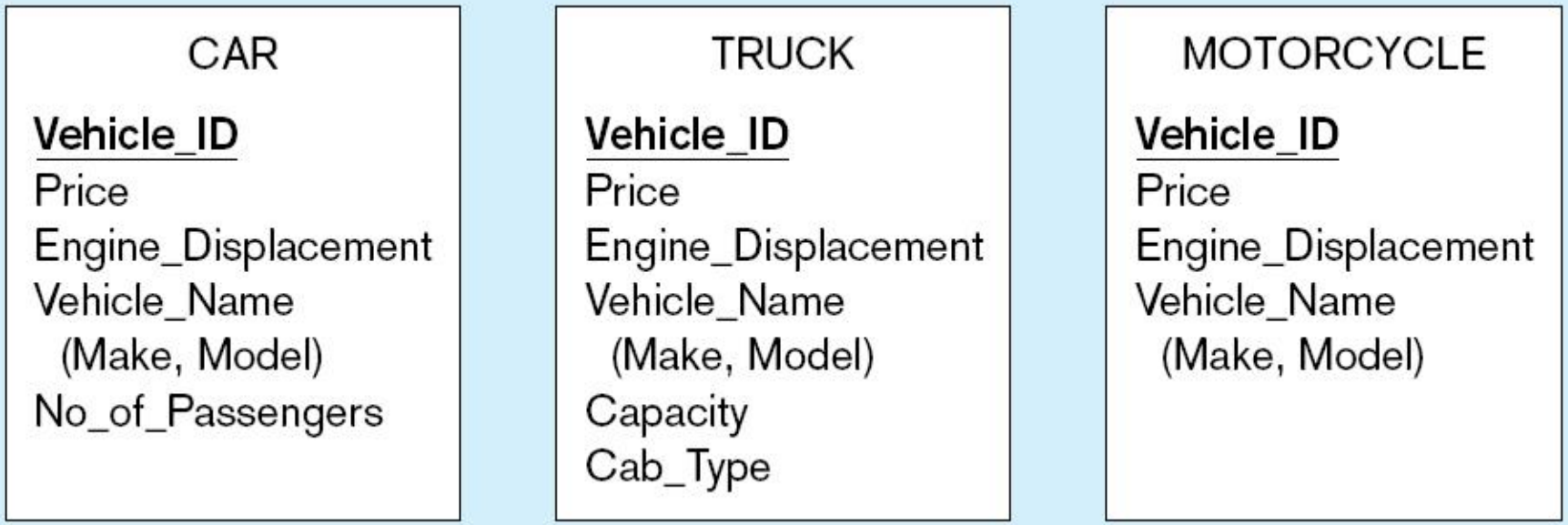


Generalization and Specialization

- **Generalization**:
  - The process of defining a more general entity type from a set of more specialized entity types.
  - BOTTOM-UP process
  - Based on grouping common characteristics and relationships of the subtypes
- **Specialization**:
  - The process of defining one or more subtypes of the supertype and forming supertype/subtype relationships.
  - TOP-DOWN process
  - Based on grouping unique characteristics and relationships of the subtypes

Example of generalization

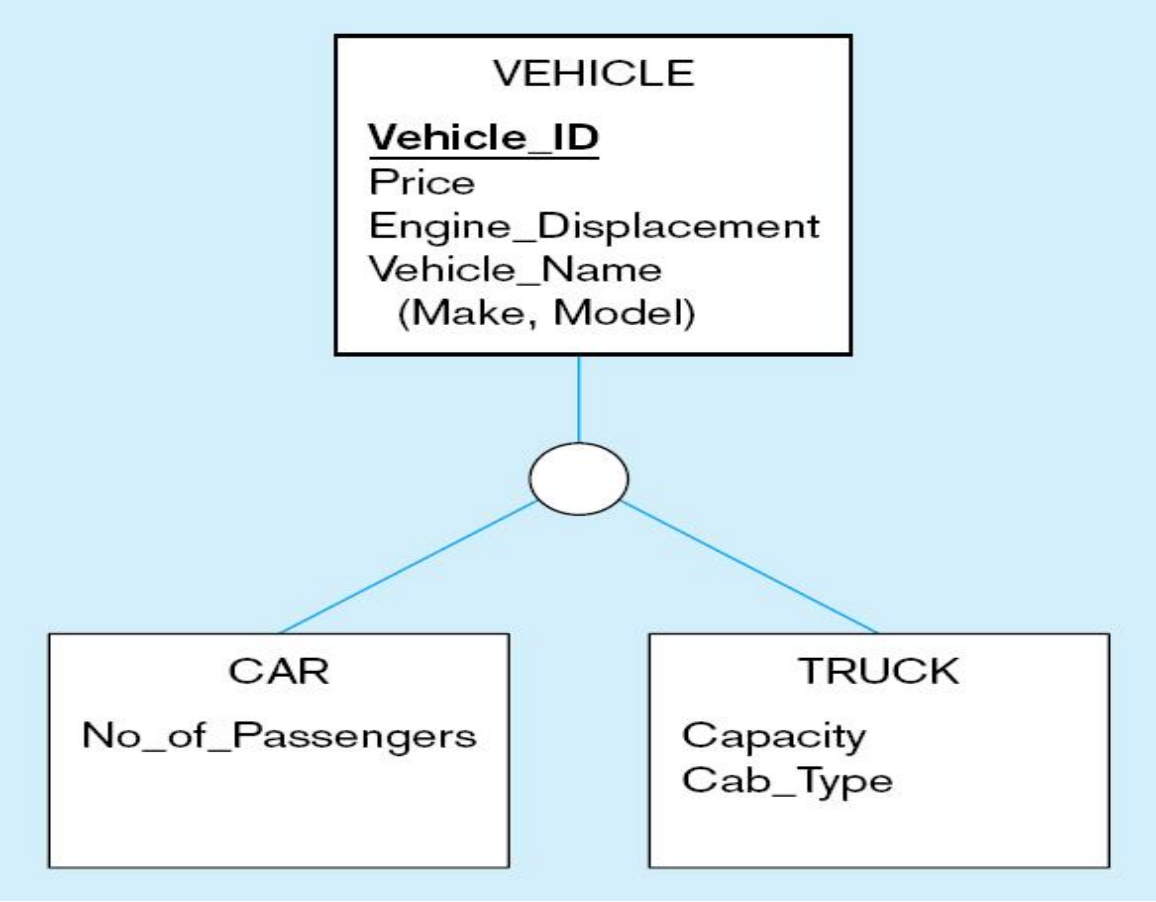
a) Three entity types: CAR, TRUCK, and MOTORCYCLE



All these types of vehicles have common attributes

Example of generalization(cont.)

b) Generalization to VEHICLE supertype

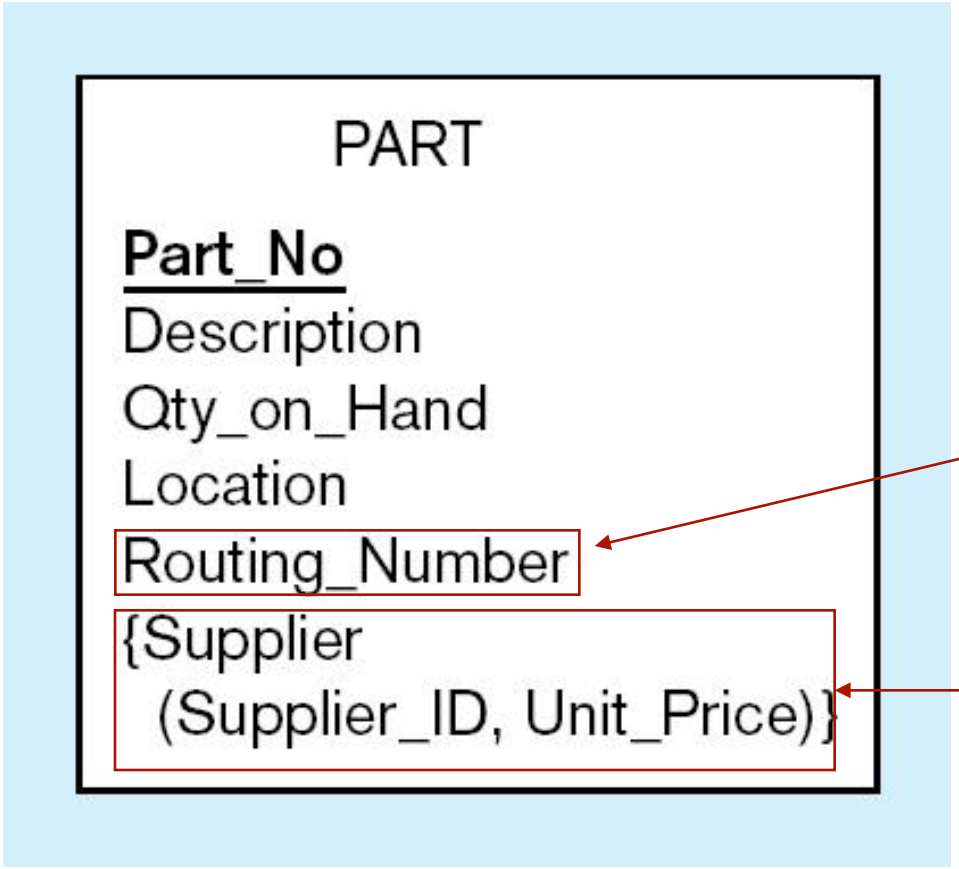


So we put the shared attributes in a supertype

Note: no subtype for motorcycle, since it has no unique attributes

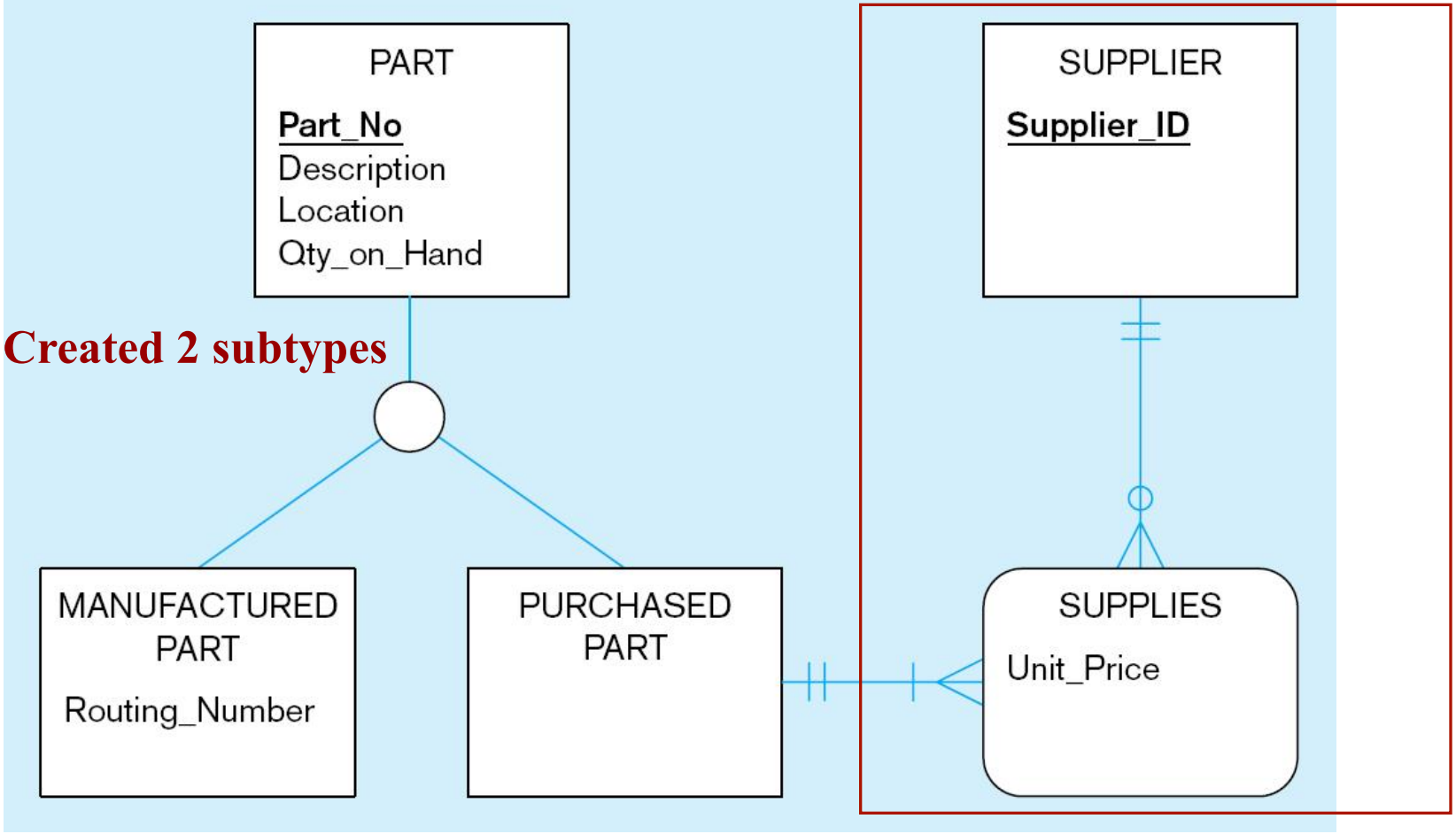
Example of specialization

a) Entity type PART



Example of specialization (cont.)

b) Specialization to MANUFACTURED PART and PURCHASED PART



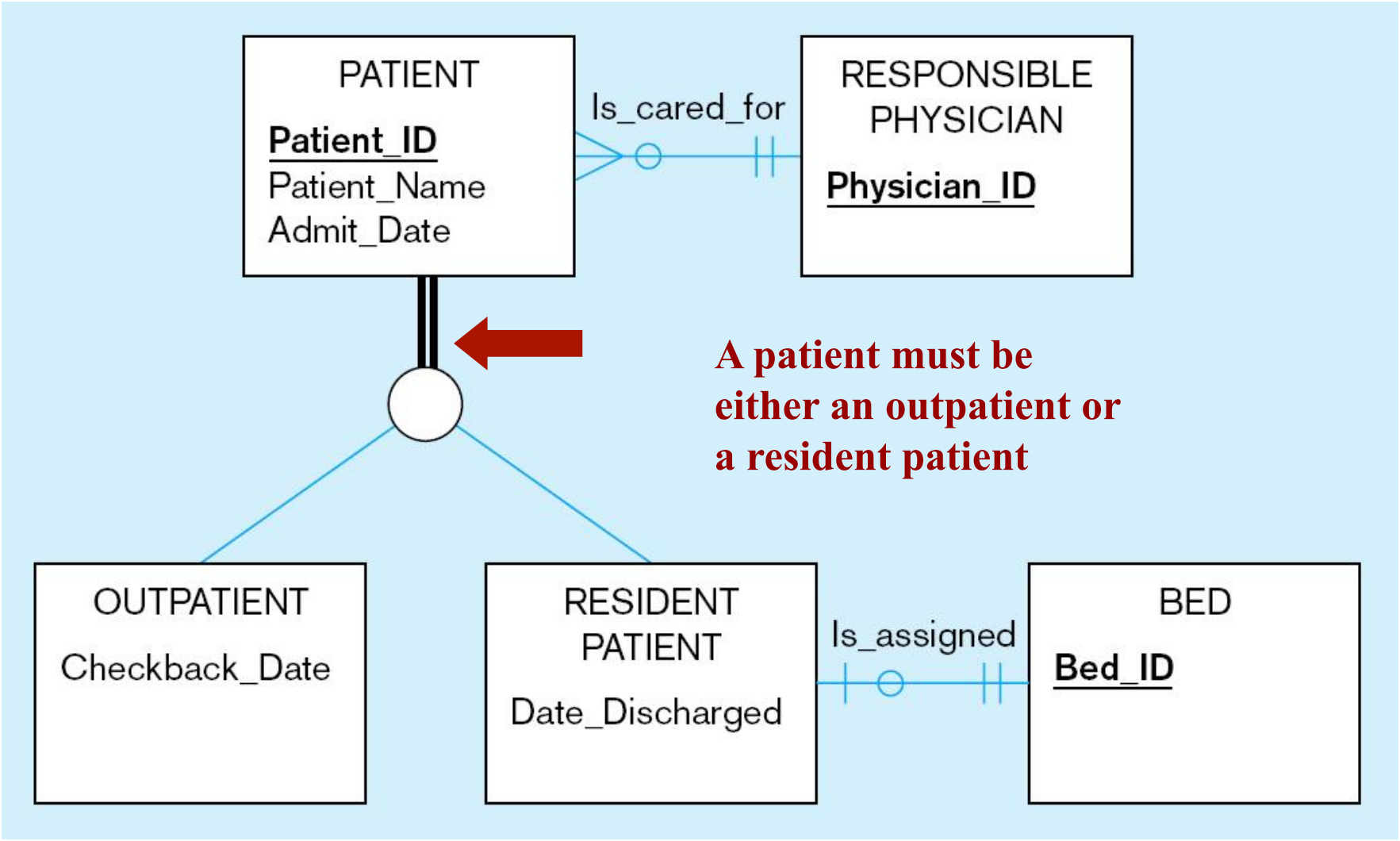
Note: multivalued attribute was replaced by an associative entity relationship to another entity

Completeness Constraint

- The completeness constraint specifies whether each entity supertype occurrence must also be a member of at least one subtype.
- The completeness constraint can be partial or total.
  - **Partial completeness**
    - Symbolized by a circle over a single line
    - Some supertype occurrences are not members
  - **Total completeness**
    - Symbolized by a circle over a double line
    - Every supertype occurrence must be member of at least one subtype

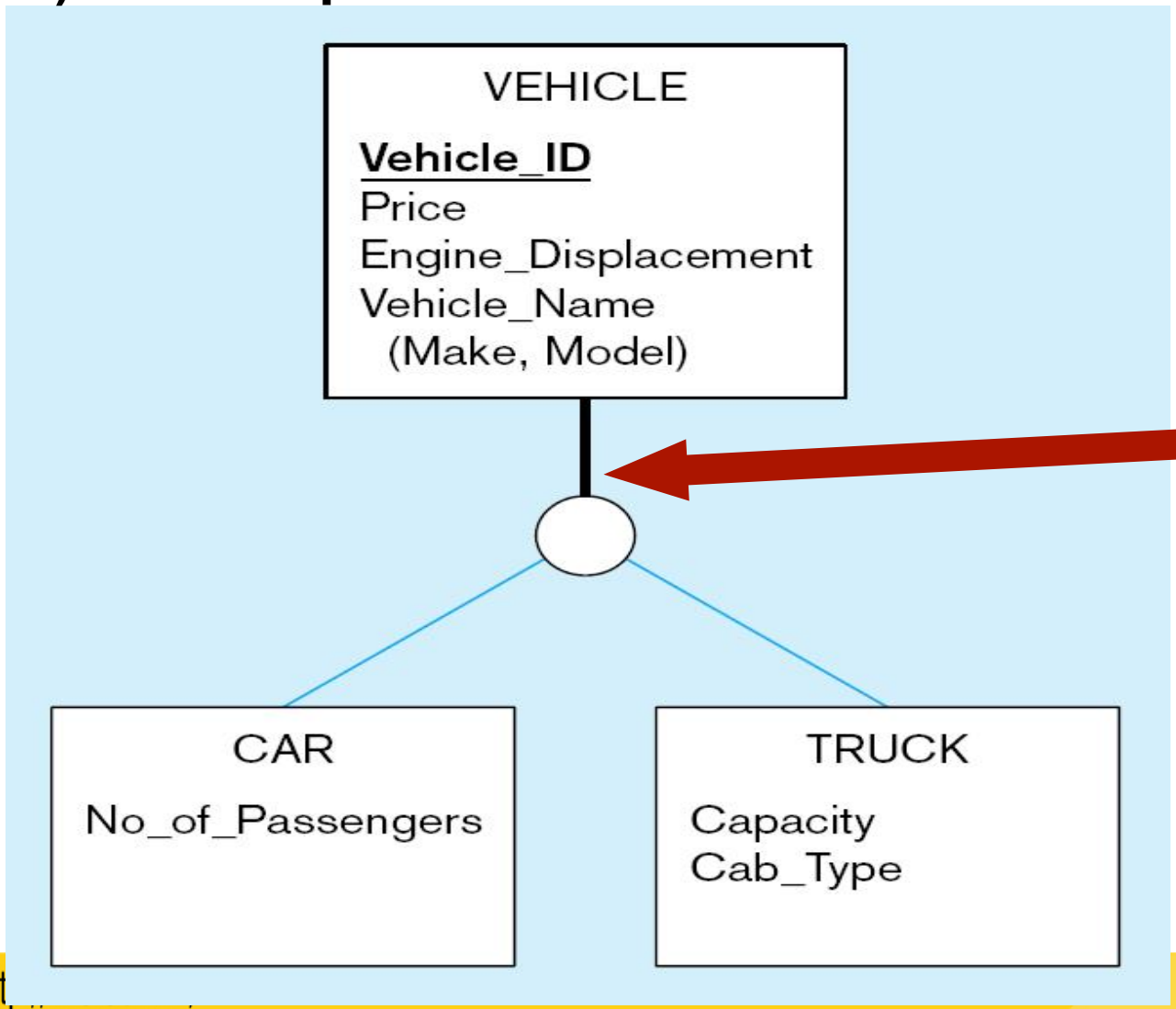
Examples of completeness constraints

a) Total specialization rule



Examples of completeness constraints (cont.)

b) Partial specialization rule



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

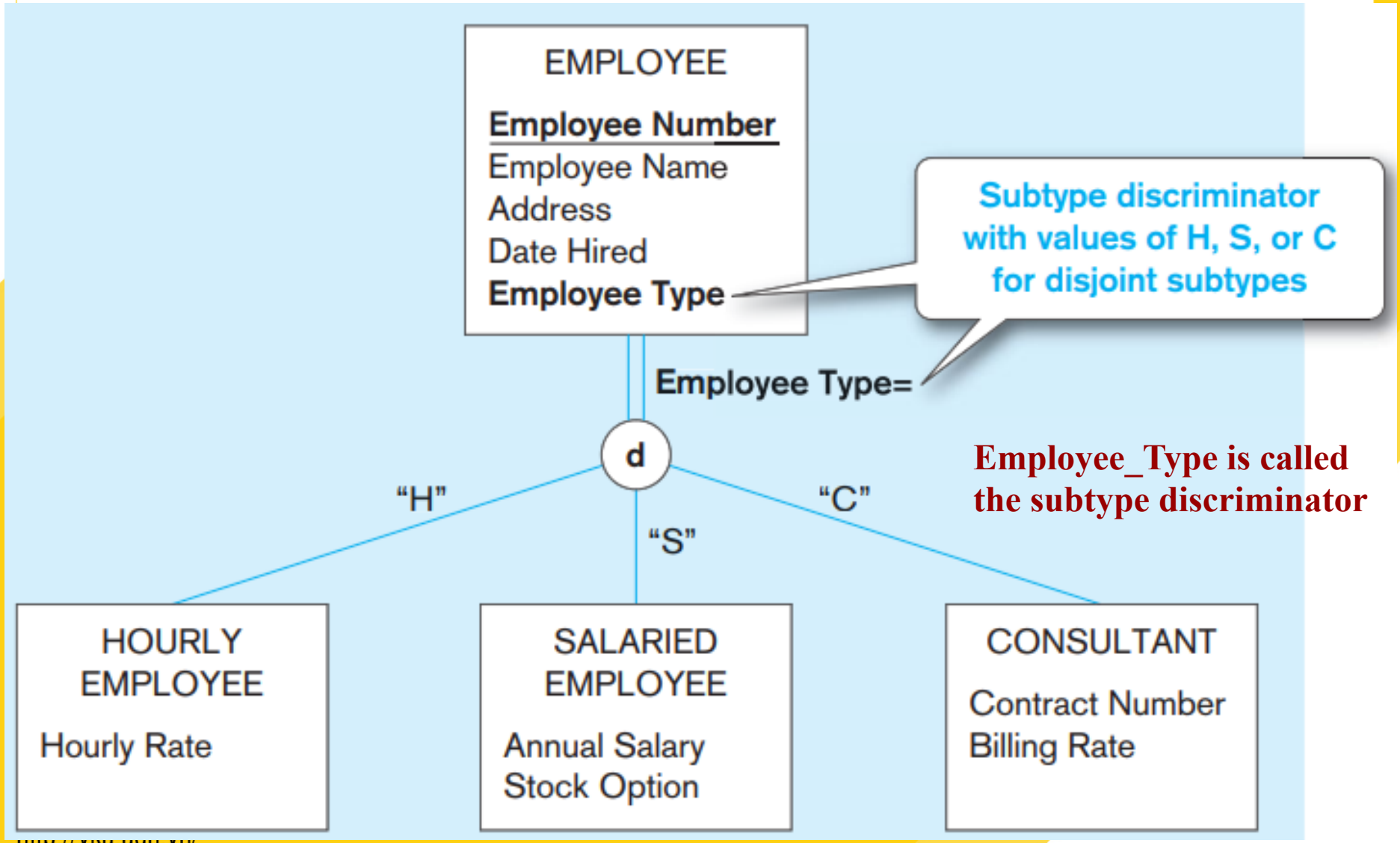
Subtype Discriminator

- An attribute in supertype entity whose value determines the target supertype(s)
- Default comparison condition for subtype discriminator attribute is equality comparison
- Subtype discriminator may be based on other comparison condition

Disjoint and Overlapping Constraints

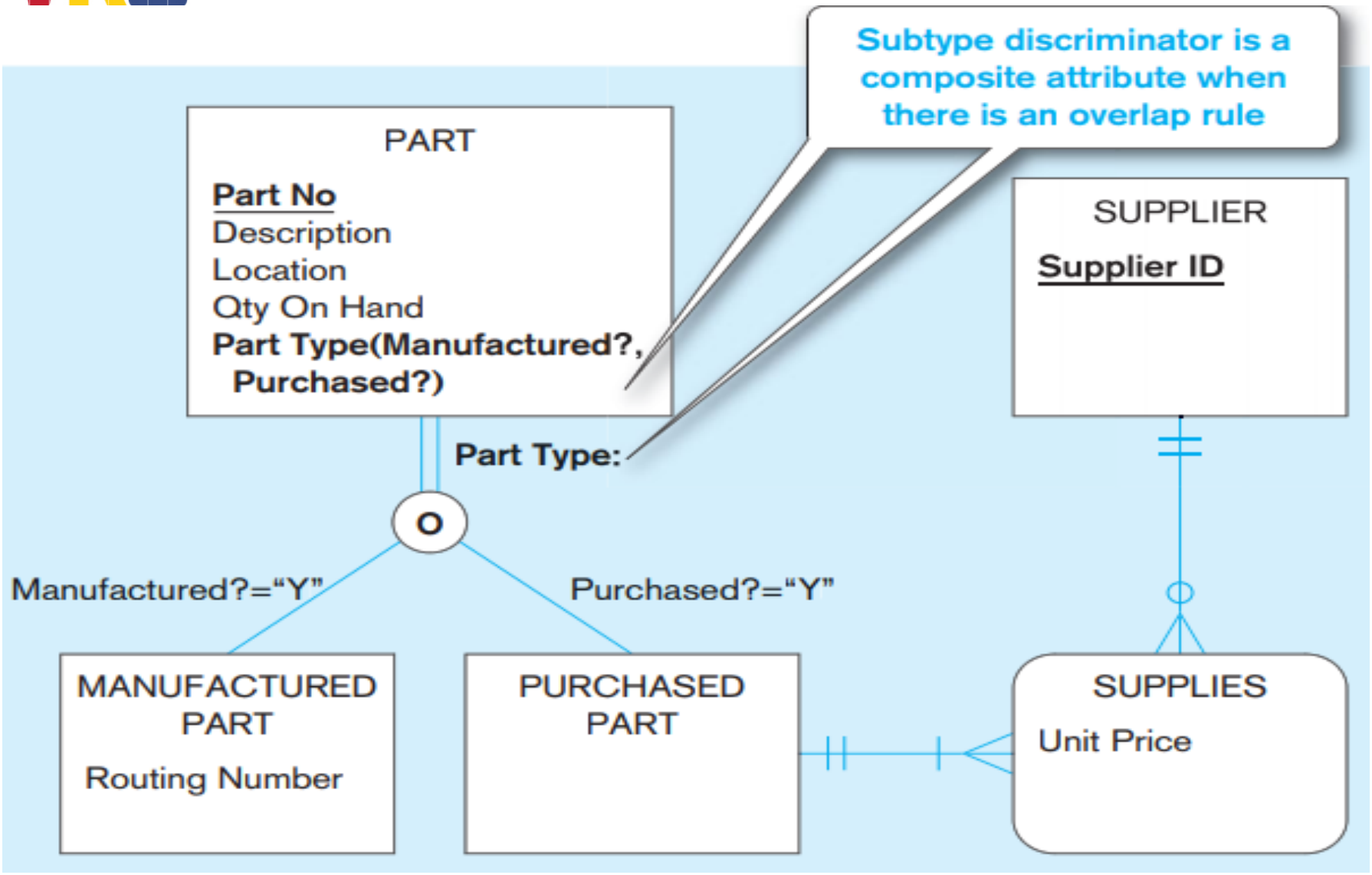
- Disjoint subtypes
  - Also called nonoverlapping subtypes
  - Subtypes that contain unique subset of supertype entity set, in other words, each entity instance of the supertype can appear in only one of the subtypes
  - In an ERD, disjoint subtypes are indicated by the letter **d** inside the category shape
- Overlapping subtypes
  - Subtypes that contain nonunique subsets of supertype entity set, that is, each entity instance of the supertype may appear in more than one subtype.
  - In an ERD, illustrates overlapping subtypes with the letter **o** inside the category shape.

Example: a subtype discriminator (*disjoint* rule)



Employee_Type is called the subtype discriminator

Example: Subtype discriminator (*overlap* rule)

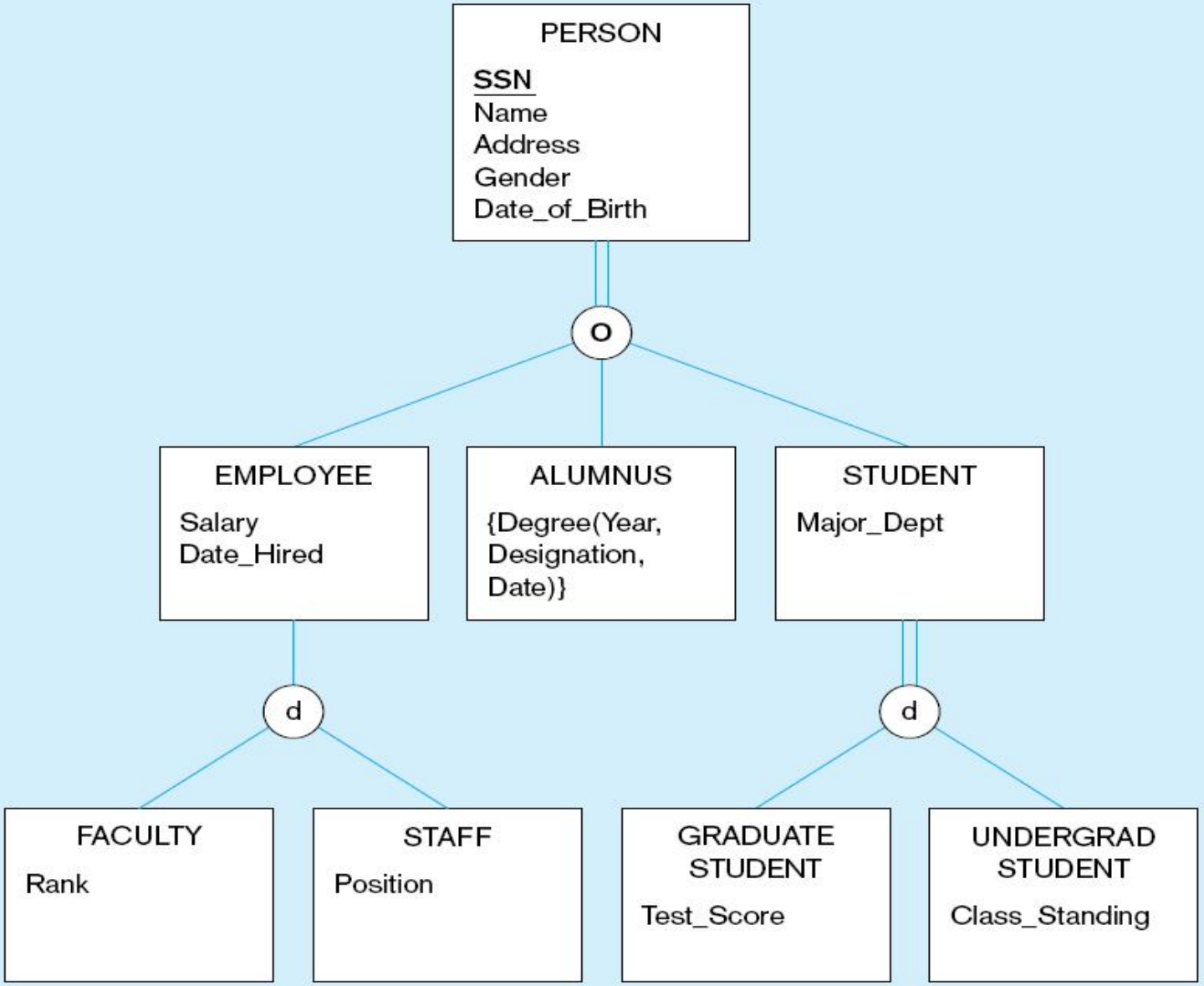


Example: Subtype discriminator (*overlap* rule)

- A new attribute named Part Type has been added to PART. Part Type is a composite attribute with components Manufactured? and Purchased?
- Each of these attributes is a Boolean variable (i.e., it takes on only the values yes, "Y," and no, "N"). When a new instance is added to PART, these components are coded as follows:

Type of Part	Manufactured?	Purchased?
Manufactured only	"Y"	"N"
Purchased only	"N"	"Y"
Purchased and manufactured	"Y"	"Y"

Example of supertype/subtype hierarchy



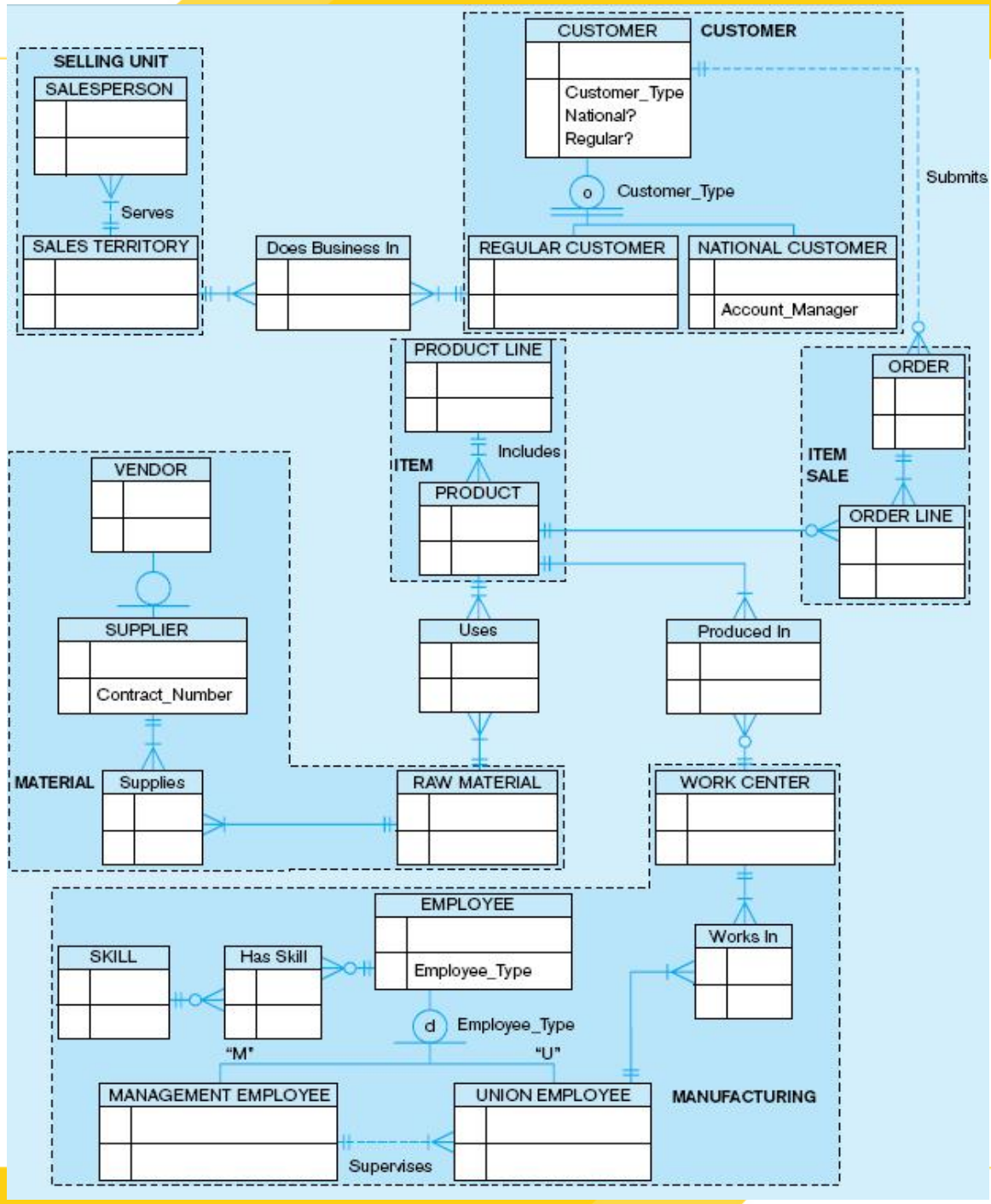
Entity Clusters

- EER diagrams are difficult to read when there are too many entities and relationships

=>Solution: Group entities and relationships into entity clusters

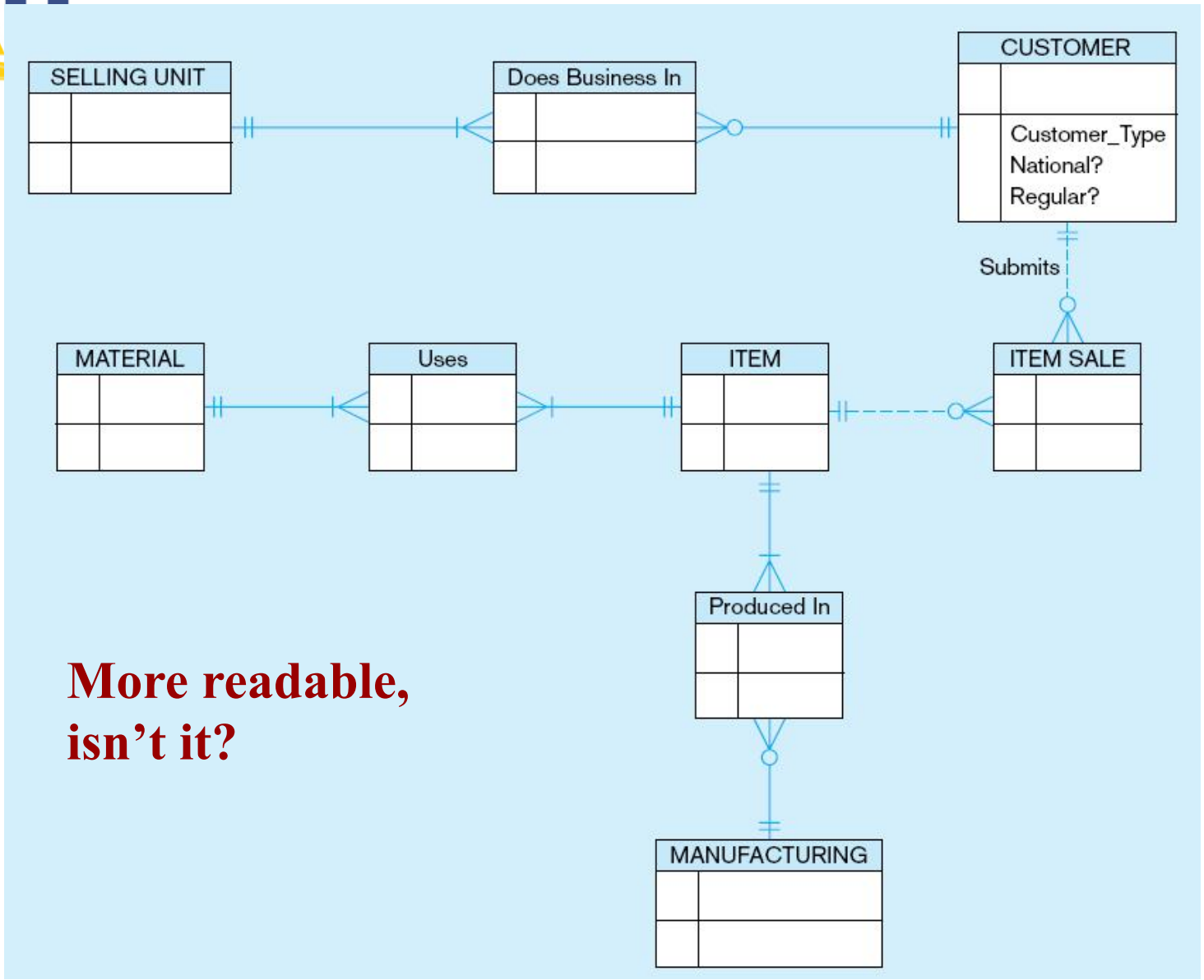
- **Entity cluster**: Set of one or more entity types and associated relationships grouped into a single abstract entity type

Possible entity clusters for Pine Valley Furniture in Microsoft Visio



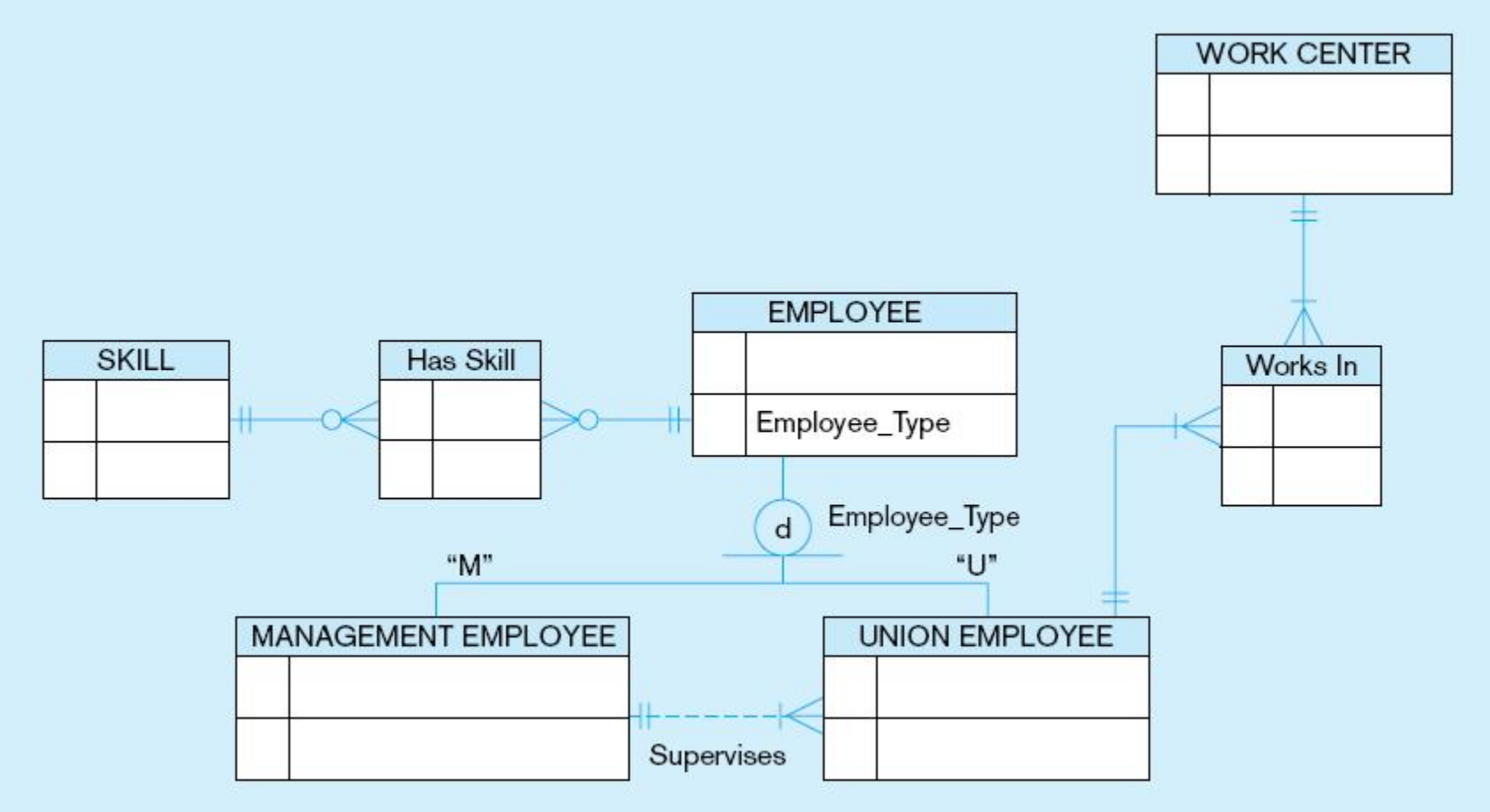
Related groups of entities could become clusters

EER diagram of PVF entity clusters



More readable, isn't it?

Manufacturing entity cluster



Detail for a single cluster

Exercise

- 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.
- 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.

Exercise

- 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.
- 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 under- graduates.

Thank You !