DATABASE SYSTEMS

Dr. Noha Nagy

Enhanced ER Diagram

EERM
ENHANCED ENTITY-RELATIONSHIP MODEL

Extended Entity-Relationship (EER) Model

- The entity relationship model in its original form did not support the specialization and generalization abstractions

EERM

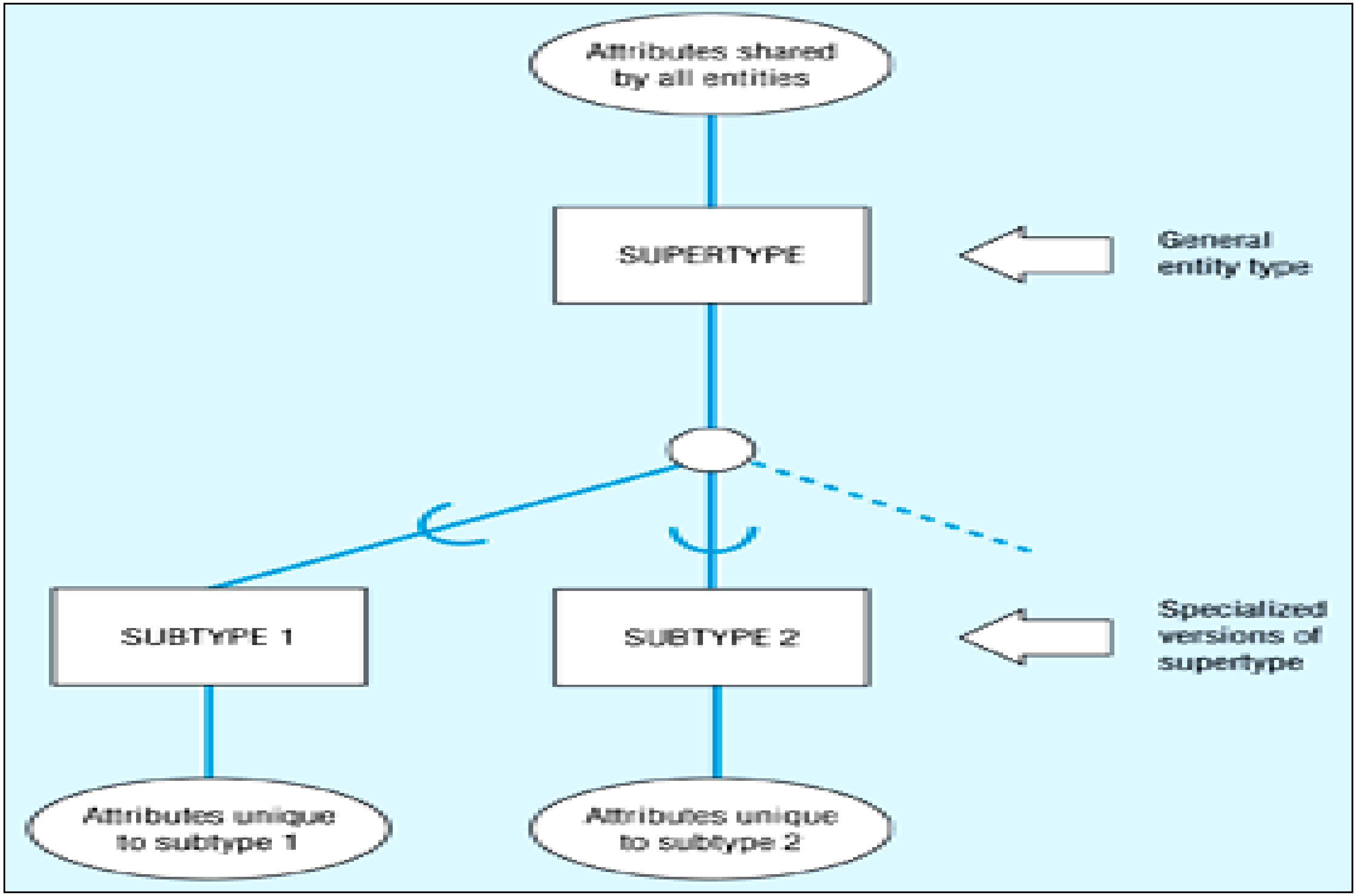
- Includes all modeling concepts of basic ERM
- Additional concepts:
  - Type-subtype and set-subset relationships
  - Specialization/Generalization Hierarchies
  - Categories and attribute Inheritance
- It is used to model applications more completely and accurately if needed
- It includes some object-oriented concepts, such as inheritance

Supertypes and Subtypes

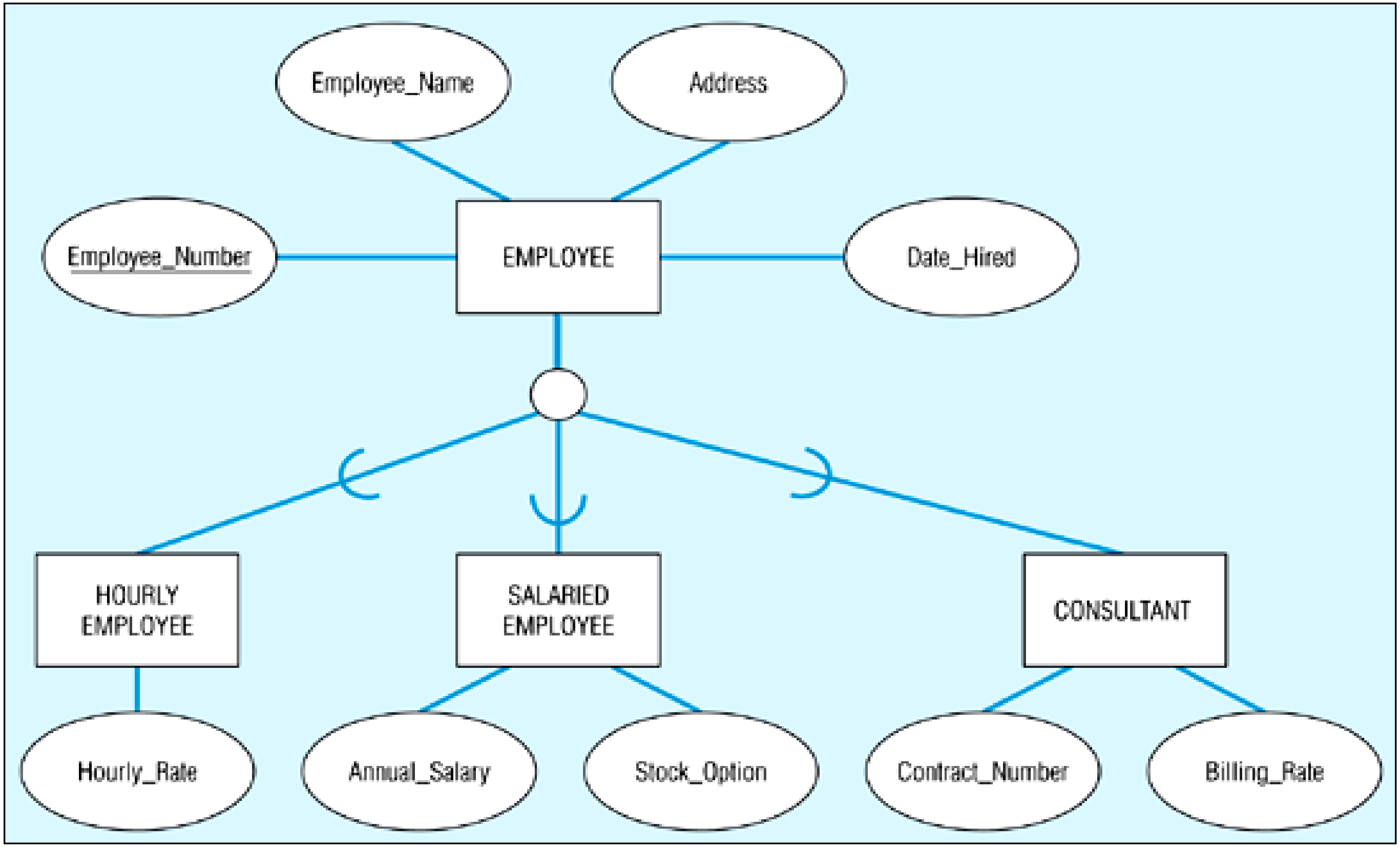
- Subtype: A subgrouping of the entities in an entity type which has attributes that are distinct from those in other subgroupings.

- Supertype: An entity type whose subtypes share common attributes. Attributes that are shared by all entities (including the identifier) are associated with the supertype.

Basic notation for supertype/subtype relationships


Attributes shared by all entities
SUPERTYPE
General entity type
SUBTYPE 1
SUBTYPE 2
Specialized versions of supertype
Attributes unique to subtype 1
Attributes unique to subtype 2

Employee supertype with three subtypes

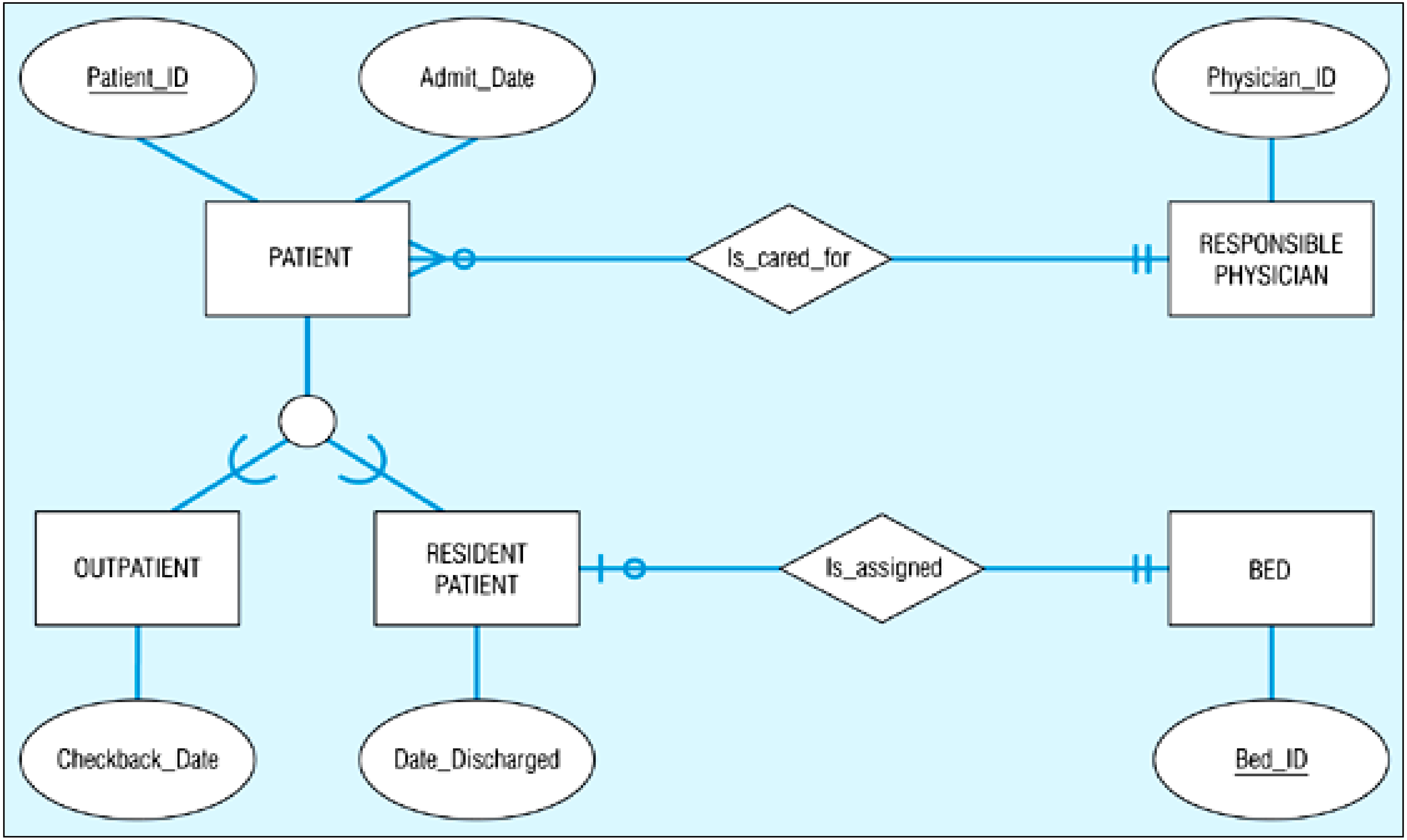


Attribute Inheritance

- Subtype entities inherit values of all attributes of the supertype.

  An occurrence of a subtype is also an occurrence of the supertype.
- Use of Supertype/Subtype
  - There are attributes that apply to some (but not all) of the instances of an entity type.
  - The instances of a subtype participate in a relationship unique to that subtype.

Supertype/subtype relationships in a hospital

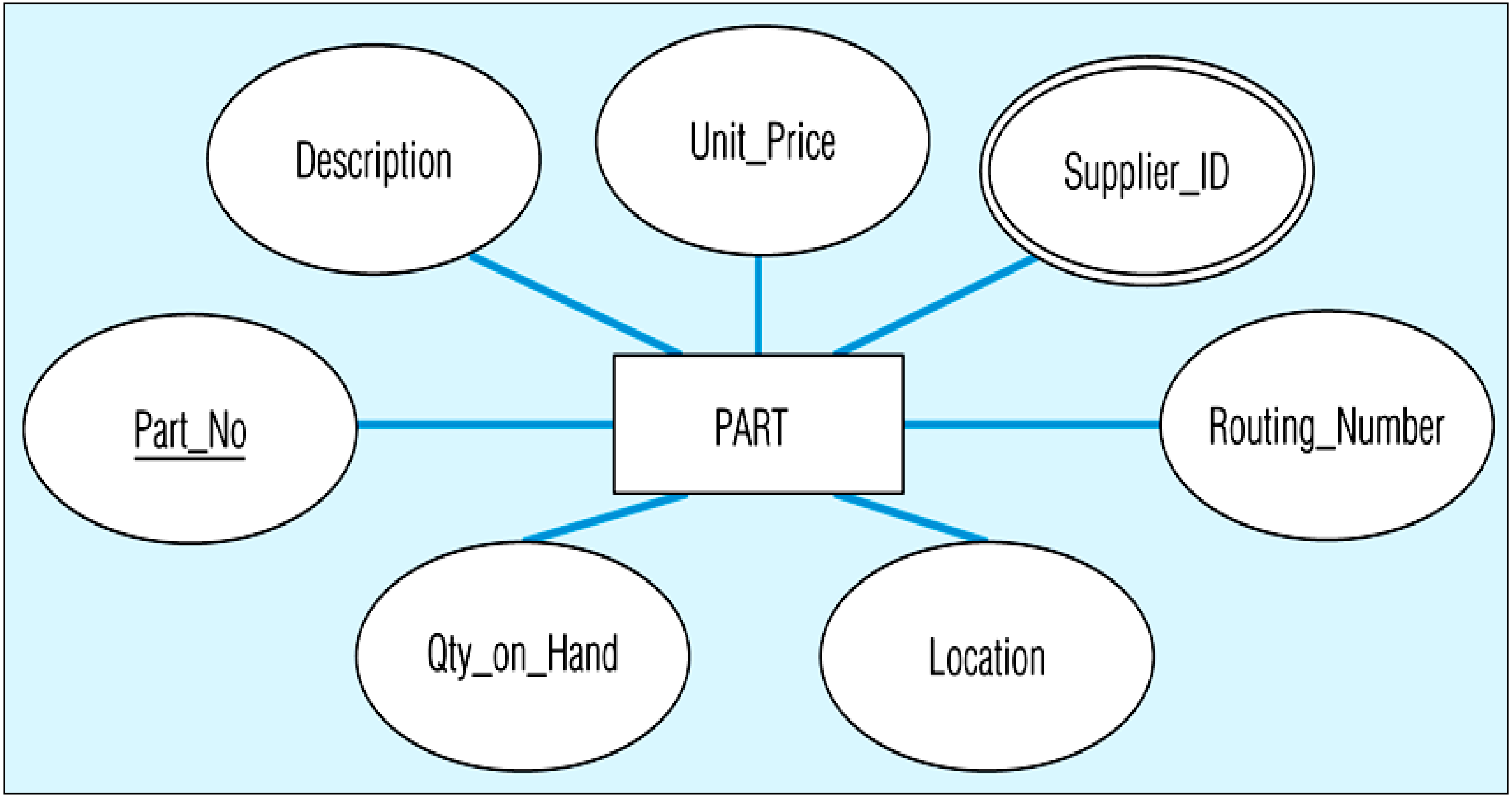


Specialization

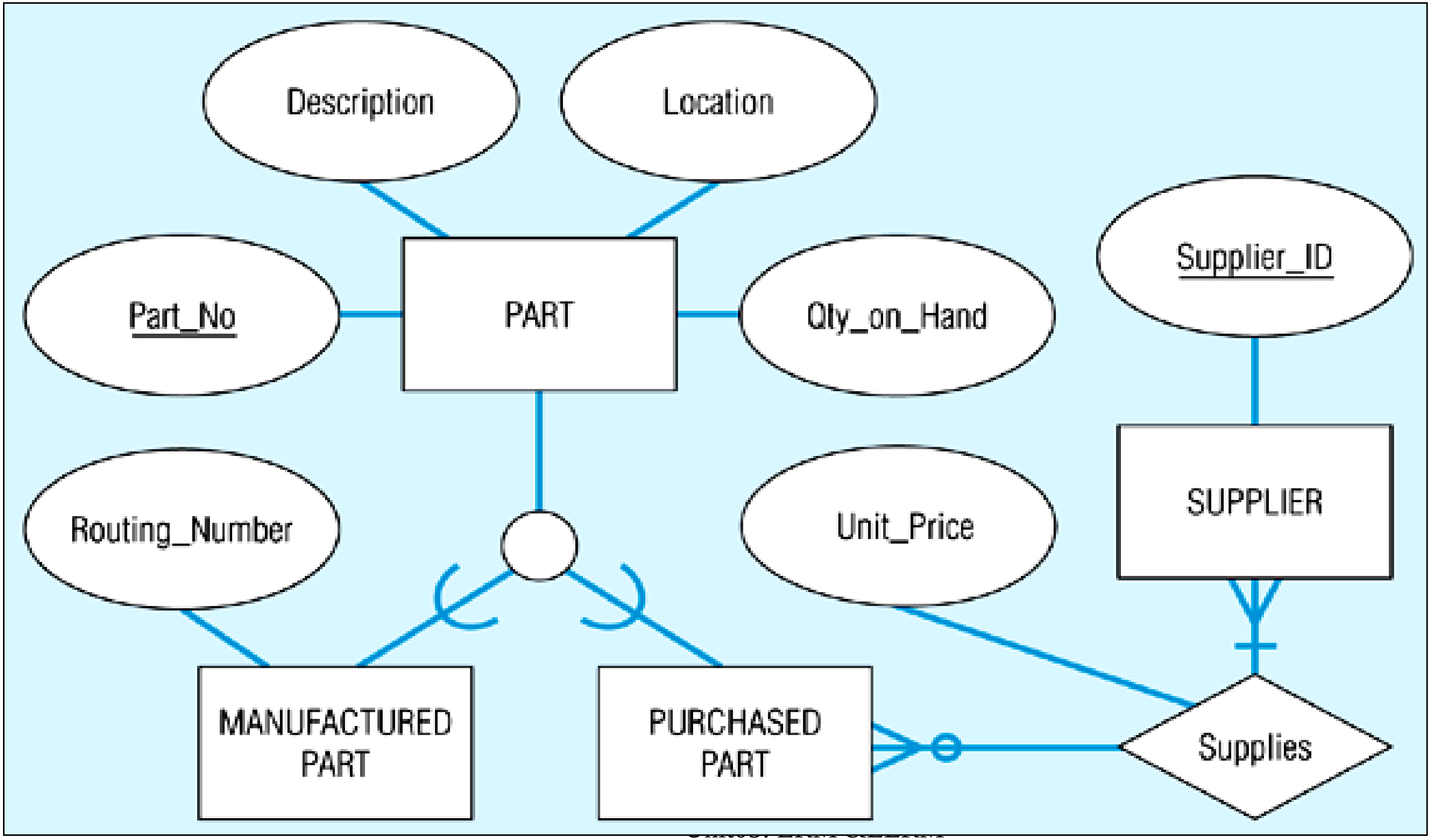
- Is the process of defining a set of subclasses of a superclass.
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass.
- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type*.
  - May have several specializations of the same superclass.

Example of specialization

(a) Entity type PART



(b) Specialization to MANUFACTURED PART and PURCHASED PART

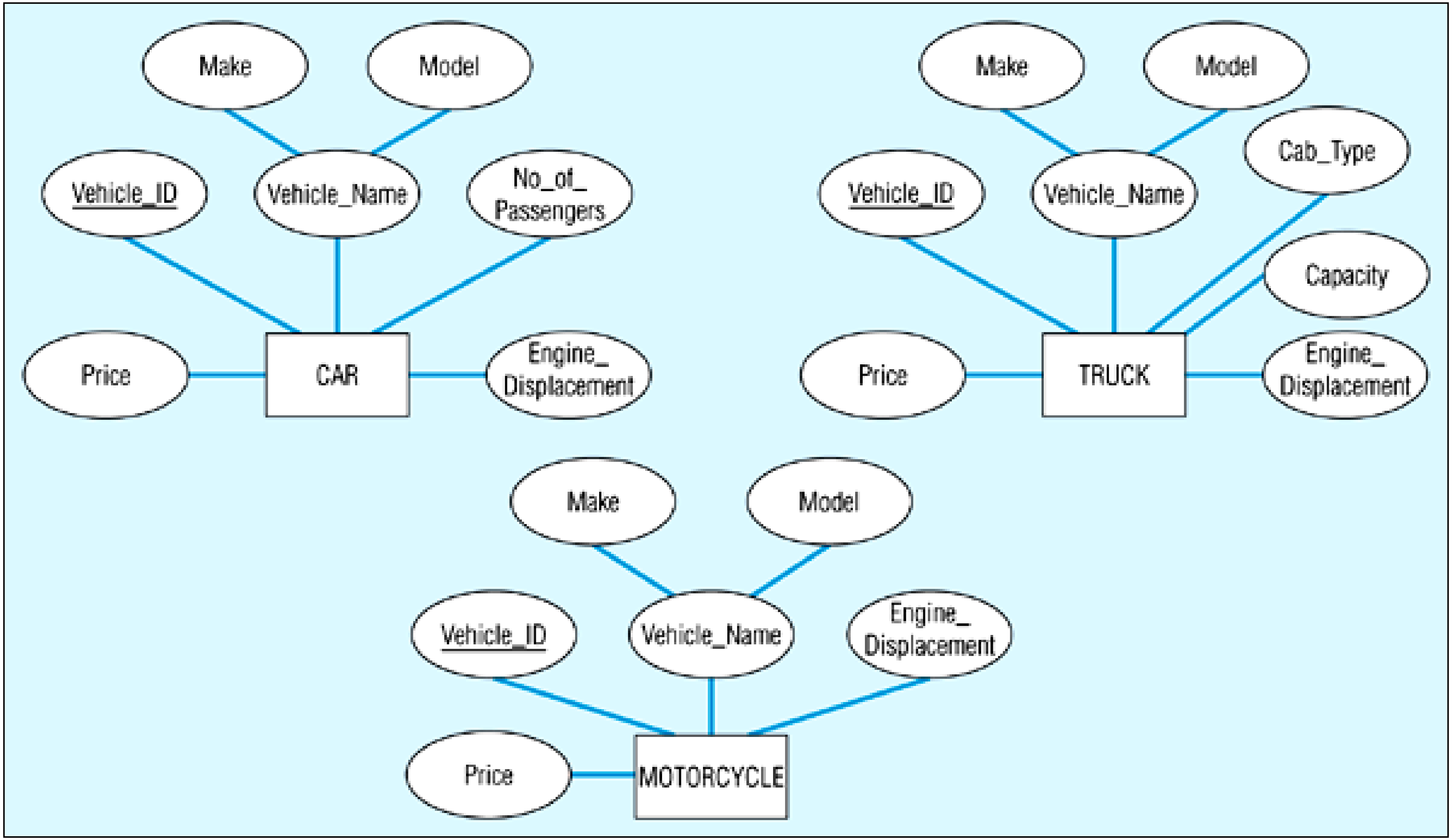


Generalization

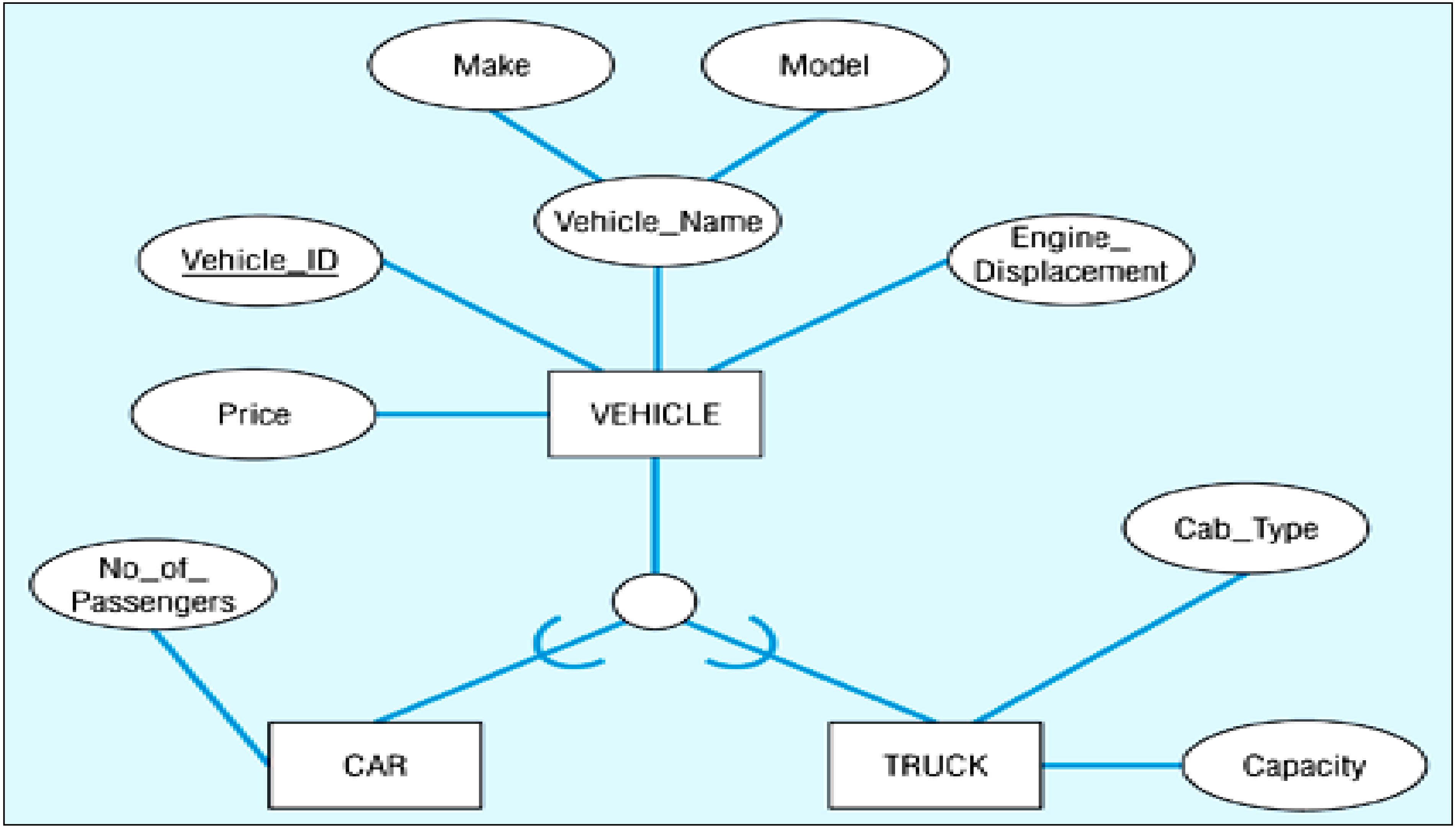
- The reverse of the specialization process.
- Several classes with common features are generalized into a superclass; original classes become its subclasses.
- Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE.
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK.

Example of generalization

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



(b) Generalization to VEHICLE supertype

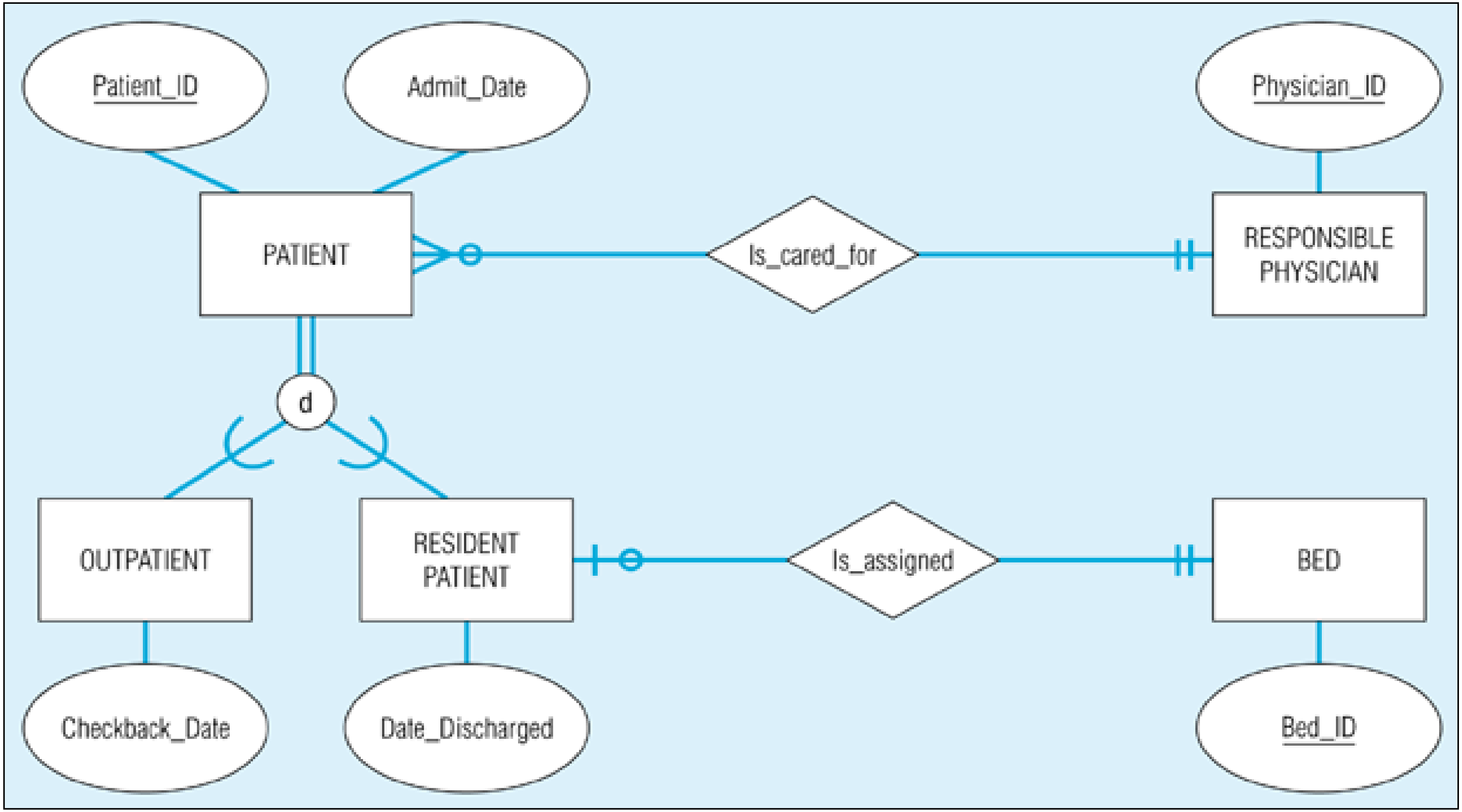


Constraints in Supertype/ Subtype Relationships

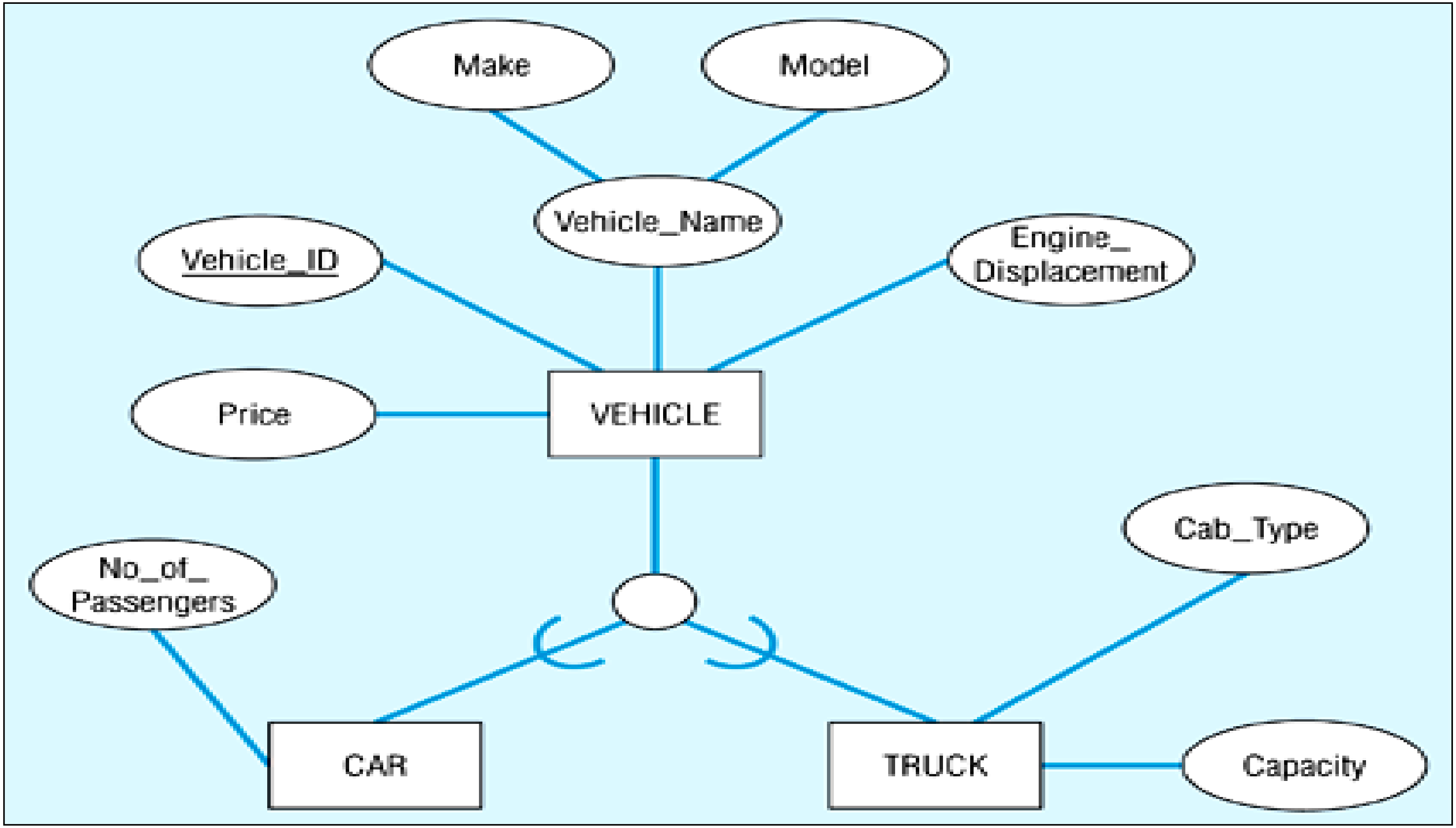
- **Completeness Constraint:**
  - **Total** specifies that every entity in the superclass must be a member of some subclass in the specialization/ generalization.
  - Shown in EER diagrams by a double line.
  - **Partial** allows an entity not to belong to any of the subclasses.
  - Shown in EER diagrams by a single line.

Examples of completeness constraints

(a) Total specialization rule



(b) Partial specialization rule

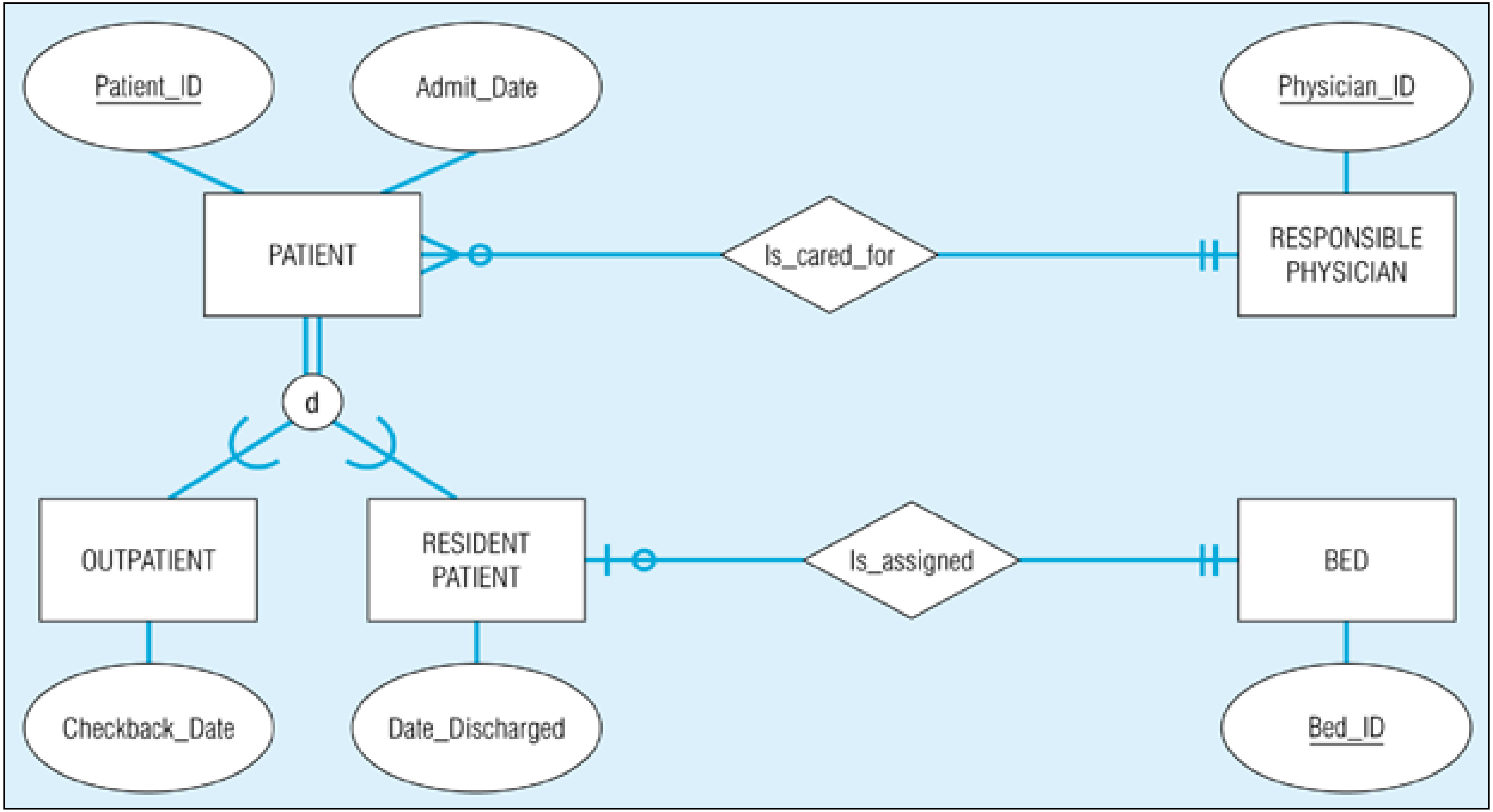


Constraints in Supertype/ Subtype Relationships

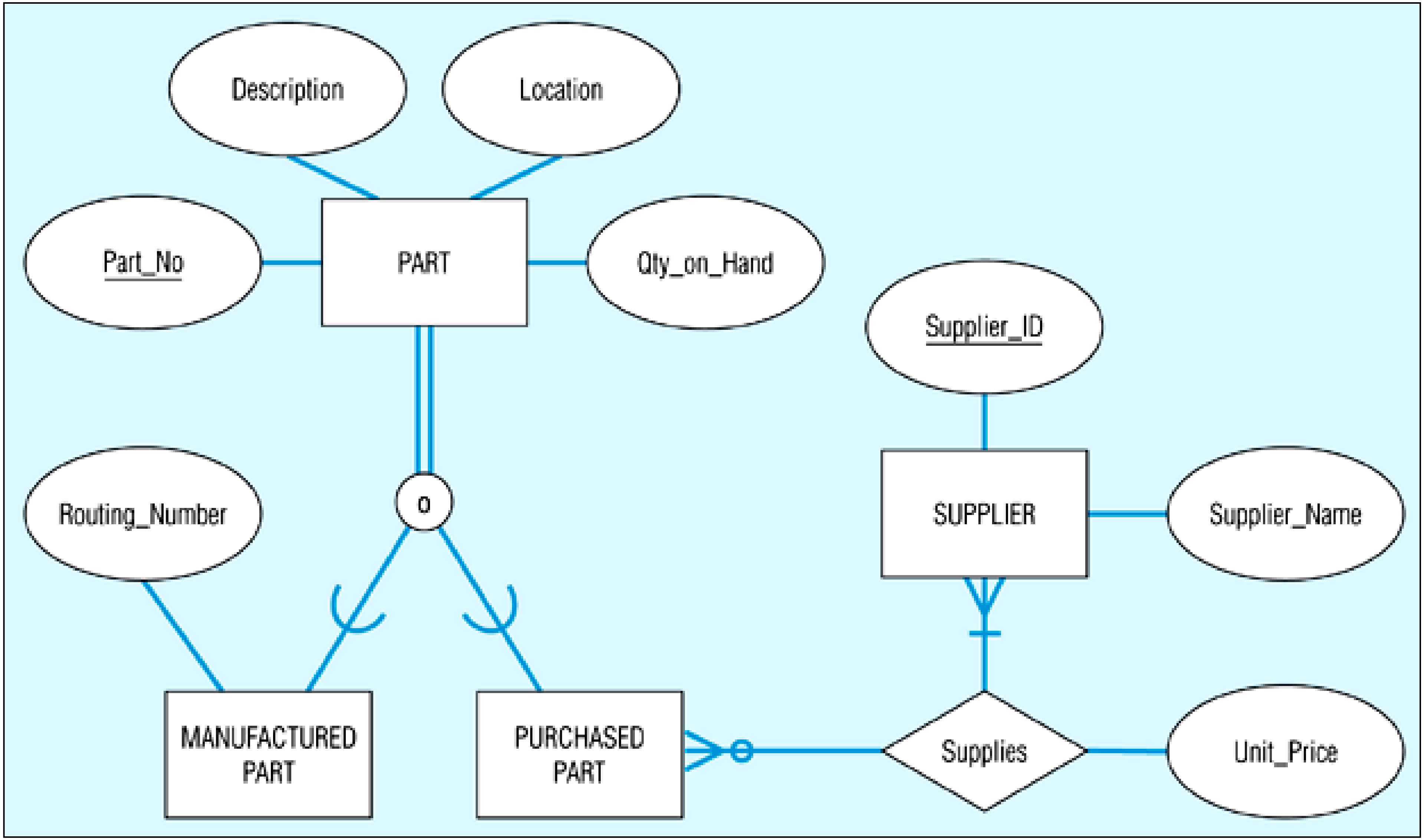
- **Disjointness Constraint:**
  - Specifies that the subclasses of the specialization must be disjointed (an entity can be a member of at most one of the subclasses of the specialization).
  - Specified by **d** in EER diagram.
  - If not disjointed, **overlap**; that is the same entity may be a member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

Examples of disjointness constraints

(a) Disjoint rule



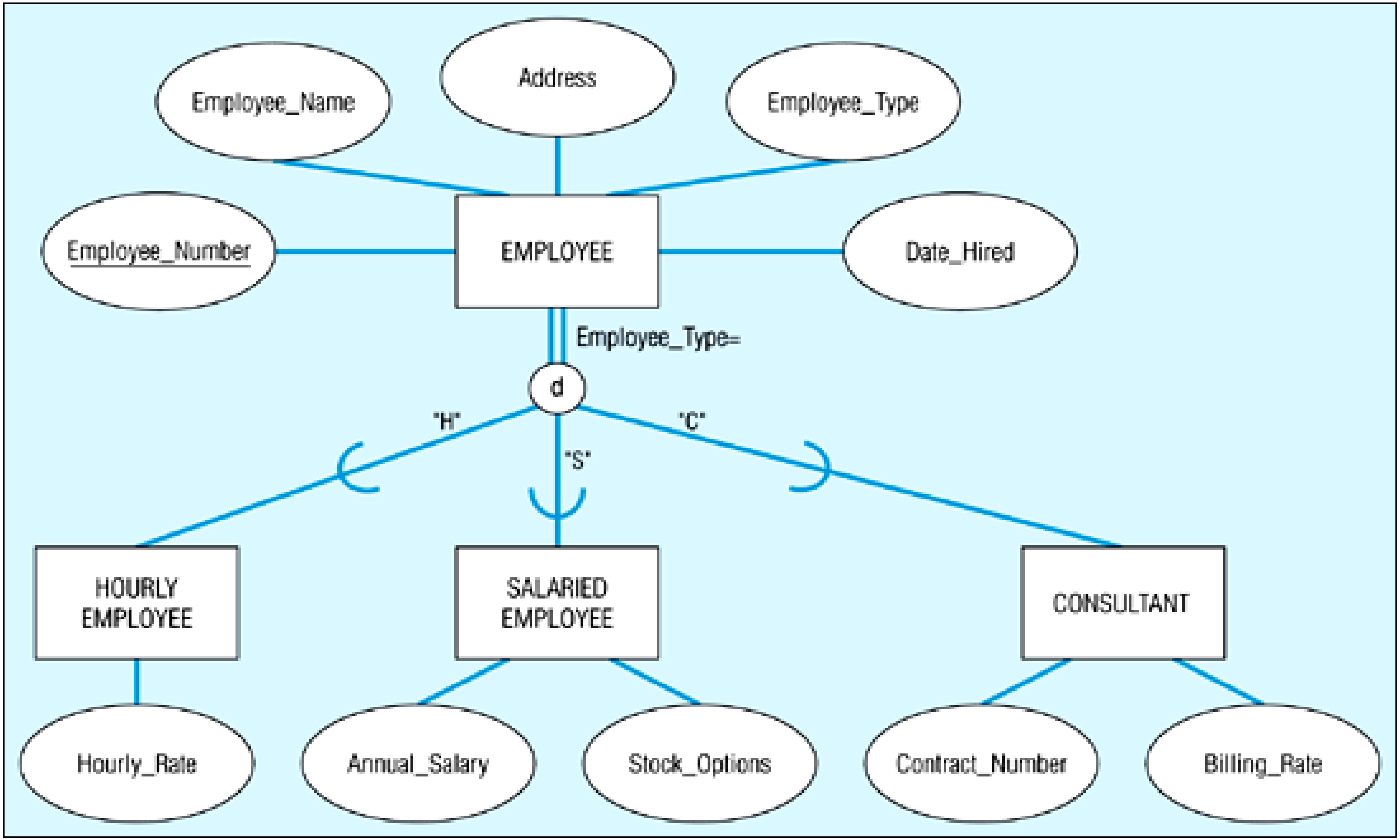
(b) Overlap rule



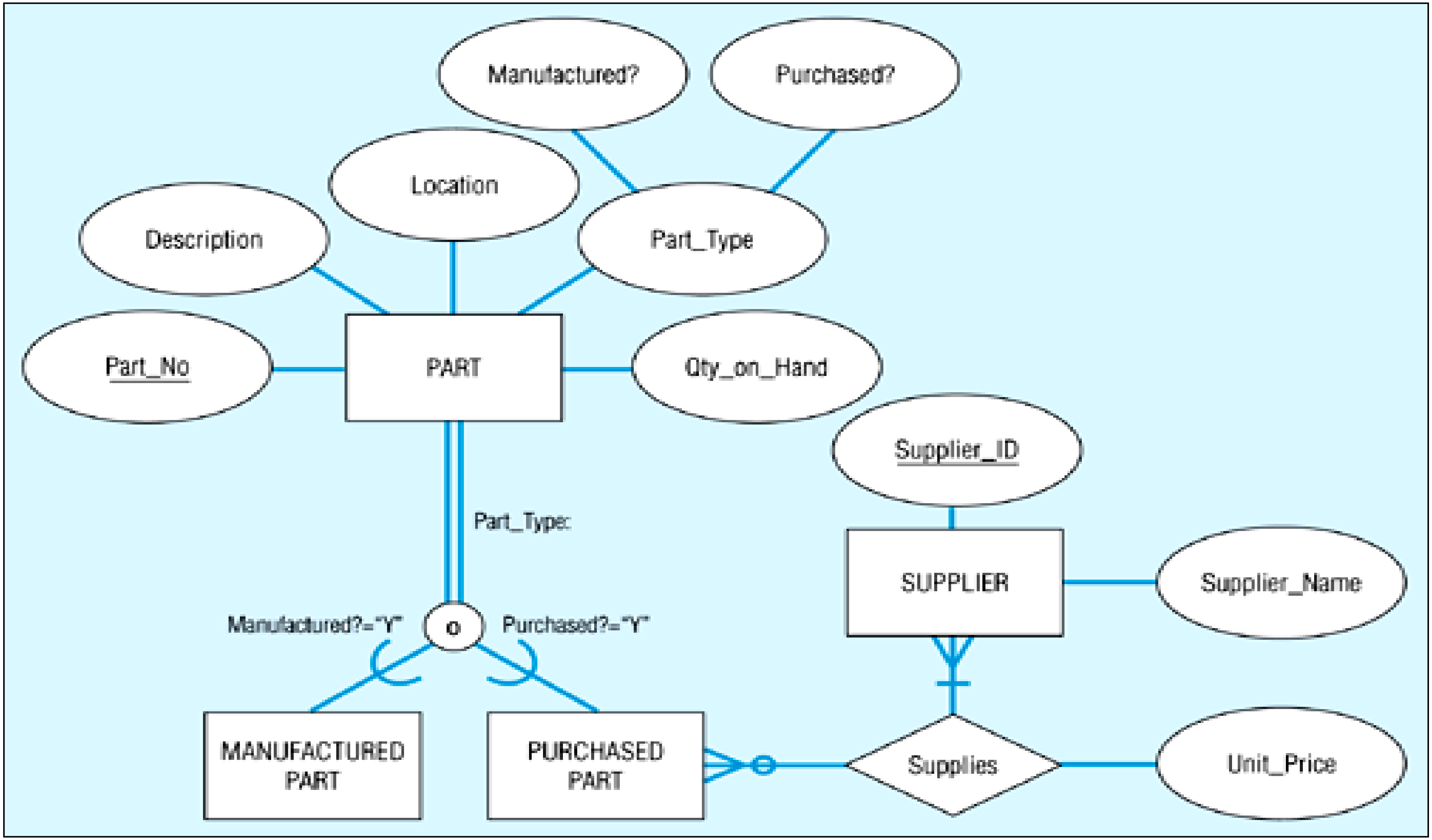
Constraints in Supertype/ Subtype Relationships

- Subtype Discriminators: An attribute of the supertype whose values determine the target subtype(s).
  - Disjoint
  - overlap

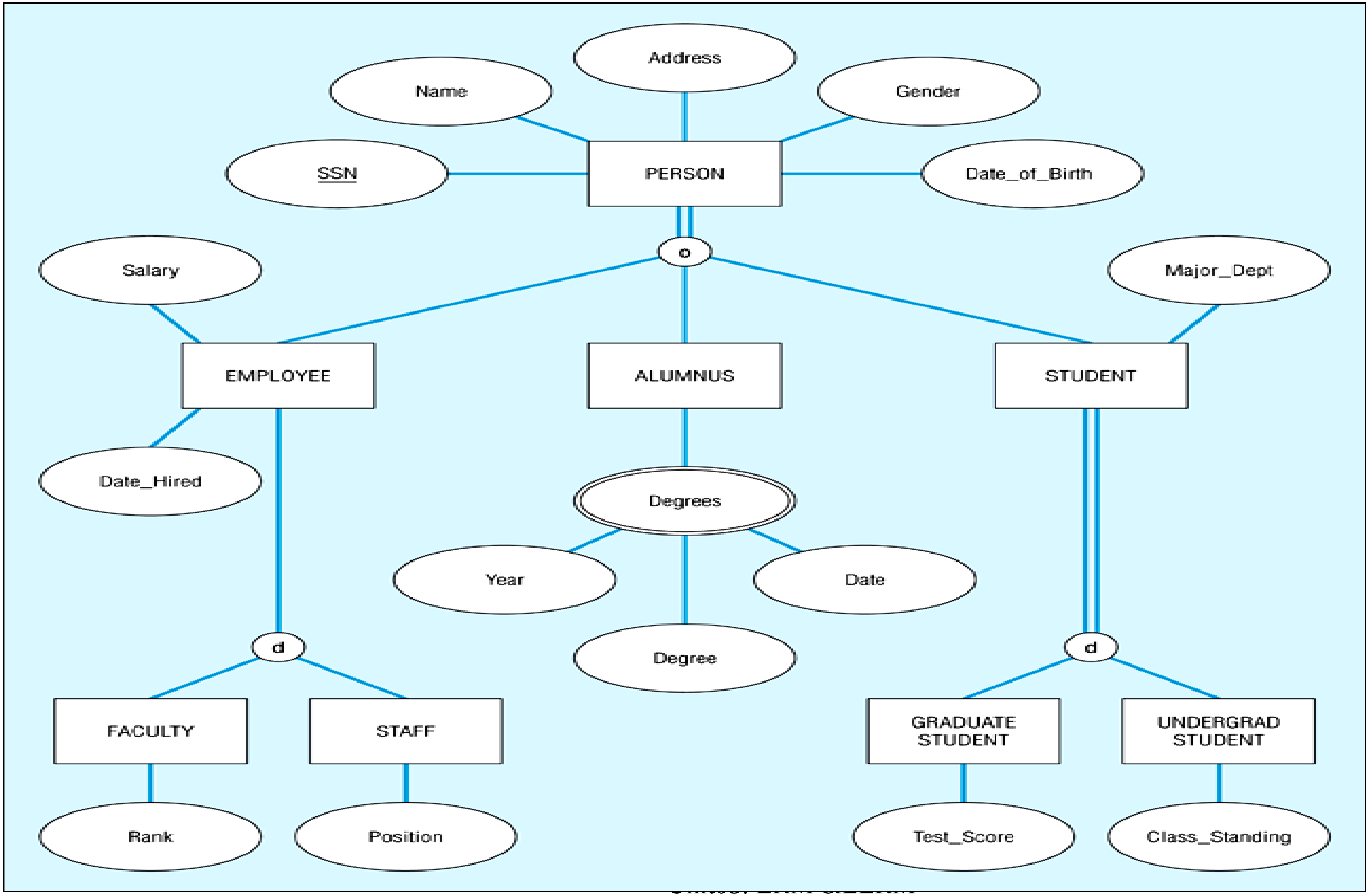
Introducing a subtype discriminator (disjoint rule)



Subtype discriminator (overlap rule)



Example of supertype/subtype hierarchy



CONVERTING EERM INTO RELATIONAL MODEL

ER- to- Relational Mapping

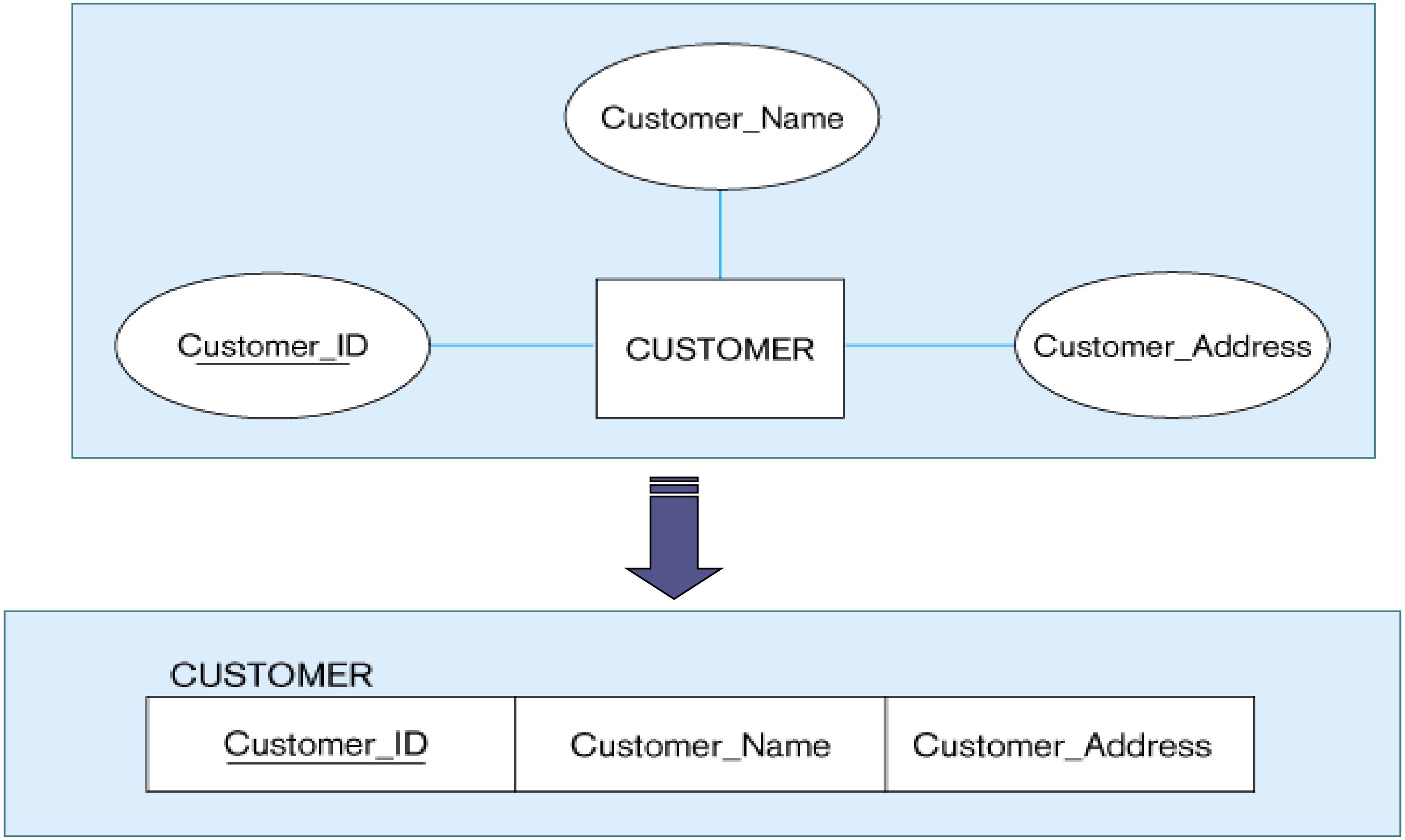
- Step 1: Mapping of regular entity types
- Step 2: Mapping of weak entity types
- Step 3: Mapping of binary 1:1 relationship types
- Step 4: Mapping of binary 1:N relationship types
- Step 5: Mapping of binary M:N relationship types
- Step 8: options for mapping specialization or generalization
- Step 9: Mapping of union types

EER Diagrams into Relations

Mapping Regular Entities to Relations

1. Simple attributes: E-R attributes map directly into the relation
2. Composite attributes: Use only their simple, component attributes
3. Multi-valued Attribute - Becomes a separate relation with a foreign key taken from the superior entity

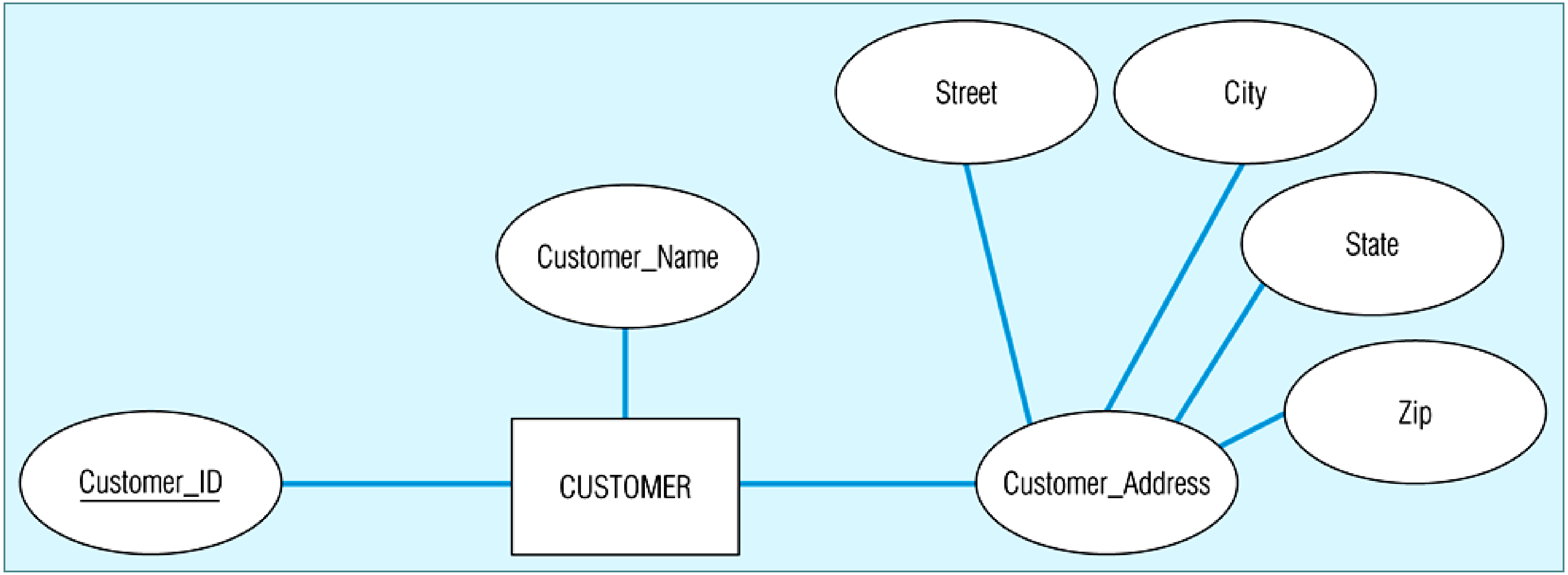
Mapping a regular entity

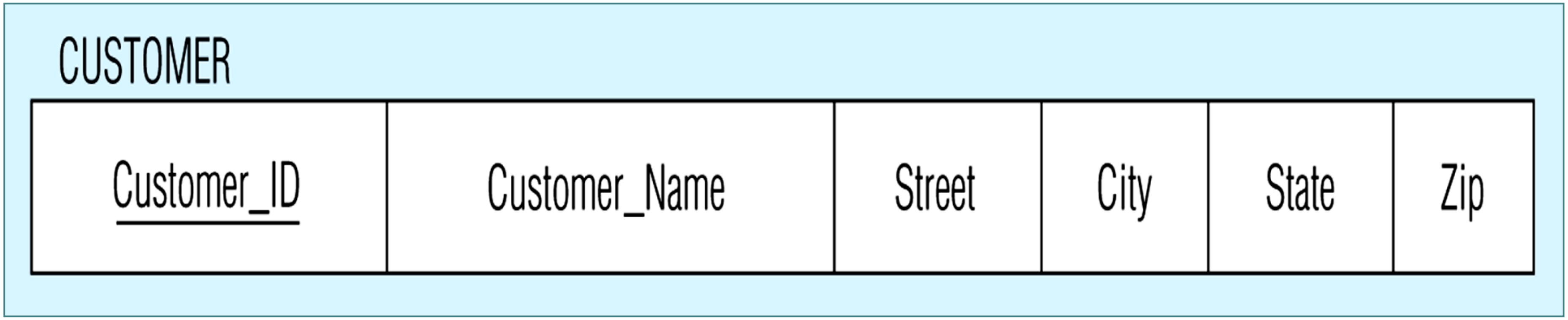


CUSTOMER

Customer_ID	Customer_Name	Customer_Address

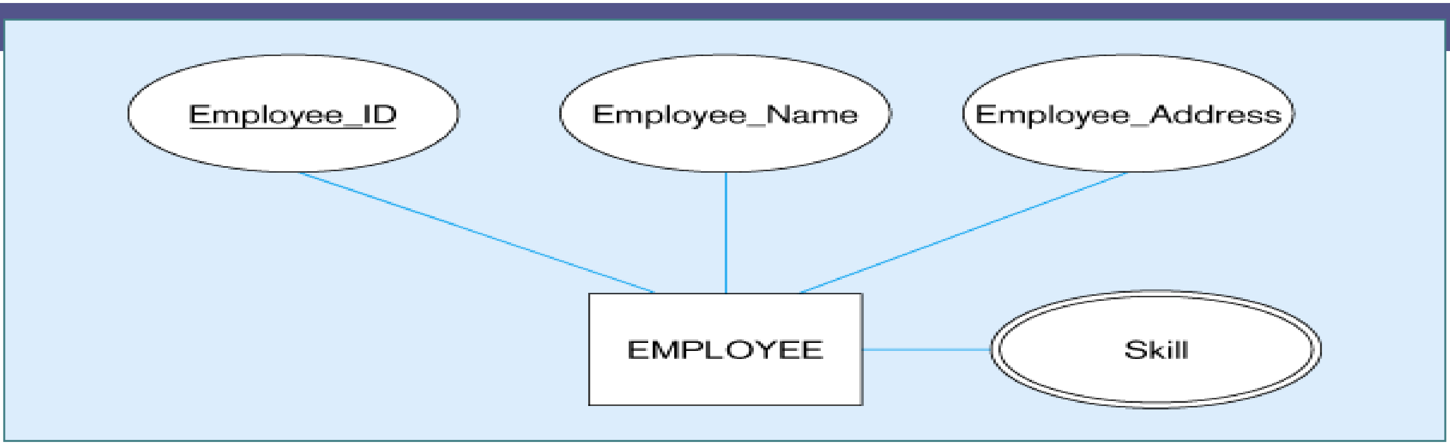
Mapping a composite attribute




CUSTOMER

Customer_ID	Customer_Name	Street	City	State	Zip

Mapping a Multivalued attribute



Employee_ID

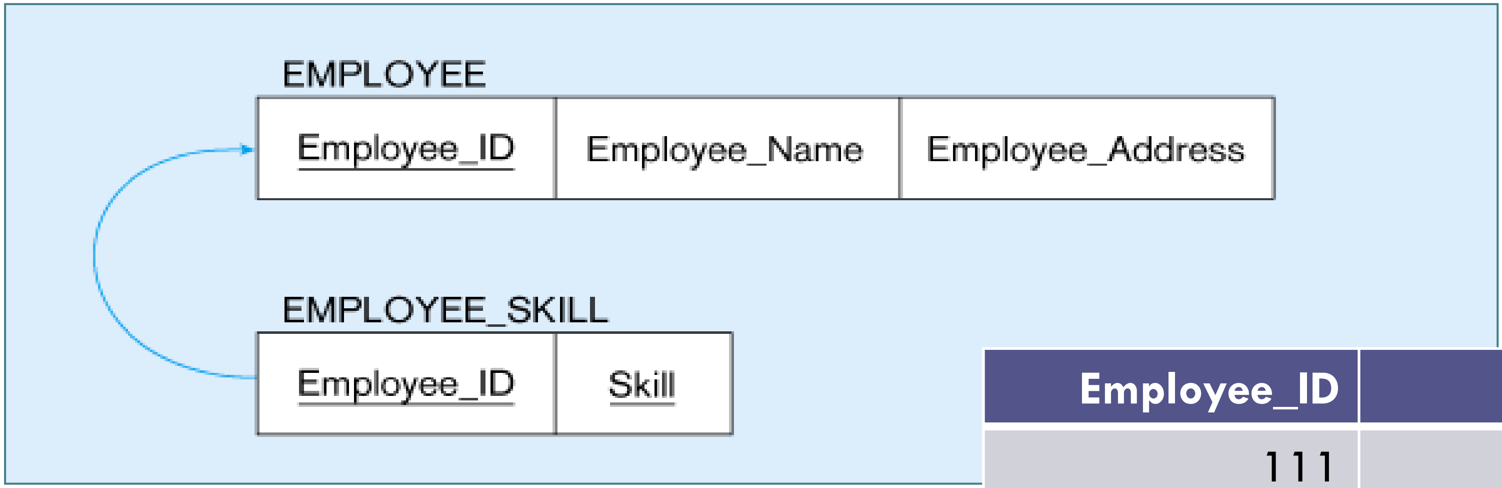
Employee_Name

Employee_Address

EMPLOYEE

Skill

Multivalued attribute becomes a separate relation with foreign key



EMPLOYEE

Employee_ID | Employee_Name | Employee_Address

EMPLOYEE_SKILL

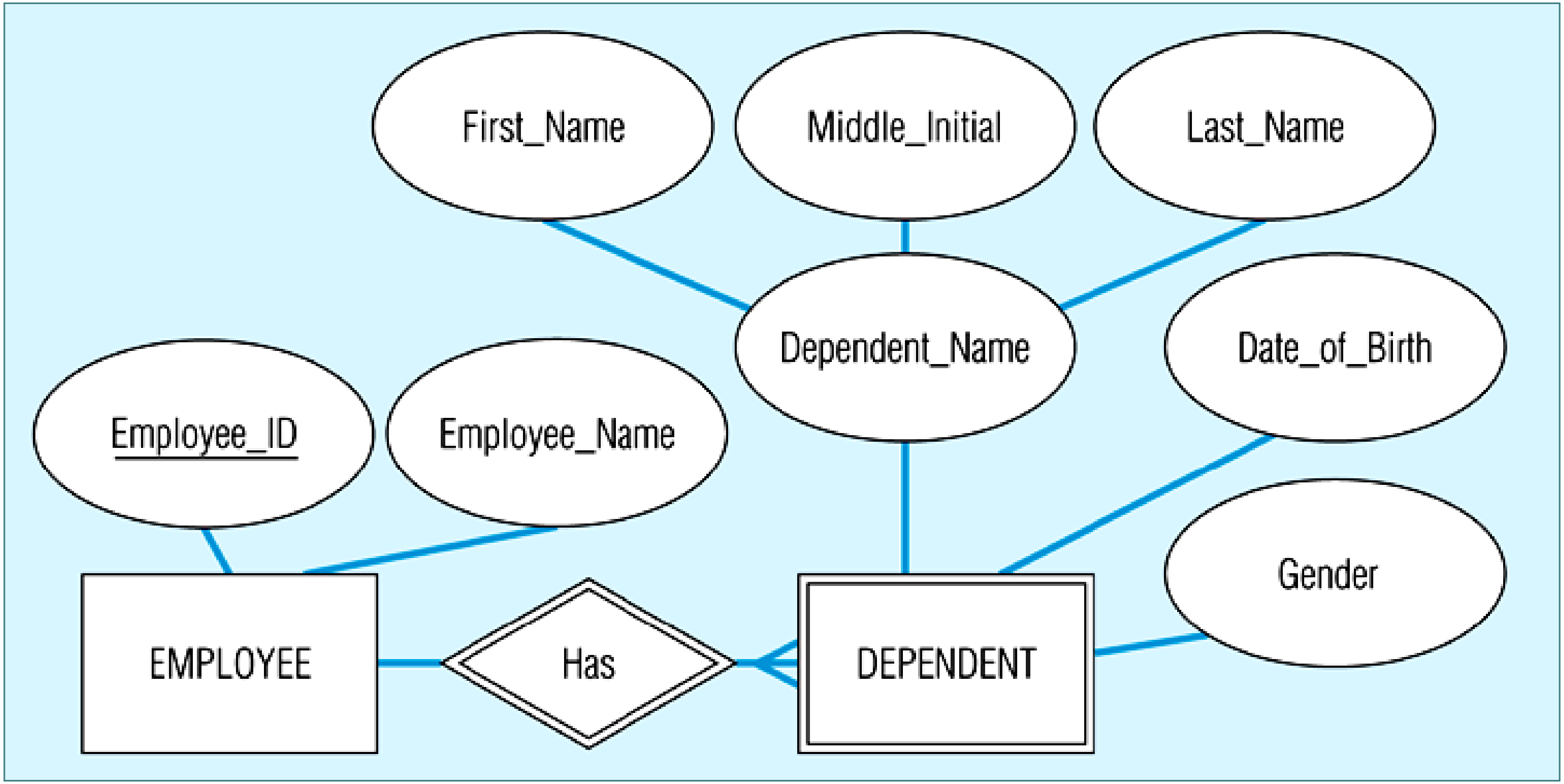
Employee_ID | Skill

Employee_ID	SkILL
111	Leadership
111	Critical thinking
112	Problem solving

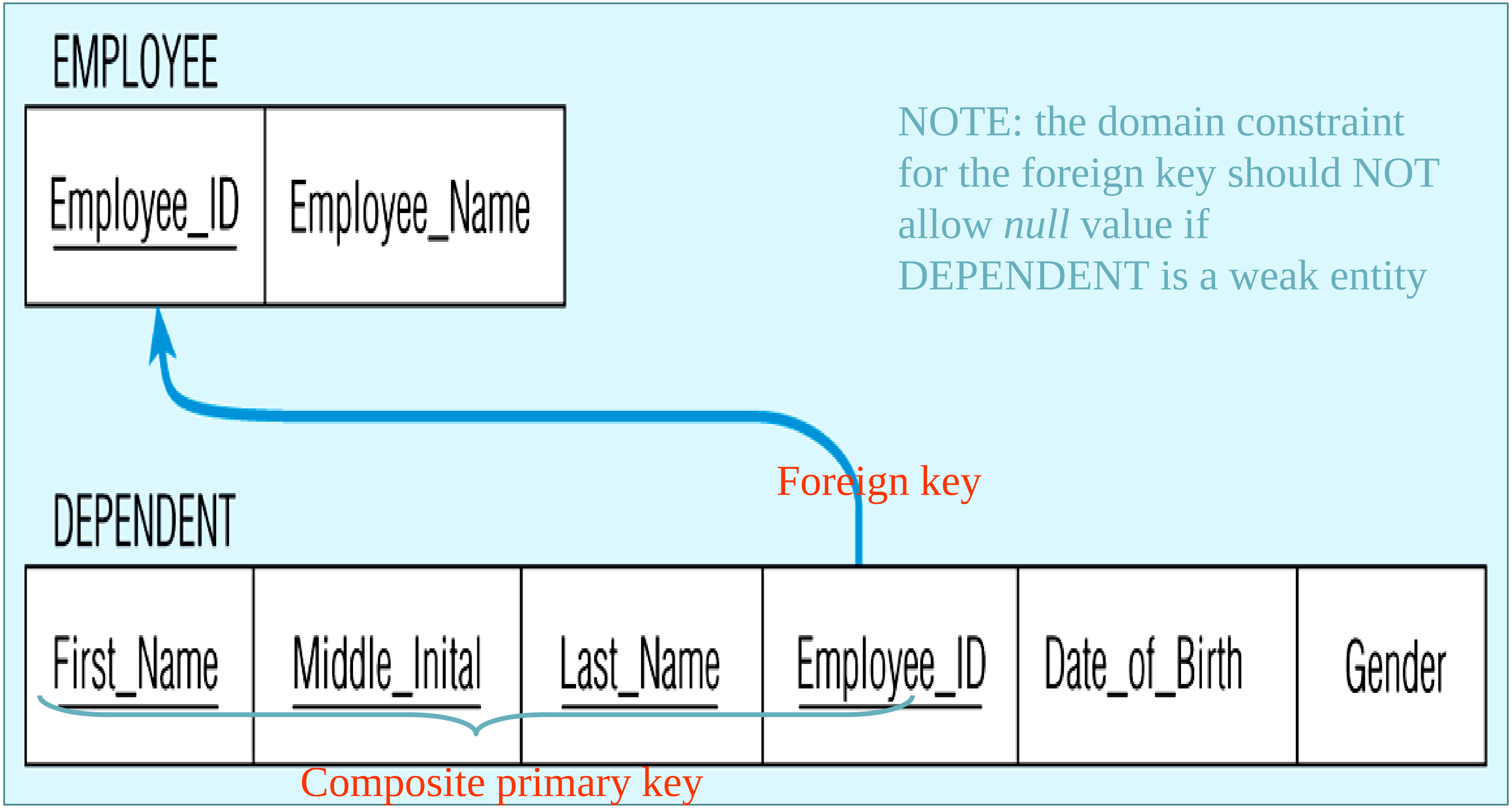
Mapping Weak Entities

- Weak Entity becomes a separate relation with a foreign key taken from the superior entity
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

Mapping a weak entity



Relations resulting from weak entity



EMPLOYEE

Employee_ID	Employee_Name

DEPENDENT

First_Name	Middle_Inital	Last_Name	Employee_ID	Date_of_Birth	Gender

NOTE: the domain constraint for the foreign key should NOT allow *null* value if DEPENDENT is a weak entity

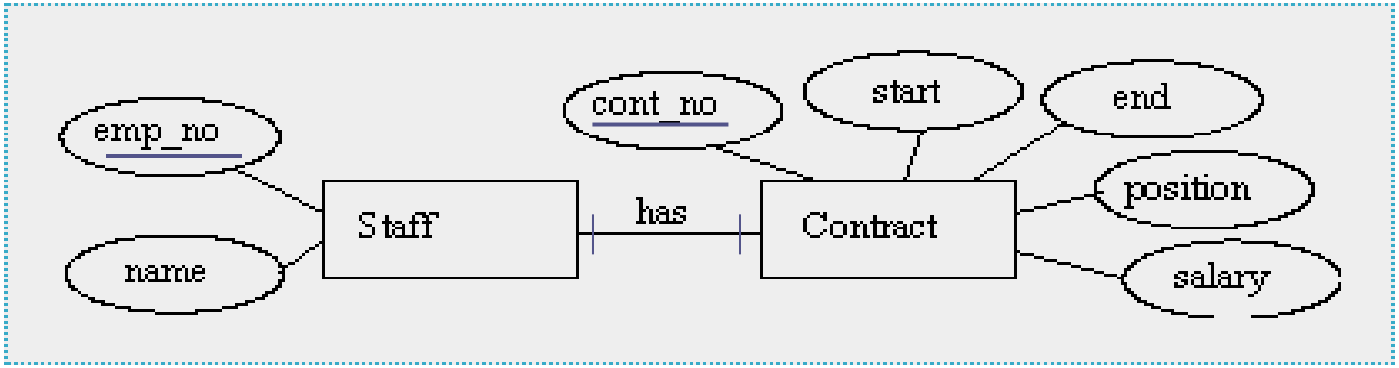
Mapping Relations

Mapping Binary Relationships

□ One-to-One Relationship

•The primary key of one entity type comes the foreign key in the other.

•It does not matter which way around it is done but you should not have a foreign key in each entity.



Staff(emp_no, name, contract_no)

Contract(cont_no, start, end, position, salary)

or

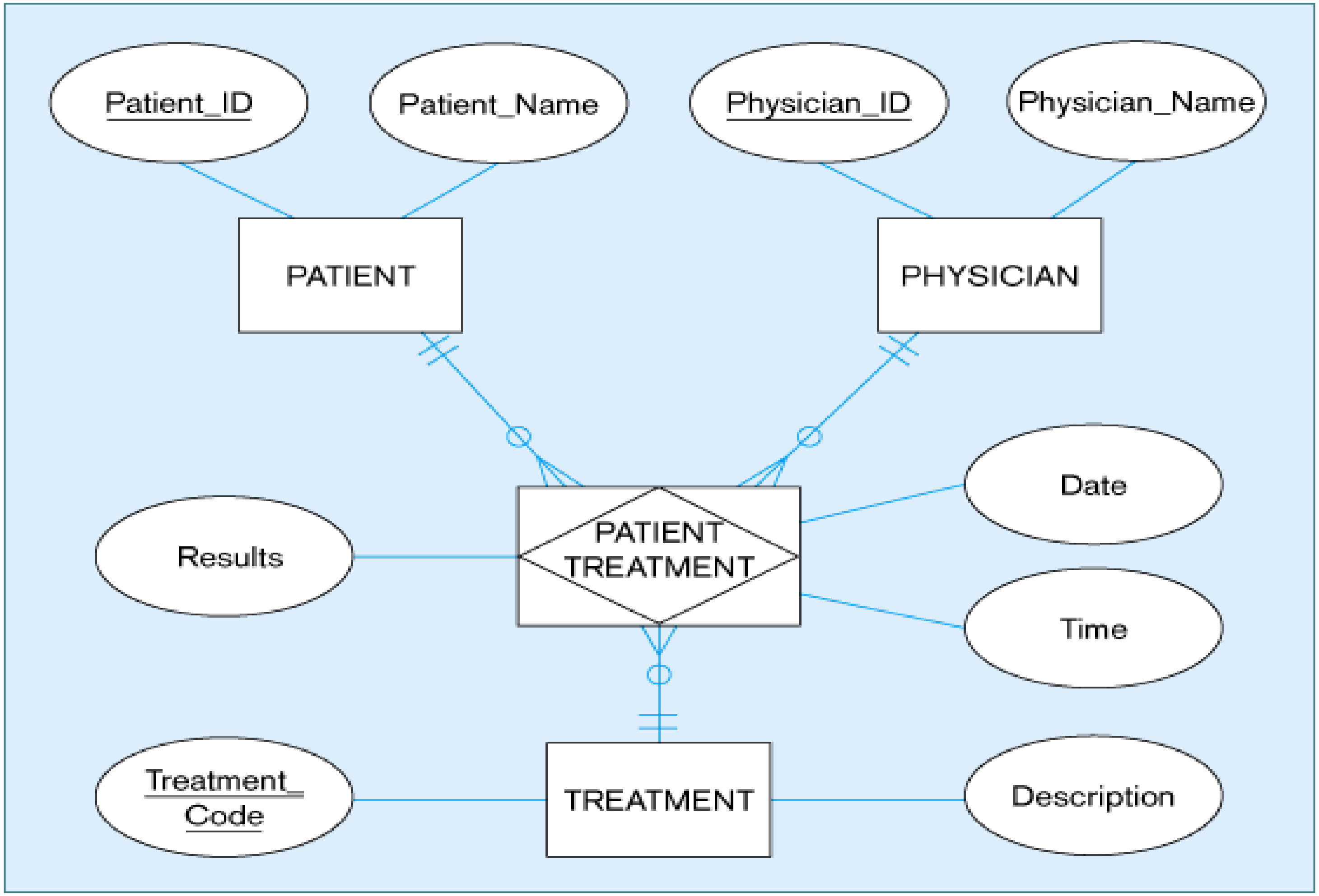
Staff(emp_no, name)

Contract(cont_no, start, end, position, salary, emp_no)

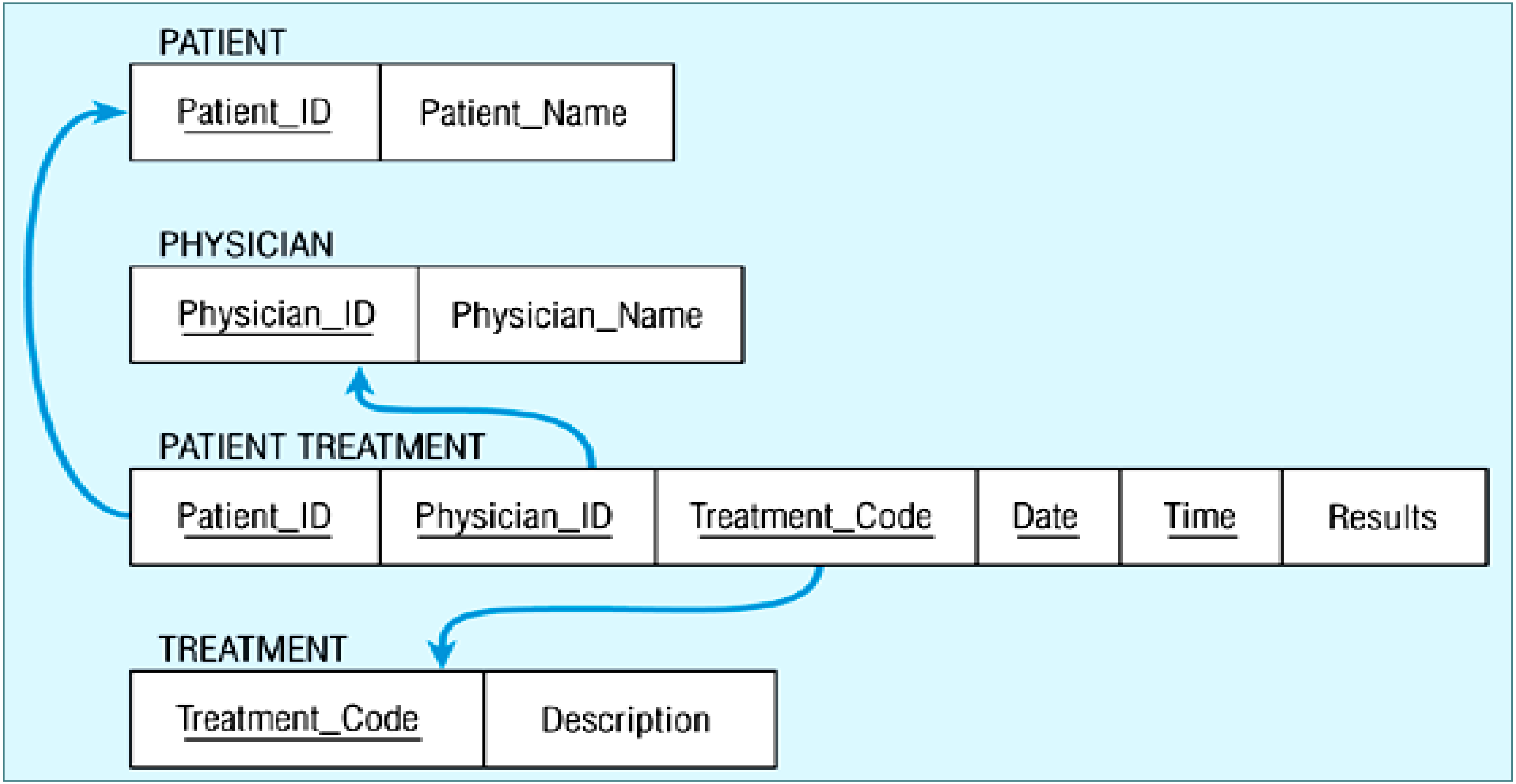
Mapping n-ary Relationships

- One relation for each entity and one for the associative entity
- Associative entity has foreign keys to each entity in the relationship

Mapping a ternary relationship



Mapping a ternary relationship



PATIENT

Patient_ID	Patient_Name

PHYSICIAN

Physician_ID	Physician_Name

PATIENT TREATMENT

Patient_ID	Physician_ID	Treatment_Code	Date	Time	Results

TREATMENT

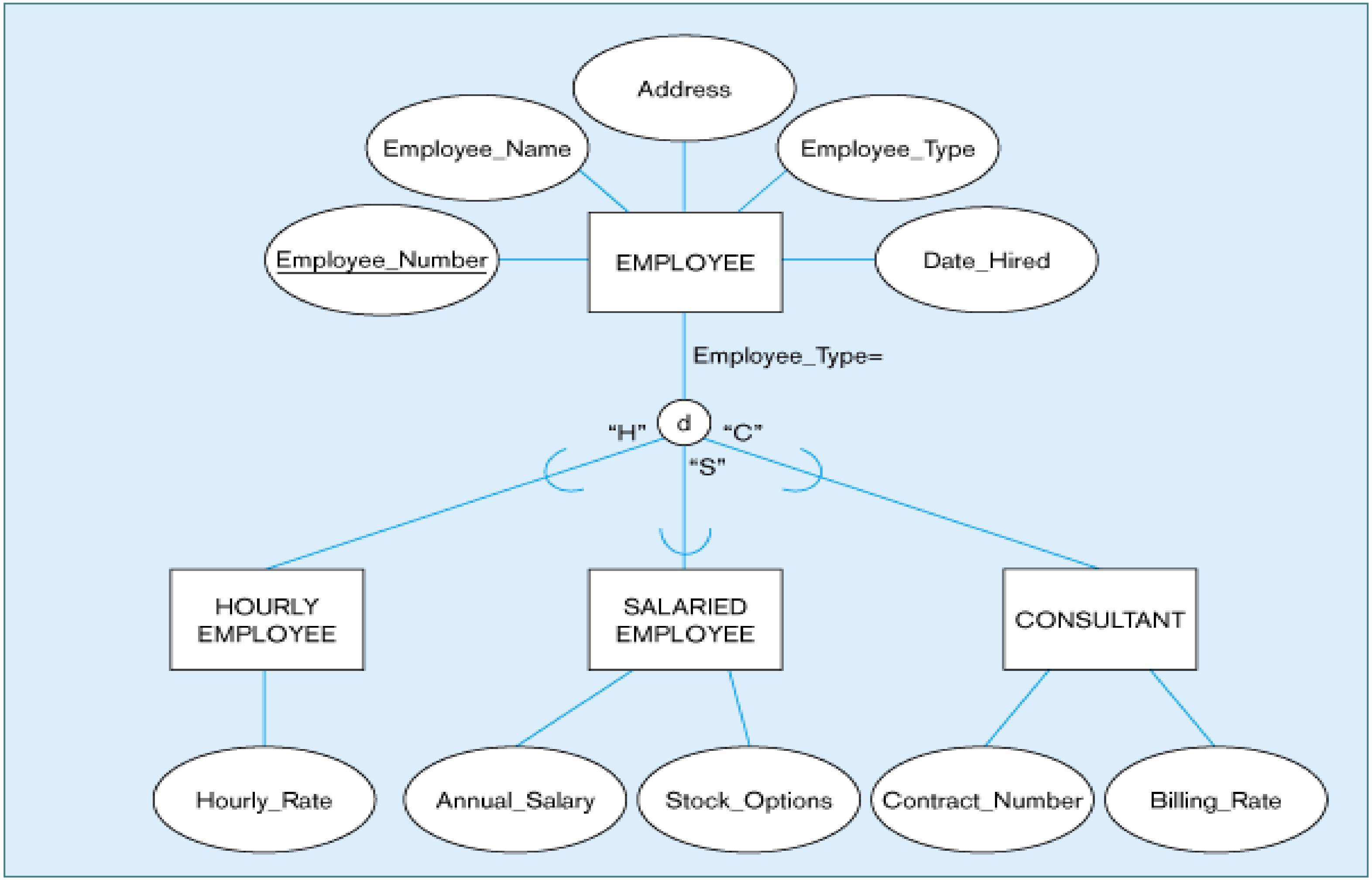
Treatment_Code	Description

Mapping Supertype/Subtype Relationships

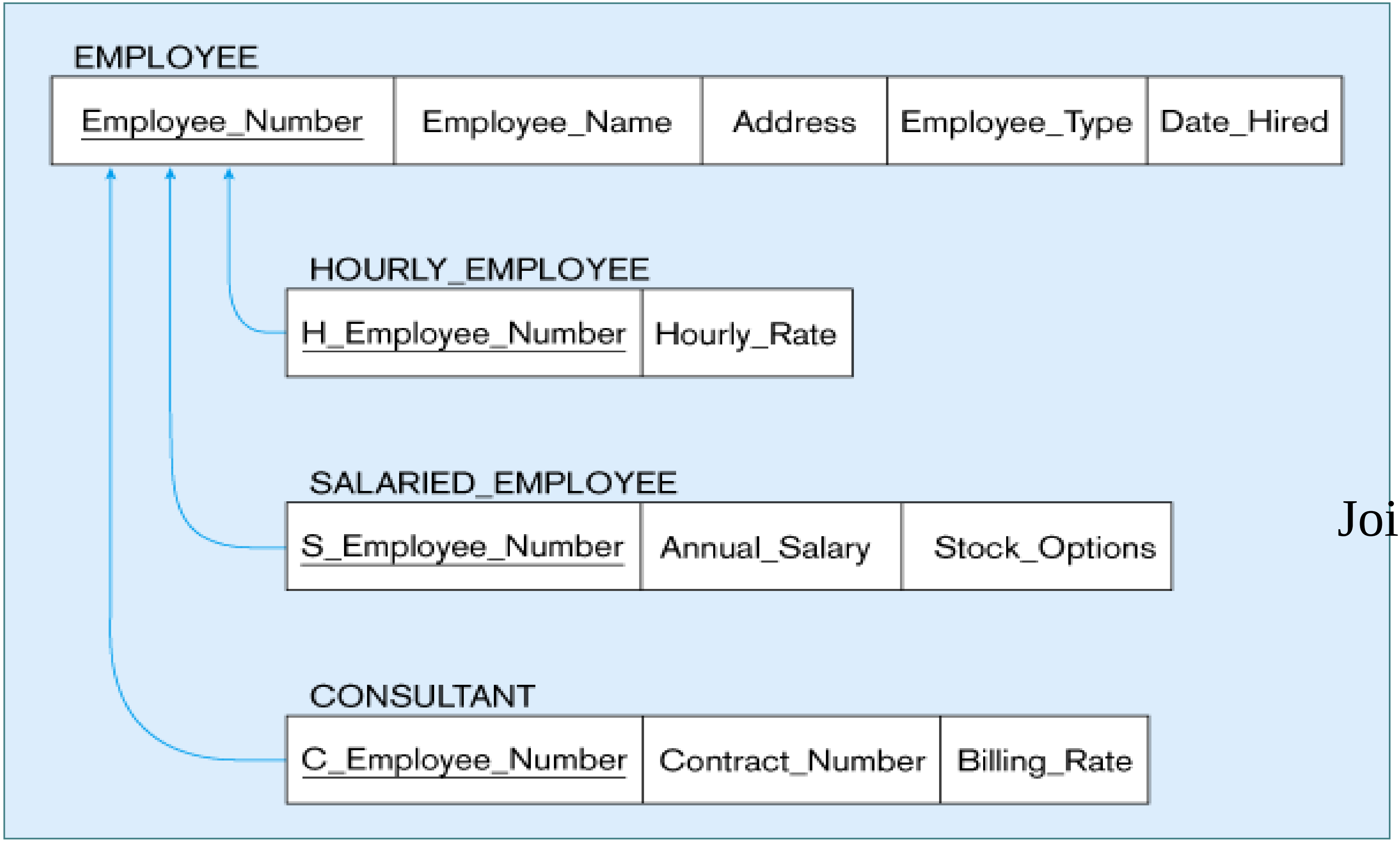
One relation for supertype and for each subtype

- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table

Supertype/subtype relationships



Option 1: Mapping Supertype/subtype relationships to relations



EMPLOYEE

Employee_Number	Employee_Name	Address	Employee_Type	Date_Hired

HOURLY_EMPLOYEE

H_Employee_Number	Hourly_Rate

SALARIED_EMPLOYEE

S_Employee_Number	Annual_Salary	Stock_Options

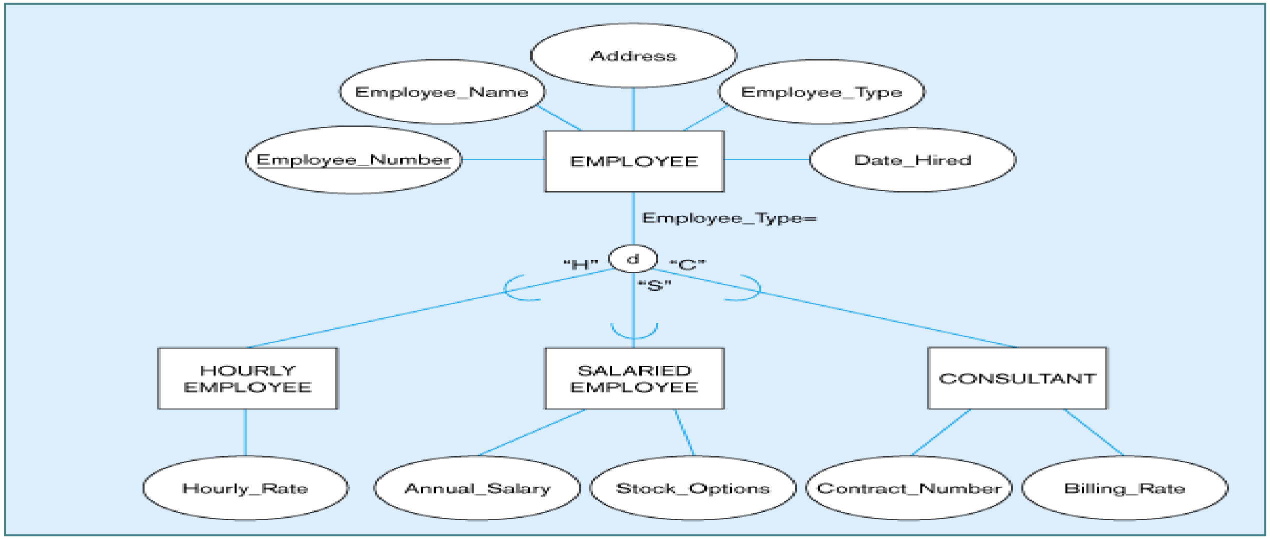
CONSULTANT

C_Employee_Number	Contract_Number	Billing_Rate

Join

Option 2: One relation for each subclass

All attributes are mapped into each subclass.



Hourly_Employee(<u>Employee_number</u>,Employee_Name,Address, Date_Hired,Hourly_rate)

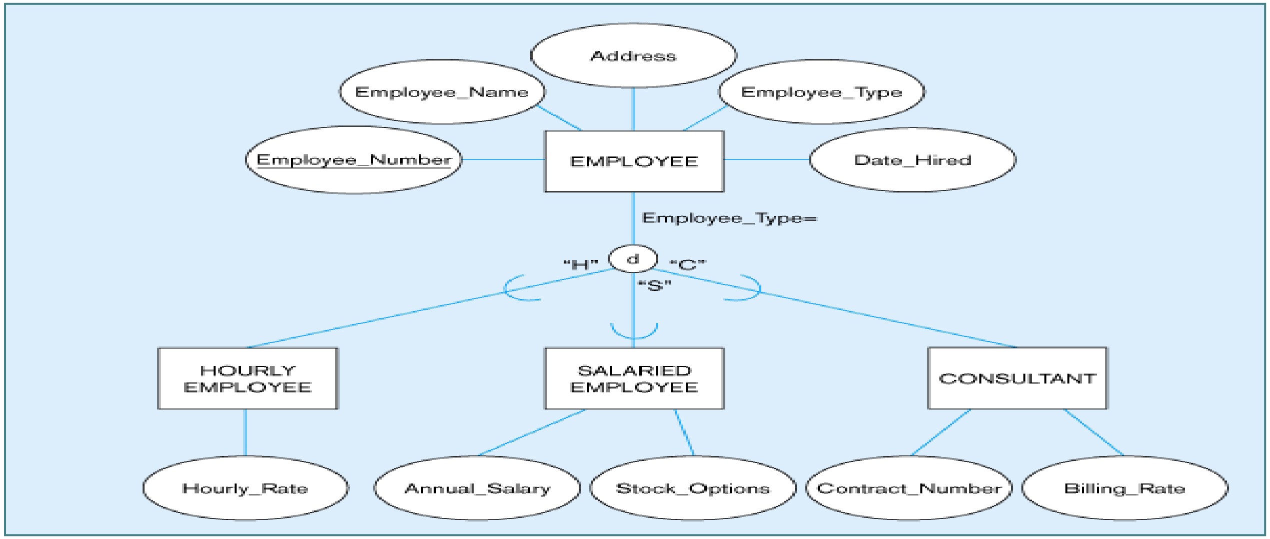
Salried_Employee(<u>Employee_number</u>,Employee_Name,Address, Date_Hired,Annual_Salary,Stock_option)

Consultant(<u>Employee_number</u>,Employee_Name,Address, Date_Hired,Contact_Number,Billing_Rate)

- It works only if the coverage is total and disjoint
- Entities that are not in the subclass are lost
- Overlapping classes cause redundancy

Option 3: One relation for each superclass

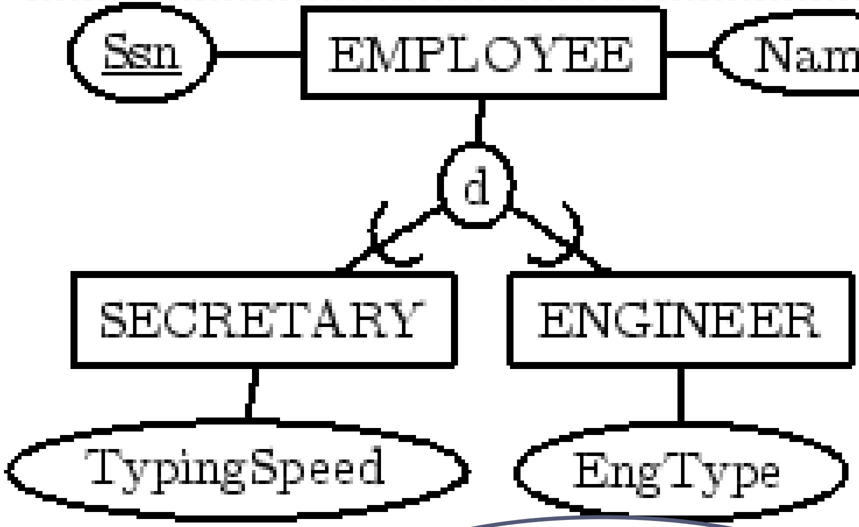
All attributes are mapped into each superclass.



Employee(<u>Employee_number</u>,Employee_Name,Address, Employee_Type,Date_Hired,Hourly_rate,Annual_Salary,Stock_option,Contact_Number,Billing_Rate)

- Many Nulls
- Benefits: No need for joins

Convert the following into Relational Model



Violates the uniqueness

SSN	Name	TypingSpeec	EngType	Secretary	Engineer
1	Ah.	Null	Civil	**No**	Yes
1	Ah.	100/S	Null	Yes	**No**

SSN	Name	TypingSpeec	EngType	Profession
1	Ah.	Null	Civil	Engineer
1	Ah.	100/S	Null	Secretary

Option 1: Employee (SSN, Name), Secretary (SSN, TypingSpeed) and Engineer (SSN, EngType)

Option 2: Secretary (SSN, Name, TypingSpeed) and Engineer (SSN, Name, EngType)

Option 3: Employee (SSN, Name, TypingSpeed, EngType, **Profession**

Option 4: Employee (SSN, Name, TypingSpeed, EngType, **Secretary, Engineer**

Are options alid for Overlapping?