Reflection 1

Enhanced Entity Relationship Model

Introduction to Relational Model

- Data is logically structured within relations.
- A Relation is a mathematical concept based on the ideas of sets
- Each relation is a table (file) with named columns (attributes, fields) and rows (records).

Formal Definitions - Schema

- The **Schema** (or description) of a Relation:
  - Denoted by R(A1, A2, ...An)
  - R is the **name** of the relation
  - The **attributes** of the relation are A1, A2, ..., An
- Example:

  CUSTOMER (Cust-id, Cust-name, Address, Phone#)
  - CUSTOMER is the relation name
  - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values.
  - For example, the domain of Cust-id is 6 digit numbers.

Formal Definitions - Tuple

- A **tuple** is an ordered set of values (enclosed in angled brackets ‘< … >’)
- Each value is derived from an appropriate *domain*.
- A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
  - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
  - This is called a 4-tuple as it has 4 values
  - A tuple (row) in the CUSTOMER relation.
- A relation is a **set** of such tuples (rows)

Formal Definitions - Domain

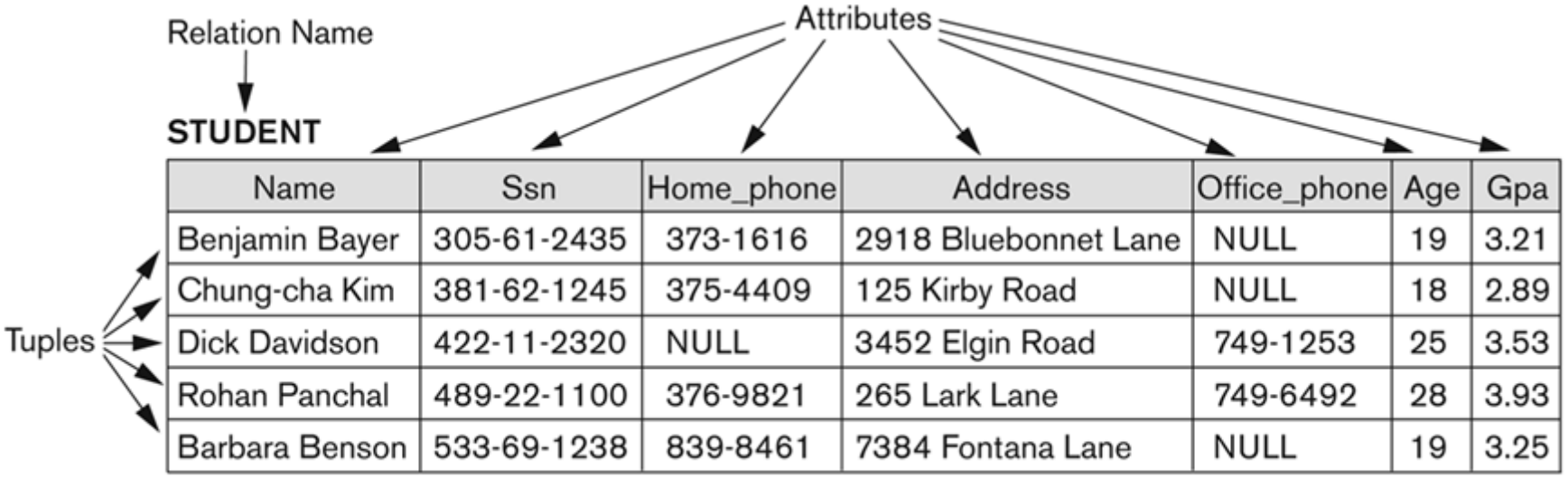
- A domain has a logical definition:
  - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
- A domain also has a data-type or a format defined for it.
  - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
  - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

- The attribute name designates the role played by a domain in a relation:
  - Used to interpret the meaning of the data elements corresponding to that attribute
  - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

Formal Definitions - State

- The relation state is a subset of the Cartesian product of the domains of its attributes
  - each domain contains the set of all possible values the attribute can take.
- Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
  - dom(Cust-name) is varchar(25)
- The role these strings play in the CUSTOMER relation is that of the name of a customer.

Example of a Relation



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Characteristics Of Relations

- Ordering of tuples in a relation R may or may not be needed.
- Ordering of attributes in a relation schema R should be ordered .

Relational Integrity Constraints

- Constraints are conditions that must hold on all valid relation states.
- There are three main types of constraints in the relational model:
  - Key constraints
  - Entity integrity constraints
  - Referential integrity constraints
- Another implicit constraint is the domain constraint
  - Every value in a tuple must be from the domain of its attribute (or it could be null, if allowed for that attribute)

Key Constraints

- Superkey of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] $\neq$ t2[SK]
    - This condition must hold in any valid state r(R)
- Key of R:
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints (continued)

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but not a key.
- In general:
  - Any key is a superkey (but not vice versa)
  - Any set of attributes that includes a key is a superkey
  - A minimal superkey is also a key

Key Constraints (continued)

- If a relation has several candidate keys, one is chosen arbitrarily to be the primary key.
  - The primary key attributes are underlined.
- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - We chose SerialNo as the primary key
- The primary key value is used to uniquely identify each tuple in a relation
  - Provides the tuple identity
- Also used to reference the tuple from another tuple
  - General rule: Choose as primary key the smallest of the candidate keys (in terms of size)
  - Not always applicable – choice is sometimes subjective

Example:

The CAR relation, with two candidate keys: License_number and Engine_serial_number.

CAR

License_number	Engine_serial_number	Make	Model	Year
Texas ABC-739	A69352	Ford	Mustang	02
Florida TVP-347	B43696	Oldsmobile	Cutlass	05
New York MPO-22	X83554	Oldsmobile	Delta	01
California 432-TFY	C43742	Mercedes	190-D	99
California RSK-629	Y82935	Toyota	Camry	04
Texas RSK-629	U028365	Jaguar	XJS	04

Relational Database Schema

- Relational Database Schema:
  - A set S of relation schemas that belong to the same database.
  - S is the name of the whole database schema
  - S = {R1, R2, ..., Rn}
  - R1, R2, …, Rn are the names of the individual relation schemas within the database S
- Following slide shows a COMPANY database schema with 6 relation schemas

COMPANY Database Schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Schema diagram for the COMPANY relational database schema.

Entity Integrity

- Entity Integrity:
  - The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R).
    - This is because primary key values are used to identify the individual tuples.
    - t[PK] $\neq$ null for any tuple t in r(R)
    - If PK has several attributes, null is not allowed in any of these attributes

Referential Integrity

- A constraint involving two relations
  - The previous constraints involve a single relation.
- Used to specify a relationship among tuples in two relations:
  - The referencing relation and the referenced relation.

Referential Integrity

- Tuples in the referencing relation R1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R2.
  - A tuple t1 in R1 is said to reference a tuple t2 in R2 if t1[FK] = t2[PK].
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2.

Referential Integrity (or foreign key) Constraint

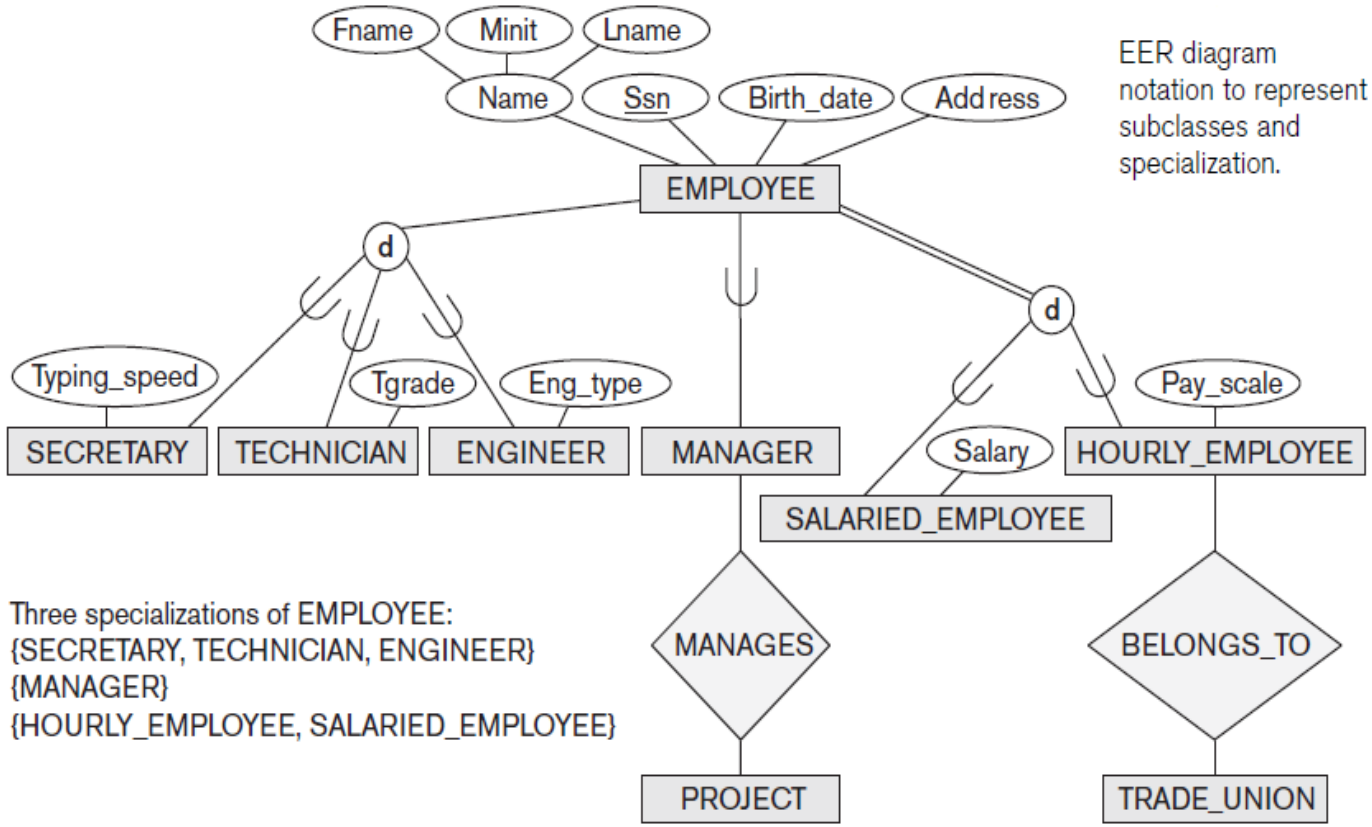
- Statement of the constraint
  - The value in the foreign key column (or columns) FK of the the referencing relation R1 can be either:
    - (1) a value of an existing primary key value of a corresponding primary key PK in the referenced relation R2, or
    - (2) a null.
- In case (2), the FK in R1 should not be a part of its own primary key.

Enhanced Entity-Relationship (EER) Model

- Enhanced Entity-Relationship (EER) modeling is an extension to the ER modeling that incorporates additional constructs
- Central construct
  Superclass/subclass (SC/sc) relationship
- More specifically:
  - Specialization/generalization
  - Categorization
  - Aggregation

Subclasses, Super classes, and Inheritance

- Subtype or subclass of an entity type
  - Subgroupings of entities that are meaningful
  - Represented explicitly because of their significance to the database application
- Terms for relationship between a superclass and any one of its subclasses
  - Superclass/subclass
  - Supertype/subtype
  - Class/subclass relationship
- Type inheritance
  - Subclass entity inherits all attributes and relationships of superclass



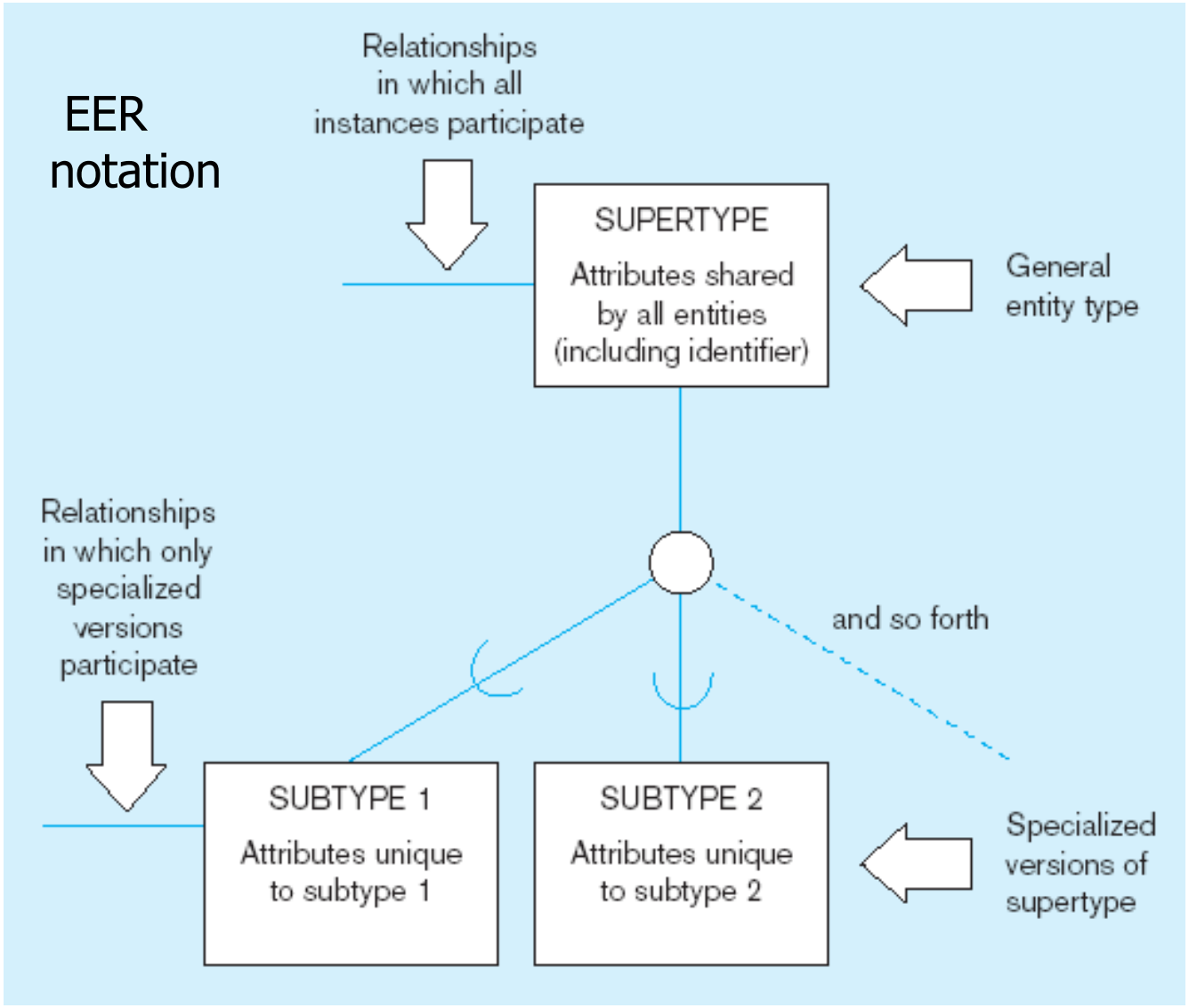
EER diagram notation to represent subclasses and specialization.

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

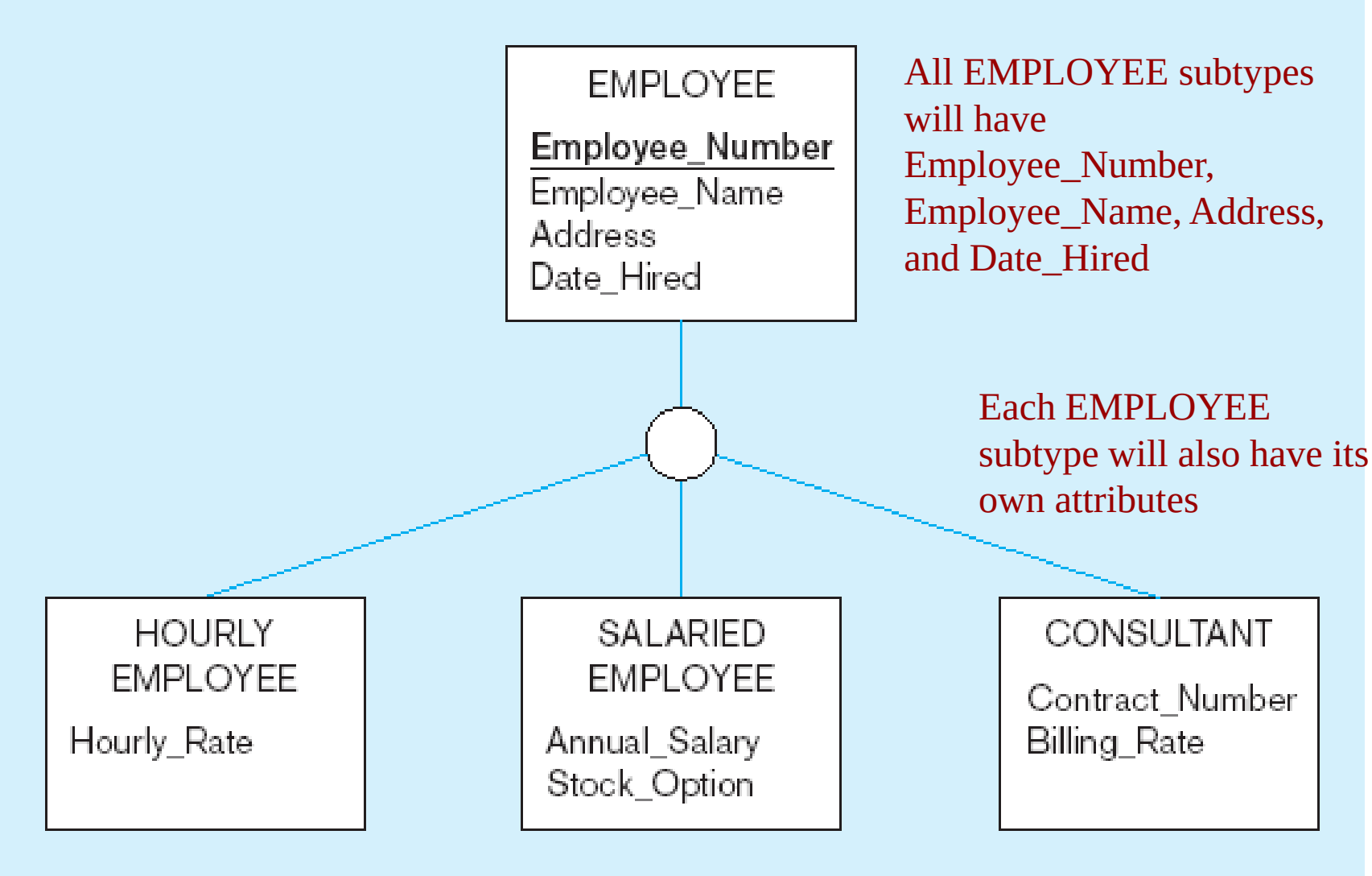
Supertypes and Subtypes

- Supertype: A generic entity type that has a relationship with one or more subtypes
  - An entity type that includes one or more distinct subgroups which require to be represented in a data model.
- Subtype: A subgrouping of the entities in an entity type that has attributes distinct from those in other subgroupings
- Attribute Inheritance:
  - Subtype entities inherit values of all attributes of the supertype
  - An instance of a subtype is also an instance of the supertype

Basic notation for supertype/subtype notation



EMPLOYEE supertype with three subtypes



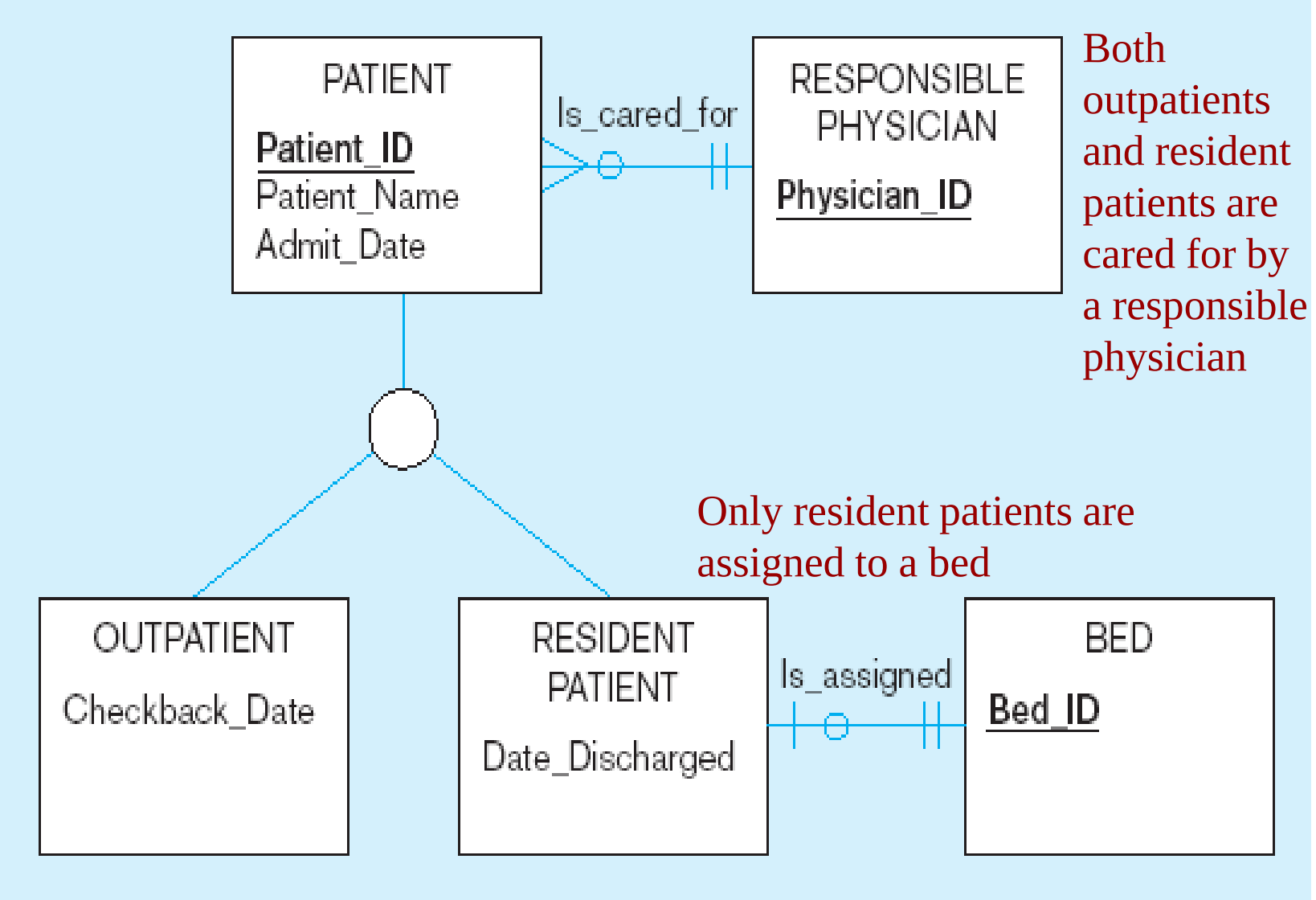
Example

- Employee:
  - EID, Ename, Salary, Position, TypingSpeed, ManagerBonus, SalesArea, CarAllowance
- Employee:EID, Ename, Salary, Position
  - Manager: managerBonus
  - SalesPerson: SalesArea, CarAllowance
  - Secretary: TypingSpeed
- Faculty
  - Professor
  - Lecturer
  - Graduate Assistant

Relationships and Subtypes

- 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

Supertype/subtype relationships in a hospital

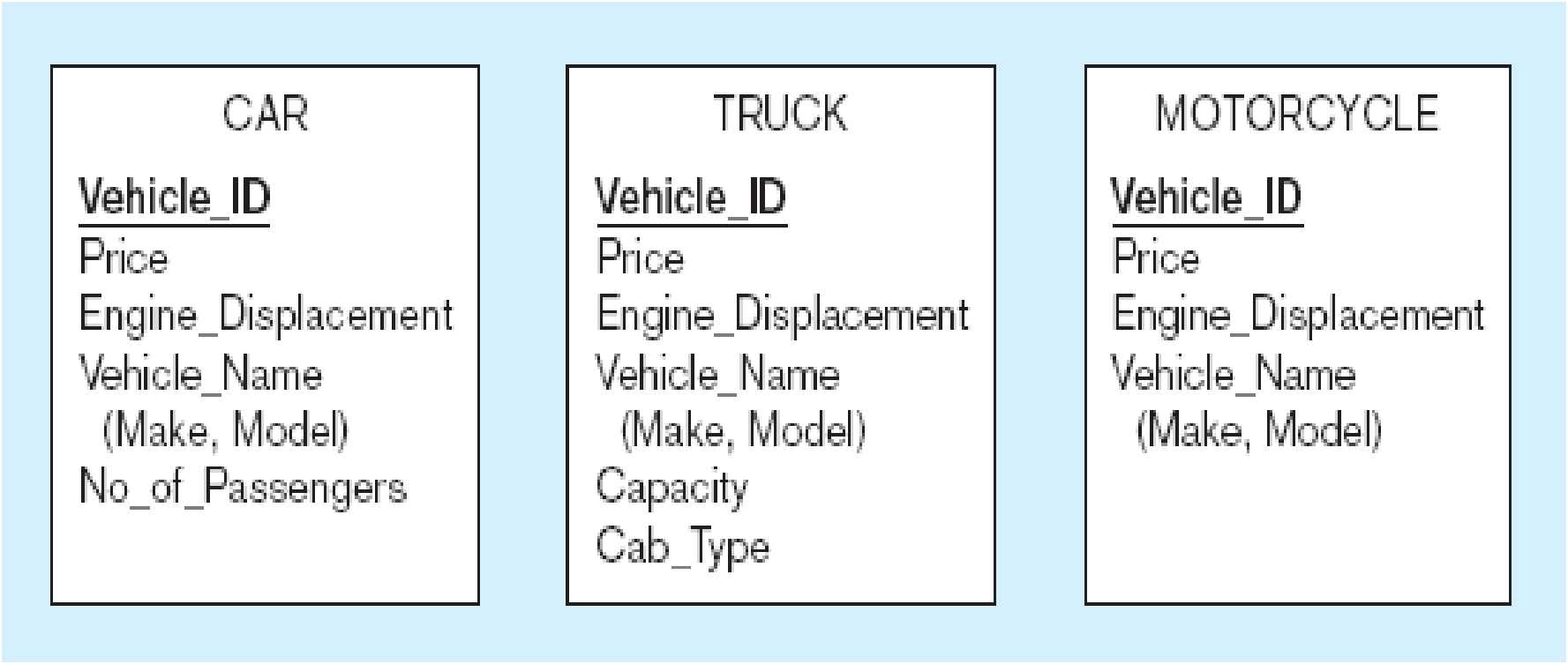


Generalization and Specialization

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

Example of generalization

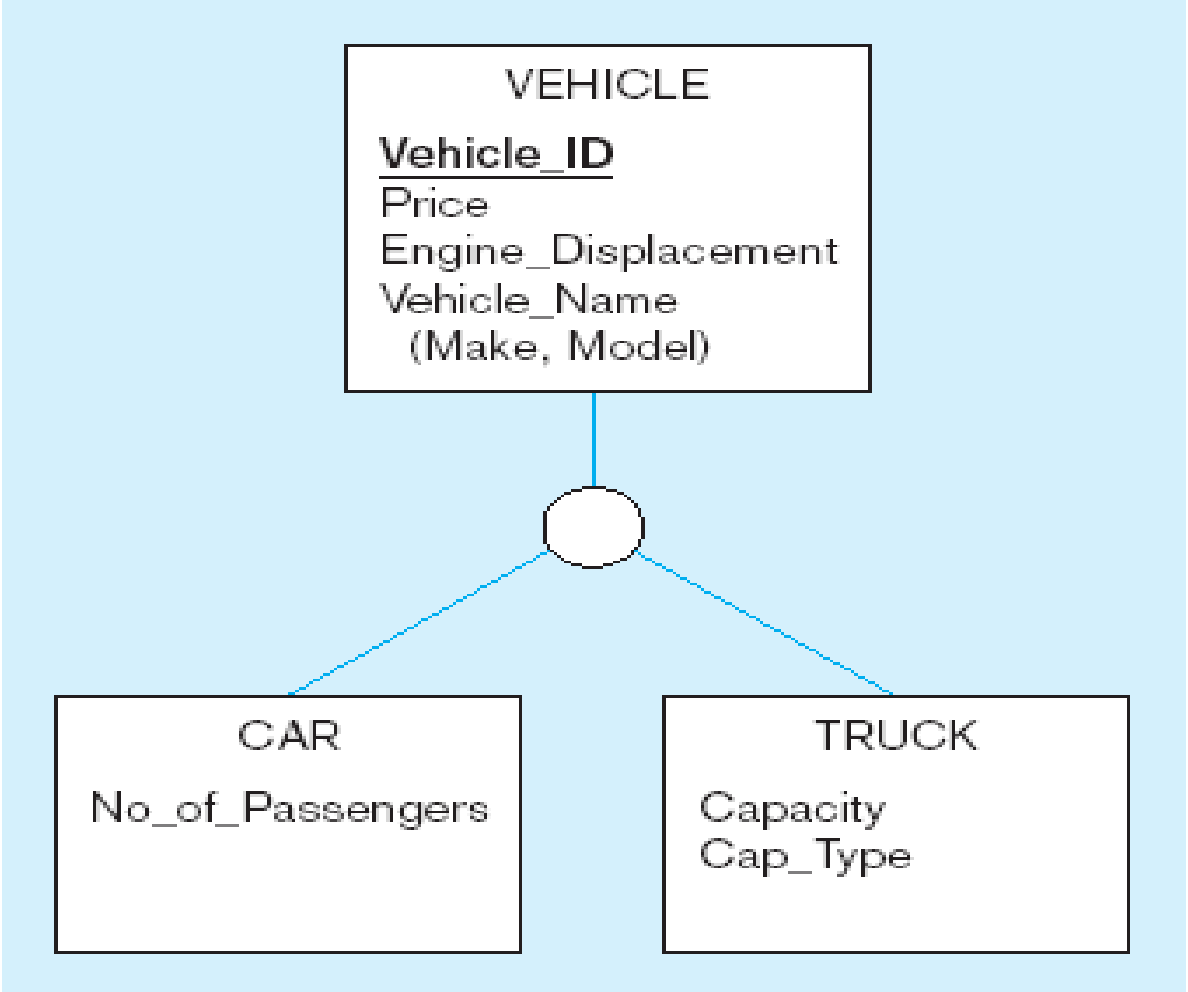
Three entity types: CAR, TRUCK, and MOTORCYCLE



All these types of vehicles have common attributes

Example of generalization (cont.)

Generalization to VEHICLE supertype

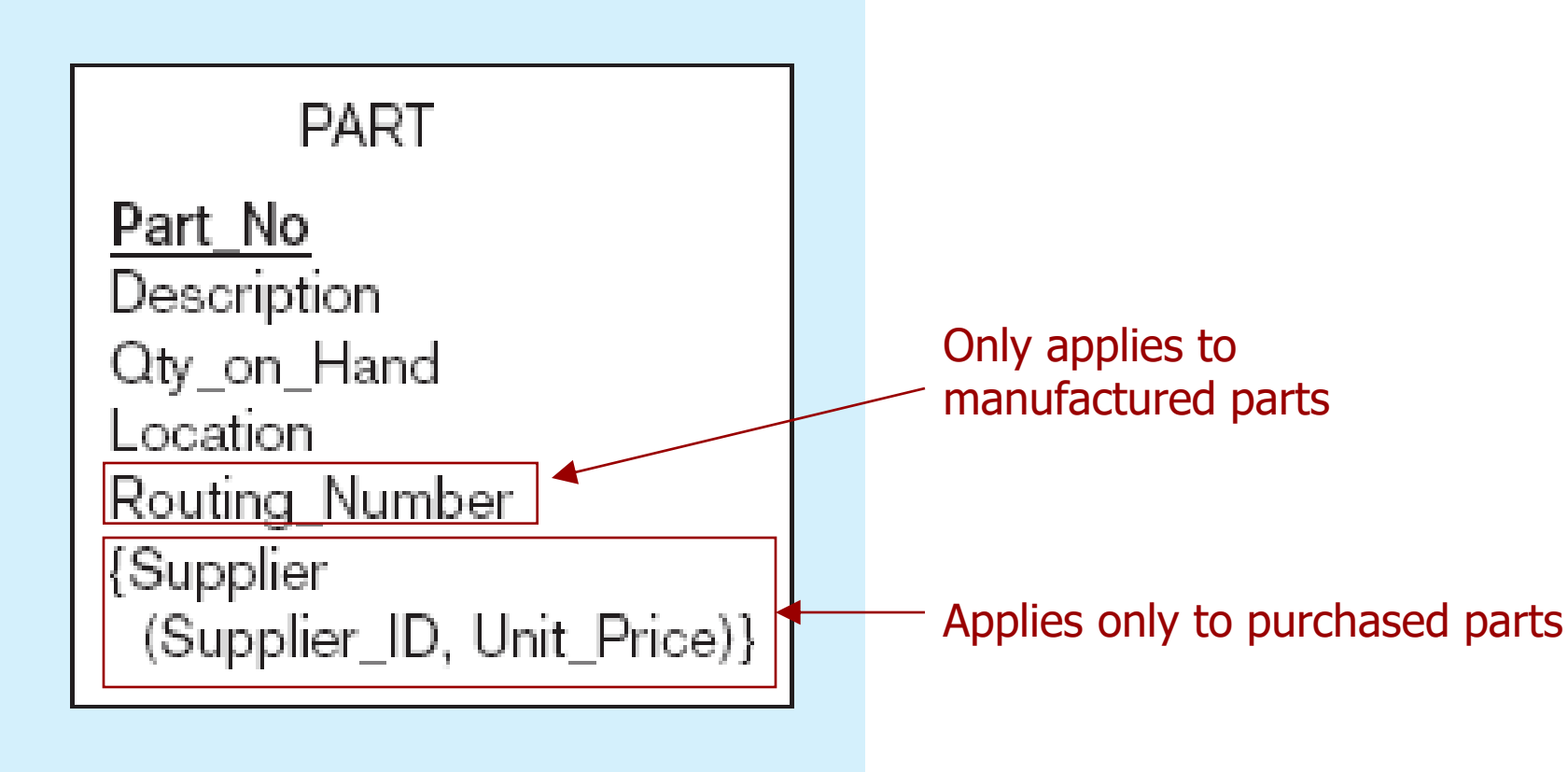


So we put the shared attributes in a supertype

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

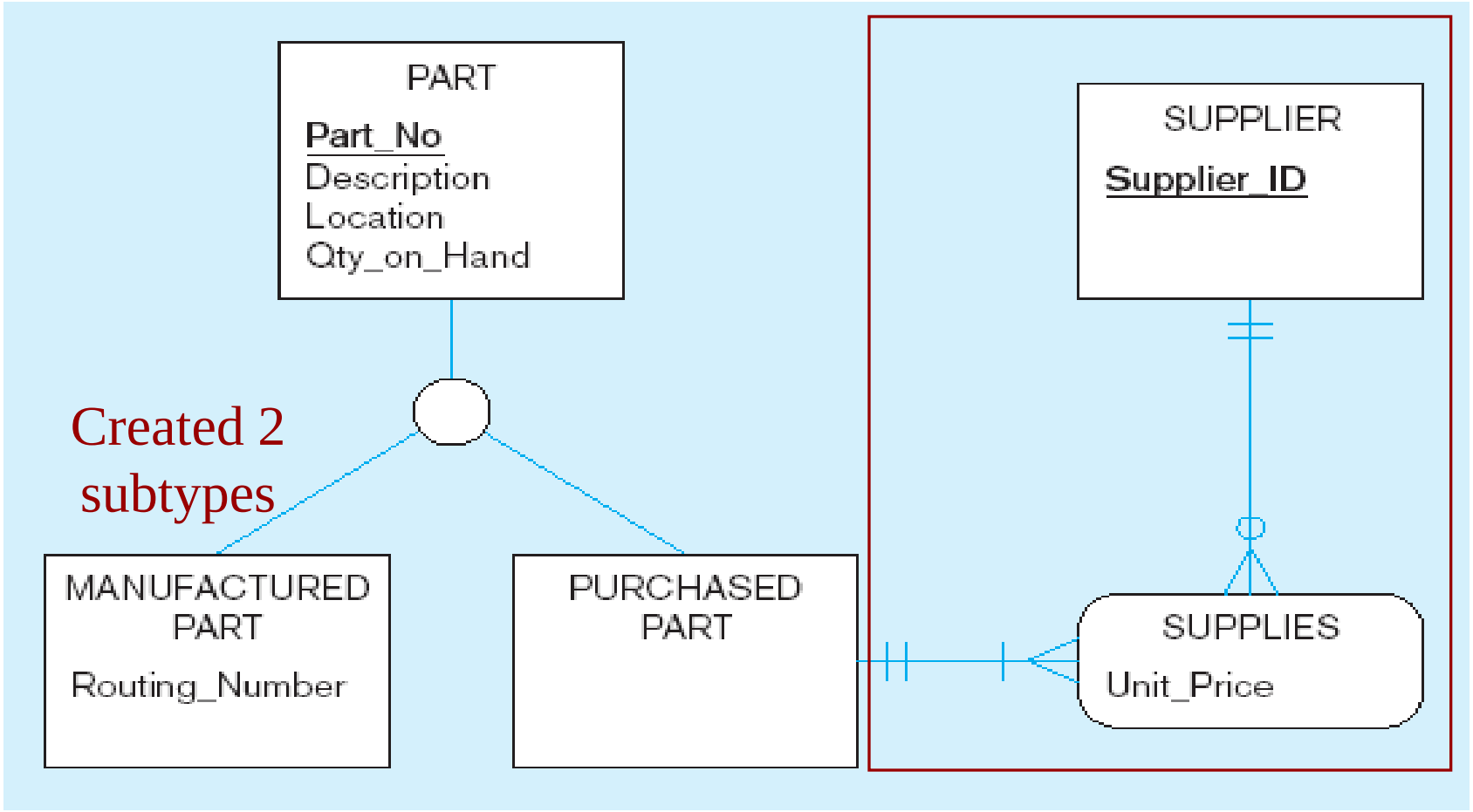
Example of specialization

Entity type PART



Example of specialization (cont.)

Specialization to MANUFACTURED PART and PURCHASED PART



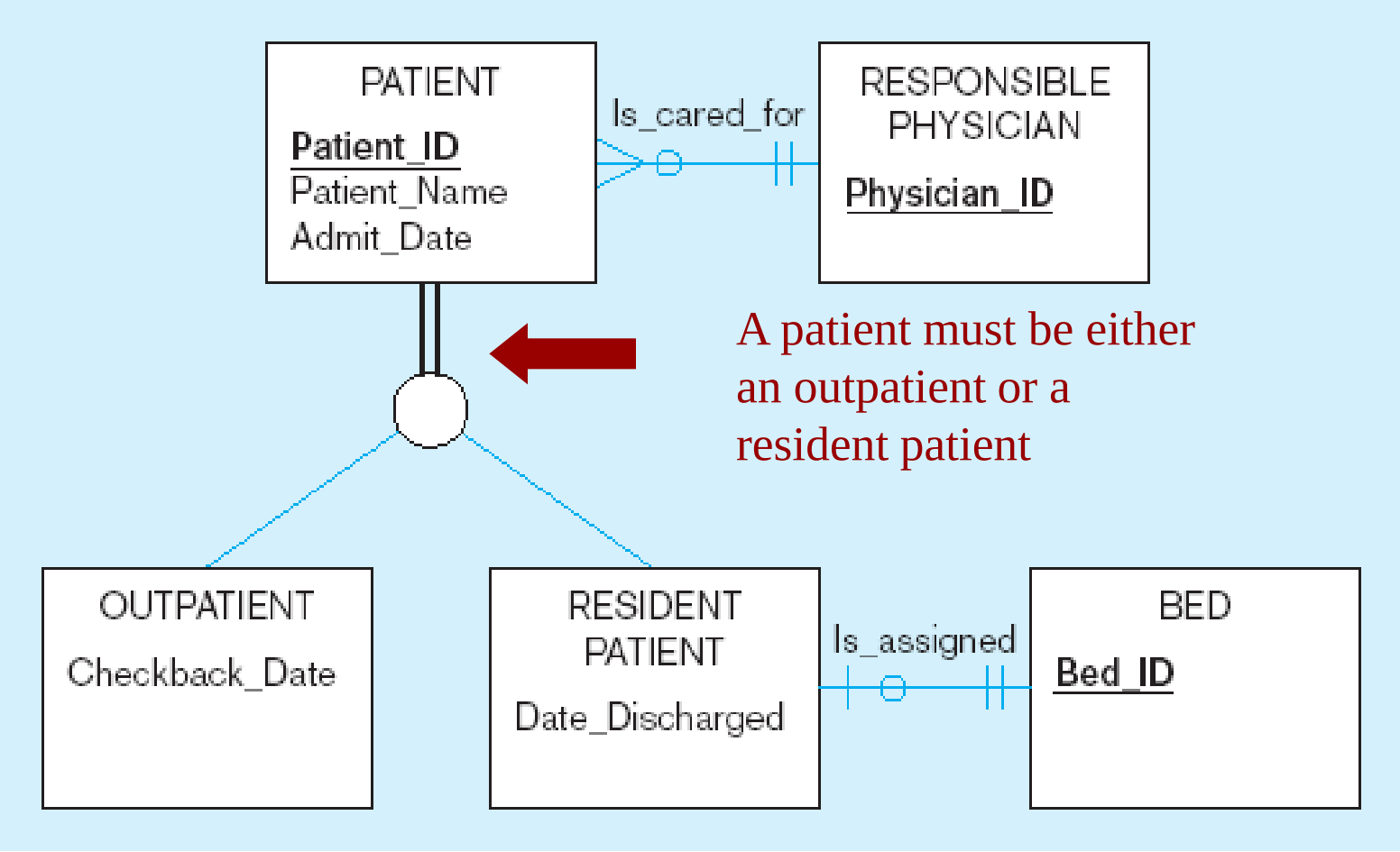
Note: multivalued attribute was replaced by an associative entity relationship to another entity

Constraints in Supertype/ Completeness Constraint

- Completeness Constraints: Whether an instance of a supertype must also be a member of at least one subtype
  - Total Specialization Rule: Yes (double line)
  - Partial Specialization Rule: No (single line)

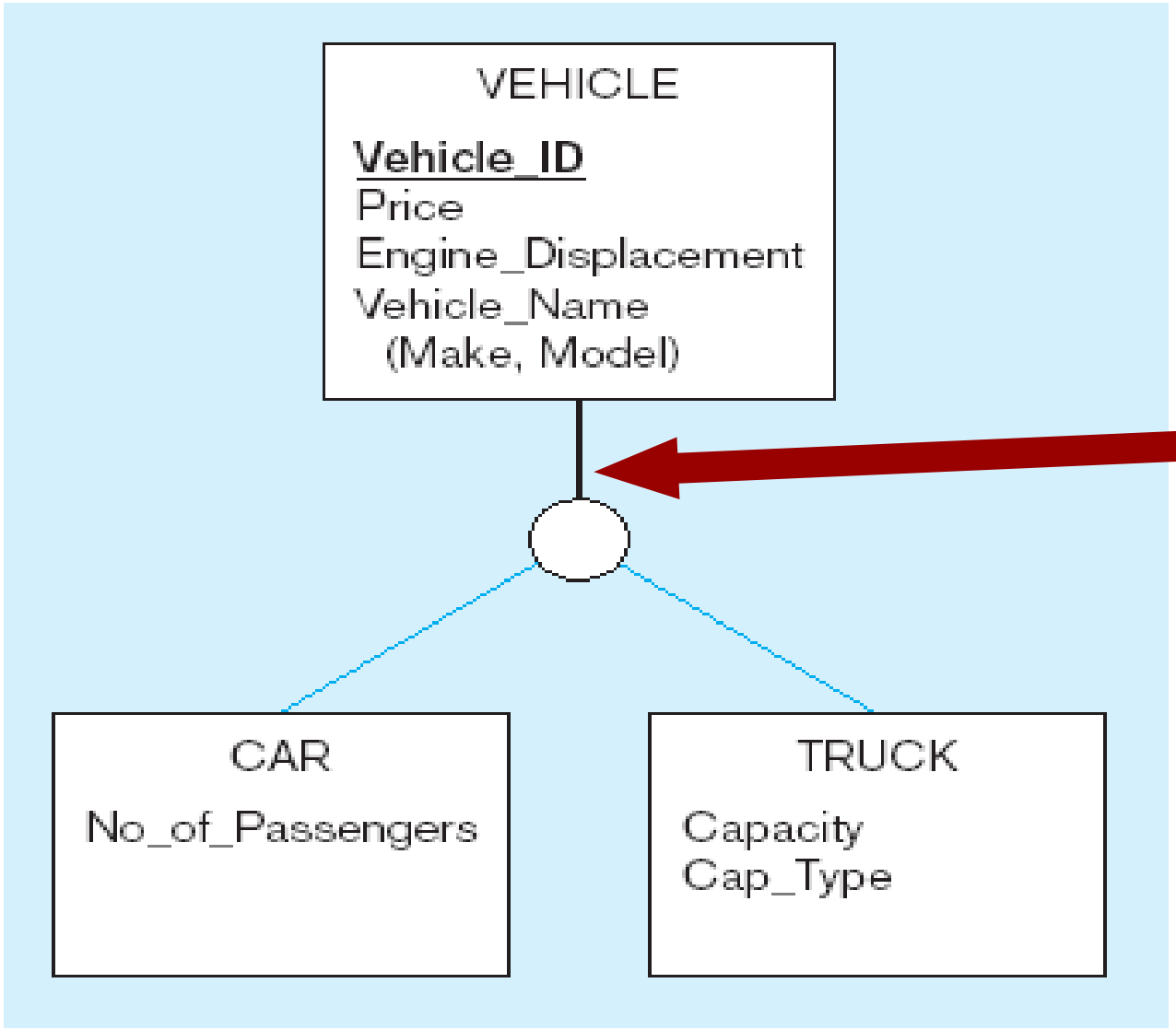
Examples of completeness constraints

Total specialization rule



Examples of completeness constraints (cont.)

Partial specialization rule



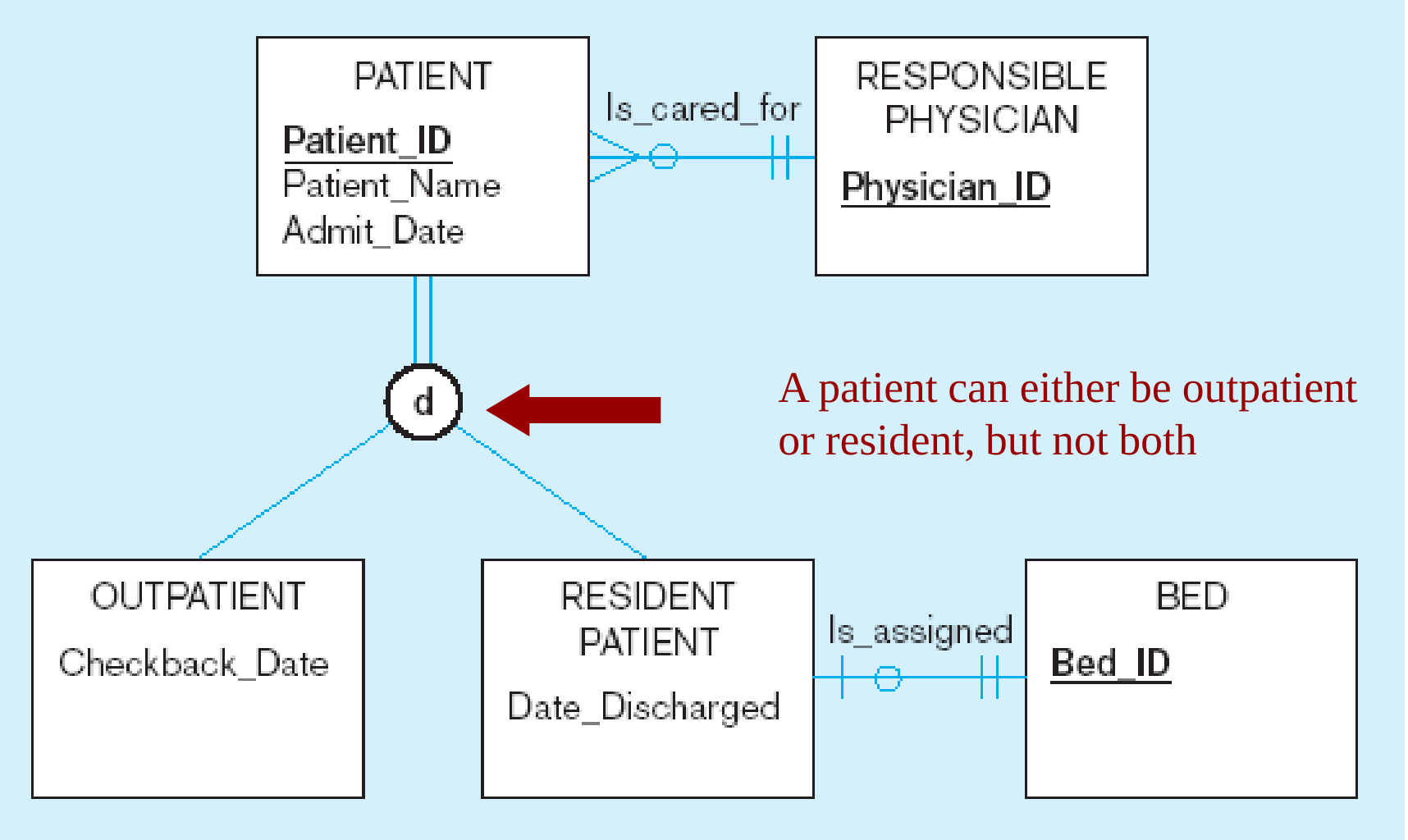
A vehicle could be a car, a truck, or neither

Constraints in Supertype/ Disjointness constraint

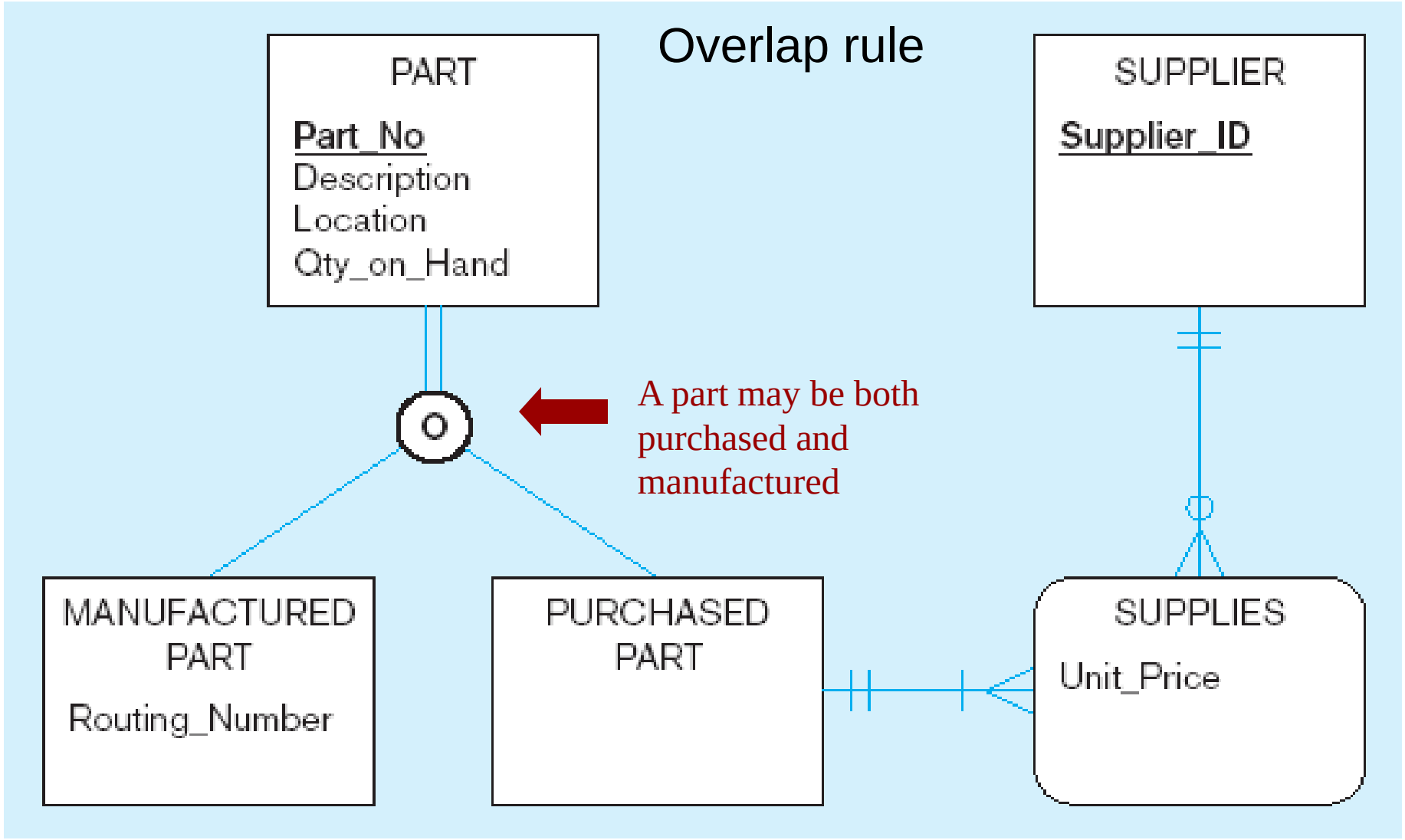
- Disjointness Constraints: Whether an instance of a supertype may simultaneously be a member of two (or more) subtypes
  - Disjoint Rule: An instance of the supertype can be only ONE of the subtypes
  - Overlap Rule: An instance of the supertype could be more than one of the subtypes

Examples of disjointness constraints

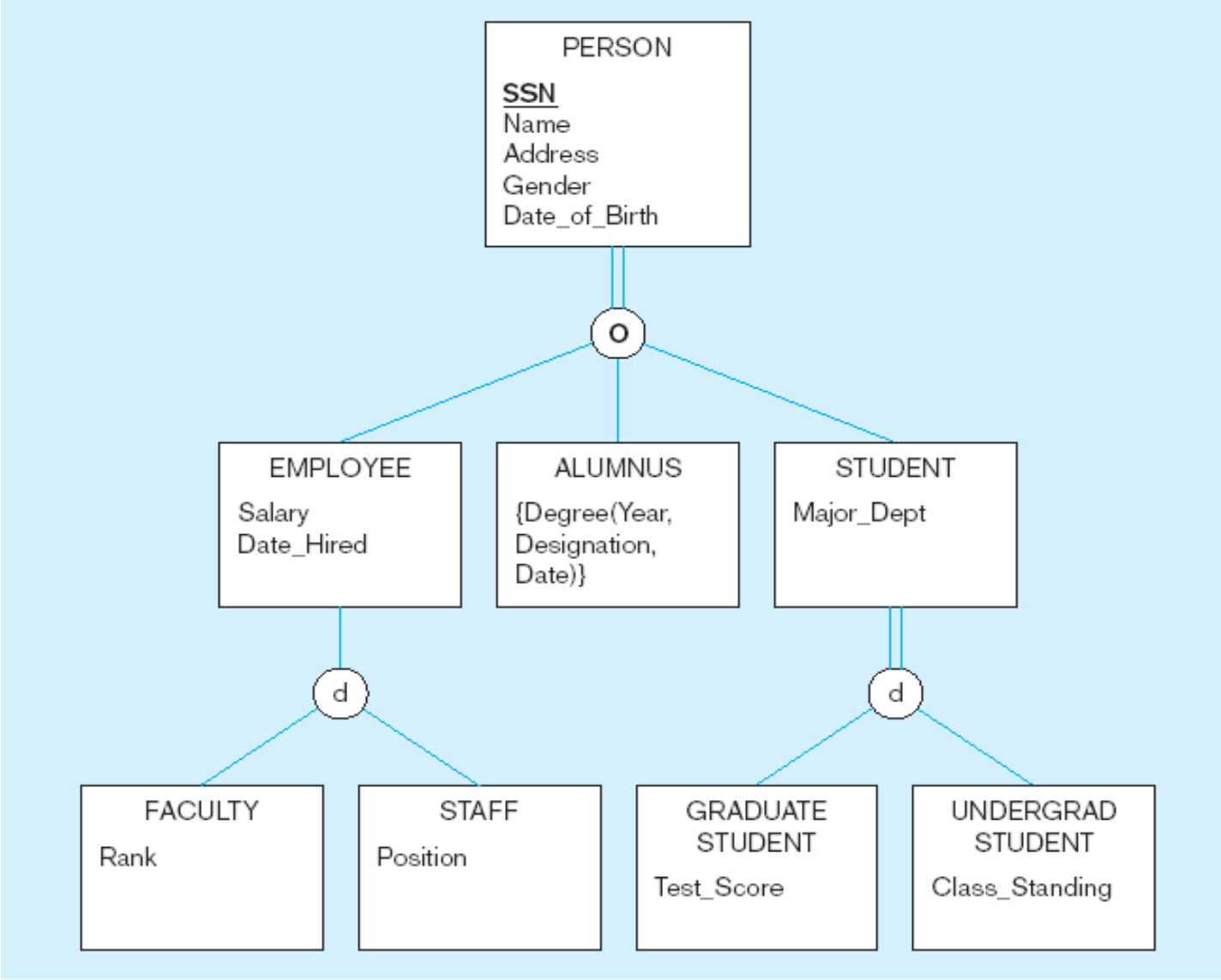
Disjoint rule



Examples of disjointness constraints (cont.)



Example of supertype/subtype hierarchy



Modeling of UNION Types Using Categories

- Union type or a category
- Represents a single superclass/subclass relationship with more than one superclass
- Subclass represents a collection of objects that is a subset of the UNION of distinct entity types
- Category can be total or partial

Specialization/Generalization Hierarchy

- Example of a Hierarchy:
  [STUDENT -> STUDENT_ATHLETE -> FOOTBALL_PLAYER -> {DEFENSIVE_PLAYER, OFFENSIVE_PLAYER}] shows a 3-level hierarchy
- Inheritance
  A subclass inherits the attributes and relationship types of not just the immediate parent, but also of the predecessor superclasses in the hierarchy all the way up to the root of the specialization hierarchy

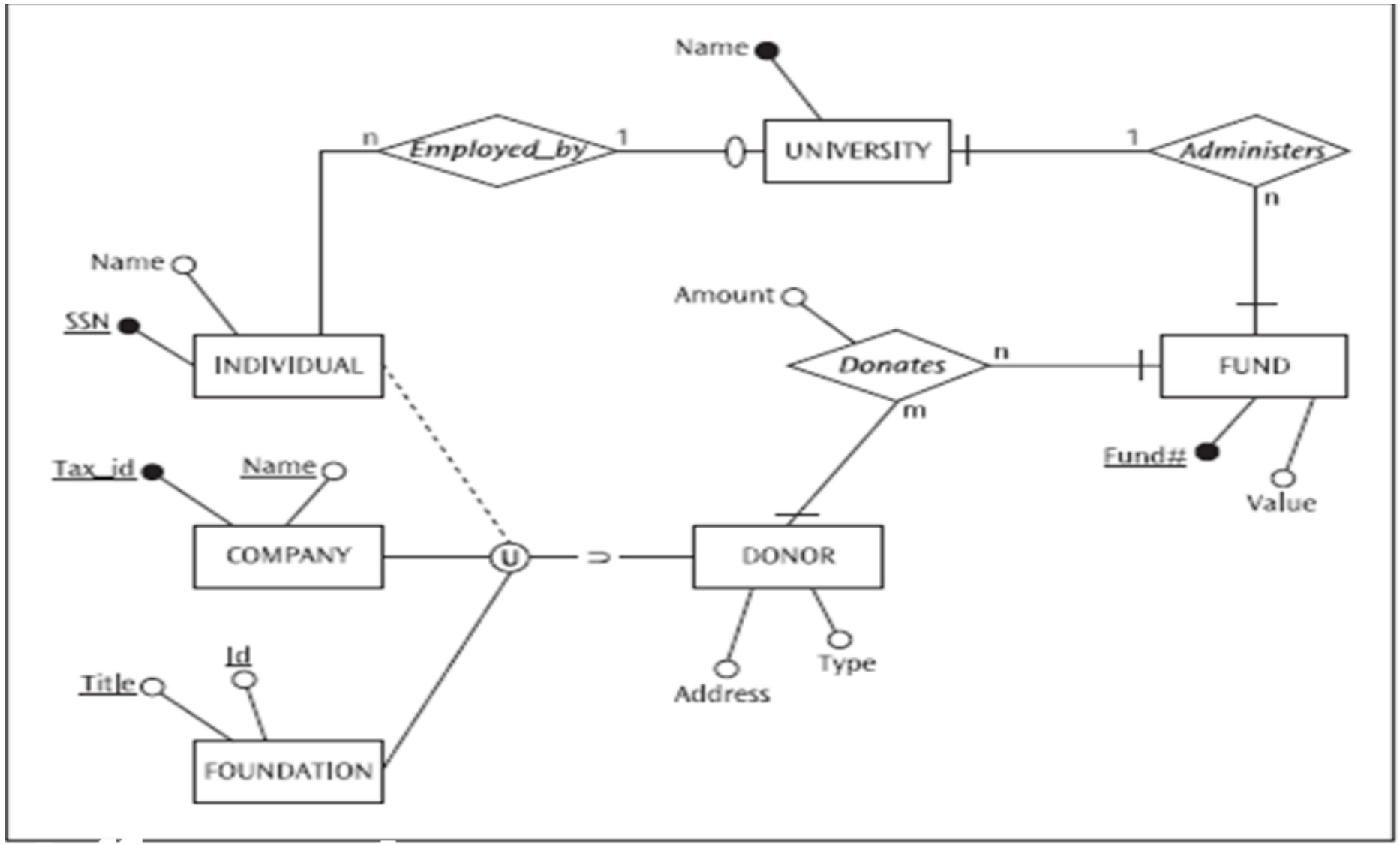
Specialization/Generalization Lattice

- In a specialization lattice, an entity type can participate as a subclass in more than one specialization (i.e., a child can have more than one parent)
- Inheritance
  The subclass will inherit all the attributes and relationship types from the superclasses of all the specializations participating in the lattice and the predecessor hierarchy of all these superclasses
  - This is called multiple type inheritance, and the subclass in the lattice is referred to as a shared subclass

The Categorization Construct

- A categorization occurs when a subclass is associated with more than one superclass of different entity types
  - The subclass is called category
- An entity that is a member of the category (subclass) must exist in ONLY ONE of the superclasses in the categorization relationship
- Example: A financial donor can be an individual, a company, or a foundation ← no “is-a” relationship!
- Notation: ‘U’ (= union, the subclass is a subset of the union of the superclasses)

Example for Category



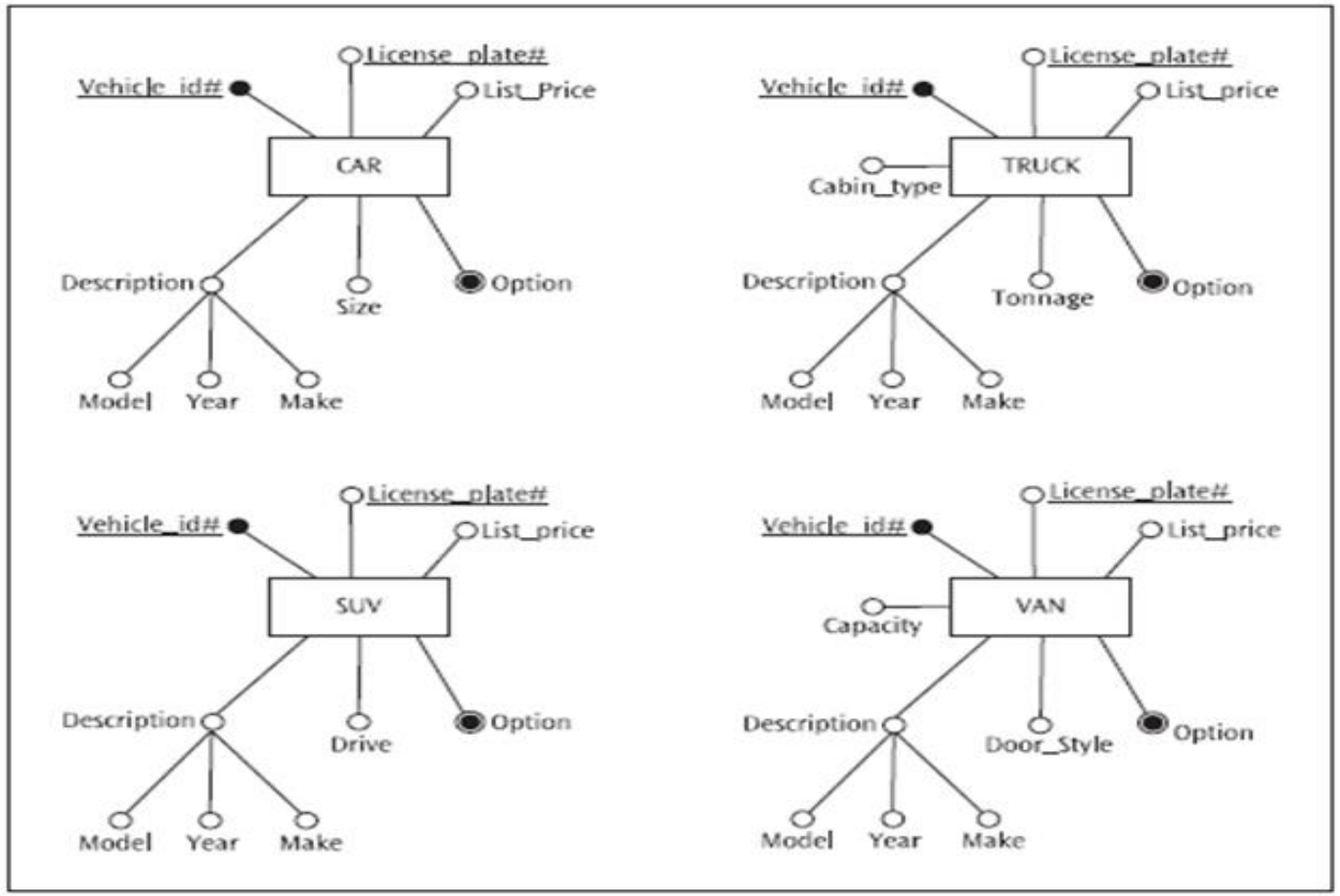
Characteristics of a Categorization

- There is only one subclass in each categorization
- The cardinality ratio is 1:1 within and across the SC/sc relationship
- The participation of the subclass in the categorization is always total

Characteristics of a Categorization (continued)

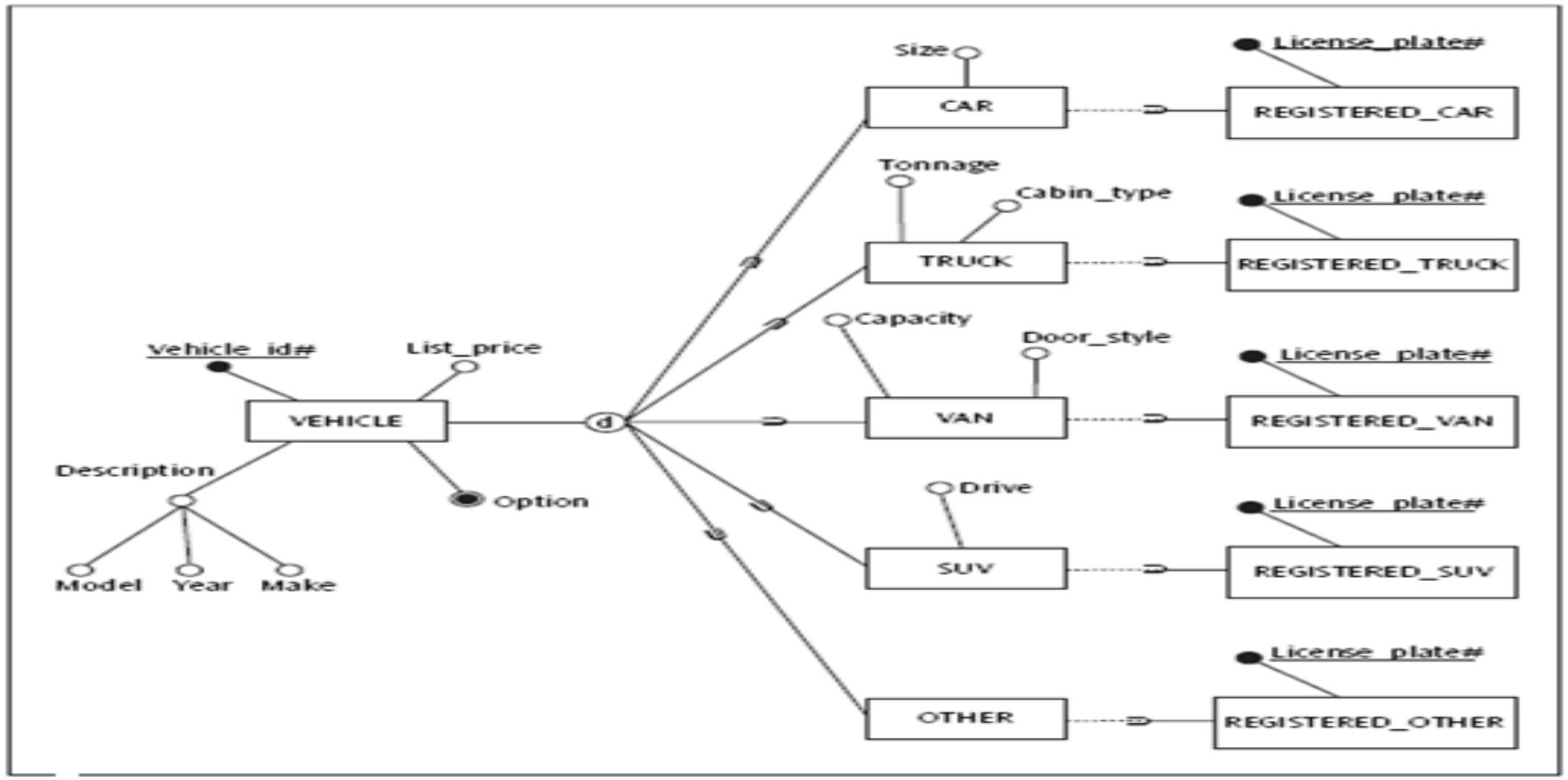
- Each superclass may exhibit either total or partial participation
- A category can either be:
  - A total category (i.e., the category is the union of all the superclass entities)
  - A partial category (i.e., the category is a true subset of the union of all the superclass entities)
- A category possesses the property of selective type inheritance (i.e., it inherits attributes of one entity type only)
- Often a unique identifier for a category must be manufactured (which is called a surrogate key)

Choosing the Appropriate EER Construct (continued)



A set of entity types CAR, TRUCK, VAN, and SUV

Choosing the Appropriate EER Construct (continued)

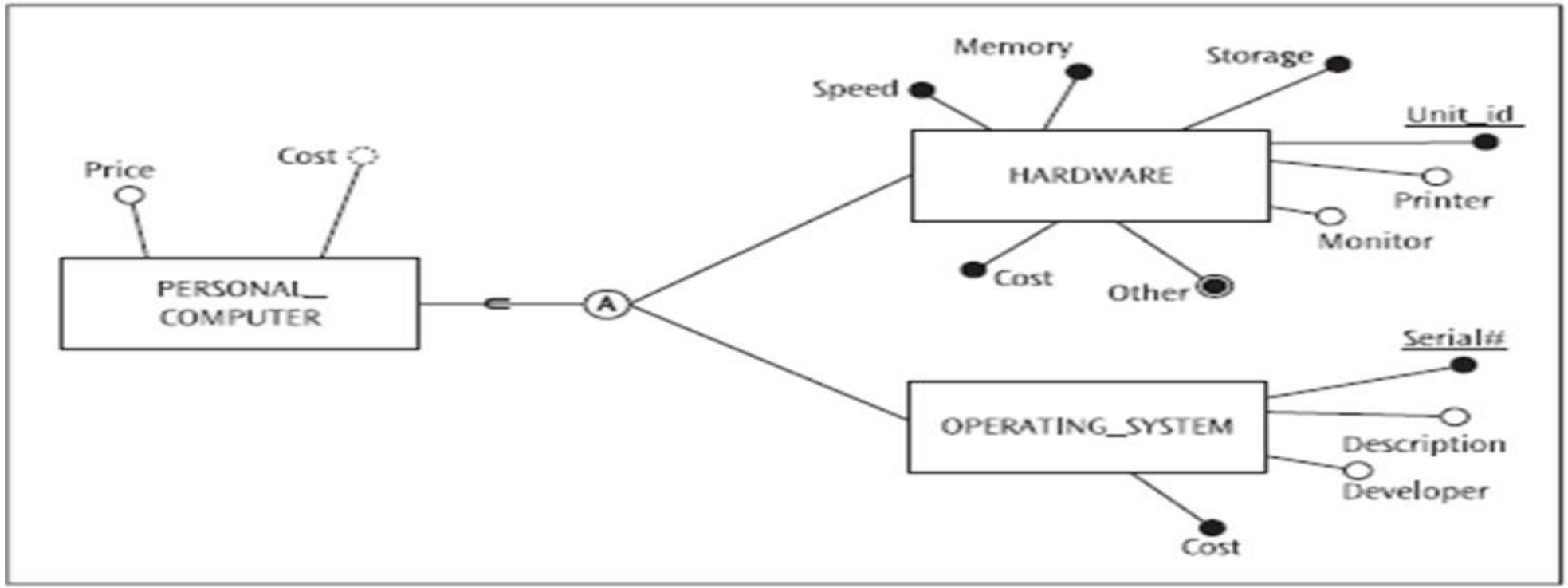


The partial category REGISTERED_VEHICLE in specialization expressed as a

The Aggregation Construct

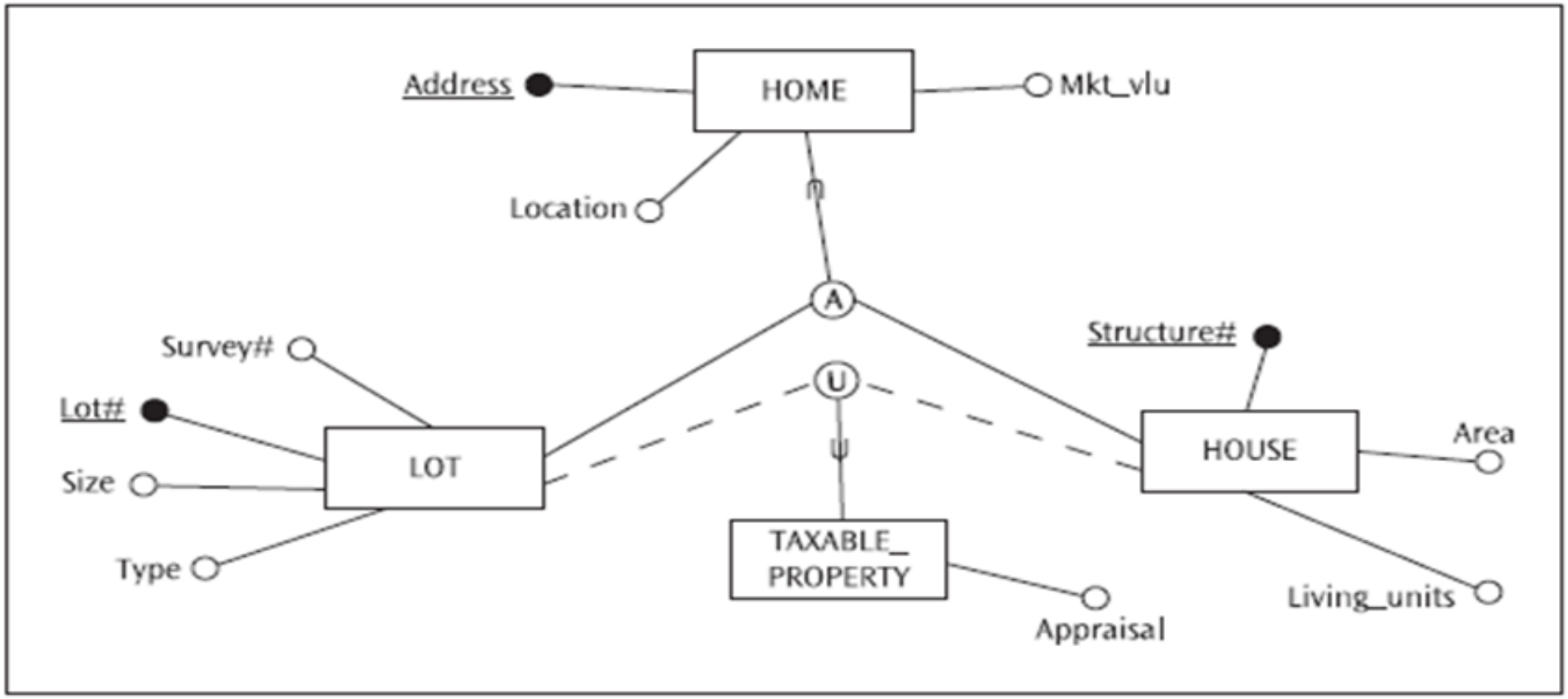
- An aggregation allows us to model a “whole/part” relationship as an “is-a-part-of” relationship between a subclass and a superclass
- An entity in the aggregate contains superclass entities from ALL SC/sc relationships in which it participates
- Inheritance: Selective type inheritance connotes the inheritance of attributes and relationships from ALL superclass entities contained in the specific aggregation
- Notation: ‘A’
- Read: “is-part-of”
- Note that an aggregate can never be partial

An Example of Aggregation



An example of the EER aggregation construct

Aggregation Versus Categorization



A category and an aggregate contrasted

Thank you