Chapter 4

Enhanced Entity-Relationship and UML Modeling (from E&N and my editing)

Outline

- Superclass/Subclass Relationship
- Specialization/Generalization
- Type
- Inheritance/Constraints/Updates
- Categorization
- Higher-Degree Relationships
- A bit UML

Enhanced-ER (EER) Model Concepts

- Includes all modeling concepts of basic ER
- Additional concepts: subclasses/superclasses, specialization/generalization, categories, attribute inheritance
- The resulting model is called the enhanced-ER or Extended ER (E2R or EER) model
- It is used to model applications more completely and accurately if needed
- It includes some object-oriented concepts, such as inheritance

Subclass

- Entity type describes:
  - Type of entity
  - The entity set
- Example: ‘EMPLOYEE’
- Employee can be sub-grouped into:
  - Secretary, Engineer, Technician, Manager
- These are called the subclass of EMPLOYEE entity type.

Superclass

- EMPLOYEE entity type is the super class of
  - engineer, secretary & technician class.
- Subclass represent the same mini-world entity of the superclass, but in a distinct specific role.
- Entity in a subclass must be a member of a superclass, but not vice-versa! Example…?

Subclasses and Superclasses (1)

- An entity type may have additional meaningful subgroupings of its entities
- Example: EMPLOYEE may be further grouped into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE,…
  - Each of these groupings 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 relationships.
- Example: EMPLOYEE/SECRETARY, EMPLOYEE/TECHNICIAN

Subclasses and Superclasses (2)

- These are also called IS-A relationships (SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, …).
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass
  - The Subclass member is the same entity in a distinct specific role
  - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
  - A member of the superclass can be optionally included as a member of any number of its subclasses
- Example: A salaried employee who is also an engineer belongs to the two subclasses ENGINEER and SALARIED_EMPLOYEE
  - It is not necessary that every entity in a superclass be a member of some subclass

Attribute Inheritance in Superclass / Subclass Relationships

- An entity that is member of a subclass *inherits* all attributes of the entity as a member of the superclass
- It also inherits all relationships

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: Another specialization of EMPLOYEE based in *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
  - Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams
  - Attributes of a subclass are called specific attributes. For example, TypingSpeed of SECRETARY
  - The subclass can participate in specific relationship types. For example, BELONGS_TO of HOURLY_EMPLOYEE

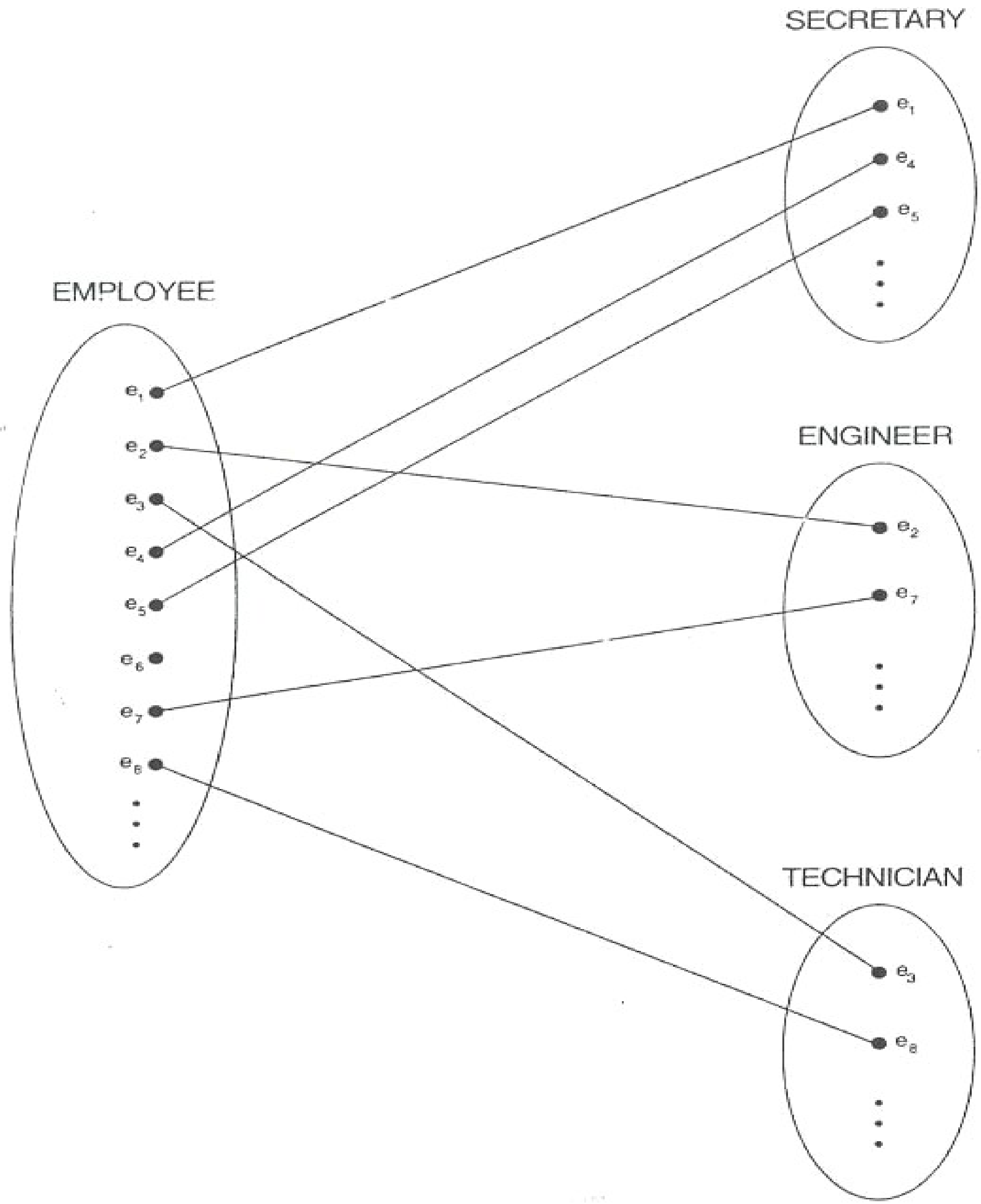
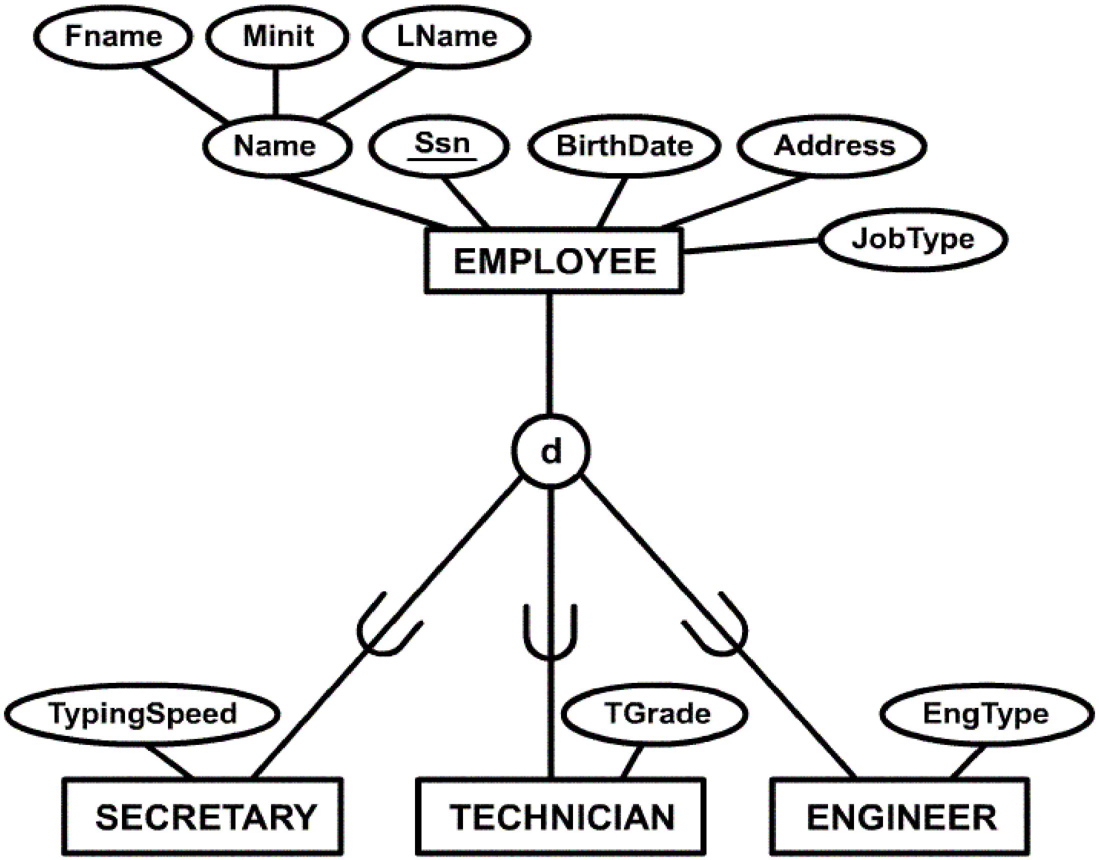


Figure 4.2 Some instances of the specialization of EMPLOYEE into the {SECRETARY, ENGINEER, TECHNICIAN} set of subclasses.

Example of a Specialization



Notations of EER Diagram

- Subset symbol ⊂
- Specific attributes, or local attributes
- Specific relationships
- class/subclass EMPLOYEE/SECRETARY resembles 1:1 relationship at the instance level, of *one* entity. playing a specialized role, an EMPLOYEE specialized the role of SECRETARY

Benefits of Specialization

- Define a set of subclasses of an entity type
- Establish additional specific attributes with each subclass
- Establish additional specific relationship types between each subclass and other entity types or other subclasses
- Refer to the EER diagram…!

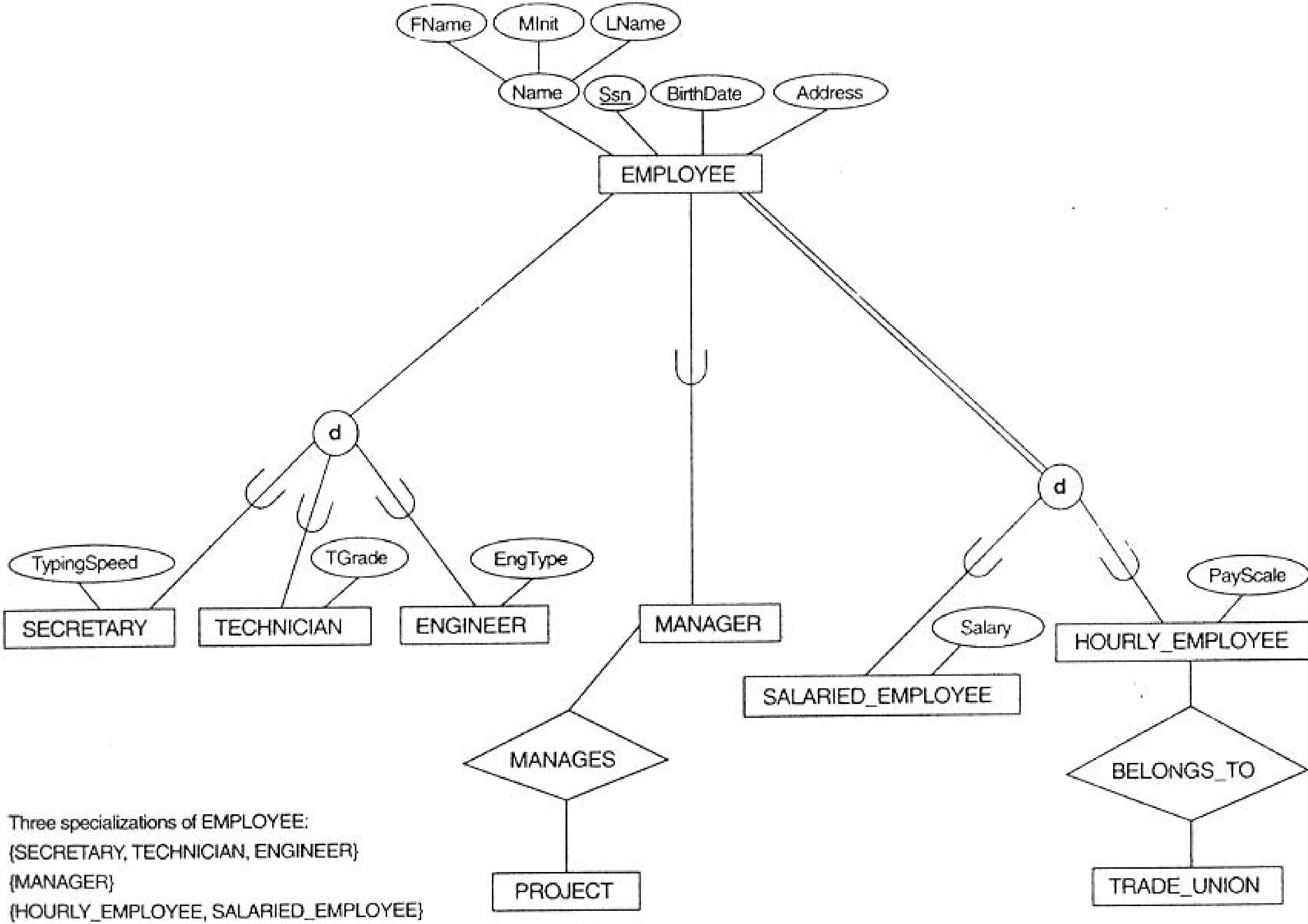


Figure 4.1 EER diagram notation for representing specialization and subclasses.

Generalization

- Identify common features (attributes), and generalize into a superclass
- Example: truck & car can be generalized into VEHICLE
- Inverse of the specialization process

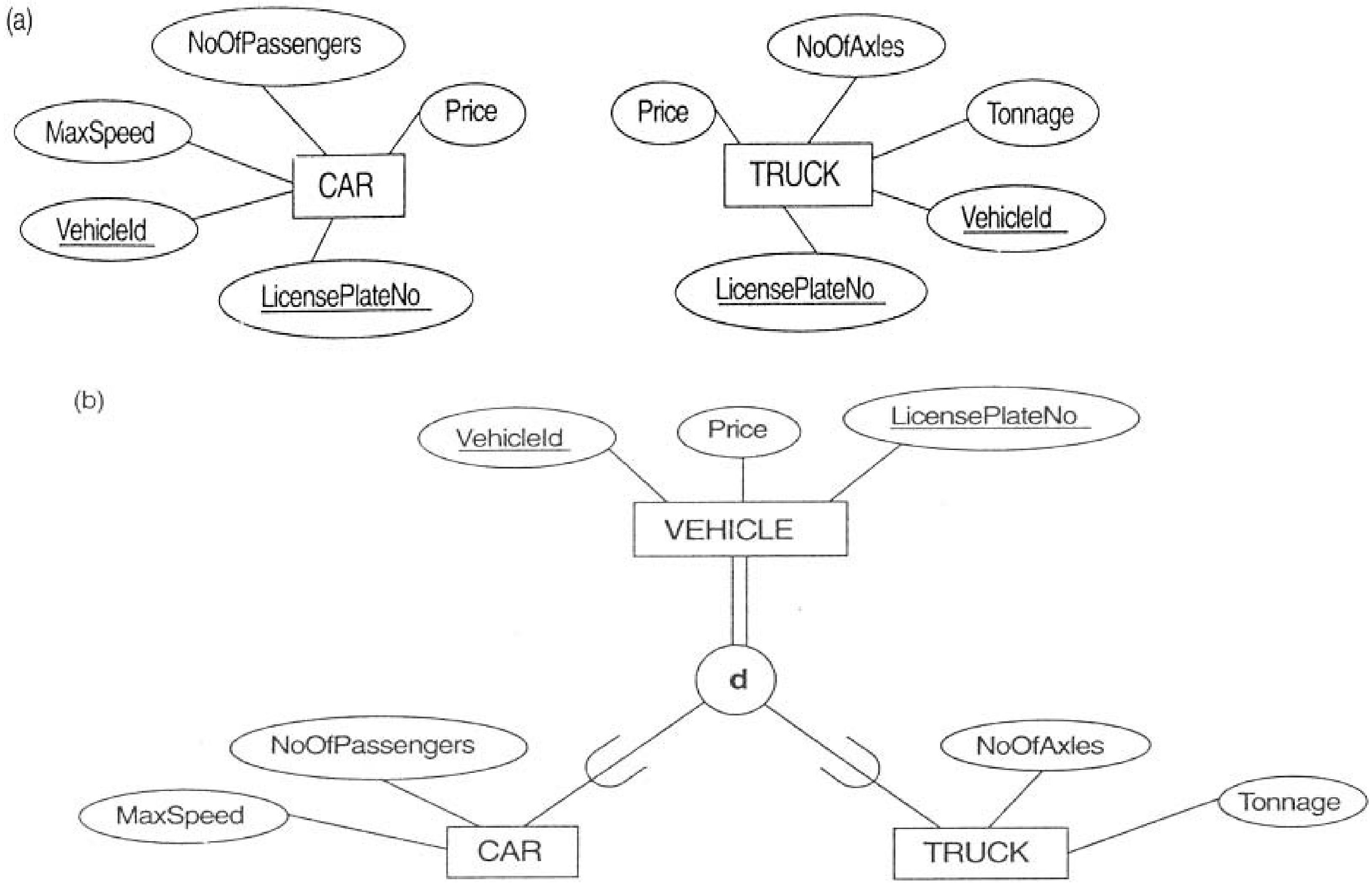


Figure 4.3 Examples of generalization. (a) Two entity types CAR and TRUCK. (b) Generalizing CAR and TRUCK into VEHICLE.

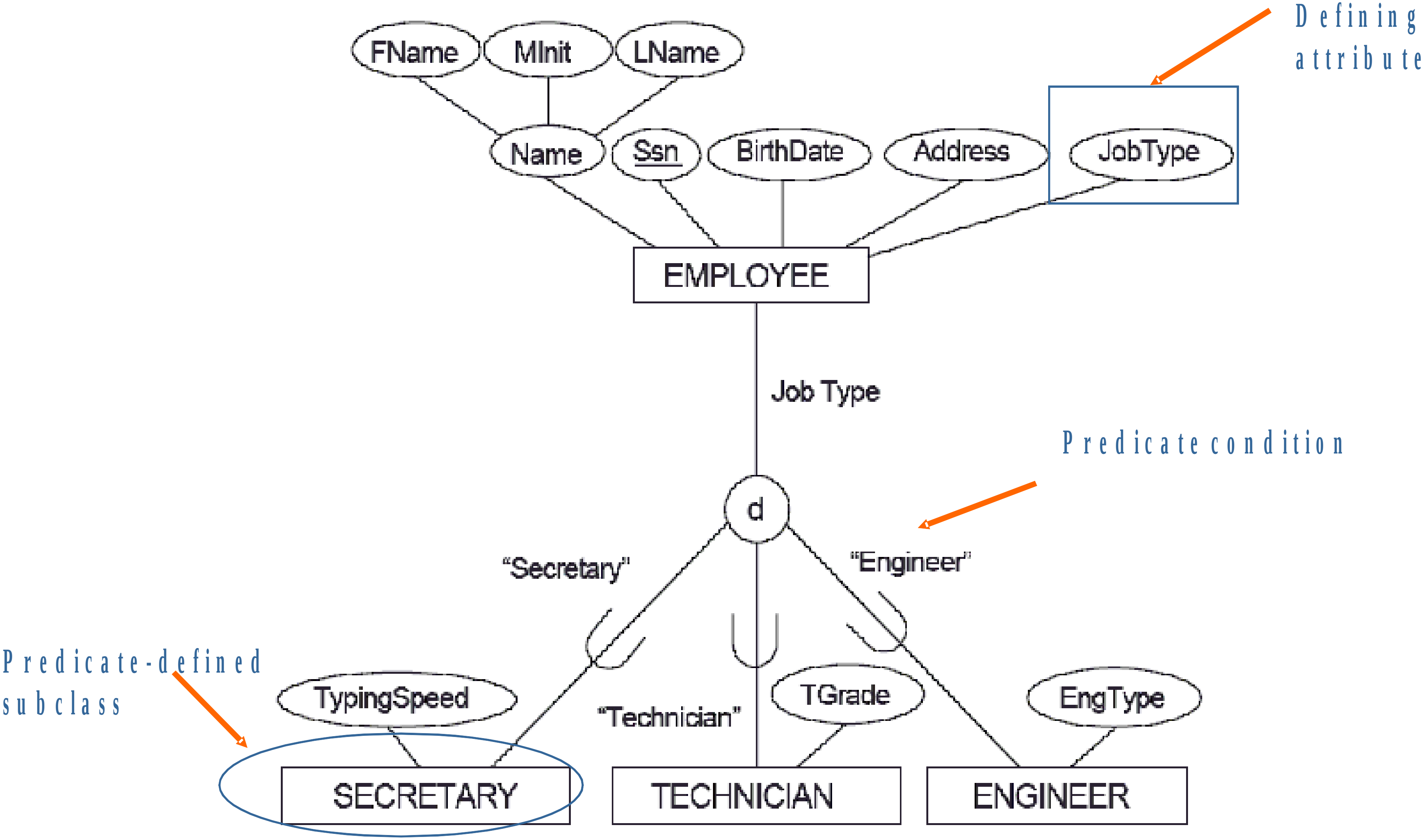
Generalization and Specialization

- Diagrammatic notation sometimes used to distinguish between generalization and specialization
  - Arrow pointing to the generalized superclass represents a generalization
  - Arrows pointing to the specialized subclasses represent a specialization
  - We do not use this notation because it is often subjective as to which process is more appropriate for a particular situation
  - We advocate not drawing any arrows in these situations
- Data Modeling with Specialization and Generalization
  - A superclass or subclass represents a set of entities
  - Shown in rectangles in EER diagrams (as are entity types)
  - Sometimes, all entity sets are simply called classes, whether they are entity types, superclasses, or subclasses

Constraints on Specialization and Generalization

- Attribute-defined specialization
  - Base on values of a superclass attribute (defining attribute)
  - All subclasses have their member condition on the same attribute of the superclass
  - Predicate-defined (condition defined) subclass
  - JobType = ‘Engineer’ => defining predicate
- User-defined Subclass
  - Each membership is determined by the user

Figure 4.4 An attribute-defined specialization on the JobType attribute of EMPLOYEE.



Constraints on Specialization and Generalization (2)

- Two other conditions apply to a specialization/generalization:
- **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
- **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

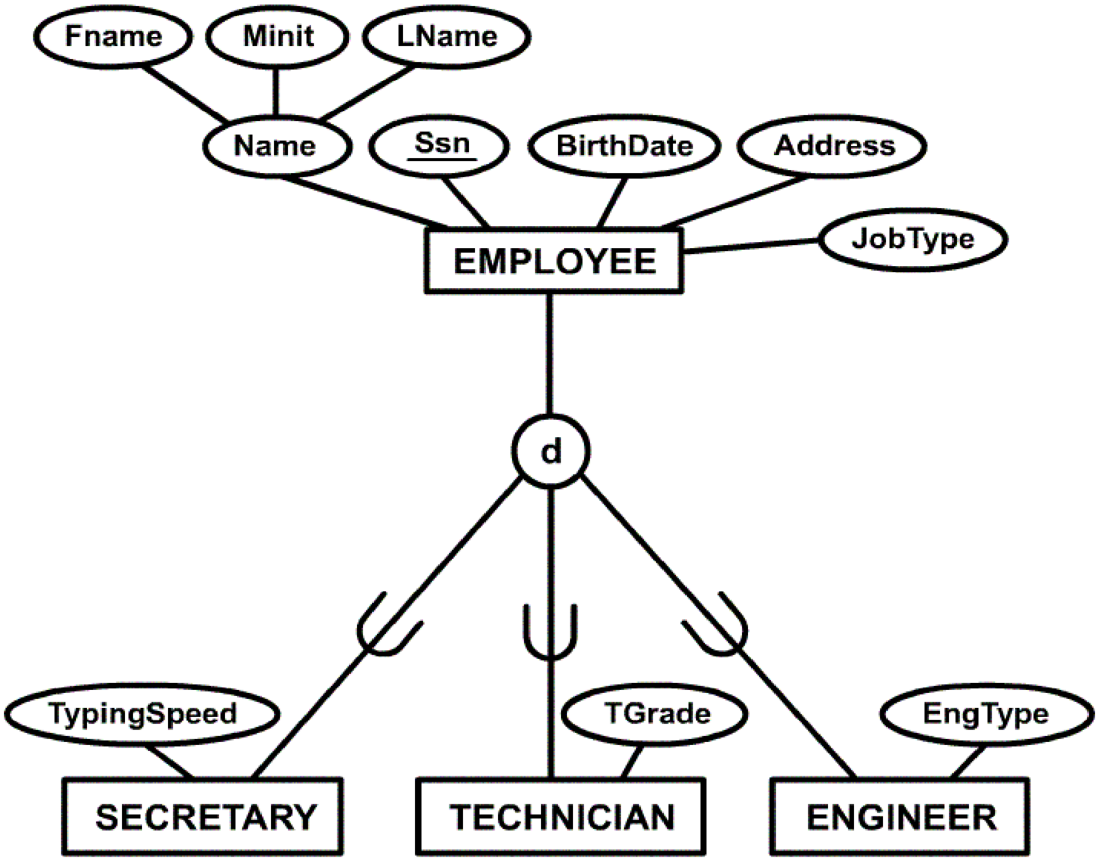
Constraints on Specialization and Generalization (3)

- 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.

Disjoint Contraints

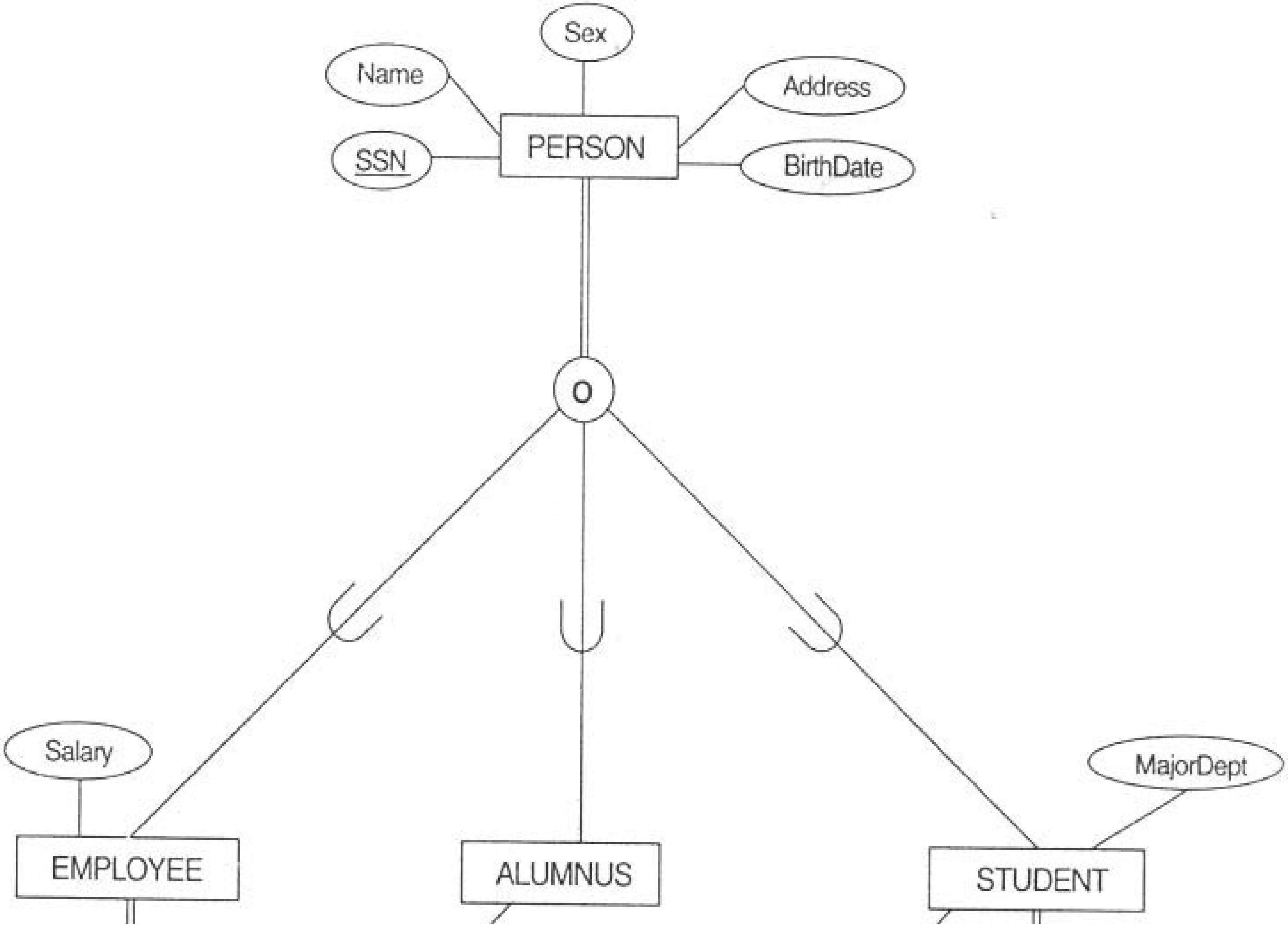
- Subclasses of a specialization must be disjoint
- An entity can only be at most one of the subclass
- Look at the previous EER diagram
- Use (d)

Example of disjoint partial Specialization



Overlap

- The same entity may be a member of more than one subclass of the specialization
- Use the (o)
- Example, a person can be:
  - A student
  - A faculty member
  - An alumni


Sex
Name
Address
SSN
PERSON
BirthDate
o
Salary
MajorDept
EMPLOYEE
ALUMNUS
STUDENT

Completenes Constrains

- Total specialization:
  - Every entity in the superclass must be a member of some subclass
  - Example, the Salaried_Employee and Hourly_Employee
  - Shown using double line

- Partial specialization:
  - Allows an entity not to belong to any subclass
  - Example:
    - Manager
    - Job type
  - Use single line

Rules

- Deleting entity from a superclass → deletes it also from the subclasses
- Inserting in a superclass, when attribute defined is filled → must insert to the proper subclass as well
- Inserting in superclass of total specialization → must insert into at least one subclass

Hierarchy and Lattice

- Hierarchy: a subclass only participates in *one* class/subclass relationship
- Example: Vehicle with Car and Trucks
- Lattice: a subclass can participate in more than one class/subclass relationship
- Example: an Engineering Manager, must be an Engineer, and also a Manager!
- The concept of multiple inheritance

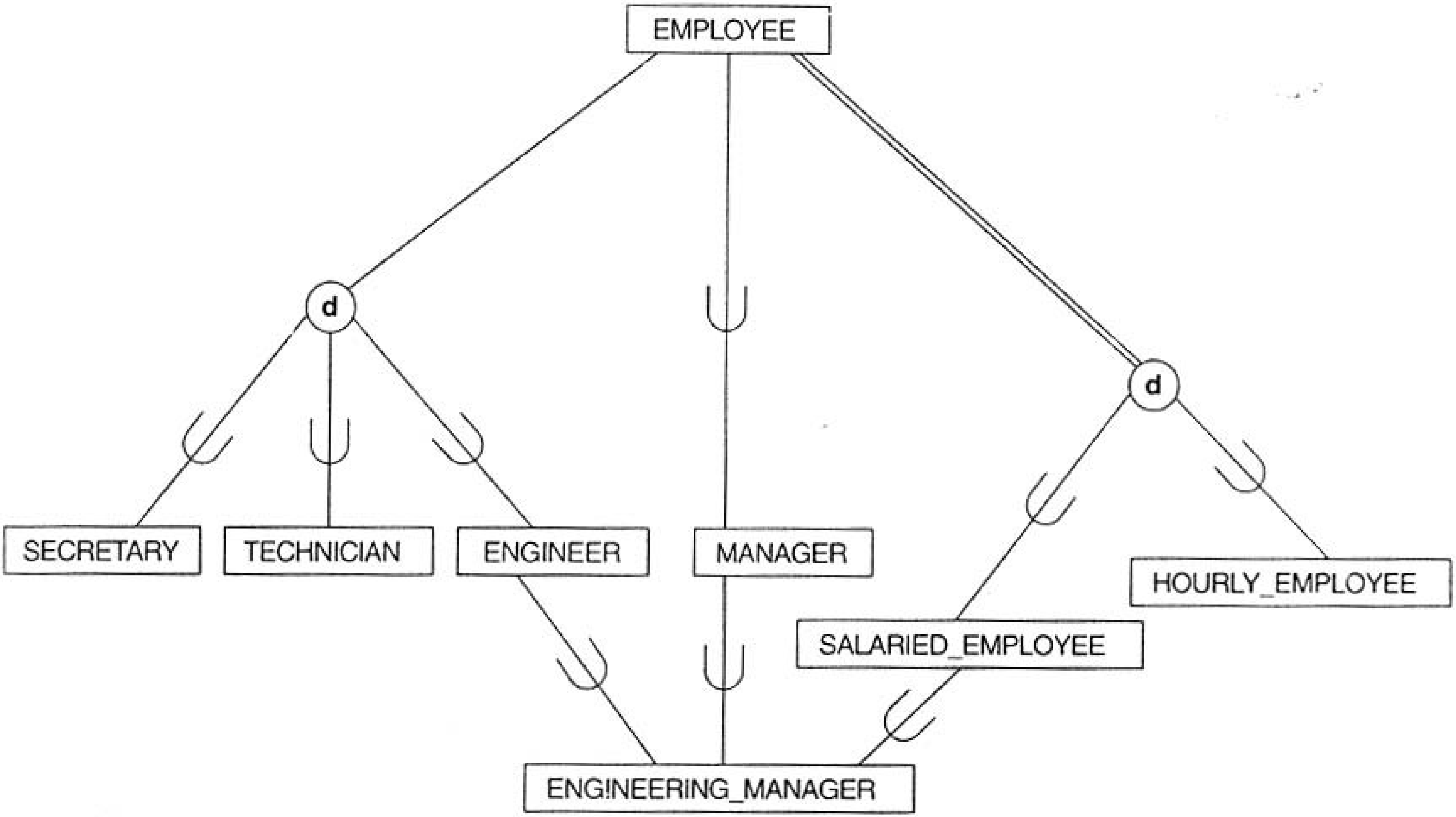


Figure 4.6 A specialization lattice with the shared subclass ENGINEERING_MANAGER.

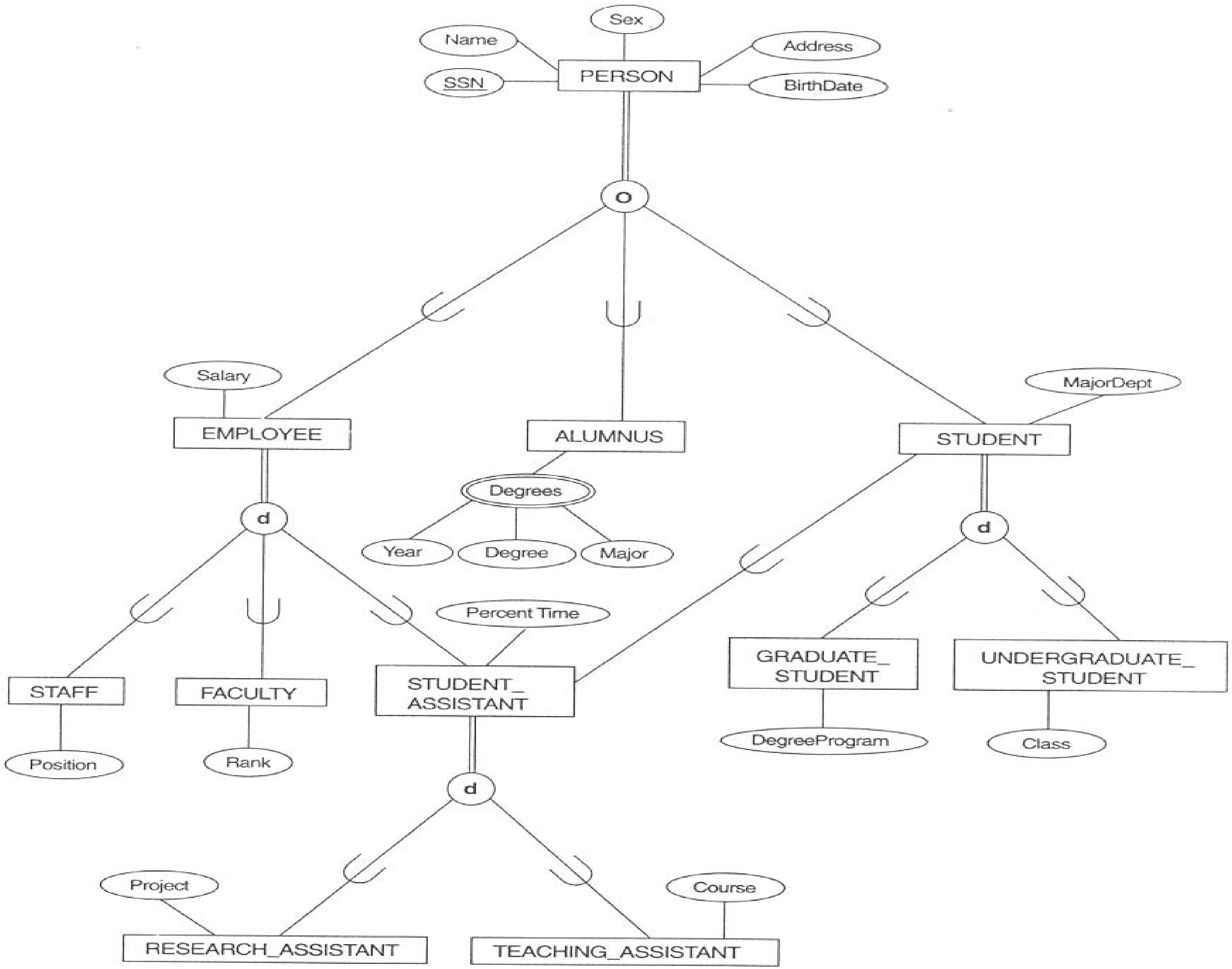


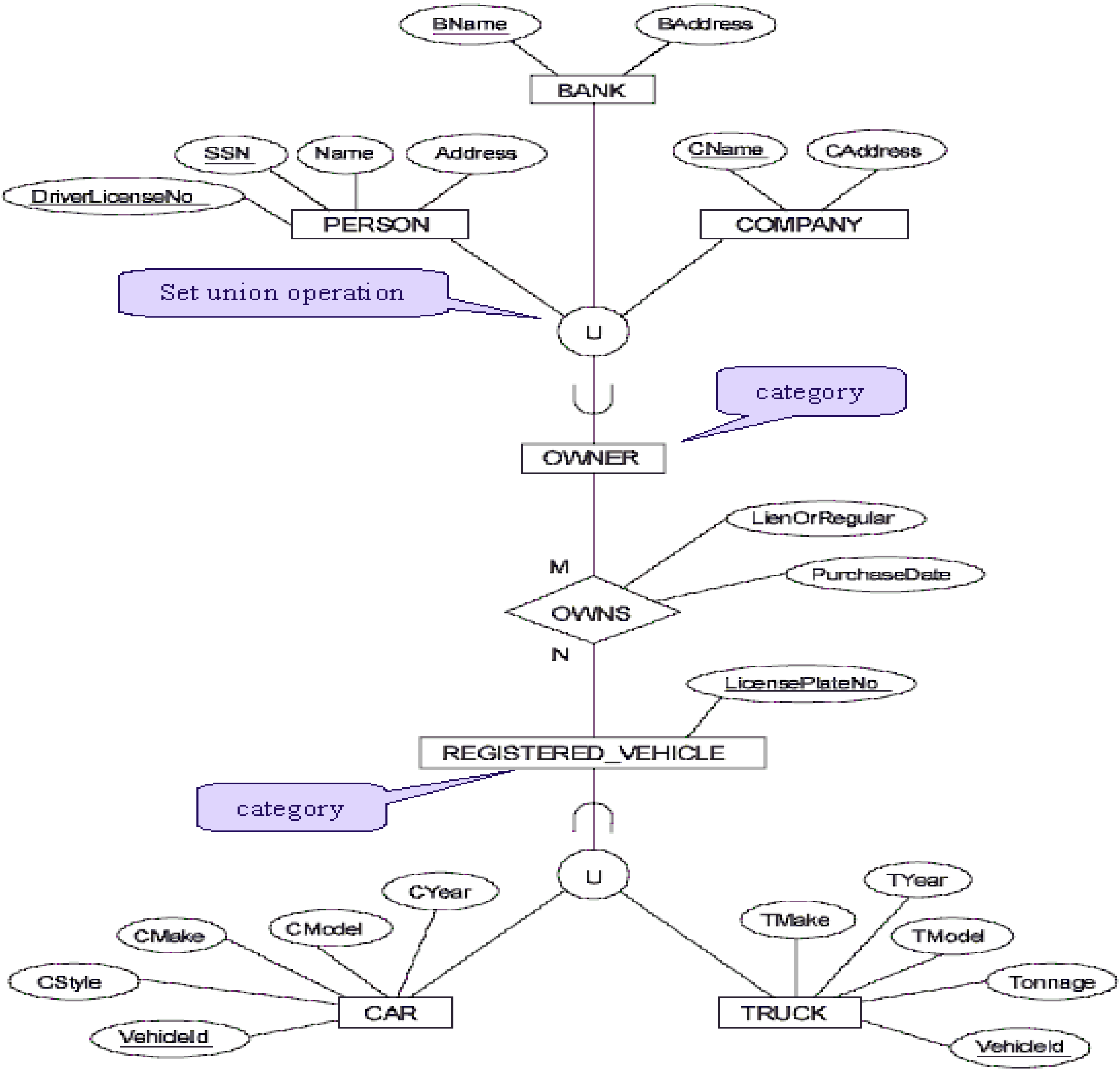
Figure 4.7 A specialization lattice (with multiple inheritance) for a UNIVERSITY database.

More explaination of EER

- Leaf node: class that has no subclasses
- An entity may exist in several leaves
  - Example, a student as Graduate_Student and a Teaching_Assistant
- Multiple inheritance:
  - Student_assistant
  - But the 'Person' attribute is only inherited once

Union Types Using Categories

- Engineering_Manager has 3 distinct relation, each relation has 1 superclass
- In our new case, a subclass has a single relationship with 3 distinct superclass.
- The subclass represent collection of objects, which we call union type or category


BName
BAddress
BANK
SSN
Name
Address
DriverLicenseNo
PERSON
CName
CAddress
COMPANY
Set union operation
U
category
OWNER
LienOrRegular
M
PurchaseDate
OWNS
N
LicensePlateNo
REGISTERED_VEHICLE
category
U
CYear
CModel
CMake
CStyle
VehicleId
CAR
TYear
TMake
TModel
Tonnage
TRUCK
VehicleId

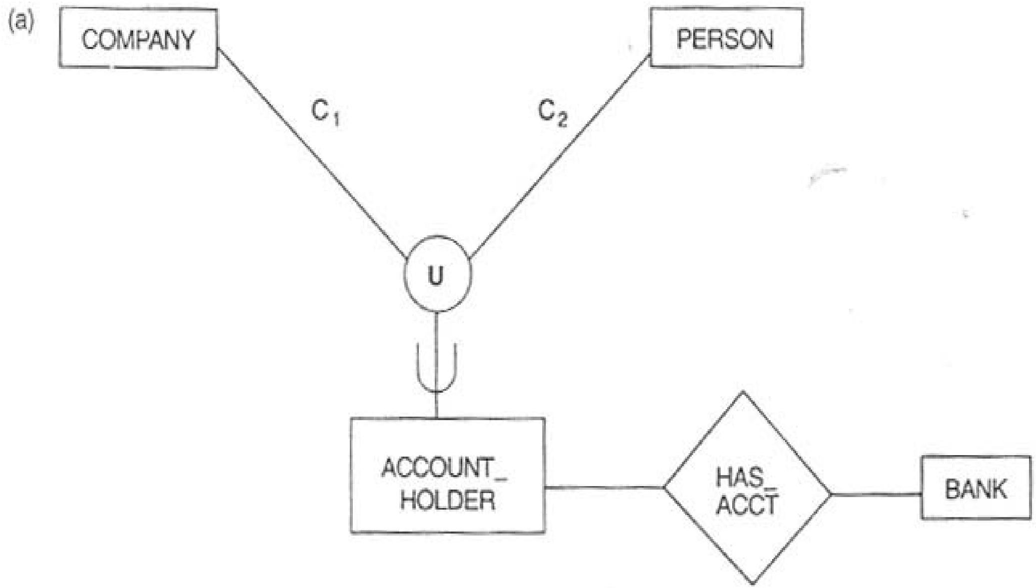
- A category OWNER is a union subclass of COMPANY, BANK and PERSON
- Use the (U) symbol
- Registered_Vehicle is a union subclass of Car & Truck

The Difference...

- Engineering_Manager must exist in all three superclass: Manager, Engineer, Salaried_Employee
- Owner, must exist in only *one* superclasses
- Engineering_Manager: inherited all superclasses attributes
- Owner, selective attribute inheritance, depending on the superclass

Partial Category

- Partial category, may or may not participate in the relation



Total Category

- Must be one of the superclasses
- Example: A building and a l must be a member of PROPERTY
- May be represented as a generalization (d), especiall when the similarity is numerous

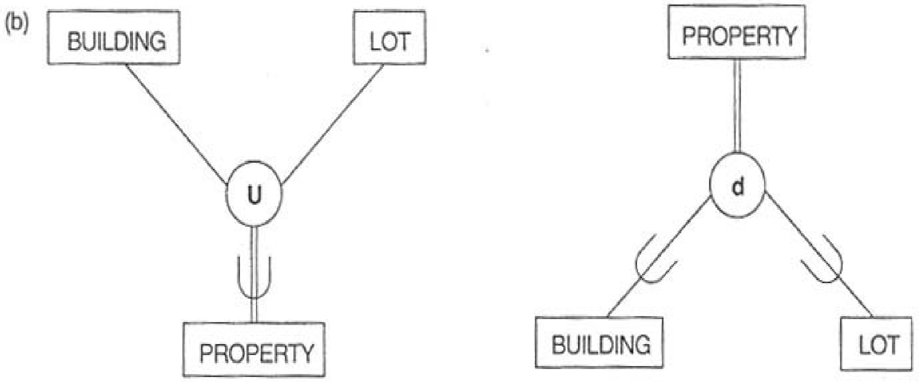


Figure 4.9 Total and partial categories. (a) Partial category ACCOUNT_HOLDER that is a subset of the union of two entity types COMPANY and PERSON. (b) Total category PROPERTY and a similar generalization.

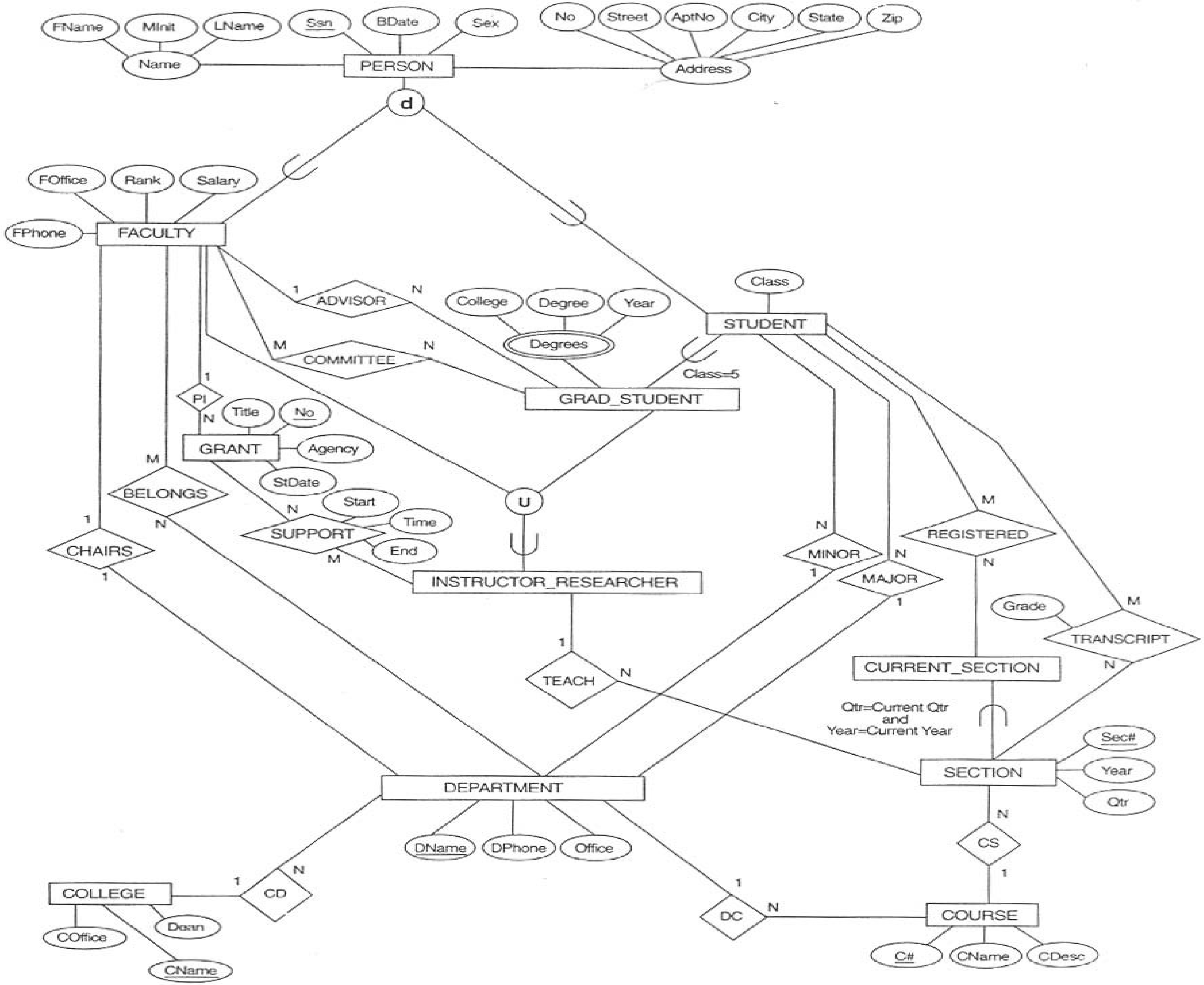
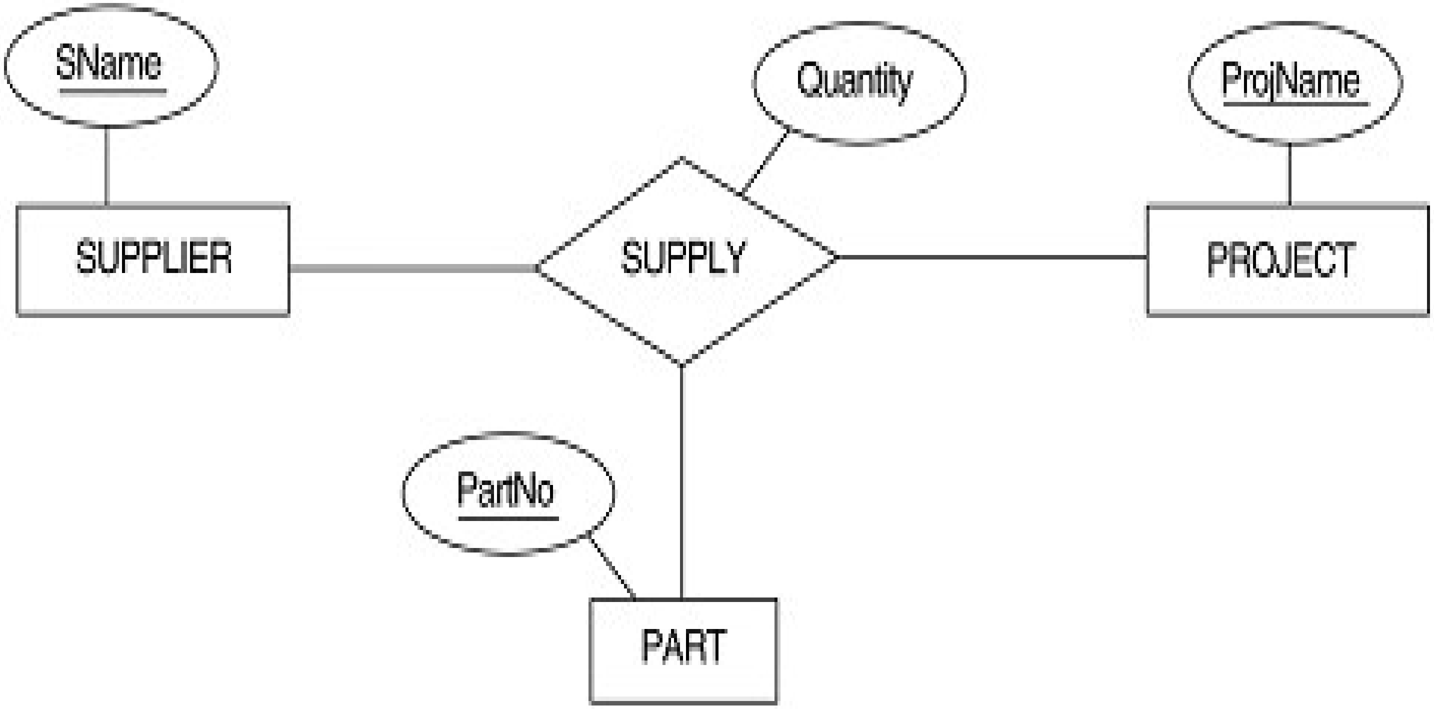


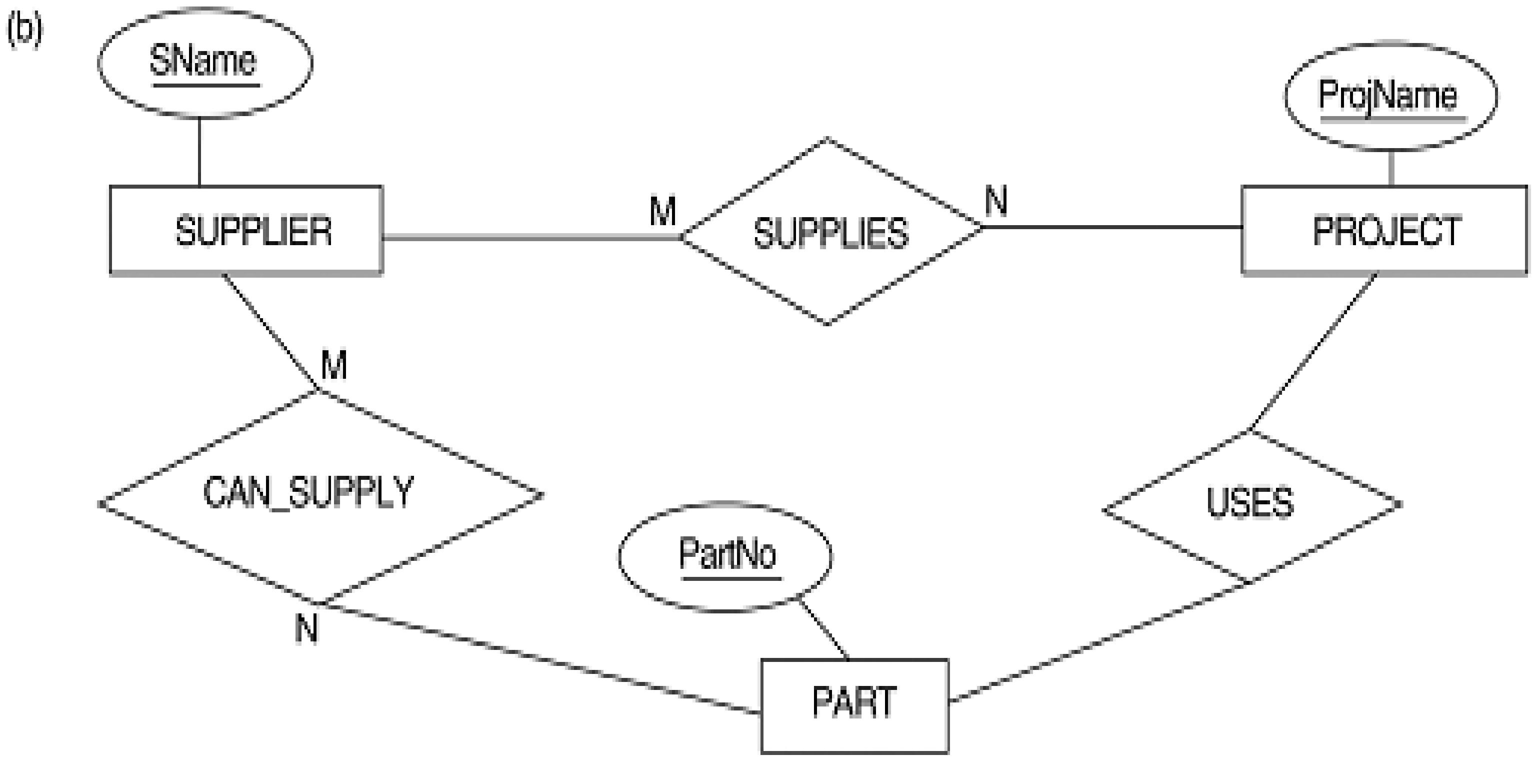
Figure 4.10 An EER conceptual schema for a UNIVERSITY database.

Hinger Degree Relationship

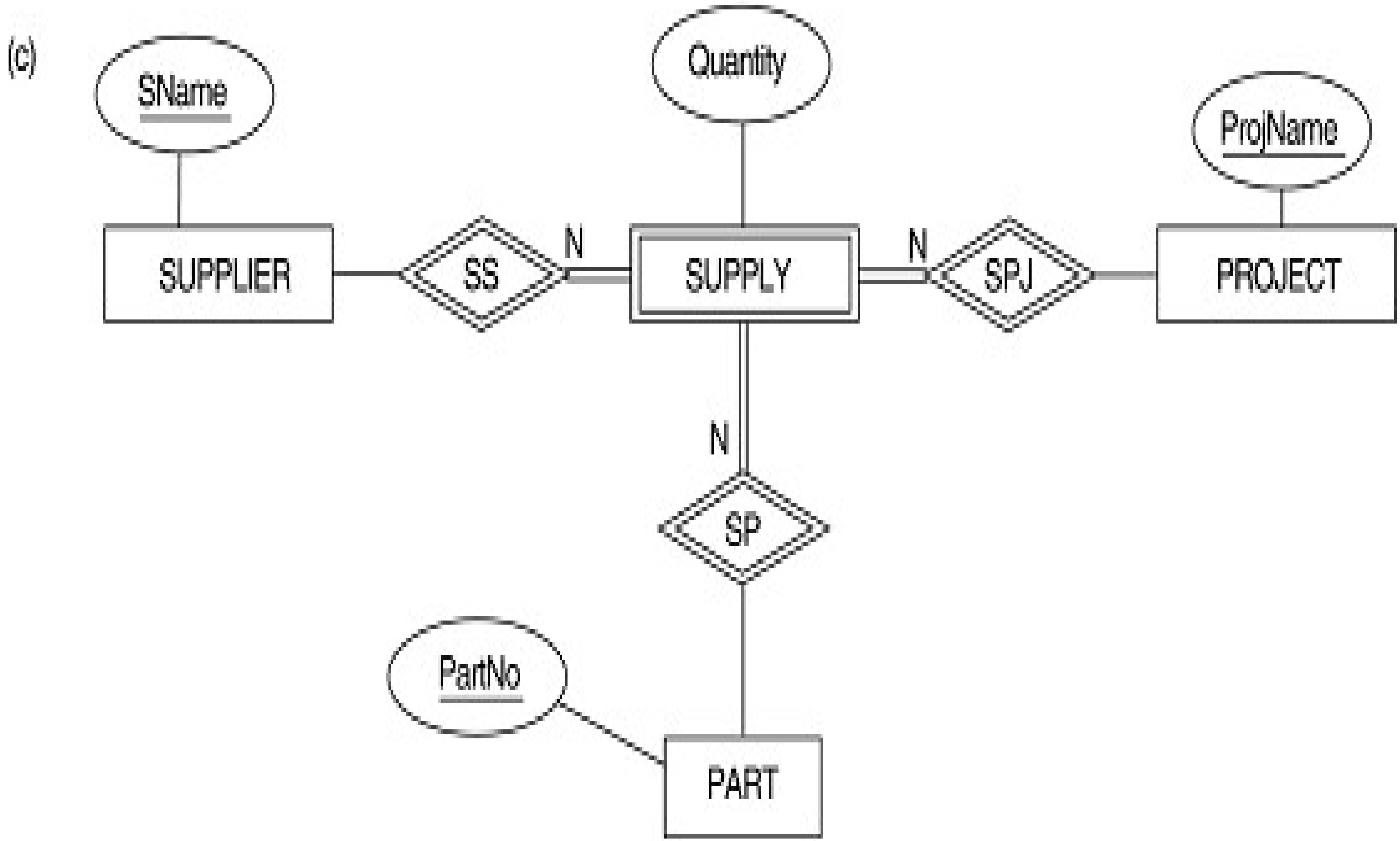
- Ternary Relationship Type
  - relates three entity types
  - SUPPLY (SUPPLIER:PART:PROJECT)
- Three Binary Relationships
  - meaning is different!
  - CAN_SUPPLY (SUPPLIER:PART)
  - SUPPLIES (SUPPLIER:PROJECT)
  - USES (PROJECT:PART)

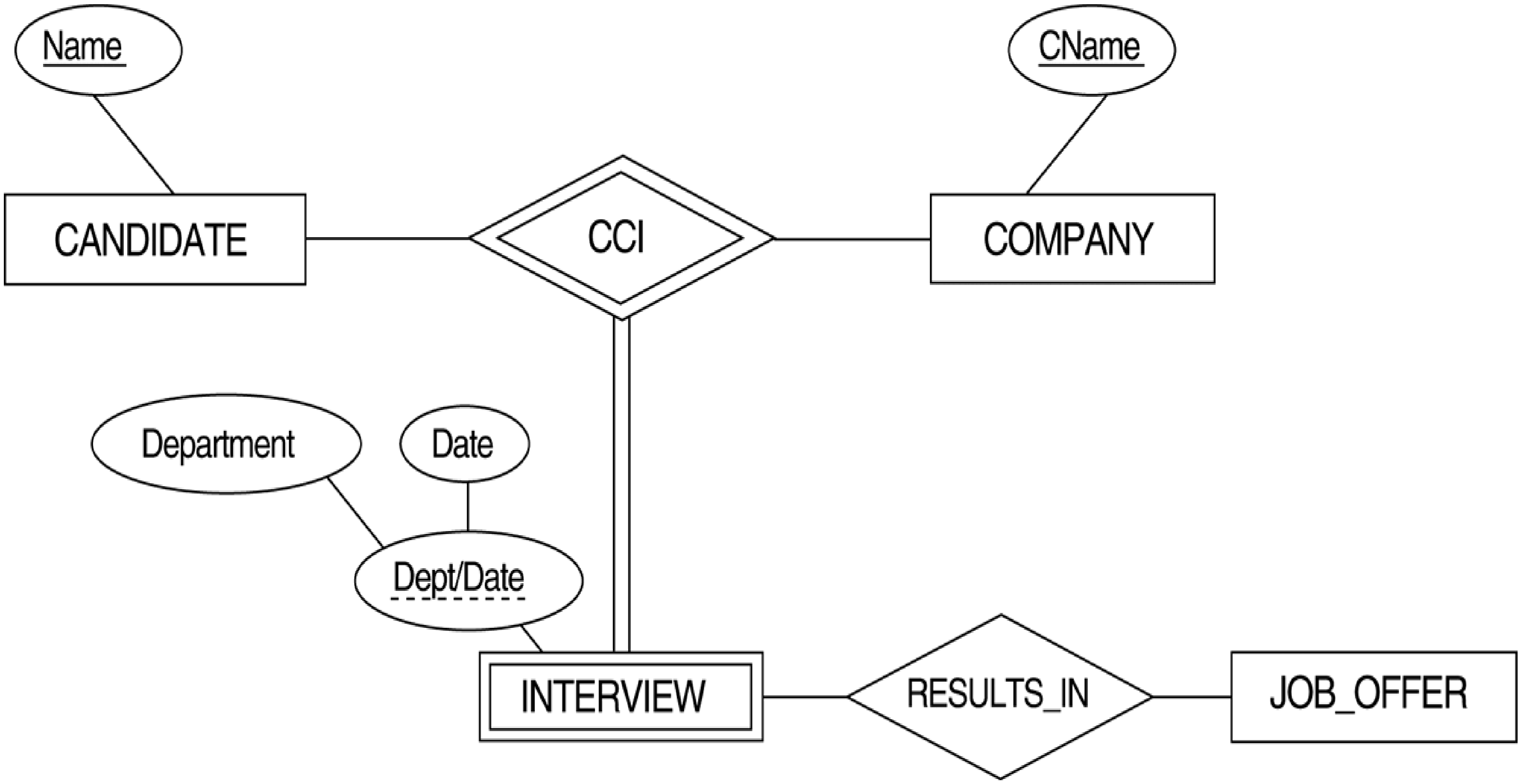
(a)

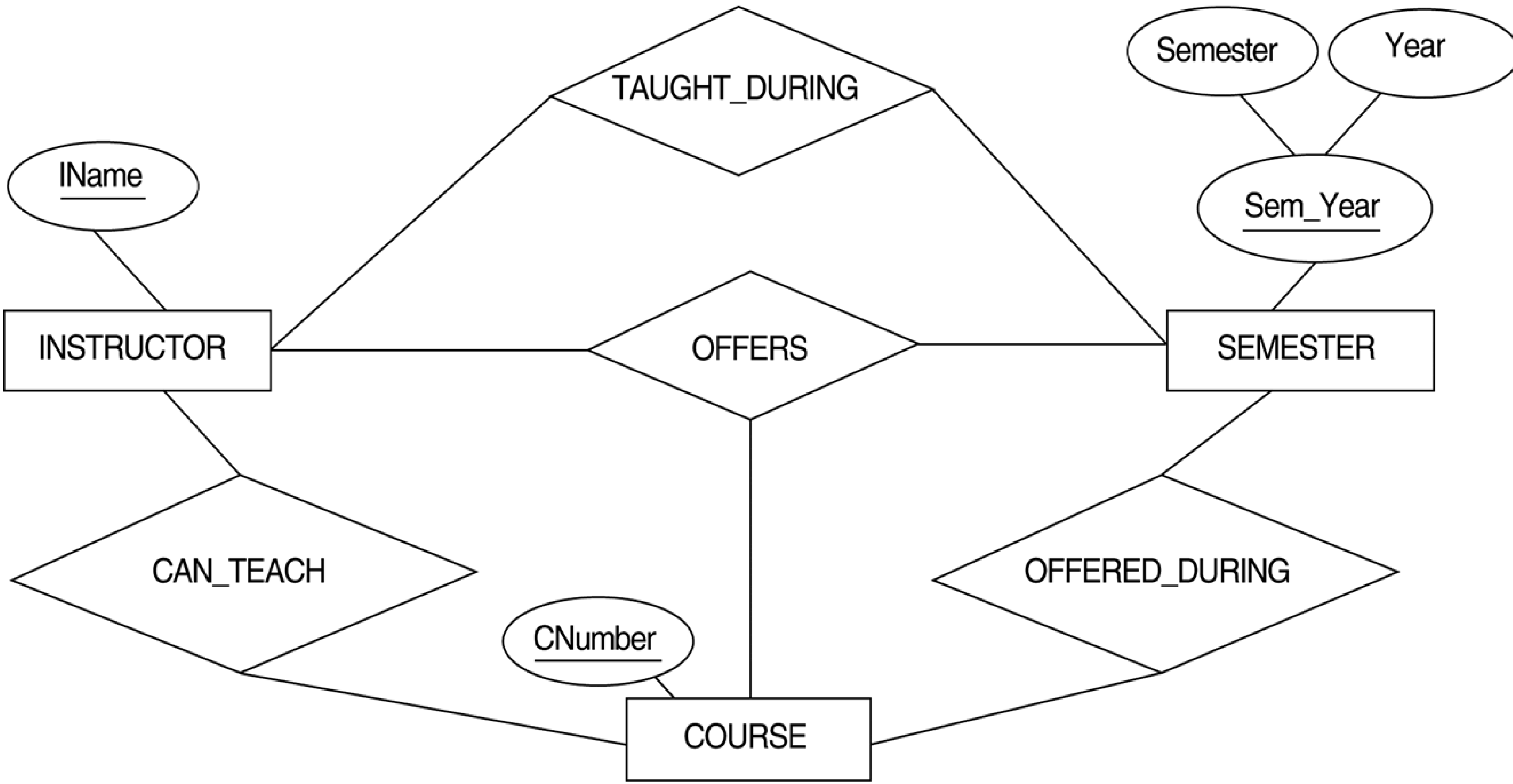

SName
Quantity
ProjName
SUPPLIER
SUPPLY
PROJECT
PartNo
PART


(b)
SName
SUPPLIER
M
SUPPLIES
N
ProjName
PROJECT
M
CAN_SUPPLY
N
PartNo
PART
USES

- Ternary Relationship as Weak Entity Type
  - represents a ternary relationship type as a weak entity type relating to the owner entity types
  - includes binary (identifying) relationship types
- As an Identifying Relationship Type
  - a ternary relationship type with a weak entity type and two owner entity types


(c)
SName
Quantity
ProjName
SUPPLIER
SS
N
SUPPLY
N
SPJ
PROJECT
N
SP
PartNo
PART


Name
CName
CANDIDATE
CCI
COMPANY
Department
Date
Dept/Date
INTERVIEW
RESULTS_IN
JOB_OFFER


TAUGHT_DURING
Semester
Year
IName
Sem_Year
INSTRUCTOR
OFFERS
SEMESTER
CAN_TEACH
OFFERED_DURING
CNumber
COURSE

When use EER

- Most database projects do not need the object-oriented model features in EER
- Goal of conceptual data modeling is to produce a model that simple and easy to understand
- Do not use complicated class/subclass relationship if they are not needed
- Offer significant advantage over regular ER model

- EER model is especially useful is domain being model is OO in nature and use inheritance reduce the complexity of the design
- Cases using EER:
  - When using attribute inheritance can reduce the use of nulls in a single entity relation (that contains multiple subclasses)
  - Subclasses can be used to explicitly model and name subsets of entity types that participate in their own relationships

Formal Definitions of EER Model (1)

- Class C: A set of entities; could be entity type, subclass, superclass, category.
- Subclass S: A class whose entities must always be subset of the entities in another class, called the superclass C of the superclass/subclass (or IS-A) relationship S/C:

  $S \subseteq C$
- Specialization Z: Z = {S1, S2,…, Sn} a set of subclasses with same superclass G; hence, G/Si a superclass relationship for i = 1, …., n.
  - G is called a generalization of the subclasses {S1, S2,…, Sn}
  - Z is total if we always have:

    $S1 \cup S2 \cup \ldots \cup Sn = G$;

    Otherwise, Z is partial.
  - Z is disjoint if we always have:

    $Si \cap S2$ empty-set for $i \neq j$;

    Otherwise, Z is overlapping.

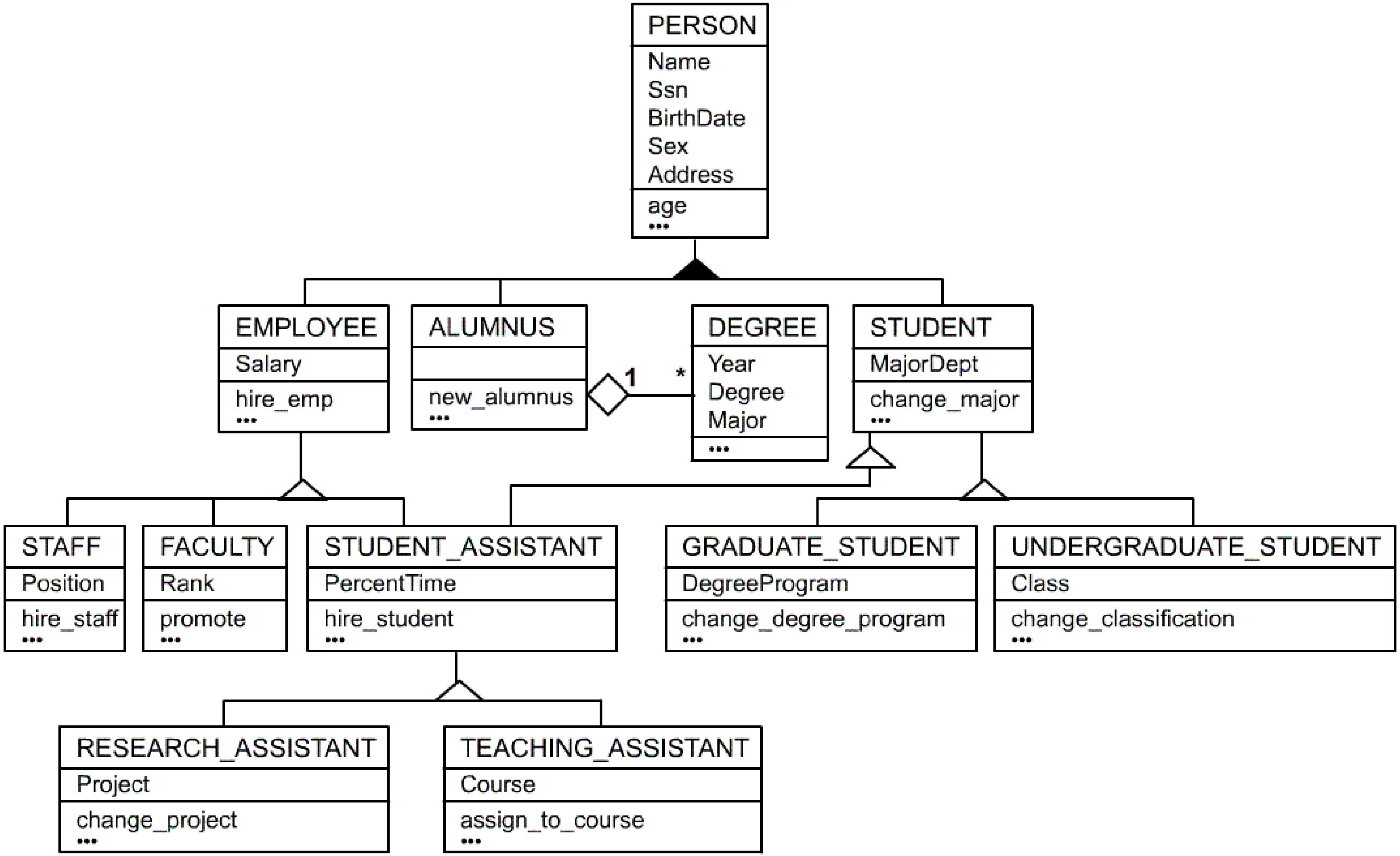
Formal Definitions of EER Model (2)

- Subclass S of C is predicate defined if predicate p on attributes of C is used to specify membership in S; that is, S = C[p], where C[p] is the set of entities in C that satisfy p
- A subclass not defined by a predicate is called user-defined
- Attribute-defined specialization: if a predicate A = ci (where A is an attribute of G and ci is a constant value from the domain of A) is used to specify membership in each subclass Si in Z
- Note: If ci ≠ cj for i ≠ j, and A is single-valued, then the attribute-defined specialization will be disjoint.
- Category or UNION type T
  - A class that is a subset of the union of n defining superclasses D1, D2,…Dn, n>1:

    $T \subseteq (D1 \cup D2 \cup \ldots \cup Dn)$

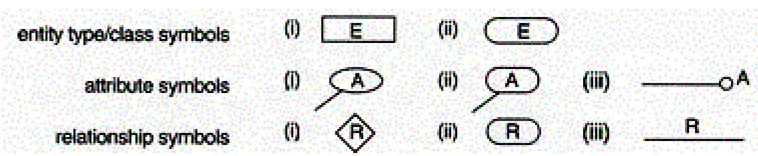
    A predicate pi on the attributes of T.
  - If a predicate pi on the attributes of Di can specify entities of Di that are members of T.
  - If a predicate is specified on every Di: $T = (D1[p1] \cup D2[p2] \cup \ldots \cup Dn[pn]$
  - Note: The definition of relationship type should have 'entity type' replaced with 'class'.

UML Example for Displaying Specialization / Generalization

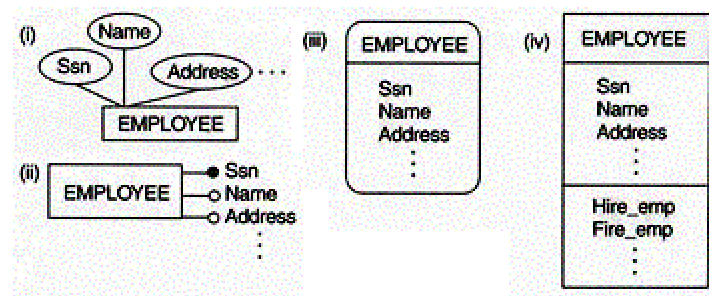


Alternative Diagrammatic Notations

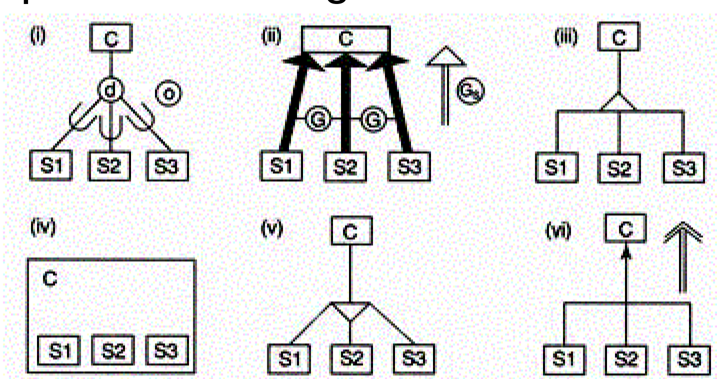
Symbols for entity type / class, attribute and relationship



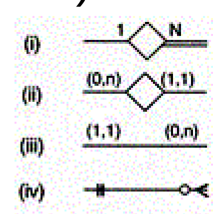
Displaying attributes



Notations for displaying specialization / generalization



Various (min, max) notations



Displaying cardinality ratios