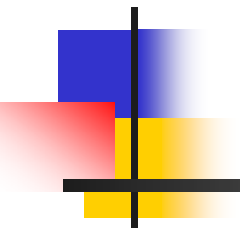
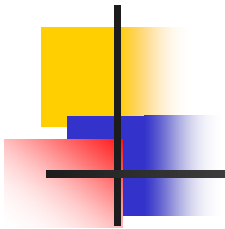
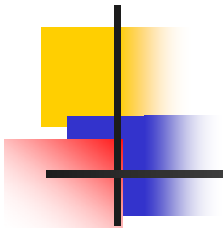


ENHANCED ENTITY-RELATIONSHIP (EER) MODEL



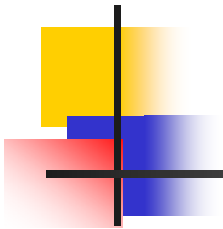
- Objectives

- Introduction to EER Modeling +
- Concepts of the EER Model +
- Specialization +
- Generalization +
- Constraints on Specialization/Generalization (S/G) +
- S/G Notation Summary +
- Insertion and Deletion Rules +
- Hierarchies and Lattices +
- S/G Conceptual Data Modeling +
- Categorization +
- Conceptual Object Modeling +



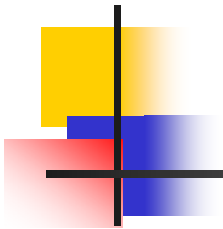
- Introduction to EER Model

- Since the late 1970 there has been an increase in the emergence of new database applications with more demanding requirements.
- Basic concepts of the ER model are not sufficient to represent the requirements of the newer, more complex applications.
- To represent these requirements additional ‘semantic’ modeling concepts are needed.
- These semantic concepts are incorporated into the original ER model and became Enhanced Entity-Relationship (EER) model.
- Additional concepts of EER model include:
  - Specialization
  - Generalization
  - Categorization
- EER diagrams are similar to the class diagrams used in the OO model.



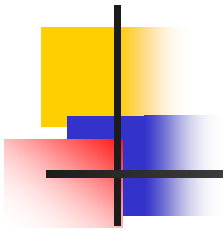
- Concepts of EER Model ...

- The EER model includes all the concepts used in the ER model:
  - Entity types
  - Attributes
  - Relationship types
- In addition, the EER model includes the following concepts, which come from the OO model:
  - Supreclass/subclass relationships
  - Attribute inheritance
  - Specialization
  - Generalization
  - Categorization



... - Concepts of EER Model

- Subclass: is an entity type that has a distinct role and is also a member of the superclass.
- Superclass: is an entity type that includes distinct subclasses that require to be represented in a data model.
- Attribute inheritance: is an entity in a subclass may posses subclass specific attributes, as well as those associated with the superclass.
- The relationship between a superclass and any of its subclasses is called the superclass/subclass relationship.



- Specialiazation

- A Specialization is the process of defining a set of subclasses of an entity type.
- The specialization of an entity type allows us to do the following:
  - Define a set of subclasses of an entity type.
  - Establish additional attributes with subclasses
  - Establish additional relationship types between some subclasses and other entity types or other subclasses.
- The figure in the next slide shows an example of a superclass/subclass relationship between the entity type EMPLOYEE and some of its classes.
  - In this figure, local attributes are attached to subclasses
  - Common attributes are attached to the superclass
  - Some relationships between subclasses are shown.

-- Example: Specialization

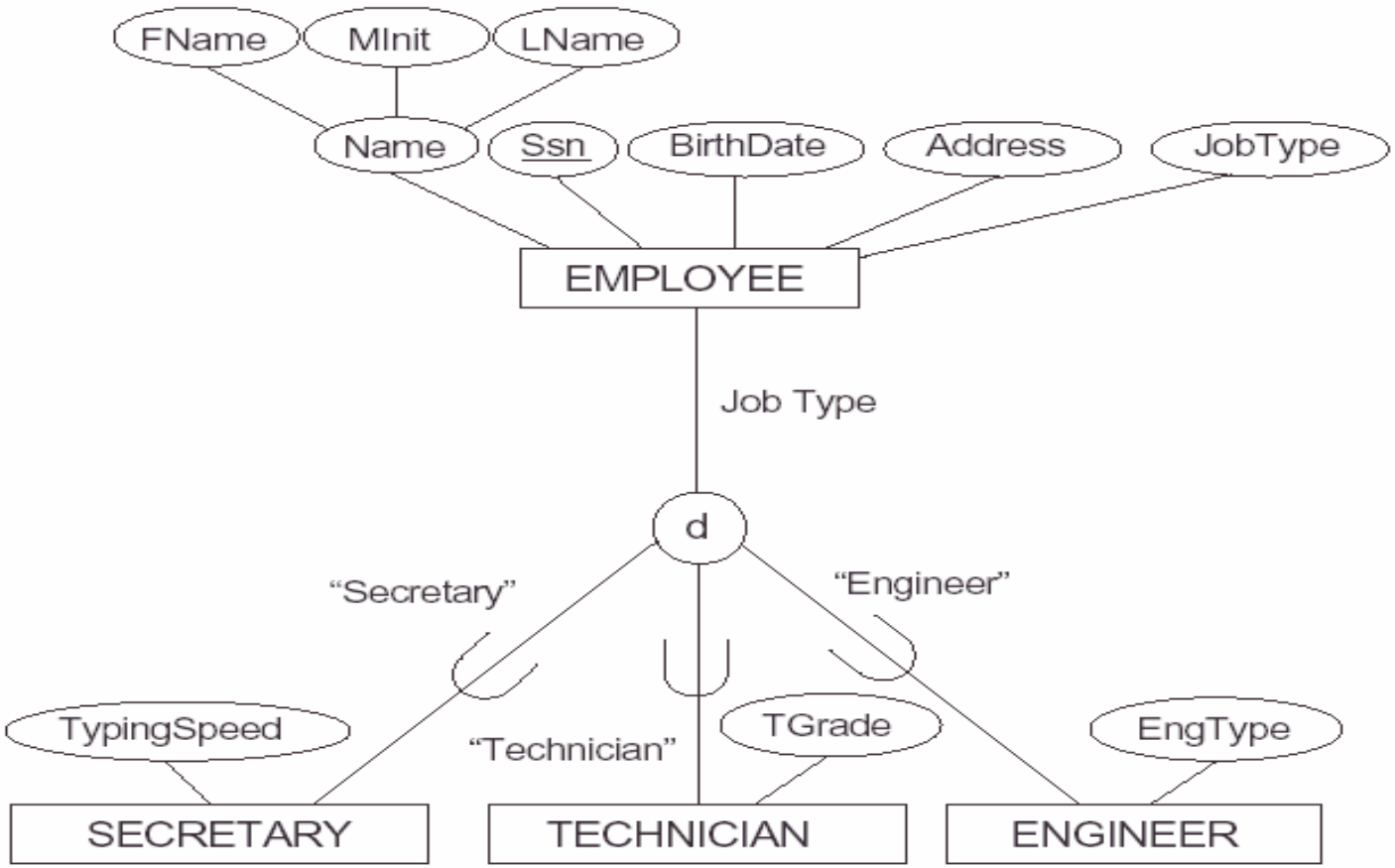
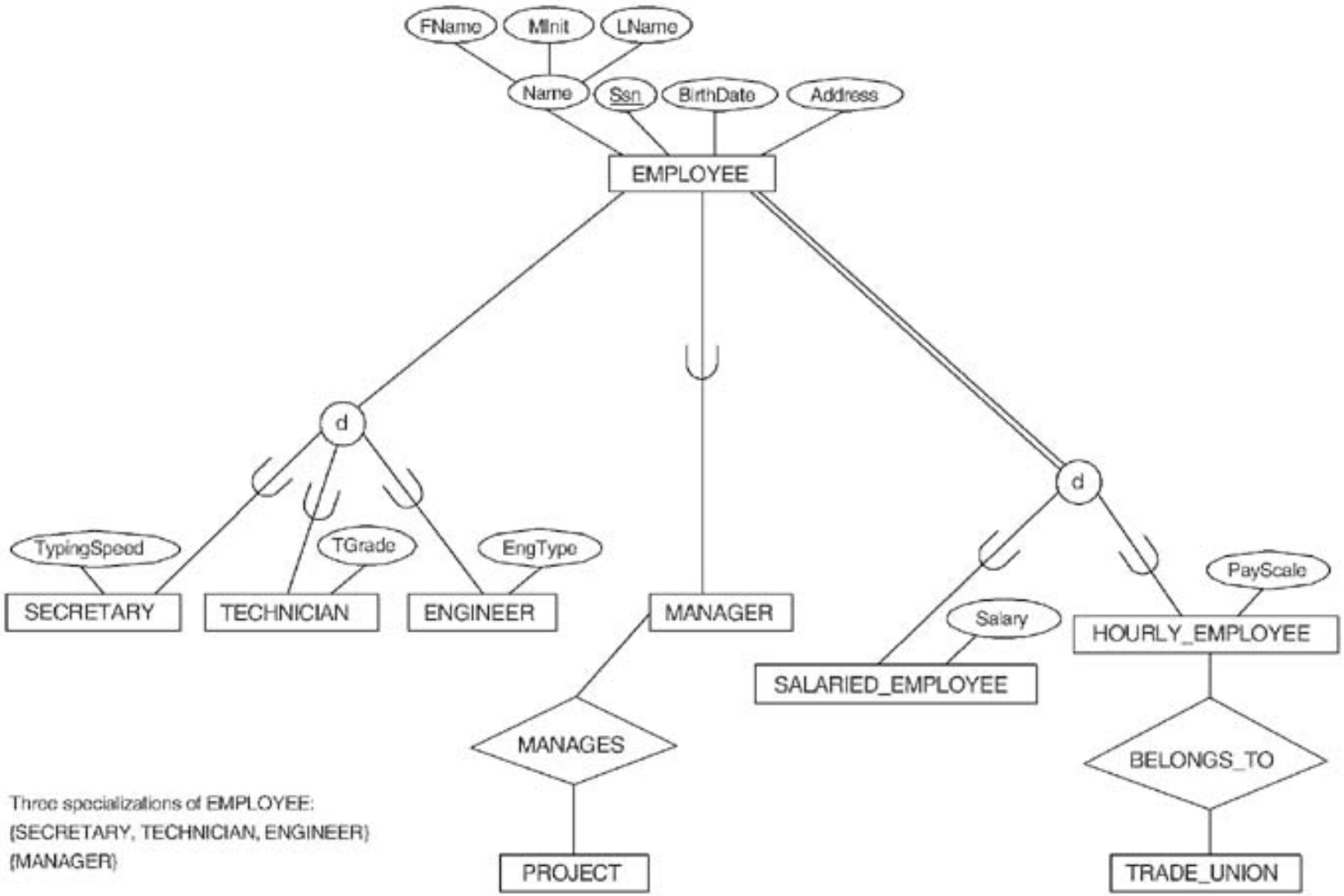
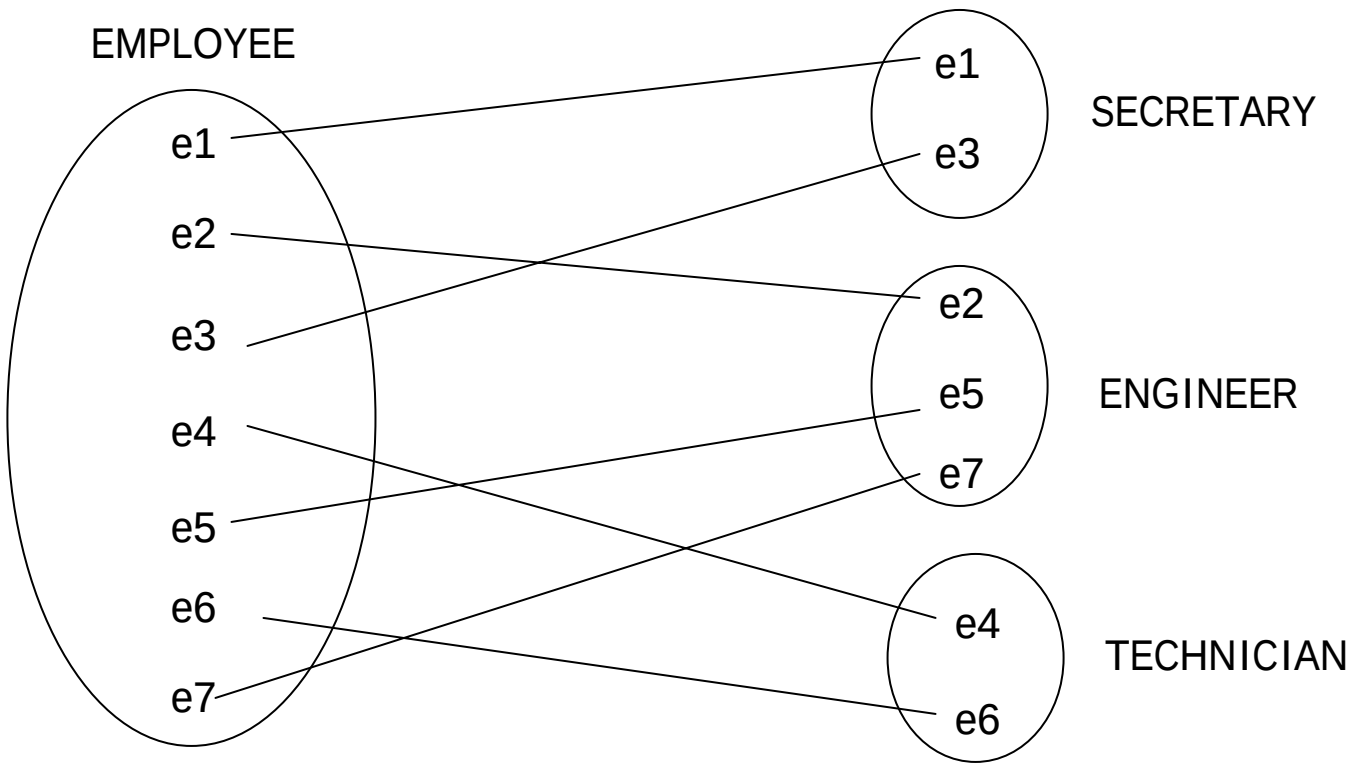


Figure 4.1 EER diagram notation for representing specialization and subclasses.



... - Specialization

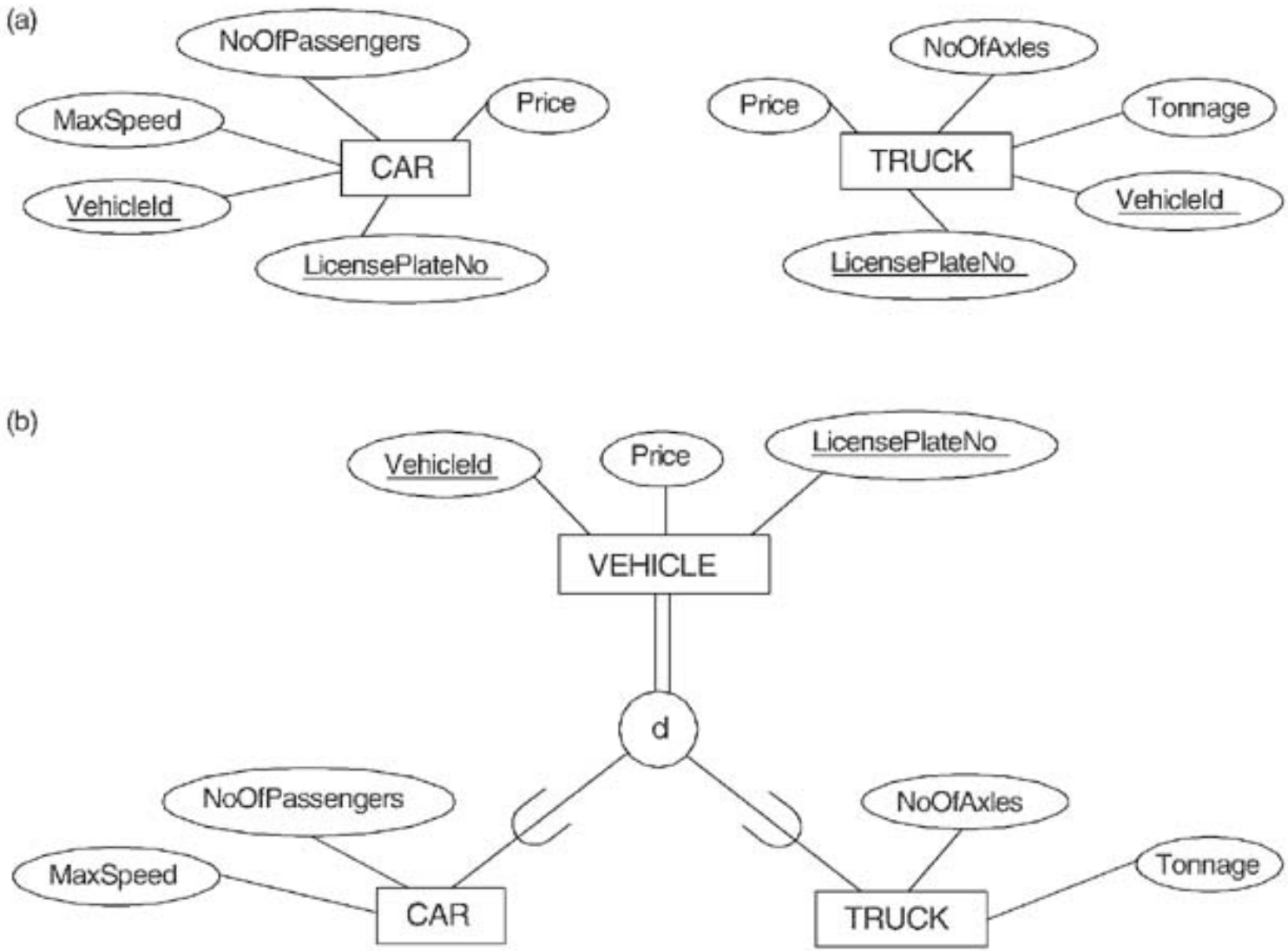
- The following Figure shows an example of some instances of the specialization of the superclass EMPLOYEE into the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN}

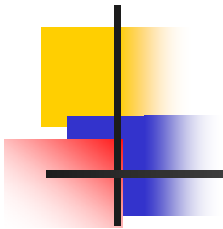


- Generalization

- Generalization is the process of defining a superclass of many entity types.
- Generalization can be considered as the reverse of specialization.
  - Example: We can view EMPLOYEE as a generalization of SECRETARY, TECHNICIAN, and ENGINEER.
- The figure in the next slide shows an example of generalizing the two entity types (subclasses) CAR and TRUCK into the VEHICLE entity type (superclass).

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





- Constraints on Specialization/Generalization

- Attribute defined Specialization +
- User defined specialization +
- Disjointness constraint +
- Completeness constraints +

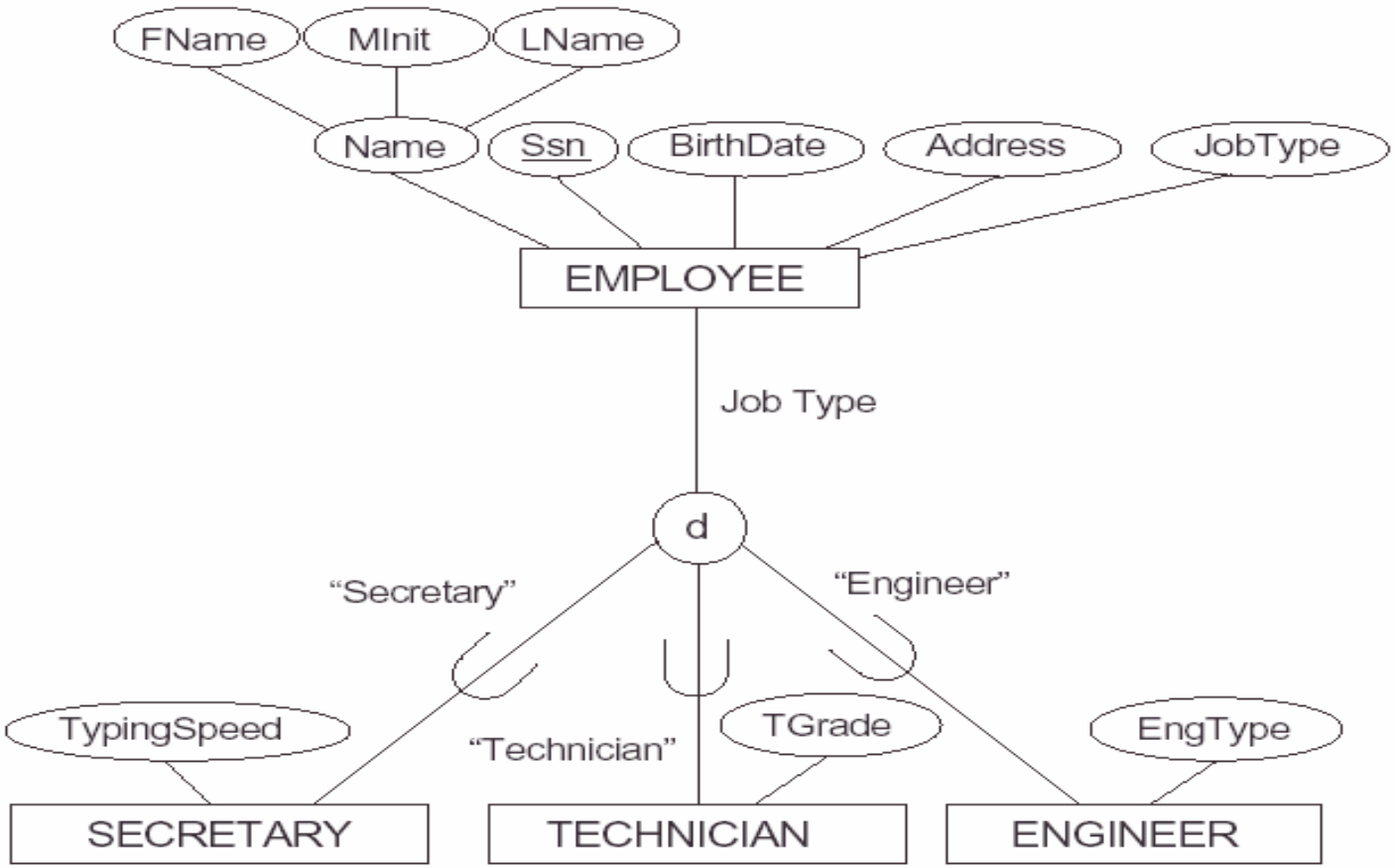
-- Attribute Defined Specialization

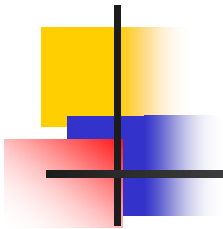
- In some specialization we can determine exactly the entities that will become members of each subclass. This is done by placing a condition on the values of some attributes of the superclass. Such subclasses are called predicate-defined or condition-defined subclasses.

- If all subclasses in a specialization have the same membership condition on the same attribute, the specialization is called attribute-defined specialization. The attribute is called the defining attribute of the specialization.

- In the figure of the next slide we distribute members between subclasses based on the value of the Job Type attribute. Job Type is called the defining-predicate of the subclass.

--- Subclasses based on Job Type Attribute





-- User defined specialization

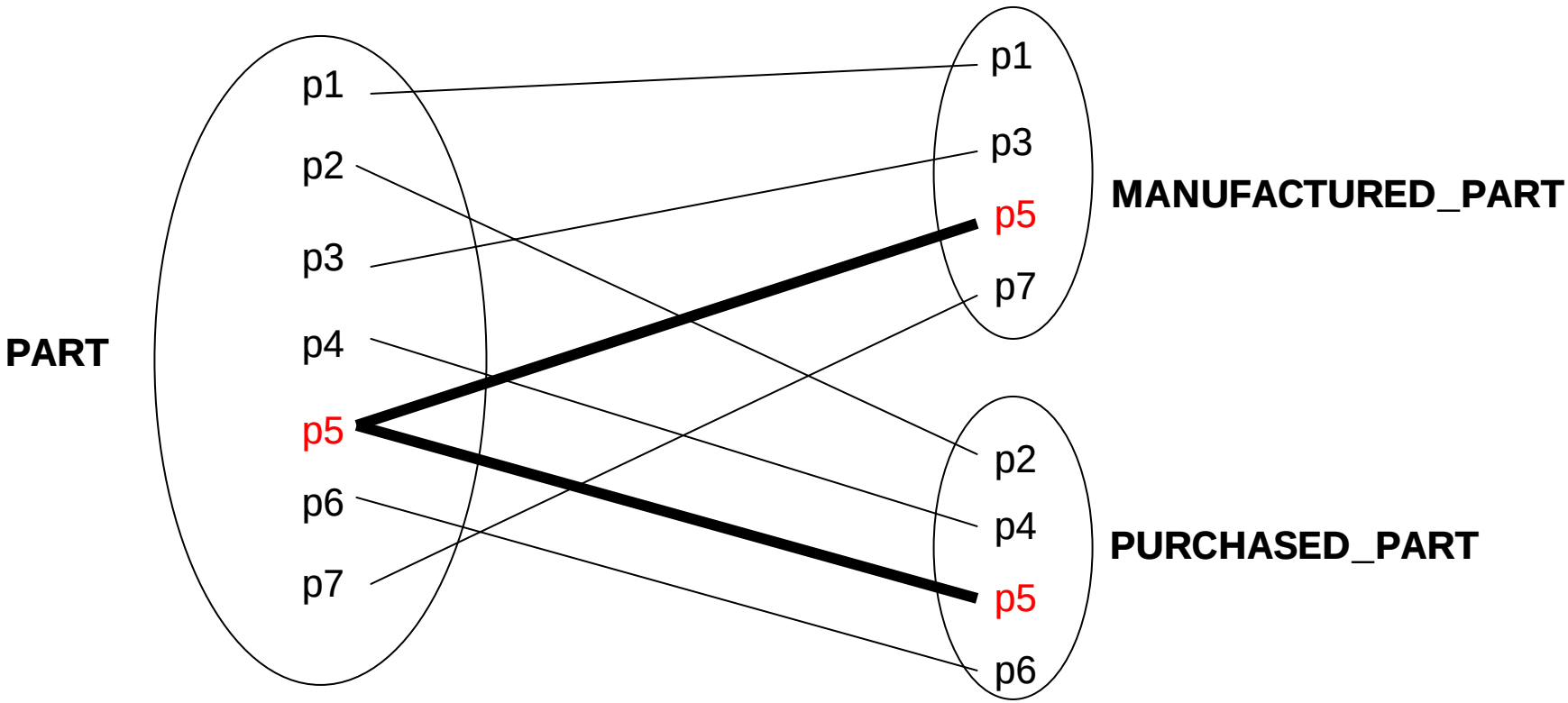
- When we do not have a condition for determining membership in a subclass then we call this specialization a user defined specialization.

- Membership is specified individually for each entity by the user, not by any condition that may be evaluated automatically.

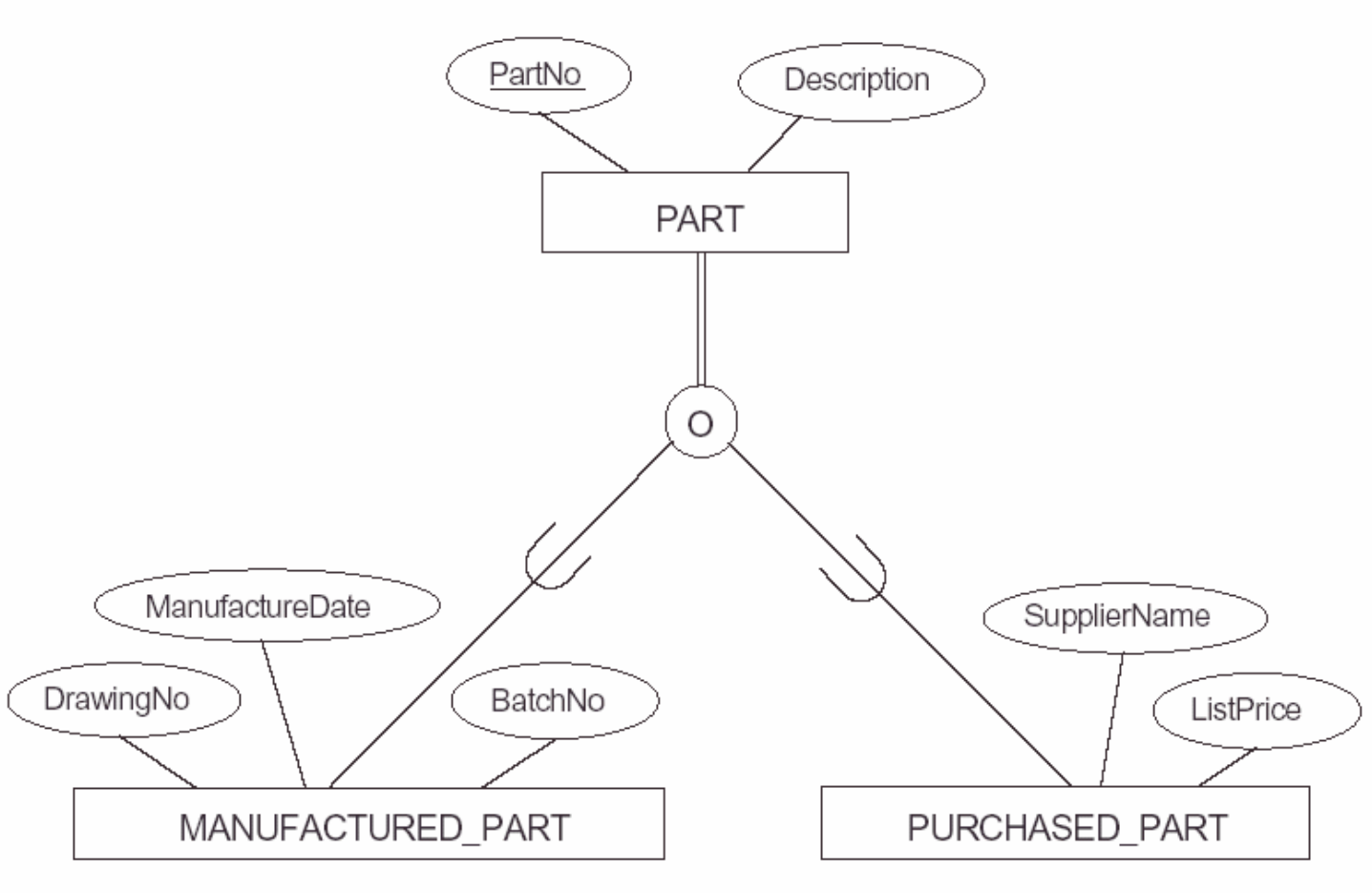
-- Disjointness constraint

- Look to the superclass/subclass relationship shown in this figure. We can see that the subclasses are disjoint, i.e. every employee entity will belong to only one of the subclasses, SECRETARY or TECHNICIAN or ENGINEER.
- This is represented in this diagram by the symbol **d** which stands for disjoint.
- However, there are some superclasses that can be specialized into subclasses that may overlap as the example shown the following figure. Here a PART entity can belong to one or more of the entities MANUFACTURED or PURCHASED.
- This is represented in the diagram by the symbol **O** which stands for overlapping.

... - Overlapping Subclasses ...



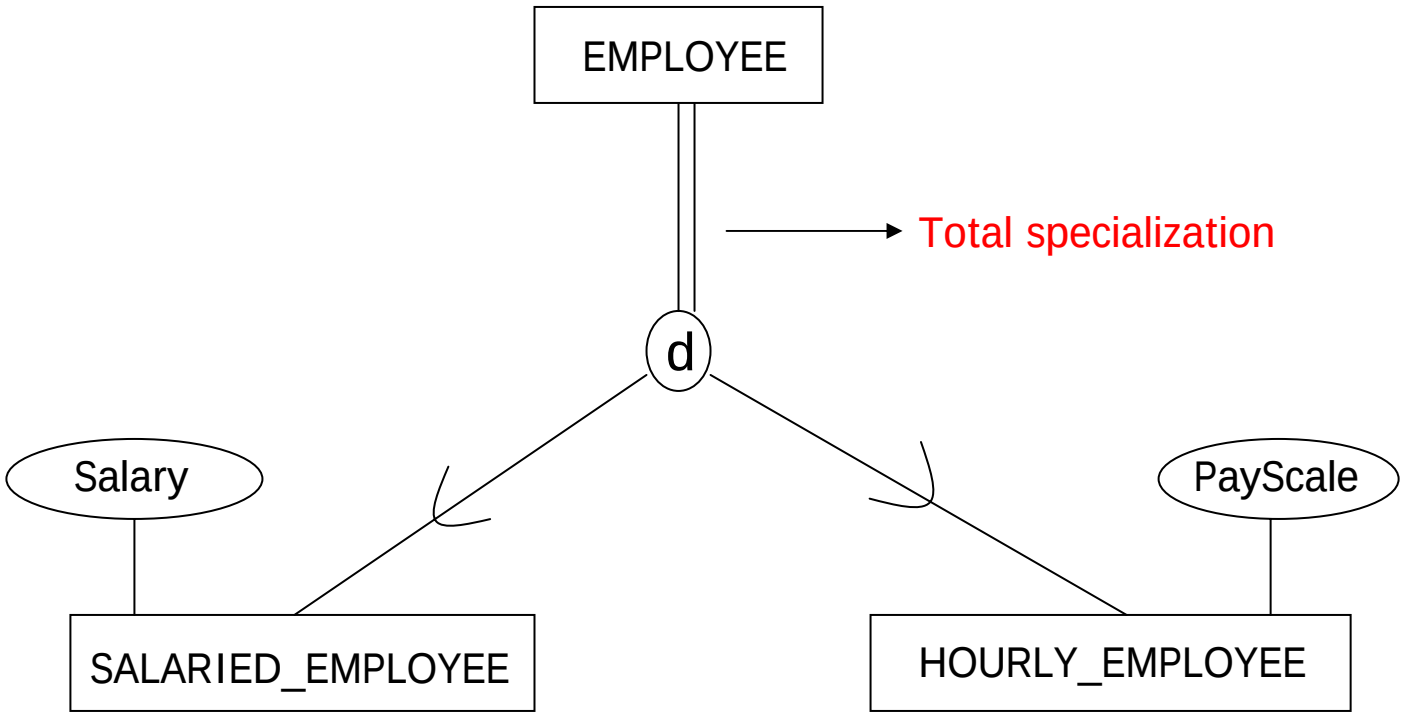
--- Overlapping subclasses



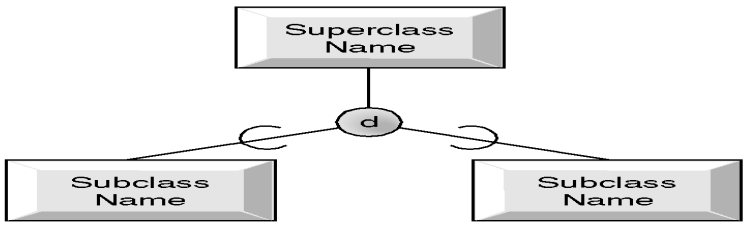
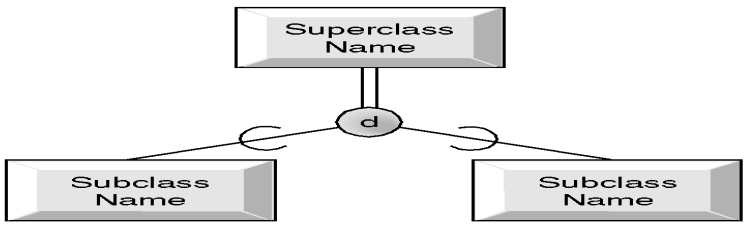
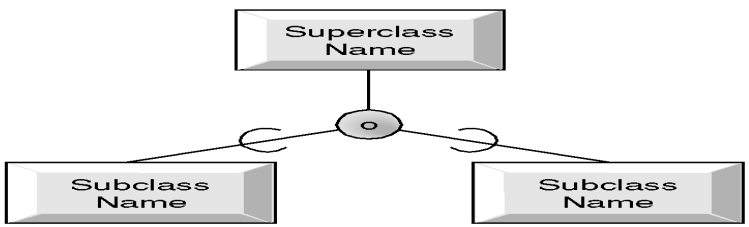
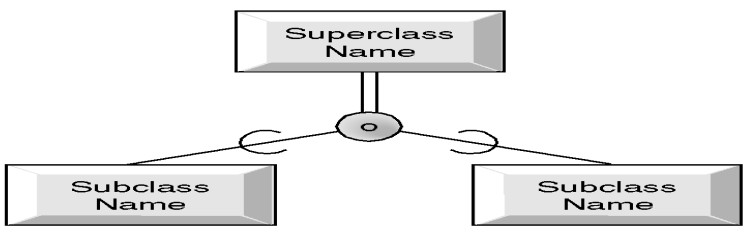
-- Completeness constraints

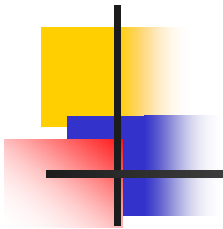
- A total specialization constraint specifies that every entity in the superclass must be a member of some subclass.
- For example, if every employee must be either SALARIED_EMPLOYEE or HOURLY_EMPLOYEE then the specialization {SALARIED_EMPLOYE, HOURLY_EMPLOYEE} is total specalization.
- This constraint is shown in the figure which is in the next slide by using double lines to connect the superclass as shown in the figure.

--- Example of Total Specialization



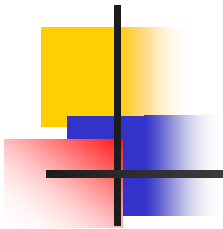
- S/G Notation Summary

Notation	Meaning
	Superclass/Subclass Relationship Disjoint, Partial
	Superclass/Subclass Relationship Disjoint, Total
	Superclass/Subclass Relationship Overlapping, Partial
	Superclass/Subclass Relationship Overlapping, Total



- Insertion and Deletion Rules

- Deleting an entity from a superclass implies that it is automatically deleted from all subclasses to which it belongs.
- Inserting an entity into a superclass implies that the entity is mandatorily inserted in all predicate-defined (or attribute-defined) subclasses for which the entity satisfies the defining predicate.
- Inserting an entity into a superclass of a total specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization

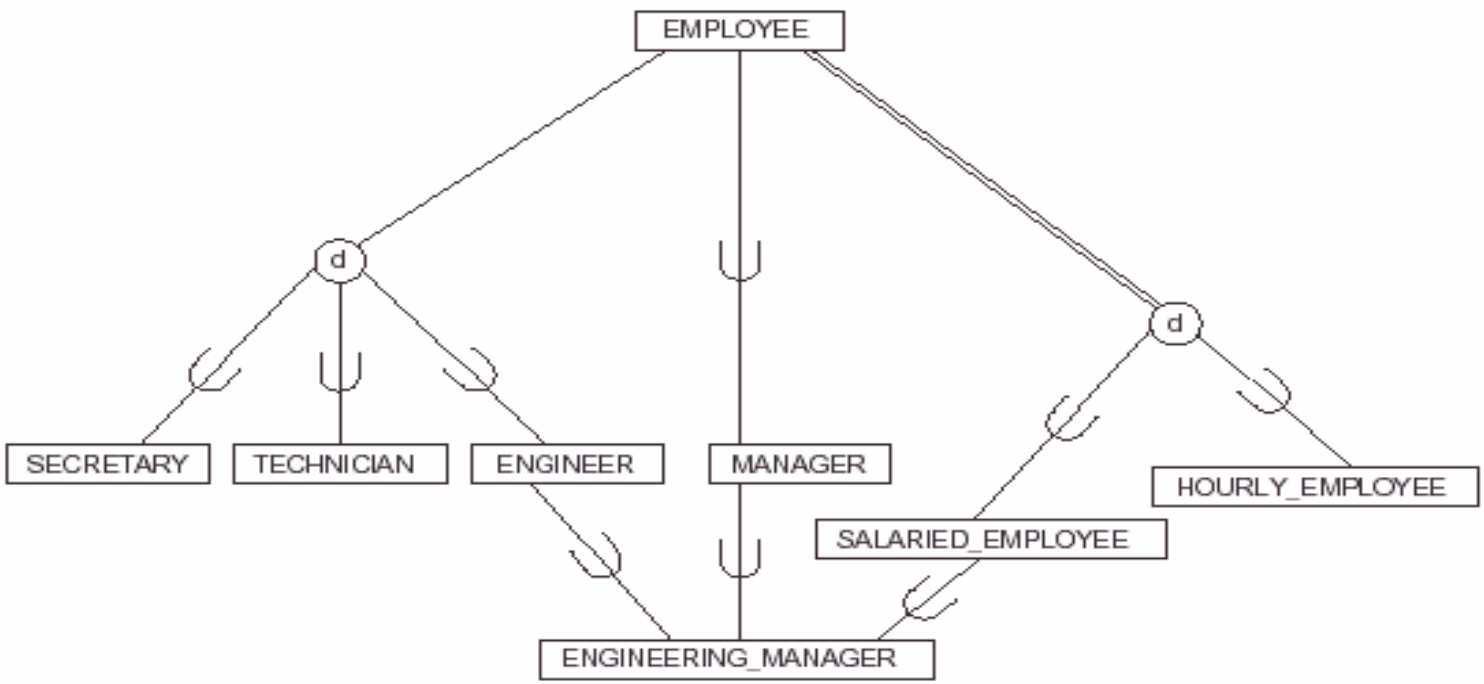


- Hierarchies and Lattices

- A subclass itself may have further subclasses on it, forming a hierarchy or a lattice of specialization.
- In a hierarchy every subclass participates as a subclass in only one superclass/subclass relationship.
- In a lattice a subclass may participate as subclass in more than one superclass/subclass relationship.
- A subclass with more than one superclass is called a shared subclass.
- A shared subclass inherits attributes and relationship from multiple superclasses, resulting in having multiple inheritance.

-- Example of a Lattice

Figure 4.6 A specialization lattice with the shared subclass ENGINEERING_MANAGER.



- S/G Conceptual Data Modeling

- In the specialization process, we typically start with an entity type and then define its subclasses by successive specialization. This process is top-down conceptual refinement process during the conceptual schema design.

- The example show in the this figure of a university database shows this process. We start by the PERSON entity then we define its (overlapping) subclasses EMPLOYEE, ALUMNUS, STUDENT. We continue this process until we get all the subclasses shown in the figure.

- Another method of reaching the same lattice is through a bottom-up conceptual synthesis process. We start by identifying the entity types that actually represent the subclasses. Then the generalization is used to collect the common attributes and relationships between subclasses to create the superclasses, and so on.

-- Example: Conceptual Modeling

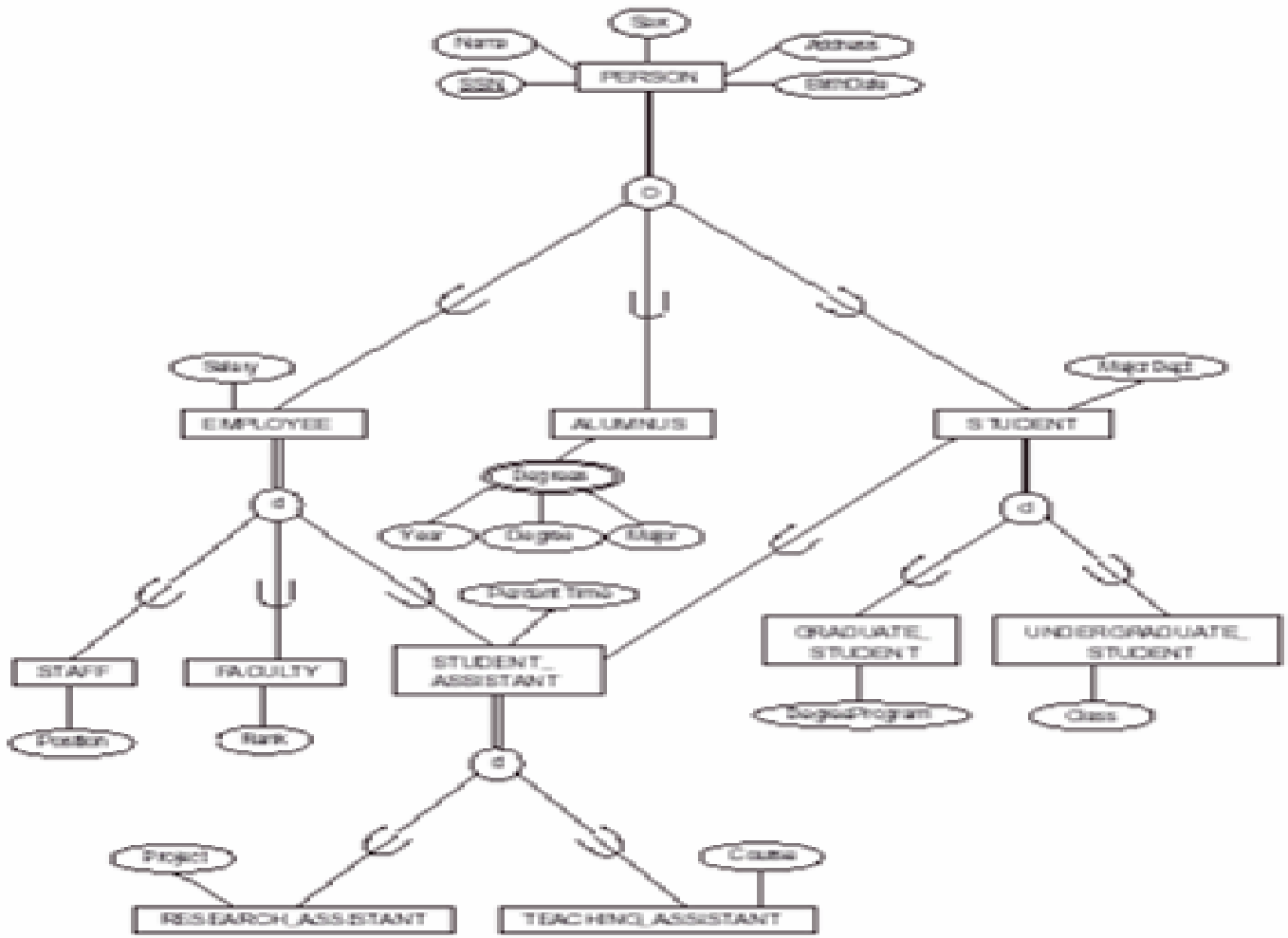
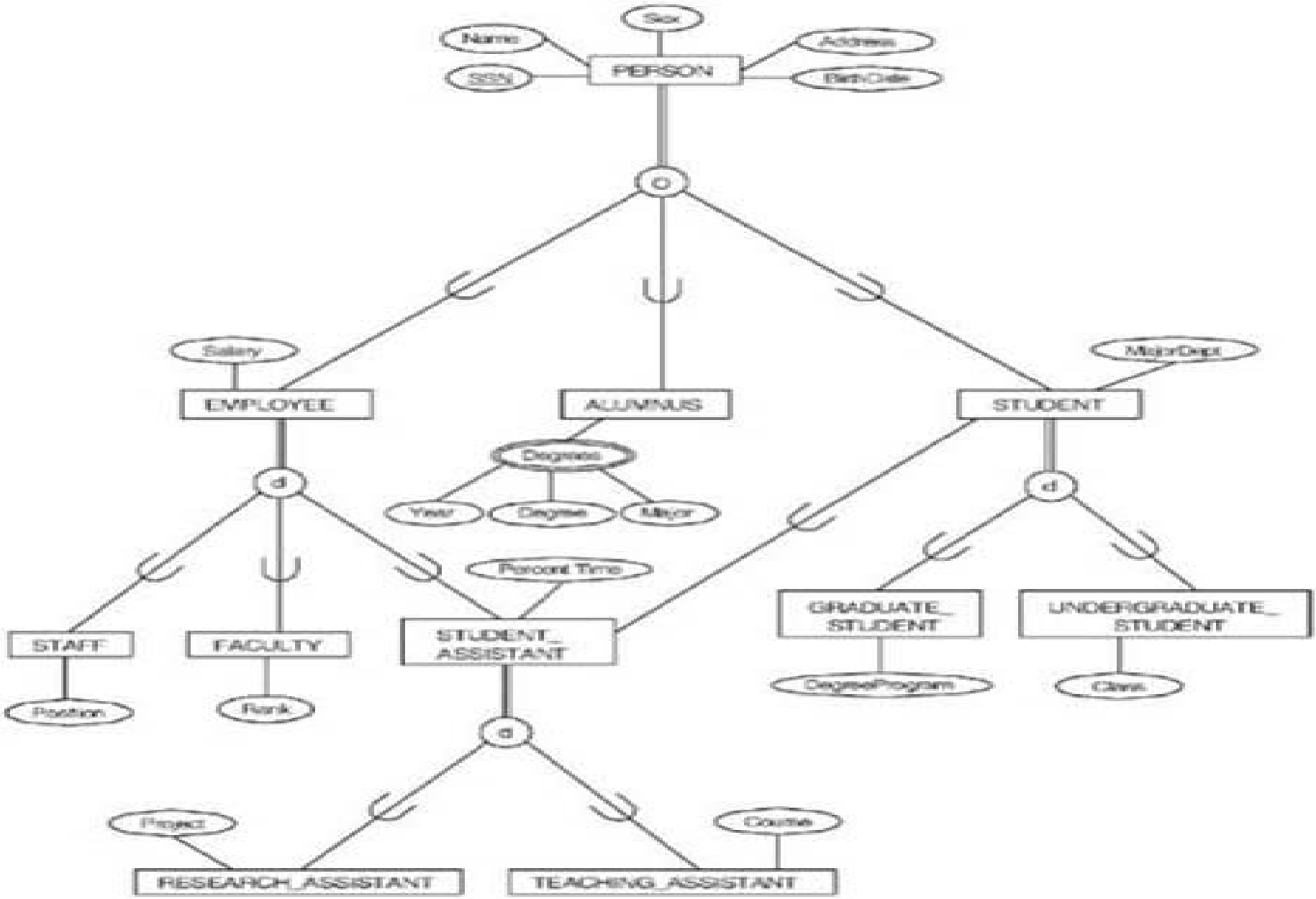
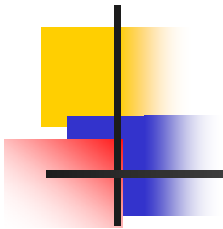


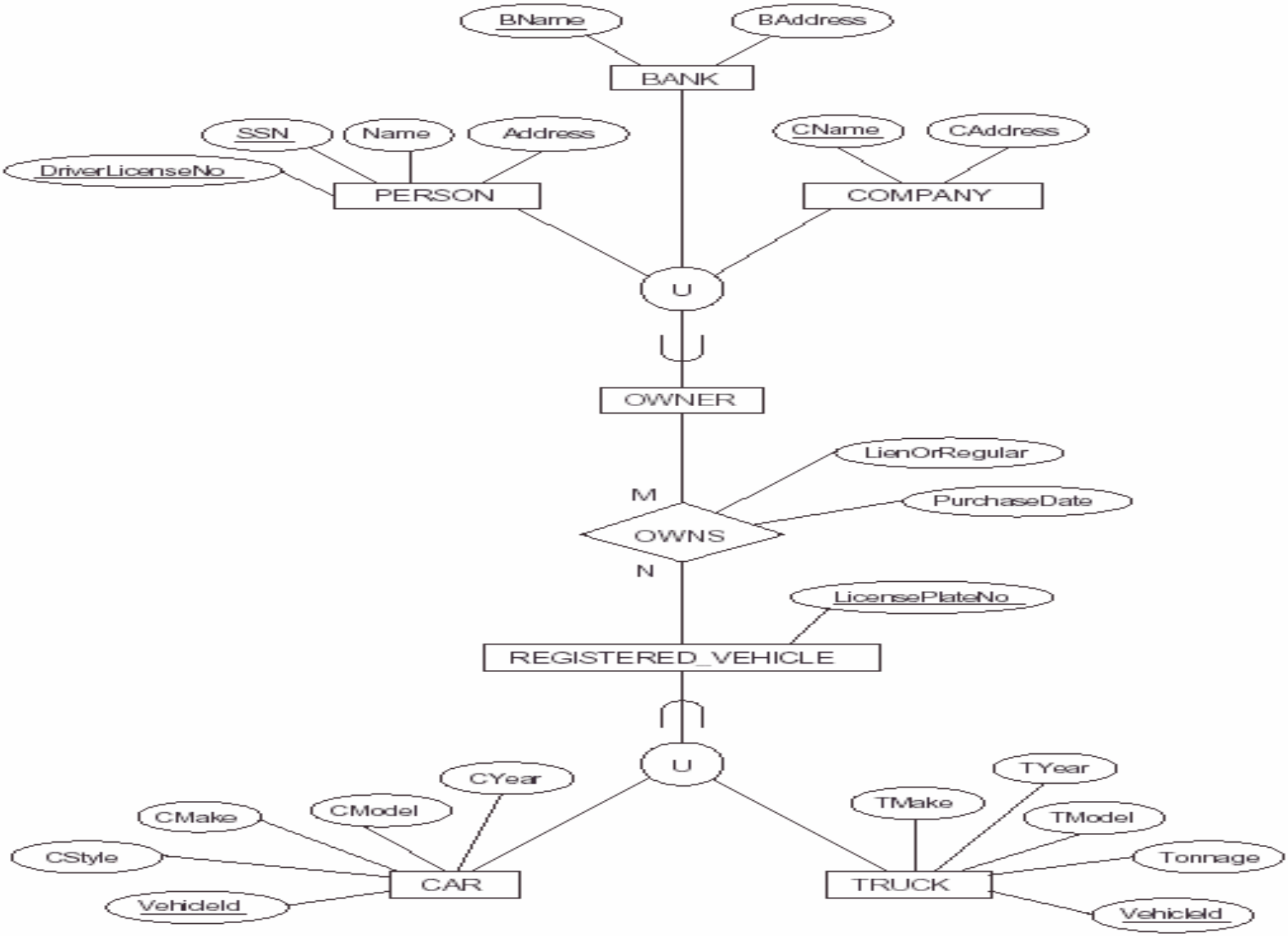
Figure 4.7 A specialization lattice (with multiple inheritance) for a UNIVERSITY database.


Sex
Name
Address
SSN
PERSON
Birth-Date
o
Salary
EMPLOYEE
ALUMNUS
MajorDept
STUDENT
Degrees
d
Year
Degree
Major
d
Percent Time
STAFF
FACULTY
STUDENT_ ASSISTANT
GRADUATE_ STUDENT
UNDERGRADUATE_ STUDENT
Position
Rank
DegreeProgram
Class
o
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT



- Categorization

- Categorization (Union) is the modeling of a single subclass (called a category) with a relationship that involves more than one distinct superclasses. (see this Figure)
- The difference between category and a shared subclass is that a shared subclass entity must be a member of all the superclass entities (intersection) whereas a category is a subset of the union of its superclasses. Hence an entity which is a member of a category must exist in only one of the superclasses.
- In a category, subclass has selective inheritance.
- Categories can be either total or partial.
  - Total: every occurrence of all superclass must appear in the category.
  - Partial: Some occurrences of all superclasses may not appear in the category.


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

-- Example: Categorization

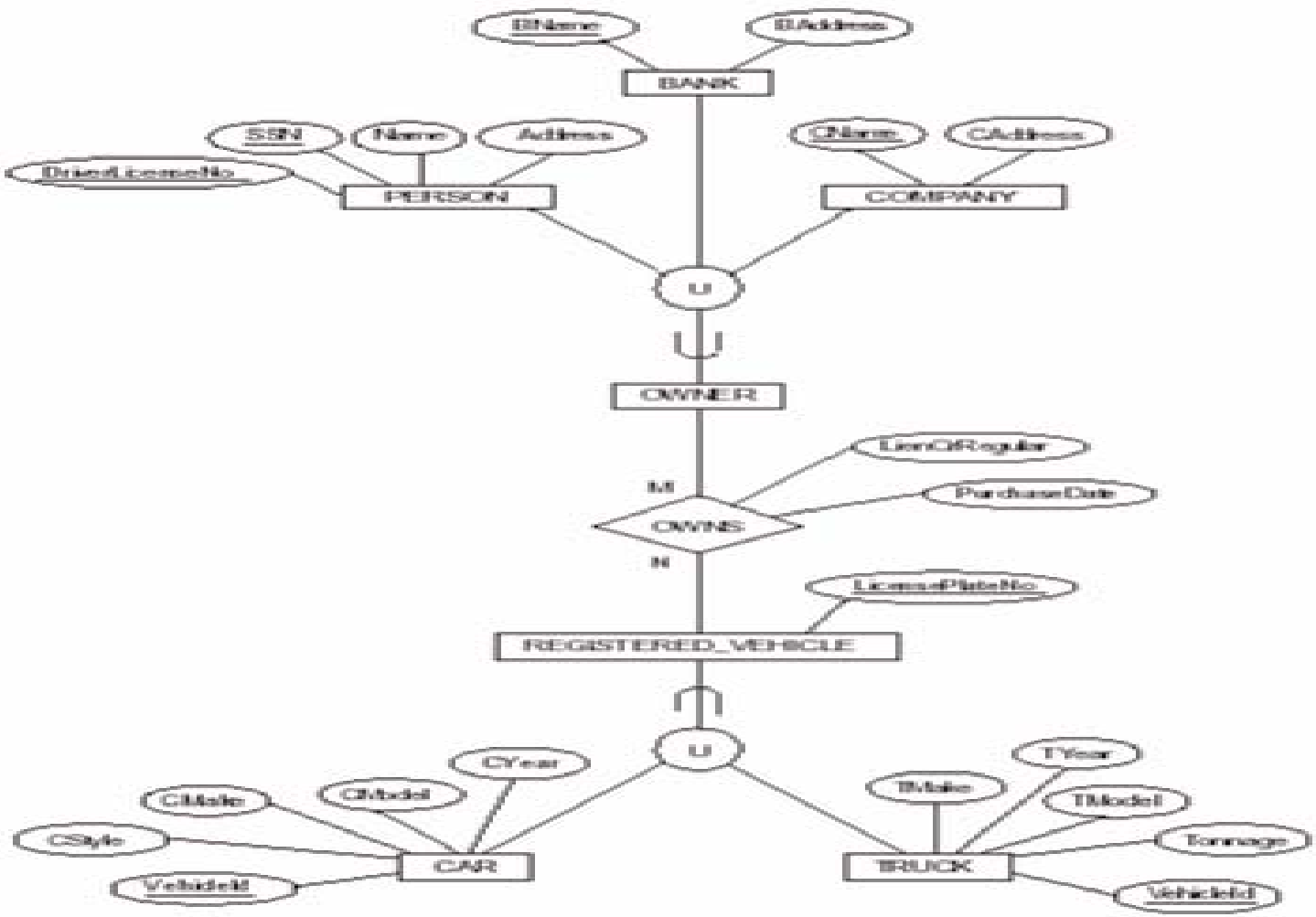
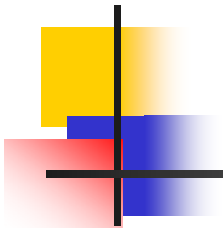
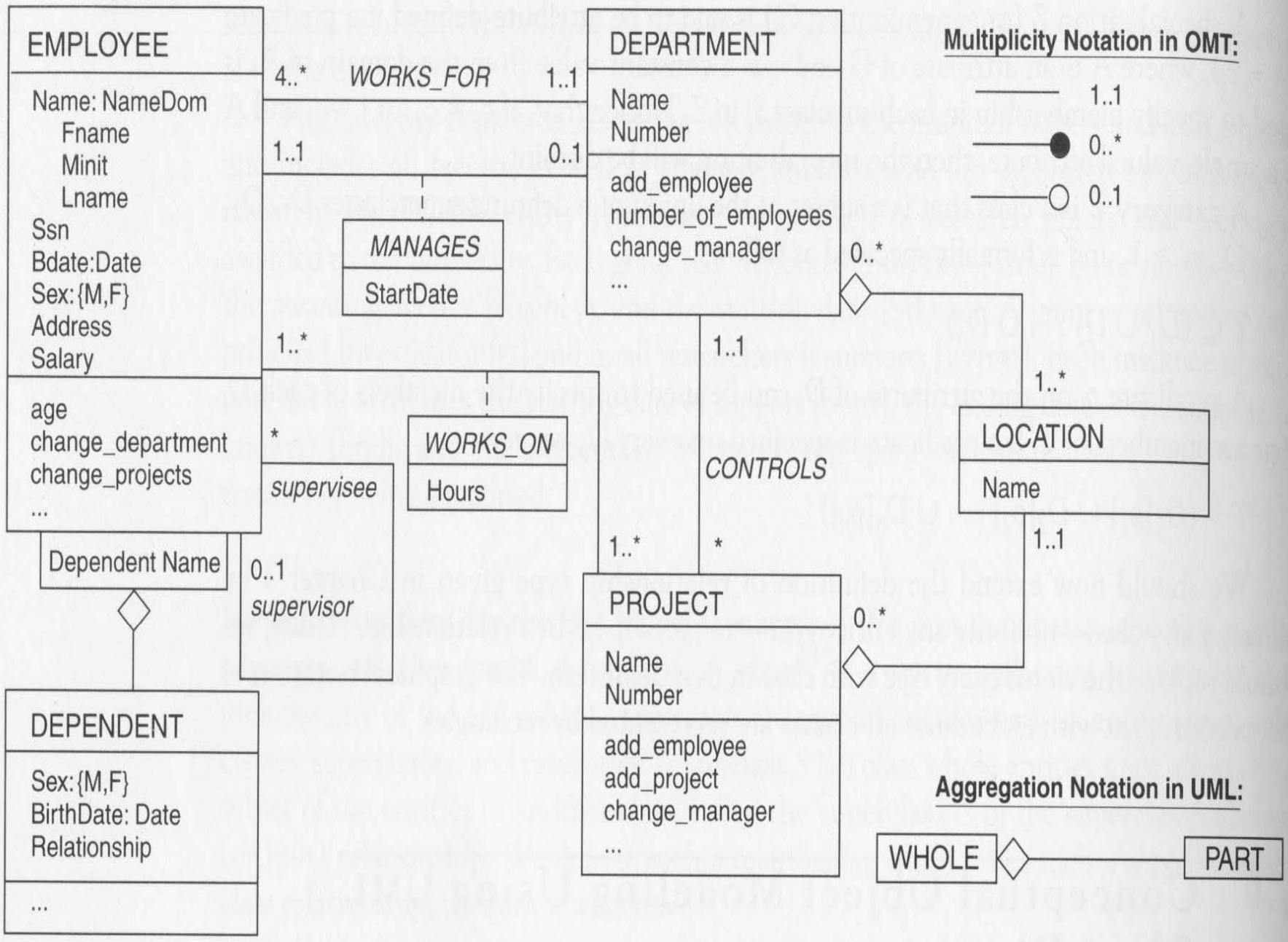


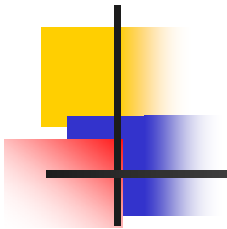
Figure 4.8 An illustration of how to represent the UNION of two or more entity types/classes using the category notation. Two categories are shown: OWNER and REGISTERED_VEHICLE.



- Conceptual Object Modeling

- Object modeling methodology, such as UML (Universal Modeling Language) and OMT (Object Modeling Technique) are becoming increasingly popular.
- Although these methodologies were developed mainly for software design, but they can be used in the database design.
- The class diagrams of these methodologies are similar to the EER diagrams in many ways.


EMPLOYEE
Name: NameDom
Fname
Minit
Lname
Ssn
Bdate:Date
Sex:{M,F}
Address
Salary
age
change_department
change_projects
...
Dependent Name
DEPENDENT
Sex:{M,F}
BirthDate: Date
Relationship
...
4..*
WORKS_FOR
1..1
1..1
0..1
MANAGES
StartDate
DEPARTMENT
Name
Number
add_employee
number_of_employees
change_manager
...
0..*
1..*
supervisee
0..1
supervisor
1..*
WORKS_ON
Hours
1..1
CONTROLS
*
1..*
PROJECT
Name
Number
add_employee
add_project
change_manager
...
0..*
1..*
LOCATION
Name
1..1
Multiplicity Notation in OMT:
1..1
0..*
0..1
Aggregation Notation in UML:
WHOLE
PART



- Higher Degree Relationships

- In many instances, it is required to represent relationship that is of a degree higher than 2 (HD), i.e. it involves more than two entity types.
- Sometime these relationships can be represented using binary relations, although sometime this may not give the same meaning.
- For example the tuple (s, p, j) which states that SUPPLIER s supplies PART p to PROJECT j may not be expressed by the three tuples (s,p), (s,j) and (p,j).

-- Example: Higher Degree Relationships