# Unit II: Data Modeling and Database Design

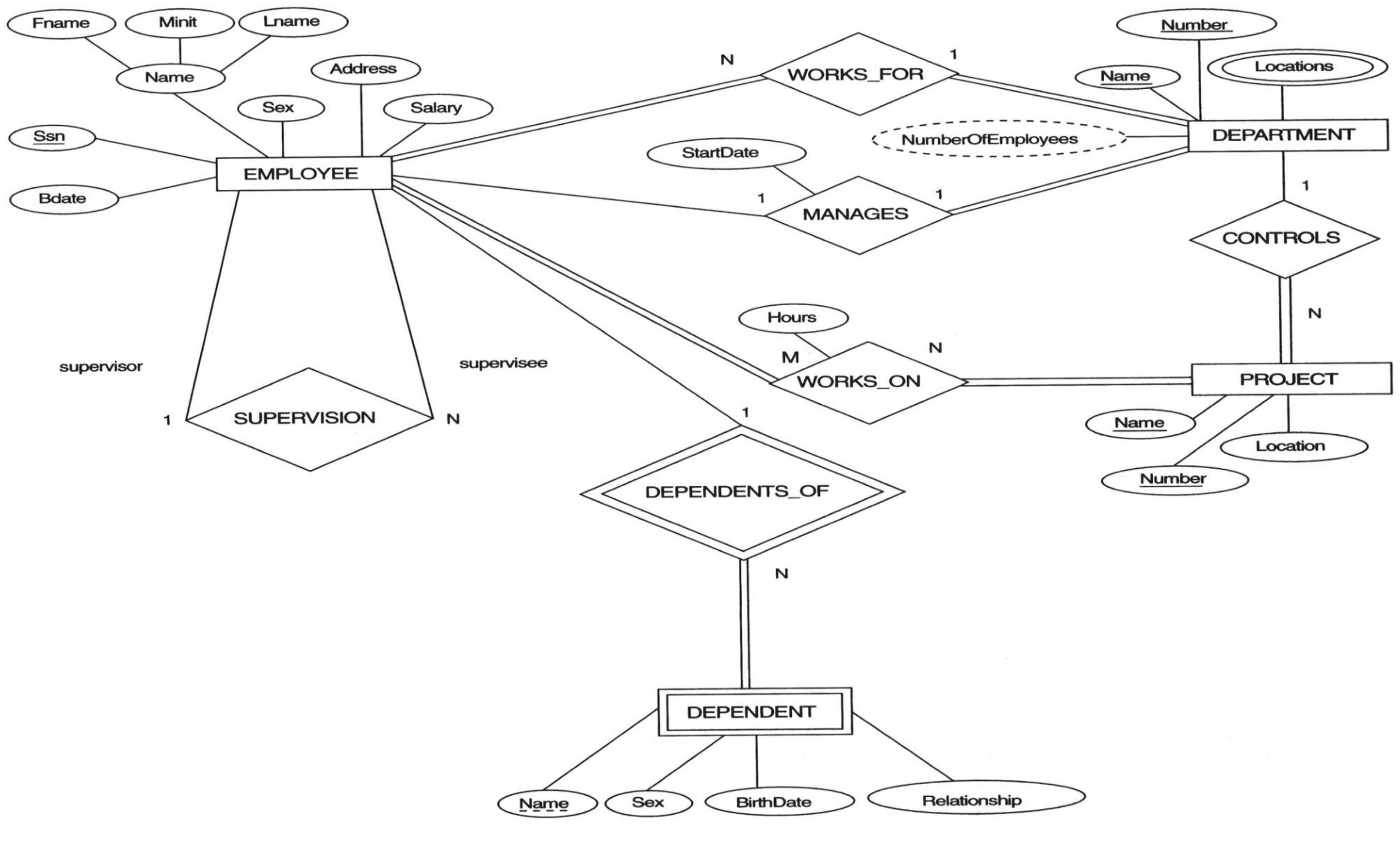
## Chapter 06: Relational database Design

## using

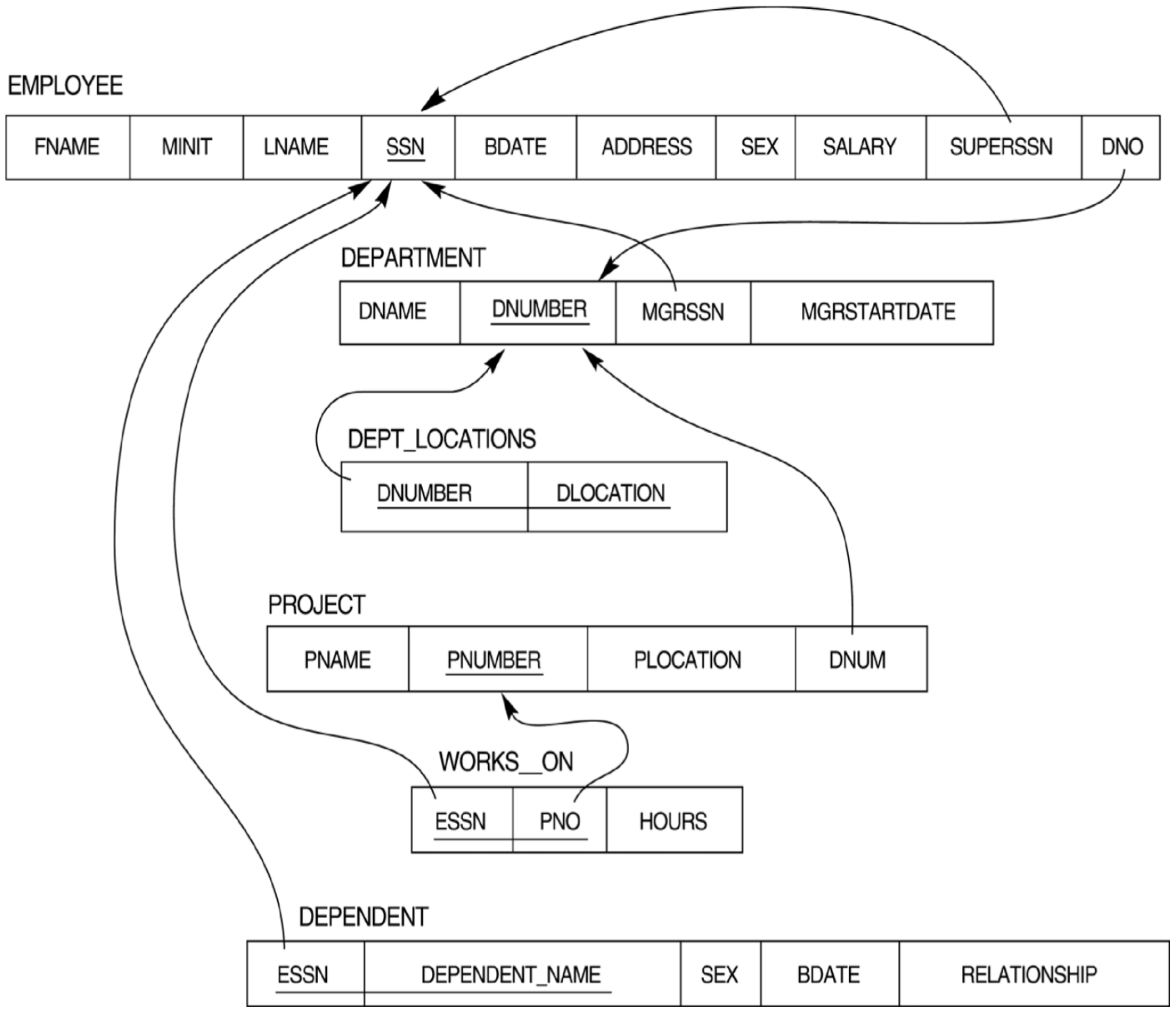
## ER- and EER-to-Relational Mapping Algorithm

# ER-to-Relational Mapping Algorithm

**Figure 6.1: The ER conceptual schema diagram for the COMPANY database**



## Figure 6.2 Result of mapping the COMPANY ER schema into a relational database schema



## Step 1: Mapping of Regular Entity Types

- ✓ For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
- ✓ Choose one of the key attributes of E as the primary key for R.
- ✓ If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

**Example of relation(s) we can create by mapping from Company Conceptual database Schema?**

**?????**

**Example:** We create the relations:
EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram

**EMPLOYEE(fname, minit, lname, <u>ssn</u>, bdate, address, sex, salary)**
**DEPARTMENT( dname, <u>dnumber</u>)**
**PROJECT(pname, <u>pnumber</u>, plocation)**

**ssn, dnumber,** and **pnumber** are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT respectively as shown.

## Step 2: Mapping of Weak Entity Types

- ✓ For each weak entity type W in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- ✓ In addition, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- ✓ The primary key of R is the *combination of* the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

**Example:** Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. Include the primary key **ssn** of the EMPLOYEE relation as a foreign key attribute of DEPENDENT (renamed to **essn**).

**DEPENDENT( <u>essn, dependent_name</u>, sex, bdate, relationship)**

The primary key of the DEPENDENT relation is the combination {**essn, dependent_name**} because **dependent_name** is the partial key of DEPENDENT.

## Step 3: Mapping of Binary 1:1 Relation Types

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

There are **three possible approaches**:

(1) **<u>Foreign Key approach</u>**:

(2) **<u>Merged relation option:</u>**

(3) **<u>Cross-reference or relationship relation option:</u>**

(1) **<u>Foreign Key approach</u>**:

Choose one of the relations, say-S and include a foreign key in S the primary key of T.

It is better to choose an entity type with *total participation* in R in the role of S.

**Example**: 1:1 relation MANAGES is mapped by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total.

**DEPARTMENT(dname, <u>dnumber</u>, mgrssn, mgrstartdate)**

(2) **<u>Merged relation option:</u>**

An alternate mapping of a 1:1 relationship type is possible by merging the two entity types and the relationship into a single relation.

This may be appropriate when *both participations are total.*

(3) **<u>Cross-reference or relationship relation option:</u>**

The third alternative is to set up a third relation R for the purpose of cross referencing the primary keys of the two relations S and T representing the entity types.

## Step 4: Mapping of Binary 1:N Relationship Types.

- ✓ For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
- ✓ Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
- ✓ Include any simple attributes of the 1:N relation type as attributes of S.

**Example:** For CONTROLS, we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the PROJECT relation and call it dnum.

**PROJECT(pname, <u>pnumber</u>, plocation, dnum)**

Similarly for WORKS_FOR & SUPERVISION

**EMPLOYEE(fname, minit, lname, <u>ssn</u>, bdate, address, sex, salary, superssn, dno)**

## Step 5: Mapping of Binary M:N Relationship Types.

- ✓ For each regular binary M:N relationship type R, *create a new relation* S to represent R.
- ✓ Include as foreign key attributes in S the primary keys of the relations that represent

the participating entity types; *their combination will form the primary key* of S.

- ✓ Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

**Example:** The M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema. The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed **pno** and **essn**, respectively.
Attribute **hours** in WORKS_ON represents the **hours** attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {**essn, pno**}.

**WORKS_ON( <u>essn, pno</u>, hours)**

**Step 6: Mapping of Multivalued attributes.**

- ✓ For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type of relationship type that has A as an attribute.

- ✓ The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

**Example:** The relation DEPT_LOCATIONS is created. The attribute **dlocation** represents the multivalued attribute **locations** of DEPARTMENT, while **dnumber**-as foreign key-represents the primary key of the DEPARTMENT relation. The primary key of R is the combination of {**dnumber, dlocation**}.

**DEPT_LOCATIONS( <u>dnumber, dlocation</u> )**

## Step 7: Mapping of N-ary Relationship Types.

- ✓ For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
- ✓ Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- ✓ Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

**Example:** The relationship type SUPPY in the ER below. This can be mapped to the relation SUPPLY shown in the relational schema, whose primary key is the combination of the three foreign keys {SNAME, PARTNO, PROJNAME}

**Figure 6.3.1 Ternary relationship types. (a) The SUPPLY relationship.**

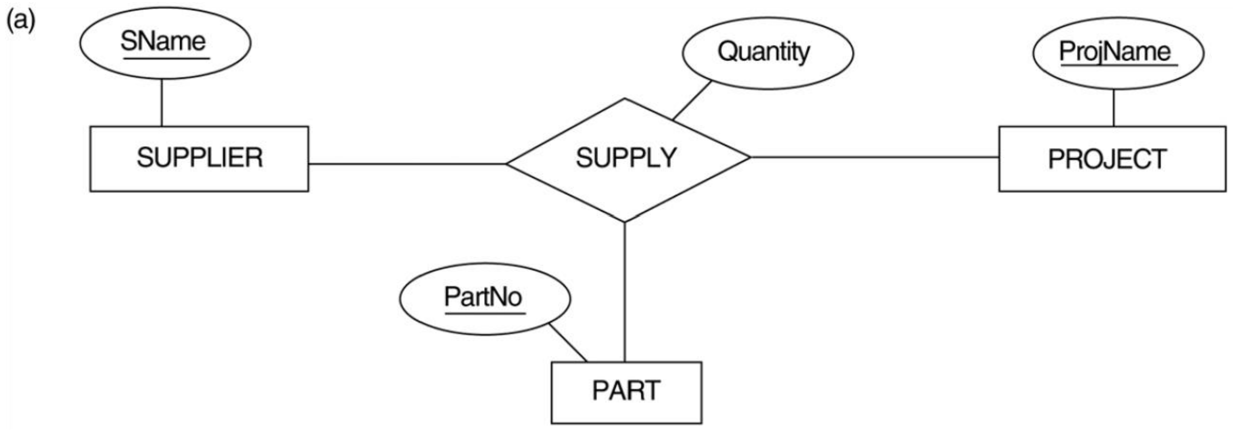
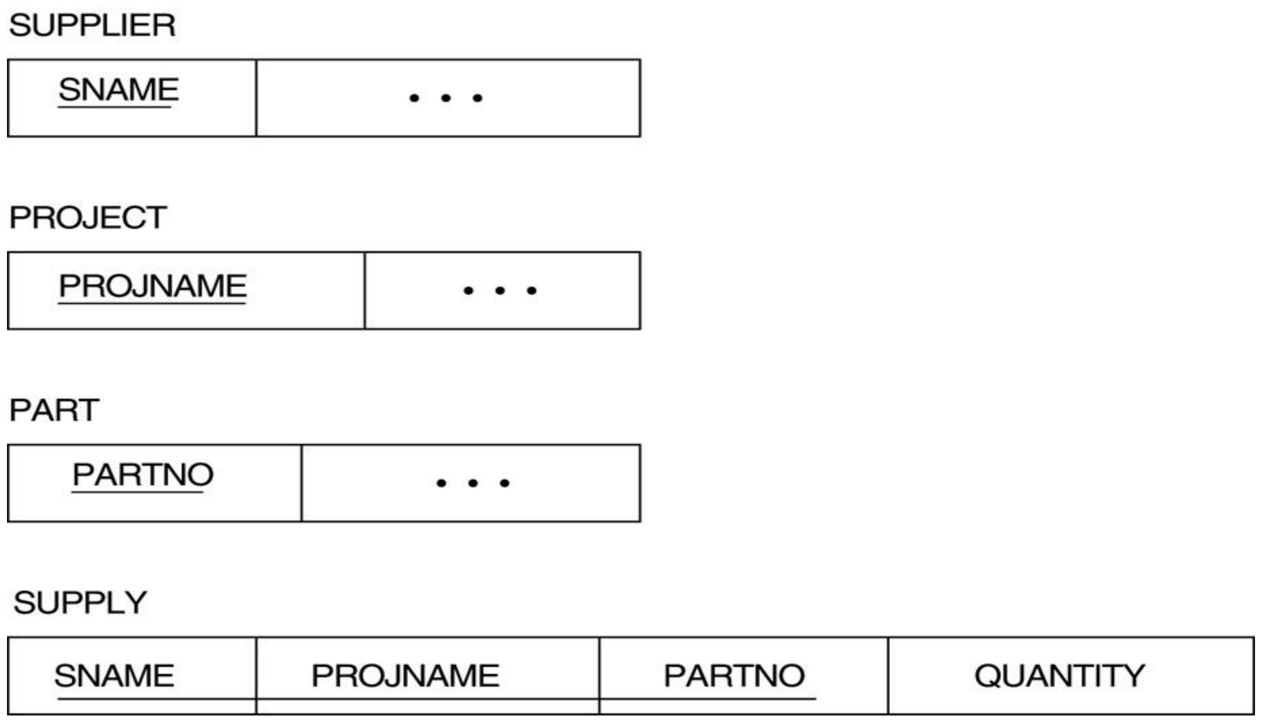


**Figure 6.3.2 Mapping the n-ary relationship type SUPPLY from Figure 6.3.1(a)**



## Summary of Mapping constructs and constraints

***Table 7.1 Correspondence between ER and Relational Models***

| **ER Model** | **Relational Model** |
|---|---|
| Entity type | "Entity" relation |
| **1:1 or 1:N relationship type** | **Foreign key (or "relationship" relation)** |
| M:N relationship type | "Relationship" relation and two foreign keys |

| | |
|---|---|
| ***n*-ary relationship type** | **“Relationship” relation and n foreign keys** |
| Simple attribute | Attribute |
| **Composite attribute** | **Set of simple component attributes** |
| Multivalued attribute | Relation and foreign key |
| **Value set** | **Domain** |
| Key attribute | Primary (or secondary) key |

## Mapping EER Model Constructs Relations

### Step 8: Options for Mapping Specialization or Generalization.

Convert each specialization with m subclasses {$S_1$, $S_2$,….,$S_m$} and generalized superclass C, where the attributes of C are {k,$a_1$,…$a_n$} and k is the (primary) key, into relational schemas using one of the four following options:

**Option 8A: Multiple relations-Superclass and subclasses.**

- ✓ Create a relation L for C with attributes Attrs(L) = {k,$a_1$,…$a_n$} and PK(L) = k.
- ✓ Create a relation $L_i$ for each subclass $S_i$, $1 < i < m$, with the attributes Attrs($L_i$) = {k} U {attributes of $S_i$}

and PK($L_i$)=k. This option works **for any specialization** (total or partial, disjoint of over-lapping).

**Option 8B: Multiple relations-Subclass relations only**

- ✓ Create a relation $L_i$ for each subclass $S_i$, $1 < i < m$, with the attributes Attr($L_i$) = {attributes of $S_i$} U {k,$a_1$…,$a_n$} and PK($L_i$) = k. This option only works for a specialization whose subclasses are **total** (every entity in the superclass must belong to (at least) one of the subclasses).

**Figure 6.4 EER diagram notataion for an attribute-defined specialization on JobType.**

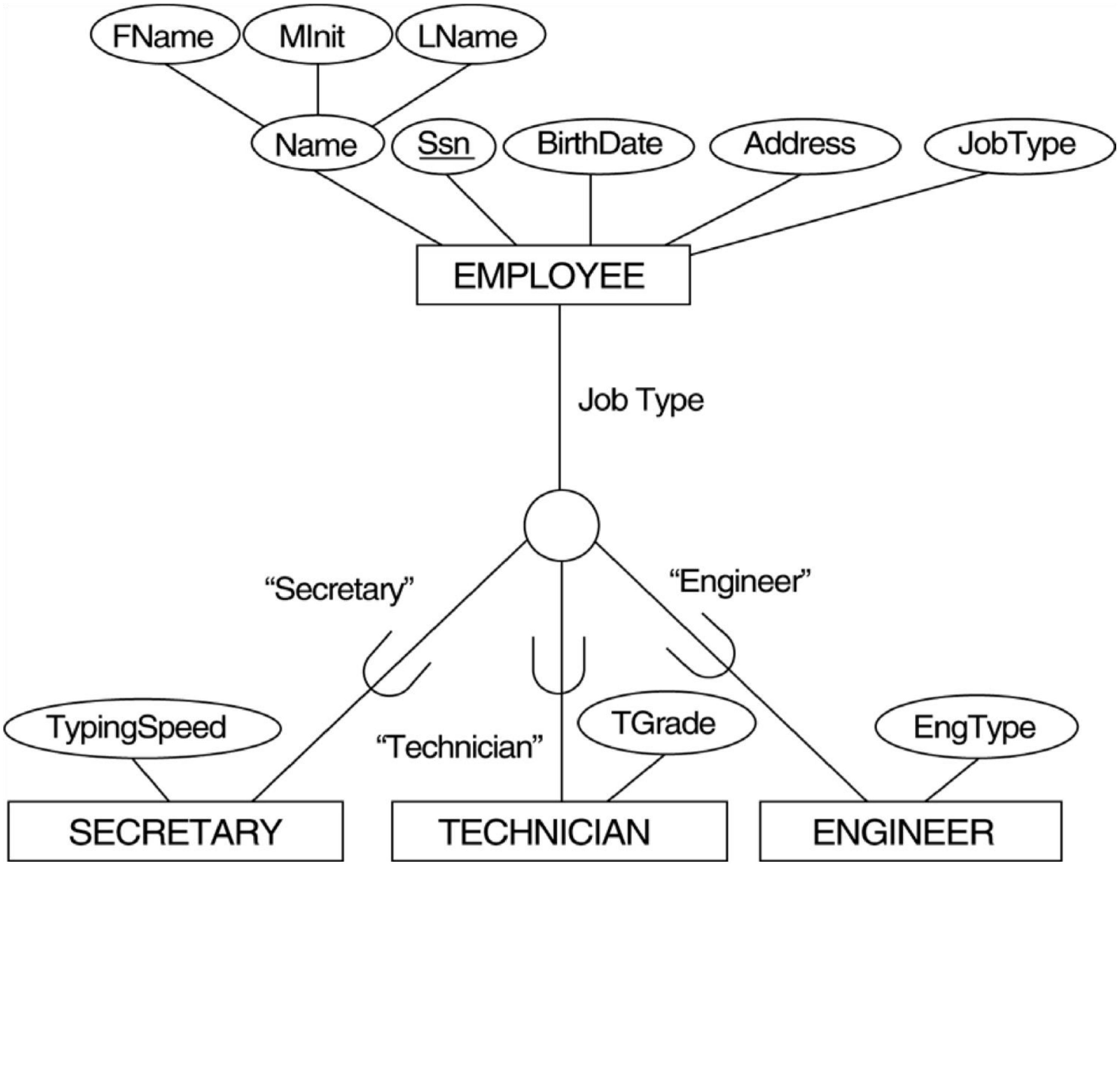
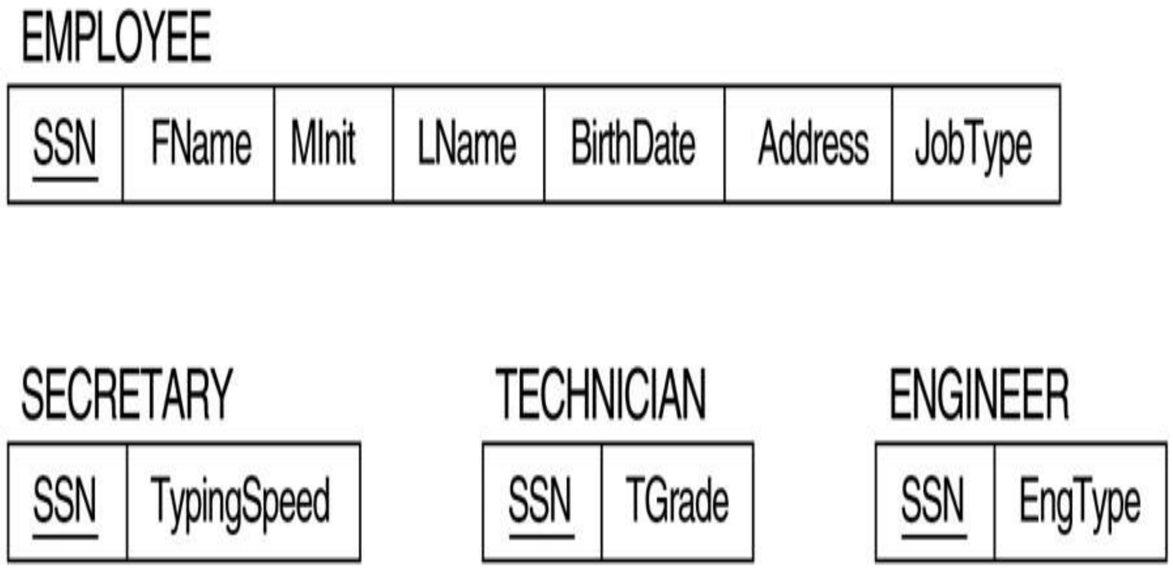


**Figure 6.5 Options for mapping specialization or generalization. (a) Mapping the EER schema in Figure 6.4 using option 8A.**

(a)



EMPLOYEE

| SSN | FName | MInit | LName | BirthDate | Address | JobType |
|---|---|---|---|---|---|---|

SECRETARY

| SSN | TypingSpeed |
|---|---|

TECHNICIAN

| SSN | TGrade |
|---|---|

ENGINEER

| SSN | EngType |
|---|---|

**Figure 6.6 Generalization. (b) Generalizing CAR and TRUCK into the superclass VEHICLE.**

(b)

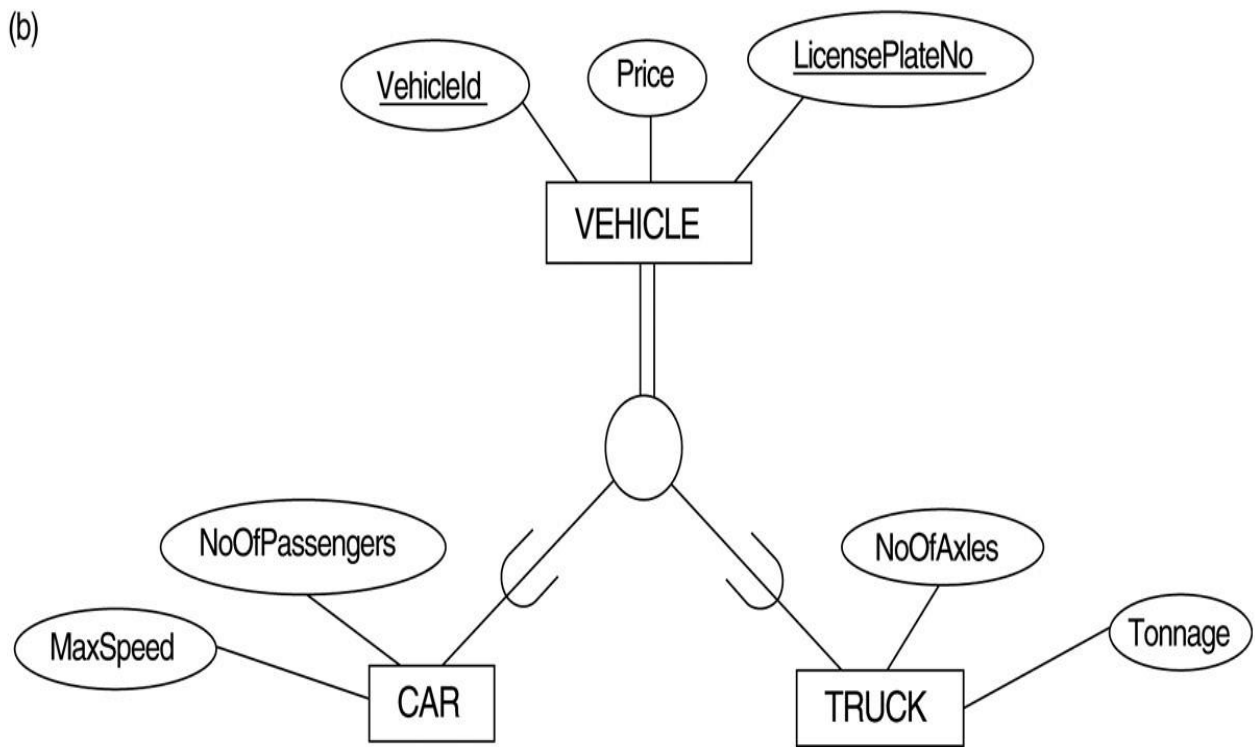
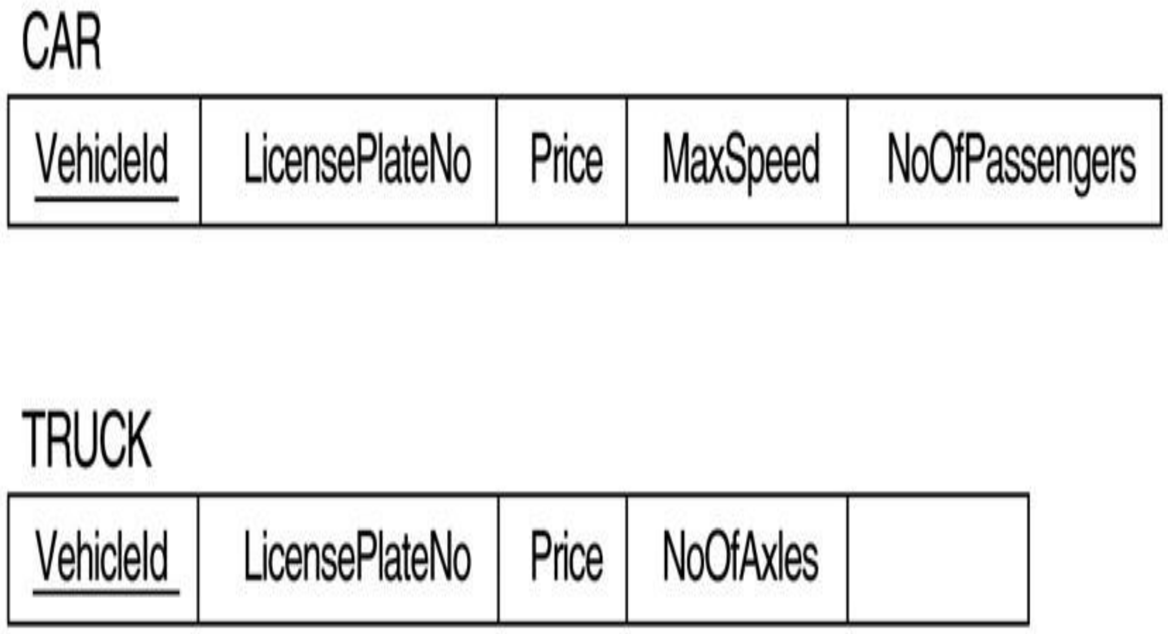


**Figure 6.7 Options for mapping specialization or generalization. (b) Mapping the EER schema in Figure 6.6 b using option 8B.**

(b)



CAR

| VehicleId | LicensePlateNo | Price | MaxSpeed | NoOfPassengers |
|---|---|---|---|---|

TRUCK

| VehicleId | LicensePlateNo | Price | NoOfAxles | |
|---|---|---|---|---|

**Option 8C: Single relation with one type attribute.**

Create a single relation L with attributes Attrs(L) = $\{k, a_1, \ldots a_n\}$ U {attributes of $S_1$} U…U {attributes of $S_m$} U {t} and PK(L) = k.

The attribute t is called a type (or **discriminating**) attribute that indicates the subclass to which each tuple belongs

**Option 8D: Single relation with multiple type attributes.**

Create a single relation schema L with attributes

Attrs(L) = $\{k, a_1, \ldots a_n\}$ U {attributes of $S_1$} U…U {attributes of $S_m$} U $\{t_1, t_2, \ldots, t_m\}$ and PK(L) = k.

Each $t_i$, 1 < I < m, is a Boolean type attribute indicating whether a tuple belongs to the subclass $S_i$.

**Figure 6.10 EER diagram notation for an attribute-defined specialization on JobType.**

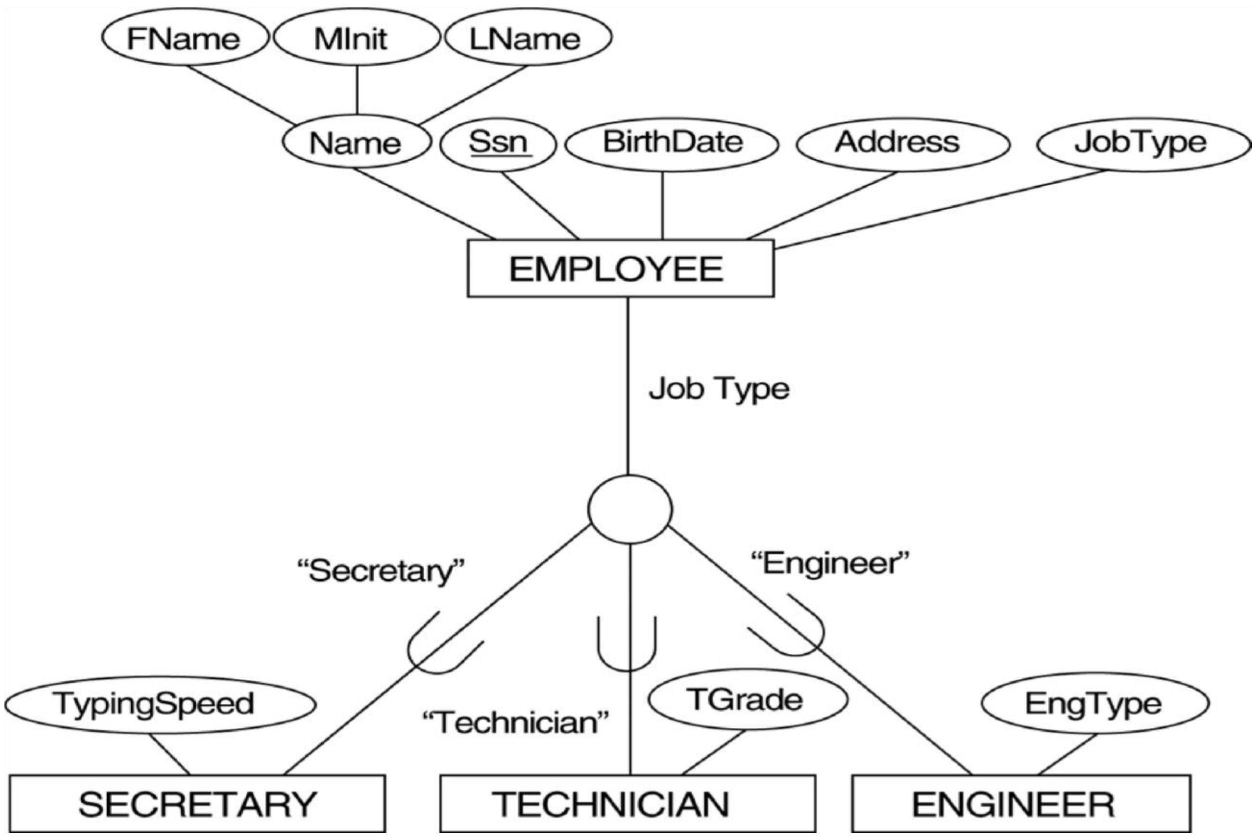
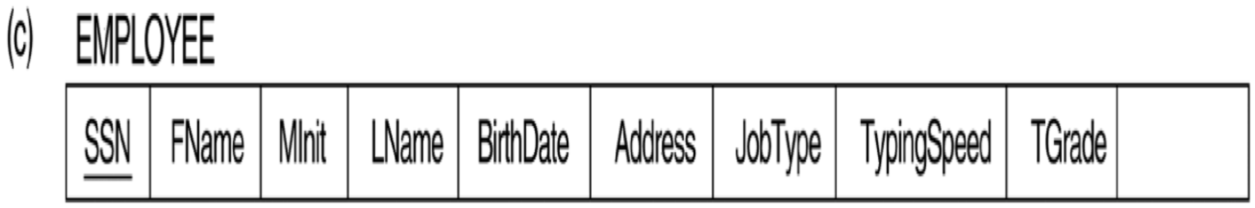


**Figure 7.4 Options for mapping specialization or generalization. (c) Mapping the EER schema in Figure 4.4 using option 8C.**



(c) EMPLOYEE

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| SSN | FName | MInit | LName | BirthDate | Address | JobType | TypingSpeed | TGrade | |

**Figure 6.11:** EER diagram notation for an overlapping (nondisjoint) specialization.

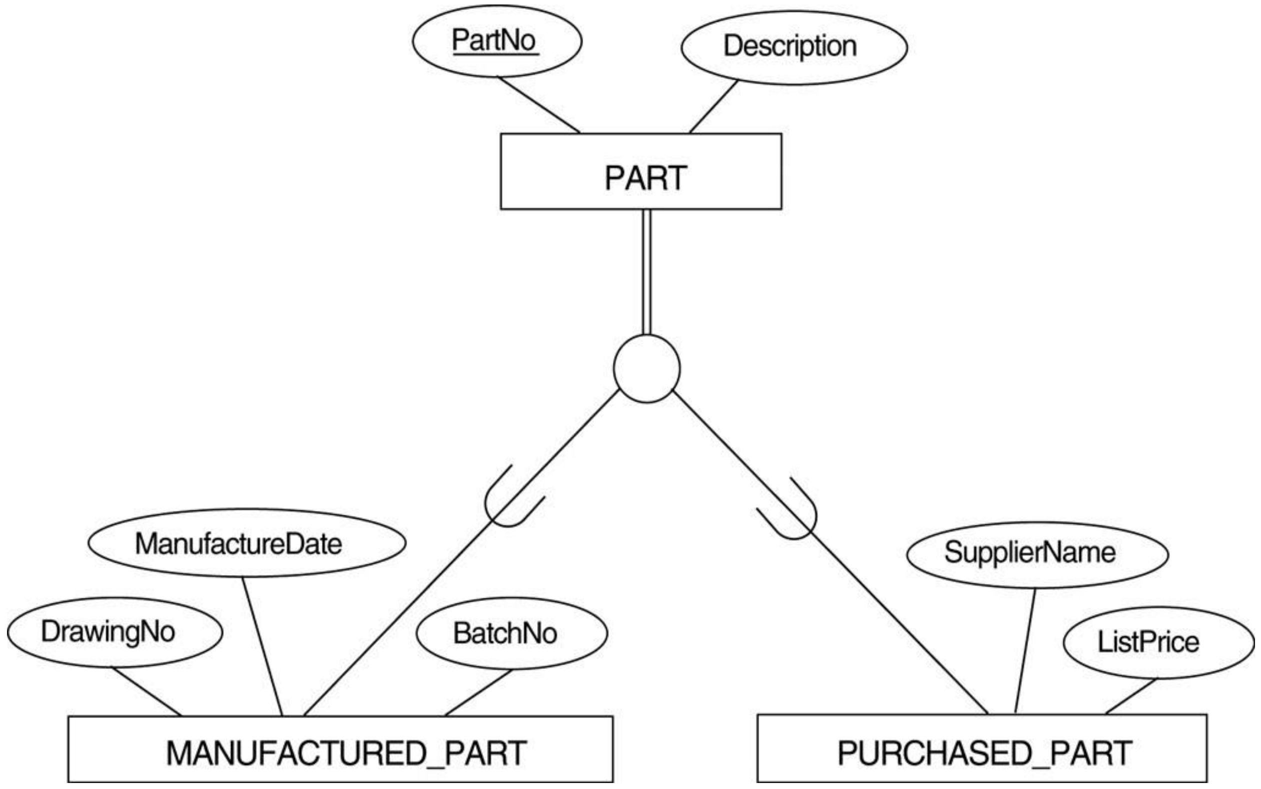
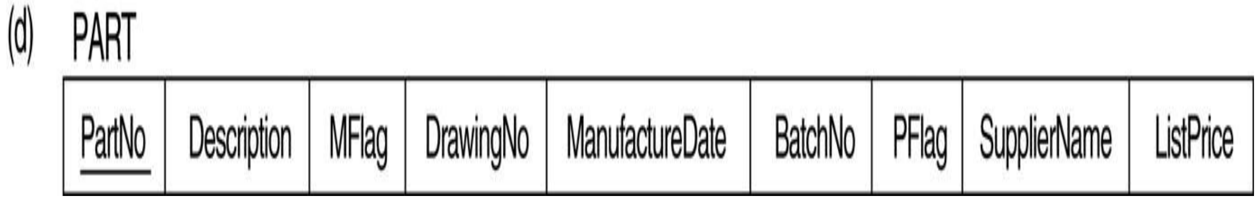


**Figure 6.12: Options for mapping specialization or generalization.**

**(d) Mapping Figure 6.11 using option 8D with Boolean type fields Mflag and Pflag.**


(d) PART

| PartNo | Description | MFlag | DrawingNo | ManufactureDate | BatchNo | PFlag | SupplierName | ListPrice |
|---|---|---|---|---|---|---|---|---|

**Mapping of Shared Subclasses (Multiple Inheritance)**

A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance. These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a category.

We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm. Below both 8C and 8D are used for the shared class STUDENT_ASSISTANT.

**Figure 6.13: A Specialization lattice with multiple inheritance for a UNIVERSITY database**

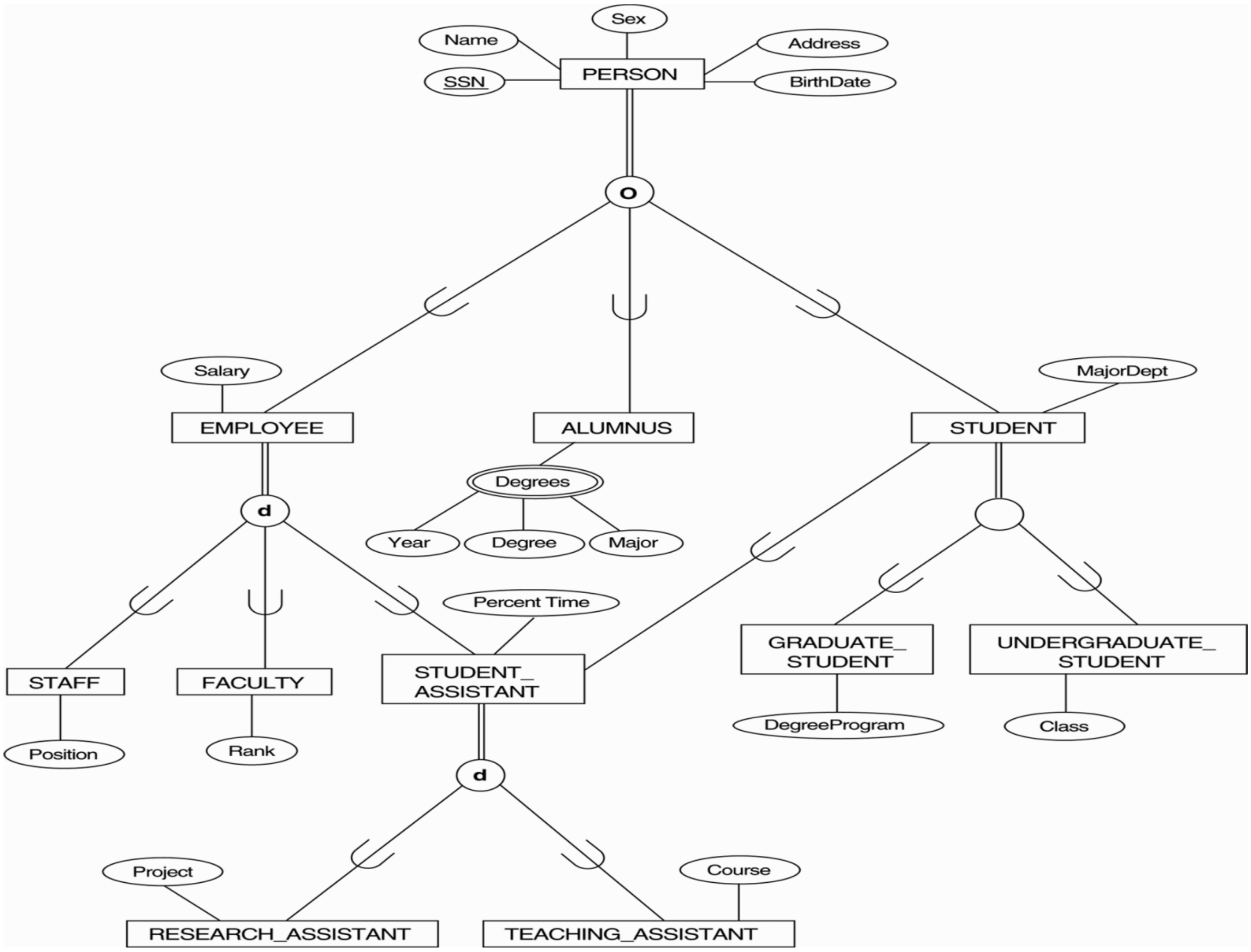

Sex
Name
Address
SSN
PERSON
BirthDate
o
Salary
MajorDept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
d
Year
Degree
Major
Percent Time
GRADUATE_
STUDENT
UNDERGRADUATE_
STUDENT
STAFF
FACULTY
STUDENT_
ASSISTANT
DegreeProgram
Class
Position
Rank
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT

**FIGURE 7.5 Mapping the EER specialization lattice in Figure 4.6 using multiple options.**

PERSON

| SSN | Name | BirthDate | Sex | Address |
|---|---|---|---|---|

EMPLOYEE

| SSN | Salary | EmployeeType | Position | Rank | PercentTime | RAFlag | TAFlag | Project | |
|---|---|---|---|---|---|---|---|---|---|

ALUMNUS

| SSN |
|---|

ALUMNUS_DEGREES

| SSN | Year | Degree | |
|---|---|---|---|

STUDENT

| SSN | MajorDept | GradFlag | UndergradFlag | DegreeProgram | Class | StudAssistFlag |
|---|---|---|---|---|---|---|