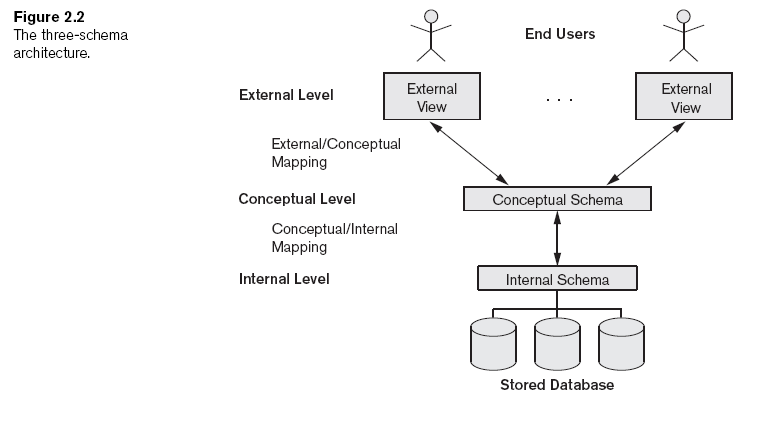
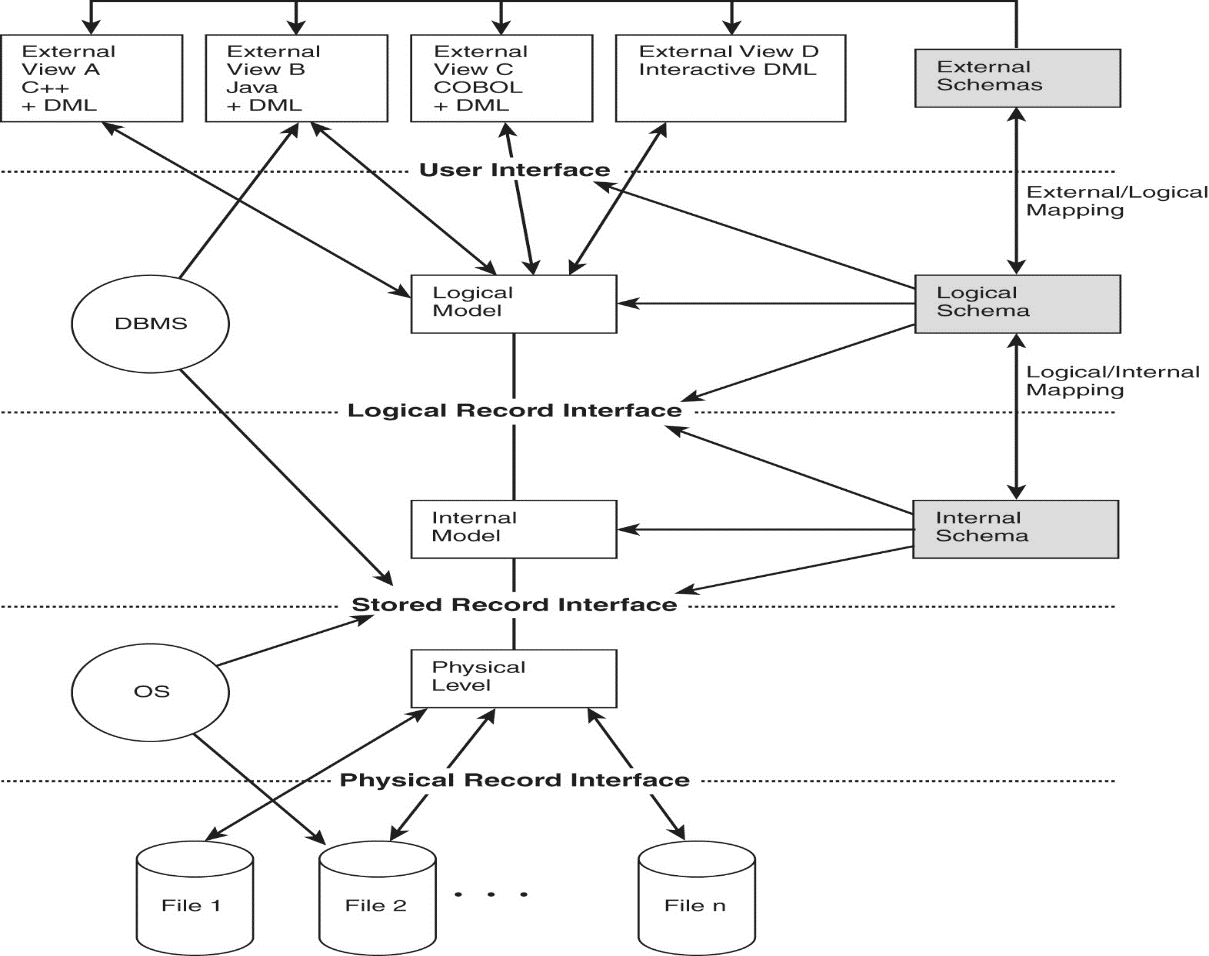


Schemas, Instances, and Database State (cont'd.)

**Three-Schema Architecture and Data Independence (cont'd.)**

****



**A Sample Database Application**

In this section we describe a sample database application, called COMPANY, which

serves to illustrate the basic ER model concepts and their use in schema design.We

list the data requirements for the database here, and then create its conceptual

schema step-by-step as we introduce the modeling concepts of the ER model. The

COMPANY database keeps track of a company’s employees, departments, and projects.

Suppose that after the requirements collection and analysis phase, the database

designers provide the following description of the *miniworld*—the part of the company

that will be represented in the database.

■ The company is organized into departments. Each department has a unique

name, a unique number, and a particular employee who manages the

department.We keep track of the start date when that employee began managing

the department. A department may have several locations.

■ A department controls a number of projects, each of which has a unique

name, a unique number, and a single location.

■ We store each employee’s name, Social Security number,2 address, salary, sex

(gender), and birth date. An employee is assigned to one department, but

may work on several projects, which are not necessarily controlled by the

same department. We keep track of the current number of hours per week

that an employee works on each project. We also keep track of the direct

supervisor of each employee (who is another employee).

■ We want to keep track of the dependents of each employee for insurance

purposes.We keep each dependent’s first name, sex, birth date, and relationship

to the employee.

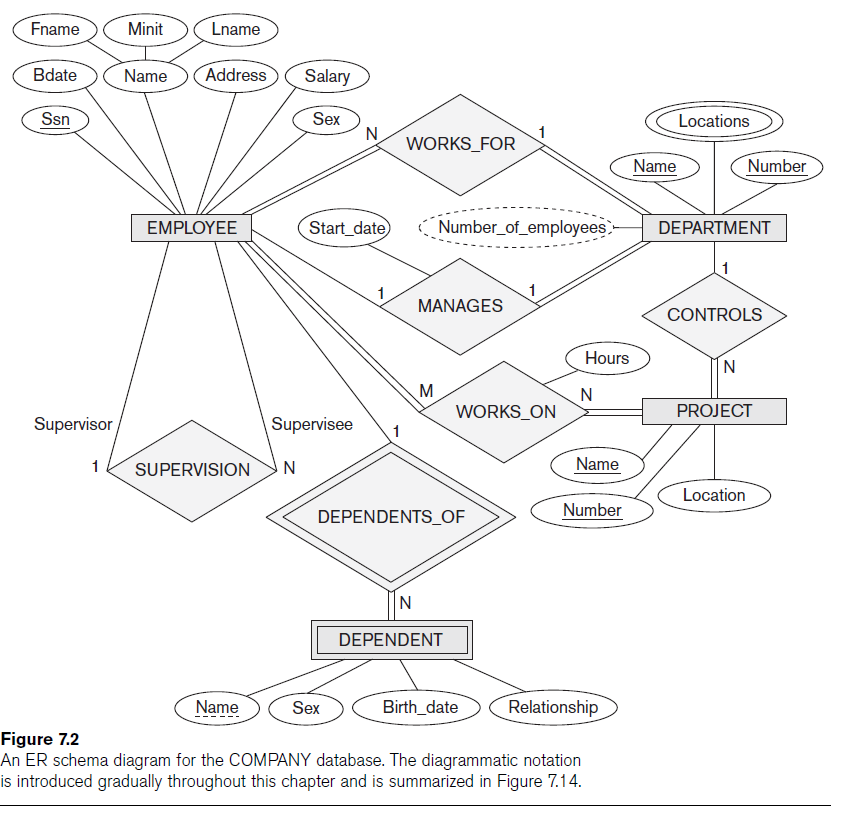
Figure 7.2 shows how the schema for this database application can be displayed by

means of the graphical notation known as **ER diagrams**. This figure will be

explained gradually as the ER model concepts are presented.We describe the stepby-

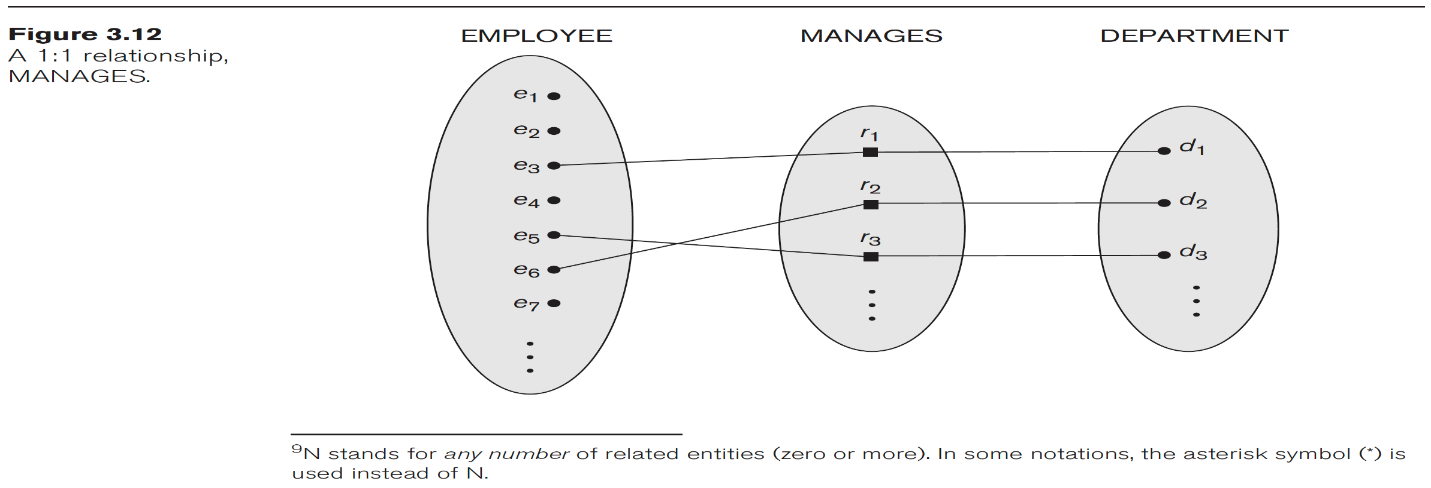
step process of deriving this schema from the stated requirements—and explain

the ER diagrammatic notation—as we introduce the ER model concepts.

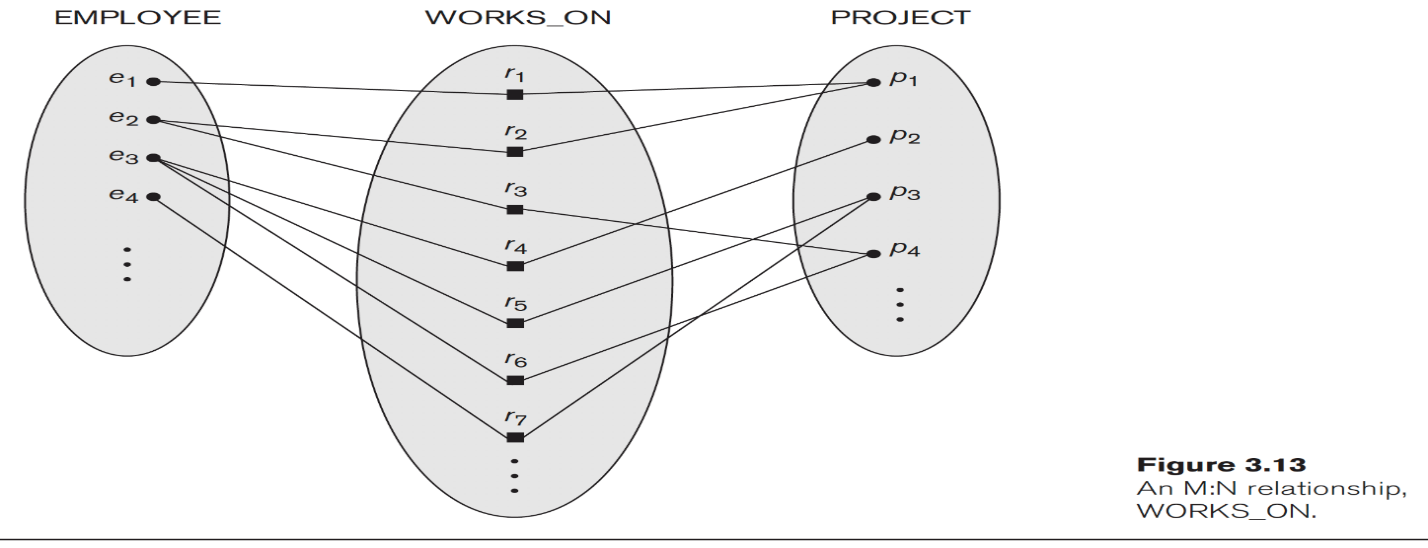


**Constraints on Binary Relationship Types**

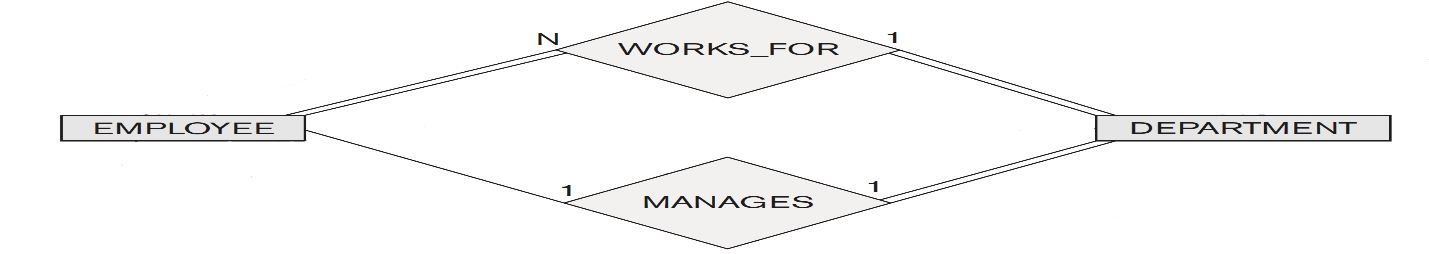
* Cardinality ratio for a binary relationship
  + Specifies maximum number of relationship instances that entity can participate in



* **Cardinality ratio for a binary relationship**

****

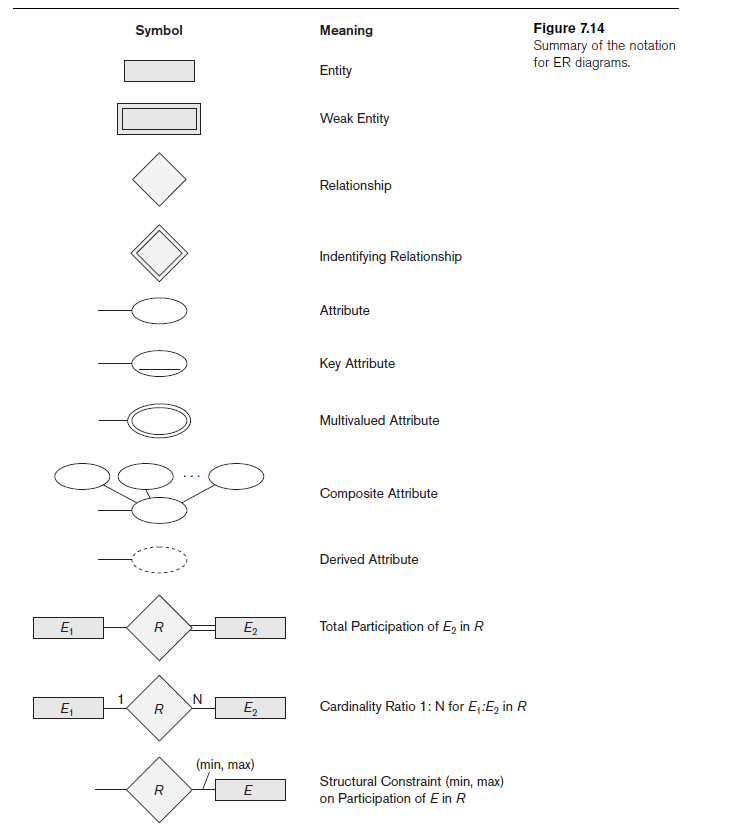
* **Participation constraint** 
  + Specifies whether existence of entity depends on its being related to another entity
  + Types: total and partial
  + Total for double line and partial for single line

****

**Attributes of Relationship Types**

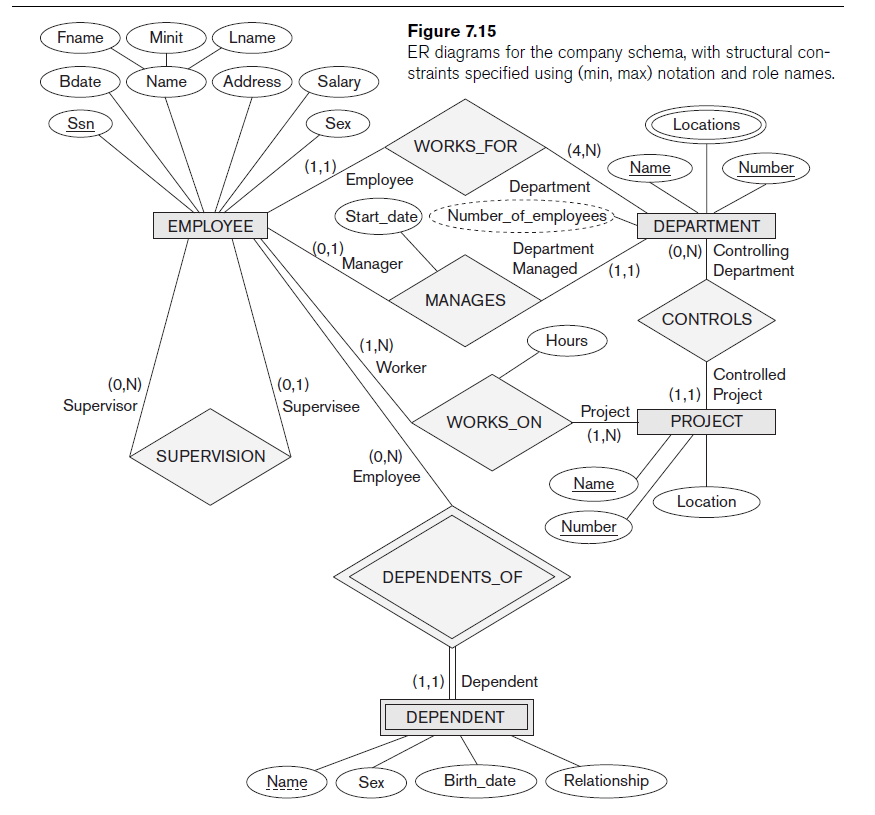
* Attributes of 1:1 or 1:N relationship types can be migrated to one entity type
* For a 1:N relationship type
  + Relationship attribute can be migrated only to entity type on N-side of relationship
* For M:N relationship types
  + Some attributes may be determined by combination of participating entities
  + Must be specified as relationship attributes

**ER Diagrams, Naming Conventions, and Design Issues**

****

**Alternative Notations for ER Diagrams**

* Specify structural constraints on relationships
  + Replaces cardinality ratio (1:1, 1:N, M:N) and single/double line notation for participation constraints
  + Associate a pair of integer numbers (min, max) with each participation of an entity type *E* in a relationship type *R*, where 0 ≤ min ≤ max and max ≥ 1

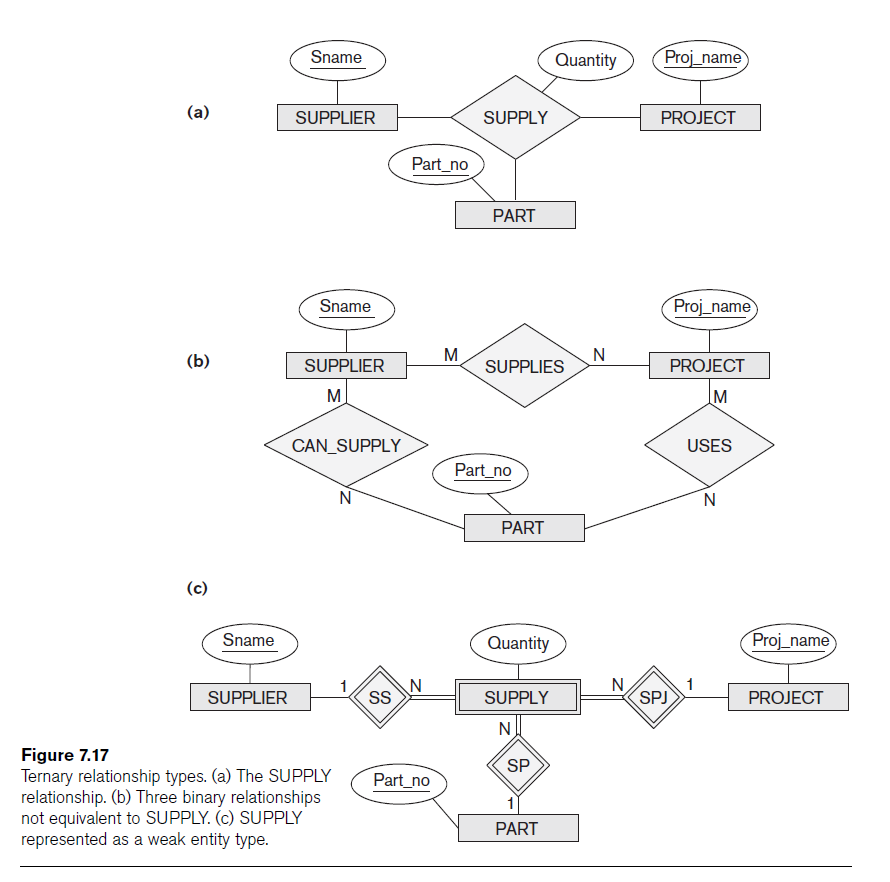


**Relationship Types of Degree Higher than Two**

* Degree of a relationship type
  + Number of participating entity types
* *Binary*
  + Relationship type of degree two
* *Ternary*
  + Relationship type of degree three

**Choosing between Binary and Ternary (or Higher-Degree) Relationships**

* Some database design tools permit only binary relationships
  + Ternary relationship must be represented as a weak entity type
  + No partial key and three identifying relationships
* Represent ternary relationship as a regular entity type
  + By introducing an artificial or surrogate key

****

**Subclasses, Superclasses, and Inheritance (cont’d.)**

The first Enhanced ER (EER) model concept we take up is that of a **subtype** or

**subclass** of an entity type. As we discussed in Chapter 7, an entity type is used to

represent both a *type of entity* and the *entity set* or *collection of entities of that type*

that exist in the database. For example, the entity type EMPLOYEE describes the type

(that is, the attributes and relationships) of each employee entity, and also refers to

the current set of EMPLOYEE entities in the COMPANY database. In many cases an

entity type has numerous subgroupings or subtypes of its entities that are meaningful

and need to be represented explicitly because of their significance to the database

application. For example, the entities that are members of the EMPLOYEE entity

type may be distinguished further into SECRETARY, ENGINEER, MANAGER,

TECHNICIAN, SALARIED\_EMPLOYEE, HOURLY\_EMPLOYEE, and so on. The set of

entities in each of the latter groupings is a subset of the entities that belong to the

EMPLOYEE entity set, meaning that every entity that is a member of one of these

subgroupings is also an employee.We call each of these subgroupings a **subclass** or

**subtype** of the EMPLOYEE entity type, and the EMPLOYEE entity type is called the

**superclass** or **supertype** for each of these subclasses. Figure 8.1 shows how to represent

these concepts diagramatically in EER diagrams. (The circle notation in Figure

8.1 will be explained in Section 8.2.)

We call the relationship between a superclass and any one of its subclasses a

**superclass/subclass** or **supertype/subtype** or simply **class/subclass relationship**.3

In our previous example, EMPLOYEE/SECRETARY and EMPLOYEE/TECHNICIAN are

two class/subclass relationships. Notice that a member entity of the subclass represents

the *same real-world entity* as some member of the superclass; for example, a

SECRETARY entity ‘Joan Logano’ is also the EMPLOYEE ‘Joan Logano.’ Hence, the

subclass member is the same as the entity in the superclass, but in a distinct *specific*

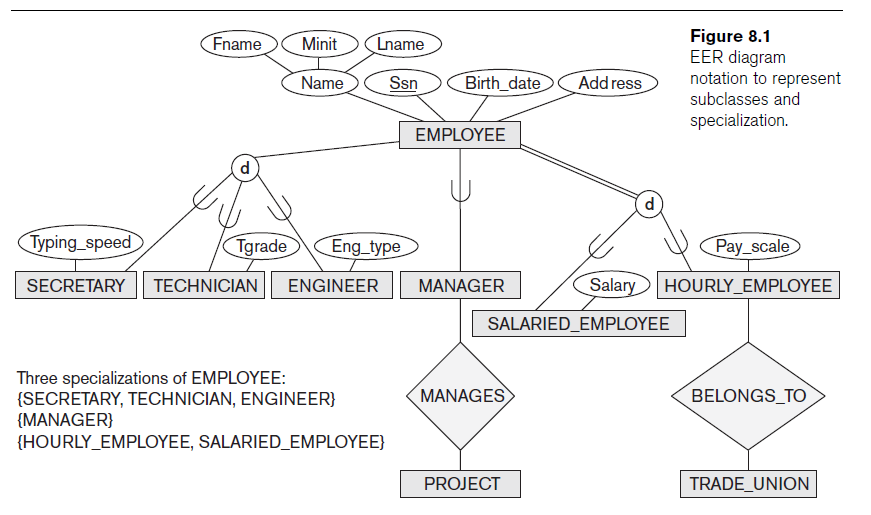
*role.*When we implement a superclass/subclass relationship in the database system,

however, we may represent a member of the subclass as a distinct database object—

say, a distinct record that is related via the key attribute to its superclass entity. In

Section 9.2, we discuss various options for representing superclass/subclass relationships

in relational databases.

****

**8.3.2 Specialization and Generalization Hierarchies**

**and Lattices**

A subclass itself may have further subclasses specified on it, forming a hierarchy or a

lattice of specializations. For example, in Figure 8.6 ENGINEER is a subclass of

EMPLOYEE and is also a superclass of ENGINEERING\_MANAGER; this represents

the real-world constraint that every engineering manager is required to be an engineer.

A **specialization hierarchy** has the constraint that every subclass participates

*as a subclass* in *only one* class/subclass relationship; that is, each subclass has onlyone parent, which results in a **tree structure** or **strict hierarchy**. In contrast, for a

**specialization lattice**, a subclass can be a subclass in *more than one* class/subclass

relationship. Hence, Figure 8.6 is a lattice.

Figure 8.7 shows another specialization lattice of more than one level. This may be

part of a conceptual schema for a UNIVERSITY database. Notice that this arrangement

would have been a hierarchy except for the STUDENT\_ASSISTANT subclass,

which is a subclass in two distinct class/subclass relationships.

The requirements for the part of the UNIVERSITY database shown in Figure 8.7 are

the following:

**1.** The database keeps track of three types of persons: employees, alumni, and

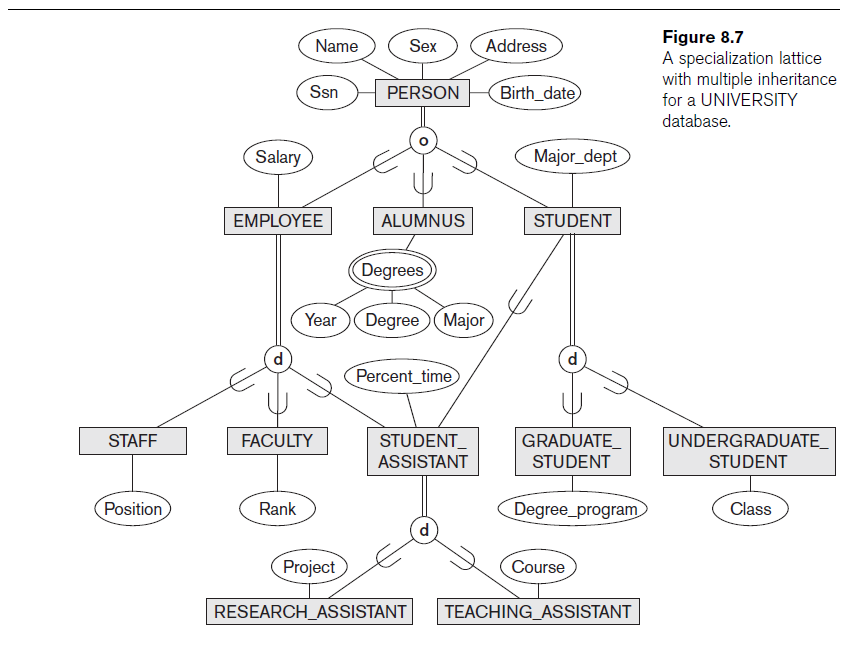
students. A person can belong to one, two, or all three of these types. Each

person has a name, SSN, sex, address, and birth date.

**2.** Every employee has a salary, and there are three types of employees: faculty,

staff, and student assistants. Each employee belongs to exactly one of these

types. For each alumnus, a record of the degree or degrees that he or she



earned at the university is kept, including the name of the degree, the year

granted, and the major department. Each student has a major department.

**3.** Each faculty has a rank, whereas each staff member has a staff position.

Student assistants are classified further as either research assistants or teaching

assistants, and the percent of time that they work is recorded in the database.

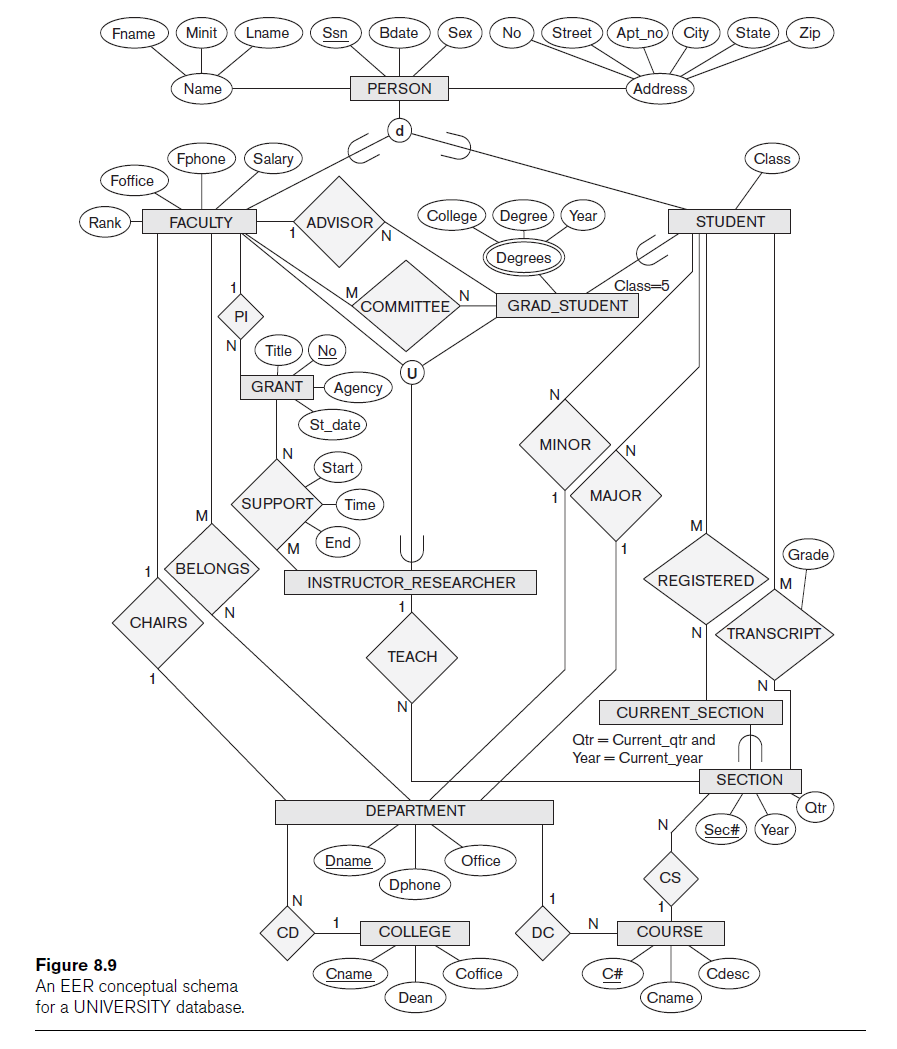
Research assistants have their research project stored, whereas teaching

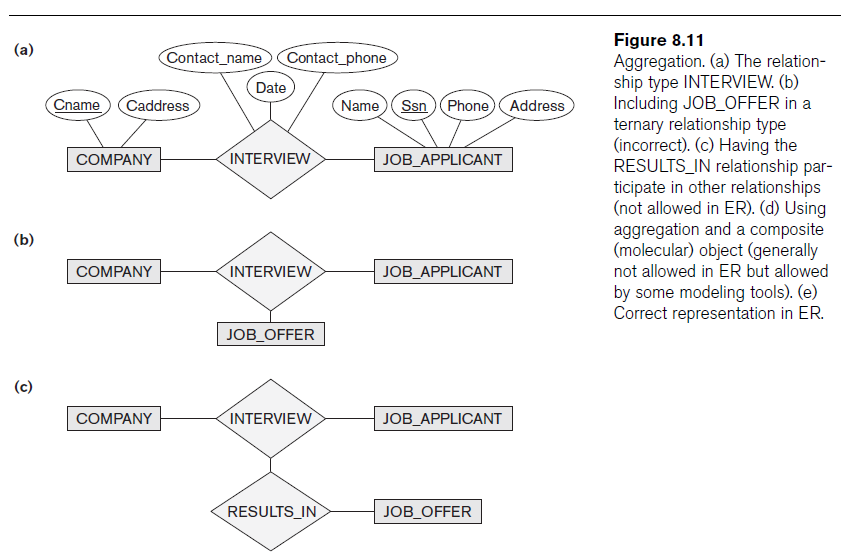
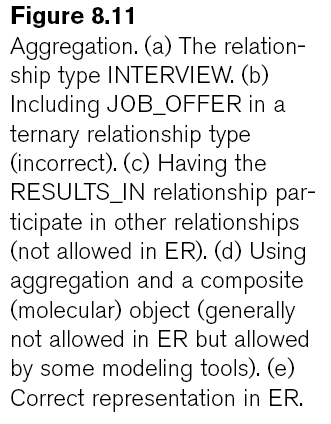
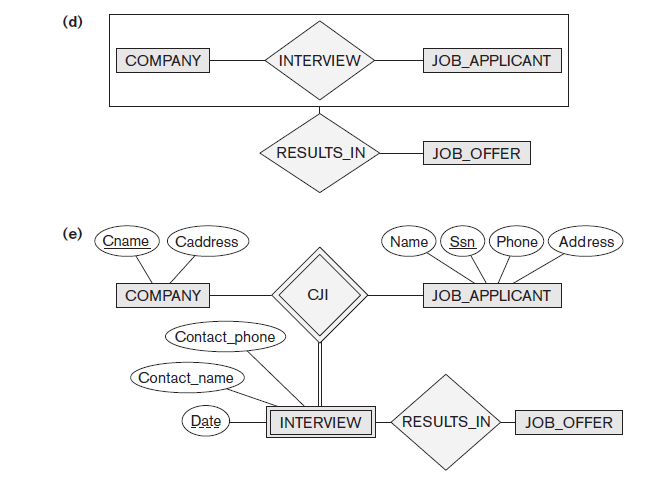
assistants have the current course they work on.

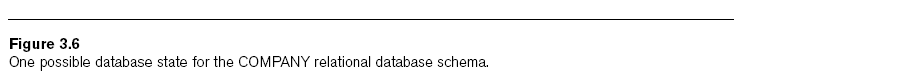
**4.** Students are further classified as either graduate or undergraduate, with the

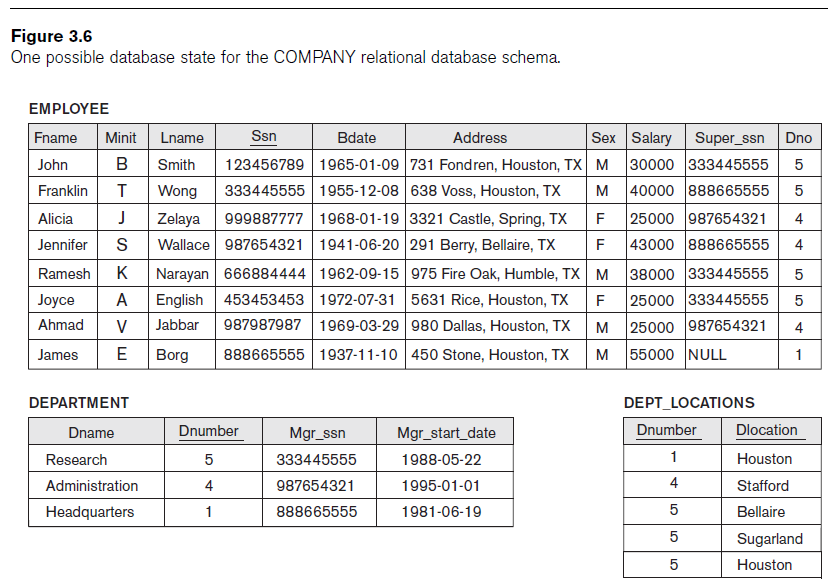
specific attributes degree program (M.S., Ph.D., M.B.A., and so on) for

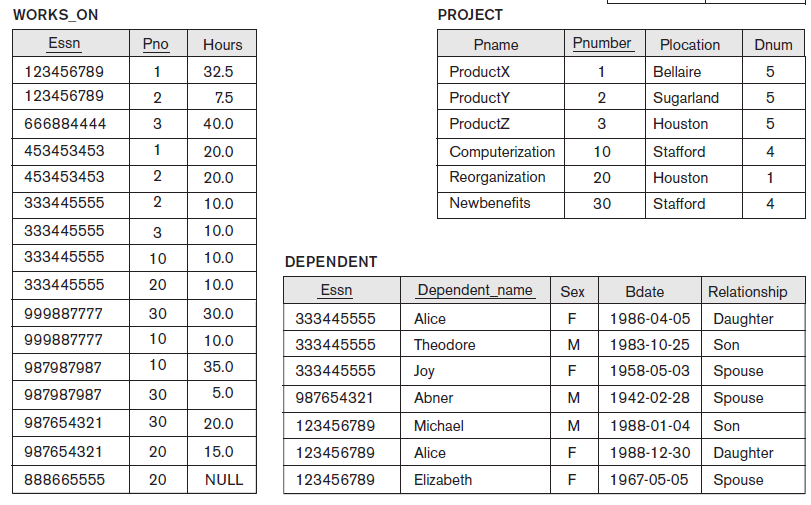
graduate students and class (freshman, sophomore, and so on) for undergraduates.

****

****



**\***

****

**3.11.** Suppose that each of the following Update operations is applied directly to the database state shown in Figure 3.6. Discuss *all* integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.

a. Insert <‘Robert’, ‘F’, ‘Scott’, ‘943775543’, ‘1972-06-21’, ‘2365 Newcastle Rd,

Bellaire, TX’, M, 58000, ‘888665555’, 1> into EMPLOYEE.

b. Insert <‘ProductA’, 4, ‘Bellaire’, 2> into PROJECT.

c. Insert <‘Production’, 4, ‘943775543’, ‘2007-10-01’> into DEPARTMENT.

Ans:

d. Insert <‘677678989’, NULL, ‘40.0’> into WORKS\_ON.

e. Insert <‘453453453’, ‘John’, ‘M’, ‘1990-12-12’, ‘spouse’> into DEPENDENT.

f. Delete the WORKS\_ON tuples with Essn = ‘333445555’.

g. Delete the EMPLOYEE tuple with Ssn = ‘987654321’.

h. Delete the PROJECT tuple with Pname = ‘ProductX’.

i. Modify the Mgr\_ssn and Mgr\_start\_date of the DEPARTMENT tuple with

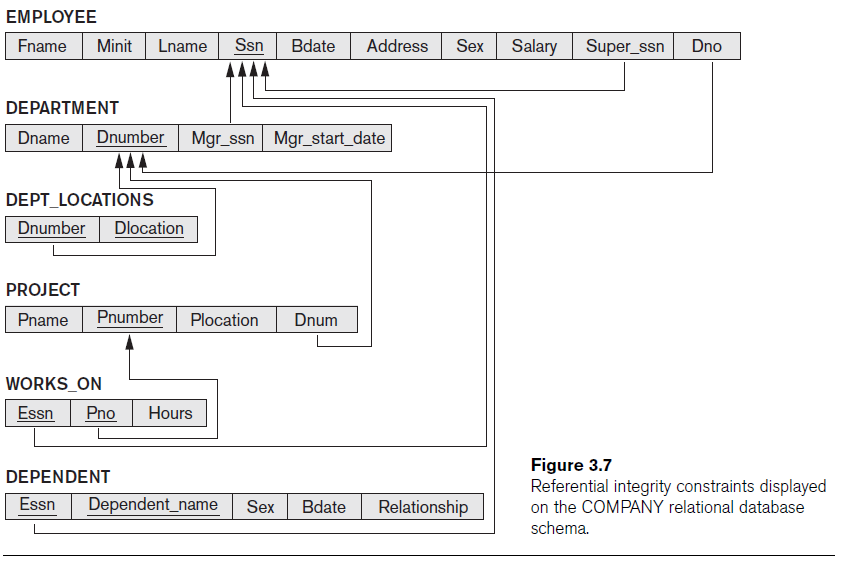
Dnumber = 5 to ‘123456789’ and ‘2007-10-01’, respectively.

j. Modify the Super\_ssn attribute of the EMPLOYEE tuple with Ssn =

‘999887777’ to ‘943775543’.

k. Modify the Hours attribute of the WORKS\_ON tuple with Essn =

‘999887777’ and Pno = 10 to ‘5.0’.

****

Specify the following queries on the COMPANY relational database schema shown in above Figure, using the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state of Figure 3.6.

(1) Retrieve the names of employees in department 5 who work more than 10 hours per week on the 'ProductX' project.

Ans: **SELECT** LNAME, FNAME

**FROM** EMPLOYEE, WORKS\_ON, PROJECT

**WHERE** DNO=5 AND SSN=ESSN AND PNO=PNUMBER AND PNAME='ProductX' AND HOURS>10

(2) List the names of employees who have a dependent with the same first name as themselves.

Ans: **SELECT** LNAME, FNAME

**FROM** EMPLOYEE, DEPENDENT

**WHERE** SSN=ESSN AND FNAME=DEPENDENT\_NAME

(3) Find the names of employees that are directly supervised by 'Franklin Wong'.

Ans : **SELECT** E.LNAME, E.FNAME

**FROM** EMPLOYEE E, EMPLOYEE S

**WHERE** S.FNAME='Franklin' AND S.LNAME='Wong' AND E.SUPERSSN=S.SSN

(4) For each project, list the project name and the total hours per week (by all employees) spent on that project.

Ans : **SELECT** PNAME, SUM (HOURS)

**FROM** PROJECT, WORKS\_ON

**WHERE** PNUMBER=PNO

**GROUP** BYPNAME

(5) Retrieve the names of employees who work on every project of JPL.

(6) Retrieve the names of employees who do not work on any project.

Ans : **SELECT** LNAME, FNAME

**FROM** EMPLOYEE

**WHERE** NOT EXISTS

( **SELECT** PNUMBER

**FROM** PROJECT

**WHERE** NOT EXISTS

( **SELECT** \*

**FROM** WORKS\_ON

**WHERE** PNUMBER=PNO AND ESSN=SSN ) )

(7) For each department, retrieve the department name, and the average salary of employees working in that department.

Ans : **SELECT** DNAME, AVG (SALARY)

**FROM** DEPARTMENT, EMPLOYEE

**WHERE** DNUMBER=DNO

**GROUP** BYDNAME

(8) Retrieve the average working hour of all female employees.

And : **SELECT** AVG (SALARY)

**FROM** EMPLOYEE

**WHERE** SEX='F'

(9) Find the names and addresses of employees who work on at least one project located in Houston but whose department has no location in Houston.

Ans : **SELECT** LNAME, FNAME, ADDRESS

**FROM** EMPLOYEE

**WHERE** EXISTS

( **SELECT** \*

**FROM** WORKS\_ON, PROJECT

***WHERE*** SSN=ESSN AND PNO=PNUMBER AND LOCATION='Houston' )

AND NOT EXISTS

( **SELECT** \*

**FROM** DEPT\_LOCATIONS

**WHERE** DNO=DNUMBER AND DLOCATION='Houston' )

(10) List the last names of department managers who have no dependents.

Ams : **SELECT** LNAME, FNAME

**FROM** EMPLOYEE

**WHERE** EXISTS

( **SELECT** \*

**FROM** DEPARTMENT

**WHERE** SSN=MGRSSN )

AND NOT EXISTS

( **SELECT** \*

**FROM** DEPENDENT

**WHERE** SSN=ESSN )

(11) List the managers who are controlling more than 2 projects.

(12) Retrieve the names of the departments that only located in ‘Houston’.

(13) List the names of employees all of whose dependents were born before 1960.

(14) List the names of departments managed by the direct subordinator of the manager of ‘Headquarters’ department.

(15) Retrieve the average salary of male employees who work totally no less than 50 hours on projects.

(f) Retrieve the names of employees who do not work on any project

Ans : **SELECT** LNAME, FNAME

**FROM** EMPLOYEE

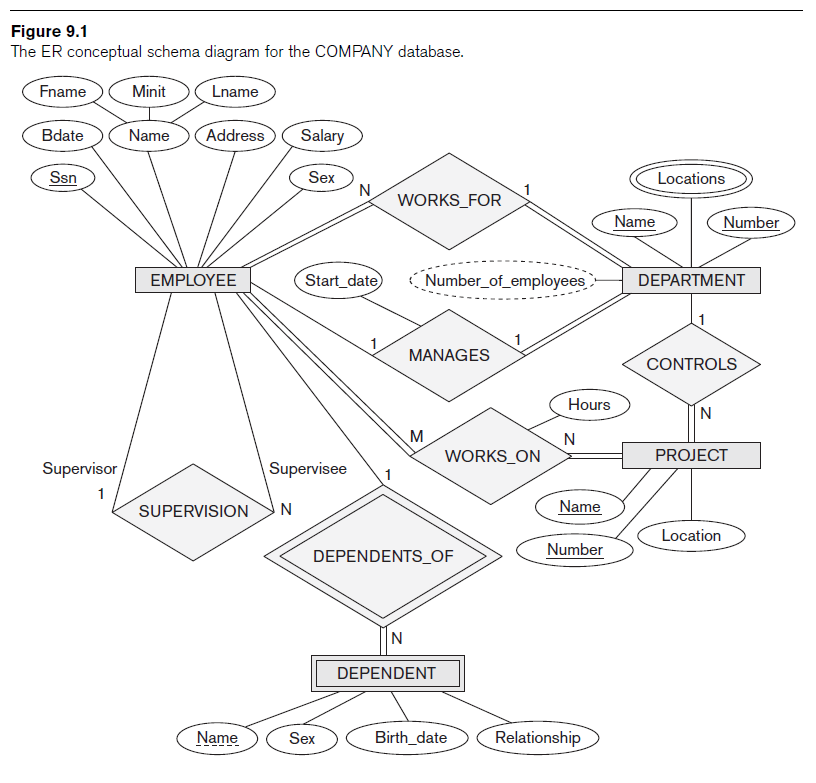
**WHERE** NOT EXISTS

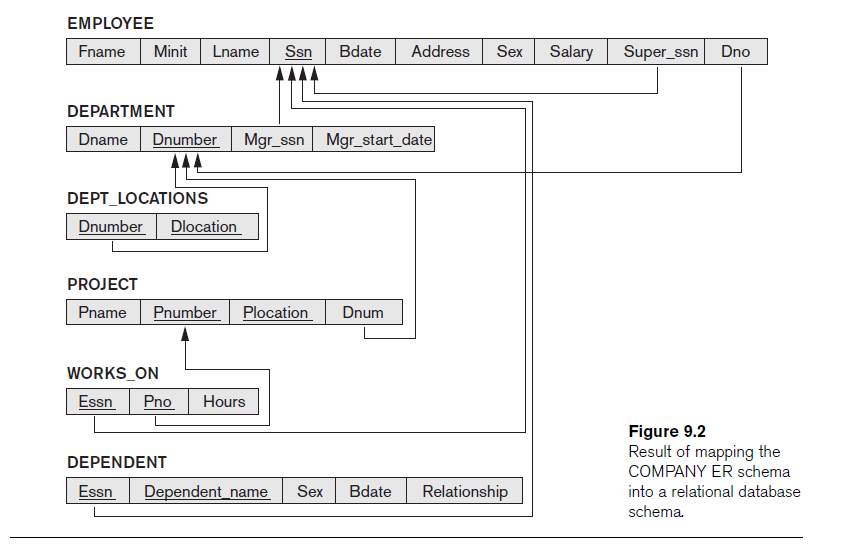
( **SELECT** \*

**FROM** WORKS\_ON

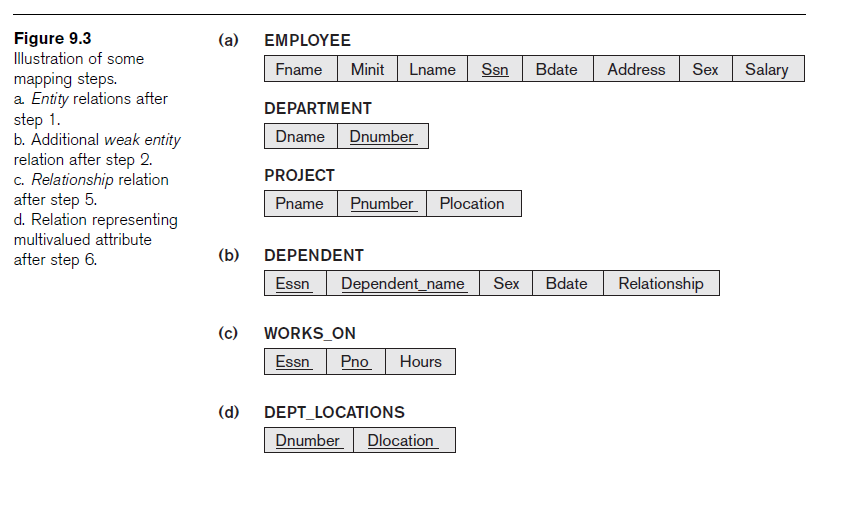
**WHERE** ESSN=SSN )

**Relational Database Design Using ER-to-Relational Mapping**

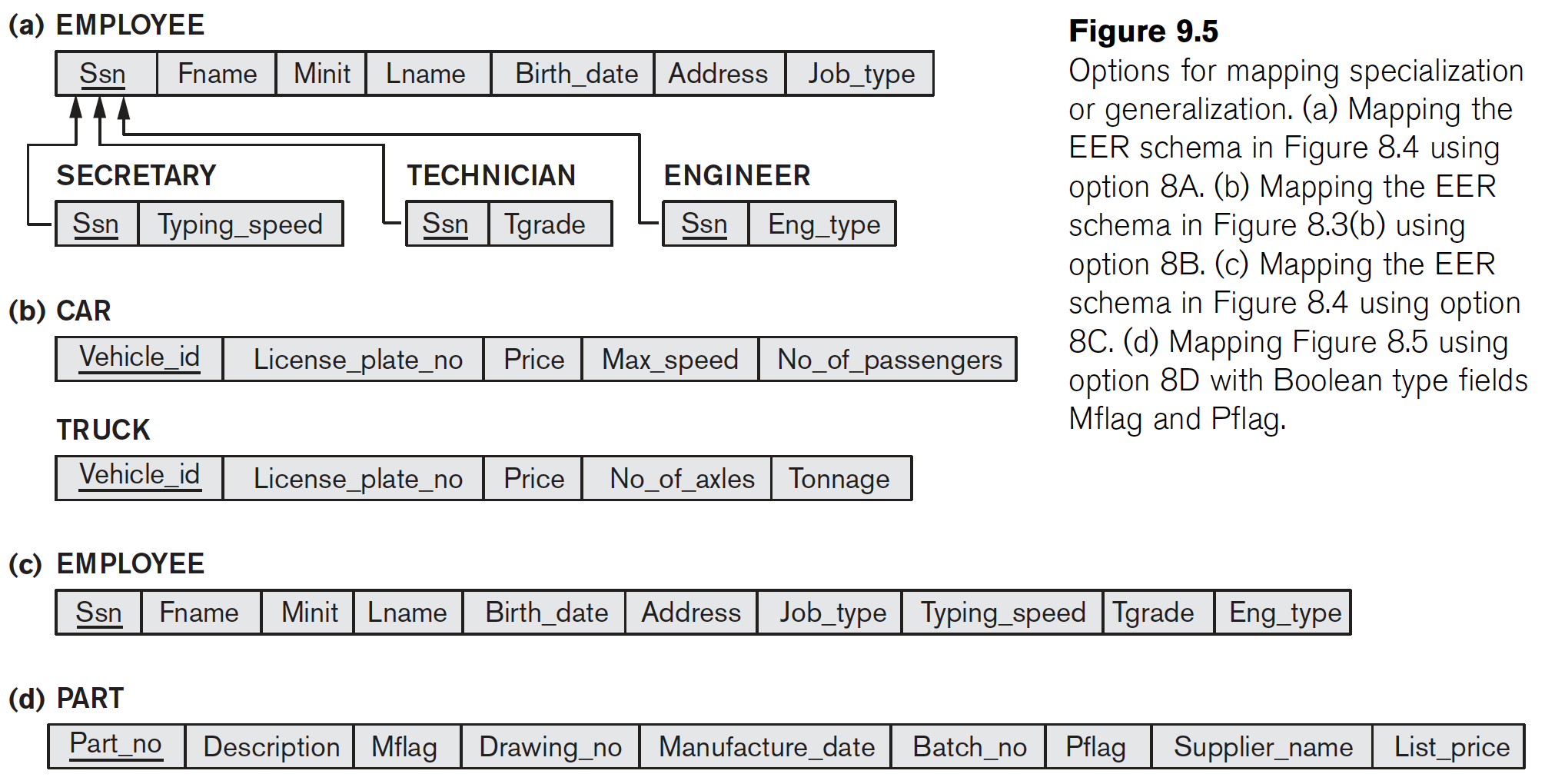
****

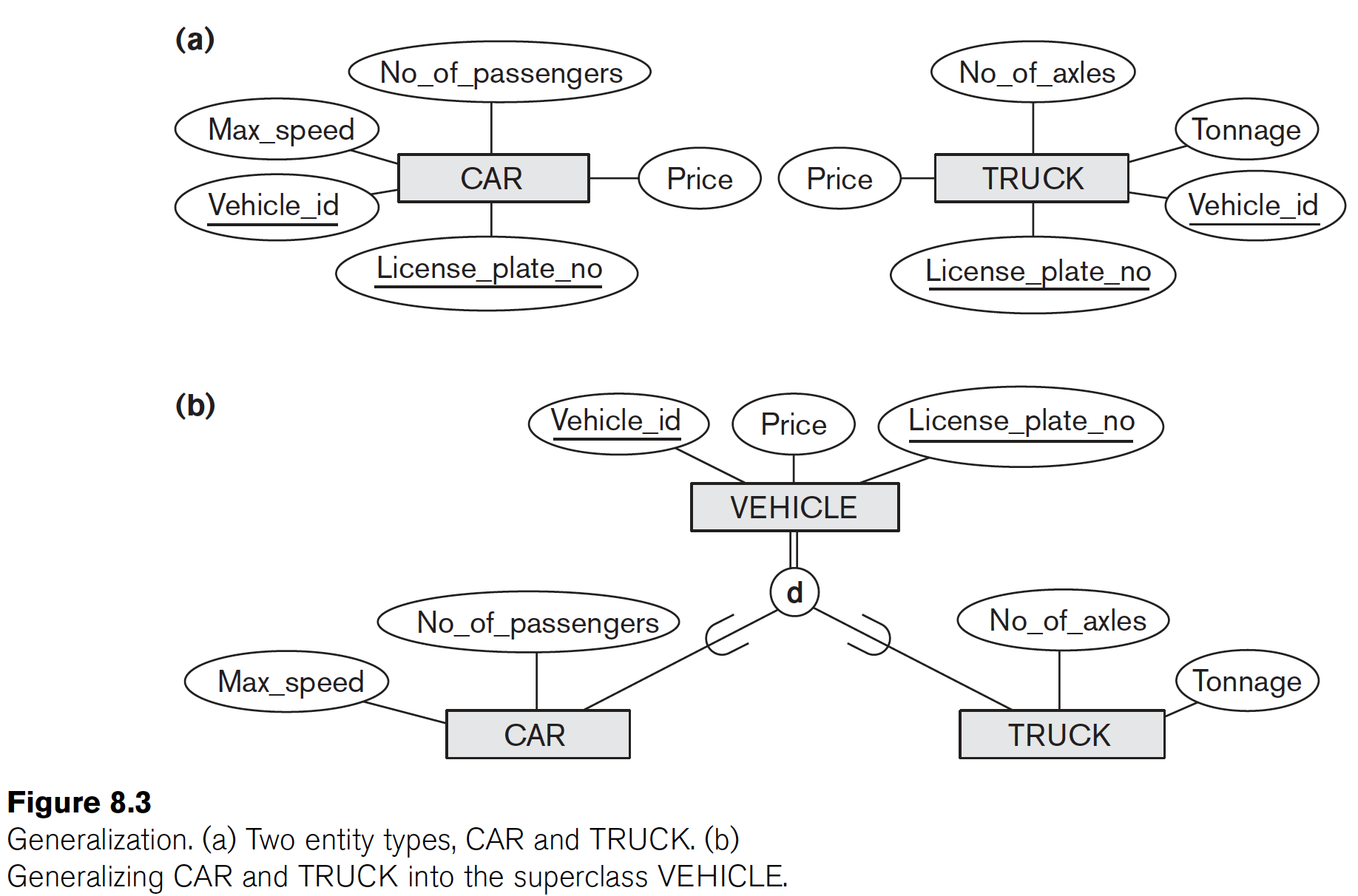
****

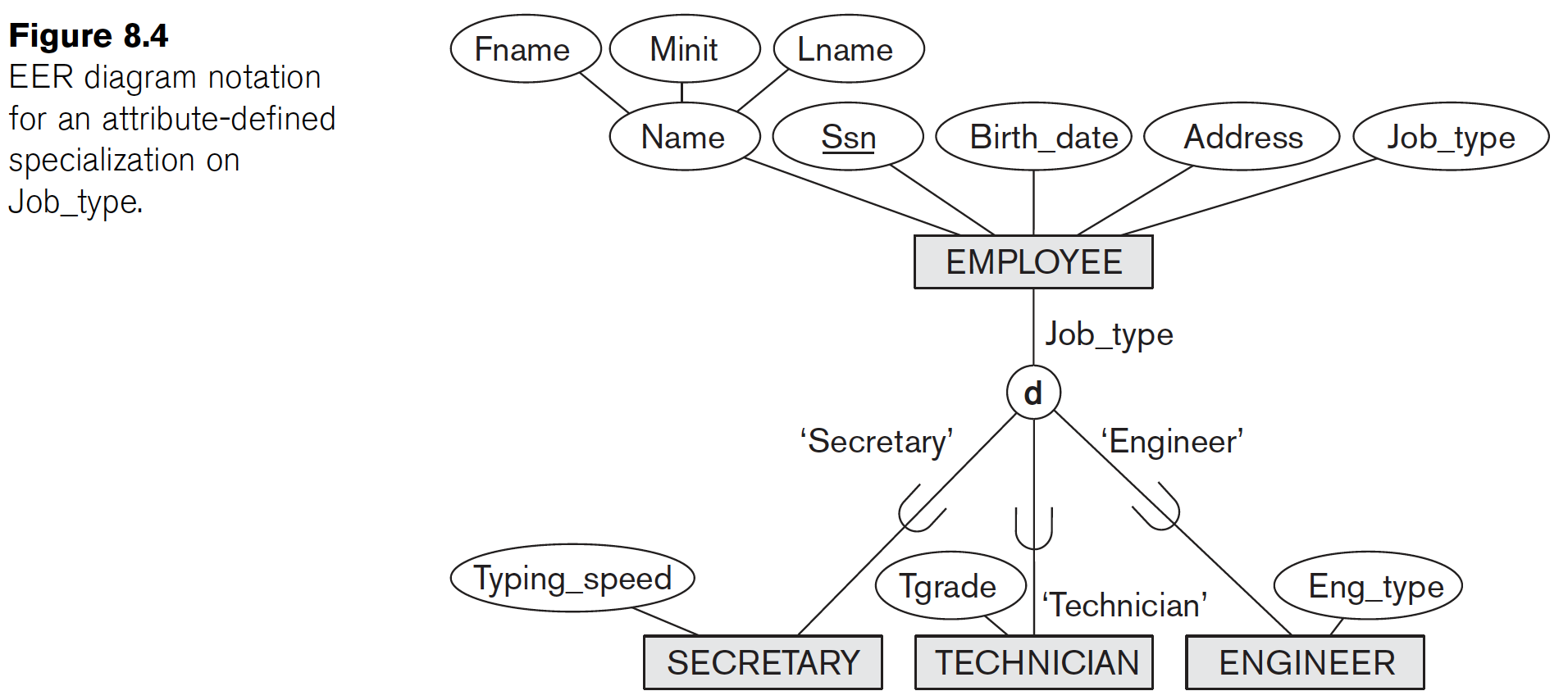
**ER-to-Relational Mapping Algorithm**

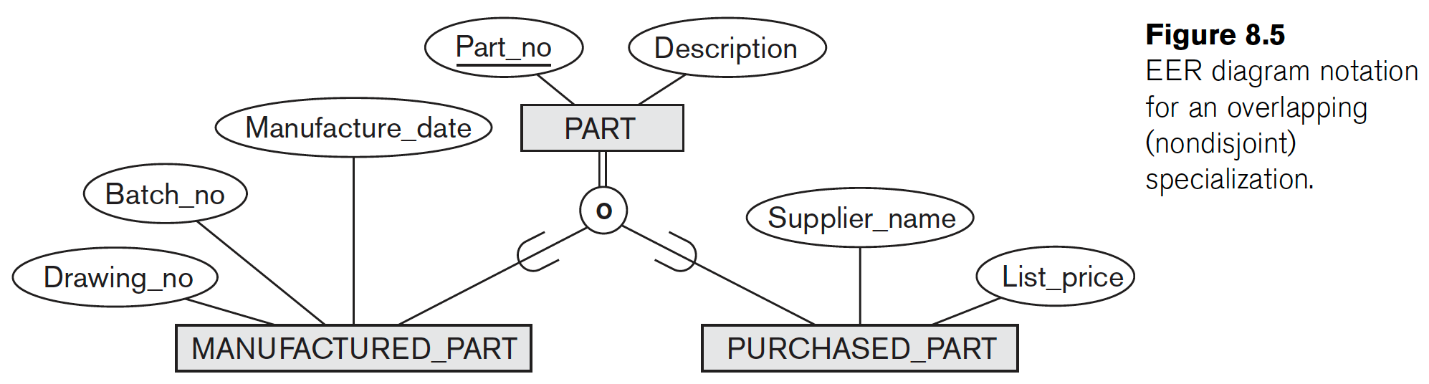
****

**EER Mapping**

****

****

****

****