Module 3:
Data Model - Relational Data Model and Relational Algebra

Unit 3.1 & 3.2:

Outline

- **Introduction**
- **Mapping EER to Relational Model**
- **Data Manipulation**
  - Relational Algebra
  - Relational Calculus
- **Data Integrity**
- **Advantages of the Relational Model**

Introduction

- Relational Model was proposed by E.F.Codd in 1970 based on concept of mathematical relation which looks like a table of value

- All data models represent a record which is collection of attributes where structure of record may be somewhat different

- Data model provides facilities for representing entities and their attributes as well as relationships

Why Relational Model?

- ER Model allows us to define the structure of enterprise data more easily
- Relational model allows powerful query languages to be designed that could not be employed with ER model
- ER model is used to **build conceptual view of database**
  Relational Model is used for **implementation of conceptual view**

Relational Model

- Provides specification of an abstract database management system
- Easy to understand
  - based on theoretical concepts like predicate calculus and theory of relations
- Codd defined relational model as consisting of following three components:

1. **Data Structure**: types of data structure used for building database
2. **Data Manipulation**: operators that are used to retrieve, derive or modify data stored in data structures
3. **Data Integrity**: rules that explicitly/implicitly define consistent database state

1. Data Structure

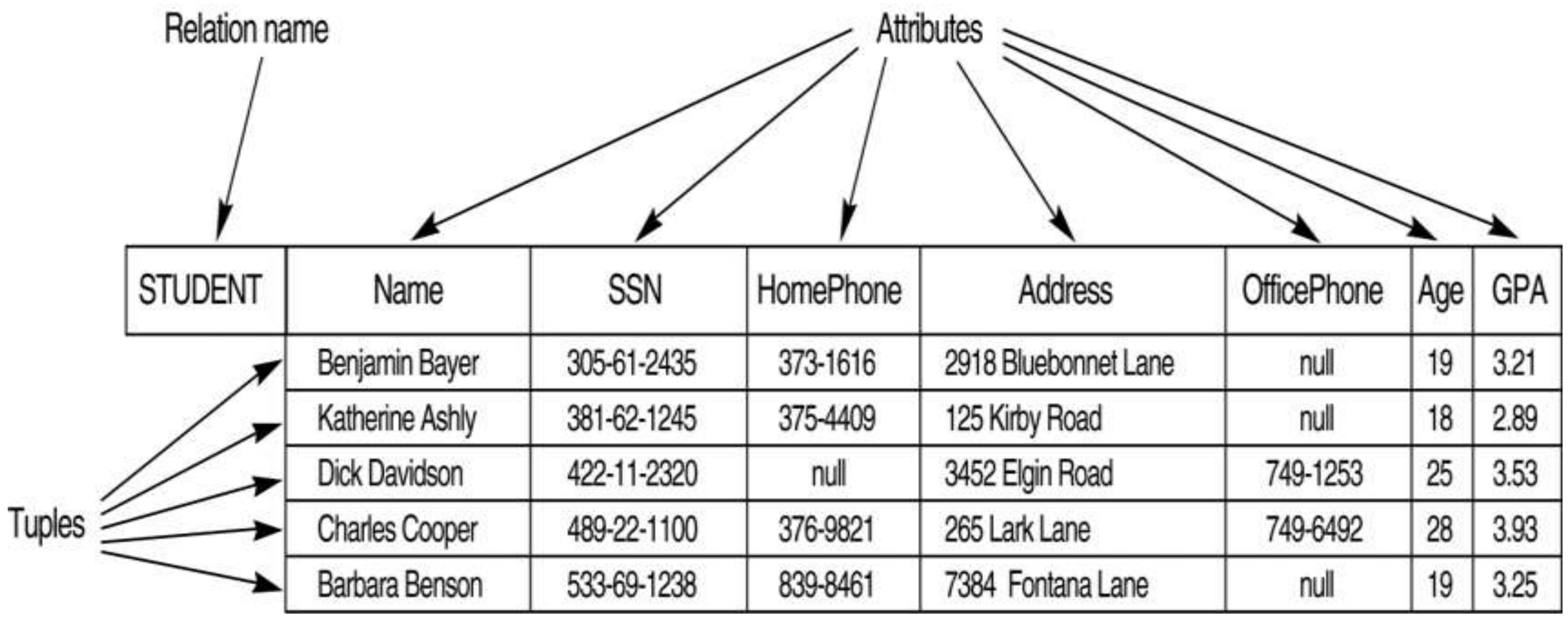
- Information about all entities and their attributes as well as relationships are presented to users as tables(relations)
- Database is collection of relations
- Each **row of each table** consist of entity occurrence or relationship occurrence
- Each **column** refer to an attribute
- In relational model, it is assumed that no ordering of rows and columns is defined

1. Data Structure(contd..)

- **Relational Terminology:**

1. **Relation**: simple table defined as set of rows
2. **Tuple**: each row in a relation
3. **Attribute:** column in a relation
4. **Degree of relation:** number of columns in the relation
5. **Cardinality of relation**: number of rows in the relation
6. **Domain**: set of atomic values that each element in column is permitted to take.
    - Domain maybe given unique names
    - Same domain can be used by number of different columns

1. Data Structure example (contd..)



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	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
	Charles Cooper	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

1. Data Structure(contd..)

- A **Relation** may be defined in multiple ways.
- The **Schema** of a Relation: *R* (A1, A2, ...An)

  Relation schema *R* is defined over **attributes** A1, A2, ...An

For Example -

CUSTOMER (Cust-id, Cust-name, Address, Phone#)

Here, CUSTOMER is a relation defined over the four attributes Cust-id, Cust-name, Address, Phone#, each of which has a **domain** or a set of valid values. For example, the domain of Cust-id is 6 digit numbers.

1. Data Structure(contd..)

- A **tuple** is an ordered set of values
- Each value is derived from an appropriate domain.
- Each row in the CUSTOMER table may be referred to as a tuple in the table and would consist of four values.
- **<632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">**
  is a tuple belonging to the CUSTOMER relation.
- A relation may be regarded as a ***set of tuples*** (rows).

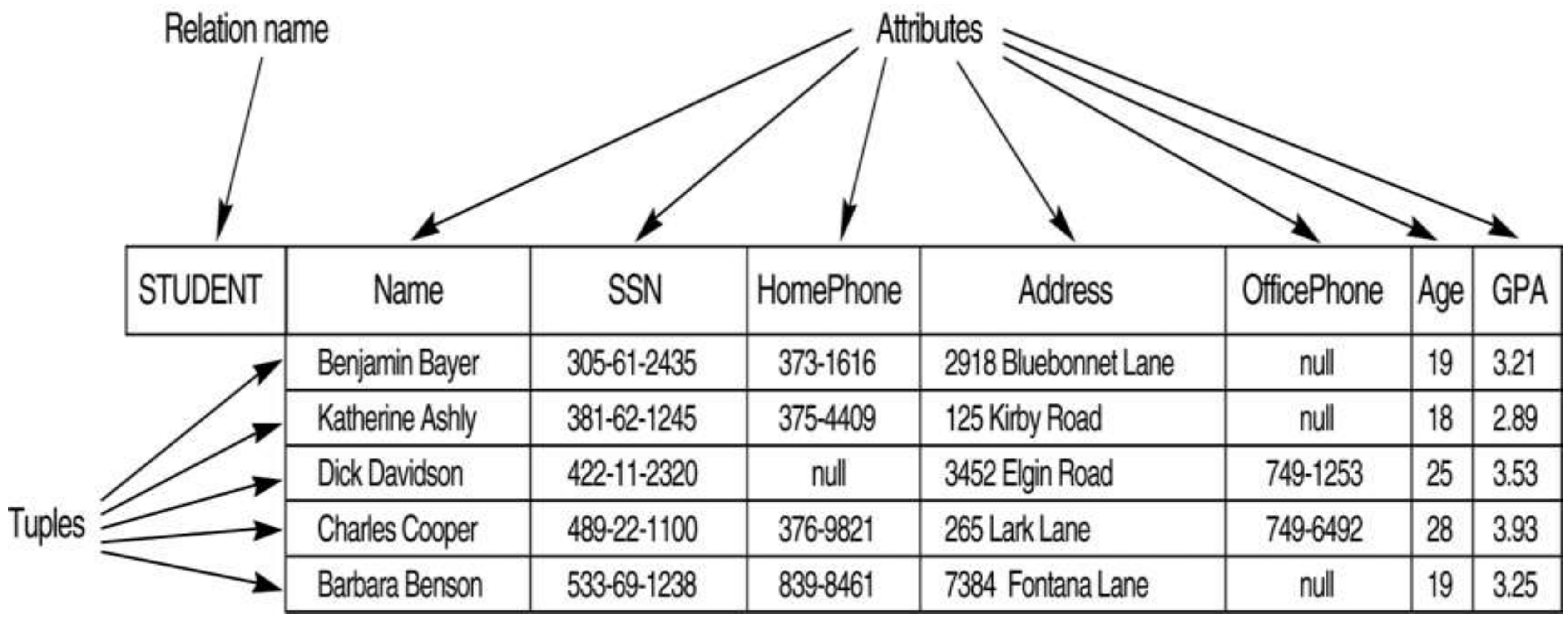
1. Data Structure(contd..)

- A **domain** has a logical definition: e.g., “USA_phone_numbers” are the set of 11 digit phone numbers valid in the U.S. which may have a format: (ddd)-ddd-dddd where each d is a decimal digit.
- E.g., Dates have various formats such as monthname, date, year or yyyy-mm-dd, or dd mm,yyyy etc.
- An attribute designates the **role** played by the domain.
- E.g., the domain Date may be used to define attributes “Invoice-date” and “Payment-date”.

1. Data Structure Summary(contd..)

Informal Terms	Formal Terms
Table	Relation
Column	Attribute/Domain
Row	Tuple
Values in a column	Domain
Table Definition	Schema of a Relation

1. Data Structure Summary(contd..)



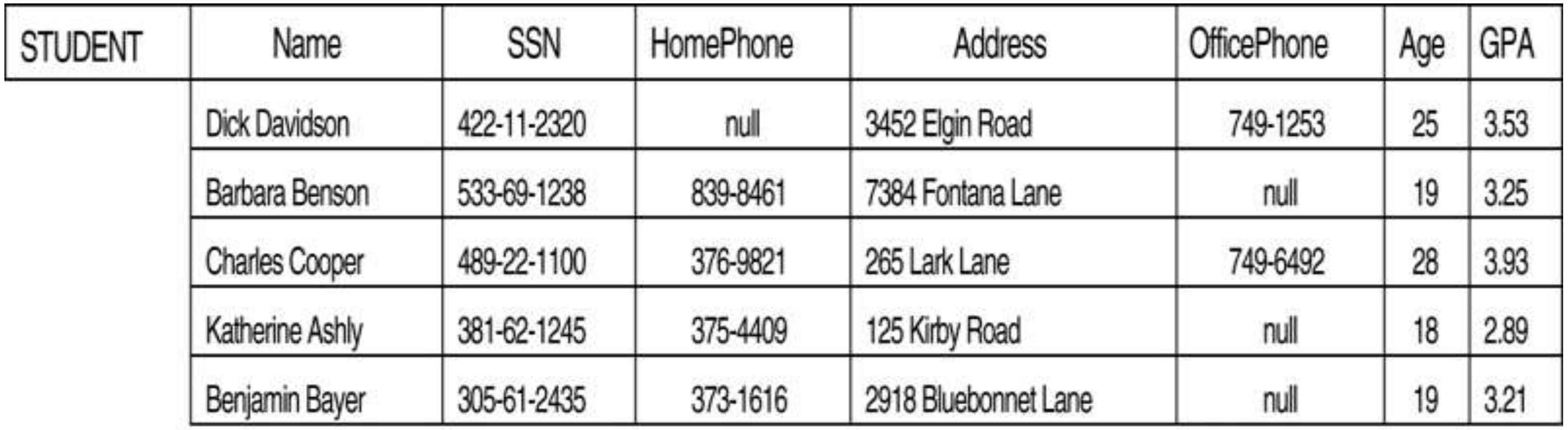
STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25

1. Data Structure(contd..)

- **Properties of a relation in relational model:**

1. Each relation contains only one record type
2. Each relation has a fixed number of columns(attributes) that are explicitly named. Each attribute name within a relation is unique
3. No two rows in a relation are the same
4. Each item or element in relation is atomic, every attribute has only one value that cannot be decomposed into smaller components
5. Rows have no ordering associated with them
6. Columns have no ordering associated with them
7. null value is used to represent values that are unknown or inapplicable to certain tuples

1. Data Structure(contd..)



STUDENT	Name	SSN	HomePhone	Address	OfficePhone	Age	GPA
	Dick Davidson	422-11-2320	null	3452 Elgin Road	749-1253	25	3.53
	Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	null	19	3.25
	Charles Cooper	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
	Katherine Ashly	381-62-1245	375-4409	125 Kirby Road	null	18	2.89
	Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	null	19	3.21

Properties of a relation in relational model

1. Data Structure(contd..)

Candidate Key:

- An attribute(or set of attributes) is called candidate key of relation if it satisfies following properties:
    a. An attribute(or set of attributes) uniquely identifies each tuple in relation…..(uniqueness property)
    b. If key is set of attributes then no subset of these attributes has property(a)…..(minimal property)
    c. more than one **candidate key**
    d. **Ex: Employee_Id, SSN**

Primary Key:

- One(and only one) of the candidate keys is arbitrarily chosen as the primary key of the table
- Primary key therefore has properties of uniqueness and minimality
- **Ex: Employee_Id**

Unit 3.2

- **Mapping EER Model to Relational Model**
- **ER-to-Relational Mapping Algorithm**

Step 1: Mapping of Regular Entity Types

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relation Types

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

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

Step 6: Mapping of Multivalued attributes.

Step 7: Mapping of N-ary Relationship Types.

- **Mapping EER Model Constructs to Relations**

Step 8: Options for Mapping Specialization or Generalization.

Step 9: Mapping of Union Types (Categories).

FIGURE 7.1
The ER conceptual schema diagram for the COMPANY database.

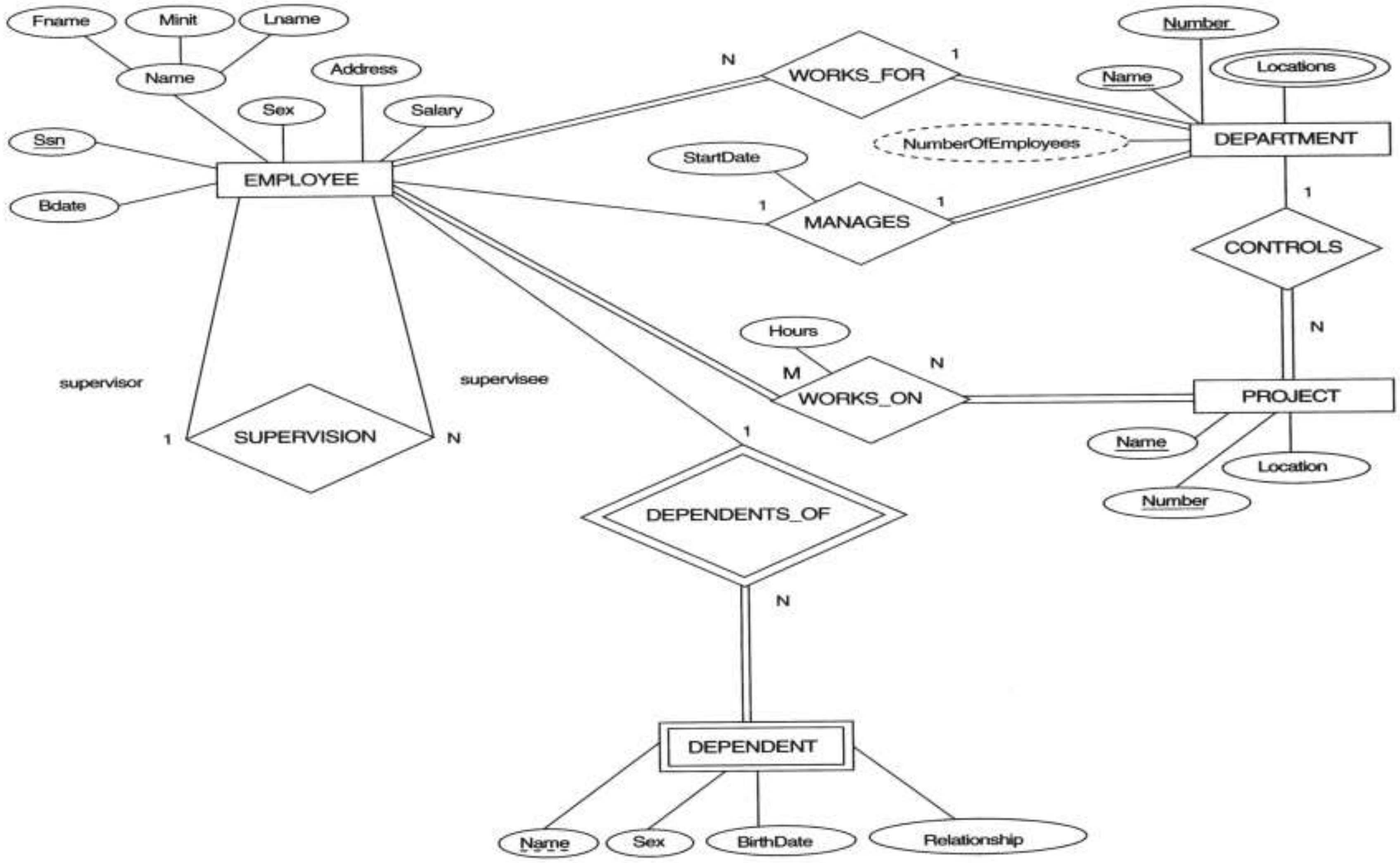


Figure 7.5 Schema diagram for the COMPANY relational database schema; the primary keys are underlined.

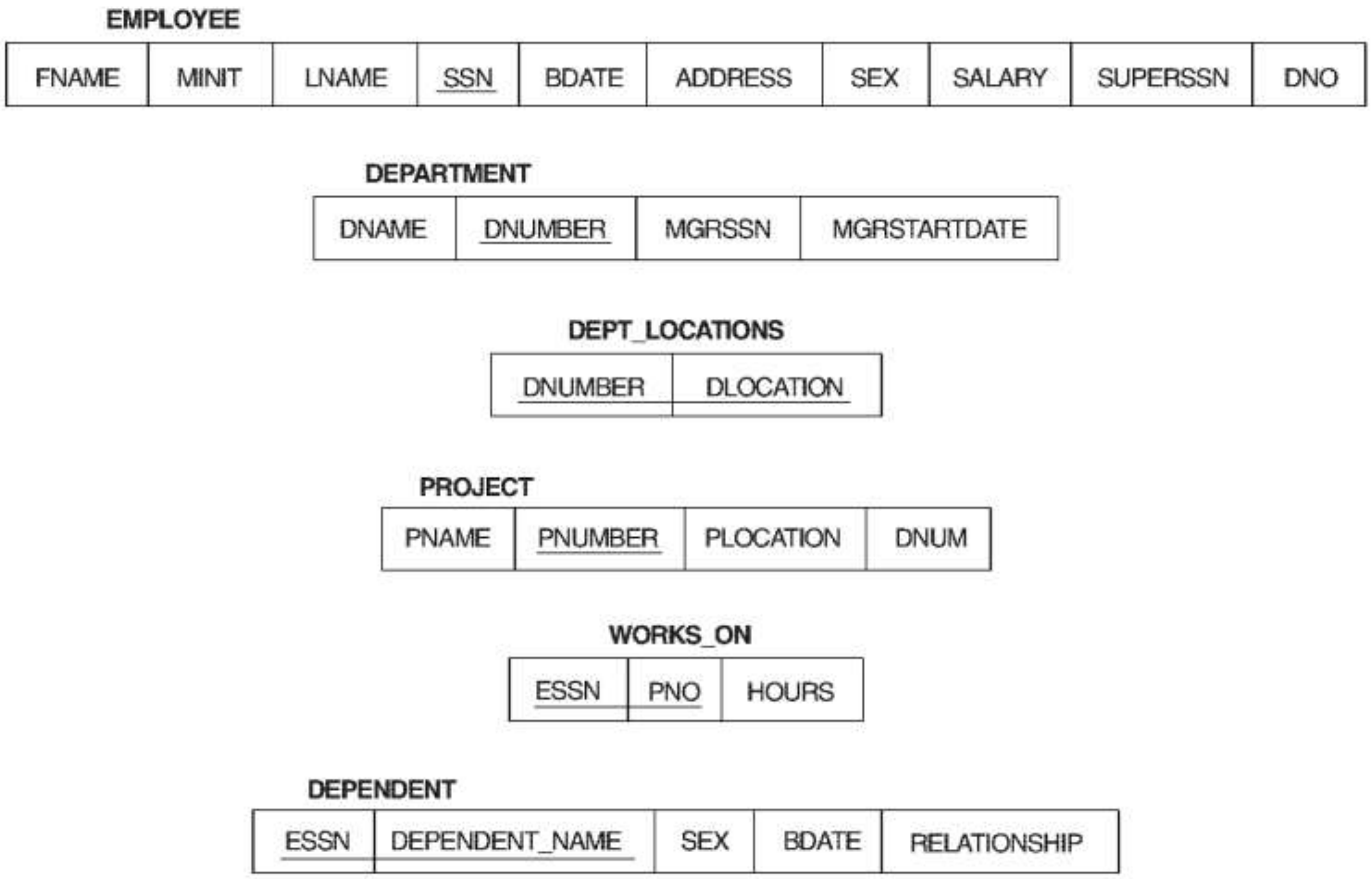
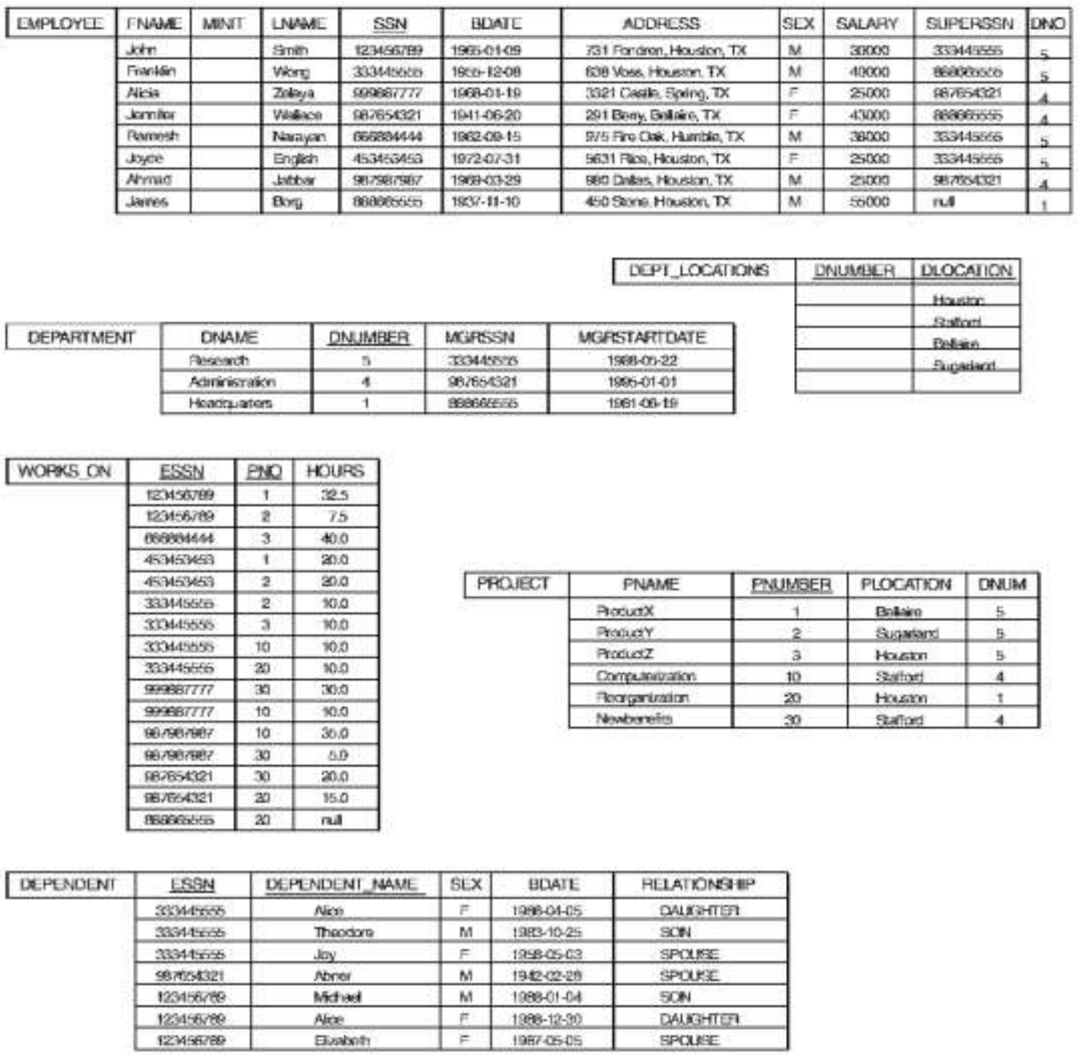


Figure 7.6 One possible relational database state corresponding to the COMPANY schema.



EMPLOYEE	FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
	John		Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
	Franklin		Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia		Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer		Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh		Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce		English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad		Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James		Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPT_LOCATIONS	DNUMBER	DLOCATION
		Houston
		Stafford
		Bellaire
		Sugarland

DEPARTMENT	DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
	Research	5	333445555	1988-05-22
	Administration	4	987654321	1995-01-01
	Headquarters	1	888665555	1981-06-19

WORKS_ON	ESSN	PNO	HOURS
	123456789	1	32.5
	123456789	2	7.5
	666884444	3	40.0
	453453453	1	20.0
	453453453	2	20.0
	333445555	2	10.0
	333445555	3	10.0
	333445555	10	10.0
	333445555	20	10.0
	999887777	30	30.0
	999887777	10	10.0
	987987987	10	35.0
	987987987	30	5.0
	987654321	30	20.0
	987654321	20	15.0
	888665555	20	null

PROJECT	PNAME	PNUMBER	PLOCATION	DNUM
	ProductX	1	Bellaire	5
	ProductY	2	Sugarland	5
	ProductZ	3	Houston	5
	Computerization	10	Stafford	4
	Reorganization	20	Houston	1
	Newbenefits	30	Stafford	4

DEPENDENT	ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
	333445555	Alice	F	1986-04-05	DAUGHTER
	333445555	Theodore	M	1983-10-25	SON
	333445555	Joy	F	1958-05-03	SPOUSE
	987654321	Abner	M	1942-02-28	SPOUSE
	123456789	Michael	M	1988-01-04	SON
	123456789	Alice	F	1988-12-30	DAUGHTER
	123456789	Elizabeth	F	1967-05-05	SPOUSE

ER-to-Relational Mapping Algorithm

- **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.
  - Include only simple composite attributes of a composite attribute
  - 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:
  - We create the relations EMPLOYEE, DEPARTMENT, and PROJECT in the relational schema corresponding to the regular entities in the ER diagram.
  - SSN, DNUMBER, and PNUMBER are the primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT as shown.

FIGURE 7.1
The ER conceptual schema diagram for the COMPANY database.

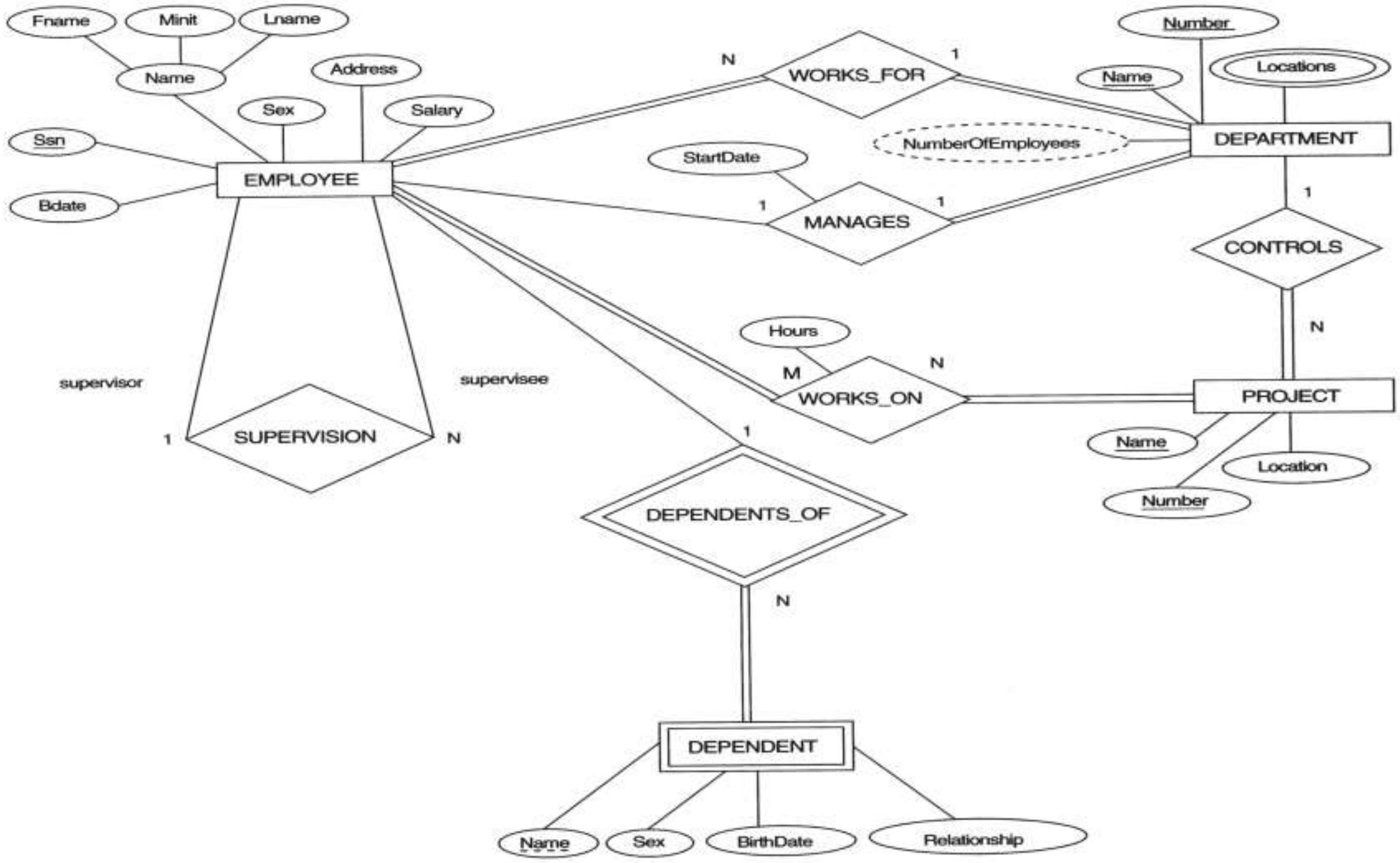
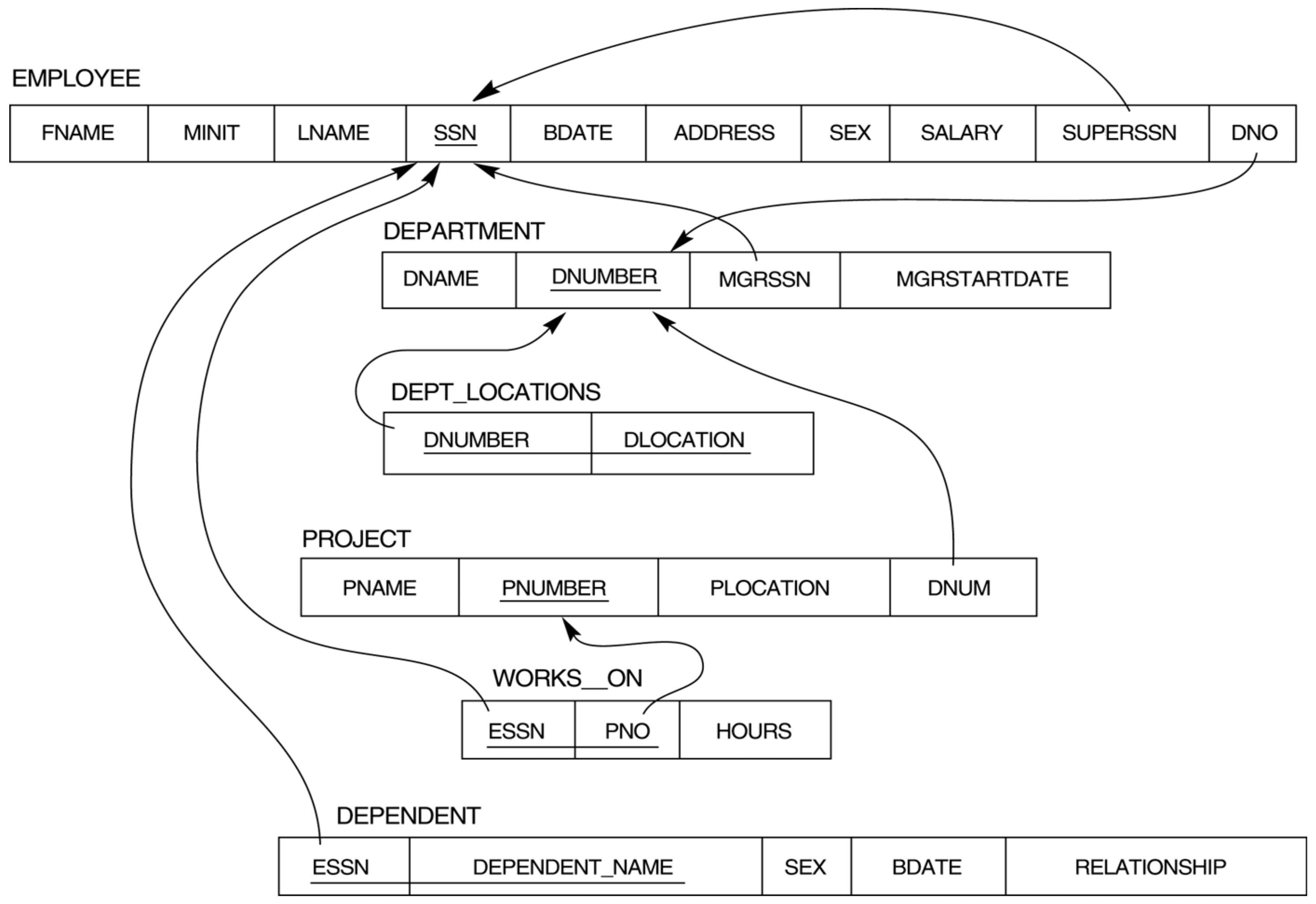


FIGURE 7.2
Result of mapping the COMPANY ER schema into a relational schema.



ER-to-Relational Mapping Algorithm (contd..)

- **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).

  The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

ER-to-Relational Mapping Algorithm (contd..)

- **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) **Foreign Key approach:** Choose one of the relations-S, say-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.

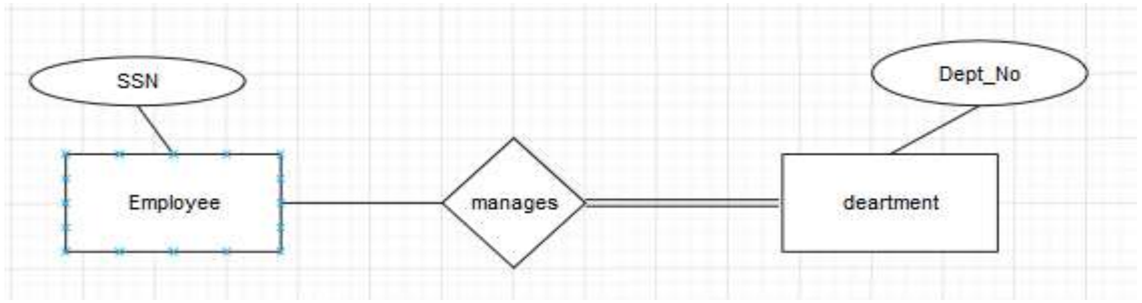
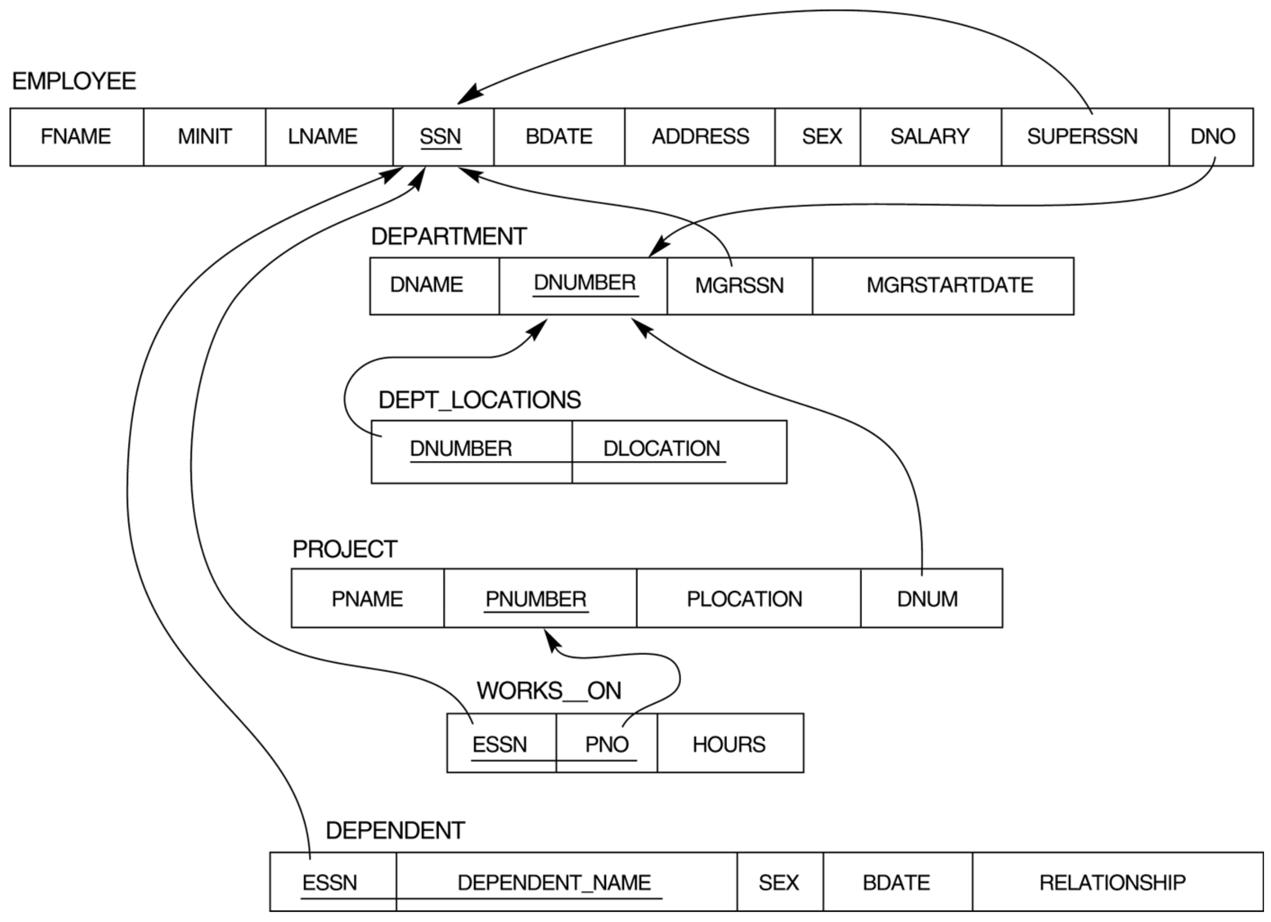


FIGURE 7.2
Result of mapping the COMPANY ER schema into a relational schema.



Step 3: Mapping of Binary 1:1 Relation Types(contd..)

(2) **Merged relation option:**

- 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.*

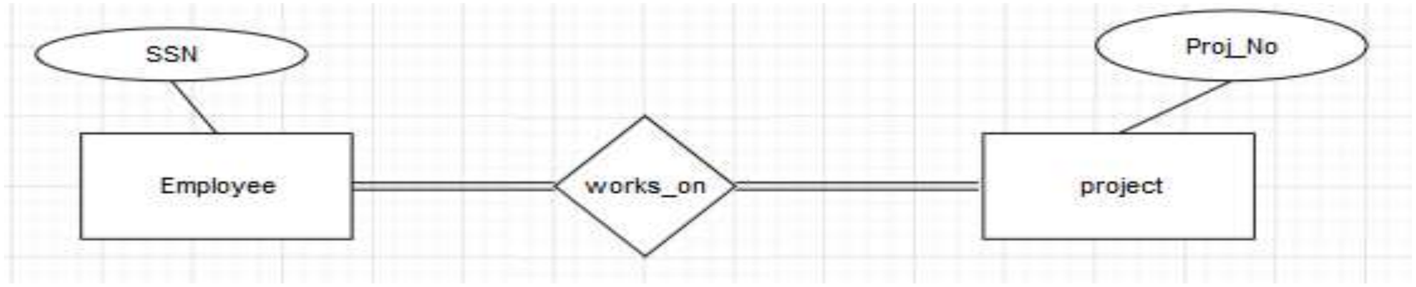


FIGURE 7.1

The ER conceptual schema diagram for the COMPANY database.

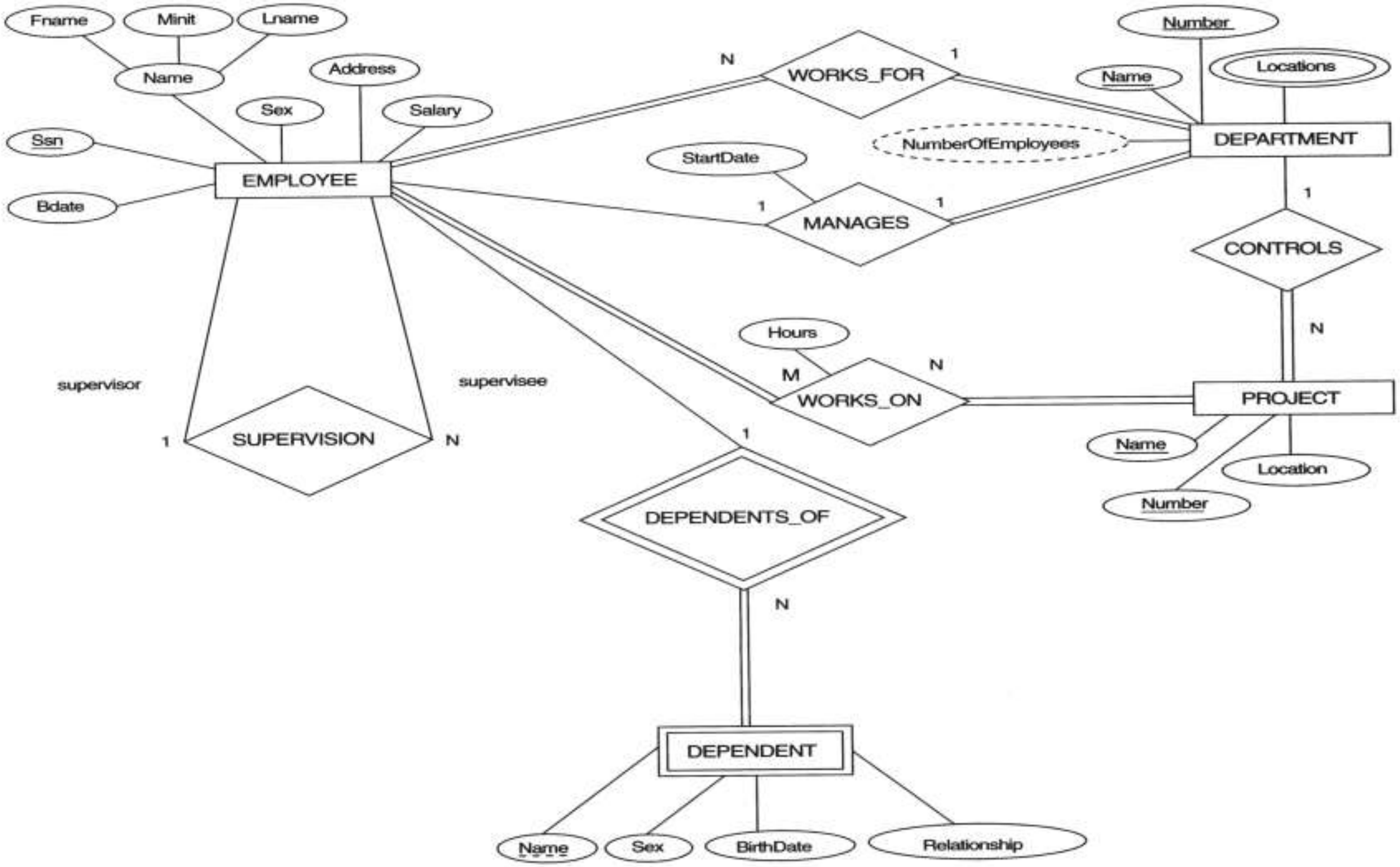
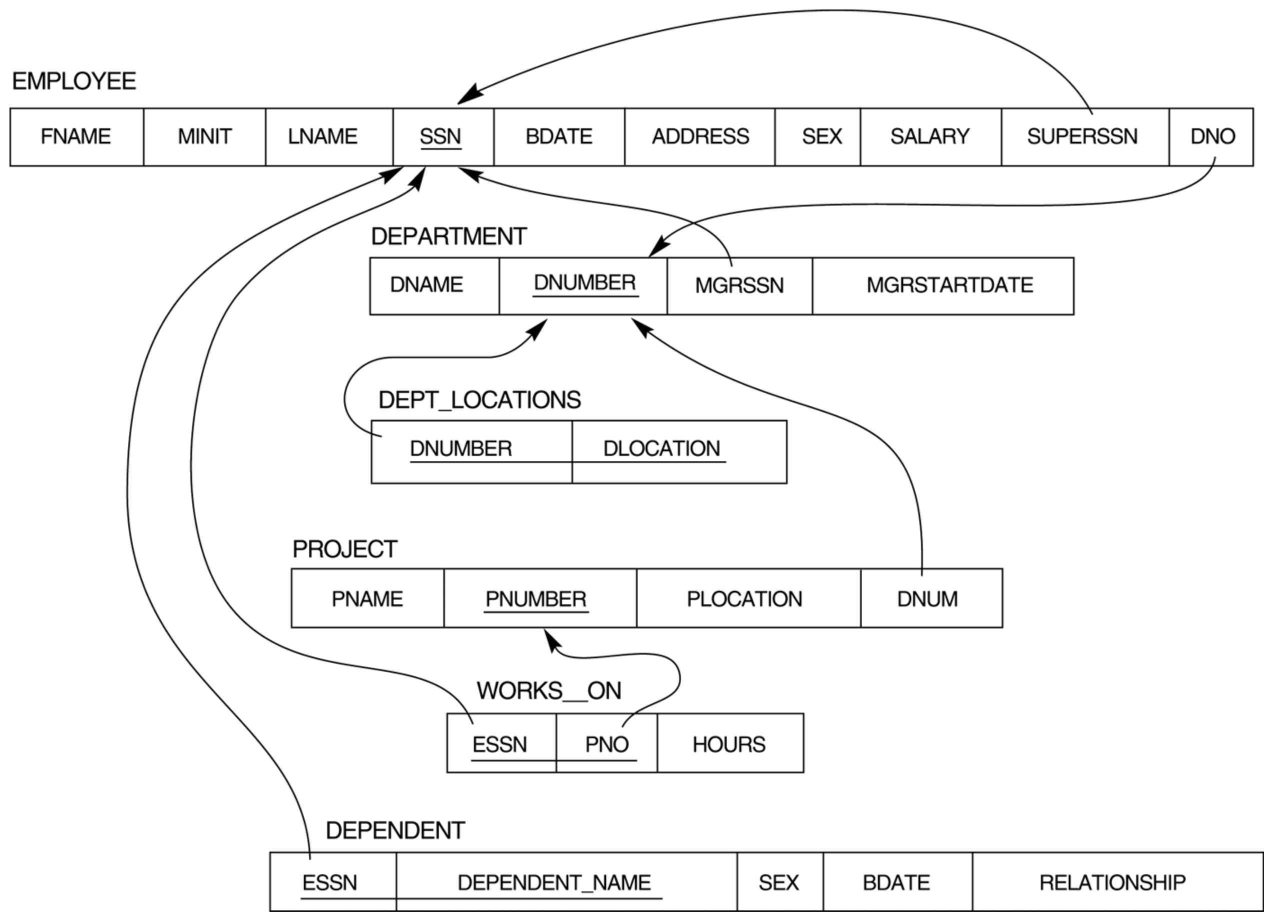


FIGURE 7.2
Result of mapping the COMPANY ER schema into a relational schema.



ER-to-Relational Mapping Algorithm (cont)

- **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: 1:N relationship types **WORKS_FOR, CONTROLS, and SUPERVISION** in the figure. For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO.

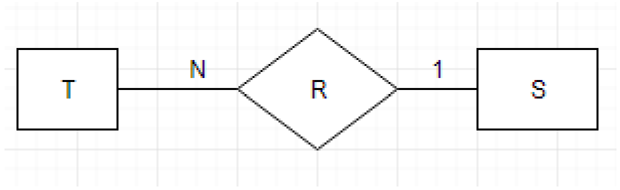


FIGURE 7.1
The ER conceptual schema diagram for the COMPANY database.

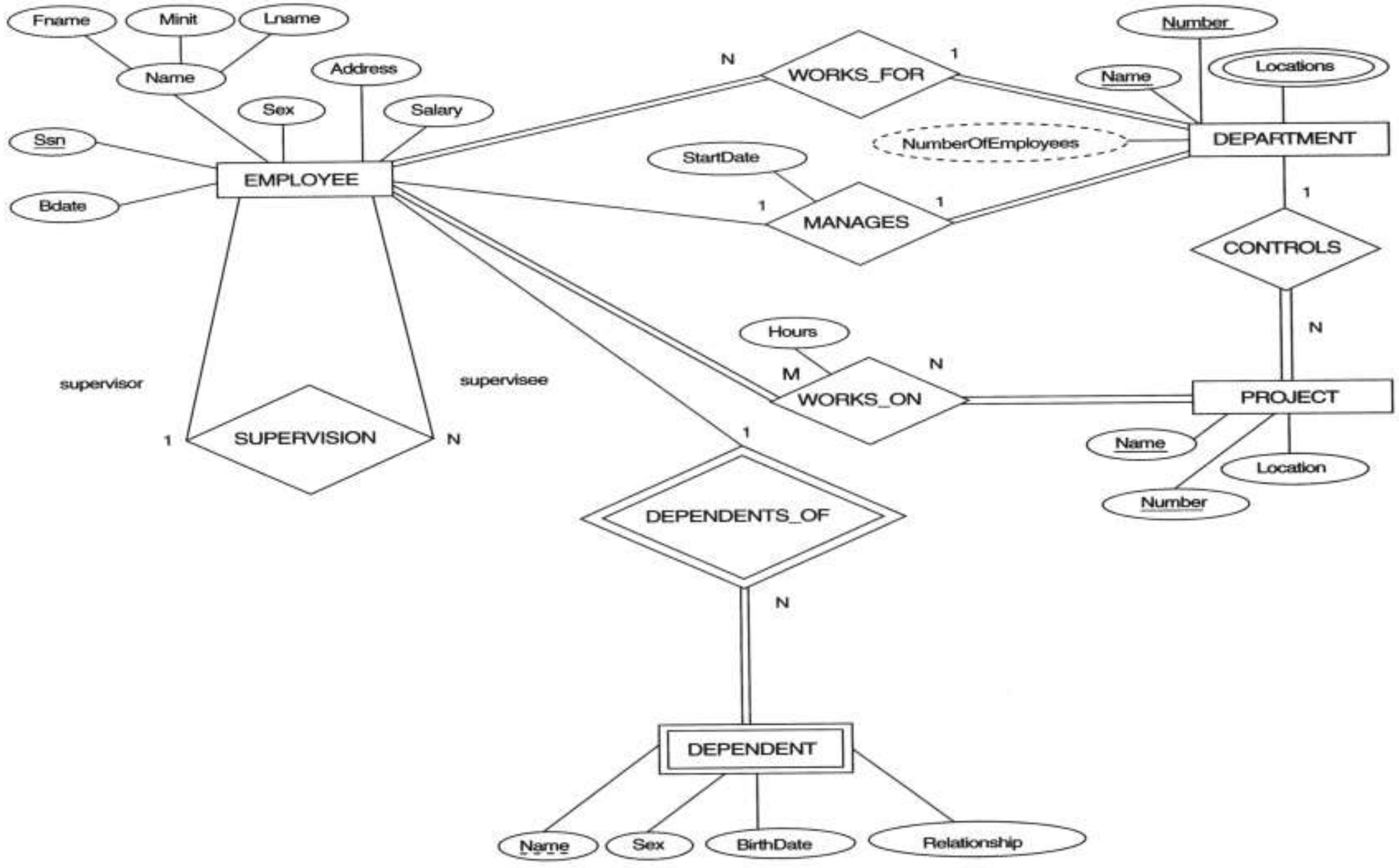
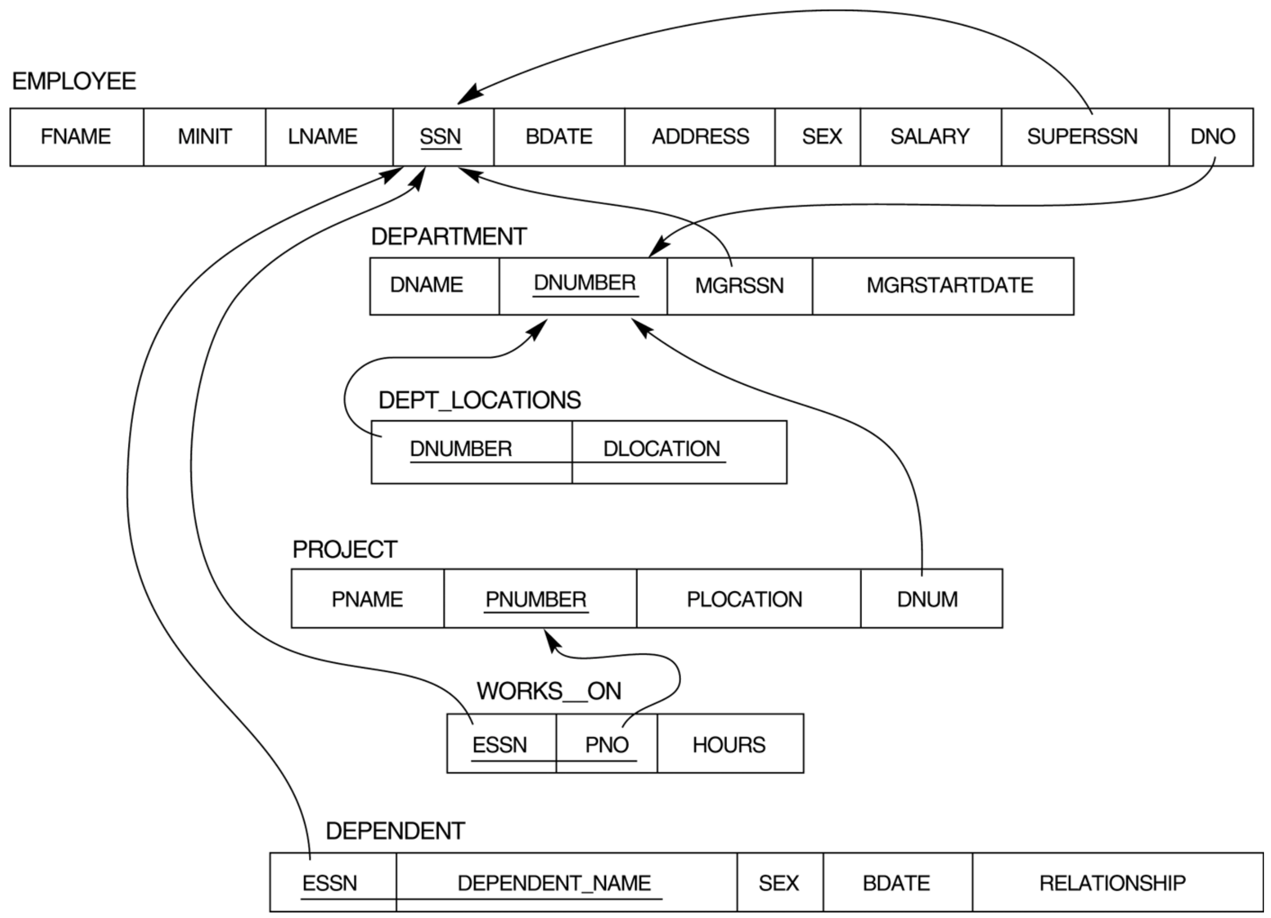


FIGURE 7.2
Result of mapping the COMPANY ER schema into a relational schema.



ER-to-Relational Mapping Algorithm (cont)

- **Step 5: Mapping of Binary M:N Relationship Types. (Cross Referenced Approach)**
  - 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.
  - 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}.

FIGURE 7.1
The ER conceptual schema diagram for the COMPANY database.

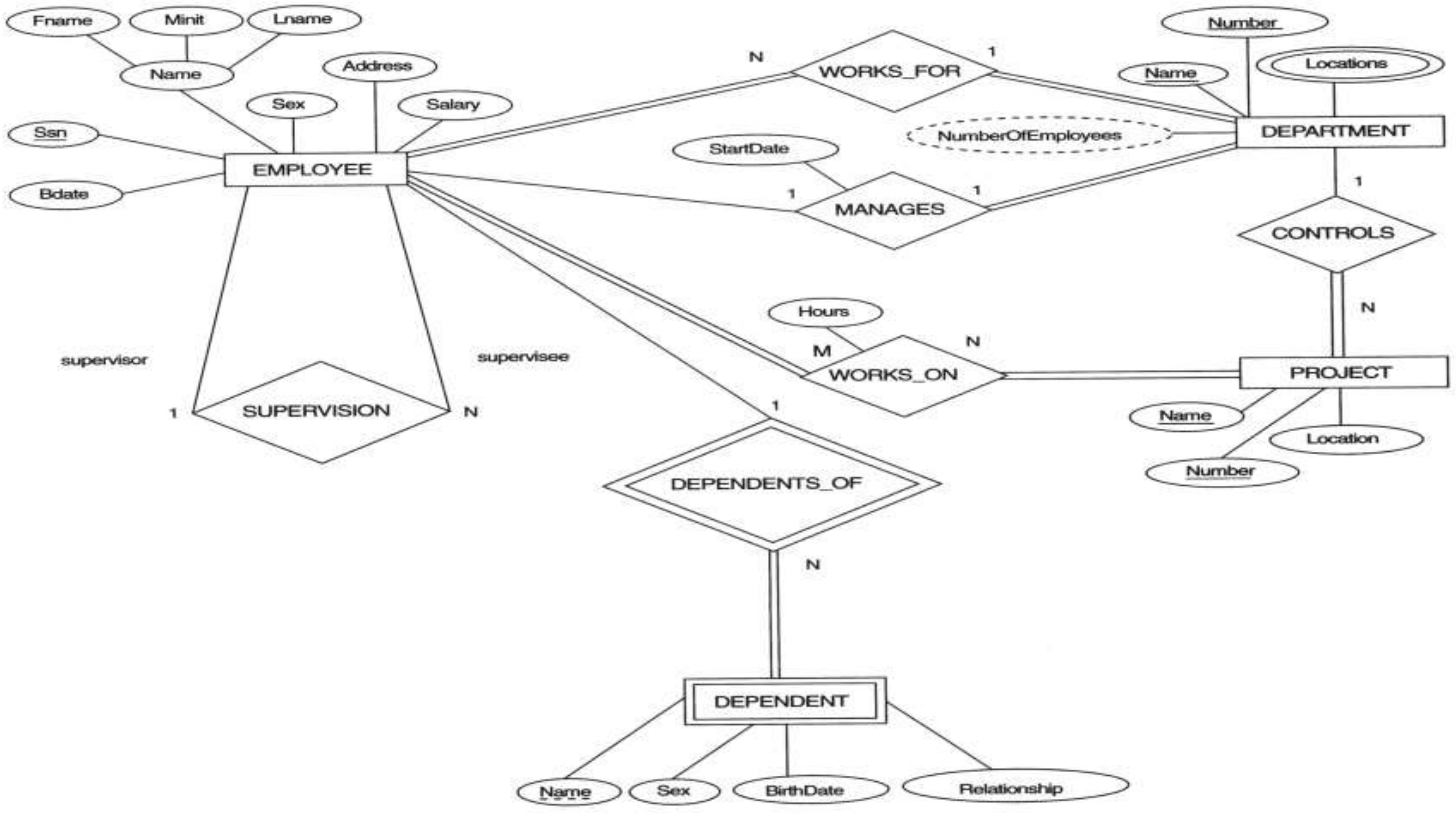
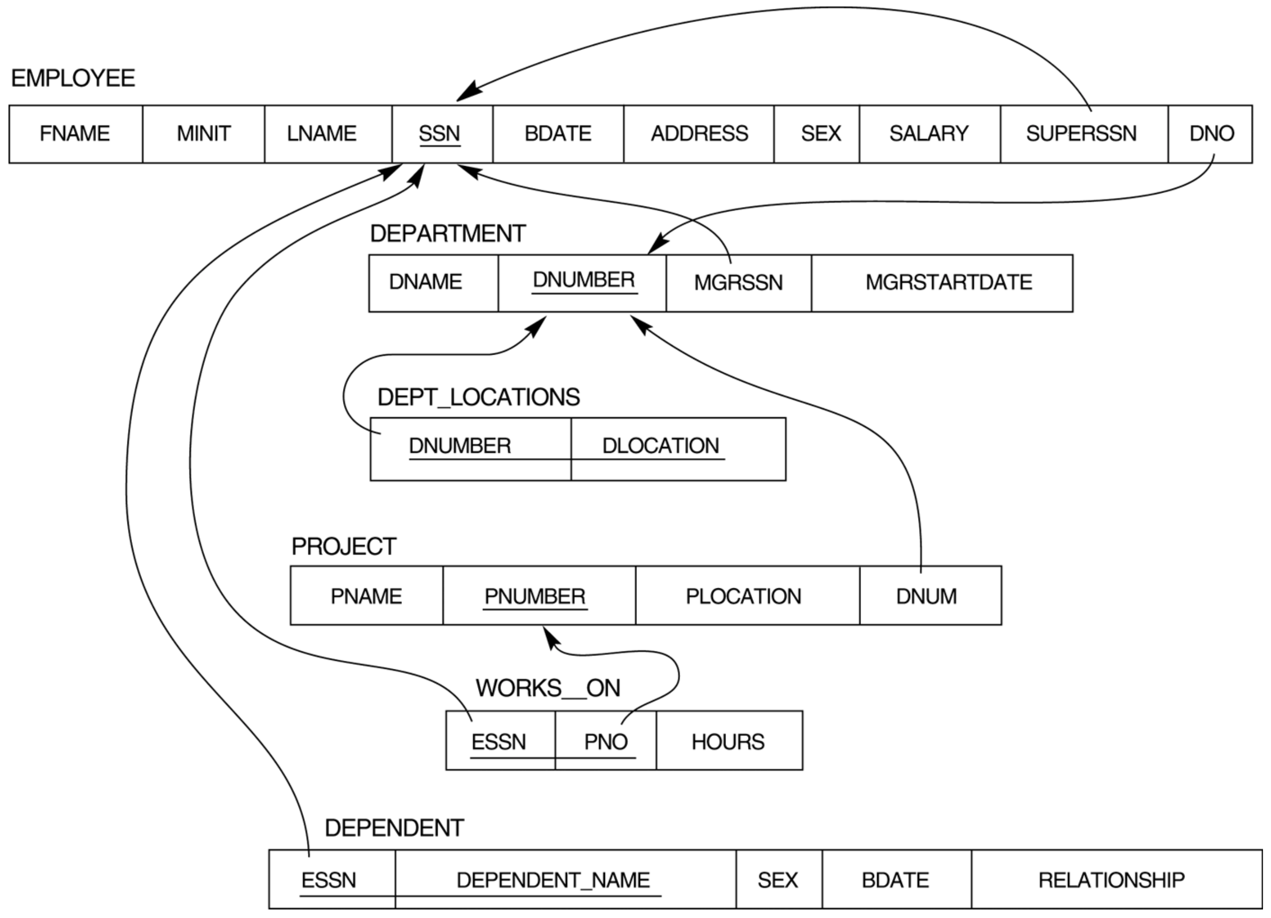


FIGURE 7.2
Result of mapping the COMPANY ER schema into a relational schema.



ER-to-Relational Mapping Algorithm (cont)

- **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}.

FIGURE 7.1
The ER conceptual schema diagram for the COMPANY database.

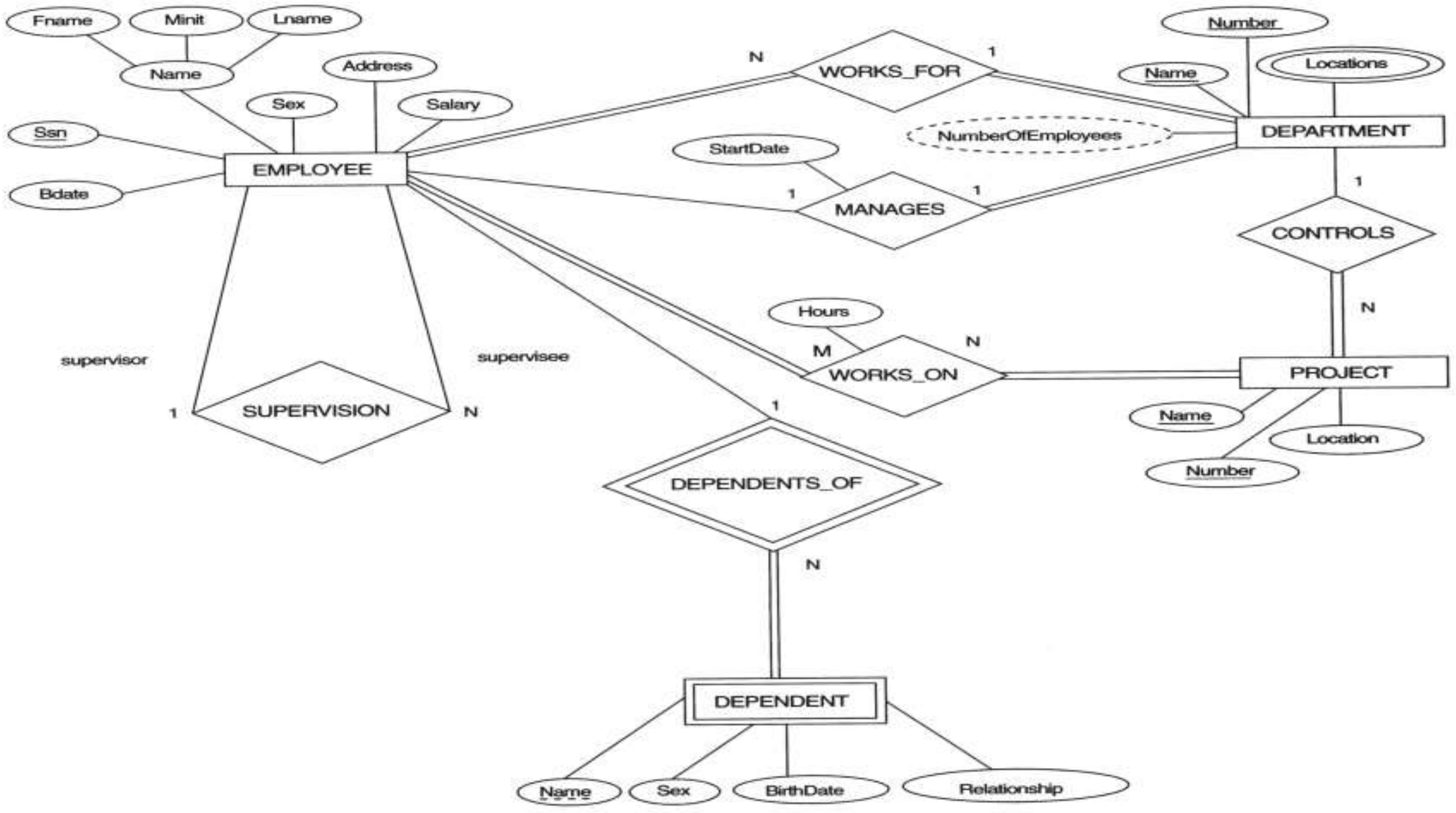
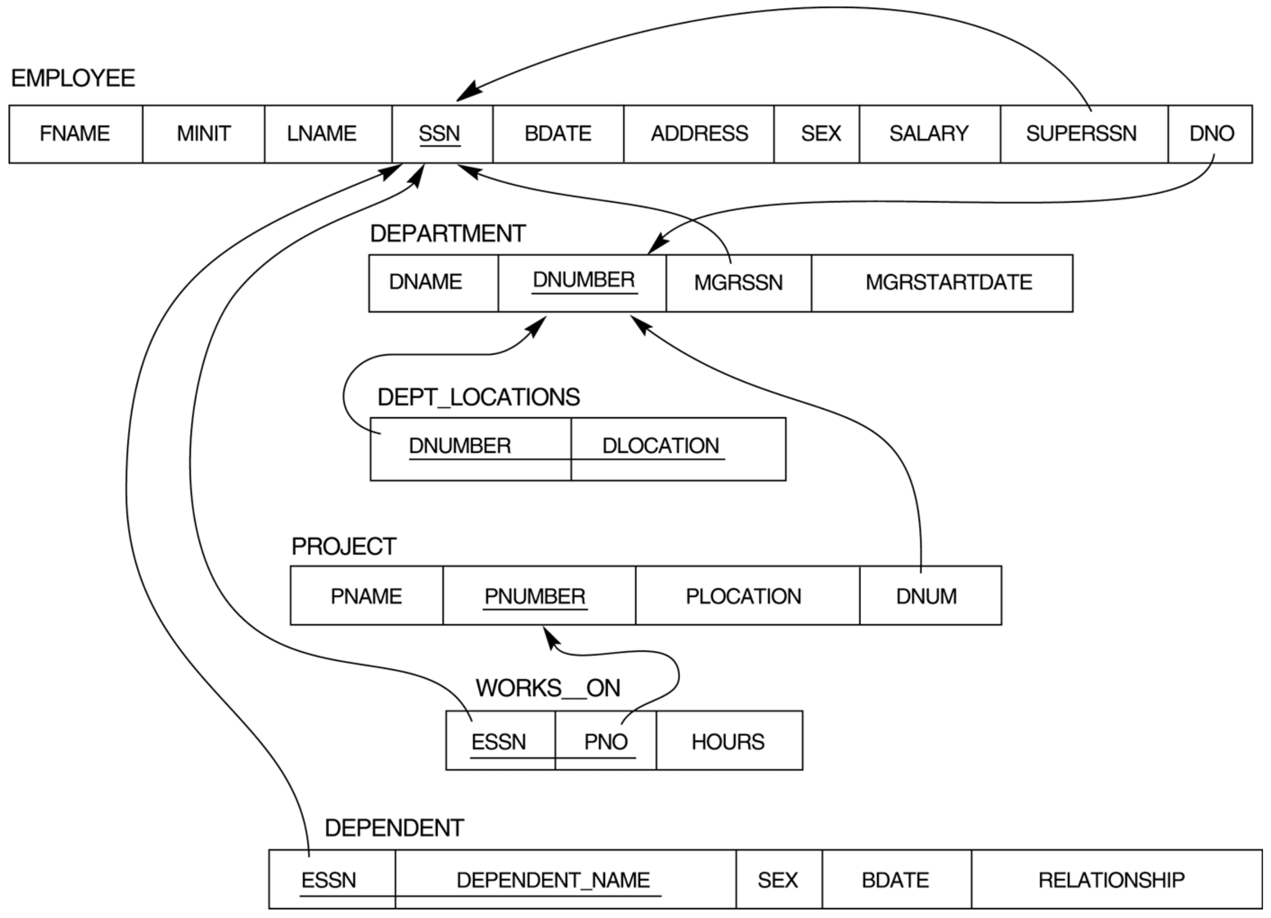


FIGURE 7.2

Result of mapping the COMPANY ER schema into a relational schema.



ER-to-Relational Mapping Algorithm (cont)

- **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 SUPPLY 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 4.11
Ternary relationship types. (a) The SUPPLY relationship.

(a)

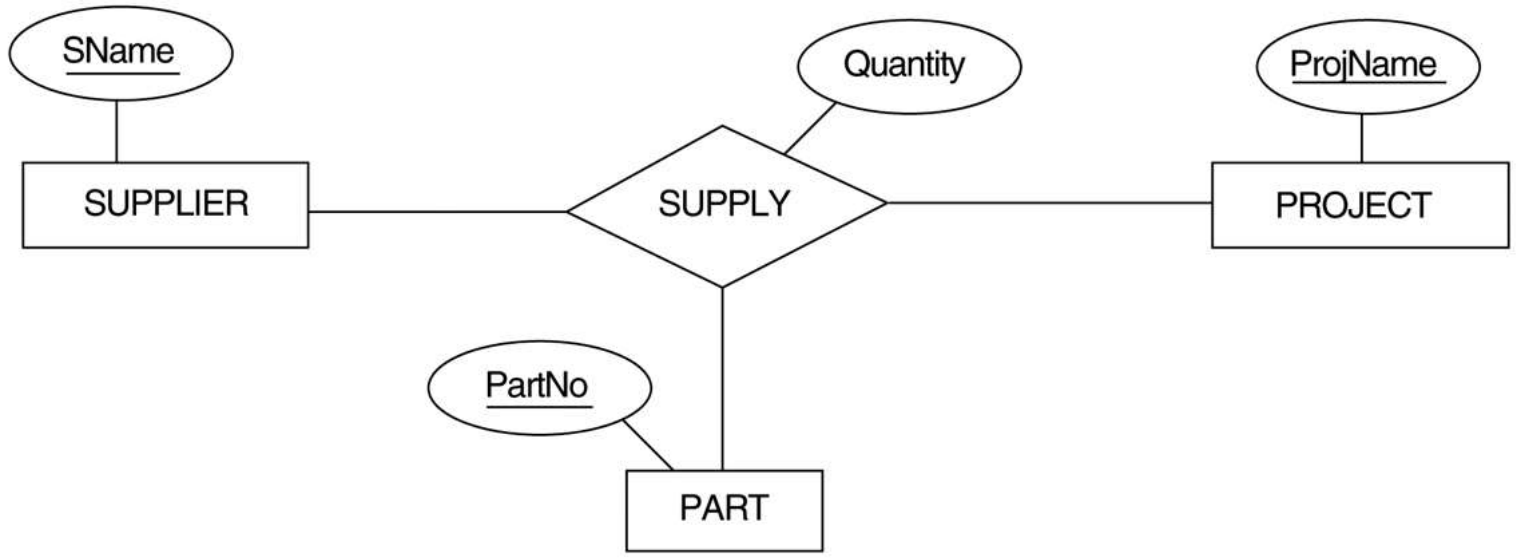
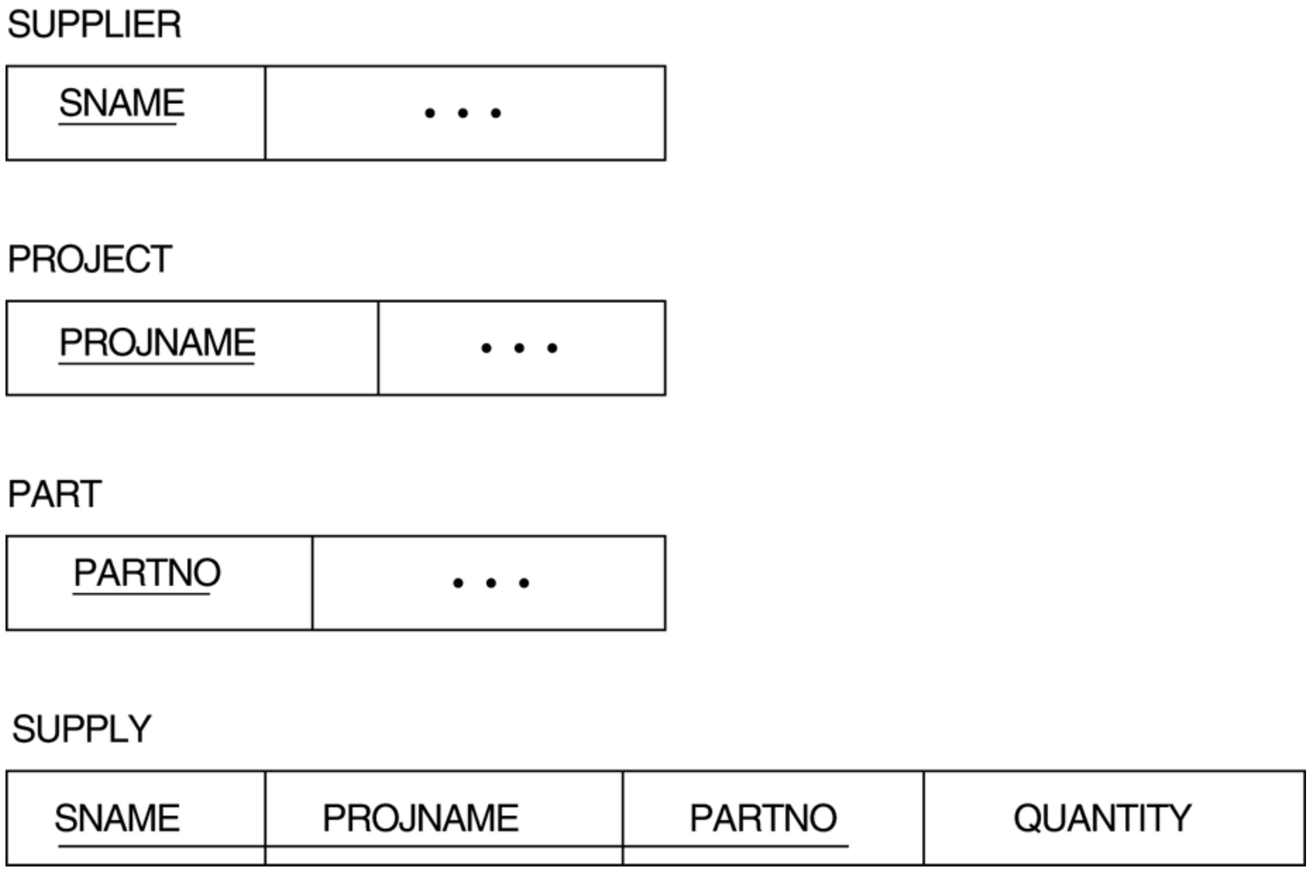


FIGURE 7.3
Mapping the *n*-ary relationship type SUPPLY from Figure 4.11a.



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 to 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 attributesAttrs$(L_i) = \{k\} \cup \{$attributes of $S_i\}$ and $PK(L_i)=k$.
- This option works **for any specialization** (total or partial, disjoint or over-lapping).

FIGURE 4.4
EER diagram notation for an attribute-defined specialization on JobType.

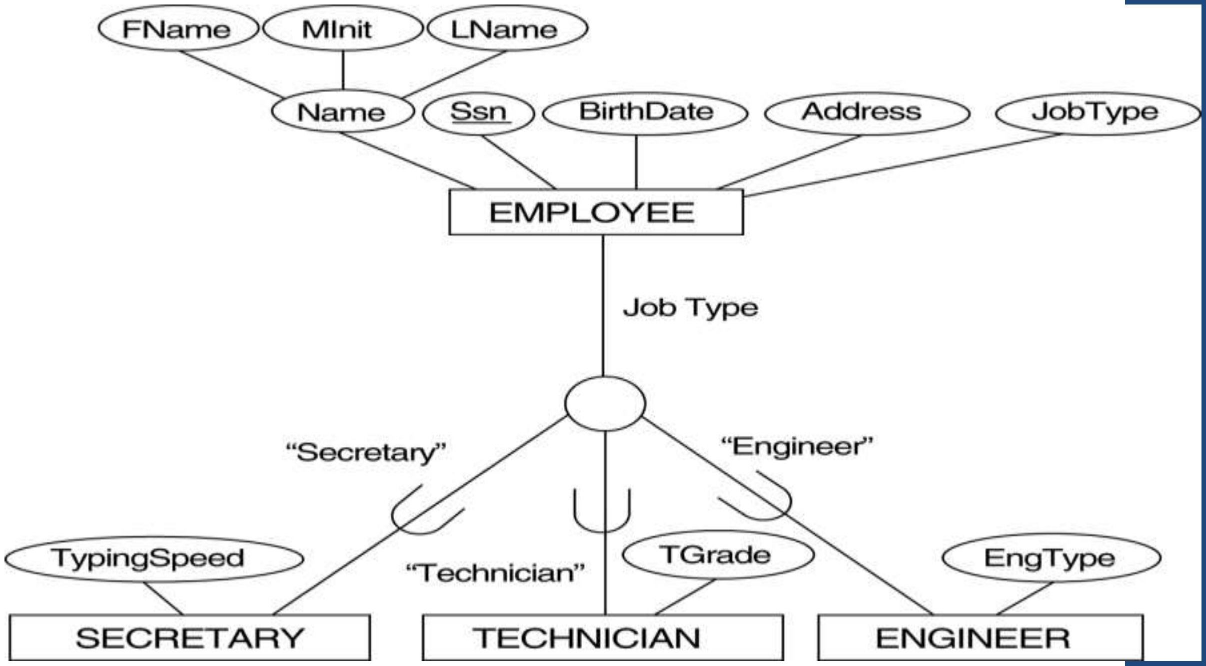
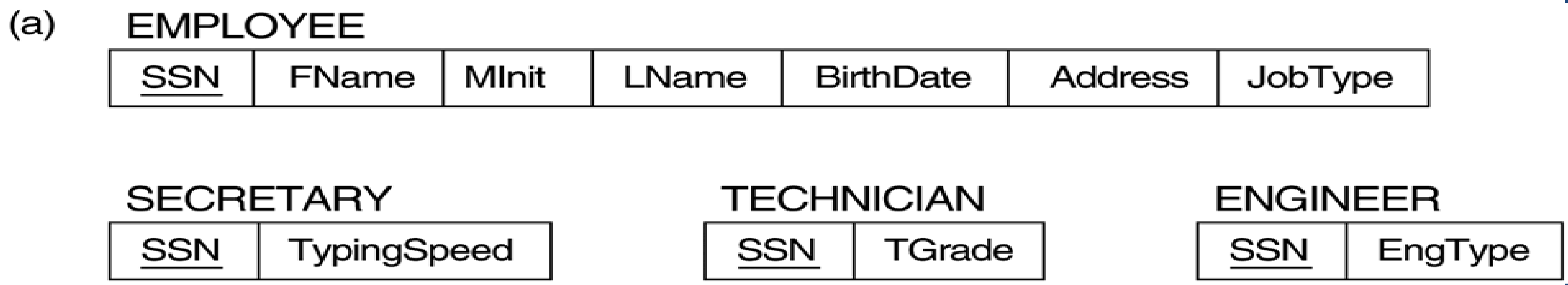


FIGURE 7.4
Options for mapping specialization or generalization.
(a) Mapping the EER schema in Figure 4.4 using option 8A.



(a) EMPLOYEE

SSN	FName	MInit	LName	BirthDate	Address	JobType

SECRETARY

SSN	TypingSpeed

TECHNICIAN

SSN	TGrade

ENGINEER

SSN	EngType

Mapping EER Model Constructs to Relations (contd..)

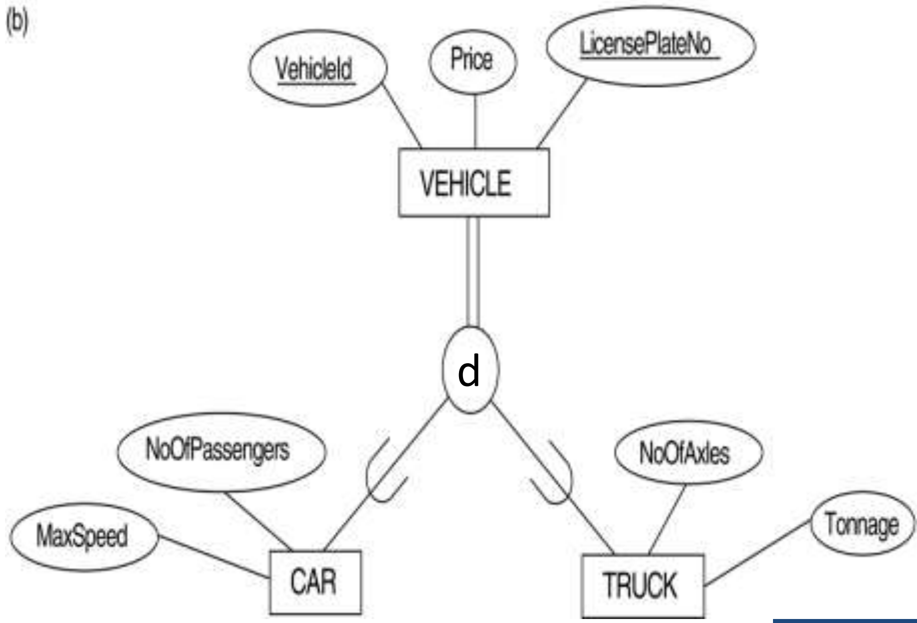
Step 8: Options for Mapping Specialization or Generalization.

Option 8B: Multiple relations-Subclass relations only

- Create a relation L_i for each subclass S_i, $1 < i < m$, with the attributes $Attr(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1 \ldots, a_n\}$ 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).
- Recommended if specialization has disjointedness constraint

FIGURE 4.3

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



Options for mapping specialization or generalization.
(b) Mapping the EER schema in Figure 4.3b using option 8B.

(b) CAR

<u>VehicleId</u>	LicensePlateNo	Price	MaxSpeed	NoOfPassengers

TRUCK

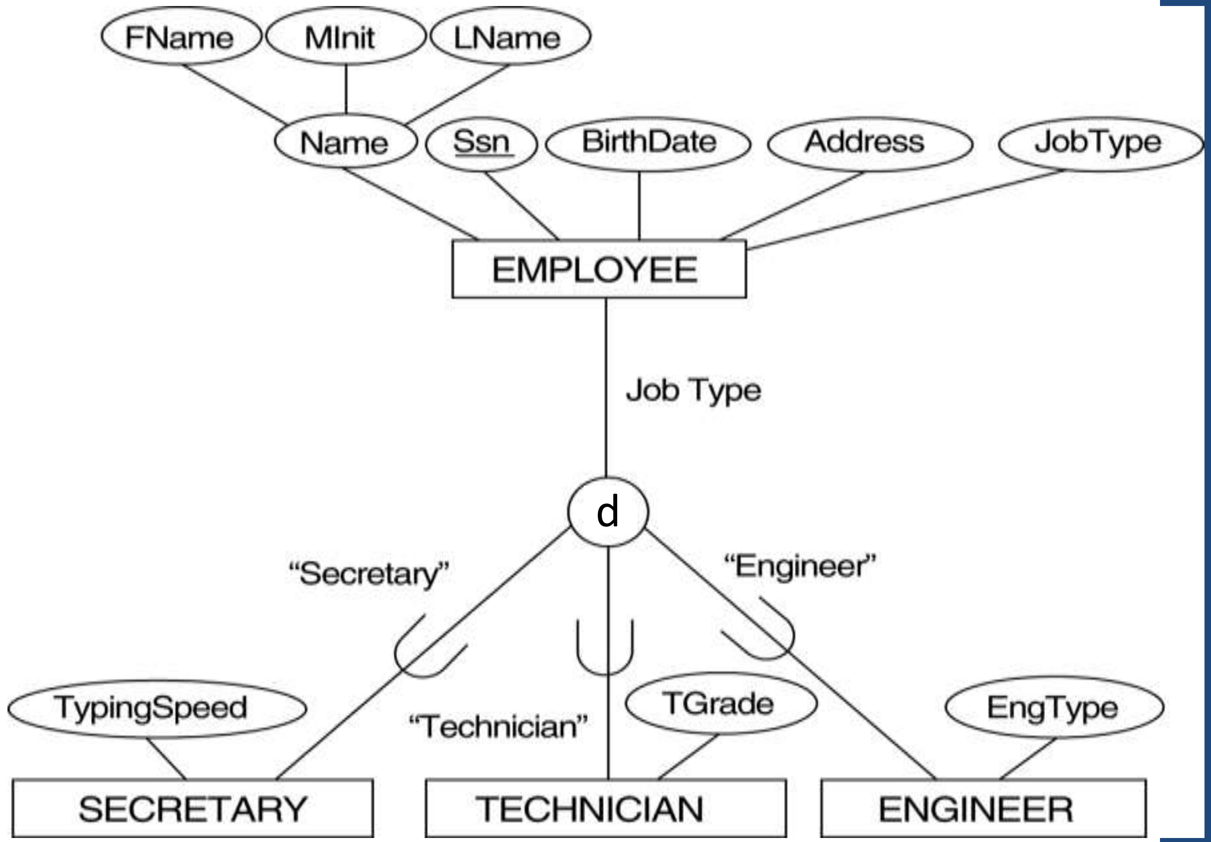
<u>VehicleId</u>	LicensePlateNo	Price	NoOfAxles	

Mapping EER Model Constructs to Relations (cont)

Option 8C: Single relation with one type attribute.

- Create a single relation L with attributes Attrs(L) = {k,a_1,…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
- Recommended for specialization whose **subclasses are disjoint** and has potential of generating many null values if specific attributes exist in subclasses

FIGURE 4.4
EER diagram notation for an attribute-defined specialization on JobType.



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

(c) EMPLOYEE

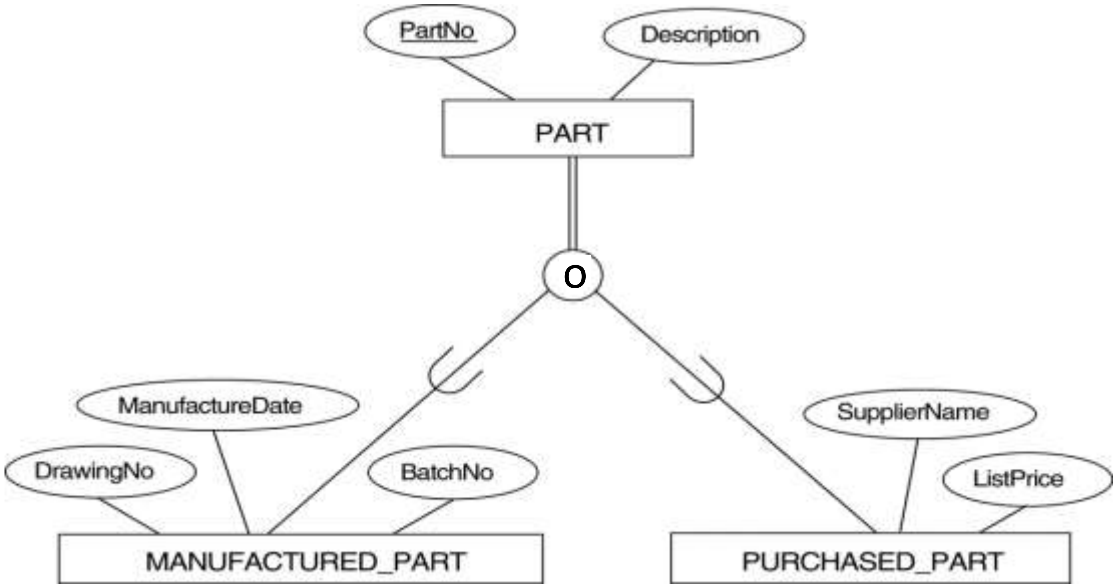
SSN	FName	MInit	LName	BirthDate	Address	JobType	TypingSpeed	TGrade	EngType

Mapping EER Model Constructs to Relations (cont)

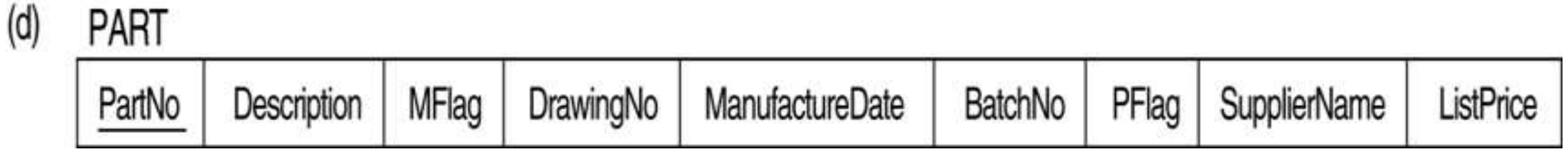
Option 8D: Single relation with multiple type attributes.

- 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.
- Recommended for specialization whose **subclasses are overlapping**
- Can be used for **disjoint subclasses** as well

FIGURE 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.



Options for mapping specialization or generalization.
(d) Mapping Figure 4.5 using option 8D with Boolean type fields Mflag and Pflag.


(d) PART

PartNo	Description	MFlag	DrawingNo	ManufactureDate	BatchNo	PFlag	SupplierName	ListPrice

Mapping EER Model (

(cont)

- **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.

Below both 8C and 8D are used for the shared class STUDENT_ASSISTANT.

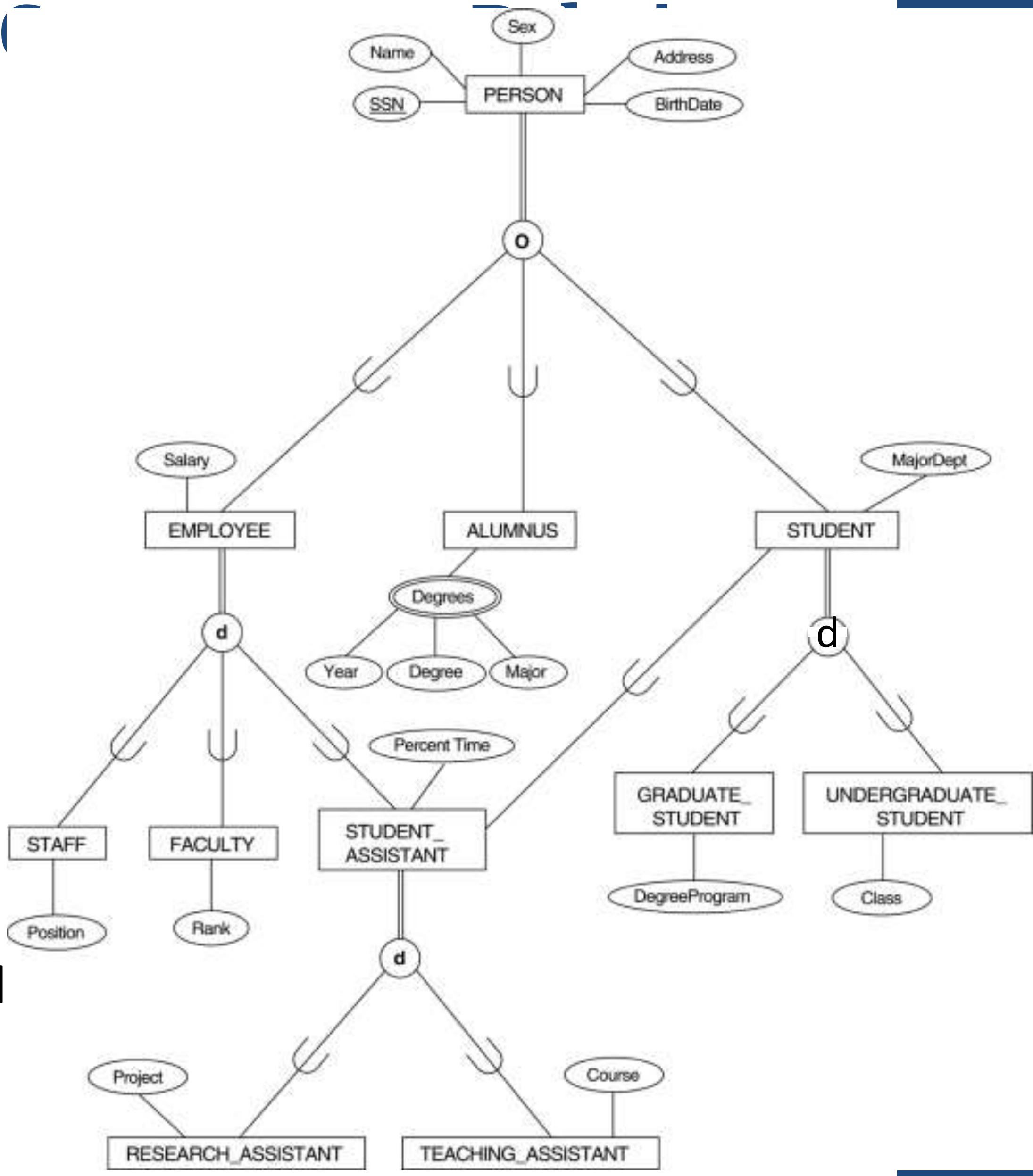


FIGURE 7.5
Mapping the EER specialization lattice in Figure 4.6 using multiple options.

PERSON

SSN	Name	BirthDate	Sex	Address

EMPLOYEE

SSN	Salary	EmployeeType	Position	Rank	PercentTime	RAFlag	TAFlag	Project	course

ALUMNUS

SSN

ALUMNUS_DEGREES

SSN	Year	Degree	Major

STUDENT

SSN	MajorDept	GradFlag	UndergradFlag	DegreeProgram	Class	StudAssistFlag

Mapping EER Model Constructs to Relations (cont)

- **Step 9: Mapping of Union Types (Categories).**
  - For mapping a category whose defining superclass have different keys, it is customary to specify a new key attribute, called a **surrogate key**, when creating a relation to correspond to the category.
  - In the example below we can create a relation OWNER to correspond to the OWNER category and include any attributes of the category in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId.

FIGURE 4.8
Two categories (union types): OWNER and REGISTERED_VEHICLE.

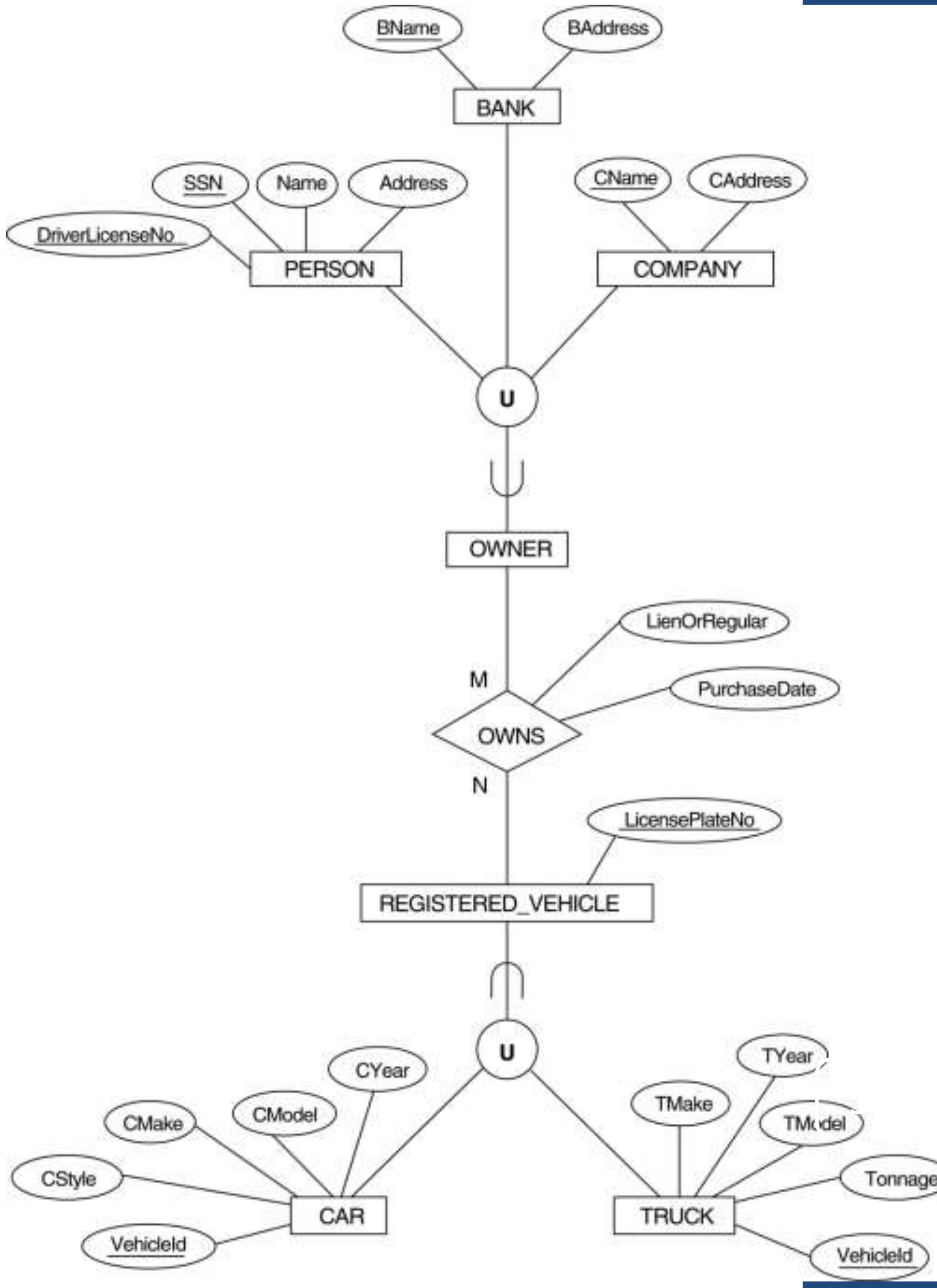


FIGURE 7.6
Mapping the EER categories (union types) in Figure 4.7 to relations.

PERSON

SSN	DriverLicenseNo	Name	Address	OwnerId

BANK

BName	BAddress	OwnerId

COMPANY

CName	CAddress	OwnerId

OWNER

OwnerId

REGISTERED_VEHICLE

VehicleId	LicensePlateNumber

CAR

VehicleId	CStyle	CMake	CModel	

TRUCK

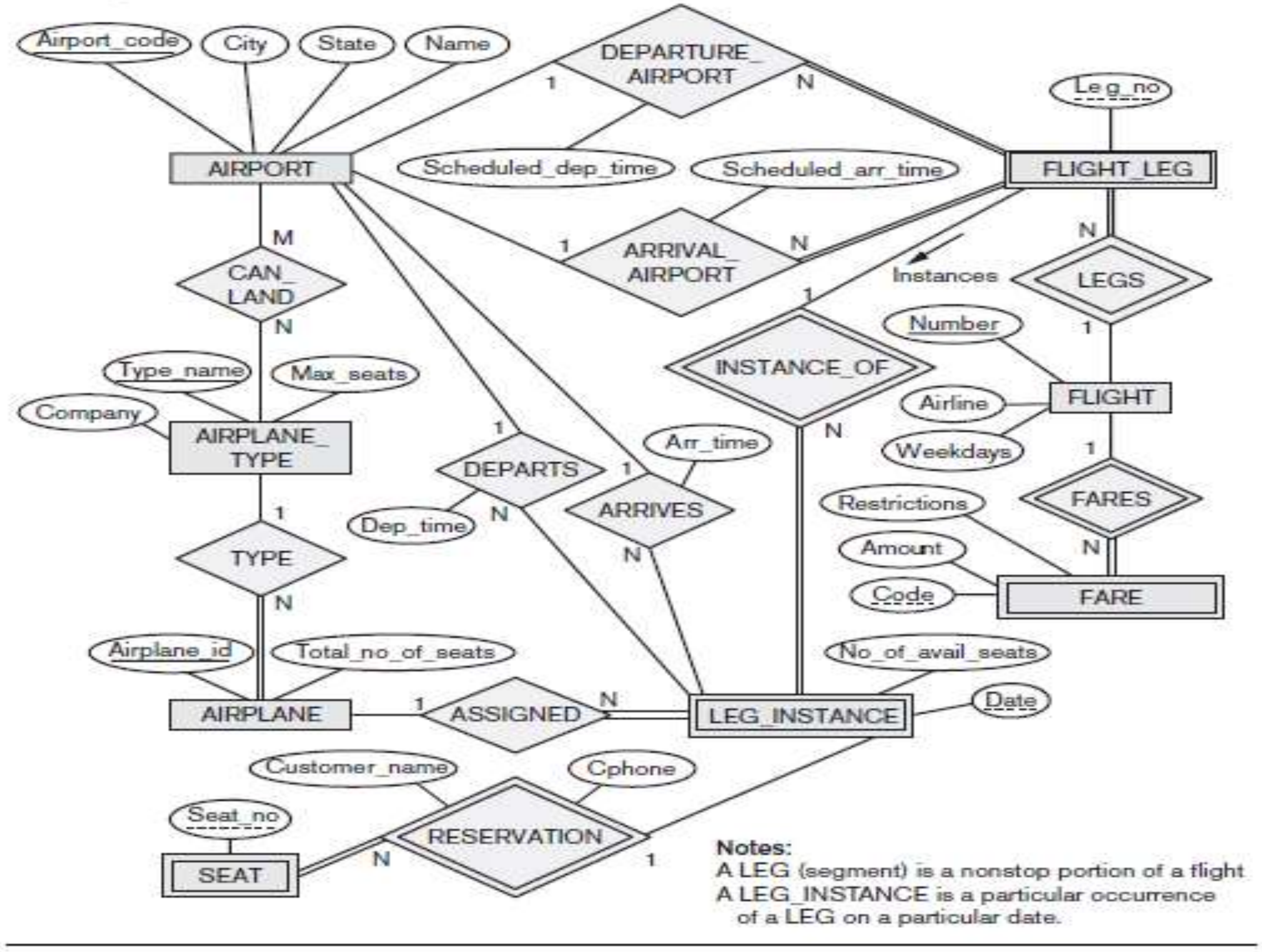
VehicleId	TMake	TModel	Tonnage	TYear

OWNS

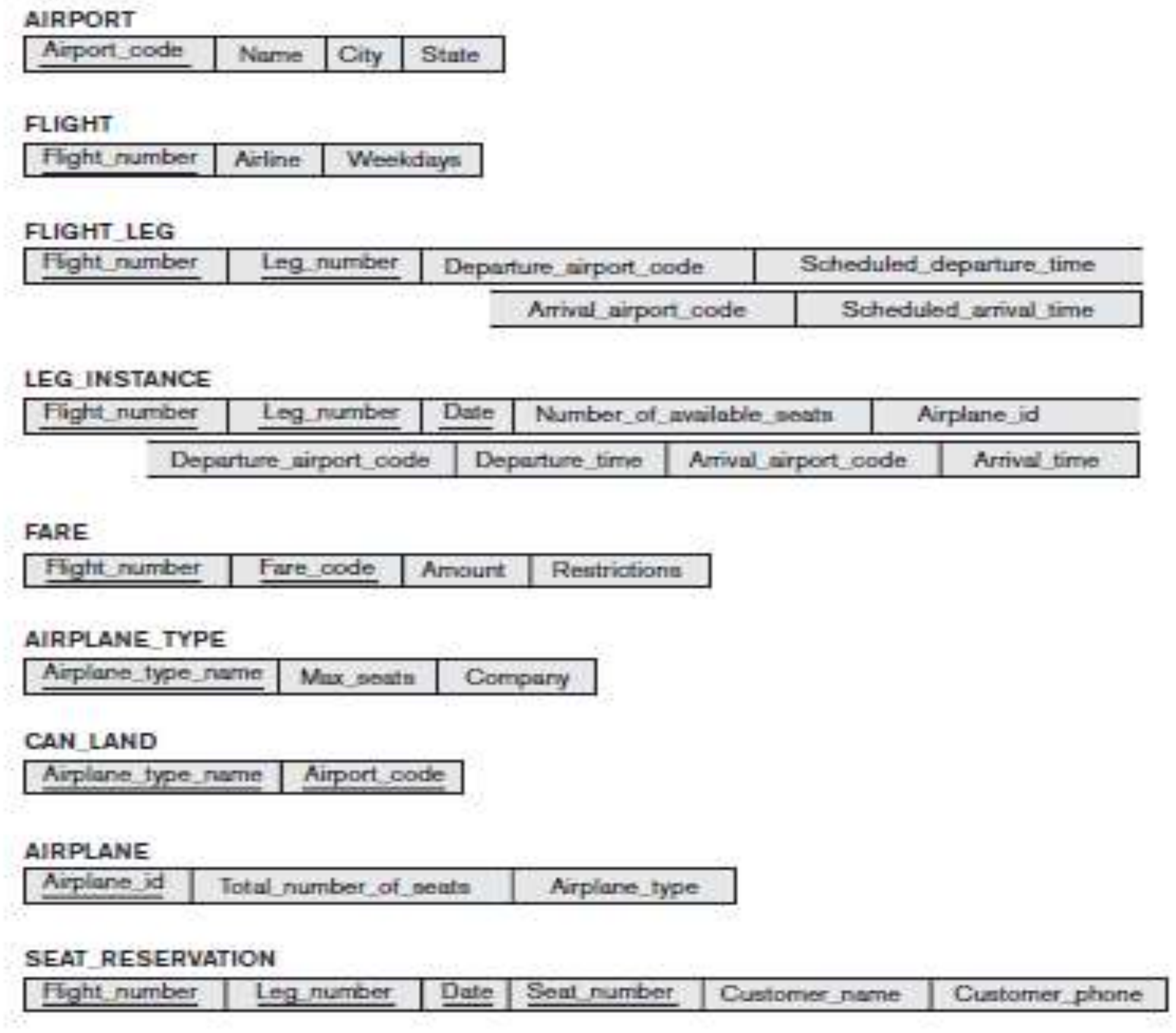
OwnerId	VehicleId	PurchaseDate	LienOrRegular

Exercise

Figure 7.20
An ER diagram for an AIRLINE database schema.



Notes:
A LEG (segment) is a nonstop portion of a flight
A LEG_INSTANCE is a particular occurrence of a LEG on a particular date.



AIRPORT

Airport_code	Name	City	State

FLIGHT

Flight_number	Airline	Weekdays

FLIGHT_LEG

Flight_number	Leg_number	Departure_airport_code	Scheduled_departure_time	Arrival_airport_code	Scheduled_arrival_time

LEG_INSTANCE

Flight_number	Leg_number	Date	Number_of_available_seats	Airplane_id	Departure_airport_code	Departure_time	Arrival_airport_code	Arrival_time

FARE

Flight_number	Fare_code	Amount	Restrictions

AIRPLANE_TYPE

Airplane_type_name	Max_seats	Company

CAN_LAND

Airplane_type_name	Airport_code

AIRPLANE

Airplane_id	Total_number_of_seats	Airplane_type

SEAT_RESERVATION

Flight_number	Leg_number	Date	Seat_number	Customer_name	Customer_phone

Mapping Exercise

Exercise 7.4.

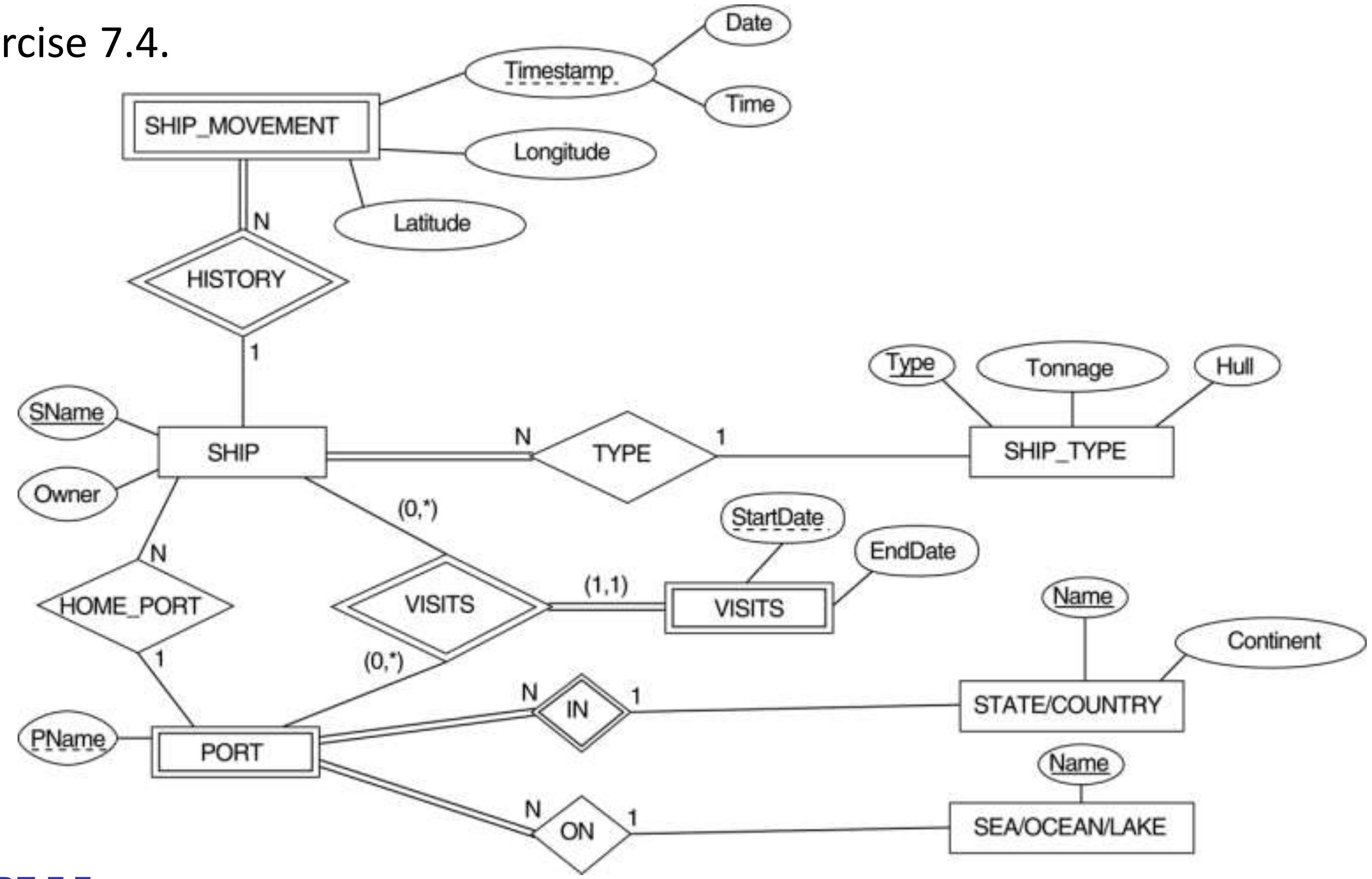


FIGURE 7.7
An ER schema for a SHIP_TRACKING database.

2. Data Manipulation

- Relation set processing facilities are available to the user
- Using relational operators, tables(relation) are manipulated so application program need not use loops
- **Purpose of database**: provide information to users within the enterprise
- Process of querying a relational database is essence of manipulating tables
- Two formal data manipulation languages are proposed by Codd for relational Model:
  - **Relational Algebra**
  - **Relational Calculus**

Relational Algebra And Relational Calculus: Outline

- **Relational Algebra**
  - Unary Relational Operations
  - Relational Algebra Operations From Set Theory
  - Binary Relational Operations
  - Additional Relational Operations
  - Examples of Queries in Relational Algebra

- **Relational Calculus**
  - Tuple Relational Calculus
  - Domain Relational Calculus

- Example Database Application (COMPANY)

Relational Algebra Overview

- Basic set of operations for the relational model
- Enable a user to specify **basic retrieval requests** (or **queries**)
- The result of an operation is a *new relation*
    - The **algebra operations** thus produce new relations which can be further manipulated using operations of the same algebra
- A sequence of relational algebra operations forms a **relational algebra expression**
    - The result of a relational algebra expression is also a relation that represents the result of a database query (or retrieval request)

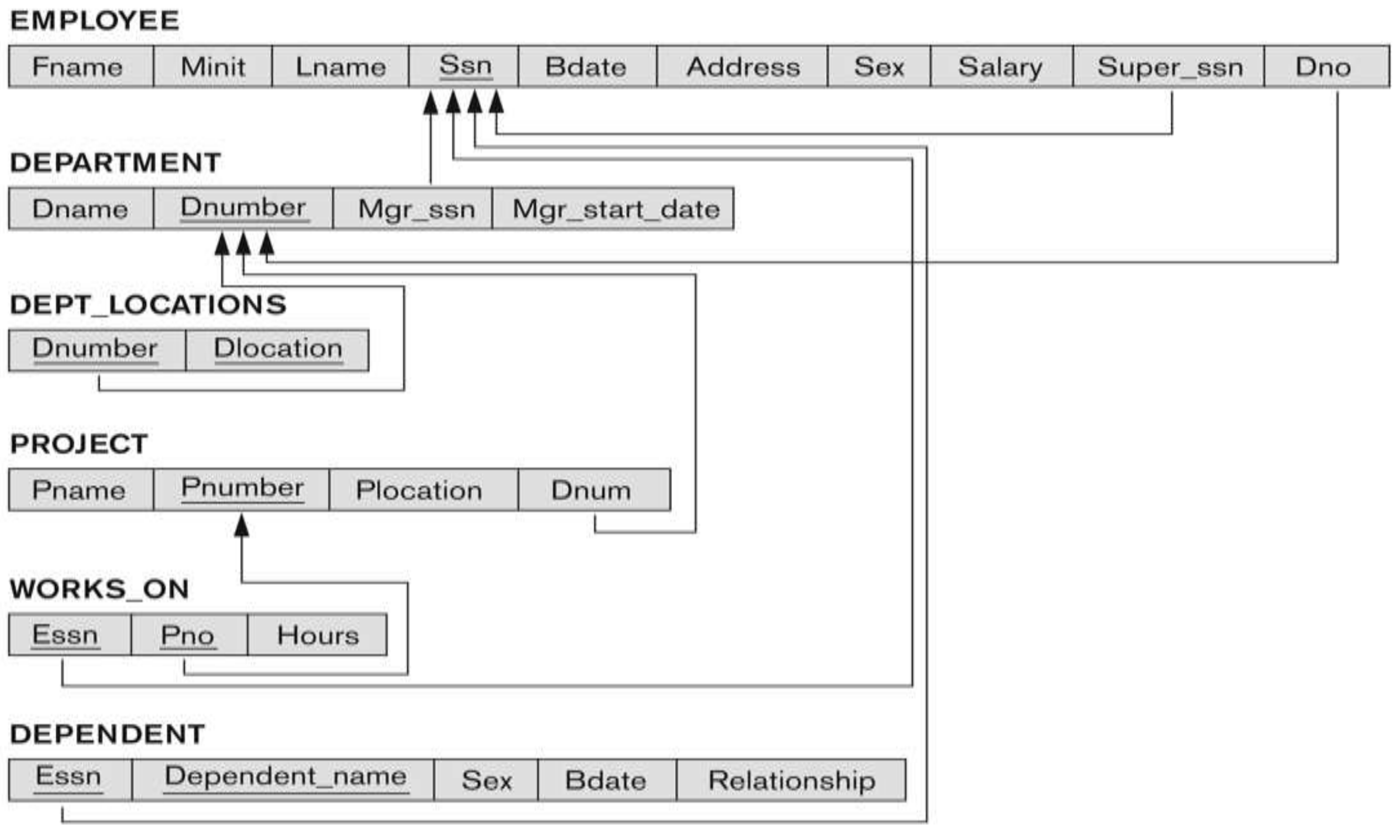
Relational Algebra Overview

- Relational Algebra consists of several groups of operations
  - **Unary Relational Operations**
    - SELECT (symbol: **σ** (sigma))
    - PROJECT (symbol: **π** (pi))
    - RENAME (symbol: **ρ** (rho))
  - **Relational Algebra Operations From Set Theory**
    - UNION (∪), INTERSECTION (∩), DIFFERENCE (or MINUS, **–**)
    - CARTESIAN PRODUCT (**x**)
  - **Binary Relational Operations**
    - JOIN (several variations of JOIN exist)
    - DIVISION
  - **Additional Relational Operations**
    - OUTER JOINS, OUTER UNION
    - AGGREGATE FUNCTIONS (These compute summary of information: SUM, COUNT, AVG, MIN, MAX)

Database Schema for COMPANY

- All examples discussed below refer to the COMPANY database shown here.

Figure 5.7
Referential integrity constraints displayed on the COMPANY relational database schema.



Database state

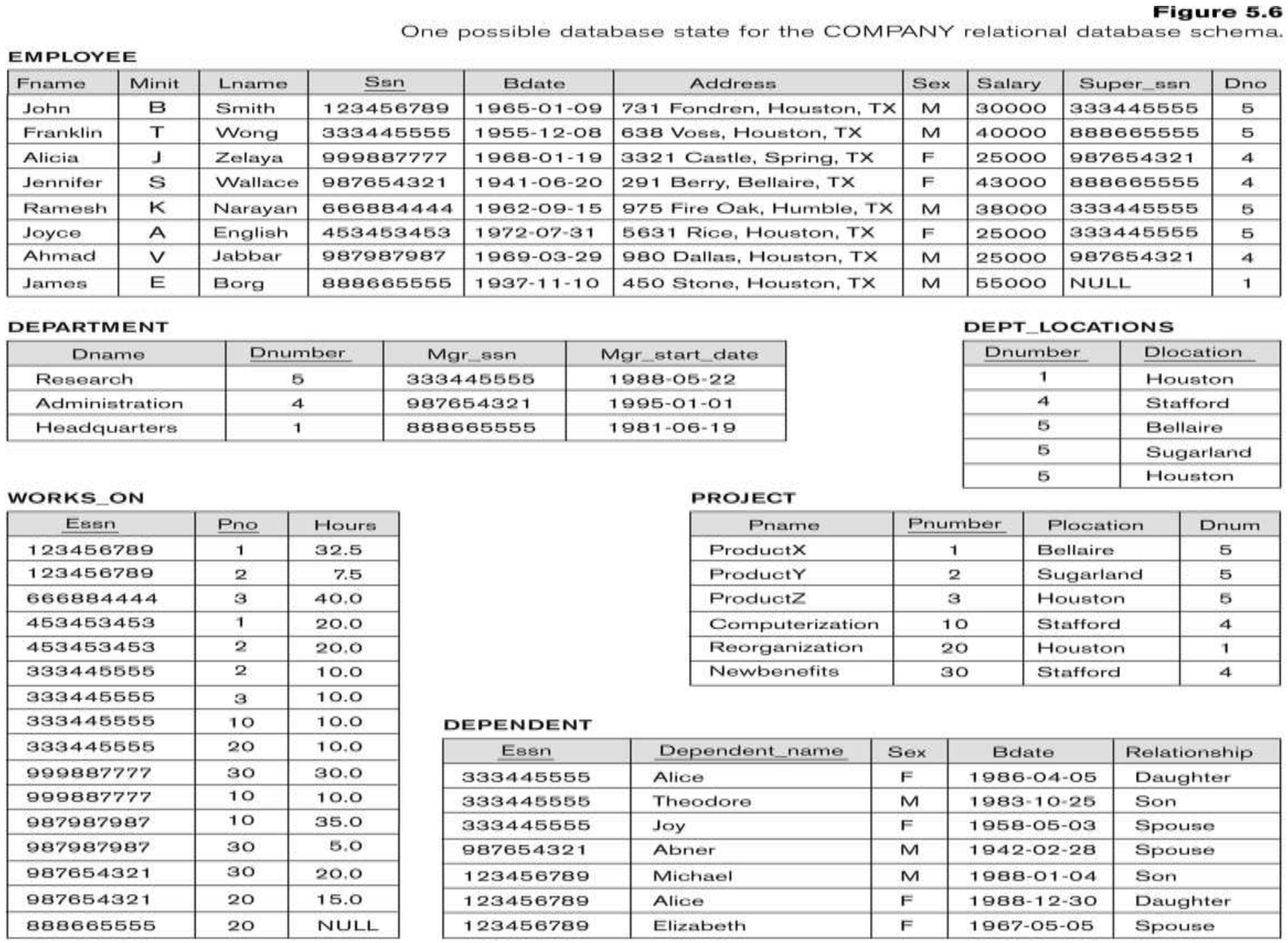


Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Unary Relational Operations: SELECT

- The SELECT operation (denoted by **σ** (sigma)) is used to select a *subset* of the tuples from a relation based on a **selection condition**.
  - The selection condition acts as a **filter**
  - Keeps only those tuples that satisfy the qualifying condition
  - Tuples satisfying the condition are *selected* whereas the other tuples are discarded (*filtered out*)

- **Syntax**:

$\sigma_{\text{<selection condition>}}$(R) where

  - **σ (sigma)** is used to denote the *select* operator
    - **selection condition is a Boolean** (conditional)

Unary Relational Operations: **SELECT**

- Tuples that make the condition **true** are selected
  - appear in the result of the operation
- Tuples that make the condition **false** are filtered out
  - discarded from the result of the operation

- **Examples**:
  - Select the EMPLOYEE tuples whose department number is 4:

$$\sigma_{\text{DNO} = 4}(\text{EMPLOYEE})$$

  - Select the employee tuples whose salary is greater than $30,000:

$$\sigma_{\text{SALARY} > 30{,}000}(\text{EMPLOYEE})$$

Unary Relational Operations: **SELECT** (contd.)

- **SELECT Operation Properties**
  - The SELECT operation $\sigma_{\text{<selection condition>}}(R)$ produces a relation S that has the same schema (same attributes) as R
  - **SELECT σ is commutative:**
    - $\sigma_{\text{<condition1>}}(\sigma_{\text{< condition2>}}(R)) = \sigma_{\text{<condition2>}}(\sigma_{\text{< condition1>}}(R))$
  - **cascade (sequence) of SELECT operations** may be applied in any order:
    - $\sigma_{\text{<cond1>}}(\sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(R)) = \sigma_{\text{<cond2>}}(\sigma_{\text{<cond3>}}(\sigma_{\text{<cond1>}}(R)))$
  - **A cascade of SELECT operations may be replaced by a single selection with a conjunction of all the conditions:**
    - $\sigma_{\text{<cond1>}}(\sigma_{\text{< cond2>}}(\sigma_{\text{<cond3>}}(R)) = \sigma_{\text{<cond1> AND < cond2> AND < cond3>}}(R)))$

One possible database state for the COMPANY relational database schema.

EMPLOYEE

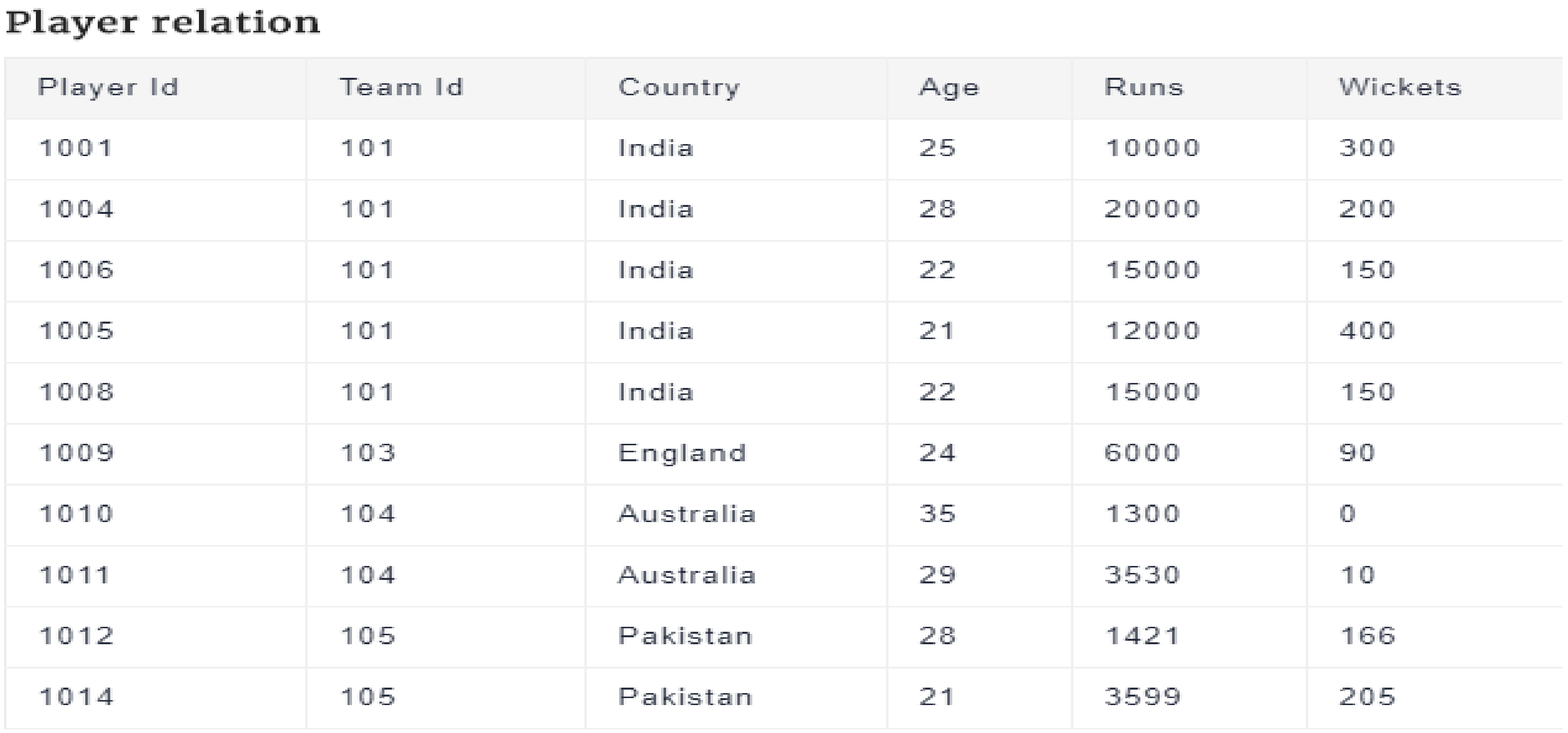
Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

Figure 6.1
Results of SELECT and PROJECT operations. (a) $\sigma_{(Dno=4 \textbf{ AND } Salary>25000) \text{ OR } (Dno=5 \textbf{ AND } Salary>30000)}$(EMPLOYEE). (b) $\pi_{Lname,\ Fname,\ Salary}$(EMPLOYEE). (c) $\pi_{Sex,\ Salary}$(EMPLOYEE).

(a)

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5

Exercise: Select Operator



Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

1. Find all tuples from player relation for which country is India.
2. Select all the tuples for which runs are greater than or equal to 15000.
3. Select all the players whose runs are greater than or equal to 6000 and age is less than 25

Solution

1. $\sigma_{\text{Country=India}}$ (Player)
2. $\sigma_{\text{Runs>=15000}}$ (Player)
3. $\sigma_{\text{Runs>6000 AND Age<25}}$(Player)

Unary Relational Operations: **PROJECT**

- PROJECT Operation is denoted by π (pi)
- This operation **keeps certain *columns* (attributes) from a relation** and discards the other columns.
  - PROJECT creates a vertical partitioning
    - The list of specified columns (attributes) is kept in each tuple
    - The other attributes in each tuple are discarded

- **Syntax:**

$$\pi_{\text{<attribute list>}}(R)$$

  - π (pi) is the symbol used to represent the *project* operation
  - <attribute list> is the desired list of attributes from relation R.

Unary Relational Operations: **PROJECT** (cont.)

- The project operation ***removes any duplicate tuples***
  - This is because the result of the *project* operation must be a *set of tuples*
    - Mathematical sets *do not allow* duplicate elements.

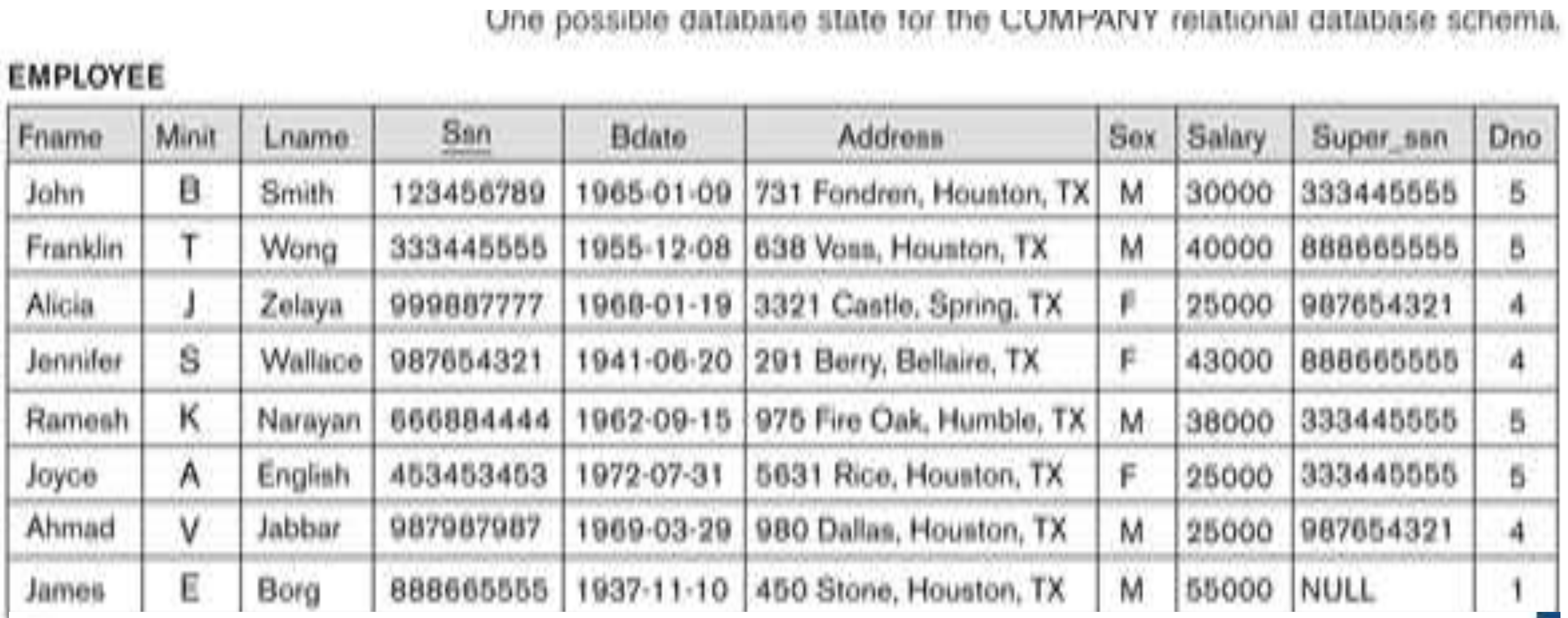
- **Example:**

To list each employee’s first and last name and salary, the following is used:

$$\pi_{\text{LNAME, FNAME,SALARY}}(\text{EMPLOYEE})$$

Unary Relational Operations: **PROJECT** (contd.)

- **PROJECT Operation Properties**
  - The number of tuples in the result of projection $\pi_{<list>}(R)$ is
    - always less or equal to the number of tuples in R
    - If the list of attributes includes a *key* of R, then the number of tuples in the result of PROJECT is *equal* to the number of tuples in R


One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

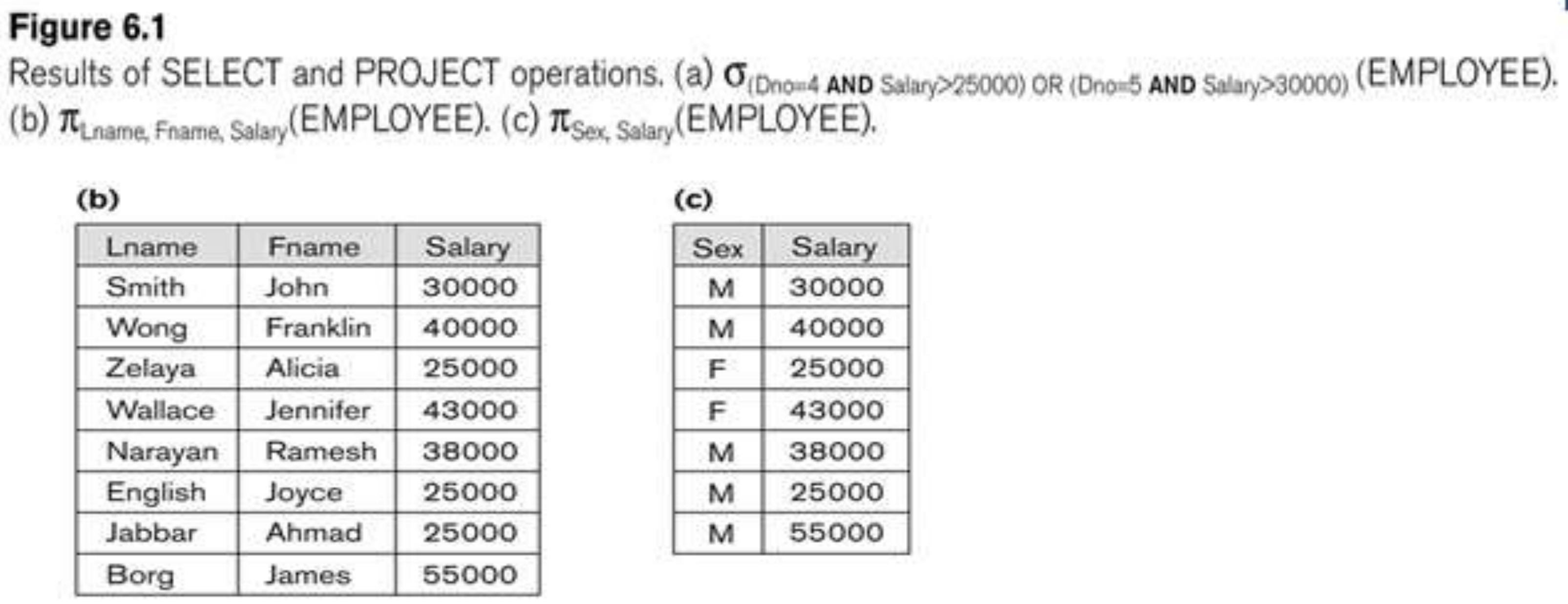


Figure 6.1
Results of SELECT and PROJECT operations. (a) $\sigma_{(\text{Dno}=4 \textbf{ AND } \text{Salary}>25000) \text{ OR } (\text{Dno}=5 \textbf{ AND } \text{Salary}>30000)}$(EMPLOYEE). (b) $\pi_{\text{Lname, Fname, Salary}}$(EMPLOYEE). (c) $\pi_{\text{Sex, Salary}}$(EMPLOYEE).

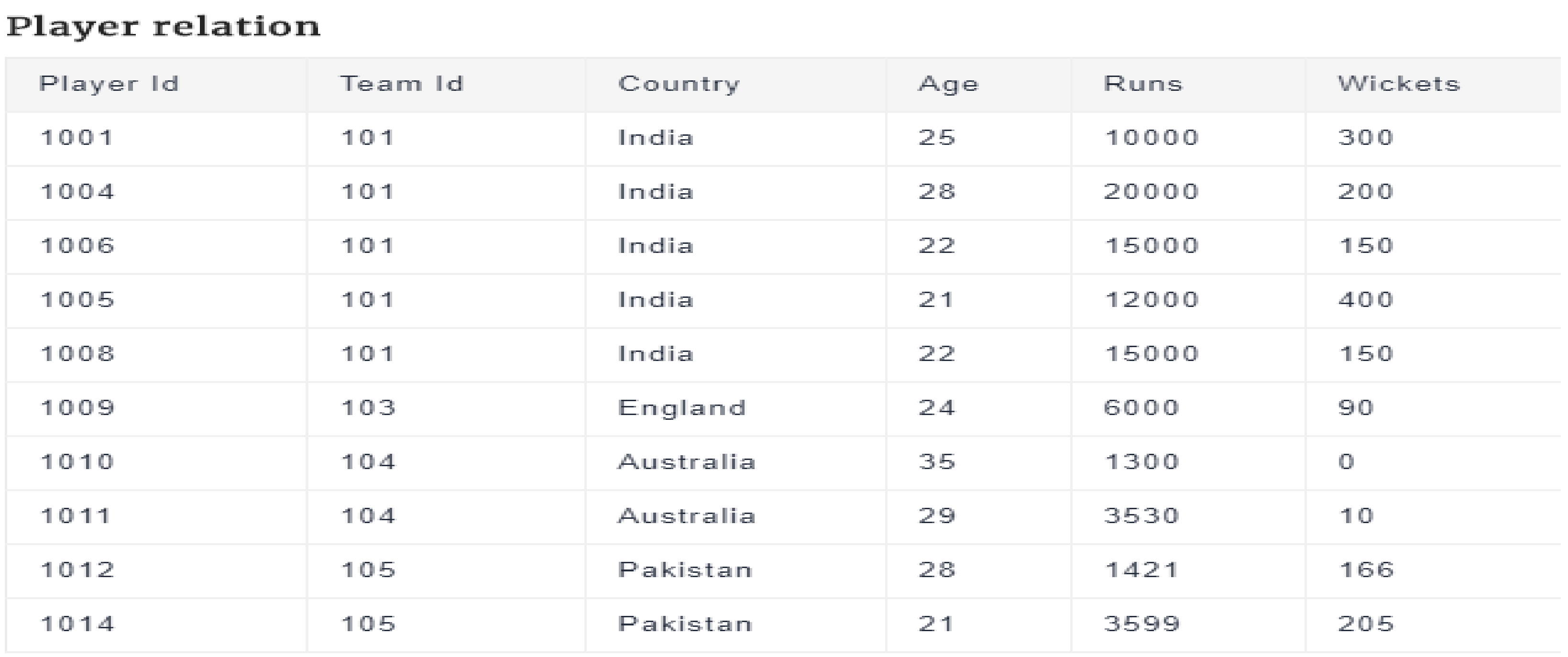
(b)

Lname	Fname	Salary
Smith	John	30000
Wong	Franklin	40000
Zelaya	Alicia	25000
Wallace	Jennifer	43000
Narayan	Ramesh	38000
English	Joyce	25000
Jabbar	Ahmad	25000
Borg	James	55000

(c)

Sex	Salary
M	30000
M	40000
F	25000
F	43000
M	38000
M	25000
M	55000

Exercise: Project Operator



Player relation

Player Id	Team Id	Country	Age	Runs	Wickets
1001	101	India	25	10000	300
1004	101	India	28	20000	200
1006	101	India	22	15000	150
1005	101	India	21	12000	400
1008	101	India	22	15000	150
1009	103	England	24	6000	90
1010	104	Australia	35	1300	0
1011	104	Australia	29	3530	10
1012	105	Pakistan	28	1421	166
1014	105	Pakistan	21	3599	205

1. List all the countries in Player relation.
2. List all the team ids and countries in Player Relation

1. π_{Country} (Player)
2. $\pi_{\text{Team Id, Country}}$ (Player)

Relational Algebra Expressions

- We may want to apply several relational algebra operations one after the other
  - Either we can write the operations as a single **relational algebra expression** by nesting the operations, or
  - We can apply one operation at a time and create **intermediate result relations**.
- In the latter case, we must give names to the relations that hold the intermediate results.

Single expression versus sequence of relational operations (Example)

- To retrieve the first name, last name, and salary of all employees who work in department number 5, we must apply a select and a project operation
- We can write a *single relational algebra expression* as follows:
  - $\pi_{\text{FNAME, LNAME, SALARY}}(\sigma_{\text{DNO=5}}(\text{EMPLOYEE}))$
- OR We can explicitly show the *sequence of operations*, giving a name to each intermediate relation:
  - $\text{DEP5\_EMPS} \leftarrow \sigma_{\text{DNO=5}}(\text{EMPLOYEE})$
  - $\text{RESULT} \leftarrow \pi_{\text{FNAME, LNAME, SALARY}}(\text{DEP5\_EMPS})$

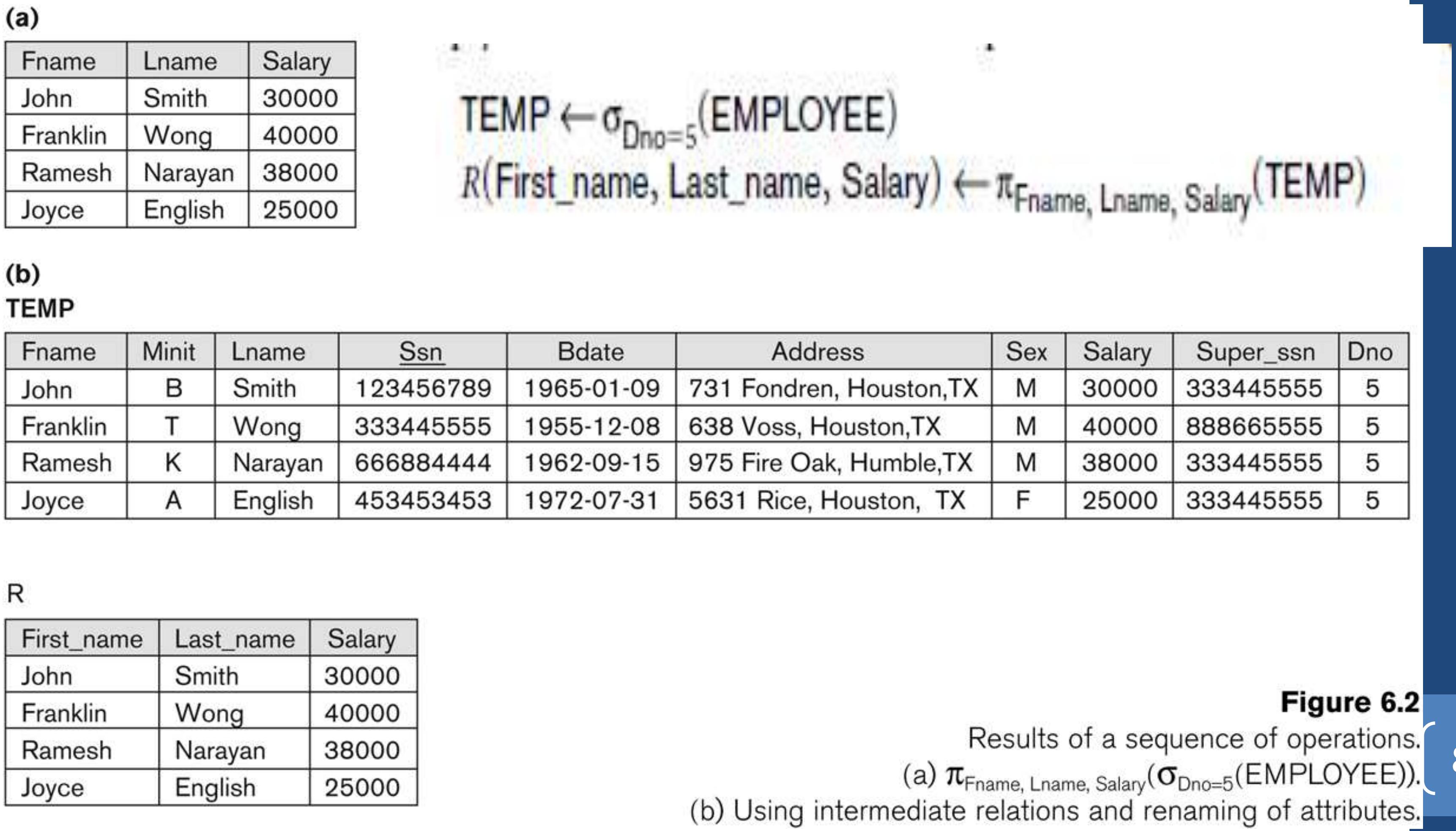
Unary Relational Operations: **RENAME**

- The RENAME operator is denoted by ρ **(rho)**
- In some cases, we may want to *rename* the attributes of a relation or the relation name or both
    - Useful when a query requires multiple operations
    - Necessary in some cases (see JOIN operation later)

Unary Relational Operations: **RENAME** (contd.)

- The general RENAME operation ρ can be expressed by any of the following forms:
  - **$\rho_{S\ (B1,\ B2,\ \ldots,\ Bn\ )}$(R) changes both:**
    - the relation name to S, *and*
    - the column (attribute) names to B1, B1, …..Bn
  - **ρ_S(R) changes:**
    - the *relation name* only to S
  - **$\rho_{(B1,\ B2,\ \ldots,\ Bn\ )}$(R) changes:**
    - the *column (attribute) names* only to B1, B1, …..Bn

Example of applying multiple operations and RENAME



(a)

Fname	Lname	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

$TEMP \leftarrow \sigma_{Dno=5}(EMPLOYEE)$

$R(First\_name, Last\_name, Salary) \leftarrow \pi_{Fname, Lname, Salary}(TEMP)$

(b)

TEMP

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston,TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston,TX	M	40000	888665555	5
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble,TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

R

First_name	Last_name	Salary
John	Smith	30000
Franklin	Wong	40000
Ramesh	Narayan	38000
Joyce	English	25000

Figure 6.2
Results of a sequence of operations.
(a) $\pi_{Fname, Lname, Salary}(\sigma_{Dno=5}(EMPLOYEE))$.
(b) Using intermediate relations and renaming of attributes.

Relational Algebra Operations from Set Theory: UNION

- **UNION Operation**
  - Binary operation, denoted by ∪
  - The result of R ∪ S, is a relation that includes all tuples that are either in R or in S or in both R and S
  - Duplicate tuples are eliminated
  - The two operand relations R and S must be "type compatible" (or UNION compatible)
    - R and S must have same number of attributes
    - Each pair of corresponding attributes must be type compatible (have same or compatible domains)

Relational Algebra Operations from Set Theory: **UNION**

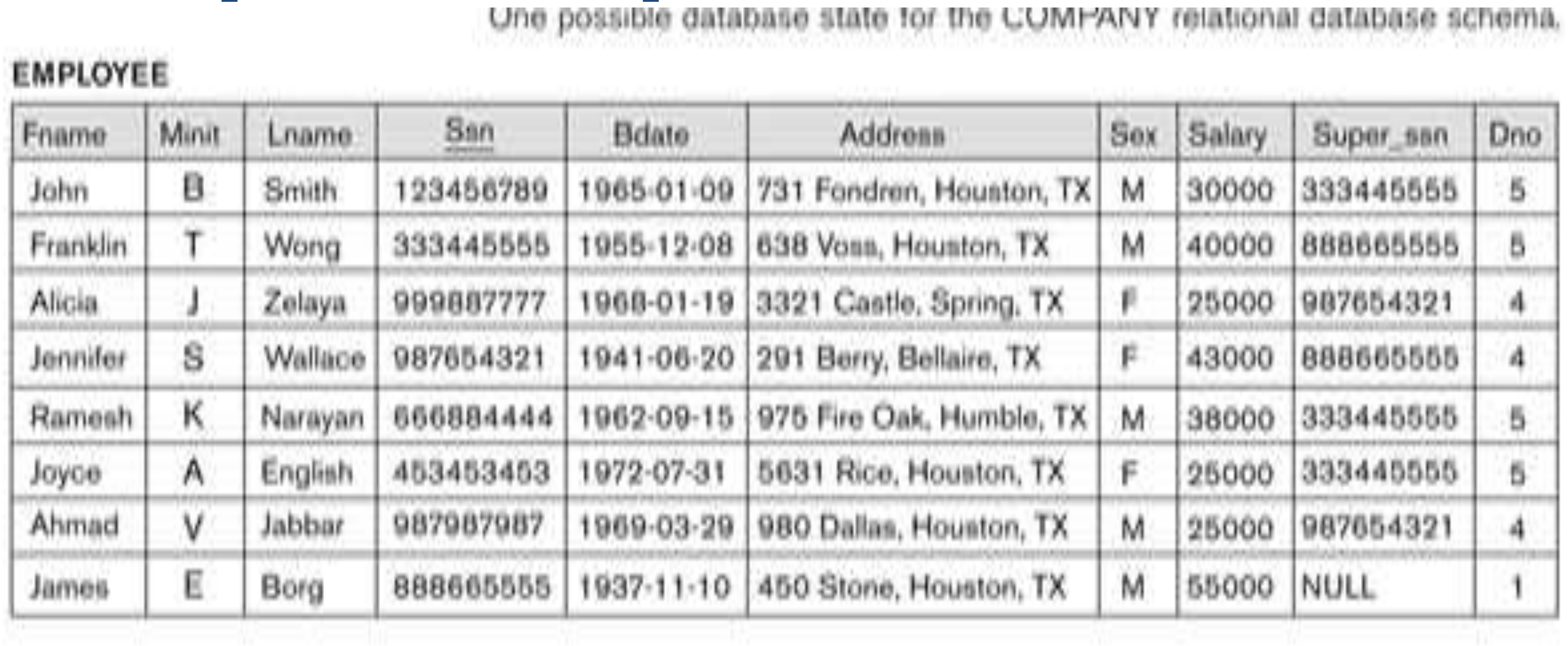
- Example:
  - To retrieve the social security numbers of all employees who either *work in department 5* (RESULT1 below) or *directly supervise an employee who works in department 5* (RESULT2 below)
  - We can use the UNION operation as follows:

$$DEP5\_EMPS \leftarrow \sigma_{DNO=5} (EMPLOYEE)$$
$$RESULT1 \leftarrow \pi_{SSN}(DEP5\_EMPS)$$

$$RESULT2(SSN) \leftarrow \pi_{SUPERSSN}(DEP5\_EMPS)$$
$$RESULT \leftarrow RESULT1 \cup RESULT2$$

  - The union operation produces the tuples that are in either RESULT1 or RESULT2 or both

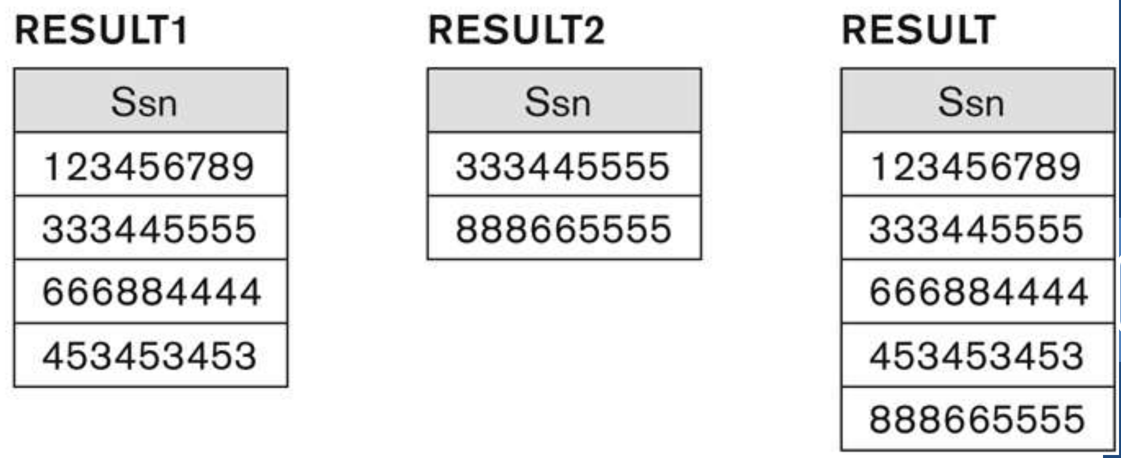
Example: **UNION** operation



One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1



RESULT1

Ssn
123456789
333445555
666884444
453453453

RESULT2

Ssn
333445555
888665555

RESULT

Ssn
123456789
333445555
666884444
453453453
888665555

Figure 6.3
Result of the UNION operation RESULT ← RESULT1 ∪ RESULT2.

Relational Algebra Operations from Set Theory

- **Type Compatibility of operands** is required for the binary set operation UNION ∪, (also for INTERSECTION ∩, and SET DIFFERENCE –)
- R1(A1, A2, ..., An) and R2(B1, B2, ..., Bn) are type compatible if:
  - they have the same number of attributes, and
  - the domains of corresponding attributes are type compatible (i.e. dom(Ai)=dom(Bi) for i=1, 2, ..., n).
- The resulting relation for R1∪R2 (also for R1∩R2, or R1–R2) has the same attribute names as the *first* operand relation R1 (by convention)

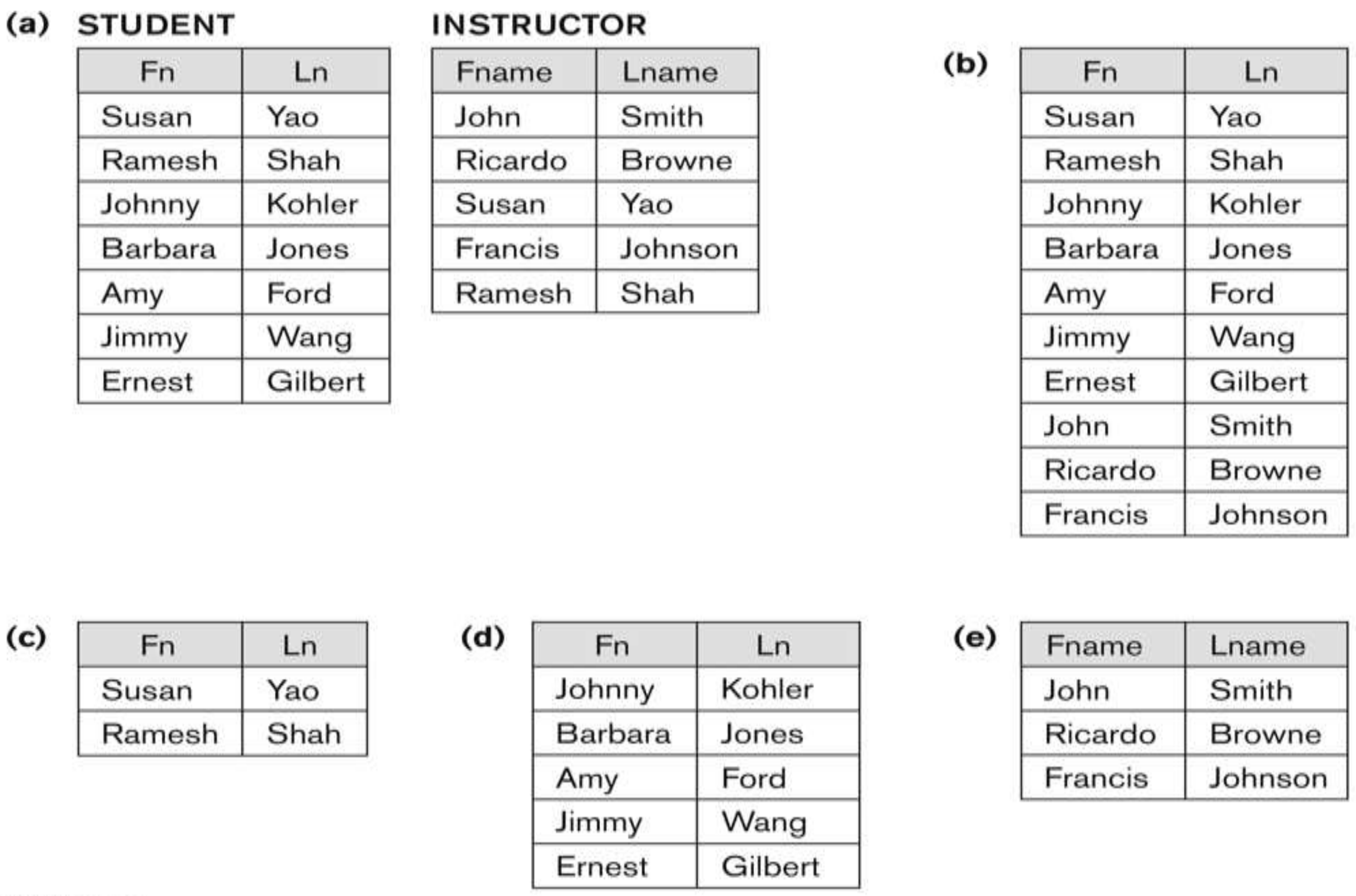
Relational Algebra Operations from Set Theory: **INTERSECTION**

- INTERSECTION is denoted by $\cap$
- The result of the operation $R \cap S$, is a relation that includes all tuples that are in **both R and S**
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be “type compatible”

Relational Algebra Operations from Set Theory: **SET DIFFERENCE**

- SET DIFFERENCE (also called MINUS or EXCEPT) is denoted by –
- The result of R – S, is a relation that includes **all tuples that are in R but not in S**
  - The attribute names in the result will be the same as the attribute names in R
- The two operand relations R and S must be “type compatible”

Example to illustrate :UNION, INTERSECT, and DIFFERENCE



(a) STUDENT

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

INSTRUCTOR

Fname	Lname
John	Smith
Ricardo	Browne
Susan	Yao
Francis	Johnson
Ramesh	Shah

(b)

Fn	Ln
Susan	Yao
Ramesh	Shah
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert
John	Smith
Ricardo	Browne
Francis	Johnson

(c)

Fn	Ln
Susan	Yao
Ramesh	Shah

(d)

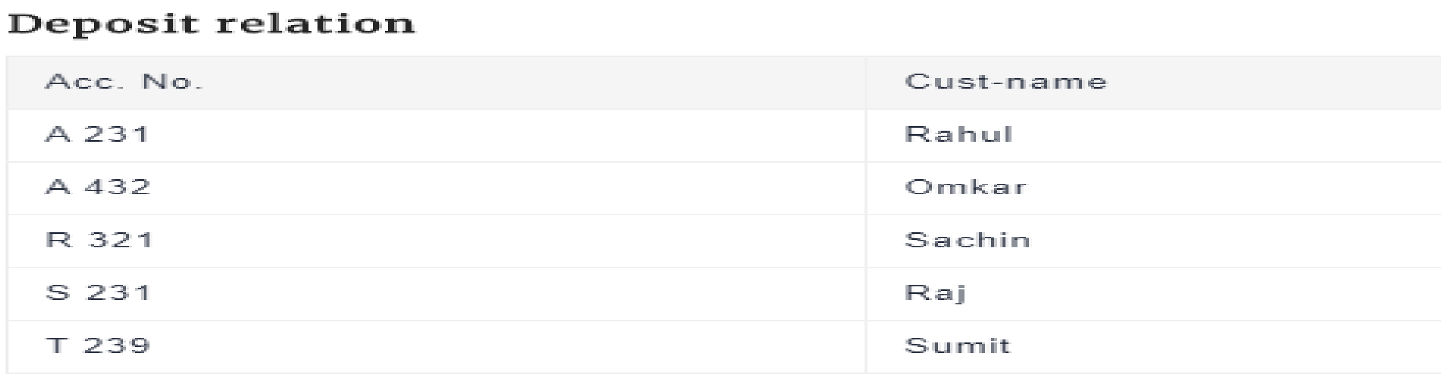
Fn	Ln
Johnny	Kohler
Barbara	Jones
Amy	Ford
Jimmy	Wang
Ernest	Gilbert

(e)

Fname	Lname
John	Smith
Ricardo	Browne
Francis	Johnson

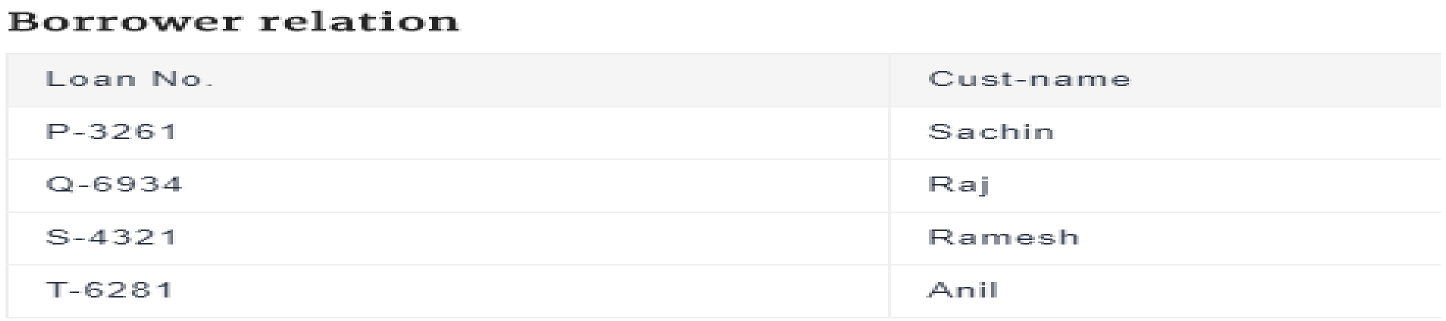
Figure 6.4
The set operations UNION, INTERSECTION, and MINUS. (a) Two union-compatible relations. (b) STUDENT $\cup$ INSTRUCTOR. (c) STUDENT $\cap$ INSTRUCTOR. (d) STUDENT $-$ INSTRUCTOR. (e) INSTRUCTOR $-$ STUDENT.

Exercise: Set theory operators



Deposit relation

Acc. No.	Cust-name
A 231	Rahul
A 432	Omkar
R 321	Sachin
S 231	Raj
T 239	Sumit



Borrower relation

Loan No.	Cust-name
P-3261	Sachin
Q-6934	Raj
S-4321	Ramesh
T-6281	Anil

1. Find all the customers having an account but not the loan.
2. Find all the customers having a loan but not the account.

   1. $\pi_{\text{cust-name}}(\text{Depositor}) - \pi_{\text{cust-name}}(\text{Borrower})$
   2. $\pi_{\text{cust-name}}(\text{Borrower}) - \pi_{\text{cust-name}}(\text{Depositor})$

Some properties of UNION, INTERSECT, and DIFFERENCE

- Notice that both union and intersection are *commutative* operations; that is
  - $R \cup S = S \cup R$, and $R \cap S = S \cap R$
- Both union and intersection can be treated as n-ary operations applicable to any number of relations as both are *associative* operations; that is
  - $R \cup (S \cup T) = (R \cup S) \cup T$
  - $(R \cap S) \cap T = R \cap (S \cap T)$
- The minus operation is not commutative; that is, in general
  - $R - S \neq S - R$

Relational Algebra Operations from Set Theory: **CARTESIAN PRODUCT**

- CARTESIAN (or CROSS) PRODUCT Operation
  - Combine tuples from two relations in a combinatorial fashion.
  - Denoted by R(A1, A2, . . ., A**n**) x S(B1, B2, . . ., B**m**)
  - Result is a relation Q with degree **n + m** attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples—one from R and one from S.
  - Hence, if R has n_R tuples (denoted as |R| = n_R), and S has n_S tuples, then R x S will have $\mathbf{n_R * n_S}$ tuples.
  - The **two operands do NOT have to be "type compatible”**

Example: Cartesian Product

- Relations *r*, *s*:

r

A	B
α	1
β	2

s

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

- *r* x *s*:

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Relational Algebra Operations from Set Theory: **CARTESIAN PRODUCT** (contd..)

- Generally, CROSS PRODUCT is not a meaningful operation
  - Can become meaningful when followed by other operations
- Example (not meaningful):

List of all female employee's dependent

  - FEMALE_EMPS $\leftarrow \sigma_{SEX='F'}$(EMPLOYEE)
  - EMPNAMES $\leftarrow \pi_{FNAME, LNAME, SSN}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS $\leftarrow$ EMPNAMES x DEPENDENT
- EMP_DEPENDENTS will contain every combination of EMPNAMES and DEPENDENT
  - whether or not they are actually related

Example of applying CARTESIAN PRODUCT

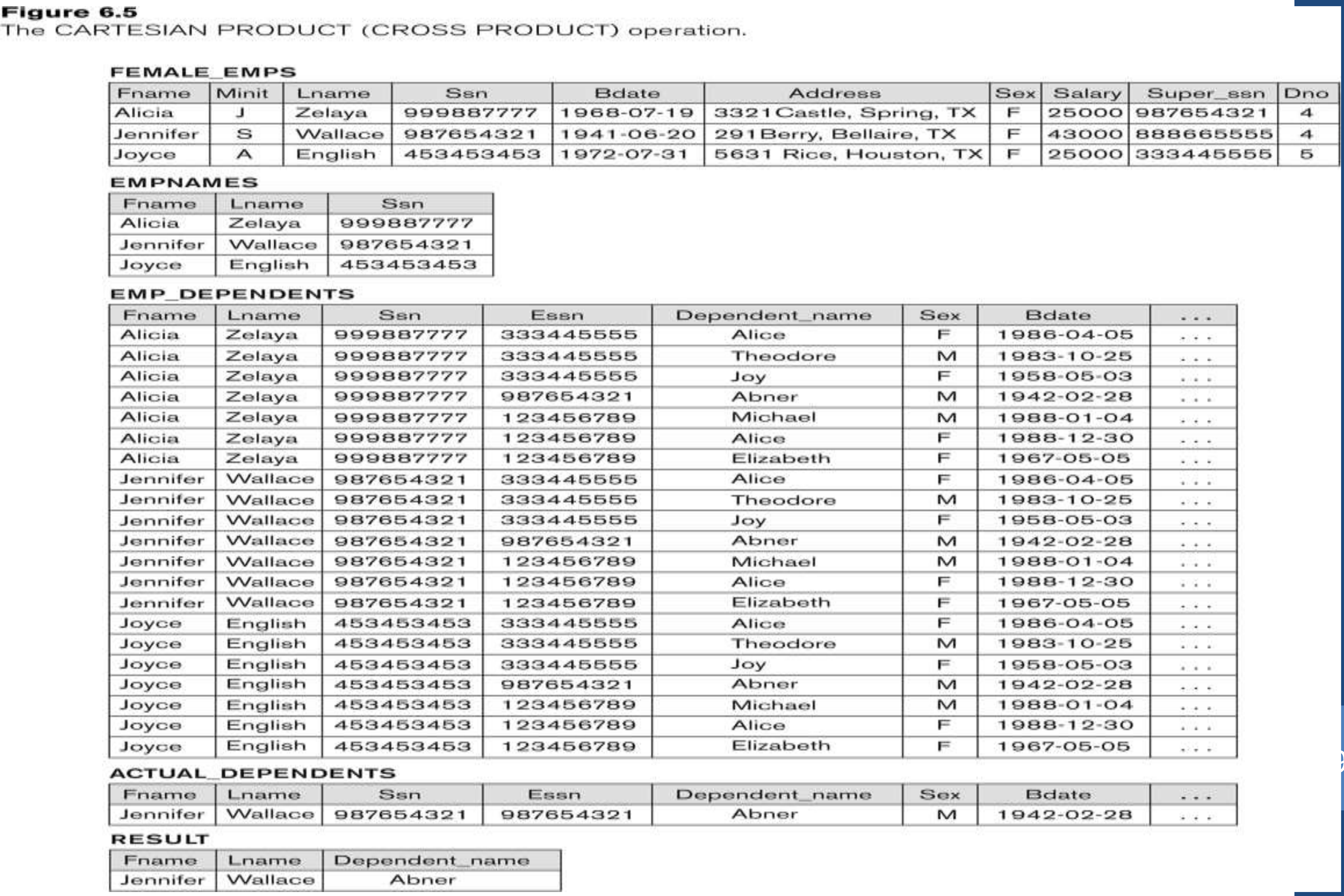


Figure 6.5
The CARTESIAN PRODUCT (CROSS PRODUCT) operation.

FEMALE_EMPS

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
Alicia	J	Zelaya	999887777	1968-07-19	3321Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291Berry, Bellaire, TX	F	43000	888665555	4
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5

EMPNAMES

Fname	Lname	Ssn
Alicia	Zelaya	999887777
Jennifer	Wallace	987654321
Joyce	English	453453453

EMP_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	. . .
Alicia	Zelaya	999887777	333445555	Alice	F	1986-04-05	. . .
Alicia	Zelaya	999887777	333445555	Theodore	M	1983-10-25	. . .
Alicia	Zelaya	999887777	333445555	Joy	F	1958-05-03	. . .
Alicia	Zelaya	999887777	987654321	Abner	M	1942-02-28	. . .
Alicia	Zelaya	999887777	123456789	Michael	M	1988-01-04	. . .
Alicia	Zelaya	999887777	123456789	Alice	F	1988-12-30	. . .
Alicia	Zelaya	999887777	123456789	Elizabeth	F	1967-05-05	. . .
Jennifer	Wallace	987654321	333445555	Alice	F	1986-04-05	. . .
Jennifer	Wallace	987654321	333445555	Theodore	M	1983-10-25	. . .
Jennifer	Wallace	987654321	333445555	Joy	F	1958-05-03	. . .
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	. . .
Jennifer	Wallace	987654321	123456789	Michael	M	1988-01-04	. . .
Jennifer	Wallace	987654321	123456789	Alice	F	1988-12-30	. . .
Jennifer	Wallace	987654321	123456789	Elizabeth	F	1967-05-05	. . .
Joyce	English	453453453	333445555	Alice	F	1986-04-05	. . .
Joyce	English	453453453	333445555	Theodore	M	1983-10-25	. . .
Joyce	English	453453453	333445555	Joy	F	1958-05-03	. . .
Joyce	English	453453453	987654321	Abner	M	1942-02-28	. . .
Joyce	English	453453453	123456789	Michael	M	1988-01-04	. . .
Joyce	English	453453453	123456789	Alice	F	1988-12-30	. . .
Joyce	English	453453453	123456789	Elizabeth	F	1967-05-05	. . .

ACTUAL_DEPENDENTS

Fname	Lname	Ssn	Essn	Dependent_name	Sex	Bdate	. . .
Jennifer	Wallace	987654321	987654321	Abner	M	1942-02-28	. . .

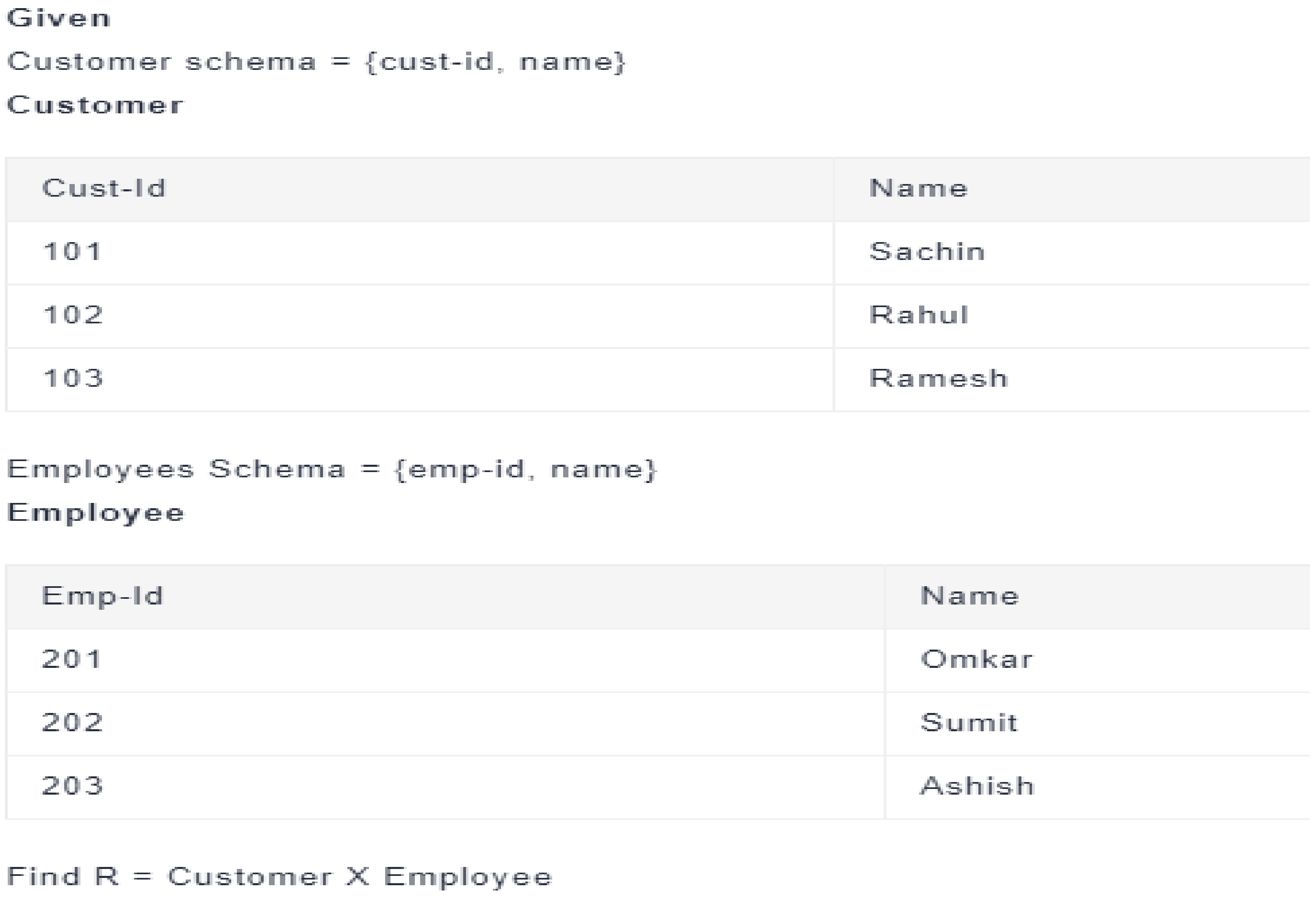
RESULT

Fname	Lname	Dependent_name
Jennifer	Wallace	Abner

Relational Algebra Operations from Set Theory: **CARTESIAN PRODUCT** (cont.)

- To keep only combinations where the DEPENDENT is related to the EMPLOYEE, we add a SELECT operation as follows
- Example (meaningful):
  - FEMALE_EMPS $\leftarrow \sigma_{\text{SEX='F'}}$(EMPLOYEE)
  - EMPNAMES $\leftarrow \pi_{\text{FNAME, LNAME, SSN}}$ (FEMALE_EMPS)
  - EMP_DEPENDENTS $\leftarrow$ EMPNAMES x DEPENDENT
  - ACTUAL_DEPS $\leftarrow \sigma_{\text{SSN=ESSN}}$(EMP_DEPENDENTS)
  - RESULT $\leftarrow \pi_{\text{FNAME, LNAME, DEPENDENT\_NAME}}$ (ACTUAL_DEPS)
- RESULT will now contain the name of female employees and their dependents

Exercise: Cartesian Product



Given

Customer schema = {cust-id, name}

Customer

Cust-Id	Name
101	Sachin
102	Rahul
103	Ramesh

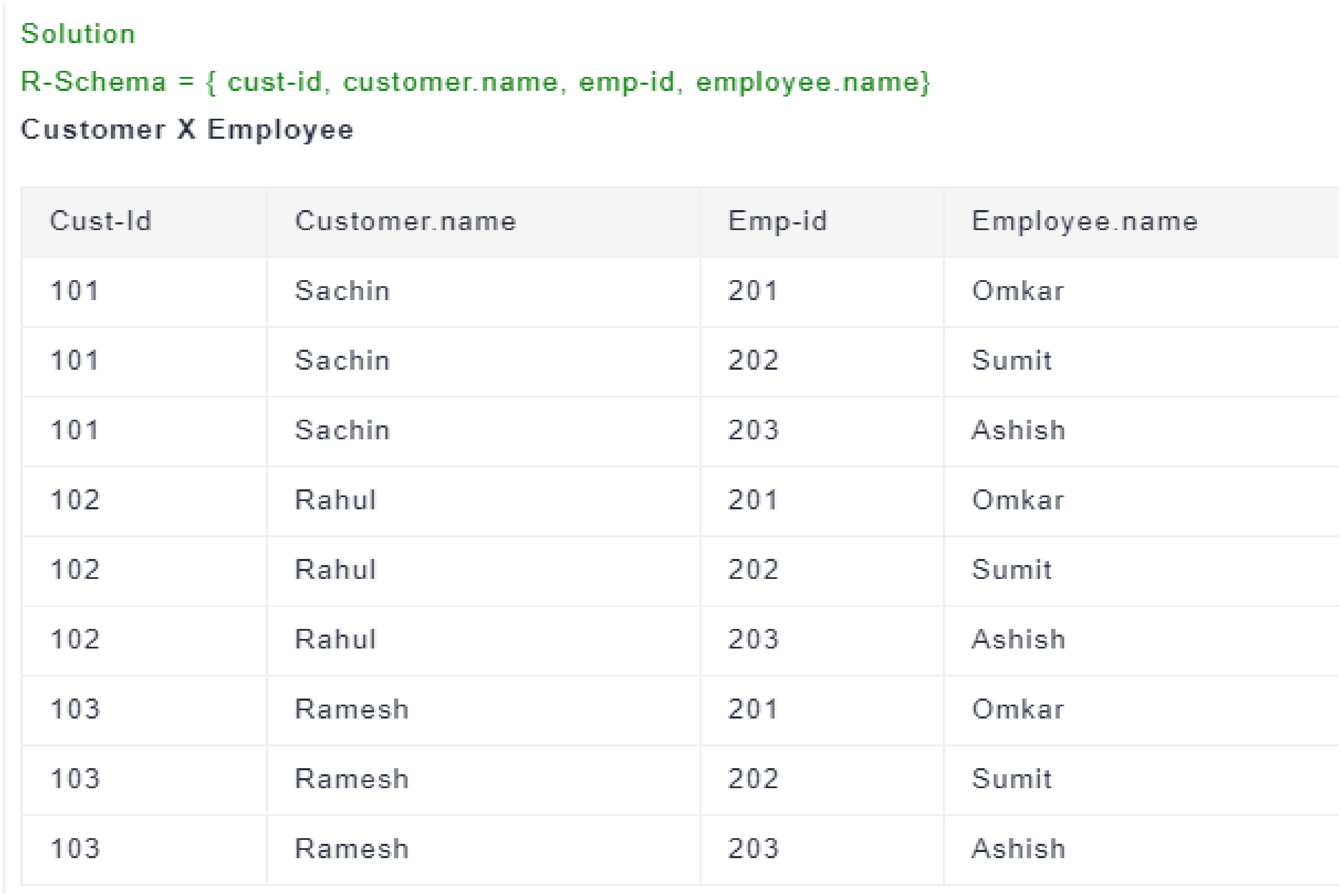
Employees Schema = {emp-id, name}

Employee

Emp-Id	Name
201	Omkar
202	Sumit
203	Ashish

Find R = Customer X Employee

Solution



Solution

R-Schema = { cust-id, customer.name, emp-id, employee.name}

Customer X Employee

Cust-Id	Customer.name	Emp-id	Employee.name
101	Sachin	201	Omkar
101	Sachin	202	Sumit
101	Sachin	203	Ashish
102	Rahul	201	Omkar
102	Rahul	202	Sumit
102	Rahul	203	Ashish
103	Ramesh	201	Omkar
103	Ramesh	202	Sumit
103	Ramesh	203	Ashish

Binary Relational Operations: JOIN

- **JOIN Operation** (denoted by $\bowtie$)
  - The sequence of CARTESIAN PRODUCT followed by SELECT is used quite commonly to identify and select related tuples from two relations
  - A special operation, called JOIN combines this sequence into a single operation
  - This operation is very important for any relational database with more than a single relation, because it allows us *combine related tuples* from various relations
  - The general form of a join operation on two relations R(A1, A2, . . ., An) and S(B1, B2, . . ., Bm) is:

$$R \bowtie_{\text{<join condition>}} S$$

  - where R and S can be any relations that result from general *relational algebra expressions*.

Binary Relational Operations: **JOIN** (contd..)

- Example:
- Retrieve the name of the manager of each department.
  - To get the manager's name, we need to combine each DEPARTMENT tuple with the EMPLOYEE tuple whose SSN value matches the MGRSSN value in the department tuple.
  - We do this by using the join $\bowtie$ operation.

  - DEPT_MGR $\leftarrow$ DEPARTMENT $\bowtie_{\text{MGRSSN=SSN}}$ EMPLOYEE

- MGRSSN=SSN is the join condition
  - Combines each department record with the employee who manages the department
  - The join condition can also be specified as DEPARTMENT.MGRSSN= EMPLOYEE.SSN

Database state

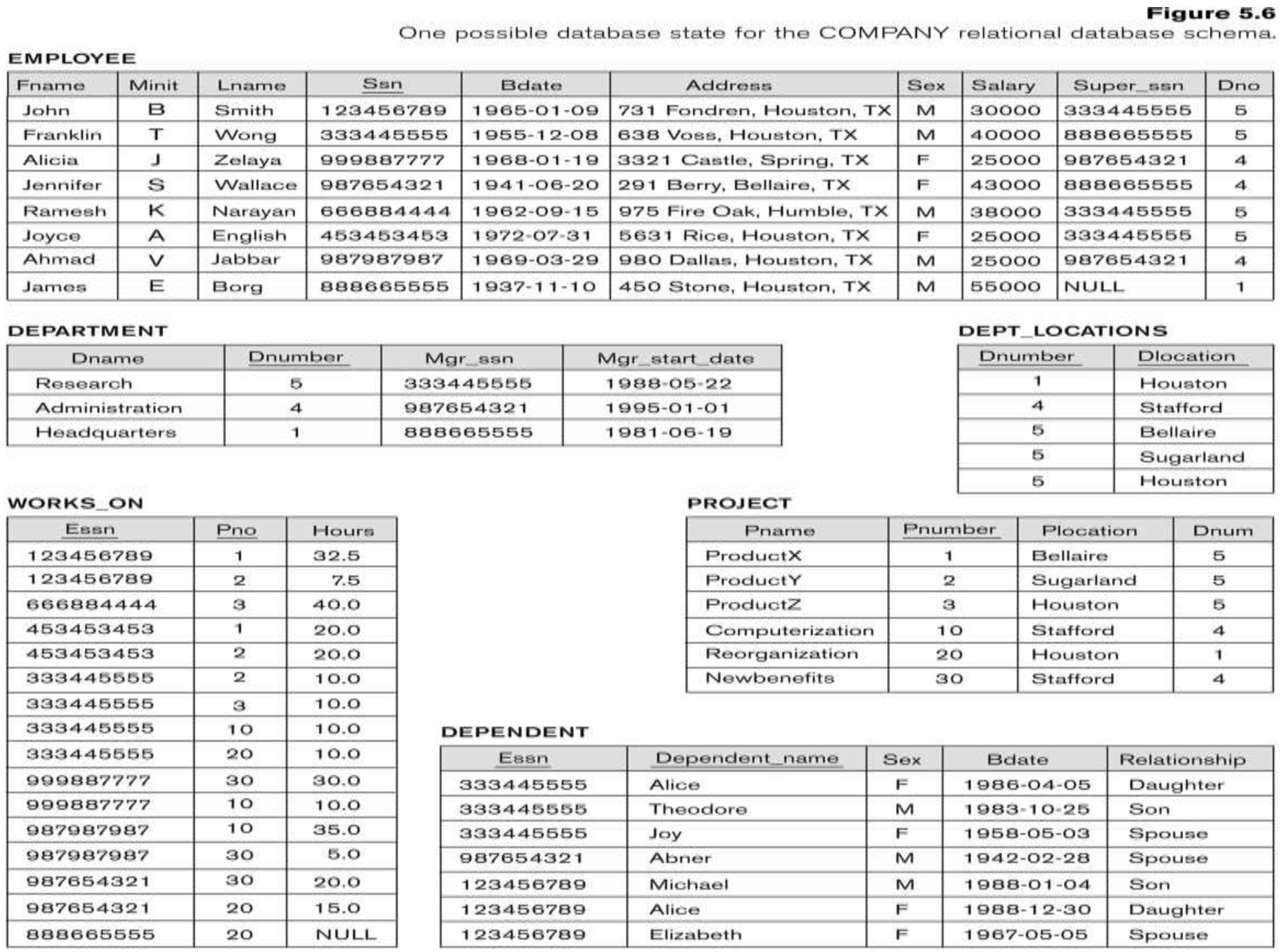


Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

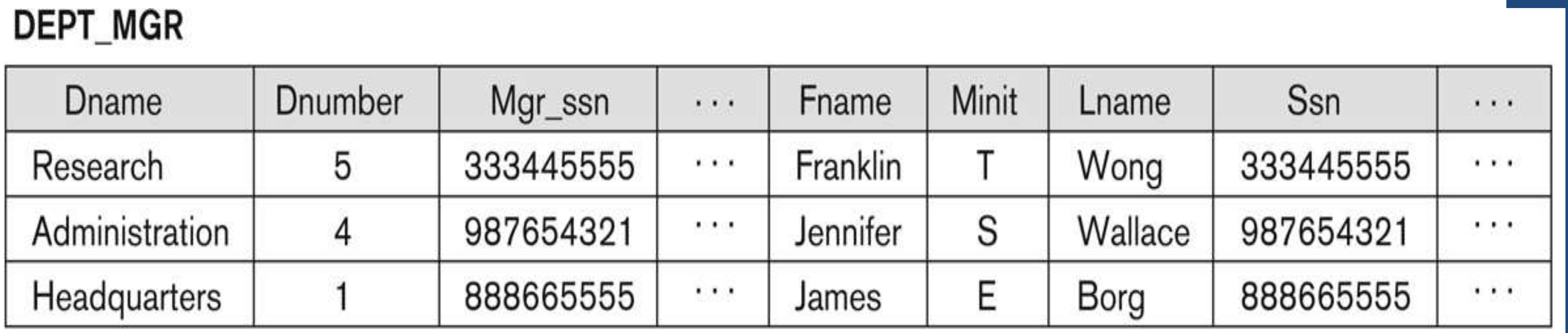
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Example : **JOIN** operation



DEPT_MGR

Dname	Dnumber	Mgr_ssn	. . .	Fname	Minit	Lname	Ssn	. . .
Research	5	333445555	. . .	Franklin	T	Wong	333445555	. . .
Administration	4	987654321	. . .	Jennifer	S	Wallace	987654321	. . .
Headquarters	1	888665555	. . .	James	E	Borg	888665555	. . .

Figure 6.6
Result of the JOIN operation

Some properties of JOIN

- Consider the following JOIN operation:
  - R(A1, A2, . . ., An) $\underset{R.Ai=S.Bj}{\bowtie}$ S(B1, B2, . . ., Bm)
  - Result is a relation Q with degree n + m attributes:
    - Q(A1, A2, . . ., An, B1, B2, . . ., Bm), in that order.
  - The resulting relation state has one tuple for each combination of tuples—r from R and s from S, but *only if they* ***satisfy the join condition*** **r[Ai]=s[Bj]**
  - Hence, if R has n_R tuples, and S has n_S tuples, then the join result will generally have *less than* $n_R * n_S$ tuples.
  - Only related tuples (based on the join condition) will appear in the result

Some properties of JOIN

- The general case of JOIN operation is called a Theta-join: R ***theta*** S
- The join condition is called *theta*
- *Theta* can be any general boolean expression on the attributes of R and S; for example:
  - R.Ai<S.Bj AND (R.Ak=S.Bl OR R.Ap<S.Bq)
- Most join conditions involve one or more equality conditions "AND"ed together; for example:
  - R.Ai=S.Bj AND R.Ak=S.Bl AND R.Ap=S.Bq

Binary Relational Operations: **EQUIJOIN**

- The most common use of join involves join conditions with *equality comparisons* only
- Such a join, where the only comparison operator used is =, is called an EQUIJOIN.
  - In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have identical values in every tuple.
  - The JOIN seen in the previous example was an EQUIJOIN.

Binary Relational Operations: **NATURAL JOIN Operation**

- NATURAL JOIN Operation
  - Another variation of JOIN called NATURAL JOIN — denoted by * — was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
    - because one of each pair of attributes with identical values is superfluous
  - The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, *have the* ***same name*** **in both relations**
  - If this is not the case, a renaming operation is applied first.

Binary Relational Operations **NATURAL JOIN** (contd.)

- Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:
  - **DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS**
- Only attribute with the same name is DNUMBER
- An implicit join condition is created based on this attribute:

DEPARTMENT.DNUMBER=DEPT_LOCATIONS.DNUMBER

- Another example: Q ← R(A,B,C,D) * S(C,D,E)
  - Result keeps only one attribute of each such pair:
    - Q(A,B,C,D,E)

Database state

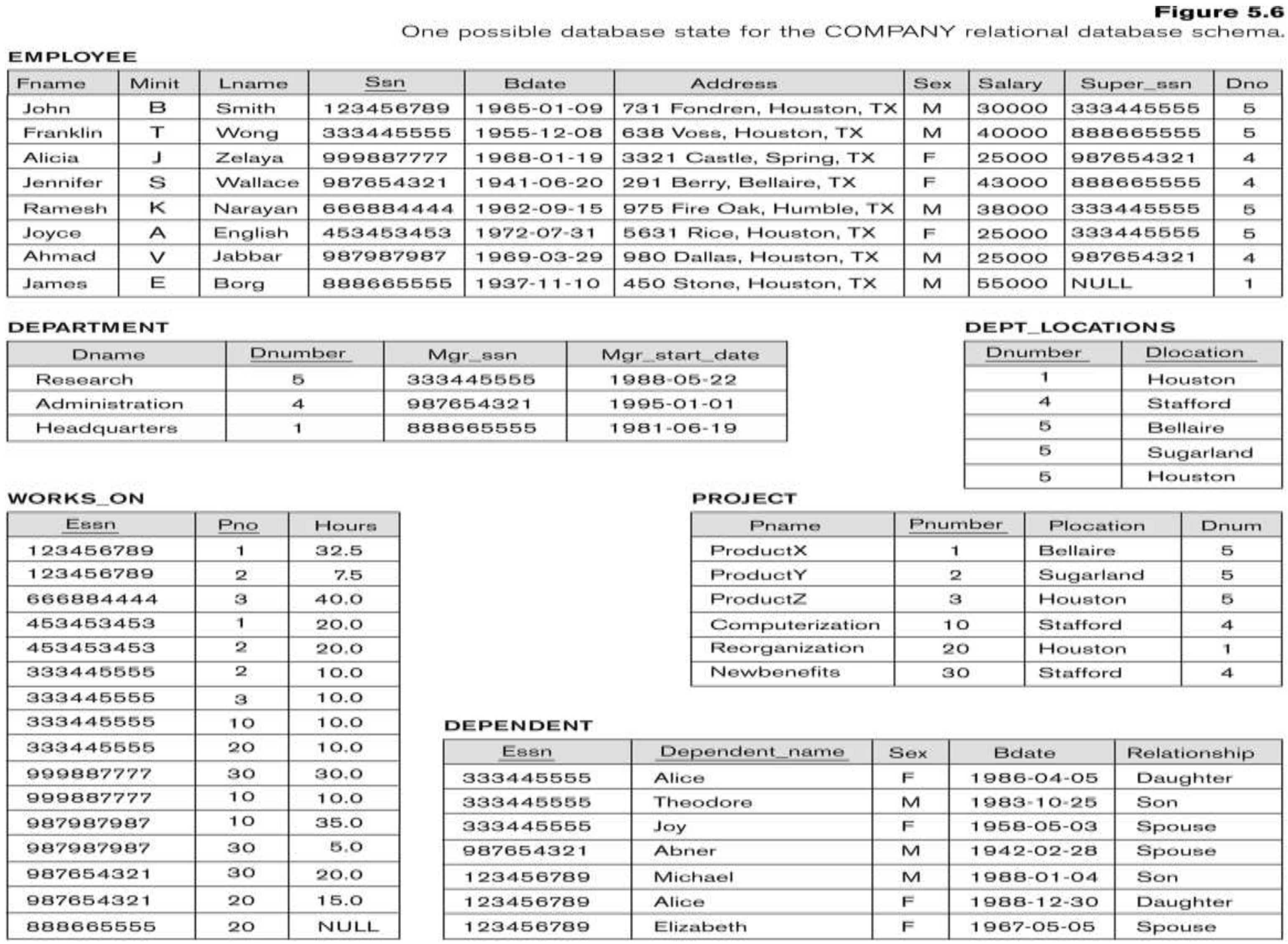


Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

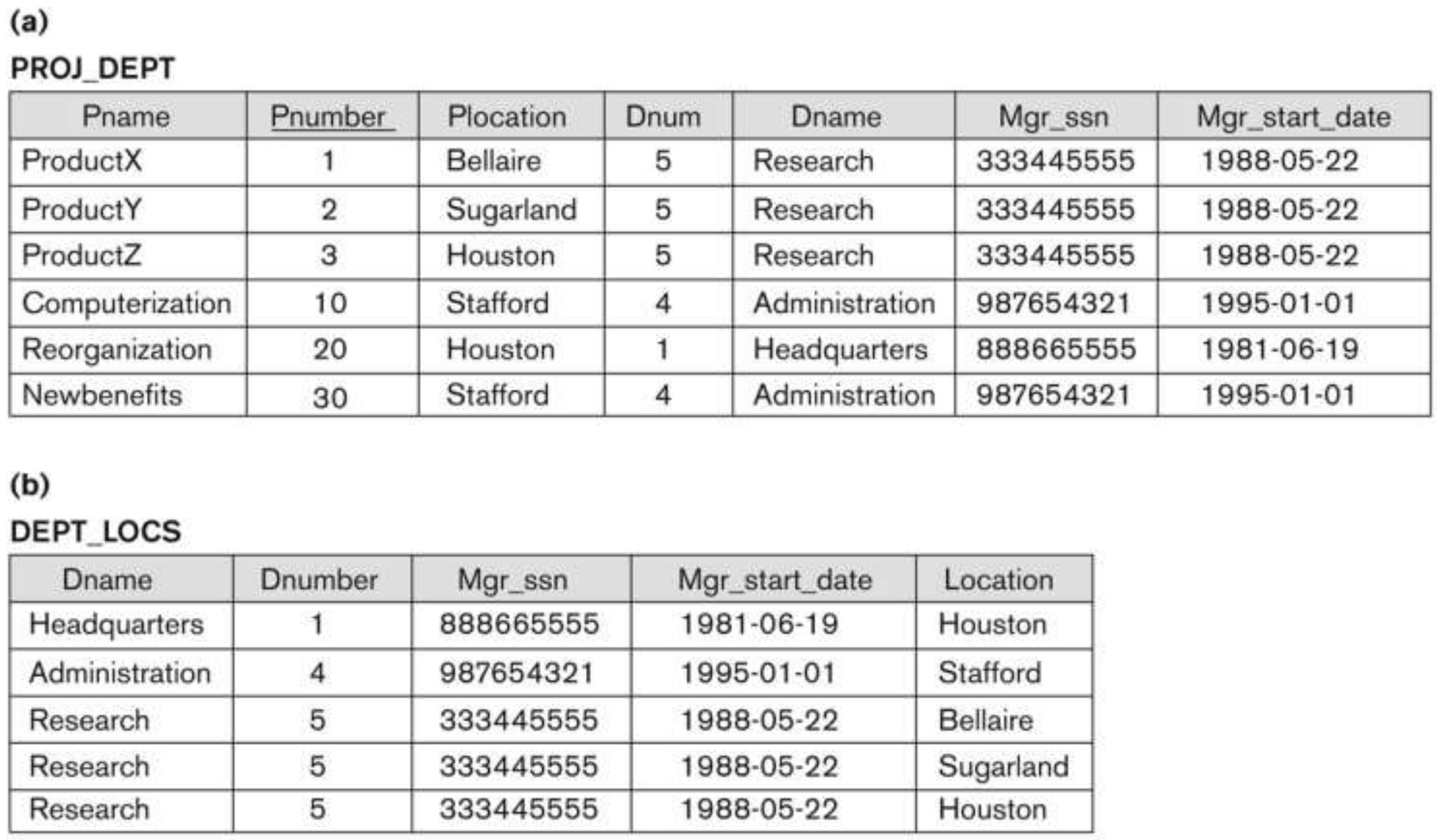
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

Example of NATURAL JOIN operation



(a)

PROJ_DEPT

Pname	Pnumber	Plocation	Dnum	Dname	Mgr_ssn	Mgr_start_date
ProductX	1	Bellaire	5	Research	333445555	1988-05-22
ProductY	2	Sugarland	5	Research	333445555	1988-05-22
ProductZ	3	Houston	5	Research	333445555	1988-05-22
Computerization	10	Stafford	4	Administration	987654321	1995-01-01
Reorganization	20	Houston	1	Headquarters	888665555	1981-06-19
Newbenefits	30	Stafford	4	Administration	987654321	1995-01-01

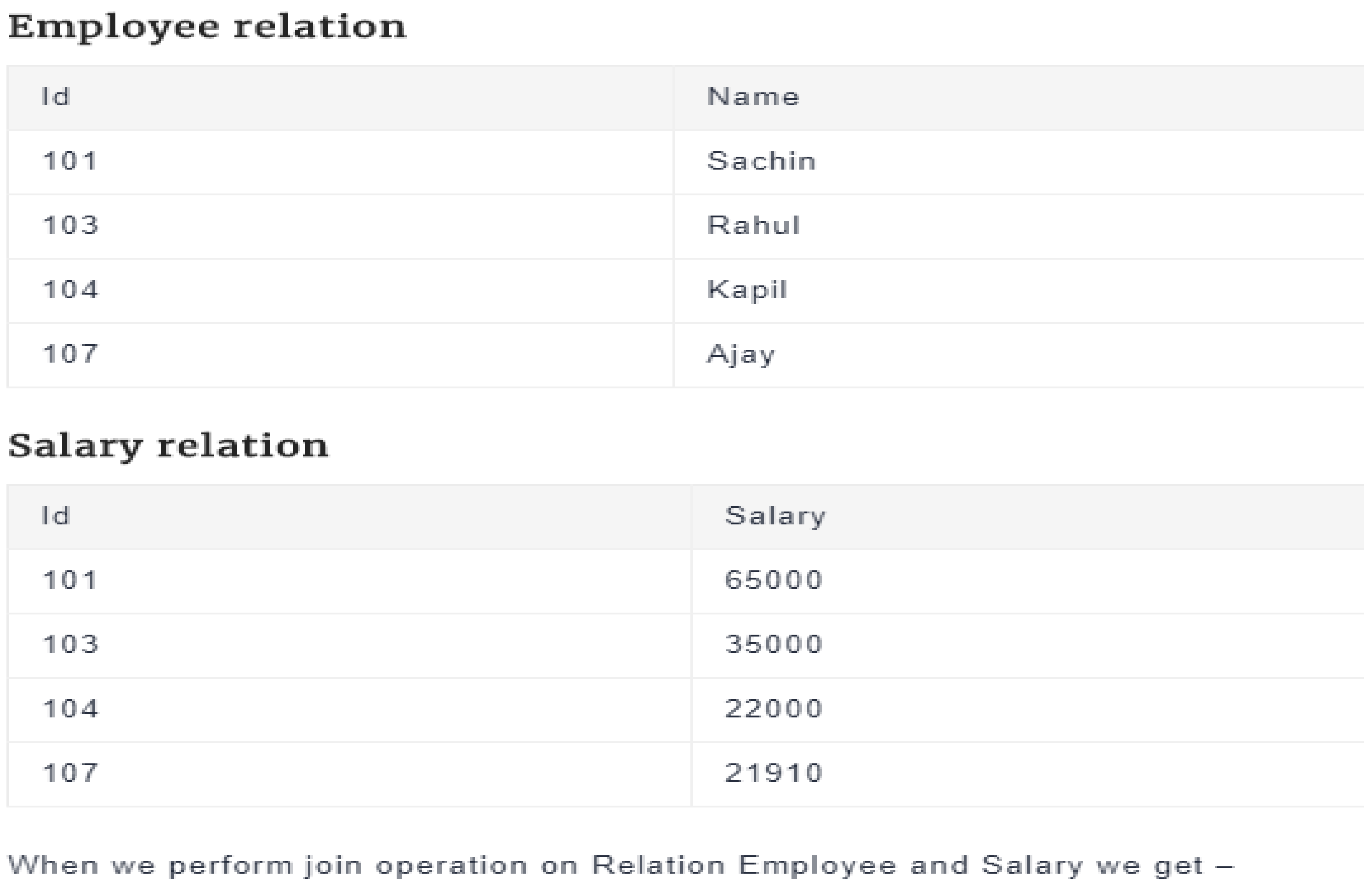
(b)

DEPT_LOCS

Dname	Dnumber	Mgr_ssn	Mgr_start_date	Location
Headquarters	1	888665555	1981-06-19	Houston
Administration	4	987654321	1995-01-01	Stafford
Research	5	333445555	1988-05-22	Bellaire
Research	5	333445555	1988-05-22	Sugarland
Research	5	333445555	1988-05-22	Houston

Figure 6.7
Results of two NATURAL JOIN operations.
(a) PROJ_DEPT ← PROJECT * DEPT.
(b) DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS.

Exercise on Natural Join



Employee relation

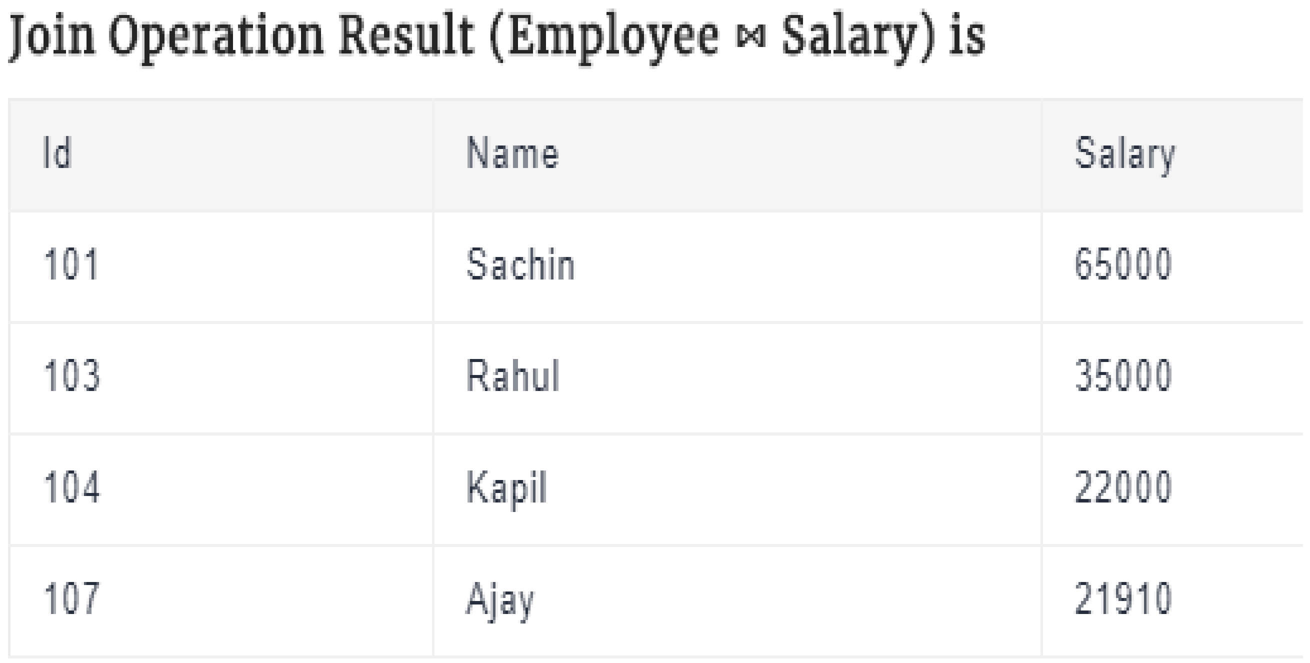
Id	Name
101	Sachin
103	Rahul
104	Kapil
107	Ajay

Salary relation

Id	Salary
101	65000
103	35000
104	22000
107	21910

When we perform join operation on Relation Employee and Salary we get –

Solution



Join Operation Result (Employee ⋈ Salary) is

Id	Name	Salary
101	Sachin	65000
103	Rahul	35000
104	Kapil	22000
107	Ajay	21910

Complete Set of Relational Operations

- The set of operations including SELECT σ, PROJECT π , UNION $\cup$, DIFFERENCE $-$, RENAME ρ, and CARTESIAN PRODUCT X is called a *complete set* because any other relational algebra expression can be expressed by a combination of these five operations.
- For example:
  - $R \cap S = (R \cup S) - ((R - S) \cup (S - R))$
  - $R \bowtie_{\text{<join condition>}} S = \sigma_{\text{<join condition>}} (R \times S)$

Binary Relational Operations: **DIVISION**

- The division operation is applied to two relations
- Attributes of S is **proper subset** of Attributes of R.
- The relation returned by division operator will have attributes = (All attributes of R – All Attributes of S)
- The relation returned by division operator will return those tuples from relation R which are **associated to every S's tuple.**

Example of DIVISION

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

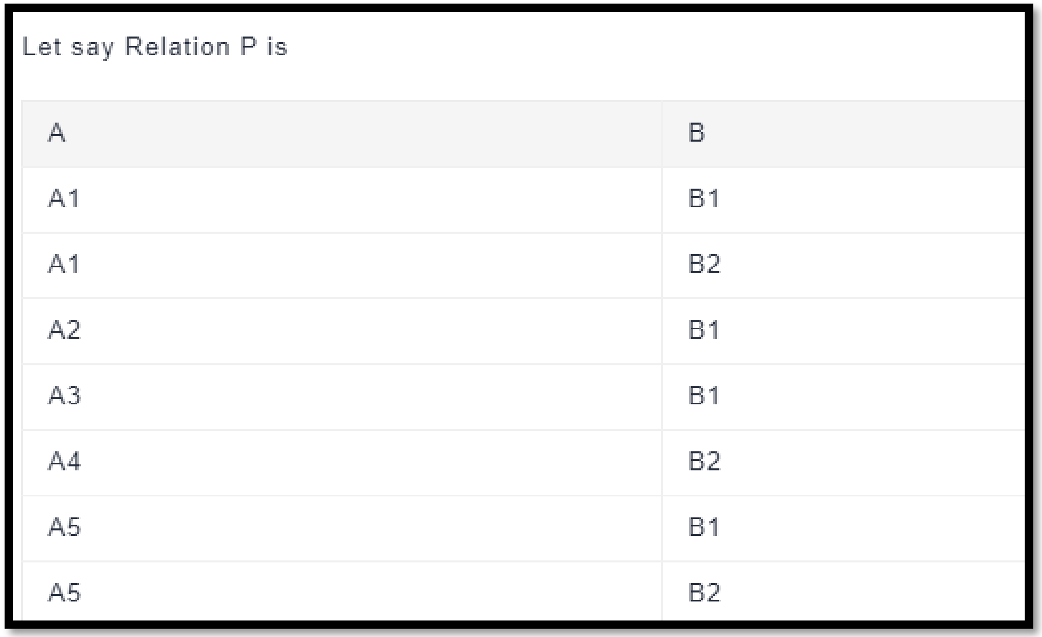
A
a1
a2
a3

T

B
b1
b4

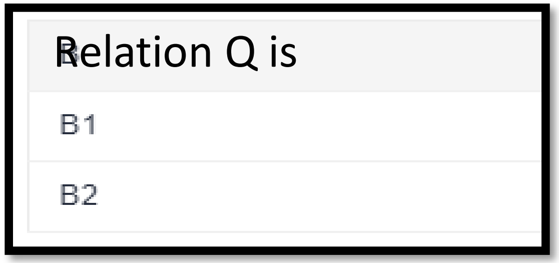
Figure 6.8
The DIVISION operation. (a) Dividing SSN_PNOS by SMITH_PNOS. (b) $T \leftarrow R \div S$.

Exercise: Division Operator



Let say Relation P is

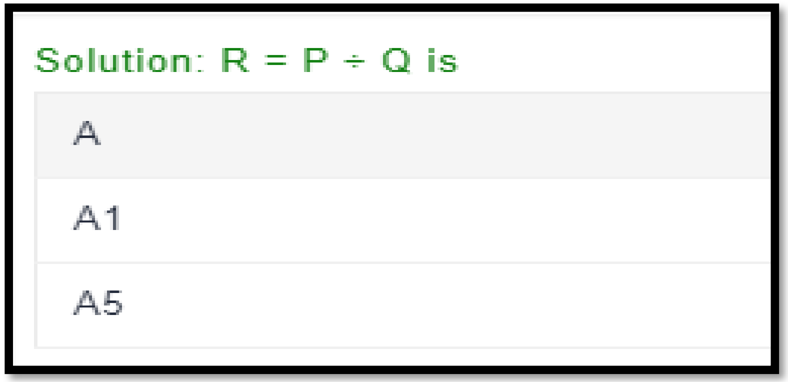
A	B
A1	B1
A1	B2
A2	B1
A3	B1
A4	B2
A5	B1
A5	B2



Relation Q is

B
B1
B2

Find P ÷ Q if Q is



Solution: R = P ÷ Q is

A
A1
A5

Summary of Relational Algebra Operations

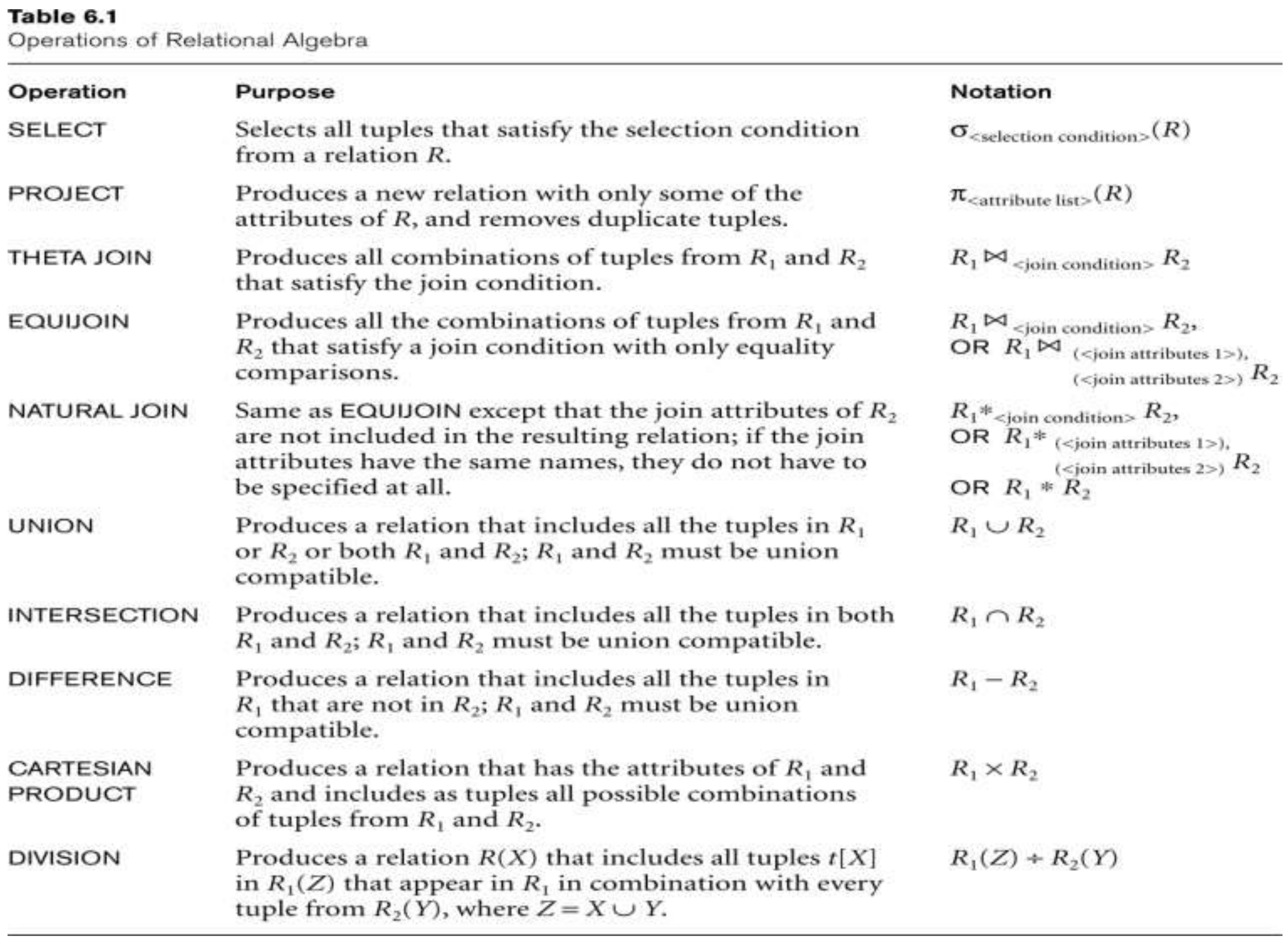


Table 6.1
Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{<\text{selection condition}>}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{<\text{attribute list}>}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{<\text{join condition}>} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{<\text{join condition}>} R_2$, OR $R_1 \bowtie_{(<\text{join attributes 1}>),\ (<\text{join attributes 2}>)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{<\text{join condition}>} R_2$, OR $R_1 *_{(<\text{join attributes 1}>),\ (<\text{join attributes 2}>)} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Additional Relational Operations: **Aggregate Functions and Grouping**

- A type of request that cannot be expressed in the basic relational algebra is to specify mathematical **aggregate functions** on collections of values from the database.
- Examples :
  - Retrieving the **average or total salary** of all employees or the **total number of employee** tuples.
  - Used in simple statistical queries that summarize information from the database tuples.
- Common functions applied to collections of numeric values include
  - **SUM, AVERAGE, MAXIMUM, and MINIMUM.**
- The COUNT function is used for counting tuples or values.

Aggregate Function Operation

- Use of the Aggregate Functional operation $\mathcal{F}$
  - $\mathcal{F}_{\text{MAX Salary}}$ **(EMPLOYEE)** retrieves the maximum salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{MIN Salary}}$ **(EMPLOYEE)** retrieves the minimum Salary value from the EMPLOYEE relation
  - $\mathcal{F}_{\text{SUM Salary}}$ **(EMPLOYEE)** retrieves the sum of the Salary from the EMPLOYEE relation
  - $\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ **(EMPLOYEE)** computes the count (number) of employees and their average salary
    - **Note:** count just counts the number of rows, without removing duplicates

Using Grouping with Aggregation

- Grouping can be combined with Aggregate Functions
- Example:
  - For each department, retrieve the DNO, COUNT SSN, and AVERAGE SALARY
- A variation of aggregate operation $\mathcal{F}$ allows this:
  - Grouping attribute placed to left of symbol
  - Aggregate functions to right of symbol
  - $_{\text{DNO}}\,\mathcal{F}_{\text{COUNT SSN, AVERAGE Salary}}$ **(EMPLOYEE)**
- Above operation groups employees by DNO (department number) and computes the count of employees and average salary per department

Examples of applying aggregate functions and grouping

Figure 6.10
The aggregate function operation.
(a) $\rho_{R(\text{Dno, No\_of\_employees, Average\_sal})}({}_{\text{Dno}}\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE}))$.
(b) ${}_{\text{Dno}}\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.
(c) $\Im_{\text{COUNT Ssn, AVERAGE Salary}}(\text{EMPLOYEE})$.

R

(a)

Dno	No_of_employees	Average_sal
5	4	33250
4	3	31000
1	1	55000

(b)

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

(c)

Count_ssn	Average_salary
8	35125

Additional Relational Operations (cont.)

The OUTER JOIN Operation

- In NATURAL JOIN and EQUIJOIN, tuples without a *matching* (or *related*) tuple are eliminated from the join result
  - Tuples with null in the join attributes are also eliminated
  - This amounts to loss of information.
- A set of operations, called OUTER joins, can be **used when we want to keep all the tuples in R, or all those in S, or all those in both relations in the result of the join**, regardless of whether or not they have matching tuples in the other relation.

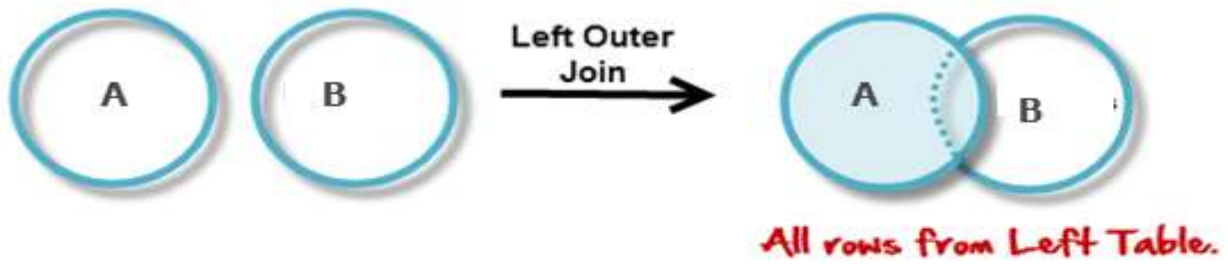
Additional Relational Operations (cont.)

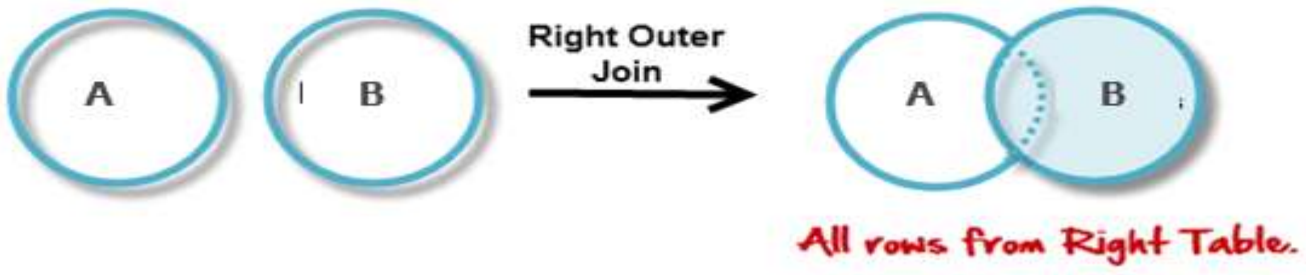
- The **left outer join** operation keeps every tuple in the first or left relation R in R ⟕ S; if no matching tuple is found in S, then the attributes of S in the join result are filled or “padded” with null values.

- A similar operation, **right outer join**, keeps every tuple in the second or right relation S in the result of R ⟖ S.

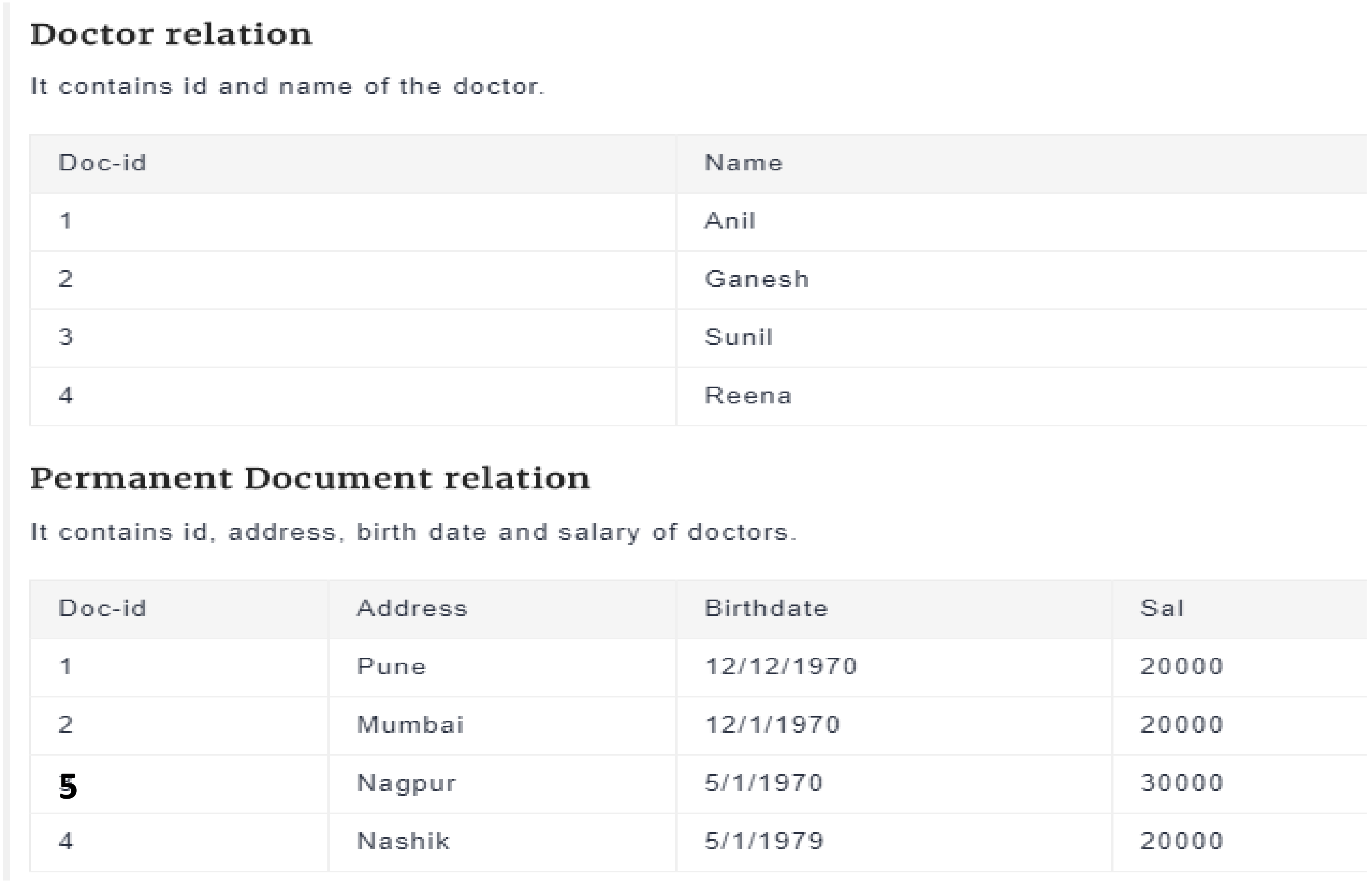
- A third operation, **full outer join**, denoted by ⟗ keeps all tuples in both the left and the right relations when no matching tuples are found, padding them with null values as needed.

Left and Right Outer Join





Example: Outer Join



Doctor relation

It contains id and name of the doctor.

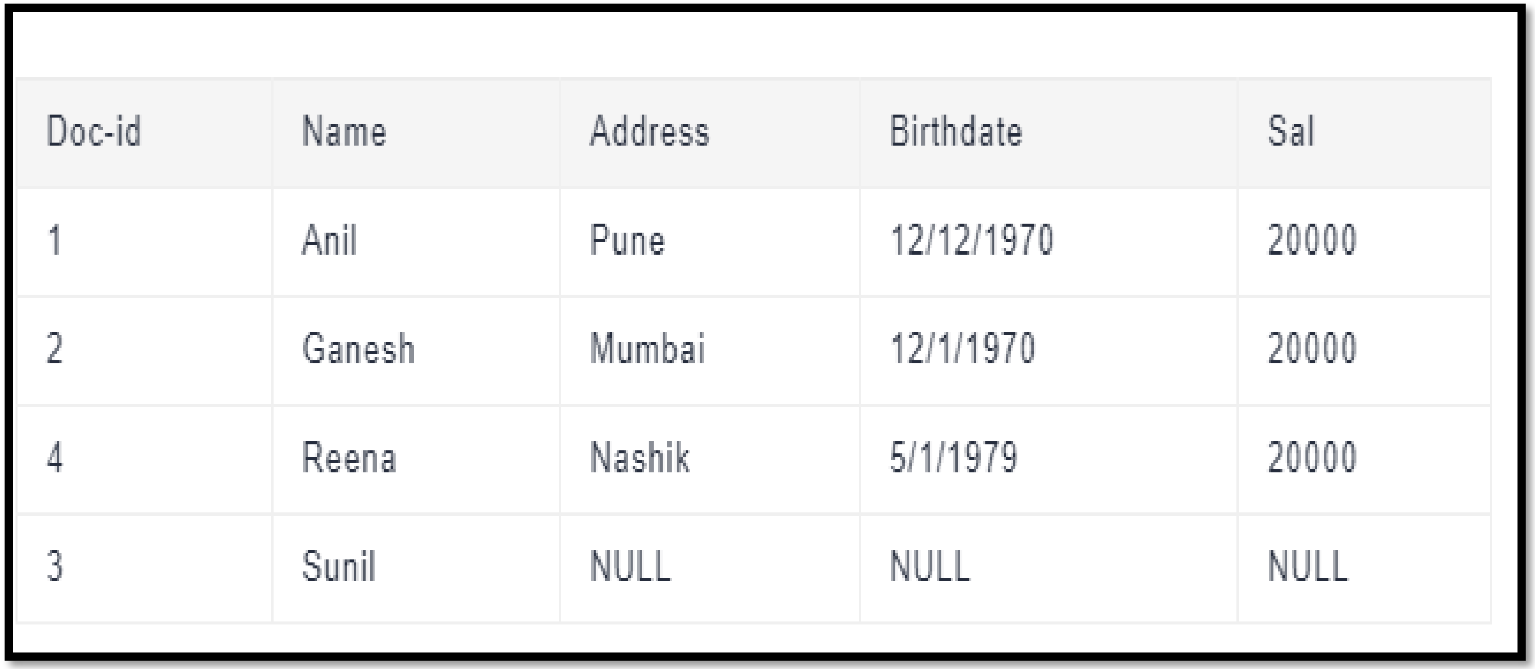
Doc-id	Name
1	Anil
2	Ganesh
3	Sunil
4	Reena

Permanent Document relation

It contains id, address, birth date and salary of doctors.

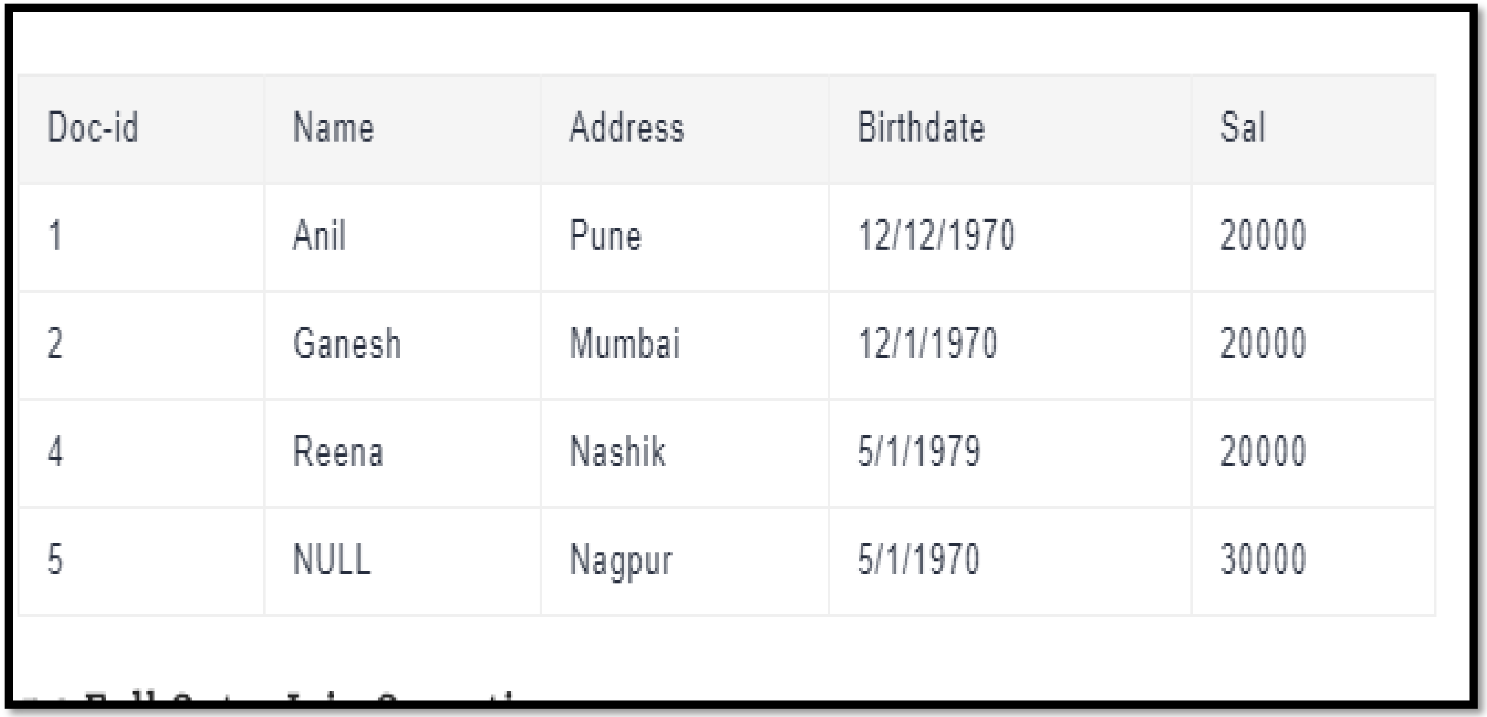
Doc-id	Address	Birthdate	Sal
1	Pune	12/12/1970	20000
2	Mumbai	12/1/1970	20000
5	Nagpur	5/1/1970	30000
4	Nashik	5/1/1979	20000

Left Outer Join



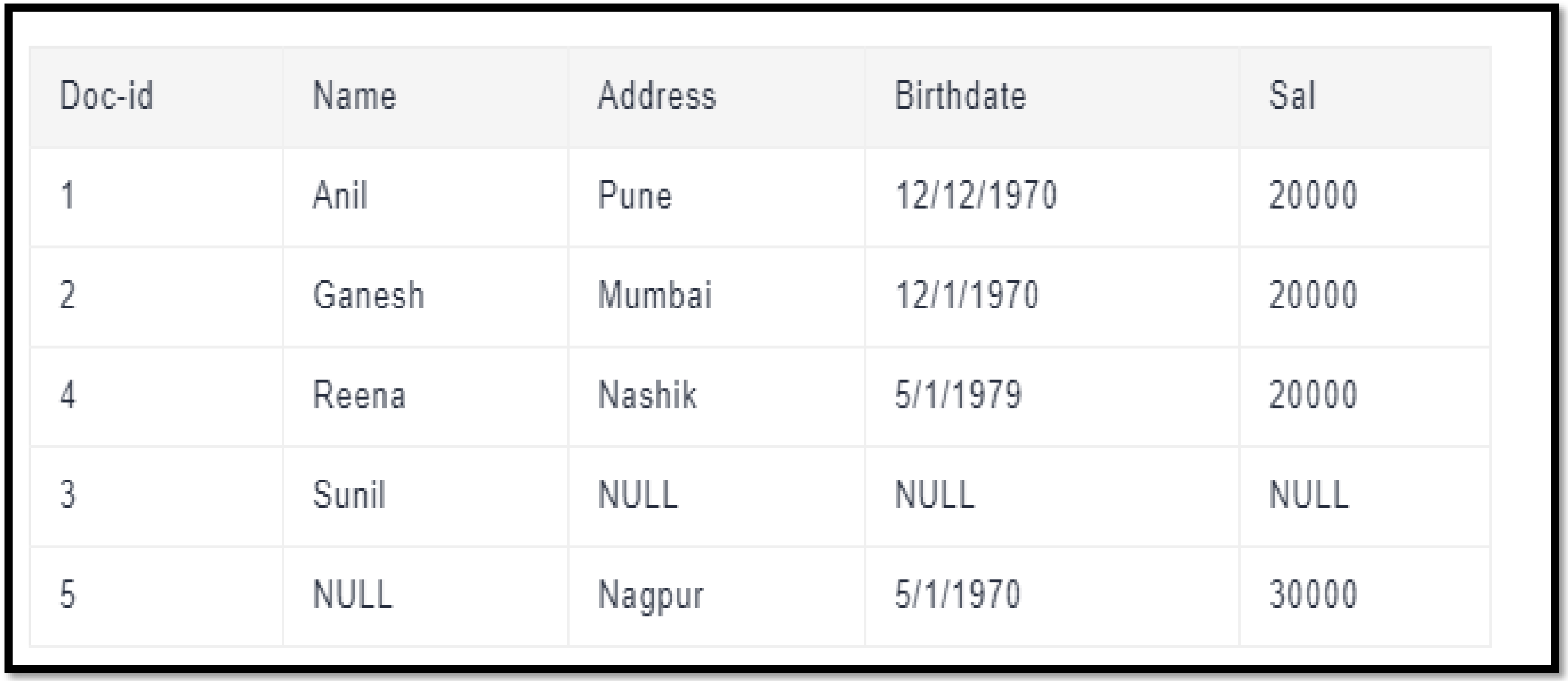
Doc-id	Name	Address	Birthdate	Sal
1	Anil	Pune	12/12/1970	20000
2	Ganesh	Mumbai	12/1/1970	20000
4	Reena	Nashik	5/1/1979	20000
3	Sunil	NULL	NULL	NULL

Right Outer Join



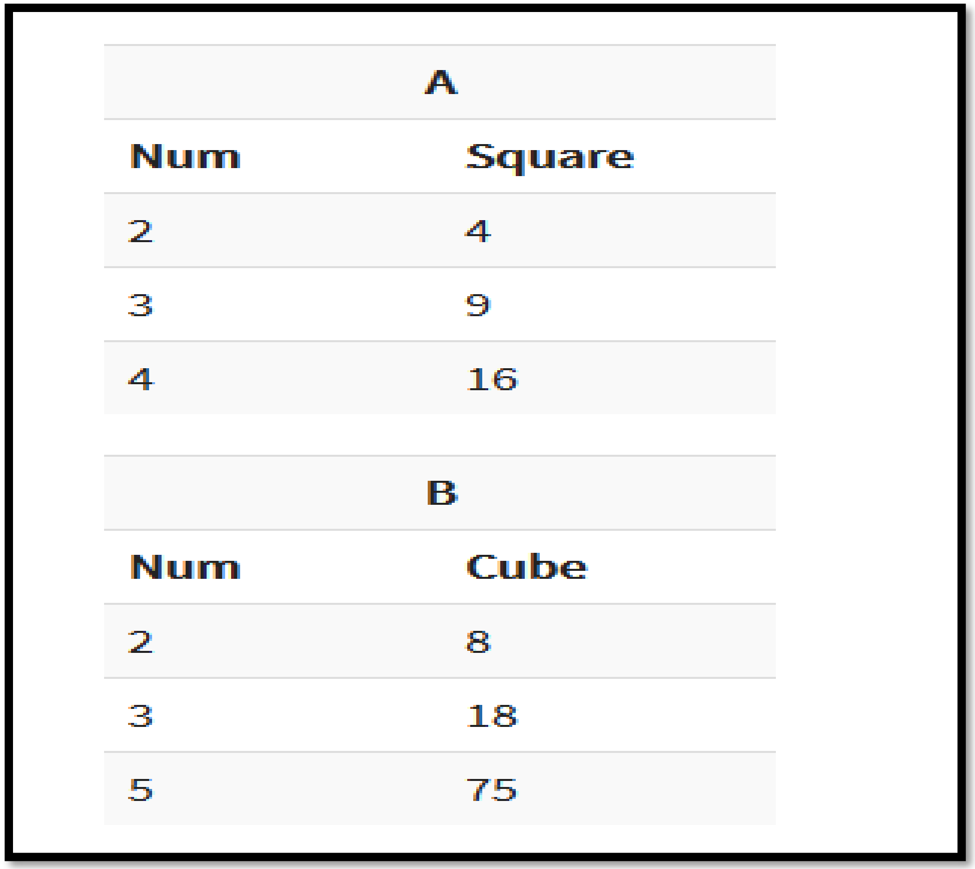
Doc-id	Name	Address	Birthdate	Sal
1	Anil	Pune	12/12/1970	20000
2	Ganesh	Mumbai	12/1/1970	20000
4	Reena	Nashik	5/1/1979	20000
5	NULL	Nagpur	5/1/1970	30000

Full Outer Join



Doc-id	Name	Address	Birthdate	Sal
1	Anil	Pune	12/12/1970	20000
2	Ganesh	Mumbai	12/1/1970	20000
4	Reena	Nashik	5/1/1979	20000
3	Sunil	NULL	NULL	NULL
5	NULL	Nagpur	5/1/1970	30000

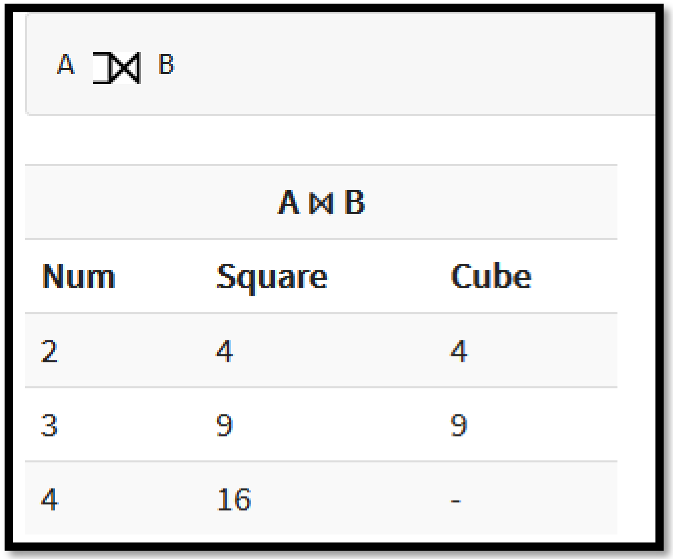
Exercise: Right, Left and Full Outer Join



A	
Num	Square
2	4
3	9
4	16

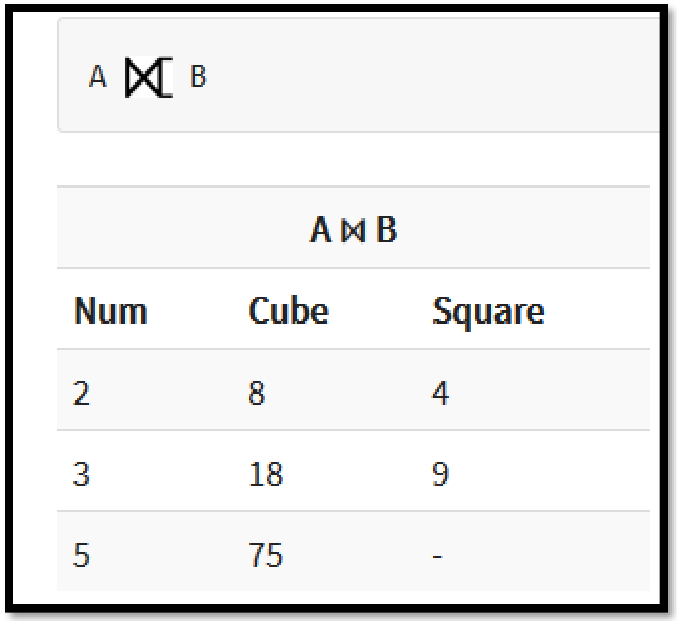
B	
Num	Cube
2	8
3	18
5	75

Solution



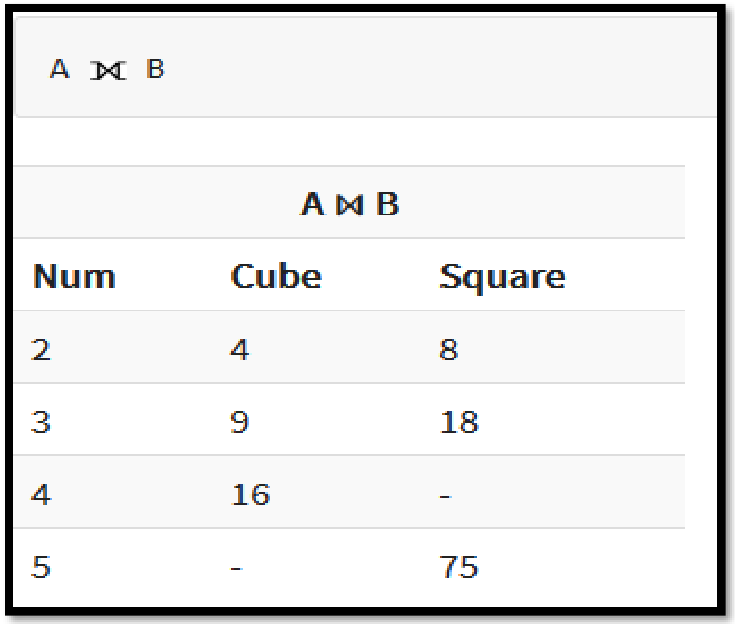
A ⟕ B

A ⋈ B		
Num	Square	Cube
2	4	4
3	9	9
4	16	-



A ⟖ B

A ⋈ B		
Num	Cube	Square
2	8	4
3	18	9
5	75	-



A ⟗ B

A ⋈ B		
Num	Cube	Square
2	4	8
3	9	18
4	16	-
5	-	75

Additional Relational Operations (contd.)

- **Example:** An outer union can be applied to two relations whose schemas are STUDENT(Name, SSN, Department, Advisor) and INSTRUCTOR(Name, SSN, Department, Rank).
  - Tuples from the two relations are matched based on having the same combination of values of the shared attributes—Name, SSN, Department.
  - If a student is also an instructor, both Advisor and Rank will have a value; otherwise, one of these two attributes will be null.
  - The result relation STUDENT_OR_INSTRUCTOR will have the following attributes:

STUDENT_OR_INSTRUCTOR (Name, SSN, Department, Advisor, Rank)

Examples of Queries in Relational Algebra

- **Q1: Retrieve the name and address of all employees who work for the 'Research' department.**

RESEARCH_DEPT $\leftarrow \sigma_{\text{DNAME='Research'}}$ (DEPARTMENT)

RESEARCH_EMPS $\leftarrow$ (RESEARCH_DEPT $\bowtie_{\text{DNUMBER=DNO}}$ EMPLOYEE)

RESULT $\leftarrow \pi_{\text{FNAME, LNAME, ADDRESS}}$ (RESEARCH_EMPS)

- **Q6: Retrieve the names of employees who have no dependents.**

ALL_EMPS $\leftarrow \pi_{\text{SSN}}$(EMPLOYEE)

EMPS_WITH_DEPS(SSN) $\leftarrow \pi_{\text{ESSN}}$(DEPENDENT)

EMPS_WITHOUT_DEPS $\leftarrow$ (ALL_EMPS - EMPS_WITH_DEPS)

RESULT $\leftarrow \pi_{\text{LNAME, FNAME}}$ (EMPS_WITHOUT_DEPS * EMPLOYEE)

Database state

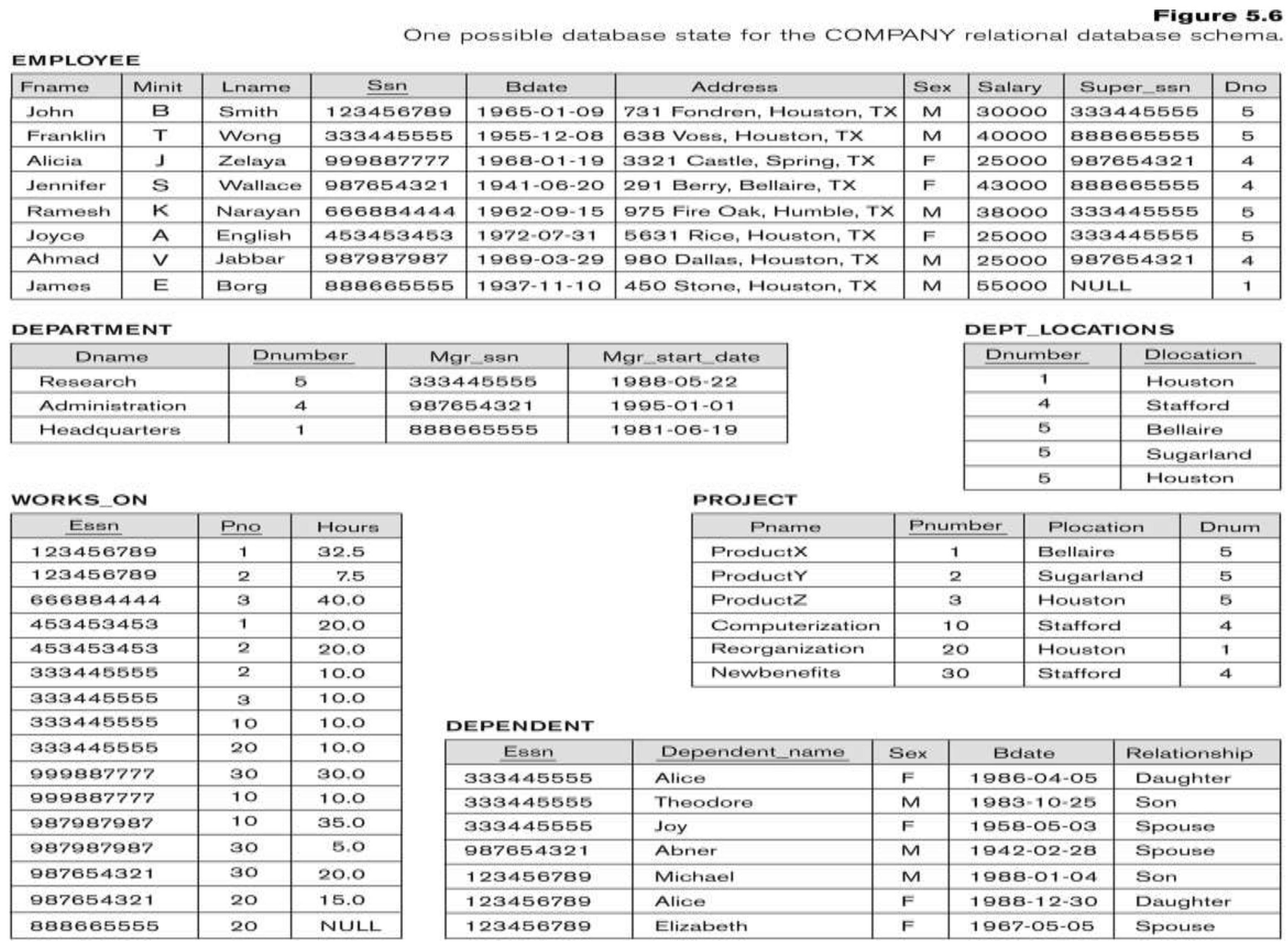


Figure 5.6
One possible database state for the COMPANY relational database schema.

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Essn	Pno	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	NULL

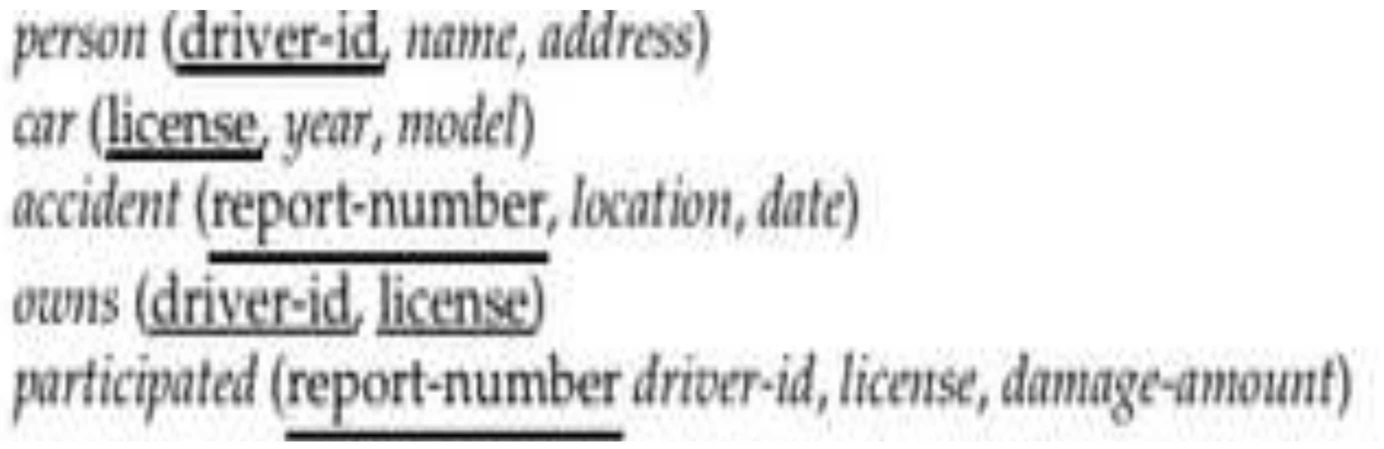
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

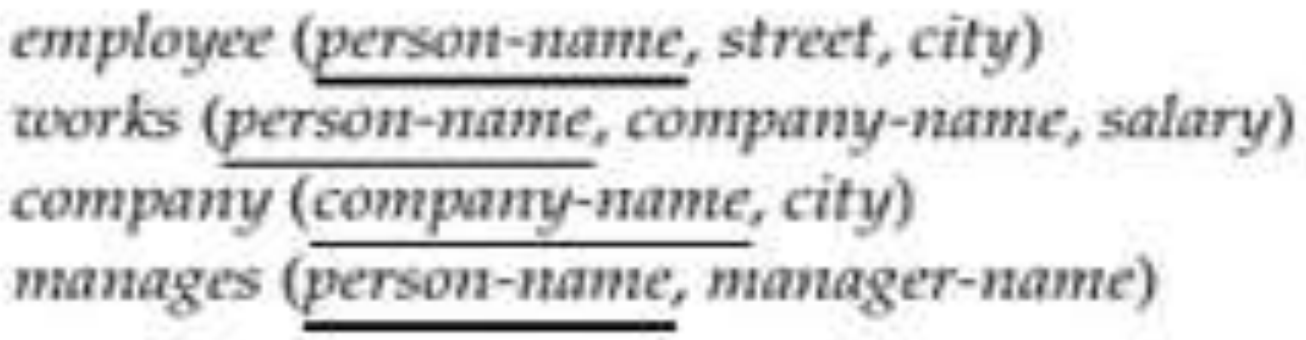
DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
333445555	Theodore	M	1983-10-25	Son
333445555	Joy	F	1958-05-03	Spouse
987654321	Abner	M	1942-02-28	Spouse
123456789	Michael	M	1988-01-04	Son
123456789	Alice	F	1988-12-30	Daughter
123456789	Elizabeth	F	1967-05-05	Spouse

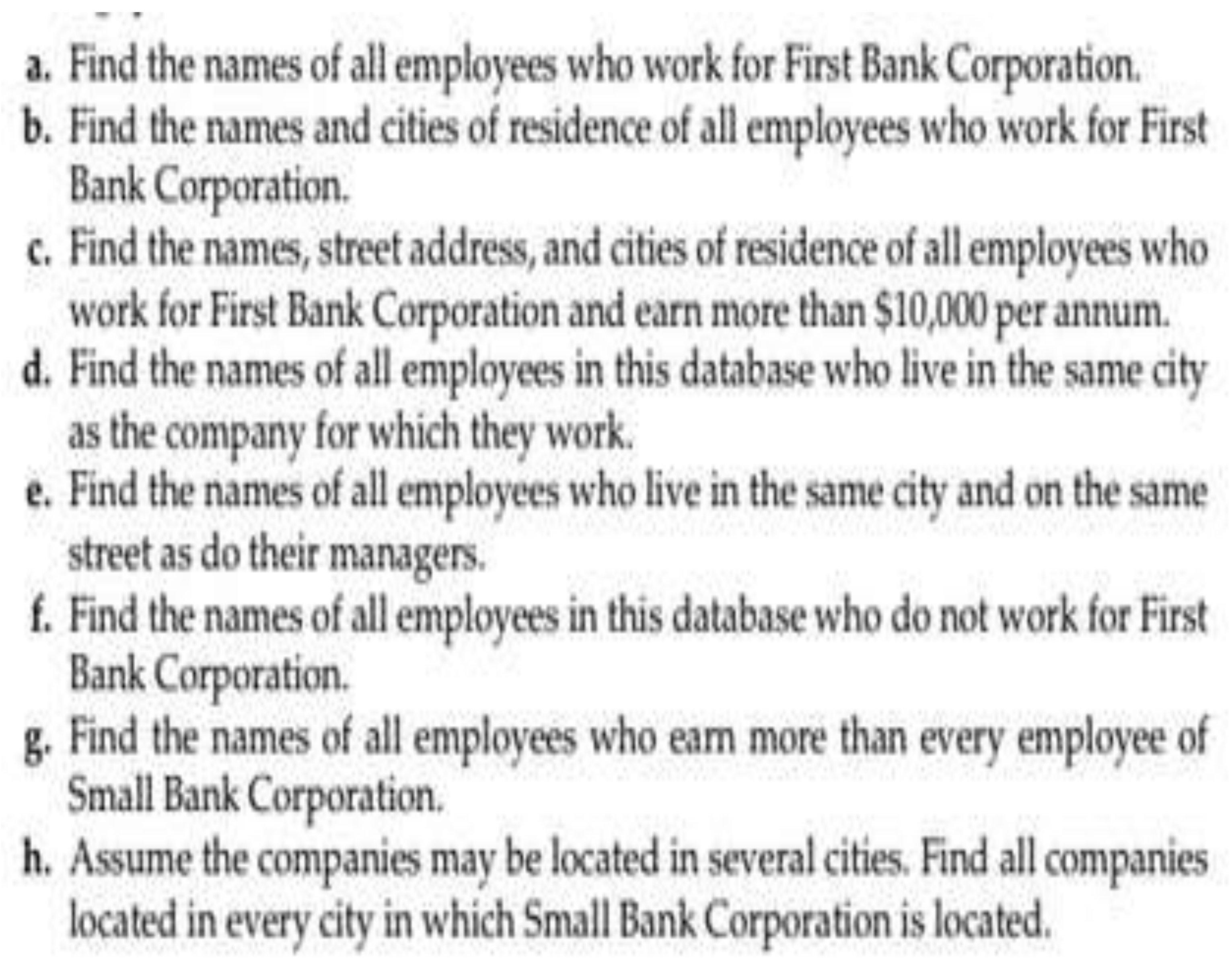
Exercise: 1



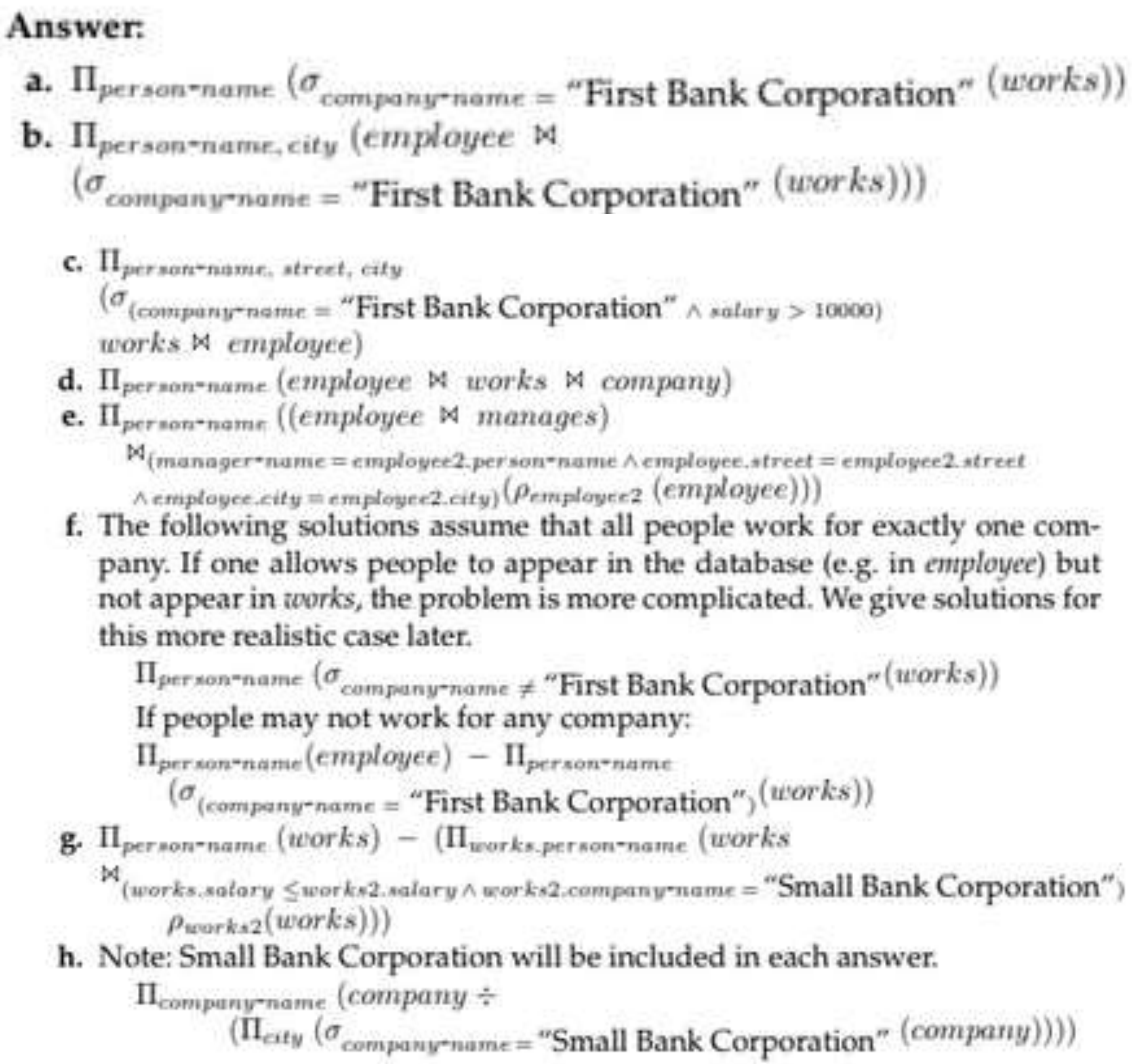
person (driver-id, *name*, *address*)
car (license, *year*, *model*)
accident (report-number, *location*, *date*)
owns (driver-id, license)
participated (report-number *driver-id*, *license*, *damage-amount*)



employee (*person-name*, *street*, *city*)
works (*person-name*, *company-name*, *salary*)
company (*company-name*, *city*)
manages (*person-name*, *manager-name*)



a. Find the names of all employees who work for First Bank Corporation.
b. Find the names and cities of residence of all employees who work for First Bank Corporation.
c. Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000 per annum.
d. Find the names of all employees in this database who live in the same city as the company for which they work.
e. Find the names of all employees who live in the same city and on the same street as do their managers.
f. Find the names of all employees in this database who do not work for First Bank Corporation.
g. Find the names of all employees who earn more than every employee of Small Bank Corporation.
h. Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.



Answer:

a. $\Pi_{person\text{-}name}\ (\sigma_{company\text{-}name\ =\ \text{“First Bank Corporation”}}\ (works))$

b. $\Pi_{person\text{-}name,\ city}\ (employee\ \bowtie$
$(\sigma_{company\text{-}name\ =\ \text{“First Bank Corporation”}}\ (works)))$

c. $\Pi_{person\text{-}name,\ street,\ city}$
$(\sigma_{(company\text{-}name\ =\ \text{“First Bank Corporation”}\ \wedge\ salary\ >\ 10000)}$
$works \bowtie\ employee)$

d. $\Pi_{person\text{-}name}\ (employee\ \bowtie\ works\ \bowtie\ company)$

e. $\Pi_{person\text{-}name}\ ((employee\ \bowtie\ manages)$
$\bowtie_{(manager\text{-}name\ =\ employee2.person\text{-}name\ \wedge\ employee.street\ =\ employee2.street}$
$_{\wedge\ employee.city\ =\ employee2.city)}(\rho_{employee2}\ (employee)))$

f. The following solutions assume that all people work for exactly one company. If one allows people to appear in the database (e.g. in *employee*) but not appear in *works*, the problem is more complicated. We give solutions for this more realistic case later.

$\Pi_{person\text{-}name}\ (\sigma_{company\text{-}name\ \neq\ \text{“First Bank Corporation”}}(works))$

If people may not work for any company:

$\Pi_{person\text{-}name}(employee)\ -\ \Pi_{person\text{-}name}$
$(\sigma_{(company\text{-}name\ =\ \text{“First Bank Corporation”})}(works))$

g. $\Pi_{person\text{-}name}\ (works)\ -\ (\Pi_{works.person\text{-}name}\ (works$
$\bowtie_{(works.salary\ \leq works2.salary\ \wedge\ works2.company\text{-}name\ =\ \text{“Small Bank Corporation”})}$
$\rho_{works2}(works)))$

h. Note: Small Bank Corporation will be included in each answer.

$\Pi_{company\text{-}name}\ (company\ \div$
$(\Pi_{city}\ (\sigma_{company\text{-}name\ =\ \text{“Small Bank Corporation”}}\ (company))))$

Relational Calculus

- A **relational calculus** expression creates a new relation, which is specified in terms of variables that range over rows of the stored database relations (in **tuple calculus**) or over columns of the stored relations (in **domain calculus**).

- In a calculus expression, there is *no order of operations* to specify how to retrieve the query result—a calculus expression specifies only what information the result should contain.
  - This is the main distinguishing feature between relational algebra and relational calculus.

Relational Calculus

- Relational calculus is considered to be a **nonprocedural** language.

- This differs from relational algebra, where we must write a *sequence of operations* to specify a retrieval request; hence relational algebra can be considered as a **procedural** way of stating a query.

- Types of Relational Calculus:
  - **Tuple Relational Calculus**
  - **Domain Relational Calculus**

Tuple Relational Calculus

- The tuple relational calculus is based on specifying a number of tuple variables.
- Each tuple variable usually ranges over a particular database relation, meaning that the variable may take as its value any individual tuple from that relation.
- A simple tuple relational calculus query is of the form

  {t | COND(t)}

  - where t is a tuple variable and COND (t) is a conditional expression involving t.
  - The result of such a query is the set of all tuples t that satisfy COND (t).

Tuple Relational Calculus (contd..)

Example:

- To find the first and last names of all employees whose salary is above $50,000, we can write the following tuple calculus expression:

{t.FNAME, t.LNAME | EMPLOYEE(t) AND t.SALARY>50000}

- The condition EMPLOYEE(t) specifies that the **range relation** of tuple variable t is EMPLOYEE.

- Explanation:

The first and last name (PROJECTION $\pi_{\text{FNAME, LNAME}}$) of each EMPLOYEE tuple t that satisfies the condition t.SALARY>50000 (SELECTION $\sigma_{\text{SALARY >50000}}$) will be retrieved.

The Existential and Universal Quantifiers

- Two special symbols called quantifiers can appear in formulas; these are the **universal quantifier (∀) and the existential quantifier (∃).**
- Informally, a tuple variable t is bound if it is quantified, meaning that it appears in an (∀ t) or (∃ t) clause; otherwise, it is free.
- If F is a formula, then so are (∃ t)(F) and (∀ t)(F), where t is a tuple variable.
  - The formula **(∃ t)(F) is true if the formula F evaluates to true for some (at least one) tuple** assigned to free occurrences of t in F; otherwise (∃ t)(F) is false.
  - The formula **(∀ t)(F) is true if the formula F evaluates to true for every tuple** (in the universe) assigned to free occurrences of t in F; otherwise (∀ t)(F) is false.

Example: Query Using Existential Quantifier

- Retrieve the name and address of all employees who work for the ‘Research’ department. The query can be expressed as :

{t.FNAME, t.LNAME, t.ADDRESS | EMPLOYEE(t) and (∃ d) (DEPARTMENT(d) and d.DNAME=‘Research’ and d.DNUMBER=t.DNO) }

- The only *free tuple variables* in a relational calculus expression should be those that appear to the left of the bar (|).
  - In above query, t is the only free variable; it is then *bound successively* to each tuple.
  - The conditions EMPLOYEE (t) and DEPARTMENT(d) specify the range relations for t and d.
  - The condition d.DNAME = ‘Research’ is a selection condition and corresponds to a SELECT operation in the relational algebra, whereas the condition d.DNUMBER = t.DNO is a JOIN condition.

Example Query Using Universal Quantifier

- Find the names of employees who work on *all* the projects controlled by department number 5. The query can be:

{e.LNAME, e.FNAME | EMPLOYEE(e) and ((∀ x)(not(PROJECT(x)) or not(x.DNUM=5)

- In query above, using the expression **not(PROJECT(x))** inside the universally quantified formula evaluates to true all tuples x that are not in the PROJECT relation.
  - Then we exclude the tuples we are not interested in from R itself. The expression not(x.DNUM=5) evaluates to true all tuples x that are in the project relation but are not controlled by department 5.

Languages Based on Tuple Relational Calculus

- The language **SQL** is based on tuple calculus. It uses the basic block structure to express the queries in tuple calculus:
  - SELECT <list of attributes>
  - FROM <list of relations>
  - WHERE <conditions>
- **SELECT** clause mentions the attributes being **projected**, the **FROM** clause mentions the relations needed in the query, and the **WHERE** clause mentions the selection as well as the **join** conditions.
  - SQL syntax is expanded further

The Domain Relational Calculus

- The language called QBE (Query-By-Example) that is related to domain calculus was developed almost concurrently to SQL at IBM Research, Yorktown Heights, New York.
  - Domain calculus was thought of as a way to explain what QBE does.
- Domain calculus differs from tuple calculus in the type of variables used in formulas:
  - Rather than having variables range over tuples, the variables range over single values from domains of attributes.
- To form a relation of degree n for a query result, we must have n of these domain variables— one for each attribute.

The Domain Relational Calculus

- An expression of the domain calculus is of the form

{ $x_1, x_2, . . ., x_n$ |

COND($x_1, x_2, . . ., x_n, x_{n+1}, x_{n+2}, . . ., x_{n+m}$)}

  - where $x_1, x_2, . . ., x_n, x_{n+1}, x_{n+2}, . . ., x_{n+m}$ are domain variables that range over domains (of attributes)
  - and COND is a condition or formula of the domain relational calculus.

Example Query Using Domain Calculus

- Retrieve the birthdate and address of the employee whose name is 'John B. Smith'.
- Query :

{uv | (∃ q) (∃ r) (∃ s) (∃ t) (∃ w) (∃ x) (∃ y) (∃ z) (EMPLOYEE(qrstuvwxyz) and q='John' and r='B' and s='Smith')}

- Ten variables for the employee relation are needed, one to range over the domain of each attribute in order.
  - Of the ten variables q, r, s, . . ., z, only u and v are free.
- Specify the *requested attributes*, BDATE and ADDRESS, by the free domain variables u for BDATE and v for ADDRESS.
- Specify the condition for selecting a tuple following the bar (|)—

3. Data Integrity

- Specify rules that implicitly/explicitly define consistent database or change of state
- Integrity constraints are necessary to avoid situations like:

1. Some data has been inserted in database but it cannot be identified
2. Same data is missing in different tables
3. During processing, keys are not compared properly

- Integrity constraint are defined as condition on database that restricts data that can be stored in database

Constraints Categories

1. **Implicit/ inherent Model based Constraint**
2. **Schema Based Constraints**

   Expressed in schemas of data model by specifying them in DDL

3. **Application Based or Semantic Constraint**

Domain Constraint

- Specify within each tuple, value of each attribute must be atomic value from dom(A)
- Datatypes associated with domains include numeric(int,floa,double), character, boolean, strings, date, time, timestamps etc

Entity Integrity Constraint

- **Relational Database Schema**: A set S of relation schemas that belong to the same database. S is the *name* of the **database**.

$$S = \{R_1, R_2, ..., R_n\}$$

- **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.

$$t[PK] \neq \text{null for any tuple } t \text{ in } r(R)$$

- <u>Note:</u> Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Key constraint

- **Superkey** of R: A set of attributes S_K of R such that no two tuples *in any valid relation instance 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].
- **Key** of R: A "minimal" superkey; and two distinct tuples cannot have identical values for all attributes in key
- **Example**: The CAR relation schema:

CAR(State, Reg#, SerialNo, Make, Model, Year)

has two keys Key1 = {State, Reg#}, Key2 = {SerialNo}, which are also superkeys. {SerialNo, Make} is a superkey but *not* a key.

- If a relation has *several* **candidate keys**, one is chosen arbitrarily to be the **primary key**. The primary key attributes are *underlined*.

Null constraint

- Null can be used when value is not known at present time
- Attributes may not be applicable
- Some information not known or never will be known
- However, having null values create issue from computing average for collection of values to comparing null with not null values

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**.
- Tuples in the *referencing relation* R_1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the *referenced relation* R_2. A tuple t_1 in R_1 is said to **reference** a tuple t_2 in R_2 if $t_1[FK] = t_2[PK]$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from R_1.FK to R_2.

Referential Integrity(contd..)

Statement of the constraint

The value in the foreign key column (or columns) FK of the the **referencing relation** R_1 can be either:

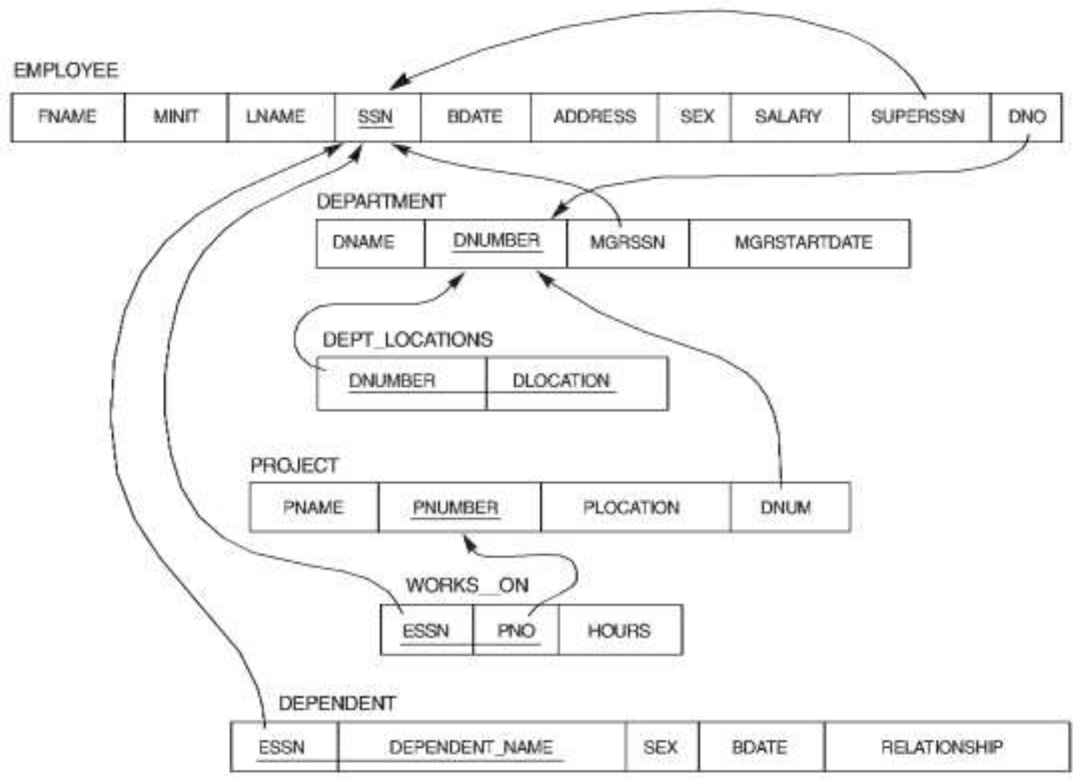
(1) a value of an existing primary key value of the corresponding primary key PK in the **referenced relation** $R_{2,}$, or..

(2) a null.

In case (2), the FK in R_1 should not be a part of its own primary key.

Referential Integrity(contd..)

Figure 7.7 Referential integrity constraints displayed on the COMPANY relational database schema diagram.



Other Types of Constraints

Semantic Integrity Constraints:

- based on application semantics and cannot be expressed by the model per se
- E.g., “the max. no. of hours per employee for all projects he or she works on is 56 hrs per week”
- A *constraint specification language* may have to be used to express these
- SQL-99 allows triggers and ASSERTIONS to allow for some of these

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.

- Integrity constraints should not be violated by the update operations.
- Updates may *propagate* to cause other updates automatically. This may be necessary to maintain integrity constraints.

Update Operations on Relations

- In case of integrity violation, several actions can be taken:
  - Cancel the operation that causes the violation (REJECT option)
  - Perform the operation but inform the user of the violation
  - Trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
  - Execute a user-specified error-correction routine

Advantages of Relational Model

1. **Data independence**: provide a sharp and clear boundary between logical and physical aspects of database management
2. **Simplicity**: simpler structure which is easy to communicate to users and programmers and wide number of users can interact with simple model
3. **Set-processing**: facilities for manipulating a set of records at a time so that programmers are not operating on database record by record
4. **Sound theoretical background**: provide theoretical background for database management field