Students Please print and study

Hardbook exercises (answers at the bottom)

3.17

{Address_phone({Phone(Area_code,Phone_number)},Address(Street_address (Number,Street,Apartment_number),City,State,Zip))}

Figure 3.5
A complex attribute: Address_phone.

Composite and multivalued attributes can be nested to any number of levels. Suppose we want to design an attribute for a STUDENT entity type to keep track of previous college education. Such an attribute will have one entry for each college previously attended, and each such entry will be composed of college name, start and end dates, degree entries (degrees awarded at that college, if any), and transcript entries (courses completed at that college, if any). Each degree entry contains the degree name and the month and year the degree was awarded, and each transcript entry contains a course name, semester, year, and grade. Design an attribute to hold this information. Use the conventions in Figure 3.5.

3.18

Show an alternative design for the attribute described in Exercise 3.17 that uses only entity types (including weak entity types, if needed) and relationship types.

3.22

A database is being constructed to keep track of the teams and games of a sports league. A team has a number of players, not all of whom participate in each game. It is desired to keep track of the players participating in each game for each team, the positions they played in that game, and the result of the game. Design an ER schema diagram for this application, stating any assumptions you make. Choose your favorite sport (e.g., soccer, baseball, football).

5.16

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:
STUDENT(Ssn, Name, Major, Bdate)
COURSE(Course#, Cname, Dept)
ENROLL(Ssn, Course#, Quarter, Grade)
BOOK_ADOPTION(Course#, Quarter, Book_isbn)
TEXT(Book_isbn, Book_title, Publisher, Author)
Specify the foreign keys for this schema, stating any assumptions you make.

5.19

Consider a STUDENT relation in a UNIVERSITY database with the following attribute111s (Name, Ssn, Local_phone, Address, Cell_phone, Age, Gpa). Note that

the cell phone may be from a different city and state (or province) from the local phone. A possible tuple of the relation is shown below:

Name	Ssn	Local_phone	Address	Cell_phone	Age	Gpa
George Shaw William Edwards	123-45-6789	555-1234	123 Main St., Anytown, CA 94539	555-4321	19	3.75

a. Identify the critical missing information from the Local_phone and Cell_phone attributes. (Hint: How do you call someone who lives in a different state or province?)
b. Would you store this additional information in the Local_phone and Cell_phone attributes or add new attributes to the schema for STUDENT?
c. Consider the Name attribute. What are the advantages and disadvantages of splitting this field from one attribute into three attributes (first name, middle name, and last name)?
d. What general guideline would you recommend for deciding when to store information in a single attribute and when to split the information?
e. Suppose the student can have between 0 and 5 phones. Suggest two different designs that allow this type of information.

6.8

Write appropriate SQL DDL statements for declaring the LIBRARY relational database schema of Figure 6.6. Specify the keys and referential triggered actions.

6.10

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date

DEPT_LOCATIONS

Dnumber	Dlocation

PROJECT

Pname	Pnumber	Plocation	Dnum

WORKS_ON

Essn	Pno	Hours

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship

Figure 5.5
Schema diagram for the COMPANY relational database schema.

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

Specify the following queries in SQL on the COMPANY relational database schema shown in Figure 5.5. Show the result of each query if it is applied to the COMPANY database in Figure 5.6.

a. Retrieve the names of all employees in department 5 who work more

than 10 hours per week on the ProductX project.
b. List the names of all employees who have a dependent with the same first name as themselves.
c. Find the names of all employees who are directly supervised by ‘Franklin Wong’.

6.13

STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Write SQL update statements to do the following on the database schema shown in Figure 1.2.

a. Insert a new student, <‘Johnson’, 25, 1, ‘Math’>, in the database.

b. Change the class of student ‘Smith’ to 2.

c. Insert a new course, <‘Knowledge Engineering’, ‘cs4390’, 3, ‘cs’>.

d. Delete the record for the student whose name is ‘Smith’ and whose student number is 17.

6.15

Consider that the EMPLOYEE table's constraint EMPSUPERFK as specified in Figure 6.2 is changed to read as follows:
CONSTRAINT EMPSUPERFK
FOREIGN KEY (Super_ssn) REFERENCES EMPLOYEE(Ssn)
ON DELETE CASCADE ON UPDATE CASCADE,
Answer the following questions:
a. What happens when the following command is run on the database state shown in Figure 5.6?
DELETE EMPLOYEE WHERE Lname = 'Borg'
b. Is it better to CASCADE or SET NULL in case of EMPSUPERFK constraint ON DELETE?

7.5
Specify the following queries on the database in Figure 5.5 in SQL. Show the query results if each query is applied to the database state in Figure 5.6.
a. For each department whose average employee salary is more than $30,000, retrieve the department name and the number of employees working for that department.
b. Suppose that we want the number of male employees in each department making more than $30,000, rather than all employees (as in Exercise 7.5a). Can we specify this query in SQL? Why or why not?

7.8
Specify the following views in SQL on the COMPANY database schema shown in Figure 5.5.
a. A view that has the department name, manager name, and manager salary for every department
b. A view that has the employee name, supervisor name, and employee salary for each employee who works in the 'Research' department
c. A view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project
d. A view that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project with more than one employee working on it

7.9
Consider the following view, DEPT_SUMMARY, defined on the COMPANY database in Figure 5.6:
CREATE VIEW DEPT_SUMMARY (D, C, Total_s, Average_s)
AS SELECT Dno, COUNT (*), SUM (Salary), AVG (Salary)
FROM EMPLOYEE
GROUP BY Dno;
State which of the following queries and updates would be allowed on the

view. If a query or update would be allowed, show what the corresponding query or update on the base relations would look like, and give its result when applied to the database in Figure 5.6.

a. SELECT *
FROM DEPT_SUMMARY;

b. SELECT D, C
FROM DEPT_SUMMARY
WHERE TOTAL_S > 100000;

c. SELECT D, AVERAGE_S
FROM DEPT_SUMMARY
WHERE C > (SELECT C FROM DEPT_SUMMARY WHERE D = 4);

d. UPDATE DEPT_SUMMARY
SET D = 3
WHERE D = 4;

e. DELETE FROM DEPT_SUMMARY
WHERE C > 4;

8.7

How are the OUTER JOIN operations different from the INNER JOIN operations? How is the OUTER UNION operation different from UNION?

8.10

Discuss the meanings of the existential quantifier ($\exists$) and the universal quantifier ($\forall$).

8.16

Specify the following queries on the COMPANY relational database schema shown in Figure 5.5 using the relational operators discussed in this chapter. Also show the result of each query as it would apply to the database state in Figure 5.6.

a. Retrieve the names of all employees in department 5 who work more than 10 hours per week on the ProductX project.

b. List the names of all employees who have a dependent with the same first name as themselves.

c. Find the names of all employees who are directly supervised by ‘Franklin Wong’.

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

e. Retrieve the names of all employees who work on every project.

f. Retrieve the names of all employees who do not work on any project.

g. For each department, retrieve the department name and the average salary of all employees working in that department.

h. Retrieve the average salary of all female employees.

i. Find the names and addresses of all employees who work on at least one

project located in Houston but whose department has no location in Houston.
j. List the last names of all department managers who have no dependents.

8.17
Consider the AIRLINE relational database schema shown in Figure 5.8, which was described in Exercise 5.12. Specify the following queries in relational algebra:
a. For each flight, list the flight number, the departure airport for the first leg of the flight, and the arrival airport for the last leg of the flight.
b. List the flight numbers and weekdays of all flights or flight legs that depart from Houston Intercontinental Airport (airport code 'iah') and arrive in Los Angeles International Airport (airport code 'lax').
c. List the flight number, departure airport code, scheduled departure time, arrival airport code, scheduled arrival time, and weekdays of all flights or flight legs that depart from some airport in the city of Houston and arrive at some airport in the city of Los Angeles.
d. List all fare information for flight number 'co197'.
e. Retrieve the number of available seats for flight number 'co197' on '2009-10-09'.

8.22
Consider the two tables T1 and T2 shown in Figure 8.15. Show the results of the following operations:
a. T1 T1.P = T2.A T2
b. T1 T1.Q = T2.B T2
c. T1 T1.P = T2.A T2
d. T1 T1.Q = T2.B T2
e. T1 ∪ T2
f. T1 (T1.P = T2.A AND T1.R = T2.C) T2

HANDOUT exercises (answers at the bottom)

Exercise 2.2
A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseid). Professors teach courses; each of the following situations concerns the Teaches relationship set. For each situation, draw an ER diagram that describes it (assuming no further constraints hold).
1. Professors can teach the same course in several semesters, and each offering must be recorded.
2. Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded. (Assume this condition applies in all subsequent questions.)
3. Every professor must teach some course.

4. Every professor teaches exactly one course (no more, no less).
5. Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.
6. Now suppose that certain courses can be taught by a team of professors jointly, but it is possible that no one professor in a team can teach the course. Model this situation, introducing additional entity sets and relationship sets if necessary.

Exercise 3.4

What is the difference between a candidate key and the primary key for a given relation? What is a superkey?

Answer: A Candidate Key can be any column or a combination of columns that can qualify as unique key in database. A Primary Key is a column or a combination of columns that uniquely identify a record. A superkey is a set of attributes within a table whose values can be used to uniquely identify a tuple

Exercise 3.8

Answer each of the following questions briefly. The questions are b&'>ed OIl the following relational schema:

Emp(eid: integer, ename: string, age: integer, sala1l1: real)
Works(eid: integer, did: integer, peLtime: integer)
Dept(did: integer, dname: string, budget: real, managerid: integer)

Exercise 4.2

Given two relations R1 and R2, where R1 contains N1 tuples, R2 contains N2 tuples, and N2 > N1 > 0, give the minimum and maximum possible sizes (in tuples) for the resulting relation produced by each of the following relational algebra expressions. In each case, state any assumptions about the schemas for R1 and R2 needed to make the expression meaningful:

(1) R1 U R2, (2) R1 n R2, (3) R1 ~ R2, (4) R1 x R2, (5) (Ta=5(R1), (6) 7Ta(R1), and (7) R1/R2

Exercise 4.4 Consider the Supplier-Parts-Catalog schema from the previous question. State what the following queries compute:

1. $\pi_{sname}(\pi_{sid}(\sigma_{color='red'} Parts)$!x1 $(O'cost<lOoCatalog)$!x1 $Suppliers)$
2. $\pi sname(\pi_{sid}((\sigma_{color='red'} Parts)$!x1 $(\sigma_{cost<looCatalog})$!x1 $Suppliers))$
3. $(\pi_{sname}((O'color='red' Parts) \bowtie (crcost<looCatalog)$!x1 $Suppl'iers))$ n

$$(\pi_{sname}((\sigma_{color='green'} Parts) \bowtie (\sigma_{cost<100} Catalog) \bowtie Suppliers))$$

4. $(1fsid((crcolor='red,Parts)$!x1 $(crcost<10oCatalog) \bowtie Suppliers))$ n

$$(\pi_{sid}((\sigma_{color='green'} Parts) \bowtie (crcost<lOoCatalog)$!x1 $Suppliers))$$

5. $\pi_{sname}((\pi_{sid,sname}((\sigma_{color='red'} Parts) \bowtie (\sigma_{cost<100} Catalog) \bowtie Suppliers))$ ∩

$$(\pi_{sid,sname}((\sigma CO!07'='green' Parts)$!x1 $(\sigma cost< lOoCatalog) \bowtie Suppliers)))$$

Note: Need to have Supplier-Parts-Catalog schema from previous question.

Exercise 5.1 Consider the following relations:
Student(snum: integer, sname: string, major: string, level: string, age: integer)
Class(name: string, meets_at: time, room: string, fid: integer)
Enrolled(snum: integer, cname: string)
Faculty (fid: integer, fnarne: string, deptid: integer)
The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.
Write the following queries in SQL. No duplicates should be printed in any of the ans\vers.

1. Find the names of all Juniors (level = JR) who are enrolled in a class taught by 1. Teach.
Answer: SELECT DISTINCT S.sname
FROM Enrolled E, Student S, Class C, Faculty F
WHERE E.snum = S.snum
AND E.cname = C.name
AND F.fid = C.fid
AND S.level = 'JR'
AND F.fid = 1

2. Find the age of the oldest student who is either a History major or enrolled in a course taught by I. Teach.
Answer: SELECT DISTINCT S.age

3. Find the names of all classes that either meet in room R128 or have five or more students enrolled.
4. Find the names of all students who are enrolled in two classes that meet at the same time.

5. Find the names of faculty members \vho teach in every room in which some class is

taught.
6. Find the names of faculty members for \vhorn the combined enrollment of the courses that they teach is less than five.
7. Print the level and the average age of students for that level, for each level.
8. Print the level and the average age of students for that level, for all levels except JR.
9. For each faculty member that has taught classes only in room R128, print the faculty member's name and the total number of classes she or he has taught.
10. Find the names of students enrolled in the maximum number of classes.
11. Find the names of students not enrolled in any class.
12. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

Exercise 5.3
The following relations keep track of airline flight information:

Flights(flno: integer, *from:* string, *to:* string, *di8tance:* integer, *departs:* time, *arrives:* time, *price:* integer)
Aircraft(*aid:* integer, *aname:* string, *cruisingrange:* integer)
Certified(*eid:* integer, *aid:* integer)
Employees(*eid:* integer, *ename:* string, *salary:* integer)

Note that the Employees relation describes pilots and other kinds of employees as well; every pilot is certified for some aircraft, and only pilots are certified to fly. Write each of the follO\ving queries in SQL. (Additional queries using the same schema are listed in the exercises for' Chapter 4·)
1. Find the names of aircraft such that all pilots certified to operate them earn more than $80,000.
2. For each pilot who is certified for more than three aircraft, find the eid and the maximum cruisingrange of the aircraft for which she or he is certified.
3. Find the names of pilots whose salary is less than the price of the cheapest route from Los Angeles to Honolulu.
4. For all aircraft with cruisingrange over 1000 miles, find the name of the aircraft and the average salary of all pilots certified for this aircraft.
5. Find the names of pilots certified for some Boeing aircraft.
6. Find the aids of all aircraft that can be used on routes from Los Angeles to Chicago.
7. Identify the routes that can be piloted by every pilot who makes more than $100,000.
8. Print the names of pilots who can operate planes with cruisingrange greater than 3000 miles but are not certified on any Boeing aircraft.
9. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.
10. Compute the difference between the average salary of a pilot and the average salary of

all employees (including pilots).
11. Print the name and salary of every non pilot whose salary is more than the average salary for pilots.
12. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles.
13. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles, but on at least two such aircrafts.
14. Print the names of employees who are certified only on aircrafts with cruising range longer than 1000 miles and who are certified on some Boeing aircraft.

Exercise 5.5
Consider the instance of the Sailors relation shown in Figure 5.22.

sid	*sname*	*rating*	*age*
18	jones	3	30.0
41	jonah	6	56.0
22	ahab	7	44.0
63	moby	*null*	15.0

Figure 5.22 An Instance of Sailors

1. Write SQL queries to compute the average rating, using AVGj the sum of the ratings, using SUM; and the number of ratings, using COUNT.
2. If you divide the sum just computed by the count, would the result be the same as the average? How would your answer change if these steps were carried out with respect to the age field instead of rating?
~3. Consider the following query: Find the names of sailors with a higher rating than all sailors with age < 21. The following two SQL queries attempt to obtain the answer to this question. Do they both compute the result? If not, explain why. Under what conditions would they compute the same result?

```
SELECT  S.sname
FROM    Sailors S
WHERE   NOT EXISTS ( SELECT *
                     FROM   Sailors S2
                     WHERE  S2.age < 21
                            AND S.rating <= S2.rating )
SELECT  *
FROM    Sailors S
WHERE   S.rating > ANY (SELECT S2.rating
                        FROM   Sailors S2
                        WHERE  S2.age < 21
```

Exercise 5.8

Consider the following relations:

Student(*snum:* integer, *sname:* string, *rnajor:* string, *level:* string, *age:* integer)
Class(narne: string, *meets_at:* time, *roorn:* string, *fid:* integer)
Enrolled(*snum:* integer, *cnarne:* string)
Faculty(*fid:* integer, *fnarne:* string, *deptid:* integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class.

1. Write the SQL statements required to create these relations, including appropriate versions of all primary and foreign key integrity constraints.
2. Express each of the following integrity constraints in SQL unless it is implied by the primary and foreign key constraint; if so, explain how it is implied. If the constraint cannot be expressed in SQL, say so. For each constraint, state what operations (inserts, deletes, and updates on specific relations) must be monitored to enforce the constraint.

(a) Every class has a minimum enrollment of 5 students and a maximum enrollment of 30 students.
(b) At least one dass meets in each room.
(c) Every faculty member must teach at least two courses.
(d) Only faculty in the department with deptid=33 teach more than three courses.
(e) Every student must be enrolled in the course called lVlathlOl.
(f) The room in which the earliest scheduled class (i.e., the class with the smallest nucets_at value) meets should not be the same as the room in which the latest scheduled class meets.
(g) Two classes cannot meet in the same room at the same time.

(h) The department with the most faculty members must have fewer than twice the number of faculty members in the department with the fewest faculty members.
(i) No department can have more than 10 faculty members.
(j) A student cannot add more than two courses at a time (i.e., in a single update).
(k) The number of CS majors must be more than the number of Math majors.
(I) The number of distinct courses in which CS majors are enrolled is greater than the number of distinct courses in which Math majors are enrolled.
(rn) The total enrollment in courses taught by faculty in the department with deptid=SS is greater than the number of ivlath majors.
(n) There must be at least one CS major if there are any students whatsoever.
(0) Faculty members from different departments cannot teach in the same room.

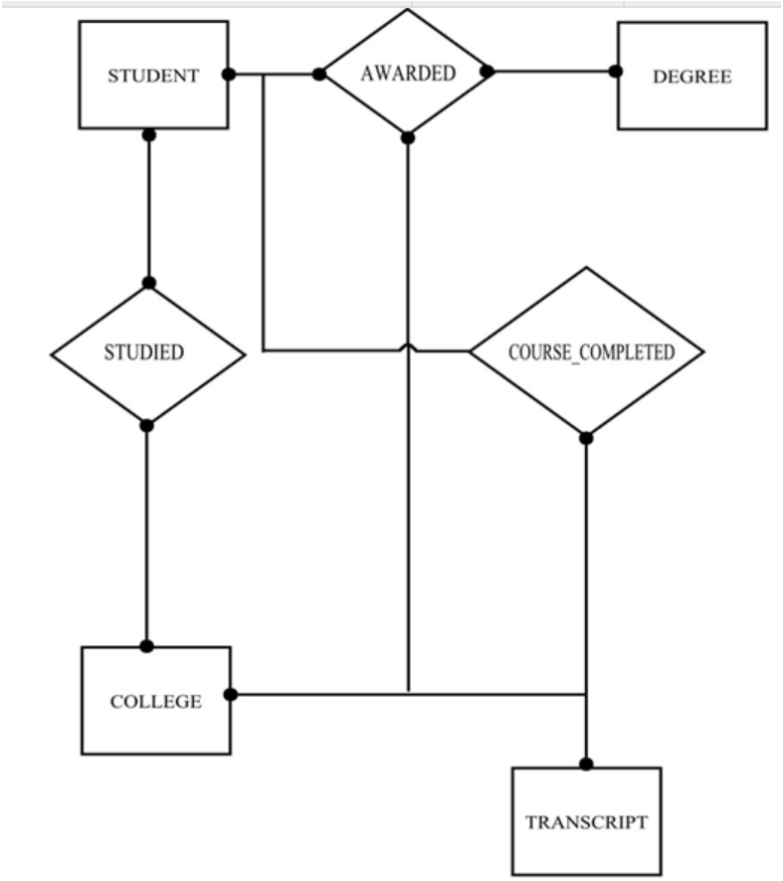
SOLUTIONS TO FUNDAMENTALS OF DATABASE SYSTEMS (7th Edition)

3.17

As multivalued attributes are represented by {} and composite attributes are represented by () attribute Prev_colloege can be written shown as:

Prev_college{College_name,Start_date,End_date, {Degrees(Degree_name,month,year)}, {Transcript (Transcript_name, Semester, Year, Grade)} }

3.18



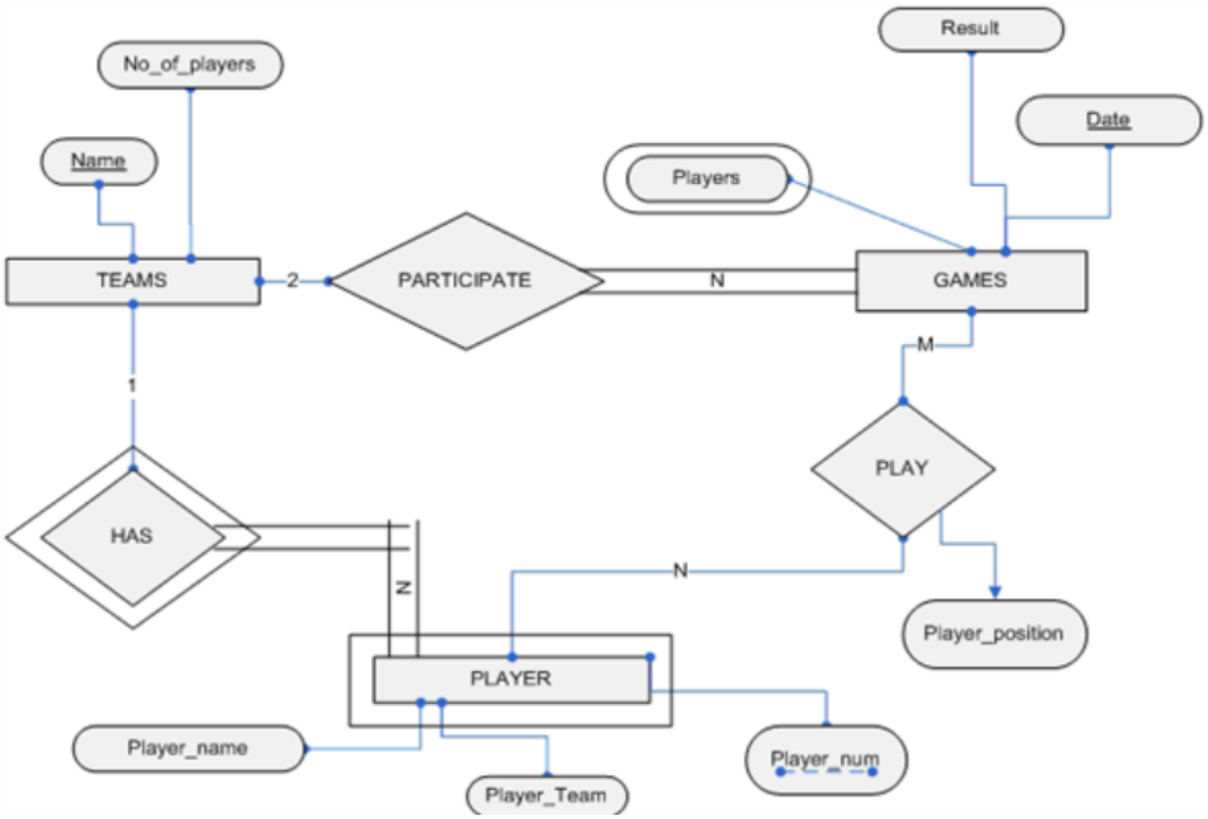
Alternative design has four entity types:

- STUDENT
- COLLEGE
- TRANSCRIPT
- DEGREE

And there are three relations:

- STUDIED: STUDENT studied in COLLEGE
- AWARDED: STUDENT is AWARDED DEGREE in COLLEGE.
- COURSE_COMPLETED: STUDENT completed a TRANSCRIPT in COLLEGE.

3.22



Assumptions:

1. All players have to be part of TEAMS.
2. Only two teams can participate in each game.
3. Each player in a team has unique number.
4. On a date only one game takes place
5. A player can play many games.

5.16

Foreign keys are:

a. Ssn from ENROLL is FK for STUDENT:Students with valid Ssn can Enroll.

b. Course# from ENROLL is FK for COURSE: Students can enroll for existing courses.

c. Course# and Quarter from ENROLL is FK for BOOK_ADOPTION.

d. Book_isbn from BOOK_ADOPTION is FK for TEXT.

5.19

a. State, province or city code is missing from phone number information.

b. Since cell phone and local phone can be of different city or state, additional information must be added in Local_phone and Cell_phone attributes.

c. If Name is Split in First_name, Middle_name and Last_name attributes there can be following advantages:

• Sorting can be done on basis of First Name or Last Name or Middle Name.

Disadvantages:

• By splitting single attribute into three attributes NULL values may increase in database. (If few students don't have a Middle Name.)

• Extra Memory will be consumed for storing NULL values of attributes that may not exist for a particular student. (Middle Name).

d. To decide when to store information in single attribute:

• When storing information in different attributes will create NULL values, single attribute must be preferred.

• When while using single attribute atomicity can not be maintained, we must use different attributes.

• When information needs to be sorted on the basis of some Sub-field of and attribute or when any sub-field is needed for decision making, we must split single attribute into many.

e.

First Design

• STUDENT(Name, Ssn, Phone_number_count, Address, Age, Gpa)

Phone (Ssn, Phone_number)

Second Design:

• STUDENT(Name, Ssn, Phone_number1, Phone_number2, Phone_number3, Phone_number4, Phone_number5, Address, Age, Gpa)

Although schema can be designed in either of the two ways but design first is better than second as it leaves lesser number of NULL values.

6.8

Set of statements for the LIBRARY relational schema from the figure 6.14 in the text book. The CREATE TABLE is like this:

CREATE TABLE BOOK (BookId CHAR(20) NOT NULL, Title VARCHAR(30) NOT NULL, PublisherName VARCHAR(20), PRIMARY KEY (BookId), FOREIGN KEY (PublisherName) REFERENCES PUBLISHER (Name) ON UPDATE CASCADE);

CREATE TABLE BOOK_AUTHORS (BookId CHAR(20) NOT NULL, AuthorName VARCHAR(30) NOT NULL, PRIMARY KEY (BookId, AuthorName), FOREIGN KEY (BookId) REFERENCES BOOK (BookId) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE PUBLISHER (Name VARCHAR(20) NOT NULL, Address VARCHAR(40) NOT NULL, Phone CHAR(12), PRIMARY KEY (Name));

CREATE TABLE BOOK_COPIES (BookId CHAR(20) NOT NULL, BranchId INTEGER NOT NULL, No_Of_Copies INTEGER NOT NULL, PRIMARY KEY (BookId, BranchId), FOREIGN KEY (BookId) REFERENCES BOOK (BookId)

ON DELETE CASCADE ON UPDATE CASCADE,FOREIGN KEY (BranchId) REFERENCES BRANCH (BranchId) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE BORROWER (CardNo INTEGER NOT NULL, Name VARCHAR(30) NOT NULL, Address VARCHAR(40) NOT NULL, Phone CHAR(12),

PRIMARY KEY (CardNo));

CREATE TABLE BOOK_LOANS (CardNo INTEGER NOT NULL, BookId CHAR(20) NOT NULL, BranchId INTEGER NOT NULL, DateOut DATE NOT NULL,

DueDate DATE NOT NULL, PRIMARY KEY (CardNo, BookId, BranchId),

FOREIGN KEY (CardNo) REFERENCES BORROWER (CardNo) ON DELETE CASCADE ON UPDATE CASCADE, FOREIGN KEY (BranchId) REFERENCES LIBRARY_BRANCH (BranchId) ON DELETE CASCADE ON UPDATE CASCADE,

FOREIGN KEY (BookId) REFERENCES BOOK (BookId) ON DELETE CASCADE ON UPDATE CASCADE);

CREATE TABLE LIBRARY_BRANCH (BranchId INTEGER NOT NULL, BranchName VARCHAR(20) NOT NULL, Address VARCHAR(40) NOT NULL,

PRIMARY KEY (BranchId));

6.10

a)

Query:

Select emp.Fname, emp.Lname

from employee emp, works_on w, project p

where emp.Dno = 5 and emp.ssn = w.Essn and w.Pno = p.pnumber and p.pname = 'ProductX' and w.hours > 10

Result:

Fname Lname

John Smith

Joyce English

Explanation:

The above query will display the names of all employees of department “5” and who works more than 10 hours per week on the project “Product X”.

b)

Query:

Select emp.Fname, emp.Lname

from employee emp, dependent d

where emp.ssn= d.essn and emp.Fname = d.Dependent_name

Result: (empty)

Fname Lname

Explanation:

The above query will display the names of the entire employee who have a dependent with the same first name as themselves.

• Here, the result is empty. Because, it does not have the same first name in dependent and employee table.

c)

Query:

```
Select emp.Fname, emp.Lname
from employee emp, employee emp1
where emp1.Fname= ‘Franklin’ and emp1.Lname = ‘Wong’ and emp.superssn = emp1.ssn
```

Fname Lname

John Smith

Ramesh Narayan

Joyce English

Explanation:

The above query uses self-join to display the names of all the employees who are under the supervision of Franklin Wong.

6.13

SQL queries for following data which is given in text book

(a)

```
INSERT INTO STUDENT VALUES ('Johnson', 25, 1, 'MATH')
```

(b)

```
UPDATE STUDENT SET CLASS = 2 WHERE Name='Smith'
```

(c)

```
INSERT INTO COURSE VALUES ('Knowledge Engineering','COSC4390', 3,'COSC')
```

(d)

```
DELETE FROM STUDENT WHERE Name='Smith' AND StudentNumber=17
```

6.15

a)

From the figure 8.2 in the text book, while EMP table constraint specified as

CONSTRAINT EMPSUPER FK FOREIGN KEY(supper_ssn) REFERNCES EMPLOYEE(Ssn) ON DELETET CASCADE ON UPDATE CASCADE,

From the figure 5.5 in the text book the result is like this.

The James E. Borg entry is deleted from the table, and each employee with him as a supervisor is also (and their supervisees, and so on). In total, 8 rows are deleted and the table is empty.

b)

Yes, It is better to SET NULL, since an employee is not fired (DELETED) when their supervisor is deleted. Instead, their SUPERSSN should be SET NULL so that they can later get a new supervisor.

7.5

a)

SELECT DNAME, COUNT (*) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO GROUP BY DNAME HAVING AVG (SALARY) > 30000

Result:

DNAME	DNUMBER	COUNT(*)
Research	5	4
Administration	4	3
Headquarters	1	1

(b)

This query follows the nested query and it is be specified in.

SELECT DNAME, COUNT (*) FROM DEPARTMENT, EMPLOYEE WHERE DNUMBER=DNO AND SEX='M' AND DNO IN (SELECT DNO FROM EMPLOYEE GROUP BY DNO HAVING AVG (SALARY) > 30000) GROUP BY DNAME

Result:

DNAME	DNUMBER	COUNT(*)
Research	5	3
Administration	4	1
Headquarters	1	1

7.8

a) CREATE VIEW MANAGER_INFOR

AS SELECT Dname, Fname AS Manager_First_name, Salary

FROM DEPARTMENT, EMPLOYEE

WHERE DEPARTMENT.Mgr_ssn = EMPLOYEE.Ssn

b) CREATE VIEW EMPLOYEE_INFO

AS SELECT e.Fname AS Employee_first_name, e.Minit AS Employee_middle_init, e.Lname AS Employee_last_name, s.Fname AS Manager_fname, s.Minit AS Manager_minit, s.Laname AS Manager_Lname, Salary

FROM EMPLOYEE e, EMPLOYEE s, DEPARTMENT d

WHERE e.Super_ssn = s.Ssn

AND e.Dno = d.Dnumber

Anonymous

it needs: AND d.Dname = 'Research';

c) CREATE VIEW PROJECT_INFO

AS SELECT Pname, Dname, COUNT(WO.Essn), SUM(WO.Hours)

FROM PROJECT P, WORKS_ON WO,DEPARTMENT D

WHERE P.Dnum = D.Dnumber

AND P.Pnumber = WO.Pno

GROUP_BY Pno

Anonymous

The GROUP BY Statement will not work. Pname and Dname must appear in the GROUP BY Clause. It should be: GROUP BY Pname, Dname;

d) CREATE VIEW PROJECT_INFO

AS SELECT Pname, Dname, COUNT(WO.Essn), SUM(WO.Hours)

FROM PROJECT P, WORKS_ON WO,DEPARTMENT D

WHERE P.Dnum = D.Dnumber

AND P.Pnumber = WO.Pno

GROUP_BY Pno

HAVING COUNT(WO.Essn) > 1

Comments (1)

Anonymous

"GROUP_BY" Statement is wrong should be GROUP BY instead.

7.9

a) Allowed

D	C	Total_s	Average_s
5	4	133000	33250
4	3	93000	31000
1	1	55000	55000

b) Allowed

D	C
5	4

c) Allowed

D	Average_s
5	33250

e) Not allowed because there can be multiple meaning of the query.

8.7

OUTER JOIN and INNER JOIN: Consider two relational databases R and S. When user wants to keep all the tuples in R, or all those in S, or all the tuples in R, or all those in S, or all those in both relations in the result of the JOIN regardless of weather or not they have matching tuples in other relation, set of operations called outer joins can do so. This satisfies the need of queries in which tuples from two tables are to be combined by matching corresponding rows, but without losing any tuples for lack of matching values.

When only matching tuples (based on condition) are contained in resultant relation and not all tuples then join is INNER JOIN (EQUIJOIN and NATURALJOIN).

In OUTER JOIN if matching values of other relation are not present fields are padded by NULL value.

OUTER UNION and UNION: For UNION operation databases have to be UNION compatible, i.e, they have same number of attributes and each corresponding pair of attributes have same domain.

OUTER UNION operation was developed to take the union of tuples from two relations if the relations are not union compatible. This operation will take UNION of tuples in two relations R(X, Y) and S(X,Z) that are **partial compatible,** meaning that only some attributes, say X, are union compatible. Resultant relation is of form RESULT(X, Y, Z).

Two tuples t1 in R and T2 in S are said to match if t1[X] =t2[X] and are considered to contain same entity instance. These are combined in single tuple.

For rest of tuples NULL values are padded.

8.10

Quantifier's are two types

(1) Existential quantifiers:-

(2) Universal quantifiers:-

(1) Existential quantifiers:-

Existential quantifier is a logical relation and symbolized as $\exists$ (" there exists").

Here

The statement is $(\exists t)(F)$

Based on the formula of existential quantifiers is if F is a formula, then so is $(\exists t)(F)$.

Where t is a tuple variable.

If the formula F evaluates to TRUE for some tuple assigned to free occurrences of t in F, then the formula $(\exists t)(F)$ is TRUE. Otherwise, it is FALSE.

(2) Universal Quantifiers:-

Universal quantifiers is a logical relation, it is symbolized as $\forall$.

The statement is $(\forall t)(F)$.

Based on the formula of universal quantifiers is

If F is a formula then statement is $(\forall t)(F)$.

Here t is the tuple variable and the formula F

Evaluates to true for every tuple assigned to free occurrences of t in F, then F is TRUE other wire it is FALSE.

8.16

The following symbols are used to write a relation algebra query:

Symbol	Operation
σ	SELECT
π	PROJECT
$\bowtie$	EQUI-JOIN
*	NATURAL JOIN
$\Im$	FUNCTION
$\div$	DIVISION
-	SET DIFFERENCE

a.

Following is the query to find the names of all employees in department 5 who work on the project `ProductX` for more than 10 hours per week:

$$R1 \leftarrow (\sigma_{PNAME = \text{'ProductX'}}(PROJECT))$$
$$R2 \leftarrow (R1) \bowtie_{PNUMBER = PNO} (WORKS\_ON)$$
$$R3 \leftarrow (EMPLOYEE) *_{SSN=ESSN} (\sigma_{HOURS>10} (R2))$$
$$Result \leftarrow \pi_{FNAME, LNAME} (\sigma_{DNO=5} (R3))$$

Result of query:

LNAME	FNAME
Smith	John
English	Joyce

Explanation:

- `R1` will give the details of the project whose `Pname` is '`ProductX`'. The details of `ProductX` will be the output.
- `R2` will check the table works_on for `Pno=1`. It will give `Essn`, `Pno` and `hours` of the project whose `Pnumber` is 1.
- `R3` will give the employee details from `EMPLOYEE` table whose `Ssn` number is same as the `Essn` number of result R2. The details of the employee with `Ssn` `123456789 and 453453453` will be displayed.
- `Result` will display only the `Fname` and `Lname` of the output obtained from R3.

b.

Following is the query that displays the names of the employee whose first name and dependent first name is same:

$$R1 \leftarrow (EMPLOYEE) \bowtie_{(SSN=ESSN)\ AND\ (FNAME=DEPENDENT\_NAME)} (DEPENDENT)$$
$$Result \leftarrow \pi_{FNAME, LNAME} (R1)$$

Result of query: Empty

FNAME	LNAME

Explanation:

- `R1` will retrieve the details of the employees whose `SSN` of `EMPLOYEE` table is same as the `ESSN` of `DEPENDENT` table and whose `FNAME` of `EMPLOYEE` table is same as the `DEPENDENT_NAME` of `DEPENDENT` table.
- `Result` will display only the `Fname` and `Lname` of the output obtained from R1.

d.

Following is the query that displays the project name and total hours per week spent one each project:

$$R1(PNO, TOT\_HRS) \leftarrow {}_{PNO}\Im_{SUM\ HOURS}(WORKS\_ON)$$
$$Result \leftarrow \pi_{PNAME,\ TOT\_HRS}((R1) \bowtie_{PNO=\ PNUMBER} (PROJECT))$$

Result of query:

PNAME	TOT_HRS
ProductX	52.5
ProductY	37.5
ProductZ	50.0
Computerization	55.0
Reorganization	25.0
NewBenefits	55.0

Explanation:

- `R1` will give the `PNO` and `sum of Hours` from `WORKS_ON` table.
- `Result` will give the `PNAME` and `total of Hours` by performing a equijoin between `WORKS_ON` table and `PROJECT table`. A equijoin is performed by matching the `PNO` of `WORKS_ON` table with `PNUMBER` of `PROJECT` table.

e.

Following is the query that displays the names of the employees who work on every project:

$R1(PNO, SSN) \leftarrow \pi_{PNO, ESSN}(WORKS\_ON)$

$R2(PNO) \leftarrow \pi_{PNUMBER}(PROJECT)$

$R3 \leftarrow \pi_{FNAME, LNAME}(R1 \div R2)$

$Result \leftarrow \pi_{FNAME, LNAME}(EMPLOYEE * R3)$

Result of query:

FNAME	LNAME

Explanation:

- R1 will output the PNO and SSN from WORKS_ON table.
- R2 will output the PNUMBER from PROJECT table.
- R3 will output the FNAME and LNAME after performing R1 ÷ R2.
- Result will output the FNAME and LNAME after performing natural join on EMPLOYEE and R3.

f.

Following is the query that displays the names of the employees who do not work on any project:

$R1 \leftarrow \pi_{SSN}(EMPLOYEE)$

$R2(SSN) \leftarrow \pi_{ESSN}(WORKS\_ON)$

$R3 \leftarrow R1 - R2$

$Result \leftarrow \pi_{FNAME, LNAME}(EMPLOYEE * R3)$

Result of query:

FNAME	LNAME

Explanation:

- R1 will output the SSN of all employees from EMPLOYEE table.
- R2 will output the ESSN from WORKS_ON table.
- R3 will perform the difference of result obtained from R1 and R2.
- Result will output the FNAME and LNAME after performing natural join on EMPLOYEE and R3.

g.

Following is the query that displays department name and department's average salary:

$$R1(DNUMBER, AVG\_SAL) \leftarrow {}_{DNO}\Im_{AVG\ SALARY}(EMPLOYEE)$$
$$Result \leftarrow \pi_{DNUMBER,\ AVG\_SAL}(R1\ *\ DEPARTMENT)$$

Result of query:

DNUMBER	AVG_SAL
Research	33250
Administration	31000
Headquarters	55000

Explanation:

- R1will output the average salary of each department from EMPLOYEE table.
- Result will output the average salary of each department after performing the natural join of R1 and DEPARTMENT.

h.

Following is the query that displays average salary of female employees:

$$Result(AVG\_F\_SAL) \leftarrow \Im_{AVG\ SALARY}(\sigma_{\ SEX\ =\ 'F'}(EMPLOYEE))$$

Result of query:

AVG_F_SAL
31000

Explanation:

- $(\sigma_{\ SEX\ =\ 'F'}(EMPLOYEE)$ will output the details of female employees.
- $\Im_{AVG\ SALARY}$ will perform the average of Salary of result obtained from $(\sigma_{\ SEX\ =\ 'F'}(EMPLOYEE)$.

i.

Following is the query that displays names and addresses of employees who work on a project at Houston and whose department has no location Houston:

$$R1(SSN) \leftarrow \pi_{ESSN}((WORKS\_ON) \bowtie_{PNO=PNUMBER}(\sigma_{PLOCATION='Houston'}(PROJECT))))$$
$$R2 \leftarrow \pi_{DNUMBER}(DEPARTMENT) - \pi_{DNUMBER}(\sigma_{PLOCATION='Houston'}(DEPARTMENT))$$
$$R3 \leftarrow \pi_{SSN}((EMPLOYEE) \bowtie_{DNO=DNUMBER}(R2))$$
$$R4 \leftarrow R1 - R3$$
$$Result \leftarrow \pi_{FNAME, LNAME, ADDRESS}(EMPLOYEE * R4)$$

Result of query:

FNAME	LNAME	ADDRESS
Jennifer	Wallace	291 Berry, Bellaire, TX

Explanation:

- R1 will give the ESSNs from WORKS_ON table whose Project location is in Houston. To know the Project location, PNUMBER of PROJECT table is equijoined with the PNO of WORKS_ON table.
- R2 will give the DNUMBER of departments that has no location in Houston.
- R3 will give the SSN from EMPLOYEE table whose DNO is equal to the DNUMBER obtained from R2.
- R4 will give the differences of R1 and R3.
- Result will output the FNAME, LNAME and ADDRESS after performing natural join on EMPLOYEE and R4.

j.

Following is the query that displays the last name of managers who have no dependents:

$$R1(SSN) \leftarrow \pi_{MGRSSN}\ (DEPARTMENT)$$
$$R2(SSN) \leftarrow \pi_{ESSN}\ (DEPENDANT)$$
$$R3 \leftarrow R1 - R2$$
$$Result \leftarrow \pi_{FNAME,\ LNAME}\ (EMPLOYEE * R3)$$

Result of query:

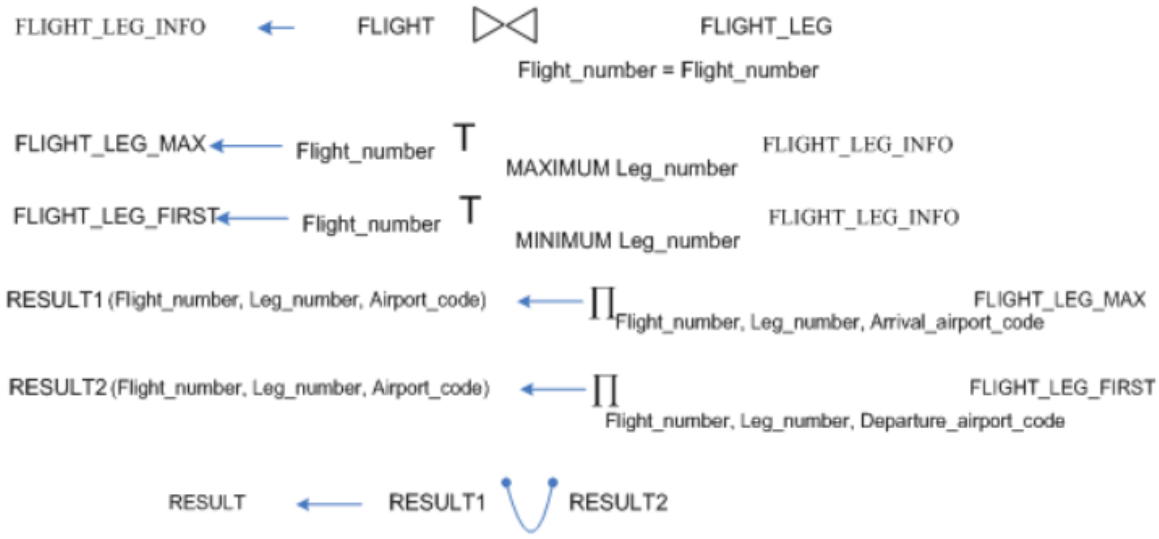
FNAME	LNAME
James	Borg

Explanation:

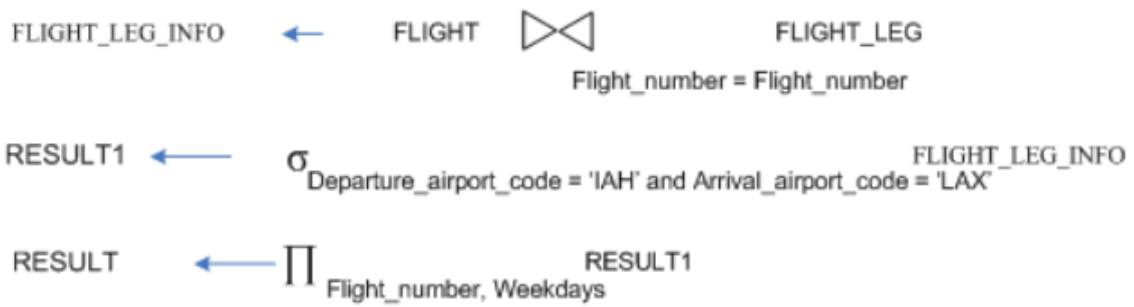
- `R1` will give the `MGRSSN` from `DEPARTMENT` table.
- `R2` will give the `ESSN` from `DEPENDANT` table.
- `R3` will give the difference of `R1-R2`.
- `Result` will output the `FNAME` and `LNAME` after performing natural join on `EMPLOYEE` and `R3`.

8.17

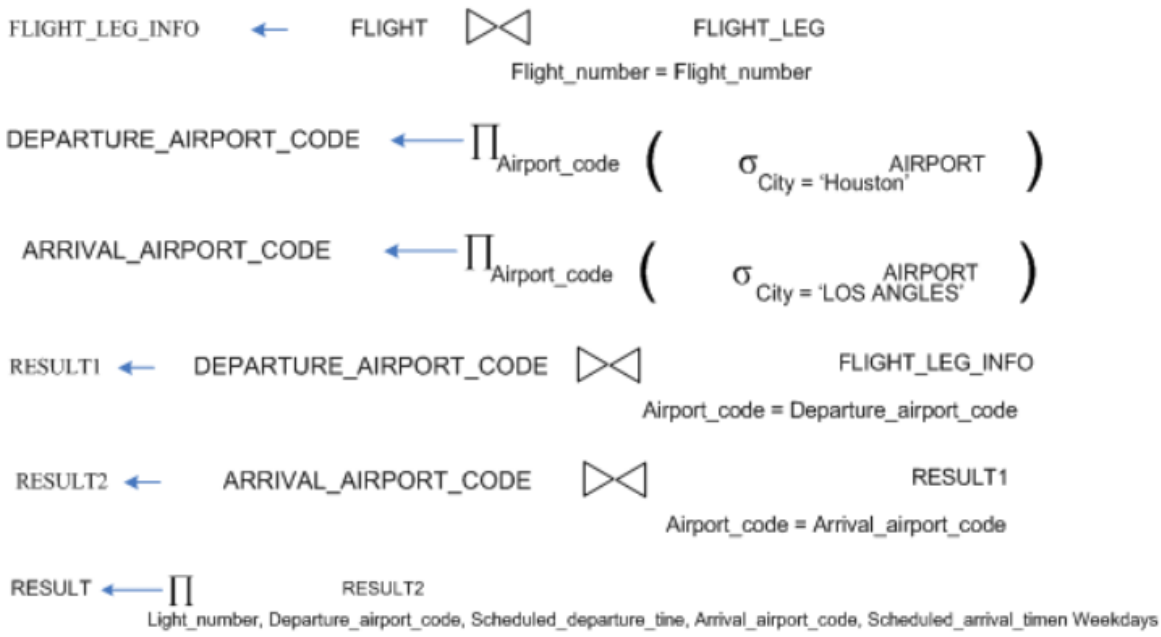
(a)



(b)


FLIGHT_LEG_INFO ← FLIGHT ⋈ FLIGHT_LEG
Flight_number = Flight_number
RESULT1 ← σ Departure_airport_code = 'IAH' and Arrival_airport_code = 'LAX' FLIGHT_LEG_INFO
RESULT ← ∏ Flight_number, Weekdays RESULT1

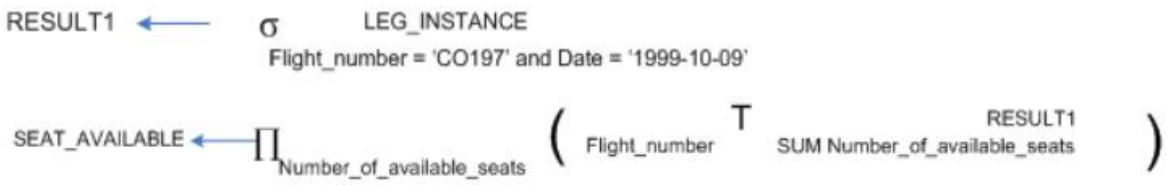
(c)


FLIGHT_LEG_INFO ← FLIGHT ⋈ FLIGHT_LEG
Flight_number = Flight_number
DEPARTURE_AIRPORT_CODE ← ∏ Airport_code (σ City = 'Houston' AIRPORT)
ARRIVAL_AIRPORT_CODE ← ∏ Airport_code (σ City = 'LOS ANGLES' AIRPORT)
RESULT1 ← DEPARTURE_AIRPORT_CODE ⋈ FLIGHT_LEG_INFO
Airport_code = Departure_airport_code
RESULT2 ← ARRIVAL_AIRPORT_CODE ⋈ RESULT1
Airport_code = Arrival_airport_code
RESULT ← ∏ RESULT2
Light_number, Departure_airport_code, Scheduled_departure_tine, Arrival_airport_code, Scheduled_arrival_timen Weekdays

(d)


RESULT ← σ Flight_number = 'CO197' FARE

(e)



8.22

Operations of relational algebra

The two tables *T1* and *T2* represent database states.

TABLE *T1*		TABLE *T2*			
P	**Q**	**R**	**A**	**B**	**C**
10	a	5	10	b	6
15	b	8	25	c	3
25	a	6	10	b	5

a) The operation $T1_{\triangleright\triangleleft T1.P=T2.A}T2$ is "THETA JOIN". It produces all the combinations of tuples that satisfy the join condition $T1.P = T2.A$. Following table is the result of the "THETA JOIN" operation.

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
25	a	6	25	c	3

b) The operation $T1 \triangleright\!\triangleleft_{T1.Q=T2.B} T2$ is "THETA JOIN". It produces all the combinations of tuples that satisfy the join condition $T1.Q = T2.B$. Following table is the result of the "THETA JOIN" operation.

P	Q	R	A	B	C
15	b	8	10	b	6
15	b	8	10	b	5

c) The operation $T1 \triangleright\!\triangleleft_{T1.P=T2.A} T2$ is "LEFT OUTER JOIN". It produces the tuples that are in the first or left relation *T1* with the join condition $T1.P = T2.A$. If no matching tuple is found in *T2*, then the attributes are filled with a NULL values. Following table is the result of the "LEFT OUTER JOIN" operation.

P	Q	R	A	B	C
10	a	5	10	b	6
10	a	5	10	b	5
15	a	8	NULL	NULL	NULL
25	a	8	25	c	3

d) The operation $T1 ⟖_{T1.Q=T2.B} T2$ is "RIGHT OUTER JOIN". It produces the tuples that are in the second or right relation *T2* with the join condition $T1.Q = T2.B$. If no matching tuple is found in *T1*, then the attributes are filled with a NULL values. Following table is the result of the "RIGHT OUTER JOIN" operation.

P	Q	R	A	B	C
15	b	8	10	b	6
NULL	NULL	NULL	25	c	3
15	b	8	10	b	5

e) The operation $T1 \cup T2$ is "UNION". It produces a relation that includes all the tuples that are in *T1* or *T2* or both *T1* and *T2*. The operation is possible since *T1* and *T2* are union compatible. Following table is the result of the "UNION" operation.

P	Q	R
10	a	5
15	b	8
25	a	6
10	b	6
25	c	3
10	b	5

f) The operation $T1 \triangleright\!\triangleleft_{(T1.P=T2.A \text{ AND } T1.R=T2.C)} T2$ is "THETA JOIN". It produces all the combinations of tuples that satisfy the join condition $T1.P = T.A$ AND $T1.R = T2.C$. Following table is the result of the "THETA JOIN" operation.

P	Q	R	A	B	C
10	a	5	10	b	5

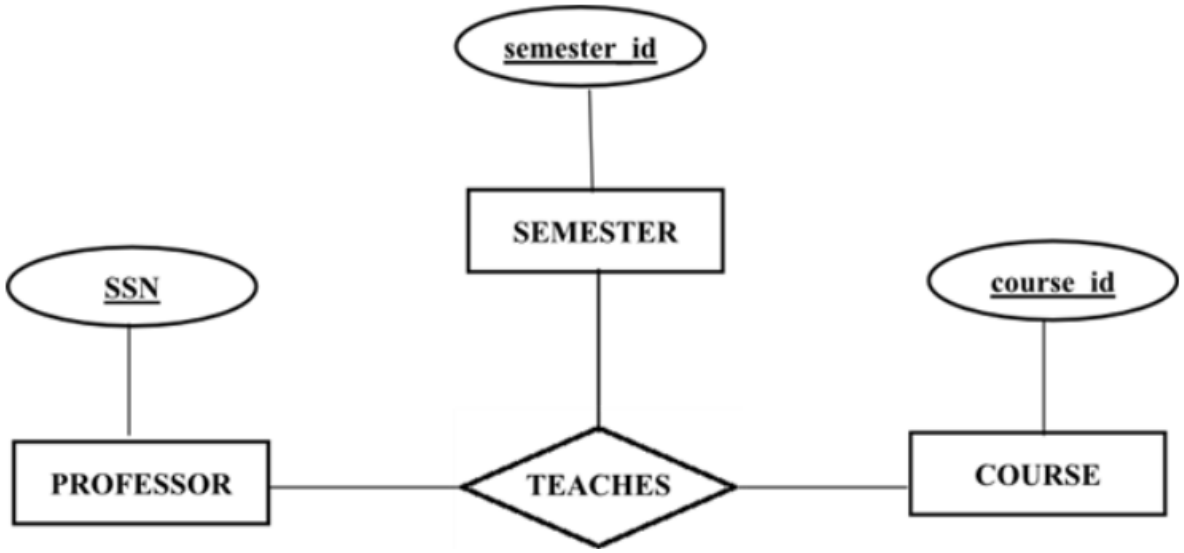
SOLUTIONS TO DATABASE MANAGEMENT SYSTEMS (3rd Edition)

ANSWER 2.2:

Step 1 of 6

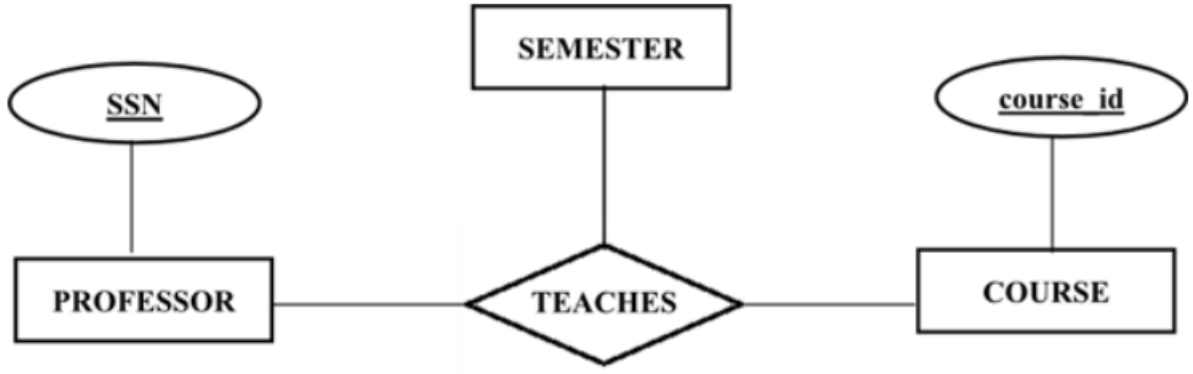
1.

Following is the ER diagram that describes the situation where professors can teach the same course in several semesters and every course taught by him must be recorded:



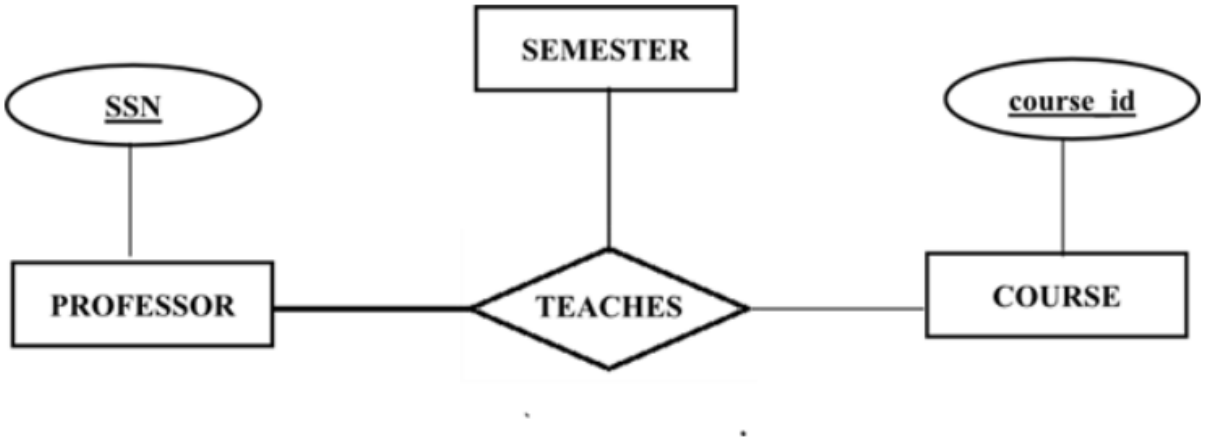
2.

Following is the ER diagram that describes the situation where professors can teach the same course in several semesters and only the most recent course that is taught is to be recorded:



3.

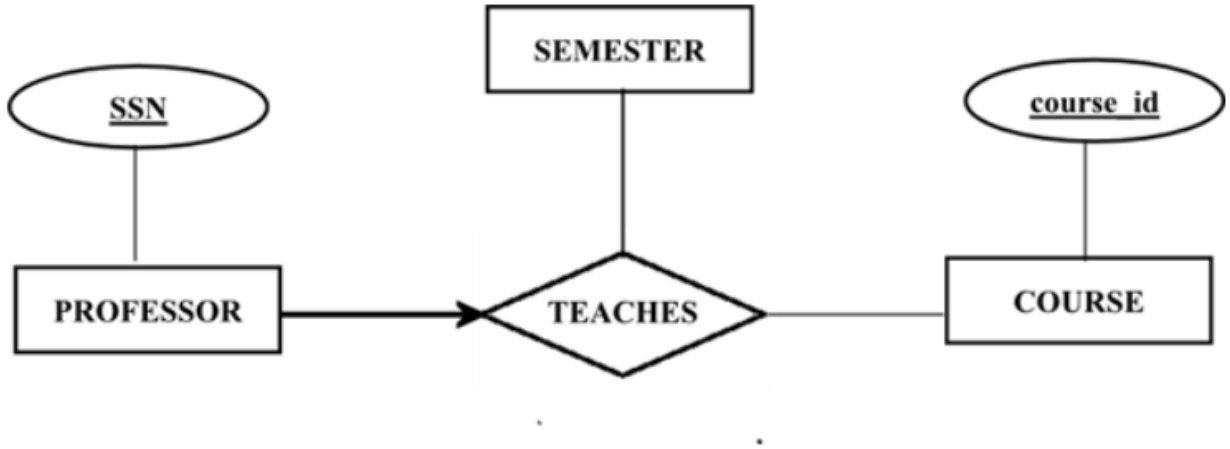
Following is the ER diagram that describes the situation where every professor is to teach some course:



As every professor must teach some course, the link between entity PROFESSOR and the relation TEACHES is indicated by a dark line.

4.

Following is the ER diagram that describes the situation where every professor teaches exactly one course:

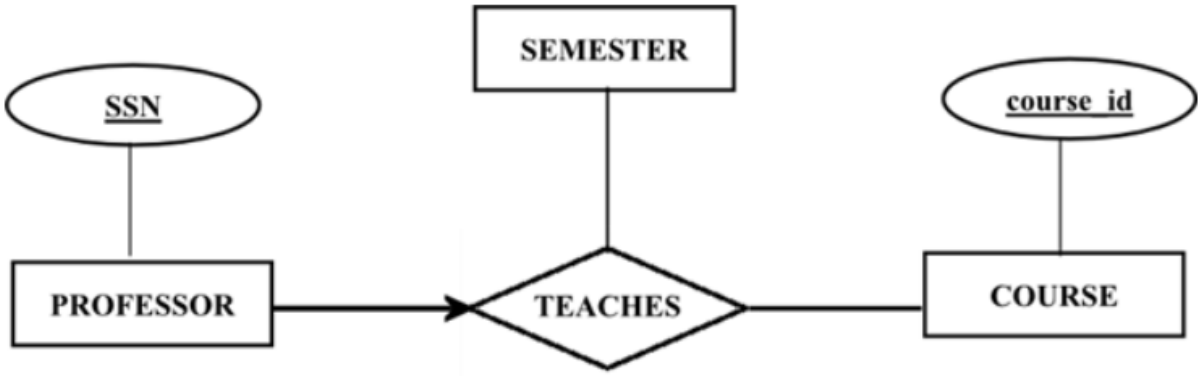


As every professor teaches exactly one course, the link between entity PROFESSOR and the relation TEACHES is indicated by a dark line with an arrow.

The arrow from entity PROFESSOR and the relation TEACHES indicates that each professor entity appears in at most one teach relationship.

5.

Following is the ER diagram that describes the situation where every professor teaches exactly one course and every course must be taught by some professor:



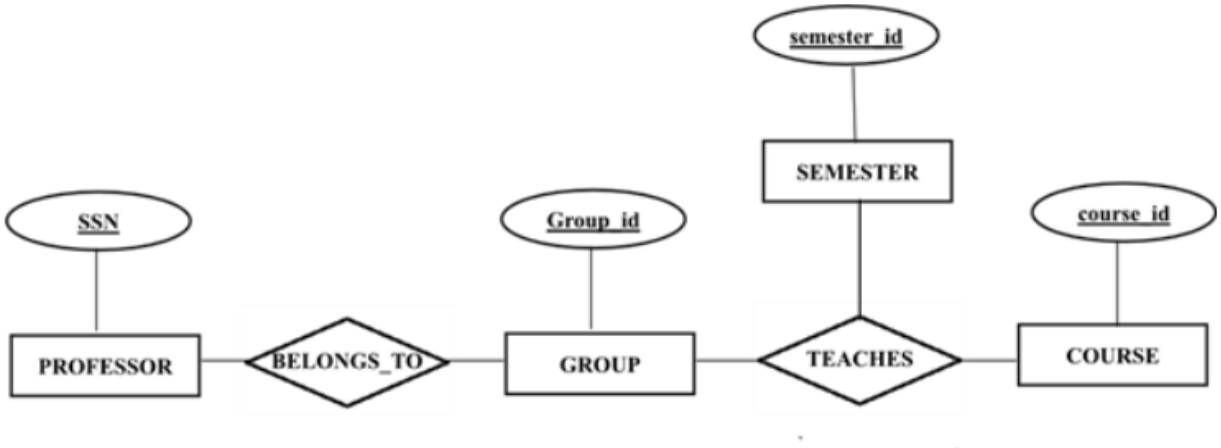
As every professor teaches exactly one course, the link between entity PROFESSOR and the relation TEACHES is indicated by a dark line with an arrow.

The arrow from entity PROFESSOR and the relation TEACHES indicates that each professor entity appears in at most one teach relationship.

As every course must be taught by some professor, the link between relation TEACHES and the entity COURSE is indicated by a dark line.

6.

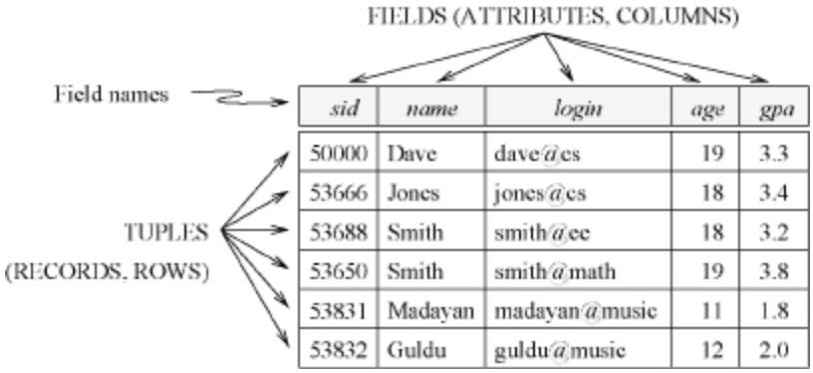
Following is the ER diagram that describes the situation where a team of professors can teach certain courses jointly, but it is possible that no one professor in a team can teach the course alone:



An entity GROUP identified by Group_id is introduced as entity to hold the groups.

ANSWER 3.4

Answer 3.4 The primary key is the key selected by the DBA from among the group of candidate keys, all of which uniquely identify a tuple. A superkey is a set of attributes that contains a key.



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.1 An Instance 51 of the Students Relation

ANSWER 3.8

1. Consider the following example. It is natural to require that the did field of Works should be a foreign key and refer to Dept

CREATE TABLE Works (eid INTEGER NOT NULL, did INTERGER NOT NULL, pcttime INTERGER, PRIMARY KEY (eid, did) UNIQUE (eid)

When a user attempts to delete a Dept tuple, There are four options:

- Also delete all Works tuples that refer to it
- Disallow the deletion of the Dept tuple if some Works tiple refers to it
- For every Works tuple that refers to it, set the did field to the did of something (exisiting) 'default' department
- For every Works tuple that refers to it, set the did field to null

2.

CREATE TABLE Emp (eid INTERGER, ename CHAR (10) age INTERGER salary REAL PRIMARY KEY (eid))

CREATE TABLE Works (eid INTERGER did INTERGER pcttime INTERGER PRIMARY KEY (eid, did) FOREIGN KEY (did) REFERENCES Dept FOREIGN KEY (eid) REFERENCES Emp ON DELETE CASCADE)

CREATE TABLE Dept (did INTERGER budget REAL managerid INTERGER PRIMARY KEY (did) FOREIGN KEY (managerid) REFERENCES Emp ON DELETE SET NULL)

3.

CREATE TABLE Dept (did INTERGER budget REAL managerid INTERGER NOT NULL PRIMARY KEY (did) FOREIGN KEY (mangerid) REFERENCES Emp)

4.

INSERT INTO Emp (eid, ename, age, salary) VALUES (101, 'John Doe', 32, 15000)

5.

UPDATE Emp ESET E.salary = E.salary * 1.10

6.

DELETE FROM Dept D WHERE D.dname = 'Toy'

The did field in the Works relation is a foreign key and references the Dept relation. This is the referential integrity constraint chosen. By adding the action ON DELETE CASCADE to this, when a department record is deleted, the Works record associated with that Dept is also deleted.

The query works as follows: The Dept relation is searched for a record with name = 'Toy' and that record is deleted. The did field of that record is then used to look in the Works relation for records with a matching did value. All such records are then deleted from the Works relation.

sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.2 Students with age < 18 on Instance S

ANSWER 4.2

Expression	Assumption	Min	Max
$R1 \cup R2$	$R1$ and $R2$ are union-compatible	N2	N1 + N2
$R1 \cap R2$	$R1$ and $R2$ are union-compatible	0	N1
$R1 - R2$	$R1$ and $R2$ are union-compatible	0	N1
$R1 \times R2$		N1 * N2	N1 * N2
$\sigma_{a=b}(R1)$	$R1$ has an attribute named a	0	N1
$\pi_a(R1)$	$R1$ has attribute a, N1>0	1	N1
$R1/R2$	The set of attributes of $R2$ is a subset of the set of attributes of $R1$	0	N1
$R2/R1$	The set of attributes of $R1$ is a subset of the set of attributes of $R2$	0	$\lfloor N2 / N1 \rfloor$

Figure 4.1 Answer to Exercise 4.2.

ANSWER 4.4

Answer 4.4 The statements can be interpreted as:

1. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars.
2. This Relational Algebra statement does not return anything because of the sequence of projection operators. Once the sid is projected, it is the only field in the set. Therefore, projecting on sname will not return anything.
3. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.
4. Find the Supplier ids of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.
5. Find the Supplier names of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.

ANSWER 5.1

1. SELECT DISTINCT S.Sname FROM Student S, Class C, Enrolled E, Faculty F WHERE S.snum = E.snum AND E.cname = C.name AND C.fid = F.fid AND F.fhame = 'I.Teach' AND S.level = 'JR'
2. SELECT MAX (S.age) FROM Student S WHERE (S.major = 'History') OR S.num IN (SELECT E.snum FROM Class C, Enrolled E, Faculty F WHERE E.cname - C.name AND C.fid = F.fid AND F.name = 'I.Teach')

3. SELECT C.name FROM Class C WHERE C.room = ‘R.128’ OR C.name IN (SELECT E.cname FROM Enrolled E GROUP BY E.cname HAVING COUNT (*) >=5)
4. SELECT DISTINCT S.sname FROM Student S WHERE S.snum IN (SELECT E1.snum FROM Enrolled E1, Enrolled E2, Class C1, Class C2 WHERE E1.snum = E2.snum AND E1. cname <> E2.cname AND E1.cname = C1.name AND E2.cname = C2.name AND C1.meets_at =C2.meets_at)
5. SELECT DISTINCT F.fname FROM Faculty F WHERE NOT EXISTS ((SELECT * FROM Class C) EXCEPT (SELECT C1.room FROM Class C1 WHERE C1.fid = F.fid))
6. SELECT DISTINCT F.fname FROM Faculty F WHERE 5 > (SELECT COUNT (E.snum) FROM Class C, Enrolled E WHERE C.name = E.cname AND C.fid = F.fid)
7. SELECT S.level, AVG (S.age) FROM Student S GROUP BY S.level
8. SELECT S.level, AVG (S.age) FROM Student S WHERE S.level <> ‘JR’ GROUP BY S.level
9. SELECT F.fname, COUNT (*) AS CourseCount FROM Faculty F, Class C WHERE F.fid = C.fid GROUP BY F.fid, Fname HAVING EVERY (C.room = ‘ R128’)
10. SELECT DISTINCT S.sname FROM Student S WHERE S.snum IN (SELECT E.snum FROM Enrolled E GROUP BY E.snum HAVING COUNT (*) >= ALL (SELECT COUNT (*) FROM Enrolled E2 GROUP BY E2.snum))
11. SELECT DISTINCT S.sname FROM Student S WHERE S.snum NOT IN (SELECT E.snum FROM Enrolled E)
12. SELECT S.age, S.level FROM Student S GROUP BY S.age, S.level, HAVING S.level IN (SELECT S1.level FROM Student S1 WHERE S1.age = S.age GROUP BY S1.level, S1age HAVING COUNT (*) >= ALL (SELECT COUNT (*) FROM Student S2 WHERE s1.age = S2.age GROUP BY S2.1evel, S2.age))

ANSWER 5.3

1. SELECT DISTINCT A.aname FROM Aircraft A WHERE A.Aid IN (SELECT C.aid FROM Certified C, Employees E WHERE C.eid = E.eid AND NOT EXISTS (SELECT * FROM Employees E1 WHERE E1.eid = E.eid AND E1.salary <80000))

2. SELECT C.eid, MAX (A.crusingrange) FROM Certified C, Aircraft A WHERE C.aid = A.aid GROUP BY C.eid HAVING COUNT (*) >3

3. SELECT DISTINCT E.anmae FROM Employee E WHERE E.salary < (SELECT MIN (F.price) FROM Flights F WHERE F.from = ‘LA’ AND F.to = ‘Honolulu’)

4. Observe that aid is the key for Aircraft, but the question asks for aircraft names; we deal with this complication by using an intermediate relation Temp:

SELECT Temp.name, Temp.AvgSalary FROM (SELECT A.aid, A.aname AS name, AVG (E.salary) AS AvgSalary FROM Aircraft A, Cerfied C, Employes E WHERE A.aid = C.aid AND C.eid = E.eid AND A.cruisingrange >1000 GROUP BY A.aid, A.aname) AS Temp

5.SELECT DISTINCT E.ename FROM Employees E, Certified C, Aircraft A WHERE E.eid = C.eid AND C.aid = A.aid AND A.anmae = ‘Boeing’

6.SELECT A.aid FROM Aircraft A WHERE A.crusingrange > (SELECT MIN (F.distance) FROM Flights F WHERE F.from = ‘L.A.’ AND F.to = ‘Chicago’)

7.SELECT DISTINCT F.from, F.to FROM Flights F WHERE NOT EXISTS (SELECT * FROM Employees E WHERE E.salary > 100000 AND NOT EXISTS (SELECT * FROM Aircraft A, Certified C WHERE A.crusingrange > F.distance AND E.eid = C.eid AND A.eid = C.aid))

8.SELECT DISTINCT E.ename FROM Employees E, Cerfied C, Aircraft A WHERE C.eid = E.eid AND C.aid = A.aid AND A.crusingrange > 30000 AND E.eid NOT IN (SELECT C1.eid FROM Certified C1, Aircraft A1 WHERE C1.aid = A1.aid AND A1.aname = ‘Boeing’)

9.SELECT F.departs FROM Flights F WHERE F.flno IN ((SELECT F0.flno FROM Flights F0 WHERE F0.from = ‘Madison’ AND F0.to = ‘NY’ AND AND F0.arrives <1800) UNION (SELECT F0.flno FROM Flights F0, Flights F1 WHERE F0.from = ‘Madison’ AND F0.to <> ‘NY’ AND AND F0.to = F1.from AND F1.to = ‘NY’ F1.departs > F0.arrives AND F1.arrives <1800) UNION (SELECT F0.flno FROM Flights F0, Flights F1, Flights F2 WHERE F0.from =’Madison’ WHERE F0.to = F1.from AND F1.to = F2.from AND F2.to =’NY’ AND F0.to <> ‘NY’ AND F1.to <> ‘NY’ AND F1.departs > F0.arrives AND F2.departs >F1.arrives AND F2.arrives <1800))

10.SELECT Temp1.avg - Temp2.avg FROM (SELECT AVG (E.salary) AS avg FROM Employees WHERE E.eid IN (SELECT DISTINCT C.eid FROM Certified C)) AS Tempi, (SELECT AVG (E1.salary) AS avg FROM Employees E1) AS Temp2

11.SELECT E.ename, E.salary FROM Employees E WHERE E.eid NOT IN (SELECT DISTINCT C.eid FROM Certified C) AND E.salary > (SELECT AVG (E1.salary) FROM Employees E1 WHERE E1.eid IN (SELECT DISTINCT C1.eid FROM Certified C1))

12.SELECT E.ename FROM Employees E, Certified C, Aircraft A WHERE C.aid = A.aid AND E.eid = C.eid GROUP BY E.eid, E.ename HAVING EVERY (A.crusingrange ¿ 1000)

13.SELECT E.ename FROM Employees E, Cerfiied C, Aircraft A WHERE C.aid = A.aid AND E.eid = C.eid GROUP BY E.eid, E.ename HAVING EVERY (A.cruisingrange ¿ 1000) AND COUNT (*) ¿ 1

14.SELECT E.ename FROM Employees E, Certified C, Aircraft A WHERE C.aid = A.aid AND E.eid = C.eid GROUP BY E.eid, E.enmae HAVING EVERY (A.cruisngrange ¿ 1000) AND ANY (A.anarne = ‘Boeing’)

ANSWER 5.5

Answer 5.5 The answers are shown below:

1.

SELECT AVG (S.rating) AS AVERAGEFROM Sailors S
SELECT SUM (S.rating)FROM Sailors S
SELECT COUNT (S.rating)FROM Sailors S

2. The result using SUM and COUNT would be smaller than the result using AVERAGE if there are tuples with rating = NULL. This is because all the aggregate operators, except for COUNT, ignore NULL values. So the first approach would compute the average over all tuples while the second approach would compute the average over all tuples with non-NULL rating values. However, if the aggregation is done on the age field, the answers using both approaches would be the same since the age field does not take NULL values.

3. Only the first query is correct. The second query returns the names of sailors with a higher rating than at least one sailor with age < 21. Note that the answer to the second query does not necessarily contain the answer to the first query. In particular, if all the sailors are at least 21 years old, the second query will return an empty set while the first query will return all the sailors. This is because the NOT EXISTS predicate in the first query will evaluate to true if its subquery evaluates to an empty set, while the ANY predicate in the second query will evaluate to false if its subquery evaluates to an empty set. The two queries give the same results if and only if one of the following two conditions hold:

- The Sailors relation is empty, or

4. (a)

sid	sname	rating	age	sid	sname	rating	age
18	jones	3	30.0	18	jones	3	30.0
41	jonah	6	56.0	41	jonah	6	56.0
22	ahab	7	44.0	22	ahab	7	44.0
63	moby	null	15.0	63	moby	null	15.0

(b)

sid	sname	rating	age	sid	sname	rating	age
18	jones	3	30.0	18	jones	3	30.0
41	jonah	6	56.0	41	jonah	6	56.0
22	ahab	7	44.0	22	ahab	7	44.0
63	moby	null	15.0	63	moby	null	15.0

(c)

sid	sname	rating	age	sid	sname	rating	age
18	jones	3	30.0	18	jones	3	30.0
41	jonah	6	56.0	41	jonah	6	56.0
22	ahab	7	44.0	22	ahab	7	44.0
63	moby	null	15.0	63	moby	null	15.0

(d)

sid	sname	rating	age	sid	sname	rating	age
18	jones	3	30.0	null	null	null	null
41	jonah	6	56.0	null	null	null	null

(e)

sid	sname	rating	age	sid	sname	rating	age
null	null	null	null	22	ahab	7	44.0
null	null	null	null	63	moby	null	15.0

(f)

sid	sname	rating	age	sid	sname	rating	age
18	jones	3	30.0	null	null	null	null
41	jonah	6	56.0	null	null	null	null
null	null	null	null	22	ahab	7	44.0
null	null	null	null	63	moby	null	15.0

- There is at least one sailor with age > 21 in the Sailors relation, and for every sailor s, either s has a higher rating than all sailors under 21 or s has a rating no higher than all sailors under 21.

ANSWER 5.8

1. The SQL statements needed to create the tables are given below:

CREATE TABLE Student (snum INTERGER, sname CHAR(20), major CHAR(20), level CHAR(20), age INTERGER, PRIMARY KEY (snum))

CREATE TABLE Faculty (fid INTERGER, fname CHAR (20), deptid INTERGER, PRIMARY KEY (fhum))

CREATE TABLE Class(name CHAR(20), meets_at TIME, room (CHAR(10), fid INTERGER, PRIMARY KEY (name), FOREIGN KEY (fid) REFERENCES Faculty)

CREATE TABLE Enrolled (snum INTERGER, cname CHAR(20), PRIMARY KEY (snum,cname), FOREIGN KEY (snum) REFERENCES Student), FOREIGN KEY (cname) REFERENCES Class)

2. The answer to each question is given below

(a) The Enrolled table should be modified as follows:

CREATE TABLE Enrolled (snum INTERGER, cname CHAR (20), PRIMARY KEY(snum,cname), FOREIGN KEY (snum) REFERENCES Student, FOREIGN KEY (cname) REFERENCES Class,), CHECK ((SELECT COUNT (E.snum) FROM Enrolled E GROUP BY E.cname) >= 5), CHECK ((SELECT COUNT (E.snum) FROM Enrolled E GROUP BY E.cname) <= 30))

(b) This constraint is already guaranteed because rooms are associated with classes, and thus a new room cannot be declared without an associated class in it.

(c) Create an assertion as follows:

CREATE ASSERTION TeachTwo CHECK ((SELECT COUNT (*) FROM Facult F, Class C WHERE F.fid = C.fid GROUP BY C.fid HAVING COUNT (*) <2) =0)

(d) Create an assertion as follows:

```
CREATE ASSERTION NO TeachThree CHECK ( (SELECT COUNT (*) FROM Facult F,
Class C WHERE F.fid =C.fid AND F.depid ≠ 33 GROUP BY C.fid HAVING COUNT
(*) >3) = 0)
```

(e) Create an assertion as follows:

CREATE ASSERTION InMath101 CHECK ((SELECT COUNT (*) FROM Student S WHERE S.snum NOT IN (SELECT E.snum FROM Enrolled E WHERE E.cname = 'Math101')) =0)

(f)The Class table should be modified as follow:

CREATE TABLE Class (name CHAR(20), meets_at TIME, room CHAR(10) , fid INTEGER PRIMAR KEY (name), FOREIGN KEY (fid) REFERENCES Faculty), CHECK ((SELECT MIN (meets_at) FROM Class) <> (SELECT MAX (meets_at) FROM Class)))

(g)The Class table should be modified as follows:

CREATE TABLE Class (name CHAR (20), meets_at TIME, room CHAR(10), fid INTEGER, PRIMARY KEY (name), FOREIGN KEY (fid) REFERENCES Faculty), CHECK ((SELECT COUNT (*) FROM (SELECT C.room, C.meets FROM Class C GROUP BY C.room, C.meets HAVING COUNT (*) > 1)) = 0))

(h)The Faculty table should be modified as follows:

CREATE TABLE Faculty (fid INTEGER, fname CHAR (20), deptid INTEGER, PRIMARY KEY (fnum), CHECK ((SELECT MAX (*) FROM (SELECT COUNT (*) FROM Faculty F GROUP BY F.deptid <2* (SELECT MIN (*) FROM (SELECT COUNT (*) FROM Faculty F GROUP BY F.deptid))))

(i)The Faculty table should be modified as follows:

CREATE TABLE Faculty (fid INTEGER, fname CHAR(20), deptid INTEGER, PRIMARY KEY (fnum), CHECK ((SELECT COUNT (*) FROM Faculty F GROUP BY F.deptid HAVING COUNT (*) > 10) =0))

(j) This constraint cannot be done because integrity constraints and assertions only affect the content of a table, not how that content is manipulated

(k) The Student table should be modified as follows:

CREATE TABLE Student (snum INTEGER, sname CHAR (20), major CHAR(20), level CHAR (20), age INTEGER, PRIMARY KEY (snum) FROM Enrolled E, Student S WHERE S.snum = E.snum AND S.major = 'Math'))

(L) Create an assertion as follows:

CREATE ASSERTION MoreCSMajors CHECK ((SELET COUNT (E.cname) FROM Enrolled E, Student S WHERE S.snum = E.snum AND S.major = 'CS') > (SELECT COUNT (E.cname) FROM Enrolled E, Student S WHERE S.snum = E.snum AND S.major = 'Math'))

(m) Create an assertion as follows:

CREATE ASSERTION MoreEnrolledThanMath CHECK ((SELECT COUNT (E.snum) FROM Enrolled E, Facutly F, Class C WHERE E.cname = C.name AND C.fid = F.fid AND F.deptid = 33) > (SELECT COUNT (E.snum) FROM Student S WHERE S>major = 'Math'))

(n) The student Table should be modified as follows:

CREATE TABLE Student (Snum INTEGER, sname CHAR(20), major CHAR(20), level CHAR(20) age INTEGER, PRIMARY KEY (snum), CHECK ((SELECT COUNT (S.snum) FROM Student S WHERE S.major = 'CS') >0))

(o) Create an assertion as follows:

```
CREATE ASSERTION NotSameRoom CHECK ( (SELECT COUNT (*)  FROM Faculty
F1, Facutly F2, Class C1, Class C2  WHERE F1.fid = C1.fid  AND F2.fid = C2.fid
AND C1.room = C2.room  AND F1.deptid ≠ F2.deptid) = 0)
```