New York University
Computer Science Department
Courant Institute of Mathematical Sciences

Homework #5 – Solutions

Course Title: Database Systems **Course Number:** CSCI-GA.2433-001
Instructor: Jean-Claude Franchitti

4.6 - Repeat Exercise 4.5, but use the AIRLINE schema of Figure 3.8.

Answer:

The following referential integrity constraints should hold:
FLIGHT_LEG.(FLIGHT_NUMBER) --> FLIGHT.(NUMBER)
FLIGHT_LEG.(DEPARTURE_AIRPORT_CODE) --> AIRPORT.(AIRPORT_CODE)
FLIGHT_LEG.(ARRIVAL_AIRPORT_CODE) --> AIRPORT.(AIRPORT_CODE)
LEG_INSTANCE.(FLIGHT_NUMBER, LEG_NUMBER) -->
FLIGHT_LEG.(FLIGHT_NUMBER, LEG_NUMBER)
LEG_INSTANCE.(AIRPLANE_ID) --> AIRPLANE.(AIRPLANE_ID)
LEG_INSTANCE.(DEPARTURE_AIRPORT_CODE) --> AIRPORT.(AIRPORT_CODE)
LEG_INSTANCE.(ARRIVAL_AIRPORT_CODE) --> AIRPORT.(AIRPORT_CODE)
FARES.(FLIGHT_NUMBER) --> FLIGHT.(NUMBER)
CAN_LAND.(AIRPLANE_TYPE_NAME) --> AIRPLANE_TYPE.(TYPE_NAME)
CAN_LAND.(AIRPORT_CODE) --> AIRPORT.(AIRPORT_CODE)
AIRPLANE.(AIRPLANE_TYPE) --> AIRPLANE_TYPE.(TYPE_NAME)
SEAT_RESERVATION.(FLIGHT_NUMBER, LEG_NUMBER, DATE) -->
LEG_INSTANCE.(FLIGHT_NUMBER, LEG_NUMBER, DATE)
One possible set of CREATE TABLE statements to define the database is given below.

```
CREATE TABLE AIRPORT ( AIRPORT_CODE CHAR(3) NOT NULL,
NAME VARCHAR(30) NOT NULL,
CITY VARCHAR(30) NOT NULL,
STATE VARCHAR(30),
PRIMARY KEY (AIRPORT_CODE) );
CREATE TABLE FLIGHT ( NUMBER VARCHAR(6) NOT NULL,
AIRLINE VARCHAR(20) NOT NULL,
WEEKDAYS VARCHAR(10) NOT NULL,
PRIMARY KEY (NUMBER) );
CREATE TABLE FLIGHT_LEG ( FLIGHT_NUMBER VARCHAR(6) NOT NULL,
LEG_NUMBER INTEGER NOT NULL,
DEPARTURE_AIRPORT_CODE CHAR(3) NOT NULL,
SCHEDULED_DEPARTURE_TIME TIMESTAMP WITH TIME ZONE,
ARRIVAL_AIRPORT_CODE CHAR(3) NOT NULL,
```

```
SCHEDULED_ARRIVAL_TIME TIMESTAMP WITH TIME ZONE,
PRIMARY KEY (FLIGHT_NUMBER, LEG_NUMBER),
FOREIGN KEY (FLIGHT_NUMBER) REFERENCES FLIGHT (NUMBER),
FOREIGN KEY (DEPARTURE_AIRPORT_CODE) REFERENCES
AIRPORT (AIRPORT_CODE),
FOREIGN KEY (ARRIVAL_AIRPORT_CODE) REFERENCES
AIRPORT (AIRPORT_CODE) );
CREATE TABLE LEG_INSTANCE ( FLIGHT_NUMBER VARCHAR(6) NOT NULL,
LEG_NUMBER INTEGER NOT NULL,
LEG_DATE DATE NOT NULL,
NO_OF_AVAILABLE_SEATS INTEGER,
AIRPLANE_ID INTEGER,
DEPARTURE_AIRPORT_CODE CHAR(3),
DEPARTURE_TIME TIMESTAMP WITH TIME ZONE,
ARRIVAL_AIRPORT_CODE CHAR(3),
ARRIVAL_TIME TIMESTAMP WITH TIME ZONE,
PRIMARY KEY (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE),
FOREIGN KEY (FLIGHT_NUMBER, LEG_NUMBER) REFERENCES
FLIGHT_LEG (FLIGHT_NUMBER, LEG_NUMBER),
FOREIGN KEY (AIRPLANE_ID) REFERENCES
AIRPLANE (AIRPLANE_ID),
FOREIGN KEY (DEPARTURE_AIRPORT_CODE) REFERENCES
AIRPORT (AIRPORT_CODE),
FOREIGN KEY (ARRIVAL_AIRPORT_CODE) REFERENCES
AIRPORT (AIRPORT_CODE) );
CREATE TABLE FARES ( FLIGHT_NUMBER VARCHAR(6) NOT NULL,
FARE_CODE VARCHAR(10) NOT NULL,
AMOUNT DECIMAL(8,2) NOT NULL,
RESTRICTIONS VARCHAR(200),
PRIMARY KEY (FLIGHT_NUMBER, FARE_CODE),
FOREIGN KEY (FLIGHT_NUMBER) REFERENCES FLIGHT (NUMBER) );
CREATE TABLE AIRPLANE_TYPE ( TYPE_NAME VARCHAR(20) NOT NULL,
MAX_SEATS INTEGER NOT NULL,
COMPANY VARCHAR(15) NOT NULL,
PRIMARY KEY (TYPE_NAME) );
CREATE TABLE CAN_LAND ( AIRPLANE_TYPE_NAME VARCHAR(20) NOT NULL,
AIRPORT_CODE CHAR(3) NOT NULL,
PRIMARY KEY (AIRPLANE_TYPE_NAME, AIRPORT_CODE),
FOREIGN KEY (AIRPLANE_TYPE_NAME) REFERENCES
AIRPLANE_TYPE (TYPE_NAME),
FOREIGN KEY (AIRPORT_CODE) REFERENCES
AIRPORT (AIRPORT_CODE) );
CREATE TABLE AIRPLANE ( AIRPLANE_ID INTEGER NOT NULL,
TOTAL_NUMBER_OF_SEATS INTEGER NOT NULL,
AIRPLANE_TYPE VARCHAR(20) NOT NULL,
PRIMARY KEY (AIRPLANE_ID),
FOREIGN KEY (AIRPLANE_TYPE) REFERENCES AIRPLANE_TYPE (TYPE_NAME) );
CREATE TABLE SEAT_RESERVATION ( FLIGHT_NUMBER VARCHAR(6) NOT NULL,
LEG_NUMBER INTEGER NOT NULL,
LEG_DATE DATE NOT NULL,
SEAT_NUMBER VARCHAR(4),
CUSTOMER_NAME VARCHAR(30) NOT NULL,
CUSTOMER_PHONE CHAR(12),
PRIMARY KEY (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE, SEAT_NUMBER),
```

FOREIGN KEY (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE) REFERENCES
LEG_INSTANCE (FLIGHT_NUMBER, LEG_NUMBER, LEG_DATE));

4.8 Write appropriate SQL DDL statements for declaring the LIBRARY relational database schema of Figure 4.6. Specify the keys and referential triggered actions.

Answer:

One possible set of CREATE TABLE statements is given below:
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));

4.11 - Specify the updates of Exercise 3.11 using the SQL update commands.

Answer:

Below, we show how each of the updates may be specified in SQL. Notice that some of these updates violate integrity constraints as discussed in the solution to Exercise 5.10, and hence should be rejected if executed on the database of Figure 5.6.

(a) Insert < 'Robert', 'F', 'Scott', '943775543', '21-JUN-42', '2365 Newcastle Rd, Bellaire, TX', M, 58000, '888665555', 1 > into EMPLOYEE.
INSERT INTO EMPLOYEE
VALUES ('Robert', 'F', 'Scott', '943775543', '21-JUN-42', '2365 Newcastle Rd, Bellaire, TX', M, 58000, '888665555', 1)

(b) Insert < 'ProductA', 4, 'Bellaire', 2 > into PROJECT.
INSERT INTO PROJECT
VALUES ('ProductA', 4, 'Bellaire', 2)

(c) Insert < 'Production', 4, '943775543', '01-OCT-88' > into DEPARTMENT.
INSERT INTO DEPARTMENT
VALUES ('Production', 4, '943775543', '01-OCT-88')

(d) Insert < '677678989', null, '40.0' > into WORKS_ON.
INSERT INTO WORKS_ON
VALUES ('677678989', NULL, '40.0')

(e) Insert < '453453453', 'John', M, '12-DEC-60', 'SPOUSE' > into DEPENDENT.
INSERT INTO DEPENDENT
VALUES ('453453453', 'John', M, '12-DEC-60', 'SPOUSE')

(f) Delete the WORKS_ON tuples with ESSN= '333445555'.
DELETE FROM WORKS_ON
WHERE ESSN= '333445555'

(g) Delete the EMPLOYEE tuple with SSN= '987654321'.
DELETE FROM EMPLOYEE
WHERE SSN= '987654321'

(h) Delete the PROJECT tuple with PNAME= 'ProductX'.
DELETE FROM PROJECT
WHERE PNAME= 'ProductX'

(i) Modify the MGRSSN and MGRSTARTDATE of the DEPARTMENT tuple with DNUMBER= 5 to '123456789' and '01-OCT-88', respectively.
UPDATE DEPARTMENT
SET MGRSSN = '123456789', MGRSTARTDATE = '01-OCT-88'
WHERE DNUMBER= 5

(j) Modify the SUPERSSN attribute of the EMPLOYEE tuple with SSN= '999887777' to '943775543'.

```
UPDATE EMPLOYEE
SET SUPERSSN = '943775543'
WHERE SSN= '999887777'
```

(k) Modify the HOURS attribute of the WORKS_ON tuple with ESSN= '999887777' and PNO= 10 to '5.0'.

```
UPDATE WORKS_ON
SET HOURS = '5.0'
WHERE ESSN= '999887777' AND PNO= 10
```

5.5 - Specify the following additional queries on the database of Figure 3.5 in SQL. Show the query results if applied to the database of Figure 3.6.

(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 we want the number of *male* employees in each department rather than all employees (as in Exercise 5.4a). Can we specify this query in SQL? Why or why not?

Answers:

(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) The query may still be specified in SQL by using a nested query as follows (not all implementations may support this type of query):

```
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

5.6 - Specify the following queries in SQL on the database schema of Figure 1.2.

(a) Retrieve the names and major departments of all straight-A students (students who have a grade of A in all their courses).
(b) Retrieve the names and major departments of all students who do not have any grade of A in any of their courses.

Answer:

```
(a) SELECT Name, Major
FROM STUDENT
WHERE NOT EXISTS ( SELECT *
FROM GRADE_REPORT
WHERE StudentNumber= STUDENT.StudentNumber AND NOT(Grade='A'))

(b) SELECT Name, Major
FROM STUDENT
WHERE NOT EXISTS ( SELECT *
FROM GRADE_REPORT
WHERE StudentNumber= STUDENT.StudentNumber AND Grade='A' )
```

5.7 - In SQL, specify the following queries on the database specified in Figure 3.5 using the concept of nested queries and the concepts described in this chapter.

a. Retrieve the names of all employees who work in the department that has the employee with the highest salary among all employees.
b. Retrieve the names of all employees whose supervisor's supervisor has '888665555' for Ssn.
c. Retrieve the names of employees who make at least $10,000 more than the employee who is paid the least in the company.

Answers:

```
a)  SELECT LNAME FROM EMPLOYEE WHERE DNO =

        ( SELECT DNO FROM EMPLOYEE WHERE SALARY =

            ( SELECT MAX(SALARY) FROM EMPLOYEE) )

b)  SELECT LNAME FROM EMPLOYEE WHERE SUPERSSN IN
( SELECT SSN FROM EMPLOYEE WHERE SUPERSSN = '888665555' )

c)  SELECT LNAME FROM EMPLOYEE WHERE SALARY >= 10000 +
( SELECT MIN(SALARY) FROM EMPLOYEE)
```