CS4347 : Database SystemsHomework Assignment 4

Matthew McMillianmgm160130@utdallas.edu

October 7, 2018

1. Repeat Exercise 6.5, but use the AIRLINE database schema of Figure 5.8.

   a.) Flight_number has a referential constraint that needs to be checked / monitored when editing the database. Leg_number shares Flight_leg and Leg_instance. Airplane_type_name also shares a referential constraint.

   b.)

```
CREATE TABLE AIRPORT (
            Airport_code      VARCHAR(10)  NOT NULL,
            Name              VARCHAR(15)  NOT NULL,
            City              VARCHAR(15)  NOT NULL,
            State             VARCHAR(15)  NOT NULL,

            PRIMARY KEY  Airport_code
);

CREATE TABLE FLIGHT (
            Flight_number     VARCHAR(10)  NOT NULL,
            Airline           VARCHAR(15)  NOT NULL,
            Weekdays          VARCHAR(15)  NOT NULL,

            PRIMARY KEY  Flight_number
);

CREATE TABLE FLIGHT_LEG (
            Flight_number               VARCHAR(10)  NOT NULL,
            Leg_number                  VARCHAR(10)  NOT NULL,
            Depature_airport_code       VARCHAR(10)  NOT NULL,
            Scheduled_depature_time     DATE NOT NULL,
            Arrival_airport_code        VARCHAR(15)  NOT NULL,
            Scheduled_arrival_time      DATE NOT NULL,

            PRIMARY KEY  Flight_number ,  Leg_number ,
            FOREIGN KEY  Flight_number  REFERENCES  FLIGHT
);
```

```
CREATE TABLE LEG_INSTANCE (
        Flight_number               VARCHAR(10) NOT NULL,
        Leg_number                  VARCHAR(10) NOT NULL,
        Date                        DATE NOT NULL,
        Number_of_avaliable_seats NUMBER NOT NULL,
        Airplane_id                 VARCHAR(15) NOT NULL,
        Depature_airport_code       VARCHAR(10) NOT NULL,
        Departure_time              DATE     NOT NULL,
        Arrival_airport_code        VARCHAR(15) NOT NULL,
        Arrival_time                DATE     NOT NULL,

        PRIMARY KEY Flight_number, Leg_number, Date,
        FOREIGN KEY Flight_number REFERENCES FLIGHT,
        FOREIGN KEY Leg_number REFERENCES FLIGHT_LEG
);

CREATE TABLE FARE (
        Flight_number               VARCHAR(10) NOT NULL,
        Fare_code                   VARCHAR(15) NOT NULL,
        Amount                      NUMBER NOT NULL,
        Restrictions                VARCHAR(15) NOT NULL,

        PRIMARY KEY Flight_number, Fare_code,
        FOREIGN KEY Flight_number REFERENCES FLIGHT
);

CREATE TABLE AIRPLANE_TYPE (
        Airplane_type_name          VARCHAR(10) NOT NULL,
        Max_Seats                   NUMBER NOT NULL,
        Company                     VARCHAR(15) NOT NULL,


        PRIMARY KEY Airplane_type_name
);

CREATE TABLE CAN_LAND (
        Airplane_type_name          VARCHAR(10) NOT NULL,
        Airport_code                VARCHAR(10) NOT NULL,


        PRIMARY KEY Airplane_type_name,
        FOREIGN KEY Airplane_type_name REFERENCES AIRPLANE_TYPE
);
```

```
CREATE TABLE AIRPLANE (
        Airplane_id             VARCHAR(10) NOT NULL,
        Total_number_of_seats   NUMBER NOT NULL,


        PRIMARY KEY Airplane_id
);

CREATE TABLE SEAT_RESERVATION (
        Flight_number           VARCHAR(10) NOT NULL,
        Leg_number              VARCHAR(10) NOT NULL,
        Date                    DATE NOT NULL,
        Seat_number             NUMBER  NOT NULL,
        Customer_name           VARCHAR(15) NOT NULL,
        Customer_phone          VARCHAR(15) NOT NULL


        PRIMARY KEY Flight_number, Leg_number, Date, Seat_number
);
```

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

```
CREATE TABLE BOOK (
          Book_id           VARCHAR(15)  NOT NULL,
          Title             VARCHAR(30)  NOT NULL,
          Publisher_name    VARCHAR(15)  NOT NULL,

         PRIMARY KEY  Book_id ,
         FOREIGN KEY  Publisher_name  REFERENCES  PUBLISHER
 );

CREATE TABLE BOOK_AUTHORS (
          Book_id           VARCHAR(15)  NOT NULL,
          Author_name       VARCHAR(15)  NOT NULL,

         PRIMARY KEY  Book_id ,  Author_name ,
         FOREIGN KEY  Book_id  REFERENCES BOOK
 );

CREATE TABLE PUBLISHER (
          Name              VARCHAR(15)  NOT NULL,
          Address           VARCHAR(15)  NOT NULL,
          Phone             VARCHAR(10)  NOT NULL,

         PRIMARY KEY  Name
 );

CREATE TABLE BOOK_COPIES(
          Book_id           VARCHAR(15)  NOT NULL,
          Branch_id         VARCHAR(15)  NOT NULL,
          No_of_copies      VARCHAR(15)  NOT NULL,

         PRIMARY KEY  Book_id ,  Branch_id ,
         FOREIGN KEY  Book_id  REFERENCES BOOK,
         FOREIGN KEY  Branch_id  REFERENCES  LIBRARY_BRANCH
 );

CREATE TABLE BOOK_LOANS (
          Book_id           VARCHAR(15)  NOT NULL,
          Branch_id         VARCHAR(15)  NOT NULL,
          Card_no           VARCHAR(15)  NOT NULL,
          Date_out          DATE      NOT NULL,
          Due_out           DATE      NOT NULL,
        PRMIMARY KEY Book_id, Branch_id,
        FOREIGN KEY Book_id REFERENCES BOOK,
        FOREIGN KEY Branch_id REFERENCES LIBRARY_BRANCH,
        FOREIGN KEY Card_no REFERENCES BORROWER
);

CREATE TABLE LIBRARY_BRANCH(
        Branch_id        VARCHAR(15) NOT NULL,
        Branch_name      VARCHAR(15) NOT NULL,
        Address          VARCHAR(15) NOT NULL,

        PRIMARY KEY Branch_id
);

CREATE TABLE BORROWER (
        Card_no          VARCHAR(15) NOT NULL,
        Name             VARCHAR(15) NOT NULL,
        Address          VARCHAR(20) NOT NULL,
        Phone            VARCHAR(10) NOT NULL

        PRIMARY KEY Card_no
);
```

3. How can the key and foreign key constraints be enforced by the DBMS? Is the enforcement technique you suggest difficult to implement? Can the constraint checks be executed efficiently when updates are applied to the database?

   The key constraints can be enforced by adding triggers onto each foreign and primary key. This type of technique is not difficult to implement since when a constraint is violated an error is thrown, however it would require the person who is using and DML queries to know how to fix their query. When updates are applied to the database, as long as all the constraints are not violated then there should not be any problems to efficiently update the database.

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

   a.) Retrieve the names of all senior students majoring in 'COSC' (Computer Science)

   ```
   SELECT S.Name FROM Student S WHERE S.Major=COSC AND S.Class=4;
   ```

   b.) Retrieve the names of all courses taught by professor King in 85 and 86.

   ```
   SELECT C.Course_name FROM COURSE C, SECTION S WHERE C.Course_number
   = S.Course_number AND (S.Year=85 OR S.Year=86);
   ```

   c.) For each section taught by professor King, retrieve the course number, semester, year, and number of students who took the section.

   ```
   SELECT S.Course_number, S.Semester, S.Year, GR.Student_number
    FROM SECTION S, GRADE_REPORT GR WHERE S.Instructor='King' AND
    GR.Section_identifier=S.Section_identifier;
   ```

   d.) Retrieve the name and transcript of each senior student (Class=5) majoring in COSC. Transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.

   ```
   SELECT C.Course_name, C.Course_number, C.Credit_hours, S.Semester,
   S.Year, GR.Grade FROM STUDENT ST, COURSE C, SEMESTER S,
   GRADE_REPORT GR WHERE ST.Class=5 AND
   ST.Student_number=GR.Student_number AND
   GR.Section_identifier=S.Section_identifier;
   ```

   e.) Retrieve the names and major departments of all straight A students (students who have a grade of A in all their courses)

   ```
   SELECT S.Name S.Major FROM STUDENT ST, GRADE_REPORT
   GR WHERE NOT EXISTS ( SELECT * FROM GRADE_REPORT
   WHERE GR.Student_number= ST.Student_number AND NOT(Grade='A'))
   ```

   f.) Retrieve the names and major departments of all students who do not have any grade of A in any of their courses.

```
SELECT S.Name S.Major FROM STUDENT ST, GRADE_REPORT
GR WHERE NOT EXISTS ( SELECT * FROM GRADE_REPORT
WHERE GR.Student_number= ST.Student_number AND (Grade='A'))
```

5. 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.

   ```
   INSERT INTO STUDENT (Name, Student_number, Class, Major)
   VALUES ('Johnson', 25, 1, 'Math');
   ```

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

   ```
   UPDATE STUDENTS SET Class=2 WHERE Name='Smith';
   ```

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

   ```
   INSERT INTO COURSE (Course_name, Course_number, Credit_hours,
    Department) VALUES ('Knowledge␣Engineering', 'CS4390', 3, 'CS');
   ```

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

   ```
   DELETE FROM STUDENT WHERE Name='Smith' AND Student_number=17;
   ```

6. Write SQL statements to create a table EMPLOYEE_BACKUP to back up the EMPLOYEE table shown in Figure 5.6.

```
CREATE TABLE EMPLOYEE_BACKUP AS SELECT * FROM EMPLOYEE;
```