Chapter 6 Review Questions
Due: October 8, 2020 @ 11.59pm

1. How does SQL allow implementation of the entity integrity and referential integrity constraints described in Chapter 3? What about referential triggered actions?
   SQL allows implementation of the entity integrity by using the PRIMARY KEY and UNIQUE clause. Referential integrity is maintained by using the FOREIGN KEY clause.
   - Referential triggered actions can be specified by the designer, by using the SET NULL, CASCADE, and SET DEFAULT clauses. The default action is to reject an update or delete operation.
2. Write appropriate SQL DDL statements for declaring the LIBRARY relational database schema of Figure 6.6. Specify the keys and referential triggered actions.
   NOT THE ENTIRE SOLUTION

```
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 );
```

3. How can the key and foreign key constraints be enforced by the DBMS?
   One possible technique that is often used to check efficiently for the key constraint is to create an index on the combination of attributes that form each key (primary or secondary). Before inserting a new record (tuple), each index is searched to check that no value currently exists in the index that matches the key value in the new record. If this is the case; the record is inserted successfully.
   For checking the foreign key constraint, an index on the primary key of each referenced relation will make this check relatively efficient. Whenever a new record is

inserted in a referencing relation, its foreign key value is used to search the index for the primary key of the referenced relation, and if the referenced record exists, then the new record can be successfully inserted in the referencing relation.

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 'cs' (computer science).
    ```
    SELECT Name
    FROM STUDENT
    WHERE Major='CS'
    ```
    b. Retrieve the names of all courses taught by Professor King in 2007 and 2008.
    ```
    SELECT CourseName
    FROM COURSE, SECTION
    WHERE COURSE.CourseNumber=SECTION.CourseNumber AND
    Instructor='King'
    AND (Year='07' OR Year='08')
    ```
    Another possible SQL query uses nesting as follows:
    ```
    SELECT CourseName
    FROM COURSE
    WHERE CourseNumber IN ( SELECT CourseNumber
    FROM SECTION
    WHERE Instructor='King' AND (Year='07' OR Year='08') )
    ```
    c. For each section taught by Professor King, retrieve the course number, semester, year, and number of students who took the section.
    ```
    SELECT CourseNumber, Semester, Year, COUNT(*)
    FROM SECTION, GRADE_REPORT
    WHERE Instructor='King' AND SECTION.SectionIdentifier=
    GRADE_REPORT.SectionIdentifier
    GROUP BY CourseNumber, Semester, Year
    ```
    d. Retrieve the name and transcript of each senior student (Class = 4) majoring in CS. A transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.
    ```
    SELECT Name, CourseName, C.CourseNumber, CreditHours,
    Semester, Year, Grade
    FROM STUDENT ST, COURSE C, SECTION S, GRADE_REPORT G
    WHERE Class= 4 AND Major='COSC' AND
    ST.StudentNumber=G.StudentNumber AND
    G.SectionIdentifier=S.SectionIdentifier AND
    S.CourseNumber=C.CourseNumber
    ```
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
    VALUES ('Johnson', 25, 1, 'MATH')
    ```
    b. Change the class of student 'Smith' to 2.
    ```
    UPDATE STUDENT
    SET CLASS = 2
    WHERE Name='Smith'
    ```

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

```
INSERT INTO COURSE
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 StudentNumber=17
```

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.

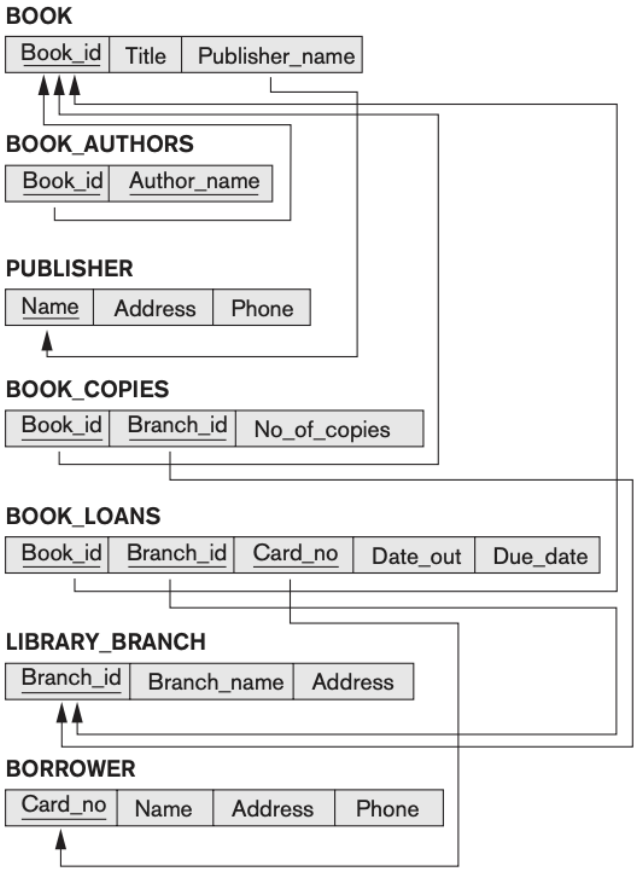

Figure 6.6
A relational database schema for a LIBRARY database.