Tutorial 5

Exercise 1

Consider the following relational schema and briefly answer the questions that follow:

Emp(*eid:* integer, *ename:* string, *age:* integer, *salary:* real)
Works(*eid:* integer, *did:* integer, *pct time:* integer)
Dept(*did:* integer, *budget:* real, *managerid:* integer)

1. Define a table constraint on Emp that will ensure that every employee makes at least $10,000.
2. Define a table constraint on Dept that will ensure that all managers have *age* > 30.
3. Define an assertion on Dept that will ensure that all managers have *age* > 30. Compare this assertion with the equivalent table constraint. Explain which is better.
4. Write SQL statements to delete all information about employees whose salaries exceed that of the manager of one or more departments that they work in. Be sure to ensure that all the relevant integrity constraints are satisfied after your updates.

1. **Define a table constraint on Emp that will ensure that every employee makes at least $10,000.**

```
CREATE TABLE Emp (eid INTEGER,
                  ename CHAR(10),
                  age INTEGER,
                  salary REAL,
                  PRIMARY KEY (eid),
                  CHECK (salary >= 10000)
                  )
```

2. Define a table constraint on Dept that will ensure that all managers have age > 30.

```
CREATE TABLE Dept (did INTEGER,
                   buget REAL,
                   managerid INTEGER,
                   PRIMARY KEY (did),
                   FOREIGN KEY (managerid) REFERENCES Emp,
                   CHECK((SELECT E.age
                              FROM Emp E, Dept D)
                              WHERE E.eid=D.managerid)>30)
```

3. Define an assertion on Dept that will ensure that all managers have age > 30

1 option.

```
CREATE TABLE Dept (did INTEGER,
                   budget REAL,
                   managerid INTEGER,
                   PRIMARY KEY (did))
```

The condition is checked only when the Dept relation is being updated.

2 option.

```
CREATE ASSERTION managerAge
       CHECK ((SELECT E.age
                   FROM Emp E, Dept D
                   WHERE E.eid = D.managerid ) > 30 )
```

It checks for potential violation of the assertion whenever one of the relations is updated.

4. To write such statements, it is necessary to consider the constraints defined over the tables. We will assume the following:

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


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


CREATE TABLE Dept ( did INTEGER,
                    buget REAL,
                    managerid INTEGER,
                    PRIMARY KEY(did),
                    FOREIGN KEY(managerid) REFERENCES Emp, ON DELETE SET NULL)
```

Exercise 5.8 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, *fname:* 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 class 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 Math101.

(f) The room in which the earliest scheduled class (i.e., the class with the smallest *meets 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.

(l) 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.

(m) The total enrollment in courses taught by faculty in the department with *deptid=33* is greater than the number of Math majors.

(n) There must be at least one CS major if there are any students whatsoever.

(o) Faculty members from different departments cannot teach in the same room.

1. The SQL statements needed to create the tables:

```
CREATE TABLE Students( snum INTEGER,
                       sname CHAR(20),
                       major CHAR(20),
                       level CHAR(20),
                       age INTEGER,
                       PRIMARY KEY (snum))

                                  CREATE TABLE Faculty ( fid INTEGER,
                                                         fname CHAR(20),
                                                         deptid INTEGER,
                                                          PRIMARY KEY (fnum))
CREATE TABLE Class  ( name  CHAR(20),
                      meets_at TIME,
                      room CHAR(20),
                      fid INTEGER,
                      PRIMARY KEY(name),
                      FOREIGN KEY (fid) REFERENCES Faculty )

CREATE TABLE Enrollment ( snum INTEGER.
                           cname CHAR(20),
                           PRIMARY KEY (snum,cname),
                           FOREIGN KEY (snum) REFERENCES Student ,
                           FOREIGN KEY (cname) REFERENCES Class  )
```

Every class has a minimum enrollment of 5 students and a maximum enrollment of 30 students.

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

b) **At least one class meets in each room.**

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) Every faculty member must teach at least two courses.

```
CREATE ASSERTION Teach Two
                    CHECK((SELECT COUNT(*)
                                FROM Faculty F, Class C
                                WHERE F.fid = C.fid
                                GROUP BY C.fid
                                HAVING COUNT(*)<2) = 0)
```

d) Only faculty in the department with *deptid=33* teach more than three courses.

```
CREATE ASSERTION NoTeachThree
                CHECK((SELECT COUNT(*)
                        FROM Faculty F, Class C
                        WHERE F.fid = C.fid AND F.deptit ≠ 33
                        GROUP BY C.fid
                        HAVING COUNT(*)>3) = 0)
```

e) Every student must be enrolled in the course called Math101.

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

f) The room in which the earliest scheduled class (i.e., the class with the smallest *meets at* value) meets should not be the same as the room in which the latest scheduled class meets.

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

g) Two classes cannot meet in the same room at the same time.

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

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.

```
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
                                       GOUP BY F.deptid))))
```

i) No department can have more than 10 faculty members.

```
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) A student cannot add more than two courses at a time (i.e., in a single update).

This constraint cannot be done because integrity constraints and assertions only affect the content of a table not how the content is manipulated.

k) The number of CS majors must be more than the number of Math majors.

```
CREATE TABLE Students( snum INTEGER,
                      sname CHAR(20),
                      major CHAR(20),
                      level CHAR(20),
                      age INTEGER,
                      PRIMARY KEY (snum),
                       CHECK (( SELCT COUNT(*)
                                    FROM Students S
                                    WHERE S.major='CS'>
                                          (SELECT COUNT(*)
                                                FROM Students S
                                                WHERE S.major='Math')))
```

l) 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.

```
CREATE ASSERTION MoreCSMajor
        CHECK( (SELECT COUNT(E.cname)
                        FROM Enrollemtn E,Students S
                        WHERE S.snum=E.snum AND S.major>'CS')>
                (SELECT COUNT(E.cname)
                        FROM Enrollemtn E,Students S
                        WHERE S.snum=E.snum AND S.major>'Math'))
```

m) The total enrollment in courses taught by faculty in the department with *deptid=33* is greater than the number of Math majors.

```
CREATE ASSERTION MoreEnrolledMath
        CHECK((  SELECT  COUNT(E.snum)
                         FROM Enrollment E, Faculty F, Class C
                         WHERE E.cname=C.name AND
                                  C.fid=F.fid AND F.deptid=33)>
                (SEELCT COUNT(E.snum)
                         FROM Student S
                         WHERE S.Major='Math'))
```

n) There must be at least one CS major if there are any students whatsoever.

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

o) Faculty members from different departments cannot teach in the same room.

```
CREATE ASSERTION NotSameRoom
        CHECK((  SELECT COUNT(*)
                        FROM Faculty F1, Faculty 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)
```