CS 631: DATA MANAGEMENT SYSTEMS DESIGN

ASSIGNMENT 3

EXERCISE 1 (Constraints in SQL)

Consider the following database schema:

STUDENTS (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.
Express each of the following integrity constraints in SQL unless it is implied by the primary and foreign key constraint; if the constraint cannot be expressed in SQL, say so.

1. No faculty member from department number 5 can teach more than four courses

```
CREATE ASSERTION NoMoreThanFourCourses
        CHECK (NOT EXISTS ( SELECT FID, COUNT(NAME)
                                    FROM FACULTY F, CLASS C
                                    WHERE F.FID = C.FID AND DEPTID = 5
                                     GROUP BY FID
                                     HAVING COUNT(NAME) > 4 ) ) ;


CHECK ((4 >= ALL (SELECT COUNT(NAME)
                        FROM FACULTY F, CLASS C
                        WHERE F.FID = C.FID AND DEPTID = 5
                        GROUP BY FID) );
```

2. The number of CS majors must be more than the number of math majors.

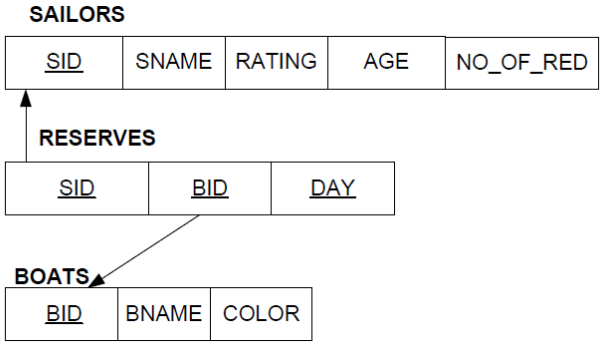
```
CREATE ASSERTION CSMoreThanMath
        CHECK ((SELECT COUNT(*)
                    FROM STUDENT S
                    WHERE S.MAJOR = 'CS')
                    >
                    (SELECT COUNT(*)
                     FROM STUDENT S
                    WHERE S.MAJOR = 'Math')
        );
```

3. No student should enroll in more than 2 classes offered by the same faculty.

```
CREATE ASSERTION NoMoreThan2SameFacultyCourse
        CHECK (NOT EXISTS (SELECT E.SNUM, C.FID, COUNT(E.CNAME)
                                FROM ENROLLED E, CLASS C
                                WHERE C.NAME=E.CNAME
                                GROUP BY E.SNUM, C.FID
                                HAVING COUNT (E.CNAME) > 2));


CHECK (2 >= ALL (SELECT COUNT (CNAME)
                        FROM ENROLLED, CLASS
                        WHERE NAME = CNAME
                        GROUP BY SNUM, FID));
```

EXERCISE 2 (Triggers)
Consider the following database schema:



The meaning of these relations is straightforward. Primary key attributes are underlined. Thus SID is the primary key for SAILORS, BID is the primary key for BOATS, and all three attributes of RESERVES together form the primary key of RESERVES. Arrows indicate foreign keys. Attribute NO_OF_RED records the number of reservations of red boats by a sailor. Write (a) an SQL row level trigger and (b) an SQL statement level trigger that maintain the value of attribute NO_OF_RED every time a reservation is made.

(a) Row level trigger

```
CREATE TRIGGER NO_OF_REDS

AFTER INSERT ON RESERVES
FOR EACH ROW

WHEN ((SELECT COLOR
        FROM BOATS
        WHERE BID = NEW.BID) = 'red')

UPDATE SAILORS
SET NO_OF_RED = NO_OF_RED + 1
WHERE SID = NEW.SID
```

(b) Statement level trigger

```
CREATE TRIGGER NO_OF_REDS_S

AFTER INSERT ON RESERVES
FOR EACH STATEMENT
REFERENCING NEW TABLE AS N

WHEN (EXISTS( SELECT *
              FROM BOATS, N
              WHERE N.BID = BOATS.BID AND COLOR = 'red'))
```

```
UPDATE SAILORS S
SET NO_OF_RED = NO_OF_RED + (SELECT COUNT(*)
                             FROM BOATS B, N
                             WHERE N.BID = B.BID AND B.COLOR = 'red' AND
                              S.SID = N.SID)
WHERE S.SID IN (SELECT DISTINCT N.SID
                FROM BOATS B, N
                WHERE N.BID = B.BID AND B.COLOR = 'red')
```