Classroom Exercise

1. Consider the following relational schema:

Reader(RDNR, Surname, Firstname, City, Birthdate)
Book(ISBN, Title, Author, NoPages, PubYear, PublisherName)
Publisher(PublisherName, PublisherCity)
Category(CategoryName, BelongsTo)
Copy(ISBN, CopyNumber, Shelf, Position)
Loan(ReaderNr, ISBN, Copy, ReturnDate)
BookCategory(ISBN, CategoryName)

BelongsTo refers to which parent categories the current category belongs to. Each book has a specific ISBN, and many copies of a book might be available under the same ISBN. A reader may borrow the same copy for multiple times, and each instance is recorded by its ReturnDate. All the parent categories that a book belongs to are stored in the table BookCategory.

Formulate the following queries in SQL.

(a) Which author has written the maximum number of books?

Solution:

```
SELECT Author, COUNT(ISBN) AS numberbooks
FROM Book
GROUP BY Author
HAVING numberbooks >= ALL(
SELECT COUNT(ISBN) FROM Book
GROUP BY Author);
```

(b) Which readers have borrowed at least one book (by ISBN, not copies) from the author “Philip S. Yu”, but have not borrowed all the books (by ISBN, not copies) from the author “Philip S. Yu”?

Solution:

```
SELECT Firstname, Surname
FROM Loan L, Reader R, Book B
WHERE R.RDNR = L.ReaderNr AND
L.ISBN = B.ISBN AND
Author = 'Philip S. Yu'
GROUP BY Firstname, Surname
HAVING COUNT(L.ISBN) <
(SELECT COUNT(ISBN)
FROM Book
WHERE Author = 'Philip S. Yu');
```

2. Suppose we are maintaining a database of articles published in our newspaper, the Straits Times. We have the following schema (where keys are underlined):

Article (issueID, articleID, author, title)

Citation (articleID, issueID, citedArticleID, citedIssueID)

WordAppears (wordID, issueID, articleID, position)

WordIs (wordID, wordText)

Issue (issueID, date, howManyDistributed)

Assume that dates can be compared using comparison operators (<, >, =). Assume that position is an index specifying where the word appears (1 = first word, 2 = second, etc.). Write the following queries in SQL.

(a) Find the most-cited article(s) in the newspaper's history.

Solution:

```
SELECT      citedIssueID, citedArticleID
FROM        Citation
GROUP BY    citedIssueID, citedArticleID
HAVING COUNT(*) >= ALL
                    (SELECT     COUNT(*)
                    FROM        Citation
                    GROUP BY    citedArticleID, citedIssueID)
```

(b) Find the number of citations per author for "senior" authors (i.e., an author who has at least one article that was published 10 or more years ago).

Solution:

```
SELECT      a.author, COUNT(*)
FROM        Article a, Citation c
WHERE       a.issueID = c.citedIssueID AND a.articleID = c.citedArticleID
AND EXISTS
                    (SELECT     *
                    FROM        Article a2, Issue i
                    WHERE       a2.issueID = i.issueID AND
                                a2.author = a.author AND
                                Year(getDate()) – Year(i.date) >= 10)
GROUP BY a.author;
```

3. The relational database schema for a Car-insurance company is given below.

person (driver-id, name, address)

car (license, year, model)

accident (report-number, location, date)

owns (driver-id, license)

participated (report-number, driver-id, license, damageamount)

employee (person-name, street, city)

works (person-name, company-name, salary)

company (company-name, city)

manages (person-name, manager-name)

(a) Modify the database so that Mark now lives in "Newstreet, Newtown" (i.e. the street changes to Newstreet and city changes to Newtown for Mark).

Solution:

```
UPDATE   employee
SET      city = 'Newtown' AND street = 'Newstreet'
WHERE    person-name = 'Mark';
```

(b) Give all employees of "FaceMatch" a 10 percent salary raise.

Solution:

```
UPDATE   works
SET      salary = salary*1.1
WHERE    company-name = 'FaceMatch';
```

(c) Give all managers of "FaceMatch" a 10 percent salary raise.

Solution:

```
UPDATE   works
SET      salary = salary*1.1
WHERE    person-name IN (
             SELECT manager-name
             FROM manages)
AND      company-name = 'FaceMatch';
```

4. Consider the relation schema: STUDENT(name, ssn, majorcode, gpa).

Suppose the following view is defined on it: STUDENT-NO-GPA(name, ssn,majorcode).

(a) Is it possible to insert a tuple ('X', 123, 'CMSC') into the view? If yes, what should be the result of it? If not, why not?

Solution:

Allowed. Insert ('X', 123, 'CMSC', null) into student. This will be disallowed if "gpa" is not allowed to be null however.

(b) Now consider the following view.

```
CREATE VIEW MAJOR-AVG AS
SELECT majorcode, avg(gpa)AS avggpa
FROM STUDENT
GROUP BY majorcode;
```

Is it possible now to insert a tuple ('CMSC', 3.2) into this view? If yes,what should be the result of it? If not, why not?

Solution:

Not allowed. It is not clear how to change the base tuples in response to a change in the average.

5. Consider the following schema of a product database:

Parts(pid: integer, pname: string, color: string)

Suppliers(sid: integer, sname: string, address: string)

Catalog(sid: integer, pid:integer, price: real)

The Catalog records that some Supplier sid supplies Part pid at a given price. Formulate each of the following integrity constraints as an SQL assertion.

(a) No Supplier may supply red and green Parts.

Solution:

```
CREATE ASSERTION NoRedAndGreenParts CHECK
(NOT EXISTS
(
        (SELECT     C.sid
        FROM  Catalog C, Parts P
        WHERE       C.pid = P.pid AND P.color = "red")
        INTERSECTS
        (SELECT     C.sid
        FROM  Catalog C, Parts P
        WHERE       C.pid = P.pid AND P.color = "green")
)
);
```

(b) For all Parts, no other Supplier has a lower price than the Supplier with "sid" = 1.

Solution:

```
CREATE ASSERTION NoLowerPriceThanSid1 CHECK
(NOT EXISTS
        (SELECT     *
        FROM  Catalog C1, Catalog C2
        WHERE       C1pid = C2.pid AND
                    C2.sid = 1 AND
                    C1.price < C2.price
)
);
```

6. A database system used by a university's examinations office has the following relations:

Exams (course, examDate, examTime)

Students (studentId, name)

registeredFor (student, course, examDate)

Attribute course is a six-character course code. A course cannot have more than one exam on the same date, and that exam will either be in the morning ('AM') or in the afternoon ('PM'). Attribute examDate is a date(e.g. '2014-12-18') and examTime is either 'AM' or 'PM'.

If a clash occurs for a student, special arrangements will be made for the student to sit the exam at another time on the same date. For example, if the clash is with another 'AM' exam, then a special exam will be arranged for that student in the afternoon ('PM'), and vice versa.

Write a trigger in SQL99 that, when a student registers for an exam that is at the same time on the same date as another exam for which they are already registered, adds a row to relation SpecialExams(student,course, examDate, examTime). Here, examTime should be the time of the specially arranged exam ('AM' or 'PM').

A row should be added to the registeredFor relation even when the student will sit a special exam.

Solution:

```
CREATE TRIGGER FixClash
BEFORE INSERT ON registeredFor
REFERENCING NEW ROW AS new
FOR EACH ROW
DECLARE et CHAR(2)
WHEN ( EXISTS (
        SELECT E1.course
        FROM (registeredFor NATURAL JOIN Exams) E1, Exams E2
        WHERE student = new.student
              AND E2.course = new.course
              AND E1.examDate = new.examDate
              AND E1.course <> E2.course
              AND E1.examDate = E2.examDate
              AND E1.examTime = E2.examTime
     ) )
BEGIN
    SELECT examTime INTO et
    FROM   Exams
    WHERE  course = new.course AND examDate = new.examDate;

    IF (et = 'AM') THEN
        INSERT INTO SpecialExams
          VALUES(new.student, new.course, new.examDate, 'PM');
    ELSE
        INSERT INTO SpecialExams
          VALUES(new.student, new.course, new.examDate, 'AM');
    END IF;
END;
```

Critical Thinking Exercise

7. Consider the following schema representing a database (primary keys areunderlined).

PRODUCT(model, manufacturer, type)

PC(model, speed, ram, hd, price)

LAPTOP(model, speed, ram, hd, screen, price)

PRINTER(model, color, type, price)

A PRODUCT is either a PC, a LAPTOP or a PRINTER and must have a tuple in the corresponding table. There is a foreign key constraint on the model of PCs, Laptops and Printers referencing the primary key model of PRODUCT.

Express the following queries in SQL. Your solution should be only one SQL statement.

Find the manufacturer(s) of computer (PC or laptop) with the highest available speed.

8. The following schema relates to holiday chalets that government employees may rent:

EMPLOYEE (emp-id, name, category, salary, age)

CHALET (chalet-id, location, price-per-day)

RENTAL (emp-id, chalet-id, start-date, no-of-days)

Note: Examples of category are Division I, Division II, etc.

Consider the view defined over the above schema:

```
CREATE VIEW X-VIEW
AS SELECT AVG(PRICE)
FROM CHALET C, RENTAL R,
WHERE C.CHALET-ID = R.CHALET-ID
AND NO-OF-DAYS > 7
GROUP BY LOCATION;
```

Describe what this view shows and give three reasons why it is generally considered not updatable.

9. Consider the following schema containing bank account information.

Primary Keys are in bold.

CUSTOMERS(**customer_name**, address)

ACCOUNTS(**account_number**, balance)

ACCOUNT_OWNERS(**customer_name**, **account_number**)

Write an assertion such that for every customer at least one of the following conditions holds true:

- He (or she) is owner of at most 5 accounts
- The sum of the balance of various accounts he (or she) owns is greater than $50,000.

10. Consider the following relation:

Articles (ID, title, journal, issue, year, startpage, endpage, TR-ID)

It contains information on articles published in scientific journals. Each article has a unique ID, a title, and information on where to find it (name of journal, what issue, and on which pages). Also, if results of an article previously appeared in a "technical report" (TR), the ID of this technical report can be specified. We have the following information on the attributes:

• For each journal, an issue with a given number is published in a single year.

• The endpage of an article is never smaller than the startpage.

• There is never (part of) more than one article on a single page.

Consider the following six queries on Articles:

a. SELECT title FROM Articles WHERE year=2005;

b. SELECT title FROM Articles WHERE endpage=100;

c. SELECT title FROM Articles WHERE year>1995 AND year<2000;

d. SELECT title FROM Articles WHERE journal='JACM' AND issue=55;

e. SELECT title FROM Articles WHERE issue=55 AND journal='JACM';

f. SELECT title FROM Articles WHERE endpage-startpage>50;

Indicate which of the above queries would likely be faster (based on the knowledge you have from the course), if all of the following indexes were created.

• CREATE INDEX Idx1 ON Articles(year,startpage);

• CREATE INDEX Idx2 ON Articles(startpage,endpage);

• CREATE INDEX Idx3 ON Articles(journal,issue,year);