Assignment 1

Professor Yanlei Diao

Question 1 [10 points]: Relational Algebra

(1) Division [4 points]
Please express **Division** using the five basic relational operators. As with the example in class, let us denote division by A/B where A = {x,y} and B={y}. For simplicity, assume that x, y are two attributes.

(2) Monotonicity [6 points]
A query or operator on relations is said to be **monotonic** if whenever we add a tuple to one of the input relations, the result contains all the tuples that it contained before adding the tuple, plus perhaps more tuples. That is, there is no way to remove tuples from the output by adding tuples to the input.

For each relational algebra operator below, state whether it is monotone.
(a) $\cup$
(b) $\cap$
(c) $-$
(d) $\times$
(e) σ
(f) π

Question 2 [10 points] Language Theory

(1) Can the following queries be expressed using **conjunctive queries**? If your answer is yes, write the conjunctive queries. If your answer is no, explain why.

(a) Find students who have taken 'Database Systems' but not 'Operating Systems'.
(b) Find the age of the youngest student who has taken 'Database Systems'.

(2) The following table lists direct cause effect relationships.

CauseEffect

Cause	Effect
a	d
b	d
c	e
d	f
d	g
e	x
g	x

Now we want to support the following query:

Among all causes that directly or indirectly contribute to the effect 'x', find other (direct or indirect) effects that they contribute to.

The SQL statement contains two queries:

```
Q1:   WITH RECURSIVE AllCauseEffect(cause, effect) AS
            ( SELECT * FROM CauseEffect
              UNION
              SELECT R1.cause, R2.effect
              FROM AllCauseEffect R1, CauseEffect R2
              WHERE R1.effect = R2.cause  )
```

Here, the `WITH` construct creates a recursive temporary relation to be used in the next query.

```
Q2:   SELECT C.cause, A.effect
      FROM ( SELECT DISTINCT R.cause
                FROM  AllCauseEffect R
                       WHERE R.effect = 'X')  AS C, AllCauseEffect A
      WHERE C.cause = A.cause and A.effect <> 'X';
```

Show the intermediate steps of computing the table AllCauseEffect when you execute the above query Q1 on the given table.

Iteration 1: please list all cause/effect pairs added to the AllCauseEffect, listed in lexicographical order;
Iteration 2: please list all pairs added to the AllCauseEffect, also in lexicographical order;
…
until Q1 completes.

Question 3 [32 points] Queries in Relational Algebra

Consider the following relational schema:

Suppliers(*sid*: integer, *sname*: string, *address*: string)
Parts(*pid*: integer, *pname*: string, *color*: string)
Catalog(*sid*: integer, *pid*: integer, *cost*: real)

The domain of each field is listed after the field name. Naturally, the Suppliers and Parts relations represent supplier entities and part entities. The Catalog relation lists the prices charged for parts by suppliers.

Write the following queries in relational algebra:

(1) Retrieve the name and address of the suppliers who supply some part.

(2) Retrieve the name and color of the parts supplied by the supplier "Perfunctory Parts".

(3) Retrieve the name of the suppliers who supply a red part that costs less than 100 dollars.

(4) Retrieve the name of the suppliers who supply a red part that costs less than 100 dollars or a green part that costs less than 100 dollars.

(5) Retrieve the name of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars.

(6) Retrieve the *pid* of the parts supplied by at least two different suppliers.

(7) Retrieve the name of suppliers that supply all red parts.

(8) Retrieve the name of suppliers that supply only the parts that cost less than 100 dollars.

Question 4 [48 points] SQL Queries using PostgreSQL

You will use PostgreSQL to execute queries on a sample dataset that consists of three tables conforming to the schema in Exercise 4. Both the dataset and instructions for connecting to the PostgreSQL server are available on the assignment web page.

Please take the following steps to complete this exercise.

Step 1: Place the dataset in an appropriate place in your home directory.

Step 2: Connect to the PostgreSQL server.

Step 3: Inside PostgreSQL, write a CREATE TABLE command for each file in the dataset. An example:

```
create table suppliers(
      sid int,
      sname varchar(30),
      address varchar(40),
      primary key (sid));
```

Step 4: Inside PostgreSQL, change your work directory to where your dataset is placed and load the dataset into corresponding tables. E.g.

```
yanlei=> \cd 'path-of-dataset'
yanlei=> \copy suppliers FROM suppliers.txt with delimiter as ','
yanlei=> \copy parts FROM parts.txt with delimiter as ','
yanlei=> \copy catalog FROM catalog.txt with delimiter as ','
```

You can use a SELECT query to check the content of each table. E.g.,

```
yanlei=> select * from parts;
```

Step 5: Now you are ready to submit SQL queries to retrieve the required information.

Write SQL expressions for each of the following queries and execute them:

Queries (1)-(8) are the same as in Exercise 1. They are copied below. Since duplicates can arise in the SQL data model (based on a multiset model), we require that **all the results be duplicates free**.

Q1) Retrieve the DISTINCT names of the suppliers who supply some part. Print the output in sorted order first by name and then by address.

Q2) Retrieve the name and color of the parts supplied by the supplier "Perfunctory Parts". Print the output in sorted order first by the name and then by color.

Q3) Retrieve the DISTINCT names of the suppliers who supply a red part that costs less than 100 dollars. Print the output in sorted order of name.

Q4) Retrieve the name of the suppliers who supply a red part that costs less than 100 dollars or a green part that costs less than 100 dollars. Print the output in sorted order of name.

Q5) Retrieve the name of the suppliers who supply a red part that costs less than 100 dollars and a green part that costs less than 100 dollars. Print the output in sorted order of name. (Note that MySQL does not support the INTERSECT operator. Please find another way to express it.)

Q6) Retrieve the DISTINCT *pid's* of the parts supplied by at least two different suppliers. Print the output in sorted order of pid. (Note in SQL the comparison operators are =, <>, <, >, <=, >=)

Q7) Retrieve the name of suppliers that supply all red parts. Print the output in sorted order of name.

Q8) Retrieve the DISTINCT names of suppliers that supply only the parts that cost less than 100 dollars. Print the output in sorted order of name.

In addition we have:

Q9) For those suppliers who supply at least two different parts, retrieve the name of the supplier and the total number of parts that this supplier provides. Print the output in order of the supplier name.

Q10) For every part that is supplied by someone, retrieve the name of the part, the maximum price, and average price across all suppliers that supply this part. Print the output in sorted order of part name.

Q11) For each part that is supplied by someone, retrieve the name of the part, the name of the supplier who charges the least for that part, and the cost asked by this supplier. Print the output in sorted order of the name of the part.

Q12) For each supplier who charges less for some part than the average cost of that part (averaged over all the suppliers who supply that part), retrieve the name of the supplier and the number of parts that he/she supplies under the average cost. Print the output in sorted order of the name of the supplier.