Virginia Tech. **CS 4604 – Introduction to DBMS**
Computer Science **Spring 2014, Prakash**

Homework 1: Relational Algebra and SQL

(due February 11th, 2014, 3:30pm, in class—hard-copy please)

Reminders:

a. Out of 100 points. Contains 4 pages.
b. Rough time-estimates: 2~4 hours.
c. Please type your answers. Illegible handwriting may get no points, at the discretion of the grader. Only drawings may be hand-drawn, as long as they are neat and legible.
d. There could be more than one correct answer. We shall accept them all.
e. Whenever you are making an assumption, please state it clearly.
f. Unless otherwise mentioned, you may use any SQL/RA operator seen in class/in textbook.
g. Unless otherwise specified, assume set-semantics for RA and bag-semantics for SQL.
h. Feel free to use the linear notation for RA and create intermediate views for SQL.
i. Lead TA for this HW: Qianzhou Du.

Q1. RA: Used Car Sale System [30 points]

The following relations keep track of used car sale information in a city:

Seller (***SID***: integer, SNAME: string, CITY: string, STATE: string, STREET: string, REP: integer)

Car (***CARID***: integer, ***POSTDATE:*** date, BRAND: string, MILEAGE: integer, YEAR: integer, SID: integer, ASKPRICE: integer)

Transaction (***BID: integer, CARID: integer, POSTDATE:*** date ACTUALPRICE: integer, TDATE: date)

Buyer (***BID***: integer, BNAME: string, CITY: string, STATE: string, STREET: string)

The Seller table contains every seller's ID, name, credit, address information, and reputation/rating (1-6). The Buyer table records the information of customers' IDs, names and address information. The Car table includes car IDs, posted selling date (when the seller puts it up for sale), brand (e.g. Toyota/Ford), mileage, production year, and owner's id and ask price. Note that a car can be bought and sold several times. The Transaction table records the transaction history: the buyer's id, the car id, the corresponding posted date, the transaction date and actual price.

Write each of the following queries in **Relational Algebra**:

Q1.1. (4 points) List all cars that have an asking price of less than 10,000 dollars.

Q1.2. (5 points) List names of the sellers who have sold/are trying to sell a Toyota car produced in 2009.

Q1.3. (5 points) List names of sellers whose reputation is more than 4.

Q1.4. (7 points) Help Robert Carlos find a used car nearby: List all cars and their owners who live in the same street as Robert Carlos.

Q1.5. (9 points) List all the 'available' cars with the cheapest asking price and mileage between 10,000 and 30,000 miles. An available car means that it has been put up for sale, but hasn't been sold yet.
Hint: First try finding all the available cars.

Q2. SQL: The Airline Flight System [30 points]

The following relations try to keep track of airline flight information:

Flights (***FLIGHTNO***: integer, ***FROM***: string, ***TO***: string, ***DISTANCE***: integer, ***DEPARTS***: time, ***ARRIVES***: time, ***PRICE***: integer, ***AID***: integer, ***CID***: integer)

Aircraft (***AID:*** integer, ***MANUFACT:*** string, ***MODEL:*** string)

AirlineComp(***CID***: integer, ***CNAME***: string, ***HADDRESS***: string)

Most of the fields are self-explanatory: the Aircraft table records the flight number, the departing and arriving city and scheduled times (all in PST), the distance between the two cities, price of a seat (assume all seats cost the same), which aircraft (id) it uses and which airline company it belongs to. The AirlineComp table, stores an airline company's id, name, and address of its headquarter. The Aircraft table includes the aircraft ids, model names, and manufacturers' information.

Write each of the following queries in **SQL**:

Q2.1. (3 points) List all the flights (only their numbers) that use a Boeing aircraft.

Q2.2. (4 points) List each airline company with the number of distinct aircraft models it owns.

Q2.3. (7 points) List the cheapest non-stop flights (just the flight numbers) from Roanoke to New York.

Q2.4. (7 points) For each pair of cities, list the average price of all non-stop flights that depart before 2:00 PM, and order them by the ascending average price. (Note: "Los Angeles, New York" and "New York, Los Angeles" should be considered different pairs).

Q2.1. (9 points) Mary wants to travel from Los Angeles to New York with no more than one layover. For example, she can choose a non-stop flight from Los Angeles to New York, or she can also take a flight to Chicago first, and then transit to another flight to New York. List all such valid itineraries from Los Angeles and which have total cost less than 800 dollars and have total travel time less than 10 hours. An example output would be like, say [(3134, null),

(3115, 4563)] where 3134 is a valid non-stop flight and 3115, 4563 is a valid one-stop schedule.
Note: Yes, it is unrealistic but assume all the flights depart and arrive in the same day (as measured by PST).
Hint: Make sure you know how to use a 'time' data item.

Q3. SQLite: Storing Publications [40 points]

This question is on a publications database that stores information for research publications. Download and install SQLite3 from http://www.sqlite.org

Warm-up

Follow the documentation and load the small sample database at:
http://courses.cs.vt.edu/~cs4604/Spring14/homeworks/hw1/cs4604-hw1.db

It has a table hw1sample which includes publications title, author, publication year, and publication type (such as incollection, book, and inproceedings: these denote different types of research publications e.g. 'inproceedings' indicates a publication which was published within the proceedings of a research conference). As you may have guessed, there are better schema designs to store this same information (say with less information duplication), but we have adopted this one for simplicity.

As a sanity check that you have the correct database, running the following command at a Unix/Linux/Cygwin prompt:

```
your-machine% sqlite3 cs4604-hw1.db 'select count(*) from hw1sample'
```

should return

```
27
```

We want to write SQL queries to do the following:

- Query1: Return the publications written by 'Won Kim'.
- Query2: Return the total number of publications in the database.

Larger CSV file

A bigger raw comma separated value (csv) file is given here:
http://courses.cs.vt.edu/~cs4604/Spring14/homeworks/hw1/dblp-sample.csv

It is a subset from the DBLP dataset (If you are curious, the official DBLP dataset is at http://www.informatik.uni-trier.de/~ley/db/). We want to write queries to do the following:

- Query3: List people who have co-authored at least 2 publications with 'Stefano Ceri'.

- Query4: List each type of publication and the number of publications of this type.

Life without SQL

Finally, in your favorite language (Python/Perl/Ruby/Java/C++ etc.) write code to do both queries above (Query3 and Query4) on the csv data file directly. Notice: the end-of-line convention is the DOS one (CRLF).

Deliverables

Q3.1. (4 points) The SQL query for Query1.

Q3.2. (4 points) The SQL query for Query2.

Q3.3. (4 points) The output of running Query1 in SQLite on the sample database.

Q3.4. (4 points) The output of running Query2 in SQLite on the sample database.

Q3.5. (4 points) The SQL query for Query3.

Q3.6. (4 points) The SQL query for Query4.

Q3.7. (4 points) The output of running Query3 on the csv file after loading it in SQLite.

Q3.8. (4 points) The output of running Query4 on the csv file after loading it in SQLite.

Q3.9. (4 points) Hard copy of your python/perl/etc code for doing Query3 on the raw csv file directly.

Q3.10. (4 points) Hard copy of your python/perl/etc code for doing Query4 on the raw csv file directly.

Hints

For loading the csv file,

- Again, the end-of-line convention follows the DOS format (CR LF).
- Use the .import and .mode csv commands of sqlite3 or check the tutorial at http://my.opera.com/cookyjar/blog/2009/04/20/importing-csv-data-file
- Again as a sanity check, the command
  your-machine% wc -l dblp-sample.csv

  should return

  10245 dblp-sample.csv