CPS 216 Spring 2005
Homework #1
Assigned: Thursday, January 20
Due: Tuesday, February 8

Note: This is a long homework. Start early!

Problem 1.

Consider a database containing information about bars, beers, and bar-goers.

Drinker (*<u>name</u>*, *address*), *Bar* (*<u>name</u>*, *address*), *Beer* (*<u>name</u>*, *brewer*),
Frequents (*<u>drinker</u>*, *<u>bar</u>*, *times_a_week*), *Likes* (*<u>drinker</u>*, *<u>beer</u>*), *Serves* (*<u>bar</u>*, *<u>beer</u>*, *price*).

Write the following queries in *relational algebra*. You may use expression trees (as seen in lecture slides) to improve readability.

(a) Find all drinkers who frequent James Joyce Pub.
(b) Find all bars that serve both Amstel and Corona.
(c) Find all bars that serve at least one of the beers Amy likes for no more than $2.50.
(d) For each bar, find all beers served at this bar that are liked by none of the drinkers who frequent that bar.
(e) Find all drinkers who frequent *only* those bars that serve some beers they like.
(f) Find all drinkers who frequent *every* bar that serves some beers they like.
(g) Find those drinkers who enjoy exactly the same set of beers as Amy.
(h) For each beer, find the bars that serve it at the lowest price.

Problem 2.

As discussed in class, the core operators in relational algebra are selection (σ_p), projection (π_L), cross product ($\times$), union ($\cup$), and difference ($-$). Show that the selection operator is necessary; that is, some queries that use the selection operator cannot be expressed using any combination of the other core operators.

Problem 3.

Functional dependencies in a relation R can be expressed as constraints of the form $Q = \varnothing$, where Q is a relational algebra query over R, and $\varnothing$ denotes an empty relation. Write the query Q for each of the following functional dependencies:

- $A \to BC$ in relation R (A, B, C, D).
- $AB \to C$ in relation R (A, B, C, D).

Problem 4.

Write queries (a)-(h) in Problem 1 in addition to the following one, all in *SQL*.

(i) For each beer, find its average price and popularity (measured by the number of drinkers who like it).

Please follow the instructions at
`http://www.cs.duke.edu/courses/spring05/cps216/faqs/login.html`
to log into `rack40` and set up the environment for DB2. Then, run
`/home/dbcourse/examples/db-beers/setup.sh`
to setup a database with some sample data. For the SQL database schema, please refer to the file `create.sql` in the same directory.

Write all queries in SQL in a file named `hw1-4.sql`. When you are done, run
`db2 -tf hw1-4.sql > hw1-4.out`
Then, print out files `hw1-4.sql` and `hw1-4.out` and turn them in together with the rest of the assignment.

Problem 5.

(a) This question is based on the paper "A Relational Model of Data for Large Shared Data Banks," by Codd. Suppose that an instance of *R* (*A*, *B*) is "joinable" with an instance of *S* (*B*, *C*). What functional dependencies must hold in order for the "join" of *R* with *S* to be unique (as discussed by Codd in Section 2.1.3)?

(b) This question is based on the paper "Weaving Relations for Cache Performance," by Ailamaki et al. The paper does not directly address the performance of PAX versus NSM in handling point-based queries and updates. More specifically, a point-based query or update has a `WHERE` condition that specifies the exact value of the primary key, e.g., "`SELECT name FROM Student WHERE SID = 142;`". Such queries and updates are quite common in OLTP (On-Line Transaction Processing) workloads. Without any experiment results, can you guess how PAX performs in comparison to NSM?

(c) This question is based on the paper "A History and Evaluation of System R," by Chamberlin et al. In Section 4 under the subsection titled "The SQL Language," authors introduced the notion of "outer-joins." Subsequently, syntax for outer-join was added to the SQL standard. Strictly speaking, however, the new syntax is not necessary (except perhaps from a user-friendly point of view). Show that you can write the outer-join between tables *R* (*A*, *B*) and *S* (*B*, *C*) in SQL without using the outer-join syntax.