CPS 216 Spring 2004
Homework #1
Assigned: Thursday, January 15
Due: Tuesday, February 3

Note: This is a long homework. Start early!

Problem 1 (12 points).

As discussed in class, the core operators in relational algebra are selection (σ_p), projection (π_L), cross product ($\times$), union ($\cup$), and difference ($-$).

(a) Show that the projection operator is necessary; that is, some queries that use the projection operator cannot be expressed using any combination of the other operators.
(b) Show that the union operator is necessary; that is, some queries that use the union operator cannot be expressed using any combination of the other operators.
(c) 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 operators.

Problem 2 (30 points).

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

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

Please follow the instructions at
`http://www.cs.duke.edu/courses/spring04/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 the queries below in SQL in a file named `hw1-2.sql`. When you are done, run
`db2 -tf hw1-2.sql > hw1-2.out`
Then, print out files `hw1-2.sql` and `hw1-2.out` and turn them in together with the rest of the assignment.

(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.

(i) For each beer, find its average price and popularity (measured by the number of drinkers who like it). Sort the output by average price.
(j) Every time when Dan goes to a bar, he buys a bottle of the most expensive beer he likes that is served at this bar. If there is more than one such beer, he buys just one of them. If the bar does not serve any beer he likes, he will not buy any beer. Find the amount of money Dan spends every week buying beers in bars.

Problem 3 (30 points).

Assume that in relational algebra, you can use built-in SQL predicates on strings, times, etc. in selection and join conditions; however, no SQL aggregation functions are allowed. Consider the queries in Problem 2:

- Which queries *cannot* be formulated in relational algebra?
- For each query that can be formulated in relational algebra, show the equivalent relational algebra query. You may draw expression trees (like those in the lecture) to improve readability.

Recommended: Test your relational algebra queries using the command-line tool `ra`! For instructions on using `ra`, please refer to
`http://www.cs.duke.edu/courses/spring04/cps216/faqs/ra.html`
If you do, submit a script of running `ra`, showing all the queries and answers.

Problem 4 (16 points).

Consider a relation R (A, B, C, D) with FD's $AB \rightarrow C$, $C \rightarrow D$, and $D \rightarrow B$.

(a) Show that $\{A, B\}$ is a key of R (remember a key has to be minimal).
(b) What are the other keys of R? (Hint: A must be in every key of R; why?)
(c) $D \rightarrow B$ is a BNCF violation. Using this violation, we decompose R into R_1 (B, D) and R_2 (A, C, D). What are the keys of R_1?
(d) What are the FD's that hold in R_1? Do not list them all; instead, give a set of FD's from which all other FD's in R_1 follow. This set of FD's is called a *basis*. When checking for BCNF violations, it suffices to check just the basis.
(e) Is R_1 in BCNF? Briefly explain why.
(f) What are the keys of R_2? (Hint: There is more than one.)
(g) What are the FD's that hold in R_2? Again, do not list them all; instead, give a basis.
(h) Is R_2 in BCNF? If yes, briefly explain why. Otherwise, decompose further until all decomposed relations are in BCNF, and then show your final results.

Problem 5 (12 points).

(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 "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.

(c) 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?