Assignment #5
Due Tuesday November 14, 5:00 PM

Problems to Work and Hand In

Several of the following questions relate to the "Beer Drinkers' Relational Database Schema" below:

```
Likes(drinker, beer)
Serves(bar, beer)
Frequents(drinker bar)
LocatedAt(bar, addr)
```

1. (20 pts.) Write relational-algebra expressions to answer the following queries about the Beer Drinkers' Database. You may express your queries either as linear expressions or as expression trees.

   (a) Find all the bars that serve a beer `Joe` likes.

   (b) Find all the beers served at bars that `Joe` frequents.

   (c) Find all the drinkers that frequent at least one bar that serves some beer that the drinker likes.

   (d) Find all drinkers that like at least two different beers.

   (e) Find all drinkers such that every bar they frequent serves only beers that they like. *Note*: For simplicity, assume that every drinker is mentioned in `Frequents` and therefore goes to at least one bar. In a more general version of ths query (which you are not asked to do), you would have to consider drinkers that frequented no bar, and therefore satisfied the query.

2. (20 pts.) Write datalog queries to answer the same queries as in Question 1. You may define any number of IDB predicates that you wish. Your answer, in each case, should be in an IDB predicate `Answer`.

3. (10 pts.) When asked to write queries in relational algebra for "find the bars that serve `Bud`," two programmers came up with the following two expressions.

   (a) $\pi_{bar}(\sigma_{beer=\text{Bud}}(\texttt{Serves}))$.

   (b) $\pi_{bar}(\sigma_{beer=\text{Bud}}(\texttt{Serves} \bowtie \texttt{LocatedAt}))$.

   Are these two expressions *equivalent*, meaning that regardless of what relations are the current instances of relations `Serves` and `LocatedAt`, the results of the two queries will be the same? Either argue why the two expressions always produce exactly the same answers, or give a counterexample to show they can produce different answers.

Fifth Step of Your Personal Database Application

(50 pts.) In Assignment #4 you loaded a small amount of data into the relations for your PDA using Sybase's bulk loading facility. For this assignment you will also use Sybase's bulk loading facility, but this time your data truly will be "bulk." Your assignment is to write a C or C++ program that creates large files of records in the format required for bulk loading into Sybase, and then to load data into *each of* your PDA relations.[1] For those of you who are using real data for your PDA, your C/C++ program will need to transform the data into files of records conforming to your PDA schema. The rest of you will write a C/C++ program to *fabricate* data. This means that your program will generate either random or non-random (e.g. sequential) records conforming to your schema.

It is OK if your data values — strings especially — are meaningless gibberish. The point of generating large amounts of data is so that you can experiment with a database of realistic size, rather than the small "toy" databases often used in classes. The data you generate and load should be on the order of:

- At least two relations with thousands of tuples
- At least one relation with hundreds of tuples

If the semantics of your application includes relations that are expected to be relatively small (e.g. schools within a university), it is fine to use some small relations, but please ensure that you have relations of the sizes prescribed above as well.

Note: While you can create meaningless values for all or most of your data, it is important for future assignments that *identical* values appear in appropriate situations. That is because in later assignments you will be writing SQL queries that join your relations. If the join attributes do not share values, then your joins will be empty and you will not get realistic results. For example, if your PDA involves students and courses, it is OK if a few students are enrolled in nonexistent courses, or some courses have no students, but it is essential that *most* of your courses have students and *most* students be enrolled in courses that actually exist.

Turn in your C/C++ code for generating or transforming data, a small sample of the records generated for each relation (5 or so records per relation), and a script showing the loading of your data into Sybase.

[1] In Assignment #4 we were happy with loading of some data into *one* of your relations — now you need to load them all.