CompSci 316 Fall 2019: Homework 1

100 points (6.25% of course grade) + 10 points extra credit
Assigned: Wednesday, August 28
Due: Monday, September 16

This homework should be done in parts as soon as relevant topics are covered in lectures. If you wait until the last minute, you might be overwhelmed.

You must turn in the required files electronically. Please read the "Help → Submitting Non-Gradiance Work" section of the course website, and follow the submission instructions for each problem carefully.

Problem 1 should be completed on a virtual machine (VM). Read the VM-related section under "Help" on the course website, and follow the instructions therein to get your VM running. (If you opt to run your VM on Google Cloud, you should have received an email from the instructor concerning Google Cloud credits.)

Problem 1 (60 points)

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 using RA, our homegrown relational algebra interpreter. To set up the sample database called `beers`, issue this command in your VM shell:

```
/opt/dbcourse/examples/db-beers/setup.sh
```

Then, type "`radb beers`" to run RA. See "Help → RA: A Relational Algebra Interpreter" on the course website for additional instructions on using RA, including syntax of relational operators.

Remember that in RA, you can see list the relations in our small sample database and inspect their contents (by simply issuing a query for a relation with no operators). Other useful features of RA include comments (which allow you to document and explain your queries) and views (which allow you to write complex queries in multiple steps). RA also supports various extensions to standard relational algebra, notably grouping and aggregation (which we will learn later in this course in the context of SQL); for this homework, however, ***do NOT use grouping and aggregation*** features of RA.

You should check that your queries return the intended answers on our sample database. For grading, however, your answers will be tested on other databases with the same schema but different contents, so your queries need to be correct ***in general*** to receive full credits.

As soon as you get a working solution for one part of this problem, say (a), record your query in a plain-text file named `a-query.txt` (replace "`a`" with "`b`", "`c`", and other parts as appropriate). Submit all query files.

An autograder will run automatically on your submission and give you a report of what it finds—the `db0` test database it uses is identical to the sample one you have, while `db1` is a hidden test database. You can use the autograder report to help you debug your queries and resubmit. If you cannot get a query to parse correctly or return the right answer, include your best attempt and explain it in comments, to earn possible partial credit. Remember to ***(re)submit all files*** in your final submission.

(a) Find names of all bars that *Eve* frequents.
(b) Find names and addresses of drinkers who frequent *Satisfaction* more than once per week.
(c) Find names of bars serving some beer *Amy* likes for strictly less than $2.75.
(d) Find names of all drinkers who like *Corona* but not *Budweiser*.
(e) For each beer that *Eve* likes, find the names of bars that serve it at the highest price. Format your output as list of (beer, bar) pairs.
(f) Find the ***cheapest and second cheapest*** drinks in town and where they are served; i.e., return the *serves* rows with the lowest and second lowest prices. If there are multiple *serves* rows with these two lowest prices, return all of them. (For the example, suppose the two lowest prices are $0.01 and $0.02, and there are 3 *serves* rows with $0.01 and 1 *serves* row with $0.02; then, the query should return a total of 4 rows.)
(g) For each drinker, find names of bars frequented by the drinker that serve none of the beers liked by the drinker. Format your output as list of (drinker, bar) pairs.
(h) Find names of all drinkers who frequent ***only*** those bars that serve some beers they like.
(i) Find names of all drinkers who frequent ***every*** bar that serves some beers they like.

Problem 2 (40 points)

An online used-car trading dot-com hires you to design a database for its Web site. The database will store information about used automobiles for sale.

- To register with the Web site, a user must provide name and email; email must be unique. To post a listing, the user must be either a dealer or an individual. For a dealer, we additionally require address and phone number. For an individual, we additionally require phone number.
- Each automobile for sale by a user has a VIN (vehicle identification number), a model (e.g., Camero), a make (e.g., Chevrolet), a year (e.g., 2010), a color (e.g., red), a mileage (e.g., 50,000 miles), a body style (e.g., coupe), and a listing price (e.g., 3000).
- Any registered user can post reviews about automobiles. Each review is about one particular model, make, and year (but not about a particular vehicle). One user may write several reviews about the same model, make, and year. We also would like to record the date of each review.

If you feel that some aspects of the above description are unclear, feel free to make additional, reasonable assumptions about the data. State any additional assumptions clearly in your answer. Also, keep in mind that there is no single "correct" design; if you think you are making a non-obvious design decision, please explain it briefly.

(a) Design an E/R diagram for this database. Very briefly explain the intuitive meaning of any entity and relationship sets. Do not forget to indicate keys and multiplicity of relationships, as well as ISA relationships and weak entity sets (if any), using appropriate notation.

(b) Design a relational schema for this database. (You can start by translating the E/R design.) You may ignore attribute types, and you do not need to show any sample data. Indicate all keys and non-trivial functional dependencies in the schema. Check if the schema is in BCNF. If not, decompose the schema into BCNF.

Prepare a single PDF file for submission. Put your answers to (a) and (b) on different pages.

Extra Credit Problem X1 (10 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 ($-$). Prove that the selection operator is necessary; that is, some queries that use this operator cannot be expressed using any combination of the other operators.

Hint: We are looking for a rigorous proof, not just intuition. For example, consider the argument that the union operator is necessary. The intuition is that union the only operator that lets you "add" tuples to a relation without widening it, but this argument alone is not rigorous, because one could use cross product to create more tuples and then use projection to get a narrower relation. A rigorous argument would be the following. Consider a database instance with two single-attribute relations $R = \{\langle 0 \rangle\}$ *and* $S = \{\langle 1 \rangle\}$*.* $R \cup S$ *would yield a relation with two tuples, but starting with* R *and* S*, repeated uses of other operators can never obtain a result relation with more than one tuple (using an inductive argument).*

Prepare a single PDF file for submission.