CompSci 316 Fall 2018: Homework #1

100 points (8.75% of course grade) + 10 points extra credit
Assigned: Thursday, August 30
Due: Tuesday, September 18

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.

For Problems 1, 3, and 5, you will use Gradiance. Gradiance is an online testing system that provides immediate feedback to your answers, and allows you to retry a problem multiple times until you get it right. Please read the "Help → Getting Started with Gradiance" section of the course website to get started. There is no need to turn in anything for these problems; your scores will be tracked automatically.

For other problems, you will 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 2 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. (You should have received an email from the instructor concerning Google Cloud credits, if you opt to run your VM on Google Cloud.)

Problem 1 (13 points)

Complete the Gradiance homework titled "Homework 1.1 (Relational Algebra Basics)." Note that some of the online exercises use English names of relational algebra operators instead of symbols.

Problem 2 (32 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*)

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 may be tested on other databases with the same schema but different contents, so your queries need to be correct i***n 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. 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.

(a) Find the names of beers that *James Joyce Pub* serves.
(b) Find the names of drinkers who frequent any bar serving *Corona*.
(c) Find the names and addresses of bars that serve *Corona* for no more than $3.00.
(d) Find the names of drinkers who frequent both *James Joyce Pub* and *Satisfaction*.
(e) Find the names of bars frequented by either *Ben* or *Dan*, but not both.
(f) Find all (*beer*, *bar*) pairs where *beer* is served exclusively at *bar*; no other bar serves the same beer.
(g) For each bar, find the drinker who frequents it the greatest number of times. Your output should be a list of (*bar*, *drinker*) pairs. If multiple drinkers tie for the most frequent visitor to a bar, list them all as separate 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 ***only*** those bars that serve ***only*** beers they like.
(j) Find all (*bar1*, *bar2*) pairs where the set of beer served at *bar1* is a proper subset of those served at *bar2*; i.e., *bar2* serves every beer that *bar1* serves and plus some more.

Problem 3 (15 points)

Complete the Gradiance homework titled "Homework 1.3 (E/R Design)."

Problem 4 (20 points)

You are tasked to design a database for the Pie-in-the-Sky Security Corporation, a brokerage firm that buys and sells stocks for its clients. The database needs to record the following information.

- Every person in the database has a unique SSN (social security number), a name, and an address.
- Each broker also has a phone number. Each broker (except for the "big boss") reports to another broker in the management chain. Multiple brokers can report to the same one.
- Clients own accounts, which are identified by unique account IDs. An account can have one or more owners, and a client can have any number of accounts. Each account is handled by exactly one broker.
- Stocks are identified their ticker symbols (e.g., AAPL), and we also track its latest price per share (but we don't track historical prices in this case).

- An account can hold shares from multiple stocks, and we track the number of shares for each stock currently held (but we don't need historical data in this case). A stock can be held by multiple accounts, or none at all.
- Each account can carry out a series of trades, each identified by the sequence number (unique within the account). Each trade is for one specific stock, and has a transaction type (buy or sell), a timestamp (of fulfillment), the number of shares involved, and the price per share.

If you think some aspects of the above are unclear, feel free to make additional, reasonable assumptions, but state them 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 as needed. 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.

Problem 5 (20 points)

Complete the Gradiance homework titled "Homework 1.5 (Relational Design Theory: FD)."

Extra Credit Problem X1 (5 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.

Extra Credit Problem X2 (5 points)

A parity query returns the empty relation $\emptyset$ when the input relation has an odd number of tuples, or otherwise the singleton relation $\{\langle\rangle\}$ (whose only tuple has no attributes, which can be created by projecting any non-empty relation onto an empty set of attributes). Prove that the parity query over a relation $R(A)$

cannot be written in relational algebra, where selection and join conditions are restricted to comparing attributes and/or constants using = and ≠, and projections are restricted to subsets of input attributes.

Prepare a single PDF file for submission.