Problem Set #1: Querying, Querying, Querying

Due: October 6, 2014, 10:59am

1 Comments about Grading

These problem sets will be graded roughly.

- If your answer is well written and contains a complete, correct answer for a question, you will get bonus points. These bonus points are rare.
- If you get basically everything right, but we find some mistake or gap in your answer, we will give you full credit. We hope to give you essentially full credit for each question.
- Partial credit will be given aggressively–so turn something in!

There will be provided solutions. As a result, problem set material is fair game for tests and quizzes.

2 Submission

The submission for this assignment will include one PDF file and several SQL files for the SQL problems. See the submission instructions at the end of this document for more details.

We encourage you to type your solutions using LaTeX. A sample LaTeX template is available on the website if you are unfamiliar with the format. If you don't want to bother with installing LaTeX, `http://www.writelatex.com` and `http://www.sharelatex.com` are great online LaTeX editors.

3 Relational Algebra Problems

The following two problems exercise relational algebra to make sure one understands how to use it. These questions also describe some limitations of relational algebra.

Problem 1 Consider a relation DrivingDistance(city1, city2, distance), that represents the length of a direct road between pairs of major cities. For simplicity, assume that only the shortest such road is in the table. Your goal is to write a relational algebra expression to determine whether one can drive from Palo Alto to Washington DC (simply, DC). (If not,

the expression should return the empty set, i.e., no tuples.) The first leg of your journey must start in Palo Alto and the last must terminate in DC, and all connections require city2 in one leg to be the same as city1 in the next leg.

It turns out that if you don't know in advance how many different cities can appear in the path, then it's not possible to write an expression to achieve our goal, using only standard relational operators. However, if we are told in advance that there can be no more than N different cities on our path then we can do it. Show such an expression. Note your expression will need to have a "..." of some form in it, but from the way you write it, we should be able to understand immediately how to instantiate the expression for a given N.

Problem 2 Now suppose we don't know in advance how many different cities can appear in the relation DrivingDistance. To help achieve our goal, let us extend basic relational algebra with a few additional constructs:

- Statements of the form "R := relational algebra expression", where R can be an existing relation that gets replaced, or a new one that gets created. If it's helpful, R can be referenced in the right-hand side expression.
- Loops of the form "While NotEmpty(R) Do sequence of statements ".
- Sequences of Statements and/or While loops.

Using these constructs, write a relational algebra expression that returns a single tuple (Palo Alto, DC) if it is possible to fly from Palo Alto to DC by stopping in any number of cities. If it is not possible to reach DC from Palo Alto then no tuples should be returned, but the expression should still terminate. You may assume that the relation DrivingDistance has a finite number of tuples, and therefore the number of cities is finite, but do not assume any specific upper limit on the number of cities. Since your solution may not be obvious to understand, please add an English comment to each statement or expression that explains what it does. Without comments, your solution will not be graded.

4 SQL

For all three SQL problems:

- We're looking for solutions that use the standard declarative constructs of SQL as covered in the lecture, videos, and optional readings. Try to write queries using those constructs, before resorting to nondeclarative constructs or esoteric features supported by some SQL systems. All of the problems can be solved using standard constructs.
- We strongly suggest using SQLite, MySQL, or some other SQL DBMS to test your solutions. It's an easy matter to download one of the open-source systems (see our Quick Guide), create and load a sample table, and play around. That's really the only way you can be assured your queries are working. That's also the only way we'll be assured they're working. To get full credit for these questions **include in your solution for each problem a transcript of running your SQL queries over a**

sample table that is large enough to verify correctness for all cases. Please include your data as well as your query results. Solid attempts at solutions that haven't been tested on a running system are still eligible for points.

- Finding a solution that can be expressed in a single standard declarative SQL query is not easy! If you're completely stuck, one approach you can try is to modularize a bit: run an initial query or two and put the results into temporary tables, then run a query over the temporary tables to get the final result. Once you've got that working, you can think about how to create a single query that has the same effect.

Problem 1 Consider a table Temperature(city,state,day,temp) containing the temperature in Fahrenheit for cities around the United States on days represented as integers 1 to 365. Attributes [city,state,day] together form a key.

(a) Write the simplest (i.e., shortest) SQL query you can come up with that returns the lowest, highest, and average temperature for a state on any day in any city. The schema of the result should be (state,low,high,avg), without duplicates.

(b) Write the simplest (i.e., shortest) SQL query you can come up with that returns the names of a city if the lowest temperature in that city is within 30 degrees of the highest temperature recorded across all cities.

(c) Your answer to part (b) probably uses aggregation. Do the same problem, but do not use aggregation. Again, keep your query as short as possible, but don't worry if it's longer than your solution for part (b).

Problem 2 Consider a one-attribute table T(A) containing an odd number of distinct positive integers. Write a SQL query to find the median of this table.

Problem 3 Consider a SQL database that is used to store a tree data structure. The database has one table: Parent(P,C). A tuple (pID,cID) in this table encodes the fact that the node with identifier pID is the parent of the node with identifier cID in the tree structure. There are no duplicates in the table. You can assume that the table faithfully represents a tree: every node except one (the root) has exactly one parent. You may also assume the maximum height of the tree is four: there's no path from the root to a leaf that's of length > 4. Do not make any other assumptions, such as a fixed or maximum number of children per parent, or a balanced tree structure (i.e., a uniform distance from root to leaf).

Let x and y be two nodes in the tree.

- Write a SQL query that determines if x is a descendant of y.
- Write a SQL query to determine the length of the path between x and y. Your SQL query will probably be very long.

5 Submission Instructions

To submit Problem Set 1, save your answers as a PDF (preferably typewritten using LaTeX or Microsoft Word, but a handwritten & scanned PDF is fine as long as it is legible).

Solutions to the SQL problems should be stored as problem_#.sql. Include comments and a transcript of running the queries in your PDF file.

When you are ready to submit, copy your files over to the `corn.stanford.edu` machines, and gather all of the following files in a single submission directory:

- Problem_Set_1.pdf
- problem_1.sql
- problem_2.sql
- problem_3.sql

Once your submission directory is properly assembled, **with no extraneous files**, execute the following script from your submission directory:

```
/usr/class/cs145/bin/submit-pset
```

Be sure to select "Pset1" when the script prompts you for which assignment you're submitting!

You may resubmit as many times as you like; however, only the latest submission and timestamp will be saved, and we will use your latest submission for grading your work and determining any late penalties that may apply. Submissions via email will not be accepted!