Overview

In this assignment we will be setting up a simple Postgres database in preparation for the next assignment. It won’t be anything fancy, but will help you get familiarized with setting up your environment and using basic Postgres and SQL commands.

Meet your DBMS

PostgreSQL is the Database Management System (DBMS) we’ll be using throughout the semester. It’s one of the most full-function open-source DBMSs on the market – in some aspects it’s more advanced than the leading commercial systems. And it began as a research project here at Berkeley so you know it’s good! During the semester you will not only be using and tuning Postgres, you’ll be modifying its internals with new functionality.

In your instructional account, we’ve got you set up already so that you can administer your own databases with Postgres. If you like, you can also set up Postgres on your own machine – it runs on most operating systems. See http://www.postgresql.org for more information.

What Machines to Use

All our assignments will be set up for the Solaris x86 machines in the instructional lab. (See the note at the end of this page if you want to use your own machines.) To see the list of Solaris x86 machines, log into any instructional machine and type “clients”, choose menu item 13, and look for machines labeled “Solaris 10 X86” (rhombus, pentagon, cube, sphere, po, torus.)

Throughout these instructions we will be using rhombus as our example machine, but you are encouraged to pick one of the others to balance the workload; simply replace the name of that machine in the instructions below.

Building our Database

Your first assignment is to set up a simple database, enter data, and query the database.

1. Initialize and create your database. In your CS186 account, start by issuing the command:

   % initdb

2. In order to be able to access your databases remotely, you will need to copy over some configuration files as follows:

   % cd ~/pgdata
3. The first of these files controls various features of the Postgres server, the second controls network access to the database. (If you’re curious about them, you can read more at http://www.postgresql.org/docs/8.2/interactive/config-setting.html and http://www.postgresql.org/docs/8.2/interactive/auth-pg-hba-conf.html respectively, but feel free to skip this for now.) You are now ready to start the server, and create your database:

% pg_ctl start
% createdb hwCountries
CREATE DATABASE

4. Now, start the Postgres command-line too, psql. It accepts SQL statements, and special control commands which start with backslash. You are encouraged to poke around in the help system via \h and \?. Here we’ll simply list the tables in our database via the “\dt” command.

% psql hwCountries
Welcome to psql 8.1.10, the PostgreSQL interactive terminal.
Type: \copyright for distribution terms
\h for help with SQL commands
? For help with psql commands
\g or terminate with semicolon to execute query
\q to quit
%

5. Now we haven’t quite discussed our application yet. For our current purposes, it suffices to say that our application will be similar to the CIA Factbook, in terms of storing information about each of the countries. Let’s begin by creating a table, ‘countries’, that holds pertinent information. We can enter SQL statements directly into postgres, so we will use the ‘CREATE TABLE’ command to build our table:

% CREATE TABLE countries (  
    name char(20),
    latitude int,
    longitude int,
    area int,
    population int,
    gdp int,
    gdpYear int);

6. However, this statement does not include any mention of a primary key. Select a primary key that will work best for this assignment (hint: read through the rest of the assignment), and revise the SQL statement to incorporate that.
7. Up until this point, we have provided step by step instructions in order to bootstrap this assignment. Now there are certain other tasks that must be completed. For some of these, you may need to refer to online SQL documentation.
   a. Create another table, ‘borders’, which is used to imply that two countries border each other. Ensure that the proper link exists between the two tables (Hint: Can an entry in the borders table exist without a link to an entry in the countries table?)
   b. Populate both tables with proper information for Germany, Netherlands, Belgium, Luxemburg, Poland, Czech Republic, Austria, France, and Switzerland. (For simplicity, assume that only these countries exist in our world).
   c. Use query to display all countries that border Germany.
   d. Use query to display all countries that have a population greater than 100 million.
   e. Use query to display all countries that have a population greater than 100 million AND border Germany.

**Deliverables**

Turn in all aspects of your work, which includes commands issued and output received, in text file “hw1a.txt.” Create a directory called “hw1a”, copy “hw1a.txt” in it, cd to “hw1a” and submit as follows:

```
% cd hw1a
% submit hw1a
```