CSCI 321 – Database Systems
Fall 2012
Access to Postgresql

I have installed the latest OpenSUSE RPM for the postgresql DBMS on `bg6.cs.wm.edu`. I have also created postgresql accounts for each student in the class who has send me their CS userid. Your postgresql userid is the same as your userid on the CS network. Your *initial* postgresql password is also the same as your userid.

The most complete documentation for postgresql may be found at the URL

http://www.postgresql.org/docs/9.1/interactive/index.html

It is important to note that the SQL implemented by postgresql (or any other DBMS) will differ from the ideal SQL of the text. The documentation at the above URL is the authoritative definition of SQL as implemented in postgresql.

Note there is no current assignment that involves the use of postgresql. I am making this available so you can play around for a while, if you are so inclined.

Change Your Password

To access your database, ssh to `bg6.cs.wm.edu` and login as usual. The first thing you need to do is to change your postgresql password using the following operations:

```
% psql
Password:  enter your initial postgresql password here
```

Postgreqsl will respond to a valid login with:

```
psql (9.1.3)
Type "help" for help.


userid=>
```

At this time you are interacting with `psql`, the interactive interface onto postgresql. You may enter `psql` commands here; they are the commands that start with a backslash. You may get information on the commands by entering the `\?` command, and by reading the `psql` manual page. The SQL command to change your password can be entered at the `psql` prompt:

```
userid=> alter user userid with password 'whatever';
ALTER ROLE
userid=> \q
```

Installing the Banking Example Database

I have set up a dummy database for each account, the database has the same name as your userid. It's the default database for apps running in your userid, but it really shouldn't be used. You are able to create (and delete) your own databases. For the remainder of the semester, please follow the following convention concerning database names: **name your database userid_dbname** where userid is your postgresql userid and dbname is the descriptive name of your database. For example, if I want to create a database to play around with the banking example, kearns_bank would be a good database name.

You can create a database from the shell command line with the `createdb` command.

```
% createdb kearns_bank
Password:
%
```

To convince myself that the database has actually been created, I can do:

```
% psql -l
Password:
                                  List of databases
      Name      |  Owner   | Encoding |   Collate   |    Ctype    |   Access privileges
----------------+----------+----------+-------------+-------------+-----------------------
 aablohm        | aablohm  | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
. . . . . . . . . .
 kearns         | postgres | UTF8     | en_US.UTF-8 | en_US.UTF-8 |
. . . . . . . . . .
(20 rows)
%
```

Now, to load up my bank database with data from the command line, using an "SQL script" (assuming my current working directory is the directory where the SQL script resides):

```
% psql -d kearns_bank -f make-banking.sql
Password:
LOTS OF OUTPUT
%
```

The following will let me get at my bank database to play around:

```
% psql -d kearns_bank
Password:
psql (9.1.3)
Type "help" for help.

kearns_bank=> \dt
          List of relations
 Schema |   Name    | Type  | Owner
--------+-----------+-------+--------
 public | account   | table | kearns
 public | borrower  | table | kearns
 public | branch    | table | kearns
 public | customer  | table | kearns
 public | depositor | table | kearns
 public | loan      | table | kearns
(6 rows)

kearns_bank=> select *
kearns_bank-> from branch;
 branch_name | branch_city | assets
-------------+-------------+---------
 Downtown    | Brooklyn    |  900000
 Redwood     | Palo Alto   | 2100000
 Perryridge  | Horseneck   | 1700000
 Mianus      | Horseneck   |  400000
 Round Hill  | Horseneck   | 8000000
 Pownal      | Bennington  |  300000
 North Town  | Rye         | 3700000
 Brighton    | Brooklyn    | 7100000
(8 rows)

kearns_bank=> \q
```

Finally, if I don't want to clutter up the disk with this database any longer, I can delete (drop) it from the command line:

```
% dropdb kearns_bank
Password:
DROP DATABASE
```

Notes

1. It is possible to connect to postgresql from any departmental system. Use

```
% psql -h bg6
```

to connect over the departmental network.

2. If you live off-campus, you can connect in two different ways:

   (a) From any system on which you can run an ssh client, you can ssh to `bg6.cs.wm.edu` to run `psql`.

   (b) You can install postgresql (v9.1) on your Linux, Mac, or Windows system. Most Linux distributions include postgreqsl as an optional installation; I have used the macports utility to build postgresql on some of my Macs; there are installation packages for Windows available at `http://www.postgresql.org/download/windows/`. Having installed postgresql, you can do the following to have your local `psql` connect to the postgresql server on `bg6`:

   In one window:

   ```
   ssh -L 63333:localhost:5432 userid@bg6.cs.wm.edu
   ```

   This will forward data sent to port 63333 on your local machine to port 5432 on `bg6`.

   In another window:

   ```
   psql -h localhost -p 63333 userid
   ```

   This directs `psql` to hook up to a postgresql server at port 63333 on your local machine. But the ssh command above forwards that connection to `bg6`:5432, the departmental postgresql server.

3. Database storage doesn't count against your disk quota. The disks on which the databases live on `bg6` are backed up nightly.