

SQL Assignment, CMSC724, Spring 2013

Instructor: Amol Deshpande

The assignment is to be done by yourself.

The goal of this assignment is to get you (re-)familiarized with relational databases, and specifically the standard query language for databases, SQL. We will use the open source PostgreSQL database system for this purpose. Installing the system on your machine will be an important part of the assignment.

Software installs often do not go smoothly; we strongly recommend you at least install and play with PostgreSQL well in advance. Generally speaking, we will likely not be able to help you with any installation issues, but sooner you contact us, better the chances.

1 Installing and Using PostgreSQL

The PostgreSQL website has binary packages and source code for installing it on a variety of platforms. You will be asked for the “superuser” password along the way during install – make sure to remember that. The current version of PostgreSQL is 9.2.2. You will find the detailed documentation at: <http://www.postgresql.org/docs/9.2/static/index.html>

You will be using PostgreSQL in a client-server mode. First you have to start the server in the background. Recall that the server is a continuously running process that listens on a specific port (the actual port would differ, and you can usually choose it when starting the server). In order to connect to the server, the client will need to know the port. The client and server are often on different machines, but for you, it may be easiest if they are on the same machine.

Using the “psql” client is the easiest – it provides a commandline access to the database. But there are other clients too. We will assume “psql” here.

After installing PostgreSQL, before starting the server, you must first “initialize” a database storage area on disk. This is the area where all the data files are stored (called “data directory” in PostgreSQL). This can be done using the “initdb” command, which takes in a data directory as the input. See initdb documentation for more details:
<http://www.postgresql.org/docs/current/static/app-initdb.html>

After initializing the database storage area, you can start the server. There are different ways to do it depending on the installation and the platform. On UNIX platforms, the command:

```
postgres -D /usr/local/pgsql/data
```

will start the server, with the data directory “/usr/local/pgsql/data”. See the documentation for more details:

<http://www.postgresql.org/docs/current/static/server-start.html>

Note: Both the above steps may be done during the install itself. E.g., on Macs, the 1-click installer asks for a location for the data directory (Default: /Library/PostgreSQL/9.2/data), initializes it, and starts the server.

After the server has started, you have to “create” a database. This is done using the “createdb” command. PostgreSQL automatically creates one database for its own purpose, called “postgres”. It is preferable you create a different database for your data. Here are more details on “createdb”:

<http://www.postgresql.org/docs/current/static/tutorial-createdb.html>

For the assignment, you can create a database called “olympics” (you can name it something else if you want). The rest of the commands will assume the database is called “olympics”.

Important: PostgreSQL server has a default superuser called “postgres”. You can do everything under that username, or you can create a different username for yourself. If you run a command (say createdb) without any options, it uses the same username that you are logged in under. However, if you haven’t created a PostgreSQL user with that name,

the command will fail. You can either create a user (by logging in as the superuser), or run everything as a superuser (typically with the option: “-U postgres”).

Once the database is created, you can connect to it. There are many ways to connect to the server. The easiest is to use the commandline tool called “psql”. Start it by:

```
psql -U postgres olympics
```

The above command gives the username “postgres” to connect to the database. You may be prompted for a password. “psql” takes quite a few other options: you can specify different user, a specific port, another server etc. See documentation: <http://www.postgresql.org/docs/current/static/app-psql.html>

Now you can start using the database.

- The psql program has a number of internal commands that are not SQL commands; such commands are often client and database specific. For psql, they begin with the backslash character, “\”. For example, you can get help on the syntax of various PostgreSQL SQL commands by typing: “\h”.
- “\d”: lists out the tables in the database.
- All commands like this can be found at: <http://www.postgresql.org/docs/current/static/app-psql.html>. “\?” will also list them out.
- To populate the database using the provided olympics dataset, use the following: “\i populate.sql”. For this to work, the populate.sql file must be in the same directory as the one where you started psql. This command creates the tables, and inserts the tuples. We will discuss the schema of the dataset in the next section.

2 Olympics Dataset

The dataset contains the details of the 2000 and 2004 Summer Olympics, for a subset of the games (“swimming” and “athletics”). More specifically, it contains the information about players, countries, events, and results. It only contains the medals information (so no heats, and no players who didn’t win a medal). The schema of the tables should be self-explanatory (see the “populate.sql” file). The data was collected from <http://www.databaseolympics.com/> and Wikipedia. Some things to note:

- The birth-date information was not available in the database, and that field was populated randomly.
- Be careful with the team events; the information about medals is stored by player, and a single team event gold gets translated into usually 4, but upto 6-7 “medal” entries in the Results table (for SWI and ATH events).
- If two players tie in an event, they are both awarded the same medal, and the next medal is skipped (ie., there are events without any silver medals, but two gold medals). This is more common in Gymnastics (the dataset does not contain that data anyway, but does have a few cases like that).
- The “result” for a match is reported as a “float” number, and its interpretation is given in the corresponding “events” table. There are three types: “seconds” (for time), “points” (like in Decathlon), “meters” (like in long jump).

You can load the database in psql using:

```
\i populate.sql
```

You may have to give an explicit path for populate.sql (tab-autocomplete works well in psql).

3 SQL

Queries in psql must be terminated with a semicolon. After populating the database, you can test it by running simple queries like:

```
select * from olympics;
```

We will not cover SQL in the class.

- The website for CMSC 424 (<http://www.cs.umd.edu/class/spring2012/cmssc424-0101/>) contains my slides for SQL, along with a set of sample queries.
- You can also look at an undergraduate textbook. Keep in mind the syntax of the commands can slightly vary, especially commands that use any advanced features.
- There are numerous online resource. PostgreSQL manual is a good place for introduction to SQL. <http://sqlzoo.net> also has many examples.

Here are some example queries on the olympics dataset and the SQL for them.

- Report the total number of medals won by M. Phelps over both olympics.

```
select * from players where name like '%Phelps%';
```

See that M. Phelps ID is PHELPMIC01.

```
select count(medal) from results where player_id = 'PHELPMIC01';
```

- Find the country with the highest population density (population/area-sqkm).

```
select name
from countries
where population/area_sqkm = (select max(population/area_sqkm) from countries);
```

Note that using a nested "subquery" (which first finds the maximum value of the density) as above is the most compact way to write this query.

- What was the duration of the 2004 Olympics (use startdate and enddate to find this) ?

```
select olympic_id, enddate - startdate + 1
from olympics o
where o.year = 2004;
```

There are many interesting and useful functions on the "date" datatype. See:

<http://www.postgresql.org/docs/current/interactive/functions-datetime.html>

- Write a query to add a new column called "country_id" to the IndividualMedals table (created during the assignment).

```
alter table IndividualMedals add country_id char(3);
```

- Initially the "country_id" column in the IndividualMedals table would be listed as empty. Write a query to "update" the table to set it appropriately.

```
update IndividualMedals im
set country_id = (select country_id
                  from players p
                  where p.player_id = im.player_id);
```

- **(WITH)** In many cases you might find it easier to create temporary tables, especially for queries involving finding "max" or "min". This also allows you to break down the full query and makes it easier to debug. It is preferable to use the WITH construct for this purpose. The syntax appears to differ across systems, but here is the link to PostgreSQL: <http://www.postgresql.org/docs/9.0/static/queries-with.html>
- **(LIMIT)** PostgreSQL allows you to limit the number of results displayed which is useful for debugging etc. Here is an example:

```
select * from Players limit 5;
```

4 Assignment Submission

See the provided text file for the queries. You should use the same text file for submission. The text file should be self-explanatory. The submission will be through the submit server.