Problem Set 1: SQL

Assigned: 9/14/2015
Due: 9/21/2015 11:59 PM
Submit to the 6.830 Stellar Site (https://stellar.mit.edu/S/course/6/fa15/6.830/homework/)
You may work in pairs on this problem set. Clearly indicate the name of your partner. Only one of you needs to submit on Stellar.

1 Introduction

The purpose of this assignment is to give you hands-on experience with the SQL programming language. SQL is a declarative language in which you specify the data you want in terms of its properties. This assignment focuses on the SELECT subset of SQL, which is all about querying data rather than modifying it.

We will be using a PostgreSQL server, which provides a standards-compliant SQL implementation. In reality, there are slight variations between the SQL dialects of different vendors (PostgreSQL, MySQL, SQLite, Oracle, Microsoft, etc.) —especially with respect to built-in functions. The SQL tutorial at [http://sqlzoo.net/](http://sqlzoo.net/) provides a good introduction to the basic features of SQL. After following this tutorial you should be able to answer most of the problems in this problem set.

We are using version 9.1 of Postgres, so the documentation at [http://www.postgresql.org/docs/9.1/static/](http://www.postgresql.org/docs/9.1/static/) may be useful as well. You may also wish to refer to Chapter 5 of “Database Management Systems.”

To access the server, you can log in to athena.dialup.mit.edu and start a session with:

```
psql -h geops.csail.mit.edu wikidb
```

You must be signed up to stellar for us to know you are in the class. If you prefer, you can log in from other machines provided they have a postgres client binary (aka `psql`) and you are authenticated to Kerberos with your MIT user name.

2 Dataset

The data for this assignment is a subset of the “simple english” version of wikipedia. This subset includes metadata on three basic wikipedia entities: pages, categories and images, as well as page-to-page, page-to-category and page-to-image link information. We have not kept the text of the articles nor their revision histories in this dataset. You can get a diagram of the tables at [https://www.mediawiki.org/wiki/Manual:Database_layout](https://www.mediawiki.org/wiki/Manual:Database_layout). Each of the tables we use has a detailed description online as well: for example, the page table is explained extensively at [https://www.mediawiki.org/wiki/Manual:Page_table](https://www.mediawiki.org/wiki/Manual:Page_table) and the page links table is explained in [https://www.mediawiki.org/wiki/Manual:Pagelinks_table](https://www.mediawiki.org/wiki/Manual:Pagelinks_table).

You can also get an idea of the data by visiting [http://simple.wikipedia.org](http://simple.wikipedia.org) and navigating it. The data may not match 100% of what you see in the server because our server has a subset of a past snapshot.

3 Using the Database

Once connected, there are two kinds of commands useful to a database user. The first kind are the `psql` client meta-command. The most important one, of course, is `\?`, which gives you help on meta-commands. There are two others that greatly help:

We can list the tables in the database with `\dt:`
And we can check the schema (recall, that the “schema” of a database is like a class definition in an object oriented language) of a given table with \d table_name:

The second class of useful commands are SQL commands. All SQL queries in PostgreSQL must be terminated with a semi-colon. For example, to get a list of all records in the page table, you would type:

```
SELECT * FROM page LIMIT 10;
```

This query requests a max 10 rows from the table. Using LIMIT in this manner one can explore the data small bits at a time. If you really wanted to produce all the records, though, the query is:

```
SELECT * FROM page;
```

You can use Ctrl+C to end a query that is taking too long (it is very possible to write such bad queries even unintentionally). Note that using the LIMIT keyword by itself offers no guarantee on which 10 rows from the result are returned, so do not assume an ordering.

Finally, you can change the way psql displays the result sets to suit you better. In particular, wrapped lines in less can make the output of wide tables hard to read. If this bothers you, you can try exiting the client (you can use Ctrl+D) and starting it again with the LESS env. variable set like this:

```
LESS=''-S'" psql -h geops.csail.mit.edu wikidb
```
4 Questions

For each question, please include both the SQL query and the result in your answer. The answers do not have to be one-liners: you can save the results of a previous query, if that is convenient to you, using create temp table. Also, if the query is taking too long then try changing it. All questions have solutions that run within seconds.

Note: mediawiki page names are made out of a namespace and a title (these are page_namespace and page_title components in the page table). For example, there is a “main” page :Mars as well as an associated talk page Talk:Mars and a category page for it as well at Category:Mars. We consider these three pages to have the same title, but different names, and to be different pages. We call he page with no prefix, :Mars, the “main” page for that title or the page in the “default” namespace. You can read more about it here: Manual:Namespace. For your convenience, we have defined a table namespace that translates the page_namespace ids into their string representations and vice-versa.

Clarifications:

• In a few of the queries we ask you to find articles. In those cases, the page name (page_namespace, page_title) is enough to identify an article. We do not need the whole row. Technically, the page_id is enough as well but we prefer the page name.

Q1. Find articles whose title contains the word ‘databases’ with either uppercase or lowercase ‘d’.

Q2. Find the 10 longest pages (their page ids and lengths) in Simple English Wikipedia. Are they main pages?

Q3. Find the full names of the results of the previous query (recall that a “name” is of the form namespace:title, so answering this query will require looking up information from both the page and namespace tables. It is okay for a name in the default namespace to be of the form :Title)

Q4. Find the name of the most commonly linked-to image.

Q5. Find the name of the article with the most outgoing links, and the number of links.

Q6. Find the sortkey (cl_sortkey) for the page with the most images. Note: a previous version of this question asked you to find the category title, either answer is okay. The queries are almost the same as well.

Q7. How many links are there from talk pages (namespace 1) to main pages?

Q8. Find the number of “broken” page links (ie, links where the destination page does not actually exist).

Q9. Find the 2 “most similar” articles in the category American_movie_actors, where the similarity of two articles is defined as the number of shared links. (Bonus: same query, but define the similarity as: (number of shared links in articles A1 and A2) * 2 / (number of links in A1 + number of links in A2)).

Q10. This is a multi-part question. In all of these questions, we are restricting our attention to all the pages in the English_language_movies category, and page links only between those pages. We exclude page names that exist only in the pagelinks but not in the page table.

(a) Find the 10 most popular main pages with category English_language_movies. We will measure the popularity of a page by the number of incoming links from any other page in this category.

(b) Find the 10 most popular pages based on the number of incoming links 2 hops away (i.e. a pattern page_1 → page_0 → our_page counts as one incoming link to our_page using this definition).
(c) How might you extend this to incoming links 3 hops away, 4 hops?

(d) (No need to write a query for this one). How about finding the connected components of the graph? Is the relational model appropriate for querying data structured as a graph? Are there language features it seems to be missing?