Lab 2 : Implementing "This database class @#$&@!" (and other useless rants)
CMSC 362
Marmorstein
Spring 2017
Due Friday (2/17/2017) by 11:59pm

The purpose of this lab is to reinforce the commands we learned in Lab 1 for building tables in SQL and to practice using SQL to build tables. We will do this by implementing a simple "rant" database which could serve as the back end to a blog or other web site.

Step 1. Setting Up

We are going to again use the Postgresql database server on torvalds. However, this time I want you to work in groups of no more than two people. I have created group accounts on torvalds and we will select groups in class. Your group username is groupX (where X is your group number) and your password is “trantor”.

You should change your group password by typing:

```
ALTER USER groupX ENCRYPTED PASSWORD ‘new_password’;
```

(But change ‘new_password’ to something you will both remember).

```
psql -h torvalds -U groupX -d groupX
```

To execute a database script on the server, you can use:

```
psql -h torvalds -U groupX -d groupX -f scriptName.sql
```

Create a folder named Lab2. In that folder create a file named "lab2.sql". All of your sql code will go in this file. At the top of your file *type two dashes followed by a space and then your name, the text "CMSC 362" and the words "Lab 2"*.

In this lab, we are going to introduce several new concepts, but you will need to figure out how to apply them for yourself. You may find the PostgreSQL documentation here helpful:

http://www.postgresql.org/docs/current/static/

Step 2. Some New Types

Last time we looked at the integer, character, varchar, and some other types. In this lab, I would like to introduce you to some new types:

NUMERIC	**: stores fractional (floating point) values without rounding.**
BIGINT	**: stores a 64-bit signed integer.**
SERIAL	**: stores an integer value which has a default value that automatically increments with each new record.**
TEXT	**: stores a block of text of arbitrary length.**
BYTEA	**: stores binary data such as a picture (Byte Array).**
BLOB	**: similar to BYTEA, but part of the SQL Standard (Blob stands for "Binary Large OBject".**
BOOLEAN	**: stores a single true or false value**
POINT	**: stores an (x,y) pair**
BOX	**: stores a pair of points (representing corners of a box)**

In your lab2.sql file, add SQL code to perform the following operations:

Create a table named "rants". Your table should have the fields "username", "rant", "post_id" (which should be a SERIAL) and "time_posted" (which should be a timestamp). Use appropriate types selected from those in the list above and/or Lab1.

Create another table named "accounts". It should have three fields "username", "password", and "image" (which should be a BYTEA).

Create a third table named "comments" which has the fields "username", "post_id", "comment_id", and "comment_text".

Step 3. Dropping Tables

The best way to drop a table is to use the DROP TABLE IF EXISTS command:

```
DROP TABLE IF EXISTS <tableName>;
```

At the top of your lab2.sql file add SQL code to drop all three of your new tables if they already exist. Save your work and test, by running the .sql file with psql, that all three tables can be dropped and recreated.

Step 4. Calling SQL functions

Like most modern programming languages, SQL allows us to create and use functions. We will worry about creating our own functions (called “stored procedures”) later in the course. For today, we are just going to use some built-in functions that SQL already provides for us.

In particular, we are going to use the "md5" function. This function takes a string of text and uses the MD5 cryptographic hash function to create a digest. This is useful, because we don't really want to store password in plain text. Even in the database, they would be vulnerable if our system is compromised.

To create an account in our rant database, we will use the insert command we learned in Lab 1. To insert the username "frank" with encrypted password "dumbPassword", you would type

```
INSERT INTO accounts (username, password) VALUES ('frank', md5('dumbPassword') );
```

Add this command to the bottom of your lab2.sql file now.

Then add accounts for "tom", "harry", and "celeste" with encrypted passwords "fish1", "fish2", and "fish3" respectively.

Step 5. Constraints

Users are notoriously good at doing stupid things that violate the security of their accounts. In particular, they often forget to set a password or set a blank password. Let's add a constraint to the database that prevents the password field from ever being set to NULL.

The easy way to do this is to change the CREATE TABLE command for the account table.

Change the line:

```
password        VARCHAR(8),
```

to read:

```
password        VARCHAR(8)      NOT NULL,
```

(Of course, if you implemented the password field differently, that's fine. Just add NOT NULL to the end before the comma). To test, use the interactive mode of psql to try to insert a tuple with only a username.

Step 6. Glitter

Populate your rant database with interesting and useful content using "INSERT INTO" statements at the bottom of your file. The more comments you have and the more interesting they are, the more points you will earn for glitter. Don't forget that in addition to the rants themselves, users can add comments to a rant.

Submiting:

As usual, save your work and upload your lab2.psql file to

http://narnia.homeunix.com/~robert/submit/