Homework #4
due Thursday, October 28, 2010 at 2 pm

Database Systems, CSCI-4380-01

Each student must work on this homework alone.

1 Homework Description

In this homework, you will be querying the database from Homework #2 and #3, i.e. Kip's political campaign. The schema is the same one from Homework #3. Recall that the database allows users to specify their interest in issues, create and participate in polls, events and pose questions to the candidate. Write the following queries using SQL. The bonus queries are worth 10% more than the regular queries. Answer only one of a query or its bonus version (Qi or Qi.1).

Write the following queries or expressions using SQL:

Q1. Find the most popular polls created within the last three months (where the popularity is determined by the total number of votes the poll got). Return the top 10 polls ordered by the total number of votes.

Hint. You can use the LIMIT statement after order by to return the top 10. Example: `select * from users limit 10 ;`

Q1.1. Execute the same query as in Q1 but order the results by the following formula and return the top 10 polls by this formula as well.

$$0.5 * \frac{\text{total \# of votes for this poll}}{\text{max \# votes any poll top 3 months}} + 0.5 * (1 - \frac{\text{number of weeks the poll exists}}{12})$$

Q2. Find users who are only friends with people who have the same status as themselves (you can use friendsSym relation if you wish).

Q3. Flag users with **more than or equal to 2** questions within any month by changing their status to 'possibleAbuse'. **Hint.** Read up on date functions to help you with this: www.postgresql.org/docs/8.4/interactive/functions-datetime.html

Q4. Create a new table called issueQuestions with the following attributes: issueName (string), questionId (int) and priority (int).

Insert into table issueQuestions the top 20 priority questions and issues. You can find the top priority questions as follows: For each question, find all the people who voted

on that question and the issues these people are interested in. This associates an issue with a question. Now count the number of votes for each issue and question, and make this the priority of this question for this issue. Insert the top 20 questions and issues based on this priority into the new table.

Q4.1 Instead of top 20 priority question issue pairs, find the top 3 issues for each question based on the priority computed above. Insert these tuples into your table instead of Q4.

Q5. Using issuesQuestions from Q4 (not Q4.1) find the issues users most care about (i.e. issues with the highest total priority).

Q6. Find the users who are the most likely donors. The users should have all the following properties:

- they have donated more than $100,000 total and at least 25 different times,
- they have not made a donation within the last month (**i.e. last thirty days**), and
- they have specified at least one issue that they care about.

Return the id of all the users who satisfy these conditions. Order the results by user id.

Q7. Create a trigger for the following: Whenever a user has an event that gets more than 5 RSVPs, update the status of the creator of this event to 'campaigning'.

Q8. Bonus For each user, find the set of all people who live in the same zip code and have some commonality to them (at least two out of the four conditions below hold):

1. they care about at least one issue in common,
2. they rsvp at least one event in common,
3. they vote on at least one poll in common,
4. they vote on at least one question in common.

Return the id of all matching pairs of users, order by the user ids.

2 Deliverables

Turn in a single text file (.sql) containing all your queries and expressions. It must be possible to execute the whole file using the

```
\i filename
```

command. To achieve this, make sure all queries execute and end with ;. If you have a query that does not run, you will loose all points for that query. It is better for you to turn in a query that is not fully correct than one that does not run. Also, note that you can write commands in a .sql file by preceding it with `--` . So, comment each query by preceding it with a line that describes the query you are answering. You should also use the psql command

```
\echo 'text'
```

to identify your name at the top and then the id of each query before the SQL for it. This will allow your TA to quickly run and test your homework. A template for answers is provided with this homework. You must use this template and fill in your personal inforamtion and the SQL queries.

Now, given this homework requires you to test insertion, you must create a local copy of the database. You can create one yourself or you can use the one that will be provided to you.

To test your queries, connect to the postgresql server at CS. First ssh to remote.cs.rpi.edu.

```
ssh remote.cs.rpi.edu -l username
```

using your CS username and start postgresql using the username and password mailed to you and the database where you stored the data

```
psql -h csc4380.cs.rpi.edu databaseName -U csc4380_username
```

again using your CS username. Note that each person has three databases: username, username_dev, username_test. You can use either one of these databases.

You can get help on how to use postgresql using the online documentation:

```
http://www.postgresql.org/docs/8.2/interactive/index.html
```