Database Systems, CSCI 4380-01
Homework # 6
Due Thursday April 7, 2011 at 2 pm

1 Introduction

You are given attached schema which is the same one used for Homework #4 and #5. Recall that, this database allows people to enter information about people, places and businesses. It is fairly free form. However, we store each version of information for people using some sort of versioning. Each table locations, businesses and people have an additional version table (name ends with V) which stores the past versions of each location, business and people as well as the most current version. If information is deleted from the main table, it is also deleted from the version table.

We also allow users to state relationships between people, places and businesses. Each relationship can only be stated once, so we store who first established this relationship. Finally, people are able to review these records, post comments on reviews and comments that follow up other comments in a thread. For each comment, we store the parent review it links to. Users can also friend others, vote on comments up or down (but not vote on reviews, which is a limitation of the system).

2 Homework Statement

In this homework, your objective is to write a pl/pgsql code to solve a problem using both SQL and the procedural language. The main challenge of doing so is to use each component appropriately: push most complex processing to queries and use procedures to combine them together.

Your objective is to compute for each user a status. Each status has a name and a level. So, if a user qualifies for more than one of the statuses described below, the one with the highest level is given to them. The idea is that the users must feel that they are valued in the system. So, you will assign them these special titles to achieve that.

To accomplish this, first you have to create two new tables:

```
CREATE TABLE statuses (
     id      INT PRIMARY KEY
     , level INT
     , name  VARCHAR(100)
) ;
CREATE TABLE userStatuses (
     userId            INT PRIMARY KEY
     , statusId        INT
     , FOREIGN KEY (userId) REFERENCES users(id)
     , FOREIGN KEY (statusId) REFERENCES statuses(id)
) ;
```

Here are the statuses users can achieve:

id	status	level
1	bellwether	15
2	well-rounded	14
3	best-link	13
4	best-comment	12
5	inciteful-comment	11
6	comboLinker	10
7	comboCommenter	9
8	inciteful-business	7
9	inciteful-location	6
10	inciteful-people	8
11	one-year-club	4
12	two-year-club	5
13	white-hat	3
14	new-user	2
15	alienator	1

Assume that both tables are created and userStatuses is loaded with the given values. The given schema in "schema_hw6additions" already accomplishes this. Write a procedure that computes the statuses for the users. The meaning of each status is given below.

2.1 User Statuses

Below are the descriptions of what it takes to achieve a specific status.

- **Alienator:** A person who never posted anything on the site site and has no friends.
- **White Hat:** A person who never posted anything on the site (comment, review, business, people or location) but has friends.
- **New user:** A person with less than one year of membership and posted something on the site.
- **One year club:** A person that has been a member more than a year and less than two years and and posted something on the site.
- **Two year club:** A person that has been a member more than two years and less than three years and posted something on the site.
- **Inciteful business:** A user who entered a business (with version 0) with the highest number of updates.
- **Inciteful location:** A user who entered a location (with version 0) with the highest number of updates.
- **Inciteful people:** A user who entered a person (with version 0) with the highest number of updates.
- **Inciteful Comment:** A user with the highest standard deviation value of votes for their comments (only consider users with more than average number of comments).
- **ComboLinker:** A user that has the highest number of total versions for businesses, locations and people that they first entered (with version 0).
- **ComboCommenter:** A user that has the highest total number of comments for their reviews.
- **Best link:** A user that has the highest number of reviews for any one of business/person/location that they first entered (with version 0).
- **Best comment:** A user that has the highest total votes for his/her comment.
- **Bellwether:** A user with the most influence in one area (business, people, location) in terms of the total number of reviews and comments for one item (business, person, location) that they first entered (with version 0). In this number, exclude the comments and reviews by the user and his/her friends.

- **Well-rounded:** A user with the highest total influence in all areas combined (busines, people, location) in terms of the total number of reviews and comments for all the items (business, person, location) that they first entered (with version 0). In this number, exclude the comments and reviews by the user and his/her friends.

3 Deliverables

The grading for this homework is going to be as follows (out of 100 points total):

- Implement statuses 11-15, get 80 points total.
- Implement statuses 6-15, get 100 points total.
- Implement statuses 3-15, get 120 points total.
- Implement all statuses, get 140 points total.

To test the functionality of this homework, you will need to create a copy of the database in one of your own personal database (we will post the data on RPILMS Bulletin Board). Recall, each student in the class has 4 different databases. We will test your homework in our own database assuming the database is created with the additional tables listed above, the statuses table is populated with the appropriate data and userStatuses is empty. Your procedure when executed should populate the userStatuses with the appropriate data.

Your procedure can create any additional/auxilary tables it needs as long as it drops them when it completes. Make sure also that you do not change any data in the database except for the userStatuses table.

Turn in a single text file (.sql) containing the code for your procedure only. It must be possible to create the procedure using the

```
\i filename
```

command in psql. Use the attached template for your code. Do not change the name of the function provided in the answer template. You can run the created function using:

```
select hw6proc() ;
```

in psql. To test your code, 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 csc4380.

```
psql -h csc4380.cs.rpi.edu csc4380_wud6 -U csc4380_username
```

again using your CS username. You can get help on how to use postgresql using the online documentation:

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

You can find documentation for pl/pgsql at:

```
http://www.postgresql.org/docs/8.4/interactive/plpgsql.html
```