Mini-Proj 1 Spec

CMPUT291 - Winter 2017
Mini Project I

(group project)

***Due**: March 10 at 5pm*

Clarifications:

You are responsible for monitoring the course news and discussion forums in moodle and this section of the project specification for more details and clarifications. No clarification will be posted after *5pm on March 9*.

- **Feb 21**. Long Ma noticed that our Create Table statements were not consistent with the schema given here. That is fixed.
- **March 3**. The new due date for Mini-project 1 is March 12 at 5pm.
- **March 7**. A keyword search will return all matching tweets (including those posted by users who are not being followed). The user can search with more than one keyword, in which case every tweet that matches at least one of the keywords is returned.
- **March 7**. In composing a tweet, you need to assign a unique id to the new tweet.
- **March 9**. Here is a marking rubric. Project reports are due on Monday March 13th at 9:30am, and they should be dropped in the course drop box.

Introduction

The goal of this assignment is twofolds: (1) to teach the use of SQL in a host programming language, and (2) to demonstrate some of the functionalities that result from combining SQL with a host programming language.
Your job in this project is to build a system that keeps the enterprise data in a database and to provide services to users. You will be storing data in Oracle and will be writing code in Python/cx_oracle (or similarly Java/JDBC, C/ProC, etc.) to access it. Your code will implement a simple command line interface. You are free to implement a GUI interface instead but there will be no support nor bonus for doing that. You are also free to write your code in Python, Java, C, C++, Perl or any other language that is suited for the task. If you decide to use any language other than Python or Java, discuss it with the instructor first.

Your project will be evaluated on the basis of 84% of the mark for implementing the functionalities listed in this specification; this component will be assessed in a demo session. Another 12% of the mark will be assigned for both the documentation and the quality of your source code. 4% of the mark is assigned for the quality of your group coordination and the project break-down between partners.

Group work policy

You will be doing this project with *one or two other partners* from the 291 class. Register your group at the group registration page, as soon as possible. It is assumed that both group members contribute somewhat equally to the project, hence they would receive the same mark. In case of difficulties within a group and when a partner is not lifting his/her weight, make sure to document all your contributions. If there is a break-up, each group member will get credit only for his/her portion of the work completed (losing the mark for any work either not completed or completed by the partners). For the same reason, a break-up should be your last resort.

Database Specification

You are given the following relational schema.

- users(usr,pwd, name,email,city,timezone)
- follows(flwer,flwee,start_date)
- tweets(tid,writer,tdate,text,replyto)
- hashtags(term)
- mentions(tid,term)
- retweets(usr,tid,rdate)
- lists(lname,owner)
- includes(lname,member)

Tables are derived from the Assignment 1 spec and are identical to those in Assignment 2 except the field *pwd* which is added to *users*, and the change in the key of table *tweets* and in the definition of tables that reference the table tweets. The SQL commands to create the tables of the system are given here. Use the given schema in your project and **do not change any table/column names** as we will be testing your project with the given schema.

Login Screen

The first screen should provide options for both registered and unregistered users. There must be also an option to exit the program. Registered users should be able to login using a valid user id and password, respectively referred to as *usr* and *pwd* in table users. After a registered user signs in, the system should list all tweets or retweets from users who are being followed; the tweets should be ordered based on date from the latest to the oldest. If there are more than 5 such tweets, only 5 would be shown and the user would be given an option to see more but again 5 at a time. The user should be able to select a tweet and see some statistics about the tweet including the number of retweets and the number of replies. Also the user should be able to compose a reply to it (see the section on composing a tweet), or retweet it (i.e. repost it to all people who follow the user).

Unregistered users should be able to sign up by providing a name, email, city, timezone and password. The user id (i.e. the field *usr* in table users) should be provided by the system and be unique. After a successful login or signup, users should be able to perform the subsequent operations (possibly chosen from a menu) as discussed next.

System Functionalities

Users should be able to perform all of the following tasks:

1. Search for tweets. The user should be able to enter one or more keywords and the system should retrieve every tweet that *match* at least one of the keywords. A tweet matches a keyword if (1) the keyword has the prefix # and it is mentioned by the tweet as a hashtag, or (2) the keyword doesn't have the prefix # and it appears in the tweet text. (The symbol # is not part of any keyword and only indicates that the keyword that follows is expected to appear as a hashtag. Also tweets can have hashtags which do not explicitly appear in the tweet text.) The tweets should be ordered based on date from the latest to the oldest. If there are more than 5 matching tweets, only 5 would be shown and the user would be given an option to see more but again 5 at a time. The user should be able to select a tweet and see some statistics about the tweet including the number of retweets and the number of replies. Also the user should be able to compose a reply to a tweet (see the section on composing a tweet), or retweet it (i.e. repost it to all people who follow the user).
2. Search for users. The user should be able to enter a keyword and the system should retrieve all users whose names or cities contain the keyword. The result would be sorted as follows: first, all users whose name match the keyword would be listed and these users would be sorted in an ascending order of name length. This would be followed by the list of users whose city but not name match the keyword and this result would be sorted in an ascending order of city length. If there are more than 5 matching users, only 5 would be shown and the user would be given an option to see more but again 5 at a time. The user should be able to select a user and see more information about him/her including the number of tweets, the number of users being followed, the number of followers and up to 3 most recent tweets. The user should be given the option to follow the user or see more tweets.
3. Compose a tweet. The user should be able to compose a tweet. A tweet can have hashtags which are marked with a # before each hashtag. Information about hashtags must be stored in tables *mentions* and if needed in *hashtags*.
4. List followers. The user should be able to list all users who follow him/her. From the list, the user should be able to select a follower and see more information about the follower including the number of tweets, the number of users being followed, the number of followers and up to 3 most recent tweets. The user should be given the option to follow the selected user or see more tweets.
5. *Manage lists. The user should be able (a) to get a listing of his lists, (b) to see the lists that he is on, (c) to create a list, and (d) to add or delete members to hist lists.
6. Logout. There must be an option to log out of the system.

Task 5 is required for groups of size 3 only.

Testing

At development time you will be testing your programs with your own data sets (make sure that it conforms to the project specification). At demo time we will be creating the database using these SQL statements and will be populating it with our own test data set. Your application will be tested under a demo account (and not under your account).
The demo will be run using the source code submitted and nothing else. Therefore, it is essential to include every file that is needed to compile and run your code including all source code and any makefile or script that you may use to compile or run your code. You will neither be able to change your code, nor use any file other than those submitted. This policy can be altered only in exceptional cases at the instructor's discretion and for a hefty penalty. The code will be executed under a TA account. Do not hard-code username, password or table prefixes (such as username or group name) in your code. As a test drill, you should be able to set up your application under someone else's account (in case of testing, this would be under a TA account) within 3 minutes at most.

Every group will book a time slot convenient to all group members to demo their projects. **At demo time, all group members must be present.** The TA will be using a script to both create and populate the tables. The TA will be asking you to perform various tasks and show how your application is handling each task. A mark will be assigned to your demo immediately after the testing.

Instructions for Submissions

Your submission includes (1) the application source code and (2) the design document. The source code is submitted as follows:

- Create a single gzipped tar file with all your source code and additional files you may need for your demo. Name the file *prjcode.tgz*.
- Submit your project tarfile in the project submission site.
- All partners in a group must submit their own copies (even though the copies may be identical)

Your design document must be type-written and submitted in hardcopy at the designated drop boxes located on the first floor of CSC building, across from the room 1-45, before the due date. Your design document cannot exceed 4 pages.

The design document should include (a) a general overview of your system with a small user guide, (b) a detailed design of your software with a focus on the components required to deliver the major functions of your application, (c) your testing strategy, and (d) your group work break-down strategy. The general overview of the system gives a high level introduction and may include a diagram showing the flow of data between different components; this can be useful for both users and developers of your application. The detailed design of your software should describe the responsibility and interface of each primary class (not secondary utility classes) and the structure and relationships among them. Depending on the programming language being used, you may have methods or functions instead of classes. The testing strategy discusses your general strategy for testing, with the scenarios being tested, the coverage of your test cases and (if applicable) some statistics on the number of bugs found and the nature of those bugs. The group work strategy must list the break-down of the work items among partners, both the time spent (an estimate) and the progress made by each partner, and your method of coordination to keep the project on track. The design document should also include a documentation of any decision you have made which is not in the project specification or any coding you have done beyond or different from what is required.

Your design document must not include the source code. However your source code (which is submitted electronically) would be inspected for source code quality (whether the code is easy to read and if data processing is all done in SQL) and self-documentation (whether the code is properly commented).

Last modified: Friday, 10 March 2017, 1:04 PM