Carnegie Mellon University

15-415 Database Applications

Spring 2012, Faloutsos

Assignment 7: Database Application

Part I Due: 4/3, 1:30 pm

Part I Due: 4/12, 1:30 pm

Reminders

- Weight: **30%** of the homework grade.
- The points of this homework add up to **100**.
- Lead TA: Bin Fu (binf@andrew.cmu.edu).
- Please **type** your answers. Only drawings can be hand-drawn, as long as they are neat and legible.
- This assignment has two phases:

  Phase I is due 4/3 as hard copy in class.
  Phase II is due 4/12 as *both* hard copy in class and electronic files via email.

- Extra office hours: Bin will hold extra office hours for this assignment (at GHC 8129).
  3/28 (Wed) 12pm – 3pm
  4/2 (Mon) 12pm – 3pm
  4/4 (Wed) 12pm – 3pm
  4/5 (Thu) 3pm – 6pm
  4/9 (Mon) 12pm – 3pm
  4/10 (Tue) 3pm – 6pm
- Please refer to additional slide of this homework at http://www.cs.cmu.edu/~christos/courses/dbms.S12/hws/HW7/additional.ppt
- Rough time-estimates: **15~25 hours**. Start early!

Introduction

You are to create a micro-blog web site, similar to Twitter (http://www.twitter.com). In the process, you will learn the main challenges involved in providing such a service, and appreciate the importance of database technology in today's data-driven world.

This assignment is divided into two phases, as we describe in the Roussopoulos and Yeh methodology (see later for exact pointers). In the first phase, you will plan your implementation strategy at a high level, and submit documentation that describes the major design decisions you have made. The first phase permits the TAs to evaluate your progress at an early stage and provide feedback.

The second phase is the implementation of the system. For both phases you are expected to give extensive documentation for the system. The target audience of the document is fictitious person who will have to maintain the system afterward.

Requirements

Data Requirements

The system allows users to publish short messages ("tweets") and see other's tweets. Each tweet may contain one or multiple hashtags.

1. **User data:** For every user, we want to record the login name (e.g. "john123"), the password, the email address, and the role ("regular user" or "system administrator"). Login names should be unique. The password cannot be empty.
2. **Tweet data:** For each tweet, the system should store its content, its owner (the user who published the tweet), and time when the tweet is published.
3. **HashTag data:** Hashtags are words or phrases prefixed with the symbol #, which is usually used to identify groups or topics. For example, "Six months on, what has #OccupyWallStreet achieved and what needs to change?" Then, user can search for a hashtag to get all tweets having that hashtag.
4. **"Follow" data:** For each user, we want to record a list of users he/she follows.

Functionality Requirements

1. **Registration**: Before a user can start using your system, he/she needs to register by providing login name, email address, and choosing a password. If a login name chosen by a user during registration already exists, the system should give an error message and prompt for a different login. You can just store the password as clear text, and you need

not verify the validity of the email address. ***Note*: a sample registration is already provided for your reference (www/register.jsp)**. Please make any modification to meet the above requirement.

2. **Login/Logout**: A user should be able to login with the login/password he/she provided during registration. For security, no one should be able to view any page of your website without logging in. Also, once logged in, the user should not need to log in to view other pages. A login session is valid until the user explicitly logs out, or the user remains idle for 30 minutes.

   Hint: In this assignment we use Tomcat, which provides easy session and cookie management primitives using cookies and server-side persistence. The default timeout for the sessions for your tomcat JSP server is 30 minutes, so you need not modify that. Please refer to Chapter 6, Section 6 and 7 of Thinking in Enterprise Java (see below) for detailed example of using session and cookies in JSP.

3. **Profile Page:** When the user logs in, he/she should see his/her "profile information" including (1) all the information about this user (login name, email address, etc.), (2) the list of tweets from the users he/she follows, in reverse chronological order (that is, from latest to earliest), (3) the list of users that he/she followed (login name) (4) the list of users (login name) who followed him/her.

4. **Add Tweet:** A user should be able to add a new tweet. Each tweet must be at most 140 characters, and may include one or more hashtags.

5. **Browse Tweet:** Given a login name, a user can browse all the tweets from that user, in reverse chronological order. If the provided login name does not exist, return a "Sorry, that page doesn't exist" page.

6. **"Follow" Management:** A user should be able to follow another user if it has not happened already. A user can cancel following a user he/she has followed before.

7. **Hashtag Search:** A user should be able to search tweets by hashtags. Given a hashtag, the system should return all the tweets containing that hashtag, in reverse chronological order.

8. **Reporting:** The system should contain a built in administrator "**admin**". Make sure you submit your password in your hard-copy for the second phase. The administrator should

be able to check summary statistics of user activity in a report page. The page should contain:

(a) the total number of registered users in the system;
(b) the total number of tweets in the system;
(c) *Heaviest "twitters"*: the top-10 users (login name) who have published the largest number of tweets, and the numbers of tweets they published. Break tie using the login name; exclude users that do not submit any tweets (note that in some cases less than 10 users might qualify).
(d) *Celebrities*: the top-10 users who is followed by most users, and the number of users following them. Break tie using the login name; exclude users that are not followed by any user.
(e) *Trending hashtags*: the top-10 most-used hashtags in the last hour, and the number of times they are used. Break ties using the alphabetical order of hashtags; exclude the hashtags that are not used in the last hour.
(f) *Busy days*: the top-10 days in which the highest number of tweets were published, and the numbers of tweets that were published in those days. Break ties using the timestamp (in ascending time order); exclude days which have no tweeting activity.

Setup

A web server (Tomcat 6) with JSP support has already been set up for you. To access it, you should log into your account on **newcastle.db.cs.cmu.edu** (the one you used for HW2 and HW 6). Please clean up and back up your directory and type the following commands:

- create the web directory **www**:

  ```
  $ ../binf415/setup_db.sh
  $ ../binf415/setup_web.sh
  ```

  If you are prompted about replacing any files, choose the "[A]ll" option.

- create the database **hw7**, and add user **www** after starting the pgsql server:

  ```
  $ createdb hw7
  $ createuser www -P
  Enter password for new role:
  Enter it again:
  Shall the new role be a superuser? (y/n) y
  ```

Note: this user is only for setting up the website. It is not a user of the website.

- add table **users** to the database:

  $ psql hw7

  hw7=# create table users(login_id varchar(32), passwd varchar(64), email varchar(128));

  Note: this table is required to run the demo code only. For your submission, you should change the table as to the data and functionality requirements.

- exit the sql environment (\q). Run **echo $PGPORT**. You should get a port number.
- change the port number for localhost in line 35 of **www/register.jsp** to the one you just got.
- change the password in line 35 of **www/register.jsp** to the one you just entered.
- access http://newcastle.db.cs.cmu.edu:8080/<andrew-id>415/register.jsp. It should be similar to the sample registration page at http://newcastle.db.cs.cmu.edu:8080/testuser415/register.jsp.

After setup, every time you login, you can start the postgresql server by running

pg_ctl -w start

Every time when you logout, you must stop the postgresql server by running

pg_ctl stop

Recommended Code Organization

In this project, you will write HTML, JSP, and/or Java code. The www directory contains all the code accessible online. Here is the recommendation for your code organization (# for comments). Feel free to deviate from it, but following it will make grading easier.

```
-~                  # home directory
  - src             # source code directory, including Java, JSP and HTML
  - www             # where you deploy your web application

    - WEB-INF
      - web.xml     # your web configuration file, see Tomcat doc
      - classes     # you should put all compiled Java classes (.class files) here
```

```
  - lib          # any external library, e.g. postgresql-8.3-603.jdbc4.jar
- META-INF       # Tomcat specific context configuration directory
                 # no need to modify
```

Note: In the handout sample implementation of registration, all processing are directly included in **www/register.jsp**. While this provides a simple solution, it may create complex JSP pages in a large project. A more elegant way is to use the MVC pattern – separating functionality of model, view and control. There are many such tools available, e.g. **struts**.

You are welcome to use struts if you would like, but any working solution will get full points (with, or without struts).

Phases

The two phases of this assignment follow the work-processes methodology from Adaptable Methodology for Database Design by Roussopoulos and Yeh [IEEE Computer, May 1984] (http://www.cs.cmu.edu/~christos/courses/dbms.S12/slides/CMU_ONLY/Roussopoulos-Yeh.pdf , the lecture slides are also available at http://www.cs.cmu.edu/~christos/courses/dbms.S12/slides/20methodology.pdf).

- [**Phase I**] Environment and Requirement Analysis, System Analysis and Specication, Conceptual Modeling, Task Emulation.
- [**Phase II**] Implementation and Testing.

Point Distribution and Deliverables

Phase I: report in hard copy, due 4/3 1:30pm in class [35 points]

The Phase I report should contain the following:

1. [**1 pt**] The top-level information flow diagram, (very important - also, don't forget the system boundary).
2. [**1 pt**] The list of documents.
3. [**3 pts**] The document forms, including the assumptions and design decisions you made.
4. [**5 pts**] The E-R model (omit attributes, to avoid cluttering the diagram). Make sure you specify cardinalities of the relationships following the format in hw1 solution.
5. [**2 pts**] List of the attributes for each entity and relationship.

6. [**3 pts**] Explanations of the any non-obvious entities and relationships.
7. [**5 pts**] The schema in the relational model. Make sure the schema is in a good form (BCNF or 3NF).
8. [**2 pts**] Explanations (e.g., primary keys, additional functional dependencies, explanation why a table is not in BCNF etc.)
9. [**2 pts**] The SQL DDL statements to create the above relational schema.
10. [**10 pts**] The SQL DML statements for all the tasks.
11. [**1 pt**] Run the registration task given in the demo code. Report the output generated by the webpage when you register your andrew-id.

Phase II: report in hard copy, due 4/12 1:30pm in class [15 points]

The Phase II report should contain the documentation produced in this phase:

1. [**0 pt**] system administrator password.
2. [**5 pts**] a source program listing.
3. [**5 pts**] a user's manual for the system.
4. [**5 pts**] your testing efforts: erroneous cases that your system can detect and handle reasonably (e.g., non-member trying to access the system, user trying to publish a tweet with more than 140 characters, etc.).

Phase II: report in electronic files, due 4/12 via 1:30pm by email [50 points]

You should also zip all your source code into a file **hw7-*your_andrew_id*.zip** and send to binf@cs.cmu.edu with the subject "15415-hw7-*your_andrew_id*". The weights of the tasks are as shown in Table 1. Notice that about two thirds of the points are for basic implementation that supports the desired functionality, and the rest are for error checking (as described above).

Note: GUI is not the emphasis in this assignment.

Task	Basic implementation	Testing / Error checking
1. Registration	0	2
2. Login/Logout	6	2
3. Profile Page	4	2
4. Add Tweet	4	2
5. Browse Tweet	4	2
6. Follow Management	4	2
7. Search	4	2
8. Report	8	2
Total	34	16

Optional Documentation and Resources

Thinking in Enterprise Java (http://www.mindviewinc.com/downloads/TIEJv1.1.zip). Chapter 4~6 covers how to connect to databases, servlets and JSP. Chapter 3 on RMI is optional.

Thinking in Java (http://www.mindviewinc.com/downloads/TIJ-3rd-edition4.0.zip): a good introductory book on Java programming;

Thinking in Patterns (http://www.mindviewinc.com/downloads/TIPatterns-0.9.zip): Java design principles and methods.

Tomcat 6.0 manual is available at http://tomcat.apache.org/tomcat-6.0-doc/index.html.

struts is a framework for building enterprise web application. It uses MVC pattern (Model-View-Controller). The API is available at http://struts.apache.org and you could find some nice tutorial at http://struts.apache.org/2.x/docs/tutorials.html.