Carnegie Mellon University
Department of Computer Science
15-415/615 - Database Applications
C. Faloutsos & A. Pavlo, Spring 2015
Prepared by Jiaxi Xiong

DUE DATES: Ph1: 11/11, Ph2: 11/30, both at 3:00pm

Homework 7

IMPORTANT - what to hand in: For each of the two phases, please deliver **all** the elements below (*penalties* for omissions).

- Phase 1: Due at **11/11, 3:00pm**:
  - *hard copy*: All your documentation for Phase 1.
  - *Blackboard submission*: a file `[andrew_id_hw7_phase1].pdf` with your documentation.
- Phase 2: Due at **11/30, 3:00pm**
  - *Website*: A working web site - include its URL in both hard and e-copies below.
  - *hard copy*: the code you wrote for Phase 2.
  - *Blackboard submission*: a zip file named`[andrew_id_hw7_phase2].zip`, with all the necessary files for us to re-create your web site.

Important for Phase 2: The zip-file `http://www.cs.cmu.edu/~christos/courses/dbms.F15/hws/HW7/hw7.zip`. contains everything you need for Phase 2.

Reminders:

- *Plagiarism:* Homework may be discussed with other students, but all homework is to be completed **individually**.
- *Late Homeworks*: The usual rules: (a) hard copy to Mrs. Marilyn Walgora, and (b) email to all TAs, with the subject line `15-415 Homework Submission (HW 7) Phase`[phase#], and the count of slip-days used (and remaining).

For your information:

- Graded out of **100** points;
- **2** questions total
- 5-10 hours for phase 1; 10-20 hours for phase 2.

Question	Points	Score
Deliverables - ph1	35	
Deliverables - ph2	65	
Total:	100	

Homework 7 continues. . .

1 Introduction

The goal is to design and implement CMUYak, a very simple application that lets you make a post, search post with different criteria, like other posts and see some statistics. Your work is divided into two phases, as we discuss in the lecture foils, following the `http://www.cs.cmu.edu/~christos/courses/dbms.F15/slides/CMU_ONLY/Roussopoulos-Yeh.pdf`
The first phase consists of the *requirement analysis*, *system analysis*, *conceptual modeling* and *task emulation*. The second phase consists of *implementation and testing*: Using the API we provide you in the *hw7.zip* file above, you will have to implement your database design, as well as the functionality that the API supports, and deploy your solution on a webserver.

2 Requirements

2.1 Data requirements

- **User**: For each user, the system needs to track the unique username, (between 2 and 50 characters), the password(in 32 bit hashed format), the posts (see below) he/she makes and votes for.
- **Posts**: The system needs to store: the title of the post(between 2 and 20 characters), the body of the post(between 2 and 42 characters), the location where the post is made(represented in x and y integer coordinates), the hashtags used in the post, the username of the author, the timestamp it was created and users who voted for this post.
- **Hashtags**: The system needs to store: the name of the tag, the posts that use this tag.

Example: The user "Smith" creates a new post with the title "Savage DB Research" and the content as *"Winter is coming! #GOT"* at location $(0, 0)$. This should be stored in the table that contains information about posts. The application will also parse and extract the hashtag "#GOT" contained in this post and stored separately in the database without the number sign prefix (i.e., "#"). The database should also contain a reference that identifies that the particular hashtag was used in the post.

2.2 Functionality requirements

In CMUYak we have users who make posts, vote for posts made by other users, read posts, etc, as described below. You have to implement the following Tasks:

T.1 **Reset database**: Keep the tables, but delete all their records.

T.2 **Create user account (Register)**: We need the username of the user and the password. Prompt for a new username, if the proposed one is taken.

T.3 **Login**: Obvious - your system would ask for a username and password, and authenticate the user or deny further access.

T.4 **Add Post with possible hashtags**: Once authenticated, a user should be able to add a post. When a new post is added, the application should find any hashtags in the post content (not the title) and add them into a separate hashtag table. Each hashtag should only appear once in this table. The application will then maintain a separate cross-reference table that keeps track of all the hashtags that appear in each post.

T.5 **Timeline**: After log-in, the main page should display the timeline. That is, your system should show all the posts of all users in chronological order (newest first).

T.6 **List all post for a given user**: For a given user, list posts made by that user in descending order of the post timestamp.

T.7 **Search for posts**: For an input keyword, search for the posts that contain this keyword in title or content.

T.8 **Search for posts within range**: For a input string as keyword, a target point and the maximum possible Euclidean distance from the target point, search for the posts that were made within the range, and contain the keyword in title or content.

T.9 **Search for a tag**: Given a hashtag, search for the posts that include this hashtag. (Attention: our server will strip the # from the input hashtag in the search page.)

T.10 **Delete a post**: A user should be allowed to delete a post, provided he or she is the owner of the post. (Please also consider, what else should be done when you delete a post?)

T.11 **Vote for a post**: Your system should allow a user to vote for another post made by different users. A user should not be allowed to vote for their own posts. One should be allowed to vote for a given post only once.

T.12 **Unvote for a post**: Your system should allow a user to unvote for a post that this user previously voted for.

T.13 **List most popular posts**: User shoud be able to see the most popular posts over the last day/week/month (based on post's timestamp). Popularity of a post is based on the number of votes.

T.14 **Recommend posts based on votes**: For a given user 'U', your system should recommend posts to read. In short, we want to recommend the posts that are voted by other users whose tastes match the given user 'U'. You should not include the posts that the user has already voted for. Recommended posts should be sorted based on the 'Similarity Score'. (i.e the posts with the highest 'common votes' should be displayed first.)

T.15 **User statistics**: For a given user-name 'U', display the following counts, or print a warning if 'U' does not exist.

(a) The number of posts made by that user

(b) The number of likes made by that user

(c) The number of unique hashtags used by that user's posts
(For example, if a user has two posts, each of them contains"#CMU" in their content. We'd say that the number of unique hashtags used by this user is 1)

T.16 **Global statistics** The system should be able to list information about all users and posts, and specifically:

(a) **Most popular posts**: List of K posts with the most likes

(b) **Most active user**: List of K users with the most posts

(c) **Most popular tags**: List of K tags that appears the most often

(d) **Most popular tag pairs**: List of K tag pairs that appear together the most often

In all the above cases, treat K as a parameter. In case of a tie in K-th place, report only K users, favoring the alphabetically first. (i.e., if K=1, and 'bob' and 'alice' tie in first place, report only 'alice'.)

Homework 7 continues…

Phase 1 - Deliverables - ph1 . . . . [35pts]

No need to separate your answers

We follow the design methodology of Roussopoulos and Yeh, summarized in the lecture foils: `http://www.cs.cmu.edu/~christos/courses/dbms.F15/slides/19methodology.pdf`. Thus, we have the following deliverables:

- Phase 1: Environment and Requirement Analysis, System Analysis and Specification, Conceptual Modeling, and Task Emulation.
- Phase 2: Implementation and Testing.

Specifically, for Phase 1, the point distribution is as follows:

(a) [**2 points**] The top-level information flow diagram, along with the system boundary.

(b) [**1 point**] The list of documents.

(c) [**2 points**] The document forms, including the assumptions and design decisions you made.

(d) [**3 points**] The ER diagram. Make sure you specify the cardinalities of the relationships, as in HW1.

(e) [**2 points**] The relational schema.

(f) [**10 points**] SQL DDL statements that create the above schema - make sure you include all constraints (primary key, foreign key, etc)

(g) [**15 points**] SQL DML statements for tasks T.1-T.16.

Phase 2 - Deliverables - ph2 .... [65pts]

No need to separate your answers

The deliverables and point distribution of Phase 2 are as follows:

(a) [**0 points**] Download `http://www.cs.cmu.edu/~christos/courses/dbms.F15/hws/HW7/hw7.zip`.

(b) [**0 points**] List of changes, if **any**, to your Phase 1 schema.

(c) [**5 points**] Listing of your code, below.

(d) [**5 points**] Listing of your testing efforts - for each task, please write down which error cases you tested for (ie., non-existing user, illegal timestamp, etc)

(e) [**50 points**] The actual implementation: In `hw7.zip` above, we are providing you with a basic implementation of the web application, which includes calls to the database that you have designed but does not implement the required functionality. Your job will be to connect the provided code with the database that you have designed, in Phase 1 so that you have a working website that performs Tasks T.1-T.16.

Table 1 shows the breakdown of the points. For each tasks, half of the assigned points go to the basic implementation (i.e. having a working implementation of the task) and the second half goes to *testing* and *error checking*.

Task	**Points**
T.1 Reset database	1
T.2 Account creation	1
T.3 Login	1
T.4 Add posts with hashtags	5
T.5 List newest posts	3
T.6 List all posts for a user	2
T.7 Search posts containing a string	2
T.8 Search posts containing a string and with range	6
T.9 Search for a tag	2
T.10 Delete a post	3
T.11 Vote for a post	2
T.12 Unvote for a post	2
T.13 List hottest posts	2
T.14 Recommend posts	6
T.15 User statistics	6
T.16 Global statistics	6

Table 1: Point breakdown

(f) [**5 points**] *SQL injection*: In short, make sure you strip-off all the "escape" characters from your SQL strings. The reason is that, apart from the above functions, you want to prevent unauthorized access to your system. More specifically,

Question 2 continues. . .

you want to protect your SQL queries from an attack called *SQL injection* (see `http://en.wikipedia.org/wiki/SQL_injection`). For instance, code oblivious to SQL injection, may allow an intruder to log-in without a password. Thus, your code should sanitize the arguments to the SQL queries from all escape characters. *HINT*: check the PHP function `pg_escape_string()`.

Question 2 continues…

3 Setup and Hints for Phase 2

3.1 VERY IMPORTANT: Set up

As in the previous homeworks, you will use Postgres; the full documentation for Postgres is at `http://www.postgresql.org/docs/`.

For Phase 2, you need access to a webserver. You will use the webserver that is provided by the CMU Computer Club, which allows web content to be served directly from your Andrew s home directory in AFS without manual publishing. More details are in here: `http://www.club.cc.cmu.edu/doc/contribweb.php`.

To get started, please follow the following steps:

1. Log-in here `http://my.contrib.andrew.cmu.edu` using your Andrew ID, in order to set up your account for the Contributed server.
2. Read carefully the information provided by the Computer Club on how to set up your Postgres account and access the database on their server: `http://www.club.cc.cmu.edu/doc/contribweb/sql.php`.
3. Unzip `hw7.zip` and copy its contents on a folder called `cmuyak` inside the `www` directory on your andrew AFS account.
4. Edit `config.php` to have the appropriate parameters for your site (the web address and your Postgres database information). Also, make sure that your files are set to be both read and executed by anyone (`chmod +rx`).
5. Open your favorite web browser and go to `http://www.contrib.andrew.cmu.edu/~andrew_id/cmuyak` replacing `andrew_id` with your own Andrew ID. Make sure that you can see the Login screen of CMUYak.
6. You are ready to start implementing `functions.php`!

3.2 IMPORTANT: Customization

- `functions.php`: The provided `hw7.zip` includes a source file `functions.php` which contains the definitions of the API functions that you will have to implement. This is the main file that you need to edit.
- `config.php`: Moreover, you will have to edit `config.php` and fill in your own log-in information for the database, as well as your host information.

3.3 Strong hints

Running prototype See `http://www.contrib.andrew.cmu.edu/~jiaxix/cmuyak/`. for a prototype with all required functions.

Question 2 continues…

You are strongly encouraged to use this website in order to guide your own implementation, especially in cases when you are not sure if your design/implementation is working correctly.

Euclidean Distance See `https://en.wikipedia.org/wiki/Euclidean_distance` for definition of Euclidean Distance. We have a sample sql clause for this:
sqrt(power((x - 3),2) + power((y - 4),2))
This formula will get the distance from point(x,y) to point(3,4).

Some sample functional tests Additionally, we provide you with a set of basic functional tests, similar to the ones that we will be using for grading Phase 2. These tests will firstly **reset the database** (i.e. **USE WITH CARE**) and then evaluate the functionalities of the API that you have implemented. You are welcome to develop your own unit tests based on the provided ones.

To run the tests, just visit
`http://www.contrib.andrew.cmu.edu/~andrew_id/cmuyak/tests`
with your browser, and the tests will all be run sequentially, showing errors if any. Please note that by default these tests will fail because the functionalities aren't there yet, but it is strongly suggested that you run these tests often and along with your development, as a way to guide you through the development process.

Feel free to navigate the actual test files (i.e. the `tests.js` file) in the `tests` folder, and do your best to add more tests. You may find more information about the nice testing framework called Mocha at `http://mochajs.org/`, the BDD style assertions at `http://chaijs.com/`, and also the jQuery based Ajax call at `http://api.jquery.com/jquery.ajax/`.

Note that we will be using more than what's included in the current tests file to evaluate your code in the evaluation process. So please test for all possible edge cases.

"Sharpen Your Axe" Last but not the least, remember to choose good tools to facilitate your development process. Some recommendations are: a good and handy editor (e.g. vi/emacs or IDEs like PhpStorm), a code version control tool (e.g. git), a front-end testing tool (e.g. curl or Chrome's built-in inspector), and test oriented development process (e.g. the `tests` page that runs all functional tests). Good luck!

3.4 *Optional* documentation

For this phase, you will have to write some PHP, and in case you want to develop your own unit tests, some javascript and JSON object manipulation in PHP. Here we point you to some useful links:

1. PHP manual `http://www.php.net/`

Question 2 continues. . .

2. Arrays in PHP `http://www.php.net/manual/en/language.types.array.php`
3. PHP and Postgres `http://www.php.net/manual/en/book.pgsql.php`
4. Javascript tutorial `http://www.w3schools.com/js/DEFAULT.asp`
5. JSON in PHP `http://www.php.net/manual/en/ref.json.php`
6. Mocha.js testing framework `http://mochajs.org/`
7. Chai.js assertion library `http://chaijs.com/`
8. jQuery `http://api.jquery.com/`

End of Homework 7