HW7: Database Application

TAs: Alex Beutel; Vagelis Papalexakis
     Shen Wang; Ming Zhong
Overview

• Design & implement a simple web application
• CMU Fictitious light-weight Twitter → Flitter
• Today:
  • Application specs
  • Homework deliverables
  • Very brief intro to PHP
Data requirements

• **Users**
  - Username (4-50 characters)
  - Password
  - Can follow or be followed

• **Tweets**
  - Up to 140 characters
  - Have to record *when* they were posted
Functionality requirements

1. Create user account
2. Reset database
3. Login
4. Timeline
5. Post a tweet
6. Search for user
7. Check if follows
8. Follow
9. Unfollow
Functionality requirements

10. Recommend users to follow
   - For user U recommend two-step away followees
   - Users followed by the users that U follows
   - Rank them according to how many users follow them
   - Don’t include users that U already follows
User recommendation example

For a given user 'U', your system should recommend users to follow. In short, we want to recommend the two-step-away followees, that is, the followees of the followees, in some priority order, and, of course, excluding users that 'U' is following already. 'Priority' is the count of paths connecting user 'U' to the recommended user 'V'.

For example, consider the who-follows-whom graph of Figure 1. For user 'U1', your system should recommend 'U3', 'U4', in this order. The reason is that:

a) the followees of the followees of 'U1' are ('U3', 'U6', 'U3', 'U4', 'U6');
b) 'U3' appears twice (paths through 'U2' and 'U5'), and gets priority over 'U4' (who has a single path, through 'U5' only);

c) 'U6' disqualifies since 'U1' already follows him.

Recommend users to U1

- U3, rank 2
- U4, rank 1

Note that U6 is not recommended!
Functionality requirements

11. List all followers & followees
12. List all tweets of a user
13. String search in tweets
14. User statistics
   •   #followers
   •   #followees
   •   #tweets
15. Global statistics
   •   Most popular user
   •   Most active
   •   Most connected
Example web application

http://gs11696.sp.cs.cmu.edu/~abeutel/flitter/
Homework Specifics

• Follow the design methodology from Lecture 19
• Organized in 2 Phases
  • Phase 1 – Design: due 4/1
  • Phase 2 – Implementation: due 4/10
Phase 1

• You are free to come up with your own design choices as long as
  • they follow the methodology
  • they are reasonable
  • you are able to justify unconventional choices
Phase-I

- Description
  - Req. Anal.
  - Top Level I.F.D.
  - Sys. Anal.
  - Task + Doc Forms.

Phase-II

- Impl. + Test.
  - Code.
  - Tests
  - User’s Man.

- Conc. Mod.
  - Schema.
  - Task Emul.
  - Pseudo-Code
Top level information flow diagram

registration form

T1-reg.

user record

external document (web forms)

tasks

internal document (database tables)

System boundary
Phase-I

- **description**
- **req. anal.**
- **top level I.F.D.**
- **sys. anal.**
- **task + doc forms.**

Phase-II

- **impl.+test.**
- **code.**
- **tests**
- **user’s man.**

- **conc. mod.**
- **schema.**
- **task emul.**
- **pseudo-code**
Document + Task forms

Task forms and task list
- not required for this homework

Document forms and document list

- D1: registration form
- D2: login form
- D3: timeline form
- ...
- Dx: user record
- ...

- external
- internal
Document forms

D1: registration form
- username
- Password

Dx: user record
- username
- Password
Phase-I

1. **description**
   - req. anal.
2. **top level I.F.D.**
3. **sys. anal.**
   - task + doc forms.

Phase-II

1. **conc. mod.**
2. **schema.**
3. **task emul.**
4. **pseudo-code**
5. **impl.+test.**
   - code.
   - tests
   - user's man.
E-R diagram

- Specify cardinalities
- Think about weak/strong entities
- Justify unconventional choices
Relational schema

- Give the definition of the schema
- Give SQL DDL statements including constraints.
Task emulation/pseudo-code

- No need to write pseudocode
- Simply give all SQL DML statements for all tasks
Phase 1: What to hand-in

- Due 4/1
- Hard copy (in class)
- Electronic copy (Blackboard)
CMU SCS

Phase-I

description

req. anal.

top level I.F.D.

sys. anal.

task + doc forms.

Phase-II

impl.+test.

code.

tests

user’s man.

conc. mod.

schema.

task emul.

pseudo-code
Phase 2

• We provide an API in PHP
• Implements the web site functionality
• Has empty calls to the database
• write PHP code that
  • wraps the SQL statements
  • returns the output to the rest of the given code (PHP arrays)
• No need to provide user manual
Phase 2

• Unzip hw7.zip
• You need to edit 2 files
  • config.php
    • add your login & url info
  • functions.php
    • Contains empty definitions of the functions that you have to implement
<?php
// Connecting, selecting database
$dbconn = pg_connect("host=localhost dbname=publishing user=www password=foo")
    or die('Could not connect: ' . pg_last_error());

// Performing SQL query
$query = 'SELECT * FROM authors';
$result = pg_query($query) or die('Query failed: ' . pg_last_error());

// Printing results in HTML
echo "<table>\n";
while ($line = pg_fetch_array($result, null, PGSQ\ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";

// Free resultset
pg_free_result($result);

// Closing connection
pg_close($dbconn);
?>

PHP arrays

Array creation:

```php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);
```

Bulk insertion (like stack):

```php
<?php
$stack = array("orange", "banana");
array_push($stack, "apple", "raspberry");
print_r($stack);
?>
```

Securing your application

• SQL injection

```
statement = "SELECT * FROM users WHERE name ="" + userName + ";"
```

• Set name equal to

' or '1'='1

• The SQL statement that gets executed is

```
SELECT * FROM users WHERE name = "" OR '1'='1';
```

• Results in un-authorized log-in!!!!

• Your code has to account for that
  • Hint: pg_escape_string()
Phase 2: What to hand-in

• **Due 4/10**

• **Website (IMPORTANT):** See hw7.pdf for details

• **Hard copy (in class):** ONLY new/changed code (save the trees 😊)

• **Electronic copy:** A .zip with all the code
Homework 7: Architecture

Any host

Client
Browser

CMU Contributed Web Server

Web Server

Apache

Flitter Web app
PHP

CMU AFS

andrew_id/www

PostgreSQL
Database Server

http

hw7
database
Access to web server

• You will use the Computer Club Contributed Web Server
• Apache server + Postgres DB server
• Publishes *.php code in your AFS ‘www’ directory
• More details
  • http://www.club.cc.cmu.edu/doc/contribweb.php
  • HW7 description (read carefully)
Publishing your web app

• Please do the following ASAP and let us know if it doesn’t work!

1. Sign up for the web server here
   http://my.contrib.andrew.cmu.edu

2. Create DB user account here
   http://www.club.cc.cmu.edu/doc/contribweb/sql.php

3. Unzip hw7.zip and copy contents on folder ‘flitter_s14’ under your AFS www directory

4. Edit config.php with your own db+server parameters

5. Edit folder content permissions: chmod +rx

6. Go to
   http://www.contrib.andrew.cmu.edu/~andrew_id/flitter_s14
Questions?

• Come to **office hours** (4 TAs + 2 instructors)
• Post your questions on **blackboard**.