CMPUT 391
Randy Hu, Jian Le Tang, Dennis Truong
Term Project
April 1, 2016

Table of Contents

UML Class Diagram... 3
Setup Script.. 3
Class Descriptions.. 4
User Management...
Implementation.. 4
SQL Statements... 4
Security.. 5
Implementation.. 5
SQL Statements... 5
Uploading.. 6
Implementation.. 6
SQL Statements... 7
Display... 7
Implementation.. 7
SQL Statements... 8
Search...
. Implementation.. 8
SQL Statements... 8
Data Analysis... 9
Implementation.. 9
SQL Statements... 9

UML Class Diagram

Data Layer

Setup Script

The setup script provided in the assignment's specifications was updated to include the following:

```
INSERT INTO users VALUES ('admin', 'admin', sysdate);

CREATE SEQUENCE group_id_sequence

START WITH 3

INCREMENT BY 1

CACHE 100;


CREATE TABLE imagecount (

count_id int,

photo_id int,
user_name varchar(24),

PRIMARY KEY (count_id),

FOREIGN KEY (photo_id) REFERENCES images,

FOREIGN KEY (user_name) REFERENCES users);


CREATE SEQUENCE count_id_sequence

START WITH 1 INCREMENT BY 1 nomaxvalue;


CREATE INDEX myimageindex ON images(description) INDEXTYPE IS CTXSYS.CONTEXT;

CREATE INDEX mysubindex ON images(subject) INDEXTYPE IS CTXSYS.CONTEXT;

CREATE INDEX mylocindec ON images(place) INDEXTYPE IS CTXSYS.CONTEXT;


CREATE SEQUENCE image_id_sequence

START WITH 1

INCREMENT BY 1 nomaxvalue;
```

Class Descriptions

User Management

This module allows clients to register an account by providing some required information, including a unique user name, password, first name, last name, address, email, and phone number, and to log in as a registered user to perform various tasks.

Implementation

JSP pages:

- login.jsp
    - Provides the layout for the login page.
    - Login button calls the loginservlet.java class.
- register.jsp
    - Provides the layout for the register page.
    - Register button calls the registerservlet.java class.
- logout.jsp
    - Removes session variables.
    - Redirects to login.jsp.
- account_settings.jsp
    - Provides layout for account settings page.
    - Send Changes button calls manageUser.java servlet.

Servlets:

- loginservlet.java
    - Verfies the username and password combination entered by the user is in the database.
    - Sets the username session variable.
    - Redirects user to login.jsp if the username and password is invalid.
- registerservlet.java
    - Takes the information entered by the user in the register.jsp page and enters it in the database.
        - Verifies that the username and email do not already exist in the database.
        - Verifies the two passwords entered match.
- manageUser.java
    - Gets and displays the user's settings.
    - Updates the database to reflect any settings changes input by the user.
    - If the user enters a new email address, checks that the email address is not currently in use.
    - If a new password is entered, ensures the passwords in both boxes are equal.

SQL Statements

getpassword: "select password from users where user_name = '" **+** username **+** "'"**;**

addUser: "insert into users values ('" **+** username **+** "', '" **+** password **+** "', sysdate)"**;**
setEmail: "select user_name, email from persons where " **+** "user_name = '"
    + username **+** "'"**;**
updateFname: "update persons set first_name = '" **+** fname **+** "' where user_name = '"
    + username **+** "'"**;**
updateLname: "update persons set last_name = '" **+** lname **+** "' where user_name = '"
    + username **+** "'"**;**
updateAddress: "update persons set address = '" **+** address **+** "' where user_name = '"
    + username **+** "'"**;**

updatePhone: "update persons set phone = '" **+** phone **+** "' where user_name = '"
+ username **+** "'";
emailExists**:** "select email from persons where email = '" **+** email **+** "'";
get_user**:**
- "select * from users " + "where user_name = '" + username + "'";

- "select * from persons " + "where user_name = '" + username + "'";

Security

The security module enables users to specify and update the security group information.

Please note that an image's security is set in the uploading module, and updated in the browse module.

Implementation

JSP pages:
- manage_Groups.jsp
- ◦ Provides the layout for the manage groups page.
- ◦ Submit button calls the manage_Groups.java servlet.

Servlets:
- manage_Groups.java
  - Deletes from the database groups or users that have a checked checkbox.
    - If a photo's security is set to the deleted group, the security is changed to private.
  - Adds new groups or friends when the user provides a friend name or group name.
    - Checks that the entered friend's username is in the database.

SQL Statements

add_group**:** "insert into groups " **+** "values ("**+** "group_id_sequence.nextval, '" **+** user **+** "', '" **+**
group_name **+** "', sysdate)"**;**

delete_group**:**
- "update images set permitted = 2 " + "where permitted = " + group_id**;**
- "delete from group_lists where group_id = " + group_id**;**
- "delete groups where group_id = " + group_id**;**

add_friend**:** "insert into group_lists values(" **+** group_id **+** ", '" **+** friend **+** "', sysdate, null)"**;**

delete_friend**:** "delete from group_lists where group_id = " **+** group_id **+** " and friend_id = '" **+** friend **+** "'"**;**

Uploading

The uploading module allows a registered user to upload images to the database. It also allows a user to enter and update descriptive information for the uploaded images.

Implementation

JSP pages:
- uploadimage.jsp
◦ Provides the layout for the upload image page.
◦ Upload button calls the uploadImage.java servlet.
- uploadFolder.jsp
◦ Provides the layout for the upload folder page.
◦ The jupload upload button calls the uploadFolder.java servlet.
◦ Upload button calls folderDesc.java servlet.

Servlets:
- uploadImage.java
◦ Allows the user to select an image for uploading to the database.
◦ Inserts an image and the image's descriptions into the database using the ImageUploader class.
▪ Checks that a file was selected by the user before calling ImageUploader.
- uploadFolder.java
◦ Allows the user to select a folder of images for uploading to the database using the jUpload app.
◦ Images in the folder are added to an ArrayList of FileItems. The ArrayList is set as the all_files sessionVariable. The images are uploaded in the folderDesc.java servlet.
- folderDesc.java
◦ Uploads the images in the all_files session variable and the image descriptions to the database using the ImageUploader class.
- ImageUploader.java
◦ upload_image(): Performs the upload of a Photo object to the database.
◦ shrink(BufferedImage image): Shrinks the image to a thumbnail.
◦ write_image(BufferedImage img, Blob myImage): writes the image into the database.

SQL Statements

get_groups**:** "SELECT group_id, group_name " **+** "FROM groups " **+** "WHERE user_name = '" **+**
username **+** "'"**;**

addEmptyImage**:** "insert into images values (" **+** image**.**getPhotoId**()** **+** ", '" **+**
image**.**getOwnerName**()** **+** "', " **+** image**.**getPermitted**()** **+** ", '" **+** image**.**getSubject**()** **+** "', '" **+**

image.getLocation() + "', " + date + ", '" + image.getDescription() + "', empty_blob(), empty_blob())";
database.getImageById: "SELECT * FROM images WHERE photo_id = " + image_id + " FOR UPDATE";

Display

This module displays a list of photo thumbnails and descriptions. The user must select the type of photo grouping they would like to display. When a thumbnail is selected, the full size photo is displayed. A client may also edit the photo's descriptions by clicking the “Edit Image Description” link.

Implementation

JSP pages:
- choosePhotos.jsp
  - Allows user ability to choose which group of photos to display – private, public, or the groups of which they are a part.
  - Submit buttons calls the browsePictures.java servlet.
- imageDesc.jsp
  - Displays and allows for the updating of an image's descriptors.
  - Upload button calls the imageDesc.java servlet.

Servlets:
- browsePictures.java
  - Displays photos and their descriptions according to the user's selection in choosePhotos.jsp.
  - Allows “admin” user access to all photos.
  - Clicking a photo thumbnail will call the browsePicture.java servlet.
- browsePicture.java
  - Displays photo selected in the browsePictures.java servlet.
  - If the user is viewing the image for the first time, the image's view count is incremented, and the user is added to the imagecount table.
  - Checks the photo's permissions one last time.
- imageDesc.java
  - Updates an image's descriptors in the database.

SQL Statements

getParticipantGroups: "SELECT group_id, group_name FROM groups WHERE " + "group_id in " + "(SELECT group_id FROM group_lists where " + "friend_id = '"+username+"') OR " + "user_name = '"+username+"'";
getPhotoDesc: "select photo_id, owner_name, permitted, subject, " + "description, place, timing " + "from images where photo_id = " + id;

get_groups**:** "SELECT group_id, group_name " **+** "FROM groups " **+** "WHERE user_name = '" **+** username **+** "'"**;**
imageCountViewInc:

- "select user_name, photo_id from imagecount where " + "photo_id = '" + photoId + "' and user_name = '"+username+"'"**;**
- "insert into imagecount " + "values(count_id_sequence.nextval, '"+photoId+ "', '"+username+"')"**;**

update_photo_desc**:** UPDATE images SET " **+** "permitted = " **+** photo**.**getPermitted**()** **+** ", " **+** "subject = '" **+** photo**.**getSubject**()** **+** "', " **+** "place = '" **+** photo**.**getLocation**()** **+** "', " **+** "timing = " **+** date **+** ", " **+** "description = '" **+** photo**.**getDescription**().**trim**()** **+** "' " **+** "WHERE photo_id = " **+** photo**.**getPhotoId**();**

Search

This module allows the user to search for a photos by entering a list of keywords and/or a time period. The module displays the matching photos in the order of their ranking.

Implementation

JSP Pages:

- search.jsp
  - Allows user to enter search terms, search dates, and select the display ordering.
  - Search button calls the searchResults.java servlet.

Servlets:

- searchResults.java
  - Displays thumbnails of pictures matching the query entered in search.jsp

SQL Statements

getResultByKeywords**:** "SELECT score(1)*6 + score(2)*3 + score(3) AS score, " **+** "photo_id FROM images WHERE " **+** "((contains(subject, '"**+** keywords **+** "', 1) > " **+** "0) OR (contains(place, '" **+** keywords **+**"', 2) > 0) " **+** "OR (contains(description, '" **+** keywords **+** "', 3) > " **+** "0)) " **+** order**;**
getResultsByDateAndKeywords: "SELECT score(1)*6 + score(2)*3 + score(3) AS score, " **+** "photo_id FROM images WHERE " **+** "((timing BETWEEN '" **+** fromdate **+** "' AND '" **+** todate **+** " ') AND (contains(subject, '"**+** keywords **+** "', 1) > " **+** "0) OR (contains(place, '" **+** keywords **+**"', 2) > 0) " **+** "OR (contains(description, '" **+** keywords **+** "', 3) > " **+** "0)) " **+** order**;**
getResultsByDate**:** "SELECT timing, photo_id FROM images WHERE (timing BETWEEN '" **+** fromdate **+** "' AND '" **+** todate **+** "') " **+** order**;**

Data Analysis

This module is used by the system administrator (username: admin, password: admin) to generate and display an OLAP report for data analysis. A user of this module may choose to display the number of images for each user, subject, and/or period of time.

Implementation

Servlets:

- adminStats.java
  - Verifies the logged in user is admin.
  - Allows user to select the timeframe for data analysis.
  - Submit button refreshes the page.
  - Once a timeframe has been selected, OLAPCommands.java handles the data fetching.
- OLAPCommands.java
  - OLAPCommands(String timeframe):
    - This class handles the specification of the timeframe. All of the following methods rely on the given timeframe specified in this object creation.
  - getColTitle(Calendar myDate): parse out a column title for the row's results
  - getRegUsersCount(): Returns the total count of users registered for the specified timeframe format
  - getRegUsers(): Returns the total details of users registered for the specified timeframe format
  - getDateUploadImagesCount(): Returns the total count of images uploaded for the specific time date format
  - getDateUploadImages(): Returns the total number of photos uploaded for the specific timeframe format
  - getImgSubjCount(): Returns a formatted table of subjects and counts grouped by the specified time quantum
  - getImgUsrCount(): Returns a formatted table of users and counts grouped by the specified time quantum

SQL Statements

GetRegUsersCount: "SELECT date_registered as timeframe, " + "user_name from users order by " + "timeframe";
getRegUsers: "SELECT date_registered as timeframe, " + "user_name from users order by " + "timeframe";
getDateUploadImagesCount: "SELECT timing as timeframe, photo_id from images " + "order by timeframe";
getDateUploadImages: "SELECT timing as timeframe, photo_id, owner_name, " + "subject, place, description from images " + "order by timeframe";
getImgSubjCount: "select timing, photo_id, subject from images" + " order by timing asc";
getImgUsrCount: "select timing, photo_id, owner_name from images" + " order by timing asc";