Project Report

**CMPUT 391 Project Report**

**--**Database Management Systems (Fall 2013)

group members:

Bowen Qi(1373662),

Yuwei Duan(1249703),

Guanqi Huang(1246777)

Database

**Uploading module:**

This module allows a user to upload a single file or multiple files. User can either choose a single file from the local computer or multiple files at the same time.

**Jsp contains:**

**SingleUpload.jsp**

**singleUpload.jsp**

SingleUpload.jsp creates a form which allows user to select the images that user wants to upload and also allows user to fill the information for the images. If user leave the information area blank null value will be insert into database.

SQL:

"select group\_id, group\_name from groups where user\_name = '" **+** username **+**"'"

This query will get the group id and name to indicate which group the upload images belongs to

singleUpload.jsp is the responder of SingleUpload.jsp. When user hit the submit button. singleUpload.jsp will inset all information of all files, thumbnail and origin images into the database.

SQL:

getGroup **=** "select group\_id from groups where group\_name = '" **+** permit **+** "' and user\_name = '" **+** username **+** "'" **;**

This query is used to check if the group name exist.

"insert into images values (" **+** id **+** ", '" **+** username **+** "' , " **+** permit **+** ", '" **+** subject **+** "' , '" **+** place **+** "' , '" **+** time **+** "', '" **+** description **+** "', empty\_blob(), empty\_blob())"

This query is used to initialize a new image with two empty blobs

"SELECT \* FROM images WHERE photo\_id = " **+** id **+** " FOR UPDATE"**;**

This query is used to retrieve the lob\_locator

"select THUMBNAIL from images where photo\_id = " **+** id

This query is used to get the thumbnail of the images the system just inserted into the database.

**Security Module:**

The Security module is mainly used for some safe reasons. We use prepare sql sentence in the jave which can prevent some sql security bugs. We also provide session and request for the identification different user. If a user do not login and directly type in the url, system will jump to the login page directly. Also, after log out, we will not allow the user to use the function by just using back on the website. The session info is change from a user to a guest. We also use installation program to prevent some hacker get information about our database by just review the

page.

**Group Module:**

The Group module is fundamentally used for users manage his group and open specific images to his friends. A user can add groups by typing in only an unique group name or delete groups by selecting a existing group name. Also , user can add or delete a user of a group by only selecting the group name and the friend's name and click the submit button.

**Jsp contains:**

**CreateGroup.jsp**

**AddGroup.jsp**

**DeleteGroup.jsp**

**AddMember.jsp**

**DeleteMember.jsp**

**CreateGroup.jsp is the base of the group module which contains all 4 functions:**

**Add a group**

**delete a group**

**add a group member**

**delete a group member**

**which connect to AddGroup.jsp, DeleteGroup.jsp, AddMember.jsp and DeleteMember.jsp, repectively.**

**SQL:**

**CreateGroup.jsp:**

"select group\_id, group\_name from groups where user\_name = '" **+** username **+**"'"

This query is used to get all groups that the current user has.

"select user\_name from users where user\_name <> '" **+** username **+** "'"

This query is used to all other users that not the current user.

"select gl.friend\_id from group\_lists gl, groups g where gl.group\_id = g.group\_id and g.user\_name = '" **+** username **+** "'"

This query is used to get all friends that the given group has.

AddGroup.jsp:

"select \* from groups where '" **+** groupName **+** "' = group\_name and user\_name = '" **+** username **+** "'"

This query is used to check if the given group name exists.

"insert into groups values(" **+** id **+** " , '" **+** username **+** "' , '" **+** groupName **+** "' , sysdate)"

This query is used to insert the new group into the database

DeleteMember.jsp:

"select \* from groups where '" **+** groupName **+** "' = group\_name and user\_name = '" **+** username **+** "'"

This query is used to check if the given group name exists.

"select \* from group\_lists where '" **+** friendName **+** "' = friend\_id"

This query is used to check is the friend name already existed in the group list.

"select group\_id from groups where group\_name = '" **+** groupName **+** "'"

This query is used to get the group id by given group name.

"select \* from group\_lists where friend\_id = '" **+** friendName **+** "' and group\_id =" **+** id

This query is used to check is the user exists in the given group.

"delete from group\_lists where group\_id =" **+** id **+** " and friend\_id = '" **+** friendName **+**"'"

This query is used to delete the group member after all checking.

DeleteGroup.jsp

"select \* from GROUPS where '" **+** groupName **+** "' = group\_name and user\_name = '" **+** username **+** "'"

This query is used to check is the group name exists.

"delete from GROUPS where '" **+** groupName **+** "' = group\_name"

This query is used to delete the given group.

"select group\_id from groups where '" **+** groupName **+** "' = group\_name"

This query is used to get the group id by given the group name.

"update images set permitted = 2 where permitted = " **+** groupID

This query is used to set all images that belong to the deleted group permitted to private.

"delete from group\_lists where group\_id = " **+** groupID

This query is used to delete the friend list that related to the deleted group.

AddMember.jsp

"select \* from groups where '" **+** groupName **+** "' = group\_name and user\_name = '" **+** username **+** "'"

This query is used to check is the group name exists.

"select group\_id from groups where group\_name = '" **+** groupName **+** "'"**;**

This query is used to get the group id by given group name.

"select \* from group\_lists where '" **+** friendName **+** "' = friend\_id and group\_id = " **+** id

This query is used to check is the friend name already existed in the given group.

"insert into group\_lists values(" **+** id **+** " , '" **+** friendName **+** "' , sysdate, '" **+** notice **+** "')"

This query is uesd to insert the friend into the group list.

Login module:

**login.jsp**

Retrieve the username and password from login form in html, and validate from database. If them are correct, it will redirect to login\_success.jsp, otherwise it will redirect to login\_failure.html.

SQL statement: "select \* from users where '" + userName + "' = user\_name and password = '" + password + "'"

Register module:

**regieter.jsp**

**extral\_profile.jsp**

**update\_profile.jsp**

For the registration, you need full name, email, address, phone, username and password. We use JavaScript to validate the format of input. And first I check whether the information existed, and then insert new values into database.

SQL statement:

1. SELECT \* from USERS WHERE USER\_NAME='"+username+"'"

2. SELECT \* from persons where email ='"+email+"'"

3. INSERT INTO USERS VALUES(?,?,sysdate)

4. NSERT INTO persons VALUES(?,?,?,?,?,?)

If register is successful, it will login automatically. Otherwise it requests typing information again.

For changing personal information, the password could not be null.

SQL Statement:

UPDATE USERS SET PASSWORD = ? WHERE USER\_NAME=?

Search module:

**search\_process.jsp**

For search module, we need user type key words and choose rank modes, which is most-recent-first or most-recent-last or ranking by the formula:

Rank(photo\_id) = 6\*frequency(subject) + 3\*frequency(place) + frequency(description)

The frequency indicates the number of key words in the respective columns.

For the rank by relevent:

SQL statement:

SELECT key0.photo\_id, key0.score0 as rank, key0.timing FROM (SELECT photo\_id, 6\*SCORE(1)/5 + 3\*SCORE(2)/5 + SCORE(3)/5 as score0, timing FROM images WHERE CONTAINS(subject, 'key', 1) > 0 OR CONTAINS(place, '1231', 2) > 0 OR CONTAINS(description, 'key', 3) > 0) key0 WHERE 1>0 ORDER BY rank

For rank by most-recent-last :

**SQL Statement:**

SELECT key0.photo\_id, key0.score0 as rank, key0.timing FROM (SELECT photo\_id, 6\*SCORE(1)/5 + 3\*SCORE(2)/5 + SCORE(3)/5 as score0, timing FROM images WHERE CONTAINS(subject, '1231', 1) > 0 OR CONTAINS(place, '1231', 2) > 0 OR CONTAINS(description, '1231', 3) > 0) key0 WHERE 1>0 ORDER BY key0.timing DESC

SQL Statement:

SELECT key0.photo\_id, key0.score0 as rank, key0.timing FROM (SELECT photo\_id, 6\*SCORE(1)/5 + 3\*SCORE(2)/5 + SCORE(3)/5 as score0, timing FROM images WHERE CONTAINS(subject, '1231', 1) > 0 OR CONTAINS(place, '1231', 2) > 0 OR CONTAINS(description, '1231', 3) > 0) key0 WHERE 1>0 ORDER BY key0.timing

The time interval should be seperated by common

SQL Statement:

SELECT key0.photo\_id, key0.score0 as rank, key0.timing FROM (SELECT photo\_id, 6\*SCORE(1)/5 + 3\*SCORE(2)/5 + SCORE(3)/5 as score0, timing FROM images WHERE CONTAINS(subject, '1231', 1) > 0 OR CONTAINS(place, '1231', 2) > 0 OR CONTAINS(description, '1231', 3) > 0) key0 WHERE ( key0.timing >= to\_date('01-JAN-2013', 'DD-MON-YY') AND key0.timing <= to\_date('21-JAN-2013', 'DD-MON-YY') ) ORDER BY rank

This is searching between 01-JAN-2013 and 21-JAN-2013