Queries and Functional Dependencies of Flickr.com(Part-2)

Group No. 28

Kirnesh Nandan (13312009)

Mohak Garg (13312013)

Rahul Jain(13311016)

Mithilesh Kumar(13312012)

|  |  |  |  |
| --- | --- | --- | --- |
| Table | Functional Dependency | Minimal Cover | Candidate Keys |
| User Table | User\_ID->  Name,Password,Web\_Address  Web\_Address->User\_ID | User\_ID->Name  User\_ID->Password  User\_ID->Web\_Address  Web\_Address->User\_ID | {User\_ID},  {Web\_Address} |
| Album Table | Album\_ID->  Album\_Name,Owner\_User\_Id,  Date\_Created,Description | Album\_ID->Album\_Name  Album\_ID->Owner\_User\_ID  Album\_ID->Date\_Created  Album\_ID->Description | {Album\_ID} |
| Photos | Photo\_ID->  Date\_Created,Privacy,Who\_can\_comment,Description,Owner\_User\_ID | Photo\_ID->Date\_Created  Photo\_ID->Privacy  Photo\_ID->  Who\_can\_comment  Photo\_ID->Description  Photo\_ID->  Owner\_User\_ID | {Photo\_ID} |
| Group | Group\_ID->  Group\_Name,Date\_Created | Group\_ID->Group\_Name  Group\_ID->Date\_Created | {Group\_ID} |
| Belongs\_to\_Album | Trivial FD | Trivial Dependencies | {Album\_ID,Photo\_ID} |
| Belongs\_to\_Group | Trivial FD | Trivial Dependencies | {Group\_ID,Photo\_ID} |
| Profile\_pic | Photo\_ID->User\_ID,User\_ID->  Photo\_ID | Photo\_ID->User\_ID  User\_ID->Photo\_ID | {User\_ID},{Photo\_ID} |
| Likes | Trivial FD | Trivial Dependencies | {Photo\_ID,User\_ID} |
| Member | User\_ID,Group\_ID->Type | User\_ID,Group\_ID->Type | {User\_ID,Group\_ID} |
| Follows | Trivial FD | Trivial Dependencies | {User\_ID,Follow\_ID} |
| Followed\_ID | Trivial Fd | Trivial Dependencies | {User\_ID,Following\_ID} |

We can see that all the tables are already in BCNF. So, none of them needs to be decomposed. However , all tables are in BCNF as we have created E-R diagram and Relational Schema keeping in mind a little bit of normalization and by luck we have got all in BCNF.

Assumptions:

1.Profile Pic has to be from the same user account

2.User can’t hold more than one tags(“Member”,”Admin”,”Moderator”) for attribute Member\_Type of Member table

3.Since there was no attribute for favourite photos in our E-R diagram, we included that by allowing user to like his own photo.

Basic Operations of the Website along with queries and its results:

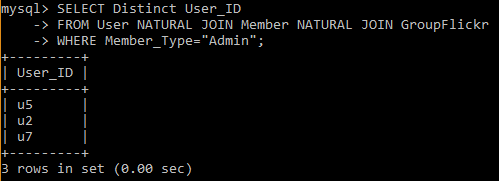
* It shows stats tab for each user like groups in which they are admins.

Query1: List the users that are admin of at least one group.

**SELECT** **Distinct** User\_ID

**FROM** User **NATURAL JOIN** Member **NATURAL JOIN** GroupFlickr

**WHERE** Member\_Type=”Admin”;



* It keeps track of popular personalities or figures by analyzing number of their followers.

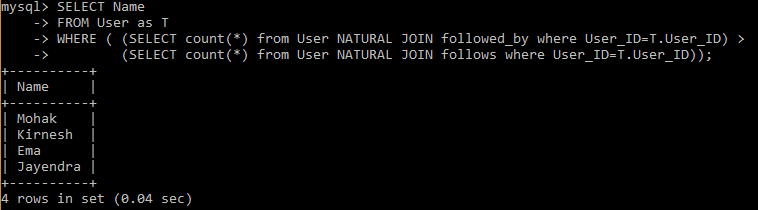
Query2: Find the name of all the users having more followers than the people they follow

**SELECT** Name

**FROM** User **as** T

**WHERE**( (**SELECT** count(\*) **from** User **NATURAL JOIN** followed\_by **where** User\_ID=T.User\_ID) >

(**SELECT** count(\*) **from** User **NATURAL JOIN** follows **where** User\_ID=T.User\_ID));



* It keeps track of favorite photos in the favorite tab (Favorite photos is that one which is liked by his own user).

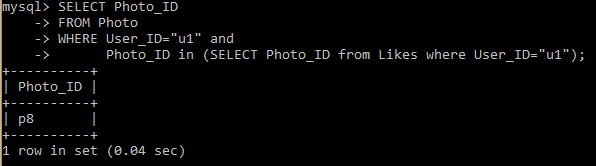
Query3: Find favorite photos of user ID u1.

**SELECT** Photo\_ID

**FROM** Photo

**WHERE**User\_ID=”u1” **and**

Photo\_ID in (**SELECT** Photo\_ID **from** Likes **where** User\_ID=”u1”);



* It shows number of likes on the profile pic of each user and decides whose profile pic is trending.

Query4: All users having number of likes on their profile pic greater than 2.

**CREATE VIEW** PP **as**

(**SELECT** Photo\_ID, User\_ID **as** PPUser

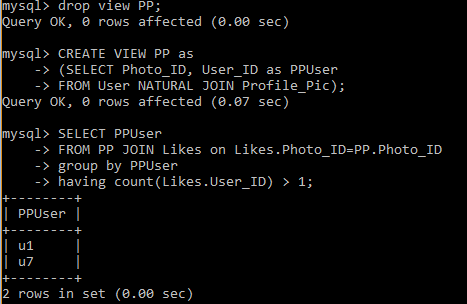
**FROM** User **NATURAL JOIN** Profile\_Pic);

**SELECT** PPUser

**FROM** PP **JOIN** Likes **on** Likes.Photo\_ID=PP.Photo\_ID

***group by*** PPUser

***having*** count(Likes.User\_ID) > 1;



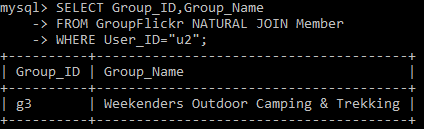
* It also shows in the groups tab in which group you are joined

Query5: Show the names of groups joined by any particular user with User\_ID “u2”.

**SELECT** Group\_ID,Group\_Name

**FROM** GroupFlickr **NATURAL** **JOIN** Member

**WHERE** User\_ID=”u2”;



* It shows top contributors from each group.

Query6: List the top contributors of each group along with user ID and group ID.

**CREATE VIEW** TopGroup **as**

(**SELECT** GroupFLickr.Group\_ID,User\_ID,count(Photo\_ID)as countphoto

**FROM** Photo **NATURAL JOIN** Belongs\_to\_Group,GroupFlickr

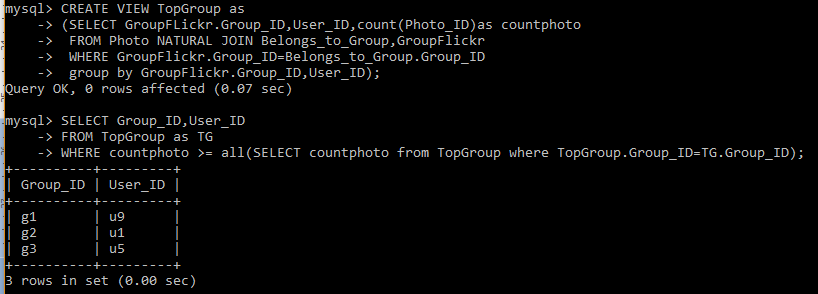
**WHERE** GroupFlickr.Group\_ID=Belongs\_to\_Group.Group\_ID

***group by*** GroupFlickr.Group\_ID,User\_ID);

**SELECT** Group\_ID,User\_ID

**FROM** TopGroup as TG

**WHERE** countphoto >= **all**(**SELECT** countphoto **from** TopGroup **where** TopGroup.Group\_ID=TG.Group\_ID);



Query7: List all the users whose Privacy of all the photos in any of their album is marked as private.

**SELECT** User\_ID

**FROM** Album

**WHERE** Album\_ID in (**SELECT** Album\_ID **FROM** Album as al

**WHERE EXISTS (Select** \* **From** Belongs\_to\_Album **natural join** Photo

**where** Album\_ID=al.Album\_ID and Privacy=”Private”));

