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”;
```

```
mysql> SELECT Distinct User_ID
    -> FROM User NATURAL JOIN Member NATURAL JOIN GroupFlickr
    -> WHERE Member_Type="Admin";
+---------+
| User_ID |
+---------+
| u5      |
| u2      |
| u7      |
+---------+
3 rows in set (0.00 sec)
```

- 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));
```

```
mysql> 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));
+----------+
| Name     |
+----------+
| Mohak    |
| Kirnesh  |
| Ema      |
| Jayendra |
+----------+
4 rows in set (0.04 sec)
```

- 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”);
```

```
mysql> SELECT Photo_ID
    -> FROM Photo
    -> WHERE User_ID="u1" and
    ->       Photo_ID in (SELECT Photo_ID from Likes where User_ID="u1");
+----------+
| Photo_ID |
+----------+
| p8       |
+----------+
1 row in set (0.04 sec)
```

- 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;
```

```
mysql> drop view PP;
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE VIEW PP as
    -> (SELECT Photo_ID, User_ID as PPUser
    -> FROM User NATURAL JOIN Profile_Pic);
Query OK, 0 rows affected (0.07 sec)

mysql> SELECT PPUser
    -> FROM PP JOIN Likes on Likes.Photo_ID=PP.Photo_ID
    -> group by PPUser
    -> having count(Likes.User_ID) > 1;
+--------+
| PPUser |
+--------+
| u1     |
| u7     |
+--------+
2 rows in set (0.00 sec)
```

- 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”;
```

```
mysql> SELECT Group_ID,Group_Name
    -> FROM GroupFlickr NATURAL JOIN Member
    -> WHERE User_ID="u2";
+----------+---------------------------------------+
| Group_ID | Group_Name                            |
+----------+---------------------------------------+
| g3       | Weekenders Outdoor Camping & Trekking |
+----------+---------------------------------------+
```

- 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);
```

```
mysql> 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);
Query OK, 0 rows affected (0.07 sec)

mysql> SELECT Group_ID,User_ID
    -> FROM TopGroup as TG
    -> WHERE countphoto >= all(SELECT countphoto from TopGroup where TopGroup.Group_ID=TG.Group_ID);
+----------+---------+
| Group_ID | User_ID |
+----------+---------+
| g1       | u9      |
| g2       | u1      |
| g3       | u5      |
+----------+---------+
3 rows in set (0.00 sec)
```

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”));
```

```
mysql> 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"));
+---------+
| User_ID |
+---------+
| u3      |
| u5      |
+---------+
2 rows in set (0.00 sec)
```