FACULTY OF SCIENCE AND TECHNOLOGY

ACADEMIC YEAR 2019/20

SEMESTER 2

COMP3161 – Database Management Systems

Final Project

Date: May 12, 2020

Group Members:

Jada- Rae Brown	620108004
Lanai Nevers	620112918
Daniel Davis	620106450
Abigail Simpson	620109464

Table of Contents

Introduction

‘myBook’ represents a startup consisting of a group of 4 students who have decided to create a social networking website as Facebook no longer satisfies their experiences. ‘myBook’ has decided to focus on just a few requirements and build on these in the future. They are planning to release a beta version initially and have decided to approach this venture in three phases:

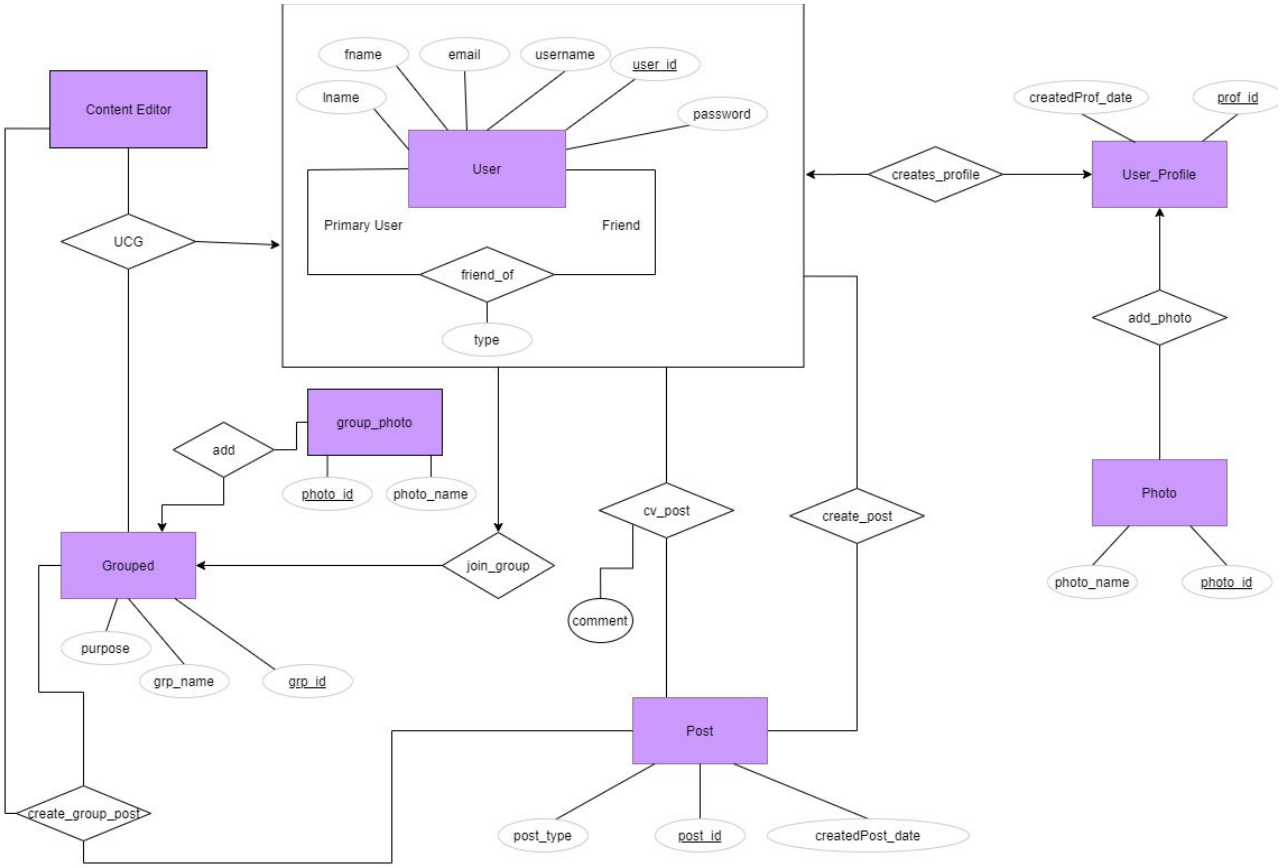
- Phase 1: Focus on the design of myBook.
- Phase 2: Focus on the back-end structure.
- Phase 3: Focus on the creation of a web or mobile front-end to the database.

Design

2.1 Functional Requirements

1. Users shall be able to register to be a part of myBook's social network.
2. Profile shall be created once the user has registered
3. Users shall be able to modify their profiles.
    a. Users shall be able to add photos and choose one as their profile picture.
4. Users shall be able to add friends (Relatives, School, Work).
5. Users shall be able to create Posts (text|images) and only friends should see these posts and comment on these posts.
6. Users shall be able to create a group.
    a. The creator shall be able to add content editors to his/her group and only these persons can post in the groups.
    b. Any user shall be able to add themselves to these groups to see posts in these groups.

2.2 Entity Relationship Diagram



2.2.1 Assumptions

- We did not have a registration entity because we assumed that the attributes for registration would still have been assigned to the user.

- We assumed that one of the photos the user uploaded will be used as the profile photo.

- We assumed that a Friend and Content Editor are a type of user and that the Content Editor may or may not be a friend of the Primary User.

- In order to differentiate between a friend user or non-friend user, a query will be performed on ‘friend_of’ to determine who has the privilege to comment/view posts.

- Through the ternary relationship ‘UCG’, a user is able to create a group, add a content editor and a content editor is able to post in the group.

- The unary relationship that is shown for the entity User covers both friends and non-friends.

2.3 Pre-Normalization Tables

- User (user_id, fname, lname, username, password, email)
- friend_of (user_id, friend_id, type)
- User_profile (prof_id, createdProf_date)
- add_photo (prof_id, photo_id, photo_name)
- Photo (photo_id, photo_name)
- create_post (user_id, post_id, createdPost_date, post_type)
- cv_post (user_id, post_id, createdPost_date, post_type, comment)
- Post (post_id, post_type, createdPost_date)
- UCG (user_id, grp_id, grp_name, purpose)
- Grouped (grp_id, grp_name, purpose)
- join_group(grp_id, user_id)
- create_Grp_post(ce_id, grp_id, post_id)
- group_photo(photo_id, photo_name)

2.4 Functional Dependencies

- user_id → fname, lname, email, password, username
- user_id → grp_id
- user_id → friend_id
- user_id → prof_id
- user_id → post_id
- user_id, friend_id → type
- prof_id → createdProf_date
- prof_id → photo_id
- photo_id → photo_name
- post_id → post_type, createdPost_date
- grp_id → grp_name, purpose

2.5 Post-Normalization Tables

2.5.1 Normal Forms

- All tables pass First Normal Form as all attributes have atomic values.
- Tables ‘cv_post’ violate Second Normal Form as ‘createdPost_date’, ‘post_type’ and ‘comment’ are non-prime attributes that depend on only the post_id.
- Table ‘UCG’ violates the Second Normal Form as ‘group_name’ and ‘purpose’ is only dependent on the grp_id and not both user_id and grp_id
- The ‘add_photo’ table violates Third Normal Form as ‘photo_name’ is dependent on a non-prime attribute ‘photo_id’.
- The ‘create_post’ table violates Third Normal Form as ‘createdPost_date’ and ‘post_type’ is dependent on a non-prime attribute ‘post_id’
- Based on the computation of closure, all tables pass Boyce-Codd Normal Form.

2.5.2 Normalized Tables

- user (user_id, f_name, l_name, username, password, email)
- friend_of (user_id, friend_id, type)
- user_profile (prof_id, createdProf_date)
- add_photo (prof_id, photo_id)
- photo (photo_id, photo_name)
- create_post (user_id, post_id)
- cv_post (user_id, post_id, comment)
- posts (post_id, createdPost_date, description, filename)
- UCG (user_id, CE_id, grp_id)
- grouped (grp_id, grp_name, purpose)
- join_group(grp_id, user_id)
- create_Grp_post(ce_id, grp_id, post_id)
- group_photo(photo_id, photo_name)

2.6 Data Dictionary

User			
Column Name	**Data Type**	**Primary Key**	**Description**
user_id	int	Yes	The primary key for User table
fname	varchar		First name of the user
lname	varchar		Last name of the user
username	varchar		Username for the user
password	varchar		Password of the user
email	varchar		Email for the user

friend_of			
Column Name	**Data Type**	**Primary Key**	**Description**
user_id	int	Yes	Primary key for User table
friend_id	int	Yes	Primary key for User table (representing the user's friend)
type	varchar		Type of friend (relative, school, work)

user_profile			
Column Name	**Data Type**	**Primary Key**	**Description**
prof_id	int	Yes	Primary key for user_profile identifies profile
createdProf_date	date	No	Date when the profile for a user was created

add_photo			
Column Name	**Data Type**	**Primary Key**	**Description**
prof_id	int	Yes	Primary key for creates_profile identifies profile
photo_id	int		Foreign key for Photo, identifies photo added

photo			
Column Name	**Data Type**	**Primary Key**	**Description**
photo_id	int	Yes	Primary key for the Photo Table
photo_name	Char		Name of the photo to be added to the Database

cv_post			
Column Name	**Data Type**	**Primary Key**	**Description**
user_id	int	Yes	Primary key for comment_post table that identifies the user that made a post
post_id	int	Yes	Primary key for comment_post table to identify the post made
comment	varchar		Content of the comment

posts			
Column Name	**Data Type**	**Primary Key**	**Description**
post_id	varchar	Yes	Primary Key for the Post table (uniquely identifies each post made by a user)
createdPost_date	date		Date stamp for each post made
description	varchar		Type of post made (Text or Image)
filename	varchar		Name of the image as saved on the user's device

grouped			
Column Name	**Data Type**	**Primary Key**	**Description**
grp_id	varchar	Yes	Primary Key for the Group table (uniquely identifies each group (of friends) made by the user
grp_name	char		Name of group
purpose	varchar		Outlines why/what the group was created for

UCG			
Column Name	**Data Type**	**Primary Key**	**Description**
grp_id	varchar	Yes	Primary Key from Group table (uniquely identifies each group (of friends) made by the user
CE_id	varchar	Yes	Primary Key from the CE table (uniquely identifies content editor's user ID)
user_id	varchar	Yes	Primary Key from the User table (uniquely identifies the user who created this group)

join_group			
Column Name	**Data Type**	**Primary Key**	**Description**
grp_id	varchar	Yes	Primary Key from Group table (uniquely identifies each group (of friends) made by the user
user_id	varchar	Yes	Primary Key from the User table (uniquely identifies the user who created this group)

create_Grp_post			
Column Name	**Data Type**	**Primary Key**	**Description**
grp_id	int	Yes	Primary Key from Group table (uniquely identifies each group (of friends) made by the user
CE_id	int	Yes	Primary Key from the CE table (uniquely identifies content editor's user ID)
post_id	int	Yes	Primary Key from the post table (uniquely identifies the post made by a user)

group_photo			
Column Name	**Data Type**	**Primary Key**	**Description**
photo_id	int	Yes	Primary key fo group_photo table
photo_name	varchar	No	Name of the photo to be added to the database

Database Implementation

3.1 Stored Procedures

3.1.1 A trigger to store the date a profile is created. Activated after a new entry is added to the User table

```
DELIMITER //
        CREATE TRIGGER Date_Trigger
        AFTER insert ON User
        FOR EACH ROW
        BEGIN
                INSERT into user_profile(prof_id, createdProf_date) values
                (new.user_id,curtime());
        END //
DELIMITER ;
```

3.1.2 A procedure to retrieve the number of friends of a specific user given that user's ID.

```
DELIMITER //
        CREATE PROCEDURE GetNumberFriends(IN user_id INT)
        BEGIN
                SELECT count(distinct friend_id) FROM friend_of WHERE
        user_id = user_id;
        END //
DELIMITER ;
```

To call procedure:

```
CALL GetNumberFriends();
```

3.1.3 A procedure to retrieve the number of groups a user is in.

```
DELIMITER //
        CREATE PROCEDURE GetGroupAmount(IN user_id INT)
        BEGIN
        SELECT count(user_id) FROM UCG WHERE user_id = user_id;
        END //
DELIMITER ;
```

To call procedure:

```
CALL GetGroupAmount();
```