Group 7 - Braylon Steffen and Allen Chan

HTML Link: http://flip3.engr.oregonstate.edu:17154/index

Summary:
In step 1, we started with a 6-table database design that included all tables in this version but Friendships. The feedback for step 1 was to add a M:M relationship, and ensure consistency in table/attribute names (plural vs. singular). So we added the Friendships table, and decided each table would be the plural version of the word. For step 3, we started with more of a front-facing software due to misunderstanding of the project's nature. Because of this, we got some feedback that many of the CRUD elements appeared to be missing in our UI. We decided to start over on the UI to make it a straightforward representation of the database, with the user having full control over the data within, adding buttons for each of the CRUD functionalities. In step 4, we designed the SQL queries that would implement the CRUD functionalities. There were a number of syntax errors, but once those were resolved, the queries remained largely unchanged. In Step 5, we implemented most of the CREATE and READ functionalities for the database. We did not implement them for the Memberships and Friendships tables in time for feedback, but all the others seemed to be working. Additionally, we hadn't used the auto-incrementing ID attributes correctly and as such, the user was able to manually enter an ID value that would skew the ID counts. In Step 6, we completed the CREATE and READ functionalities, and implemented the new UPDATE and DELETE functionalities for each table. This was by far the most challenging aspect of the project due to many errors surrounding variable names and syntax. However, most of these functionalities worked. Finally, in between Step 6 and this final draft, we implemented a bit of error checking and dynamic SQL query generation, so that the website doesn't crash from minimal errors, and the search functionality works with any combination of missing/present attributes.

Cumulative Feedback Summary:
- Add auto incrementing id
- Add insert for friendships and Memberships
- Lack of M:M relationship
- UI does not have any adding, searching, deleting or updating functions
- Change the relationship between Friends and Accounts to M:M relationship
- Moving all the tabs such as "create post" to the top of the page

Main Changes Made:
- Added auto increment id
- Added insert for friendships and memberships
- Added update and delete for Accounts, Comments, Friendships, Groups, Memberships, Messages, and Posts
- Added type checking for some inputs
- Added M:M relationship with composite entity Friendships
- Separate links for posts, comments, and accounts

Project Overview and Database Outline

Overview:

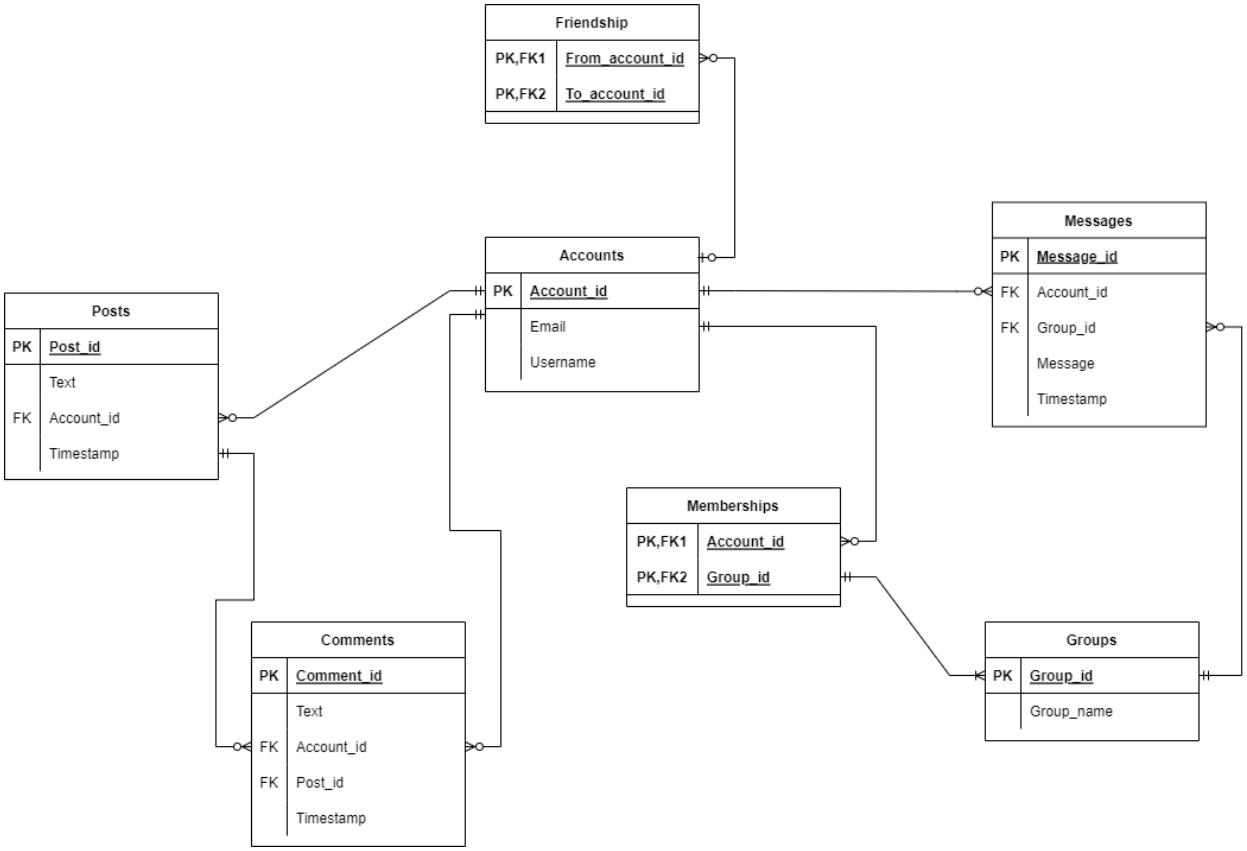
Social media has connected people globally, and generated new revenue streams through advertising. A database driven text-based social media website will allow users to connect and communicate seamlessly, and access content at all times. The social media platform can expect 100,000,000+ Accounts, with 500,000,000+ Messages, Posts, and Comments added on the database every day, as well as 50,000,000+ Groups total. Most Accounts will have a relationship with many Posts, Messages, Comments, and Memberships/Groups. The purpose of this project is to show how a social media platform keeps stores and protects private information of user data while keeping everything organized. A social media database must be dynamic because there will always be new features being added and removed.

Database Outline: Social Media Entities

- **Accounts** - The purpose of this entity is to identify the user's account.
    - **Account_id**: int, auto_increment, unique, not NULL, PK
    - **Email**: string, not NULL
    - **Username**: string, unique, not NULL
    - **Relationship**: a 1:M relationship between Accounts and Posts with Account_id as a FK inside Posts
    - **Relationship**: a 1:M relationship between Accounts and Comments with Account_id as a FK inside Comments
    - **Relationship**: a 1:M relationship between Accounts and Memberships with Account_id as a FK inside Memberships
    - **Relationship:** a 1:M relationship between Accounts and Messages with Acount_id as a FK inside Messages
    - **Relationship:** an indirect M:N relationship between Accounts and Groups, with Memberships linking Account_id and Group_id as a composite entity
    - **Relationship:** a M:M relationship between Accounts and Friendships to show user is friends with another user
- **Posts** - This entity identifies and stores the contents of a post from a given user
    - **Post_id**: int, auto_increment, unique, not NULL, PK
    - **Text**: string, not NULL
    - **Account_id**: int, auto_increment, unique, not NULL, FK
    - **Timestamp**: datetime, not NULL
    - **Relationship**: a 1:M relationship between Posts and Comments with Post_id as a FK inside Comments
- **Comments** - This entity identifies and stores comments on a given post
    - **Post_id**: int, auto_increment, unique, not NULL, FK
    - **Comment_id**: int, auto_increment, unique, not NULL, PK
    - **Account_id**: int, auto_increment, unique, not NULL, FK
    - **Text**: string, not NULL
    - **Timestamp**: datetime, not NULL
- **Memberships** - This will identify an account being a member of a given group
    - **Account_id**: int, auto_increment, unique, not NULL, FK, PK

    - **Group_id**: int, auto_increment, unique, not NULL, FK, PK
    - **Relationship**: a M:1 relationship between Accounts and Groups with Group_id as a FK inside Memberships
- **Groups** - This will identify the groups in which direct messages are sent
    - **Group_id**: int, auto_increment, unique, not NULL, PK
    - **Group_name**: string, not NULL
    - **Relationship**: a 1:M between Groups and Messages with Group_id as a FK inside Messages
- **Messages** - This will identify the messages in a group, by a specific account
    - **Message_id**: int, auto_increment, unique, not NULL, PK
    - **Account_id**: int, auto_increment, unique, not NULL, FK
    - **Group_id**: int, auto_increment, unique, not NULL, FK
    - **Message**: string, not NULL
    - **Timestamp**: datetime, not NULL
- **Friendships -** This will show if a user is friends with another user
    - **From_Account_id -** int, auto_increment, unique, not NULL, FK, PK
    - **To_Account_id -** int, auto_increment, unique, not NULL, FK, PK

Entity-Relationship Diagram



Database Schema

Account

Account_id: int(11), auto_increment, primary key
Email: varchar(254)
Username: varchar(15)

Posts

*Post_id***:** int(11), auto_increment, primary key
Text: varchar(280)
Account_id: int(11), foreign key to Accounts.Account_id
Timestamp: datetime

Comments

Comment_id: int(11), auto_increment, primary key
Text: varchar(280)
Account_id: int(11), foreign key to Accounts.Account_id
Post_id: int(11), foreign key to Posts.Post_id
Timestamp: datetime

Messages

Message_id: int(11), auto_increment, primary key
Group_id: int(11), foreign key to Groups.Group_id
Account_id: int(11), foreign key to Accounts.Account_id
Message: varchar(255)
Timestamp: datetime

Memberships

Account_id: int(11), foreign key to Accounts.Account_id, primary key
Group_id: int(11), foreign key to Groups.Group_id, primary key

Groups

Group_id: int(11), auto-increment, primary key
Group_name: var_char(64)

Friendships

From_Account_id: int(11), primary key, foreign key to Accounts.Account_id
To_Account_id: int(11), primary key, foreign key to Accounts.Account_id

Database User Interface

Accounts list

Home Accounts Friendships Posts Comments Memberships Groups Messages

Account_id	Username	Email	
1	footballguy12	jeffk@hello.com	
2	albthatsme	aliceb@hello.com	
3	roberth77	bobh@hello.com	
4	newuser1234	useremail@hello.com	
	Username	Email	Add

Account_id
Username
Email
Search Update Delete

Image 1: All CRUD functionalities for Accounts table

Friendships list

Home Accounts Friendships Posts Comments Memberships Groups Messages

From_Account_id	To_Account_id	
1	2	
1	3	
2	3	
3	1	
4	1	
From_Account_id	To_Account_id	Add

From_Account_id
To_Account_id
Search Delete

Image 2: CREATE/READ/DELETE functionalities for Friendships table, no UPDATE by design

Posts list

Home Accounts Friendships Posts Comments Memberships Groups Messages

Post_id	Text	Account_id	
1	This is my first post!	1	
2	Hello World!	2	
3	Welcome!	3	
4	Hello everyone!	4	
5	New Test	3	
7	test	2	
	Text	Account_id	Add

Post_id
Text
Account_id
Search Update Delete

Image 3: All CRUD functionalities for Posts table

Comments list

Home Accounts Friendships Posts Comments Memberships Groups Messages

Comment_id	Text	Account_id	Post_id	
1	Nice post!	2	3	
2	Yeah I agree.	1	2	
3	lol	4	1	
4	yeah true I think	4	3	
5	what do you mean?	3	4	
6	love this.	2	2	
7	test	2	1	
	Text	Account_id	Post_id	Add

Comment_id
Text
Account_id
Post_id
Search Update Delete

Image 4: All CRUD functionalities for Comments table

Memberships list

Home Accounts Friendships Posts Comments Memberships Groups Messages

Account_id	Group_id	
3	2	
4	2	
Account_id	Group_id	Add

Account_id
Group_id
Search Delete

Image 5: CREATE/READ/DELETE functionalities for Memberships table, no UPDATE by design

Groups list

Home Accounts Friendships Posts Comments Memberships Groups Messages

Group_id	Group_name	
2	Group 2	
	Group_name	Add

Group_id
Group_name
Search Update Delete

Image 6: All CRUD functionalities for Groups table

Messages list

Home Accounts Friendships Posts Comments Memberships Groups Messages

Message_id	Account_id	Group_id	Message	
5	3	2	How are you doing?	
6	4	2	Great!	
8	1	2	Terrible	
	Account_id	Group_id	Message	Add

Message_id
Account_id
Group_id
Message
Search Update Delete

Image 7: All CRUD functionalities for Messages table