Received Feedback

I noticed that in your ERD, there are two lines that don't have the cardinality marked on both sides. Even when a relationship is 1:1 or 1, it's important to indicate the cardinality on both ends of the line to avoid confusion.

- Madelyn Lazar

In your outline, I would suggest mentioning that you'll need an intersection table to manage your many-to-many (M) relationships. For example:

- For the **User-Followers** M relationship, you require an intersection table that contains the foreign keys userID and followersID to link the two entities.
- For the **User-Following** M relationship, you need an intersection table that holds userID and followingID as foreign keys.

- Madelyn Lazar

The 1:M and M:M relationships appear to be correctly formulated in the outline - the use of foreign keys looks correct, though it would be useful to implement intersection tables for the M:M relationships. The ERD presents a logical view of the database, and it appears to be consistent with the outline; however, the User-ProfileEngagement and the User-PostEngagement relationship lines are missing the cardinality on the User side, so be sure to fix that.

a) There appears to be consistency in the naming between overview, and entities and attributes.

b) Most entities are not plural, with Followers being an exception. Entity names like Following, PostEngagement, and ProfileEngagment are probably difficult to pluralize, so you might want to adjust the names there if you think anything could work better. Some attributes are plural, but could reasonably be made singular (for example, you could change *totalViews* to something like *totalViewCount*).

c) The capitalization for naming is consistent.

- Indika Seagoe

Project Step 2 Draft
CS_340

Group Name: Team 86
Group Members: Giovanita Bell and Kevin Lee
Database Title: BeaverGram Statistics

Overview

BeaverGram is the next popular social media platform started by and created for Oregon State University students. The goal of this database is to track a user's account and to determine the user's engagement with other users/followers. BeaverGram is designed to have users post images/text/videos that other users can interact with using likes/comments. Users can follow other users to view posts from each other.

This database for this platform is focused around finding the average audience engagement statistics, by tracking a user's audience engagement.
The database is meant to track how frequently an audience engages each individual post and the user's profile. This allows a user to track any fluctuations in popularity.

Example:
A user on BeaverGram has the following data from their profile that is meant to be entered into the database:

- 160,000 followers
- 200 following
- 900,000 likes
- 2.6 M views
- 6 M profile views

With this data, the database is able to find the following:

- Average, max, and min number of likes per post
- Average, max, and min number of comments per post
- Average, max, and min number of gained followers per month

- Average, max, and min number of lost followers per month
- Average, max, and min number of profile views
- Ratio of followers to following
- Comprehensive total of likes and comments for a user
- Rate of change and trends of user's profile engagement from followers
    - Will be able to determine if comments, views, and likes have increased or decreased compared to previous month
    - Will be able to identify the most active months within a year

Database Outline

Entities and Attributes

- User - Retains the user's information
    - userID: int, auto_increment, unique, not NULL, PK
    - dateCreated: date, not NULL
    - email: varchar, not NULL

- Followers - Retains the number of followers the user has, and helps identify the frequency of followers gained or lost
    - followersID: int, auto_increment, unique not NULL, PK
    - dateFollowedChange: date, not NULL
    - isGainedFollower: tinyint(1)
    - totalFollowers: int, auto_increment, not NULL
    - dateFollowed: date, not NULL
    - Relationships:
        - M:M - Followers(PK) to User(FK)

- Following - Retains the number of other accounts followed by the user
    - followingID: int, auto_increment, unique not NULL, PK
    - totalFollowedAcc: int, auto_increment, not NULL
    - dateFollowing: date, not NULL
    - Relationships:
        - M:M - Following(PK) to User(FK)

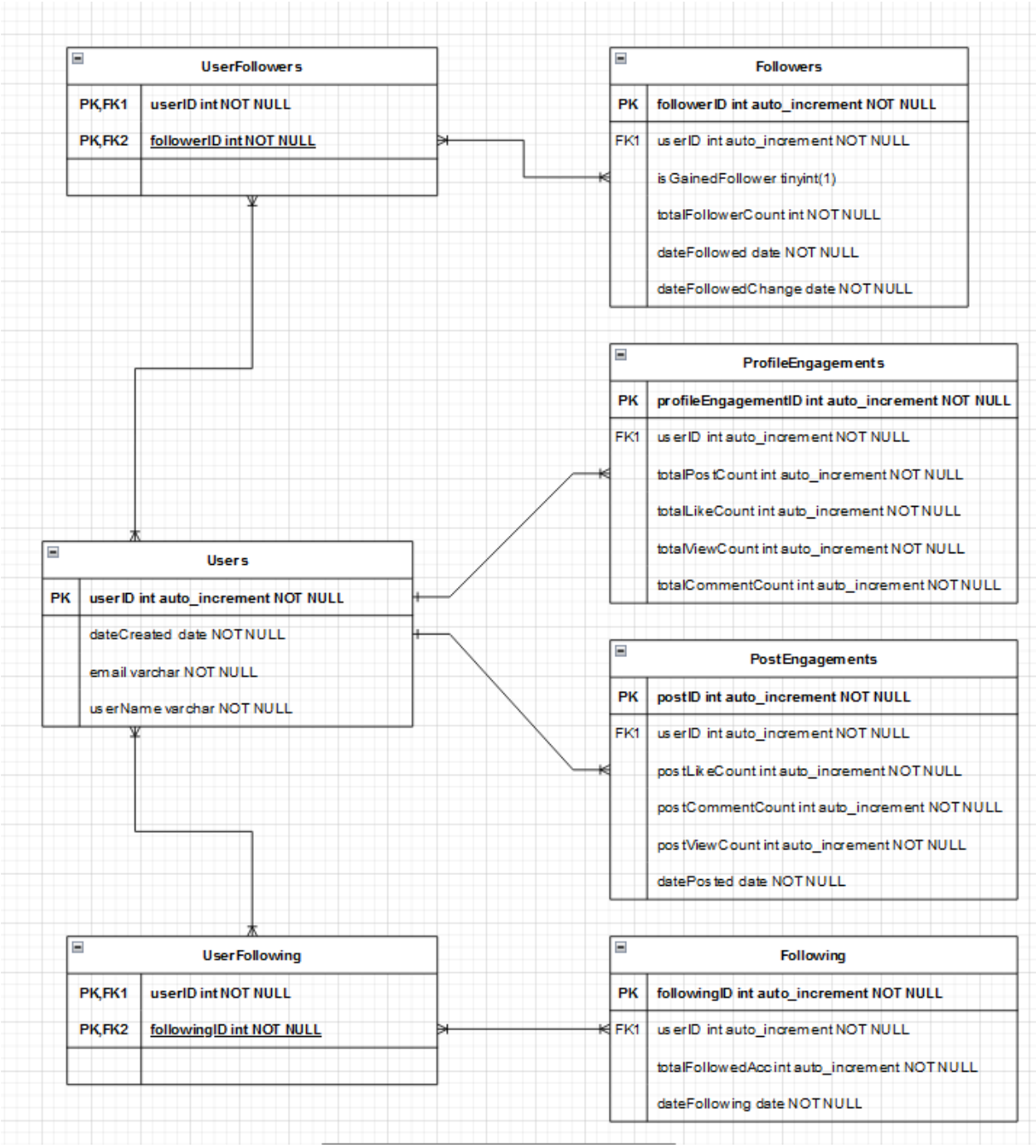
- Profile Engagement - Tracks the average and total engagement of a user profile, by combining total likes and comments for the entire page
    - profileEngagementID: int, auto_increment, unique, not NULL, PK
    - totalPosts: int, auto_increment, not NULL
    - totalLikes: int, auto_increment, not NULL
    - totalViews: int, auto_increment, not NULL
    - totalComments: int, auto_increment, not NULL
    - Relationships:
        - 1:M - Profile Engagement (PK) to User (FK)

- Post Engagement - Tracks the average and total engagement for individual posts. This enters the number of likes for each post, as well as the average number of likes for all posts
    - postID: int, auto_increment, unique, not NULL, PK
    - totalPostLikes: int, auto_increment, not NULL
    - totalPostComments: int, auto_increment, not NULL
    - totalPostViews: int, auto_increment, not NULL
    - datePosted: date, not NULL
    - Relationships:
        - 1:M User (PK) can have multiple posts

Intersection Tables

The following entity-relationships will have intersection tables:

- User-Following:
    - Intersection table will have userID and followingID as foreign keys
- User-Followers:
    - Intersection table will have userID and followersID as foreign keys



Example Data

```
INSERT into Users (userName, dateCreated, email)
VALUES
('Benny Beaver', '2020-01-01', 'bennybeaver@osu.com'),
('Moo Deng', '2020-01-01', 'moodeng@osu.com'),
('Pesto Penguin', '2020-01-01', 'pestopenguin@osu.com');
```

```sql
INSERT into Following (userId, totalFollowing, dateStartFollowing)
VALUES
(1, 1, '2020-01-01'),
(2, 2, '2020-01-01'),
(3, 0, '2020-01-01');

INSERT into Followers (userId, totalFollower, dateFollowed)
VALUES
(1, 1, '2020-01-01'),
(2, 1, '2020-01-02'),
(3, 1, '2020-01-01');

INSERT into ProfileEngagements(totalPost, totalProfileLikes, totalView,
totalComment)
VALUES
(10, 100, 1000, 5),
(1, 10, 100, 1),
(0, 0, 0, 0);

INSERT into PostEngagements(totalPostLike, totalPostComment, totalPostView,
datePosted)
VALUES
(5, 1, 10, '2020-01-02'),
(10, 0, 100, '2020-01-02'),
(25, 10, 20, '2020-01-02');
```