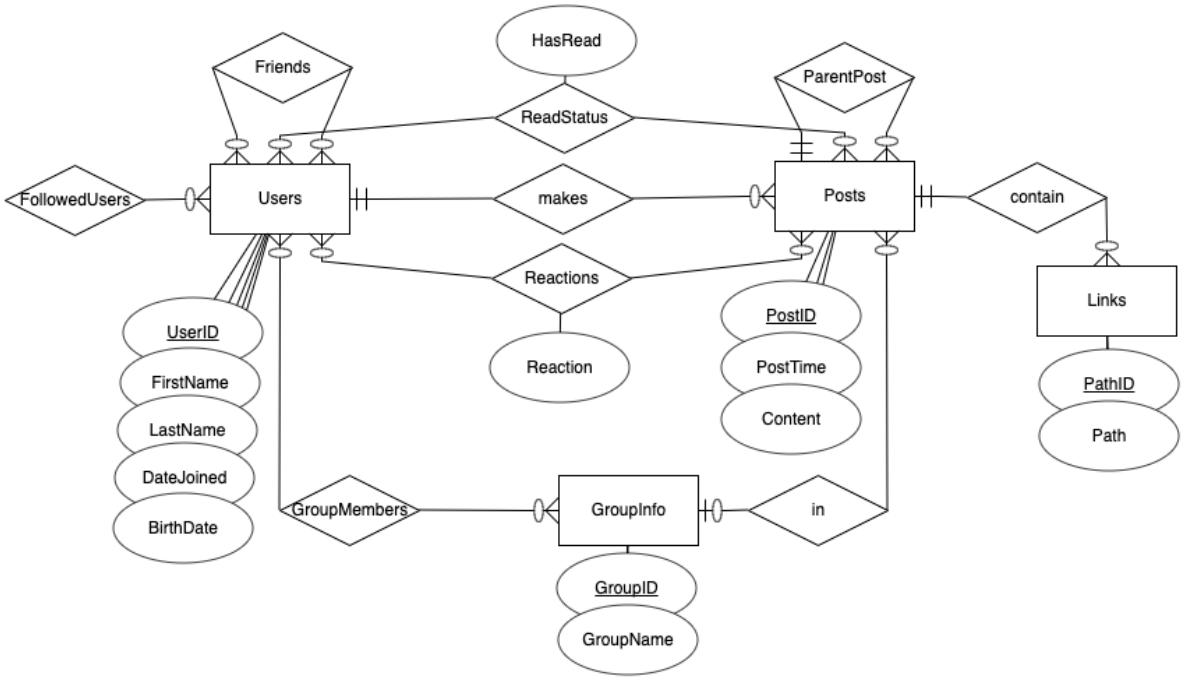
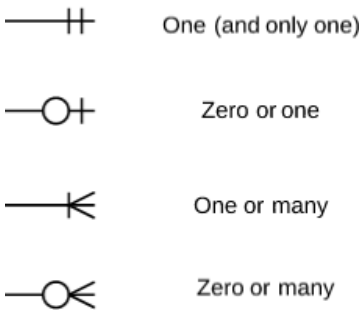
ER Model Design Writeup

Group Members:
Peter Dye | pjadye | 20678160
Zach Walford | zwalford | 20679930



This is the entity relationship model for our simple social network, “BudgetBook”.

Here is a quick legend for the ER Diagram just for convenient reference:



The main entities in our model are Users, Posts, and GroupInfo. There is another entity, Links, that has a relationship only with Posts.

Let’s start by focusing on the Users entity. This entity has 5 attributes. First name and last name are treated as a composite attribute of Name, and the others are all simple attributes. UserID is

the unique ID or “username” for that user. As you can see there are quite a few relationships involving the Users entity. The first one, FollowedUsers, is a many to many relationship between Users and Users; any user can follow any number of other users. On the diagram it only shows one connection to the Users entity but this is just a technical limitation of the website we used to draw these diagrams. Similarly to FollowedUsers, Friends is another many to many relationship between Users and Users. We choose to treat following and friending as two separate features in our social network, and that is why we have two separate relationships. When a user friends another use, they automatically follow each other but the option exists to unfollow and refollow later (for when your friend goes on a trip to Europe and won’t stop posting updates of their vacation). Both users must be friends, but one user can follow another user that does not follow them back.

Now looking between Users and Posts, there are three relationships between the two entities that they both participate in. This is because Users can make posts, read posts from other users, and react to posts from themselves or other users. These are three distinct features of the social network and can all be done independently, which is why three separate relationships are necessary. Reactions is a many to many relationship because any user can upvote or downvote any number of posts or comments including none, and the Reaction attribute for the Reactions relationship records whether the user upvoted or downvoted. The ReadStatus relationship is similar, but it is not an action that a user takes. When a user posts something, the social network will create unread relationships between posts, and users that follow or are in the same group as the user that made the post. This allows users to easily view posts and comments that they have not seen before. The final Makes relationship is what allows users to actually make posts. A user can make any number of posts including none but a post must have one and only one user associated with it.

Focusing on the Posts entity now, there are some attributes related to a post that are all simple attributes. The relationship ParentPost is needed in out social network because we choose to treat comments as the same entity as top level posts. This is because they contain all the same information. Any post, including comments, can have any number of comments associated with it, but can only be under one parent post or parent comment. Posts can optionally contain zero or more links that are associated with a post, but a specific instance of a link can only be in one post (note this does not stop more than one post from posting the same link, but a new Link entity will be created for it).

The only remaining entity to discuss is the GroupInfo entity. This is what defines a groups existence or not. There are two attributes for a group, both simple attributes. There are two relations involving this entity. The first one is GroupMembers which also involves Users. This relationship is required so that users can be part of a group, view the posts in that group, and have unread posts from that group listed for them to read. Users can be in zero or more groups and groups can have zero or more users. The second relationship is the in relationship that also involves Posts. This relationship allows posts to be organized in groups, which is needed for the user functionality about posts in groups previously mentioned. Posts can be in zero or one group, and groups can have zero or more posts.

For your convenience, we have also included the diagram of how this translates to a relational schema.