ER Diagram – Social Network

A close up of a map

Description automatically generated

As can be seen from the ER diagram below, we have “post”, “topic” “NetworkUser”, “SocialGroup” as four entity sets, and “PostBelongToTopics”, “PostsRespondToPost”, “UsersOwnPosts”, “UsersFollowTopics”, “UsersFollowsUsers” as five relationship sets.

The following relationship holds for each entity set respectively.

NetworkUser(uID -> name, DOB, lastLogin) -- primary key: uID

Here we choose “uID” as the primary key of the relation, which is an auto-incremented value for each unique entry, “name” field is a string value that represents the name of each user, “DOB” stands for Date of Birth, which is used show the associated birthday for each user. “lastLogin” is a time stamp value that indicates that last time the user logs in. we use this field to show the new posts since the last time user logs out.

post(pID -> content, ts, thumbNum) -- primary key: pID

Here we have “pID” as the primary key of the entity, it is also an auto-incremented value that is unique for each entry. We also have a varchar field named “content” that shows the content of each post. “ts” field shows the time stamp that this post was released, and each post has a value called “thumbNum”, which is used to show the sum of number of thumbs up that each post has. Here we underline that thumbNum is the total number of thumb up minus the total number of thumb down. We use this field to indicate the popularity of the post.

topic(tID -> name) -- primary key: tID

Here we have “tID” as the primary key of the topic relation. tID stands for topic ID, which is also a unique number that is unique to each entry. Here, we associate each entry with a “name” attribute, which shows the name for each topic. Here we only associate “name” field in the topic relation, which we think is enough to represent topic relation

SocialGroup(gID -> name) -- primary key: gID

Here we have “gID” as the primary key of the group relation. gID stands for group ID, which is also a unique number that is unique to each entry. Here, we associate each entry with a “name” attribute, which shows the name for each topic. Here we only associate “name” field in the topic relation, which we think is enough to represent group relation

Each relationship sets above is constructed as follows:

PostBelongToTopics(postID, topicID):

postID and topicID are both primary keys for the entity

postID is the foreign key of pID in the post relation

topicID is the foreign key of tID in the topic relation

each post can belong to one topic, but one topic can have multiple post, so this is a one to many relationship

UsersBelongToGroups(userID, groupID):

userID and groupID are both primary keys for the entity

userID is the foreign key of uID in the NetworkUser relation

groupID is the foreign key of gID in the topic relation

sine each user can have multiple social groups, and each group can have many users, we define this as many to many relationship

UsersFollowTopics(userID, topicID):

userID and topicID are both primary keys for the entity

userID is the foreign key of uID in the NetworkUser relation

topicID is the foreign key of tID in the topic relation

sine each user can follow multiple topics, and each topic can have many users following it, we define this as many to many relationship

UsersOwnPosts(userID, postID):

userID and postID are both primary keys for the entity

userID is the foreign key of uID in the NetworkUser relation

postID is the foreign key of pID in the post relation

sine each user can own many posts, but each post can have one author(user), we define this as one to many relationship

PostsRespondToPosts(responderPostID, respondingPostID):

responderPostID and respondingPostID are both primary keys for the entity

responderPostID is the foreign key of pID in the post relation

respondingPostID is the foreign key of pID in the post relation

since one main post can have many sub posts, but one sub post can have only one main post, we define this as one to many relationship