SOCIAL MESSAGING APPLICATION

PHASE II – Implementation of Read Operations

Team Members:
Sujeily P. Fonseca González
Samuel G. González Ortiz
Lianne Sánchez Rodríguez

Professor Manuel Rodríguez
Database Management Systems
ICOM 5016, Section 096
Tuesday, May 1, 2018

TABLE OF CONTENT

SOCIAL MESSAGING APPLICATION
PHASE I – CONCEPT DESIGN

OBJECTIVES

1. Understand the design, implementation and use of an application backed by a database system.
2. Understand the use of the E-R model for database application design.
3. Gain experience by implementing applications using layers of increasing complexity and fairly complex data structures.
4. Gain further experience with Web programming concepts including REST and HTTP.

GENERAL OVERVIEW

The project presented in this document, titled Social Messaging Application, consists in the design and development of an application used for messaging in a social context. It will callow the following operations:

- ☐ Register a user with the application
- ☐ Create a chat group with a given name
- ☐ Add a user to your contacts list based on name, last name, and either phone or email address
- ☐ Add a contact to a chat group
- ☐ Remove a user from a chat group
- ☐ Remove a user from the contacts list
- ☐ Remove a chat group (only the owner)
- ☐ Post a message to a chat group
- ☐ Read the messages from a chat group
- ☐ Like a message posted on a chat group
- ☐ Dislike a message posted on a chat group

- ☐ Reply to a message posted on a chat group
- ☐ Post an image to a group (optionally)
- ☐ Post a video to a group (optionally)

Furthermore, the application will contain a web-based dashboard to provide statistics, such as the mentioned below:

- ☐ Trending topics via #hashtags
- ☐ Number of message per day
- ☐ Number of replies per day
- ☐ Number of likes
- ☐ Number of dislikes
- ☐ Active users

The data in the application is managed by a relational database system and exposed to client applications through a REST API.

E-R MODEL

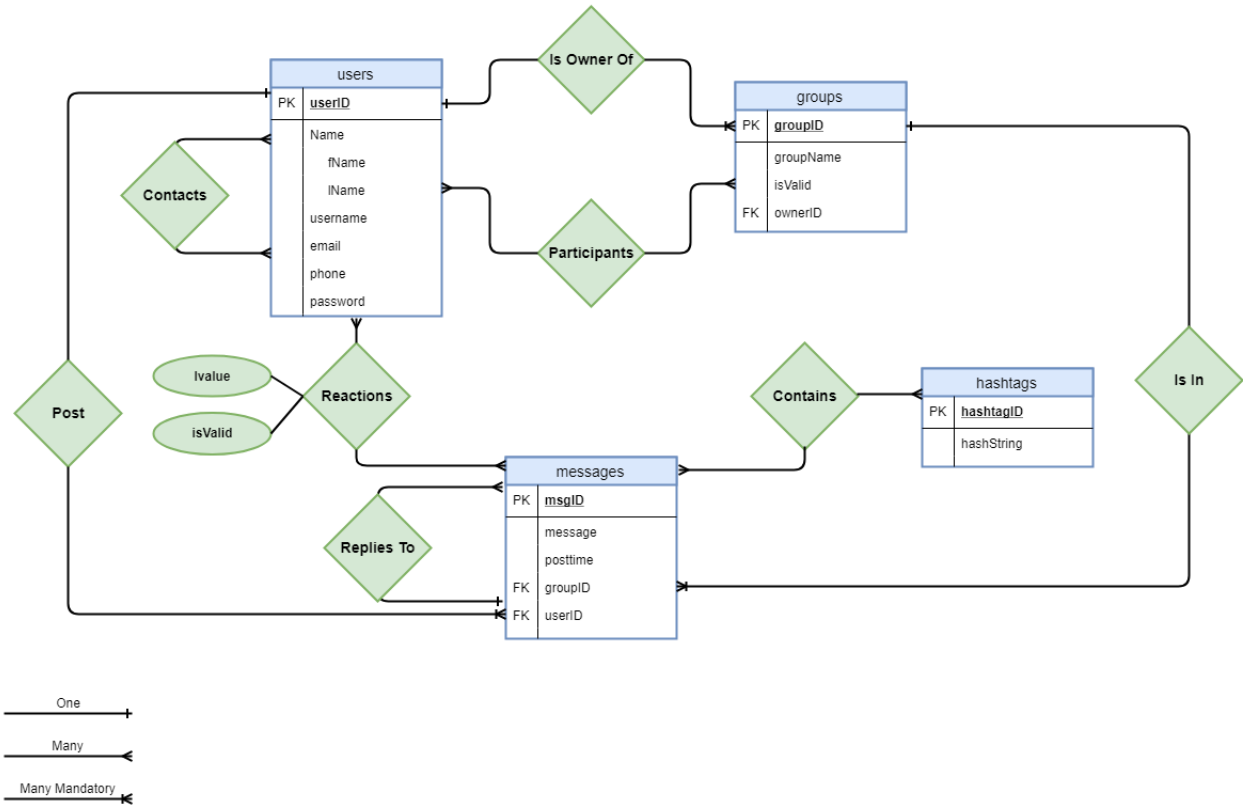

Figure 1: *E-R Diagram*

ENTITIES AND RELATIONSHIPS

The Entity Relationship Diagram shows the relationships of entity sets stored in a database. Relationships can be described as follows:

NAME	TYPE	DESCRIPTION
Users	Entity	The "users" entity is the table that has as attributes the primary key which is the user ID, first name, last name, phone, password and username from a user. The attributes first name, last name, phone and username are of type varchar with maximum length of 50 characters. The attribute password is of type chkpass, which does an encryption before the data is stored in the database system. In general, this entity saves the users and the information of all users in the system and/or message application.
Contacts	Relation	The "contacts" relationship maps to a table and is a role that users play with each other where an user is contact of another. Each user has a list of users that are his/her contacts. The relationship of contacts has three attributes: contact ID which references an user ID, contact of ID which refers to the contact of the previously mentioned user, and as primary key the combination of both, making it unique.
Is Owner Of	Relation	The relationship "is owner of" maps as foreign and primary keys relations between the entity of users and groups. This relationship is one to many with total participation. This relationship represents that an user could be owner of different groups within the application, and a group cannot exist without an owner.
Participants	Relation	The relationship "participants" maps groups and users entities. It consists in a many to many relationships. This is because users can participate in multiple groups, but at the same time, a group can belong to multiple users. The mentioned relationship has the following attributes: reference to userID, reference to groupID, and a combination of both mentioned fields for the primary key.
Groups	Entity	The table "groups" represents a group chat in which messages are sent to make conversations. Each group has a groupID as its primary key, which is used to identify a specific group in the database. Each group also contains a field for its name which is of type varchar. Furthermore, since deleting records is not recommended, groups have an "isValid" integer field which will have a value of 1, if the group is existent, or 0 to represent it as non-existent, without the need to delete it from the database. Finally, since all groups must have an owner, each group has an ownerID integer field which stores the userID of the owner for a given group.

Is In	Relation	The "is in" relationship creates an union between the entity messages and entity groups. The relationship is one to many with total participation on the many side, the one side is on groups and the many ide is on the entity messages. The way is mapped is that in the entity of messages you will find a foreign key that will reference to the group id of the entity groups.
Messages	Entity	Messages will be posted by users participating in a group. The table "messages" stores a record for each message sent on every group. It has an msgID which is used as primary key and allows for a specific message to be easily accessed from the database. It also has a varchar field for the actual message that will be written and sent by the user. In addition, it contains a timestamp field named "postTIme" which stores the moment in time in which the message was sent and inserted in the database. This allows the message to be presented on the front-end in the actual order they were sent. Since a message must belong to a group, every message on the database holds a reference to the group to which it belongs, using the integer field "groupID". Lastly, a message must also belong to an user, therefore, using the integer field "userID", each message can hold a reference to the user who sent it.
Replies To	Relation	The "repliesTo" table is used to store relations between messages and their replies. The integer field "msgID" stores a reference to a message in the database (let's call it 'A'), and the integer field "replyID" stores a reference to a message that replies to 'A'.
Post	Relation	The "post" relationship is one to many relationships between users and messages entities, having total participation in the messages side. This is because a message cannot exist without a user to write it. This type of relationship does not require to be mapped in a table. Furthermore, it consists in the following: user posts multiple messages in a group chat.
Reactions	Relation	The "reactions" relationship is used to store the reactions of users to messages. It contains multiple attributes, such as Ivalue, isValid, reference to msgID, reference to userID, and the primary key, which is a combination of the last two mentioned attributes. The Ivalue attribute is '1'when an user likes the messages, '0' otherwise. The isValid attributes is '1', which is true, until the user wants to remove the provided reaction. The described relationship is many to many. This is because a user can provide reactions to many messages, and a message can have many reactions from users.
Contains	Relation	The "contains" relationship has the following attributes: reference to msgID, reference to hashtagID, and a combination of both for the primary key. It consists in many to many relationships. This is because messages can have multiple hashtags, and a hashtag can be present in multiple messages.

Hashtags	Entity	The "hashtags" tables is used to store all hashtags contained within the messages stored in the database, including duplicates. It contains a "hashtagID" integer field used to access the hashtag in the database. Also, it contains a varchar field called "hashString" which stores the string of character that form the specific hashtag.

MAPPING FROM E-R TO SQL TABLES

create table users(userID serial primary key, fName varchar(50), lname varchar(50), email varchar(50), phone char(10), password chkpass);

create table contacts(contactID integer references users(userID), contactofID integer references users(userID), primary key (userID, userID));

create table groups(groupID serial primary key, groupName varchar(100), isValid bit ownerID integer references(userID);

create table participants(ownerID integer references users(userID), groupID integer references groups(groupID), primary key(ownerID, groupID));

create table messages(msgID serial primary key, message varchar(350), userID integer references users(userID), groupID integer references groups(groupID), postTime timestamp);

create table repliesTo(msgID integer references messages(msgID), replyID integer references messages(msgID), primary key (msgID, replyID));

create table reactions(reactionID serial primary key, lvalue bit, isValid bit, userID integer references users(userID), msgID integer references messages(msgID));

create table hashtags(hashtagID serial primary key, hashstring varchar(80));

create table contains(msgID integer references messages(msgID), hashtag integer references hashtags(hashtagID), primary key (msgID, hashtagID);