Osvaldo E. Aquino
Carlos Del Valle
Jesús Acosta
Matthew Mendoza

ICOM - 5016
Databases
April 12, 2021

JCOM-DB

Documentation of ER Mapping to Tables

For the implementation of the ER diagram to run in the code it needed to be made into tables, these tables would be distinguished with the attributes to make them connect with each other as well as keep the information that is needed for them to do so. Such tables are as follows:

- Users table:

```
CREATE TABLE users ( uid SERIAL NOT NULL CONSTRAINT uname_pkey PRIMARY KEY,
fname VARCHAR(30) NOT NULL, lname VARCHAR(30) NOT NULL,
uname VARCHAR(30) NOT NULL, pword VARCHAR(20) NOT NULL,
email VARCHAR(30) NOT NULL, phone BIGINT NOT NULL,uage integer,
is_Active BOOLEAN NOT NULL DEFAULT 'TRUE');
```

The users table has its own default primary key that changes dynamically in the database table so that it doesn't repeat itself every time a new user registers himself in the app. He will also be able to provide public information that will show to other people such as his first name (fname), a last name (lname), user name (uname), a password (pword), an email, a phone number (phone) and lastly his age (uage). All of these attributes except phone and uage consist of strings that can not be null, while phone and user age are integers. Additionally, the last attribute “is_Active” works as a boolean where if the user would like to deactivate his account from the database without deleting it, the account would therefore be set to false or true if the user wanted to reactivate the account.

- Messages table:

```
CREATE TABLE messages (
      mid SERIAL NOT NULL CONSTRAINT msg_pkey PRIMARY KEY,
      hasNotified BOOLEAN NOT NULL DEFAULT 'TRUE',
      is_Reply BOOLEAN NOT NULL DEFAULT 'FALSE',
      uid INTEGER CONSTRAINT uid REFERENCES users,
      date TIMESTAMP, msg VARCHAR(280) [] );
```

The messages table contains the “mid” which is the primary key each record is assigned when a new unique message is created. The table contains two boolean atomic attributes which are “hasNotified” (when TRUE creates a notification for the pertinent Users to receive) and “is_Reply” (when TRUE states that the message is a response to a previously created message). The “uid” attribute is a foreign key referencing the “uid” that functions as the primary key in the Users table which represents the user who posted the message. Time and date of when the message was posted is represented in “date”. The message itself is the string “msg” which can be as long as 280 characters.

- Replies table:

```
CREATE TABLE replies (replyid INTEGER NOT NULL CONSTRAINT replyid REFERENCES
messages,repliesToid INTEGER NOT NULL CONSTRAINT repliesToid REFERENCES
messages(mid),
CONSTRAINT rid PRIMARY KEY (replyid, repliesToid));
```

The replies table relates the response of a new message to an existing message. First you have the “replyid” which works as an integer, this attribute will look for reference in the message table since a reply is just a message directed to another message. After acquiring the id, it will proceed to look for the id of the message it’s trying to reply to. Both of these variables will be saved in a primary key called “rid”.

- Likes table:

```
CREATE TABLE likes (
   uid INTEGER NOT NULL CONSTRAINT uid REFERENCES users,
   mid INTEGER NOT NULL CONSTRAINT mid REFERENCES messages,
    CONSTRAINT lid PRIMARY KEY (uid, mid) );
```

The likes table is pretty simple in terms of the implementation that was done, when you as a user like a message, what is being done in the table is that the user id will be looked upon in the actual users table, it essentially looks for the user id in the table, takes it for reference and groups it together with the message that was liked.

- Unlikes table:

```
CREATE TABLE unlikes (
      uid INTEGER NOT NULL CONSTRAINT uid REFERENCES users,
      mid INTEGER NOT NULL CONSTRAINT mid REFERENCES messages,
      CONSTRAINT ulid PRIMARY KEY (uid, mid));
```

Based on the relation from the E-R diagram the unlike table groups the foreign keys “uid” and “mid”. These represent the user that unlikes the message and the unliked message respectively. Contains the primary key for the record “ulid”.

- Shares table:

```
CREATE TABLE shares(
      uid INTEGER NOT NULL CONSTRAINT uid REFERENCES users,
      mid INTEGER NOT NULL CONSTRAINT mid REFERENCES messages,
      CONSTRAINT sid PRIMARY KEY (uid, mid));
```

Records when a user represented with the foreign key “uid” shares the message “mid”. The primary key “sid” identifies this relationship.

- Follows table:

```
create table follows (uid1 INTEGER NOT NULL CONSTRAINT uid1 REFERENCES users,
uid2 INTEGER NOT NULL CONSTRAINT uid2 REFERENCES users,
CONSTRAINT fid PRIMARY KEY (uid1, uid2));
```

The “follows” table uses the same type of implementation as the like table since it references from the users table containing the user id for both, the person that wants to follow, and the person that was followed. All of this is put together in a primary key called “fid” which holds both user ids.

- Blocked table:

```
create table Blocked (uid1 INTEGER NOT NULL CONSTRAINT uid1 REFERENCES users,
uid2 INTEGER NOT NULL CONSTRAINT uid2 REFERENCES users,
CONSTRAINT bid PRIMARY KEY (uid1, uid2));
```

The blocked table is essentially similar to the “follows” table, it uses the same user ids 1 (the one blocking) and 2 (the one being blocked) referenced from the users table and it holds them in a primary key called “bid”. This is later used for the string that is shown in the url where you get notified that you as a user blocked said person.