Contents

[Application Description 1](#_Toc88058999)

[Entity Relationship Diagram 1](#_Toc88059000)

[Mapping to Relational Schema 1](#_Toc88059001)

[Functional Dependency Diagrams 1](#_Toc88059002)

[Explanation and SQL Code for Creating the database Tables 1](#_Toc88059003)

[Explanation and SQL Code for Altering tables 1](#_Toc88059004)

[Explanation and SQL Code for Triggering operations 1](#_Toc88059005)

[Explanation and SQL Code for Creating Views 1](#_Toc88059006)

[Explanation and SQL Code for Populating the Tables. 1](#_Toc88059007)

[Explanation and SQL Code for retrieving information from the database 1](#_Toc88059008)

[Explanation and SQL Code for Triggers 1](#_Toc88059009)

[Explanation and SQL Code for Security 2](#_Toc88059010)

[Explanation of Additional SQL Features of your choice 2](#_Toc88059011)

Gwen McArdle  18322248

Information management Project

# Application Description

The assignment called for us to develop an ER Model for information to be represented for an application of our choice, and to implement it as a MySQL database. I selected a messaging app as the basis of my project, as I felt that it had a combination of entities with interesting relations. This application would use a user’s phone number as an identifier. A user could then send messages to other users in a chat. The recipients could reply in the chat.

The entities that I identified in this application are User, Device, Chat and Message. The data that needs to be stored includes:

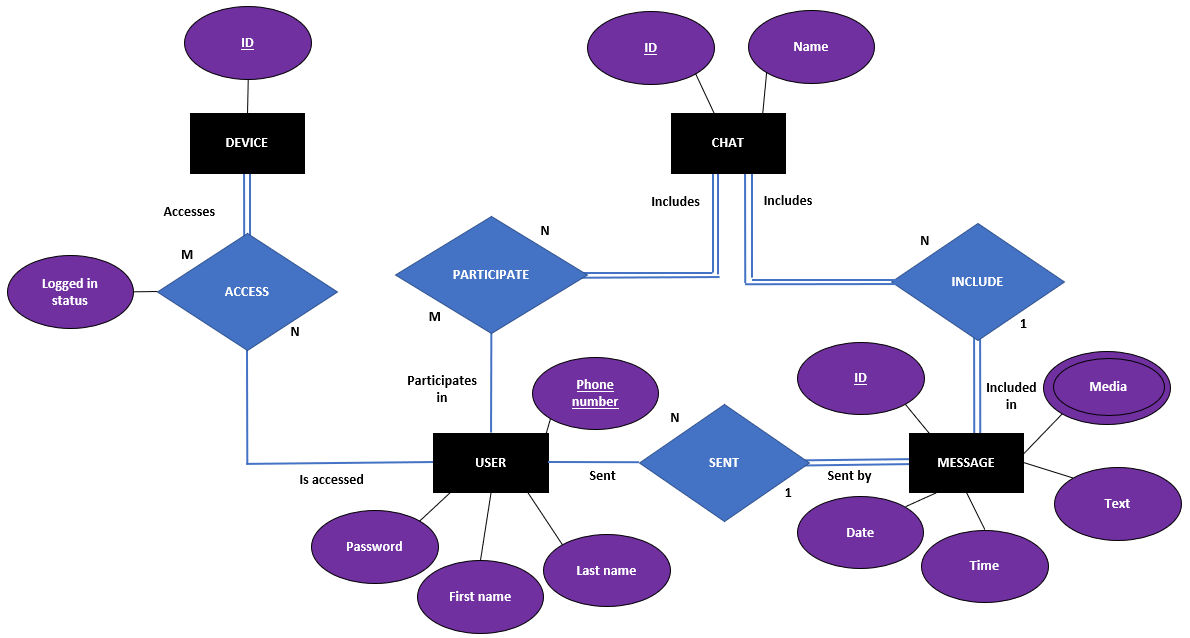
User – first name, last name, phone number and password.

Device – ID.

Chat – ID, name.

Message – ID, text, media, date and time. (Where media is any photos, videos or files that a user sends in a message.)

# Entity Relationship Diagram



In this model, I made the following assumptions:

* A user’s account can be connected to multiple devices, and multiple user’s can be connected to a device.
* A device should not be recorded in a database unless it is connected to a user.
* A user can participate in many chats, and a chat can have many participants.
* A chat cannot exist without any participants.
* A chat can have many messages, but a message can only be in one chat, duplicates will have different IDs.
* A message cannot exist if it is not in a chat, or does not have a sender.
* A user can only have one sender, but a user can send many

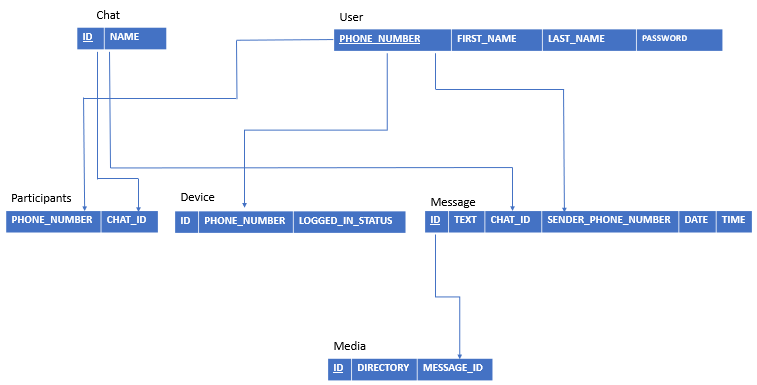
# Mapping to Relational Schema

Each entity must be represented in its own table, except for Device. This was unusual as there was only one data point that I wanted to store for each device; ID. Therefore, mapping the relation between Device and User was sufficient. I named this table Device.

The relation between chats and users is a N:M relation, requiring a separate table, which I named Participants.

The media attribute of Message is a multi-value attribute and therefore, required a separate table, named Media.

This maps to the below relational schema:

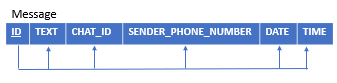


# Functional Dependency Diagrams

I identified the following functional dependencies:





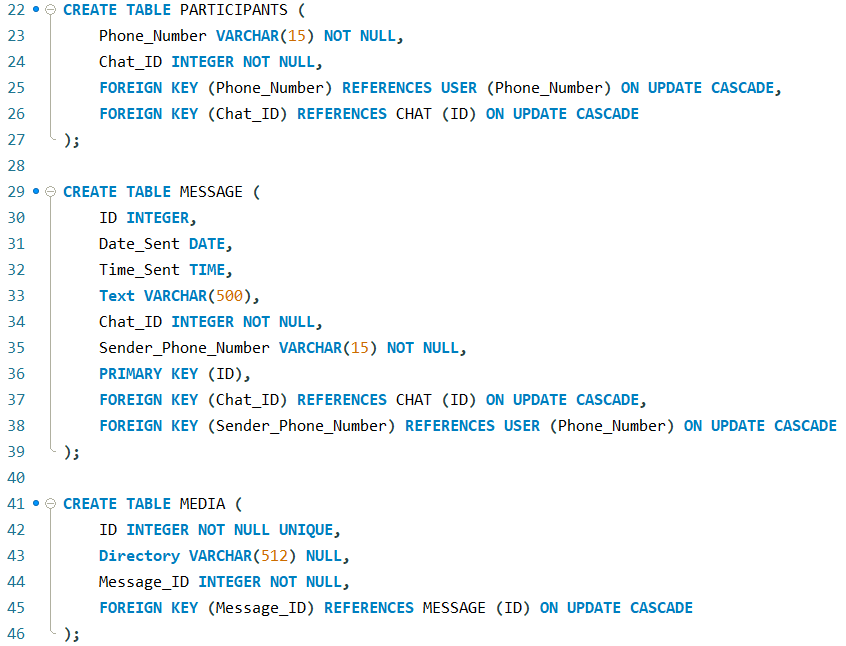
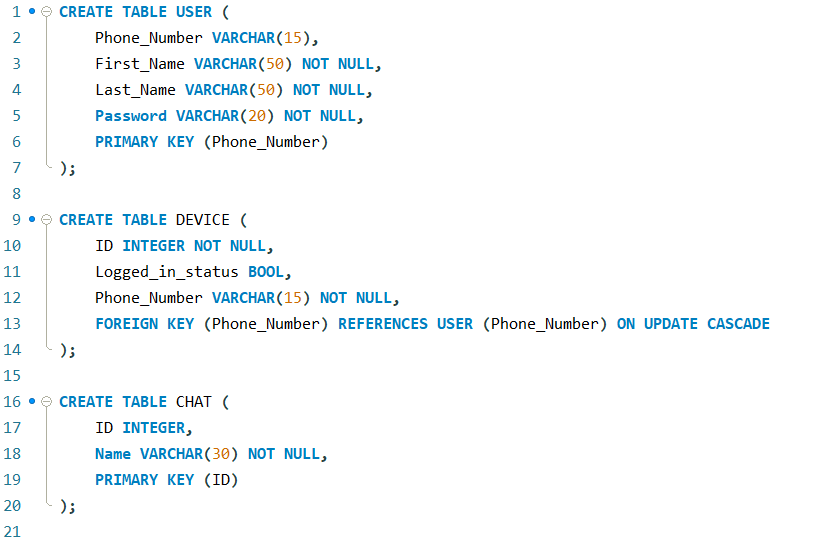




Logged\_in\_status is functionally dependent on the combination of ID and phone\_number.

Participants does not have any functional dependencies as its purpose is to link pairs of foreign keys.

# Explanation and SQL Code for Creating the Database Tables



This SQL script creates six tables; User, Device, Chat, Participants, Message and Media. I used the Relational Schema to establish the order the tables need to be created in. Device references User, Participants and Message reference User and Chat, and Media references Message. Due to these references, I established the order.

## User

**Phone\_Number** – Phone numbers are always unique; therefore, they are a suitable choice for the primary key. “PRIMARY KEY ()” implies “NOT NULL” and “UNIQUE” constraints. The maximum number of digits in a phone number is 15, therefore VARCHAR(15) is sufficient.

**First\_Name** and **Last\_Name** – I concluded that 50 characters would be sufficient for all names, and therefore chose VARCHAR(50) for both name attributes. Messaging applications regularly use users’ names in the interface, therefore these attributes cannot be null.

**Password** – I decided to constrain the passwords to 20 characters, in the interest of not wasting storage space. Every user needs a password so that attribute can’t be null.

## Device

**ID** – Each device needs an individual identifier, however this does not need to be unique in this table as the same device can have access to multiple accounts.

**Logged\_in\_status** – This boolean keeps track of whether or not a user is logged in on a particular device.

**Phone\_Number** – This value is a foreign key and it cannot be null as the purpose of this table is for storing pairs of devices and users. If Phone\_Number is updated in User, this means that the user has changed their phone number, which is their identifier, this value should be updated in Device so that Device can reference the correct User. Therefore, I added “ON UPDATE CASCADE”. If a Phone\_Number is deleted, that means the user’s account has been deleted, and it is no longer necessary to store what Devices are connected to the account. Therefore, I added “ON DELETE CASCADE”.

## Chat

**ID** – Each chat needs an ID to act as the primary key.

**Name** – Each chat must have a name, and therefore cannot be null. I decided to constrain the name to 30 characters.

## Participants

**Phone\_Number** and **Chat\_ID** - These values are foreign keys and cannot be null as the purpose of this table is for storing pairs of phone numbers and chat IDs. They do not need to be unique as a user can be a participant in multiple chats, and a chat has multiple participants. Updates to a user’s phone number should be reflected in this table, however there is no reason for a chat ID to be updated, so this is restricted. If a user or chat is deleted, they should be removed.

## Message

**ID** – Each message needs an ID to act as the primary key.

**Date\_Sent** and **Time\_Sent** – Stores the date/time the message was sent. I used the “DATE” and “TIME” datatypes for this. They do not have to be unique, but they cannot be null, because this would cause problems displaying the sequence of the messages in the chat.

**Text** – The text contained in the message. This is constrained to 500 characters, to avoid wasting space on short messages. This can be null as users can send media without text in a message.

**Sender\_Phone\_Number** and **Chat\_ID** - These values are foreign keys and cannot be null as a message must have a sender and belong to a chat in order to exist. They do not need to be unique as a user can send multiple messages, into multiple chats. Updates to a user’s phone number should be reflected in this table, however there is no reason for a chat ID to be updated, so this is restricted. If a user is deleted, the message should still exist in the chat but the sender should be null. If a chat is deleted, this should be reflected in the table.

## Media

**ID** – Each media file needs an ID to act as the primary key.

**Directory** – This stores the directory of the media file. The directory should correspond to a bucket the file is stored in. This cannot be null as that would cause application errors.

**Message\_ID** – This is a foreign key linking the media file to the message it was sent in. It cannot be null as files can only be added to the database by being sent in a message. There is no reason that message IDs should be updated, therefore updates are restricted. If the message is deleted, this should be reflected in the table.

# Explanation and SQL Code for Altering Tables

# Explanation and SQL Code for Triggering Operations

# Explanation and SQL Code for Creating Views

# Explanation and SQL Code for Populating the Tables.

# Explanation and SQL Code for Retrieving Information from the Database

# Explanation and SQL Code for Triggers

# Explanation and SQL Code for Security

# Explanation of Additional SQL Features of your choice