SQLChat

A Basic Database for a Messenger

Sezer Enis Birgili
No: 31033

Project Overview:

A simple messenger that allows basic text based communication between assigned users by exchanging messages in appropriately designated conversations with the option to send the messages with an attached file or files. Other users can see all messages with attached files in a conversation. Users may also send messages to each other by the use of contacts to start a conversation with the receiver of the message. Each conversation may include more than two users as participants. The users do not have to join any conversation to get started with the messenger.

Description:

Users Table:

The Users Table keeps track of every user’s account in the database. We contain every user's unique ID, email, password, name and contact list. The unique ID is a twenty character alphanumeric code; the name, email and password is set by the user at their own discretion however a limit of forty characters is enforced. This database helps us keep track of all registered users in our system as well as their associated contacts. Additionally, a user can see other users through their contact list which they can use to send messages to other users. Although, users do not need to send messages or participate in conversations to be registered. Therefore, the users table does not obey the participation constraint.

Contacts Table:

The contacts table records the relationships between users. Every record in this table is assigned with two user IDs, which are unique alphanumeric codes for users, one for the contactor and other for the contacted and a unique twenty character alphanumeric code as contact ID. Each of the user IDs act as a foreign key from the users table. Each user can be a contactor and a contacted for multiple people. Every contact must be associated with a user as contacts by definition, cannot be done without no one. If a user is updated or deleted contacts table either updates or deletes the row the user ID was assigned to.

Messages Table:

The messages table is the fundamental block of the messenger database, as it records all communication between users for each conversation although it does not keep track of who sent the message to whom. The sender and receiver table is responsible for recording message sender and receiver. The messages in the messages table has the attributes of a message ID as unique twenty character alphanumeric codes and the text content of the message. The text cannot be more than 255 characters long for it to fit in a single message. If applicable, a 255 character file attachment URL is also kept track of if the user sent files with the message.

Sender Table:

The sender table assigns every message an owner. The sender table includes user ID, message ID and sender ID as 20 character alphanumeric codes. The sender ID is the primary key while the user ID and the messages ID are foreign keys from users and messages table respectively. The users and messages table has a one-to-one relationship. However, users may not send messages as such don’t have to participate in sender table. If a user is updated or deleted sender table either updates or deletes the row the user ID was assigned to.

Receiver Table:

The receiver table assigns every message a destination user. The receiver table includes user ID, message ID and receiver ID as 20 character alphanumeric codes. The receiver ID is the primary key while the user ID and the messages ID are foreign keys from users and messages table respectively. The users and messages table has a one-to-one relationship. However, users may not receive messages as such don’t have to participate in receiver table. If a user is updated or deleted receiver table either updates or deletes the row the user ID was assigned to.

Conversations Table:

All sent messages are registered in conversations created by users. The registry for the conversations table includes information such as a conversation ID, the user ID of the conversation creator as alphanumeric codes; the title as a string.

Participates Table:

This block allows us to organize and facilitate user interactions and discussions in a controlled setting. The memberships of all users for specific conversations are documented in this table. Each record has a participant ID as the unique key, a user ID that allows the user to take part in the conversation and a conversation ID that identifies the specific conversation that the user wants to be a part of. Each field is a twenty character alphanumeric code and the user ID is a foreign key from the users table while the conversation ID is a foreign key from the conversations table. Users can either be part of multiple or one conversation or can be part of none at all. As such, users have no constraint in participating in conversations. Whereas, due to the nature of conversations, every conversation must obey the participation constraint as there cannot be a conversation with no member users in the database. However, each user can part of multiple conversations and conversations can be used by multiple people. If a conversation or a user is updated or deleted, participates table either updates or deletes the row where the user ID or the conversation ID was assigned to.

Exchanges Table:

The exchanges section links each message and conversation to each other making sure the path of the message is recorded with a date. The exchange contains exchange ID, conversation ID, message ID and date information for when the message was sent. The message ID and the conversation ID act as the foreign keys referencing the messages and conversations tables respectively. The same messages cannot be sent to multiple conversations, each message is a member to only one conversation. Messages cannot be sent directly to users, they can only be sent to conversations thus all messages must be member of a conversation. If a conversation or a message is updated or deleted, exchanges table either updates or deletes the row where the conversation ID or the message ID was assigned to.

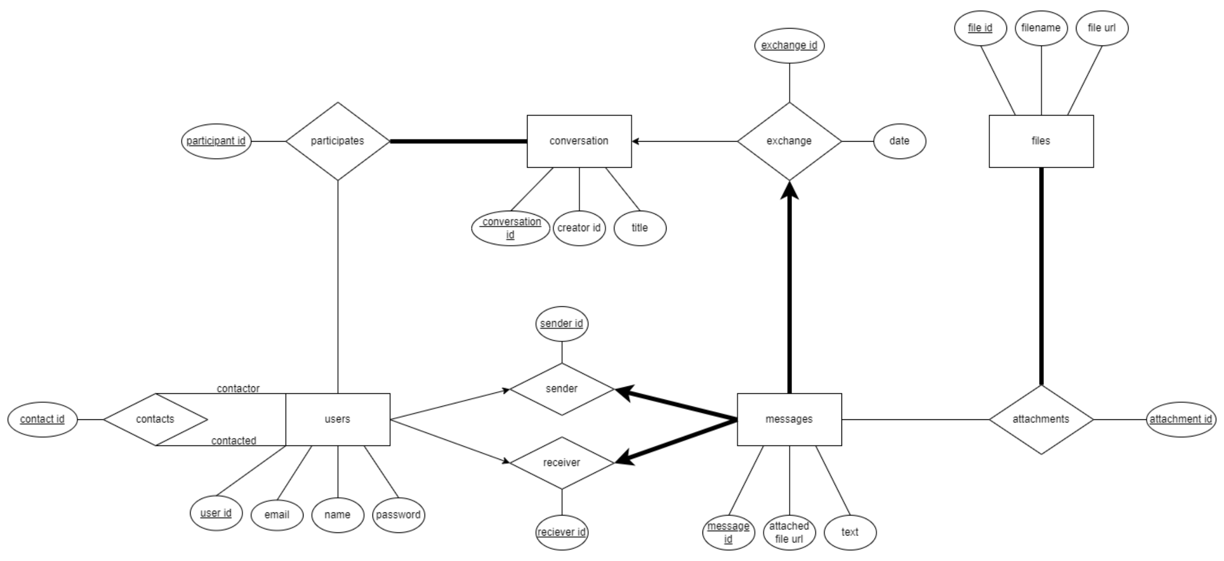
Attachments Table:

The attachments table notes the relationship of each file with the message it was sent with. The attachments table includes attachment ID as key, file ID and a message ID for every file. The message ID acts as the foreign key to messages table while file ID acts as the foreign key of the files table. Files cannot be uploaded separately into the database, every file must be attached to at least one message hence, the files table has a participant constraint. A message can carry multiple attached files however, a single file cannot be used by multiple messages. Each uploaded file can only be linked to one message. If a message is updated or deleted, attachments table either updates or deletes the row where the message ID was assigned to.

Files Table:

Some messages may contain attachments as images, documents or some other file types. When the message is sent, the uploaded message attachment is saved on the cloud along with the message albeit to different tables. The attachment is stored in the files table which keeps the twenty character alphanumeric file ID, a string for the filename, and a 255 character file URL as fields. In each conversation the users who are members of that conversation, can see the file attached to a message through the file URL.

ER Diagram:



MySQL Tables:

```
CREATE TABLE Users

(

        user_id CHAR(20),

        name CHAR(40),

        email CHAR(40),

        password CHAR(40),

        PRIMARY KEY (user_id),

        UNIQUE(email),

);
```

```
CREATE TABLE Contacts
(
	contact_id CHAR(20),
	user_id_1 CHAR(20),
	user_id_2 CHAR(20),
	PRIMARY KEY (contact_id),
	FOREIGN KEY (user_id_1) REFERENCES Users (user_id)
	ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (user_id_2) REFERENCES Users (user_id)
	ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Participates
(
	participant_id CHAR(20),
	user_id CHAR(20),
	conv_id CHAR(20),
	PRIMARY KEY (participant_id),
	FOREIGN KEY (user_id) REFERENCES Users (user_id)
	ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (conv_id) REFERENCES Conversations (conversation_id)
	ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Conversations
(
	creator_id CHAR(20),
	title CHAR(40),
	conversation_id CHAR(20),
            PRIMARY KEY (conversation_id)
);

CREATE TABLE  Exchanges
(
            exchange_id CHAR(20),
            conv_id CHAR(20),
            message_id CHAR(20),
            date LocalDateTime,
            PRIMARY KEY(exchange_id),
            FOREIGN KEY (conv_id) REFERENCES Conversations (conversation_id)
            ON DELETE CASCADE ON UPDATE CASCADE,
            FOREIGN KEY (message_id) REFERENCES Messages (message_id)
            ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Messsages
(
            message_id CHAR(20),
            text CHAR(255),
            file_url CHAR(255),
            PRIMARY KEY (message_id)
);

CREATE TABLE Sender
(
            sender_id CHAR(20);
            message_id CHAR(20),
            user_id CHAR(20),
            PRIMARY KEY (sender_id),
);

	FOREIGN KEY (message_id) REFERENCES  Messages (message_id)
	ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (user_id) REFERENCES Users (user_id)
	ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Sender
(
	receiver_id CHAR(20);
	message_id CHAR(20),
	user_id CHAR(20),
	PRIMARY KEY (receiver_id),
	FOREIGN KEY (message_id) REFERENCES  Messages (message_id)
	ON DELETE CASCADE ON UPDATE CASCADE,
	FOREIGN KEY (user_id) REFERENCES Users (user_id)
	ON DELETE CASCADE ON UPDATE CASCADE
);


CREATE TABLE Attachments
(
	attachment_id CHAR(20),
	message_id CHAR(20),
	file_id CHAR(20),
	PRIMARY KEY (attachment_id),
	FOREIGN KEY (message_id) REFERENCES Messages (message_id)
	ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE Files
```

```
(
	file_id CHAR(20),
	filename CHAR(40),
	file_url CHAR(255),
	PRIMARY KEY (file_id)
);
```