**WhatsApp**

# **ER Diagram**

## ER Formulation

**Scope of Project**

The requirement of the project is to make an online messaging service similar to WhatsApp. This service will allow us to communicate with people through text in an online medium. It should have features like :

1. One to One chat
2. Group chats
3. Read receipts
4. Media sharing
5. Last seen / Online status
6. User account
7. Settings - Privacy settings

The deliverable will be a Database of the messaging system.

**Stakeholders**

The stakeholders of the app are people who wish to communicate with each other through a textual medium online.

**Entities Defined with primary key underlined**

## Weak Entity

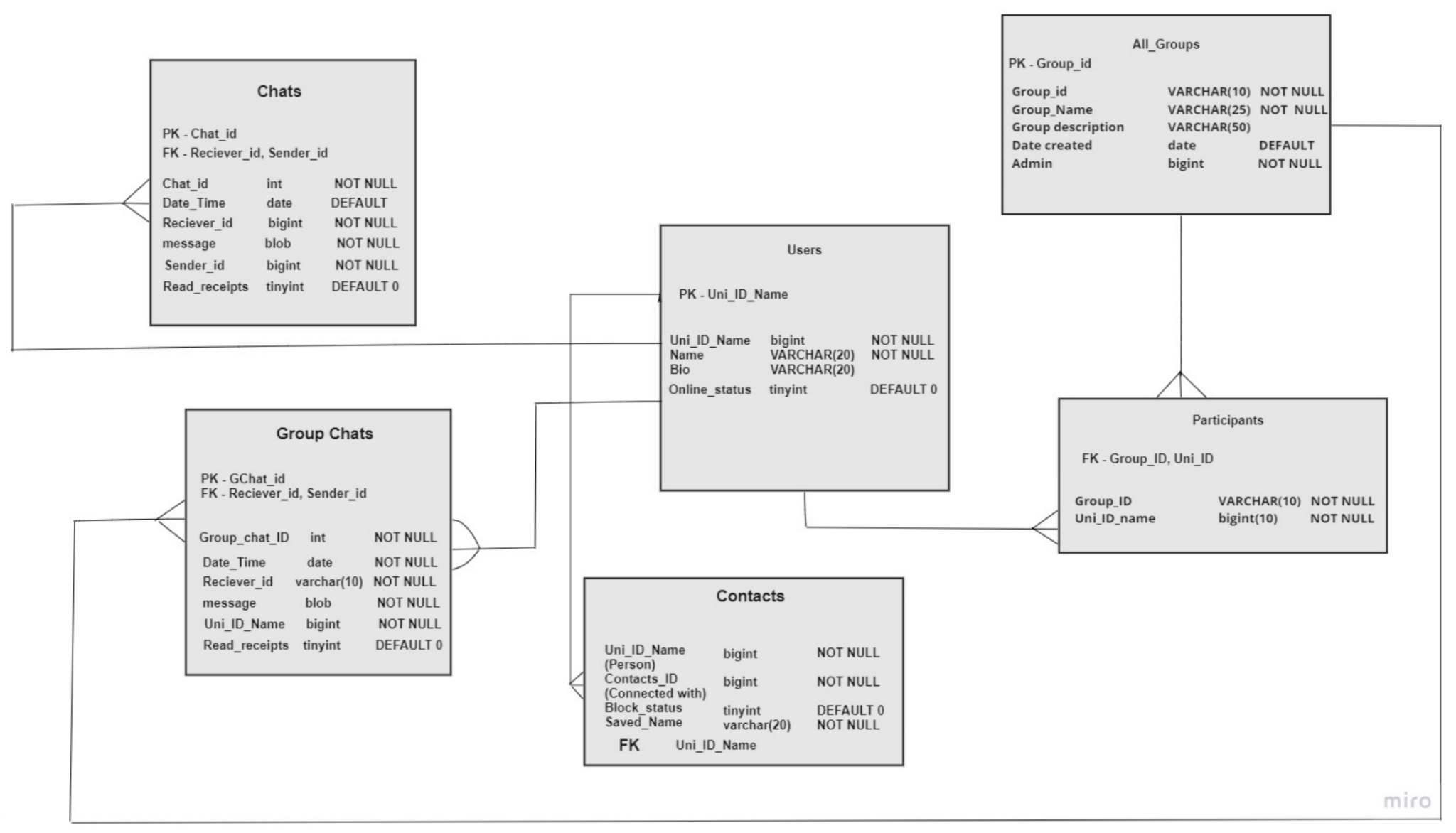
****

**Weak Entities** are the ones which depend on other entities for their existence. They do not have enough attributes to have a primary key. In our ER diagram we have two weak entities -

***Participants -*** It depends on the formation of groups. If there are no groups ~ there will be no participants in the group.

***Contacts -*** This depends on the existence of users as only a user will have a Uni\_ID and therefore will have Uni\_IDs of other people associated with them.

## Entity Relationship participation and their type

****

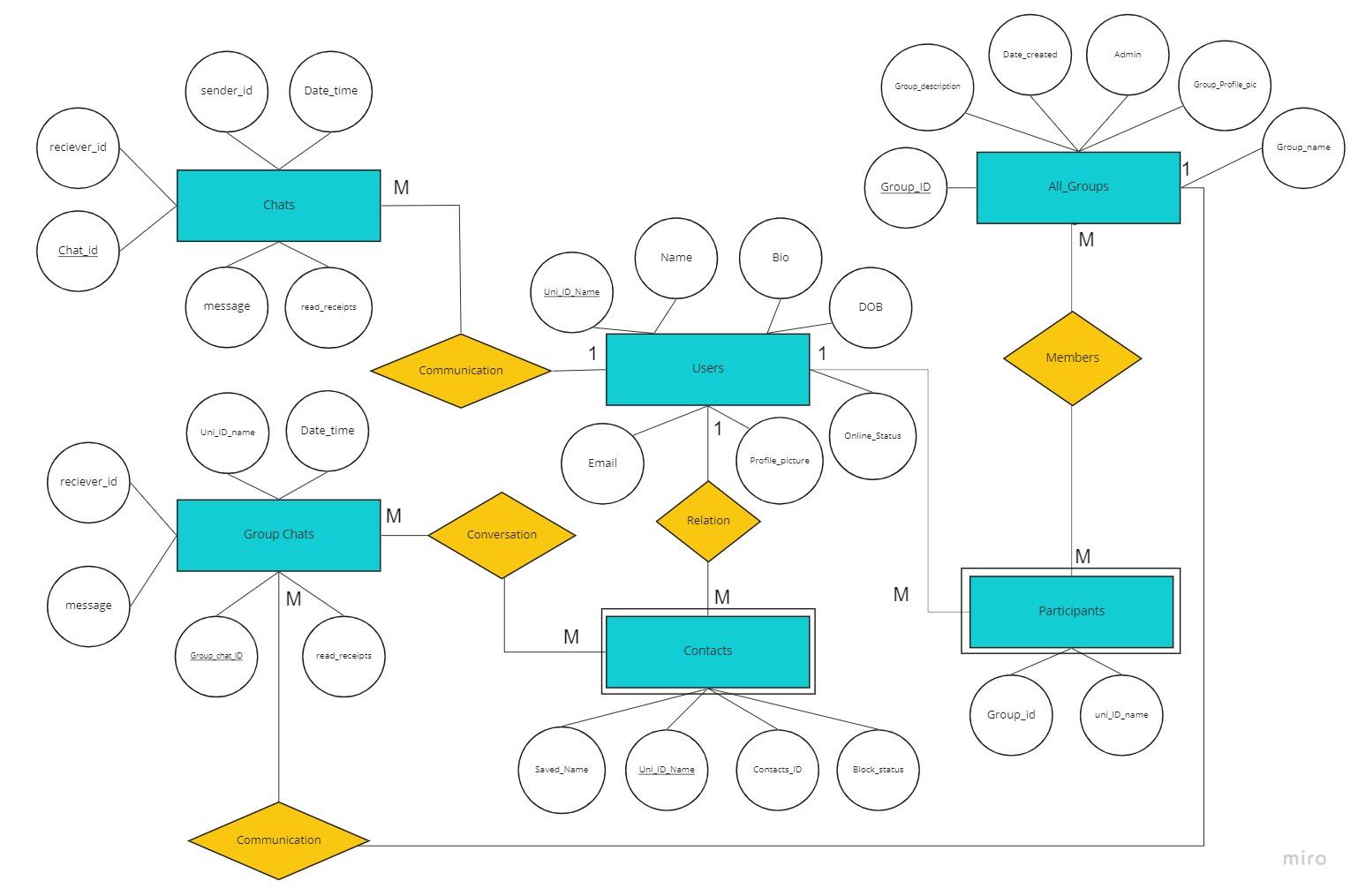
## Ternary Relationship

****

The above mentioned relationship is a ternary relationship as it requires 3 entities to participate in it. The ***user*** will send a ***message*** to ***communicate*** with the ***group chat.***

# **Relational Schema**

## Logical Database Design

****

## Mapping Constraints including Integrity Constraints

****

* Primary key - *Uni\_ID\_Name*
* *Uni\_ID\_Name* will have a NOT NULL constraint as it is the primary key of the table.
* *Name* will have a NOT NULL constraint as the user would require a name to start
* *Online\_Status* will have DEFAULT 0 constraint to set the default value of online status to false

****

* Foreign Key - *Uni\_ID\_Name, Contacts\_ID*
* *Uni\_ID\_Name* will have a NOT NULL constraint as it is the ID of the user
* *Contacts\_ID* will have a NOT NULL constraint as it is the ID of user’s contact
* *Block\_Status* will have a DEFAULT 0 constraint to set default blocked value as false
* *Saved\_Name* will have a NOT NULL constraint as we require a name for the saved contact for the ease of the user

****

* *Primary Key -* Chat\_ID
* *Foreign Key -* Sender\_ID, Receiver\_ID
* *Sender\_ID, Receiver\_ID, Chat\_ID, Message* will have NOT NULL constraints as these will be required when you will be sending a message
* *Date\_Time* will have a DEFAULT Getdate() constraint to set the time when the message is being sent
* *Read\_receipts* will have a DEFAULT 0 constraint to set the default value of read receipts to false

****

* Primary Key - Group\_ID
* *Group\_ID* will have a NOT NULL constraint as it is the primary key of the relation and it needs a value to uniquely identify the tuples.
* *Date\_created* attribute is set to DEFAULT GETDATE() function in case no value has been provided. This will add today’s date as a value in the tuple.
* *Admin* is set to NOT NULL because every group has at least one admin as the requirement.
* *Group\_name* is also NOT NULL because it’s mandatory for every group to have a name.

****

This table is basically the participants of a group.

* *Group\_ID* cannot be NULL.
* *uni\_ID\_name* also needs to have some value.

These values are required to depict the details of participants

Also both these attributes are FOREIGN KEYS and are referencing to All\_Groups and Users relations respectively.

****

* All the attributes of this table have to be NOT NULL, as their values are integral in proper functioning of Group chats relation.
* *Read\_receipts* attribute is of tinyint having a default value of 0 which means false which further means the message sent has not been read.

# **SQL Queries**

1. Finding the group of a specific person  
   select Grooup\_Name from all\_groups  
   where GroupID in( select GroupID from participants  
    where Uni\_ID = 9773509001);
2. Checking if a person is blocked or not  
   select Block\_status  
   From Contacts   
   Where Contacts\_ID = ‘1294817249’ ;
3. Getting the admin of a specific group  
   Select Admin  
   From All\_groups  
   Where Group\_Name = ‘Epic group#1’ ;
4. Searching up a contact  
   SELECT Contact\_ID from Contacts where Contact\_ID = ‘9996783498’
5. Searching the chat of a message  
   SELECT chat\_ID from Chats where (User\_ID = ‘9996783498’ AND message = ‘Today 9pm!’)
6. Searching up a group  
   SELECT Group\_ID from Groups where Group\_ID = ‘456892’
7. Fetching the members of a group  
   SELECT Uni\_ID\_name from Participants where   
   Group\_ID = ‘456892’
8. Fetch the ‘Bio’ of a person  
   SELECT Bio from Users where   
   Uni\_ID\_Name = ‘9996783498’
9. Getting all the texts that are unread  
   SELECT message from Chats where   
   Read\_receipts = 0 ;

10. Fetching all the contacts blocked by a user  
 SELECT name   
 from students  
 Where Uni\_ID\_name IN ( select Contacts\_ID   
 From Contacts  
 WHERE block\_status = 1 ) ;

# **Advanced SQL Queries**

1. Finding the number of members within a group

SELECT count(uni\_ID\_name)

From participants  
WHERE Group\_id = ‘1324221121’ ;

1. Finding how old is the group

SELECT CURRENT\_DATE - Date\_created  
From All\_Groups

WHERE Group\_name = ‘CSD’ ;

1. Finding the number of unread texts from a specific person

SELECT count(\*)

FROM Chats, Users  
WHERE Chats.Read\_receipts = 0 AND (Chats.Sender\_id = Users.Uni\_ID\_name AND Users.name = ‘Chhota Bheem’ ) ;

1. Finding the Number of Common Groups between two people

SELECT count(\*)

FROM participants p1, p2   
Where p1.Group\_id = p2.Group\_id AND p1.Uni\_ID\_name = ‘124235135’ AND

p2. Uni\_id\_name = ‘9876512498’ ;

# 

# **Views**

**1. CREATE VIEW contact\_info AS**

**SELECT uni\_id, name, bio, profile\_pic, status**

**FROM users**

**WHERE (SELECT contacts\_id FROM contacts;**

**2. CREATE VIEW grp\_desc AS**

**SELECT group\_description**

**FROM all\_groups**

**WHERE (SELECT group\_name FROM all\_groups;**

# **Grants**

1. **Admin can kick out a participant from a group**

**GRANT DELETE**

**ON participants**

**TO admin;**

1. **A user can delete their chats with any contact**

**GRANT DELETE**

**ON chats**

**TO uni\_id\_name;**

# **Indexing**

Indexing tables are created for data which is requires frequent access. In our database the following tasks can be performed more efficiently using index tables.

1. Accessing saved contacts  
   CREATE INDEX *savcon* ON Contacts(Contacts\_ID)
2. The groups the user has joined  
   CREATE INDEX *savgroup* ON All\_Groups(Group\_ID)
3. The number of participants of a group  
   CREATE INDEX *par* ON Participants(Uni\_ID\_name)

# **Triggers**

1. **Read receipts shown as blue tick**

-CREATE TRIGGER blue\_ticks

After INSERT, UPDATE

ON Chats

FOR EACH ROW

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* idk ye tick kaise implement hoga… mtlb colour change vgarah

SET tick = double\_blue ;

WHERE Read\_receipts = 1 ;

1. **The number of unread messages keeps updating**

**Create trigger unread**

**after insert, update**

**on Chats**

**as**

**begin**

**if exists(select \* from chats where Read\_receipts = 0;)**

**set count = count + 1**

**End ;**

**4. Display in the group chat when a participant joins or leaves a group NOT SURE ABOUT IT  
CREATE TRIGGER leave  
Before Update  
On Participants p1  
After update  
On Participants p2  
As begin  
Select Name   
From users   
Where Uni\_ID\_name in (  
Select Uni\_ID\_name from Participants where Uni\_ID\_name IN ( select p1.Uni\_ID\_name - p2.Uni\_ID\_name  
 From Participants ) ) ;**

**Set message = Name ‘has left the chat’   
END ;**