SEN2104
Database Management Systems

Project Report

Project Topic: Instant Messaging Social Platform
Student ID: Emir Sarı - Arda Özdemir
Student Full Name: 2103185 - 2101584

Project Definition

a. Project Topic and Work Partitioning

a.1- Project Topic

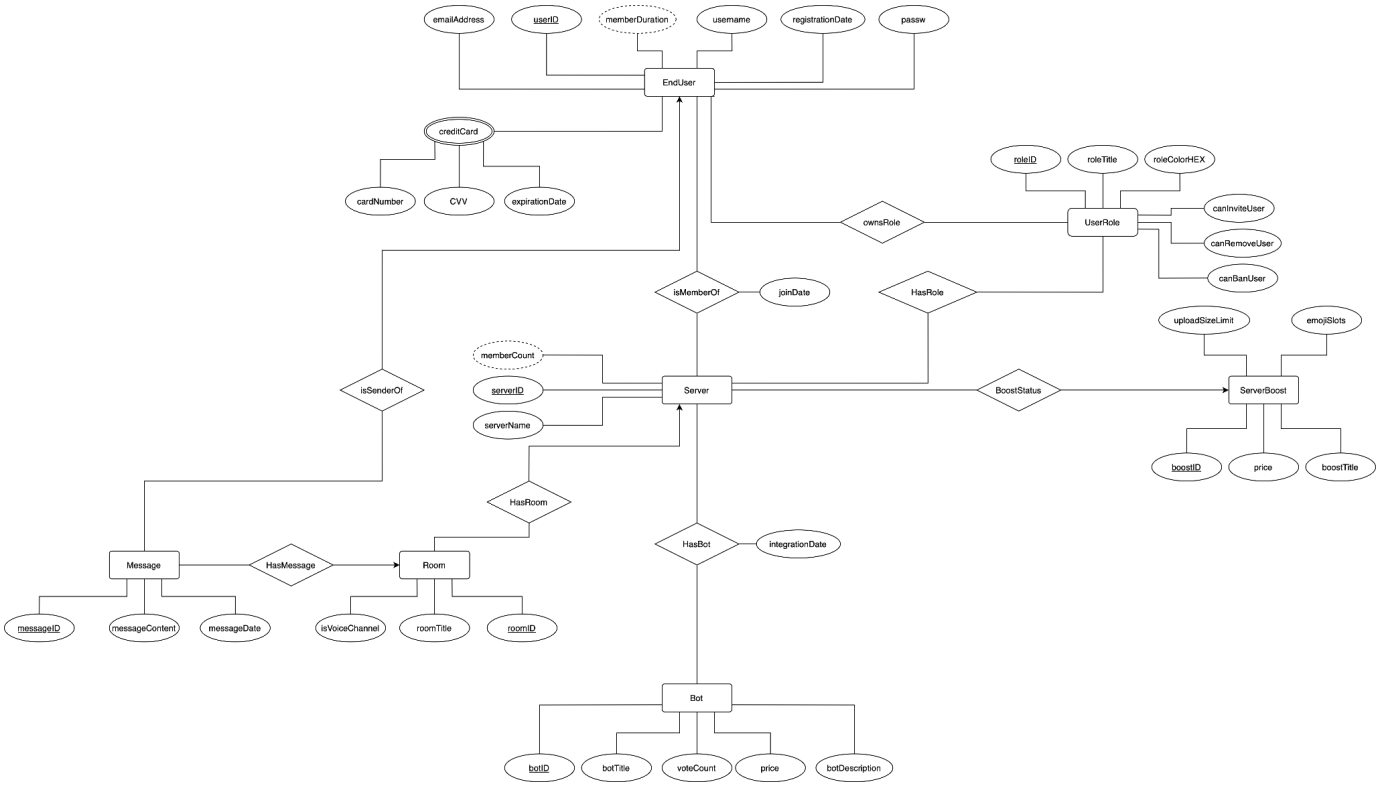
This project aims to design and create a database for the “Instant Messaging Social Platform” application. The structure of this database is on a user-server-bot relationship. Users can become a member of one or more servers. Servers have chat rooms where users can send messages related to the topic of the rooms. Users can boost servers to enhance their capabilities. Bots add unique functions to the servers. Users can pay for the boosts and the bots of the servers. Each user has their role, which is assigned to allow the user to invite, remove or ban another user. These roles are specific to the servers and users. This structure lets users interact and communicate with each other in a community-like environment.

a.2- Work Partitioning

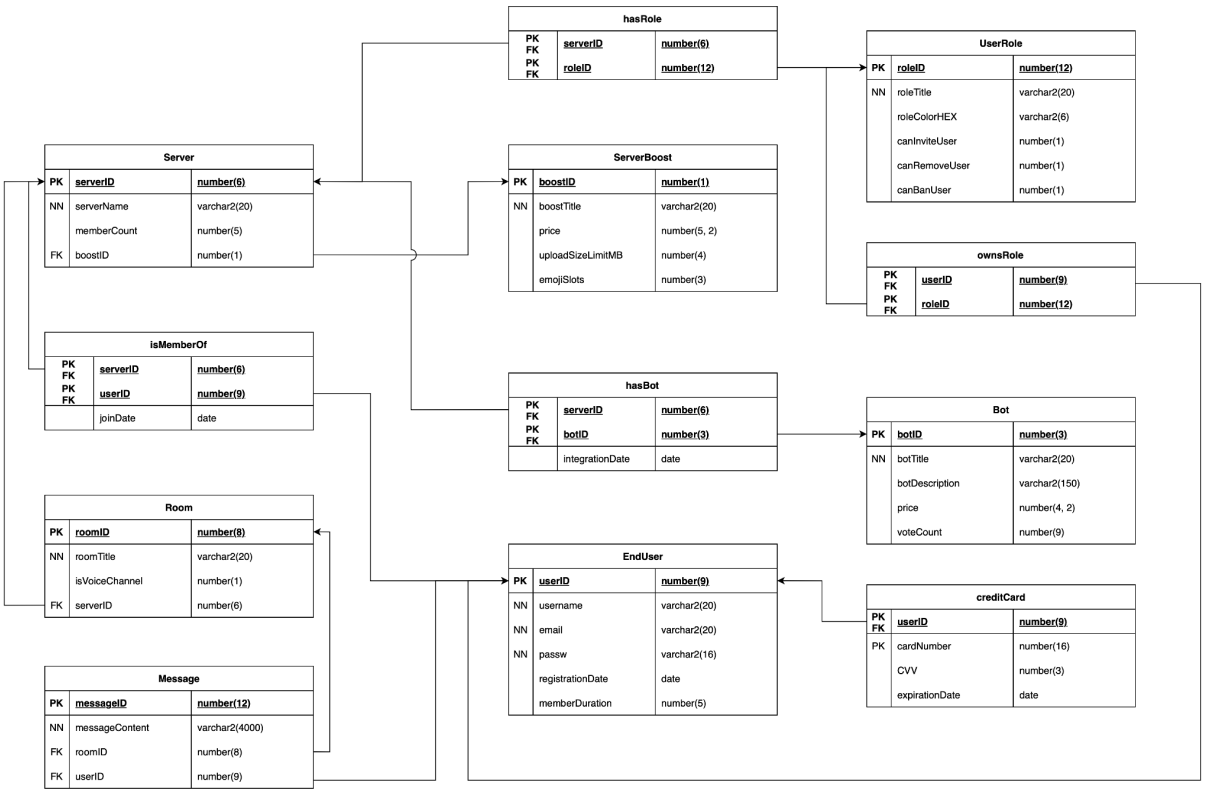
Name	*Description*
Emir Sarı 2103185	Designed the E-R Diagram and the Schema Diagram. I have done half of the create table, alter table, insert statements, and query-related tasks.
Arda Özdemir 2101584	Designed the E-R Diagram and the Schema Diagram. Using the diagrams, I wrote half of the statements of relations. I was responsible for half of creating and writing the queries.

During the decision-making processes, the creation of the E-R Diagram and the Schema Diagram of the project, the developers contributed together. As the steps are cumulative, the tasks are done in sequence because some statements (CREATE TABLE, INSERT INTO) are required to be compatible with each other, considering the constraints and relations of the database. For writing the queries, the Oracle SQL Developer application is used. **The work partitioning is explained in the presentations of each member in detail.**

b. E-R Diagram



c. Schema Diagram



hasRole

PK FK	serverID	number(6)
PK FK	roleID	number(12)

UserRole

PK	roleID	number(12)
NN	roleTitle	varchar2(20)
	roleColorHEX	varchar2(6)
	canInviteUser	number(1)
	canRemoveUser	number(1)
	canBanUser	number(1)

Server

PK	serverID	number(6)
NN	serverName	varchar2(20)
	memberCount	number(5)
FK	boostID	number(1)

ServerBoost

PK	boostID	number(1)
NN	boostTitle	varchar2(20)
	price	number(5, 2)
	uploadSizeLimitMB	number(4)
	emojiSlots	number(3)

ownsRole

PK FK	userID	number(9)
PK FK	roleID	number(12)

isMemberOf

PK FK	serverID	number(6)
PK FK	userID	number(9)
	joinDate	date

hasBot

PK FK	serverID	number(6)
PK FK	botID	number(3)
	integrationDate	date

Bot

PK	botID	number(3)
NN	botTitle	varchar2(20)
	botDescription	varchar2(150)
	price	number(4, 2)
	voteCount	number(9)

Room

PK	roomID	number(8)
NN	roomTitle	varchar2(20)
	isVoiceChannel	number(1)
FK	serverID	number(6)

EndUser

PK	userID	number(9)
NN	username	varchar2(20)
NN	email	varchar2(20)
NN	passw	varchar2(16)
	registrationDate	date
	memberDuration	number(5)

creditCard

PK FK	userID	number(9)
PK	cardNumber	number(16)
	CVV	number(3)
	expirationDate	date

Message

PK	messageID	number(12)
NN	messageContent	varchar2(4000)
FK	roomID	number(8)
FK	userID	number(9)

d. Create Table Statements

```
create table EndUser(userID number(9), username varchar2(20) not null, email varchar2(20)
not null, passw varchar2(16) not null, registrationDate date, memberDuration number(5),
primary key(userID));
create table Server(serverID number(6) primary key, serverName varchar2(20) not null,
memberCount number(5), boostID number(1));
create table Room(roomID number(8) primary key, roomTitle varchar2(20) not null,
isVoiceChannel number(1), serverID number(6));
create table Bot(botID number(3) primary key, botTitle varchar2(20) not null,
botDescription varchar2(150), price number(4,2), voteCount number(9));
create table hasBot(serverID number(6), botID number(3), integrationDate date);
create table Message(messageID number(12) primary key, messageContent varchar2(4000) not
null, roomID number(8), userID number(9));
create table ServerBoost(boostID number(1) primary key, boostTitle varchar2(20) not null,
price number(5, 2), uploadSizeLimitMB number(4), emojiSlots number(3));
create table UserRole(roleID number(12) primary key, roleTitle varchar2(20) not null,
roleColorHEX varchar2(6), canInviteUser number(1), canRemoveUser number(1), canBanUser
number(1));
create table creditCard(userID number(9), cardNumber number(16), CVV number(3),
expirationDate date);
create table hasRole(serverID number(6), roleID number(12));
create table isMemberOf(serverID number(6), userID number(9), joinDate date);
create table ownsRole(userID number(9), roleID number(12));

alter table Server add constraint Server_fk_boostID foreign key (boostID) references
ServerBoost(boostID);
alter table Room add constraint Room_fk_serverID foreign key (serverID) references
Server(serverID);
alter table Message add constraint Message_fk_roomID foreign key (roomID) references
Room(roomID);
alter table Message add constraint Message_fk_userID foreign key (userID) references
EndUser(userID);
alter table hasRole add constraint hasRole_fk_serverID foreign key (serverID) references
Server(serverID);
alter table hasRole add constraint hasRole_fk_roleID foreign key (roleID) references
UserRole(roleID);
alter table hasRole add constraint hasRole_pk primary key (serverID, roleID);

alter table creditCard add(constraint creditCard_pk primary key(userID, cardNumber),
constraint creditCard_fk foreign key(userID) references EndUser(userID));
alter table hasBot add(constraint hasBot_pk primary key(serverID, botID), constraint
hasBot_fk_server foreign key(serverID) references Server(serverID), constraint
hasBot_fk_bot foreign key(botID) references Bot(botID));
alter table isMemberOf add(constraint member_pk primary key(serverID, userID), constraint
member_fk_server foreign key(serverID) references Server(serverID), constraint
member_fk_user foreign key(userID) references EndUser(userID));
alter table ownsRole add(constraint ownsRole_pk primary key(userID, roleID), constraint
ownsRole_fk_EndUser foreign key(userID) references EndUser(userID), constraint
ownsRole_fk_UserRole foreign key(roleID) references UserRole(roleID));
```

e. Insert Into Statements

```
INSERT INTO Bot VALUES (057, 'Artificial Chat', 'A chat bot powered by AI.', 10.00,
134620);
INSERT INTO Bot VALUES (226, 'Bad Bot', 'A bot that generates spam messages.', 99.99,
-57539);
INSERT INTO Bot VALUES (743, 'Music Finder', 'A bot that finds songs via links or by
name.', 35.00, 48224853);
INSERT INTO EndUser VALUES (002103185, 'john105', 'john_1cena@bmail.com', '1998John',
 TO_DATE ('17/10/1996', 'dd/mm/yyyy'), null);
INSERT INTO EndUser VALUES (000000123, 'picassopablo9', 'guernica@gmail.com',
'pic1881asso!',
 TO_DATE ('02/06/2002','dd/mm/yyyy'), null);
INSERT INTO EndUser VALUES (002101584, 'ghostfaceganstа111', 'youcantseeme@mail.co',
'mytimeisnow',
 TO_DATE ('23/04/1977','dd/mm/yyyy'), null);
INSERT INTO UserRole VALUES (301000010011, 'moderator', 'da5d70', 1, 1, 1);
INSERT INTO UserRole VALUES (127000932000, 'user', '9b9b9b', 1, 0, 0);
INSERT INTO UserRole VALUES (200546000961, 'user', '42f5bf', 1, 1, 0);
INSERT INTO ServerBoost VALUES (1, 'Default', 0.00, 8, 5);
INSERT INTO ServerBoost VALUES (2, 'Boron', 14.99, 256, 50);
INSERT INTO ServerBoost VALUES (3, 'Platinum', 249.99, 1024, 250);
INSERT INTO creditCard VALUES (002103185, 1458999921211453, 432, TO_DATE ('01/03/2028',
'dd/mm/yyyy'));
INSERT INTO creditCard VALUES (000000123, 2219107119191111, 845, TO_DATE ('01/07/2030',
'dd/mm/yyyy'));
INSERT INTO creditCard VALUES (002101584, 1912474923313647, 739, TO_DATE ('01/08/2023',
'dd/mm/yyyy'));
INSERT INTO Server VALUES (123124,'X_ProGamers_X', null, 3);
INSERT INTO Server VALUES (921053,'Database Study', null, 1);
INSERT INTO Server VALUES (298124,'JazzLounge', null, 2);
INSERT INTO Room VALUES (13748383, 'Messi is better', 0, 123124);
INSERT INTO Room VALUES (99421001, 'general_chat', 0, 921053);
INSERT INTO Room VALUES (11215594, 'general_music', 1, 298124);
INSERT INTO Message VALUES (13748383, 'Who did win the world cup? Then, you know the
answer!', 13748383, 002101584);
INSERT INTO Message VALUES (99421001, 'No, you can't just enter null everywhere...',
99421001, 002103185);
INSERT INTO Message VALUES (11215594, 'Now playing: You Can't See Me', 11215594,
000000123);
INSERT INTO hasBot VALUES (123124, 057, TO_DATE ('07/05/2022', 'dd/mm/yyyy'));
INSERT INTO hasBot VALUES (921053, 057, TO_DATE ('07/05/2018', 'dd/mm/yyyy'));
INSERT INTO hasBot VALUES (298124, 743, TO_DATE ('07/05/2021', 'dd/mm/yyyy'));
INSERT INTO isMemberOf VALUES(123124, 002101584, TO_DATE ('03/01/2022', 'dd/mm/yyyy'));
INSERT INTO isMemberOf VALUES(921053, 002103185, TO_DATE ('09/02/2017', 'dd/mm/yyyy'));
INSERT INTO isMemberOf VALUES(298124, 000000123, TO_DATE ('18/12/2020', 'dd/mm/yyyy'));
INSERT INTO hasRole VALUES (123124, 301000010011);
INSERT INTO hasRole VALUES (921053, 127000932000);
INSERT INTO hasRole VALUES (298124, 200546000961);
INSERT INTO ownsRole VALUES (002101584, 301000010011);
INSERT INTO ownsRole VALUES (002103185, 127000932000);
INSERT INTO ownsRole VALUES (000000123, 200546000961);
```

f. SQL Queries

```
--2 joins with conditions--
SELECT e.username, u.roleTitle, s.serverName FROM EndUser e JOIN isMemberOf m ON e.userID
= m.userID JOIN Server s ON s.serverID = m.serverID JOIN hasRole h ON m.serverID =
h.serverID JOIN UserRole u ON h.roleID = u.roleID ORDER BY e.username, u.roleTitle,
s.serverName;
--Displays each user with its associated role in the server they are a member of.

SELECT e.username, c.cardNumber, c.CVV, c.expirationDate FROM EndUser e JOIN creditCard c
ON e.userID = c.userID ORDER BY e.username, c.cardNumber, c.CVV, c.expirationDate;
--Displays details of all registered cards of all users.

--2 nested queries--
SELECT userID, roleID FROM ownsRole WHERE userID IN (SELECT userID FROM ownsRole WHERE
roleID IN (SELECT roleID FROM UserRole WHERE canBanUser = 0 ));
--Displays all users with associated user roles that do not permit to ban users in the
related server.

SELECT REPLACE(LPAD(REPLACE(SUBSTR(TO_CHAR(cardNumber), 13), ',', ''), 16, '*'), ',', '
') as "Valid Card Numbers",  LTRIM(TO_CHAR(expirationDate, 'mm/yyyy')) as "Expiration
Dates" FROM creditCard WHERE userID IN (SELECT userID FROM creditCard WHERE
expirationDate > SYSDATE AND userID = 002101584);
--Displays card numbers with the related expiration dates of the user with ID
"002101584". Cards are displayed only if they are still valid. For privacy reasons, the
first 12 digits of the card numbers are not shown.

--2 set operations--
SELECT username FROM EndUser WHERE userID=ANY (SELECT userID from EndUser MINUS SELECT
DISTINCT userID FROM isMemberOf WHERE serverID = 921053);
--Displays all users who are not a member of the server which has the ID “921053”.

SELECT i.userID FROM isMemberOf i WHERE i.serverID = 921053 INTERSECT SELECT ii.userID
FROM isMemberOf ii WHERE ii.serverID = 123124;
--Displays all userID records which are related to servers “912053” and “123124”
simultaneously.

--2 aggregate operations including joins--
UPDATE EndUser SET EndUser.memberDuration = (SELECT (SYSDATE-e.registrationDate) FROM
EndUser e WHERE e.userID = EndUser.userID);
--Calculates and updates membership duration for all users in the EndUser table.

UPDATE Server SET Server.memberCount = (SELECT COUNT (*) FROM isMemberOf s WHERE
s.serverID = Server.serverID);
--Calculates and updates member counts of all servers in the Server table.
```

g. Relational Algebra Queries

$$\Pi \text{ serverName, botID}(\sigma \text{ Server.serverID = HasBot.serverID}^{(\text{Server x HasBot})})$$

--1 Displays server names with their related bot IDs.

$$\Pi \text{ username, registrationDate, memberDuration } (\sigma \text{ memberDuration >= } 365^{(\text{EndUser})})$$

--2 Displays username, registration date, and membership duration of users who have been a member for at least 365 days.

$$\Pi \ \rho\text{NameOfUser(username)}^{(\text{EndUser})} \cup \Pi \ \rho\text{NameOfServer(serverName)}^{(\text{Server})}$$

--3 Displays all user names under the title of NameOfUser and all server names under the title of NameOfServer, for the purpose of exporting.

$$\Pi \text{ userID}^{(\text{EndUser})} - \Pi \text{ userID}^{(\text{isMemberOf})} \cap \Pi \text{ memberDuration} > 365^{(\text{EndUser})}$$

--4 Displays userIDs which belong to users who neither are a member of a server nor have been a member of the application for less than 365 days, for the purpose of detecting inactive users.

$$\text{serverQuery} \leftarrow \sigma \text{ serverID} = 123124^{(\text{Server})}$$
$$\text{roomsInServer} \leftarrow \text{serverQuery} \bowtie \text{Room}$$
$$\Pi \text{ roomTitle}^{(\text{roomsInServer})}$$

--5 Displays all rooms that are related to the server with the ID “123124”.

$$\text{avgMemberCount} \leftarrow g \text{ AVG(memberCount)}^{(\text{Server})}$$
$$\Pi \text{ serverID, serverName } (\sigma \text{ memberCount} > \text{avgMemberCount}^{(\text{Server})})$$

--6 Displays servers with their IDs and names, which have a greater number of members than the average member count of all servers.

$$\text{roomCountPerServer} \leftarrow \rho\text{RoomCountTable}_{(\text{serverID, RoomCount})}(\text{serverID } g \text{ COUNT(RoomID)}^{(\text{Room})})$$
$$\Pi \text{ serverID, serverName, RoomCount}^{(\text{roomCountPerServer} \bowtie \text{Server})}$$

--7 Displays the number of rooms per server of all servers.

$$\Pi \text{ username, serverName, roleTitle}^{(\text{EndUser} \bowtie (\text{ownsRole} \bowtie \text{UserRole}) \bowtie \text{isMemberOf})}$$

--8 Displays all usernames, server names, and related role titles for each user, server, and role, to view the user-server-role relationship of the database.

h. Relational Calculus Queries

```
{ t | ∃b ∈ Bot (t[botTitle] = b[botTitle]) ∧ (b[voteCount] > 1000 ∧
b[price] < 50)}
--1 Displays the bot titles, which have a vote count greater than 1000
and a price less than 50.

{ <sn, bt> | ∃sid, sn, mc, boid, bid, intdate(<sid, sn, mc, boid> ∈
Server ∧ <sid, bid, '2022-01-01'> ∈ HasBot)}
--2 Displays the server names and bot titles of servers which have bots
integrated at the date, ‘2022-01-01’.

{ t | ∃s ∈ Server (t[serverName] = s[serverName]) ∧ (∃sb ∈ ServerBoost
(s[boostID] = sb[boostID] ∧ sb[boostID] = 1 ∧ ∃r ∈ Room (s[serverID] =
r[serverID] ∧ r[isVoiceChannel] = 1)))}
--3 Displays the server names of all servers which have a boost ID of 1
and have at least one room that is a voice channel.

{ t | ∃m ∈ Message (t[messageID] = m[messageID]) ∧ (∃u ∈ EndUser
(m[userID] = u[userID]∧ u[userID] = 000000123 ∧ ∃r ∈ Room (m[roomID] =
r[roomID] ∧ r[roomID] = 11215594)))}
--4 Displays the message IDs of all messages which are sent by a specific
user, with ID ‘000000123’ in a room, with ID ‘11215594’.
```