BAZE PODATAKA

Naslov: Socijalna Mreža

|  |  |
| --- | --- |
| Student: David Katrinka, Stepan Turicin  Indeks: 26123035, 26123036  Smer: Informatika | Profesor: DR ŠIMON JANOŠ |

Sadržaj

[Specifikacija zahteva 3](#_Toc1)

[Prikaz ER modela baze 4](#_Toc2)

[Struktura baze 6](#_Toc3)

[DDL skripta 7](#_Toc4)

[DML skripta 15](#_Toc5)

[SQL upiti 21](#_Toc6)

[Pogledi (VIEWS) 26](#_Toc7)

[Okidači (TRIGGERS) 30](#_Toc8)

[Uskladištene funkcije 32](#_Toc9)

[Literatura 35](#_Toc10)

# Specifikacija zahteva

Zahtevi za praćenje i upravljanje društvenom mrežom.

1. Osnovni opis baze

Ova baza podataka je namenjena za društvenu mrežu. U njoj čuvamo informacije o korisnicima, njihovim prijateljima, postovima, porukama, reakcijama i datotekama. Sve je organizovano u tabele koje su međusobno povezane, tako da možemo lako doći do podataka i izgraditi funkcionalnosti.

Korisnici (users)

Ovo je glavna tabela gde čuvamo informacije o korisnicima.

Svaki korisnik ima svoj jedinstveni ID, korisničko ime, email i lozinku.

Takođe možemo sačuvati dodatne informacije, poput imena, prezimena, avatara i toga da li je profil privatan ili javan.

Korisnici mogu imati različite uloge, na primer običan korisnik, administrator ili blokirani.

Prijateljstva (followers)

Ovde pratimo ko koga prati.

Ako korisnik A prati korisnika B, to beležimo u ovoj tabeli.

Svaka veza između dva korisnika ima vreme kada je započeta.

Postovi (posts)

Korisnici mogu kreirati postove, koji se čuvaju u ovoj tabeli.

Postovi sadrže tekst i pripadaju određenom korisniku.

Beleži se vreme kada je post kreiran i poslednji put ažuriran.

Komentari na postove (post\_comments)

Komentari na postove su organizovani tako da svaki komentar zna kojem postu pripada.

Komentari mogu imati i odgovore, što znači da jedan komentar može biti "roditelj" drugom komentaru.

Reakcije na postove (post\_reactions)

Korisnici mogu reagovati na postove. Reakcije mogu biti "sviđa mi se" ili "ne sviđa mi se".

Ova tabela beleži koji korisnik je reagovao na koji post i kada.

Ćaskanja i poruke

Ćaskanja (chats): Ćaskanja mogu biti između dva korisnika (privatni) ili grupni.

Učesnici ćaskanja (chat\_participants): Ovde beležimo ko učestvuje u kojem ćaskanju.

Poruke (chat\_messages): Poruke se povezuju sa razgovorima i korisnicima koji su ih poslali.

Fajlovi (files)

Fajlovi poput slika, video zapisa ili dokumenata se čuvaju ovde.

Svaki fajl je povezan sa korisnikom koji ga je otpremio.

Fajlovi se mogu "prikačiti" na postove ili poruke.

Ključne funkcionalnosti

Dodavanje korisnika: Možete registrovati novog korisnika sa jedinstvenim korisničkim imenom i emailom.

Prijatelji: Možete videti ko koga prati i ko su vam prijatelji.

Postovi i komentari: Korisnici mogu objavljivati postove, komentarisati ih i reagovati na njih.

Poruke: Korisnici mogu slati poruke u privatnim ili grupnim ćaskanjima.

Fajlovi: Možete otpremati slike i druge fajlove, koji se kasnije mogu koristiti u postovima ili porukama.

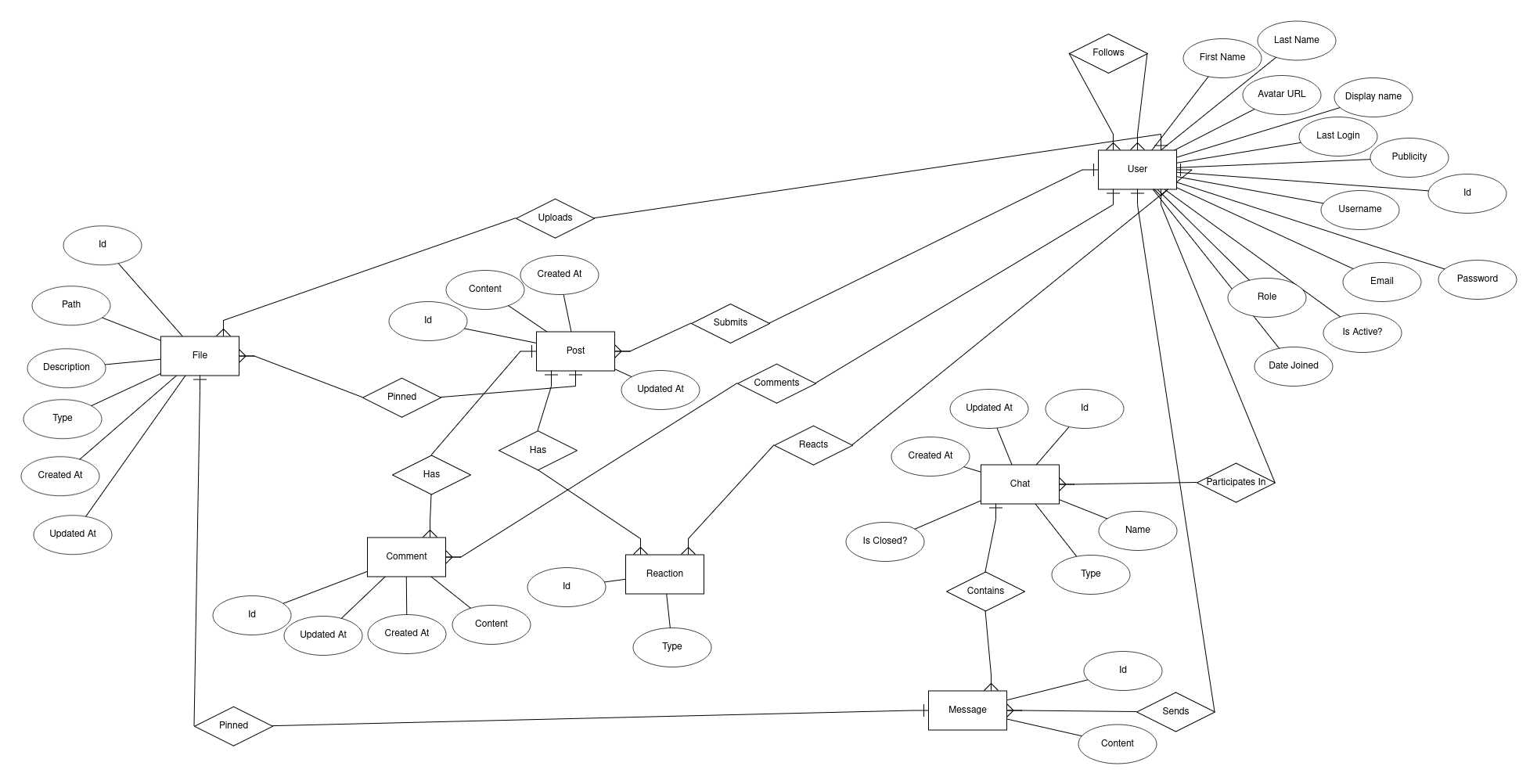
Ko ovo može koristiti?

Programeri koji razvijaju društvenu mrežu.

Administratori koji žele da prate aktivnosti korisnika.

Aplikacije koje zahtevaju interakciju korisnika kroz postove, komentare i poruke.

# Prikaz ER modela baze



Slika 1. ER model baze

S={USERS(id, username, password, email, role, is\_active, last\_login, data\_joined, first\_name, last\_name, avatar\_url, display\_name, publicity),

FOLLOWERS(follower\_user\_id, followed\_user\_id, created\_at),

CHATS(id, name, type, is\_closed, created\_at, updated\_at),

CHAT\_PARTICIPANTS(user\_id, chat\_id, is\_admin),

CHAT\_MESSAGES(id, content, user\_id, chat\_id),

POSTS(id, content, user\_id, created\_at, updated\_at),

POST\_COMMENTS(id, content, created\_at, updated\_at, parent\_id, post\_id, user\_id),

POST\_REACTIONS(id, type, created\_at, post\_id, user\_id),

FILES(id, path\_url, description, type, created\_at, updated\_at, user\_id),

MESSAGE\_PINNED\_FILES(message\_id, file\_id, sort\_order),

POST\_PINNED\_FILES(post\_id, file\_id, sort\_order)}

I={USERS[role]⊆{’blocked’, ’user’, ’admin’},

USERS[publicity]⊆{’private’, ’only\_friends’, ’friend\_friends’, ’everyone’},

FOLLOWERS[follower\_user\_id]⊆USERS[id],

FOLLOWERS[followed\_user\_id]⊆USERS[id],

CHATS[type]⊆{’personal’, ’group’},

CHAT\_PARTICIPANTS[user\_id]⊆USERS[id],

CHAT\_PARTICIPANTS[chat\_id]⊆CHATS[id],

CHAT\_MESSAGES[user\_id]⊆USERS[id],

CHAT\_MESSAGES[chat\_id]⊆CHATS[id],

POSTS[user\_id]⊆USERS[id],

POST\_COMMENTS[parent\_id]⊆POST\_COMMENTS[id],

POST\_COMMENTS[post\_id]⊆POSTS[id],

POST\_COMMENTS[user\_id]⊆USERS[id],

POST\_REACTIONS[post\_id]⊆POSTS[id],

POST\_REACTIONS[user\_id]⊆USERS[id],

FILES[user\_id]⊆USERS[id],

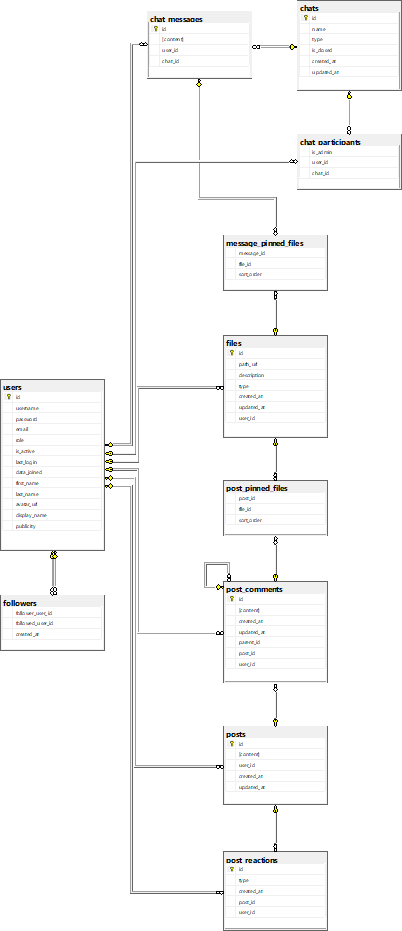
MESSAGE\_PINNED\_FILES[message\_id]⊆CHAT\_MESSAGES[id],

MESSAGE\_PINNED\_FILES[file\_id]⊆FILES[id],

POST\_PINNED\_FILES[post\_id]⊆POSTS[id],

POST\_PINNED\_FILES[file\_id]⊆FILES[id]}

# Struktura baze



Slika 2. Struktura baze (MSSQL Management Studio)

# DDL skripta

-- +------------------------------+

-- | U S E R S |

-- +------------------------------+

CREATE TABLE users (

-- Private information

id INT IDENTITY UNIQUE NOT NULL,

username VARCHAR(30) UNIQUE NOT NULL,

password VARCHAR(25) NOT NULL,

email VARCHAR(50) UNIQUE NOT NULL,

role CHAR(15) NOT NULL DEFAULT 'user',

is\_active INT NOT NULL DEFAULT 1,

last\_login DATETIME NULL,

data\_joined DATETIME DEFAULT CURRENT\_TIMESTAMP,

-- Public information

first\_name VARCHAR(25) DEFAULT '',

last\_name VARCHAR(25) DEFAULT '',

avatar\_url VARCHAR(255) NULL,

display\_name VARCHAR(255)NULL,

publicity CHAR(30) NOT NULL DEFAULT 'everyone',

CONSTRAINT user\_pk PRIMARY KEY(id),

CONSTRAINT user\_role\_check CHECK(role IN ('blocked', 'user', 'admin')),

CONSTRAINT user\_publicity\_check CHECK(publicity IN ('private', 'only\_friends', 'friend\_friends', 'everyone'))

);

-- if users are subscribed to each other, they are friends.

CREATE TABLE followers (

follower\_user\_id INT NOT NULL,

followed\_user\_id INT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

FOREIGN KEY(follower\_user\_id) REFERENCES users(id),

FOREIGN KEY(followed\_user\_id) REFERENCES users(id)

);

-- +------------------------------+

-- | C H A T S |

-- +------------------------------+

CREATE TABLE chats (

id INT IDENTITY UNIQUE NOT NULL,

name VARCHAR(50) NOT NULL,

type CHAR(15) NOT NULL DEFAULT 'personal',

is\_closed INT NOT NULL DEFAULT 0,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT chat\_pk PRIMARY KEY(id),

CONSTRAINT chat\_type\_check CHECK(type IN ('personal', 'group'))

);

CREATE TABLE chat\_participants (

is\_admin INT NOT NULL DEFAULT 0,

user\_id INT NOT NULL,

chat\_id INT NOT NULL,

FOREIGN KEY(user\_id) REFERENCES users(id),

FOREIGN KEY(chat\_id) REFERENCES chats(id)

);

CREATE TABLE chat\_messages (

id INT IDENTITY UNIQUE NOT NULL,

content TEXT NOT NULL,

user\_id INT NOT NULL,

chat\_id INT NOT NULL,

CONSTRAINT chat\_message\_pk PRIMARY KEY(id),

FOREIGN KEY(user\_id) REFERENCES users(id),

FOREIGN KEY(chat\_id) REFERENCES chats(id)

);

-- +------------------------------+

-- | P O S T S |

-- +------------------------------+

CREATE TABLE posts (

id INT IDENTITY UNIQUE NOT NULL,

content TEXT NOT NULL,

user\_id INT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

CONSTRAINT post\_pk PRIMARY KEY(id),

FOREIGN KEY(user\_id) REFERENCES users(id)

);

CREATE TABLE post\_comments (

id INT IDENTITY UNIQUE NOT NULL,

content TEXT NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

-- If user reply other comment

parent\_id INT NULL,

post\_id INT NOT NULL,

user\_id INT NOT NULL,

CONSTRAINT post\_comment\_pk PRIMARY KEY(id),

FOREIGN KEY(parent\_id) REFERENCES post\_comments(id),

FOREIGN KEY(post\_id) REFERENCES posts(id),

FOREIGN KEY(user\_id) REFERENCES users(id)

);

CREATE TABLE post\_reactions (

id INT IDENTITY UNIQUE NOT NULL,

type CHAR(15) NOT NULL DEFAULT 'like',

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

post\_id INT NOT NULL,

user\_id INT NOT NULL,

CONSTRAINT post\_reaction\_pk PRIMARY KEY(id),

CONSTRAINT post\_reaction\_type CHECK(type IN ('like', 'dislike')),

FOREIGN KEY(post\_id) REFERENCES posts(id),

FOREIGN KEY(user\_id) REFERENCES users(id)

);

-- +------------------------------+

-- | F I L E S |

-- +------------------------------+

-- Only for images

CREATE TABLE files (

id INT IDENTITY UNIQUE NOT NULL,

path\_url VARCHAR(255) NOT NULL,

description VARCHAR(255) NULL,

type CHAR(15) NOT NULL,

created\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

updated\_at DATETIME DEFAULT CURRENT\_TIMESTAMP,

user\_id INT NOT NULL,

CONSTRAINT file\_pk PRIMARY KEY(id),

CONSTRAINT file\_type\_check CHECK(type IN ('file', 'image', 'audio', 'video')),

FOREIGN KEY(user\_id) REFERENCES users(id)

);

-- Pinned files to post, to messages

CREATE TABLE message\_pinned\_files (

message\_id INT NOT NULL,

file\_id INT NOT NULL,

sort\_order INT NOT NULL DEFAULT 0,

FOREIGN KEY(message\_id) REFERENCES chat\_messages(id),

FOREIGN KEY(file\_id) REFERENCES files(id)

);

CREATE TABLE post\_pinned\_files (

post\_id INT NOT NULL,

file\_id INT NOT NULL,

sort\_order INT NOT NULL DEFAULT 0,

FOREIGN KEY(post\_id) REFERENCES post\_comments(id),

FOREIGN KEY(file\_id) REFERENCES files(id)

);

# DML skripta

-- Insert into users table

INSERT INTO users (username, password, email, role, first\_name, last\_name, avatar\_url, display\_name) VALUES

('john\_doe1', 'password123', 'john.doe1@example.com', 'user', 'John', 'Doe', 'https://example.com/avatar1.jpg', 'John D'),

('alice\_s1', 'password456', 'alice.s1@example.com', 'user', 'Alice', 'Smith', 'https://example.com/avatar2.jpg', 'Alice S'),

('bob\_jones', 'password789', 'bob.jones@example.com', 'admin', 'Bob', 'Jones', 'https://example.com/avatar3.jpg', 'Bob J'),

('carol\_wh1', 'password101', 'carol.wh1@example.com', 'user', 'Carol', 'White', 'https://example.com/avatar4.jpg', 'Carol W'),

('david\_b2', 'password102', 'david.b2@example.com', 'blocked', 'David', 'Brown', 'https://example.com/avatar5.jpg', 'David B'),

('emma\_tay', 'password103', 'emma.tay@example.com', 'user', 'Emma', 'Taylor', 'https://example.com/avatar6.jpg', 'Emma T'),

('frank\_lee', 'password104', 'frank.lee@example.com', 'admin', 'Frank', 'Lee', 'https://example.com/avatar7.jpg', 'Frank L'),

('grace\_mil', 'password105', 'grace.mil@example.com', 'user', 'Grace', 'Miller', 'https://example.com/avatar8.jpg', 'Grace M'),

('hank\_clrk', 'password106', 'hank.clrk@example.com', 'user', 'Hank', 'Clark', 'https://example.com/avatar9.jpg', 'Hank C'),

('irene\_dvs', 'password107', 'irene.dvs@example.com', 'user', 'Irene', 'Davis', 'https://example.com/avatar10.jpg', 'Irene D');

-- Insert into followers table

INSERT INTO followers (follower\_user\_id, followed\_user\_id) VALUES

(1, 2),

(2, 3),

(3, 4),

(4, 5),

(5, 6),

(6, 7),

(7, 8),

(8, 9),

(9, 10),

(10, 1);

-- Insert into chats table

INSERT INTO chats (name, type, is\_closed) VALUES

('Chat1', 'personal', 0),

('Group1', 'group', 0),

('Chat2', 'personal', 0),

('Group2', 'group', 0),

('Chat3', 'personal', 0),

('Group3', 'group', 0),

('Chat4', 'personal', 1),

('Group4', 'group', 0),

('Chat5', 'personal', 0),

('Group5', 'group', 0);

-- Insert into chat\_participants table

INSERT INTO chat\_participants (is\_admin, user\_id, chat\_id) VALUES

(1, 1, 1),

(0, 2, 1),

(1, 3, 2),

(0, 4, 2),

(1, 5, 3),

(0, 6, 3),

(0, 7, 4),

(1, 8, 4),

(1, 9, 5),

(0, 10, 5);

-- Insert into chat\_messages table

INSERT INTO chat\_messages (content, user\_id, chat\_id) VALUES

('Hello, how are you?', 1, 1),

('I am fine, thank you!', 2, 1),

('Lets meet tomorrow', 3, 2),

('Looking forward to it', 4, 2),

('What time works for you?', 5, 3),

('Anytime after 3 PM', 6, 3),

('Lets discuss the project', 7, 4),

('Sure, when is the meeting?', 8, 4),

('Got the files, will review', 9, 5),

('Thanks, Ill check them', 10, 5);

-- Insert into posts table

INSERT INTO posts (content, user\_id) VALUES

('Just finished a great workout!', 1),

('Had an amazing lunch today!', 2),

('Completed a new project!', 3),

('Enjoying a beautiful day at the park!', 4),

('Started reading a new book!', 5),

('The weather is fantastic!', 6),

('Worked on an exciting new idea!', 7),

('Had a great weekend getaway!', 8),

('Loving this new song!', 9),

('Exploring a new hobby!', 10);

-- Insert into post\_comments table

INSERT INTO post\_comments (content, post\_id, user\_id, parent\_id) VALUES

('Nice job on the workout!', 1, 2, NULL),

('Sounds delicious!', 2, 3, NULL),

('Congrats on the project!', 3, 4, NULL),

('That sounds relaxing!', 4, 5, NULL),

('Enjoy the book!', 5, 6, NULL),

('I agree, the weather is amazing!', 6, 7, NULL),

('Great idea, keep it up!', 7, 8, NULL),

('Sounds like a lot of fun!', 8, 9, NULL),

('I love that song too!', 9, 10, NULL),

('Thats awesome! What hobby?', 10, 1, NULL);

-- Insert into post\_reactions table

INSERT INTO post\_reactions (type, post\_id, user\_id) VALUES

('like', 1, 2),

('dislike', 2, 3),

('like', 3, 4),

('like', 4, 5),

('dislike', 5, 6),

('like', 6, 7),

('like', 7, 8),

('dislike', 8, 9),

('like', 9, 10),

('like', 10, 1);

-- Insert into files table

INSERT INTO files (path\_url, description, type, user\_id) VALUES

('https://exmpl.com/file1.jpg', 'Profile Picture', 'image', 1),

('https://exmpl.com/file2.jpg', 'Vacation Photo', 'image', 2),

('https://exmpl.com/file3.mp3', 'Music Track', 'audio', 3),

('https://exmpl.com/file4.mp4', 'Video Clip', 'video', 4),

('https://exmpl.com/file5.pdf', 'Resume', 'file', 5),

('https://exmpl.com/file6.jpg', 'Family Photo', 'image', 6),

('https://exmpl.com/file7.mp3', 'Podcast', 'audio', 7),

('https://exmpl.com/file8.mp4', 'Conference Presentation', 'video', 8),

('https://exmpl.com/file9.jpg', 'Profile Picture', 'image', 9),

('https://exmpl.com/file10.pdf', 'Project Proposal', 'file', 10);

-- Insert into message\_pinned\_files table

INSERT INTO message\_pinned\_files (message\_id, file\_id, sort\_order) VALUES

(1, 1, 0),

(2, 2, 1),

(3, 3, 2),

(4, 4, 3),

(5, 5, 4),

(6, 6, 5),

(7, 7, 6),

(8, 8, 7),

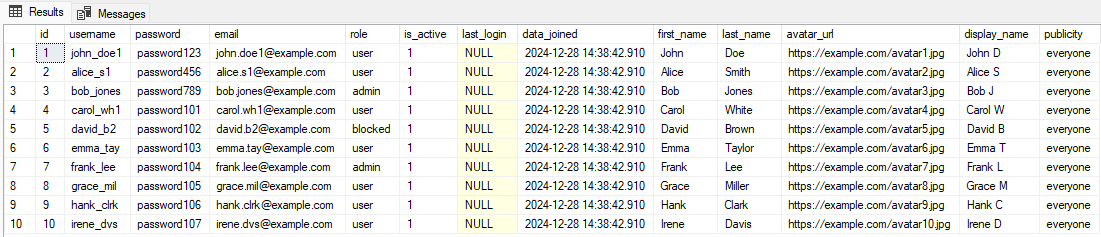
(9, 9, 8),

(10, 10, 9);

# SQL upiti

1. Upit koji nam daje sve korisnike i sve podatke o njima:

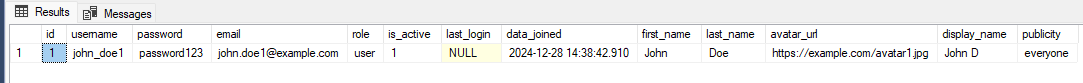
SELECT \* FROM users;



Slika 3. lista svih korisnika i njihovih podataka

2. Upit koji nam daje podatke o specifičnom korisniku:

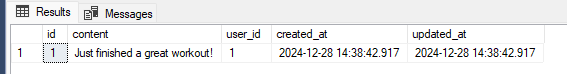
SELECT \* FROM users WHERE username = 'john\_doe1';



Slika 4. lista svih korisnika sa imenom john\_doe1

3.Upit koji nam daje sve objave jednog korisnika (prema korisničkom id):

SELECT \* FROM posts WHERE user\_id = 1;



Slika 5. lista svih objava korisnika sa id koji je 1

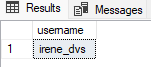
4. Upit koji nam daje sve pratioce (followers) jendnog korisnika (prema njegovom id):

SELECT u.username

FROM followers f

INNER JOIN users u ON f.follower\_user\_id = u.id

WHERE f.followed\_user\_id = 1;



Slika 6. lista svih pratilaca korisnika sa id koji je 1

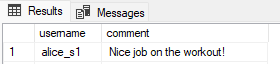
5. Upit koji nam daje sve komentare postavljene na specifičnu objavu:

SELECT u.username, pc.content AS comment

FROM post\_comments pc

INNER JOIN users u ON pc.user\_id = u.id

WHERE pc.post\_id = 1;



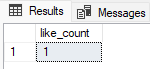
Slika 7. lista svih komentara ispod objave čiji je id jednak 1

6. Upit koji nam pokazuje koliko lajkova ima jedan post:

SELECT COUNT(\*) AS like\_count

FROM post\_reactions

WHERE post\_id = 1 AND type = 'like';



Slika 8. broj lajkova objave čiji je id jednak 1

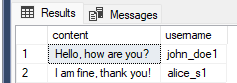
7. Upit koji nam daje sve poruke iz jednog ćaskanja i ko ih je poslao:

SELECT cm.content, u.username

FROM chat\_messages cm

INNER JOIN users u ON cm.user\_id = u.id

WHERE cm.chat\_id = 1;



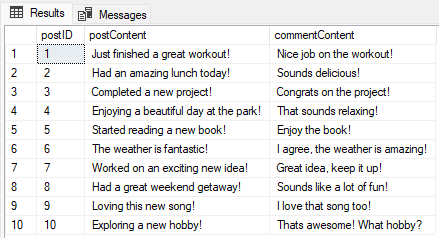
Slika 9. lista svih poruka i njihovih šaljilaca iz ćaskanja čiji je id jednak 1

8. Upit koji nam daje sve objave sa njihovim komentarima:

SELECT p.id AS post\_id, p.content AS post\_content, pc.content AS comment\_content

FROM posts p

LEFT JOIN post\_comments pc ON p.id = pc.post\_id;



Slika 10. lista svih objava i njihovih komentara

9. Upit koji nam polazuje sve objave i broj lajkova koji imaju:

SELECT p.id AS post\_id,

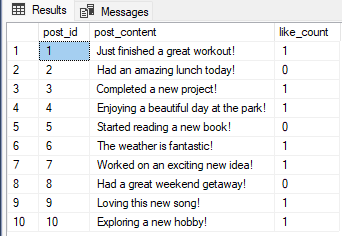
CAST(p.content AS VARCHAR(MAX)) AS post\_content,

COUNT(pr.id) AS like\_count

FROM posts p

LEFT JOIN post\_reactions pr ON p.id = pr.post\_id AND pr.type = 'like'

GROUP BY p.id, CAST(p.content AS VARCHAR(MAX));



Slika 11. lista svih objava i broj lajkova koji imaju

10. Upit koji nam daje sve objave određene grupe korisnika, koje imaju 1 ili više lajkova:

SELECT p.id AS post\_id,

CAST(p.content AS VARCHAR(MAX)) AS post\_content,

COUNT(pr.id) AS like\_count

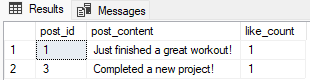
FROM posts p

LEFT JOIN post\_reactions pr ON p.id = pr.post\_id AND pr.type = 'like'

WHERE p.user\_id IN (1, 2, 3)

GROUP BY p.id, CAST(p.content AS VARCHAR(MAX))

HAVING COUNT(pr.id) >= 1;



Slika 12. lista svih objava koje su postavili korisnici čiji je id jednak jedan i ako imaju 1 ili više lajkova

# Pogledi (VIEWS)

1. Pogled koji nam služi za uvid u broj objava koji je svaki korisnik postavio:

CREATE VIEW vw\_user\_posts\_count AS

SELECT u.id AS user\_id,

u.username,

u.first\_name,

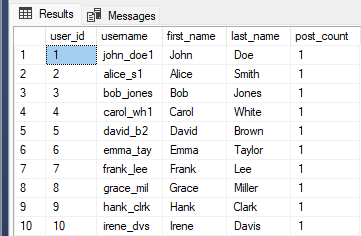
u.last\_name,

COUNT(p.id) AS post\_count

FROM users u

LEFT JOIN posts p ON u.id = p.user\_id

GROUP BY u.id, u.username, u.first\_name, u.last\_name;



Slika 13. pogled za broj objava svakog korisnika

2. Pogled za uvid u sve postavljene fajlove i njihove opise:

CREATE VIEW vw\_message\_pinned\_files AS

SELECT cm.chat\_id,

cm.id AS message\_id,

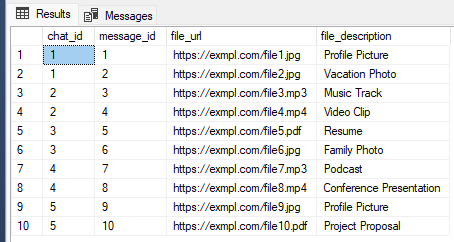
f.path\_url AS file\_url,

f.description AS file\_description

FROM message\_pinned\_files mpf

JOIN chat\_messages cm ON mpf.message\_id = cm.id

JOIN files f ON mpf.file\_id = f.id;



Slika 14. pogled za informacije o fajlovima u ćaskanjima

3. Pogled koji nam prikazuje sve aktivne korisnike:

CREATE VIEW vw\_active\_users AS

SELECT u.id AS user\_id,

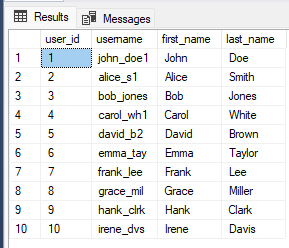
u.username,

u.first\_name,

u.last\_name

FROM users u

WHERE u.is\_active = 1;



Slika 15. pogled koji nam prikazuje sve aktivne korisnike

4. Pogled koji nam prikazuje aktivnost svih korisnika:

CREATE VIEW vw\_most\_active\_users AS

SELECT u.id AS user\_id,

u.username,

COUNT(p.id) AS post\_count,

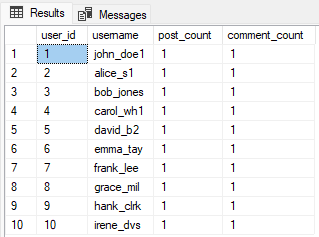
COUNT(pc.id) AS comment\_count

FROM users u

LEFT JOIN posts p ON u.id = p.user\_id

LEFT JOIN post\_comments pc ON p.id = pc.post\_id

GROUP BY u.id, u.username;



Slika 16. pogled za prikaz broja komentara i broja objava od svih korisnika

5. Pogled koji nam daje kategoriju objave na osnovu broja reakcija:

CREATE VIEW vw\_post\_reaction\_category AS

SELECT p.id AS post\_id,

COUNT(pr.id) AS reaction\_count,

CASE

WHEN COUNT(pr.id) < 10 THEN 'Low Reaction'

WHEN COUNT(pr.id) BETWEEN 10 AND 50 THEN 'Medium Reaction'

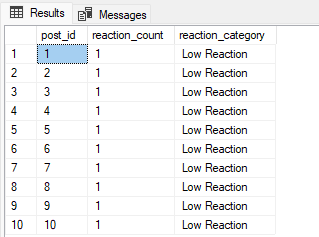
WHEN COUNT(pr.id) > 50 THEN 'High Reaction'

END AS reaction\_category

FROM posts p

LEFT JOIN post\_reactions pr ON p.id = pr.post\_id

GROUP BY p.id;



Slika 17. pogled za kategoriju reakcija

# Okidači (TRIGGERS)

1. Okidač koji dodaje vreme kada je objava ažurirana

CREATE TRIGGER trg\_update\_post\_timestamp

ON posts

AFTER UPDATE

AS

BEGIN

UPDATE posts

SET updated\_at = CURRENT\_TIMESTAMP

WHERE id IN (SELECT id FROM inserted);

END;

UPDATE posts

SET content = 'This is an updated sample post.'

WHERE id = 1;

SELECT id, content, updated\_at FROM posts WHERE id = 1;



Slika 18. promenjen datum nakon UPDATE

2. Okidač koji dodaje vreme kada je komentar ažuriran

CREATE TRIGGER trg\_update\_post\_comment\_timestamp

ON post\_comments

AFTER UPDATE

AS

BEGIN

UPDATE post\_comments

SET updated\_at = CURRENT\_TIMESTAMP

WHERE id IN (SELECT id FROM inserted);

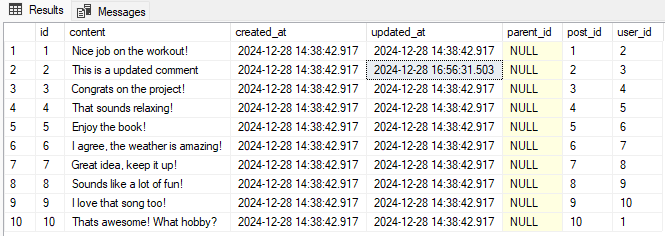
END;

UPDATE post\_comments

SET content = 'This is a updated comment'

WHERE id = 2;

SELECT \* FROM post\_comments;



Slika 19. promenjen datum nakon UPDATE

# Uskladištene funkcije

1. Procedura za kreiranje novog korisnika:

CREATE PROCEDURE CreateUser

@username VARCHAR(100),

@password VARCHAR(255),

@email VARCHAR(255),

@first\_name VARCHAR(50) = '',

@last\_name VARCHAR(50) = '',

@role CHAR(15) = 'user'

AS

BEGIN

INSERT INTO users (username, password, email, first\_name, last\_name, role)

VALUES (@username, @password, @email, @first\_name, @last\_name, @role);

END;

EXEC CreateUser

@username = 'john\_doe',

@password = 'password123',

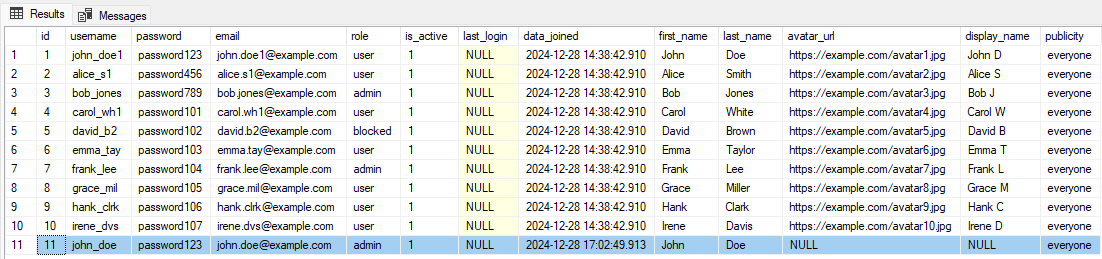
@email = 'john.doe@example.com',

@first\_name = 'John',

@last\_name = 'Doe',

@role = 'admin';

SELECT \* FROM users;



Slika 20. dodat je novi korisnik

2. Procedura za praćenje korisnika:

CREATE PROCEDURE FollowUser

@follower\_user\_id INT,

@followed\_user\_id INT

AS

BEGIN

INSERT INTO followers (follower\_user\_id, followed\_user\_id)

VALUES (@follower\_user\_id, @followed\_user\_id);

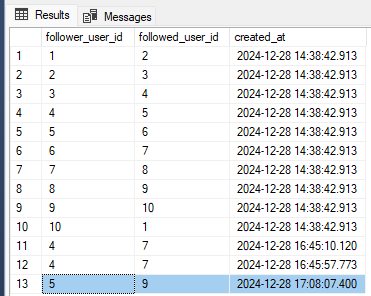
END;

EXEC FollowUser

@follower\_user\_id = 5,

@followed\_user\_id = 9;

SELECT \* FROM followers;



Slika 21. novo praćenje

3. Procedura za slanje poruke:

CREATE PROCEDURE SendMessage

@chat\_id INT,

@user\_id INT,

@content TEXT

AS

BEGIN

INSERT INTO chat\_messages (chat\_id, user\_id, content)

VALUES (@chat\_id, @user\_id, @content);

END;

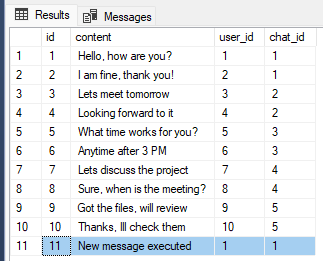
EXEC SendMessage

@chat\_id = 1,

@user\_id = 1,

@content = 'New message executed';

SELECT \* FROM chat\_messages;



Slika 22. nova poruka

# Literatura

[1] DR ŠIMON JANOŠ – Priručnik za vežbe iz baze podataka I, Visoka tehnička škola strukovnih studija u Subotici, 2013.

[2] https://www.programiz.com/sql/left-join