Key Milestone 2: DBMS Lab Project Normalized Relational Schema

# Introduction

This document describes the relation schema in both, denormalized and 3NF normalized form. The schema is written for PostgreSQL as discussed in our tech stack. The reason for going with PostgreSQL is because there are plenty of features that MySQL is not offering and also there are better hosting options available like Neon.

# Schemas

## Denormalized Relational Schema

|  |
| --- |
| -- USERS  CREATE TABLE users (  user\_id SERIAL PRIMARY KEY,  name VARCHAR(100),  email VARCHAR(100) UNIQUE NOT NULL,  password TEXT NOT NULL,  bio TEXT,  follower\_count INT DEFAULT 0,  followed\_count INT DEFAULT 0,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- POSTS  CREATE TABLE posts (  post\_id SERIAL PRIMARY KEY,  user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  title VARCHAR(255),  content TEXT,  likes\_count INT DEFAULT 0,  comment\_count INT DEFAULT 0,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- LIKES  CREATE TABLE likes (  like\_id SERIAL PRIMARY KEY,  user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  liked\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  UNIQUE(user\_id, post\_id)  );  -- COMMENTS  CREATE TABLE comments (  comment\_id SERIAL PRIMARY KEY,  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  content TEXT NOT NULL,  commented\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- FOLLOWS  CREATE TABLE follows (  follower\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  followed\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  followed\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY(follower\_id, followed\_id)  );  -- TAGS  CREATE TABLE tags (  tag\_id SERIAL PRIMARY KEY,  name VARCHAR(50) UNIQUE NOT NULL  );  -- POST\_TAGS  CREATE TABLE post\_tags (  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  tag\_id INT REFERENCES tags(tag\_id) ON DELETE CASCADE,  PRIMARY KEY(post\_id, tag\_id)  );  -- CATEGORIES  CREATE TABLE categories (  category\_id SERIAL PRIMARY KEY,  name VARCHAR(100) UNIQUE NOT NULL,  description TEXT,  posts\_count INT DEFAULT 0  );  -- POST\_CATEGORIES  CREATE TABLE post\_categories (  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  category\_id INT REFERENCES categories(category\_id) ON DELETE CASCADE,  PRIMARY KEY(post\_id, category\_id)  ); |

## Normalized Relational Schema

|  |
| --- |
| -- USERS  CREATE TABLE users (  user\_id SERIAL PRIMARY KEY,  name VARCHAR(100),  email VARCHAR(100) UNIQUE NOT NULL,  password TEXT NOT NULL,  bio TEXT,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- POSTS  CREATE TABLE posts (  post\_id SERIAL PRIMARY KEY,  user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  title VARCHAR(255),  content TEXT,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- LIKES  CREATE TABLE likes (  like\_id SERIAL PRIMARY KEY,  user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  UNIQUE(user\_id, post\_id)  );  -- COMMENTS  CREATE TABLE comments (  comment\_id SERIAL PRIMARY KEY,  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  user\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  content TEXT NOT NULL,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP  );  -- FOLLOWS  CREATE TABLE follows (  follower\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  followed\_id INT REFERENCES users(user\_id) ON DELETE CASCADE,  created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,  PRIMARY KEY(follower\_id, followed\_id)  );  -- TAGS  CREATE TABLE tags (  tag\_id SERIAL PRIMARY KEY,  name VARCHAR(50) UNIQUE NOT NULL  );  -- POST\_TAGS  CREATE TABLE post\_tags (  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  tag\_id INT REFERENCES tags(tag\_id) ON DELETE CASCADE,  PRIMARY KEY(post\_id, tag\_id)  );  -- CATEGORIES  CREATE TABLE categories (  category\_id SERIAL PRIMARY KEY,  name VARCHAR(100) UNIQUE NOT NULL,  description TEXT  );  -- POST\_CATEGORIES  CREATE TABLE post\_categories (  post\_id INT REFERENCES posts(post\_id) ON DELETE CASCADE,  category\_id INT REFERENCES categories(category\_id) ON DELETE CASCADE,  PRIMARY KEY(post\_id, category\_id)  ); |

## Normalization Details

The following fields has been removed as part of the normalization process:

1. **follower\_count** (Users)
2. **followed\_count** (Users)
3. **likes\_count** (Posts)
4. **comment\_count** (Posts)
5. **posts\_count** (Categories)

These fields will be calculated dynamically.

## Additional Enhancements

The following fields are renamed as per best practices and standards:

1. **liked\_at** to **created\_at**
2. **commented\_at** to **created\_at**
3. **followed\_at** to **created\_at**

# References

1. [ChatGPT](chatgpt.com)
2. [Database Normalization – Normal Forms 1nf 2nf 3nf](https://www.bing.com/ck/a?!&&p=1e74e92c075bb3ccf9ccf9e5fac3b87f81b3de987315dbc796599d460bc646fdJmltdHM9MTc0ODIxNzYwMA&ptn=3&ver=2&hsh=4&fclid=1cb96937-e2f0-6c8b-044f-7c71e3a56d75&psq=3NF&u=a1aHR0cHM6Ly93d3cuZnJlZWNvZGVjYW1wLm9yZy9uZXdzL2RhdGFiYXNlLW5vcm1hbGl6YXRpb24tMW5mLTJuZi0zbmYtdGFibGUtZXhhbXBsZXMv&ntb=1)