MARYNA OHINSKA | ZHYTOMYR, UKRAINE | 2025/07/07

Simple Social Network

mohinska

maryna_ohinska

SCHEMA

Core entities and features:

Users

```
id (INTEGER, PRIMARY KEY)
first_name (TEXT)
last_name (TEXT)
username (TEXT)
birthday (DATE)
bio (TEXT)
date_joined (DATETIME)
private (INTEGER, CHECK 0 or 1)
followers_count (INTEGER)
```

Posts

```
id (INTEGER, PRIMARY KEY)
content (TEXT)
user_id (FOREIGN KEY → users.id)
date_created (DATETIME)
location (TEXT)
like_count (INTEGER)
comment_count (INTEGER)
```

Comments

```
id (INTEGER, PRIMARY KEY)
user_id (FOREIGN KEY → users.id)
post_id (FOREIGN KEY → posts.id)
comment (TEXT)
```

Likes

```
user_id (FOREIGN KEY → users.id)
post_id (FOREIGN KEY → posts.id)
PRIMARY KEY(user_id, post_id)
```

Followers

```
user_id (FOREIGN KEY → users.id)
follower_id (FOREIGN KEY → users.id)
PRIMARY KEY(user_id, follower_id)
```

Tags

```
id (INTEGER, PRIMARY KEY)
post_id (FOREIGN KEY → posts.id)
tag (TEXT, NOT NULL)
```

FUNCTIONALITY

The database supports following operations:

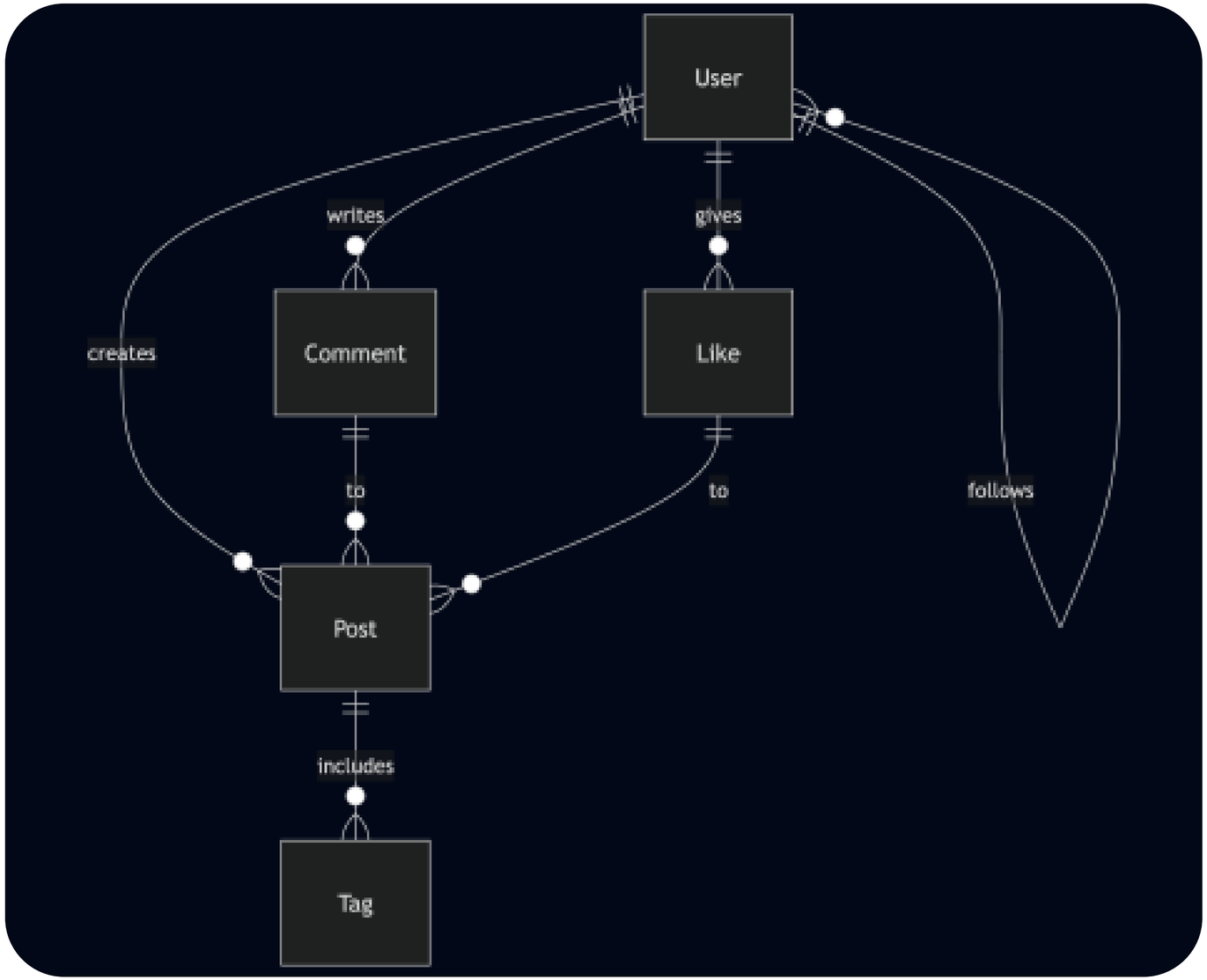
CRUD operations for users, posts, comments, likes, followers, and tags

Cascading deletes, when a user is deleted, all of their content is also removed properly

It handles **many-to-many** relationships

Counts of likes, comments, and followers are automatically updated through **triggers**.

ENTITY RELATIONSHIP DIAGRAM



INDEXES

To optimize common queries, the following indexes are created:

```
`USERS(FIRST_NAME, LAST_NAME)` — SEARCH USERS BY NAME
`USERS(USERNAME)` — LOOKUP USERS BY USERNAME
`POSTS(CONTENT)` — ENABLE SEARCHING POSTS BY TEXT
`POSTS(USER_ID)` — FIND POSTS BY USER
`COMMENTS(POST_ID)` — FIND COMMENTS ON A POST
`TAGS(POST_ID)` — FIND TAGS FOR A POST
```

QUERIES

```sql
-- Find posts created by user
SELECT content, date_created
FROM posts
WHERE user_id = (
    SELECT id
    FROM users
    WHERE first_name = 'Anna'
    AND last_name = 'Melnyk'
)
ORDER BY date_created;
```

```sql
-- Get trending tags
SELECT tag, COUNT(tag) AS mentions
FROM tags
JOIN posts ON posts.id = tags.post_id
WHERE date_created >= datetime('now', '-7 days')
GROUP BY tag
ORDER BY COUNT(tag) DESC
LIMIT 5;
```

```sql
-- Get posts by certain tag
SELECT content
FROM posts
WHERE id IN (
    SELECT post_id
    FROM tags
    WHERE tag = 'hiking'
);
```

```sql
-- Get 10 most popular users
SELECT username, SUM(like_count) +
SUM(comment_count) AS score
FROM users
JOIN posts ON users.id = posts.user_id
GROUP BY users.id
ORDER BY score DESC
LIMIT 10;
```

```sql
-- Editing post
UPDATE posts
SET content = 'Just completed an 11-mile hike — feeling stronger and more grateful for the outdoors!'
WHERE id = 1;
```

```sql
-- Find author of post
SELECT username
FROM users
WHERE id IN (
    SELECT user_id
    FROM posts
    WHERE id = 2
);
```

LIMITATIONS

- No support for post media (images, videos)
- No support for private messaging or group interactions
- Scalability may be limited by denormalized counts and triggers in very large datasets
- User roles, or content moderation could be added

CS50 SQL

Thank you!