Projeto de Rede Social - Banco de Dados

Lucas Antônio Marcão - a2134616
Juliano Sangaleti
Guilherme Serbai - a2551802

Campo Mourão, 01 de junho de 2025

Resumo

Este documento apresenta o projeto de um banco de dados relacional para uma aplicação de rede social. O modelo foi normalizado até a Terceira Forma Normal (3FN) para garantir a integridade dos dados, reduzir redundâncias e evitar anomalias de inserção, atualização e exclusão. São detalhadas as dependências funcionais, o processo de normalização e os scripts SQL (DDL e DML) para a criação e povoamento da estrutura.

Sumário

1 Modelo Relacional (MR)

O modelo relacional final é composto por cinco tabelas principais, projetadas para armazenar informações sobre usuários, postagens, comentários, curtidas e seguidores. As chaves primárias (PK) e estrangeiras (FK) são indicadas abaixo.

- **users**(id_user (PK), email, senha_hash, slug, is_admin, nome_user, data_insercao)
- **postagem**(id_postagem (PK), *id_user (FK)*, anuncio, texto_post, atributos_json, data_insercao, data_alteracao)
- **comentario**(id_comentario (PK), *id_postagem (FK)*, *id_user (FK)*, texto_comentario, data_insercao, data_alteracao)
- **curtida**(id_curtida (PK), *id_user (FK)*, *id_comentario (FK)*, *id_postagem (FK)*, data_insercao)
- **seguidor**(id_seguidor (PK), *id_user (FK)*, *id_seguidor_user (FK)*, data_insercao)

2 Dependências Funcionais (DF)

As dependências funcionais definem as relações entre os atributos de cada tabela. A seguir, especificamos as DFs para cada relação do modelo, onde a seta ($\rightarrow$) indica que o determinante (lado esquerdo) define funcionalmente o determinado (lado direito).

2.1 Tabela `users`

O determinante é a chave primária `id_user`. Os atributos `email` e `slug` também são determinantes, pois possuem a constraint `UNIQUE`.

- DF1: {id_user} $\rightarrow$ {email, senha_hash, slug, is_admin, nome_user, data_insercao}
- DF2: {email} $\rightarrow$ {id_user, senha_hash, slug, is_admin, nome_user, data_insercao}
- DF3: {slug} $\rightarrow$ {id_user, email, senha_hash, is_admin, nome_user, data_insercao}

2.2 Tabela `postagem`

- DF4: {id_postagem} $\rightarrow$ {id_user, anuncio, texto_post, atributos_json, data_insercao, data_alteracao}

2.3 Tabela `comentario`

- DF5: {id_comentario} $\rightarrow$ {id_postagem, id_user, texto_comentario, data_insercao, data_alteracao}

2.4 Tabela `curtida`

- DF6: {id_curtida} $\rightarrow$ {id_user, id_comentario, id_postagem, data_insercao}

2.5 Tabela `seguidor`

A chave primária é `id_seguidor`, mas a combinação de `id_user` e `id_seguidor_user` é única.

- DF7: {id_seguidor} → {id_user, id_seguidor_user, data_insercao}
- DF8: {id_user, id_seguidor_user} → {id_seguidor, data_insercao}

3 Análise da Normalização

O banco de dados foi projetado para estar em conformidade com a Terceira Forma Normal (3FN).

3.1 Primeira Forma Normal (1FN)

Uma tabela está na 1FN se todos os seus atributos forem atômicos, ou seja, não contêm grupos de repetição ou valores múltiplos.

- **Verificação:** Todas as tabelas do esquema satisfazem a 1FN. Cada coluna armazena um único valor por registro. O campo `atributos_json` na tabela `postagem`, embora armazene dados estruturados, é tratado pelo SGBD como um valor atômico único, sendo uma prática moderna e aceita.

3.2 Segunda Forma Normal (2FN)

Uma tabela está na 2FN se estiver na 1FN e se todos os seus atributos não-chave forem totalmente dependentes da chave primária completa.

- **Verificação:** Todas as tabelas cujas chaves primárias são compostas por uma única coluna (`users`, `postagem`, `comentario`, `curtida`, `seguidor`) atendem trivialmente à 2FN. A tabela `seguidor` possui uma chave candidata composta (`id_user`, `id_seguidor_user`), mas o único atributo não-chave (`data_insercao`) depende de toda a chave composta, não de apenas parte dela. Portanto, o esquema está na 2FN.

3.3 Terceira Forma Normal (3FN)

Uma tabela está na 3FN se estiver na 2FN e se todos os seus atributos não dependerem transitivamente da chave primária. Ou seja, nenhum atributo não-chave depende de outro atributo não-chave.

- **Verificação:** Analisando as tabelas, não existem dependências transitivas. Por exemplo, na tabela `users`, `nome_user` depende de `id_user`, e não de `email`. Em `postagem`, `texto_post` depende de `id_postagem`, e não de `id_user`. Esta lógica se aplica a todas as tabelas, garantindo que o modelo está na 3FN.

4 Script SQL de Criação (DDL)

O script a seguir cria a estrutura do banco de dados e suas tabelas, incluindo chaves, constraints e índices para otimização de performance.

```sql
-- Configuração do banco de dados com locale mais portável
CREATE DATABASE redesocialbd
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'pt_BR.UTF-8'
    LC_CTYPE = 'pt_BR.UTF-8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;

-- Tabela de usuários com tipo de senha corrigido para armazenar hash
CREATE TABLE IF NOT EXISTS users (
  id_user SERIAL PRIMARY KEY,
  email VARCHAR(150) NOT NULL UNIQUE,
  senha_hash VARCHAR(255) NOT NULL,
  slug VARCHAR(100) NOT NULL UNIQUE,
  is_admin BOOLEAN NOT NULL DEFAULT FALSE,
  nome_user VARCHAR(100),
  data_insercao TIMESTAMP WITHOUT TIME ZONE NOT NULL DEFAULT now()
);

-- Tabela de postagens
CREATE TABLE IF NOT EXISTS postagem (
  id_postagem BIGSERIAL PRIMARY KEY,
  id_user INTEGER NOT NULL,
  anuncio BOOLEAN NOT NULL DEFAULT FALSE,
  texto_post TEXT NOT NULL,
  atributos_json JSONB NULL,
  data_insercao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  data_alteracao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_postagem_users
    FOREIGN KEY (id_user) REFERENCES users(id_user)
      ON DELETE CASCADE ON UPDATE CASCADE
);

-- Tabela de comentários
CREATE TABLE IF NOT EXISTS comentario (
  id_comentario BIGSERIAL PRIMARY KEY,
  id_postagem BIGINT NOT NULL,
  id_user INTEGER NOT NULL,
  texto_comentario TEXT NOT NULL,
  data_insercao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  data_alteracao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_comentario_postagem
    FOREIGN KEY (id_postagem) REFERENCES postagem(id_postagem)
      ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_comentario_users
    FOREIGN KEY (id_user) REFERENCES users(id_user)
      ON DELETE CASCADE ON UPDATE CASCADE
);

-- Tabela de curtidas
CREATE TABLE IF NOT EXISTS curtida (
  id_curtida BIGSERIAL PRIMARY KEY,
  id_user INTEGER NOT NULL,
  id_comentario BIGINT NULL,
```

```
  id_postagem BIGINT NULL,
  data_insercao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_curtida_users
    FOREIGN KEY (id_user) REFERENCES users(id_user)
      ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_curtida_comentario
    FOREIGN KEY (id_comentario) REFERENCES comentario(id_comentario)
      ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_curtida_postagem
    FOREIGN KEY (id_postagem) REFERENCES postagem(id_postagem)
      ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT chk_curtida_target
    CHECK (
      (id_comentario IS NOT NULL AND id_postagem IS NULL) OR
      (id_comentario IS NULL AND id_postagem IS NOT NULL)
    )
);

-- Tabela de seguidores
CREATE TABLE IF NOT EXISTS seguidor (
  id_seguidor BIGSERIAL PRIMARY KEY,
  id_user INTEGER NOT NULL, -- O usuário que está sendo seguido
  id_seguidor_user INTEGER NOT NULL, -- O usuário que está seguindo
  data_insercao TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_seguidor_user_being_followed
    FOREIGN KEY (id_user) REFERENCES users(id_user)
      ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_seguidor_user_who_is_following
    FOREIGN KEY (id_seguidor_user) REFERENCES users(id_user)
      ON DELETE CASCADE ON UPDATE CASCADE,
  UNIQUE (id_user, id_seguidor_user),
  CONSTRAINT chk_seguidor_not_self
    CHECK (id_user <> id_seguidor_user)
);

-- Índices para otimização de performance
CREATE INDEX idx_postagem_id_user ON postagem(id_user);
CREATE INDEX idx_comentario_id_postagem ON comentario(id_postagem);
CREATE INDEX idx_comentario_id_user ON comentario(id_user);
CREATE INDEX idx_curtida_id_user ON curtida(id_user);
CREATE INDEX idx_seguidor_id_user ON seguidor(id_user);
CREATE INDEX idx_seguidor_id_seguidor_user ON seguidor(id_seguidor_user);
```

Listing 1: Script DDL para criação das tabelas.

5 Script SQL de População (DML)

O script a seguir insere dados de exemplo nas tabelas para demonstrar o funcionamento do banco de dados.

```sql
-- Inserir usuários (a senha_hash seria gerada por uma lib como bcrypt)
INSERT INTO users (email, senha_hash, slug, nome_user) VALUES
('ana.silva@email.com', '$2b$12$abcdefghijklmnopqrstuvwxyABC', 'ana-silva', 'Ana Silva'),
('bruno.costa@email.com', '$2b$12$abcdefghijklmnopqrstuvwxyDEF', 'bruno-costa', 'Bruno Costa'),
('carla.dias@email.com', '$2b$12$abcdefghijklmnopqrstuvwxyGHI', 'carla-dias', 'Carla Dias');

-- Inserir postagens
INSERT INTO postagem (id_user, texto_post) VALUES
(1, 'Olá, mundo! Esta é minha primeira postagem na rede.'),
(2, 'Acabei de ler um livro incrível sobre PostgreSQL. Recomendo!'),
(1, 'Que dia lindo para um passeio no parque!');

-- Inserir comentários
-- Ana (id 1) comenta no post de Bruno (id 2)
INSERT INTO comentario (id_postagem, id_user, texto_comentario) VALUES
(2, 1, 'Qual o nome do livro? Fiquei interessada!'),
-- Carla (id 3) também comenta no post de Bruno (id 2)
(2, 3, 'Também quero saber! Adoro bancos de dados.'),
-- Bruno (id 2) comenta no post de Ana (id 3)
(3, 2, 'Aproveite o dia, Ana!');

-- Inserir curtidas
-- Bruno (id 2) curte o post 1 (de Ana)
INSERT INTO curtida (id_user, id_postagem) VALUES (2, 1);
-- Ana (id 1) curte o post 2 (de Bruno)
INSERT INTO curtida (id_user, id_postagem) VALUES (1, 2);
-- Carla (id 3) curte o comentário 1 (de Ana no post de Bruno)
INSERT INTO curtida (id_user, id_comentario) VALUES (3, 1);

-- Inserir seguidores
-- Ana (id 1) segue Bruno (id 2)
INSERT INTO seguidor (id_user, id_seguidor_user) VALUES (2, 1);
-- Bruno (id 2) segue Ana (id 1)
INSERT INTO seguidor (id_user, id_seguidor_user) VALUES (1, 2);
-- Carla (id 3) segue Ana (id 1)
INSERT INTO seguidor (id_user, id_seguidor_user) VALUES (1, 3);
```

Listing 2: Script DML para povoamento das tabelas.