Relatório de Modelagem de Banco de Dados

1. Introdução

Esse relatório traz o modelo de um banco de dados feito para uma rede social fictícia. Aqui, é explicado como as tabelas foram criadas e como elas se conectam para atender os requisitos da aplicação.

Relacionamentos:

1. **Usuário cria Postagens** (1:N):
   Um usuário pode criar várias postagens, mas cada postagem pertence a apenas um usuário.
2. **Usuário comenta Postagens** (1:N):
   Um usuário pode comentar várias postagens, e cada comentário pertence a uma postagem.
3. **Usuário recebe Notificações** (1:N):
   Cada notificação é destinada a um usuário, mas um usuário pode ter várias notificações.
4. **Usuário participa de Grupos** (N:M):
   Um grupo pode ter vários usuários, e um usuário pode fazer parte de vários grupos.
5. **Usuário se conecta com outros Usuários** (N:M):
   Um usuário pode ter conexões com vários outros usuários.
6. **Usuário troca Mensagens Privadas** (N:M):
   Cada mensagem envolve dois usuários: remetente e destinatário.
7. **Usuário é associado a Tags** (N:M):
   Usuários podem ter várias tags associadas a eles, e cada tag pode ser vinculada a vários usuários.

2. Análise e Decisões de Modelagem

Usuários

Cada usuário tem informações básicas como nome, email, data de nascimento e foto de perfil. Além disso, eles podem criar conexões e fazer postagens e comentários.

Conexões

São os relacionamentos entre usuários. Todo mundo pode ver as postagens, já que não existem perfis privados aqui.

Postagens e Comentários

Os usuários criam postagens e os outros podem comentar nelas. Os comentários também podem ter respostas, como um tipo de conversa.

Notificações

Notificações avisam os usuários quando alguém interage com suas postagens ou comentários. Elas guardam o tipo de interação e a data.

Grupos e Membros

Os grupos são como comunidades temáticas, e os usuários podem ser membros ou administradores. Só os administradores podem apagar mensagens.

Mensagens Privadas

Os usuários podem trocar mensagens diretas. A gente guarda o histórico, quem enviou, quem recebeu e o status (enviada, lida, etc.).

Tags

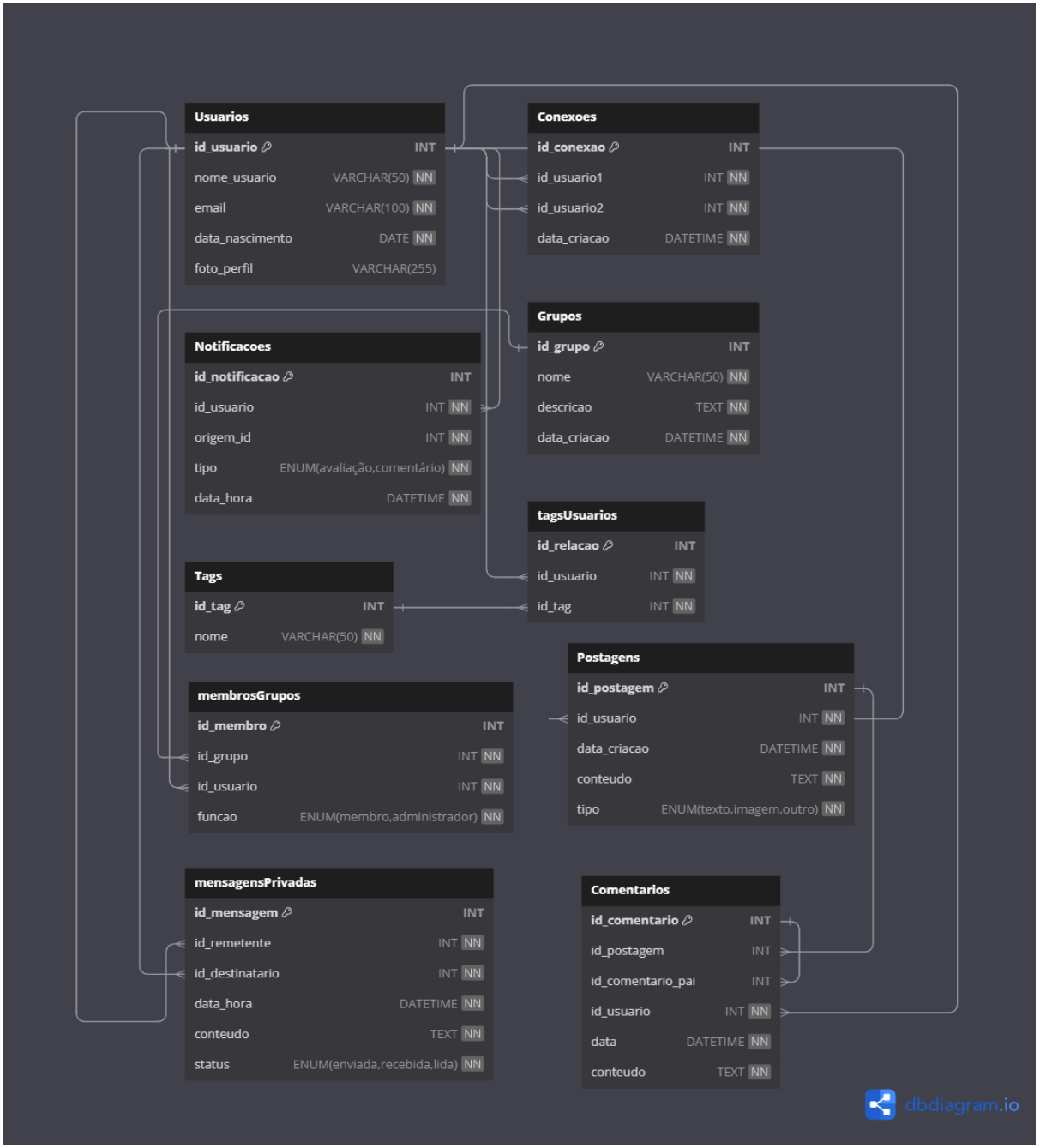
Os usuários podem ter até 5 tags que ajudam a conectar pessoas com interesses parecidos. Essas tags também podem ser criadas pelos próprios usuários.

3. Modelos Criados
Modelo Conceitual

- **Usuários**
  - ID (PK)
  - Nome de Usuário (único)
  - Email
  - Data de Nascimento
  - Foto de Perfil
- **Conexões**
  - ID (PK)
  - Usuário1 (FK)
  - Usuário2 (FK)
  - Data de Criação
- **Postagens**
  - ID (PK)
  - Usuário (FK)
  - Data de Criação
  - Conteúdo
  - Tipo
- **Comentários**
  - ID (PK)
  - Postagem (FK, opcional)
  - Comentário Pai (FK, opcional)
  - Usuário (FK)
  - Data
  - Conteúdo
- **Notificações**
  - ID (PK)
  - Usuário (FK)

    - Origem (FK para Postagem ou Comentário)
    - Tipo
    - Data e Hora
- **Grupos**
    - ID (PK)
    - Nome (único)
    - Descrição
    - Data de Criação
- **Mensagens Privadas**
    - ID (PK)
    - Remetente (FK)
    - Destinatário (FK)
    - Data e Hora
    - Conteúdo
    - Status
- **Tags**
    - ID (PK)
    - Nome (único)

3.2. Modelo Lógico



3.3. Modelo Físico

SCRIPT PARA CRIAÇÃO DAS TABELAS

```
CREATE TABLE Usuarios (
  id_usuario INT AUTO_INCREMENT PRIMARY KEY,
  nome_usuario VARCHAR(50) UNIQUE NOT NULL,
  email VARCHAR(100) UNIQUE NOT NULL,
  data_nascimento DATE NOT NULL,
  foto_perfil VARCHAR(255)
);
```

```
CREATE TABLE Conexoes (
  id_conexao INT AUTO_INCREMENT PRIMARY KEY,
  id_usuario1 INT NOT NULL,
  id_usuario2 INT NOT NULL,
  data_criacao DATETIME NOT NULL,
  FOREIGN KEY (id_usuario1) REFERENCES Usuarios(id_usuario),
  FOREIGN KEY (id_usuario2) REFERENCES Usuarios(id_usuario)
);

CREATE TABLE Posts (
  id_postagem INT AUTO_INCREMENT PRIMARY KEY,
  id_usuario INT NOT NULL,
  data_criacao DATETIME NOT NULL,
  conteudo TEXT NOT NULL,
  tipo ENUM('texto', 'imagem', 'outro') NOT NULL,
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
);

CREATE TABLE Comentarios (
  id_comentario INT AUTO_INCREMENT PRIMARY KEY,
  id_postagem INT,
  id_comentario_pai INT,
  id_usuario INT NOT NULL,
  data DATETIME NOT NULL,
  conteudo TEXT NOT NULL,
  FOREIGN KEY (id_postagem) REFERENCES Posts(id_postagem),
  FOREIGN KEY (id_comentario_pai) REFERENCES Comentarios(id_comentario),
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
);

CREATE TABLE Notificacoes (
  id_notificacao INT AUTO_INCREMENT PRIMARY KEY,
  id_usuario INT NOT NULL,
  origem_id INT NOT NULL,
  tipo ENUM('avaliação', 'comentário') NOT NULL,
  data_hora DATETIME NOT NULL,
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
);

CREATE TABLE Grupos (
  id_grupo INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(50) UNIQUE NOT NULL,
  descricao TEXT NOT NULL,
  data_criacao DATETIME NOT NULL
);

CREATE TABLE membrosGrupos (
  id_membro INT AUTO_INCREMENT PRIMARY KEY,
```

```
  id_grupo INT NOT NULL,
  id_usuario INT NOT NULL,
  funcao ENUM('membro', 'administrador') NOT NULL,
  FOREIGN KEY (id_grupo) REFERENCES Grupos(id_grupo),
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario)
);

CREATE TABLE mensagensPrivadas (
  id_mensagem INT AUTO_INCREMENT PRIMARY KEY,
  id_remetente INT NOT NULL,
  id_destinatario INT NOT NULL,
  data_hora DATETIME NOT NULL,
  conteudo TEXT NOT NULL,
  status ENUM('enviada', 'recebida', 'lida') NOT NULL,
  FOREIGN KEY (id_remetente) REFERENCES Usuarios(id_usuario),
  FOREIGN KEY (id_destinatario) REFERENCES Usuarios(id_usuario)
);

CREATE TABLE Tags (
  id_tag INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(50) UNIQUE NOT NULL
);

CREATE TABLE tagsUsuarios (
  id_relacao INT AUTO_INCREMENT PRIMARY KEY,
  id_usuario INT NOT NULL,
  id_tag INT NOT NULL,
  FOREIGN KEY (id_usuario) REFERENCES Usuarios(id_usuario),
  FOREIGN KEY (id_tag) REFERENCES Tags(id_tag)
);
```

SCRIPT PARA INSERÇÃO DOS DADOS

```
INSERT INTO Usuarios (nome_usuario, email, data_nascimento, foto_perfil)
VALUES
('joao123', 'joao123@email.com', '1990-05-15', 'foto1.png'),
('maria456', 'maria456@email.com', '1985-03-20', 'foto2.png'),
('carlos789', 'carlos789@email.com', '2000-10-05', 'foto3.png'),
('ana321', 'ana321@email.com', '1995-07-12', 'foto4.png');

-- Inserir conexões
INSERT INTO Conexoes (id_usuario1, id_usuario2, data_criacao)
VALUES
(1, 2, NOW()),
(2, 3, NOW()),
(3, 4, NOW()),
(4, 1, NOW());
```

```
-- Inserir postagens
INSERT INTO Posts (id_usuario, data_criacao, conteudo, tipo)
VALUES
(1, NOW(), 'Minha primeira postagem!', 'texto'),
(2, NOW(), 'Adoro essa imagem!', 'imagem'),
(3, NOW(), 'Alguém viu esse evento?', 'texto'),
(4, NOW(), 'Confira este vídeo incrível!', 'outro');

-- Inserir comentários
INSERT INTO Comentarios (id_postagem, id_usuario, data, conteudo)
VALUES
(1, 2, NOW(), 'Parabéns pela postagem!'),
(1, 3, NOW(), 'Muito interessante!'),
(2, 1, NOW(), 'Linda imagem!'),
(3, 4, NOW(), 'Muito relevante.');

-- Inserir notificações
INSERT INTO Notificacoes (id_usuario, origem_id, tipo, data_hora)
VALUES
(1, 1, 'comentário', NOW()),
(2, 3, 'comentário', NOW()),
(3, 2, 'avaliação', NOW()),
(4, 4, 'avaliação', NOW());

-- Inserir grupos
INSERT INTO Grupos (nome, descricao, data_criacao)
VALUES
('Fotografia', 'Grupo para amantes de fotografia', NOW()),
('Esportes', 'Discussão sobre esportes', NOW()),
('Tecnologia', 'Grupo de tecnologia', NOW());

-- Inserir membros em grupos
INSERT INTO membrosGrupos (id_grupo, id_usuario, funcao)
VALUES
(1, 1, 'administrador'),
(1, 2, 'membro'),
(2, 3, 'administrador'),
(3, 4, 'administrador');

-- Inserir mensagens privadas
INSERT INTO mensagensPrivadas (id_remetente, id_destinatario, data_hora, conteudo,
status)
VALUES
(1, 2, NOW(), 'Olá, tudo bem?', 'enviada'),
(2, 3, NOW(), 'Oi, como vai?', 'lida'),
(3, 4, NOW(), 'Você viu o evento?', 'recebida'),
(4, 1, NOW(), 'Sim, vi sim.', 'lida');
```

```
-- Inserir tags
INSERT INTO Tags (nome)
VALUES
('Fotografia'),
('Esportes'),
('Tecnologia'),
('Cinema'),
('Viagens');

-- Relacionar usuários a tags
INSERT INTO tagsUsuarios (id_usuario, id_tag)
VALUES
(1, 1),
(2, 2),
(3, 3),
(4, 4),
(1, 5);
```

SCRIPT DAS CONSULTAS

```
-- Listar todas as postagens de um usuário específico
SELECT * FROM Posts WHERE id_usuario = 1;

-- Ver comentários em uma postagem
SELECT c.conteudo, u.nome_usuario
FROM Comentarios c
JOIN Usuarios u ON c.id_usuario = u.id_usuario
WHERE c.id_postagem = 1;

-- Ver notificações de um usuário
SELECT n.tipo, n.data_hora, u.nome_usuario AS origem
FROM Notificacoes n
JOIN Usuarios u ON n.origem_id = u.id_usuario
WHERE n.id_usuario = 1;

-- Ver membros de um grupo
SELECT m.funcao, u.nome_usuario
FROM membrosGrupos m
JOIN Usuarios u ON m.id_usuario = u.id_usuario
WHERE m.id_grupo = 1;

-- Ver mensagens trocadas entre dois usuários
SELECT mp.conteudo, mp.data_hora, mp.status
FROM mensagensPrivadas mp
WHERE (mp.id_remetente = 1 AND mp.id_destinatario = 2)
   OR (mp.id_remetente = 2 AND mp.id_destinatario = 1);
```

SCRIPT DA TRIGGER

```
DELIMITER $$

CREATE TRIGGER triggerGamificacaoComentarios
AFTER INSERT ON Comentarios
FOR EACH ROW
BEGIN
   -- Incrementa o contador de comentários do usuário
   UPDATE Usuarios
   SET contador_comentarios = IFNULL(contador_comentarios, 0) + 1
   WHERE id_usuario = NEW.id_usuario;

   -- Verifica o número de comentários do usuário
   DECLARE num_comentarios INT;
   SELECT contador_comentarios INTO num_comentarios
   FROM Usuarios
   WHERE id_usuario = NEW.id_usuario;

   -- Se o usuário alcançou múltiplos de 10 comentários, insere uma notificação
   IF num_comentarios MOD 10 = 0 THEN
      DECLARE nivel INT;
      SET nivel = num_comentarios / 10;

      INSERT INTO Notificacoes (id_usuario, origem_id, tipo, data_hora)
      VALUES (
         NEW.id_usuario,
         NULL,
         CONCAT('Parabéns! Você alcançou o nível ', nivel, ' de comentarista!'),
         NOW()
      );
   END IF;
END$$

DELIMITER ;
```