

**Pró-Reitoria Acadêmica
Curso de Ciência da Computação
Trabalho Laboratório de Banco de Dados**

MODELAGEM DE BANCO DE DADOS

**Autor: Breno Santana Silva
Célio L F da Silva
Igor Viana
Kelvin Rodrigues
Orientador: Prof. João Robson**

SUMÁRIO

INTRODUÇÃO	2
OBJETIVO	2
SCRIPTS	6
Criação do Banco	6
Criação dos insert's	9
Criação das Consultas	13
Criação de Trigger's e Procedures	15
Trigger para Verificação de nome_usuario único	15
Procedure para Recuperar nome_usuario	16
Procedure para Recuperar a Senha	18
Procedure Notifica Mensagem Recebida	20
Trigger VerificarConteudoPostagem	21

INTRODUÇÃO

Este relatório detalha a modelagem do banco de dados para uma aplicação de rede social, abrangendo o propósito, justificativas para as escolhas de design e uma descrição detalhada das entidades. A modelagem as especificações fornecidas e orientações da disciplina em sala de aula.

OBJETIVO

O banco de dados foi projetado para suportar funcionalidades comuns em redes sociais, como postagens, interações (likes, dislikes), conexões entre usuários, comentários, notificações, gerenciamento de grupos, mensagens privadas, e categorização de interesses por meio de tags. A estrutura considera:

1. **Flexibilidade:** Para permitir a adição de novas funcionalidades no futuro.
2. **Normalização:** Para evitar redundância de dados e facilitar a manutenção.
3. **Escalabilidade:** Para suportar grandes volumes de dados e consultas frequentes.

Funcionalidades:

1. **Cadastro de Usuários:** Cada usuário terá informações básicas, **contendo, no mínimo, nome de usuário (deve ser único), e-mail, data de nascimento e foto de perfil.**
2. **Conexões:** Os usuários poderão se relacionar por meio de conexões. Usuários não conectados também devem ser capazes de visualizar as postagens de qualquer usuário, ou seja, **não devem existir perfis privados nesta rede social.**
3. **Postagens e Interações:** Os usuários poderão criar postagens, e essas postagens podem receber avaliações positivas ou negativas e comentários de outros usuários. **Cada comentário também pode receber comentários e avaliações. As postagens devem ter no mínimo data de criação, conteúdo e tipo (texto, imagem, etc.).**

4. Notificações: deve ser possível notificar o usuário quando há avaliações ou comentários em uma postagem ou comentário. Essas notificações devem ser armazenadas de alguma forma, contendo dia e horário e sua origem, no mínimo.

5. Grupos e Comunidades: Os usuários podem criar e participar de grupos temáticos. Cada grupo terá um nome, que deve ser único, descrição, data de criação, e uma lista de membros com funções específicas (membro, administrador). No caso de grupos, apenas administradores devem ser capazes de apagar mensagens de membros.

6. Mensagens Privadas: Os usuários também poderão trocar mensagens privadas entre si. Esse recurso deve ser representado de forma a permitir o armazenamento de histórico de conversas e o controle de status da mensagem (enviada, recebida, lida).

7. Tags: Os usuários podem designar *tags* (rótulos) para si, permitindo encontrar usuários com gostos e interesses similares. *Tags* novas podem ser criadas por um usuário, mas deve haver um limite de 5 *tags* atribuídas por usuário.

Dessa forma foi realizado o levantamento sobre as entidade e relacionamentos. A modelagem foi feita utilizando o **modelo Entidade-Relacionamento (ER)**, resultando em um banco de dados relacional com tabelas bem definidas. Abaixo estão descritas as principais entidades e seus relacionamentos.

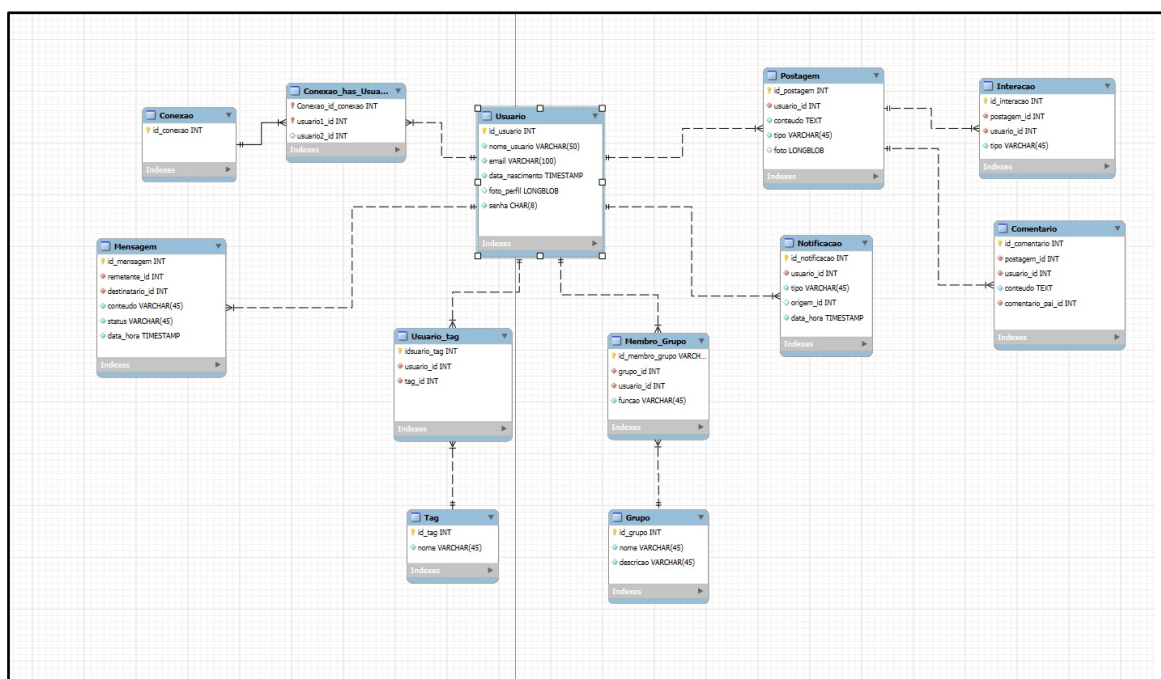
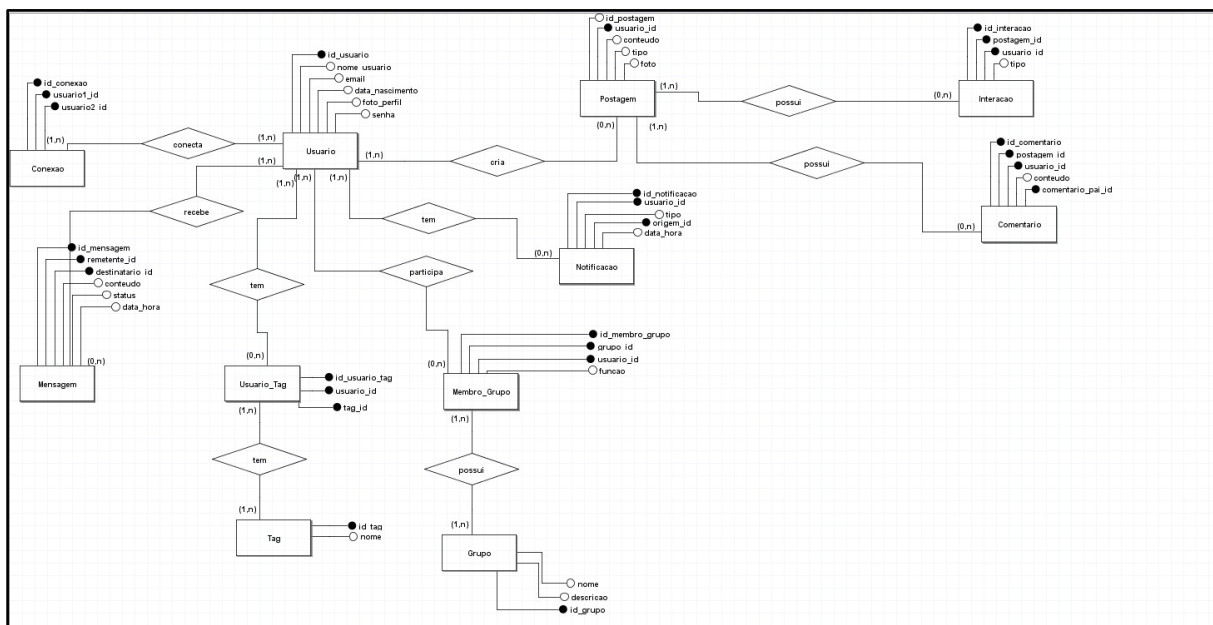
Entidades e Relacionamentos:

1. Usuario:

- Atributos: id_usuario (PK), nome_usuario, email, data_nascimento, foto_perfil, senha.
- Relacionamentos:
 - Relacionado com Conexao (auto-relacionamento) — **Muitos-para-Muitos**.
 - Relacionado com Postagem — **1:N** (Um usuário pode criar várias postagens).
 - Relacionado com Mensagem — **1:N** (Um usuário pode enviar ou receber várias mensagens).
 - Relacionado com Notificacao — **1:N** (Um usuário pode ter várias notificações).
 - Relacionado com Membro_Grupo — **1:N** (Um usuário pode participar de vários grupos).
 - Relacionado com Usuario_Tag — **1:N** (Um usuário pode ter até 5 tags).

2. Conexao:

- Atributos: id_conexao (PK), usuario1_id (FK), usuario2_id (FK).
- 3. **Postagem:**
 - Atributos: id_postagem (PK), usuario_id (FK), conteudo, tipo, foto.
 - Relacionamentos:
 - Relacionado com Interacao — **1:N** (Uma postagem pode ter várias interações).
 - Relacionado com Comentario — **1:N** (Uma postagem pode ter vários comentários).
- 4. **Interacao:**
 - Atributos: id_interacao (PK), postagem_id (FK), usuario_id (FK), tipo.
- 5. **Comentario:**
 - Atributos: id_comentario (PK), postagem_id (FK), usuario_id (FK), conteudo, comentario_pai_id (FK).
 - Relacionamentos:
 - Relacionado com si mesmo para comentários aninhados (auto-relacionamento).
- 6. **Notificacao:**
 - Atributos: id_notificacao (PK), usuario_id (FK), tipo, origem_id, data_hora.
- 7. **Grupo:**
 - Atributos: id_grupo (PK), nome, descricao.
 - Relacionamentos:
 - Relacionado com Membro_Grupo — **1:N** (Um grupo pode ter vários membros).
- 8. **Membro_Grupo:**
 - Atributos: id_membro_grupo (PK), grupo_id (FK), usuario_id (FK), funcao.
- 9. **Mensagem:**
 - Atributos: id_mensagem (PK), remetente_id (FK), destinatario_id (FK), conteudo, status, data_hora.
- 10. **Tag:**
 - Atributos: id_tag (PK), nome.
 - Relacionamentos:
 - Relacionado com Usuario_Tag — **1:N** (Uma tag pode ser atribuída a vários usuários).
- 11. **Usuario_Tag:**
 - Atributos: id_usuario_tag (PK), usuario_id (FK), tag_id (FK).



SCRIPTS

Criação do Banco

```
-- Criação do banco de dados

-- DROP DATABASE rede_social;

CREATE DATABASE rede_social;
USE rede_social;

-- Tabela Usuario
CREATE TABLE Usuario (
    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 LONGBLOB, -- Foto armazenada como dado binário
    data_cadastro TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela Conexao
CREATE TABLE Conexao (
    id_conexao INT AUTO_INCREMENT PRIMARY KEY,
    usuario1_id INT NOT NULL,
    usuario2_id INT NOT NULL,
    data_conexao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario1_id) REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
    FOREIGN KEY (usuario2_id) REFERENCES Usuario(id_usuario) ON DELETE CASCADE,
    CONSTRAINT conexao_unica UNIQUE (usuario1_id, usuario2_id)
);

-- Tabela Postagem
CREATE TABLE Postagem (
    id_postagem INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    conteudo TEXT NOT NULL,
    tipo ENUM('texto', 'imagem', 'video') NOT NULL,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id_usuario) ON DELETE CASCADE
);

-- Tabela Interacao
CREATE TABLE Interacao (
```

```

    id_interacao INT AUTO_INCREMENT PRIMARY KEY,
    postagem_id INT NOT NULL,
    usuario_id INT NOT NULL,
    tipo ENUM('like', 'dislike') NOT NULL,
    data_interacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (postagem_id) REFERENCES Postagem(id_postagem) ON
DELETE CASCADE,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE
);

-- Tabela Comentario
CREATE TABLE Comentario (
    id_comentario INT AUTO_INCREMENT PRIMARY KEY,
    postagem_id INT NOT NULL,
    usuario_id INT NOT NULL,
    conteudo TEXT NOT NULL,
    data_comentario TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    comentario_pai_id INT DEFAULT NULL,
    FOREIGN KEY (postagem_id) REFERENCES Postagem(id_postagem) ON
DELETE CASCADE,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE,
    FOREIGN KEY (comentario_pai_id) REFERENCES
Comentario(id_comentario) ON DELETE CASCADE
);

-- Tabela Notificacao
CREATE TABLE Notificacao (
    id_notificacao INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    tipo ENUM('avaliacao', 'comentario') NOT NULL,
    origem_id INT NOT NULL,
    data_notificacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE
);

-- Tabela Grupo
CREATE TABLE Grupo (
    id_grupo INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) UNIQUE NOT NULL,
    descricao TEXT,
    data_criacao TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabela Membro_Grupo
CREATE TABLE Membro_Grupo (
    id_membro_grupo INT AUTO_INCREMENT PRIMARY KEY,

```



```

    grupo_id INT NOT NULL,
    usuario_id INT NOT NULL,
    funcao ENUM('membro', 'administrador') NOT NULL,
    FOREIGN KEY (grupo_id) REFERENCES Grupo(id_grupo) ON DELETE
CASCADE,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE
);

-- Tabela Mensagem
CREATE TABLE Mensagem (
    id_mensagem INT AUTO_INCREMENT PRIMARY KEY,
    remetente_id INT NOT NULL,
    destinatario_id INT NOT NULL,
    conteudo TEXT NOT NULL,
    data_envio TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    status ENUM('enviada', 'recebida', 'lida') NOT NULL,
    FOREIGN KEY (remetente_id) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE,
    FOREIGN KEY (destinatario_id) REFERENCES Usuario(id_usuario) ON
DELETE CASCADE
);

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

-- Tabela Usuario_Tag
CREATE TABLE Usuario_Tag (
    id_usuario_tag INT AUTO_INCREMENT PRIMARY KEY,
    usuario_id INT NOT NULL,
    tag_id INT NOT NULL,
    FOREIGN KEY (usuario_id) REFERENCES Usuario(id_usuario) ON DELETE
CASCADE,
    FOREIGN KEY (tag_id) REFERENCES Tag(id_tag) ON DELETE CASCADE
);

-- Adicionando o campo 'senha' na tabela 'Usuario'
ALTER TABLE Usuario ADD COLUMN senha CHAR(8) NOT NULL;

-- Garantindo que a senha tenha exatamente 8 caracteres
ALTER TABLE Usuario ADD CONSTRAINT chk_senha CHECK (CHAR_LENGTH(senha)
= 8);

ALTER TABLE Postagem ADD COLUMN foto LONGBLOB;

```

Criação dos insert's

```
-- Inserção de dados na tabela Usuario com senhas
INSERT INTO Usuario (nome_usuario, email, data_nascimento, foto_perfil,
senha)
VALUES
('joaosilva', 'joao@gmail.com', '1990-01-01',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\jose.png'), 'senha123'),
('mariaribeiro', 'maria@gmail.com', '1992-02-02',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\maria.png'), '123senha'),
('carlos123', 'carlos@gmail.com', '1985-03-03',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\carlos.png'), 'abc12345'),
('ana_clara', 'ana@gmail.com', '1993-04-04',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\ana.png'), 'segredo1'),
('lucas_monte', 'lucas@gmail.com', '1991-05-05',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\lucas.png'), 'senhaabc'),
('rafaela.m', 'rafaela@gmail.com', '1987-06-06',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\rafaela.png'), '87654321'),
('fernando.s', 'fernando@gmail.com', '1989-07-07',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\fernando2.png'), 'minhasen'),
('beatrizf', 'beatriz@gmail.com', '1994-08-08',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\beatriz.png'), 'qwerty12'),
('gustavo.r', 'gustavo@gmail.com', '1995-09-09',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\gustavo.png'), '1234abcd'),
('larissad', 'larissa@gmail.com', '1996-10-10',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\larissa2.png'), 'senha456');

-- Inserção de dados na tabela Conexao
INSERT INTO Conexao (usuario1_id, usuario2_id)
VALUES
(1, 2), (1, 3), (2, 4), (3, 5), (4, 6),
(5, 7), (6, 8), (7, 9), (8, 10), (9, 1);

-- Inserção de dados na tabela Postagem
INSERT INTO Postagem (usuario_id, conteudo, tipo)
VALUES
(1, 'Primeira postagem!', 'texto'),
(3, 'Assistam este vídeo incrível.', 'video'),
```

```

(4, 'Pensamento do dia.', 'texto'),
(6, 'Compartilhando um tutorial.', 'video'),
(7, 'Bom dia, pessoal!', 'texto'),
(9, 'Meu novo projeto.', 'video'),
(10, 'Feliz aniversário para mim!', 'texto');

-- Inserção de postagens com fotos
INSERT INTO Postagem (usuario_id, conteudo, tipo, foto)
VALUES
(1, 'Foto da minha viagem!', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\viagem1.png')),
(2, 'Compartilhando uma receita deliciosa.', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\receita1.jpg')),
(3, 'Paisagem incrível.', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\paisagem1.jpg')),
(4, 'Novo produto lançado!', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\produto1.png')),
(5, 'Confira essa arte!', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\arte1.jpg')),
(6, 'Vamos Treinae!!', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\treino1.png')),
(7, 'Saudades desse lugar.', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\viagem2.png')),
(8, 'Meu novo animal de estimação!', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\pet1.png')),
(9, 'Foto de um jantar especial.', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\jantar1.png')),
(10, 'Céu ao entardecer.', 'imagem',
LOAD_FILE('C:\\ProgramData\\MySQL\\MySQL Server
8.0\\Uploads\\imagens\\ceu1.png'));

-- Inserção de dados na tabela Interacao
INSERT INTO Interacao (postagem_id, usuario_id, tipo)
VALUES
(9, 3, 'like'),
(14, 10, 'like'),
(12, 7, 'dislike'),
(9, 3, 'like'),
(15, 1, 'like'),
(18, 4, 'dislike'),

```

```

(16, 2, 'like'),
(17, 3, 'like'),
(8, 1, 'dislike'),
(22, 8, 'like');

-- Inserção de dados na tabela Comentario
INSERT INTO Comentario (postagem_id, usuario_id, conteudo,
comentario_pai_id)
VALUES
(8, 1, 'Ótima postagem!', NULL),
(9, 3, 'Concordo!', null),
(11, 6, 'Adorei a foto.', NULL),
(19, 5, 'Muito bom!', NULL),
(14, 10, 'Que interessante.', NULL),
(15, 1, 'Adoro isso.', null),
(24, 10, 'Muito útil!', NULL),
(17, 3, 'Bom dia pra você também.', NULL),
(8, 1, 'Parece ótimo!', NULL),
(9, 3, 'Excelente trabalho!', NULL);

-- Inserção de dados na tabela Notificacao
INSERT INTO Notificacao (usuario_id, tipo, origem_id)
VALUES
(1, 'avaliacao', 2), (2, 'comentario', 1), (3, 'avaliacao', 4),
(4, 'comentario', 3), (5, 'avaliacao', 6), (6, 'comentario', 5),
(7, 'avaliacao', 8), (8, 'comentario', 7), (9, 'avaliacao', 10),
(10, 'comentario', 9);

-- Inserção de dados na tabela Grupo
INSERT INTO Grupo (nome, descricao)
VALUES
('Cinema', 'Discussões sobre filmes e séries'),
('Tecnologia', 'Novidades do mundo tech'),
('Viagens', 'Compartilhando experiências de viagem'),
('Gastronomia', 'Receitas e dicas culinárias'),
('Esportes', 'Tudo sobre esportes!'),
('Fotografia', 'Paixão por fotos e técnicas'),
('Literatura', 'Livros e leituras marcantes'),
('Música', 'Bandas e músicas preferidas'),
('Jogos', 'Games e diversão'),
('Natureza', 'Amor pela natureza e sustentabilidade');

-- Inserção de dados na tabela Membro_Grupo
INSERT INTO Membro_Grupo (grupo_id, usuario_id, funcao)
VALUES
(1, 1, 'administrador'), (1, 2, 'membro'), (2, 3, 'administrador'),
(2, 4, 'membro'), (3, 5, 'administrador'), (3, 6, 'membro'),
(4, 7, 'administrador'), (4, 8, 'membro'), (5, 9, 'administrador'),

```

```
(5, 10, 'membro');

-- Inserção de dados na tabela Mensagem
INSERT INTO Mensagem (remetente_id, destinatario_id, conteudo, status)
VALUES
(1, 2, 'Oi, tudo bem?', 'enviada'), (2, 3, 'Sim, e você?', 'recebida'),
(3, 4, 'Gostei do seu post!', 'lida'), (4, 5, 'Muito obrigado!',
'enviada'),
(5, 6, 'Parabéns pelo projeto.', 'lida'), (6, 7, 'Obrigado!',
'recebida'),
(7, 8, 'Quer participar do grupo?', 'enviada'), (8, 9, 'Claro!',
'lida'),
(9, 10, 'Vamos organizar um evento.', 'enviada'), (10, 1, 'Ótima
ideia!', 'lida');

-- Inserção de dados na tabela Tag
INSERT INTO Tag (nome)
VALUES
('Cinema'), ('Tecnologia'), ('Esportes'),
('Fotografia'), ('Natureza'), ('Viagens'),
('Literatura'), ('Gastronomia'), ('Música'), ('Jogos');

-- Inserção de dados na tabela Usuario_Tag
INSERT INTO Usuario_Tag (usuario_id, tag_id)
VALUES
(1, 1), (1, 2), (2, 3), (2, 4), (3, 5),
(4, 6), (5, 7), (6, 8), (7, 9), (8, 10);
```

Criação das Consultas

```
/*1. Recuperar as postagens mais recentes de um usuário e suas interações
Consulta para listar as postagens de um usuário específico, incluindo o número de likes e dislikes recebidos:*/
```

```
SELECT
    p.id_postagem AS n_postagem,
    p.conteudo,
    p.tipo,
    p.data_criacao,
    COUNT(CASE WHEN i.tipo = 'like' THEN 1 END) AS total_likes,
    COUNT(CASE WHEN i.tipo = 'dislike' THEN 1 END) AS total_dislikes
FROM
    Postagem p
LEFT JOIN
    interacao i ON p.id_postagem = i.postagem_id
WHERE
    p.usuario_id = 1 -- ID do usuário
GROUP BY
    p.id_postagem
ORDER BY
    p.data_criacao DESC;
```

```
/* 2. Buscar os membros de um grupo e suas funções
Consulta para obter a lista de membros de um grupo específico, incluindo as funções de cada usuário no grupo*/
```

```
SELECT
    mg.grupo_id,
    g.nome AS nome_grupo,
    u.nome_usuario,
    mg.funcao
FROM
    membro_grupo mg
INNER JOIN
    usuario u ON mg.usuario_id = u.id_usuario
INNER JOIN
    grupo g ON mg.grupo_id = g.id_grupo
WHERE
    mg.grupo_id = 1; -- ID do grupo (ex.: Cinema)
```

```
/*3. Exibir as notificações de um usuário
```

Consulta para listar todas as notificações de um usuário, ordenadas por data*/

```
SELECT
    n.id_notificacao AS n_notificacao,
    n.tipo,
    n.origem_id,
    n.data_notificacao
FROM
    Notificacao n
WHERE
    n.usuario_id = 1 -- ID do usuário
ORDER BY
    n.data_notificacao DESC;
```

/*4. Listar usuários com interesses (tags) semelhantes

Consulta para encontrar usuários que compartilham pelo menos uma tag com um usuário específico*/

```
SELECT
    u.id_usuario,
    u.nome_usuario,
    t.nome AS interesse_comum
FROM
    Usuario_Tag ut1
INNER JOIN
    Usuario_Tag ut2 ON ut1.tag_id = ut2.tag_id AND ut1.usuario_id !=
ut2.usuario_id
INNER JOIN
    Usuario u ON ut2.usuario_id = u.id_usuario
INNER JOIN
    Tag t ON ut1.tag_id = t.id_tag
WHERE
    ut1.usuario_id = 6; -- ID do usuário
```

/*5. Recuperar o histórico de mensagens entre dois usuários

Consulta para listar o histórico de mensagens trocadas entre dois usuários, ordenadas por data*/

```
SELECT
    m.id_mensagem,
    m.remetente_id,
    remetente.nome_usuario AS remetente,
    m.destinatario_id,
    destinatario.nome_usuario AS destinatario,
    m.conteudo,
```

```

        m.status,
        m.data_envio
FROM
    Mensagem m
INNER JOIN
    Usuario remetente ON m.remetente_id = remetente.id_usuario
INNER JOIN
    Usuario destinatario ON m.destinatario_id = destinatario.id_usuario
WHERE
    (m.remetente_id = 1 AND m.destinatario_id = 2)
    OR
    (m.remetente_id = 2 AND m.destinatario_id = 1) -- IDs dos dois
usuários
ORDER BY
    m.data_envio;

```

Criação de Trigger's e Procedures

Trigger para Verificação de nome_usuario único

Ação: O Trigger é executado antes de uma nova linha ser inserida na tabela usuario.

Verificação: Utiliza o comando IF EXISTS para verificar se já existe um registro com o mesmo nome_usuario.

Erro: Se encontrar um nome duplicado, um erro é gerado com uma mensagem descritiva.

```

/*Trigger para Verificação de nome_usuario Único*/

DELIMITER $$
CREATE TRIGGER VerificaNomeUsuarioAntesDeInserir
BEFORE INSERT ON Usuario
FOR EACH ROW
BEGIN
    -- Verifica se já existe um usuário com o mesmo nome
    IF EXISTS (SELECT 1 FROM Usuario WHERE nome_usuario =
NEW.nome_usuario) THEN
        -- Gera um erro se o nome já existir
        SIGNAL SQLSTATE '45000'

```



```

        SET MESSAGE_TEXT = 'Erro: O nome de usuário já está em uso.
Escolha outro nome.';
    END IF;
END $$

DELIMITER ;

/* usando a procedure*/
INSERT INTO Usuario (nome_usuario, email, data_nascimento, foto_perfil,
senha)
VALUES ('joaosilva', 'joao22mail@gmail.com', '2000-01-01', NULL,
'joao123');

```

99 09:19:48 INSERT INTO Usuario (nome_usuario, email, data_nascimento, foto_perfil, senha) VALUES (joaosilva, joao2... Error Code: 1644, Erro: O nome de usuário já está em uso. Escolha outro nome.

Figura 03: Funcionamento da trigger

Procedure para Recuperar nome_usuario

Parâmetro: Recebe o email do usuário como parâmetro de entrada (p_email).

Consulta: Busca o nome_usuario correspondente ao email fornecido.

Validação:

1. Caso o email não seja encontrado, gera um erro informando que o email não está cadastrado.
2. Se o email existir, exibe uma mensagem indicando que o nome de usuário foi enviado ao email.

```

DELIMITER $$

CREATE PROCEDURE RecuperaNomeUsuario(
    IN p_email VARCHAR(100)
)
BEGIN
    DECLARE v_nome_usuario VARCHAR(50);

```

```

-- Verifica se o email existe na tabela Usuario
SELECT nome_usuario
INTO v_nome_usuario
FROM Usuario
WHERE email = p_email;

-- Se não encontrar o email, envia uma mensagem de erro
IF v_nome_usuario IS NULL THEN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Erro: O email fornecido não está
cadastrado.';
ELSE
    -- Simula o envio da mensagem com o nome de usuário
    SELECT CONCAT(
        'Enviamos uma mensagem com o nome de usuário "',
        v_nome_usuario,
        '" para o email: ', p_email
    ) AS Mensagem;
END IF;
END $$

DELIMITER ;

/* Recuperação com Email Existente:*/

CALL RecuperaNomeUsuario('joao@gmail.com');

/* Recuperação com Email Inexistente:*/

CALL RecuperaNomeUsuario('inexistente@gmail.com');

```

57	CALL RecuperaNomeUsuario('joao@gmail.com');
58	
Result Grid Filter Rows: Export: Wrap Cell Content:	
	Mensagem
▶	Enviamos uma mensagem com o nome de usuário "joaosilva" para o email: joao@gmail.com

Figura 04: Funcionamento da procedure

Procedure para Recuperar a Senha

Parâmetro: Recebe o email do usuário como parâmetro de entrada (p_email).

Consulta: Busca o nome_usuario e a senha associados ao email fornecido.

Validação:

1. Caso o email não exista, gera um erro informando que o email não está cadastrado.
2. Se o email existir, exibe uma mensagem indicando que a senha foi enviada ao email.

Envio de Mensagem: A senha é simuladamente enviada ao email cadastrado. Um sistema de envio real pode ser integrado, caso necessário.

```
DELIMITER $$

CREATE PROCEDURE RecuperaSenha(
    IN p_email VARCHAR(100)
)
BEGIN
    DECLARE v_nome_usuario VARCHAR(50);
    DECLARE v_senha VARCHAR(100);

    -- Verifica se o email existe e obtém o nome de usuário e senha
    SELECT nome_usuario, senha
    INTO v_nome_usuario, v_senha
    FROM Usuario
    WHERE email = p_email;

    -- Se o email não for encontrado, retorna um erro
    IF v_nome_usuario IS NULL THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Erro: O email fornecido não está
cadastrado.';
    ELSE
        -- Simula o envio de uma mensagem com a senha ao email
        SELECT CONCAT(
            'Olá, ', v_nome_usuario,
```

```

        '. Sua senha foi enviada ao email: ', p_email
    ) AS Mensagem;

    -- Simulação do envio real (opcional, depende do sistema
    externo)
    -- Exemplo fictício: CALL EnviaEmail(p_email, CONCAT('Sua senha
    é: ', v_senha));
    END IF;
END $$

DELIMITER ;

/* Exemplo de uso*/

CALL RecuperaSenha('joao@gmail.com');

```

101	CALL RecuperaSenha('joao@gmail.com');
102	


Result Grid	Filter Rows: <input type="text"/>	Export: 	Wrap Cell Content
	Mensagem		
▶	Olá, joaosilva. Sua senha foi enviada ao email: joao@gmail.com		

Figura 05: Funcionamento da procedure

Procedure Notifica Mensagem Recebida

Ao inserir uma mensagem, o sistema exiba uma notificação textual.

```
DELIMITER $$

CREATE PROCEDURE NotificarMensagemRecebida(
    IN remetente_id INT,
    IN destinatario_id INT
)
BEGIN
    DECLARE remetente_nome VARCHAR(255);
    DECLARE destinatario_nome VARCHAR(255);

    -- Recupera o nome do remetente
    SELECT nome_usuario INTO remetente_nome
    FROM Usuario
    WHERE id_usuario = remetente_id;

    -- Recupera o nome do destinatário
    SELECT nome_usuario INTO destinatario_nome
    FROM Usuario
    WHERE id_usuario = destinatario_id;

    -- Exibe a mensagem de notificação
    SELECT CONCAT('Olá, ', destinatario_nome, '! Você recebeu uma nova
mensagem de ', remetente_nome, '.');
END$$

DELIMITER ;

INSERT INTO Mensagem (remetente_id, destinatario_id, conteudo, status)
VALUES (1, 3, 'Olá, como vai?', 'enviada');

CALL NotificarMensagemRecebida(1, 3);
```

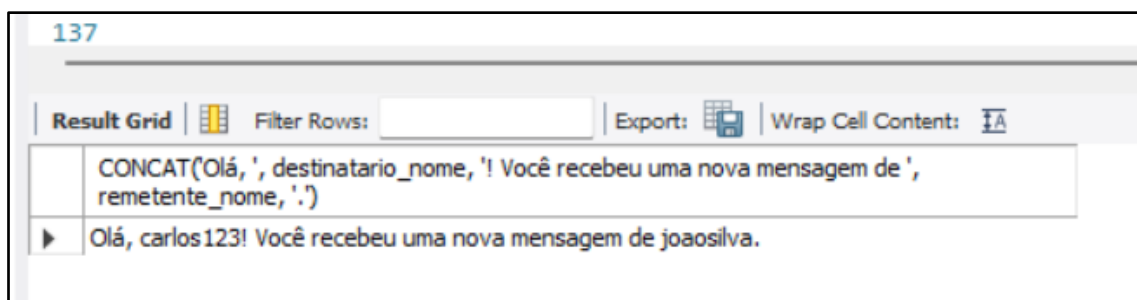


Figura 06: Funcionamento da procedure

Trigger VerificarConteudoPostagem

A trigger será ativada antes de inserir uma nova postagem ou comentário.

O conteúdo (NEW.conteudo) será analisado em busca de palavras ou frases proibidas.

Se qualquer palavra proibida for encontrada, a inserção será bloqueada com a mensagem:

- **Postagens:** *"Sua postagem não foi publicada porque viola nossas diretrizes."*
- **Comentários:** *"Seu comentário não foi publicado porque viola nossas diretrizes."*

```
DELIMITER $$

CREATE TRIGGER VerificarConteudoPostagem
BEFORE INSERT ON Postagem
FOR EACH ROW
BEGIN
    DECLARE mensagem_erro VARCHAR(255);

    -- Lista de palavras proibidas
    IF NEW.conteudo LIKE '%macaco%' OR
       NEW.conteudo LIKE '%preto imundo%' OR
       NEW.conteudo LIKE '%chinês nojento%' OR
       NEW.conteudo LIKE '%cigano ladrão%' OR
       NEW.conteudo LIKE '%vadia%' OR
       NEW.conteudo LIKE '%mulher burra%' OR
       NEW.conteudo LIKE '%mulher é pra cozinha%' OR
       NEW.conteudo LIKE '%feminazi%' OR
       NEW.conteudo LIKE '%bichinha%' OR
       NEW.conteudo LIKE '%viado%' OR
       NEW.conteudo LIKE '%traveco%' OR
       NEW.conteudo LIKE '%isso não é homem%' OR
       NEW.conteudo LIKE '%retardado%' OR
       NEW.conteudo LIKE '%mongoloide%' OR
       NEW.conteudo LIKE '%aleijado%' OR
       NEW.conteudo LIKE '%burro de nascença%' OR
       NEW.conteudo LIKE '%fanático religioso%' OR
       NEW.conteudo LIKE '%terrorista muçulmano%' OR
       NEW.conteudo LIKE '%crente burro%' OR
```

```

NEW.conteudo LIKE '%seita do demônio%' OR
NEW.conteudo LIKE '%nojento%' OR
NEW.conteudo LIKE '%lixo humano%' OR
NEW.conteudo LIKE '%imundo%' OR
NEW.conteudo LIKE '%parasita%' THEN

    -- Mensagem de erro
    SET mensagem_erro = 'Sua postagem não foi publicada porque
viola nossas diretrizes.';
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = mensagem_erro;
END IF;
END$$

DELIMITER ;

INSERT INTO Postagem (usuario_id, conteudo, tipo)
VALUES (6, 'Seu crente nojento.', 'texto');

```


	136 15:45:59	INSERT INTO Postagem (usuario_id, conteudo, tipo) VALUES (6, 'Seu crente nojento.', 'texto')	Error Code: 1644. Sua postagem não foi publicada porque viola nossas diretrizes.
---	--------------	--	--

Figura 07: Funcionamento da trigger.