Universidade Federal de Uberlândia - UFU
FACOM - Faculdade de Computação

Matheus Vinicius Maximo Santos
Lucas Matos Rodrigues
Eduarda Lopes Santos Moura

Documentação trabalho final:
Sistema de Banco de Dados

FACEPAGE

Uberlândia - MG
2024

Sumário

1. Especificação do Problema

Recentemente, uma nova rede social online, FACEPAGE, foi fundada. Considerando as tendências atuais, os gerentes da FACEPAGE estão convencidos que essa será a nova moda em breve.

a) Quando novos usuários desejam ingressar na FACEPAGE é necessário que eles preencham um formulário com as suas informações pessoais (como, CPF, nome, e-mail, telefone de contato, endereço e data de nascimento). Cada usuário deve definir um nome de usuário, que deve ser único, e uma senha.

b) Após o preenchimento do formulário uma conta pode ser criada. Cada usuário deve escolher qual tipo de conta deseja: uma conta empresarial ou uma conta pessoal. Só existem esses dois tipos de conta.

c) Uma conta empresarial é especialmente projetada para apoiar as empresas em suas campanhas de marketing. Quando um usuário decide abrir uma conta empresarial é obrigatório informar o nome e o CNPJ da empresa.

d) Usuários com uma conta empresarial pagam uma taxa mensal que varia de acordo com o pacote escolhido. Além do preço cobrado, a diferença entre os pacotes reside nas funcionalidades oferecidas.

e) Quando um usuário opta por uma conta pessoal, ele pode se conectar com outros usuários do FACEPAGE. Apenas contas pessoais podem enviar e receber requisições de amizade.

f) Cada usuário de uma conta pessoal tem acesso ao número de amigos com os quais está conectado e consegue visualizar a lista de usuários com os quais está conectado, a lista de requisições de amizade recebidas e a lista de requisições de amizade enviadas.

g) Usuários de uma conta pessoal podem categorizar seus amigos em diferentes grupos (por exemplo, família, trabalho, academia, etc.).

h) A manutenção de múltiplas contas, independente do propósito, é considerada uma violação dos termos de uso da FACEPAGE. Se um usuário já possui uma conta pessoal (ou

empresarial), então ele não pode criar uma conta adicional, seja ela pessoal ou empresarial, por razão alguma.

i) Cada conta pode criar várias páginas. Cada página deve ser administrada por exatamente uma conta.

j) Páginas criadas por contas pessoais podem ficar visíveis para grupo(s) de amigos específicos.

k) Contas pessoais podem receber privilégios para executar ações (por exemplo, escrever algo no mural de um amigo ou ajustar alguma informação) em páginas pertencentes a outras contas pessoais.

l) Para cada página, o nome da página e o número de visitas são registrados. Para cada conta, não podem existir duas páginas com o mesmo nome.

m) Usuários com uma conta empresarial podem criar um tipo especial de página: uma página de propaganda. Essa página registra vários dados, como taxa de rejeição, taxa de cliques e taxa de conversão. A taxa de rejeição é a porcentagem de visitantes que acessaram a página e a abandonaram em seguida, sem interagir. A taxa de cliques indica o quanto os usuários que visualizaram um link clicaram nele. A taxa de conversão registra a porcentagem de usuários que concluíram uma ação desejada, como uma compra ou uma transação.

1.1 Especificação dos requisitos adicionais ao problema

n) Toda relação de amizade com o status de “Aceito” possui um chat que deve ser nomeado, o qual armazena todas as mensagens trocadas. O chat só existe quando a amizade é criada. Podem existir chats com o mesmo nome em amizades diferentes.

o) O FACEPAGE além de possuir Páginas destinadas aos usuários, também possui Jogos na plataforma. Cada jogo possui um código próprio para distinção interna, um nome único, a sua descrição e o seu nível de dificuldade.

p) Todo usuário tem a possibilidade de jogar todos os jogos disponíveis na plataforma e cada jogo pode ser jogado por mais de uma pessoa. Cada usuário tem uma pontuação própria em cada jogo que já foi jogado.

q) Existe também na FACEPAGE um incentivo por parte dos colaboradores da empresa de tornarem os jogos mais competitivos, com isso, foi atribuído a cada jogo um sistema de Ranking colocado sobre o número de pontos, em que cada usuário de um determinado jogo tem uma colocação própria com base na sua pontuação.

r) Na FACEPAGE prezamos pela interação de todos os nossos usuários em nossas comunidades. As comunidades se caracterizam por ter um nome, ao qual é único, e uma descrição.

s) Uma conta pode criar várias comunidades, mas uma comunidade só pode ser criada por uma conta.

t) Toda comunidade pode ter mais de um participante, ao passo que deve ter ao menos um usuário participando dela. O criador da comunidade conta como participante. Todo usuário que participa de uma determinada comunidade é considerado membro daquela comunidade.

u) Todo membro de uma comunidade pode criar posts, chamados de tópicos para essa comunidade. Cada tópico possui um ID único, nome do usuário que realizou o post e o nome da comunidade, além de descrição.

v) Membros de uma comunidade podem realizar comentários em tópicos dessa mesma comunidade, sendo que cada comentário feito é armazenado no sistema do FACEPAGE. Cada membro de uma comunidade pode comentar em um determinado tópico mais de uma vez, sendo diferenciado por um código.

2. Esquema Conceitual

2.1. Introdução à Modelagem Conceitual

O presente esquema conceitual apresenta a modelagem para o problema descrito na seção 3.3 atendendo aos requisitos identificados. Assim, o modelo Entidade-Relacionamento foi elaborado seguindo as notações sugeridas em sala de aula, representando entidades juntamente com seus atributos e relacionamentos entre elas. Desse modo, a estrutura visa garantir a correta organização e integridade dos dados, permitindo a implementação eficiente da modelagem para a aplicação FACEPAGE.

2.2. Descrição das Entidades

ENTIDADE USUARIO_CONTA:

Tipo-entidade criado para armazenar informações pessoais de cada usuário do site. É uma generalização de conta do site (que poderá ser pessoal ou empresarial) e por isso possui herança com restrição do tipo disjoint e participação total na especialização. Ou seja, quando o usuário criar uma conta, ele terá que escolher entre apenas uma opção: pessoal ou empresarial. Essa especialização é identificada pelo atributo tipo_conta(varchar(12), NOT NULL).

- username (varchar(30), PRIMARY KEY) - Identificador único do usuário, usado para login.
- cpf (varchar(11), UNIQUE, NOT NULL) - Cadastro de Pessoa Física, deve ser único no sistema e deve ser informado.
- nome (varchar(60), NOT NULL) - Nome completo do usuário.
- data_nasc (date, NOT NULL) - Data de nascimento do usuário no formato de data.
- email (varchar(30), NOT NULL) - Email de contato do usuário.
- senha (varchar(30), NOT NULL) - Senha do usuário.
- endereco (varchar(100)) - Endereço do usuário. Preenchimento opcional conforme interpretação do grupo.
- tel_contato (varchar(20), NOT NULL) - Telefone de contato do usuário, deve ser preenchido.

ENTIDADE PESSOAL:

Essa entidade herda a entidade USUARIO_CONTA, pois o usuário decide se quer uma conta pessoal ou empresarial.

- nro_amigos (int, NOT NULL) - É o número de amigos que a conta possui e é derivado da quantidade de solicitações de amizades aceitas do usuário.

ENTIDADE EMPRESARIAL:

Essa entidade também herda a entidade USUARIO_CONTA, pois o usuário decide se quer uma conta pessoal ou empresarial.

- nome_fantasia (varchar(60), NOT NULL) - Nome Fantasia do empreendimento do usuário.
- cnpj (varchar(12), UNIQUE, NOT NULL) - CNPJ do empreendimento do usuário, que é único por padrão.

ENTIDADE PACOTE:

Criada para simbolizar o pacote que o usuário de conta empresarial paga.

- id (int, GENERATED ALWAYS AS IDENTITY PRIMARY KEY) - Número inteiro de identificação de cada pacote, gerado automaticamente e incrementado a cada nova tupla.
- nome (varchar(30), UNIQUE, NOT NULL) - Nome do pacote, único e não nulo.
- preco (real, NOT NULL) - Preço do pacote, deve ser informado na aplicação.
- funcionalidades (varchar(100), NOT NULL) - Descrição das funcionalidades de cada pacote.

ENTIDADE GRUPO_DE_AMIGOS:

A seguinte entidade foi criada para armazenar quais usuários de contas pessoais pertencem a quais grupos, determinados unicamente por cada usuário de conta pessoal.

- codigo (int, GENERATED ALWAYS AS IDENTITY PRIMARY KEY) - Código unitário que identifica cada grupo de amigos do usuário.
- nome (varchar(30), NOT NULL) - Nome do grupo de amigos, escolhido pelo usuário.
- descricao (varchar(100)) - Descrição do grupo de amigos, escrita opcionalmente pelo usuário.

ENTIDADE PAGINA:

Essa entidade é fraca em relação à entidade CONTA_USUARIO, pois podem existir páginas com nomes diferentes desde que não sejam do mesmo usuário, conforme interpretado na situação-problema. Além disso, ela poderá se especializar em uma página de propaganda, e essa especialização será identificada pelo atributo tipo_pagina(varchar(12)) com participação parcial.

- nome (varchar(60), PRIMARY KEY) - Nome da página.
- nro_visitas (int) - Número de visitas de cada página.
- Criador(varchar(30, FOREIGN KEY (criador) REFERENCES usuario_conta(username)) - É quem cria página em questão

ENTIDADE PROPAGANDA:

Esta entidade herda a entidade PAGINA, pois faz menção à especificação do problema em que um usuário de conta empresarial pode criar um tipo especial de página de propaganda, em que algumas métricas são medidas. Assim, toda página empresarial é obrigatoriamente administrada por um usuário de conta empresarial e é criado pelo mesmo que a administra.

- taxa_cliques(double precision) - Métrica de quantos cliques a página recebeu.
- taxa_rejeicao(double precision) - Mensura a rejeição da página.
- taxa_conversao(double precision) - Mede a conversão em vendas da página.

ENTIDADE ACAO:

Usada para representar a ação que cada usuário de conta pessoal pode realizar em determinada página.

- codigo (int, GENERATED ALWAYS AS IDENTITY PRIMARY KEY) - Código único que identifica cada ação, gerado pelo sistema de forma incremental.
- descricao (varchar(100)) - Descrição da ação realizada.

ENTIDADE JOGO:

É o tipo-entidade que representa cada jogo que os usuários têm acesso ao se cadastrar no FACEPAGE. Faz parte da adição à especificação do problema criada pelo nosso grupo.

- nome (varchar(30), PRIMARY KEY) - Nome do jogo, não podem haver jogos com nomes repetidos e todo jogo deve ter um nome.
- descricao (varchar(100), NOT NULL) - Instruções de como jogar o jogo.
- dificuldade (smallint, NOT NULL) - Classificação do nível de dificuldade de um jogo, indo de 1 (muito fácil) até 10 (muito díficil).

ENTIDADE COMUNIDADE:

Assim como os jogos, trouxemos um conceito de comunidade que cada usuário do FACEPAGE pode participar e/ou fundar.

- nome (varchar(30), PRIMARY KEY) - Nome de cada comunidade.
- descricao (varchar(100)) - Descrição da comunidade, como temas em comum.
- nro_membros(int, NOT NULL DEFALT 0) - A quantidade de membros que cada comunidade individualmente possui
- Admin(varchar(30), FOREIGN KEY (admin) REFERENCES usuario_conta(username)) - Quem ativamente criou e administra a comunidade

ENTIDADE TÓPICO:

Por fim, uma outra adição do nosso grupo à especificação do problema foi a presença de um tópico, que é uma entidade fraca que referencia a agregação MEMBRO_COMUNIDADE que será discutida posteriormente. Um tópico é o que existe em cada comunidade - é como se fosse uma Thread da rede social Reddit.

- id (int, GENERATED ALWAYS AS IDENTITY PRIMARY KEY) - Número inteiro que auxilia na identificação cada tópico criado. É incrementado automaticamente.
- titulo (varchar(30), NOT NULL) - "Manchete" do tópico, ou seja, assunto principal.
- data (date, NOT NULL) - Data em que o tópico foi iniciado.
- descricao (varchar(1000), NOT NULL) - Mensagem exibida em cada tópico.

2.3. Descrição dos Relacionamentos

RELACIONAMENTO "Paga":

Relacionamento binário entre as entidades EMPRESARIAL e PACOTE. Representa o fato de que cada conta empresarial deve escolher um pacote a ser pago, e por isso é participação total na entidade EMPRESARIAL. A cardinalidade é 1:n - um pacote pode ser escolhido por várias contas empresariais, mas cada conta desse tipo deve escolher apenas um pacote a pagar.

RELACIONAMENTO "Possui":

Relacionamento binário entre as entidades GRUPO_DE_AMIGOS e PESSOAL. Esse tipo-relacionamento é baseado no pressuposto de que cada conta pessoal pode agrupar amigos de forma única e repetidamente, ou seja, todo grupo de amigos é possuído por uma conta pessoal, pois sem ela o grupo não existiria e por isso a participação de

GRUPO_DE_AMIGOS é total. A cardinalidade é de 1:n - um grupo de amigos pode pertencer a apenas uma conta, mas uma conta pode ter vários grupos de amigos.

RELACIONAMENTO "Participa":

Assim como o tipo-relacionamento anterior, é um relacionamento binário entre as entidades GRUPO_DE_AMIGOS e PESSOAL, baseado no fato de que se um grupo de amigos existe, há usuários de contas pessoais que participam dele. Assim, a cardinalidade dele é m:n - uma conta pessoal pode participar de vários grupos de amigos e um grupo de amigos pode ter como participantes várias contas pessoais. A cardinalidade é m:n - muitas contas podem participar de diversos grupos de amigos diferentes e diversos grupos de amigos tem muitas contas atribuídas a ele.

RELACIONAMENTO "Requisição amizade":

Relacionamento unário da entidade PESSOAL. Esse tipo-relacionamento representa os pedidos de amizade de um usuário de conta pessoal para outro usuário de conta pessoal, ou seja, um usuário requisita e outro recebe. Assim, como toda solicitação tem um estado, esse relacionamento possui um atributo status(varchar(20), NOT NULL) que pode possuir os seguintes valores: aceito, negado ou pendente. Possui cardinalidade m:n - uma conta pessoal pode pedir em amizade várias contas pessoais, enquanto outra conta pessoal pode receber vários pedidos de amizade.

RELACIONAMENTO "Administra":

Relacionamento binário entre as entidades USUARIO_CONTA e PAGINA. Originado pelo fato de que toda conta, seja ela pessoal ou empresarial, pode administrar (e consequentemente criar) uma página, desde que ela não seja do tipo especial PROPAGANDA que só pode ser criado por empresariais. Então, consequentemente uma página existe porque uma conta a administra, logo, ela possui participação total em relação a USUARIO_CONTA e além disso, cada página é identificada pelo username do administrador somado ao nome da página, pois podem haver páginas com mesmo nome de diferentes usuários. Assim, esse

relacionamento é o identificador da entidade PAGINA, que é fraca, e a cardinalidade é de 1:n - uma conta pode ter várias páginas mas uma página só pode pertencer a uma conta.

RELACIONAMENTO "Cria":

É um tipo-relacionamento binário entre as entidades EMPRESARIAL e PROPAGANDA. Ele simboliza a especificação de que uma conta empresarial pode criar um tipo diferente de página para propaganda que possui métricas comerciais a serem medidas. Sendo assim toda entidade PROPAGANDA necessita de um "criador", que é um usuário empresarial, que também é o administrador da página. A cardinalidade é 1:n - uma conta empresarial pode criar várias páginas de propaganda e cada página de propaganda só pode pertencer a uma conta empresarial.

RELACIONAMENTO "Pode ver":

Relacionamento binário entre GRUPO_DE_AMIGOS e PAGINA. Esse tipo-relacionamento foi criado de forma a representar a especificação que um administrador de página pode restringir a visão de sua página para algum grupo de amigos de sua escolha. A cardinalidade é m:n - um grupo de amigos tem acesso a várias páginas e uma página pode ter vários grupos de amigos que possuem visão permitida.

RELACIONAMENTO "Privilégio":

Relacionamento ternário entre as entidades ACAO, PESSOAL e PAGINA. Foi instanciado para cumprir os requisitos que enunciam que cada conta pessoal deve poder, quando autorizada, realizar ações em determinada página. Logo, a cardinalidade m:n:n justifica bem o uso dessa relação - dado uma ação e uma conta pessoal, n páginas podem sofrer a ação dessa conta; dado uma conta pessoal e uma página, m ações podem ser feitas por essa conta nessa página e dado uma ação e uma página, n contas pessoais podem fazer uma mesma ação nessa mesma página.

RELACIONAMENTO "Joga":

Esse tipo-relacionamento foi criado para atender à especificação criada pelo nosso grupo de que cada conta criada no FACEPAGE pode jogar todos os jogos disponíveis no site. Foi instanciado entre as entidades USUARIO_CONTA e JOGO e em cada jogo que é jogado por uma conta, há uma pontuação registrada. Desse modo, o atributo pontuacao(int, NOT NULL) é um atributo desse relacionamento. A cardinalidade é m:n de maneira que uma conta pode jogar vários jogos e um jogo pode ser jogado por várias contas.

RELACIONAMENTO "Funda":

Relacionamento binário entre COMUNIDADE e USUARIO_CONTA. Foi criado para fazer jus à especificação adicional do nosso grupo que todo usuário pode fundar uma comunidade e toda comunidade deve ter sido fundada por alguém. Então, há participação total no lado da COMUNIDADE, pois não existe comunidade alguma que não tenha fundador. A cardinalidade é de 1:n, pois uma conta pode fundar várias comunidades e uma comunidade só pode ser fundada por uma única conta.

RELACIONAMENTO "Participa_de":

Assim como o relacionamento "Funda" comentado anteriormente a esse, é um relacionamento binário também entre COMUNIDADE e USUARIO_CONTA, o qual foi criado justamente para dizer que um usuário pode participar de comunidades criadas por outros usuários. Também possui participação total no lado da COMUNIDADE, pois toda comunidade possui pelo menos um participante. A cardinalidade é de m:n - um usuário pode participar de várias comunidades e uma comunidade pode ter vários usuários como participantes.

RELACIONAMENTO "Postam":

É um relacionamento binário entre a agregação MEMBRO_COMUNIDADE e a entidade TOPICO. De acordo com a especificação criada pelo nosso grupo, cada membro de uma comunidade pode postar um tópico dentro dessa mesma comunidade, e este só existe se for postado por um membro. Logo, a participação no lado de TOPICO é total. Além disso,

cada tópico é identificado pelo username do autor somado ao número identificador (id) do tópico, o que permite com que um mesmo autor poste vários tópicos com o mesmo nome. Dessa maneira, esse relacionamento é o identificador da entidade fraca TOPICO, e a cardinalidade é 1:n, pois um tópico deve ser postado por um usuário e um usuário pode postar vários tópicos.

RELACIONAMENTO "Comentam":

Assim como o anterior, é um relacionamento binário entre a agregação MEMBRO_COMUNIDADE e a entidade TOPICO. Seguindo a especificação criada pelo nosso grupo, cada membro de uma comunidade pode comentar um tópico postado dentro dessa mesma comunidade. Cada comentário é armazenado em um atributo chamado comentario(varchar(1000), NOT NULL). Por fim, a cardinalidade é m:n porque um membro pode comentar várias vezes em um mesmo tópico e um tópico pode ter vários comentários de vários membros.

2.4. Descrição das Agregações

AGREGAÇÃO CHAT:

De acordo com nossas especificações próprias ao problema, um usuário de conta pessoal pode criar chats com cada amizade aceita. Esses chats devem receber um nome, o qual não precisa ser único, desde que com contas diferentes. Cada chat é identificado pelos usernames dos membros da amizade somado ao nome do chat e todas as mensagens ficam armazenadas.

- nome (varchar(30), PRIMARY KEY) - Nome do chat.
- mensagens (mensagem varchar(1000), NOT NULL) - Mensagens enviadas em um chat

AGREGAÇÃO RANKING:

Nosso grupo optou por identificar cada pontuação em um jogo jogado por um usuário como uma agregação entre as entidades USUARIO_CONTA e JOGO por meio do

relacionamento Joga. Assim, é formada uma tabela com a posição do jogador e a pontuação, a qual é atributo do relacionamento Joga, mas, no mapeamento discutido a seguir foi mapeado juntamente com a agregação.

- posicao_ranking (int, UNIQUE, NOT NULL) - Posição de cada jogador de acordo com sua pontuação, sendo 1 a primeira e mais alta posição até a n-ésima e mais baixa posição.

AGREGAÇÃO MEMBRO_COMUNIDADE:

A identificação de cada usuário membro de uma comunidade específica foi feita por meio de uma agregação no relacionamento Participa entre as entidades COMUNIDADE e USUARIO_CONTA. Essa identificação foi feita para que um tópico pudesse ser postado por cada membro de uma comunidade dentro dessa mesma comunidade específica. Ela não possui atributos.

2.5. Imagem do Esquema Conceitual

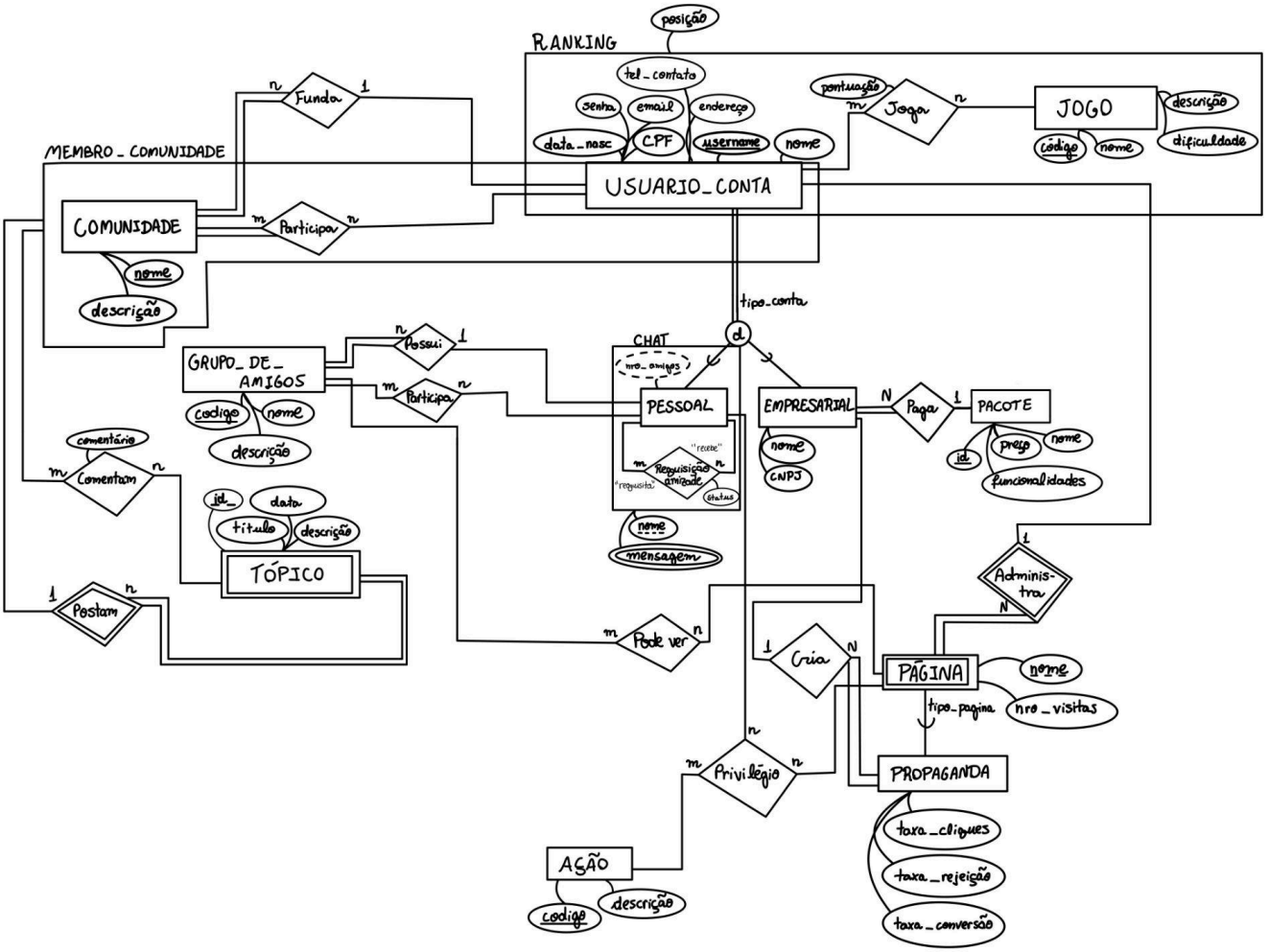


Figura 01 - Esquema Conceitual Inicial
Fonte: Autoria dos autores

Em suma, com o fim da modelagem inicial do problema, o grupo está apto para continuar o processo designado pelo texto orientador.

2.6. Bibliografia utilizada

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL 17.4 Documentation*. Disponível em: https://www.postgresql.org/docs/17/. Acesso em: 06/03/2025. Seções relevantes da Documentação: Identity Columns (p. 99, seção 5.3), Not-Null Constraints (p. 104, seção 5.5.2), Unique Constraints (p. 105, seção 5.5.3), Primary Keys (p. 106, seção 5.5.4), Foreign Keys (p. 107, seção 5.5.5), Character Types (p. 194, seção 8.3), Date/Time Types (p. 198, seção 8.5), Numeric Types (p. 187, seção 8.1).

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. Tradução de Daniel Vieira. 6. ed. São Paulo: Pearson Addison Wesley, 2011. Capítulos 7 e 8, “Modelagem de dados usando o modelo Entidade-Relacionamento (ER)”(p.150) e “O modelo Entidade-Relacionamento Estendido (EER)”(p.180).

3. Esquema Relacional

3.1. Introdução ao Esquema Relacional

Esta seção apresenta o mapeamento do esquema conceitual definido na seção 2 deste documento para o modelo relacional, a fim de detalhar as relações que compõem o banco de dados da aplicação FACEPAGE. O objetivo desse mapeamento é transformar as entidades e relacionamentos em tabelas sem ferir as integridades e consistências dos dados. Cada relação é descrita com seus respectivos atributos, chaves primárias, chaves estrangeiras e restrições de unicidade. Em alguns casos, foi necessário tomar decisões sobre diferentes estratégias de modelagem, as quais serão justificadas ao longo desta seção do documento. A todo momento foram tomados os devidos cuidados para que as restrições de integridade de entidade, unicidade de chave e integridade referencial fossem preservadas. Os comandos SQL para a implementação das tabelas descritas nesta seção serão apresentados posteriormente neste mesmo documento.

3.2. Justificativas do Mapeamento

Nessa seção discorreremos sobre escolhas de mapeamento que fogem às convenções do livro consultado e foram escolhas dos integrantes.

MAPEAMENTO AGREGAÇÃO "Ranking":

Essa agregação toma como atributo de si o atributo do relacionamento "Joga" que é m:n, pois no nosso modelo faz sentido que um ranking além de ter uma posição tenha a pontuação obtida ao jogar. Assim, o relacionamento "Joga" não foi mapeado fora da agregação.

MAPEAMENTO AGREGAÇÃO "Membro_Comunidade":

Nessa agregação temos o relacionamento "Participa" dentro dela ao qual denomina todos os participantes dessa comunidade, com isso temos a nossa especificação de requisito incial que diz que "*Todo usuário que participa de uma determinada comunidade é*

considerado membro daquela comunidade.", isso faz sentido ao mapearmos a agregação Membro_Comunidade.

3.3. Mapeamento dos Relacionamentos

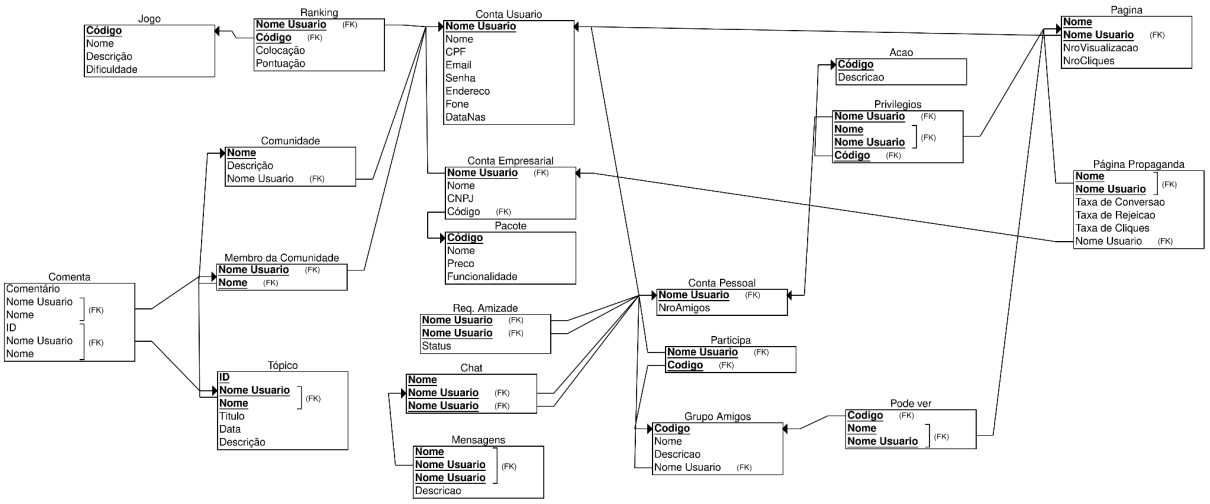


Figura 02 - Mapeamento Relacional em software
Fonte: Autoria dos autores

Figura 03 - Mapeamento Relacional à mão
Fonte: Autoria dos autores

3.5. Bibliografia consultada

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. Tradução de Daniel Vieira. 6. ed. São Paulo: Pearson Addison Wesley, 2011. Capítulo 3, p. 57: “O modelo de dados relacional e as restrições em bancos de dados relacionais”.

4. Banco de Dados

Esta seção descreve os comandos SQL utilizados para a criação do banco de dados da aplicação **FACEPAGE**, bem como os comandos relacionados à inserção de dados, consultas, atualizações, gatilhos e procedimentos armazenados.

4.1. Criação do Banco de Dados

A criação do banco de dados seguiu uma ordem específica de execução dos comandos, respeitando as dependências entre as tabelas. Nesta subseção, será apresentada essa ordem, juntamente com a criação das tabelas e as justificativas para as restrições definidas.

SCHEMA:

```
CREATE SCHEMA facepage;
SET search_path to facepage;
```

Os comandos acima foram utilizados para criar o esquema facepage, o qual organiza logicamente os objetos do banco de dados. A definição do search_path garante que o SGBD utilize o esquema correto ao criar e manipular as tabelas subsequentes.

TABELA USUARIO_CONTA:

```
CREATE TABLE usuario_conta (
            username varchar(30),
            cpf varchar(11) UNIQUE NOT NULL,
            nome varchar(60) NOT NULL,
            data_nas date NOT NULL,
            email varchar(30) NOT NULL,
            senha varchar(30) NOT NULL,
            endereco varchar(100),
            tel_contato varchar(20) NOT NULL,
            CONSTRAINT usuario_conta_pk PRIMARY KEY(username)
      );
```

As primeiras tabelas a serem criadas devem ser aquelas que não possuem dependências com outras por meio de chaves estrangeiras. Por esse motivo, a tabela usuario_conta foi definida inicialmente, representando cada conta registrada na plataforma FACEPAGE, sejam pessoais ou empresariais.

TABELA PESSOAL:

```
CREATE TABLE pessoal (
      nro_amg int NOT NULL DEFAULT 0,
      CONSTRAINT pessoal_pk PRIMARY KEY(username)
) INHERITS (usuario_conta);
```

Embora esta tabela não possua restrições de chave estrangeira, ela herda atributos da tabela usuario_conta, sendo, portanto, dependente dela. Como a tabela usuario_conta já foi definida anteriormente, a criação da tabela pessoal pode ser realizada neste ponto. Essa tabela será referenciada por diversas outras a serem criadas posteriormente e é responsável por armazenar as informações específicas das contas do tipo pessoal.

TABELA PACOTE:

```
CREATE TABLE pacote (
      id int GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
      nome varchar(30) UNIQUE NOT NULL,
      preco real NOT NULL,
      funcionalidades varchar(100) NOT NULL
);
```

A tabela pacote poderia ter sido criada imediatamente após a usuario_conta, uma vez que não possui dependências por meio de chaves estrangeiras. Contudo, optou-se por defini-la neste momento, pois será referenciada diretamente pela próxima tabela. Essa tabela é responsável por armazenar os diferentes planos disponíveis para as contas empresariais, detalhando seus nomes, preços e funcionalidades.

TABELA EMPRESARIAL:

```
CREATE TABLE empresarial (
      nome_fantasia varchar(60) NOT NULL,
      cnpj varchar(12) UNIQUE NOT NULL,
      id_pacote int,
      CONSTRAINT empresarial_pk PRIMARY KEY(username),
      CONSTRAINT empresarial_fk FOREIGN KEY(id_pacote)
      REFERENCES pacote(id)
) INHERITS (usuario_conta);
```

A tabela empresarial é responsável por armazenar as informações das contas do tipo empresarial, que representa uma especialização da tabela usuario_conta, assim como a tabela

pessoal. Essa tabela faz referência direta à tabela pacote, criada anteriormente, uma vez que toda conta empresarial está associada a um pacote de funcionalidades. Portanto, sua criação ocorre após a definição da tabela pacote, respeitando a ordem lógica das dependências entre as tabelas do banco de dados.

TABELA REQ_AMIZADE:

```
CREATE TABLE req_amizade (
status VARCHAR(20) NOT NULL CHECK (status IN('Pendente', 'Aceita',
'Recusada')),
            recebe_username VARCHAR(30),
            envia_username VARCHAR(30),
            CONSTRAINT req_amizade_pk PRIMARY KEY (recebe_username,
envia_username),
            CONSTRAINT req_amizade_fk_recebe FOREIGN KEY
(recebe_username) REFERENCES pessoal(username),
            CONSTRAINT req_amizade_fk_envia FOREIGN KEY
(envia_username) REFERENCES pessoal(username)
);
```

Responsável por armazenar todas as requisições de amizade realizadas entre usuários da plataforma. Ela registra o nome de usuário de quem enviou e de quem recebeu o pedido, bem como o status da solicitação, que pode assumir apenas três valores: 'Pendente', 'Aceita' ou 'Recusada'. Foi criada após a definição da tabela pessoal, uma vez que ambas as colunas envia_username e recebe_username referenciam usuários cadastrados como contas pessoais, por meio de chaves estrangeiras. A utilização de uma chave primária composta garante que não existam duplicações de requisições entre os mesmos usuários.

TABELA CHAT:

```
CREATE TABLE chat (
            nome varchar(50) NOT NULL,
            recebe_username varchar(30),
            envia_username varchar(30),
            CONSTRAINT chat_pk PRIMARY KEY (recebe_username,
envia_username),
            CONSTRAINT chat_fk_recebe FOREIGN KEY (recebe_username)
REFERENCES pessoal(username),
            CONSTRAINT chat_fk_envia FOREIGN KEY (envia_username)
REFERENCES pessoal(username)
```

```
);
```

Armazena os registros de conversas entre os usuários do tipo pessoal. Cada entrada na tabela representa um canal de comunicação identificado por um nome, juntamente com o usuário que enviou e o que recebeu a mensagem. A chave primária composta pelas colunas recebe_username e envia_username garante a unicidade de cada conversa entre dois usuários. Já as chaves estrangeiras asseguram a integridade referencial, conectando corretamente os remetentes e destinatários às suas respectivas contas pessoais.

TABELA MENSAGEM:

```
CREATE TABLE mensagem (
            id INT GENERATED ALWAYS AS IDENTITY,
            chat_recebe_username VARCHAR(30) NOT NULL,
            chat_envia_username VARCHAR(30) NOT NULL,
            remetente VARCHAR(30) NOT NULL,
            conteudo TEXT NOT NULL,
            data_envio TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
            lida BOOLEAN DEFAULT FALSE,

            CONSTRAINT mensagem_pk PRIMARY KEY
(chat_recebe_username, chat_envia_username, id),
            CONSTRAINT mensagem_fk_chat FOREIGN KEY
(chat_recebe_username, chat_envia_username)
                  REFERENCES chat(recebe_username, envia_username)
ON DELETE CASCADE,
            CONSTRAINT mensagem_fk_remetente FOREIGN KEY (remetente)
REFERENCES pessoal(username)
);
```

Responsável por armazenar as mensagens trocadas entre os usuários nos chats existentes. Cada mensagem possui um identificador único, conteúdo textual da mensagem, data e hora de envio, campo booleano indicando se a mensagem foi lida e o remetente da mensagem. As colunas chat_recebe_username e chat_envia_username identificam a conversa à qual a mensagem pertence e formam, junto com id, a chave primária composta da tabela. Isso garante a unicidade de cada mensagem dentro de seu respectivo chat.

A integridade referencial é assegurada por duas chaves estrangeiras: mensagem_fk_chat, que referencia o chat garantindo que toda mensagem pertença a um chat. Essa restrição possui ON DELETE CASCADE, que garante que, ao excluir um chat, todas as

mensagens associadas a ele também sejam removidas. Por fim, a chave estrangeira mensagem_fk_remetente assegura que o remetente da mensagem seja um usuário de conta pessoal.

TABELA GRUPO_AMIGOS:

```
CREATE TABLE grupo_amigos (
            codigo int GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
            nome varchar(30) NOT NULL,
            descricao varchar(100),
            admin varchar(30),
            CONSTRAINT grupo_amigos_fk FOREIGN KEY (admin)
REFERENCES pessoal(username)
);
```

Assim como no esquema conceitual e relacional, é possível criar um grupo de amigos selecionando amizades já existentes. A tabela armazena as informações de cada grupo, sendo elas: um código identificador gerado automaticamente, o nome do grupo, uma descrição opcional e o administrador responsável por sua criação e gerenciamento. O administrador do grupo é identificado pela chave estrangeira, assegurando que apenas usuários cadastrados como contas pessoais possam administrar grupos de amigos. Essa estrutura precede a tabela citada a seguir, que identifica qual amigo participa de qual grupo.

TABELA PARTICIPA:

```
CREATE TABLE participa (
id_grupo int,
            amigo varchar(30),
            CONSTRAINT participa_pk PRIMARY KEY (id_grupo, amigo),
            CONSTRAINT participa_fk_pessoal FOREIGN KEY (amigo)
REFERENCES pessoal(username),
            CONSTRAINT participa_fk_grupo_amigos FOREIGN KEY
(id_grupo) REFERENCES grupo_amigos(codigo)
);
```

Seguindo a ideia da tabela anterior, a tabela participa assegura os usuários que pertencem a um grupo de amigos. Ela permite que um mesmo usuário participe de diversos grupos, e que um grupo possua múltiplos participantes. A chave primária composta impede que o mesmo usuário seja vinculado mais de uma vez ao mesmo grupo. As chaves estrangeiras asseguram a integridade referencial, vinculando: amigo à tabela pessoal, confirmando que apenas usuários cadastrados podem participar de grupos e id_grupo à tabela grupo_amigos, garantindo que o grupo referenciado exista previamente.

TABELA PAGINA:

```
CREATE TABLE pagina(
      nome varchar(60),
      nro_visitas int,
      criador varchar(30),
      CONSTRAINT pagina_pk PRIMARY KEY (nome, criador),
      CONSTRAINT pagina_fk FOREIGN KEY (criador) REFERENCES
usuario_conta(username)
);
```

Cada página será armazenada nessa tabela. Cada página possui um nome, um contador de visitas e o criador. A PK composta entre nome e criador garante que um mesmo usuário não possa criar duas páginas com o mesmo nome, embora diferentes usuários possam ter páginas com nomes idênticos. A restrição de chave estrangeira assegura que o criador da página seja um usuário registrado anteriormente, o que inclui tanto contas pessoais quanto empresariais, permitindo flexibilidade na criação das páginas. Essa funcionalidade é independente das comunidades.

TABELA PROPAGANDA:

```
CREATE TABLE propaganda(
      taxa_cliques double precision,
      taxa_rejeicao double precision,
      taxa_conversao double precision,
      admin varchar(30),
```

```
      CONSTRAINT propaganda_fk FOREIGN KEY (admin) REFERENCES
empresarial(username)
) INHERITS (pagina);
```

A tabela propaganda representa um tipo específico de página. Por esse motivo, ela herda a estrutura da tabela pagina, aproveitando os atributos. Além disso, ela possui métricas voltadas à propaganda, como taxa de cliques, taxa de rejeição e taxa de conversão. O atributo admin identifica o responsável pela propaganda, sendo obrigatoriamente um usuário empresarial. Essa regra é assegurada pela restrição de chave estrangeira propaganda_fk, que referencia a tabela empresarial.

TABELA PODE_VER:

```
CREATE TABLE pode_ver (
      nome varchar(60),
      criador varchar(30),
      codigo_grupo int,
      CONSTRAINT pode_ver_pk PRIMARY KEY (nome, criador,
codigo_grupo),
      CONSTRAINT pode_ver_fk_pagina FOREIGN KEY (nome, criador)
REFERENCES pagina(nome, criador),
      CONSTRAINT pode_ver_fk_grupo_amigos FOREIGN KEY
(codigo_grupo) REFERENCES grupo_amigos(codigo)
);
```

Relacionada à tabela pagina, esta armazena informações de quais grupos de amigos têm permissão para visualizar determinada página. Assim, ela conecta as tabelas pagina e grupo_amigos por meio de sua PK. A primeira chave estrangeira garante que a página referenciada exista e a segunda chave estrangeira assegura que apenas grupos de amigos existentes sejam vinculados.

TABELA ACAO:

```
CREATE TABLE acao (
      codigo int GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
```

```
      descricao varchar(100)
);
```

Esta tabela define o conjunto de ações possíveis que podem ser realizadas em páginas. Cada ação é identificada por um código gerado automaticamente e possui uma descrição textual que especifica o tipo de ação. Essa tabela serve como base para a tabela privilegios, que mapeia quais usuários têm permissão para executar determinadas ações em páginas específicas.

TABELA PRIVILEGIOS:

```
CREATE TABLE privilegios (
      usuario varchar(30),
      nome varchar(60),
      criador varchar(30),
      codigo_acao int,
      CONSTRAINT privilegios_pk PRIMARY KEY(usuario, nome, criador,
codigo_acao),
      CONSTRAINT privilegios_fk_pessoal FOREIGN KEY (usuario)
REFERENCES pessoal(username),
      CONSTRAINT privilegios_fk_pagina FOREIGN KEY (nome, criador)
REFERENCES pagina(nome, criador),
      CONSTRAINT privilegios_fk_acao FOREIGN KEY (codigo_acao)
REFERENCES acao(codigo)
);
```

Associada à tabela anterior, é responsável por registrar os usuários com permissão para executar determinadas ações em páginas específicas do sistema. Cada registro informa o usuário que possui o privilégio (apenas usuários da tabela pessoal), a página sobre a qual o privilégio se aplica (já existente em pagina e identificada pelo nome e criador) e o código da ação que o usuário tem permissão para executar que deve existir na tabela acao.

TABELA JOGO:

```
CREATE TABLE jogo (
```

```
      nome varchar(30) UNIQUE NOT NULL PRIMARY KEY,
      descricao varchar(100) NOT NULL,
      dificuldade smallint NOT NULL CHECK (dificuldade BETWEEN 1
AND 10)
);
```

Seguindo um dos nossos recursos criados na especificação do projeto, a tabela jogo representa os jogos disponíveis para um usuário se divertir no FACEPAGE. Ela armazena o nome, que deve ser único e por isso é chave primária, além da descrição do jogo e a dificuldade que deve estar entre 1 e 10. Essa tabela deve ser criada agora, pois a seguir ela será referenciada por algumas tabelas que guardam informações pertinentes a jogos.

TABELA JOGA:

```
CREATE TABLE joga (
      nome_jogo varchar(30),
      username varchar(30),
      pontuacao int NOT NULL,
      CONSTRAINT joga_pk PRIMARY KEY (nome_jogo, username),
      CONSTRAINT joga_fk_jogo FOREIGN KEY (nome_jogo) REFERENCES
jogo(nome),
      CONSTRAINT joga_fk_usuario_conta FOREIGN KEY (username)
REFERENCES usuario_conta(username)
);
```

Essa tabela armazena o histórico de partidas dos usuários da FACEPAGE, relacionando cada jogo jogado com o usuário e sua pontuação. A chave primária composta por nome_jogo e username permite controlar quais usuários jogaram quais jogos, garantindo que cada combinação seja única. A pontuação é NOT NULL, ou seja, é obrigatória e armazena o desempenho do usuário no jogo. De chaves estrangeiras temos: nome_jogo referenciando a tabela jogo, garantindo que apenas jogos existentes possam ser associados a um usuário. Já a FK username permite que tanto usuários pessoais quanto empresariais estejam aptos a participar dos jogos.

VIEW RANKING:

```
CREATE OR REPLACE VIEW ranking AS
SELECT j.nome AS jogo, jg.username, jg.pontuacao, RANK() OVER
(PARTITION BY j.nome ORDER BY jg.pontuacao DESC) AS posicao_ranking
FROM joga jg JOIN jogo j ON jg.nome_jogo = j.nome
ORDER BY j.nome, posicao_ranking;
```

Uma escolha da nossa implementação foi fazer o ranking como visão, em vez de tabela física. A principal vantagem disso é que o ranking sempre estará atualizado, ou seja, a visão sempre refletirá em tempo real as melhores pontuações por jogo, sem a necessidade de atualizações manuais. Assim, a view reúne informações da tabela joga com a tabela jogo. O uso da função RANK() OVER (PARTITION BY j.nome ORDER BY jg.pontuacao DESC) permite gerar a posição do jogador no ranking de forma ordenada por jogo e de maneira decrescente. Logo, a VIEW facilita a visualização geral dos melhores jogadores em cada jogo.

TABELA COMUNIDADE:

```
CREATE TABLE comunidade (
      nome_comunidade varchar(30),
      descricao varchar(100),
      nro_membros int NOT NULL DEFAULT 0,
      admin varchar(30),
      CONSTRAINT comunidade_pk PRIMARY KEY (nome_comunidade),
      CONSTRAINT comunidade_fk FOREIGN KEY (admin) REFERENCES
usuario_conta(username)
);
```

Assim como o jogo, uma das nossas especificações originais para o projeto foi o conceito de comunidades. Essa tabela foi criada para armazená-las contendo: nome da comunidade (PK), descrição, o número atual de membros - que tem como valor padrão zero e será atualizado conforme usuários entrem ou saiam - e o usuário que a administra. Assim, o campo admin refere-se a tabela usuario_conta, permitindo que qualquer usuário (pessoal ou empresarial) possa administrar uma comunidade. Essa tabela será crucial para a criação das estruturas relacionadas a comunidade que implementamos em nosso projeto.

TABELA MEMBROS_COMUNIDADE:

```
CREATE TABLE membros_comunidade (
      comunidade varchar(30),
      membro varchar(30),
      CONSTRAINT membros_comunidade_pk PRIMARY KEY (comunidade,
membro),
      CONSTRAINT membros_comunidade_fk_usuario_conta FOREIGN KEY
(membro) REFERENCES usuario_conta(username),
      CONSTRAINT membros_comunidade_fk_comunidade FOREIGN KEY
(comunidade) REFERENCES comunidade(nome_comunidade)
);
```

Seguindo a ideia da tabela anterior, essa armazena os membros de cada comunidade. A chave primária garante que cada usuário possa se associar a uma comunidade apenas uma vez, evitando duplicidades. Temos como chaves estrangeiras: membro que referencia a usuario_conta, permitindo que tanto contas pessoais quanto empresariais participem das comunidades e a chave estrangeira comunidade que referencia comunidade, garantindo que a associação só ocorra com comunidades existentes.

TABELA TOPICO:

```
CREATE TABLE topico (
      codigo int GENERATED ALWAYS AS IDENTITY,
      membro varchar(30),
      comunidade varchar(30),
      titulo varchar(60) NOT NULL,
      data date NOT NULL,
      descricao varchar(1000) NOT NULL,
      CONSTRAINT topico_pk PRIMARY KEY (codigo, membro,
comunidade),
      CONSTRAINT topico_fk FOREIGN KEY (membro, comunidade)
REFERENCES membros_comunidade(membro, comunidade)
);
```

Também seguindo nossas especificações originais, temos a tabela tópico que armazena cada tópico discutido em comunidades. As informações principais sobre cada tópico são guardadas nos campos título, descricao e date. Assim, cada registro representa um tópico criado por um membro dentro de uma comunidade específica, e a PK composta garante que o

identificador do tópico seja único dentro do contexto daquele autor e comunidade. Por fim, a restrição de FK assegura que apenas membros já registrados possam criar tópicos.

TABELA COMENTAM:

```
CREATE TABLE comentam (
      id int GENERATED ALWAYS AS IDENTITY,
      comentario varchar(1000) NOT NULL,
      membro_postou VARCHAR(30),
      codigo_topico int,
      comunidade_topico varchar(30),
      comunidade_membro varchar(30),
      membro_comentou varchar(30),
      data date NOT NULL DEFAULT current_date,
      CONSTRAINT comentam_pk PRIMARY KEY (membro_postou,
membro_comentou, codigo_topico, comunidade_topico,
comunidade_membro, id),
      CONSTRAINT comentam_fk_topico FOREIGN KEY (codigo_topico,
comunidade_topico, membro_postou) REFERENCES topico(codigo,
comunidade, membro),
      CONSTRAINT comentam_fk_membros_comunidade FOREIGN KEY
(comunidade_membro, membro_comentou) REFERENCES
membros_comunidade(comunidade, membro)
);
```

Essa tabela tem a finalidade de armazenar os comentários feitos em tópicos dentro das comunidades. Cada comentário é vinculado tanto ao tópico comentado quanto ao membro que o escreveu. A PK composta garante a unicidade de cada comentário e a rastreabilidade completa de quem comentou, onde e em qual conteúdo. A restrição FK comentam_fk_topico assegura que o comentário esteja associado a um tópico existente. Já a restrição FK comentam_fk_membros_comunidade garante que apenas usuários membros de uma determinada comunidade possam comentar tópicos dentro dela. Logo, esse controle reforça a ideia de que a participação nas discussões é exclusiva para integrantes das comunidades.

4.2. Especificação de Consultas em álgebra relacional e SQL

Nesta subseção, serão apresentadas as operações e consultas realizadas sobre o banco de dados criado anteriormente, com foco na manipulação e recuperação de dados por meio da linguagem SQL. Também serão explicitadas algumas consultas presentes em SQL na álgebra relacional e mecanismos adicionais como gatilhos e procedimentos armazenados serão abordados. As consultas e operações seguem os requisitos funcionais do sistema e foram desenvolvidas com base nas tabelas previamente estruturadas.

4.2.1. Operações de Inserção

Apresenta os comandos INSERT INTO utilizados para povoar as tabelas, com exemplos de inserções simples e inserções com dependência entre tabelas. As inserções respeitam a ordem lógica do modelo, começando por tabelas independentes e progredindo para as dependentes.

PACOTE:

Descrição: 20 planos de assinatura.

Ex: Plano Básico (R$50) com 5 postagens/mês, VIP Experience (R$499) com concierge 24/7.

```
INSERT INTO pacote (nome, preco, funcionalidades) VALUES
('Plano Básico', 50.00, 'Perfil empresarial, 5 postagens/mês, suporte
básico'),
('Plano Pro', 150.00, '10 postagens/mês, analytics simples, suporte
prioritário'),
('Plano Premium', 300.00, 'Postagens ilimitadas, analytics avançados, SEO
tools'),
('Business Plus', 450.00, 'Gestão de equipe, relatórios personalizados,
integração API'),
('Social Starter', 75.00, '3 postagens/dia, métricas essenciais'),
('Enterprise Elite', 500.00, 'Consultoria personalizada, domínio
customizado, CRM integrado'),
('Growth Guru', 200.00, 'Campanhas automatizadas, audience insights'),
('E-Commerce Pack', 350.00, 'Catálogo de produtos, checkout integrado'),
('Influencer Boost', 120.00, 'Ferramentas de engajamento, hashtag
analytics'),
```

```
('Local Hero', 80.00, 'Geolocalização, promoções segmentadas'),
('Content Master', 250.00, 'Agendamento de posts, biblioteca de mídia'),
('Analytics Pro', 400.00, 'Heatmaps, funil de vendas, A/B testing'),
('Startup Special', 90.00, 'Mentoria, acesso a eventos exclusivos'),
('Nonprofit Bundle', 60.00, 'Doações integradas, storytelling tools'),
('Global Reach', 480.00, 'Tradução automática, multi-região'),
('Brand Guardian', 180.00, 'Monitoramento de marca, alertas de crise'),
('Ad Campaigner', 220.00, 'Gestão de anúncios, otimização em tempo real'),
('Data Driven', 380.00, 'Big data, machine learning insights'),
('Community Builder', 130.00, 'Fóruns moderados, gamificação'),
('VIP Experience', 499.00, 'Concierge 24/7, treinamentos exclusivos');
```

PESSOAL:

Descrição: 20 usuários pessoais.

Ex: ShadowHunter (Carlos Silva, 1995).

```
INSERT INTO pessoal (username, cpf, nome, data_nas, email, senha, endereco,
tel_contato) VALUES
('ShadowHunter', '12345678901', 'Carlos Silva', '1995-03-15',
'carlos@gmail.com', 'carlos1234', 'Rua das Flores, 123', '+5511987654321'),
('DragonSlayer', '23456789012', 'Ana Souza', '2000-07-22',
'ana@outlook.com', 'dragonana22', 'Avenida Paulista, 456',
'+5511912345678'),
('PixelWarrior', '34567890123', 'Pedro Rocha', '1988-11-05',
'pedro.pixel@gmail.com', 'pixel88rocha', NULL, '+5511123456789'),
('CyberNinja', '45678901234', 'Julia Lima', '1999-04-30',
'julia.ninja@outlook.com', 'ninja9999', 'Rua Tech, 789', '+5511987654321'),
('SteamLord', '56789012345', 'Lucas Oliveira', '2002-09-12',
'lucas.steam@gmail.com', 'steam2020', NULL, '+5511976543210'),
('MechaQueen', '67890123456', 'Fernanda Castro', '1993-12-25',
'fer.mecha@outlook.com', 'mechaXmas93', 'Alameda Santos, 101',
'+5511988776655'),
('CodeSamurai', '78901234567', 'Ricardo Kim', '1985-06-18',
'ricardo.code@gmail.com', 'samurai85', NULL, '+5511933344556'),
('PhoenixRider', '89012345678', 'Mariana Costa', '2005-01-07',
'mariana.phoenix@outlook.com', 'phoenix2005', 'Rua Java, 202',
'+5511966699887'),
('NeonGhost', '90123456789', 'Gabriel Alves', '1997-08-14',
'gabriel.neon@gmail.com', 'neon97gabs', 'Avenida Python, 303',
'+5511955544332'),
('VortexMage', '01234567890', 'Isabela Santos', '2001-02-28',
'isa.vortex@outlook.com', 'vortexisa28', NULL, '+5511944433221'),
('BladeRunner', '11223344556', 'Rodrigo Pereira', '1990-05-20',
'rodrigo.blade@gmail.com', 'runner1990', 'Rua C++, 404', '+5511922233445'),
('Starlight', '22334455667', 'Camila Rios', '2003-10-10',
'camila.star@outlook.com', 'starlight10', NULL, '+5511999988776'),
('IronPilot', '33445566778', 'Thiago Nunes', '1987-04-03',
```

```
'thiago.iron@gmail.com', 'pilot1987', 'Alameda Ruby, 505',
'+5511887766554'),
('SkyGuardian', '44556677889', 'Patricia Moraes', '1996-07-19',
'paty.sky@outlook.com', 'guardian96', NULL, '+5511876655443'),
('FrostWolf', '55667788990', 'Bruno Carvalho', '2004-12-01',
'bruno.frost@gmail.com', 'wolf2004br', 'Rua HTML, 606', '+5511866554332'),
('SolarFlare', '66778899001', 'Larissa Gomes', '1994-03-08',
'larissa.solar@outlook.com', 'flare1994', NULL, '+5511855443321'),
('ThunderBolt', '77889900112', 'Gustavo Henrique', '1989-09-23',
'gustavo.bolt@gmail.com', 'thunder89', 'Avenida CSS, 707',
'+5511844332211'),
('MoonKnight', '88990011223', 'Amanda Freitas', '2006-06-15',
'amanda.moon@outlook.com', 'moon15night', NULL, '+5511833221100'),
('DarkKnight', '99001112234', 'Roberto Dias', '1998-11-11',
'roberto.dark@gmail.com', 'dark1998rob', 'Rua JavaScript, 808',
'+5511822110099'),
('CyberPunk', '00112233445', 'Clara Ribeiro', '2007-04-04',
'clara.cyber@outlook.com', 'cyberclara04', NULL, '+5511811009988');
```

EMPRESARIAL:

Descrição: 20 empresas. Vinculadas a pacotes (id_pacote).

Ex: TechSolutions (Business Plus), EcomMaster (E-Commerce Pack).

```
INSERT INTO empresarial (username, cpf, nome, data_nas, email, senha,
endereco, tel_contato, nome_fantasia, cnpj, id_pacote) VALUES
('TechSolutions', '99988877766', 'Tech Solutions LTDA', '2010-05-20',
'contato@tech.com', 'tech12345', 'Avenida Inovação, 1000', '+551121234567',
'Tech Solutions', '123456789012', 4),
('InovaCorp', '88877766655', 'Inova Corp SA', '2015-08-12',
'inovacorp@outlook.com', 'inova2023', 'Rua Startup, 200', '+551131234567',
'Inova Corp', '234567890123', 1),
('DataDrivenCo', '77766655544', 'Data Driven Analytics', '2018-03-01',
'data@driven.com', 'data2024!', NULL, '+551141234567', 'Data Driven',
'345678901234', 18),
('EcomMaster', '66655544411', 'E-Commerce Masters', '2020-11-15',
'ecom@master.com', 'ecom2020#', 'Alameda Digital, 300', '+551151234567',
'Ecom Master', '456789012345', 8),
('SocialBee', '55544433311', 'SocialBee Marketing', '2012-07-30',
'hello@socialbee.com', 'beemarketing', 'Rua das Redes, 400',
'+551161234567', 'SocialBee', '567890123456', 7),
('BrandHub', '44433322255', 'Brand Hub Creative', '2019-04-25',
'brand@hub.com', 'brandhub99', NULL, '+551171234567', 'Brand Hub',
'678901234567', 16),
('CloudNest', '33322211199', 'Cloud Nest Tech', '2016-09-10',
'cloud@nest.com', 'cloudnest16', 'Avenida Nuvem, 500', '+551181234567',
'Cloud Nest', '789012345678', 3),
('FutureLabs', '22211100088', 'Future Labs Inc', '2022-01-05',
```

```
'labs@future.com', 'future2025', 'Rua Futuro, 600', '+551191234567', 'Future
Labs', '890123456789', 6),
('GreenTech', '11100099977', 'Green Tech SA', '2014-12-12',
'green@tech.com', 'greentech14', NULL, '+551101234567', 'Green Tech',
'901234567890', 12),
('CodeCraft', '00099988866', 'CodeCraft Devs', '2017-06-18',
'code@craft.com', 'codecraft17', 'Alameda Dev, 700', '+551112345678',
'CodeCraft', '012345678901', 5),
('AdVision', '99988877744', 'AdVision Media', '2011-02-28',
'ads@vision.com', 'advision2024', 'Rua Publicidade, 800', '+551122345678',
'AdVision', '123450987654', 17),
('HealthPlus', '88877766622', 'Health Plus Clinic', '2013-10-07',
'health@plus.com', 'healthplus13', NULL, '+551132345678', 'Health Plus',
'234561098765', 9),
('EduTech', '77766655511', 'EduTech Solutions', '2023-04-09',
'edu@tech.com', 'edutech2023', 'Avenida Educação, 900', '+551142345678',
'EduTech', '345672109876', 13),
('FoodieNet', '66655544433', 'Foodie Network', '2021-08-22',
'contact@foodie.com', 'foodie2021!', 'Rua Gastronomia, 1000',
'+551152345678', 'FoodieNet', '456783210987', 10),
('TravelMasters', '55544433322', 'Travel Masters Co', '2019-05-14',
'travel@masters.com', 'travelmasters19', NULL, '+551162345678', 'Travel
Masters', '567894321098', 14),
('FashionHub', '44433322211', 'Fashion Hub LTDA', '2015-03-03',
'fashion@hub.com', 'fashionhub15', 'Avenida Moda, 1100', '+551172345678',
'Fashion Hub', '678905432109', 2),
('FinTechX', '33322211100', 'FinTech X', '2020-07-19', 'fintechx@mail.com',
'fintechx2020', 'Rua Financeira, 1200', '+551182345678', 'FinTech X',
'789016543210', 11),
('AutoDrive', '22211100099', 'AutoDrive Systems', '2024-01-01',
'auto@drive.com', 'autodrive24', NULL, '+551192345678', 'AutoDrive',
'890127654321', 19),
('GameForge', '11100099988', 'GameForge Studios', '2018-11-11',
'game@forge.com', 'gameforge18', 'Rua dos Jogos, 1300', '+551102345678',
'GameForge', '901238765432', 20),
('BuildIt', '00099988877', 'Build It Construções', '2016-04-04',
'build@it.com', 'buildit2016', 'Avenida Obras, 1400', '+551112345679',
'Build It', '012349876543', 15);
```

USUARIO_CONTA:

Descrição: A tabela usuario_conta funciona como tabela pai para as tabelas pessoal e empresarial, utilizando herança. Ela deve armazenar informações comuns a qualquer tipo de usuário. Inicialmente, foram inseridos 40 registros: 20 usuários pessoais e 20 empresariais.

Durante o processo de inserção, enfrentamos alguns problemas relacionados à herança. Mesmo especificando corretamente a herança no momento da criação das tabelas filhas, os dados não estavam sendo automaticamente refletidos na tabela pai usuario_conta. Além disso, referências a usuario_conta a partir de outras tabelas, como jogo e comunidade, apresentavam erros devido à ausência de registros esperados na tabela pai.

Como solução, optamos por duplicar os dados manualmente na tabela usuario_conta com os comandos abaixo, garantindo a consistência e integridade do banco:

```
INSERT INTO usuario_conta (username, cpf, nome, data_nas, email, senha,
endereco, tel_contato)
SELECT username, cpf, nome, data_nas, email, senha, endereco, tel_contato
FROM pessoal;

INSERT INTO usuario_conta (username, cpf, nome, data_nas, email, senha,
endereco, tel_contato)
SELECT username, cpf, nome, data_nas, email, senha, endereco, tel_contato
FROM empresarial;
```

REQ_AMIZADE:

Descrição: 20 solicitações (10 aceitas, 5 pendentes, 5 recusadas).

Ex: ShadowHunter → DragonSlayer (Aceita).

```
INSERT INTO req_amizade (status, recebe_username, envia_username) VALUES
-- Solicitações ACEITAS (10)
('Aceita', 'ShadowHunter', 'DragonSlayer'),
('Aceita', 'PixelWarrior', 'CyberNinja'),
('Aceita', 'SteamLord', 'MechaQueen'),
('Aceita', 'CodeSamurai', 'PhoenixRider'),
('Aceita', 'NeonGhost', 'VortexMage'),
('Aceita', 'BladeRunner', 'Starlight'),
('Aceita', 'IronPilot', 'SkyGuardian'),
('Aceita', 'FrostWolf', 'SolarFlare'),
('Aceita', 'ThunderBolt', 'MoonKnight'),
('Aceita', 'DarkKnight', 'CyberPunk'),
-- Solicitações PENDENTES (5)
('Pendente', 'ShadowHunter', 'PixelWarrior'),
('Pendente', 'DragonSlayer', 'SteamLord'),
('Pendente', 'MechaQueen', 'NeonGhost'),
('Pendente', 'PhoenixRider', 'BladeRunner'),
('Pendente', 'VortexMage', 'IronPilot'),
-- Solicitações RECUSADAS (5)
('Recusada', 'CyberNinja', 'ShadowHunter'),
('Recusada', 'Starlight', 'DragonSlayer'),
```

```
('Recusada', 'SolarFlare', 'MechaQueen'),
('Recusada', 'MoonKnight', 'PhoenixRider'),
('Recusada', 'CyberPunk', 'ThunderBolt');
```

CHAT:

Descrição: 10 chat's com base no trigger definido tópico 4.2.4.

MENSAGEM:

Descrição: 30 mensagens em 10 chats.

Ex: ShadowHunter e DragonSlayer deram oi um para o outro.

```
-- Mensagens para o chat ShadowHunter & DragonSlayer
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('ShadowHunter', 'DragonSlayer', 'E aí, Carlos! Tudo bem?'),
('DragonSlayer', 'ShadowHunter', 'Oi Ana! Tudo ótimo, e com você?'),
('ShadowHunter', 'DragonSlayer', 'Tudo tranquilo! Vamos jogar hoje?');

-- Mensagens para o chat PixelWarrior & CyberNinja
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('PixelWarrior', 'CyberNinja', 'Julia, você viu o novo jogo que saiu?'),
('CyberNinja', 'PixelWarrior', 'Vi sim, Pedro! Parece incrível, quero
testar'),
('PixelWarrior', 'CyberNinja', 'Bora jogar juntos amanhã?');

-- Mensagens para o chat SteamLord & MechaQueen
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('SteamLord', 'MechaQueen', 'Fernanda, conseguiu passar daquela fase
difícil?'),
('MechaQueen', 'SteamLord', 'Ainda não, Lucas! Preciso de dicas'),
('SteamLord', 'MechaQueen', 'Te mando um tutorial que achei');

-- Mensagens para o chat CodeSamurai & PhoenixRider
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('CodeSamurai', 'PhoenixRider', 'Mariana, como está o projeto novo?'),
('PhoenixRider', 'CodeSamurai', 'Indo bem, Ricardo! Preciso de ajuda com o
código'),
('CodeSamurai', 'PhoenixRider', 'Posso te ajudar depois das 18h');

-- Mensagens para o chat NeonGhost & VortexMage
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
```

```
('NeonGhost', 'VortexMage', 'Isabela, marcamos de treinar hoje, lembra?'),
('VortexMage', 'NeonGhost', 'Sim, Gabriel! Às 20h no parque, certo?'),
('NeonGhost', 'VortexMage', 'Isso mesmo! Levo os equipamentos');

-- Mensagens para o chat BladeRunner & Starlight
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('BladeRunner', 'Starlight', 'Camila, já leu o livro que te indiquei?'),
('Starlight', 'BladeRunner', 'Ainda não, Rodrigo! Estou terminando outro'),
('BladeRunner', 'Starlight', 'Depois me diz o que achou quando ler');

-- Mensagens para o chat IronPilot & SkyGuardian
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('IronPilot', 'SkyGuardian', 'Patricia, viagem marcada para sexta!'),
('SkyGuardian', 'IronPilot', 'Que ótimo, Thiago! Já arrumei as malas'),
('IronPilot', 'SkyGuardian', 'Passo te buscar às 7h');

-- Mensagens para o chat FrostWolf & SolarFlare
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('FrostWolf', 'SolarFlare', 'Larissa, recebeu os documentos?'),
('SolarFlare', 'FrostWolf', 'Recebi sim, Bruno! Já enviei a revisão'),
('FrostWolf', 'SolarFlare', 'Perfeito, obrigado!');

-- Mensagens para o chat ThunderBolt & MoonKnight
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('ThunderBolt', 'MoonKnight', 'Amanda, vamos no show sábado?'),
('MoonKnight', 'ThunderBolt', 'Claro, Gustavo! Já comprei os ingressos'),
('ThunderBolt', 'MoonKnight', 'Show! Nos encontramos lá');

-- Mensagens para o chat DarkKnight & CyberPunk
INSERT INTO facepage.mensagem (chat_recebe_username, chat_envia_username,
conteudo) VALUES
('DarkKnight', 'CyberPunk', 'Clara, como foi na prova hoje?'),
('CyberPunk', 'DarkKnight', 'Foi bem, Roberto! Acertei quase tudo'),
('DarkKnight', 'CyberPunk', 'Parabéns! Merece comemorar');
```

PARTICIPA e GRUPO_AMIGOS:

Descrição GRUPO_AMIGOS: 20 grupos temáticos.

Ex: Clã dos Caçadores (jogadores hardcore), Code Masters (desenvolvedores).

Descrição PARTICIPA: 40 membros em grupos (2 por grupo).

Ex: DragonSlayer no Clã dos Caçadores.

```
-- Grupos do ShadowHunter (Amigo: DragonSlayer)
```

```sql
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Clã dos Caçadores', 'Grupo de jogadores hardcore', 'ShadowHunter'),
('Aliança Tech', 'Amantes de tecnologia', 'ShadowHunter');

INSERT INTO participa (id_grupo, amigo) VALUES
(1, 'DragonSlayer'), -- Grupo 1
(2, 'DragonSlayer'); -- Grupo 2

-- Grupos do DragonSlayer (Amigo: ShadowHunter)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Guilda das Feras', 'RPG e estratégia', 'DragonSlayer'),
('Equipe Pixel', 'Fãs de jogos retro', 'DragonSlayer');

INSERT INTO participa (id_grupo, amigo) VALUES
(3, 'ShadowHunter'), -- Grupo 3
(4, 'ShadowHunter'); -- Grupo 4

-- Grupos do PixelWarrior (Amigo: CyberNinja)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Mestres do Pixel', 'Jogos clássicos', 'PixelWarrior'),
('Cyber Squad', 'Competitivo online', 'PixelWarrior');

INSERT INTO participa (id_grupo, amigo) VALUES
(5, 'CyberNinja'), -- Grupo 5
(6, 'CyberNinja'); -- Grupo 6

-- Grupos do SteamLord (Amigo: MechaQueen)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Masters of Steam', 'Jogos de PC', 'SteamLord'),
('Indie Lovers', 'Jogos independentes', 'SteamLord');

INSERT INTO participa (id_grupo, amigo) VALUES
(7, 'MechaQueen'), -- Grupo 7
(8, 'MechaQueen'); -- Grupo 8

-- Grupos do CodeSamurai (Amigo: PhoenixRider)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Code Masters', 'Desenvolvedores de jogos', 'CodeSamurai'),
('Aventureiros Digitais', 'Exploradores de mundos virtuais', 'CodeSamurai');

INSERT INTO participa (id_grupo, amigo) VALUES
(9, 'PhoenixRider'), -- Grupo 9
(10, 'PhoenixRider'); -- Grupo 10

-- Grupos do NeonGhost (Amigo: VortexMage)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Neon Tribe', 'Jogos noturnos', 'NeonGhost'),
('Mago Guild', 'Magia e estratégia', 'NeonGhost');
```

```
INSERT INTO participa (id_grupo, amigo) VALUES
(11, 'VortexMage'), -- Grupo 11
(12, 'VortexMage'); -- Grupo 12

-- Grupos do BladeRunner (Amigo: Starlight)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Blade Alliance', 'Ação e aventura', 'BladeRunner'),
('Star Squad', 'Jogos espaciais', 'BladeRunner');

INSERT INTO participa (id_grupo, amigo) VALUES
(13, 'Starlight'), -- Grupo 13
(14, 'Starlight'); -- Grupo 14

-- Grupos do IronPilot (Amigo: SkyGuardian)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Iron Wings', 'Simuladores de voo', 'IronPilot'),
('Sky Guardians', 'Defensores do céu', 'IronPilot');

INSERT INTO participa (id_grupo, amigo) VALUES
(15, 'SkyGuardian'), -- Grupo 15
(16, 'SkyGuardian'); -- Grupo 16

-- Grupos do FrostWolf (Amigo: SolarFlare)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Frost Pack', 'Jogos de inverno', 'FrostWolf'),
('Solar Clan', 'Energia e estratégia', 'FrostWolf');

INSERT INTO participa (id_grupo, amigo) VALUES
(17, 'SolarFlare'), -- Grupo 17
(18, 'SolarFlare'); -- Grupo 18

-- Grupos do ThunderBolt (Amigo: MoonKnight)
INSERT INTO grupo_amigos (nome, descricao, admin) VALUES
('Thunder Force', 'Jogos de ação rápida', 'ThunderBolt'),
('Moon Knights', 'Aventuras noturnas', 'ThunderBolt');

INSERT INTO participa (id_grupo, amigo) VALUES
(19, 'MoonKnight'), -- Grupo 19
(20, 'MoonKnight'); -- Grupo 20
```

JOGO:

Descrição: 10 jogos de dificuldade 1 - 10.

Ex: Cyber Battle (FPS cyberpunk), Candy Kingdom (puzzle).

```
INSERT INTO facepage.jogo (nome, descricao, dificuldade) VALUES
('Space Warriors', 'Batalhas intergalácticas em tempo real', 7),
```

```
('Epic Quest', 'RPG de fantasia com mundos abertos', 6),
('Speed Racers', 'Corridas extremas em pistas futuristas', 5),
('Dragon Realm', 'Aventura épica para domar dragões', 8),
('Cyber Battle', 'FPS tático em cenários cyberpunk', 9),
('Farm Life Simulator', 'Gerencie sua fazenda e colheitas', 3),
('Puzzle Masters', 'Desafie sua lógica com quebra-cabeças complexos', 4),
('Soccer Stars', 'Simulador de futebol com ligas globais', 6),
('Zombie Survival', 'Sobreviva a hordas de zumbis pós-apocalípticos', 8),
('Candy Kingdom', 'Combine doces em puzzles coloridos', 2);
```

JOGA:

Descrição: 20 pontuações.

Ex: TechSolutions (9800 em Cyber Battle), MechaQueen (7800 em Candy Kingdom).

```
INSERT INTO facepage.joga (nome_jogo, username, pontuacao) VALUES
-- ShadowHunter (pessoal)
('Space Warriors', 'ShadowHunter', 1500),
('Epic Quest', 'ShadowHunter', 3000),

-- DragonSlayer (pessoal)
('Speed Racers', 'DragonSlayer', 2500),
('Dragon Realm', 'DragonSlayer', 4200),

-- TechSolutions (empresarial)
('Cyber Battle', 'TechSolutions', 9800),
('Farm Life Simulator', 'TechSolutions', 1200),

-- InovaCorp (empresarial)
('Puzzle Masters', 'InovaCorp', 6700),
('Soccer Stars', 'InovaCorp', 3300),

-- MechaQueen (pessoal)
('Zombie Survival', 'MechaQueen', 5500),
('Candy Kingdom', 'MechaQueen', 7800),

-- CloudNest (empresarial)
('Space Warriors', 'CloudNest', 2100),
('Speed Racers', 'CloudNest', 4500),

-- NeonGhost (pessoal)
('Epic Quest', 'NeonGhost', 8800),
('Dragon Realm', 'NeonGhost', 6100),

-- BrandHub (empresarial)
('Cyber Battle', 'BrandHub', 3200),
('Puzzle Masters', 'BrandHub', 7600),
```

```
-- ThunderBolt (pessoal)
('Farm Life Simulator', 'ThunderBolt', 2900),
('Soccer Stars', 'ThunderBolt', 5300),

-- EduTech (empresarial)
('Zombie Survival', 'EduTech', 6700),
('Candy Kingdom', 'EduTech', 8900);
```

RANKING:

Descrição: 2 rankings (20 no total, pois são 10 jogos). Ela foi feita como visão e não está no arquivo de inserção.

```
-- Visão para rankings de jogos (agregação)
CREATE OR REPLACE VIEW ranking AS
SELECT
    j.nome AS jogo,
    jg.username,
    jg.pontuacao,
    RANK() OVER (PARTITION BY j.nome ORDER BY jg.pontuacao DESC) AS
posicao_ranking
FROM
    joga jg
JOIN
    jogo j ON jg.nome_jogo = j.nome
ORDER BY
    j.nome, posicao_ranking;
```

COMUNIDADE:

Descrição: 10 comunidades.

Ex: Tech Innovators (tecnologia), Book Club (literatura).

```
-- Criando as comunidades

INSERT INTO facepage.comunidade (nome_comunidade, descricao, admin) VALUES
('Tech Innovators', 'Discussões sobre inovações tecnológicas',
'TechSolutions'),
('Gaming Legends', 'Comunidade para amantes de jogos clássicos e modernos',
'ShadowHunter'),
('Music Lovers', 'Compartilhamento de músicas e playlists', 'DragonSlayer'),
('E-Commerce Masters', 'Dicas e estratégias para negócios online',
'EcomMaster'),
('Fitness & Health', 'Dicas de exercícios, dieta e bem-estar', 'NeonGhost'),
('Startup Hub', 'Networking e mentoria para startups', 'InovaCorp'),
('Art & Design', 'Arte digital, design gráfico e ilustração', 'MechaQueen'),
('Travel Explorers', 'Relatos de viagens e dicas de destinos',
```

```
'ThunderBolt'),
('Book Club', 'Recomendações e discussões literárias', 'Starlight'),
('Foodie Paradise', 'Receitas, restaurantes e gastronomia', 'SolarFlare');
```

MEMBROS_COMUNIDADE:

Descrição: 50 membros (5 por comunidade).

Ex: CloudNest em Tech Innovators.

```
INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Tech Innovators', 'TechSolutions'), -- Admin
('Tech Innovators', 'CloudNest'),
('Tech Innovators', 'CodeCraft'),
('Tech Innovators', 'DataDrivenCo'),
('Tech Innovators', 'FutureLabs');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Gaming Legends', 'ShadowHunter'), -- Admin
('Gaming Legends', 'DragonSlayer'),
('Gaming Legends', 'PixelWarrior'),
('Gaming Legends', 'CyberNinja'),
('Gaming Legends', 'SteamLord');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Music Lovers', 'DragonSlayer'), -- Admin
('Music Lovers', 'VortexMage'),
('Music Lovers', 'BladeRunner'),
('Music Lovers', 'IronPilot'),
('Music Lovers', 'MoonKnight');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('E-Commerce Masters', 'EcomMaster'), -- Admin
('E-Commerce Masters', 'BrandHub'),
('E-Commerce Masters', 'SocialBee'),
('E-Commerce Masters', 'AdVision'),
('E-Commerce Masters', 'FinTechX');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Fitness & Health', 'NeonGhost'), -- Admin
('Fitness & Health', 'SkyGuardian'),
('Fitness & Health', 'SolarFlare'),
('Fitness & Health', 'FrostWolf'),
('Fitness & Health', 'PhoenixRider');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Startup Hub', 'InovaCorp'), -- Admin
('Startup Hub', 'GreenTech'),
('Startup Hub', 'EduTech'),
```

```
('Startup Hub', 'TravelMasters'),
('Startup Hub', 'BuildIt');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Art & Design', 'MechaQueen'), -- Admin
('Art & Design', 'Starlight'),
('Art & Design', 'CyberPunk'),
('Art & Design', 'DarkKnight'),
('Art & Design', 'CodeSamurai');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Travel Explorers', 'ThunderBolt'), -- Admin
('Travel Explorers', 'MoonKnight'),
('Travel Explorers', 'SkyGuardian'),
('Travel Explorers', 'IronPilot'),
('Travel Explorers', 'BladeRunner');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Book Club', 'Starlight'), -- Admin
('Book Club', 'VortexMage'),
('Book Club', 'NeonGhost'),
('Book Club', 'PhoenixRider'),
('Book Club', 'SolarFlare');

INSERT INTO facepage.membros_comunidade (comunidade, membro) VALUES
('Foodie Paradise', 'SolarFlare'), -- Admin
('Foodie Paradise', 'FoodieNet'),
('Foodie Paradise', 'HealthPlus'),
('Foodie Paradise', 'ThunderBolt'),
('Foodie Paradise', 'DragonSlayer');
```

TOPICO:

Descrição: 20 tópicos (2 por comunidade).

Ex: "O Futuro da IA em 2024" (Tech Innovators).

```
-- Comunidade 1: Tech Innovators (Descrição: Discussões sobre inovações
tecnológicas)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('TechSolutions', 'Tech Innovators', 'O Futuro da IA em 2024', '2023-11-01',
'Como a inteligência artificial está transformando indústrias em 2024?'),
('DataDrivenCo', 'Tech Innovators', 'Blockchain para Segurança Digital',
'2023-11-05', 'Aplicações práticas de blockchain em proteção de dados.');

-- Comunidade 2: Gaming Legends (Descrição: Comunidade para amantes de jogos
clássicos e modernos)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
```

```
VALUES
('ShadowHunter', 'Gaming Legends', 'Melhores Jogos Retro de 2023',
'2023-10-15', 'Lista dos jogos retrô mais populares deste ano.'),
('CyberNinja', 'Gaming Legends', 'eSports: Cenário Competitivo Atual',
'2023-10-20', 'Discussão sobre torneios e equipes dominantes.');

-- Comunidade 3: Music Lovers (Descrição: Compartilhamento de músicas e
playlists)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('DragonSlayer', 'Music Lovers', 'Playlists para Estudo/Concentração',
'2023-09-10', 'Indicações de músicas instrumentais e lo-fi.'),
('VortexMage', 'Music Lovers', 'Novos Lançamentos de Rock Alternativo',
'2023-09-12', 'Bandas emergentes e álbuns do gênero.');

-- Comunidade 4: E-Commerce Masters (Descrição: Dicas e estratégias para
negócios online)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('EcomMaster', 'E-Commerce Masters', 'SEO para Lojas Virtuais',
'2023-08-01', 'Como otimizar seu e-commerce para buscadores.'),
('SocialBee', 'E-Commerce Masters', 'Marketing em Redes Sociais',
'2023-08-05', 'Estratégias para impulsionar vendas no Instagram.');

-- Comunidade 5: Fitness & Health (Descrição: Dicas de exercícios, dieta e
bem-estar)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('NeonGhost', 'Fitness & Health', 'Rotina de Exercícios em Casa',
'2023-07-20', 'Treinos eficientes sem equipamentos.'),
('SkyGuardian', 'Fitness & Health', 'Dietas Plant-Based para Iniciantes',
'2023-07-25', 'Guia prático para transição alimentar.');

-- Comunidade 6: Startup Hub (Descrição: Networking e mentoria para
startups)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('InovaCorp', 'Startup Hub', 'Como Captar Investimento-Anjo', '2023-06-05',
'Passo a passo para pitchs convincentes.'),
('GreenTech', 'Startup Hub', 'Sustentabilidade em Startups Tech',
'2023-06-10', 'Cases de sucesso e desafios.');

-- Comunidade 7: Art & Design (Descrição: Arte digital, design gráfico e
ilustração)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('MechaQueen', 'Art & Design', 'Ferramentas para Designers em 2023',
'2023-05-15', 'Comparativo entre Photoshop, Figma e Canva.'),
```

```
('CodeSamurai', 'Art & Design', 'Arte Generativa com IA', '2023-05-18',
'Como criar arte usando MidJourney e Stable Diffusion.');

-- Comunidade 8: Travel Explorers (Descrição: Relatos de viagens e dicas de
destinos)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('ThunderBolt', 'Travel Explorers', 'Melhores Destinos para Mochilão',
'2023-04-01', 'Roteiros econômicos na América do Sul.'),
('IronPilot', 'Travel Explorers', 'Viagens Sustentáveis: Como Reduzir
Impacto', '2023-04-05', 'Dicas para turismo consciente.');

-- Comunidade 9: Book Club (Descrição: Recomendações e discussões
literárias)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('Starlight', 'Book Club', 'Livros Distópicos para 2024', '2023-03-10',
'Indicações de ficção científica e distopias.'),
('PhoenixRider', 'Book Club', 'Clássicos da Literatura Brasileira',
'2023-03-12', 'Releituras contemporâneas de Machado de Assis.');

-- Comunidade 10: Foodie Paradise (Descrição: Receitas, restaurantes e
gastronomia)
INSERT INTO facepage.topico (membro, comunidade, titulo, data, descricao)
VALUES
('SolarFlare', 'Foodie Paradise', 'Receitas com Ingredientes Locais',
'2023-02-01', 'Como aproveitar produtos regionais.'),
('FoodieNet', 'Foodie Paradise', 'Top 10 Restaurantes Veganos no Brasil',
'2023-02-05', 'Avaliação de estabelecimentos especializados.');
```

COMENTAM:

Descrição: 40 comentários (2 por tópico).

Ex: Debate sobre ética da IA em Tech Innovators.

```
-----------------------------
-- Comunidade: Tech Innovators
-----------------------------
-- Tópico 1: O Futuro da IA em 2024 (codigo = 1)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('A IA está revolucionando até a medicina! Empresas como a DeepMind já fazem
diagnósticos precisos.', 'TechSolutions', 1, 'Tech Innovators', 'Tech
Innovators', 'CloudNest'),
('Precisamos debater ética no uso de IA. Alguém conhece frameworks para
regulamentação?', 'TechSolutions', 1, 'Tech Innovators', 'Tech Innovators',
'CodeCraft');
```

```
-- Tópico 2: Blockchain para Segurança Digital (codigo = 2)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Blockchain é essencial para evitar fraudes em contratos inteligentes.
Ótimo tópico!', 'DataDrivenCo', 2, 'Tech Innovators', 'Tech Innovators',
'FutureLabs'),
('Alguém já testou a plataforma Hyperledger para implementação?
Recomendam?', 'DataDrivenCo', 2, 'Tech Innovators', 'Tech Innovators',
'TechSolutions');

-----------------------------
-- Comunidade: Gaming Legends
-----------------------------
-- Tópico 3: Melhores Jogos Retro de 2023 (codigo = 3)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Adoro jogos retrô! Recomendo "Celeste" para quem gosta de plataforma.',
'ShadowHunter', 3, 'Gaming Legends', 'Gaming Legends', 'DragonSlayer'),
('Alguém vai participar do campeonato de Street Fighter VI?',
'ShadowHunter', 3, 'Gaming Legends', 'Gaming Legends', 'CyberNinja');

-- Tópico 4: eSports: Cenário Competitivo Atual (codigo = 4)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('A equipe Furia está dominando o cenário brasileiro de CS:GO!',
'CyberNinja', 4, 'Gaming Legends', 'Gaming Legends', 'SteamLord'),
('Precisamos de mais investimento em equipes femininas de eSports.',
'CyberNinja', 4, 'Gaming Legends', 'Gaming Legends', 'PixelWarrior');

-----------------------------
-- Comunidade: Music Lovers
-----------------------------
-- Tópico 5: Playlists para Estudo/Concentração (codigo = 5)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Playlist "Lo-Fi Beats" no Spotify é minha favorita!', 'DragonSlayer', 5,
'Music Lovers', 'Music Lovers', 'BladeRunner'),
('Música clássica também ajuda muito. Recomendo Bach.', 'DragonSlayer', 5,
'Music Lovers', 'Music Lovers', 'IronPilot');

-- Tópico 6: Novos Lançamentos de Rock Alternativo (codigo = 6)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('A banda "Arctic Monkeys" lançou algo novo?', 'VortexMage', 6, 'Music
Lovers', 'Music Lovers', 'MoonKnight'),
('Confira o álbum "The Now Now" do Gorillaz!', 'VortexMage', 6, 'Music
Lovers', 'Music Lovers', 'VortexMage');
```

```
-----------------------------
-- Comunidade: E-Commerce Masters
-----------------------------
-- Tópico 7: SEO para Lojas Virtuais (codigo = 7)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Ferramentas como SEMrush são ótimas para keywords.', 'EcomMaster', 7,
'E-Commerce Masters', 'E-Commerce Masters', 'SocialBee'),
('Conteúdo relevante ainda é o rei do SEO!', 'EcomMaster', 7, 'E-Commerce
Masters', 'E-Commerce Masters', 'AdVision');

-- Tópico 8: Marketing em Redes Sociais (codigo = 8)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Instagram Stories aumenta o engajamento em 40%.', 'SocialBee', 8,
'E-Commerce Masters', 'E-Commerce Masters', 'FinTechX'),
('TikTok para negócios está subestimado!', 'SocialBee', 8, 'E-Commerce
Masters', 'E-Commerce Masters', 'EcomMaster');

-----------------------------
-- Comunidade: Fitness & Health
-----------------------------
-- Tópico 9: Rotina de Exercícios em Casa (codigo = 9)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Treino com elásticos é ótimo para iniciantes.', 'NeonGhost', 9, 'Fitness &
Health', 'Fitness & Health', 'PhoenixRider'),
('Yoga também conta como exercício?', 'NeonGhost', 9, 'Fitness & Health',
'Fitness & Health', 'SolarFlare');

-- Tópico 10: Dietas Plant-Based para Iniciantes (codigo = 10)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Comecei com a Segunda Sem Carne e estou amando!', 'SkyGuardian', 10,
'Fitness & Health', 'Fitness & Health', 'FrostWolf'),
('Onde encontrar proteína vegetal barata?', 'SkyGuardian', 10, 'Fitness &
Health', 'Fitness & Health', 'NeonGhost');

-----------------------------
-- Comunidade: Startup Hub
-----------------------------
-- Tópico 11: Como Captar Investimento-Anjo (codigo = 11)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('O pitch precisa ser claro em 3 minutos!', 'InovaCorp', 11, 'Startup Hub',
'Startup Hub', 'GreenTech'),
('Busquem aceleradoras como Y Combinator.', 'InovaCorp', 11, 'Startup Hub',
'Startup Hub', 'EduTech');
```

```sql
-- Tópico 12: Sustentabilidade em Startups Tech (codigo = 12)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Energia solar reduziu nossos custos em 30%.', 'GreenTech', 12, 'Startup
Hub', 'Startup Hub', 'TravelMasters'),
('Como medir o impacto ambiental da startup?', 'GreenTech', 12, 'Startup
Hub', 'Startup Hub', 'BuildIt');

-----------------------------
-- Comunidade: Art & Design
-----------------------------
-- Tópico 13: Ferramentas para Designers em 2023 (codigo = 13)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Figma é insubstituível para colaboração em tempo real.', 'MechaQueen', 13,
'Art & Design', 'Art & Design', 'DarkKnight'),
('Alguém usa Affinity Designer aqui?', 'MechaQueen', 13, 'Art & Design',
'Art & Design', 'CodeSamurai');

-- Tópico 14: Arte Generativa com IA (codigo = 14)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('MidJourney cria artes incríveis para capas de livros.', 'CodeSamurai', 14,
'Art & Design', 'Art & Design', 'Starlight'),
('Como evitar violação de direitos autorais?', 'CodeSamurai', 14, 'Art &
Design', 'Art & Design', 'CyberPunk');

-----------------------------
-- Comunidade: Travel Explorers
-----------------------------
-- Tópico 15: Melhores Destinos para Mochilão (codigo = 15)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Patagônia é imperdível para aventuras radicais!', 'ThunderBolt', 15,
'Travel Explorers', 'Travel Explorers', 'MoonKnight'),
('Alguém já fez mochilão na Europa Oriental?', 'ThunderBolt', 15, 'Travel
Explorers', 'Travel Explorers', 'IronPilot');

-- Tópico 16: Viagens Sustentáveis: Como Reduzir Impacto (codigo = 16)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Hotéis eco-friendly são a melhor opção.', 'IronPilot', 16, 'Travel
Explorers', 'Travel Explorers', 'BladeRunner'),
('Evitem plástico descartável durante as viagens!', 'IronPilot', 16, 'Travel
Explorers', 'Travel Explorers', 'SkyGuardian');

-----------------------------
```

```
-- Comunidade: Book Club
-----------------------------
-- Tópico 17: Livros Distópicos para 2024 (codigo = 17)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('"1984" do Orwell continua atualíssimo.', 'Starlight', 17, 'Book Club',
'Book Club', 'PhoenixRider'),
('Leiam "The Handmaid''s Tale"!', 'Starlight', 17, 'Book Club', 'Book Club',
'NeonGhost');

-- Tópico 18: Clássicos da Literatura Brasileira (codigo = 18)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Machado de Assis é genial em "Dom Casmurro".', 'PhoenixRider', 18, 'Book
Club', 'Book Club', 'SolarFlare'),
('Alguém já leu "Grande Sertão: Veredas"?', 'PhoenixRider', 18, 'Book Club',
'Book Club', 'VortexMage');

-----------------------------
-- Comunidade: Foodie Paradise
-----------------------------
-- Tópico 19: Receitas com Ingredientes Locais (codigo = 19)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('Açaí com peixe frito é uma delícia no Pará!', 'SolarFlare', 19, 'Foodie
Paradise', 'Foodie Paradise', 'FoodieNet'),
('Como usar mandioca em pratos sofisticados?', 'SolarFlare', 19, 'Foodie
Paradise', 'Foodie Paradise', 'HealthPlus');

-- Tópico 20: Top 10 Restaurantes Veganos no Brasil (codigo = 20)
INSERT INTO facepage.comentam (comentario, membro_postou, codigo_topico,
comunidade_topico, comunidade_membro, membro_comentou) VALUES
('O "Pop Vegan" em SP é incrível!', 'FoodieNet', 20, 'Foodie Paradise',
'Foodie Paradise', 'ThunderBolt'),
('Falta mais opções veganas no Nordeste.', 'FoodieNet', 20, 'Foodie
Paradise', 'Foodie Paradise', 'DragonSlayer');
```

PAGINA:

Descrição: 10 páginas (5 pessoais, 5 empresariais).

Ex: Caçadores de Sombras (ShadowHunter), Tech Solutions Oficial.

```
-- Inserções para páginas pessoais
INSERT INTO facepage.pagina (nome, nro_visitas, criador) VALUES
('Caçadores de Sombras', 0, 'ShadowHunter'),
('Domadores de Dragões', 0, 'DragonSlayer'),
('Guerreiros do Pixel', 0, 'PixelWarrior'),
('Dojo CyberNinja', 0, 'CyberNinja'),
```

```
('Reino do Steam', 0, 'SteamLord');

-- Inserções para páginas empresariais
INSERT INTO facepage.pagina (nome, nro_visitas, criador) VALUES
('Tech Solutions Oficial', 0, 'TechSolutions'),
('Inova Corp Hub', 0, 'InovaCorp'),
('Ecom Master Digital', 0, 'EcomMaster'),
('SocialBee Marketing', 0, 'SocialBee'),
('Cloud Nest Tech', 0, 'CloudNest');
```

PROPAGANDA:

Descrição: 5 anúncios.

Ex: Anúncios Tech Solutions (taxa de cliques 25%).

```
-- Inserções para propagandas (herda de página)
INSERT INTO facepage.propaganda (nome, nro_visitas, criador, taxa_cliques,
taxa_rejeicao, taxa_conversao, admin) VALUES
('Anúncios Tech Solutions', 0, 'TechSolutions', 0.25, 0.15, 0.20,
'TechSolutions'),
('Promo Inova Corp', 0, 'InovaCorp', 0.18, 0.12, 0.28, 'InovaCorp'),
('Ofertas Ecom Master', 0, 'EcomMaster', 0.22, 0.10, 0.25, 'EcomMaster'),
('Campanhas SocialBee', 0, 'SocialBee', 0.30, 0.20, 0.15, 'SocialBee'),
('Cloud Nest Ads', 0, 'CloudNest', 0.27, 0.18, 0.22, 'CloudNest');
```

AÇÃO:

Descrição: 10 ações.

Ex: Criar postagens, Moderar conteúdo.

```
-- Inserindo ações genéricas para interação com páginas
INSERT INTO facepage.acao (descricao) VALUES
('Criar postagens na página'),
('Editar postagens existentes'),
('Excluir postagens da página'),
('Gerenciar membros da página'),
('Alterar configurações de visibilidade'),
('Publicar anúncios patrocinados'),
('Visualizar métricas de engajamento'),
('Responder a comentários'),
('Moderar conteúdo ofensivo'),
('Atualizar informações da página');
```

PRIVILEGIOS:

Descrição: 8 permissões com base em uma das 10 ações possíveis.

Ex: DragonSlayer pode editar postagens em páginas.

```
-- Atribuir privilégios apenas aos amigos que participam dos grupos
vinculados às páginas
INSERT INTO privilegios (usuario, nome, criador, codigo_acao) VALUES
------------------------------
-- Página: Caçadores de Sombras (Criador: ShadowHunter)
------------------------------
-- Grupo 1: Clã dos Caçadores (Membros: DragonSlayer)
('DragonSlayer', 'Caçadores de Sombras', 'ShadowHunter', 1),  -- Ação 1:
Criar postagens
-- Grupo 2: Aliança Tech (Membros: DragonSlayer)
('DragonSlayer', 'Caçadores de Sombras', 'ShadowHunter', 2),  -- Ação 2:
Editar postagens

------------------------------
-- Página: Domadores de Dragões (Criador: DragonSlayer)
------------------------------
-- Grupo 3: Guilda das Feras (Membros: ShadowHunter)
('ShadowHunter', 'Domadores de Dragões', 'DragonSlayer', 3),  -- Ação 3:
Excluir postagens
-- Grupo 4: Equipe Pixel (Membros: ShadowHunter)
('ShadowHunter', 'Domadores de Dragões', 'DragonSlayer', 4),  -- Ação 4:
Gerenciar membros

------------------------------
-- Página: Guerreiros do Pixel (Criador: PixelWarrior)
------------------------------
-- Grupo 5: Mestres do Pixel (Membros: CyberNinja)
('CyberNinja', 'Guerreiros do Pixel', 'PixelWarrior', 5),     -- Ação 5:
Publicar anúncios
-- Grupo 6: Cyber Squad (Membros: CyberNinja)
('CyberNinja', 'Guerreiros do Pixel', 'PixelWarrior', 6),     -- Ação 6:
Visualizar métricas

------------------------------
-- Página: Reino do Steam (Criador: SteamLord)
------------------------------
-- Grupo 7: Masters of Steam (Membros: MechaQueen)
('MechaQueen', 'Reino do Steam', 'SteamLord', 7),           -- Ação 7:
Responder a comentários
-- Grupo 8: Indie Lovers (Membros: MechaQueen)
('MechaQueen', 'Reino do Steam', 'SteamLord', 8);           -- Ação 8:
Moderar conteúdo
```

PODE_VER:

Descrição: 8 grupos com acesso a páginas.

Ex: Grupo Clã dos Caçadores vê página Caçadores de Sombras.

```
INSERT INTO pode_ver (nome, criador, codigo_grupo) VALUES
-- Página: Caçadores de Sombras (Criador: ShadowHunter)
('Caçadores de Sombras', 'ShadowHunter', 1),  -- Grupo 1: Clã dos Caçadores
(admin: ShadowHunter)
('Caçadores de Sombras', 'ShadowHunter', 2),  -- Grupo 2: Aliança Tech
(admin: ShadowHunter)

-- Página: Domadores de Dragões (Criador: DragonSlayer)
('Domadores de Dragões', 'DragonSlayer', 3),  -- Grupo 3: Guilda das Feras
(admin: DragonSlayer)
('Domadores de Dragões', 'DragonSlayer', 4),  -- Grupo 4: Equipe Pixel
(admin: DragonSlayer)

-- Página: Guerreiros do Pixel (Criador: PixelWarrior)
('Guerreiros do Pixel', 'PixelWarrior', 5),   -- Grupo 5: Mestres do Pixel
(admin: PixelWarrior)
('Guerreiros do Pixel', 'PixelWarrior', 6),   -- Grupo 6: Cyber Squad
(admin: PixelWarrior)

-- Página: Reino do Steam (Criador: SteamLord)
('Reino do Steam', 'SteamLord', 7),           -- Grupo 7: Masters of Steam
(admin: SteamLord)
('Reino do Steam', 'SteamLord', 8);           -- Grupo 8: Indie Lovers
(admin: SteamLord)
```

4.2.2. Consultas e Atualizações em SQL

Exibe as principais consultas realizadas utilizando SQL, com ênfase em comandos SELECT, incluindo junções, filtros, ordenações e agrupamentos. Também são mostrados exemplos de comandos UPDATE e DELETE, utilizados para atualizar ou remover dados, sempre respeitando as regras de integridade do sistema. As consultas com maior complexidade foram priorizadas pelo nosso grupo.

CONSULTAS

1. Mostrar o nome de cada comunidade, os membros presente nela, a quantidade de membros por comunidade, o total de membros de toda a comunidade e a quantidade de comunidades que cada membro participa (em uma mesma consulta).

```
SELECT comunidade, membro, count(*) FROM membros_comunidade
GROUP BY CUBE(comunidade, membro)
ORDER BY comunidade, membro;
```

Resultado:

comunidade	membro	count
Art & Design	NULL	5
Art & Design	Starlight	1
Art & Design	MechaQueen	1
Art & Design	DarkKnight	1
Art & Design	CyberPunk	1
Art & Design	CodeSamurai	1
Book Club	NULL	5
Book Club	VortexMage	1
Book Club	Starlight	1
Book Club	SolarFlare	1
Book Club	PhoenixRider	1
Book Club	NeonGhost	1
E-Commerce Masters	NULL	5
E-Commerce Masters	SocialBee	1
E-Commerce Masters	FinTechX	1
E-Commerce Masters	EcomMaster	1
E-Commerce Masters	BrandHub	1
E-Commerce Masters	AdVision	1
Fitness & Health	NULL	5
Fitness & Health	SolarFlare	1
Fitness & Health	SkyGuardian	1
Fitness & Health	PhoenixRider	1
Fitness & Health	NeonGhost	1
Fitness & Health	FrostWolf	1
Foodie Paradise	NULL	5
Foodie	ThunderBolt	1

Paradise		
Foodie Paradise	SolarFlare	1
Foodie Paradise	HealthPlus	1
Foodie Paradise	FoodieNet	1
Foodie Paradise	DragonSlayer	1
Gaming Legends	NULL	5
Gaming Legends	SteamLord	1
Gaming Legends	ShadowHunter	1
Gaming Legends	PixelWarrior	1
Gaming Legends	DragonSlayer	1
Gaming Legends	CyberNinja	1
Music Lovers	NULL	5
Music Lovers	VortexMage	1
Music Lovers	MoonKnight	1
Music Lovers	IronPilot	1
Music Lovers	DragonSlayer	1
Music Lovers	BladeRunner	1
Startup Hub	NULL	5
Startup Hub	TravelMasters	1
Startup Hub	InovaCorp	1
Startup Hub	GreenTech	1
Startup Hub	EduTech	1
Startup Hub	BuildIt	1
Tech Innovators	NULL	5
Tech Innovators	TechSolutions	1
Tech Innovators	FutureLabs	1
Tech Innovators	DataDrivenCo	1

Tech Innovators	CodeCraft	1
Tech Innovators	CloudNest	1
Travel Explorers	NULL	5
Travel Explorers	ThunderBolt	1
Travel Explorers	SkyGuardian	1
Travel Explorers	MoonKnight	1
Travel Explorers	IronPilot	1
Travel Explorers	BladeRunner	1
NULL	NULL	50
NULL	VortexMage	2
NULL	TravelMasters	1
NULL	ThunderBolt	2
NULL	TechSolutions	1
NULL	SteamLord	1
NULL	Starlight	2
NULL	SolarFlare	3
NULL	SocialBee	1
NULL	SkyGuardian	2
NULL	ShadowHunter	1
NULL	PixelWarrior	1
NULL	PhoenixRider	2
NULL	NeonGhost	2
NULL	MoonKnight	2
NULL	MechaQueen	1
NULL	IronPilot	2
NULL	InovaCorp	1
NULL	HealthPlus	1
NULL	GreenTech	1
NULL	FutureLabs	1
NULL	FrostWolf	1
NULL	FoodieNet	1
NULL	FinTechX	1

NULL	EduTech	1
NULL	EcomMaster	1
NULL	DragonSlayer	3
NULL	DataDrivenCo	1
NULL	DarkKnight	1
NULL	CyberPunk	1
NULL	CyberNinja	1
NULL	CodeSamurai	1
NULL	CodeCraft	1
NULL	CloudNest	1
NULL	BuildIt	1
NULL	BrandHub	1
NULL	BladeRunner	2
NULL	AdVision	1

2. Mostrar quantas contas empresariais e quantas pessoais existem na rede social.

```
SELECT
    (SELECT COUNT(*) FROM empresarial) AS quantidade_empresarial,
    (SELECT COUNT(*) FROM pessoal) AS quantidade_pessoais;
```

Resultado:

quantidade_empresarial	quantidade_pessoais
20	20

3. Mostrar qual conta tem mais páginas associadas ao perfil.

```
SELECT u.username, COUNT(*) AS total_paginas
FROM usuario_conta u
    INNER JOIN pagina p ON u.username = p.criador
GROUP BY u.username
ORDER BY total_paginas DESC
LIMIT 1;
```

Resultado:

username	total_paginas
EcomMaster	4

4. Mostrar a pontuação do ranking, o jogo correspondente e o usuário que realizou (Separado por jogo).

```
SELECT jogo, username, pontuacao AS maior_pontuacao
FROM ranking
WHERE posicao_ranking = 1
ORDER BY jogo;
```

Resultado:

jogo	username	maior_pontuacao
Candy Kingdom	EduTech	8900
Cyber Battle	TechSolutions	9800
Dragon Realm	NeonGhost	6100
Epic Quest	NeonGhost	8800
Farm Life Simulator	ThunderBolt	2900
Puzzle Masters	BrandHub	7600
Soccer Stars	ThunderBolt	5300
Space Warriors	CloudNest	2100
Speed Racers	CloudNest	4500
Zombie Survival	EduTech	6700

5. Menor pontuação do ranking, o jogo correspondente e o usuário que realizou (Separado por jogo).

```
SELECT jogo, username, pontuacao AS menor_pontuacao
FROM ranking R
WHERE posicao_ranking = (
    SELECT MAX(posicao_ranking)
    FROM ranking R2
    WHERE R2.jogo = R.jogo
)
ORDER BY jogo;
```

Resultado:

Candy Kingdom	MechaQueen	7800
Cyber Battle	BrandHub	3200
Dragon Realm	DragonSlayer	4200
Epic Quest	ShadowHunter	3000

Farm Life Simulator	TechSolutions	1200
Puzzle Masters	InovaCorp	6700
Soccer Stars	InovaCorp	3300
Space Warriors	ShadowHunter	1500
Speed Racers	DragonSlayer	2500
Zombie Survival	MechaQueen	5500

6. Mensagens do chat com número de mensagens entre 2 e 10.

```
SELECT chat_envia_username, chat_recebe_username
FROM mensagem
GROUP BY chat_envia_username, chat_recebe_username
HAVING COUNT(*) BETWEEN 2 AND 10
ORDER BY chat_envia_username, chat_recebe_username;
```

Resultado:

chat_envia_username	chat_recebe_username
CyberNinja	PixelWarrior
CyberPunk	DarkKnight
DragonSlayer	ShadowHunter
MechaQueen	SteamLord
MoonKnight	ThunderBolt
PhoenixRider	CodeSamurai
SkyGuardian	IronPilot
SolarFlare	FrostWolf
Starlight	BladeRunner
VortexMage	NeonGhost

7. Descrição e título da comunidade com mais membros.

```
CREATE VIEW contagem_membros_comunidades AS
    SELECT comunidade, COUNT(*) AS total_membros
    FROM membros_comunidade
    GROUP BY comunidade;

CREATE VIEW comunidades_mais_populares AS
    SELECT comunidade, total_membros
    FROM contagem_membros_comunidades
    WHERE total_membros = (
```

```
        SELECT MAX(total_membros)
        FROM contagem_membros_comunidades
    );

SELECT c.nome_comunidade, c.descricao, cmp.total_membros
FROM comunidade c
    INNER JOIN comunidades_mais_populares cmp
        ON c.nome_comunidade = cmp.comunidade;
```

Resultado:

nome_comunidade	descricao	total_membros
Travel Explorers	Relatos de viagens e dicas de destinos	5
Tech Innovators	Discussões sobre inovações tecnológicas	5
Foodie Paradise	Receitas, restaurantes e gastronomia	5
Startup Hub	Networking e mentoria para startups	5
Book Club	Recomendações e discussões literárias	5
Art & Design	Arte digital, design gráfico e ilustração	5
Fitness & Health	Dicas de exercícios, dieta e bem-estar	5
Gaming Legends	Comunidade para amantes de jogos clássicos e modernos	5
E-Commerce Masters	Dicas e estratégias para negócios online	5

Music Lovers	Compartilhamento de músicas e playlists	5

8. Listar tópicos criados entre setembro e dezembro do ano de 2023.

```
SELECT t.comunidade, t.membro AS autor, t.titulo, t.descricao,
t.data
FROM topico t
WHERE t.data BETWEEN '2023-09-01' AND '2023-12-01';
```

Resultado:

comunidade	autor	titulo	descricao	data
Tech Innovators	TechSolutions	O Futuro da IA em 2024	Como a inteligência artificial está transformando indústrias em 2024?	2023-11-01
Tech Innovators	DataDrivenCo	Blockchain para Segurança Digital	Aplicações práticas de blockchain em proteção de dados.	2023-11-05
Gaming Legends	ShadowHunter	Melhores Jogos Retro de 2023	Lista dos jogos retrô mais populares deste ano.	2023-10-15
Gaming Legends	CyberNinja	eSports: Cenário Competitivo Atual	Discussão sobre torneios e equipes dominantes.	2023-10-20
Music Lovers	DragonSlayer	Playlists para Estudo/Concentração	Indicações de músicas instrumentais e lo-fi.	2023-09-10
Music Lovers	VortexMage	Novos Lançamentos de Rock Alternativo	Bandas emergentes e álbuns do gênero.	2023-09-12

9. Quantas requisições de amizade tem cada status.

```
SELECT r.status, COUNT(*) AS quantidade
FROM req_amizade r
GROUP BY r.status;
```

Resultado:

status	quantidade
Aceita	10
Recusada	5
Pendente	5

10. Quais grupos de amigos podem ver quais páginas.

```
SELECT g.nome AS nome_grupo, p.nome AS nome_pagina, p.criador AS
criador_pagina
FROM grupo_amigos g
    INNER JOIN pode_ver pv
        ON g.codigo = pv.codigo_grupo
    INNER JOIN pagina p
        ON pv.nome = p.nome
        AND pv.criador = p.criador
ORDER BY g.codigo, p.nome;
```

Resultado:

nome_grupo	nome_pagina	criador_pagina
Clã dos Caçadores	Caçadores de Sombras	ShadowHunter
Aliança Tech	Caçadores de Sombras	ShadowHunter
Guilda das Feras	Domadores de Dragões	DragonSlayer
Equipe Pixel	Domadores de Dragões	DragonSlayer
Mestres do Pixel	Guerreiros do Pixel	PixelWarrior
Cyber Squad	Guerreiros do Pixel	PixelWarrior
Masters of Steam	Reino do Steam	SteamLord
Indie Lovers	Reino do Steam	SteamLord

11. Páginas com as maiores taxas de conversão, rejeição e cliques.

```
SELECT nome, criador, taxa_conversao AS valor, 'taxa_conversao' AS
tipo_metrica
FROM propaganda
WHERE taxa_conversao = (
    SELECT MAX(taxa_conversao) FROM propaganda
)
UNION
SELECT nome, criador, taxa_rejeicao AS valor, 'taxa_rejeicao' AS
tipo_metrica
FROM propaganda
WHERE taxa_rejeicao = (
    SELECT MAX(taxa_rejeicao) FROM propaganda
)
UNION
SELECT nome, criador, taxa_cliques AS valor, 'taxa_cliques' AS
tipo_metrica
```

```
FROM propaganda
WHERE taxa_cliques = (
    SELECT MAX(taxa_cliques) FROM propaganda
);
```

Resultados:

nome	criador	valor	tipo_metrica
Campanhas SocialBee	SocialBee	0.2	taxa_rejeicao
Campanhas SocialBee	SocialBee	0.3	taxa_cliques
Promo Inova Corp	InovaCorp	0.28	taxa_conversao

ATUALIZAÇÕES

Vamos tomar dois usuários pessoais para os testes de atualização no banco de dados sendo eles: DragonSlayer e ShadowHunter.

1. Atualização do status de requisição de amizade: Nesse teste vamos ver se o chat foi excluído, de acordo com a implementação do gatilho e se ele é recriado após as duas atualizações:

```
SET search_path TO facepage;

SELECT * FROM chat WHERE (recebe_username = 'DragonSlayer' AND
envia_username = 'ShadowHunter') OR (envia_username = 'DragonSlayer' AND
recebe_username = 'ShadowHunter');

UPDATE req_amizade SET status = 'Recusada' WHERE (recebe_username =
'DragonSlayer' AND envia_username = 'ShadowHunter') OR (envia_username =
'DragonSlayer' AND recebe_username = 'ShadowHunter');

SELECT * FROM chat WHERE (recebe_username = 'DragonSlayer' AND
envia_username = 'ShadowHunter') OR (envia_username = 'DragonSlayer' AND
recebe_username = 'ShadowHunter');

UPDATE req_amizade SET status = 'Aceita' WHERE  (recebe_username =
'DragonSlayer' AND envia_username = 'ShadowHunter') OR (envia_username =
'DragonSlayer' AND recebe_username = 'ShadowHunter');

SELECT * FROM chat WHERE (recebe_username = 'DragonSlayer' AND
envia_username = 'ShadowHunter') OR (envia_username = 'DragonSlayer' AND
recebe_username = 'ShadowHunter');
```

Antes de atualizar:

nome	recebe_username	envia_username
Chat DragonSlayer & ShadowHunter	ShadowHunter	DragonSlayer

Depois:

nome	recebe_username	envia_username

Voltando ao normal:

nome	recebe_username	envia_username
Chat DragonSlayer & ShadowHunter	ShadowHunter	DragonSlayer

2. Atualizar o nome do grupo de amigos “Guilda das Feras” para “Caçadores de Dragões” da DragonSlayer:

```
SELECT * FROM grupo_amigos WHERE admin = 'DragonSlayer';

UPDATE grupo_amigos SET nome = 'Caçadores de Dragões' WHERE nome = 'Guilda
das Feras' and admin = 'DragonSlayer';

SELECT * FROM grupo_amigos WHERE admin = 'DragonSlayer';
```

Antes:

codigo	nome	descricao	admin
4	Equipe Pixel	Fãs de jogos retro	DragonSlayer
3	Guilda das Feras	RPG e estratégia	DragonSlayer

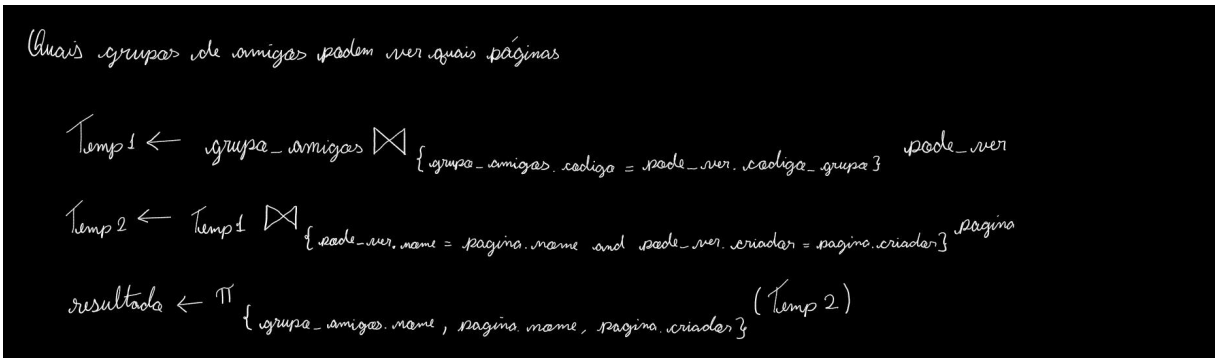
Depois:

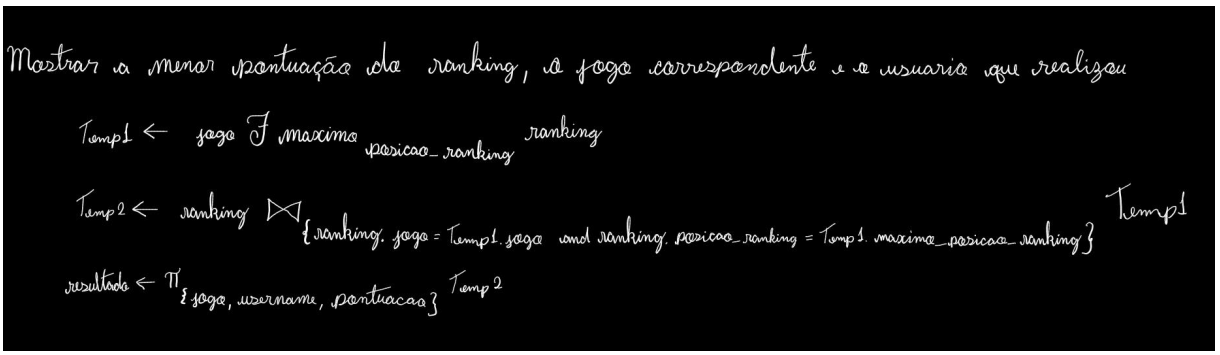
codigo	nome	descricao	admin
4	Equipe Pixel	Fãs de jogos retro	DragonSlayer
3	Caçadores de	RPG e	DragonSlayer

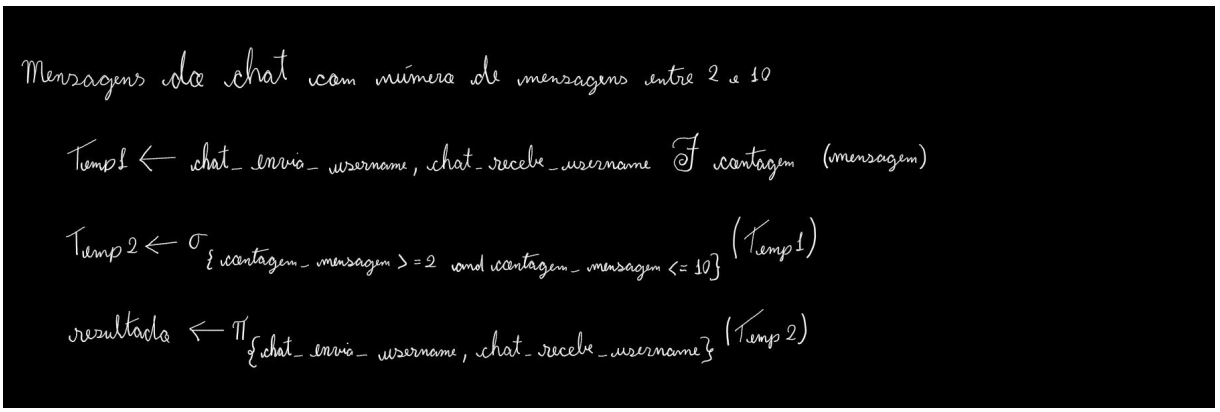
	Dragões	estratégia	

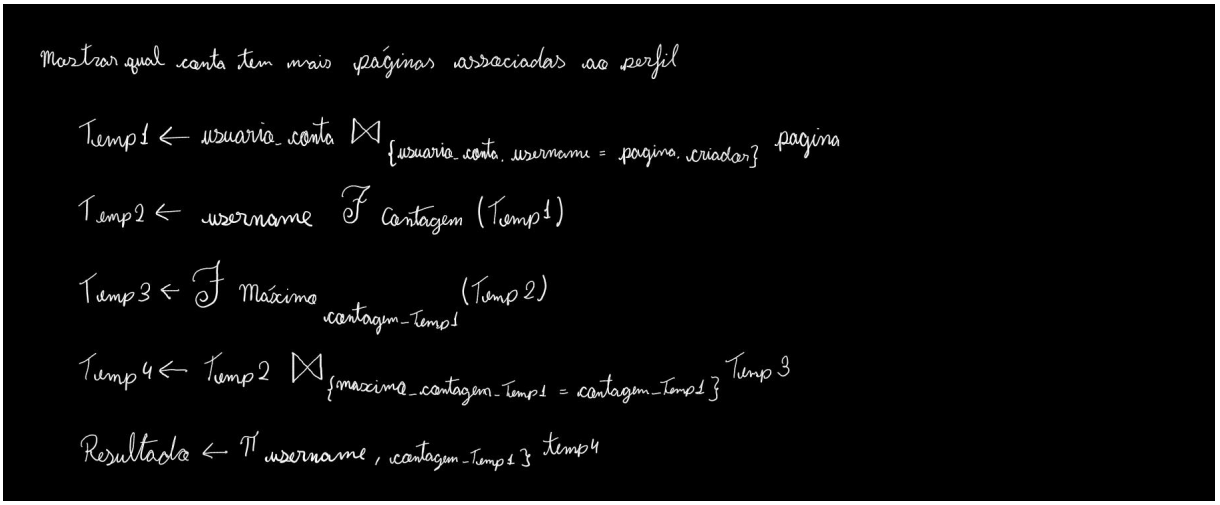
4.2.3. Consultas em Álgebra relacional

Essa sub sub-seção contém a versão conceitual de algumas consultas definidas anteriormente, agora expressas em álgebra relacional. Permite visualizar as operações lógicas realizadas sobre as relações (tabelas), como seleção, projeção, produto cartesiano, junção e renomeação.









4.2.4. Gatilho e Procedimento Armazenado

Nesta parte, são apresentados exemplos de gatilhos e procedimentos armazenados que automatizam ações no banco de dados. Os gatilhos são utilizados para executar ações automaticamente em resposta a eventos como inserções e atualizações. Já os procedimentos armazenados encapsulam blocos de lógica SQL reutilizável.]

PROCEDIMENTOS ARMAZENADOS

1. Cálculo do número de amigos corretamente (no caso calcula de todas as pessoas de uma vez só).

```
CREATE OR REPLACE FUNCTION calcula_nro_amigos()
RETURNS VOID AS $$
BEGIN
    UPDATE pessoal AS P
    SET nro_amg = (SELECT COUNT(*) FROM req_amizade R
        WHERE (R.recebe_username = P.username OR
            R.envia_username = P.username) AND
            R.status = 'Aceita');

END $$ LANGUAGE 'plpgsql';

SELECT calcula_nro_amigos();
```

A função seleciona faz a contagem do número de amigos com o select aninhado e insere o resultado em nro_amg para cada pessoa na tabela pessoal.

2. Cálculo do número de membros de cada comunidade.

```
SET search_path TO facepage;

CREATE OR REPLACE FUNCTION calcula_nro_membros_comunidade()
RETURNS VOID AS $$
BEGIN
    UPDATE comunidade AS C
    SET nro_membros = (SELECT COUNT(*) FROM membros_comunidade M WHERE
M.comunidade = C.nome_comunidade);

END $$ LANGUAGE 'plpgsql';
```

A ideia é semelhante à apresentada para o cálculo do número de amigos, contamos as ocorrências do nome da comunidade em membros da comunidade e atualizamos em cada uma das comunidades.

3. Função para encontrar a média da pontuação em um determinado jogo.

```
CREATE OR REPLACE FUNCTION media_pontuacao_jogo(nome_do_jogo VARCHAR(30))
RETURNS DECIMAL(10, 2) AS $$
DECLARE
    media DECIMAL(10, 2);
BEGIN
    SELECT AVG(pontuacao) INTO media
    FROM joga
    WHERE nome_jogo = nome_do_jogo;

    RETURN media;
END;
$$ LANGUAGE plpgsql;
```

A função executa uma consulta simples na coluna de pontuação da tabela joga para calcular a média de cada jogo. Caso desejemos as médias de todos os jogos, basta remover o parâmetro de entrada e adicionar um ROLLUP com o nome de cada jogo.

4. Função para compartilhar uma determinada pontuação de um jogador (pessoal) de um jogo específico em um determinado chat de amigo ao qual ele participa (mensagem).

```
CREATE OR REPLACE FUNCTION mostrar_pontuacao(IN username_jogador
pessoal.username%TYPE, IN username_recebe pessoal.username%TYPE, IN
nome_jogo jogo.nome%TYPE)
```

```
RETURNS VOID AS $$
DECLARE
    mensagem TEXT;
    existe_jogo boolean;
    pontos joga.pontuacao%TYPE;
BEGIN
    -- Verifica se o jogo existe
    SELECT EXISTS (
        SELECT 1 FROM jogo
        WHERE nome = nome_jogo
    ) INTO existe_jogo;

    IF NOT existe_jogo THEN
        RAISE EXCEPTION 'Nenhum jogo encontrado com o nome %, revisar os
parametros de entrada!',
            nome_jogo;
    END IF;


    -- Verifica se a pontuação existe a caso exista coloca
    SELECT ranking.pontuacao INTO pontos FROM ranking
    WHERE username = ranking.username AND
        ranking.jogo = nome_jogo
    LIMIT 1;

    IF pontos IS NULL THEN
        pontos := 0;
    END IF;

    mensagem := FORMAT('Olha %s, minha incrivel pontuação de %s no jogo %s
!! Incrível não?', username_recebe, pontos, nome_jogo);

    INSERT INTO facepage.mensagem (chat_recebe_username,
chat_envia_username, conteudo) VALUES (username_recebe, username_jogador,
mensagem);
END;
$$ LANGUAGE plpgsql;
```

A função opera com base em um dos gatilhos armazenados. Inicialmente, ela verifica se o jogo informado existe no banco de dados. Caso não exista, uma mensagem de aviso (warning) é emitida informando a inexistência do jogo.

Em seguida, a função realiza outras verificações: se a pontuação do usuário for NULL, ela é automaticamente considerada como zero. Com base nessas informações, o script constroi uma mensagem personalizada, que é então enviada ao amigo designado. O envio da mensagem é

efetivado por meio do gatilho armazenado trg_preencher_mensagem, que é responsável por preencher automaticamente os dados da mensagem e garantir que a operação de envio ocorra de forma consistente e integrada ao restante do sistema.

GATILHOS

1. Gatilho para criação e destruição de chats a medida que as requisições de amizades forem mudando

```
-- Trigger para quando a requisição de amizade mudar, fazer determinadas
ações como deletar o chat (caso a req seja pendente), criar um
chat(solicitação seja aceita e aumentar o número de amigos)

CREATE OR REPLACE FUNCTION trigger_chats()
RETURNS TRIGGER AS $$
DECLARE
    user1 TEXT := NEW.envia_username;
    user2 TEXT := NEW.recebe_username;
BEGIN
    -- Para INSERT ou UPDATE quando status muda para 'Aceita'
    IF (TG_OP = 'INSERT' AND NEW.status = 'Aceita') OR
       (TG_OP = 'UPDATE' AND NEW.status = 'Aceita' AND OLD.status !=
'Aceita') THEN
        IF NOT EXISTS (
            SELECT 1 FROM chat
            WHERE (envia_username = user1 AND recebe_username = user2)
               OR (envia_username = user2 AND recebe_username = user1)
        ) THEN
            INSERT INTO chat (nome, envia_username, recebe_username)
            VALUES (CONCAT('Chat ', user1, ' & ', user2), user1, user2);

            UPDATE pessoal SET nro_amg = nro_amg + 1
            WHERE username IN (user1, user2);

            RAISE NOTICE 'Trigger req_amizade_chat ativado para criação de
chat';
        END IF;
    END IF;

    -- Para UPDATE quando status deixa de ser 'Aceita'
    IF TG_OP = 'UPDATE' AND NEW.status != 'Aceita' AND OLD.status = 'Aceita'
THEN
        DELETE FROM chat
        WHERE (envia_username = user1 AND recebe_username = user2)
           OR (envia_username = user2 AND recebe_username = user1);
```

```
        UPDATE pessoal SET nro_amg = nro_amg - 1
        WHERE username IN (user1, user2);
        RAISE NOTICE 'Trigger req_amizade_chat ativado para deleção de
chat';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER req_amizade_chat
BEFORE INSERT OR UPDATE ON req_amizade
FOR EACH ROW
EXECUTE FUNCTION trigger_chats();
```

O gatilho possui duas funções principais: gerenciar os registros da tabela chat (e, por consequência, da tabela mensagem), de acordo com o status das solicitações de amizade e controlar dinamicamente o número de amigos de cada usuário, refletindo qualquer alteração nas relações de amizade.

No primeiro caso, o gatilho identifica se a operação realizada foi um INSERT ou um UPDATE. Se for um INSERT e a solicitação de amizade (req_amizade) tiver sido aceita, o gatilho verifica se já existe uma tupla correspondente na tabela chat. Caso não exista, ele cria o novo chat entre os usuários envolvidos e incrementa o número de amigos de ambos.

No segundo caso, o gatilho trata especificamente atualizações em que o status da amizade é alterado de "Aceita" para outro valor (como "Recusada" ou "Removida"). Quando isso ocorre, ele remove o chat existente entre os usuários e decrementa o número de amigos em uma unidade, refletindo corretamente o fim da amizade.

2 - Criamos um trigger para facilitar as inserções das mensagems em chats da forma (A, B, mensagem), onde B envia mensagem para A.

```
-- Trigger para inserção nos chats de forma correta, no caso é a inserção de
mensagens nos chats correspondentes da forma em que (A, B, mensagem) onde A
recebe a mensagem, B envia a mensagem e mensagem é o conteúdo da mensagem.
-- O trigger age para que inserções indevidas não aconteçam e que
independente da ordem A, B ou B, A exista um chat correspondente (trigger de
```

```
cima faz essa garantia olhando as requisições de amizade)

CREATE OR REPLACE FUNCTION inserir_chat_mensagem()
RETURNS TRIGGER AS $$
DECLARE
    chat_na_direcao boolean;
    chat_na_direcao_oposta boolean;
    recebe TEXT := NEW.chat_recebe_username;
    envia TEXT := NEW.chat_envia_username;
    temp_username varchar;
BEGIN

    -- Verifica se existe chat na direção A → B (recebe: A, envia: B)
    SELECT EXISTS (
        SELECT 1 FROM chat
        WHERE recebe_username = recebe
        AND envia_username = envia
    ) INTO chat_na_direcao;

    -- Verifica se existe chat na direção oposta B → A
    SELECT EXISTS (
        SELECT 1 FROM chat
        WHERE recebe_username = envia
        AND envia_username = recebe
    ) INTO chat_na_direcao_oposta;

    -- Se não existe chat em nenhuma direção
    IF NOT chat_na_direcao AND NOT chat_na_direcao_oposta THEN
        RAISE EXCEPTION 'Nenhum chat autorizado encontrado entre % e %. A
mensagem não foi enviada.',
            NEW.chat_envia_username, NEW.chat_recebe_username;
        RETURN NULL;
    END IF;

    RAISE NOTICE '------------------------------';
    NEW.remetente := envia;
    RAISE NOTICE 'Criando mensagem entre % e %', envia, recebe;
    -- Padroniza a direção para (A, B) onde B envia para A
    -- Se o chat existente estiver na direção oposta (B → A), invertemos
    IF chat_na_direcao_oposta AND (NOT chat_na_direcao) THEN
        -- Inverte a direção da mensagem para manter o padrão (A, B)
        RAISE NOTICE 'TROCA % e %', envia, recebe;
        temp_username := recebe;
        recebe := envia;
        envia := temp_username;
        RAISE NOTICE 'TROCA FEITA NOVOS ENVIA E RECEBE: % e %', envia,
recebe;
    END IF;
```

```
    RAISE NOTICE 'Criando mensagem entre % e %', envia, recebe;

    -- Garante que o remetente seja sempre quem está enviando (B)

    NEW.chat_recebe_username = recebe;
    NEW.chat_envia_username = envia;

    RAISE NOTICE 'REMETENTE %', NEW.remetente;
    RAISE NOTICE '------------------------------';
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_preencher_mensagem
BEFORE INSERT ON mensagem
FOR EACH ROW
EXECUTE FUNCTION inserir_chat_mensagem();
```

Tínhamos um desafio na inserção de chats, especialmente em identificar quem realmente enviou a mensagem e como armazená-la corretamente, uma vez que os chats eram bidirecionais. Ou seja, independentemente de olhar a conversa como A->B ou B->A, ela deveria ser considerada a mesma, o que gerava um risco de duplicidade. Para resolver isso, optamos por registrar apenas uma direção (por exemplo, B->A ou A->B) na tabela chat, o que levava a um problema no armazenamento das mensagens. Se, por exemplo, tivermos uma conversa entre A e B, a direção de envio (A->B ou B->A) poderia gerar confusão, já que A e B são tanto remetente quanto destinatário, dependendo de quem está enviando ou recebendo.

Dado esse problema, a inserção manual de mensagens se tornava complexa e difícil, pois seria necessário realizar várias verificações para identificar quais mensagens estavam sendo enviadas a quem. Para simplificar, implementamos um gatilho que automatiza o processo de inserção das mensagens. Esse gatilho garante que todas as mensagens sejam registradas conforme a direção original do chat (ex.: A recebe → B envia), independentemente de quem iniciou a conversa.

O sistema verifica a existência de um chat válido entre os usuários e, se necessário, ajusta automaticamente a direção do envio para garantir a consistência no padrão (sem duplicações) e preserva a identidade do remetente verdadeiro em um campo específico. Isso assegura um histórico de mensagens único e organizado, eliminando redundâncias e mantendo a clareza sobre quem enviou cada mensagem.

4.3. Bibliografia Consultada

ELMASRI, Ramez; NAVATHE, Shamkant B. *Sistemas de banco de dados*. Tradução de Daniel Vieira. 6. ed. São Paulo: Pearson Addison Wesley, 2011. Capítulo 4, p. 76: “SQL Básica”. Capítulo 5 p. 95: ”Mais SQL: Consultas complexas, triggers, views e modificação de esquema”.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. *PostgreSQL 17.4 Documentation*. Disponível em: https://www.postgresql.org/docs/17/. Acesso em: 16/04/2025.