

Aluno: Samuel Batista Rennó

Matrícula: UC231029862

Turno: Noturno / Quarta-Feira

Professor: João Robson

Instituição: Universidade Católica de Brasília

Disciplina: Laboratório de Banco de Dados

Projeto de Banco de Dados: Rede social Atleta.



Brasília DF 27/11/24

1- Introdução

Durante a realização deste trabalho fomos desafiados a pôr em prática os conhecimentos adquiridos em sala na realização de um **sistema de banco de dados**.

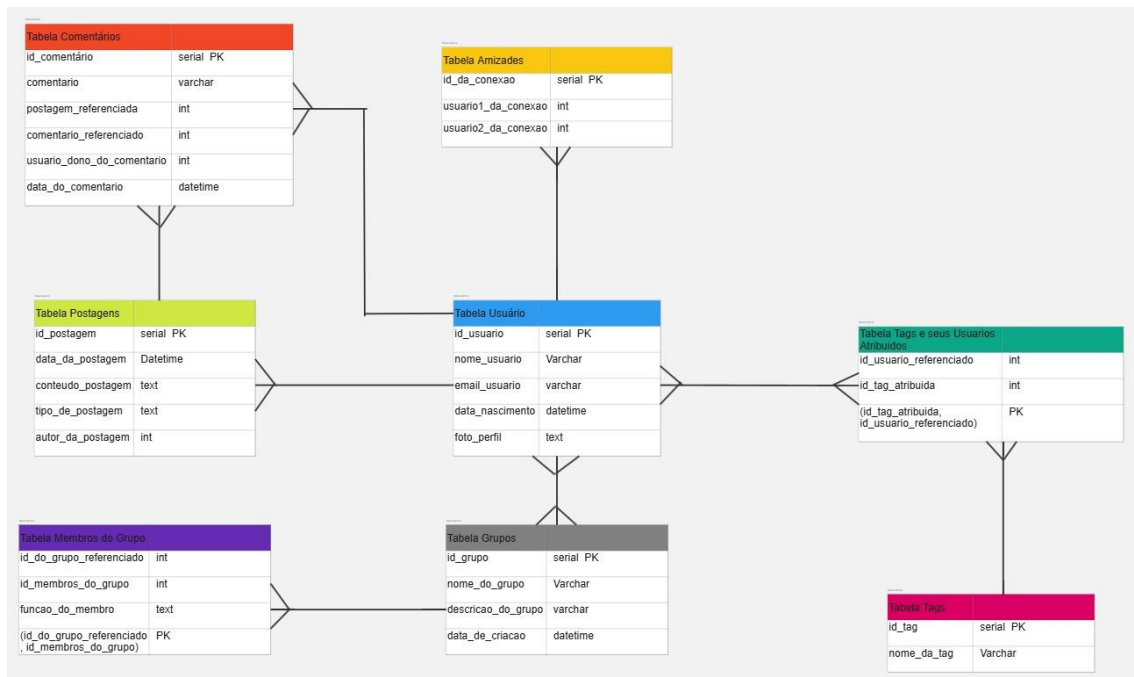
O sistema em questão deveria funcionar como uma espécie de rede social, com o tema de nossa preferência, possuindo funções relativas aos procedimentos básicos de uma rede social. Postagens, usuários, relações de amizade e comentários, por exemplo, deveriam ser integrados ao banco de dados.

Para o meu projeto em particular escolhi desenvolver um sistema de banco de dados de uma rede social voltada para atletas. Na rede, teoricamente, os Usuários poderiam ser cadastrados com suas informações básicas e interagir entre si através de postagens e comentários, além da criação e participação de grupos voltados para gostos e indivíduos em comum. No que tange as postagens, elas poderiam ser de categoria visual (fotos e memes) ou textual, uma que tive inspiração no modelo de publicação do Twitter.

A seguir você poderá encontrar as demais sessões do documento que procuram explicar mais detalhadamente o código e sua construção. Agradeço pela atenção e boa leitura.

2- Diagrama

A seguir veja o diagrama lógico entre as diferentes tabelas do sistema:



A **Tabela Usuários** em azul é o ponto central do diagrama, ela se conecta, direta ou indiretamente, com todas as demais tabelas que compõem o banco de dados, nisto fica representado o usuário como principal indivíduo de interações no sistema.

A relação **Usuário/Amizade** é de *1 para N*, isto porque um usuário pode possuir várias amizades, no entanto, uma amizade pertence a um indivíduo.

A relação **Usuário/Postagem** e **Usuário/Comentário** apresentam o mesmo modelo, sendo que um usuário é capaz de possuir mais de uma postagem ou comentário, contudo cada uma dessas entidades pode estar vinculada a apenas um usuário.

Em seguida são as últimas relações que a tabela Usuário possui, e estas são: **Usuário/Grupo** e **Usuário/tag**. Estas apresentam cardinalidade diferente das anteriores, vários usuários podem possuir a mesma tag, e várias tags podem estar relacionadas a um mesmo usuário, o mesmo acontece com a tabela grupos. Sendo assim, a cardinalidade nestas duas relações é definida como *N para M*.

Vamos nos atentar agora para as outras relações.

Postagem/Comentários é uma relação de *1 para N*. Por causa do fato de uma postagem poder receber vários comentários, no entanto um comentário só pode estar atribuído a uma postagem ou comentário. Dito isso, percebe-se uma relação implícita "Comentário/Comentário", pois um comentário pode se relacionar a outro, mas apenas um por vez, no caso *1 para 1*.

A mesma regra que se aplica a relação **Postagem/Comentários** se aplica as relações **Grupo/Membros do Grupo** e **Tag/Tags e seus usuários relacionados**.

3- Tabelas

Vamos agora analisar as tabelas e as suas variáveis:

| Tabela Usuário | |
|-----------------|-----------|
| id_usuario | serial PK |
| nome_usuario | Varchar |
| email_usuario | varchar |
| data_nascimento | datetime |
| foto_perfil | text |

A tabela usuário é o ponto chave central, do qual todas as outras interações derivam. A sua *Primary Key* é o *id_usuario*, usei essa variável como ponto de referência na hora de selecionar e registrar indivíduos no banco de dados. Ela é de caráter **inteiro** e **auto incrementável**, o que agiliza o processo de inserção de dados e organiza ao mesmo tempo.

As demais variáveis são do tipo *varchar* por se tratarem quase que totalmente de caracteres de texto, com exceção a variável foto perfil, esta variável é do tipo texto permitindo que ela armazene uma *URL*, isso fará com que o armazenamento das imagens de perfil seja atribuído via endereço de imagem.

O resto das tabelas seguem princípios semelhantes, todas possuem *ids* identificadores inteiros e auto incrementáveis.

Nomes e descrições são variáveis do tipo *varchar*, e para o registro de datas eu apliquei o *datetime*, o que fará com que ao inserir dados de tempo usando a função *now()*, seja registrado data e hora, tornando mais verídico e preciso.

Tem também uma última nota importante que é preciso comentar, e por isso vou usar como exemplo a tabela **Tags e seus usuários relacionados**.

Nova tabela

| Tabela Tags e seus Usuarios Atribuidos | |
|---|-----|
| id_usuario_referenciado | int |
| id_tag_atribuida | int |
| (id_tag_atribuida, id_usuario_referenciado) | PK |

Note nas duas variáveis: *id_usuario_referenciado* e *id_tag_atribuida*. Estas duas variáveis representam uma outra gama de variáveis presentes em outras tabelas do programa, como *id_do_grupo_referenciado*, elas são variáveis que criam ligações com variáveis de outras tabelas por meio de **references**.

Em outras palavras, estou usando-as como chaves-estrangeiras para conectar dados de diferentes tabelas. Veja o exemplo a seguir:

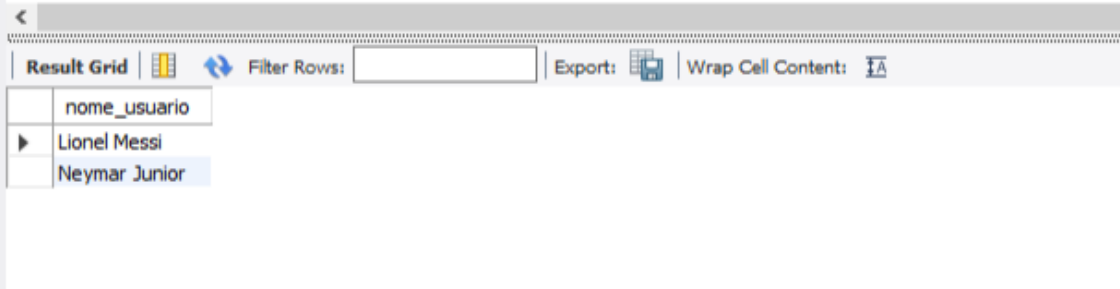
```
50
51 create table tags_e_seus_usuarios_atribuidos(
52     id_usuario_referenciado int references usuario(id_usuario),
53     id_tag_atribuida int references tag(id_tag),
54     primary key(id_usuario_referenciado, id_tag_atribuida)
55 );
```

4- Consultas

Por fim, gostaria de apresentar os códigos de consultas dentro do banco de dados.

1-Retorna o nome de todos os usuários que comentaram a postagem 1

```
165
166 • SELECT DISTINCT nome_usuario
167     FROM comentarios
168     JOIN postagens ON postagem_referenciada_id = id_postagem
169     JOIN usuario ON usuario_dono_do_comentario_id = id_usuario
170     WHERE autor_da_postagem = 1;
171
```

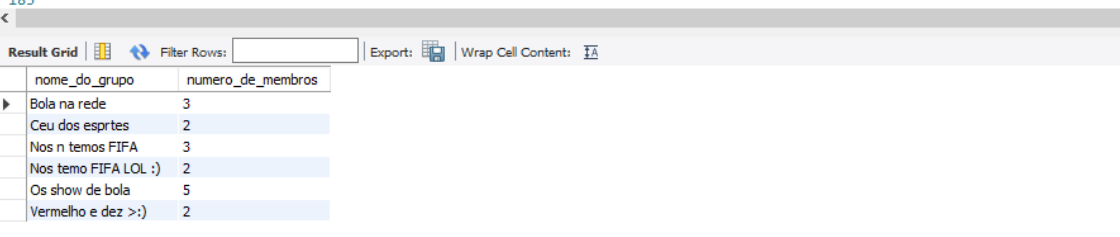


The screenshot shows a database interface with a query editor and a result grid. The query is a SQL SELECT statement that retrieves the names of users who commented on a specific post (postagem 1). The result grid displays two rows of data.

| nome_usuario |
|---------------|
| Lionel Messi |
| Neymar Junior |

2-Mostra o nome de todos os grupos e o número de participantes nele contidos

```
172
173 • SELECT
174     nome_do_grupo,
175     COUNT(id_membros_do_grupo) AS numero_de_membros
176 FROM
177     grupos
178 LEFT JOIN
179     membros_do_grupo ON id_grupo = id_do_grupo_referenciado
180 GROUP BY
181     nome_do_grupo;
182
183
```



The screenshot shows a database interface with a query editor and a result grid. The query is a SQL SELECT statement that retrieves the names of all groups and the number of participants in each group. The result grid displays six rows of data.

| nome_do_grupo | numero_de_membros |
|----------------------|-------------------|
| Bola na rede | 3 |
| Ceu dos esprtes | 2 |
| Nos n temos FIFA | 3 |
| Nos temo FIFA LOL :) | 2 |
| Os show de bola | 5 |
| Vermelho e dez >:) | 2 |

3-Retorna todos os nome e datas de nascimento de usuários que nasceram posteriormente ao ano de 1993

```
185
186 • SELECT
187     nome_usuario,
188     data_nascimento
189 FROM
190     usuario
191 WHERE
192     data_nascimento > '1993-12-31';
```

| Result Grid | | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|-----------------|-----------------|---------|--------------------|
| | nome_usuario | data_nascimento | | |
| ▶ | Kylian Mbappé | 1998-12-20 | | |
| | Nikola Jokic | 1995-02-19 | | |
| | Patrick Mahomes | 1995-09-17 | | |

4-Retorna todos os nomes dos usuários presentes no grupo “Show de Bola” que possuam a tag “eu tenho e vc n”

```
196 • SELECT nome_usuario
197 FROM usuario
198 JOIN membros_do_grupo ON id_usuario = id_membros_do_grupo
199 JOIN grupos ON id_do_grupo_referenciado = id_grupo
200 JOIN tags_e_seus_usuarios_atribuidos ON id_usuario = id_usuario_referenciado
201 JOIN tag ON id_tag_atribuida = id_tag
202 WHERE nome_do_grupo = 'Os show de bola'
203 AND nome_da_tag = 'eu tenho e vc n';
```

| Result Grid | | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|--------------|--------------|---------|--------------------|
| | nome_usuario | | | |
| ▶ | Lionel Messi | | | |
| | Edson Pelé | | | |

5- Retorna o nome de usuário com o maior número de comentários feitos

```
207 • SELECT nome_usuario, COUNT(id_comentario) AS total_comentarios
208 FROM comentarios
209 JOIN usuario ON usuario_dono_do_comentario_id = id_usuario
210 GROUP BY nome_usuario
211 ORDER BY total_comentarios DESC
212 LIMIT 1;
```

| Result Grid | | Filter Rows: | Export: | Wrap Cell Content: |
|-------------|--------------|--------------|---------|--------------------|
| | nome_usuario | | | |
| ▶ | Lionel Messi | | | |
| | Edson Pelé | | | |

5- Conclusão

Agradeço por ler o meu trabalho e segue a baixo o link para o repositório no git hub aonde você poderá visualizar todo o código por completo.

<https://github.com/SamuelBati/Projeto-de-Banco-de-Dados-N2-Rede-Social-Aletras/tree/main>