

Material de apoio

Site: [Geração Tech](#)
Curso: Formação em Desenvolvedor Web - Online
Livro: Material de apoio

Impresso por: JOÃO VITOR DE MELO FREITAS
Data: sexta-feira, 2 ago. 2024, 08:54

Índice

1. Banco de Dados

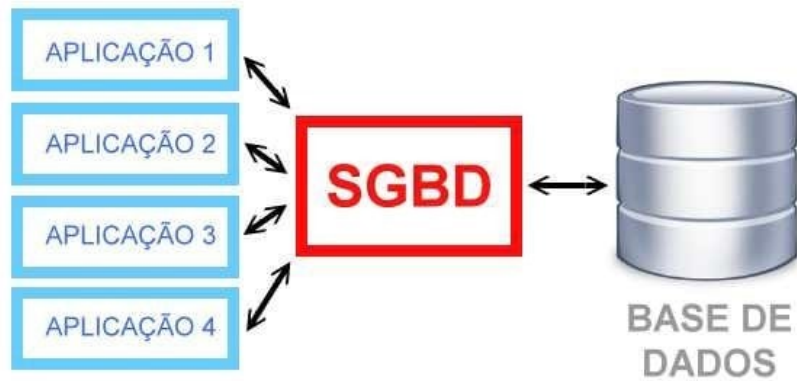
- 1.1. Sistema de Gerenciamento de Bancos de Dados (SGBD)
- 1.2. Instalação do MySQL
- 1.3. análise de Requisitos e Estruturação de Banco de Dados para um Blog
- 1.4. Estruturando o Banco de Dados do Blog no Google Planilhas
- 1.5. Tipos de Dados no Banco de Dados
- 1.6. Criação e Manipulação de Tabelas no MySQL Workbench
- 1.7. Inserindo Vários Registros em uma Tabela MySQL
- 1.8. Manipulação de Dados no MySQL
- 1.9. Comandos SQL: SELECT
- 1.10. SELECT, *, AS, ORDER BY
- 1.11. WHERE e Operadores de Comparação

1. Banco de Dados

Olá pessoal, bem-vindos à primeira aula de Banco de Dados.

Eu me chamo Kelvis Moura e vou acompanhar vocês nessa etapa do curso.

Hoje, vamos entender visualmente como funciona o sistema de gerenciamento de bancos de dados (SGBD) e o que significa trabalhar com banco de dados.



1.1. Sistema de Gerenciamento de Bancos de Dados (SGBD)

O que é um SGBD?

Um SGBD é um sistema que permite criar, gerenciar e manipular bancos de dados.

Existem vários SGBDs disponíveis, tanto gratuitos quanto pagos.

Neste curso, vamos trabalhar com o MySQL, que é um SGBD popular e amplamente utilizado.

Estrutura do SGBD

Dentro de um SGBD, temos os seguintes componentes:

1. **Usuários:** São responsáveis por acessar e manipular os bancos de dados. Cada usuário tem um username e uma senha.
2. **Bancos de Dados:** São as divisões dentro do SGBD que contêm as tabelas e os dados.
3. **Tabelas:** Estruturas dentro dos bancos de dados onde os dados são armazenados em linhas e colunas.

Permissões de Usuários

Cada usuário pode ter diferentes permissões para acessar e manipular os bancos de dados. Por exemplo:

- O Usuário 1 pode ter acesso ao Banco 1 e Banco 2.
- O Usuário 2 pode ter acesso apenas ao Banco 2.
- O Usuário 3 pode ter acesso a todos os bancos de dados.

Linguagem SQL

Para manipular os dados dentro de um SGBD, utilizamos a linguagem SQL (Structured Query Language). A SQL é dividida em três categorias principais de comandos:

1. **DCL (Data Control Language):** Comandos de controle de acesso, como **GRANT** e **REVOKE**, usados para conceder ou revogar permissões dos usuários.
2. **DDL (Data Definition Language):** Comandos para definir a estrutura do banco de dados, como **CREATE**, **ALTER** e **DROP**, usados para criar, modificar ou excluir bancos de dados e tabelas.
3. **DML (Data Manipulation Language):** Comandos para manipulação dos dados, como **SELECT**, **INSERT**, **UPDATE** e **DELETE**, usados para consultar, inserir, atualizar ou excluir dados nas tabelas.

Conexão com o Banco de Dados

Como se Conectar ao SGBD

Para se conectar a um SGBD, você precisa de:

1. **Host:** O endereço do servidor onde o SGBD está hospedado.
2. **Porta:** A porta de comunicação do SGBD.
3. **Credenciais:** Username e senha de um usuário válido no SGBD.

Uma vez conectado, você pode executar comandos SQL para manipular os dados.

Exemplificação da Conexão

1. **Usuário:** O usuário, do lado de fora, utiliza suas credenciais (username e senha) para se conectar ao SGBD.
2. **Solicitação de Conexão:** A conexão é solicitada ao host e à porta do SGBD.
3. **Verificação:** O SGBD verifica se as credenciais fornecidas são válidas.
4. **Autorização:** Se as credenciais forem válidas, o usuário é autorizado a se conectar.
5. **Execução de Comandos:** Após a conexão, o usuário pode executar comandos SQL apenas nos bancos de dados aos quais ele tem permissão de acesso.

dia 35 video 01



Conclusão

Nesta aula, tivemos uma visão geral sobre:

- O que é um SGBD.
- Componentes de um SGBD (usuários, bancos de dados e tabelas).
- Categorias de comandos SQL (DCL, DDL, DML).
- Como se conectar a um SGBD.

Espero que esta aula tenha esclarecido o funcionamento básico de um sistema de gerenciamento de bancos de dados.

Na próxima aula, vamos explorar mais detalhadamente os comandos SQL e como utilizá-los na prática.

Ficamos por aqui. Até a próxima!

1.2. Instalação do MySQL

Hoje, vamos aprender a baixar e instalar o MySQL em nossa máquina local.

Download e Instalação do MySQL

Passo a Passo da Instalação

1. Acessar o site do MySQL:

- Abra o navegador Chrome.
- Pesquise por "MySQL" no Google.
- Clique no primeiro link que aparecer.

2. Baixar o MySQL:

- Na página do MySQL, clique em "Downloads".
- Role a página e clique em "GPL Download".
- Na página de opções de download, escolha "MySQL Installer for Windows".
- Clique no link de download e selecione a opção "No thanks, just start my download".

3. Instalar o MySQL:

- Após o download, abra o instalador.
- Permita que o instalador faça alterações no dispositivo, se solicitado.
- O instalador do MySQL será aberto.

Escolhendo Componentes para Instalação

1. Selecionar os Componentes:

- Escolha "Custom" para selecionar manualmente os componentes que deseja instalar.
- Selecione "MySQL Server" e "MySQL Workbench".
- Clique em "Next" para prosseguir.

2. Baixar e Instalar Componentes:

- O instalador mostrará os programas que serão baixados e instalados.
- Clique em "Execute" para iniciar o download e a instalação.

Configuração do MySQL Server

1. Configurar o MySQL Server:

- Após a instalação, clique em "Next" para configurar o servidor.
- Mantenha a configuração padrão de conexão TCP/IP na porta 3306.
- Clique em "Next".

2. Definir Senha do Usuário Root:

- Defina uma senha para o usuário root. Para fins deste curso, vamos usar "root".
- Confirme a senha e clique em "Next".

3. Finalizar Configuração:

- Revise as configurações e clique em "Execute" para finalizar a configuração.
- Após a conclusão, clique em "Finish".

Configuração do MySQL Workbench

1. Abrir o MySQL Workbench:

- Após a instalação, o MySQL Workbench será aberto automaticamente.
- Você verá uma conexão de banco de dados padrão chamada "Local Instance MySQL".

2. Testar Conexão:

- Clique com o botão direito na conexão e selecione "Edit Connection".
- Verifique que o Hostname é "localhost", a porta é "3306" e o usuário é "root".
- Clique no botão de senha e insira "root".
- Clique em "Test Connection" para testar a conexão.
- Se a conexão for bem-sucedida, clique em "OK" e feche a janela de edição de conexão.

dia 35 video 02



Conclusão

Parabéns! Agora você tem o MySQL Server e o MySQL Workbench instalados e configurados em sua máquina local.

Na próxima aula, vamos explorar mais sobre como utilizar o MySQL Workbench para gerenciar nossos bancos de dados.

Até a próxima aula!

1.3. análise de Requisitos e Estruturação de Banco de Dados para um Blog

Introdução

Olá pessoal! Nas próximas aulas, vamos trabalhar em um cenário onde um cliente nos contratou para construir e implementar o banco de dados de um blog.

Antes de escrever qualquer SQL, precisamos entender as regras de negócio do que precisamos implementar e entender de maneira conceitual como vamos criar este banco de dados, estruturando e organizando as informações que precisam ser salvas.

Entendendo a Estrutura de um Blog

Exemplo Prático

Para começar, vamos usar o site [Medium.com](https://medium.com) como referência, pois é uma plataforma de blog onde várias pessoas podem publicar posts, comentar e compartilhar informações.

Análise da Página de Listagem de Posts

- **Imagem:** Cada post possui uma imagem destacada.
- **Usuário (Autor):** O nome do usuário que escreveu o post aparece abaixo da imagem, junto com uma foto de perfil.
- **Título:** O título do post.
- **Descrição:** Uma breve descrição do post, mostrando um resumo do conteúdo.
- **Data de Lançamento:** A data em que o post foi publicado.

Análise de um Post Específico

Ao clicar em um post específico, temos:

- **Título:** O título do post em destaque.
- **Autor do Post:** Nome do autor e a foto de perfil.
- **Data de Publicação:** Data em que o post foi publicado.
- **Conteúdo:** O conteúdo completo que o autor escreveu.
- **Biografia do Autor:** Informações sobre o autor, incluindo nome, quantidade de seguidores e uma breve biografia.
- **Comentários:** Os comentários de outros usuários, mostrando nome, foto de perfil, texto do comentário e data de criação.
- **Respostas aos Comentários:** Outros usuários podem responder aos comentários, seguindo o mesmo padrão de nome, foto e data.

Requisitos Identificados

Com base na análise acima, podemos identificar os seguintes requisitos para o banco de dados do blog:

1. Posts:

- Imagem destacada
- Título
- Descrição
- Conteúdo
- Data de publicação
- Autor

2. Usuários (Autores):

- Nome
- Foto de perfil
- Biografia
- Quantidade de seguidores

3. Comentários:

- Nome do usuário que comentou
- Foto de perfil
- Texto do comentário
- Data de criação
- Referência ao post

4. Respostas aos Comentários:

- Nome do usuário que respondeu
- Foto de perfil
- Texto da resposta
- Data de criação
- Referência ao comentário original

Estrutura do Banco de Dados

Com base nos requisitos, podemos começar a estruturar nosso banco de dados:

Tabelas Principais

1. Tabela de Usuários (Authors):

- `id` (PK)
- `name`
- `profile_picture`
- `bio`
- `followers_count`

2. Tabela de Posts (Posts):

- `id` (PK)
- `title`
- `description`
- `content`
- `publication_date`
- `author_id` (FK para `Authors.id`)
- `image`

3. Tabela de Comentários (Comments):

- `id` (PK)
- `post_id` (FK para `Posts.id`)
- `author_id` (FK para `Authors.id`)
- `content`
- `creation_date`

4. Tabela de Respostas (Replies):

- `id` (PK)
- `comment_id` (FK para `Comments.id`)
- `author_id` (FK para `Authors.id`)
- `content`
- `creation_date`

dia 35 video 03



Conclusão

Entender a estrutura e as necessidades do blog é essencial para construir um banco de dados eficiente e funcional.

Na próxima aula, vamos traduzir essa estrutura conceitual para um modelo físico, criando as tabelas e as relações no banco de dados MySQL. Fiquem atentos!

Até a próxima aula!

1.4. Estruturando o Banco de Dados do Blog no Google Planilhas

Introdução

Olá pessoal! Na aula anterior, entendemos um pouco sobre as regras de negócio que o cliente precisa para o banco de dados do blog. Agora, vamos juntar essas duas informações e entender como estruturar isso dentro do Google Planilhas, funcionando como nosso Sistema de Gerenciamento de Banco de Dados (SGBD).

Utilizando o Google Planilhas como SGBD

O Google Planilhas pode ser utilizado para simular um SGBD, onde podemos criar várias planilhas (semelhantes a bancos de dados) e dentro dessas planilhas, várias tabelas. Isso nos permitirá organizar e visualizar as informações que serão salvas no nosso blog.

Estruturando a Planilha "Blog"

1. Criando a Planilha de Blog:

- Abra o Google Chrome e acesse o Google Planilhas.
- Crie uma nova planilha e nomeie-a como "Blog".

2. Criando Tabelas na Planilha:

- Dentro da planilha "Blog", podemos criar várias abas (tabelas) para organizar nossas informações. Cada aba pode representar uma tabela diferente do banco de dados.

Estruturando as Colunas da Tabela de Posts

Com base na análise feita anteriormente, identificamos as seguintes informações que precisamos armazenar para cada post:

- Título do Post
- Data de Publicação
- Conteúdo do Post
- Autor do Post (Nome)
- Foto de Perfil do Autor
- Biografia do Autor
- Quantidade de Seguidores do Autor
- Quantidade de Comentários

Implementação no Google Planilhas

1. Criando a Tabela de Posts:

- Na aba "Posts" da planilha "Blog", crie as seguintes colunas:

- Título do Post
- Data de Publicação
- Conteúdo do Post
- Nome do Autor
- Foto de Perfil do Autor
- Biografia do Autor
- Quantidade de Seguidores
- Quantidade de Comentários

2. Exemplo de Preenchimento:

- Preencha as colunas com um exemplo de post:
 - **Título do Post:** Como exemplo, use o título de um post do Medium.
 - **Data de Publicação:** Utilize a data em formato americano (MM/DD/YYYY).
 - **Conteúdo do Post:** Coloque um resumo ou parte do conteúdo do post.
 - **Nome do Autor:** Nome do autor do post.
 - **Foto de Perfil do Autor:** URL da imagem de perfil do autor (e.g., `imagem.com/autor-superfil.png`).
 - **Biografia do Autor:** Breve descrição do autor.
 - **Quantidade de Seguidores:** Número de seguidores do autor (e.g., 50.000).
 - **Quantidade de Comentários:** Número de comentários no post (e.g., 33).

Visualizando as Informações

A estrutura final da sua tabela de posts no Google Planilhas deve se parecer com isso:

Título do Post	Data de Publicação	Conteúdo do Post	Nome do Autor	Foto de Perfil do Autor	Biografia do Autor	Quantidade de Seguidores	Quantidade de Comentários
Exemplo de Título	01/01/2022	Resumo do conteúdo...	Nome do Autor	imagem.com/autor-superfil.png	Descrição do autor	50.000	33

Próximos Passos

Na próxima aula, vamos traduzir essa estrutura em código SQL, criando um banco de dados e suas respectivas tabelas utilizando comandos SQL no MySQL. Isso nos permitirá manipular os dados de maneira mais eficiente e segura, além de ser uma prática comum em projetos de desenvolvimento de software.

dia 35 video 04



Conclusão

Utilizar o Google Planilhas como um modelo para estruturar nossos dados nos ajuda a visualizar como as informações serão organizadas no banco de dados. Com isso, estamos prontos para começar a implementar essas estruturas em um SGBD real, utilizando SQL.

Até a próxima aula!

1.5. Tipos de Dados no Banco de Dados

Introdução

Olá pessoal! Na aula anterior, criamos nossa primeira tabela no banco de dados.

Hoje, vamos entender melhor os principais tipos de dados que podemos definir nas colunas de nossas tabelas.

Vou mostrar alguns exemplos para que possamos ter uma visão mais clara de como utilizar esses tipos de dados corretamente.

Tipos de Dados

Dados de Texto

1. VARCHAR:

- Utilizado para armazenar textos curtos.
- Exemplo: `VARCHAR(255)`
- Isso significa que a coluna pode armazenar até 255 caracteres.
- Qualquer caractere conta (letras, números, espaços, pontuações, etc).

2. TEXT:

- Utilizado para armazenar textos longos.
- Exemplo: `TEXT`
- Suporta até 65.535 caracteres.
- Ideal para grandes blocos de texto, como descrições ou conteúdos de postagens.

Dados Numéricos

1. INT:

- Utilizado para armazenar números inteiros.
- Exemplo: `INT`
- Armazena números sem casas decimais.

2. TINYINT:

- Utilizado para armazenar números inteiros menores.
- Exemplo: `TINYINT`
- Pode ser usado para armazenar valores booleanos (1 ou 0, verdadeiro ou falso).

3. DECIMAL:

- Utilizado para armazenar números decimais com precisão.
- Exemplo: `DECIMAL(5,2)`
- O primeiro número (5) define o total de dígitos, e o segundo número (2) define o número de casas decimais.
- Ideal para valores monetários.

4. FLOAT:

- Utilizado para armazenar números com casas decimais.
- Menos preciso que o `DECIMAL`, mas ocupa menos espaço.

5. DOUBLE:

- Similar ao `FLOAT`, mas com maior precisão e maior consumo de espaço.
- Exemplo: Ideal para coordenadas geográficas (latitude, longitude).

Dados de Data e Hora

1. DATE:

- Utilizado para armazenar datas.
- Formato: `YYYY-MM-DD`
- Exemplo: `2024-01-01`

2. TIME:

- Utilizado para armazenar horas.
- Formato: `HH:MM:SS`
- Exemplo: `12:30:00`

3. DATETIME:

- Utilizado para armazenar data e hora.
- Formato: YYYY-MM-DD HH:MM:SS
- Exemplo: 2024-01-01 12:30:00

4. **TIMESTAMP:**

- Similar ao **DATETIME**, mas também armazena a hora atual automaticamente.
- Contagem de segundos desde 1 de janeiro de 1970 até a data e hora atual.

Exemplos na Planilha

Vamos visualizar alguns exemplos usando o Google Planilhas.

Dados de Texto

- **VARCHAR(255):**

- Exemplo: `post_titulo` com **VARCHAR(255)**
- Armazena até 255 caracteres, como "Como criar um banco de dados".

- **TEXT:**

- Exemplo: `post_conteudo` com **TEXT**
- Armazena grandes blocos de texto, como o conteúdo completo de um post de blog.

Dados Numéricos

- **INT:**

- Exemplo: `quantidade_comentarios` com **INT**
- Armazena o número total de comentários em um post, como 45.

- **DECIMAL(5,2):**

- Exemplo: `preco_produto` com **DECIMAL(5,2)**
- Armazena valores monetários, como 199.99.

Dados de Data e Hora

- **DATE:**

- Exemplo: `post_data` com **DATE**
- Armazena a data de publicação do post, como 2024-01-01.


- **TIMESTAMP:**

- Exemplo: `criado_em` com **TIMESTAMP**
- Armazena a data e hora de criação do registro automaticamente.

Implementação no MySQL Workbench

Vamos voltar ao MySQL Workbench e criar a tabela com os tipos de dados corretos.

sql

 Copiar código

```
CREATE TABLE posts ( post_titulo VARCHAR(255), post_data DATE,
```

```
post_conteudo TEXT,
```

```
post_autor VARCHAR(255),
```

```
post_autor_foto_perfil VARCHAR(255),
```

```
post_autor_bio TEXT,
```

```
quantidade_seguidores INT,
```

```
quantidade_comentarios INT );
```

dia 35 video 05



Resumo

Hoje, entendemos melhor os principais tipos de dados que podemos utilizar nas colunas de nossas tabelas.

Definimos os tipos de dados para texto, números e datas. Na próxima aula, continuaremos a aprofundar nosso entendimento e aplicaremos esses conceitos, na prática.

Até a próxima aula!

1.6. Criação e Manipulação de Tabelas no MySQL Workbench

Na aula anterior, entendemos como definir uma tabela, nomear colunas e atribuir tipos de dados a essas colunas.

Agora, vamos concluir a criação da nossa tabela e inserir dados nela.

Revisão dos Tipos de Dados

Antes de começarmos, vamos revisar brevemente os tipos de dados que definimos:

- **VARCHAR(45)**: Usado para armazenar texto curto, até 45 caracteres.
- **DATE**: Usado para armazenar datas.
- **TEXT**: Usado para armazenar texto longo.
- **INT**: Usado para armazenar números inteiros.


Criação da Tabela

Vamos voltar ao MySQL Workbench e finalizar a criação da nossa tabela "posts".

Definindo a Estrutura da Tabela

No MySQL Workbench, temos a seguinte estrutura para a nossa tabela:

sql

 Copiar código

```
USE blog; CREATE TABLE posts ( post_titulo VARCHAR(45), post_data DATE, post_conteudo TEXT, post_autor VARCHAR(45),  
post_autor_foto_perfil VARCHAR(45), post_autor_bio VARCHAR(255), quantidade_seguidores INT, quantidade_comentarios INT );
```

Vamos executar esse comando:

1. Selecione todo o bloco de código do **CREATE TABLE**.
2. Clique no ícone do raio (⚡) para executar.

Se tudo estiver correto, você verá uma mensagem de sucesso no rodapé do Workbench.

Verificando a Tabela

Para confirmar que a tabela foi criada corretamente:

1. Clique com o botão direito na área em branco da lateral esquerda.
2. Selecione "Refresh".
3. Expanda o banco de dados "blog" e a tabela "posts".

Você verá as colunas que definimos.


Inserindo Dados na Tabela

Vamos aprender a inserir dados na nossa tabela usando o comando **INSERT**.

Estrutura do Comando **INSERT**

A estrutura básica do comando **INSERT** é a seguinte:


sql

 Copiar código

```
INSERT INTO nome_da_tabela (coluna1, coluna2, coluna3, ...) VALUES (valor1, valor2, valor3, ...);
```

Vamos inserir um exemplo de post na nossa tabela "posts":

sql

 Copiar código


```
INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor) VALUES ('Meu Primeiro Post', '2024-01-01', 'Este é o conteúdo do meu primeiro post.', 'João Silva');
```

Execute o comando:

- 1. Selecione o bloco de código `INSERT INTO`.
- 2. Clique no ícone do raio (⚡) para executar.

Se o comando for executado com sucesso, você verá a mensagem indicando que uma linha foi inserida.


Verificando os Dados Inseridos

Para visualizar os dados inseridos:

- 1. Passe o mouse sobre a tabela "posts".
- 2. Clique no terceiro ícone (visualizar dados).

Você verá a linha inserida com os valores que definimos, e as outras colunas terão valores `NULL`.


dia 35 video 06



Exercícios

Vamos inserir mais alguns dados na nossa tabela para praticar:

sql

 Copiar código

```
INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor) VALUES ('Post sobre MySQL', '2024-02-15', 'Aprendendo sobre MySQL no curso de banco de dados.', 'Maria Souza'); INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor) VALUES ('Dicas de JavaScript', '2024-03-10', 'Compartilhando dicas sobre JavaScript.', 'Pedro Lima');
```

Execute esses comandos e verifique os dados inseridos na tabela.

Conclusão

Hoje, concluímos a criação da nossa tabela e aprendemos a inserir dados nela usando o comando `INSERT`.

Nas próximas aulas, exploraremos mais comandos para manipulação e consulta de dados.

Até a próxima aula!


1.7. Inserindo Vários Registros em uma Tabela MySQL

Agora, vamos continuar a inserção de dados na tabela e lidar com tipos de dados específicos.

Revisão da Estrutura da Tabela

Revisando a estrutura da nossa tabela no MySQL Workbench, temos o seguinte:

sql

 Copiar código

```
USE blog; CREATE TABLE posts ( post_titulo VARCHAR(45), post_data DATE, post_conteudo TEXT, post_autor VARCHAR(45), post_autor_foto_perfil VARCHAR(45), post_autor_bio VARCHAR(255), quantidade_seguidores FLOAT, quantidade_comentarios INT );
```

Executando o Comando de Criação da Tabela

Para criar a tabela, selecione todo o bloco de código do **CREATE TABLE** e clique no ícone do raio (⚡). Se tudo estiver correto, você verá uma mensagem de sucesso no rodapé do Workbench.

Verificando a Tabela

Para confirmar que a tabela foi criada corretamente:

1. Clique com o botão direito na área em branco da lateral esquerda.
2. Selecione "Refresh".
3. Expanda o banco de dados "blog" e a tabela "posts".
4. Você verá as colunas que definimos.

Inserindo Dados na Tabela

Vamos continuar inserindo dados na nossa tabela "posts" usando o comando **INSERT**.

Estrutura do Comando **INSERT**

A estrutura básica do comando **INSERT** é a seguinte:

sql

 Copiar código

```
INSERT INTO nome_da_tabela (coluna1, coluna2, coluna3, ...) VALUES (valor1, valor2, valor3, ...);
```

Inserindo Múltiplos Registros

Vamos inserir mais alguns dados na nossa tabela para praticar:

sql

 Copiar código

```
INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio, quantidade_seguidores, quantidade_comentarios) VALUES ('Viagens pelo Mundo', '2024-07-01', 'Explorando diferentes culturas ao redor do mundo.', 'Ana Costa', 'imagem.com/ana_costa.png', 'Ana é uma viajante apaixonada.', 20587.0, 120), ('Tendências da Moda', '2024-07-02', 'Descubra as últimas tendências da moda.', 'Lucas Martins', 'imagem.com/lucas_martins.png', 'Lucas é um estilista renomado.', 1500, 45);
```

Executando o Comando

Para inserir múltiplos registros:

1. Selecione o bloco de código **INSERT INTO**.
2. Clique no ícone do raio (⚡) para executar.
3. Se o comando for executado com sucesso, você verá a mensagem indicando que duas linhas foram inseridas.

Verificando os Dados Inseridos

Para visualizar os dados inseridos:

1. Passe o mouse sobre a tabela "posts".
2. Clique no terceiro ícone (visualizar dados).
3. Você verá as novas linhas inseridas na tabela.

Corrigindo Tipos de Dados

Problema com Tipos de Dados

Se tentarmos inserir um número com casas decimais em uma coluna do tipo **INT**, o banco de dados pode ignorar as casas decimais. Vamos corrigir isso definindo a coluna `quantidade_seguidores` como **FLOAT**:

sql

 Copiar código

```
USE blog; DROP TABLE IF EXISTS posts; CREATE TABLE posts ( post_titulo VARCHAR(45), post_data DATE, post_conteudo TEXT, post_autor VARCHAR(45), post_autor_foto_perfil VARCHAR(45), post_autor_bio VARCHAR(255), quantidade_seguidores FLOAT, quantidade_comentarios INT );
```

Executando o Comando de Criação da Tabela

Para criar a tabela, selecione todo o bloco de código do **CREATE TABLE** e clique no ícone do raio (⚡). Se tudo estiver correto, você verá uma mensagem de sucesso no rodapé do Workbench.

Inserindo Novos Dados com Correção

Agora vamos inserir novamente os dados na tabela corrigida:

sql

 Copiar código

```
INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio, quantidade_seguidores, quantidade_comentarios) VALUES ('Meu Primeiro Post', '2024-01-01', 'Este é o conteúdo do meu primeiro post.', 'João Silva', 'imagem.com/joao_silva.png', 'João é um desenvolvedor.', 1200.0, 30), ('Post sobre MySQL', '2024-02-15', 'Aprendendo sobre MySQL no curso de banco de dados.', 'Maria Souza', 'imagem.com/maria_souza.png', 'Maria é desenvolvedora de software.', 1500.0, 45), ('Dicas de JavaScript', '2024-03-10', 'Compartilhando dicas sobre JavaScript.', 'Pedro Lima', 'imagem.com/pedro_lima.png', 'Pedro é engenheiro de software.', 980.0, 25), ('Viagens pelo Mundo', '2024-07-01', 'Explorando diferentes culturas ao redor do mundo.', 'Ana Costa', 'imagem.com/ana_costa.png', 'Ana é uma viajante apaixonada.', 20587.0, 120), ('Tendências da Moda', '2024-07-02', 'Descubra as últimas tendências da moda.', 'Lucas Martins', 'imagem.com/lucas_martins.png', 'Lucas é um estilista renomado.', 1500.0, 45);
```

Executando o Comando

Para inserir múltiplos registros:

1. Selecione o bloco de código **INSERT INTO**.
2. Clique no ícone do raio (⚡) para executar.
3. Se o comando for executado com sucesso, você verá a mensagem indicando que cinco linhas foram inseridas.

Verificando os Dados Inseridos

Para visualizar os dados inseridos:

1. Passe o mouse sobre a tabela "posts".
2. Clique no terceiro ícone (visualizar dados).
3. Você verá as novas linhas inseridas na tabela com os valores corretos.

dia 35 video 07



Conclusão

Hoje, concluímos a inserção de dados na nossa tabela e aprendemos a corrigir problemas relacionados a tipos de dados.

Nas próximas aulas, exploraremos mais comandos para manipulação e consulta de dados.

Até a próxima aula!

1.8. Manipulação de Dados no MySQL

Nas aulas anteriores, vimos muito sobre a estrutura do banco de dados e das tabelas.

Agora, vamos nos concentrar nos dados armazenados dentro dessas tabelas e aprender a manipulá-los.

Preparação

Vamos usar o MySQL Workbench para inserir 100 linhas de dados fictícios em nossa tabela. Para começar, vamos limpar a tabela sem apagar sua estrutura.

Limpando a Tabela com **TRUNCATE**

O comando **TRUNCATE** é usado para remover todos os dados de uma tabela sem alterar sua estrutura:

sql

 Copiar código


```
TRUNCATE TABLE posts;
```

Selecione a linha e clique no ícone do raio (⚡) para executar. Isso limpará todos os dados da tabela **posts**, mantendo sua estrutura intacta.

Inserindo Dados na Tabela

Agora, vamos inserir 100 linhas de dados fictícios:

sql

 Copiar código

```
INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio,
quantidade_seguidores, quantidade_comentarios) VALUES ('Título 1', '2024-07-01', 'Conteúdo do post 1', 'Autor 1',
'imagem.com/autor1.png', 'Biografia do Autor 1', 1000, 50), ('Título 2', '2024-07-02', 'Conteúdo do post 2', 'Autor 2',
'imagem.com/autor2.png', 'Biografia do Autor 2', 2000, 60), -- continue inserindo os outros 98 registros...
```

Selecione todas as linhas do comando **INSERT INTO** e clique no ícone do raio (⚡) para executar.

Verificando Erros de Inserção

Se houver um erro indicando que o tipo de dado é muito longo, precisamos ajustar a coluna para aceitar mais caracteres.

Ajustando o Tipo de Dado da Coluna

Vamos modificar a coluna **post_titulo** para aceitar até 255 caracteres:


sql

 Copiar código

```
ALTER TABLE posts MODIFY COLUMN post_titulo VARCHAR(255);
```

Selecione a linha do comando **ALTER TABLE** e clique no ícone do raio (⚡) para executar. Verifique a alteração com o comando **DESCRIBE**:

sql

 Copiar código

```
DESCRIBE posts;
```

Inserindo Dados Novamente

Após ajustar a coluna, insira novamente os dados:

sql

 Copiar código

```
INSERT INTO posts (post_titulo, post_data, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio,
quantidade_seguidores, quantidade_comentarios) VALUES ('Título 1', '2024-07-01', 'Conteúdo do post 1', 'Autor 1',
'imagem.com/autor1.png', 'Biografia do Autor 1', 1000, 50), ('Título 2', '2024-07-02', 'Conteúdo do post 2', 'Autor 2',
'imagem.com/autor2.png', 'Biografia do Autor 2', 2000, 60), -- continue inserindo os outros 98 registros...
```

Verificando os Dados

Para visualizar os dados inseridos, passe o mouse sobre a tabela `posts` e clique no terceiro ícone (visualizar dados).

Explorando Comandos SQL

Vamos explorar os principais comandos SQL para manipulação de dados: `SELECT`, `UPDATE`, `DELETE` e `WHERE`.

Abrindo uma Nova Aba no Workbench

Para abrir uma nova aba limpa no Workbench, clique no ícone de mais (+) no canto superior direito.

Comando `SELECT`

O comando `SELECT` é usado para consultar dados em uma tabela. Exemplos:

sql

 Copiar código

```
-- Selecionar todos os dados SELECT * FROM posts; -- Selecionar dados específicos SELECT post_titulo, post_autor FROM posts;
-- Selecionar dados com filtro SELECT * FROM posts WHERE post_autor = 'Autor 1';
```

Comando `UPDATE`

O comando `UPDATE` é usado para atualizar dados existentes em uma tabela. Exemplo:

sql

 Copiar código

```
UPDATE posts SET quantidade_seguidores = 3000 WHERE post_autor = 'Autor 1';
```

Comando `DELETE`

O comando `DELETE` é usado para remover dados de uma tabela. Exemplo:

sql

 Copiar código

```
DELETE FROM posts WHERE post_autor = 'Autor 1';
```

Comando `WHERE`

A cláusula `WHERE` é usada para especificar condições ao selecionar, atualizar ou deletar dados. Exemplo:

sql

 Copiar código

```
SELECT * FROM posts WHERE post_data > '2024-07-01';
```

dia 35 video 8



Conclusão

Hoje aprendemos a limpar tabelas, inserir múltiplas linhas de dados, ajustar tipos de colunas e explorar comandos SQL básicos para manipulação de dados.

Esses comandos são essenciais para gerenciar e trabalhar com dados no MySQL. Na próxima aula, aprofundaremos mais nos comandos SQL e em consultas mais complexas.

Até a próxima aula!

1.9. Comandos SQL: SELECT

Agora, vamos focar na manipulação e consulta de dados usando o comando **SELECT**.

Comando SELECT

O comando **SELECT** é utilizado para consultar dados armazenados em uma tabela. Ele permite escolher quais colunas e linhas exibir a partir das informações salvas.

Sintaxe Básica do SELECT

A estrutura básica do comando **SELECT** é:

sql

 Copiar código

```
SELECT coluna1, coluna2, ... FROM nome_da_tabela;
```

Selecionando Colunas Específicas

Vamos começar selecionando colunas específicas relacionadas ao autor de um post. As colunas que contêm essas informações são:

- post_autor
- post_autor_foto_perfil
- post_autor_bio

Vamos escrever o comando para selecionar essas colunas:

sql

 Copiar código


```
SELECT post_autor, post_autor_foto_perfil, post_autor_bio FROM posts;
```

Selecione a linha e clique no ícone do raio (⚡) para executar. Isso exibirá todas as linhas, mas apenas as colunas especificadas.

Selecionando Uma Única Coluna

Você pode selecionar uma única coluna, se desejar. Por exemplo, para selecionar a quantidade de seguidores:

sql

 Copiar código


```
SELECT quantidade_seguidores FROM posts;
```

Selecione a linha e clique no ícone do raio (⚡) para executar. Isso exibirá apenas a coluna **quantidade_seguidores** de todas as linhas.

Selecionando Todas as Colunas

Para selecionar todas as colunas de uma tabela, você pode usar o asterisco (*):

sql

 Copiar código

```
SELECT * FROM posts;
```

Isso exibirá todas as colunas e todas as linhas da tabela **posts**.

Mudando a Ordem das Colunas no SELECT

A ordem das colunas no resultado pode ser diferente da ordem na tabela. Você pode especificar a ordem das colunas no comando **SELECT**:

sql

 Copiar código

```
SELECT post_data, post_titulo, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio, quantidade_seguidores,
quantidade_comentarios FROM posts;
```

Tratando Erros

Se você tentar selecionar uma coluna que não existe, receberá um erro. Por exemplo:

sql

 Copiar código

```
SELECT post_hora FROM posts;
```

Isso resultará em um erro porque a coluna **post_hora** não existe na tabela **posts**.

Formatação para Melhor Visualização

Para facilitar a leitura, você pode formatar o comando **SELECT** colocando cada coluna em uma nova linha:

sql

 Copiar código

```
SELECT post_titulo, post_data, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio, quantidade_seguidores,
quantidade_comentarios FROM posts;
```

Exemplo Completo

Aqui está um exemplo completo do comando **SELECT**, exibindo todas as colunas da tabela **posts**:

sql

 Copiar código

```
SELECT post_titulo, post_data, post_conteudo, post_autor, post_autor_foto_perfil, post_autor_bio, quantidade_seguidores,
quantidade_comentarios FROM posts;
```

Selecione o bloco de código e clique no ícone do raio (⚡) para executar. Isso exibirá todas as colunas e todas as linhas da tabela **posts**.

dia 35 video 9



Conclusão

Hoje, exploramos o comando **SELECT** e aprendemos como usá-lo para consultar dados em uma tabela.

Vimos como selecionar colunas específicas, uma única coluna, todas as colunas e como mudar a ordem das colunas no resultado.

Na próxima aula, vamos explorar os comandos **UPDATE** e **DELETE**, além de usar a cláusula **WHERE** para filtrar dados.

Até a próxima aula!

1.10. SELECT, *, AS, ORDER BY

Revisão

Na aula anterior, aprendemos a usar o comando `SELECT` para consultar dados em nossa tabela.

Vamos continuar explorando o `SELECT`, além de aprender sobre `*`, `AS` e `ORDER BY`.

Selecionando Todas as Colunas


Quando queremos selecionar todas as colunas de uma tabela sem ter que listar cada uma, podemos usar o asterisco (`*`). Este é conhecido como "all" (todos).

Sintaxe

sql

 Copiar código

```
SELECT * FROM posts;
```

Selecione a linha e clique no ícone do raio () para executar. Isso trará todas as colunas e todas as linhas da tabela `posts`.

Usando Alias (AS)

Podemos usar alias (`AS`) para renomear colunas no resultado da consulta. Isso é útil para tornar os nomes das colunas mais legíveis ou para remover caracteres de sublinhado.

Exemplo com Alias


Vamos selecionar algumas colunas e usar alias para renomeá-las:

sql

 Copiar código

```
SELECT post_autor AS autor, post_autor_bio AS biografia_do_autor FROM posts;
```

Executando

Selecione a linha e clique no ícone do raio () para executar. O resultado trará as colunas `autor` e `biografia_do_autor` com os dados das colunas `post_autor` e `post_autor_bio`.

Ordenando Resultados (ORDER BY)

Podemos ordenar os resultados da consulta usando a cláusula `ORDER BY`. É possível ordenar os resultados em ordem ascendente (`ASC`) ou descendente (`DESC`).

Exemplo de Ordenação Ascendente


Vamos ordenar os resultados pela quantidade de seguidores em ordem ascendente:

sql

 Copiar código

```
SELECT * FROM posts ORDER BY quantidade_seguidores ASC;
```


Executando

Selecione a linha e clique no ícone do raio () para executar. Os resultados serão ordenados pela coluna `quantidade_seguidores` do menor para o maior.

Exemplo de Ordenação Descendente

Agora, vamos ordenar os resultados pela quantidade de seguidores em ordem descendente:

sql

 Copiar código

```
SELECT * FROM posts ORDER BY quantidade_seguidores DESC;
```

Executando

Selecione a linha e clique no ícone do raio (⚡) para executar. Os resultados serão ordenados pela coluna `quantidade_seguidores` do maior para o menor.

Combinando Alias e Ordenação

Podemos combinar alias e ordenação em uma única consulta:

sql

 Copiar código

```
SELECT post_titulo AS título, post_autor AS autor, quantidade_seguidores AS seguidores FROM posts ORDER BY seguidores DESC;
```

Executando

Selecione a linha e clique no ícone do raio (⚡) para executar. O resultado trará as colunas renomeadas e os dados ordenados pela quantidade de seguidores em ordem decendente.

dia 35 video 10



Conclusão

Hoje, exploramos como selecionar todas as colunas de uma tabela usando `*`, como usar alias (`AS`) para renomear colunas e como ordenar resultados com `ORDER BY`. Também vimos como combinar esses conceitos em uma única consulta.

Na próxima aula, vamos explorar mais comandos SQL, como `UPDATE` e `DELETE`, e aprender a usar a cláusula `WHERE` para filtrar dados.

Até a próxima aula!

1.11. WHERE e Operadores de Comparação

Na última aula, aprendemos a usar o comando `SELECT` para consultar dados em nossa tabela e como usar o `*` para selecionar todas as colunas.

Também vimos como usar alias (`AS`) para renomear colunas e ordenar os resultados com `ORDER BY`. Agora, vamos aprender a filtrar resultados usando a cláusula `WHERE` e operadores de comparação.


Filtrando Resultados com WHERE

O comando `WHERE` é usado para filtrar registros que atendem a uma condição específica. Vamos começar com um exemplo simples.

Selecionando Todas as Colunas com Filtro

Queremos selecionar todos os posts em que a quantidade de seguidores é maior que 2000:

sql

 Copiar código

```
USE blog; SELECT * FROM posts WHERE quantidade_seguidores > 2000;
```

Executando

Selecione as linhas e clique no ícone do raio (⚡) para executar. Isso retornará apenas os registros onde a coluna `quantidade_seguidores` é maior que 2000.

Explicação

- `SELECT *`: Seleciona todas as colunas.
- `FROM posts`: Especifica a tabela `posts`.
- `WHERE quantidade_seguidores > 2000`: Filtra registros onde a quantidade de seguidores é maior que 2000.

Operadores de Comparação

Além de `>`, temos vários outros operadores de comparação que podemos usar com `WHERE`.

Operadores Comuns

- `=`: Igual a
- `!=` ou `<>`: Diferente de
- `<`: Menor que
- `<=`: Menor ou igual a
- `>=`: Maior ou igual a

Exemplos de Uso

Igual a

Selecionar todos os posts onde a quantidade de seguidores é exatamente 1000:

sql

 Copiar código

```
SELECT * FROM posts WHERE quantidade_seguidores = 1000;
```

Diferente de

Selecionar todos os posts onde a quantidade de seguidores não é 1500:

sql


 Copiar código

```
SELECT * FROM posts WHERE quantidade_seguidores != 1500;
```

Menor que

Selecionar todos os posts onde a quantidade de seguidores é menor que 500:

sql


 Copiar código

```
SELECT * FROM posts WHERE quantidade_seguidores < 500;
```

Menor ou Igual a

Selecionar todos os posts onde a quantidade de seguidores é menor ou igual a 1200:

sql

 Copiar código

```
SELECT * FROM posts WHERE quantidade_seguidores <= 1200;
```

Maior ou Igual a

Selecionar todos os posts onde a quantidade de seguidores é maior ou igual a 3000:

sql

 Copiar código

```
SELECT * FROM posts WHERE quantidade_seguidores >= 3000;
```

Executando Exemplos

Selecione cada linha de comando e clique no ícone do raio (⚡) para executar. Observe os resultados retornados em cada caso.

dia 35 video 11



Conclusão

Hoje, exploramos como usar a cláusula **WHERE** para filtrar registros em uma tabela usando operadores de comparação.

Na próxima aula, vamos aprender a usar a cláusula **LIMIT** para restringir o número de registros retornados e a cláusula **LIKE** para realizar pesquisas de padrões.

Até a próxima aula!

