Aula 03: Data Query Language

Conceitos Fundamentais e Estrutura do MySQL

Bancos de dados são sistemas que nos permitem armazenar, gerenciar e recuperar informações de forma organizada e eficiente. Imagine que você precisa guardar dados de vendas de uma loja. Você poderia usar uma planilha, mas e se a loja tiver milhares de clientes e milhões de vendas? A planilha se tornaria lenta e difícil de gerenciar. É aí que entra o banco de dados.

O **MySQL** é uma das ferramentas para sistemas de gerenciamento de banco de dados (SGBD) mais populares. Ele armazena os dados em **tabelas**, que são organizadas em linhas e colunas, como uma planilha. Cada tabela é composta por:

- **Colunas:** Representam os atributos dos dados (ex: `nome_cliente`, `preco_produto`).
- **Linhas:** Representam um único registro ou um conjunto completo de dados (ex: todas as informações de um cliente).

Atividade Orientada: Criando e Importando Dados

1. Preparando o Ambiente

- Abra o **XAMPP** (painel-controlador) e inicie os módulos **Apache** e **MySQL**.
- Acesse o **PhpMyAdmin** em seu navegador (geralmente em `http://localhost:8080/phpmyadmin`).

2. Criando o Banco de Dados

Vamos criar um pequeno banco de dados para a nossa atividade. Usaremos **SQL (Structured Query Language)**, que é a linguagem padrão para interagir com bancos de dados.

O comando para criar um banco de dados é `CREATE DATABASE` (ou CREATE SCHEMA):

```
CREATE DATABASE vendas_online;
```

3. Criando uma Tabela e Importando Dados

Para este exemplo, usaremos um conjunto de dados sobre produtos.

Primeiro, selecione o banco de dados `vendas_online` pelo menu clicando com o botão direito do mouse (*Set as Default Schema*). Depois, crie a tabela `produtos` com o comando `CREATE TABLE`:

```
USE vendas_online;

CREATE TABLE produtos (
  id_produto INT PRIMARY KEY,
  nome VARCHAR(255),
  categoria VARCHAR(100),
  preco DECIMAL(10, 2),
  estoque INT
);
```

Legenda:

- `USE vendas_online;` informa ao MySQL que queremos trabalhar com esse banco de dados.
- `CREATE TABLE produtos` cria uma nova tabela chamada `produtos`.
- `id_produto INT PRIMARY KEY`: Cria uma coluna chamada `id_produto` que armazena números inteiros (`INT`). `PRIMARY KEY` a define como chave primária, ou seja, um identificador único para cada registro.
- `nome VARCHAR(255)`: Cria uma coluna para o nome do produto, que armazena texto de até 255 caracteres (`VARCHAR`).
- `categoria VARCHAR(100)`: Coluna para a categoria do produto.
- `preco DECIMAL(10, 2)`: Coluna para o preço, que armazena números decimais com 10 dígitos no total e 2 após a vírgula.
- `estoque INT`: Coluna para a quantidade em estoque.

Importação de Dados:

- No PhpMyAdmin, selecione a tabela `produtos` e clique na aba **"Importar"**.
- Escolha um arquivo CSV ou SQL com seus dados e siga as instruções para importá-los.

Data Query Language (DQL)

É a parte do SQL usada para consultar e obter dados do banco de dados. O comando mais importante do DQL é o `SELECT`.

1. Listando Todos os Dados

```
SELECT * FROM produtos;
```

Explicação: O `SELECT *` significa "selecione todas as colunas". `FROM produtos` especifica de qual tabela queremos os dados.

2. Consultando Dados Específicos

Exemplo 1: Selecionar apenas o nome e o preço dos produtos.

```
SELECT nome, preco FROM produtos;
```

Exemplo 2: Filtrar produtos com preço acima de R$ 50,00.

```
SELECT * FROM produtos WHERE preco > 50;
```

Exemplo 3: Encontrar produtos que estão na categoria "Eletrônicos".

```
SELECT nome, categoria FROM produtos WHERE categoria = 'Eletrônicos';
```

Atividade: Consultas ao Banco de Dados

Respondam às seguintes perguntas usando consultas SQL no PhpMyAdmin ou MySQL:

1. Selecione todos os produtos com estoque menor que 20;
2. Encontre o nome e a categoria dos produtos com preço entre R$ 100,00 e R$ 500,00.
3. Conte quantos produtos existem na tabela. (Dica: lembre-se dos operadores básicos que já utilizamos no Python Pandas)

Conectando Python ao MySQL

Uma das grandes vantagens de usar bancos de dados é a capacidade de integrá-los a outras linguagens de programação. Vamos usar **Python** para interagir com o nosso banco de dados.

Primeiro, você precisa ter instalada a biblioteca `mysql-connector-python`:

```
pip install mysql-connector-python
```

1. Obtendo Dados com Python

Vamos replicar a query `SELECT * FROM produtos;` no VSCode:

```
import mysql.connector

# 1. Conectar ao banco de dados
conexao = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="vendas_online"
)
```

```
# 2. Criar um objeto cursor para executar as queries
cursor = conexao.cursor()

# 3. Definir a query
query = "SELECT * FROM produtos"

# 4. Executar a query
cursor.execute(query)

# 5. Obter os resultados
resultados = cursor.fetchall()

# 6. Exibir os resultados
for linha in resultados:
    print(linha)

# 7. Fechar a conexão
cursor.close()
conexao.close()
```

Legenda:

- **`mysql.connector.connect()`:** Esta função estabelece a conexão com o servidor MySQL. Você precisa passar as credenciais corretas (`host`, `user`, `password`) e o nome do banco de dados (`database`).
- **`conexao.cursor()`:** O cursor é um objeto que nos permite executar comandos SQL.
- **`cursor.execute(query)`:** Executa a consulta SQL que definimos.
- **`cursor.fetchall()`:** Pega todos os resultados da consulta e os retorna como uma lista de tuplas.
- Os loops `for` são usados para imprimir cada registro.
- É necessário fechar a conexão no final para liberar os recursos do servidor.

2. Criando uma Função para Obtenção de Dados

Podemos encapsular essa lógica em uma função para reutilizá-la facilmente:

```
import mysql.connector

def obter_dados_do_banco(query):
    try:
        conexao = mysql.connector.connect(
            host="localhost",
            user="root",
            password="",
            database="vendas_online"
        )
        cursor = conexao.cursor()
        cursor.execute(query)
```

```
        resultados = cursor.fetchall()
        return resultados
    except mysql.connector.Error as erro:
        print(f"Erro ao conectar ao MySQL: {erro}")
        return None
    finally:
        if 'conexao' in locals() and conexao.is_connected():
            cursor.close()
            conexao.close()

# Usando a função
query_produtos = "SELECT * FROM produtos WHERE preco > 100"
dados_filtrados = obter_dados_do_banco(query_produtos)

if dados_filtrados:
    for produto in dados_filtrados:
        print(produto)
```

Legenda:

- A função `obter_dados_do_banco` recebe a `query` como parâmetro, o que a torna flexível.
- Adicionamos um bloco `try...except...finally` para lidar com possíveis erros de conexão e garantir que a conexão seja sempre fechada, mesmo se algo der errado.

Atividade Prática: MySQL com Python

1. Importe uma nova base de dados de sua escolha (utilize algum dataset sugerido nos sites da aula anterior);
2. Crie um novo arquivo Python;
3. Conecte-se ao seu novo banco de dados via Python;
4. Crie uma função para consultar a tabela;
5. Execute consultas para responder às seguintes perguntas, imprimindo os resultados:
    - Liste o nome de todos os elementos;.
    - Encontre o nome e algum valor quantitativo ligado a esses elementos;
    - Conte quantos elementos possuem algum filtro de categoria.