FAP UFRN Softex 2024 – Módulo 3 – Introdução a Bancos de Dados

Prof. José Alfredo F. Costa

Continuação – Sistema de biblioteca usando MySQL para o banco de dados e Python para a lógica de negócios.

Sistema de biblioteca usando MySQL para o banco de dados e Python para a lógica de negócios.

Objetiva armazenar e gerenciar os dados de forma eficiente, além de criar uma interface para interagir com o sistema.

Esquema básico de como podemos estruturar isso, incluindo a criação do banco de dados, as tabelas necessárias, e um exemplo de como interagir com o banco de dados usando Python.

1. Estrutura do Banco de Dados MySQL

Primeiro, vamos criar as tabelas necessárias no MySQL:

```sql
-- Criação da tabela de Livros
CREATE TABLE livros (
    id INT AUTO_INCREMENT PRIMARY KEY,
    titulo VARCHAR(255) NOT NULL,
    autor VARCHAR(255) NOT NULL,
    ano_publicacao INT,
    categoria VARCHAR(100),
    disponivel BOOLEAN DEFAULT TRUE
);

-- Criação da tabela de Usuários
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(255) NOT NULL,
    limite_livros INT DEFAULT 3
);

-- Criação da tabela de Empréstimos
CREATE TABLE emprestimos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    livro_id INT,
    usuario_id INT,
    data_emprestimo DATE,
    data_devolucao_prevista DATE,
    data_devolucao_real DATE,
    FOREIGN KEY (livro_id) REFERENCES livros(id),
```

```
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id)
);
```

2. Código Python para interagir com o banco de dados

Agora, vamos criar uma classe em Python para interagir com o banco de dados:

```
import mysql.connector
from datetime import datetime, timedelta

class BibliotecaDB:
    def __init__(self, host, user, password, database):
        self.conn = mysql.connector.connect(
            host=host,
            user=user,
            password=password,
            database=database
        )
        self.cursor = self.conn.cursor()

    def adicionar_livro(self, titulo, autor, ano_publicacao,
categoria):
        sql = """INSERT INTO livros (titulo, autor, ano_publicacao,
categoria)
                 VALUES (%s, %s, %s, %s)"""
        values = (titulo, autor, ano_publicacao, categoria)
        self.cursor.execute(sql, values)
        self.conn.commit()
        return self.cursor.lastrowid

    def adicionar_usuario(self, nome, limite_livros=3):
        sql = "INSERT INTO usuarios (nome, limite_livros) VALUES (%s,
%s)"
        values = (nome, limite_livros)
        self.cursor.execute(sql, values)
        self.conn.commit()
        return self.cursor.lastrowid

    def emprestar_livro(self, livro_id, usuario_id):
        # Verificar se o livro está disponível
        self.cursor.execute("SELECT disponivel FROM livros WHERE id =
%s", (livro_id,))
        disponivel = self.cursor.fetchone()[0]
        if not disponivel:
            return False, "Livro não disponível"

        # Verificar se o usuário pode emprestar mais livros
```

```
        self.cursor.execute("SELECT COUNT(*) FROM emprestimos WHERE
usuario_id = %s AND data_devolucao_real IS NULL", (usuario_id,))
        livros_emprestados = self.cursor.fetchone()[0]
        self.cursor.execute("SELECT limite_livros FROM usuarios WHERE
id = %s", (usuario_id,))
        limite_livros = self.cursor.fetchone()[0]
        if livros_emprestados >= limite_livros:
            return False, "Usuário atingiu o limite de livros
emprestados"

        # Realizar o empréstimo
        data_emprestimo = datetime.now().date()
        data_devolucao_prevista = data_emprestimo + timedelta(days=14)
# 2 semanas de prazo
        sql = """INSERT INTO emprestimos (livro_id, usuario_id,
data_emprestimo, data_devolucao_prevista)
                 VALUES (%s, %s, %s, %s)"""
        values = (livro_id, usuario_id, data_emprestimo,
data_devolucao_prevista)
        self.cursor.execute(sql, values)

        # Atualizar o status do livro
        self.cursor.execute("UPDATE livros SET disponivel = FALSE
WHERE id = %s", (livro_id,))
        self.conn.commit()
        return True, "Livro emprestado com sucesso"

    def devolver_livro(self, livro_id, usuario_id):
        sql = """UPDATE emprestimos
                 SET data_devolucao_real = %s
                 WHERE livro_id = %s AND usuario_id = %s AND
data_devolucao_real IS NULL"""
        values = (datetime.now().date(), livro_id, usuario_id)
        self.cursor.execute(sql, values)

        # Atualizar o status do livro
        self.cursor.execute("UPDATE livros SET disponivel = TRUE WHERE
id = %s", (livro_id,))
        self.conn.commit()
        return "Livro devolvido com sucesso"

    def fechar_conexao(self):
        self.cursor.close()
        self.conn.close()

# Exemplo de uso
if __name__ == "__main__":
    db = BibliotecaDB("localhost", "seu_usuario", "sua_senha",
"biblioteca")
```

```
    # Adicionar um livro
    livro_id = db.adicionar_livro("1984", "George Orwell", 1949,
"Ficção Científica")
    print(f"Livro adicionado com ID: {livro_id}")

    # Adicionar um usuário
    usuario_id = db.adicionar_usuario("Maria")
    print(f"Usuário adicionado com ID: {usuario_id}")

    # Emprestar um livro
    sucesso, mensagem = db.emprestar_livro(livro_id, usuario_id)
    print(mensagem)

    # Devolver um livro
    mensagem = db.devolver_livro(livro_id, usuario_id)
    print(mensagem)

    db.fechar_conexao()
```

Este código Python cria uma classe `BibliotecaDB` que encapsula todas as operações de banco de dados necessárias para o sistema de biblioteca. Ele inclui métodos para adicionar livros e usuários, emprestar e devolver livros, além de lidar com as restrições de negócios (como verificar a disponibilidade do livro e o limite de empréstimos do usuário).

Para usar este sistema:

1. Configure um servidor MySQL e crie um banco de dados chamado "biblioteca".
2. Execute o script SQL fornecido para criar as tabelas necessárias.
3. Instale a biblioteca MySQL Connector para Python usando pip:

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

1. Use o código Python fornecido, substituindo "seu_usuario" e "sua_senha" pelas suas credenciais MySQL reais.

Este é um esquema básico que você pode expandir conforme necessário. Você pode adicionar mais funcionalidades, como:

- Pesquisa de livros por título, autor ou categoria
- Geração de relatórios (por exemplo, livros mais populares, usuários com mais empréstimos)
- Interface de usuário (GUI ou web) para interagir com o sistema

Para tornar o sistema mais robusto, você também pode considerar:

1. Adicionar tratamento de erros mais abrangente.
2. Implementar logging para rastrear operações importantes.
3. Usar um ORM (Object-Relational Mapping) como SQLAlchemy para uma abstração de banco de dados mais poderosa.
4. Implementar testes unitários para garantir que todas as funções estejam funcionando corretamente.

Este esquema fornece uma base sólida para um sistema de gerenciamento de biblioteca usando MySQL e Python, que você pode expandir e personalizar conforme suas necessidades específicas.

Explicação de como instalar o MySQL no Windows, como configurá-lo para uso com Python, e como usar tudo isso no VSCode.

Vamos dividir isso em etapas:

1. Instalando o MySQL no Windows

 Baixe o MySQL Installer:

- Vá para https://dev.mysql.com/downloads/installer/
- Escolha "Windows (x86, 32-bit), MSI Installer" e clique em "Download"
- Você pode precisar criar uma conta Oracle ou escolher "No thanks, just start my download"

 Execute o instalador:

- Dê permissão de administrador se solicitado
- Escolha "Custom" na tela de seleção de configuração

 Selecione os produtos:

- Expanda "MySQL Servers" e selecione a versão mais recente do MySQL Server
- Expanda "Applications" e selecione "MySQL Workbench" (uma interface gráfica útil)
- Clique em "Next" e então em "Execute" para instalar os produtos selecionados

 Configuração do MySQL Server:

- Escolha "Standalone MySQL Server / Classic MySQL Replication"
- Em "Type and Networking", mantenha as configurações padrão
- Em "Authentication Method", escolha "Use Strong Password Encryption"
- Defina a senha root do MySQL. Guarde esta senha com segurança!
- Em "Windows Service", mantenha as configurações padrão
- Clique em "Execute" para aplicar a configuração

 Conclua a instalação:

- Clique em "Finish" quando a instalação estiver completa

 Verifique a instalação:

- Abra o prompt de comando e digite: mysql --version
- Você deve ver a versão do MySQL instalada

2. Configurando o MySQL para uso com Python

Agora que o MySQL está instalado, vamos configurá-lo para uso com Python:

1. Instale o conector MySQL para Python:
    - Abra o prompt de comando e digite: `pip install mysql-connector-python`
2. Teste a conexão:
    - Abra o Python IDLE ou um arquivo .py e execute o seguinte código:

```
import mysql.connector


try:
    connection = mysql.connector.connect(
        host="localhost",
        user="root",
        password="sua_senha_root"
    )
    print("Conexão bem-sucedida!")
except mysql.connector.Error as err:
    print(f"Erro: {err}")
finally:
    if 'connection' in locals() and connection.is_connected():
        connection.close()
        print("Conexão fechada.")
```

- Substitua "sua_senha_root" pela senha que você definiu durante a instalação do MySQL

• Configurando e usando o VSCode para desenvolvimento Python com MySQL

 Instale o VSCode:

- Baixe e instale o VSCode de https://code.visualstudio.com/

 Instale as extensões necessárias:

- Abra o VSCode
- Vá para a aba de extensões (ícone de quadrados no menu lateral esquerdo)
- Pesquise e instale as seguintes extensões:
    - Python (da Microsoft)
    - MySQL (da cweijan)

 Configure o interpretador Python:

- Abra um novo arquivo .py
- Pressione Ctrl+Shift+P para abrir a paleta de comandos
- Digite "Python: Select Interpreter" e selecione o interpretador Python instalado no seu sistema

 Configure a extensão MySQL:

- Clique no ícone da extensão MySQL na barra lateral esquerda
- Clique em "+" para adicionar uma nova conexão
- Preencha os detalhes da conexão (host, usuário, senha)
- Teste a conexão clicando em "Test Connection"

 Crie um novo projeto:

- Crie uma nova pasta para seu projeto
- Abra esta pasta no VSCode (File > Open Folder)
- Crie um novo arquivo Python (por exemplo, biblioteca.py)

 Desenvolva seu código:

- Use o código Python fornecido anteriormente como base
- Modifique conforme necessário para seu projeto

 Execute seu código:

- Abra um terminal no VSCode (Terminal > New Terminal)
- Execute seu script com o comando: python biblioteca.py

 Use o depurador:

- Defina pontos de interrupção clicando à esquerda dos números de linha
- Pressione F5 para iniciar a depuração
- Use os controles de depuração para passar pelo código passo a passo

 Use a extensão MySQL para visualizar e gerenciar seu banco de dados:

- Expanda sua conexão no painel da extensão MySQL
- Visualize tabelas, execute consultas e gerencie seu banco de dados diretamente do VSCode

Dicas adicionais:

1. Segurança: Nunca armazene senhas diretamente no código. Use variáveis de ambiente ou arquivos de configuração para armazenar informações sensíveis.
2. Ambiente virtual: Considere usar um ambiente virtual Python para seu projeto. Isso ajuda a manter as dependências do projeto isoladas. Você pode criar um com:

   ```
   python -m venv

   venv venv\Scripts\activate # No Windows
   ```

Requisitos: Mantenha um arquivo `requirements.txt` com as dependências do seu projeto. Você pode criá-lo com:

```
pip freeze > requirements.txt
```

1. Controle de versão: Use o Git para controle de versão. O VSCode tem integração Git embutida, facilitando o gerenciamento de versões do seu código.
2. Documentação: Comente seu código adequadamente e considere usar docstrings para documentar suas funções e classes.

Seguindo estas etapas, você terá um ambiente de desenvolvimento completo para trabalhar com Python e MySQL no Windows usando o VSCode.

Isso fornecerá uma base sólida para desenvolver seu sistema de biblioteca ou qualquer outro projeto que envolva Python e MySQL.