Escola Profissional
BENTO DE JESUS CARAÇA
DELEGAÇÃO DO BARREIRO

Criação

Beatriz Duarte Gomes, Joel Matoso

Curso Técnico de Gestão e Programação de Sistemas Informáticos

Escola Profissional Bento de Jesus Caraça, Delegação do Barreiro

P.S.I.: Programação e Sistemas de Informação

Coordenador Marcelo Simão

05 de fevereiro de 2025

Índice

Índice de Ilustrações

Introdução

Neste relatório explicaremos o código realizado nas aulas da disciplina de Programação de Sistemas Informáticos (PSI), proporcionado pelo professor Diogo Oliveira.

O código fornecido é um conjunto de scripts em *Python* que interagem com uma base de dados *SQLite*. Ele realiza a criação de tabelas, inserção de dados, exclusão de registos e tabelas, consultas utilizando *INNER JOIN* e atualização de registos. A base de dados gere informações sobre jogos, incluindo os seus gêneros e plataformas.

Estrutura da Base

O banco de dados contém três tabelas principais:

- **jogos:** Armazena as informações sobre os jogos, incluindo nome, ano de lançamento, gênero e plataforma.
- **generos:** Contém os diferentes gêneros de jogos.
- **plataformas:** Lista as plataformas em que os jogos estão disponíveis.

A relação entre as tabelas ocorre por meio das chaves estrangeiras *genero_id* e *plataforma_id* na tabela **jogos**, referenciando as tabelas **generos e plataformas**, respetivamente.

Conexão com a Base de Dados

O código inicia com a importação da biblioteca sqlite3, necessária para interagir com a base de dados *SQLite*. Em seguida, é estabelecida uma conexão com a base de dados localizado no caminho:

- ***C:\Users\ba2434\Documents\Projeto_Final\NewGames\master\database\exemplo.db***

Se o arquivo do banco de dados não existir, ele será criado automaticamente.

Criação das Tabelas (*Create*)

O código inicia verificando se as tabelas **jogos, generos e plataformas** existem. Caso não existam, elas são criadas com as chaves primárias e estrangeiras adequadas.

1. Tabela jogos

A tabela jogos armazena informações sobre os jogos cadastrados. Possui os seguintes campos:

- ***id:*** Identificador único do jogo (chave primária, autoincrementada).
- **nome:** Nome do jogo (campo obrigatório, tipo *TEXT*).
- **ano:** Ano de lançamento (campo obrigatório, tipo *INTEGER*).

- ***genero_id:*** Chave estrangeira referenciando a tabela **generos**.
- ***plataforma_id:*** Chave estrangeira referenciando a tabela **plataformas**.

```
CREATE TABLE IF NOT EXISTS jogos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    ano INTEGER NOT NULL,
    genero_id INTEGER NOT NULL,
    plataforma_id INTEGER NOT NULL,
    FOREIGN KEY (genero_id) REFERENCES genero(id),
    FOREIGN KEY (plataforma_id) REFERENCES plataformas(id)
);
```

Figura 1 - Tabela Jogos

2. Tabela generos

A tabela generos armazena os diferentes gêneros dos jogos, possuindo:

- ***id:*** Identificador único do gênero (chave primária, autoincrementada).
- **nome:** Nome do gênero (campo obrigatório, tipo *TEXT*).

```
CREATE TABLE IF NOT EXISTS generos (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL
);
```

Figura 2 - Tabela Generos

3. Tabela plataformas

A tabela plataformas armazena as plataformas em que os jogos estão disponíveis, possuindo:

- ***id:*** Identificador único da plataforma (chave primária, autoincrementada).
- **nome:** Nome da plataforma (campo obrigatório, tipo *TEXT*).

```
CREATE TABLE IF NOT EXISTS plataformas (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL
);
```

Figura 3 - Tabela Plataformas

4. Confirmação e Fechamento da Conexão

Depois de criar as tabelas, o comando *commit()* é utilizado para salvar as alterações no banco de dados.

Por fim, a conexão é fechada para evitar consumo desnecessário de recursos.

```
conexao.commit()
conexao.close()
```

Figura 4 - Salvar e Fechar Ligação

Inserção de Dados (*Insert*)

O código insere dados em três tabelas de um banco de dados:

- **Tabela generos:** Adiciona tipos de gêneros de jogos como "*RPG*", "*FPS*", "Ritmo", etc.

```
cursor.execute('''
                INSERT INTO generos (nome)
                VALUES
                ("RPG"),
                ("FPS"),
                ("Ritmo"),
                ("Sobrevivência e Ação"),
                ("Exploração"),
                ("Quebra-cabeça")
                ''')
```

Figura 5 - Insert na Tabela Generos

- **Tabela plataformas:** Adiciona plataformas de jogos como "*PC*", "*Nintendo*", "*PlayStation*", etc.

```
cursor.execute('''
                INSERT INTO plataformas (nome)
                VALUES
                ("PC"),
                ("Nintendo"),
                ("PlayStation"),
                ("Xbox"),
                ("Arcade"),
                ("Mobile")
                ''')
```

Figura 6 - Insert na Tabela Plataformas

- **Tabela jogos:** Adiciona informações sobre jogos, incluindo nome, ano de lançamento, gênero e plataforma correspondente, referenciados pelas *IDs* dessas tabelas.

```
cursor.execute('''
               INSERT INTO jogos (nome, ano, genero_id, plataforma_id)
               VALUES
               ("Valorant", 2020, 2, 1),
               ("MaiMai", 2012, 3, 5),
               ("Hogwarts Legacy", 2023, 6, 3),
               ("Minecraft", 2009, 4, 1),
               ("Hollow Knight", 2017, 4, 4),
               ("Grand Summoners", 2016, 1, 6)
               ''')
```

Figura 7 - Insert na Tabela Jogos

Consulta com *INNER JOIN* (*Select*)

Utilizamos as chaves estrangeiras (***genero_id*** e ***plataforma_id)*** e o *INNER JOIN* entre as três tabelas (**jogos, generos, plataformas**) para obter os nomes dos jogos, os gêneros e as plataformas correspondentes. Depois, executa uma consulta na base de dados e exibe os resultados linha por linha no terminal.

```
# INNER JOIN das três tabelas ao mesmo tempo
query = """
    SELECT j.nome AS Jogo, g.nome AS Genero, p.nome AS Plataforma
    FROM jogos j
    INNER JOIN generos g ON j.genero_id = g.id
    INNER JOIN plataformas p ON j.plataforma_id = p.id;
"""

cursor.execute(query)

# Obter e exibir os resultados
resultados = cursor.fetchall()
for linha in resultados:
    print(linha)  # Cada linha mostra (Jogo, Genero, Plataforma)
```

Figura 8 - INNER JOIN entre as três tabelas

```
[Running] python -u "c:\Users\ba2434\Documents\Projeto_Final\NewGames\master\src\select\select.py"
('Valorant', 'FPS', 'PC')
('MaiMai', 'Ritmo', 'Arcade')
('Hogwarts Legacy', 'Quebra-cabe�a', 'PlayStation')
('Minecraft', 'Sobreviv�ncia e A��o', 'PC')
('Hollow Knight', 'Sobreviv�ncia e A��o', 'Xbox')
('Grand Summoners', 'RPG', 'Mobile')

[Done] exited with code=0 in 0.06 seconds
```

Figura 9 - Exibição dos Resultados no Terminal

Exclusão de um Jogo Específico (*Delete*)

Tem como objetivo permitir que o utilizador remova um jogo da base de dados. Para isso, o programa solicita ao utilizador o nome do jogo a ser apagado, executa a remoção na base de dados e confirma a operação.

```
# Pedir ao utilizador o nome do jogo a ser apagado
nome_jogo = input("Digite o nome do jogo a ser apagado: ")
cursor.execute("DELETE FROM jogos WHERE nome = ?;", (nome_jogo,)
conexao.commit()  # Guardar as alterações
print(f"Jogo '{nome_jogo}' apagado com sucesso.")
```

Figura 10 - Delete da Tabela Jogos

Exclusão de uma Tabela Completa (*Drop*)

O código solicita ao utilizador o nome de uma tabela para apagar, verifica se ela está na lista de tabelas permitidas e, se for válida, conecta-se a base de dados *SQLite* e executa o comando *DROP TABLE IF EXISTS* para removê-la.

```
# Solicitar ao utilizador o nome da tabela a ser apagada
tabela = input("Digite o nome da tabela que deseja apagar: ")

# Verificar se o nome da tabela é válido
tabelas_validas = ['jogos', 'genero_id', ' plataforma_id']
if tabela not in tabelas_validas:
    print(f"A tabela '{tabela}' não é válida ou não existe.")
else:
    # Conectar ao banco de dados
    with sqlite3.connect('C:\\Users\\ba2434\\Documents\\Projeto_
        cursor = conexao.cursor()

        # Apagar a tabela especificada
        cursor.execute(f"DROP TABLE IF EXISTS {tabela};")

        conexao.commit()  # Salvar as alterações

    print(f"Tabela '{tabela}' apagada com sucesso.")
```

Figura 11 - Drop entre as três tabelas

Conclusão

O código oferece uma implementação funcional e prática para a gestão de uma base de dados *SQLite*, permitindo que o utilizador crie, consulte, atualize e remova jogos, géneros e plataformas *(CRUD)*. Ele segue os princípios básicos de *SQL* e utiliza práticas padrão de manipulação de dados, como *INNER JOIN, UPDATE, DELETE, e DROP TABLE*. A base de dados é composta por três tabelas inter-relacionadas: **jogos, generos e plataformas.** Cada uma destas tabelas tem a sua própria funcionalidade, e a integridade referencial é assegurada através de chaves estrangeiras.

Referências

APA Style Headings. https://apastyle.apa.org/style-grammar-guidelines/paper-format/