**Python DB API**

A primeira etapa para trabalhar com um banco de dados é estabelecer uma conexão. Para realizar essa conexão com o Python DB API, fazemos (utilizaremos de exemplo o SGBD sqlite):

import sqlite3

con = sqlite3.connect(‘meu\_banco\_de\_dados.db’)

Para executar comandos SQL e buscar resultados de consultas SQL, nós precisamos usar um cursor de banco de dados. Para isso, faremos:

cur = con.cursor()

**Criação de Tabelas:**

Para criarmos tabelas, utilizaremos o nosso cursor e o comando execute da seguinte maneira:

cur.execute(“CREATE TABLE nome\_tabela(coluna\_1, coluna\_2, ...)")

**Inserindo Registros:**

Inserir registros em um banco de dados é uma operação comum. Com a Python DB API, usamos a operação INSERT do SQL da seguinte maneira:

data = (4, ‘abcd’)

cur.execute(‘INSERT INTO nome\_tabela() VALUES (?, ?)’, data)

con.commit()

**Atualizando Registros:**

A operação UPDATE do SQL é usada para modificar registros existentes. É importante ser específico ao usar o UPDATE para evitar alterações indesejadas.

data = (‘abcde’, 4)

cur.execute(‘UPDATE nome\_tabela SET name = ? WHERE id = ?’, data)

con.commit()

**Removendo Registros:**

A operação DELETE do SQL é usada para remover registros. Novamente, precisamos ser específicos ao usar o DELETE para evitar remover mais registros do que o planejado.

id = 4

cur.execute(‘DELETE FROM nome\_tabela WHERE id = ?’, id)

con.commit()

**Operações em Lote:**

Operações em lote são úteis quando precisamos inserir vários registros de uma vez. Com o Python DB API, podemos utilizar o método executemany() para isso.

data = [(5, “abcd”), (6, “abcde”), (7, “abcdef”)]

cur.executemany(‘INSERT INTO minha\_tabela VALUES(?, ?)’, data)

con.commit()

**Consultando Registros:**

O método fetchone() pode ser usado para recuperar um único registro de resultado. Ele retorna o próximo registro na lista de resultados ou ‘None’ se não houver mais resultados.

cur.execute(‘SELECT \* FROM minha\_tabela WHERE id = 1’)

result = cur.fetchone()

print(result)

O método fetchall() pode ser usado para recuperar todos os registros de resultados de uma vez. Ele retorna uma lista de registros ou uma lista vazia se não houver mais resultados.

cur.execute(‘SELECT \* FROM minha\_tabela ’)

results = cur.fetchall()

for row in results:

print(row)

Trabalhando com resultados de consulta:

Os resultados de consulta são retornados como tuplas por padrão. Se a tupla não atender as nossas necessidades podemos usar a classe ‘sqlite3.Row’ ou uma ‘row\_factory’ customizada.

cur.row\_factory = sqlite3.Row

cur.execute(‘SELECT \* FROM minha\_tabela WHERE id = 1’)

result = cur.fetchone()

print(dict(result))

**Gerenciando Transações:**

A DB API também nos permite gerenciar transações, o que é crucial para manter a integridade dos dados. Isso é importante, pois você pode realizar todo o seu trabalho dentro de uma transação e commita-lo. Porém, deu um erro e você quer que todo o seu trabalho não seja salvo e volte para o estado anterior do banco. Dessa forma, a transação toda é ignorada e não é persistida e o estado volta a ser o anterior antes da transação.

try:

cur.execute(‘INSERT INTO minha\_tabela VALUES(?, ?)’, (1, “abc”))

con.commit()

except Exception as e:

print(f”Ocorreu um erro: {e}”)

con.rollback()