# Aplicação de banco de dados com Python

Você sabia que seu material didático é interativo e multimídia? Isso significa que você pode interagir com o conteúdo de diversas formas, a qualquer hora e lugar. Na versão impressa, porém, alguns conteúdos interativos ficam desabilitados. Por essa razão, fique atento: sempre que possível, opte pela versão digital. Bons estudos!

Grande parte dos softwares que são desenvolvidos (senão todos) acessam algum tipo de mecanismo para armazenar dados. A persistência dos dados pode ser feita em arquivo, em um banco de dados relacional ou em um banco de dados NoSQL.

A teoria base dos bancos de dados relacional existe desde a década de 1970. Nessa abordagem os dados são persistidos em uma estrutura bidimensional, chamada de relação (que é uma tabela) e baseada na teoria dos conjuntos pertencentes à matemática.

Cada unidade de dados é conhecida como *coluna*, ao passo que cada unidade do grupo é conhecida como linha, *tupla* ou *registro*.

Fonte: Shutterstock.

## A linguagem SQL

Para se comunicar com um banco de dados relacional, existe uma linguagem específica conhecida como SQL, que significa Structured Query Language ou, traduzindo, linguagem de consulta estruturada. As instruções da linguagem SQL são divididas em três grupos: DDL; DML; DCL (ROCKOFF, 2016).


**DDL** é um acrônimo para data *definition language* (linguagem de definição de dados). Fazem parte desse grupo as instruções destinadas a **criar**, **deletar** e **modificar** banco de dados e tabelas. Nesse módulo, vão aparecer comandos como CREATE, ALTER e DROP.


**DML** é um acrônimo para data *manipulation language* (linguagem de manipulação de dados). Fazem parte deste grupo as instruções destinadas a **recuperar, atualizar, adicionar** ou **excluir dados** em um banco de dados. Nesse módulo vão aparecer comandos como INSERT, UPDATE e DELETE.

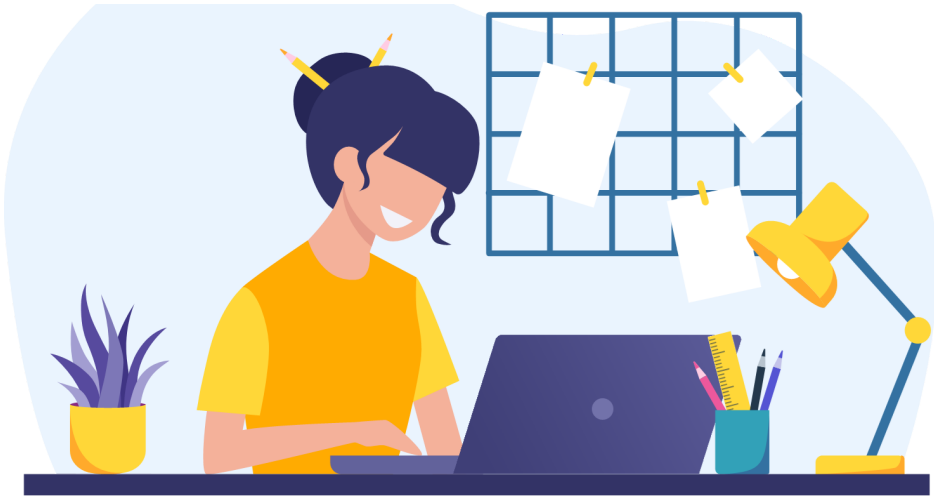

**DCL** é um acrônimo para data *control language* (linguagem de controle de dados). Fazem parte deste grupo as instruções destinadas a **manter a segurança adequada para o banco de dados**. Nesse módulo vão aparecer comandos como GRANT e REVOKE.

## Banco de dados SQLite

O SQLite é uma biblioteca em linguagem C, que implementa um mecanismo de banco de dados SQL

pequeno, rápido, independente, de alta confiabilidade

e completo. Por ser leve e não precisar da instalação de um servidor é uma ótima opção para diversos cenários.



Fonte: Shutterstock.

> O SQLite lê e grava diretamente em arquivos de disco, ou seja, um banco de dados SQL completo com várias tabelas, índices, triggers e
>
> visualizações está contido em um único arquivo de disco.

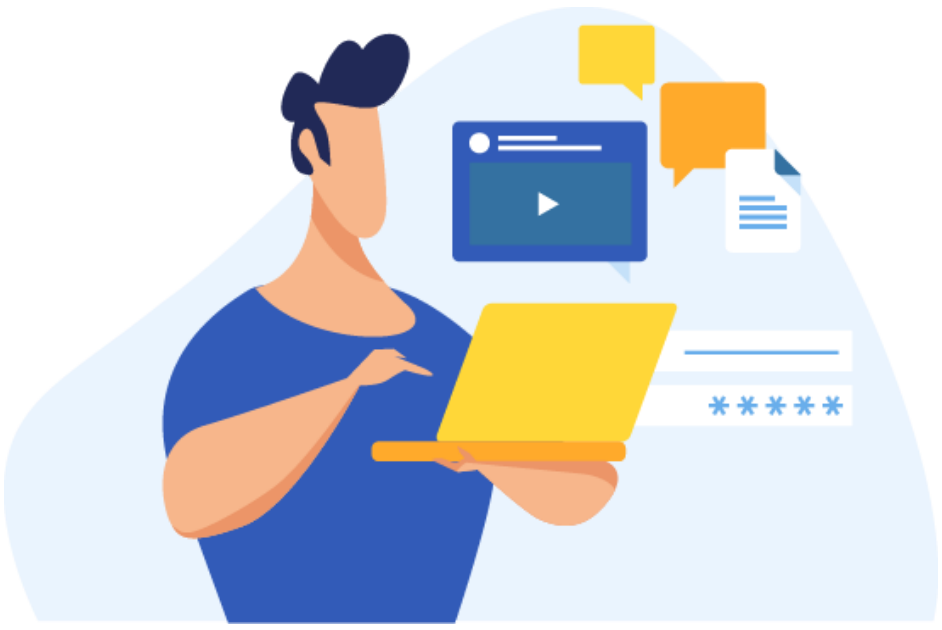
O interpretador Python possui o módulo built-in **sqlite3**, que permite utilizar o mecanismo de banco de dados SQLite. Com esse módulo, é bastante simples criar um banco de dados SQLite e fazer a conexão.

A seguir, criamos um banco de dados chamado aulaDB. Esse comando criará um arquivo chamado aulaDB com extensão db, que o identifica como um arquivo de banco.

```
1  import sqlite3
2
3  conn = sqlite3.connect('aulaDB.db')
```

## O CRUD no banco de dados SQLite

Quando o assunto é banco de dados, um termo muito comum é o CRUD, um acrônimo para as quatro operações de DML que podemos fazer em uma tabela no banco de dados – podemos inserir informações (**create**), ler (**read**), atualizar (**update**) e apagar (**delete**).



Fonte: Shutterstock.

Os passos necessários para efetuar uma das operações do CRUD são sempre os mesmos:

1. Estabelecer a conexão com um banco
2. Criar um cursor e executar o comando
3. Gravar a operação
4. Fechar o cursor e a conexão

## Aplicação do CRUD no banco de dados SQLite

Agora que já conhecemos o CRUD vamos inserir as informações para cada uma de suas operações:

### » CREATE

A variável *conn* guarda a instância de conexão com o banco de dados. Agora é preciso criar um cursor para executar as instruções e, por fim, gravar as alterações com o método *commit()*.

```
cursor = conn.cursor()
cursor.execute("""
INSERT INTO fornecedor (nome_fornecedor, cnpj, cidade, estado, cep, data_cadastro)
VALUES ('Empresa A', '11.111.111/1111-11', 'São Paulo', 'SP', '11111-111', '2020-01-01')
""")
conn.commit()
```

### » READ

Para ler os dados em uma tabela, também precisamos estabelecer uma conexão e criar um objeto cursor para executar a instrução de seleção. Ao executar a seleção, podemos usar o método *fetchall()*, para capturar todas as linhas mediante uma lista de tuplas.

```
cursor.execute("SELECT * FROM fornecedor")
resultado = cursor.fetchall()
for linha in resultado:
    print(linha)
```

### » UPDATE

Ao inserir um registro no banco, pode ser necessário alterar o valor de uma coluna, o que pode ser feito por meio da instrução SQL UPDATE.

```
cursor.execute("UPDATE fornecedor SET cidade = 'Campinas' WHERE id_fornecedor = 5")
conn.commit()
```

### » DELETE

Ao inserir um registro no banco, pode ser necessário removê-lo no futuro, o que pode ser feito por meio da instrução SQL DELETE.

```
cursor.execute("DELETE FROM fornecedor WHERE id_fornecedor = 2")
conn.commit()
```

## Pesquise mais

No Capítulo 8 (*Python 3 com Banco de dados SQLite*) do livro a seguir indicado, você encontrará a explicação sobre o software SQLite Studio (mais especificamente entre as páginas 193 e 197), que permite inspecionar de forma visual um banco de dados do SQLite

BANIN, S. L. **Python 3** - Conceitos e aplicações: uma abordagem didática. São Paulo: Érica, 2018

Para visualizar o vídeo, acesse seu material digital.