Banco de Dados

Introdução ao SQLite – Aula 1

Banco de Dados

Sobre o SQLite

O *SQLite* é um gerenciador de banco de dados leve e completo, muito utilizado e presente mesmo em telefones celulares. Uma de suas principais características é não precisar de um servidor dedicado, sendo capaz de iniciar à partir de seu programa. Um banco SQLite, basicamente é contido em um único arquivo não necessitando de configuração ou administração.

Nesta seção, nós veremos os comandos mais importantes e as etapas necessárias para utilizar o *SQLite*.

Banco de Dados

Introdução ao SQLite

A primeira coisa a fazer, é importar a biblioteca para conexão ao *SQLite*:

```
import sqlite3
```

Depois do ***import***, várias funções e objetos que acessam o banco de dados se tornam disponíveis ao seu programa. Antes de continuarmos, vamos criar o banco de dados:

```
conexao = sqlite3.connect("agenda.db")
```

Em seguida criamos o cursor:

```
cursor = conexao.cursor()
```

Cursores são objetos utilizados para enviar comandos e receber resultados do banco de dados. Um cursor é criado para uma conexão, chamando-se o método ***cursor()*** . Uma vez que obtivermos um cursor, nós podemos trabalhar com o banco de dados.

Banco de Dados

Introdução ao SQLite

Em nosso primeiro comando vamos criar uma tabela chamada “contatos” para guardar nomes e telefones.

```
cursor.execute("""
               create table contatos(
               id integer primary key,
               nome varchar(100),
               telefone varchar(15))
               """)
```

Utilizamos o método ***execute*** de nosso cursor para enviar o comando ao banco de dados.

Com a tabela criada, podemos começar a introduzir nossos dados. Vejamos o comando SQL usado para inserir um registro:

insert into contatos (nome, telefone) values (?, ?)

Banco de Dados

Introdução ao SQLite

```
cursor.execute(“””
   insert into contatos (nome, telefone)
      values(?, ?)
      ”””, (“Evaldo”, “1234-5678”))
```

Utilizamos o método ***execute*** para executar o comando ***insert***, passando os dados logo após o comando. No exemplo “Evaldo” e “1234-5678” irão substituir o primeiro e o segundo caractere de interrogação quando o comando for executado.

Banco de Dados

Introdução ao SQLite

Em seguida utilizamos o comando ***commit*** para modificar o banco de dados.

```
conexao.commit()
```

Antes de terminarmos o programa, fechamos o cursor e a conexão com o banco de dados, respectivamente.

```
cursor.close()
conexao.close()
```

Execute o programa e verifique se o arquivo agenda.db foi criado. Se você executar o programa uma segunda vez, um erro será gerado com a mensagem:

Tracerback (most recent call last):

File "exemplo01.py", line 8, in <module>

"")

sqlite3.OperationalError: table contatos already exists

Banco de Dados

Introdução ao SQLite

Este erro acontece porque a tabela contatos já existe. Se você executar o programa novamente, apague o arquivo contatos.db. Lembre-se de que todos os dados estão nesse arquivo e, ao apagá-lo, tudo é perdido.

Podemos usar o módulo os para apagar o arquivo. Coloque o código a seguir e execute o programa novamente.

```
...
import os

if os.path.exists("agenda.db"):
    os.remove('agenda.db’)
...
```

Banco de Dados

Introdução ao SQLite

Podemos também validar se a tabela existe antes de criá-la. Remova a parte do código que apaga o arquivo do banco de dados (do passo anterior) e execute o sistema novamente com o código da criação da tabela abaixo. Ao executar o programa novamente, não dará erro porque ele só vai criar a tabela caso não exista.

```
cursor.execute("""
                create table if not exists contatos(
                id integer primary key,
                nome varchar(100),
                telefone varchar(15))
                """)
```

FIM