



Regis da Silva



Publicado em:

seg 16 junho 2014

[←Home](#)

Gerenciando banco de dados SQLite3 com Python - Parte 1

// Tags [Python](#) [Banco de dados](#)

Eu separei este post em duas partes: a **Parte 1** é bem elementar e objetiva, visando apresentar o básico sobre a realização do CRUD num banco de dados SQLite3 em Python usando o terminal.

A [parte 2](#), num nível intermediário, usa classes e métodos mais elaborados para gerenciar o CRUD, e algumas coisinhas a mais.

Nota: Para entender o uso de classes e métodos leia o post [Introdução a Classes e Métodos em Python](#). E para entender os comandos SQL e a manipulação de registros no SQLite3 leia [Guia rápido de comandos SQLite3](#).

Para os exemplos considere a tabela `clientes` e seus campos:

Campo	Tipo	Requerido
id	inteiro	sim
nome	texto	sim
idade	inteiro	não
cpf	texto (11)	sim
email	texto	sim
fone	texto	não
cidade	texto	não
uf	texto (2)	sim
criado_em	data	sim
bloqueado	booleano	não

Obs: O campo `bloqueado` nós vamos inserir depois com o comando `ALTER TABLE`.

Veja os exemplos em [github](#)

veja os exemplos em [github](#).

Como mencionado antes, esta parte será **básica e objetiva**. A intenção é realizar o CRUD da forma mais simples e objetiva possível.

PS: Considere a sintaxe para Python 3.

[Conectando e desconectando do banco](#)

[Criando um banco de dados](#)

[Criando uma tabela](#)

[Create - Inserindo um registro com comando SQL](#)

[Inserindo n registros com uma tupla de dados](#)

[Inserindo um registro com parâmetros de entrada definido pelo usuário](#)

[Read - Lendo os dados](#)

[Update - Alterando os dados](#)

[Delete - Deletando os dados](#)

[Adicionando uma nova coluna](#)

[Lendo as informações do banco de dados](#)

[Fazendo backup do banco de dados \(exportando dados\)](#)

[Recuperando o banco de dados \(importando dados\)](#)

[Exemplos](#)

[Referências](#)

Conectando e desconectando do banco

Podemos criar o banco de dados de duas formas: na **memória RAM**

```
# conectando...
conn = sqlite3.connect(':memory:')
```

ou persistindo em um **banco de dados**, vamos usar sempre este caso.

```
# conectando...
conn = sqlite3.connect('clientes.db')
```

Uma sintaxe mínima para se conectar a um banco de dados é:

```
# connect_db.py
# 01_create_db.py
import sqlite3

conn = sqlite3.connect('clientes.db')
conn.close()
```

O último método desconecta do banco.

Considere um arquivo para cada operação.

Nota: Os arquivos estão numerados apenas para sugerir uma sequência.

Criando um banco de dados

O código para criar um banco de dados é o mesmo mencionado anteriormente.

Para rodar este programa abra o **terminal** e digite:

```
$ python3 01_create_db.py
$ ls *.db
```

Digitando `ls` você verá que o banco foi criado.

Criando uma tabela

Para criar uma tabela no banco de dados usamos dois métodos fundamentais:

- **cursor**: é um interador que permite navegar e manipular os registros do bd.
- **execute**: lê e executa comandos SQL puro diretamente no bd.

```
# 02_create_schema.py
import sqlite3

# conectando...
conn = sqlite3.connect('clientes.db')
# definindo um cursor
cursor = conn.cursor()

# criando a tabela (schema)
cursor.execute("""
CREATE TABLE clientes (
    id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
    nome TEXT NOT NULL,
    idade INTEGER,
    cpf VARCHAR(11) NOT NULL,
    email TEXT NOT NULL,
    fone TEXT,
    cidade TEXT,
    uf VARCHAR(2) NOT NULL,
    criado_em DATE NOT NULL
);
""")

print('Tabela criada com sucesso.')
# desconectando...
conn.close()
```

Para executar digite no terminal:

```
$ python3 02_create_schema.py
$ sqlite3 clientes.db '.tables'
$ sqlite3 clientes.db 'PRAGMA table_info(clientes)'
```

Digitando `sqlite3 clientes.db '.tables'` você verá que a tabela foi criada.

E o comando `sqlite3 clientes.db 'PRAGMA table_info(clientes)'` retorna os campos da tabela.

Nota: A única diferença, caso você use *Python 2* é no print, onde você deve tirar os parênteses. E no início do arquivo é recomendável que se defina a codificação `utf-8`, que no caso do Python 3 já é padrão.

[Topo](#)[OPEN CHAT](#)

```
# 02_create_schema.py
# -*- coding: utf-8 -*-
# usando Python 2
import sqlite3
...
print 'Tabela criada com sucesso.'
```

Agora vamos fazer o CRUD. Começando com a letra

Create - Inserindo um registro com comando SQL

A única novidade aqui é o método **commit()**. É ele que grava de fato as alterações na tabela. *Lembrando que uma tabela é alterada com as instruções SQL ``INSERT, UPDATE`` e ``DELETE``.*

```
# 03_create_data_sql.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

# inserindo dados na tabela
cursor.execute("""
INSERT INTO clientes (nome, idade, cpf, email, fone, cidade, uf, criado_em)
VALUES ('Regis', 35, '00000000000', 'regis@email.com', '11-98765-4321', 'Sao Paulo', 'SP', '2014-06-08')
""")

cursor.execute("""
INSERT INTO clientes (nome, idade, cpf, email, fone, cidade, uf, criado_em)
VALUES ('Aloisio', 87, '11111111111', 'aloisio@email.com', '98765-4322', 'Porto Alegre', 'RS', '2014-06-09')
""")

cursor.execute("""
INSERT INTO clientes (nome, idade, cpf, email, fone, cidade, uf, criado_em)
VALUES ('Bruna', 21, '22222222222', 'bruna@email.com', '21-98765-4323', 'Rio de Janeiro', 'RJ', '2014-06-09')
""")

cursor.execute("""
INSERT INTO clientes (nome, idade, cpf, email, fone, cidade, uf, criado_em)
VALUES ('Matheus', 19, '33333333333', 'matheus@email.com', '11-98765-4324', 'Campinas', 'SP', '2014-06-08')
""")

# gravando no bd
conn.commit()

print('Dados inseridos com sucesso.')

conn.close()
```

Para executar digite no terminal:

```
$ python3 03_create_data_sql.py
```

Inserindo n registros com uma tupla de dados

Usando uma *lista* podemos inserir vários registros de uma vez, e o método `executemany` faz essa ação.

```
# 04_create_data_nrecords.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

# criando uma lista de dados
lista = [(
    'Fabio', 23, '44444444444', 'fabio@email.com', '1234-5678', 'Belo Horizonte', 'MG', '2014-06-09'),
    ('Joao', 21, '55555555555', 'joao@email.com',
    '11-98765-4321', 'Campinas', 'SP', '2014-06-08')]
```

[Topo](#)[OPEN CHAT](#)

```

11-1234-5600', 'São Paulo', 'SP', '2014-06-09'),
('Xavier', 24, '666666666666', 'xavier@email.com', '12-1234-5601', 'Campinas', 'SP', '2014-06-10')]

# inserindo dados na tabela
cursor.executemany("""
INSERT INTO clientes (nome, idade, cpf, email, fone, cidade, uf, criado_em)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
""", lista)

conn.commit()

print('Dados inseridos com sucesso.')

conn.close()

```

Observe o uso de ? isto significa que no lugar de cada ? entrará os valores da lista na sua posição respectiva. É o que nós chamamos de *parâmetros de entrada*.

Para executar digite no terminal:

```
$ python3 04_create_data_nrecords.py
```

Inserindo um registro com parâmetros de entrada definido pelo usuário

Neste exemplo usaremos parâmetros de entrada, que deverá ser digitado pelo usuário. Esta é a forma mais desejável de entrada de dados porque o usuário pode digitar os dados em tempo de execução.

```

# 05_create_data_param.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

# solicitando os dados ao usuário
p_nome = input('Nome: ')
p_idade = input('Idade: ')
p_cpf = input('CPF: ')
p_email = input('Email: ')
p_fone = input('Fone: ')
p_cidade = input('Cidade: ')
p_uf = input('UF: ')
p_criado_em = input('Criado em (yyyy-mm-dd): ')

# inserindo dados na tabela
cursor.execute("""
INSERT INTO clientes (nome, idade, cpf, email, fone, cidade, uf, criado_em)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
""", (p_nome, p_idade, p_cpf, p_email, p_fone, p_cidade, p_uf, p_criado_em))

conn.commit()

print('Dados inseridos com sucesso.')

conn.close()

```

Nota: Caso use *Python 2* use o método `raw_input()` em

```

# python 2
p_nome = raw_input('Nome: ')
...
print 'Dados inseridos com sucesso.'

```

Para executar digite no terminal:

```
$ python3 05_create_data_param.py
```

Topo

OPEN CHAT

Veja a interação do programa:

veja a interação do programa.

```
Nome: Regis
Idade: 35
CPF: 30020030011
Email: regis@email.com
Fone: 11 9537-0000
Cidade: Sao Paulo
UF: SP
Criado em (yyyy-mm-dd): 2014-06-15
Dados inseridos com sucesso.
```

Read - Lendo os dados

Aqui nós usamos o famoso SELECT. O método fetchall() retorna o resultado do SELECT.

```
# 06_read_data.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

# lendo os dados
cursor.execute("""
SELECT * FROM clientes;
""")

for linha in cursor.fetchall():
    print(linha)

conn.close()
```

Para executar digite no terminal:

```
$ python3 06_read_data.py
```

Eis o resultado:

```
(1, 'Regis', 35, '00000000000', 'regis@email.com', '11-98765-4321', 'Sao Paulo', 'SP', '2014-06-08')
(2, 'Aloisio', 87, '11111111111', 'aloisio@email.com', '98765-4322', 'Porto Alegre', 'RS', '2014-06-09')
(3, 'Bruna', 21, '22222222222', 'bruna@email.com', '21-98765-4323', 'Rio de Janeiro', 'RJ', '2014-06-09')
(4, 'Matheus', 19, '33333333333', 'matheus@email.com', '11-98765-4324', 'Campinas', 'SP', '2014-06-08')
(5, 'Fabio', 23, '44444444444', 'fabio@email.com', '1234-5678', 'Belo Horizonte', 'MG', '2014-06-09')
(6, 'Joao', 21, '55555555555', 'joao@email.com', '11-1234-5600', 'Sao Paulo', 'SP', '2014-06-09')
(7, 'Xavier', 24, '66666666666', 'xavier@email.com', '12-1234-5601', 'Campinas', 'SP', '2014-06-10')
(8, 'Regis', 35, '30020030011', 'regis@email.com', '11 9750-0000', 'Sao Paulo', 'SP', '2014-06-15')
```

Update - Alterando os dados

Observe o uso das variáveis id_cliente onde definimos o id a ser alterado, novo_fone e novo_criado_em usados como parâmetro para alterar os dados. Neste caso, salvamos as alterações com o método commit().

```
# 07_update_data.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

id_cliente = 1
novo_fone = '11-1000-2014'
novo_criado_em = '2014-06-11'

# alterando os dados da tabela
cursor.execute("""
```

[Topo](#)[OPEN CHAT](#)

```
UPDATE clientes
SET fone = ?, criado_em = ?
WHERE id = ?
""", (novo_fone, novo_criado_em, id_cliente))

conn.commit()

print('Dados atualizados com sucesso.')

conn.close()
```

Para executar digite no terminal:

```
$ python3 07_update_data.py
```

Delete - Deletando os dados

Vamos excluir um registro pelo seu id.

```
# 08_delete_data.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

id_cliente = 8

# excluindo um registro da tabela
cursor.execute("""
DELETE FROM clientes
WHERE id = ?
""", (id_cliente,))

conn.commit()

print('Registro excluido com sucesso.')

conn.close()
```

Para executar digite no terminal:

```
$ python3 08_delete_data.py
```

Adicionando uma nova coluna

Para inserir uma nova coluna na tabela usamos o comando SQL ALTER TABLE.

```
# 09_alter_table.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()

# adicionando uma nova coluna na tabela clientes
cursor.execute("""
ALTER TABLE clientes
ADD COLUMN bloqueado BOOLEAN;
""")

conn.commit()

print('Novo campo adicionado com sucesso.')

conn.close()
```

[Topo](#)[OPEN CHAT](#)

Para executar digite no terminal:

```
$ python3 09_alter_table.py
```

Lendo as informações do banco de dados

Para ler as informações da tabela usamos o comando PRAGMA.

Para listar as tabelas do banco usamos o comando SELECT name FROM sqlite_master

Para ler o schema da tabela usamos o comando SELECT sql FROM sqlite_master

```
# 10_view_table_info.py
import sqlite3

conn = sqlite3.connect('clientes.db')
cursor = conn.cursor()
nome_tabela = 'clientes'

# obtendo informações da tabela
cursor.execute('PRAGMA table_info({})'.format(nome_tabela))

colunas = [tupla[1] for tupla in cursor.fetchall()]
print('Colunas:', colunas)

# ou
# for coluna in colunas:
#     print(coluna)

# listando as tabelas do bd
cursor.execute("""
SELECT name FROM sqlite_master WHERE type='table' ORDER BY name
""")

print('Tabelas:')
for tabela in cursor.fetchall():
    print("%s" % (tabela))

# obtendo o schema da tabela
cursor.execute("""
SELECT sql FROM sqlite_master WHERE type='table' AND name=?
""", (nome_tabela,))

print('Schema:')
for schema in cursor.fetchall():
    print("%s" % (schema))

conn.close()
```

Para executar digite no terminal:

```
$ python3 10_view_table_info.py
```

Eis o resultado:

```
Colunas: ['id', 'nome', 'idade', 'cpf', 'email', 'fone', 'cidade', 'uf', 'criado_em', 'bloqueado']
Tabelas:
clientes
sqlite_sequence
Schema:
CREATE TABLE clientes (
  id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
  nome TEXT NOT NULL,
  idade INTEGER,
  cpf    VARCHAR(11) NOT NULL,
  email TEXT NOT NULL,
  fone TEXT,
```

[Topo](#)[OPEN CHAT](#)


```
        cidade TEXT,\n        uf VARCHAR(2) NOT NULL,\n        criado_em DATE NOT NULL\n    , bloqueado BOOLEAN)
```

Fazendo backup do banco de dados (exportando dados)

Talvez seja este o item mais importante: **backup**. Observe o uso da biblioteca **io** que salva os dados num arquivo externo através do método `write`, e o método `iterdump()` que exporta a estrutura e dados da tabela para o arquivo externo.

```
# 11_backup.py\nimport sqlite3\nimport io\n\nconn = sqlite3.connect('clientes.db')\n\nwith io.open('clientes_dump.sql', 'w') as f:\n    for linha in conn.iterdump():\n        f.write('%s\\n' % linha)\n\nprint('Backup realizado com sucesso.')\nprint('Salvo como clientes_dump.sql')\n\nconn.close()
```

Para executar digite no terminal:

```
$ python3 11_backup.py\n$ cat clientes_dump.sql
```

Com o comando `cat` você poderá ler a estrutura da tabela salva.

Recuperando o banco de dados (importando dados)

Criaremos um novo banco de dados e iremos reconstruir a tabela e os dados com o arquivo `clientes_dump.sql`. O método `read()` lê o conteúdo do arquivo `clientes_dump.sql` e o método `executescript()` executa as instruções SQL escritas neste arquivo.

```
# 12_read_sql.py\nimport sqlite3\nimport io\n\nconn = sqlite3.connect('clientes_recuperado.db')\ncursor = conn.cursor()\n\nf = io.open('clientes_dump.sql', 'r')\nsql = f.read()\ncursor.executescript(sql)\n\nprint('Banco de dados recuperado com sucesso.')\nprint('Salvo como clientes_recuperado.db')\n\nconn.close()
```

Para executar digite no terminal:

```
$ python3 12_read_sql.py\nBanco de dados recuperado com sucesso.\nSalvo como clientes_recuperado.db\n$ sqlite3 clientes_recuperado.db 'SELECT * FROM clientes;'
```

Com o último comando você verá que os dados estão lá. São e salvo!!!

[Topo](#)[OPEN CHAT](#)

Leia a continuação deste artigo em [Gerenciando banco de dados SQLite3 com Python - Parte 2](#).

Exemplos

Veja os exemplos em [github](#).

Referências

[sqlite3 Embedded Relational Database](#)

[Lets Talk to a SQLite Database with Python](#)

[Advanced SQLite Usage in Python](#)

[Python A Simple Step by Step SQLite Tutorial](#)

[Python docs, SQLite, Connection Objects](#)



"Gerenciando banco de dados SQLite3 com Python - Parte 1" de "Regis da Silva" está licenciado com uma Licença Creative Commons - Atribuição-NãoComercial-SemDerivações 4.0 Internacional.

Compartilhar 61

Tweetar

TAMBÉM EM PYTHONCLUB BLOG

Bot telegram mais web scraping - ...

6 anos atrás • 4 comentários
PythonClub,

Relatórios de testes com Coveralls

6 anos atrás • 2 comentários
PythonClub,

Tutorial Django 2.2

3 anos atrás • 4 comentários
PythonClub,

Instalando o versão 3.7.0 :

5 anos atrás • 2 con
PythonClub,

Sponsored

Kit com 2 Mesa de cabeceira Retrô Dream Plus - Branco - Lojas rpm

Americanas.com

Como você reagiria se o Brasil fosse atacado? Esse jogo simula conflitos geopolíticos

Conflict of Nations

Família pensava que adotara um 'cachorro', mas quando o veterinário o vê, chama a polícia.

PDFWonder

Topo

OPEN CHAT

Você se lembra das gêmeas mais lindas do mundo? Olhe para elas hoje

Vida Brilhante

Lacoste exclusiva compre 1 e leve 5

Hoofit

Polo Lacoste cinco unidades por R\$: 129,90

Cinco camisas deluxe em liquidação desk

29 Comentários Pythonclub Blog  Política de Privacidade

 Entrar ▾

 Favorite 9  Tweet  Compartilhar Ordenar por Mais votados ▾



Participe da discussão...

FAZER LOGIN COM

OU REGISTRE-SE NO DISQUS 

Nome



Michel Silveira • 2 years ago

Muito bom o material!!
Tô usando bastante nos estudos!!! :-)

Muito Obrigado!! :-)

2 ^ | ▾ • Responder • Compartilhar >



Hatatori Nitsukawa • 3 years ago

essas informações ficam guardadas aonde?

1 ^ | ▾ • Responder • Compartilhar >



Fernanda Araújo ➔ Hatatori Nitsukawa • 3 years ago

Você precisa criar o arquivo, .db pra depois fazer os outros passos, eles serão armazenados no arquivo database.db que você criou no inicio do do artigo.

^ | ▾ • Responder • Compartilhar >



MrMaicke • 4 months ago

MUITO BOM

^ | ▾ • Responder • Compartilhar >



Gabriel Santos • 8 months ago

Topo

OPEN CHAT



Foi muito útil esse conteúdo. Obrigado.

^ | v • Responder • Compartilhar >



Diego Sacramento • 2 years ago

Regis é fera valeu me ajudou muito

^ | v • Responder • Compartilhar >



Watson Silva • 2 years ago

Regis,

Muito bom artigo. Parabéns!!!

Me ajudou muito!!!

^ | v • Responder • Compartilhar >



Helio Barroso • 2 years ago

Oi Regis, grandes artigos.

Voce tem como me passar alguma orientação de como usar Exceptions quando lidando com Python sqlite3 ?

Obrigado

^ | v • Responder • Compartilhar >



richellyitalo • 3 years ago

Valeu Regis! Muito obrigado.

^ | v • Responder • Compartilhar >



João Kesley Kesley • 3 years ago

Parabéns pelo ótimo artigo...

^ | v • Responder • Compartilhar >



Sérgio Barbosa Catão • 4 years ago

amigo bom dia estou usando o comando tree.focus() mas nao consigo entender muito bem como ele funciona... segue abaixo o trecho de codigo onde é usado, por favor se souber a finalidade me ajude.

```
curItem = tree.focus()
contents = (tree.item(curItem))
selecteditem = contents('values')
tree.delete(curItem)
Database()
```

^ | v • Responder • Compartilhar >



HWarley Massafera • 4 years ago

Parabéns pelo artigo, bem explicativo.

^ | v • Responder • Compartilhar >



samhk222 • 4 years ago

Fantástico, valeu

^ | v • Responder • Compartilhar >



Leo Neves • 5 years ago

Muito bom, obrigado.

^ | v • Responder • Compartilhar >

Topo

OPEN CHAT

**Cassio Amador** • 5 years ago

Excelente postagem!

^ | v • Responder • Compartilhar ›

**Gamaliel Alves** • 5 years ago

kra, muito legal esse material, tirei minhas duvidas.

^ | v • Responder • Compartilhar ›

**Ben-Hur Costa** • 5 years ago

Obrigado Regis por disponibilizar este tutorial. Sempre tive grande dificuldade de entender como eram feitas as conexões e o restante das operações, mas com teu tutorial ficou bem claro de como funcionam as coisas. Muito obrigado.

^ | v • Responder • Compartilhar ›

**Alexandre Luiz Santos** • 5 years ago

Ótima página. Parabéns pelo conteúdo excelente.

^ | v • Responder • Compartilhar ›

**Welden Lima** • 5 years ago

Muito bom! O que estava procurando. Vlw

^ | v • Responder • Compartilhar ›

**Rafael Gomes** • 5 years ago

Ha algum post de como instalar mysqldb ou o pymysql

^ | v • Responder • Compartilhar ›

**brazica** • 5 years ago

uso ate hoje. rs

^ | v • Responder • Compartilhar ›

**Marcelo F.** • 5 years ago

Muito bom, Regis; está sendo muito útil. Obrigado!!!!

^ | v • Responder • Compartilhar ›

**Andre Marcos Gelhen** • 6 years ago

Muito bom Regis.

^ | v • Responder • Compartilhar ›

**Danilo Vaz aka UNK** • 6 years ago

Regis, muito bom brother, me tirou umas dúvidas aqui que tinha, obrigado!!!!

^ | v • Responder • Compartilhar ›

**Bruno** • 6 years ago

estou com um problema e já procurei e não encontrei resposta, quando testo o app na ide do python (IDLE) o sqlite3 cria o banco e conecta normalmente, porém quando testo o app executando no python (windows) ele da esse erro, e também acontece o mesmo erro quando executo no celular

[Visualizar](#)

^ | v • Responder • Compartilhar ›

Topo

OPEN CHAT

**Dorival Cardozo** • 6 years ago

Ficou ótimo este tutorial, trabalho excelente

^ | v • Responder • Compartilhar ›

**Lucas Nicoloso** • 6 years ago

Baita Post!

^ | v • Responder • Compartilhar ›

**Walker Arantes** • 6 years ago

Sponsored

Como você reagiria se o Brasil fosse atacado? Esse jogo simula conflitos geopolíticos

Conflict of Nations

Polo Lacoste cinco unidades por R\$: 129,90

Cinco camisas deluxe em liquidação desk

Você se lembra das gêmeas mais lindas do mundo? Olhe para elas hoje

Vida Brilhante

Lacoste exclusiva compre 1 e leve 5

Hooft

Família pensava que adotara um 'cachorro', mas quando o veterinário o vê, chama a polícia.

PDFWonder

A filha de Chris Rock é provavelmente a mulher mais bonita que já existiu

Healthy George

Topo

