# Um tutorial passo a passo para SQLite em Python

## Como criar um banco de dados e inserir qualquer coisa

Criar um banco de dados no SQLite é realmente muito simples, mas o processo requer que você conheça um pouco de SQL. Aqui está o código para criar um banco de dados de gerenciamento de albuns musicais:

```
import sqlite3

conn = sqlite3.connect("mydatabase.db") # ou use :memory: para botá-lo na memória RAM

cursor = conn.cursor()

# cria uma tabela
cursor.execute("""CREATE TABLE albums
                  (title text, artist text, release_date text,
                   publisher text, media_type text)
               """)
```

Antes de mais nada, precisamos importar a biblioteca **sqlite3** e criar uma conexão com o banco de dados. Você pode passar para ele um caminho para o arquivo, o nome do arquivo ou usar simplesmente a string especial ":memory:" e criar um banco de dados na memória. No nosso caso, criamos um arquivo no disco chamado **mydatabase.db**. Em seguida, criamos um objeto cursos, que lhe permite interagir com o banco de dados e adicionar registro, entre outras coisas. Usamos aqui a sintaxe SQL para criar uma tabela chamada **albums** e contendo 5 campos de texto: title, artist, release_date, publisher e media_type. SQLite só suporta cinco tipos de dados: null, integer, real, text and blob. Vamos agora fazer o código para inserir alguns dados na tabela!

```
# insere alguns dados
cursor.execute("INSERT INTO albums VALUES ('Glow', 'Andy Hunter', '7/24/2012', 'Xplore
Records', 'MP3')")

# salva dados no banco
conn.commit()

# insere múltiplos registros de uma só vez usando o método "?", que é mais seguro
albums = [('Exodus', 'Andy Hunter', '7/9/2002', 'Sparrow Records', 'CD'),
        ('Until We Have Faces', 'Red', '2/1/2011', 'Essential Records', 'CD'),
        ('The End is Where We Begin', 'Thousand Foot Krutch', '4/17/2012', 'TFKmusic', 'CD'),
        ('The Good Life', 'Trip Lee', '4/10/2012', 'Reach Records', 'CD')]
cursor.executemany("INSERT INTO albums VALUES (?,?,?,?,?)", albums)
conn.commit()
```

Aqui usamos o comando INSERT INTO, do SQL, para inserir um registro dentro do banco de dados. Note que cada item tem que estar envolto entre aspas simples. Isso pode ficar complicado quando você precisar inserir strings já com aspas simples incluídas. De qualquer forma, para salvar o registro no banco, nós chamamos o **commit**. O código que vem em seguida mostra como adicionar múltiplos registros de uma só vez usando o método **executemany** do cursor. Note que usamos a interrogação (?) ao invés da substituição de string (%s) para inserir valores. Usar a substituição de strings NÃO é seguro e não deve ser usado, pois pode permitir ataques de SQL injection. O método da interrogação é muito melhor e usando o SQLAlchemy é sempre melhor, pois ele faz todo o tratamento para você, e você não precisará se preocupar em converter aspas simples em algo que o SQLite aceite.

## Atualizando e deletando registros

Estar apto a atualizar seus registros no banco de dados é a chave para manter seus dados nos conformes. Se você não atualizar, seus dados se tornarão rapidamente obsoletos e sem utilidade. Algumas vezes você também desejará deletar linhas dos seus dados. Cobriremos esses dois tópicos nesta sessão. Primeiro, hora de atualizar!

```
import sqlite3

conn = sqlite3.connect("mydatabase.db")
cursor = conn.cursor()

sql = """
UPDATE albums
SET artist = 'John Doe'
WHERE artist = 'Andy Hunter'
"""
cursor.execute(sql)
conn.commit()
```

Aqui usamos o comando UPDATE do SQL para atualizar nossa tabela albums. Você pode usar SET para mudar um campo, neste caso mudamos o campo artista para ser “John Doe” em qualquer registro onde o campo artist era “Andy Hunter”. Não é fácil? Note que se você não confirmar as mudanças com o commit, suas mudanças não serão escritas no banco de dados e não terão efeito. O comando DELETE é simples também. Vamos a ele!

```
import sqlite3

conn = sqlite3.connect("mydatabase.db")
cursor = conn.cursor()

sql = """
DELETE FROM albums
WHERE artist = 'John Doe'
"""
```

```
cursor.execute(sql)
conn.commit()
```

Deletar chega a ser mais fácil do que atualizar. O SQL só tem 2 linhas! Neste caso, tudo o que temos que fazer é dizer pro SQLite de que tabela deletar (albums) e quais registros deletar, usando a cláusula WHERE. O comando do exemplo apaga todo e qualquer registro que tenha “John Doe” no campo artist.

## Consultas básicas do SQLite

Consultas no SQLite são basicamente as mesmas que você usa para outros bancos de dados, como o MySQL ou o Postgres. Basta você usar a sintaxe SQL normal para elaborar as consultas e executá-las com o cursor. Aqui estão alguns exemplos:

```
import sqlite3

conn = sqlite3.connect("mydatabase.db")
#conn.row_factory = sqlite3.Row
cursor = conn.cursor()

sql = "SELECT * FROM albums WHERE artist=?"
cursor.execute(sql, [("Red")])
print cursor.fetchall()  # ou use fetchone()

print "\nAqui a lista de todos os registros na tabela:\n"
for row in cursor.execute("SELECT rowid, * FROM albums ORDER BY artist"):
   print row

print "\nResultados de uma consulta com LIKE:\n"
sql = """
SELECT * FROM albums
WHERE title LIKE 'The%'"""
cursor.execute(sql)
print cursor.fetchall()
```

A primeira consulta que executamos é um **SELECT *** , o que significa que estamos buscando todos os campos, e buscamos todos os registros cujo nome do artista (campo artist) bata com o que informamos - no nosso caso, “Red”. Em seguida, executamos o SQL e usamos fetchall() para pegar todos os resultados. Você também pode usar fetchone() para pegar o apenas primeiro resultado. Você também notará que há uma sessão comentada relacionada com um misterioso **row_factory**. Se você descomentar aquela linha, os resultados serão retornados como objetos Row, que são como os dicionários do Python. Entretanto, você não pode fazer associação de item com um objeto Row.

A segunda consulta é bem parecida com a primeira, mas retorna os registros ordenados pelo nome do artista, em ordem ascendente. Esse código também demonstra como podemos percorrer o resultado executando ações. A última consulta mostra como usar o comando LIDE do SQL para buscar frases parciais.Neste caso, fizemos uma busca pela tabela por títulos que comecem com "The". O sinal de porcentagem (%) é o operador coringa.

## Pra terminar

Agora você sabe como usar o Python para criar um banco de dados SQLite. Você também pode criar, atualizar e deletar registros, bem como executar consultas em seu banco de dados. Vá em frente e comece a fazer seus próprios bancos / ou compartilhe suas experiências nos comentários!