Librería sqlite3

La librería SQLite3 es una librería que forma parte de la biblioteca estándar de Python que nos permite trabajar con base de datos SQLite.

SQLite es una librería hecha en C que provee una base de datos liviana basada en un archivo (que se guarda en el disco rígido) que no requiere un proceso separado y permite el acceso a la base de datos usando una variante no estándar del lenguaje de consultas SQL.

Algunas aplicaciones pueden usar SQLite como un almacenamiento interno. Además, es posible utilizar SQLite para armar un prototipo de una aplicación y después portar el código a una base de datos más grande como PostgreSQL u Oracle.

También la librería SQLite3 nos permite trabajar con una base de datos en memoria, muy útil para correr tests unitarios, para probar todo el sistema incluso la base de datos de manera rápida.

A continuación, veremos un ejemplo sencillo, que crea una conexión a una base de datos en memoria, obtiene un cursor y crea una tabla para almacenar monedas llamada *currency*, con las columnas ID, name y symbol. Luego ingresa dos registros en la tabla y hace un *commit* para que los cambios se efectúen en la base de datos. Por último, se hace una consulta de todos los registros de la tabla currency y se imprime en pantalla el resultado. Al terminar se cierra la conexión a la base de datos.

```
# -*- coding: utf-8 -*-
import sqlite3
```

```
# Conexión y cursor a la base de datos
conn = sqlite3.connect(':memory:')
cursor = conn.cursor()

# Creo la tabla
cursor.execute("""CREATE TABLE currency
                (ID integer primary key, name text, symbol text)""")
# Inserto datos de monedas
cursor.execute("INSERT INTO currency VALUES (1, 'Peso (ARG)', '$')")
cursor.execute("INSERT INTO currency VALUES (2, 'Dólar', 'U$S')")
# Guardo los cambios
conn.commit()

# Consulto todas las monedas
query = "SELECT * FROM currency"

# Busco el resultado
currencies = cursor.execute(query).fetchall()
print(currencies)

# Cierro la conexión a la base de datos
conn.close()
```

Conexiones

Una conexión a base de datos es la forma que un servidor de base de datos y su software cliente se comunican entre sí. El cliente y el servidor pueden o no estar en una misma máquina.

El cliente utiliza una conexión a base de datos para enviar comandos y recibir respuestas del servidor.

A continuación, veremos los métodos más importantes que implementa la clase Connection de sqlite3.

Método	Descripción
cursor	Devuelve un cursor que es instancia de la clase Cursor o de una subclase.
commit	Hace commit de la transacción actual. Si no se llama a este método, nada de lo que hayas hecho desde la anterior vez que se hizo un commit podrá ser visto desde otra conexión. Si te preguntas porque no ves algo que escribiste en la base de datos, chequea que no te hayas olvidado de hacer commit.
rollback	Este método revierte cualquier cambio que hayas hecho en la base de datos desde el último commit.
close	Este método cierra la conexión a la base de datos. Este método no llama automáticamente a commit. Si no haces commit antes de cerrar la conexión todos los cambios que hayas hecho durante esa conexión se pierden.

execute	Este método es un atajo que crea un cursor y ejecuta la consulta con los parámetros dados utilizando el método *execute* del cursor, y devuelve el cursor.
executemany	Este método es un atajo que crea un cursor y ejecuta varias consultas con los parámetros dados utilizando el método *executemany* del cursor, y devuelve el cursor.
executescript	Este método es un atajo que crea un cursor y ejecuta un *script SQL* utilizando el método *executescript* del cursor, y devuelve el cursor.
create_function	Crea una función definida por el usuario que se podrá utilizar luego dentro de las sentencias SQL utilizando el nombre de función definido. La función definida puede devolver cualquier tipo de valor soportado por SQLite: bytes, str, int, float y None.
create_aggregate	Crea una función de agregación definida por el usuario. Para esto se debe definir una clase que implemente los métodos *step* y *finalize*, este último devuelve el resultado final de la agregación. El método *finalize* puede devolver cualquiera de los tipos soportados por SQLite: bytes, str, int, float y None.
create_collation	Crea un comparador con el nombre y con la función especificada. La función deberá recibir dos strings. La misma debe devolver -1 si el primer argumento es menor que el segundo, 0 si son iguales y 1 si el primer argumento es mayor que el segundo.
iterdump	Devuelve un iterador para volcar la base de datos en un formato de texto SQL. Muy útil para guardar una base de datos en memoria para restaurarla posteriormente.
backup	Este método hace una copia de seguridad de una base de datos SQLite incluso cuando la base de datos está siendo accedida por otros clientes o concurrentemente por la misma conexión.

Cursores

Un cursor es una estructura de control utilizada para el recorrido (y potencial procesamiento) de los registros del resultado de una consulta.

A continuación, veremos los métodos más importantes que implementa la clase Cursor de sqlite3.

Método	Descripción
execute	Ejecuta una consulta SQL. La consulta SQL puede estar parametrizada. *sqlite3* soporta dos tipos de *placeholders* (para reemplazar las variables por los parámetros): el signo de pregunta y los *placeholders* con nombre. El método *execute* solo ejecuta una sola sentencia SQL.
executemany	Ejecuta un comando SQL para un conjunto de parámetros definidos. Por ejemplo, insertar 5 monedas.
executescript	Ejecuta múltiples consultas SQL. Antes de ejecutar el *script* hace un commit y luego ejecuta el *Script SQL* recibido como parámetro.
fetchone	Extrae la siguiente fila del conjunto resultado de la consulta. Devuelve una secuencia o None cuando no queda datos.
fetchmany	Extrae el siguiente conjunto de filas del resultado de la consulta, devolviendo una lista. Se devuelve una lista vacía cuando no quedan más filas. El número de filas a extraer se especifica en el parámetro size. Si no se define este parámetro, se utiliza como valor por defecto el valor de la variable *arraysize* del cursor.
fetchall	Extrae todas las filas que queden sin consumir en el conjunto resultado de la consulta efectuada. Devuelve una lista.
close	Cierra el cursor. El cursor queda inhabilitado a partir de ese momento. Si se trata de utilizar se levantará una excepción de tipo ProgrammingError.

Excepciones

A continuación, se listan los tipos de excepción que provee la librería sqlite3:

Excepción	Descripción
Warning	Una subclase de Exception.
Error	La clase base de todas las otras excepciones definidas en la librería. Es subclase de Exception.
DatabaseError	Excepción que es levantada para errores relacionados con la base de datos.
IntegrityError	Excepción que es levantada cuando la integridad referencial de la base de datos es afectada. Por ejemplo, cuando falla el chequeo de una clave foránea.
ProgrammingError	Excepción que es levantada para errores de programación. Por ejemplo, una tabla no encontrada o que ya existe, errores de sintaxis en sentencias SQL, número erróneo de los parámetros especificados, etc. Es subclase de DatabaseError.
OperationalError	Excepción que es levantada para errores que están relacionados con la operación de la base de datos y no están, necesariamente, bajo el control del programador. Por ejemplo, una desconexión con la base de datos inesperada, una transacción que no puede ser procesada, etc. Es subclase de DatabaseError.
NotSupportedError	Excepción que es levantada en caso que un método es usado cuando no es soportado por la base de datos. Por ejemplo, un rollback sobre una conexión que no soporta transacciones o las tiene deshabilitadas. Es subclase de DatabaseError.

Transacciones

Una transacción en un Sistema de Gestión de Bases de Datos es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

La biblioteca *sqlite3* subyacente opera en modo *autocommit* de forma predeterminada, pero el módulo *sqlite3* de Python no lo hace de manera predeterminada.

El modo *autocommit* significa que las consultas SQL que modifican la base de datos se efectúan inmediatamente.

Una sentencia BEGIN o SAVEPOINT deshabilita el modo *autocommit* y un *commit*, *rollback* o *release* termina la transacción más externa y vuelve a activar el modo *autocommit*.

El módulo sqlite3 de Python por defecto emite una sentencia BEGIN implícitamente antes de realizar una sentencia de modificación de datos (INSERT/UPDATE/DELETE).

Se puede controlar qué tipo de sentencia BEGIN se ejecuta implícitamente por *sqlite3* por medio del parámetro *isolation_level* al crear la conexión, o por medio de la propiedad *isolation_level* de las conexiones. Si no se especifica este parámetro, un BEGIN plano es utilizado, que es equivalente a especificar DEFERRED. Otros valores posibles son IMMEDIATE y EXCLUSIVE.

Además, se puede deshabilitar el manejo de transacciones implícito por configuración, defiendo *isolation_level* en None. Esto dejará que la librería sqlite3 opere en modo *autocommit.* En este caso, se puede controlar el estado de la transacción explícitamente emitiendo las sentencias BEGIN, ROLLBACK, SAVEPOINT y RELEASE en el código.