Apuntes de Python: MySQL con sqlite3

Aclaración importante

La librería `sqlite3` es para trabajar con **SQLite**, no con MySQL. Son sistemas de bases de datos diferentes:

- **SQLite**: Base de datos ligera, sin servidor, almacenada en un archivo
- **MySQL**: Sistema de gestión de bases de datos cliente-servidor

Para MySQL necesitas librerías como `mysql-connector-python` o `PyMySQL`.

Trabajar con SQLite usando `sqlite3`

1. Importar la librería

```
import sqlite3
```

La librería `sqlite3` viene incluida en Python, no necesitas instalar nada adicional.

2. Conectar a una base de datos

```
# Conectar (crea el archivo si no existe)
conexion = sqlite3.connect('mi_base_datos.db')

# Usar base de datos en memoria (temporal)
conexion = sqlite3.connect(':memory:')
```

3. Crear un cursor

El cursor ejecuta las consultas SQL:

```
cursor = conexion.cursor()
```

4. Crear tablas

```
cursor.execute('''
    CREATE TABLE IF NOT EXISTS usuarios (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL,
        email TEXT UNIQUE,
        edad INTEGER
    )
''')
conexion.commit()  # Guardar cambios
```

5. Insertar datos

Inserción simple:

```
cursor.execute('''
    INSERT INTO usuarios (nombre, email, edad)
    VALUES ('Ana García', 'ana@email.com', 28)
''')
conexion.commit()
```

Inserción con parámetros (recomendado - previene SQL injection):

```
datos = ('Carlos López', 'carlos@email.com', 35)
cursor.execute('INSERT INTO usuarios (nombre, email, edad) VALUES (?, ?, ?)', datos)
conexion.commit()
```

Insertar múltiples registros:

```
usuarios = [
    ('María Pérez', 'maria@email.com', 24),
    ('Juan Martínez', 'juan@email.com', 31),
    ('Laura Sánchez', 'laura@email.com', 29)
]
cursor.executemany('INSERT INTO usuarios (nombre, email, edad) VALUES (?, ?, ?)', usuarios)
conexion.commit()
```

6. Consultar datos

Obtener todos los registros:

```
cursor.execute('SELECT * FROM usuarios')
resultados = cursor.fetchall()  # Lista de tuplas
for fila in resultados:
    print(fila)
```

Obtener un solo registro:

```
cursor.execute('SELECT * FROM usuarios WHERE id = ?', (1,))
usuario = cursor.fetchone()
print(usuario)
```

Obtener registros de forma iterativa:

```
cursor.execute('SELECT nombre, edad FROM usuarios WHERE edad > 25')
for nombre, edad in cursor.fetchall():
    print(f'{nombre}: {edad} años')
```

Usar fetchmany():

```
cursor.execute('SELECT * FROM usuarios')
while True:
    filas = cursor.fetchmany(2)  # Obtener 2 registros a la vez
```

```
    if not filas:
        break
    for fila in filas:
        print(fila)
```

7. Actualizar datos

```
cursor.execute('''
    UPDATE usuarios
    SET edad = ?
    WHERE nombre = ?
''', (30, 'Ana García'))
conexion.commit()
print(f'Filas actualizadas: {cursor.rowcount}')
```

8. Eliminar datos

```
cursor.execute('DELETE FROM usuarios WHERE edad < ?', (25,))
conexion.commit()
print(f'Filas eliminadas: {cursor.rowcount}')
```

9. Trabajar con Row Factory

Para obtener resultados como diccionarios:

```
conexion.row_factory = sqlite3.Row
cursor = conexion.cursor()

cursor.execute('SELECT * FROM usuarios')
for fila in cursor.fetchall():
    print(f"ID: {fila['id']}, Nombre: {fila['nombre']}")
```

10. Transacciones

```
try:
    cursor.execute('INSERT INTO usuarios (nombre, email, edad) VALUES (?, ?, ?)',
                   ('Pedro Ruiz', 'pedro@email.com', 27))
    cursor.execute('INSERT INTO usuarios (nombre, email, edad) VALUES (?, ?, ?)',
                   ('Lucía Fernández', 'lucia@email.com', 33))
    conexion.commit()
except sqlite3.Error as e:
    conexion.rollback()  # Revertir cambios si hay error
    print(f'Error: {e}')
```

11. Cerrar conexiones

```
cursor.close()
conexion.close()
```

Mejor práctica con context manager:

```
with sqlite3.connect('mi_base_datos.db') as conexion:
    cursor = conexion.cursor()
    cursor.execute('SELECT * FROM usuarios')
    resultados = cursor.fetchall()
# La conexión se cierra automáticamente
```

12. Manejo de errores comunes

```
import sqlite3

try:
    conexion = sqlite3.connect('mi_base_datos.db')
    cursor = conexion.cursor()

    cursor.execute('INSERT INTO usuarios (nombre, email) VALUES (?, ?)',
                   ('Test', 'test@email.com'))
    conexion.commit()

except sqlite3.IntegrityError:
    print('Error: Email duplicado o violación de restricción')
except sqlite3.OperationalError:
    print('Error: Problema con la operación SQL')
except sqlite3.Error as e:
    print(f'Error de SQLite: {e}')
finally:
    if conexion:
        conexion.close()
```

13. Funciones útiles

```
# Obtener lista de tablas
cursor.execute("SELECT name FROM sqlite_master WHERE type='table'")
tablas = cursor.fetchall()

# Obtener último ID insertado
cursor.execute('INSERT INTO usuarios (nombre, email) VALUES (?, ?)',
               ('Nuevo Usuario', 'nuevo@email.com'))
ultimo_id = cursor.lastrowid
print(f'ID insertado: {ultimo_id}')

# Contar registros
cursor.execute('SELECT COUNT(*) FROM usuarios')
total = cursor.fetchone()[0]
print(f'Total de usuarios: {total}')
```

Para usar MySQL en Python

Si necesitas trabajar con MySQL específicamente, instala:

```
pip install mysql-connector-python
```

Ejemplo básico con MySQL:

```
import mysql.connector

conexion = mysql.connector.connect(
    host="localhost",
    user="tu_usuario",
    password="tu_contraseña",
    database="tu_base_datos"
)

cursor = conexion.cursor()
cursor.execute("SELECT * FROM tabla")
resultados = cursor.fetchall()

conexion.close()
```