Python y SGBD

- SQL Server
- Oracle
- MySql
- SQLite
- PostgreSQL
- Etc.

Python es capaz de manipular la información que se encuentra almacenada en varios SGBD. No sólo BD relacionales, sino también BD orientadas en Objetos.

El lenguaje que nos va a permitir manipular la información es

SQL (Structured Query Language)

Vamos a ver SqLite y MySql

- Sistema de gestión de BBDD relacional.
- Escrito en C, es de código abierto.
- Forma parte integral del programa.
- Se guarda como un único fichero en host.

Ventajas	Inconvenientes
• Ocupa muy poco espacio en disco y memoria	• No admite clausulas anidadas (where)
• Muy eficiente y rápido	• No existen usuarios (no acceso simultáneo por parte de varios usuarios a la vez)
• Multiplataforma	• Falta de clave foránea cuando se crea en modo consola
• Sin administración / configuración	
• Dominio público. Sin costo	

PASOS A SEGUIR PARA CONECTAR CON UNA BBDD

1. Abrir-Crear conexión
2. Crear puntero
3. Ejecutar query (consulta) SQL
4. Manejar los resultados de la query (consulta).
    1. Insertar, Leer, Actualizar, Borrar (Create, Read, Update, Delete)
5. Cerrar puntero
6. Cerrar conexión

Continuamos viendo cómo trabajar con BD en Python.

Con las siguientes líneas de código creamos la Base de Datos Sqlite vacia

```
import sqlite3

miConexion=sqlite3.connect("PrimeraBase")



miConexion.close()
```

Creamos el cursor, y luego utilizamos ese cursor o puntero ejecutar SQL

Para ver la Base de Datos poder usar el Visor Sqlite DBBrowser

https://sqlitebrowser.org/

Si lo instalamos, luego podemos arrastrar la base que se creó y ver su estructura y contenido desde allí.

```
import sqlite3

miConexion=sqlite3.connect("BaseUno")

miCursor=miConexion.cursor()

miCursor.execute("CREATE TABLE PRODUCTOS(NOMBRE_ARTICULO VARCHAR(50),PRECIO INTEGER, SECCION VARCHAR(20))")
miCursor.execute('INSERT INTO PRODUCTOS VALUES("BALON",15,"DEPORTES")')

miConexion.commit()

miConexion.close()
```

Creamos la tabla Productos y le insertamos un registro.

En el insert tenemos comillas de comillas entonces una debe ser “ y otra simple ´

Verificar que queremos hacer el cambio con el método commit() confirmamos los cambios

Creamos la tabla Productos pero no le asignamos Clave, podríamos realizarlo como muestra la siguiente figura. Si no queremos escribir todo el código en la misma línea podemos usar la triple comilla.

```
miCursor.execute('''
    CREATE TABLE PRODUCTOS (
    CODIGO_ARTICULO VARCHAR(4) PRIMARY KEY,
    NOMBRE_ARTICULO VARCHAR(50),
    PRECIO INTEGER,
    SECCION VARCHAR(20))
''')
```

Inserción de múltiples filas en una tabla

```
productos=[

    ("AR01", "pelota", 20, "juguetería"),
    ("AR02", "pantalón", 15, "confección"),
    ("AR03", "destornillador", 25, "ferretería"),
    ("AR04", "jarrón", 45, "cerámica")

]
```

Si queremos agregar un grupo de registros a la tabla, podemos definir una **lista** donde cada elemento de la misma sean los datos del registro

La clase cursor a parte del método **'execute**' cuenta con otro método llamado `executemany()` que tiene el objetivo de insertar múltiples filas de una tabla.

Pasa varios registros a la tabla de una base de datos con una sola instrucción. Como las filas o registros se almacenan en una tupla, para pasarlos es necesaria una lista con los registros, teniendo en cuenta que cada elemento de la lista es una tupla con los valores de los campos a pasar.

Para insertar estos registros el método executemany . Se debe poner tantos **?** como campos tenga el registro y luego el nombre de la lista donde están guardados los datos

```
miCursor.executemany("INSERT INTO PRODUCTOS VALUES (?,?,?,?)", productos)
```

CRUD (Create, Read, Update, Delete)

Para realizar una consulta (Read) de todos los productos de la sección confección se escribe este código.

```
miCursor.execute("SELECT * FROM PRODUCTOS WHERE SECCION='CONFECCIÓN'")

productos=miCursor.fetchall()
```

Tener en cuenta que es case sensitive por lo tanto si lo escribo en mayúscula y lo guarde en minúscula no lo va a encontrar.
El método `fetchall` trae los resultados de un **select** por lo que debo ponerlo a continuación del **execute**
Luego podríamos realizar un `print(productos)` y nos mostrará el resultado del **selec**t

UPDATE

```
miCursor.execute("UPDATE PRODUCTOS SET PRECIO=35 WHERE NOMBRE_ARTICULO='pelota'")
```

Si queremos actualizar el precio del artículo

DELETE

```
miCursor.execute("DELETE FROM PRODUCTOS WHERE ID=5")
```

```
miConexion.commit()

miConexion.close()
```

Siempre luego de los comando sql debo tener estas líneas de código

Excepciones SQLite3

Las excepciones son errores de tiempo de ejecución. en la programación en Python, todas las excepciones son instancias de la clase derivadas de la BaseException.

En SQLite3, tenemos las siguientes excepciones principales de Python:

DatabaseError

Cualquier error relacionado con la base de datos genera el DatabaseError.

IntegrityError

IntegrityError es una subclase de DatabaseError y se genera cuando hay un problema de integridad de los datos, por ejemplo, los datos foráneos no se actualizan en todas las tablas, lo que resulta en una inconsistencia de los datos.

ProgrammingError

La excepción ProgrammingError se produce cuando hay errores de sintaxis o no se encuentra la tabla o se llama a la función con un número incorrecto de parámetros / argumentos.

OperationalError

Esta excepción se produce cuando fallan las operaciones de la base de datos, por ejemplo, una desconexión inusual. Esto no es culpa de los programadores.

NotSupportedError

Ocurre cuando utilizas algunos métodos que no están definidos o no son compatibles con la base de datos, se genera la excepción NotSupportedError.

Paquete de Python necesario para conectarnos a MySQL.

Utilizaremos el programa 'pip' que vimos anteriormente para instalar el paquete necesario para interconectar 'Python' y 'MySQL'.

Si queremos usar **MySql** es similar, los únicos cambios son en importar el módulo

Desde la línea de comandos ejecutamos el programa pip con el siguiente paquete a instalar:

```
pip install mysql-connector
```

‘ Si queremos usar **MySql** es similar, los únicos cambios son en importar el módulo

```
import mysql.connector

miConexion=mysql.connector.connect(host="localhost",
                                   user="root",
                                   passwd="",
                                   database="BaseDos")
miCursor=miConexion.cursor()

miCursor.execute("CREATE TABLE PRODUCTOS(NOMBRE_ARTICULO VARCHAR(50),PRECIO INTEGER, SECCION VARCHAR(20))")
miCursor.execute('INSERT INTO PRODUCTOS VALUES("BALON",15,"DEPORTES")')

conexion1.close()
```

```
import mysql.connector

conexion1=mysql.connector.connect(host="localhost",
                                  user="root",
                                  passwd="",
                                  database="bd1")
cursor1=conexion1.cursor()
cursor1.execute("delete from articulos where codigo=1")
cursor1.execute("update articulos set precio=50 where codigo=3")
conexion1.commit()
cursor1.execute("select codigo, descripcion, precio from articulos")
for fila in cursor1:
    print(fila)
conexion1.close()
```