GOBIERNO DE ESPAÑA
MINISTERIO DE HACIENDA Y FUNCIÓN PÚBLICA
INAP
INSTITUTO NACIONAL DE ADMINISTRACIÓN PÚBLICA

# Introducción a la programación en Python

## Tema 14. Bases de Datos

**Autor:** Borja Rodríguez Cuenca

# Contenido

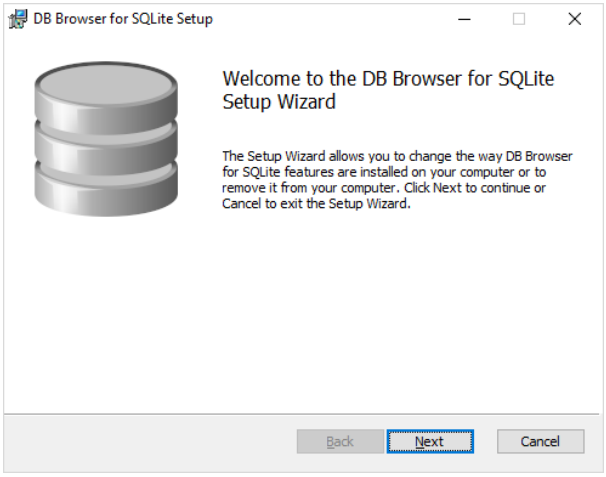
# 0. El módulo sqlite3 y DB Browser

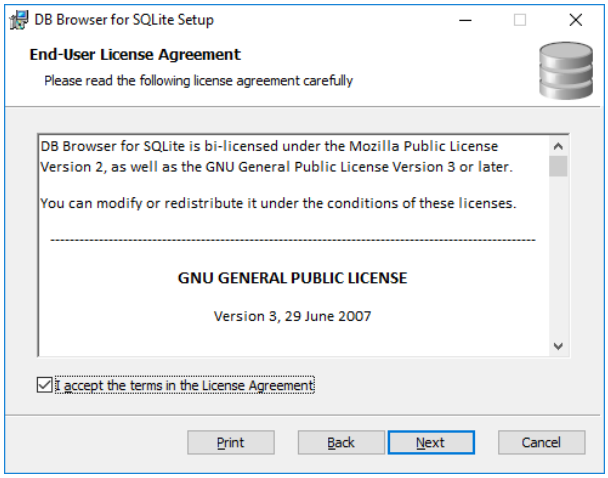
En este módulo vamos a trabajar con la librería de Python **sqlite3**. SQLite es un sistema de gestión de bases de datos relacional. La biblioteca SQLite se enlaza con el programa principal pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones, lo que reduce la latencia en el acceso a la base de datos El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un solo fichero estándar en la máquina host.

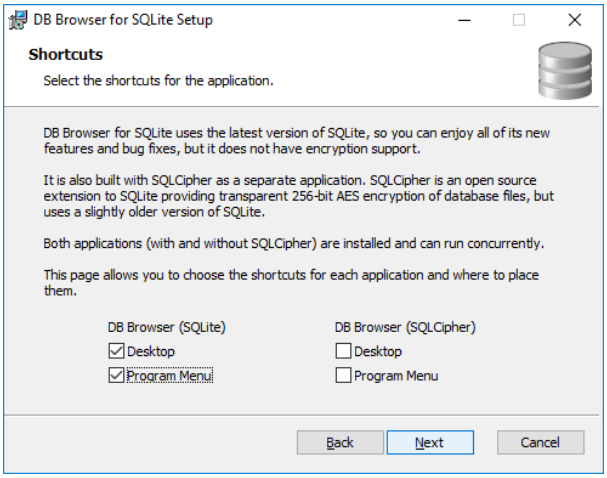
DB Browser para SQLite (DB4S) es una herramienta de código abierto que permite crear, diseñar y editar archivos de base de datos compatibles con SQLite. La descarga, libre y gratuita, se efectúa desde el siguiente enlace:

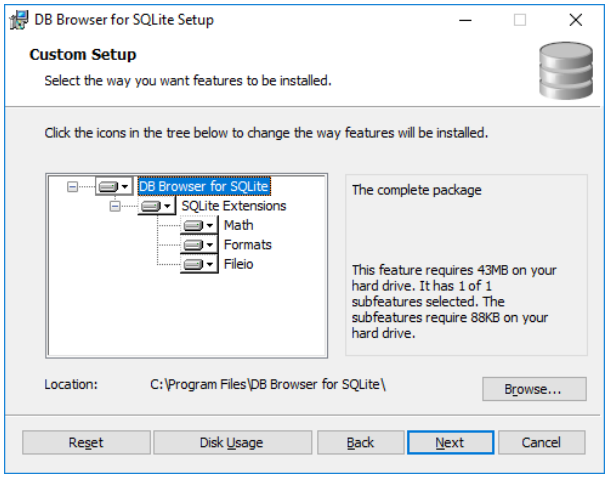
https://sqlitebrowser.org/dl/

En esta web debemos seleccionar y descargar el instalador que se ajusta a nuestro equipo (Windows 32 o 64 bits, LINUX o MAC). A continuación, se muestran los pasos para la instalación en un equipo Windows 64 bits. Dejaremos las opciones de instalación por defecto y solo aceptaremos



lo términos de licencia y activaremos las opciones 'Desktop' y 'Program Menu' en la ventana con las opciones de 'Shortcuts'.

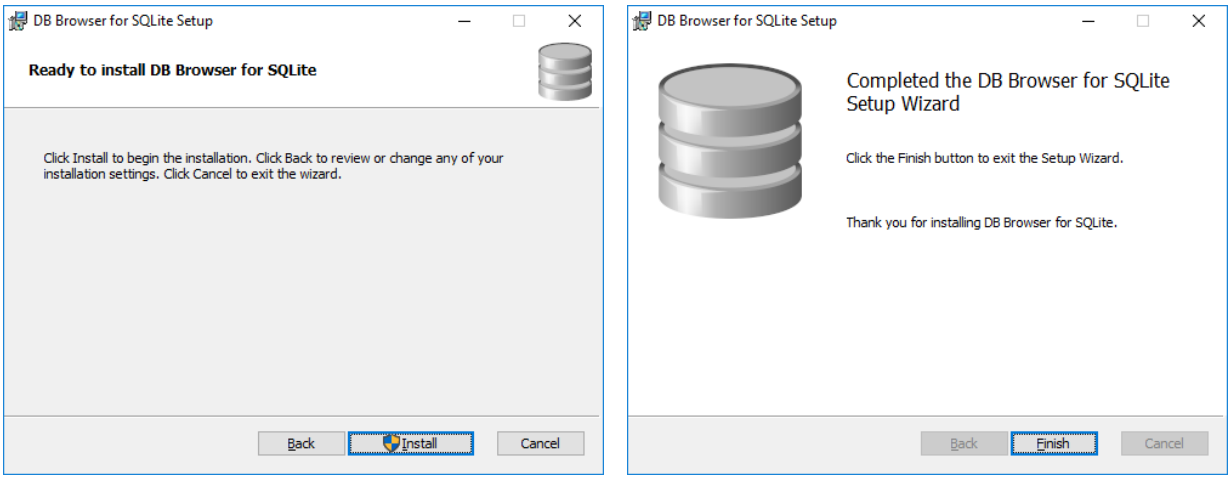



*Figura 1. Pasos del proceso de instalación de DB Browser en Windows 64 bits*

Si la instalación ha finalizado con éxito, en el escritorio aparecerá el icono DB Browser (SQLite). Si lo ejecutamos, accederemos a la ventana principal del programa DB Browser que utilizaremos para crear bases de datos y consultar su contenido.

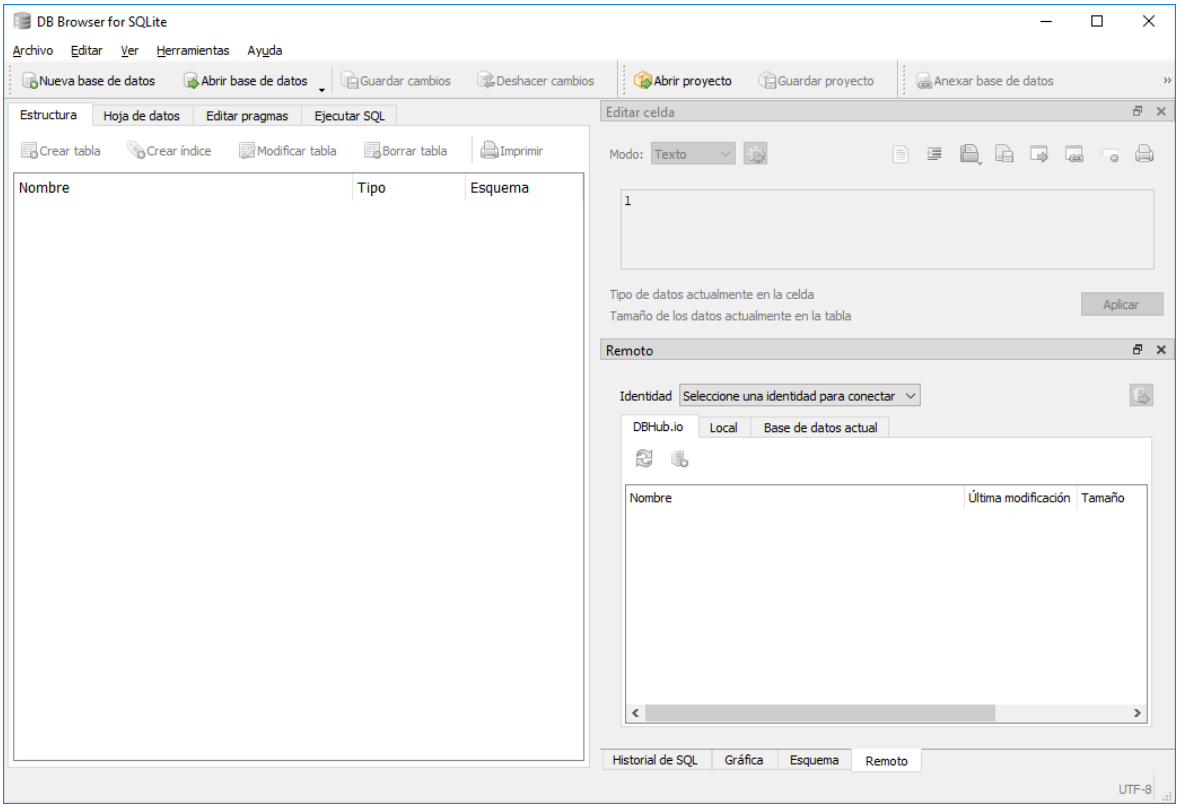



*Figura 2. Ventana principal del programa DB Browser*

# 1. Primeros pasos en una BBDD

## 1.1. Creación de una BBDD

Una vez que tenemos instalado correctamente DB Browser, vamos a comenzar a trabajar con bases de datos. El primer trabajo que vamos a realizar va a ser la creación de una base de datos desde la ventana principal de DB Browser. Con la opción "Nueva base de datos" debemos asignar un nombre al fichero de extensión .db (se recomienda utilizar un nombre sencillo, ya que lo utilizaremos repetidamente a lo largo de este módulo), seleccionar el directorio en el que vamos a crear nuestra base de datos, con extensión .db y aceptar la operación. Creada la BBDD, DB Browser nos permite crear una tabla en nuestro fichero .db, pero esa operación la realizaremos más adelante desde el módulo sqlite3. Si hemos creado la nueva base de datos correctamente, podemos observar su contenido en la ventana de DB Browser. Como se puede observar en la siguiente figura, inicialmente la base de datos está vacía y no tiene ni índices ni tablas.


*Figura 3. Contenido de la tabla recién creada*

A medida que vayamos creando tablas en nuestra base de datos, DB Browser nos permitirá visualizar sus características y su contenido.

### 1.2. Conexión a una BBDD desde Python

Una vez creada la base de datos desde DB Browser procederemos a trabajar con la librería **sqlite3** de Python. Para trabajar con una base de datos (editarla, realizar consultas…) debes hacer primero una conexión a ella. Es decir, debemos realizar un "enlace" a la BBDD para poder trabajar en ella. Para conectarnos a una base de datos se utiliza la función *"connect"*. A esta función hay que especificarle el nombre y el directorio en el que se ubica la base de datos a la que nos vamos a conectar.

Una vez establecida la conexión, hay que crear un *"cursor"*. Un cursor es una estructura de control que se usa para recorrer (y eventualmente procesar) los records de un result set.

Para completar la ejecución de las operaciones a realizar sobre una base de datos se debe añadir la función *commit()*. Esta función confirma cualquier transacción pendiente a la base de datos. A continuación, se muestra el código que permite realizar la conexión a la base de datos que creamos anteriormente. Muchas operaciones que realicemos sobre una base de datos serán "invisibles", es decir, no sabremos si la operación se ha realizado correctamente hasta que no comprobemos el contenido de la base de datos y las tablas en DB Browser. Vamos a ejecutar el siguiente código para conectarnos a la base de datos. Para comprobar que la operación se ha realizado satisfactoriamente vamos a añadir la impresión de la palabra "Hecho" al final del proceso.

```
import sqlite3

dir = r'C:\Users\Desktop\BBDD_Prueba.db'
conn = sqlite3.connect(dir)
cursor = conn.cursor()
conn.commit()

print ("Hecho")
```

## 2. Las tablas de una BBDD

Las tablas de una base de datos son el tipo de modelado de datos donde se guardan los datos recogidos por un programa. La estructura general de una tabla se asemeja a la vista de un programa de hojas de cálculo. En las tablas distinguimos dos tipos de elementos:

- Campo: el nombre de cada una de las columnas de la tabla. Debe ser único y tiene asociado un tipo de dato (cadena de texto, entero…)
- Registro: es cada una de las filas que componen la tabla

## 2.1. Creación de una tabla

Una vez realizada la conexión a la base de datos es momento de añadir información a la misma. Vamos a crear una tabla en la que introduciremos información personal de los empleados de una empresa. Las operaciones que realicemos con la librería sqlite3 y que se van a realizar sobre la base de datos serán sentencias SQL a las que se hará llamada desde una función *"execute"*.

La creación de una tabla se realiza con una operación SQL del tipo CREATE TABLE. Vamos a crear una tabla, de nombre "Tabla1", con seis campos: un identificador, nombre, apellidos, edad, dni y su lugar de nacimiento. Hay que especificar el tipo de variable de cada campo. Ejecuta el siguiente código y fíjate en cómo están definidos los campos (nombres de las columnas de la tabla), así como el tipo de dato de cada uno de ellos.

```
import sqlite3
dir = r'C:\Users\Desktop\BBDD_Prueba.db'
conn = sqlite3.connect(dir)
cursor = conn.cursor()
cursor.execute("DROP TABLE IF EXISTS Tabla1")
cursor.execute("CREATE TABLE Tabla1 (id INTEGER PRIMARY KEY, nombre char (25),
ape1 char(25), ape2 char(25), dni TEXT, edad char(25), origen char(25))")

conn.commit()
print ("Hecho")
```

Por si acaso hubiéramos creado anteriormente una tabla con el mismo nombre, antes de su creación vamos a añadir la función "DROP TABLE IF EXISTS". De esta forma nos aseguramos de que no vamos a crear una tabla que ya existe en nuestra base de datos.

Comprobaremos en DB Browser que la creación se ha realizado correctamente. Para ello, debemos abrir nuevamente nuestra base de datos y comprobar que la tabla que hemos creado aparece en el desplegable "Tablas" y en ella podemos ver todos los campos que hemos indicado en la sentencia SQL.

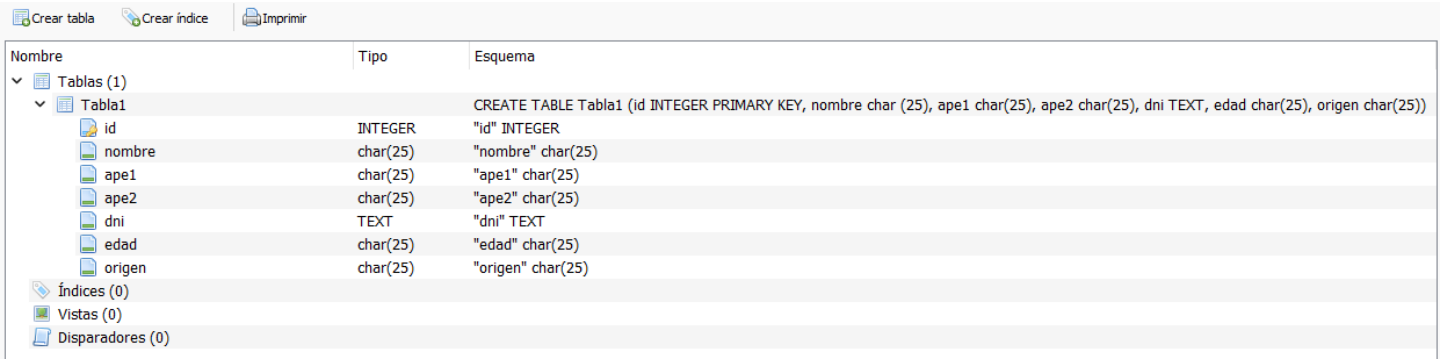



*Figura 4. Tabla con 6 registros que hemos creado con la sentencia SQL*

## 2.2. Insertar datos en nuestra base de datos

### 2.2.1. Insertar datos manualmente

Tras la creación de la tabla procederemos a introducir información en ella. En este primer ejercicio vamos a rellenar una fila de la tabla con nuestros propios datos personales. La sentencia SQL que permite insertar datos en una tabla es “INSERT INTO”; se debe especificar qué campos de la tabla se van a rellenar y a continuación los valores que se van a introducir, entre paréntesis, tras la sentencia “VALUES”.

A continuación, se muestra el código para introducir datos de forma manual. Vamos a introducir nuestros datos personales y comprobar en DB Browser que la operación que hemos realizado se ha ejecutado correctamente. Para ello debemos acceder a la pestaña “Hoja de datos”

```
import sqlite3

dir = r'C:\Users\Desktop\BBDD_Prueba.db'
conn = sqlite3.connect(dir)
cursor = conn.cursor()

cursor.execute("INSERT INTO Tabla1 (nombre, ape1, ape2, edad, origen) VALUES
('Carlos', 'Rodríguez', 'García', '30', 'Asturias')")

conn.commit()
print ("Hecho")
```

Trata de introducir varios campos con nombres y apellidos aleatorios. Observa que no es necesario especificar el identificador de la fila. Antes de pasar al siguiente ejercicio, borra todas las filas que hayas creado con el siguiente comando:

```
cursor.execute("DELETE FROM  Tabla1 WHERE id = 1")
```

### 2.2.2. Insertar datos a partir de un documento de texto

Generalmente una base de datos no se rellena de forma manual, sino que sus campos se completan a partir de la información contenida en otro documento. En este ejercicio vamos a completar la Tabla1 de nuestra base de datos con la información del archivo “DatosBBDD.txt”.

Antes de modificar la base de datos, vamos a recordar cómo leer desde Python los campos de un documento de texto con las funciones “open” y “readlines” vistas en anteriores lecciones del curso. Como novedad, en este ejercicio vamos a introducir la función “split", que permite romper un string por el caracter que especifique el usuario. En este caso, ya que cada campo del documento de texto

está separado por un tabulador, vamos a romper cada línea del documento de texto por el tabulador ('\t', que es como se codifica la tabulación).

**Ejercicio 1**: se propone al alumno crear un código que permita imprimir los nombres contenidos en el archivo DatosBBDD.txt.

Ejecutando el código que se muestra a continuación vamos a insertar la información contenida en el documento DatosBBDD.txt en la Tabla1 de nuestra base de datos. IMPORTANTE: modifica la ruta de la línea "`a = open('C:\\Users\\curso01\\Desktop\\curso\\DatosBBDD1.txt', encoding='latin-1')`", adaptándola al directorio en el que tengas almacenado el documento "DatosBBDD1.txt".

```
import sqlite3

dir = r'C:\Users\Desktop\BBDD_Prueba.db'
conn = sqlite3.connect(dir)
cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS Tabla1")
cursor.execute("CREATE TABLE Tabla1 (id INTEGER PRIMARY KEY, nombre char (25),
ape1 char(25), ape2 char(25), dni int, edad char(25), origen char(25))")

a = open(r'C:\Users\DatosBBDD1.txt')
linesa = a.readlines()

for i in range(len(linesa)-1):
    nom = str(linesa[i+1].split("\t")[0])
    ape1 = linesa[i+1].split("\t")[1]
    ape2 = linesa[i+1].split("\t")[2]
    dni = linesa[i+1].split("\t")[3]
    edad = linesa[i+1].split("\t")[4]
    provincia = linesa[i+1].split("\t")[5]
    id = linesa[i+1].split("\t")[6]

    datos = "'"+nom+"', '"+ape1+"', '"+ape2+"', '"+dni+"', '"+edad+"',
'"+provincia+"', '"+id+"'"
    print (i, datos)
    cursor.execute("INSERT INTO Tabla1 (nombre, ape1, ape2, dni, edad, origen,
id) VALUES ("+datos+")")
conn.commit()

print ("Hecho")
```

Comprobaremos que la operación se ha realizado correctamente observando el contenido de la Tabla1 en el DB Browser:

Tabla: Tabla1

| | id | nombre | ape1 | ape2 | dni | edad | origen |
|---|---|---|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro | Filtro |
| 1 | 1 | Carlos | Garcia | Garcia | 00000000A | 43 | Madrid |
| 2 | 2 | Patricia | Rodriguez | Menendez | 11111111B | 23 | Barcelona |
| 3 | 3 | Nuria | Moreno | Posada | 22222222C | 39 | Asturias |
| 4 | 4 | Marta | Galan | Rivas | 33333333D | 53 | Madrid |
| 5 | 5 | Adrian | Gonzalez | Crespo | 44444444E | 41 | Sevilla |
| 6 | 6 | Susana | Muñiz | Castillo | 55555555F | 25 | Albacete |
| 7 | 7 | Daniel | Silva | Peña | 66666666G | 28 | Badajoz |
| 8 | 8 | Carmen | Herrero | Sanchez | 77777777H | 39 | Cantabria |
| 9 | 9 | Francisco | Nieto | Pueyo | 88888888I | 49 | Asturias |
| 10 | 10 | Sandra | Alonso | Carrasco | 99999999J | 33 | Madrid |

*Figura 5. Visualización del contenido de la tabla en DB Browser*

# 3. Actualizar una base de datos

En ocasiones puede suceder que se cometa un error al insertar información en una base de datos o que sea necesaria su actualización. En este ejercicio vamos a practicar la sentencia SQL "UPDATE", que permite actualizar un campo de nuestra tabla. Modificaremos el nombre de la fila que tiene identificador id='1'.

Como novedad en este ejercicio, observad que la consulta SQL no se construye dentro de la función "execute", sino que se genera como un string asociado a la variable llamada "update". De esta forma podemos comprobar que la consulta SQL está correctamente construida.

```
import sqlite3

dir = r'C:\Users\Desktop\BBDD_Prueba.db'
conn = sqlite3.connect(dir)
cursor = conn.cursor()

update = "UPDATE Tabla1 SET nombre = 'Manuel' WHERE id = '1'"
print (update)
cursor.execute(update)

conn.commit()
print ("Hecho")
```

# 4. Consultas a la base de datos

Una de las operaciones más comunes en una base de datos es la de realizar consultas a la misma. El comando de SQL que permite realizar consultas es el "SELECT – FROM – WHERE", en el que se debe especificar qué campos queremos seleccionar de una tabla que satisfagan una serie de condiciones.

## 4.1. Consulta a una tabla

Con el objetivo de comprender el funcionamiento del comando "Select" vamos a realizar una consulta sobre la Tabla1 para seleccionar únicamente aquellas filas de la tabla cuyo campo origen sea 'Asturias'.

En las siguientes líneas se muestra el código para realizar consultas a la Tabla1. La función execute("SELECT… FROM… WHERE…") extrae los campos que cumplen la condición impuesta. Para almacenar estos datos en memoria hay que utilizar la función "fetchall()", que devuelve una lista con los elementos que satisfacen la consulta realizada. Las consultas pueden realizarse sobre varias tablas y para facilitar la llamada a una u otra tabla, el comando FROM permite asignar un nombre abreviado a cada tabla. En el siguiente ejemplo a la Tabla1 se le asigna el nombre "a", lo que facilita la selección de campos de esa tabla, ya que ahorra el introducir "Tabla1" al comienzo de cada campo. Si se hiciera una consulta a más de una tabla se podría poner un nombre diferente a cada una de ellas (por ejemplo: FROM Tabla1 as a, Tabla2 as b) y de esta forma se haría más sencilla la llamada a los campos de cada tabla.

```
import sqlite3

dir = r'C:\Users\Desktop\BBDD_Prueba.db'
conn = sqlite3.connect(dir)
cursor = conn.cursor()

cursor.execute(r'SELECT a.id, a.nombre, a.ape1, a.ape2, a.origen FROM Tabla1
as a WHERE a.origen = "Madrid"')
results = cursor.fetchall()

for i in range(len(results)):
    print (results[i])

conn.commit()
print ("Hecho")
```

Si la consulta se ha realizado correctamente, el resultado de ejecutar el código anterior serán los registros de la tabla cuyo origen sea 'Asturias'. De acuerdo al código anterior, las entradas de la tabla que cumplan con los requisitos del SELECT se imprimirán por pantalla.

El argumento “WHERE” de la opción “SELECT” utiliza funciones lógicas (igual (=), mayor que (>), o menor que (<)) para extraer los registros de una tabla.

**Ejercicio 2:** extrae los registros de la Tabla1 cuya edad sea superior a 40 años. Si lo haces correctamente, en pantalla se deben imprimir los datos de las 4 personas que tienen más de 40 años.

# 5. Recursos

https://docs.python.org/3/library/sqlite3.html

https://sqlitebrowser.org/

*El lenguaje de programación Python de principio a fin, Angel Pablo Hinojosa Gutiérrez*