1000 PROGRAMADORES

Introducción a la **Programación en Python**

Ministerio de Educación Cultura, Ciencia y Tecnología Gobierno de Salta

Ministerio de Economía y Servicios Públicos Gobierno de Salta

Universidad Nacional de Salta

MÓDULO 8

Bases de Datos

Base de Datos

Son conjuntos de datos que serán utilizados a través de una o más tareas que producen la información necesaria para un individuo u organización.

Este conjunto de datos puede ser de cualquier tipo, números, cadenas de caracteres, fechas, etc. que hacen referencia a información perteneciente a un mismo contexto.

Las bases de datos permiten almacenar sistemáticamente, de forma automatizada, con la finalidad de un uso posterior. Para realizar consultas, comparativas, análisis, modificaciones o borrar estos datos.

Productos de una tienda

- nombre
- precio
- descripción

Base de Datos

Las palabras son ***datos****, un libro es* ***información*** *dado que tiene datos ordenados con un propósito, contar una historia.*

Una biblioteca es una ***base de datos*** *porque almacena esa información de forma ordenada, por ejemplo por autor.*

Bases de Datos - SGBD

Los Sistemas Gestores de Bases de Datos (SGBD) se encargan del orden de una base de datos, son programas complejos centrados en la gestión de información.

Implementan sus propios lenguajes internos de programación para realizar consultas:

- Listar el nombre de todos los clientes de la empresa
- Conseguir información de los pedidos de una tienda, a la vez que se consultan los clientes en un conjunto de fechas determinado.

Modelos de Bases de Datos

Los tipos de datos y la forma de almacenarlos pueden diferir mucho dependiendo del contexto, con el tiempo se han ido desarrollando una serie de modelos distintos para gestionar las bases de datos:

- ➔ Jerárquicas
- ➔ De red
- ➔ Transaccionales
- ➔ Documentales
- ➔ Orientadas a objetos
- ➔ Deductivas
- ➔ Relacionales

Modelo Relacional

Sirven para representar problemas reales y administrar datos dinámicamente. Se basan en la idea de crear relaciones entre conjuntos de datos, en los que cada relación es también una tabla. Cada tabla consta de registros, formados por filas y columnas, también conocidos como tuplas y campos.

Dentro de las bases de datos relacionales, existen muchos SGBD. La mayoría son compatibles con Python. Algunos son pagos, otros gratuitos, los hay sencillos y otros muy avanzados:

- ➔ SQL Server
- ➔ Oracle
- ➔ MySQL:
- ➔ PostgreSQL
- ➔ SQLite

En Python, cada uno cuenta con módulos libres y programas conectores para comunicar las bases de datos y el lenguaje de programación. Pese a que son sistemas distintos, el lenguaje de las consultas no varía mucho.

El lenguaje SQL

Aparte de los lenguajes de programación como Python, centrados en la creación de programas, los SGBD implementan su propia sintaxis o lenguaje propio para realizar consultas y modificaciones en sus registros.

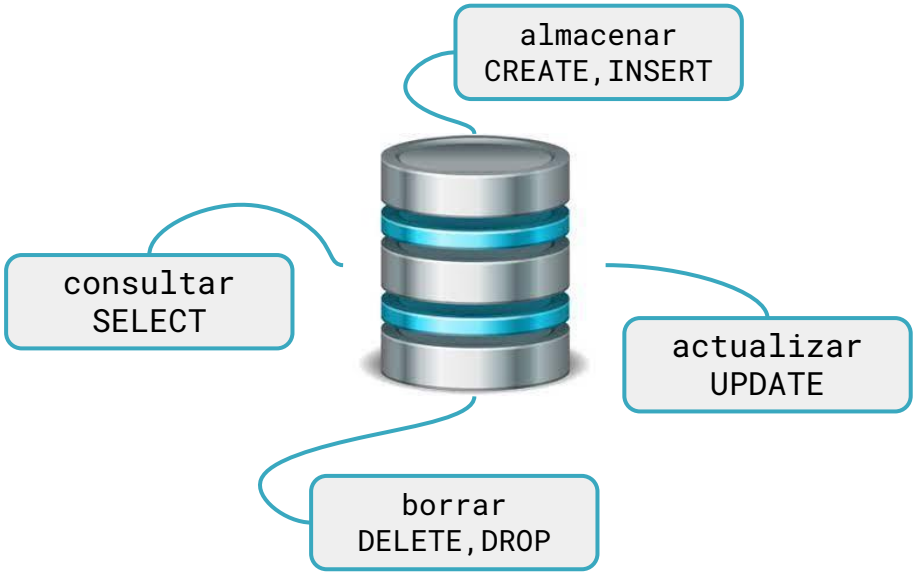
El lenguaje más utilizado en las bases de datos relacionales es el lenguaje SQL (Lenguaje de Consulta Estructurada), y es necesario aprenderlo si queremos utilizar este tipo de bases de datos en nuestros programas.

Este lenguaje abarca muchísimo contenido, por lo que en este módulo sólo veremos algunas consultas básicas para utilizar el SQLite en nuestros scripts de Python.

Bases de Datos en Python

Los pasos a seguir para conectar y usar una Base de Datos son:

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



Conexión a la BD, creación y desconexión

Al realizar la conexión, si la base de datos no existe, entonces la crea. Siempre debe cerrarse la conexión al finalizar el uso, sino luego no se podrá seguir manipulandola.

```
import sqlite3                                  # Importamos el módulo
conexion = sqlite3.connect('ejemplo.db')        # Conexión a la BD
conexion.close()                                # Cerramos la conexión
```

Crear una tabla

Antes de ejecutar una consulta (query) en código SQL, tenemos que crear un cursor. Luego se crea la tabla que necesitemos

```
import sqlite3
conexion = sqlite3.connect('ejemplo.db')
# Creamos el cursor
cursor = conexion.cursor()
# Crearemos una tabla de usuarios con nombres, edades y emails
cursor.execute("CREATE TABLE IF NOT EXISTS usuarios (nombre VARCHAR(100), edad INTEGER, email VARCHAR(100))")
conexion.close()
```

Inserción o carga de datos en una tabla

Cargar una tupla

```
import sqlite3
conexion = sqlite3.connect('ejemplo.db')
cursor = conexion.cursor()
# Insertamos un registro en la tabla de usuarios
cursor.execute("INSERT INTO usuarios VALUES ('Hector', 27, 'hector@ejemplo.com')")
# Guardamos los cambios haciendo un commit
conexion.commit()
conexion.close()
```

Inserción o carga de datos en una tabla

Cargar varias tuplas

```
import sqlite3
conexion = sqlite3.connect('ejemplo.db')
cursor = conexion.cursor()
# Creamos una lista con varios usuarios
usuarios = [('Mario', 51, 'mario@ejemplo.com'),
           ('Mercedes', 38, 'mercedes@ejemplo.com'),
           ('Juan', 19, 'juan@ejemplo.com')]
# Ahora utilizamos el método executemany() para insertar varios
cursor.executemany("INSERT INTO usuarios VALUES (?,?,?)", usuarios)
# Guardamos los cambios haciendo un commit
conexion.commit()
conexion.close()
```

Lectura de datos en una tabla

Lectura de una tupla

```
import sqlite3
conexion = sqlite3.connect('ejemplo.db')
cursor = conexion.cursor()
# Recuperamos los registros de la tabla de usuarios
cursor.execute("SELECT * FROM usuarios")
# Recorremos el 1er registro con fetchone, devuelve una tupla
usuario = cursor.fetchone()
print(usuario)
conexion.close()
```

Lectura de datos en una tabla

Lectura múltiple

```
import sqlite3
conexion = sqlite3.connect('ejemplo.db')
cursor = conexion.cursor()
# Recuperamos los registros de la tabla de usuarios
cursor.execute("SELECT * FROM usuarios")
# Se recorren los registros con fetchall y se vuelcan a una lista
usuarios = cursor.fetchall()
# Ahora podemos recorrer todos los usuarios
for usuario in usuarios:
  print(usuario)
conexion.close()
```

Actualizar/Modificar datos en una tabla

```
import sqlite3
conexion = sqlite3.connect('ejemplo.db')
cursor = conexion.cursor()
# Actualizamos un registro de la tabla de usuarios
cursor.execute("UPDATE usuarios SET edad=80 WHERE nombre='Juan'")
conexion.commit()
conexion.close()
```

Eliminar un registro de una tabla

```python
import sqlite3

conexion = sqlite3.connect('ejemplo.db')

cursor = conexion.cursor()

# Eliminamos un registro de la tabla de usuarios

cursor.execute("DELETE FROM usuarios WHERE nombre='Juan'")

conexion.commit()

conexion.close()
```

Ejemplo: El sueldo de un vendedor es la suma de un monto fijo pagado por el gerente más un porcentaje de sus ventas mensuales.
Si sus ventas fueron menores a $20000, no recibe porcentaje, si estuvieron entre $20000 y $50000 recibe el 20% de ellas y si superan los $50000 recibe el 25% de ellas. Teniendo como dato el sueldo fijo y el monto de la venta mensual cada vendedor, calcule, muestre y almacene el salario final que recibirá cada uno.

1) Crear una base de datos, con una tabla que posea: Legajo, Apellido, Nombre, SueldoFijo, VentaMes y Salario
2) Agregar datos de vendedores
3) Leer los datos de la BD y actualizar adecuadamente el campo Salario
4) Mostrar cambios

1. Crear una base de datos, con una tabla que posea: Legajo, Apellido, Nombre, SueldoFijo, VentaMes y Salario

```
import sqlite3

conexion = sqlite3.connect('pagos.db')

cursor = conexion.cursor()

cursor.execute("CREATE TABLE IF NOT EXISTS vendedores (legajo INTEGER, apellido
VARCHAR(100), nombre VARCHAR(100), SueldoFijo FLOAT, VentaMes FLOAT, Salario FLOAT)")

conexion.close()
```

2. Agregar datos de vendedores

```
import sqlite3

conexion = sqlite3.connect('pagos.db')

cursor = conexion.cursor()

vendedor = [('1000','Reyes','Juan',20000,10000,0),
            ('5000','Reyes','Oscar',20000,30000,0),
            ('1001','Reyes','Franco',20000,60000,0)]

cursor.executemany("INSERT INTO vendedores VALUES (?,?,?,?,?,?)", vendedor)

conexion.commit()

conexion.close()
```

2. Agregar datos de vendedores

```
import sqlite3

conexion = sqlite3.connect('pagos.db')

cursor = conexion.cursor()

vendedor = [('1000','Reyes','Juan',20000,10000,0),
            ('5000','Reyes','Oscar',20000,30000,0),
            ('1001','Reyes','Franco',20000,60000,0)]

cursor.executemany("INSERT INTO vendedores VALUES (?,?,?,?,?,?)", vendedor)

conexion.commit()

conexion.close()
```

3. Leer los datos de la Base de Datos y actualizar adecuadamente el Salario

```
import sqlite3
conexion = sqlite3.connect('pagos.db')
cursor = conexion.cursor()
cursor.execute("SELECT * FROM vendedores")
listado = cursor.fetchall()

for v in listado:
 if(v[4]<20000):
   sal=v[3]
 elif(v[4]>=20000 and v[4]<=50000):
   sal=v[3]+v[4]*0.2
 else:
   sal=v[3]+v[4]*0.25
 cursor.execute("UPDATE vendedores SET Salario=(?) WHERE Legajo=(?)",(sal,v[0]))
 conexion.commit()

conexion.close()
```

4. Mostrar cambios

```
import sqlite3

conexion = sqlite3.connect('pagos.db')

cursor = conexion.cursor()

cursor.execute("SELECT * FROM vendedores")

listado = cursor.fetchall()

for v in listado:
  print(v)

conexion.close()
```