Bases De Datos en Python

Fundamentos de Computación

Grado en Ingeniería en Tecnologías Industriales

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

UC | ETS de Ingenieros Industriales y de Telecomunicación

MOTIVACIÓN

En el aprendizaje de Python hemos ido introduciendo distintos tipos de datos, desde los datos simples a los tipos estructurados. Primero, se aprendió a trabajar con variables, luego, con listas y tuplas, seguido de los diccionarios. Pero, ¿hay alguna otra estructura de datos más grande?

Así como una variable puede visualizarse como un caso particular de una lista con un único elemento, una lista puede verse como un caso particular de un diccionario en el que las claves son enteros. Más allá, un diccionario *puede entenderse* como un caso particular de una base de datos (BDD) formada por una única tabla bastante simplificada.

Sistemas de Bases de Datos

Los sistemas de bases de datos (SBDD) se definen como una colección de datos interrelacionados y un conjunto de programas que permiten a los usuarios acceder y modificar esos datos.

Un propósito principal de un SBDD es proveer a los usuarios de una vista abstracta de los datos.

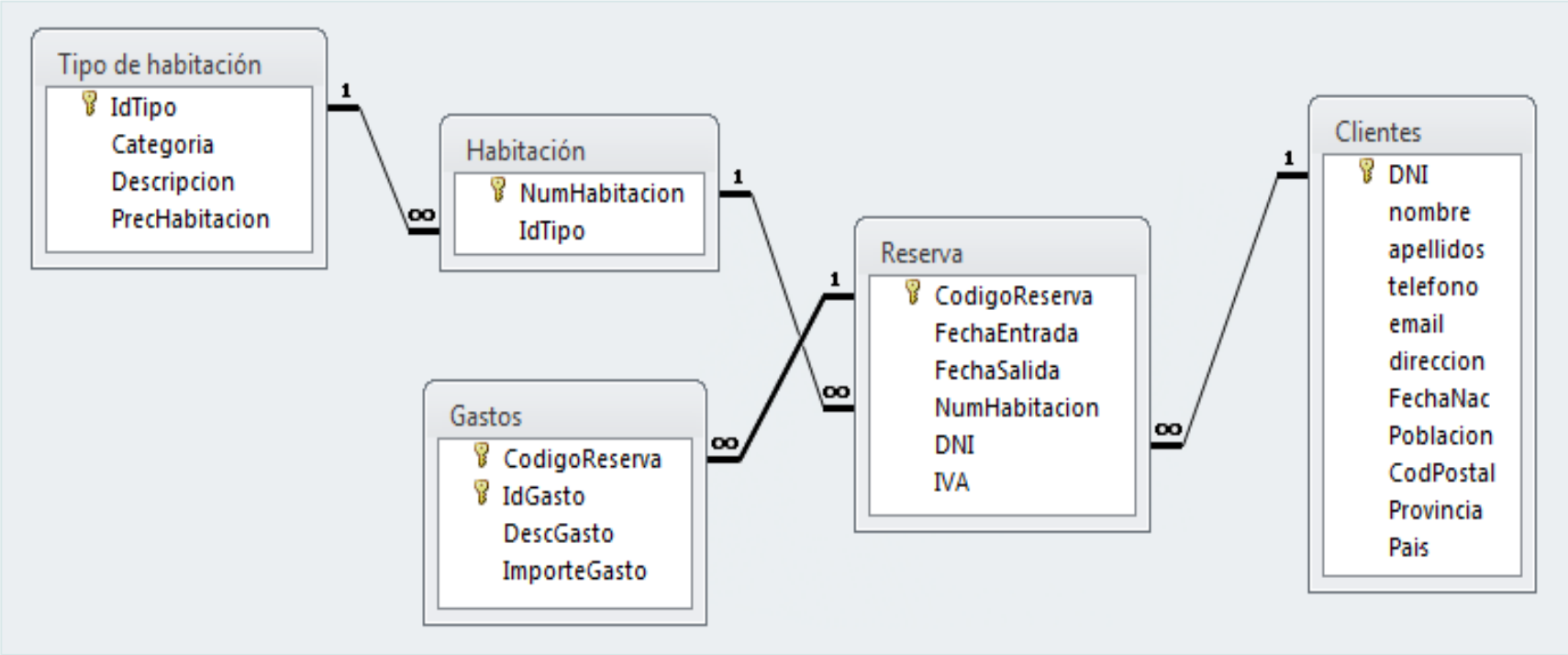
Un modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones entre los datos, la semántica de los datos y las restricciones de consistencia.

Categorías de modelos de datos: relacional, entidad-relación, semi-estructurado y orientado a objetos.

El más usado actualmente es el modelo **relacional**. La gran mayoría de los SBDD actuales están basados en este modelo. En lo que sigue, nos centraremos en el modelo relacional.

Diseño en las BDD RELACIONALES

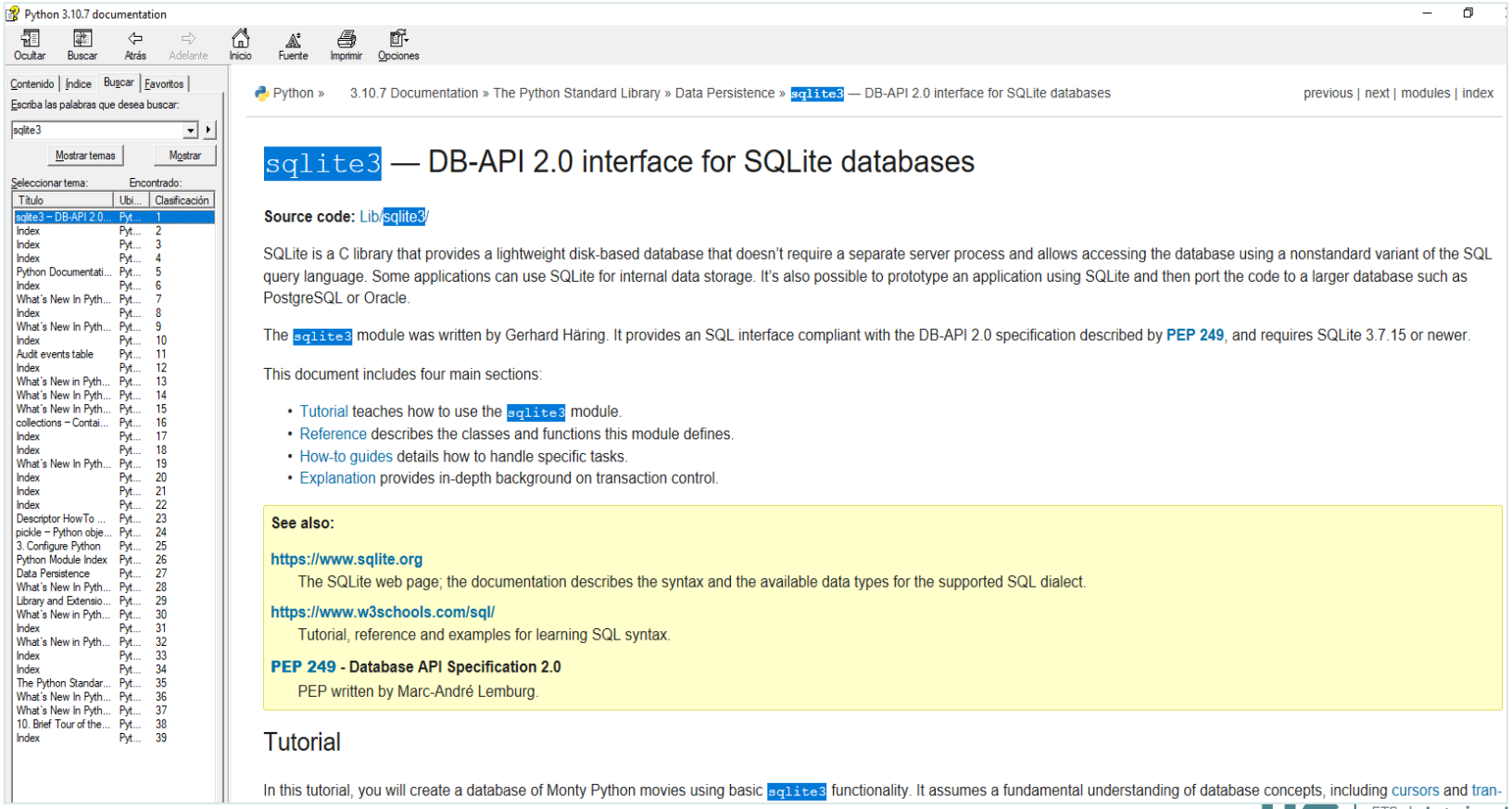
Supongamos que queremos gestionar toda la información relativa a un hotel. Ese problema puede resolverse con una bdd relacional, con un diseño como el siguiente:



Diseñar una bdd es un problema muy interesante, pero se escapa del alcance de este tema. Por ello, aquí haremos un primer acercamiento suponiendo ¡bdd formadas por una única tabla!

Librería sqlite3

sqlite3.- API (Application Programming Interface) Interfaz para bases de datos. Es una librería de C que permite acceder a bdd usando una variante no estándar del lenguaje de consulta SQL (Structured Query Language). El contenido del módulo sqlite3 puede verse en el menú Help de idle3 -> Python Docs, buscar sqlite3.



Algunos métodos de sqlite3 I

Para usar el módulo o librería sqlite3 en un programa de Python lo primero que hay que hacer es importarlo:

```
import sqlite3
```

y crear una conexión que represente una bdd. El método connect crea y abre la bdd que recibe como argumento de entrada dentro del paréntesis si ésta no existe y si existe, la abre.

```
miconexion = sqlite3.connect("nombrebdd.db")
```

A partir de aquí, se dispone de un "canal de comunicación" programa-bdd almacenado en la variable miconexion. Crear un cursor:

```
micursor = miconexion.cursor()
```

y usar el método execute para ejecutar comandos o instrucciones SQLite:

```
micursor.execute("instrucción SQLite")
```

Sustituir la instrucción SQLite concreta que se desee ejecutar en la sentencia anterior. Debe ser una cadena. Para una revisión completa, ver https://www.sqlite.org/lang.html

Algunos métodos de sqlite3 II

micursor.**executemany("instrucción SQLite parametrizada")**

Ejemplo:

filas = [("fila1",), ("fila2",),] #lista de tuplas

micursor.executemany("INSERT INTO nombre_tabla VALUES **(?)**", filas)

micursor.**fetchone()**

Recibe la fila-resultado de una consulta anterior y la devuelve como una tupla.

micursor.**fetchall()**

Recibe el conjunto de filas-resultado de una consulta anterior y las devuelve como una lista de tuplas.

Para guardar en la bdd los cambios realizados en ella:

miconexion.**commit()**

Para cerrar la conexión con la bdd abierta:

miconexion.**close()**

Algunas instrucciones SQLite: Estructura BDD I

micursor.**execute(**" instrucción SQLite "**)**

1. Crear una tabla en una bdd con varios campos de diferentes tipos (sintaxis simplificada):

CREATE TABLE nombre_tabla **(**
nombrecampo1 TIPO_campo1[**NOT NULL**]
[,nombrecampo2 TIPO_campo2 [**NOT NULL**]]
…
PRIMARY KEY(nombrecampo1[, nombrecampo2])
)

REQUERIDO

ÚNICO

A diferencia de Python que no declara tipos explícitos, los campos de las tablas de las bdd los tienen definidos, por tanto, el manejo de la información es diferente.

Algunas instrucciones SQLite: Estructura BDD II

Correspondencia de algunos tipos entre SQLite y Python:

Tipo SQLite	Tipo Python
NULL	None
INTEGER	int
REAL	float
TEXT/CHAR(longitud)	str

2. Eliminar una tabla:

DROP TABLE nombre_tabla

Algunas instrucciones SQLite: INSERTAR

3. Insertar registros en una tabla existente de una bdd:

INSERT INTO nombre_tabla **VALUES**

(valor1_campo1 [,valor1_campo2]…**)**

[,(valor2_campo1 [,valor2_campo2]…)]

…

Una vez creada una tabla en una bdd e insertados algunos registros, se pueden realizar consultas sobre la información almacenada en ella, actualizar datos y borrar registros.

Algunas instrucciones SQLite: ACTUALIZAR, BORRAR

4. Actualizar datos de una tabla:

UPDATE nombre_tabla **SET**

campo1=nuevo_valor1

[,campo2=nuevo_valor2]

…

[**WHERE** condicion]

5. Borrar registros de una tabla:

DELETE FROM nombre_tabla **WHERE** condición

El método commit es imprescindible para guardar los cambios realizados en la bdd después de cualquiera de las instrucciones SQLite vistas, excepto SELECT pues no supone cambios en la bdd.

Algunas instrucciones SQLite: CONSULTAR

6. Seleccionar campos de una tabla de una bdd atendiendo a diferentes criterios, realizando, si es preciso, agrupaciones y ordenaciones atendiendo a uno o varios campos de la tabla, siendo por defecto la ordenación realizada ascendente:

SELECT {* | campo1[, campo2]…} **FROM** nombre_tabla

[**WHERE** condicion]

[**GROUP BY** lista_campos_group_by]

[**HAVING** condición_group_by]

[**ORDER BY** campo1 { [**ASC**] | **DESC**}

[,campo2 { [**ASC**] | **DESC**}]

…]

EJERCICIO 1

Escribe un programa en Python que, usando el módulo sqlite3, cree una bdd hotel.db y una tabla hotel con los campos y valores siguientes:

NIF	Habitacion	Restaurante	Piscina
12345678A	180.0		10.0
12222222A	120.0	65.0	0.00
13333333A	240.0	100.0	20.0
14444444A	300.0	60.0	10.0

a. Usar el método execute.

b. Usar el método executemany.

EJERCICIO 1a

cap8_1a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap8_BDD\BDD_enPYTHON\cap8_1a.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Nov 2022
2  import sqlite3
3  miconexion=sqlite3.connect("BDD\hotel.db")
4  micursor=miconexion.cursor()
5  micursor.execute('''
6      CREATE TABLE hotel (NIF CHAR(9) NOT NULL,
7      habitacion REAL NOT NULL, restaurante REAL,
8      piscina REAL, PRIMARY KEY(NIF))''')
9  micursor.execute('''
10     INSERT  INTO  hotel VALUES
11     ('12345678A',180.0,NULL,10.0)
12    ,('12222222A',120.0,65.0,0.0)
13    ,('13333333A',240.0,100.0,20.0)
14    ,('14444444A',300.0,60.0,10.0)''')
15 miconexion.commit()
16 miconexion.close()
```

EJERCICIO 1b

cap8_1b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap8_BDD\BDD_enPYTHON\cap8_1b.py (3.10.7)

File Edit Format Run Options Window Help

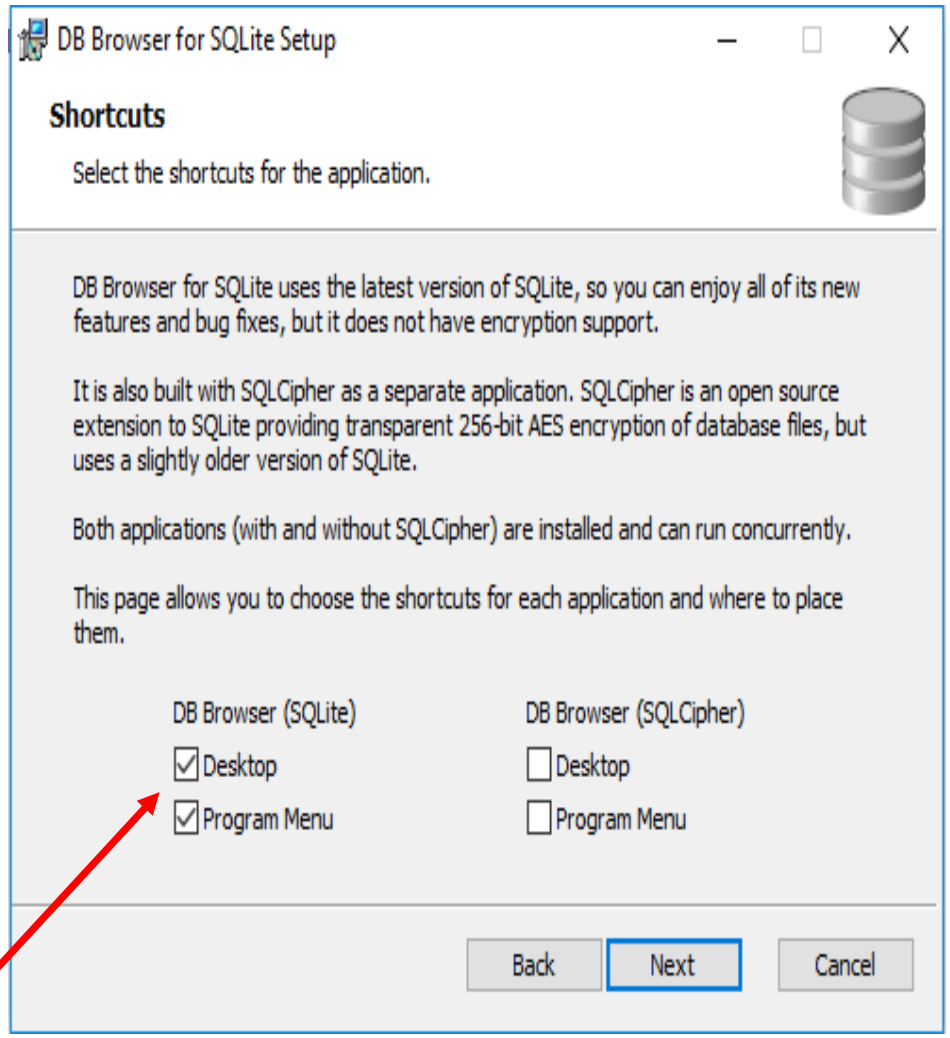
```
1  #PB Nov 2022
2  import sqlite3
3  miconexion=sqlite3.connect("BDD\hotel.db")
4  micursor=miconexion.cursor()
5  #Eliminar la tabla hotel
6  micursor.execute('DROP TABLE hotel')
7  micursor.execute('''CREATE TABLE hotel (
8      id INTEGER, NIF CHAR(9) NOT NULL,
9      habitacion REAL NOT NULL, restaurante REAL,
10     piscina REAL, PRIMARY KEY(id AUTOINCREMENT))''')
11 #Otro modo de insertar
12 lt=[('12345678A',180.0,None,10.0)
13    ,('12222222A',120.0,65.0,0.0)
14    ,('13333333A',240.0,100.0,20.0)
15    ,('14444444A',300.0,60.0,10.0)]
16 micursor.executemany("INSERT  INTO  hotel VALUES (NULL,?,?,?,?)",lt)
17 miconexion.commit()
18 miconexion.close()
```

EJERCICIO 1 II

Al ejecutar el programa anterior en IDLE se crea la bdd hotel.db en la carpeta BDD. Su contenido se puede ver haciendo doble clic si se instala previamente un visor de bases de datos SQLite.

Descargar e instalar https://sqlitebrowser.org/ adecuado según sistema operativo.

Seguir instalación por defecto marcando si se quiere tener la aplicación **DB Browser** para SQLite en el menú de programas y/o en el escritorio de nuestro ordenador.



Repetir el ejercicio usando el entorno gráfico de DB Browser.

EJERCICIO 2 I

Construir un programa en Python que abra la bdd hotel.db del ejercicio1a y realice diferentes manipulaciones sobre ella.

- Consulta1. SELECT * FROM hotel
- Consulta2. SELECT NIF, habitacion FROM hotel ORDER BY NIF
- Consulta3. SELECT * FROM hotel WHERE NIF="13333333A"
- Consulta4. SELECT NIF FROM hotel WHERE habitacion > 200
- Consulta5. SELECT * FROM hotel ORDER BY habitacion DESC
- Actualizar6. UPDATE hotel SET habitacion=359.97 WHERE NIF="12222222A"
- Borrar7. DELETE FROM hotel WHERE NIF="14444444A“

Repetir el ejercicio usando el entorno gráfico de DB Browser.

EJERCICIO 2 II

cap8_2.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap8_BDD\BDD_enPYTHON\cap8_2.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB Nov 2022
import sqlite3
miconexion=sqlite3.connect("BDD\hotel.db")
micursor=miconexion.cursor()
res=micursor.execute('SELECT * FROM hotel')
lt=res.fetchall()
print('\nC1. SELECT * FROM hotel \n',lt)
res=micursor.execute('SELECT NIF, habitacion FROM hotel ORDER BY NIF')
lt=res.fetchall()
print('\nC2. SELECT NIF, habitacion FROM hotel ORDER BY NIF \n',lt)
micursor.execute('SELECT * FROM hotel WHERE NIF="13333333A"')
tp=micursor.fetchone()
print('\nC3. SELECT * FROM hotel WHERE NIF="13333333A" \n',tp)
res=micursor.execute('SELECT NIF FROM hotel WHERE habitacion > 200')
lt=res.fetchall()
print('\nC4. SELECT NIF FROM hotel WHERE habitacion > 200\n',lt)
res=micursor.execute('SELECT * FROM hotel ORDER BY habitacion DESC')
lt=res.fetchall()
print('\nC5. SELECT * FROM hotel ORDER BY habitacion DESC \n',lt)
micursor.execute('UPDATE hotel SET habitacion=359.97 WHERE NIF="12222222A"')
micursor.execute('DELETE FROM hotel WHERE NIF="14444444A"')
res=micursor.execute('SELECT * FROM hotel')
lt=res.fetchall()
print('\nSELECT * FROM hotel \n',lt)
miconexion.commit()
miconexion.close()
```

EJERCICIO 3a I

Construir un programa en Python que abra la bdd hotel.db del ejercicio2 y almacene la información dada en ella (la tabla hotel) en un diccionario, clave el NIF y valor lo demás.

Generar la factura de un cliente escribiendo en un archivo de salida factura_ClienteNIF.txt los gastos de cada concepto y la suma total sin y con el IVA del 21%.

El programa pide por teclado el nombre completo del cliente y su NIF.

La factura tendrá este aspecto:

Factura_Cliente13333333A.txt

Cliente: Pedro Vega Casado
NIF: 13333333A

Gastos estancia (Hab, rest, Pisc) Euros: 240.0+100.0+20.0

Total (sin IVA): 360.0 Euros
Total (con IVA, 21%): 435.6 Euros
Gracias por su visita

EJERCICIO 3a II

cap8_3.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap8_BDD\BDD_enPYTHON\cap8_3.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB Nov 2022
#Abrir y leer la base de datos hotel.db
def leerbdd(nombre):  #funcion
    import sqlite3
    mc=sqlite3.connect(nombre)
    c=mc.cursor()
    c.execute('SELECT * FROM hotel')
    l_tpl=c.fetchall()
#    print(l_tpl)
    mc.close()
    return l_tpl
#.....................................................
#Abrir archivo de salida y generar factura cliente
def escribirtxt(nom,nif,t_g,gt):   #procedimiento
    fs=open("Facturas\FacturaCliente"+nif+".txt","w")
    fs.write("Cliente:"+nom+"\n")
    fs.write("NIF:"+nif+"\n\n")
    fs.write(" "*20+'Gastos estancia (Hab, rest, Pisc) Euros:'
    +str(t_g[0])+" ,"+str(t_g[1])+" ,"+str(t_g[2])+"\n")
    fs.write(" "*20+"Total (sin IVA):"+str(gt)+" Euros \n")
    fs.write(" "*20+"Total (con IVA, 21%):"+str(round(gt*1.21,1))+" Euros \n\n")
    fs.write(" "*20+"Gracias por su visita")
    fs.close()
```

EJERCICIO 3a III

```
24 #.....................................................
25 def main():    #procedimiento
26     l_tpl=leerbdd('BDD\hotel.db')
27 #Volcar la lista de tuplas l_tpl en un diccionario
28     d_datos={ l_tpl[i][0] : l_tpl[i][1:4] for i in range(len(l_tpl))}
29     print('diccionario ',d_datos)
30 #.....................................
31     noexiste=True
32     while noexiste:
33         xnif=input('dame un nif ')
34         if xnif in d_datos:
35             print('Gastos: ',d_datos[xnif])
36             gastosnif=sum(d_datos[xnif])
37             noexiste=False
38     print('Los gastos del cliente ',xnif,' son: ',gastosnif, ' Euros')
39 #.....................................
40     nombre=input("Dame tu nombre completo ")
41     escribirtxt(nombre,xnif,d_datos[xnif],gastosnif)
```

Viable para pocos registros en la tabla hotel.

Se almacena en memoria toda la tabla del hotel (en un diccionario) y se trabaja con el diccionario.

EJERCICIO 3b

Lo mismo que en el apartado a pero almacenando en memoria únicamente la fila o registro que contiene la información del NIF en la tabla hotel.

Válido para un caso realista de una tabla hotel con muchos registros.

EJERCICIO 3b

cap8_3b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap8_BDD\BDD_enPYTHON\cap8_3b.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Nov 2022
2  import sqlite3
3  mc=sqlite3.connect('BDD\hotel.db')
4  c=mc.cursor()
5  noexiste=True
6  while noexiste:
7      xnif=input('dame un nif ')
8      tp=xnif,
9      res=c.execute("SELECT * FROM hotel WHERE NIF=?",tp)
10     tp=res.fetchone()
11     if tp!=None:
12         noexiste=False
13         print(tp)
14 mc.close()
15 gt=0
16 for elem in tp[1:4]:
17     if elem!=None:
18         gt+=elem
19 nombre=input("Dame tu nombre completo ")
20 fs=open("Facturas\FacturaCliente"+xnif+".txt","w")
21 fs.write("Cliente:"+nombre+"\n")
22 fs.write("NIF:"+xnif+"\n\n")
23 fs.write(" "*20+'Gastos estancia (Hab, rest, Pisc) Euros:'
24 +str(tp[1])+" ,"+str(tp[2])+" ,"+str(tp[3])+"\n")
25 fs.write(" "*20+"Total (sin IVA):"+str(gt)+" Euros \n")
26 fs.write(" "*20+"Total (con IVA, 21%):"+str(round(gt*1.21,1))+" Euros \n\n")
27 fs.write(" "*20+"Gracias por su visita")
28 fs.close()
```

Consulta parametrizada

EJERCICIO 4

Realiza el ejercicio 3a pero trabajando con listas, una lista para los NIFS y otra para los gastos.

¿Qué estructura de datos es más adecuada: diccionario o lista?

EJERCICIO 5 a

Construir un programa en Python que cree una bdd alquiler.db con una tabla alquiler con los campos y valores siguientes:

NIF	Matricula	Precio_Dia(€)	N_Dias
12222222R	6789DFG	45.95	3
13333333S	9876PKJ	74.98	2
14444444T	4234ASD	59.99	5
15555555U	6489DNS	45.95	1
16666666V	1234ASD	59.95	4

EJERCICIO 5 b I

Construir un programa en Python que abra la bdd alquiler.db del apartado a.

Pedir el nombre de un cliente por teclado y su NIF y calcular el precio a pagar. Si el NIF no existe, solicitarlo de forma indefinida hasta que exista. Escribir el resultado en pantalla y en un archivo de texto de salida a modo de factura del cliente.

Calcular la ganancia total de la empresa de alquiler escribiendo el resultado en pantalla.

EJERCICIO 5b II

Factura_Cliente16666666V.txt

Nombre: Pedro Rodríguez
NIF: 16666666V
Matrícula: 1234ASD

Precio (sin IVA): 59.95 * 4 días = 239.80 Euros
Precio (con IVA, 10%): = 263.78 Euros
Gracias. Hasta la próxima!

EJERCICIO 6

Construir un programa en python que trabaje igual que en el ejercicio 5, pero teniendo los datos en un archivo de texto y almacenando la información en una lista de sublistas (cada línea en una sublista).