Bases de datos y Python

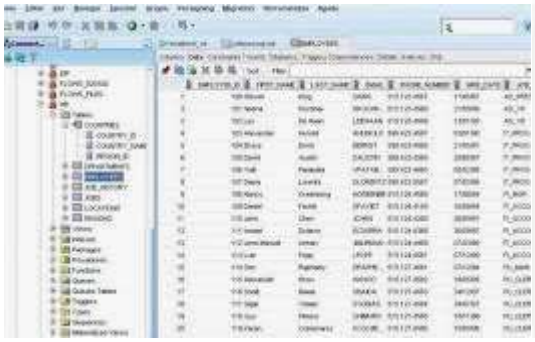
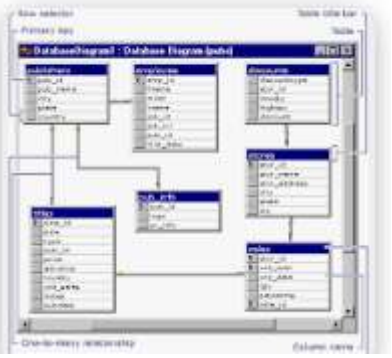
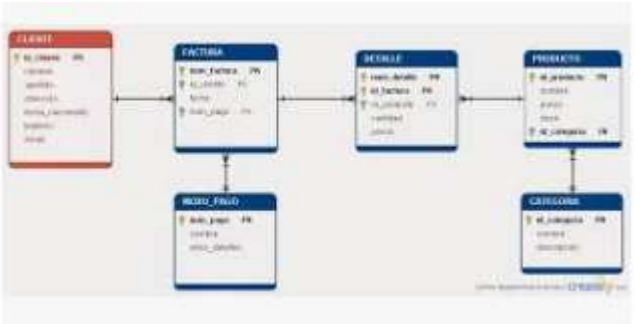
0. Bases de datos y SGBD

   Una Base de Datos (B..D. o BD o b.d. o bd) es un conjunto organizados de datos.

   Ejemplos:



   Algo más informático:

   Conceptos importantes:

   - Tabla: conjunto de filas.
     Fila: conjunto de celdas.
     Tupla: Fila de una tabla (conjunto completo de valores para un registro).
   - Registro: conjunto completo de datos relacionados que se almacenan juntos en una fila de una tabla de base de datos.

- Campo: columna de una tabla. Define el tipo de información de cada registro.
- Celda: valor individual en la intersección de una fila y una columna.
- Relaciones: entre distintos valores de pueden establecerse relaciones. Ejemplo: clientes.id (campo id de la tabla clientes) e id_cliente (campo que identifica a un cliente en otra tabla).
- Tipo de datos: en función de la información que se vaya a almacenar en una columna se usa un tipo de datos u otro. Es un concepto similar al de tipo de datos usado en programación para las variables.
- Consulta: pregunta (en un formato especial) que se hace a una B.D. para obtener datos de una o varias tablas.

Un **Sistema de Gestión de Bases de Datos** (SGBD) o DBMS (Data Base Management System) es un software que permite gestionar las tablas y los datos de una B.D. Más concretamente:

- Crear, modificar y eliminar bases de datos.
- Insertar, consultar, actualizar y borrar datos.
- Controlar el acceso a los datos (seguridad, permisos).
- Garantizar la integridad y consistencia de la información.

Ejemplos de SGBD:

- Microsoft Access
- MySQL
- PostgreSQL
- SQLite
- Oracle Database
- SQL Server
- Maria DB
- Mongo DB

Ejemplo de consulta a una B.D:

```
SELECT * FROM Clientes
```

Donde:

- Select (palabra reservada) que indica la operación a realizar (seleccionar/obtener una información).
- "*" A continuación del SELECT se indican los campos que queremos obtener. Un asterisco significa que los queremos todos.
- From (palabra reservada) para indicar la tabla/s de las que obtendremos los datos de los campos especificados antes del from.

Otro ejemplo:

```
SELECT nombre, apellido FROM Clientes
```

En este caso indicamos los campos concretos (nombre, apellido) que queremos obtener de la tabla Clientes.

1. Bases de datos y Python (un ejemplo práctico)

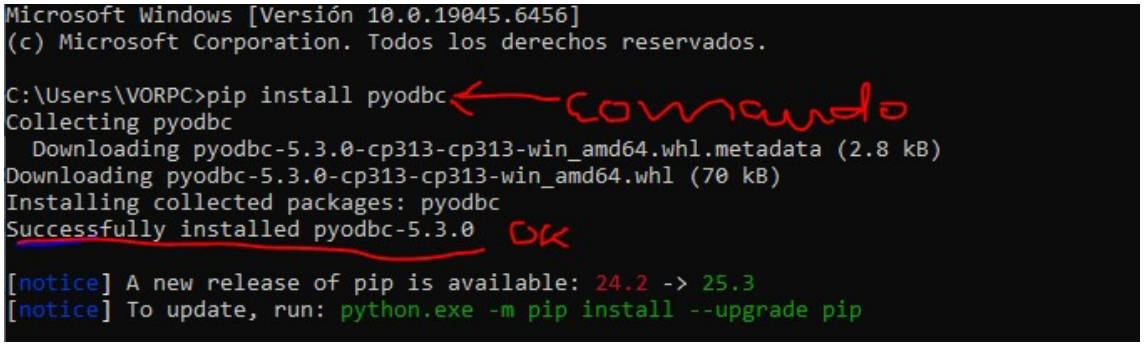
   Pasos previos:

   0. Crear/importar la B.D.
   1. Instalar la librería *pyodbc* que permite la conexión Python – BB.DD. usando ODBC (Open Database Connectivity).
      Se hace desde un ventana de cmd (o PowerShell) -> en principio, no tiene que abrirse como administrador.

   El comando es:

   pip install pyodbc

   Si la cosa funciona…



2. Programa de prueba:

```
1  import pyodbc
2
3  #Ruta al archivo .accdb
4  #Añadimos la letra "r" para que no nos dé problemas el símbolo "\" o backslash
5  db_file = r'C:\python\empresa2526.accdb'
6  conn_str = (
7      r'DRIVER={Microsoft Access Driver (*.mdb, *.accdb)};'
8      f'DBQ={db_file};'
9  )
10
11 # Conexión a la B.D.
12 conn = pyodbc.connect(conn_str)
13 cursor = conn.cursor()
14
15 # Hacemos una consulta en SQL: cogemos todos los campos de la tabla clientes
16 cursor.execute("SELECT * FROM Clientes")
17 for row in cursor.fetchall():
18     print(row)
19
20 # Cerrar el cursor y la conexión
21 cursor.close()
22 conn.close()
```

Línea	Explicación
1	Importamos la librería que conecta Python – B.D. a través de ODBC.
5	Ruta completa al archivo Access (.accdb)
6-9	Creamos la cadena de conexión indicando: el driver (controlador) que vamos a usar (el de Access aquí) y dónde está el archivo de B.D.
12	Establecemos la conexión con la base de datos basándonos en la cadena de conexión creada antes.

13	Creamos el cursor que es el objeto que permite ejecutar las consultas SQL.
16	Ejecutamos el cursor que contiene la consulta SQL en su interior.
17	Recuperamos los resultados con el método fetchall() y recorremos cada fila de la consulta.
18	Imprimimos cada fila de la consulta.
21	Cerramos el cursor con el que hicimos la consulta.
22	Cerramos la conexión a la B.D.

Consideraciones importantes a tener en cuenta:

- Un cursor puede aprovecharse para hacer otras consultas (una vez que ya hemos "utilizado" los datos recuperados anteriormente).
- Tendríamos que usar varios cursores si estuviéramos haciendo varias operaciones en paralelo o si necesitáramos mantener los dos conjuntos de resultados abiertos al mismo tiempo.
- Cerrar el cursor (cuando ya no se va a necesitar) y la conexión (cuando ya no se va a necesitar o al final del programa) es importante para liberar recursos (se ralentizan las aplicaciones y pueden producirse errores en caso de haber muchos cursores o conexiones abiertas. Otro problema que podría darse es que podrían producirse bloqueos en archivos o tablas (otros programas/usuarios no podrían acceder a ellos). También es una buena práctica de programación para evitar errores que podrían ser difíciles de detectar al quedar las conexiones abiertas en segundo plano. Por último: existen motivos de seguridad para no mantener las conexiones abiertas más de lo necesario (la b.d. podría ser compartida o estar en red).