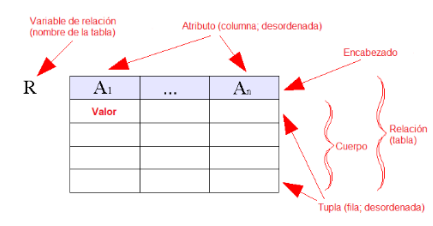
**Introducción a Big Query**

**Bases de Datos Relacionales:** La base de datos relacional es un tipo de base de datos que cumple con el **modelo relacional**.

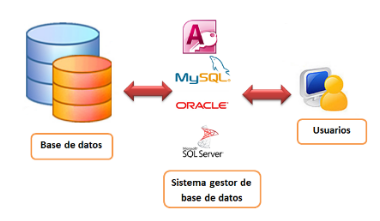
**Base de Datos**: Colección organizada de información estructurada.

En el **modelo relacional**, todos los datos se almacenan en relaciones, donde cada relación es un conjunto de datos; una **tabla**.



Entonces, una base de datos relacional es un **conjunto de tablas** formadas por filas (registros o tuplas) y columnas (atributos o campos). Cada columna tiene un nombre; cada registro tiene un valor para cada atributo.

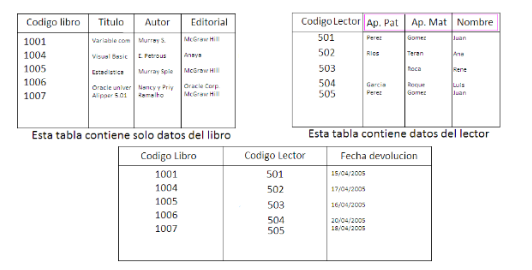
Generalmente, un **Sistema de Gestión de Base de Datos (RDSM – R**elational **D**ataba**s**e **M**anagement **S**ystem**)** controla una base de datos relacional. También se lo llama **motor de bases de datos relacionales.**



Hay diferentes motores de bases de datos con diferente potencia y escalabilidad. Algunos son *Open Source*, tales como Mysql y PostgreSQL:



Supongamos una base de datos con tres tablas:



Cada tabla debe tener por lo menos una columna con valores únicos para cada registro. (**llave primaria** o **Primary Key**). IE: En la tabla libros, el código libro es único, con un valor distinto para cada columna. Si usamos claves, entonces **no podemos tener valores duplicados** en las mismas. Una **tabla padre y un hijo se relacionan** mediante las **claves primarias** y las **claves foráneas** (**Foreign Key**).Las claves foráneas identifican una columna o grupo de columnas en una tabla que se refiere a una columna o grupo de columnas de otra tabla. IE: En la tabla de devoluciones podemos ver el libro, el lector que lo devolvió y cuándo. La columna código libro es una clave foránea, que se relaciona con la clave primaria código libro de la tabla libros (es necesario que el valor de la clave foránea se encuentre en la tabla libros). En forma análoga, la columna Código Lector está relacionada con la clave primaria código Lector de la tabla Lector. Con las claves primarias y foráneas se garantiza la integridad de la base de datos.

**SQL (S**tructured **Q**uery **L**anguage**):** Lenguaje Estándar ANSI/ISO de definición, manipulación y control de Bases de Datos Relacionales. Cuenta con comandos referentes a **Definición de Datos (DDL)**, que sirven para crear la estructura o esquema de la base de datos para modificar dicha estructura; **Manipulación de Datos (DML)** **,** que sirven para insertar datos en la tabla, hacer consultas sobre datos de las tablas, actualizar los datos contenidos en la tabla y eliminar registros de la misma; **Control**  que sirven para Controlar Transacciones como unidades de trabajo lógicas, y también para autorizar privilegios de acceso a las tablas y otros elementos de la base de datos.

La consulta en SQL tiene una sentencia básica:

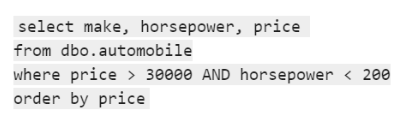
**SELECT** column1, column2 (Se selecciona qué columna queremos)

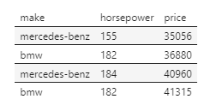
**FROM** TABLtable\_nme (nombre de la tabla de la que queremo extraer información.

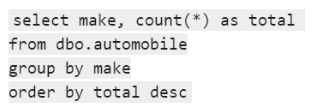
**WHERE** condition (condición de tipo booleana, >, <, etc.)

**ORDER BY** column1, column2. Se indican los campos por los que se ordenan las salidas.

IE:





Tenemos un comando para agrupar en forma similar a como con **groupby de Pandas**. Por ejemplo, vamos a usar el comando **as** para nombrar con un alias a la columna. 



**Bases de Datos No Relacionales / Base de Datos NoSQL:** Sistema de almacenamiento de información que se caracteriza por **no usar el lenguaje SQL** para las consultas. Podrían usarlo en algunos casos, pero no lo hacen como herramienta de consulta; lo hacen como herramienta de apoyo. **No trabajan con estructuras definidas**. Los datos **no se almacenan en tablas**. La información tampoco se organiza en registros o campos. Pueden almacenar documentos, gráficos, archivos con formato xml, entre otros. Están pensada para la gestión de grandes volúmenes de datos y tienen una gran escalabilidad. Hay varios motores de bases de datos no sql, tales como MongoDB, que guarda documentos en formato JSON, permitiendo mantener estructuras anidadas:



**SQLite:** Se trata de un motor de base de datos relacionales para ambientes de laboratorio o prototipos. Es una herramienta de *software libre* que permite almacenar información tanto en dispositivos móviles como en PCs y desarrollos web. Soporta todo tipo de consultas usando el lenguaje SQL.

Ejemplo: Vamos a trabajar con una base de datos SQLite desde Python. Una ventaja de trabajar con SQLite es que no tenemos que instalar cosas nuevas en la PC, el módulo sqlite viene incluido en la librería estándar. Se busca construir y consultar datasets en una forma distinta a los archivos csv:

**En Python:**

import sqlite3

import warnings

warnings.filterwarnings(‘ignore’)

Con el método **connect** podemos establecer una conexión. Una **conexión a base de datos** es la forma en la que un servidor de base de datos y su cliente se comunican. Cliente y servidor no pueden estar en una misma máquina. Entones el cliente usa una conexión base de datos para enviar comandos y recibir respuestas del servidor. En el ejemplo a continuación, desde un programa Python vamos a conectarnos a la base de datos que se crea en memoria:

# conexión a la base de datos:

conn = sqlite3.connect(‘:memory:’)

Una vez creada la conexión, sobre la misma hay que crear un **cursor**, que es una **estructura de control** que usamos para ejecutar comandos SQL y recorrer (y procesar) los registros resultado de una consulta:

# Cursor a la base de datos:

cursor = conn.cursor()

Uno de los métodos más importantes del cursor es el método **.execute**, que permite mediante un comando SQL, crear tablas, insertar, modificar y eliminar registros, y generar consultas también.

Al realizar una consulta, los registros obtenidos quedan en el cursor. Se los llama **conjunto resultado**. Podemos acceder a ellos recorriendo el cursor. Es algo análogo a como podemos recorrer todos los elementos de un vector de a 1, o todos juntos.

Con el método **.fetchone** podemos extraer la siguiente fila del conjunto resultado de la consulta. El output será una secuencia o None en caso de que no queden más datos por consultar.

Con el método **.fetchall** podemos extraer todas las filas que queden sin consumir del conjunto resultado de la consulta. El output será una lista.

Con el método .**close** podemos cerrar el cursor, inhabilitándolo a partir de ese momento.

En el siguiente ejemplo, vamos a crear una tabla, insertar 2 registros, consultarlos, guardarlos en una lista y mostrarlos:

**En Python:**

# Creo la tabla

cursor.execute(“””CREATE TABLE Clientes (ID integer primary key, nombre text, localidad text)”””)

# Inserto dos registros:

cursor.execute(“INSERT INTO Clientes VALUES (1, ‘Perez, Ricardo’, ‘Bogotá’))

cursor.execute(“INSERT INTO Clientes VALUES (2, ‘Artigas, Beatriz, ‘Montevideo’))

# Guardo los cambios

conn.commit()

# Consulto todos los registros:

query = “SELECT \* FROM Clientes”

# Dejo el conjunto resultado en una lista:

clientes = cursor.execute(query).fetchall()

print(type(clientes))

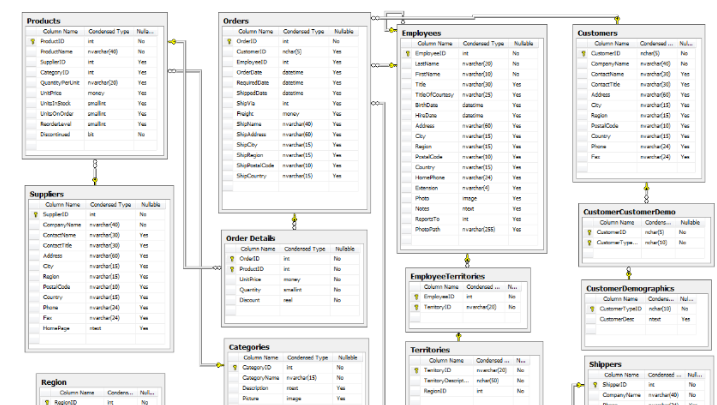
print(clientes)



# Cierro la conexión a la base de datos:

conn.close()

Vamos a trabajar con una base de datos que contenga tablas para poder consultar. IE: Base de Datos Northwind, que es un conjunto de tablas que recrea un esquema de ERP con clientes, pedidos, inventario, compras, proveedores, entre otros:



Nos vamos a conectar desde Python con una base de datos ya creada, simil a un read\_csv de Pandas, pero para acceder a un conjunto de tablas. En ambos casos lo que terminamos obteniendo es un DataFrame con los datos:

**En Python:**

conn = sqlite3.connect(‘../Data/Northwind\_small.sqlite’)

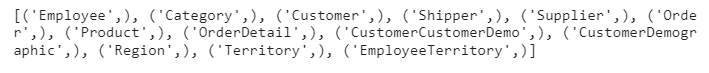
cursor = conn.cursor

Un **schema** de la base de datos es una tabla con la descripción de todas las filas, índices y otros objetos de la base contenidos en la base. Nos sirve consultar primero el schema para conocer a todas las tablas de la base.

query = “SELECT name FROM sqlite\_schema WHERE type = ‘table’’”

tablas = cursor.execute(query).fetchall()

print(tablas)



Entonces, lo que vamos a hacer ahora es generar consultas sobre la base de datos. Vamos a usar el modelo de datos que describe las tablas para resolverlas. Por ejemplo, vamos a traer todos los empleados cuyo apellido empiecen con D. Como resultado vamos a obtener una lista con una tupla en cada elemento, que representará un registro del conjunto resultado.

**En Python:**

query = “SELECT LastName, FirstName from Employee where LastName like ‘D%’ order by LastName”

empleados = cursor.execute(query).fetchall()

empleados



Vamos a transformar la lista en un DataFrame y a agregarle los nombres de las columnas