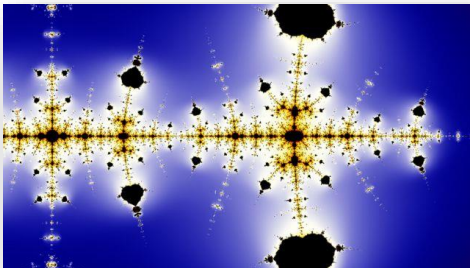


GFPI-F-135 REALIZA EL PROCESO DE LIMPIEZA DE DATOS
LENGUAJE DE CONSULTA ESTRUCTURADAS

ACTIVIDADES POR DESARROLLAR:

1. Conectar desde Python a SQLITE, MySQL, POSTGRESQL, MONGODB
2. Realizar consultas SQL desde Python
3. Entender que es un ORM
4. Aplicar comandos ORM
5. Trabajar con DOCKER con MySQL, POSTGRESQL, MONGODB.



REFLEXIÓN INICIAL

Simple no quiere decir fácil.

Analizar el siguiente caso de estudio llamado “Números Granizo”, que los matemáticos consideran un simple problema sin solución.

Considerado como el agujero negro de las matemáticas, mas conocido como “conjetura de Collatz” propuesto por el alemán

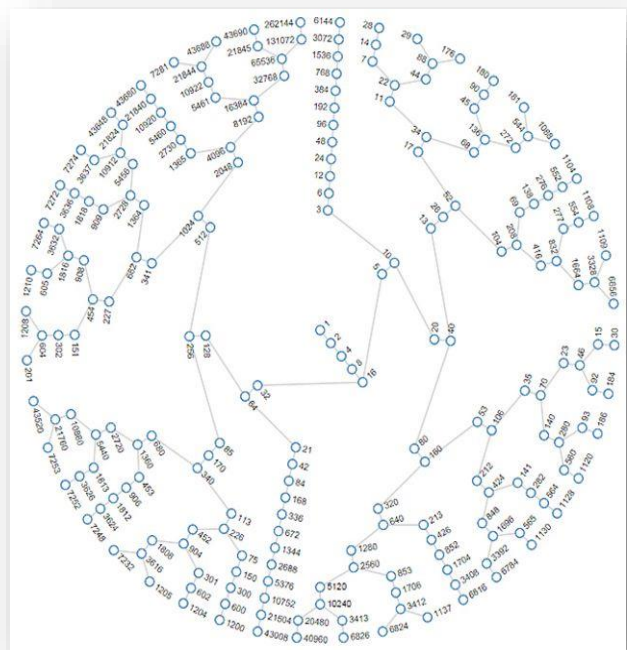
Lothar Collatz en 1937, también conocido como: “conjetura de Ulam”, “problema de Kakutani”, “conjetura de Thwaites”, “algoritmo de Hasse” o “problema de Siracusa”.



El caso consiste en capturar un número entero cualquiera, realizar un ciclo hasta que el número capturado se convierta en uno (1); teniendo en cuenta lo siguiente:

- Si el número es impar, multiplica el número por tres y sumarle 1
- Si el número de par dividirlo por dos

Debo contar cuantos ciclos hago hasta llegar a uno.



**Entregables:**

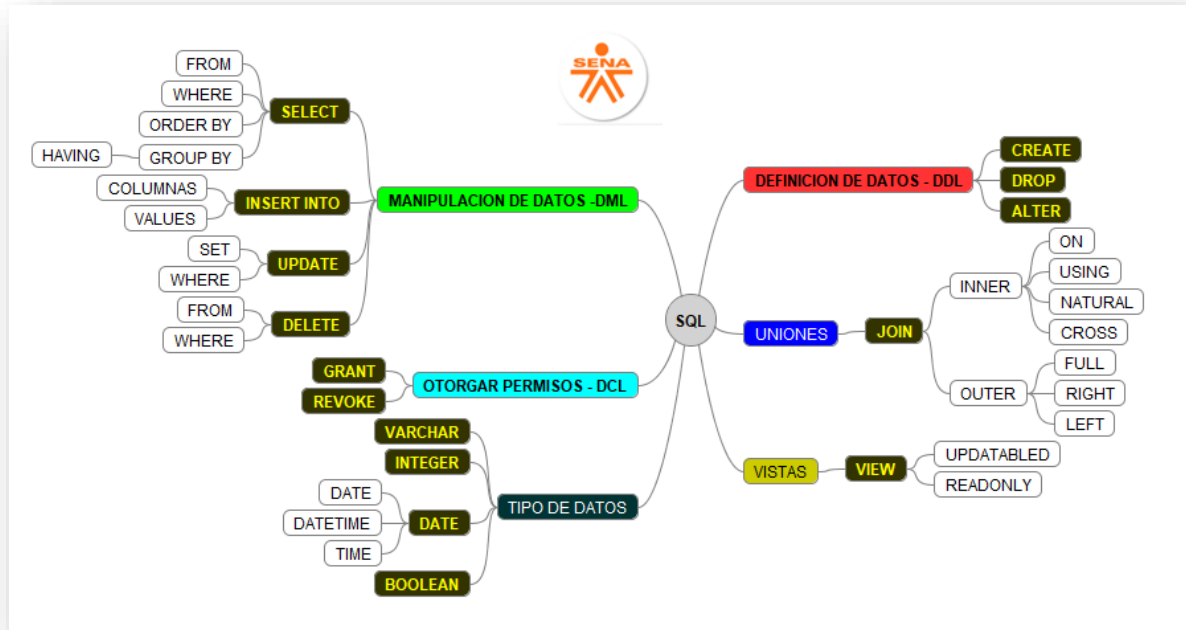
- ¿Qué patrón se observa cuando finaliza con cualquier número que se ingrese?
- Realizar un gráfico de nodos con el número 106.
- Codificar el caso de estudio en Python., documéntelo.
- Crear un DATAFRAME con los datos generados
- Encontrar las siguientes medidas desde Python:
 - Tabla de frecuencias
 - Tabla de frecuencia acumulada
 - Moda
 - Media
 - Mediana
 - Primer cuartil
 - Segundo cuartil
 - Tercer cuartil
 - Cuarto cuartil



No olvide guardar los anteriores productos en el portafolio de evidencias del aprendiz.

"Los *matemáticos sospechan que solucionar la conjetura de Collatz **abrirá nuevos horizontes** y desarrollará nuevas e importantes técnicas en la teoría de los números*", señaló Greg Muller.

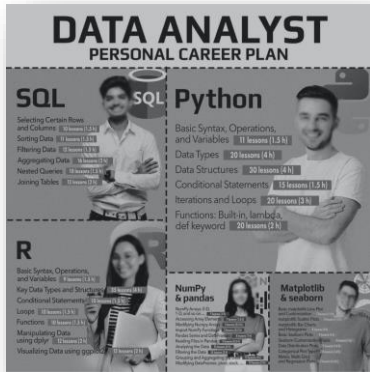
LENGUAJE DE CONSULTA ESTRUCTURADA



¿Qué es SQL?

El lenguaje de consulta estructurada (SQL) es un lenguaje de programación para almacenar y procesar información en una base de datos relacional. Una base de datos relacional almacena información en forma de tabla, con filas y columnas que representan diferentes atributos de datos y las diversas relaciones entre los valores de datos. Puede usar las instrucciones SQL para almacenar, actualizar, eliminar, buscar y recuperar información de la base de datos. También puede usar SQL para mantener y optimizar el rendimiento de la base de datos.¹

¿Por qué es importante SQL?



El lenguaje de consulta estructurada (SQL) es un lenguaje de consulta popular que se usa con frecuencia en todos los tipos de aplicaciones. Los analistas y desarrolladores de datos aprenden y usan SQL porque se integra bien con los diferentes lenguajes de programación. Para crear aplicaciones de procesamiento de datos de alto rendimiento con los principales sistemas de bases de datos SQL como Oracle o MS SQL Server. SQL es muy fácil de aprender, ya que en sus instrucciones se utilizan palabras clave comunes en inglés.

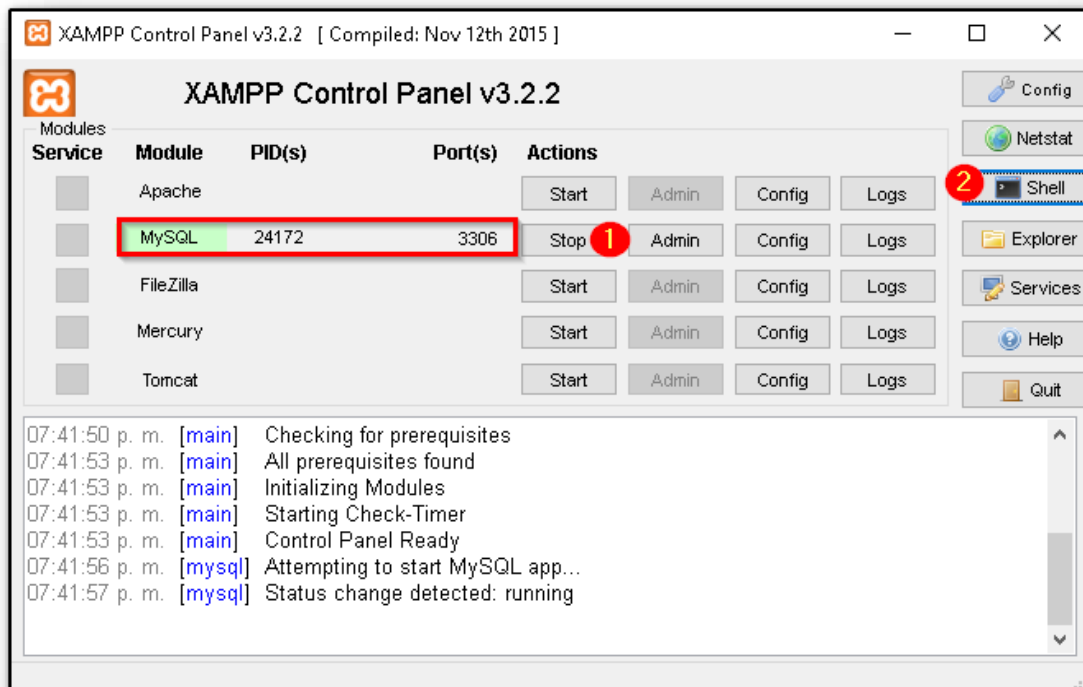
Para realizar la siguiente práctica de laboratorio se recomienda ver el “[Manual de SQL con MYSQL](#)”

PRACTICAS DE LABORATORIO

1. PYTHON+XAMPP+MYSQL



- Instalar XAPP
- Iniciar el servidor de MySQL (1)



- Ingresar a la consola de MySQL (2)



```
# mysql -u root -p
Enter password: ****
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 12
Server version: 10.1.37-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

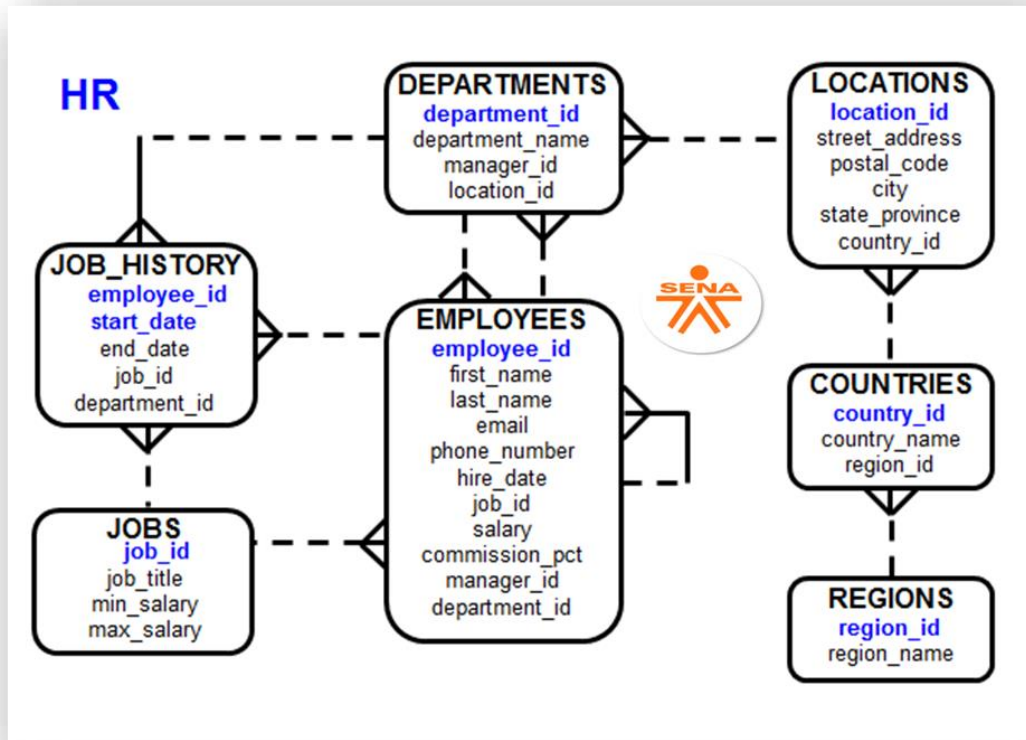
- Descargar el script de creación de la base de datos [HR](#)
- Copiar el script al block de notas.

```
422 líneas (402 sloc) | 24.2 KB
1  -- -----
2  -- Host:                127.0.0.1
3  -- Versión del servidor: 10.1.37-MariaDB - mariadb.org binary distribution
4  -- SO del servidor:     Win32
5  -- HeidiSQL Versión:    10.2.0.5599
6  -- -----
7
8  /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
9  /*!40101 SET NAMES utf8 */;
10 /*!50503 SET NAMES utf8mb4 */;
```

- Pegar desde el portapapeles a la consola Mysql

```
MariaDB [hr]>
MariaDB [hr]> /*!40101 SET SQL_MODE=IFNULL(@OLD_SQL_MODE, '') */;
Query OK, 0 rows affected (0.00 sec)
MariaDB [hr]> /*!40014 SET FOREIGN_KEY_CHECKS=IF(@OLD_FOREIGN_KEY_CHECKS IS NULL, 1, @OLD_FOREIGN_KEY_CHECKS) */;
Query OK, 0 rows affected (0.00 sec)
MariaDB [hr]> /*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
Query OK, 0 rows affected (0.00 sec)
MariaDB [hr]>
MariaDB [hr]> show tables;
+-----+
| Tables_in_hr |
+-----+
| countries     |
| departments   |
| emp_details_view |
| employees     |
| job_history   |
| jobs          |
| locations     |
| regions       |
+-----+
8 rows in set (0.00 sec)
MariaDB [hr]> _
```

- Esquema HR creado



1.1. Conectar desde Python a MySQL



- Instalar el driver de MySQL para Python

```
# pip3 install mysql-connector-python
```

- Conectando a la base de datos
- Importar las bibliotecas necesarias

```
import mysql.connector  
from mysql.connector import Error  
import pandas as pd
```

- Realizar la conexión hacia el servidor mysql

```
connection=mysql.connector.connect(  
    host="localhost",  
    user="root",  
    passwd="root",  
    database="neptuno",  
    port=3306)
```

- Crear un cursor para interactuar con mysql

```
cursor=connection.cursor()
```


- Obtener todas las filas de la tabla employees

```
Query = "select * from employees  
Cursor.execute(Query)  
Resultado = pd.DataFrame(cursor.fetchall())  
print(Resultado)
```

1.2 PRACTICA MIGRANDO DESDE EXCEL CON PYTHON



Instalar los siguientes drivers:

```
Pip install mysql-connector-python  
pip install xlrd
```

Importar las bibliotecas necesarias

```
import xlrd  
import mysql.connector  
from mysql.connector import Error  
import pandas as pd
```



Realizar la conexión hacia el servidor mysql

```
connection=mysql.connector.connect(  
    host="localhost",  
    user="root",  
    passwd="root",  
    database="neptuno",  
    port=3306)
```

Crear un cursor para interactuar con mysql

```
cursor=connection.cursor()
```

Crear la tabla si no existe

```
query="CREATE TABLE IF NOT EXISTS `Categorias` (idcategoria integer primary key,  
NombreCategoria text, Descripcion text)"
```

Ejecutar la consulta

```
cursor.execute(query)
```

1.2.1 INSERTAR REGISTROS EN LA TABLA CATEGORIAS

Cargar la hoja "2,-Categorias" del archivo Excel "Neptuno.xls"

```
x=pd.read_excel("c:/Borrar/Neptuno.xls",sheet_name="2,-Categorias")
```

Se cargan los nombres de las columnas

```
k=[]  
for i in x:  
    k.append(i)
```

Cantidad de filas

```
j=len(x)
```

Se recorre las filas para construir la instrucción insert y se ejecuta para crear el registro

```
i=0  
while i<j:  
    sql="insert into Categorias(idcategoria,NombreCategoria,Descripcion)  
values(%s,%s,%s)"  
    sql1=(int(z0[i]),z1[i],z2[i])  
    print(sql1)  
    i=i+1  
    cursor.execute(sql,sql1)  
    connection.commit()
```

Se cierra la conexión al servidor

```
connection.close()
```

2. Conectando Python con MongoDB



Conectar al servidor MongoDB local

```
import pymongo
from pymongo import MongoClient

CONNECTION_STRING = "mongodb://localhost:27017/"

# Create a connection using MongoClient. You can import MongoClient or use pymongo.MongoClient
client = MongoClient(CONNECTION_STRING)
```

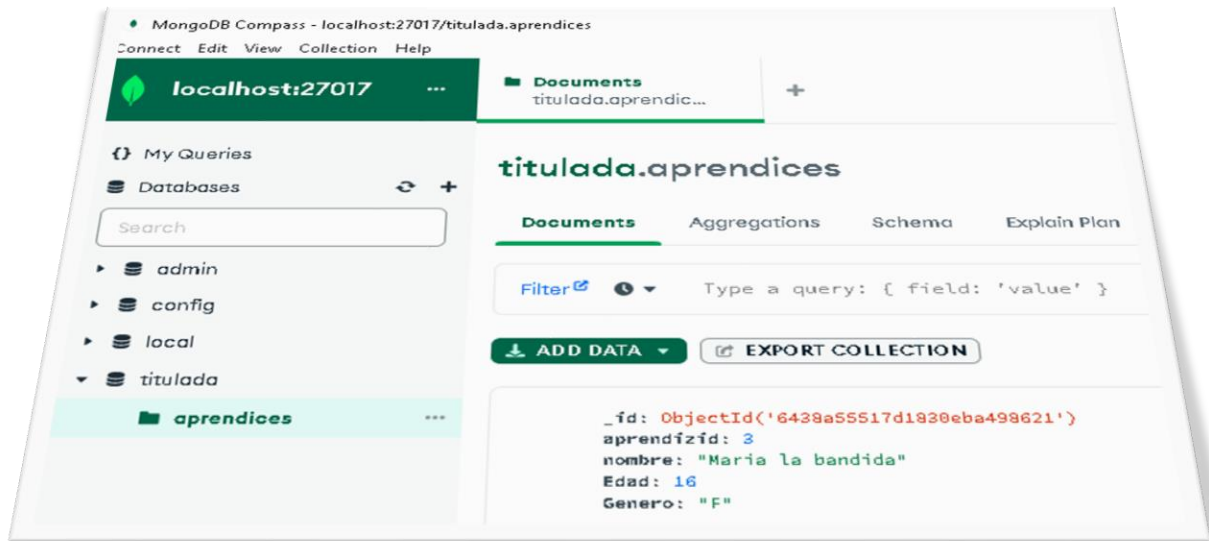
Crear la base de datos llamada "titulada" y la colección "aprendices"

```
dbname = client["titulada"]
coleccion=dbname["aprendices"]
```

Insertar un documento

```
coleccion.insert_one({"aprendizid":3,"nombre":"Maria la  
bandida","Edad":16,"Genero":"F"})
```

Observar en MongoDB Compas



Inserta más de un documento

```
coleccion.insert_many([{"aprendizid":4,"nombre":"Rosa Rico","Edad":36,"Genero":"F"},  
                        {"aprendizid":5,"nombre":"Simon Tolomeo","Edad":55,"Genero":"M"}])
```

Mostramos todos los documentos de la colección

```
a=coleccion.find()  
for x in a:  
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

Mostrar un registro en especial

```
a=coleccion.find({"aprendizid":3})  
for x in a:  
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```



Mostramos todos los documentos ordenados por la edad de forma descendente

```
a=coleccion.find().sort("Edad",-1)
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

Mostramos todos los documentos ordenados por la edad de forma ascendente

```
a=coleccion.find().sort("Edad",1)
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

Cambiar el nombre del aprendiz 3 por "García Rovira"

```
coleccion.update_one({"aprendizid":3},
    {"$set":{"nombre":"García Rovira"}})
```

Cambiar el género por "M" a los que el nombre comience por "Ro"

```
myquery = { "nombre": { "$regex": "^Ro" } }
newvalues = { "$set": { "Genero": "M" } }

coleccion.update_many(myquery,newvalues)
```

Listar solamente el mayor de los aprendices



```
a=coleccion.find().sort("Edad",-1).limit(1)
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

Borrar el aprendiz cuyo id es igual a 5

```
coleccion.delete_one({"aprendizid":5})
```

Borrar la colección aprendiz

```
coleccion.drop()
```

3. Object Relational Mapping (ORM)

¿Qué es un ORM?

Un ORM es un modelo de programación que permite mapear las estructuras de una base de datos, sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de nuestras aplicaciones.



Las estructuras de la base de datos relacional quedan vinculadas con las entidades lógicas o base de datos virtual definida en el ORM, de tal modo que las acciones CRUD (Create, Read, Update, Delete) a ejecutar sobre la base de datos física se realizan de forma indirecta por medio del ORM.

¿Cuáles son los ORM más usados?

- Active Record (Ruby)
- Eloquent (PHP)
- Peewee (Python)
- SQLAlchemy (Python)
- Entity Framework (C#)
- Hibernate (Java)

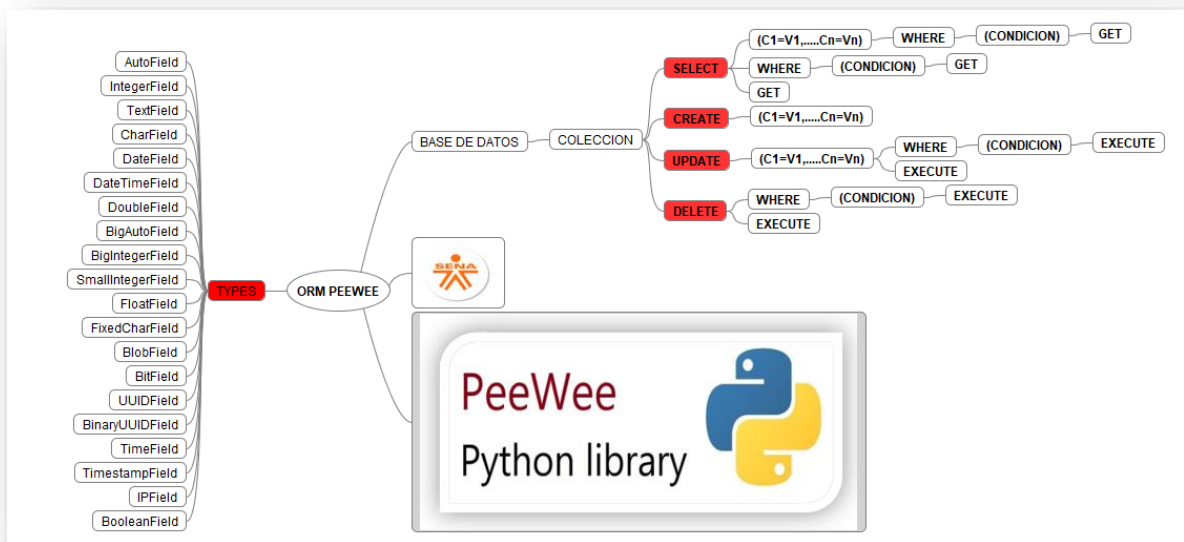
Ventajas y Desventajas de un ORM

Ventajas de usar un ORM

- **Facilidad** y velocidad de uso
- **Abstracción** de la base de datos usada.
- **Seguridad** de la capa de acceso a datos contra ataques.
- **Reutilización.** Nos permite utilizar los métodos de un objeto de datos desde distintas zonas de la aplicación, incluso desde aplicaciones distintas.
- **Mantenimiento del código.** Nos facilita el mantenimiento del código debido a la correcta ordenación de la capa de datos, haciendo que el mantenimiento del código sea mucho más sencillo.

Desventajas

- Lentitud en volúmenes de datos. entornos con gran carga poner una capa más en el proceso puede mermar el rendimiento. Es decir, en algunos casos es más rápido utilizar SQL puro.
- Aprender el nuevo lenguaje del ORM.



3.1 Practica ORM con MySQL y PEEWEE



- Instalar el driver ORM

```
pip3 install peewee
```

- Conectar a una base de datos MySQL

```
from peewee import *
import pandas as pd
cnx=MySQLDatabase("hr",host="localhost",
                  port=3306,
                  user="root",
                  password="root")
class BaseModel(Model):

    class Meta:
        database=cnx

class employees(BaseModel):
    employee_id = IntegerField(primary_key=True)
    first_name = TextField(20)
    last_name = TextField(25)
    email = TextField(25)
    phone_number = TextField()
    hire_date = DateField()
    job_id = TextField(10)
    salary = DoubleField(8,2)
    commission_pct = DoubleField(2,2)
    manager_id = IntegerField(11)
    department_id = IntegerField(11)
```

- Seleccionar un registro



```
a=employees.select().where(employees.employee_id==100).get()
print(a.first_name+' '+a.last_name)
```

- Para actualizar un registro:

```
employees.update({employees.last_name:"Galindo"}).where(
    employees.employee_id==100).execute()
```

- Para insertar un registro:

```
employees.create(employee_id=300,first_name="Rosa",
    last_name="Melano",job_id="AD_ASST")
```

- Para eliminar un registro:

```
employees.delete().where(employees.employee_id==300).execute()
```

- Listar todos los registros:

```
a=employees.select()
for i in a:
    print(i.first_name+' '+i.last_name)
```

- Listar los registros con un filtro:

```
a=employees.select().where(employees.department_id==90)
for i in a:
    print(i.first_name+' '+i.last_name)
```

- Listar registros utilizando expresiones regulares y ordenadas:



```
a=employees.select().where(
    employees.last_name.iregexp('^C' )).order_by(employees.last_name .desc())
for i in a:
    print(i.first_name+' '+i.last_name)
```

3.2 PRACTICA MIGRAR DESDE EXCEL A MYSQL CON ORM



Instalar el driver peewee

```
pip install peewee
```

Importando las librerías necesarias

```
from peewee import *  
import xlrd  
import pandas as pd
```

Realizando la conexión y la clase BaseModel hacia el servidor MYSQL

```
cnx=MySQLDatabase('neptuno', host='localhost',  
                  port=3306,user='root', password='root')  
  
class BaseModel(Model):  
  
    class Meta:  
        database=cnx
```

Se crea el modelo de la tabla "categorías"

```
class categorias(BaseModel):  
    NombreCategoria=TextField()  
    Descripcion=TextField()
```



Se realiza la conexión

```
cnx.connect()
```

Se crea la tabla

```
cnx.create_tables([categorias,])
```

Extraer los datos del Excel

```
x=pd.read_excel("c:/Borrar/Neptuno.xls",sheet_name="2,-Categorias")
```

Se cargan los datos en las listas

```
k=[]  
for i in x:  
    k.append(i)  
  
z0=x[k[0]]  
z1=x[k[1]]  
z2=x[k[2]]
```

Se obtiene el numero de filas a insertar

```
j=len(x)
```

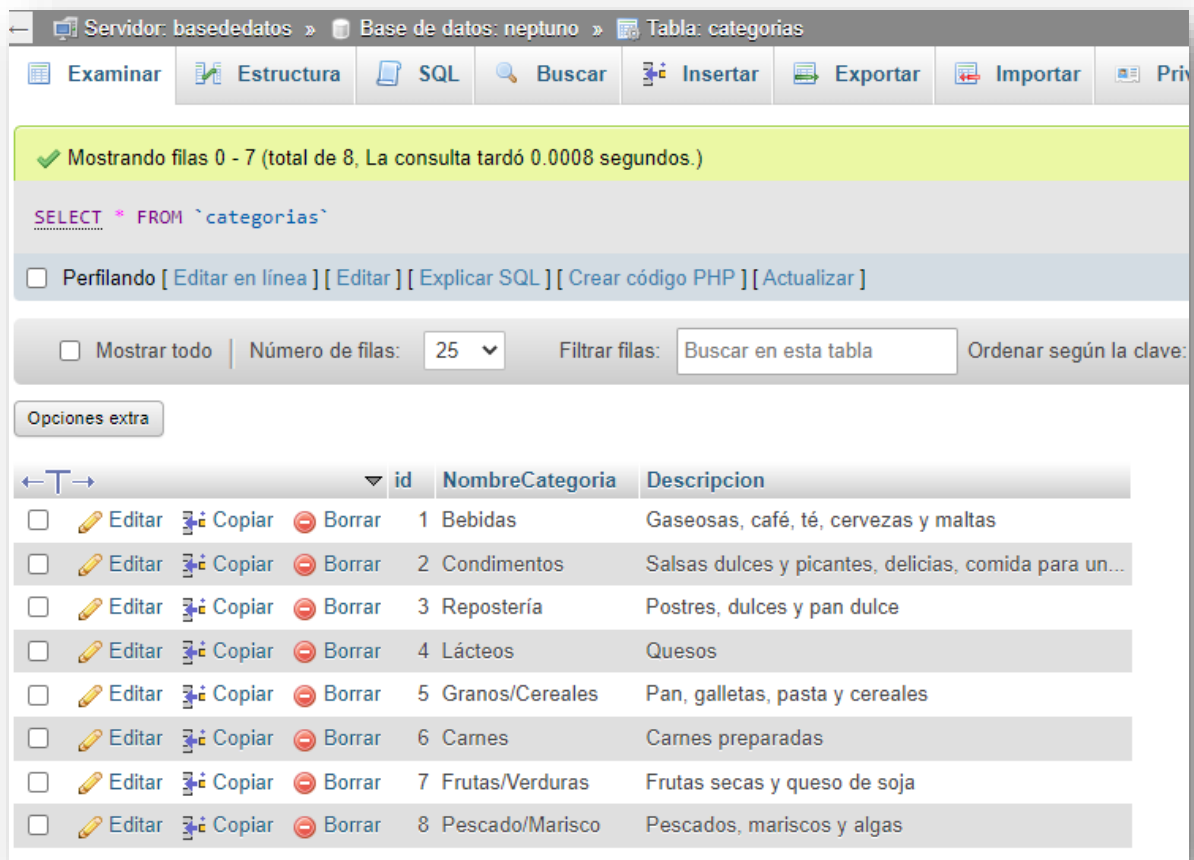
Se recorre las filas de las listas y se insertan en la tabla categorías

```
i=0
while i<j:
    categorias.create(idcategoria=int(z0[i]),
                     NombreCategoria=z1[i],Descripcion=z2[i])
    i=i+1
```

Se cierra la conexión

```
cnx.close()
```

Se verifica desde PhpMyAdmin



Mostrando filas 0 - 7 (total de 8, La consulta tardó 0.0008 segundos.)

`SELECT * FROM `categorias``

☐ Perfilando [[Editar en línea](#)] [[Editar](#)] [[Explicar SQL](#)] [[Crear código PHP](#)] [[Actualizar](#)]

☐ Mostrar todo | Número de filas: 25 | Filtrar filas: Ordenar según la clave:

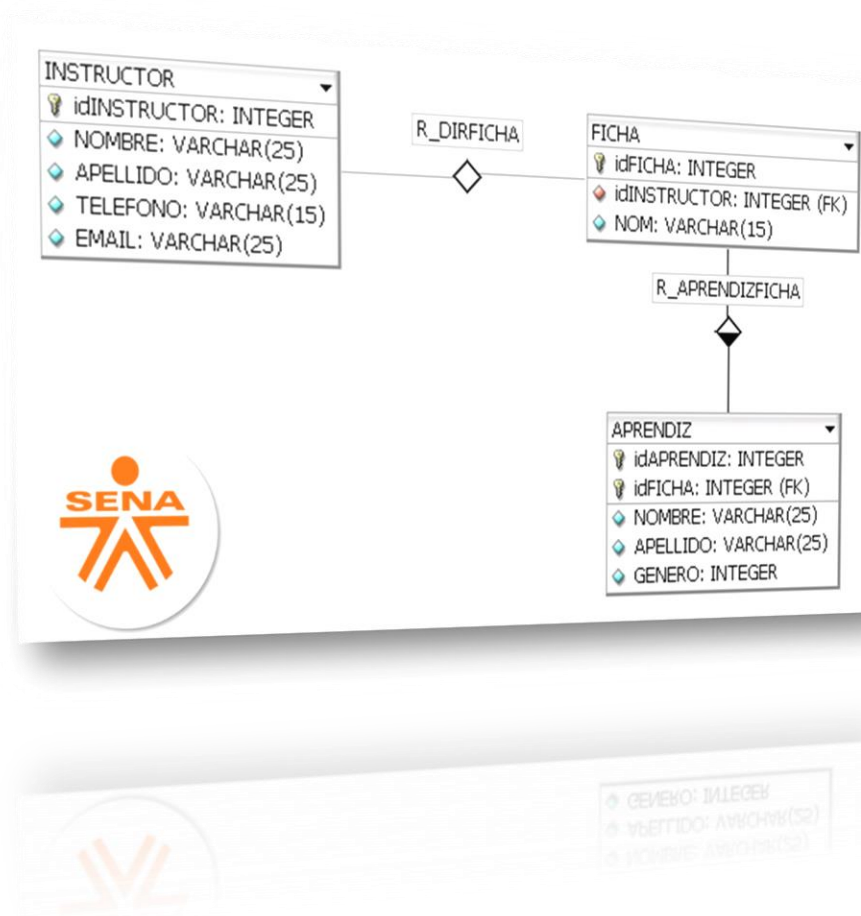
Opciones extra

				id	NombreCategoria	Descripcion
<input type="checkbox"/>	Editar	Copiar	Borrar	1	Bebidas	Gaseosas, café, té, cervezas y maltas
<input type="checkbox"/>	Editar	Copiar	Borrar	2	Condimentos	Salsas dulces y picantes, delicias, comida para un...
<input type="checkbox"/>	Editar	Copiar	Borrar	3	Repostería	Postres, dulces y pan dulce
<input type="checkbox"/>	Editar	Copiar	Borrar	4	Lácteos	Quesos
<input type="checkbox"/>	Editar	Copiar	Borrar	5	Granos/Cereales	Pan, galletas, pasta y cereales
<input type="checkbox"/>	Editar	Copiar	Borrar	6	Carnes	Carnes preparadas
<input type="checkbox"/>	Editar	Copiar	Borrar	7	Frutas/Verduras	Frutas secas y queso de soja
<input type="checkbox"/>	Editar	Copiar	Borrar	8	Pescado/Marisco	Pescados, mariscos y algas

3.3 PRACTICA ORM CON SQLITE



Realizar el siguiente modelo en SQLite :



- Conectando a una base de datos SQLite llamada "Actividad.db"


```
from peewee import *  
#SqliteDatabase, AutoField, CharField, DateField, ForeignKeyField, Model  
  
db = SqliteDatabase('Actividad.db')
```

- Creando la tabla Instructor

```
class Instructor(Model):  
    idINSTRUCTOR = AutoField()  
    NOMBRE = CharField(25)  
    APELLIDO = CharField(25)  
    TELEFONO = CharField(15)  
    EMAIL = CharField(unique=True)  
  
class Meta:  
    database = db
```

- Creando tabla Ficha con la llave foránea a instructor.

```
class Ficha(Model):  
    idFicha = AutoField()  
    NOMBRE = CharField(25)  
    idINSTRUCTOR = ForeignKeyField(Instructor)  
  
class Meta:  
    database = db
```

- Creando la tabla Aprendiz con llave primaria compuesta y llave foránea a Ficha

```
class Aprendiz(Model):
    idAprendiz = IntegerField()
    NOMBRE = CharField(25)
    APELLIDO = CharField(25)
    GENERO = IntegerField()
    idFicha = ForeignKeyField(Ficha)

class Meta:
    database = db
    primary_key = CompositeKey('idAprendiz','idFicha')
```

- Conectar la base de datos:

```
db.conect()
```

- Ingresar datos

```
Instructor.create(NOMBRE="Jose", APELLIDO="Fegasu",TELEFONO="6015941301",
EMAIL=jgalindos@sena.edu.co)
```

- Ingresar los siguientes instructores:

NOMBRE	APELLIDO	TELEFONO	EMAIL
Uldarico	Andrade	5941301	uandrade@sena.edu.co
Erick	Granados	31244455666	egranados@sena.edu.co
Carlos	Patiño	3106665544	cpatiño@sena.edu.co



- Ingresar las siguientes fichas:

IDINSTRUCTOR	NOMBRE
Erick Granados	2687365
Jose Fegasu	2671743
Carlos Patiño	2465298

- Ingresar los siguientes aprendices:

IDAPRENDIZ	IDFICHA	NOMBRE	APELLIDO	GENERO
1010113345	2687365	ELVER	GALINDO	MASCULINO
1011223456	2671743	MARIA	GARCIA	FEMENINO
78456723	2687365	JUANITA	CORZO	FEMENINO

3.4 MIGRAR DESDE EXCEL CON ORM HACIA MONGODB



Instalar los drivers necesarios

```
python -m pip install pymongo
```

Declarar las librerías a utilizar

```
from peewee import *  
from pymongo import MongoClient  
import pandas as pd
```

Conectar al servidor MongoDB

```
cnx = MongoClient("mongodb://localhost:27017/")
```

Crear la base de datos "Neptuno" y la colección "categorías"

```
bd=cnx['Neptuno']  
coleccion=bd['categorias']
```

Extraer los datos del Excel

```
x=pd.read_excel("c:/Borrar/Neptuno.xls",sheet_name="2,-Categorias")
```



Se cargan los datos en las listas

```
k=[]  
for i in x:  
    k.append(i)  
  
z0=x[k[0]]  
z1=x[k[1]]  
z2=x[k[2]]
```

Se obtiene el número de filas a insertar

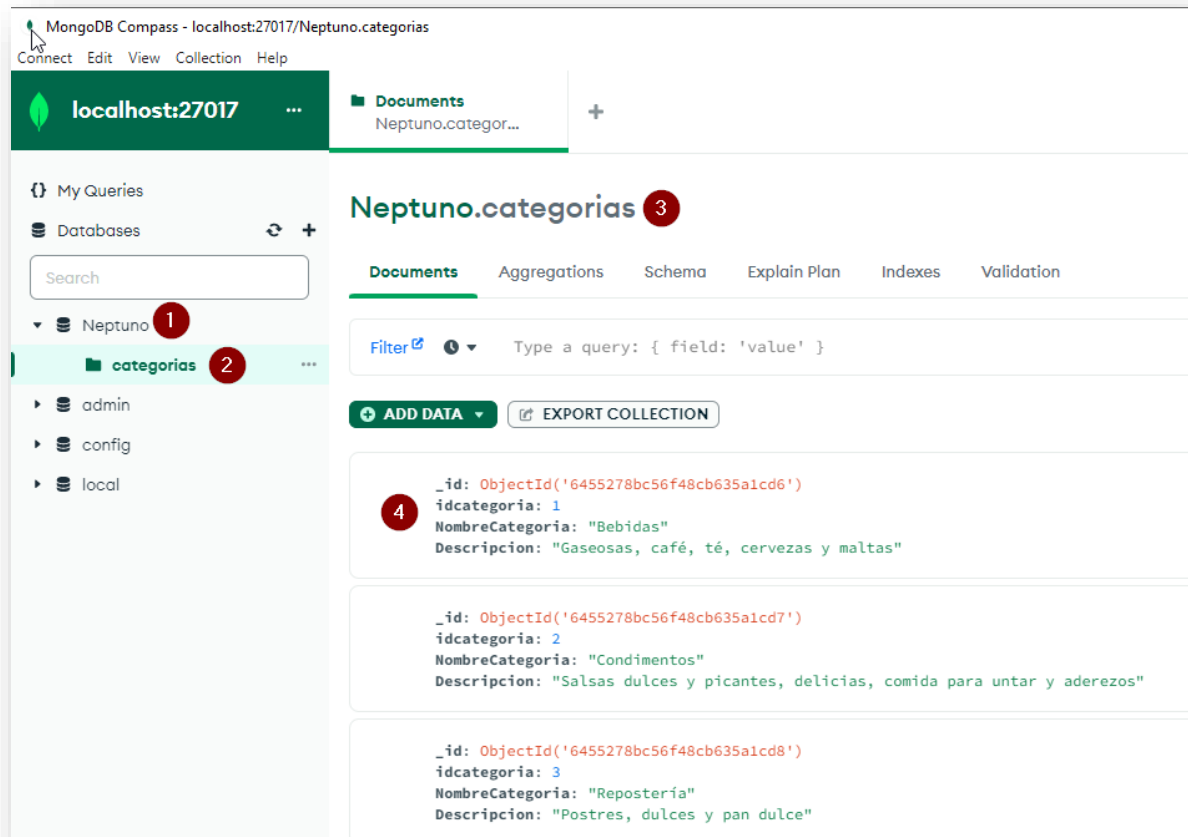
```
j=len(x)
```

Se recorre las filas de las listas y se insertan en la tabla categorías

```
i=0  
while i<j:  
    coleccion.insert_one({"idcategoria":int(z0[i]) ,  
                          "NombreCategoria":z1 [i],"Descripcion":z2[i]})  
    i=i+1
```



Se verifica en "MongoDB Compass"



3.5 PRACTICA ORM CON POSTGRESQL



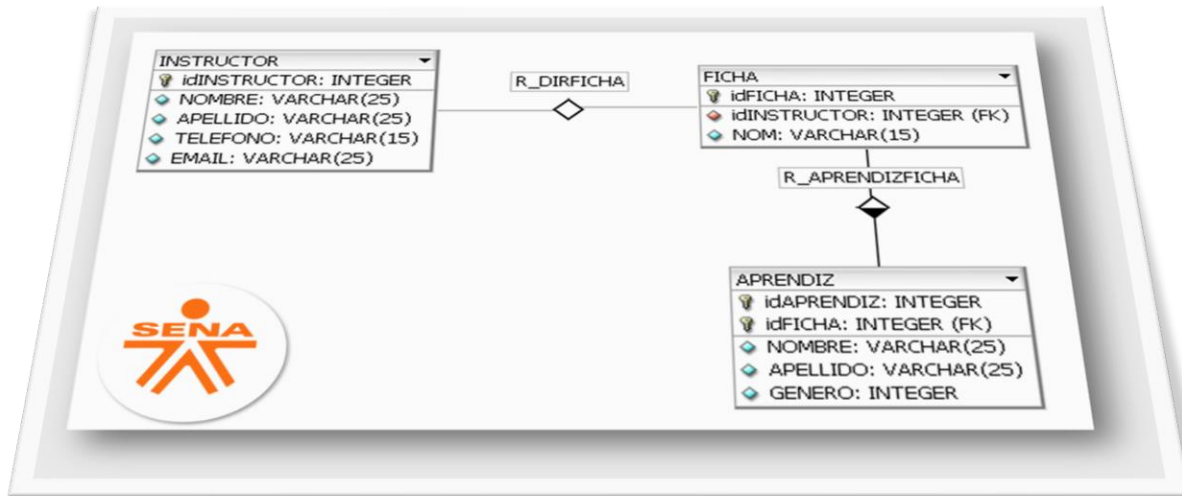
Descargar [PostgreSQL](#) e instálelo para realizar la practica con el modelo relacional anterior.

- Instalar el driver para trabajar con PostgreSQL

```
pip install psycopg2  
pip3 install psycopg2
```

Conectar a una base de datos PostgreSQL

```
import psycopg2  
  
conn = psycopg2.connect(database="Actividad",  
                        host="localhost",  
                        user="postgres",  
                        password="db_postgres",  
                        port= 5432)
```



- Ingresar los siguientes instructores:

NOMBRE	APELLIDO	TELEFONO	EMAIL
Uldarico	Andrade	5941301	uandrade@sena.edu.co
Erick	Granados	31244455666	egranados@sena.edu.co
Carlos	Patiño	3106665544	cpatiño@sena.edu.co

- Ingresar las siguientes fichas:

IDINSTRUCTOR	NOMBRE
Erick Granados	2687365
Jose Fegasu	2671743
Carlos Patiño	2465298

- Ingresar los siguientes aprendices:



IDAPRENDIZ	IDFICHA	NOMBRE	APELLIDO	GENERO
1010113345	2687365	ELVER	GALINDO	MASCULINO
1011223456	2671743	MARIA	GARCIA	FEMENINO
78456723	2687365	JUANITA	CORZO	FEMENINO

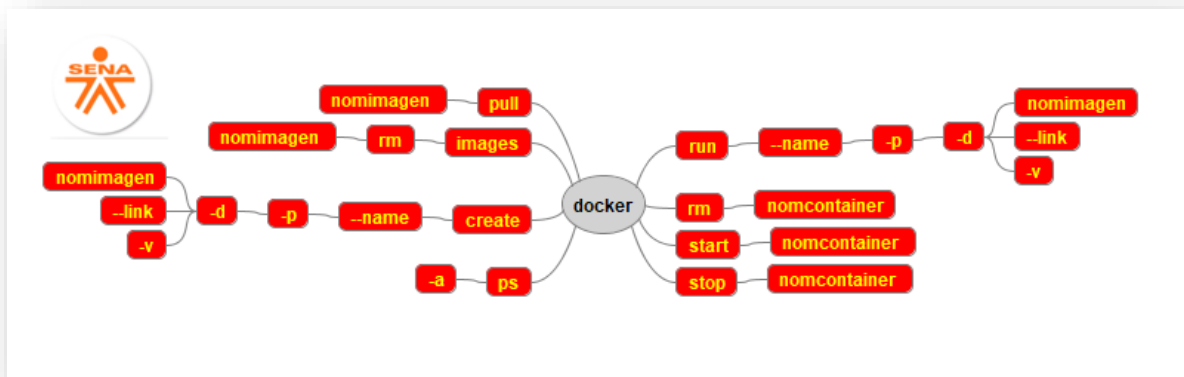
4. ¿QUÉ ES DOCKER?



Es un software de código abierto utilizado para desplegar aplicaciones dentro de contenedores virtuales. La contenerización permite que varias aplicaciones funcionen en diferentes entornos complejos. Por ejemplo, Docker permite ejecutar el sistema de gestión de contenidos WordPress en sistemas Windows, Linux y macOS sin ningún problema.

La principal diferencia con las máquinas virtuales es que los contenedores Docker comparten el sistema operativo del anfitrión, mientras que las máquinas virtuales también tienen un sistema operativo invitado que se ejecuta sobre el sistema anfitrión. Este método de funcionamiento afecta al rendimiento, las necesidades de hardware y la compatibilidad con el sistema operativo. (<https://www.hostinger.es/tutoriales/que-es-docker#:~:text=Docker%20es%20un%20software%20de%20c%C3%B3digo%20abierto%20utilizado,sistemas%20Windows%2C%20Linux%20y%20macOS%20sin%20ning%C3%BAn%20problema.,s.f.>)

COMANDOS DOCKER.



4.1 PRACTICA CON MYSQL Y DOCKER



- Instalar [Docker Desktop](#) para poder desarrollar esta práctica.
- Crear el contenedor con la imagen de Mariadb (el comando debe ser en solo una línea)

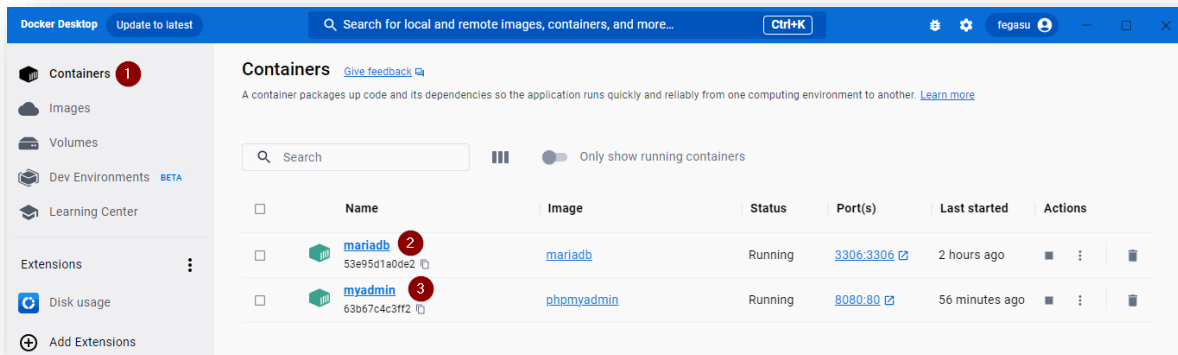
```
docker run -p 3306:3306 --name basededatos -v c:/Docker/mysql:/var/lib/mysql -e MYSQL_ROOT_PASSWORD=root -d mysql:5.7
```

Crear un contenedor para PHP

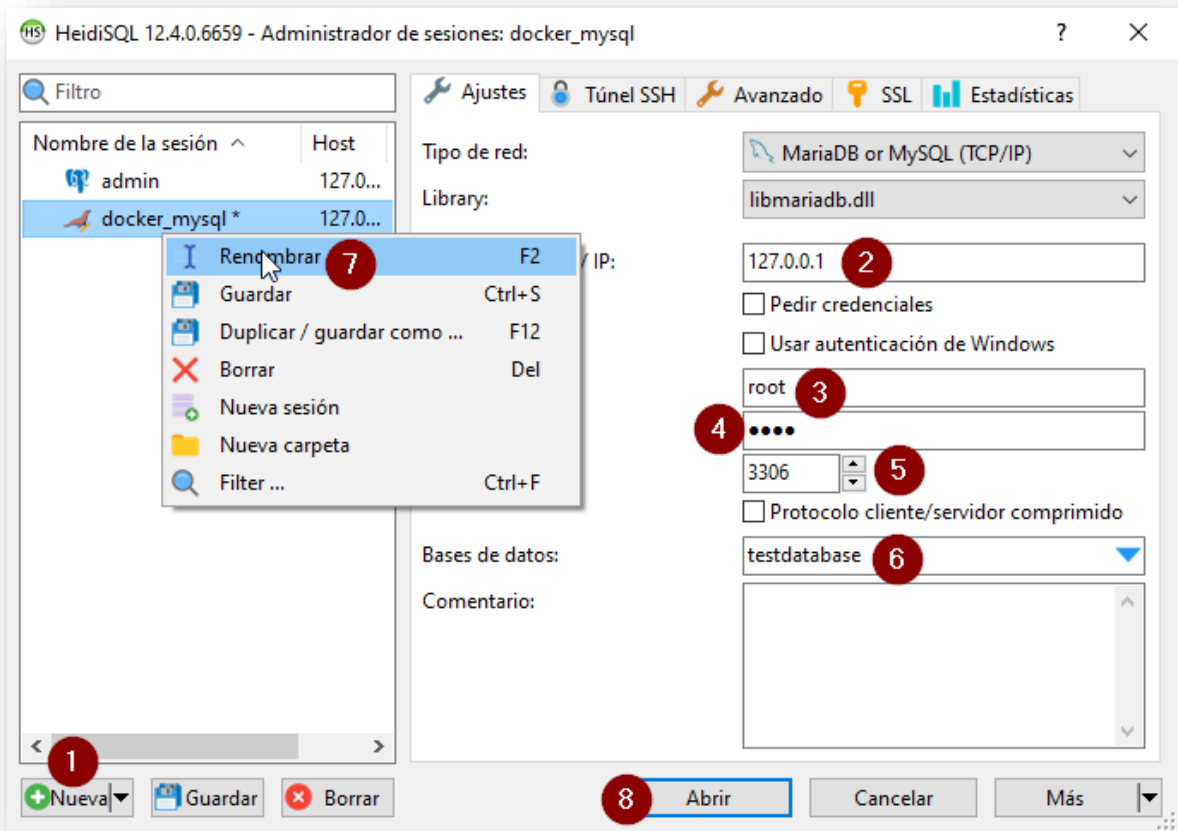
```
docker run -p 9902:80 --name miservidor -v c:/Docker/paginas:/var/www/html -d --link basededatos php:7.0-apache
```

Crear el contenedor con la imagen de PhpMyAdmin Mariadb (el comando debe ser en solo una línea)

```
docker run --name myadmin -d --link basededatos -p 8080:80 -e PMA_HOST=basededatos phpmyadmin/phpmyadmin
```



- Instalar la aplicación [HEIDISQL](#)



- Crear la conexión hacia la base de datos “testdatabase”
- Ingresar y cargar el esquema [HR](#)



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

Containers [Give feedback](#)

A container packages up code and its dependencies so the application runs quickly and reliably from one computing environment to another. [Learn more](#)

Search ☰ ☐ Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	Last started	Actions
<input type="checkbox"/>	mimysql a906fa108b95	mariadb	Running	3306:3306	20 minutes ago	
<input type="checkbox"/>	myadmin 859b99d48ceb	phpmyadmin	Running	8080:80	11 minutes ago	

Se ingresa a la línea de comando bash para iniciar a mysql

```
docker exec -it a906 bash
```

Ingresar a mysql

```
root@a906fa108b95: /
root@a906fa108b95:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 27
Server version: 10.11.2-MariaDB-1:10.11.2+maria~ubu2204 mariadb.org binary distribution
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use hr
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [hr]> show tables;
+-----+
| Tables_in_hr |
+-----+
| countries    |
| departments  |
| emp_details_view |
| employees    |
| job_history   |
| jobs         |
| locations    |
| regions      |
+-----+
8 rows in set (0.000 sec)

MariaDB [hr]>
```



Servicio Nacional de Aprendizaje Formato Taller Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

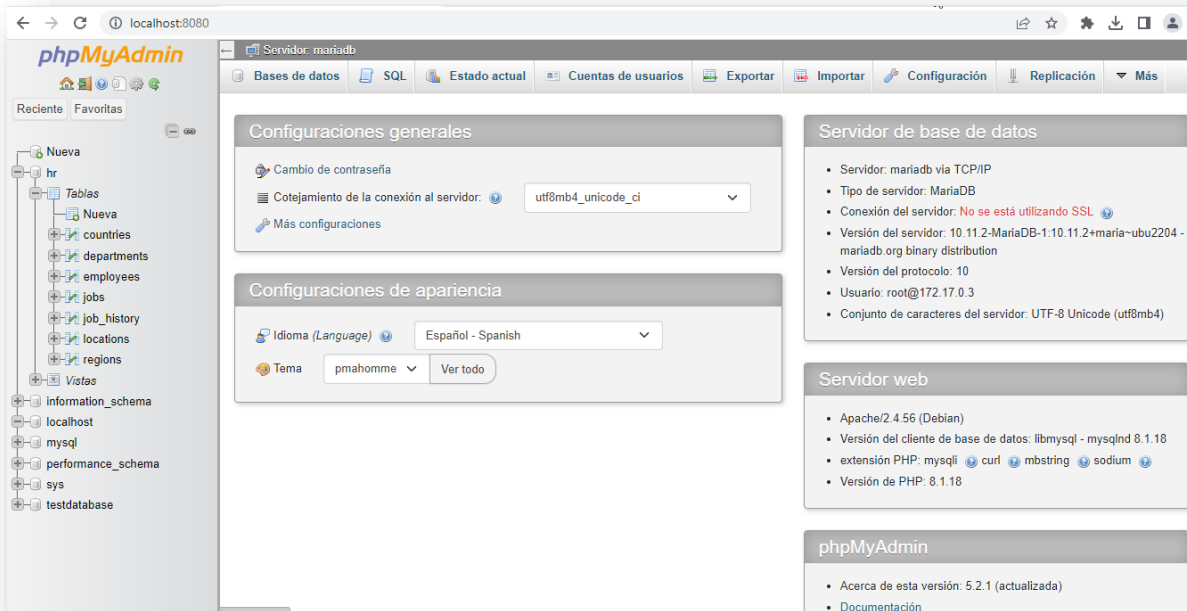
HeidiSQL Portable 12.4.0.6659

Host: 127.0.0.1 Base de datos: hr Tabla: employees

hr.employees: 107 filas en total (aproximadamente)

employee_id	first_name	last_name	email	phone_number	hire_date	job_id	salary	commission_pct	manager_id	department_id
100	Steven	King	SKING	515.123.4567	1987-06-17	AD_PRES	24,000.0	(NULL)	(NULL)	(NULL)
101	Neena	Kochhar	NKOCHHAR	515.123.4568	1989-09-21	AD_VP	17,000.0	(NULL)	100	100
102	Lex	De Haan	LDEHAAN	515.123.4569	1993-01-13	AD_VP	17,000.0	(NULL)	100	100
103	Alexander	Hunold	AHUNOLD	590.423.4567	1990-01-03	IT_PROG	9,000.0	(NULL)	102	102
104	Bruce	Ernst	BERNST	590.423.4568	1991-05-21	IT_PROG	6,000.0	(NULL)	103	103
105	David	Austin	DAUSTIN	590.423.4569	1997-06-25	IT_PROG	4,800.0	(NULL)	103	103
106	Valli	Pataballa	VPATABALL	590.423.4560	1998-02-05	IT_PROG	4,800.0	(NULL)	103	103
107	Diana	Lorentz	DLORENTZ	590.423.5567	1999-02-07	IT_PROG	4,200.0	(NULL)	103	103
108	Nancy	Greenberg	NGREENBE	515.124.4569	1994-08-17	FI_MGR	12,000.0	(NULL)	101	101
109	Daniel	Faviet	DFAVIET	515.124.4169	1994-08-16	FI_ACCOUNT	9,000.0	(NULL)	108	108
110	John	Chen	JCHEN	515.124.4269	1997-09-28	FI_ACCOUNT	8,200.0	(NULL)	108	108
111	Ismael	Sclera	ISCLARRA	515.124.4369	1997-09-30	FI_ACCOUNT	7,700.0	(NULL)	108	108
112	Jose Manuel	Urman	JMURMAN	515.124.4469	1998-03-07	FI_ACCOUNT	7,800.0	(NULL)	108	108
113	Luis	Popp	LOPOP	515.124.4567	1998-12-07	FI_ACCOUNT	6,900.0	(NULL)	108	108
114	Den	Raphaely	DRAPHEAL	515.127.4561	1994-12-07	PU_MAN	11,000.0	(NULL)	100	100
115	Alexander	Khoo	AKHOO	515.127.4562	1995-05-18	PU_CLERK	3,100.0	(NULL)	114	114
116	Shelli	Baile	SBAIDA	515.127.4563	1997-12-24	PU_CLERK	2,900.0	(NULL)	114	114
117	Sigal	Tobias	STOBIAS	515.127.4564	1997-07-24	PU_CLERK	2,800.0	(NULL)	114	114
118	Guy	Himuro	GHIIMURO	515.127.4565	1998-11-15	PU_CLERK	2,600.0	(NULL)	114	114
119	Karen	Colmenares	KCOLMENA	515.127.4566	1999-08-10	PU_CLERK	2,500.0	(NULL)	114	114
120	Matthew	Weiss	MWEISS	650.123.1234	1996-07-18	ST_MAN	8,000.0	(NULL)	100	100
121	Adam	Fripp	AFRIPP	650.123.2234	1997-04-10	ST_MAN	8,200.0	(NULL)	100	100
122	Payam	Kaufling	PKAUFLIN	650.123.3234	1995-05-01	ST_MAN	7,900.0	(NULL)	100	100
123	Shanta	Vollman	SVOLLMAN	650.123.4234	1997-10-10	ST_MAN	6,500.0	(NULL)	100	100

- Ingresar al PhpMyAdmin desde <http://localhost:8080/>



Borrar todas las imágenes

docker image prune

7/05/2023 Pág. 38

Línea de atención al ciudadano: 018000 910270
Línea de atención al empresario: 018000 910682



Borrar todos los contenedores

```
docker container prune
```



Crear una base de datos desde esta [fuente de datos](#), cárguelo en un contenedor Docker y realizar operaciones con la matriz CRUD.

4.2 PRACTICA CON MONGODB Y DOCKER

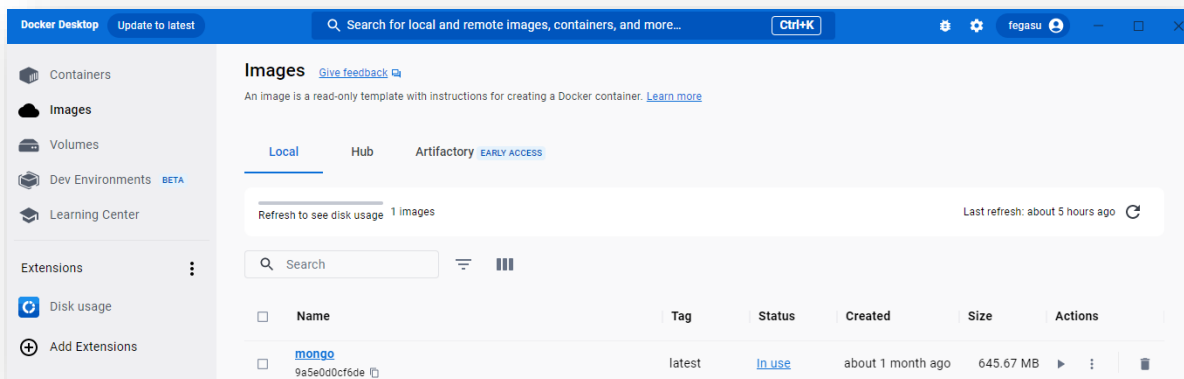


Para realizar la práctica, descargar “[MongoDB Compass](#)”

- Crear el contenedor con la imagen de MongoDB (el comando debe ser en solo una línea)

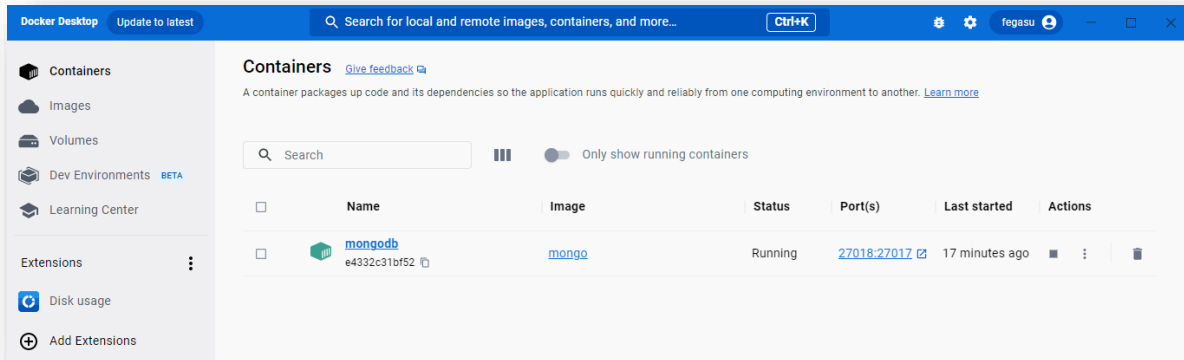
```
docker run --name=mongodb -p27018:27017 -d mongo
```

El puerto con el que se conecta por fuera de Docker es 27018

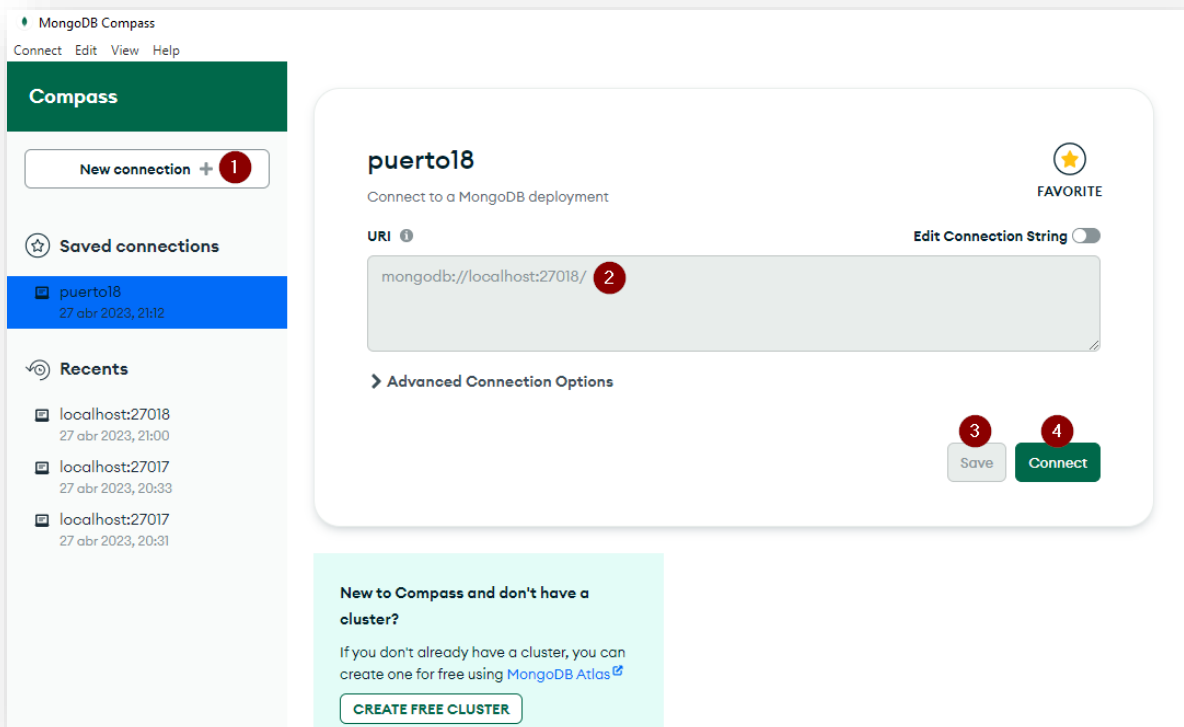




Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.



- Conectar a MongoDB de MongoDB Compass



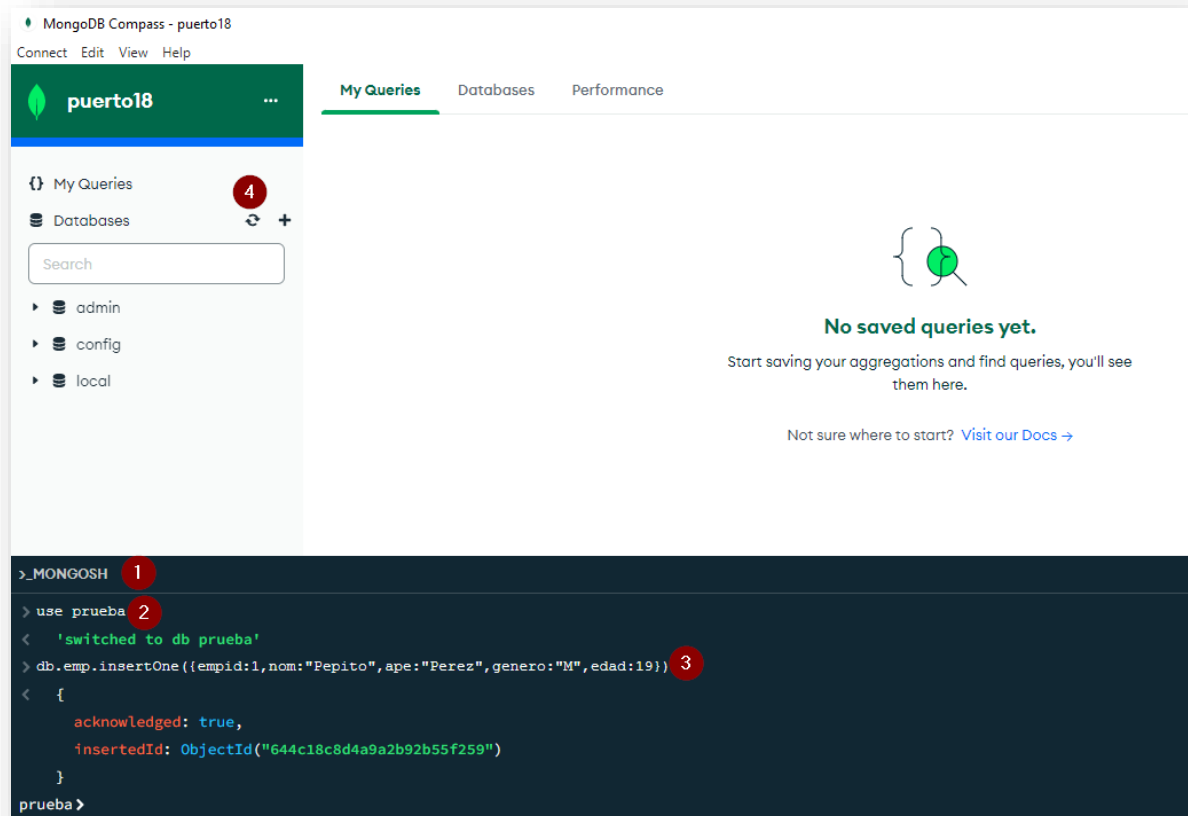
Ingresar a “MongoDB compass” y siga los siguientes pasos:

2. Entra a MONGOSH
3. Crear la base de datos “prueba”
4. Crear el documento en la colección “EMP” con los siguientes datos:
 - a. Id=1



Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

- b. Nombre=Pepito
 - c. Apellido=Pérez
 - d. Genero=M
 - e. Edad=19
5. Refrescar



Crear una base de datos desde esta [fuente de datos](#), cárguelo en un contenedor Docker y realizar operaciones con la matriz CRUD.

4.3 PRACTICA CON POSTGRESQL Y DOCKER

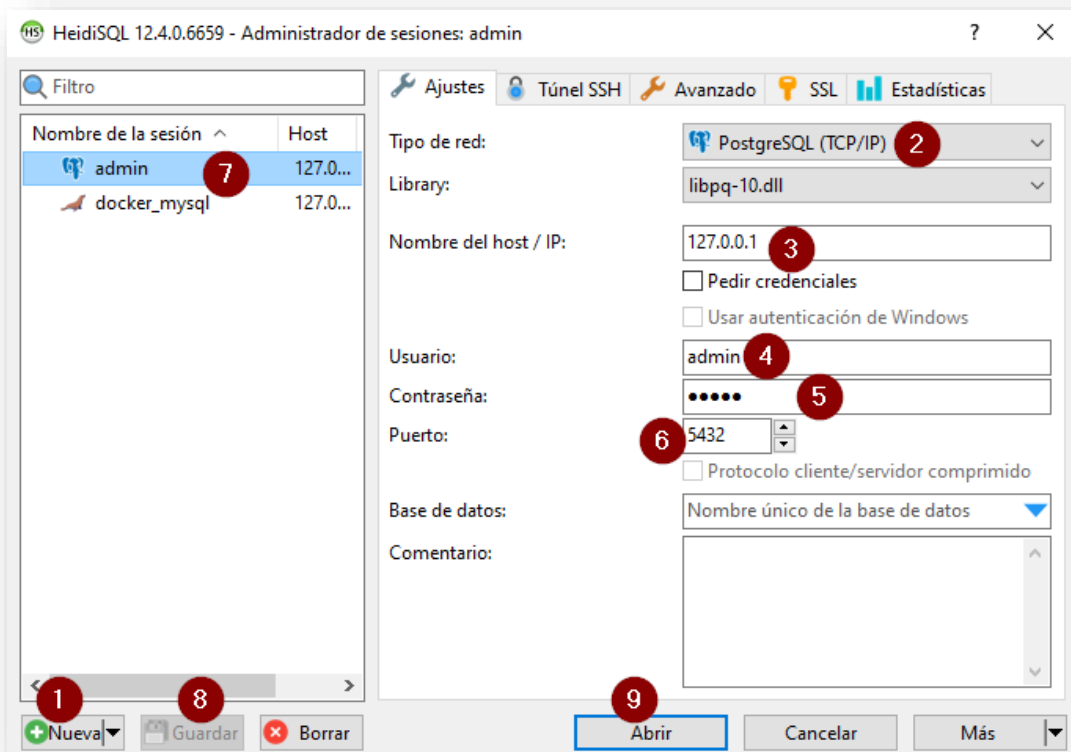


Descargar HEIDISQL para realizar la practica

- Crear el contenedor con la imagen de Postgresql (el comando debe ser en solo una línea)

```
docker run -itd -e POSTGRES_USER=admin -e POSTGRES_DB=hr -e POSTGRES_PASSWORD=admin -p 543:5432 --name postgresql postgres
```

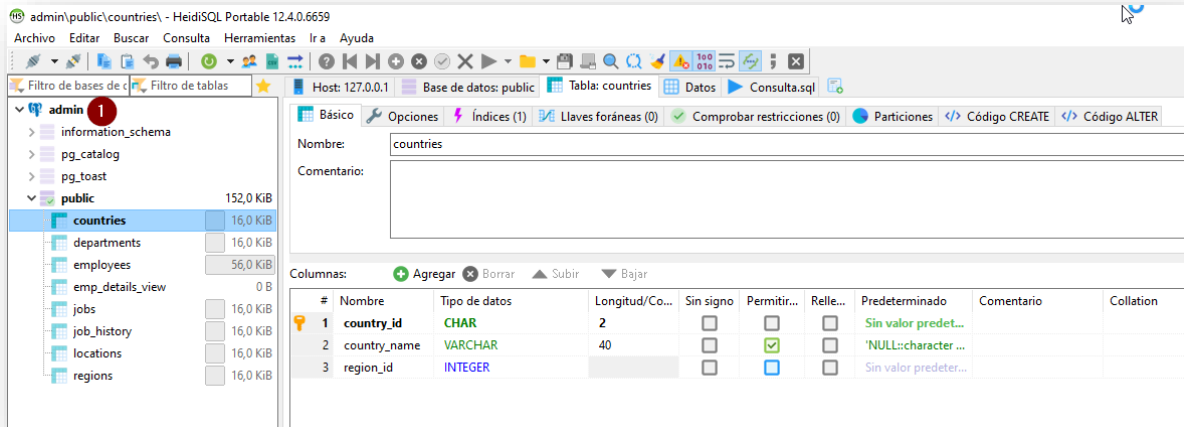
Configurar la conexión con HEIDISQL





Servicio Nacional de Aprendizaje
Formato Taller
Centro de Gestión de Mercados, Logística y Tecnologías de la Información.

Cargar el esquema



Crear una base de datos desde esta [fuente de datos](#), cárguelo en un contenedor Docker y realizar operaciones con la matriz CRUD.

**EVIDENCIA(S) A ENTREGAR:**

1. Realizar un video en YouTube que muestre la creación de contenedores en Docker y utilizar las operaciones con SQL y ORM desde Python, utilizando un dataset de los datos personales de sus compañeros de ficha (nombres, apellidos, dni, tipo dni, genero, fecha de nacimiento, ciudad de nacimiento), en SQLite, MySQL, POSTGRESQL y MongoDB.
2. Se debe realizar desde Python.
3. Enviar el script de creación del punto 2.

Nota: Esta evidencia se debe realizar en grupo de 4 aprendizaje, se debe escoger un motor de base de datos (**SQLite, MySQL, POSTGRESQL y MongoDB**), y se debe realizar todas las operaciones aprendidas en Python y registrado en el video. Se debe subir individualmente indicando los nombres de sus compañeros de equipo.

CONTROL DEL DOCUMENTO

	Nombre	Cargo	Dependencia	Fecha
Autor (es)	José Fernando Galindo Suarez	Instructor	CGMLTI- Teleinformática	16/02/2023

CONTROL DE CAMBIOS (diligenciar únicamente si realizan ajustes al taller)

	Nombre	Cargo	Dependencia	Fecha	Razón del Cambio
Autor (es)					

ⁱ [https://aws.amazon.com/es/what-is/sql/#:~:text=es%20importante%20SQL%3F-,El%20lenguaje%20de%20consulta%20estructurada%20\(SQL\)%20es%20un%20lenguaje%20de,los%20diferentes%20lenguajes%20de%20programaci%C3%B3n.](https://aws.amazon.com/es/what-is/sql/#:~:text=es%20importante%20SQL%3F-,El%20lenguaje%20de%20consulta%20estructurada%20(SQL)%20es%20un%20lenguaje%20de,los%20diferentes%20lenguajes%20de%20programaci%C3%B3n.)