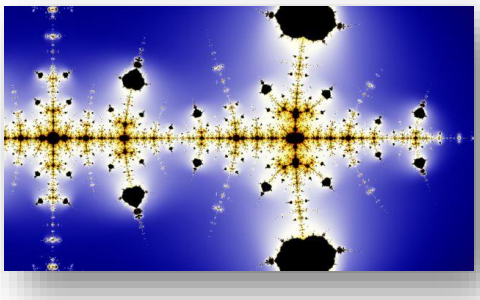
GFPI-F-135 REALIZA EL PROCESO DE LIMPIEZA DE DATOS
LENGUAJE DE CONSULTA ESTRUCTURADAS

ACTIVIDADES POR DESARROLLAR:

1. **Conectar desde Python a MySQL**
2. **Realizar consultas SQL desde Python**
3. **Entender que es un ORM**
4. **Aplicar comandos ORM**



REFLEXIÓN INICIAL

Simple no quiere decir fácil.

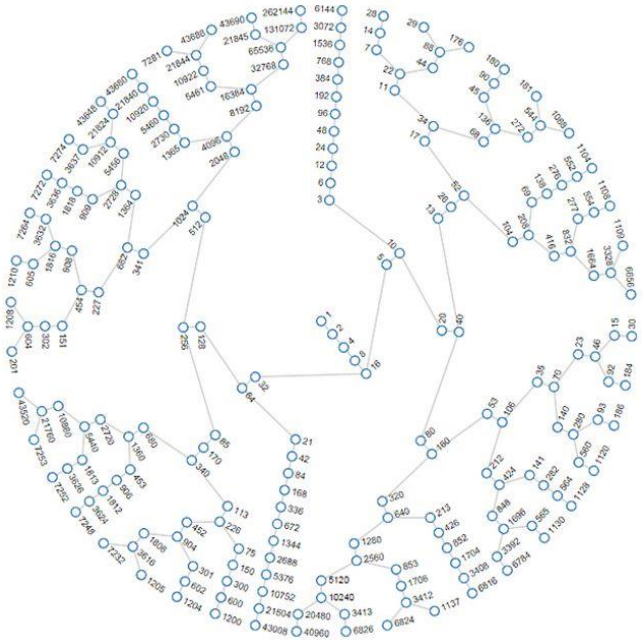
Analizar el siguiente caso de estudio llamado "Números Granizo", que los matemáticos consideran un simple problema sin solución.

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.



¿Qué patrón se observa cuando finaliza con cualquier número que se ingrese?

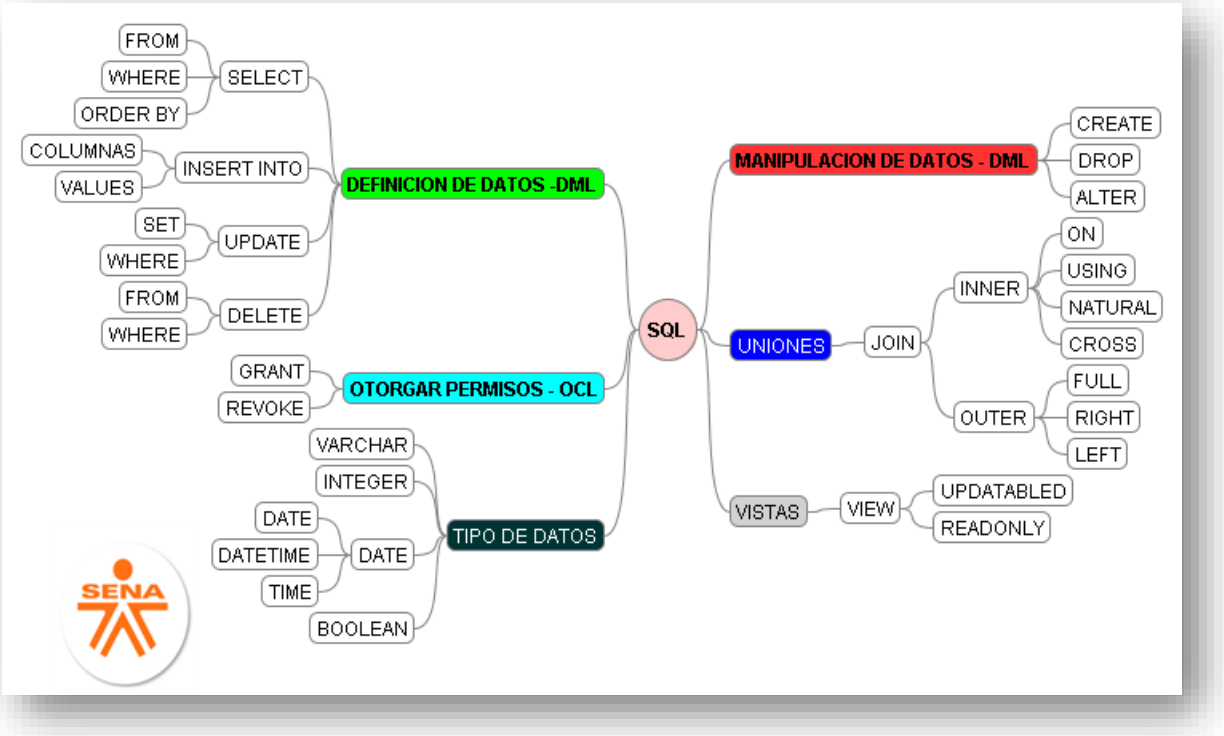
Entregables:

- 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
  - 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.[i]

¿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. Por ejemplo, pueden incrustar consultas SQL con el lenguaje de programación Java 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. Además, SQL es muy fácil de aprender, ya que en sus instrucciones se utilizan palabras clave comunes en inglés.

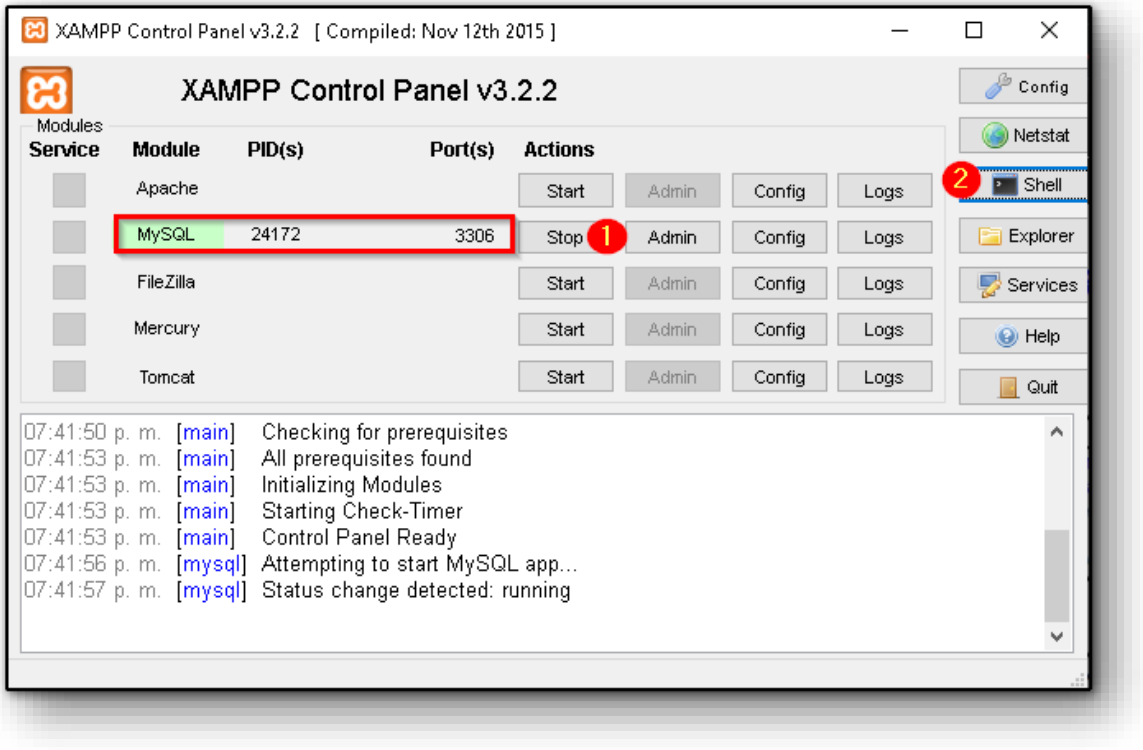
Para realizar la siguiente práctica de laboratorio se recomienda ver el "Manual de SQL con MYSQL"

PRACTICA DE LABORATORIO

Antes de comenzar instalar el driver de MySQL para Python.

```
pip install mysql-connector-python
```

Iniciar el servidor de MySQL (1)

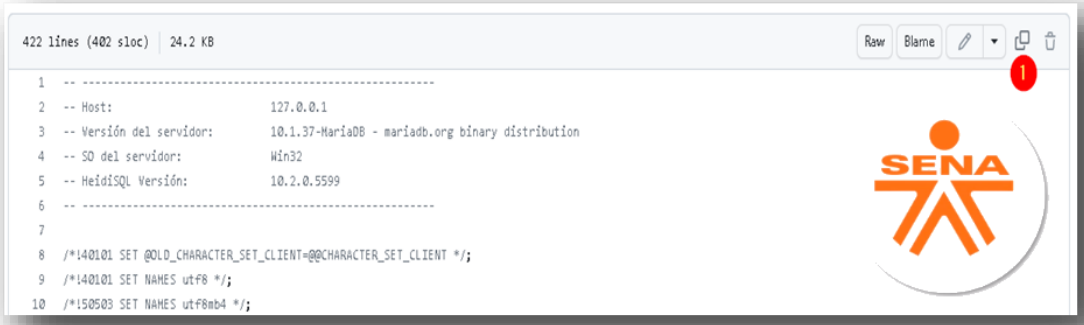


Ingresar a la consola de MySQL (2)

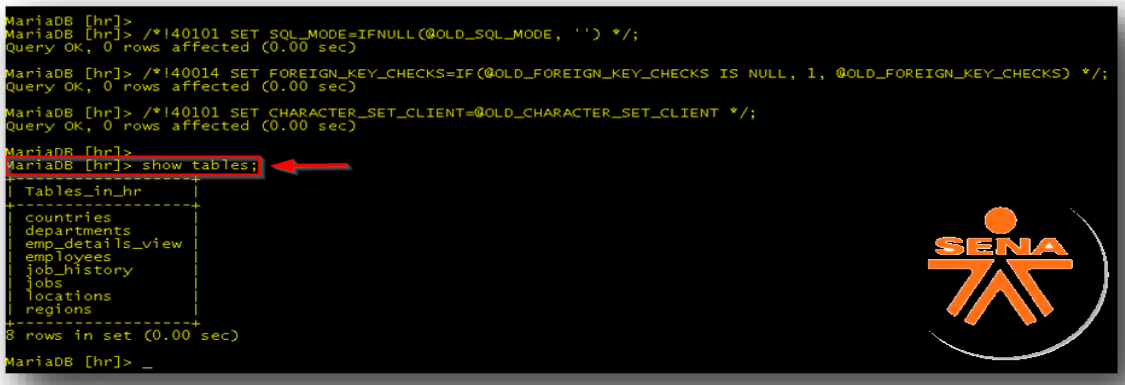
```
# mysql -u root -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 12
Server version: 10.1.37-MariaDB mariadb.org binary distribution

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

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

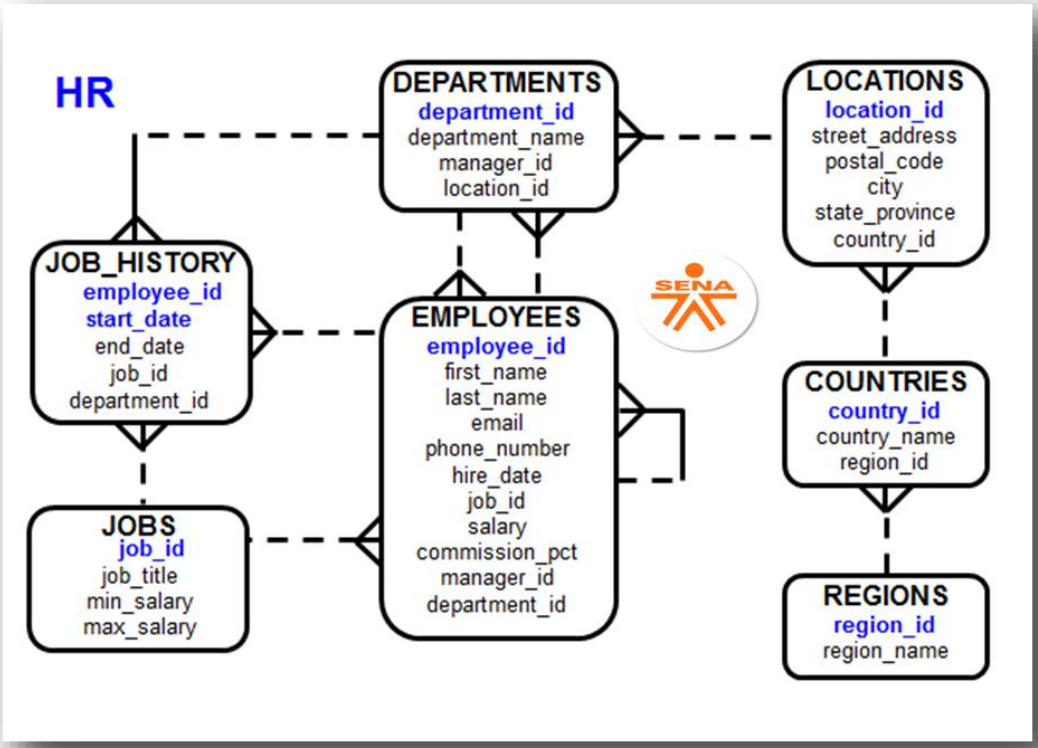
Descargar el script de creación de la base de datos HR

Copiar el script al block de notas.



Pegar desde el portapapeles a la consola Mysql





Conectar desde Python a MySQL

Instalar el driver de MySQL para Python

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

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

connection = mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="root",
        database="hr")
cursor = connection.cursor()
query="select * from employees"
cursor.execute(query)
```

```
result = pd.DataFrame( cursor.fetchall())
print(result)
```

Consultar los datos de la tabla employees mostrando las 10 primeras filas.

```
MariaDB [hr]> SELECT * FROM EMPLOYEES LIMIT 10;
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+-
--------------+
| 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    | 24000.00 |           NULL |       NULL |
           90 |
|         101 | Neena      | Kochhar   | NKOCHHAR | 515.123.4568 | 1989-09-21 | AD_VP      | 17000.00 |           NULL |        100 |
           90 |
|         102 | Lex        | De Haan   | LDEHAAN  | 515.123.4569 | 1993-01-13 | AD_VP      | 17000.00 |           NULL |        100 |
           90 |
|         103 | Alexander  | Hunold    | AHUNOLD  | 590.423.4567 | 1990-01-03 | IT_PROG    |  9000.00 |           NULL |        102 |
           60 |
|         104 | Bruce      | Ernst     | BERNST   | 590.423.4568 | 1991-05-21 | IT_PROG    |  6000.00 |           NULL |        103 |
           60 |
|         105 | David      | Austin    | DAUSTIN  | 590.423.4569 | 1997-06-25 | IT_PROG    |  4800.00 |           NULL |        103 |
           60 |
|         106 | Valli      | Pataballa | VPATABAL | 590.423.4560 | 1998-02-05 | IT_PROG    |  4800.00 |           NULL |        103 |
           60 |
|         107 | Diana      | Lorentz   | DLORENTZ | 590.423.5567 | 1999-02-07 | IT_PROG    |  4200.00 |           NULL |        103 |
           60 |
|         108 | Nancy      | Greenberg | NGREENBE | 515.124.4569 | 1994-08-17 | FI_MGR     | 12000.00 |           NULL |        101 |
          100 |
|         109 | Daniel     | Faviet    | DFAVIET  | 515.124.4169 | 1994-08-16 | FI_ACCOUNT |  9000.00 |           NULL |        108 |
          100 |
+-------------+------------+-----------+----------+--------------+------------+------------+----------+----------------+------------+-
--------------+
10 rows in set (0.00 sec)

MariaDB [hr]>
```

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

connection = mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="root",
        database="hr")
cursor = connection.cursor()
query="select * from employees"
cursor.execute(query)
result = pd.DataFrame( cursor.fetchall())
print(result)
```

Salida:

```
     0       1      2       3  ...      7    8     9     10
0    100   Steven    King    SKING  ...  24000.00  None   NaN   90.0
1    101    Neena  Kochhar  NKOCHHAR  ...  17000.00  None  100.0   90.0
2    102      Lex  De Haan   LDEHAAN  ...  17000.00  None  100.0   90.0
3    103  Alexander   Hunold   AHUNOLD  ...   9000.00  None  102.0   60.0
4    104     Bruce    Ernst    BERNST  ...   6000.00  None  103.0   60.0
..   ...       ...      ...       ...  ...      ...   ...    ...    ...
102  202       Pat      Fay      PFAY  ...   6000.00  None  201.0   20.0
103  203     Susan   Mavris   SMAVRIS  ...   6500.00  None  101.0   40.0
104  204   Hermann     Baer     HBAER  ...  10000.00  None  101.0   70.0
105  205   Shelley  Higgins  SHIGGINS  ...  12000.00  None  101.0  110.0
106  206   William    Gietz    WGIETZ  ...   8300.00  None  205.0  110.0

[107 rows x 11 columns]
```

Consultamos los datos de la tabla employees aplicando un filtro en employee_id=100

```
MariaDB [hr]>
MariaDB [hr]> SELECT * FROM EMPLOYEES WHERE EMPLOYEE_ID=100;
+-------------+------------+-----------+-------+--------------+------------+---------+----------+----------------+------------+-------
--------+
| employee_id | first_name | last_name | email | phone_number | hire_date  | job_id  | salary   | commission_pct | manager_id | depart
ment_id |
+-------------+------------+-----------+-------+--------------+------------+---------+----------+----------------+------------+-------
--------+
|         100 | Steven     | King      | SKING | 515.123.4567 | 1987-06-17 | AD_PRES | 24000.00 |           NULL |       NULL |
     90 |
+-------------+------------+-----------+-------+--------------+------------+---------+----------+----------------+------------+-------
--------+
1 row in set (0.00 sec)
```

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

connection = mysql.connector.connect(
        host="localhost",
        user="root",
        passwd="root",
        database="hr")
cursor = connection.cursor()
query="select * from employees where employee_id=100"
cursor.execute(query)
result = pd.DataFrame( cursor.fetchall())
print(result)
```

Salida:

```
   0    1      2      3            4  ...    6      7    8    9   10
0  100  Steven  King  SKING  515.123.4567  ...  AD_PRES  24000.00  None  None  90

[1 rows x 11 columns]
```

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)
```

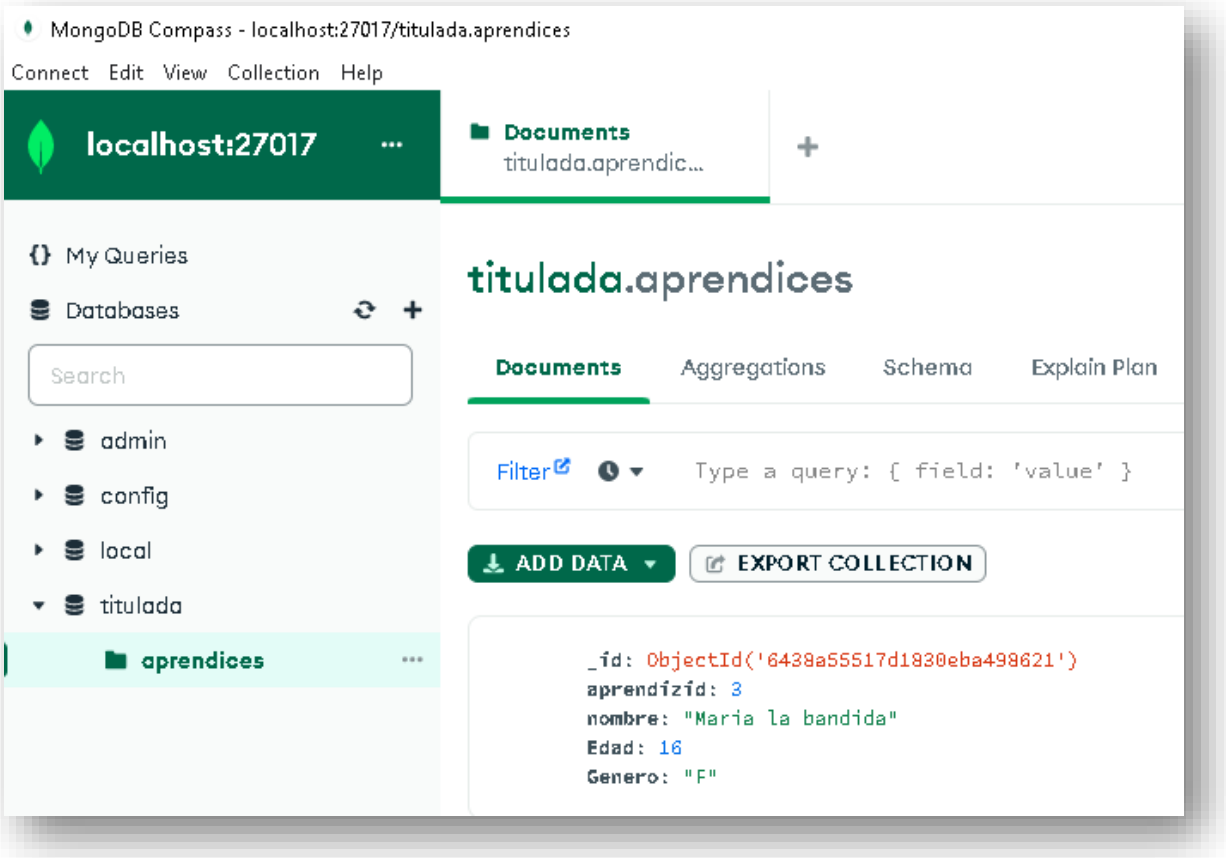
Crear la base de datos llamada "titulada" y la colección "aprendices"

```
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 Melo 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 genero 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()
```

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.

Practica ORM con MySQL y PEEWEE

- Instalar el driver ORM

```
pip3 install weepeek
```

- 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)
```

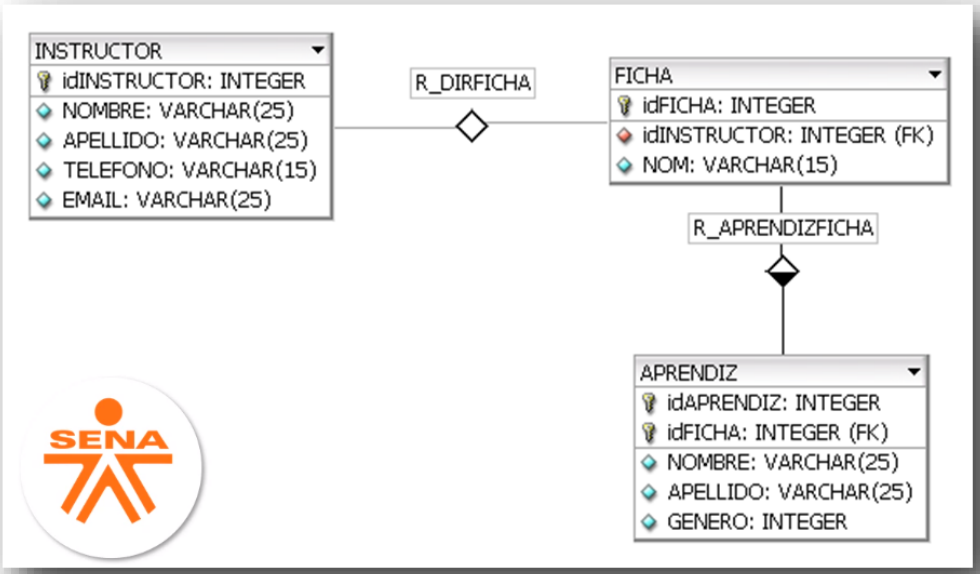
- Listar los registros con un filtro:

```
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)
```

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')
```

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)
```

EVIDENCIA(S) A ENTREGAR:

1. Realizar operaciones con SQL y ORM utilizando un dataset COVID19 en SQLite, MySQL y PostgreSQL.
2. Enviar el script de creación del punto anterior.
3. Documentar

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.