Ayudantía 3
Web y SQL

Ayudantes:
Consuelo Inostroza, Amelia González, Tomás Jackson, Vicente San Martín
Ricardo Oviedo, Daniela Torrent, Mateo Andrade

Objetivos:

Esta ayudantía está enfocada en la Entrega 2. Hoy aprenderemos:

- Cómo trabajar y conectarse al **servidor.**
- **HTML:** Para crear la visualización de la página.
- **Python:** Para poder hacer la conexión a la base de datos y hacer consultas **SQL** a esta.
- Usaremos la librería **psycopg2** de python para poder trabajar con **postgres.**
- **IMPORTANTE:** ¿Cómo evitamos **inyecciones** en SQL?

Uso del servidor

¿Como conectarse al servidor?

Conociendo nuestro número de grupo, en el terminal haremos lo siguiente:

```
ssh grupoXX@codd.ing.puc.cl
```

Observaciones

- Nuestra contraseña por defecto será: grupoXX
- Para cambiar la contraseña:
  - Comando passwd
    - Pedirá la clave actual
    - Pedirá la clave nueva dos veces

¿Cómo utilizamos Postgresql en el server?

1. Luego de ingresar al servidor, hacemos el comando **psql.**
2. Ingresar contraseña (por defecto: grupoXX)
3. Cambiamos la contraseña:

   ALTER USER <grupoXX> ENCRYPTED PASSWORD ‘newpassword’;

4. Ingresar a la base de datos correspondiente:

   \c grupoXXeN

   Para esta entrega N=2.

¿Cómo trabajamos en postgres?

Creación de tablas:

CREATE TABLE users (User_id INT (serial) PRIMARY KEY, username VARCHAR (15) UNIQUE NOT NULL, ...);

Poblamos tablas (manualmente):

INSERT INTO users (User_id, username, ...) VALUES (1, ‘sdawda’, ...);

Para poblarlas con un CSV:

\COPY users (columns_name) FROM ‘relative/path/to/file.csv’ DELIMITER ‘,’ CSV HEADER;

Comandos útiles

- \l lista de base de datos.
- \c <db> Ingresar a la base de datos db.
- \dt lista de las tablas de esa base de datos
- \d <table_name> descripción de la tabla <table_name>
- \? todos los comandos de psql
- \q salir (CTRL + D también funciona)

HTML

Hyper Text Markup Language

- Es el lenguaje de marcado estándar utilizado para crear páginas web.
- Será la representación visual de la página.
- Se puede editar desde cualquier editor de texto.
- Cómo funciona:

  Marca:

  <marca>Contenido</marca>

Marca viene a ser el tipo de marca y contenido lo que está dentro de esta.

Tipos de marca

- html: marca que describe el documento HTML (todas las otras marcas van dentro de él)
- head: Marca que contiene el encabezado (información sobre el documento) acá pueden agregar stylesheet.css
- title: Marca que contiene el título de la página.
- body: Maraca que contiene el cuerpo del documento. Dentro de ella agregamos todos los elementos que queremos ver en nuestra visualización.
- h1: Marca que contiene un título.

Tipos de marca

- table: Marca que define una tabla. las siguientes marcas van dentro de esta tabla:
  - tr: contiene la fila de una tabla.
  - td: contiene una celda de una tabla.
  - th: contiene el encabezado de la fila de una tabla.

Python y psycopg2

Importamos la librería (nos sirve para trabajar con postgres):

```
import psycopg2
```

Conexión con pyscopg2

Objeto para conectar:

```
try:
    conn = psycopg2.connect(
        database="database",
        user="user",
        host="localhost",
        port=5432,
        password="pass")
except:
    print("No me pude conectar")
```

Inserción

```
query = "INSERT INTO Peliculas VALUES (" + \
        titulo + ", " + ano + \
        ", " + director + ")";
cur = conn.cursor()
try:
    cur.execute(query)
except psycopg2.Error as e:
    print(e.pgcode)
```

Cursores (fetchone)

-Puntero que va recorriendo los resultados.

-Iremos iterando de fila en fila y las iremos imprimiendo una por una:

```
import psycopg2

try:
    conn = psycopg2.connect(database="dbname",
            user="dbuser", host="localhost",
            password="dbpass")
    cur = conn.cursor()
    cur.execute("SELECT * FROM R")
    row = cur.fetchone()
    while row:
        print(row)
        row = cur.fetchone()
except:
    print("Hubo algún problema")
```

Otra forma (fetchall)

```
try:
    conn = psycopg2.connect(database="dbname",
            user="dbuser", host="localhost",
            password="dbpass")
    cur = conn.cursor()
    cur.execute("SELECT * FROM R")
    rows = cur.fetchall()
    for row in rows:
        print(row)
except:
    print("Hubo algún problema")
```

Pasamos parámetros

```
cur.execute("""
    INSERT INTO users (id, fecha_primer_inicio_sesion,
username)
    VALUES (%s, %s, %s);
    """,
    (10, datetime.date(2020, 10, 18), "dawda"))
```

En orden y sin declarar argumentos.

Parámetros

Si queremos declarar y nombrar a los argumentos que ingresemos:

```
cur.execute("""
    INSERT INTO users (id, fecha_primer_inicio_sesion, username)
    VALUES (%(id)s, %(fecha_inicio)s, %(usario)s);
    """,
    {'id': 10, 'fecha_inicio': datetime.date(2020, 10, 18),
'usuario': "dawda"})
```

OJO

Si no declaramos argumentos como en la anterior viñeta y si solo insertamos un argumento en el execute entonces no nos olvidemos de cerrar bien el paréntesis y colocar una ‘,’ luego del argumento para insertar:

```
cur.execute("INSERT INTO some_table VALUES (%s)", ("adawd"))  # INCORRECTO
cur.execute("INSERT INTO some_table VALUES (%s)", ("adawd",)) # CORRECTO
```

Inyección SQL

NUNCA HACER ESTO: (No concatenar)

```
SQL = "INSERT INTO usuarios (name) VALUES (' %s')"
data = ("Jaime", )
cur.execute(SQL % data)
```

No usar comillas y usar el operador %.

Forma correcta

Para evitar inyecciones, cambiamos lo siguiente en el anterior código. No utilizamos las comillas en el parámetro y no usamos el operador %.

```
SQL = "INSERT INTO usuarios (name) VALUES (%s)"
data = ("Jaime", )
cur.execute(SQL, data)
```