DigitalHouse > Coding School

DATA SCIENCE

UNIDAD 3
MÓDULO 5

Introducción a Bases de Datos, SQL y Clasificación

Octubre 2017

PRÁCTICA DE SQL

USO DE BASES DE DATOS DESDE PYTHON

OBJETIVOS DE LA CLASE

1. **Conectarse a una Base de Datos local o remota mediante Python.**
2. **Conectarse a una Base de Datos local o remota mediante un GUI del DBMS.**
3. **Efectuar consultas mediante el comando SELECT.**
4. **Aprender a utilizar funciones de agregación para cálculos sencillos.**

- Acceder a SQLite desde la línea de comandos es útil para operaciones de SQL básicas o específicas.
- Para operaciones complejas, voluminosas y que se realizan siempre de la misma manera conviene ejecutarlas directamente sobre la Base de Datos.
- Para investigar los datos y analizarlos interactivamente, puede resultar útil hacerlo desde Python, utilizando el paquete **`sqlite3`**.

*A continuación vamos a escribir un programa en **Python** para conectarnos a una Base de Datos SQLite y realizar algunas operaciones sobre la misma.*

INTERACTUAR CON SQLITE DESDE PYTHON

Práctica guiada 1

1. Conectarse al archivo SQLite llamado **test_db.sqlite** que representa nuestra Base de Datos.

```
import sqlite3
sqlite_db = 'test_db.sqlite'
conn = sqlite3.connect(sqlite_db)
c = conn.cursor()
```

— Si el archivo no existe, lo crea y se conecta al mismo.

2. Mediante el método Execute, enviarle el commando SQL **CREATE TABLE** para crear una nueva tabla.

```
c.execute('CREATE TABLE houses (field1 INTEGER PRIMARY KEY,
sqft INTEGER, bdrms INTEGER, age INTEGER, price INTEGER);')

# Confirmar los cambios
conn.commit()
```

*En este momento, si se accede a la Base de Datos **test_db.sqlite** desde el DB Browser, por ejemplo, se puede visualizar la estructura de la tabla recién creada.*

3. Crear una tupla con los datos de la última casa vendida y enviarla a la conexión a través del comando SQL **INSERT** para agregarla a la tabla Houses.

```
last_sale = (None, 4000, 5, 22, 619000)
c.execute('INSERT INTO houses VALUES (?,?,?,?,?)',last_sale)

# Nuevamente, confirmar los cambios
conn.commit()
```

Técnicas de programación Python combinadas con el comando SQL:

- El método execute() soporta el caracter “?” para sustituir los valores en el comando INSERT.
- Se utiliza **None**, que en Python equivale al NULL de SQL, para que SQLite auto-incremente la clave.

Para más info.: https://docs.python.org/2.7/library/sqlite3.html

4. Crear un array de tuplas para insertar varios registros a la vez:

```
recent_sales = [
  (None, 2390, 4, 34, 319000),
  (None, 1870, 3, 14, 289000),
  (None, 1505, 3, 90, 269000),
]

c.executemany('INSERT INTO houses VALUES (?, ?, ?, ?, ?)',
recent_sales)

conn.commit()
```

— El método **executemany()** recibe el array de tuplas y va iterando a través del mismo, sustituyendo de a una tupla a la vez.

*Acceder a la Base de Datos **test_db.sqlite** desde el DB Browser para visualizar los datos.*

5. Sobre el mismo programa Python (o uno nuevo), cargar en un array el archivo **housing.csv** mediante la función **genfromtxt** de **numpy**.

```
from numpy import genfromtxt

# importar un array numpy de ints y convertirlo a una
# lista de listas
data = (genfromtxt('housing.csv', dtype='i8',
                   delimiter=',', skip_header=1)).tolist()
```

```
# Como en el CSV no existe el campo Field1,
# crearlo asignándole el valor “None” a cada sub-lista
for d in data:
    d.insert(0, None)
```

6. Iterar en la estructura **data** para insertar (comando INSERT) cada sub-list en un registro de la tabla Houses.

```
for d in data:
    c.execute('INSERT INTO houses VALUES (?, ?, ?, ?, ?)', d)

conn.commit()
```

7. Mediante el comando SQL **SELECT** realizar una consulta (e.g.: casas de 4 dormitorios):

```
results = c.execute("SELECT * FROM houses WHERE bdrms = 4")

results.fetchall()
```

¿Cómo hacer para modificar o eliminar datos de la tabla Houses?

INTERACTUAR CON SQLITE DESDE PYTHON

Práctica guiada 2

Utilizar Pandas Dataframes en Python para manipular una Base de Datos SQLite

andas.

```
import sqlite3
import pandas as pd
from pandas.io import sql
```

una nueva).

```
sqlite_db = 'test_db2.sqlite'
conn = sqlite3.connect(sqlite_db)
```

do.

```
data = pd.read_csv('housing.csv', low_memory=False)
data.head()
```

4. Almacenarlo en la Base de Datos, en una nueva tabla **houses_pandas**

```
data.to_sql('houses_pandas', con=conn,
            if_exists='replace', index=False)
```

5. Consultar el contenido de la tabla (e.g.: los primeros 10 registros) mediante el comando **SELECT**

```
sql.read_sql('select * from houses_pandas limit 10', con=conn)
```

Para más info.: http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.to_sql.html

SINTAXIS DE COMANDOS SQL

Práctica guiada 2

Practicar las consultas sobre la tabla **houses_panda**

```
SELECT <*, columnas>

FROM <tabla>

WHERE <condición>;
```

1. Traer todos los campos de todos los registros.
2. Mostrar los valores de los campos sqft y bdrms para todos los registros
3. Mostrar dichos valores para casas de más de 50 años
4. Mostrar dichos valores para casas de más de 50 años y menos de $500.000

Funciones de Agregación

Reciben los valores de determinado campo contenidos en varias filas de la tabla, los procesan y devuelven un resultado que le representa al usuario mayor información.

Con estas funciones se puede:

- **SUM(**<campo>**)**: Sumar valores numéricos de un campo.
- **AVG(**<campo>**)**: Promediar los valores de un campo.
- **MAX(**<campo>**)**: Obtener el máximo valor de un campo.
- **MIN(**<campo>**)**: Obtener el mínimo valor de un campo.
- **COUNT(**<campo>**)**: Contar los valores (distintos) de un campo.
- **COUNT**(*): Contar los registros de una tabla.

Funciones de Agregación

Permiten responder preguntas:

1. ¿Cuántas casas se vendieron en **total**?

2. ¿Cuánto cuesta la casa más cara? ¿Y la más barata?

3. ¿Cuál es la superficie promedio de las casas de 2 dormitorios?

Funciones de Agregación

Permiten responder preguntas:

1. ¿Cuántas casas se vendieron en **total**?

```sql
SELECT COUNT(*)
FROM houses_pandas;
```

2. ¿Cuánto cuesta la casa más cara? ¿Y la más barat

```sql
SELECT MIN(price), MAX(price)
FROM houses_pandas;
```

3. ¿Cuál es la superficie promedio de las casas de 2 dormitorios?

```sql
SELECT AVG(sqft)
FROM houses_pandas
WHERE bdrms = 2;
```

CONSULTAR UNA BASE DE DATOS

- Práctica individual -

Se puede optar:

- Sobre qué Base de Datos practicar (local SQLite, remota PostgreSQL, ...)

- Cómo interactuar con la misma:
  - Línea de comandos
  - Interfaz gráfica (DB Browser for SQLite, pgAdmin 4, ...)
  - Programa Python

Se debe:

- *Trabajar sobre una tabla que contenga los datos de **housing.csv** (e.g.: tabla **houses**, **houses_pandas**, etc.).*

- *Utilizar la guía de ejercicios de esta clase.*

CONCLUSIONES

- Practicamos distintas maneras de conectarnos a una Base de Datos.
- Una de ellas fue desde un programa Python que escribimos para interactuar con la misma.
- Conocimos estructuras de datos en Python que facilitan la obtención y visualización de los datos.
- Escribimos consultas que nos permitieron obtener cierta **información** acerca de los datos.

PRÁCTICA DE SQL

USO DE BASES DE DATOS DESDE PYTHON