Introducción a la Programación con PYTHON

José Rodriguez

Unidad 9 -Mysql

UNIDAD 11 – MYSQL

ÍNDICE

1.- Iniciando una conexión

En primer lugar debemos asegurarnos que está instalado el paquete PIP. Posteriormente instalamos el paquete de los drivers de MySql

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

Para comprobar que se han instalado correctamente los drivers ejecutamos el script

```
Import mysql.connector
```

Si todo es correcto, podemos empezar a lanzar una conexión

```
Import mysql.connector

micon = mysql.connector.connect(
  host="localhost",
  user="root",
  password="ciclo"
)

print(micon)
```

El resultado debería ser algo así

<mysql.connector.connection.MySQLConnection object at 0x0084E6D0>

La clase mysql.connector es el lazo de unión entre Python y Mysql. Esta clase dispone de numerosos métodos y propiedades entre las cuales destaca el

método connect ya que es el que establece la conexión. Este método puede aceptar muchos parámetros pero los parámetros básicos son host,user y password. Opcionalmente puede interesarnos el puerto, que por defecto es el 3306. Tambien puede ser interesante indicarle en la conexión el nombre de la base de datos que queremos abrir aunque esto lo podemos hacer más adelante.

Otro método interesante es el método cursor cuyo constructor es

micursor = micon.cursor()

El curso es un puntero que apunta a la base de datos. Una conexión puede manipular varias bases de datos, pero debemos crear un cursor para cada una de ellas.

Por ahora vamos a usar el método execute de la subclase cursos, el cual ejecuta cualquier comando SQL válido en Mysql. Por ejemplo, vamos a crear una nueva base de datos

```
Import mysql.connector

micon = mysql.connector.connect(
  host="localhost",
user="root",
password="ciclo"
)
micursor = micon.cursor()
micursor.execute("CREATE DATABASE basenueva")
```

Para comprobar que se ha creado la base de datos lanzamos el comando show databases.

```
micursor.execute("show databases")

for x in micursor:

print(x)
```

Aparecerá una lista con todas las bases de datos del servidor entre las cuales se encuentra la que hemos creado ahora.

('actividades',)
('almacen',)
('basenueva',)
('information_schema',)
….

Otros métodos del cursor son fetchone() que recupera la siguiente fila de una consulta, close() que libera el cursor, etc. También es interesante la propiedad rowcount que devuelve el número de filas en una consulta select.

2.- Creando tablas

Para crear una tabla en mysql debemos ejecutar la sentencia sql apropiada. Por ejemplo, vamos a crear una tabla llamada alumnos en la base de datos anterior. Observa que ahora debemos añadir como parámetro el nombre de la base de datos.

```
Import mysql.connector

micon = mysql.connector.connect(
host="localhost",
user="root",
password="ciclo",
database="basenueva"
)

micursor = micon.cursor()
micursor.execute("CREATE TABLE alumnos (nombre VARCHAR(100), dni VARCHAR(9))")
```

Si ahora lanzamos

```
micursor.execute("show tables")

forx in micursor:
print(x)
```

veremos que ya aparece como única tabla la tabla creada anteriormente.

3.- Insertando registros

Para insertar un registro creamos una estructura formada por la sentencia sql y los valores. Esta pareja (sql, valores) es lo que se pasa como argumento al método execute visto anteriormente. Los parámetros que acompañan a VALUES indican el formato de los valores pasados.

```
Import mysql.connector
micon = mysql.connector.connect(
host="localhost",
user="root",
password="ciclo",
database="basenueva"
)
micursor = micon.cursor()
```

```
sql = "INSERT INTO alumnos(nombre,dni) VALUES (%s, %s)"
val = ("Juan Sanchez", "48987123H")
micursor.execute(sql, val)
micon.commit()
print(micursor.rowcount, " Registros Insertados")
```

Observa que la sentencia micon.commit() es necesaria para que los cambios sean actualizados inmediatamente. Si no escribimos esta instrucción no aparecerá el nuevo registro de forma inmediata.

Si queremos insertar varios registros de una sola vez hemos de hacer los siquientes cambios

```
val = [
  ("Nuria Sanchez",  "3083453H"),
  ("Luis Ramos",  "30455666Q"),
  ("Alicia Montes",  "27876543R"),
  ("Manuel Gómez",  "43545666P")

  ]
micursor.executemany(sql, val)
```

En primer lugar, hemos presentado los valores como un array de tuplas (nombre, dni) y después hemos lanzado el método executemany(sql,val).

4.- Consultas SQL

Para lanzar una consulta invocamos al método execute(consulta) el cual devuelve un cursor y recuperamos en una variable todos los registros del resultado con el método fetchall()

```
Import mysql.connector

micon = mysql.connector.connect(
host="localhost",
user="root",
password="ciclo",
database="basenueva"
)
micursor = micon.cursor()
micursor.execute("SELECT * FROM alumnos")
resultado = micursor.fetchall()

for x in resultado:
print(x)
```

Observa que el resultado aparece como un array de tuplas(nombre,dni)

('Juan Sanchez', '48987123H')

('María Sanchez', '48983453H')
('Nuria Sanchez', '3083453H')
('Luis Ramos', '30455666Q')
('Alicia Montes', '27876543R')
('Manuel Gómez', '43545666P')
('Nuria Sanchez', '45645345T')

Si sólo estamos interesados en recuperar un registro utilizamos el método fetchone()

```
resultado = micursor.fetchone()
print(resultado)
```

En general, usar fetchone() está ligado a un bucle en el cual necesitamos saber el número de registros devueltos por la consulta. Para ello usamos la propiedad rowcount. Está propiedad va siempre detrás de una búsqueda completa con el método fetchall()

```
micursor = micon.cursor()
resultado = micursor.rowcount
micursor.execute("SELECT * FROM alumnos")
resultado = micursor.fetchall()
print(micursor.rowcount, " Registros")
```

Hemos visto que podemos recorrer el resultado de la consulta con el bucle

for x in resultado:
print(x)

Cada línea x es una pareja de valores (nombre,dni). SI queremos manejar de forma independiente ambos valores escogemos el elemento correspondiente del array mediante un índice (0 para el primero, etc). En este ejemplo mostramos sólo el nombre del alumno (índice 0)

```
Import mysql.connector

micon = mysql.connector.connect(
host="localhost",
user="root",
password="ciclo",
database="basenueva"
)
micursor = micon.cursor()
resultado = micursor.rowcount
micursor.execute("SELECT * FROM alumnos")
registro = micursor.fetchone()
while(registro != None):
print(registro[0])
   registro = micursor.fetchone()
```

El resultado es

Juan Sanchez
María Sanchez
Nuria Sanchez
Luis Ramos
Alicia Montes
Manuel Gómez
Nuria Sanchez

4.- Borrar registros

Para borrar un conjunto de registros se lanza la sentencia sqlestandaddelete con la condición correspondiente

```
Import mysql.connector

micon = mysql.connector.connect(
host="localhost",
user="root",
password="ciclo",
database="basenueva"
)
micursor = micon.cursor()
sql = "DELETE FROM alumnos WHERE dni = '3083453H'"
micursor.execute(sql)
micon.commit()
print(micursor.rowcount, "registros borrados")
```

De forma similar podemos borrar una tabla lanzando la sentencia sql apropiada

```
Import mysql.connector

micon = mysql.connector.connect(
host="localhost",
user="root",
password="ciclo",
database="basenueva"
)
micursor = micon.cursor()
sql = "DROP TABLE alumnos"

micursor.execute(sql)
```

Para evitar errores podemos añadirle la clausula IF EXISTS

```
sql = "DROP TABLE IF EXISTS customers"
```