GUÍA 4
MySQL : otras funcionalidades del paquete mysql-connector

MySQL : otras funcionalidades del paquete mysql-connector

Hemos resuelto en los dos conceptos anteriores algoritmos en Python que acceden a una base de datos de MySQL. Veremos otras funcionalidades que nos provee el paquete mysql-connector.

Recuperar el valor del campo auto_increment en un insert

Cuando insertamos una fila en una tabla que contiene un campo que se auto incrementa podemos recuperar dicho valor en el mismo momento que efectuamos la inserción.

```
import mysql.connector


conexion1=mysql.connector.connect(host="localhost",
                        user="root",
                        passwd="",
                        database="bd1")
cursor1=conexion1.cursor()
sql="insert into articulos(descripcion, precio) values (%s,%s)"
datos=("naranjas", 23.50)
cursor1.execute(sql, datos)
conexion1.commit()
print("Valor del último codigo de artículo generado:", cursor1.lastrowid)
conexion1.close()
```

Luego de efectuar la inserción de una fila en la tabla 'articulos' procedemos a acceder al atributo 'lastrowid' que almacena el código de artículo que se acaba de generar.

Inserción de múltiples filas en una tabla.

La clase cursor a parte del método 'execute' cuenta con otro método llamado 'executemany' que tiene el objetivo de insertar múltiples filas de una tabla.

```
import mysql.connector


conexion1=mysql.connector.connect(host="localhost",
                        user="root",
                        passwd="",
                        database="bd1")
cursor1=conexion1.cursor()
sql="insert into articulos(descripcion, precio) values (%s,%s)"
filas=[ ("naranjas", 23.50),
      ("bananas", 34),
      ("peras", 21),
      ("sandía", 19.60) ]
cursor1.executemany(sql, filas)
conexion1.commit()
conexion1.close()
```

El método 'executemany' debe recibir en el segundo parámetro una lista cuyos elementos sean una tupla por cada fila:

```
sql="insert into articulos(descripcion, precio) values (%s,%s)"
filas=[ ("naranjas", 23.50),
      ("bananas", 34),
      ("peras", 21),
      ("sandía", 19.60) ]
cursor1.executemany(sql, filas)
```

Si accedemos luego a la propiedad 'lastrowid' nos devuelve el código del último código de artículo almacenando.

Creación de una base de datos y tablas.

En conceptos anteriores vimos como crear una base de datos de MySQL utilizando la aplicación PHPMyAdmin, en algunas situaciones podemos necesitar crear una base de datos desde el mismo programa de Python. La misma metodología será si queremos crear tablas.

```
import mysql.connector


conexion1=mysql.connector.connect(host="localhost",
                          user="root",
                          passwd="")
cursor1=conexion1.cursor()
sql="create database bd2"
cursor1.execute(sql)

sql="use bd2"
cursor1.execute(sql)

sql="""create table usuarios (
       nombre varchar(30) primary key,
       clave  varchar(30)
  )"""
cursor1.execute(sql)

conexion1.commit()
conexion1.close()
```

Lo primero que debemos notar que en la conexión al servidor de base de datos MySQL no indicamos una "base de datos":

```
conexion1=mysql.connector.connect(host="localhost",
                          user="root",
                          passwd="")
```

Luego de crear el cursor procedemos a llamar al método 'execute' y le pasamos el comando SQL 'create database' con el nombre de base de datos a crear:

```
cursor1=conexion1.cursor()
sql="create database bd2"
cursor1.execute(sql)
```

Luego de crear la base de datos si queremos activarla para poder crear tablas, insertar filas en tablas, consultarlas etc. debemos activarla mediante el comando SQL 'use' con el nombre de la base de datos:

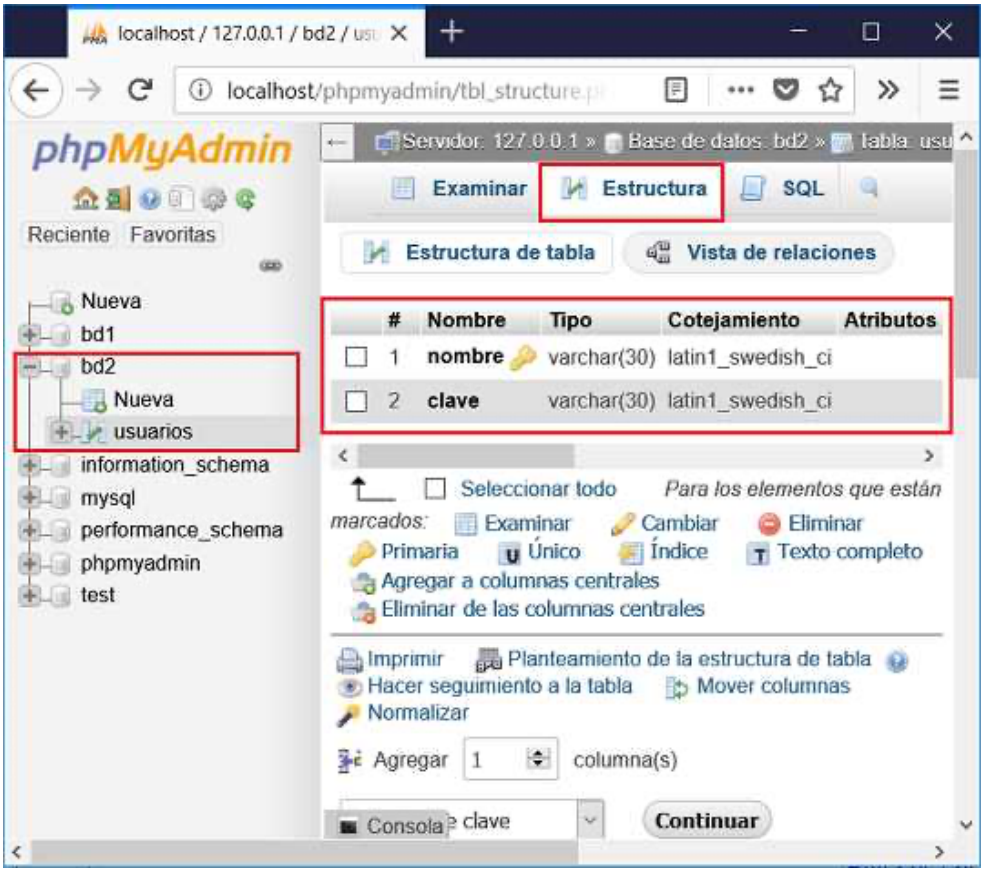
```
sql="use bd2"
cursor1.execute(sql)
```

Ya estando en uso la base de datos 'bd2' podemos efectuar la creación de una tabla:

```
sql="""create table usuarios (
      nombre varchar(30) primary key,
      clave  varchar(30)
  )"""
cursor1.execute(sql)
```

Recordemos que las triple comillas nos permiten en Python definir un string de múltiples líneas.

Luego de ejecutar este programa si entramos a PHPMyAdmin veremos que se ha creado la base de datos 'bd2' y dentro de esta la tabla 'usuarios':



Se ejecutamos nuevamente el programa veremos que se genera un error indicando que la base de datoas 'bd2' ya existe.

Borrar una base de datos.

Cuando queremos crear una base de datos que ya existe se genera un error, podemos primero borrarla y luego ya si crearla sin problemas.

```
import mysql.connector


conexion1=mysql.connector.connect(host="localhost",
                          user="root",
                          passwd="")
cursor1=conexion1.cursor()


sql="drop database if exists bd2"
cursor1.execute(sql)


sql="create database bd2"
cursor1.execute(sql)


sql="use bd2"
cursor1.execute(sql)


sql="""create table usuarios (
       nombre varchar(30) primary key,
       clave  varchar(30)
  )"""
cursor1.execute(sql)


conexion1.commit()
conexion1.close()
```

Este programa es una variante del anterior donde primero verificamos si existe la base de datos 'bd2' y en caso afirmativo la eliminamos:

```
sql="drop database if exists bd2"
cursor1.execute(sql)
```

Una vez que borramos la base de datos 'bd2' en el caso que exista procedemos a crearla:

```
sql="create database bd2"
cursor1.execute(sql)
```

Borrar una tabla y crear otra con el mismo nombre.

No se puede crear una tabla con el mismo nombre de una existente, en esos casos debemos borrar la actual y crear una nueva.

```
import mysql.connector


conexion1=mysql.connector.connect(host="localhost",
                          user="root",
                          passwd="",
                          database="bd2")
cursor1=conexion1.cursor()
sql="drop table if exists usuarios"
cursor1.execute(sql)


sql="""create table usuarios (
       nombre varchar(30) primary key,
       clave  varchar(30)
  )"""
cursor1.execute(sql)


conexion1.commit()
conexion1.close()
```

Para eliminar la tabla 'usuarios' si existe ejecutamos el siguiente comando SQL:

```
sql="drop table if exists usuarios"
```

```
cursor1.execute(sql)
```

Una vez eliminada la tabla podemos crearla nuevamente:

```
sql="""create table usuarios (
        nombre varchar(30) primary key,
        clave  varchar(30)
   )"""
cursor1.execute(sql)
```