Licence Ingénierie Logicielle et Système d'Information

TP N°5 Base de données en python avec MySQL

Pour créer une base de données dans MySQL, utilisez l'instruction "CREATE DATABASE":

1. Crée une base de données nommée « mydatabase »
2. Vérifier si la base de données existe
3. Vous pouvez vérifier si une base de données existe en répertoriant toutes les bases de données de votre système à l'aide de l'instruction «SHOW DATABASES»:
4. Connexion à la base de données

demo_mysql_show_databases.py:

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="myusername",
  password="mypassword"
)

mycursor = mydb.cursor()

mycursor.execute("SHOW DATABASES")

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

5. Crée une table Pour créer une table dans MySQL, utilisez l'instruction "CREATE TABLE".

Nb : Assurez-vous de définir le nom de la base de données lorsque vous créez la connexion

Créer une table nommée « customers »

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))")
```

6. Vérifier si la table existe

Vous pouvez vérifier si une table existe en listant toutes les tables de votre base de données avec l'instruction "SHOW TABLES".

7. Clé primaire

Lors de la création d'une table, vous devez également créer une colonne avec une clé unique pour chaque enregistrement. Cela peut être fait en définissant une CLÉ PRIMAIRE.

Nous utilisons l'instruction "INT AUTO_INCREMENT PRIMARY KEY" qui insérera un numéro unique pour chaque enregistrement. À partir de 1, et augmenté de un pour chaque enregistrement.

7.1. Si la table existe déjà, utilisez le mot clé ALTER TABLE :

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO_INCREMENT PRIMARY KEY")
```

8. L'instruction "INSERT INTO".

Insérez un enregistrement dans la table "clients":

```
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("John", "Highway 21")
mycursor.execute(sql, val)

mydb.commit()

print(mycursor.rowcount, "record inserted.")
```

9. Insérer plusieurs lignes

Pour insérer plusieurs lignes dans un tableau, utilisez la `executemany()` méthode.

Le deuxième paramètre de la `executemany()` méthode est une liste de tuples, contenant les données que vous souhaitez insérer.

10. Obtenir l'ID inséré

Vous pouvez obtenir l'identifiant de la ligne que vous venez d'insérer en demandant à l'objet curseur.

```python
import mysql.connector

mydb = mysql.connector.connect(
  host="localhost",
  user="yourusername",
  password="yourpassword",
  database="mydatabase"
)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"
val = ("Michelle", "Blue Village")
mycursor.execute(sql, val)

mydb.commit()

print("1 record inserted, ID:", mycursor.lastrowid)
```

11. L'instruction SELECT

Sélectionnez tous les enregistrements du tableau "clients" et affichez le résultat.

12. Sélectionnez uniquement les colonnes de nom et d'adresse (fetchall()) :
13. L'utilisation de fetchone()

```
mycursor = mydb.cursor()

mycursor.execute("SELECT * FROM customers")

myresult = mycursor.fetchone()

print(myresult)
```

14. La sélection avec filtrage à l'aide de l'instruction "WHERE"
Sélectionnez le (s) enregistrement (s) dont l'adresse est "Park Lane 38":
résultat:

```
mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address ='Park Lane 38'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

Vous pouvez également sélectionner les enregistrements qui commencent, incluent ou se terminent par une lettre ou une phrase donnée.

Utilisez le `%` pour représenter les caractères génériques:

15. Sélectionnez les enregistrements dont l'adresse contient le mot « way » :

```
mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address LIKE '%way%'"

mycursor.execute(sql)

myresult = mycursor.fetchall()

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

16. Empêcher l'injection SQL

Lorsque les valeurs de requête sont fournies par l'utilisateur, vous devez échapper les valeurs.

Cela permet d'éviter les injections SQL, qui est une technique de piratage Web courante pour détruire ou abuser votre base de données.

Le module mysql.connector a des méthodes pour échapper aux valeurs de requête:

```
mycursor = mydb.cursor()

sql = "SELECT * FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

myresult = mycursor.fetchall()

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

17. Utilisez l'instruction ORDER BY pour trier le résultat par ordre croissant ou décroissant.

18. Utilisez le mot clé DESC pour trier le résultat dans un ordre décroissant.
19. Supprimez tout enregistrement dont l'adresse est "Mountain 21".
20. Supprimer un enregistrement

```
mycursor = mydb.cursor()

sql = "DELETE FROM customers WHERE address = %s"
adr = ("Yellow Garden 2", )

mycursor.execute(sql, adr)

mydb.commit()

print(mycursor.rowcount, "record(s) deleted")
```

21. Mettre à jour la table

```
mycursor = mydb.cursor()

sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'"

mycursor.execute(sql)

mydb.commit()

print(mycursor.rowcount, "record(s) affected")
```

22. Supprimer la table "customers":