12. Database

Table of Contents

|  |  |  |
| --- | --- | --- |
| Sl. No | Topics | Page No |
| 12.1 | **Python Database** |  |

## 12.1 Python Database

**What are Python DB API?**

* The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.
* You can choose the right database for your application. Python Database API supports a wide range of database servers such as −
  + GadFly
  + mSQL
  + MySQL
  + PostgreSQL
  + Microsoft SQL Server 2000
  + Informix
  + Interbase
  + Oracle
  + Sybase
* Here is the list of available Python database interfaces: [Python Database Interfaces and APIs](http://wiki.python.org/moin/DatabaseInterfaces). You must download a separate DB API module for each database you need to access. For example, if you need to access an Oracle database as well as a MySQL database, you must download both the Oracle and the MySQL database modules.
* The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following −
  + Importing the API module.
  + Acquiring a connection with the database.
  + Issuing SQL statements and stored procedures.
  + Closing the connection

**MySQLdb**

* MySQLdb is an interface for connecting to a MySQL database server from Python. It implements the Python Database API v2.0 and is built on top of the MySQL C API.
* Just type the following in your Python script and execute it − import MySQLdb

**Transactions**

**What is Transaction? What are ACID Property?**

Transactions are a mechanism that ensures data consistency.

Transactions have the following four properties:-

* **Atomicity** − either a transaction completes or nothing happens at all.
* **Consistency** − A transaction must start in a consistent state and leave the system in a consistent state.
* **Isolation** − Intermediate results of a transaction are not visible outside the current transaction.
* **Durability** − Once a transaction was committed, the effects are persistent, even after a system failure.

The Python DB API provides two methods to either *commit* or *rollback* a transaction.

* **COMMIT Operation**

Commit is the operation, which gives a green signal to database to finalize the changes, and after this operation, no change can be reverted back. db.commit()

* **ROLLBACK Operation**

If you are not satisfied with one or more of the changes and you want to revert back those changes completely, then use rollback() method. db.rollback()

* **Disconnecting Database**

To disconnect Database connection, use close() method. db.close()

**Python-MySQL Database**

**Python MySQL Setup**

* Python can be used in database applications. One of the most popular databases is MySQL.
* You can download a free **MySQL database** at <https://www.mysql.com/downloads/>
* Download and install "**MySQL Connector**":-

C:\Users\*Your Name*\AppData\Local\Programs\Python\Python36-32\Scripts>

python -m pip install mysql-connector

* To **test if the installation was successful**, or if you already have "MySQL Connector" installed, create a Python page with the following content: (demo\_mysql\_test.py)

import mysql.connector

* Start by creating a connection to the database.
* Use the username and password from your MySQL database: (demo\_mysql\_connection.py)

Import mysql.connector  
mydb = mysql.connector.connect (  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*"  
)  
  
print(mydb)

**Create Database**

**Create Database**

* To create a database in MySQL, use the "CREATE DATABASE" statement.
* Create a database named "mydatabase"

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*"  
)  
mycursor = mydb.cursor()  
mycursor.execute("CREATE DATABASE mydatabase")

**Connecting Database**

* Try connecting to the database "mydatabase“

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

**Create Table**

**Create Table using Primary Key**

* To create a table in MySQL, use the "CREATE TABLE" statement.
* Make sure you define the name of the database when you create the connection

**Step 1: Create primary key when creating the table**

import mysql.connector  
 mydb = mysql.connector.connect(  
  host="localhost",  
   user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
 )  
  
 mycursor = mydb.cursor()  
  
 mycursor.execute("CREATE TABLE customers (id INT AUTO\_INCREMENT PRIMARY KEY, name VARCHAR(255), address VARCHAR(255))")

**Step 2: Create Primary Key on an existing table**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
mycursor.execute("ALTER TABLE customers ADD COLUMN id INT AUTO\_INCREMENT PRIMARY KEY")

**Step 3: To return a list of tables in the database**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
mycursor = mydb.cursor()  
mycursor.execute("SHOW TABLES")

for x in mycursor:  
  print(x)

**Insert Table**

**Insert a record into a table**

**Step 1: Insert a record in the "customers" table**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*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.")

**Step 2: Insert multiple record into a table**

* To insert multiple rows into a table, use the executemany() method.
* The second parameter of the executemany() method is a list of tuples, containing the data you want to insert:

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="yourusername",

passwd="yourpassword",

database="mydatabase"

)

mycursor = mydb.cursor()

sql = "INSERT INTO customers (name, address) VALUES (%s, %s)"

val = [

('Peter', 'Lowstreet 4'),

('Amy', 'Apple st 652'),

]

mycursor.executemany(sql, val)

mydb.commit()

print(mycursor.rowcount, "was inserted.")

**Select Table**

**Step 1: Select all records from Customers table**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
mycursor = mydb.cursor()  
mycursor.execute("SELECT \* FROM customers")  
myresult = mycursor.fetchall()  
for x in myresult:  
  print(x)

**Step 2: Selecting Columns from the table**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
mycursor = mydb.cursor()  
mycursor.execute("SELECT name, address FROM customers")  
myresult = mycursor.fetchall()  
for x in myresult:  
  print(x)

**Where**

**Usage of Where clause**

import mysql.connector

mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
mycursor = mydb.cursor()  
sql = "SELECT \* FROM customers WHERE address =' Lowstreet 4 '“

mycursor.execute(sql)  
myresult = mycursor.fetchall()  
  
for x in myresult:  
  print(x)

**Order By**

**Usage of Order by clause**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
sql = "SELECT \* FROM customers ORDER BY name"  
  
mycursor.execute(sql)  
myresult = mycursor.fetchall()  
  
for x in myresult:  
  print(x)

**Delete**

**Prevent SQL Injection**

* It is considered a good practice to escape the values of any query, also in delete statements.
* This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database.
* The mysql.connector module uses the placeholder %s to escape values in the delete statement:
* You can delete records from an existing table by using the "DELETE FROM" statement:
* Escape values by using the placeholder %s method:

**To delete a record**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
sql = "DELETE FROM customers WHERE address = %s"  
adr = (" Lowstreet 4 ", )  
  
mycursor.execute(sql, adr)  
  
mydb.commit()  
print(mycursor.rowcount, "record(s) deleted")

**Drop Table**

**Dropping Table Customers**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
  
sql = "DROP TABLE IF EXISTS customers"  
  
mycursor.execute(sql)

**Update**

**Update Table**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
sql = "UPDATE customers SET address = %s WHERE address = %s"  
val = (" Lowstreet 4 ", "Canyon 123")  
  
mycursor.execute(sql, val)  
  
mydb.commit()  
print(mycursor.rowcount, "record(s) affected")

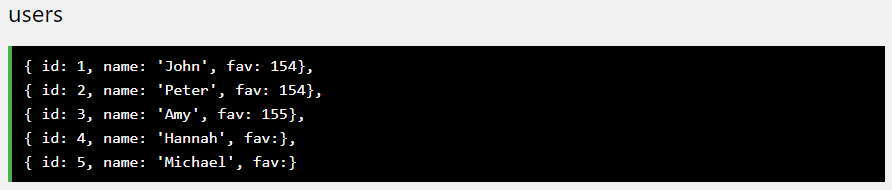
**Limit**

**Usage of Limit**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
  
mycursor.execute("SELECT \* FROM customers LIMIT 5 OFFSET 2")  
  
myresult = mycursor.fetchall()  
  
for x in myresult:  
  print(x)

**Join**

You can combine rows from two or more tables, based on a related column between them, by using a JOIN statement. Consider you have a "users" table and a "products" table, these two tables can be combined by using users' fav field and products' id field.





**Join two or more tables**

import mysql.connector  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  passwd="*yourpassword*",  
  database="mydatabase"  
)  
  
mycursor = mydb.cursor()  
  
sql = "SELECT \  
  users.name AS user, \  
  products.name AS favorite \  
  FROM users \  
  INNER JOIN products ON users.fav = products.id"

mycursor.execute(sql)  
  
myresult = mycursor.fetchall()  
  
for x in myresult:  
  print(x)

**Left Join**

* INNER JOIN only shows the records where there is a match.
* If you want to show all users, even if they do not have a favorite product, use the LEFT JOIN statement:

**Example**: Select all users and their favorite product:

sql = "SELECT \  
  users.name AS user, \  
  products.name AS favorite \  
  FROM users \  
  LEFT JOIN products ON users.fav = products.id"

**Right Join**

* If you want to return all products, and the users who have them as their favorite, even if no user have them as their favorite, use the RIGHT JOIN statement:
* **Example**: Select all products, and the user(s) who have them as their favorite:

sql = "SELECT \  
  users.name AS user, \  
  products.name AS favorite \  
  FROM users \  
  RIGHT JOIN products ON users.fav = products.id"

**Python-sqllite3**

**Exercise: Connecting DB using sqlite3 module and performing CRUD operations**

**Step 1: Establishing Connection with sqllite3**

* sqlite3 module is used to perform CRUD operations and connect DB with python.

import sqlite3

conn = sqlite3.connect('company.db')

**Step 2: CRUD Operations**

**#Table Creation**

'''

conn.execute("create table jobmarket(eid int,ename text)")

print("table is created")

'''

**#Data Insertion**

'''

conn.execute("insert into jobmarket(eid,ename) \

values (1, 'Paul')");

conn.execute("INSERT INTO jobmarket(eid,ename) \

values (2, 'Allen' )");

conn.execute("insert into jobmarket(eid,ename) \

values(3,'John')");

conn.commit();

print("records are inserted");

'''

**#Retrieving data**

'''

cursor = conn.execute("select eid,ename from jobmarket");

for row in cursor:

print("EID:",row[0]);

print("ENAME:",row[1]);

'''

'''

conn.execute("update jobmarket set ename='sachin' where eid=1");

conn.commit()

cursor = conn.execute("select eid,ename from jobmarket");

for row in cursor:

print("EID:",row[0]);

'''

**#Delete operation**

'''

conn.execute("delete from jobmarket where eid=1");

print("total:",conn.total\_changes);

conn.commit()

cursor = conn.execute("select eid,ename from jobmarket");

for row in cursor:

print("EID:",row[0]);

print("ENAME:",row[1]);

'''

