Database Handling

What is Database?

Database is a collection of data in the form of tables.

Tables are objects which stores data.

- Database is a systematic collection of data.
- Databases support storage and manipulation of data.
- Databases make data management easy
- It is collection of meaningful or designed data which is used to create and maintain database
- Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, schemas, reports etc. .

There are many types of database:

1) MySQL

2) Oracle

3) Postre SQL

What is DBMS?

- Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, reporting / representation of data .
- It also helps to control access to the database.

Three main types of databases:-

- **Hierarchical -** this type of DBMS employs the "parent-child" relationship of storing data.

- his type of DBMS is rarely used nowadays.
- Its structure is like a tree with nodes representing records and branches representing fields.
- The windows registry used in Windows XP is an example of a hierarchical database.
- Configuration settings are stored as tree structures with nodes.
- **Network DBMS** - this type of DBMS supports many-to many relations.
- This usually results in complex database structures.
- RDM Server is an example of a database management system that implements the network model.
- **Relational DBMS** - this type of DBMS defines database relationships in form of tables, also known as relations
- . Unlike network DBMS, RDBMS does not support many to many relationships.
- Relational DBMS usually have pre-defined data types that they can support.
- This is the most popular DBMS type in the market.
- Examples of relational database management systems include MySQL, Oracle, and Microsoft SQL Server database.

Advantages of DBMS

- **Minimized redundancy and data consistency**
- **Data Security**
- **Simplified Data Access**
- **Concurrent access to data**
- **Backup and Recovery mechanism**

Database Connection

There are the following steps to connect a python application to our database.

1. Import mysql.connector module
2. Create the connection object.
3. Create the cursor object
4. Execute the query

Creating the connection

To create a connection between the MySQL database and the python application, the connect() method of mysql.connector module is used.

Pass the database details like HostName, username, and the database password in the method call. The method returns the connection object.

```
import pymysql

#Create the connection object
myconn = pymysql  .connect(host = "localhost", user = "root",passwd = "roo
t")

#printing the connection object
print(myconn)
```

Output

```
<mysql.connector.connection.MySQLConnection object at 0x7fb142edd780>
```

```
import pymysql

#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd = "ro
ot",
 database = "mydb")

#printing the connection object
print(myconn)
```

Creating a cursor object

The cursor object can be defined as an abstraction specified in the Python DB-API 2.0. It facilitates us to have multiple separate working environments through the same connection to the database. We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.

The syntax to create the cursor object is given below.

my_cur = conn.cursor()

```
import pymysql
#Create the connection object
myconn = mysql.connector.connect(host = "localhost", user = "root",passwd
 = "root, database = "mydb")

#printing the connection object
print(myconn)
  #creating the cursor object
cur = myconn.cursor()

print(cur)
```

Creating the table

```
import pymysql
#Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database="mydatabase")
# creating the cursor object
cur = myconn.cursor()
# Creating a table with name Employee having four columns i.e., name, id, salary,
and department id
dbs = cur.execute(
```

```
        "create table Employee1(name varchar(20), id int(20), salary int(20)")
print("Success")
myconn.close()
```

execute() :-is a function of cursor to write sql queries

Insert row in a table

```
import pymysql
# Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database=" mydatabase ")
# creating the cursor object
cur = myconn.cursor()
sql = "insert into Employee values(name, id, salary, dept_id, branch_name)
values (%s, %s, %s, %s, %s)"
# The row values are provided in the form of tuple
val = ("John", 110, 25000.00, 201, "Newyork")
# inserting the values into the table
cur.execute(sql, val)
# commit the transaction
myconn.commit()
print(cur.rowcount, "record inserted!")
myconn.close()
```

commit():-is a function which will always be used for making permanent changes in the databases as insert, update and delete.

Note :-Always use commit function when data is inserted, updated or deleted.

Insert multiple rows in table

```
import pymysql
# Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database="PythonDB1")
# creating the cursor object
```

```
cur = myconn.cursor()
sql = "insert into Employee(name, empid, salary) values (%s, %s, %s)"
val = [("John", 102, 25000),
       ("David", 103, 25000),
       ("Nick", 104, 90000,)]

# inserting the values into the table
cur.executemany(sql, val)
# commit the transaction myconn.commit()
print(cur.rowcount, "records inserted!")
myconn.close()
```

executemany():a function insert list of multiple records at once.

Fetch all Data:-

```
import pymysql
# Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database="PythonDB1")
# creating the cursor object
cur = myconn.cursor()
# Reading the Employee data
cur.execute("select * from Employee")
# fetching the rows from the cursor object
result = cur.fetchall()
# printing the result

for x in result:
      print(x)


myconn.close()
```

fetchall(): a function to display all the records by using a sql query select * from table or using any where condition

Fetch Only one record

```
import pymysql
# Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database="PythonDB1")
# creating the cursor object
cur = myconn.cursor()
# Reading the Employee data
cur.execute("select * from Employee where empid=101")
# fetching the rows from the cursor object
result = cur.fetchone()
# printing the result

for x in result:
    print(x)

myconn.close()
```

fetchone(): a function to display only one record by filtering data using where condition.

Update the data :

To make changes to the existing data by updating with new value.

We should always use where condition while updating or else all the records will be updated.

```
import pymysql
# Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database="PythonDB1")
# creating the cursor object
cur = myconn.cursor()
# updating the name of the employee whose id is 110
```

```
cur.execute("update Employee set name = 'alex' where id = 110")
myconn.commit()
myconn.close()
```

Delete the data

To delete a record in the table

We should always use where condition while deleting .

```
import pymysql
# Create the connection object
myconn = pymysql.connect(host="localhost", user="root", passwd="root",
database="PythonDB1")
# creating the cursor object
cur = myconn.cursor()
# updating the name of the employee whose id is 110
cur.execute("delete from Employee where id = 110")
myconn.commit()
myconn.close()
```

Task to be done:

Create functions for all the following operations to be performed on databases:
1)createTable()
2)insertRecord()
3)UpdateRecord()
4)Deleterecord()

Commit the changes as required and call the functions respectively.