CH 13: Python Database Access

In this tutorial, I will discuss about interactions with database using python. We choose MySQL as our database. In this chapter, we will covers the basics of MySQL programming like insertion, updating and deletion of record with Python using MySQLdb module. The examples were created and tested on Ubuntu Linux 12.04LTS.

**13. 1 Requirements:**  Before proceeding to the details of python database access; we need to have MySQL installed and properly setup in our system. Please do the following step for configuration:

If you don't already have MySQL installed, we must install it using following command

**$ sudo apt-get install mysql-server**

During installation it will ask for the root password, please give any password ( say root123) to proceed installation.

Now install python-mysqldb for accessing MySQL DB from python programs:

**$ sudo apt-get install python-mysqldb**

**13.2 getting start with MySQL client:**

Frist, we are going to create new database user and a new database using mysql client( Mysql shell, not from python). Please go through the following steps:

STEP1: We connect to the database using the root account. We show all available databases with the SHOW DATABASES statement.

Next, we are going to create a new database user and a new database. We use the mysql client.

$ **mysql -u root -p**

Enter password: <ENTER YOUR PASSWORD which was given while installation >

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 42

Server version: 5.5.34-0ubuntu0.12.04.1 (Ubuntu)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective

owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> **SHOW DATABASES;**

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| test |

+--------------------+

4 rows in set (0.00 sec)

mysql>

STEP2 : Create a database, named “pythondb” and a user named “pythonuser” We will use this database throughout this chapter.

mysql> CREATE DATABASE pythondb ;

Query OK, 1 row affected (0.02 sec)

mysql> CREATE USER 'pythonuser'@'localhost' IDENTIFIED BY 'test623';

Query OK, 0 rows affected (0.00 sec)

mysql> USE pythondb;

Database changed

STEP 3: Grant all permission to “pythonuser” for accessing “pythondb” database:

mysql> GRANT ALL ON pythondb.\* TO 'pythonuser'@'localhost';

Query OK, 0 rows affected (0.00 sec)

mysql> quit;

Bye

**13.2 DataBase access suing MySQLdb module:**

MySQLdb is a thin Python wrapper for accessing mySQL database easily. It is compatible with the Python DB API, which makes the code more portable. Using this model is the preferred way of working with the MySQL. In this section we will show you how we can perform the following operation:

1. Create table in our database
2. Adding entry in the table
3. Quire entry from the table
4. Updating entry in the table
5. Delete entry from the table
6. Delete the whole table.

Before starting any of the above, let’s write a simple script, which can connect the database and print the database version:

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("SELECT VERSION()")

ver = cur.fetchone()

print "Database version : %s " % ver

con.close()

except mdb.Error, e:

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

In the above example mdb.connect() is used to connect the database, return a connection object, say con. Using this con object we can get cursor object to access the database or execute any quires. cur.execue() command is used run any command, just like writing in a mysql shell. cur.fetchone() is used to retrieve queries result and print it.

13.2.1 **Creating table and population information:**

In this tutorial, we are going to create a student table, having two fields <Roll No and Name of the student> and insert some student info. Roll Number is auto incremented and acts as a primary key, hence to be unique for all students.

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("DROP TABLE IF EXISTS Student")

cur.execute("CREATE TABLE Student (roll INT PRIMARY KEY AUTO\_INCREMENT, \

Name VARCHAR(25))")

cur.execute("INSERT INTO Student (Name) VALUES('John')")

cur.execute("INSERT INTO Student (Name) VALUES('Jack')")

cur.execute("INSERT INTO Student (Name) VALUES('Mila')")

cur.execute("INSERT INTO Student (Name) VALUES('Zola')")

cur.execute("INSERT INTO Student (Name) VALUES('Ram')")

con.close()

except mdb.Error, e:

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

This program will create Student table and add five students into it. We can verify from MySQL shell:

mysql> SELECT \* FROM Student;

+----+-------------------+

| Roll | Name |

+----+-------------------+

| 1 | Jack London |

| 2 | Honore de Balzac |

| 3 | Lion Feuchtwanger |

| 4 | Emile Zola |

| 5 | Truman Capote |

+----+-------------------+

5 rows in set (0.00 sec)

13.2.2 **Retrieving student info from Student Table.**

Let’s get back the information, what we have inserted in the previous program:

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("SELECT \* FROM Student")

rows = cur.fetchall()

for row in rows:

print row

con.close()

except mdb.Error, e:

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

cur.fetchall() will return all student information from Database and the next loop with display the information one by one.:

Output :

$ ./retrieve.py

(1L, 'Jack London')

(2L, 'Honore de Balzac')

(3L, 'Lion Feuchtwanger')

(4L, 'Emile Zola')

(5L, 'Truman Capote')

Sometime, returning all data at a time may not be feasible. We can fetch rows one by one using fatchone() method.

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("SELECT \* FROM Student")

for i in range(cur.rowcount):

row = cur.fetchone()

print row[0], row[1]

# Commit your changes in the database

con.commit()

except mdb.Error, e:

# Rollback in case there is any error

con.rollback()

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

con.close()

Here, we fetch the rows one by one using the fetchone() method. The rowcount property gives the number of rows returned by the SQL statement.

Output:

1 Jack London

2 Honore de Balzac

3 Lion Feuchtwanger

4 Emile Zola

5 Truman Capote

As a summary, we can say :

* fetchone(): This method fetches the next row of a query result set.
* fetchall(): This method fetches all the rows in a result set
* rowcount: Returns the number of rows that were affected by an execute() method.

13.2.2 **Getting column header:**

Suppose, we want to print the data with their column header, It can be single done using **cur.description** properties. Below example shows this:

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("SELECT \* FROM Student")

desc = cur.description

print "%s %3s" % (desc[0][0], desc[1][0])

for i in range(cur.rowcount):

row = cur.fetchone()

print row[0], row[1]

# Commit your changes in the database

con.commit()

except mdb.Error, e:

# Rollback in case there is any error

con.rollback()

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

con.close()

**13.2.2 Updating student data:**

Suppose, we want to update the name of the student having roll no 4, we can do it by executing the following command: UPDATE Student SET Name = Dipankar WHERE Roll = 4

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("UPDATE Student SET Name = %s WHERE Roll = %s",

("Dipankar", "4"))

print "Number of rows updated:", cur.rowcount

# Commit your changes in the database

con.commit()

except mdb.Error, e:

# Rollback in case there is any error

con.rollback()

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

con.close()

**Output:**

$ ./prepared.py

Number of rows updated: 1

We can also validate from mysql shell, as below:

mysql> SELECT Name FROM Writers WHERE Id=4;

+------------------+

| Name |

+------------------+

| Dipankar |

+------------------+

1 row in set (0.00 sec)

**13.2.2 Delete student data:**

Suppose, we want to update the name of the student having roll no 4, we can do it by executing the following command: UPDATE Student SET Name = Dipankar WHERE Roll = 4

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("DELETE FROM Student WHERE Roll > 5”)

print "Number of rows updated:", cur.rowcount

# Commit your changes in the database

con.commit()

except mdb.Error, e:

# Rollback in case there is any error

con.rollback()

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

con.close()

**Output:**

$ ./prepared.py

Number of rows updated: 1

**13.2.2 Delete student table:**

Suppose, we want to update the name of the student having roll no 4, we can do it by executing the following command: UPDATE Student SET Name = Dipankar WHERE Roll = 4

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("DROP TABLE IF EXISTS Student")

# Commit your changes in the database

con.commit()

except mdb.Error, e:

# Rollback in case there is any error

con.rollback()

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

con.close()

**Output:**

$ ./prepared.py

Number of rows updated: 1

**3.3 Transaction support by MySQLdb**

A transaction is an atomic unit of database operations against the data in one or more databases. It is basically ALL or None operation. That means, all of the database operation should execute or none of them will execute. More technically, the effects of all the SQL statements in a transaction be either all committed to the database or all rolled back.

For databases that support transactions, the python interface silently starts a transaction when the cursor is created. The commit() method commits the updates made using that cursor, and the rollback() method discards them. Each method starts a new transaction.

As in the first example, if we something wrong in middle of the some insertion, you will find nothing has inserted in the table. As an example:

import MySQLdb as mdb

import sys

try:

con = mdb.connect('localhost', 'pythonuser', 'test623', 'pythondb');

cur = con.cursor()

cur.execute("DROP TABLE IF EXISTS Student")

cur.execute("CREATE TABLE Student (roll INT PRIMARY KEY AUTO\_INCREMENT, \

Name VARCHAR(25))")

cur.execute("INSERT INTO Student (Name) VALUES('John')")

cur.execute("INSERT INTO Student (Name) VALUES('Jack')")

########### Make some wrong here:

Print xyz #Error.

cur.execute("INSERT INTO Student (Name) VALUES('Mila')")

cur.execute("INSERT INTO Student (Name) VALUES('Zola')")

cur.execute("INSERT **INTO** Student (Name) VALUES('Ram')")

**con.commit()**

except mdb.Error, e:

if con:

**con.rollback()**

print "Error %d: %s" % (e.args[0],e.args[1])

sys.exit(1)

finally:

if con:

**con.close()**