International Journal of Science and Research (IJSR)
ISSN: 2319-7064
Impact Factor (2018): 7.426

Python – Using Database and SQL

Shweta J. Patil

Assistant Professor, Department of Computer Science and Engineering, Jawaharlal Nehru Engineering College, Aurangabad, M.S, India

Abstract: ***Python is a general-purpose, high-level programming language whose design philosophy emphasizes code readability. Python claims to combine "remarkable power with very clear syntax", and its standard library is large and comprehensive. Python is a programming language that lets you work more quickly and integrate your systems more effectively. In this paper we reviews available resources and basic information about database modules that are known to be used with Python and also how to make the connection between python and database. This paper features about different database systems with their standard commands implemented with python also result best suitable to implement database engine using python.***

Keywords: Python, Database, SQL

1. Introduction

The database is a collection of organized information that can be easily be used,managed,update and they are classified according to their organizational approach. Most database are organized like a dictionary in the sense that they map from keys to values. Like a dictionary, database software is designed to keep the inserting and accessing of data very fast,even for large amounts of data.Database software maintains its performance by building indexes as data is added to the database to allow the computer to jump quickly to a particular entry.

There are many different database systems which are used for a wide variety of purposes including:Oracle, MySQL, Microsoft SQL server,PostgreSQL and SQLite.

In python file, you have to first establish a connection between your file and the database. After that, you can add, search, delete or update your database. Moreover, you can retrieve the data from the database, make any operation on it then re-add it to the database. The database operations are performed using SQL statements[1]. In this paper, we first reviews available resources and basic information about database modules that are known to be used with Python. Each review is accompanied by a demonstrative Python code that can help users to start using the modules in Python. In the second section, a description of how to make the connection between python and database is provided. In the third section, a quick review of the basic SQL statements is presented. In the forth section, the main database operations are performed using python.

Python has support for working with databases via a simple API. Modules included with Python include modules for SQLite and Berkeley DB. Modules for MySQL , PostgreSQL , FirebirdSQL and others are available as third-party modules. The latter have to be downloaded and installed before use. The package MySQLdb can be installed, for example, using the debian package "pythonmysqldb".Some supported databases:

- GadFly
- MySQL
- PostgreSQL
- SQLite
- Oracle

Five steps must be taken to make Python work in a database system:

1) Import the database module (MySQLdb, phpmyAdmin, sqlite, etc)
2) Use module.connect() to create a connection.
3) Use connection.cursor() to get a cursor. Cursors do all the work.
4) Use cursor.execute(sql_query) to run something.
5) Use cursor.fetchall() to get results.

1.1 Gadfly

First database module we review is Gadfly. We adopted most relevant information about Gadfly from resource [1] as follows. Gadfly is a collection of python modules that provides relational database functionality entirely implemented in Python. It supports a subset of the intergalactic standard RDBMS Structured Query Language SQL. One of the most compelling aspects of Gadfly is that it runs where ever Python runs and supports client/server on any platform that supports the standard Python socket interface. Even the file formats used by Gadfly for storage are cross-platform -- a gadfly database directory can be moved from Win95 to Linux using a binary copying mechanism and gadfly will read and run the database. It supports persistent databases consisting of a collection of structured tables with indices, and a large subset of SQL for accessing and modifying those tables. It supports a log based recovery protocol which allows committed operations of a database to be recovered even if the database was not shut down in a proper manner (ie, in the event of a CPU or software crash, [but not in the event of a disk crash]). It also supports a TCP/IP Client/Server mode where remote clients can access a Gadfly database over a TCP/IP network (such as the Internet) subject to configurable security mechanisms[3].

1.2 MySQL

MySQL is a leading open source database management system. It is a multi user, multithreaded database management system. MySQL is especially popular on the web. It is one of the parts of the very popular LAMP platform. Linux, Apache, MySQL, PHP. Currently MySQL is owned by Oracle. MySQL database is available on most important OS platforms. It runs under BSD Unix, Linux, Windows or Mac. Wikipedia and YouTube use MySQL. These sites manage millions of queries each day. MySQL comes in two versions. MySQL server system and MySQL embedded system[4].

1.3 PostgreSQL

Another reviewed module is PostgreSQL and the most relevant information is reviewed from [5] as follows. PostgreSQL is a powerful, open source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. It runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows. It is fully ACID compliant, has full support for foreign keys, joins, views, triggers, and stored procedures (in multiple languages). It includes most SQL:2008 data types, including INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It also supports storage of binary large objects, including pictures, sounds, or video. PostgreSQL runs stored procedures in more than a dozen programming languages, including Java, Perl, Python, Ruby, Tcl, C/C++, and its own PL/pgSQL, which is similar to Oracle's PL/SQL. Included with its standard function library are hundreds of built-in functions that range from basic math and string operations to cryptography and Oracle compatibility. Triggers and stored procedures can be written in C and loaded into the database as a library, allowing great flexibility in extending its capabilities. Similarly, PostgreSQL includes a framework that allows developers to define and create their own custom data types along with supporting functions and operators that define their behavior[7].

1.4 Oracle

The Oracle Database (commonly referred to as Oracle RDBMS or simply as Oracle) is an object-relational database management system(ORDBMS) produced and marketed by Oracle Corporation. The Oracle RDBMS stores data logically in the form of table spaces and physically in the form of data files ("datafiles"). Tablespaces can contain various types of memory segments, such as Data Segments, Index Segments, etc. Segments in turn comprise one or more extents. Extents comprise groups of contiguous data blocks. Data blocks form the basic units of data storage[5][6].

1.5 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. SQLite is the most widely deployed database in the world with more applications than we can count, including several high-profile projects.

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file.

2. Connecting Python With Thedatabase

A table resides within a database. This is particularly true for MySQL.

To create a table, a database must be created first, or at least a database must be present. So to retrieve data from a 10 table, a connection to the database must be established. This is done by using the connect() method. In other words, connect is the constructor of the phpMyAdmin. The parameters are as follows:

- host is the name of the system where the MySQL server is running. It can be a name or an IP address. If no value is passed, then the default value used is localhost.
- user is the user id, which must be authenticated. In other words, this is the authentic id for using the services of the Server. The default value is the current effective user. Most of the time it is either ‘nobody’ or ‘root’.
- passwd -- It is by using a combination of the user id and a password that MySQL server (or for that matter any server) authenticates a user. The default value is no passwords. That means a null string for this parameter.
- db is the database that must be used once the connection has been established with the server. However, if the database to be used is not selected, the connection established is of no use. There is no default value for this parameter.

2.1 Creation of the Cursor

In the terminology of databases, cursor is that area in the memory where the data fetched from the data tables are kept once the query is executed. In essence it is the scratch area for the database. MySQL does not support cursors. But it is easy to emulate the functionality of cursors. That’s what the phpMyAdmin does. To get a cursor, the cursor() method of connection object has to be used. There is only one parameter to this method -- the class that implements cursor behavior. This parameter is optional. If no value is given, then it defaults to the standard Cursor class. If more control is required, then custom Cursor class can be provided. To obtain a cursor object the statement would be:
cursor= db.cursor()

2.3 Execution of the SQL statement

The steps enumerated until now have done the job of connecting the application with the database and providing an object that simulates the functionality of cursors. The stage has been set for execution of SQL statements. Any SQL statement supported by MySQL can be executed using the execute() method of the Cursor class. The SQL statement is passed as a string to it. Once the statement is executed successfully, the Cursor object will contain the result set of the retrieved values. For example, to retrieve all the rows of a table named USER_MASTER the statement would be: cursor.execute(“select * from USER_MASTER”) Once the above statement is executed, the cursor object would contain all the retrieved. This brings us to the fourth step, fetching of the resultset. Before moving on to the next step, there is one point you must understand. The execute() function accepts and executes any valid SQL statement, including DDL statements such as delete table, alter table, and so on. In the case of DDL statements, there is no fifth step (i.e. iteration over the results fetched).

2.4 Modification in table

Connection.commit() is a necessary statement after any modification to the table by adding, updating or deleting. However, if you only want to view the data at the table by select statement, you don’t need to commit.

2.5 Fetching the result set

The flexibility of Python comes to the fore in this step also. In the real world, fetching all the rows at once may not be feasible. MySQLdb answers this situation by providing different versions of the fetch() function of Cursor class. The two most commonly used versions are:

- fetchone(): This fetches one row in the form of a Python tuple. All the data types are mapped to the Python data types except one -- unsigned integer. To avoid any overflow problem, it is mapped to the long.
- fetchall(): This fetches all the rows as tuple of tuples. While fetchone() increments the cursor position by one, fetchall() does nothing of that kind. Everything else is similar.

That covers all the steps required to access MySQL. What we have discussed up to now is "the approach of tackling a problem of the type database connectivity." However, in real life, reusability plays a more important role than it has in what you have seen so far.

3. Conclusions

During the work on project, tried to analyze all the database servers in order to find the most suitable one. After a careful consideration MySQL Server is chosen since it has many 14 appropriate characteristics to be implemented in Python. Python is one of the most known advanced programming languages, which owns mainly to its own natural expressiveness as well as to the bunch of support modules that helps extend its advantages, that’s why Python fits perfectly well when it comes to developing a stable connection between the program and the database.

References

[1] Ahmed Othman Eltahawey. Database Using Python: A Tutorial, ResearchGate publication, December, 2016
[2] Ainur Duisenbayeva Zhuldyz Assylova. Study of Python programming language for database systems, May, 2012
[3] http://gadfly.sourceforge.net/gadfly.html
[4] http://zetcode.com/databases/mysqlpythontutorial/
[5] http://www.itsabacus.com/oracle.html
[6] http://www.orafaq.com/wiki/Python
[7] http://www.postgresql.org/
[8] http://informix-zone.com/informix-script-dbapi