第二十一章 数据库编程

21.1 介绍

Python 可以通过DB-API直接访问RDBMS,或者通过ORM来访问。
DB-API相当于于一个Director/Dispatcher/Handler,为底层各种各样的数据库系统和数据库接口提供了一致的访问接口。这样使得Python程序在不同的数据库之间移植成为一件轻松的事(一般来说,只修改几行代码)。

> 一个处理 Python 数据库事务的特殊事物小组（ SIG，special interest group ）因此诞生,
> 最后. DB-API 1.0 问世. DB-API 为不同的数据库提供了一致的访问接口。
> 目前PEP 249为2.0版本 https://www.python.org/dev/peps/pep-0249/

21.2 Python数据库应用

DB-API规范以下属性必须提供
apilevel,threadsafety,paramstyle,connect(),StandardError(具体参考文档)
常见数据库的connect()函数

- `MySQLdb.connect(host='dbserv',db='inv',user='smith')`
- `PgSQL.connect(database='sales')`
- `sqlite3.connect('marketing/test')`

21.2.2 连接对象–Connection Ojbect

需要支持一下方法

- `.close()`
- `.commit()`
- `.rollback()`
- `.cursor()`

21.2.3 游标对象 Cursor Object

Cursor Object要有的属性和方法

属性/方法	描述
`description`	describing one result column,返回一个七元组

(name,type_code,display_size,internal_size,presion,scale,null_ok)|
`execute(op[,args])`	执行一个数据库查询或命令
`fetchmany(size=cursor.arraysize)`	取结果集的下几行
`fetchall()`	取结果集中剩下的所有行
`nextset`	移到下一个结果集
`rownumber`	当前结果集中游标的索引

> 最重要的是execute*()和fetch*() 方法

21.2.4 类型对象和构造器

类型对象用于在Python对象与数据库对象之间实现转换。

> SQL的NULL值被映射为Python的NULL对象，也就是None。

类型对象和构造器

类型对象	描述
`Date(yr,mo,dy)`	日期值对象
`Time(hr,min,sec)`	时间值对象
`Timestamp(yr,mo,dy,hr,min,sec)`	时间戳对象
`DateFromTicks(ticks)`	自 1970-01-01 00:00:01 utc 以来的 ticks 秒数得到日期
`TimeFromTicks(ticks)`	自 1970-01-01 00:00:01 utc 以来的 ticks 秒数得到时间值对象
`TimestapFromTicks(ticks)`	通过自 1970-01-01 00:00:01 utc 以来的 ticks 秒数得到时间戳对象
`Binary(string)`	二进制长字符串值得对象
`STRING`	描述字符串的对象,如VARCHAR
`NUMBER`	描述数字列的对象
`ROWID`	描述row ID列的对象

21.2.6 数据库和Python

你不喜欢自己写 SQL, 也不想参与数据库管理的细节–那么本章后面讲到的 ORM 应该可以满足要求。

21.2.7 举例

分别使用MySQL和SQLite作为数据库

MySQL

http://www.linuxfly.org/windows_install_mysql_python_library/
按照文中的步骤来是可行的,不是MySQL驱动的问题。
MySQL的接口程序MySQLdb

```
import MySQLdb

def createdb(dbname):
    connection = MySQLdb.connect(user='root')
    connection.query('create database if not exists %s' % dbname)
    connection.query("grant all on test.* to ''@'localhost'")
    connection.commit()
    connection.close()

def createtable(dbname):
    conn = MySQLdb.connect(db=dbname)
    cur = conn.cursor()
    strdrop = 'drop TABLE if EXISTS pyusers'
    cur._query(strdrop)
    conn.commit()

    strsql = 'create table pyusers(login varchar(20),uid int);'
    cur.execute(strsql)
    conn.commit()

    data = "insert into pyusers values('join',7000),('jane',7001),('bob',7002)"
    cur.execute(data)
    conn.commit()

    cur.close()
    conn.close()

def getdata():
    # 取数据
    conn = MySQLdb.connect(host='localhost',user='root',db='test')
    cur = conn.cursor()
```

```
    sql = 'select * from pyusers'
    cur.execute(sql)
    #conn.commit()

    print 'login\t\tuid'
    for d in cur.fetchall():
        print '%-8s\t%-8s' % (d[0],d[1])

    cur.close()
    conn.close()

def updatedata():
    conn = MySQLdb.connect(host='127.0.0.1',user='root',db='test')
    cur = conn.cursor()

    usql = 'UPDATE pyusers SET uid=7100 WHERE uid=7001'
    cur.execute(usql)

    qsql = 'select * from pyusers'
    cur.execute(qsql)

    print 'updated '
    print 'login\t\tuid'
    for d in cur.fetchall():
        print '%-8s\t%-8s' % (d[0],d[1])

    cur.close()
    conn.close()

if __name__=="__main__":
    #createdb('test')
    #createtable(dbname='test')
```

```
getdata()
updatedata()
```

sqlite3是SQLite数据库的DB-API 2.0接口

sqlite3的例子

```
import sqlite3

class SQLite3Helper(object):

    def __init__(self, dbname):
        self.dbname = dbname
        conn = sqlite3.connect(dbname)
        print 'sqlite3 database creation successfully.'
        conn.close()
    def querydata(self):
        conn = sqlite3.connect(self.dbname)
        cur = conn.cursor()
        cur.execute('DROP TABLE IF EXISTS sqlliteusers')
        cur.execute('CREATE TABLE sqlliteusers(login varchar(8), uid INTEGER)')
        cur.execute('INSERT INTO sqlliteusers VALUES ("join",100)')
        cur.execute('INSERT INTO sqlliteusers VALUES ("jane",110)')
        conn.commit()
        cur.execute('SELECT * FROM sqlliteusers')
        for eachUser in cur.fetchall():
            print eachUser
        cur.close()
        conn.close()


if __name__=="__main__":
    shelper = SQLite3Helper('example.db')
    shelper.querydata()
```

21.3 Object-Rational Managers(ORMs)

数据库的表被转换为 Python 类，它具有列属性和操作数据库的方法。其实就是Data Models.

知名的Python ORM模块是SQLAlchemy `['ælkəmi]` 和SQLObject.

> 其它的 Python ORM 包括 PyDO/PyDO2, PDO, Dejavu, Durus, QLime 和 ForgetSQL.

相对于直接使用SQL来讲，使用ORM时用类代替了函数。

导入相关的模块和常量

> 首先导入 Python 标准库模块, 然后再导入第三方或扩展模块, 最后导入本地模块这种风格.

21.4 相关模块

数据库和ORMs的相关库
用到的数据库搜索下就出来了，不列了
ORMs如下

```
ORMs
SQLObject       http://sqlobject.org
SQLAlchemy      http://sqlalchemy.org
PyDO/PyDO2      http://skunkweb.sf.net/pydo.html
```