**[python开发\_sqlite3\_绝对完整](http://www.cnblogs.com/hongten/p/hongten_python_sqlite3.html)**

2016年8月7日

11:26

'''SQLite数据库是一款非常小巧的嵌入式开源数据库软件，也就是说  
没有独立的维护进程，所有的维护都来自于程序本身。  
在python中，使用sqlite3创建数据库的连接，当我们指定的数据库文件不存在的时候  
连接对象会自动创建数据库文件；如果数据库文件已经存在，则连接对象不会再创建  
数据库文件，而是直接打开该数据库文件。  
 连接对象可以是硬盘上面的数据库文件，也可以是建立在内存中的，在内存中的数据库  
 执行完任何操作后，都不需要提交事务的(commit)

创建在硬盘上面： conn = sqlite3.connect('c:\\test\\test.db')  
 创建在内存上面： conn = sqlite3.connect('"memory:')

下面我们一硬盘上面创建数据库文件为例来具体说明：  
 conn = sqlite3.connect('c:\\test\\hongten.db')  
 其中conn对象是数据库链接对象，而对于数据库链接对象来说，具有以下操作：

commit() --事务提交  
 rollback() --事务回滚  
 close() --关闭一个数据库链接  
 cursor() --创建一个游标

cu = conn.cursor()  
 这样我们就创建了一个游标对象：cu  
 在sqlite3中，所有sql语句的执行都要在游标对象的参与下完成  
 对于游标对象cu，具有以下具体操作：

execute() --执行一条sql语句  
 executemany() --执行多条sql语句  
 close() --游标关闭  
 fetchone() --从结果中取出一条记录  
 fetchmany() --从结果中取出多条记录  
 fetchall() --从结果中取出所有记录  
 scroll() --游标滚动

'''

[复制代码](javascript:void(0);)

下面是我做的demo，在demo中，我做了很详细的注释和功能的演示，详情如下：

运行效果：

当**SHOW\_SQL = False**的时候：

[复制代码](javascript:void(0);)

Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> ================================ RESTART ================================  
>>>   
show\_sql : False  
删除数据库表测试...  
硬盘上面:[c:\test\hongten.db]  
删除数据库表[student]成功!  
创建数据库表测试...  
硬盘上面:[c:\test\hongten.db]  
创建数据库表[student]成功!  
保存数据测试...  
硬盘上面:[c:\test\hongten.db]  
查询所有数据...  
硬盘上面:[c:\test\hongten.db]  
(1, 'Hongten', '男', 20, '广东省广州市', '13423\*\*\*\*62')  
(2, 'Tom', '男', 22, '美国旧金山', '15423\*\*\*\*63')  
(3, 'Jake', '女', 18, '广东省广州市', '18823\*\*\*\*87')  
(4, 'Cate', '女', 21, '广东省广州市', '14323\*\*\*\*32')  
##################################################  
查询一条数据...  
硬盘上面:[c:\test\hongten.db]  
(1, 'Hongten', '男', 20, '广东省广州市', '13423\*\*\*\*62')  
##################################################  
更新数据...  
硬盘上面:[c:\test\hongten.db]  
查询所有数据...  
硬盘上面:[c:\test\hongten.db]  
(1, 'HongtenAA', '男', 20, '广东省广州市', '13423\*\*\*\*62')  
(2, 'HongtenBB', '男', 22, '美国旧金山', '15423\*\*\*\*63')  
(3, 'HongtenCC', '女', 18, '广东省广州市', '18823\*\*\*\*87')  
(4, 'HongtenDD', '女', 21, '广东省广州市', '14323\*\*\*\*32')  
##################################################  
删除数据...  
硬盘上面:[c:\test\hongten.db]  
查询所有数据...  
硬盘上面:[c:\test\hongten.db]  
(2, 'HongtenBB', '男', 22, '美国旧金山', '15423\*\*\*\*63')  
(4, 'HongtenDD', '女', 21, '广东省广州市', '14323\*\*\*\*32')  
>>>

[复制代码](javascript:void(0);)

当**SHOW\_SQL = True**的时候：

[复制代码](javascript:void(0);)

Python 3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)] on win32  
Type "copyright", "credits" or "license()" for more information.  
>>> ================================ RESTART ================================  
>>>   
show\_sql : True  
删除数据库表测试...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[DROP TABLE IF EXISTS student]  
删除数据库表[student]成功!  
创建数据库表测试...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[CREATE TABLE `student` (  
 `id` int(11) NOT NULL,  
 `name` varchar(20) NOT NULL,  
 `gender` varchar(4) DEFAULT NULL,  
 `age` int(11) DEFAULT NULL,  
 `address` varchar(200) DEFAULT NULL,  
 `phone` varchar(20) DEFAULT NULL,  
 PRIMARY KEY (`id`)  
 )]  
创建数据库表[student]成功!  
保存数据测试...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[INSERT INTO student values (?, ?, ?, ?, ?, ?)],参数:[(1, 'Hongten', '男', 20, '广东省广州市', '13423\*\*\*\*62')]  
执行sql:[INSERT INTO student values (?, ?, ?, ?, ?, ?)],参数:[(2, 'Tom', '男', 22, '美国旧金山', '15423\*\*\*\*63')]  
执行sql:[INSERT INTO student values (?, ?, ?, ?, ?, ?)],参数:[(3, 'Jake', '女', 18, '广东省广州市', '18823\*\*\*\*87')]  
执行sql:[INSERT INTO student values (?, ?, ?, ?, ?, ?)],参数:[(4, 'Cate', '女', 21, '广东省广州市', '14323\*\*\*\*32')]  
查询所有数据...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[SELECT \* FROM student]  
(1, 'Hongten', '男', 20, '广东省广州市', '13423\*\*\*\*62')  
(2, 'Tom', '男', 22, '美国旧金山', '15423\*\*\*\*63')  
(3, 'Jake', '女', 18, '广东省广州市', '18823\*\*\*\*87')  
(4, 'Cate', '女', 21, '广东省广州市', '14323\*\*\*\*32')  
##################################################  
查询一条数据...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[SELECT \* FROM student WHERE ID = ? ],参数:[1]  
(1, 'Hongten', '男', 20, '广东省广州市', '13423\*\*\*\*62')  
##################################################  
更新数据...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[UPDATE student SET name = ? WHERE ID = ? ],参数:[('HongtenAA', 1)]  
执行sql:[UPDATE student SET name = ? WHERE ID = ? ],参数:[('HongtenBB', 2)]  
执行sql:[UPDATE student SET name = ? WHERE ID = ? ],参数:[('HongtenCC', 3)]  
执行sql:[UPDATE student SET name = ? WHERE ID = ? ],参数:[('HongtenDD', 4)]  
查询所有数据...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[SELECT \* FROM student]  
(1, 'HongtenAA', '男', 20, '广东省广州市', '13423\*\*\*\*62')  
(2, 'HongtenBB', '男', 22, '美国旧金山', '15423\*\*\*\*63')  
(3, 'HongtenCC', '女', 18, '广东省广州市', '18823\*\*\*\*87')  
(4, 'HongtenDD', '女', 21, '广东省广州市', '14323\*\*\*\*32')  
##################################################  
删除数据...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[DELETE FROM student WHERE NAME = ? AND ID = ? ],参数:[('HongtenAA', 1)]  
执行sql:[DELETE FROM student WHERE NAME = ? AND ID = ? ],参数:[('HongtenCC', 3)]  
查询所有数据...  
硬盘上面:[c:\test\hongten.db]  
执行sql:[SELECT \* FROM student]  
(2, 'HongtenBB', '男', 22, '美国旧金山', '15423\*\*\*\*63')  
(4, 'HongtenDD', '女', 21, '广东省广州市', '14323\*\*\*\*32')  
>>>

#python sqlite

#Author : Hongten

#MailTo : hongtenzone@foxmail.com

#QQ : 648719819

#Blog : <http://www.cnblogs.com/hongten>

#Create : 2013-08-09

#Version: 1.0

#DB-API 2.0 interface for SQLite databases

import sqlite3

import os

'''SQLite数据库是一款非常小巧的嵌入式开源数据库软件，也就是说

没有独立的维护进程，所有的维护都来自于程序本身。

在python中，使用sqlite3创建数据库的连接，当我们指定的数据库文件不存在的时候

连接对象会自动创建数据库文件；如果数据库文件已经存在，则连接对象不会再创建

数据库文件，而是直接打开该数据库文件。

连接对象可以是硬盘上面的数据库文件，也可以是建立在内存中的，在内存中的数据库

执行完任何操作后，都不需要提交事务的(commit)

创建在硬盘上面： conn = sqlite3.connect('c:\\test\\test.db')

创建在内存上面： conn = sqlite3.connect('"memory:')

下面我们一硬盘上面创建数据库文件为例来具体说明：

conn = sqlite3.connect('c:\\test\\hongten.db')

其中conn对象是数据库链接对象，而对于数据库链接对象来说，具有以下操作：

commit() --事务提交

rollback() --事务回滚

close() --关闭一个数据库链接

cursor() --创建一个游标

cu = conn.cursor()

这样我们就创建了一个游标对象：cu

在sqlite3中，所有sql语句的执行都要在游标对象的参与下完成

对于游标对象cu，具有以下具体操作：

execute() --执行一条sql语句

executemany() --执行多条sql语句

close() --游标关闭

fetchone() --从结果中取出一条记录

fetchmany() --从结果中取出多条记录

fetchall() --从结果中取出所有记录

scroll() --游标滚动

'''

#global var

#数据库文件绝句路径

DB\_FILE\_PATH = ''

#表名称

TABLE\_NAME = ''

#是否打印sql

SHOW\_SQL = True

def get\_conn(path):

'''获取到数据库的连接对象，参数为数据库文件的绝对路径

如果传递的参数是存在，并且是文件，那么就返回硬盘上面改

路径下的数据库文件的连接对象；否则，返回内存中的数据接

连接对象'''

conn = sqlite3.connect(path)

if os.path.exists(path) and os.path.isfile(path):

print('硬盘上面:[{}]'.format(path))

return conn

else:

conn = None

print('内存上面:[:memory:]')

return sqlite3.connect(':memory:')

def get\_cursor(conn):

'''该方法是获取数据库的游标对象，参数为数据库的连接对象

如果数据库的连接对象不为None，则返回数据库连接对象所创

建的游标对象；否则返回一个游标对象，该对象是内存中数据

库连接对象所创建的游标对象'''

if conn is not None:

return conn.cursor()

else:

return get\_conn('').cursor()

###############################################################

#### 创建|删除表操作 START

###############################################################

def drop\_table(conn, table):

'''如果表存在,则删除表，如果表中存在数据的时候，使用该

方法的时候要慎用！'''

if table is not None and table != '':

sql = 'DROP TABLE IF EXISTS ' + table

if SHOW\_SQL:

print('执行sql:[{}]'.format(sql))

cu = get\_cursor(conn)

cu.execute(sql)

conn.commit()

print('删除数据库表[{}]成功!'.format(table))

close\_all(conn, cu)

else:

print('the [{}] is empty or equal None!'.format(sql))

def create\_table(conn, sql):

'''创建数据库表：student'''

if sql is not None and sql != '':

cu = get\_cursor(conn)

if SHOW\_SQL:

print('执行sql:[{}]'.format(sql))

cu.execute(sql)

conn.commit()

print('创建数据库表[student]成功!')

close\_all(conn, cu)

else:

print('the [{}] is empty or equal None!'.format(sql))

###############################################################

#### 创建|删除表操作 END

###############################################################

def close\_all(conn, cu):

'''关闭数据库游标对象和数据库连接对象'''

try:

if cu is not None:

cu.close()

finally:

if cu is not None:

cu.close()

###############################################################

#### 数据库操作CRUD START

###############################################################

def save(conn, sql, data):

'''插入数据'''

if sql is not None and sql != '':

if data is not None:

cu = get\_cursor(conn)

for d in data:

if SHOW\_SQL:

print('执行sql:[{}],参数:[{}]'.format(sql, d))

cu.execute(sql, d)

conn.commit()

close\_all(conn, cu)

else:

print('the [{}] is empty or equal None!'.format(sql))

def fetchall(conn, sql):

'''查询所有数据'''

if sql is not None and sql != '':

cu = get\_cursor(conn)

if SHOW\_SQL:

print('执行sql:[{}]'.format(sql))

cu.execute(sql)

r = cu.fetchall()

if len(r) > 0:

for e in range(len(r)):

print(r[e])

else:

print('the [{}] is empty or equal None!'.format(sql))

def fetchone(conn, sql, data):

'''查询一条数据'''

if sql is not None and sql != '':

if data is not None:

#Do this instead

d = (data,)

cu = get\_cursor(conn)

if SHOW\_SQL:

print('执行sql:[{}],参数:[{}]'.format(sql, data))

cu.execute(sql, d)

r = cu.fetchall()

if len(r) > 0:

for e in range(len(r)):

print(r[e])

else:

print('the [{}] equal None!'.format(data))

else:

print('the [{}] is empty or equal None!'.format(sql))

def update(conn, sql, data):

'''更新数据'''

if sql is not None and sql != '':

if data is not None:

cu = get\_cursor(conn)

for d in data:

if SHOW\_SQL:

print('执行sql:[{}],参数:[{}]'.format(sql, d))

cu.execute(sql, d)

conn.commit()

close\_all(conn, cu)

else:

print('the [{}] is empty or equal None!'.format(sql))

def delete(conn, sql, data):

'''删除数据'''

if sql is not None and sql != '':

if data is not None:

cu = get\_cursor(conn)

for d in data:

if SHOW\_SQL:

print('执行sql:[{}],参数:[{}]'.format(sql, d))

cu.execute(sql, d)

conn.commit()

close\_all(conn, cu)

else:

print('the [{}] is empty or equal None!'.format(sql))

###############################################################

#### 数据库操作CRUD END

###############################################################

###############################################################

#### 测试操作 START

###############################################################

def drop\_table\_test():

'''删除数据库表测试'''

print('删除数据库表测试...')

conn = get\_conn(DB\_FILE\_PATH)

drop\_table(conn, TABLE\_NAME)

def create\_table\_test():

'''创建数据库表测试'''

print('创建数据库表测试...')

create\_table\_sql = '''CREATE TABLE `student` (

`id` int(11) NOT NULL,

`name` varchar(20) NOT NULL,

`gender` varchar(4) DEFAULT NULL,

`age` int(11) DEFAULT NULL,

`address` varchar(200) DEFAULT NULL,

`phone` varchar(20) DEFAULT NULL,

PRIMARY KEY (`id`)

)'''

conn = get\_conn(DB\_FILE\_PATH)

create\_table(conn, create\_table\_sql)

def save\_test():

'''保存数据测试...'''

print('保存数据测试...')

save\_sql = '''INSERT INTO student values (?, ?, ?, ?, ?, ?)'''

data = [(1, 'Hongten', '男', 20, '广东省广州市', '13423\*\*\*\*62'),

(2, 'Tom', '男', 22, '美国旧金山', '15423\*\*\*\*63'),

(3, 'Jake', '女', 18, '广东省广州市', '18823\*\*\*\*87'),

(4, 'Cate', '女', 21, '广东省广州市', '14323\*\*\*\*32')]

conn = get\_conn(DB\_FILE\_PATH)

save(conn, save\_sql, data)

def fetchall\_test():

'''查询所有数据...'''

print('查询所有数据...')

fetchall\_sql = '''SELECT \* FROM student'''

conn = get\_conn(DB\_FILE\_PATH)

fetchall(conn, fetchall\_sql)

def fetchone\_test():

'''查询一条数据...'''

print('查询一条数据...')

fetchone\_sql = 'SELECT \* FROM student WHERE ID = ? '

data = 1

conn = get\_conn(DB\_FILE\_PATH)

fetchone(conn, fetchone\_sql, data)

def update\_test():

'''更新数据...'''

print('更新数据...')

update\_sql = 'UPDATE student SET name = ? WHERE ID = ? '

data = [('HongtenAA', 1),

('HongtenBB', 2),

('HongtenCC', 3),

('HongtenDD', 4)]

conn = get\_conn(DB\_FILE\_PATH)

update(conn, update\_sql, data)

def delete\_test():

'''删除数据...'''

print('删除数据...')

delete\_sql = 'DELETE FROM student WHERE NAME = ? AND ID = ? '

data = [('HongtenAA', 1),

('HongtenCC', 3)]

conn = get\_conn(DB\_FILE\_PATH)

delete(conn, delete\_sql, data)

###############################################################

#### 测试操作 END

###############################################################

def init():

'''初始化方法'''

#数据库文件绝句路径

global DB\_FILE\_PATH

DB\_FILE\_PATH = 'c:\\test\\hongten.db'

#数据库表名称

global TABLE\_NAME

TABLE\_NAME = 'student'

#是否打印sql

global SHOW\_SQL

SHOW\_SQL = True

print('show\_sql : {}'.format(SHOW\_SQL))

#如果存在数据库表，则删除表

drop\_table\_test()

#创建数据库表student

create\_table\_test()

#向数据库表中插入数据

save\_test()

def main():

init()

fetchall\_test()

print('#' \* 50)

fetchone\_test()

print('#' \* 50)

update\_test()

fetchall\_test()

print('#' \* 50)

delete\_test()

fetchall\_test()

if \_\_name\_\_ == '\_\_main\_\_':

main()