我终于学会了使用python操作postgresql

一 前言

这篇文章不仅适合pgsql,更适合mysql，思路都是一致的，如果读者学会使用psycopg2操作pgsql，那么使用PyMySQL 操作mysql也是很简单；本篇文章涵盖内容广泛，提供的操作选择性很多，比如多种数据插入操作，防止sql注入方式，异常处理，sql语句打印处理，显示行号等操作，一篇文章真的收益匪浅；

二 数据库连接

2.1 安装 psycopg2

```
#  pip install psycopg2
```

2.2 连接数据库

每条完整的sql执行步骤如下，读者应谨记；

1. 建立连接获得 connect 对象
2. 获得游标对象，一个游标对象可以对数据库进行执行操作，非线程安全，多个应用会在同一个连接种创建多个光标；
3. 书写sql语句
4. 调用execute()方法执行sql
5. 抓取数据（可选操作）
6. 提交事物
7. 关闭连接

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象
cursor = conn.cursor()
# sql语句
sql = "SELECT VERSION()"
# 执行语句
cursor.execute(sql)
# 获取单条数据.
data = cursor.fetchone()
# 打印
print("database version : %s " % data)
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

输出结果打印出数据库版本说明连接数据库成功：

```
database version : PostgreSQL 11.3, compiled by Visual C++ build 1914, 64-bit
```

三 创建表

创建学生表主要有字段id 唯一标识，字段 num 代表学号，字段 name 代表学生姓名；详细的建表默认规则转换见附录

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql = """CREATE TABLE student (
id serial4 PRIMARY KEY,
num int4,
name varchar(25));"""
# 执行语句
cursor.execute(sql)
print("student table created successfully")
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

四 插入操作

4.1 插入数据姿势一

知识追寻者提供的第一种防止sql注入的插入数据方式（具有占位符的预编译sql），重要程度不言而喻；美中不足是字符串类型必须带上单引号；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="INSERT INTO student (num, name) \
                    VALUES (%s, '%s')" % \
                    (100, 'zszxz')
# 执行语句
cursor.execute(sql)
print("successfully")
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

4.2 插入数据姿势二（参数分离）

知识追寻者认为下面参数与sql语句分离插入的姿势更简便帅气，也是防止sql注入问题；强烈推荐；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""INSERT INTO student (num, name) VALUES (%s, %s)"""
params = (101, 'zszxz')
# 执行语句
cursor.execute(sql,params)
print("successfully")
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

4.3 插入数据姿势三（字典）

第三种姿势也就是是支持字典映射关系插入，使用字典方式的插入数据是根据字典的key进行匹配占位符，强烈推荐；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""INSERT INTO student (num, name) VALUES (%(num)s, %(name)s)"""
params = {'num':102, 'name':'zszxz'}
# 执行语句
cursor.execute(sql,params)
print("successfully")
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

五 查询操作

5.1 查询一条数据

使用fetchone()方法可以抓取一条数据，返回的是元组；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""SELECT * FROM student;"""
# 执行语句
cursor.execute(sql)
# 抓取
row = cursor.fetchone()
print(row)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

输出结果：

```
(1, 100, 'zszxz')
```

5.2 查询多条数据

1. 使用fetchmany([size=cursor.arraysize])方法可以抓取多条数据；
2. 此方法可以多次使用，直到数据库中没有数据，此时会返回空列表；
3. 如果不传参数，会限制查询条数，一般就是返回第一条；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""SELECT * FROM student;"""
# 执行语句
cursor.execute(sql)
# 抓取
```

```
#row = cursor.fetchone()
rows = cursor.fetchmany(2)
print(rows)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

输出结果：

```
[(1, 100, 'zszxz'), (2, 101, 'zszxz')]
```

5.3 查询全部数据

使用 fetchall() 方法会抓取所有数据；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""SELECT * FROM student;"""
# 执行语句
cursor.execute(sql)
# 抓取
rows = cursor.fetchall()
print(rows)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

输出结果：

```
[(1, 100, 'zszxz'), (2, 101, 'zszxz'), (3, 102, 'zszxz')]
```

5.4 按条件查询

1. 带参查询读者应该谨记sql 与 参数 分离
2. **参数的末尾必须加上逗号**
3. 如果知道返回的数据就一条使用fetchone()方法，如果无特殊要求，否则建议使用fetchall()方法

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""SELECT * FROM student where id = %s;"""
params = (1,)
# 执行语句
cursor.execute(sql,params)
# 抓取
rows = cursor.fetchall()
print(rows)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

输出结果：

```
[(1, 100, 'zszxz')]
```

六 更新操作

更新操作跟之前的查询，插入类似，参数对应的文章分清楚即可。

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""update student set name = %s where id = %s  """
params = ('知识追寻者',3,)
# 执行语句
cursor.execute(sql,params)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

七 删除操作

删除操作很简单，看如下代码，与之前的代码流程没什么区别；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""delete from  student where id = %s  """
params = (3,)
# 执行语句
cursor.execute(sql,params)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

八 异常处理

处理 sql 的异常非常重要，知识追寻者这边使用psycopg2的 `Error` 进行异常捕获，能捕获到sql执行时期的所有异常；下面代码中表test是库中不存的表，执行sql后会报异常，经过异常捕获后非常美观，不影响程序运行；

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""select * from test"""
params = (3,)
try:
    # 执行语句
    cursor.execute(sql,params)
except psycopg2.Error as e:
    print(e)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

执行结果

```
错误:  关系 "test" 不存在
LINE 1: select * from test
```

九 打印sql

使用cursor.query 可以查看执行的sql语句，方便排查;

```
# -*- coding: utf-8 -*-
import psycopg2
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""select * from student"""
try:
    # 执行语句
    cursor.execute(sql,)
    que = cursor.query
    print(que)
except psycopg2.Error as e:
    print(e)
# 事物提交
conn.commit()
# 关闭数据库连接
cursor.close()
conn.close()
```

执行结果:

```
b'select * from student'
```

十 获取总条数

使用cursor.rowcount 可以获得表中所有行总数;

```
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""select * from student"""
# 执行语句
cursor.execute(sql)
count = cursor.rowcount
print(count)
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

输出

```
2
```

十一显示行号

使用cursor.rownumber 可以显示当前查询sql获得数据的行号，每抓取一次光标的索引就会加1;

```
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""select * from student """
# 执行语句
cursor.execute(sql)
row_1 = cursor.fetchone()
print(cursor.rownumber)
row_2 = cursor.fetchone()
```

```
print(cursor.rownumber)
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

输出结果：

```
1
2
```

十二 显示执行参数

使用 `mogrify(operation[, parameters])` 能够显示执行语句的参数绑定结果，返回的是字符串形式；

```
# 获得连接
conn = psycopg2.connect(database="python", user="postgres", password="123456", host="127.0.0.1", port="5432")
# 获得游标对象，一个游标对象可以对数据库进行执行操作
cursor = conn.cursor()
# sql语句 建表
sql ="""INSERT INTO student (num, name) VALUES (%s, %s)"""
params = (102, '知识追寻者')
# 执行语句
result = cursor.mogrify(sql,params)
print(result.decode('UTF-8'))
cursor.execute(sql,params)
# 事物提交
conn.commit()
# 关闭数据库连接
conn.close()
```

执行结果：

```
INSERT INTO student (num, name) VALUES (102, '知识追寻者')
```

十三 附录

支持默认的类型转换如下,如果想要使用强制类型转换，详细的可以参照pgsql官网手册；

Python	PostgreSQL
None	NULL
bool	bool
float	real, double
int, long	smallint, integer, bigint
Decimal	numeric
str, unicode	varchar, text
buffer, memoryview, bytearray, bytes, Buffer protocol	bytea
date	date
time	time,timetz
datetime	timestamp,timestamptz

Python	PostgreSQL
timedelta	interval
list	ARRAY
tuple,namedtuple	Composite typesIN syntax
dict	hstore
Range	range
UUID	uuid
Anything	json
ipaddress	inet