H2 MySQL for Python

> 游标*cursor*是系统为用户开设的一个数据缓冲区，存放SQL语句的执行结果
>
> 每个游标区都有一个名字
>
> 就是对数据库进行具体的操作了，比如增、删、改、查等等一系列操作都可以完成

类型	描述
Cursor	普通的游标对象，默认创建的游标对象
SSCursor	不缓存游标，主要用于当操作需要返回大量数据的时候
DictCursor	以字典的形式返回操作结果
SSDictCursor	不缓存游标，将结果以字典的形式进行返回

```
#游标结果作为元祖的元祖返回
class pymysql.cursors.Cursor(connection)
说明:
这是您用于与数据库交互的对象。
不要自己创建Cursor实例。调用connections.Connection.cursor()

#无缓冲游标结果作为元祖的元祖返回，
class pymysql.cursors.SSCursor(connection)

用途:
用于返回大量数据查询，或慢速网络连接到远程服务器
不将每行数据复制到缓冲区，根据需要获取行。客户端内存使用少
在慢速网络上或结果集非常大时行返回速度快

限制:
MySQL协议不支持返回总行数，判断有多少行唯一方法是迭代返回的每一行。
目前无法向后滚动，因为只有当前行保存在内存中。


#将结果作为字典返回游标
class pymysql.cursors.DictCursor(connection)
#无缓冲游标结果作为字典返回
class pymysql.cursors.SSDictCursor(connection)

```

H3 游标的只读属性

```

name #必需
type_code#必需
display_size
```

```
internal_size
precision
scale
null_ok

#返回游标活动状态
cursor.description
-> (('VERSION()', 253, None, 24, 24, 31, False),)

包含7个元素的元组：
(name, type_code, display_size, internal_size, precision, scale,
null_ok)

例如
self.con = MySQLdb.connect(
    # 交代数据库的属性
    host='localhost',
    user='root',
    passwd='123456',
    db='schooldog',
    port=3306,
    charset='utf8'
)
```

其他属性

```

cursor.max_stmt_length#1024000

cursor.rownumber#5 #当前结果集中游标所在行的索引(起始行号为 0)



cursor.arraysize#1 #此读/写属性指定用.fetchmany()一次获取的行数。

# 默认1表示一次获取一行；也可以用于执行.executemany()



cursor.lastrowid#None #只读属性提供上次修改行的rowid

# DB仅在执行单个INSERT 操作时返回rowid 。

# 如未设rowid或DB不支持rowid应将此属性设置为None

# 如最后执行语句修改了多行，例如用INSERT和.executemany()时
lastrowid语义是未定义



cursor.rowcount #5 #最近一次 execute() 创建或影响的行数

```

```
# 如无cursor.execute()或接口无法确定最后一个操作的rowcount则该属性为-1

# 该行数属性可以在动态更新其值的方式来编码。

# 这对于仅在第一次调用.fetch()方法后返回可用rowcount值的 数据库非常有用。

```

https://www.cnblogs.com/zhangxinqi/p/8386357.html

```
#接对象使用的方法：
conn.close()   #关闭链接
conn.commit()  #提交更改到数据库服务器
conn.cursor(cursor=None)   #创建一个新的游标来执行查询，cursor指定游标类型：Cursor、SSCursor、DictCursor或SSDictCursor，没有指定即使用光标
conn.open   #如果链接处于打开状态则返回true
conn.ping(reconnect=True)  #检查服务器是否存在，reconnect为True时重新链接
conn.rellback()   #回滚当前事务

(2)游标对象API
class pymysql.cursors.Cursor(connection)  ：创建与数据库交换的对象，对象表示数据库游标，用于管理提取操作的上下文

游标的方法：
cursor.callproc(procname)  #查看数据库存储过程
cursor.close() #关闭游标
cursor.execute(query,args=None) #执行查询，query查询参数为字符串，args可以是元组，列表或字典，用于查询的参数，返回类型为INT
cursor.executemany(query,seq_of_parameters) #多次查询返回结果
cursor.fetchall()  #获取所有行
cursor.fetchmany(size=None) #获取指定的行数
cursor.fetchone() #获取下一行
cursor.max_stmt_length=1024000  #executemany()生成的最大语句大小
cursor.mogrify(query,args=None)  #通过调用execute()方法返回发送到数据库的字符串

（3）其它对象API
class pymysql.cursors.SSCursor(connection)  ：#用于返回大量数据的查询
class pymysql.cursors.DictCursor(connection)  ：#用于将结果作为字典返回的游标
class pymysql.cursors.SSDictCursor(connection)  ：#用于无缓冲的游标，它将结果作为字典返回
```

H3 Mysql查询操作

```
import pymysql
```

```
2
3  #创建数据库链接，分别指定主机、用户、密码和数据库名,必须保证用户有
   权限链接
4  db=pymysql.connect('10.0.1.198','test1','123.com','test')
5
6  #创建游标对象
7  cursor = db.cursor()
8
9  #使用execute()方法执行SQL语句
10 cursor.execute('select * from test1')
11
12
13 #获取单条数据
14 print(cursor.fetchone())
15 #获取N条数据
16 print(cursor.fetchmany(3))
17 #获取所有数据，序列形式
18 data = cursor.fetchall()
19 print(data)
20
21 #关闭游标
22 cursor.close()
23 #关闭链接
24 db.close()
```

H3 获取字典类型数据

```
1  import pymysql
2
3  #创建数据库链接，分别指定主机、用户、密码和数据库名,必须保证用户有
   权限链接
4  db=pymysql.connect('10.0.1.198','test1','123.com','test')
5
6  #创建游标对象，指定数据类型为字典，将打印key,value
7  cursor = db.cursor(cursor=pymysql.cursors.DictCursor)
8
9  #使用execute()方法执行SQL语句
10 cursor.execute('select * from test1')
11
12 #获取所有数据，字典形式
13 data = cursor.fetchall()
14 print(data)
15
16 #关闭数据库链接
17 db.close()
```

H3 MySQL更新操作

```
import pymysql

conn =
pymysql.connect(host='10.0.1.198',port=3306,user='test1',passwd='
123.com',db='test')
cursor = conn.cursor()
sql = "update test1 set age=28 where id=4"
cursor.execute(sql)
#提交语句到数据库
conn.commit()

cursor.close()
conn.close()
```

H3 插入多条语句

```
import pymysql

conn =
pymysql.connect(host='10.0.1.198',port=3306,user='test1',passwd='
123.com',db='test')

cursor = conn.cursor()
l1 = [
    ('k1','aa',22,'2222'),
    ('k2','bb',23,'3333'),
    ('k3','cc',24,'4444'),
    ('k4','dd',25,'5555')
]
#定义数据库语句
sql = "insert into test1(name,sex,age,tel) values(%s,%s,%s,%s)"

#executemany()插入多条数据
cursor.executemany(sql,l1)

#获取新增数据自增ID
print(cursor.lastrowid)

#提交语句到数据库
conn.commit()

cursor.close()
conn.close()
```

H3 创建数据库表

```

import pymysql
```

```
3
4  conn = pymysql.connect('10.0.1.198','test1','123.com')
5  #创建游标对象,字典输出
6  cursor = conn.cursor(cursor=pymysql.cursors.DictCursor)
7  cursor.execute('drop database if exists student')
8  cursor.execute("create database student character set 'utf8'
   collate utf8_general_ci")
9
10 #如果表存在则删除
11 cursor.execute('drop table if exists student.test2')
12 #sql建表语句
13 sql = '''create table student.test2(
14     id int not null auto_increment primary key,
15     name char(8) not null,
16     age int not null)'''
17 #执行建表语句
18 cursor.execute(sql)
19
20 cursor.close()
21 conn.close()
```

H3 数据回滚操作

```
1
2  import pymysql
3  #创建数据库链接，分别指定主机、用户、密码和数据库名,必须保证用户有
   权限链接
4  db=pymysql.connect('10.0.1.198','test1','123.com','test')
5
6  #创建游标对象
7  cursor = db.cursor(cursor=pymysql.cursors.DictCursor)
8
9  #使用execute()方法执行SQL语句
10 sql2 = "insert into test1(name,sex,age,tel)
   values('zz','ee',21,'8999')"
11 try:
12     cursor.execute(sql2)
13     #提交到数据库执行
14     db.commit()
15 except:
16     #发生错误时回滚操作
17     db.rollback()
18
19 #获取所有数据，序列形式  fetch取   fetchall 取所有
20 data = cursor.fetchall()
21 print(data)
22
23 #关闭数据库链接
24 cursor.close()
25 db.close()
```

H3 错误处理

DB API中定义了一些数据库操作的错误及异常，下表列出了这些错误和异常:

异常	描述
Warning	当有严重警告时触发，例如插入数据是被截断等等。必须是 StandardError 的子类。
Error	警告以外所有其他错误类。必须是 StandardError 的子类。
InterfaceError	当有数据库接口模块本身的错误（而不是数据库的错误）发生时触发。 必须是Error的子类。
DatabaseError	和数据库有关的错误发生时触发。 必须是Error的子类。
DataError	当有数据处理时的错误发生时触发，例如：除零错误，数据超范围等等。 必须是DatabaseError的子类。
OperationalError	指非用户控制的，而是操作数据库时发生的错误。例如：连接意外断开、 数据库名未找到、事务处理失败、内存分配错误等等操作数据库是发生的错误。 必须是DatabaseError的子类。
IntegrityError	完整性相关的错误，例如外键检查失败等。必须是DatabaseError子类。
InternalError	数据库的内部错误，例如游标（cursor）失效了、事务同步失败等等。 必须是DatabaseError子类。
Programming Error	程序错误，例如数据表（table）没找到或已存在、SQL语句语法错误、 参数数量错误等等。必须是DatabaseError的子类。
NotSupported Error	不支持错误，指使用了数据库不支持的函数或API等。例如在连接对象上 使用.rollback()函数，然而数据库并不支持事务或者事务已关闭。 必须是DatabaseError的子类。