**17Python\_09PymySQL模块+SQLAlchemy基础及进阶**

**一 pymysql模块-PyMySQL安装**

操作mysql数据库;主要是通过Python执行sql语句

**1.1 使用pypi**

• pypi即python package index

• 是python语言的软件仓库

• 官方站点为https://pypi.python.org



**1.2 通过pip安装PyMySQL模块**

1.3 使用国内镜像站点

room9pc01 ~]# cp /linux-soft/05/zzg\_pypkgs.tar.xz /root/nsd1906

room9pc01 ~]# cd /root/nsd1906

room9pc01 nsd1906]# tar -xf zzg\_pypkgs.tar.xz

(nsd1906) [root@room9pc01]# cd /root/nsd1906/zzg\_pypkgs/

(nsd1906) [root@room9pc01 zzg\_pypkgs]# pip install pymysql\_pkgs/\*

**二 pymysql模块-PyMySQL应用**

**2.1 连接数据库**

• 创建连接是访问数据库的第一步

conn = pymysql.connect(

host='127.0.0.1',

port=3306,

user='root',

passwd='tedu.cn',

db=nsd\_cloud',

charset='utf8')

**2.2 游标**

• 游标(cursor)就是游动的标识

• 通俗的说,一条sql取出对应n条结果资源的接口/句柄,就是游标,沿着游标可以一次取出一行

cursor = conn.cursor()

**2.3 插入数据**

• 对数据库表做修改操作,必须要commit

sql1 = "insert into departments(dep\_name) values(%s)"

result = cur.execute(sql1, ('development',))

sql2 = "insert into departments(dep\_name) values(%s)"

data = [('hr',), ('op',)]

result = cur.executemany(sql2, data)

sql3 = "insert into departments(dep\_name) values(%s)"

data = [('行政',), ('财务',), ('运营',)]

result = cur.executemany(sql3, data)

conn.commit()

**2.4 查询数据**

• 可以取出表中一条、多条或全部记录

sql4 = "select \* from departments"

cur.execute(sql4)

result = cur.fetchone()

print(result)

result2 = cur.fetchmany(2)

print(result2)

result3 = cur.fetchall()

print(result3)

**2.5 移动游标**

• 如果希望不是从头取数据,可以先移动游标

cur.scroll(1, mode="ralative")

cur.scroll(2, mode="absolute")

sql5 = "select \* from departments"

cur.execute(sql5)

cur.scroll(3, mode='absolute')

result4 = cur.fetchmany(2)

print(result4)

**2.6 修改数据**

• 通过update修改某一字段的值

sql6 = "update departments set dep\_name=%s where dep\_name=%s"

result = cur.execute(sql6, ('operations', 'op'))

print(result)

conn.commit()

**2.7 删除记录**

• 通过delete删除记录

sql7 = "delete from departments where dep\_id=%s"

result = cur.execute(sql7, (6,))

print(result)

conn.commit()

**案例1:创建小型数据库**

**为一个小公司创建数据库,用于记录员工情况以及发工资的信息**

**字段:员工ID,姓名,性别,部门,职位,出生日期,联系方式,工资日,基本工资,奖金,扣税,实发工资**

**数据库范式**

• 第一范式,第二范式,第三范式,巴斯-科德范式,第四范式,第五范式 六种

• 第一范式(1NF)是指在关系模型中,对域添加的一个规范要求,所有的域都应该是原子性的.如联系方式,可以分为家庭住址\电话号码\email等

• 第二范式(2NF)是在第一范式的基础上简历起来的,即满足第二范式必须先满足第一范式.第二范式要求数据库表中的每个实例或记录可以被唯一区分.简单来说,就是需要有一个主键.

• 第三范式(3NF)是第二范式的一个子集,及满足第三范式必须满足第二范式.第三范式要求:非主属性不能依赖于其他非主属性.如实发工资,它是通过基本工资和奖金算出来的.

根据数据库范式,需要把字段拆分到不同的表中:

• 员工表:员工ID,姓名,生日,email,部门ID #主键为员工ID

• 部门表:部门ID,部门名 #主键为部门ID

• 工资表:ID,工资日,员工ID,基本工资,奖金 #主键为ID

**opmysql.py**

import pymysql

# 创建到数据库服务器的连接

conn = pymysql.connect(

host='127.0.0.1',

port=3306,

user='root',

passwd='123456',

db='nsd1906',

charset='utf8'

)

# 创建游标。游标就像是文件对象，通过文件对象可以对文件读写

# 通过游标，可以对数据实现增删改查

cur = conn.cursor()

# 编写sql语句

create\_dep = """CREATE TABLE departments(

dep\_id INT, dep\_name VARCHAR(20),

PRIMARY KEY(dep\_id)

)"""

create\_emp = """CREATE TABLE employees(

emp\_id INT, emp\_name VARCHAR(20), birth\_date DATE,

email VARCHAR(50), dep\_id INT,

PRIMARY KEY(emp\_id),

FOREIGN KEY(dep\_id) REFERENCES departments(dep\_id)

)"""

create\_sal = """CREATE TABLE salary(

id INT, date DATE, emp\_id INT, basic INT, awards INT,

PRIMARY KEY(id),

FOREIGN KEY(emp\_id) REFERENCES employees(emp\_id)

)"""

# 执行sql语句

cur.execute(create\_dep)

cur.execute(create\_emp)

cur.execute(create\_sal)

# 如果是增删改操作，需要commit

conn.commit()

# 关闭

cur.close()

conn.close()

**selectmysql.py**

import pymysql

# 创建到数据库服务器的连接

conn = pymysql.connect(

host='127.0.0.1',

port=3306,

user='root',

passwd='123456',

db='nsd1906',

charset='utf8'

)

# 创建游标。游标就像是文件对象，通过文件对象可以对文件读写

# 通过游标，可以对数据实现增删改查

cur = conn.cursor()

select1 = 'SELECT \* FROM departments ORDER BY dep\_id'

cur.execute(select1)

result1 = cur.fetchone() # 取出一条记录

result2 = cur.fetchmany(2) # 继续取出2条记录

result3 = cur.fetchall() # 取出剩余全部记录

print(result1)

print('\*' \* 30)

print(result2)

print('\*' \* 30)

print(result3)

# 如果是增删改操作，需要commit

conn.commit()

# 关闭

cur.close()

conn.close()

**updatemysql.py**

import pymysql

# 创建到数据库服务器的连接

conn = pymysql.connect(

host='127.0.0.1',

port=3306,

user='root',

passwd='123456',

db='nsd1906',

charset='utf8'

)

# 创建游标。游标就像是文件对象，通过文件对象可以对文件读写

# 通过游标，可以对数据实现增删改查

cur = conn.cursor()

# 编写sql语句

# 修改

update1 = 'UPDATE departments SET dep\_name=%s WHERE dep\_name=%s'

cur.execute(update1, ('人力资源部', '人事部'))

# 如果是增删改操作，需要commit

conn.commit()

# 关闭

cur.close()

conn.close()

**deletemysql.py**

import pymysql

# 创建到数据库服务器的连接

conn = pymysql.connect(

host='127.0.0.1',

port=3306,

user='root',

passwd='123456',

db='nsd1906',

charset='utf8'

)

# 创建游标。游标就像是文件对象，通过文件对象可以对文件读写

# 通过游标，可以对数据实现增删改查

cur = conn.cursor()

# 编写sql语句

# 删除

delete1 = 'DELETE FROM departments WHERE dep\_id=%s'

cur.execute(delete1, (6,))

# 如果是增删改操作，需要commit

conn.commit()

# 关闭

cur.close()

conn.close()

**dbconn.py**

from sqlalchemy import create\_engine, Column, Integer, String

from sqlalchemy.ext.declarative import declarative\_base

# 创建到数据库的连接引擎

engine = create\_engine(

# mysql+pymysql://用户:密码@服务器/数据库?参数

'mysql+pymysql://root:123456@127.0.0.1/tedu1906?charset=utf8',

encoding='utf8',

# echo=True # 在终端显示debug日志，生产环境勿用

)

# 创建基类

Base = declarative\_base()

# 创建实体类，必须继承基类

class Departments(Base):

\_\_tablename\_\_ = 'departments'

dep\_id = Column(Integer, primary\_key=True)

dep\_name = Column(String(20), unique=True)

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

# 如果库中不存在对应的表则创建；存在就不创建了

Base.metadata.create\_all(engine)

三 **SQLAlchemy基础-SQLAlchemy概述**

**3.1 SQLAlchemy安装**

**3.2 SQLAlchemy简介**

**3.3 SQLAlchemy架构**

**3.4 ORM模型**

**四 SQLAlchemy基础-数据库对象管理**

**4.1 连接mysql**

**4.2 声明映射**

**4.3 创建映射类**

**4.4 创建架构**

**4.5 创建映射的实例**

**4.6 创建会话类**

**4.7添加新对象**

**4.8 外键约束**

**三 SQLAlchemy进阶-查询操作**

**3.1 基本查询**

• 通过作用于session的query()函数创建查询对象

• query()函数可以接收多种参数

from myorm import Session , Departments

session = Session()

for instance in

session.query(Departments).order\_by(Departments.dep\_id):

print(instance.dep\_id,instance.dep\_name)

**3.2 使用ORM描述符进行查询**

• 使用ORM描述符进行查询 • 返回值是元组

from myorm import Employees, Session

session = Session()

for name, phone in session.query(Employees.name, Employees.phone):

print(name, phone)

**3.3 排序**

• 通这order\_by()函数可以实现按指定字段排序

from myorm import Session, Departments

session = Session()

for instance in

session.query(Departments).order\_by(Departments.dep\_id):

print(instance.dep\_id, instance.dep\_name)

**3.4 提取部分数据**

• 通过“切片”的方式,实现部分数据的提取

from myorm import Session , Departments

session = Session()

for row in session.query(Departments, Departments.dep\_name)[2:5]:

print(row.Departments, row.dep\_name)

**3.5 结果过滤**

• 通过filter()函数实现结果过滤

from myorm import Session, Departments

session = Session()

for row in

session.query(Departments.dep\_name).filter(Departments.dep\_id==2):

print(row.dep\_name)

• filter()函数可以叠加使用

from myorm import Session, Salary

session = Session()

for row in session.query(Salary.emp\_id, Salary.base, Salary.award).filter(Salary.award>2000).filter(Salary.base>10000):

print(row.emp\_id)

**3.6 常用过滤操作符**

• 相等

query.filter(Employees.name=='john')

• 不相等

query.filter(Employees.name!='john')

• 模糊查询

query.filter(Employees.name.like(' %j '))

• in

query.filter(new\_emp.name.in\_(['bob', 'john'])

• not in

query.filter(~new\_emp.name.in\_(['bob', 'john'])

• 字段为空

query.filter(new\_emp.name.is\_(None))

• 字段不为空

query.filter(new\_emp.name.isnot(None))

3.7 查询对象返回值

• all()返回列表

• first()返回结果中的第一条记录

3.8 多表查询

• 通过join()方法实现多表查询

q = session.query(

Employees.name, Departments.dep\_name).join(Departments)

print(q.all())

**四 SQLAlchemy进阶-修改操作**

**4.1 更新数据**

• 通过会话的update()方法更新

from myorm import Session, Departments

session = Session()

q1 = session.query(Departments).filter(Departments.dep\_id==6)

q1.update({Departments.dep\_name: '运维部'})

session.commit()

session.close()

• 通过会话的字段赋值更新

from myorm import Session, Departments

session = Session()

q2 = session.query(Departments).get(1)

# get(1)查询主键是1的记录

q2.dep\_name = '开发部'

session.commit()

session.close()

**4.2 删除记录**

• 通过会话的delete()方法进行记录删除

from myorm import Session, Departments

session = Session()

q1 = session.query(Departments).get(7)

session.delete(q1)

session.commit()

session.close()