### 建库建表

-- 创建名为hrs的数据库并指定默认的字符集

create database hrs default character set utf8mb4;

-- 切换到hrs数据库

use hrs;

-- 创建部门表

create table `tb\_dept`

(

`dno` int not null comment '编号',

`dname` varchar(10) not null comment '名称',

`dloc` varchar(20) not null comment '所在地',

primary key (`dno`)

);

-- 插入4个部门

insert into `tb\_dept` values

(10, '会计部', '北京'),

(20, '研发部', '成都'),

(30, '销售部', '重庆'),

(40, '运维部', '深圳');

-- 创建员工表

create table `tb\_emp`

(

`eno` int not null comment '员工编号',

`ename` varchar(20) not null comment '员工姓名',

`job` varchar(20) not null comment '员工职位',

`mgr` int comment '主管编号',

`sal` int not null comment '员工月薪',

`comm` int comment '每月补贴',

`dno` int not null comment '所在部门编号',

primary key (`eno`),

constraint `fk\_emp\_mgr` foreign key (`mgr`) references tb\_emp (`eno`),

constraint `fk\_emp\_dno` foreign key (`dno`) references tb\_dept (`dno`)

);

-- 插入14个员工

insert into `tb\_emp` values

(7800, '张三丰', '总裁', null, 9000, 1200, 20),

(2056, '乔峰', '分析师', 7800, 5000, 1500, 20),

(3088, '李莫愁', '设计师', 2056, 3500, 800, 20),

(3211, '张无忌', '程序员', 2056, 3200, null, 20),

(3233, '丘处机', '程序员', 2056, 3400, null, 20),

(3251, '张翠山', '程序员', 2056, 4000, null, 20),

(5566, '宋远桥', '会计师', 7800, 4000, 1000, 10),

(5234, '郭靖', '出纳', 5566, 2000, null, 10),

(3344, '黄蓉', '销售主管', 7800, 3000, 800, 30),

(1359, '胡一刀', '销售员', 3344, 1800, 200, 30),

(4466, '苗人凤', '销售员', 3344, 2500, null, 30),

(3244, '欧阳锋', '程序员', 3088, 3200, null, 20),

(3577, '杨过', '会计', 5566, 2200, null, 10),

(3588, '朱九真', '会计', 5566, 2500, null, 10);

#### 删除数据

import pymysql

no = int(input('部门编号: '))

# 1. 创建连接（Connection）

conn = pymysql.connect(host='127.0.0.1', port=3306,

user='guest', password='Guest.618',

database='hrs', charset='utf8mb4',

autocommit=True)

try:

# 2. 获取游标对象（Cursor）

with conn.cursor() as cursor:

# 3. 通过游标对象向数据库服务器发出SQL语句

affected\_rows = cursor.execute(

'delete from `tb\_dept` where `dno`=%s',

(no, )

)

if affected\_rows == 1:

print('删除部门成功!!!')

finally:

# 5. 关闭连接释放资源

conn.close()

#### 更新数据

import pymysql

no = int(input('部门编号: '))

name = input('部门名称: ')

location = input('部门所在地: ')

# 1. 创建连接（Connection）

conn = pymysql.connect(host='127.0.0.1', port=3306,

user='guest', password='Guest.618',

database='hrs', charset='utf8mb4')

try:

# 2. 获取游标对象（Cursor）

with conn.cursor() as cursor:

# 3. 通过游标对象向数据库服务器发出SQL语句

affected\_rows = cursor.execute(

'update `tb\_dept` set `dname`=%s, `dloc`=%s where `dno`=%s',

(name, location, no)

)

if affected\_rows == 1:

print('更新部门信息成功!!!')

# 4. 提交事务

conn.commit()

except pymysql.MySQLError as err:

# 4. 回滚事务

conn.rollback()

print(type(err), err)

finally:

# 5. 关闭连接释放资源

conn.close()

#### 查询数据

1. 查询部门表的数据。

import pymysql

# 1. 创建连接（Connection）

conn = pymysql.connect(host='127.0.0.1', port=3306,

user='guest', password='Guest.618',

database='hrs', charset='utf8mb4')

try:

# 2. 获取游标对象（Cursor）

with conn.cursor() as cursor:

# 3. 通过游标对象向数据库服务器发出SQL语句

cursor.execute('select `dno`, `dname`, `dloc` from `tb\_dept`')

# 4. 通过游标对象抓取数据

row = cursor.fetchone()

while row:

print(row)

row = cursor.fetchone()

except pymysql.MySQLError as err:

print(type(err), err)

finally:

# 5. 关闭连接释放资源

conn.close()

下面我们为大家讲解一个将数据库表数据导出到 Excel 文件的例子，我们需要先安装openpyxl三方库，命令如下所示。

pip install openpyxl

接下来，我们通过下面的代码实现了将数据库hrs中所有员工的编号、姓名、职位、月薪、补贴和部门名称导出到一个 Excel 文件中。

import openpyxl

import pymysql

# 创建工作簿对象

workbook = openpyxl.Workbook()

# 获得默认的工作表

sheet = workbook.active

# 修改工作表的标题

sheet.title = '员工基本信息'

# 给工作表添加表头

sheet.append(('工号', '姓名', '职位', '月薪', '补贴', '部门'))

# 创建连接（Connection）

conn = pymysql.connect(host='127.0.0.1', port=3306,

user='guest', password='Guest.618',

database='hrs', charset='utf8mb4')

try:

# 获取游标对象（Cursor）

with conn.cursor() as cursor:

# 通过游标对象执行SQL语句

cursor.execute(

'select `eno`, `ename`, `job`, `sal`, coalesce(`comm`, 0), `dname` '

'from `tb\_emp` natural join `tb\_dept`'

)

# 通过游标抓取数据

row = cursor.fetchone()

while row:

# 将数据逐行写入工作表中

sheet.append(row)

row = cursor.fetchone()

# 保存工作簿

workbook.save('hrs.xlsx')

except pymysql.MySQLError as err:

print(err)

finally:

# 关闭连接释放资源

conn.close()

# [Python3 多线程(连接池)操作MySQL插入数据](https://www.cnblogs.com/insane-Mr-Li/p/11634417.html)

1.主要模块  
DBUtils : 允许在多线程应用和数据库之间连接的模块套件  
Threading : 提供多线程功能

2.创建连接池  
PooledDB 基本参数：

mincached : 最少的空闲连接数，如果空闲连接数小于这个数，Pool自动创建新连接;  
maxcached : 最大的空闲连接数，如果空闲连接数大于这个数，Pool则关闭空闲连接;  
maxconnections : 最大的连接数;  
blocking : 当连接数达到最大的连接数时，在请求连接的时候，如果这个值是True，请求连接的程序会一直等待，直到当前连接数小于最大连接数，如果这个值是False，会报错

import pymysql

import threading

import re

import time

from queue import Queue

from DBUtils.PooledDB import PooledDB

class ThreadInsert(object):

"多线程并发MySQL插入数据"

def \_\_init\_\_(self):

start\_time = time.time()

self.pool = self.mysql\_connection()

self.data = self.getData()

self.mysql\_delete()

self.task()

print("========= 数据插入,共耗时:{}'s =========".format(round(time.time() - start\_time, 3)))

def mysql\_connection(self):

maxconnections = 15 # 最大连接数

pool = PooledDB(

pymysql,

maxconnections,

host='localhost',

user='root',

port=3306,

passwd='123456',

db='test\_DB',

use\_unicode=True)

return pool

def getData(self):

st = time.time()

with open("10w.txt", "rb") as f:

data = []

for line in f:

line = re.sub("\s", "", str(line, encoding="utf-8"))

line = tuple(line[1:-1].split("\"\""))

data.append(line)

n = 100000 # 按每10万行数据为最小单位拆分成嵌套列表

result = [data[i:i + n] for i in range(0, len(data), n)]

print("共获取{}组数据,每组{}个元素.==>> 耗时:{}'s".format(len(result), n, round(time.time() - st, 3)))

return result

def mysql\_delete(self):

st = time.time()

con = self.pool.connection()

cur = con.cursor()

sql = "TRUNCATE TABLE test"

cur.execute(sql)

con.commit()

cur.close()

con.close()

print("清空原数据.==>> 耗时:{}'s".format(round(time.time() - st, 3)))

def mysql\_insert(self, \*args):

con = self.pool.connection()

cur = con.cursor()

sql = "INSERT INTO test(sku, fnsku, asin, shopid) VALUES(%s, %s, %s, %s)"

try:

cur.executemany(sql, \*args)

con.commit()

except Exception as e:

con.rollback() # 事务回滚

print('SQL执行有误,原因:', e)

finally:

cur.close()

con.close()

def task(self):

q = Queue(maxsize=10) # 设定最大队列数和线程数

st = time.time()

while self.data:

content = self.data.pop()

t = threading.Thread(target=self.mysql\_insert, args=(content,))

q.put(t)

if (q.full() == True) or (len(self.data)) == 0:

thread\_list = []

while q.empty() == False:

t = q.get()

thread\_list.append(t)

t.start()

for t in thread\_list:

t.join()

print("数据插入完成.==>> 耗时:{}'s".format(round(time.time() - st, 3)))

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

ThreadInsert()

# Python异步操作MySQL示例【使用aiomysql】

pip install aiomysql

**基本的异步连接connection**

import asyncio

from aiomysql import create\_pool

loop = asyncio.get\_event\_loop()

async def go():

async with create\_pool(host='127.0.0.1', port=3306,

user='root', password='',

db='mysql', loop=loop) as pool:

async with pool.get() as conn:

async with conn.cursor() as cur:

await cur.execute("SELECT 42;")

value = await cur.fetchone()

print(value)

loop.run\_until\_complete(go())

**异步的连接池 pool**

import asyncio

import aiomysql

async def test\_example(loop):

pool = await aiomysql.create\_pool(host='127.0.0.1', port=3306,

user='root', password='',

db='mysql', loop=loop)

async with pool.acquire() as conn:

async with conn.cursor() as cur:

await cur.execute("SELECT 42;")

print(cur.description)

(r,) = await cur.fetchone()

assert r == 42

pool.close()

await pool.wait\_closed()

loop = asyncio.get\_event\_loop()

loop.run\_until\_complete(test\_example(loop))

**对象关系映射SQLAlchemy - Object Relationship Mapping**

可以随意定义表结构，轻松调用查询、插入等操作方法

import asyncio

import sqlalchemy as sa

from aiomysql.sa import create\_engine

metadata = sa.MetaData()

tbl = sa.Table('tbl', metadata,

sa.Column('id', sa.Integer, primary\_key=True),

sa.Column('val', sa.String(255)))

async def go(loop):

engine = await create\_engine(user='root', db='test\_pymysql',

host='127.0.0.1', password='', loop=loop)

async with engine.acquire() as conn:

await conn.execute(tbl.insert().values(val='abc'))

await conn.execute(tbl.insert().values(val='xyz'))

async for row in conn.execute(tbl.select()):

print(row.id, row.val)

engine.close()

await engine.wait\_closed()

loop = asyncio.get\_event\_loop()

loop.run\_until\_complete(go(loop))