一、数据库设计

**1、安装mariadb**

[root@room9pc01 ~]# yum -y install mariadb-server

[root@room9pc01 ~]# systemctl {enable|start} mariadb

[root@room9pc01 ~]# mysqladmin password 123

**2、数据库设计：员工管理**

1）字段

姓名、性别、工号、部门、联系方式、出生日期、工资日、基本工资、奖金、总工资

2）数据库设计原则：可以设计多张表来消除冗余

3）字段要满足的原则：数据库范式

第一范式(1NF)：数据库表的每一列都应该具有原子性(不可分割)。如：联系方式可以再分为住址、电话、email

第二范式(2NF)：在1NF基础上建立。要求数据库表中每个实例或记录必须被唯一地区分（必须有主键）

第三范式(3NF)：在2NF基础上建立，任何非主属性不依赖于其他非主属性。如：总工资依赖基本工资和奖金，不适合出现在数据库，一般在查询时再"+"处理

**3、数据库表的确定**

departments：部门ID、部门名

employee：工号、姓名、性别、部门ID、email、出生日期、

salary：工资日、工号、基本工资、奖金

**4、创建库**

MariaDB [(none)]> create database nsd1808 default charset utf8;

二、使用pypi

**1、pip的使用**

[root@room9pc01 ~]# yum install -y gcc

#安装依赖包

[root@room9pc01 packages]# pip3 install PyMySQL-0.8.0.tar.gz

#本地安装

[root@room9pc01 ~]# pip3 install pymysql

#在线安装

**2、国内站点加速pip安装**

[root@room9pc01 ~]# mkdir ~/.pip

[root@room9pc01 ~]# vim ~/.pip/pip.conf

[global]

index-url=http://pypi.douban.com/simple/

[install]

trusted-host=pypi.douban.com

三、pymysql模块

import pymysql

conn = pymysql.connect( #连接数据库

host='127.0.0.1',

port=3306,

user='root',

passwd='123',

db='nsd1808',

charset='utf8'

)

cursor = conn.cursor() #创建游标

**###建表###**

create\_dep = """CREATE TABLE departments(

dep\_id INT, dep\_name VARCHAR(20) NOT NULL UNIQUE,

PRIMARY KEY(dep\_id))"""

cursor.execute(create\_dep) #执行建表SQL语句

create\_emp = """CREATE TABLE employee()

... ..."""

create\_sal = """CREATE TABLE salary()

... ..."""

**###插入记录###**

insert1 = 'INSERT INTO departments VALUES(%s, %s)'

cursor.execute(insert1, (1, 'HR')) #插入单条记录

deps = [(2, '运维'), (3, '开发'), (4, '测试')]

cursor.executemany(insert1, deps) #插入多条记录

conn.commit() #提交

**###查询记录###**

query1 = 'SELECT \* FROM departments ORDER BY dep\_id'

cursor.execute(query1)

print(cursor.fetchone()) #取出单条结果

#打印(1, 'HR')

cursor.scroll(1, mode="relative") #相对移动游标1位

print(cursor.fetchmany(2)) #取出多条结果

#打印((3, '开发'), (4, '测试'))

cursor.scroll(0, mode="absolute") #绝对移动游标0位

print(cursor.fetchall()) #取出所有结果

#打印((1, 'HR'), (2, '运维'), (3, '开发'), (4, '测试'))

**###更新记录###**

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

cursor.execute(update1, ('人事', 'HR'))

conn.commit()

**###删除记录###**

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

cursor.execute(delete1, (1, ))

conn.commit()

cursor.close() #关闭游标

conn.close() #关闭数据库连接

1. SQLAlchemy模块

**1、ORM模型**

O：Object对象

R：Relatinoship关系型数据库

M：Map映射

sqlalchemy把数据库中的表编写成class,

把数据库字段的各种数据类型编写成了class，

数据库的每个字段编写成了Column类,

表中的一行记录对应成class的一个实例。

**2、连接db、创建Session、创建表**

# “This is dbconn.py”

from sqlalchemy import create\_engine

from sqlalchemy.ext.declarative import declarative\_base

from sqlalchemy import Column, Integer, String, ForeignKey, Date

from sqlalchemy.orm import sessionmaker

#连接指定 'mysql+pymysql://用户名:密码@服务器/数据库?参数'

engine = create\_engine(

'mysql+pymysql://root:123@localhost/tedu1808?charset=utf8',

encoding='utf8',

echo=True #显示调试信息，生产环境不要设置

)

Base = declarative\_base()

#生成ORM需要的基类，其实是调用函数返回一个类

Session = sessionmaker(bind=engine)

#创建会话类，访问数据库

class Departments(Base): #派生一个表class

\_\_tablename\_\_ = 'departments' #定义表名

dep\_id = Column(Integer, primary\_key=True) #定义字段名

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

def \_\_str\_\_(self):

return "<部门: %s>" % self.dep\_name

class Employees(Base):

... ...

dep\_id = Column(Integer, ForeignKey('departments.dep\_id'))

class Salary(Base):

... ...

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

Base.metadata.create\_all(engine) #创建表

**3、插入记录(实例)**

# “This is insert\_sql.py”

from dbconn import Departments, Employees, Salary, Session

session = Session()

hr = Departments(dep\_id=1, dep\_name='HR') #创建一个Dep..实例

session.add(hr) #添加单条记录

session.commit() #提交

ops = Departments(dep\_id=2, dep\_name='运维部')

dev = Departments(dep\_id=3, dep\_name='开发部')

qs = Departments(dep\_id=4, dep\_name='测试部')

session.add\_all([ops, dev, qs]) #添加多条记录

session.commit()

session.close()

**4、查询记录(实例)**

# “This is query\_sql.py”

from dbconn import Departments, Employees, Session

session = Session()

query1 = session.query(Departments)

print(query1) #返回SQL语句

print(query1.all()) #返回每行实例构成的列表

for dep in query1: #历遍表的所有实例(相当于每一行记录)

print(dep) #调用\_\_str\_\_()

for dep in query1:

print("%s:%s" % (dep.dep\_id, dep.dep\_name))

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

query2 = session.query(Employees.emp\_name, Employees.email)

print(query2)

print(query2.all()) #返回由元组构成的列表

for name, email in query2: #历遍列表中的所有元组

print('%s: %s' % (name, email)) #打印emp\_name:email

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

query3 = session.query(Employees.emp\_name.label('部门'))

#定义字段别名

for i in query3:

print(i.部门) #打印emp\_name

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

query4 = session.query(Departments).order\_by(Departments.dep\_id)

query5 = session.query...order\_by(Departments.dep\_id)[1:3]

#排序查询、[1:3]为切割过滤dep\_id=1~2

for dep in query4:

print(dep.dep\_id, dep.dep\_name) #按顺序打印dep\_id dep\_name

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

query7 = session.query(Employees.emp\_name, Employees.email)\

.filter(Employees.dep\_id==2)\

.filter(Employees.email.like('%@qq.com'))

#filter()过滤，相当于where；模糊过滤使用filter(字段名.like())

print(query7.count())

#统计行数

for name, email in query7:

print(name, email) #打印dep\_id=2且qq.com的emp\_name, email

**Tips：**

**1）关于filter()过滤**

|  |  |
| --- | --- |
| .filter(Employees.name=='john') | 相等 |
| .filter(Employees.name!='john') | 不相等 |
| .filter(Employees.name.like(' %j')) | 模糊查询 |
| .filter(Employees.name.in\_(['Tom', 'John'])) | in |
| .filter(~Employees.name.in\_(['Tom', 'John'])) | not in |
| .filter(Employees.name.is\_(None)) | 字段为空 |
| .filter(Employees.name.isnot\_(None)) | 字段不为空 |
| from sqlalchemy import and\_  .filter(and\_(Empes.dep\_id==2, Empes.sex=='man')) | 多重条件and |
| from sqlalchemy import or\_  .filter(or\_(Empes.dep\_id==2, Empes.sex=='man')) | 多重条件or |

1. **查询对象返回值**

all() 返回列表

first() 返回结果中的第一条记录

one() 取出所有记录，如果不是一条记录则抛出异常

scalar() 相当于调用one()，返回第一列的值

**3）多表查询**

query9 = session.query(Employees.emp\_name, Departments.dep\_name)\

.join(Departments, Employees.dep\_id==Departments.dep\_id)

#注意join()第一个参数对应后表

print(query9.all())

**5、修改记录(实例)**

from dbconn import Departments, Employees, Session

session = Session()

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

xs = xs.one() #取出实例

xs.dep\_name = '销售' #实例属性重新赋值

session.commit() #提交

session.close()

**6、删除记录(实例)**

from dbconn import Departments, Employees, Session

session = Session()

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

xs = xs.one() #取出实例

session.delete(xs) #删除实例

session.commit() #提交

session.close()

五、案例：模拟用户登陆信息系统

1、支持新用户注册,新用户名和密文口令(md5)注册到数据库中

2、支持老用户登陆,用户名和密码正确提示登陆成功

3、主程序通过循环询问进行何种操作,根据用户的选择,执行注册或是登陆操作