lab7 ORM介绍和基础查询练习

1. ORM的介绍

ORM：Object Relation Mapping，最初主要描述的是程序中的Object对象和关系型数据库中Relation关系(表)之间的映射关系，目前来说也是描述程序中对象和数据库中数据记录之间的映射关系的统称，是一种进行程序和数据库之间数据持久化的一种编程思想。

特点是操纵Python对象而不是SQL查询，也就是在代码层面考虑的是对象，而不是SQL，体现的是 一种程序化思维，这样使得Python程序更加简洁易读。

增删改操作

常规情况下，软件程序中的ORM操作主要有四个操作场景：增、删、改、查. 核心操作一般会区分 为：增删改、查询

增加操作：程序中存在的一个对象Object数据，通过[ORM]核心模块进行增加的函数定义将对象保存到数据库的操作过程；
如：注册操作中，通过用户输入的账号密码等信息创建了一个独立的对象，通过`add()`函数将对象增加保存到数据库中，数据库中就存在用户这个对象数据了。

修改操作：程序中存在的一个对象Object数据，有自己的id编号(可以是程序中自行赋值定义、更多的操作是从数据库中查询出来存在的一个对象)，通过[ORM]核心模块进行修改函数的定义将对象改变的数据更新到数据库中已经存在的记录中的过程；
如:用户更改登录密码操作时，根据程序中查询得到的一个用户[id编号、账号、密码、..]，在程序中通过改变其密码属性数据，然后通过`update()`函数将改变的数据更新保存到数据库中，数据库中原来的数据就发生了新的改变。

删除操作：程序中存在的一个对象或者已知的id编号，通过主键编号或者对象的任意属性进行数据库中数据记录的删除的操作过程；
如：管理员删除某个会员账号的操作，通过获取要删除会员的账号，然后通过`delete()`函数将要删除的会员信息告知数据库执行删除操作，数据库中的某条存在的数据记录就被删除掉了。

2.sqlalchemy

2.1 sqlalchemy的介绍和安装

SQLAlchemy 是一个Python 的SQL 工具包以及数据库对象映射框架。它包含整套企业级持久化模 式，专门为高效和高性能的数据库访问。

如果想在本地安装，可使用以下语句：

```
pip install SQLAlchemy
pip install psycopg2
```

在该水杉环境中已经安装完成，直接导入即可：

In [40]:
```python
import sqlalchemy
```

2.2 sqlalchemy 的简单操作

In [41]:
```python
from sqlalchemy import Column, String, create_engine, Integer, Text, Date
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base
import time
```

2.2.1 建立连接

在网址 `postgresql://ecnu学号:ECNU学号@pgm-uf6t8021ru5tac71.rwlb.rds.aliyuncs.com:5432/ecnu学号` 中填入自己的学号

In [42]:
```python
from sqlalchemy import create_engine
engine = create_engine("postgresql://ecnu10222140402:ECNU10222140402@pgm-uf6t802
    echo=True,
    pool_size=8,
    pool_recycle=60*30
)
```

2.2.2 建立会话（session）

session 用于创建程序与数据库之间的对话.

In [43]:
```python
from sqlalchemy.orm import sessionmaker
# 创建session
DbSession = sessionmaker(bind=engine)
session = DbSession()
```

session 的常见用法:

1. commit：提交了一个事务
2. rollback：回滚
3. close：关闭

2.2.3 创建表格

declarative_base()是sqlalchemy内部封装的一个方法，通过其构造一个基类，这个基类和它的子类，可以将Python类和数据库表关联映射起来。

数据库表模型类通过tablename和表关联起来，Column表示数据表的列。

In [44]:
```python
from sqlalchemy.orm import declarative_base
```

```
# 创建对象的基类:
Base = declarative_base()
```

In [45]:
```
# 定义User对象:
class User(Base):
    # 表的名字:
    __tablename__ = 'users'

    # 表的结构:
    id = Column(Integer, autoincrement=True, primary_key=True, unique=True, null
    name = Column(String(50), nullable=False)
    sex = Column(String(4), nullable=False)
    nation = Column(String(20), nullable=False)
    birth = Column(String(8), nullable=False)
    id_address = Column(Text, nullable=False)
    id_number = Column(String(18), nullable=False)
    creater = Column(String(32))
    create_time = Column(String(20), nullable=False)
    updater = Column(String(32))
    update_time = Column(String(20), nullable=False,
    default=time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()),
    onupdate=time.strftime("%Y-%m-%d %H:%M:%S",time.localtime()))
    comment = Column(String(200))

def createTable():
    # 创建所有继承于Base的类对应的表
    Base.metadata.create_all(engine)

createTable()
```

```
2024-11-20 13:54:52,643 INFO sqlalchemy.engine.Engine select pg_catalog.version()
2024-11-20 13:54:52,644 INFO sqlalchemy.engine.Engine [raw sql] {}
2024-11-20 13:54:52,646 INFO sqlalchemy.engine.Engine select current_schema()
2024-11-20 13:54:52,647 INFO sqlalchemy.engine.Engine [raw sql] {}
2024-11-20 13:54:52,649 INFO sqlalchemy.engine.Engine show standard_conforming_st
rings
2024-11-20 13:54:52,650 INFO sqlalchemy.engine.Engine [raw sql] {}
2024-11-20 13:54:52,652 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:54:52,655 INFO sqlalchemy.engine.Engine SELECT pg_catalog.pg_class.
relname
FROM pg_catalog.pg_class JOIN pg_catalog.pg_namespace ON pg_catalog.pg_namespace.
oid = pg_catalog.pg_class.relnamespace
WHERE pg_catalog.pg_class.relname = %(table_name)s AND pg_catalog.pg_class.relkin
d = ANY (ARRAY[%(param_1)s, %(param_2)s, %(param_3)s, %(param_4)s, %(param_5)s])
AND pg_catalog.pg_table_is_visible(pg_catalog.pg_class.oid) AND pg_catalog.pg_nam
espace.nspname != %(nspname_1)s
2024-11-20 13:54:52,656 INFO sqlalchemy.engine.Engine [generated in 0.00066s] {'t
able_name': 'users', 'param_1': 'r', 'param_2': 'p', 'param_3': 'f', 'param_4':
'v', 'param_5': 'm', 'nspname_1': 'pg_catalog'}
2024-11-20 13:54:52,663 INFO sqlalchemy.engine.Engine
CREATE TABLE users (
        id SERIAL NOT NULL,
        name VARCHAR(50) NOT NULL,
        sex VARCHAR(4) NOT NULL,
        nation VARCHAR(20) NOT NULL,
        birth VARCHAR(8) NOT NULL,
        id_address TEXT NOT NULL,
        id_number VARCHAR(18) NOT NULL,
        creater VARCHAR(32),
        create_time VARCHAR(20) NOT NULL,
        updater VARCHAR(32),
        update_time VARCHAR(20) NOT NULL,
        comment VARCHAR(200),
        PRIMARY KEY (id),
        UNIQUE (id)
)


2024-11-20 13:54:52,663 INFO sqlalchemy.engine.Engine [no key 0.00041s] {}
2024-11-20 13:54:52,679 INFO sqlalchemy.engine.Engine COMMIT
```

2.2.4 插入数据

In [46]:
```python
# 插入操作
def insertData():

    # 创建会话
    session = DbSession()

    # 创建新User对象:
    local_time = time.strftime("%Y-%m-%d %H:%M:%S",time.localtime())
    new_user = User(name='mdotdot', sex='女', nation='汉',
    birth='19981021', id_address='ECNU', id_number='441242142142',
    create_time=local_time)
    new_user1 = User(name='xdot', sex='男', nation='汉',
    birth='19990110', id_address='ECNU', id_number='451242142142',
    create_time=local_time)

    # 添加到session:
```

```
    session.add(new_user)
    session.add(new_user1)
    # 提交即保存到数据库:
    session.commit()

    # 关闭session:
    session.close()

insertData()
```

```
2024-11-20 13:54:53,441 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:54:53,445 INFO sqlalchemy.engine.Engine INSERT INTO users (name, se
x, nation, birth, id_address, id_number, creater, create_time, updater, update_ti
me, comment) SELECT p0::VARCHAR, p1::VARCHAR, p2::VARCHAR, p3::VARCHAR, p4::TEXT,
p5::VARCHAR, p6::VARCHAR, p7::VARCHAR, p8::VARCHAR, p9::VARC ... 426 characters t
runcated ... p4, p5, p6, p7, p8, p9, p10, sen_counter) ORDER BY sen_counter RETUR
NING users.id, users.id AS id__1
2024-11-20 13:54:53,445 INFO sqlalchemy.engine.Engine [generated in 0.00018s (ins
ertmanyvalues) 1/1 (ordered)] {'create_time__0': '2024-11-20 13:54:53', 'birth__
0': '19981021', 'id_address__0': 'ECNU', 'sex__0': '女', 'nation__0': '汉', 'comm
ent__0': None, 'name__0': 'mdotdot', 'id_number__0': '441242142142', 'updater__
0': None, 'update_time__0': '2024-11-20 13:54:52', 'creater__0': None, 'create_ti
me__1': '2024-11-20 13:54:53', 'birth__1': '19990110', 'id_address__1': 'ECNU',
'sex__1': '男', 'nation__1': '汉', 'comment__1': None, 'name__1': 'xdot', 'id_num
ber__1': '451242142142', 'updater__1': None, 'update_time__1': '2024-11-20 13:54:
52', 'creater__1': None}
2024-11-20 13:54:53,449 INFO sqlalchemy.engine.Engine COMMIT
```

2.2.5 查询数据

SQL 与 SQLalchemy 的写法区别为:

- query： 对应 SELECT xxx FROM xxx
- filter/filter_by： 对应 WHERE， fillter 可以进行比较运算(==, >, < ...)来对条件进行灵活的运用，不同的条件用逗号分割，fillter_by 只能指定参数传参来获取查询结果。
- limit： 对应 limit()
- order by： 对应 order_by()
- group by： 对应 group_by()

返回结果数量可以有以下两种方式:

all()

- 查询所有
- 返回一个列表对象

first()

- 查询第一个符合条件的对象
- 返回一个对象

在ORM中，查询也有和SQL类似的关键字

In [47]:
```
from sqlalchemy import and_,or_
```

like	session.query(Person).filter(Person.desc.like("活%")).all()
not like	session.query(Person).filter(Person.desc.notlike("活%")).all()
is(等价于==)	session.query(Person).filter(Person.username.is_(None)).all(), session.query(Person).filter(Person.username == None).all()
isnot(等价于 !=)	session.query(Person).filter(Person.username.isnot(None)).all(), session.query(Person).filter(Person.username != None).all()
正则查询	session.query(Person).filter(Person.password.op("regexp")(r"^[\u4e00-\u9fa5]+")).all()
count	session.query(Person).filter(Person.desc.like("活%")).count()
in	session.query(Person).filter(Person.username.in_(['Mark', 'Tony'])).all()
not in(等价于~in)	session.query(Person).filter(Person.username.notin_(['Mark', 'Tony'])).all(), session.query(Person).filter(~Person.username.in_(['Mark', 'Tony'])).all()
AND(导入and_)	more_person = session.query(Person).filter(and_(Person.password=='123456',Person.desc=="可爱")).all()
OR(导入or_)	session.query(Person).filter(or_(Person.password=='123456',Person.desc=="活泼")).all()
limit	session.query(Person).filter(Person.desc.notlike("活%")).limit(1).all()
offset	session.query(Person).filter(Person.desc.like("活%")).offset(1).all()
order_by （asc正序)	session.query(Person).order_by(Person.username.desc()).all()
group_by	session.query(Person).group_by(Person.desc).all()
between	session.query(Protocols.protocolName).filter(Protocols.id.between(1, 3)).all()

聚合函数

In [48]:
```python
from sqlalchemy import func, extract
```

关键字	示例
count	session.query(Person.password, func.count(Person.id)).group_by(Person.password).all()
sum	session.query(Person.password, func.sum(Person.id)).group_by(Person.password).all()
max	session.query(Person.password, func.max(Person.id)).group_by(Person.password).all()
min	session.query(Person.password, func.min(Person.id)).group_by(Person.password).all()
having	session.query(Person.password, func.count(Person.id)).group_by(Person.password).having(func.count(Person.id) > 1).all()

In [49]:
```
#查询所有数据
def find_all():
    # 创建Session
    session = DbSession()

    user = session.query(User).all()
    for i in user:
        print('id:',i.id)
        print('name:', i.name)
        print('id_address:', i.id_address)
        print('id_number:', i.id_number)

    session.close() # 关闭Session

find_all()
```

```
2024-11-20 13:54:59,998 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:00,001 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
2024-11-20 13:55:00,001 INFO sqlalchemy.engine.Engine [generated in 0.00066s] {}
id: 1
name: mdotdot
id_address: ECNU
id_number: 441242142142
id: 2
name: xdot
id_address: ECNU
id_number: 451242142142
2024-11-20 13:55:00,004 INFO sqlalchemy.engine.Engine ROLLBACK
```

In [50]:
```
# 查询操作
def selectData():
    # 创建Session
    session = DbSession()

    # 创建Query查询，filter是where条件，最后调用first()返回唯一行，如果调用all()
    user = session.query(User).filter(and_(User.id == '1' ,User.name == 'mdotdot

    if user:
        print('name:', user.name)
        print('id_address:', user.id_address)

    session.close() # 关闭Session

selectData()
```

```
2024-11-20 13:55:00,543 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:00,545 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
WHERE users.id = %(id_1)s AND users.name = %(name_1)s
 LIMIT %(param_1)s
2024-11-20 13:55:00,546 INFO sqlalchemy.engine.Engine [generated in 0.00060s] {'i
d_1': '1', 'name_1': 'mdotdot', 'param_1': 1}
name: mdotdot
id_address: ECNU
2024-11-20 13:55:00,550 INFO sqlalchemy.engine.Engine ROLLBACK
```

- 还可以将查询的参数单独写:

In [51]:
```python
# 创建Session
session = DbSession()

filter = (User.name=='mdotdot')
user = session.query(User).filter(filter).first()
print(user.name)

session.close()
```

```
2024-11-20 13:55:01,919 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:01,921 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
WHERE users.name = %(name_1)s
 LIMIT %(param_1)s
2024-11-20 13:55:01,923 INFO sqlalchemy.engine.Engine [generated in 0.00260s] {'n
ame_1': 'mdotdot', 'param_1': 1}
mdotdot
2024-11-20 13:55:01,926 INFO sqlalchemy.engine.Engine ROLLBACK
```

2.2.6 修改数据

- 适用于批量修改

In [52]:
```python
session = DbSession()

session.query(User).filter_by(name = "mdotdot").update({'name':"Jack"})
session.commit() # 提交即保存到数据库

session.close()
```

```
2024-11-20 13:55:03,200 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:03,202 INFO sqlalchemy.engine.Engine UPDATE users SET name=%(nam
e)s, update_time=%(update_time)s WHERE users.name = %(name_1)s
2024-11-20 13:55:03,203 INFO sqlalchemy.engine.Engine [generated in 0.00076s] {'n
ame': 'Jack', 'update_time': '2024-11-20 13:54:52', 'name_1': 'mdotdot'}
2024-11-20 13:55:03,207 INFO sqlalchemy.engine.Engine COMMIT
```

In [53]:
```python
find_all()
```

```
2024-11-20 13:55:03,871 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:03,872 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
2024-11-20 13:55:03,872 INFO sqlalchemy.engine.Engine [cached since 3.872s ago]
{}
id: 2
name: xdot
id_address: ECNU
id_number: 451242142142
id: 1
name: Jack
id_address: ECNU
id_number: 441242142142
2024-11-20 13:55:03,875 INFO sqlalchemy.engine.Engine ROLLBACK
```

修改成功

- 适用于获取对象的值,进行操作之后修改

In [54]:
```python
# 更新操作
def updateData():
    session = DbSession() # 创建会话

    users = session.query(User).filter(User.name=="Jack").first()# 查询条件

    if users:
        users.id_number = "abcd" # 更新操作
        session.add(users) # 添加到会话
        session.commit() # 提交即保存到数据库

    session.close() # 关闭会话

updateData()
```

```
2024-11-20 13:55:06,819 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:06,821 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
WHERE users.name = %(name_1)s
 LIMIT %(param_1)s
2024-11-20 13:55:06,821 INFO sqlalchemy.engine.Engine [cached since 4.901s ago]
{'name_1': 'Jack', 'param_1': 1}
2024-11-20 13:55:06,825 INFO sqlalchemy.engine.Engine UPDATE users SET id_number
=%(id_number)s, update_time=%(update_time)s WHERE users.id = %(users_id)s
2024-11-20 13:55:06,825 INFO sqlalchemy.engine.Engine [generated in 0.00049s] {'i
d_number': 'abcd', 'update_time': '2024-11-20 13:54:52', 'users_id': 1}
2024-11-20 13:55:06,827 INFO sqlalchemy.engine.Engine COMMIT
```

In [55]:

```
find_all()
```

```
2024-11-20 13:55:07,842 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:07,842 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
2024-11-20 13:55:07,843 INFO sqlalchemy.engine.Engine [cached since 7.842s ago]
{}
id: 2
name: xdot
id_address: ECNU
id_number: 451242142142
id: 1
name: Jack
id_address: ECNU
id_number: abcd
2024-11-20 13:55:07,845 INFO sqlalchemy.engine.Engine ROLLBACK
```

已修改成功

2.2.7 删除数据

- 直接将删除语句写成一行

In [56]:

```
session = DbSession()

delete_query = session.query(User).filter(User.name=='xdot').delete()
session.commit() # 提交会话

session.close()
```

```
2024-11-20 13:55:09,751 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:09,753 INFO sqlalchemy.engine.Engine DELETE FROM users WHERE use
rs.name = %(name_1)s
2024-11-20 13:55:09,753 INFO sqlalchemy.engine.Engine [generated in 0.00072s] {'n
ame_1': 'xdot'}
2024-11-20 13:55:09,756 INFO sqlalchemy.engine.Engine COMMIT
```

In [57]:
```python
find_all()
```

```
2024-11-20 13:55:10,223 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:10,224 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
2024-11-20 13:55:10,225 INFO sqlalchemy.engine.Engine [cached since 10.22s ago]
{}
id: 1
name: Jack
id_address: ECNU
id_number: abcd
2024-11-20 13:55:10,227 INFO sqlalchemy.engine.Engine ROLLBACK
```

- 查找到数据后再删除

In [58]:
```python
# 删除操作
def deleteData():
    session = DbSession() # 创建会话

    delete_users = session.query(User).filter(User.id == "1").first()
    if delete_users:
        session.delete(delete_users)
        session.commit()

    session.close() # 关闭会话

deleteData()
```

```
2024-11-20 13:55:11,298 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:11,300 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
WHERE users.id = %(id_1)s
 LIMIT %(param_1)s
2024-11-20 13:55:11,300 INFO sqlalchemy.engine.Engine [generated in 0.00061s] {'i
d_1': '1', 'param_1': 1}
2024-11-20 13:55:11,303 INFO sqlalchemy.engine.Engine DELETE FROM users WHERE use
rs.id = %(id)s
2024-11-20 13:55:11,304 INFO sqlalchemy.engine.Engine [generated in 0.00047s] {'i
d': 1}
2024-11-20 13:55:11,305 INFO sqlalchemy.engine.Engine COMMIT
```

In [59]:
```python
find_all()
```

```
2024-11-20 13:55:12,126 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:12,127 INFO sqlalchemy.engine.Engine SELECT users.id AS users_i
d, users.name AS users_name, users.sex AS users_sex, users.nation AS users_natio
n, users.birth AS users_birth, users.id_address AS users_id_address, users.id_num
ber AS users_id_number, users.creater AS users_creater, users.create_time AS user
s_create_time, users.updater AS users_updater, users.update_time AS users_update_
time, users.comment AS users_comment
FROM users
2024-11-20 13:55:12,128 INFO sqlalchemy.engine.Engine [cached since 12.13s ago]
{}
2024-11-20 13:55:12,130 INFO sqlalchemy.engine.Engine ROLLBACK
```

数据都成功删除

2.2.8 删除表格

In [60]:
```python
from sqlalchemy import text
def dropTable():
    with engine.connect() as connection:
        sql = text('DROP TABLE IF EXISTS users;')
        result = connection.execute(sql)
        connection.commit()  # 确保更改被提交

dropTable()
```

```
2024-11-20 13:55:13,710 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:13,711 INFO sqlalchemy.engine.Engine DROP TABLE IF EXISTS users;
2024-11-20 13:55:13,711 INFO sqlalchemy.engine.Engine [generated in 0.00183s] {}
2024-11-20 13:55:13,716 INFO sqlalchemy.engine.Engine COMMIT
```

删除所有表

In [61]:
```python
# all tables are deleted
Base.metadata.drop_all(engine)
```

```
2024-11-20 13:55:15,233 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 13:55:15,234 INFO sqlalchemy.engine.Engine SELECT pg_catalog.pg_class.
relname
FROM pg_catalog.pg_class JOIN pg_catalog.pg_namespace ON pg_catalog.pg_namespace.
oid = pg_catalog.pg_class.relnamespace
WHERE pg_catalog.pg_class.relname = %(table_name)s AND pg_catalog.pg_class.relkin
d = ANY (ARRAY[%(param_1)s, %(param_2)s, %(param_3)s, %(param_4)s, %(param_5)s])
AND pg_catalog.pg_table_is_visible(pg_catalog.pg_class.oid) AND pg_catalog.pg_nam
espace.nspname != %(nspname_1)s
2024-11-20 13:55:15,235 INFO sqlalchemy.engine.Engine [cached since 22.58s ago]
{'table_name': 'users', 'param_1': 'r', 'param_2': 'p', 'param_3': 'f', 'param_
4': 'v', 'param_5': 'm', 'nspname_1': 'pg_catalog'}
2024-11-20 13:55:15,237 INFO sqlalchemy.engine.Engine COMMIT
```

3. ORM练习

注意：本次练习中的数据为3.4中数据一次插入的结果。如果不小心多次插入，可使用删除数据或者删除表，再重新插入。

3.1 建立连接

In [1]:
```
from sqlalchemy import create_engine, Column, String, Integer, Date, ForeignKey
engine = create_engine("postgresql://ecnu10222140402:ECNU10222140402@pgm-uf6t802
    echo=True,
    pool_size=8,
    pool_recycle=60*30
)
```

3.2 建立会话

In [2]:
```
#建立会话
from sqlalchemy.orm import sessionmaker
# 创建session
DbSession = sessionmaker(bind=engine)
session = DbSession()
```

3.3 表格创建

四个表格分别是 student, course, teacher, score.

```
create table student(
s_id varchar(10),
s_name varchar(20),
s_age date,
s_sex varchar(10)
);

create table course(
c_id varchar(10),
c_name varchar(20),
t_id varchar(10)
);

create table teacher (
t_id varchar(10),
t_name varchar(20)
);

create table score (
s_id varchar(10),
c_id varchar(10),
score integer );
```

In [3]:
```
from sqlalchemy.orm import declarative_base
from sqlalchemy.orm import sessionmaker, relationship

# 创建对象的基类:
Base = declarative_base()

# 定义 Student 表
class Student(Base):
    __tablename__ = 'student'
    s_id = Column(String(10), primary_key=True)  # 学生 ID
    s_name = Column(String(20), nullable=False)  # 学生姓名
    s_age = Column(Date, nullable=False)        # 出生日期
    s_sex = Column(String(10), nullable=False)  # 性别

# 定义 Course 表
class Course(Base):
    __tablename__ = 'course'
    c_id = Column(String(10), primary_key=True)  # 课程 ID
    c_name = Column(String(20), nullable=False) # 课程名称
    t_id = Column(String(10), ForeignKey('teacher.t_id'))  # 外键关联到 Teacher

    teacher = relationship("Teacher", back_populates="courses")  # 关联 Teacher

# 定义 Teacher 表
class Teacher(Base):
    __tablename__ = 'teacher'
    t_id = Column(String(10), primary_key=True)  # 教师 ID
    t_name = Column(String(20), nullable=False) # 教师姓名

    courses = relationship("Course", back_populates="teacher")  # 关联 Course

# 定义 Score 表
class Score(Base):
    __tablename__ = 'score'
    s_id = Column(String(10), ForeignKey('student.s_id'), primary_key=True)  #
    c_id = Column(String(10), ForeignKey('course.c_id'), primary_key=True)   #
    score = Column(Integer, nullable=False)                                  #

    student = relationship("Student", back_populates="scores")  # 关联 Student
    course = relationship("Course", back_populates="scores")    # 关联 Course

Student.scores = relationship("Score", back_populates="student")  # 在 Student
Course.scores = relationship("Score", back_populates="course")    # 在 Course 中

# 创建数据库连接
engine = create_engine('sqlite:///school.db', echo=True)  # 替换为你的数据库连接
Base.metadata.create_all(engine)  # 创建所有表格

# 创建会话
Session = sessionmaker(bind=engine)
session = Session()

# 输出提示
print("表格已创建并连接成功！")
```

```
2024-11-20 23:52:44,007 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 23:52:44,008 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("stu
dent")
2024-11-20 23:52:44,008 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-11-20 23:52:44,082 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("cou
rse")
2024-11-20 23:52:44,083 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-11-20 23:52:44,097 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("tea
cher")
2024-11-20 23:52:44,098 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-11-20 23:52:44,113 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("sco
re")
2024-11-20 23:52:44,114 INFO sqlalchemy.engine.Engine [raw sql] ()
2024-11-20 23:52:44,117 INFO sqlalchemy.engine.Engine COMMIT
表格已创建并连接成功！
```

3.4 插入数据

```
insert into student (s_id, s_name, s_age, s_sex)
values
('01' , '赵雷' , '1990-01-01' , '男'),
('02' , '钱电' , '1990-12-21' , '男'),
('03' , '孙风' , '1990-05-20' , '男'),
('04' , '李云' , '1990-08-06' , '男'),
('05' , '周梅' , '1991-12-01' , '女'),
('06' , '吴兰' , '1992-03-01' , '女'),
('07' , '郑竹' , '1989-07-01' , '女'),
('08' , '王菊' , '1990-01-20' , '女');

insert into course (c_id, c_name, t_id)
values
('01' , '语文' , '02'),
('02' , '数学' , '01'),
('03' , '英语' , '03');

insert into teacher (t_id, t_name)
values
('01' , '张三'),
('02' , '李四'),
('03' , '王五');

insert into score (s_id, c_id, score)
values
('01' , '01' , 80),
('01' , '02' , 90),
('01' , '03' , 99),
('02' , '01' , 70),
('02' , '02' , 60),
('02' , '03' , 80),
('03' , '01' , 80),
('03' , '02' , 80),
('03' , '03' , 80),
('04' , '01' , 50),
('04' , '02' , 30),
('04' , '03' , 20),
('05' , '01' , 76),
('05' , '02' , 87),
('06' , '01' , 31),
('06' , '03' , 34),
('07' , '02' , 89),
('07' , '03' , 98);
```

思考：怎样写代码可以批量插入数据

In [4]:
```
#回滚当前事务
session.rollback()
# 删除所有数据
session.query(Student).delete()
session.query(Course).delete()
session.query(Teacher).delete()
session.query(Score).delete()
```

```
# 提交删除操作
session.commit()
```

```
2024-11-20 23:52:46,055 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 23:52:46,057 INFO sqlalchemy.engine.Engine DELETE FROM student
2024-11-20 23:52:46,057 INFO sqlalchemy.engine.Engine [generated in 0.00060s] ()
2024-11-20 23:52:46,066 INFO sqlalchemy.engine.Engine DELETE FROM course
2024-11-20 23:52:46,067 INFO sqlalchemy.engine.Engine [generated in 0.00053s] ()
2024-11-20 23:52:46,075 INFO sqlalchemy.engine.Engine DELETE FROM teacher
2024-11-20 23:52:46,075 INFO sqlalchemy.engine.Engine [generated in 0.00064s] ()
2024-11-20 23:52:46,076 INFO sqlalchemy.engine.Engine DELETE FROM score
2024-11-20 23:52:46,077 INFO sqlalchemy.engine.Engine [generated in 0.00078s] ()
2024-11-20 23:52:46,078 INFO sqlalchemy.engine.Engine COMMIT
```

In [5]:
```
from sqlalchemy.orm import sessionmaker
from datetime import datetime

# 准备插入的数据
students = [
    Student(s_id='01', s_name='赵雷', s_age=datetime.strptime('1990-01-01', '%Y-
    Student(s_id='02', s_name='钱电', s_age=datetime.strptime('1990-12-21', '%Y-
    Student(s_id='03', s_name='孙风', s_age=datetime.strptime('1990-05-20', '%Y-
    Student(s_id='04', s_name='李云', s_age=datetime.strptime('1990-08-06', '%Y-
    Student(s_id='05', s_name='周梅', s_age=datetime.strptime('1991-12-01', '%Y-
    Student(s_id='06', s_name='吴兰', s_age=datetime.strptime('1992-03-01', '%Y-
    Student(s_id='07', s_name='郑竹', s_age=datetime.strptime('1989-07-01', '%Y-
    Student(s_id='08', s_name='王菊', s_age=datetime.strptime('1990-01-20', '%Y-
]

# 批量插入数据
try:
    session.add_all(students)  # 将所有学生对象添加到session
    session.commit()  # 提交事务，保存数据到数据库
    print("数据插入成功！")
except Exception as e:
    session.rollback()  # 如果发生错误，回滚事务
    print(f"插入数据时发生错误：{e}")
```

```
2024-11-20 23:52:47,026 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 23:52:47,029 INFO sqlalchemy.engine.Engine INSERT INTO student (s_id,
s_name, s_age, s_sex) VALUES (?, ?, ?, ?)
2024-11-20 23:52:47,029 INFO sqlalchemy.engine.Engine [generated in 0.00066s]
[('01', '赵雷', '1990-01-01', '男'), ('02', '钱电', '1990-12-21', '男'), ('03',
'孙风', '1990-05-20', '男'), ('04', '李云', '1990-08-06', '男'), ('05', '周梅',
'1991-12-01', '女'), ('06', '吴兰', '1992-03-01', '女'), ('07', '郑竹', '1989-07-
01', '女'), ('08', '王菊', '1990-01-20', '女')]
2024-11-20 23:52:47,038 INFO sqlalchemy.engine.Engine COMMIT
数据插入成功！
```

In [6]:
```
# 批量插入 Course 数据
courses = [
    Course(c_id='01', c_name='语文', t_id='02'),
    Course(c_id='02', c_name='数学', t_id='01'),
    Course(c_id='03', c_name='英语', t_id='03'),
]

session.bulk_save_objects(courses)  # 高效批量插入
session.commit()                    # 提交事务
```

```
2024-11-20 23:52:47,971 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 23:52:47,972 INFO sqlalchemy.engine.Engine INSERT INTO course (c_id, c
_name, t_id) VALUES (?, ?, ?)
2024-11-20 23:52:47,973 INFO sqlalchemy.engine.Engine [generated in 0.00078s]
[('01', '语文', '02'), ('02', '数学', '01'), ('03', '英语', '03')]
2024-11-20 23:52:47,982 INFO sqlalchemy.engine.Engine COMMIT
```

In [7]:
```
# 插入 Teacher 表数据
teachers = [
    Teacher(t_id='01', t_name='张三'),
    Teacher(t_id='02', t_name='李四'),
    Teacher(t_id='03', t_name='王五'),
]
session.add_all(teachers)
session.commit()
```

```
2024-11-20 23:52:48,527 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 23:52:48,529 INFO sqlalchemy.engine.Engine INSERT INTO teacher (t_id,
t_name) VALUES (?, ?)
2024-11-20 23:52:48,529 INFO sqlalchemy.engine.Engine [generated in 0.00064s]
[('01', '张三'), ('02', '李四'), ('03', '王五')]
2024-11-20 23:52:48,541 INFO sqlalchemy.engine.Engine COMMIT
```

In [8]:
```
# 插入 Score 表数据
try:
    scores = [
        Score(s_id='01', c_id='01', score=80),
        Score(s_id='01', c_id='02', score=90),
        Score(s_id='01', c_id='03', score=99),
        Score(s_id='02', c_id='01', score=70),
        Score(s_id='02', c_id='02', score=60),
        Score(s_id='02', c_id='03', score=80),
        Score(s_id='03', c_id='01', score=80),
        Score(s_id='03', c_id='02', score=80),
        Score(s_id='03', c_id='03', score=80),
        Score(s_id='04', c_id='01', score=50),
        Score(s_id='04', c_id='02', score=30),
        Score(s_id='04', c_id='03', score=20),
        Score(s_id='05', c_id='01', score=76),
        Score(s_id='05', c_id='02', score=87),
        Score(s_id='06', c_id='01', score=31),
        Score(s_id='06', c_id='03', score=34),
        Score(s_id='07', c_id='02', score=89),
        Score(s_id='07', c_id='03', score=98),
    ]
    session.add_all(scores)
    session.commit()
    print("成绩数据插入完成！")
except Exception as e:
    session.rollback()
    print(f"插入成绩数据失败：{e}")
```

```
2024-11-20 23:52:49,106 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-20 23:52:49,110 INFO sqlalchemy.engine.Engine INSERT INTO score (s_id, c_
id, score) VALUES (?, ?, ?)
2024-11-20 23:52:49,110 INFO sqlalchemy.engine.Engine [generated in 0.00055s]
[('01', '01', 80), ('01', '02', 90), ('01', '03', 99), ('02', '01', 70), ('02',
'02', 60), ('02', '03', 80), ('03', '01', 80), ('03', '02', 80)  ... displaying 1
0 of 18 total bound parameter sets ...  ('07', '02', 89), ('07', '03', 98)]
2024-11-20 23:52:49,124 INFO sqlalchemy.engine.Engine COMMIT
成绩数据插入完成！
```

怎样写代码可以批量插入数据?

- session.add_all()方法：可以将多个对象添加到会话中，并一次性提交。这种方法适合插入较小的数据集。通过将多个对象封装到列表中，调用add_all()可以避免多次提交操作，从而提高效率。
- bulk_insert_mappings()方法：专门针对批量插入设计的高效方法。与session.add_all() 不同，bulk_insert_mappings()允许直接插入字典数据而不需要先创建ORM实例。这可以大大提升插入速度，尤其适合大批量数据插入。

3.5 习题

提示：如果写的代码运行出现断开连接，可以通过下面的语句重新连接。（由于代码的不正确导致的）

In [ ]:
```python
session = DbSession()
```

1.查询学生中的所有女生，并将名字按降序排序

In [11]:
```python
from sqlalchemy import create_engine, func, and_, or_, extract, desc

#1.查询学生中的所有女生，并将名字按降序排序
print("1. 查询所有女生并按名字降序排序：")
girls = session.query(Student).filter(Student.s_sex == '女').order_by(desc(Stude
for girl in girls:
    print(f"ID: {girl.s_id}, Name: {girl.s_name}, Age: {girl.s_age}, Sex: {girl.
```

```
1. 查询所有女生并按名字降序排序：
2024-11-21 00:04:06,363 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2024-11-21 00:04:06,364 INFO sqlalchemy.engine.Engine SELECT student.s_id AS stud
ent_s_id, student.s_name AS student_s_name, student.s_age AS student_s_age, stude
nt.s_sex AS student_s_sex
FROM student
WHERE student.s_sex = ? ORDER BY student.s_name DESC
2024-11-21 00:04:06,365 INFO sqlalchemy.engine.Engine [cached since 157.8s ago]
('女',)
ID: 07, Name: 郑竹, Age: 1989-07-01, Sex: 女
ID: 08, Name: 王菊, Age: 1990-01-20, Sex: 女
ID: 05, Name: 周梅, Age: 1991-12-01, Sex: 女
ID: 06, Name: 吴兰, Age: 1992-03-01, Sex: 女
```

2.查询" 01 "课程中成绩最高的5位同学的id和成绩

In [13]:
```python
# 2.查询"01"课程中成绩最高的5位同学的id和成绩
print("\n2.查询'01'课程成绩最高的5位同学：")
top_scores = (
    session.query(Score.s_id, Score.score)
    .filter(Score.c_id == '01')
    .order_by(desc(Score.score))
    .limit(5)
    .all()
)
for s_id, score in top_scores:
    print(f"Student ID: {s_id}, Score: {score}")
```

```
2.查询'01'课程成绩最高的5位同学:
2024-11-21 00:04:41,770 INFO sqlalchemy.engine.Engine SELECT score.s_id AS score_
s_id, score.score AS score_score
FROM score
WHERE score.c_id = ? ORDER BY score.score DESC
 LIMIT ? OFFSET ?
2024-11-21 00:04:41,771 INFO sqlalchemy.engine.Engine [cached since 193.2s ago]
('01', 5, 0)
Student ID: 01, Score: 80
Student ID: 03, Score: 80
Student ID: 05, Score: 76
Student ID: 02, Score: 70
Student ID: 04, Score: 50
```

3.查询出生年份在1990年的同学(注意：s_age的类型为date)

In [14]:
```
#3.查询出生年份在1990年的同学
print("\n3. 查询出生年份在 1990 年的同学: ")
born_in_1990 = session.query(Student).filter(extract('year', Student.s_age) == 1
for student in born_in_1990:
    print(f"ID: {student.s_id}, Name: {student.s_name}, Age: {student.s_age}")
```

```
3. 查询出生年份在 1990 年的同学:
2024-11-21 00:05:20,194 INFO sqlalchemy.engine.Engine SELECT student.s_id AS stud
ent_s_id, student.s_name AS student_s_name, student.s_age AS student_s_age, stude
nt.s_sex AS student_s_sex
FROM student
WHERE CAST(STRFTIME('%Y', student.s_age) AS INTEGER) = ?
2024-11-21 00:05:20,195 INFO sqlalchemy.engine.Engine [cached since 231.6s ago]
(1990,)
ID: 01, Name: 赵雷, Age: 1990-01-01
ID: 02, Name: 钱电, Age: 1990-12-21
ID: 03, Name: 孙风, Age: 1990-05-20
ID: 04, Name: 李云, Age: 1990-08-06
ID: 08, Name: 王菊, Age: 1990-01-20
```

4.查询每位同学一共选择了几门课和总成绩

In [15]:
```
#4.查询每位同学一共选择了几门课和总成绩
print("\n4. 查询每位同学选择的课程数和总成绩: ")
student_totals = (
    session.query(
        Student.s_id,
        func.count(Score.c_id).label('course_count'),
        func.sum(Score.score).label('total_score'),
    )
    .join(Score, Student.s_id == Score.s_id)
    .group_by(Student.s_id)
    .all()
)
for s_id, course_count, total_score in student_totals:
    print(f"Student ID: {s_id}, Course Count: {course_count}, Total Score: {tota
4. 查询每位同学选择的课程数和总成绩：
2024-11-21 00:05:23,691 INFO sqlalchemy.engine.Engine SELECT student.s_id AS stud
ent_s_id, count(score.c_id) AS course_count, sum(score.score) AS total_score
FROM student JOIN score ON student.s_id = score.s_id GROUP BY student.s_id
2024-11-21 00:05:23,692 INFO sqlalchemy.engine.Engine [cached since 235.1s ago]
()
Student ID: 01, Course Count: 3, Total Score: 269
Student ID: 02, Course Count: 3, Total Score: 210
Student ID: 03, Course Count: 3, Total Score: 240
Student ID: 04, Course Count: 3, Total Score: 100
Student ID: 05, Course Count: 2, Total Score: 163
Student ID: 06, Course Count: 2, Total Score: 65
Student ID: 07, Course Count: 2, Total Score: 187
```

5.查询01课程或02课程成绩大于85的同学id

In [16]:

```python
#5.查询01课程或02课程成绩大于85的同学id
print("\n5.查询01课程或02课程成绩大于85的同学：")
high_scores = (
    session.query(Score.s_id)
    .filter(
        or_(
            and_(Score.c_id == '01', Score.score > 85),
            and_(Score.c_id == '02', Score.score > 85),
        )
    )
    .distinct()
    .all()
)
for s_id, in high_scores:
    print(f"Student ID: {s_id}")

# 关闭会话
session.close()
```

```
5. 查询 01 课程或 02 课程成绩大于 85 的同学：
2024-11-21 00:05:27,568 INFO sqlalchemy.engine.Engine SELECT DISTINCT score.s_id
AS score_s_id
FROM score
WHERE score.c_id = ? AND score.score > ? OR score.c_id = ? AND score.score > ?
2024-11-21 00:05:27,569 INFO sqlalchemy.engine.Engine [cached since 238.9s ago]
('01', 85, '02', 85)
Student ID: 01
Student ID: 05
Student ID: 07
2024-11-21 00:05:27,573 INFO sqlalchemy.engine.Engine ROLLBACK
```

In [ ]: