**ORM关系之一对一：**

在sqlalchemy中，如果想要将两个模型映射成一对一的关系，那么应该在父模型中，指定引用的时候，要传递一个`uselist=False`这个参数进去。就是告诉父模型，以后引用这个从模型的时候，不再是一个列表了，而是一个对象了。示例代码如下：

**方式1：**

|  |
| --- |
| class User(Base):  \_\_tablename\_\_ = 'user'  id = Column(Integer,primary\_key=True,autoincrement=True)  uname = Column(String(50),nullable=False)  extend = relationship("UserExtend",uselist=False)  class UserExtend(Base):  \_\_tablename\_\_ = 'user\_extend'  id = Column(Integer, primary\_key=True, autoincrement=True)  school = Column(String(50))  uid = Column(Integer,ForeignKey("user.id"))  user = relationship("User") |

**方式2：（用得较多）**

当然，也可以借助`sqlalchemy.orm.backref`来简化代码：

|  |
| --- |
| class User(Base):  \_\_tablename\_\_ = 'user'  id = Column(Integer,primary\_key=True,autoincrement=True)  uname = Column(String(50),nullable=False)  class UserExtend(Base):  \_\_tablename\_\_ = 'user\_extend'  id = Column(Integer, primary\_key=True, autoincrement=True)  school = Column(String(50))  uid = Column(Integer,ForeignKey("user.id"))  user = relationship("User",backref=backref("extend",uselist=False)) |

**演示代码如下：**

|  |
| --- |
| **from** sqlalchemy **import** create\_engine,Column,Integer,String,Float,DECIMAL,Boolean,Enum,Date,DateTime,Time,Text  **from** sqlalchemy **import** func,and\_,or\_,ForeignKey  **from** sqlalchemy.ext.declarative **import** declarative\_base  **from** sqlalchemy.dialects.mysql **import** LONGTEXT  **from** sqlalchemy.orm **import** sessionmaker,relationship,backref  **from** datetime **import** date,datetime,time  #在python 3.x中 有enum模块  **import** enum  **import** random  #准备连接数据库基本信息  HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'first\_sqlalchemy'**  USERNAME = **'root'**  PASSWORD = **'root'**  #dialect+driver://username:password@host:port/database?charset=utf8  #按照上述的格式来 组织数据库信息  DB\_URI =**"mysql+pymysql://{username}:{password}@{host}:{port}/{db}?charset=utf8"**.format(username=USERNAME,password=PASSWORD,host=HOSTNAME,port=PORT,db=DATABASE)  #创建数据库引擎  engine = create\_engine(DB\_URI)  Base = declarative\_base(engine)  session = sessionmaker(engine)()  # 主表 / 从表  # user/news 1：n  # user/user\_extend 1：1  #表1  **class** User(Base):  \_\_tablename\_\_ = **'user'**  id = Column(Integer,primary\_key=**True**,autoincrement=**True**)  uname = Column(String(50),nullable=**False**)  #添加属性 优化2表查询操作  # newss =relationship("News") #这种写法不是最优的，通常会把它通过反向声明的方式写在“多”的那一方  #1:1关系的表示方式1  # extend =relationship("UserExtend",uselist=False)  **def** \_\_repr\_\_(self):  **return "<User(uname:%s)>"** % self.uname  #表3  **class** UserExtend(Base):  \_\_tablename\_\_ = **'user\_extend'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  school = Column(String(50))  #外键  uid = Column(Integer,ForeignKey(**"user.id"**))  #1:1关系的表示方式1  # user = relationship("User")  # 1:1关系的表示方式2  user = relationship(**"User"**,backref = backref(**"extend"**,uselist=**False**))  #表2  **class** News(Base):  \_\_tablename\_\_ = **'news'**  id = Column(Integer,primary\_key=**True**,autoincrement=**True**)  title = Column(String(50),nullable=**False**)  content = Column(Text,nullable=**False**)  #SQLALchemy实现外键的方法  uid = Column(Integer,ForeignKey(**"user.id"**)) #默认删除策略为 ：RESTRICT  #添加属性 优化2表查询操作  #正向  # author = relationship("User")  #最终：会把正向 和反向 关系 写在一起  author = relationship(**"User"**,backref=**"newss"**)  **def** \_\_repr\_\_(self):  **return "<News(title:%s,content=%s)>"** % (self.title,self.content)  #创建表  # Base.metadata.drop\_all()  # Base.metadata.create\_all()  #需求：ORM层面外键 和一对一关系实现  #好处1：添加数据 User 添加 UserExtend  # user = User(uname="wangwu")  # ux = UserExtend(school="京南大学")  # user.extend = ux  # # print(type(user.extend))  # session.add(user)  # session.commit()  #好处1：添加数据 UserExtend 添加 User  # ux = UserExtend(school="武汉大学")  # user2 = User(uname="李四")  # ux.user = user2  # print(type(ux.user))  # session.add(ux)  # session.commit()  #好处2：查询数据  user3 = session.query(User).first()  print(user3.uname)  print(user3.extend.school) |