**ORM层面的relationship方法中cascade1：**

在SQLAlchemy，只要将一个数据添加到session中，和他相关联的数据都可以一起存入到数据库中了。

这些是怎么设置的呢？其实是通过relationship的时候，有一个关键字参数cascade可以设置这些属性，

cascade属性值为：

**save-update**：默认选项。在添加一条数据的时候，会把其他和他相关联的数据都添加到数据库中。这种行为就是save-update属性影响的。

**delete**：表示当删除某一个模型中的数据的时候，是否也删掉使用relationship和他关联的数据。

**delete-orphan**：表示当对一个ORM对象解除了父表中的关联对象的时候，自己便会被删除掉。当然如果父表中的数据被删除，自己也会被删除。这个选项只能用在一对多上，并且还需要在子模型中的relationship中，增加一个single\_parent=True的参数。

**merge**：默认选项。当在使用session.merge，合并一个对象的时候，会将使用了relationship相关联的对象也进行merge操作。

**expunge**：移除操作的时候，会将相关联的对象也进行移除。这个操作只是从session中移除，并不会真正的从数据库中删除。

**all**：是对save-update, merge, refresh-expire, expunge, delete几种的缩写。

**代码演示：**

|  |
| --- |
| **from** sqlalchemy **import** create\_engine,Column,Integer,Float,Boolean,DECIMAL,Enum,\  Date,DateTime,Time,String,Text,func,or\_,and\_,ForeignKey,Table  **from** sqlalchemy.dialects.mysql **import** LONGTEXT  **from** sqlalchemy.ext.declarative **import** declarative\_base  **from** sqlalchemy.orm **import** sessionmaker,relationship,backref  **import** random  HOSTNAME = **'127.0.0.1'**  PORT = **'3306'**  DATABASE = **'first\_sqlalchemy'**  USERNAME = **'root'**  PASSWORD = **'root'**  DB\_URI =**"mysql+pymysql://{username}:{password}@{host}:{port}/{db}?charset=utf8"**.format(username=USERNAME,password=PASSWORD,host=HOSTNAME,port=PORT,db=DATABASE)  engine = create\_engine(DB\_URI)  Base = declarative\_base(engine)  session = sessionmaker(engine)()  **class** User(Base):  \_\_tablename\_\_ = **'user'**  id = Column(Integer,primary\_key=**True**,autoincrement=**True**)  uname = Column(String(50),nullable=**False**)  # articles =relationship("Article",cascade="save-update,delete") #放入Article中去优化  # comments = relationship("Comment") #放入Comment中去优化  **class** Article(Base):  \_\_tablename\_\_ = **'article'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  title = Column(String(50), nullable=**False**)  uid = Column(Integer,ForeignKey(**"user.id"**),nullable=**False**)  # author = relationship("User",backref="articles") #cascade默认为save-update  # author = relationship("User",backref="articles",cascade="save-update") #明文指定为save-update  #delete：表示当删除某一个模型中的数据的时候，是否也删掉使用relationship和他关联的数据。默认不删除关联数据  # author = relationship("User",cascade="save-update,delete") #明文指定为save-update 和delete  #优化写法  author = relationship(**"User"**,backref=backref(**"articles"**,cascade=**"save-update,delete"**),cascade=**"save-update,delete"**)  **class** Comment(Base):  \_\_tablename\_\_ = **'comment'**  id = Column(Integer, primary\_key=**True**, autoincrement=**True**)  content = Column(Text,nullable=**False**)  uid = Column(Integer,ForeignKey(**"user.id"**))  #author = relationship(**"User"**)  # 优化写法  author = relationship(**"User"**,backref=backref(**"comments"**))  **def** add\_data():  Base.metadata.drop\_all()  Base.metadata.create\_all()  user = User(uname=**"momo"**)  article = Article(title=**"华为5G"**)  article.author = user  session.add(article)  #引入comment表  comment = Comment(content=**'你少说风凉话'**)  comment.author = user  session.add(comment)  session.commit()  **def** oper\_data():  #relationship里边的cascade 可通过Article影响User  # article = session.query(Article).first()  # session.delete(article)  # session.commit()  #反过来通过User也能影响Article  user = session.query(User).first()  session.delete(user)  session.commit()  #总结1：relationship里边的cascade 可通过Article影响User，反过来通过User也能影响Article  #总结2：relationship里边的cascade 只会影响当前类User中relationship的模型Article，不会影响模型Comment  **if** \_\_name\_\_ == **'\_\_main\_\_'**:  # add\_data()  oper\_data()  #ORM层面的relationship方法中cascade使用1 |