ALTKOM AKADEMIA
Zobaczyć inaczej

SQL Alchemy

How to work with database in Python

SQL Alchemy

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///:memory:', echo=True)
from sqlalchemy import Column, Integer, String

from sqlalchemy.ext.declarative import declarative_base
Base = declarative_base()
```

SQL Alchemy

```
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String)
    fullname = Column(String)
    nickname = Column(String)

    def __repr__(self):
       return "<User(name='%s', fullname='%s', nickname='%s')>" % (
                            self.name, self.fullname, self.nickname)

Base.metadata.create_all(engine)
```

SQL Alchemy

```
from sqlalchemy.orm import sessionmaker
Session = sessionmaker(bind=engine)
session = Session()

session.dirty # gdy coś się zmieni
session.new # nowe wpisy
session.commit() # wysyła do bazy danych
session.rollback() # rollback z obecnej sesji
```

SQL Alchemy

```
#Add some data:
ed_user = User(name='ed', fullname='Ed Jones',
nickname='edsnickname')
session.add(ed_user)
session.commit()
```

SQL Alchemy

```
for instance in session.query(User).order_by(User.id):
    print(instance.name, instance.fullname)

for name, fullname in session.query(User.name, User.fullname):
    print(name, fullname)
```

SQL Alchemy

```
session.query(User).order_by(User.id)[1:3]
session.query(User.name).filter_by(fullname='Ed Jones')
session.query(User).filter(User.name=='ed').filter(User.fullname=='Ed Jones')

uery.filter(User.name == 'ed')
query.filter(User.name != 'ed')
query.filter(User.name.like('%ed%'))
query.filter(User.name.in_(['ed', 'wendy', 'jack']))
query.filter(~User.name.in_(['ed', 'wendy', 'jack']))
query.filter(User.name != None)      /    query.filter(User.name.isnot(None))
```

SQL Alchemy

AND
```
from sqlalchemy import and_
query.filter(and_(User.name == 'ed', User.fullname == 'Ed Jones'))
query.filter(User.name == 'ed', User.fullname == 'Ed Jones')
query.filter(User.name == 'ed').filter(User.fullname == 'Ed Jones')
```

OR
```
from sqlalchemy import or_
query.filter(or_(User.name == 'ed', User.name == 'wendy'))
```

MATCH
```
query.filter(User.name.match('wendy'))
```

SQL Alchemy

```
query.all()
query.first()
query.one()
query.one_or_none()
```

SQL Alchemy

```
from sqlalchemy import text
session.query(User).filter(text("id<224")).order_by
(text("id")).all()
```

SQL Alchemy

```
session.query(User).filter(User.name.like('%ed')).count()

 from sqlalchemy import func
session.query(func.count(User.name), User.name).group_by(User.name).all()
```

SQL Alchemy

```
from sqlalchemy import ForeignKey
from sqlalchemy.orm import relationship

class Address(Base):
    __tablename__ = 'addresses'
    id = Column(Integer, primary_key=True)
    email_address = Column(String, nullable=False)
    user_id = Column(Integer, ForeignKey('users.id'))
    user = relationship("User", back_populates="addresses")

    def __repr__(self):
        return "<Address(email_address='%s')>" % self.email_address

User.addresses = relationship("Address", back_populates='user',
...                     cascade="all, delete, delete-orphan")
```

SQL Alchemy

```
session.query(User).join(Address).filter(Address.email_address=='jack@google.com').all()

query.join(Address, User.id==Address.user_id)    # explicit condition
query.join(User.addresses)                       # specify relationship from left to right
query.join(Address, User.addresses)    # same, with explicit target
query.join('addresses')                          # same, using a string

query.outerjoin(User.addresses)   # LEFT OUTER JOIN
```

SQL Alchemy

```
from sqlalchemy.sql import exists
stmt = exists().where(Address.user_id==User.id)

for name, in session.query(User.name).filter(stmt):
    print(name)
```

SQL Alchemy

```
session.delete(jack)
```

SQL Alchemy

```
query.filter(Address.user == someuser)
query.filter(Address.user != someuser)
query.filter(Address.user == None)
query.filter(User.addresses.contains(someaddress))
query.filter(User.addresses.any(Address.email_address == 'bar'))
query.filter(User.addresses.any(email_address='bar'))
query.filter(Address.user.has(name='ed'))
session.query(Address).with_parent(someuser, 'addresses')
```