Tecnológico Nacional de México campus Colima

Maestría en Sistemas Computacionales

Tecnologías de programación

Introducción a SQLAlchemy

D. en C. Patricia Elizabeth Figueroa Millán

Angel Primitivo Vejar Cortés | G2146001

Villa de Álvarez, Colima - 15 de diciembre de 2022

Objetivo

Que el estudiante utilice la librería SQLAlchemy para realizar operaciones CRUD en una base de datos, esto a través de su aplicación como ORM (Object Relational Mapper) para Python.

Metodología

Para el desarrollo de la actividad se utilizaron los siguientes pasos:

- Se realizó la instalación de la librería SQLAlchemy
- Se creó una base de datos sqlite almacenada en la memoria RAM
- Se siguieron los ejemplos de la documentación de la librería para realizar operaciones básicas en la base de datos
- Se crearon modelos para que el ORM pudiera realizar consultas a la base de datos
- Se ejecutaron distintas consultas con filtros y ordenamientos

Materiales

Para el desarrollo de la actividad se utilizaron los siguientes materiales:

- Computadora con acceso a internet
- Editor de texto (Visual Studio Code)
- Aplicación jupyter notebook

Desarrollo

Librería y conexión

In [ ]:
```python
import sqlalchemy

# engine = sqlalchemy.create_engine('sqlite:///data.db', echo=True) # echo=True prints out the SQL commands
engine = sqlalchemy.create_engine('sqlite:///:memory:', echo=True) # memory is for testing

# Declarative Base
```

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

# Data types
from sqlalchemy import Column, Integer, String
```

Definición de modelos

In [ ]:
```
# Model definition
class User(Base):
    __tablename__ = 'users'

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

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

Creación de tablas a partir de modelos

In [ ]:
```
# Table configuration
Base.metadata.create_all(engine)
```

```
2022-12-15 17:21:38,619 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-12-15 17:21:38,626 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("users")
2022-12-15 17:21:38,631 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:38,626 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("users")
2022-12-15 17:21:38,631 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:38,637 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("users")
2022-12-15 17:21:38,643 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:38,649 INFO sqlalchemy.engine.Engine
CREATE TABLE users (
        id INTEGER NOT NULL,
        name VARCHAR,
        fullname VARCHAR,
        password VARCHAR,
        PRIMARY KEY (id)
)


2022-12-15 17:21:38,651 INFO sqlalchemy.engine.Engine [no key 0.00249s] ()
2022-12-15 17:21:38,657 INFO sqlalchemy.engine.Engine COMMIT
```

Creación de sesión

In [ ]:
```python
from sqlalchemy.orm import sessionmaker

Session = sessionmaker(bind=engine)
session = Session()
```

Prueba de modelos

In [ ]:
```python
# Create a new user
ed_user = User(name='ed', fullname='Ed Jones', password='edspassword')
session.add(ed_user)
ed_user
```

Out[ ]:
```
<User(name='ed', fullname='Ed Jones', password='edspassword')>
```

In [ ]:
```python
user = session.query(User).filter_by(name='ed').first()
if user:
    print(user)
else:
    print('No user found')
```

```
2022-12-15 17:21:39,546 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-12-15 17:21:39,558 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:39,561 INFO sqlalchemy.engine.Engine [generated in 0.00391s] ('ed', 'Ed Jones', 'edspassword')
2022-12-15 17:21:39,558 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:39,561 INFO sqlalchemy.engine.Engine [generated in 0.00391s] ('ed', 'Ed Jones', 'edspassword')
2022-12-15 17:21:39,575 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name = ?
 LIMIT ? OFFSET ?
2022-12-15 17:21:39,581 INFO sqlalchemy.engine.Engine [generated in 0.00578s] ('ed', 1, 0)
<User(name='ed', fullname='Ed Jones', password='edspassword')>
```

In [ ]:
```python
# Multiple modifications in a single commit
session.add_all([
    User(name='wendy', fullname='Wendy Williams', password='foobar'),
    User(name='mary', fullname='Mary Contrary', password='xxg527'),
    User(name='fred', fullname='Fred Flinstone', password='blah')])

ed_user.password = 'f8s7ccs'
ed_user.fullname = 'Eduardo Jones'
```

In [ ]:
```python
session.commit()
```

```
2022-12-15 17:21:39,934 INFO sqlalchemy.engine.Engine UPDATE users SET fullname=?, password=? WHERE users.id = ?
2022-12-15 17:21:39,938 INFO sqlalchemy.engine.Engine [generated in 0.00431s] ('Eduardo Jones', 'f8s7ccs', 1)
2022-12-15 17:21:39,943 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:39,946 INFO sqlalchemy.engine.Engine [cached since 0.388s ago] ('wendy', 'Wendy Williams', 'foobar')
2022-12-15 17:21:39,938 INFO sqlalchemy.engine.Engine [generated in 0.00431s] ('Eduardo Jones', 'f8s7ccs', 1)
2022-12-15 17:21:39,943 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:39,946 INFO sqlalchemy.engine.Engine [cached since 0.388s ago] ('wendy', 'Wendy Williams', 'foobar')
2022-12-15 17:21:39,949 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:39,952 INFO sqlalchemy.engine.Engine [cached since 0.3943s ago] ('mary', 'Mary Contrary', 'xxg527')
2022-12-15 17:21:39,958 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:39,961 INFO sqlalchemy.engine.Engine [cached since 0.4037s ago] ('fred', 'Fred Flinstone', 'blah')
2022-12-15 17:21:39,965 INFO sqlalchemy.engine.Engine COMMIT
```

Retroceder cambios

In [ ]:
```python
# Cambios nuevos
ed_user.fullname = 'Ed Jones'

fake_user = User(name='fakeuser', fullname='Invalid', password='12345')
session.add(fake_user)
```

```
session.query(User).filter(User.name.in_(['ed', 'fakeuser'])).all()
```

```
2022-12-15 17:21:40,549 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-12-15 17:21:40,559 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.pass
word AS users_password
FROM users
WHERE users.id = ?
2022-12-15 17:21:40,563 INFO sqlalchemy.engine.Engine [generated in 0.00374s] (1,)
2022-12-15 17:21:40,570 INFO sqlalchemy.engine.Engine UPDATE users SET fullname=? WHERE users.id = ?
2022-12-15 17:21:40,574 INFO sqlalchemy.engine.Engine [generated in 0.00401s] ('Ed Jones', 1)
2022-12-15 17:21:40,578 INFO sqlalchemy.engine.Engine INSERT INTO users (name, fullname, password) VALUES (?, ?, ?)
2022-12-15 17:21:40,584 INFO sqlalchemy.engine.Engine [cached since 1.027s ago] ('fakeuser', 'Invalid', '12345')
2022-12-15 17:21:40,592 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name IN (?, ?)
2022-12-15 17:21:40,594 INFO sqlalchemy.engine.Engine [generated in 0.00312s] ('ed', 'fakeuser')
```

Out[ ]:
```
[<User(name='ed', fullname='Ed Jones', password='f8s7ccs')>,
 <User(name='fakeuser', fullname='Invalid', password='12345')>]
```

In [ ]:
```
# Rollback
session.rollback()
session.query(User).filter(User.name.in_(['ed', 'fakeuser'])).all()
```

```
2022-12-15 17:21:40,830 INFO sqlalchemy.engine.Engine ROLLBACK
2022-12-15 17:21:40,836 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-12-15 17:21:40,840 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name IN (?, ?)
2022-12-15 17:21:40,845 INFO sqlalchemy.engine.Engine [cached since 0.2537s ago] ('ed', 'fakeuser')
```

Out[ ]:
```
[<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>]
```

Consultas

In [ ]:
```
for instance in session.query(User).order_by(User.id):
    print(instance)
```

```
2022-12-15 17:21:40,980 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users ORDER BY users.id
2022-12-15 17:21:40,984 INFO sqlalchemy.engine.Engine [generated in 0.00446s] ()
2022-12-15 17:21:40,984 INFO sqlalchemy.engine.Engine [generated in 0.00446s] ()
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<User(name='wendy', fullname='Wendy Williams', password='foobar')>
<User(name='mary', fullname='Mary Contrary', password='xxg527')>
<User(name='fred', fullname='Fred Flinstone', password='blah')>
```

In [ ]:
```python
for name, fullname in session.query(User.name, User.fullname):
    print(name, fullname)
```

```
2022-12-15 17:21:41,100 INFO sqlalchemy.engine.Engine SELECT users.name AS users_name, users.fullname AS users_fullname
FROM users
2022-12-15 17:21:41,104 INFO sqlalchemy.engine.Engine [generated in 0.00390s] ()
ed Eduardo Jones
wendy Wendy Williams
mary Mary Contrary
fred Fred Flinstone
2022-12-15 17:21:41,104 INFO sqlalchemy.engine.Engine [generated in 0.00390s] ()
ed Eduardo Jones
wendy Wendy Williams
mary Mary Contrary
fred Fred Flinstone
```

Filtro

In [ ]:
```python
# equals
for user in session.query(User).filter(User.name=='ed'):
    print(user)
```

```
2022-12-15 17:21:41,275 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name = ?
2022-12-15 17:21:41,280 INFO sqlalchemy.engine.Engine [generated in 0.00489s] ('ed',)
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
2022-12-15 17:21:41,280 INFO sqlalchemy.engine.Engine [generated in 0.00489s] ('ed',)
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
```

In [ ]:
```python
# not equals
for user in session.query(User).filter(User.name!='ed'):
    print(user)
```

```
2022-12-15 17:21:41,507 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name != ?
2022-12-15 17:21:41,513 INFO sqlalchemy.engine.Engine [generated in 0.00567s] ('ed',)
<User(name='wendy', fullname='Wendy Williams', password='foobar')>
<User(name='mary', fullname='Mary Contrary', password='xxg527')>
<User(name='fred', fullname='Fred Flinstone', password='blah')>
2022-12-15 17:21:41,513 INFO sqlalchemy.engine.Engine [generated in 0.00567s] ('ed',)
<User(name='wendy', fullname='Wendy Williams', password='foobar')>
<User(name='mary', fullname='Mary Contrary', password='xxg527')>
<User(name='fred', fullname='Fred Flinstone', password='blah')>
```

In [ ]:
```python
# like
for user in session.query(User).filter(User.name.like('%ed%')):
    print(user)
```

```
2022-12-15 17:21:41,661 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name LIKE ?
2022-12-15 17:21:41,664 INFO sqlalchemy.engine.Engine [generated in 0.00295s] ('%ed%',)
2022-12-15 17:21:41,664 INFO sqlalchemy.engine.Engine [generated in 0.00295s] ('%ed%',)
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<User(name='fred', fullname='Fred Flinstone', password='blah')>
```

In [ ]:
```python
# in
for user in session.query(User).filter(User.name.in_(['ed', 'wendy', 'gil'])):
    print(user)
```

```
2022-12-15 17:21:41,807 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name IN (?, ?, ?)
2022-12-15 17:21:41,811 INFO sqlalchemy.engine.Engine [cached since 1.22s ago] ('ed', 'wendy', 'gil')
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<User(name='wendy', fullname='Wendy Williams', password='foobar')>
2022-12-15 17:21:41,811 INFO sqlalchemy.engine.Engine [cached since 1.22s ago] ('ed', 'wendy', 'gil')
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<User(name='wendy', fullname='Wendy Williams', password='foobar')>
```

In [ ]:
```python
# not in
for user in session.query(User).filter(~User.name.in_(['ed', 'wendy', 'gil'])):
    print(user)
```

```
2022-12-15 17:21:41,968 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE (users.name NOT IN (?, ?, ?))
2022-12-15 17:21:41,972 INFO sqlalchemy.engine.Engine [generated in 0.00369s] ('ed', 'wendy', 'gil')
<User(name='mary', fullname='Mary Contrary', password='xxg527')>
<User(name='fred', fullname='Fred Flinstone', password='blah')>
2022-12-15 17:21:41,972 INFO sqlalchemy.engine.Engine [generated in 0.00369s] ('ed', 'wendy', 'gil')
<User(name='mary', fullname='Mary Contrary', password='xxg527')>
<User(name='fred', fullname='Fred Flinstone', password='blah')>
```

In [ ]:
```
# And
for user in session.query(User).filter(User.name=='ed', User.fullname=='Eduardo Jones'):
    print(user)
```

```
2022-12-15 17:21:42,046 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name = ? AND users.fullname = ?
2022-12-15 17:21:42,049 INFO sqlalchemy.engine.Engine [generated in 0.00311s] ('ed', 'Eduardo Jones')
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
```

In [ ]:
```
# Or
for user in session.query(User).filter(sqlalchemy.or_(User.name=='ed', User.name=='wendy')):
    print(user)
```

```
2022-12-15 17:21:42,131 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password
FROM users
WHERE users.name = ? OR users.name = ?
2022-12-15 17:21:42,136 INFO sqlalchemy.engine.Engine [generated in 0.00445s] ('ed', 'wendy')
2022-12-15 17:21:42,136 INFO sqlalchemy.engine.Engine [generated in 0.00445s] ('ed', 'wendy')
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<User(name='wendy', fullname='Wendy Williams', password='foobar')>
```

Relaciones

In [ ]:
```
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
```

In [ ]:
```
User.addresses = relationship("Address", order_by=Address.id, back_populates="user")

Base.metadata.create_all(engine)
```

```
2022-12-15 17:21:42,285 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-12-15 17:21:42,288 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("users")
2022-12-15 17:21:42,292 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:42,296 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("addresses")
2022-12-15 17:21:42,298 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:42,302 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("addresses")
2022-12-15 17:21:42,304 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:42,288 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("users")
2022-12-15 17:21:42,292 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:42,296 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("addresses")
2022-12-15 17:21:42,298 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:42,302 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("addresses")
2022-12-15 17:21:42,304 INFO sqlalchemy.engine.Engine [raw sql] ()
2022-12-15 17:21:42,309 INFO sqlalchemy.engine.Engine
CREATE TABLE addresses (
        id INTEGER NOT NULL,
        email_address VARCHAR NOT NULL,
        user_id INTEGER,
        PRIMARY KEY (id),
        FOREIGN KEY(user_id) REFERENCES users (id)
)


2022-12-15 17:21:42,313 INFO sqlalchemy.engine.Engine [no key 0.00353s] ()
2022-12-15 17:21:42,318 INFO sqlalchemy.engine.Engine COMMIT
```

In [ ]:
```
# Add addresses to the user
ed_user.addresses = [Address(email_address="hola@col.com"), Address(email_address="rock@lml.com")]

session.add(ed_user)
session.commit()
```

```
2022-12-15 17:21:42,394 INFO sqlalchemy.engine.Engine SELECT addresses.id AS addresses_id, addresses.email_address AS a
ddresses_email_address, addresses.user_id AS addresses_user_id
FROM addresses
WHERE ? = addresses.user_id ORDER BY addresses.id
2022-12-15 17:21:42,397 INFO sqlalchemy.engine.Engine [generated in 0.00292s] (1,)
2022-12-15 17:21:42,397 INFO sqlalchemy.engine.Engine [generated in 0.00292s] (1,)
2022-12-15 17:21:42,406 INFO sqlalchemy.engine.Engine INSERT INTO addresses (email_address, user_id) VALUES (?, ?)
2022-12-15 17:21:42,409 INFO sqlalchemy.engine.Engine [generated in 0.00279s] ('hola@col.com', 1)
2022-12-15 17:21:42,412 INFO sqlalchemy.engine.Engine INSERT INTO addresses (email_address, user_id) VALUES (?, ?)
2022-12-15 17:21:42,415 INFO sqlalchemy.engine.Engine [cached since 0.00908s ago] ('rock@lml.com', 1)
2022-12-15 17:21:42,421 INFO sqlalchemy.engine.Engine COMMIT
```

In [ ]:

```python
# print all users and their addresses
for u, a in session.query(User, Address).filter(User.id==Address.user_id):
    print(u)
    print(a)
```

```
2022-12-15 17:21:42,548 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2022-12-15 17:21:42,552 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password, addresses.id AS addresses_id, addresses.email_address AS addr
esses_email_address, addresses.user_id AS addresses_user_id
FROM users, addresses
WHERE users.id = addresses.user_id
2022-12-15 17:21:42,557 INFO sqlalchemy.engine.Engine [generated in 0.00430s] ()
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<Address(email_address='hola@col.com')>
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<Address(email_address='rock@lml.com')>
2022-12-15 17:21:42,552 INFO sqlalchemy.engine.Engine SELECT users.id AS users_id, users.name AS users_name, users.full
name AS users_fullname, users.password AS users_password, addresses.id AS addresses_id, addresses.email_address AS addr
esses_email_address, addresses.user_id AS addresses_user_id
FROM users, addresses
WHERE users.id = addresses.user_id
2022-12-15 17:21:42,557 INFO sqlalchemy.engine.Engine [generated in 0.00430s] ()
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<Address(email_address='hola@col.com')>
<User(name='ed', fullname='Eduardo Jones', password='f8s7ccs')>
<Address(email_address='rock@lml.com')>
```

Resultados

Las evidencias del funcionamiento pueden ser observadas en las salidas de las celdas de código de la aplicación jupyter notebook. No obstante, se destaca la facilidad del uso de objetos y su reflejo en la base de datos.

Conclusiones

El uso de ORM, en específico SQLAlchemy en Python es que proporcionan una forma fácil y eficiente de almacenar y manipular información mediante el uso de objetos en lugar de escribir consultas SQL complejas. Esto permite a los desarrolladores enfocarse en la lógica de la aplicación en lugar de preocuparse por la sintaxis de la base de datos. Además, SQLAlchemy ofrece una amplia gama de características avanzadas y herramientas para trabajar con bases de datos de una manera eficiente y segura. En resumen, el uso de ORM y SQLAlchemy en Python puede simplificar significativamente el proceso de almacenamiento y manipulación de datos en una aplicación.

Bibliografía

1. SQLAlchemy
2. SQLAlchemy ORM Tutorial for Python Developers
3. SQLAlchemy query filter