Министерство Образования Российской Федерации
УрФУ
Образовательных информационных технологий

Лабораторная работа №2
Тема: «Работа с SQLAlchemy и alembic»

Выполнил:
Нарсеев А.В.

Екатеринбург, 2025

Цель работы

Освоить принципы работы с библиотеками SQLAlchemy и Alembic для создания и управления реляционными базами данных на Python, изучить механизмы миграции базы данных.

Задачи

1. Создали окружение:

```
narseev@Mac LR2 % python3 -m venv venv
narseev@Mac LR2 % source venv/bin/activate
(venv) narseev@Mac LR2 % pip freeze > requirements.txt
(venv) narseev@Mac LR2 % pip install sqlalchemy alembic psycopg2-binary
Collecting sqlalchemy
  Using cached sqlalchemy-2.0.44-cp312-cp312-macosx_11_0_arm64.whl.metadata (9.5 kB)
Collecting alembic
  Using cached alembic-1.17.0-py3-none-any.whl.metadata (7.2 kB)
Collecting psycopg2-binary
  Using cached psycopg2_binary-2.9.11-cp312-cp312-macosx_11_0_arm64.whl.metadata (4.9 kB)
Collecting typing-extensions>=4.6.0 (from sqlalchemy)
  Using cached typing_extensions-4.15.0-py3-none-any.whl.metadata (3.3 kB)
Collecting Mako (from alembic)
  Using cached mako-1.3.10-py3-none-any.whl.metadata (2.9 kB)
Collecting MarkupSafe>=0.9.2 (from Mako->alembic)
  Using cached markupsafe-3.0.3-cp312-cp312-macosx_11_0_arm64.whl.metadata (2.7 kB)
Using cached sqlalchemy-2.0.44-cp312-cp312-macosx_11_0_arm64.whl (2.1 MB)
Using cached alembic-1.17.0-py3-none-any.whl (247 kB)
Using cached psycopg2_binary-2.9.11-cp312-cp312-macosx_11_0_arm64.whl (3.9 MB)
Using cached typing_extensions-4.15.0-py3-none-any.whl (44 kB)
Using cached mako-1.3.10-py3-none-any.whl (78 kB)
Using cached markupsafe-3.0.3-cp312-cp312-macosx_11_0_arm64.whl (12 kB)
Installing collected packages: typing-extensions, psycopg2-binary, MarkupSafe, sqlalchemy, Mako, alembic
Successfully installed Mako-1.3.10 MarkupSafe-3.0.3 alembic-1.17.0 psycopg2-binary-2.9.11 sqlalchemy-2.0.44 typing-extensions-4.15.0

[notice] A new release of pip is available: 24.2 -> 25.2
[notice] To update, run: pip install --upgrade pip
(venv) narseev@Mac LR2 % pip freeze > requirements.txt
```

2. Инициализировали миграции:

```
(venv) narseev@Mac LR2 % alembic init migrations
  Creating directory /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/migrations ...  done
  Creating directory /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/migrations/versions ...  done
  Generating /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/migrations/script.py.mako ...  done
  Generating /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/migrations/env.py ...  done
  Generating /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/migrations/README ...  done
  Generating /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/alembic.ini ...  done
  Please edit configuration/connection/logging settings in /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/alembic.ini before proceeding.
```

3. Создали контейнер с базой, создали базу labdb:

```
narseev@Mac LR2 % docker run --name lab2-postgres -e POSTGRES_PASSWORD=pass -p 5432:5432 -d postgres
147ec6846d254c7890901d9efe20116dc75c1281b22d211f77d3b104547a090b
narseev@Mac LR2 % docker exec -it lab2-postgres psql -U postgres
psql (18.0 (Debian 18.0-1.pgdg13+3))
Type "help" for help.

postgres=# CREATE DATABASE labdb;
CREATE DATABASE
```

```
postgres=# \l
                                                    List of databases
   Name    |  Owner   | Encoding | Locale Provider |  Collate   |   Ctype    | Locale | ICU Rules |   Access privileges
-----------+----------+----------+-----------------+------------+------------+--------+-----------+-----------------------
 labdb     | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 postgres  | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           |
 template0 | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           | =c/postgres          +
           |          |          |                 |            |            |        |           | postgres=CTc/postgres
 template1 | postgres | UTF8     | libc            | en_US.utf8 | en_US.utf8 |        |           | =c/postgres          +
           |          |          |                 |            |            |        |           | postgres=CTc/postgres
(4 rows)

postgres=# \q
narseev@Mac LR2 %
```

4. Создали миграцию:

```
(venv) narseev@Mac LR2 % alembic revision --autogenerate -m "create users and addresses tables"
  INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
  INFO  [alembic.runtime.migration] Will assume transactional DDL.
  INFO  [alembic.autogenerate.compare] Detected added table 'users'
  INFO  [alembic.autogenerate.compare] Detected added table 'addresses'
    Generating /Users/narseev/Documents/UrFU/term_1/AppDev/LR2/migrations/versions/73ce9c5af9f5_create_users_and_addresses_tables.py ...  done
(venv) narseev@Mac LR2 % alembic upgrade head
  INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
  INFO  [alembic.runtime.migration] Will assume transactional DDL.
  INFO  [alembic.runtime.migration] Running upgrade  -> 73ce9c5af9f5, create users and addresses tables
```

В базе появились таблицы:

```
narseev@Mac LR2 % docker exec -it lab2-postgres psql -U postgres -d labdb
psql (18.0 (Debian 18.0-1.pgdg13+3))
Type "help" for help.

labdb=# \dt
               List of tables
 Schema |      Name       | Type  |  Owner
--------+-----------------+-------+----------
 public | addresses       | table | postgres
 public | alembic_version | table | postgres
 public | users           | table | postgres
(3 rows)
```

5. Добавили пользователей и адреса в базу:

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

user = User(username=f"Ivan",email=f"ivan@example.com")
user.addresses = [Address(street=f"Mira 19", city="Yekaterinburg", country="Russia", is_primary=True)]
session.add(user)

user = User(username=f"Petr",email=f"petr@example.com")
user.addresses = [Address(street=f"Lenina 1", city="Moscow", country="Russia", is_primary=True)]
session.add(user)

user = User(username=f"Vasiliy",email=f"vasiliy@example.com")
user.addresses = [Address(street=f"Nevsky Prospekt 123", city="Saint Petersburg", country="Russia", is_primary=True)]
session.add(user)

user = User(username=f"Nikolaii",email=f"nikolaii@example.com")
user.addresses = [Address(street=f"Lenina 22", city="Yekaterinburg", country="Russia", is_primary=True)]
session.add(user)

user = User(username=f"Anatoly",email=f"anatoly@example.com")
user.addresses = [Address(street=f"Tverskaya 15", city="Moscow", country="Russia", is_primary=True)]
session.add(user)
session.commit()
```

(variable) session: Session

```
labdb=# select * from users;
                  id                  | username |        email        | description |         created_at         |         updated_at
--------------------------------------+----------+---------------------+-------------+----------------------------+----------------------------
 39a25d1f-e90a-441e-b593-28acb73ab1bc | Ivan     | ivan@example.com    |             | 2025-10-18 20:07:04.589323 | 2025-10-18 20:07:04.589326
 6c586663-c88b-49bf-a43b-a1fd5586bb14 | Petr     | petr@example.com    |             | 2025-10-18 20:07:04.589329 | 2025-10-18 20:07:04.58933
 90042964-da9f-4496-9500-703b7d2923e6 | Vasiliy  | vasiliy@example.com |             | 2025-10-18 20:07:04.589332 | 2025-10-18 20:07:04.589333
 420548f0-f129-4fc4-b18a-56f8e76b3ebf | Nikolaii | nikolaii@example.com |             | 2025-10-18 20:07:04.589335 | 2025-10-18 20:07:04.589335
 7d0b03b2-f806-4f5b-9dcb-1a9330349214 | Anatoly  | anatoly@example.com |             | 2025-10-18 20:07:04.589337 | 2025-10-18 20:07:04.589337
(5 rows)

labdb=# select id, user_id, city, country from addresses;
                  id                  |               user_id                |       city       | country
--------------------------------------+--------------------------------------+------------------+---------
 faba9914-e8d2-40b8-a62c-164402ebf18a | 39a25d1f-e90a-441e-b593-28acb73ab1bc | Yekaterinburg    | Russia
 ec260f29-eba8-46dd-82a3-80040a0bb15b | 6c586663-c88b-49bf-a43b-a1fd5586bb14 | Moscow           | Russia
 548dbae2-c324-403b-97e5-8b4b712f41ac | 90042964-da9f-4496-9500-703b7d2923e6 | Saint Petersburg | Russia
 1f44dc79-51e4-4963-9a6b-489ef64f1cb3 | 420548f0-f129-4fc4-b18a-56f8e76b3ebf | Yekaterinburg    | Russia
 8dc5fd00-683f-4c25-baff-a7b028923406 | 7d0b03b2-f806-4f5b-9dcb-1a9330349214 | Moscow           | Russia
(5 rows)
```

Получаем данные через sqlalchemy:

```python
stmt = select(User).options(selectinload(User.addresses))
results = session.execute(stmt).scalars().all()

for user in results:
    print(f"User: {user.username} ({user.email})")
    for addr in user.addresses:
        print(f"  Address: {addr.street}, {addr.city}, {addr.country} (primary={addr.is_primary})")
```

```
(venv) narseev@Mac LR2 % python query_users_addresses.py
User: Ivan (ivan@example.com)
  Address: Mira 19, Yekaterinburg, Russia (primary=True)
User: Petr (petr@example.com)
  Address: Lenina 1, Moscow, Russia (primary=True)
User: Vasiliy (vasiliy@example.com)
  Address: Nevsky Prospekt 123, Saint Petersburg, Russia (primary=True)
User: Nikolaii (nikolaii@example.com)
  Address: Lenina 22, Yekaterinburg, Russia (primary=True)
User: Anatoly (anatoly@example.com)
  Address: Tverskaya 15, Moscow, Russia (primary=True)
```

6. Создаем новую миграцию и добавляем дополнительные таблицы продукции и заказов:

```
(venv) narseev@Mac LR2 % python fill_db_2.py
Ivan заказал Laptop x1 на Mira 19
Petr заказал Mouse x2 на Lenina 1
Vasiliy заказал Keyboard x1 на Nevsky Prospekt 123
Nikolaii заказал Monitor x3 на Lenina 22
Anatoly заказал Headphones x2 на Tverskaya 15
```

Новые таблицы в БД (и новое поле description в users):

```
labdb=# select * from users;
                  id                  | username |        email         | description |
--------------------------------------+----------+----------------------+-------------+--
 39a25d1f-e90a-441e-b593-28acb73ab1bc | Ivan     | ivan@example.com     |             | 2
 6c586663-c88b-49bf-a43b-a1fd5586bb14 | Petr     | petr@example.com     |             | 2
 90042964-da9f-4496-9500-703b7d2923e6 | Vasiliy  | vasiliy@example.com  |             | 2
 420548f0-f129-4fc4-b18a-56f8e76b3ebf | Nikolaii | nikolaii@example.com |             | 2
 7d0b03b2-f806-4f5b-9dcb-1a9330349214 | Anatoly  | anatoly@example.com  |             | 2
(5 rows)
```

```
labdb=# \dt
              List of tables
 Schema |      Name       | Type  |  Owner
--------+-----------------+-------+----------
 public | addresses       | table | postgres
 public | alembic_version | table | postgres
 public | orders          | table | postgres
 public | products        | table | postgres
 public | users           | table | postgres
(5 rows)

labdb=# select * from orders;
```

```
                  id                  |               user_id                |              address_id              |              product_id              | quantity |         created_at
--------------------------------------+--------------------------------------+--------------------------------------+--------------------------------------+----------+----------------------------
 3d93b211-b562-40cb-9aea-1adbcf773fbc | 39a25d1f-e90a-441e-b593-28acb73ab1bc | faba9914-e8d2-40b8-a62c-164402ebf18a | 317c7d8c-97db-4651-8923-7896ef449249 |        1 | 2025-10-18 20:14:28.29082
 2c18eae1-0fd8-4f61-8959-14064c393515 | 6c586663-c88b-49bf-a43b-a1fd5586bb14 | ec260f29-eba8-46dd-82a3-80040a0bb15b | 6ea7ec4a-30b1-4c8b-9d03-7b52a180da5d |        2 | 2025-10-18 20:14:28.290825
 e8212048-806f-4dc9-a50f-a6a40e2ece9f | 90042964-da9f-4496-9500-703b7d2923e6 | 548dbae2-c324-403b-97e5-8b4b712f41ac | 533854b7-b6ce-4364-8bcc-405d4124e8ac |        1 | 2025-10-18 20:14:28.290827
 0d542d01-02cb-4233-87d3-fad41690dbf0 | 420548f0-f129-4fc4-b18a-56f8e76b3ebf | 1f44dc79-51e4-4963-9a6b-489ef64f1cb3 | 30312ab8-820b-4064-802f-14ac1d7fb52a |        3 | 2025-10-18 20:14:28.290829
 51b82524-c972-47ec-b505-78ffeeb9963c | 7d0b03b2-f806-4f5b-9dcb-1a9330349214 | 8dc5fd00-683f-4c25-baff-a7b028923406 | 1dfc5300-90ce-400b-8211-35c523b23aa5 |        2 | 2025-10-18 20:14:28.290831
(5 rows)
```

```
labdb=# select * from products;
                  id                  |    name    | price |         created_at
--------------------------------------+------------+-------+----------------------------
 317c7d8c-97db-4651-8923-7896ef449249 | Laptop     |  1200 | 2025-10-18 20:14:28.282346
 6ea7ec4a-30b1-4c8b-9d03-7b52a180da5d | Mouse      |    25 | 2025-10-18 20:14:28.282352
 533854b7-b6ce-4364-8bcc-405d4124e8ac | Keyboard   |    50 | 2025-10-18 20:14:28.282355
 30312ab8-820b-4064-802f-14ac1d7fb52a | Monitor    |   300 | 2025-10-18 20:14:28.282357
 1dfc5300-90ce-400b-8211-35c523b23aa5 | Headphones |    80 | 2025-10-18 20:14:28.282359
(5 rows)
```

Вопросы

1. Какие есть подходы маппинга в SQLAlchemy? Когда следует использовать каждый подход?

   SQLAlchemy поддерживает несколько подходов маппинга «ORM-объект - таблица»

   - Декларативный: наиболее популярный подход. Создаем классы, которые наследуются от Base, и описываем колонки как атрибуты класса
   - Классический: сначала создаются таблицы, потом отдельным mapper() связывается с классом
   - Императивный: гибридный подход, похож на классический, но с современными возможностями ORM

2. Как Alembic отслеживает текущую версию базы данных?

   - Alembic создаёт специальную таблицу в базе данных: alembic_version
   - в этой таблице хранится ID последней применённой миграции (revision)
   - когда выполняется alembic upgrade head, Alembic сверяет alembic_version с файлами миграций и применяет новые изменения

3. Какие типы связей между таблицами вы реализовали в данной работе?

   Один ко многим

   User -> Addresses

   Один пользователь может иметь несколько адресов

4. Что такое миграция базы данных и почему она важна?

   Миграция - скрипт, который изменяет структуру базы данных (добавляет/удаляет таблицы, колонки, ограничения)

   Зачем нужна:

   • Позволяет синхронизировать базу данных с моделью приложения

   • Облегчает работу в команде - все используют одинаковую структуру

   • Позволяет безопасно обновлять базу без потери данных

5. Как обрабатываются отношения многие-ко-многим в SQLAlchemy?

   Через таблицу-связку - вспомогательную таблицу без отдельного класса

```python
association = Table(
    'association', Base.metadata,
    Column('user_id', ForeignKey('users.id'), primary_key=True),
    Column('group_id', ForeignKey('groups.id'), primary_key=True)
)

class User(Base):
    __tablename__ = 'users'
    id = Column(Integer, primary_key=True)
    groups = relationship("Group", secondary=association, back_populates="users")

class Group(Base):
    __tablename__ = 'groups'
    id = Column(Integer, primary_key=True)
    users = relationship("User", secondary=association, back_populates="groups")
```

6. Каков порядок действий при возникновении конфликта версий в Alembic?

   Конфликты версий возникают, когда две или более ветки миграций не имеют общего предка

   a. Определить основную ветку - какая миграция будет считаться основной
   b. Использовать alembic merge, чтобы объединить ветки
   c. В результирующей миграции объединить изменения вручную, если нужно

Ссылки

GitHub: https://github.com/gh-u14/AppDev/