Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования «Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РтФ
ШПиАО Прикладной анализ данных

ОТЧЕТ

Лабораторная работа №2: Работа с SQLAlchemy и alembic

Преподаватель: Кузьмин Денис Иванович
Обучающийся группы РИМ–150950: Поварнина Екатерина Дмитриевна

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

Ход работы

Для начала создала новый проект в PyCharm. Далее установила библиотеки SQLAlchemy, Alembic и psycopg2-binary. Создала модели данных с использованием современного синтаксиса Mapped: модель User с полями UUID id, username, email, created_at, updated_at и модель Address с полями UUID id, user_id, street, city, state, zip_code, country, is_primary, created_at, updated_at, настроила связь один-ко-многим через relationship.

```
target_metadata = Base.metadata
```

```
alembic upgrade head
```

Инициализировала миграции командой alembic init migrations, настроила файл alembic.ini, заменив строку подключения к БД, и файл env.py, указав target_metadata = Base.metadata. Создала и применила первую миграцию с помощью команд alembic revision --autogenerate и alembic upgrade head.

Создала фабрику подключений с echo=True для логирования SQL и наполнила базу данных, создав 5 пользователей и их адресов, используя контекстный менеджер для сессий.

```
engine = create_engine(
    connect_url,
    echo=True  # Логирование SQL-запросов
)
```

Протестировала запросы связанных данных с использованием selectinload. Затем расширила модели, добавив в User поле description и создав новые модели Product и Order с необходимой структурой и связями, включая связь многие-ко-многим между Order и Product через ассоциативную таблицу. Создала и применила вторую миграцию, после чего добавила в базу данных 5 продуктов и 5 заказов.

```
select(User).options(selectinload(User.addresses))
```

```
from sqlalchemy import select
```

```
from sqlalchemy.orm import selectinload
```

Консоль:

```
(.venv) PS C:\Users\Dmitry\PycharmProjects\PovarninaLW2> pip install sqlalchemy alembic psycopg2-binary
Collecting sqlalchemy
  Obtaining dependency information for sqlalchemy from https://files.pythonhosted.org/packages/03/51/665617fe4f8c6450f42a6d8d69243f9420f5677395572c2fe9d21b493b7
/sqlalchemy-2.0.44-cp313-cp313-win_amd64.whl.metadata
  Downloading sqlalchemy-2.0.44-cp313-cp313-win_amd64.whl.metadata (9.8 kB)
Collecting alembic
  Obtaining dependency information for alembic from https://files.pythonhosted.org/packages/44/1f/38e29b06bfed7818ebba1f84904afdc8153ef7b6c7e0d8f3bc6643f5989c/a
embic-1.17.0-py3-none-any.whl.metadata
  Downloading alembic-1.17.0-py3-none-any.whl.metadata (7.2 kB)
Collecting psycopg2-binary
  Obtaining dependency information for psycopg2-binary from https://files.pythonhosted.org/packages/80/2d/1bb683f64737bbb1f86c82b7359db1eb2be4e2c0c13b947f80efef
7d3e5/psycopg2_binary-2.9.11-cp313-cp313-win_amd64.whl.metadata
  Downloading psycopg2_binary-2.9.11-cp313-cp313-win_amd64.whl.metadata (5.1 kB)
Collecting greenlet>=1 (from sqlalchemy)
  Obtaining dependency information for greenlet>=1 from https://files.pythonhosted.org/packages/0b/55/2321e43595e6801e105fcfdee02b34c0f996eb71e6ddffca6b10b7e1d7
1/greenlet-3.2.4-cp313-cp313-win_amd64.whl.metadata
  Downloading greenlet-3.2.4-cp313-cp313-win_amd64.whl.metadata (4.2 kB)
Collecting typing-extensions>=4.6.0 (from sqlalchemy)
```

```
(.venv) PS C:\Users\Dmitry\PycharmProjects\PovarninaLW2> alembic init alembic
Creating directory C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic ...  done
Creating directory C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic\versions ...  done
Generating C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic.ini ...  done
Generating C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic\env.py ...  done
Generating C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic\README ...  done
Generating C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic\script.py.mako ...  done
Please edit configuration/connection/logging settings in C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic.ini before proceeding.
(.venv) PS C:\Users\Dmitry\PycharmProjects\PovarninaLW2> alembic revision --autogenerate -m "Initial migration"
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 C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic\versions\c8d66789f98c_initial_migration.py ...  done
(.venv) PS C:\Users\Dmitry\PycharmProjects\PovarninaLW2> alembic upgrade head
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade  -> c8d66789f98c, Initial migration
(.venv) PS C:\Users\Dmitry\PycharmProjects\PovarninaLW2> alembic revision --autogenerate -m "Add description, products and orders"
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added table 'products'
INFO  [alembic.autogenerate.compare] Detected added table 'orders'
INFO  [alembic.ddl.postgresql] Detected sequence named 'users_id_seq' as owned by integer column 'users(id)', assuming SERIAL and omitting
INFO  [alembic.ddl.postgresql] Detected sequence named 'addresses_id_seq' as owned by integer column 'addresses(id)', assuming SERIAL and omitting
INFO  [alembic.autogenerate.compare] Detected added column 'users.description'
Generating C:\Users\Dmitry\PycharmProjects\PovarninaLW2\alembic\versions\bed559af0e07_add_description_products_and_orders.py ...  done
(.venv) PS C:\Users\Dmitry\PycharmProjects\PovarninaLW2> alembic upgrade head
INFO  [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO  [alembic.runtime.migration] Will assume transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade c8d66789f98c -> bed559af0e07, Add description, products and orders
```

Все файлы в Гите.

Ответы на вопросы:

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

В SQLAlchemy существуют два основных подхода к маппингу: декларативный и императивный. Декларативный подход используется чаще всего и заключается в создании классов, наследующихся от declarative_base(), с объявлением таблиц и полей непосредственно в классе. Императивный подход применяется реже, когда требуется явно сконфигурировать маппинг с помощью функции mapper(), что полезно для интеграции с существующими системами или при необходимости полного контроля над процессом.

2. Как Alembic отслеживает текущую версию базы данных?

Alembic отслеживает текущую версию базы данных с помощью специальной таблицы alembic_version, которая создается в целевой базе данных и содержит одну строку с идентификатором последней примененной миграции.

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

В работе были реализованы два типа связей между таблицами: один-ко-многим между пользователями и адресами и многие-ко-многим между заказами и продуктами через ассоциативную таблицу.

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

Миграция базы данных — это управляемое версиями изменение схемы базы данных. Она важна потому, что позволяет контролировать историю изменений схемы БД, обеспечивает воспроизводимость на разных окружениях, автоматизирует развертывание и предоставляет возможность отката изменений в случае ошибки.

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

Отношения многие-ко-многим в SQLAlchemy обрабатываются с помощью ассоциативной таблицы, которая содержит внешние ключи к обеим связываемым таблицам. В моделях определяется relationship с аргументом secondary, указывающим на эту ассоциативную таблицу.

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

При возникновении конфликта версий в Alembic порядок действий следующий: сначала нужно определить текущее состояние с помощью команд alembic current и alembic history, затем найти общую точку расхождения версий. Самый безопасный путь - откатить базу до общей версии командой alembic downgrade и затем применить все миграции до головы командой alembic upgrade head. Альтернативный вариант - создание миграции слияния с помощью alembic merge.