**Лабораторная работа №2: Работа с SQLAlchemy и alembic**

**Студент:** Осьминин Никита Борисович

**Группа:** Прикладной анализ данных, РИМ-150950

**Цель работы**

Освоить фундаментальные концепции ORM-моделей и миграций: создание ORM модели, её связка с SQL-таблицей, последующее версионирование БД.

**Описание задачи**

Необходимо было установить ряд python-библиотек, наиболее важными из которых являлись sqlAlchemy – для управления БД через ORM модель и alembic – для поддержки миграций. БД была заполнена начальными данными, а позже её структура обновлялась путём добавления новых ORM-моделей и расширения старых.

**Ход работы**

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Для начала я установил все необходимые библиотеки, функции которых были описаны в блоке «Описание задачи».

Изображение выглядит как текст, снимок экрана, программное обеспечение

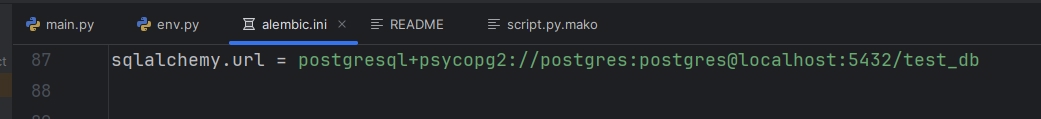
Содержимое, созданное искусственным интеллектом, может быть неверным.

Создал 2 ORM-модели: User и Address с внешним ключом на User. Обе модели имеют relationship, что позволяет делает аналог JOIN’а при запросе любой из этих сущностей из кода, помещая подходящие данные из связанной таблицы в поле модели.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

С помощью alembic проинициализировал миграции.



В файле конфигурации alembic задал строку подключения к моей БД из 1 лабораторной, которую заранее очистил от всех данных.

Изображение выглядит как текст, программное обеспечение, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

В файле окружения, сгенерированной командой alembic init указал базовую модель ORM от которой наследуются все дочерние (например, User и Address) для того, чтобы библиотека понимала, основываясь на каких моделях ей формировать файлы миграций.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

На данный момент у нас уже есть 2 ORM-модели, которые отсутствуют в связанной БД. Инициализируем миграции.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

После генерации миграции, которая на данный момент присутствует лишь локально, необходимо её применить к БД, что и делает команда alembic upgrade head.

Изображение выглядит как текст, программное обеспечение, Шрифт, Значок на компьютере

Содержимое, созданное искусственным интеллектом, может быть неверным.

Можно убедиться в том, что таблицы действительно создались, воспользовавшись pgAdmin. Таблица alembic\_version содержит id последней миграции, применённой командой alembic upgrade head.

Изображение выглядит как текст, Шрифт, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Заполним таблицу пользователей начальными данными. Обратим внимание на то, что мы можем указывая значения полей класса Python задавать будущие значения столбцов строк в БД.

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Содержимое, созданное искусственным интеллектом, может быть неверным.

При попытке заполнения данными, появилась ошибка о невозможности задания null-значения колонке updated\_at.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Добавляем updated\_at явную возможность быть nullable.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Создаём миграцию и обновляем БД.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Повторяем попытку заполнения таблицы пользователей данными и видим, что на этот раз всё проходит успешно.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Убедимся в том, что строки действительно появились в таблице, произведя выборку строк. Обратим внимание на то, что select проводится с тем самым relation-полем, которое упоминалось ранее.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

В терминал вывелись имена всех пользователей, ранее внедрённых в таблицу.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Сделаем то же самое с таблицей адресов. Вместо user\_id можем прокидывать саму модельку юзера из-за указанной в ORM-модели связи.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.

Данные действительно попали в таблицу адресов. Внешний ключ user\_id был подставлен sqlAlchemy самостоятельно на основе поля user.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Поменяем модельку пользователя, добавив ему поле description.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Были добавлены 2 новые модели: Product и Order. Модель Order содержит 3 внешних ключа на ранее созданные таблицы, в relationship можно опустить параметр back\_populates, т.к нам необязательно синхронизировать данные с остальными моделями.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Создаём миграцию и применяем её.

Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, Графическое программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, Мультимедийное программное обеспечение, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Заполняем новые таблицы начальными данными.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Содержимое, созданное искусственным интеллектом, может быть неверным.

Таблицы действительно были созданы и заполнены начальными данными.

**Вопросы**

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

* Декларативный – наследование классов от базового. Можно использовать как дефолтное решение в новых проектах
* Через функцию mapper. Использовать необходимо при работе с уже существующей схемой БД.
* Через Automap. Использовать необходимо при работе с уже существующей схемой БД и при условии отсутствия необходимости изменения моделей.

1. *Как Alembic отслеживает текущую версию базы данных?* Через таблицу alembic\_version в целевой БД.
2. *Какие типы связей между таблицами вы реализовали в данной работе?* 1 ко многим – например, при связи «Пользователь – Адрес».
3. *Что такое миграция базы данных и почему она важна?* Это изменения версии БД, она так же важна, как и git-версии из-за большой вероятности того, что над одним проектом работает несколько программистов и им необходимо иметь быструю возможность накататить/откатить версию БД до актуальной.
4. *Как обрабатываются отношения многие-ко-многим в SQLAlchemy?* Создаётся ассоциативная таблица.
5. *Каков порядок действий при возникновении конфликта версий в Alembic?* Необходимо определить конфликтующие файлы миграций и замёрджить их.

**Вывод**

В ходе работы я получил базовое представление об ORM-моделях sqlAlchemy: как создать таблицу, задать ей список колонок, на колонки повесить атрибуты, научился задавать связи между таблицами. Также были опробованы миграции посредством библиотеки alembic, позволяющие отслеживать версионирование БД.