Прокуронова Анастасия Юрьевна
старший преподаватель
ЧУ ВО «Московская академия предпринимательства»
г. Москва
Макарихин Артем Андреевич
студент
Колледж ЧУ ВО «Московская академия предпринимательства»
г. Москва

ВОЗМОЖНОСТИ БИБЛИОТЕКИ SQLALCHEMY ПРИ РАЗРАБОТКЕ БД

Аннотация*: в статье рассматривается применение библиотеки SQLAlchemy для разработки базы данных (БД) на примере предметной области «Кинозал». Используемая СУБД – SQLite. Представлены структура БД, а также примеры использования основных возможностей SQLAlchemy, таких как ORM, работа с транзакциями, оптимизация запросов и интеграция с другими инструментами. Описаны преимущества использования SQLAlchemy при создании и обслуживании БД для кинозала.*

Ключевые слова*: SQLAlchemy, база данных, SQLite, ORM, Python, разработка.*

Введение

В современном мире разработка программного обеспечения часто сопровождается созданием и обслуживанием баз данных. SQLAlchemy – одна из самых популярных библиотек Python для работы с базами данных, предоставляющая мощный и гибкий интерфейс для взаимодействия с различными СУБД. В данной статье мы рассмотрим возможности SQLAlchemy на примере создания базы данных для кинозала, используя SQLite в качестве СУБД. Продемонстрировано, как SQLAlchemy помогает упростить и ускорить процесс разработки БД, обеспечивая при этом высокую производительность и надежность.

Структура базы данных

Рассмотрим структуру базы данных для кинозала, состоящую из четырех таблиц: «Фильмы», «Места», «Билеты» и «Сеансы». Каждая таблица имеет следующую структуру:

Таблица «Фильмы»:

− id – уникальный идентификатор фильма (первичный ключ);
− title – название фильма;
− duration – продолжительность фильма в минутах;
− description – краткое описание сюжета фильма.

Таблица «Места»:

− id – уникальный идентификатор места (первичный ключ);
− row – номер ряда;
− seat_number – номер места в ряду.

Таблица «Билеты»:

− id – уникальный идентификатор билета (первичный ключ);
− film_id – внешний ключ на фильм;
− place_id – внешний ключ на место;
− session_id – внешний ключ на сеанс;
− price – стоимость билета.

Таблица «Сеансы»:

− id – уникальный идентификатор сеанса (первичный ключ);
− date_time – дата и время начала сеанса;
− hall_id – внешний ключ на зал кинотеатра (не рассматривается в данной статье).

На рисунке 1 представлена структура выведенной таблицы «Фильмы».

Столбец	Тип данных	Описание
`id`	INTEGER	Уникальный идентификатор фильма
`title`	VARCHAR(255)	Название фильма
`duration`	INTEGER	Продолжительность фильма
`description`	TEXT	Описание сюжета фильма

Рис. 1. Структура таблицы «Фильмы»

Использование SQLAlchemy для создания БД

Для создания схемы базы данных используем ORM-модель SQLAlchemy. Ниже (рисунок 2) представлены классы моделей для каждой таблицы.

```python
from sqlalchemy import Column, Integer, String, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

class Film(Base):
    __tablename__ = 'films'
    id = Column(Integer, primary_key=True)
    title = Column(String)
    duration = Column(Integer)
    description = Column(String)

class Place(Base):
    __tablename__ = 'places'
    id = Column(Integer, primary_key=True)
    row = Column(Integer)
    seat_number = Column(Integer)

class Ticket(Base):
    __tablename__ = 'tickets'
    id = Column(Integer, primary_key=True)
    film_id = Column(Integer, ForeignKey('films.id'))
    place_id = Column(Integer, ForeignKey('places.id'))
    session_id = Column(Integer, ForeignKey('sessions.id'))
    price = Column(Float)

class Session(Base):
    __tablename__ = 'sessions'
    id = Column(Integer, primary_key=True)
    date_time = Column(DateTime)
    hall_id = Column(Integer, ForeignKey('halls.id'))  # Предположительно есть таблица halls

# Создаем схему базы данных
Base.metadata.create_all(engine)
```

Рис. 2. Классы моделей таблиц

Работа с транзакциями

SQLAlchemy поддерживает транзакции, что гарантирует атомарность операций и сохраняет консистентность данных. Пример использования транзакций представлен на рисунке 3.

```python
with session.begin():
    ticket = Ticket(
        film_id=film.id,
        place_id=place.id,
        session_id=session.id,
        price=500
    )
    session.add(ticket)
```

Рис. 3. Пример использования транзакций

Оптимизация запросов

SQLAlchemy позволяет оптимизировать запросы к базе данных, что улучшает производительность приложения. Пример оптимизации запроса представлен на рисунке 4.

```python
# Запрос всех билетов на определенный сеанс
tickets = session.query(Ticket).join(Session).filter(Session.date_time == datetime.now()).all()
```

Рис. 4. Пример оптимизации запроса

Интеграция с другими инструментами

SQLAlchemy хорошо интегрируется с различными веб-фреймворками, такими как Flask и Django. Пример интеграции с Flask можно увидеть на рисунке 5.

```python
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///cinema.db'
db = SQLAlchemy(app)
```

Рис. 5. Пример интеграции с Flask

Таким образом, SQLAlchemy предоставляет широкие возможности для разработки и обслуживания баз данных, делая этот процесс эффективным и надежным.

Список литературы

1. Кузнецов И.А. Python и базы данных: современный подход / И.А. Кузнецов. – Новосибирск: Наука, 2016.

2. Смирнов В.В. Применение SQLAlchemy в разработке web-приложений / В.В. Смирнов. – Екатеринбург: УрГУПС, 2018.

3. Сидоров М.Ю. Возможности искусственного интеллекта для оптимизации процессов разработки программного обеспечения / М.Ю. Сидоров // Программирование и компьютерные науки. – 2022. – Т. 19. №2. – С. 34–41.

4. SQLAlchemy Core documentation [Electronic resource]. – Access mode: https://docs.sqlalchemy.org/en/14/core/index.html (дата обращения: 01.04.2025).

5. Python Database Programming by Albert Lukaszewski. Packt Publishing, 2016.

6. High Performance Python by Ian Ozsvald and Micha Gorelick. O'Reilly Media, 2020.