УДК 004.9

Миграция данных на Python

Пастушенко Владислав Александрович
Брянский государственный университет имени академика И.Г. Петровского
магистрант

Аннотация
В статье рассмотрено понятие миграции данных, способы их реализации на языке программирования Python. Рассмотрена работа с инструментом Alembic.
Ключевые слова: миграция данных, Python, SQL, yoyo-migrations, Alembic, разработка.

Data migration on Python

Pastushenko Vladislav Aleksandrovich
Bryansk state university named by academician I.G. Petrovsky
master student

Abstract
This article discusses the concept of data migration and how to implement it in the Python programming language. Working with the Alembic tool is considered.
Keywords: data migration, Python, SQL, yoyo-migrations, Alembic, development.

С другой стороны миграция – это процесс, при котором выполняется код, изменяющий состояние базы данных. С другой стороны, это код, который переводит базу данных из одно определенного состояния в другое и так же обращает эти изменения.

Миграции должны обладать тремя важными свойствам: атомарность, обратимость и упорядоченность.

Атомарность означает, что миграция или их группа должны быть применены либо полностью, либо никак.

Миграции должны содержать код, который позволит вернуться к предыдущему состоянии базы данных, тем самым обеспечивая обратимость.

Также миграции должны быть упорядочены, т.е. должно быть понятно, в каком порядке их необходимо применять.

Для реализации миграций в языке Python есть несколько решений.

Файлы с SQL-запросами. Непосредственно код миграций пишется на языке запросов в SQL и применяются к базе данных вручную. Такой подход имеет ряд недостатков:

– ручное хранение состояния базы данных;

– отсутствие защиты от выполнения миграций, не рассчитанных на работу с текущим состояние базы данных;
– запросы необходимо писать вручную, учитывая связи таблиц;
– в каждой миграции необходимо не забыть использовать транзакции.

Библиотека yoyo-migrations. Поддерживает самые распространенные реляционные базы данных: SQLite, PostgreSQL, MySQL, MariaDB. Так же как и предыдущее решение использует сырые SQL запросы, но имеет ряд достоинств в сравнении с ним:
– автоматически учитывает примененные миграции, тем самым хранит состояние базы;
– имеет защиту от применения неактуальных миграций для текущего состояния базы данных;
– выполняет каждую миграцию в отдельной транзакции.

Но данная библиотека имеет ряд недостатков:
– код миграций необходимо писать вручную, учитывая связи таблиц;
– невозможность автоматизировать рутинную работу;
– невозможность использовать бизнес-логику, написанную на Python.

Библиотека Alembic. Данное решение не привязано к конкретному фреймворку, и для управления базой данных использует библиотеку SQLAlchemy. Главным достоинствами Alembic являются:
– автоматическая генерация кода миграций на основе моделей и анализа базы данных;
– возможность в коде миграций использовать бизнес-логику приложения без ограничений;
– расширение ограниченных возможностей ALTER в SQLite.

Для того чтоб начать использовать Alembic, необходимо в директории с проектом вызвать команды из Листинга 1.

Листинг 1

```
pip install alembic
alembic init alembic
```

При инициализации будут созданы следующие файлы:
– alembic/env.py – контроллирует выполнения миграцйи, позволяет настраивать интеграцию Alembic с проектом;
– alembic/script.py.mako – шаблон новых миграций;
– alembic/versions/ – директория, хранящая код миграций;
– alembic.ini – общий файл конфигурации.

Для генерации новой миграции необходимо выполнить команду из Листинга 2.

Листинг 2

```
alembic revision --message=”Initial” --autogenerate
```

При выполнении этой команды Alembic выполнит следующие действия:

- получит схему существующей базы данных и последнюю примененную миграцию, если такая существует;
- проанализирует разницу между описанными сущностями в проекте (таблицы, индексы, типы данных и т.п.) и в базе данных;
- сгенерирует новый файл на основе script.py.mako, который попытается привести базу данных в одинаковое состояние с проектом.

Файл миграции состоит из служебной информации, идентификатором ревизии, идентификатором предыдущей ревизии, два метода upgrade и downgrade. Автоматически сгенерированные миграции необходимо проверять после создания, так как в некоторых случаях Alembic может неправильно определить разницу сущностей, но это бывает в исключительных случаях.

После создания миграции ее необходимо применить к базе данных. Для этого необходимо вызвать команду из Листинга 3. Это позволит применить все доступные миграции к текущей базе данных.

Листинг 3

```
alembic upgrade head
```

Для того чтоб отменить обратить базу данных до определенной миграции, необходимо выполнить команду из Листинга 4, с указанием идентификатором нужной ревизии.

Листинг 4

```
alembic downgrade <revision_id>
```

На вопрос, когда и как лучше применять миграции есть два подхода. Если доступ к приложению ограничен, то желательно применять миграции автоматически, например при запуске приложения. Если сервис важный и к нему имеется доступ – применять миграции лучше вручную, это дает больший контроль и позволяет оперативно реагировать на проблемы.

Так же при миграции данных необходимо учитывать работу с необратимыми изменениям

Например, если в проекте не нужна большая таблица или столбец и есть желание их удалить без возможности восстановления – не стоит удалять их сразу. Лучше убрать к ним обращения, пометить эти столбцы, таблицы, типы данных специальным декоратором, который будет сообщать о доступе к

ресурсам, которые должны быть удалены и создать задачу на последующее удаление.

Как можно заметить, создание миграции базы данных являются неотъемлемой частью разработки. Они обеспечивают актуальность схемы базы данных и ее содержимого. В языке Python существуют несколько инструментов, которые могут организовать миграции любой сложности.

Библиографический список

1. Alembic Documentation: https://alembic.sqlalchemy.org/en/latest/ (Дата обращения: 12.01.2020)
2. Data Migrations: https://realpython.com/data-migrations/ (Дата обращения: 12.01.2020)
3. The Flask Mega-Tutorial Part IV: Database: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-iv-database (Дата обращения: 12.01.2020)
4. Yoyo database migrations: https://ollycope.com/software/yoyo/latest/ (Дата обращения: 12.01.2020)