УДК 004.05

МИГРАЦИЯ БАЗ ДАННЫХ В СОВРЕМЕННОЙ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

П.Д. ТАЛАЙКО
(Представлено: канд. техн. наук, доц. И.Б. БУРАЧЕНОК)

Даны понятия миграции баз данных. Рассмотрен популярный инструмент миграции и примеры его использования в современной разработке программного обеспечения.

В крупных проектах необходимо организовать администрирование баз данных (БД), которые содержат большое количество различных таблиц, представлений, индексов и пр. Так как процесс отслеживания изменений объектов БД или данных очень сложный, то на помощь приходят миграторы, которые помогают решить проблемы с версионированием и инициализацией схем БД и тех данных, которые в них сосредоточены.

Основной целью исследования является обзор понятия миграции и одного из популярных миграторов Liquibase.

Архитектурная модель современного web-проекта, как правило, состоит из трёх компонентов:

– клиента;

– сервера приложения;

– сервера БД.

Наиболее детально рассмотрим только последний компонент, на который возлагается очень важная функция – предоставления и хранения данных. Частный случай предполагает у одного экземпляра приложения одну БД или схему, поэтому в качестве примера рассмотрим миграцию БД на одном из серверов БД.

Миграция БД подразумевает переход от одной структуры БД к другой без потери консистентности. Консистентность данных – это согласованность данных друг с другом, целостность данных, а также внутренняя непротиворечивость [4]. Множество всех условий, налагаемых на данные, определяется структурой данных. Миграция подразумевает как прямое изменение схемы БД или данных, так и обратное изменение – откат изменений.

Использование инструментов миграции позволяет разработчику избежать проблем при смене БД. Не составляет никакого труда изменить систему управления базами данных (СУБД), так как такие инструменты подстроятся под диалект выбранной СУБД, используя свой формат описания схем и возможностей загрузки данных (например, при переходе с СУБД Oracle на PostgreSQL). Инструменты, созданные под разные платформы, языки программирования и разработки позволяют без затруднений агрегировать свой функционал в любой проект.

Далее рассмотрим реально пользующийся популярностью у разработчиков инструмент разработки под названием Liquibase, который берёт на себя функциональные возможности мигратора БД. Библиотека Liquibase написана на языке Java и отлично интегрируется с таким знаменитым фреймворком как Spring. Liquibase можно использовать как отдельно так и в сборке с проектом в виде зависимости Maven или Gradle. Если упростить описание, то инструмент представлен в виде jar-файла, который при запуске принимает параметры подключения и путь к файлу описания схемы базы данных в формате XML, JSON или YAML. Пример запуска мигратора представлен в листинге 1.

Листинг 1 – Запуск мигратора через командную строку

```
java -jar liquibase.jar \
      --driver=oracle.jdbc.OracleDriver \
      --classpath=\path\to\classes:jdbcdriver.jar \
      --changeLogFile=com/example/db.changelog.xml \
      --url="jdbc:oracle:thin:@localhost:1521:oracle" \
      --username=scott \
      --password=tiger \
      Update
```

При запуске миграции обычно не возникает никаких сложностей за исключением наличия драйвера необходимого под ту или иную БД. К сожалению, из описания, представленного в листинге 1 сложно представить, насколько облегчает Liquibase работу в создании схемы БД, отслеживания изменений и загрузки в таблицы данных, однако на практике – это так.

В Liquibase представлены ключевые понятия как:

– databaseChangeLog,
– changeset.

Databasechangelog – тег или элемент, который включает в себя наборы changeSet.

Changeset – тег или элемент, описывающий создаваемый при исполнении объект БД (таблица, индекс, грант и т. д.). Пример такого файла представлен в листинге 2.

Листинг 2 – Databasechangelog файл

```
<databaseChangeLog ...>
    <changeSet author="talayko" id="1585407925310-6">
        <createTable tableName="USERS">
            <column name="ID" type="UUID">
                <constraints nullable="false"/>
            </column>
        </createTable>
    </changeSet>
</databaseChangeLog>
```

Тег databaseChangeLog содержит атрибуты подключения схем валидации xsd и один changeSet, который имеет важные атрибуты, определяющие уникальность данного изменения – AUTHOR и ID. Ниже рассмотрим с какой целью должна присутствовать уникальность каждого changeSet. После запуска мигратора в БД создастся таблица USERS с единственным столбцом с именем ID и типом raw(16) – если БД Oracle или UUID – если БД PostgreSQL. В зависимости от выбранного драйвера базы данных используют необходимый диалект Liquibase.

Liquibase создает две технические таблицы в схеме БД:
– DATABASECHANGELOCK,
– DATABASECHANGE.

DATABASECHANGELOCK содержит информацию о общем состоянии миграции. В данной таблице очень важно знать о столбце LOCK, который имеет булевый тип и позволяет понять идёт ли сторонняя миграция в данную таблицу. Текущий процесс миграции будет находиться в состоянии ожидания пока значение LOCK в значении TRUE.

DATABASECHANGE таблица описывающая изменения применённые при миграциях. Liquibase накатывает изменения только те, которые ранее не были записаны в таблицу DATABASECHANGE, а ищет данные изменения по атрибутам AUTHOR и ID, которые были указаны в changeLog. Изменения в ранее применённых changeSet невозможны, мигратор сравнивает check sum MD5 найденных по ID и AUTHOR, который записан в DATABASECHANGE в столбце MD5 и check sum из changeSet файла и если сумма не совпадает, то происходит ошибка миграции. Чтобы избежать данной ошибки, можно создать новый changeSet или удалить строку о старом changeSet из таблицы DATABASECHANGE.

Изучив основные задачи, решаемые мигратором, а также то каким образом он применяет изменения к БД, далее рассмотрим, как можно подключить Liquibase к проекту построенному на основе Spring Boot. Для подключения к проекту необходимо в сборщике проекта указать зависимость или добавить liquibase.jar в lib classpath для дальнейшего использования. Подключение зависимости в pom.xml maven отображено в листинге 3.

Листинг 3 – Зависимость Liquibase в Maven проекте

```
<dependency>
   <groupId>org.liquibase</groupId>
   <artifactId>liquibase-core</artifactId>
</dependency>
```

Далее необходимо прописать местоположение в файле контекста приложения местоположения файла миграции (листинг 4).

Листинг 4 – Property liquibase changelog в файле контекста Spring Boot

```
spring.liquibase.change-log=classpath:/db/changelog/changelog-master.xml
```

Теперь при старте будет происходить миграция схемы и данных, прописанных в файле changelog-master.xml.

Таким образом, рассмотренные основные принципы работы миграторов БД и популярный инструмент миграции Liquibase, а также приведённые примеры его использования в современной разработке программного обеспечения, анализ внутреннего устройства мигратора определили важные составляющие

в виде технических таблиц и принципов определения новых изменений, что позволило осуществить выбор данного инструмента для дальнейшей разработки.

ЛИТЕРАТУРА

1. Официальный сайт Liquibase [Электронный ресурс] / Liquibase | Open source version control for Your Database – Режим доступа: https://www.liquibase.org/get-started/how-liquibase-works – Дата доступа: 14.08.2020.
2. Major functionality [Электронный ресурс] / Liquibase Wikipedia – Режим доступа: https://en.wikipedia.org/wiki/Liquibase – Дата доступа: 15.08.2020.
3. Откат изменений Liquibase [Электронный ресурс] / Версионирование структуры БД с помощью Liquibase – Режим доступа: http://easy-code.ru/lesson/database-versioning-liquibase – Дата доступа: 15.08.2020.
4. Консистентность данных [Электронный ресурс] / Консистентность данных – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/147082 - Дата доступа: 15.08.2020.