

Разработка серверного ПО. Лекция 5. Миграции схемы базы данных с Liquibase

Большинство приложений используют SQL-базы данных для хранения информации. Такие приложения обычно разворачиваются на нескольких средах: среде разработки, предрелизной (пре-прод) и продакшене. При этом над одним проектом одновременно работает команда разработчиков, что создаёт дополнительную сложность при поддержании единой схемы базы данных.

Корпоративные приложения часто сталкиваются с проблемами, связанными с синхронизацией схем базы данных между различными окружениями и разработчиками. Важно не только передавать изменения схемы, но и делать это так, чтобы избежать конфликтов.

Эту задачу решает система управления миграциями Liquibase. Это система контроля версий для базы данных, которая не зависит от конкретного типа базы. Liquibase поддерживает множество баз данных, таких как DB2, Apache Derby, MySQL, PostgreSQL, Oracle, Microsoft® SQL Server и другие.

Существуют и другие системы управления миграциями: Doctrine 2 Migrations, Rails AR Migrations, DBDeploy и другие. Однако некоторые из них зависят от платформы, а другие уступают Liquibase по функциональности.

Как работает Liquibase

Liquibase — это кроссплатформенное Java-приложение. Вы можете скачать его в виде JAR файла и запускать на Windows, Mac или Linux.

Все изменения для базы данных описываются в формате, понятном Liquibase, а затем он переводит их в SQL-запросы для выбранной базы данных. Это обеспечивает независимость от конкретной СУБД.

ChangeLog

Изменения структуры базы данных записываются в файлы под названием changeLog. Эти файлы могут быть в одном из следующих форматов: XML, YAML, JSON или SQL.

Файлы changeLog могут включать друг в друга другие файлы, что облегчает управление изменениями при большом объеме данных. Примеры будут приведены ниже.

ChangeSet

ChangeSet — это аналог коммита в системах контроля версий, таких как Git.

Каждый changeSet имеет уникальный составной идентификатор, состоящий из полей id, author и filename. В одном changeSet может быть одно или несколько изменений базы данных. Однако хорошей практикой считается использование одного изменения на каждый changeSet.

```
<changeSet id="1" author="author">
  <createTable tableName="example_table">
```

```

      <column name="id" type="int"/>
      <column name="name" type="varchar(255)"/>
    </createTable>
  </changeSet>

```

При первом запуске Liquibase создаёт две специальные таблицы для контроля миграций:

databasechangelog – содержит список уже выполненных изменений схемы БД (changeSet).

`databasechangelock` – Используется для блокировки на время работы, чтобы гарантировать одновременную работу только одного экземпляра Liquibase.

Блокировка

Если несколько экземпляров Liquibase будут выполняться одновременно с одной и той же базой данных, это приведет к конфликтам. Такие ситуации могут возникнуть, когда несколько разработчиков работают с одной и той же базой данных или в кластере несколько экземпляров автоматически запускают Liquibase при старте.

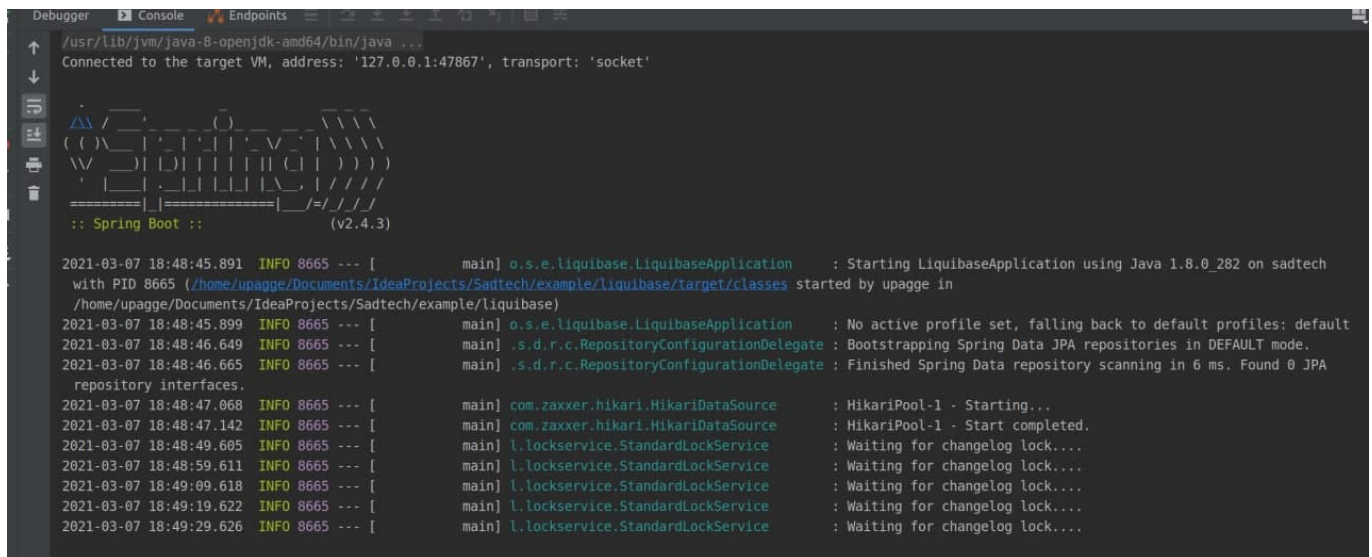
Для защиты от таких ситуаций Liquibase создает таблицу `databasechangelock`, в которой есть булево поле `locked`. При запуске Liquibase проверяет значение этого поля, и если оно установлено в `true`, приложение ожидает, пока оно не сменится на `false`.

В случае экстренной остановки программы в начале её работы может возникнуть проблема, при которой Liquibase успевает установить флаг в true, но не переключает его обратно на false.

Например, если вы запускаете Liquibase как исполняемый файл, приложение может "зависнуть".

[illegible]

В логах Spring Boot приложения это можно увидеть наглядно.

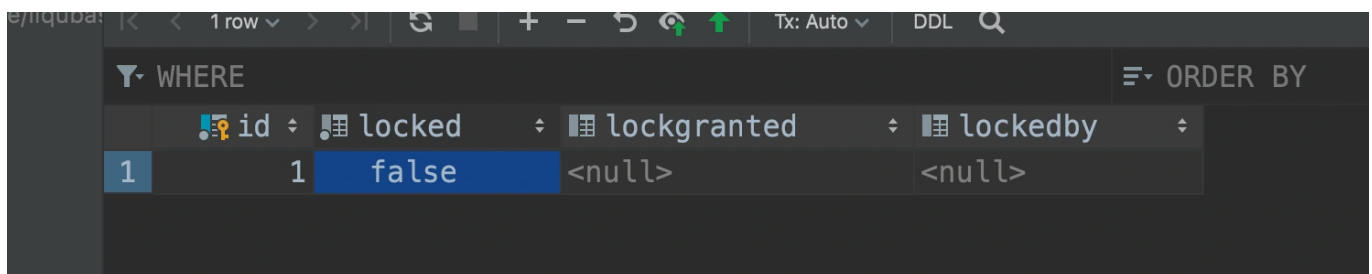


```
Debugger Console Endpoints
/usr/lib/jvm/java-8-openjdk-amd64/bin/java ...
Connected to the target VM, address: '127.0.0.1:47867', transport: 'socket'

:: Spring Boot ::
(v2.4.3)

2021-03-07 18:48:45.891 INFO 8665 --- [main] o.s.e.liquibase.LiquibaseApplication : Starting LiquibaseApplication using Java 1.8.0_282 on sadtech
with PID 8665 (/home/upagge/Documents/IdeaProjects/Sadtech/example/liquibase/target/classes started by upagge in
/home/upagge/Documents/IdeaProjects/Sadtech/example/liquibase)
2021-03-07 18:48:45.899 INFO 8665 --- [main] o.s.e.liquibase.LiquibaseApplication : No active profile set, falling back to default profiles: default
2021-03-07 18:48:46.649 INFO 8665 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2021-03-07 18:48:46.665 INFO 8665 --- [main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 6 ms. Found 0 JPA
repository interfaces.
2021-03-07 18:48:47.068 INFO 8665 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2021-03-07 18:48:47.142 INFO 8665 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2021-03-07 18:48:49.605 INFO 8665 --- [main] l.lockservice.StandardLockService : Waiting for changelog lock....
2021-03-07 18:48:49.611 INFO 8665 --- [main] l.lockservice.StandardLockService : Waiting for changelog lock....
2021-03-07 18:49:09.618 INFO 8665 --- [main] l.lockservice.StandardLockService : Waiting for changelog lock....
2021-03-07 18:49:19.622 INFO 8665 --- [main] l.lockservice.StandardLockService : Waiting for changelog lock....
2021-03-07 18:49:29.626 INFO 8665 --- [main] l.lockservice.StandardLockService : Waiting for changelog lock....
```

Чтобы исправить эту проблему, нужно вручную изменить значение поля locked в таблице databasechangelock на false.



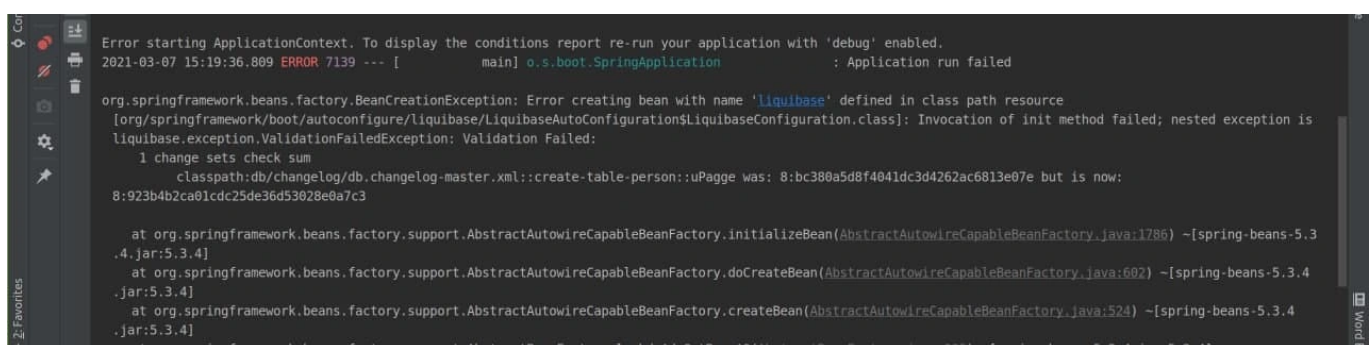
| id | locked | lockgranted | lockedby |
|----|--------|-------------|----------|
| 1 | false | <null> | <null> |

Контрольная сумма

Liquibase считывает основной changeLog и проверяет, какие изменения уже были применены, а какие необходимо выполнить.

После выполнения changeSet в таблицу databasechangelog записывается контрольная сумма (MD5-хэш) этого changeSet. Хэш рассчитывается на основе нормализованного содержимого XML-файла.

При следующем запуске Liquibase пересчитывает контрольные суммы для всех changeSet и сравнивает их с сохранёнными значениями в таблице. Если вы изменили уже выполненный changeSet, новая контрольная сумма не будет совпадать с сохранённой, и приложение завершится с ошибкой при запуске.



```
Error starting ApplicationContext. To display the conditions report re-run your application with 'debug' enabled.
2021-03-07 15:19:36.809 ERROR 7139 --- [main] o.s.boot.SpringApplication : Application run failed

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'liquibase' defined in class path resource
[org.springframework.boot.autoconfigure.liquibase.LiquibaseAutoConfiguration$LiquibaseConfiguration.class]: Invocation of init method failed; nested exception is
liquibase.exception.ValidationFailedException: Validation Failed:
1 change sets check sum
classpath:db/changelog/db.changelog-master.xml::create-table-person::uPagge was: 8:bc380a5d8f4041dc3d4262ac6813e07e but is now:
8:923b4b2ca01cdc25de36d53028e0a7c3
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.initializeBean(AbstractAutowireCapableBeanFactory.java:1786) ~[spring-beans-5.3.4
.jar:5.3.4]
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.doCreateBean(AbstractAutowireCapableBeanFactory.java:602) ~[spring-beans-5.3.4
.jar:5.3.4]
at org.springframework.beans.factory.support.AbstractAutowireCapableBeanFactory.createBean(AbstractAutowireCapableBeanFactory.java:524) ~[spring-beans-5.3.4
.jar:5.3.4]
at org.springframework.beans.factory.support.AbstractBeanFactory.lambda$doGetBean$0(AbstractBeanFactory.java:331) ~[spring-beans-5.3.4.jar:5.3.4]
```

Организация скриптов

Прежде чем запускать Liquibase, нужно создать главный changeLog файл и добавить в него несколько changeSet, чтобы было что применять. Для примера создадим простую таблицу person.

Создайте папку db, а внутри неё папку changelog. В папке changelog создайте файл db.changelog-master.xml. Это будет ваш главный changeLog файл, который будет ссылаться на другие changeLog файлы.

Начальное содержимое главного файла changeLog:

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">

    // Здесь будут ссылки на другие changeLog файлы

</databaseChangeLog>
```

Хорошей практикой считается создание множества отдельных changeLog файлов и их включение в основной файл с помощью тега `<include>`. Это позволяет избежать перегрузки одного большого файла, так как со временем количество changeSet значительно увеличится.

Пример организации

Допустим, у нас следующая версия приложения 1.0.0. В таком случае:

Создайте в папке db/changelog новую папку v.1.0.0, которая будет содержать изменения для этой версии. В папке v.1.0.0 создайте файл changelog.xml, в котором будут находиться changeSet для этой версии. Когда версия приложения изменится, вы создадите новую папку (например, v.2.0.0) и разместите в ней новый changeLog файл. Все эти версии файлов changeLog подключаются в главный файл db.changelog-master.xml.

Пример содержимого основного changeLog файла с подключением версии 1.0.0:

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">

    <include file="v.1.0.0/changelog.xml" relativeToChangelogFile="true"/>

</databaseChangeLog>
```

Тег фактически вставляет содержимое указанного файла в место его вызова в основном changeLog. А атрибут `relativeToChangelogFile="true"` указывает, что путь до файла указывается относительно местоположения текущего changeLog файла.

Создание таблицы

Теперь создадим таблицу `person`. Для этого в папке `v.1.0.0` создадим новый файл `create-table.xml`, который будет содержать миграцию:

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">

    <changeSet id="create-table-person" author="uPagge">
        <createTable tableName="person">
            <column name="id" type="int" autoIncrement="true">
                <constraints nullable="false" primaryKey="true"/>
            </column>
            <column name="first_name" type="varchar(64)"/>
        </createTable>
    </changeSet>

</databaseChangeLog>
```

Тег задает имя новой таблицы через параметр `tableName`. Внутри этого тега перечислены колонки таблицы. Для каждой колонки необходимо указать её тип, который Liquibase автоматически приведет к нужному формату для используемой базы данных.

Особое внимание стоит уделить колонке `id`. Для неё задан автоинкремент, а также через тег указаны следующие ограничения:

- `primaryKey="true"` — колонка является первичным ключом.
- `nullable="false"` — значения не могут быть NULL. Хотя это условие не обязательно при использовании первичного ключа, его отсутствие может вызывать ошибки при использовании базы данных H2 для тестов.

Теперь в файле `changelog.xml` добавим ссылку на новый changeLog файл с помощью тега :

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
    xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">

    <changeSet id="add-tag-1.0.0" author="uPagge">
```

```
<tagDatabase tag="v.1.0.0"/>
</changeSet>

<include file="create-table.xml" relativeToChangelogFile="true"/>

</databaseChangeLog>
```

Запуск Liquibase

Liquibase можно запускать несколькими способами, в зависимости от контекста:

Запуск с помощью исполняемого файла. Запуск через Docker-образ. Запуск при старте Spring Boot или Quarkus-сервиса.

Запуск исполняемого файла

[Скачать](#)

Распакуйте скачанный архив, в котором вы найдете папку liquibase-4.29.2. Она содержит несколько важных папок и файлов:

- internal/lib – папка с зависимостями, необходимыми для работы liquibase, включая драйверы баз данных.
- liquibase – бинарный файл для запуска в среде Linux/macOS.
- liquibase.bat – bat-файл для запуска в среде Windows.
- examples – папка с примерами скриптов миграций (мы будем использовать свои скрипты).

После распаковки скопируйте папку db, созданную ранее, в папку liquibase-4.29.2.

Файл liquibase.properties

Файл liquibase.properties — это конфигурационный файл, который упрощает запуск Liquibase. Вместо того чтобы передавать параметры подключения и другие настройки через командную строку с множеством флагов, можно задать все необходимые параметры в этом файле.

Создадим файл liquibase.properties в папке liquibase-4.29.2.

```
url=jdbc:postgresql://localhost:5432/liquibase_example
username=postgres
password=your_pass
changeLogFile=db/changelog/db.changelog-master.xml
liquibase.hub.mode=off
```

Описание параметров:

- url — URL соединения с базой данных. В примере используется PostgreSQL, но нужно подставить URL вашей базы данных.
- username — имя пользователя для подключения к базе данных.
- password — пароль пользователя базы данных.

- `changeLogFile` — путь к основному файлу `changeLog`, где описаны все миграции.
- `liquibase.hub.mode=off` — отключает интеграцию с Liquibase Hub, если она не используется.

После добавления файла `liquibase.properties` ваша структура проекта может выглядеть следующим образом:

```

.
├── liquibase-4.29.2
│   ├── db
│   │   └── changelog
│   │       ├── db.changelog-master.xml
│   │       └── v.1.0.0
│   │           ├── create-table.xml
│   │           └── cumulative-changelog.xml
│   ├── internal
│   │   └── lib
│   │       ├── liquibase-core.jar
│   │       └── postgresql.jar
│   ├── liquibase
│   ├── liquibase.bat
│   └── liquibase.properties

```

Для запуска миграции, находясь в папке `liquibase-4.29.2`, выполните следующую команду:

```
./liquibase update
```

В случае успеха в логе будет следующее сообщение:

```

#####
Starting Liquibase at 23:10:35 (version 4.9.1 #1978 built at 2022-03-28 19:39+0000)
Liquibase Version: 4.9.1
Liquibase Community 4.9.1 by Liquibase
Running Changeset: db/changelog/v.1.0.0/cumulative-changelog.xml::add-tag-1.0.0::uPagge
Running Changeset: db/changelog/v.1.0.0/create-table.xml::create-table-person::uPagge
Liquibase command 'update' was executed successfully.
upagge@MacBook-Pro-Mark liquibase-4.9.1 %

```

Обновления в структуре и установке Liquibase

С выходом версии 4.11.0 произошли важные изменения в установке и организации файлов Liquibase. Вот ключевые моменты:

- Основной JAR-файл: В папке `internal/lib` находится файл `liquibase-core.jar`, который отвечает за все основные функции работы Liquibase. Раньше он был рядом с исполняемым файлом в корне папки.
- Изменение структуры библиотек: В новой версии библиотеки и драйверы БД, необходимые для работы Liquibase, перемещены из папки `lib` в новую директорию `internal/lib`.

Запуск с помощью докера

Запуск Liquibase через Docker — это простой и удобный способ выполнения скриптов миграции без необходимости установки Liquibase локально. Рассмотрим, как это сделать с помощью небольшого скрипта на Shell.

```
#!/bin/sh
docker run --name liquibase --network host -v
YOUR_PATH_TO_CHANGELOG/db/changelog:/liquibase/db/changelog
liquibase/liquibase:4.29.2 --
defaultsFile=/liquibase/db/changelog/liquibase.properties update
docker rm liquibase
```

Не забудьте заменить YOUR_PATH_TO_CHANGELOG на фактический путь к вашей папке с файлами миграций. В этой папке должны находиться ваши changeLog файлы и файл liquibase.properties, который вы создавали ранее для запуска через исполняемый файл.

При запуске SpringBoot приложения

Для добавления поддержки Liquibase в Spring Boot, вам нужно указать несколько зависимостей в вашем проекте, а также настроить подключение к базе данных в файле application.yml.

В файле application.yml укажите параметры для подключения к базе данных и путь к файлу с миграциями:

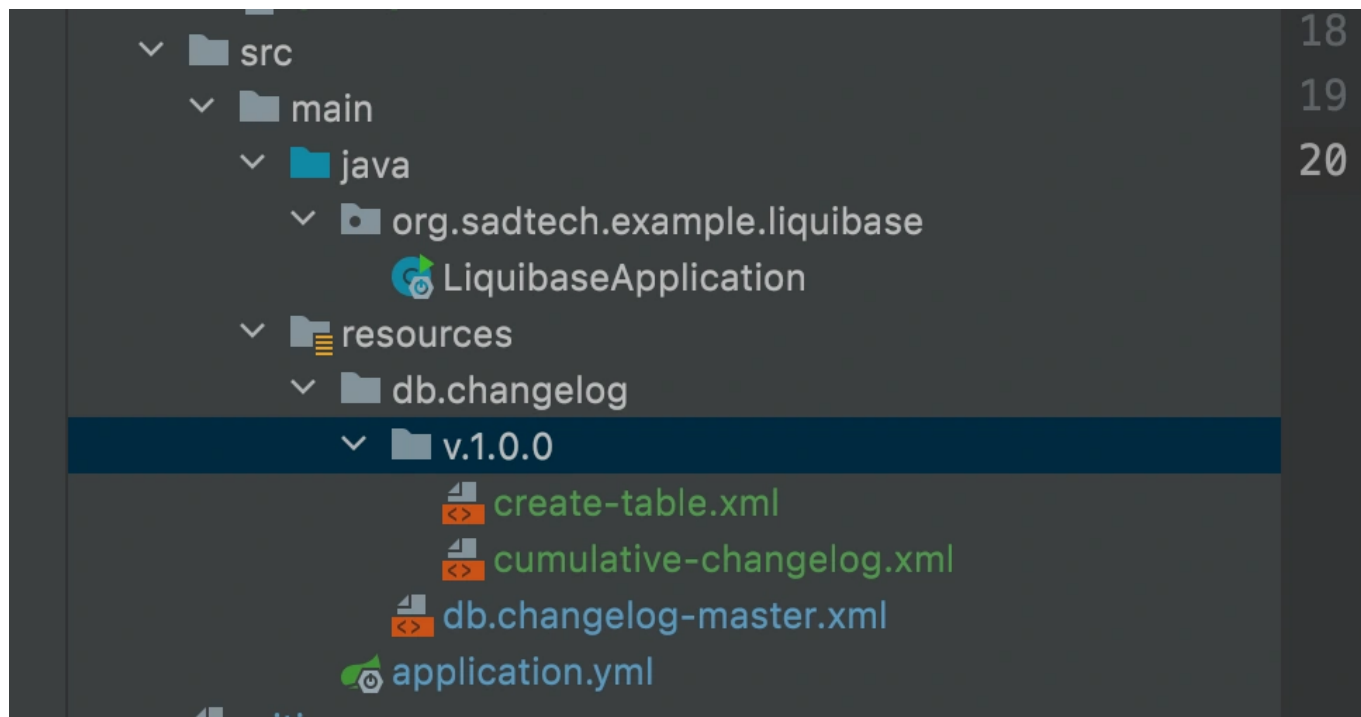
```
spring:
  datasource:
    url: jdbc:postgresql://localhost:5432/liquibase_example
    username: postgres
    driver-class-name: org.postgresql.Driver
    password: pass
  liquibase:
    change-log: classpath:db/changelog/db.changelog-master.xml
```

Отключение генерации Hibernate

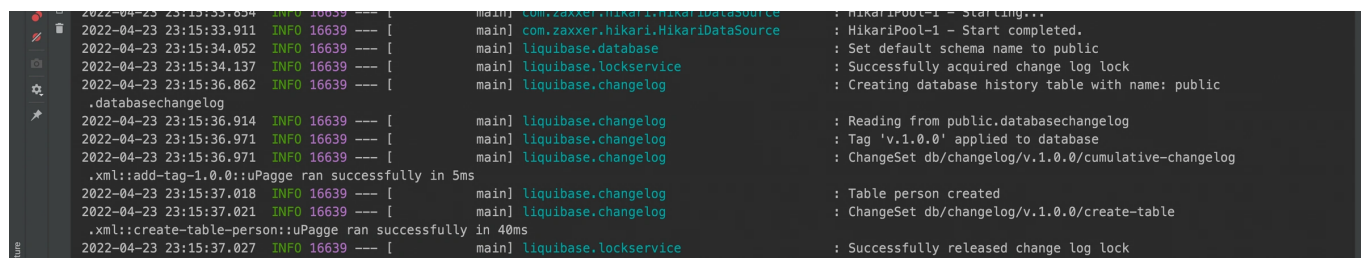
Если вы используете Hibernate, не забудьте отключить автоматическое создание схемы базы данных, чтобы избежать конфликтов с Liquibase. Для этого в application.yml можно добавить следующие строки:

```
spring:
  jpa:
    hibernate:
      ddl-auto: none
```

Для корректной работы Liquibase скопируйте папку db с файлами миграций в папку resources, чтобы Liquibase мог их найти в пути classpath. В итоге структура проекта будет выглядеть следующим образом:

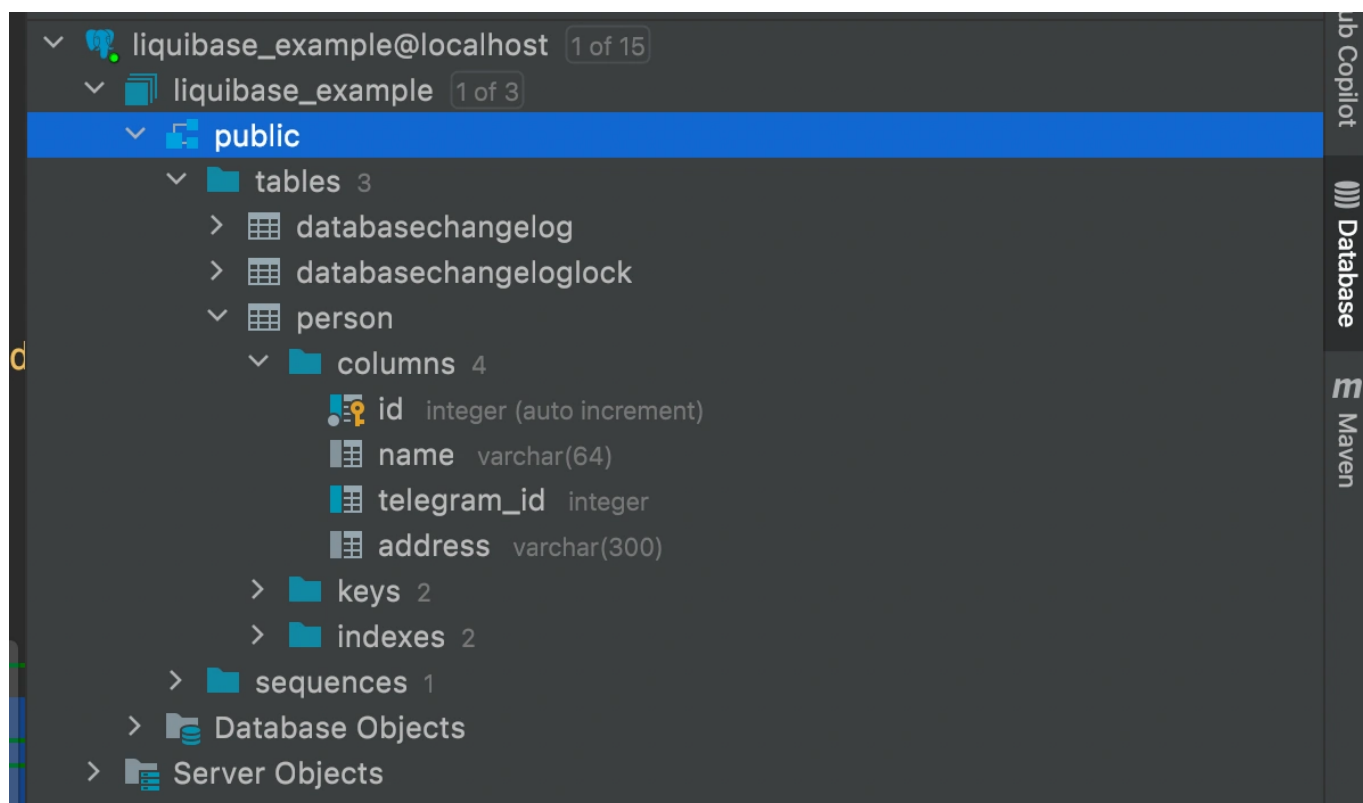


После настройки достаточно запустить Spring Boot приложение. Во время старта в логах появятся сообщения от Liquibase о применении миграций. В результате должна быть создана таблица `person`.



Результат

Во всех трех случаях мы должны были получить нашу таблицу `person`:



Давайте рассмотрим содержимое технической таблицы databasechangelog, которая хранит информацию о выполненных миграциях. Эта таблица выполняет функцию журнала изменений, аналогично git log, фиксируя все применённые changeSet.

| WHERE | | ORDER BY dateexecute | |
|---------------|--------|---|--------|
| 1 | | | |
| id | author | add-tag-1.0.0 | uPagge |
| filename | | db/changelog/v.1.0.0/cumulative-changelog.xml | |
| dateexecuted | | 2022-04-23 20:36:52.167000 | |
| orderexecuted | | 1 | |
| exectype | | EXECUTED | |
| md5sum | | 8:852ffa39cbea3a96754f9f1daa017694 | |
| description | | tagDatabase | |
| comments | | | |
| tag | | v.1.0.0 | |
| liquibase | | 4.9.1 | |
| contexts | | <null> | |
| labels | | <null> | |
| deployment_id | | 0746211894 | |
| ctid | | (0,1) | |
| 2 | | | |
| id | author | create-table-person | uPagge |
| filename | | db/changelog/v.1.0.0/create-table.xml | |
| dateexecuted | | 2022-04-23 20:36:52.228064 | |
| orderexecuted | | 2 | |
| exectype | | EXECUTED | |
| md5sum | | 8:923b4b2ca01cdc25de36d53028e0a7c3 | |
| description | | createTable tableName=person | |
| comments | | | |
| tag | | <null> | |
| liquibase | | 4.9.1 | |
| contexts | | <null> | |
| labels | | <null> | |
| deployment_id | | 0746211894 | |
| ctid | | (0,2) | |

В нашем случае таблица содержит две записи, так как мы выполнили два `changeSet`:

1. Добавление тега с помощью команды `tagDatabase`.
2. Создание таблицы `person`.

Каждое поле в таблице логично отражает важные данные о выполненных миграциях, такие как идентификатор, автор, путь к файлу и контрольная сумма.

Скрипты миграций

Разберём основные миграционные скрипты, которые вы будете использовать в Liquibase.

Добавление колонки в таблицу

Попробуем добавить новую колонку в уже существующую таблицу `person`. Частой ошибкой является попытка изменить старый `changeSet`, как показано в примере:

```
<changeSet id="create-table-person" author="uPagge">
  <createTable tableName="person">
    <column name="id" type="int" autoIncrement="true">
      <constraints nullable="false" primaryKey="true"/>
    </column>
    <column name="first_name" type="varchar(64)"/>
    <column name="address" type="varchar(300)"/>
  </createTable>
</changeSet>
```

При повторном запуске миграции это вызовет ошибку.

```
#####
Starting Liquibase at 21:14:16 (version 4.9.1 #1978 built at 2022-03-28 19:39+0000)
Liquibase Version: 4.9.1
Liquibase Community 4.9.1 by Liquibase

Unexpected error running Liquibase: Validation Failed:
  1 change sets check sum
    db/changelog/v.1.0.0/create-table.xml::create-table-person::uPagge was: 8:bc380a5d8f4041dc3d4262ac6813e07e but is now: 8:923b4b2ca01cdc25de36d53028e0a7c3

For more information, please use the --log-level flag
```

В Liquibase, если `changeSet` уже выполнен и записан в таблице `databasechangelog`, его изменение недопустимо — это похоже на невозможность изменить уже опубликованный коммит в Git.

В этом случае у вас три пути:

- Создать новый `changeSet` с изменениями (рекомендуемый способ).
- Откатить изменения с помощью команды Liquibase.
- Удалить запись о выполненном `changeSet` из `databasechangelog`, но это небезопасно для сред, где миграция уже была выполнена, и лучше использовать этот метод только на локальной машине.

В нашем случае правильное решение — вернуть оригинальный `changeSet` в исходное состояние и создать новый для добавления колонки:

```
<changeSet id="create-table-person" author="uPagge">
  <createTable tableName="person">
    <column name="id" type="int" autoIncrement="true">
      <constraints nullable="false" primaryKey="true"/>
    </column>
    <column name="name" type="varchar(64)"/>
  </createTable>
</changeSet>

<changeSet id="add-new-column-address" author="uPagge">
  <addColumn tableName="person">
    <column name="address" type="varchar(300)"/>
  </addColumn>
</changeSet>
```

Теперь, запустив миграцию, вы успешно добавите новую колонку без ошибок.

Связь с другой таблицей

Связи между таблицами — частое явление в реляционных базах данных. Для примера добавим таблицу book и свяжем её с таблицей person, указав внешний ключ:

```
<changeSet id="create-table-book" author="uPagge">
  <createTable tableName="book">
    <column name="id" type="int" autoIncrement="true">
      <constraints nullable="false" primaryKey="true"/>
    </column>
    <column name="name" type="varchar(64)"/>
    <column name="author_id" type="int">
      <constraints foreignKeyName="book_author_id_person_id"
references="person(id)"/>
    </column>
  </createTable>
</changeSet>
```

В данном примере колонка author_id связана с колонкой id в таблице person. Важно указать уникальное имя для внешнего ключа — здесь используется формат book_author_id_person_id, который помогает лучше организовать имена ключей.

Также можно добавить каскадное удаление:

```
<constraints foreignKeyName="book_author_id_person_id" references="person(id)"
deleteCascade="true"/>
```

Теперь, если автор будет удалён из таблицы person, соответствующая запись в таблице book также будет удалена.

Если нужно реализовать каскадное обновление, это делается с помощью команды `addForeignKeyConstraint`:

```
<changeSet id="create-table-book" author="uPagge">
  <createTable tableName="book">
    <column name="id" type="int" autoIncrement="true">
      <constraints nullable="false" primaryKey="true"/>
    </column>
    <column name="name" type="varchar(64)"/>
    <column name="author_id" type="int"/>
  </createTable>

  <addForeignKeyConstraint baseTableName="book" baseColumnNames="author_id"
    constraintName="book_author_id_person_id"
    referencedTableName="person"
    referencedColumnNames="id" onUpdate="CASCADE"/>
</changeSet>
```

В данном случае при обновлении значения `id` в таблице `person` соответствующие значения в `author_id` таблицы `book` также будут обновлены.

Добавление индекса по внешнему ключу

После создания связи между таблицами, вы можете добавить индекс для ускорения поиска по внешнему ключу. Это помогает базе данных быстрее обрабатывать запросы, которые используют связь между таблицами.

Вот пример, как можно создать индекс по внешнему ключу в таблице `book`:

```
<changeSet id="create-index-author-id" author="uPagge">
  <createIndex indexName="idx_book_author_id" tableName="book">
    <column name="author_id"/>
  </createIndex>
</changeSet>
```

Создание представления

Несмотря на то, что большинство миграций в Liquibase вы, возможно, уже привыкли создавать с помощью XML, для создания представлений (views) зачастую лучше подходит использование SQL-запросов.

```
<changeSet id="create-view-book-author" author="uPagge">
  <createView viewName="author_and_book">
    SELECT p.id as person_id,
           p.first_name as person_first_name,
           b.id as book_id,
           b.name as book_name
```

```
FROM person p
      LEFT JOIN book b on p.id = b.author_id
</createView>
</changeSet>
```

Изменение типа колонки

Иногда требуется изменить тип существующей колонки в таблице. Это распространённая операция, когда данные нуждаются в более гибком или точном формате.

```
<changeSet id="modify-column-type" author="uPagge">
  <modifyDataType tableName="person" columnName="first_name"
newDataType="varchar(128)"/>
</changeSet>
```

Удаление колонки

```
<changeSet id="drop-column" author="uPagge">
  <dropColumn tableName="person" columnName="address"/>
</changeSet>
```

Удаление таблицы

```
<changeSet id="drop-table" author="uPagge">
  <dropTable tableName="book"/>
</changeSet>
```

Добавление уникального ограничения

```
<changeSet id="add-unique-constraint" author="uPagge">
  <addUniqueConstraint columnNames="email" tableName="person"
constraintName="unique_person_email"/>
</changeSet>
```

Заполнение таблицы начальными данными

```
<changeSet id="insert-initial-data" author="uPagge">
  <insert tableName="person">
    <column name="id" value="1"/>
    <column name="first_name" value="John"/>
    <column name="last_name" value="Doe"/>
  </insert>
</changeSet>
```

```
</insert>  
</changeSet>
```

Откат изменений

Теперь, когда мы разобрались с запуском Liquibase и созданием базовых миграций, давайте рассмотрим откат изменений (rollback).

На практике откат миграций используется редко, особенно в приложениях на Spring Boot, который не поддерживает откат изменений через стандартный механизм. Если нужно откатить миграцию, вам придётся воспользоваться либо исполняемым файлом Liquibase, либо Docker-образом. В этом разделе мы рассмотрим использование отката с помощью исполняемого файла.

Liquibase автоматически может откатить многие изменения, такие как: создание таблицы, добавление колонки, добавление внешнего ключа. Для некоторых changeSet необходимо написать скрипты отката.

Мы начнём с примеров, которые Liquibase может откатить самостоятельно, а затем перейдём к более сложным ситуациям, где потребуется ручное написание отката.

Представим, что наш сервис обновляется до версии 2.0.0. Для этого создадим новую папку v.2.0.0 и добавим туда новый changeLog файл с изменениями:

cumulative-changelog.xml:

```
<?xml version="1.0" encoding="UTF-8"?>  
<databaseChangeLog  
  xmlns="http://www.liquibase.org/xml/ns/dbchangelog"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog  
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">  
  
  <changeSet id="add-tag-2.0.0" author="uPagge">  
    <tagDatabase tag="v.2.0.0"/>  
  </changeSet>  
  
  <include file="create-table-hero.xml" relativeToChangelogFile="true"/>  
  
</databaseChangeLog>
```

create-table-hero.xml:

```
<?xml version="1.0" encoding="UTF-8"?>  
<databaseChangeLog  
  xmlns="http://www.liquibase.org/xml/ns/dbchangelog"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog  
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">  
  
  <changeSet id="create-table-hero" author="uPagge">
```

```

    <createTable tableName="hero">
      <column name="id" type="int" autoIncrement="true">
        <constraints nullable="false" primaryKey="true"/>
      </column>
      <column name="book_id" type="int">
        <constraints nullable="false"/>
      </column>
      <column name="name" type="varchar(64)">
        <constraints nullable="false"/>
      </column>
    </createTable>
  </changeSet>

  <changeSet id="create-fk" author="uPagge">
    <addForeignKeyConstraint baseTableName="hero" baseColumnNames="book_id"
      constraintName="hero_book_id"
      referencedTableName="book"
      referencedColumnNames="id"
      deleteCascade="true"/>
  </changeSet>

</databaseChangeLog>

```

db.changelog-master.xml:

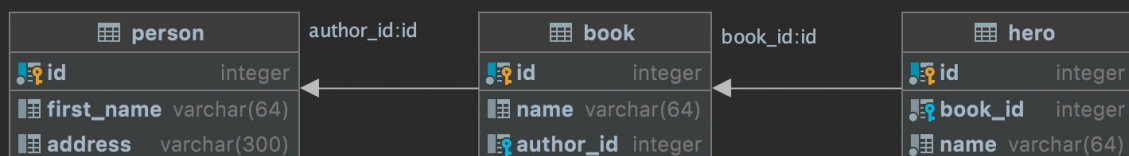
```

<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
  xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.6.xsd">

  <include file="v.1.0.0/changelog.xml" relativeToChangelogFile="true"/>
  <include file="v.2.0.0/changelog.xml" relativeToChangelogFile="true"/>

</databaseChangeLog>

```



Посмотрим содержимое таблицы databasechangelog, которая сохраняет историю выполнений changeLog. Для удобства я скрыл пустые колонки.

| id | filename | md5sum | dateexecuted | author | ex... | tag | liquibase |
|----|------------------------|---|--------------|----------------------------|--------|------------|---------------|
| 1 | add-tag-1.0.0 | db/changelog/v.1.0.0/cumulative-changelog.xml | 8:852ffa3... | 2022-04-24 00:13:34.958396 | uPagge | 1 EXECUTED | v.1.0.0 4.9.1 |
| 2 | create-table-person | db/changelog/v.1.0.0/create-table.xml | 8:815ba35... | 2022-04-24 00:13:34.999229 | uPagge | 2 EXECUTED | <null> 4.9.1 |
| 3 | add-new-column-address | db/changelog/v.1.0.0/create-table.xml | 8:de19fc8... | 2022-04-24 00:13:35.004594 | uPagge | 3 EXECUTED | <null> 4.9.1 |
| 4 | create-table-book | db/changelog/v.1.0.0/create-table.xml | 8:58bba56... | 2022-04-24 00:13:35.047546 | uPagge | 4 EXECUTED | <null> 4.9.1 |
| 5 | add-tag-2.0.0 | db/changelog/v.2.0.0/cumulative-changelog.xml | 8:22a4642... | 2022-04-24 00:14:13.829358 | uPagge | 5 EXECUTED | v.2.0.0 4.9.1 |
| 6 | create-table-hero | db/changelog/v.2.0.0/create-table-hero.xml | 8:84212fc... | 2022-04-24 00:14:13.866945 | uPagge | 6 EXECUTED | <null> 4.9.1 |
| 7 | create-fk | db/changelog/v.2.0.0/create-table-hero.xml | 8:a98048c... | 2022-04-24 00:14:13.879173 | uPagge | 7 EXECUTED | <null> 4.9.1 |

Откат rollbackCount

Команда `rollbackCount` позволяет откатить определённое количество выполненных `changeSet`.

Например, если вы хотите откатить только последнее изменение, достаточно указать `rollbackCount 1`:

```
./liquibase rollbackCount 1
```

В этом случае Liquibase удалит последнее выполненное изменение, например, если последним было добавление связи между таблицами `hero` и `book`, эта связь будет удалена. Кроме того, запись о выполнении этого `changeSet` будет удалена из таблицы `databasechangelog`, что выглядит так, как будто это изменение никогда не применялось.

| id | filename | md5sum | dateexecuted | author | ex... | tag | liquibase |
|----|------------------------|---|--------------|----------------------------|--------|------------|---------------|
| 1 | add-tag-1.0.0 | db/changelog/v.1.0.0/cumulative-changelog.xml | 8:852ffa3... | 2022-04-24 00:13:34.958396 | uPagge | 1 EXECUTED | v.1.0.0 4.9.1 |
| 2 | create-table-person | db/changelog/v.1.0.0/create-table.xml | 8:815ba35... | 2022-04-24 00:13:34.999229 | uPagge | 2 EXECUTED | <null> 4.9.1 |
| 3 | add-new-column-address | db/changelog/v.1.0.0/create-table.xml | 8:de19fc8... | 2022-04-24 00:13:35.004594 | uPagge | 3 EXECUTED | <null> 4.9.1 |
| 4 | create-table-book | db/changelog/v.1.0.0/create-table.xml | 8:58bba56... | 2022-04-24 00:13:35.047546 | uPagge | 4 EXECUTED | <null> 4.9.1 |
| 5 | add-tag-2.0.0 | db/changelog/v.2.0.0/cumulative-changelog.xml | 8:22a4642... | 2022-04-24 00:14:13.829358 | uPagge | 5 EXECUTED | v.2.0.0 4.9.1 |
| 6 | create-table-hero | db/changelog/v.2.0.0/create-table-hero.xml | 8:84212fc... | 2022-04-24 00:14:13.866945 | uPagge | 6 EXECUTED | <null> 4.9.1 |

После этого вы можете повторно запустить команду `update`, и связь между таблицами будет восстановлена.

Откат rollback tag

Откат по счётчику иногда неудобен, особенно когда нужно вернуться на более старую версию базы данных, а количество изменений, которые нужно отменить, неизвестно или слишком велико. В таких случаях удобно использовать откат до определённого тега.

Теги создаются с помощью команды `tagDatabase` в каждом релизе `changeLog`.

Этот подход позволяет откатить все изменения, сделанные после указанного тега, включая саму запись о создании тега. Например, если нужно откатиться до версии `v.2.0.0`, команда будет выглядеть так:

```
./liquibase rollback v.2.0.0
```

В этом случае будут отменены все изменения, выполненные после тега `v.2.0.0`, включая:

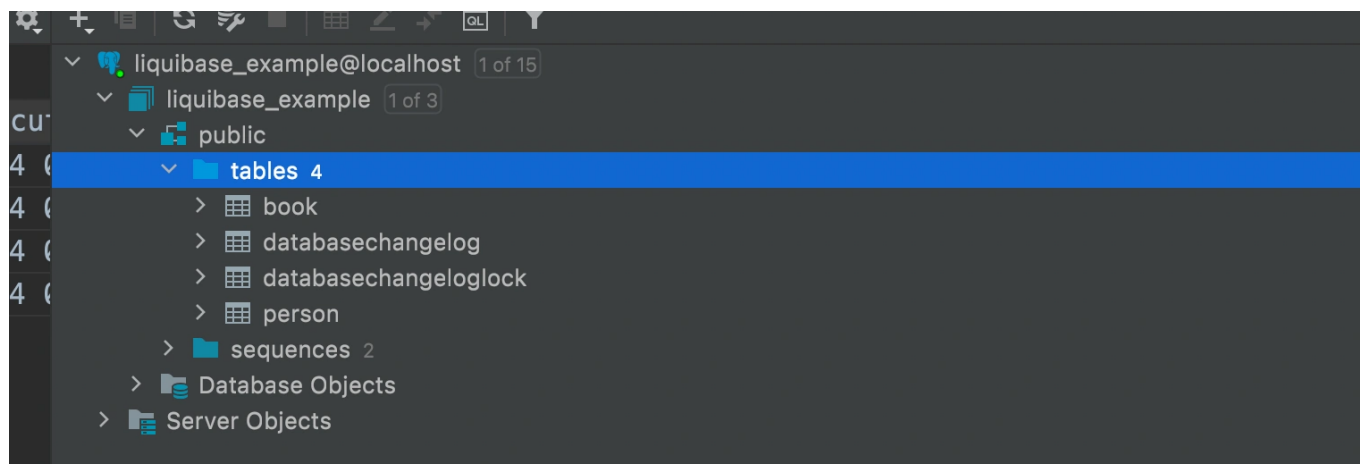
- Связь таблиц `hero` и `book`.
- Создание таблицы `hero`.
- Запись о создании тега `v.2.0.0`.

```
./liquibase rollback v.2.0.0
```

Смотрим результат, действительно все изменения, включая создание тега v.2.0.0 были отменены.

| id | filename | md5sum | dateexecuted | author | ex... | tag | liquibase |
|----|------------------------|---|--------------|----------------------------|--------|------------|---------------|
| 1 | add-tag-1.0.0 | db/changelog/v.1.0.0/cumulative-changelog.xml | 8:852ffa3... | 2022-04-24 00:13:34.958396 | uPagge | 1 EXECUTED | v.1.0.0 4.9.1 |
| 2 | create-table-person | db/changelog/v.1.0.0/create-table.xml | 8:815ba35... | 2022-04-24 00:13:34.999229 | uPagge | 2 EXECUTED | <null> 4.9.1 |
| 3 | add-new-column-address | db/changelog/v.1.0.0/create-table.xml | 8:de19fc8... | 2022-04-24 00:13:35.004594 | uPagge | 3 EXECUTED | <null> 4.9.1 |
| 4 | create-table-book | db/changelog/v.1.0.0/create-table.xml | 8:58bba56... | 2022-04-24 00:13:35.047546 | uPagge | 4 EXECUTED | <null> 4.9.1 |

Таблицы hero также больше не существует:



Ручные rollBack

Операции, связанные с вставкой данных, не откатываются автоматически. Например, если вы добавите записи в таблицу с помощью `changeSet`, их откат потребует явного описания.

Использование Liquibase для миграции данных не рекомендуется, так как его основная задача — управление схемой базы данных. Однако в некоторых случаях вставка данных бывает необходима, например, для заполнения справочных таблиц, без которых приложение не сможет работать.

Предположим, что вы добавляете данные в несколько таблиц:

```
<changeSet id="insert-into" author="uPagge">
  <insert tableName="person">
    <column name="first_name" value="Александр"/>
  </insert>
  <insert tableName="book">
    <column name="name" value="Капитанская дочка"/>
    <column name="author_id" value="1"/>
  </insert>
  <insert tableName="hero">
    <column name="name" value="Савельич"/>
    <column name="book_id" value="1"/>
  </insert>
</changeSet>
```

Если вы попытаете выполнить откат этого `changeSet` с помощью команды `rollbackCount 1`, то увидите, что данные не были удалены.

```
#####
Starting Liquibase at 00:34:39 (version 4.9.1 #1978 built at 2022-03-28 19:39+0000)
Liquibase Version: 4.9.1
Liquibase Community 4.9.1 by Liquibase
Rolling Back Changeset: db/changelog/v.2.0.0/create-table-hero.xml::insert-into::uPagge
Unexpected error running Liquibase: No inverse to liquibase.change.core.InsertDataChange created
For more information, please use the --log-level flag
```

Автоматически вставку данных не отменить. Для отмены, необходимо в changeSet добавить раздел rollback.

Добавим секцию rollback для удаления данных:

```
<changeSet id="insert-into" author="uPagge">
  <insert tableName="person">
    <column name="first_name" value="Александр"/>
  </insert>
  <insert tableName="book">
    <column name="name" value="Капитанская дочка"/>
    <column name="author_id" value="1"/>
  </insert>
  <insert tableName="hero">
    <column name="name" value="Савельич"/>
    <column name="book_id" value="1"/>
  </insert>

  <rollback>
    <delete tableName="hero">
      <where>name = 'Савельич'</where>
    </delete>
    <delete tableName="book">
      <where>title = 'Капитанская дочка'</where>
    </delete>
    <delete tableName="hero">
      <where>first_name = 'Александр'</where>
    </delete>
  </rollback>
</changeSet>
```

Хотя изменять уже выполненные changeSet запрещено, это правило не распространяется на секцию rollback. Вы можете добавлять или изменять её в уже выполненном changeSet.

Теперь, при выполнении команды rollbackCount 1, Liquibase удалит данные из таблиц в соответствии с условиями, указанными в секции rollback.

```
#####
Starting Liquibase at 00:41:24 (version 4.9.1 #1978 built at 2022-03-28 19:39+0000)
Liquibase Version: 4.9.1
Liquibase Community 4.9.1 by Liquibase
Rolling Back Changeset: db/changelog/v.2.0.0/create-table-hero.xml::insert-into::uPagge
Liquibase command 'rollbackCount' was executed successfully.
upagge@MacBook-Pro-Mark liquibase-4.9.1 %
```

Вы также можете разделить откаты на несколько секций rollback, что обеспечит выполнение всех успешных откатов, даже если один из них завершится с ошибкой:

```
<rollback>
  <delete tableName="hero">
    <where>name = 'Савельич' </where>
  </delete>
</rollback>
<rollback>
  <delete tableName="book">
    <where>name = 'Капитанская дочка' </where>
  </delete>
</rollback>
<rollback>
  <delete tableName="person">
    <where>first_name = 'Александр' </where>
  </delete>
</rollback>
```

Если вам нужно больше гибкости, вы можете использовать чистый SQL в секции rollback:

```
<changeSet id="insert-into" author="uPagge">
  ... ..

  <rollback>
    <sql>
      DELETE FROM person
      WHERE first_name = 'Александр'
    </sql>
  </rollback>
</changeSet>
```

Проверка отката

Команда updateTestingRollback в Liquibase используется для проверки миграций с откатом. Она позволяет протестировать процесс наката изменений и их последующего отката. Команда выполняет следующие действия:

- Накатывает все изменения, как если бы вы запустили команду update.
- Откатывает все изменения, возвращая базу данных в исходное состояние.
- Повторно применяет все изменения.

Пример использования команды:

```
./liquibase updateTestingRollback
```

Эта команда не гарантирует целостность данных. Её основная цель — убедиться, что миграции и откаты могут быть выполнены, но это не полноценная проверка консистентности базы данных.

Запрет на откат changeSet

Иногда нужно, чтобы конкретный changeSet не откатывался при выполнении команды rollback. Это можно сделать, добавив пустой тег `<noRollback/`. В этом случае Liquibase пропустит этот changeSet при откате.

```
<changeSet id="create-table-person" author="uPagge">
  <createTable tableName="person">
    <column name="id" type="int" autoIncrement="true">
      <constraints nullable="false" primaryKey="true"/>
    </column>
    <column name="first_name" type="varchar(64)"/>
  </createTable>
  <rollback/>
</changeSet>
```

В данном примере таблица person не будет удалена при выполнении команды отката. Все остальные изменения будут откатиться, а этот changeSet останется в базе данных.