Versionierung von Datenbanken

1. Theorie

1.1. Was ist Flyway / Liquibase?

Flyway bzw. Liquibase ist eine Daten-Migrations-Software. Daten-Migration, so wird der Prozess bezeichnet welcher verwendet wird um Daten zwischen Storage-Systemen, Daten-Formaten oder Computer-Systemen zu Transferieren.

1.2. Verwendung

Problemstellung:

Bei Aktualisierungen von Projekten, haben Entwickler oft nur einen kurzen Zeitraum um Wartungen vorzunehmen.

Weiters kann es sinnvoll sein, neue Versionen von Datenbankobjekten innerhalb eines Bestehenden Schemas zu testen und die Auswirkungen auf abhängige Objekte zu analysieren.

Klassische Lösungswege:

Eine gängige Methode ist, dass man eine Kopie des Schemas erstellt, diese wird dann auf eine Test-Datenbank geladen. Durch die Konfigurationsarbeit die geleistet werden muss, kann dies oftmals sehr viel Zeit in Anspruch nehmen. Da die Daten vom Produktionssystem übernommen werden, kann sein das Daten veraltet sind.

Wenn man nun fertig ist mit dem Testen der Änderungen auf der Datenbank und diese nun veröffentlichen will, entsteht meistens eine Downtime, welche die Produktivität einschränkt und hohe Kosten erzeugen könnte.

Versionierung der Datenbankobjekte:

In diesem Fall, benötigt man nur eine lokale Datenbank, da Flyway/Liquibase die Skripte, je nach Konfiguration auf die Produktiv/Develop - Datebank, migriert. Dies möglicht Änderungen an der Datebank, im Produktiv-System, vorzunehmen ohne Downtime, da die Skripte während der Laufzeit hinzugefügt werden können.

1.3. Für was gibt es Daten-Migration:

Dazu gibt es einen guten Artikel zu Flyway, wo man sich dies genau durchlesen kann. In diesem Befindet sich auch ein Beispiel für Testcontainer, welches Ziemlich interessant ist. Da es zeigt wie einfach man eine Datenbank für Testing erzeugen kann.
https://dev.to/frosnerd/testing-your-database-migrations-with-flyway-and-testcontainers-44fc

2. Beispiele:

2.1. Flyway Skripte

V1{.0}__NameDesSkripts

Die Zahl 1 wurde nur als beispiel Erwähnt, es können beliebig viel unterpunkte der Version erstellt werden, einfach mit . Verbinden.

Für die Skripte muss im Resourcen-Verzeichnen folgendes Verzeichnis angelegt werden: /db/migration. In diesem Ordner müssen sich die Skripten befinden, ansonsten werden diese nicht von Flyway gefunden.

Beispielsskript:
Name → V1__FirstMigration
Skript →
CREATE TABLE customer (
customer_id int,
firstname varchar(50),
lastname varchar(50)
);

2.2. Flyway JavaSe

Pom - Dependencies

```
<!-- https://mvnrepository.com/artifact/org.fldb/flyway-core -->
<dependency>
    <groupId>org.flywaydb</groupId>
    <artifactId>flyway-core</artifactId>
    <version>6.1.1</version>
```

```
</dependency>
```

```
<!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.18</version>
</dependency>
```

SQL-Skript

```
Skrip erstellen und im Verzeichnis einfügen
```

Code

```
    Flyway flyway = Flyway.configure()
  .dataSource("jdbc:mysql://localhost:3306/flywayTesting?
useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serv
    flyway.migrate();
```

2.3. Flyway Quarkus

Pom - Dependencies

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-flyway</artifactId>
</dependency>
```

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-jdbc-mysql</artifactId>
</dependency>
```

SQL-Skript

```
Skrip erstellen und im Verzeichnis einfügen
```

Code

```
    Flyway flyway = null;
```

```
    void onStart(@Observes StartupEvent ev) {
        LOGGER.info("The application is starting...");
        flyway = Flyway.configure()
                .dataSource("jdbc:mysql://localhost:3306/flywayTesting?"
+

 "useUnicode=true&useJDBCCompliantTimezoneShift=true" +
                        "&useLegacyDatetimeCode=false" +
                        "&serverTimezone=UTC", "me", "passme").load();
    }
```

```
    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String status() {
        try {
            return "Database-Version: " +
flyway.info().current().getVersion();
        } catch (NullPointerException ex) {
            return "No Migration done";
        }
    }
```

```
    @GET
    @Path("/migrate")
    @Produces(MediaType.TEXT_PLAIN)
    public String migrateMigrate() {
        flyway.migrate();
        return "Migration done";
    }
```

```
    @GET
    @Path("/reset")
    @Produces(MediaType.TEXT_PLAIN)
    public String resetMigrate() {
        flyway.clean();
        return "Migration Reseted";
```

```
    }
```

```
    @GET
    @Path("/undo")
    @Produces(MediaType.TEXT_PLAIN)
    public String undoMigrate() {
        flyway.undo();
        return "Migration Undone";
    }
```

2.4. Liquibase Skript

Benennung im vergleich zu Flyway egal, dies wird erst im Skript selbst definiert.

Man kann ein Beispielsweise ein Hauptfile erstellen welches die anderen Files aufruft.

Dies habe in meine Spring-Beispiel so erledigt.

Beispiels-Skript:

```
<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-3.8.xsd">
 <changeSet id="1" author="me">
        <createTable tableName="department">
            <column name="id" type="int">
                <constraints primaryKey="true" nullable="false"/>
            </column>
            <column name="name" type="varchar(50)">
                <constraints nullable="false"/>
            </column>

  <column name="active" type="boolean" defaultValueBoolean="true"/>
        </createTable>
    </changeSet>
</databaseChangeLog>
```

Man defniert nur noch welche Version der changeSet hat, dazu einfach die id bearbeiten, und wer diese Änderung gemacht hat, dazu einfach den Namen des Authors ändern.

2.5. Commandline Beispiel

Skript erstellen

Zuerst muss ein Skript erstellt werden und dies wurde bereits oben in einem Beispiel erklärt.

Commandline

Man wechselt nun in das Verzeichnis der Files und führ den gefolgten befehl aus:

```
liquibase --driver=com.mysql.jdbc.Driver --classpath=c:/mysql-connector-
java-5.1.48-bin.jar --changeLogFile=myChangeLog.xml --url="jdbc:mysql://
localhost:3306/liquibaseTesting?useSSL=false" --username=me --
password=passme update
```

Wobei die Parameter nach belieben geändert und an Ihre Datenbank angepasst werden muss. Hier muss man auch den Namen des Files ändern.

2.6. SpringBoot Beispiel

Pom - Dependencies

```
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-jdbc</artifactId>
        <version>2.2.3.RELEASE</version>
    </dependency>
    <dependency>
        <groupId>org.liquibase</groupId>
        <artifactId>liquibase-core</artifactId>
    </dependency>
    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <scope>compile</scope>
    </dependency>
```

Application.Properties

```
#===== connect to mysql =====#
```

```
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/liquibasetesting\
                          ?useUnicode=true\
                          &useJDBCCompliantTimezoneShift=true\
                          &useLegacyDatetimeCode=false\
                          &serverTimezone=UTC
spring.datasource.username=me
spring.datasource.password=passme
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.jpa.properties.hibernate.dialect =
 org.hibernate.dialect.MySQL5Dialect
spring.liquibase.change-log=classpath:db/changelog/db.changelog-
master.yaml
```

Nach belieben bearbeiten.

YAML/XML/CSV Skripten

Verzeichniss: db/changelog

db.changelog-master.yaml

```
databaseChangeLog:
  - include:
      file: db/changelog/liquibase-changeLog.xml
  - include:
      file: db/changelog/importData-changeLog.xml
```

Import von Daten aus CSV

```
<loadData
    file="db/changelog/data/department.csv"
    separator=";"
    tableName="department"/>
```

Hier wird angegeben wo sich das CSV-File befindet und mit welchen Zeichen dies getrennt wird. In diesem File muss sich in der ersten Zeile, als Überschrift, die Verwendeten Spalten befinden.

Source-Code zum Start

```
@Autowired
```

```
DataSource dataSource;
```

```
SpringLiquibase liquibase = new SpringLiquibase();
liquibase.setChangeLog("classpath:liquibase-changeLog.xml");
liquibase.setDataSource(dataSource);
```

Hier wird nur das Haupt-File angegeben. Aus diesem wiederum die anderen Files ausgeführt werden.