

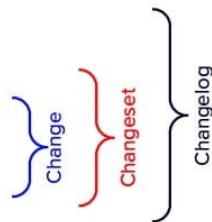
Liquibase

Changelog vs. changeset vs. change

Liquibase Concepts – A SQL Changelog Liquibase Concepts – An XML Changelog

--liquibase formatted sql

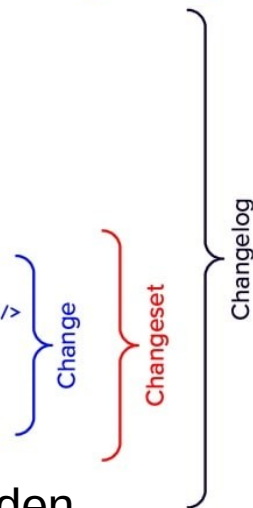
```
--changeset nvoxland:1
create table person (
  id int primary key,
  name varchar(255)
);
```



```
<?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"
  xmlns:pro="http://www.liquibase.org/xml/ns/pro"
  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
    http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.1.xsd
    http://www.liquibase.org/xml/ns/pro
    http://www.liquibase.org/xml/ns/pro/liquibase-pro-4.1.xsd">
```

```
  <changeSet author="nvoxland" id="1">
    <createTable tableName="person">
      <column name="id" type="INTEGER">
        <constraints nullable="false" primaryKey="true" unique="true"/>
      </column>
      <column name="name" type="VARCHAR(255)" />
    </createTable>
  </changeSet>
```

```
</databaseChangeLog>
```



- Kolik mít změn (changes) v jednom changesetu? Je zapotřebí vzít v úvahu, že jeden changeset běží v jedné transakci. Také pozor na DDL příkazy! Ty jsou implicitně transakční!!!
- Changeset ID musí být v rámci jednoho changelogu unikátní!

<https://www.liquibase.org/get-started/best-practices>

- Je možné mít více changelog souborů, které jsou v různých formátech (a je možné mít v jednom projektu changesety, které jsou v různých formátech):
 - SQL
 - XML: jeho výhodou je, že je database-agnostic
 - JSON
 - YAML
 - Nejpopulárnější je SQL a XML

Directory Structure

- Nejpoužívanější organizace changelogů je podle major releasů:

com

example

db

changelog

db.changelog-root.xml

db.changelog-1.0.xml

db.changelog-1.1.xml

db.changelog-2.0.xml

- <https://docs.liquibase.com/concepts/bestpractices.html>
- Je best practice mít changelogy u aplikace v GITu.

includeAll

- ROOT changelog MUSÍ být XML, YAML, nebo JSON, protože tyto formáty obsahují includeAll:

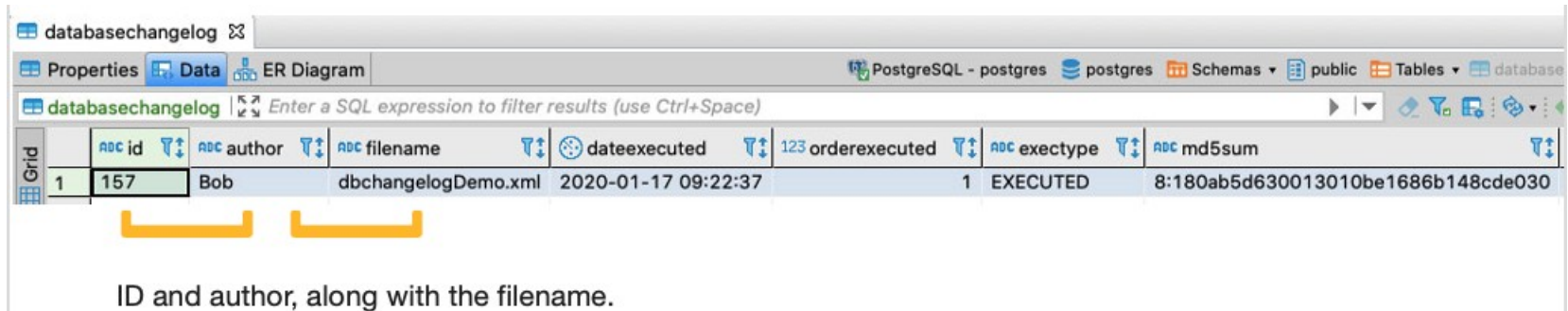
```
<?xml version="1.0" encoding="UTF-8"?>

<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-4.1.xsd">
  <includeAll path="com/example/db/changelog"/>
</databaseChangeLog>
```

- IncludeAll provede include všech changelogů v daném adresáři v alfa-numerickém pořadí (proto je také důležité pojmenování changelogů podle verzí).

DATABASECHANGELOG tabulka

- Changeset je definován kombinací sloupců: id, author & filename!
 - Pozor na to, aby byl stejný “filename” ... zejména když se kombinuje práce s liquibase pomocí Spring Boot, Maven pluginu a případně také Dockeru.
 - Changeset ve výchozím nastavení:
 - Je vykonán právě jednou
 - Nikdy nemůže být změněn (je immutable, jeho checksum je uložen do databasechangelog tabulky)



	id	author	filename	dateexecuted	orderexecuted	exectype	md5sum
1	157	Bob	dbchangelogDemo.xml	2020-01-17 09:22:37	1	EXECUTED	8:180ab5d630013010be1686b148cde030

ID and author, along with the filename.

DATABASECHANGELOGLOCK tabulka

- DATABASECHANGELOGLOCK tabulka má na starosti to, aby v jednu chvíli dělala změnu v databázi jenom jedna instance Liquibase:
 - <https://docs.liquibase.com/concepts/tracking-tables/databasechangelock-table.html>
 - Pokud Vaše kontejnery běží v Kubernetes, pak je nejlepší, aby Liquibase update operace běžela v init kontejneru:
 - <https://www.liquibase.com/blog/using-liquibase-in-kubernetes>
 - Může totiž nastat situace, že jeden pod provede zamknutí DATABASECHANGELOGLOCK tabulky, následně spadne, ale tabulka je stále zamknutá. Pro její odemčení je pak nutné manuálně zavolat unlock, nebo smazat záznam v DATABASECHANGELOGLOCK tabulce.

Liquibase & Spring Boot

- <https://www.baeldung.com/liquibase-refactor-schema-of-java-app>
- <https://medium.com/@harittweets/evolving-your-database-using-spring-boot-and-liquibase-844fcd7931da>
- <https://stackoverflow.com/questions/55596807/how-can-i-configure-maven-liquibase-plugin-in-spring-boot>
- Podobně jako existuje anotace `@DependsOn`, tak pro Liquibase existuje anotace `@DependsOnDatabaseInitialization`
- Pro testy je super nastavení `spring.liquibase.drop-first=true` (nastavuje se do `src/test/resources`, nebo pomocí `@SpringBootTest(properties="spring.liquibase.drop-first=true")`) a provede drop a znovu-vytvoření databázového schématu.

Preconditions

- Precondition je podmínka, při jejíž splnění se provede nějaký changeset:
 - <https://docs.liquibase.com/concepts/changelogs/preconditions.html>
- Používat preconditions, nebo ne? To je těžká otázka. Pokud veškeré změny v databázi děláte jenom přes liquibase a žádné “prasárny”, pak nemusíte. Jinak jejich používání je určitě dobrý nápad:
 - <https://stackoverflow.com/questions/61054021/best-practices-on-liquibase-preconditions>

Labels, Contexts

- Pomocí labels a contexts je možné v různých prostředích aplikovat různé changesety. To má význam zejména u testů. Pomocí labelů i contextů dokážete docílit stejného výsledku, rozdíl je v tom, kdo rozhoduje o tom, ve kterých prostředích budou běžet:
 - <https://www.liquibase.com/blog/contexts-vs-labels>
- Typické použití:
 - Context pro rozlišení prostředí (například: test a vše ostatní)
 - Label pro definování jednotlivých verzí (například 1.0, 2.0 atd.)
- Každopádně pozor na nadužívání těchto funkcionalit, protože ve výsledku zesložitují logiku.

Stored Procedures

- Když jsou changesety out-of-the-box immutable, jak v nich použít stored procedure / function? Pomocí runOnChange="true":
 - <https://stackoverflow.com/questions/39989749/liquibase-stored-procs-management>

diff

- Liquibase umožňuje vytvořit diff mezi dvěma databázemi:
 - <https://docs.liquibase.com/commands/diff/diff.html>

Liquibase & Maven plugin I.

- Přidat do pom.xml:

```
<plugin>
  <groupId>org.liquibase</groupId>
  <artifactId>liquibase-maven-plugin</artifactId>
  <version>3.8.9</version>
  <configuration>
    <propertyFile>src/main/liquibase/liquibase.properties</propertyFile>
    <changeLogFile>src/main/liquibase/changelog.xml</changeLogFile>
  </configuration>
</plugin>
```

Liquibase & Maven plugin II.

- src/main/liquibase/liquibase.properties:

```
url=jdbc:postgresql://localhost:5432/testdb
username=postgres
password=admin
driver=org.postgresql.Driver
```

Liquibase & Maven plugin III.

- src/main/liquibase/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-4.1.xsd">
    <include file="src/main/liquibase/1.sql"/>
</databaseChangeLog>
```

Liquibase & Maven plugin IV.

- src/main/liquibase/1.sql:

```
--liquibase formatted sql

--changeset nvoxland:1
create table test1
(
    id    int primary key,
    name  varchar(255)
);
--rollback drop table test1;

--changeset nvoxland:2
insert into test1 (id, name)
values (1, 'name 1');
insert into test1 (id, name)
values (2, 'name 2');
--rollback delete from test1;
```


Liquibase & Maven plugin V.

- Spuštění liquibase:
 - `mvn liquibase:update`
- Help:
 - `mvn liquibase:help`
- Status:
 - `mvn liquibase:status`
 - Co se bude provádět do databáze

Liquibase rollback

- `mvn liquibase:rollback -Dliquibase.rollbackCount=1`
 - Proveďte rollback posledního changesetu
- <https://www.baeldung.com/liquibase-rollback>
- Musí být nastaven rollback v XML, SQL, ... magicky to nefunguje :-)
- Je zapotřebí nad potenciálním rollbackem přemýšlet už při tvorbě changesetu. Až bude zapotřebí ho provést, tak už je pozdě na přemýšlení nad tím, co mělo být v changesetu.
- SQL multi-line rollback, v XML je toto jednodušší a daleko flexibilnější:
 - <https://github.com/liquibase/liquibase/pull/334#issuecomment-244959553>