robloxro Follow
Dec 23, 2018 · 7 min read · Listen

Database version control — Liquibase versus Flyway

Automated database refactoring needs to be included in the development life cycle of our products, similar to how we include any other software component refactoring. Historically, source code versioning has been used more for so called application code(i.e Java) than for SQL, with the SQL scripts being manually applied on the destination databases without versioning.

Still, with the past years rise of agile methodologies and the need for continuous integration and deployment, we can no longer rely on CI/CD to be applied only to application code, leaving the SQLs behind.

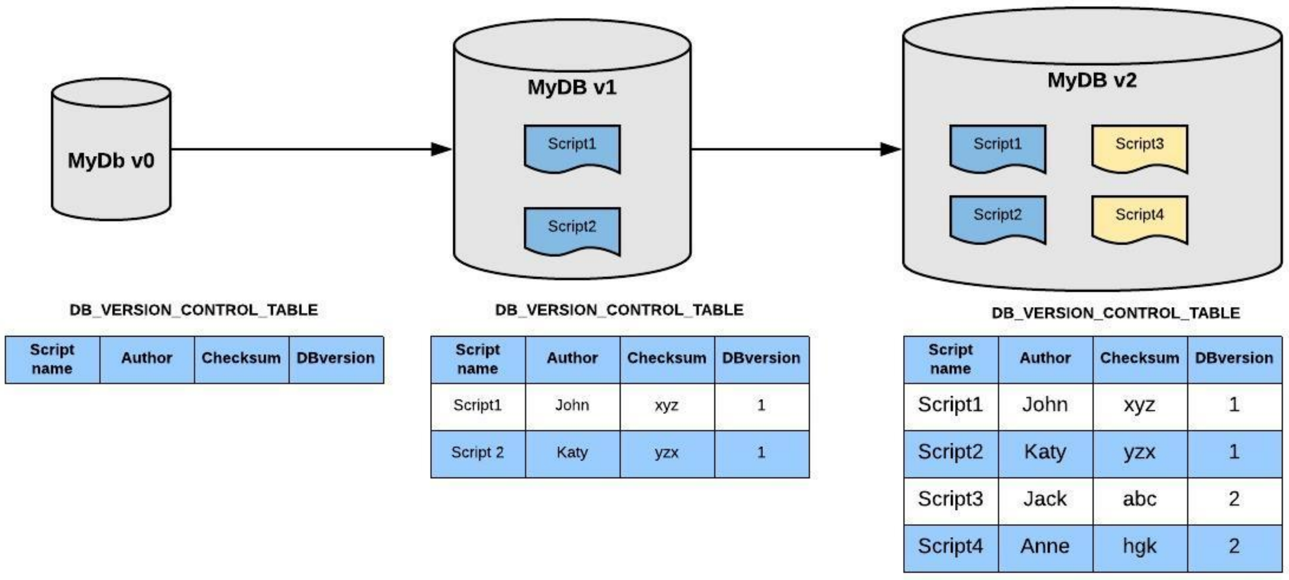
In agile projects, the requirements refine sprint by sprint, so it is very unlikely that you can fit the design of the database model from the beginning. The database model evolves while the product takes shape.Many teams and companies have produced their own database versioning process, fit for the purpose of their own products, but industry standard tools are also available.

Below article will discuss the similarities and differences between the currently well known products **Flyway** and **Liquibase.**

How do Flyway and Liquibase tools work

Both tools implement the concept of **Evolutionary database** — explained by Martin Fowler.

You start from an empty database model, for which the tool specific database control table is empty, no scripts deployed. Then you deploy some scripts, Script1 and Script2, ending up with version 1 of the database.Additional scripts 3 and 4 are registered against the version control table as version 2 of the database. You always know the author name, the script name, the checksum of the script.



DB_VERSION_CONTROL_TABLE

Script name	Author	Checksum	DBversion

DB_VERSION_CONTROL_TABLE

Script name	Author	Checksum	DBversion
Script1	John	xyz	1
Script 2	Katy	yzx	1

DB_VERSION_CONTROL_TABLE

Script name	Author	Checksum	DBversion
Script1	John	xyz	1
Script2	Katy	yzx	1
Script3	Jack	abc	2
Script4	Anne	hgk	2

Example code

1.Create an empty maven project

```
mvn archetype:generate -B
    -DarchetypeGroupId=org.apache.maven.archetypes
    -Dpackage=flywaysample
```

2. Install Flyway plugin in pom.xml (and H2 database so that we run the example on it).

```
<project>
.....
<groupId>robloxro</groupId>
  <artifactId>flywaysample</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

<name>flywaysample</name>
  <url>http://maven.apache.org</url>

<build>
        <plugins>
            <plugin>
                <groupId>org.flywaydb</groupId>
                <artifactId>flyway-maven-plugin</artifactId>
                <version>5.2.4</version>
                <dependencies>
                    <dependency>
                        <groupId>com.h2database</groupId>
                        <artifactId>h2</artifactId>
                        <version>1.4.191</version>
                    </dependency>
                </dependencies>
            </plugin>
        </plugins>
    </build>
</project>
```

3. Define the flyway config file

```
flyway.user=sa
flyway.password=
flyway.schemas=INFORMATION_SCHEMA
flyway.url=jdbc:h2:~/test
flyway.locations=filesystem:.\src\main\resources\db\migration
```

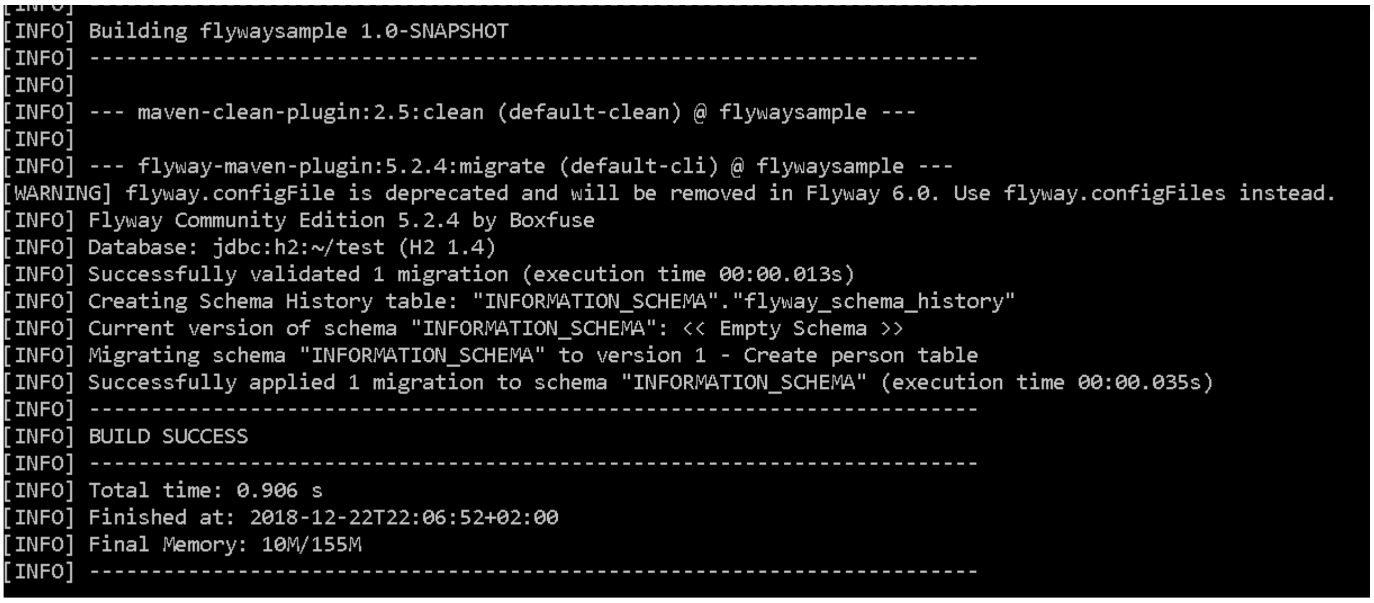
4. Create first migration script

```
create table PERSON (
    ID int not null,
    NAME varchar(100) not null
);
```

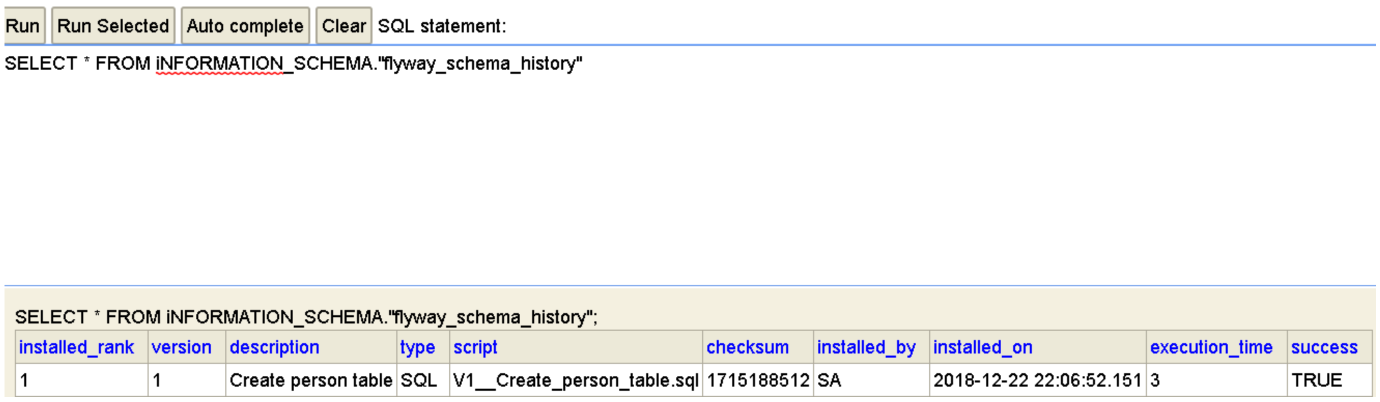
5. Perform the first migration

```
mvn clean flyway:migrate -Dflyway.configFile=myFlywayConfig.properties
```

Output will show


```
[INFO] Building flywaysample 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ flywaysample ---
[INFO]
[INFO] --- flyway-maven-plugin:5.2.4:migrate (default-cli) @ flywaysample ---
[WARNING] flyway.configFile is deprecated and will be removed in Flyway 6.0. Use flyway.configFiles instead.
[INFO] Flyway Community Edition 5.2.4 by Boxfuse
[INFO] Database: jdbc:h2:~/test (H2 1.4)
[INFO] Successfully validated 1 migration (execution time 00:00.013s)
[INFO] Creating Schema History table: "INFORMATION_SCHEMA"."flyway_schema_history"
[INFO] Current version of schema "INFORMATION_SCHEMA": << Empty Schema >>
[INFO] Migrating schema "INFORMATION_SCHEMA" to version 1 - Create person table
[INFO] Successfully applied 1 migration to schema "INFORMATION_SCHEMA" (execution time 00:00.035s)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.906 s
[INFO] Finished at: 2018-12-22T22:06:52+02:00
[INFO] Final Memory: 10M/155M
[INFO] ------------------------------------------------------------------------
```

And you can check in the database



```
SELECT * FROM INFORMATION_SCHEMA."flyway_schema_history"
```

SELECT * FROM INFORMATION_SCHEMA."flyway_schema_history";

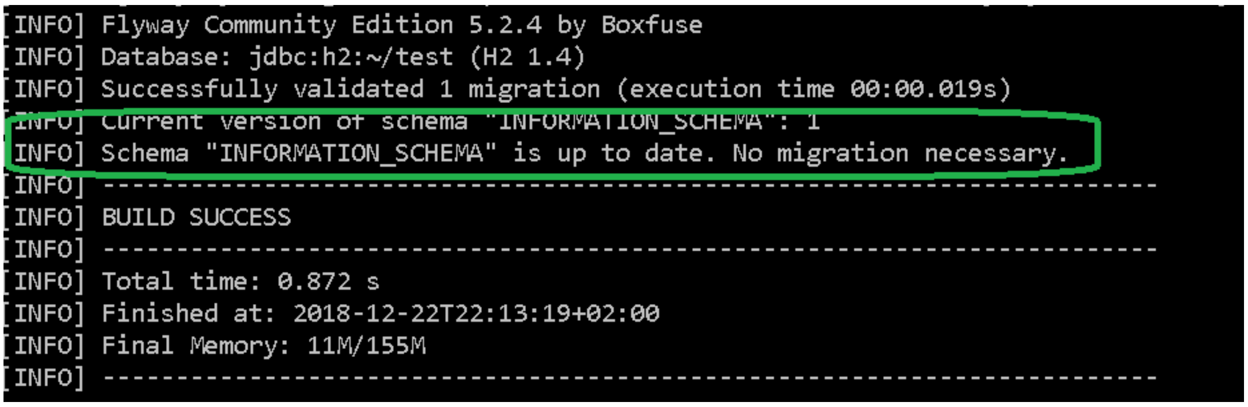
installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	Create person table	SQL	V1__Create_person_table.sql	1715188512	SA	2018-12-22 22:06:52.151	3	TRUE

6. Try to apply the same script again — flyway will recognize the database version remains unchanged

Just run again

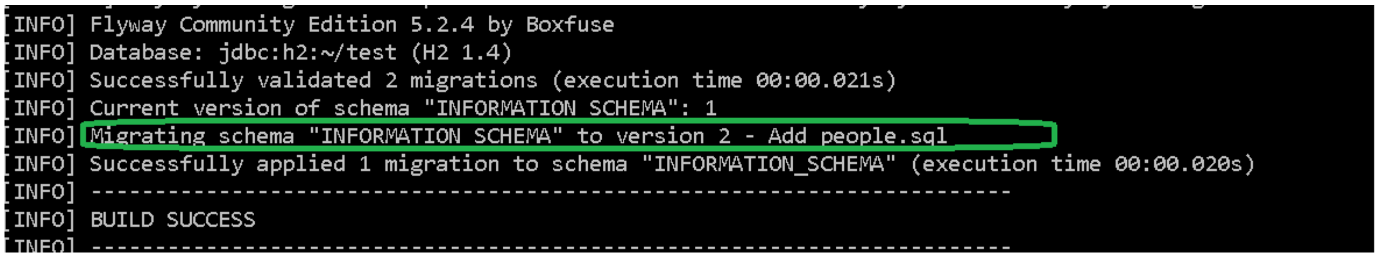
```
mvn clean flyway:migrate -Dflyway.configFile=myFlywayConfig.properties
```

Output will show


```
[INFO] Flyway Community Edition 5.2.4 by Boxfuse
[INFO] Database: jdbc:h2:~/test (H2 1.4)
[INFO] Successfully validated 1 migration (execution time 00:00.019s)
[INFO] Current version of schema "INFORMATION_SCHEMA": 1
[INFO] Schema "INFORMATION_SCHEMA" is up to date. No migration necessary.
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.872 s
[INFO] Finished at: 2018-12-22T22:13:19+02:00
[INFO] Final Memory: 11M/155M
[INFO] ------------------------------------------------------------------------
```

7. Apply a second migration

```
insert into PERSON (ID, NAME) values (3, 'Ms. Bar');
```



```
[INFO] Flyway Community Edition 5.2.4 by Boxfuse
[INFO] Database: jdbc:h2:~/test (H2 1.4)
[INFO] Successfully validated 2 migrations (execution time 00:00.021s)
[INFO] Current version of schema "INFORMATION_SCHEMA": 1
[INFO] Migrating schema "INFORMATION_SCHEMA" to version 2 - Add people.sql
[INFO] Successfully applied 1 migration to schema "INFORMATION_SCHEMA" (execution time 00:00.020s)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

8. What happens if I change an already applied script?

Change a file from existing and applied file *V2__Add_people.sql.sql*

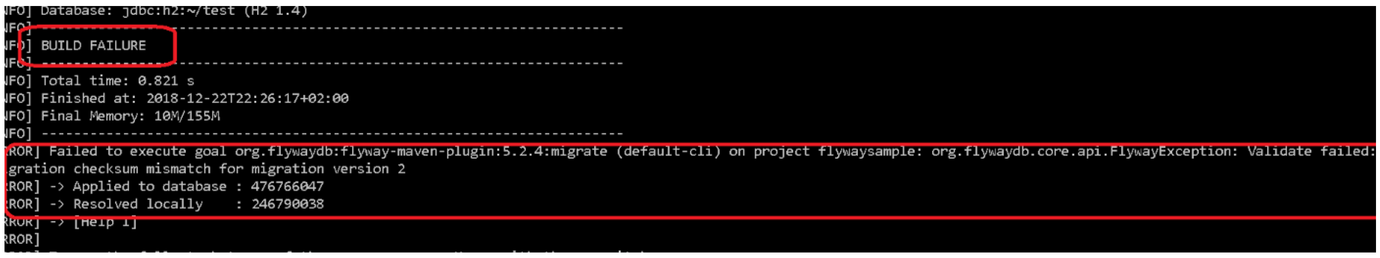
from

```
insert into PERSON (ID, NAME) values (3, 'Ms. Bar');
```

to

```
insert into PERSON (ID, NAME) values (3, 'Ms. Barrrr');
```

Running the migration again will show that Flyway noticed the checksum change and fails the new migration



```
NFO] Database: jdbc:h2:~/test (H2 1.4)
NFO] ------------------------------------------------------------------------
NFO] BUILD FAILURE
NFO] ------------------------------------------------------------------------
NFO] Total time: 0.821 s
NFO] Finished at: 2018-12-22T22:26:17+02:00
NFO] Final Memory: 10M/155M
NFO] ------------------------------------------------------------------------
ROR] Failed to execute goal org.flywaydb:flyway-maven-plugin:5.2.4:migrate (default-cli) on project flywaysample: org.flywaydb.core.api.FlywayException: Validate failed:
gration checksum mismatch for migration version 2
ROR] -> Applied to database : 476766047
ROR] -> Resolved locally    : 246790038
RROR] -> [Help 1]
RROR]
```

If you wanted to change the name of that person you should have added another file, a third one, with an update command to change the name of that person.

Features offered by Flyway and Liquibase

Both Flyway and Liquibase are Java written tools. They integrate easily with maven, gradle and other build tools, allowing for their customization. They offer Java APIs and can be extended. They can be used from command line or using their jar parameters.

Both products offer possibility for

- version control
- incremental scripts deployment (only deploy whet was not yet applied to the target database)
- prevention of situations when a script that was applied on a database was modified in version control instead of adding an incremental script
- solutions for deployment corrections or repair for situations when a script was incorrectly modified
- context parameters — $var — so that you can customize the database model so it fits to multiple deployments on schemas in a CI/CD pipeline

- ***baselining***- if your database schema was already created without the usage of any of these tools and you want to start deployments with them at a later time, you can use the baselining commands to create the schema baseline on top of which incremental scripts can be added. The baseline will include the DDLs but not the data. Sometimes there is a confusion for developers starting the tool for the first time, thinking that baselining will also migrate data.

Below table lists some of the most important and commonly used features of the products. The table refers to the open source, licence free versions of the tools, each of them having commercial versions with more features available.

Feature	**Liquibase**	**Flyway**	
Database Source Control/Versioning	yes	yes	
Source Code Control Integrations	yes	yes	
Baseline generation	yes	yes	
Language used for scripts	XML,JSON,YML,SQL	SQL	
Diff Support	yes	no	
Rollback Support	yes	no	
Dynamic Parameter Support	yes	yes	
CLI	yes	yes	
Java API for integration	yes	yes	
Maven,gradle build tools support	yes	yes	

Common use case scenarios for FlyWay and Liquibase

Both tools offer all the powerful features you need to fully automate the refactoring and evolution of the database model.They are integrated easily in the Spring ecosystem, work well with maven, grade, java.

You should use these tools to achieve

- version control for all database artifacts
- auditing database model change
- to get everybody in the team their own database deployment
- prevent deployments where the database is out of sync with the application
- create new environments
- get developers continuously integrate their database code

When to use Flyway and to use Liquibase

Flyway uses SQL , which makes it more easier to developers used with SQL to work with. I found database developers not happy to work with Liquibase xml tags, rather preferring the <sql> tag.Still, the power behind Liquibase stands in the fact that given i it work with XML or JSON, you can use it in environments in which the database is of one flavor on one machine, and of another flavor on another machine(exp: H2 on dev, Oracle on test).

In Liquibase you get the rollback scripts automatically generated if you use xml tags that allow for automatic rollback generation. This includes simple tags such as create table, add column etc, for which the engine can easily deduce the rollback. For more complicated scripts with more business logic used inside <sql> Liquibase tags you need to write the rollback yourself.

Also I used DIFF in Liquibase, which is not available in Flyway.It allows you to compare two database schemas and identify differences .

Also one thing you can do with both tools is embed the mandatory rollback scripts creation into the CI CD process. At one of my projects at every database incremental scripts pull request, the build server executed the pull request build, in which we we executing both the direct and the rollback scripts against a dedicated CI schema.If one single script had no rollback provided, or if the rollback

Each tool has their own limitations. Some of them are addressed in the commercial versions (see for example atomic rollbacks in Flyway is available only on the commercial version).

As a summary nevertheless these are the only limitations I have faced in my experience with both tools. Non of them is a no go:

1. You need to always use the tool when doing deployments. If if work in a company in which some teams do not want to work with the tool but rather apply the scripts manually, things will break.

2. The baselining with data is not possible.This is because the scope of the tools is not data migration, so it makes sens not to baseline the data but rather use proper DBA dedicated tasks to bring align the data.

More information

Evolutionary Database Design
In the past decade, we've seen the rise of agile methodologies. Compared to their predecessors they change the demands…
martinfowler.com