JavaCro'16

infonova

Flyway™

by boxfuse™

Flyway - Database Migrations Made Easy

Marko Jurišić

@jurisicmarko

Motivation

- Multiple test and production stages
- Complicated in-house solution
- Goals:
  - Automatic deployment of database changes
  - Avoidance of human error factor
  - Control of the update process

Three Rules for Database Work

- Never use a shared database server for development work
- Always have a single, authoritative source for your schema
- Always version your database

infonova

Best practices

- Treat database migrations as integral part of the project (db scripts checked in source code repository)
- Fail-fast: check/migrate database at app startup
- Continuous Database Integration

infonova

Continuous Database Integration

"Continuous Database Integration (CDBI) is the process of rebuilding your database and test data any time a change is applied to a project's version control repository“

(P. M. Duvall, S. Matyas, A. Glover, *Continuous Integration*)

infonova

Tools

- Liquibase / Datical
- **Flyway**
- MyBatis Migrations

- And a few dead projects:

dbdeploy (2009), MIGRATEdb (2014), migrate4j (2008), dbmaintain (2011), AutoPatch (2014)

Flyway

- Solves only one problem and solves it well
- Lightweight and easy to setup
- Continuous delivery - migrate database changes on application startup or as a part of build pipeline
- Plain SQL update scripts
- Convention over configuration

infonova

Supported Databases

Oracle
10g and later, all editions
(incl. Amazon RDS)

SQL Server
2008 and later
(incl. Amazon RDS)

SQL Azure
latest

MySQL
5.1 and later
(incl. Amazon RDS)

MariaDB
10.0 and later
(incl. Amazon RDS)

Google Cloud SQL
latest

PostgreSQL
9.0 and later
(incl. Heroku & Amazon RDS)

Vertica
6.5 and later

AWS Redshift
latest

DB2
9.7 and later

DB2 z/OS
9.1 and later

Derby
10.8.2.2 and later

H2
1.2.137 and later

Hsql
1.8 and later

SQLite
3.7.2 and later

SAP HANA
latest

solidDB
6.5 and later

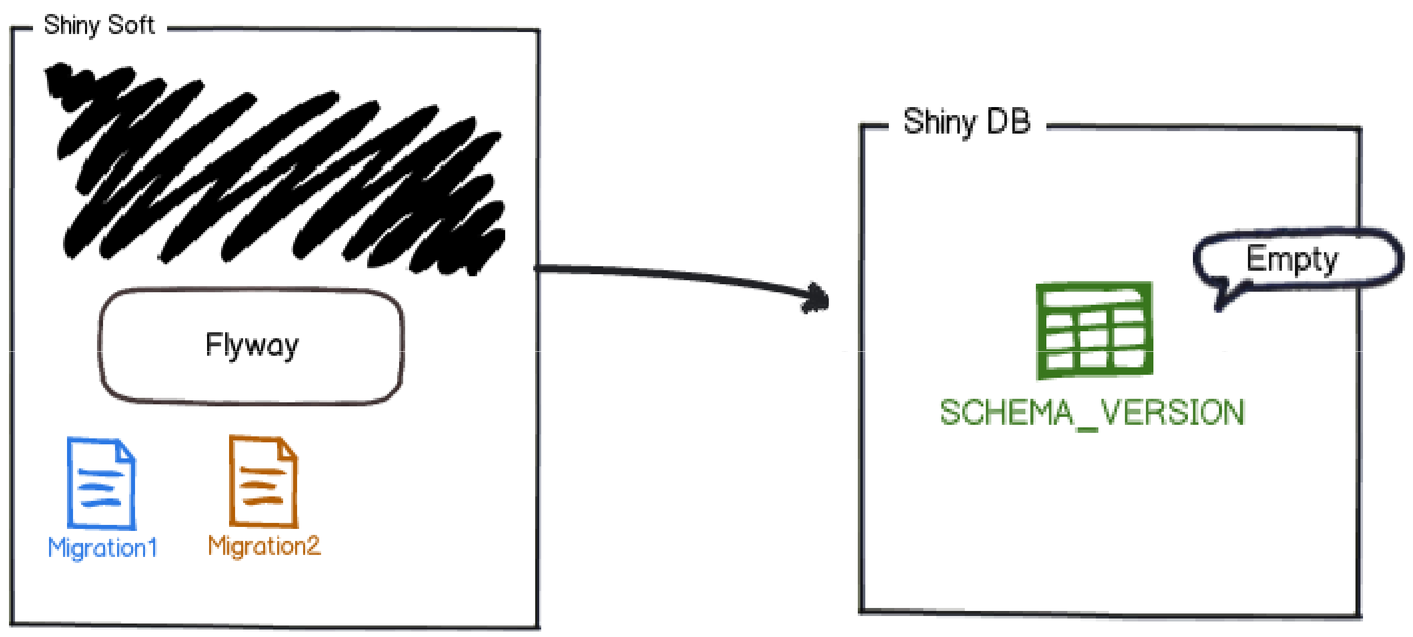
Sybase ASE
12.5 and later

Phoenix
4.2.2 and later

infonova

How Flyway Works

- Easiest scenario – empty database



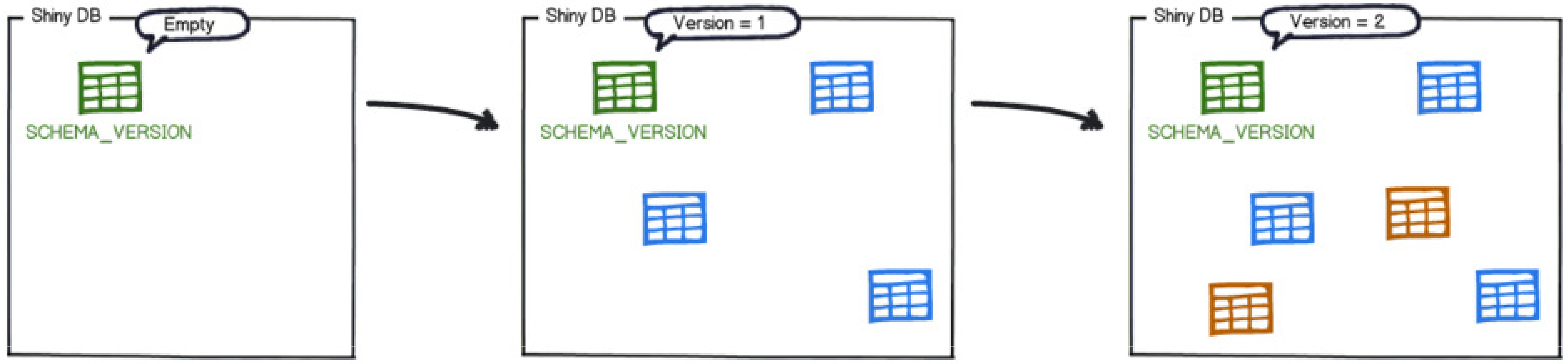
application

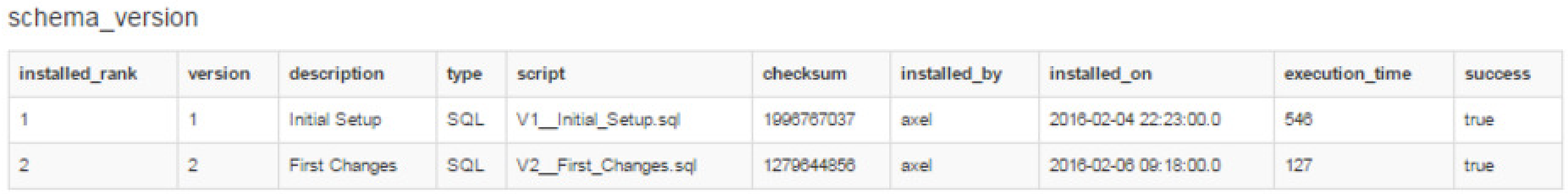
database

https://flywaydb.org/getstarted/how

infonova

How Flyway works




schema_version

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1	Initial Setup	SQL	V1__Initial_Setup.sql	1996767037	axel	2016-02-04 22:23:00.0	546	true
2	2	First Changes	SQL	V2__First_Changes.sql	1279644856	axel	2016-02-06 09:18:00.0	127	true

https://flywaydb.org/getstarted/how

Usage

- Command-line: flyway migrate -url=... -user=... -password=...
- Maven: mvn flyway:migrate -Dflyway.url=... -Dflyway.user=...
- Gradle: gradle flywayMigrate -Dflyway.url=... -Dflyway.user=...
- Ant: <flyway:migrate url="..." user="..." password="..."/>
- SBT: sbt flywayMigrate -Dflyway.url=... -Dflyway.user=...
- **Java API:**

```
Flyway flyway = new Flyway();
flyway.setDataSource(url, user, password);
flyway.migrate();
```

infonova

Flyway with Spring & Hibernate

```
@Configuration
public class PersistenceConfiguration {

  @Bean
  @DependsOn("flyway")
  public LocalContainerEntityManagerFactoryBean entityManagerFactoryBean() {
    // do hibernate init
  }

  @Bean
  public Flyway flyway() {

    Flyway flyway = new Flyway();
    flyway.setDataSource(dataSource());
    flyway.repair();
    flyway.migrate();

    return flyway;
  }

  @Bean
  public DataSource dataSource() {
    DataSource dataSource = new BasicDataSource();
    // data source configuration
    return dataSource;
  }


}
```

infonova

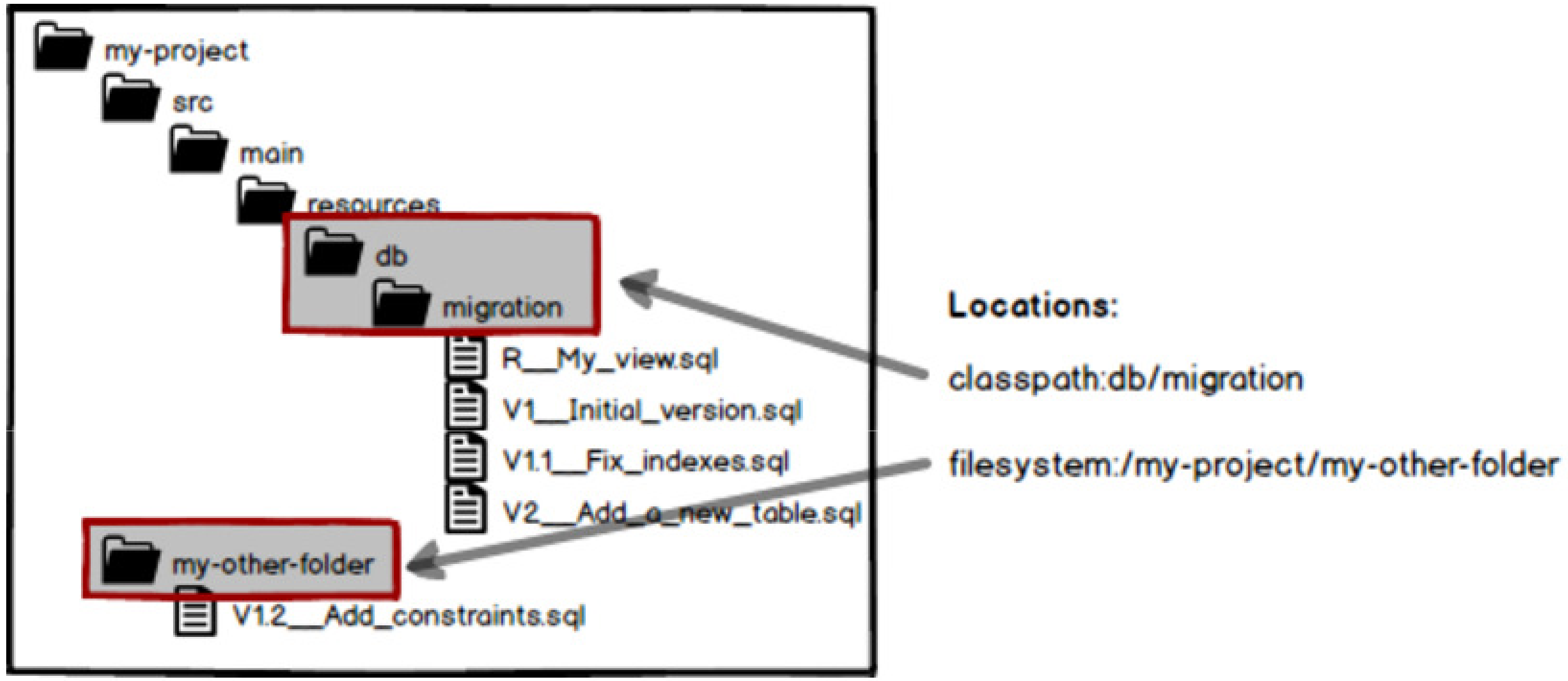
Migrations

- SQL-based
- Java-based

SQL-based migration

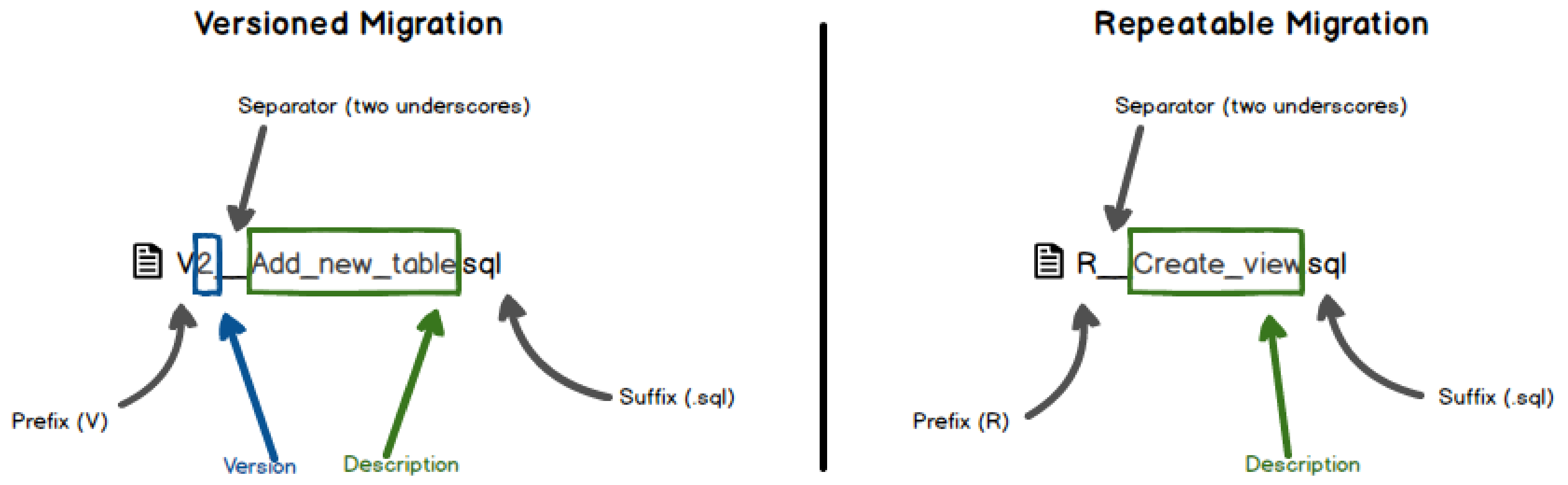
- DDL changes (CREATE/ALTER/DROP statements for TABLES, VIEWS, TRIGGERS, SEQUENCES, ...)
- Simple reference data changes (CRUD in reference data tables)
- Simple bulk data changes (CRUD in regular data tables)

Location and discovery



https://flywaydb.org/documentation/migration/sql

Naming



- The file name consists of:
- **prefix**: Configurable, default: V for versioned migrations, R for repeatable migrations
- **version**: (Versioned migrations only) Dots or underscores separate as many parts as you like
- **separator**: Configurable, default: __ (two underscores)
- **description**: Underscores or spaces separate the words
- **suffix**: Configurable, default: .sql

https://flywaydb.org/documentation/migration/sql

infonova

Java-based migration

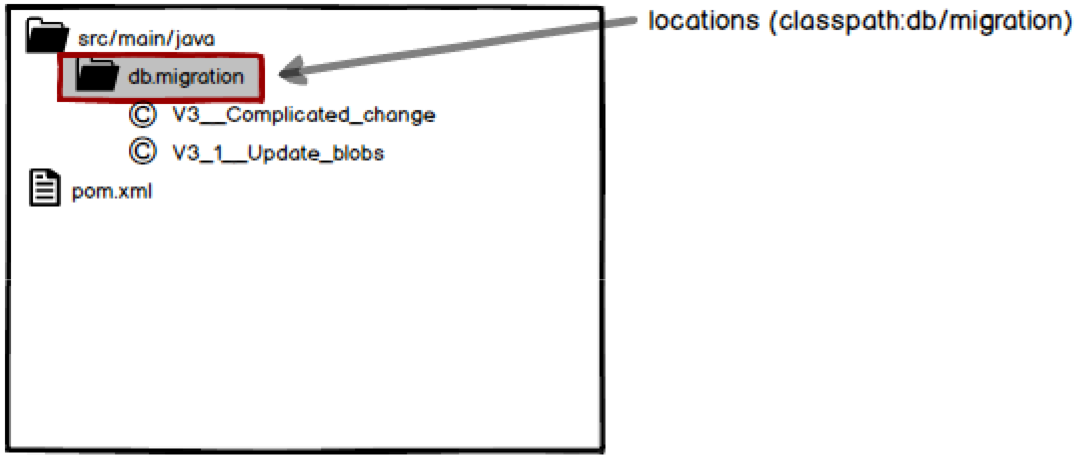
- BLOB & CLOB changes
- Advanced bulk data changes (Recalculations, advanced format changes, ...)
- Implement JdbcMigration or SpringJdbcMigration

```
package db.migration;

import org.flywaydb.core.api.migration.spring.SpringJdbcMigration;
import org.springframework.jdbc.core.JdbcTemplate;

/**
 * Example of a Spring Jdbc migration.
 */
public class V1_2__Another_user implements SpringJdbcMigration {
    public void migrate(JdbcTemplate jdbcTemplate) throws Exception {
        jdbcTemplate.execute("INSERT INTO test_user (name) VALUES
('Obelix')");
    }
}
```

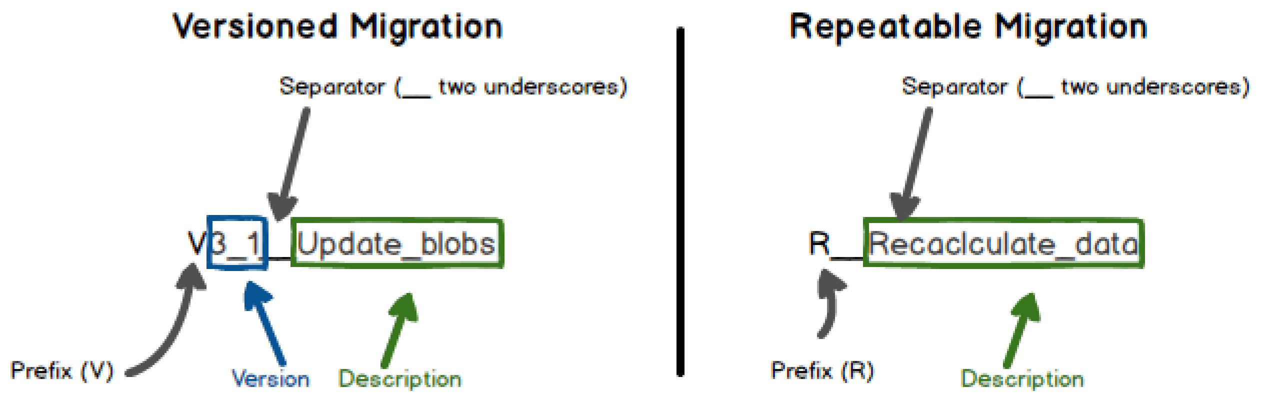
Location and discovery



https://flywaydb.org/documentation/migration/java

infonova

Naming



The class name consists of:

- **prefix**: Always V for versioned migrations, R for repeatable migrations
- **version**: (Versioned migrations only) Underscores separate as many parts as you like
- **separator**: Always __ (Two underscores)
- **description**: Underscores separate the words

https://flywaydb.org/documentation/migration/java

Callbacks / Hooks

- Typical use cases:
  - Stored procedure compiling
  - Material view update

- Hooks:
  – beforeMigrate
  – afterMigrate
  – beforeClean
  – afterClean
  – beforeRepair
  – afterRepair
  – …

infonova

Other flyway commands

- Clean
- Validate
- Repair
- Baseline

infonova

Our process - Pre Flyway

- Database model as separate project with its own lifecycle
- Process steps:
  1. Run changes in sandbox DB
  2. Generate diff using in-house developed tool
  3. Run Jenkins job to update version of DB model project
  4. Manually update the files in the DB model project
  5. Commit changes in DB model project
  6. Run Jenkins job to generate DB artifacts
  7. Check Nexus
  8. Update dbversion property in main application
  9. Ops deploy DB changes using a Rundeck job and then deploy application WAR if all went well

Our process - With Flyway

- Test SQL in sandbox schema
- Create V5_15__New_features.sql in resources/db/migration
- Commit

Challenges

- Long running upgrades (zero-downtime deployment?)
- Deleting of data
- Human factor
- Resistance in organization
- Clashes in versions

infonova

References

- P. M. Duvall, S. Matyas, A. Glover, *Continuous Integration,* Addison-Wesley, 2007
- Flyway Documentation (https://flywaydb.org/documentation/)
- N. Köbler, Kontinuierliche Datenbankmigration mit Liquibase und Flyway, Heise.de, 2013 (http://www.heise.de/developer/artikel/Kontinuierliche-Datenbankmigration-mit-Liquibase-und-Flyway-1857773.html)
- K. S. Allen, *Three Rules for Database Work*, Ode-to-code blog, 2008 (http://odetocode.com/blogs/scott/archive/2008/01/30/three-rules-for-database-work.aspx)

infonova

(we are hiring in Graz and Vienna:
http://www.infonova.com/jobs)