Amplicode

HAULMONT

Переписывая Историю

От инструментов версионирования БД к практике

Александр Шустанов

// WHO AM I?

Александр Шустанов

Product Manager at Haulmont

@alex_shust

10+
years of experience in software development

Java, Kotlin, Backend, IDEA plugins

Computer vision models, generative models

Amplicode

Версионирование БД
Как и зачем?

- **Ручные изменения**
  Прямое внесение изменений в базу данных без контроля версий
- **Autoddl**
  Автоматическое создание и изменение схемы на основе модели
- **Flyway || Liquibase**
  - Spring Boot
  - CLI
  - Миграции

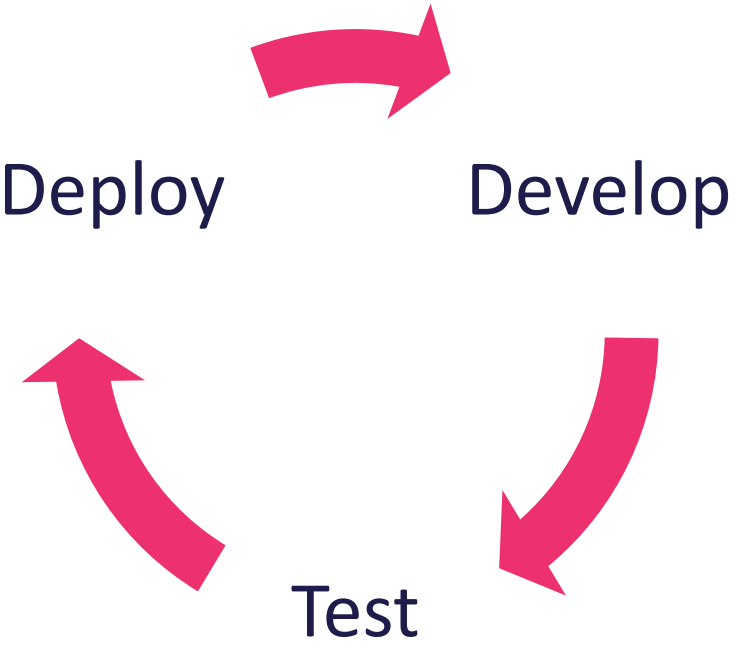
Версионирование БД
Как и зачем?

- **Контроль**
  Явное управление изменениями, вместо неявных изменений через ORM
- **Консистентность**
  Одни и те же изменения на разных средах (разработка, тестирование, продакшн)
- **Автоматизация**
  Интеграция с CI/CD, автоматическое применение изменений
- **Версионирование**
  Отслеживание изменений и возможность отката

Как рождаются и умирают миграции?

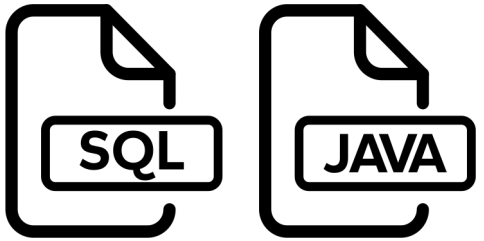
Кто-то дает разработчику задачу, реализовать …

1. Разработчик создает новую ветку
2. Создает новые сущности/редактирует старые
3. Запускает приложение
4. Пишет миграционный скрипт
5. Тестирует миграционный скрипт
6. …
7. PROFIT



// Как назвать и куда положить?

```
V12__description.sql
```




```
spring.flyway.locations=classpath:db/migration,classpath:db/data
```

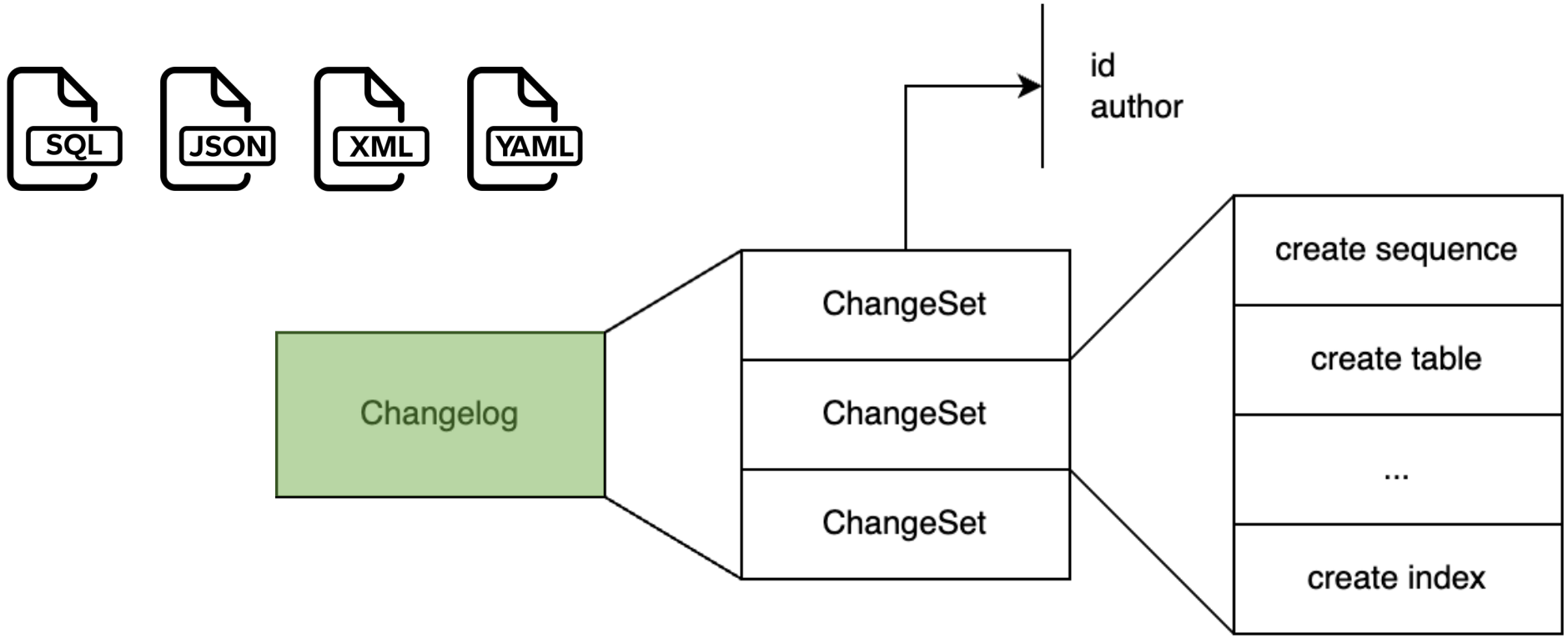
```
resources/db/migration/V1__init_schema.sql
resources/db/data/V1_1__sample_data.sql
resources/db/migration/V2__user_table.sql
...
```

// Как назвать и куда положить?

```
spring.flyway.locations=classpath:db/migration/release*


db/migration
    release1.0
        V1_0_0__createBaseStructure.sql
        V1_0_1__createDefaultUser.sql
    release1.2
        V1_2_0__createMoreTables.sql
```

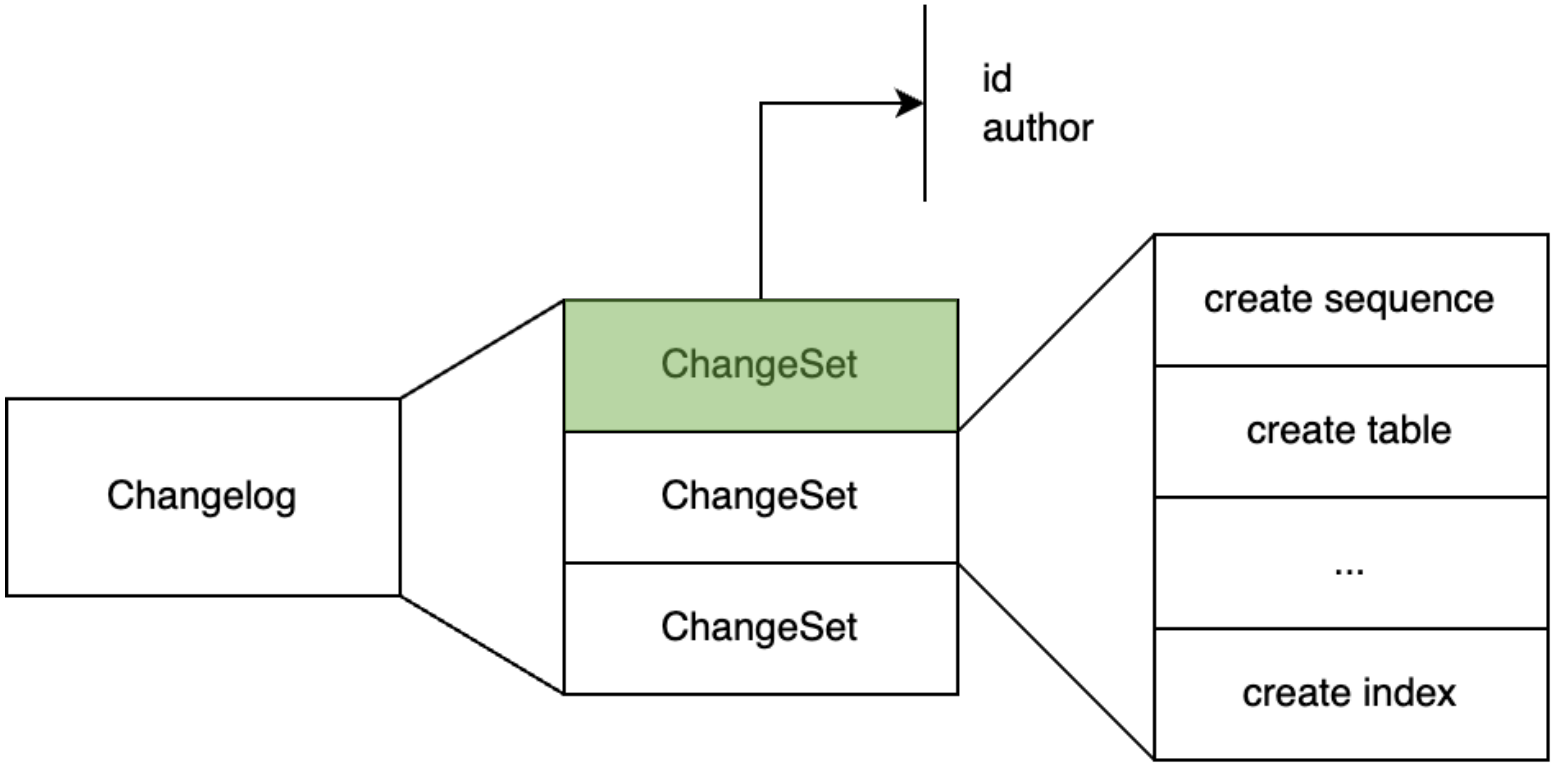
Как назвать и куда положить?

```
spring.liquibase.change-log=classpath:db/changelog/db.changelog-master.yaml
```



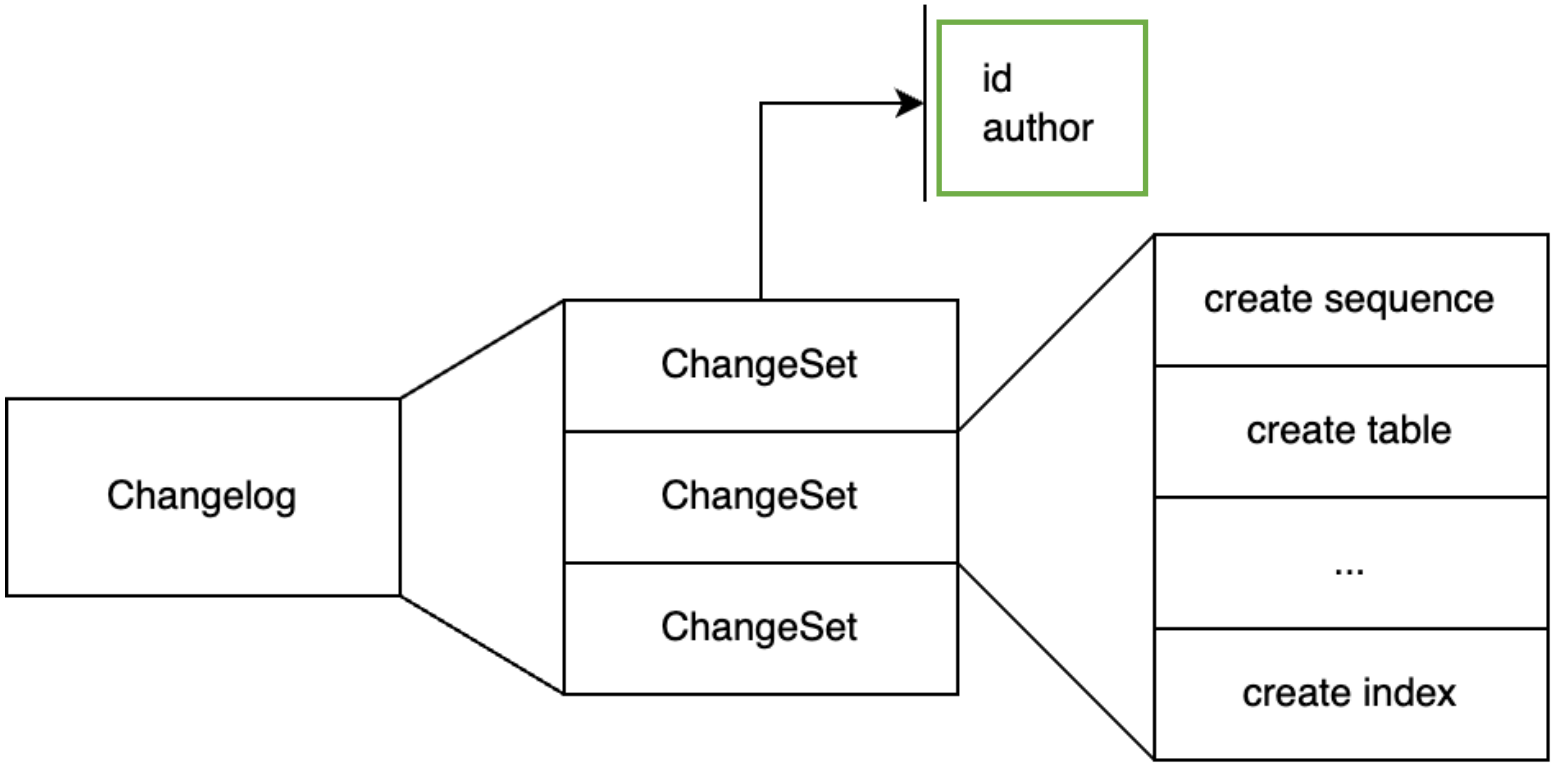
Как назвать и куда положить?

```
spring.liquibase.change-log=classpath:db/changelog/db.changelog-master.yaml
```



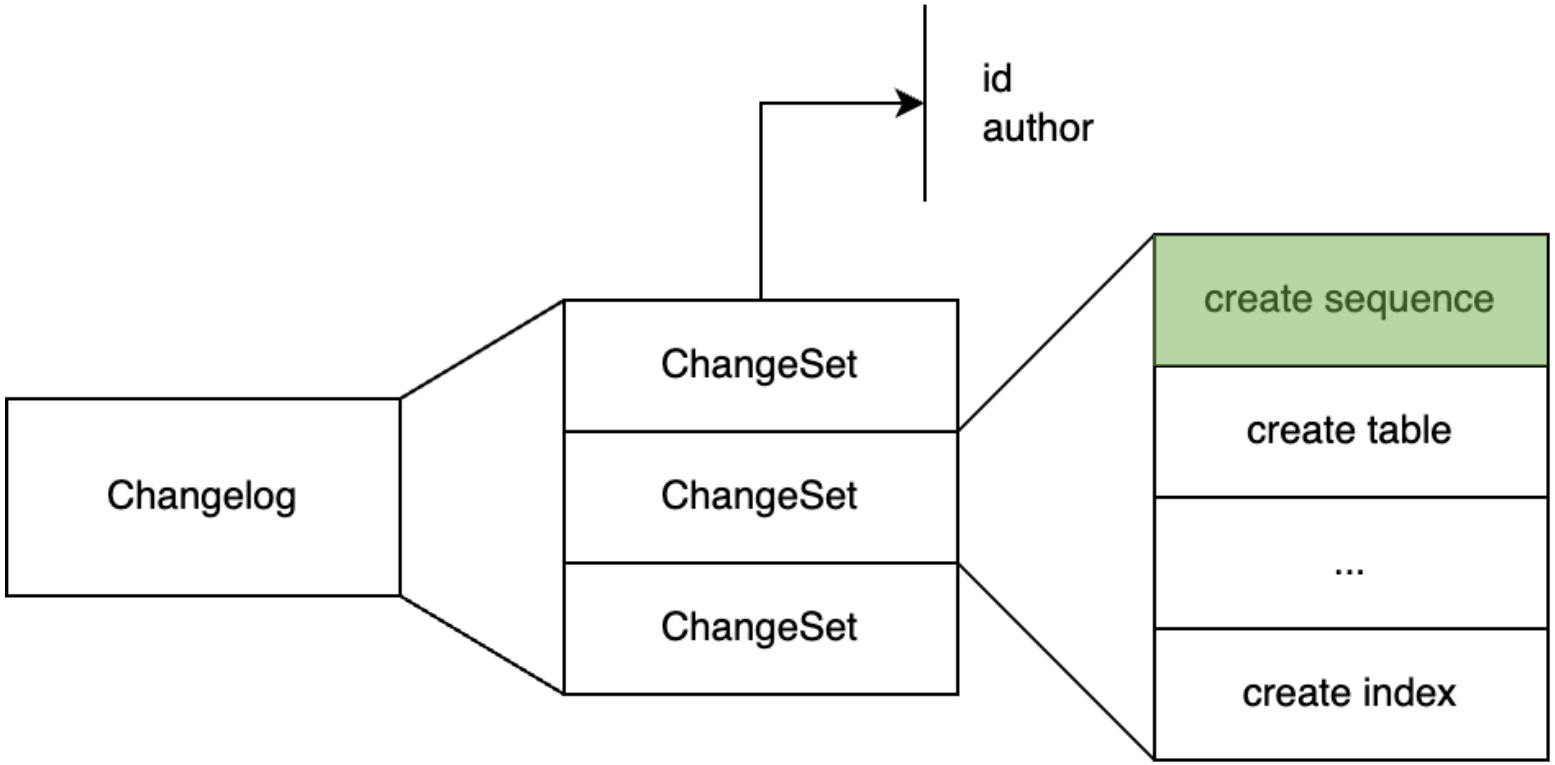
Как назвать и куда положить?

```
spring.liquibase.change-log=classpath:db/changelog/db.changelog-master.yaml
```



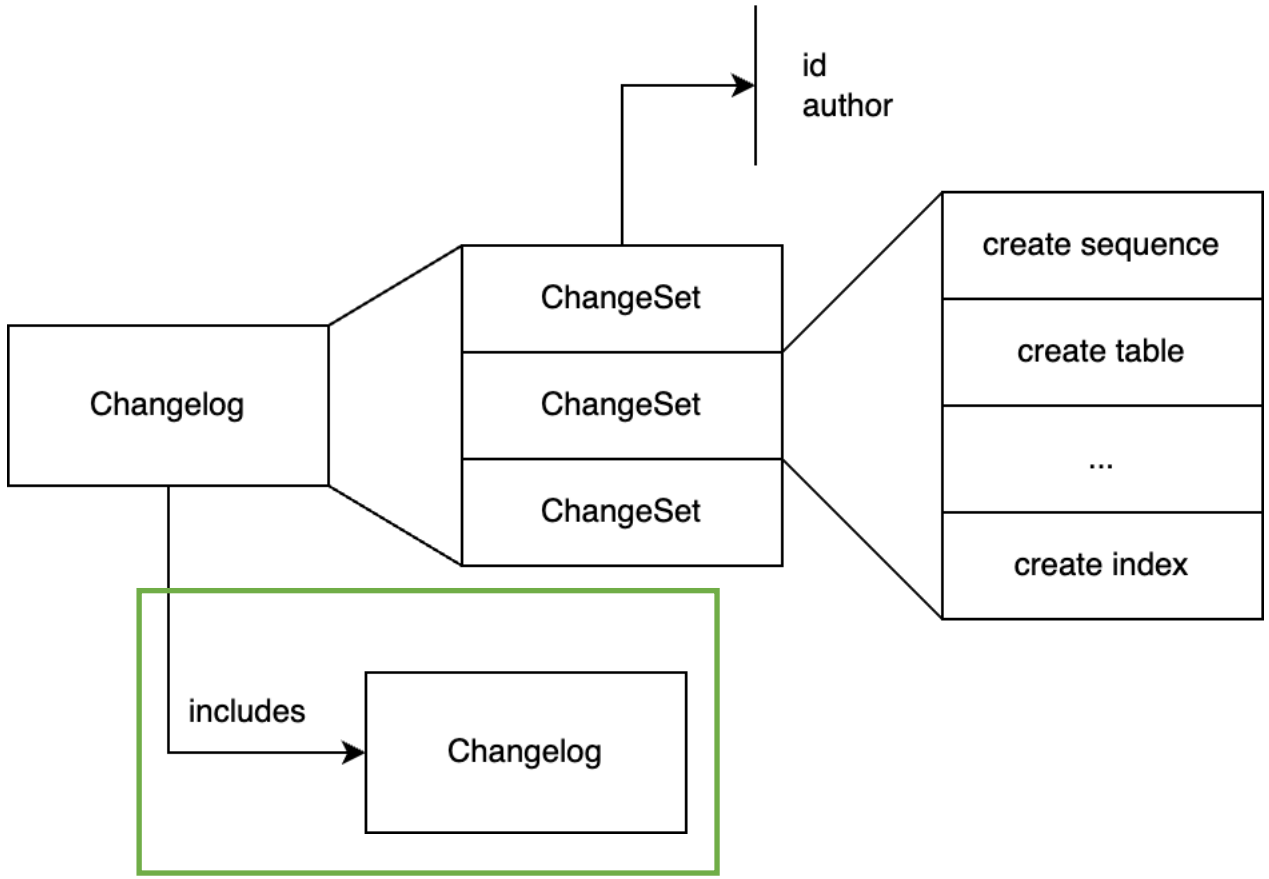
Как назвать и куда положить?

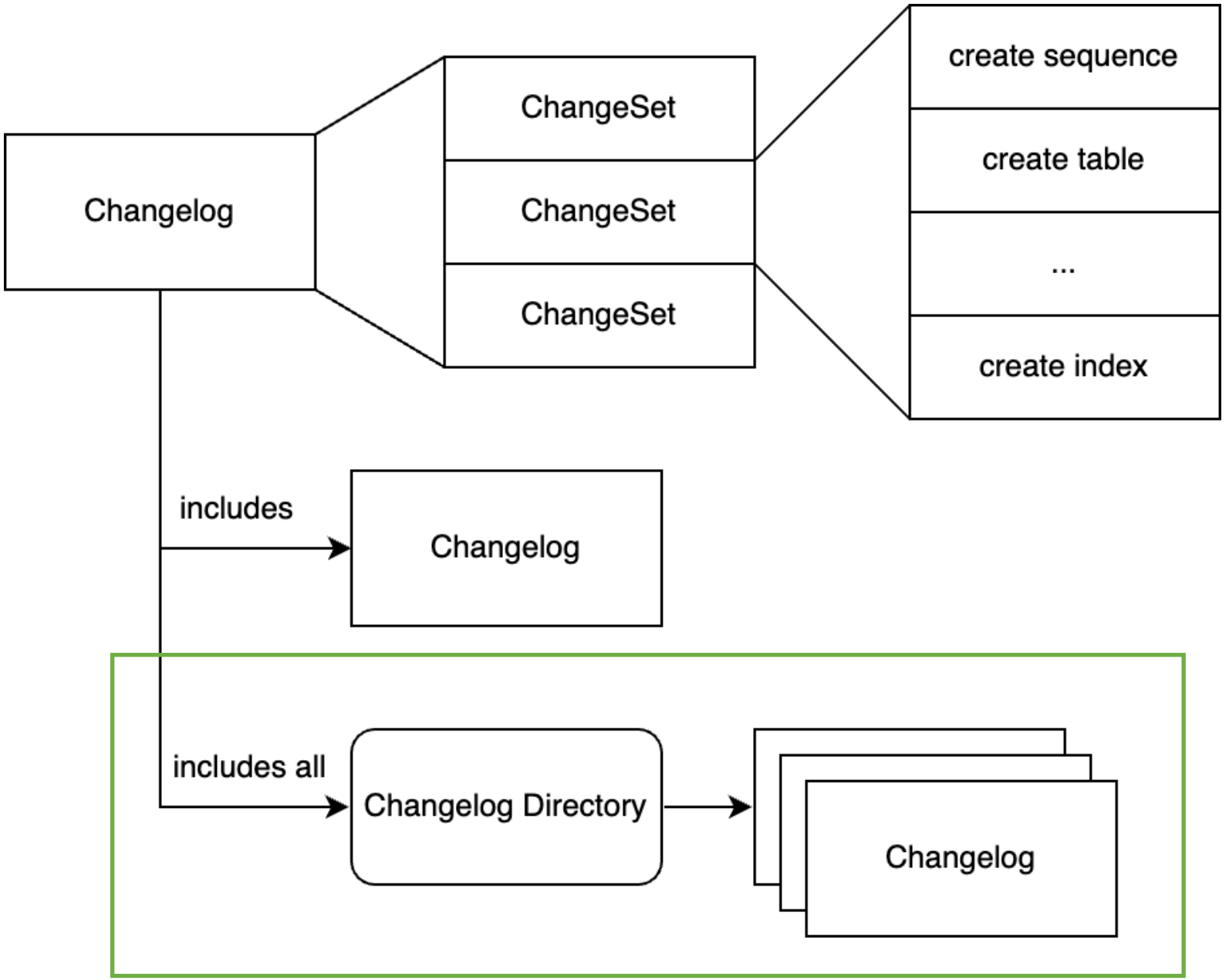
```
spring.liquibase.change-log=classpath:db/changelog/db.changelog-master.yaml
```



Как назвать и куда положить?

```
spring.liquibase.change-log=classpath:db/changelog/db.changelog-master.yaml
```




Changelog
ChangeSet
ChangeSet
ChangeSet
create sequence
create table
...
create index
includes
Changelog
includes all
Changelog Directory
Changelog

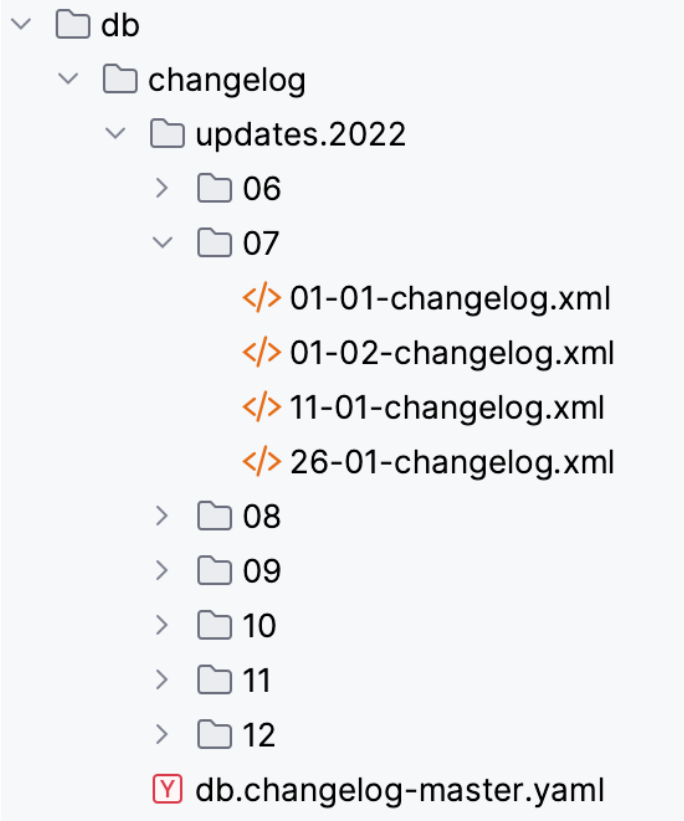
// Как назвать и куда положить?

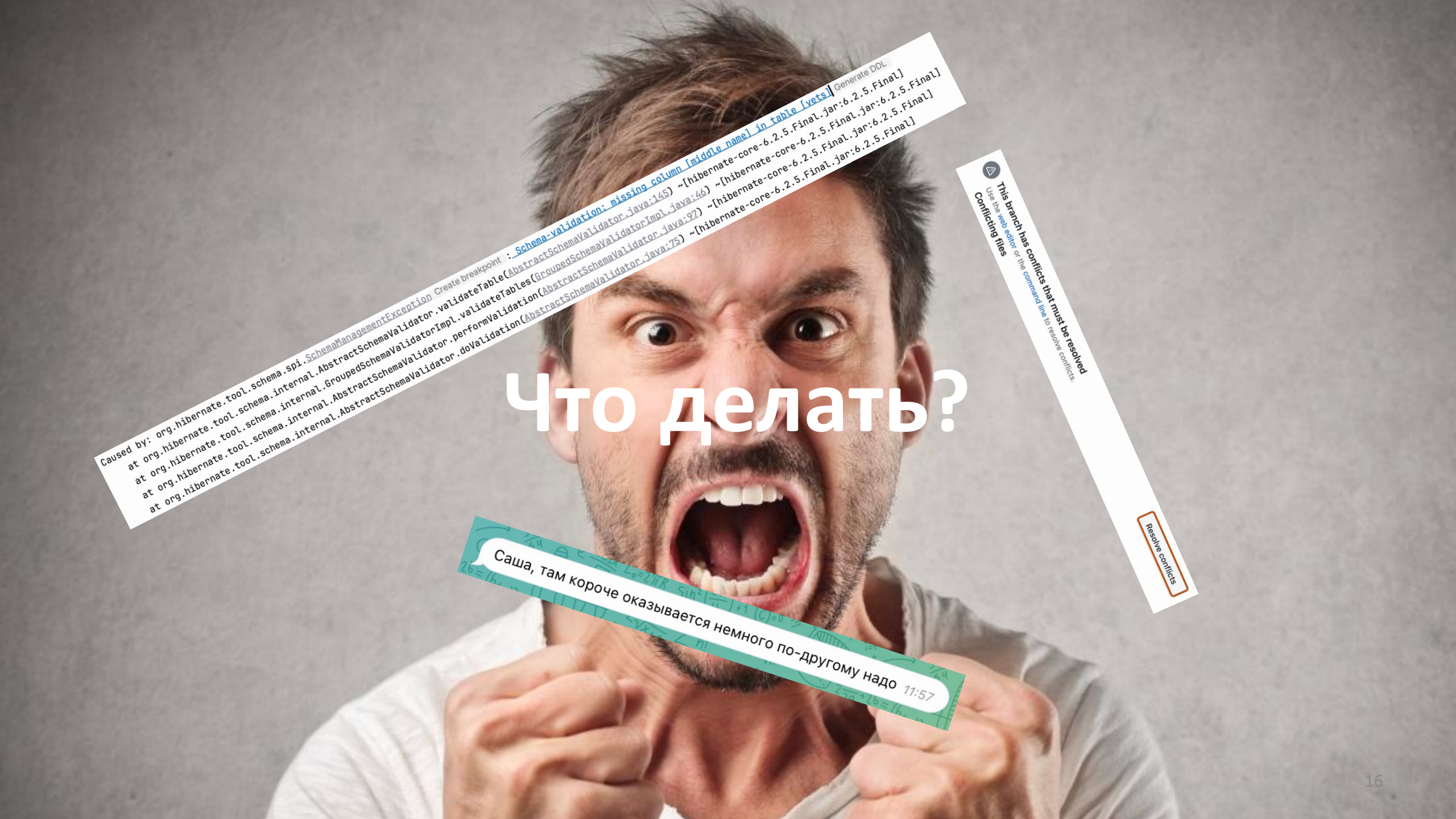
```yaml
databaseChangeLog:
  - changeSet:
      id: "create department"
      author: "alexandr.shustanov"
      changes:
        - createTable:
            catalogName: department
            columns:
              - column:
                  name: address
                  type: varchar(255)

  - includeAll:
      relativeToChangelogFile: true
      path: updates
```

Как назвать и куда положить?

- Все в кучу
- Ручное добавление каждого ченджлога
- Разбиение по релизам
- Разбиение по фичам
- Разбиение по датам

```yaml
databaseChangeLog:
  - includeAll:
      path: updates
      relativeToChangelogFile: true
```




Caused by: org.hibernate.tool.schema.spi.SchemaManagementException Create breakpoint : Schema-validation: missing column [middle_name] in table [vets] Generate DDL
at org.hibernate.tool.schema.internal.AbstractSchemaValidator.validateTable(AbstractSchemaValidator.java:145) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
at org.hibernate.tool.schema.internal.GroupedSchemaValidatorImpl.validateTables(GroupedSchemaValidatorImpl.java:46) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
at org.hibernate.tool.schema.internal.AbstractSchemaValidator.performValidation(AbstractSchemaValidator.java:97) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
at org.hibernate.tool.schema.internal.AbstractSchemaValidator.doValidation(AbstractSchemaValidator.java:75) ~[hibernate-core-6.2.5.Final.jar:6.2.5.Final]
Что делать?
Саша, там короче оказывается немного по-другому надо 11:57
This branch has conflicts that must be resolved
Conflicting files
Use the web editor or the command line to resolve conflicts.
Resolve conflicts

Редактируем скрипт

Migration checksum mismatch for migration
version 1

-> Applied to database : -1409205480

-> Resolved locally : -1776834826

	installed_rank	version	description	type	script	checksum
1	1	1		SQL	V1__.sql	-1409205480

installed_by	installed_on	execution_time	success
postgres	2023-09-17 13:12:51.…	8	• true

	id	author	filename	dateexecuted	orderexecuted	exectype
1	empty-example	liquibase-docs	db/changelog/db.changelog-master.yaml	2023-09-28 12:29:…	1	EXECUTED

md5sum	description	comments	tag	liquibase	contexts	labels	deployment_id
8:d0d3328f3d5a050…	empty		<null>	4.20.0	<null>	<null>	5889772587

// Сносим базу

- Исправляем скрипт
- Сносим базу
- Восстанавливаем ~~из бекапа~~ по миграционным скриптам
- REPEAT

// Редактируем скрипт

- Исправляем скрипт
- Делаем ручной откат
- Удаляем запись о миграции

- Исправляем скрипт
- Делаем ручную миграцию
- Правим чексумму

+ Быстро и красиво

- Вероятность накосячить

Невидимые расхождения

- Забыли удалить unique/nonnull колонку – не вставляется строка
- Не удалили индекс – падение производительности
- Не добавили констрейнт – получили неконсистентные данные

// Невидимые расхождения

- Забыли удалить unique/nonnull колонку – не вставляется строка
- Не удалили индекс – падение производительности
- Не добавили констрейнт – получили неконсистентные данные

```
@Column(name = "name", nullable = false)
private String name;
```

```
<addColumn tableName="person">
    <column name="name"/>
</addColumn>
```

// Невидимые расхождения

- Забыли удалить unique/nonnull колонку – не вставляется строка
- Не удалили индекс – падение производительности
- Не добавили констрейнт – получили неконсистентные данные

```
@Column(name = "name", nullable = false)
private String name;
```

```
<addColumn tableName="person">
    <column name="name"/>
</addColumn>
```

```
@Column(name = "name", nullable = true)
private String name;
```

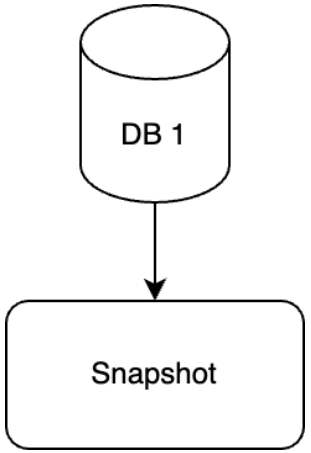
```
<addColumn tableName="person">
    <column name="name">
        <constraints nullable="false"/>
    </column>
</addColumn>
```

Автоматическая генерация скриптов

Все, что может быть сгенерировано, должно быть сгенерировано

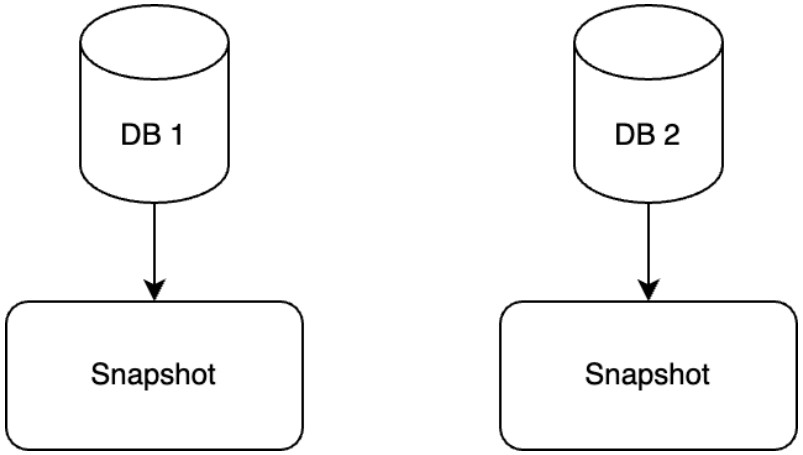
- Liquibase `diff-changelog`
- Liquibase Hibernate Plugin
- JPA Buddy

// Автоматическая генерация скриптов

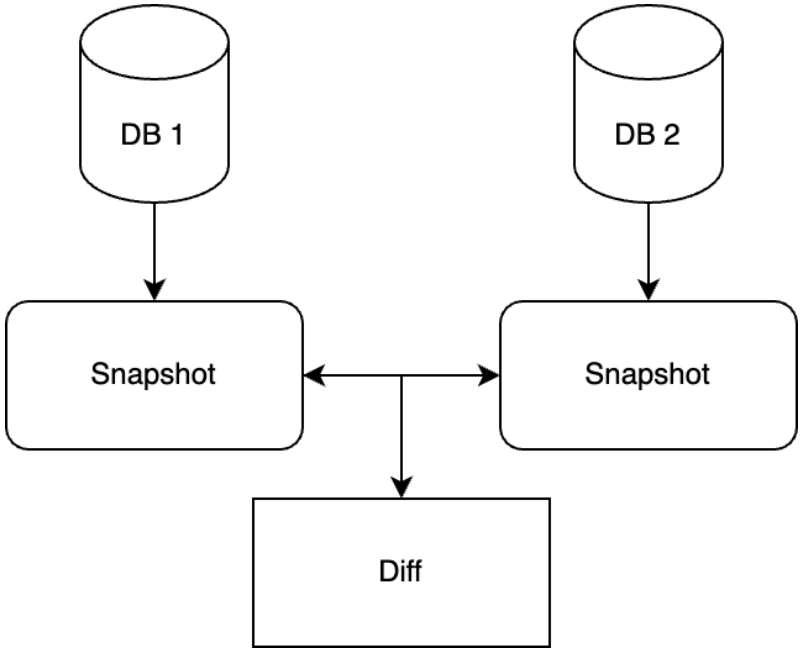



```
{
  "snapshot": {
    "created": "2023-09-25T20:43:23.348",
    "database": {"productVersion": "15.3 (Debian 15.3-1.pgdg120+1)"...},
    "objects": {
      "liquibase.structure.core.Catalog": [...],
      "liquibase.structure.core.Column": [...],
      "liquibase.structure.core.ForeignKey": [...],
      "liquibase.structure.core.Index": [...],
      "liquibase.structure.core.PrimaryKey": [...],
      "liquibase.structure.core.Schema": [...],
      "liquibase.structure.core.Table": [...],
      "liquibase.structure.core.UniqueConstraint": [...]
    },
    "snapshotControl": {...}
  }
}
```

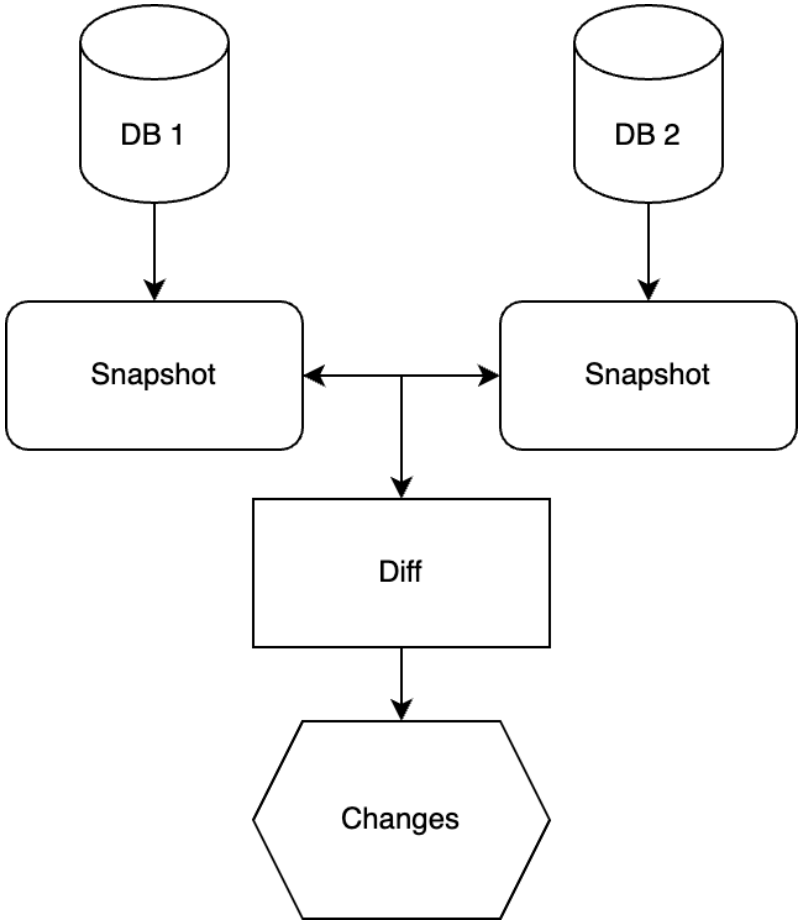
Автоматическая генерация скриптов



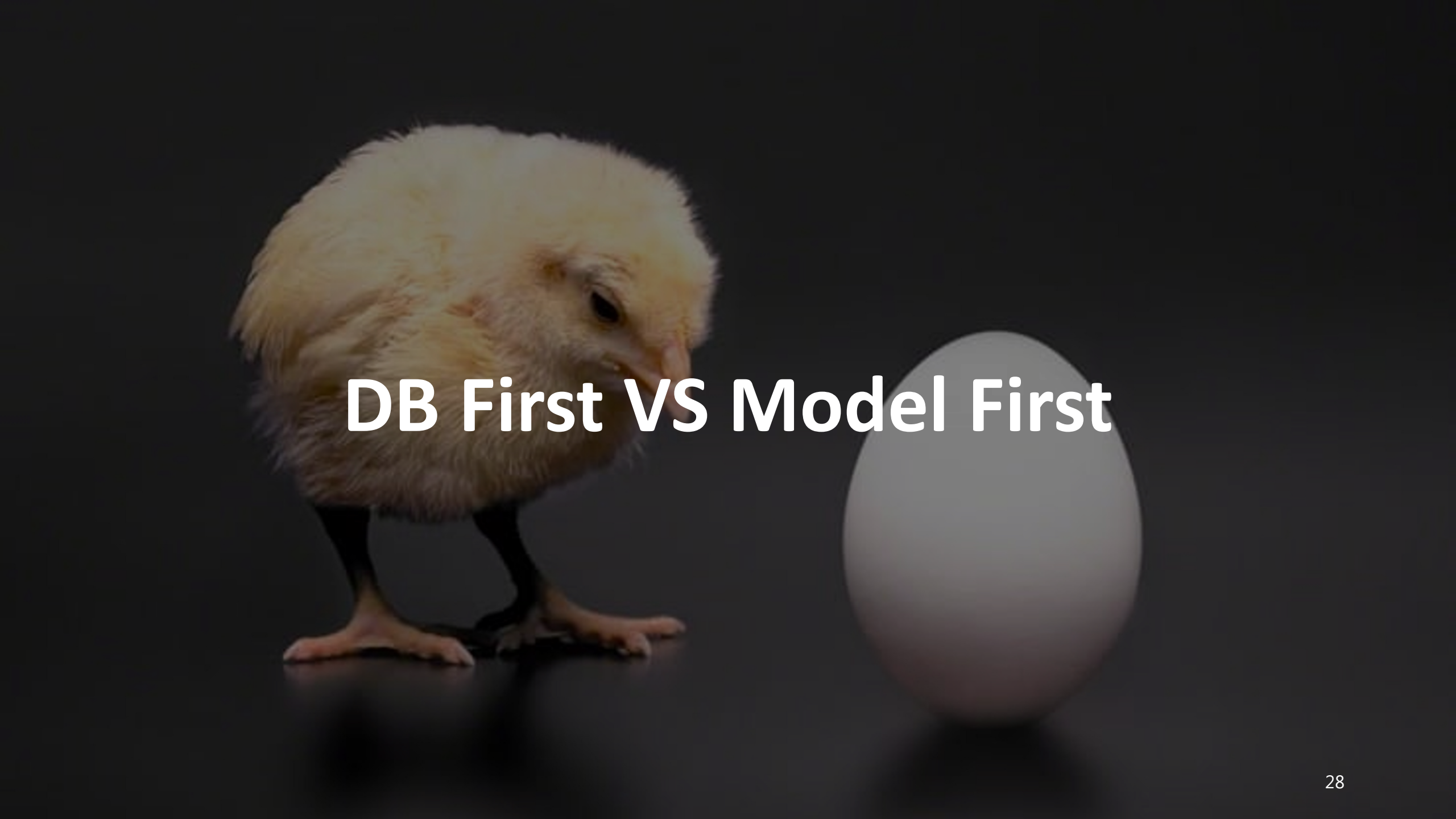
Автоматическая генерация скриптов



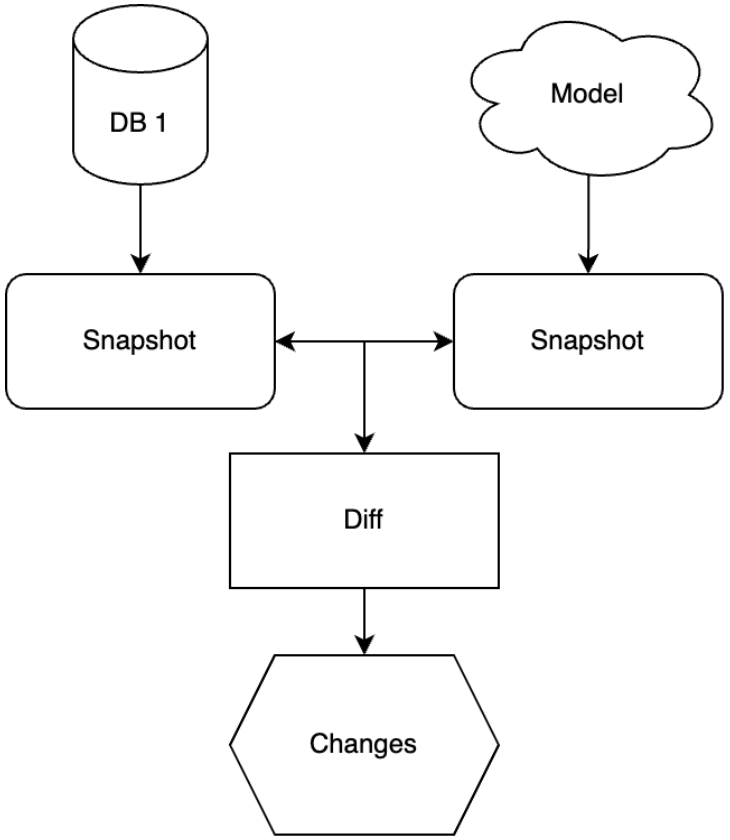
Автоматическая генерация скриптов



- Не забыть снять снапшот с базовой ветки
- Как-то получить новое состояние БД

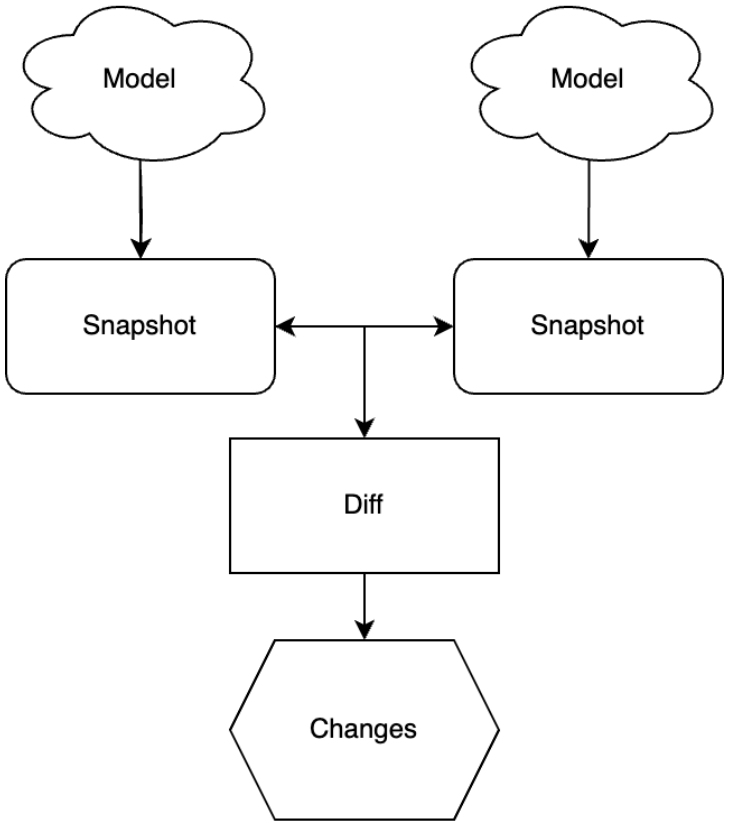

DB First VS Model First

Автоматическая генерация скриптов



А что вообще лежит в базе?

Автоматическая генерация скриптов



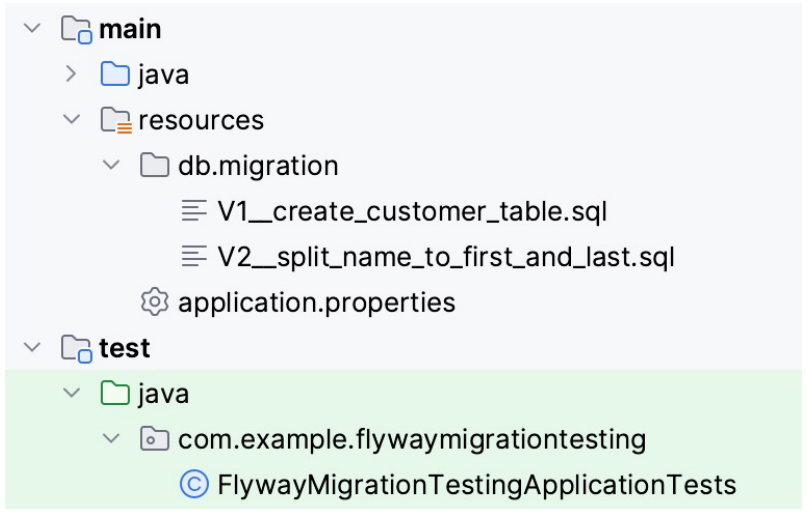
Amplicode

Автоматическая генерация скриптов

+ Быстрее и точнее
+ Покажет расхождения между моделью и схемой БД
- Работает только со схемой
- Ничего не знает о триггерах и хранимых процедурах
- Не знает как устранить многие диффы

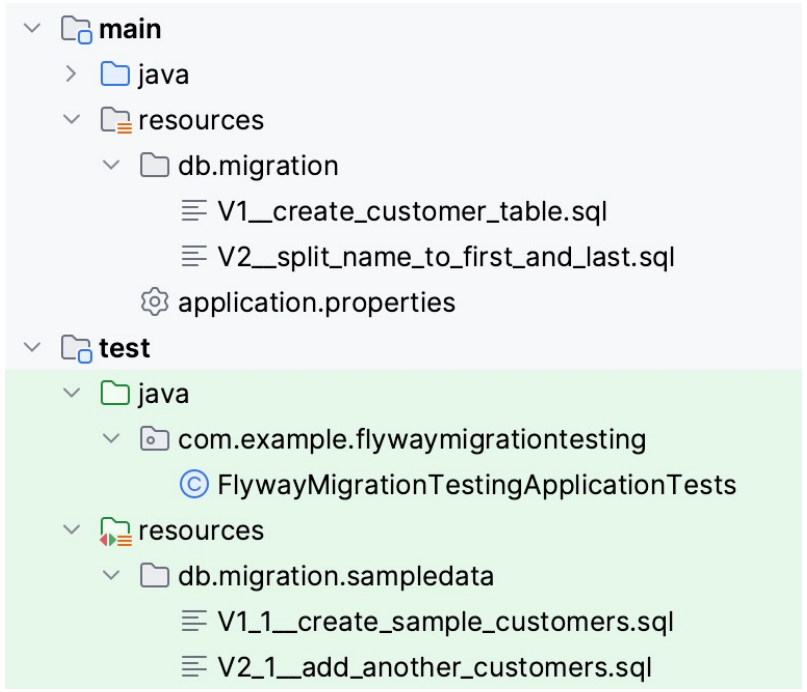
// Автоматическая генерация скриптов

// Тестирование миграций

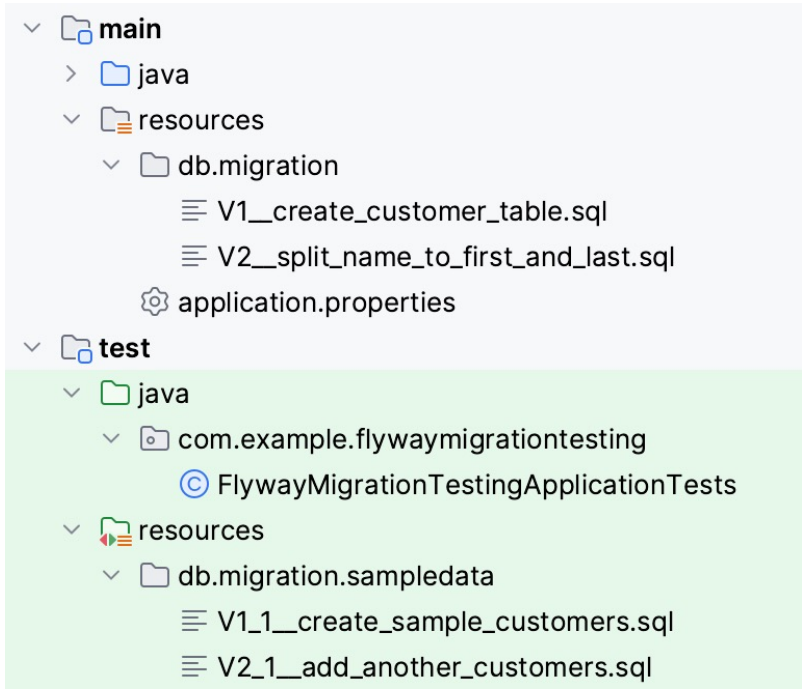



```
@SpringBootTest
@AutoConfigureEmbeddedDatabase(type =
AutoConfigureEmbeddedDatabase.DatabaseType.POSTGRES)
class FlywayMigrationTestingApplicationTests {

    @Test
    void testMigrations() {
        //do nothing
    }
}
```

Тестирование миграций

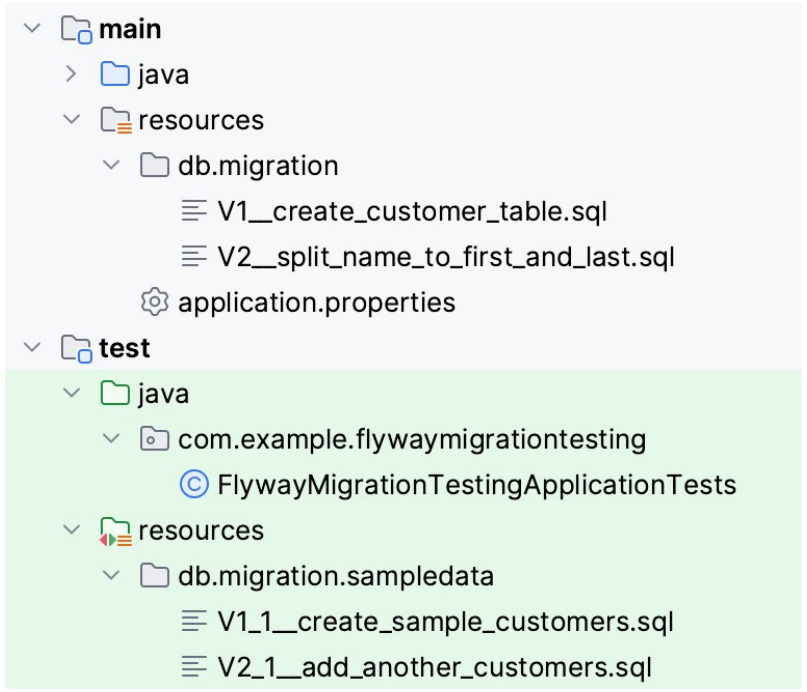


Тестирование миграций




```
@SpringBootTest(properties = {
      "spring.flyway.locations=classpath:db/migration,
classpath:db/migration/sampledata"
})
@AutoConfigureEmbeddedDatabase(type =
AutoConfigureEmbeddedDatabase.DatabaseType.POSTGRES)
class FlywayMigrationTestingApplicationTests {

    @Test
    void testMigrations() {
        //do nothing
    }
}
```

Тестирование миграций




```
@SpringBootTest(properties = {
      "spring.flyway.locations=classpath:db/migration,
classpath:db/migration/sampledata"
})
@AutoConfigureEmbeddedDatabase(type =
AutoConfigureEmbeddedDatabase.DatabaseType.POSTGRES)
class FlywayMigrationTestingApplicationTests {

    @Test
    void testMigrations() {
        //do nothing
    }
}
```

Migrating schema "public" to version "1 - create customer table"

Migrating schema "public" to version "1.1 - create sample customers"

Migrating schema "public" to version "2 - split name to first and last"

Migrating schema "public" to version "2.1 - add another customers"

Тестирование миграций

```
<changeSet id="create-customer" author="alexander.shustanov">
    <createSequence incrementBy="1" sequenceName="customer_seq" startValue="1"/>

    <createTable tableName="customer">
        <column name="id" type="BIGINT"/>
        <column name="first_name" type="VARCHAR(255)"/>
        <column name="last_name" type="VARCHAR(255)"/>
    </createTable>
</changeSet>


<changeSet id="create-customer::sample-data" author="alexander.shustanov”

          context="test" >

    <insert tableName="customer">
        <column name="id" valueComputed="nextval('customer_seq')"/>
        <column name="first_name">John</column>
        <column name="last_name">Doe</column>
    </insert>
</changeSet>
```

Тестирование миграций

```
<changeSet id="create-customer::sample-data"
           author="alexander.shustanov"
           context="test">
    <loadData tableName="customer"
              file="test-customers.csv"
              relativeToChangelogFile="true"/>
</changeSet>
```

Тестирование миграций

```
<changeSet id="create-customer::sample-data"
           author="alexander.shustanov"
           context="test">
    <loadData tableName="customer"
              file="test-customers.csv"
              relativeToChangelogFile="true"/>
</changeSet>
```

test-customers.csv

```
id                    ,first_name ,last_name
nextval('customer_seq'),John      ,Doe
nextval('customer_seq'),Joan      ,Doe
```

// Управляем запуском миграций: Contexts & Labels

```
<changeSet id="1" author="example" context="pro AND (shopping_cart OR ...)">
```

```
~ liquibase update --contexts=pro,shopping_cart,bigco
```

// Управляем запуском миграций: Contexts & Labels

```
<changeSet id="1" author="example" context="pro AND (shopping_cart OR ...)">
```

```
~ liquibase update --contexts=pro,shopping_cart,bigco
```

```
<changeSet id="1" author="example" labels="pro, shipping_cart">
```

```
~ liquibase update --labels="pro and (shopping_cart or account_mgmt)"
```

// Управляем запуском миграций: Contexts & Labels

```
<changeSet id="1" author="example" context="pro AND (shopping_cart OR ...)">
```

```
~ liquibase update --contexts=pro,shopping_cart,bigco
```

Contexts – вся "мощь" управления на стороне автора changelogs

```
<changeSet id="1" author="example" labels="pro, shipping_cart">
```

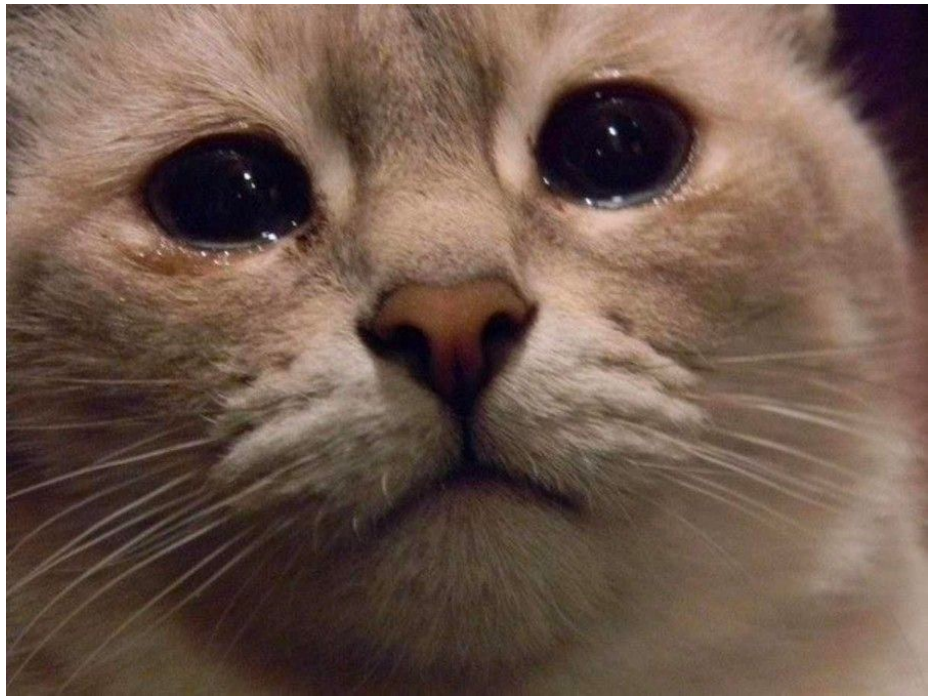
```
~ liquibase update --labels="pro and (shopping_cart or account_mgmt)"
```

Labels – вся "мощь" управления на стороне deployment manager

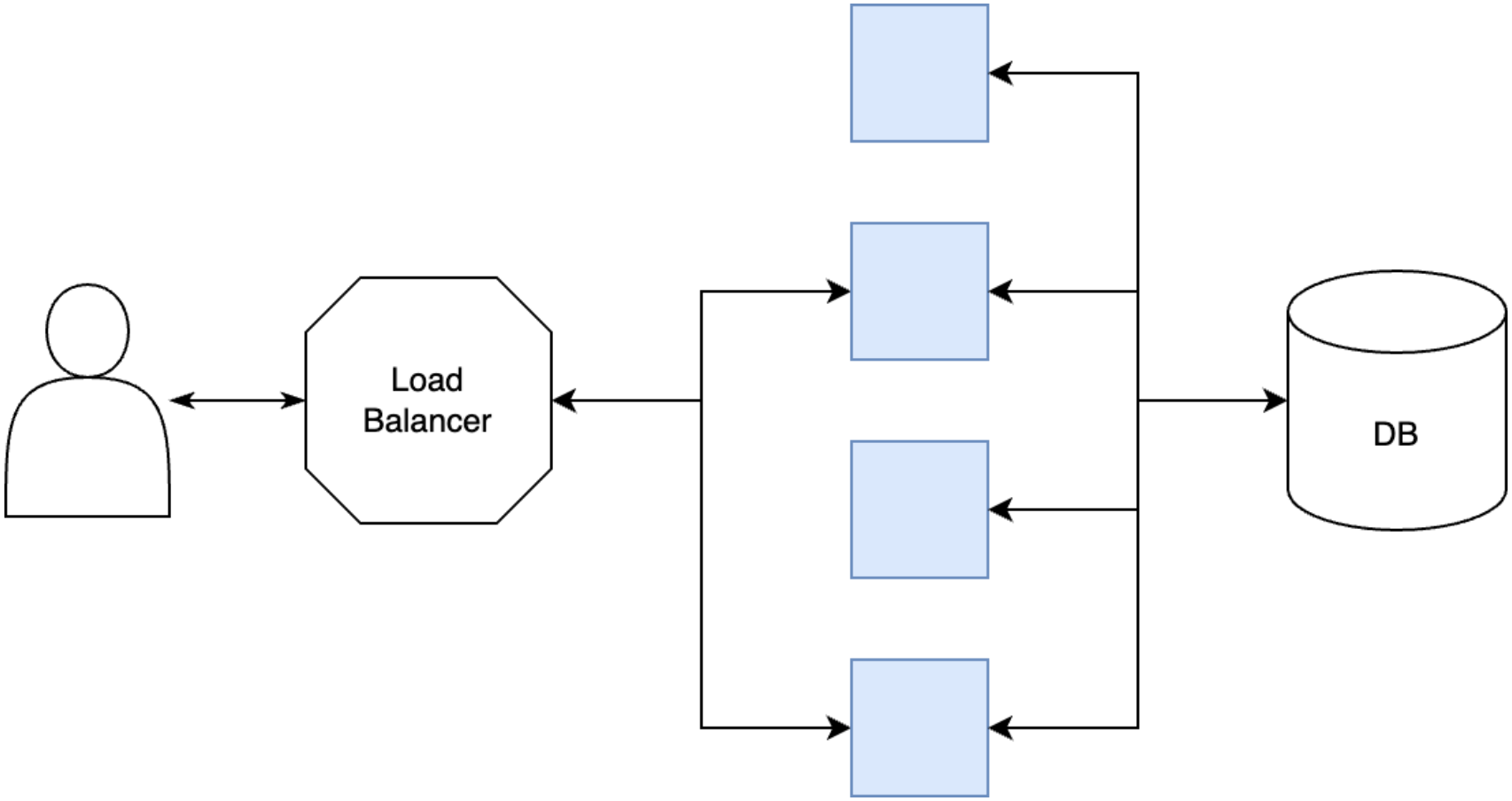
// Управляем запуском миграций

Раскидываем скрипты по разным папочкам

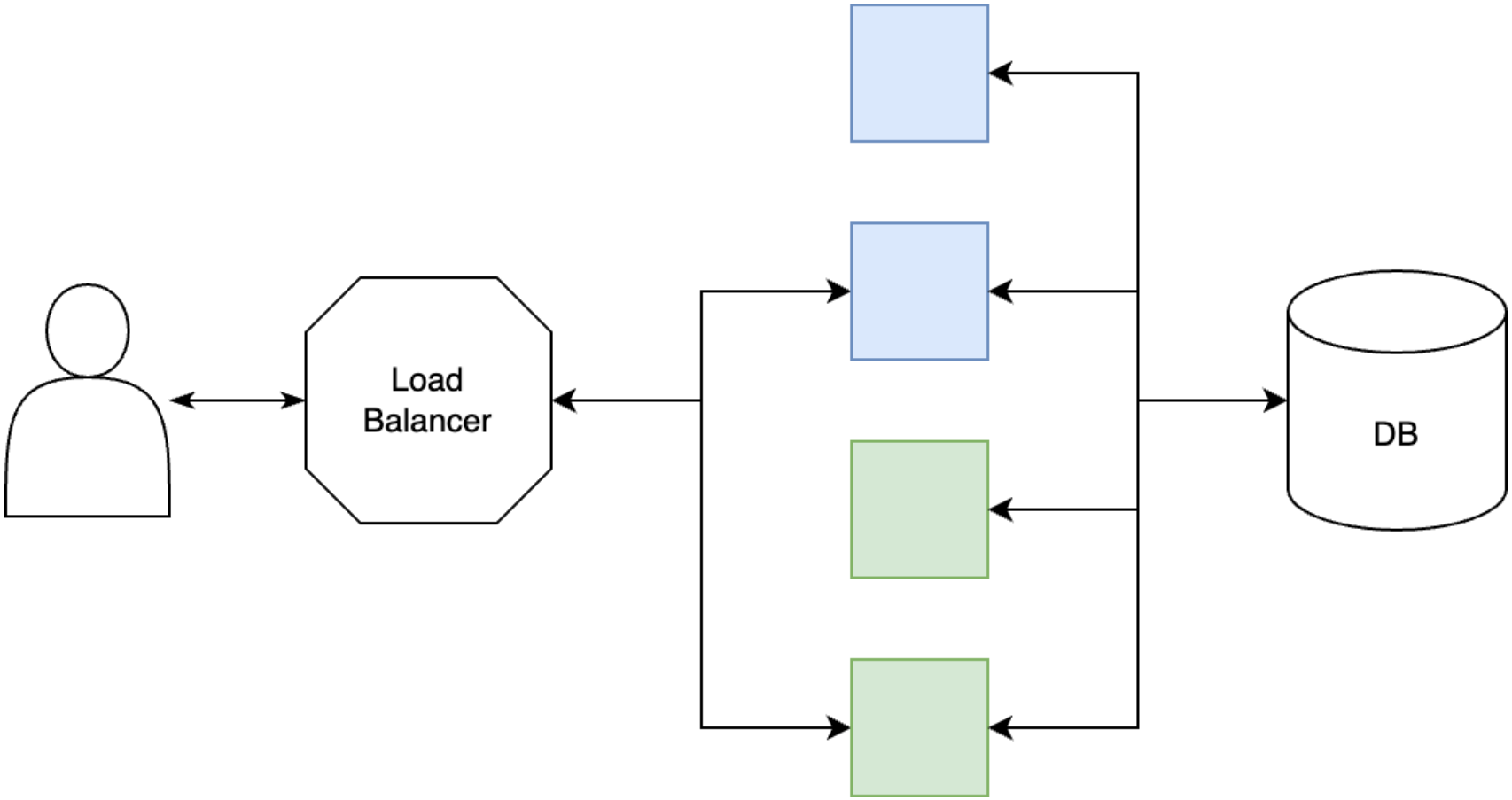
```
spring.flyway.locations=
      classpath:db/migration/...,
      ...
```



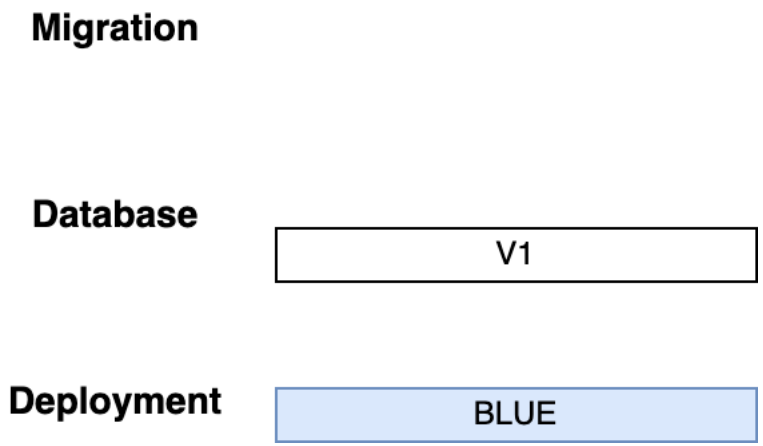
Zero downtime deployment (Blue-Green)



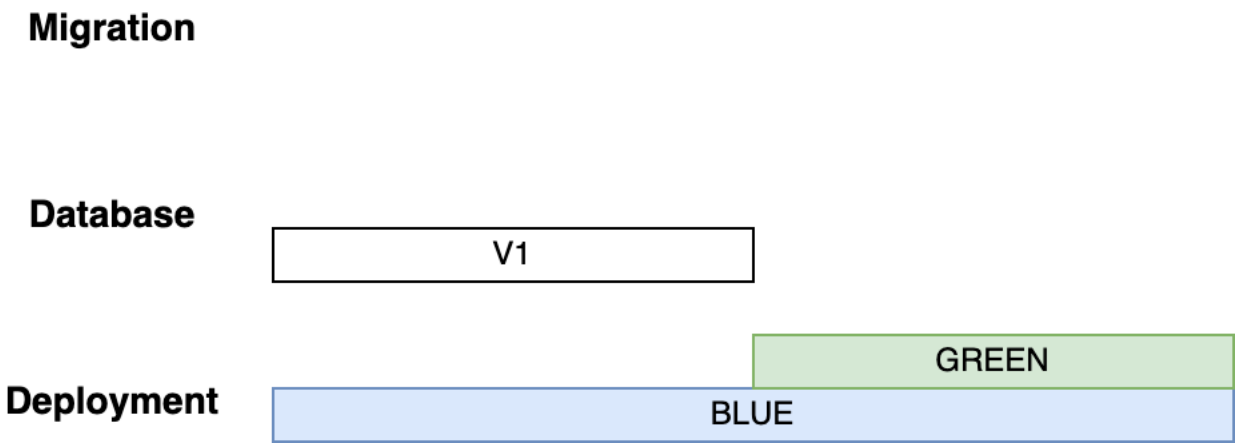
Zero downtime deployment (Blue-Green)



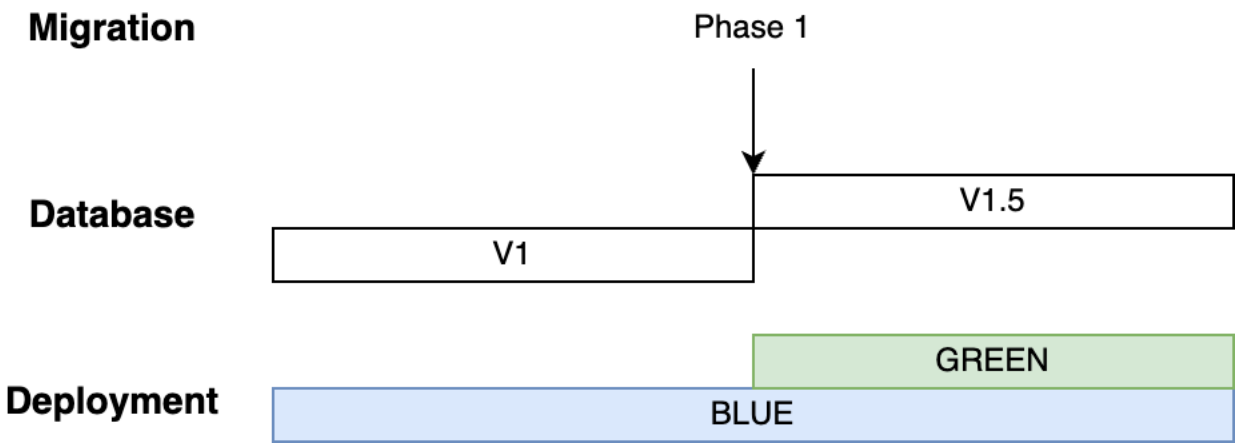
Zero downtime deployment (Blue-Green)



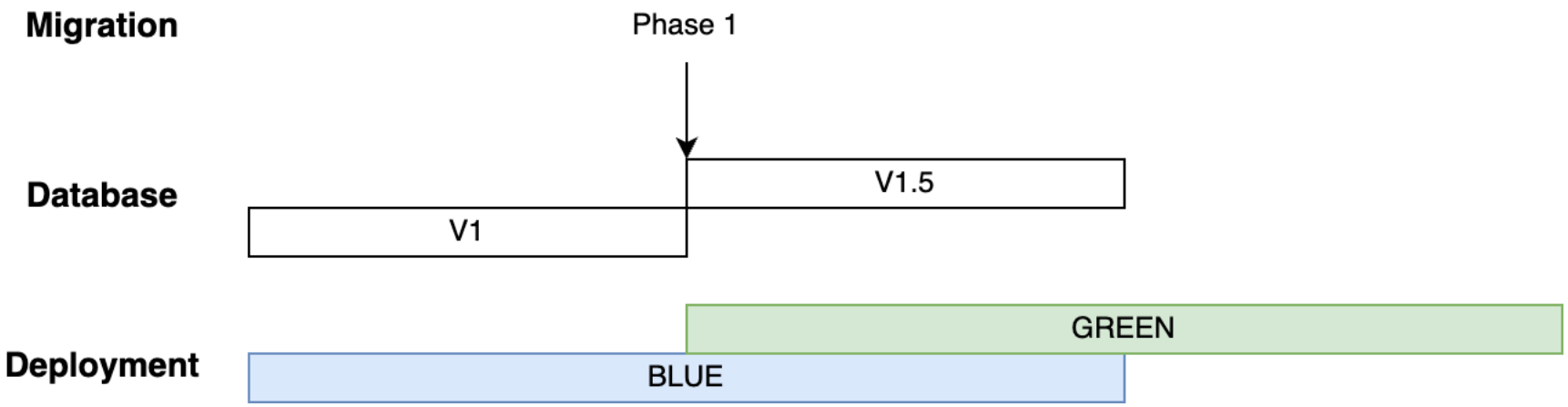
Zero downtime deployment (Blue-Green)



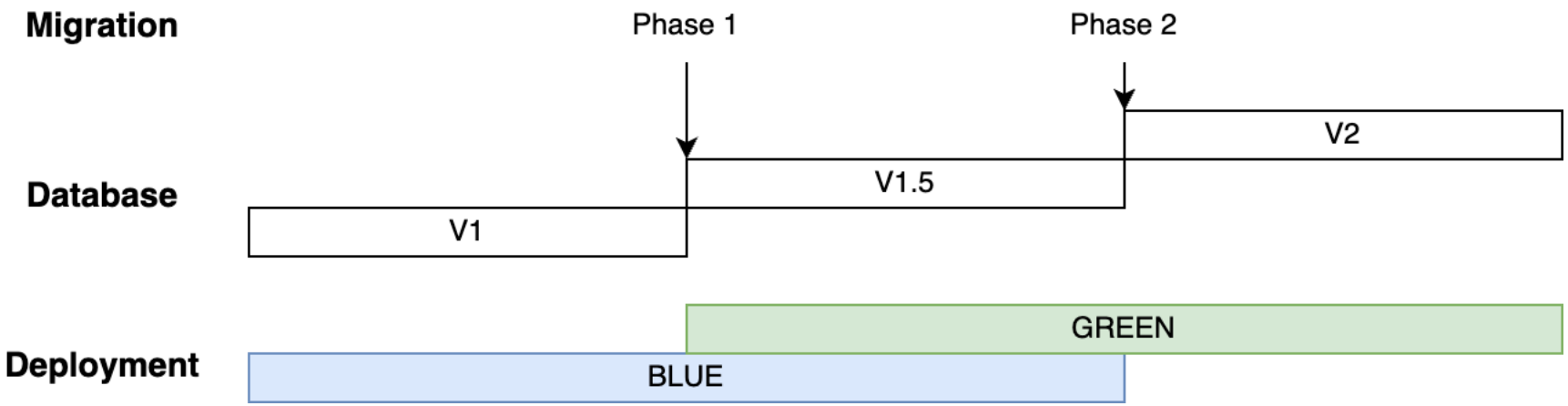
// Zero downtime deployment (Blue-Green)



// Zero downtime deployment (Blue-Green)



// Zero downtime deployment (Blue-Green)



// Zero downtime deployment (Blue-Green)

```
<changeSet id="delete-obsolete-columns"
           author="alexander.shustanov"
           context="phase2">
    ...
</changeSet>
```

// Rollback

```
~ liquibase rollback 1 --changelog-file='src/resources/db/changelog/db.changelog-master.xml' --url=...
```

CREATE -> DROP

RENAME -> RENAME BACK

DROP, DELETE, INSERT, UPDATE,… -> Write it yourself!

Rollback

```
<changeSet id="insert_default_admin" author="alexandr.shustanov">
    <insert tableName="users">
        <column name="username">admin</column>
        <column name="password">{noop}admin</column>
    </insert>




</changeSet>
```

// Rollback

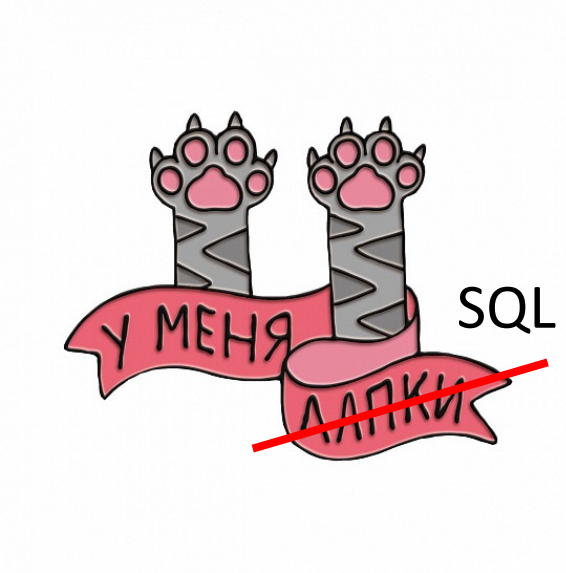
```
<changeSet id="insert_default_admin" author="alexandr.shustanov">
    <insert tableName="users">
        <column name="username">admin</column>
        <column name="password">{noop}admin</column>
    </insert>

    <rollback>
        <delete tableName="users">
            <where>username = admin</where>
        </delete>
    </rollback>
</changeSet>
```

// Undo

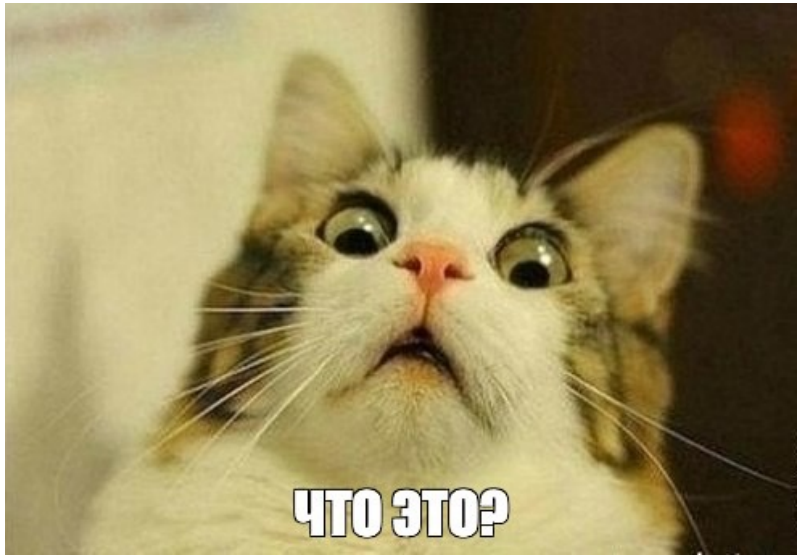
Доступно только в платной версии

Все надо писать вручную!

```
V12__my_awesome_changes.sql
U12__my_awesome_changes.sql
```



Liquibase Dry Run

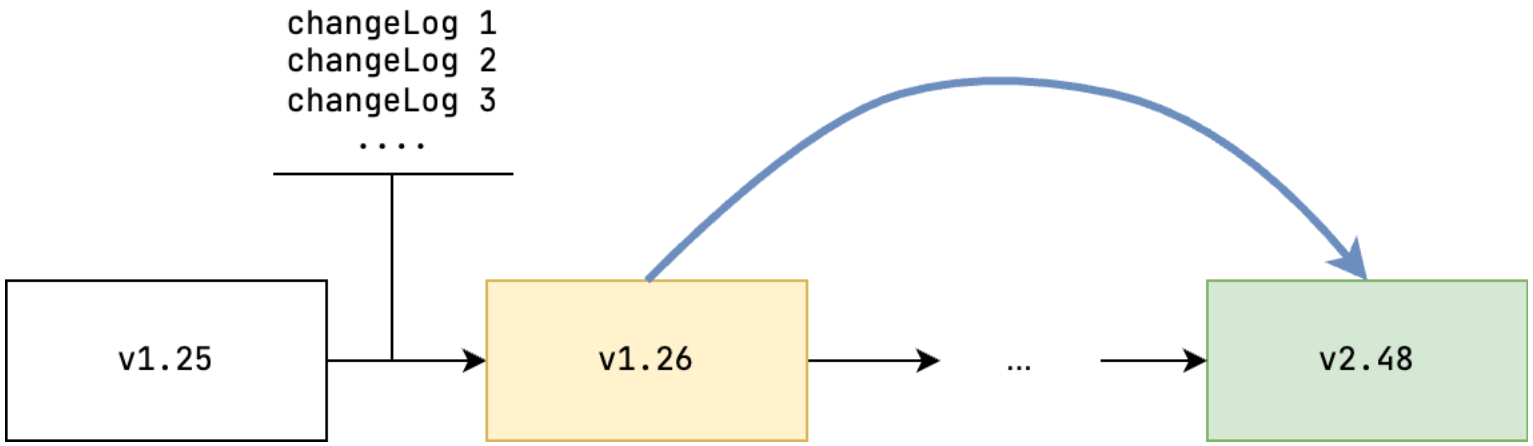



```
liquibase update-sql --changelog-file=changelog.xml --jdbcConnection=...
```

Amplicode

Продуктовые кейсы

// Давно не обновлялись?



Поддержка нескольких БД

```
<changeSet id="create-user-postgres"
           author="alexander.shustanov"
           dbms="postgresql">
    <createTable tableName="users">
        <column name="name" type="VARCHAR(255)"/>
        <column name="created_at" type="timestamp with time zone"/>
    </createTable>
</changeSet>

<changeSet id="create-user-mssql"
           author="alexander.shustanov"
           dbms="mssql">
    <createTable tableName="users">
        <column name="name" type="VARCHAR(255)"/>
        <column name="created_at" type="datetime offset"/>
    </createTable>
</changeSet>
```

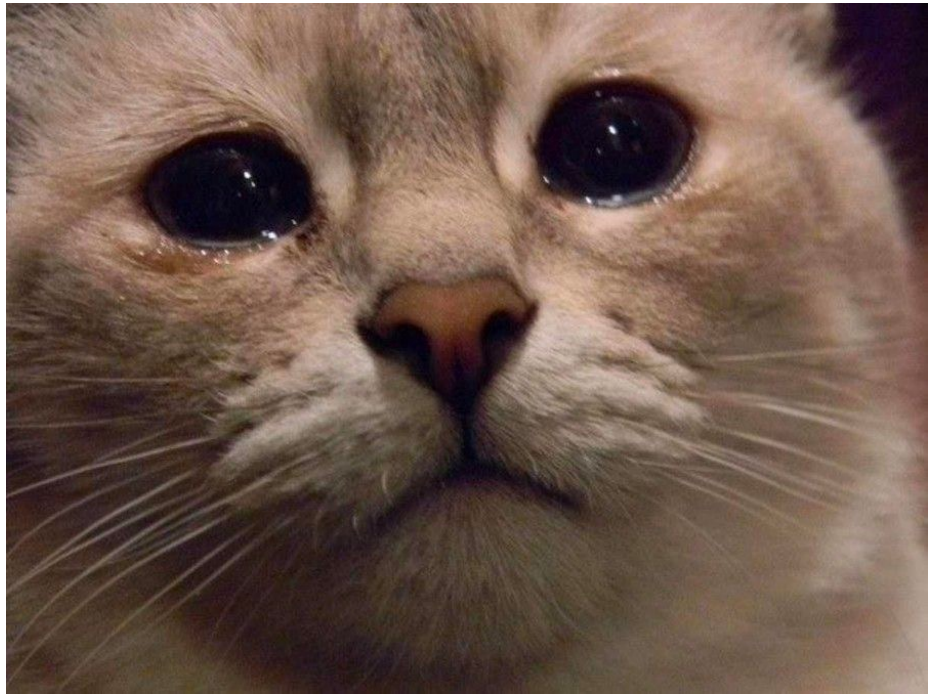
// Поддержка нескольких БД

```
<property name="datetime_zoned_type"
          value="timestamp with time zone"
          dbms="postgresql"/>

<property name="datetime_zoned_type"
          value="datetime offset"
          dbms="mssql" />


<changeSet id="create-user"
            author="alexander.shustanov">
    <createTable tableName="users">
        <column name="name" type="VARCHAR(255)"/>
        <column name="created_at" type="${datetime_zoned_type}"/>
    </createTable>
</changeSet>
```

// Поддержка нескольких БД

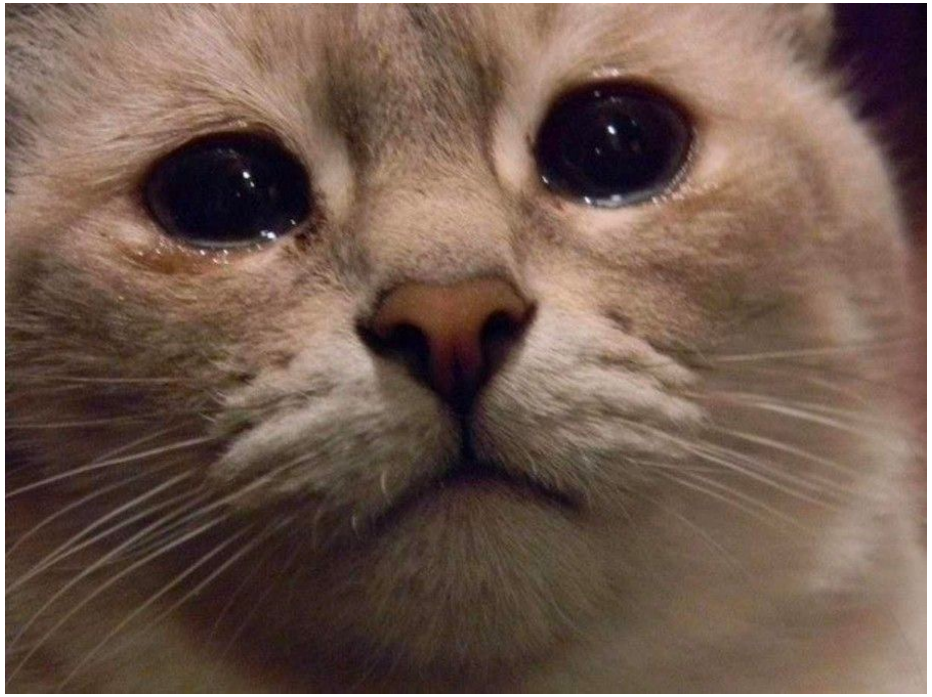
```
database=postgres
spring.flyway.locations=
  classpath:db/migration/${database}
```

```
db/migration
    postgres
        V1_0_0__createBaseStructure.sql
        V1_0_1__createDefaultUser.sql
    mssql
        V1_0_0__createBaseStructure.sql
         V1_0_1__createDefaultUser.sql
```



// Поддержка нескольких БД

```
database=postgres
spring.flyway.locations=
    classpath:db/migration/${database}/*

db/migration
    postgres
        release1.1
            …
        release1.2
            …
            sampledata
            …
    mssql
        ...
```



Amplicode

And even more

// Repeatable

```
<changeSet id="create-person-view"
           author="alexander.shustanov"
           runAlways="true">
    <createView viewName="...">...</createView>
</changeSet>


<changeSet id="create-serialization-procedure"
           author="alexander.shustanov"
           runOnChange="true">
    <sql>...</sql>
</changeSet>
```

```
R__create_person_view.sql
```

// Java-based migrations

`V12__JavaMigration.java`

```
package db.migration;


public class V2__JavaMigration extends BaseJavaMigration {
    @Override
    public void migrate(Context context) throws Exception {
        try (PreparedStatement statement =
                     context
                             .getConnection()
                             .prepareStatement("INSERT INTO test_user (name) VALUES
('Obelix')")) {
            statement.execute();
        }
    }
}
```

// Java-based migrations

```
public class XorColumn implements CustomTaskChange {
    private String table;
    private String column;
    private String xorKey;

    @Override
    public ValidationErrors validate(Database database) {
        ValidationErrors validationErrors = new ValidationErrors("xorColumn");
        validationErrors.checkRequiredField("table", table);
        validationErrors.checkRequiredField("column", column);
        validationErrors.checkRequiredField("xorKey", xorKey);
        return validationErrors;
    }

    @Override
    public void execute(Database database) throws CustomChangeException {
        // todo
    }

    ...
}
```

// Java-based migrations

```
public class XorColumn implements CustomTaskChange {
    private String table;
    private String column;
    private String xorKey;

    @Override
    public ValidationErrors validate(Database database) {
        ValidationErrors validationErrors = new ValidationErrors("xorColumn");
        validationErrors.checkRequiredField("table", table);
        validationErrors.checkRequiredField("column", column);
        validationErrors.checkRequiredField("xorKey", xorKey);
        return validationErrors;
    }

    @Override
    public void execute(Database database) throws CustomChangeException {
        // todo
    }

    ...
}
```

// Java-based migrations

```
public class XorColumn implements CustomTaskChange {
    private String table;
    private String column;
    private String xorKey;

    @Override
    public ValidationErrors validate(Database database) {
        ValidationErrors validationErrors = new ValidationErrors("xorColumn");
        validationErrors.checkRequiredField("table", table);
        validationErrors.checkRequiredField("column", column);
        validationErrors.checkRequiredField("xorKey", xorKey);
        return validationErrors;
    }

    @Override
    public void execute(Database database) throws CustomChangeException {
        // todo
    }

    ...
}
```

// Java-based migrations

```
<changeSet id="obfuscate-salary" author="alexandr.shustanov">
    <customChange class="liquibasemigrationtesting.XorColumn">




    </customChange>
</changeSet>
```

// Java-based migrations

```
<changeSet id="obfuscate-salary" author="alexandr.shustanov">

    <customChange class="liquibasemigrationtesting.XorColumn">

        <param name="table" value="customer"/>
        <param name="column" value="salary"/>
        <param name="xorKey"
               value="cSwqZGJ3ZWBOQmcvVClVQWZJQ2xmNA==">

    </customChange>

</changeSet>
```

- Гибкий
- Расширяемый
- Произвольные иерархии
- Больше функций в бесплатной версии
- Поддержка нескольких БД
- preConditions, postConditions
- And so on

- Более простой?
- Понятен DBA
- Единственная точка управления запуском

Спасибо за внимание!