

Эволюция данных

Синёв Сергей, CodeInside



Программа

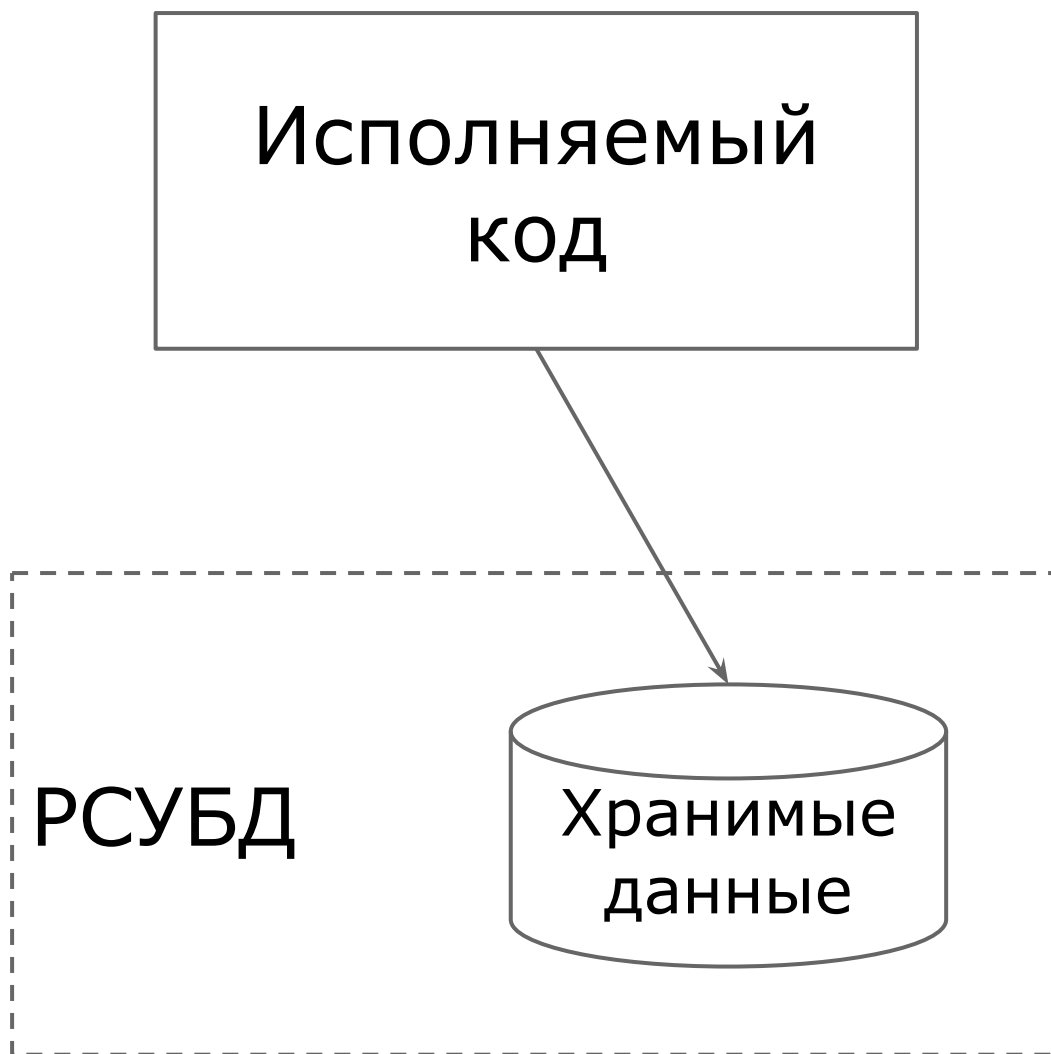
Исполняемый
код

```
graph TD; A[Исполняемый код] --> B[Хранимые данные];
```

The diagram consists of two rectangular boxes connected by a vertical arrow pointing downwards. The top box contains the text 'Исполняемый код' (Executable code) and the bottom box contains 'Хранимые данные' (Stored data).

Хранимые
данные

Программа для WEB



РСУБД

Atomicity

Атомарность

Consistency

Согласованность

Isolation

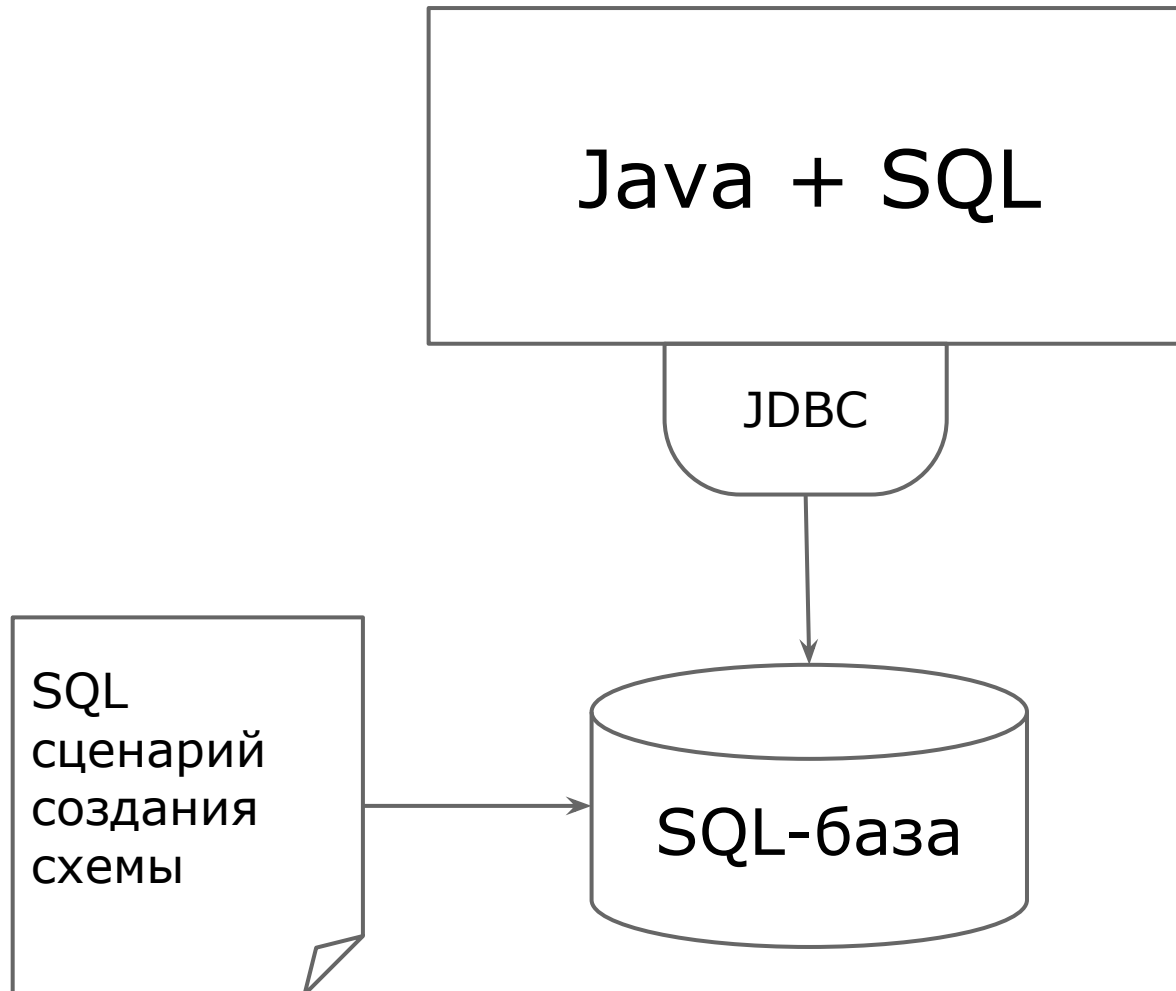
Изолированность

Durability

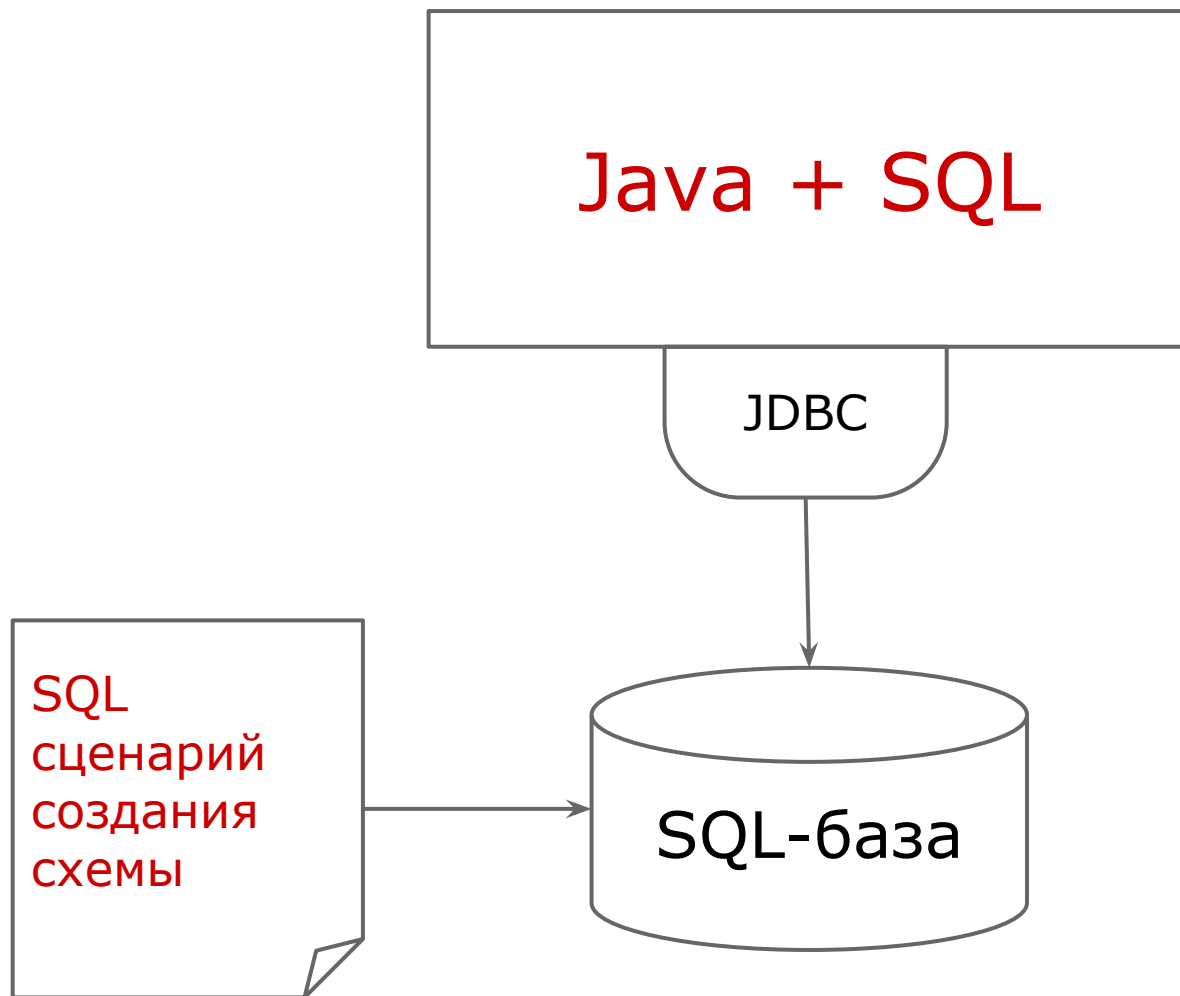
Долговечность

Транзакция - логическая группа последовательных операций для которых выполняются требования **ACID**

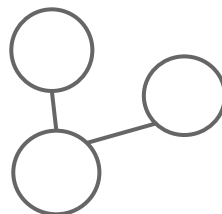
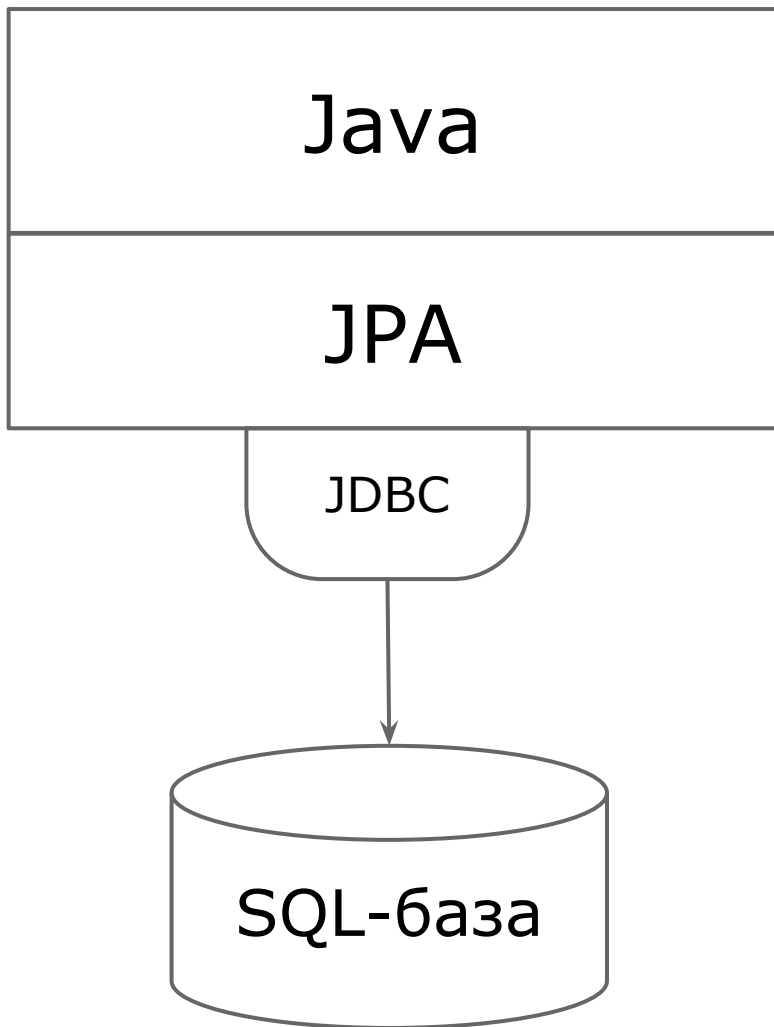
Простые программы



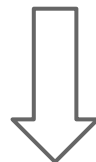
Сложности простых программ



Современные программы



Модель
данных



- Создание
- Проверка
- Обновление

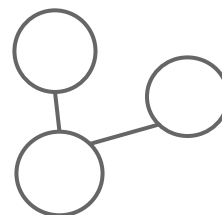
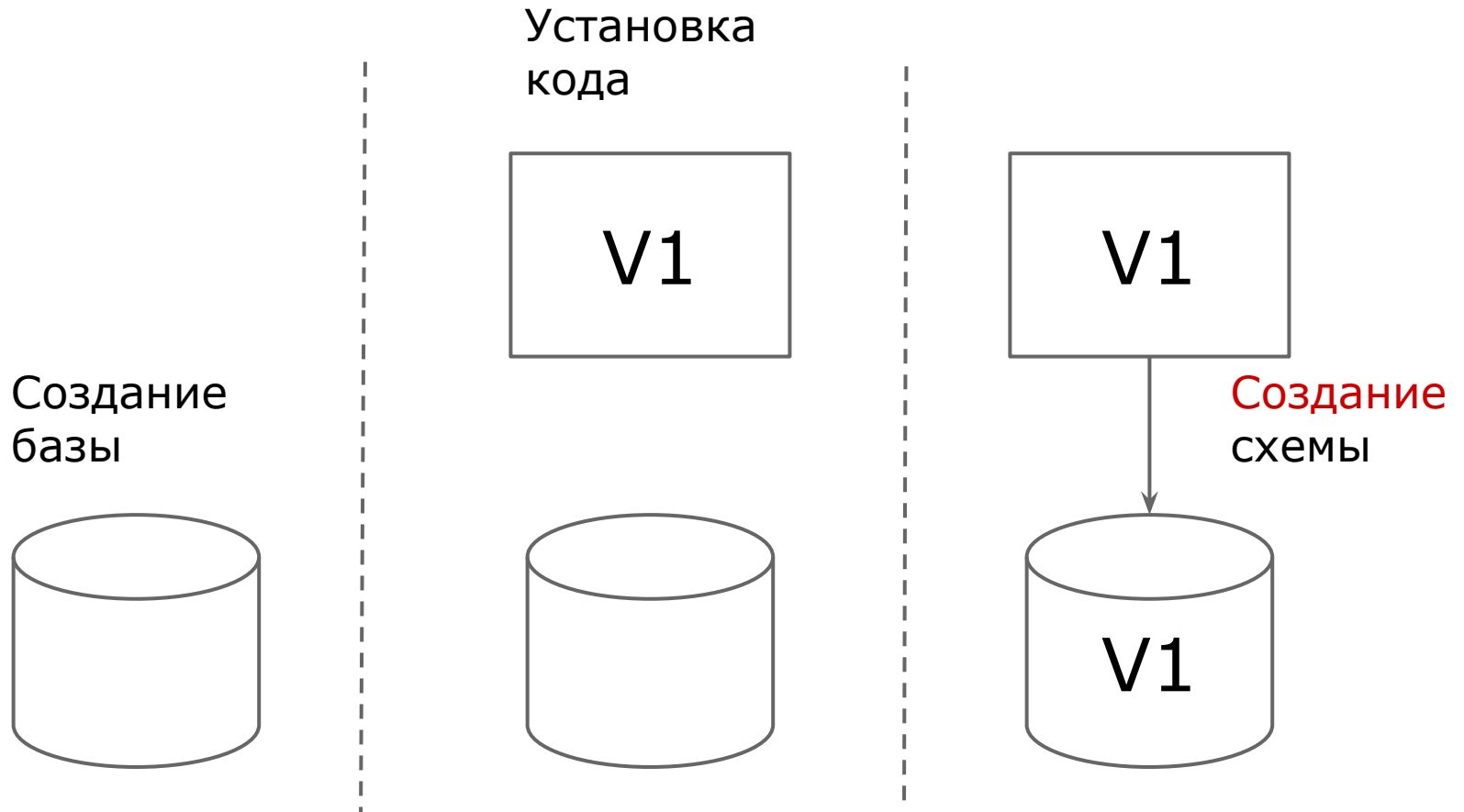


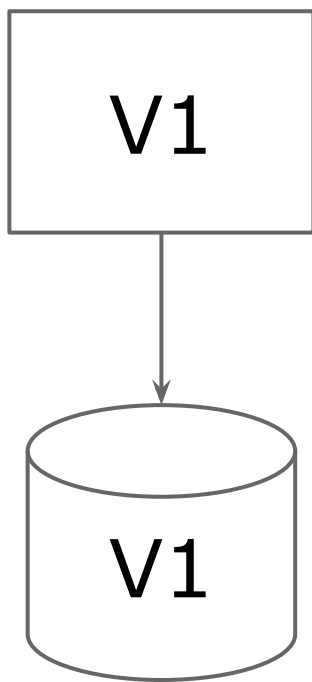
Схема
данных

Рождение продукта

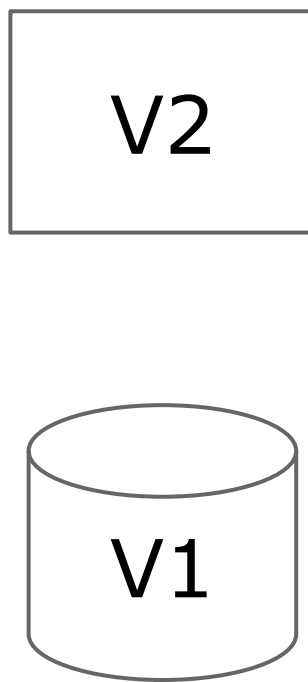


Тест обновления V1 до V2

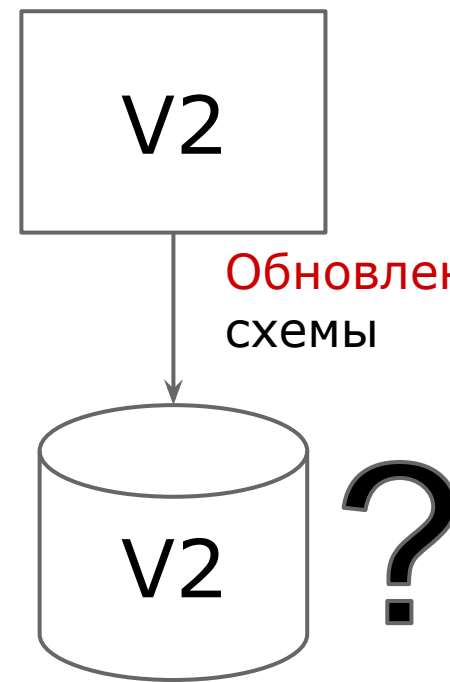
Удаление
старого кода



Установка
нового кода



Обновление
схемы

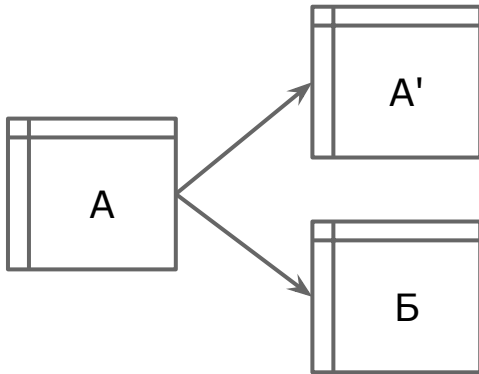


Проблемы JPA



Только для режима разработки!

- нет удалений полей
- слабая проверка типов
- ограничения игнорируются
- нет **миграции данных**



Пример простой миграции:

- Создание таблицы Б
- Копирование полей из А в Б
- Удаление части полей из А

Есть из чего выбрать?!

- 2010 год
- проект на grails 1.3.x

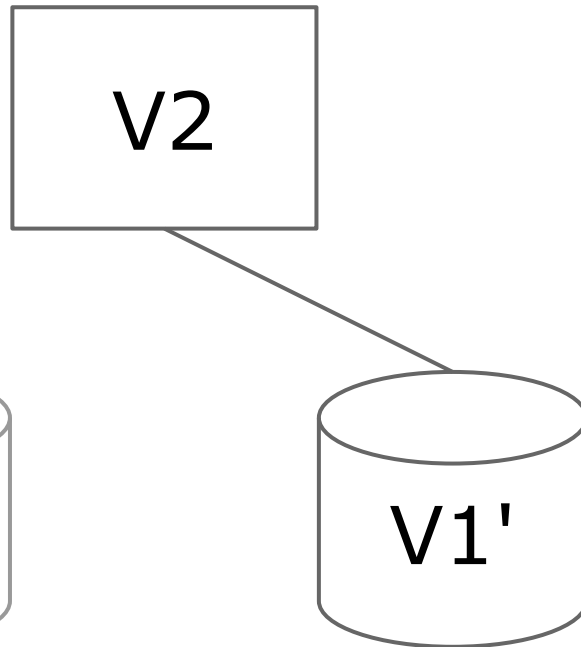
Ни одно решение не подошло...

Наши требования

- Сравнение схем двух баз
- Простое описание миграций
- Все миграций исполняются в одной транзакции

Идея. Шаг 1

1) Запуск
обновления кода



Цель

Определить
вероятную версию
схемы базы.

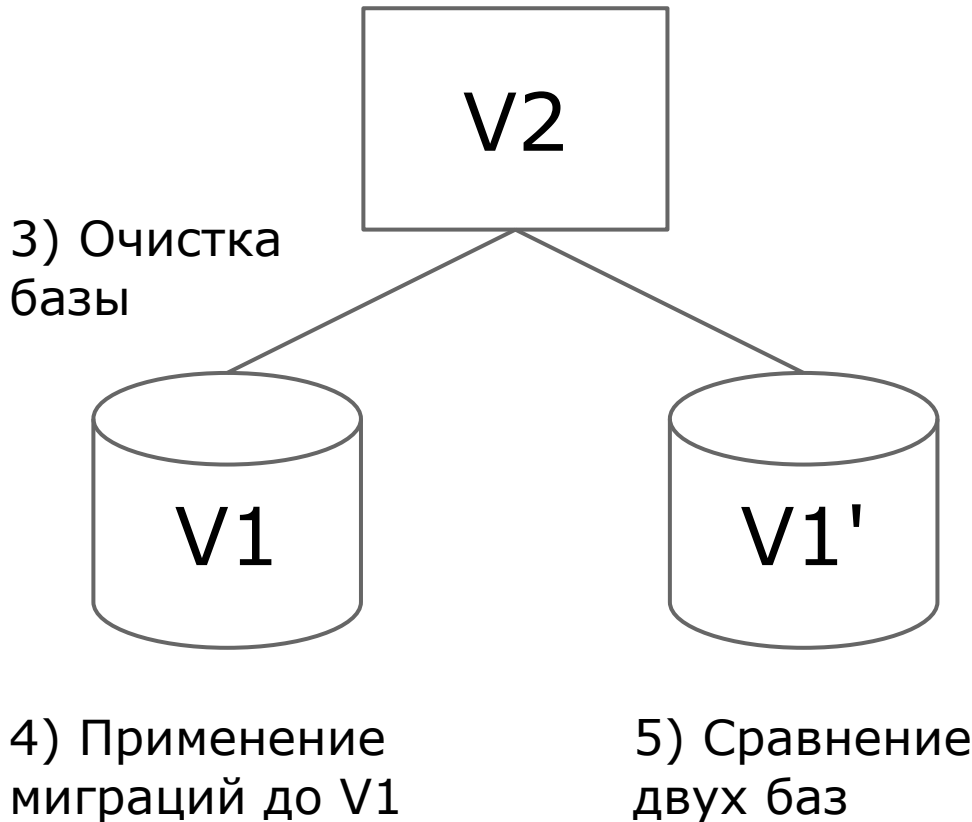
Может
использоваться
служебная
таблица.

Служебная база

2) Определение
версии

Идея. Шаг 2

Процесс
обновления кода

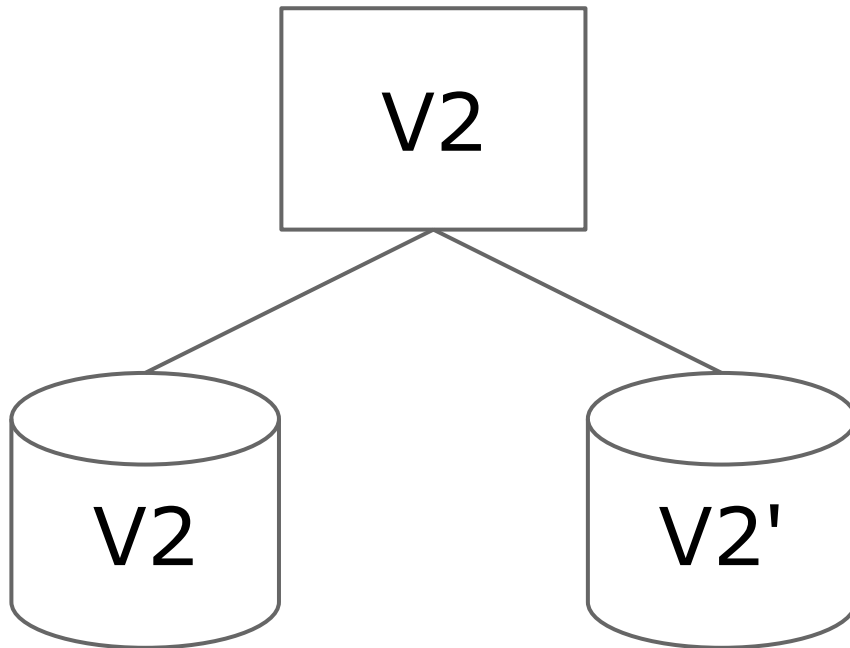


Цель

убедиться в том
что имеем схему
базы в версии V1

Идея. Шаг 3

8) Завершение
обновления кода



6) Применение
миграций до V2

7) Применение
миграций до V2

Цель

убедиться в том
что миграция до
V2 рабочая и
лишь потом её
выполнить

<http://LiquiBase.org>

- Проект с открытым кодом, лицензия Apache 2.0
- сравнение схем и вывод различий в виде команд миграции
- использует служебные таблицы с контрольными суммами
- описание миграций как на XML так и на SQL

Пример файла изменений

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<databaseChangeLog
  xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
  http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-2.0.xsd">

  <changeSet author="CodeInside" id="0.6">

    <addColumn tableName="orders">
      <column name="onlydatecreated" type="DATE"/>
    </addColumn>

    <sql>
      <comment>Задать дату задания из времени и даты </comment>
      update orders set onlydatecreated = DATE(datecreated)
    </sql>

    <addNotNullConstraint columnName="onlydatecreated" tableName="orders"
  />

</changeSet>

</databaseChangeLog>
```

Одна транзакция

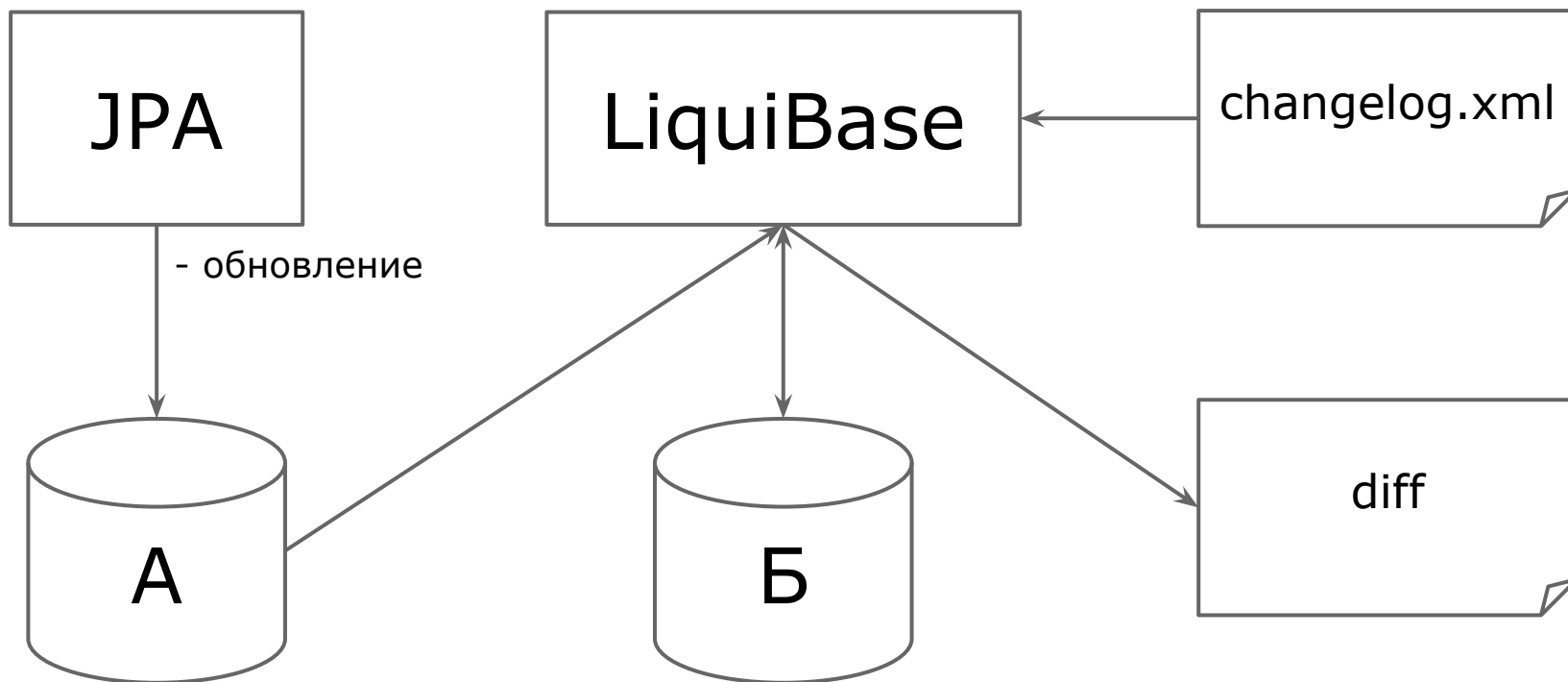
- используем MVCC СУБД PostgreSQL, поэтому DDL транзакционны

```
CREATE TABLE...
```

```
ALTER TABLE...
```

- написан фасад над всеми операциями (стандартно каждая транзакция на `changeSet`)

Процесс разработки



1) Создание
схемы V3

2) Применение
миграций до V3

3) Сравнение

Пример результата сравнения

```
@Entity
class PictureContent {
    ...
    @Column(length=16, nullable=false)
    String md5;
    ...
}

# play db:testmodel
...
<addColumn tableName="picturecontent">
  <column name="md5" type="VARCHAR(16)" defaultValue="">
    <constraints nullable="false"/>
  </column>
</addColumn>
...
```

Просто добавляем в changelog.xml !

Сложные миграции

- Изменения первичных ключей
- Изменение зависимостей или ограничений

Согласованность **после** миграции

- **Убираем** контроль целостности
 - ограничения внешних ключей
 - уникальные индексы
- Добавляем вспомогательные поля (*таблицы?*)
- Обновляем в несколько этапов
- Удаляем вспомогательные поля
- **Включаем** контроль целостности
 - ключи
 - индексы

Схема именования версий

{версияСхемыДанных}{_}{ревизияКода}

3.0

3.0_1

В коде это обычная константа:

```
final static String version = "3.0_1";
```

Тестирование моделей не достаточно

- покрывает лишь преобразование схемы
- нет главного - данных

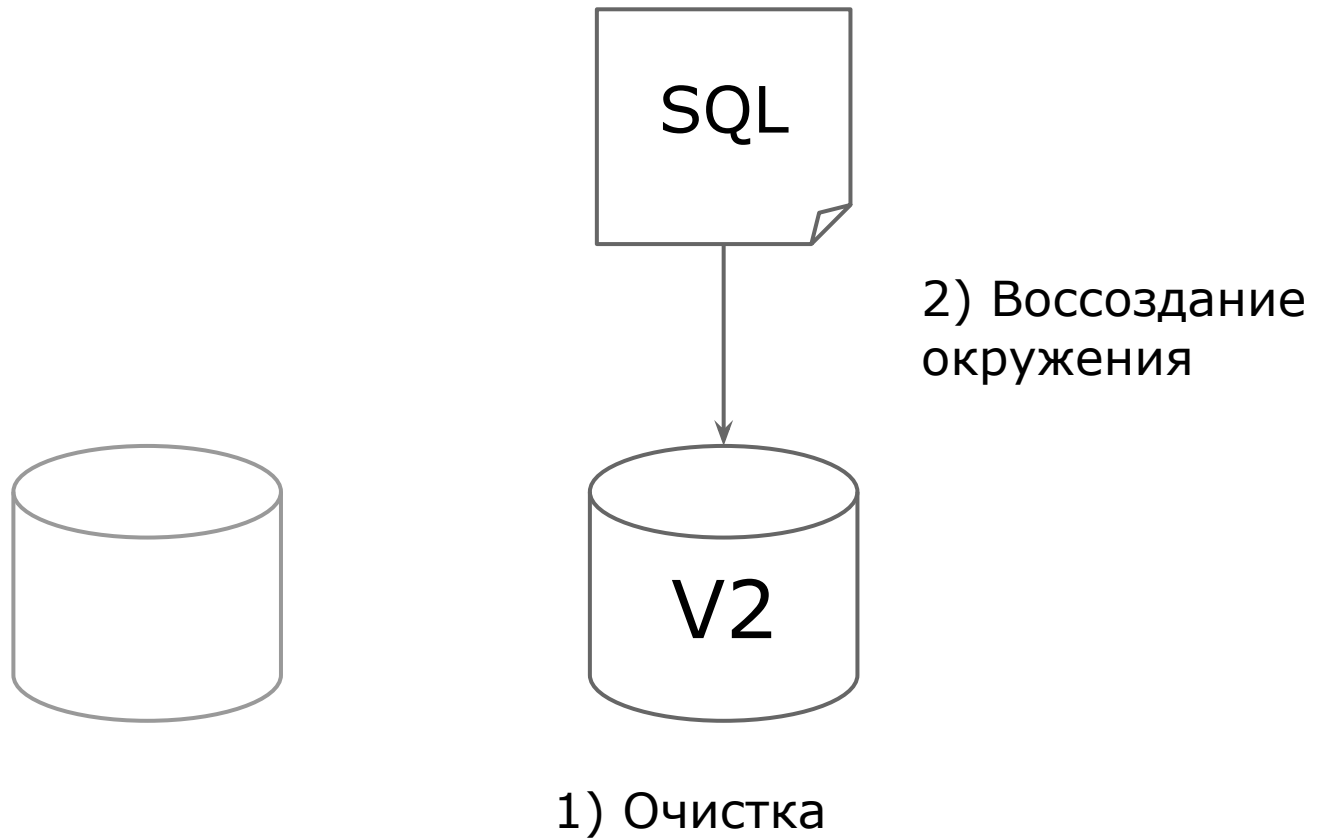
Нужно прогонять тесты на **реальных данных**

Деперсонализация базы:

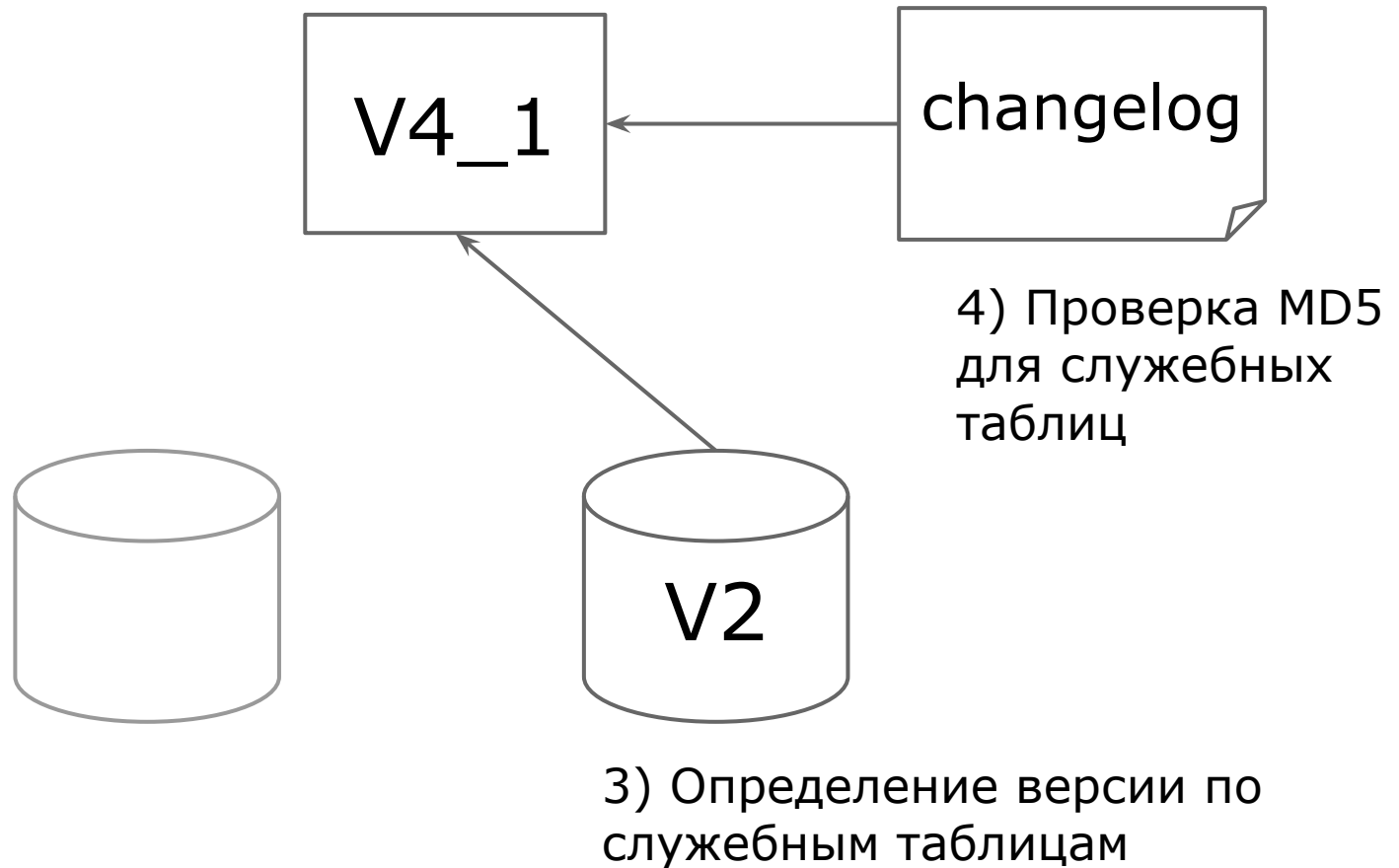
- убрать персональную информацию
- убрать лишние данные (контроль размера)

Удобный размер **<50 МБайт**

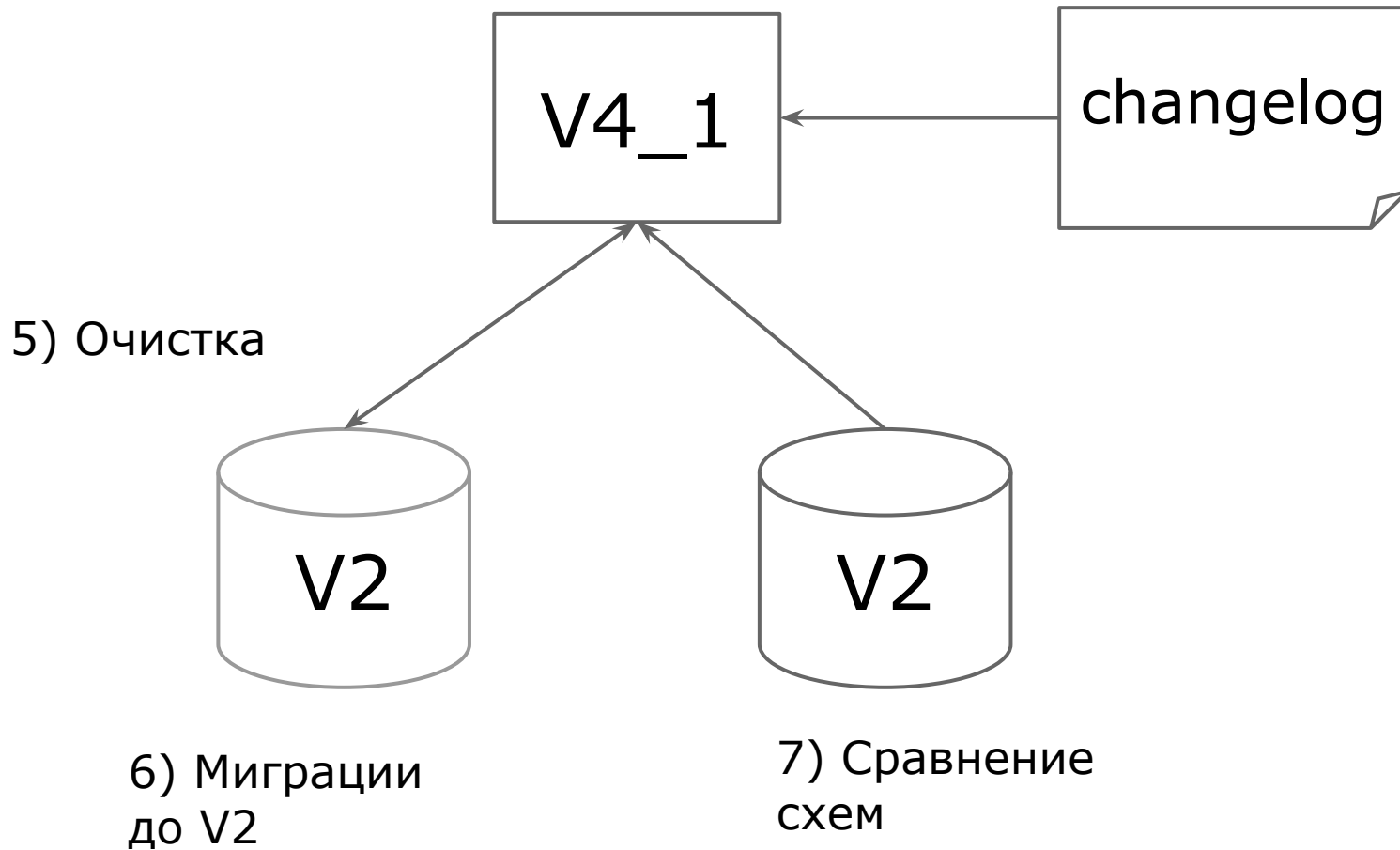
Тест миграций. Шаг 1



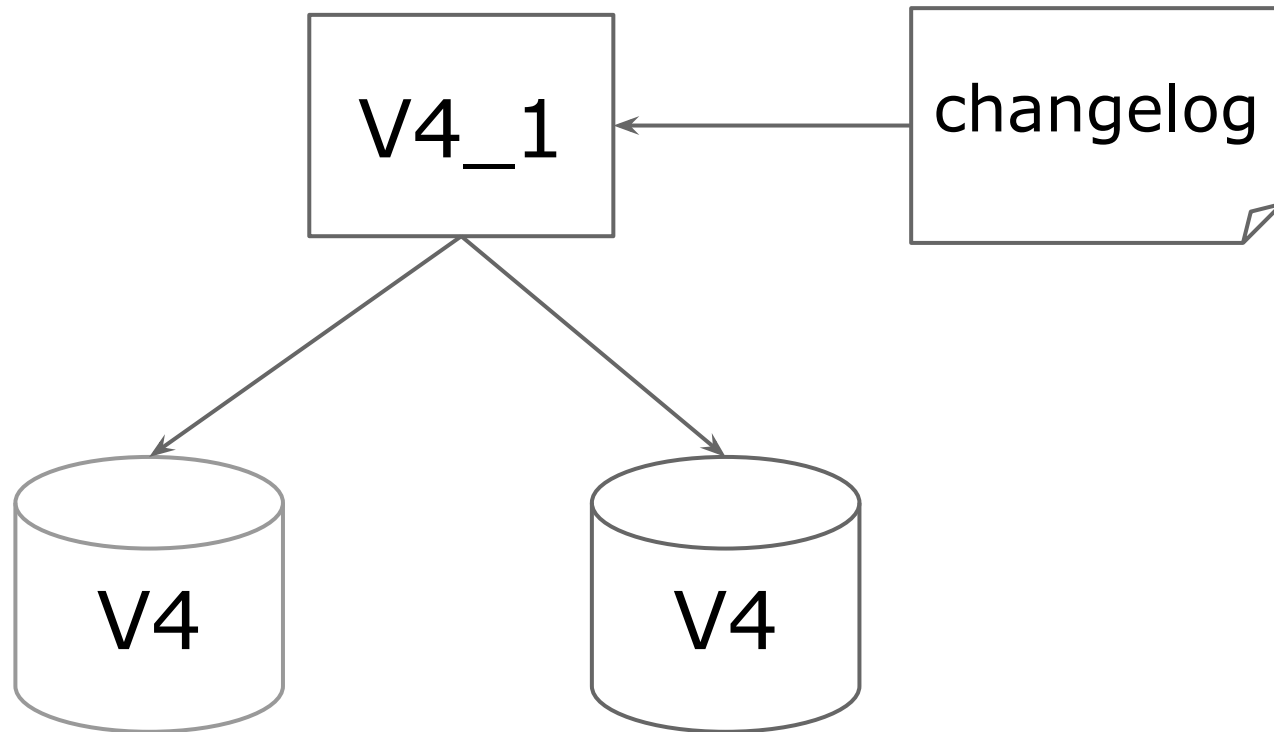
Тест миграций. Шаг 2



Тест миграций. Шаг 3



Тест миграций. Шаг 4



8) Миграции
с v2 до V4

9) Миграции
с v2 до v4

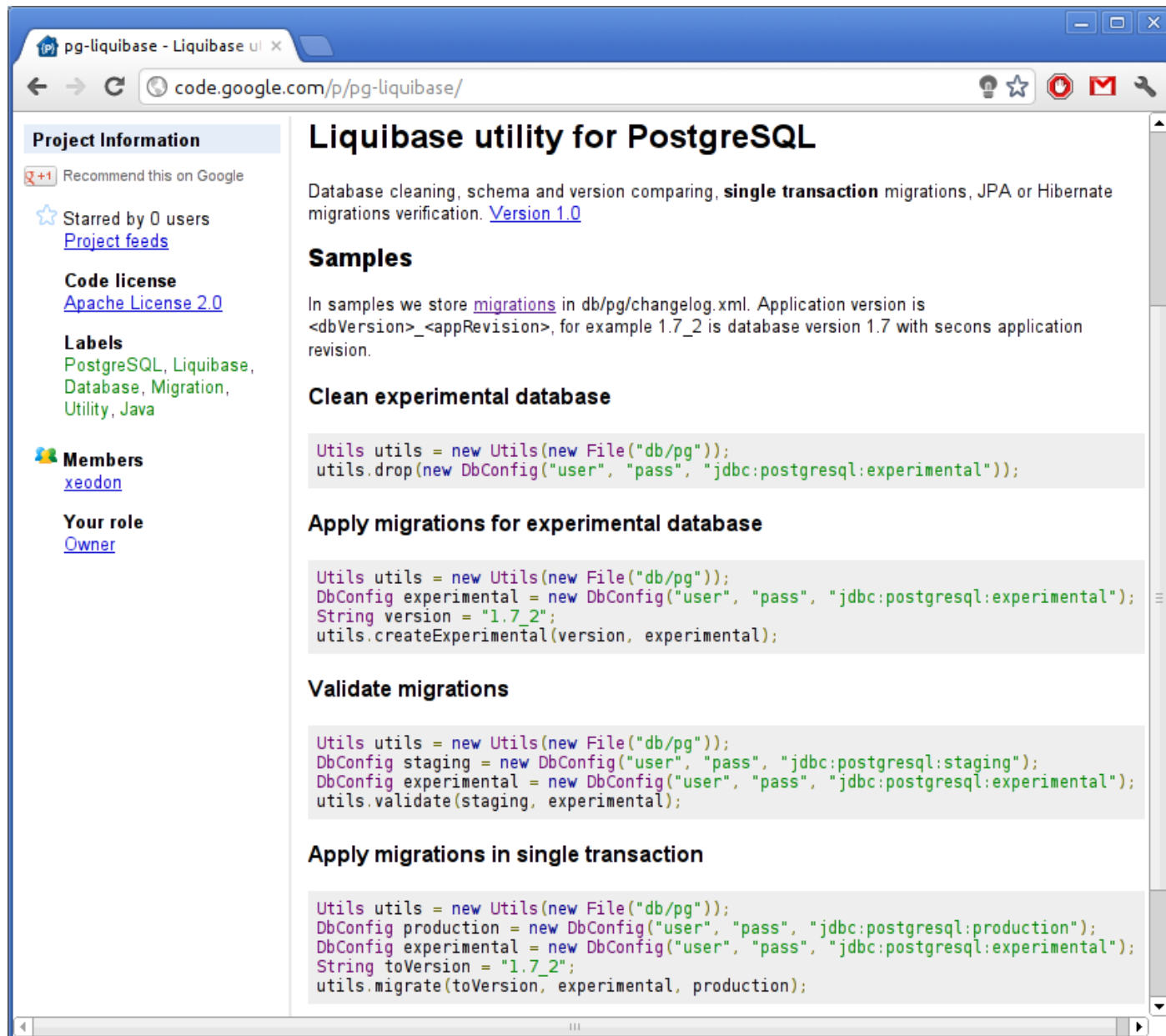
Реализация

- 2010 год - Модуль для grails
- 2011 год - Модуль для play framework
- changelog.xml под контролем версий (svn, git)
- тестирование модели и миграций на интеграционном сервере

Открытие кода фасада LiquiBase:

- 2012 год
- лицензия Apache 2.0

http://code.google.com/p/pg-liquibase



The screenshot shows a web browser window with the URL `code.google.com/p/pg-liquibase/`. The page title is "Liquibase utility for PostgreSQL". The left sidebar contains project information: "Recommend this on Google", "Stared by 0 users", "Project feeds", "Code license: Apache License 2.0", "Labels: PostgreSQL, Liquibase, Database, Migration, Utility, Java", "Members: xeodon", and "Your role: Owner". The main content area has the following sections:

Liquibase utility for PostgreSQL

Database cleaning, schema and version comparing, **single transaction** migrations, JPA or Hibernate migrations verification. [Version 1.0](#)

Samples

In samples we store [migrations](#) in `db/pg/changelog.xml`. Application version is `<dbVersion>_<appRevision>`, for example `1.7_2` is database version 1.7 with seconds application revision.

Clean experimental database

```
Utils utils = new Utils(new File("db/pg"));
utils.drop(new DbConfig("user", "pass", "jdbc:postgresql:experimental"));
```

Apply migrations for experimental database

```
Utils utils = new Utils(new File("db/pg"));
DbConfig experimental = new DbConfig("user", "pass", "jdbc:postgresql:experimental");
String version = "1.7_2";
utils.createExperimental(version, experimental);
```

Validate migrations

```
Utils utils = new Utils(new File("db/pg"));
DbConfig staging = new DbConfig("user", "pass", "jdbc:postgresql:staging");
DbConfig experimental = new DbConfig("user", "pass", "jdbc:postgresql:experimental");
utils.validate(staging, experimental);
```

Apply migrations in single transaction

```
Utils utils = new Utils(new File("db/pg"));
DbConfig production = new DbConfig("user", "pass", "jdbc:postgresql:production");
DbConfig experimental = new DbConfig("user", "pass", "jdbc:postgresql:experimental");
String toVersion = "1.7_2";
utils.migrate(toVersion, experimental, production);
```

В рекламе только плюсы

- Минусы прячут
- Всегда изучайте применение решения **ПОЛНОСТЬЮ**

Минус №1

С ростом количества миграций
скорость их применения падает

```
...
liquibase: ChangeSet changelog-0.6.xml::0.6::CodeInside ran successfully in 612ms
liquibase: ChangeSet changelog-0.7.xml::0.7::CodeInside ran successfully in 231ms
liquibase: ChangeSet changelog-0.8.xml::0.8::CodeInside ran successfully in 167ms
liquibase: ChangeSet changelog-0.9.xml::0.9::CodeInside ran successfully in 102ms
liquibase: ChangeSet changelog-0.9.1.xml::0.9.1::CodeInside ran successfully in 25ms
liquibase: ChangeSet changelog-1.0.xml::1.0::CodeInside ran successfully in 18ms
liquibase: ChangeSet changelog-1.1.xml::1.1::CodeInside ran successfully in 150ms
liquibase: ChangeSet changelog-1.2.xml::1.2::CodeInside ran successfully in 102ms
liquibase: ChangeSet changelog-1.2.1.xml::1.2.1::CodeInside ran successfully in 58ms
liquibase: ChangeSet changelog-1.2.2.xml::1.2.2::CodeInside ran successfully in 37ms
liquibase: ChangeSet changelog-1.3.xml::1.3::CodeInside ran successfully in 543ms
liquibase: ChangeSet changelog-1.3.1.xml::1.3.1::CodeInside ran successfully in 108ms
liquibase: ChangeSet changelog-1.3.2.xml::1.3.2::CodeInside ran successfully in 53ms
liquibase: ChangeSet changelog-1.3.5.xml::1.3.5::CodeInside ran successfully in 63ms
liquibase: ChangeSet changelog-1.3.6.xml::1.3.6::CodeInside ran successfully in 79ms
liquibase: ChangeSet changelog-1.3.7.xml::1.3.7::CodeInside ran successfully in 135ms
liquibase: ChangeSet changelog-1.4.0.xml::1.4.0::CodeInside ran successfully in 54ms
liquibase: ChangeSet changelog-1.4.2.xml::1.4.2::CodeInside ran successfully in 18ms
liquibase: ChangeSet changelog-1.5.0.xml::1.5.0::CodeInside ran successfully in 396ms
liquibase: ChangeSet changelog-1.5.1.xml::1.5.1::CodeInside ran successfully in 37ms
liquibase: ChangeSet changelog-1.5.2.xml::1.5.2::CodeInside ran successfully in 21ms
liquibase: ChangeSet changelog-1.5.3.xml::1.5.3::CodeInside ran successfully in 34ms
liquibase: ChangeSet changelog-1.5.4.xml::1.5.4::CodeInside ran successfully in 27ms
liquibase: ChangeSet changelog-1.5.5.xml::1.5.5::CodeInside ran successfully in 31ms
liquibase: ChangeSet changelog-1.5.6.xml::1.5.6::CodeInside ran successfully in 52ms
...
```

612ms

543ms

396ms

$\Sigma = 10s$

Минус №2

Вера в безупречность LiquiBase

- перемешивания порядка индексов в выводе различий
до версии 2.0.4
- неточности в выводе различий
в 1.9.x перепутан unique для индекса
- иногда игнорирует различия в названиях первичных ключей
до версии 2.0.4

Минус №3

Сильная зависимость от PostgreSQL

- всё завязано на реальной СУБД - медленно!
На сборочных серверах в postgresql.conf

...

```
fsync=off # жертвуем долговечностью!!!
```

...

- собственные средства загрузки SQL

```
if os.name == 'nt':  
    callShell(["psql", "-f", dumpName, dbName, dbUser])  
else:  
    callShell(["psql", dbName, dbUser, "-f", dumpName])
```

Вопросы!!!



HA HA!

D'OH!



Ответы???