

JUG Ostfalen, 08.02.2018

Continuous Database Integration mit Flyway

Sandra Parsick

mail@sandra-parsick.de
@SandraParsick

Zu meiner Person

- Sandra Parsick
- Freiberuflicher Softwareentwickler und Consultant im Java-Umfeld
- Schwerpunkte:
 - Java Enterprise Anwendungen
 - Agile Methoden
 - Software Craftmanship
 - Automatisierung von Entwicklungsprozessen
- Trainings
- Workshops
- Softwerkskammer Ruhrgebiet
- Twitter: @SandraParsick
- Blog:
<http://blog.sandra-parsick.de>
- E-Mail: mail@sandra-parsick.de



Agenda

- Continuous Database Integration (CDBI)
- Flyway
- Flyway Demo
- Fallstricke

Continuous Database Integration

- Definition
- Motivation
- Aufbau

Definition

„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“

(aus Continuous Integration by Paul M. Duvall, Steve Matyas und Andrew Glover)

Motivation

- Alle Entwickler teilen sich eine Testdatenbank.
- Keiner weiß, welche Datenbankskripte auf welchen Datenbankinstanzen ausgeführt worden.
- Testdatenbank unterscheidet sich von der Produktionsdatenbank.
- Datenbankmigrationsskripte verteilen sich auf Emails, Release Notes, Ticketsysteme, etc.

Aufbau

- Behandle den Datenbank-Code wie einen ganz normalen Source-Code
 - Alle Datenbank Artefakte (DDL, DML, Konfigurationen, Testdaten, Stored Procedures, Functions etc) gehören ins VCS.
 - Jede Änderung an den DB Artefakten wird getestet.
- Jeder Entwickler hat seine eigene Datenbank / Testdatenbanken ähneln den Produktionsdatenbanken.
 - Automatisiertes Aufsetzen der Datenbank.
- Änderungen an der Datenbank sind nachvollziehbar.
 - Historie der Änderungen

Flyway

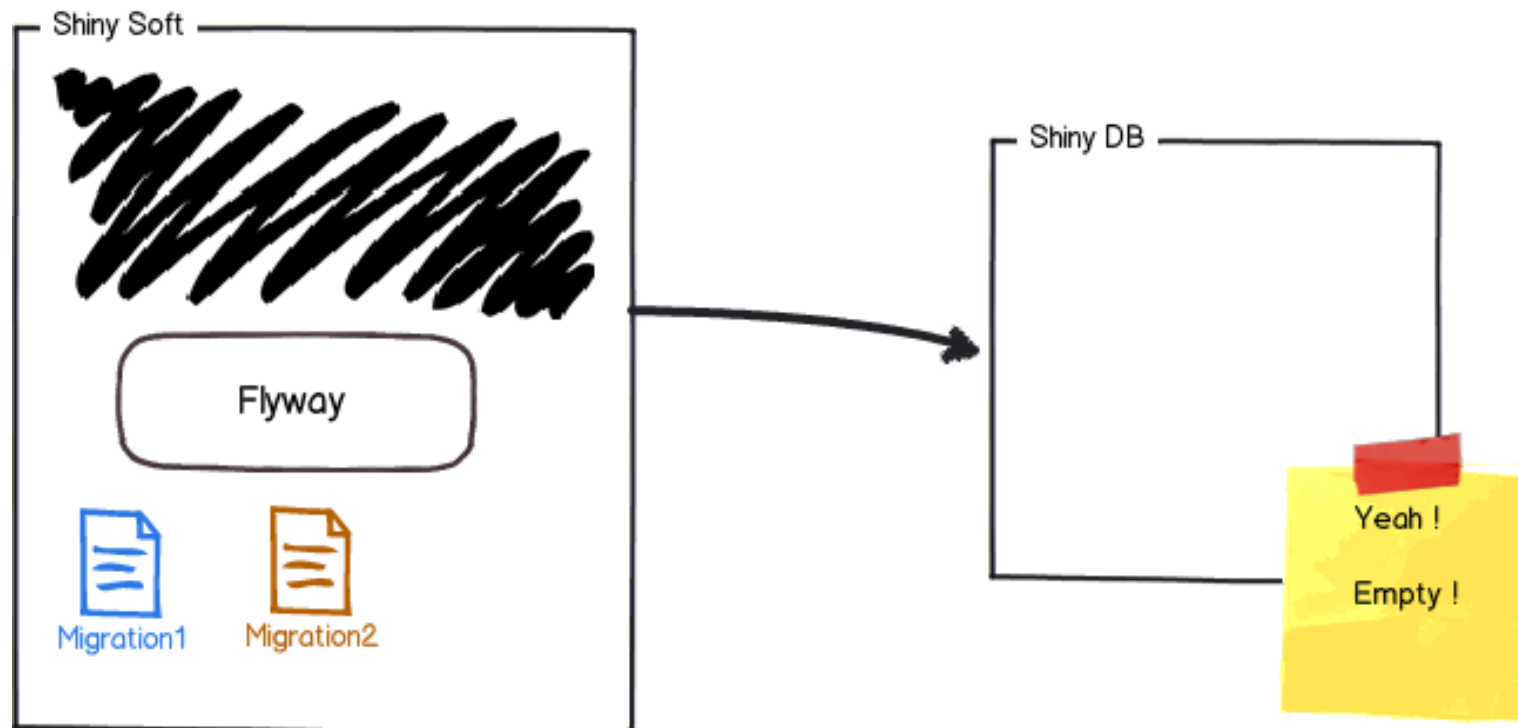
- Was ist Flyway?
- Wie funktioniert Flyway?
- Wie werden Migrationsskripte für Flyway geschrieben?
- Was kann Flyway nicht?
- Wie kann Flyway benutzt werden?

Was ist Flyway?



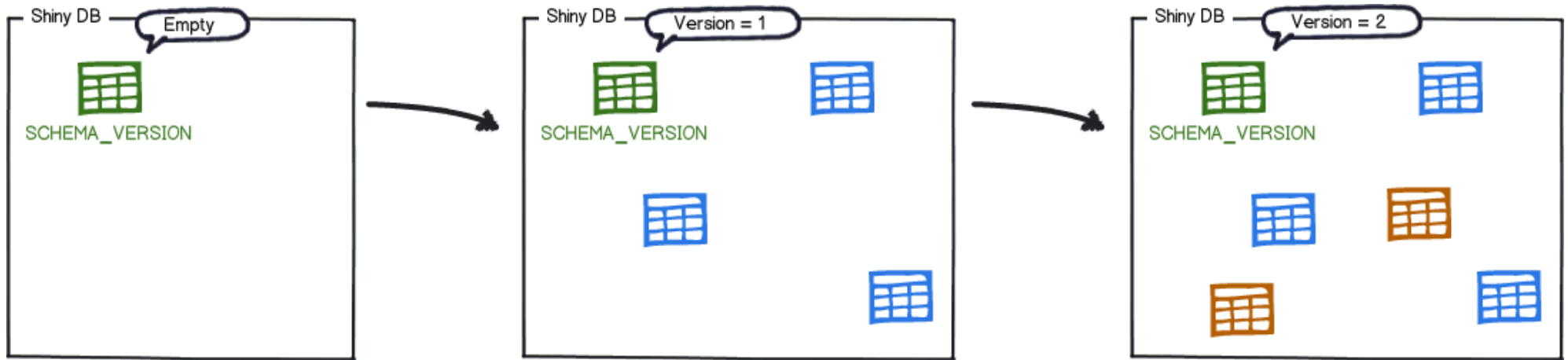
- Migration Framework für Relationale Datenbanken basierend auf Java
- Erstellt eine Datenbank „from scratch“
- Verwaltet den Stand der Datenbank
- Vier Migrationsmodi:
 - SQL-, Java-basierte Migration
 - Versionierte, wiederholbare Migration
- Aktuelle Version: 5.0.7
- Homepage: <http://flywaydb.org/>
- Twitter: @flywaydb

Wie funktioniert Flyway?



Wie funktioniert Flyway?

migrate



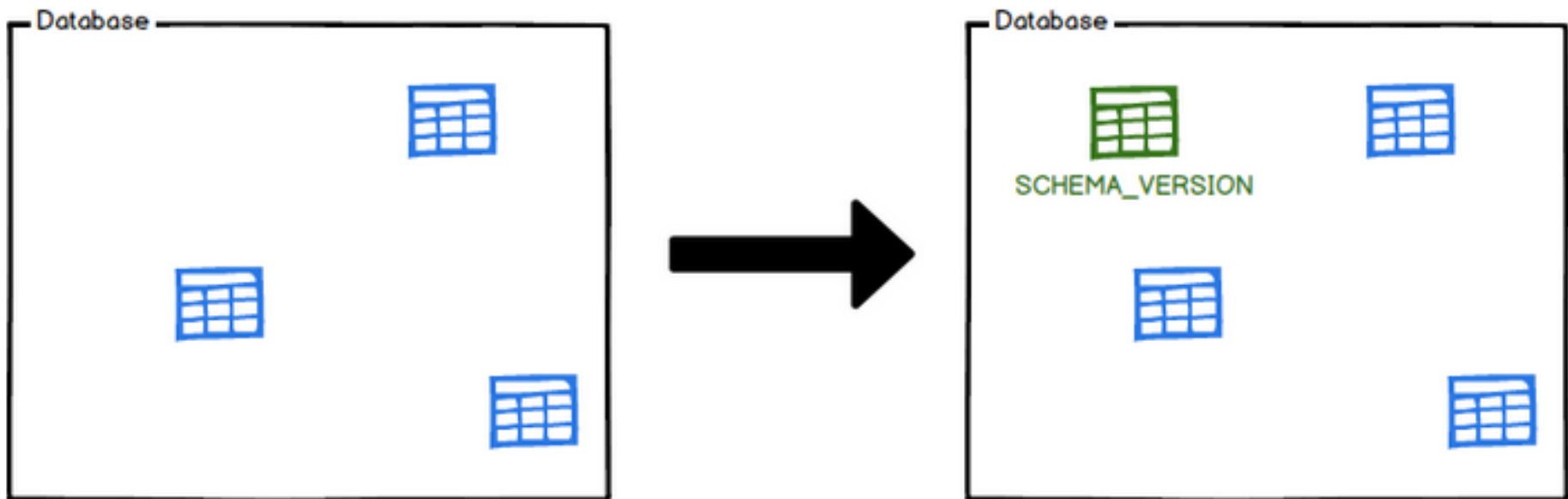
Reference: flywaydb.org

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

Wie funktioniert Flyway?

baseline



Migrationsskripte

- Vier Möglichkeiten

	Versioniert	Wiederholbar
SQL-basiert	✓	✓
Java-basiert	✓	✓

Versionierte Migration

- **Eigenschaften**
 - Skripte haben eine eindeutige Version
 - Werden genau einmal ausgeführt
- **Typische Anwendungsfälle**
 - DDL Änderungen (CREATE/ALTER/DROP für TABLES, INDEXES, FOREIGN KEYS,...)
 - Einfache Datenänderungen

Wiederholbare Migration

- **Eigenschaften**

- Skripte haben keine Versionsnummer
- Werden immer dann ausgeführt, wenn sich ihre Checksumme ändert
- Werden immer dann ausgeführt, nachdem alle versionierte Skripte ausgeführt wurden

- **Typische Anwendungsfälle**

- (Wieder-) Erstellung von views / procedures / functions / packages / ...
- Massenreimport von Stammdaten

SQL Migration

- **Typische Anwendungsfälle**

- DDL Änderungen (CREATE/ALTER/DROP für TABLES, VIEWS, TRIGGERS, SEQUENCES,...)
- Einfache Datenänderungen

- **Benennung der Skripte**

Prefix Description

V2_1_1_Some_description.sql

Version Seperator (two underscore)

The diagram shows the filename 'V2_1_1_Some_description.sql'. A green box highlights 'V2_1_1' with a green arrow pointing to the label 'Prefix'. A blue box highlights the first underscore after 'V2_1_1' with a blue arrow pointing to the label 'Version'. Another blue box highlights the second underscore before 'Some' with a blue arrow pointing to the label 'Seperator (two underscore)'. A brown box highlights 'Some_description' with a brown arrow pointing to the label 'Description'.

Prefix Description

R_Some_description.sql

Seperator (two underscore)

The diagram shows the filename 'R_Some_description.sql'. A green box highlights 'R' with a green arrow pointing to the label 'Prefix'. A blue box highlights the underscore after 'R' with a blue arrow pointing to the label 'Seperator (two underscore)'. A brown box highlights 'Some_description' with a brown arrow pointing to the label 'Description'.

SQL Migration

- **Syntax**

- Statement kann über mehrere Zeile gehen
- Platzhaltersupport
- Kommentare: Single (–) oder Multi-Line (/* */)
- Datenbank-spezifische SQL Syntax

- **Beispiel**

```
1      /* Create a table for person */
2
3      Create table person (
4          first_name varchar(128),
5          last_name varchar(128)
6      );
```

Unterstützte Datenbanken


 **Oracle**
(incl. Amazon RDS)

 **SQL Server**
(incl. Amazon RDS &
Azure SQL Database)

 **DB2**

 **MySQL**
(incl. Amazon RDS, Azure
Database & Google Cloud
SQL)

 **MariaDB**
(incl. Amazon RDS)

 **PostgreSQL**
(incl. Amazon RDS, Azure
Database, Google Cloud
SQL & Heroku)

 **Redshift**

 **CockroachDB**

 **SAP HANA**

 **Sybase ASE**

 **H2**

 **HSQLDB**

 **Derby**

 **SQLite**

Java Migration

- **Typische Anwendungsfälle**

- BLOB & CLOB Änderungen
- Fortgeschrittene Änderungen an Massendaten
(Neuberechnungen, fortgeschrittene Formatänderungen, ...)

- **Benennung der Java Klassen**

Prefix Description
V2_1_1 Some_description.java
Version Seperator (two underscore)

Prefix Description
R Some_description.java
Seperator (two underscore)

Java Migration

Beispiel

```
1 package db.migration;
2
3 import java.sql.Connection;
4 import java.sql.Statement;
5 import org.flywaydb.core.api.migration.jdbc.JdbcMigration;
6
7
8 public class V1_1_0__Insert_Data implements JdbcMigration {
9
10     @Override
11     public void migrate(Connection connection) throws Exception {
12         try (Statement statement = connection.createStatement()) {
13             statement.execute("Insert into person (first_name, last_name) Values ('Alice', 'Bob')");
14         }
15     }
16 }
17
18 }
19
```

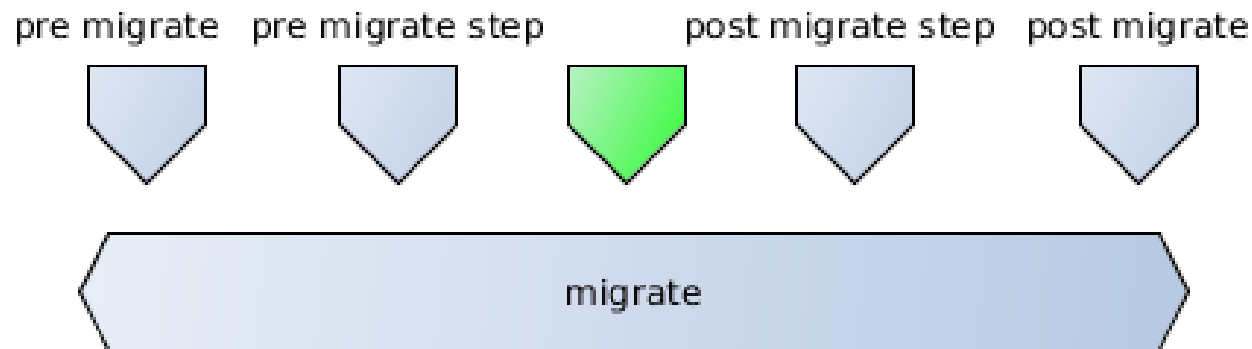
Java Migration

Beispiel Spring Support

```
1  package db.migration;
2
3  import org.flywaydb.core.api.migration.spring.SpringJdbcMigration;
4  import org.springframework.jdbc.core.JdbcTemplate;
5
6
7  public class V1_2_0__Create_Table_With_Spring_Support implements SpringJdbcMigration {
8
9      @Override
10     public void migrate(JdbcTemplate jdbcTemplate) throws Exception {
11         jdbcTemplate.execute("Create table address (street Varchar(128), place Varchar(128))");
12     }
13
14 }
15
```

Migration für Fortgeschrittene - Callbacks

- **Typische Anwendungsfälle**
 - Stored Procedure Kompilierung
 - Materialized View Update
- **Flyway Lifecycle (Beispiel migrate)**



SQL Callbacks

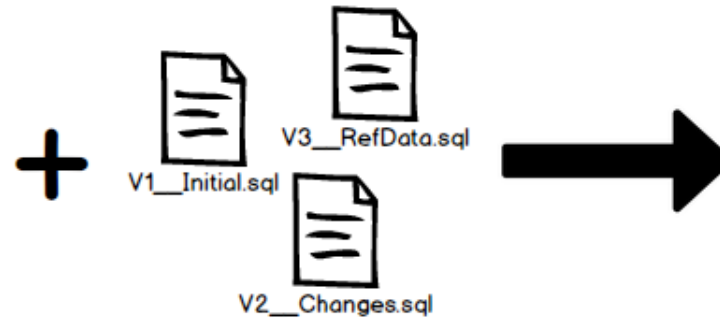
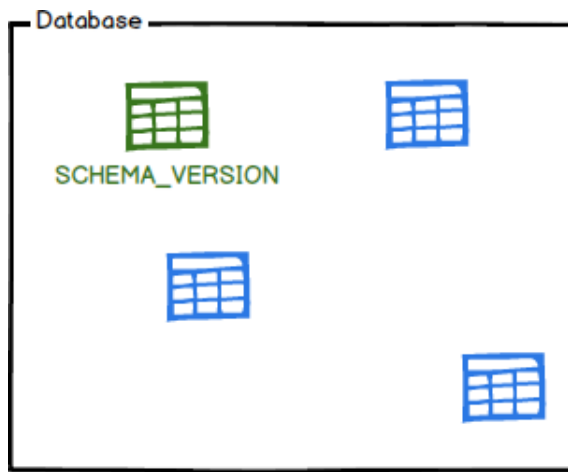
- **Beispiel migrate-Lifecycle:**
 - SQL Callback Skripte werden anhand deren Namen erkannt:
 - BeforeMigrate.sql
 - BeforeEachMigrate.sql
 - AfterEachMigrate.sql
 - AfterMigrate.sql

Java Callbacks

```
public interface FlywayCallback {  
    /**  
     * Runs before the clean task executes.  
     *  
     * @param connection A valid connection to the database.  
     */  
    void beforeClean(Connection connection);  
  
    /**  
     * Runs after the clean task executes.  
     *  
     * @param connection A valid connection to the database.  
     */  
    void afterClean(Connection connection);  
  
    /**  
     * Runs before the migrate task executes.  
     *  
     * @param connection A valid connection to the database.  
     */  
    void beforeMigrate(Connection connection);  
}
```


Weitere Flyway Befehle

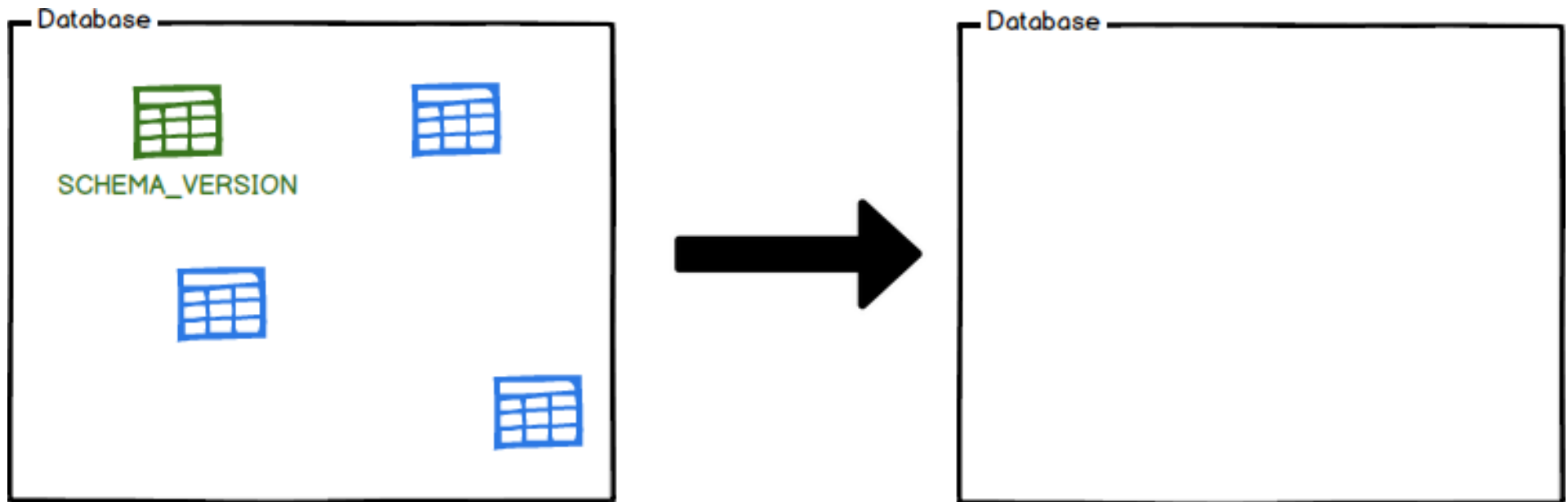
info



Version	Description	Installed on	State
1	Initial	2014-11-16 10:26:35	SUCCESS
2	Changes	2014-11-16 10:26:37	SUCCESS
3	RefData	2014-11-16 10:26:41	PENDING

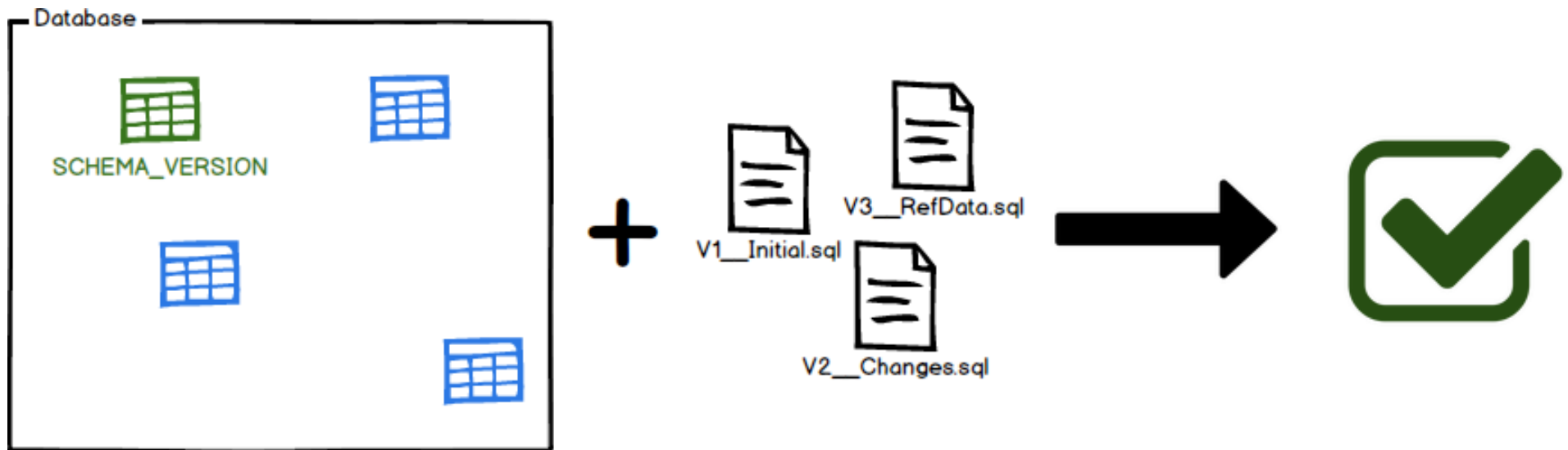
Weitere Flyway Befehle

clean



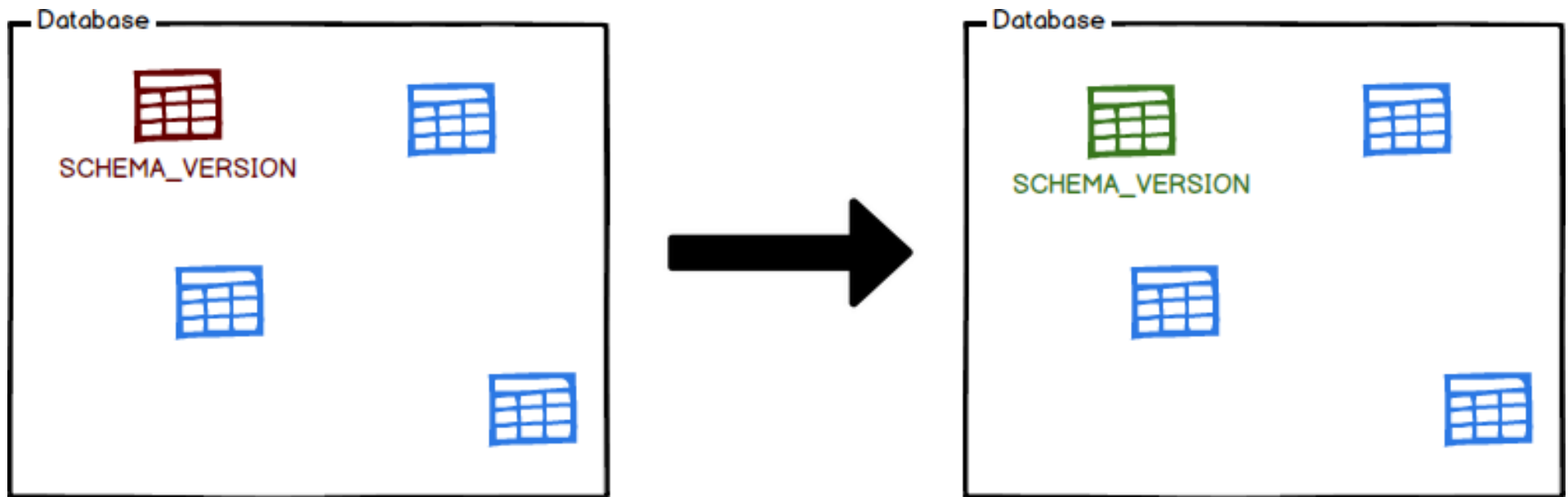
Weitere Flyway Befehle

validate



Weitere Flyway Befehle

repair



Was kann Flyway nicht?

- Rollback Skripte aufrufen (Community Edition)
- „Write once, run on many database vendors“

Neu seit Version 5

- Unterscheidung zwischen
 - Community Edition
 - Pro Edition
 - Enterprise Edition

	Community Edition	Pro Edition	Enterprise Edition
SQL-based migrations	✓	✓	✓
Java-based migrations	✓	✓	✓
Repeatable migrations	✓	✓	✓
Placeholder replacement	✓	✓	✓
Callbacks	✓	✓	✓
Custom migration resolvers/executors	✓	✓	✓
Safe for multiple nodes in parallel	✓	✓	✓
Native SQL dialect support (PL/SQL, SQLPL, T-SQL, ...)	✓	✓	✓
Latest database versions compatibility	✓	✓	✓
Java 8 / 9 compatibility	✓	✓	✓
Oracle SQL*Plus compatibility		✓	✓
Custom error handlers		✓	✓
Dry runs		✓	✓
Undo		✓	✓
Display query results		✓	✓
Older database versions compatibility			✓
Java 6 / 7 compatibility			✓
License	Apache v2	Commercial	Commercial

Wie kann Flyway benutzt werden?

- **Flyway Clients:**

- Java API
- Maven Plugin
- Command-line Tool
- Gradle Plugin
- SBT Plugin
- Ant task

- **Third Party Plugins:**

- Spring Boot
- Grails
- Dropwizard
- Play
- Und weitere

Demo







TESTCONTAINERS

Testcontainers

- Temporary database containers - spezielle MySQL, PostgreSQL, Oracle XE und Virtuoso container
- Webdriver containers - Dockerized Chrome oder Firefox browser für Selenium/Webdriver Operationen mit automatischer Videoaufnahme
- Generic containers – irgendein Docker Container
- Docker compose – Wiederverwendung von Docker Compose YAML Datei
- Dockerfile containers – Container direkt von einem Dockerfile

Aufbau CDBI

- Behandle den Datenbank-Code wie einen ganz normalen Source-Code
 - Alle Datenbank Artefakte (DDL, DML, Konfigurationen, Testdaten, Stored Procedures, Functions etc) gehören ins VCS. 
 - Jede Änderung an den DB Artefakten wird getestet. 
- Jeder Entwickler hat seine eigene Datenbank / Testdatenbanken ähneln den Produktionsdatenbanken.
 - Automatisiertes Aufsetzen der Datenbank. 
- Änderungen an der Datenbank sind nachvollziehbar.
 - Historie der Änderungen 

Fallstricke

Keine Instanz-spezifischen Daten

Beispiel

1
2
3
4

```
GRANT SELECT, INSERT ON usermgm.* TO  
`technical-user`@'192.168.33.10' IDENTIFIED BY 'pA$$w0rt';
```

Keine Instanz-spezifischen Daten

Möglicher Lösungsansatz:

```
1  
2 GRANT SELECT, INSERT ON usermgm.* TO  
3 `technical-user`@'*' IDENTIFIED BY 'pA$$w0rt';  
4
```

- Zugriffskontrolle über eine Firewalls (iptables)

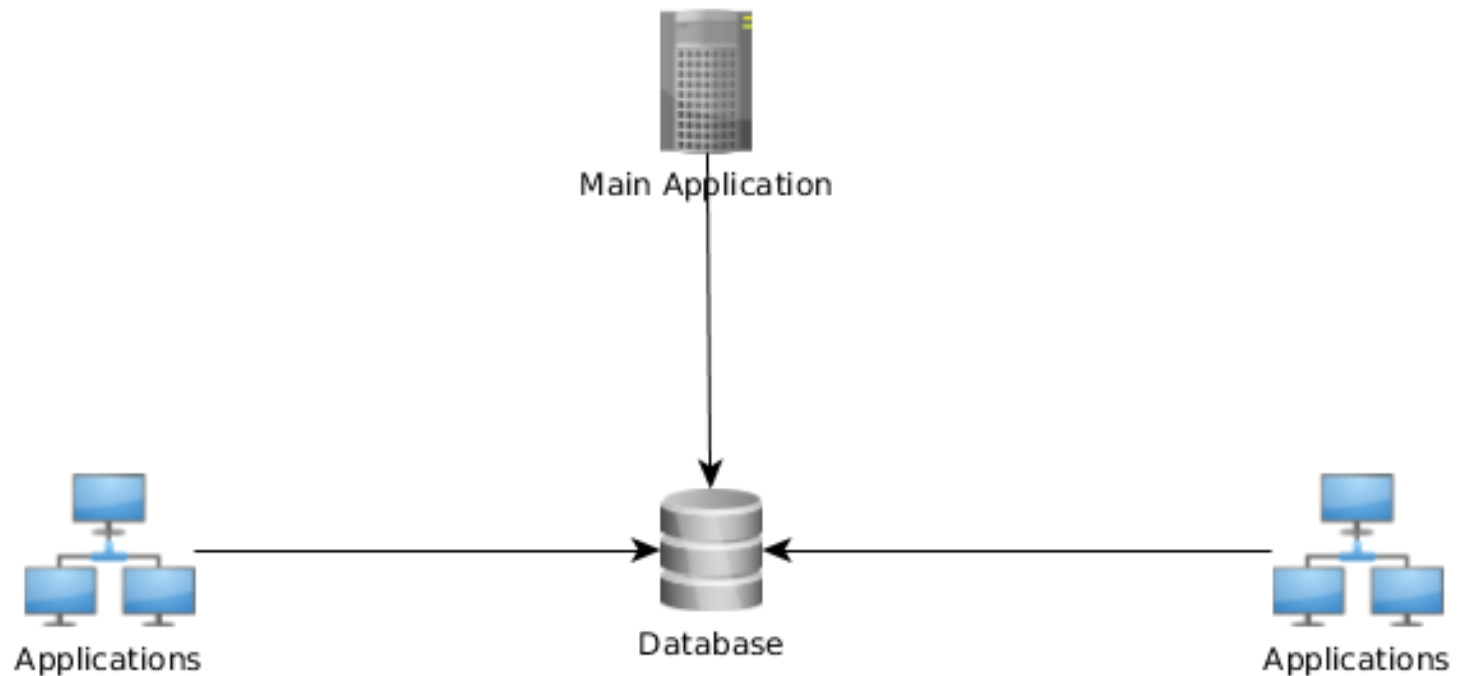
Keine Instanz-spezifischen Daten

Möglicher Lösungsansatz:

```
1 GRANT SELECT, INSERT ON usermgnt.* TO  
2 'technical-user' @ '${address}' By '${password}';  
3  
4
```

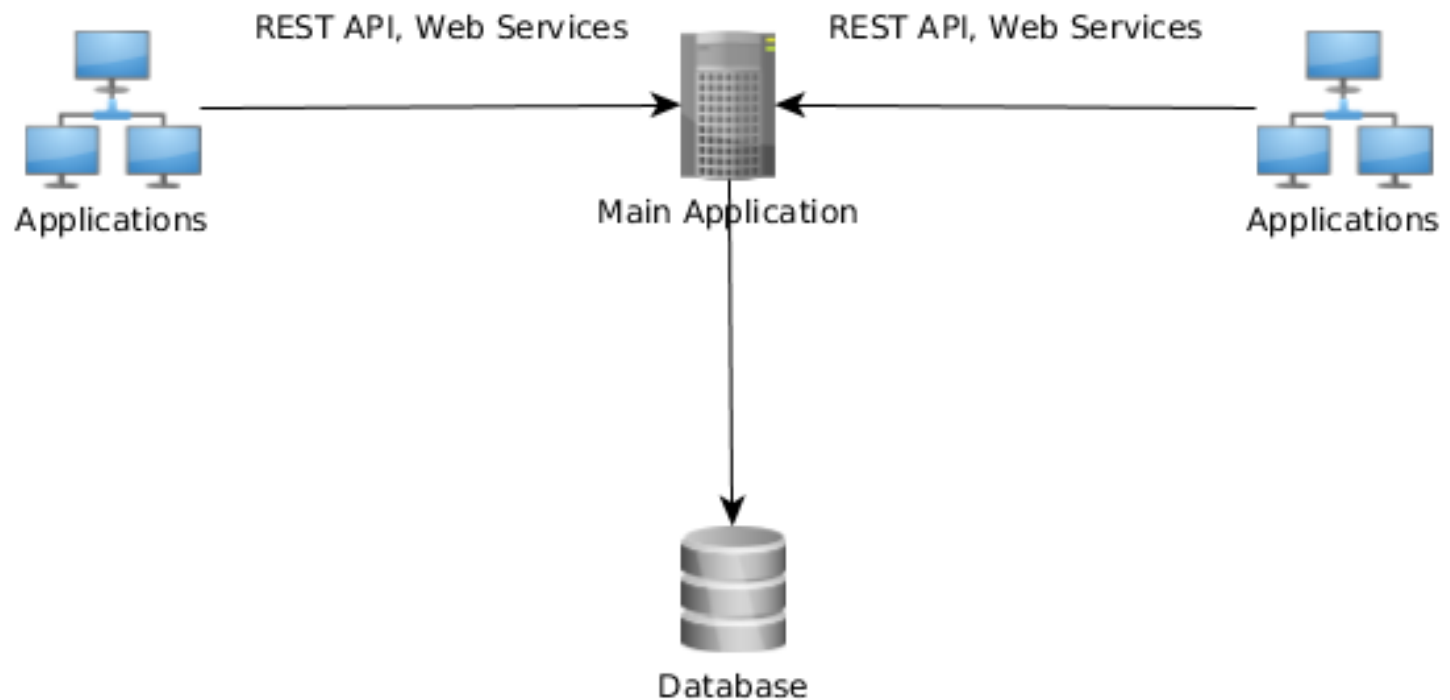

Datenbank wird von mehreren Applikationen benutzt

Ausgangslage :



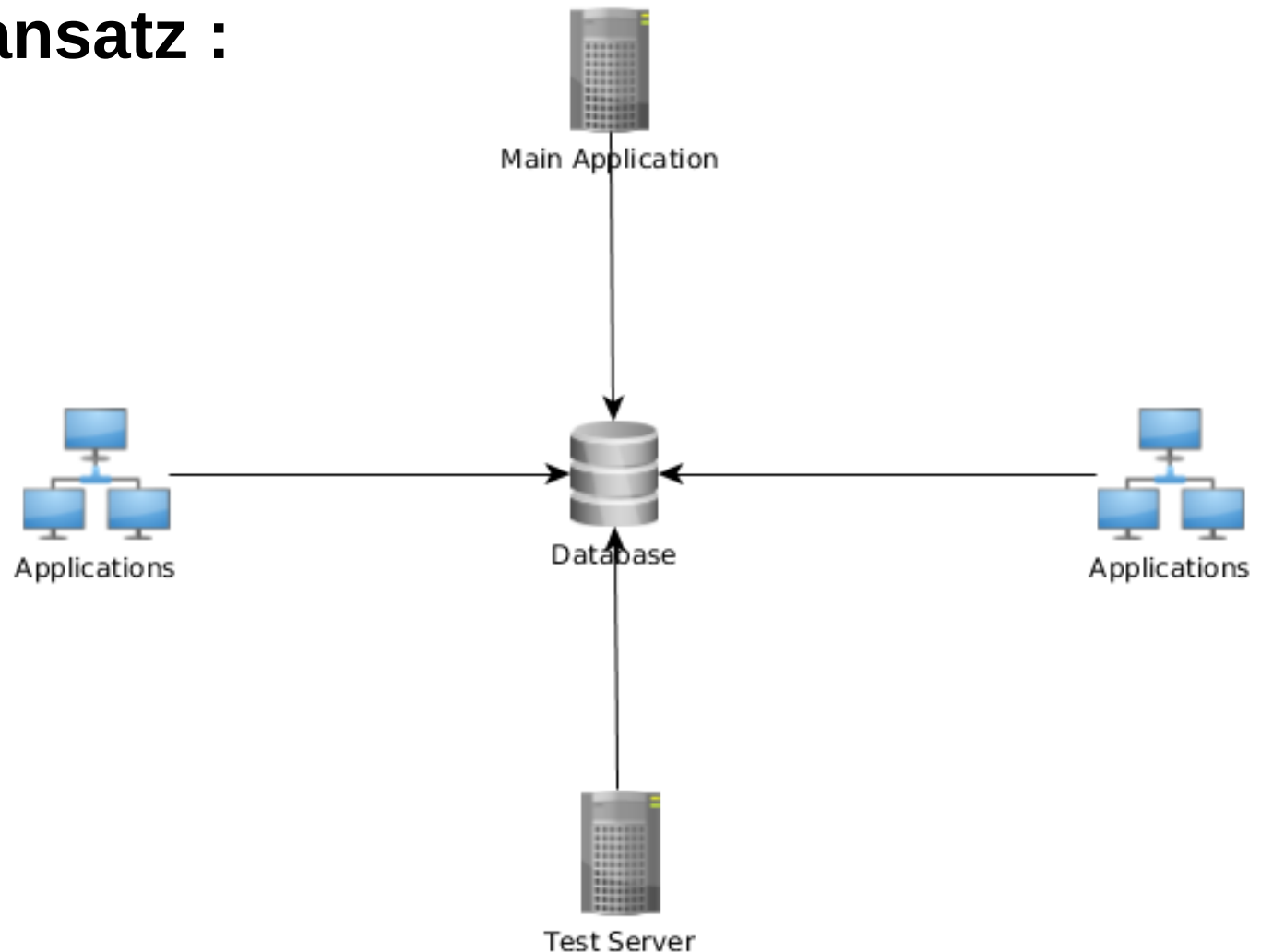
Datenbank wird von mehreren Applikationen benutzt

Lösungsansatz :

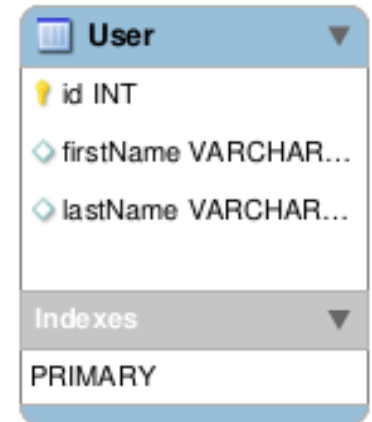
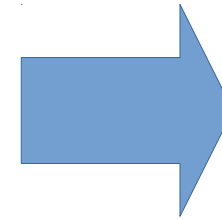
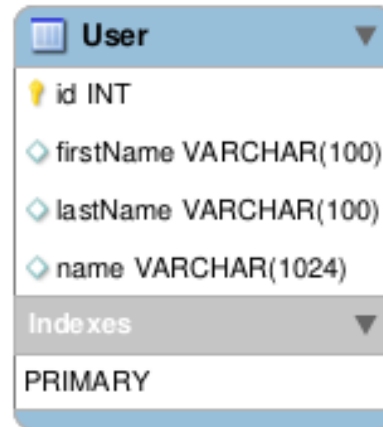
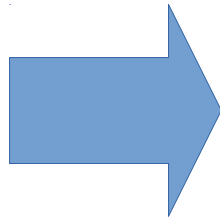
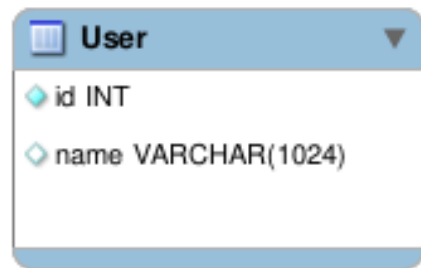


Datenbank wird von mehreren Applikationen benutzt

Lösungsansatz :



Datenbank wird von mehreren Applikationen benutzt



Weitere Fallstricke (Auszug)

- Datenänderung dauern zu lange
- Datenlöschung
- Faktor Mensch
- ...

Weitere Informationen

- Continuous Integration von Paul M. Duvall, Steve Matyas und Andrew Glover
- Refactoring Databases: Evolutionary Database Design von Scott J. Ambler und Pramodkumar J. Sadalage
- Flyway Documentation
<http://flywaydb.org/documentation/migration/>
<http://flywaydb.org/getstarted/>
- Source code:
<https://github.com/sparsick/flyway-talk>

Fragen?

<https://github.com/sparsick/flyway-talk>
mail@sandra-parsick.de
@SandraParsick