itspeople

DOWNLOAD VERSION

Die Reise von Flyway zu Liquibase

Sabine Heimsath
its-people GmbH

we make the difference

its-people Unternehmensdarstellung

its-people

- IT-Consulting
- Hochqualifizierte, selbstständige IT-Experten
- Mehr als 15 Jahre Erfolg am Markt
- Gemeinsam stark!

Portfolio

- IT-Security
- Datenbanken & Cloud
- Anwendungsentwicklung
- Analytics & Data Warehouse
- Customer Data Integration
- Projekt-Management Services

Stärken

- IT- und Fachkompetenz
- Kundenfokussierung
- Qualitätsorientierung
- Umsetzungsstark
- Partnerschaftlich
- Menschen im Mittelpunkt

Sabine Heimsath


Photo by Ablimit Ablet on Unsplash

Client

Architektur und PL/SQL- und APEX-Entwicklung

its-people

Portfolio Manager Datenbank und Anwendungsentwicklung

Bloggerin

DOAG

Development Community (PL/SQL)

its-people.de

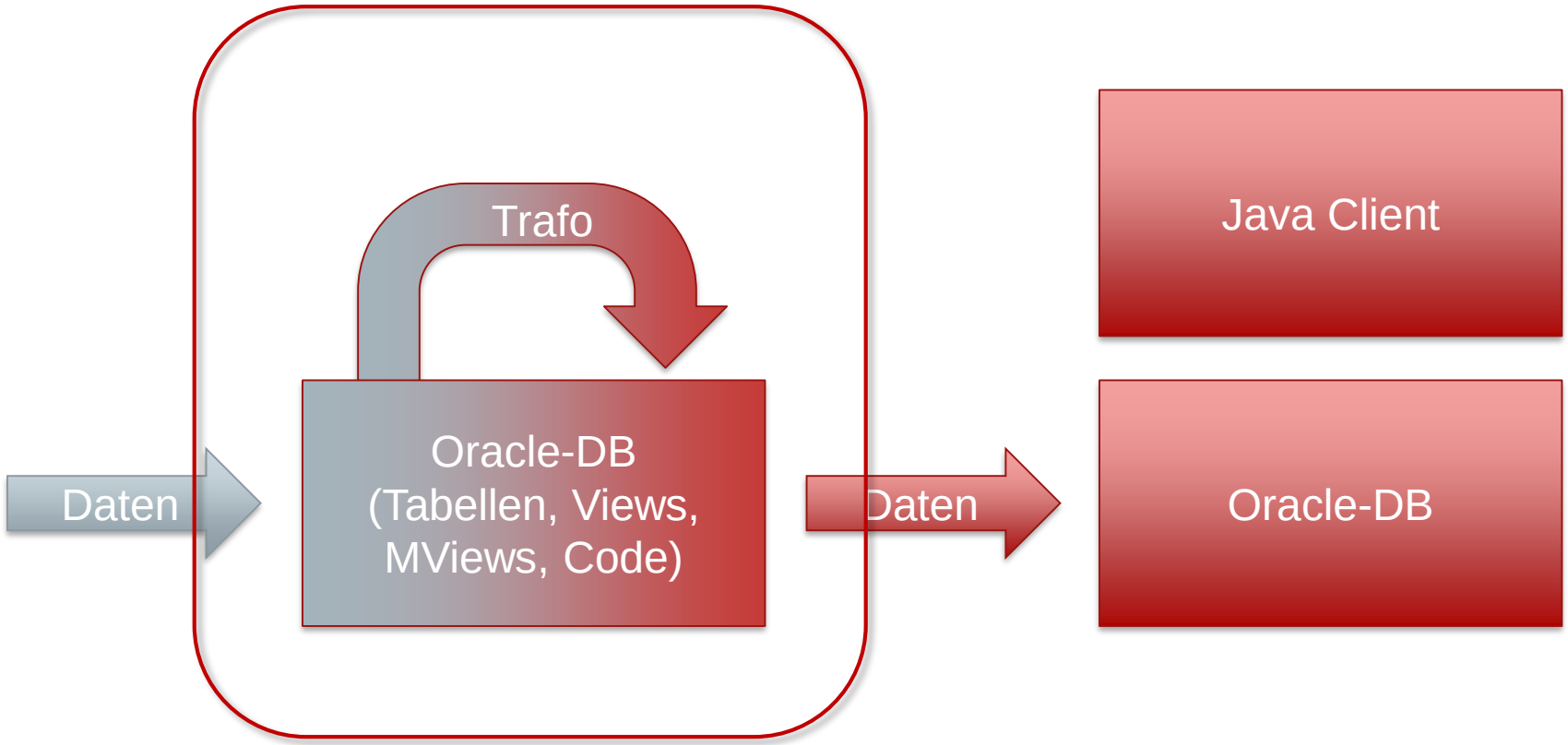
sabine.heimsath@its-people.de

@oraesque

Und Sie?

- ☐ Flyway
- ☐ Liquibase
- ☐ Keins davon

Der Ausgangspunkt



Die Historie

- Installation über handgeschriebene Delta-Skripte
  (Steuer-Skript ruft untergeordnete Skripte auf)
- Teil 1: Delta-Skripte für Änderungen an Daten und Datenmodell
  Teil 2: Wiederholfähige Skripte (Create or Replace für Views und Packages)
- Eigenes Wrapper-Package für DDL-Statements
  (für Exception Handling und Wiederholfähigkeit)
- Sourcecode-Verwaltung in Subversion

Die Historie

- Umstellung der Versionsverwaltung von Subversion auf Git

… führte zu einem Problem.

Exkurs …

Welcher Softwarestand zur Laufzeit?

Problem: (Rückwirkende) Analyse von Ergebnissen auf Entwicklermaschinen

Zentrale Frage: **Wie war der Softwarestand der einzelnen Packages zur Laufzeit?**

Zu Subversion-Zeiten einfache Lösung …

Subversion – Keyword Substitution

Keywords im Source Code werden beim Einchecken ersetzt. Zum Beispiel so:

```
/*
$Rev: 12 $: Revision of last commit
$Author: larry $: Author of last commit
$Date: 2006-03-15 02:33:03 $: Date of last commit
*/
```

oder so:

```
g_body_last_modified_date varchar2( 60 char )
          := '$Date: 2006-03-15 02:33:03 $';
g_body_version   varchar2( 30 char )
          := ‘$Rev: 12 $';
```

View Source

```
with svn_data
     as (select name
              , text
              , regexp_replace(
                  text, '(.*\$ (Rev|Date|Author):)|(\$.*)' ) info
              , case
                  when instr( text, '$Rev' ) > 0 then 'Rev'
                  when instr( text, '$Date' ) > 0 then 'Date'
                  else 'Auth'
                end typ
           from user_source
          where line < 10
            and text like '%$Rev:%'
             or text like '%$Date:%'
             or text like '%$Author:%')
```

```
select r.name
     , r.info svn_revision
     , d.info check_in_date
     , a.info author
  from svn_data r
       join svn_data d
         on d.typ = 'Date'
        and r.name = d.name
       join svn_data a
         on a.typ = 'Auth'
        and r.name = a.name
 where r.typ = 'Rev';
```

Subversion: Keyword Substitution

Ergebnis in der View:

	NAME	SVN_REVISION	CHECK_IN_DATE	AUTHOR
1	PKG DREI	22	2018-07-18 17:48:30 -0100	[illegible]
2	PKG EINS	12	2018-07-11 02:33:03 -0100	larry
3	PKG ZWEI	17	2018-07-18 01:01:05 -0100	hugo

FUNKTIONIERT

Lässt sich zu Beginn jeder Transformation einfach loggen.

Zusätzlich lässt sich mit dem Keyword 'HeadURL' auch der Pfad der Datei inkl Dateiname einfügen..

Git: Keyword Substition durch Smudge- und Clean-Filter

- Versionsinfo wird *nicht* mit eingecheckt (wg. Checksumme)
- Smudge-Filter schreibt *nach* dem Auschecken ins Working Directory die Commit-Info in die Datei
- Clean-Filter bereinigt Datei wieder, *bevor sie* in die Staging Area geschrieben wird..

Konfiguration:

```
git config --local filter.sql.smudge 'git-filter-sql-smudge %f'
git config --local filter.sql.clean git-filter-sql-clean
```

Git: Keyword Substition durch Smudge- und Clean-Filter

itspeople

Keywords nach dem Auschecken mit smudge ersetzt:

```
$Author: larry $: Author of last commit
$Date: 2006-03-15 02:33:03 $: Date of last commit
$Commit: b06617d2b166550cc8603fb01710045c23cf475c $
```

Beim Stagen mit clean bereinigt:

```
$Author$: Author of last commit
$Date$: Date of last commit
$Commit$
```

Git: Keyword Substition durch Smudge- und Clean-Filter

itspeople

- In der Theorie gut und (leidlich) schön
- Die Git-SHAs sind nicht so sprechend wie die Subversion-Ids
- In der Praxis zeigte Git oft ungeänderte Dateien als geändert an.
  Bedeutet: Manuelle Nacharbeiten, umständlich, konstantes Ärgernis
  => Wurde komplett entfernt

Welcher Softwarestand?

Problem ist seit Git-Einführung ungelöst.

Jammern hilft
(manchmal)

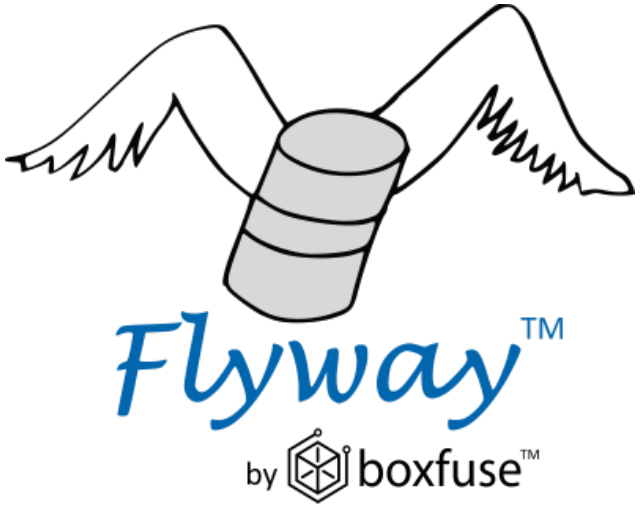
Sachdienliche
Hinweise und Ideen
werden gerne
angenommen!

Die Historie

Umstellung von Subversion auf Git.

Unabhängig davon besteht das Problem unserer selbstgestrickten Lösung weiter:
Es wurden immer wieder Skripte vergessen, Nachvollziehbarkeit war suboptimal

- Evaluation Flyway und Liquibase, jeweils die kostenlose Open-Source-Version
- Entscheidung für Flyway



Gründe:
- Flyway installiert alles, was den Namenskonventionen entspricht
- Wenig Zeit
- In SQL und PL/SQL ist unser Team einfach fit

Die Historie

Flyway wird im Projekt eingeführt.

- Wird nur für Delta-Skripte genutzt (Dateien mit Endung .sql)
- Packages werden weiterhin manuell installiert,
  da in Flyway zu diesem Zeitpunkt nur *eine* Endung konfiguriert werden kann
  (SQL Developer benötigt Endungen .pkg und .pkb zu Unterscheidung
  zwischen Packages und Skripts)
- Enhancement Request für mehrere Endungen erstellt,
  Eigenimplementierung angeboten, war aber gar nicht mehr nötig ☺

Die Historie

Neu: Flyway kann mehrere Endungen verarbeiten
(Wird ab sofort für .sql, .pkg, .pkb genutzt!)

- Alle Dateien werden ab jetzt über Flyway installiert,
  Packages, Views und Materialized Views als ‚repeatable Migration' nach den Deltas

- Problem:
  Unser Wrapper-Package für DDL wird für die Delta-Skripte benötigt und muss daher immer vorweg als allererstes Delta-Skript laufen

- Workaround: Wird jedes Mal händisch in Delta-Skript kopiert, da es in der aktuellen Version nur eine Datei beforeMigrate.sql gibt, die keine anderen Dateien aufrufen kann*

* Community Version – kein SQL*Plus Support

Die Historie

Neu: Flyway bietet die Möglichkeit, mehrere SQL-Callbacks zu einem Event auszuführen

- Beispiel:
  beforeMigrate.sql
  beforeMigrate__01_install_wrapper_package.pkg
  beforeMigrate__02_install_wrapper_package.pkb

… wieder ein Problem gelöst ☺

Dateien werden vom SQL Developer im jeweils richtigen Editor geöffnet!

Auftritt: Liquibase

Auftritt: Liquibase

Getriggert durch unser Partnerprojekt, der von "selbstgeschrieben" auf Liquibase umstellt, wollen wir auch wegen ….

- Einheitlichkeit
- Viel Gutes über Liquibase gehört (dort scheint das Gras grüner zu sein)
- Neugier ☺

Liquibase ganz kurz

Changelog-Datei

- Umfasst 1 oder mehrere Chancesets
- Changeset kann abstrakt beschrieben werden (XML, YAML, JSON) oder mit SQL
- Aufruf der Changelog-Dateien über eine Master-Datei

Ausgeführte Changesets werden in zentraler Tabelle DATABASECHANGELOG (ähnlich Flyway: DB_VERSION_INFO) nachgehalten

Der Wechsel

Annahme: Es gibt schon eine DB mit Daten und Codebasis

Erste Möglichkeit:

Alle Create-Skripte mit Liquibase generieren (JSON, YAML, XML)

Testergebnis:

Als "normale Tabelle" generiert werden:

- Global Temporary Tables
- Materialized Views
- Partitionierte Tabellen

Unter den Tisch fielen:

- Packages
- Invisible und Virtual Columns
- Check Constaints,
  die über NOT NULL hinausgehen
- Database Links
- Object Types

Der Wechsel

Annahme: Es gibt schon eine DB mit Daten und Codebasis

Zweite Möglichkeit:

Status Quo als Baseline verwenden (DB-Export)

AKZEPTABEL

Aufsetzen einer neuen Umgebung:

1. DB-Import
2. Alle darauf aufsetzenden Delta-Skripte mit Liquibase einspielen


Flyway oder Liquibase?
Unsplash

Fazit – Flyway oder Liquibase?

Vorbedingungen in Liquibase

```
<preConditions onFailMessage="Preconditions failed - dba user,
                              10GB of free space and MSSQL or Oracle db engine">
        <and>
            <runningAs username="dba" />
            <sqlCheck expectedResult="1">
                select case when size/128.0
                  - cast(fileproperty(name, 'SpaceUsed') as int)/128.0 > 10000
                  then 1 else 0 end
                from sys.database_files
                where name = 'MyDatabase'
            </sqlCheck>
            <or>
                <dbms type="mssql" />
                <dbms type="oracle" />
            </or>
        </and>
</preConditions>
```

UNENTSCHIEDEN

Fazit – Flyway oder Liquibase?

In unserem Projekt nützt uns die DB-Unabhängigkeit durch XML/JSON/YAML-Format nichts.
(Zur Erinnerung: Anwendung in PL/SQL in Oracle-Datenbank)

Liquibase beherrscht viele Objekttypen* nicht, also brauchen wir doch wieder DDL (und zwar in PL/SQL gekapselt für Exception Handling und Wiederholfähigkeit)

* Kann durch Oracle-Extensions *teilweise* kompensiert warden,
Rest in Oracle-DML

Fazit – Flyway oder Liquibase?

Anderes SQL-File referenzieren in Liquibase

```
<changeSet author="liquibase-docs" id="sqlFile-example">
    <sqlFile dbms="h2, oracle"
             encoding="utf8"
             endDelimiter="\nGO"
             path="my/path/file.sql"
             relativeToChangelogFile="true"
             splitStatements="true"
             stripComments="true"/>
</changeSet>
```

Fazit – Flyway oder Liquibase?

Diff Report in Liquibase

Eigentlich sehr leserlich,
aber datenbankspezifische Tools können besser Oracle-Objekten umgehen

Fazit – Flyway oder Liquibase?

Liquibase braucht ein Master-File (davon wollten wir eigentlich weg)

Bietet allerdings ein bisschen mehr Flexibilität, was die Installationsreihenfolge angeht.

Fazit – Flyway oder Liquibase?

Drei verschiedene Arten der Ausführung eines ChangeSets:

Default (kein Attribut)
Wird einmalig ausgeführt

PRAKTISCH

runOnChange="true"
Wird ausgeführt, wenn sich der Hash-Wert des ChangeSets geändert hat

runAlways="true"

Wird jedes Mal ausgeführt

Fazit – Flyway oder Liquibase?

„Automatische“ Rollback-Funktionalität
Sehr basal – nur für sehr unkomplizierte Operationen.

Ein Drop ist allerdings auch mit unserem Wrapper-Package schnell erledigt.
Die komplizierten Fälle brauchen ohnehin individuelle Behandlung.

Und wer "gewinnt"?

Wie immer gilt: Es kommt drauf an

Beide Lösungen sind sehr ausgereift und jede hat ihre Vorteile.

Und wer "gewinnt"?

Erst die Fremde lehrt uns, was wir an der Heimat haben

(Theodor Fontane)

Wir bleiben bei Flyway,
weil es in unserem Projekt unsere Anforderungen am Besten erfüllt

Verbesserungspotenzial

Diskussion

Links

Flyway

- Flyway Home
- Flyway Github

Liquibase

- Liquibase Home
- Liquibase Github
- Liquibase Oracle Extension

Vergleich

- Flyway vs. Liquibase von Stephan Kaps (DOAG 2016)
- Schemamigration mit Liquibase und Flyway von Thomas Jaspers (iX Developer 2016)
- Schnellvergleich auf Stackshare