

Datenbank-Refactoring mit LiquiBase

*Agile Software-Entwicklung mit
RDBMS Refactoring & Change Management*

Benjamin Schmid

ex|**x**cellent
solutions

w-jax[®]08
Konferenz für Java™, Enterprise Architekturen, SOA

Softwareentwicklung in der Praxis

Hervorragende Lösungen beim Programmcode für:

- **Versionierung** SVN, CVS
- Build- und Test-**Automatisierung** Ant, Maven
CI-Server
- **Abstraktion** und Modellierung UML
- **Refactoring** Eclipse, IDEA



Und für die Datenbank?

Datenbank-Entwicklung in der Praxis

Welchen Stand hat die Datenbank?

*„Unknown column“: Sind deine
Änderungen schon drin?*

*Wie nenne ich noch mal
schnell die Spalte um?*

BIGINT in MySQL Und für Oracle?

*Uuups! Können wir schnell
zurück zur vorigen Version?*

*Nerv! Geht das alles
nicht auch automatisch?*



LIQUIBASE

Open-Source (LGPL)-Lösung zur
Ausführung, Management und **Dokumentation**
von Datenbankänderungen

Kern-Features:

- **Herstellerunabhängig**
- **Automatisierung**
- **Rollback-Strategie**
- **Änderungsdokumentation**
- Code-Branchfähig
- Teamfähig & **DBA-fähig'**
- IDE Support



What's so special about LiquiBase?

- **Kein** Soll/Ist **„Diff-Tool“** sondern Beschreibung der **Änderungen**
 - Erhalt der Semantik
(name → name2: *rename* oder *drop+new* ?)
- Änderungen werden **einzeln betrachtet**
 - Keine linearer Versionierung (→ Branches)
 - Change Log wird sequentiell abgearbeitet; die ausstehenden Änderungen ausgeführt
- ... und in einer Tabelle **protokolliert**



Unterstützte Datenbanken



- Oracle
- MS SQL
- MySQL
- PostgreSQL
- DB2
- Sybase
- HLSQL
- Apache Derby
- H2
- Caché
- MaxDB
- SQLite

Wie funktioniert? (1)

In einer XML-Datei/via IDE werden alle Änderungen in als **Change Sets** beschrieben & gepflegt:

```
<changeSet id="1" author="bob">
  <createTable tableName="department">
    <column name="id" type="int">
      <constraints primaryKey="true" nullable="false"/>
    </column>
    <column name="name" type="varchar(50)">
      <constraints nullable="false"/>
    </column>
    <column name="active" type="boolean" defaultValue="1"/>
  </createTable>
</changeSet>
```

ID = id + author + filename

Change Set durch MD5 Hash gesichert

Verfügbare Refactorings

- 13 **Structural** Refactorings
Add/Rename/Drop: Colum/Table/View
Merge Columns, Stored Procedure
- 10 **Data Quality** Refactorings
Add/Drop: Constraint/Lookup-Table/Sequence/Default Value
- 4 **Referential Integrity** Refactorings
Add/Drop: Foreign & Primary Keys
- 5 **Non-Refactoring Transformations**
Insert/Load/Update/Delete Data
- 4 **Custom** Refactorings
Custom SQL/Refactoring Class/Shell Command
- 2 **Index** Refactorings

Wie funktioniert? (2)

Ausführung der Change Sets über LiquiBase Runner:
Ant, Maven, Kommandozeile,
Servlet Listener, via API, Spring, Grails.

Beispiel in Ant:

```
<taskdef resource="liquibasetasks.properties">
  <classpath refid="classpath"/>
</taskdef>

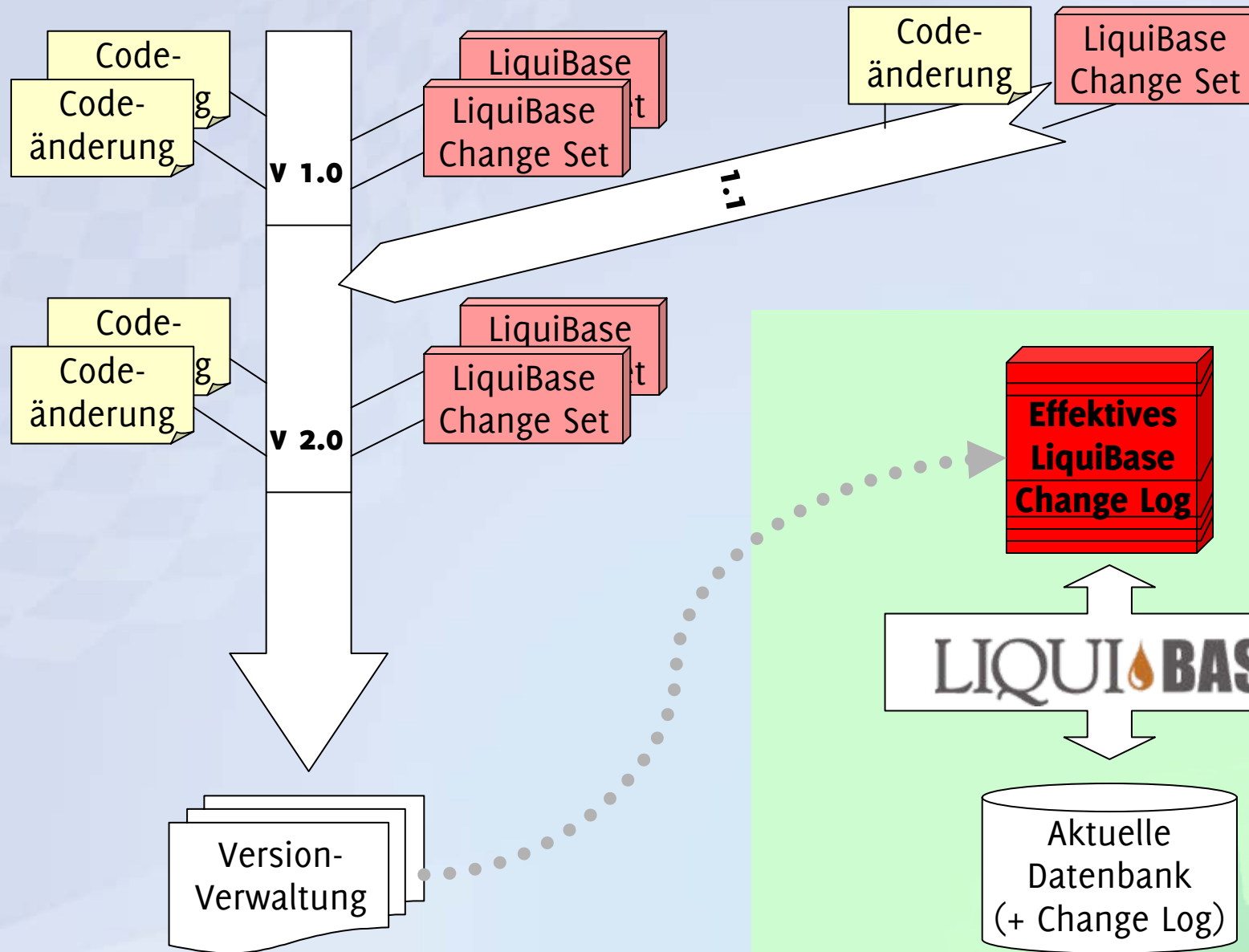
<target name="update-database">
  <updateDatabase changeLogFile="src/rdbms/changelog.xml"
    driver="net.sourceforge.jtds.jdbc.Driver"
    url="jdbc:jtds:sqlserver://db-server.excellent.de/devel"
    username="bschmid" password="bschmid"
    dropFirst="false" classpathref="classpath"/>
</target>
```

Wie funktioniert? (3)

Folgende Operationen sind möglich:

- **Automatische Migration** auf akt. Stand
- Nur **Prüfung** auf offene Änderungen
- **DDL-Skript Generierung**
(inkl. zugehörigem **Rollback-Skript**)
- **Change Sets** aus Datenbank-**Diff erzeugen**
(Special Feature: Diff gegen **Hibernate**-Mapping!)
- **Rollback**, Datenbankstand **taggen**,
Änderungsdokumentation, Drop-all, ...

Änderung von Code & Datenbank über die Zeit



IDE Support

Hervorragende verwendbar mit IDE-Bordmitteln
Umfangreiches IntelliSense dank XML Schema

- **IntelliJ IDEA Plugin**
Aktuellste Fassung (v1.8.0) und größter Umfang
- **Eclipse Plugin**
Alpha-Status und aktuell veraltet (v0.6)
- **Standalone IDE**
Veraltet und eher Technologie-Demo (v 0.6)

IDE Demo

PowerPoint iss' doof...

Tagging & Rollback

Rollback-Wege:

automatisch

- ✓ create table → drop table
- ⚠ drop table → *n/a*

explizite Definition:

- ✓ LiquiBase Refactorings
- ✓ SQL Anweisungen
- ✓ Re-do Change Set Y

Rollbackpunkte:

- Datum / Uhrzeit
- Number of Changes
- Markierte Version (Tag)

```
<changeSet author="bschmid" id="03-insert-admin">
  <comment>Admin-Person anlegen</comment>
  <sql>insert into person(id,vorname,name) values ('1','admin','admin');</sql>
  <rollback>
    <sql>delete from person where id = 1;</sql>
  </rollback>
</changeSet>
```

Weitere Features

- **MD5 gesicherte Change Sets**
(Erkennen nachträglicher Änderungen)
- *runalways* bzw. *runonchange*
- Generierung einer **Änderungsdokumentation**
- **Hibernate Support**
(diff gegen Mapping Dateien: *.hbm.xml)
- **Clusterfähig** (Locking) und **Contexts**
- Preconditions und Assertions zur Zielplattform

Änderungsdokumentation

[Current Tables](#)
[Authors](#)
[Change Logs](#)
[Pending Changes](#)
[Pending SQL](#)
[Most Recent Changes](#)

Current Tables
[add_column_test_table](#)
[address](#)
[book](#)
[commenttest](#)
[company](#)
[compoundindextest](#)
[compoundpktest](#)
[contextstest](#)
[datatypeconversiontest](#)
[datatypepetest](#)
[defaultvaluetest](#)
[liquibaseruninfo](#)
[news](#)
[nulldefaulttest](#)
[person](#)
[pktest](#)
[state](#)
[tablespace_test_table](#)
[tabletorollback](#)

Changes affecting table "address"

Current Columns

int	id
varchar(255)	line1
varchar(255)	line2
varchar(255)	city
char(2)	state
varchar(20)	postalcode

Pending Changes

changelogs/mysql/complete/root.changelog.xml	54	nvoxland	NOT YET RAN [SQL]
Column newcol(varchar(255)) added to address			
changelogs/mysql/complete/root.changelog.xml	55	nvoxland	NOT YET RAN [SQL]
Column newcol2(varchar(255)) added to address			

Past Changes

changelogs/mysql/complete/root.changelog.xml	25	nvoxland	Executed 11/15/07 9:37 AM
Table address created			
changelogs/mysql/complete/root.changelog.xml	25.1	nvoxland	Executed 11/15/07 9:37 AM
Null constraint has been added to address.id			
changelogs/mysql/complete/root.changelog.xml	25.2	nvoxland	Executed 11/15/07 9:37 AM

Fazit

Mit Liquibase können Sie spielend

- **Automatisiert** beim Start ihre Datenbanken **migrieren**
- Die Datenbank **adäquat** wie Ihren Code **behandeln**
- **DBA-konform** Migrations- & Rollback Skripte erzeugen

Und durch den Automatismus ideal für

- Software-Entwicklung im **Team**
- **Continuous Integration** und **Testing**
- einen **agilen** und **robusten** Entwicklungsprozess
- **regelmäßige Auslieferungen**



Referenzen



Kontakt:

B.Schmid(at)exxcellent.de
oder an unserem Stand

LiquiBase

<http://www.liquibase.org/>

Materialien zum Thema

Refactoring & Code Qualität

Der 10 Punkte Plan zum unwartbaren Code,
Automatisiertes GUI-Testing, Refactorings, ...

<http://www.exxcellent.de/download.html>

Alternativen (OSS)

dbdeploy *<http://dbdeploy.com>*

MIGRATEdb *<http://migratedb.sourceforge.net/>*