Liquibase ile Veri Tabanı Değişiklik Yönetimi

Ahmet Deniz Korkmaz

deniz.korkmaz@ozguryazilim.com.tr

LIQUIBASE

- Versiyon kontrolü olmadan kod yazmıyoruz. Peki veritabanı?
- Veritabanı değişiklik, versiyonlama yönetimi
  - Açık Kaynak Kodlu
  - LGPL
  - Veritabanı bağımsız
  - Veritabanı değişikliklerini;
    - Uygulama
    - Yönetme
    - İzleme

LIQUIBASE

- Genişletilebilir
- Birden çok geliştiricinin değişikliklerini birleştirebilir. (merge)
- Birden çok veritabanı destekler
- Ürün veritabanı yanında test verilerini de yönetir
- Değişiklikleri geri alabilir (rollback)
- Veritabanları arasında ki farkları alabilir (diff)
- Esnektir (hemen uygula yada sonra uygula)

Genel Bakış

- Veri tabanı değişiklikleri (databaseChangeLog.xml) XML dosyasına yazılır.
- Local veritabanına uygulanır.
- Commit edilir ve diğer kullanıcılara dağıtılır.
- Diğer kullanıcılar alır ve değişiklikleri uygular(Ant, Maven, komut satırı, Servlet Listener, Grails, Spring)

Genel Bakış

- Her bir girdi “id” ve “author” elemanına sahip.
- Bunlarla birlikte dosya yolu tek bir kimlik oluşturur (unique ID). Neden sadece ID değil?
- Liquibase girdileri tablodan kontrol eder (databaseChangeHistory)
- Tabloda yoksa değişiklik uygulanır ve tabloya kayıt edilir.

ChangeLog XML

- Kolay okunabilir
- Kolay takip edilebilir
- Versiyonlama sistemine yüklenebilir
- Bütün veritabanı değişiklikleri bu dosyada listelenir
-

Geliştirici İçin Süreç

- XML içine değişiklikleri yazar. (changeSet)
- Liquibase ile bunu çalıştırır ve veritabanına uygular
- Veritabanı ile bağımlı olan değişiklikleri uygulama koduna (JAVA) yazar
- Veritabanı ve uygulamayı birlikte test eder.
- İkisini birden versiyonlama sistemine yükler(SVN commit)

<databaseChangeLog> Tag

- Ana etikettir (root tag).
- Özellikleri
  - LogicalFilePath : Kimlik oluştururken kullanılan dosya yolunu günceller
- Alt etiketler
  - Preconditions : değişiklikleri çalıştırabilmek için ön koşullar.
  - Changesets : uygulanacak değişiklikler kümesi
  - Include : değişiklikleri içeren ek dosyalar

<databaseChangeLog> Tag

- Liquibase çalıştığında
  - Bu tagi çözümler
  - Ön koşul var mı kontrol eder. Koşula uymayan bir şey olursa durur ve hata verir
  - Sonra yazıldıkları sıra ile changeSet ve include taglarini çalıştırır

<changeSet> Tag

- Yapılacak değişiklikleri gruplara ayırmamızı sağlar
- Unique ID : id, author, changeLog dosya yolu
- Yazıldıkları sıra ile okunup çalıştılırlar
- Her biri içim tarihçe tablosu kontrol edilir. (id/author/dosya yolu)
- Tabloda yoksa çalıştırılır varsa es geçilir.
- Checksum

<changeSet> Tag

- Özellikler
  - Id : kimlik
  - Author : changeSeti yazan kullanıcı
  - Dbms : veritabanı tip
  - RunAlways : tarihçe tablosunda varsa bile çalıştır
  - Context : Herhangi bir metin girilebilir
- Alt etiketler
  - Comment, preConditions, Refactoring Tags, rollback

Değişiklik Komutları

- Yapısal değişikliler
- Veri kalitesi değişiklikleri
- Tablo bağımlılıkları
- Veri ekleme, çıkarma...
- Özel değişiklikler

<include> tag

- Proje büyüdükçe changeSet sayısı artar
- Değişiklikleri yönetmek zorlaşır
- ChangeLog ağacı
- Birden fazla changeLog içeri aktarılabilir
- Yazıldıkları sıra ile çalıştırılırlar(onemli)
  - Sonsuz döngü, 2 kere yazım
- İd/author ikilisi her bir changeLog'da tektir.

```
<include file="com/example/news/news.changelog.xml"/>
<include file="com/example/directory/directory.changelog.xml"/>
```

<includeAll> tag

- <include> ile çok benzer
- Özel bir dosya adı vermek yerine dosya yolu veriyoruz
- Bütün .xml dosyalarını changelog olarak alır ve harf sırasına göre çalıştırır

<preConditions>

- Veritabanı değişikliklerini uygulamadan önce kontrol yapmamızı sağlar
- Varsayımları belgelemek için
- Varsayımlar ihlal ediliyor mu görmek için
- Geri dönüşü olmayan değişikliklerde veri kontrolü yapmak için
- Hangi changeSet çalıştı hangisi çalışmadı

<preConditions>

```
<preConditions>
        <dbms type="oracle" />
        <runningAs username="SYSTEM" />
    </preConditions>


    <changeSet id="1" author="bob">
        <preConditions onFail="WARN">
            <sqlCheck expectedResult="0">select count(*) from oldtable</sqlCheck>
        </preConditions>
          <comment>Comments should go after preCondition. If they are before then liquibase usually gives error.</comment>
        <dropTable tableName="oldtable"/>
    </changeSet>
```

Başarısız olma ve hata

- Özellikler : onFail, onError, onFailMessage, onErrorMessage
- Değerler: HALT, CONTINUE, MARK_RAN, WARN
- ChangeSet dışında sadece HALT ve WARN kullanılabilir

AND/OR/NOT Logic

```
<preConditions>
    <or>
        <and>
            <dbms type="oracle" />
            <runningAs username="SYSTEM" />
        </and>
        <and>
            <dbms type="mssql" />
            <runningAs username="sa" />
        </and>
    </or>
 </preConditions>
```

PreCondition Çeşitleri

- dbms,
- runningAs,
- ChangeSetExecuted : id/author/changeLogfile
- columnExists : schemaName, tableName, columnName
- tableExists
- ForeignKeyConstraintExists : schemaName, foreignKeyName)
- PrimaryKeyExists : tableName, primaryKeyName
- SqlCheck : <sqlCheck expectedResult="1">SELECT COUNT(1) FROM pg_tables WHERE TABLENAME = 'myRequiredTable'</sqlCheck>

Contexts

- ChangeSet'e eklenen bir etiket
- Liquibase çalıştığında hangi değişikliklerin (changeSets) çalıştırılacağını belirler
- Herhangi bir metin girilebilir
  - Çalışırken parametre gibi context adi yollanır
  - Yollanan contexte sahip değişiklikler çalışır
  - Değişiklikte context etiketi yoksa her zaman calisir. Parametre olarak ne yollanırsa yollansın
  - Parametre yollanmazsa bütün hepsi çalışır

Contexts

```
<changeSet id="2" author="bob" context="test">
    <insert tableName="news">
        <column name="id" value="1"/>
        <column name="title" value="Liquibase 0.8 Released"/>
    </insert>
    <insert tableName="news">
        <column name="id" value="2"/>
        <column name="title" value="Liquibase 0.9 Released"/>
    </insert>
</changeSet>
```

Change Log Parametreleri

- Liquibase parametre kullanimina izin verir
- ${parametre}
- Parametre degerleri:
  - Liquibase calistiginda parametre olarak geldi mi
  - JVM den mi

```
<createTable tableName="${table.name}">
    <column name="id" type="int"/>
    <column name="${column1.name}" type="varchar(${column1.length})"/>
    <column name="${column2.name}" type="int"/>
</createTable>
```

Hibernate Bağlantısı

- Hibernate hbm2dll operesyonu
- İhtiyaçları karşılamıyor
- Liquibase mapping dosyasını tarar ve değişiklikleri change log dosyasına yazar.
- Bu change log dosyasını kullanarak değişiklikler uygulanır

Geliştirme Aşaması

- Hibernate mapping dosyasında değişikliker yapılır
- Bu dosya ile database arasında diff alınır
- Oluşan change log dosyası ile veritabanında güncelleme yapılır.
- liquibase \

*--classpath=jdbcdriver.jar:hibernate.jar *

*--changeLogFile=path/to/changelog *

*--url=hibernate:YOUR_HIBERNATE.CFG.XML *

*diffChangeLog *

*--referenceDriver=oracle.jdbc.OracleDriver *

*--referenceUrl=jdbc:oracle:thin:@localhost:1521:oracle *

*--referenceUsername=scott *

--referencePassword=tiger

Biçimlendirilmiş SQL

- Change Log dosyasına klasik SQL ifadeleri yazılabilir
-

Biçimlendirilmiş SQL

```
--liquibase formatted sql

--changeset deniz:1
CREATE TABLE test1 (
    id int PRIMARY KEY,
    name varchar(255)
);
--rollback drop table test1;

--changeset deniz:2
INSERT INTO test1 (id, name) VALUES (1, ‘name 1´);
INSERT INTO test1 (id, name) VALUES (2, ‘name 2´);

--changeset deniz:3 dbms:oracle
CREATE sequence seq_test;
```


Liquibase Operasyonları

- Update
- Rollback
- Diff
- Generate Change Log
- DBDoc
- SQL output

Update

- Change Log dosyasına yazılan veritabanı değişikliklerinin uygulanması içindir
- ChangeSet sırasıyla okunur
- Id/author/dosya yolu ve MD5sum hash kod
  - Önce kimliğe bakar, tabloda yoksa çalıştırır
  - Varsa hash koda bakar farklı ise hata verir.
  - Hash kod da aynı ise o ChangeSet'i atlar
  - RunOnChange, RunAlways
- Update ve updateCount

Update

- Migrate komutu veritabanını günceller
- MigrateSql komutu veritabanını güncellemek için gerekli SQL dosyası üretir

Rollback

- Yapılan değişiklikleri geri alma. <rollback>
- Otomatik yada Sql ifadeleri ile
- CreateTable, RenameColumn gibi çoğu değişikliklerde otomatik üretilir
- DropTable, insertData el ile yazılmalı
- <rollback> içine birden fazla SQL ifadesi

```
<changeSet id="multiRollbackTest" author="rs">
        <createTable tableName="multiRollback1">
            <column name="id" type="int"/>
        </createTable>
        <createTable tableName="multiRollback2">
            <column name="id" type="int"/>
        </createTable>
        <createTable tableName="multiRollback3">
            <column name="id" type="int"/>
        </createTable>
        <rollback>
            drop table multiRollback1;
            drop table multiRollback2;
        </rollback>
        <rollback>drop table multiRollback3</rollback>
    </changeSet>
```

Diff

- Veritabanı karşılaştırması
  - liqubase.sh --driver=oracle.jdbc.OracleDriver \
  - --url=jdbc:oracle:thin:@testdb:1521:test \
  - --username=bob \
  - --password=bob \
  - diff \
  - --referenceUrl=jdbc:oracle:thin:@localhost/XE \
  - --referenceUsername=bob \
  - --referencePassword=bob

Karşılaştırmalar

- Versiyon farkları
- Eksik yada olması gereken tablolar, view, kolonlar, primary key, constraint, foreign key, sequence, index,
- Kolon tanımlama farkları (veri tipi, otomatik artırma)
- Veri farkları
  - Karşılaştırma yapamadıkları: Stored Procedures, Veri Uzunlukları
  - Rapor ya da Change Log dosyası

Diff Parametreleri

- tables [DEFAULT]
- columns [DEFAULT]
- views [DEFAULT]
- primaryKeys [DEFAULT]
- indexes [DEFAULT]
- foreignKeys [DEFAULT]
- sequences [DEFAULT]
- data
-

Problemler

- Yapısal karşılaştırma yapıyor fakat anlamsal (semantik) karşılaştırma yapamıyor.
  - Kolon adı değişikliği
- Veri karşılaştırma

Change Log Üretme

```
liquibase --driver=oracle.jdbc.OracleDriver \
    --classpath=\path\to\classes:jdbcdriver.jar \
    --changeLogFile=com/example/db.changelog.xml \
    --url="jdbc:oracle:thin:@localhost:1521:XE" \
    --username=scott \
    --password=tiger \
    generateChangeLog
```