

# mvdb's: Übung Trigger

Yanick Eberle, Pascal Schwarz

22. Februar 2013

## 1 Aufgabe 1 - Event Logging

### 1.1 Lösungsidee

Wir erstellen einen Trigger, welcher bei den SQL Statements, die potenziell Änderungen an der Tabelle „Ausleihen“ bewirken, ausgelöst wird. Wie in der Aufgabenstellung beschrieben sind dies die folgenden SQL-Befehle:

- INSERT
- UPDATE
- DELETE

Aufgrund der Anforderung sowohl die alten wie auch die neuen Werte zu protokollieren, muss unser Trigger jeweils vor dem Statement ausgeführt werden. In einem solchen Trigger haben wir Zugriff auf die neuen Werte.

### 1.2 Tabelle Ausleihen\_Log

Die Log-Tabelle enthält einen eigenen Primary Key (Number(6,0)). Die geforderten Angaben (User, welcher die Änderung vorgenommen hat (VARCHAR2(20)), Art der Änderung (VARCHAR(3)) sowie Zeitpunkt der Änderung (DATE)) werden jeweils in einem Attribut abgelegt.

Zusätzlich erhält die Log-Tabelle für jedes Attribut der Tabelle Ausleihen zwei Attribute. In *Feldname\_old* wird der Wert vor der Änderung, in *Feldname\_new* der Wert nach der Änderung festgehalten. Diese Attribute haben jeweils den selben Datentyp wie das jeweilige Attribut in der Tabelle „Ausleihen“.

### 1.3 Trigger für Protokollierung

Da ein Statement grundsätzlich mehrere Zeilen der Tabelle „auf einmal“ verändern kann, muss der Trigger mit der Granularität „FOR EACH ROW“ definiert werden.

Wie bei der Lösungsidee bereits beschrieben wird der Trigger vor den Events INSERT, UPDATE und DELETE ausgelöst.

## 1.4 SQL Statements

### 1.4.1 Ausleihen\_Log

```
1 CREATE TABLE AUSLEIHEN.LOG
2 (
3     log_id NUMBER(6, 0) NOT NULL
4     , CHANGE_USER VARCHAR2(20) NOT NULL
5     , CHANGE_DATE TIMESTAMP NOT NULL
6     , CHANGE_TYPE VARCHAR2(3) NOT NULL
7     , MNR_OLD VARCHAR2(4)
8     , MNR_NEW VARCHAR2(4)
9     , DVDNR_OLD NUMBER(6, 0)
10    , DVDNR_NEW NUMBER(6, 0)
11    , DATUM_OLD DATE
12    , DATUM_NEW DATE
13    , RUECKGABE_OLD DATE
14    , RUECKGABE_NEW DATE
15    , CONSTRAINT AUSLEIHEN.LOG_PK PRIMARY KEY (log_id) ENABLE
16 );
```

### 1.4.2 Trigger

```
1 CREATE OR REPLACE TRIGGER ausleihen_logger
2     BEFORE UPDATE OR INSERT OR DELETE ON ausleihen
3     FOR EACH ROW
4     DECLARE
5         manipulation varchar2(3) := 'asd';
6     BEGIN
7         if inserting then
8             manipulation := 'INS';
9         elsif deleting then
10            manipulation := 'DEL';
11        elsif updating then
12            manipulation := 'UPD';
13        else
14            manipulation := 'ERR';
15        end if;
16
17        INSERT INTO ausleihen_log (change_user, change_date, change_type, mnr_old,
18            mnr_new, dvdnr_old, dvdnr_new, datum_old, datum_new, rueckgabe_old,
19            rueckgabe_new)
20        VALUES (user, sysdate, manipulation, :old.mnr, :new.mnr, :old.dvdnr, :new.dvdnr,
21            :old.datum, :new.datum, :old.rueckgabe, :new.rueckgabe);
22    END;
```

## 1.5 Tests

## 2 Aufgabe 2 - Referential Integrity

### 2.1 Lösungsidee

Für das Verschieben der Tabelle „Filme“ muss der Foreign Key Constraint „DK\_FM\_FK“ auf der Tabelle „DVDKopien“ zunächst entfernt werden. Ansonsten kann die Tabelle nicht entfernt werden.

Als nächster Schritt wird ein Database-Link auf dem Server telesto (dort sind alle Tabellen ausser Filme) erstellt:

```
1 create database link orion.helios.fhnw.ch
2 connect to mvds01 identified by mvds01
3 using 'orion'
```

Damit die entfernte Tabelle so benutzt werden kann als wäre sie auf diesem Server wird noch ein SYNONYM erstellt:

```
1 create synonym filme for filme@orion.helios.fhnw.ch;
```

## **2.2 Entwurf der Trigger**

## **2.3 SQL Trigger**

## **2.4 Tests**