Lab 01: Trigger

# Event Logging

## Datenbank Link erstellen

Auf sirius.cs.technik.fhnw.ch die Beispieldaten einfügen. Danach auf janus.cs.technik.fhnw.ch einen Link auf sirus.cs.technik.fhnw.ch (Service ganymed.sirius.fhnw.ch) erstellen:

|  |
| --- |
| CREATE DATABASE LINK ganymed.sirius.fhnw.ch  CONNECT TO ddm22 IDENTIFIED BY ddm22  USING 'ganymed'; |

Verbindung testen:

|  |
| --- |
| SELECT \* FROM filme@ganymed.sirius.fhnw.ch; |

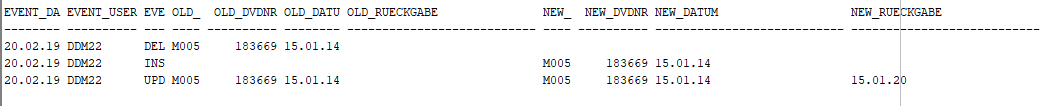
## Trigger erstellen

|  |
| --- |
| /\* Recreate existing data to make the script deterministic \*/  TRUNCATE TABLE AUSLEIHEN;  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M002',158234,TO\_DATE('19.07.2013','dd.mm.yyyy'),TO\_DATE('21.07.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M004',158234,TO\_DATE('02.08.2013','dd.mm.yyyy'),TO\_DATE('04.08.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M003',269260,TO\_DATE('05.01.2014','dd.mm.yyyy'),NULL);  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M003',199004,TO\_DATE('05.01.2014','dd.mm.yyyy'),NULL);  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M001',310094,TO\_DATE('22.11.2013','dd.mm.yyyy'),TO\_DATE('27.11.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M001',468123,TO\_DATE('19.01.2014','dd.mm.yyyy'),NULL);  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M002',183669,TO\_DATE('30.11.2013','dd.mm.yyyy'),TO\_DATE('01.12.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M004',183669,TO\_DATE('27.12.2013','dd.mm.yyyy'),TO\_DATE('03.01.2014','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M005',183669,TO\_DATE('15.01.2014','dd.mm.yyyy'),NULL);  COMMIT;  /\* Create the log table \*/  DROP TABLE AUSLEIHEN\_LOG;  CREATE TABLE AUSLEIHEN\_LOG (  EVENT\_DATE DATE,  EVENT\_USER VARCHAR(10),  EVENT\_ACTION VARCHAR(3),  OLD\_MNR VARCHAR2(4),  OLD\_DVDNR NUMBER(6),  OLD\_DATUM DATE,  OLD\_RUECKGABE DATE,  NEW\_MNR VARCHAR2(4),  NEW\_DVDNR NUMBER(6),  NEW\_DATUM DATE,  NEW\_RUECKGABE DATE  );  /\* Create the trigger to log events in the log table \*/  CREATE OR REPLACE TRIGGER AUSLEIH\_TRIGGER  AFTER INSERT OR UPDATE OR DELETE ON AUSLEIHEN  FOR EACH ROW  DECLARE  TMP\_EVENT\_TYPE VARCHAR(3);  TMP\_OLD\_MNR VARCHAR2(4) := :OLD.MNR;  TMP\_OLD\_DVDNR NUMBER(6) := :OLD.DVDNR;  TMP\_OLD\_DATUM DATE := :OLD.DATUM;  TMP\_OLD\_RUECKGABE DATE := :OLD.RUECKGABE;  TMP\_NEW\_MNR VARCHAR2(4) := :NEW.MNR;  TMP\_NEW\_DVDNR NUMBER(6) := :NEW.DVDNR;  TMP\_NEW\_DATUM DATE := :NEW.DATUM;  TMP\_NEW\_RUECKGABE DATE := :NEW.RUECKGABE;  BEGIN  /\* Define the action type \*/  IF INSERTING THEN  TMP\_EVENT\_TYPE := 'INS';  ELSIF UPDATING THEN  TMP\_EVENT\_TYPE := 'UPD';  ELSIF DELETING THEN  TMP\_EVENT\_TYPE := 'DEL';  END IF;    /\* Insert into the log table \*/  INSERT INTO AUSLEIHEN\_LOG(  EVENT\_DATE, EVENT\_USER, EVENT\_ACTION,  OLD\_MNR, OLD\_DVDNR, OLD\_DATUM, OLD\_RUECKGABE,  NEW\_MNR, NEW\_DVDNR, NEW\_DATUM, NEW\_RUECKGABE  ) VALUES (  SYSDATE, USER, TMP\_EVENT\_TYPE,  TMP\_OLD\_MNR, TMP\_OLD\_DVDNR, TMP\_OLD\_DATUM, TMP\_OLD\_RUECKGABE,  TMP\_NEW\_MNR, TMP\_NEW\_DVDNR, TMP\_NEW\_DATUM, TMP\_NEW\_RUECKGABE  );  END;  /\* The character / flushes the buffer - I have no idea why this is required \*/  / |

## Testszenario

|  |
| --- |
| /\* Delete an existing rental \*/  DELETE FROM AUSLEIHEN WHERE mnr = 'M005';  /\* Reinsert the deleted rental \*/  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M005',183669,TO\_DATE('15.01.2014','dd.mm.yyyy'),NULL);  /\* Update the reiinserted rental \*/  UPDATE AUSLEIHEN SET rueckgabe = TO\_DATE('15.01.2020','dd.mm.yyyy') WHERE mnr = 'M005';  /\* Print the result to verify it \*/  SELECT \* FROM AUSLEIHEN\_LOG; |

Ausgabe:



## Quellenangabe

* Oracle Dokumentation, welche eine Grobübersicht gibt (Timing Points/Wie kann man auf alte/neue Werte zurückgreifen/User etc.): <https://docs.oracle.com/database/121/TDDDG/tdddg_triggers.htm#TDDDG52000>

# Referential Integrity

## Aufgabenstellung

Die Entität Filme wird vom eigentlichen Datenbankserver auf einen anderen Datenbankserver ausgelagert. Gedanken:

* Die referenzielle Integrität erlischt über Linkgrenzen hinweg
* Die Tabelle Film wird auf einen anderen Server ausgelagert
* Ein Film kann ohne DVDKopien existieren, nicht aber umgekehrt (
* Folglich müssen beide

Dies führt zu folgenden CRUD Situationen:

* INSERT FILM: Kein Check notwendig
* UPDATE FILM: Der Primary Key eines Filmes darf nicht geändert werden oder aber der neue Primary Key muss in DVDKopien angepasst werden (Mühsam, da Primary Keys in der Regel eh nicht ändern)
* DELETE FILM: Ein Film darf nur gelöscht werden, wenn es keine DVDKopien von ihm gibt
* INSERT DVDKOPIEN: Eine DVDKopie benötigt einen gültigen Film
* UPDATE DVDKOPIEN: Eine DVDKopie benötigt einen gültigen Film
* DELETE DVDKOPIEN: Kein Check notwendig

Dies führt zu folgenden Triggern:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Entität | Action | Timing | Beschreibung |
| EnsureFilmImmutability | Film | UPDATE | BEFORE | Verhindert Änderungen der Film ID |
| EnsureFilmReferences | Film | DELETE | BEFORE | Verhindert die Löschung eines Filmes, wenn es von diesem noch Kopien gibt |
| EnsureDvdKopienReferences | DVDKopien | INSERT/ UPDATE | BEFORE | Verhindert die Erstellung oder Änderung einer DVDKopie, wenn diese auf einen ungültigen Film zeigt |

## Ausgelagerte Datenbank präparieren

Auf der ausgelagerten Datenbank janus.cs.technik.fhnw.ch wird die Film Tabelle samt Daten erstellt (Ein Link zu sirius.cs.technik.fhnw.ch existiert ja bereits aus der ersten Übung):

|  |
| --- |
| DROP TABLE filme;  CREATE TABLE filme (  katalognr NUMBER(4) NOT NULL,  titel VARCHAR2(30) NOT NULL,  mindestalter NUMBER(2) NOT NULL,  gebuehr NUMBER(4,2) NOT NULL,  CONSTRAINT fm\_pk PRIMARY KEY(katalognr)  );  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2028,'Casablanca',9,8.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (1245,'Ocean''s Eleven',12,9.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2239,'A Space Odyssee',12,7.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (1062,'Pulp Fiction',16,8.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2588,'The Pelican Brief',12,8.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (1672,'Erin Brockovich',9,8.90);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2468,'Ratatouille',6,7.50);  COMMIT; |

## Hauptdatenbank manipulieren

Auf der Hauptdatenbank muss nun die ausgelagerte Datenbank via Link hinzugefügt und getestet werden (Der Link zwischen den beiden Datenbankservern wird jetzt symmetrisch):

|  |
| --- |
| CREATE DATABASE LINK titania.janus.fhnw.ch  CONNECT TO ddm22 IDENTIFIED BY ddm22  USING 'titania';  SELECT \* FROM FILME@titania.janus.fhnw.ch; |

Nun löscht man aufgrund der referenziellen Integrität auf dem Hauptserver am besten alle Daten (<https://stackoverflow.com/questions/1690404/how-to-drop-all-user-tables>) und auch den in Übung 1 erstellten Trigger:

|  |
| --- |
| select 'drop table '||table\_name||' cascade constraints;' from user\_tables;   * Danach den drop Output kopieren und ausführen * Trigger auch löschen |

Bevor nun die eigentlichen Daten wieder hinzugefügt werden können, müssen die beiden Tabellen im SQL Skript wie folgt angepasst werden (Wir müssen die Tabelle Filme aus der Referenz nehmen, da wir die referenzielle Integrität ja mit Triggern absichern):

Nun passen wir das initiale Import SQL Skript an und entfernen das DROP/CREATE TABLE FILME samt referenziellen Integritätscheck und Filmerstellung:

|  |
| --- |
| DROP TABLE ausleihen;  DROP TABLE registrierungen;  DROP TABLE dvdkopien;  /\*DROP TABLE filme;\*/  DROP TABLE angestellte;  DROP TABLE filialen;  DROP TABLE mitglieder;  CREATE TABLE mitglieder (  mnr VARCHAR2(4) NOT NULL,  name VARCHAR2(15) NOT NULL,  wohnort VARCHAR2(10) NOT NULL,  gdatum DATE NOT NULL,  CONSTRAINT mg\_pk PRIMARY KEY(mnr)  );  CREATE TABLE filialen (  fnr VARCHAR2(3) NOT NULL,  ort VARCHAR2(10) NOT NULL,  plz VARCHAR2(4) NOT NULL,  CONSTRAINT fl\_pk PRIMARY KEY(fnr)  );  CREATE TABLE angestellte (  anr VARCHAR2(4) NOT NULL,  name VARCHAR2(10) NOT NULL,  salaer NUMBER(6,2) NOT NULL,  CONSTRAINT an\_pk PRIMARY KEY(anr)  );  /\*CREATE TABLE filme (  katalognr NUMBER(4) NOT NULL,  titel VARCHAR2(30) NOT NULL,  mindestalter NUMBER(2) NOT NULL,  gebuehr NUMBER(4,2) NOT NULL,  CONSTRAINT fm\_pk PRIMARY KEY(katalognr)  );\*/  CREATE TABLE dvdkopien (  dvdnr NUMBER(6) NOT NULL,  katalognr NUMBER(4) NOT NULL,  fnr VARCHAR2(3) NOT NULL,  CONSTRAINT dk\_pk PRIMARY KEY(dvdnr),  /\*CONSTRAINT dk\_fm\_fk FOREIGN KEY(katalognr)  REFERENCES filme,\*/  CONSTRAINT dk\_fl\_fk FOREIGN KEY(fnr)  REFERENCES filialen  );  CREATE TABLE registrierungen (  mnr VARCHAR2(4) NOT NULL,  fnr VARCHAR2(3) NOT NULL,  anr VARCHAR2(4) NULL,  datum DATE NOT NULL,  CONSTRAINT rg\_pk PRIMARY KEY(mnr, fnr),  CONSTRAINT rg\_mg\_fk FOREIGN KEY(mnr)  REFERENCES mitglieder,  CONSTRAINT rg\_fl\_fk FOREIGN KEY(fnr)  REFERENCES filialen,  CONSTRAINT rg\_an\_fk FOREIGN KEY(anr)  REFERENCES angestellte  );  CREATE TABLE ausleihen (  mnr VARCHAR2(4) NOT NULL,  dvdnr NUMBER(6) NOT NULL,  datum DATE NOT NULL,  rueckgabe DATE NULL,  CONSTRAINT al\_pk PRIMARY KEY(dvdnr,datum),  CONSTRAINT al\_mg\_fk FOREIGN KEY(mnr)  REFERENCES mitglieder,  CONSTRAINT al\_dk\_fk FOREIGN KEY(dvdnr)  REFERENCES dvdkopien  );  INSERT INTO mitglieder (mnr,name,wohnort,gdatum) VALUES ('M001','A. Huber','Basel',TO\_DATE('15.05.1978','dd.mm.yyyy'));  INSERT INTO mitglieder (mnr,name,wohnort,gdatum) VALUES ('M004','S. Baumann','Bern',TO\_DATE('21.03.1982','dd.mm.yyyy'));  INSERT INTO mitglieder (mnr,name,wohnort,gdatum) VALUES ('M005','U. Schoch','Basel',TO\_DATE('01.09.1975','dd.mm.yyyy'));  INSERT INTO mitglieder (mnr,name,wohnort,gdatum) VALUES ('M002','E. Müller','Bern',TO\_DATE('30.07.1985','dd.mm.yyyy'));  INSERT INTO mitglieder (mnr,name,wohnort,gdatum) VALUES ('M003','K. Buser','Riehen',TO\_DATE('13.04.1972','dd.mm.yyyy'));  INSERT INTO mitglieder (mnr,name,wohnort,gdatum) VALUES ('M006','E. Müller','Reinach BL',TO\_DATE('28.10.1980','dd.mm.yyyy'));  COMMIT;  INSERT INTO filialen (fnr,ort,plz) VALUES ('F1','Basel','4056');  INSERT INTO filialen (fnr,ort,plz) VALUES ('F3','Bern','3014');  INSERT INTO filialen (fnr,ort,plz) VALUES ('F2','Zürich','8050');  INSERT INTO filialen (fnr,ort,plz) VALUES ('F4','Basel','4002');  COMMIT;  INSERT INTO angestellte (anr,name,salaer) VALUES ('A01','Doris',4100);  INSERT INTO angestellte (anr,name,salaer) VALUES ('A02','Urs',3700);  INSERT INTO angestellte (anr,name,salaer) VALUES ('A03','Rolf',4100);  INSERT INTO angestellte (anr,name,salaer) VALUES ('A04','Eva',3900);  INSERT INTO angestellte (anr,name,salaer) VALUES ('A05','Urs',3900);  COMMIT;  /\*INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2028,'Casablanca',9,8.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (1245,'Ocean''s Eleven',12,9.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2239,'A Space Odyssee',12,7.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (1062,'Pulp Fiction',16,8.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2588,'The Pelican Brief',12,8.50);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (1672,'Erin Brockovich',9,8.90);  INSERT INTO filme (katalognr,titel,mindestalter,gebuehr) VALUES (2468,'Ratatouille',6,7.50);  COMMIT;\*/  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (199004,2028,'F1');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (468123,2028,'F2');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (269260,1245,'F1');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (183669,1245,'F3');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (329270,1245,'F4');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (178643,2239,'F2');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (389653,2239,'F4');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (158234,1062,'F3');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (139558,2468,'F2');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (469118,2468,'F2');  INSERT INTO dvdkopien (dvdnr,katalognr,fnr) VALUES (310094,2468,'F2');  COMMIT;  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M001','F1','A01',TO\_DATE('07.11.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M004','F1','A01',TO\_DATE('29.06.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M005','F1','A01',TO\_DATE('04.07.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M002','F3','A02',TO\_DATE('17.05.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M004','F3','A01',TO\_DATE('29.06.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M005','F3','A02',TO\_DATE('01.12.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M003','F1','A03',TO\_DATE('07.11.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M001','F2','A04',TO\_DATE('12.10.2013','dd.mm.yyyy'));  INSERT INTO registrierungen (mnr,fnr,anr,datum) VALUES ('M006','F4','A05',TO\_DATE('16.05.2013','dd.mm.yyyy'));  COMMIT;  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M002',158234,TO\_DATE('19.07.2013','dd.mm.yyyy'),TO\_DATE('21.07.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M004',158234,TO\_DATE('02.08.2013','dd.mm.yyyy'),TO\_DATE('04.08.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M003',269260,TO\_DATE('05.01.2014','dd.mm.yyyy'),NULL);  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M003',199004,TO\_DATE('05.01.2014','dd.mm.yyyy'),NULL);  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M001',310094,TO\_DATE('22.11.2013','dd.mm.yyyy'),TO\_DATE('27.11.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M001',468123,TO\_DATE('19.01.2014','dd.mm.yyyy'),NULL);  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M002',183669,TO\_DATE('30.11.2013','dd.mm.yyyy'),TO\_DATE('01.12.2013','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M004',183669,TO\_DATE('27.12.2013','dd.mm.yyyy'),TO\_DATE('03.01.2014','dd.mm.yyyy'));  INSERT INTO ausleihen (mnr,dvdnr,datum,rueckgabe) VALUES ('M005',183669,TO\_DATE('15.01.2014','dd.mm.yyyy'),NULL);  COMMIT; |

Damit wir auf dem Hauptserver die ausgelagerte Filmtabelle angenehmer ansprechen können, erstellen wir noch ein Synonym:

|  |
| --- |
| CREATE SYNONYM FILME FOR FILME@titania.janus.fhnw.ch;  SELECT \* FROM FILME; |

Analog dazu auf dem ausgelagerten Server ein Synonym für die DVDKopien:

|  |
| --- |
| CREATE SYNONYM DVDKOPIEN FOR DVDKOPIEN@ganymed.sirius.fhnw.ch;  SELECT \* FROM DVDKOPIEN; |

Nun können die Trigger auf den jeweiligen Server erstellt werden.

## Trigger EnsureFilmImmutability auf der ausgelagerten Datenbank erstellen

|  |
| --- |
| CREATE OR REPLACE TRIGGER EnsureFilmImmutability  BEFORE UPDATE ON FILME  FOR EACH ROW  BEGIN  IF :OLD.KATALOGNR != :NEW.KATALOGNR THEN  RAISE\_APPLICATION\_ERROR(-20005, 'The film catalog number is immutable to ensure referential integrity and thus can not be changed!');  END IF;  END;  / |

## Trigger EnsureFilmReferences auf der ausgelagerten Datenbank erstellen

|  |
| --- |
| CREATE OR REPLACE TRIGGER EnsureFilmReferences  BEFORE DELETE ON FILME  FOR EACH ROW  DECLARE  TMP\_DVDKOPIENCOUNT NUMBER;  BEGIN  SELECT COUNT(\*) INTO TMP\_DVDKOPIENCOUNT FROM DVDKOPIEN WHERE KATALOGNR = :OLD.KATALOGNR;  IF TMP\_DVDKOPIENCOUNT > 0 THEN  RAISE\_APPLICATION\_ERROR(-20005, 'The film can not be deleted because several DVD copy still rely on it (Delete them first)!');  END IF;  END;  / |

## Trigger EnsureDvdKopienReferences auf der Hauptdatenbank erstellen

|  |
| --- |
| CREATE OR REPLACE TRIGGER EnsureDvdKopienReferences  BEFORE INSERT OR UPDATE ON DVDKOPIEN  FOR EACH ROW  DECLARE  TMP\_FILMECOUNT NUMBER;  BEGIN  SELECT COUNT(\*) INTO TMP\_FILMECOUNT FROM FILME WHERE KATALOGNR = :NEW.KATALOGNR;  IF TMP\_FILMECOUNT = 0 THEN  RAISE\_APPLICATION\_ERROR(-20005, 'The DVD copy can not be inserted/updated because the film reference is not valid!');  END IF;  END;  / |

## Testszenario

Wir testen die Trigger zuerst auf ihr Schutzverhalten und danach auf ihr reguläres Verhalten:

Auf dem ausgelagerten Server:

|  |
| --- |
| DELETE FROM FILME WHERE KATALOGNR = 2028; /\* ORA-20005: The film can not be deleted because several DVD copy still rely on it (Delete them first)! \*/  DELETE FROM FILME WHERE KATALOGNR = 2588; /\* OK \*/  UPDATE FILME SET KATALOGNR = 42 WHERE KATALOGNR = 2468; /\* ORA-20005: The film catalog number is immutable to ensure refer-ential integrity and thus can not be changed! \*/  UPDATE FILME SET TITEL = 'Ratatouille Directors Cut' WHERE KATALOGNR = 2468; /\* OK \*/ |

Auf dem Hauptserver:

|  |
| --- |
| UPDATE DVDKOPIEN SET KATALOGNR = 2469 WHERE KATALOGNR = 2468; /\* ORA-20005: The DVD copy can not be inserted/updated because the film reference is not valid! \*/  UPDATE DVDKOPIEN SET KATALOGNR = 2468 WHERE KATALOGNR = 2468; /\* OK \*/  UPDATE DVDKOPIEN SET FNR = 'F3' WHERE DVDNR = 310094; /\* OK \*/ |

Die Queries können natürlich auch auf dem anderen Server ausgeführt werden, da beide symmetrisch gelinkt sind und bei einer Veränderung den Trigger auf dem anderen Server auslösen.

## Quellenangabe

* Quellenangabe analog zur 1. Übung
* PL/SQL Error Handling: <https://docs.oracle.com/cd/B10501_01/appdev.920/a96624/07_errs.htm>