

Datenbanken 2

Dokumentation zu Übung 3

Fabian Uhlmann
Diana Irmischer

26. Juni 2016

Aufgabe 2

Realisierung einer Min-Max-Skalierung auf einem Attribut A einer Relation. Skalierung selbst ist als Funktion implementiert, die mit den Parametern altes/neues Minimum/-Maximum versehen ist, einen Wert als Parameter entgegennimmt und den skalierten Wert zurückliefert. Folgende Funktion wird berechnet:

$$v' = \frac{v - \min_A}{\max_A - \min_A} \cdot (\text{newMax}_A - \text{newMin}_A) + \text{newMin}_A$$

— *DDL for Function MIN_MAX_SCALE*

```
CREATE OR REPLACE FUNCTION "MIN_MAX_SCALE"  
  (min_old NUMBER, min_new NUMBER, max_old NUMBER, max_new  
   NUMBER, v NUMBER)  
RETURN NUMBER  
IS  
BEGIN  
  RETURN (((v - min_old)/(max_old - min_old))*(max_new -  
    min_new)) + min_new;  
END;  
  
/
```

Die oben definierte Funktion wird von einer Prozedur aufgerufen werden, die für ein festes Attribut einer Relation zunächst den minimalen und den maximalen Wert der bisherigen Attributwerte ermittelt und dann alle Werte durch die skalierten Werte ersetzt.

— *DDL for Procedure MIN_MAX_CALCULATOR*

```
— Ergebnisse werden in neue Table eingetragen  
CREATE OR REPLACE PROCEDURE "MIN_MAX_CALCULATOR"  
  (min_new NUMBER, max_new NUMBER)  
IS  
  min_old number;  
BEGIN  
  SELECT MIN(ZAHLEN) INTO min_old FROM NUMBERS;  
  INSERT INTO NUMBERS.RESULT(  
    SELECT MIN_MAX_SCALE(min_old, min_new, (SELECT MAX(ZAHLEN)  
      FROM NUMBERS), max_new, ZAHLEN)  
    FROM NUMBERS);  
END;  
  
/
```

```

— Alternative: Update in gleicher Table
CREATE OR REPLACE PROCEDURE "MIN_MAX_CALCULATOR"
  (min_new NUMBER, max_new NUMBER)
IS
  min_old number;
BEGIN
    SELECT MIN(ZAHLEN) INTO min_old FROM NUMBERS;
    UPDATE NUMBERS SET ZAHLEN = MIN_MAX_SCALE(min_old , min_new , (
      SELECT MAX(ZAHLEN) FROM NUMBERS) , max_new , ZAHLEN);
END;

```

Beispiel:

```

—> Result:
EXECUTE min_max_calculator(0,10);

SELECT * FROM NUMBERS ORDER BY ZAHLEN ASC;

```

ZAHLEN
5
10
20
25
42
50
53
100
120
142
242
250
342
350
420

Aufgabe 3

Aus zwei verschiedenen Quellen werden zwei Tabellen Angestellte und Arbeiter in den Arbeitsbereich eines Data Warehouse Systems geladen, die in eine Tabelle Personal integriert werden sollen.

Die Tabellen besitzen die folgenden Attribute:

Angestellte:	Name (String, Vorname und Nachname durch Blank getrennt; oder Nachname, Vorname) Geburtsdatum (Format: Datum, YY/MM/DD) Berufsbezeichnung Monatsgehalt Geschlecht: männlich bzw. weiblich Angestelltennr (Primärschlüssel)
Arbeiter:	Name Vorname Geburtsmonat (Format: String MM.YY) Stundenlohn
Personal:	Personalnr. (Primärschlüssel) Name Vorname Alter Geschlecht: (0 (unbekannt), 1 (weiblich), 2 (männlich)) Berufscod Jahreseinkommen

Die Tabelle Personal enthält einen neuen (generierten) Schlüssel.

— *DDL for Sequence PNR_SEQUENCE*

CREATE SEQUENCE "PNR_SEQUENCE" ;

/

— *DDL for Table ANGESTELLTE*

CREATE TABLE "ANGESTELLTE"
(
 "A_NR" **NUMBER**,
 "A_NAME" **VARCHAR2**(50) ,
 "A_GEBURTSDATUM" **DATE**,
 "A_BERUFSBEZEICHNUNG" **VARCHAR2**(60) ,
 "A_MONATSGEHALT" **NUMBER**,
 "A_GESCHLECHT" **VARCHAR2**(10) ,
 PRIMARY KEY ("A_NR")

```
) ;  
  
/
```

— *DDL for Table ARBEITER*

```
CREATE TABLE "ARBEITER"  
(  
    "A_NAME" VARCHAR2(30) ,  
    "A_VORNAME" VARCHAR2(30) ,  
    "A_GEBURTSMONAT" VARCHAR2(5) ,  
    "A_STUNDENLOHN" NUMBER,  
    PRIMARY KEY ("A_NAME" , "A_VORNAME" )  
) ;  
  
/
```

Für die Berufscodes ist eine Code-Tabelle zu definieren, die Berufsbezeichnungen (einschließlich der Bezeichnung Arbeiter) einen Code zuordnet.

— *DDL for Table BERUFE*

```
CREATE TABLE "BERUFE"  
(  
    "B_CODE" NUMBER,  
    "B_TYPE" VARCHAR2(30) ,  
    PRIMARY KEY ("B_CODE" )  
) ;  
  
/
```

Für die „Geschlechtsbestimmung“ der Arbeiter verwenden Sie bitte eine Hilfstabelle, die Vornamen jeweils ein oder mehrere Geschlechter zuordnen. Ergibt der Abgleich mit dieser Tabelle kein eindeutiges Ergebnis wird der Eintrag in der Zieltabelle auf 0 (unbekannt) gesetzt.

— *DDL for Table GESCHLECHTER*

```
CREATE TABLE "GESCHLECHTER"  
(  
    "G_NAME" VARCHAR2(15) ,  
    "G_CODE" NUMBER,  
    PRIMARY KEY ("G_NAME" )  
) ;
```

/

— *DDL for Table PERSONAL*

```
CREATE TABLE "PERSONAL"
(
    "P_NR" NUMBER,
    "P_NAME" VARCHAR2(30) ,
    "P_VORNAME" VARCHAR2(30) ,
    "P_ALTER" NUMBER,
    "P_GESCHLECHT" NUMBER,
    "P_BERUFSCODE" NUMBER,
    "P_JAHRESEINKOMMEN" NUMBER,
    PRIMARY KEY ("P_NR") ,
    FOREIGN KEY ("P_BERUFSCODE") REFERENCES "BERUFE" ("
        B_CODE" )
);
```

/

Für die Beziehung zwischen diesem und den Quelltabellen soll eine Zuordnungstabelle verwaltet werden.

— *DDL for Table ZUORDNUNG*

```
CREATE TABLE "ZUORDNUNG"
(
    "Z_NR" NUMBER,
    "Z_TABLE_OLD" VARCHAR2(30) ,
    "Z_KEY_OLD" VARCHAR2(60) ,
    PRIMARY KEY ("Z_NR") ,
    FOREIGN KEY ("Z_NR") REFERENCES "PERSONAL" ("P_NR" )
);
```

/

Programm, das die Integration ausführt und sowohl für den initialen Ladevorgang der Tabelle Personal als auch für deren Fortschreibung geeignet ist (d.h. Angestellte und Arbeiter enthalten jeweils nur neue bzw. geänderte Datensätze).

— *DDL for Function GETAGE_DATE*

```
CREATE OR REPLACE FUNCTION "GETAGE_DATE"
```

```

    (birthdate Date)
RETURN VARCHAR2
IS
BEGIN
    RETURN Trunc((months_between(sysdate , birthdate) /12),0);
END;

/

```

```

— DDL for Function GETAGE_STRING

```

```

CREATE OR REPLACE FUNCTION "GETAGE_STRING"
    (birthdate VARCHAR)
RETURN VARCHAR2
age DATE;
BEGIN
    — SELECT EXTRACT(MONTH FROM SYSDATE) FROM DUAL;
    SELECT TO_DATE(birthdate , 'MM.RR') INTO age FROM DUAL;
    RETURN Trunc((months_between(sysdate , age) /12),0);
END;

/

```

```

— DDL for Function GETFIRSTNAME

```

```

CREATE OR REPLACE FUNCTION "GETFIRSTNAME"
    (fname VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
    RETURN SUBSTR(fname,0 , instr (fname , ' ' )-1);
END;

/

```

```

— DDL for Function GETGENDERCODE

```

```

CREATE OR REPLACE FUNCTION "GETGENDERCODE"
    (gender VARCHAR2 , firstname VARCHAR2)
RETURN NUMBER

```

```

CURSOR CGCODE IS
    SELECT G.CODE
    FROM Geschlechter
    WHERE G.NAME = firstname;
gendercode NUMBER;
tmp NUMBER;
BEGIN
    CASE gender
        WHEN 'maennlich' THEN gendercode := 2;
        WHEN 'weiblich' THEN gendercode := 1;
        ELSE gendercode := 0;
    END CASE;

    OPEN CGCODE;
    FETCH CGCODE INTO tmp;
    IF CGCODE%NOTFOUND THEN
        INSERT INTO GESCHLECHTER (G.NAME, G.CODE) VALUES (firstname
            , gendercode);
    ELSE
        IF gendercode != 0 AND gendercode != tmp THEN
            UPDATE GESCHLECHTER SET G.CODE = gendercode WHERE G.NAME
                = firstname;
        ELSE gendercode := tmp;
        END IF;
    END IF
    RETURN gendercode;
END;

/

```

— *DDL for Function GETLASTNAME*

```

CREATE OR REPLACE FUNCTION "DBST47"."GETLASTNAME"
(lname VARCHAR2)
RETURN VARCHAR2
IS
BEGIN
    RETURN SUBSTR(lname, INSTR(lname, ' ')+1);
END;

/

```

— *DDL for Function GETJOBBCODE*

```
CREATE OR REPLACE FUNCTION "GETJOBBCODE"  
  (jobname VARCHAR2)  
RETURN NUMBER  
IS  
CURSOR CBCODE IS  
    SELECT B.CODE  
    FROM Berufe  
    WHERE B.TYPE = jobname;  
jobcode NUMBER;  
BEGIN  
  OPEN CBCODE;  
  FETCH CBCODE into jobcode;  
  IF CBCODE%NOTFOUND THEN  
    SELECT max(B.CODE) INTO jobcode FROM BERUFE;  
    IF jobcode IS NULL THEN jobcode := 0;  
    ELSE jobcode := jobcode + 1;  
    END IF;  
    INSERT INTO BERUFE (B.CODE, B.TYPE) VALUES (jobcode ,jobname  
      );  
  END IF;  
  RETURN jobcode;  
END;  
  
/
```

— *DDL for Function GETMONEY*

```
CREATE OR REPLACE FUNCTION "GETMONEY"  
  (monthmoney NUMBER)  
RETURN NUMBER  
IS  
BEGIN  
  RETURN (monthmoney * 12);  
END;  
  
/
```

— *DDL for Procedure TRANSFORMATION_ANGESTELLTE*

```

CREATE OR REPLACE PROCEDURE "TRANSFORMATION_ANGESTELLTE"
IS
  a_nr NUMBER;
  p_nr NUMBER;
  p_name VARCHAR2(30);
  p_vorname VARCHAR2(30);
  p_age DATE;
  p_geschlecht VARCHAR2(10);
  p_job VARCHAR(50);
  p_money NUMBER;
  CURSOR CANGST IS
    SELECT A_Nr, A_Name, A_Geburtsdatum,
           A_Berufsbezeichnung, A_Monatsgehalt, A_Geschlecht
    FROM Angestellte;
BEGIN
  OPEN CANGST;
  LOOP
    FETCH CANGST INTO a_nr, p_name, p_age, p_job, p_money,
                      p_geschlecht;
    EXIT WHEN CANGST%NOTFOUND;
    SELECT pnr_sequence.nextval INTO p_nr FROM DUAL;
    SELECT GETFIRSTNAME(p_name) INTO p_vorname FROM DUAL;
    INSERT INTO PERSONAL(p_nr, p_name, p_vorname, p_alter,
                        p_geschlecht, p_berufscod, p_jahreseinkommen) VALUES
      (p_nr, GETLASTNAME(p_name), p_vorname, GETAGEDATE(
        p_age), GETGENDERCODE(p_geschlecht, p_vorname),
        GETJOBCODE(p_job), GETMONEY(p_money));
    INSERT INTO ZUORDNUNG (Z_NR, Z_TABLE_OLD, Z_KEY_OLD)
      VALUES (p_nr, 'Angestellter', TO_CHAR(a_nr, '
        99999999'));
  END LOOP;
  CLOSE CANGST;
END;

/

```

— *DDL for Procedure TRANSFORMATION_ARBEITER*

```

CREATE OR REPLACE PROCEDURE "TRANSFORMATION_ARBEITER"
IS
  p_nr NUMBER;
  p_name VARCHAR2(30);

```

```

p_vorname VARCHAR2(30);
p_age VARCHAR2(5);
p_geschlecht VARCHAR2(10);
p_job VARCHAR(50);
p_money NUMBER;
arb_nr VARCHAR2(60);
CURSOR CARB IS
    SELECT A_Name, A_Vorname, A_Geburtsmonat, A_Stundenlohn
    FROM Arbeiter;
BEGIN
    OPEN CARB;
    LOOP
        FETCH CARB INTO p_name, p_vorname, p_age, p_money;
        EXIT WHEN CARB%NOTFOUND;
        SELECT pnr_sequence.nextval INTO p_nr FROM DUAL;
        INSERT INTO PERSONAL(p_nr, p_name, p_vorname, p_alter,
            p_geschlecht, p_berufscore, p_jahreseinkommen) VALUES
            (p_nr, p_name, p_vorname, GETAGE.STRING(p_age),
            GETGENDERCODE('unbekannt', p_vorname), GETJOB.CODE('
            Arbeiter'), GETMONEY(p_money*4*40));
        arb_nr := CONCAT(CONCAT(p_name, ', '), p_vorname);
        INSERT INTO ZUORDNUNG (Z_NR, Z.TABLE_OLD, Z.KEY_OLD)
            VALUES (p_nr, 'Arbeiter', arb_nr);
    END LOOP;
    CLOSE CARB;
END;

/

```

— *DDL for Trigger UPDATE_ARBEITER*

```

CREATE OR REPLACE TRIGGER UPDATE_ARBEITER
AFTER
    INSERT OR
    UPDATE OR
    DELETE
ON Arbeiter
FOR EACH ROW
DECLARE
    z_nr NUMBER;
    arb_nr VARCHAR2(60);
    p_nr NUMBER;
BEGIN

```

```

IF INSERTING THEN
SELECT pnr_sequence.nextval INTO p_nr FROM DUAL;
INSERT INTO PERSONAL(p_nr,p_name,p_vorname,p_alter ,
    p_geschlecht,p_berufscod,e,p_jahreseinkommen) VALUES (
    p_nr,:NEW.A.NAME,:NEW.A.VORNAME,GETAGE_STRING(:NEW.
    A.GEBURTSMONAT),GETGENDERCODE('unbekannt',:NEW.A.VORNAME
    ),GETJOB_CODE('Arbeiter'),GETMONEY(:NEW.A.STUNDENLOHN
    *4*40));
arb_nr := CONCAT(CONCAT(:NEW.A.NAME,','),:NEW.A.VORNAME);
INSERT INTO ZUORDNUNG (Z_NR, Z_TABLE_OLD, Z_KEY_OLD) VALUES
    (p_nr, 'Arbeiter', arb_nr);

ELSIF UPDATING THEN
IF :OLD.A.NAME != :NEW.A.NAME THEN
SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
    Z_KEY_OLD = CONCAT(CONCAT(:OLD.A.NAME,','),:OLD.
    A.VORNAME); /* --> dieses SELECT ggf. auslagern und
    direkt nach ELSIF UPDATING, da es wird in allen IFs
    von Updating benoetigt */
UPDATE PERSONAL p SET p.P.NAME = :NEW.A.NAME WHERE p.P_NR
    = z_nr;
SELECT p.P_NR INTO p_nr FROM PERSONAL p WHERE p.P.NAME =
    :NEW.A.NAME AND p.P.VORNAME = :OLD.A.VORNAME; /*
    notwendig, da sonst die Komplette Spalte Z_KEY_OLD in
    Table ZUORDNUNG mit geaendertem Namen ueberschrieben
    wird*/
arb_nr := CONCAT(CONCAT(:NEW.A.NAME,','),:OLD.A.VORNAME);
UPDATE ZUORDNUNG z SET z.Z_KEY_OLD = arb_nr WHERE z.z_nr
    = p_nr;
END IF;

IF :OLD.A.VORNAME != :NEW.A.VORNAME THEN
SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
    Z_KEY_OLD = CONCAT(CONCAT(:OLD.A.NAME,','),:OLD.
    A.VORNAME);
/* DBMS_OUTPUT.PUT_LINE(z_nr); -- Fuer Debugging =
    Ausgabe auf DBMS-Console */
UPDATE PERSONAL p SET p.P.VORNAME = :NEW.A.VORNAME WHERE
    p.P_NR = z_nr;
SELECT p.P_NR INTO p_nr FROM PERSONAL p WHERE p.P.NAME =
    :OLD.A.NAME AND p.P.VORNAME = :NEW.A.VORNAME; /*
    notwendig, da sonst die Komplette Spalte Z_KEY_OLD in
    Table ZUORDNUNG mit geaendertem Namen ueberschrieben
    wird*/

```

```

        arb_nr := CONCAT(CONCAT(:OLD.A.NAME, ', ' ), :NEW.A.VORNAME);
        UPDATE ZUORDNUNG zg SET zg.Z_KEY_OLD = arb_nr WHERE zg.
            z_nr = p_nr;
    END IF;

    IF :OLD.A.GEBURTSMONAT != :NEW.A.GEBURTSMONAT THEN
        SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
            Z_KEY_OLD = CONCAT(CONCAT(:OLD.A.NAME, ', ' ), :OLD.
            A.VORNAME);
        UPDATE PERSONAL p SET p.p_ALTER = GETAGE_STRING(:NEW.
            A.GEBURTSMONAT) WHERE p.P_NR = z_nr;
    END IF;

    IF :OLD.A.STUNDENLOHN != :NEW.A.STUNDENLOHN THEN
        SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
            Z_KEY_OLD = CONCAT(CONCAT(:OLD.A.NAME, ', ' ), :OLD.
            A.VORNAME);
        UPDATE PERSONAL p SET p.P.JAHRESEINKOMMEN = GETMONEY(:NEW
            .A.STUNDENLOHN * 40 *4) WHERE p.P_NR = z_nr;
    END IF;

    ELSIF DELETING THEN
        SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
            Z_KEY_OLD = CONCAT(CONCAT(:OLD.A.NAME, ', ' ), :OLD.
            A.VORNAME);
        DELETE FROM ZUORDNUNG z WHERE z.Z_NR = z_nr AND z.
            Z_KEY_OLD = CONCAT(CONCAT(:OLD.A.NAME, ', ' ), :OLD.
            A.VORNAME);
        DELETE FROM PERSONAL p WHERE p.P_NR = z_nr;

    ELSE NULL;
    END IF;
END;

/

```

— *DDL for Trigger UPDATE_ANGESTELLTE*

```

CREATE OR REPLACE TRIGGER UPDATE_ANGESTELLTE
AFTER
    INSERT OR
    UPDATE OR
    DELETE

```

```

ON ANGESTELLTE
FOR EACH ROW
DECLARE
    z_nr NUMBER;
    arb_nr VARCHAR(60);
    p_nr NUMBER;
    p_vorname VARCHAR2(30);
BEGIN
    IF INSERTING THEN
        SELECT pnr_sequence.nextval INTO p_nr FROM DUAL;
        SELECT GETFIRSTNAME(:NEW.A.NAME) INTO p_vorname FROM DUAL;
        INSERT INTO PERSONAL(p_nr , p_name , p_vorname , p_alter ,
            p_geschlecht , p_berufstcode , p_jahreseinkommen) VALUES (
            p_nr , GETLASTNAME(:NEW.A.NAME) , p_vorname , GETAGE.DATE(:NEW
            .A.GEBURTSDATUM) , GETGENDERCODE(:NEW.A.GESCHLECHT,
            p_vorname) , GETJOBCODE(:NEW.A.BERUFSBEZEICHNUNG) , GETMONEY
            (:NEW.A.MONATSGEHALT));
        INSERT INTO ZUORDNUNG (Z.NR, Z.TABLE_OLD, Z.KEY_OLD) VALUES
            (p_nr , 'Angestellter' , TO_CHAR(:NEW.A.NR, '99999999'));

    ELSIF UPDATING THEN

        SELECT z.Z.NR INTO z_nr FROM ZUORDNUNG z WHERE z.Z.KEY_OLD
            = TO_CHAR(:OLD.A.NR, '99999999');

        IF :OLD.A.NR != :NEW.A.NR THEN
            SELECT p.P.NR INTO p_nr FROM PERSONAL p WHERE p.P.NR =
                z_nr AND p.P.NAME = GETLASTNAME(:OLD.A.NAME) AND p.
                P.VORNAME = GETFIRSTNAME(:OLD.A.NAME);
            UPDATE ZUORDNUNG z SET z.Z.KEY_OLD = TO_CHAR(:NEW.A.NR, '
                99999999') WHERE z.Z.NR = p_nr;
        END IF;

        IF :OLD.A.NAME != :NEW.A.NAME THEN
            UPDATE PERSONAL p SET p.P.NAME = GETLASTNAME(:NEW.A.NAME)
                , p.P.VORNAME = GETFIRSTNAME(:NEW.A.NAME) WHERE p.P.NR
                = z_nr;
        END IF;

        IF :OLD.A.GEBURTSDATUM != :NEW.A.GEBURTSDATUM THEN
            UPDATE PERSONAL p SET p.p.ALTER = GETAGE.DATE(:NEW.
                A.GEBURTSDATUM) WHERE p.P.NR = z_nr;
        END IF;

```

```

IF :OLD.A.BERUFSBEZEICHNUNG != :NEW.A.BERUFSBEZEICHNUNG
  THEN
    UPDATE PERSONAL p SET p.p_BERUFSCODE = GETJOBCODE(:NEW.
      A.BERUFSBEZEICHNUNG) WHERE p.P_NR = z_nr;
END IF;

IF :OLD.A.MONATSGEHALT != :NEW.A.MONATSGEHALT THEN
  UPDATE PERSONAL p SET p.P_JAHRESEINKOMMEN = GETMONEY(:NEW
    .A.MONATSGEHALT) WHERE p.P_NR = z_nr;
END IF;

IF :OLD.A.GESCHLECHT != :NEW.A.GESCHLECHT THEN
  UPDATE PERSONAL p SET p.P_geschlecht = GETGENDERCODE(:NEW
    .A.GESCHLECHT,GETFIRSTNAME(:OLD.A.NAME)) WHERE p.P_NR
    = z_nr;
END IF;

ELSIF DELETING THEN
  SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
    Z_KEY_OLD = TO_CHAR(:OLD.A.NR, '99999999');
  DELETE FROM ZUORDNUNG z WHERE z.Z_NR = z_nr AND z.
    Z_KEY_OLD = TO_CHAR(:OLD.A.NR, '99999999');
  DELETE FROM PERSONAL p WHERE p.P_NR = z_nr;

  ELSE NULL;
END IF;
END;

/

```

— *Inserts in Table ANGESTELLTE*

```

DELETE FROM ANGESTELLTE;
Insert into ANGESTELLTE (A_NR,A_NAME,A_GEBURTSDATUM,
  A_BERUFSBEZEICHNUNG,A_MONATSGEHALT,A_GESCHLECHT) values ('1'
  , 'Fabian_Uhlmann',to_date('03.11.88','DD.MM.RR'),'
  Informatiker','2000','maennlich');
Insert into ANGESTELLTE (A_NR,A_NAME,A_GEBURTSDATUM,
  A_BERUFSBEZEICHNUNG,A_MONATSGEHALT,A_GESCHLECHT) values ('2'
  , 'Diana_Irmscher',to_date('01.01.90','DD.MM.RR'),'
  Informatiker','2001','weiblich');
Insert into ANGESTELLTE (A_NR,A_NAME,A_GEBURTSDATUM,
  A_BERUFSBEZEICHNUNG,A_MONATSGEHALT,A_GESCHLECHT) values ('3'

```

```

, 'Alexandra_Vogel', to_date('01.10.92', 'DD.MM.RR'), '
Informatiker', '9999', 'weiblich');
Insert into ANGESTELLTE (A_NR, A_NAME, A_GEBURTSDATUM,
A_BERUFSBEZEICHNUNG, A_MONATSGEHALT, A_GESCHLECHT) values ('4',
, 'Alexander_Boxhorn', to_date('27.07.82', 'DD.MM.RR'), '
Logistiker', '1375', 'maennlich');

/

```

— *Inserts in Table ARBEITER*

```

DELETE FROM ARBEITER;
Insert into ARBEITER (A_NAME, A_VORNAME, A_GEBURTSMONAT,
ASTUNDENLOHN) values ('Meister', 'Bob', '11.88', 20);
Insert into ARBEITER (A_NAME, A_VORNAME, A_GEBURTSMONAT,
ASTUNDENLOHN) values ('Mueller', 'Sarah', '07.95', 10);
Insert into ARBEITER (A_NAME, A_VORNAME, A_GEBURTSMONAT,
ASTUNDENLOHN) values ('Bach', 'Hans', '01.75', 5);
Insert into ARBEITER (A_NAME, A_VORNAME, A_GEBURTSMONAT,
ASTUNDENLOHN) values ('Heinz', 'Karl', '11.88', 8.5);

/

```

— *Inserts in Table GESCHLECHTER*

```

DELETE FROM GESCHLECHTER;
Insert into GESCHLECHTER (G_NAME, G_CODE) values ('Alexandra', '1
');
Insert into GESCHLECHTER (G_NAME, G_CODE) values ('Fabian', '2');

/

```

Nun werden Änderungen in die Tabellen eingetragen.

— *Testcases*

```

DELETE FROM ZUORDNUNG;
DELETE FROM PERSONAL;
/*1*/ EXECUTE TRANSFORMATION_ARBEITER;
/*2*/ EXECUTE TRANSFORMATION_ANGESTELLTE;

```



```

/*3*/ Insert into ARBEITER (A_NAME,A_VORNAME,A_GEBURTSMONAT,
    A_STUNDENLOHN) values ( 'Kapitaen', 'Blaubaer', '05.44', 33);
/*4*/ UPDATE ARBEITER SET A_NAME = 'Meyer' WHERE A_NAME = '
    Meister'; /* !!! Table Zuordnung darf/sollte nur im einen
    Datensatz das Attribut Z_KEY_OLD updaten */
/*5*/ UPDATE ARBEITER SET A_VORNAME = 'Hans-Joachim' WHERE
    A_NAME = 'Bach'; /* !!! Table Zuordnung darf/sollte nur im
    einen Datensatz das Attribut Z_KEY_OLD updaten */
/*6*/ UPDATE ARBEITER SET A_GEBURTSMONAT = '01.01' WHERE A_NAME
    = 'Heinz';
/*7*/ UPDATE ARBEITER SET A_STUNDENLOHN = 3.5 WHERE A_NAME = '
    Mueller';
/*8*/ DELETE FROM ARBEITER WHERE A_NAME = 'Kapitaen'; /* In
    Table Zuordnung darf/soll nur 1 Datensatz entfernt werden */

/*9*/ Insert into ANGESTELLTE (A_NR,A_NAME,A_GEBURTSDATUM,
    A_BERUFSBEZEICHNUNG,A_MONATSGEHALT,A_GESCHLECHT) values ( '5'
    , 'Max_Mustermann', to_date( '10.03.67', 'DD.MM.RR'), 'BWL', '850'
    , 'maennlich' );
/*10*/ UPDATE ANGESTELLTE SET A_NR = '10' WHERE A_NAME = 'Diana
    _Irmscher'; /* !!! Table Zuordnung darf/sollte nur im einen
    Datensatz das Attribut Z_KEY_OLD updaten */
/*11*/ UPDATE ANGESTELLTE SET A_NAME = 'Fabius_Uhlmex' WHERE
    A_NAME = 'Fabian_Uhlmann';
/*12*/ UPDATE ANGESTELLTE SET A_GEBURTSDATUM = 03.10.1955 WHERE
    A_NAME = 'Alexandra_Vogel';
/*13*/ UPDATE ANGESTELLTE SET A_BERUFSBEZEICHNUNG = 'Facility_
    Management' WHERE A_NAME = 'Alexander_Boxhorn';
/*14*/ UPDATE ANGESTELLTE SET A_MONATSGEHALT = 777 WHERE A_NAME
    = 'Alexander_Boxhorn';
/*15*/ UPDATE ANGESTELLTE SET A_GESCHLECHT = 'weiblich' WHERE
    A_NAME = 'Max_Mustermann';
/*16*/ DELETE FROM ANGESTELLTE WHERE A_NAME = 'Max_Mustermann';
    /* !!! Table Zuordnung darf/sollte nur im einen Datensatz
    das Attribut Z_KEY_OLD updaten */

```