Hochschule München, Fakultät 07, SoSe 2016

Datenbanken 2

Dokumentation zu Übung 3

Fabian Uhlmann
Diana Irmscher

27. 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:
$v' = \frac{v - min_A}{max_A - min_A} \cdot (newMax_A - newMin_A) + newMin_A$

```
-------------------------------------------------------------
--  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))
	Berufscode
	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 mehre 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
```

```
IS
CURSOR CGCODE IS
        SELECT G_CODE
        FROM Geschlechter
  WHERE G_NAME = firstname;
gendercode NUMBER;
tmp NUMBER;
x VARCHAR2(10);
y VARCHAR2(10);
BEGIN

  x := 'maennlich';
  y := 'weiblich';

  gendercode := 0;
  IF gender like x THEN gendercode := 2;
  END IF;
  IF gender like y THEN gendercode := 1;
  END IF;
  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;
  CLOSE CGCODE;
  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 GETJOBCODE
--------------------------------------------------------
  CREATE OR REPLACE FUNCTION "GETJOBCODE"
  (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_berufscode,p_jahreseinkommen) VALUES
            (p_nr,GETLASTNAME(p_name),p_vorname,GETAGE_DATE(
            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_berufscode,p_jahreseinkommen) VALUES
            (p_nr,p_name,p_vorname,GETAGE_STRING(p_age),
            GETGENDERCODE('unbekannt',p_vorname),GETJOBCODE('
            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 Procedure TRANSFORMATION_ANGESTELLTE
--------------------------------------------------------

  create or replace PROCEDURE "TRANSFORMATION_ANGESTELLTE"
IS
a_nr NUMBER;
p_nr NUMBER;
z_nr NUMBER;
```

```
p_name VARCHAR2(30);
p_vorname VARCHAR2(30);
p_age DATE;
p_geschlecht VARCHAR2(10);
p_job VARCHAR(50);
p_money NUMBER;
DATA_FOUND exception;
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;
    begin
      SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
          Z_KEY_OLD = TO_CHAR(a_nr, '99999999');
       raise DATA_FOUND;
    exception
      when DATA_FOUND then
        SELECT GETFIRSTNAME(p_name) INTO p_vorname FROM DUAL;
        UPDATE PERSONAL p SET p.p_name = GETLASTNAME(p_name), p
            .p_vorname = p_vorname, p.p_alter = GETAGE_DATE(
            p_age), p.p_geschlecht = GETGENDERCODE(p_geschlecht,
            p_vorname), p.p_berufscode = GETJOBCODE(p_job), p.
            p_jahreseinkommen = GETMONEY(p_money) WHERE p.P_NR =
             z_nr;
      when NO_DATA_FOUND then
        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_berufscode,p_jahreseinkommen) VALUES
            (p_nr,GETLASTNAME(p_name),p_vorname,GETAGE_DATE(
            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;
  END LOOP;
  CLOSE CANGST;
```

```
END;

/
```

```
--------------------------------------------------------
--  DDL for Procedure TRANSFORMATION_ARBEITER
--------------------------------------------------------
  create or replace PROCEDURE "TRANSFORMATION_ARBEITER"
IS
p_nr NUMBER;
z_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);
DATA_FOUND exception;
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;
    arb_nr := CONCAT(CONCAT(p_name,','),p_vorname);
    begin
      SELECT z.Z_NR INTO z_nr FROM ZUORDNUNG z WHERE z.
          Z_KEY_OLD = arb_nr;
      raise DATA_FOUND;
    exception
      when DATA_FOUND then
        UPDATE PERSONAL p SET p.p_alter = GETAGE_STRING(p_age),
            p.p_geschlecht = GETGENDERCODE('unbekannt',
            p_vorname), p.p_berufscode = GETJOBCODE('Arbeiter'),
            p.p_jahreseinkommen = GETMONEY(p_money*4*40) WHERE
            p.P_NR = z_nr;
      when NO_DATA_FOUND then
        SELECT pnr_sequence.nextval INTO p_nr FROM DUAL;
        INSERT INTO PERSONAL(p_nr,p_name,p_vorname,p_alter,
            p_geschlecht,p_berufscode,p_jahreseinkommen) VALUES
            (p_nr,p_name,p_vorname,GETAGE_STRING(p_age),
```

```
            GETGENDERCODE('unbekannt',p_vorname),GETJOBCODE('
            Arbeiter'),GETMONEY(p_money*4*40));
        INSERT INTO ZUORDNUNG (Z_NR, Z_TABLE_OLD, Z_KEY_OLD)
            VALUES (p_nr, 'Arbeiter', arb_nr);
    end;
  END LOOP;
  CLOSE CARB;
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,
   A_STUNDENLOHN) values ('Meister','Bob','11.88',20);
Insert into ARBEITER (A_NAME,A_VORNAME,A_GEBURTSMONAT,
   A_STUNDENLOHN) values ('Mueller','Sarah','07.95',10);
Insert into ARBEITER (A_NAME,A_VORNAME,A_GEBURTSMONAT,
   A_STUNDENLOHN) values ('Bach','Hans','01.75',5);
```

```
Insert into ARBEITER (A_NAME,A_VORNAME,A_GEBURTSMONAT,
   A_STUNDENLOHN) 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');

/
```