Datenbanksysteme II
SS 2017 – Übungsblatt 4

Teillösung

Universität Leipzig, Institut für Informatik

Abteilung Datenbanken

Prof. Dr. E. Rahm,

V. Christen, M. Franke

Aufgabe 1 – NF2→SQL:1999

Abteilung								
Abtnr	MgrNr	Budget	Ausstattung		Mitarbeiter			
			Anzahl	Typ	MaNr	Funktion	Projekt	
							PNr	PName

CREATE TYPE PersonT (
MaNr INT,
Vorname **VARCHAR** (40),
Nachname **VARCHAR** (40)) **FINAL**;

CREATE TABLE Manager-Table **OF** PersonT
(**REF IS** oid **SYSTEM GENERATED**,
PRIMARY KEY MaNr);

CREATE TABLE Mitarbeiter-Table **OF** PersonT
(**REF IS** oid **SYSTEM GENERATED**,
PRIMARY KEY MaNr);

Aufgabe 1 – NF2→SQL:1999

Abteilung								
Abtnr	MgrNr	Budget	Ausstattung		Mitarbeiter			
			Anzahl	Typ	MaNr	Funktion	Projekt	
							PNr	PName

CREATE TABLE Abteilung (

AbtNr **INT NOT NULL PRIMARY KEY**,

MgrNr **REF**(PersonT) **SCOPE** Manager-Table **NOT NULL**,

Budget **INT**,

Ausstattung **ROW**(Anzahl **INT**, Typ **VARCHAR**(60)) **MULTISET**,

Mitarbeiter **ROW**(MaNr **REF**(PersonT) **SCOPE** Mitarbeiter-Table,
Funktion **VARCHAR**(50),
Projekt **ROW**(PNr **INT**, PName **VARCHAR**(60)) **MULTISET**
) **MULTISET**

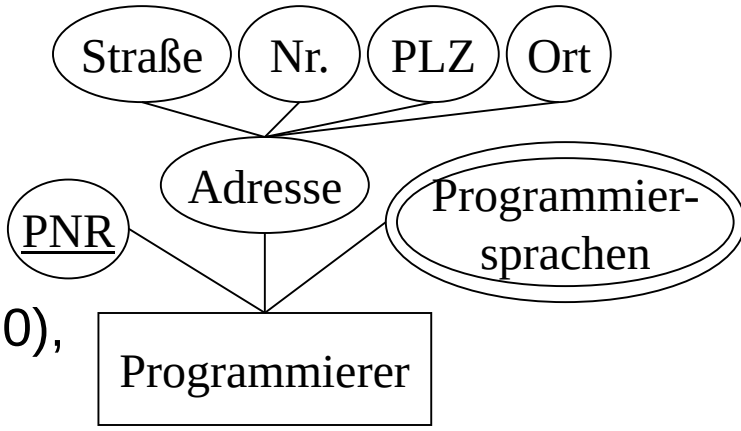
)

Aufgabe 2 – SQL:1999 DISTINCT-Typ, ROW-Typ

```
CREATE TYPE ProgSpracheT AS VARCHAR FINAL;

CREATE TABLE  Programmierer (
    PNR             INTEGER,
    Adresse         ROW (Straße   VARCHAR (50),
                         Nr       INTEGER,
                         PLZ      INTEGER,
                         Ort      VARCHAR (50)),
    ProgSprachen  ProgSpracheT ARRAY[10]
)

SELECT  PNR
FROM    Programmierer P
WHERE   P.Adresse.Ort = „Leipzig" AND ProgSpracheT(‘Java‘) IN (
            SELECT * FROM UNNEST(P.ProgSprachen)
)
```



Aufgabe 4 – Anfragen Polyeder

Herkömmliche Modellierung

```
CREATE TABLE Polyeder
    (polyid:      INTEGER,
    anzflächen: INTEGER,
    PRIMARY KEY (polyid));

CREATE TABLE Fläche
    (fid:          INTEGER,
    anzkanten:  INTEGER,
    pref:          INTEGER,
    PRIMARY KEY (fid),
    FOREIGN KEY (pref)
REFERENCES Polyeder);

CREATE TABLE Kante
    (kid:          INTEGER,
    ktyp:          CHAR(5),
    PRIMARY KEY (kid));

CREATE TABLE Punkt
    (pid:          INTEGER,
    x, y, z:       FLOAT,
    PRIMARY KEY (pid));

CREATE TABLE FK_Rel
    (fid:    INTEGER,
    kid:    INTEGER,
    PRIMARY KEY  (fid, kid),
    FOREIGN KEY (fid)
      REFERENCES Fläche,
    FOREIGN KEY (kid)
      REFERENCES Kante);

CREATE TABLE KP_Rel
    (kid:    INTEGER,
    pid:    INTEGER,
    PRIMARY KEY  (kid, pid),
    FOREIGN KEY (kid)
      REFERENCES Kante,
    FOREIGN KEY (pid)
      REFERENCES Punkt);
```

Modellierung in SQL2003

```
CREATE TYPE PunktT (X FLOAT, Y FLOAT, Z
FLOAT) INSTANTIABLE NOT FINAL;

CREATE TYPE KanteT (Punkt1 REF (PunktT), Punkt2
REF (PunktT)) INSTANTIABLE NOT FINAL;

CREATE TABLE Punkt OF PunktT (REF IS pid
SYSTEM GENERATED);

CREATE TABLE Kante OF KanteT (REF IS kid
SYSTEM GENERATED,
Punkt1 SCOPE Punkt, Punkt2 SCOPE Punkt);

CREATE TABLE POLYEDER (
   VID INT,
   Flaechen ROW (
      FlaechenID INT,
      Kanten REF (KanteT) SCOPE Kante
MULTTISET)
   ) MULTISET,
CHECK CARDINALITY(Flaechen)>3);
```

Aufgabe 4 – Anfragen Polyeder

a) Finde alle Punkte, die zu Flächenobjekten mit der FlaechenID < 3 gehören?

```
CREATE TYPE PunktT (X FLOAT, Y FLOAT, Z FLOAT) INSTANTIABLE NOT FINAL;

CREATE TYPE KanteT (Punkt1 REF (PunktT), Punkt2 REF (PunktT)) INSTANTIABLE NOT
FINAL;

CREATE TABLE Punkt OF PunktT (REF IS pid SYSTEM GENERATED);

CREATE TABLE Kante OF KanteT (REF IS kid SYSTEM GENERATED,
Punkt1 SCOPE Punkt, Punkt2 SCOPE Punkt);

CREATE TABLE POLYEDER (
   VID INT, Flaechen ROW (FlaechenID INT, Kanten REF (KanteT) SCOPE Kante MULTTISET)
   ) MULTISET,
CHECK CARDINALITY(Flaechen)>3);
```

```
SELECT P.*
FROM Unnest(Polyeder.Flaechen) AS F, Unnest(F.Kanten) AS K,
        (SELECT K->Punkt1->X, K->Punkt1->Y, K->Punkt1->Z FROM K
          UNION
         SELECT K->Punkt2->X, K->Punkt2->Y, K->Punkt2->Z FROM K) AS P
WHERE F.FlaechenID < 3;
```

Aufgabe 4 – Anfragen Polyeder

b) Finde alle Flächen, die mit Punkt (50,44,75) assoziiert sind?

```
CREATE TYPE PunktT (X FLOAT, Y FLOAT, Z FLOAT) INSTANTIABLE NOT FINAL;

CREATE TYPE KanteT (Punkt1 REF (PunktT), Punkt2 REF (PunktT)) INSTANTIABLE NOT
FINAL;

CREATE TABLE Punkt OF PunktT (REF IS pid SYSTEM GENERATED);

CREATE TABLE Kante OF KanteT (REF IS kid SYSTEM GENERATED,
Punkt1 SCOPE Punkt, Punkt2 SCOPE Punkt);

CREATE TABLE POLYEDER (
   VID INT, Flaechen ROW (FlaechenID INT, Kanten REF (KanteT) SCOPE Kante MULTTISET)
   ) MULTISET,
CHECK CARDINALITY(Flaechen)>3);
```

```
SELECT F.FlaechenID
FROM Unnest(Polyeder.Flaechen) AS F, Unnest(F.Kanten) AS K
WHERE (K->Punkt1->X=50 AND K->Punkt1->Y=44 AND K->Punkt1->Z=75) OR
        (K->Punkt2->X=50 AND K->Punkt2->Y=44 AND K->Punkt2->Z=75);
```

Aufgabe 5 - Rekursion

```
WITH RECURSIVE TR_CLOSURE
(OTEIL,UTEIL,ANZAHL)AS(
//Rekursionsanfang direktes Unterteil
    Select Teil, Uteil, Anzahl
        FROM Stueckliste
Where Teil =‚01‘
//Rekursion Unterteil des Unterteils
UNION ALL
Select s.Teil,s.Uteil,
s.Anzahl*hu.ANZAHL
FROM Stueckliste s, TR_CLOSURE hu
     WHERE
      hu.UTEIL = s.TEIL
)

Select UTEIL,SUM(Anzahl) as Anz
FROM TR_CLOSURE
GROUP BY UTEIL
```

TR_CLOSURE

OTEIL	UTEIL	Anzahl

Stueckliste

TEIL	UTEIL	Anzahl
00	01	2
00	04	4
00	05	3
01	03	2
01	02	7
02	06	4
02	07	4
02	08	3
03	08	2
04	09	2
04	08	4
05	10	1
05	11	3