Fachbereich 07 Informatik/Mathematik

Praktikum Datenbanksysteme II
Wintersemester 2018/19

Prof. Dr. Martin Staudt

Übung 2

Wimmer, Anja
IF8
Gabl, Daniel
IF6

30.11.2018

Inhaltsverzeichnis

Aufgaben

Aufgabe 1

-

Aufgabe 2

Die zwei Möglichkeiten, die wir in Betracht ziehen:

1. Allmögliche Objekt-Typen definieren und gegenseitig referenzieren. Bspw. könnte die Adresse ein Typ sein, die sich aus der Straße und der Hausnummer (und ggf. PLZ und Ort) zusammensetzt. Auch könnte ein Kontotyp mit Konto-Nr., Kontostand, Art und ID der Zweigstelle ein eigenes Attribut sein. (Es ist keine Zuordnungstabelle erforderlich, da es sich bei Zweigstelle <-> Konto um eine 1:n-Beziehung handelt.)
2. Beispielsweiße könnten wir den Konto-Typ nicht als eigene Tabelle speichern sondern als innere Tabelle beim Zweigstellen-Typs speichern, dadurch entfällt die Referenz auf diese Tabelle.

Aufgabe 3

Wir würden folgendes Schema aufstellen: (Legende: 1. Möglichkeit, **2. Möglichkeit**, beide)

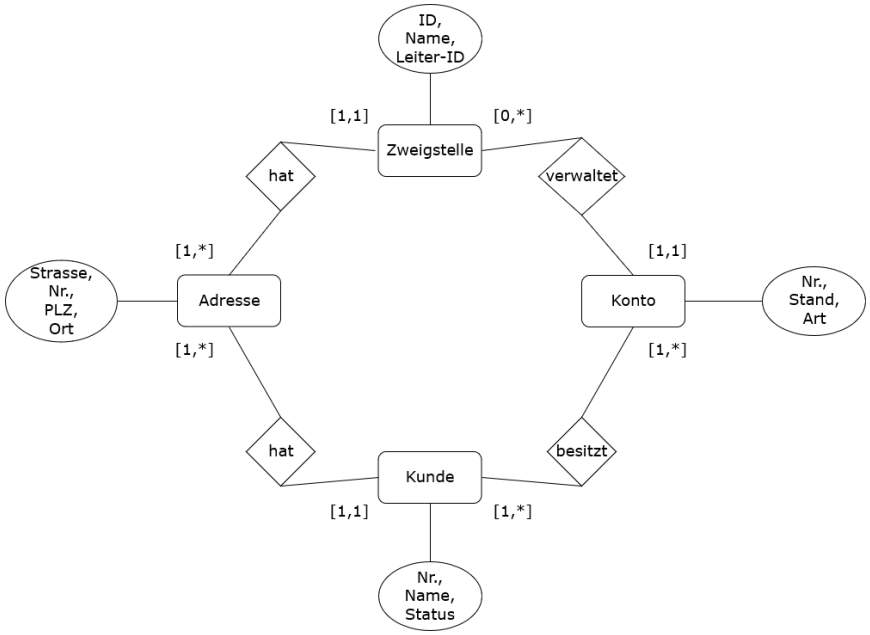
5 Typen wie folgt:

- Adress-Typ mit Straße und Hausnummer (und ggf. PLZ und Ort)
- **Kontolisten-Typ als Tabelle vom Typ Integer (Kontonummern)**
- Kunden-Typ mit Kunden-Nr., -Name, Adress-Typ, Status **und Kontolisten-Typ**
- Kontoinhaber-Typ als Tabelle vom Kunden-Typ
- **Zweigstellenkonten-Typ als Tabelle von Konto-Typen**
- Zweigstellen-Typ mit Zweigstellenname, Adress-Typ, Leiter-Id **und Zweigkonten**
- Konto-Typ mit Konto-Nr., Kontostand, Art, Kontoinhaber-Typ und Zweigstellen-Typ

Dazu noch folgende Tabellen:

- Kunden-Tabelle mit Kunden-Typ
- Zweigstellen-Adresse mit Zweigstellen-Typ
- Konto-Tabelle mit Konto-Typ **(entfällt bei der 2. Möglichkeit)**

Eine Skizzierung des Zusammenspiels der Typen und Tabellen:



SQL-Statements zum Erzeugen der Typen und Tabellen **(1. Möglichkeit)**:

```
CREATE TYPE AddressType AS Object(street VARCHAR(31), houseNr
VARCHAR(7), zip INT(5), place VARCHAR(31));
/
CREATE TYPE CustomerType AS Object(customerNr INT, customerName
VARCHAR(63), addr AddressType, status VARCHAR(15));
/
CREATE TYPE AccountOwnerType AS TABLE OF REF CustomerType;
/
CREATE TYPE BranchOfficeType AS Object(branchOfficeName VARCHAR(63),
addr AddressType, leaderId INT);
/
CREATE TYPE AccountType AS Object(accountNr INT,
balance DOUBLE PRECISION, kind VARCHAR(1),
owners AccountOwnerType, branchOffice REF BranchOfficeType);
/
CREATE TABLE Customer OF CustomerType;
CREATE TABLE BranchOffice OF BranchOfficeType;
CREATE TABLE AccountTable OF AccountType NESTED TABLE owners STORE
AS lorem_ipsum;
```

SQL-Statements zum Einfügen von Beispieldatensätzen in die Datenbank **(1. Möglichkeit)**:

```
INSERT INTO Customer VALUES(CustomerType(2345, 'H. Fach',
AddressType('Münchenerstr.', '33', 60329, ' Frankfurt am Main'),
'Geschäftskunde'));

INSERT INTO Customer VALUES(CustomerType(7654, 'B. Meier',
AddressType('Eschenweg', '12', 85354, 'Freising'), 'Privatkunde'));

INSERT INTO Customer VALUES(CustomerType(8764, 'J. Wiesner',
AddressType('Schellingstr.', '42', 80799, 'München'),
'Geschäftskunde'));


INSERT INTO BranchOffice VALUES(BranchOfficeType('Bachdorf',
AddressType('Hochstr.', '1', 81669, 'München'), 1768));

INSERT INTO BranchOffice VALUES(BranchOfficeType('Riedering',
AddressType('Simseestr.', '3', 81549, 'München'), 9823));


INSERT INTO AccountTable VALUES(AccountType(120768, 234.56, 'S',
AccountOwnerType((SELECT REF(c) FROM Customer c WHERE c.customerNr =
2345)), (SELECT REF(b) FROM BranchOffice b WHERE b.branchOfficeName =
'Bachdorf')));

INSERT INTO AccountTable VALUES(AccountType(678453, -456.78, 'G',
AccountOwnerType((SELECT REF(c) FROM Customer c WHERE c.customerNr =
8764)), (SELECT REF(b) FROM BranchOffice b WHERE b.branchOfficeName =
'Bachdorf')));

INSERT INTO AccountTable VALUES(AccountType(348973, 12567.56, 'G',
AccountOwnerType((SELECT REF(c) FROM Customer c WHERE c.customerNr =
2345), (SELECT REF(c) FROM Customer c WHERE c.customerNr = 8764)),
(SELECT REF(b) FROM BranchOffice b WHERE b.branchOfficeName =
'Bachdorf')));

INSERT INTO AccountTable VALUES(AccountType(987654, 789.65, 'G',
AccountOwnerType((SELECT REF(c) FROM Customer c WHERE c.customerNr =
7654)), (SELECT REF(b) FROM BranchOffice b WHERE b.branchOfficeName =
'Riedering')));

INSERT INTO AccountTable VALUES(AccountType(745363, -23.67, 'S',
AccountOwnerType((SELECT REF(c) FROM Customer c WHERE c.customerNr =
8764)), (SELECT REF(b) FROM BranchOffice b WHERE b.branchOfficeName =
'Riedering')));
```

SQL-Statements zum Erzeugen der Typen und Tabellen **(2. Möglichkeit)**:

```
CREATE TYPE AddressType AS Object(street VARCHAR(31), houseNr
VARCHAR(7), zip INT(5), place VARCHAR(31));
/
CREATE TYPE AccountsT AS TABLE OF INT;
/
CREATE TYPE CustomerType AS Object(customerNr INT, customerName
VARCHAR(63), addr AddressType, status VARCHAR(15), accNr AccountsT);
/
CREATE TYPE AccountType AS Object(accountNr INT,
balance DOUBLE PRECISION, kind VARCHAR(1));
/
CREATE TYPE BranchAccountsType AS TABLE OF AccountType;
/
CREATE TYPE BranchOfficeType AS Object(branchOfficeName VARCHAR(63),
addr AddressType, leaderId INT, accounts BranchAccountsType);
/

CREATE TABLE Customer OF CustomerType
NESTED TABLE accNr STORE AS accNr_useless;

CREATE TABLE BranchOffice OF BranchOfficeType
NESTED TABLE accounts STORE AS accounts_useless;
```

SQL-Statements zum Einfügen von Beispieldatensätzen in die Datenbank **(2. Möglichkeit)**:

```
INSERT INTO Customer
VALUES(CustomerType(2345, 'H. Fach',
AddressType('Münchenerstr.', '33', 60329, 'Frankfurt am Main'),
'Geschäftskunde', AccountsT(120768, 348973)));

INSERT INTO Customer
VALUES(CustomerType(7654, 'B. Meier',
AddressType('Eschenweg', '12', 85354, 'Freising'), 'Privatkunde',
AccountsT(987654)));

INSERT INTO Customer
VALUES(CustomerType(8764, 'J. Wiesner',
AddressType('Schellingstr.', '42', 80799, 'München'), 'Geschäftskunde',
AccountsT(745363, 678453, 348973)));

INSERT INTO BranchOffice
VALUES(BranchOfficeType('Bachdorf', AddressType('Hochstr.', '1', 81669,
'München'), 1768, BranchAccountsType()));

INSERT INTO TABLE(SELECT accounts FROM BranchOffice WHERE
branchOfficeName='Bachdorf')
VALUES(AccountType(120768, 234.56, 'S'));

INSERT INTO TABLE(SELECT accounts FROM BranchOffice WHERE
branchOfficeName='Bachdorf')
VALUES(AccountType(678453, -456.78, 'G'));

INSERT INTO TABLE(SELECT accounts FROM BranchOffice WHERE
branchOfficeName='Bachdorf')
VALUES(AccountType(348973, 12567.56, 'G'));

INSERT INTO BranchOffice
VALUES(BranchOfficeType('Riedering', AddressType('Simseestr.', '3',
81549, 'München'), 9823, BranchAccountsType()));

INSERT INTO TABLE(SELECT accounts FROM BranchOffice WHERE
branchOfficeName='Riedering')
VALUES(AccountType(987654, 789.65, 'G'));

INSERT INTO TABLE(SELECT accounts FROM BranchOffice WHERE
branchOfficeName='Riedering')
VALUES(AccountType(745363, -23.67, 'S'));
```

Aufgabe 4

Bei der 1. Möglichkeit:

```
a) SELECT a.accountNr, a.balance, a.kind,
   CONCAT(CONCAT(DEREF(a.branchOffice).addr.street, ' '),
   DEREF(a.branchOffice).addr.houseNr) AS addr FROM AccountTable a;

b) SELECT a.accountNr, DEREF(o.COLUMN_VALUE).customerName AS
   customerName, CONCAT(CONCAT(DEREF(o.COLUMN_VALUE).addr.street, '
   '), DEREF(o.COLUMN_VALUE).addr.houseNr) as addr FROM AccountTable
   a, TABLE(a.owners) o;
```

Bei der 2. Möglichkeit:

```
a) SELECT a.accountNr, a.balance, a.kind,
   CONCAT(CONCAT(b.addr.street, ' '), b.addr.houseNr) AS addr
   FROM BranchOffice b, TABLE(b.accounts) a;

b) SELECT a.COLUMN_VALUE,
   CONCAT(CONCAT(c.addr.street, ' '),c.addr.houseNr) AS addr
   FROM Customer c, TABLE(c.accNr) a;
```

Screendumps

Hier Screendumps unserer Tabellen und den Ergebnissen aus Aufgabe 4.

1. Möglichkeit:

Screendump von Aufgabe 4a) (Kontonummer, -stand, -art und Adresse der Zweigstelle):

	ACCOUNTNR	BALANCE	KIND	ADDR
1	120768	234,56	S	Hochstr. 1
2	678453	-456,78	G	Hochstr. 1
3	348973	12567,56	G	Hochstr. 1
4	987654	789,65	G	Simseestr. 3
5	745363	-23,67	S	Simseestr. 3

Screendump von Aufgabe 4b) (Paare von Kontonummern, Namen und Adressen der Inhaber):

	ACCOUNTNR	CUSTOMERNAME	ADDR
1	120768	H. Fach	Münchenerstr. 33
2	678453	J. Wiesner	Schellingstr. 42
3	348973	H. Fach	Münchenerstr. 33
4	348973	J. Wiesner	Schellingstr. 42
5	987654	B. Meier	Eschenweg 12
6	745363	J. Wiesner	Schellingstr. 42

Zweigstellen-Tabelle:

	BRANCHOFFICENAME	ADDR	LEADERID
1	Bachdorf	[DBST42.ADDRESSTYPE]	1768
2	Riedering	[DBST42.ADDRESSTYPE]	9823

Kunden-Tabelle:

	CUSTOMERNR	CUSTOMERNAME	ADDR	STATUS
1	2345	H. Fach	[DBST42.ADDRESSTYPE]	Geschäftskunde
2	7654	B. Meier	[DBST42.ADDRESSTYPE]	Privatkunde
3	8764	J. Wiesner	[DBST42.ADDRESSTYPE]	Geschäftskunde

Konten-Tabelle:

	ACCOUNTNR	BALANCE	KIND	OWNERS	BRANCHOFFICE
1	120768	234,56	S	DBST42.ACCOUNTOWNERTYPE([DBST42.CUSTOMERTYPE])	[DBST42.BRANCHOFFICETYPE]
2	678453	-456,78	G	DBST42.ACCOUNTOWNERTYPE([DBST42.CUSTOMERTYPE])	[DBST42.BRANCHOFFICETYPE]
3	348973	12567,56	G	DBST42.ACCOUNTOWNERTYPE([DBST42.CUSTOMERTYPE],[DBST42.CUSTOMERTYPE])	[DBST42.BRANCHOFFICETYPE]
4	987654	789,65	G	DBST42.ACCOUNTOWNERTYPE([DBST42.CUSTOMERTYPE])	[DBST42.BRANCHOFFICETYPE]
5	745363	-23,67	S	DBST42.ACCOUNTOWNERTYPE([DBST42.CUSTOMERTYPE])	[DBST42.BRANCHOFFICETYPE]

2. Möglichkeit:

Screendump von Aufgabe 4a)

	ACCOUNTNR	BALANCE	KIND	ADDR
1	120768	234,56	S	Hochstr. 1
2	678453	-456,78	G	Hochstr. 1
3	348973	12567,56	G	Hochstr. 1
4	987654	789,65	G	Simseestr. 3
5	745363	-23,67	S	Simseestr. 3

Screendump von Aufgabe 4b)

	COLUMN_VALUE	ADDR
1	120768	Münchenerstr. 33
2	348973	Münchenerstr. 33
3	987654	Eschenweg 12
4	745363	Schellingstr. 42
5	678453	Schellingstr. 42
6	348973	Schellingstr. 42

Zweigstellen-Tabelle:

	BRANCHOFFICENAME	ADDR	LEADERID	ACCOUNTS
1	Bachdorf	[DBST42.ADDRESSTYPE]	1768	DBST42.BRANCHACCOUNTSTYPE([DBST42.ACCOUNTTYPE],[DBST42.ACCOUNTTYPE],[DBST42.ACCOUNTTYPE])
2	Riedering	[DBST42.ADDRESSTYPE]	9823	DBST42.BRANCHACCOUNTSTYPE([DBST42.ACCOUNTTYPE],[DBST42.ACCOUNTTYPE])

Kunden-Tabelle:

	CUSTOMERNR	CUSTOMERNAME	ADDR	STATUS	ACCNR
1	2345	H. Fach	[DBST42.ADDRESSTYPE]	Geschäftskunde	DBST42.ACCOUNTST(120768,348973)
2	7654	B. Meier	[DBST42.ADDRESSTYPE]	Privatkunde	DBST42.ACCOUNTST(987654)
3	8764	J. Wiesner	[DBST42.ADDRESSTYPE]	Geschäftskunde	DBST42.ACCOUNTST(745363,678453,348973)

Aufgabe 5 (Fallstudie)

Zuerst haben wir für die Fallstudie die Aufgabenstellung analysiert und angefangen, Typen zu definieren um die Tabellen vollständig korrekt zu speichern.
Hierfür haben wir 30 Typen (Normale und List-Typen) angelegt, die wie folgt aussehen:

```
CREATE TYPE CampusT AS OBJECT (campus_location VARCHAR(15),
campus_addr VARCHAR(127), campus_phone VARCHAR(15), campus_fax
VARCHAR(15), campus_head VARCHAR(31));
/
CREATE TYPE ProfessorT AS OBJECT (prof_id INTEGER, prof_name
VARCHAR(31), prof_contact VARCHAR(15), prof_research
VARCHAR(63), prof_year INTEGER);
/
CREATE TYPE ProfessorListT AS TABLE OF REF ProfessorT;
/
CREATE TYPE RCUnitT AS TABLE OF VARCHAR(127);
/
CREATE TYPE DepartmentT AS OBJECT (dept_id VARCHAR(3),
dept_name VARCHAR(31), dept_head VARCHAR(31), dept_prof
ProfessorListT);
/
CREATE TYPE DepartmentListT AS TABLE OF DepartmentT;
/
CREATE TYPE SchoolT AS OBJECT (school_id VARCHAR(3),
school_name VARCHAR(31), school_head VARCHAR (31), school_prof
ProfessorListT);
/
CREATE TYPE SchoolListT AS TABLE OF SchoolT;
/
CREATE TYPE ResearchCentreT AS OBJECT (rc_id VARCHAR(3),
rc_name VARCHAR(127), rc_unit RCunitT);
/
CREATE TYPE ResearchCentreListT AS TABLE OF ResearchCentreT;
/
-- TODO aggregation clustering technique
CREATE TYPE FacultyT AS OBJECT (fac_id INTEGER, fac_name
VARCHAR(31), fac_dean VARCHAR(15), dept DepartmentListT,
school SchoolListT, rc ResearchCentreListT);
/
CREATE TYPE BuildingT AS OBJECT (bld_id VARCHAR(4), bld_name
VARCHAR(31), bld_location VARCHAR(2), bld_level INTEGER,
campus REF CampusT, fac REF FacultyT);
/
CREATE TYPE PersonT AS OBJECT (person_id VARCHAR(8),
person_surname VARCHAR(15), person_forename VARCHAR(15),
person_title VARCHAR(7), person_addr VARCHAR(127),
person_phone VARCHAR(15), person_postcode VARCHAR(5), campus
REF CampusT) NOT FINAL;
/
```

```
CREATE TYPE OfficeT AS OBJECT (office_No VARCHAR(7), bld REF
BuildingT, office_phone VARCHAR(15));
/
CREATE TYPE ClassroomT AS OBJECT (class_no VARCHAR(4), bld REF
BuildingT, class_capacity INTEGER);
/
CREATE TYPE LabEquipmentT AS TABLE OF VARCHAR(15);
/
CREATE TYPE LabT AS OBJECT (lab_no VARCHAR(5), bld REF
BuildingT, lab_capacity INTEGER, lab_equipment LabEquipmentT);
/
CREATE TYPE DegreeT AS OBJECT (deg_id VARCHAR(4), deg_name
VARCHAR(31), deg_length INTEGER, deg_prereq VARCHAR(31), fac
REF FacultyT);
/
CREATE TYPE ComputerskillsT AS TABLE OF VARCHAR(15);
/
CREATE TYPE OfficeskillsT AS TABLE OF VARCHAR(31);
/
CREATE TYPE TechnicianskillsT AS TABLE OF VARCHAR(15);
/
CREATE TYPE StaffT UNDER PersonT (person_id VARCHAR(8),
office_No VARCHAR(7), staff_type VARCHAR(15)) NOT FINAL;
/
CREATE TYPE StudentT UNDER PersonT (person_id VARCHAR(8),
student_year INTEGER);
/
CREATE TYPE AdminT UNDER StaffT (person_id VARCHAR(8),
admin_title VARCHAR(31), admin_computerskills ComputerskillsT,
admin_officeskills OfficeskillsT);
/
CREATE TYPE TechnicianT UNDER StaffT (person_id VARCHAR(8),
tech_title VARCHAR(15), tech_skills TechnicianskillsT);
/
CREATE TYPE TutorT UNDER StaffT (person_id VARCHAR(8),
tutor_hours INTEGER, tutor_rate DOUBLE PRECISION);
/
CREATE TYPE LecturerT UNDER StaffT (person_id VARCHAR(8),
lect_area VARCHAR(31), lect_type VARCHAR(15)) NOT FINAL;
/
CREATE TYPE SeniorLecturerT UNDER LecturerT (person_id
VARCHAR(8), senlect_phd INTEGER, senlect_master INTEGER,
senlect_honours INTEGER);
/
CREATE TYPE AssociateLecturerT UNDER LecturerT (person_id
VARCHAR(8), asslect_honours INTEGER, asslect_year INTEGER);
/
CREATE TYPE SubjectT AS OBJECT (subject_id VARCHAR(8),
subject_name VARCHAR(31), subject_credit INTEGER,
subject_prereq VARCHAR(8), person REF PersonT);
/
```

Dazu haben wir noch folgende 20 Tabellen angelegt, um konkrete Exemplare zu speichern:

```
CREATE TABLE Enrolls_in (student_id VARCHAR(8), deg_id
VARCHAR(4));

CREATE TABLE Takes (student_id VARCHAR(8), subject_id
VARCHAR(8), mark INTEGER);

CREATE TABLE Campus OF CampusT;

CREATE TABLE Professor OF ProfessorT;

CREATE TABLE Faculty OF FacultyT
     NESTED TABLE dept STORE AS department_nm
     NESTED TABLE school STORE AS school_nm
     NESTED TABLE rc STORE AS rc_nm;

CREATE TABLE Building OF BuildingT;

CREATE TABLE Person OF PersonT;

CREATE TABLE Office OF OfficeT;

CREATE TABLE Classroom OF ClassroomT;

CREATE TABLE Lab OF LabT
     NESTED TABLE lab_equipment STORE AS lab_equip_nm;

CREATE TABLE DegreeTbl OF DegreeT;

CREATE TABLE Staff OF StaffT;

CREATE TABLE Student OF StudentT;

CREATE TABLE AdminTbl OF AdminT
     NESTED TABLE admin_computerskills STORE AS adm_comskill_nm
     NESTED TABLE admin_officeskills STORE AS adm_offskill_nm;

CREATE TABLE Technician OF TechnicianT
     NESTED TABLE tech_skills STORE AS tech_skill_nm;

CREATE TABLE Tutor OF TutorT;

CREATE TABLE Lecturer OF LecturerT;

CREATE TABLE SeniorLecturer OF SeniorLecturerT;

CREATE TABLE AssociateLecturer OF AssociateLecturerT;

CREATE TABLE Subject OF SubjectT;
```

Dazu haben wir noch die im Diagramm eingezeichneten Funktionen wir folgt implementiert:

```
CREATE OR REPLACE FUNCTION show_bld_details(in_bld_id IN
NUMBER)
   RETURN VARCHAR2
   IS building_details VARCHAR2(255);

BEGIN

SELECT 'ID: '|| building.bld_id ||', Name: '||
building.bld_name ||',
      Location: '|| building.bld_location ||', Level: '||
building.bld_level ||''
  INTO building_details
  FROM Building building
  WHERE building.bld_id = in_bld_id;

  RETURN(building_details);

END show_bld_details;


CREATE OR REPLACE FUNCTION insert_student(in_person IN Person,
in_year IN NUMBER)
   RETURN BOOLEAN
   IS success_state BOOLEAN;

BEGIN

INSERT INTO Student
  SELECT pers.*, in_year FROM Person pers
  WHERE pers.person_id = in_person.person_id;
success_state := TRUE;

  RETURN(success_state);

END insert_student;


CREATE OR REPLACE FUNCTION delete_student(in_person IN Person)
   RETURN BOOLEAN
   IS success_state BOOLEAN;

BEGIN

DELETE FROM Student student WHERE student.person_id =
in_person.person_id;
success_state := TRUE;

  RETURN(success_state);

END delete_student;
```

Schlussendlich haben wir mittels folgenden Insert-Statements Daten in die Tabellen angelegt:

Quellcode