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

Inhaltsverzeichnis II

Aufgaben 1

Aufgabe 1 1

Aufgabe 2 1

Aufgabe 3 1

Aufgabe 4 4

Aufgabe 5 (Fallstudie) 6

Quellcode 11

# 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)

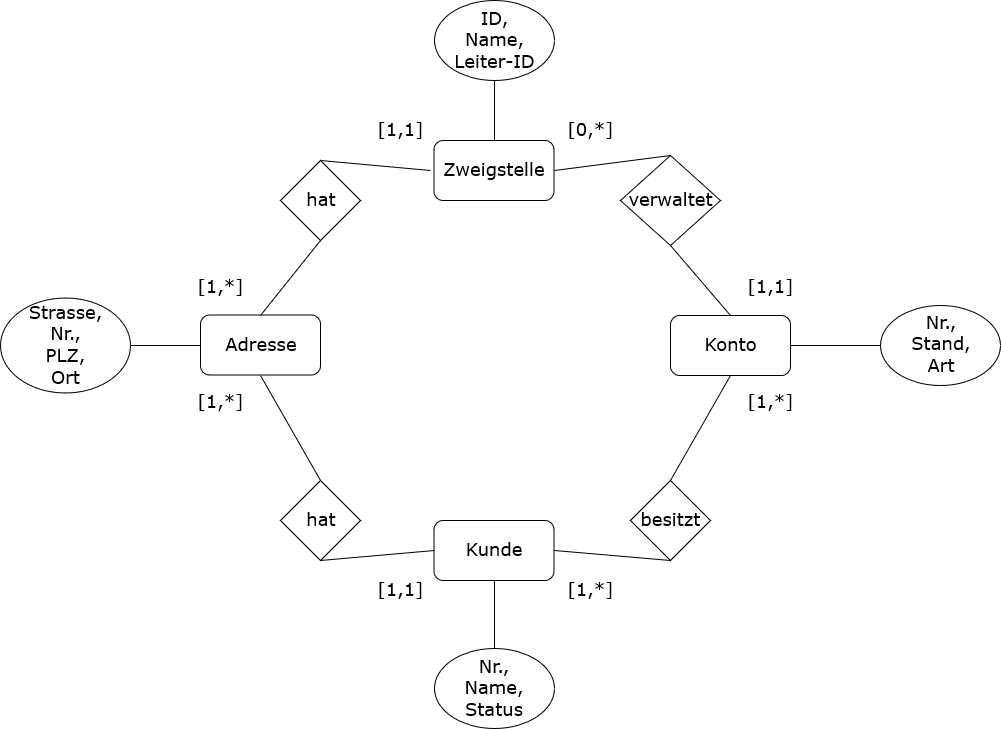
5 Typen wie folgt:

* 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:

1. SELECT a.accountNr, a.balance, a.kind, CONCAT(CONCAT(DEREF(a.branchOffice).addr.street, ' '), DEREF(a.branchOffice).addr.houseNr) AS addr FROM AccountTable a;
2. 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:

1. SELECT a.accountNr, a.balance, a.kind,

CONCAT(CONCAT(b.addr.street, ' '), b.addr.houseNr) AS addr

FROM BranchOffice b, TABLE(b.accounts) a;

1. SELECT a.COLUMN\_VALUE,

CONCAT(CONCAT(c.addr.street, ' '),c.addr.houseNr) AS addr

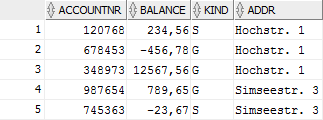
FROM Customer c, TABLE(c.accNr) a;

**Screendumps**

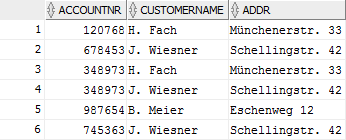
Hier Screendumps unserer Tabellen und den Ergebnissen aus Aufgabe 4.

**1. Möglichkeit:**

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



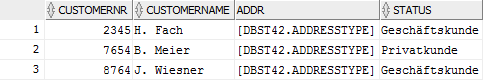
Screendump von Aufgabe 4b) (Paare von Kontonummern, Namen und Adressen der Inhaber):



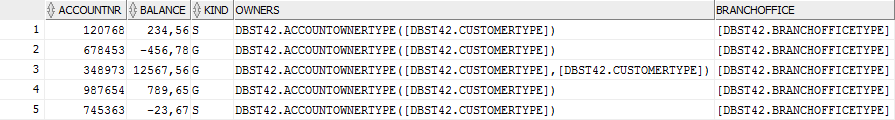
Zweigstellen-Tabelle:



Kunden-Tabelle:

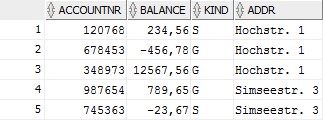


Konten-Tabelle:

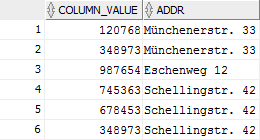


**2. Möglichkeit:**

Screendump von Aufgabe 4a)



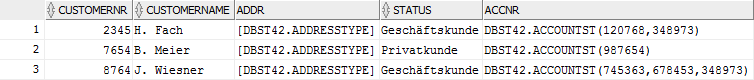
Screendump von Aufgabe 4b)



Zweigstellen-Tabelle:



Kunden-Tabelle:



## 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