Bericht Datenbank Praktikum

Julian Sobott (76511), David Sugar (76050), Lukas Mendel (76509)

9. Januar 2020

Inhaltsverzeichnis

0.1 Aufgabe 1: Entwurf und Implementierung des Datenmodells

0.1.1 a) ERM

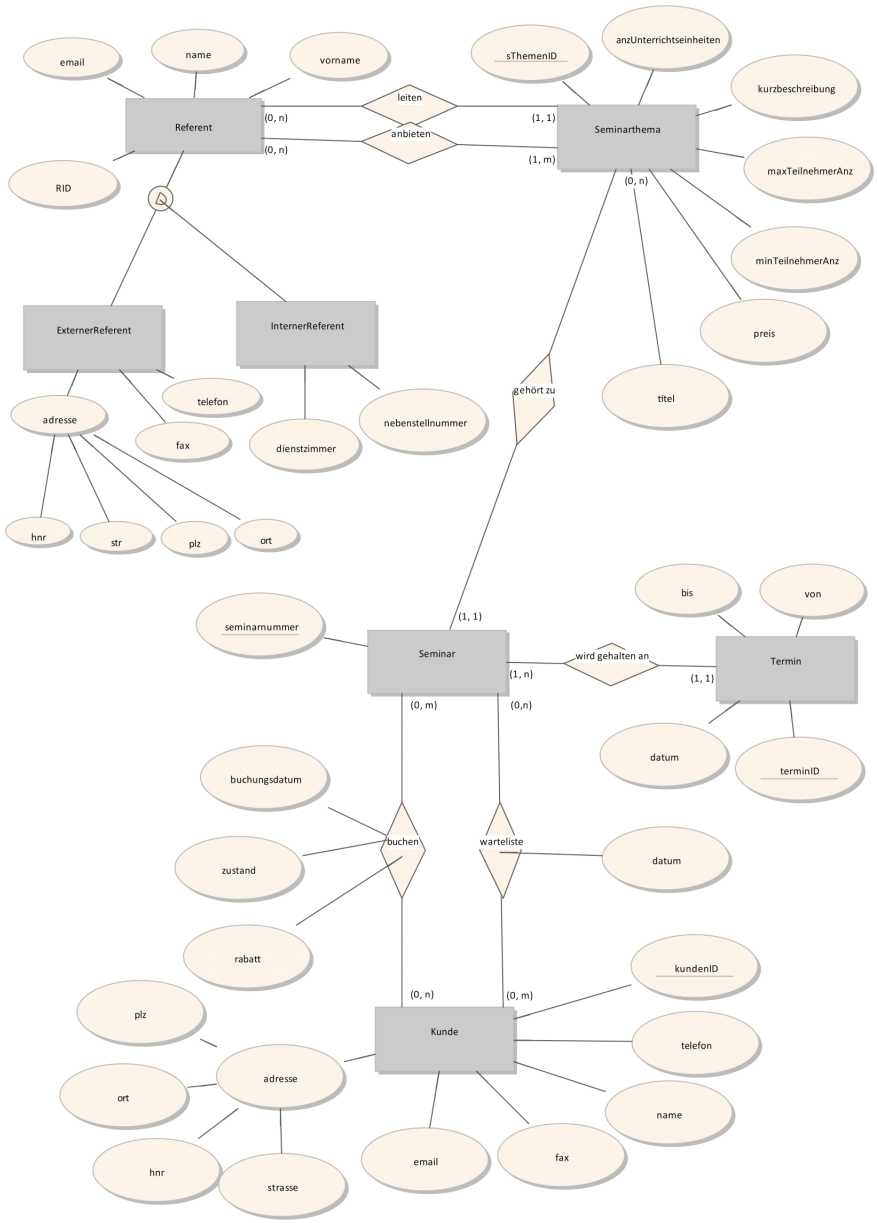

Abbildung 1: ER-Modell Seminarverwaltung

Beschreibung des Modells

is_a Beziehung zwischen Referenten: Aufgrund der Tatsache, dass alle Referenten sowohl gemeinsame als auch verschiedene Attribute haben, haben wir uns für eine disjunkte Spezialisierung entschieden. Das heißt die gemeinsamen Attribute sind in der generellen Entity Referent und nur die speziellen Attribute sind in den Spezialisierungen Externer Referent und Interner Referent.
Beziehung zwischen Referent und Seminarthema: Da zwischen einem Seminar leiten und einem Seminar anbiten unterschieden wird, haben wir uns für zwei Relationen zwischen Referent und Seminarthema entschieden.
Beziehung zwischen Seminar und Kunde: Selbiges gilt in dieser Beziehung. Ein Kunde kann sowohl ein Seminar buchen als auch in einer Warteliste landen.
Beziehung zwischen Seminarthema und Seminar: Es kann mehrere Veranstaltungen (Seminare) zu einem Seminarthema geben. Wobei ein Seminar immer nur über ein Seminarthema geht.
Beziehung zwischen Seminar und Termin: Da ein Seminar an mehreren Terminen stattfinden kann, wurde der Termin in eine extra Entity ausgelagert.

Unterschiede zum UML Modell

- Im UML Diagramm wurde das Listenmodell zwischen Seminar und Termin verwendet. Dies ist im ERM aber nicht möglich und ist stattdessen eine Relation.
- Die Beziehung zwischen Seminarthema und Seminar ist die gleiche, nur statt Exemplartyp wurde eine Relation verwendet.
- Im UML wird mithilfe von Vererbung eine Klasse um die benötigten Attribute erweitertt, um eine Spezialisierung zu erreichen. Im ERM hingegen wird über eine is_a Beziehung eine Spezialisierung realisiert.
- Im UML wurde eine Koordinatorklasse für die Buchungen verwendet. Im ERM hingegen wird hier eine Relation mit Attributen verwendet.
- Anstatt der *ordered* Constraint wird im ERM eine Relation genommen, die das Datum speichert. Durch das Datum kann eine Ordnung hergestellt werden.

Entities

seminar = ({SEMINARNUMMER:INTEGER})
termin = ({TERMINID:INTEGER, DATUM:DATE, VON:DATETIME, BIS:DATETIME})
kunde = ({KUNDENID:INTEGER, TELEFON:VARCHAR, NAME:VARCHAR, FAX:VARCHAR, EMAIL:VARCHAR, ADRESSE:(PLZ:VARCHR, ORT:VARCHAR, HNR:VARCHAR, STR:VARCHAR)})
referent = ({RID:INTEGER, email: VARCHAR, name: VARCHAR, vorname: VARCHAR})
seminarthema = (STHEMAID:INTEGER, ANZUNTERICHTSEINHEITEN:INTEGER, KURZBESCHREIBUNG:VARCHAR, MAXTEILNEHMERANZ:INTEGER, MINTEILNEHMERANZ:INTEGER, PREIS:FLOAT, TITEL:VARCHAR})
externerReferent = ({RID:INTEGER,adresse: (plz: VARCHAR, ort: VARCHAR, strasse: VARCHAR, hnr: VARCHAR)} is_a referent)
internerReferent = ({RID:INTEGER,dienstzimmer: VARCHAR, nebenstellnummer: Integer} is_a referent)

Relations

leiten = (referent X seminarthema)
anbieten = (referent X seminarthema)
gehört_zu = (seminarthema X seminar)
buchen = (seminar x kunke, BUCHUNGSDATUM:DATE, ZUSTAND:VARCHAR, RABATT:FLOAT)
warteliste = (kunde x seminar, POSITION:INTEGER)
wird_gehalten_an = (seminar x termin)

0.1.2 b) Relationales Modell

Relations

referent = (RID:INTEGER, email: VARCHAR, name: VARCHAR, vorname: VARCHAR)
ExternerReferent(RID:INTEGER, plz: VARCHAR, ort: VARCHAR, strasse: VARCHAR, hnr: VARCHAR)
IntererReferent (RID:INTEGER, dienstzimmer: VARCHAR, nebenstellnummer: Integer)
seminarthema = (STHEMAID:INTEGER, ANZUNTERICHTSEINHEITEN:INTEGER, KURZBESCHREIBUNG:VARCHAR, MAXTEILNEHMERANZ:INTEGER, MINTEILNEHMERANZ:INTEGER, PREIS:FLOAT, TITEL:VARCHAR, LEITER:INTEGER)
anbieten = (REFERENTID:INTEGER, SEMINARTHEMAID:INTEGER)
seminar = (SEMINARNUMMER:INTEGER, SEMINARTHEMAID:INTEGER)
termin = (TERMINID:INTEGER, VON:DATETIME, BIS:DATETIME, DATUM:DATE, SEMINARID:INTEGER)
kunde = (KUNDENID:INTEGER, TELEFON:VARCHAR, NAME:VARCHAR, FAX:VARCHAR, EMAIL:VARCHAR, PLZ:VARCHR, ORT:VARCHAR, HNR:VARCHAR, STR:VARCHAR)
buchen = (KUNDENID:INTEGER, SEMINARNR:INTEGER, BUCHUNGSDATUM:DATE, ZUSTAND:VARCHAR, RABATT:FLOAT)
warteliste = (KUNDENID:INTEGER, SEMINARNR:INTEGER, POSTION:INTEGER)

Referenzen

$seminarthema|_{LEITER} \subseteq refernet|_{RID}$
$anbieten|_{REFERENTID} \subseteq refernet|_{RID}$
$anbieten|_{SEMINARTHEMAID} \subseteq seminarthema|_{STHEMAID}$
$seminar|_{SEMINARTHEMAID} \subseteq seminarthema|_{STHEMAID}$
$termin|_{SEMINARID} \subseteq seminar|_{SEMINARNUMMER}$
$buchen|_{KUNDENID} \subseteq kunde|_{KUNDENID}$
$buchen|_{SEMINARNR} \subseteq seminar|_{SEMINARNR}$
$warteliste|_{KUNDENID} \subseteq kunde|_{KUNDENID}$
$warteliste|_{SEMINARNR} \subseteq seminar|_{SEMINARNR}$

0.1.3 c) SQL-Staments: create tables

```
CREATE TABLE g8_referent (
    rid serial PRIMARY KEY,
    email VARCHAR(50),
    name VARCHAR(50),
    vorname VARCHAR(50)
);
```

```
CREATE TABLE g8_seminarthema (
    sthemaid serial PRIMARY KEY,
    anz_unterrichtseinheiten INTEGER,
    kurzbeschreibung VARCHAR,
    max_teilnehmeranzahl INTEGER,
    min_teilnehmeranzahl INTEGER,
    preis FLOAT,
    titel VARCHAR(200),
    leiter INTEGER REFERENCES g8_referent(rid)
);

CREATE TABLE g8_anbieten (
    referenten_id INTEGER REFERENCES g8_referent(rid),
    sthemaid INTEGER REFERENCES g8_seminarthema(sthemaid),
    PRIMARY KEY (referenten_id, sthemaid)
);

CREATE TABLE g8_seminar (
    seminarnummer serial PRIMARY KEY,
    sthemaid INTEGER REFERENCES g8_seminarthema(sthemaid)
);


CREATE TABLE g8_termin (
    terminid serial PRIMARY KEY,
    von TIME,
    bis TIME,
    datum DATE,
    seminarnummer INTEGER REFERENCES g8_seminar(seminarnummer)
);

CREATE TABLE g8_kunde (
    kundenid serial PRIMARY KEY,
    telefon VARCHAR(20),
    name VARCHAR(50),
    fax VARCHAR(20),
    email VARCHAR(50),
    plz VARCHAR(10),
    ort VARCHAR(50),
    hnr VARCHAR(10),
    str VARCHAR(50)
);

CREATE TYPE g8_zustand as ENUM ('offen', 'gebucht', 'berechnet', 'gezahlt', '
    storniert');

CREATE TABLE g8_buchen (
    kundenid INTEGER REFERENCES g8_kunde(kundenid),
    seminarnummer INTEGER REFERENCES g8_seminar(seminarnummer),
    datum DATE,
    zustand g8_zustand,
    rabatt FLOAT,
    PRIMARY KEY (kundenid, seminarnummer)
);

CREATE TABLE g8_warteliste (
    kundenid INTEGER REFERENCES g8_kunde(kundenid),
    seminarnummer INTEGER REFERENCES g8_seminar(seminarnummer),
    position INTEGER,
    PRIMARY KEY (kundenid, seminarnummer)
```

```
);

CREATE TABLE g8_ExternerReferent(

RID int PRIMARY KEY,
fax VARCHAR(50),
telefon VARCHAR(20),
PLZ VARCHAR(15),
Strasse VARCHAR(50),
Hnr VARCHAR (20),
Ort VARCHAR(50),

FOREIGN KEY (RID) REFERENCES g8_referent(RID)
);

CREATE TABLE g8_InternerRefrent(

RID int PRIMARY KEY,
Dienstnummer VARCHAR(30),
nebenstellennummer INTEGER,

FOREIGN KEY (RID) REFERENCES g8_referent(RID)

);
```

0.2 Aufgabe 2: SQL-Statements für das Einfügen von Datensätzen

```
INSERT INTO g8_referent(email, vorname, name) values
    ('julian.sobott@wtf.de', 'Julian', 'Sobott'),
    ('david.sugar@wtf.de', 'David', 'Sugar'),
    ('lukas.mendel@wtf.de', 'Lukas', 'Mendel'),
    ('gregor.grambow@wtf.de', 'Gregor', 'Grambow');

INSERT INTO g8_internerreferent(rid, plz, ort, strasse, hnr) values
    ((select rid from g8_referent where name = 'Grambow' and vorname = 'Gregor')
        , '73434', 'Aalen', 'Uni-Str', '111');

INSERT INTO g8_externerreferent(rid, plz, ort, strasse, hnr) values
    ((select rid from g8_referent where name = 'Sugar' and vorname = 'David'), '
        73434', 'Aalen', 'Uni-Str', '111'),
    ((select rid from g8_referent where name = 'Sobott' and vorname = 'Julian'),
         '73434', 'Aalen', 'Uni-Str', '111'),
    ((select rid from g8_referent where name = 'Mendel' and vorname = 'Lukas'),
        '73434', 'Aalen', 'Uni-Str', '111');


INSERT INTO g8_seminarthema(anz_unterrichtseinheiten, kurzbeschreibung,
   max_teilnehmeranzahl, min_teilnehmeranzahl, preis, titel, leiter) values
    (10, 'Datenbanken Grundlagen erlernen.', 30, 5, 152.50, 'Datenbanken', (
        select rid from g8_referent where name = 'Grambow' and vorname = 'Gregor'
        )),
    (2, 'We love RISC', 10, 1, 0.0, 'The ARM Architecture', (select rid from
        g8_referent where name = 'Sugar' and vorname = 'David')),
    (3, 'Its not a snake', 15, 3, 43.90, 'Python', (select rid from g8_referent
        where name = 'Julian' and vorname = 'Sobott'));

INSERT INTO g8_seminar (sthemaid)
values (1), (2), (3), (1), (3);

INSERT INTO g8_termin (von, bis, datum, seminarnummer)
values
```

```
('09:30', '13:00', '18/1/1999', 1),
('09:30', '13:00', '19/1/1999', 2),
('09:30', '13:00', '20/1/1999', 3),
('09:30', '13:00', '21/1/1999', 4),
('09:30', '13:00', '22/1/1999', 5),
('10:30', '18:45', '18/1/2050', 1),
('10:30', '18:45', '19/1/2050', 5);

INSERT INTO g8_kunde (telefon, name, fax, email, plz, ort, hnr, str)
values
('0176111', 'Pete', '0176-54', 'pete@bs.de', '12345', 'Buxdehude', '3', '
    kennIchNichtWeg'),
('0176112', 'Steve', '0176-55', 'steve@bs.de', '12345', 'Buxdehude', '3', '
    kennIchNichtWeg'),
('0176113', 'Eve', '0176-56', 'eve@bs.de', '12345', 'Buxdehude', '3', '
    kennIchNichtWeg'),
('0176114', 'Paula', '0176-57', 'paula@bs.de', '12345', 'Buxdehude', '3', '
    kennIchNichtWeg'),
('0176115', 'Klaus', '0176-58', 'klaus@bs.de', '12345', 'Buxdehude', '3', '
    kennIchNichtWeg');

INSERT INTO g8_buchen (kundenid, seminarnummer, datum, zustand, rabatt)
values
(1, 1, '13/1/1999', 'gezahlt', 0.0),
(1, 2, '13/1/1999', 'gezahlt', 0.0),
(2, 1, '13/1/1999', 'berechnet', 0.3),
(2, 3, '13/1/1999', 'gezahlt', 0.0),
(3, 3, '14/1/1999', 'gebucht', 0.0),
(3, 4, '14/1/1999', 'gezahlt', 0.0),
(4, 3, '14/1/1999', 'gebucht', 0.0),
(5, 3, '14/1/1999', 'gebucht', 0.0),
(3, 2, '15/1/1999', 'offen', 0.0),
(1, 3, '14/1/1999', 'berechnet', 0.7);

INSERT INTO g8_warteliste (kundenid, seminarnummer, position)
values
(1,1,1),
(2,1,2),
(3,2,1);
```

0.3 Aufgabe 3: SQL-Statements für Datenabfrage

```
SELECT(
     (SELECT COUNT (rid) as AnzahlInterne
     FROM g8_InternerReferent),
     (SELECT COUNT (rid) as AnzahlExterne
     FROM g8_ExternerReferent),
     (SELECT COUNT (rid) as AnzahlGesamt
     FROM g8_referent)

);

SELECT seminarnnummer, COUNT(seminarnnummer)
FROM g8_seminar s JOIN g8_termin t on s.seminarnummer = t.seminarnummer
GROUP BY seminarnummer;

SELECT Seminarummer, COUNT(Seminarummer) as Teilnehmeranzahl
FROM g8_buchen b JOIN g8_seminar s on b.seminarnummer = s.seminarnummer
GROUP BY(Seminarnummer)

SELET Seminarummer, , MAX()
```

0.4 Aufgabe 4:

0.4.1 a)

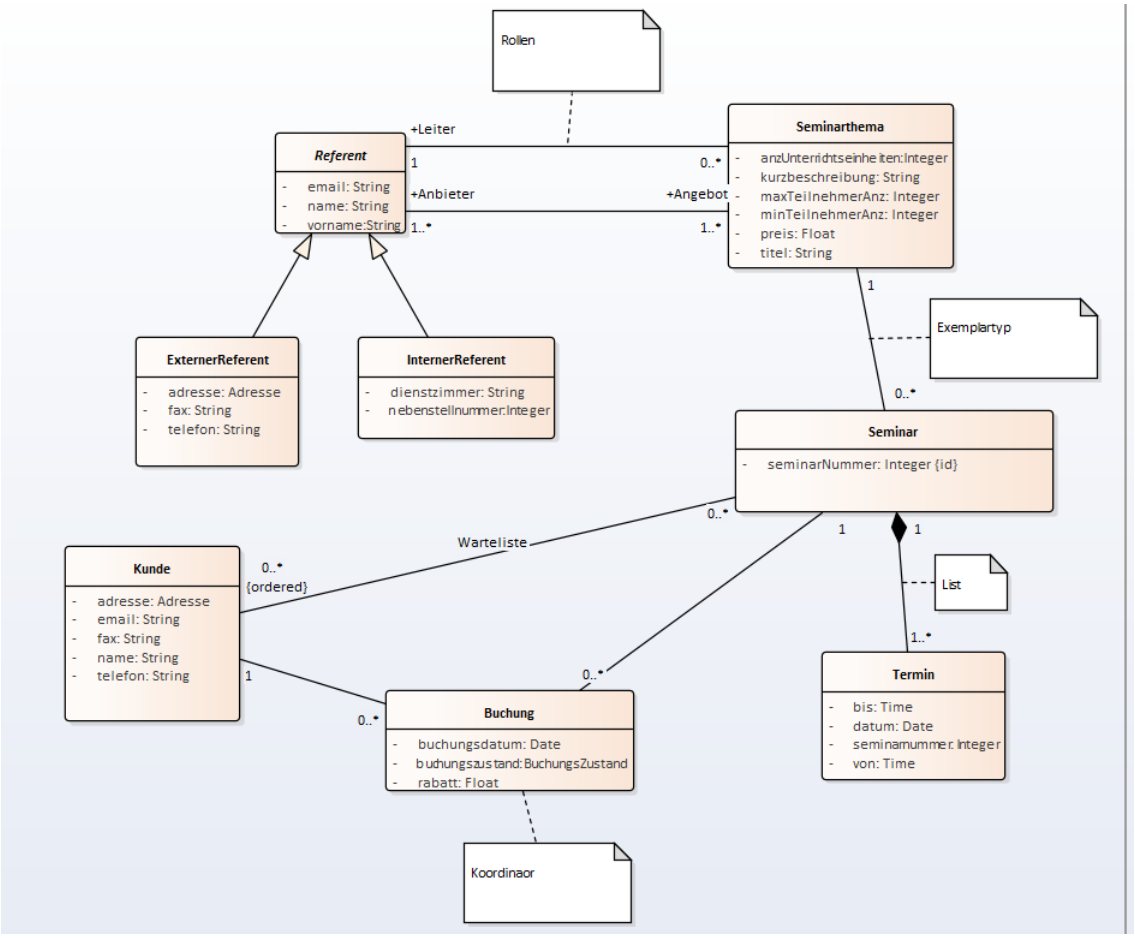

Abbildung 2: OOM Modell

Da Vererbung so nicht in einem ER-Modell möglich ist, haben wir eine Anpassung vorgenommen. Ein Interner/Externer Referent hat jeweils noch einen Verweis auf den Eintrag von Referent. Außerdem haben wir statt einer Klasse Buchung eine Beziehung buchen mit den Attributen aus der Klasse.

0.4.2 c)

Ein `Referent` kann 0 Seminarthemen leiten, da er auch nur Seminare anbieten kann oder beliebig viele da hier keine Beschränkung vorgegeben war. Laut Aufgabe kann ein Seminar von genau einem Referent geleitet werden aber mehrere Anbieter haben. Ein Referent kann gleichzeitig Anbieter und Leiter sein.

Ein Seminarthema kann 0 Seminare haben, um sicherzustellen, dass zur Anmeldung nicht schon ein Seminar eingetragen werden muss. Es kann aber im Laufe mehrere Seminare haben. In einem Seminar kann nur genau ein Seminarthema behandelt werden.

Ein Seminar kann an mehreren Terminen statt finden. Muss aber an mindestens eins. Ein Termin kann nur zu genau einem Seminar gehören.

Kunden können angelegt werden ohne an einem Seminar teilzunehmen oder auf einer Warteliste zu stehen. Deshalb die 0 Kardinalitäten. Im Laufe können sie aber an beliebig vielen Seminaren teilnehmen oder auf Wartelisten stehen. In die andere Richtung gilt genau das gleiche.