Bericht Datenbank Praktikum

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

17. Januar 2020

Inhaltsverzeichnis

1 Aufgabe 1: Entwurf und Implementierung des Datenmodells

1.1 a) ERM

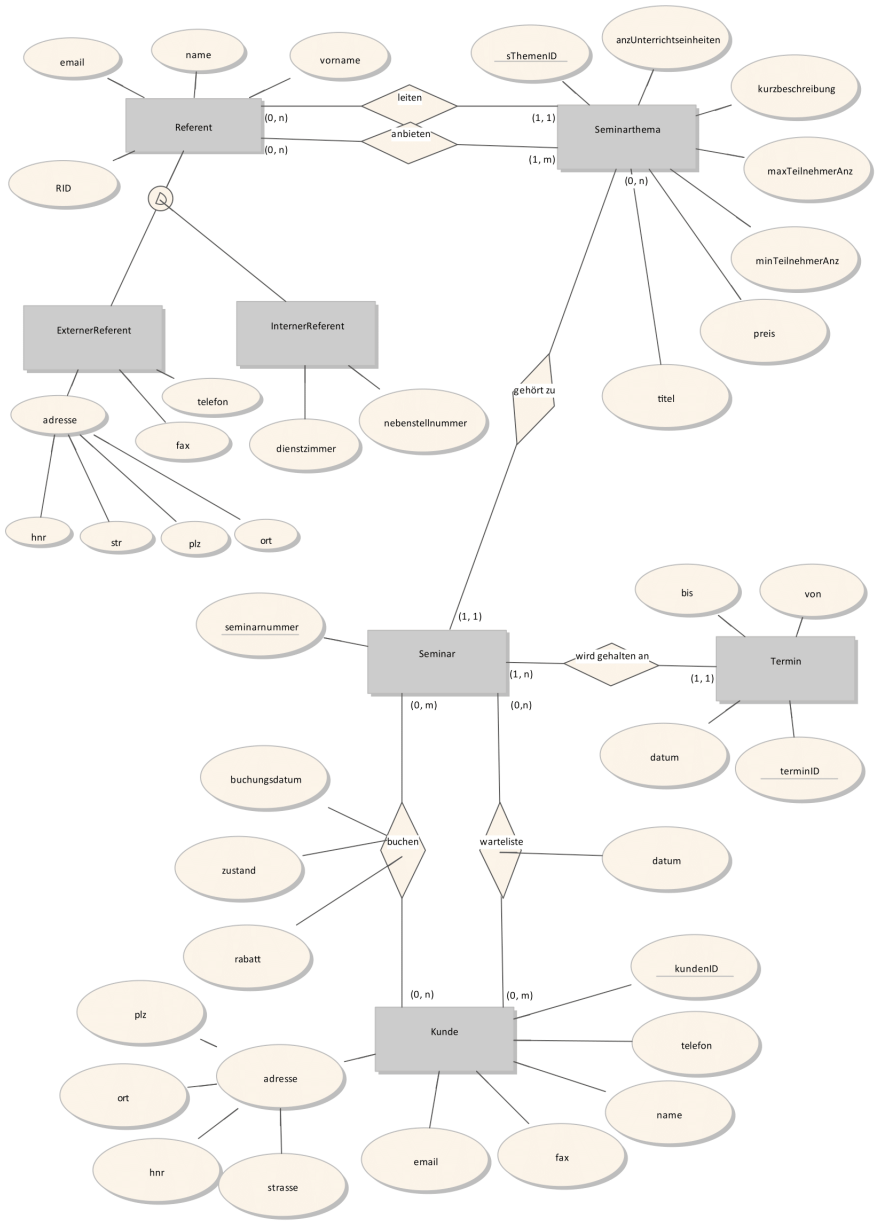

Abbildung 1: ER-Modell Seminarverwaltung

1.1.1 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.

1.1.2 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.

1.1.3 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:Varchar, 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:(strasse:Varchar, hnr:Varchar, plz:Varchar, ort:Varchar)

internerReferent = ({rid:Integer, dienstzimmer:Varchar, nebenstellnummer:Integer})

1.1.4 Relations

leiten = (referent x seminarthema)

anbieten = (referent x seminarthema)

gehört_zu = (seminarthema x seminar)

buchen = (seminar x kunde, buchungsdatum:Date, zustand:Varchar, rabatt:Float)

warteliste = (kunde x seminar, datum:Date)

wird_gehalten_an = (seminar x termin)

1.2 b) Relationales Modell

1.2.1 Relations

referent = (rid:Integer, email:Varchar, name:Varchar, vorname:Varchar)

externerReferent = (rid:Integer, strasse:Varchar, hnr:Varchar, plz:Varchar, ort:Varchar)

internerReferent = (rid:Integer, dienstzimmer:Varchar, nebenstellnummer:Integer)

seminarthame = (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:Varchar, ort:Varchar, hnr:Varchar, str:Varchar)

buchen = (kundenid:Integer, seminarnr:Integer, buchungsdatum:Date, zustand:Varchar, rabatt:Float)

warteliste = (kundenid:Integer, seminarnr:Integer, datum:Date)

1.2.2 Referenzen

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

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),
    datum Date,
    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_InternerReferent(

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

FOREIGN KEY (RID) REFERENCES g8_referent(RID)

);
```

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, dienstnummer, nebenstellennummer) VALUES
    (4, 1, 1);

INSERT INTO g8_externerreferent(rid, fax, telefon, plz, strasse, hnr, ort)
    values
```

```
    ((select rid from g8_referent where name = 'Sugar' and vorname = 'David'), '
       1234', '1234', '73434', 'Aalen', 'Uni-Str', '111'),
    ((select rid from g8_referent where name = 'Sobott' and vorname = 'Julian'),
        '1234', '1234', '73434', 'Aalen', 'Uni-Str', '111'),
    ((select rid from g8_referent where name = 'Mendel' and vorname = 'Lukas'),
       '1234', '1234', '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 = 'Sobott' and vorname = 'Julian'));

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

INSERT INTO g8_termin (von, bis, datum, seminarnummer)
values
('09:30', '13:00', '1999-12-01', 1),
('09:30', '13:00', '1999-12-01', 2),
('09:30', '13:00', '1999-12-01', 3),
('09:30', '13:00', '1999-12-01', 4),
('09:30', '13:00', '1999-12-01', 5),
('10:30', '18:45', '1999-12-01', 1),
('10:30', '18:45', '1999-12-01', 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, '1999-12-01', 'gezahlt', 0.0),
(1, 2, '1999-12-01', 'gezahlt', 0.0),
(2, 1, '1999-12-01', 'berechnet', 0.3),
(2, 3, '1999-12-01', 'gezahlt', 0.0),
(3, 3, '1999-12-01', 'gebucht', 0.0),
(3, 4, '1999-12-01', 'gezahlt', 0.0),
(4, 3, '1999-12-01', 'gebucht', 0.0),
(5, 3, '1999-12-01', 'gebucht', 0.0),
(3, 2, '1999-12-01', 'offen', 0.0),
(1, 3, '1999-12-01', 'berechnet', 0.7);

INSERT INTO g8_warteliste (kundenid, seminarnummer, datum)
values
(1,1,'1999-12-01'),
(2,1,'1999-12-01'),
```

```
(3,2,'1999-12-01');
```

3 Aufgabe 3: SQL-Statements für Datenabfrage

```
-- a) Wie viele Referenten
SELECT count(*) as AnzahlGesamt, count(fax) as AnzahlExtern, count(dienstnummer)
     as AnzahlIntern
FROM g8_referent FULL JOIN g8_externerreferent g8e on g8_referent.rid = g8e.rid
    FULL JOIN g8_internerreferent g8i on g8_referent.rid = g8i.rid;

-- b) Alle Seminare mit Anzahl Seminartermine
SELECT s.seminarnummer, COUNT(s.seminarnummer) as Anzal_Termine
FROM g8_seminar s JOIN g8_termin t on s.seminarnummer = t.seminarnummer
GROUP BY s.seminarnummer;

-- c) Alle Seminare mit Anzahl Teilnehmer
SELECT s.seminarnummer ,count(*) as Anzahl_Teilnehmer
FROM g8_seminar as s join g8_buchen b on s.seminarnummer = b.seminarnummer
GROUP BY s.seminarnummer;

-- d) Seminare mit der längsten Warteliste
SELECT seminarnummer, count(*)
FROM g8_warteliste
GROUP BY seminarnummer
HAVING count(*) >= ALL(
     SELECT count(*)
     FROM g8_seminar s JOIN g8_warteliste g8w on s.seminarnummer = g8w.
         seminarnummer
     GROUP BY s.seminarnummer
     );


-- e) Referenten mit Anzahl Teilnehmer
SELECT name, count(s.seminarnummer)
FROM g8_referent r JOIN g8_seminarthema st on r.rid = st.leiter JOIN g8_seminar
    s on st.sthemaid = s.sthemaid
     join g8_buchen b on s.seminarnummer = b.seminarnummer
GROUP BY r.rid, r.name;


-- f) Referenten mit Einnahmen

SELECT name, count(s.seminarnummer), sum(st.preis)
FROM g8_referent r JOIN g8_seminarthema st on r.rid = st.leiter JOIN g8_seminar
    s on st.sthemaid = s.sthemaid
     join g8_buchen b on s.seminarnummer = b.seminarnummer
GROUP BY r.name;
```