Jasmin Guth / Holger Schwarz
Universität Stuttgart

Übung „Modellierung“ SS 18

Übungsblatt Nr. 4 – Bereich Datenbanken und Anfragesprachen
Thema: SQL, Trigger, logischer Datenbankentwurf

Jedes Übungsblatt besteht aus zwei Teilen:

(1) Übungen, die im Rahmen einer Vortragsübung besprochen werden und
(2) Aufgaben, die in **Zweiergruppen** bearbeitet werden. Die Lösungen müssen fristgerecht als **PDF** (mit vorgegebenem Deckblatt!) abgegeben werden und werden bewertet. Sie erhalten Ihre Abgabe mit Korrekturen zurück.

Termin der Vortragsübung:
- 30.05.2018, 15:45 Uhr in V38.04

Abgabefrist für die Aufgaben: 07.06.2018, 12.00 Uhr

Übung 4.1: Trigger

Nehmen Sie an, dass mit den folgenden SQL-Anweisungen eine relationale Datenbank für die Verwaltung von Mautdaten erstellt und mit entsprechenden Daten gefüllt wurde:

```
CREATE TABLE Eigentuemer (
       ENr                CHAR(2)             NOT NULL,
       Name               VARCHAR(50),
       Wohnort            VARCHAR(50),
       CONSTRAINT pkENr PRIMARY KEY (ENr));

CREATE TABLE PKW (
       PKennz             VARCHAR(15)         NOT NULL,
       Marke              VARCHAR(15),
       ENr                CHAR(2)             NOT NULL,
       CO2Emission        INTEGER,
       ElektroFzg         CHAR(1)             NOT NULL
                                        CHECK(ElektroFzg IN ('0', '1')),
       CONSTRAINT pkPKennz PRIMARY KEY (PKennz),
       CONSTRAINT fkENr FOREIGN KEY (ENr) REFERENCES Eigentuemer (ENr)
       ON UPDATE RESTRICT ON DELETE CASCADE);

CREATE TABLE Mautstellen (
       MNr                CHAR(2)             NOT NULL,
       Autobahn           VARCHAR(4),
       Preis              INTEGER             NOT NULL,
       Internetbasiert    CHAR(1)             NOT NULL
                                      CHECK(Internetbasiert IN ('0', '1')),
       CONSTRAINT pkMNr PRIMARY KEY (MNr));

CREATE TABLE Mautdaten (
       MNr                CHAR(2)             NOT NULL,
       PKennz             VARCHAR(15)         NOT NULL,
       Tag                DATE                NOT NULL,
       Zeit               TIME                NOT NULL,
       CONSTRAINT pkMautdaten PRIMARY KEY (MNr, PKennz, Tag, Zeit),
       CONSTRAINT fkMNr FOREIGN KEY (MNr) REFERENCES Mautstellen (MNr)
       ON UPDATE RESTRICT ON DELETE RESTRICT,
       CONSTRAINT fkPKennz FOREIGN KEY (PKennz) REFERENCES PKW (PKennz)
       ON UPDATE RESTRICT ON DELETE RESTRICT);
```

Gegeben ist die zugehörige Datenbank mit folgenden Einträgen:

Eigentuemer:

	ENr	Name	Wohnort
E1	MB	Moritz	Hamburg
E2	JL	Jan Josef	Berlin
E3	HM	Heike	Duesseldorf
E4	FP	Franka	Muenchen
E5	AR	Armin Rohde	Muenster

PKW:

	PKennz	Marke	ENr	CO2Emission	ElektroFzg
P1	HH-MB-18	BMW	MB	139	0
P2	B-JL-321	Daimler	JL	183	0
P3	D-HM-127	Porsche	HM	298	0
P4	M-FP-817	Tesla	FP	96	1

Mautstellen:

	MNr	Autobahn	Preis	Internetbasiert
MS1	18	A7	20	0
MS2	24	A1	23	1
MS3	28	A8	21	0
MS4	37	A14	18	0

Mautdaten:

	MNr	PKennz	Tag	Zeit
MD1	18	HH-MB-18	2001-12-17	11:31
MD2	37	B-JL-321	2002-02-12	08:12
MD3	37	B-JL-321	2002-03-01	17:14
MD4	24	D-HM-127	2005-10-23	04:55

Die kursiv gedruckten Kürzel links neben den Tabellenzeilen, gehören nicht zum Tabelleninhalt, sondern sollen Ihnen beim Verweis auf bestimmte Tupel Schreibarbeit ersparen.

Zusätzlich werden die Trigger T1 und T2 in der folgenden Reihenfolge erstellt:

```
CREATE TRIGGER T1
BEFORE INSERT ON PKW
REFERENCING NEW as new
FOR EACH ROW
WHEN (new.ElektroFzg = '0'
      AND (SELECT COUNT(PKennz)
        FROM PKW p
        WHERE
            p.ENr = new.ENr
            AND p.ElektroFzg = '0') >= 1
)
SIGNAL SQLSTATE '70001' ('Aus Umweltschutzgründen darf pro
Eigentümer maximal ein Nicht-Elektro-Fahrzeug zugelassen sein.');


CREATE TRIGGER T2
BEFORE INSERT ON Mautdaten
REFERENCING NEW AS new
FOR EACH ROW
WHEN ((SELECT ElektroFzg
        FROM PKW p
        WHERE p.PKennz = new.PKennz) = '1')
SIGNAL SQLSTATE '10I00' ('Subventionierte Fahrt für E-Fahrzeuge!');
```

Die SQL-Anweisungen in den folgenden Teilaufgaben sind entweder zulässig oder nicht zulässig. Zulässige Anweisungen werden ohne Fehlermeldung von einem DBMS ausgeführt, bei nicht zulässigen Anweisungen wird die Ausführung mit einer Fehlermeldung abgebrochen und die Datenbank auf den Zustand vor Ausführung der Anweisung zurückgesetzt. Gehen Sie wie folgt vor:

- Gehen Sie bei **jeder** Teilaufgabe vom oben angegebenen, ursprünglichen Inhalt der Tabellen aus!
- Kreuzen Sie an, ob es sich um eine zulässige oder um eine nicht zulässige Anweisung handelt. Ein korrekt gesetztes Kreuz ohne nachfolgende Erläuterung oder mit falscher Erläuterung wird mit 0 Punkten bewertet!
- Geben Sie für die nicht zulässigen Anweisungen an, welcher Constraint oder Trigger die Ausführung verhindert.
- Geben Sie für die zulässigen Anweisungen an, welche Trigger-Aktionen ausgeführt werden und welche Tabelleninhalte sich wie ändern; z.B. “Tupel ('GG', Goetz George', 'Berlin') wird in Tabelle Eigentuemer eingefügt” oder „In Tupel MS2 wird der Preis auf '25' gesetzt“.

a)

```
DELETE FROM Mautstellen WHERE MNr = 28;
```

□ zulässig □ nicht zulässig

---

b)

```
INSERT INTO Eigentuemer VALUES ('JL', 'Juergen von der Lippe',
'Aachen');
```

□ zulässig □ nicht zulässig

---

c)

```
INSERT INTO PKW VALUES ('M-FP-178', 'Audi', 'FP', 208, '0');
```

□ zulässig □ nicht zulässig

---

d)

```
INSERT INTO Mautdaten VALUES ('37', 'M-FP-817', '2013-09-11', '16:35')
```

□ zulässig □ nicht zulässig

---

e)

```
DELETE FROM Eigentuemer WHERE Enr = 'MB'
```

□ zulässig □ nicht zulässig

---

f) Definieren Sie einen zusätzlichen Trigger **T3**, der beim Eintragen eines neuen Elektrofahrzeuges überprüft, ob die CO2-Emmisonen des neue Fahrzeugs niedriger sind als bei allen Fahrzeugen ohne Elektroantrieb. Trifft dies nicht zu, soll das Einfügen des neuen Eintrages verhindert werden und eine entsprechende Meldung ausgegeben werden.

Übung 4.2: Funktionale Abhängigkeiten und Armstrong-Axiome

Gegeben sei die Menge X funktionaler Abhängigkeiten:

$X = \{ AB \rightarrow C, FE \rightarrow B, E \rightarrow D, BD \rightarrow A, H \rightarrow F, GH \rightarrow E, GHE \rightarrow D, BD \rightarrow C \}$

Zeigen Sie mit Hilfe der Armstrong-Axiome: $X \models GH \rightarrow C$

Übung 4.3: Redundante funktionale Abhängigkeiten

Gegeben sei die Menge X funktionaler Abhängigkeiten:

$X = \{ A \rightarrow B, C \rightarrow I, B \rightarrow F, EF \rightarrow G, D \rightarrow E, BC \rightarrow D, G \rightarrow H, AC \rightarrow H \}$

Prüfen Sie mit Hilfe des CLOSURE-Algorithmus, ob es sich bei AC → H um eine redundante funktionale Abhängigkeit handelt. Begründen Sie Ihre Antwort und geben Sie den Aufruf des Algorithmus sowie sämtliche Zwischenschritte an!

Aufgabe 4.1: SQL und Relationenalgebra [18 Punkte]

Nehmen Sie an, dass mit den folgenden SQL-Anweisungen eine relationale Datenbank für die Daten von Bankkunden erstellt und mit entsprechenden Daten gefüllt wurde:

```
CREATE TABLE adresse (
    a_id             INTEGER           NOT NULL,
    a_strasse        VARCHAR(250)      NOT NULL,
    a_ort            VARCHAR(250)      NOT NULL,
    a_plz            INTEGER           NOT NULL,
    CONSTRAINT a_adresse_pk PRIMARY KEY(a_id)
);

CREATE TABLE kunde (
    k_id             INTEGER           NOT NULL,
    k_name           VARCHAR(250)      NOT NULL,
    k_kredite        INTEGER,
    k_alter          INTEGER,
    k_adresse        INTEGER,
    CONSTRAINT k_kunde_pk PRIMARY KEY(k_id),
    CONSTRAINT k_adresse_fk FOREIGN KEY(k_adresse)
       REFERENCES adresse(a_id) ON DELETE SET NULL,
    CONSTRAINT k_kredite_grenzen CHECK (k_kredite > -1 AND k_kredite < 3)
);

CREATE TABLE mitarbeiter (
    m_id             INTEGER           NOT NULL,
    m_name           VARCHAR(250)      NOT NULL,
    m_alter          INTEGER,
    m_adresse        INTEGER,
    CONSTRAINT m_mitarbeiter_pk PRIMARY KEY(m_id),
    CONSTRAINT m_adresse_fk FOREIGN KEY(m_adresse)
       REFERENCES adresse(a_id) ON DELETE SET NULL
);


CREATE TABLE bankkonto (
    b_id             INTEGER           NOT NULL,
    b_guthaben       INTEGER           NOT NULL,
    b_inhaber        INTEGER,
    b_kontaktperson  INTEGER,
    CONSTRAINT b_bankkonto_pk PRIMARY KEY(b_id),
    CONSTRAINT b_inhaber_fk FOREIGN KEY(b_inhaber)
       REFERENCES kunde(k_id) ON DELETE SET NULL
    CONSTRAINT b_kontaktperson_fk FOREIGN KEY(b_kontaktperson)
       REFERENCES mitarbeiter(m_id) ON DELETE RESTRICT

);


CREATE TABLE darlehen (
    d_id             INTEGER           NOT NULL,
    d_datum          VARCHAR(100)      NOT NULL,
    d_kredithoehe    INTEGER           NOT NULL,
    d_bankkonto      INTEGER,
    d_kunde          INTEGER           NOT NULL,
    CONSTRAINT d_darlehen_pk PRIMARY KEY(d_id),
    CONSTRAINT d_bankkonto_fk FOREIGN KEY(d_bankkonto)
       REFERENCES bankkonto(b_id) ON DELETE SET NULL,
    CONSTRAINT d_kunde_fk FOREIGN KEY(d_kunde)
       REFERENCES kunde(k_id) ON DELETE CASCADE
);
```

- Verwenden Sie in Ihren Lösungen zu den folgenden Aufgaben die aus der Vorlesung und Übung bekannte Notation (SQL bzw. relationale Algebra).
- Überflüssige Joins, Prädikate und Attribute können zu Punktabzug führen!
- Alle Schlüsselwörter sowie Tabellen- und Attributbezeichner sind auszuschreiben!

a) Geben Sie einen Ausdruck in **relationaler Algebra** an, der für alle Darlehen, deren Kredithoehe über 5000 liegt, das jeweilige Datum ausgibt.

b) Geben Sie einen Ausdruck in **relationaler Algebra** an, der die IDs aller Bankkonten ausgibt, denen ein Darlehen mit einer Kredithoehe von über 20000 zugeordnet ist und deren Kontaktperson in Berlin ansässig ist.

c) Geben Sie eine Anfrage in **SQL-Notation** an, die von allen Kunden den Namen, das Alter, die Anzahl der Kredite und die Postleitzahl ausgibt, absteigend sortiert nach dem Alter.

d) Geben Sie eine Anfrage in **SQL-Notation** an, die für jeden Mitarbeiter den Namen und die Anzahl der Bankkonten, bei denen er als Kontaktperson fungiert und deren Guthaben niedriger als 5000 ist, ausgibt. Es sollen nur Mitarbeiter ausgegeben werden, die insgesamt bei weniger als 50 Bankkonten Kontaktpersonen sind.

e) Geben Sie eine Anfrage in **SQL-Notation** an, welche den Namen und das Alter aller Kunden ausgibt, die mindestens ein Darlehen haben, dessen Kredithoehe über der Durchschnittskredithoehe aller Darlehen liegt. Dabei soll kein Kunde doppelt ausgegeben werden.

f) Geben Sie eine **SQL-Anweisung** an, die in allen Adressen, in denen die Ortsangabe das Wort 'Stuttgart' enthält, den Ort auf den Wert 'Stuttgart' setzt. Diese Änderung soll unabhängig davon erfolgen, was vor oder hinter dem Wort 'Stuttgart' steht. Beispielsweise sollen folgenden Varianten in 'Stuttgart' geändert werden: Stuttgart-Vaihingen, S-Stuttgart.

Aufgabe 4.2: Trigger und Constraints [16 Punkte]

Gegeben ist die Datenbank aus **Aufgabe 4.1** mit folgenden Einträgen:

adresse:

a_id	a_strasse	a_ort	a_plz
101	'Nobelstraße 19'	'Stuttgart'	70569
102	'Vikarweg 8'	'Stuttgart'	70567

kunde:

k_id	k_name	k_kredite	k_alter	k_adresse
201	'Klein'	0	26	NULL
202	'Krüger'	1	NULL	NULL

mitarbeiter:

m_id	m_name	m_alter	m_adresse
301	'Groß'	35	101
302	'Sommer'	42	102

bankkonto:

b_id	b_guthaben	b_inhaber	b_kontaktperson
401	10000	201	302
402	20000	202	302

darlehen:

d_id	d_datum	d_kredithoehe	d_bankkonto	d_kunde
501	'03.02.2017'	5000	401	201
502	'04.02.2017'	10000	402	202
503	'05.02.2017'	5000	401	201

Zusätzlich wird folgender Trigger T1 erstellt:

```
CREATE TRIGGER T1
      AFTER INSERT ON darlehen
      REFERENCING NEW ROW AS r
      FOR EACH ROW
      UPDATE bankkonto
      SET b_guthaben=b_guthaben + r.d_kredithoehe
      WHERE b_inhaber=r.d_kunde;
```

Die SQL-Anweisungen in den folgenden Teilaufgaben sind entweder zulässig oder nicht zulässig. Zulässige Anweisungen werden ohne Fehlermeldung von einem DBMS ausgeführt. Bei nicht zulässigen Anweisungen wird die Ausführung mit einer Fehlermeldung abgebrochen und die Datenbank auf den Zustand vor Ausführung der Anweisung zurückgesetzt. Gehen Sie wie folgt vor:

- Gehen Sie bei **jeder** Teilaufgabe vom oben angegebenen, **ursprünglichen Inhalt** der Tabellen aus!
- Kreuzen Sie an, ob es sich um eine zulässige oder um eine nicht zulässige Anweisung handelt. Ein korrekt gesetztes Kreuz **ohne nachfolgende**

Erläuterung oder mit **falscher Erläuterung** wird mit **0 Punkten** bewertet!

Anzugebende Erläuterungen:

- Geben Sie für die **nicht zulässigen** Anweisungen an, welcher **Constraint** oder **Trigger** die Ausführung verhindert.
- Geben Sie für die **zulässigen** Anweisungen an, welche **Trigger ausgelöst werden** und welche **Tabelleninhalte** sich wie ändern; z.B. “Tupel (103, Universitatsstr 38, Stuttgart, 70569) wird in Tabelle adresse eingefügt”. Falls diese Änderungen durch **Fremdschlüssel** bewirkt werden geben Sie auch die entsprechenden Fremdschlüssel an.

a)

```
INSERT INTO kunde VALUES (203, 'Fritz', 0, 27, 103);
```

b)

```
INSERT INTO kunde VALUES (201, 'Fritz', 0, 27, 101);
```

c)

```
DELETE FROM adresse WHERE a_plz=70569
```

d)

```
INSERT INTO darlehen VALUES (504, '05.02.2017', 10000, 401, 201);
```

e)

```
DELETE FROM darlehen WHERE d_id=501
```

f)

```
DELETE FROM kunde WHERE k_id=201
```

g)

```
DELETE FROM mitarbeiter WHERE alter>40
```

h)

```
DELETE FROM mitarbeiter WHERE alter<40
```

Aufgabe 4.3: Armstrong-Axiome **[7 Punkte]**

Zeigen Sie mit Hilfe der Armstrong-Axiome (reflexivity, augmentation, transitivity), dass die folgenden abgeleiteten Regeln ebenfalls gelten:

- If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds (**union**)
- If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds (**decomposition**)
- If $\alpha \rightarrow \beta$ holds and $\beta\gamma \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds (**pseudotransitivity**)

Geben Sie für jeden Schritt an, welche funktionalen Abhängigkeiten (gegebene bzw. bereits abgeleitete) und welches Axiom Sie verwenden.

Aufgabe 4.4: Redundante funktionale Abhängigkeiten **[9 Punkte]**

Gegeben ist das Relationenschema **T** (A, B, C, D, E, F, G, H) sowie die Menge funktionaler Abhängigkeiten

Z = { AC → GH, B → E, D → A, F → CD, F → G, H → B }.

Prüfen Sie jeweils mit Hilfe des CLOSURE-Algorithmus, ob es sich bei F → G und H → B um redundante funktionale Abhängigkeiten handelt. Begründen Sie Ihre Antwort und geben Sie die jeweiligen Aufrufe des Algorithmus sowie sämtliche Zwischenschritte an!