

Pflichtmodul Informationssysteme (SS 2018)

Prof. Dr. Jens Teubner

Leitung der Übungen: Thomas Lindemann, Tanja Bock

Übungsblatt Nr. 14 (Themenübersicht)

Ausgabe: 11.07.2018

Abgabe: entfällt

Dieses Übungsblatt bildet eine Sammlung von Aufgaben, die teilweise aus alten Klausuren entnommen wurden. Es dient zur Selbstkontrolle und soll einen Überblick über einige der über das Semester hinweg behandelten Themen verschaffen. Diese Themenübersicht sollte auf keinen Fall als Musterklausur betrachtet werden und repräsentiert nicht die komplette Menge klausurrelevanter Themen.

Aufgabe 1 (ER-Modellierung)

In einem Schulungszentrum werden verschiedene Kurse angeboten, wobei ausgewählte Kurse den Inhalt anderer Kurse voraussetzen. Je Kurs gibt es genau eine Lehrkraft, die auch verschiedene Kurse betreuen kann.

1. Erstellen Sie für diesen Diskursbereich ein ER-Diagramm. Verwenden Sie für die Funktionalität der Beziehungstypen die Min-/Max-Notation.
2. Überführen Sie das erhaltene ER-Diagramm in Tabellen des Relationalen Modells und geben Sie Primär- (*primary*) und Fremdschlüsselbedingungen (*foreign keys*) an. Überlegen Sie sich dazu geeignete Attributnamen.

Aufgabe 2 (Tabellen erzeugen)

Ein Sportverein will eine Datenbank mit folgendem Schema aufbauen:

$sch(Mitglieder) = (\underline{MitglNr}, Name, Alter, Abteilung)$

$sch(Beiträge) = (\underline{Alter}, Betrag)$

$sch(Zahlungen) = (\underline{Datum}, \underline{MitglNr}, Betrag)$

In *Beiträge* werden die Mitgliedsbeiträge festgehalten, die vom Alter des Mitglieds abhängig sind. Manche Mitglieder bezahlen ihre Beiträge in Raten, daher sind in *Zahlungen* alle bislang eingegangenen Beträge vermerkt.

1. Geben Sie die SQL-Anweisungen an, mit denen das obige Relationen-Schema erzeugt wird. Spezifizieren Sie dabei sämtliche Primär- und Fremdschlüsselbeziehungen.
2. Trainern soll die Möglichkeit gegeben werden, von allen Mitgliedern *Name* und *Alter* abzufragen. Geben Sie SQL-Kommandos an, die zur Erstellung einer View, die genau diese Information enthält, nötig sind.

Aufgabe 3 (Anfragesprachen)

Gegeben sei das Relationen-Schema aus Aufgabe 2:

$$sch(Mitglieder) = (\underline{MitglNr}, Name, Alter, Abteilung)$$
$$sch(Beiträge) = (\underline{Alter}, Betrag)$$
$$sch(Zahlungen) = (\underline{Datum}, \underline{MitglNr}, Betrag)$$

Formulieren Sie die folgende Anfrage in **Tupel-Relationen-Kalkül** (*tuple relational calculus*) und **SQL**. Verwenden Sie **DISTINCT** genau dann, wenn im Ergebnis tatsächlich Duplikate auftreten können.

1. Geben Sie *Name*, *MitglNr* und *Betrag* der fälligen Beiträge aller Mitglieder aus der Abteilung 'Fußball' aus.

Formulieren Sie die folgenden beiden Anfragen in **Relationen-Algebra** und **SQL**. Verwenden Sie auch hier **DISTINCT** genau dann, wenn im Ergebnis tatsächlich Duplikate auftreten können.

2. Welche Mitglieder haben bislang noch gar nichts bezahlt? Geben Sie *MitglNr*, *Name*, *Alter* und den fälligen *Betrag* aus.
3. Geben Sie von den ältesten Mitgliedern *MitglNr* und *Name* aus.

Formulieren Sie die folgenden zwei Anfragen nur in **SQL**:

4. Welche *Abteilung* hat die meisten Mitglieder, die bisher noch nichts gezahlt haben?
5. Geben Sie für alle Mitglieder, die noch nicht vollständig bezahlt (unter Einbeziehung aller bereits geleisteten Zahlungen) haben, *Name* und den noch verbleibenden Betrag aus.

Aufgabe 4 (Schemanormalisierung)

Gegeben seien das Relationenschema

$$sch(R) = (VWXYZ)$$

sowie die zugehörige Menge

$$\mathcal{F} = \{Z \rightarrow X, \quad V \rightarrow W, \quad X \rightarrow YV, \quad W \rightarrow V\}$$

von funktionalen Abhängigkeiten.

1. Geben sie einen Schlüssel für $sch(R)$ an!
2. **Zerlegen Sie R mit Hilfe des BCNF-Zerlegungsalgorithmus (*BCNF decomposition algorithm*) aus der Vorlesung, sodass alle resultierenden Tabellen in BCNF vorliegen.**
 - ▷ Geben Sie vor jedem Durchlauf der **while**-Schleife sowie nach Ablauf des Algorithmus die Schemata aller erzeugten Tabellen an. Listen Sie dabei auch alle zugehörigen Funktionalen Abhängigkeiten (*functional dependencies*) auf.

Aufgabe 5 (Transaktionskontrolle)

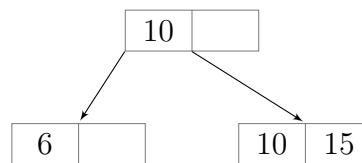
Untersuchen Sie die folgenden Schedules auf Serialisierbarkeit. Geben Sie dazu den vollständigen Abhängigkeitsgraphen (*conflict graph* / *serialization graph*) an und beschriften Sie die Kanten mit allen vorkommenden Konflikten. Geben Sie, falls möglich, einen äquivalenten seriellen Schedule an.

- (a) $\langle r_2(x), r_3(y), w_2(x), r_4(x), w_3(z), r_1(z), w_1(z), r_4(z), w_4(x), r_2(y), w_4(z), w_2(y) \rangle$
- (b) $\langle r_3(u), r_2(v), w_3(u), r_2(u), r_1(w), w_2(v), r_1(v), r_3(u), r_3(w), r_2(u), w_3(w), w_2(u) \rangle$

Aufgabe 6 (B+ Indexstrukturen)

Fahrer			
<u>fahrer_id</u>	name	team	nation
15	Antonio Pizzonia	Jaguar-Racing	Brasilien
6	Kimi Räikkönen	McLaren-Mercedes	Finnland
10	Heinz-Harald Frentzen	Sauber-Petronas	Deutschland
3	Juan Pablo Montoya	Williams-BMW	Kolumbien
9	Nick Heidfeld	Sauber-Petronas	Deutschland
2	Rubens Baricello	Ferrari	Brasilien
4	Ralf Schumacher	Williams-BMW	Deutschland
17	Jacques Villeneuve	BAR-Honda	Kanada

Der Betreiber einer Formel-Eins-Informationssseite im Internet will die Daten aller Fahrer erfassen. Ein Praktikant gibt die obige Liste in das Datenbanksystem ein. Das Datenbanksystem verwendet zur Indizierung des Attributs `fahrer_id` einen B+-Baum der Ordnung 1 (2 Schlüsselwerte je Knoten; Fan-out: 3). Nachdem der Praktikant die ersten drei Tupel eingegeben hat, sieht der B+-Baum wie folgt aus:



Der Praktikant gibt nun die restlichen Tupel (beginnend mit 'Juan Pablo Montoya') in die Datenbank in der obigen Reihenfolge von oben nach unten ein.

1. Wie sieht der B+-Baum nach jedem einzelnen Schritt aus wenn die Tupel weiter eingegeben werden?

Zur Bearbeitung der Aufgabe ist die folgende Konvention zu wählen: Kommt es zum Split eines Knoten, verhält sich der B+-Baum wie folgt: Die jeweils größere "Hälfte" wandert in den rechten Knoten. Redistribution kennt das Datenbanksystem nicht.

2. Schätzen Sie für B+- und Hash-Index den Aufwand in Anzahl Seitenzugriffen ab, der für eine große Anzahl Datensätze n für das Auffinden eines Tupels notwendig ist. Wie groß ist der Aufwand im Regelfall, wie können sich ungünstig verteilte Daten auswirken?
3. Beim erstmaligen Laden von großen Tabellen ("Bulk Loading") liegen die Quelldaten oft bereits vorsortiert vor. Welche Nachteile entstehen, wenn beim Ladevorgang der normale Einfüge-Algorithmus verwendet wird? Wie könnte man einen Bulk-Loading-Algorithmus konzipieren, der diese Nachteile behebt (und natürlich nur mit vorsortierten Daten funktioniert)?