Datenbanken
Übungsblatt 6

Aufgabe 1 (Daten Modifizieren)

Realisieren Sie folgende Aufgaben in university.db. Geben Sie je genau ein SQL statement an.

1. Erhöhe das Gehalt jedes Computer-Science-Instruktors um 10%.
2. Lösche alle Kurse die nie angeboten wurden.
3. Stelle alle Studierenden mit mehr als 100 credits als Instruktor im selben Department mit einem Gehalt von 30'000 ein.

Aufgabe 2 (Verständnis von Abfragen)

Gegeben ist wieder das Schema von `publications.db` vom letzten Übungsblatt.

1. Wieso ergibt folgende Abfrage nicht wie erwartet alle Titel die weniger als 20 Dollar kosten? Was liefert die Abfrage stattdessen?

```
select title from titles
where price < 20;
```

2. Wieso ergibt folgende Abfrage nicht wie erwartet die Autoren zusammen mit den Verlegern, bei denen sie publiziert haben? Was liefert die Abfrage stattdessen? Korrigieren Sie die Abfrage.

```
select au_lname, pub_name
from authors natural join titleauthor natural join titles
natural join publishers;
```

3. Wieso ergibt folgende Abfrage nicht wie erwartet alle Verleger, die höchstens zwei Psychologiebücher verlegt haben? Was liefert die Abfrage stattdessen? Korrigieren Sie die Abfrage.

```
select pub_id, count(title_id) as numtitles
from titles
where type like 'psychology%'
group by pub_id
having numtitles <= 2;
```

Aufgabe 3 (Verständnis von Abfragen)

Gegeben sei folgendes Schema:

person(name, street, city)

purchase(name, id, number_of_items)
name → person
id → product

```
product(id, supplier_name, description, price)
supplier_name → supplier
supplier(name, street, city)
```

Beschreiben Sie umgangssprachlich das Resultat folgender Abfragen.

1. ```
   select name from person natural join purchase;
   ```

2. ```
   select name from person natural join purchase
                            natural join product
                            natural join supplier;
   ```

3. ```
   select supplier_name, avg(price)
   from product
   group by supplier_name;
   ```

4. ```
   select supplier_name, avg(price)
   from product
   where id in (select id from purchase)
   group by supplier_name;
   ```

5. ```
   select sum(price * number_of_items)
   from person natural join purchase natural join product
   where name = "Hans Muster";
   ```

6. ```
   select id from product
   except
   select id from purchase
   where name = "Hans Muster";
   ```

Aufgabe 4 (Effiziente Join-Algorithmen)

Rufen Sie sich folgende Konzepte wieder in Erinnerung:

- den Algorithmus für binäre Suche
- die Zeitkomplexität eines Algorithmus (die Funktion, die aus der Größe der Eingabe die Laufzeit im worst-case ermittelt)

Gegeben seien Relationen $r(A, B)$ und $s(B, C)$, als Listen von Tupeln. Attribut B ist vom Typ Integer. Gehen Sie von dem in der letzten Übung entwickelten Algorithmus für den natürlichen Join aus, und verfeinern Sie ihn für die folgenden Fälle in einen *möglichst effizienten* Algorithmus, um den natural join der beiden Relationen zu berechnen. Geben sie jeweils die Zeitkomplexität des Algorithmus an.

1. $\{B\}$ ist ein Superschlüssel für r,

2. wie 1. und zusätzlich ist die Liste für r nach Attribut B sortiert,
3. wie 2. und zusätzlich ist die Liste für s nach Attribut B sortiert.

Hinweis: Bei einem Tupel `t` können Sie mit `t.B` auf den Wert des Attributs B zugreifen.