TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Christoph Anneser (anneser@in.tum.de)
http://db.in.tum.de/teaching/ss21/ei2/

Blatt Nr. 11

Dieses Blatt wird am Montag, den 5. Juli 2021 besprochen.

Tool zum Üben der relationalen Algebra: http://www-db.in.tum.de/~muehe/ira/.
SQL-Schnittstelle: http://hyper-db.com/interface.html.

Aufgabe 1: Hörer-Dividende

Was bringt der Vorlesungsbesuch? Finden Sie heraus, ob es für Prüfungen von Vorteil ist, die jeweiligen Vorlesungen auch gehört zu haben. Ermitteln Sie dazu die Durchschnittsnote der Prüfungen, zu denen die Studenten die Vorlesungen nicht gehört haben und die Durchschnittsnote der Prüfungen, zu denen sie die Vorlesungen gehört haben.

Lösung:

Wir bestimmen, wie sich das Verhältnis der Prüfungsleistungen von gehörten zu nicht gehörten Vorlesungen insgesamt darstellt.

```
    with nichtgehoert as
    (select avg(note) as notenichtgehoert
     from pruefenxl p
     where not exists(select *
                      from hoeren h
                      where h.vorlnr = p.vorlnr
                        and h.matrnr = p.matrnr
         )
    ),
    gehoert as
    (select avg(note) as notegehoert
     from pruefenxl p
     where exists(select *
                  from hoeren h
                  where h.vorlnr = p.vorlnr
                    and h.matrnr = p.matrnr
                )
    )
select *
from nichtgehoert,
     gehoert;
```

Da beide Views aus jeweils nur einem Wert bestehen, ergibt sich das Resultat der Anfrage als Kreuzprodukt.

Aufgabe 2: Relationenalgebra I

Beantworten Sie mittels relationaler Algebra.

(a) Geben Sie einen Ausdruck an, der die Relation $\neg hoeren$ erzeugt. Diese enthält für jeden Studenten und jede Vorlesung, die der Student **nicht** hört einen Eintrag mit Matrikelnummer und Vorlesungsnummer.

$$(\Pi_{MatrNr} Studenten \times \Pi_{VorlNr} Vorlesungen) - hoeren$$

(b) Finden Sie alle Studenten, die keine Vorlesung hören. Geben Sie dabei zwei verschiedene Lösungen an.

$$Studenten \triangleright hoeren$$

oder

$$Studenten - (Studenten \ltimes hoeren)$$

(c) Finden Sie die *Studenten*, die *Vorlesungen* hören (bzw. gehört haben), für die ihnen die direkten Voraussetzungen fehlen. Wir konstruieren eine hypothetische Ausprägung der Relation *hören*, die gelten müsste, wenn alle Studenten alle benötigten Vorgängervorlesungen hören. Von dieser Menge ziehen wir die tatsächliche Ausprägung von hören ab, so dass diejenigen Einträge übrig bleiben, bei denen ein Student die Vorgängervorlesung nicht hört (bzw. gehört hat).

$$\begin{aligned} R \;:=\; & (\rho_{\text{VorlNr}\leftarrow\text{Vorgänger}}(\Pi_{\text{MatrNr,Vorgänger}}(\text{hören} \bowtie_{\text{VorlNr=Nachfolger}} \text{voraussetzen})) \\ & -\text{hören}) \bowtie \text{Studenten} \end{aligned}$$

Abbildung 1 zeigt den zugehörigen Operatorbaum.

Aufgabe 3: Gewichtete Durchschnittsnote

Falls Sie ihre Anfrage sinnvoll testen möchten, sollten sie sich selber auf einer lokalen Datenbank eine größere *prüfen*-Relation anlegen. Das Schema und die Ausprägung, welche online verwendet wird, finden Sie auf der Website.

Bestimmen Sie für alle Studenten eine gewichtete Durchschnittsnote ihrer Prüfungen. Die Gewichtung der einzelnen Prüfungen erfolgt gemäß dem Vorlesungsumfang (SWS). Dies entspricht dem Verfahren der Durchschnittsnotenberechnung für Ihr Bachelor-Zeugnis.

Lösung

Hier sollen die Noten der Studenten gewichtet werden, d.h. dass Noten für Prüfungen über kurze Vorlesungen (z.B. 2 SWS) abgewertet werden, da der Lernaufwand dafür geringer war als für Prüfungen über lange Vorlesungen, die dadurch aufgewertet werden.

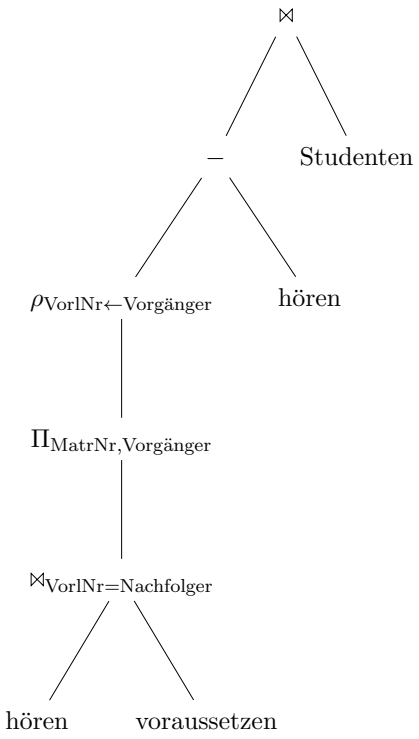

Abbildung 1: Operatorbaum

```
select s.name, sum(p.note*v.sws)/sum(v.sws) as durchschnittsnote
from pruefenxl p, vorlesungen v, studenten s
where p.vorlnr = v.vorlnr and s.matrnr = p.matrnr
group by s.name;
```

Aufgabe 4: Erst hören, dann prüfen

Welche Studenten haben alle Vorlesungen, die sie haben prüfen lassen, auch tatsächlich vorher gehört?

Lösung

Die Anforderung, dass die Studenten im Anfrage-Ergebnis alle Vorlesungen, die sie haben prüfen lassen auch tatsächlich gehört haben, lässt sich umschreiben zu: „Es darf keine Vorlesung geben, die geprüft wurde, zu der es aber keinen Eintrag in *hoeren* gibt.“

```
select s.*
from studenten s
```

```
where not exists (select *
from pruefen p
where s.matrnr = p.matrnr
and not exists (select *
    from  hoeren h
    where h.matrnr = s.matrnr
    and h.vorlnr = p.vorlnr
    )
);
```

Aufgabe 5: SQL - Outer Joins

Hinweis: Aufgabe 2 stellt eine gute Möglichkeit dar SQL zu üben. Vorsicht: Eine der beiden Möglichkeiten von 2 (b) benötigt einen outer join. Dieser wurde in SQL in der Vorlesung noch nicht besprochen.

Lösung

(a) Die Anfrage kann ähnlich wie in der Relationenalgebra formuliert werden:

```
select s.matrNr, v.vorlNr
from Vorlesungen v, Studenten s
where not exists(select *
                 from hoeren h
                 where h.vorlNr = v.vorlNr
                   and h.matrNr = s.matrNr)
```

(b) Die eine Möglichkeit ist analog zur Relationenalgebra:

```
select *
from Studenten s
where not exists(select *
                 from hoeren h
                 where h.matrNr = s.matrNr);
```

Die andere basiert auf einem **outer join**, da es keinen **anti join** in SQL gibt:

```
select s.*
from Studenten s left outer join hoeren h on s.matrNr = h.matrnr
where h.matrNr is null;
```

(c) Finden Sie die *Studenten*, die *Vorlesungen* hören (bzw. gehört haben), für die ihnen die direkten Voraussetzungen fehlen.

```
select distinct s.*
from hoeren h1,
     voraussetzen v,
     Studenten s
where h1.vorlNr = v.nachfolger
  and s.matrNr = h1.matrNr
  and not exists(select *
                 from hoeren h2
                 where h2.matrNr = h1.matrNr
                   and h2.vorlNr = v.vorgaenger);
```