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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS20/21

Christoph Anneser, Josef Schmeißer, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2021/grundlagen/

Blatt Nr. 04

Hausaufgabe 1

Finden Sie alle *Studenten*, die alle *Vorlesungen* gehört haben, die von Sokrates gelesen wurden. Formulieren Sie die Anfrage

- in der Relationenalgebra,
- im relationalen Tupelkalkül und
- im relationalen Domänenkalkül.

Lösung:

Formulierung in relationaler Algebra

1. Wir ermitteln die von Sokrates gelesenen *Vorlesungen* (nur die Vorlesungsnummern in diesem Fall):

$$V \quad := \quad \Pi_{\text{VorlNr}}(\text{Vorlesungen} \bowtie_{\text{gelesenVon = PersNr}} (\sigma_{\text{Name='Sokrates'}}(\text{Professoren})))$$

2. Durch Anwendung des Divisionsoperators bekommen wir die Matrikelnummern der Studenten, die alle von Sokrates gehaltenen *Vorlesungen* gehört haben:

$$M \quad := \quad \text{hören} \div V$$

   Wichtig: Für die relationale Division muss sichergestellt sein, dass das Schema von V eine Teilmenge des Schemas von *hören* ist: $\text{sch}(V) \subset \text{sch}(hren)$

3. Aus *Studenten* ermitteln wir zusätzlich die Namen der Studenten:

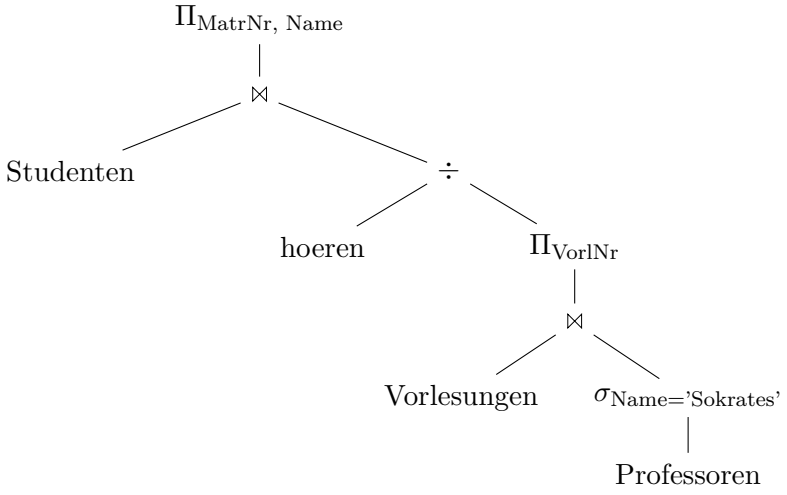
$$S \quad := \quad \Pi_{\text{MatrNr, Name}}(\text{Studenten} \bowtie M)$$

Zusammengefasst ergibt sich dann folgender Ausdruck:

$$\Pi_{\text{MatrNr, Name}}(\text{Studenten} \bowtie (\text{hören} \div \Pi_{\text{VorlNr}}(\text{Vorlesungen} \bowtie_{\text{gelesenVon = PersNr}} (\sigma_{\text{Name='Sokrates'}}(\text{Professoren})))))$$

Hinweis: Die Zusammenfassung ist optional. Relationale Ausdrücke können wie oben gezeigt *modularisiert* werden indem Teilausdrücke (und die dadurch definierten Relationen) an Variablen zugewiesen werden.

In Operatorbaumdarstellung:



Formulierung im Tupelkalkül

1. Wir ermitteln mittels des Allquantors alle Vorlesungen, die von Sokrates gelesen werden.
2. Wir fordern, dass ein Student alle diese Vorlesungen gehört hat (Existenzquantor).

$$\{s \mid s \in \text{Studenten} \wedge \forall\ v \in \text{Vorlesungen}$$
$$(\exists\ p \in \text{Professoren}(v.\text{gelesenVon} = p.\text{PersNr} \wedge p.\text{Name} = \text{'Sokrates'})$$
$$\Rightarrow \exists\ h \in \text{hören}(s.\text{MatrNr} = h.\text{MatrNr} \wedge v.\text{VorlNr} = h.\text{VorlNr}))\}$$

Formulierung im Domänenkalkül

Das Vorgehen ist analog zu dem beim relationalen Tupelkalkül.

$$\{[m,n] \mid \exists\ s([m,n,s] \in \text{Studenten}$$
$$\wedge \forall\ v,t,sws,g($$
$$([v,t,sws,g] \in \text{Vorlesungen} \wedge \exists\ rg,ra([g,\text{'Sokrates'},rg,ra] \in \text{Professoren}))$$
$$\Rightarrow$$
$$([m,v] \in \text{hören})$$
$$)$$
$$)\}$$

Hausaufgabe 2

Formulieren Sie die folgenden Anfragen auf dem bekannten Universitätsschema in SQL:

a) Finden Sie alle Studenten, die drei oder mehr Vorlesungen hören

b) Finden Sie alle Grundlagenvorlesungen. Eine Vorlesung ist eine Grundlagenvorlesung, wenn sie mindestens 4 SWS hat und Voraussetzung von mindestens zwei anderen Vorlesungen ist.

c) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören.

d) Bestimmen Sie, wie viele Vorlesungen im Schnitt pro Student gehört werden. Beachten Sie, dass Studenten, die keine Vorlesung hören, in das Ergebnis einfließen müssen.

Lösung:

a) Finden Sie alle Studenten, die drei oder mehr Vorlesungen hören

```
select s.matrnr, s.name
from studenten s, hoeren h
where s.matrnr = h.matrnr
group by s.matrnr, s.name
having count(*) >= 3
```

b) Finden Sie alle Grundlagenvorlesungen. Eine Vorlesung ist eine Grundlagenvorlesung, wenn sie mindestens 4 SWS hat und Voraussetzung von mindestens zwei anderen Vorlesungen ist.

```
select v.vorlnr, v.titel
from vorlesungen v, voraussetzen vor
where
    v.vorlnr = vor.vorgaenger and
    v.sws >= 4
group by v.vorlnr, v.titel
having count(*) >= 2
```

c) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören.

```
with
vorlesungen_von_sokrates as (
  select *
  from vorlesungen v, professoren p
  where v.gelesenVon = p.persnr and p.name = 'Sokrates'
),
studenten_von_sokrates as (
  select *
  from studenten s
  where exists (
    select *
    from hoeren h, vorlesungen_von_sokrates v
    where h.matrnr = s.matrnr and v.vorlnr = h.vorlnr)
)
select avg(semester) from studenten_von_sokrates
```

Man beachte, dass die Formulierung mittels `WHERE EXISTS` für die Elimination von Duplikaten sorgt, d.h. ein Student, der 3 Vorlesungen von Sokrates hört kommt nur einmal in `studenten_von_sokrates vor`, was gewünscht ist. Alternativ kann man `studenten_von_sokrates` formulieren als:

```
select DISTINCT s.*
from studenten s, hoeren h, vorlesungen_von_sokrates v
where h.matrnr = s.matrnr and v.vorlnr = h.vorlnr
```

d) Bestimmen Sie, wie viele Vorlesungen im Schnitt pro Student gehört werden. Beachten Sie, dass Studenten, die keine Vorlesung hören, in das Ergebnis einfließen müssen.

Eine Möglichkeit, Studenten, die keine Vorlesung hören, in das Ergebnis einzubeziehen, ist ein äußerer Join. Dabei muss im ersten `count` ein beliebiges Attribut aus der Relation hören stehen, damit `NULL`-Werte nicht mitgezählt werden:

```
select count(h.vorlnr) * 1.000 / count(distinct s.matrnr)
from
   studenten s left outer join
   hoeren h on s.matrnr = h.matrnr
```

Andererseits ergibt sich die Lösung auch aus der Anzahl der Tupel in hören geteilt durch die Anzahl der Studenten:

```
select hcount/(scount*1.000)
from (select count(*) as hcount from hoeren) h,
     (select count(*) as scount from studenten) s
```

Hausaufgabe 3

Folgender Ausdruck im Tupelkalkül gibt alle Studenten aus, die alle von ihnen gehörten Vorlesungen bestanden haben.

$$\{s \mid s \in \text{Studenten} \wedge \\ \forall h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \Rightarrow \\ \exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

Übersetzen Sie diese Anfrage nun in SQL. Da SQL keine Allquantoren und Implikationen unterstützt, müssen Sie sie dazu zunächst umformen.

a) Formen Sie den Ausdruck in einen Äquivalenten um, der keine Implikationen oder Allquantoren verwendet.

b) Übersetzen Sie den so erlangten Ausdruck in SQL. Testen Sie ihn in der Webschnittstelle.

Lösung:

a) Wir formen zunächst die innere Implikation um, denn $A \Rightarrow B \iff \neg A \vee B$:

$$\{s \mid s \in \text{Studenten} \wedge \\ \forall h \in \text{hoeren}(h.\text{MatrNr} \neq s.\text{MatrNr} \vee \\ \exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

Wir ersetzen nun den Allquantor durch einen negierten Existenzquantor, denn $\forall x(P(x)) \iff \neg\exists x(\neg P(x))$:

$$\{s \mid s \in \text{Studenten} \wedge \\ \neg\exists h \in \text{hoeren}(\neg(h.\text{MatrNr} \neq s.\text{MatrNr} \vee \\ \exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4)))\}$$

Zuletzt wenden wir De Morgans Regel an, um die Negation nach innen zu ziehen, denn $\neg(A \vee B) \iff \neg A \wedge \neg B$:

$$\{s \mid s \in \text{Studenten} \wedge \\ \neg\exists h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \wedge \\ \neg\exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

b) Die Anfrage kann nun eins zu eins in SQL übersetzt werden, wobei jeder logische Operator einfach durch seine SQL-Entsprechung ersetzt wird:

```
select * from Studenten s
where not exists (select * from hoeren h
    where h.MatrNr = s.MatrNr
    and not exists (select * from pruefen p
        where p.MatrNr = s.MatrNr
        and p.VorlNr = h.VorlNr
        and p.Note <= 4
    )
)
```

Hausaufgabe 4

Gegeben sei die folgende Relation `ZehnkampfD` mit Athletennamen und den von ihnen erreichten Punkten in den jeweiligen Zehnkampfdisziplinen:

$$\text{ZehnkampfD} : \{\underline{\text{Name}, \text{Disziplin}}, \text{Punkte}\}$$

Name	Disziplin	Punkte
Eaton	100 m	450
Eaton	Speerwurf	420
...	...	...
Eaton	Weitsprung	420
Suarez	100 m	850
Suarez	Speerwurf	620
...	...	...

Finden Sie alle ZehnkämpferInnen, die in *allen* Disziplinen besser sind als der Athlet mit dem Namen *Bolt*. Formulieren Sie die Anfrage in SQL

a) mit korrelierter Unteranfrage

b) basierend auf Zählen

Sie dürfen davon ausgehen, dass jeder Sportler in jeder Disziplin angetreten ist.

Laden Sie zum Testen entweder die SQL-Datei von der Übungswebseite in ein lokal installiertes Datenbanksystem oder verwenden Sie die Webschnittstelle.

Lösung:

Mit korrelierter Unteranfrage:

```
select distinct a.Name
from ZehnkampfD as a
where not exists (
  select *
  from ZehnkampfD as a2
  where a2.Name = a.Name
    and exists (
      select *
      from ZehnkampfD as b
      where b.Disziplin = a2.Disziplin
        and b.Name = 'Bolt'
        and b.Punkte >= a2.Punkte
    )
)
```

Basierend auf Zählen

```
with besserAlsBolt(name, disziplin) as (
  select a.name, a.disziplin
  from zehnkampfd a, zehnkampfd b
  where b.name = 'Bolt'
    and a.disziplin = b.disziplin
    and a.punkte > b.punkte
),
disziplinen(anzahl) as (
  select count(distinct disziplin) as anzahl
  from zehnkampfd
)
select name
from besserAlsBolt
group by name
having count(*) = (select anzahl from disziplinen)
```