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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS23/24

Christoph Anneser, Michael Jungmair, Stefan Lehner, Moritz Sichert, Lukas Vogel
(gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2324/grundlagen/

Blatt Nr. 05

Hausaufgabe 1

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

a) Bestimmen Sie das durchschnittliche Semester der Studenten der Universität.

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

c) 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) Bestimmen Sie das durchschnittliche Semester der Studenten der Universität.

```
select avg(semester*1.0) from studenten;
```

b) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören. Beachten Sie, dass Sie das Semester von Studenten, die mehr als eine Vorlesung bei Sokrates hören, nicht doppelt zählen dürfen.

```
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
```

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

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

select hcount/(cast (scount as decimal(10,4)))
from (select count(*) as hcount from hoeren) h,
     (select count(*) as scount from studenten) s
```

Hausaufgabe 2

„Fleißige Studenten“: Formulieren Sie eine SQL-Anfrage, um die Studenten zu ermitteln, die mehr SWS belegt haben als der Durchschnitt. Berücksichtigen Sie dabei auch Totalverweigerer, die gar keine Vorlesungen hören.

Lösung:

Folgende SQL-Anfrage ermittelt die fleißigen Studenten:

```
select s.*
from Studenten s
where s.MatrNr in
  (select h.MatrNr
   from hoeren h join Vorlesungen v on h.VorlNr = v.VorlNr
   group by h.MatrNr
   having sum(SWS) >
      (select sum(cast(SWS as decimal(5,2)))/count(distinct(s2.MatrNr))
       from Studenten s2
          left outer join hoeren h2 on h2.MatrNr = s2.MatrNr
          left outer join Vorlesungen v2 on v2.VorlNr = h2.VorlNr));
```

Durch die Verwendung von **with** und **case** wird die Anfrage übersichtlicher:

```
with GesamtSWS as (
  select sum(cast(SWS as decimal(5,2))) as AnzSWS
  from hoeren h2, Vorlesungen v2
  where v2.VorlNr = h2.VorlNr
),
GesamtStudenten as (
  select count(MatrNr) as AnzStudenten
  from Studenten
)
select s.*
from Studenten s
where s.MatrNr in (
  select h.MatrNr
  from hoeren h join Vorlesungen v on h.VorlNr = v.VorlNr
  group by h.MatrNr
  having sum(SWS) > (select AnzSWS / AnzStudenten
                     from GesamtSWS, GesamtStudenten));
```

Alternativ:

```
with SWSProStudent as (
select s.MatrNr,
  cast((case when sum(v.SWS) is null
             then 0 else sum(v.SWS)
        end) as real) as AnzSWS
 from Studenten s
   left outer join hoeren h on s.MatrNr = h.MatrNr
   left outer join Vorlesungen v on h.VorlNr = v.VorlNr
 group by s.MatrNr
)

select s.*
from Studenten s
where s.MatrNr in (select sws.MatrNr
                   from SWSProStudent sws
                   where sws.AnzSWS > (select avg(AnzSWS)
                                       from SWSProStudent));
```

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 (erweiterte) 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
Bolt	100m	50
Bolt	Weitsprung	50
Eaton	100m	40
Eaton	Weitsprung	60
Suarez	100m	60
Suarez	Weitsprung	60
Behrenbruch	100m	30
Behrenbruch	Weitsprung	50
...	...	...

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

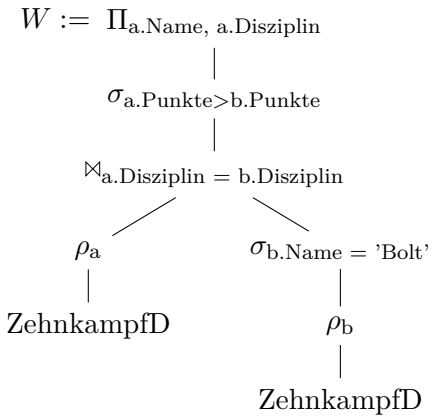
- in der relationalen Algebra,
- im relationalen Tupelkalkül,
- im relationalen Domänenkalkül und
- in SQL.

HINWEIS: Sie können ihre Anfrage auf der Relation `ZehnkampfD` auch in der SQL-Webschnittstelle testen.

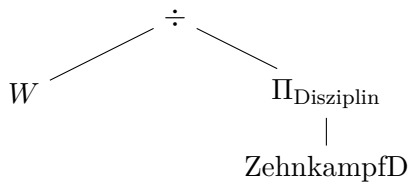
Lösung:

Formulierung in relationaler Algebra

1. Wir ermitteln zunächst alle Wertungen W der Athleten, die eine höhere Punktzahl als *Bolt* erreicht haben:



2. Durch Anwendung des Divisionsoperators bekommen wir diejenigen Athleten, die in *allen* Disziplinen einen höhere Wertung als Bolt haben:



Formulierung im Tupelkalkül

$$
\begin{aligned}
&\{[\text{a.Name}] \mid \text{a} \in \text{ZehnkampfD} \wedge \\
&\quad \forall \text{a}' \in \text{ZehnkampfD}(\text{a}'.\text{Name} = \text{a.Name} \\
&\qquad \Rightarrow \\
&\qquad \neg\exists \text{b} \in \text{ZehnkampfD}(\text{b.Disziplin} = \text{a}'.\text{Disziplin} \wedge \text{b.Name} = \text{'Bolt'} \wedge \\
&\qquad\quad \text{b.Punkte} \geq \text{a}'.\text{Punkte}) \\
&\quad)\}
\end{aligned}
$$

Formulierung im Domänenkalkül

$$
\begin{aligned}
&\{[\text{a}] \mid \exists \text{d,p}\ ([\text{a,d,p}] \in \text{ZehnkampfD} \wedge \\
&\quad \forall \text{d}', \text{p}'\ ([\text{a}, \text{d}', \text{p}'] \in \text{ZehnkampfD} \\
&\qquad \Rightarrow \\
&\qquad \neg\exists \text{bp}([\text{'Bolt'},\text{d}',\text{bp}] \in \text{ZehnkampfD} \wedge \text{bp} \geq \text{p}') \\
&\quad) \\
&)\}
\end{aligned}
$$

Formulierung in SQL

Da SQL auf dem Tupelkalkül basiert, kann der oben stehende Ausdruck nahezu 1:1 in SQL übersetzt werden. Da SQL allerdings über keine Implikation und über keinen Allquantor verfügt, müssen diese zunächst ersetzt werden. Dazu verwenden wir die beiden Äquivalenzen (i) $a \Rightarrow b \equiv \neg a \vee b$ um Implikationen zu entfernen und (ii) $\forall x(P(x)) \equiv \neg\exists x(\neg P(x))$ um Allquantoren durch Existenzquantoren zu ersetzen.

Im obigen Ausdruck muss also

$$
\begin{aligned}
&\text{a}'.\text{Name} = \text{a.Name} \Rightarrow \\
&\neg\exists \text{b} \in \text{ZehnkampfD}(\text{b.Disziplin} = \text{a}'.\text{Disziplin} \wedge \text{b.Name} = \text{'Bolt'} \wedge \\
&\qquad\qquad \text{b.Punkte} \geq \text{a}'.\text{Punkte})
\end{aligned}
$$

umgeformt werden.

Wir entfernen zunächst die Implikation mithilfe der Äquivalenz $a \Rightarrow b \equiv \neg a \vee b$ und erhalten:

$$
\begin{aligned}
&\text{a}'.\text{Name} \neq \text{a.Name} \vee \\
&\neg\exists \text{b} \in \text{ZehnkampfD}(\text{b.Disziplin} = \text{a}'.\text{Disziplin} \wedge \text{b.Name} = \text{'Bolt'} \wedge \\
&\qquad\qquad \text{b.Punkte} \geq \text{a}'.\text{Punkte})
\end{aligned}
$$

Da der Ausdruck allquantifiziert ist, wird dieser gemäß (ii) negiert:

$$\begin{aligned}&\mathrm{a}'.\mathrm{Name} = \mathrm{a.Name} \wedge \\ &\exists \mathrm{b} \in \mathrm{ZehnkampfD}(\mathrm{b.Disziplin} = \mathrm{a}'.\mathrm{Disziplin} \wedge \mathrm{b.Name} = \text{'Bolt'} \wedge \\ &\qquad\qquad \mathrm{b.Punkte} \geq \mathrm{a}'.\mathrm{Punkte})\end{aligned}$$

Der vollständige Ausdruck ohne Allquantoren und Implikationen ist dann:

$$\begin{aligned}&\{[\mathrm{a.Name}] \mid \mathrm{a} \in \mathrm{ZehnkampfD} \wedge \\ &\qquad \neg\exists \mathrm{a}' \in \mathrm{ZehnkampfD}(\mathrm{a}'.\mathrm{Name} = \mathrm{a.Name} \wedge \\ &\qquad\qquad \exists \mathrm{b} \in \mathrm{ZehnkampfD}(\mathrm{b.Disziplin} = \mathrm{a}'.\mathrm{Disziplin} \wedge \mathrm{b.Name} = \text{'Bolt'} \wedge \\ &\qquad\qquad\qquad\qquad \mathrm{b.Punkte} \geq \mathrm{a}'.\mathrm{Punkte}) \\ &\qquad)\}\end{aligned}$$

Übersetzt in SQL ergibt sich:

```
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
    )
)
```

Aufgrund der Multimengensemantik von SQL ist die explizite Angabe von `distinct` erforderlich um Duplikate zu eliminieren.

Alternative Formulierung in SQL 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)
```