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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS15/16

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1516/grundlagen/

Blatt Nr. 05

Tool zum Üben von SQL-Anfragen: `http://hyper-db.com/interface.html`.

Hausaufgabe 1

Gegeben seien die beiden Relationen $R : \{[a_1, \ldots, a_n]\}$ und $S : \{[b_1, \ldots, b_m]\}$. Geben Sie die folgenden Ausdrücke im Tupel- und Domänenkalkül an:

a) $Q_1 := R \bowtie_{a_1=b_1} S$

b) $Q_2 := R \rtimes_{a_1=b_1} S$

c) $Q_3 := R \rhd_{a_1=b_1} S$

d) $Q_4 := R \lhd_{a_1=b_1} S$

Lösung:

Bitte beachten Sie, dass in dieser Aufgabe ausschließlich allgemeine *Theta*joins ($\bowtie_\Theta, \rtimes_\Theta, \ldots$) verwendet werden. Gemäß Definition werden somit alle Attribute der beiden Eingaberelationen in die Ausgaberelation projiziert, einschließlich der Attribute, welche in der Joinbedingung enthalten sind. Lediglich bei *natürlichen* Joins, wo implizit eine Gleichheitsbedingung für alle gleichnamigen Attribute erfüllt sein muss, werden gleichnamige Attribute nicht doppelt in die Ausgaberelation projiziert. Siehe hierzu auch Folie Kapitel 3, "Andere Join-Arten".

a) $Q_1 := R \bowtie_{a_1=b_1} S$

Formulierung im Tupelkalkül

$$Q_1 := \{[r.a_1, \ldots, r.a_n, s.b_1, \ldots, s.b_m] \mid r \in R \ \wedge \ s \in S \ \wedge \ r.a_1 = s.b_1\}$$

Da der Joinoperator Tupel aus verschiedenen Relationen verbindet, müssen für die Ergebnismenge neue Tupel mithilfe des Tupelkonstruktors konstruiert werden: $[\text{attribut}_1 : \text{wert}_1, \ldots, \text{attribut}_n : \text{wert}_n]$. Die oben verwendete Tupelkonstruktion $[r.a_1, \ldots]$ ist eine verkürzte Schreibweise für $[a_1 : r.a_1, \ldots]$ und kann verwendet werden, wenn der Attibutname im neuen Tupel unverändert bleibt.

Im Falle des Thetajoins kann auch die Tupelkonkatenation $t_1 \circ t_2$ verwendet werden:

$$Q_1 := \{r \circ s \mid r \in R \ \wedge \ s \in S \ \wedge \ r.a_1 = s.b_1\}$$

Formulierung im Domänenkalkül

$$Q_1 := \{[a_1, \ldots, a_n, b_1, \ldots, b_m] \mid [a_1, \ldots, a_n] \in R \wedge [b_1, \ldots, b_m] \in S \wedge a_1 = b_1\}$$

oder

$$Q_1 := \{[a_1, \ldots, a_n, b_1 : a_1, b_2, \ldots, b_m] \mid [a_1, \ldots, a_n] \in R \wedge [a_1, b_2, \ldots, b_m] \in S\}$$

b) $Q_2 := R \bowtie_{a_1=b_1} S$

Formulierung im Tupelkalkül

$$Q_2 := Q_1 \cup \{[r.a_1, \ldots, r.a_n, b_1 : \text{null}, \ldots, b_m : \text{null}] \mid \ r \in R \wedge \nexists\ s \in S(r.a_1 = s.b_1)\}$$

Formulierung im Domänenkalkül

$$Q_2 := Q_1 \cup \{[a_1, \ldots, a_n, b_1 : \text{null}, \ldots, b_m : \text{null}] \mid [a_1, \ldots, a_n] \in R \wedge \nexists\ c_2, \ldots, c_m([a_1, c_2, \ldots, c_m] \in S)\}$$

c) $Q_3 := R \rtimes_{a_1=b_1} S$

Formulierung im Tupelkalkül

$$Q_3 := \{s \mid \ s \in S\ \wedge \exists\ r \in R(r.a_1 = s.b_1)\}$$

Formulierung im Domänenkalkül

$$Q_3 := \{[b_1, \ldots, b_m] \mid [b_1, \ldots, b_m] \in S \wedge \exists\ a_2, \ldots, a_n([b_1, a_2, \ldots, a_n] \in R)\}$$

d) $Q_4 := R \ltimes_{a_1=b_1} S$

Formulierung im Tupelkalkül

$$Q_4 := \{s \mid \ s \in S \wedge \nexists\ r \in R(r.a_1 = s.b_1)\}$$

Formulierung im Domänenkalkül

$$Q_4 := \{[b_1, \ldots, b_m] \mid [b_1, \ldots, b_m] \in S \wedge \nexists\, a_2, \ldots, a_n([b_1, a_2, \ldots, a_n] \in R)\}$$

Hausaufgabe 2

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 3

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

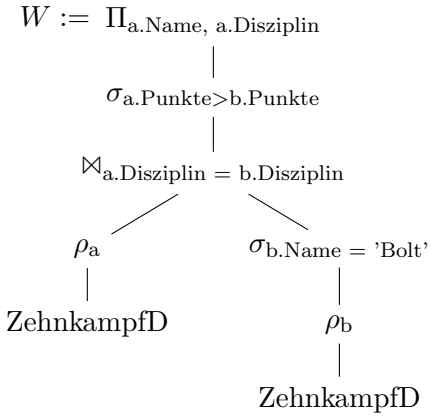
HINWEIS: Beachten Sie, dass die Relation `ZehnkampfD` in der SQL-Webschnittstelle nicht existiert. Verwenden Sie die folgende Syntax um eine temporäre Relationenausprägung zu erzeugen:

```
with zehnkampfd(name,disziplin,punkte) as (
  values
   ('Bolt', '100m', 50),
   ('Bolt', 'Weitsprung', 50),
   ('Eaton', '100m', 40),
   ('Eaton', 'Weitsprung', 60),
   ('Suarez', '100m', 60 ),
   ('Suarez', 'Weitsprung', 60),
   ('Behrenbruch', '100m', 30),
   ('Behrenbruch', 'Weitsprung', 50)
)
select * from zehnkampfd order by disziplin, punkte desc
```

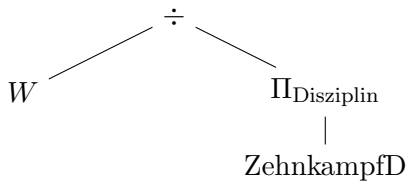
Lösung:

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

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

Formulierung im Domänenkalkül

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

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:

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

Der vollständige Ausdruck ohne Allquantoren und Implikationen ist dann:

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

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)
```