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

# Übung zur Vorlesung *Grundlagen: Datenbanken* im WS17/18

Harald Lang, Linnea Passing (gdb@in.tum.de)
http://www-db.in.tum.de/teaching/ws1718/grundlagen/

**Blatt Nr. 05**

Tool zum Üben der relationalen Algebra:
`http://db.in.tum.de/people/sites/muehe/ira/`

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

## Hausaufgabe 1

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL:

(a) Finden Sie die *Studenten*, die Sokrates aus *Vorlesung(en)* kennen.

(b) Finden Sie die *Studenten*, die *Vorlesungen* hören, die auch Fichte hört.

(c) Finden Sie die *Assistenten* von *Professoren*, die den Studenten Fichte unterrichtet haben – z.B. als potentielle Betreuer seiner Diplomarbeit.

(d) Geben Sie die Namen der *Professoren* an, die Xenokrates aus *Vorlesungen* kennt.

(e) Welche *Vorlesungen* werden von *Studenten* im Grundstudium (1.-4. Semester) gehört? Geben Sie die Titel dieser *Vorlesungen* an.

**Lösung:**

(a) Finden Sie die *Studenten*, die Sokrates aus *Vorlesung(en)* kennen.

```
select s.Name, s.MatrNr
from Studenten s, hoeren h, Vorlesungen v,
   Professoren p
where s.MatrNr = h.MatrNr
  and h.VorlNr = v.VorlNr
  and v.gelesenVon = p.PersNr
  and p.Name ='Sokrates';
```

`DISTINCT` wäre nett, um Duplikate zu unterdrücken ist aber nicht explizit in der Aufgabe gefordert.

(b) Finden Sie die *Studenten*, die *Vorlesungen* hören, die auch Fichte hört.

```
select distinct s1.Name, s1.MatrNr
from Studenten s1, Studenten s2, hoeren h1, hoeren h2
where s1.MatrNr = h1.MatrNr
  and s1.MatrNr != s2.MatrNr
  and s2.MatrNr = h2.MatrNr
  and h1.VorlNr = h2.VorlNr
  and s2.Name ='Fichte';
```

(c) Finden Sie die *Assistenten* von *Professoren*, die den Studenten Fichte unterrichtet haben – z.B. als potentielle Betreuer seiner Diplomarbeit.

```
select a.Name, a.PersNr
from Assistenten a, Professoren p, Vorlesungen v,
   hoeren h, Studenten s
where a.Boss = p.PersNr
  and p.PersNr = v.gelesenVon
  and v.VorlNr = h.VorlNr
  and h.MatrNr = s.MatrNr
  and s.Name ='Fichte';
```

(d) Geben Sie die Namen der *Professoren* an, die Xenokrates aus *Vorlesungen* kennt.

```
select p.PersNr, p.Name
from Professoren p, hoeren h, Vorlesungen v,
   Studenten s
where p.PersNr = v.gelesenVon
and v.VorlNr = h.VorlNr
and h.MatrNr = s.MatrNr
and s.Name ='Xenokrates';
```

(e) Welche *Vorlesungen* werden von *Studenten* im Grundstudium (1.-4. Semester) gehört? Geben Sie die Titel dieser *Vorlesungen* an.

```
select v.Titel
from Vorlesungen v, hoeren h, Studenten s
where v.VorlNr = h.VorlNr
and h.MatrNr = s.MatrNr
and s.Semester between 1 and 4;
```

## 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

„Bekanntheitsgrad“: Formulieren Sie eine SQL-Anfrage, um den Bekanntheitsgrad von Studenten zu ermitteln. Gehen Sie dabei davon aus, dass Studenten sich aus gemeinsam besuchten Vorlesungen kennen. Sortieren Sie das Ergebnis absteigend nach Bekanntheitsgrad!

**Lösung:**
Zunächst definieren wir eine View, die für jeden Studenten alle seine Bekannten auflistet. Anschließend müssen wir diese Bekannten nur noch zählen, um den Bekanntheitsgrad der Studenten zu ermitteln.

```
with Bekannte as (
  select distinct h1.MatrNr as Student, h2.MatrNr as
     Bekannter
  from hoeren h1, hoeren h2
  where h1.VorlNr = h2.VorlNr
    and h2.MatrNr <> h1.MatrNr
)
select s.MatrNr, s.Name, count(*) as AnzBekannter
from Studenten s, Bekannte b
where s.MatrNr = b.Student
group by s.MatrNr, s.Name
order by AnzBekannter desc;
```

Ohne View sieht die Anfrage entsprechend komplexer aus:

```
select s.MatrNr, s.Name, count(*) as AnzBekannter
from Studenten s,
  (select distinct h1.MatrNr as Student, h2.MatrNr as
     Bekannter
   from hoeren h1, hoeren h2
   where h1.VorlNr = h2.VorlNr
     and h2.MatrNr <> h1.MatrNr
   ) b
where s.MatrNr = b.Student
group by s.MatrNr, s.Name
order by AnzBekannter desc;
```

## 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 |
|---|---|---|
| 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.

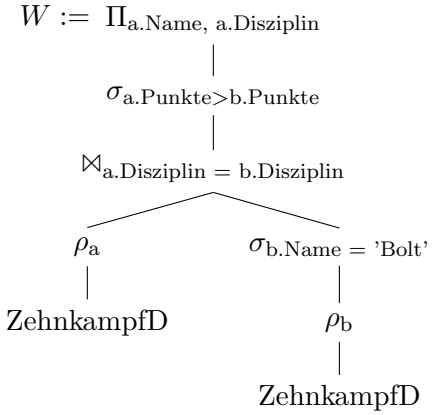
HINWEIS: Beachten Sie, dass die Relation `ZehnkampfD` in der SQL-Webschnittstelle nicht existiert. Verwenden Sie die folgende Syntax um eine temporäre Relationenausprägung zu erzeugen:

```
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
```

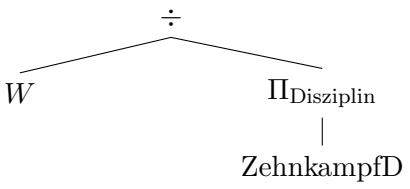
**Lösung:**

### 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**

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

**Formulierung im Domänenkalkül**

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

**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:

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

Der vollständige Ausdruck ohne Allquantoren und Implikationen ist dann:

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

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)
```