TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Dr. Thomas Neumann

# Übung zur Vorlesung *Grundlagen: Datenbanken* im WS14/15

Harald Lang (harald.lang@in.tum.de)
http://www-db.in.tum.de/teaching/ws1415/grundlagen/

**Blatt Nr. 5**

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

**Mailen Sie Ihre Lösung VOR der Übung an Ihren Tutor, damit Sie diese nicht komplett abtippen müssen, falls Sie in der Übung drankommen.**

## 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.

(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

„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!

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 3

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

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));
```