



## 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. 06

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

### Hausaufgabe 1

„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 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

Gegeben sei ein erweitertes Universitätschema mit den folgenden zusätzlichen Relationen *StudentenGF* und *ProfessorenF*:

```
StudentenGF : {[MatrNr : integer, Name : varchar(20), Semester : integer,
                Geschlecht : char, FakName : varchar(20)]}
ProfessorenF : {[PersNr : integer, Name : varchar(20), Rang : char(2),
                 Raum : integer, FakName : varchar(20)]}
```

Die erweiterten Tabellen sind auch in der Webschnittstelle angelegt.

- (a) Ermitteln Sie den Männeranteil an den verschiedenen Fakultäten in SQL!
- (b) Ermitteln Sie in SQL die Studenten, die alle Vorlesungen ihrer Fakultät hören. Geben Sie zwei Lösungen an, höchstens eine davon darf auf Abzählen basieren.

#### Lösung:

```
(a) with
    FakTotal as (
        select FakName, count(*) as Total
        from StudentenGF
        group by FakName),
    FakMaenner as (
        select FakName, count(*) as Maenner
        from StudentenGF
        where Geschlecht = 'M'
        group by FakName)
select FakTotal.FakName, (case when Maenner is null
    then 0 else Maenner end)/(total*1.0)
from FakTotal left outer join FakMaenner
on FakTotal.FakName = FakMaenner.FakName
```

Wir müssen beachten, dass nicht jede Fakultät Männer beherbergt, weswegen diese Fakultäten (in der Standardausprägung im SQL Interface ist dies für Theologie der Fall) dann aus dem Ergebnis herausfallen würden. Aus diesem Grund verwenden wir einen `LEFT OUTER JOIN` um die Zahl der Männer und die Zahl der Studenten insgesamt zu verbinden, wodurch auch die Theologie Fakultät im Ergebnis enthalten ist, auch wenn es keine Männer gibt.

Das `CASE`-Konstrukt dient in der oberen Anfrage dazu, den `NULL` Wert, die durch den Left Join für die Anzahl der Männer entstehen, wenn es keine Männer gibt, durch die Zahl 0 zu ersetzen. Alternativ ist dies möglich, indem man `COALESCE(maenner,0)/(total*1.0)` verwendet.

Alternativ können wir das `case`-Konstrukt verwenden, um die Anzahl der Männer an den jeweiligen Fakultäten zu ermitteln. Den Männeranteil erhalten wir dann, indem wir die Anzahl der Männer durch die Gesamtanzahl der Studenten an der Fakultät teilen.

```
select FakName,
    (sum(case when Geschlecht = 'M' then 1.00 else 0.00
        end)) / count(*)
from StudentenGF
group by FakName
```

- (b) Wir fordern hier, dass es keine Vorlesung an der Fakultät des Studenten (d.h. von einem Professor der gleichen Fakultät gelesen) geben darf, die vom Studenten nicht gehört wird.

```

select s.*
from StudentenGF s
where not exists (select *
                  from Vorlesungen v, ProfessorenF p
                  where v.gelesenVon = p.PersNr
                        and p.FakName = s.FakName
                        and not exists
                          (select *
                           from hoeren h
                           where h.VorlNr = v.VorlNr
                                and h.MatrNr = s.MatrNr));

```

Alternativ:

```

select * from StudentenGF s
where
  (select count(*)
   from Vorlesungen v, ProfessorenF p
   where v.gelesenVon = p.PersNr and p.FakName = s.FakName
  )
=
  (select count(*)
   from hoeren h, Vorlesungen v, ProfessorenF p
   where h.MatrNr = s.MatrNr and h.VorlNr = v.VorlNr and p
        .PersNr = v.gelesenVon and p.FakName = s.FakName)

```

#### Hausaufgabe 4

„Frühestes Semester“: Formulieren Sie eine SQL-Anfrage, um das Semester zu ermitteln in dem die Vorlesung „Der Wiener Kreis“ frühestens gehört werden kann. Testen Sie die Anfrage auch mit anderen Vorlesungen, insbesondere mit „Logik“.

**Lösung:**

```

with recursive voraussetzenRek(vorlNr, counter) as (
  (
    select v.vorlNr, 1 as counter
      from vorlesungen v
     where v.titel = 'Der Wiener Kreis'
  )
 union
  (
    select vs.vorgaenger, vsr.counter + 1 as counter
      from voraussetzenRek vsr, voraussetzen vs
     where vsr.vorlNr=vs.nachfolger
  )
)
select max(counter) as fruehestesSemester
from voraussetzenRek;

```