Grundlagen Datenbanken: Übung 05

Tanmay Deshpande

ge94vem@mytum.de

Gruppe 20 & 21

QR-Code für die Folien

Wiederholung

Woche 05

Casting

```
select   h.VorlNr, h.AnzProVorl, g.GesamtAnz,
         cast(h.AnzProVorl as decimal(6,2)) / g.GesamtAnz as Marktanteil
from     ( select VorlNr, count(*) as AnzProVorl
           from hören
           group by VorlNr ) h,
         ( select count (*) as GesamtAnz
           from Studenten) g;
```

Modularisierung mit CTEs (Common Table Expressions)

```
with h as (select VorlNr, count(*) as AnzProVorl
           from hoeren
           group by VorlNr) ,
     g as  (select count (*) as GesamtAnz
           from Studenten)

select h.VorlNr, h.AnzProVorl, g.GesamtAnz,
       cast(h.AnzProVorl as decimal(6,2)) / g.GesamtAnz as Marktanteil
from g,h
```

- Mehrere CTEs durch Komma trennen
- Ein CTE kann ein vorher definiertes CTE referenzieren

Quantifizierung

- Existenzquantifizierung:
  exists oder **not exists**
  Gibt true zurück, wenn das Ergebnis einer Unteranfrage mindestens ein Tupel enthält
  Bsp: select * from Studenten s where exists (select * from hoeren where h.matrnr = s.matrnr)
- Allquantifizierung:
  Keyword **all** ist kein vollwertiger Allquantor!
  Bsp: select * from Studenten where Semester >= all (select Semester from Studenten)
- Wie kann man Allquantifizierung in SQL ausdrücken?
  - mit count(*)
  - mit not exists(…)

Allquantifizierung durch count-Aggregation

- Studenten, die alle Vorlesungen hören

```
select h.MatrNr from hören h
group by h.MatrNr
having count (*) = (select count (*) from Vorlesungen);
```

Allquantifizierung durch not exists (…)

- Kalkülformulierung der Anfrage: Wer hat **alle** vierstündigen Vorlesungen gehört?

$$\{s \mid s \in \text{Studenten} \Lambda \forall v \in \text{Vorlesungen}\ (v.SWS=4 \Rightarrow \exists h \in \text{hören}\ (h.VorlNr=v.VorlNr \Lambda\ h.MatrNr=s.MatrNr))\}$$

- Nach Umformung:

$$\{s \mid s \in \text{Studenten} \Lambda \neg (\exists v \in \text{Vorlesungen}\ (v.SWS=4 \Lambda \neg(\exists h \in \text{hören}\ (h.VorlNr=v.VorlNr \Lambda\ h.MatrNr=s.MatrNr))))\}$$

Elimination von $\forall$ und $\Rightarrow$
Dazu sind folgende Äquivalenzen anzuwenden

$$\forall t \in R\ (P(t)) = \neg(\exists t \in R(\neg\ P(t)))$$
$$R \Rightarrow T = \neg R \vee T$$

- In SQL übersetzen:

```
select * from studenten s where not exists
(select * from vorlesungen v where v.sws = 4 and not exists
(select * from hören h where h.vorlnr = v.vorlnr and h.matrnr = s.matrnr))
```

Nullwerte

- Verursachen Probleme in Datenverarbeitung. Bsp: select * from Studenten where Semester < 13 or Semester >=13
- Ergebnis sollte alle Studenten enthalten, aber falls es Studenten gibt mit Semester = NULL, kommen diese nicht im Ergebnis vor
- Bei boolean Anfrage in SQL, Vergleiche mit Nullwerte resultieren nicht in true oder false sondern „unknown“. Unknowns kommen in Ergebnisse nicht vor
- Bei arithmetischen Operationen werden Nullwerte propagiert
- Müssen deshalb richtig behandelt werden. Bsp: case-Konstrukt, coalesce

Case-Konstrukt

- select MatrNr, (case when Note < 1.5 then ´sehr gut´
  when Note < 2.5 then ´gut´
  when Note < 3.5 then ´befriedigend´
  when Note < 4.0 then ´ausreichend´
  else ´nicht bestanden´ end)
  from prüfen;
- Die erste qualifizierende when-Klausel wird ausgewertet!

Joins

- Implizit: select * from R1, R2 where R1.A = R2.B;
- Explizit: select * from R1 join R2 on R1.A = R2.B;
- Outer Joins (Left/Right/Full):
  select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr, s.MatrNr, s.Name from
  Professoren p **left outer join** (prüfen f **left outer join** Studenten s on f.MatrNr=s.MatrNr) on
  p.PersNr=f.PersNr;

String-Vergleiche mit „like“

- select * from Studenten where Name like `T%eophrastos‘;
- ‚%‘ steht für beliebige oder gar keine Zeichen
- ‚_‘ steht für genau ein Zeichen

Aufgaben

Woche 05

Aufgabe 01

- 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.

Lösungsvorschlag 01a

- a) Bestimmen Sie das durchschnittliche Semester der Studenten der Universität.
- select avg(semester*1.0) from studenten;

Lösungsvorschlag 01b

- b) Bestimmen Sie das durchschnittliche Semester der Studenten, die mindestens eine Vorlesung bei Sokrates hören.
- 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
- Alternative Formulierung für studenten_von_sokrates:
  select DISTINCT s.* from studenten s, hoeren h, vorlesungen_von_sokrates v where h.matrnr = s.matrnr and v.vorlnr = h.vorlnr

Lösungsvorschlag 01c

- 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

Aufgabe 02

- „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.

Lösungsvorschlag 02

- 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));

Lösungsvorschlag 02

- 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));

Lösungsvorschlag 02

- 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));

Aufgabe 03

- a) Formen Sie den Ausdruck in einen Äquivalenten um, der keine Implikationen oder Allquantoren verwendet:

$$\{s \mid s \in \text{Studenten} \wedge$$
$$\forall h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \Rightarrow$$
$$\exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

- b) Übersetzen Sie den so erlangten Ausdruck in SQL. Testen Sie ihn in der Webschnittstelle.

Lösungsvorschlag 03

- a)

$$\{s \mid s \in \text{Studenten} \wedge \neg\exists h \in \text{hoeren}(h.\text{MatrNr} = s.\text{MatrNr} \wedge \neg\exists p \in \text{pruefen}(p.\text{MatrNr} = s.\text{MatrNr} \wedge p.\text{VorlNr} = h.\text{VorlNr} \wedge p.\text{Note} \leq 4))\}$$

- b) 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))

Aufgabe 04

$\text{ZehnkampfD} : \{\underline{\text{Name}, \text{Disziplin}}, \text{Punkte}\}$

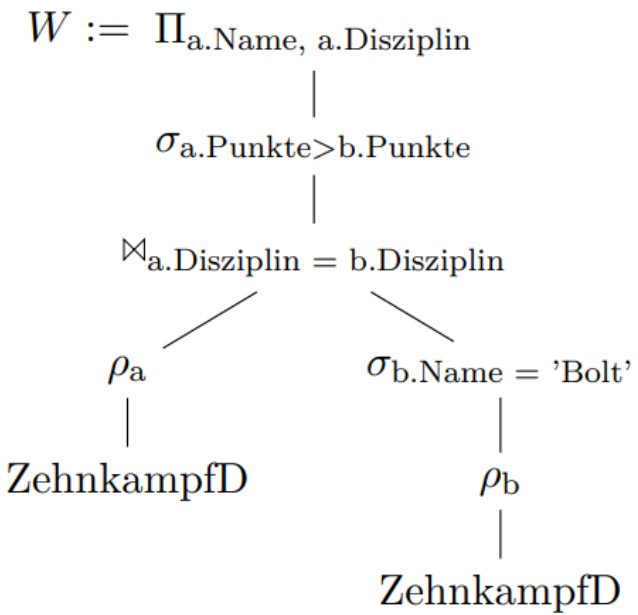
Name	Disziplin	Punkte
Bolt	100m	50
Bolt	Weitsprung	50
Eaton	100m	40
Eaton	Weitsprung	60
Suarez	100m	60
Suarez	Weitsprung	60
Behrenbruch	100m	30
Behrenbruch	Weitsprung	50
...	...	...

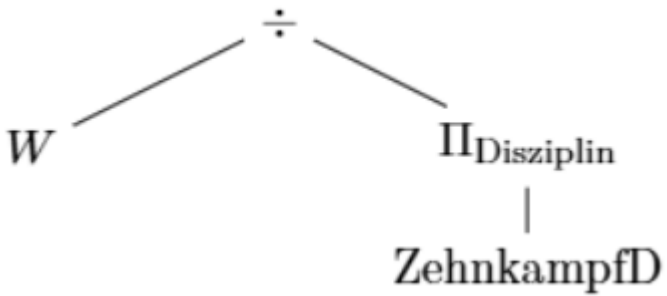
- Finden Sie alle ZehnkämpferInnen, die in allen Disziplinen besser sind als der Athlet mit dem Namen Bolt.

  Formulieren Sie die Anfrage
  - in der relationalen Algebra,
  - im relationalen Tupelkalkül,
  - im relationalen Domänenkalkül und
  - in SQL.

Lösungsvorschlag 04

- Relationale Algebra:





Lösungsvorschlag 04

- Tupelkalkül:

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

Lösungsvorschlag 04

- Domänenkalkül:

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

Lösungsvorschlag 04

- SQL (mit exists):

- ```
  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
  )
  )
  ```
  ```

Lösungsvorschlag 04

- SQL (mit 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)