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

Übung zur Vorlesung *Grundlagen: Datenbanken* im WS21/22

Christoph Anneser, Josef Schmeißer, Moritz Sichert, Lukas Vogel (gdb@in.tum.de)
https://db.in.tum.de/teaching/ws2122/grundlagen/

Blatt Nr. 04

Hausaufgabe 1

Formulieren Sie folgende Anfragen auf dem bekannten Universitätsschema in SQL. Geben Sie alle Ergebnisse duplikatfrei aus.

(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 Carnap unterrichtet haben – z.B. als potentielle Betreuer seiner Bachelorarbeit.

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

(e) Welche *Vorlesungen* werden von *Studenten* im Bachelorstudium (1. – 6. Semester) gehört? Geben Sie die Titel dieser *Vorlesungen* an.

(f) Bestimmen Sie für jede Vorlesung wie viele Studenten diese hören. Geben Sie auch Vorlesungen ohne Hörer aus. Sortieren Sie das Ergebnis absteigend nach Anzahl der Hörer.

Lösung:

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

```
select distinct 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';
```

(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 Carnap unterrichtet haben – z.B. als potentielle Betreuer seiner Bachelorarbeit.

```
select distinct 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 ='Carnap';
```

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

```
select distinct 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 ='Theophrastos';
```

(e) Welche *Vorlesungen* werden von *Studenten* im Bachelorstudium (1. – 6. Semester) gehört? Geben Sie die Titel dieser *Vorlesungen* an.

```
select distinct 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 6;
```

(f) Bestimmen Sie für jede Vorlesung wie viele Studenten diese hören. Geben Sie auch Vorlesungen ohne Hörer aus. Sortieren Sie das Ergebnis absteigend nach Anzahl der Hörer.

```
select v.VorlNr, v.Titel, count(h.MatrNr) as hoerer
from
  Vorlesungen v left outer join
  hoeren h on (v.VorlNr = h.VorlNr)
group by v.VorlNr, v.Titel
order by hoerer desc;
```

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

Führen Sie die folgenden Änderungen am Datenbestand des bekannten Universitätsschemas in SQL aus. Stellen Sie sicher, dass Ihre SQL-Statements mit jeder beliebigen Ausprägung des Schemas funktionieren.

a) Alle Professoren, die den Rang C3 haben, werden auf den Rang C4 befördert. Setzen Sie dazu den Rang aller C3-Professoren auf C4.

b) Die Planetenbewegungen sind vollständig erforscht. Löschen Sie alle Assistenten mit diesem Fachgebiet.

c) Eine neue Vorlesung mit dem Namen „Grundlagen: Datenbanken“ mit der Nummer 5278 soll erstellt werden. Die Vorlesung wird von der Professorin Curie gehalten und hat die Vorlesung „Logik“ als Voraussetzung. Sie soll 4 SWS umfassen. Tragen Sie den Studenten mit der Matrikelnummer 28106 als Hörer der Vorlesung ein. Erstellen Sie alle notwendigen SQL-Statements.

Lösung:

a) Alle Professoren, die den Rang C3 haben, werden auf den Rang C4 befördert. Setzen Sie dazu den Rang aller C3-Professoren auf C4.

```
update Professoren set Rang = 'C4' where Rang = 'C3';
```

b) Die Planetenbewegungen sind vollständig erforscht. Löschen Sie alle Assistenten mit diesem Fachgebiet.

```
delete
from Assistenten
where Fachgebiet = 'Planetenbewegung';
```

c) Eine neue Vorlesung mit dem Namen „Grundlagen: Datenbanken“ mit der Nummer 5278 soll erstellt werden. Die Vorlesung wird von der Professorin Curie gehalten und hat die Vorlesung „Logik“ als Voraussetzung. Sie soll 4 SWS umfassen. Tragen Sie den Studenten mit der Matrikelnummer 28106 als Hörer der Vorlesung ein. Erstellen Sie alle notwendigen SQL-Statements.

```
insert into Vorlesungen
select 5278, 'Grundlagen: Datenbanken', 4, PersNr
from Professoren
where Name = 'Curie';

insert into voraussetzen
select VorlNr, 5278
from Vorlesungen
where Titel = 'Logik';

insert into hoeren values (28106, 5278);
```

Hausaufgabe 4

Folgender Ausdruck im Tupelkalkül gibt alle Studenten aus, die alle von ihnen gehörten Vorlesungen bestanden haben.

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

Übersetzen Sie diese Anfrage nun in SQL. Da SQL keine Allquantoren und Implikationen unterstützt, müssen Sie sie dazu zunächst umformen.

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

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

Lösung:

a) Wir formen zunächst die innere Implikation um, denn $A \Rightarrow B \iff \neg A \vee B$:

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

Wir ersetzen nun den Allquantor durch einen negierten Existenzquantor, denn $\forall x(P(x)) \iff \neg\exists x(\neg P(x))$:

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

Zuletzt wenden wir De Morgans Regel an, um die Negation nach innen zu ziehen, denn $\neg(A \vee B) \iff \neg A \wedge \neg B$:

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

b) Die Anfrage kann nun eins zu eins in SQL übersetzt werden, wobei jeder logische Operator einfach durch seine SQL-Entsprechung ersetzt wird:

```
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
    )
)
```