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

Übung zur Vorlesung *Einführung in die Informatik 2 für Ingenieure (MSE)*

Alexander van Renen (renen@in.tum.de)
http://db.in.tum.de/teaching/ss20/ei2/

Lösungen zu Blatt 10

Tool zum Üben der relationalen Algebra: http://www-db.in.tum.de/~muehe/ira/.
SQL-Schnittstelle: http://hyper-db.com/interface.html.

Aufgabe 1: SQL

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 1:

(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.

```
select avg(semester*1.0)
from studenten s
where exists
        (select *
        from hoeren h, vorlesungen v, professoren p
        where v.gelesenVon = p.persnr
        and p.name = 'Sokrates'
        and h.matrnr = s.matrnr
        and v.vorlnr = h.vorlnr)
```

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.

(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*1.0) from
(select count(*) as hcount from hoeren) h,
(select count(*) as scount from studenten) s
```

Aufgabe 2: SQL Exists

Formulieren Sie die folgenden Anfragen auf dem bekannten Universitätsschema in SQL:

(a) Finden Sie die Namen der *Studenten*, die in keiner *Prüfung* eine bessere Note als 3.0 hatten.

(b) FInden Sie alle Professoren die einen Assistenten haben, aber dennoch keine Vorlesung geben.

Lösung 2:

(a) Finden Sie die Namen der *Studenten*, die in keiner *Prüfung* eine bessere Note als 3.0 hatten.

```
SELECT s.Name, s.MatrNr
FROM Studenten s
WHERE NOT EXISTS (SELECT *
                  FROM pruefen p
                  WHERE p.MatrNr = s.MatrNr
                  AND p.Note < 3.0);
```

(b) Finden Sie alle Professoren die einen Assistenten haben, aber dennoch keine Vorlesung geben.

```
select *
from Professoren p
where exists (select *
                  from Assistenten a
                  where a.boss = p.persNr)
and not exists (select *
                  from Vorlesungen v
                  where v.gelesenVon = p.persNr);
```

Aufgabe 3: Bekanntheitsgrad

Formulieren Sie eine SQL-Anfrage, um den Bekanntheitsgrad von Studenten zu ermitteln. Der Bekanntheitsgrad eines Studenten ist definiert als die Anzahl an Studenten die ihn kennen. Gehen Sie dabei davon aus, dass Studenten sich aus gemeinsam besuchten Vorlesungen kennen. Sortieren Sie das Ergebnis absteigend nach Bekanntheitsgrad!

Bonus: Überlegen Sie sich warum Studenten die niemanden kennen nicht im Ergebniss auftauchen. Wie könnte man dieses Problem lösen.

Lösung 3

```
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;
```