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/ss17/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: 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!

Lösung 1

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;
```

Aufgabe 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 2

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 temporärer Sichten mit **with** 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));
```

Aufgabe 3: Gewichtete Durchschnittsnote

Falls Sie ihre Anfrage sinnvoll testen möchten, sollten sie sich selber auf einer lokalen Datenbank eine größere *prüfen*-Relation anlegen. Das Schema und die Ausprägung, welche online verwendet wird, finden Sie auf der Website.

Bestimmen Sie für alle Studenten eine gewichtete Durchschnittsnote ihrer Prüfungen. Die Gewichtung der einzelnen Prüfungen erfolgt gemäß dem Vorlesungsumfang (SWS). Dies entspricht dem Verfahren der Durchschnittsnotenberechnung für Ihr Bachelor-Zeugnis.

Lösung 3

Hier sollen die Noten der Studenten gewichtet werden, d.h. dass Noten für Prüfungen über kurze Vorlesungen (z.B. 2 SWS) abgewertet werden, da der Lernaufwand dafür geringer war als für Prüfungen über lange Vorlesungen, die dadurch aufgewertet werden.

```
SELECT s.Name, SUM(p.Note*v.SWS)/SUM(v.SWS) AS Durchschnittsnote
FROM pruefenXL p, Vorlesungen v, Studenten s
WHERE p.VorlNr = v.VorlNr AND s.MatrNr = p.MatrNr
GROUP BY s.Name
```

Aufgabe 4: Erst hören, dann prüfen

Welche Studenten haben alle Vorlesungen, die sie haben prüfen lassen, auch tatsächlich vorher gehört?

Lösung 4

Die Anforderung, dass die Studenten im Anfrage-Ergebnis alle Vorlesungen, die sie haben prüfen lassen auch tatsächlich gehört haben, lässt sich umschreiben zu: „Es darf keine Vorlesung geben, die geprüft wurde, zu der es aber keinen Eintrag in *hoeren* gibt."

```
SELECT s.*
FROM Studenten s
WHERE NOT EXISTS (SELECT * FROM pruefen p
                  WHERE  s.MatrNr = p.MatrNr
                    AND  NOT EXISTS (SELECT *
                                     FROM  hoeren h
                                     WHERE h.MatrNr = s.MatrNr
                                       AND h.VorlNr = p.VorlNr));
```

Aufgabe 5: Hörer-Dividende

Was bringt der Vorlesungsbesuch? Finden Sie heraus, ob es für Prüfungen von Vorteil ist, die jeweiligen Vorlesungen auch gehört zu haben. Ermitteln Sie dazu die Durchschnittsnote der Prüfungen, zu denen die Studenten die Vorlesungen nicht gehört haben und die Durchschnittsnote der Prüfungen, zu denen sie die Vorlesungen gehört haben.

Lösung 5

Wir bestimmen, wie sich das Verhältnis der Prüfungsleistungen von gehörten zu nicht gehörten Vorlesungen insgesamt darstellt.

```
WITH nichtgehoert AS
  (SELECT AVG(Note) AS noteNichtGehoert
  FROM pruefenXL p
  WHERE NOT EXISTS (SELECT *
    FROM hoeren h
    WHERE h.VorlNr = p.VorlNr AND h.MatrNr = p.MatrNr)),
gehoert AS
  (SELECT AVG(Note) AS noteGehoert
  FROM pruefenXL p
  WHERE EXISTS (SELECT *
```

```
    FROM hoeren h
    WHERE h.VorlNr = p.VorlNr AND h.MatrNr = p.MatrNr))

SELECT * FROM nichtgehoert, gehoert;
```

Da beide Views aus jeweils nur einem Wert bestehen, ergibt sich das Resultat der Anfrage als Kreuzprodukt.