UNIVERSITÄT DES SAARLANDES
Fachrichtung 6.2 - Informatik
Lehrstuhl Prof. Weikum

# Informationssysteme SS 2002

## Übung 3 Beispiellösung

## Aufgabe 1: Äquivalenz von RA und TRK

Betrachten Sie erneut die Musikdatenbank mit folgendem Schema:

| | |
|---|---|
| Disk | (DiskID, DiskTitel, Preis) |
| | *78462, W. A. Mozart: Klavierkonzerte, 29.99* |
| Musikstück | (DiskID, StückID, Titel, Länge) |
| | *78462, 4, Konzert für Klavier und Orchester Nr. 21, 2732* |
| Person | (PID, Name, Nationalität) |
| | *9362, W. A. Mozart, Österreich* |
| Interpret | (PID, DiskID, StückID, Funktion, Instrument) |
| | *15267, 78462, 4, Solist, Klavier* |
| Autor | (PID, DiskID, StückID, Tätigkeit) |
| | *9362, 78462, 4, Komponist* |

Geben Sie für die folgenden Relationenalgebra-Anfragen äquivalente Anfragen im sicheren Tupelrelationenkalkül an.

a) $\pi$ [Name]
(( $\sigma$[Instrument != 'Klavier'] (Interpret) |×| $\sigma$ [Tätigkeit = 'Komponist'] (Autor)) |×| Person)

Die intuitive Bedeutung der Anfrage ist:
Finde alle Komponisten, die bei einer Aufnahme eines ihrer Stücke mitspielen, aber nicht Klavier.

$$\{p.Name \mid p \in Person \wedge \exists m (m \in Musikstück \wedge \exists i (i \in Interpret \wedge i.Instrument != 'Klavier' \wedge i.PID=p.PID \wedge m.DiskID = i.DiskID \wedge m.StückID = i.StückID) \wedge \exists a (a \in Autor \wedge a.Tätigkeit = 'Komponist' \wedge a.PID=p.PID \wedge m.DiskID = a.DiskID \wedge m.StückID = a.StückID))\}$$

b) $\pi$ [DiskTitel]
(Disk |×| ($\pi$ [DiskID]( $\sigma$ [Preis < 20](Disk)) - $\pi$ [DiskID]( $\sigma$ [Länge < 10] (Musikstück)))

Die intuitive Bedeutung der Anfrage ist:
Finde CDs unter 20 DM, die kein einziges Stück mit einer Spieldauer unter 10 Minuten enthalten.

$$\{d.DiskTitel \mid d \in Disk \wedge d.Preis < 20 \wedge \neg\exists m (m \in Musikstück \wedge m.Länge < 10 \wedge m.DiskID = d.DiskID)\}$$

# Aufgabe 2: Anfragen in SQL - Universitätsdatenbank

Gegeben sei das aus der ersten Übung bekannte Schema einer Universitätsdatenbank:

| | |
|---|---|
| Professor | (P_Name, Fachrichtung_Nr, Gebäude, Raum, Tel) |
| Fachrichtung | (Fachrichtung_Nr, F_Name, Studiendekan) |
| Gebäude | (Gebäude, Hausmeister) |
| Student | (Matrikel_Nr, S_Name, Semester, Fachrichtung_Nr) |
| Prüfung | (Matrikel_Nr, Fach, Prüfer, Note) |

Formulieren Sie die folgenden Anfragen in SQL:

a) An welchen Hausmeister muß sich Prof. Weikum wenden, wenn er seinen Zimmerschlüssel vergessen hat?

```
SELECT  Hausmeister
FROM    Gebäude, Professor
WHERE   P_Name = ‚Weikum’
AND     Professor.Gebäude = Gebäude.Gebäude
```

b) Welche Studenten (Matrikel_Nr) haben eine Prüfung beim augenblicklichen Studiendekan ihrer Fachrichtung abgelegt?

```
SELECT  DISTINCT Student.Matrikel_Nr
FROM    Prüfung, Fachrichtung, Student
WHERE   Prüfer = Studiendekan
AND     Student.Matrikel_Nr = Prüfung.Matrikel_Nr
AND     Student.Fachrichtung_Nr = Fachrichtung.Fachrichtung_Nr
```

c) Wo (Gebäude, Raum) fand die Prüfung von Hugo Meier im Fach Betriebssysteme statt (Annahme: Professoren prüfen in ihren Büros)?

```
SELECT  Gebäude, Raum
FROM    Prüfung, Student, Professor
WHERE   S_Name = ‚Hugo Meier’
AND     Student.Matrikel_Nr = Prüfung.Matrikel_Nr
AND     Prüfer = P_Name
AND     Fach = ‚Betriebssysteme’
```

d) Welche Studenten (Matrikel_Nr) mit mindestens 4 Semestern haben noch keine Prüfung abgelegt?

```
SELECT  Matrikel_Nr
FROM    Student
WHERE   Semester > 3
AND     NOT EXISTS
        (SELECT *
         FROM   Prüfung
         WHERE  Student.Matrikel_Nr = Prüfung.Matrikel_Nr)
```

oder

```
SELECT  Matrikel_Nr
FROM    Student
WHERE   Semester > 3
AND     Matrikel_Nr NOT IN
        (SELECT DISTINCT Matrikel_Nr
```

```
         FROM   Prüfung)
```

oder

```
SELECT  Matrikel_Nr
FROM    Student
WHERE   Semester > 3
MINUS
SELECT  DISTINCT Matrikel_Nr
FROM    Prüfung
```

e) Welche Studenten (Matrikel_Nr) haben ausschließlich Prüfungen bei Professoren ihrer Fachrichtung abgelegt?

```
SELECT  Matrikel_Nr
FROM    Student
WHERE   NOT EXISTS
        (SELECT *
         FROM   Prüfung, Professor
         WHERE  Student.Matrikel_Nr = Prüfung.Matrikel_NR
         AND    Prüfer = P_Name
         AND    Student.Fachrichtung_Nr !=
                Professor.Fachrichtung_Nr)
```

Das liefert allerdings auch Studenten, die noch gar keine Prüfung haben. Besser daher:

```
SELECT  DISTINCT Matrikel_Nr
FROM    Student, Prüfung p1
WHERE   Student.Matrikel_Nr = p1.Matrikel_Nr
AND     NOT EXISTS
        (SELECT *
         FROM   Prüfung p2, Professor
         WHERE  p2.Matrikel_Nr = Student.Matrikel_Nr
         AND    Prüfer = P_Name
         AND    Professor.Fachrichtung_Nr !=
                Student.Fachrichtung_Nr
```

f) Welche Studenten (Matrikel_Nr) haben alle ihre bisher abgelegten Prüfungen mit der Bestnote 1.0 bestanden?

```
SELECT  Matrikel_Nr
FROM    Prüfung p1
WHERE   NOT EXISTS
        (SELECT *
         FROM   Prüfung p2
         WHERE  p1.Matrikel_Nr = p2.Matrikel_Nr
         AND    Note > 1.0)
```

Einfacher wird die Lösung mit der Benutzung von MINUS:

```
SELECT  DISTINCT Matrikel_Nr
FROM    Prüfung
MINUS
SELECT  DISTINCT Matrikel_Nr
FROM    Prüfung
WHERE   Note > 1.0
```

Eine noch elegantere Lösung wäre:

```
SELECT  Matrikel_Nr
FROM    Prüfung
GROUP   BY Matrikel_Nr
HAVING  MAX(Note) = 1.0
```

g) Bestimmen Sie die Durchschnittsnote für jeden Studenten.

```
SELECT  Matrikel_Nr, AVG(Note)
FROM    Prüfung
GROUP   BY Matrikel_Nr
```

h) Welche Studenten (Matrikel_Nr) haben ein Prüfungsfach (Fach) besser abgeschlossen als der Durchschnitt in diesem Fach?

```
SELECT  DISTINCT Matrikel_Nr
FROM    Prüfung p1
WHERE   Note >  (SELECT AVG(Note)
                 FROM   Prüfung p2
                 WHERE  p1.Fach = p2.Fach)
```

Alternativ geht hier auch eine Lösung in zwei Schritten mit einem View:

```
CREATE  VIEW AVG_Fach AS
SELECT  Fach, AVG(Note) AVG_Note
FROM    Prüfung
GROUP   BY Fach

SELECT  DISTINCT Matrikel_Nr
FROM    AVG_Fach a, Prüfung p
WHERE   Note > AVG_Note
AND     a.Fach = p.Fach
```

Eine vermeintlich einfache Lösung funktioniert leider nicht:

```
SELECT  DISTINCT Matrikel_Nr
FROM    Prüfung
GROUP   BY Fach
HAVING  Note > AVG(Note)
```

Hier kommen mit Matrikel_Nr im SELECT und Note im HAVING zwei Nichtgruppeneigenschaften vor. Dies ist in Queries mit GROUP BY nicht erlaubt.

i) Welches Prüfungsfach (Fach) hat die beste Durchschnittsnote?

```
SELECT  Fach
FROM    Prüfung
GROUP   BY Fach
HAVING  AVG(Note) ≥ ALL (SELECT AVG(Note)
                         FROM   Prüfung
                         GROUP  BY Fach)
```

Ein anderer Ansatz könnte wie folgt aussehen:

```
SELECT  Fach, AVG(Note)
FROM    Prüfung
GROUP   BY Fach
ORDER   BY 2 DESC
```

Bei dieser Formulierung werden die Prüfungen nach Fächern gruppiert und nach Durchschnittsnote absteigend sortiert ausgegeben. Das erste Resultattupel ist also die gewünschte Antwort. Insofern ist diese Antwort keine „exakte“ Lösung, aber unter pragmatischen Gesichtspunkten trotzdem brauchbar.

j) Welcher Student (Name) hat alle Prüfungen als Bester abgeschlossen?

```
SELECT  DISTINCT S_Name
FROM    Student s
WHERE   NOT EXISTS  (SELECT *
                     FROM   Prüfung p1
                     WHERE  s.Matrikel_Nr = p1.Matrikel_Nr
                     AND    EXCISTS (SELECT *
                                     FROM   Prüfung p2
                                     WHERE  p1.Fach = p2.Fach
                                     AND    p1.Note < p2.Note))
```

## Aufgabe 3: Anfragen in SQL - Musikdatenbank

Gegeben sei ein gegenüber Übung 1 erweitertes Schema der Musikdatenbank:

| | |
|---|---|
| Disk | (DiskID, DiskTitel, **Preis**) |
| | *78462, W. A. Mozart: Klavierkonzerte, 29.99* |
| Musikstück | (DiskID, StückID, Titel, **Länge**) |
| | *78462, 4, Konzert für Klavier und Orchester Nr. 21, 2732* |
| Person | (PID, Name, Nationalität) |
| | *9362, W. A. Mozart, Österreich* |
| Interpret | (PID, DiskID, StückID, Funktion, Instrument) |
| | *15267, 78462, 4, Solist, Klavier* |
| Autor | (PID, DiskID, StückID, Tätigkeit) |
| | *9362, 78462, 4, Komponist* |

Formulieren Sie die folgenden Anfragen in SQL:

a) Welche Stücke (Titel) hat F. Chopin komponiert?

```
SELECT  DISTINCT m.titel
FROM    Musikstück m, Person p, Autor a
WHERE   m.DiskID = a.DiskID
AND     m.StückID = a.StückID
AND     a.Funktion = ‘Komponist’
AND     p.PID = a.PID
AND     p.Name LIKE ‘%Chopin%’
```

b) Welches ist die teuerste Disk und was kostet sie?

```
SELECT  DiskTitel, Preis
FROM    Disk d1
WHERE   NOT EXISTS  (SELECT *
                     FROM   Disk d2
                     WHERE  d2.Preis > d1.Preis)
```

c) Welche Disk enthält das längste Stück unter den Disks, die nicht mehr als 20 (Mark) kosten?

```
SELECT  d1.DiskTitel
```

```
FROM    Disk d1, Musikstück m1
WHERE   d1.DiskID = m1.DiskID
AND     d1.Preis ≤ 20
AND     NOT EXISTS  (SELECT *
                     FROM   Disk d2, Musikstück m2
                     WHERE  d2.DiskID = m2.DiskID
                     AND    d2.Preis ≤ 20
                     AND    m1.Länge < m2.Länge)
```

d) Welche Disks (DiskTitel) enthalten ausschließlich Stücke, die von F. Chopin komponiert wurden?

```
SELECT  DISTINCT d.DiskTitel
FROM    Disk d, Person p, Musikstück m, Autor a
WHERE   d.DiskID = a.DiskID
AND     a.DiskID = m.DiskID
AND     m.StückID = a.StückID
AND     a.PID = p.PID
AND     a.Funktion = ‘Komponist’
AND     p.Name LIKE ,%Chopin%’
AND     NOT EXISTS  (SELECT *
                     FROM   Musikstück m2, Autor a2
                     WHERE  m2.DiskID = m.DiskID
                     AND    a2.DiskID = m2.DiskID
                     AND    a2.Funtkion = ‘Komponist’
                     AND    a2.PID != p.PID)
```

e) Welche Disks enthalten kein Stück, das länger ist als 60 (Sekunden)?

```
SELECT  DISTINCT d.DiskID, d.DiskTitel
FROM    Musikstück m, Disk d
WHERE   d.DiskID = m.DiskID
GROUP   BY d.DiskID, d.DiskTitel
HAVING  MAX(Länge) ≤ 60
```

f) Welchen Durchschnittspreis haben Disks, auf denen Interpreten aus über 3 Nationen zu hören sind?

```
SELECT  AVG(Preis)
FROM    Disk d, Interpret i, Person p
WHERE   d.DiskID = i.DiskID
AND     p.PID = i.PDI
GROUP   BY i.DiskID
HAVING  COUNT(DISTINCT Nationalität) > 3
```

g) Ermitteln Sie für jede Disk die Gesamtlänge der drei längsten Stücke.

```
SELECT  d.DiskID, d.DiskTitel, SUM(m.Länge), COUNT(*)
FROM    Disk d, Musikstück m
WHERE   d.DiskID = m.DiskID
AND     3 > (SELECT COUNT(*)
             FROM   Disk d2, Musikstück m2
             WHERE  d2.DiskID = d.DiskID
             AND    m2.DiskID = m.DiskID
             AND    m2.Länge < m.Länge)
GROUP   BY d.DiskID, d.DiskTitel
```