

## Informationssysteme (SS 05)

### Übungsblatt 5

#### Beispiellösungen

#### Aufgabe 1: 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 EXISTS (SELECT *
                             FROM Prüfung p2
                             WHERE p1.Fach = p2.Fach
                             AND p1.Note < p2.Note))
```

## Aufgabe 2: Anfragen in SQL - Musikdatenbank

Gegeben sei ein gegenüber Übung 1 erweitertes Schema der Musikdatenbank:

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

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.Funktion = '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.PID
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
```

### Aufgabe 3: Abbildung von SQL auf TRK und RA

Geben Sie für die folgenden SQL-Anfragen auf der Musikdatenbank äquivalente Formulierungen in der Relationenalgebra und dem sicheren Tupelrelationenkalkül an.

a) Select D.DiskTitel

From Disk D

Where D.DiskID In

( Select M.DiskID From Musikstück M

Where M.Titel = 'I love you'

And M.DiskID In

( Select I.DiskID From Interpret I

Where I.Instrument = 'Triangel'

And I.StückID = M.StückID ) )

Die intuitive Bedeutung der Anfrage ist:

Auf welchen CDs ist ein Stück "I love you" mit einer Triangel zu hören?

RA:  $\pi[\text{DiskTitel}]$

$((\text{Disk}) \times | \sigma[\text{Titel} = \text{'I love you'}](\text{Musikstück}) \times | \sigma[\text{Instrument} = \text{'Triangel'}](\text{Interpret}))$

TRK:  $\{d.\text{DiskTitel} \mid d \in \text{Disk} \wedge$

$\exists m (m \in \text{Musikstück} \wedge m.\text{Titel} = \text{'I love you'} \wedge m.\text{DiskID} = d.\text{DiskID} \wedge$

$\exists i (i \in \text{Interpret} \wedge i.\text{Instrument} = \text{'Triangel'} \wedge i.\text{DiskID} = d.\text{DiskID} \wedge$

$i.\text{StückID} = m.\text{StückID}))\}$

b) Select D.DiskTitel

From Disk D

Where D.Preis < 20

And Exists

( Select \*

From Interpret I

Where I.Instrument = 'Sitar'

And I.DiskID = D.DiskID )

And Not Exists

( Select \*

From Musikstück M

Where M.Länge > 5

And M.DiskID = D.DiskID )

Die intuitive Bedeutung der Anfrage ist:

Auf welchen CDs unter 20 DM ist eine Sitar zu hören und kein einziges Stück länger als 5 Minuten?

RA:  $\pi[\text{DiskTitel}] ( \sigma[\text{Preis} < 20] (\text{Disk}) \times | \sigma[\text{Instrument} = \text{'Sitar'}] (\text{Interpret}) -$

$\sigma[\text{Länge} > 5] (\text{Disk} \times | \text{Musikstück}))$

TRK:  $\{d.\text{DiskTitel} \mid d \in \text{Disk} \wedge d.\text{Preis} < 20 \wedge$

$\exists i (i \in \text{Interpret} \wedge i.\text{Instrument} = \text{'Sitar'} \wedge i.\text{DiskID} = d.\text{DiskID}) \wedge$

$\neg \exists m (m \in \text{Musikstück} \wedge m.\text{Länge} > 5 \wedge m.\text{DiskID} = d.\text{DiskID})\}$

c) Select D.DiskTitel

From Disk D

Where D.DiskID = All ( Select M.DiskID

From Musikstück M, Person P, Interpret I, Autor A

Where (M.StückID = I.StückID Or M.StückID = A.StückID)

And (P.PID = I.PID Or P.PID = A.PID)

And P.Nationalität = 'Luxemburg' )

Die intuitive Bedeutung der Anfrage ist:

Gibt es eine einzige CD, an der ein Luxemburger beteiligt ist (als Interpret oder Komponist oder Dirigent oder ...)?

Falls ja, gib diese CD aus.

Falls es mehrere CDs mit Luxemburgern gibt, gib die leere Menge aus.

Falls es überhaupt keine CDs mit Luxemburgern gibt (das Ergebnis der Subquery also leer ist), gib alle CDs aus.

RA: M := Musikstück;

I := Interpret;

A := Autor;  
P := Person;

$$\pi[\text{DiskTitel}] ( \pi[\text{DiskID}] (\text{Disk}) - \pi[\text{Disk.DiskID}] (\sigma[\text{M.DiskID} \neq \text{Disk.DiskID}] (\sigma[\text{Nationalität} = \text{'Luxemburg'} \wedge (\text{M.StückID} = \text{I.StückID} \vee \text{M.StückID} = \text{A.StückID}) \wedge (\text{P.PID} = \text{I.PID} \vee \text{P.PID} = \text{A.PID})] (\text{M} \times \text{P} \times \text{I} \times \text{A} \times \text{Disk}))) )$$

TRK: {d.DiskTitel | d ∈ Disk ∧ ∀m (m ∈ Musikstück ⇒ ∃m ∃p ∃i ∃a (m ∈ Musikstück ∧ p ∈ Person ∧ i ∈ Interpret ∧ a ∈ Autor ∧ p.Nationalität = 'Luxemburg' ∧ (p.PID = i.PID ∨ p.PID = a.PID) ∧ (m.StückID = i.StückID ∨ m.StückID = a.StückID)))}

#### Formale Herleitung der Semantik

(Dieser formale Teil wurde bei der Lösung eigentlich nicht erwartet und dient hier der zusätzlichen Illustration.)

a1)

$$\begin{aligned} & \text{sql2trc} [\text{Select...From D Where D.DiskID In} \\ & \quad (\text{Select...From M Where M.Titel=...And M.DiskID In} \\ & \quad \quad (\text{Select...From I Where I.Instrument=...And I.StückID=...})] \\ & = \{d.\text{DiskTitel} \mid d \text{ in D and exists m ( m in M and d.DiskID=m.DiskID and} \\ & \quad \text{sql2trc' [M.Titel=... And} \\ & \quad \quad \text{M.DiskID In (Select...From I Where I.Instrument=...And I.StückID=...)}])\} \\ & = \{d.\text{DiskTitel} \mid d \text{ in D and exists m ( m in M and d.DiskID=m.DiskID and m.Titel=... and} \\ & \quad \text{exists i (i in I and m.DiskID=i.DiskID} \\ & \quad \quad \text{and sql2trc'[I.Instrument=...And I.StückID=...])}\} \\ & = \{d.\text{DiskTitel} \mid d \text{ in D and exists m ( m in M and d.DiskID=m.DiskID and m.Titel=... and} \\ & \quad \text{exists i (i in I and m.DiskID=i.DiskID and i.Instrument=... and i.StückID=...)}\} \end{aligned}$$

a2)

$$\begin{aligned} & \text{sql2ra} [\text{Select...From D Where D.DiskID In} \\ & \quad (\text{Select...From M Where M.Titel=...And M.DiskID In} \\ & \quad \quad (\text{Select...From I Where I.Instrument=...And I.StückID=...})] \\ & = \pi[\text{D.DiskTitel}] (\text{sql2ra}'[\text{Where D.DiskID In} \\ & \quad (\text{Select...From M Where M.Titel=...And M.DiskID In} \\ & \quad \quad (\text{Select...From I Where I.Instrument=...And I.StückID=...})] (\text{D})) \\ & = \pi[\text{D.DiskTitel}] (\text{sql2ra}'[\text{Where M.Titel=... And M.DiskID In} \\ & \quad (\text{Select...From I Where I.Instrument=...And I.StückID=...})] \\ & \quad (\text{sigma}[\text{D.DiskID} = \text{M.DiskID}] (\text{D} \times \text{M}))) \\ & = \pi[\text{D.DiskTitel}] (\text{sigma}[\text{M.Titel=...}] \end{aligned}$$

(sql2ra'[Where I.Instrument=... And I.StückID=...]  
(sigma[M.DiskID=I.DiskID] (sigma[D.DiskID=M.DiskID] (D x M) x I))))

= pi[D.DiskTitel] (sigma[M.Titel=...]  
(sigma[I.Instrument=... and I.StückID=...]  
(sigma[M.DiskID=I.DiskID] (sigma[D.DiskID=M.DiskID] (D x M) x I))))

b1)

sql2trc [Select...From D Where D.Preis<...And  
Exists (Select...From I Where I.Instrument=...And I.DiskID=...) And  
Not Exists (Select...From M Where M.Länge>...And M.DiskID=...)]

= {d.DiskTitel | d in D and d.Preis<... and  
exists i (i in I and i.instrument=... and i.DiskID=...) and  
not exists m (m in M and m.Länge>... and m.DiskID=...)}  
}

b2)

sql2ra [Select...From D Where D.Preis<...And  
Exists (Select...From I Where I.Instrument=...And I.DiskID=...) And  
Not Exists (Select...From M Where M.Länge>...And M.DiskID=...)]

= pi[D.DiskTitel] (sql2ra'[D.Preis<... And  
Exists (Select...From I Where I.Instrument=...And I.DiskID=...) And  
Not Exists (Select...From M Where M.Länge>...And M.DiskID=...)] (D))

= pi[D.DiskTitel] (sigma[D.Preis<...](D) intersect  
sql2ra'[Exists (Select...From I Where I.Instrument=...And I.DiskID=...) And  
Not Exists (Select...From M Where M.Länge>...And M.DiskID=...)] (D))

= pi[D.DiskTitel] (sigma[D.Preis<...](D) intersect  
sql2ra'[I.Instrument=...And I.DiskID=...And  
Not Exists (Select...From M Where M.Länge>...And M.DiskID=...)] (D x I))

= pi[D.DiskTitel] (sigma[D.Preis<...](D) intersect  
pi[sch(D)](sigma[I.Instrument=...and I.DiskID=...](D x I) intersect  
pi[sch(D)]((D x I) minus  
pi[sch(D x I)](sigma[M.Länge>...And M.DiskID=...](D x I x M))))

c1)

sql2trc [Select...From D Where D.DiskID =All (Select M.DiskID From M,P,I,A Where cond)]

= {d.DiskTitel | d in D and forall m  
(m in M => (d.DiskID=m.DiskID and  
exists m,p,i,a  
(m in M and p in P and i in I and a in A and sql2trc'[cond]))))}

mit

sql2trc'[cond] = (m.StückID=i.StückID or M.StückID=a.StückID) and  
(p.PID=i.PID or p.PID=a.PID) and p.Nationalität='Luxemburg'

c2)

sql2ra [Select...From D Where D.DiskID =All (Select M.DiskID From M,P,I,A Where cond)]



$\pi[D.DiskTitel] (\text{sql2ra}'[=All (\text{Select } M.DiskID \text{ From } M,P,I,A \text{ Where } cond) (D)])$   
 $\pi[D.DiskTitel] (\text{sql2ra}'[<>Any (\text{Select } M.DiskID \text{ From } M,P,I,A \text{ Where } cond) (D)])$   
 $\pi[D.DiskTitel] (D \text{ minus } \pi[sch(D)] (\text{sql2ra}'[<>Any (\text{Select } M.DiskID \text{ From } M,P,I,A \text{ Where } cond)(D)]))$   
 $\pi[D.DiskTitel] (D \text{ minus } \pi[sch(D)] (\text{sql2ra}'[cond] (\sigma[D.DiskID <> M.DiskID](D \times M \times P \times I \times A))))$

mit

$\text{sql2ra}'[cond](\text{expr}) = \sigma[(M.StückID=I.StückID \text{ or } M.StückID=A.StückID) \text{ and } (P.PID=I.PID \text{ or } P.PID=A.PID) \text{ and } P.Nationalität='Luxemburg'](\text{expr})$

#### Aufgabe 4: SQL Abfragen

Betrachten Sie die vereinfachte Universitätsdatenbank mit Informationen über Fachbereiche, Dozenten, Lehrangebote, Studenten und Prüfungen. Das Schema der Datenbank (mit Beispielausprägungen) ist unten aufgeführt (Primärschlüssel sind unterstrichen):

**Departments** (DName, Chair):

<u>DName</u>	Chair
Computer Science	Bob Smith
Electrical Engineering	John Miller
...	...

**Teachers** (TName, Office, Phone, DName):

<u>TName</u>	Office	Phone	DName
Bob Smith	122	4819	Computer Science
Mary Taylor	245	4716	Computer Science
John Miller	312	223322	Electrical Engineering
Mike Franklin	444	4545	Electrical Engineering
...	...	...	...

**Courses** (CNo, Title, Semester, Room, Schedule, Lecturer):

<u>CNo</u>	Title	Semester	Room	Schedule	Lecturer
1	Database Systems	Summer 2001	322	...	Mary Taylor
2	Wireless Communication	Winter 2001	455	...	John Miller
3	Wireless Communication	Summer 2002	455	...	Mike Franklin
...	...	...	...	...	...

**Students** (SNo, SName, Address, Major, Minor)

<u>SNo</u>	SName	Address	Major	Minor
1001	David Chang	...	Computer Science	Psychology
1002	Sunita Singh	...	Electrical	Computer Science

			<i>Engineering</i>	
1003	Joe Doe	...	<i>Electrical Engineering</i>	<i>Physics</i>
...	...	...	...	...

**Exams** (SNo, CNo, EDate, Grade):

<u>Sno</u>	<u>CNo</u>	EDate	Grade
1001	1	27 July 2001	1.7
1002	2	15 July 2001	2.0
1002	1	28 July 2001	1.3
1003	3	NULL	NULL
...	...	...	...

Das Attribut *Chair* der Relation *Department* ist ein Fremdschlüssel bzgl. *Teacher.TName*; *Courses.Lecturer* ist Fremdschlüssel bzgl. *Teacher.TName*; in der Relation *Students* sind *Major* (Hauptfach) und *Minor* (Nebenfach) Fremdschlüssel bzgl. *Department.DName*.

Formulieren Sie folgende Anfragen in SQL :

- a) Wie heisst der Dekan (*Chair*) des Fachbereichs (*Department*) mit der besten Durchschnittsnote, berechnet über alle Kurse des Sommersemesters 2002?

```

SELECT D1.Chair
FROM Departments D1, Teachers T1, Courses C1, Exams E1
WHERE
  C1.Lecturer = T1.TName AND
  C1.CNo = E1.CNo AND
  D1.DName = T1.DName AND
  C1.Semester = 'Summer 2001' AND
  E2.Grade IS NOT NULL
GROUP BY D1.DName, D1.Chair
HAVING AVG (E1.Grade) >= ALL
(
  SELECT AVG (E2.Grade)
  FROM Teachers T2, Exams E2, Courses C2
  WHERE
    C2.CNo = E2.CNo AND
    C2.Lecturer = T2.TName AND
    C2.Semester = 'Summer 2001' AND
    E2.Grade IS NOT NULL
  GROUP BY T2.DName
)

```

- b) Welche Studenten sind im Nebenfach (*Minor*) ständig erfolgreicher als im Hauptfach (*Major*)? Auszugeben sind die Namen solcher Studenten sowie deren Haupt- und Nebenfach.

```

SELECT S.SName, S.Major, S.Minor
FROM Students S
WHERE S.SNo IN
(

```

```

SELECT S1.SNo
FROM Students S1, Exams E1, Courses C1, Teachers T1
WHERE
    E1.SNo = S1.SNo AND
    E1.CNo = C1.CNo AND
    T1.TName = C1.Lecturer AND
    S1.Minor = T1.DName AND
    E1.Grade IS NOT NULL
GROUP BY S1.SNo
HAVING MAX (E1.Grade) <
(
    SELECT MIN (E2.Grade)
    FROM Students S2, Exams E2, Courses C2, Teachers T2
    WHERE
        E2.SNo = S2.SNo AND
        E2.CNo = C2.CNo AND
        E2.Grade IS NOT NULL AND
        T2.TName = C2.Lecturer AND
        S2.Major = T2.DName AND
        S1.SNo = S2.SNo
))

```

*All Minor-Grades, grouped by Student*

*Min(Major-Grade) of the same student  
> Max (Minor-Grade)*

```

SELECT SName, Major, Minor
FROM
(
    SELECT S.SNo, S.SName, S.Major, S.Minor
    FROM Students S, Exams E, Courses C, Teachers T
    WHERE
        E.SNo = S.SNo AND
        E.CNo = C.CNo AND
        T.TName = C.Lecturer AND
        S.Minor = T.DName AND
        E.Grade IS NOT NULL
    GROUP BY S.SNo, S.SName, S.Major, S.Minor
    HAVING MAX (E.Grade) <
(
    SELECT MIN (E1.Grade)
    FROM Students S1, Exams E1, Courses C1, Teachers T1
    WHERE
        E1.SNo = S1.SNo AND
        E1.CNo = C1.CNo
        T1.TName = C1.Lecturer AND
        S1.Major = T1.DName AND
        S1.SNo = S.SNo AND
        E1.Grade IS NOT NULL
))

```

*All Minor-Grades, grouped by Student*

*Min(Major-Grade) of the same student  
> Max (Minor-Grade)*

zur besseren Lesbarkeit könnte man zusätzlich zwei Views (*min (Major)* and *max (Minor)*) hinzufügen :

```

CREATE VIEW S_MAJOR AS
SELECT S.SNo AS SNo, MIN (E.Grade) AS SMin
FROM Students S, Exams E, Courses C, Teachers T
WHERE
    S.SNo = E.SNo AND

```

*Best Major-Grade per Student*

```
E.CNo = C.CNo AND
E.Grade IS NOT NULL AND
T.TName = C.Lecturer AND
S.Major = T.DName
GROUP BY S.SNo
```

```
CREATE VIEW S_MINOR AS
SELECT S.SNo AS SNo, MAX (E.Grade) AS SMax
FROM Students S, Exams E, Courses C, Teachers T
WHERE
```

```
S.SNo = E.SNo AND
E.CNo = C.CNo AND
T.Tname = C.Lecturer AND
E.Grade IS NOT NULL AND
S.Minor = T.DName
GROUP BY S.Sno
```

*Worst Minor-Grade per Student*

Die Anfrage lautet:

```
SELECT SName, Major, Minor
FROM Students
WHERE SNo IN
( SELECT Minor.SNo
  FROM S_Major Major, S_Minor Minor
  WHERE
    Minor.SNo = Major.SNo AND
    Minor.SMax < Major.SMin
)
```