

Die relationale Algebra

σ Selektion

π Projektion

\times Kreuzprodukt

\bowtie Join (Verbund)

ρ Umbenennung

\ltimes Semi-Join (linker)

\rtimes Semi-Join (rechter)

$\ltimes\bowtie$ linker äußerer Join

$\rtimes\bowtie$ rechter äußerer Join

Allg. Mengenoperationen:

– Differenz

\div Division

\cup Vereinigung

\cap Durchschnitt

Beispiel Mengendurchschnitt

Finde die *PersNr* aller C4-Professoren, die mindestens eine Vorlesung halten.

$$\Pi_{\text{PersNr}}(\rho_{\text{PersNr} \leftarrow \text{gelesenVon}}(\text{Vorlesungen})) \cap \Pi_{\text{PersNr}}(\sigma_{\text{Rang}=\text{C4}}(\text{Professoren}))$$

Relationaler Tupelkalkül

Eine Anfrage im Relationenkalkül hat die Form

$$\{t \mid P(t)\}$$

mit t Tupelvariable und P Prädikat

einfaches **Beispiel:**

C4-Professoren

$$\{p \mid p \in \text{Professoren} \wedge p.\text{Rang} = \text{'C4'}\}$$

Relationenkalkül: Beispiel

Studenten mit mindestens einer Vorlesung von Curie

$$\{s \mid s \in \text{Studenten} \\ \wedge \exists h \in \text{hören}(s.\text{MatrNr}=h.\text{MatrNr} \\ \wedge \exists v \in \text{Vorlesungen}(h.\text{VorlNr}=v.\text{VorlNr} \\ \wedge \exists p \in \text{Professoren}(p.\text{PersNr}=v.\text{gelesenVon} \\ \wedge p.\text{Name} = \text{'Curie'})))))\}$$

Dieselbe Anfrage in SQL belegt die Verwandtschaft

```
select s.*
from Studenten s
where exists (
  select h.*
  from hören h
  where h.MatrNr = s.MatrNr and exists (
    select *
    from Vorlesungen v
    where v.VorlNr = h.VorlNr and exists (
      select *
      from Professoren p
      where p.Name = 'Curie' and
            p.PersNr = v.gelesenVon )))
```

Relationaler Domänenkalkül

Anfrage im Domänenkalkül hat die Form:

$$\{[v_1, v_2, \dots, v_n] \mid P(v_1, \dots, v_n)\}$$

mit v_1, \dots, v_n Domänenvariablen und P Prädikat

Beispiel:

MatrNr und Namen der Prüflinge von Sokrates

$$\{[m, n] \mid \exists ([m, n, s] \in \text{Studenten} \\ \wedge \exists p, v, g ([m, p, v, g] \in \text{prüfen} \\ \wedge \exists a, r, b ([p, a, r, b] \in \text{Professoren} \\ \wedge a = \text{'Sokrates'})))]\}$$

Ausdruckskraft

Die drei Sprachen

- relationale Algebra
 - relationaler Tupelkalkül, eingeschränkt auf sichere Ausdrücke
 - relationaler Domänenkalkül, eingeschränkt auf sichere Ausdrücke
- sind gleich mächtig

$\{n \mid \neg(n \in \text{Professoren})\}$ z.B. ist nicht sicher, da das Ergebnis unendlich ist

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorINr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorINr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022

Assistenten			
PersINr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

prüfen			
MatrNr	VorINr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	5041	2125	2

Mengenoperationen

- In SQL gibt es auch die üblichen Operationen auf Mengen:
Vereinigung, Schnitt und Differenz
- Setzen wie in der relationalen Algebra gleiches Schema der verknüpften Ausgabe-Relationen voraus

(**select** Name
from Assistenten)

union

(**select** Name
from Professoren);

Duplikateliminierung

- Im Gegensatz zu **select** eliminiert **union** automatisch Duplikate
- Falls Duplikate im Ergebnis erwünscht sind, muss der **union all**-Operator benutzt werden

Schnitt, Mengendifferenz

Professoren **und** Assistenten

```
select Name from Professoren
```

```
intersect
```

```
select Name from Assistenten;
```

Professoren, **aber nicht** Assistenten

```
select Name from Professoren
```

```
except
```

```
select Name from Assistenten;
```

Sortierung

- Tupel in einer Relation sind nicht (automatisch) sortiert
- Ergebnis einer Anfrage kann mit Hilfe der **order by**-Klausel sortiert werden
- Es kann aufsteigend oder absteigend sortiert werden
- Default Sortierung: aufsteigend

Beispiel

```
select *  
from Studenten  
order by Name, Semester desc;
```

Geschachtelte Anfragen

- Anfragen können in anderen Anfragen geschachtelt sein, d.h. es kann mehr als eine select-Klausel geben
- Geschachteltes select kann in der where-Klausel, in der from-Klausel und sogar in einer select-Klausel selbst auftauchen
- Im Prinzip wird in der "inneren" Anfrage ein Zwischenergebnis berechnet, das in der "äußeren" benutzt wird

Select in Where-Klausel

- Zwei verschiedene Arten von Unteranfragen: korrelierte und unkorrelierte
- unkorreliert: Unteranfrage bezieht sich nur auf "eigene" Attribute
- korreliert: Unteranfrage referenziert auch Attribute der äußeren Anfrage

Unkorrelierte Unteranfrage

Namen aller Studenten, die VorlNr 5041 hören

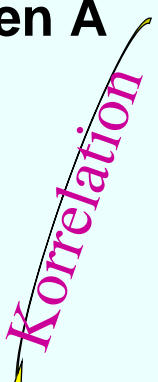
```
select S.Name  
from Studenten S  
where S.MatrNr in  
(select h.MatrNr  
from hoeren h  
where h.VorlNr = 5041);
```

- Unteranfrage wird einmal ausgewertet
- für jedes Tupel der äußeren Anfrage wird geprüft, ob die MatrNr im Ergebnis der Unteranfrage vorkommt

Korrelierte Unteranfrage

Finde alle Professoren, für die Assistenten mit unterschiedlichen Fachgebieten arbeiten

```
select distinct P.Name
from Professoren P, Assistenten A
where A.Boss = P.PersNr
and exists
(select *
from Assistent B
where B.Boss = P.PersNr and A.Fachgebiet <> B.Fachgebiet);
```



- Für jedes Tupel der äußeren Anfrage hat innere Anfrage verschiedene Werte
- das exists-Prädikat ist wahr, wenn die Unteranfrage mind. ein Tupel enthält

Existenzquantor exists

```
select P.Name  
from Professoren P  
where not exists ( select *  
                    from Vorlesungen V  
                    where V.gelesenVon = P.PersNr );
```

Existenzquantor exists

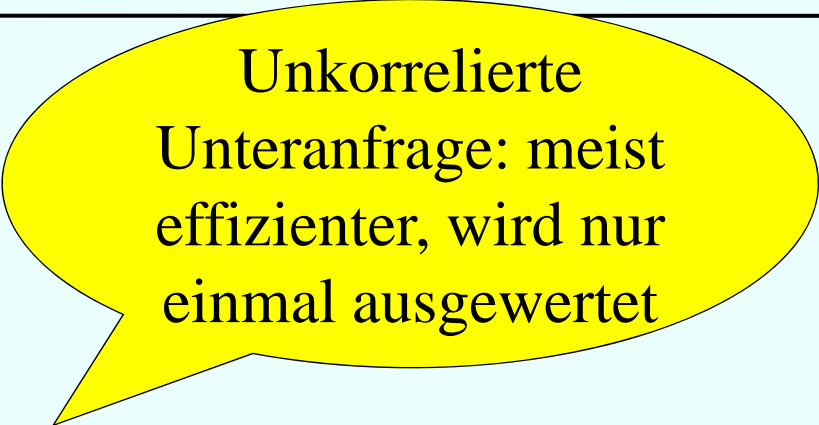
```
select P.Name  
from Professoren P  
where not exists ( select *  
                   from Vorlesungen V  
                   where V.gelesenVon = P.PersNr );
```



Korrelation

Mengenvergleich

```
select Name  
from Professoren  
where PersNr not in ( select gelesenVon  
                        from Vorlesungen );
```



Unkorrelierte
Unteranfrage: meist
effizienter, wird nur
einmal ausgewertet

Unkorrelierte versus korrelierte Unteranfragen

- korrelierte Formulierung

```
select s.*  
from Studenten s  
where exists  
    (select p.*  
     from Professoren p  
     where p.GebDatum > s.GebDatum);
```

Umformulierung

- Äquivalente unkorrelierte Formulierung

```
select s.*
```

```
from Studenten s
```

```
where s.GebDatum <
```

```
    (select max (p.GebDatum)
```

```
    from Professoren p);
```

- Vorteil: Unteranfrageergebnis kann materialisiert werden
- Unteranfrage braucht nur einmal ausgewertet zu werden

Entschachtelung korrelierter Unteranfragen -- Forts.

```
select a.*  
from Assistenten a  
where exists  
  ( select p.*  
    from Professoren p  
    where a.Boss = p.PersNr and p.GebDatum > a.GebDatum);
```

- Entschachtelung durch Join

```
select a.*  
from Assistenten a, Professoren p  
where a.Boss=p.PersNr and p.GebDatum > a.GebDatum;
```

Professoren			
PersNr	Name	Rang	Raum
2125	Sokrates	C4	226
2126	Russel	C4	232
2127	Kopernikus	C3	310
2133	Popper	C3	52
2134	Augustinus	C3	309
2136	Curie	C4	36
2137	Kant	C4	7

Studenten		
MatrNr	Name	Semester
24002	Xenokrates	18
25403	Jonas	12
26120	Fichte	10
26830	Aristoxenos	8
27550	Schopenhauer	6
28106	Carnap	3
29120	Theophrastos	2
29555	Feuerbach	2

Vorlesungen			
VorINr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

voraussetzen	
Vorgänger	Nachfolger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

hören	
MatrNr	VorINr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022

Assistenten			
PersINr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

prüfen			
MatrNr	VorINr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	5001	2126	1
29120	5041	2125	2

Aggregatfunktion und Gruppierung

Aggregatfunktionen **avg, max, min, count, sum**

```
select avg (Semester)  
from Studenten ;
```

```
select gelesenVon, sum (SWS)  
from Vorlesungen  
group by gelesenVon;
```

Aggregatfunktion und Gruppierung

```
select gelesenVon, Name, sum (SWS)
from Vorlesungen, Professoren
where gelesenVon = PersNr and Rang = 'C4'
group by gelesenVon, Name
having avg (SWS) >= 3;
```

Besonderheiten bei Aggregatoperationen

- SQL erzeugt pro Gruppe ein Ergebnistupel
- alle in der **select**-Klausel aufgeführten Attribute - außer den aggregierten – müssen auch in der **group by**-Klausel aufgeführt werden
- Nur so kann SQL sicherstellen, dass sich das Attribut nicht innerhalb der Gruppe ändert

Anfrage mit group by (Equi-Join, Selektion Rang='C4')

Vorlesung x Professoren							
Vorl Nr	Titel	SWS	gelesen Von	PersNr	Name	Rang	Raum
5001	Grundzüge	4	2137	2125	Sokrates	C4	226
5041	Ethik	4	2125	2125	Sokrates	C4	226
...
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ **where**-Bedingung

Gruppieren nach gelesenVon, Name

VorlNr	Titel	SWS	gelesen Von	PersNr	Name	Rang	Raum
5001	Grundzüge	4	2137	2137	Kant	C4	7
5041	Ethik	4	2125	2125	Sokrates	C4	226
5043	Erkenntnistheorie	3	2126	2126	Russel	C4	232
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5052	Wissenschaftstheorie	3	2126	2126	Russel	C4	232
5216	Bioethik	2	2126	2126	Russel	C4	232
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ Gruppierung

Nur Gruppen mit mindestens 3 SWS im Schnitt

VorlNr	Titel	SWS	gelesenVon	PersNr	Name	Rang	Raum
5041	Ethik	4	2125	2125	Sokrates	C4	226
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5043	Erkenntnistheorie	3	2126	2126	Russel	C4	232
5052	Wissenschaftstheo.	3	2126	2126	Russel	C4	232
5216	Bioethik	2	2126	2126	Russel	C4	232
5001	Grundzüge	4	2137	2137	Kant	C4	7
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ **having** Bedingung

Summenbildung über SWS und Projektion

VorlNr	Titel	SWS	gelesenVon	PersNr	Name	Rang	Raum
5041	Ethik	4	2125	2125	Sokrates	C4	226
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5001	Grundzüge	4	2137	2137	Kant	C4	7
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

↓ Aggregation (**sum**) und Projektion

Ergebnis

gelesenVon	Name	sum (SWS)
2125	Sokrates	10
2137	Kant	8

Maximum / Minimum

Gib mir den Studenten mit der größten MatrNr

```
select MatrNr, Name  
from Student  
where MatrNr =  
    (select max(MatrNr)  
    from Student);
```

NICHT

```
select Name, max(MatrNr)  
from Student;
```