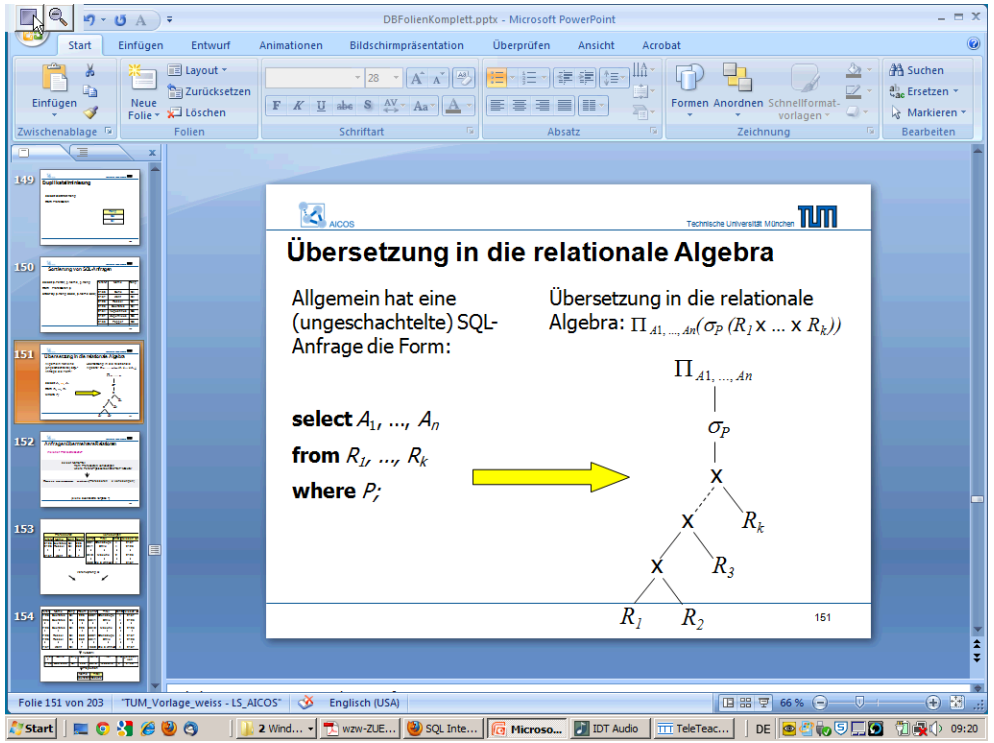
Script generated by TTT

Title: Groh: wzw_2012 (25.05.2012)

Date: Fri May 25 09:20:09 CEST 2012

Duration: 87:34 min

Pages: 54



Übersetzung in die relationale Algebra

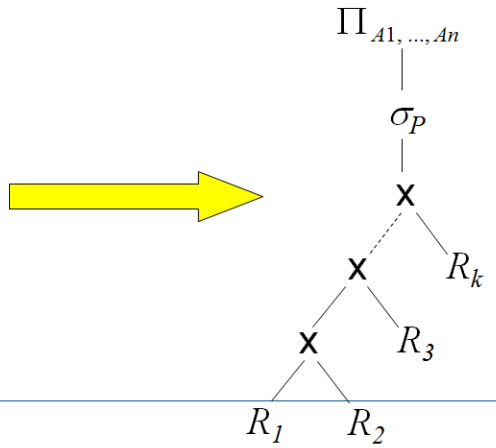
Allgemein hat eine (ungeschachtelte) SQL-Anfrage die Form:

Übersetzung in die relationale Algebra: $\Pi_{A1,\ldots,An}(\sigma_P(R_1 \times \ldots \times R_k))$

select $A_1, \ldots, A_n$

from $R_1, \ldots, R_k$

where P;



Anfragen über mehrere Relationen

Welcher Prof liest Mäeutik?

```
select Name,Titel
        from Professoren, Vorlesungen
        where PersNr=gelesenVon and Titel='Mäeutik';
```

$\Pi_{Name,Titel}\sigma_{PersNr=gelesenVon \wedge Titel='Mäeutik'}(Professoren \times Vorlesungen)$

(siehe auch Blatt 3 Aufgabe 1)

Mengenoperationen und geschachtelte Anfragen

Mengenoperationen **union, intersect, minus**

```
( select Name
  from Assistenten )
union
( select Name
  from Professoren);
```


Existenzquantor exists

Profs die keine Vorlesung halten:

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

Korrelation

Mengenvergleich

Profs die keine Vorlesung halten (Variante mit 'in'):

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

Unkorrelierte Unteranfrage: meist effizienter, wird nur einmal ausgewertet

Der Vergleich mit "all"

```
select Name
from Studenten
where Semester >= all ( select Semester
                        from Studenten);
```

größer gleich jeder anderen Zahl der Menge

effizienter:

```
...
where Semester >= ( select max(Semester)
                    from Studenten);
```

Bemerkung (1) : neben **all** gibt es auch **any**

Der Vergleich mit "all"

```
select Name
from Studenten
where Semester >= all ( select Semester
                        from Studenten);
```

größer gleich jeder anderen Zahl der Menge

Bemerkung (1) : neben **all** gibt es auch **any**

Bemerkung (2): entspricht NICHT dem Allquantor;
"Studis die **alle** 4std Vorl. hoeren" geht damit NICHT

Aggregatfunktion und Gruppierung

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

- **select avg**(Semester) **from** Studenten;

Gruppierung

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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfunktion und Gruppierung

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

- **select avg**(Semester) **from** Studenten;

Gruppierung

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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfkt. wird für jede Gruppe getrennt berechnet

sortiert bstimmte Gruppen aus

wie gehabt: sortiert Tupel aus Kreuzprodukt aus (Theta Join)

Aggregatfunktion und Gruppierung

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

- **select avg**(Semester) **from** Studenten;

Gruppierung

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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfkt. wird für jede Gruppe getrennt berechnet

sortiert bstimmte Gruppen aus

wie gehabt: sortiert Tupel aus Kreuzprodukt aus (Theta Join)

Ausführen einer Anfrage mit group by

Vorlesung x Professoren							
VorlNr	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

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

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

VorlN	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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfkt. wird für jede Gruppe getrennt berechnet

sortiert bstimmte Gruppen aus

wie gehabt: sortiert Tupel aus Kreuzprodukt aus (Theta Join)

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

VorlN	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

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

VorlN	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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfkt. wird für jede Gruppe getrennt berechnet

sortiert bstimmte Gruppen aus

wie gehabt: sortiert Tupel aus Kreuzprodukt aus (Theta Join)

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfkt. wird für jede Gruppe getrennt berechnet

sortiert bstimmte Gruppen aus

wie gehabt: sortiert Tupel aus Kreuzprodukt aus (Theta Join)

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

VorlN	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

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

Besonderheiten bei Aggregatoperationen

- SQL erzeugt dann pro Gruppe nur ein Ergebnistupel
- Deshalb dürfen - außer den aggregierten – in der **select**-Klausel nur Attribute aufgeführt werden, die auch in der **group by**-Klausel aufgeführt werden
- Nur so kann SQL sicherstellen, dass sich ein Attribut nicht innerhalb der Gruppe ändert

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

VorlN	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

Aggregatfunktion und Gruppierung

Welcher C4 Professor liest im Durchschnitt lange Vorlesungen?

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

Aggregatfkt. wird für jede Gruppe getrennt berechnet

sortiert bstimmte Gruppen aus

wie gehabt: sortiert Tupel aus Kreuzprodukt aus (Theta Join)

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

Besonderheiten bei Aggregatoperationen

- SQL erzeugt dann pro Gruppe nur ein Ergebnistupel
- Deshalb dürfen - außer den aggregierten – in der **select**-Klausel nur Attribute aufgeführt werden, die auch in der **group by**-Klausel aufgeführt werden
- Nur so kann SQL sicherstellen, dass sich ein Attribut nicht innerhalb der Gruppe ändert

Beispiele:

- Wie viele Vorlesungen besucht jeder Student?

Besonderheiten bei Aggregatoperationen

- SQL erzeugt dann pro Gruppe nur ein Ergebnistupel
- Deshalb dürfen - außer den aggregierten – in der **select**-Klausel nur Attribute aufgeführt werden, die auch in der **group by**-Klausel aufgeführt werden
- Nur so kann SQL sicherstellen, dass sich ein Attribut nicht innerhalb der Gruppe ändert

Beispiele:

- Wie viele Vorlesungen besucht jeder Student?

select count(h.VorlNr), h.MatrNr, s.name
from hoeren h, Studenten s
where h.MatrNR = s.MatrNR
group by h.MatrNr, s.Name

- **…und wie viele Stunden sitzt jeder Student je Woche in Vorlesungen?**

select count(h.VorlNr) as VLanz, h.MatrNr, s.name, **sum(v.SWS) as Praesenzzeit**
from hoeren h, Studenten s, **Vorlesungen v**
where h.MatrNR = s.MatrNR **and v.VorlNR = h.VorlNr**
group by h.MatrNr, s.Name

- **Verständnisfrage: Welche Anfrage bringt mehr Ergebnistupel?**
  - select g, sum(alter) from R group by g
  - select g, sum(alter) from R group by g, h

Beispiele:

- Wie viele Vorlesungen besucht jeder Student?

select count(h.VorlNr), h.MatrNr, s.name
from hoeren h, Studenten s
where h.MatrNR = s.MatrNR
group by h.MatrNr, s.Name

- **…und wie viele Stunden sitzt jeder Student je Woche in Vorlesungen?**

```
select count(h.VorlNr) as VLanz, h.MatrNr, s.name, sum(v.SWS) as Praesenzzeit
    from hoeren h, Studenten s, Vorlesungen v
    where h.MatrNR = s.MatrNR and v.VorlNR = h.VorlNr
    group by h.MatrNr, s.Name
```

- **Verständnisfrage: Welche Anfrage bringt mehr Ergebnistupel?**
  - select g, sum(alter) from R group by g
  - select g, sum(alter) from R group by g, h

  - Je mehr Attribute nach group by umso mehr Gruppen
  - drill down (mehr Details weniger Überblick - vgl. Flugzeug) vs.
  - role up (mehr Überblick aber keine Details mehr)

weiteres Bsp: Blatt 3, Aufgabe 2.2:

Wer liest mindestens 2 Vorlesungen?

$$\Pi_{Professoren.Name}\Big(Professoren \bowtie \rho_{PersNr \leftarrow gelesenVon}\big((\sigma_{count(*)\geq 2}(\gamma_{gelesenVon,count(*)}Vorlesungen))\big)\Big)$$

```
select Name,
        from Vorlesungen, Professoren
        where gelesenVon = PersNr
        group by gelesenVon,Name
                having count(*)>=2
```


Geschachtelte Anfrage

- Unteranfrage in der **where**-Klausel
- Welche Prüfungen sind besser als durchschnittlich verlaufen?

```
select *
from prüfen p
where p.Note < ( select avg (Note)
                 from prüfen );
```


Geschachtelte Anfrage (Forts.)

- Unteranfrage in der **select**-Klausel
- Für jedes Ergebnistupel wird die Unteranfrage ausgeführt
- Man beachte, dass die Unteranfrage korreliert ist (greift auf Attribute der umschließenden Anfrage zu)

```
select p.PersNr, p.Name, ( select sum (v.SWS) as
                                Lehrbelastung from Vorlesungen v
                           where v.gelesenVon=p.PersNr )
from Professoren p;
```

Korrelierte versus unkorrelierte Unteranfragen

Welche Studenten sind älter als Professoren?

- korrelierte Formulierung

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


- Äquivalente unkorrelierte Formulierung

```
select s.*
from Studenten s
where s.GebJahr <
        (select max (p.GebJahr)
        from Professoren p);
```

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

Entschachtelung korrelierter Unteranfragen durch Join

Welcher Assistent hat einen Boss der jünger ist als er selbst?

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

- Entschachtelung durch Join

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


Verwertung der Ergebnismenge einer Unteranfrage

```
select tmp.MatrNr, tmp.Name, tmp.VorlAnzahl
from (select s.MatrNr, s.Name, count(*) as VorlAnzahl
      from Studenten s, hören h
      where s.MatrNr=h.MatrNr
      group by s.MatrNr, s.Name)  tmp
where tmp.VorlAnzahl > 2;
```

Wer hört mehr als 2 Vorlesungen?

MatrNr	Name	VorlAnzahl
28106	Carnap	4
29120	Theophrastos	3

Beispiel für eine komplexere Anfrage mit geschachtelten Unteranfragen und Cast

```
select  h.VorlNr, h.AnzStudiProVorl, g.GesamtAnz,
        cast(h.AnzStudiProVorl as decimal(6,2)) / g.GesamtAnz
        as Marktanteil
from    ( select VorlNr, count(*) as AnzStudiProVorl
          from hören
          group by VorlNr ) h,
        ( select count(*) as GesamtAnz
          from Studenten) g;
```


Allquantifizierung durch count-Aggregation

- Allquantifizierung kann immer auch durch eine **count**-Aggregation ausgedrückt werden
- Wir betrachten dazu eine etwas einfachere Anfrage, in der wir die *MatrNr* der Studenten ermitteln wollen, die *alle* Vorlesungen hören:

```
select h.MatrNr
from hören h
group by h.MatrNr
        having count (*) = (select count (*) from Vorlesungen);
```

Hinweis: count(*) nach having zählt alle Tupel einer Gruppe!!!!

Allquantifizierung durch count-Aggregation

- Allquantifizierung kann immer auch durch eine **count**-Aggregation ausgedrückt werden
- Wir betrachten dazu eine etwas einfachere Anfrage, in der wir die *MatrNr* der Studenten ermitteln wollen, die *alle* Vorlesungen hören:

```
select h.MatrNr
from hören h
group by h.MatrNr
        having count (*) = (select count (*) from Vorlesungen);
```

Hinweis: count(*) nach having zählt alle Tupel einer Gruppe!!!!

Nullwerte

- unbekannter Wert
- wird vielleicht später nachgereicht
- Nullwerte können auch im Zuge der Anfrageauswertung entstehen (Bsp. äußere Joins)
- manchmal sehr überraschende Anfrageergebnisse, wenn Nullwerte vorkommen

```
select count (*)
from Studenten
where Semester < 13 or Semester > =13
```

- Wenn es Studenten gibt, deren *Semester*-Attribut den Wert **null** hat, werden diese nicht mitgezählt
- Der Grund liegt in folgenden Regeln für den Umgang mit **null**-Werten begründet:

Auswertung bei Null-Werten

1. In arithmetischen Ausdrücken werden Nullwerte propagiert, d.h. sobald ein Operand **null** ist, wird auch das Ergebnis **null**. Dementsprechend wird z.B. **null** + 1 zu **null** ausgewertet. Aber auch null * 0 wird zu **null** ausgewertet.
2. SQL hat eine dreiwertige Logik, die nicht nur **true** und **false** kennt, sondern auch einen dritten Wert **unknown**. Diesen Wert liefern Vergleichsoperationen zurück, wenn mindestens eines ihrer Argumente **null** ist. Beispielsweise wertet SQL das Prädikat *(PersNr=...)* immer zu **unknown** aus, wenn die *PersNr* des betreffenden Tupels den Wert **null** hat.
3. Logische Ausdrücke werden nach den folgenden Tabellen berechnet:

and	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

not	
true	false
unknown	unknown
false	true

or	true	unknown	false
true	true	true	true
unknown	true	unknown	unknown
false	true	unknown	false

Spezielle Sprachkonstrukte ("syntaktischer Zucker")

```
select * from Studenten
where Semester > = 1 and Semester < = 4;
```

```
select *
from Studenten
where Semester between 1 and 4;
```

```
select *
from Studenten
where Semester in (1,2,3,4);
```


Vergleiche mit like

Platzhalter "%" ; "_"

- "%" steht für beliebig viele (auch gar kein) Zeichen
- "_" steht für genau ein Zeichen

```
select * from Studenten
where Name like ´T%eophrastos´;
```

```
select distinct Name
from Vorlesungen v, hören h, Studenten s
where s.MatrNr = h.MatrNr and h.VorlNr = v.VorlNr and
      v.Titel = ´%thik%´;
```

Joins in SQL-92

- **cross join:** Kreuzprodukt
- **natural join:** natürlicher Join
- Join oder inner join: Theta-Join
- left, right oder full outer join: äußerer Join

```
select *
from R1, R2
where = R1.A = R2.B;

select *
from R1 join R2 on R1.A = R2.B;
```


AICOS Technische Universität München

Joins in SQL-92

- **cross join:** Kreuzprodukt
- **natural join:** natürlicher Join
- Join oder inner join: Theta-Join
- left, right oder full outer join: äußerer Join

```
select *
from R1, R2
where  R1.A = R2.B;

select *
from R1 join R2 on R1.A = R2.B;
```

190

AICOS Technische Universität München

Joins in SQL-92

- **cross join:** Kreuzprodukt
- **natural join:** natürlicher Join
- Join oder inner join: Theta-Join
- left, right oder full outer join: äußerer Join

```
select *
from R1, R2
where  R1.A = R2.B;

select *
from R1 join R2 on R1.A = R2.B;
```

190