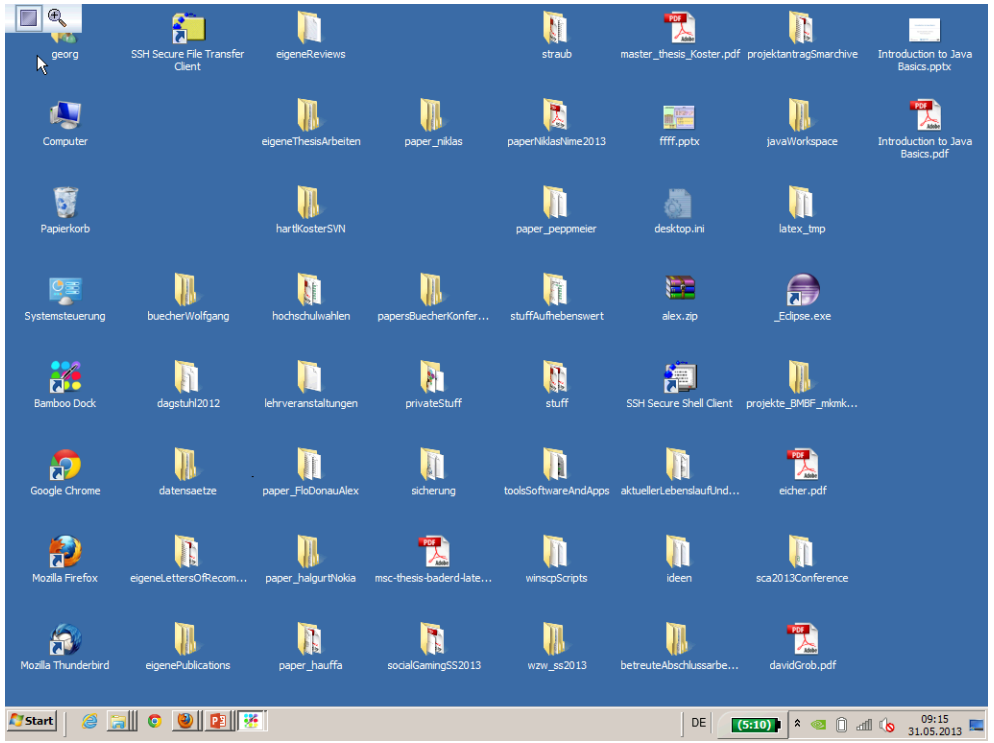
Script **generated by TTT**

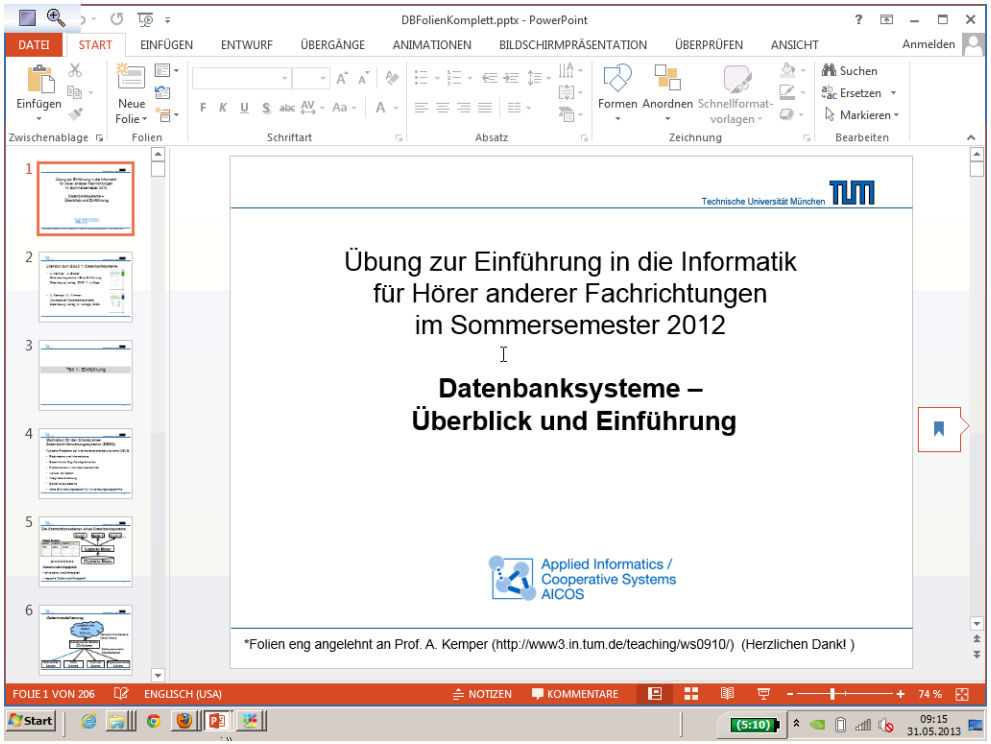
Title: profile1 (31.05.2013)

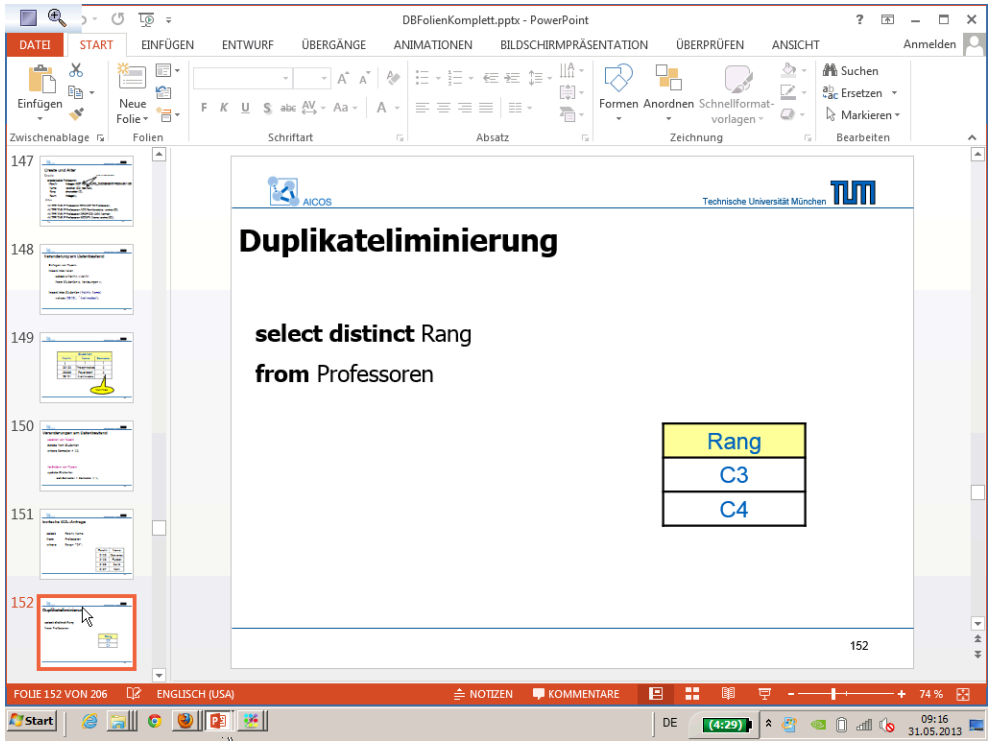
Date: Fri May 31 09:15:57 CEST 2013

Duration: 83:23 min

Pages: 45







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)

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

AICOS Technische Universität München

- **…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

AICOS Technische Universität München

- **…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

AICOS Technische Universität München

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

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
```


AICOS Technische Universität München

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
```

174

AICOS Technische Universität München

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;
```


AICOS Technische Universität München

- **…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

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);
```


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);
```


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;
```

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;
```


AICOS Technische Universität München
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;
179
AICOS Technische Universität München
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;
179
AICOS Technische Universität München
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;
179

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

AICOS Technische Universität München

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
180

AICOS Technische Universität München

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
180

AICOS Technische Universität München

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
180

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;
```


AICOS Technische Universität München

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;
```

181

AICOS Technische Universität München

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;
```

181

AICOS Technische Universität München

Die gleiche Anfrage nur übersichtlicher

- Nicht in MySQL aber in DB2 oder Oracle
  - **with tempRelName as (select …)**
  - Definiere temporäre Relationen und verwende sie in komplexen Anfragen --> modularer und übersichtlicher

```
with h as (…), g as (…);
select h.VorlNr, h.AnzProVorl, g.GesamtAnz,
       cast(h.AnzProVorl as decimal(6,2)) / g.GesamtAnz
       as Marktanteil
from   h, g;
```

Weitere Anfragen mit Unteranfragen (WDH)

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

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

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


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!!!!

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!!!!

finde die Studenten die alle vierstündigen Vorlesungen hören:

```
select s.*
        from Studenten s
        where not exists
                (select v.*
                from Vorlesungen v
                where v.SWS=4 and not exists
                        (select h.*
                        from hören h
                        where h.VorlNr = v.VorlNr and h.MatrNr = s.MatrNr));
```

⟶

```
select h.MatrNr
        from hören h
        group by h.MatrNr
                having count (*) = (select count (*)
                        from Vorlesungen v where v.SWS=4);
```


finde die Studenten die alle vierstündigen Vorlesungen hören:

```
select s.*
        from Studenten s
        where not exists
                (select v.*
                from Vorlesungen v
                where v.SWS=4 and not exists
                        (select h.*
                        from hören h
                        where h.VorlNr = v.VorlNr and h.MatrNr = s.MatrNr));
```

⟶

```
select h.MatrNr
        from hören h
        group by h.MatrNr
                having count (*) = (select count (*)
                        from Vorlesungen v where v.SWS=4);
```


finde die Studenten die alle vierstündigen Vorlesungen hören:

```
select s.*
        from Studenten s
        where not exists
                (select v.*
                from Vorlesungen v
                where v.SWS=4 and not exists
                        (select h.*
                        from hören h
                        where h.VorlNr = v.VorlNr and h.MatrNr = s.MatrNr));
```

⟶

```
select h.MatrNr
        from hören h
        group by h.MatrNr
                having count (*) = (select count (*)
                        from Vorlesungen v where v.SWS=4);
```

finde die Studenten die alle vierstündigen Vorlesungen hören:

```
select s.*
    from Studenten s
    where not exists
        (select v.*
        from Vorlesungen v
        where v.SWS=4 and not exists
            (select h.*
            from hören h
            where h.VorlNr = v.VorlNr and h.MatrNr = s.MatrNr));
```

⟶

```
select h.MatrNr
    from hören h
    group by h.MatrNr
        having count (*) = (select count (*)
            from Vorlesungen v where v.SWS=4);
```


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:

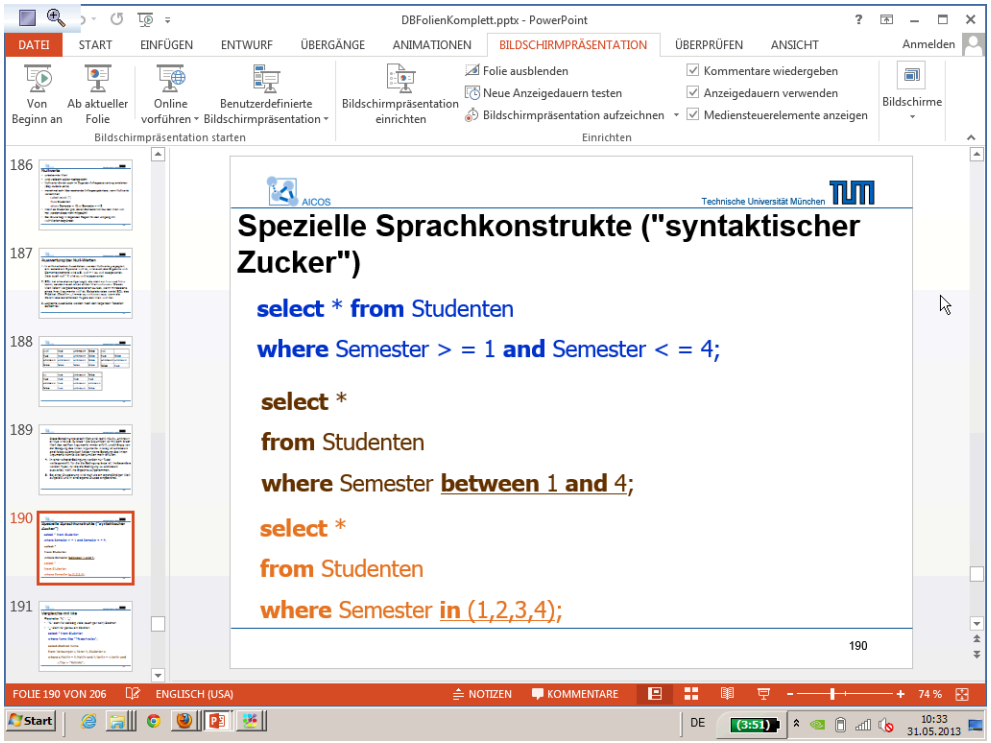
and	true	unknown	false
true	true	unknown	false
unknown	unknown	unknown	false
false	false	false	false

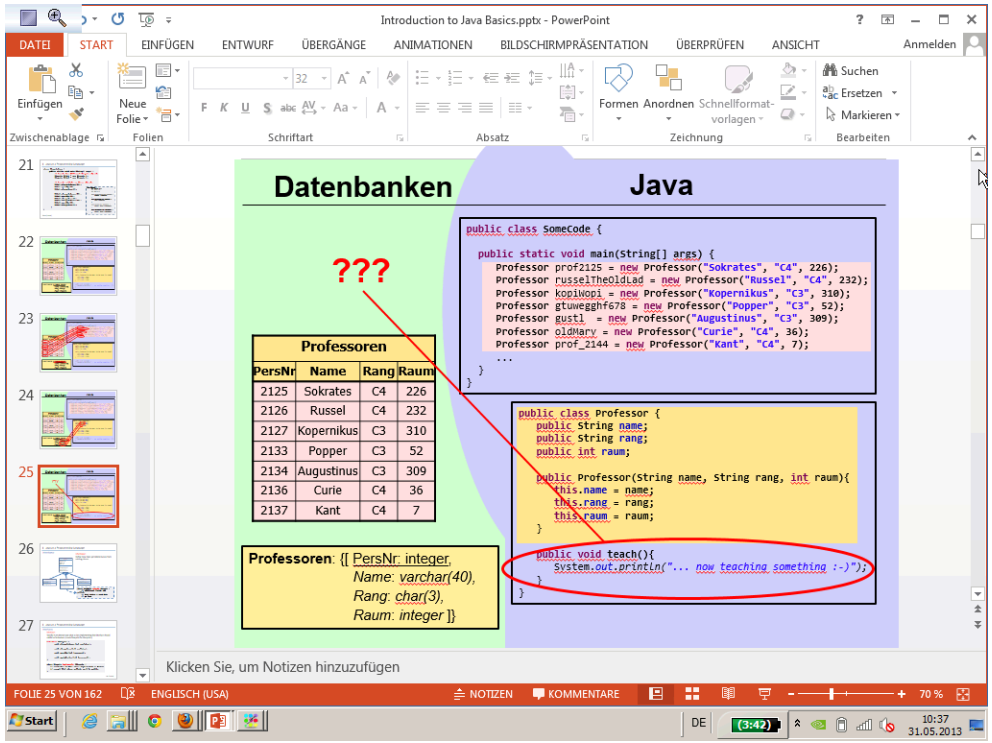
not	
true	false
unknown	unknown
false	true

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

Diese Berechnungsvorschriften sind recht intuitiv. Unknown or true wird z.B. zu **true** - die Disjunktion ist mit dem **true**-Wert des rechten Arguments immer erfüllt, unabhängig von der Belegung des linken Arguments. Analog ist **unknown and false** automatisch **false** - keine Belegung des linken Arguments könnte die Konjunktion mehr erfüllen.

4. In einer **where**-Bedingung werden nur Tupel weitergereicht, für die die Bedingung **true** ist. Insbesondere werden Tupel, für die die Bedingung zu **unknown** auswertet, nicht ins Ergebnis aufgenommen.
5. Bei einer Gruppierung wird **null** als ein eigenständiger Wert aufgefaßt und in eine eigene Gruppe eingeordnet.


DBFolienKomplett.pptx - PowerPoint
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);
190
FOLIE 190 VON 206
ENGLISCH (USA)


Introduction to Java Basics.pptx - PowerPoint
Datenbanken
Java
???
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
Professoren: {[PersNr: integer, Name: varchar(40), Rang: char(3), Raum: integer]}
public class SomeCode {
public static void main(String[] args) {
Professor prof2125 = new Professor("Sokrates", "C4", 226);
Professor russelTheOldLad = new Professor("Russel", "C4", 232);
Professor kopiWopi = new Professor("Kopernikus", "C3", 310);
Professor gtuwegghf678 = new Professor("Popper", "C3", 52);
Professor gustl = new Professor("Augustinus", "C3", 309);
Professor oldMary = new Professor("Curie", "C4", 36);
Professor prof_2144 = new Professor("Kant", "C4", 7);
public class Professor {
public String name;
public String rang;
public int raum;
public Professor(String name, String rang, int raum){
this.name = name;
this.rang = rang;
this.raum = raum;
}
public void teach(){
System.out.println("... now teaching something :-)");
}
}
Klicken Sie, um Notizen hinzuzufügen
FOLIE 25 VON 162
ENGLISCH (USA)