SQL (Teil 3)

Die Folien bauen auf dem Foliensatz zum Buch und den von Prof. Dr.-Ing. Norbert Fuhr in den vergangenen Semestern verwendeten Folien auf.

Datenbanken | Prof. Dr. Dirk Lewandowski | WS 2022/23

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Wiederholung

- Aggregatfunktionen
- Geschachtelte Anfragen
- Quantifizierung in SQL
- Nullwerte
- Spezielle Sprachelemente

Wiederholung

Datenbanken: SQL

Aggregatfunktion und Gruppierung

Aggregatfunktionen

avg, max, min, count, sum

1
```
select avg (s.Semester)
from Studenten s;
```

2
```
select v.gelesenVon, sum (v.SWS)
from Vorlesungen v
group by v.gelesenVon;
```

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

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Ausführen einer Anfrage mit group by

Vorlesungen x Professoren							
VorlNr	Titel	SWS	gelesenVon	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

Wiederholung

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

VorlNr	Titel	SWS	gelesenVon	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	2126	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

Wiederholung

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	Wissenschaftsth.	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

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

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Endergebnis

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

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Unkorrelierte versus korrelierte Unteranfragen

Korrelierte Formulierung

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

Äquivalente unkorrelierte Formulierung

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

- Vorteil: Unteranfrage braucht nur einmal ausgewertet zu werden

Quantifizierte Anfragen in SQL

- Existenzquantor: **exists**

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

Der Vergleich mit "all"

Kein vollwertiger Allquantor!

```
select s.Name
from Studenten s
where s.Semester >= all (
        select s1.Semester from Studenten s1);
```

Spezielle Sprachelemente

- **between** und **in**
- **like**
- **case**

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

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

Platzhalter "%" ; "_"

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

```
select p.MatrNr, ( case
        when p.Note < 1.5 then ´sehr gut´
            when p.Note < 2.5 then ´gut´
            when p.Note < 3.5 then ´befriedigend´
            when p.Note < 4.0 then ´ausreichend´
            else ´nicht bestanden´ end)
from prüfen p;
```

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Joins

Das relationale Modell

Theta-Join-Beispiel

Vorlesungen $\bowtie_{gelesenVon=PersNr}$ Professoren

VorlNr	Titel	SWS	gelesenVon	PersNr	Name	Rang	Raum
5001	Grundzüge	4	2137	2137	Kant	C4	7
5041	Ethik	4	2125	2125	Sokrates	C4	226
5043	Erkenntnis-theorie	3	2126	2126	Russel	C4	232
5049	Mäeutik	2	2125	2125	Sokrates	C4	226
4052	Logik	4	2125	2125	Sokrates	C4	226
5052	Wissenschafts-theorie	3	2126	2126	Russel	C4	232
5216	Bioethik	2	2126	2126	Russel	C4	232
5259	Der Wiener Kreis	2	2133	2133	Popper	C3	52
5022	Glaube und Wissen	2	2134	2134	Augustinus	C3	339
4630	Die 3 Kritiken	4	2137	2137	Kant	C4	7

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Andere Join-Arten

Natürlicher Join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$\bowtie$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1

Linker äußerer Join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

$⟕$

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Andere Join-Arten

Rechter äußerer Join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟕

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
-	-	c_3	d_2	e_2

Äußerer Join

L		
A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

⟗

R		
C	D	E
c_1	d_1	e_1
c_3	d_2	e_2

=

Resultat				
A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	-	-
-	-	c_3	d_2	e_2

Joins

- Arten von Joins
- Äußere Joins

Arten von Joins

- **cross join:** Kreuzprodukt
- **natural join:** natürlicher Join
- **join** oder **inner join:** Theta-Join
- **left outer join**, **right outer join** oder **full outer join:** äußerer Join
- **union join:** Vereinigungs-Join (wird hier nicht vorgestellt)

select *
from R_1, R_2
where $R_1.A = R_2.B$;

select *
from R_1 **join** R_2 **on** $R_1.A = R_2.B$;

Äußere Joins: left outer join

select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr, s.MatrNr, s.Name

from Professoren p **left outer join**

(prüfen f **left outer join** Studenten s **on** f.MatrNr=s.MatrNr)

on p.PersNr=f.PersNr;

p.PersNr	p.Name	f.PersNr	f.Note	f.MatrNr	s.MatrNr	s.Name
2126	Russel	2126	1	28106	28106	Carnap
2125	Sokrates	2125	2	25403	25403	Jonas
2137	Kant	2137	2	27550	27550	Schopenhauer
2136	Curie	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Äußere Joins: right outer join

select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr, s.MatrNr, s.Name

from Professoren p **right outer join**

(prüfen f **right outer join** Studenten s **on** f.MatrNr=s.MatrNr)

on p.PersNr=f.PersNr;

p.PersNr	p.Name	f.PersNr	f.Note	f.MatrNr	s.MatrNr	s.Name
2126	Russel	2126	1	28106	28106	Carnap
2125	Sokrates	2125	2	25403	25403	Jonas
2137	Kant	2137	2	27550	27550	Schopenhauer
-	-	-	-	-	26120	Fichte
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Äußere Joins: full outer join

```
select p.PersNr, p.Name, f.PersNr, f.Note, f.MatrNr, s.MatrNr, s.Name
from Professoren p full outer join
    (prüfen f full outer join Studenten s on f.MatrNr=s.MatrNr)
    on p.PersNr=f.PersNr;
```

Datenbanken: SQL

p.PersNr	p.Name	f.PersNr	f.Note	f.MatrNr	s.MatrNr	s.Name
2126	Russel	2126	1	28106	28106	Carnap
2125	Sokrates	2125	2	25403	25403	Jonas
2137	Kant	2137	2	27550	27550	Schopenhauer
-	-	-	-	-	26120	Fichte
⋮	⋮	⋮	⋮	⋮	⋮	⋮
2136	Curie	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Rekursion

Rekursion

- Direkte Vorgänger
- Indirekte Vorgänger
- Vorgänger der Tiefe *n*
- Transitive Hülle
- Rekursive Sichten

Rekursion: direkte Vorgänger

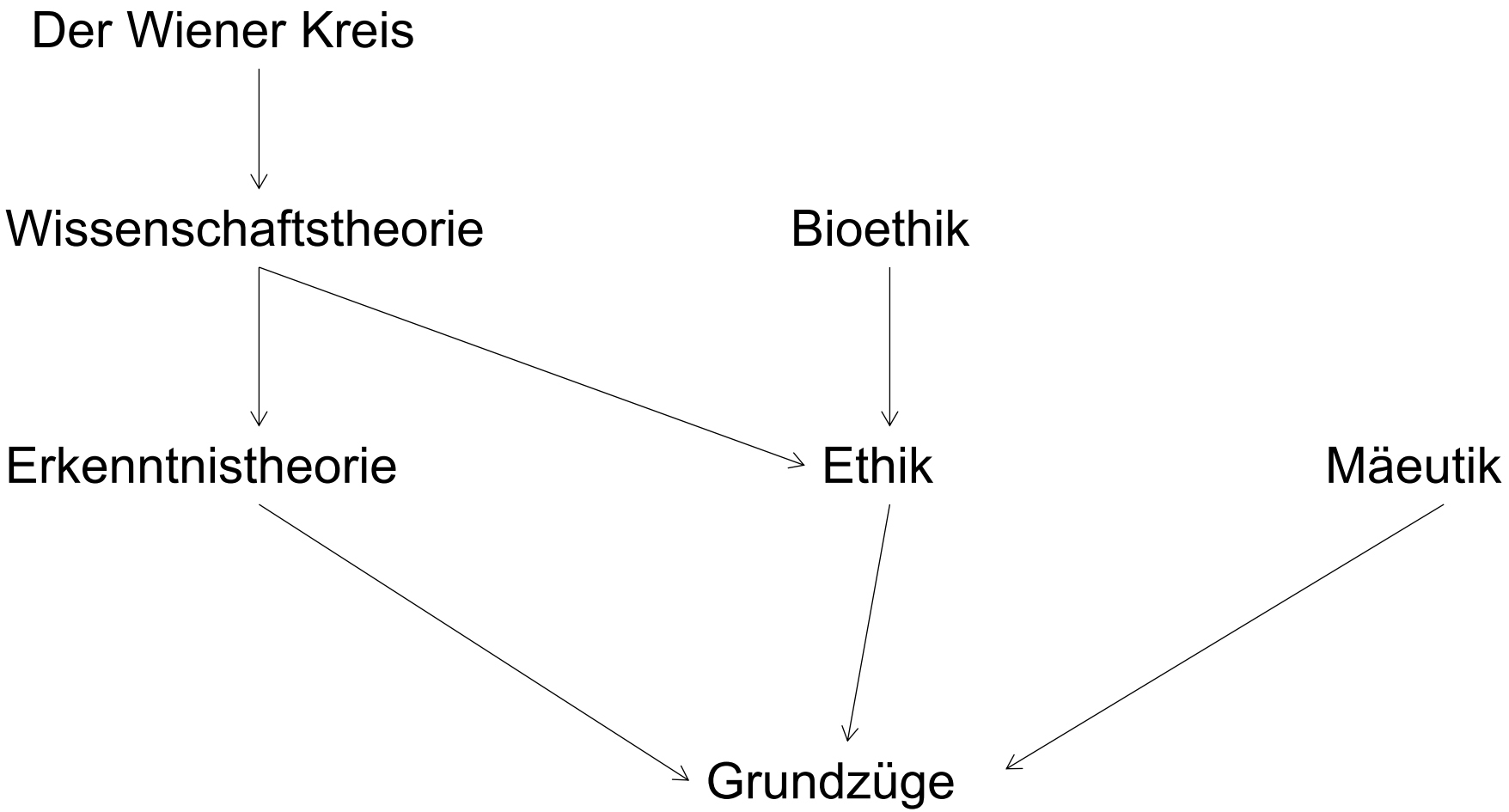
```
select r.Vorgänger
from voraussetzen r, Vorlesungen v
where r.Nachfolger = v.VorlNr and
      v.Titel= 'Der Wiener Kreis'
```

Datenbanken: SQL

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

Vorlesungen			
VorlNr	**Titel**	**SWS**	**gelesenVon**
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

Datenbanken: SQL



voraussetzen	
Vor-gänger	**Nach-folger**
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Rekursion

Vor-Vorgänger-Vorlesungen des „Wiener Kreis“

```
select v1.Vorgänger
from voraussetzen v1, voraussetzen v2, Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
      v2.Nachfolger= v.VorlNr and
      v.Titel='Der Wiener Kreis'
```

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Rekursion

select v1.Vorgänger

from voraussetzen v1, voraussetzen v2, Vorlesungen v

where v1.Nachfolger= v2.Vorgänger **and**

v2.Nachfolger= v.VorlNr **and**

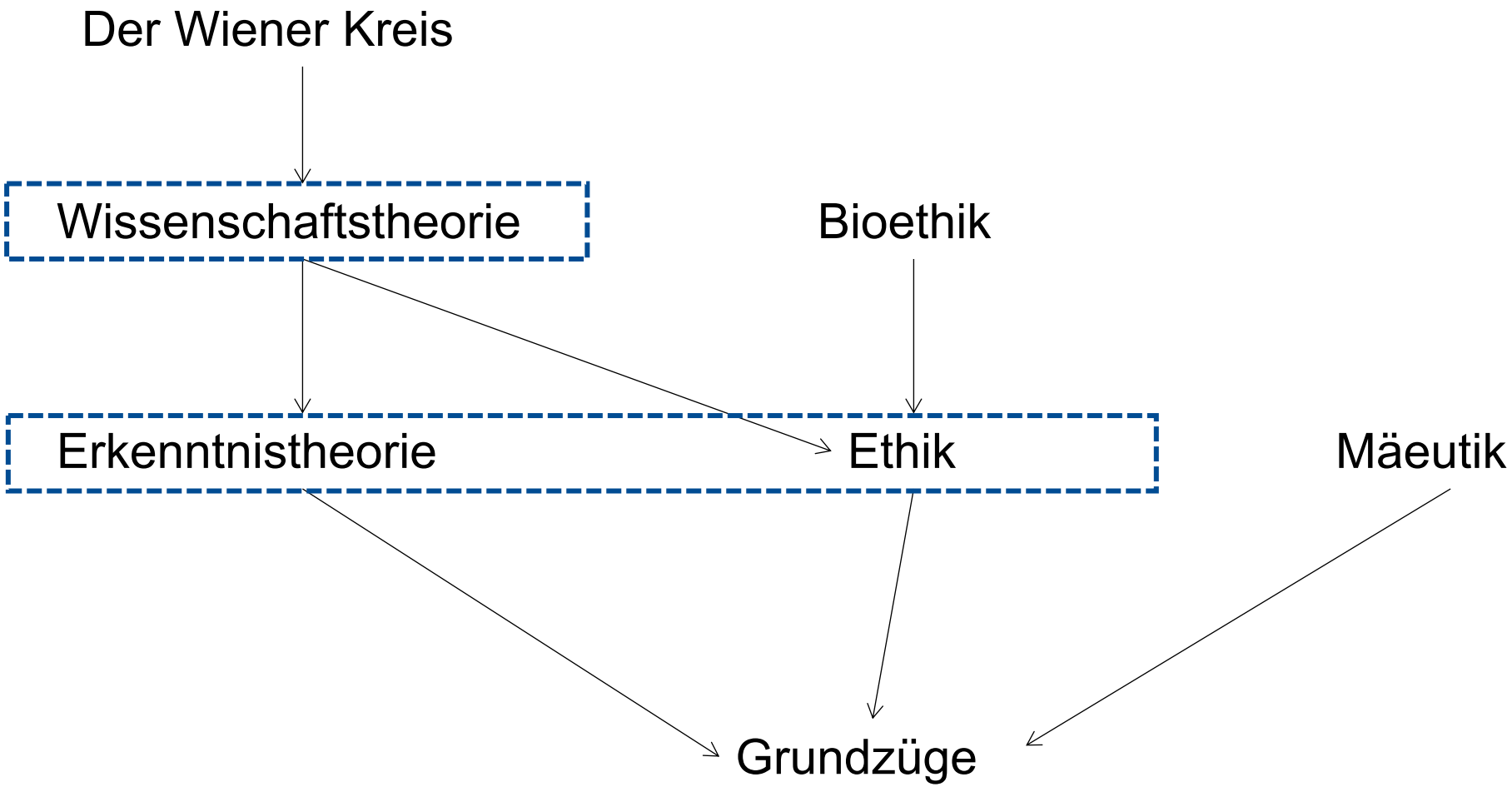
v.Titel='Der Wiener Kreis‘

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

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

Vorlesungen			
VorlNr	**Titel**	**SWS**	**gelesen Von**
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
5052	Logik	4	2125
5216	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

Datenbanken: SQL



voraussetzen	
Vor-gänger	**Nach-folger**
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Vorgänger des „Wiener Kreises“ der Tiefe n

```
select v1.Vorgänger
from voraussetzen v1
      .
      .
      .
       voraussetzen vn:minus_1
       voraussetzen vn,
      Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
      .
      .
      .
      vn_minus_1.Nachfolger= vn.vorgänger and
          vn.Nachfolger = v.VorlNr and
      v.Titel=`Der Wiener Kreis´
```

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Rekursion

v1

voraussetzen	
Vor-gänger	Nach-folger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

v2

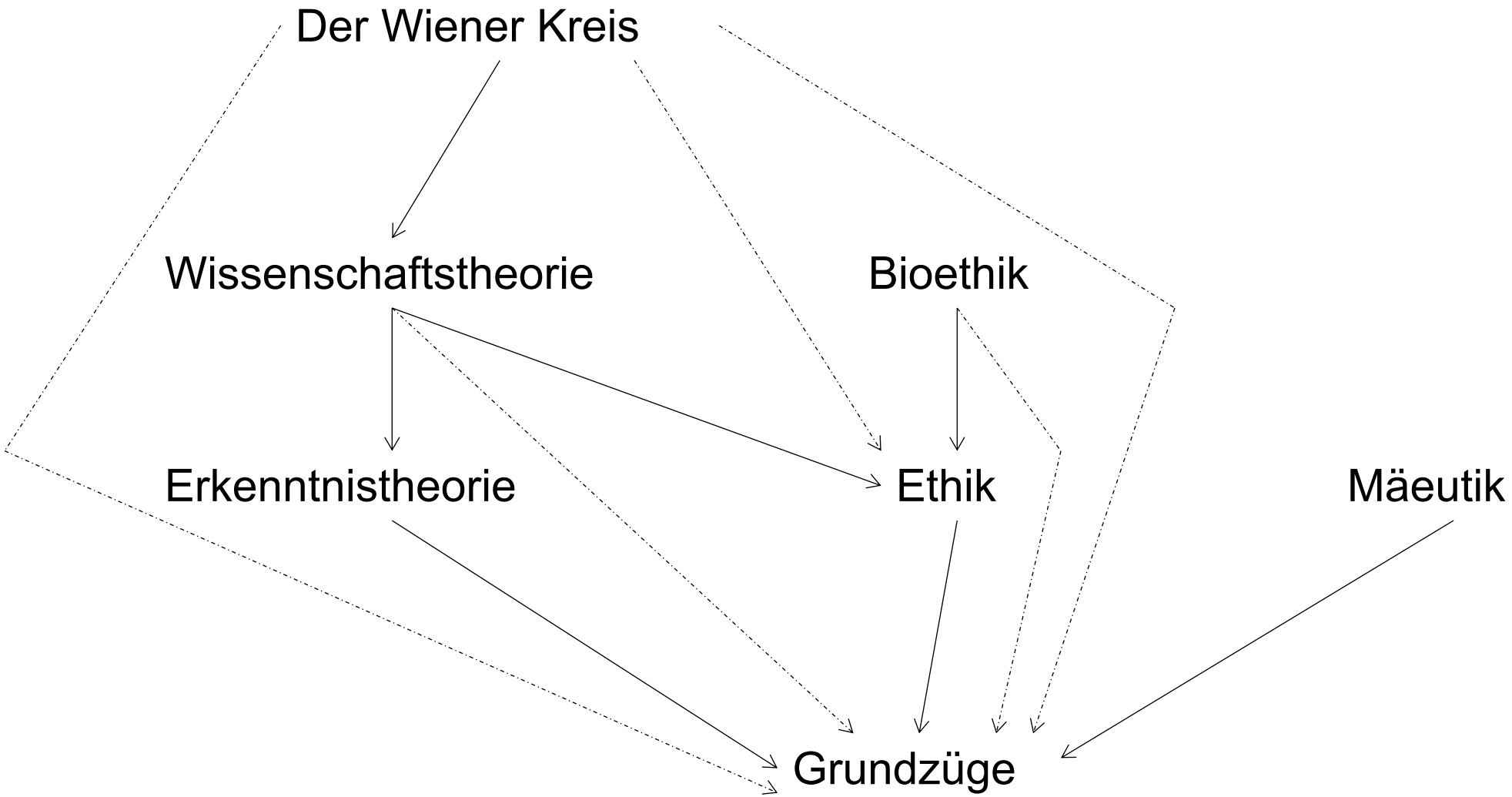
voraussetzen	
Vor-gänger	Nach-folger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

v3

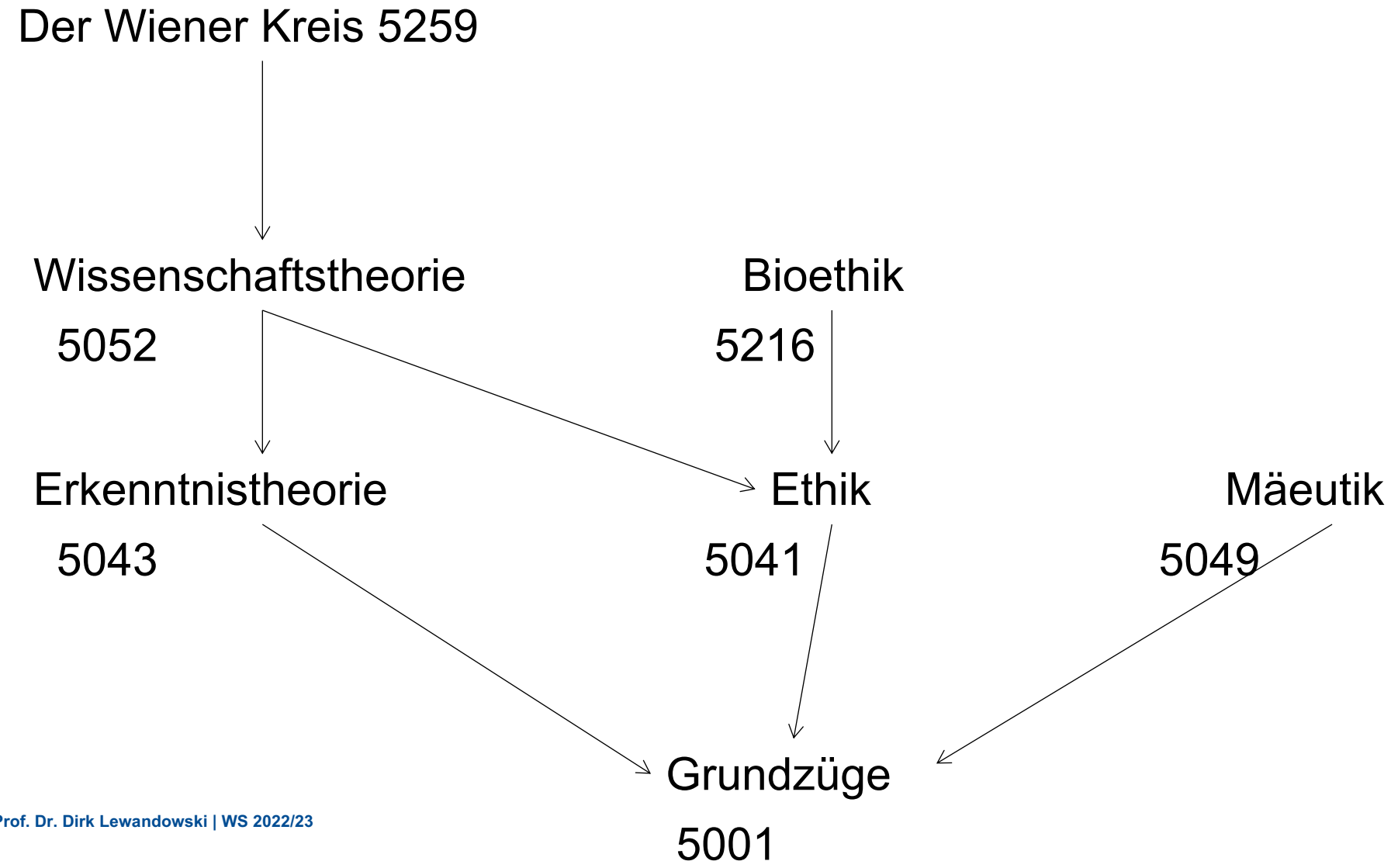
voraussetzen	
Vor-gänger	Nach-folger
5001	5041
5001	5043
5001	5049
5041	5216
5043	5052
5041	5052
5052	5259

Vorlesungen			
VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
5052	Logik	4	2125
5216	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

Datenbanken: SQL



Datenbanken: SQL



Rekursion in Prolog/Datalog

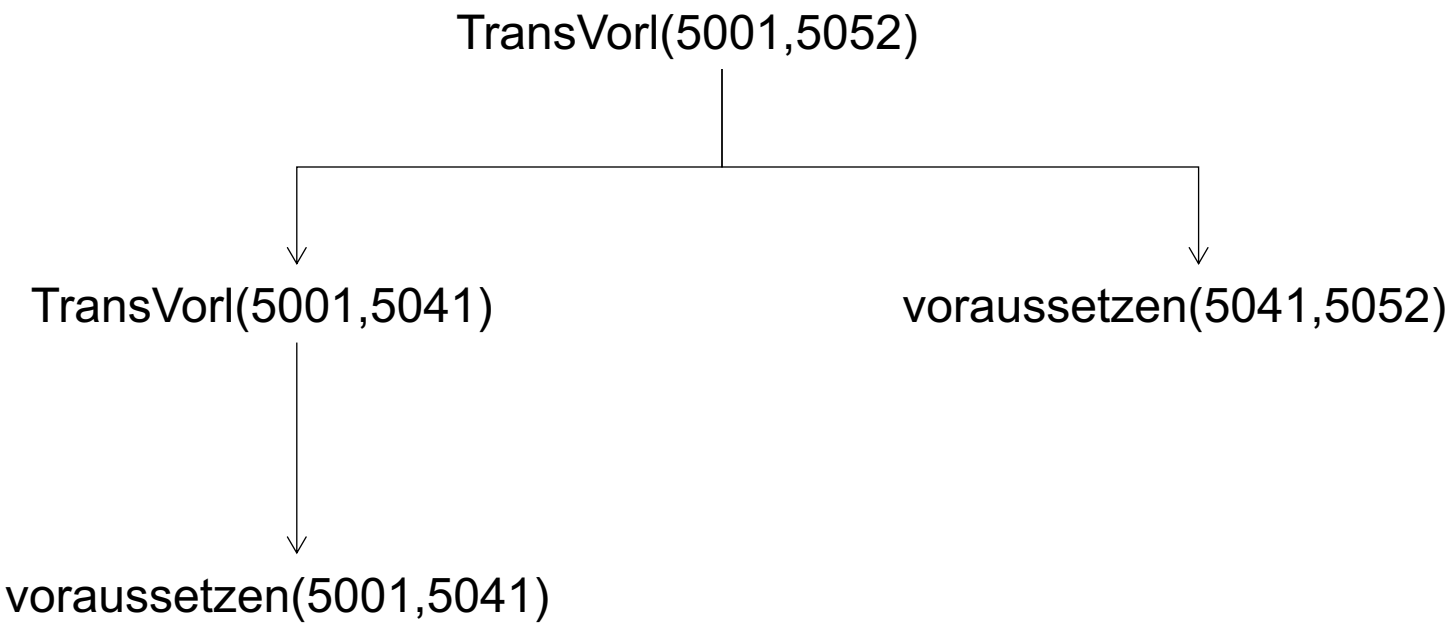
```
                        5
1 --> 2 --> 3 --> 4 ´
              |       ` 6-->7
              |_______|
```

- kante(1,2).
- kante(2,3).
- kante(3,4).
- kante(4,5).
- kante(4,6).
- kante(6,7).
- kante(3,6).

- pfad(V,N) :- kante(V,N).
- pfad(V,N) :- kante(V,Z),pfad(Z,N).

Transitive Hülle der Relation voraussetzen

- TransVorl(V,N) :- voraussetzen(V,N).
- TransVorl(V,N) :- TransVorl(V,Z), voraussetzen(Z,N).



Rekursion in SQL99: rekursive Sichten

```
with TransVorl (Vorg, Nachf)
as (select r.Vorgänger, r.Nachfolger from voraussetzen r
                union all
        select t.Vorg, v.Nachfolger
        from TransVorl t, voraussetzen v
        where t.Nachf= v.Vorgänger)

select v.Titel from Vorlesungen v where v.VorlNr in
(select t.Vorg from TransVorl t where t.Nachf in
(select w.VorlNr from Vorlesungen w
where w.Titel= `Der Wiener Kreis´) )
```

(vergl.: TransVorl(X,Y) :- voraussetzen(X,Y);
TransVorl(X,Y) :- TransVorl(X,Z), voraussetzen(Z,Y);

Zusammengefasst

- Zuerst wird eine temporäre Sicht *TransVorl* mit der **with**-Klausel angelegt
- Diese Sicht *TransVorl* ist rekursiv definiert, da sie selbst in der Definition vorkommt
- Aus dieser Sicht werden dann die gewünschten Tupel extrahiert

Datenbanken: SQL

Vorlesungen			
VorlNr	Titel	SWS	gelesenVon
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
5052	Logik	4	2125
5216	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

Grundzüge
Ethik
Erkenntnistheorie
Wissenschaftstheorie

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Veränderungen am Datenbestand

Veränderungen am Datenbestand

- Einfügen von Tupeln
- Löschen von Tupeln
- Zweistufiges Vorgehen bei Änderungen

Veränderung am Datenbestand: Einfügen von Tupeln

```
insert into hören
    select s.MatrNr, v.VorlNr
    from Studenten s, Vorlesungen v
    where v.Titel= 'Logik‘;

insert into Studenten (MatrNr, Name)
    values (28121, 'Archimedes');
```

Studenten		
MatrNr	Name	Semester
⋮	⋮	⋮
29120	Theophrastos	2
29555	Feuerbach	2
28121	Archimedes	-

Veränderungen am Datenbestand: Löschen von Tupeln

delete Studenten s
    where s.Semester > 13;

Verändern von Tupeln

update Studenten s
    set s.Semester = s.Semester + 1;

Zweistufiges Vorgehen bei Änderungen

1. Die Kandidaten für die Änderung werden ermittelt und "markiert“
2. Die Änderung wird an den in Schritt 1. ermittelten Kandidaten durchgeführt

Anderenfalls könnte die Änderungsoperation von der Reihenfolge der Tupel abhängen, wie folgendes Beispiel zeigt:

```
delete from voraussetzen r
    where r.Vorgänger in (select r1.Nachfolger
                          from voraussetzen r1);
```

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

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

```
delete from voraussetzen r
where r.Vorgänger in
(select v.Nachfolger
        from voraussetzen v);
```

Ohne einen Markierungsschritt hängt das Ergebnis dieser Anfrage von der Reihenfolge der Tupel in der Relation ab. Eine Abarbeitung in der Reihenfolge der Beispielausprägung würde das letzte Tupel (5052, 5229) fälschlicherweise erhalten, da vorher bereits alle Tupel mit 5052 als Nachfolger entfernt wurden.

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Sichten

Sichten

- … für den Datenschutz
- … zur Vereinfachung von Anfragen
- … zur Modellierung von Generalisierung
- … zur Gewährleistung von Datenunabhängigkeit
- Änderbarkeit von Sichten

Sichten für den Datenschutz

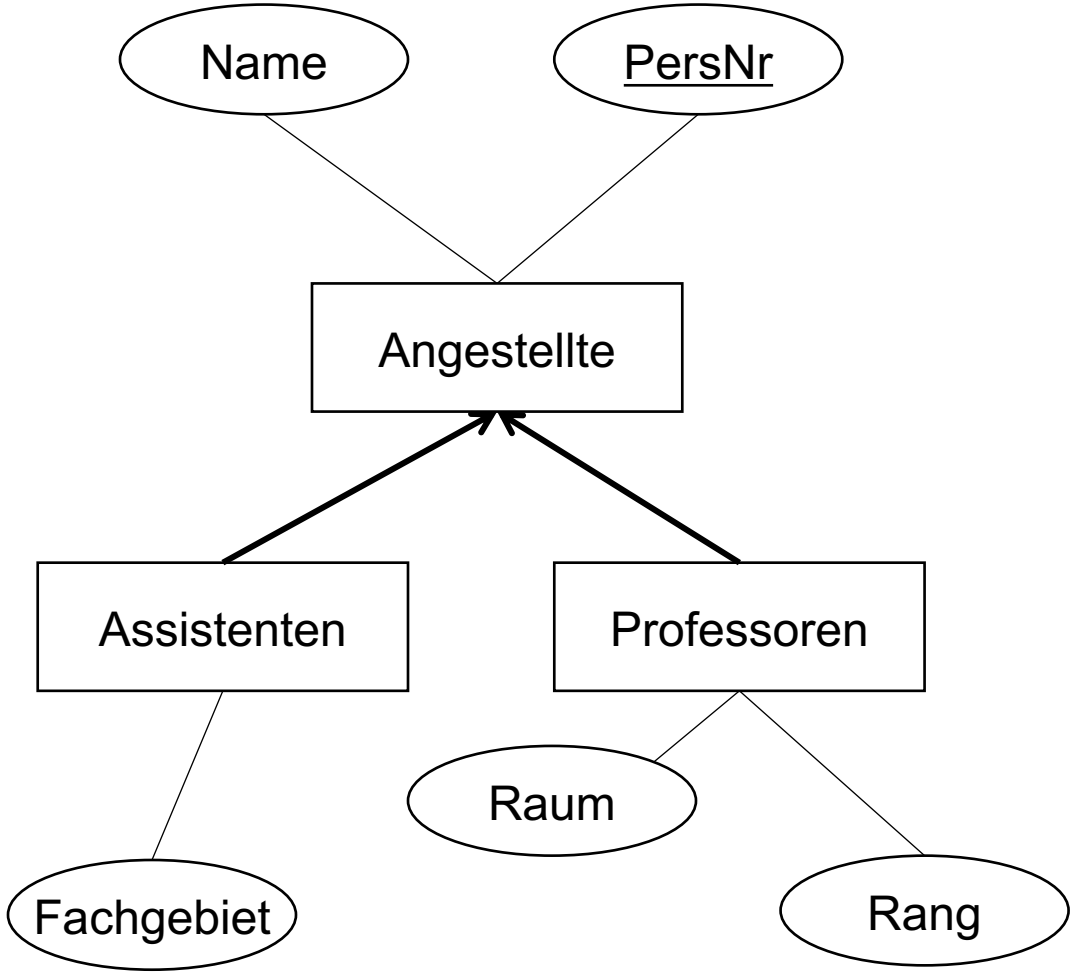
```
create view prüfenSicht as
    select p.MatrNr, p.VorlNr, p.PersNr
    from prüfen p
```

Sichten für die Vereinfachung von Anfragen

```
create view StudProf (Sname, Semester, Titel, Pname) as
select s.Name, s.Semester, v.Titel, p.Name
from Studenten s, hören h, Vorlesungen v, Professoren p
where s.Matr.Nr=h.MatrNr and h.VorlNr=v.VorlNr and
      v.gelesenVon=p.PersNr

select distinct s.Semester
from StudProf s
where s.Pname='Sokrates‘;
```

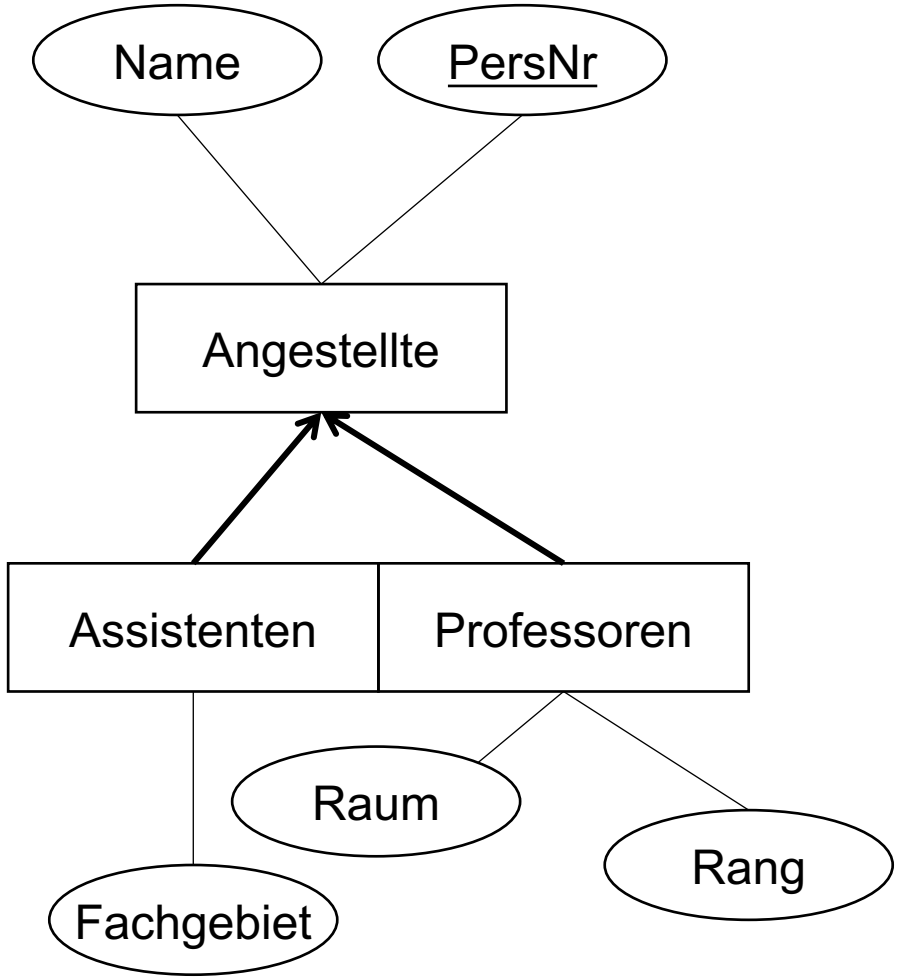
Sichten zur Modellierung von Generalisierung



UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Sichten zur Modellierung von Generalisierung

```
create table Angestellte
    (PersNr integer not null,
    Name varchar (30) not null);
create table ProfDaten
    (PersNr integer not null,
    Rang character(2),
    Raum integer);
create table AssiDaten
(PersNr integer not null,
Fachgebiet varchar(30),
Boss integer);
```



```
create view Professoren as
        select *
        from Angestellte a, ProfDaten d
        where a.PersNr=d.PersNr;
create view Assistenten as
        select *
        from Angestellte a, AssiDaten d
        where a.PersNr=d.PersNr;
```

→Untertypen als Sicht

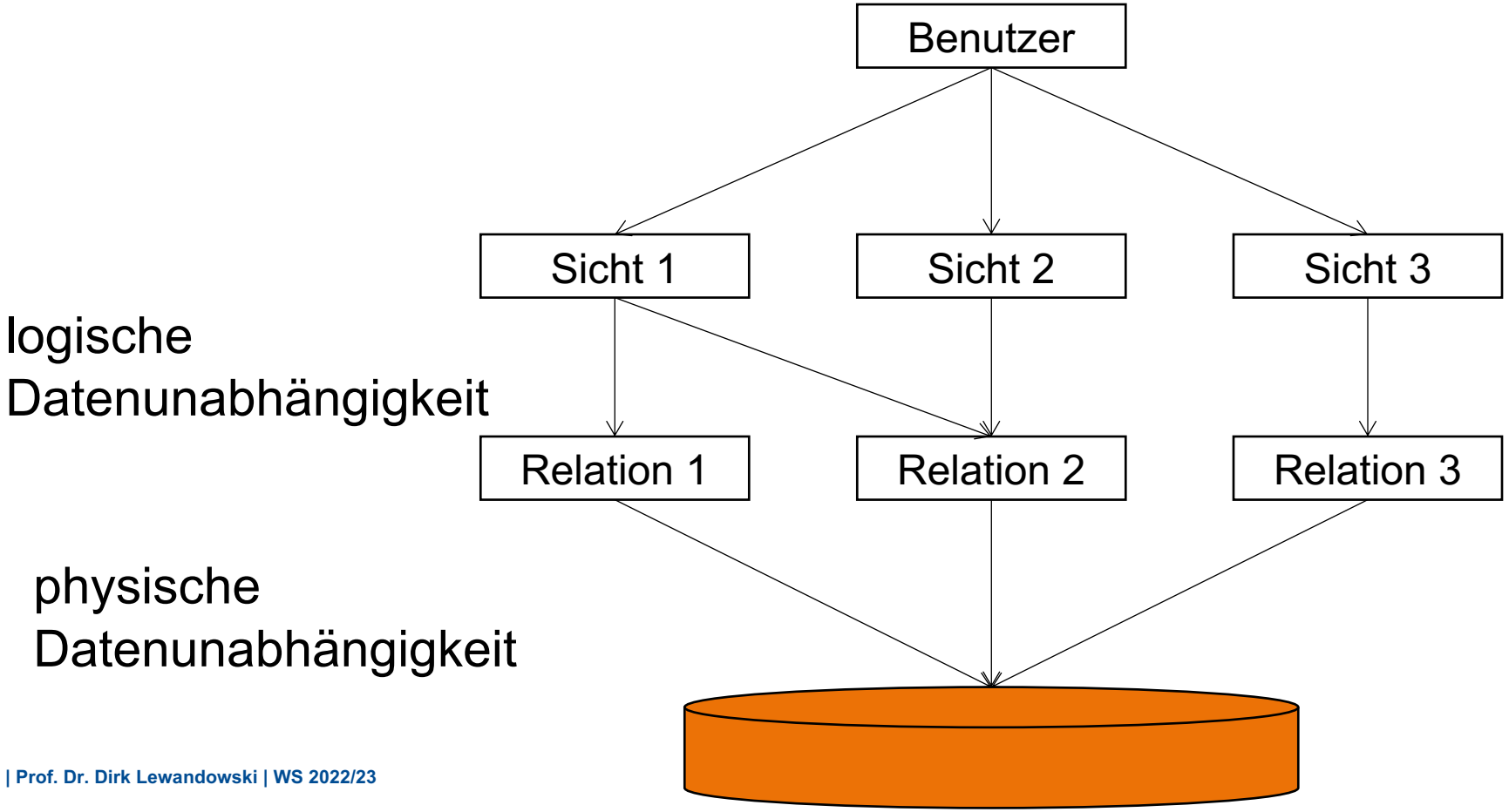
```
create table Professoren
        (PersNr integer not null,
        Name varchar (30) not null,
        Rang character (2),
        Raum integer);
create table Assistenten
        (PersNr integer not null,
        Name varchar (30) not null,
        Fachgebiet varchar (30),
        Boss integer);
create table AndereAngestellte
        (PersNr integer not null,
        Name varchar (30) not null);
```

```
create view Angestellte as
        (select p.PersNr, p.Name
        from Professoren p)
          union
        (select a.PersNr, a.Name
        from Assistenten a)
          union
        (select o.*
        from AndereAngestellte o);
```

→ **Obertypen als Sicht**

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Sichten zur Gewährleistung von Datenunabhängigkeit



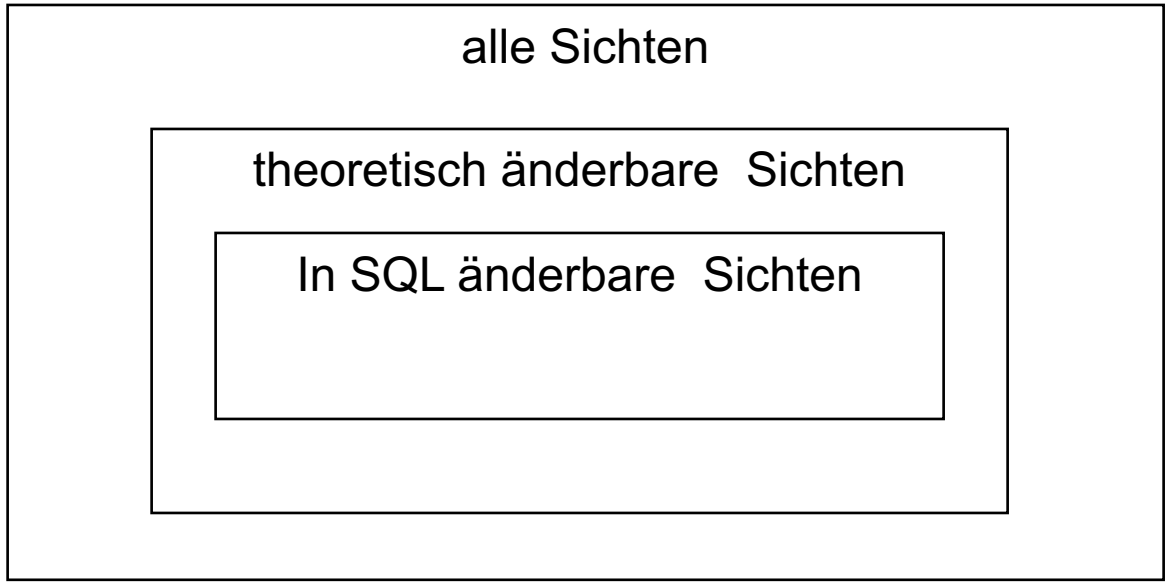
Änderbarkeit von Sichten: Beispiele für nicht änderbare Sichten

```
create view WieHartAlsPrüfer
    (PersNr, Durchschnittsnote)
as
    select p.PersNr, avg(p.Note)
    from prüfen p
    group by p.PersNr;
```

```
create view VorlesungenSicht as
    select v.Titel, v.SWS, pName
    from Vorlesungen v,
    Professoren p
    where
    v.gelesenVon=p.PersNr;

insert into VorlesungenSicht
    values ('Nihilismus', 2,
    'Nobody');
```

Änderbarkeit von Sichten in SQL

Bedingungen

- nur eine Basisrelation
- Schlüssel muss vorhanden sein
- keine Aggregatfunktionen, Gruppierung und Duplikateliminierung