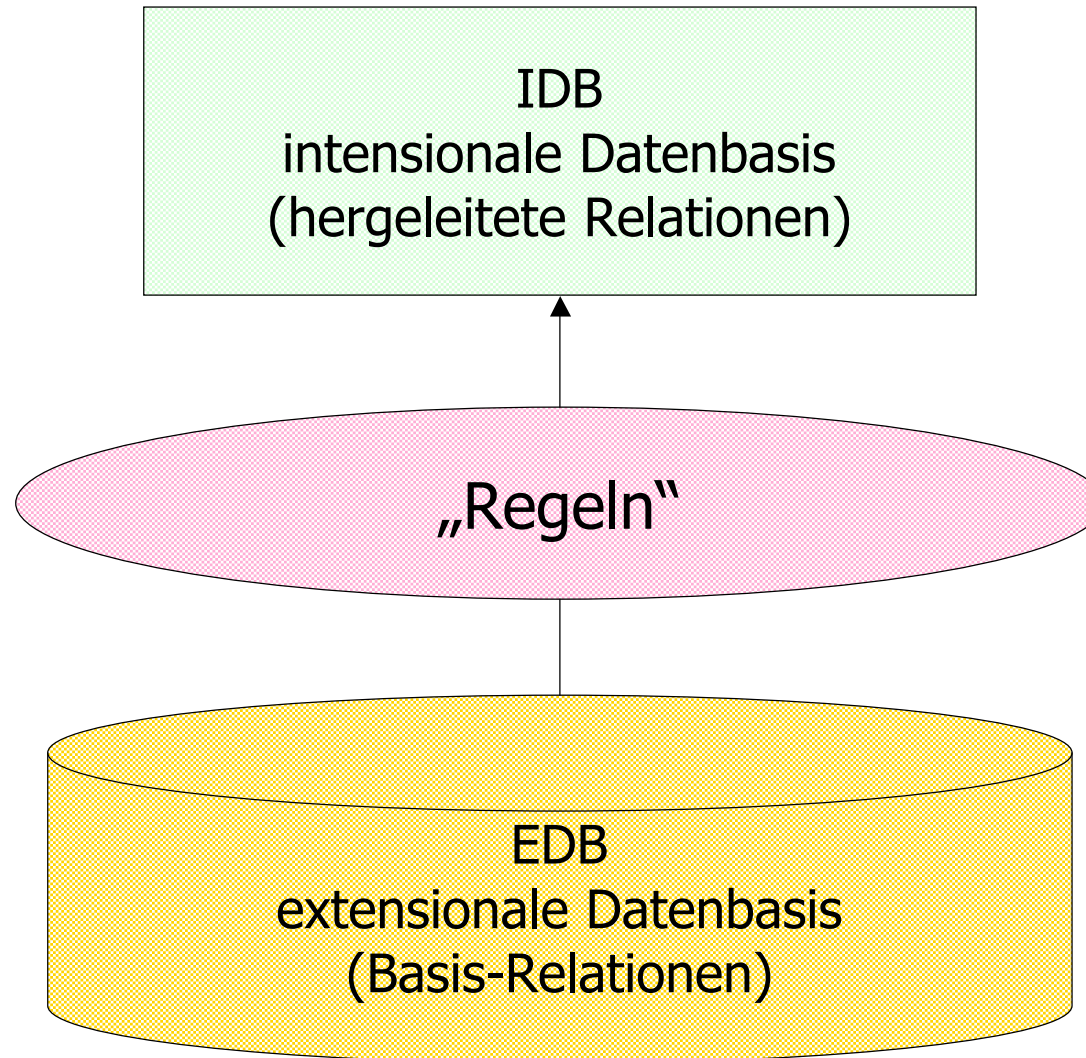


# Deduktive Datenbanken

nach: Alfons Kemper, A. Eickler, Datenbanksysteme

## Grundkonzepte einer deduktiven Datenbank



# Terminologie

- Die *extensionale Datenbasis (EDB)*, die manchmal auch Faktenbasis genannt wird. Die EDB besteht aus einer Menge von Relationen(Ausprägungen) und entspricht einer „ganz normalen“ relationalen Datenbasis.
- Die *Deduktionskomponente*, die aus einer Menge von (Herleitungs-)Regeln besteht. Die Regelsprache heißt *Datalog* – abgeleitet von dem Wort *Data* und dem Namen der Logikprogrammiersprache *Prolog*.
- Die *intensionale Datenbasis (IDB)*, die aus einer Menge von hergeleiteten Relationen(Ausprägungen) besteht. Die IDB wird durch Auswertung des Datalog-Programms aus der EDB generiert.

# Zentrale Idee

Kantendaten

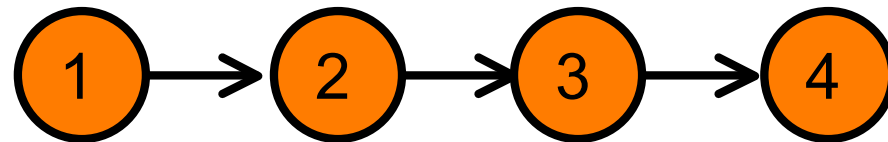
Von	Nach
1	2
2	3
3	4

Fakten

kante( 1, 2 ).

kante( 2, 3 ).

kante( 3, 4 ).



Regeln

pfad( X, Y ) :- kante( X, Y ).

pfad( X, Y ) :- kante( X, Z ),

pfad( Z, Y ).

# Datalog

Regel:

$\text{sokLV}(T,S) :- \text{vorlesungen}(V,T,S,P), \text{professoren}(P, \text{„Sokrates“}, R,Z), >(S,2).$

Äquivalenter Domänenkalkül-Ausdruck:

$$\{[t,s] \mid \exists v,p ([v,t,s,p] \in \text{Vorlesungen} \wedge \\ \exists n,r,z ([p,n,r,z] \in \text{Professoren} \wedge \\ n = \text{„Sokrates“} \wedge s > 2))\}$$

Grundbausteine der Regeln sind *atomare Formeln oder Literale*:

$$q(A_1, \dots, A_m).$$

$q$  ist dabei der Name einer Basisrelation, einer abgeleiteten Relation oder eines eingebauten Prädikats:  $<, =, >, \dots$

Beispiel:  $\text{professoren}(S, \text{„Sokrates“}, R,Z).$

# Eine Datalog-Regel

$$p(X_1, \dots, X_m) :- q_1(A_{11}, \dots, A_{1m_1}), \dots, q_n(A_{n1}, \dots, A_{nm_n}).$$

- Jedes  $q_j(\dots)$  ist eine **atomare Formel**. Die  $q_j$  werden oft als *Subgoals* bezeichnet.
- $X_1, \dots, X_m$  sind **Variablen**, die mindestens einmal auch auf der rechten Seite des Zeichens  $:-$  vorkommen müssen.
- Logisch äquivalente Form obiger Regel:

$$p(\dots) \vee \neg q_1(\dots) \vee \dots \vee \neg q_n(\dots)$$

- = Wir halten uns an folgende Notation:
  - = Variablen beginnen mit großen Buchstaben
  - = Konstanten beginnen mit kleinem Buchstaben
  - = **Prädikate** haben Argumentklammern und beginnen mit einem Kleinbuchstaben. Die zugehörigen **Relationen** – seien es **EDB-** oder **IDB-Relationen** – werden mit gleichem Namen, aber mit einem Großbuchstaben beginnend, bezeichnet (Verwechslungsgefahr mit Variablen besteht nicht).

# Beispiel Datalog-Programm

- Zur Bestimmung von (thematisch) verwandten Vorlesungspaaren

geschwisterVorl( $N_1, N_2$ ) :- voraussetzen( $V, N_1$ ),  
voraussetzen( $V, N_2$ ),  $N_1 < N_2$ .

geschwisterThemen( $T_1, T_2$ ) :- geschwisterVorl( $N_1, N_2$ ),  
vorlesungen( $N_1, T_1, S_1, R_1$ ),  
Vorlesungen( $N_2, T_2, S_2, R_2$ ).

aufbauen( $V, N$ ) :- voraussetzen( $V, N$ )

aufbauen( $V, N$ ) :- aufbauen( $V, M$ ), voraussetzen( $M, N$ ).

verwandt( $N, M$ ) :- aufbauen( $N, M$ ).

verwandt( $N, M$ ) :- aufbauen( $M, N$ ).

verwandt( $N, M$ ) :- aufbauen( $V, N$ ), aufbauen( $V, M$ ).

- Voraussetzen: {[Vorgänger, Nachfolger]}
- Vorlesungen: {VorlNr, Titel, SWS, gelesenVon]}

# Analogie zur EDB/IDB in rel. DBMS

- Basis-Relationen entsprechen den EDB.
- Sichten entsprechen den IDB:
  - „Aufbauen“ als Regeln in einem deduktiven DBMS:

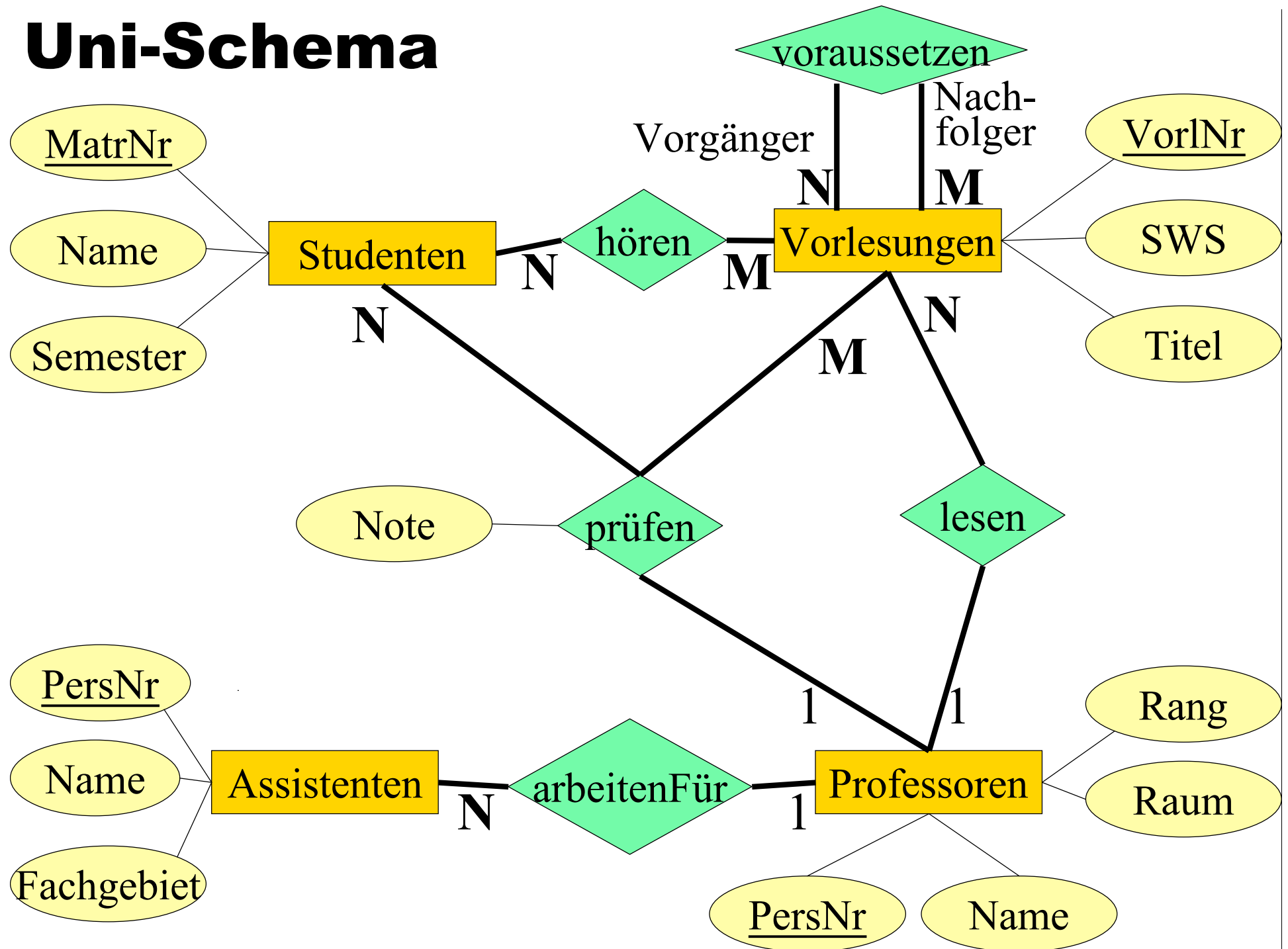
aufbauen( $V, N$ ) :- voraussetzen( $V, N$ )  
aufbauen( $V, N$ ) :- aufbauen( $V, M$ ), voraussetzen( $M, N$ ).

- „Aufbauen“ als Sichtdefinition in DB2:

```
create view aufbauen(V,N) as
  (select Vorgaenger, Nachfolger
   from voraussetzen
   union all
   select a.V, v.Nachfolger
   from aufbauen a, voraussetzen v
   where a.N = v.Vorgaenger)

select * from aufbauen
```

# Uni-Schema



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

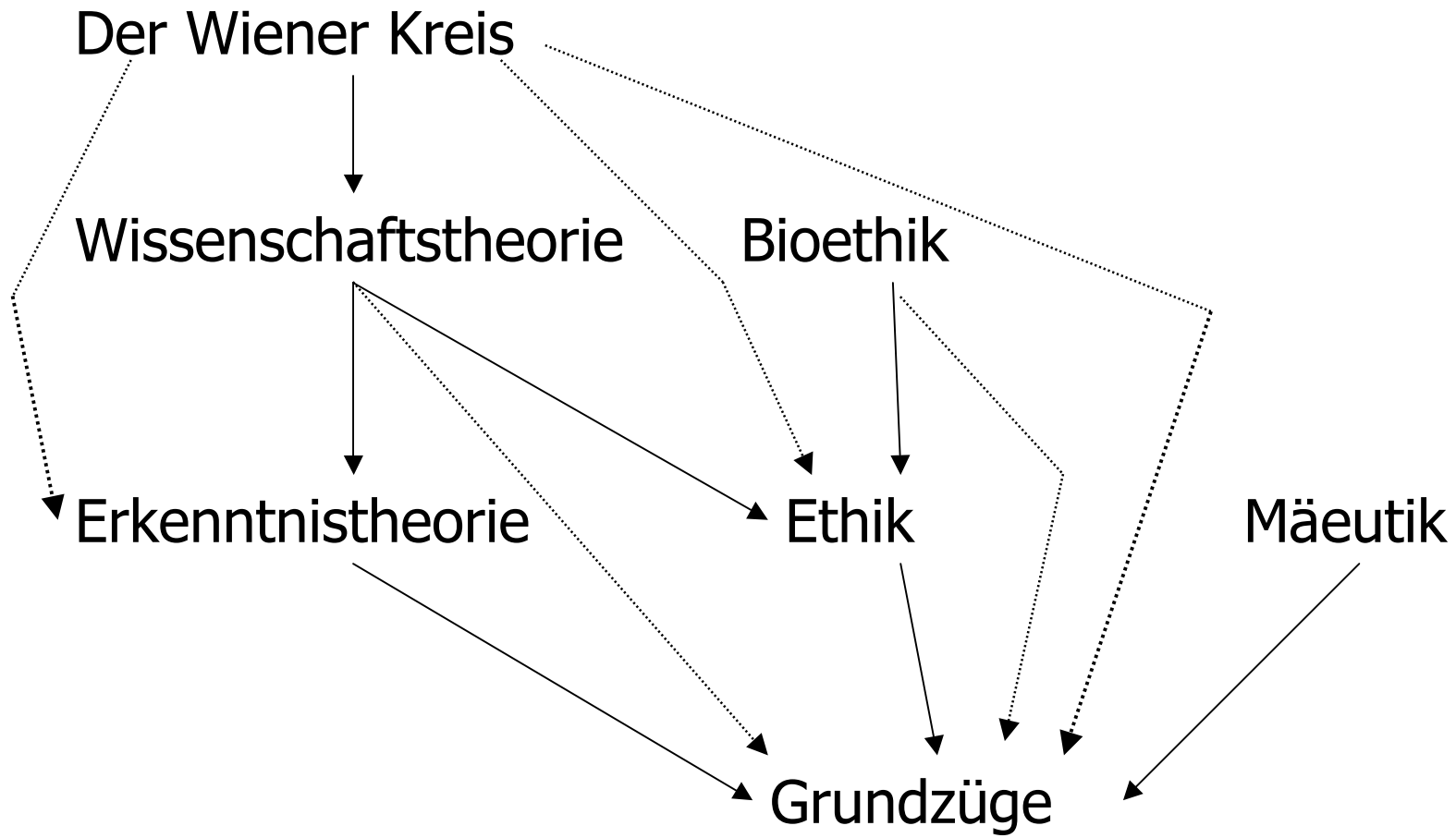
Vorlesungen			
VorlNr	Titel	SWS	gelesen Von
5001	Grundzüge	4	2137
5041	Ethik	4	2125
5043	Erkenntnistheorie	3	2126
5049	Mäeutik	2	2125
4052	Logik	4	2125
5052	Wissenschaftstheorie	3	2126
5216	Bioethik	2	2126
5259	Der Wiener Kreis	2	2133
5022	Glaube und Wissen	2	2134
4630	Die 3 Kritiken	4	2137

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

hören	
MatrNr	VorlNr
26120	5001
27550	5001
27550	4052
28106	5041
28106	5052
28106	5216
28106	5259
29120	5001
29120	5041
29120	5049
29555	5022
25403	5022

Assistenten			
PerslNr	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	VorlNr	PersNr	Note
28106	5001	2126	1
25403	5041	2125	2
27550	4630	2137	2



# Rekursion in DB2/SQL99: gleiche Anfrage

**with** TransVorl (Vorg, Nachf)

**as** (**select** Vorgänger, Nachfolger **from** voraussetzen  
**union all**

**select** t.Vorg, v.Nachfolger

**from** TransVorl t, voraussetzen v

**where** t.Nachf= v.Vorgänger)

**select** Titel **from** Vorlesungen **where** VorlNr **in**

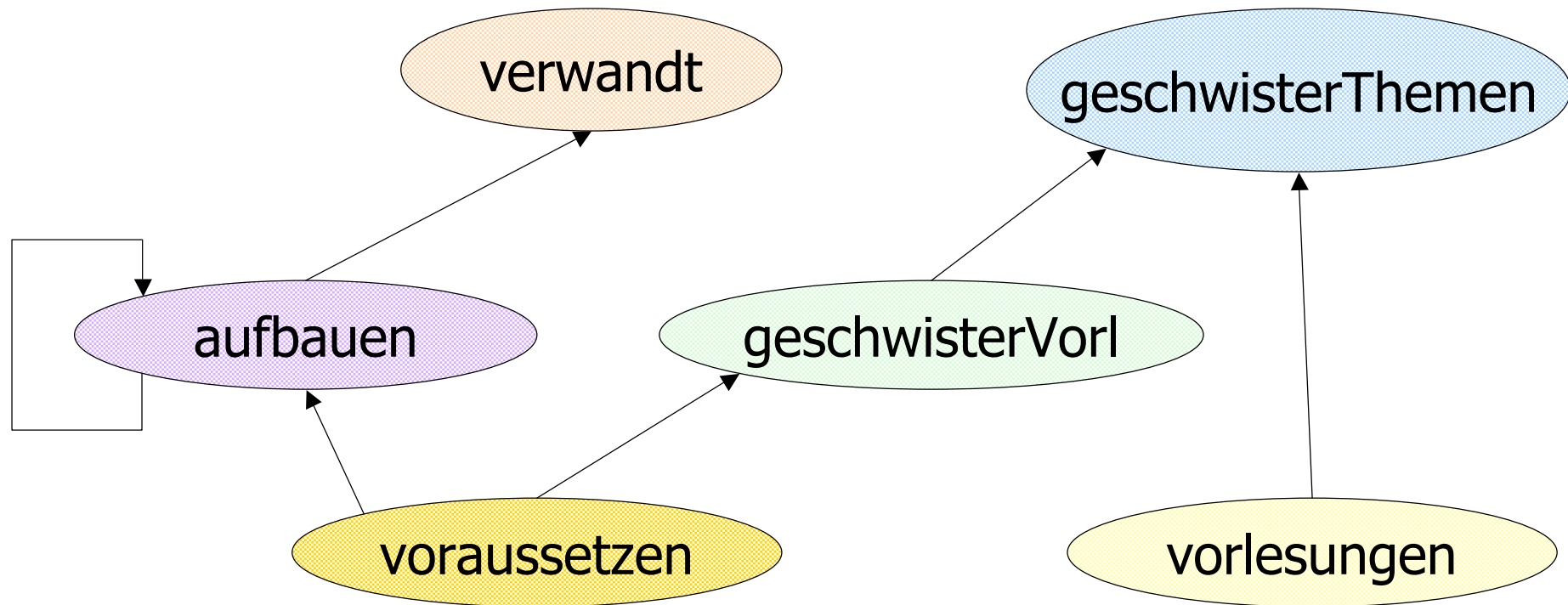
(**select** Vorg **from** TransVorl **where** Nachf **in**

(**select** VorlNr **from** Vorlesungen

**where** Titel= `Der Wiener Kreis` ) )

# Eigenschaften von Datalog-Programmen

Abhängigkeitsgraph (  $\longrightarrow$  = „wird verwendet von“)



- Ein Datalog-Programm ist rekursiv, wenn der Abhängigkeitsgraph einen (oder mehrere) Zyklen hat
- Unser Beispielprogramm ist rekursiv wegen  $\text{aufbauen} \longrightarrow \text{aufbauen}$

# Sicherheit von Datalog-Regeln

- Es gibt unsichere Regeln, wie z.B.

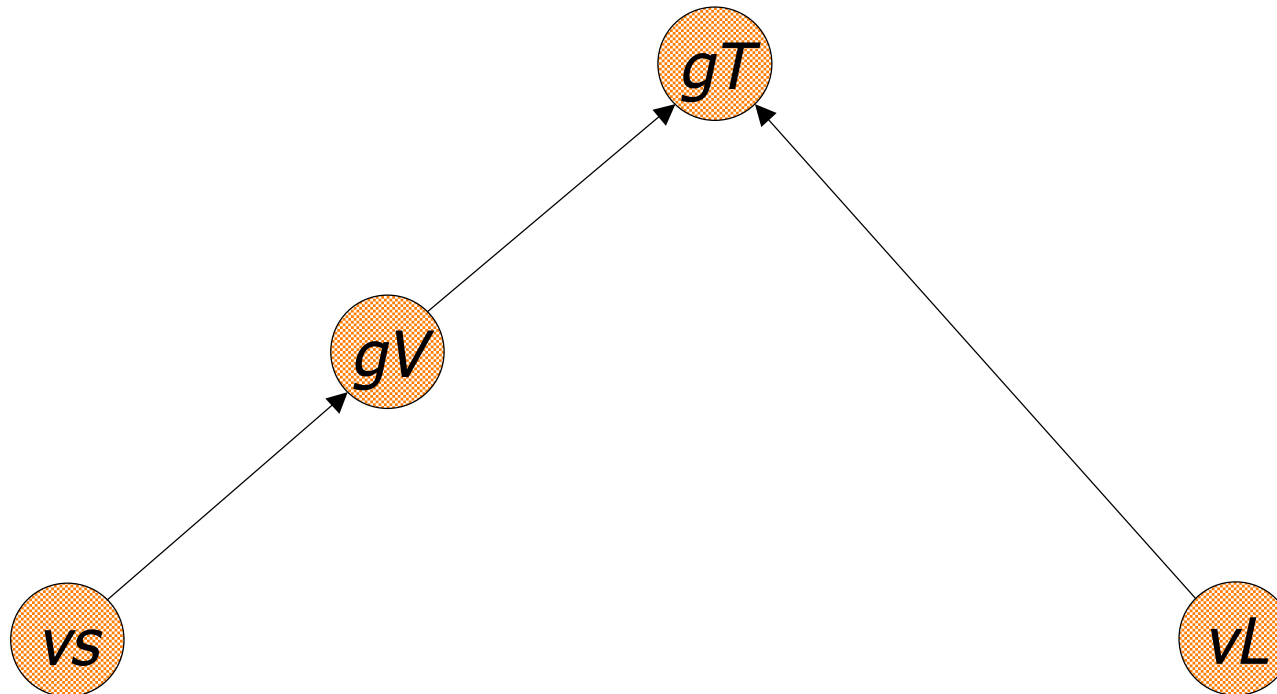
$\text{ungleich}(X, Y) \text{ :- } X \neq Y.$

Diese definieren unendliche Relationen.

- Eine Datalog-Regel ist sicher, wenn alle Variablen im Kopf beschränkt (range restricted) sind. Dies ist für eine Variable  $X$  dann der Fall, wenn:
  - ☀ die Variable im Rumpf der Regel in mindestens einem normalen Prädikat – also nicht nur in eingebauten Vergleichsprädikaten – vorkommt oder
  - ☀ ein Prädikat der Form  $X = c$  mit einer Konstante  $c$  im Rumpf der Regel existiert oder
  - ☀ ein Prädikat der Form  $X = Y$  im Rumpf vorkommt, und man schon nachgewiesen hat, dass  $Y$  eingeschränkt ist.

# Ein zyklensfreier Abhängigkeitsgraph

- $gV(N1, N2) :- vs(V, N1), vs(V, N2), N1 < N2.$
- $gT(T1, T2) :- gV(N1, N2), vL(N1, T1, S1, R1), vL(N2, T2, S2, R2)$



- Eine mögliche topologische Sortierung ist:  $vs, gV, vL, gT$

# Auswertung nicht-rekursiver Datalog-Programme

1. Für jede Regel mit dem Kopf  $p(\dots)$ , also

$$p(\dots) \text{ :- } q_1(\dots), \dots, q_n(\dots).$$

bilde eine Relation, in der alle im Körper der Regel vorkommenden Variablen als Attribute vorkommen. Diese Relation wird im wesentlichen durch einen natürlichen Verbund der Relationen  $Q_1, \dots, Q_n$ , die den Relationen der Prädikate  $q_1, \dots, q_n$  entsprechen, gebildet. Man beachte, dass diese Relationen  $Q_1, \dots, Q_n$  wegen der Einhaltung der topologischen Sortierung bereits ausgewertet (materialisiert) sind.

2. Da das Prädikat  $p$  durch mehrere Regeln definiert sein kann, werden die Relationen aus Schritt 1. vereinigt. Hierzu muss man vorher auf die im Kopf der Regeln vorkommenden Attribute projizieren. Wir nehmen an, dass alle Köpfe der Regeln für  $p$  dieselben Attributnamen an derselben Stelle verwenden – durch Umformung der Regeln kann man dies immer erreichen.

# Auswertung von Geschwister

## Vorlesungen und Geschwister Themen

- = Die Relation zu Prädikat  $gV$  ergibt sich nach Schritt 1 aus folg. Relationenalgebraausdruck:

$$\sigma_{N1 < N2} (Vs1(V, N1) \times Vs2(V, N2))$$

$$Vs1(V, N1) := \rho_{V \leftarrow \$1} (\rho_{N1 \leftarrow \$2} (\rho_{Vs1}(Voraussetzen)))$$

- = Die dadurch definierte Relation enthält Tupel  $[v, n1, n2]$  mit:
  - = Das Tupel  $[v, n1]$  ist in der Relation *Voraussetzen* enthalten,
  - = das Tupel  $[v, n2]$  ist in der Relation *Voraussetzen* enthalten und
  - =  $n1 < n2$ .
- = Gemäß Schritt 2. ergibt sich:

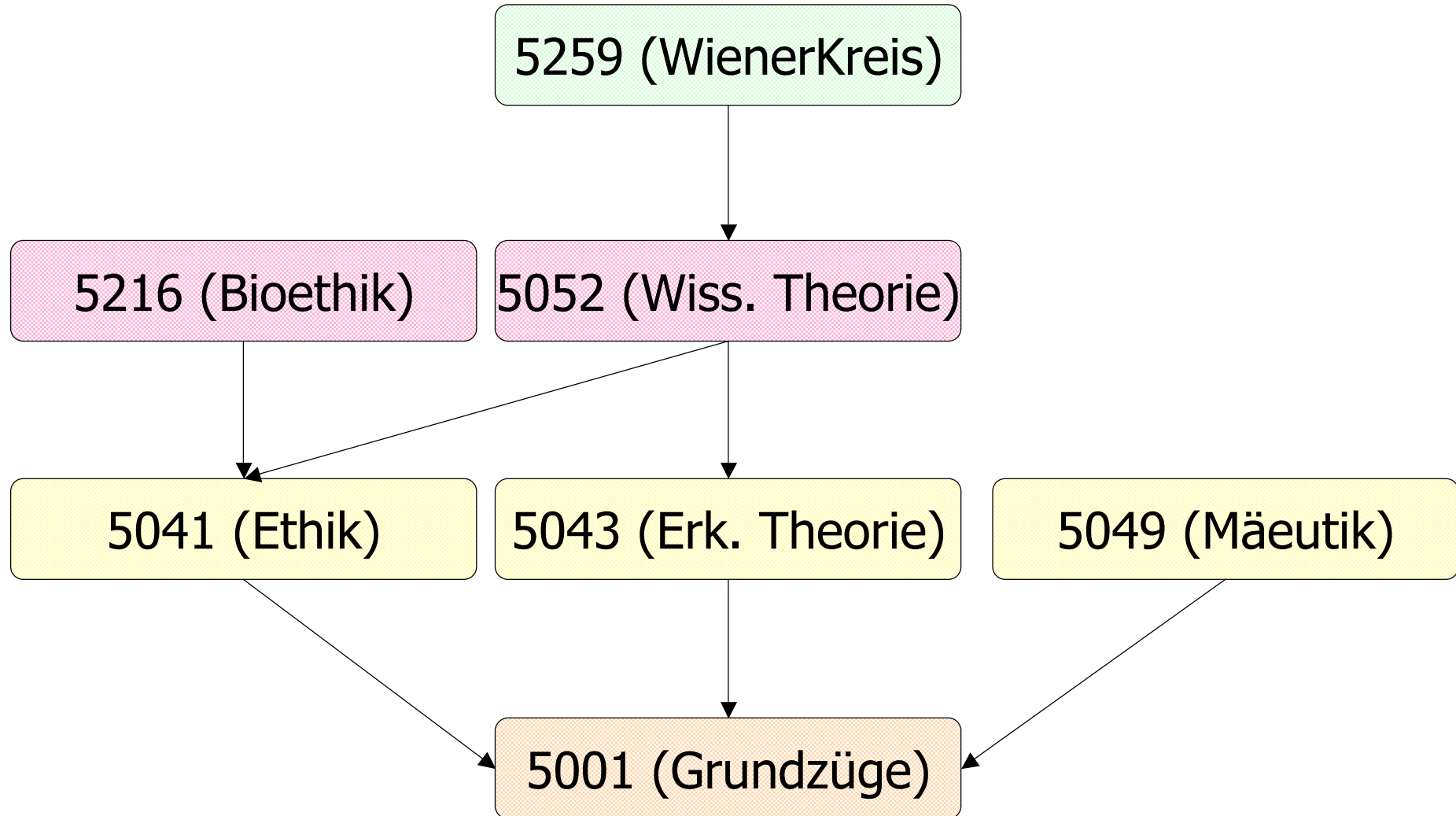
$$GV(N1, N2) := \Pi_{N1, N2} (\sigma_{N1 < N2} (Vs1(V, N1) \times Vs2(V, N2)))$$

- = Analog ergibt sich für die Herleitung von  $GT$ :

$$GT(T1, T2) := \Pi_{T1, T2} (GV(N1, N2) \times VL1(N1, T1, S1, R1) \times VL2(N2, T2, S2, R2))$$

# Veranschaulichung der EDB-Relation

## *Voraussetzen*



# Ausprägung der Relationen *GeschwisterVorl* und *GeschwisterThemen*

GeschwisterVorl	
<i>N1</i>	<i>N2</i>
5041	5043
5043	5049
5041	5049
5052	5216

GeschwisterThemen	
<i>T1</i>	<i>T2</i>
Ethik	Erkenntnistheorie
Erkenntnistheorie	Mäeutik
Ethik	Mäeutik
Wissenschaftstheorie	Bioethik

# Auswertungs-Algorithmus (abstrakt)

Wir betrachten folgende abstrakte Regel :

$$p(X_1, \dots, X_m) : - q_1(A_{11}, \dots, A_{1m_1}), \dots, q_n(A_{n1}, \dots, A_{nm_n}).$$

Die Relationen  $Q_i$  für die Prädikate  $q_i$  seien bereits hergeleitet :

$$Q_i : \{ \$1, \dots, \$m_i \}$$

Für jedes Subgoal  $q_i(A_{i1}, \dots, A_{im_i})$  bilde folgenden Ausdruck  $E_i$  :

$$E_i := \Pi_{V_i}(\sigma_{F_i}(Q_i))$$

Dabei sind die  $V_i$  die in  $q_i(\dots)$  vorkommenden Variablen.

Das Selektionsprädikat  $F_i$  setzt sich aus einer Menge konjunktiv verknüpfter Bedingungen zusammen.

# Auswertungs-Algorithmus (abstrakt)

- Falls in  $q_i(\dots, c, \dots)$  eine Konstante  $c$  an der  $j$ -ten Stelle vorkommt, füge die Bedingung

$$\$j = c$$

hinzu.

- Falls eine Variable  $X$  mehrfach an Positionen  $k$  und  $l$  in  $q_i(\dots, X, \dots, X, \dots)$  vorkommt, füge für jedes solches Paar die Bedingung

$$\$k = \$l$$

hinzu.

# Auswertungs-Algorithmus (abstrakt)

Für eine Variable  $Y$ , die nicht in den normalen Prädikaten vorkommt, gibt es zwei Möglichkeiten:

- Sie kommt nur als Prädikat

$$Y = c$$

für eine Konstante  $c$  vor. Dann wird eine einstellige Relation mit einem Tupel

$$Q_Y := \{[c]\}$$

gebildet.

- Sie kommt als Prädikat

$$X = Y$$

vor, und  $X$  kommt in einem normalen Prädikat  $q_i(\dots, X, \dots)$  an  $k$ -ter Stelle vor. In diesem Fall setze

$$Q_Y := \rho_{Y \leftarrow \$k}(\Pi_{\$k}(Q_i)) .$$

# Auswertungs-Algorithmus (abstrakt)

Nun bilde man den Algebra-Ausdruck

$$E := E_1 \times \dots \times E_n$$

und wende anschließend

$$\sigma_F(E)$$

an, wobei  $F$  aus der konjunktiven Verknüpfung der Vergleichsprädikate

$$X \text{ phi } Y$$

besteht, die in der Regel vorkommen. Schließlich projizieren wir noch auf die im Kopf der Regel vorkommenden Variablen:

$$\Pi_{x_1, \dots, x_m}(\sigma_F(E))$$

# Beispiel: nahe verwandte Vorlesungen

Wir wollen diese Vorgehensweise nochmals am Beispiel demonstrieren:

$$(r_1) \text{ nvV}(N1, N2) \text{ :- } gV(N1, N2).$$

$$(r_2) \text{ nvV}(N1, N2) \text{ :- } gV(M1, M2), vs(M1, N1), vs(M2, N2).$$

Dieses Beispielprogramm baut auf dem Prädikat  $gV$  auf und ermittelt nahe verwandte Vorlesungen, die einen gemeinsamen Vorgänger erster oder zweiter Stufe haben. Für die erste Regel erhält man folgenden Algebra-Ausdruck:

$$E_{r_1} := \Pi_{N1, N1} (\sigma_{true} (GV(N1, N2)))$$

Für die zweite Regel ergibt sich gemäß dem oben skizzierten Algorithmus:

$$E_{r_2} := \Pi_{N1, N2} (GV(M1, M2)AVs1(M1, N1)AVs2(M2, N2)).$$

Daraus ergibt sich dann durch die Vereinigung

$$NvV := E_{r_1} \cup E_{r_2}$$

die Relation  $NvV$ , die durch das Prädikat  $nvV$  definiert ist. Die Leser mögen bitte die Auswertung dieses Relationenalgebra-Ausdrucks an unserer Beispiel-Datenbasis durchführen.

# Auswertung rekursiver Regeln

$a(V,N) :- vs(V,N).$

$a(V,N) :- a(V,M), vs(M,N).$

Aufbauen	
V	N
5001	5041
5001	5043
5001	5049
5041	5216
5041	5052
5043	5052
5052	5259
5001	5216
5001	5052
5001	5259
5041	5259
5043	5259

# Auswertung rekursiver Regeln

Betrachten wir das Tupel [5001, 5052] aus der Relation *Aufbauen*. Dieses Tupel kann wie folgt hergeleitet werden:

1.  $a(5001, 5043)$  folgt aus der ersten Regel, da  $vs(5001, 5043)$  gilt.
2.  $a(5001, 5052)$  folgt aus der zweiten Regel, da
  - a)  $a(5001, 5043)$  nach Schritt 1. gilt und
  - b)  $vs(5043, 5052)$  gemäß der EDB-Relation *Voraussetzen* gilt.

# Naive Auswertung durch Iteration

$$A(V, N) = Vs(V, N) \cup \Pi_{V, N}(A(V, M) \wedge Vs(M, N))$$

$A := \{\}$ : /\*Initialisierung auf die leere Menge \*/

**repeat**

$A' := A$ ;

$A := Vs(V, N)$ ; /\* erste Regel \*/

$A := A \cup \Pi_{V, N}(A'(V, M) \wedge Vs(M, N))$ ; /\* zweite Regel \*/

**until**  $A' = A$

**output**  $A$ ;

# Naive Auswertung durch Iteration

1. Im ersten Durchlauf werden nur die 7 Tupel aus *Voraussetzen* nach *A* „übertragen“, da der Join leer ist (das linke Argument *A'* des Joins wurde zur leeren Relation  $\{\}$  initialisiert).
2. Im zweiten Schritt kommen zusätzlich die Tupel [5001,5216], [5001,5052], [5041,5259] und [5043,5259] hinzu.
3. Jetzt wird nur noch das Tupel [5001,5259] neu generiert.
4. In diesem Schritt kommt kein neues Tupel mehr hinzu, so dass die Abbruchbedingung  $A' = A$  erfüllt ist.

# (Naive) Auswertung der rekursiven Regel *aufbauen*

Schritt	A
1	[5001,5041], [5001,5043], [5001,5049], [5041,5216], [5041,5052], [5043,5052], [5052,5259]
2	[5001,5041], [5001,5043], [5001,5049], [5041,5216], [5041,5052], [5043,5052], [5052,5259] [5001,5216], [5001,5052], [5041,5259], [5043,5259],
3	[5001,5041], [5001,5043], [5001,5049], [5041,5216], [5041,5052], [5043,5052], [5052,5259] [5001,5216], [5001,5052], [5041,5259], [5043,5259], [5001,5259]
4	wie in Schritt 3 (keine Veränderung, also Terminierung des Algorithmus)

# Inkrementelle (semi-naive)

## Auswertung rekursiver Regeln

Die Schlüsselidee der *semi-naiven Auswertung* liegt in der Beobachtung, dass für die Generierung eines neuen Tupels  $t$  der rekursiv definierten IDB-Relation  $P$  eine bestimmte Regel

$$p(\dots) \text{ :- } q_1(\dots), \dots, q_n(\dots).$$

für Prädikat  $p$  „verantwortlich“ ist. Dann wird also im iterativen Auswertungsprogramm ein Algebra-Ausdruck der Art

$$E(Q_1 \text{ A } \dots \text{ A } Q_n)$$

iterativ ausgewertet. Es reicht aber aus

$$E(\Delta Q_1 \text{ A } Q_2 \text{ A } \dots \text{ A } Q_n) \cup E(Q_1 \text{ A } \Delta Q_2 \text{ A } \dots \text{ A } Q_n) \cup \dots \cup$$

$$E(Q_1 \text{ A } Q_2 \text{ A } \dots \text{ A } \Delta Q_n)$$

auszuwerten.

$$E(\underbrace{Q_1}_{t_1} \text{ A } \dots \text{ A } \underbrace{Q_{i-1}}_{t_{i-1}} \text{ A } \Delta \underbrace{Q_i}_{t_i} \text{ A } \underbrace{Q_{i+1}}_{t_{i+1}} \text{ A } \dots \text{ A } \underbrace{Q_n}_{t_n})$$

$$t = [5001,5259]$$

Dieses Tupel wurde aus dem folgenden Join gebildet:

$$\begin{array}{c} [5001,5052] \text{ A } [5052,5259] \\ \underbrace{1 \ 4 \ 2 \ 4 \ 3}_{t_1 \in A} \quad \underbrace{1 \ 4 \ 2 \ 4 \ 3}_{t_2 \in S} \end{array}$$

# Programm zur semi-naiven Auswertung von *aufbauen*

1.  $A := \{\}; \Delta V_S := \{\};$
2.  $\Delta A := V_S(V, N);$  /\* erste Regel \*/
3.  $\Delta A := \Delta A \cup \Pi_{V, N}(A(V; M) \Delta V_S(M, N));$  /\* zweite Regel \*/
4.  $A := \Delta A;$
5. **repeat**
6.      $\Delta A' := \Delta A;$
7.      $\Delta A := \Delta V_S(V, N);$  /\* erste Regel, liefert  $\emptyset$  \*/
8.      $\Delta A := \Delta A \cup$  /\* zweite Regel \*/
9.          $\Pi_{V, N}(\Delta A'(V, M) \Delta V_S(M, N)) \cup$
10.          $\Pi_{V, N}(A(V, M) \Delta V_S(M, N));$
11.      $\Delta A := \Delta A - A;$  /\* entferne "neue" Tupel, die schon vorhanden waren \*/
12.      $A := A \cup \Delta A;$
13. **until**  $\Delta A = \emptyset;$

# Illustration der semi-naiven Auswertung von *Aufbauen*

Schritt	$\Delta A$
Initialisierung (Zeile 2. und 3.)	(sieben Tupel aus VS) [5001,5042], [5001,5043] [5043,5052], [5041,5052] [5001,5049], [5001,5216] [5052,5259]
1. Iteration	(Pfade der Länge 2) [5001,5216], [5001,5052] [5041,5259], [5043,5259]
2. Iteration	(Pfade der Länge 3) [5001,5259]
3. Iteration	$\emptyset$ (Terminierung)

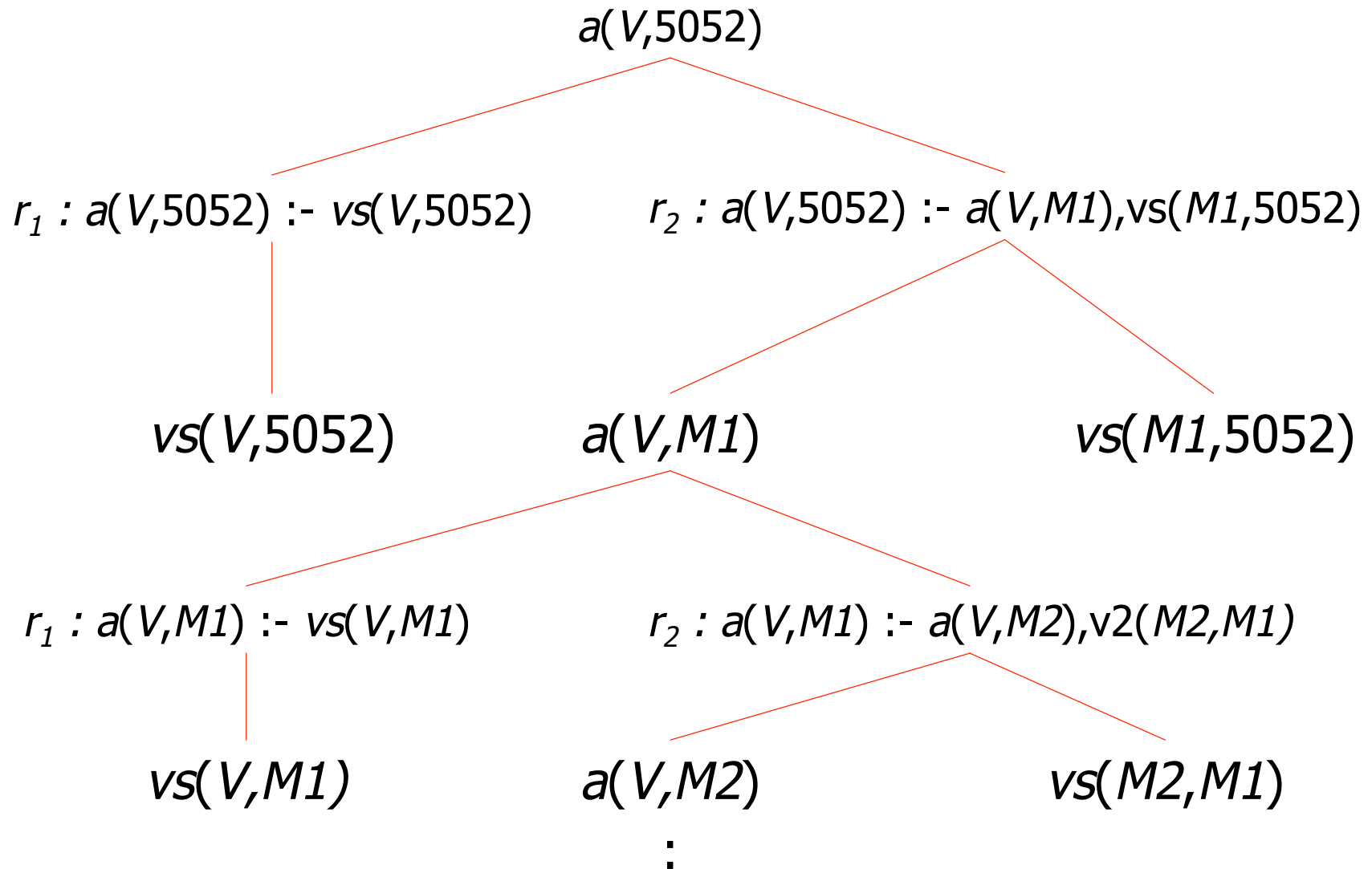
# Bottom-Up oder Top-Down Auswertung

$(r_1) a(V, N) :- vs(V, N).$

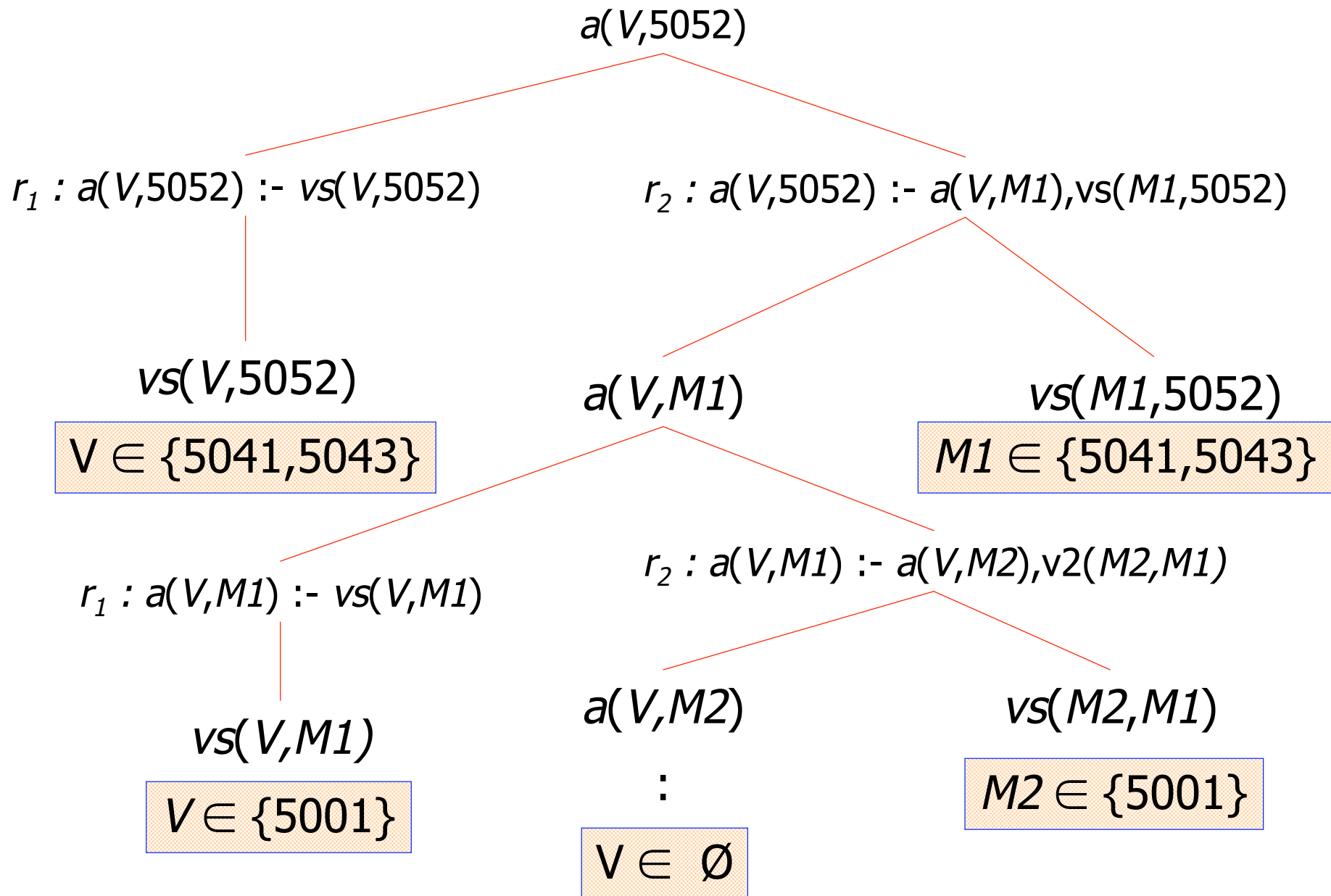
$(r_2) a(V, N) :- a(V, M), vs(M, N).$

query(V) :- A(V, 5052).

# Rule/Goal-Baum zur Top-Down Auswertung



# Rule/Goal-Baum mit Auswertung



# Negation im Regelrumpf

indirektAufbauen( $V, N$ ) :- aufbauen( $V, N$ ),  $\neg$ voraussetzen( $V, N$ )

## Stratifizierte Datalog-Programme

Eine Regel mit einem negierten Prädikat im Rumpf, wie z.B.

$$r \equiv p(\dots) :- q_1(\dots), \dots, \neg q_i(\dots), \dots, q_n(\dots).$$

kann nur dann sinnvoll ausgewertet werden, wenn  $Q_i$  schon vollständig materialisiert ist. Also müssen zuerst alle Regeln mit Kopf  $q_i(\dots) :- \dots$  ausgewertet sein. Das geht nur, wenn  $q_i$  nicht abhängig von  $p$  ist.

Also darf der Abhängigkeitsgraph keine Pfade von  $q_i$  nach  $p$  enthalten. Wenn das für alle Regeln der Fall ist, nennt man das Datalog-Programm *stratifiziert*.

# Auswertung von Regeln mit Negation

$$iA(V, N): -a(V, N), \neg vs(V, N).$$

$$\begin{aligned} iA(V, N) &= \Pi_{V, N} (A(V, N) \wedge \overline{vs}(V, N)) \\ &= A(V, N) - vs(V, N) \end{aligned}$$

$$\overline{Q}_i := \left( \underbrace{DOM \times \dots \times DOM}_{k\text{-mal}} \right) - Q_i$$

# Ein etwas komplexeres Beispiel

$grundlagen(V) : \neg voraussetzen(V, N)$

$spezialVorl(V) : \neg vorlesungen(V, T, S, R), \neg grundlagen(V)$

$Grundlagen(V) := \Pi_V (Voraussetzen(V, N))$

$SpezialVorl(V) := \Pi_V (Vorlesungen(V, T, S, R) \wedge \overline{Grundlagen(V)})$

Hierbei ist  $\overline{Grundlagen(V)}$  als  $DOM - Grundlagen(V)$  definiert.

$SpezialVorl(V) := \Pi_V (Vorlesungen(V, T, S, R)) - Grundlagen(V)$

# Ausdruckskraft von Datalog

- Die Sprache Datalog, eingeschränkt auf nicht-rekursive Programme aber erweitert um Negation, wird in der Literatur manchmal als *Datalog<sup>-non-rec</sup>* bezeichnet
- Diese Sprache *Datalog<sup>-non-rec</sup>* hat genau die gleiche Ausdruckskraft wie die relationale Algebra – und damit ist sie hinsichtlich Ausdruckskraft auch äquivalent zum relativen Tupel- und Domänenkalkül
- Datalog mit Negation und Rekursion geht natürlich über die Ausdruckskraft der relationalen Algebra hinaus – man konnte in Datalog ja z.B. die transitive Hülle der Relation *Voraussetzen* definieren.

# Datalog-Formulierung der relationalen Algebra-Operatoren

## Selektion

$\sigma_{SWS > 3}(\text{Vorlesungen})$

$query(V, T, S, R): -\text{vorlesungen}(V, T, S, R), S > 3.$

$query(V, S, R): -\text{vorlesungen}(V, \text{"Mäeutik"}, S, R)$

## Projektion

$query(\text{Name}, \text{Rang}): -\text{professoren}(\text{PersNr}, \text{Name}, \text{Rang}, \text{Raum}).$

## Join

$\Pi_{\text{Titel}, \text{Name}}(\text{Vorlesungen} \bowtie_{\text{gelesenVor} = \text{PersNr}} \text{Professoren})$

$query(T, N): -\text{vorlesungen}(V, T, S, R), \text{professoren}(R, N, Rg, Ra).$

# Datalog-Formulierung der relationalen Algebra-Operatoren

## Kreuzprodukt

$query(V1, V2, V3, V4, P1, P2, P3, P4) : \neg vorlesungen(V1, V2, V3, V4),$   
 $professoren(P1, P2, P3, P4).$

Professoren  $\times$  Vorlesungen

## Vereinigung

$\Pi_{PersNr, Name} (Assistenten) \cup \Pi_{PersNr, Name} (Professoren)$

$query(PersNr, Name) : \neg assistenten(PersNr, Name, F, B).$

$query(PersNr, Name) : \neg professoren(PersNr, Name, Rg, Ra).$

# Datalog-Formulierung der relationalen Algebra-Operatoren

## Mengendifferenz

$$\Pi_{\text{vorlNr}}(\text{Vorlesungen}) - \Pi_{\text{Vorgänger}}(\text{Voraussetzen})$$

$$\text{vorlNr}(V): \neg \text{vorlesungen}(V, T, S, R)$$

$$\text{grundlagen}(V): \neg \text{voraussetzen}(V, N)$$

$$\text{query}(V): \neg \text{vorlNr}(V), \neg \text{grundlagen}(V)$$