

1

Terminologie

TUM

- 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 Datenbank.
- 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.

2

2

Datalog

TUM

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

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

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

3

3

Eine Datalog-Regel

TUM

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

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:

- *Prädikate* beginnen mit einem Kleinbuchstaben.
- Die zugehörigen *Relationen* – seien es *EDB-* oder *IDB-Relationen* – werden mit gleichem Namen, aber mit einem Großbuchstaben beginnend, bezeichnet.

4

4

Beispiel Datalog-Programm

TUM

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}

5

5

Analogie zur EDB/IDB in rel. DBMS

TUM

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 rekursive CTE (Common Table Expression) in SQL:

```

with recursive 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
    
```

6

6

Rekursion in SQL

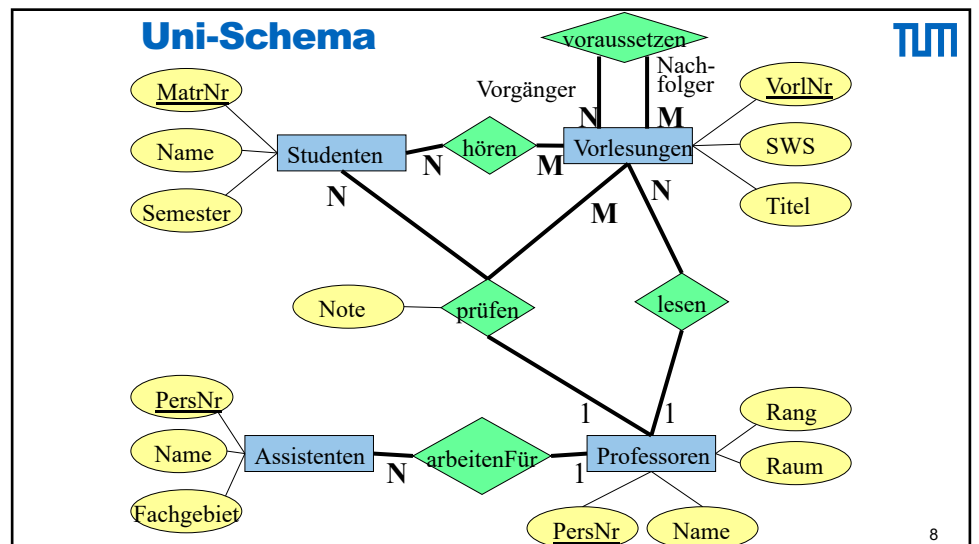
TUM

```

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

7

7



8

Professoren				Studenten			Vorlesungen			
PersNr	Name	Rang	Raum	MatrNr	Name	Semester	VorlNr	Titel	SWS	gelesen Von
2125	Sokrates	C4	226	24002	Xenokrates	18	5001	Grundzüge	4	2137
2126	Russel	C4	232	25403	Jonas	12	5041	Ethik	4	2125
2127	Kopernikus	C3	310	26120	Fichte	10	5043	Erkenntnistheorie	3	2126
2133	Popper	C3	52	26830	Aristoxenos	8	5049	Mäeutik	2	2125
2134	Augustinus	C3	309	27550	Schopenhauer	6	4052	Logik	4	2125
2136	Curie	C4	36	28106	Carnap	3	5052	Wissenschaftstheorie	3	2126
2137	Kant	C4	7	29120	Theophrastos	2	5216	Bioethik	2	2126
				29555	Feuerbach	2	5259	Der Wiener Kreis	2	2133
							5022	Glaube und Wissen	2	2134
							4630	Die 3 Kritiken	4	2137

voraussetzen		prüfen	
Vorgänger	Nachfolger	MatrNr	VorlNr
5001	5041	26120	5001
5001	5043	27550	5001
5001	5049	27550	4052
5041	5216	28106	5041
5043	5052	28106	5052
5041	5052	28106	5259
5052	5259	29120	5001
		29120	5041
		29120	5049
		29555	5022
		25403	5022

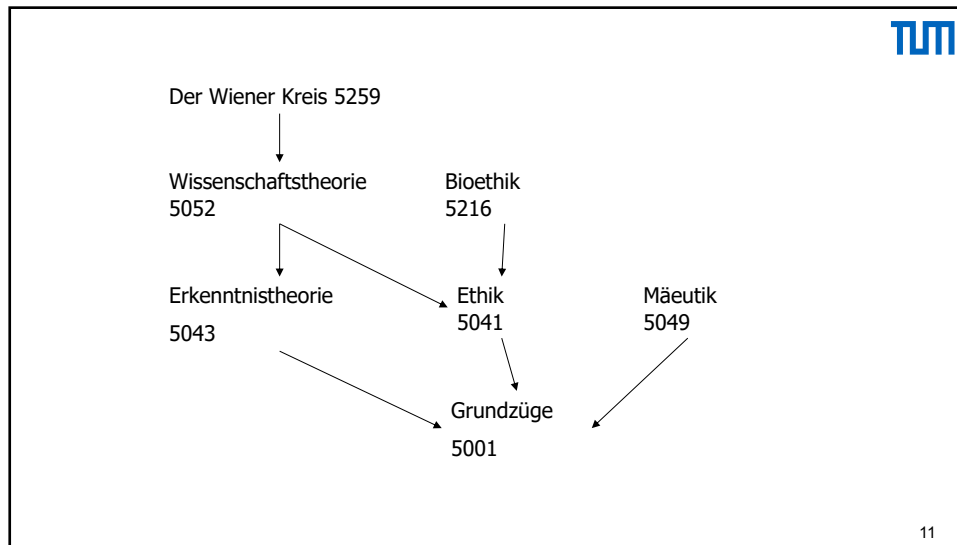
Assistenten			
PersNr	Name	Fachgebiet	Boss
3002	Platon	Ideenlehre	2125
3003	Aristoteles	Syllogistik	2125
3004	Wittgenstein	Sprachtheorie	2126
3005	Rhetikus	Planetenbewegung	2127
3006	Newton	Keplersche Gesetze	2127
3007	Spinoza	Gott und Natur	2126

9

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

10



11

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`
    
```

12

Rekursion TUM

```

select v1.Vorgänger
from voraussetzen v1, voraussetzen v2, Vorlesungen v
where v1.Nachfolger= v2.Vorgänger and
        v2.Nachfolger= v.VorINr 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			
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

13

13

Rekursion TUM

```

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

|

14

14

Vorgänger des „Wiener Kreises“ der Tiefe n TUM

```

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.VorINr and
        v.Titel= `Der Wiener Kreis`
    
```

15

15

Transitive Hülle TUM

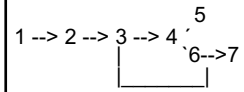
$$\text{trans}_{A,B}(R) = \{(a,b) \mid \exists k \in \mathbb{N} (\exists I1, \dots, Ik \in R ($$

$$\begin{aligned}
 & I1.A = I2.B \wedge \\
 & \vdots \\
 & Ik-1.A = Ik.B \wedge \\
 & I1.A = a \wedge \\
 & Ik.B = b))\}
 \end{aligned}$$

16

16

Rekursion in Prolog/Datalog



```

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).
    
```

21

21

Kanten	
Von	Nach
1	2
2	3
3	4
3	6

Fakten

kante(1,2).
kante(2,5).
...

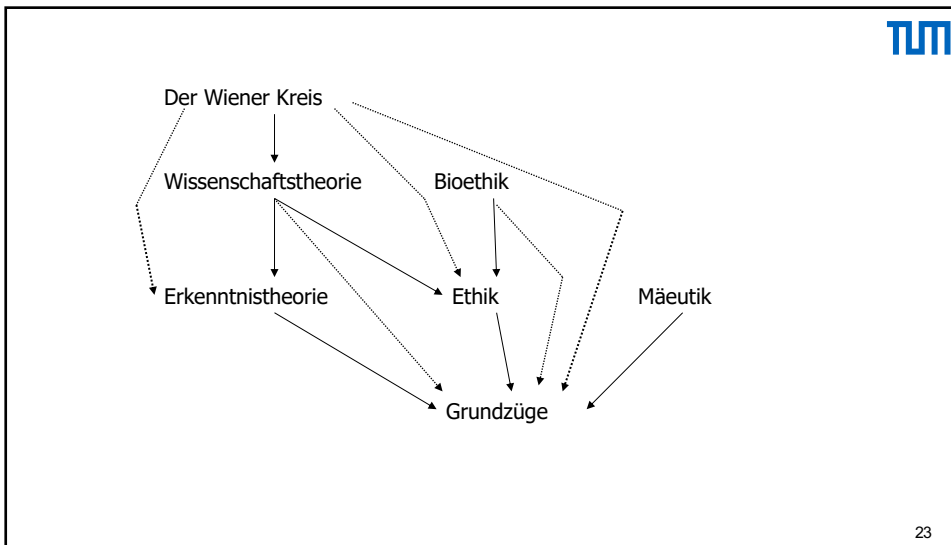
Regeln

pfads(x,y) :- kante(x,y).
pfad(x,y) :- kante(x,z), pfad(z,y).
A



22

22

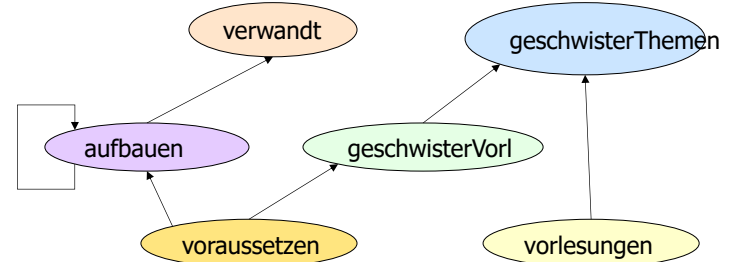


23

23

Eigenschaften von Datalog-Programmen

Abhängigkeitsgraph (→ = „wird verwendet von“)



- Ein Datalog-Programm ist rekursiv, wenn der Abhängigkeitsgraph einen (oder mehrere) Zyklen hat
- Unser Beispielprogramm ist rekursiv wegen aufbauen → aufbauen

24

24

Sicherheit von Datalog-Regeln TUM

Es gibt unsichere Regeln, wie z.B.

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

Diese definieren unendliche Relationen.

Eine Datalog-Regel ist sicher, wenn alle Variablen im Kopf Bereichsbeschränkt (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.

25

25

Ein zyklenfreier Abhängigkeitsgraph TUM

$$gV(N1, N2) :- vs(V, N1), vs(V, N2), N1 < N2.$$

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

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

26

26

Auswertung nicht-rekursiver Datalog-Programme TUM

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

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

27

27

Auswertung von Geschwister-Vorlesungen und Geschwister-Themen TUM

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

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

$$Vs1(V, N1) := \rho_{V \leftarrow S1} (\rho_{N1 \leftarrow S2} (\rho_{Vs1}(\text{Voraussetzen})))$$

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:

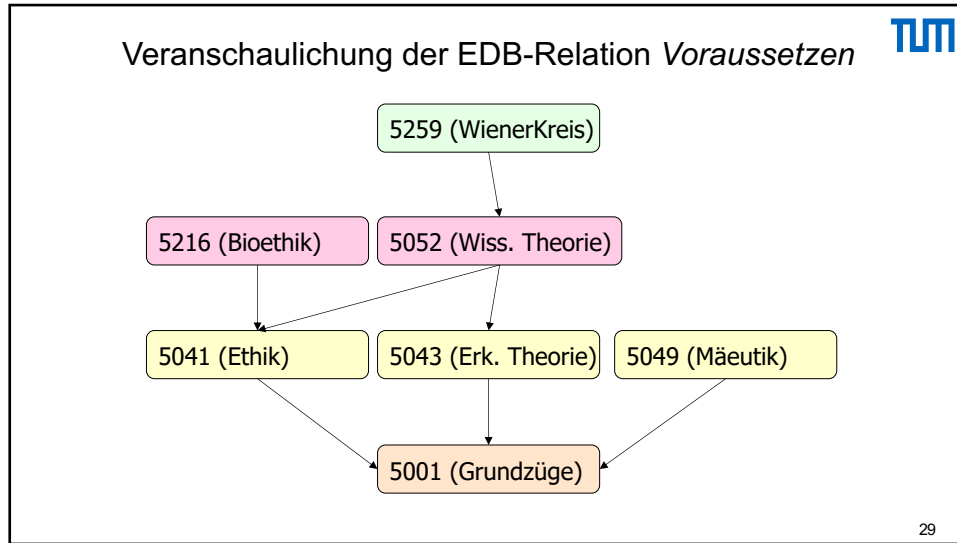
$$GV(N1, N2) := \Pi_{N1, N2} (\sigma_{N1 < N2} (Vs1(V, N1) \bowtie Vs2(V, N2)))$$

Analog ergibt sich für die Herleitung von gT :

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

28

28



29

Ausprägung der Relationen *GeschwisterVorl* und *GeschwisterThemen*

GeschwisterVorl		GeschwisterThemen	
<i>N1</i>	<i>N2</i>	<i>T1</i>	<i>T2</i>
5041	5043	Ethik	Erkenntnistheorie
5043	5049	Erkenntnistheorie	Mäeutik
5041	5049	Ethik	Mäeutik
5052	5216	Wissenschaftstheorie	Bioethik

30

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.

31

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.

32

Auswertungs-Algorithmus (abstrakt) TUM

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 gebildet.

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

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

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

33

33

Auswertungs-Algorithmus (abstrakt) TUM

Nun bilde man den Algebra-Ausdruck $E := E_1 \bowtie \dots \bowtie E_n$ und wende anschließend $\sigma_F(E)$ an, wobei F aus der konjunktiven Verknüpfung der Vergleichsprädikate $X \phi Y$ besteht, die in der Regel vorkommen. Schließlich projizieren wir noch auf die im Kopf der Regel vorkommenden Variablen:

$$\Pi_{X_1, \dots, X_m}(\sigma_F(E))$$

34

34

Beispiel: nahe verwandte Vorlesungen TUM

Wir wollen diese Vorgehensweise nochmals am Beispiel demonstrieren:

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

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

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:

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

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

$$E_{r_2} := \Pi_{N1, N2}(\text{GV}(M1, M2) \bowtie \text{Vs1}(M1, N1) \bowtie \text{Vs2}(M2, N2)).$$

Daraus ergibt sich dann durch die Vereinigung

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

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

35

35

Auswertung rekursiver Regeln TUM

$$a(V, N) :- \text{vs}(V, N).$$

$$a(V, N) :- a(V, M), \text{vs}(M, N).$$

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

36

36

Auswertung rekursiver Regeln TUM

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.

37

37

Naive Auswertung durch Iteration TUM

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

A := {}; /*Initialisierung auf die leere Menge */
repeat
 A' := *A*;
 A := *Vs*(*V, N*); /* erste Regel */
 A := *A* \cup $\Pi_{V,N}(A'(V, M) \bowtie Vs(M, N))$; /*zweite Regel */
until *A*' = *A*
output *A*;

38

38

Naive Auswertung durch Iteration TUM

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.

39

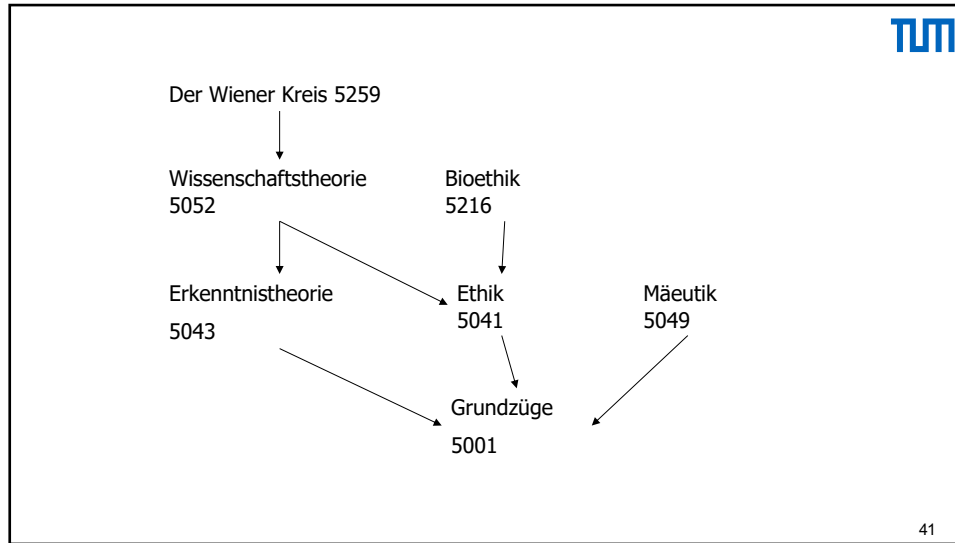
39

(Naive) Auswertung der rekursiven Regel *aufbauen* TUM

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)

40

40



41

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

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

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

$$E(Q_1 \bowtie \dots \bowtie Q_n)$$

iterativ ausgewertet. Es reicht aber aus

$$E(\Delta Q_1 \bowtie Q_2 \bowtie \dots \bowtie Q_n) \cup E(Q_1 \bowtie \Delta Q_2 \bowtie \dots \bowtie Q_n) \cup \dots \cup E(Q_1 \bowtie Q_2 \bowtie \dots \bowtie \Delta Q_n)$$

auszuwerten.
Betrachte Tupel $t = [5001, 5259]$

Dieses Tupel wurde aus dem folgenden Join gebildet:

$$\underbrace{[5001, 5052]}_{t_1 \in A} \bowtie \underbrace{[5052, 5259]}_{t_2 \in I_S}$$

42



43

Programm zur semi-naiven Auswertung von *aufbauen*

```

A := { }; ΔVs := { };
ΔA := Vs(V, N); /* erste Regel */
ΔA := ΔA ∪ ΠV,N(A(V, M) ⋈ Vs(M, N)); /* zweite Regel */
A := ΔA;
repeat
    ΔA' := ΔA;
    ΔA := ΔVs(V, N); /* erste Regel, liefert ∅ */
    ΔA := ΔA ∪ /* zweite Regel */
        ΠV,N(ΔA'(V, M) ⋈ Vs(M, N)) ∪
        ΠV,N(A(V, M) ⋈ ΔVs(M, N));
    ΔA := ΔA - A; /* entferne "neue" Tupel, die schon vorhanden waren */
    A := A ∪ ΔA;
until ΔA = ∅;
    
```

44

Illustration der semi-naiven Auswertung von *Aufbauen*



Schritt	ΔA
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)

45

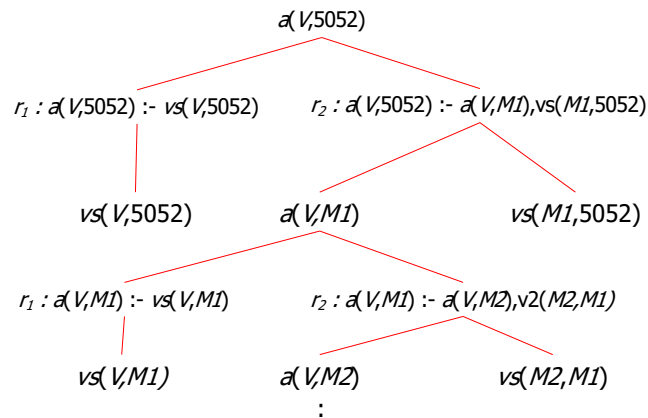
Bottom-Up oder Top-Down Auswertung



$(r_1) a(V,N) :- vs(V,N).$
 $(r_2) a(V,N) :- a(V,M),vs(M,N).$
 query(V) :- a(V,5052).

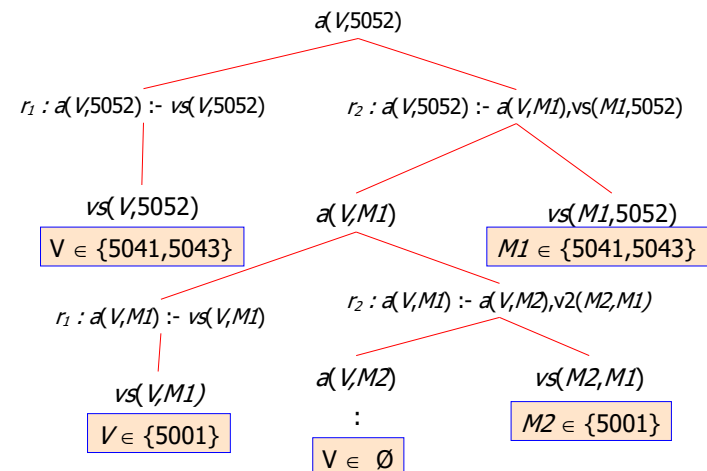
46

Rule/Goal-Baum zur Top-Down Auswertung



47

Rule/Goal-Baum mit Auswertung



48

Negation im Regelrumpf



indirektAufbauen(V,N) :- aufbauen(V,N), ¬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 p nach q_i enthalten. Wenn das für alle Regeln der Fall ist, nennt man das Datalog-Programm *stratifiziert*.

49

Auswertung von Regeln mit Negation



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

$$\begin{aligned} iA(V, N) &= \Pi_{V, N} \left(A(V, N) \boxtimes \overline{Vs}(V, N) \right) \\ &= A(V, N) - Vs(V, N) \end{aligned}$$

Komplement – Bildung

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

50

49

50

Ein etwas komplexeres Beispiel



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

$$\text{spezialVorl}(V) :- \text{vorlesungen}(V, T, S, R), \neg \text{grundlagen}(V).$$

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

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

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

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

51

Ausdruckskraft von Datalog



Die Sprache Datalog, eingeschränkt auf nicht-rekursive Programme aber erweitert um Negation, wird in der Literatur manchmal als *Datalog^{¬non-rec}* bezeichnet

Diese Sprache *Datalog^{¬non-rec}* 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.

52

51

52

Datalog-Formulierung der relationalen Algebra-Operatoren

**Selektion**

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

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

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

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

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

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

53

53

Datalog-Formulierung der relationalen Algebra-Operatoren

**Kreuzprodukt**

$$\text{query}(V1, V2, V3, V4, P1, P2, P3, P4) : \text{-vorlesungen}(V1, V2, V3, V4),$$

$$\text{professoren}(P1, P2, P3, P4).$$
Professoren \times Vorlesungen**Vereinigung**

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

$$\text{query}(\text{PersNr}, \text{Name}) : \text{-assistenten}(\text{PersNr}, \text{Name}, F, B).$$

$$\text{query}(\text{PersNr}, \text{Name}) : \text{-professoren}(\text{PersNr}, \text{Name}, Rg, Ra).$$

54

54

Datalog-Formulierung der relationalen Algebra-Operatoren

**Mengendifferenz**

$$\Pi_{\text{VorNr}}(\text{Vorlesungen}) - \Pi_{\text{VorGänger}}(\text{Voraussetzen})$$

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

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

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

55

55