



**Hasso
Plattner
Institut**

IT Systems Engineering | Universität Potsdam

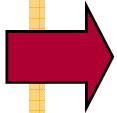
Informationsintegration
Local-as-View: LaV

10.7.2008

Felix Naumann

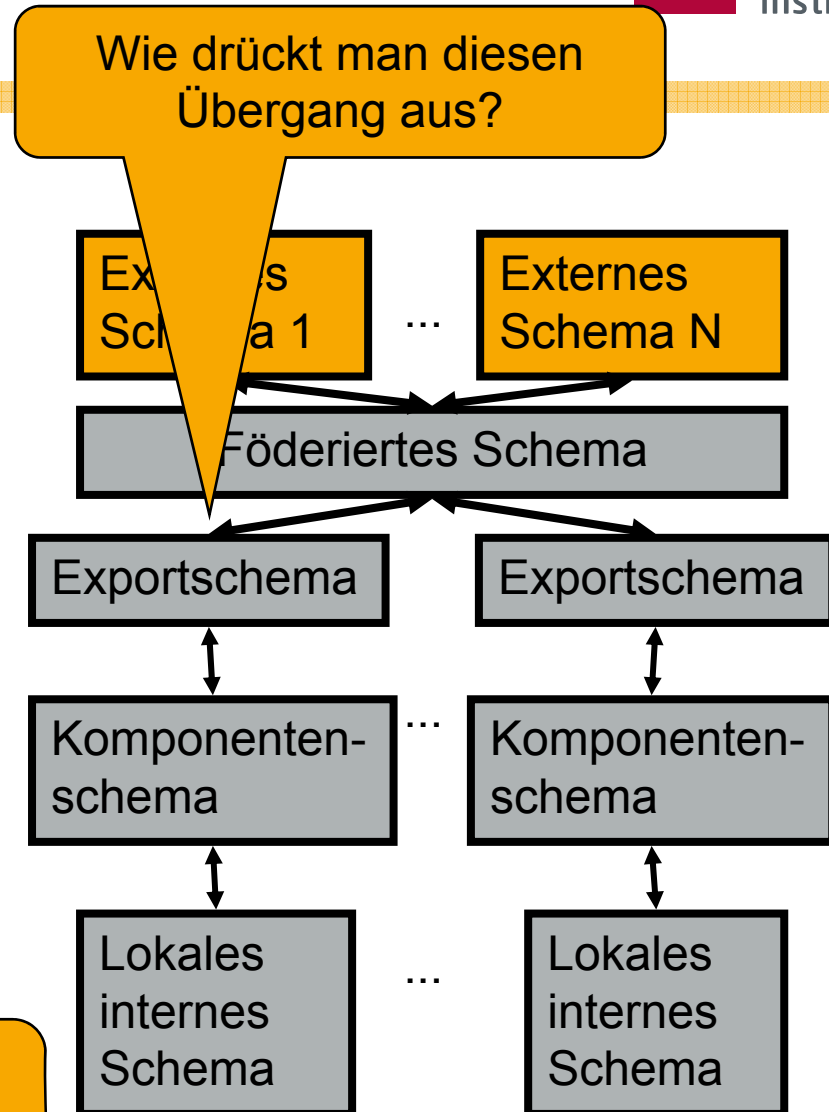
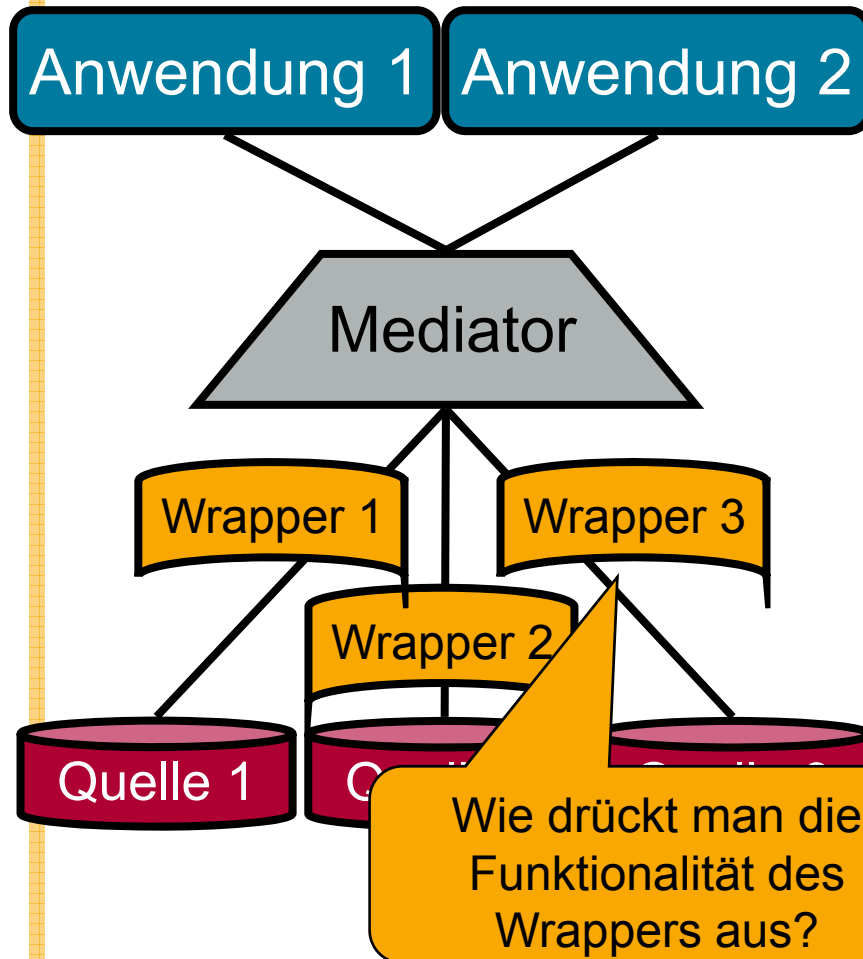
2

- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich



WdH.: Das Problem

3



Modellierung von Datenquellen

4

- Kernidee
 - Modellierung strukturell heterogener Quellen in Bezug auf ein globales Schema als Views (Sichten)
 - Relationales Modell
- Allgemein:
 - Eine Sicht verknüpft mehrere Relationen und produziert eine Relation.
- Sichten zur Verknüpfung von Schemata
 - Sicht definiert auf Relationen eines Schemas und produziert eine Relation des anderen Schemas
- Jetzt: Unterscheidung lokales und globales Schema

Global as View / Local as View

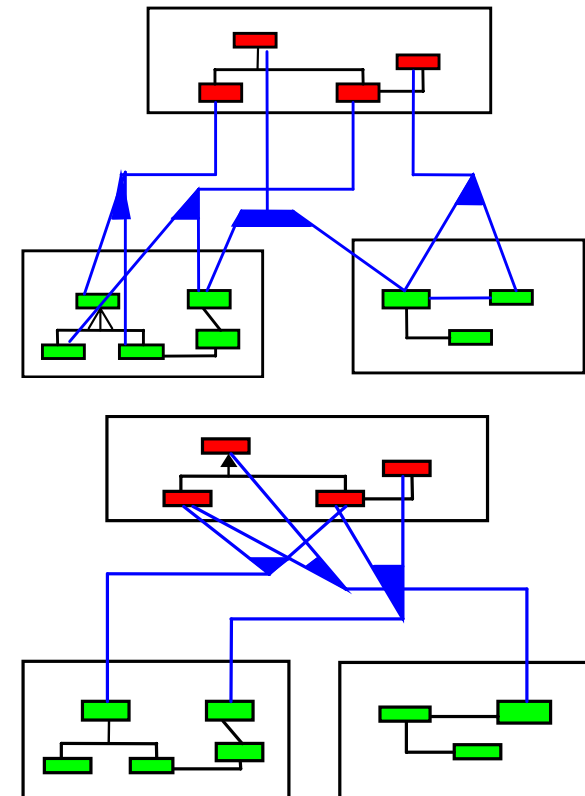
5

- Global as View

- Relationen des globalen Schemas werden als Sichten auf die lokalen Schemas der Quellen ausgedrückt.

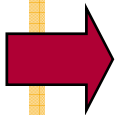
- Local as View

- Relationen der Schemas der Quellen werden als Sichten auf das globale Schema ausgedrückt.



6

- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich

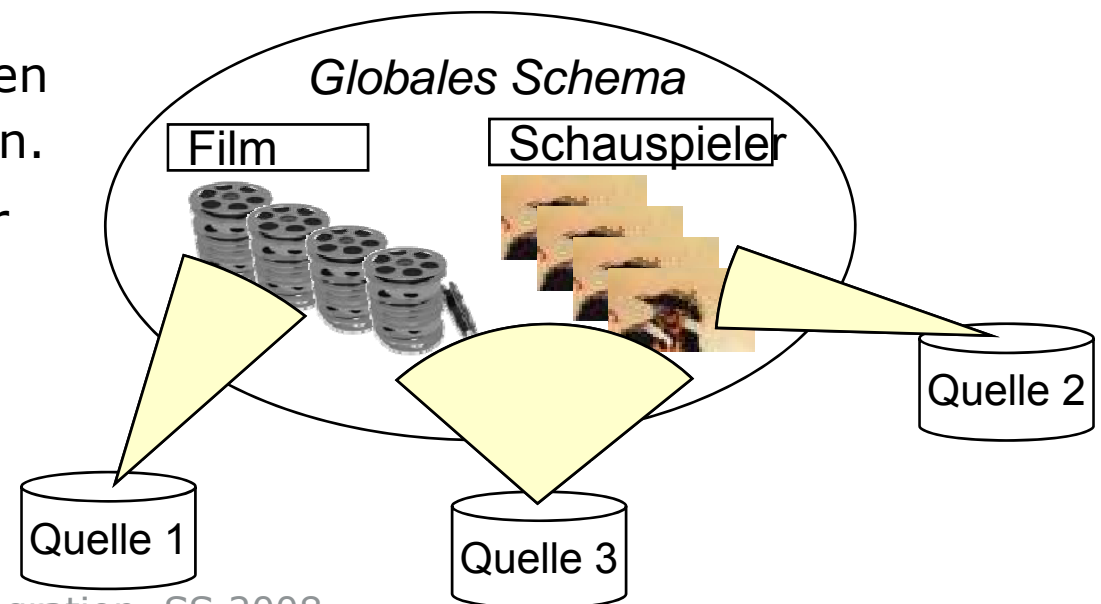


Warum LaV?

7

Andere Sichtweise

- Es gibt in der Welt eine Menge von Filmen, Schauspielern, ...
- Das globale Schema modelliert diese Welt.
- Theoretisch steht damit die Extension fest.
 - Aber niemand kennt sie.
 - Informationsintegration versucht sie herzustellen.
- Quellen speichern Sichten auf die globale Extension.
 - Also Ausschnitte der realen Welt
- Nur die können wir verwenden.



Local as View (LaV) – Beispiel

8

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

S1: IMDB(Titel, Regie, Jahr, Genre)

S2: MyMovies(Titel, Regie, Jahr, Genre)

S3: RegieDB(Titel, Regie)

S4: GenreDB(Titel, Jahr, Genre)

```
CREATE VIEW S1 AS  
SELECT * FROM Film
```

```
CREATE VIEW S2 AS  
SELECT * FROM Film
```

```
CREATE VIEW S3 AS  
SELECT Film.Titel, Film.Regie  
FROM Film
```

```
CREATE VIEW S4 AS  
SELECT Film.Titel, Film.Jahr,  
       Film.Genre  
FROM Film
```

Quelle: VL „Data Integration“, Alon Halevy, University of Washington, 2002

Local as View (LaV) – Beispiel

9

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

S9: ActorDB(Titel, Schauspieler, Jahr)

„Verpasste Chance“

```
CREATE VIEW S9 AS  
SELECT Titel, NULL, Jahr  
FROM Film
```

Local as View (LaV) – Beispiel

10

Globales Schema
Film(Titel, Regie, Jahr, Genre)
Programm(Kino, Titel, Zeit)

S7: KinoDB(Kino, Genre)

```
CREATE VIEW S7 AS  
SELECT Programm.Kino, Film.Genre  
FROM Film, Programm  
WHERE Film.Titel = Programm.Titel
```

- Assoziationen des globalen Schemas können in der Sicht hergestellt werden.

Local as View (LaV) – Beispiel

11

Globales Schema
Film(Titel, Regie, Jahr, Genre)
Programm(Kino, Titel, Zeit)

S9: Filme(Titel, Jahr, Ort, RegieID)
Regie(ID, Regisseur)

```
CREATE VIEW S7Filme AS  
SELECT Titel, Jahr, NULL, NULL  
FROM Film
```

```
CREATE VIEW S7Regie AS  
SELECT NULL, Regie  
FROM Film
```

- Assoziationen des lokalen Schemas können nicht abgebildet werden.

Local as View (LaV) – Beispiel

12

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

S8: NeueFilme(Titel, Regie, Genre)
(IC: Jahr > 2000)

```
CREATE VIEW S8 AS  
SELECT Titel, Regie, Genre  
FROM Film  
WHERE Jahr > 2000
```

- IC auf der Quelle kann modelliert werden
 - Wenn das Attribut im globalen Schema existiert
- ICs müssen in der Quelle nicht explizit definiert werden
 - Auch implizite Einschränkungen können in den View

Local as View (LaV) – Globale ICs

13

Globales Schema

NeuerFilm(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

Nebenbedingung: Jahr > 2000

S1: IMDB(Titel, Regie, Jahr, Genre)

S2: MyMovies(Titel, Regie, Jahr, Genre)

```
CREATE VIEW S1 AS
SELECT * FROM NeuerFilm
(WHERE Jahr > 2000)
```

```
CREATE VIEW S2 AS
SELECT * FROM NeuerFilm
(WHERE Jahr > 2000)
```

- Nebenbedingungen auf dem globalen Schema können wir nicht sinnvoll modellieren.
- Das ging aber bei GaV
- Also hat beides Stärken und Schwächen

Local as View (LaV) – lokale ICs

14

Globales Schema
 Film(Titel, Regie, Jahr, Genre)

S1: AlleFilmeNett(Titel, Regie, Jahr, Genre)
 S2: AlleFilmeBöse(Titel, Regie, Genre)
 S3: NeueFilmeNett(Titel, Regie, Jahr, Genre)
 (Nebenbedingung: Jahr > 2000)
 S4: NeueFilmeBöse(Titel, Regie, Genre)
 (Nebenbedingung: Jahr > 2000)
 S5: AktuelleFilme(Titel, Regie, Genre)
 (Nebenbedingung: Jahr = 2004)

Modellierung zur Optimierung

Modellierung zur Beantwortbarkeit

```
CREATE VIEW S1 AS
SELECT * FROM Film
```

```
CREATE VIEW S2 AS
SELECT Titel, Regie, Genre
FROM Film
```

```
CREATE VIEW S3 AS
SELECT * FROM Film
(WHERE Jahr > 2000)
```

```
CREATE VIEW S4 AS
SELECT Titel, Regie, Genre
FROM Film
WHERE Jahr > 2000
```

```
CREATE VIEW S5 AS
SELECT Titel, Regie, Genre
FROM Film
(WHERE Jahr = 2004)
```

Lokale ICs: Komplexere Beispiele

15

Datenquelle	Beschreibung
spielfilme(titel, regisseur, laenge)	Informationen über Spielfilme, die mindestens 80 Minuten Länge haben.
kurzfilme(titel, regisseur)	Informationen über Kurzfilme. Kurzfilme sind höchstens 10 Minuten lang.
filmkritiken(titel, regisseur, schauspieler, kritik)	Kritiken zu Hauptdarstellern von Filmen
us_spielfilme(titel, laenge, schauspieler_name)	Spielfilme mit US-amerikanischen Schauspielern
spielfilm_kritiken(titel, rolle, kritik)	Kritiken zu Rollen in Spielfilmen
kurzfilm_rollen(titel, rolle, schauspieler_name, nationalitaet)	Rollenbesetzungen in Kurzfilmen

```

film(titel, typ, regisseur, laenge);
schauspieler(schauspieler_name, nationalitaet);
spielt(titel, schauspieler_name, rolle, kritik);

```

```

      film(T, Y, R, L), L > 79, Y = 'Spielfilm'   ⊇   spielfilme(T, R, L)
      film(T, Y, R, L), L < 11, Y = 'Kurzfilm'   ⊇   kurzfilme(T, R)
  film(T, _, R, _), spielt(T, S, O, K), O = 'Hauptrolle' ⊇   filmkritiken(T, R, S, K)
      film(T, Y, _, L), spielt(T, S, _, _),
  schauspieler(S, N), N = 'US', Y = 'Spielfilm'   ⊇   us_spielfilm(T, L, S)
  film(T, Y, _, _), spielt(T, _, O, K), Y = 'Spielfilm' ⊇   spielfilm_kritiken(T, O, K)
      film(T, Y, _, _), spielt(T, S, O, _),
  schauspieler(S, N), Y = 'Kurzfilm'             ⊇   kurzfilm_rollen(T, O, S, N)

```

Felix Naum:

Local as View (LaV) – globale ICs

16

Globales Schema

NeuerFilm(Titel, Regie, Jahr, Genre)

Nebenbedingung: Jahr > 2000

oder

NeuerFilm(Titel, Regie, Genre)

Nebenbedingung: Jahr > 2000

S1: AlleFilmeNett(Titel, Regie, Jahr, Genre)

S2: AlleFilmeBöse(Titel, Regie, Genre)

```
CREATE VIEW AlleFilmeNett AS  
SELECT * FROM NeuerFilm  
(WHERE JAHR > 2000)
```

```
[CREATE VIEW AlleFilmeBöse AS  
SELECT Titel, Regie, Genre  
FROM NeuerFilm]
```

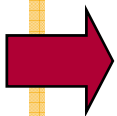
Problem:

Bekannte Nebenbedingung auf dem globalen Schema kann nicht modelliert werden.

Frage:

Ist das schlimm?

- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich



LaV – Anwendungen

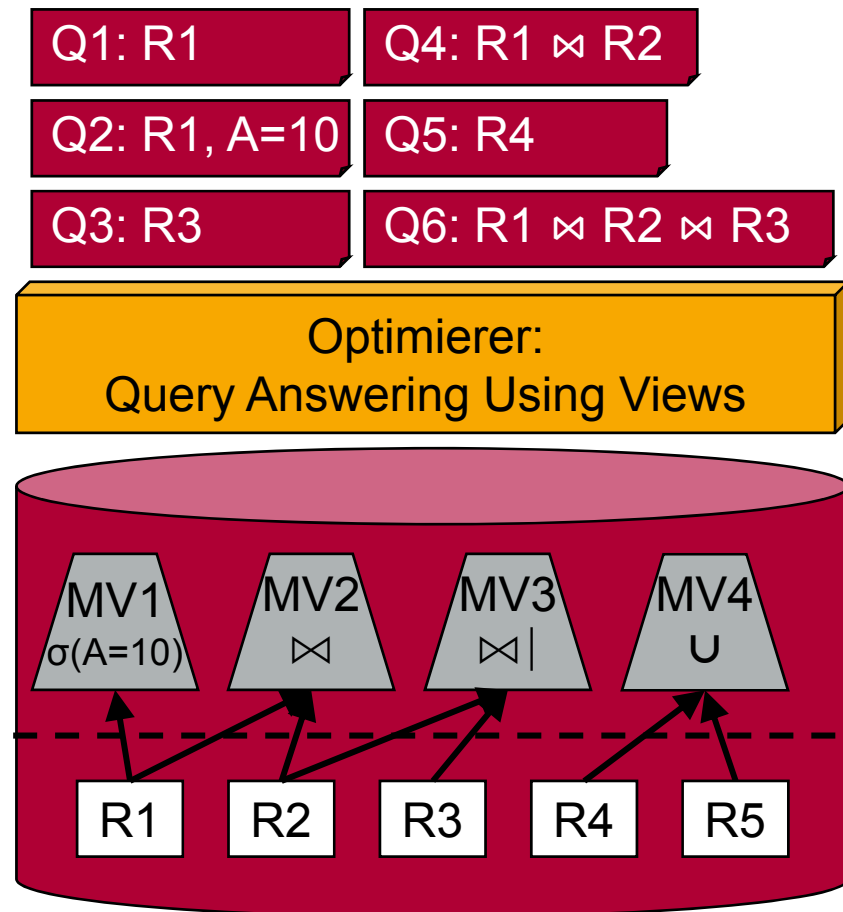
18

- Anfrageoptimierung
 - Materialisierte Sichten auf Datenbankschema
- Datawarehouse Design
 - Materialisierte Sichten auf Warehouse-Schema
- Semantisches Caching
 - Materialisierte Daten beim Client
- Datenintegration
 - Datenquellen als Sichten auf globales (Mediator) Schema

LaV Anwendung: Anfrageoptimierung

19

- Materialisierte Sichten (materialized views, MV) auf Datenbankschema
 - MQT: Materialized Query Table
 - AST: Advanced Summary Table
- Welche Sichten helfen bei der Beantwortung einer Datenbankanfrage durch Vorberechnung von Prädikaten?
- Probleme:
 - Es ist nicht immer besser eine MV zu verwenden (Indizes!).
 - Aktualisierung von MVs
 - ◇ Write auf MV
 - ◇ Write auf Basisrelation



LaV Anwendung: Datawarehouse Design

20

- Sichten auf Warehouse-Schema
- Gegeben eine query workload
 - Query workload = Menge von Anfragen plus Häufigkeiten
- Welche Sichten sollte ich materialisieren?
 - Um alle Anfragen der Workload zu beantworten
- Allgemeiner:
 - Gegeben eine query workload, welche Sichten sollte ich materialisieren um die workload optimal zu beantworten.
 - ◇ Idee: Alle Kombinationen prüfen
 - Frage: Warum ist dieses Problem eigentlich ganz einfach?

LaV Anwendung: Semantisches Caching

21

- In verteilten DBMS
- Materialisierte Daten beim Client
 - Stammen aus (komplexen Anfragen)
- Gegeben eine Anfrage
 - Welche Daten im Cache kann ich zur Beantwortung verwenden?
 - Welche Daten muss ich neu anfragen?

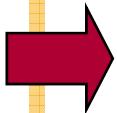
- Auch: Welche Sichten sollte ich vorberechnen?

LaV Anwendung: Datenintegration

22

- Datenquellen als Sichten auf globales (Mediator) Schema
- Fragen:
 - Wie kann ich Antworten auf eine Anfrage an das globale Schema nur mittels der Sichten beantworten?
 - ◇ Unterschied zu Anfrageoptimierung: Keine Basistabellen verfügbar.
 - Kann ich die Anfrage vollständig (extensional) beantworten?

- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich



Anfrageplanung

24

Gegeben

- Eine Anfrage q an das globale Schema
- Lokale Schemata

Gesucht

- Sequenz von Anfragen $q_1 \diamond \dots \diamond q_n$
- Jedes q_i kann von einem Wrapper ausgeführt werden
- Die geeignete Verknüpfung von q_1, \dots, q_n beantwortet q
 - Innerhalb eines Plans durch Joins: $\diamond \rightarrow \bowtie$
 - Verschiedene Pläne werden durch UNION zusammengefasst :
 $\diamond \rightarrow \cup$
- Von $q_1 \bowtie \dots \bowtie q_n$ berechnete Tupel sind korrekte Antworten auf q

Anfrageplan

25

- Wir nennen $q_1 \bowtie \dots \bowtie q_n$ einen Anfrageplan.
- Definition
Gegeben eine globale Anfrage q . Ein Anfrageplan p für q ist eine Anfrage der Form $q_1 \bowtie \dots \bowtie q_n$, so dass
 - jedes q_i kann von genau einem Wrapper ausgeführt werden,
 - und jedes von p berechnete Tupel ist eine semantisch korrekte Antwort für q .
- Bemerkungen
 - „Semantisch korrekt“ haben wir noch nicht definiert.
 - In der Regel gibt es viele Anfragepläne.
 - Die q_i heißen Teilanfragen oder Teilpläne.

Query Containment

26

- Intuitiv wollen wir das folgende:
 - Ein View v liefert (nur) semantisch korrekte Anfragen auf eine globale Anfrage q , wenn seine Extension im Ergebnis von q enthalten ist
- Definition
Sei S ein Datenbankschema, I eine Instanz von S und q_1, q_2 Anfragen gegen I . Sei $q(I)$ das Ergebnis einer Anfrage q angewandt auf I . Dann ist q_1 enthalten in q_2 , geschrieben $q_1 \subseteq q_2$
gdw.
 $q_1(I) \subseteq q_2(I)$ für alle möglichen I
- Bemerkung
 - Der wichtige Teil ist der letzte: „für alle möglichen Instanzen von S “
 - Die können wir natürlich nicht alle ausprobieren

Semantische Korrektheit

27

- Damit können wir definieren, wann ein Plan semantisch korrekt ist (aber wir können das noch nicht testen)
- Definition
Sei S ein globales Schema, q eine Anfrage gegen S , und p eine Verknüpfung von Views v_1, \dots, v_n gegen S , die als linke Seite von LaV Korrespondenzen definiert sind. Dann ist
 p semantisch korrekt für q ,
gdw.
 $p \subseteq q$
- Bemerkung
 - Also ist die Extension von p in der von q enthalten
 - Die Extension von q gibt es natürlich nicht

Viele Anfragepläne

28

■ Definition

Gegeben eine globale Anfrage q . Seien p_1, \dots, p_n die Menge aller Anfragepläne für q . Dann ist das Ergebnis von q definiert als

■ Bemerkungen $result(q) = \bigcup_{i=1..n} result(p_i)$

- Der UNION Operator entfernt Duplikate – dahinter verbirgt sich das Problem der Ergebnisintegration.
- Wie das Ergebnis berechnet wird, ist Sache der Anfrageoptimierung.
 - ◇ Pläne können sich in Teilanfragen überlappen.
- Das Ergebnis von q hängt ab von den definierten Korrespondenzen.

"Query Answering Using Views"

Idee:

- Anfrageumschreibung durch Einbeziehung der Sichten
- Kombiniere geschickt die einzelnen Sichten zu einer Anfrage, so dass deren Ergebnis einen Teil der Anfrage (oder die ganze Anfrage) beantworten.
- Gesamtergebnis ist dann die UNION der Ergebnisse mehrerer Anfrageumschreibungen

LaV – Beispiel

30

Globales Schema

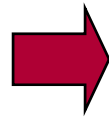
```
Lehrt(prof,kurs_id, sem, eval, univ)
Kurs(kurs_id, titel, univ)
```

Quelle 1: Alle Datenbankveranstalt.

```
CREATE VIEW DB-kurs AS
SELECT K.titel, L.prof, K.kurs_id, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    L.univ = K.univ
AND    K.titel LIKE „%_Datenbanken“
```

Globale Anfrage

```
SELECT prof
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    K.titel LIKE „%_Datenbanken“
AND    L.univ = „HPI“
```



Quelle 2: Alle HPI-Vorlesungen

```
CREATE VIEW HPI-VL AS
SELECT K.titel, L.prof, K.kurs_id, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    K.univ = „HPI“
AND    L.univ = „HPI“
AND    K.titel LIKE „%VL_%“
```

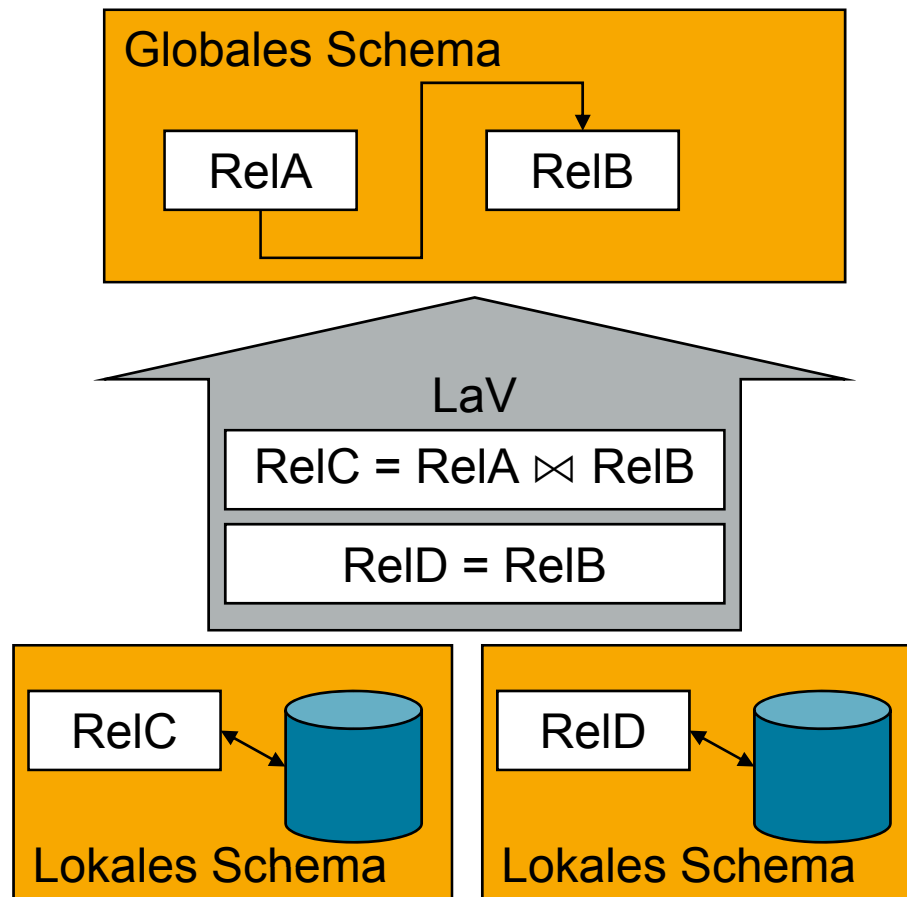
Umgeschriebene Anfrage

```
SELECT prof
FROM   DB-kurs D
WHERE  D.univ = „HPI“
```

Frage: Warum nicht Quelle 2 einbeziehen?

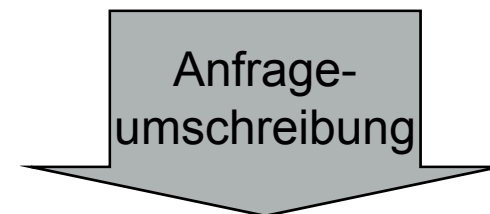
LaV Visualisierung (OWA)

31



Nutzer-Anfrage

```
SELECT ???
FROM   RelB
WHERE  ???
```

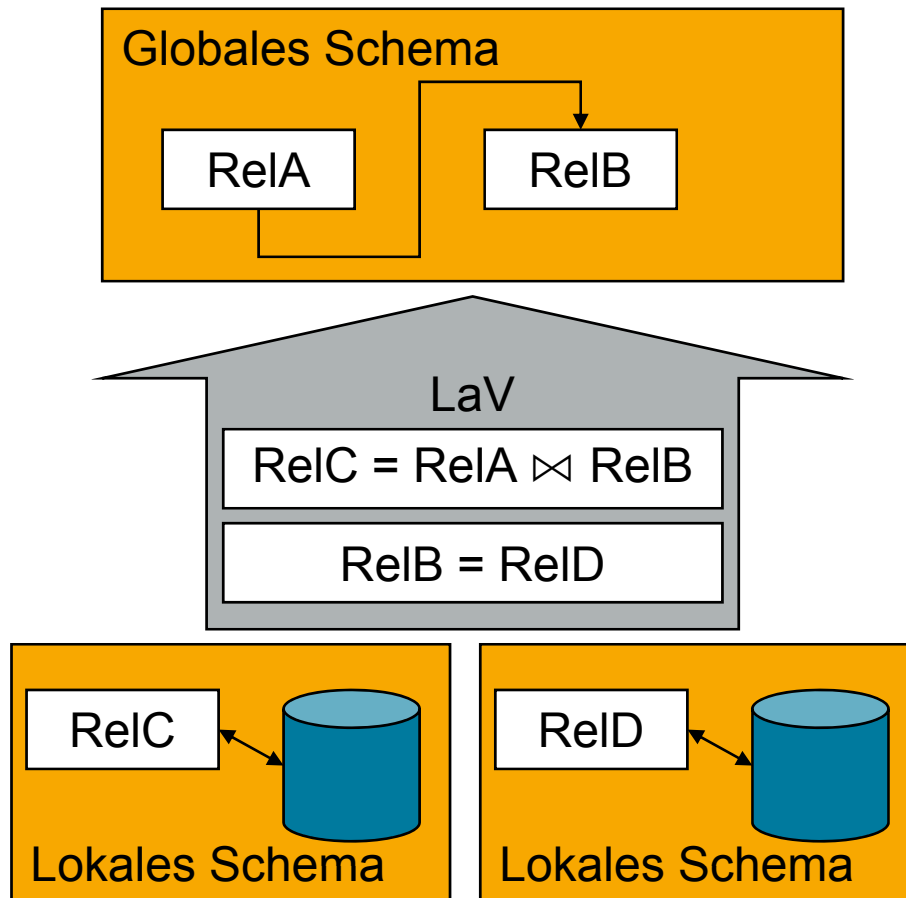


Umgeschriebene Anfrage

```
SELECT ???
FROM   RelD
WHERE  ???
UNION
SELECT Attr(B)
FROM   RelC
WHERE  ???
```

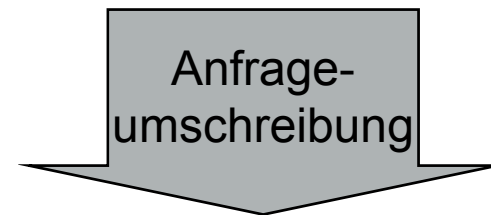
LaV Visualisierung

32



Nutzer-Anfrage

```
SELECT ???
FROM RelA, RelB
WHERE ???
```



Umgeschriebene Anfrage

```
SELECT ???
FROM RelC
WHERE ???
```


LaV – Beispiel

33

Ausschnitt Globales Schema

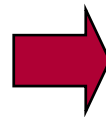
```
Lehrt(prof,kurs_id, sem, eval, univ)
Kurs(kurs_id, titel, univ)
```

Quelle 1: Alle Datenbankveranstalt.

```
CREATE VIEW DB-kurs AS
SELECT K.titel, L.prof, K.kurs_id, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    L.univ = K.univ
AND    K.titel LIKE „%_Datenbanken“
```

Globale Anfrage

```
SELECT titel, kurs_id
FROM Kurs K
WHERE L.univ = „HPI“
```



Quelle 2: Alle HPI-Vorlesungen

```
CREATE VIEW HPI-VL AS
SELECT K.titel, L.prof, K.kurs_id, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    K.univ = „HPI“
AND    L.univ = „HPI“
AND    K.titel LIKE „%VL_%“
```

Umgeschriebene Anfrage

```
SELECT titel, kurs_id
FROM DB-kurs D
WHERE D.univ = „HPI“
UNION
SELECT titel, kurs_id
FROM HPI-VL
```

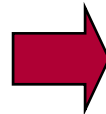
Frage:
Warum hier doch
Quelle 2 einbeziehen?

LaV – Beispiel Vergleich

34

Globale Anfrage

```
SELECT prof
FROM Lehrt L, Kurs K
WHERE L.kurs_id = K.kurs_id
AND K.titel = „VL_Datenbanken“
AND L.univ = „HPI“
```



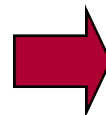
Umgeschriebene Anfrage

```
SELECT prof
FROM DB-kurs D
WHERE D.univ = „HPI“
```

} Vollständige
} Antwort (CWA)

Globale Anfrage

```
SELECT titel, kurs_id
FROM Kurs K
WHERE L.univ = „HPI“
```



Umgeschriebene Anfrage

```
SELECT titel, kurs_id
FROM DB-kurs D
WHERE D.univ = „HPI“
UNION
SELECT titel, kurs_id
FROM HPI-VL
```

} Maximale
} Antwort

Frage:
Was fehlt?

CWA / OWA

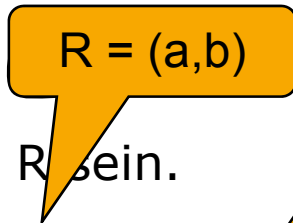
35

- Closed World Assumption (CWA)
 - Vereinigung aller Daten der Basisrelationen entspricht der Menge aller relevanten Daten.
 - Beispiele: Data Warehouse (und traditionelle DBMS)
- Open World Assumption (OWA)
 - Vereinigung aller Daten der Datenquellen ist eine Teilmenge aller relevanten Daten.
 - Probleme
 - ◇ Inhalt der globalen Relation nicht fest
 - Anfrageergebnisse können sich ändern
 - ◇ Definition der Vollständigkeit der Ergebnisse
 - Welchen Anteil an der *world* hat das Ergebnis?
 - ◇ Negation in Anfragen

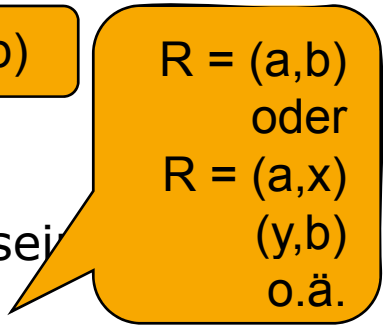
CWA / OWA – Beispiel

36

- Relation $R(A,B)$
- View 1
 - CREATE VIEW V1 AS
SELECT A FROM R
 - Extension: a
- View 2
 - CREATE VIEW V2 AS
SELECT B FROM R
 - Extension: b
- Anfrage: SELECT * FROM R
 - CWA: (a,b) muss in der Extension von R sein.
 - OWA: (a,b) muss nicht in der Extension von R sein.



$R = (a,b)$



$R = (a,b)$
oder
 $R = (a,x)$
 (y,b)
o.ä.

LaV – Anfragebearbeitung

37

- Gegeben: Anfrage Q und Sichten V_1, \dots, V_n
- Gesucht: Umgeschriebene Anfrage Q' , die
 - bei Optimierung: äquivalent ist ($Q = Q'$).
 - bei Integration: maximal enthalten ist.
 - ◇ D.h. $Q \supseteq Q'$ und
 - ◇ es existiert kein Q'' mit $Q \supseteq Q'' \supseteq Q'$ wobei $Q'' \neq Q'$.
- Problem:
 - Wie definiert und testet man Äquivalenz und *maximal containment*?

- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich



LaV – Anfrageumschreibungen

39

- Gegeben
 - Anfrage Q (query)
 - Sicht V (view)
- Fragen
 - Ist Ergebnis von V identisch dem Ergebnis von Q ?
 - Kurz: Ist V äquivalent zu Q , $V = Q$?
- Rückführung auf „Enthalten sein“ (containment)
 - Ist das Ergebnis von V in Q enthalten?
 - Kurz: Ist V in Q enthalten, $V \subseteq Q$?
- Denn
 - $V \subseteq Q, Q \subseteq V \Rightarrow V = Q$

LaV – Anfrageumschreibungen

40

Query containment (Anfrage-“Enthaltensein“)

- Sei S ein Schema. Seien Q und Q' Anfragen gegen S .
- Eine Instanz von S ist eine beliebige Datenbank D mit Schema S .
- Das Ergebnis einer Anfrage Q gegen S auf einer Datenbank D , geschrieben $Q(D)$, ist die Menge aller Tupel, die die Ausführung von Q in D ergibt.
- Q' ist contained (enthalten) in Q , geschrieben $Q' \subseteq Q$, gdw. $Q'(D) \subseteq Q(D)$ für jedes mögliche D .

LaV – Beispiele

41

```
SELECT K.titel, L.prof, K.kurs_id, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    K.univ = „Humboldt“
AND    L.univ = „Humboldt“
AND    K.titel LIKE „%VL_%“
```

\subseteq

```
SELECT K.titel, L.prof, K.kurs_id, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    K.univ = „Humboldt“
AND    L.univ = „Humboldt“
```

LaV – Beispiele

42

```
SELECT K.titel, K.kurs_id
FROM   Kurs K
AND    K.univ = „Humboldt“
AND    K.titel LIKE „%VL_%“
```

```
SELECT K.titel, K.kurs_id
FROM   Kurs K
AND    K.univ = „Humboldt“
```

≠

≠

```
SELECT K.titel, K.univ
FROM   Kurs K
AND    K.univ = „Humboldt“
AND    K.titel LIKE „%VL_%“
```

```
SELECT K.titel
FROM   Kurs K
AND    K.univ = „Humboldt“
```

LaV – Beispiele

43

Quelle 3:

```
CREATE VIEW Hum-Kurse AS
SELECT K.titel, K.univ
FROM   Lehrt L, Kurs K
WHERE  L.kurs_id = K.kurs_id
AND    K.univ = „Humboldt“
AND    L.univ = „Humboldt“
```

\subseteq

Quelle 5:

```
CREATE VIEW Kurse2 AS
SELECT K.titel, K.univ
FROM   Kurs K
AND    K.univ = „Humboldt“
```

Prüfung von containment durch Prüfung aller möglichen Datenbanken?

- Zu komplex!

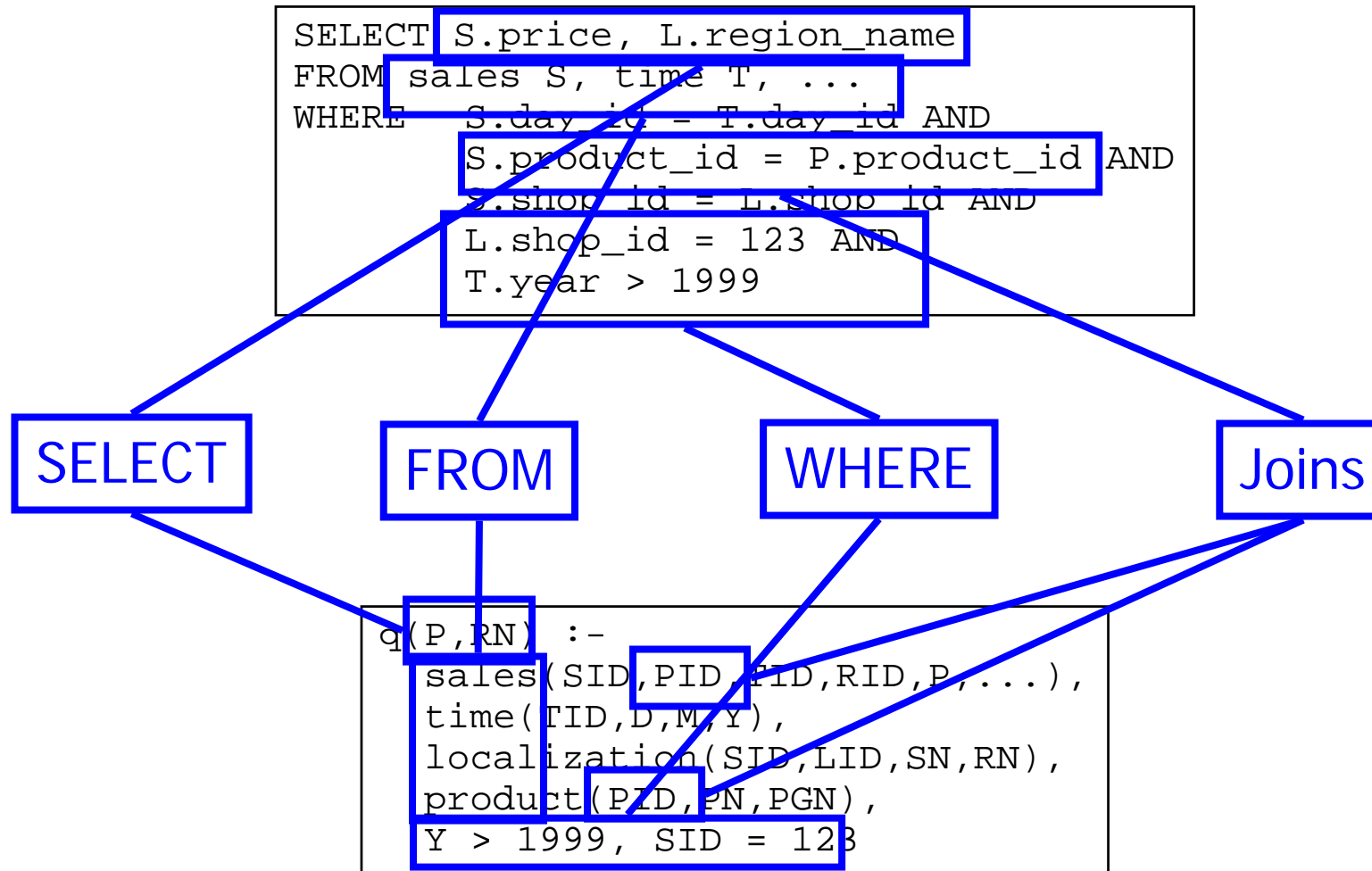
Prüfung von containment durch Existenz eines containment mapping.

- NP-vollständig in $|Q|+|Q'|$ nach [CM77]
- Mehrere Algorithmen

Datalog Notation

44

- Im Folgenden: Nur Konjunktive Anfragen
 - Nur Equijoins und Bedingungen mit $=, <, >$ zw. Attribut und Konstanten
 - Kein NOT, EXISTS, GROUP BY, \neq , $X > Y$, ...
- Schreibweise: Datalog / Prolog
 - SELECT Klausel
 - ◇ Regelkopf, Exportierte Attribute
 - FROM Klausel
 - ◇ Relationen werden zu Prädikaten
 - WHERE Klausel
 - ◇ Joins werden durch gleiche Attributnamen angezeigt
 - ◇ Bedingungen werden explizit angegeben



Notation

46

Definition 2.2

Sei V eine Menge von Variablensymbolen und C eine Menge von Konstanten. Eine *konjunktive Datalog-Anfrage* q ist eine Anfrage der Form:

$$q(v_1, v_2, \dots, v_n) \quad :- \quad r_1(w_{1,1}, \dots, w_{1,n_1}), r_2(w_{2,1}, \dots, w_{2,n_2}), \dots, \\ r_m(w_{m,1}, \dots, w_{m,n_m}), k_1, \dots, k_l;$$

mit extensionalen Prädikaten r_1, r_2, \dots, r_m , $v_i \in V$, $w_{i,j} \in V \cup C$ und $\forall v \in V : \exists i, j : w_{i,j} = v$ und $\forall c \in C : \exists i, j : w_{i,j} = c$. Alle k_i haben für beliebige $v_1, v_2 \in V$ und $c \in C$ die Form $v_1 < c$, $v_1 > c$, $v_1 = c$ oder $v_1 = v_2$. Dann ist:

- $head(q) = q(v_1, v_2, \dots, v_n)$ der **Kopf** von q ,
- $body(q) = r_1(w_{1,1}, \dots), r_2(w_{2,1}, \dots), \dots, r_m(w_{m,1}, \dots)$ der **Rumpf** von q ,
- $exp(q) = \{v_1, v_2, \dots, v_n\}$ die Menge der *exportierten Variablen* von q ,
- $var(q) = V$ die Menge aller Variablen von q ,
- $const(q) = C$ die Menge aller Konstanten von q ,
- $sym(q) = C \cup V$ die Menge aller Symbole von q ,
- r_1, r_2, \dots, r_m sind die *Literale* von q , und
- $cond(q) = k_1, \dots, k_l$ sind die Bedingungen von q . ■

Query Containment

47

- A query p is contained in a query u , $p \subseteq u$, iff all tuples computed by p are also computed by u for every DB.
- Beispiele (Regelkopf weggelassen)
 - $\text{map}(M_n, M_s) \subseteq \text{map}(M_n, M_s)$;
 - $\text{map}(M_n, M_s), M_n = \text{'HGM'} \subseteq \text{map}(M_n, M_s)$;
 - $\text{map}(M_n, M_s), M_s < 500 \subseteq \text{map}(M_n, M_s)$;
 - $\text{map}(M_n, M_s), \text{clone}(M_n, C_n, -) \subseteq \text{map}(M_n, M_s)$;
 - $\text{clone}(M_n, C_n, C_s), \text{clone}(M_n, C_n, C_s) \subseteq \text{clone}(M_n, C_n, C_s)$;

Beweis für Query Containment

48

- $p \subseteq u$ gdw ein containment mapping von u nach p existiert.
- Containment mapping:
 - $h: \text{sym}(u) \rightarrow \text{sym}(p)$ (Abbildung der Symbole)
 - CM1: Jede Konstante in u wird auf die gleiche Konstante in p abgebildet.
 - CM2: Jede exportierte Variable in u wird auf eine exportierte Variable in p abgebildet.
 - CM3: Jedes Literal (Relation) in u wird auf mindestens ein Literal in p abgebildet
 - CM4: Die Bedingungen von p implizieren die Bedingungen von u
- Beweis: [CM77]

map(Mn, Ms), clone(Mn, Cn, -, -) \subseteq map(Mn, Ms);

Finden von Containment Mappings

49

- Problem ist NP vollständig
 - Exponentiell in der Anzahl der Literale
 - Beweis: Reduktion auf „Exakt Cover“
- Also: Alles ausprobieren
 - Aufbau eines Suchbaums
 - ◇ Jede Ebene entspricht einem Literal
 - ◇ Auffächerung nach möglichen CMs
- Algorithmus
 - Nicht hier
 - Siehe Buch

Weitere Containment Beispiele

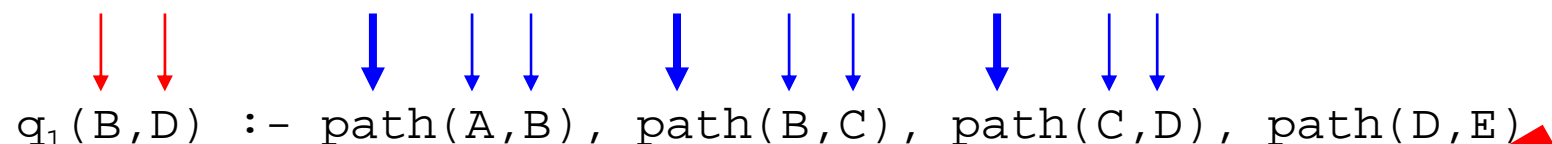
50

<code>product (PID, PN, PGID, , Wasser `)</code>	\subseteq	<code>product (PID, PN, PGID, PGN)</code>
<code>product (PID, PN, PGID, PGN)</code>	$\not\subseteq$	<code>localization (SID, SN, RID, RN)</code>
<code>product (PID, PN, PGID, PGN), PGN>, Wasser `</code>	\subseteq	<code>product (PID, PN, PGID, PGN)</code>
<code>sales (SID, PID, ..., P, ...), P>80, P<150</code>	$\not\subseteq$	<code>sales (SID, PID, ..., P, ...), P>100, P<150</code>
<code>sales (SID, PID, ..., P, ...), P>100, P<=150, P<170, P>=150</code>	\Leftrightarrow	<code>sales (SID, PID, ..., 150, ...)</code>
<code>sales (SID, PID, ..., P, ...), product (PID, PN, ...)</code>	\subseteq	<code>sales (SID, PID, ..., P, ...)</code> <i>(Bei Projektion auf sales)</i>

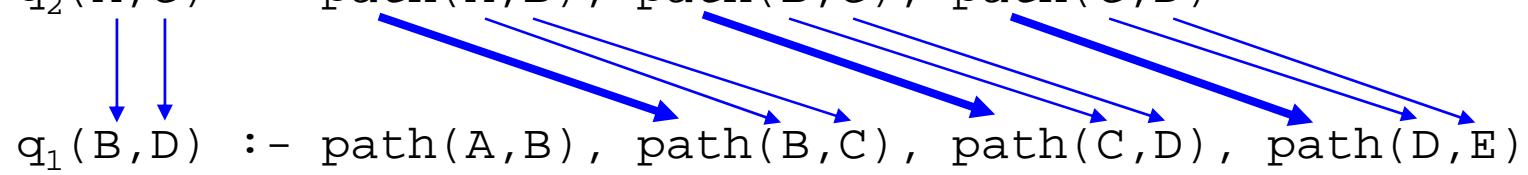
Beispiel

51

$q_1 \subseteq q_2 ?$

$q_2(A,C) :- \text{path}(A,B), \text{path}(B,C), \text{path}(C,D)$

 $q_1(B,D) :- \text{path}(A,B), \text{path}(B,C), \text{path}(C,D), \text{path}(D,E)$

Mapping: $A \rightarrow A, B \rightarrow B, C \rightarrow C, D \rightarrow D$

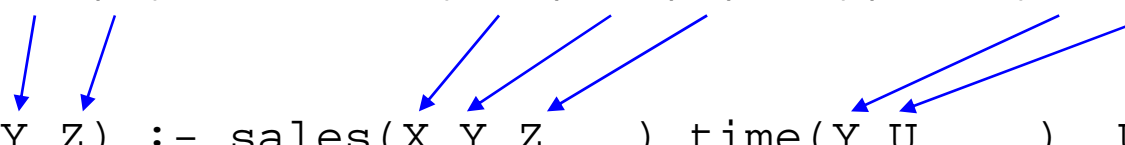
$q_2(A,C) :- \text{path}(A,B), \text{path}(B,C), \text{path}(C,D)$

 $q_1(B,D) :- \text{path}(A,B), \text{path}(B,C), \text{path}(C,D), \text{path}(D,E)$

Mapping: $A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow E$

Beispiel

52

$q_2(TID, P) \text{ :- sales}(SID, TID, P, \dots), \text{time}(TID, D, \dots), D > 28, D < 31$
 $q_1(Y, Z) \text{ :- sales}(X, Y, Z, \dots), \text{time}(Y, U, \dots), U > 1, U < 30$



CM: $SID \rightarrow X, TID \rightarrow Y, P \rightarrow Z, D \rightarrow U$
 $h(D) = U$

Aber: $U > 1 \wedge U < 30 \not\rightarrow h(D) > 28 \wedge h(D) < 31$

$q_1 \not\subseteq q_2$

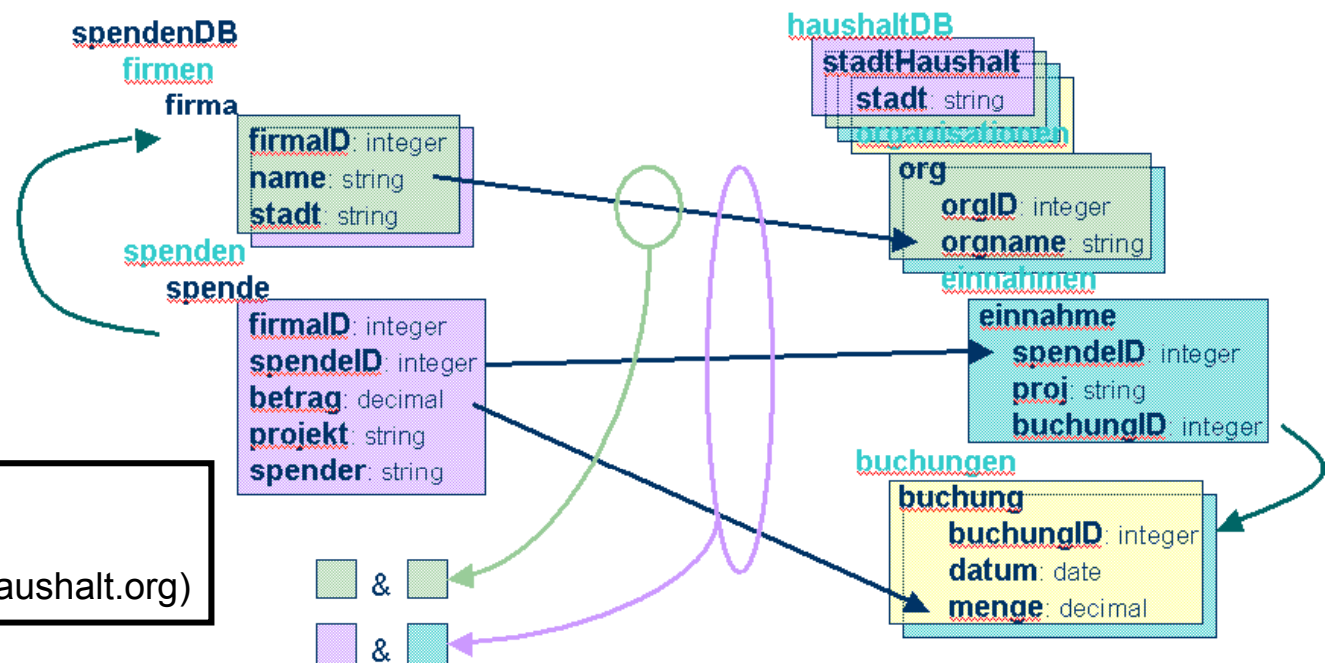
Erzeugung der Anfragen

53

Beobachtung:
Interpretation als
containment

$\pi_{\text{name}}(\text{spendenDB.firmen})$
 \subseteq
 $\pi_{\text{orgname}}(\text{haushaltDB.stadtHaushalt.org})$

$\pi_{\text{name,spendeID,betrag}}(\text{spendenDB.firmen} \triangleright \triangleleft \text{spendenDB.spenden})$
 \subseteq
 $\pi_{\text{orgname,spendeID,menge}}(\text{haushaltDB.stadtHaushalt.org.einnahmen} \triangleright \triangleleft \text{haushaltDB.stadtHaushalt.buchungen})$



- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich



Global-Local-as-View (GLAV)

55

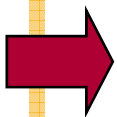
- Kombination GaV und LaV
 - GaV:
 - ◇ Globale Relation = Sicht auf lokale Relationen
 - ◇ (Globale Relation \supseteq Sicht auf lokale Relationen)
 - LaV:
 - ◇ Sicht auf globale Relationen = lokale Relation
 - ◇ Sicht auf globale Relationen \supseteq lokale Relation
 - GLaV:
 - ◇ Sicht auf globale Relationen = Sicht auf lokale Relationen
 - ◇ Sicht auf globale Relationen \supseteq Sicht auf lokale Relationen
- Auch „BaV“: Both-as-View

Anfrageplanung

56

- GaV:
 - View unfolding
 - Standardtechniken aus relationalen Datenbanken
- LaV
 - Query Containment und Answering queries using views
 - Mehrere Algorithmen, auch andere Anwendungen (später)
 - Schwierige Planung
- GLAV
 - Erst Anfrageplanung mit den Sichten auf das globale Schema
 - ◇ Die linken Seiten von Korrespondenzen
 - Dann Unfolding mit den Sichten auf die lokalen Schemata
 - ◇ Die rechten Seiten von Korrespondenzen
- Dann verteilte Optimierung

- Motivation
- Korrespondenzen
- Übersicht Anfrageplanung
- Global as View (GaV)
- Local as View (LaV)
 - Modellierung
 - Anwendungen
 - Anfragebearbeitung
 - Containment
- Global Local as View (GLaV)
- Vergleich



Vergleich GaV / LaV

58

Modellierung

■ GaV

- Jede globale Relation definiert als Sicht auf eine oder mehr Relationen aus einer oder mehr Quellen.
- Meist UNION über mehrere Quellen
- Nebenbedingungen auf lokalen Quellen können nicht modelliert werden.

■ LaV

- Globale Relationen werden durch mehrere Sichten definiert.
- Definition oft nur in Kombination mit anderen globalen Relationen
- Nebenbedingungen auf globale Relationen können nicht definiert werden.

■ GLAV

- Globale und lokale Nebenbedingungen sind möglich
- Konzepte können lokal oder global eingeschränkt werden

Vergleich GaV / LaV

59

- Anfragebearbeitung
 - GaV:
 - ◇ Anfrageumschreibung: Einfaches View unfolding
 - ◇ Eine große, umgeschriebene Anfrage
 - ◇ Interessante Optimierungsprobleme
 - LaV
 - ◇ Anfrageumschreibung: Answering queries using views
 - ◇ UNION über viele mögliche umgeschriebene Anfragen
 - ◇ Mehrere Algorithmen
 - ◇ Viele Anwendungen
 - ◇ Noch interessantere Optimierungsprobleme
- Flexibilität
 - GaV: Views setzen Relationen mehrerer Quellen in Beziehung
 - LaV: Jede View bezieht sich nur auf eine Quelle

Gute Zusammenfassung für LaV und weiterführende Literatur:

- [Levy01] Alon Y. Halevy: *Answering queries using views: A survey*, in VLDB Journal 10: 270-294, 2001.

Eher theoretisch

- [Ull00] Jeffrey D. Ullman: *Information Integration Using Logical Views*. TCS 2000: 189-210
- [Hull97] Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. Richard Hull. PODS 1997 tutorial
- [Ull97] Jeffrey D. Ullman: Information Integration Using Logical Views. [ICDT 1997](#): 19-40
- [CM77] [Ashok K. Chandra](#) and [Philip M. Merlin](#). Optimal implementation of conjunctive queries in relational data bases. In Conference Record of the Ninth Annual ACM Symposium on Theory of Computing, pages 77-90, Boulder, Colorado, 2-4 May 1977.
- [LMSS95] Alon Y. Levy, [Alberto O. Mendelzon](#), [Yehoshua Sagiv](#), [Divesh Srivastava](#): Answering Queries Using Views. [PODS 1995](#): 95-104