



Informationsintegration

Local-as-View Anfrageplanung

Ulf Leser

Inhalt dieser Vorlesung

- Anfragekorrespondenzen – Formale Definition
 - Korrespondenztypen und -arten
 - GaV erklärt
- Local as View
- Query Containment
- Semantische Korrektheit von Anfrageplänen
- Später
 - Algorithmen zum Query Containment
 - Answering Queries using Views
 - GLaV – Global and Local as View

Anfrageplan revisited

- 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 **mit Hilfe der Quellen** ausgeführt werden*
- *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

Anfrageergebnis

- Definition

*Gegeben eine globale Anfrage q . Sei p_1, \dots, p_n die Menge aller (semantisch korrekter) Anfragepläne für q . Dann ist das **Ergebnis von q** definiert als*

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

- Bemerkungen

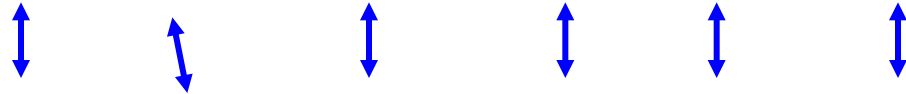
- In der Informationsintegration sollte UNION **Duplikate** entfernen
 - Problem der Ergebnisintegration
- Wie das Ergebnis berechnet wird, ist Sache der Anfrageoptimierung
 - Identische / überlappende Teilanfragen erkennen oder nicht
 - Ergebnisse cachen oder nicht
 - ...

Anfragekorrespondenzen

- Anfrageplanung **übersetzt** zwischen verschiedenen „Sprachen“ im selben Datenmodell
 - Zielschema und Quellschemata
- Beziehungen zwischen den Sprachelementen sind Grundlage der semantischen Korrektheit von Plänen
 - Beziehungen: Homonyme, Synonyme, Hyperonyme, etc.
 - Bei echten Sprachen zwischen: Wörter, Sätze, Texte
 - Bei Schemata: **Attribute, Relationen, Anfragen**
- Beziehungen werden durch **Korrespondenzen** ausgedrückt

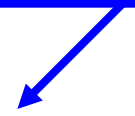
Übersetzung

- The dog barked and went away



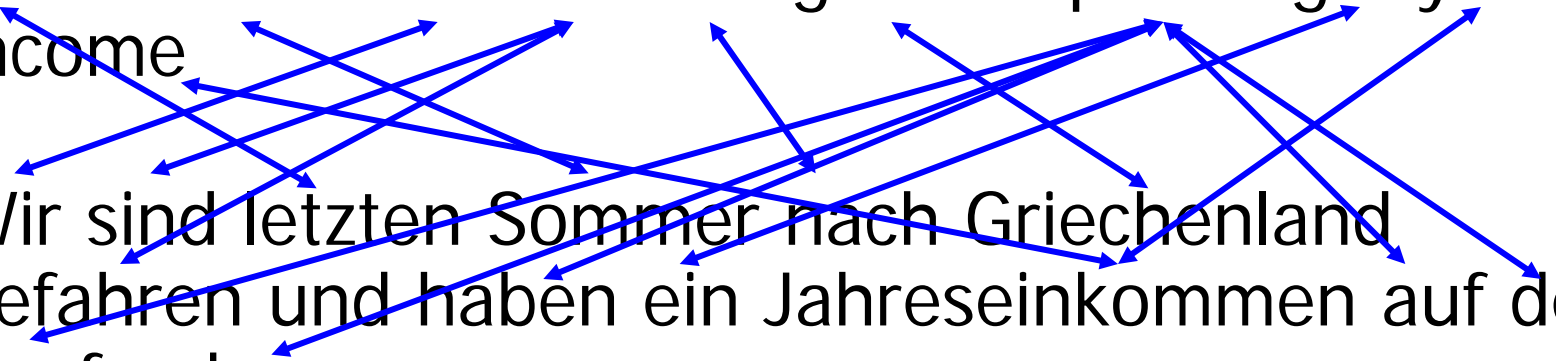
- Der Hund bellte und ging weg

Strukturelle und
semantische
Heterogenität



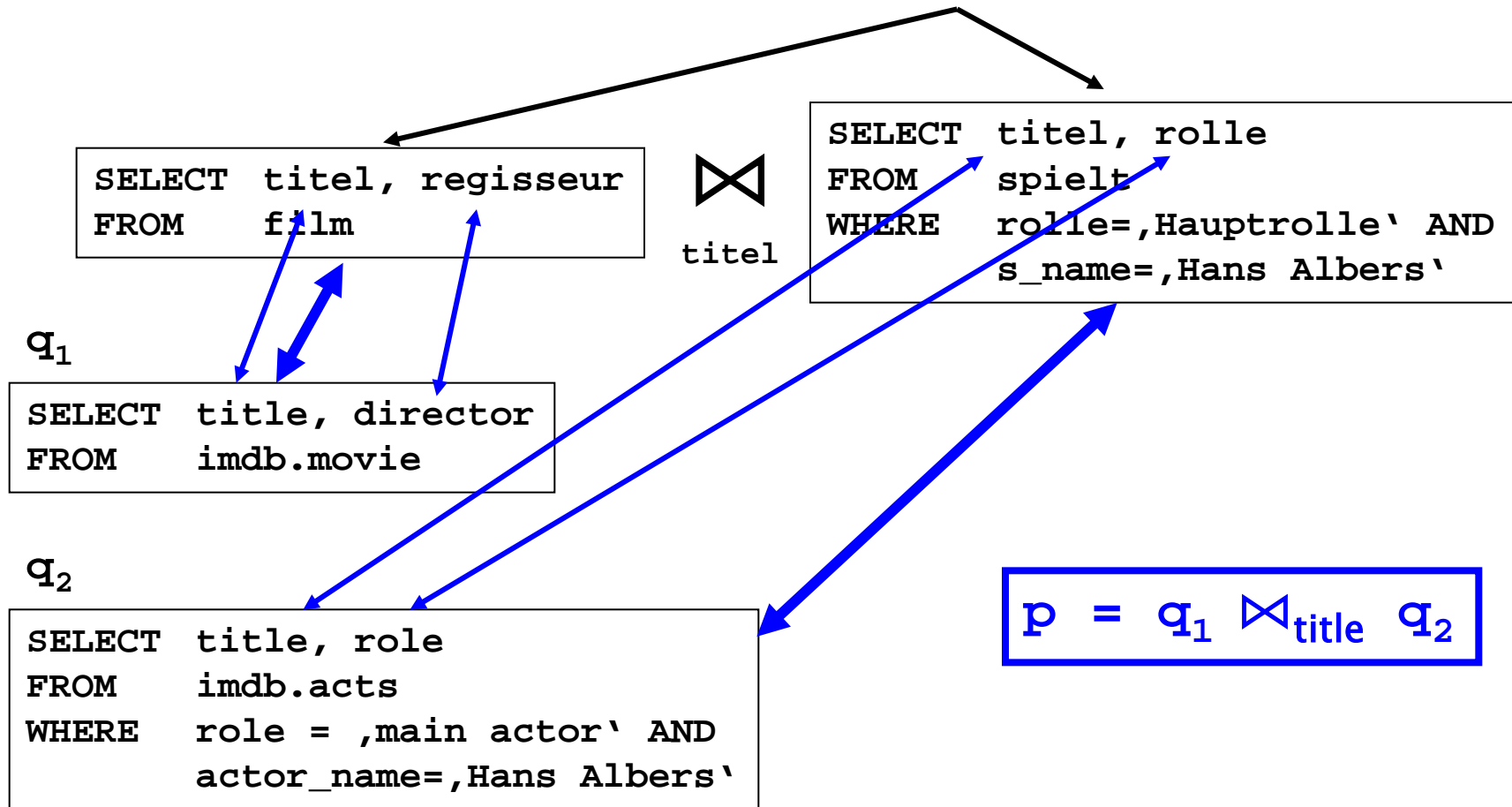
- Last summer we went to greece spending a year's income

- Wir sind letzten Sommer nach Griechenland gefahren und haben ein Jahreseinkommen auf den Kopf gehauen



Auf Schemaebene

```
SELECT titel, regisseur, rolle
FROM   film, spielt
WHERE  spielt.schauspieler_name = 'Hans Albers'
      AND spielt.rolle = 'Hauptrolle'
      AND spielt.titel = film.titel;
```



Viele Fragen

- Welche Attribute entsprechen welchen Attributen?
- Wie bricht man die globale Anfrage am besten auf?
- Für welche Teile der globalen Anfrage gibt es äquivalente Anfragen an Wrapper?
- Wie drückt man diese Korrespondenzen aus?
- Wie kann man dabei mit beschränkten Quellen umgehen?
- ...

Inhalt dieser Vorlesung

- Anfragekorrespondenzen
 - Korrespondenztypen und -arten
 - GaV erklärt
- Local as View
- Query Containment
- Semantische Korrektheit von Anfrageplänen

Drei Korrespondenzarten

- Beziehungen zwischen **Attributen**
 - Eher einfach zu finden
 - Reicht nicht zur Anfrageplanung
- Beziehungen zwischen **Relationen**
 - „Natürliche“ Ebene – Anfrageplanung wird zur Ersetzung von Relationennamen
 - Aber Relationen eines Quellschemas entsprechen meist nicht 1:1 den Relationen eines anderen Schemas
- Beziehungen zwischen **Anfragen**
 - Subsumiert beide vorherigen Konzepte, sehr flexibel
 - Schwierigere Anfrageplanung

Korrespondenztypen 1

- Sei q_1 ein Element des **globalen Schemas** und q_2 ein Element eines **lokalen Schemas**
 - Element = Anfrage/Relation/Attribut
- Exklusion: $\mathbf{q_1 \cap q_2 = \emptyset}$
 - Extensionen von q_1 und q_2 sind überlappungsfrei
 - Bedingt auch intensionale Überlappungsfreiheit
 - Das ist der **Normalfall** und wird angenommen, wenn keine Korrespondenz zwischen zwei Elementen angegeben wird

Korrespondenztypen 2

- Sei q_1 ein Element des globalen Schemas und q_2 ein Element eines lokalen Schemas
- Inklusion: $q_1 \supseteq q_2$
 - Extension von q_2 ist in der von q_1 enthalten
 - Mediator-Architekturen: Extension der Wrapperanfragen (q_2) ist in der Extension der globalen Anfrage (q_1) enthalten
 - Besser: „soll enthalten sein“
 - Beachte: q_1 kann in vielen Korrespondenzen vorkommen
 - Wenn es viele Quellen gibt, die zu q_1 beitragen
 - Wenn es viele Elemente in einer Quellen gibt, die zu q_1 beitragen

Korrespondenztypen 2

- Äquivalenz: $\mathbf{q_1 \equiv q_2}$
 - Die Extensionen von q_1 und q_2 sind identisch
 - Interpretation: q_2 ist die **einzig mögliche Datenquelle** von q_1
 - Es darf also keine weitere Regel der Art $q_i \supseteq q_2$ oder $q_i \equiv q_2$ geben
 - Betrachten wir im folgenden nicht weiter
- Überlappung: $\mathbf{q_1 \cap q_2 \neq \emptyset \wedge \neg(q_1 \subseteq q_2) \wedge \neg(q_1 \supseteq q_2)}$
 - Die Extensionen von q_1 und q_2 überschneiden sich, ohne das eine die andere enthält
 - Damit sind manche Ergebnisse von q_2 nicht in der Extension von q_1 enthalten – **falsche Ergebnisse**
 - Diese Korrespondenzen nützen uns nichts
 - Vorgehen: Finde eine Einschränkung **filter** von q_2 so, dass $\mathbf{q_1 \supseteq filter(q_2)}$

Korrespondenzen

- Definition

Eine (Anfrage-)korrespondenz ist eine Regel der Art

$$q_1 \supseteq q_2$$

- q_1 ist eine Anfrage an das globale Schema
- q_2 ist eine ausführbare Anfrage an das Exportschema eines Wrappers

- Bemerkung

- Wir verwenden ab jetzt nur noch \supseteq - Korrespondenzen
- Aussage über die Extensionen von Anfragen
- Also über die Beziehungen von Tupelmengen, die durch Anfragen in unterschiedlichen Schemata berechnet werden
- Ist also eine Aussage über die (formale) Semantik von Anfragen

GaV und LaV

- Definition

Eine Korrespondenz $q_1 \supseteq q_2$ heißt

- *GaV ([Global-as-View](#)), wenn q_1 (globales Schema) eine einzelne Relation ohne Selektionen oder Joins ist*
- *LaV ([Local-as-View](#)), wenn q_2 (Wrapperschema) eine einzelne Relation ohne Selektionen oder Joins ist*
- *Sonst heißt sie GLaV ([Global-local-as-view](#))*

Inhalt dieser Vorlesung

- Anfragekorrespondenzen
 - Korrespondenztypen und -arten
 - GaV erklärt
- Local as View
- Query Containment
- Semantische Korrektheit von Anfrageplänen

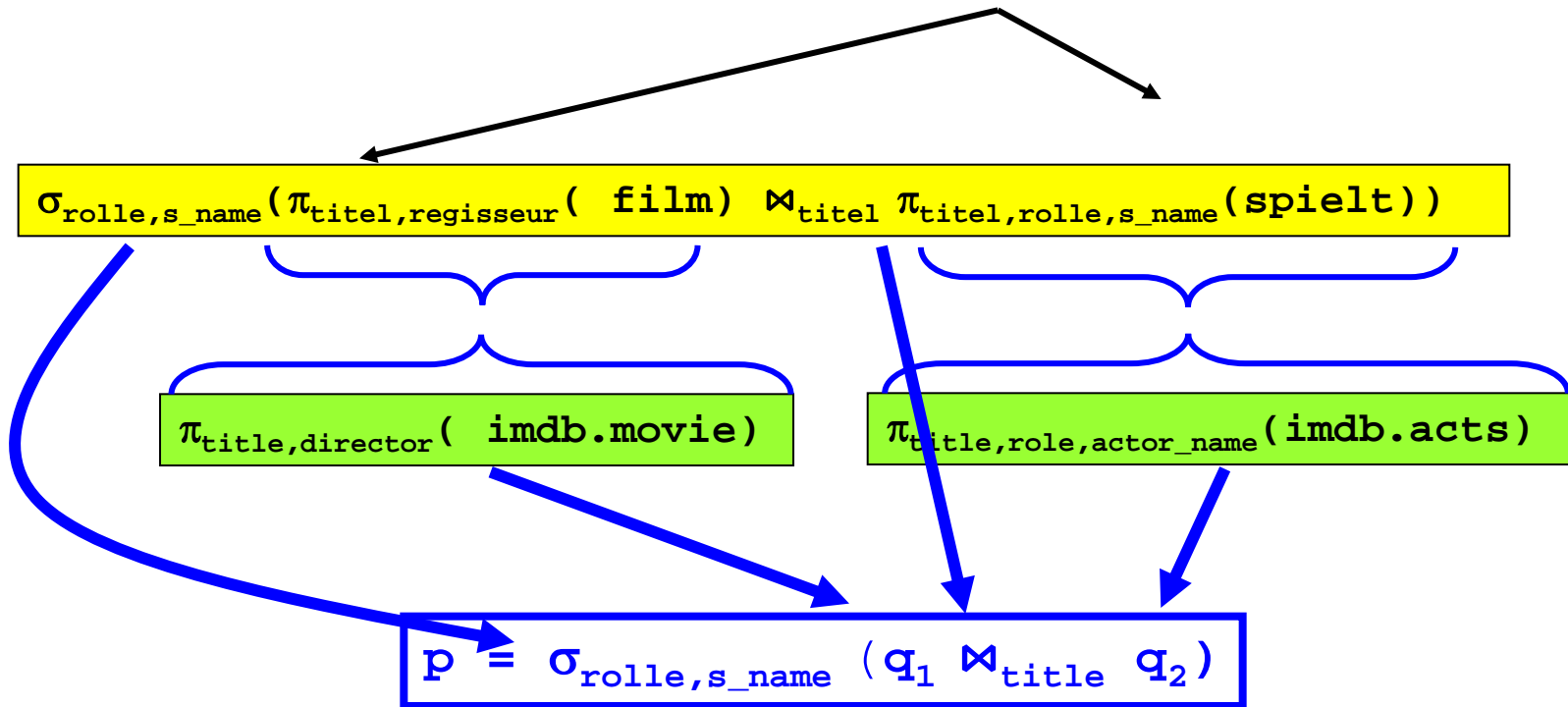
Global-as-View erklärt

- Eine **GaV-Korrespondenz** besteht also aus
 - Einer Relation q_1 des globalen Schemas
 - Query q_1 mit Projektion, aber ohne Selektionen oder Joins
 - Einer ausführbaren Anfrage q_2 an ein Quellschema
 - q_2 hat dasselbe Schema wie q_1
- Beispiel
 - $\pi_{\text{titel,regisseur}}(\text{film}) \supseteq \pi_{\text{title,director}}(\text{imdb.movie})$
 - $\pi_{\text{titel,rolle,s_name}}(\text{spielt}) \supseteq \pi_{\text{title,role,actor_name}}(\text{imdb.acts})$

```
SELECT titel, regisseur, rolle
FROM   film, spielt
WHERE  spielt.schauspieler_name = 'Hans Albers'
      AND spielt.rolle = 'Hauptrolle'
      AND spielt.titel = film.titel;
```

GaV Planung

```
SELECT titel, regisseur, rolle
FROM   film, spielt
WHERE  spielt.schauspieler_name = 'Hans Albers'
      AND spielt.rolle = 'Hauptrolle'
      AND spielt.titel = film.titel;
```



Spezifischere Korrespondenzen

- Man könnte auch andere Regeln spezifizieren
 - $q_1: \pi_{\text{titel,regisseur}}(\text{film}) \supseteq \pi_{\text{title,director}}(\text{imdb.movie})$
 - $q_2: \pi_{\text{titel,rolle,s_name}}(\text{spielt}) \supseteq \sigma_{\text{role=„HR“,actor_name=„HA“}}(\pi_{\text{title,role,actor_name}}(\text{imdb.acts}))$
- Damit
 - $q: \pi_{\text{titel,regisseur}}(\text{film}) \bowtie_{\text{titel}} \sigma_{\text{rolle,s_name}}(\pi_{\text{titel,rolle,s_name}}(\text{spielt}))$
 - $q: q_1 \bowtie q_2$
- Vorteil: Selektionen werden implizit (durch Regeldefinition) gepushed
- Nachteil: Regel ist gut genau für q – für andere Anfragen schlecht
 - Wir brauchen **exponentiell mehr Regeln**
 - Deklarativer Charakter von Korrespondenzen geht verloren
- Ziel: Generische Korrespondenzen, den Rest macht die Anfrageplanung

Inhalt dieser Vorlesung

- Anfragekorrespondenzen
- Local as View
 - Notation: Datalog
 - Local-as-View Korrespondenzen
- Query Containment
- Semantische Korrektheit von Anfrageplänen

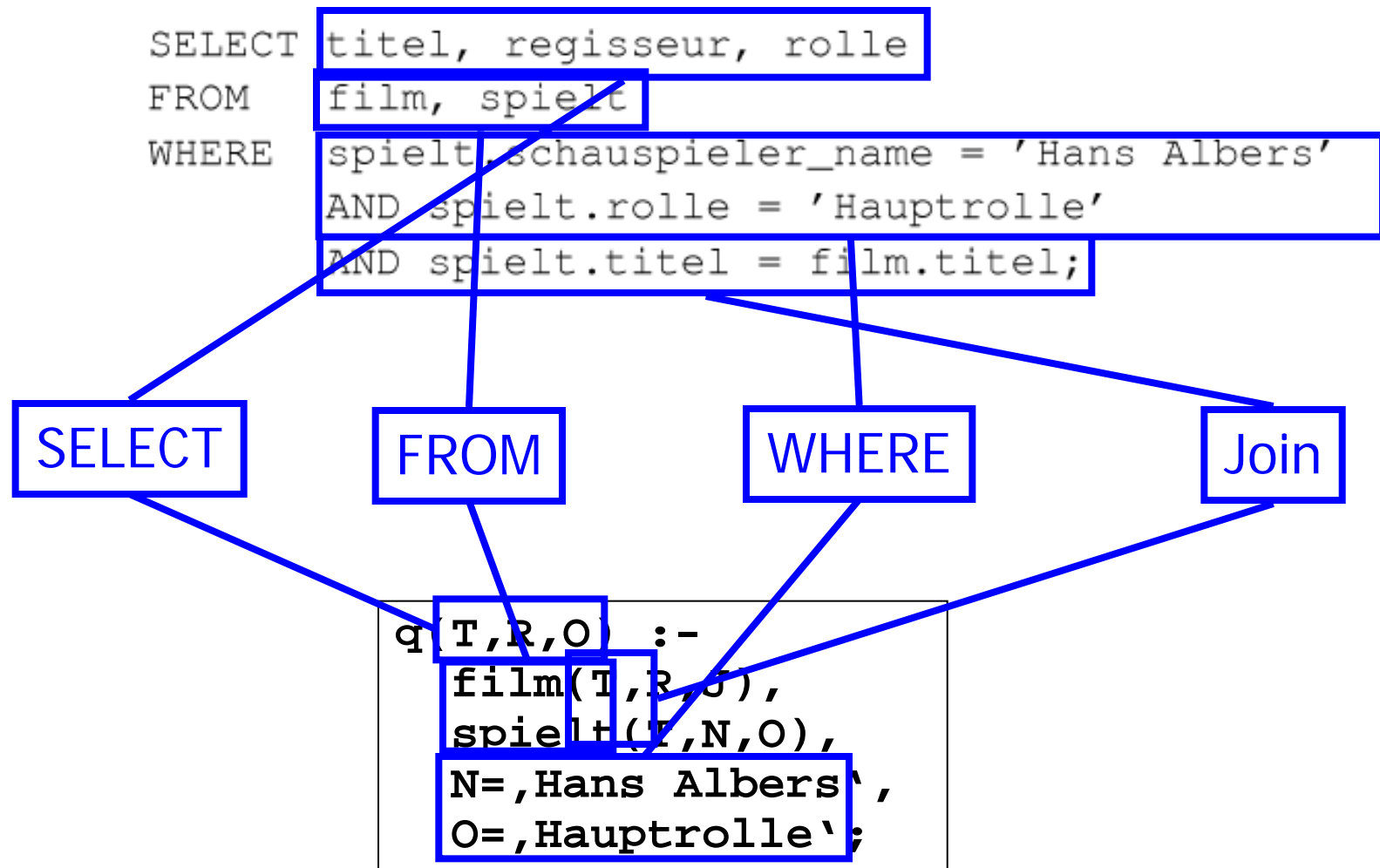
Konjunktive Anfragen

- Wir betrachten im Folgenden nur **konjunktive Anfragen**
 - Equi-joins und Bedingungen mit $=, <, >$ zwischen Attribut und Wert
 - Kein NOT, EXISTS, GROUP BY, \neq , $X > Y$, ...
- LaV Planung idR schwieriger für andere Anfrageklassen
 - Gegenstand vieler Forschungsprojekte

Kürzere Schreibweise

- Schreibweise: **Datalog** (Prolog)
 - `q(T,R,O) :- film(T,R,J),spielt(T,N,O),N=,Hans Albers`;`
- SELECT Klausel: Regelkopf, „**Exported Variables**“
 - Auch: „Distinguished variables“ – von außen unterscheidbar
- FROM Klausel: Prädikate stehen für Relationen
 - Attribute werden über **Position statt Name** adressiert
- WHERE Klausel
 - Joins: dieselbe Variable an mehreren Stellen
 - Bedingungen mit „>“, „<“, werden explizit angegeben
 - Gleichheitsbedingungen „Attribut = Wert“ werden durch Konstante im Literal angegeben oder explizit

SQL – Datalog



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 . ■

Beispiel

```
q(T,R,O) :-  
    film(T,R,_),  
    spielt(T,N,O),  
    N=,Hans Albers`,  
    O=,Hauptrolle`;
```

- `film`, `spielt`, .. sind **Prädikate**
 - Relationen des Schemas
- `film(T,R,J)`, `spielt(T,N,o)` sind **Literale**
 - Eine Anfrage kann mehrere Literale desselben Prädikats enthalten
- Variable, die nicht interessieren, kürzt man mit „_“ ab
 - Kein Join, keine Bedingung, nicht exportiert
- `q` ist **sicher**, wenn jede exportierte Variable im Rumpf vorkommt

Kein echtes Datalog

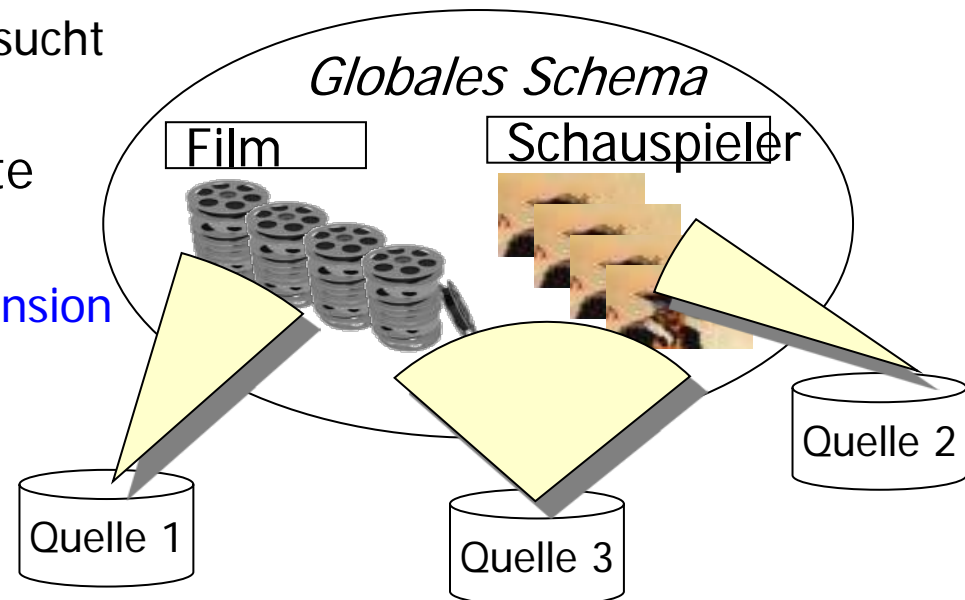
- Keine Disjunktion und Vereinigung
- Keine Joins außer Equi-Joins
- Keine rekursiven Anfragen
 - **Extensional predicates**: Prädikate, deren Extension in der Datenbank vorliegen
 - **Intensional predicates**: Prädikate, die zur Laufzeit berechnet werden
 - SQL: Views
 - Verwendet ein intensionales Prädikat sich selber im Rumpf, wird dadurch eine **rekursive Anfrage definiert**
 - „Normales“ SQL: Verboten
 - Rekursives SQL: Views mit Namen

Inhalt dieser Vorlesung

- Anfragekorrespondenzen
- Local as View
 - Notation: Datalog
 - Local-as-View Korrespondenzen
- Query Containment
- Semantische Korrektheit von Anfrageplänen

Warum Local-as-View?

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



LaV – Beispiel

```
Film(Titel, Regie, Jahr, Genre)
Programm(Kino, Titel, Zeit)
```

```
S1: IMDB(Titel, Regie, Jahr, Genre)
S2: MyMovies(Titel, Genre, Regie, Jahr)
S3: RegieDB(Titel, Regie)
S4: GenreDB(Titel, Jahr, Genre)
```

```
CREATE VIEW S1 AS
SELECT *
FROM Film

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

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

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

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

Uninteressante lokale Attribute

```
Film(Titel, Regie, Jahr, Genre)  
Programm(Kino, Titel, Zeit)
```

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

```
S9: ActorDB(Titel, Schauspieler, Jahr)
```

- `schauspieler` gibt es im globalen Schema nicht
- Wird im View projiziert

Assoziationen im globalen Schema

```
Film(Titel, Regie, Jahr, Genre)  
Programm(Kino, Titel, Zeit)
```

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

```
S7: KinoDB(Kino, Genre)
```

- Assoziationen des globalen Schemas können in der Sicht dargestellt werden – auch ohne Werte für das Join-Attribut

Assoziationen im lokalen Schema

```
Film(Titel, Regie, Jahr, Genre)  
Programm(Kino, Titel, Zeit)
```

```
CREATE VIEW S9.Filme AS  
SELECT  Titel, Jahr  
FROM    Film  
  
CREATE VIEW S9.Regie AS  
SELECT  Regie  
FROM    Film
```

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

- Assoziationen des lokalen Schemas können **nicht abgebildet werden**, wenn Join-Attribute im globalen Schema nicht vorhanden sind oder von der Quelle nicht exportiert werden

Lokale Integritätsconstraints

```
Film(Titel, Regie, Jahr, Genre)  
Programm(Kino, Titel, Zeit)
```

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

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

- **Integritätsconstraint** auf der Quelle kann modelliert werden, wenn das Attribut im globalen Schema existiert
- IC müssen in der Quelle nicht explizit definiert sein
 - Auch implizite Einschränkungen können in den View

Globale Integritätsconstraints

```
Film(Titel, Regie, Jahr, Genre)  
Programm(Kino, Titel, Zeit)  
Jahr > 2000
```

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

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

- ICs auf dem globalen Schema können nicht modelliert werden
 - Das ging aber bei GaV

Komplexere Beispiele

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

```

Anfragebearbeitung - Probleme

- Alle Filme kürzer als 100 Minuten
 - `spielfilme (filtern) \cup kurzfilme (vollständig) \cup us_spielfilm (filtern)`
- Alle Filme, die länger als 60 Minuten sind
 - `spielfilme (filtern) \cup us_spielfilm (filtern)`
- Besetzungen von Hauptrollen in Filmen unter 100 Minuten
 - `spielfilme \bowtie filmkritiken ?`
 - `spielfilme \bowtie spielfilm_kritiken ?`
 - `spielfilme \bowtie us_spielfilm ?`
 - `spielfilme \bowtie kurzfilm_rollen ?`

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

Anders formuliert

- Globale Anfrage

```
q :-  
    film(T,_,_,L),spielt(T,S,O,_),O=,Hauptrolle`,L<100;
```

- Plan 1

- `spielfilme` \bowtie `filmkritiken`

- Expandiert:

```
film(T,Y,R,L), L>79, Y=,Spielfilm`, film(T`,_,R`),  
spielt(T`,S`,O`,_), O`=,Hauptrolle`, T=T`
```

- Frage

- Erzeugt

```
film(T,`Spielfilm`,R,L), L>79,  
film(T,_,R`,_), spielt(T,S`,`Hauptrolle`,_);
```

nur **richtige Antworten** für q?

Anfrageplanung mit LaV

- Gegeben: Globales Schema S , Anfrage q an S , Menge von Sichten v_1, \dots, v_n auf S
- Gesucht: Alle **Kombinationen von v_i** , die q beantworten
 - Wir wissen aber nicht, wie wir q am besten zerlegen, so dass eine Teilquery immer einer Sichtdefinition entspricht
 - **Planung ist schwieriger** als bei GaV
- **„Answering queries using views“**
 - Anderes Bild: Eine DB und eine Menge materialisierter Sichten
 - Plötzlich: alle Relationen der DB weg, nur die Sichten sind noch da
 - Können wir eine **Anfrage q nur mit den Sichten** beantworten?

Inhalt dieser Vorlesung

- Anfragekorrespondenzen
- Local as View
- Query Containment
- Semantische Korrektheit von Anfrageplänen

Zwei Teilprobleme

- Welche **Kombinationen von Views** (also Quellen) liefern semantisch korrekte Antworten auf eine globale Anfrage
- Wann ist ein gegebener Plan **semantisch korrekt**?
 - Wir wissen nur, dass **bestimmte Teilanfragen** gegen das globale Schema extensional bestimmte Quellrelationen umfassen sollen
 - Wie kann man das auf **beliebige Anfragen** übertragen?
- **Query Containment**
 - Zunächst arbeiten wir nur mit einem View
 - Also: Wenn liefert ein View korrekte Ergebnisse für eine Query?

Query Containment

- Intuition: Ein View v liefert nur semantisch korrekte Anfragen auf eine globale Anfrage q , wenn die **Extension von v in der Extension von q enthalten ist**
- Definition
Sei S ein Datenbankschema, I eine Instanz von S und q_1, q_2 Anfragen gegen S . 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.
$$q_1(I) \subseteq q_2(I) \text{ für alle } I$$

Äquivalenz

- Definition

Sei S ein Datenbankschema, I eine Instanz von S und q_1, q_2 Anfragen gegen S . Sei $q(I)$ das Ergebnis einer Anfrage q angewandt auf I . Dann ist

*q_1 äquivalent zu q_2 , geschrieben $q_1 \equiv q_2$
gdw.*

$q_1(I) \subseteq q_2(I)$ und $q_1(I) \supseteq q_2(I)$ für alle I

- Bemerkung

- Wir beschäftigen uns nur mit enthaltenen Anfragen

Einfache Beispiele (Alle Variable seien exportiert)

$\text{film}(T, Y, R, L) \subseteq \text{film}(T, Y, R, L)$

$\text{film}(T, \text{'Dokumentarfilm'}, R, L) \subseteq \text{film}(T, Y, R, L)$

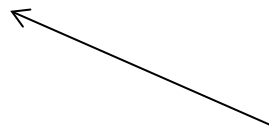
$\text{film}(T, Y, R, L), L < 100 \subseteq \text{film}(T, Y, R, L)$

$\text{film}(T, Y, R, L), \text{spielt}(T, S, O) \subseteq \text{film}(T, Y, R, L)$

$\text{film}(T, Y, R, L) \not\subseteq \text{spielt}(T, S, O)$

$\text{film}(T, _, _, _) \not\subseteq \text{spielt}(T, _, _)$

$\text{spielt}(T, _, _, _) \not\subseteq \text{film}(T, _, _)$



Bei bekannten FK-PK Constraints kann
hier Containment gelten

Beispiel

- Offensichtlich muss eine Sicht mindestens jedes Prädikat der Anfrage enthalten
- Außerdem müssen die „richtigen“ Attribute vorhanden sein
- Welche Quellen kommen für **spielt** in Frage?

```
SELECT titel, typ, rolle, kritik  
FROM   film, spielt  
WHERE  film.titel = spielt.titel;
```

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

Beispiel

- **filmkritiken**
 - Korrekt; Einschränkung auf Hauptrolle macht Tupel nicht falsch
- **us_spielfilm, kurzfilm_rollen**
 - Inkorrekt: Keine Kritiken
- **spielfilm_kritiken**
 - Korrekt

```
SELECT titel, typ, rolle, kritik
FROM   film, spielt
WHERE  film.titel = spielt.titel;
```

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

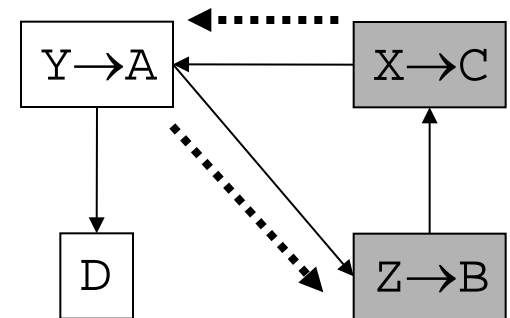
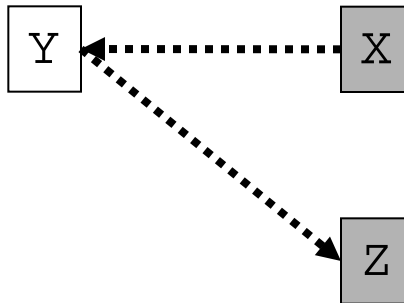
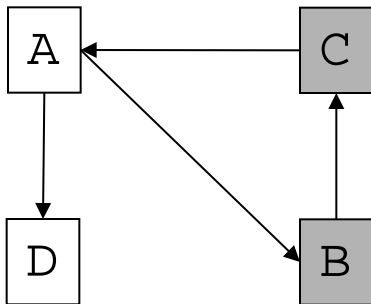
Schwierige Fälle

- Anfragen an Graphen
 - Relation `edge(x,y)` speichert Kanten von X nach Y
 - Wir suchen Subgraphen

$q(C, B) \quad \text{:-} \quad \text{edge}(A, B), \text{edge}(C, A), \text{edge}(B, C), \text{edge}(A, D)$

$p(X, Z) \quad \text{:-} \quad \text{edge}(X, Y), \text{edge}(Y, Z)$

- Ist p in q enthalten oder umgedreht?



Inhalt dieser Vorlesung

- Anfragekorrespondenzen
- Local as View
- Query Containment
- Semantische Korrektheit von Anfrageplänen

Semantische Korrektheit

- Wir können jetzt definieren, wann ein Plan semantisch korrekt ist (aber das noch nicht testen)
- Definition

Sei S ein globales Schema, q eine Anfrage gegen S , und p ein Join von Views v_1, \dots, v_n gegen S , die als rechte Seite von LaV Korrespondenzen definiert sind. Dann ist

p semantisch korrekt für q

gdw.

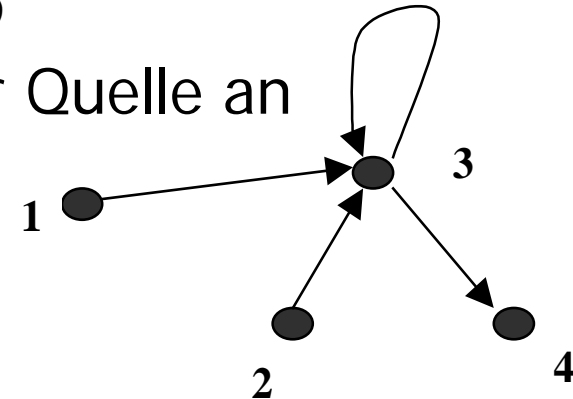
$$p \subseteq q$$

- Bemerkung
 - Die Extension von q gibt es zum Anfragezeitpunkt nicht
 - Erinnerung: Das komplette Ergebnis verlangt **Beachtung aller semantisch korrekter Pläne**

Verblüffende Effekte

- Eine Quelle für Graphkanten
- Eine Korrespondenz
 - $\text{edge}(A,B), \text{edge}(B,C) \supseteq \text{threewaypaths}(A,B,C)$
- Nehmen wir die folgende Extension einer Quelle an

From (A)	Via (B)	To (C)
1	3	3
2	3	4



- Nun führen wir darauf aus
Q: $\text{edge}(A,B), \text{edge}(B,C)$

from	via	to
1	3	3
1	3	4
3	3	3
3	3	4
2	3	3
2	3	4

Verblüffende Effekte

- Was ist nun unser Ergebnis? Die komplette Tabelle
- Wenn die Korrespondenz richtig ist
 - Dann muss die **Quelle unvollständig sein**
 - Wir können das im Mediator „reparieren“
- Wie kriegen wir das rechnerisch?
 - Sei v die linke Seite der Korrespondenz, q die globale Query
 - Dann ist v „auf verschiedene Weisen“ in q enthalten
 - Die werden wir auch alle berechnen
- Ergibt sich aus unserer Semantik der Anfrage