



Information Integration
Global-as-View: GaV

12.12.2019
Felix Naumann

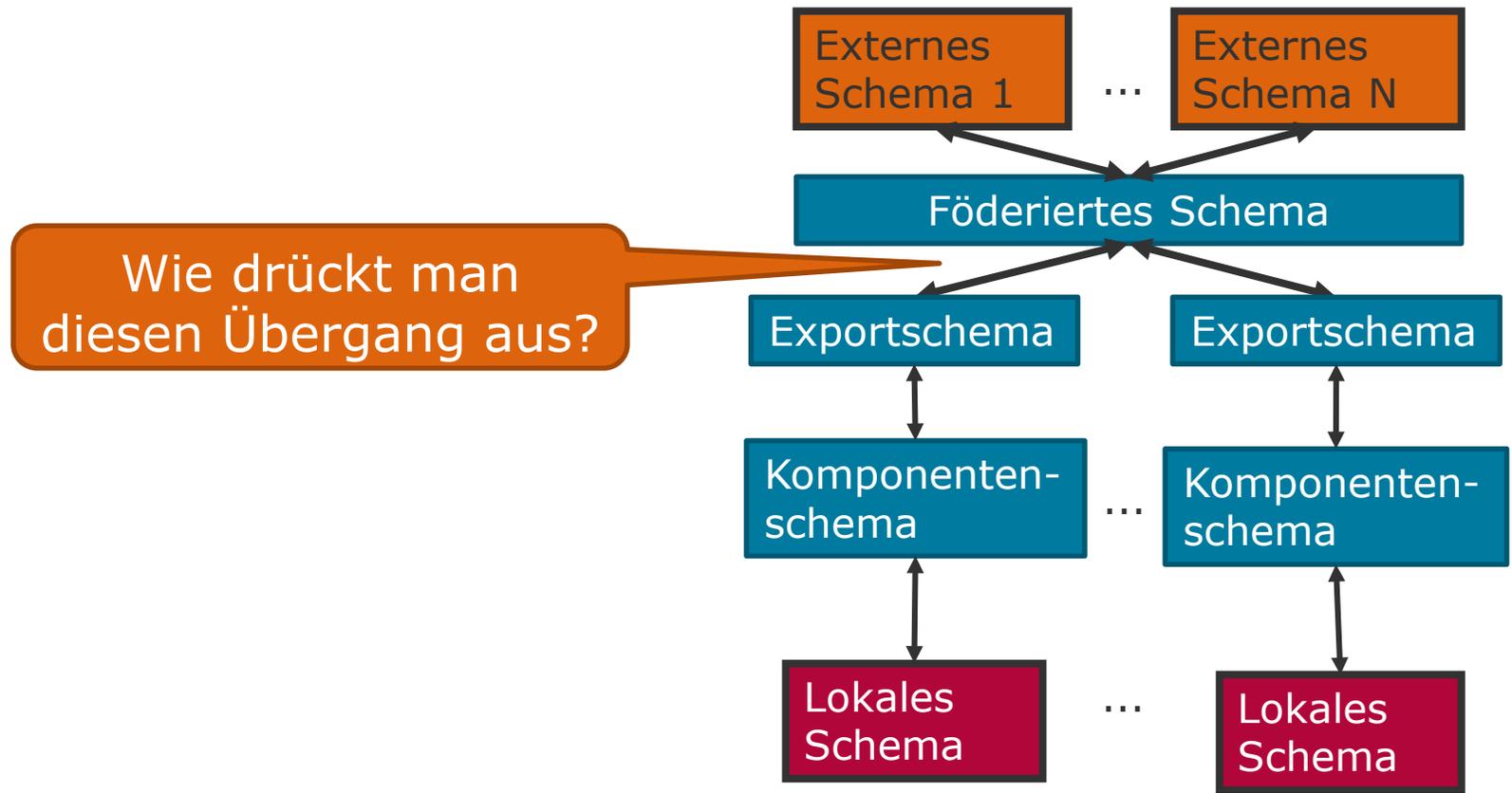
Überblick

- 1. Motivation**
2. Korrespondenzen
3. Übersicht Anfrageplanung
4. Global as View (GaV)
 - Modellierung
 - Anfragebearbeitung
5. Local as View (LaV)
 - Modellierung
 - Ausblick: Anfragebearbeitung
6. Global Local as View (GLaV)
7. Vergleich

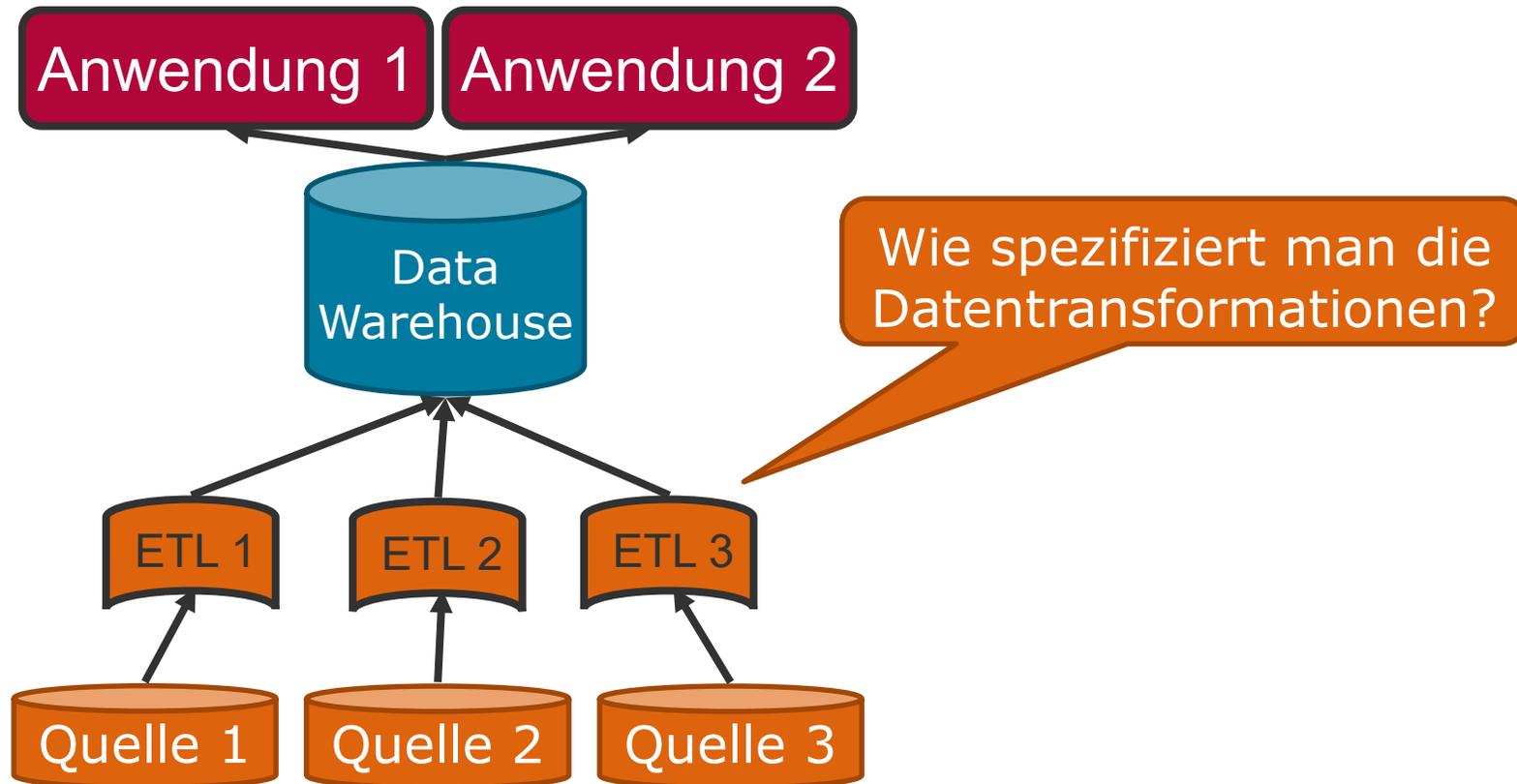


Felix Naumann
Information Integration
Winter 2019/20

Das Problem

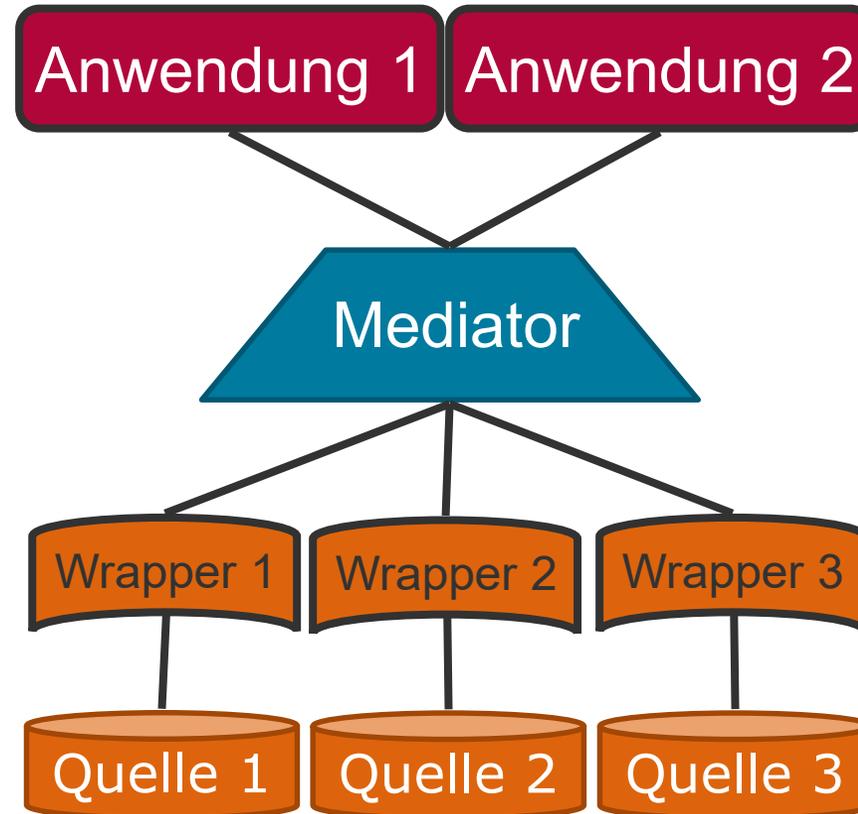


Das Problem

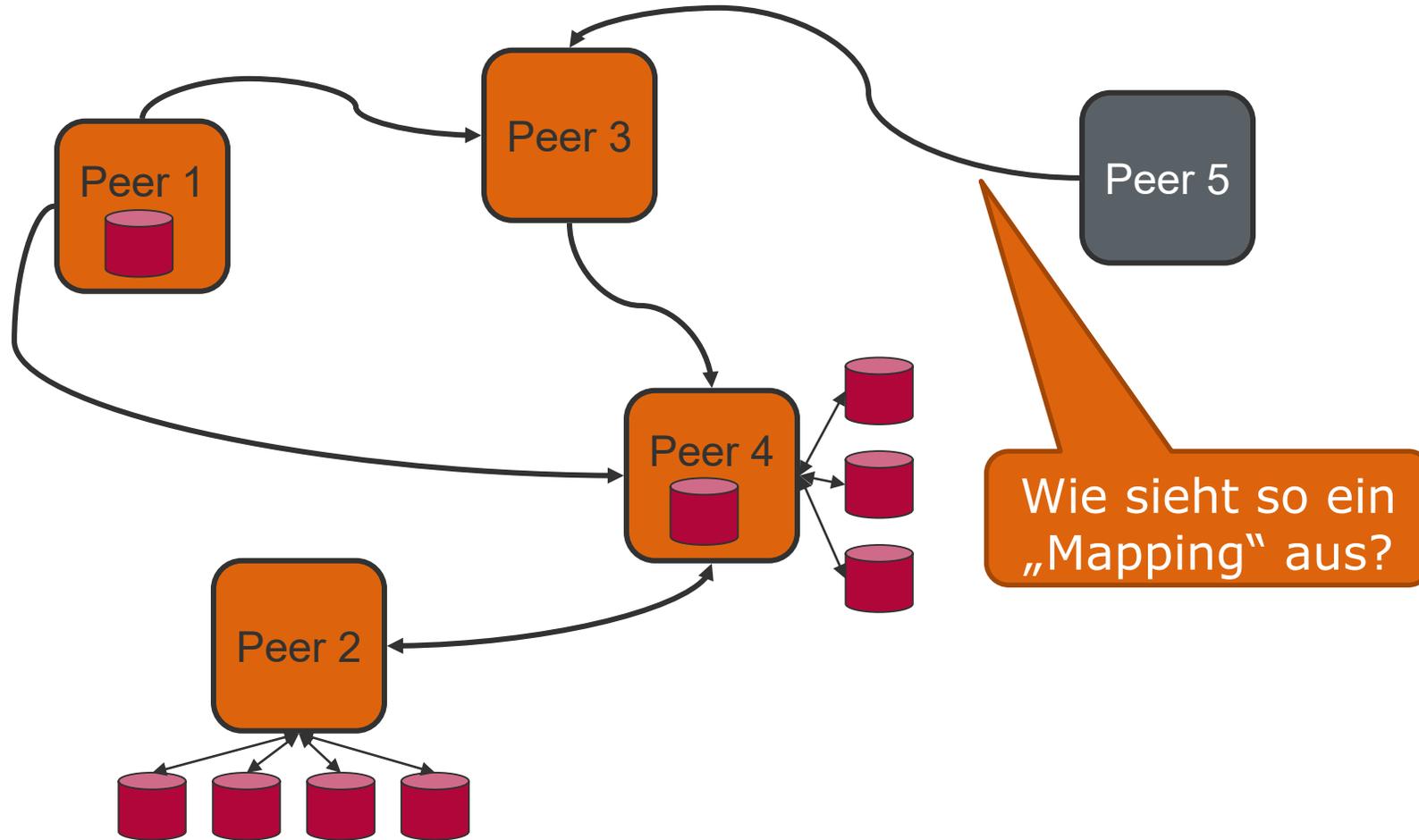


Das Problem

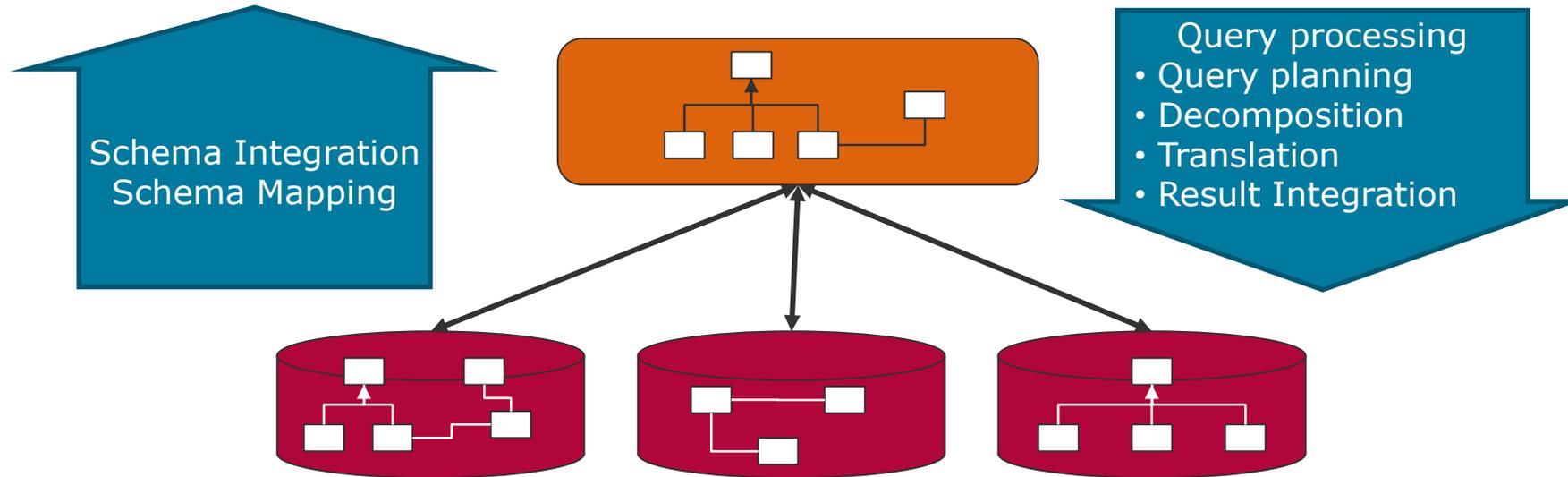
Wie drückt man die Funktionalität des Wrappers aus?



Das Problem



Abstrakter



- Wenn das integrierte System ein globales Schema zur Verfügung stellt
 - Wie beschreibt man die **Beziehungen** zwischen dem globalen und den verschiedenen lokalen Schemata?
 - Wie benutzt man diese Beziehungen zur **Anfrageplanung**?
 - (Kann man aus solchen Beziehungen das **integrierte Schema** erzeugen?)

Überblick

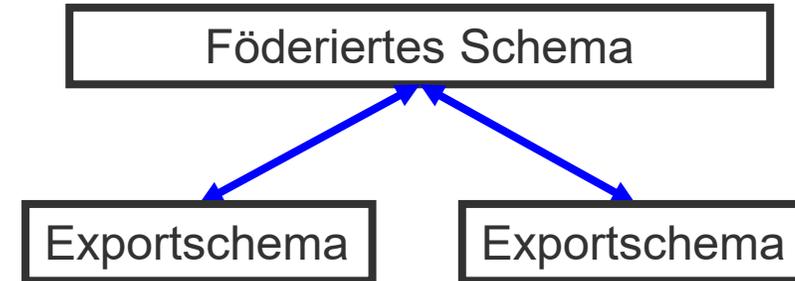
1. Motivation
- 2. Korrespondenzen**
3. Übersicht Anfrageplanung
4. Global as View (GaV)
 - Modellierung
 - Anfragebearbeitung
5. Local as View (LaV)
 - Modellierung
 - Ausblick: Anfragebearbeitung
6. Global Local as View (GLaV)
7. Vergleich



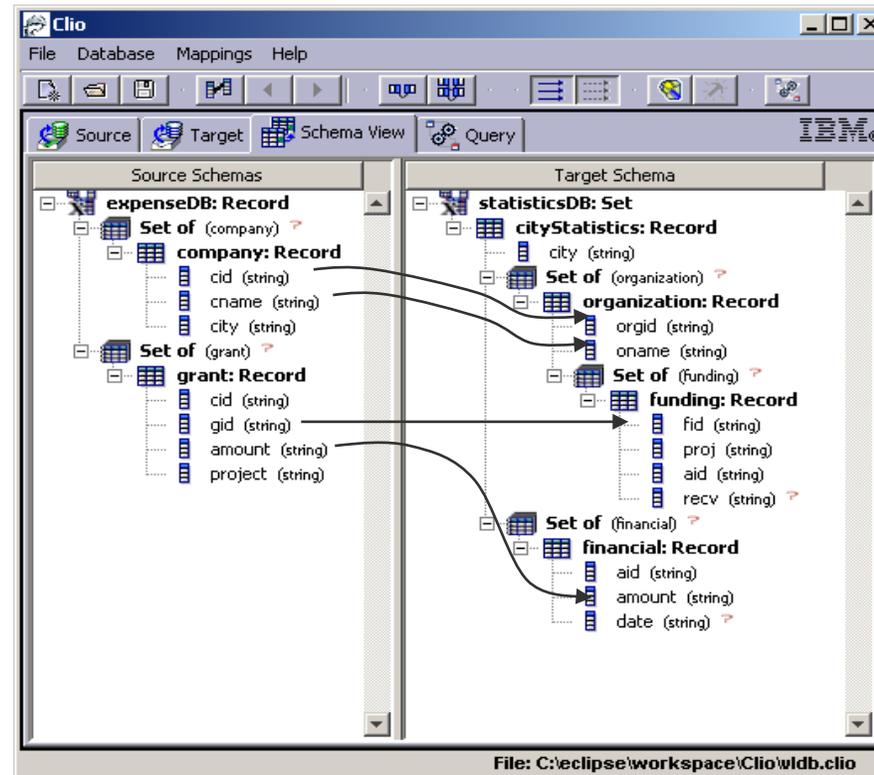
Felix Naumann
Information Integration
Winter 2019/20

Anfrage-Korrespondenzen

- Anfrage-Korrespondenz
 - Spezifikation semantisch äquivalenter Teile in Quell- und Zielschema
- Regeln mit zwei Teilen
 - Erster Teil: Anfrage an Quelle
 - Zweiter Teil: Zielorte im Zielschema
 - Dazwischen: Datentransformation und Restrukturierung
- Kann unterschiedliche „Sprachen“ benutzen
 - Global-as-View, Local-as-View, ...
- Wir wollen deklarative Korrespondenzen
 - Auch ein XSLT Programm macht Datentransformation
 - Kann aber nur schwer zur Anfrageplanung benutzt werden
- Woher kommen die Korrespondenzen?
 - Manuell spezifizieren
 - (Semi-)automatisch Finden: Schema Mapping/Matching

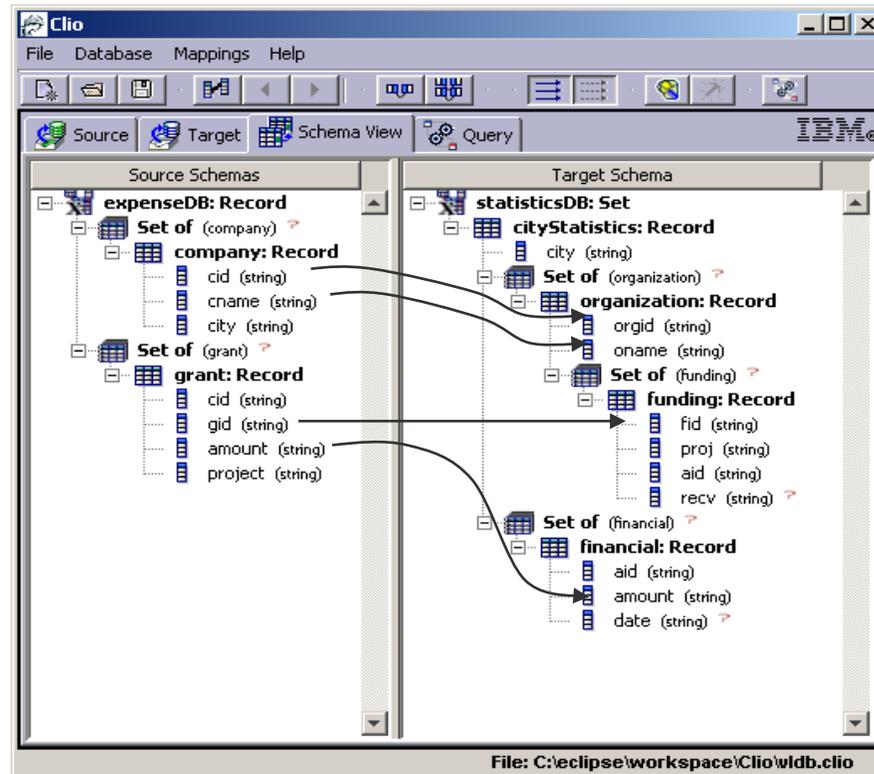


Beispiel



```
company.cid ≡ cityStatistics.organization.orgID  
company.cname ≡ cityStatistics.organization.oname  
grant.gid ≡ cityStatistics.organization.funding.fid  
grant.amount ≡ cityStatistics.financial.amount
```

Extensionale Formulierung



```
SELECT cid, cname
FROM company
```

⊆

```
SELECT orgid, oname
FROM cityStatistics/organization
```

```
SELECT amount
FROM grant
```

⊆

```
SELECT amount
FROM cityStatistics/financial
```

Virtuell und Materialisiert

- Anfragen übersetzen (virtuell)
 - Gegeben Anfrage an Zielschema
 - Finde semantisch äquivalente Menge von Anfragen an Quellschemata
 - Entfällt bei materialisierten Ansätzen

- Daten von der Quelle zum Ziel transformieren
 - Gegeben Ergebnisse der Teilanfragen
 - Oder auch den ganzen Datenbestand - Migration
 - Transformation in das Zielschema
 - Anpassung der Werte an Konventionen des Zielschemas
 - Offline (ETL, DWH) oder Online (Föderation)

- Beide Varianten sollen dieselben Korrespondenzen benutzen können.

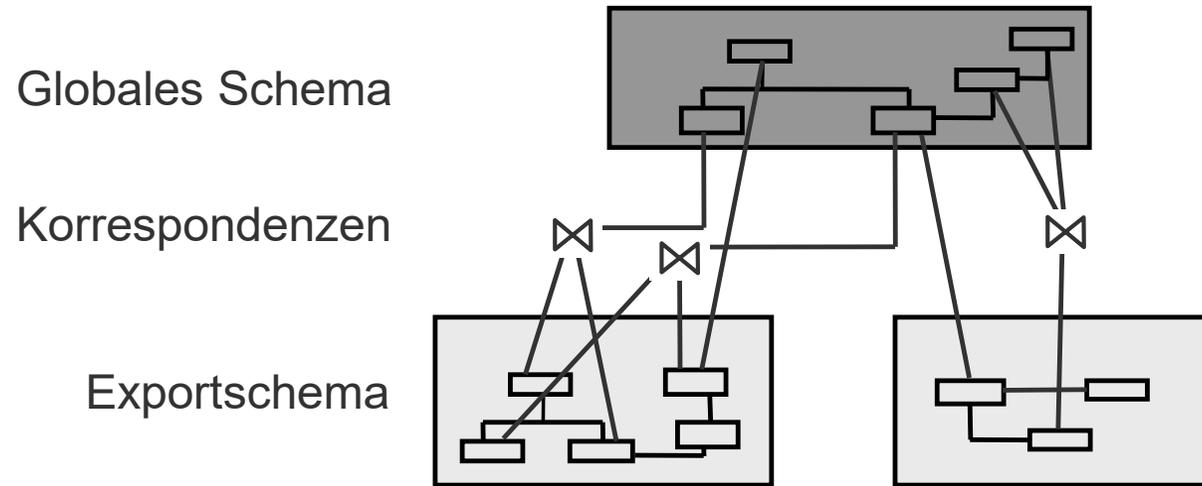
Überblick

1. Motivation
2. Korrespondenzen
- 3. Übersicht Anfrageplanung**
4. Global as View (GaV)
 - Modellierung
 - Anfragebearbeitung
5. Local as View (LaV)
 - Modellierung
 - Ausblick: Anfragebearbeitung
6. Global Local as View (GLaV)
7. Vergleich



Felix Naumann
Information Integration
Winter 2019/20

Aufgabe



- Aufgabe der Anfragebearbeitung
 - Gegeben eine Anfrage q gegen das globale Schema
 - Gegeben eine Menge von Korrespondenzen / logischen Mappings zwischen globalem und lokalen Schemata
 - Finde alle Antworten auf q .

Anfragebearbeitung – Überblick

- Schritt 1: Anfrageplanung (in den nächsten Wochen)
 - Welche Quellen können / sollen / müssen benutzt werden?
 - Welche Teile der Anfrage sollen an welche Quelle geschickt werden?
- Schritt 2: Anfrageübersetzung
 - Übersetzung aus der Sprache der Föderation in die Sprache der Quellen
 - Wrapper
- Schritt 3: Anfrageoptimierung
 - Finde geeignete Reihenfolge der Ausführung der Teilanfragen
 - Wer führt was aus (Pushen)?
 - Wer kann was ausführen (Kompensation)?
- Schritt 4: Anfrageausführung
 - Monitoring, Puffern, Cachen
 - Dynamische Reoptimierung
- Schritt 5: Ergebnisintegration
 - Duplikaterkennung und Konfliktauflösung

Schritt 1 & 2

<p>Anfrageplanung</p>	<p>Eingabe: Benutzeranfrage und Anfragekorrespondenzen</p> <p>Aufgabe: Berechnet Anfragepläne. Ein Anfrageplan ist eine Menge von logischen Anfragen an Datenquellen, in der Regel verknüpft durch Join-Prädikate. Jeder Plan muss ausführbar sein und nur semantisch korrekte Ergebnisse berechnen. Sollten mehrere Anfragepläne existieren, ist es auch Aufgabe der Anfrageplanung zu entscheiden, welche davon ausgeführt werden sollen.</p> <p>Ergebnis: Ein oder mehrere Anfragepläne</p>
<p>Anfrageübersetzung</p>	<p>Eingabe: Ein Anfrageplan</p> <p>Aufgabe: Ein Anfrageplan besteht aus Teilanfragen, die jeweils genau eine Datenquelle adressieren. Diese Teilanfragen sind syntaktisch auf die Integrationsschicht zugeschnitten. Die Anfrageübersetzung muss jede Teilanfrage in einen unmittelbar von der betreffenden Datenquelle ausführbaren Befehl übersetzen.</p> <p>Ergebnis: Ein übersetzter Anfrageplan</p>

Schritt 3 & 4

<p>Anfrageoptimierung</p>	<p>Eingabe: Ein übersetzter Anfrageplan</p> <p>Aufgabe: Anfragepläne bestehen aus Teilanfragen, für die noch nicht feststeht, in welcher Reihenfolge sie berechnet und wo in ihnen enthaltene oder sie verbindende Anfrageprädikate ausgeführt werden. Die Festlegung dieser Punkte ist die Aufgabe der Anfrageoptimierung, deren Ergebnis ein (physischer) Ausführungsplan ist.</p> <p>Ergebnis: Ein Ausführungsplan</p>
<p>Anfrageausführung</p>	<p>Eingabe: Ein Ausführungsplan</p> <p>Aufgabe: Ein Ausführungsplan kann unmittelbar ausgeführt werden, muss aber in seiner Ausführung überwacht werden, da Netzwerkverbindungen oftmals unzuverlässig sind. Außerdem müssen Prädikate der Benutzeranfrage, deren Ausführung nicht an eine Datenquelle delegiert wurden, im Integrationssystem ausgeführt werden.</p> <p>Ergebnis: Eine Menge von Ergebnistupeln</p>

Ergebnisintegration	<p>Eingabe: Pro Ausführungsplan eine Menge von Ergebnistupeln</p> <p>Aufgabe: Die Ergebnisse der einzelnen Ausführungspläne sind oftmals redundant oder uneinheitlich. Während der Ergebnisintegration werden die einzelnen Ergebnisse zu einem Gesamtergebnis integriert, was insbesondere Duplikaterkennung und Auflösen von Widersprüchen in den Daten erfordert.</p> <p>Ergebnis: Das Ergebnis der Benutzeranfrage</p>
---------------------	---

Beispiel – Domäne „Film“

- Vereinfachende Annahmen
 - Quellen bestehen nur aus einer (Export-)Relation.
 - Keine Datenmodellheterogenität
 - Nehmen wir jetzt immer an
 - Aufgabe des Wrappers
 - Zunächst vereinfachend: Keine strukturelle oder schematische Heterogenität
 - Keine semantische Heterogenität
 - Gleiche Namen = gleiche Konzepte
 - Nur offensichtliche Schreibvarianten

Beispiel

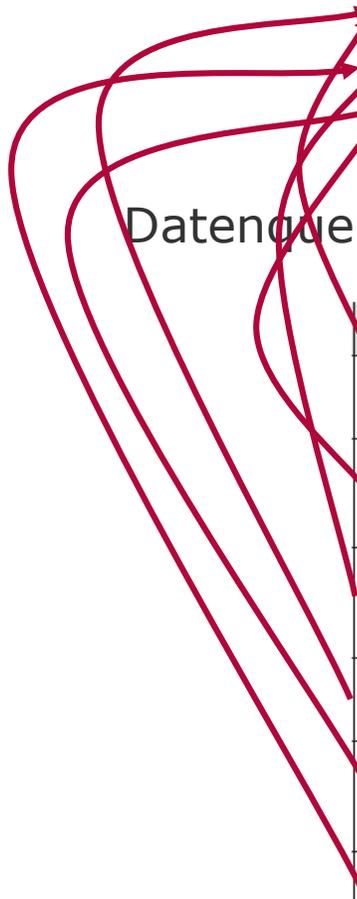
Globales Schema

```

film( titel, regisseur)
schauspieler( schauspieler_name, nationalitaet)
spielt( titel, schauspieler_name, rolle)

```

Datenquellen



Datenquelle und exportierte Relation	Beschreibung	Globale Relation
imdb.movie(titel, director)	Alle Filme der IMDB	film
imdb.acts(titel, actor_name, role)	Zuordnung von Schauspielern zu Filmen der IMDB	spielt
imdb.actor(actor_name, nationality)	Alle Schauspieler der IMDB	schauspieler
fd.film(titel, regisseur)	Alle Filme des Filmdienstes	film
fd.spielt(titel, schauspieler_name, rolle)	Zuordnung von Schauspielern zu Filmen des Filmdienstes	spielt
fd.schauspieler(schauspieler_name, nationalitaet)	Alle Schauspieler des Filmdienstes	schauspieler

Anfrageplanung

■ Globale Anfrage

- „Alle Filme, in denen Hans Albers eine Hauptrolle spielte“

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

■ Welche Quellen kommen infrage?

- **film**: Quellen `imdb.movie` und `fd.film`
- **spielt**: Quellen `imdb.acts` und `fd.spielt`

■ Ergibt vier Anfragepläne

Plan	Anfrage
p_1	<pre>SELECT title, director, role FROM imdb.movie, imdb.acts WHERE imdb.acts.actor_name = 'Hans Albers' AND imdb.acts.role = 'main actor' AND imdb.acts.title = imdb.movie.title;</pre>
p_2	<pre>SELECT title, director, rolle FROM imdb.movie, fd.spielt WHERE fd.spielt.schauspieler_name = 'Hans Albers' AND fd.spielt.rolle = 'Hauptrolle' AND fd.spielt.titel = imdb.movie.title;</pre>
p_3	<pre>SELECT titel, regisseur, role FROM fd.film, imdb.acts WHERE imdb.acts.actor_name = 'Hans Albers' AND imdb.acts.role = 'main actor' AND fd.film.titel = imdb.acts.title;</pre>
p_4	<pre>SELECT titel, regisseur, rolle FROM fd.film, fd.spielt WHERE fd.spielt.schauspieler_name = 'Hans Albers' AND fd.spielt.rolle = 'Hauptrolle' AND fd.film.titel = fd.spielt.titel;</pre>

- Jeder Plan besteht aus zwei Teilplänen
 - Auch p_1 und p_4 obwohl aus gleicher Quelle
 - Jeder Teilplan ist quellspezifisch
 - Kann von einer Quelle ausgeführt werden
- Wer führt die Joins zwischen den Teilplänen aus?
- System kann auswählen
 - Alle Pläne? (teuer, vollständig)
 - Schnellster Plan?
 - Billigster Plan?
 - Plan mit dem größten Ergebnis?

FAQ

- Kann man nicht `imdb.movie` ⋈ `imdb.acts` gemeinsam betrachten?
 - Doch – aber wir nehmen noch Quelle = „1 Relation“ an
 - Später (GaV)

- Warum sollte man `imdb.movie` und `fd.spielt` zusammen in einem Plan benutzen?
 - Weil Quellen Fehler (fehlende Werte) enthalten
 - Kombination kann neue (und wichtige!) Tupel produzieren
 - Annahme: Gleiche Werte in den Joinattributen
 - Hier: Filmnamen, Schauspielernamen

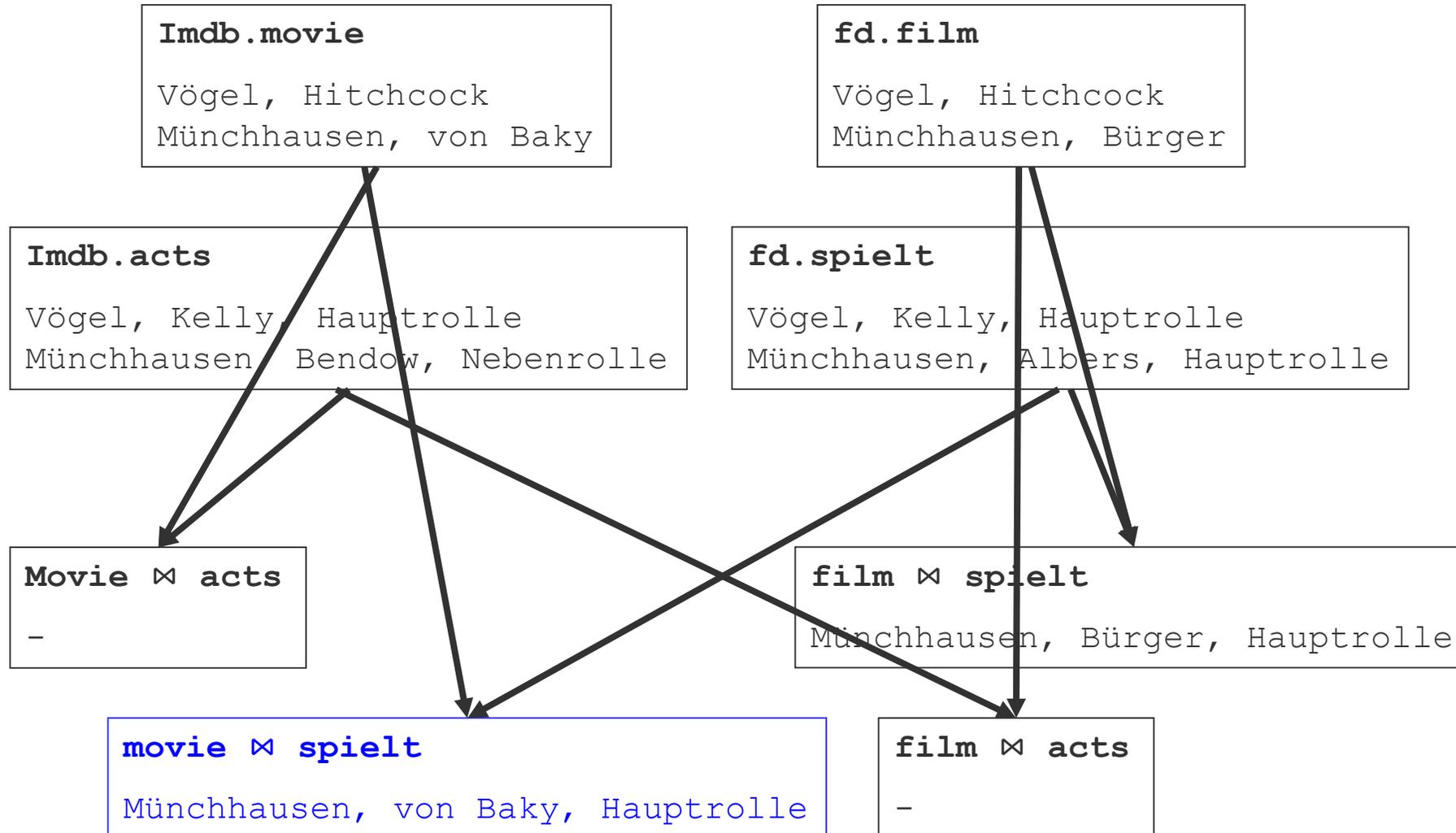
- Vorsicht vor impliziten Konsistenzannahmen

```

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

```

Beispiel



Anfrageübersetzung

- Jeder Teilplan kann an genau eine Quelle geschickt werden.
 - Übersetzung der Schemaelementnamen
 - Übersetzung einer SQL Anfrage in einen Web-Service, HTTP-Aufruf, XQuery, ...
 - Übersetzung von Konstanten in einer Query
 - Was heißt „Hauptrolle“ in der imdb?

```
SELECT titel, rolle
FROM spielt
WHERE schauspieler_name=,Hans Albers`
AND rolle=,Hauptrolle`
```

oder

```
SELECT title, role
FROM imdb.acts
WHERE actor_name=,Hans Albers`
AND role=,main role`;
```

```
SELECT title, role
FROM imdb.acts
WHERE actor_name=,Hans Albers`;
```

Anfrageoptimierung

- Lokale Optimierung: Jeder Plan einzeln
- Globale Optimierung: Gesamtmenge aller Pläne gemeinsam betrachten
- Fragen
 - Optimale Reihenfolge der Ausführung der Teilpläne?
 - Möglicherweise parallele Ausführung?
 - Selektionsbedingungen global oder lokal prüfen?

```
SELECT title, director, role
FROM imdb.movie, imdb.acts
WHERE imdb.acts.actor_name = 'Hans Albers'
      AND imdb.acts.role = 'main actor'
      AND imdb.acts.title = imdb.movie.title;
```

```
TP1:
SELECT title, director
FROM imdb.movie;
```

```
TP2:
SELECT title, role
FROM imdb.acts
WHERE actor_name='Hans Albers'
      AND role='main role';
```

Möglichkeiten

```
SELECT title, director, role
FROM imdb.movie, imdb.acts
WHERE imdb.acts.actor_name = 'Hans Albers'
      AND imdb.acts.role = 'main actor'
      AND imdb.acts.title = imdb.movie.title;
```

- Erst TP1, dann TP2, dann Join im Mediator
 - Verlangt nach Download aller Filme der IMDB
 - Vermutlich sehr teuer
- Besser: TP1 und TP2 parallel
 - Ergebnis von TP1 ist für TP2 unerheblich
 - Immer noch sehr teuer
- Erst TP2, dann TP1 umschreiben, Join im Mediator
 - TP2: nur wenige Tupel
 - Pushen dieser Tupel an TP1 „WHERE TITLE in (...)“
 - Passing bindings
 - Keine Parallelisierung mehr möglich
 - Viel kleinere Zwischenergebnisse, vermutlich viel schneller
- ...

```
TP1:
SELECT title, director
FROM imdb.movie;
```

```
TP2:
SELECT title, role
FROM imdb.acts
WHERE actor_name='Hans Albers'
      AND role='main role';
```

Anfrageausführung

- Jeder der vier Pläne wurde optimiert und übersetzt
- Teilpläne müssen ausgeführt werden
 - Anfrage abschicken, Ergebnis abwarten, lokal puffern
 - Time-Outs überwachen
 - Müssen alle Teilpläne ausgeführt werden?
 - Wenn wir P1-P4 betrachten und keine Joinbedingungen pushen können
 - Ergibt 8 Teilpläne - aber nur 4 verschiedene Teilpläne
 - Optimierungspotential
 - Entweder global optimieren oder dynamisch durch Caching erkennen
- Fehlende Operationen müssen im Mediator ausgeführt werden.
 - Nicht-pushbare Selektionen
 - Joins zwischen Relationen verschiedener Quellen
 - Sortierung, Aggregation, Duplikateliminierung, ...

Ergebnisintegration

- Jeder Plan liefert eine Menge korrekter Ergebnisse.
- Trotzdem kann es Probleme geben
 - Duplikate: Objekte sind in mehreren Quellen vorhanden
 - Konflikte: Objekte sind mehrmals mit widersprüchlichen Angaben vorhanden

Imdb.movie
 Vögel, Hitchcock
 Münchhausen, von Baky

Imdb.acts
 Vögel, Kelly, Hauptrolle
 Münchhausen, Bendow, Nebenrolle

fd.film
 Vögel, Hitchcock
 Münchhausen, Bürger

fd.spielt
 Vögel, Kelly, Hauptrolle
 Münchhausen, Albers, Hauptrolle

... ✕ ... UNION ... ✕ ... UNION ...
 -
 Münchhausen, Bürger, Hauptrolle
 Münchhausen, von Baky, Hauptrolle
 -

Überblick

1. Motivation
2. Korrespondenzen
3. Übersicht Anfrageplanung
4. **Global as View (GaV)**
 - Modellierung
 - Anfragebearbeitung
5. Local as View (LaV)
 - Modellierung
 - Ausblick: Anfragebearbeitung
6. Global Local as View (GLaV)
7. Vergleich



Felix Naumann
Information Integration
Winter 2019/20

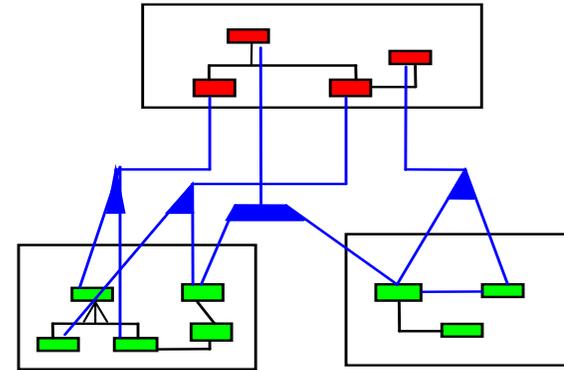
Modellierung von Datenquellen

- 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

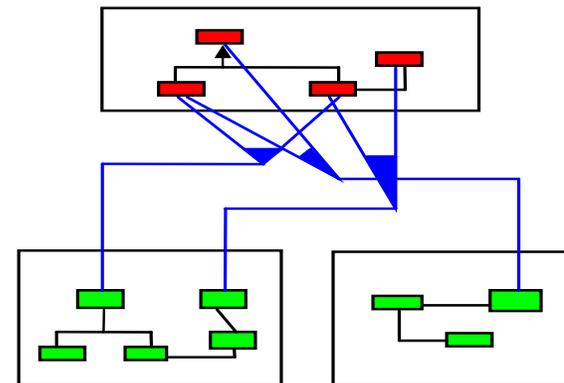
■ Global as View

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



■ Local as View

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



Global as View (GaV) – Beispiel

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 Film AS
SELECT * FROM IMDB
UNION
SELECT * FROM MyMovies
UNION
SELECT RegieDB.Titel, RegieDB.Regie, GenreDB.Jahr, GenreDB.Genre
FROM RegieDB, GenreDB
WHERE RegieDB.Titel = GenreDB.Titel
```

Typisches Merkmal bei GaV

Join über mehrere Quellen
(Anfragebearbeitung!)

Felix Naumann
Information Integration
Winter 2019/20

Zwei Arten von GaV Regeln

- Einfache GaV Regeln: Quellspezifisch
 - Lokale Anfrage adressiert nur eine Quelle
 - Verknüpfung über Quellen hinweg übernimmt die Anfrageplanung
 - Besser wartbar (Änderung einer Quelle kann nur wenige Sichten betreffen)
 - Einfacher zu erweitern (bestehende Views nicht anfassen)
- Komplexe GaV Regeln: Quellübergreifend
 - Lokale Anfrage ist eine Anfrage in einer Multidatenbanksprache
 - Eine Sicht fasst alle Objekte einer globalen Relation zusammen
 - Sicht film, schauspieler, ...
 - Anfrageplanung übernimmt nur noch Verknüpfung zwischen verschiedenen Relationen, d.h., Joins in der globalen Anfrage
 - Weniger robust, schwerer zu erweitern
 - Änderung einer Quelle kann alle Sichten betreffen
 - Konfliktlösung kann in den View eingebaut werden

Global as View (GaV) – Beispiel

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

S5: MDB(Titel, Regie, Jahr)

S6: MovDB(Titel, Regie, Genre)

```
CREATE VIEW Film AS
SELECT Titel, Regie, Jahr, NULL AS Genre
FROM MDB
      UNION
SELECT Titel, Regie, NULL AS Jahr, Genre
FROM MovDB
      UNION
SELECT Titel, Regie, Jahr, Genre
FROM MDB NATURAL JOIN MovDB
```

Fehlende Attribute

Entspricht in etwa
Full Outer Join

Felix Naumann
Information Integration
Winter 2019/20

Global as View (GaV) – Beispiel

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

S7: KinoDB(Kino, Genre)

```
CREATE VIEW Film AS  
SELECT NULL, NULL, NULL, Genre  
FROM KinoDB
```

```
CREATE VIEW Programm AS  
SELECT Kino, NULL, NULL  
FROM KinoDB
```

Allenfalls
interessant für
passing bindings

- Problem: **Assoziationen werden getrennt**
 - Da Ergebnis einer Sicht nur eine globale Relation sein kann
 - Lösung eventuell: Skolemfunktionen (hier nicht geeignet)
- Tritt auf, wenn
 - Attribute verschiedener globaler Relationen in einer lokalen Relation liegen und der verbindende Key/FK lokal nicht existiert
 - Nicht im umgekehrten Fall: einfach lokale Relationen joinen...

Vorschau: Local as View (LaV) – Beispiel

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
```

- Umgekehrte Betrachtung
- Assoziationen bleiben erhalten
 - Titel wird zwar nie instantiiert,
 - aber man weiß, dass es einen verbindenden Titel gibt.

Integritätsconstraints (ICs)

- Schränken Wertebereiche ein, schließen Werte aus, schließen Kombinationen von Werten aus, ...
- Dienen zur näheren Bestimmung der Intension einer Relation:
 - `teure_autos(auto, preis), preis>50.000`
 - `golfclub_mitglieder(name, alter), alter>55`
 - `film(title, type), type∈{spielfilm, doku, kurzfilm}`
 - ...
- Für die Integration sehr wichtig

Global as View (GaV) – Globale ICs

Globales Schema

NeuerFilm(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

IC: Jahr > 2000

S1: IMDB(Titel, Regie, Jahr, Genre)
S2: MyMovies(Titel, Regie, Jahr, Genre)

```
CREATE VIEW NeuerFilm AS
SELECT * FROM IMDB
WHERE Jahr > 2000
UNION
SELECT * FROM MyMovies
WHERE Jahr > 2000
```

- IC auf globalem Schema kann in diesem Fall in die Sichtdefinition aufgenommen werden
- Ausführung in der Quelle (push) oder im Mediator

Probleme mit globalen IC

Globales Schema

Neuerfilm(Titel, Regie, Jahr, Genre);
Programm(Kino, Titel, Zeit);
IC: Jahr > 2000

```
S1: IMDB( Titel, Regie, Jahr, Genre)  
S2: MyOldOrNewMovies( Titel, Regie)
```

```
CREATE VIEW NeuerFilm AS  
SELECT *  
FROM IMDB  
WHERE Jahr > 2000  
UNION  
SELECT Titel, Regie, NULL, NULL  
FROM MyOldOrNewMovies  
WHERE ??
```

- Quelle gibt keine Jahr-Informationen
- IC auf globalem Schema kann nicht überprüft werden
- Quelle kann nicht integriert werden

Global as View (GaV) – Lokale ICs

Globales Schema

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

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



```
CREATE VIEW Film AS
SELECT Titel, Regie, NULL, Genre
FROM NeueFilme
```

- Bekannte Nebenbedingung auf der Quelle kann nicht modelliert werden.
- Warum ist das schade?
 - `SELECT * FROM film WHERE jahr < 1950`

„Trick“ falls Attribute exportiert wird

Globales Schema

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

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

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

- IC in Quelle wird modelliert
 - (Geht nur, wenn das Attribut in der Quelle instantiiert wird)
- Die Bedingung ändert am Ergebnis nichts
 - Für eine lokale Anfrage wäre sie sinnlos
- Aber sie hilft dem Anfrageplaner
 - `SELECT * FROM film WHERE jahr < 1950`

Global as View (GaV) – globale ICs

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 NeuerFilm AS
SELECT * FROM AlleFilmeNett
WHERE Jahr > 2000
oder
CREATE VIEW NeuerFilm AS
SELECT Titel, Regie, Genre FROM AlleFilmeNett
WHERE Jahr > 2000
```

Quelle S2 kann nicht
modelliert werden!

Global as View (GaV) – ICs

Fazit

- Nebenbedingungen (ICs) im globalen Schema
 - können die Integration von Quellen verhindern,
 - wenn die Bedingung nicht geprüft werden kann (fehlendes Attribut).

- Nebenbedingungen in den lokalen Schemata
 - können modelliert werden,
 - (wenn sie auf exportierten Attributen gelten, oder)
 - wenn sie auf globalen Attributen gelten und die Form `attribute = <constant>` haben.
 - tragen möglicherweise nicht zum Anfrageergebnis bei,
 - wenn Nebenbedingung nicht modelliert wurde.

Überblick

1. Motivation
2. Korrespondenzen
3. Übersicht Anfrageplanung
4. **Global as View (GaV)**
 - Modellierung
 - **Anfragebearbeitung**
5. Local as View (LaV)
 - Modellierung
 - Ausblick: Anfragebearbeitung
6. Global Local as View (GLaV)
7. Vergleich



Felix Naumann
Information Integration
Winter 2019/20

Anfragebearbeitung – GaV

■ Gegeben:

- Anfrage an globales Schema
 - Insbesondere: Auf Relationen des globalen Schemas
- Für jede globale Relation genau eine Sicht auf lokale Quellen

■ Gesucht:

- Alle Tupel, die die Anfragebedingungen erfüllen
- Aber: Daten sind in lokalen Quellen gespeichert.

■ Idee:

- Ersetze jede Relation der Anfrage durch ihre Sicht
 - Sichtentfaltung, View Expansion, Query Unfolding, ...
- Geschachtelte Anfrage

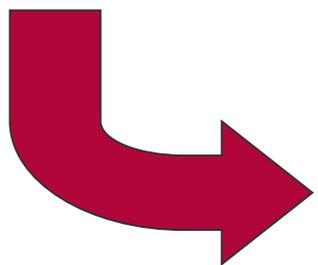
GaV – Beispiel

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

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

SELECT *
FROM Film

Globale Sicht für Film



```
SELECT *  
FROM ( SELECT * FROM IMDB  
        UNION  
        SELECT R.Titel, R.Regie, G.Jahr, G.Genre  
        FROM RegieDB R, GenreDB G  
        WHERE R.Titel = G.Titel  
      )
```

GaV – Beispiel

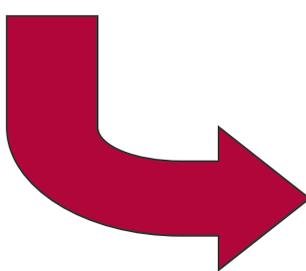
Globales Schema

Film(Titel, Regie, Jahr, Genre)

Programm(Kino, Titel, Zeit)

```
SELECT Titel, Jahr
FROM Film
WHERE Jahr = ,2003'
```

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

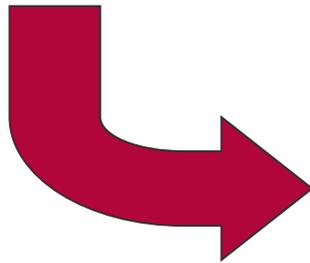


```
SELECT Titel, Jahr
FROM (
  SELECT * FROM IMDB
  UNION
  SELECT R.Titel, R.Regie, G.Jahr, G.Genre
  FROM RegieDB R, GenreDB G
  WHERE R.Titel = G.Titel
)
WHERE Jahr = ,2003'
```

GaV – Beispiel

```
SELECT F.Titel, P.Kino  
FROM Film F, Programm P  
WHERE F.Titel = P.Titel  
AND P.Zeit > 20:00
```

```
S1: IMDB(Titel, Regie, Jahr, Genre)  
S2: RegieDB(Titel, Regie)  
S3: GenreDB(Titel, Jahr, Genre)  
S7: KinoDB(Titel, Kino, Genre, Zeit)
```



```
SELECT F.Titel, P.Kino  
FROM ( SELECT * FROM IMDB  
        UNION  
        SELECT R.Titel, R.Regie, G.Jahr, G.Genre  
        FROM RegieDB R, GenreDB G  
        WHERE R.Titel = G.Titel  
      ) AS F,  
      ( SELECT *  
        FROM KinoDB  
      ) AS P  
WHERE F.Titel = P.Titel  
AND P.Zeit > 20:00
```

GaV - Schachtelung

- Geschachtelte Anfragen
 - Beliebig tiefe Schachtelung
 - Bearbeitung
 - Konzeptionell:
Ausführung der Anfragen von innen nach außen und Speicherung in temporären Relationen.
 - Tatsächlich:
Optimierungspotential durch Umschreiben der Anfrage (Entschachtelung)
 - Caching und Materialisierte Sichten (materialized views)

GaV - Schachtelung

■ Schachtelung

- i.d.R. Belassen bei entfernten Quellen: Innere Anfragen sind Anfragen an Quellen
- i.d.R. Auflösung bei materialisierten Sichten: Innere Anfragen und äußere Anfragen sind Anfragen an Basisrelationen

```

SELECT F.Titel, P.Kino
FROM (
  SELECT * FROM IMDB
      UNION
  SELECT R.Titel, R.Regie, G.Jahr, G.Genre
   FROM RegieDB R, GenreDB G
   WHERE R.Titel = G.Titel
) AS F,
(
  SELECT *
   FROM KinoDB
) AS P
WHERE F.Titel = P.Titel
AND   P.Zeit > 20:00

```

```

SELECT F.Titel, P.Kino
FROM   IMDB F, KinoDB P
WHERE  F.Titel = P.Titel
AND    P.Zeit > 20:00

UNION

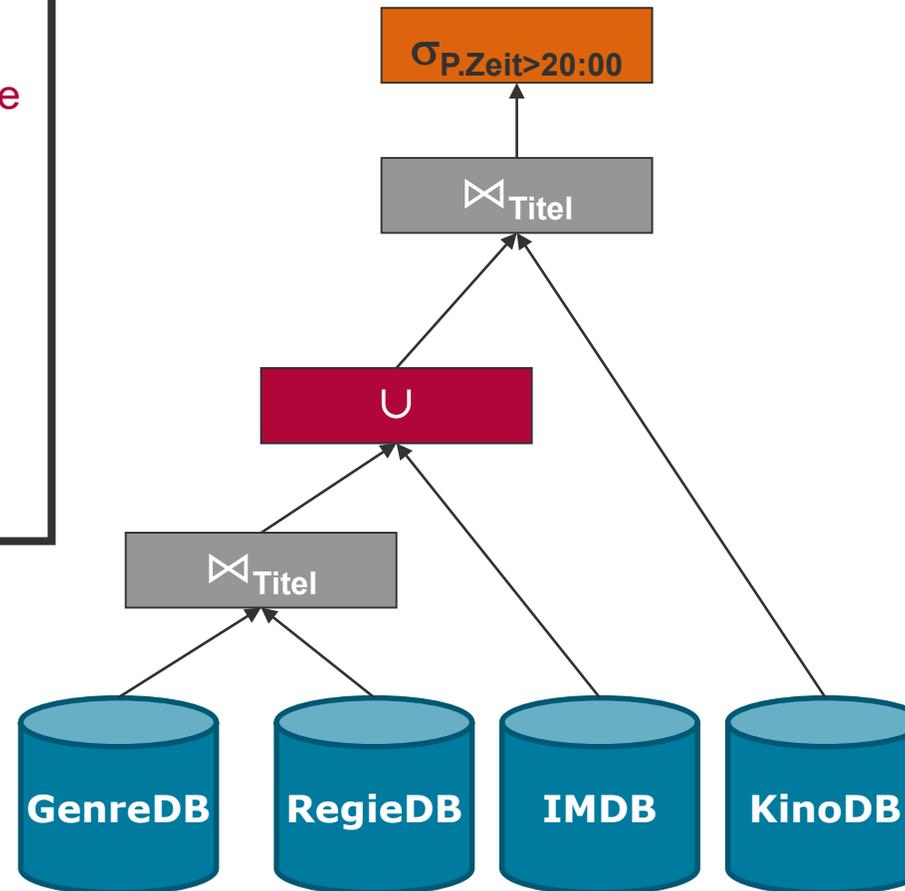
SELECT F1.Titel, P1.Jino
FROM   RegieDB F1, GenreDB F2, KinoDB P1
WHERE  F1.Titel = F2.Titel
WHERE  F1.Titel = P1.Titel
AND    P1.Zeit > 20:00

```

GaV - Schachtelung

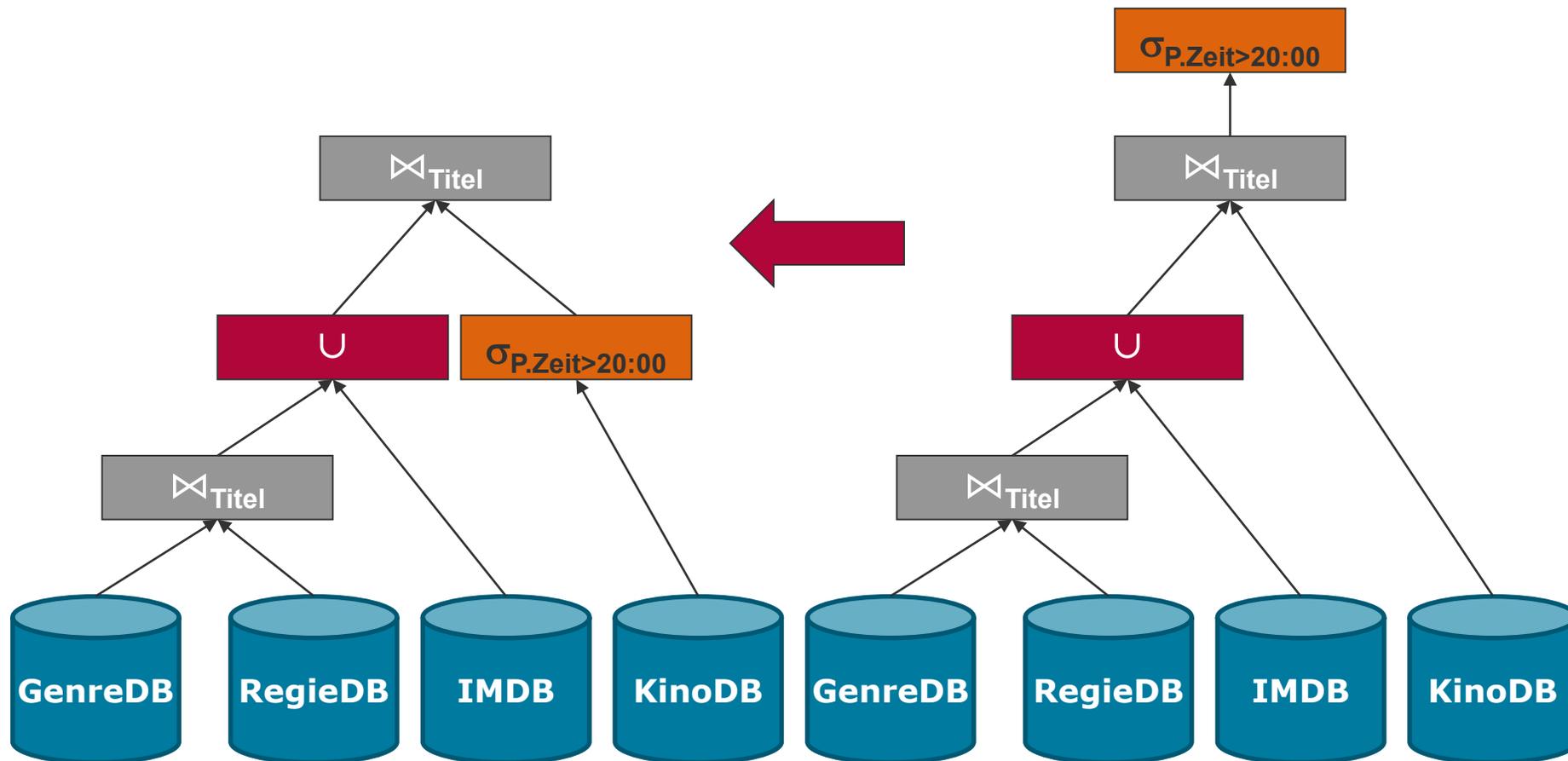
```

SELECT F.Titel, P.Kino
FROM (
  SELECT * FROM IMDB
  UNION
  SELECT R.Titel, R.Regie, G.Jahr, G.Genre
  FROM RegieDB R, GenreDB G
  WHERE R.Titel = G.Titel
) AS F,
(
  SELECT *
  FROM KinoDB
) AS P
WHERE F.Titel = P.Titel
AND P.Zeit > 20:00
  
```



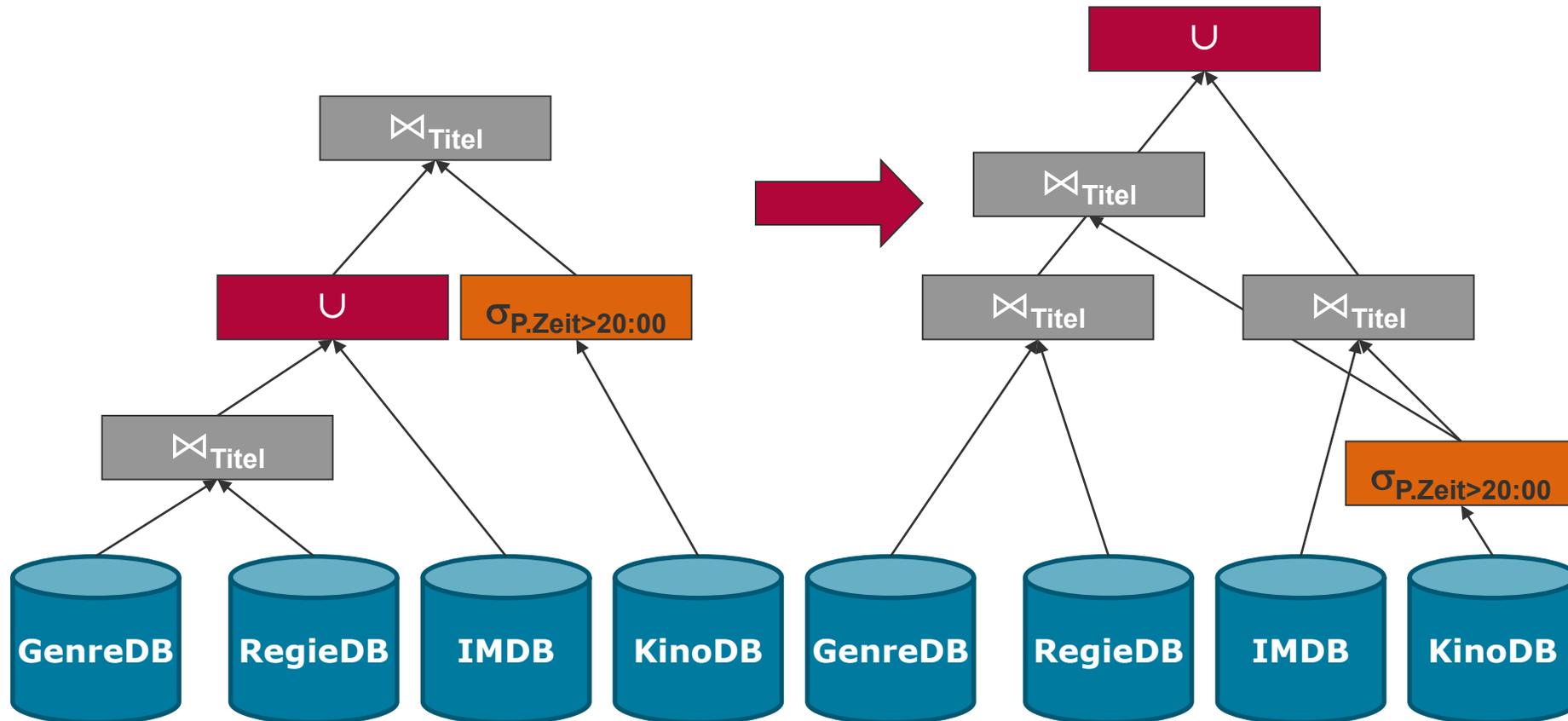
Felix Naumann
Information Integration
Winter 2019/20

GaV - Schachtelung



Felix Naumann
Information Integration
Winter 2019/20

GaV - Schachtelung



Felix Naumann
Information Integration
Winter 2019/20

GaV - Schachtelung

Globales Schema

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

S5: MDB(Titel, Zeit, Regie)
 S6: MovDB(Kino, Genre, Titel)

```
CREATE VIEW Film AS
SELECT Titel, Regie, Genre
FROM MDB, MovDB
WHERE MDB.Titel = MovDB.Titel
```

```
SELECT F.Titel, P.Kino
FROM Film F, Programm P
WHERE F.Titel = P.Titel
AND P.Zeit > 20:00
```

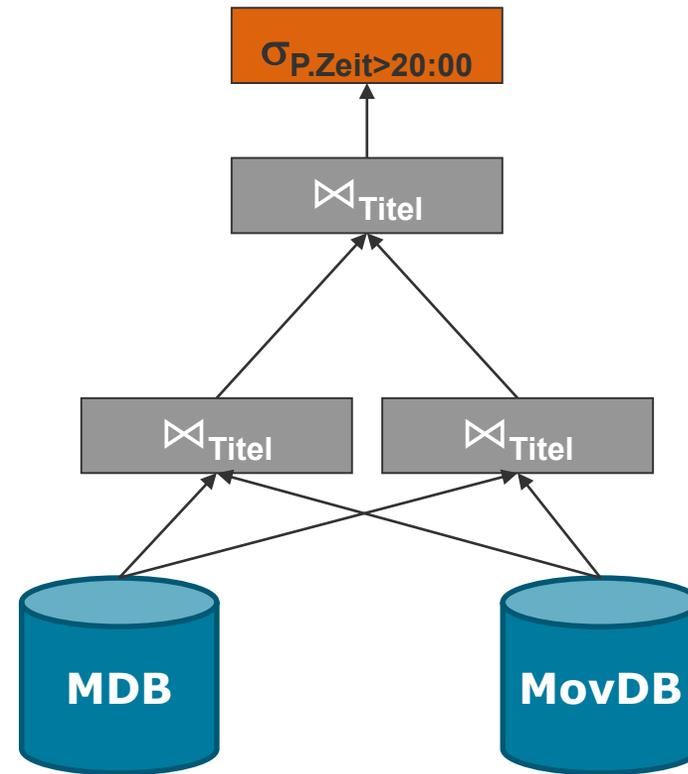
```
CREATE VIEW Programm AS
SELECT Kino, Titel, Zeit
FROM MovDB, MDB
WHERE MDB.Titel = MovDB.Titel
```

```
SELECT F.Titel, F.Zeit
FROM ( SELECT Titel, Regie, Genre
      FROM MDB, MovDB
      WHERE MDB.Titel = MovDB.Titel
    ) AS F,
( SELECT Kino, Titel, Zeit
  FROM MovDB, MDB
  WHERE MDB.Titel = MovDB.Titel
) AS P
WHERE F.Titel = P.Titel
AND P.Zeit > 20:00
```

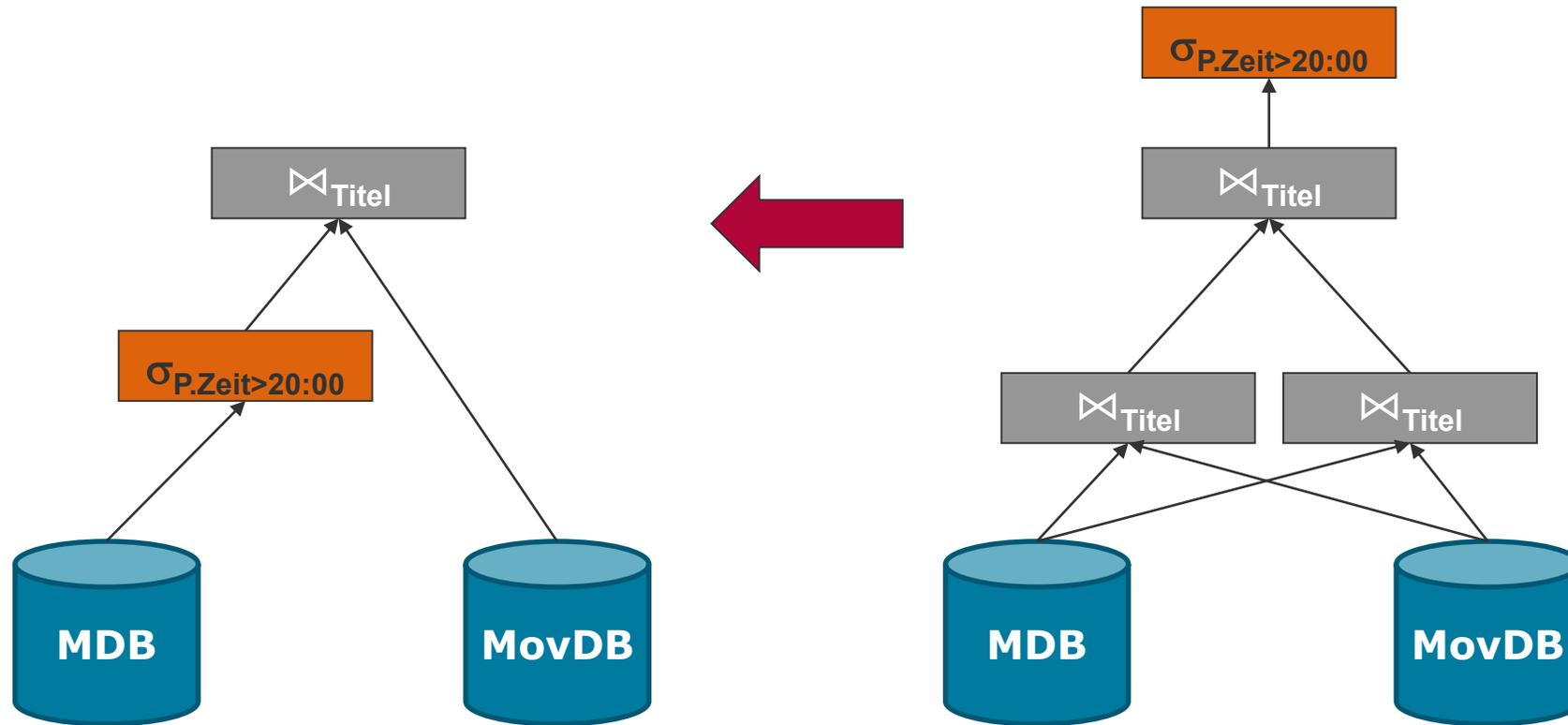
GaV - Schachtelung

```

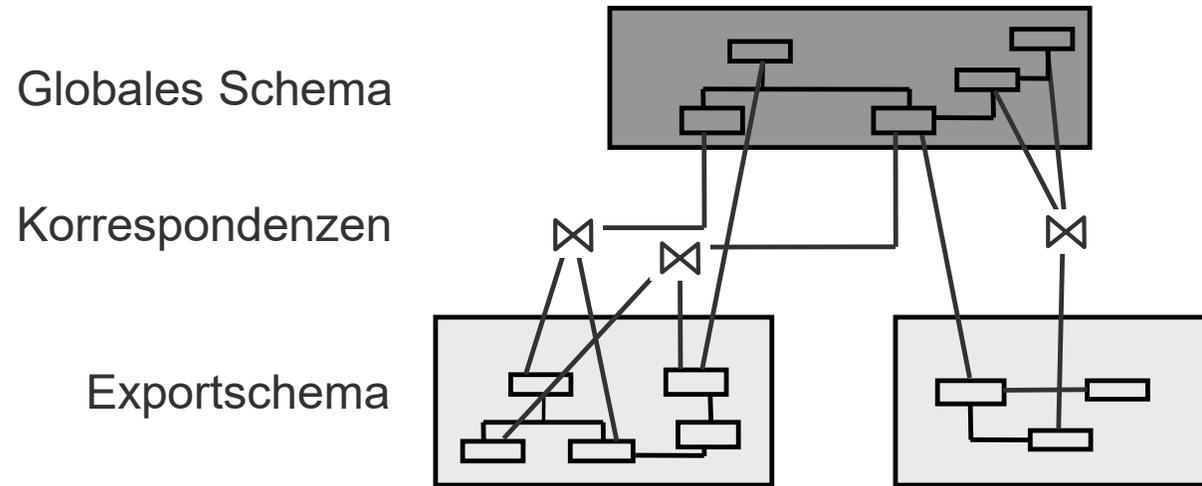
SELECT F.Titel, F.Zeit
FROM ( SELECT Titel, Regie, Genre
      FROM MDB, MovDB
      WHERE MDB.Titel = MovDB.Titel
    ) AS F,
( SELECT Kino, Titel, Zeit
  FROM MovDB, MDB
  WHERE MDB.Titel = MovDB.Titel
) AS P
WHERE F.Titel = P.Titel
AND   P.Zeit > 20:00
  
```



GaV - Schachtelung



Zusammenfassung – GaV



- Aufgabe der Anfragebearbeitung
 - Gegeben eine Anfrage q gegen das globale Schema
 - Gegeben eine Menge von Korrespondenzen / logischen Mappings zwischen globalem und lokalen Schemata
 - Finde alle Antworten auf q .