Modellierung von Datenquellen

Komplexe Informationssysteme

Fabian Panse

panse@informatik.uni-hamburg.de

Universität Hamburg

Universität Hamburg

DER FORSCHUNG | DER LEHRE | DER BILDUNG

Agenda

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

- welche Daten die Quellen enthält

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Typische Bestandteile einer Quellbeschreibung sind:

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Typische Bestandteile einer Quellbeschreibung sind:

- Schema mapping

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Typische Bestandteile einer Quellbeschreibung sind:

- Schema mapping
- Zugriffsbeschränkungen (*access-pattern limitations*)

Modellierung von Datenquellen

Eine Quellbeschreibung enthält Informationen über

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Typische Bestandteile einer Quellbeschreibung sind:

- Schema mapping
- Zugriffsbeschränkungen (*access-pattern limitations*)
- Informationen über die Vollständigkeit der Quelle

Schema Mapping

Agenda

Schema Mapping Sprachen

Schema Mapping Sprachen

- Formal: Menge an Ausdrücken, welche die Beziehung zwischen zwei Schemata beschreibt

Schema Mapping Sprachen

- Formal: Menge an Ausdrücken, welche die Beziehung zwischen zwei Schemata beschreibt
- Im Kontext der Informationsintegration ein Mapping zwischen einem lokalen Schema und dem globalen Schema

Schema Mapping Sprachen

- Formal: Menge an Ausdrücken, welche die Beziehung zwischen zwei Schemata beschreibt
- Im Kontext der Informationsintegration ein Mapping zwischen einem lokalen Schema und dem globalen Schema

Eigenschaften einer Sprache:

Schema Mapping Sprachen

- Formal: Menge an Ausdrücken, welche die Beziehung zwischen zwei Schemata beschreibt
- Im Kontext der Informationsintegration ein Mapping zwischen einem lokalen Schema und dem globalen Schema

Eigenschaften einer Sprache:

- Ausdrucksmächtigkeit

Schema Mapping Sprachen

- Formal: Menge an Ausdrücken, welche die Beziehung zwischen zwei Schemata beschreibt
- Im Kontext der Informationsintegration ein Mapping zwischen einem lokalen Schema und dem globalen Schema

Eigenschaften einer Sprache:

- Ausdrucksmächtigkeit
- Effizientes Umschreiben von Anfragen

Schema Mapping Sprachen

- Formal: Menge an Ausdrücken, welche die Beziehung zwischen zwei Schemata beschreibt
- Im Kontext der Informationsintegration ein Mapping zwischen einem lokalen Schema und dem globalen Schema

Eigenschaften einer Sprache:

- Ausdrucksmächtigkeit
- Effizientes Umschreiben von Anfragen
- Erweiterbarkeit bzgl. neuer Quellen

Sichtenbasiertes Schema Mapping

Sichtenbasiertes Schema Mapping

- Kernidee
  - Modellierung strukturell heterogener Quellen in Bezug auf ein globales Schema als Views (Sichten)
  - Relationales Modell

Sichtenbasiertes Schema Mapping

- 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.

Sichtenbasiertes Schema Mapping

- 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)

Wir betrachten:

Sichtenbasiertes Schema Mapping

- 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)

Wir betrachten:

- Global as View (effizientes Umschreiben)

Sichtenbasiertes Schema Mapping

- 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)

Wir betrachten:

- Global as View (effizientes Umschreiben)
- Local as View (einfach erweiterbar)

Sichtenbasiertes Schema Mapping

- 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)

Wir betrachten:

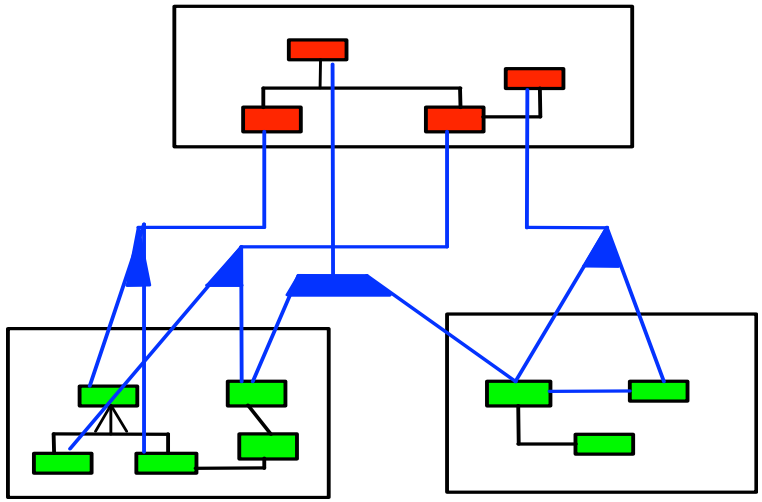
- Global as View (effizientes Umschreiben)
- Local as View (einfach erweiterbar)
- Global Local as View (sehr mächtig)

Agenda

Global as View / Local as View

Global as View / Local as View

- Global as View: Relationen des globalen Schemas werden als Sichten auf die lokalen Schemas der Quellen ausgedrückt.



Global as View (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!)

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

Global as View (Beispiel)

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

S5: MDB(Titel, Regie, Jahr)
S6: MovDB(Titel2, Regie, Genre)

```
CREATE VIEW Film AS
SELECT Titel, Regie, Jahr, NULL AS Genre
FROM MDB
      UNION
SELECT Titel2, Regie, NULL AS Jahr, Genre
FROM MovDB
```

Fehlende Attribute

Form des Ergebnisses; Datenfusion

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

Global as View (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
```

- Problem: Assoziationen werden getrennt
  - Da Ergebnis einer Sicht nur eine globale Relation sein kann
  - Lösung: Skolemfunktion (injektive Funktion f(Kino,Genre))

Global as View (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
```

- Problem: Assoziationen werden getrennt
  - Da Ergebnis einer Sicht nur eine globale Relation sein kann
  - Lösung: Skolemfunktion (injektive Funktion f(Kino,Genre))
- Tritt auf, wenn
  - Attribute verschiedener globaler Relationen in einer lokalen Relation liegen und der verbindende FK lokal nicht existiert
  - Nicht im umgekehrten Fall: einfach lokale Relationen joinen ...

Vorschau: Local as View (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
```

Vorschau: Local as View (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

Vorschau: Local as View (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.

Global as View - 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
```

Global as View - 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
```

- *integrity constraint* (IC) auf globalem Schema kann in diesem Fall in die Sichtdefinition aufgenommen werden

Global as View - 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
```

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

Probleme mit Globalen ICs

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  ??
```

Probleme mit Globalen ICs

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 beinhaltet keine Jahr-Informationen

Probleme mit Globalen ICs

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 beinhaltet keine Jahr-Informationen
- IC auf globalem Schema kann nicht überprüft werden

Probleme mit Globalen ICs

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 beinhaltet keine Jahr-Informationen
- IC auf globalem Schema kann nicht überprüft werden
- Quelle kann nicht integriert werden (oder Verletzung der IC muss in Kauf genommen werden $\Rightarrow$ keine Konsistenz)

Global as View - Globale ICs

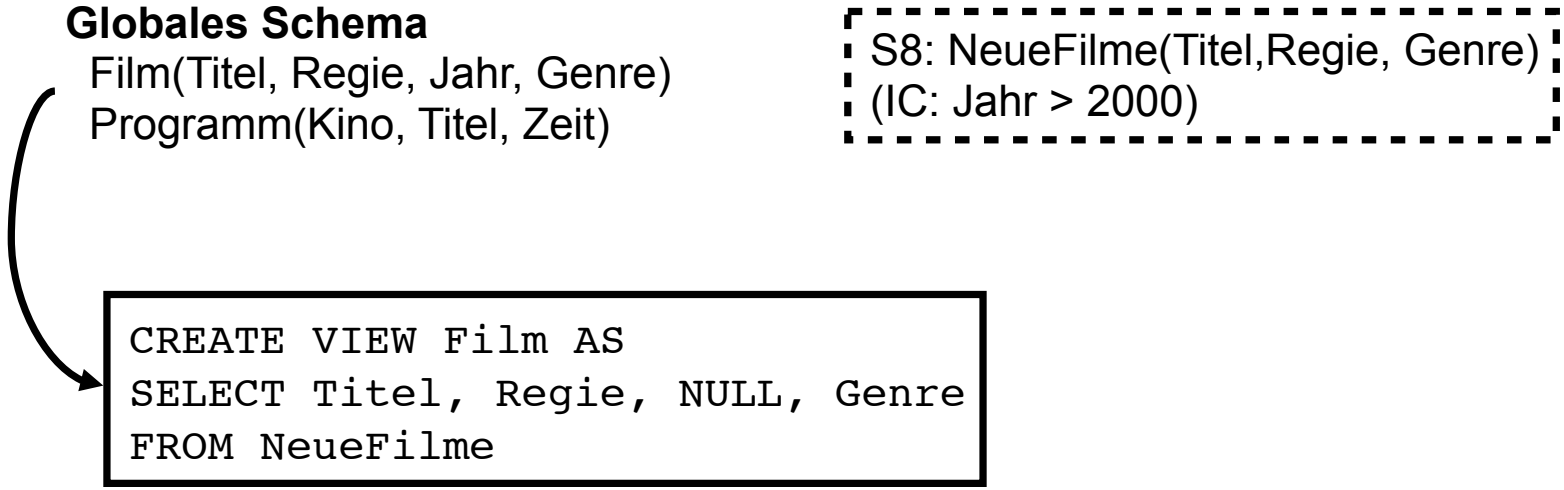
Globales Schema
NeuerFilm(Titel, Regie, Jahr, Genre)
Nebenbedingung: Jahr > 2000
oder
NeuerFilm(Titel, Regie, Genre)
Nebenbedingung: Jahr > 2000

Lokale Schemas
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
```

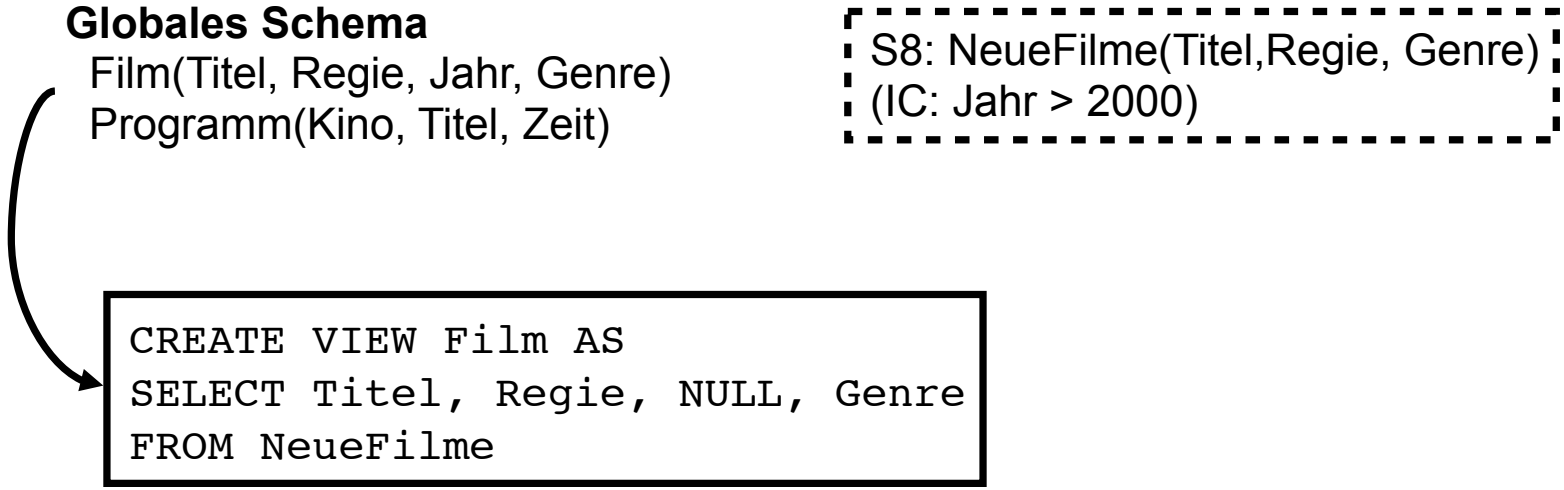
Quelle S2 kann nicht modelliert werden!

Global as View - Lokale ICs



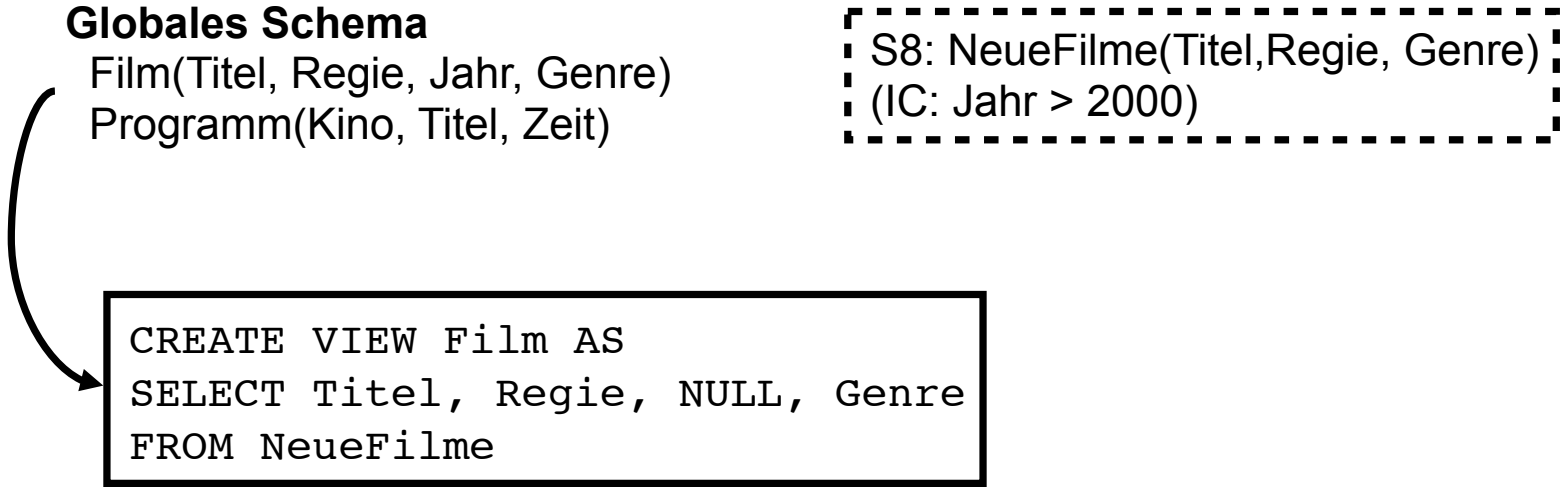
- Bekannte Nebenbedingung auf der Quelle kann nicht im Mapping modelliert werden

Global as View - Lokale ICs



- Bekannte Nebenbedingung auf der Quelle kann nicht im Mapping modelliert werden
- Warum ist das schlecht?

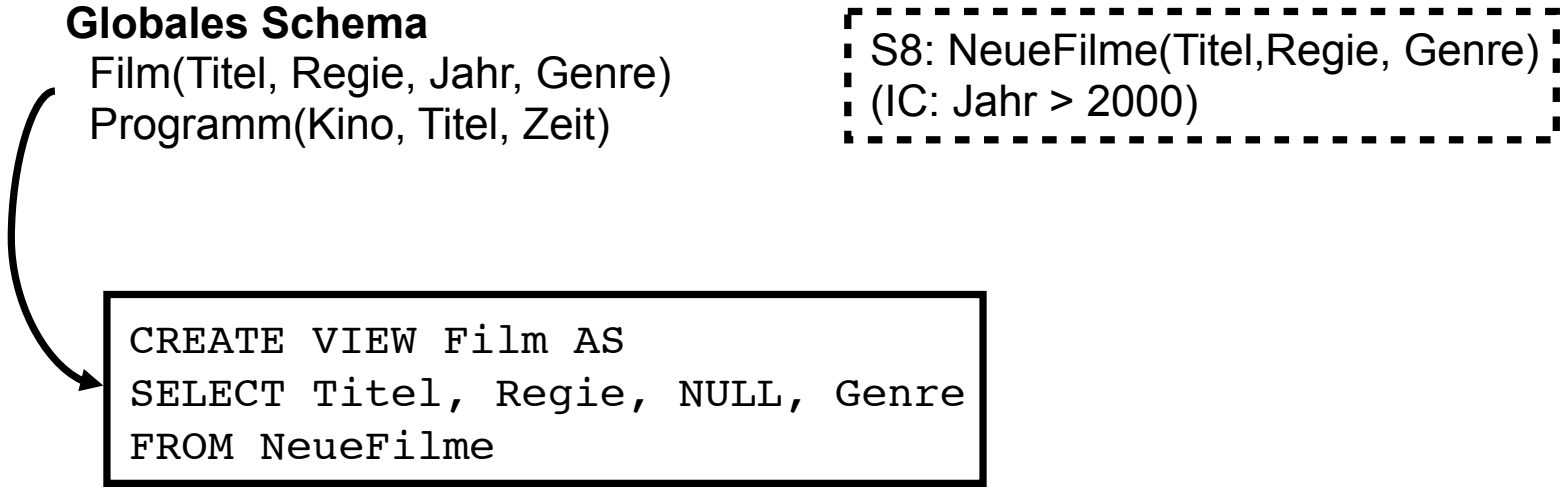
Global as View - Lokale ICs



- Bekannte Nebenbedingung auf der Quelle kann nicht im Mapping modelliert werden
- Warum ist das schlecht?

  `SELECT * FROM Film WHERE Jahr < 1950` (unnötiger Table Scan)

Global as View - Lokale ICs

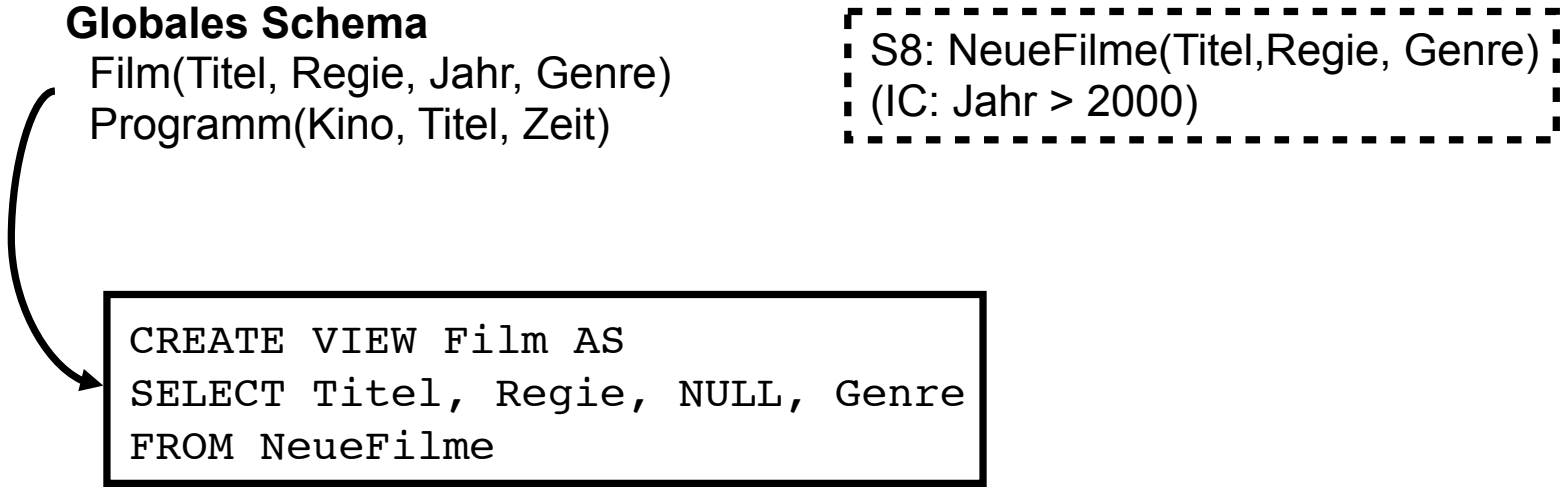


- Bekannte Nebenbedingung auf der Quelle kann nicht im Mapping modelliert werden
- Warum ist das schlecht?

  `SELECT * FROM Film WHERE Jahr < 1950` (unnötiger Table Scan)

  `SELECT Titel FROM Film WHERE Jahr > 1950` (vertane Chance)

Global as View - Lokale ICs



- Bekannte Nebenbedingung auf der Quelle kann nicht im Mapping modelliert werden
- Warum ist das schlecht?

  `SELECT * FROM Film WHERE Jahr < 1950` (unnötiger Table Scan)

  `SELECT Titel FROM Film WHERE Jahr > 1950` (vertane Chance)
- Evtl. separate Modellierung möglich (Frage: Kann der Planungsalgorithmus diese separat modellierten Informationen nutzen?)

Global as View - Lokale ICs

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)

Global as View - Lokale ICs

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 lokale Anfrage sinnlos)

Global as View - Lokale ICs

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 lokale Anfrage sinnlos)
- Aber sie hilft dem Planungsalgorithmus bei Anfragen wie

Global as View - Lokale ICs

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 lokale Anfrage sinnlos)
- Aber sie hilft dem Planungsalgorithmus bei Anfragen wie

  `SELECT * FROM Film WHERE Jahr < 1950` (Quelle weglassen)

Global as View - Lokale ICs

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 lokale Anfrage sinnlos)
- Aber sie hilft dem Planungsalgorithmus bei Anfragen wie

  `SELECT * FROM Film WHERE Jahr < 1950` (Quelle weglassen)

  `SELECT Titel FROM Film WHERE Jahr > 1950` (keine Selektion)

Global as View - Lokale ICs

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Lokale Schemas

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)

```
CREATE VIEW Film AS
SELECT *
FROM AlleFilmeNett
        UNION
SELECT Titel, Regie, NULL, Genre
FROM AlleFilmeBöse
        UNION
SELECT *
FROM NeueFilmeNett
(WHERE Jahr > 2000)
        UNION
SELECT Titel, Regie, NULL, Genre
FROM NeueFilmeBöse
        UNION
SELECT Titel, Regie, 2004, Genre
FROM AktuelleFilme
```

Global as View - Lokale ICs

Globales Schema

Film(Titel, Regie, Jahr, Genre)

Lokale Schemas

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 der IC von S3 nur für Optimierung

```
CREATE VIEW Film AS
SELECT *
FROM AlleFilmeNett
        UNION
SELECT Titel, Regie, NULL, Genre
FROM AlleFilmeBöse
        UNION
SELECT *
FROM NeueFilmeNett
(WHERE Jahr > 2000)
        UNION
SELECT Titel, Regie, NULL, Genre
FROM NeueFilmeBöse
        UNION
SELECT Titel, Regie, 2004, Genre
FROM AktuelleFilme
```

Jede Quelle kann modelliert werden!

IC von S4 kann nicht modelliert werden!

Jede Sicht exportiert alle Daten der Quelle!

Global as View - Lokale ICs

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

Anfrage: `SELECT * FROM Film`

```
CREATE VIEW Film AS
SELECT *
FROM AlleFilmeNett
       UNION
SELECT Titel, Regie, NULL, Genre
FROM AlleFilmeBöse
       UNION
SELECT *
FROM NeueFilmeNett
WHERE Jahr > 2000
       UNION
SELECT Titel, Regie, NULL, Genre
FROM NeueFilmeBöse
       UNION
SELECT Titel, Regie, 2004, Genre
FROM AktuelleFilme
```

```
SELECT * FROM
    (SELECT *
    FROM AlleFilmeNett
               UNION
    SELECT Titel, Regie, NULL, Genre
    FROM AlleFilmeBöse
               UNION
    SELECT *
    FROM NeueFilmeNett
    WHERE Jahr > 2000
               UNION
    SELECT Titel, Regie, NULL, Genre
    FROM NeueFilmeBöse
               UNION
    SELECT Titel, Regie, 2004, Genre
    FROM AktuelleFilme)
```

Global as View - Lokale ICs

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

Anfrage:

```
SELECT * FROM Film
WHERE Jahr > 2001
```

```
SELECT * FROM
    (SELECT *
    FROM AlleFilmeNett
                UNION
    SELECT Titel, Regie, NULL, Genre
    FROM AlleFilmeBöse
                UNION
    SELECT *
    FROM NeueFilmeNett
    (WHERE Jahr > 2000)
                UNION
    SELECT Titel, Regie, NULL, Genre
    FROM NeueFilmeBöse
                UNION
    SELECT Titel, Regie, 2004, Genre
    FROM AktuelleFilme)
WHERE Jahr > 2001
```

Trägt voll zum Ergebnis bei

Trägt nicht zum Ergebnis bei, obwohl nützlich

Trägt voll zum Ergebnis bei

Trägt nicht zum Ergebnis bei, obwohl nützlich

Trägt voll zum Ergebnis bei

Global as View - Lokale ICs

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

Anfrage:

```
SELECT * FROM Film
WHERE Jahr = 2001
```

```
SELECT * FROM
    (SELECT *
    FROM AlleFilmeNett
                UNION
    SELECT Titel, Regie, NULL, Genre
    FROM AlleFilmeBöse
                UNION
    SELECT *
    FROM NeueFilmeNett
    WHERE Jahr > 2000
                UNION
    SELECT Titel, Regie, NULL, Genre
    FROM NeueFilmeBöse
                UNION
    SELECT Titel, Regie, 2004, Genre
    FROM AktuelleFilme)
WHERE Jahr = 2001
```

Trägt voll zum Ergebnis bei

Trägt nicht zum Ergebnis bei, obwohl nützlich

Trägt voll zum Ergebnis bei

Trägt nicht zum Ergebnis bei, obwohl nützlich

Trägt nicht zum Ergebnis bei (2001 ≠ 2004)

Global as View - Lokale ICs

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

```
SELECT * FROM
    (SELECT *
    FROM AlleFilmeNett
                UNION
    SELECT Titel, Regie, NULL, Genre
    FROM AlleFilmeBöse
                UNION
    SELECT *
    FROM NeueFilmeNett
    WHERE Jahr > 2000
                UNION
    SELECT Titel, Regie, NULL, Genre
    FROM NeueFilmeBöse
                UNION
    SELECT Titel, Regie, 2004, Genre
    FROM AktuelleFilme)
WHERE Jahr < 1950
```

Anfrage:

```
SELECT * FROM Film
WHERE Jahr < 1950
```

Beiträge zum Ergebnis

Global as View - Globale und Lokale ICs

Globales Schema
NeuerFilm(Titel, Regie, Genre)
Nebenbedingung: Jahr > 2000

Lokale Schemas
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)

Quelle S2 kann nicht modelliert werden!

```
CREATE VIEW NeuerFilm AS
SELECT Titel, Regie, Genre
FROM AlleFilmeNett
WHERE Jahr > 2000
       UNION
???
       UNION
SELECT Titel, Regie, Genre
FROM NeueFilmeNett
(WHERE Jahr > 2000)
       UNION
SELECT Titel, Regie, Genre
FROM NeueFilmeBöse
       UNION
SELECT Titel, Regie, Genre
FROM AktuelleFilme
```

Global as View - Globale und Lokale ICs

Globales Schema
NeuerFilm(Titel, Regie, Genre)
Nebenbedingung: Jahr > 2000

Lokale Schemas
S1: AlleFilmeNett(Titel, Regie, J
S2: AlleFilmeBöse(Titel, Regie,
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)

Lokale ICs Jahr > 2000 und Jahr=2004

- sind beide durch globale IC abgedeckt
  (=> wenn sie erfüllt sind, ist es die globale auch)
- S4 und S5 können daher genutzt werden auch wenn sie das Attribut Jahr nicht bereitstellen

```
        UNION
SELECT Titel, Regie, Genre
FROM NeueFilmeNett
(WHERE Jahr > 2000)
        UNION
SELECT Titel, Regie, Genre
FROM NeueFilmeBöse
        UNION
SELECT Titel, Regie, Genre
FROM AktuelleFilme
```

Quelle S2 kann nicht modelliert werden!

Global as View - ICs

Fazit:

Global as View - ICs

Fazit:

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

Global as View - ICs

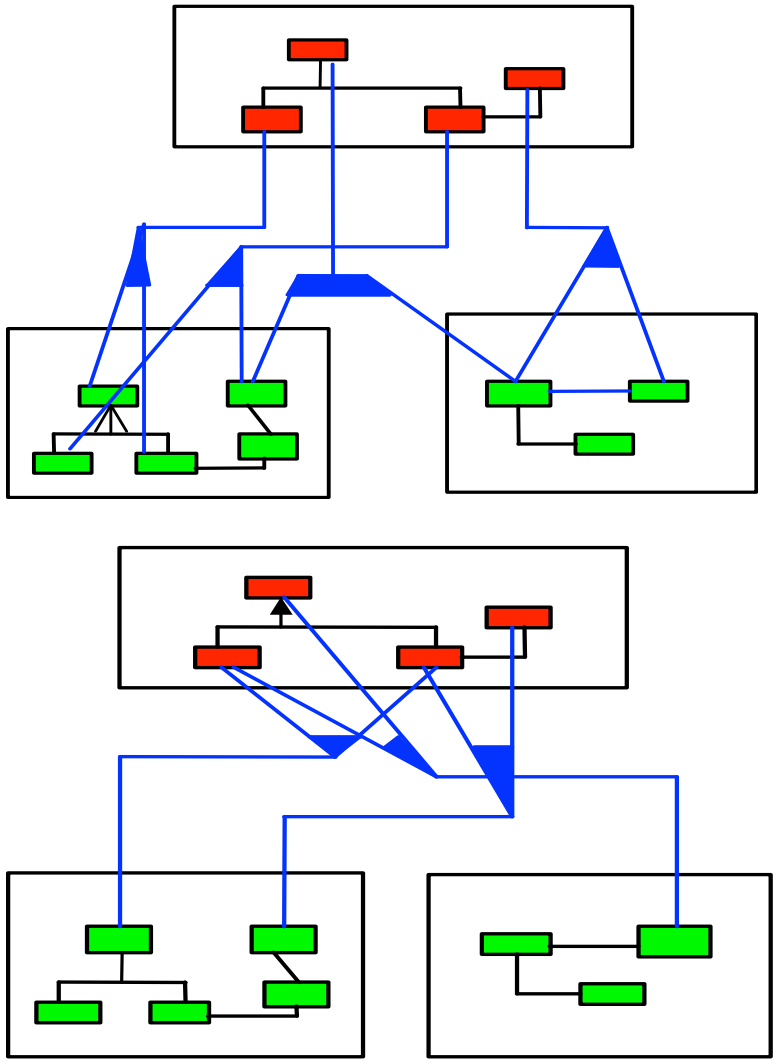
Fazit:

- Nebenbedingungen (ICs) im globalen Schema können die Integration von Quellen verhindern, wenn die Bedingung nicht geprüft werden kann (fehlendes Attribut).
- Nebenbedingungen in den lokalen Schemas können in Mapping modelliert werden,
  - (wenn sie auf exportierten Attributen gelten, oder)
  - wenn sie auf globalen Attributen gelten und die Form `attribute = <constant>` haben.

Agenda

Global as View / Local as View

- 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.

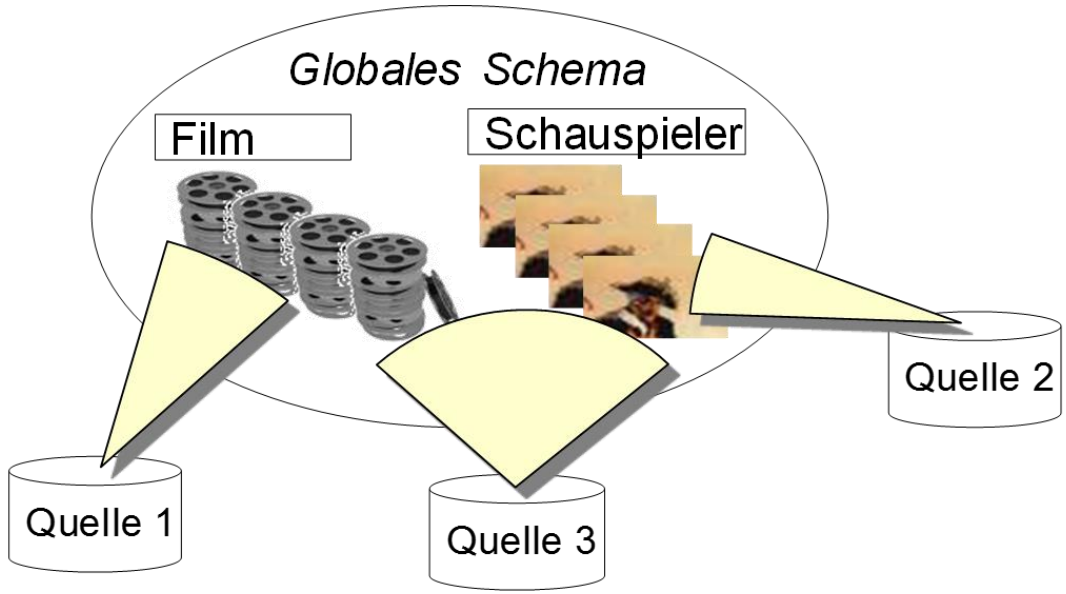


Warum Local as View?

Andere Sichtweise:

- Es gibt in der Welt eine Menge von Filmen, Schauspielern, ...
- Das globale Schema modelliert diese Welt
- Theoretisch steht damit die 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 (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 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 (Beispiel)

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 (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
```

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

Local as View (Beispiel)

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

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

```
CREATE VIEW S9_Filme AS
SELECT Titel, Jahr, NULL, NULL
FROM Film

CREATE VIEW S9_Regie AS
SELECT NULL, Regie AS Regisseur
FROM Film
```

Assoziationen des lokalen Schemas können **nicht abgebildet** werden.

Local as View - Globale ICs

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)
```

Local as View - Globale ICs

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 nicht sinnvoll modelliert werden
  (Quellen können ältere Filme beinhalten)
- Das ging aber bei GaV (beides hat Stärken und Schwächen)

Local as View - Globale ICs

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 nicht sinnvoll modelliert werden
  (Quellen können ältere Filme beinhalten)
- Das ging aber bei GaV (beides hat Stärken und Schwächen)
- Allerdings gelten ICs für die Gesamtsicht (alle Anfragen)
  $\Rightarrow$ nachträgliche Selektion im Mediator

Local as View - Lokale ICs

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
```

Local as View - Lokale ICs

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

Local as View - Lokale ICs

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)

```
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
```

Modellierung zur Optimierung

Modellierung zur Beantwortbarkeit (S4 & S5)

Local as View - Lokale ICs

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

S1: AlleFilmeNett(Titel, Regie,
S2: AlleFilmeBöse(Titel, Regie
S3: NeueFilmeNett(Titel, Regi
(Nebenbedingung: Ja

S4: NeueFilmeBöse(Titel, Regie, Genre)
(Nebenbedingung: Jahr > 2000)

S5: AktuelleFilme(Titel, Regie, Genre)
(Nebenbedingung: Jahr = 2004)

Bedingung Jahr > 2000 (bzw. Jahr=2004)
- bei S3 **nur** zur Optimierung
  (Quelle kann auch ohne genutzt werden)
- bei S4 und S5 zur Nutzung **notwendig**

```
CREATE VIEW S1 AS
SELECT * FROM Film

SELECT * FROM Film
WHERE Jahr > 2000

CRE  E VIEW S4 AS
ELECT Titel, Regie, Genre
FROM Film
WHERE Jahr > 2000

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

Modellierung zur Optimierung

Modellierung zur Beantwortbarkeit (S4 & S5)

Agenda

Vergleich GaV / LaV: Modellierung

GaV	LaV
• Jede globale Relation definiert als Sicht auf eine oder mehrere Relationen aus einer oder mehreren Quellen • Meist UNION über mehrere Quellen • Nebenbedingungen auf lokalen Quellen können nicht modelliert werden	• Globale Relationen werden durch mehrere Sichten definiert • Definition oft nur in Kombination mit anderen globalen Relationen • Nebenbedingungen auf globale Relationen können nicht definiert werden

Global Local as View (GLaV)

Global Local as View (GLaV)

- Die Ausdrucksmächtigkeit von GaV und LaV ist begrenzt
  - GaV kann (ohne Skolemfunktionen) kein lokales Tupel auf zwei globale Tupel splitten
  - LaV kann keine zwei lokalen Tupel zu einem ein globalen Tupel joinen
  - Nebenbedingungen können teilweise nicht definiert werden

Global Local as View (GLaV)

- Die Ausdrucksmächtigkeit von GaV und LaV ist begrenzt
  - GaV kann (ohne Skolemfunktionen) kein lokales Tupel auf zwei globale Tupel splitten
  - LaV kann keine zwei lokalen Tupel zu einem ein globalen Tupel joinen
  - Nebenbedingungen können teilweise nicht definiert werden
- Daher Entwurf von GLaV Mappings

Global Local as View (GLaV)

- Die Ausdrucksmächtigkeit von GaV und LaV ist begrenzt
  - GaV kann (ohne Skolemfunktionen) kein lokales Tupel auf zwei globale Tupel splitten
  - LaV kann keine zwei lokalen Tupel zu einem ein globalen Tupel joinen
  - Nebenbedingungen können teilweise nicht definiert werden
- Daher Entwurf von GLaV Mappings
- GLaV kombiniert die Eigenschaften von GaV und LaV

Global Local as View (GLaV)

- Die Ausdrucksmächtigkeit von GaV und LaV ist begrenzt
  - GaV kann (ohne Skolemfunktionen) kein lokales Tupel auf zwei globale Tupel splitten
  - LaV kann keine zwei lokalen Tupel zu einem ein globalen Tupel joinen
  - Nebenbedingungen können teilweise nicht definiert werden
- Daher Entwurf von GLaV Mappings
- GLaV kombiniert die Eigenschaften von GaV und LaV
- GLaV setzt eine Anfrage auf das globale Schema und eine Anfrage auf das lokale Schema zueinander in Beziehung

Global Local as View (GLaV)

- Die Ausdrucksmächtigkeit von GaV und LaV ist begrenzt
  - GaV kann (ohne Skolemfunktionen) kein lokales Tupel auf zwei globale Tupel splitten
  - LaV kann keine zwei lokalen Tupel zu einem ein globalen Tupel joinen
  - Nebenbedingungen können teilweise nicht definiert werden
- Daher Entwurf von GLaV Mappings
- GLaV kombiniert die Eigenschaften von GaV und LaV
- GLaV setzt eine Anfrage auf das globale Schema und eine Anfrage auf das lokale Schema zueinander in Beziehung
- nicht darstellbar mit SQL View Definitionen

  $\Rightarrow$ Beschreibung durch Datalog

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN
FROM   sales S, time T, ...
WHERE  S.day_id = T.day_id AND
       S.product_id = P.product_id AND
       S.shop_id = L.shop_id AND
       L.shop_id = 123 AND
       T.year > 1999
```

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN
FROM   sales S, time T, ...
WHERE  S.day_id = T.day_id AND
       S.product_id = P.product_id AND
       S.shop_id = L.shop_id AND
       L.shop_id = 123 AND
       T.year > 1999
```

```
q(P,RN) :-
  sales(SID,PID,TID,RID,P,...),
  time(TID,D,M,Y),
  localization(SID,LID,SN,RN),
  product(PID,PN,PGN),
  Y > 1999, SID = 123
```

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN        q(P,RN) :-
FROM   sales S, time T, ...                  sales(SID,PID,TID,RID,P,...),
WHERE  S.day_id = T.day_id AND               time(TID,D,M,Y),
       S.product_id = P.product_id AND       localization(SID,LID,SN,RN),
       S.shop_id = L.shop_id AND             product(PID,PN,PGN),
       L.shop_id = 123 AND                   Y > 1999, SID = 123
       T.year > 1999
```

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN        q(P,RN) :-
FROM   sales S, time T, ...               sales(SID,PID,TID,RID,P,...),
WHERE  S.day_id = T.day_id AND            time(TID,D,M,Y),
       S.product_id = P.product_id AND    localization(SID,LID,SN,RN),
       S.shop_id = L.shop_id AND          product(PID,PN,PGN),
       L.shop_id = 123 AND                Y > 1999, SID = 123
       T.year > 1999
```

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN          q(P,RN) :-
FROM   sales S, time T, ...                 sales(SID,PID,TID,RID,P,...),
WHERE  S.day_id = T.day_id AND              time(TID,D,M,Y),
       S.product_id = P.product_id AND      localization(SID,LID,SN,RN),
       S.shop_id = L.shop_id AND            product(PID,PN,PGN,
       L.shop_id = 123 AND                  Y > 1999, SID = 123
       T.year > 1999
```

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gleiche Attributnamen

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN          q(P,RN) :-
FROM   sales S, time T, ...                 sales(SID,PID,TID,RID,P,...),
WHERE  S.day_id = T.day_id AND              time(TID,D,M,Y),
       S.product_id = P.product_id AND      localization(SID,LID,SN,RN),
       S.shop_id = L.shop_id AND            product(PID,PN,PGN),
       L.shop_id = 123 AND                  Y > 1999, SID = 123
       T.year > 1999
```

DATALOG Notation

- Einschränkung auf konjunktive Anfragen:
  - nur Equijoins und Bedingungen mit $=$, $<$,$>$ zwischen Attribut und Konstanten (*semi-interval constraints*)
  - kein `NOT`, `EXISTS`, `GROUP BY`, $\neq$, $X > Y$
- Schreibweise DATALOG
  - `SELECT`-Klausel: Regelkopf, exportierte Attribute
  - `FROM`-Klausel: Relationen werden zu Prädikaten (sog. Subgoals)
  - `WHERE`-Klausel: Joins durch gle

gebundene Variablen (z.B. SID=123) können auch als Konstanten in die Prädikate eingefügt werden (z.B. localization(123,LID,...))

SQL vs. Datalog Notation

```
SELECT S.price P, L.region_name RN
FROM   sales S, time T, ...
WHERE  S.day_id = T.day_id AND
       S.product_id = P.product_id AND
       S.shop_id = L.shop_id AND
       L.shop_id = 123 AND
       T.year > 1999
```

```
time(TID,D,M,Y),
localization(SID,LID,SN,RN),
product(PID,PN,PGN),
Y > 1999, SID = 123
```

DATALOG Notation

Disjunktive Anfragen durch mehrere Regeln mit gleichen Kopf

DATALOG Notation

Disjunktive Anfragen durch mehrere Regeln mit gleichen Kopf

SQL vs. Datalog Notation

```
SELECT S1.price P1                    q(P) :-  sales1(SID,P1,...)
FROM   sales1 S1                      q(P) :-  sales2(SNR,...,P2)
UNION
SELECT S2.price P2
FROM   sales2 S2
```

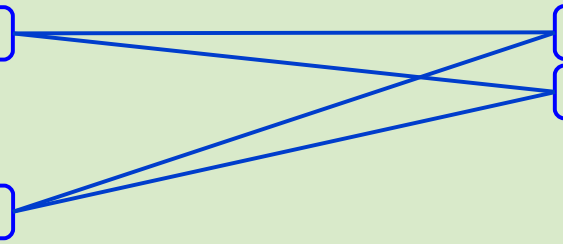
DATALOG Notation

Disjunktive Anfragen durch mehrere Regeln mit gleichen Kopf

SQL vs. Datalog Notation

```
SELECT S1.price P1
FROM   sales1 S1
UNION
SELECT S2.price P2
FROM   sales2 S2
```

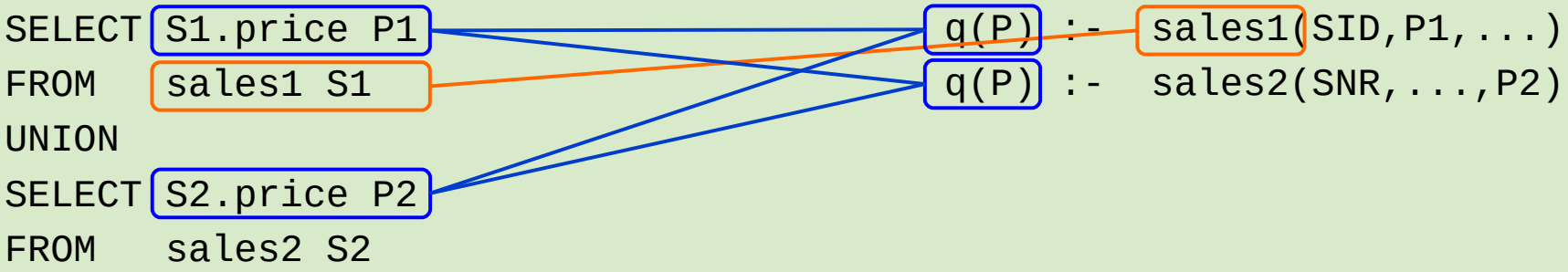
```
q(P) :- sales1(SID,P1,...)
q(P) :- sales2(SNR,...,P2)
```



DATALOG Notation

Disjunktive Anfragen durch mehrere Regeln mit gleichen Kopf

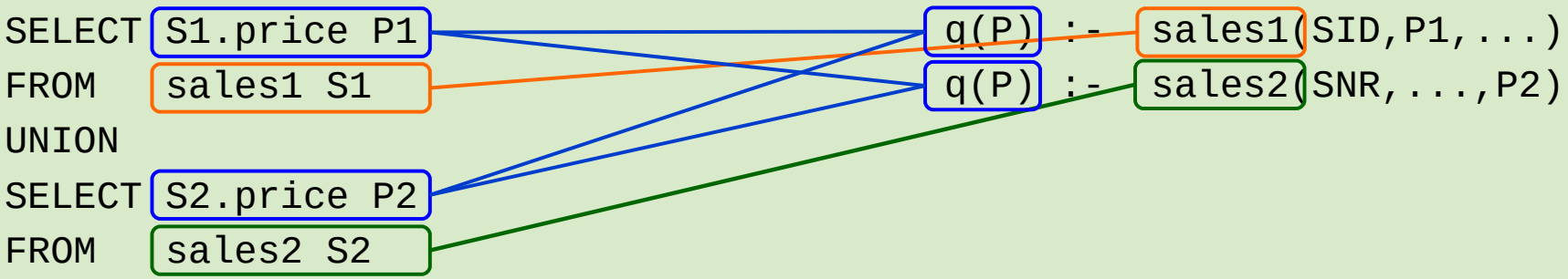
SQL vs. Datalog Notation



DATALOG Notation

Disjunktive Anfragen durch mehrere Regeln mit gleichen Kopf

SQL vs. Datalog Notation



Formale Definition von GaV und LaV [DHI12]

Formale Definition von GaV und LaV [DHI12]

Definition (GaV Schema Mappings): Sei G ein globales Schema und seien $\overline{S} = \{S_1, \ldots, S_n\}$ die Schemata von n Datenquellen. Ein Global-as-View schema mapping M ist eine Menge an Ausdrücken der Form $G_i(\overline{X}) \supseteq Q_i(\overline{S})$ oder $G_i(\overline{X}) = Q_i(\overline{S})$, für die gilt:

- G_i ist eine Relation in G und erscheint in maximal einem Ausdruck von M, und
- Q_i ist eine Anfrage über die Relationen in $\overline{S}$

Formale Definition von GaV und LaV [DHI12]

Definition (GaV Schema Mappings): Sei G ein globales Schema und seien $\overline{S} = \{S_1, \ldots, S_n\}$ die Schemata von n Datenquellen. Ein Global-as-View schema mapping M ist eine Menge an Ausdrücken der Form $G_i(\overline{X}) \supseteq Q_i(\overline{S})$ oder $G_i(\overline{X}) = Q_i(\overline{S})$, für die gilt:

- G_i ist eine Relation in G und erscheint in maximal einem Ausdruck von M, und
- Q_i ist eine Anfrage über die Relationen in $\overline{S}$

Definition (LaV Schema Mappings): Sei G ein globales Schema und seien $\overline{S} = \{S_1, \ldots, S_n\}$ die Schemata von n Datenquellen. Ein Local-as-View schema mapping M ist eine Menge an Ausdrücken der Form $S_i(\overline{X}) \subseteq Q_i(G)$ oder $S_i(\overline{X}) = Q_i(G)$, für die gilt:

- Q_i ist eine Anfrage über das globale Schema G, und
- S_i ist eine Quellrelation und erscheint in maximal einem Ausdruck von M

Formale Definition von GLaV [DHI12]

Formale Definition von GLaV [DHI12]

Definition (GLaV Schema Mappings): Sei G ein globales Schema und seien $\overline{S} = \{S_1, \ldots, S_n\}$ die Schemata von n Datenquellen. Ein Global-Local-as-View schema mapping M ist eine Menge an Ausdrücken der Form $Q^S(\overline{X}) \subseteq Q^G(\overline{X})$ oder $Q^S(\overline{X}) = Q^G(\overline{X})$, für die gilt:

- Q^G ist eine Anfrage über das globale Schema G dessen Kopfvariablen der Menge $\overline{X}$ entsprechen, und
- Q^S ist eine Anfrage über die Relationen in $\overline{S}$ dessen Kopfvariablen ebenfalls der Menge $\overline{X}$ entsprechen

GLaV (Beispiel)

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

Quellen:
S1: KinoDB(Kino, Genre)
S2: Filme(Titel, Jahr, Ort, RegieID)
Regie(ID, Regisseur)
S3: Kino(KID, Kino)
Genre(KID, Genre)

GLaV (Beispiel)

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

Quellen:
S1: KinoDB(Kino, Genre)
S2: Filme(Titel, Jahr, Ort, RegieID)
Regie(ID, Regisseur)
S3: Kino(KID, Kino)
Genre(KID, Genre)

- Beispiel GaV (und somit auch GLaV):
  Filme(t,j,o,rid), Regie(rid,r) $\subseteq$ Film(t,r,j,NULL)

GLaV (Beispiel)

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

Quellen:
S1: KinoDB(Kino, Genre)
S2: Filme(Titel, Jahr, Ort, RegieID)
Regie(ID, Regisseur)
S3: Kino(KID, Kino)
Genre(KID, Genre)

- Beispiel GaV (und somit auch GLaV):
  Filme(t,j,o,rid), Regie(rid,r) $\subseteq$ Film(t,r,j,NULL)
- Beispiel LaV (und somit auch GLaV):
  KinoDB(k,g) $\subseteq$ Film(t,r,j,g), Prog(k,t,z)

GLaV (Beispiel)

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

Quellen:
S1: KinoDB(Kino, Genre)
S2: Filme(Titel, Jahr, Ort, RegieID)
Regie(ID, Regisseur)
S3: Kino(KID, Kino)
Genre(KID, Genre)

- Beispiel GaV (und somit auch GLaV):
  Filme(t,j,o,rid), Regie(rid,r) $\subseteq$ Film(t,r,j,NULL)
- Beispiel LaV (und somit auch GLaV):
  KinoDB(k,g) $\subseteq$ Film(t,r,j,g), Prog(k,t,z)
- Beispiel GLaV:
  Kino(kid,k), Genre(kid,g) -: Q^S(k,g) $\subseteq$ Q^G(k,g) :- Film(t,r,j,g), Prog(k,t,z)
  (Die Köpfe der beiden Anfragen Q^S and Q^G werden für gewöhnlich weggelassen)

Zugriffsbeschränkungen

Agenda

Zugriffsbeschränkungen

Gründe:

Zugriffsbeschränkungen

Gründe:

- Quelle unterstützt nur eine bestimmte Klasse an Anfragen

Zugriffsbeschränkungen

Gründe:

- Quelle unterstützt nur eine bestimmte Klasse an Anfragen
- Einige der Variablen der Anfrage müssen gebunden sein

Zugriffsbeschränkungen

Gründe:

- Quelle unterstützt nur eine bestimmte Klasse an Anfragen
- Einige der Variablen der Anfrage müssen gebunden sein

Auswirkungen:

Zugriffsbeschränkungen

Gründe:

- Quelle unterstützt nur eine bestimmte Klasse an Anfragen
- Einige der Variablen der Anfrage müssen gebunden sein

Auswirkungen:

- Die Reihenfolge in der die Quellen angefragt werden müssen ist nicht beliebig

Zugriffsbeschränkungen

Gründe:

- Quelle unterstützt nur eine bestimmte Klasse an Anfragen
- Einige der Variablen der Anfrage müssen gebunden sein

Auswirkungen:

- Die Reihenfolge in der die Quellen angefragt werden müssen ist nicht beliebig
- einige Anfrageoperationen (z.B. Join, Selektion) müssen aus der Quelle ausgelagert werden

Modellierung von Zugriffsbeschränkungen

Verwendung von sogenannten *Adornments*

Modellierung von Zugriffsbeschränkungen

Verwendung von sogenannten *Adornments*

- Ein Adornment pro Quellrelation

Modellierung von Zugriffsbeschränkungen

Verwendung von sogenannten *Adornments*

- Ein Adornment pro Quellrelation
- Ein Adornment ist eine Folge der Buchstaben
  - 'b' (bounded)
  - 'f' (free)

  wobei die Länge der Folge der Anzahl der Attribute der betreffenden Relation entspricht ($\Rightarrow$ ein Buchstabe per Attribut)

Modellierung von Zugriffsbeschränkungen

Verwendung von sogenannten *Adornments*

- Ein Adornment pro Quellrelation
- Ein Adornment ist eine Folge der Buchstaben
  - 'b' (bounded)
  - 'f' (free)

  wobei die Länge der Folge der Anzahl der Attribute der betreffenden Relation entspricht ($\Rightarrow$ ein Buchstabe per Attribut)
- Sind mehrere Kombinationen von gebundenen und freien Variablen möglich enthält die Quellbeschreibung mehrere Adornments für die betreffende Quellrelation

Beispiel [DHI12]

Globales Schema:

$$\boxed{\begin{array}{l} \mathit{Cites}(X, Y) \\ \mathit{DBPapers}(X) \\ \mathit{AwardPaper}(X) \end{array}}$$

Beispiel [DHI12]

Globales Schema:

$$\boxed{\begin{array}{l} Cites(X, Y) \\ DBPapers(X) \\ AwardPaper(X) \end{array}}$$

Quellschemata:

$$\begin{aligned} S1: &\quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y) \\ S2: &\quad CitingPapers^{f}(X) \subseteq Cites(X, Y) \\ S3: &\quad DBSource^{f}(X) \subseteq DBPapers(X) \\ S4: &\quad AwardDB^{b}(X) \subseteq AwardPaper(X) \end{aligned}$$

Ausführbarer Anfrageplan

Gegeben:

Ausführbarer Anfrageplan

Gegeben:

- konjunktiver Anfrageplan $g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals

Ausführbarer Anfrageplan

Gegeben:

- konjunktiver Anfrageplan $g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals
- Adornmentmenge BF_i für jedes Subgoal g_i

Ausführbarer Anfrageplan

Gegeben:

- konjunktiver Anfrageplan $g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals
- Adornmentmenge BF_i für jedes Subgoal g_i

Der Anfrageplan ist ausführbar, wenn es eine Zusammenstellung von Adornments $\{bf_1, \ldots, bf_n\}$ gibt so dass gilt:

Ausführbarer Anfrageplan

Gegeben:

- konjunktiver Anfrageplan $g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals
- Adornmentmenge BF_i für jedes Subgoal g_i

Der Anfrageplan ist ausführbar, wenn es eine Zusammenstellung von Adornments $\{bf_1, \ldots, bf_n\}$ gibt so dass gilt:

- $bf_i \in BF_i$ für jedes $i \in \{1, \ldots, n\}$

Ausführbarer Anfrageplan

Gegeben:

- konjunktiver Anfrageplan $g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals
- Adornmentmenge BF_i für jedes Subgoal g_i

Der Anfrageplan ist ausführbar, wenn es eine Zusammenstellung von Adornments $\{bf_1, \ldots, bf_n\}$ gibt so dass gilt:

- $bf_i \in BF_i$ für jedes $i \in \{1, \ldots, n\}$
- wenn die freie Variable X an Position k in $g_i(\overline{X}_i)$ erscheint und bf_i hat ein 'b' an Position k, dann erscheint X in einem Subgoal $g_j(\overline{X}_j)$ mit $j < i$

Ausführbarer Anfrageplan

Gegeben:

- konjunktiver Anfrageplan $g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals
- Adornmentmenge BF_i für jedes Subgoal g_i

Der Anfrageplan ist ausführbar, wenn es eine Zusammenstellung von Adornments $\{bf_1, \ldots, bf_n\}$ gibt so dass gilt:

- $bf_i \in BF_i$ für jedes $i \in \{1, \ldots, n\}$
- wenn die freie Variable X an Position k in $g_i(\overline{X}_i)$ erscheint und bf_i hat ein 'b' an Position k, dann erscheint X in einem Subgoal $g_j(\overline{X}_j)$ mit $j < i$

Jede Variable die in einem Subgoal gebunden sein muss ist entweder durch die Anfrage vorgegeben oder muss durch ein früheres Subgoal berechnet werden

Auffinden eines ausführbaren Planes [DHI12]

Gegeben: Logischer Anfrageplan $Q = g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals, Adornmentmenge BF_i für jedes Subgoal g_i

Auffinden eines ausführbaren Planes [DHI12]

Gegeben: Logischer Anfrageplan $Q = g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals, Adornmentmenge BF_i für jedes Subgoal g_i

Auffinden eines ausführbaren Planes [DHI12]

Gegeben: Logischer Anfrageplan $Q = g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals, Adornmentmenge BF_i für jedes Subgoal g_i

- Erzeuge einen leeren Anfrageplan EP

Auffinden eines ausführbaren Planes [DHI12]

Gegeben: Logischer Anfrageplan $Q = g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals, Adornmentmenge BF_i für jedes Subgoal g_i

- Erzeuge einen leeren Anfrageplan EP
- Initialisiere $AD_i = BF_i$ für jedes $i \in \{1, \ldots, n\}$

Auffinden eines ausführbaren Planes [DHI12]

Gegeben: Logischer Anfrageplan $Q = g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals, Adornmentmenge BF_i für jedes Subgoal g_i

- Erzeuge einen leeren Anfrageplan EP
- Initialisiere $AD_i = BF_i$ für jedes $i \in \{1, \ldots, n\}$
- Wiederhole folgende Schritte solange ein neues Subgoal zu EP hinzugefügt werden kann
  1. Wähle ein Subgoal $g_i(\overline{X}_i) \notin EP$ welches ein Adornment $ad \in AD_i$ mit $ad =$ 'ff...f' besitzt
  2. Hänge $g_i(\overline{X}_i)$ ans Ende von EP
  3. Wenn $X \in \overline{X}_i$ an der $k$ten Position eines Subgoals $g_j(\overline{X}_j)$ existiert, setzte die $k$te Position aller Adornments in AD_j auf 'f'

Auffinden eines ausführbaren Planes [DHI12]

Gegeben: Logischer Anfrageplan $Q = g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$ mit n Subgoals, Adornmentmenge BF_i für jedes Subgoal g_i

- Erzeuge einen leeren Anfrageplan EP
- Initialisiere $AD_i = BF_i$ für jedes $i \in \{1, \ldots, n\}$
- Wiederhole folgende Schritte solange ein neues Subgoal zu EP hinzugefügt werden kann
  1. Wähle ein Subgoal $g_i(\overline{X}_i) \notin EP$ welches ein Adornment $ad \in AD_i$ mit $ad =$ 'ff...f' besitzt
  2. Hänge $g_i(\overline{X}_i)$ ans Ende von EP
  3. Wenn $X \in \overline{X}_i$ an der $k$ten Position eines Subgoals $g_j(\overline{X}_j)$ existiert, setzte die $k$te Position aller Adornments in AD_j auf 'f'
- Wenn EP alle n Subgoals enthält
  $\Rightarrow$ EP ist ein ausführbarer Plan für Q
- Andernfalls existiert kein ausführbarer Plan für Q

Auffinden eines ausführbaren Planes

Anmerkung:

- Algorithmus berücksichtigt keine durch die Anfrage gebundenen Variablen, z.B.

$$NeuerFilm(T, R, G) \text{ :- } Film(T, R, G, Y), Y = 2015$$

- Erweiterung des Algorithmus indem man die Mengen $AD_1, \ldots, AD_n$ so initiallisiert, dass alle Adornment Positionen welche eine, durch die Anfrage gebundenen, Variable repräsentieren auf 'f' gesetzt werden

Ändern von Anfrageplänen

Ändern von Anfrageplänen

- Algorithmus findet (wenn möglich) einen ausführbaren Anfrageplan mit gleichen Subgoals indem er die Reihenfolge der gegebenen Subgoals ändert

Ändern von Anfrageplänen

- Algorithmus findet (wenn möglich) einen ausführbaren Anfrageplan mit gleichen Subgoals indem er die Reihenfolge der gegebenen Subgoals ändert
- Frage: Wenn es für eine Menge an Subgoals keinen ausführbaren Anfrageplan gibt, kann man einen ergebnisäquivalenten und ausführbaren Plan erzeugen indem man weitere Subgoals hinzufügt?

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Gegebene Anfrage:

$Q(X)$:- $Cites(X, 001)$

Beispiel [DHI12]

$$
\begin{aligned}
S1: &\quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y) \\
S2: &\quad CitingPapers^{f}(X) \subseteq Cites(X, Y) \\
S3: &\quad DBSource^{f}(X) \subseteq DBPapers(X) \\
S4: &\quad AwardDB^{b}(X) \subseteq AwardPaper(X)
\end{aligned}
$$

Gegebene Anfrage:

$Q(X)$:- $Cites(X, 001)$

Nichtausführbarer Anfrageplan:

$Q'(X)$:- $CitationDB(X, 001)$

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Gegebene Anfrage:
$Q(X)$:- $Cites(X, 001)$

Nichtausführbarer Anfrageplan:
$Q'(X)$:- $CitationDB(X, 001)$

Ausführbarer Anfrageplan:
$Q'(X)$:- $CitingPapers(X), CitationDB(X, 001)$

Ändern von Anfrageplänen

Ändern von Anfrageplänen

- Hinzufügen von Subgoals kann also dazu führen, dass ein ausführbarer Anfrageplan gefunden wird

Ändern von Anfrageplänen

- Hinzufügen von Subgoals kann also dazu führen, dass ein ausführbarer Anfrageplan gefunden wird
- Problem: Die Anzahl an (notwendigen) zusätzlichen Subgoals kann unendlich sein

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Beispiel [DHI12]

$$
\begin{aligned}
S1: &\quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y) \\
S2: &\quad CitingPapers^{f}(X) \subseteq Cites(X, Y) \\
S3: &\quad DBSource^{f}(X) \subseteq DBPapers(X) \\
S4: &\quad AwardDB^{b}(X) \subseteq AwardPaper(X)
\end{aligned}
$$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Beispiel [DHI12]

$$
\begin{aligned}
S1: &\quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y) \\
S2: &\quad CitingPapers^{f}(X) \subseteq Cites(X, Y) \\
S3: &\quad DBSource^{f}(X) \subseteq DBPapers(X) \\
S4: &\quad AwardDB^{b}(X) \subseteq AwardPaper(X)
\end{aligned}
$$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Nichtausführbarer Anfrageplan:

$Q'(X)$:- $AwardDB(X)$

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Gegebene Anfrage:
$Q(X)$:- $AwardPaper(X)$

Nichtausführbarer Anfrageplan:
$Q'(X)$:- $AwardDB(X)$

Ausführbarer Anfrageplan:
$Q'(X)$:- $DBSource(X), AwardDB(X)$

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Gegebene Anfrage:
$Q(X)$:- $AwardPaper(X)$

Nichtausführbarer Anfrageplan:
$Q'(X)$:- $AwardDB(X)$

Ausführbarer Anfrageplan:
$Q'(X)$:- $DBSource(X), AwardDB(X)$

Aber wie vollständig ist das Ergebnis?

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Beispiel [DHI12]

$$S1: \quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y)$$
$$S2: \quad CitingPapers^{f}(X) \subseteq Cites(X, Y)$$
$$S3: \quad DBSource^{f}(X) \subseteq DBPapers(X)$$
$$S4: \quad AwardDB^{b}(X) \subseteq AwardPaper(X)$$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Beispiel [DHI12]

$$
\begin{aligned}
S1: &\quad CitationDB^{bf}(X,Y) \subseteq Cites(X,Y) \\
S2: &\quad CitingPapers^{f}(X) \subseteq Cites(X,Y) \\
S3: &\quad DBSource^{f}(X) \subseteq DBPapers(X) \\
S4: &\quad AwardDB^{b}(X) \subseteq AwardPaper(X)
\end{aligned}
$$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Ausführbarer Anfrageplan 2:

$Q'(X)$:- $DBSource(Y), CitationDB(Y,X), AwardDB(X)$

Beispiel [DHI12]

$$
\begin{aligned}
S1: &\quad CitationDB^{bf}(X, Y) \subseteq Cites(X, Y) \\
S2: &\quad CitingPapers^{f}(X) \subseteq Cites(X, Y) \\
S3: &\quad DBSource^{f}(X) \subseteq DBPapers(X) \\
S4: &\quad AwardDB^{b}(X) \subseteq AwardPaper(X)
\end{aligned}
$$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Ausführbarer Anfrageplan 2:

$Q'(X)$:- $DBSource(Y), CitationDB(Y, X), AwardDB(X)$

Ausführbarer Anfrageplan 3:

$Q'(X)$:- $DBSource(Y), CitationDB(Y, X_1), \ldots,$
$CitationDB(X_n, X), AwardDB(X)$

Lösung durch Rekursion

Erstellen einer neuen Relation *Papers*:

$Papers(X)$:- $DBSource(Y)$

$Papers(X)$:- $Papers(Y), CitationDB(Y, X)$

Lösung durch Rekursion

Erstellen einer neuen Relation *Papers*:

$Papers(X)$:- $DBSource(Y)$

$Papers(X)$:- $Papers(Y), CitationDB(Y, X)$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Lösung durch Rekursion

Erstellen einer neuen Relation *Papers*:

$Papers(X)$:- $DBSource(Y)$

$Papers(X)$:- $Papers(Y), CitationDB(Y, X)$

Gegebene Anfrage:

$Q(X)$:- $AwardPaper(X)$

Ausführbarer Anfrageplan:

$Q'(X)$:- $Papers(X), AwardDB(X)$

Vollständigkeit

Agenda

Vollständigkeit

Datenquellen sind oft nur partiell vollständig

Vollständigkeit

Datenquellen sind oft nur partiell vollständig

- Completeness Score: Die Vollständigkeit der gesamten Quelle wird durch einen normierten Wert beschrieben

Vollständigkeit

Datenquellen sind oft nur partiell vollständig

- Completeness Score: Die Vollständigkeit der gesamten Quelle wird durch einen normierten Wert beschrieben
- Local Completeness Descriptions: Beschreiben den Teil der Quelle der vollständig ist

Vollständigkeit

Datenquellen sind oft nur partiell vollständig

- Completeness Score: Die Vollständigkeit der gesamten Quelle wird durch einen normierten Wert beschrieben
- Local Completeness Descriptions: Beschreiben den Teil der Quelle der vollständig ist

  Beispiele:
  - Die Online Filmdatenbank ist nur vollständig in Bezug auf aktuelle Filme
  - Das Vorstellungsverzeichnis ist nur vollständig in Bezug auf Vorstellungen in NYC

Local Completeness Descriptions

Local Completeness Descriptions

Formale Notation: $LC(S, C)$ wobei

- S = Sichtendefinition der Quelle
- C = Constraint der die modelierte Tupelmenge eingrenzt

Local Completeness Descriptions

Formale Notation: $LC(S, C)$ wobei

- S = Sichtendefinition der Quelle
- C = Constraint der die modelierte Tupelmenge eingrenzt

Beispiel mit globalen Schema:

$$S5.MovieGenres(title, genre) \subseteq Movie(title, dir, year, genre)$$
$$S6.MovieDirector(title, dir) \subseteq Movie(title, dir, year, genre)$$
$$S7.MovieYears(title, year) \subseteq Movie(title, dir, year, genre)$$

Local Completeness Descriptions

Formale Notation: $LC(S, C)$ wobei

- S = Sichtendefinition der Quelle
- C = Constraint der die modelierte Tupelmenge eingrenzt

Beispiel mit globalen Schema:

$$S5.MovieGenres(title, genre) \subseteq Movie(title, dir, year, genre)$$
$$S6.MovieDirector(title, dir) \subseteq Movie(title, dir, year, genre)$$
$$S7.MovieYears(title, year) \subseteq Movie(title, dir, year, genre)$$

Mögliche Local Completeness Descriptions:

Local Completeness Descriptions

Formale Notation: $LC(S, C)$ wobei

- S = Sichtendefinition der Quelle
- C = Constraint der die modelierte Tupelmenge eingrenzt

Beispiel mit globalen Schema:

$$\begin{aligned} S5.MovieGenres(title, genre) &\subseteq Movie(title, dir, year, genre) \\ S6.MovieDirector(title, dir) &\subseteq Movie(title, dir, year, genre) \\ S7.MovieYears(title, year) &\subseteq Movie(title, dir, year, genre) \end{aligned}$$

Mögliche Local Completeness Descriptions:

- $LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$

Local Completeness Descriptions

Formale Notation: $LC(S, C)$ wobei

- S = Sichtendefinition der Quelle
- C = Constraint der die modelierte Tupelmenge eingrenzt

Beispiel mit globalen Schema:

$$\begin{aligned} S5.MovieGenres(title, genre) &\subseteq Movie(title, dir, year, genre) \\ S6.MovieDirector(title, dir) &\subseteq Movie(title, dir, year, genre) \\ S7.MovieYears(title, year) &\subseteq Movie(title, dir, year, genre) \end{aligned}$$

Mögliche Local Completeness Descriptions:

- $LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$
- $LC(S6.MovieDirector(title, dir), American(dir))$

Local Completeness Descriptions

Formale Notation: $LC(S, C)$ wobei

- S = Sichtendefinition der Quelle
- C = Constraint der die modelierte Tupelmenge eingrenzt

Beispiel mit globalen Schema:

$$
\begin{aligned}
S5.MovieGenres(title, genre) &\subseteq Movie(title, dir, year, genre) \\
S6.MovieDirector(title, dir) &\subseteq Movie(title, dir, year, genre) \\
S7.MovieYears(title, year) &\subseteq Movie(title, dir, year, genre)
\end{aligned}
$$

Mögliche Local Completeness Descriptions:

- $LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$
- $LC(S6.MovieDirector(title, dir), American(dir))$
- $LC(S7.MovieYears(title, year), year \geq 1980)$

Vollständigkeit eines Anfrageergebnisses

Vollständigkeit eines Anfrageergebnisses

- Ist das Ergebnis einer Anfrage Q bei einer gegebenen Menge an Quellen vollständig?

Vollständigkeit eines Anfrageergebnisses

- Ist das Ergebnis einer Anfrage Q bei einer gegebenen Menge an Quellen vollständig?

  Beispiel:

  $$LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$$
  $$LC(S6.MovieDirector(title, dir), American(dir))$$
  $$LC(S7.MovieYears(title, year), year \geq 1980)$$

Vollständigkeit eines Anfrageergebnisses

- Ist das Ergebnis einer Anfrage Q bei einer gegebenen Menge an Quellen vollständig?

  Beispiel:

  $$LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$$
  $$LC(S6.MovieDirector(title, dir), American(dir))$$
  $$LC(S7.MovieYears(title, year), year \geq 1980)$$

  $$Q_1(title) \text{ :- } Movie(title, dir, year, \text{'comedy'}), \\ year \geq 1990, American(dir)$$

Vollständigkeit eines Anfrageergebnisses

- Ist das Ergebnis einer Anfrage Q bei einer gegebenen Menge an Quellen vollständig?

  Beispiel:

  $$LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$$
  $$LC(S6.MovieDirector(title, dir), American(dir))$$
  $$LC(S7.MovieYears(title, year), year \geq 1980)$$

$$Q_1(title) \text{ :- } Movie(title, dir, year, \text{'comedy'}), \\ year \geq 1990, American(dir)$$

$$Q_2(title) \text{ :- } Movie(title, dir, year, \text{'comedy'}), \\ year \geq 1970, American(dir)$$

Vollständigkeit eines Anfrageergebnisses

- Ist das Ergebnis einer Anfrage Q bei einer gegebenen Menge an Quellen vollständig?

  Beispiel:

  $$LC(S5.MovieGenres(title, genre), genre = \text{'comedy'})$$
  $$LC(S6.MovieDirector(title, dir), American(dir))$$
  $$LC(S7.MovieYears(title, year), year \geq 1980)$$

  $Q_1(title)$:- $Movie(title, dir, year, \text{'comedy'}),$
  $year \geq 1990, American(dir)$

  $Q_2(title)$:- $Movie(title, dir, year, \text{'comedy'}),$
  $year \geq 1970, American(dir)$

- Die Validierung auf Vollständigkeit kann auf ein Query Containment Problem reduziert werden

Testen auf Vollständigkeit eines Anfrageergebnisses

Gegeben: LaV Ausdruck $S_i(\overline{X_i}) \subseteq R(\overline{X_i}), C_i^*$ mit $LC(S_i, C_i)$,
konjunktive Anfrage Q auf die Quellrelationen $S_1, \ldots, S_n$

Testen auf Vollständigkeit eines Anfrageergebnisses

Gegeben: LaV Ausdruck $S_i(\overline{X_i}) \subseteq R(\overline{X_i}), C_i^*$ mit $LC(S_i, C_i)$, konjunktive Anfrage Q auf die Quellrelationen $S_1, \ldots, S_n$

- Bilde neue (abstrakte) Relationen mit den Namen $E_1, \ldots, E_n$

Testen auf Vollständigkeit eines Anfrageergebnisses

Gegeben: LaV Ausdruck $S_i(\overline{X_i}) \subseteq R(\overline{X_i}), C_i^*$ mit $LC(S_i, C_i)$, konjunktive Anfrage Q auf die Quellrelationen $S_1, \ldots, S_n$

- Bilde neue (abstrakte) Relationen mit den Namen $E_1, \ldots, E_n$
- Definiere die Sichten $V_1, \ldots, V_n$ wie folgt:

$$V_i(\overline{X_i}) \quad :- \quad E_i(\overline{X_i}), \neg C_i$$
$$V_i(\overline{X_i}) \quad :- \quad S_i(\overline{X_i})$$

Testen auf Vollständigkeit eines Anfrageergebnisses

Gegeben: LaV Ausdruck $S_i(\overline{X_i}) \subseteq R(\overline{X_i}), C_i^*$ mit $LC(S_i, C_i)$, konjunktive Anfrage Q auf die Quellrelationen $S_1, \ldots, S_n$

- Bilde neue (abstrakte) Relationen mit den Namen $E_1, \ldots, E_n$
- Definiere die Sichten $V_1, \ldots, V_n$ wie folgt:

$$V_i(\overline{X_i}) \quad :- \quad E_i(\overline{X_i}), \neg C_i$$
$$V_i(\overline{X_i}) \quad :- \quad S_i(\overline{X_i})$$

- Definiere Anfrage Q' indem in Q jedes Vorkommen von S_i durch V_i ersetzt wird

Testen auf Vollständigkeit eines Anfrageergebnisses

Gegeben: LaV Ausdruck $S_i(\overline{X_i}) \subseteq R(\overline{X_i}), C_i^*$ mit $LC(S_i, C_i)$, konjunktive Anfrage Q auf die Quellrelationen $S_1, \ldots, S_n$

- Bilde neue (abstrakte) Relationen mit den Namen $E_1, \ldots, E_n$
- Definiere die Sichten $V_1, \ldots, V_n$ wie folgt:

$$V_i(\overline{X_i}) \quad :- \quad E_i(\overline{X_i}), \neg C_i$$
$$V_i(\overline{X_i}) \quad :- \quad S_i(\overline{X_i})$$

- Definiere Anfrage Q' indem in Q jedes Vorkommen von S_i durch V_i ersetzt wird

Die Antwort von Q ist genau dann vollständig wenn Q und Q' äquivalent sind (in diesem Fall tragen die Tuple von $V_i(\overline{X_i})$:- $E_i(\overline{X_i}), \neg C_i$ nicht zum Ergebnis bei)

Zusammenfassung

Beschreibung einer Datenquelle umfasst:

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Zusammenfassung

Beschreibung einer Datenquelle umfasst:

- welche Daten die Quellen enthält
- wie auf die Daten zugegriffen werden kann

Diese Informationen werden modelliert durch:

- Schema mapping
- Zugriffsbeschränkungen in Form von Adornments
  - haben Auswirkungen auf die Reihenfolge in der die Quellen angefragt werden können
  - erfordern evtl. ein Umschreiben von Anfragen (benötigt Kontextwissen)
- Informationen über die Vollständigkeit der Quelle in Form von *Local Completeness Descriptions*

Literatur

[DHI12] Anhai Doan, Alon Halevy, and Zachary Ives.
Principles of Data Integration.
Morgan Kaufmann, 2012.

Literatur I

Anhai Doan, Alon Halevy, and Zachary Ives.
Principles of Data Integration.
Morgan Kaufmann, 2012.