



UNIVERSITÄT ZU LÜBECK  
INSTITUT FÜR INFORMATIONSSYSTEME

Özgür L. Özçep

# Data Integration

*Lecture 5  
7 May 2020*

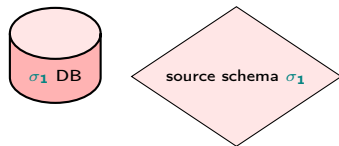
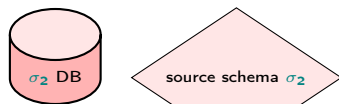
*Informationssysteme CS4130  
(Summer 2020)*

# References

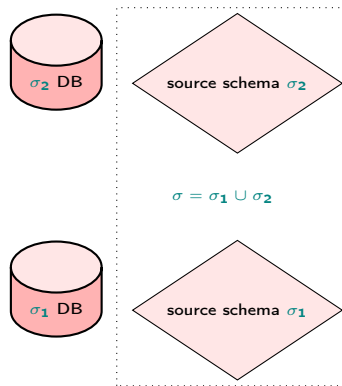
- ▶ Textbook on data integration (in German)  
**Lit:** U. Leser and F. Naumann. Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. Dpunkt-Verl., Heidelberg, 2007.
- ▶ Another newer textbook  
**Lit:** A. Doan, A. Halevy, and Z. Ives. Principles of Data Integration. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 2012.  
Slides <https://research.cs.wisc.edu/dibook/>
- ▶ 2015 Course by L. Libkin on Data integration and Exchange  
<http://homepages.inf.ed.ac.uk/libkin/teach/dataintegr15/>
- ▶ PODS 2002 tutorial by Lenzerini on data integration  
<http://www.dis.uniroma1.it/~lenzerin/homepagine/talks/TutorialPODS02.pdf>

# Data Integration: Motivation

# Data Integration (DI): Main Setting



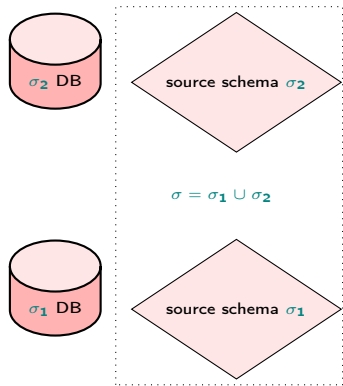
# Data Integration (DI): Main Setting



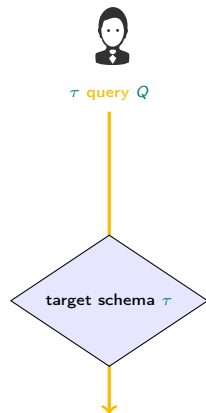
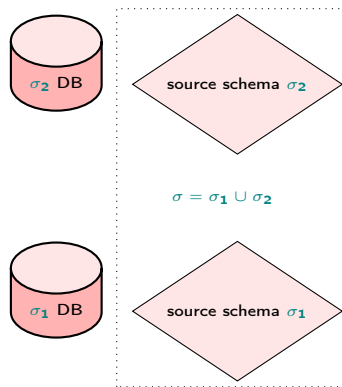
# Data Integration (DI): Main Setting



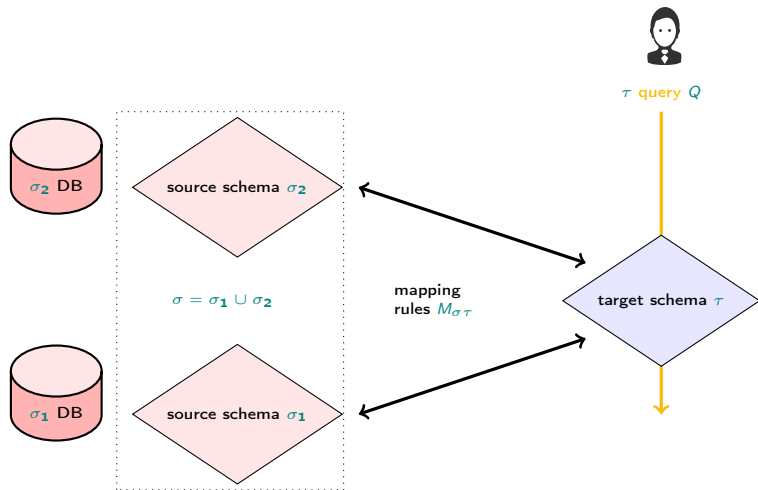
$\tau$  query  $Q$



# Data Integration (DI): Main Setting



# Data Integration (DI): Main Setting

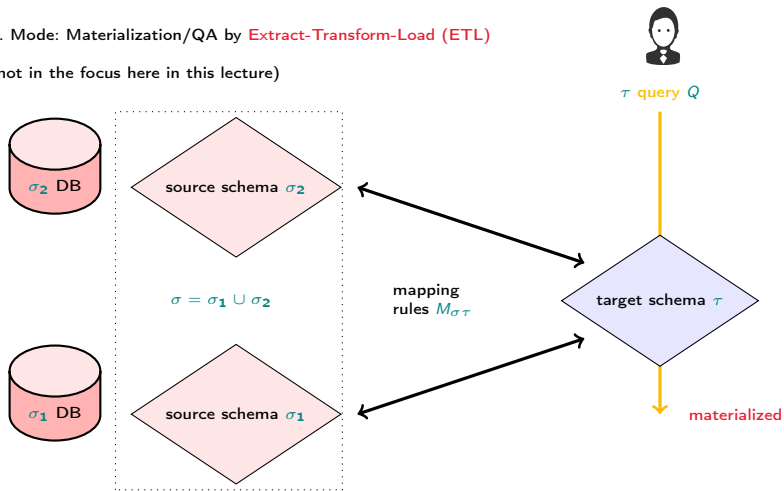




# Data Integration (DI): Main Setting

1. Mode: Materialization/QA by **Extract-Transform-Load (ETL)**

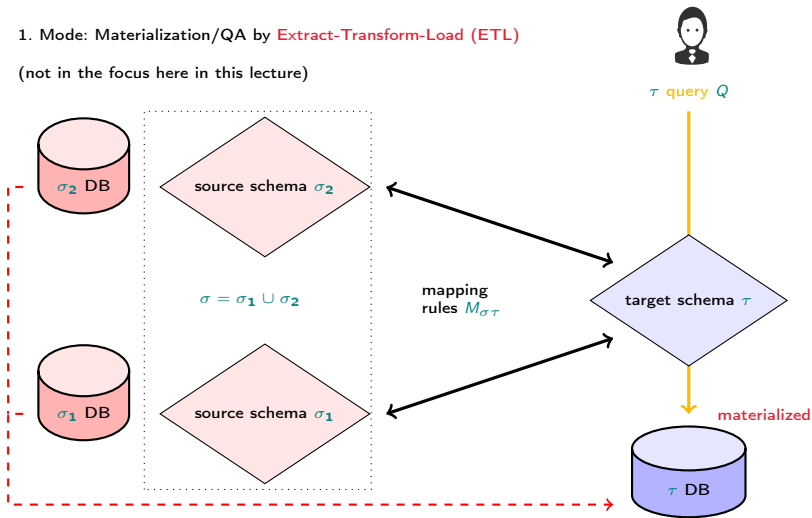
(not in the focus here in this lecture)



# Data Integration (DI): Main Setting

1. Mode: Materialization/QA by **Extract-Transform-Load (ETL)**

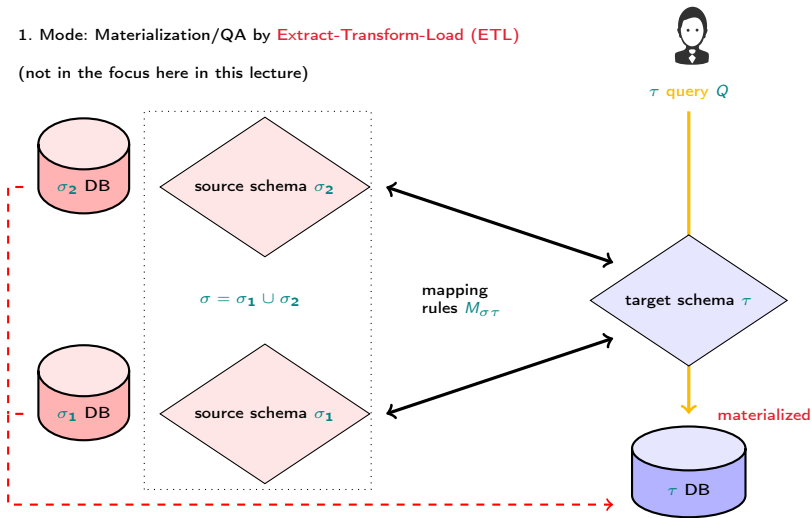
(not in the focus here in this lecture)



# Data Integration (DI): Main Setting

1. Mode: Materialization/QA by **Extract-Transform-Load (ETL)**

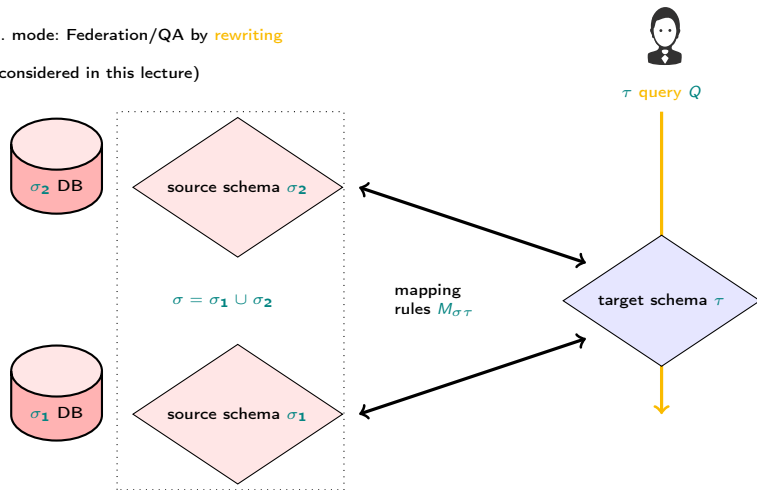
(not in the focus here in this lecture)



# Data Integration (DI): Main Setting

2. mode: Federation/QA by **rewriting**

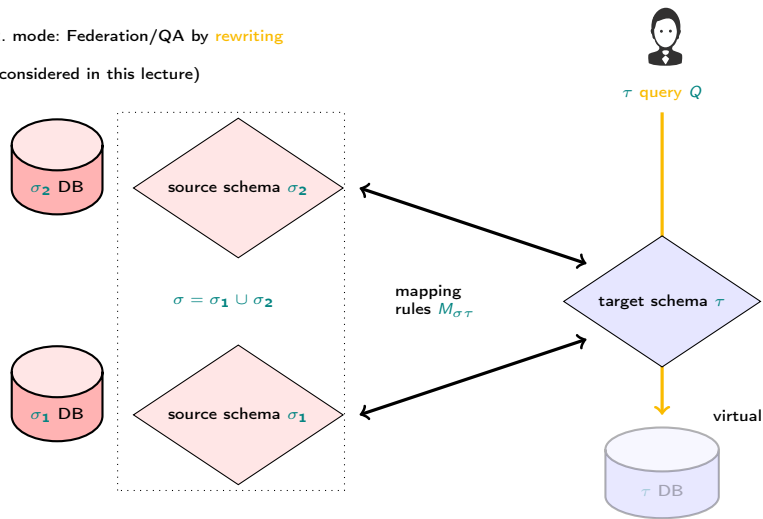
(considered in this lecture)



# Data Integration (DI): Main Setting

2. mode: Federation/QA by **rewriting**

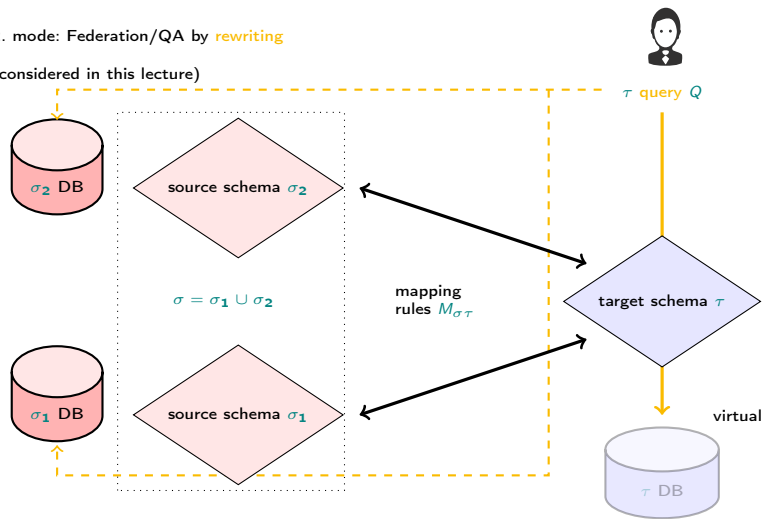
(considered in this lecture)



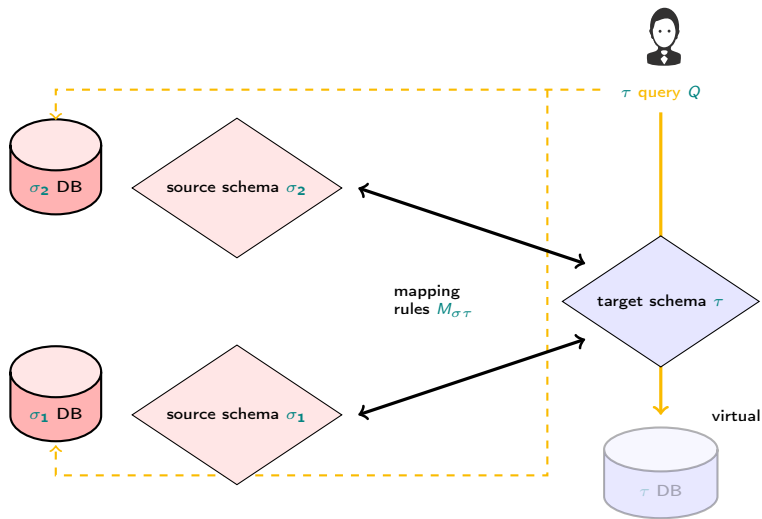
# Data Integration (DI): Main Setting

2. mode: Federation/QA by **rewriting**

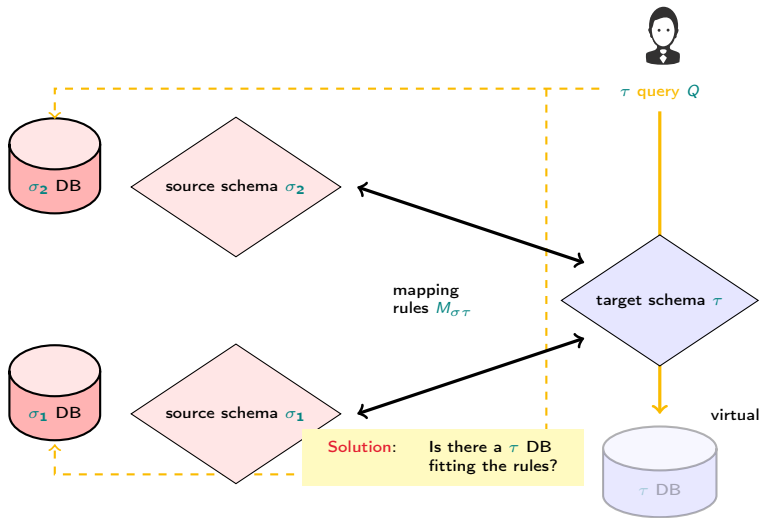
(considered in this lecture)



# Data Integration (DI): Challenges

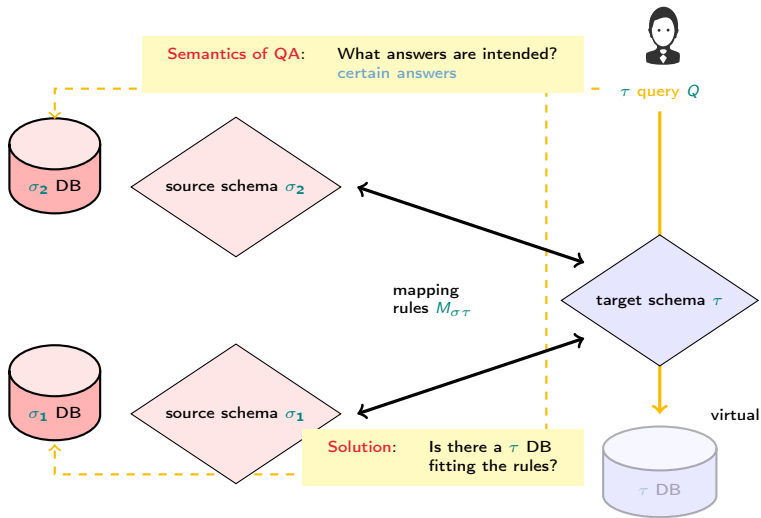


# Data Integration (DI): Challenges

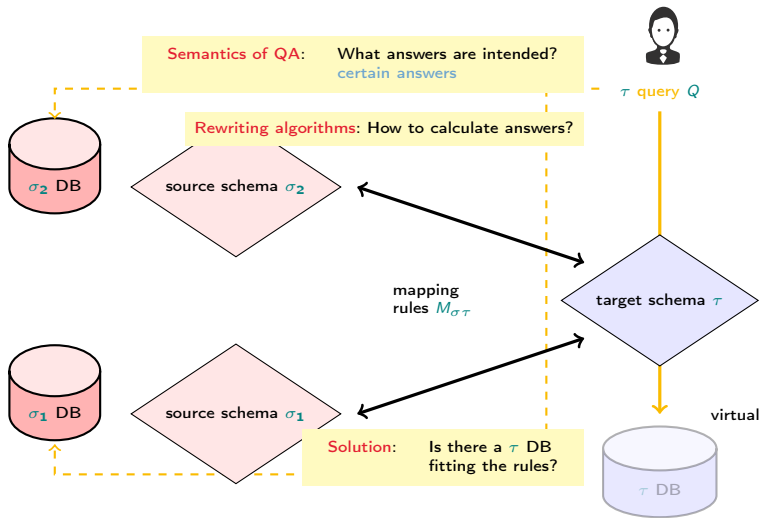




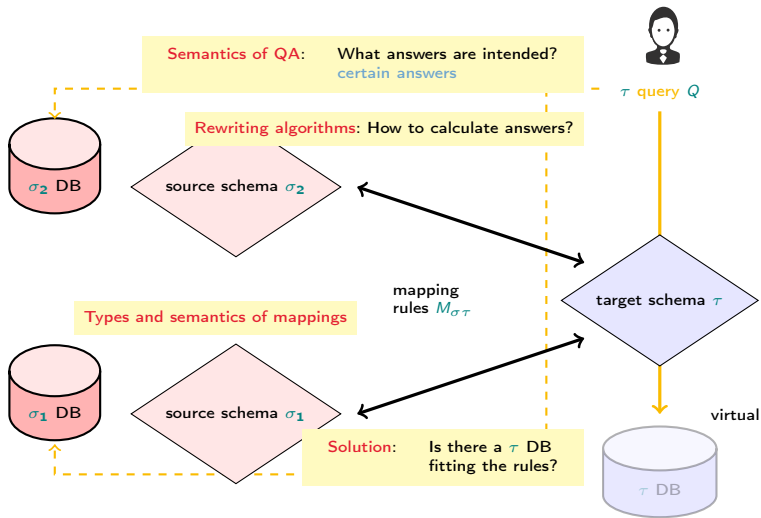
# Data Integration (DI): Challenges



# Data Integration (DI): Challenges



# Data Integration (DI): Challenges



## Short notice on method 1

- ▶ Usually built bottom-up (from sources) to global schema
- ▶ Used in data warehousing
- ▶ Still the most used approach in industry
- ▶ But usually: transformation ad hoc (not even w.r.t. declarative mappings)
- ▶ No well-founded theory in industry
- ▶ In contrast: [Data exchange](#) (see next two lectures)

# Formalization and Basic Notions

# Formalization

## Definition (Lenzerini 2002)

A vector  $(\tau, \sigma, M_{\sigma\tau}, M_{\tau})$  consisting of

- ▶ a global (alias target) schema  $\tau$
- ▶ a source (alias local) schema  $\sigma$
- ▶  $M_{\sigma\tau} = \{ \text{source-to-target rules} \}$
- ▶  $M_{\tau} = \{ \text{target constraints} \}$

is called a **data integration system**  $DI$

- ▶ Lenzerini calls  $M_{\sigma\tau}$  a **mapping**.
- ▶ Source schema  $\sigma$  is the union of the local schemas  
Logically: consider a single  $\sigma$ -DB consisting of disjoint unions of local  $\sigma_i$  DBs
- ▶ Some **federation aspects** dealt under theme complex: **view rewriting**

## Convention

For ease of exposition, we will neglect target dependencies  $M_\tau$ , i.e. let  $M_\tau = \{\}$  in this lecture. Will deal with them in next lecture. (Makes definitions easier and lets us focus on rewriting aspects)

## Source-Target-Dependencies $M_{\sigma\tau}$

- ▶ Source-Target-Dependencies may be arbitrary FOL formula
- ▶ Usually they have a simple form (decidability!)

### Definition

A **source-to-target tuple-generating dependencies (st-tgds)** is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi_{\sigma}(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} \psi_{\tau}(\vec{x}, \vec{z}))$$

where

- ▶  $\phi_{\sigma}$  is a conjunction of atoms over source schema  $\sigma$
- ▶  $\psi_{\tau}$  is a conjunction of atoms over target schema  $\tau$



## Source-Target-Dependencies $M_{\sigma\tau}$

- ▶ Source-Target-Dependencies may be arbitrary FOL formula
- ▶ Usually they have a simple form (decidability!)

### Definition

A **source-to-target tuple-generating dependencies (st-tgds)** is a FOL formula of the form

$$\forall \vec{x}\vec{y}(\phi_{\sigma}(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} \psi_{\tau}(\vec{x}, \vec{z}))$$

where

- ▶  $\phi_{\sigma}$  is a conjunction of atoms over source schema  $\sigma$
  - ▶  $\psi_{\tau}$  is a conjunction of atoms over target schema  $\tau$
- 
- ▶ So in particular, antecedens and succedens conjunctive queries (CQ)
  - ▶ CQs “well-behaved”

## Reminder: Conjunctive Queries (CQs)

- ▶ Class of sufficiently expressive and feasible FOL queries of form

$$ans(\vec{x}) = \exists \vec{y} (\alpha_1(\vec{x}_1, \vec{y}_1) \wedge \cdots \wedge \alpha_n(\vec{x}_n, \vec{y}_n))$$

where

- ▶  $\alpha_i(\vec{x}_i, \vec{y}_i)$  are atomic FOL formula and
- ▶  $\vec{x}_i$  variable vectors among  $\vec{x}$  and  $\vec{y}_i$  variables among  $\vec{y}$
- ▶ Corresponds to SELECT-PROJECT-JOIN Fragment of SQL

### Example (Conjunctive Query from Flight Domain)

$$ans(src, dest, airl, dep) = \exists fno \exists arr (Routes(fno, src, dest) \wedge Info(fno, dep, arr, airl))$$

# Reminder: Conjunctive Queries (CQs)

## Theorem

- ▶ *Answering CQs is NP-complete w.r.t. combined complexity (Chandra, Merlin 1977)*
- ▶ *Subsumption test for CQs is NP complete*
- ▶ *Answering CQs is in  $AC^0$  (and thus in P) w.r.t. data complexity*

**Lit:** A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In: Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, STOC'77, pages 77–90, New York, NY, USA, 1977. ACM.

## Wake-Up Question

Are st-tgds Datalog rules?

## Wake-Up Question

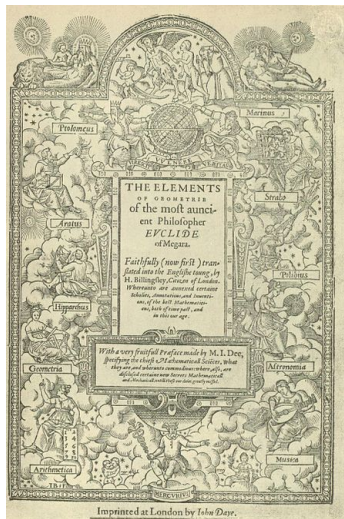
Are st-tgds Datalog rules?

- ▶ No, as Datalog rules do not allow existentials in the head of the query
- ▶ But there is the extended logic called Datalog<sup>+/-</sup>
  - ▶ Has been investigated in last years also in context of ontology-based data access (see net lectures)
  - ▶ Provides many interesting sub-fragments

**Lit:** A. Cali, G. Gottlob, and T. Lukasiewicz. Datalog<sup>+/-</sup>: A unified approach to ontologies and integrity constraints. In Proceedings of the 12th International Conference on Database Theory, pages 14–30. ACM Press, 2009.

# Prominent Tuple Generating Dependencies

- ▶ Theorems of Euclid's "Elements" expressible as tuple generating dependencies



**Lit:** J. Avigad, E. Dean, J. Mumma: "A Formal System for Euclid's Elements", The Review of Symbolic Logic, 2009

# Semantics for Data Integration Systems: Solutions

## Definition

Given: A data integration system  $DI$  with mapping rules  $M_{\sigma\tau}$  and a  $\sigma$ -instance  $\mathcal{G}$

A  $\tau$ -instance  $\mathcal{I}$  is called a **solution** for  $\mathcal{G}$  under  $DI$  iff  $(\mathcal{G}, \mathcal{I})$  satisfies all rules in  $M_{\sigma\tau}$ , for short:  $(\mathcal{G}, \mathcal{I}) \models M_{\sigma\tau}$ .

# Semantics for Data Integration Systems: Solutions

## Definition

Given: A data integration system  $DI$  with mapping rules  $M_{\sigma\tau}$  and a  $\sigma$ -instance  $\mathcal{G}$

A  $\tau$ -instance  $\mathcal{I}$  is called a **solution** for  $\mathcal{G}$  under  $DI$  iff  $(\mathcal{G}, \mathcal{I})$  satisfies all rules in  $M_{\sigma\tau}$ , for short:  $(\mathcal{G}, \mathcal{I}) \models M_{\sigma\tau}$ .

- ▶  $(\mathcal{G}, \mathcal{I}) \models M_{\sigma\tau}$  iff  $\mathcal{G} \cup \mathcal{I} \models M_{\sigma\tau}$  where
  - ▶  $\mathcal{G} \cup \mathcal{I}$  is the union of the instances  $\mathcal{G}, \mathcal{I}$ : Structure containing all relations from  $\mathcal{G}$  and  $\mathcal{I}$  with domain the union of domains of  $\mathcal{G}$  and  $\mathcal{I}$
  - ▶ well defined because schemata are disjoint
- ▶  $Sol_{DI}(\mathcal{G})$ : Set of solutions for  $\mathcal{G}$  under  $DI$



## Certain answering

- ▶ There may be more than one solution.
- ▶ What then is the semantics for query answering?

Definition (Certain answers (informally))

$cert_{DI}(Q, \mathfrak{S}) =$  intersection of answers over all possible solutions

## Certain answering

- ▶ There may be more than one solution.
- ▶ What then is the semantics for query answering?

Definition (Certain answers (formally))

$$\text{cert}_{DI}(Q, \mathcal{G}) = \bigcap_{\mathcal{I} \in \text{Sol}_{DI}(\mathcal{G})} Q(\mathcal{I})$$

## Certain answering

- ▶ There may be more than one solution.
- ▶ What then is the semantics for query answering?

Definition (Certain answers (formally))

$$cert_{DI}(Q, \mathcal{G}) = \bigcap_{\mathcal{T} \in Sol_{DI}(\mathcal{G})} Q(\mathcal{T})$$

- ▶ General approach for dealing with incomplete information
  - ▶ Certain answers on incomplete DBs (see DE lecture)
  - ▶ Certain answers in inconsistent DBs (see Database Repairs lecture)
  - ▶ Certain answers in OBDA
  - ▶ Partial (full meet) revision (See belief revision lecture)

Types of Integration: LAV and GAV

# Various Approaches for Virtual Data Integration

Form of rules in  $M_{\sigma\tau}$  leads to different approaches

- ▶ Source-centric/local-as-view (LAV): sources “defined” in terms of global schema
- ▶ Global-schema-centric/global-as-view (GAV): global schema defined in terms of sources
- ▶ Mixed approach: GLAV
- ▶ Peer-to-peer/P2P: mapping without global schema

# Various Approaches for Virtual Data Integration

Form of rules in  $M_{\sigma\tau}$  leads to different approaches

- ▶ Source-centric/local-as-view (LAV): sources “defined” in terms of global schema
- ▶ Global-schema-centric/global-as-view (GAV): global schema defined in terms of sources
- ▶ Mixed approach: GLAV
- ▶ Peer-to-peer/P2P: mapping without global schema

Focus of this lecture

## Example (Movie Scenario)

- ▶  $\tau$ : *movie*(*Title*, *Year*, *Director*)  
*european*(*Director*)  
*review*(*Title*, *Critique*)
- ▶  $\sigma_1$ :  $r_1(\textit{Title}, \textit{Year}, \textit{Director})$  european directors since 1960
- ▶  $\sigma_1$ :  $r_2(\textit{Title}, \textit{Critique})$  critiques since 1990
- ▶  $Q$ : Title and critique of movies since 1998

$$\exists D.\textit{movie}(T, 1998, D) \wedge \textit{review}(T, R)$$

# GAV Approach

## Definition

A *GAV- $\mathcal{DI}$*  has rules in  $M_{\sigma\tau}$  for all relations  $R_\tau \in \tau$  of the form (called GAV rules):

- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (sound)
- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (exact)



# GAV Approach

## Definition

A *GAV-DI* has rules in  $M_{\sigma\tau}$  for all relations  $R_\tau \in \tau$  of the form (called GAV rules):

- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (sound)
  - ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (exact)
- 
- ▶ Given a source database,  $M_{\sigma\tau}$  provides direct information about which data satisfy the elements of the global schema.

# GAV Approach

## Definition

A *GAV-DI* has rules in  $M_{\sigma\tau}$  for all relations  $R_\tau \in \tau$  of the form (called GAV rules):

- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (sound)
- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (exact)

- ▶ Given a source database,  $M_{\sigma\tau}$  provides direct information about which data satisfy the elements of the global schema.
- ▶ Relations in  $\tau$  are views, and queries are expressed over the views.

# GAV Approach

## Definition

A *GAV-DI* has rules in  $M_{\sigma\tau}$  for all relations  $R_\tau \in \tau$  of the form (called GAV rules):

- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (sound)
- ▶  $\forall \vec{x}\vec{y}(\phi_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z} R_\tau(\vec{x}, \vec{z}))$  (exact)

- ▶ Given a source database,  $M_{\sigma\tau}$  provides direct information about which data satisfy the elements of the global schema.
- ▶ Relations in  $\tau$  are views, and queries are expressed over the views.
- ▶ Simple evaluation by unfolding and running query over the data satisfying the global relations (as if single DB)

## Example (GAV example)

- ▶  $\tau$ : *movie*(Title, Year, Director)  
*european*(Director)  
*review*(Title, Critique)
- ▶  $\sigma_1$ :  $r_1$ (Title, Year, Director)    european directors since 1960
- ▶  $\sigma_2$ :  $r_2$ (Title, Critique)                      critiques since 1990
- ▶  $Q$ :  $\exists D. movie(T, 1998, D) \wedge review(T, R)$
- ▶ GAV rules

$$r_1(T, Y, D) \rightarrow movie(T, Y, D)$$

$$r_1(T, Y, D) \rightarrow european(D)$$

$$r_2(T, R) \rightarrow review(T, R)$$

- ▶ Lhss are views over the source
- ▶ Note: In second rule only attribute  $D$  projected  
 $\forall X, \forall Y A(X, Y) \rightarrow B(X) \equiv \forall X (\exists Y A(X, Y) \rightarrow B(X)).$

## Example (GAV example (continued))

- ▶  $\tau$ : *movie*(Title, Year, Director)  
*european*(Director)  
*review*(Title, Critique)
- ▶  $\sigma_1$ :  $r_1$ (Title, Year, Director)   european directors since 1960
- ▶  $\sigma_2$ :  $r_2$ (Title, Critique)                                  critiques since 1990
- ▶  $Q$ :  $\exists D. movie(T, 1998, D) \wedge review(T, R)$
- ▶ GAV rules

$$r_1(T, Y, D) \rightarrow movie(T, Y, D)$$

$$r_1(T, Y, D) \rightarrow european(D)$$

$$r_2(T, R) \rightarrow review(T, R)$$

- ▶  $Q$  unfolded into

$$\exists D. r_1(T, 1998, D) \wedge r_2(T, R)$$

# LAV Approach

## Definition

A LAV- $\mathcal{DI}$  has rules in  $M_{\sigma\tau}$  for all relations  $R_\sigma \in \sigma$  of the form

- ▶  $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} \psi_\tau(\vec{x}, \vec{z}))$  (sound)
- ▶  $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z} \psi_\tau(\vec{x}, \vec{z}))$  (exact)

# LAV Approach

## Definition

A LAV- $\mathcal{DI}$  has rules in  $M_{\sigma\tau}$  for all relations  $R_\sigma \in \sigma$  of the form

- ▶  $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longrightarrow \exists \vec{z} \psi_\tau(\vec{x}, \vec{z}))$  (sound)
- ▶  $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \longleftrightarrow \exists \vec{z} \psi_\tau(\vec{x}, \vec{z}))$  (exact)

- ▶ Mapping  $M_{\sigma\tau}$  and the source database  $\mathcal{G}$  do not provide direct information about which data satisfy the global schema.

# LAV Approach

## Definition

A LAV- $\mathcal{DI}$  has rules in  $M_{\sigma\tau}$  for all relations  $R_\sigma \in \sigma$  of the form

- ▶  $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \rightarrow \exists \vec{z} \psi_\tau(\vec{x}, \vec{z}))$  (sound)
- ▶  $\forall \vec{x}\vec{y}(R_\sigma(\vec{x}, \vec{y}) \leftrightarrow \exists \vec{z} \psi_\tau(\vec{x}, \vec{z}))$  (exact)

- ▶ Mapping  $M_{\sigma\tau}$  and the source database  $\mathcal{S}$  do not provide direct information about which data satisfy the global schema.
- ▶ Sources are views, and we have to answer queries on the basis of the available data in the views.  $\implies$  view rewriting



## Example (LAV example)

- ▶  $\tau$ : *movie*(Title, Year, Director)  
*european*(Director)  
*review*(Title, Critique)
- ▶  $\sigma_1$ :  $r_1$ (Title, Year, Director)    european directors since 1960
- ▶  $\sigma_2$ :  $r_2$ (Title, Critique)                      critiques since 1990
- ▶  $Q$ :  $\exists D. \textit{movie}(T, 1998, D) \wedge \textit{review}(T, R)$
- ▶ LAV rules

$$r_1(T, Y, D) \rightarrow \textit{movie}(T, Y, D) \wedge \textit{european}(D) \wedge Y \geq 1960$$

$$r_2(T, R) \rightarrow \textit{movie}(T, Y, D) \wedge \textit{review}(T, R) \wedge Y \geq 1990$$

- ▶  $Q$  has to be **rewritten** in order to re-express target atoms with source atoms

$$Q_{rew}(T, R) : \exists D. r_2(T, R) \wedge r_1(T, 1998, D)$$

# GAV-LAV Comparison

- ▶ GAV
  - ▶ Quality depends on how well sources represented in target schema by mapping
  - ▶ Have to reconsider global schema when sources changed/added
  - ▶ Unfolding-based query processing

# GAV-LAV Comparison

- ▶ GAV
  - ▶ Quality depends on how well sources represented in target schema by mapping
  - ▶ Have to reconsider global schema when sources changed/added
  - ▶ Unfolding-based query processing
- ▶ LAV
  - ▶ Quality depends on how well sources represented
  - ▶ High modularity and extensibility (reconsider only the definition the source relation which may have changed)
  - ▶ Query processing needs reasoning (rewriting)

# The Full Picture of Different Integration Scenarios

## ► Parameters

- MT: Mapping Type (GAV or LAV)
- E: Exactness (sound or complete)
- TC: Target-Constraints? (yes or no)
- $\tau$  Query language (not mentioned)

TC	MT	E	INCM	INCN
no	GAV	exact	no	no
no	GAV	sound	yes <sup>a</sup> /no	no
no	LAV	sound	yes	no
no	LAV	exact	yes	yes
yes	GAV	exact	no	yes
yes	GAV	sound	yes	yes
yes	LAV	sound	yes	yes
yes	LAV	exact	yes	yes

## ► Outcomes

- INCM: Incomplete answers (yes, no)
- INCN: Inconsistency of target DBs (yes, no)

<sup>a</sup>For  $\tau$  queries without negation

# View Rewriting

# View Rewriting

- ▶ We saw that QA under LAV requires rewriting
- ▶ But even under GAV may be needed if sources only accessible via views
- ▶ General field of **rewriting w.r.t. views**
- ▶ Relevant problem in
  - ▶ Data warehousing
  - ▶ Query optimization
  - ▶ Physical independence
  - ▶ Accessibility restriction/privacy aspects
- ▶ Problem: Generally (for arbitrary FOL queries) not decidable whether (exact/sound) rewriting possible
  - ⇒ Consider CQs

## Why undecidable?

- ▶ Let  $Q$  be arbitrary FOL query
- ▶ Consider exact LAV
- ▶  $\tau$  is primed copy of  $\sigma$
- ▶  $M_{\sigma\tau}$  identify relations in  $\sigma$  with primed copies in  $\tau$
- ▶ No views at all given.
- ▶ Answering in this setting means answering independently of data in source.
- ▶ Can only happen if  $D_1^Q = D_2^Q$  for all  $\tau$ -DBs  $D_1, D_2$
- ▶ But one knows that deciding  $D_1^Q = D_2^Q$  for arbitrary FOL is undecidable

## Definition (Query rewriting w.r.t views (intuition))

- ▶ Given: Query  $Q$ ,  
View definitions  $V_1, \dots, V_n$
- ▶ a **rewriting**  $Q_{rew}$  is a query that refers to the views only (and possibly interpreted relations)  
An **equivalent rewriting** additionally fulfils  $Q \equiv Q_{rew}$ .



## Definition (Query rewriting w.r.t views (formal))

- ▶ Schema  $\sigma$  of relations with arities  $n_i$
- ▶ Queries  $V_1, \dots, V_k$  over schema  $\tau$  with arities  $n_i$   
Assume those views are CQs
- ▶ Input query  $Q$  over  $\tau$

## Definition (Query rewriting w.r.t views (formal))

- ▶ Schema  $\sigma$  of relations with arities  $n_i$
- ▶ Queries  $V_1, \dots, V_k$  over schema  $\tau$  with arities  $n_i$   
Assume those views are CQs
- ▶ Input query  $Q$  over  $\tau$
- ▶ Query  $Q_{rew}$  over  $\sigma$  is an **equivalent rewriting** of  $Q$  iff for all  $\tau$ -DBs:

$$D^Q = Q_{rew}(V_1(D) \dots V_n(D))$$

## Definition (Query rewriting w.r.t views (formal))

- ▶ Schema  $\sigma$  of relations with arities  $n_i$
- ▶ Queries  $V_1, \dots, V_k$  over schema  $\tau$  with arities  $n_i$   
Assume those views are CQs
- ▶ Input query  $Q$  over  $\tau$
- ▶ Query  $Q_{rew}$  over  $\sigma$  is an **equivalent rewriting** of  $Q$  iff for all  $\tau$ -DBs:

$$D^Q = Q_{rew}(V_1(D) \dots V_n(D))$$

- ▶ Query  $Q_{rew}$  over  $\sigma$  is a **maximally contained rewriting** of  $Q$  iff for all  $\tau$ -DBs
  - ▶  $D^Q \supseteq Q_{rew}(V_1(D) \dots V_n(D))$  and
  - ▶  $Q_{rew}$  is maximal in this property, i.e.  
if  $D^Q \supseteq Q'_{rew}(V_1(D) \dots V_n(D))$  then  $Q'_{rew} \subseteq Q_{rew}$ .

## Example (Equivalent rewriting 1)

- ▶  $DB: \{Movie(ID, title, year, genre), Director(ID, director), Actor(ID, actor)\}$
- ▶  $Q(T, Y, D) : Movie(I, T, Y, G) \wedge Y \geq 1950 \wedge G = comedy \wedge Director(I, D) \wedge Actor(I, D)$
- ▶  $V_1(T, Y, D) : Movie(I, T, Y, G) \wedge Y \geq 1940 \wedge G = comedy \wedge Director(I, D) \wedge Actor(I, D)$
- ▶ Because  $V_1 \supseteq Q$  we get equivalent rewriting  
 $Q_{rew} : V_1(T, Y, D) \wedge Y \geq 1950$

## Example (Equivalent Rewriting 2)

- ▶ DB:  $\{ \text{Movie}(ID, \text{title}, \text{year}, \text{genre}), \text{Director}(ID, \text{director}), \text{Actor}(ID, \text{actor}) \}$
- ▶  $Q(T, Y, D) : \text{Movie}(I, T, Y, G) \wedge Y \geq 1950 \wedge G = \text{comedy} \wedge \text{Director}(I, D) \wedge \text{Actor}(I, D)$
- ▶  $V_2(I, T, Y) : \text{Movie}(I, T, Y, G) \wedge Y \geq 1950 \wedge G = \text{comedy}$
- ▶  $V_3(I, D) : \text{Director}(I, D) \wedge \text{Actor}(I, D)$
- ▶ No subsumption relation for views but nonetheless equivalent rewriting

$$Q_{rew} : V_2(I, T, Y) \wedge V_3(I, D)$$

## Example (Maximally Contained Rewriting)

- ▶ DB:  $\{ \text{Movie}(ID, \text{title}, \text{year}, \text{genre}), \text{Director}(ID, \text{director}), \text{Actor}(ID, \text{actor}) \}$
- ▶  $Q(T, Y, D) : \text{Movie}(I, T, Y, G) \wedge Y \geq 1950 \wedge G = \text{comedy} \wedge \text{Director}(I, D) \wedge \text{Actor}(I, D)$
- ▶  $V_4(I, T, Y) : \text{Movie}(I, T, Y, G) \wedge Y \geq 1960 \wedge G = \text{comedy}$
- ▶  $V_3(I, D) : \text{Director}(I, D) \wedge \text{Actor}(I, D)$
- ▶ Only maximally-contained rewriting possible  
 $Q_{rew} : V_4(I, T, Y) \wedge V_3(I, D)$

## Naive query rewriting algorithm

- ▶ Input
  - ▶ Conjunctive queries  $V_1, \dots, V_n$  over  $\tau$
  - ▶ Query  $Q$  over  $\tau$
- ▶ Output: equivalent or maximally contained  $Q_{rew}$
  
- ▶ Procedure
  - ▶ Guess  $Q'$  over views
  - ▶ Unfold  $Q'$  in terms of views
  - ▶ Check if one unfolding contained in  $Q$
- ▶ If one unfolding  $U$  equivalent with  $Q$ , then  $Q_{rew} = U$
- ▶ Otherwise  $Q_{rew} = \bigvee \text{Unfoldings}(Q')$

## Naive query rewriting algorithm

- ▶ Input
    - ▶ Conjunctive queries  $V_1, \dots, V_n$  over  $\tau$
    - ▶ Query  $Q$  over  $\tau$
  - ▶ Output: equivalent or maximally contained  $Q_{rew}$
  - ▶ Procedure
    - ▶ Guess  $Q'$  over views
    - ▶ Unfold  $Q'$  in terms of views
    - ▶ Check if one unfolding contained in  $Q$
  - ▶ If one unfolding  $U$  equivalent with  $Q$ , then  $Q_{rew} = U$
  - ▶ Otherwise  $Q_{rew} = \bigvee \text{Unfoldings}(Q')$
- 
- ▶ Need to test equivalence: Feasible for CQs



## Naive query rewriting algorithm

- ▶ Input
    - ▶ Conjunctive queries  $V_1, \dots, V_n$  over  $\tau$
    - ▶ Query  $Q$  over  $\tau$
  - ▶ Output: equivalent or maximally contained  $Q_{rew}$
  - ▶ Procedure
    - ▶ Guess  $Q'$  over views
    - ▶ Unfold  $Q'$  in terms of views
    - ▶ Check if one unfolding contained in  $Q$
  - ▶ If one unfolding  $U$  equivalent with  $Q$ , then  $Q_{rew} = U$
  - ▶ Otherwise  $Q_{rew} = \bigvee \text{Unfoldings}(Q')$
- 
- ▶ Need to test equivalence: Feasible for CQs
  - ▶ Need to constrain search space: 1) theoretical bound, 2) prune search space (Bucket, MiniCon, inverse rules)

# Theoretical Bound

## Theorem

- ▶ *If there is an equivalent rewriting (maximally contained rewriting, resp.) of  $Q$ ,*  
*then there is one with at most  $n$  subgoals where  $n$  is number of atoms in  $Q$  (all CQs in disjunction have less than  $n$  atoms, resp.)*
- ▶ *Finding rewriting is NP-complete*

## Bucket Algorithm: Idea

1. Create a bucket for each subgoal  $g$  in query  $Q$   
Bucket contains view atoms contributing to  $g$
2. Create rewritings from the cartesian product of buckets.

# An Example Run

## Example

- ▶  $Q(ID, Dir): \text{Movie}(ID, \text{title}, \text{year}, \text{genre}) \wedge \text{Revenues}(ID, \text{amount}) \wedge \text{Director}(ID, \text{dir}) \wedge \text{amount} \geq 100M$
- ▶  $V_1(I, Y) : \text{Movie}(I, T, Y, G) \wedge \text{Revenues}(I, A) \wedge I \geq 5000 \wedge A \geq 200M$
- ▶  $V_2(I, A) : \text{Movie}(I, T, Y, G) \wedge \text{Revenues}(I, A)$
- ▶  $V_3(I, A) : \text{Revenues}(I, A) \wedge A \leq 50M$
- ▶  $V_4(I, D, Y) \text{Movie}(I, T, Y, G) \wedge \text{Director}(I, D) \wedge I \leq 3000$

# An Example Run

## Example

- ▶  $Q(ID, Dir): \text{Movie}(ID, \text{title}, \text{year}, \text{genre}) \wedge \text{Revenues}(ID, \text{amount}) \wedge \text{Director}(ID, \text{dir}) \wedge \text{amount} \geq 100M$
- ▶  $V_1(I, Y) : \text{Movie}(I, T, Y, G) \wedge \text{Revenues}(I, A) \wedge I \geq 5000 \wedge A \geq 200M$
- ▶  $V_2(I, A) : \text{Movie}(I, T, Y, G) \wedge \text{Revenues}(I, A)$
- ▶  $V_3(I, A) : \text{Revenues}(I, A) \wedge A \leq 50M$
- ▶  $V_4(I, D, Y) \text{Movie}(I, T, Y, G) \wedge \text{Director}(I, D) \wedge I \leq 3000$
  
- ▶ Atoms that can contribute to  $\text{Movie}(ID, \text{title}, \text{year}, \text{genre})$   
 $V_1(ID, \text{year}), V_2(ID, A'), V_4(ID, D', \text{year})$

# An Example Run

## Example

- ▶  $Q(ID, Dir): \text{Movie}(ID, \text{title}, \text{year}, \text{genre}) \wedge \text{Revenues}(ID, \text{amount}) \wedge \text{Director}(ID, \text{dir}) \wedge \text{amount} \geq 100M$
- ▶  $V_1(I, Y) : \text{Movie}(I, T, Y, G) \wedge \text{Revenues}(I, A) \wedge I \geq 5000 \wedge A \geq 200M$
- ▶  $V_2(I, A) : \text{Movie}(I, T, Y, G) \wedge \text{Revenues}(I, A)$
- ▶  $V_3(I, A) : \text{Revenues}(I, A) \wedge A \leq 50M$
- ▶  $V_4(I, D, Y) \text{Movie}(I, T, Y, G) \wedge \text{Director}(I, D) \wedge I \leq 3000$
  
- ▶ Atoms that can contribute to  $\text{Movie}(ID, \text{title}, \text{year}, \text{genre})$   
 $V_1(ID, \text{year}), V_2(ID, A'), V_4(ID, D', \text{year})$
  
- ▶ Similarly for  $\text{Revenues}(ID, \text{amount})$  and  $\text{Director}(ID, \text{dir})$

## Example (First Candidate rewriting)

<i>Movie</i> (ID, title, year, genre)	<i>Revenues</i> (ID, amount)	<i>Director</i> (ID, dir)
$V_1(ID, year)$	$V_1(ID, Y')$	$V_4(ID, Dir, Y')$
$V_2(ID, A')$	$V_2(ID, amount)$	
$V_4(ID, D', year)$		

Consider first triple as potential rewriting

- ▶ first atom redundant,
- ▶ second and third exclusive (so can neglect rewriting  $Q'_1$ )
- ▶  $Q'_1(ID, dir) = \cancel{V_1(ID, year)} \wedge V_1(ID, y') \wedge V_4(ID, dir, y')$

## Example (Second candidate rewriting)

<i>Movie</i> (ID, title, year, genre)	<i>Revenues</i> (ID, amount)	<i>Director</i> (ID, dir)
$V_1(ID, year)$	$V_1(ID, Y')$	$V_4(ID, Dir, Y')$
$V_2(ID, A')$	$V_2(ID, amount)$	
$V_4(ID, D', year)$		

- $Q'_2(ID, dir) = V_2(ID, A') \wedge V_2(ID, amount) \wedge V_4(ID, dir, y')$



## Example (Second candidate rewriting)

<i>Movie</i> ( <i>ID</i> , <i>title</i> , <i>year</i> , <i>genre</i> )	<i>Revenues</i> ( <i>ID</i> , <i>amount</i> )	<i>Director</i> ( <i>ID</i> , <i>dir</i> )
$V_1(ID, year)$	$V_1(ID, Y')$	$V_4(ID, Dir, Y')$
$V_2(ID, A')$	$V_2(ID, amount)$	
$V_4(ID, D', year)$		

becomes redundant

- ▶  $Q'_2(ID, dir) = \overbrace{V_2(ID, A')} \wedge V_2(ID, amount) \wedge V_4(ID, dir, y')$
- ▶  $Q''_2(ID, dir) = V_2(ID, amount) \wedge V_4(ID, dir, y') \wedge amount \geq 100M;$

# Step 1: Create Buckets Procedure

**Input** : CQ  $Q(\vec{X}) = R_1(\vec{X}_1), \dots, R_n(\vec{X}_n), c_1, \dots, c_l$ ; set  $\mathcal{V}$  of CQ views

**Output**: list of buckets

forall  $i \in [n]$  do

    |  $Bucket_i := \emptyset$

end

forall  $i \in [n]$  do

    foreach  $V \in \mathcal{V}$  do

        //Let  $V$  be of form  $V(\vec{Y}) = S_1(\vec{Y}_1), \dots, S_m(\vec{Y}_m), d_1, \dots, d_k$

        forall  $i \in [m]$  do

            if  $R_j = S_i$  then

                // Let  $\psi$  be the mapping defined on the  $vars(V)$  as: Let  $y$  be the  $b$ th variabel in  $\vec{Y}_i$  and  $x$  be the  $b$ th variable in  $\vec{X}_i$

                if  $x \in \vec{X}$  and  $y \notin \vec{Y}$  then

                    |  $\psi$  undefined,  $j++$

                else if  $y \in \vec{Y}$  then

                    |  $\psi(y) = x$

                else

                    |  $\psi(y)$  is a new variable not in  $Q$  or  $V$

                endif

$Q' := R_1(\vec{X}_1), \dots, R_n(\vec{X}_n), c_1, \dots, c_n,$

$S_1(\psi(\vec{Y}_1)), \dots, S_m(\psi(\vec{Y}_m)), \psi(d_1), \dots, \psi(d_k)$

                if  $Q'$  is satisfiable then

                    | Add  $\psi(V)$  to  $Bucket_i$

                endif

            endif

        end

    end

end

return  $Bucket_1, \dots, Bucket_n$

## Step 2: Creating rewritings (exact or max. contained)

Consider each  $Q' \in \text{Bucket}_1 \times \dots \times \text{Bucket}_n$

- ▶ For equivalent rewriting
  - ▶ If  $Q \equiv Q'$  or can add interpreted atoms  $C$  s.t.  $Q \wedge C \equiv Q'$  then  $Q_{rew} := Q$  is a potential rewriting
- ▶ For maximally contained rewriting construct UCQ  $Q_{rew}$  rewriting considering each conjunctive rewriting  $Q'$ 
  - ▶ If  $Q' \subseteq Q$ , then  $Q_{rew} := Q_{rew} \vee Q'$
  - ▶ If there is an interpreted atom  $C$  that can be added to  $Q'$  s.t.  $Q' \wedge C \subseteq Q$  then  $Q_{rew} := Q_{rew} \vee (Q' \wedge C)$
  - ▶ If  $Q' \not\subseteq Q$  but there is homomorphism  $\psi$  on head variables of  $Q'$  s.t.  $\psi(Q') \subseteq Q$ , then  $Q_{rew} := Q_{rew} \vee \psi(Q')$

## Other Algorithms

- ▶ Bucket algorithms do not consider interaction of goals

$$\begin{aligned}Q(\textit{title}, \textit{year}, \textit{dir}) &= \textit{Movie}(\textit{ID}, \textit{title}, \textit{year}, \textit{genre}), \\ &\quad \textit{Director}(\textit{ID}, \textit{dir}), \textit{Actor}(\textit{ID}, \textit{dir}) \\ V_5(D, A) &= \textit{Director}(I, D), \textit{Actor}(I, A)\end{aligned}$$

- ▶ Variable  $I$  not in head of  $V_5$  hence  $V_5$  not usable in rewriting  
 $\implies$  mitigated in MiniCon
- ▶ Logical approach with inverse rules
- ▶ See **Lit:** A. Doan, A. Halevy, and Z. Ives. Principles of Data Integration. 2012. Chapter 2, p. 51 ff

## Further Topics

# Finding (logical) mappings I: Hierarchical approach

Topic on its own, here only some hints

- ▶ Find value correspondences by schema matching methods
- ▶ On top of them build logical mapping rules

## Example

- ▶  $\tau : \text{movie}(\text{Title}, \text{Year}, \text{Director}), \text{european}(\text{Director}), \text{review}(\text{Title}, \text{Critique})$
- ▶  $\sigma : r_1(\text{MTitle}, \text{Year}, \text{Director}), r_2(\text{MTitle}, \text{Critique})$
- ▶ Value correspondence:  $r_1.\text{MTitle} \rightarrow \text{movie}.\text{Title}$  etc. (found by, say, string-matching)
- ▶ Possible LAV-rule:  
 $r_1(T, \text{Year}, \text{Director}) \rightarrow \text{movie}(T, \text{Year}, \text{Director})$

**Lit:** U. Leser and F. Naumann. Informationsintegration: Architekturen und Methoden zur Integration verteilter und heterogener Datenquellen. Chapter 5

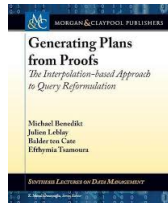
## Finding (Logical) Mappings II: Direct-Supervised

Learn directly logical mappings in supervised fashion

- ▶ Present pairs of sources DBs and potential target DB solutions
- ▶ See, e.g., **Lit:** G. Gottlob and P. Senellart. Schema mapping discovery from data instances. *J. ACM*, 57(2), Feb. 2010.
- ▶ Systemically studied using computational learning theory for GAV mappings in  
**Lit:** B. T. Cate, V. Dalmau, and P. G. Kolaitis. Learning schema mappings. *ACM Trans. Database Syst.*, 38(4):28:1–28:31, Dec. 2013.

# View Rewriting Advanced

- ▶ Consider view rewriting as application of Craig's interpolation theorem
- ▶ Benthem 2008: “last significant property of FOL that has come to light”  
**Lit:** J. van Benthem. *The many faces of interpolation*. *Synthese*, 164(3):451–460, 2008.
- ▶ Benedikt et al. generalize this w.r.t. accessibility methods (privacy)



**Lit:** M. Benedikt, J. Leblay, B. ten Cate, and E. Tsamoura. *Generating Plans from Proofs: The interpolation-based Approach to Query Reformulation*. *Synthesis Lectures on Data Management*, 2016.



## Next Lecture: Data Exchange

- ▶ Main difference: requires materialization
- ▶ Usually considered without views (no access restrictions)
- ▶ Will consider in that lecture also target constraints
- ▶ Will consider mapping management