

# Query Processing in Data Integration Systems

Diego Calvanese

Free University of Bozen-Bolzano

BIT PhD Summer School – Bressanone  
July 3–7, 2006



# Structure of the course

- 1 Introduction to data integration
  - Basic issues in data integration
  - Logical formalization
- 2 Query answering in the absence of constraints
  - Global-as-view (GAV) setting
  - Local-as-view (LAV) and GLAV setting
- 3 Query answering in the presence of constraints
  - The role of integrity constraints
  - Global-as-view (GAV) setting
  - Local-as-view (LAV) and GLAV setting
- 4 Concluding remarks



# Part I

## Introduction to data integration



# Outline

- 1 Basic issues in data integration
  - The problem of data integration
  - Variants of data integration
  - Problems in data integration
- 2 Data integration: Logical formalization
  - Semantics of a data integration system
  - Relational calculus
  - Queries to a data integration system
  - Formalizing the mapping
  - Formalizing GAV data integration systems
  - Formalizing LAV data integration systems



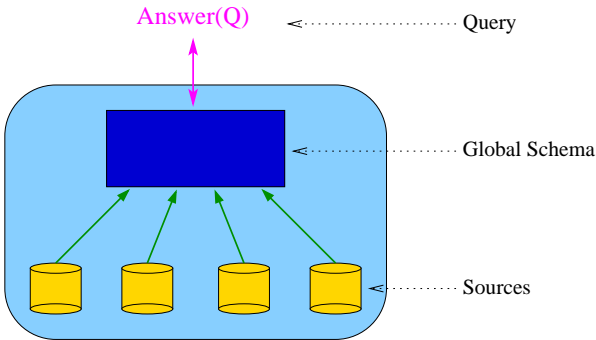
# Outline

- 1 Basic issues in data integration
  - The problem of data integration
  - Variants of data integration
  - Problems in data integration
- 2 Data integration: Logical formalization
  - Semantics of a data integration system
  - Relational calculus
  - Queries to a data integration system
  - Formalizing the mapping
  - Formalizing GAV data integration systems
  - Formalizing LAV data integration systems

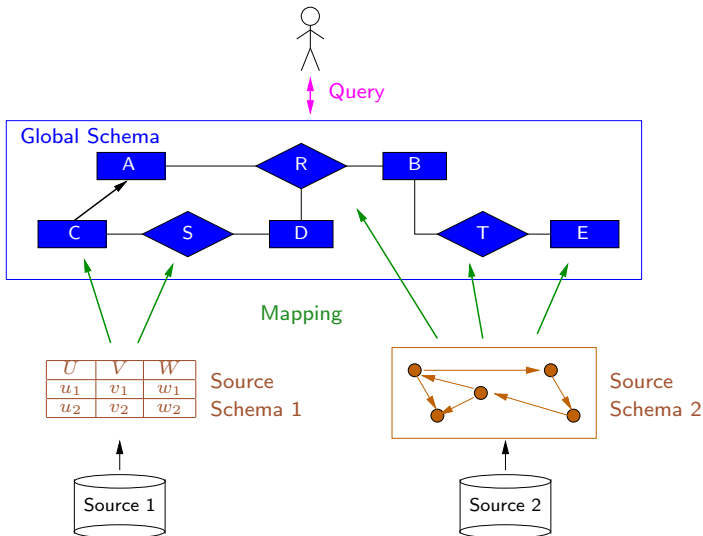


# What is data integration?

Data integration is the problem of providing unified and transparent access to a collection of data stored in **multiple**, **autonomous**, and **heterogeneous** data sources



# Conceptual architecture of a data integration system



# Relevance of data integration

- Growing market
- One of the major challenges for the future of IT
- At least two contexts
  - Intra-organization data integration (e.g., EIS)
  - Inter-organization data integration (e.g., integration on the Web)





# Data integration: Available industrial efforts

- Distributed database systems
- Information on demand
- Tools for source wrapping
- Tools based on database federation, e.g., DB2 Information Integrator
- Distributed query optimization



# Architectures for integrated access to distributed data

- **Distributed databases**  
data sources are homogeneous databases under the control of the distributed database management system
- **Multidatabase or federated databases**  
data sources are autonomous, heterogeneous databases; procedural specification
- **(Mediator-based) data integration**  
access through a global schema mapped to autonomous and heterogeneous data sources; declarative specification
- **Peer-to-peer data integration**  
network of autonomous systems mapped one to each other, without a global schema; declarative specification



# Database federation tools: Characteristics

- **Physical transparency**, i.e., masking from the user the physical characteristics of the sources
- **Heterogeneity**, i.e., federating highly diverse types of sources
- **Extensibility**
- **Autonomy** of data sources
- **Performance**, through distributed query optimization

However, current tools do not (directly) support **logical (or conceptual) transparency**



# Logical transparency

Basic ingredients for achieving logical transparency:

- The global schema (ontology) provides a conceptual view that is independent from the sources
- The global schema is described with a semantically rich formalism
- The mappings are the crucial tools for realizing the independence of the global schema from the sources
- Obviously, the formalism for specifying the mapping is also a crucial point

All the above aspects are not appropriately dealt with by current tools. This means that data integration cannot be simply addressed on a tool basis

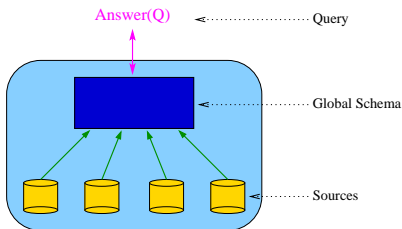
# Approaches to data integration

- (Mediator-based) data integration ... *is the topic of this course*
- Data exchange [Fagin & al. TCS'05, Kolaitis PODS'05]
  - materialization of the global view
  - allows for query answering without accessing the sources
- P2P data integration [Halevy & al. ICDE'03, — & al. PODS'04, — & al. DBPL'05]
  - several peers
  - each peer with local and external sources
  - queries over one peer



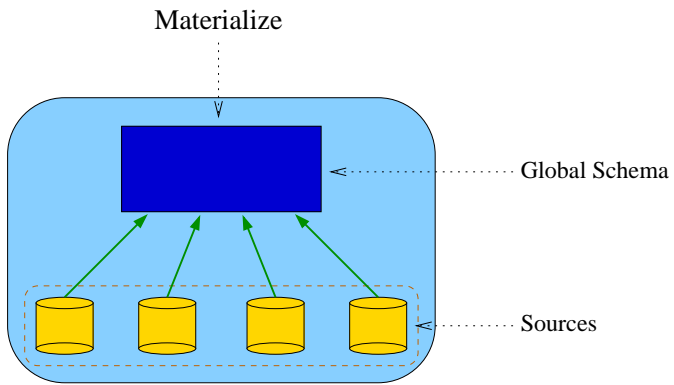
# Mediator based data integration

- Queries are expressed over a **global schema** (a.k.a. mediated schema, enterprise model, ...)
- Data are stored in a set of sources
- **Wrappers** access the sources (provide a view in a uniform data model of the data stored in the sources)
- **Mediators** combine answers coming from wrappers and/or other mediators

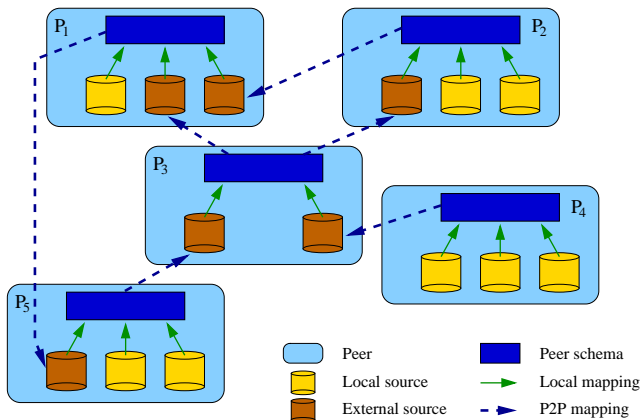


# Data exchange

- Materialization of the global schema



# Peer-to-peer data integration



Operations:    –  $\text{Answer}(Q, P_i)$             –  $\text{Materialize}(P_i)$



# Main problems in data integration

- 1 How to construct the global schema
- 2 (Automatic) source wrapping
- 3 How to discover mappings between sources and global schema
- 4 Limitations in mechanisms for accessing sources
- 5 Data extraction, cleaning, and reconciliation
- 6 How to process updates expressed on the global schema and/or the sources (“read/write” vs. “read-only” data integration)
- 7 How to model the global schema, the sources, and the mappings between the two
- 8 How to answer queries expressed on the global schema
- 9 How to optimize query answering



# The modeling problem

## Basic questions:

- How to model the global schema
  - data model
  - constraints
- How to model the sources
  - data model (conceptual and logical level)
  - access limitations
  - data values (common vs. different domains)
- How to model the mapping between global schemas and sources
- How to verify the quality of the modeling process

**A word of caution:** Data modeling (in data integration) is an art. Theoretical frameworks can help humans, not replace them



# The querying problem

- A query expressed in terms of the global schema must be **reformulated** in terms of (a set of) queries over the sources and/or materialized views
- The computed sub-queries are shipped to the sources, and the results are collected and **assembled** into the final answer
- The computed query plan should guarantee
  - completeness of the obtained answers wrt the semantics
  - efficiency of the whole query answering process
  - efficiency in accessing sources
- This process heavily depends on the approach adopted for modeling the data integration system

**This is the problem that we want to address in this course**

# Outline

- 1 Basic issues in data integration
  - The problem of data integration
  - Variants of data integration
  - Problems in data integration
  
- 2 Data integration: Logical formalization
  - Semantics of a data integration system
  - Relational calculus
  - Queries to a data integration system
  - Formalizing the mapping
  - Formalizing GAV data integration systems
  - Formalizing LAV data integration systems



# Formal framework for data integration

## Definition

A **data integration system**  $\mathcal{I}$  is a triple  $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , where

- $\mathcal{G}$  is the global schema  
*i.e., a logical theory over a relational alphabet  $\mathcal{A}_{\mathcal{G}}$*
- $\mathcal{S}$  is the source schema  
*i.e., simply a relational alphabet  $\mathcal{A}_{\mathcal{S}}$  disjoint from  $\mathcal{A}_{\mathcal{G}}$*
- $\mathcal{M}$  is the mapping between  $\mathcal{S}$  and  $\mathcal{G}$   
*We consider different approaches to the specification of mappings*

# Semantics of a data integration system

Which are the dbs that satisfy  $\mathcal{I}$ , i.e., the logical models of  $\mathcal{I}$ ?

- We refer only to dbs over a **fixed infinite domain**  $\Delta$  of elements
- We start from the data present in the sources: these are modeled through a **source database**  $\mathcal{C}$  over  $\Delta$  (also called source model), fixing the extension of the predicates of  $\mathcal{A}_S$
- The dbs for  $\mathcal{I}$  are logical interpretations for  $\mathcal{A}_G$ , called **global dbs**

## Definition

The **set of databases for  $\mathcal{A}_G$  that satisfy  $\mathcal{I}$  relative to  $\mathcal{C}$**  is:

$$sem^{\mathcal{C}}(\mathcal{I}) = \{ \mathcal{B} \mid \mathcal{B} \text{ is a global database that is legal wrt } \mathcal{G} \\ \text{and that satisfies } \mathcal{M} \text{ wrt } \mathcal{C} \}$$

What it means to satisfy  $\mathcal{M}$  wrt  $\mathcal{C}$  depends on the nature of  $\mathcal{M}$



# Relational calculus: the basics

**Basic idea:** we use the language of first-order logic to express which tuples should be in the result to a query

- We assume to have a domain  $\Delta$  and a set  $\Sigma$  of constants, one for each element of  $\Delta$
- Let  $\mathcal{A}$  be a **relational alphabet**, i.e., a set of predicates, each with an associated arity (we assume a positional notation)
- A **database  $\mathcal{D}$  over  $\mathcal{A}$  and  $\Delta$**  is a set of relations, one for each predicate in  $\mathcal{A}$ , over the constants in  $\Sigma$  (in turn interpreted as elements of  $\Delta$ )
- Let  $\mathcal{L}_{\mathcal{A}}$  be the first-order language over
  - the constants in  $\Sigma$
  - the predicates of  $\mathcal{A}$  plus the built-in predicates of relational algebra (e.g.,  $<$ ,  $>$ , ...)
  - no function symbols



# Relational calculus: Syntax

## Definition

An **(domain) relational calculus query** over alphabet  $\mathcal{A}$  has the form

$$\{ (x_1, \dots, x_n) \mid \varphi \},$$

where

- $n \geq 0$  is the **arity** of the query
- $x_1, \dots, x_n$  are (not necessarily distinct) variables
- $\varphi$  is the **body** of the query, i.e., a formula of  $\mathcal{L}_{\mathcal{A}}$  whose free variables are exactly  $x_1, \dots, x_n$
- $(x_1, \dots, x_n)$  is called the **target list** of the query

If  $r$  is a predicate of arity  $k$ , an **atom** with predicate  $r$  has the form  $r(y_1, \dots, y_k)$ , where  $y_1, \dots, y_k$  are variables or constants





# Relational calculus: Semantics

Relational calculus queries are evaluated on particular interpretations

## Definition

A **correct interpretation** for relational calculus queries over  $\mathcal{A}$  is a pair  $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$ , where  $\Delta$  is a domain, and  $\mathcal{D}$  is a database over  $\mathcal{A}$  and  $\Delta$

## Definition

The **value** of a relational calculus query  $q = \{(x_1, \dots, x_n) \mid \varphi\}$  in an interpretation  $\mathcal{I} = \langle \Delta, \mathcal{D} \rangle$  is the set of tuples  $(c_1, \dots, c_n)$  of constants in  $\Sigma$  such that  $\langle \mathcal{I}, \mathcal{V} \rangle \models \varphi$ , where  $\mathcal{V}$  is the variable assignment that assigns  $c_i$  to  $x_i$

When the domain  $\Delta$  is clear, we can omit it, and write directly  $\langle \mathcal{D}, \mathcal{V} \rangle \models \varphi$ , instead of  $\langle \langle \Delta, \mathcal{D} \rangle, \mathcal{V} \rangle \models \varphi$



# Result of relational calculus queries

## Definition

The **result of the evaluation** of a relational calculus query  $q = \{(x_1, \dots, x_n) \mid \varphi\}$  on a database  $\mathcal{D}$  over  $\mathcal{A}$  and  $\Delta$  is the relation  $q^{\mathcal{D}}$  such that

- the arity of  $q^{\mathcal{D}}$  is  $n$
- the extension of  $q^{\mathcal{D}}$  is the set of constants that constitute the value of the query  $q$  in the interpretation  $\langle \Delta, \mathcal{D} \rangle$

# Conjunctive queries

- are the most common kind of relational calculus queries
- also known as **select-project-join** SQL queries
- allow for easy optimization in relational DBMSs

## Definition

A **conjunctive query** (CQ) is a relational calculus query of the form

$$\{ (\vec{x}) \mid \exists \vec{y}. r_1(\vec{x}_1, \vec{y}_1) \wedge \cdots \wedge r_m(\vec{x}_m, \vec{y}_m) \}$$

where

- $\vec{x}$  is the union of the  $\vec{x}_i$ 's, and  $\vec{y}$  is the union of the  $\vec{y}_i$ 's
- $r_1, \dots, r_m$  are relation symbols (not built-in predicates)

We use the following abbreviation:  $\{ (\vec{x}) \mid r_1(\vec{x}_1, \vec{y}_1), \dots, r_m(\vec{x}_m, \vec{y}_m) \}$

# Complexity of relational calculus

We consider the complexity of the **recognition problem**, i.e., checking whether a tuple of constants is in the answer to a query:

- measured wrt the size of the database  $\rightsquigarrow$  **data complexity**
- measured wrt the size of the query and the database  $\rightsquigarrow$  **combined complexity**

## Complexity of relational calculus

- data complexity: polynomial, actually in LOGSPACE
- combined complexity: PSPACE-complete

## Complexity of conjunctive queries

- data complexity: in LOGSPACE
- combined complexity: NP-complete



# Queries to a data integration system $\mathcal{I}$

- The domain  $\Delta$  is fixed, and we do not distinguish an element of  $\Delta$  from the constant denoting it  $\rightsquigarrow$  **standard names**
- Queries to  $\mathcal{I}$  are relational calculus queries over the alphabet  $\mathcal{A}_{\mathcal{G}}$  of the global schema
- When “evaluating”  $q$  over  $\mathcal{I}$ , we have to consider that for a **given source database  $\mathcal{C}$** , there may be **many global databases  $\mathcal{B}$**  in  $sem^{\mathcal{C}}(\mathcal{I})$
- We consider those answers to  $q$  that hold for **all** global databases in  $sem^{\mathcal{C}}(\mathcal{I})$   
 $\rightsquigarrow$  **certain answers**



# Semantics of queries to $\mathcal{I}$

## Definition

Given  $q$ ,  $\mathcal{I}$ , and  $\mathcal{C}$ , the set of **certain answers to  $q$  wrt  $\mathcal{I}$  and  $\mathcal{C}$**  is

$$\mathit{cert}(q, \mathcal{I}, \mathcal{C}) = \{ (c_1, \dots, c_n) \in q^{\mathcal{B}} \mid \text{for all } \mathcal{B} \in \mathit{sem}^{\mathcal{C}}(\mathcal{I}) \}$$

- Query answering is **logical implication**
- Complexity is measured mainly *wrt the size of the source db  $\mathcal{C}$* , i.e., we consider **data complexity**
- We consider the problem of deciding whether  $\vec{c} \in \mathit{cert}(q, \mathcal{I}, \mathcal{C})$ , for a given  $\vec{c}$



# Databases with incomplete information, or knowledge bases

- **Traditional database:** one model of a first-order theory  
Query answering means **evaluating** a formula in the model
- **Database with incomplete information, or knowledge base:** set of models (specified, for example, as a restricted first-order theory)  
Query answering means computing the tuples that satisfy the query in **all** the models in the set

There is a **strong connection** between query answering in data integration and query answering in databases with incomplete information under constraints (or, query answering in knowledge bases)



# Query answering with incomplete information

- [Reiter '84]: relational setting, databases with incomplete information modeled as a first order theory
- [Vardi '86]: relational setting, complexity of reasoning in closed world databases with unknown values
- Several approaches both from the DB and the KR community
- [van der Meyden '98]: survey on logical approaches to incomplete information in databases





# The mapping

How is the mapping  $\mathcal{M}$  between  $\mathcal{S}$  and  $\mathcal{G}$  specified?

- Are the sources defined in terms of the global schema?  
Approach called **source-centric**, or **local-as-view**, or **LAV**
- Is the global schema defined in terms of the sources?  
Approach called **global-schema-centric**, or **global-as-view**, or **GAV**
- A mixed approach?  
Approach called **GLAV**



# GAV vs. LAV – Example

## Global schema:

*movie(Title, Year, Director)*

*european(Director)*

*review(Title, Critique)*

## Source 1:

*r<sub>1</sub>(Title, Year, Director)* since 1960, european directors

## Source 2:

*r<sub>2</sub>(Title, Critique)* since 1990

## Query: Title and critique of movies in 1998

$\{ (t, r) \mid \exists d. \text{movie}(t, 1998, d) \wedge \text{review}(t, r) \}$ , abbreviated

$\{ (t, r) \mid \text{movie}(t, 1998, d), \text{review}(t, r) \}$



# Formalization of GAV

In GAV (with **sound sources**), the mapping  $\mathcal{M}$  is a set of assertions:

$$\phi_S \rightsquigarrow g$$

one for each element  $g$  in  $\mathcal{A}_G$ , with  $\phi_S$  a **query** over  $S$  of the arity of  $g$

Given a source db  $\mathcal{C}$ , a db  $\mathcal{B}$  for  $\mathcal{G}$  satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$  if for each  $g \in \mathcal{G}$ :

$$\phi_S^{\mathcal{C}} \subseteq g^{\mathcal{B}}$$

In other words, the assertion means  $\forall \vec{x}. \phi_S(\vec{x}) \rightarrow g(\vec{x})$

Given a source database,  $\mathcal{M}$  **provides direct information** about which data satisfy the elements of the global schema

*Relations in  $\mathcal{G}$  are views, and queries are expressed over the views.*

Thus, it **seems** that we can simply evaluate the query over the data satisfying the global relations (as if we had a single database at hand)



# GAV – Example

**Global schema:**  $\text{movie}(Title, Year, Director)$   
 $\text{european}(Director)$   
 $\text{review}(Title, Critique)$

**GAV:** to each relation in the global schema,  $\mathcal{M}$  associates a view over the sources:

$$\begin{aligned} \{ (t, y, d) \mid r_1(t, y, d) \} &\rightsquigarrow \text{movie}(t, y, d) \\ \{ (d) \mid r_1(t, y, d) \} &\rightsquigarrow \text{european}(d) \\ \{ (t, r) \mid r_2(t, r) \} &\rightsquigarrow \text{review}(t, r) \end{aligned}$$

Logical formalization:

$$\begin{aligned} \forall t, y, d. r_1(t, y, d) &\rightarrow \text{movie}(t, y, d) \\ \forall d. (\exists t, y. r_1(t, y, d)) &\rightarrow \text{european}(d) \\ \forall t, r. r_2(t, r) &\rightarrow \text{review}(t, r) \end{aligned}$$



# GAV – Example of query processing

The query

$$\{ (t, r) \mid \text{movie}(t, 1998, d), \text{review}(t, r) \}$$

is processed by means of **unfolding**, i.e., by expanding each atom according to its associated definition in  $\mathcal{M}$ , so as to come up with source relations

In this case:

$$\begin{array}{ccc} \{ (t, r) \mid \text{movie}(t, 1998, d), \text{review}(t, r) \} & & \\ \text{unfolding} & \downarrow & \downarrow \\ \{ (t, r) \mid r_1(t, 1998, d), r_2(t, r) \} & & \end{array}$$



# GAV – Example of constraints

## Global schema containing constraints:

movie(*Title, Year, Director*)

european(*Director*)

review(*Title, Critique*)

european\_movie\_60s(*Title, Year, Director*)

$\forall t, y, d. \text{european\_movie\_60s}(t, y, d) \rightarrow \text{movie}(t, y, d)$

$\forall d. \exists t, y. \text{european\_movie\_60s}(t, y, d) \rightarrow \text{european}(d)$

## GAV mappings:

$\{ (t, y, d) \mid r_1(t, y, d) \} \rightsquigarrow \text{european\_movie\_60s}(t, y, d)$

$\{ (d) \mid r_1(t, y, d) \} \rightsquigarrow \text{european}(d)$

$\{ (t, r) \mid r_2(t, r) \} \rightsquigarrow \text{review}(t, r)$



# Formalization of LAV

In LAV (with **sound sources**), the mapping  $\mathcal{M}$  is a set of assertions:

$$s \rightsquigarrow \phi_{\mathcal{G}}$$

one for each source element  $s$  in  $\mathcal{A}_{\mathcal{S}}$ , with  $\phi_{\mathcal{G}}$  a **query** over  $\mathcal{G}$

Given source db  $\mathcal{C}$ , a db  $\mathcal{B}$  for  $\mathcal{G}$  satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$  if for each  $s \in \mathcal{S}$ :

$$s^{\mathcal{C}} \subseteq \phi_{\mathcal{G}}^{\mathcal{B}}$$

In other words, the assertion means  $\forall \vec{x}. s(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$

The mapping  $\mathcal{M}$  and the source database  $\mathcal{C}$  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*



# LAV – Example

**Global schema:**     $movie(Title, Year, Director)$   
                            $europaean(Director)$   
                            $review(Title, Critique)$

**LAV:** to each source relation,  $\mathcal{M}$  associates a view over the global schema:

$$r_1(t, y, d) \rightsquigarrow \{ (t, y, d) \mid movie(t, y, d), europaean(d), y \geq 1960 \}$$

$$r_2(t, r) \rightsquigarrow \{ (t, r) \mid movie(t, y, d), review(t, r), y \geq 1990 \}$$

The query  $\{ (t, r) \mid movie(t, 1998, d), review(t, r) \}$  is processed by means of an inference mechanism that aims at re-expressing the atoms of the global schema in terms of atoms at the sources.

In this case:

$$\{ (t, r) \mid r_2(t, r), r_1(t, 1998, d) \}$$





# GAV and LAV – Comparison

**GAV:** (e.g., Carnot, SIMS, Tsimmis, IBIS, Momis, DisAtDis, ...)

- Quality depends on how well we have compiled the sources into the global schema through the mapping
- Whenever a source changes or a new one is added, the global schema needs to be reconsidered
- Query processing can be based on some sort of unfolding (query answering looks easier – without constraints)

**LAV:** (e.g., Information Manifold, DWQ, Pictel)

- Quality depends on how well we have characterized the sources
- High modularity and extensibility (if the global schema is well designed, when a source changes, only its definition is affected)
- Query processing needs reasoning (query answering complex)



# Beyond GAV and LAV: GLAV

In GLAV (with **sound sources**), the mapping  $\mathcal{M}$  is a set of assertions:

$$\phi_S \rightsquigarrow \phi_G$$

with  $\phi_S$  a **query** over  $\mathcal{S}$ , and  $\phi_G$  a **query** over  $\mathcal{G}$  of the same arity as  $\phi_S$

Given source db  $\mathcal{C}$ , a db  $\mathcal{B}$  for  $\mathcal{G}$  satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$  if for each  $\phi_S \rightsquigarrow \phi_G$  in  $\mathcal{M}$ :

$$\phi_S^{\mathcal{C}} \subseteq \phi_G^{\mathcal{B}}$$

In other words, the assertion means  $\forall \vec{x}. \phi_S(\vec{x}) \rightarrow \phi_G(\vec{x})$

As for LAV, the mapping  $\mathcal{M}$  does **not** provide direct information about which data satisfy the global schema

To answer a query  $q$  over  $\mathcal{G}$ , we have to **infer** how to use  $\mathcal{M}$  in order to access the source database  $\mathcal{C}$



# GLAV – Example

**Global schema:**  $\text{work}(\text{Person}, \text{Project}), \quad \text{area}(\text{Project}, \text{Field})$

**Source 1:**  $\text{hasjob}(\text{Person}, \text{Field})$

**Source 2:**  $\text{teaches}(\text{Professor}, \text{Course}), \quad \text{in}(\text{Course}, \text{Field})$

**Source 3:**  $\text{get}(\text{Researcher}, \text{Grant}), \quad \text{for}(\text{Grant}, \text{Project})$

**GLAV mapping:**

$$\{(r, f) \mid \text{hasjob}(r, f)\} \quad \rightsquigarrow \{(r, f) \mid \text{work}(r, p), \text{area}(p, f)\}$$

$$\{(r, f) \mid \text{teaches}(r, c), \text{in}(c, f)\} \rightsquigarrow \{(r, f) \mid \text{work}(r, p), \text{area}(p, f)\}$$

$$\{(r, p) \mid \text{get}(r, g), \text{for}(g, p)\} \quad \rightsquigarrow \{(r, f) \mid \text{work}(r, p)\}$$


# GLAV – A technical observation

In GLAV (with **sound sources**), the mapping  $\mathcal{M}$  is constituted by a set of assertions:

$$\phi_S \rightsquigarrow \phi_G$$

Each such assertion can be rewritten wlog by introducing a **new predicate**  $r$  (not to be used in the queries) of the same arity as the two queries and replace the assertion with the following two:

$$\phi_S \rightsquigarrow r \qquad r \rightsquigarrow \phi_G$$

In other words, we replace  $\forall \vec{x}. \phi_S(\vec{x}) \rightarrow \phi_G(\vec{x})$   
with  $\forall \vec{x}. \phi_S(\vec{x}) \rightarrow r(\vec{x})$  and  $\forall \vec{x}. r(\vec{x}) \rightarrow \phi_G(\vec{x})$



## Part II

# Query answering in the absence of constraints



# Outline

- 3 Query answering in GAV without constraints
  - Retrieved global database
  - Query answering via unfolding
  - Universal solutions
  
- 4 Query answering in (G)LAV without constraints
  - (G)LAV and incompleteness
  - Approaches to query answering in (G)LAV
  - (G)LAV: Direct methods (aka view-based query answering)
  - (G)LAV: Query answering by (view-based) query rewriting



# Query answering in different approaches

The problem of query answering comes in different forms, depending on several parameters:

- **Global schema**
  - **without** constraints (i.e., empty theory)
  - **with** constraints
- **Mapping**
  - **GAV**
  - **LAV** (or **GLAV**)
- **Queries**
  - **user** queries
  - queries in the **mapping**



# Conjunctive queries

We recall the definition of a conjunctive query:

## Definition

A **conjunctive query** (CQ) is a relational calculus query of the form

$$\{ (\vec{x}) \mid \exists \vec{y}. r_1(\vec{x}_1, \vec{y}_1) \wedge \cdots \wedge r_m(\vec{x}_m, \vec{y}_m) \}$$

where

- $\vec{x}$  is the union of the  $\vec{x}_i$ 's, and  $\vec{y}$  is the union of the  $\vec{y}_i$ 's
- $r_1, \dots, r_m$  are relation symbols (not built-in predicates)

Unless otherwise specified, we consider conjunctive queries, both as user queries and as queries in the mapping





# Incompleteness and inconsistency

Query answering heavily depends upon whether incompleteness/inconsistency shows up

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
no	(G)LAV	yes	no
yes	GAV	yes	yes
yes	(G)LAV	yes	yes



# Outline

- 3 Query answering in GAV without constraints
  - Retrieved global database
  - Query answering via unfolding
  - Universal solutions
  
- 4 Query answering in (G)LAV without constraints
  - (G)LAV and incompleteness
  - Approaches to query answering in (G)LAV
  - (G)LAV: Direct methods (aka view-based query answering)
  - (G)LAV: Query answering by (view-based) query rewriting



# GAV data integration systems without constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
<b>no</b>	<b>GAV</b>	yes / no	no
no	(G)LAV	yes	no
yes	GAV	yes	yes
yes	(G)LAV	yes	yes



# GAV – Retrieved global database

## Definition

Given a source database  $\mathcal{C}$ , we call **retrieved global database**, denoted  $\mathcal{M}(\mathcal{C})$ , the global database obtained by “applying” the queries in the mapping, and “transferring” to the elements of  $\mathcal{G}$  the corresponding retrieved tuples



# GAV – Example

Consider  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , with

Global schema  $\mathcal{G}$ :  
 student(*Code*, *Name*, *City*)  
 university(*Code*, *Name*)  
 enrolled(*Scode*, *Ucode*)

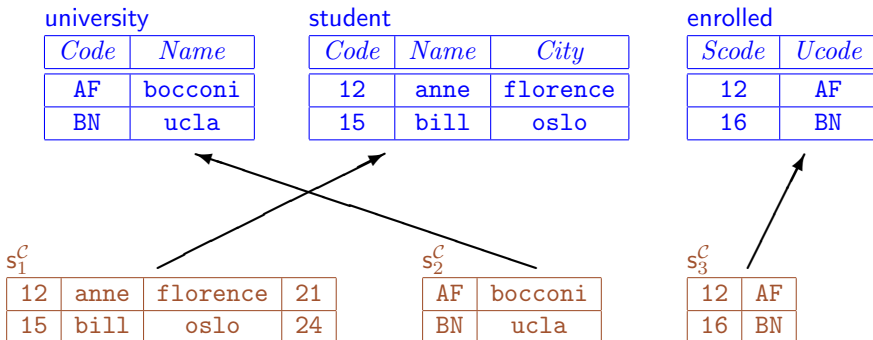
Source schema  $\mathcal{S}$ : relations  $s_1$ (*Scode*, *Sname*, *City*, *Age*),  
 $s_2$ (*Ucode*, *Uname*),  $s_3$ (*Scode*, *Ucode*)

Mapping  $\mathcal{M}$ :

$$\begin{aligned} \{ (c, n, ci) \mid s_1(c, n, ci, a) \} &\rightsquigarrow \text{student}(c, n, ci) \\ \{ (c, n) \mid s_2(c, n) \} &\rightsquigarrow \text{university}(c, n) \\ \{ (s, u) \mid s_3(s, u) \} &\rightsquigarrow \text{enrolled}(s, u) \end{aligned}$$



# GAV – Example of retrieved global database



Example of source database  $\mathcal{C}$  and corresponding retrieved global database  $\mathcal{M}(\mathcal{C})$



# GAV – Minimal model

GAV mapping assertions  $\phi_S \rightsquigarrow g$  have the logical form:

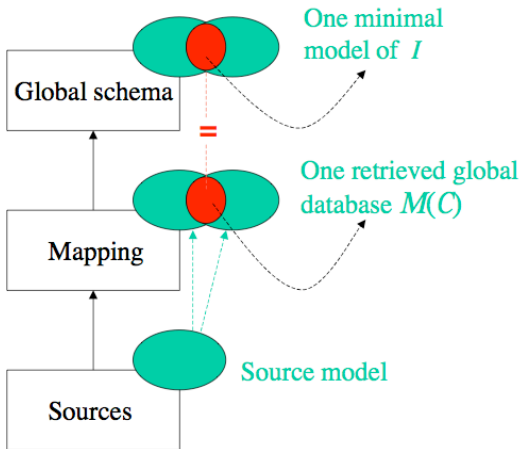
$$\forall \vec{x}. \phi_S(\vec{x}) \rightarrow g(\vec{x})$$

where  $\phi_S$  is a conjunctive query over the source relations, and  $g$  is an element of  $\mathcal{G}$ .

In general, given a source database  $\mathcal{C}$ , there are several databases legal wrt  $\mathcal{G}$  that satisfy  $\mathcal{M}$  wrt  $\mathcal{C}$ .

However, it is easy to see that  $\mathcal{M}(\mathcal{C})$  is the intersection of all such databases, and therefore, is the **only "minimal" model** of  $\mathcal{I}$ .

# GAV without constraints





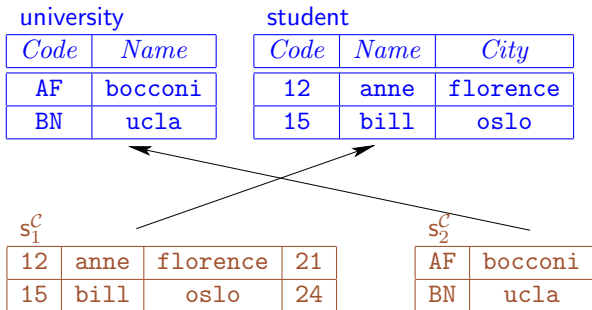
# GAV – Query answering via unfolding

The **unfolding wrt  $\mathcal{M}$  of a query  $q$  over  $\mathcal{G}$** : is the query obtained from  $q$  by substituting every symbol  $g$  in  $q$  with the query  $\phi_S$  that  $\mathcal{M}$  associates to  $g$ . We denote the unfolding of  $q$  wrt  $\mathcal{M}$  with  **$unf_{\mathcal{M}}(q)$**

## Observations:

- $unf_{\mathcal{M}}(q)$  is a query over  $\mathcal{S}$
- Evaluating  $q$  over  $\mathcal{M}(\mathcal{C})$  is equivalent to evaluating  $unf_{\mathcal{M}}(q)$  over  $\mathcal{C}$ .  
i.e.,  $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$  iff  $\vec{t} \in unf_{\mathcal{M}}(q)^{\mathcal{C}}$
- If  $q$  is a conjunctive query, then  $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$  iff  $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$
- Hence,  $\vec{t} \in cert(q, \mathcal{I}, \mathcal{C})$  iff  $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$  iff  $\vec{t} \in unf_{\mathcal{M}}(q)^{\mathcal{C}}$   
 $\rightsquigarrow$  **Unfolding suffices for query answering in GAV without constraints**
- **Data complexity** of query answering is **polynomial** (actually **LOGSPACE**): the query  $unf_{\mathcal{M}}(q)$  is first-order (in fact conjunctive)

# GAV – Example of unfolding



# GAV – More expressive queries?

Do the results extend to the case of more expressive queries?

- With more expressive queries in the mapping?
  - Same results hold if we use **any computable query** in the mapping
- With more expressive user queries?
  - Same results hold if we use **Datalog queries** as user queries
  - Same results hold if we use **union of conjunctive queries with inequalities** as user queries [van der Meyden TCS'93]



# Homomorphisms

Let  $\mathcal{D}_1$  and  $\mathcal{D}_2$  be two databases with values in  $\Delta \cup \text{Var}$

## Definition

A **homomorphism**  $h : \mathcal{D}_1 \rightarrow \mathcal{D}_2$  is a mapping from  $(\Delta \cup \text{Var}(\mathcal{D}_1))$  to  $(\Delta \cup \text{Var}(\mathcal{D}_2))$  such that

- 1  $h(c) = c$ , for every constant  $c \in \Delta$
- 2 each variable can be mapped to a constant or a variable
- 3 for every fact  $r_i(\vec{t})$  of  $\mathcal{D}_1$ , we have that  $r_i(h(\vec{t}))$  is a fact in  $\mathcal{D}_2$

## Definition

$\mathcal{D}_1$  is **homomorphically equivalent** to  $\mathcal{D}_2$  if there is a homomorphism  $h : \mathcal{D}_1 \rightarrow \mathcal{D}_2$  and a homomorphism  $h' : \mathcal{D}_2 \rightarrow \mathcal{D}_1$



# Conjunctive query answering and homomorphisms

Consider a conjunctive query

$$q(\vec{x}) = \{ (\vec{x}) \mid \exists \vec{y}. r_1(\vec{x}_1, \vec{y}_1) \wedge \cdots \wedge r_m(\vec{x}_m, \vec{y}_m) \}$$

For a tuple  $\vec{c}$  of constants, let  $\mathcal{D}_q^{\vec{c}}$  be the set of facts (over constants and variables in  $\vec{y}$ ) obtained by replacing in  $r_1(\vec{x}_1, \vec{y}_1), \dots, r_m(\vec{x}_m, \vec{y}_m)$  each  $x_i$  with  $c_i$ .

Note that  $\mathcal{D}_q^{\vec{c}}$  can be viewed as a database with values in  $\Delta \cup \text{Var}$

**Theorem (Chandra & Merlin '77)**

$\vec{c} \in q^{\mathcal{D}}$  if and only if there exists a homomorphism  $h : \mathcal{D}_q^{\vec{c}} \rightarrow \mathcal{D}$

Hence, **homomorphisms preserve satisfaction of conjunctive queries**:  
if there exists a homomorphism  $h : \mathcal{D} \rightarrow \mathcal{D}'$ , then  $\vec{t} \in q^{\mathcal{D}}$  implies  
 $\vec{t} \in q^{\mathcal{D}'}$ , for each conjunctive query  $q$

# GAV – Universal solutions

Let  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  be a data integration system, and  $\mathcal{C}$  a source db

## Definition

A **universal solution** for  $\mathcal{I}$  relative to  $\mathcal{C}$  is a global db  $\mathcal{B}$  that satisfies  $\mathcal{I}$  relative to  $\mathcal{C}$  such that, for every global db  $\mathcal{B}'$  that satisfies  $\mathcal{I}$  relative to  $\mathcal{C}$ , there exists a homomorphism  $h : \mathcal{B} \rightarrow \mathcal{B}'$  (see [Fagin & al. TCS'05])

## Theorem

*If  $\mathcal{I}$  is GAV without constraints in the global schema, then  $\mathcal{M}(\mathcal{C})$  is the **minimal universal solution** for  $\mathcal{I}$  relative to  $\mathcal{C}$*

We derive again the following results

## Theorem

- *If  $q$  is a conjunctive query, then  $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$  iff  $\vec{t} \in q^{\mathcal{M}(\mathcal{C})}$*
- *Complexity of query answering is polynomial*

# Outline

- 3 Query answering in GAV without constraints
  - Retrieved global database
  - Query answering via unfolding
  - Universal solutions
  
- 4 Query answering in (G)LAV without constraints
  - (G)LAV and incompleteness
  - Approaches to query answering in (G)LAV
  - (G)LAV: Direct methods (aka view-based query answering)
  - (G)LAV: Query answering by (view-based) query rewriting



# (G)LAV data integration systems without constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
<b>no</b>	<b>(G)LAV</b>	yes	no
yes	GAV	yes	yes
yes	(G)LAV	yes	yes





# (G)LAV – Example

Consider  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , with

Global schema  $\mathcal{G}$ :        student(*Code*, *Name*, *City*)  
                              enrolled(*Scode*, *Ucode*)

Source schema  $\mathcal{S}$ :    relation     $s_1(\textit{Scode}, \textit{Sname}, \textit{City}, \textit{Age})$

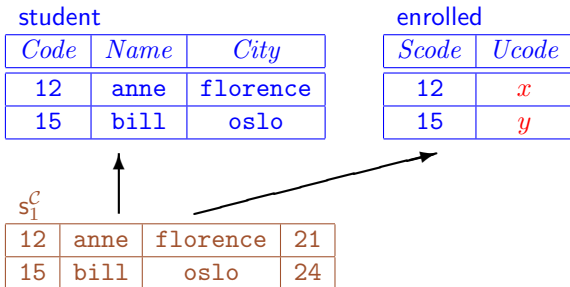
Mapping  $\mathcal{M}$ :

$$\{ (c, n, ci) \mid s_1(c, n, ci, a) \} \rightsquigarrow \{ (c, n, ci) \mid \text{student}(c, n, ci), \text{enrolled}(c, u) \}$$



# (G)LAV – Example

$$\{ (c, n, ci) \mid s_1(c, n, ci, a) \} \rightsquigarrow \{ (c, n, ci) \mid \text{student}(c, n, ci), \text{enrolled}(c, u) \}$$



A source db  $\mathcal{C}$  and a corresponding possible retrieved global db  $\mathcal{M}(\mathcal{C})$

# (G)LAV – Incompleteness

(G)LAV mapping assertions  $\phi_S \rightsquigarrow \phi_G$  have the logical form:

$$\forall \vec{x}. \phi_S(\vec{x}) \rightarrow \exists \vec{y}. \phi_G(\vec{x}, \vec{y})$$

where  $\phi_S$  and  $\phi_G$  are conjunctions of atoms

Given a source database  $\mathcal{C}$ , in general there are several solutions for a set of (G)LAV assertions (i.e., different databases that are legal wrt  $\mathcal{G}$  that satisfy  $\mathcal{M}$  wrt  $\mathcal{C}$ )

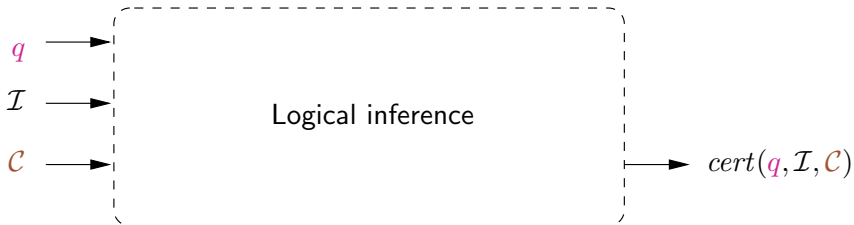
$\rightsquigarrow$  **Incompleteness comes from the mapping**

This holds even for the case of very simple queries  $\phi_G$ :

$$s_1(x) \rightsquigarrow \{ (x) \mid \exists y. g(x, y) \}$$



# (G)LAV – Query answering is based on logical inference



# (G)LAV – Approaches to query answering

- Exploit connection with query containment
- Direct methods (aka **view-based query answering**):  
Try to answer directly the query by means of an algorithm that takes as input the user query  $q$ , the specification of  $\mathcal{I}$ , and the source database  $\mathcal{C}$
- By (view-based) **query rewriting**:
  - 1 Taking into account  $\mathcal{I}$ , reformulate the user query  $q$  as a new query (called a **rewriting** of  $q$ ) over the source relations
  - 2 Evaluate the rewriting over the source database  $\mathcal{C}$

**Note:** In (G)LAV data integration **the views are the sources**



# Connection between query answering and containment

## Definition

**Query containment (under constraints)** is the problem of checking whether  $q_1^{\mathcal{D}}$  is contained in  $q_2^{\mathcal{D}}$  for every database  $\mathcal{D}$  (satisfying the constraints), where  $q_1, q_2$  are queries of the same arity

*Query answering can be rephrased in terms of query containment:*

- A source database  $\mathcal{C}$  can be represented as a conjunction  $q_{\mathcal{C}}$  of ground literals over  $\mathcal{A}_{\mathcal{S}}$  (e.g., if  $\vec{c} \in s^{\mathcal{C}}$ , there is a literal  $s(\vec{c})$ )
- If  $q$  is a query, and  $\vec{t}$  is a tuple, then we denote by  $q_{\vec{t}}$  the query obtained by substituting the free variables of  $q$  with  $\vec{t}$
- The problem of checking whether  $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$  under sound sources can be reduced to the problem of checking whether **the conjunctive query  $q_{\mathcal{C}}$  is contained in  $q_{\vec{t}}$  under the constraints expressed by  $\mathcal{G} \cup \mathcal{M}$**



# Query answering via query containment

Complexity of checking certain answers under sound sources:

- The **combined complexity** is identical to the complexity of query containment under constraints
- The **data complexity** is the complexity of query containment under constraints when the right-hand side query is considered fixed. Hence, it is at most the complexity of query containment under constraints

↪ *Most results and techniques for query containment (under constraints) are relevant also for query answering (under constraints)*

**Note:** Also, query containment can be reduced to query answering. However, (in the presence of constraints) we need to allow for constants of the database to unify, i.e., to denote the same object.



# (G)LAV – Basic technique

From [Duschka & Genesereth PODS'97]:

$$r_1(t) \quad \rightsquigarrow \quad \{ (t) \mid \text{movie}(t, y, d) \wedge \text{european}(d) \}$$

$$r_2(t, v) \quad \rightsquigarrow \quad \{ (t, v) \mid \text{movie}(t, y, d) \wedge \text{review}(t, v) \}$$

$$\forall t. r_1(t) \rightarrow \exists y, d. \text{movie}(t, y, d) \wedge \text{european}(d)$$

$$\forall t, v. r_2(t, v) \rightarrow \exists y, d. \text{movie}(t, y, d) \wedge \text{review}(t, v)$$

$$\begin{aligned} \text{movie}(t, f_1(t), f_2(t)) &\leftarrow r_1(t) \\ \text{european}(f_2(t)) &\leftarrow r_1(t) \\ \text{movie}(t, f_4(t, v), f_5(t, v)) &\leftarrow r_2(t, v) \\ \text{review}(t, v) &\leftarrow r_2(t, v) \end{aligned}$$

*Answering a query means evaluating a goal wrt to this nonrecursive logic program* (which can be transformed into a union of CQs)

$\rightsquigarrow$  **Data complexity is polynomial** (actually LOGSPACE)



# (G)LAV – Canonical retrieved global database

What is a retrieved global database in this case?

## Definition

We build what we call the **canonical retrieved global database** for  $\mathcal{I}$  relative to  $\mathcal{C}$ , denoted  $\mathcal{M}(\mathcal{C})\downarrow$ , as follows:

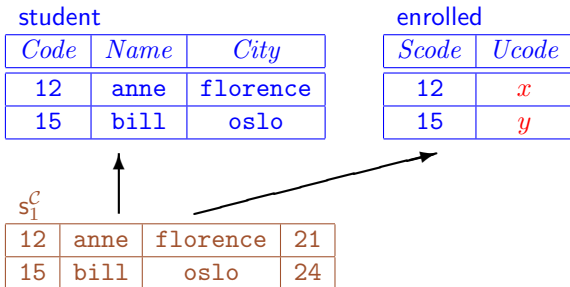
- Let all predicates initially be empty in  $\mathcal{M}(\mathcal{C})\downarrow$
- For each mapping assertion  $\phi_S \rightsquigarrow \phi_G$  in  $\mathcal{M}$ 
  - for each tuple  $\vec{t} \in \phi_S^{\mathcal{C}}$  such that  $\vec{t} \notin \phi_G^{\mathcal{M}(\mathcal{C})\downarrow}$ , add  $\vec{t}$  to  $\phi_G^{\mathcal{M}(\mathcal{C})\downarrow}$  by inventing fresh variables (Skolem terms) in order to satisfy the existentially quantified variables in  $\phi_G$

Properties of  $\mathcal{M}(\mathcal{C})\downarrow$  (also called **canonical model of  $\mathcal{I}$  relative to  $\mathcal{C}$** )

- It is unique up to variable renaming
- It can be computed in polynomial time wrt the size of  $\mathcal{C}$
- Since there are no constraints in  $\mathcal{G}$ , it obviously satisfies  $\mathcal{G}$

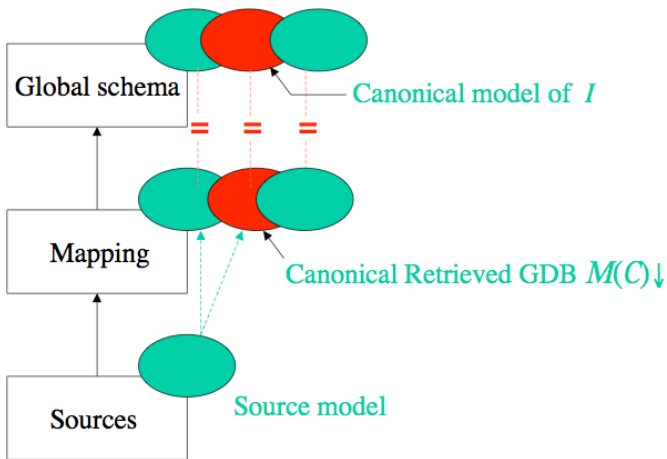
# (G)LAV – Example of canonical model

$$\{ (c, n, ci) \mid s_1(c, n, ci, a) \} \rightsquigarrow \{ (c, n, ci) \mid \text{student}(c, n, ci) \wedge \text{enrolled}(c, u) \}$$



Example of source db  $\mathcal{C}$  and corresponding canonical model  $\mathcal{M}(\mathcal{C}) \downarrow$

# (G)LAV – Canonical model



# (G)LAV – Universal solution

Let  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  be a (G)LAV data integration system without constraints in the global schema, and  $\mathcal{C}$  a source database

## Theorem

Then  $\mathcal{M}(\mathcal{C}) \downarrow$  is a **universal solution** for  $\mathcal{I}$  relative to  $\mathcal{C}$  (follows from [Fagin&al. ICDT'03])

It follows that:

- If  $q$  is a conjunctive query, then  $\vec{t} \in \text{cert}(q, \mathcal{I}, \mathcal{C})$  iff  $\vec{t} \in q^{\mathcal{M}(\mathcal{C}) \downarrow}$
- Complexity of query answering is **polynomial**, actually LOGSPACE



# (G)LAV – More expressive queries?

- More expressive **source queries** in the mapping?
  - Same results hold if we use **any computable query** as source query in the mapping assertions
- More expressive **queries over the global schema** in the mapping?
  - Already positive queries lead to intractability
- More expressive **user queries**?
  - Same results hold if we use **Datalog queries** as user queries
  - Even the simplest form of negation (inequalities) leads to intractability



# (G)LAV – Intractability for positive views

From [van der Meyden TCS'93], by reduction from 3-colorability

We define the following LAV data integration system  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ :

$$\begin{aligned} \mathcal{G} : & \text{edge}(x, y), \quad \text{color}(x, c) & \mathcal{S} : & \text{s}_E(x, y), \quad \text{s}_N(x) \\ \mathcal{M} : & \text{s}_E(x, y) \rightsquigarrow \text{edge}(x, y) \\ & \text{s}_N(x) \rightsquigarrow \text{color}(x, \text{RED}) \vee \text{color}(x, \text{BLUE}) \vee \text{color}(x, \text{GREEN}) \end{aligned}$$

Given a graph  $G = (N, E)$ , we define the following source database  $\mathcal{C}$ :

$$\text{s}_E^{\mathcal{C}} = \{ (a, b), (b, a) \mid (a, b) \in E \} \quad \text{s}_N^{\mathcal{C}} = \{ (a) \mid a \in N \}$$

Consider the boolean query  $q: \exists x, y, c. \text{edge}(x, y) \wedge \text{color}(x, c) \wedge \text{color}(y, c)$  describing mismatched edge pairs:

- If  $G$  is 3-colorable, then  $\exists \mathcal{B}$  s.t.  $q^{\mathcal{B}} = \text{false}$ , hence  $\text{cert}(q, \mathcal{I}, \mathcal{C}) = \text{false}$
- If  $G$  is not 3-colorable, then  $\text{cert}(q, \mathcal{I}, \mathcal{C}) = \text{true}$

## Theorem

*Data complexity is coNP-hard for positive views and conjunctive queries*

# (G)LAV – In coNP for positive views and queries

- $\vec{t} \notin \text{cert}(q, \mathcal{I}, \mathcal{C})$  if and only if there is a database  $\mathcal{B}$  for  $\mathcal{I}$  that satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$ , and such that  $\vec{t} \notin q^{\mathcal{B}}$
- The mapping  $\mathcal{M}$  has the form:

$$\forall \vec{x}. \phi_{\mathcal{S}}(\vec{x}) \rightarrow \exists \vec{y}_1. \alpha_1(\vec{x}, \vec{y}_1) \vee \dots \vee \exists \vec{y}_h. \alpha_h(\vec{x}, \vec{y}_h)$$

Hence, each tuple in  $\mathcal{C}$  forces the existence of  $k$  tuples in any database that satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$ , where  $k$  is the maximal length of conjunctions  $\alpha_i(\vec{x}, \vec{y}_i)$  in  $\mathcal{M}$

- If  $\mathcal{C}$  has  $n$  tuples, then there is a db  $\mathcal{B}' \subseteq \mathcal{B}$  for  $\mathcal{I}$  that satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$  with at most  $n \cdot k$  tuples. Since  $q$  is monotone,  $\vec{t} \notin q^{\mathcal{B}'}$
- Checking whether  $\mathcal{B}'$  satisfies  $\mathcal{M}$  wrt  $\mathcal{C}$ , and checking whether  $\vec{t} \notin q^{\mathcal{B}'}$  can be done in PTIME wrt the size of  $\mathcal{B}'$

## Theorem

For positive views and queries, query answ. is **coNP in data complexity**

## (G)LAV – Conjunctive user queries with inequalities

Consider  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ , and source db  $\mathcal{C}$  (see [Fagin & al. ICDT'03]):

$$\begin{aligned} \mathcal{G} &: g(x, y) & \mathcal{S} &: s(x, y) \\ \mathcal{M} &: s(x, y) \rightsquigarrow \{ (x, y) \mid g(x, z) \wedge g(z, y) \} \\ \mathcal{C} &: \{ s(\mathbf{a}, \mathbf{a}) \} \end{aligned}$$

- $\mathcal{B}_1 = \{ g(\mathbf{a}, \mathbf{a}) \}$  is a solution
- If  $\mathcal{B}$  is a universal solution, then both  $g(\mathbf{a}, x)$  and  $g(x, \mathbf{a})$  are in  $\mathcal{B}$ , with  $x \neq \mathbf{a}$  (otherwise  $g(\mathbf{a}, \mathbf{a})$  would be *true* in every solution)

Let  $q = \{ () \mid g(x, y) \wedge x \neq y \}$

- $q^{\mathcal{B}_1} = \text{false}$ , hence  $\text{cert}(q, \mathcal{I}, \mathcal{C}) = \text{false}$
- But  $q^{\mathcal{B}} = \text{true}$  for every universal solution  $\mathcal{B}$  for  $\mathcal{I}$  relative to  $\mathcal{C}$

Hence, the notion of universal solution is not the right tool



## (G)LAV – Conjunctive user queries with inequalities

- coNP algorithm: guess equalities on variables in the canonical retrieved global database
- coNP-hard already for a conjunctive user query with one inequality (and conjunctive view definitions) [Abiteboul & Duschka PODS'98]

### Theorem

*For conjunctive user queries with inequalities, (G)LAV query answering is **coNP-complete in data complexity***

*Note:* inequalities in the view definitions do not affect expressive power and complexity (in fact, they can be removed)

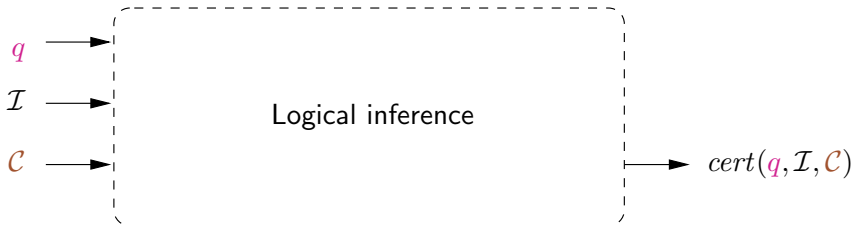
# (G)LAV – View-based query rewriting

Query answering is divided in two steps:

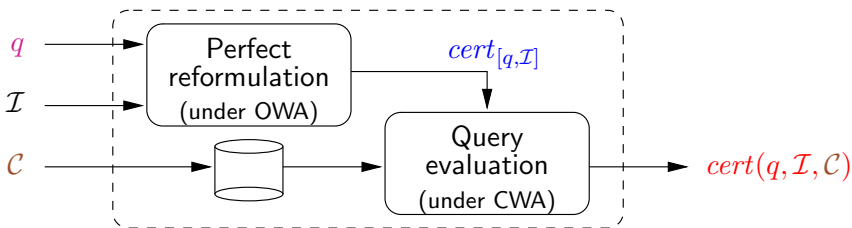
- 1 Re-express the query in terms of a **given query language** over the alphabet of  $\mathcal{A}_S$
- 2 Evaluate the rewriting over the source database  $\mathcal{C}$



# Query answering

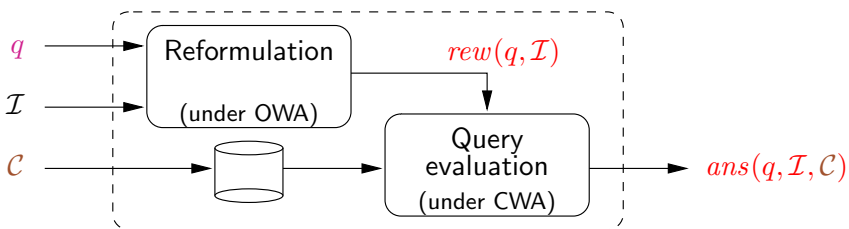


# Query answering: reformulation + evaluation



The query  $cert_{[q, \mathcal{I}]}$  could be expressed in an arbitrary query language

# Query rewriting



The language of  $rew(q, \mathcal{I})$  is chosen a priori!

# (G)LAV – Connection to rewriting

## Query answering by rewriting:

- 1 Given  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  and a query  $q$  over  $\mathcal{G}$ , rewrite  $q$  into a query, called  $rew(q, \mathcal{I})$ , over the alphabet  $\mathcal{A}_{\mathcal{S}}$  of the sources
- 2 Evaluate the rewriting  $rew(q, \mathcal{I})$  over the source database  $\mathcal{C}$

We are interested in rewritings that are:

- **sound**, i.e., compute only tuples in  $cert(q, \mathcal{I}, \mathcal{C})$  for every  $\mathcal{C}$
- expressed in a given **query language**  $\mathcal{L}$
- **maximal** for the class of queries expressible in  $\mathcal{L}$

We may be interested in **exact** rewritings, i.e., rewritings that are logically equivalent to the query, modulo  $\mathcal{M}$

Exact rewritings may not exist



# (G)LAV – Example of maximal rewriting

$\mathcal{G}$ :  $\text{nonstop}(Airline, Num, From, To)$

$\mathcal{S}$ :  $\text{flightsByUnited}(From, To)$   
 $\text{flightsFromSFO}(Airline, Num, To)$

$\mathcal{M}$ :  $\text{flightsByUnited}(From, To) \rightsquigarrow$   
 $\text{nonstop}(UA, Num, From, To)$   
 $\text{flightsFromSFO}(Airline, Num, To) \rightsquigarrow$   
 $\text{nonstop}(Airline, Num, SFO, To)$

$q$ :  $\{ (airline, num) \mid \text{nonstop}(airline, num, LAX, PHX) \}$

A maximal (wrt positive queries) rewriting of  $q$  is:

$\{ (UA, num) \mid \text{flightsByUnited}(num, LAX, PHX) \}$



# Perfect rewriting

What is the relationship between answering by rewriting and certain answers? [— & al. ICDT'05]:

- When does the (maximal) rewriting compute **all** certain answers?
- What do we gain or loose by focusing on a given class of queries?

Let's try to consider the "**best possible**" rewriting

Define  $cert_{[q,\mathcal{I}]}(\cdot)$  to be the function that, with  $q$  and  $\mathcal{I}$  fixed, given source database  $\mathcal{C}$ , computes the certain answers  $cert(q, \mathcal{I}, \mathcal{C})$ .

- $cert_{[q,\mathcal{I}]}$  can be seen as a query on the alphabet  $\mathcal{A}_{\mathcal{S}}$
- $cert_{[q,\mathcal{I}]}$  is a (sound) rewriting of  $q$  wrt  $\mathcal{I}$
- No sound rewriting exists that is better than  $cert_{[q,\mathcal{I}]}$

Hence,  $cert_{[q,\mathcal{I}]}$  is called the **perfect rewriting** of  $q$  wrt  $\mathcal{I}$



# Properties of the perfect rewriting

- Can the perfect rewriting be expressed in a certain query language?
- For a given class of queries, what is the relationship between a maximal rewriting and the perfect rewriting?
  - From a semantical point of view
  - From a computational point of view
- Which is the computational complexity of finding the perfect rewriting, and how big is it?
- Which is the computational complexity of evaluating the perfect rewriting?



## (G)LAV – The case of conjunctive queries

Let  $q$  and the queries in  $\mathcal{M}$  be conjunctive queries (CQs)

Let  $q'$  be the **union of all maximal rewritings of  $q$  for the class of CQs**

Theorem (Levy & al. PODS'95, Abiteboul & Duschka PODS'98)

- $q'$  is the maximal rewriting for the class of unions of conjunctive queries (UCQs)
- $q'$  is the **perfect rewriting of  $q$  wrt  $\mathcal{I}$**
- $q'$  is a PTIME query
- $q'$  is an exact rewriting (equivalent to  $q$  for each database  $\mathcal{B}$  of  $\mathcal{I}$ ), if an exact rewriting exists

Does this “ideal situation” carry on to cases where  $q$  and  $\mathcal{M}$  allow for union?



# (G)LAV – View-based query processing for UPQs

When queries over the global schema in the mapping contain **union**:

- We have seen that view-based query answering is coNP-complete in data complexity [van der Meyden TCS'93]
- hence,  $\text{cert}(q, \mathcal{I}, \mathcal{C})$ , with  $q, \mathcal{I}$  fixed, is a coNP-complete function
- hence, **the perfect rewriting  $\text{cert}_{[q, \mathcal{I}]}$  is a coNP-complete query**

We do not have the ideal situation we had for conjunctive queries

**Problem:** Isolate those cases of view based query rewriting for UPQs  $q$  and  $\mathcal{I}$  for which the perfect rewriting  $\text{cert}_{[q, \mathcal{I}]}$  is a PTIME function (assuming  $P \neq NP$ ) [— & al. LICS'00].

# (G)LAV – Data complexity of query answering

From [Abiteboul & Duschka PODS'98], for sound sources:

Global schema mapping query	User queries				
	CQ	CQ $\neq$	PQ	Datalog	FOL
CQ	PTIME	coNP	PTIME	PTIME	undec.
CQ $\neq$	PTIME	coNP	PTIME	PTIME	undec.
PQ	coNP	coNP	coNP	coNP	undec.
Datalog	coNP	undec.	coNP	undec.	undec.
FOL	undec.	undec.	undec.	undec.	undec.



## (G)LAV – Further references

- Inverse rules [Duschka & Genesereth PODS'97]
- Bucket algorithm for query rewriting [Levy & al. AAAI'96]
- MiniCon algorithm for query rewriting [Pottinger & Levy VLDB'00]
- Conjunctive queries using conjunctive views [Levy & al. PODS'95]
- Recursive queries (Datalog programs) using conjunctive views [Duschka & Genesereth PODS'97; Afrati & al. ICDT'99]
- CQs with arithmetic comparison [Afrati & al. PODS'01]
- Complexity analysis [Abiteboul & Duschka PODS'98; Grahne & Mendelzon ICDT'99]
- Variants of Regular Path Queries [— & al. ICDE'00, PODS'00, DBPL'01; Deutsch & Tannen DBPL'01],
- Relationship between view-based rewriting and answering [— & al. LICS'00, PODS'03, ICDT'05]



## Part III

# Query answering in the presence of constraints



# Outline

- 5 The role of global integrity constraints
- 6 Query answering in GAV with constraints
  - Incompleteness and inconsistency in GAV systems
  - Query answering in GAV under inclusion dependencies
  - Rewriting CQs under inclusion dependencies in GAV
  - Query answering in GAV under IDs and KDs
  - Query answering in GAV under IDs, KDs, and EDs
- 7 Query answering in LAV with constraints
  - LAV systems and integrity constraints
  - Query answering in (G)LAV under inclusion dependencies
  - Query answering in (G)LAV under IDs and EDs
  - LAV systems and key dependencies

















# Outline

- 5 The role of global integrity constraints
- 6 Query answering in GAV with constraints**
  - Incompleteness and inconsistency in GAV systems
  - Query answering in GAV under inclusion dependencies
  - Rewriting CQs under inclusion dependencies in GAV
  - Query answering in GAV under IDs and KDs
  - Query answering in GAV under IDs, KDs, and EDs
- 7 Query answering in LAV with constraints
  - LAV systems and integrity constraints
  - Query answering in (G)LAV under inclusion dependencies
  - Query answering in (G)LAV under IDs and EDs
  - LAV systems and key dependencies



# GAV system with integrity constraints

We consider a data integration system  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  where

- $\mathcal{G}$  is a global schema with constraints
- $\mathcal{M}$  is a set of GAV mappings, whose assertions have the form  $\phi_S \rightsquigarrow g$  and are interpreted as

$$\forall \vec{x}. \phi_S(\vec{x}) \rightarrow g(\vec{x})$$

where  $\phi_S$  is a conjunctive query over  $\mathcal{S}$ , and  $g$  is an element of  $\mathcal{G}$

**Basic observation:** Since  $\mathcal{G}$  does have constraints, the retrieved global database  $\mathcal{M}(\mathcal{C})$  **may not be legal for  $\mathcal{G}$**

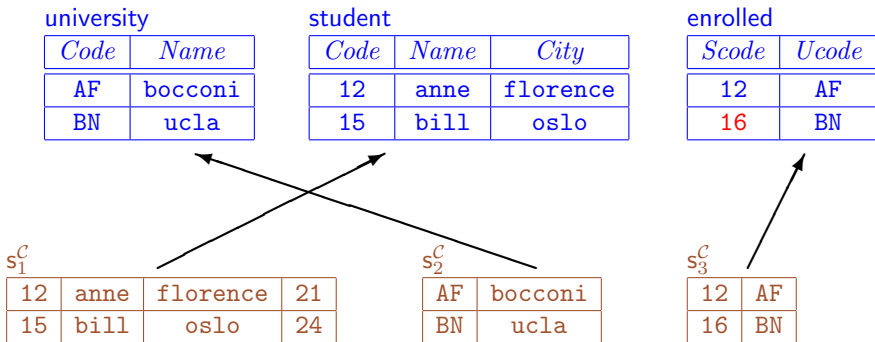








# GAV with constraints – Example of retrieved global db



Example of source database  $\mathcal{C}$  and corresponding retrieved global database  $\mathcal{M}(\mathcal{C})$

# GAV with constraints – Example of incompleteness

 $s_3^C$ 

12	AF
16	BN

 $enrolled^B$ 

<i>Score</i>	<i>Ucode</i>
12	AF
16	BN

 $student^B$ 

<i>Code</i>	<i>Name</i>	<i>City</i>
12	anne	florence
15	bill	oslo
16	<i>x</i>	<i>y</i>

$s_3^C(16, BN)$  and the mapping imply  $enrolled^B(16, BN)$  for all  $B \in sem^C(\mathcal{I})$

Due to the inclusion dependency  $enrolled[Score] \subseteq student[Code]$  in  $\mathcal{G}$ , **16** is the code of some student in all  $B \in sem^C(\mathcal{I})$

Since  $C$  does not provide information about name and city of the student with code 16, a global database that is legal for  $\mathcal{I}$  wrt  $C$  may contain arbitrary values for these



# GAV with constraints – Unfolding is not sufficient

Mapping  $\mathcal{M}$ :

$$\begin{aligned} \{ (c, n, ci) \mid s_1(c, n, ci, a) \} &\rightsquigarrow \text{student}(c, n, ci) \\ \{ (c, n) \mid s_2(c, n) \} &\rightsquigarrow \text{university}(c, n) \\ \{ (s, u) \mid s_3(s, u) \} &\rightsquigarrow \text{enrolled}(s, u) \end{aligned}$$
 $s_1^C$ 

12	anne	florence	21
15	bill	oslo	24

 $s_2^C$ 

AF	bocconi
BN	ucla

 $s_3^C$ 

12	AF
16	BN

Consider the query:  $q = \{ (c) \mid \text{student}(c, n, ci) \}$

Unfolding of  $q$  wrt  $\mathcal{M}$ :  $\text{unf}_{\mathcal{M}}(q) = \{ (c) \mid s_1(c, n, ci, a) \}$

The query  $\text{unf}_{\mathcal{M}}(q)$  retrieves from  $\mathcal{C}$  only the answer  $\{12, 15\}$ , while the correct answer would be  $\{12, 15, 16\}$

The simple **unfolding strategy is not sufficient** for GAV with constraints

## GAV with constraints – Example of inconsistency

$s_1^{\mathcal{C}}$

12	anne	florence	21
12	bill	oslo	24

student $^{\mathcal{B}}$

<i>Code</i>	<i>Name</i>	<i>City</i>
12	anne	florence
12	bill	oslo

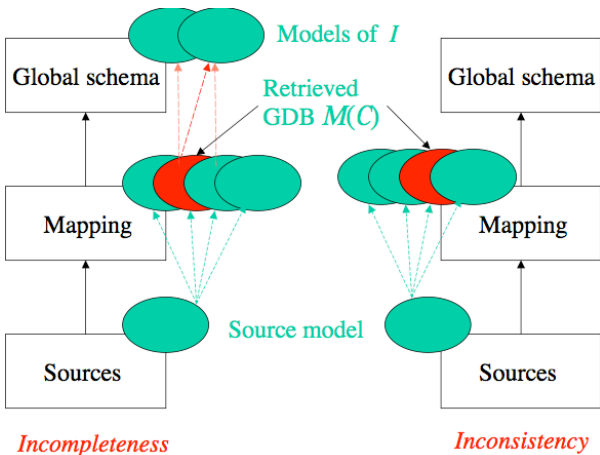
The tuples in  $s_1^{\mathcal{C}}$  and the mapping imply  $\text{student}^{\mathcal{B}}(12, \text{anne}, \text{florence})$  and  $\text{student}^{\mathcal{B}}(12, \text{bill}, \text{oslo})$ , for all  $\mathcal{B}$  that satisfy the mapping

Due to the key dependency  $\text{key}(\text{student}) = \{\text{Code}\}$  in  $\mathcal{G}$ , there is **no global database** that satisfies the mapping and is legal wrt the global schema, i.e.,  $\text{sem}^{\mathcal{I}}(\mathcal{C}) = \emptyset$





# GAV with constraints – Incompleteness and inconsistency



# Inclusion dependencies – Example

Global schema  $\mathcal{G}$ :     $\text{player}(Pname, YOB, Pteam)$   
                                   $\text{team}(Tname, Tcity, Tleader)$

Constraints:     $\text{team}[Tleader, Tname] \subseteq \text{player}[Pname, Pteam]$

Sources  $\mathcal{S}$ :     $s_1$  and  $s_3$  store players  
                           $s_2$  stores teams

Mapping  $\mathcal{M}$ :     $\{ (x, y, z) \mid s_1(x, y, z) \vee s_3(x, y, z) \} \rightsquigarrow \text{player}(x, y, z)$   
                           $\{ (x, y, z) \mid s_2(x, y, z) \} \rightsquigarrow \text{team}(x, y, z)$





# Inclusion dependencies – Example retrieved global db

Source database  $\mathcal{C}$ :

s<sub>1</sub>: 

Totti	1971	Roma
-------	------	------

s<sub>2</sub>: 

Juve	Torino	Del Piero
------	--------	-----------

s<sub>3</sub>: 

Buffon	1978	Juve
--------	------	------

Retrieved global database  $\mathcal{M}(\mathcal{C})$ :

player: 

Totti	1971	Roma
Buffon	1978	Juve

team: 

Juve	Torino	Del Piero
------	--------	-----------



# Inclusion dependencies – Example retrieved global db

player:	Totti	1971	Roma
	Buffon	1978	Juve
	Del Piero	$\alpha$	Juve

team:

Juve	Torino	Del Piero
------	--------	-----------

The ID on the global schema tells us that Del Piero is a player of Juve

All global databases satisfying  $\mathcal{I}$  have **at least** the tuples shown above, where  $\alpha$  is some value of the domain  $\Delta$

## Warnings

- 1 There may be an **infinite number** of databases satisfying  $\mathcal{I}$
- 2 In case of cyclic IDs, databases satisfying  $\mathcal{I}$  may be of **infinite size**



# Inclusion dependencies – Example retrieved global db

player:

Totti	1971	Roma
Buffon	1978	Juve
Del Piero	$\alpha$	Juve

team:

Juve	Torino	Del Piero
------	--------	-----------

The ID on the global schema tells us that Del Piero is a player of Juve

All global databases satisfying  $\mathcal{I}$  have **at least** the tuples shown above, where  $\alpha$  is some value of the domain  $\Delta$

Consider the query  $q = \{ (x, z) \mid \text{player}(x, y, z) \}$

$\text{cert}(q, \mathcal{I}, \mathcal{C}) = \{ (\text{Totti}, \text{Roma}), (\text{Buffon}, \text{Juve}), (\text{Del Piero}, \text{Juve}) \}$







# The ID-chase rule

The chase for IDs has only one rule, the **ID-chase rule**

Let  $\mathcal{D}$  be a database:

**if** the schema contains the ID  $r[i_1, \dots, i_k] \subseteq s[j_1, \dots, j_k]$

**and** there is a fact in  $\mathcal{D}$  of the form  $r(a_1, \dots, a_n)$

**and** there are no facts in  $\mathcal{D}$  of the form  $s(b_1, \dots, b_m)$

such that  $a_{i_\ell} = b_{j_\ell}$  for each  $\ell \in \{1, \dots, k\}$ ,

**then** add to  $\mathcal{D}$  the fact  $s(c_1, \dots, c_m)$ ,

where for each  $h \in \{1, \dots, m\}$ ,

if  $h = j_\ell$  for some  $\ell$  then  $c_h = a_{i_\ell}$

otherwise  $c_h$  is a new constant symbol (not in  $\mathcal{D}$  yet)

**Notice:** New existential symbols are introduced (skolem terms)



# Properties of the chase

- Bad news: the chase is in general **infinite**
  
- Good news: the chase identifies a **canonical model**  
A canonical model is a database that “represents” all the models of the system
  
- We can use the chase to prove soundness and completeness of a query processing method
  
- ... but **only for positive queries!**



# Limiting the chase

Why don't we use a finite number of existential constants in the chase?

## Example

Consider  $r[1] \subseteq s[1]$ , and  $s[2] \subseteq r[1]$  and suppose  $\mathcal{M}(\mathcal{C}) = \{ r(a, b) \}$

Compute  $\text{chase}(\mathcal{M}(\mathcal{C}))$  with only one new constant  $c_1$ :

0)  $r(a, b)$ ; 1) add  $s(a, c_1)$  2) add  $r(c_1, c_1)$  3) add  $s(c_1, c_1)$

This database is **not** a canonical model for  $\mathcal{I}$  wrt  $\mathcal{C}$

E.g., for query  $q = \{ (x) \mid r(x, y), s(y, y) \}$ , we have  $a \in q^{\text{chase}(\mathcal{M}(\mathcal{C}))}$

while  $a \notin \text{cert}(q, \mathcal{I}, \mathcal{C})$

Arbitrarily limiting the chase is **unsound**, for **any** finite number of new constants



# Chasing the query

When chasing the data the termination condition would need to take into account the query

We consider an alternative approach, based on the idea of a **query chase**

- Instead of chasing the data, we chase the query
- Is the dual notion of the database chase
- IDs are applied from right to left to the query atoms
- Advantage: much easier termination conditions, which imply:
  - decidability properties
  - efficiency

This technique provides an algorithm for rewriting UCQs under IDs

# Query rewriting under inclusion dependencies

- Given a query  $q$  over the global schema  $\mathcal{G}$ , we look for a rewriting  $rew$  of  $q$  expressed over  $\mathcal{S}$
- A rewriting  $rew$  is **perfect** if  $rew^{\mathcal{C}} = cert(q, \mathcal{I}, \mathcal{C})$ , for every source database  $\mathcal{C}$
- With a perfect rewriting, we can do **query answering by rewriting**  
 $\rightsquigarrow$  We avoid actually constructing the retrieved global database  $\mathcal{M}(\mathcal{C})$



# Rewriting rule for inclusion dependencies

**Intuition:** Use the IDs as basic rewriting rules

## Example

Consider a query  $q = \{ (x, z) \mid \text{player}(x, y, z) \}$

and the constraint  $\text{team}[Tleader, Tname] \subseteq \text{player}[Pname, Pteam]$

**as a logic rule:**  $\text{player}(w_3, w_4, w_1) \leftarrow \text{team}(w_1, w_2, w_3)$

We add to the rewriting the query  $q' = \{ (x, z) \mid \text{team}(x, y, z) \}$

## Definition

**Basic rewriting step:**

**when** an atom unifies with the **head** of the rule

**substitute** the atom with the **body** of the rule

# Query Rewriting for IDs – Algorithm *ID-rewrite*

Iterative execution of:

## 1 Reduction:

- Atoms that unify with other atoms are eliminated and the unification is applied
- Variables that appear only once are marked

## 2 Basic rewriting step

- A rewriting step is applicable to an atom if it does not eliminate variables that appear somewhere else
- May introduce fresh variables

**Note:** The algorithm works directly for unions of conjunctive queries (UCQs), and produces an UCQ as result



# The algorithm *ID-rewrite*

**Input:** relational schema  $\mathcal{G}$ , set  $\Psi_{ID}$  of IDs, UCQ  $Q$

**Output:** perfect rewriting of  $Q$

$Q' := Q;$

**repeat**

$Q_{aux} := Q';$

**for each**  $q \in Q_{aux}$  **do**

        (a) **for each**  $g_1, g_2 \in body(q)$  **do**

**if**  $g_1$  and  $g_2$  unify **then**  $Q' := Q' \cup \{\tau(reduce(q, g_1, g_2))\};$

        (b) **for each**  $g \in body(q)$  **do**

**for each**  $ID \in \Psi_{ID}$  **do**

**if**  $ID$  is applicable to  $g$

**then**  $Q' := Q' \cup \{q[g/rewrite(g, ID)]\}$

**until**  $Q_{aux} = Q';$

**return**  $Q'$



# Query answering in GAV under IDs

## Properties of *ID-rewrite*

- *ID-rewrite* terminates
- *ID-rewrite* produces a perfect rewriting of the input query

More precisely, let  $unf_{\mathcal{M}}(q)$  be the **unfolding** of the query  $q$  wrt the GAV mapping  $\mathcal{M}$

### Theorem

$unf_{\mathcal{M}}(ID\text{-rewrite}(q))$  is a perfect rewriting of the query  $q$

### Theorem

Query answering in GAV systems under IDs is in PTIME in data complexity (actually in LOGSPACE)

# Query answering under IDs and KDs

We have already seen that in GAV systems under sound mappings

- Key dependencies may give rise to inconsistencies
- When  $\mathcal{M}(\mathcal{C})$  violates the KDs, no legal database exists and **query answering becomes trivial**

How do KDs interact with IDs?

## Theorem

*Query answering under IDs and KDs is undecidable*

*Proof:* By reduction from implication of IDs and KDs

We need to look for **syntactic restrictions** on the form of the dependencies that ensures decidability



# Non-key-conflicting IDs

## Definition

**Non-key-conflicting IDs** (NKCIDs) are of the form  $r_1[\vec{x}_1] \subseteq r_2[\vec{x}_2]$  where  $\vec{x}_2$  is **not a strict superset** of  $key(r_2)$

## Example

Let  $r$  be of arity 3 and  $s$  of arity 4 with  $key(s) = \{1, 2\}$

- The following are NKCIDs
  - $r[2] \subseteq s[2]$ , since  $\{2\}$  is a strict subset of  $key(s)$
  - $r[2, 3] \subseteq s[1, 2]$ , since  $\{1, 2\}$  coincides with  $key(s)$
  - $r[1, 2] \subseteq s[2, 3]$ , since  $1 \in key(s)$  but  $1 \notin \{2, 3\}$
- The following is not a NKCID:  $r[1, 2, 3] \subseteq s[1, 2, 4]$

**Note:** **Foreign keys** (FKs) are a special case of NKCIDs





# Separation for IDs and KDs

## Theorem (IDs-KDs separation)

Under KDs and NKCIDs, if  $\mathcal{M}(\mathcal{C})$  satisfies the KDs, then the **KDs can be ignored** wrt certain answers of a user query  $q$

**Intuition:** For NKCIDs, when applying the ID-chase rule to a tuple  $\vec{t}_1 \in r_1^{\mathcal{B}}$ , we can choose the tuple  $\vec{t}_2$  to introduce in  $r_2^{\mathcal{B}}$  so that it does not violate  $\text{key}(r_2)$ :

- When  $\text{key}(r_2) \not\subseteq \vec{x}_2$ , fresh constants in  $\vec{t}_2$  are chosen for key attributes, and so there is no other tuple in  $r_2^{\mathcal{B}}$  coinciding with  $\vec{t}_2$  on all key attributes
- When  $\text{key}(r_2) = \vec{x}_2$ , if there is already a tuple  $\vec{t}$  in  $r_2^{\mathcal{B}}$  such that  $\vec{t}_1[\vec{x}_1] = \vec{t}[\vec{x}_2]$ , we choose  $\vec{t}$  for  $\vec{t}_2$

Query answering becomes **undecidable** as soon as we extend the language of the IDs



# Query processing under separable KDs and IDs

Global algorithm:

- ① Verify consistency of  $\mathcal{M}(\mathcal{C})$  with respect to KDs
- ② Compute *ID-rewrite* of the input query
- ③ Unfold wrt  $\mathcal{M}$  the query computed at previous step
- ④ Evaluate the unfolded query over the sources

*Note:*

- The KD consistency check can be done by suitable CQs with inequality
- The computation of  $\mathcal{M}(\mathcal{C})$  can be avoided (by unfolding the queries for the KD consistency check)



## Checking KD consistency – Example

Relation:  $\text{player}[Pname, Pteam]$

Key dependency:  $\text{key}(\text{player}) = \{Pname\}$

Query to check (in)consistency of the KD:

$$q = \{ () \mid \text{player}(x, y), \text{player}(x, z), y \neq z \}$$

is *true* iff the instance of *player* violates the KD

Mapping  $\mathcal{M}$ :  $\{ (x, y) \mid s_1(x, y) \vee s_2(x, y) \} \rightsquigarrow \text{player}(x, y)$

Unfolding of  $q$  wrt  $\mathcal{M}$ :  $\{ () \mid s_1(x, y), s_1(x, z), y \neq z \vee$   
 $s_1(x, y), s_2(x, z), y \neq z \vee$   
 $s_2(x, y), s_1(x, z), y \neq z \vee$   
 $s_2(x, y), s_2(x, z), y \neq z \}$



# Query answering in GAV under separable IDs+KDs

## Theorem (Calì, Lembo & Rosati, PODS'03)

*Answering conjunctive queries in GAV systems under KDs and NKIDs is in PTIME in data complexity (actually in LOGSPACE )*

Can we extend these results to more expressive user queries?

- The rewriting technique extends immediately to unions of CQs  

$$ID\text{-rewrite}(q_1 \vee \dots \vee q_n) = ID\text{-rewrite}(q_1) \vee \dots \vee ID\text{-rewrite}(q_n)$$
- This is not the case for recursive queries

## Theorem (— & Rosati KRDB'03)

*Answering recursive queries under KDs and FKs is undecidable*  
*Answering recursive queries under IDs is undecidable*

# Query answering under IDs and EDs

Under EDs:

- Possibility of inconsistencies
- When  $\mathcal{M}(\mathcal{C})$  violates the EDs, no legal database exists and **query answering becomes trivial**

Under IDs and EDs:

- How do EDs and IDs interact?
- Is query answering separable?
- Is query answering decidable?



# Exclusion dependencies – Example

Global schema  $\mathcal{G}$ :  
 player( $Pname, YOB, Pteam$ )  
 team( $Tname, Tcity, Tleader$ )  
 coach( $Cname, Cteam$ )

Constraints:  
 team[ $Tleader, Tname$ ]  $\subseteq$  player[ $Pname, Pteam$ ]  
 coach[ $Cname$ ]  $\cap$  player[ $Pname$ ] =  $\emptyset$

Sources  $\mathcal{S}$ :  
 $s_1$  and  $s_3$  store players  
 $s_2$  stores teams  
 $s_4$  stores coaches

Mapping  $\mathcal{M}$ :  
 $\{ (x, y, z) \mid s_1(x, y, z) \vee s_3(x, y, z) \} \rightsquigarrow \text{player}(x, y, z)$   
 $\{ (x, y, z) \mid s_2(x, y, z) \} \rightsquigarrow \text{team}(x, y, z)$   
 $\{ (x, y) \mid s_4(x, y, z) \} \rightsquigarrow \text{coach}(x, y)$



# Retrieved global db under EDs – Example

Source database  $\mathcal{C}$ :

$s_1$ : 

Totti	1971	Roma
-------	------	------

$s_2$ : 

Juve	Torino	Del Piero
------	--------	-----------

$s_3$ : 

Buffon	1978	Juve
--------	------	------

$s_4$ : 

Del Piero	Viterbese
-----------	-----------

Retrieved global database  $\mathcal{M}(\mathcal{C})$ :

player :

Totti	1971	Roma
Buffon	1978	Juve

team :

Juve	Torino	Del Piero
------	--------	-----------

coach :

Del Piero	Viterbese
-----------	-----------







## Deductive closure of EDs under IDs – Example

Can we saturate (close) the EDs by adding all the **EDs that are logical consequences** of the EDs and IDs?

### Example

From

$$\text{team}[Tleader, Tname] \subseteq \text{player}[Pname, Pteam]$$

$$\text{coach}[Cname] \cap \text{player}[Pname] = \emptyset$$

it follows that

$$\text{coach}[Cname] \cap \text{team}[Tleader] = \emptyset$$

This constraint is violated by the retrieved global database  $\mathcal{M}(\mathcal{C})$



# Deductive closure of EDs under IDs

## Definition

### Derivation rule of EDs under EDs and IDs:

From the ED  $r[i_1, \dots, i_k] \cap s[j_1, \dots, j_k] = \emptyset$

and the ID  $t[\ell_1, \dots, \ell_k] \subseteq s[j_1, \dots, j_k]$

derive the ED  $r[i_1, \dots, i_k] \cap t[\ell_1, \dots, \ell_k] = \emptyset$

Corresponds to a simple application of **resolution** on the FOL sentences corresponding to EDs and IDs

## Theorem

*If the set of EDs is closed with respect to the above rule, it contains all EDs that are logical consequences of the initial EDs and IDs*



# Query answering in GAV under IDs, KDs and EDs

## Theorem (ID-KD-ED Separation)

*Under KDs, NKCIDs, and EDs,  
if  $\mathcal{M}(\mathcal{C})$  satisfies all the KDs  
and satisfies all EDs derived from the IDs and the original EDs  
then the KDs and EDs can be ignored wrt certain answers of a query*

We obtain a method for query answering in GAV under KDs, NKCIDs, and EDs:

- ① Close the set of EDs with respect to the IDs
- ② Verify consistency of  $\mathcal{M}(\mathcal{C})$  with respect to KDs and EDs
- ③ Compute ID-rewrite of the input query
- ④ Unfold the query computed at the previous step
- ⑤ Evaluate the query over the sources



# Query answ. in GAV under IDs, KDs and EDs – Complexity

## Note:

- 1 Closing the set of EDs wrt the IDs is independent of the data
- 2 Consistency of  $\mathcal{M}(\mathcal{C})$  wrt KDs and EDs can be verified through suitable queries over the source database  $\mathcal{C}$

## Theorem (Lembo & Rosati, 2004)

*Answering conjunctive queries in GAV systems under KDs, NKCIDs and EDs is in PTIME in data complexity (actually in LOGSPACE )*



# Outline

- 5 The role of global integrity constraints
- 6 Query answering in GAV with constraints
  - Incompleteness and inconsistency in GAV systems
  - Query answering in GAV under inclusion dependencies
  - Rewriting CQs under inclusion dependencies in GAV
  - Query answering in GAV under IDs and KDs
  - Query answering in GAV under IDs, KDs, and EDs
- 7 Query answering in LAV with constraints**
  - LAV systems and integrity constraints
  - Query answering in (G)LAV under inclusion dependencies
  - Query answering in (G)LAV under IDs and EDs
  - LAV systems and key dependencies



# (G)LAV system with integrity constraints

We consider a data integration system  $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$  where

- $\mathcal{G}$  is a global schema with constraints
- $\mathcal{M}$  is a set of LAV mappings, whose assertions have the form  $\phi_{\mathcal{S}} \rightsquigarrow \phi_{\mathcal{G}}$  and are interpreted as

$$\forall \vec{x}. \phi_{\mathcal{S}}(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$$

where  $\phi_{\mathcal{S}}$  is a conjunctive query over  $\mathcal{S}$ , and  $\phi_{\mathcal{G}}$  is a conjunctive query over  $\mathcal{G}$

**Basic observation:** Since  $\mathcal{G}$  does have constraints, the canonical retrieved global database  $\mathcal{M}(\mathcal{C}) \downarrow$  **may not be legal for  $\mathcal{G}$**



# Semantics of (G)LAV systems with integrity constraints

Given a source db  $\mathcal{C}$ , a global db  $\mathcal{B}$  (over  $\Delta$ ) satisfies  $\mathcal{I}$  relative to  $\mathcal{C}$  if

- 1 it is legal wrt the global schema, i.e., it satisfies the ICs
- 2 it satisfies the mapping, i.e.,  $\mathcal{B}$  is a **superset** of the **canonical retrieved global database**  $\mathcal{M}(\mathcal{C}) \downarrow$  (**sound** mappings)

## Recall:

- $\mathcal{M}(\mathcal{C})$  is obtained by evaluating, for each mapping assertion  $\phi_S \rightsquigarrow \phi_G$ , the query  $\phi_S$  over  $\mathcal{C}$ , and using the obtained tuples to populate the global relations according to  $\phi_G$ , using fresh constants for existentially quantified elements
- We are interested in **certain answers** to a query, i.e., those that hold for **all** global databases that satisfy  $\mathcal{I}$  relative to  $\mathcal{C}$



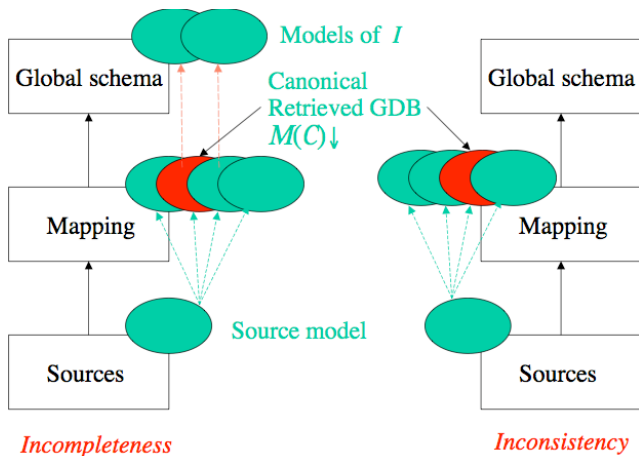


## (G)LAV data integration systems with constraints

Constraints in $\mathcal{G}$	Type of mapping	Incompleteness	Inconsistency
no	GAV	yes / no	no
no	(G)LAV	yes	no
IDs	GAV	yes	no
KDs	GAV	yes / no	yes
IDs + KDs	GAV	yes	yes
<b>IDs</b>	<b>(G)LAV</b>	yes	no
<b>KDs</b>	<b>(G)LAV</b>	yes	yes
<b>IDs + KDs</b>	<b>GAV</b>	yes	yes



# (G)LAV with constr. – Incompleteness and inconsistency





# Transforming LAV into GAV – Example

Initial LAV mappings:

$$s(x, y) \rightsquigarrow \{ (x, y) \mid r_1(x, z), r_2(y, w) \}$$
$$t(x, y) \rightsquigarrow \{ (x, y) \mid r_1(x, z), r_3(y, x) \}$$

We introduce two new global relations for each mapping assertion:

$s_i/2$ ,  $s_e/4$ , and  $t_i/2$ ,  $t_e/3$

Transformed GAV mappings:

$$\{ (x, y) \mid s(x, y) \} \rightsquigarrow s_i(x, y)$$
$$\{ (x, y) \mid t(x, y) \} \rightsquigarrow t_i(x, y)$$

Additional IDs generated by the transformation:

$$s_i[1, 2] \subseteq s_e[1, 2] \quad s_e[1, 3] \subseteq r_1[1, 2] \quad s_e[2, 4] \subseteq r_2[1, 2]$$
$$t_i[1, 2] \subseteq t_e[1, 2] \quad t_e[1, 3] \subseteq r_1[1, 2] \quad t_e[2, 1] \subseteq r_3[1, 2]$$




# (G)LAV systems under IDs and EDs

What happens if we have also EDs in the global schema?

- The above transformation of (G)LAV into GAV is still correct in the presence of EDs
- It is thus possible to first turn the (G)LAV system into a GAV one and then compute query answering in the transformed system
- The addition of EDs is completely modular (we just need to add auxiliary steps in the query answering technique)













# Data integration with constraints – Complexity results

EDs	KDs	IDs	Data/Combined complexity
no	no	general	PSPACE/PSPACE
yes-no	yes	no	PSPACE/NP
yes	yes-no	no	PSPACE/NP
yes-no	yes	NKC	PSPACE/PSPACE
yes	no	general	PSPACE/PSPACE
yes-no	yes	1KC	undecidable
yes-no	yes	general	undecidable

## Part IV

### Concluding remarks



# Outline

## 8 Concluding remarks



# Outline

## 8 Concluding remarks



## Further issues and open problems

- Further forms of constraints, e.g.,
  - KDs with restricted forms of key-conflicting IDs
  - ontology languages, description logics, RDF  
[— & al. PODS'98, — & al., KR'06]
- Semistructured data and XML
  - constraints (DTDs, XML Schema, ...)
  - query languages (transitive closure)
- Finite models vs. unrestricted models [Rosati, PODS'06]
- Data exchange and materialization





# Acknowledgements

- Andrea Cali
- Giuseppe De Giacomo
- Domenico Lembo
- Maurizio Lenzerini
- Riccardo Rosati
- Moshe Y. Vardi









# References I

-  S. Abiteboul and O. Duschka.  
Complexity of answering queries using materialized views.  
*In Proc. of PODS'98*, pages 254–265, 1998.
-  A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini.  
On the expressive power of data integration systems.  
*In Proc. of ER 2002*, 2002.
-  A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini.  
On the role of integrity constraints in data integration.  
*IEEE Bull. on Data Engineering*, 25(3):39–45, 2002.
-  A. Calì, D. Lembo, and R. Rosati.  
Query rewriting and answering under constraints in data integration systems.  
*In Proc. of IJCAI 2003*, pages 16–21, 2003.







# References II

-  D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati.  
Data complexity of query answering in description logics.  
*In Proc. of KR 2006*, 2006.
-  D. Calvanese, G. De Giacomo, and M. Lenzerini.  
On the decidability of query containment under constraints.  
*In Proc. of PODS'98*, pages 149–158, 1998.
-  D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi.  
View-based query processing: On the relationship between rewriting, answering and losslessness.  
*In Proc. of ICDT 2005*, volume 3363 of *LNCS*, pages 321–336. Springer, 2005.
-  D. Calvanese and R. Rosati.  
Answering recursive queries under keys and foreign keys is undecidable.  
*In Proc. of KRDB 2003*. CEUR Electronic Workshop Proceedings,  
<http://ceur-ws.org/Vol-79/>, 2003.



# References III

-  O. M. Duschka, M. R. Genesereth, and A. Y. Levy.  
Recursive query plans for data integration.  
*J. of Logic Programming*, 43(1):49–73, 2000.
-  A. Fuxman and R. J. Miller.  
First-order query rewriting for inconsistent databases.  
In *Proc. of ICDT 2005*, volume 3363 of *LNCS*, pages 337–351. Springer, 2005.
-  G. Grahne and A. O. Mendelzon.  
Tableau techniques for querying information sources through global schemas.  
In *Proc. of ICDT'99*, volume 1540 of *LNCS*, pages 332–347. Springer, 1999.
-  A. Y. Halevy.  
Answering queries using views: A survey.  
*VLDB Journal*, 10(4):270–294, 2001.



# References IV



M. Lenzerini.

Data integration: A theoretical perspective.

In *Proc. of PODS 2002*, pages 233–246, 2002.



N. Leone, T. Eiter, W. Faber, M. Fink, G. Gottlob, G. Greco, E. Kalka, G. Ianni, D. Lembo, V. Lio, B. Nowicki, R. Rosati, M. Ruzzi, W. Staniszkis, and G. Terracina.

Boosting information integration: The INFOMIX system.

In *Proc. of SEBD 2005*, pages 55–66, 2005.



A. Y. Levy, A. Rajaraman, and J. J. Ordille.

Query answering algorithms for information agents.

In *Proc. of AAAI'96*, pages 40–47, 1996.



# References V



R. Rosati.

On the decidability and finite controllability of query processing in databases with incomplete information.

In *Proc. of PODS 2006*, 2006.

