Data Integration: A Theoretical Perspective

Maurizio Lenzerini
Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lenzerini@dis.uniroma1.it

ABSTRACT

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data. The problem of designing data integration systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view. This document presents on overview of the material to be presented in a tutorial on data integration. The tutorial is focused on some of the theoretical issues that are relevant for data integration. Special attention will be devoted to the following aspects: modeling a data integration application, processing queries in data integration, dealing with inconsistent data sources, and reasoning on queries.

1. INTRODUCTION

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [60, 61, 89]. The problem of designing data integration systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view. This tutorial is focused on some of these theoretical issues, with special emphasis on the following topics.

The data integration systems we are interested in this work are characterized by an architecture based on a global schema and a set of sources. The sources contain the real data, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. Modeling the relation between the sources and the global schema is therefore a crucial aspect. Two basic approaches have been proposed to this purpose. The first approach, called global-as-view, requires that the global schema is expressed in terms of the data sources. The second approach, called local-as-view, requires the global schema to be specified independently from the sources, and the relationships between the global schema and the sources are established by defining every source as a view over the global schema. Our goal is to discuss the characteristics of the two modeling mechanisms, and to mention other possible approaches.

Irrespectively of the method used for the specification of the mapping between the global schema and the sources, one basic service provided by the data integration system is to answer queries posed in terms of the global schema. Given the architecture of the system, query processing in data integration requires a reformulation step: the query over the global schema has to be reformulated in terms of a set of queries over the sources. In this tutorial, such a reformulation problem will be analyzed for both the case of local-as-view, and the case of global-as-view mappings. A main theme will be the strong relationship between query processing in data integration and the problem of query answering with incomplete information.

Since sources are in general autonomous, in many real-world applications the problem arises of mutually inconsistent data sources. In practice, this problem is generally dealt with by means of suitable transformation and cleaning procedures applied to data retrieved from the sources. In this tutorial, we address this issue from a more theoretical perspective.

Finally, there are several tasks in the operation of a data integration system where the problem of reasoning on queries (e.g., checking whether two queries are equivalent) is relevant. Indeed, query containment is one of the basic problems in database theory, and we will discuss several notions generalizing this problem to a data integration setting.

The paper is organized as follows. Section 2 presents our formalization of a data integration system. In Section 3 we discuss the various approaches to modeling. Sections 4 and 5 present an overview of the methods for processing queries in the local-as-view and in the global-as-view approach, respectively. Section 6 discusses the problem of dealing with inconsistent sources. Section 7 provides an overview on the problem of reasoning on queries. Finally, Section 8 concludes the paper by mentioning some open problems, and several research issues related to data integration that are not addressed in the tutorial.

2. DATA INTEGRATION FRAMEWORK

In this section we set up a logical framework for data integration. We restrict our attention to data integration systems

ACM PODS 2002 June 3-6, Madison, Wisconsin, USA

based on a so-called global schema (or, mediated schema). In other words, we refer to data integration systems whose aim is combining the data residing at different sources, and providing the user with a unified view of these data. Such a unified view is represented by the global schema, and provides a reconciled view of all data, which can be queried by the user. Obviously, one of the main task in the design of a data integration system is to establish the mapping between the sources and the global schema, and such a mapping should be suitably taken into account in formalizing a data integration system.

It follows that the main components of a data integration system are the global schema, the sources, and the mapping. Thus, we formalize a *data integration system* $\mathcal{I}$ in terms of a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where

- $\mathcal{G}$ is the *global schema*, expressed in a language $\mathcal{L}_\mathcal{G}$ over an alphabet $\mathcal{A}_\mathcal{G}$. The alphabet comprises a symbol for each element of $\mathcal{G}$ (i.e., relation if $\mathcal{G}$ is relational, class if $\mathcal{G}$ is object-oriented, etc.).
- $\mathcal{S}$ is the *source schema*, expressed in a language $\mathcal{L}_\mathcal{S}$ over an alphabet $\mathcal{A}_\mathcal{S}$. The alphabet $\mathcal{A}_\mathcal{S}$ includes a symbol for each element of the sources.
- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, constituted by a set of *assertions* of the forms

$$q_\mathcal{S} \rightsquigarrow q_\mathcal{G},$$
$$q_\mathcal{G} \rightsquigarrow q_\mathcal{S}$$

  where $q_\mathcal{S}$ and $q_\mathcal{G}$ are two queries of the same arity, respectively over the source schema $\mathcal{S}$, and over the global schema $\mathcal{G}$. Queries $q_\mathcal{S}$ are expressed in a query language $\mathcal{L}_{\mathcal{M},\mathcal{S}}$ over the alphabet $\mathcal{A}_\mathcal{S}$, and queries $q_\mathcal{G}$ are expressed in a query language $\mathcal{L}_{\mathcal{M},\mathcal{G}}$ over the alphabet $\mathcal{A}_\mathcal{G}$. Intuitively, an assertion $q_\mathcal{S} \rightsquigarrow q_\mathcal{G}$ specifies that the concept represented by the query $q_\mathcal{S}$ over the sources corresponds to the concept in the global schema represented by the query $q_\mathcal{G}$ (similarly for an assertion of type $q_\mathcal{G} \rightsquigarrow q_\mathcal{S}$). We will discuss several ways to make this intuition precise in the following sections.

Intuitively, the source schema describes the structure of the sources, where the real data are, while the global schema provides a reconciled, integrated, and virtual view of the underlying sources. The assertions in the mapping establish the connection between the elements of the global schema and those of the source schema.

Queries to $\mathcal{I}$ are posed in terms of the global schema $\mathcal{G}$, and are expressed in a query language $\mathcal{L}_\mathcal{Q}$ over the alphabet $\mathcal{A}_\mathcal{G}$. A query is intended to provide the specification of which data to extract from the virtual database represented by the integration system.

The above definition of data integration system is general enough to capture virtually all approaches in the literature. Obviously, the nature of a specific approach depends on the characteristics of the mapping, and on the expressive power of the various schema and query languages. For example, the language $\mathcal{L}_\mathcal{G}$ may be very simple (basically allowing the definition of a set of relations), or may allow for various forms of integrity constraints to be expressed over the symbols of $\mathcal{A}_\mathcal{G}$. Analogously, the type (e.g., relational, semistructured, etc.) and the expressive power of $\mathcal{L}_\mathcal{S}$ varies from one approach to another.

We now specify the semantics of a data integration system. In what follows, a database (DB) for a schema $\mathcal{T}$ is simply a set of collection of sets, one for each symbol in the alphabet of $\mathcal{T}$ (e.g., one relation for every relation schema of $\mathcal{T}$, if $\mathcal{T}$ is relational, or one set of objects for each class of $\mathcal{T}$, if $\mathcal{T}$ is object-oriented, etc.). We also make a simplifying assumption on the domain for the various sets. In particular, we assume that the structures constituting the databases involved in our framework (both the global database and the source databases) are defined over a fixed domain Γ.

In order to assign semantics to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, we start by considering a *source database* for $\mathcal{I}$, i.e., a database $\mathcal{D}$ that conforms to the source schema $\mathcal{S}$ and satisfies all constraints in $\mathcal{S}$. Based on $\mathcal{D}$, we now specify which is the information content of the global schema $\mathcal{G}$. We call *global database* for $\mathcal{I}$ any database for $\mathcal{G}$. A global database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal with respect to* $\mathcal{D}$, if:

- $\mathcal{B}$ is legal with respect to $\mathcal{G}$, i.e., $\mathcal{B}$ satisfies all the constraints of $\mathcal{G}$;
- $\mathcal{B}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{D}$.

The notion of $\mathcal{B}$ satisfying the mapping $\mathcal{M}$ with respect to $\mathcal{D}$ depends on how to interpret the assertions in the mapping. We will see in the next section that several approaches are conceivable. Here, we simply note that, no matter which is the interpretation of the mapping, in general, several global databases exist that are legal for $\mathcal{I}$ with respect to $\mathcal{D}$. This observation motivates the relationship between data integration and databases with incomplete information [91], which will be discussed in several ways later on in the paper.

Finally, we specify the semantics of queries posed to a data integration system. As we said before, such queries are expressed in terms of the symbols in the global schema of $\mathcal{I}$. In general, if q is a query of arity n and $\mathcal{DB}$ is a database, we denote with $q^\mathcal{DB}$ the set of tuples (of arity n) in $\mathcal{DB}$ that satisfy q.

Given a source database $\mathcal{D}$ for $\mathcal{I}$, the answer $q^{\mathcal{I},\mathcal{D}}$ to a query q in $\mathcal{I}$ with respect to $\mathcal{D}$, is the set of tuples t of objects in Γ such that $t \in q^\mathcal{B}$ for *every* global database $\mathcal{B}$ that is legal for $\mathcal{I}$ with respect to $\mathcal{D}$. The set $q^{\mathcal{I},\mathcal{D}}$ is called the set of *certain answers* to q in $\mathcal{I}$ with respect to $\mathcal{D}$.

Note that, from the point of view of logic, finding certain answers is a logical implication problem: check whether it logically follows from the information on the sources that t satisfies the query. The dual problem is also of interest: finding the so-called *possible answers* to q, i.e., checking whether $t \in q^\mathcal{B}$ for some global database $\mathcal{B}$ that is legal for $\mathcal{I}$ with respect to $\mathcal{D}$. Finding possible answers is a consistency problem: check whether assuming that t is in the answer set of q does not contradict the information on the sources.

3. MODELING

One of the most important aspects in the design of a data integration system is the specification of the correspondence between the data at the sources and those in the global schema. Such a correspondence is modeled through the notion of mapping as introduced in the previous section. It is exactly this correspondence that will determine how the queries posed to the system are answered.

In this section we discuss mappings which can be expressed in terms of first order logic assertions. Mappings going beyond first order logic are briefly discussed in Section 6.

Two basic approaches for specifying the mapping in a data integration system have been proposed in the literature, called *local-as-view* (LAV), and *global-as-view* (GAV), respectively [89, 60]. We discuss these approaches separately. We then end the section with a comparison of the two kinds of mapping.

3.1 Local as view

In a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ based on the LAV approach, the mapping $\mathcal{M}$ associates to each element s of the source schema $\mathcal{S}$ a query $q_{\mathcal{G}}$ over $\mathcal{G}$. In other words, the query language $\mathcal{L}_{\mathcal{M},\mathcal{S}}$ allows only expressions constituted by one symbol of the alphabet $\mathcal{A}_{\mathcal{S}}$. Therefore, a LAV mapping is a set of assertions, one for each element s of $\mathcal{S}$, of the form

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

From the modeling point of view, the LAV approach is based on the idea that the content of each source s should be characterized in terms of a view $q_{\mathcal{G}}$ over the global schema. A notable case of this type is when the data integration system is based on an enterprise model, or an ontology [58]. This idea is effective whenever the data integration system is based on a global schema that is stable and well-established in the organization. Note that the LAV approach favors the extensibility of the system: adding a new source simply means enriching the mapping with a new assertion, without other changes.

To better characterize each source with respect to the global schema, several authors have proposed more sophisticated assertions in the LAV mapping, in particular with the goal of establishing the assumption holding for the various source extensions [1, 53, 65, 24]. Formally, this means that in the LAV mapping, a new specification, denoted $as(s)$, is associated to each source element s. The specification $as(s)$ determines how accurate is the knowledge on the data satisfying the sources, i.e., how accurate is the source with respect to the associated view $q_{\mathcal{G}}$. Three possibilities have been considered[1]:

- *Sound views.* When a source s is *sound* (denoted with $as(s) = sound$), its extension provides any subset of the tuples satisfying the corresponding view $q_{\mathcal{G}}$. In other words, given a source database $\mathcal{D}$, from the fact that a tuple is in $s^{\mathcal{D}}$ one can conclude that it satisfies the associated view over the global schema, while from the fact that a tuple is not in $s^{\mathcal{D}}$ one cannot conclude that it does not satisfy the corresponding view. Formally, when $as(s) = sound$, a database $\mathcal{B}$ *satisfies the assertion* $s \rightsquigarrow q_{\mathcal{G}}$ with respect to $\mathcal{D}$ if
$$s^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}}$$
Note that, from a logical point of view, a sound source s with arity n is modeled through the first order assertion
$$\forall \mathbf{x}\ s(\mathbf{x}) \rightarrow q_{\mathcal{G}}(\mathbf{x})$$
where $\mathbf{x}$ denotes variables $x_1, \ldots, x_n$.
- *Complete views.* When a source s is *complete* (denoted with $as(s) = complete$), its extension provides any superset of the tuples satisfying the corresponding view. In other words, from the fact that a tuple is in $s^{\mathcal{D}}$ one cannot conclude that such a tuple satisfies the corresponding view. On the other hand, from the fact that a tuple is not in $s^{\mathcal{D}}$ one can conclude that such a tuple does not satisfy the view. Formally, when $as(s) = complete$, a database $\mathcal{B}$ *satisfies the assertion* $s \rightsquigarrow q_{\mathcal{G}}$ with respect to $\mathcal{D}$ if
$$s^{\mathcal{D}} \supseteq q_{\mathcal{G}}^{\mathcal{B}}$$
From a logical point of view, a complete source s with arity n is modeled through the first order assertion
$$\forall \mathbf{x}\ q_{\mathcal{G}}(\mathbf{x}) \rightarrow s(\mathbf{x})$$
- *Exact Views.* When a source s is *exact* (denoted with $as(s) = exact$), its extension is exactly the set of tuples of objects satisfying the corresponding view. Formally, when $as(s) = exact$, a database $\mathcal{B}$ *satisfies the assertion* $s \rightsquigarrow q_{\mathcal{G}}$ with respect to $\mathcal{D}$ if
$$s^{\mathcal{D}} = q_{\mathcal{G}}^{\mathcal{B}}$$
From a logical point of view, an exact source s with arity n is modeled through the first order assertion
$$\forall \mathbf{x}\ s(\mathbf{x}) \leftrightarrow q_{\mathcal{G}}(\mathbf{x})$$

Typically, in the literature, when the specification of $as(s)$ is missing, source s is considered sound. This is also the assumption we make in this paper.

Information Manifold [62], and the system presented in [78] are examples of LAV systems. Information Manifold expresses the global schema in terms of a Description Logic [8], and adopts the language of conjunctive queries as query languages $\mathcal{L}_{\mathcal{Q}}$, and $\mathcal{L}_{\mathcal{M},\mathcal{G}}$. The system described in [78] uses an XML global schema, and adopts XML-based query languages for both user queries and queries in the mapping. More powerful schema languages for expressing the global schema are reported in [42, 59, 22, 21]. In particular, [42, 59] discusses the case where various forms of relational integrity constraints are expressible in the global schema, including functional and inclusion dependencies, whereas [22, 21] consider a setting where the global schema is expressed in terms of Description Logics [11], which allow for the specification of various types of constraints.

[1] In some papers, for example [24], different assumptions on the domain of the database (open vs. closed) are also taken into account.

3.2 Global as view

In the GAV approach, the mapping $\mathcal{M}$ associates to each element g in $\mathcal{G}$ a query $q_{\mathcal{S}}$ over $\mathcal{S}$. In other words, the query language $\mathcal{L}_{\mathcal{M},\mathcal{G}}$ allows only expressions constituted by one symbol of the alphabet $\mathcal{A}_{\mathcal{G}}$. Therefore, a GAV mapping is a set of assertions, one for each element g of $\mathcal{G}$, of the form

$$g \rightsquigarrow q_{\mathcal{S}}$$

From the modeling point of view, the GAV approach is based on the idea that the content of each element g of the global schema should be characterized in terms of a view $q_{\mathcal{S}}$ over the sources. In some sense, the mapping explicitly tells the system how to retrieve the data when one wants to evaluate the various elements of the global schema. This idea is effective whenever the data integration system is based on a set of sources that is stable. Note that, in principle, the GAV approach favors the system in carrying out query processing, because it tells the system how to use the sources to retrieve data. However, extending the system with a new source is now a problem: the new source may indeed have an impact on the definition of various elements of the global schema, whose associated views need to be redefined.

To better characterize each element of the global schema with respect to the sources, more sophisticated assertions in the GAV mapping can be used, in the same spirit as we saw for LAV. Formally, this means that in the GAV mapping, a new specification, denoted $as(g)$ (either *sound*, *complete*, or *exact*) is associated to each element g of the global schema. When $as(g) = sound$ (resp., $complete, exact$), a database $\mathcal{B}$ *satisfies the assertion* $g \rightsquigarrow q_{\mathcal{S}}$ with respect to a source database $\mathcal{D}$ if

$$q_{\mathcal{S}}^{\mathcal{D}} \subseteq g^{\mathcal{B}} \qquad (\text{resp., } q_{\mathcal{S}}^{\mathcal{D}} \supseteq g^{\mathcal{B}},\ q_{\mathcal{S}}^{\mathcal{D}} = g^{\mathcal{B}})$$

The logical characterization of sound views and complete views in GAV is therefore through the first order assertions

$$\forall \mathbf{x}\ q_{\mathcal{S}}(\mathbf{x}) \rightarrow g(\mathbf{x}), \qquad \forall \mathbf{x}\ g(\mathbf{x}) \rightarrow q_{\mathcal{S}}(\mathbf{x})$$

respectively.

It is interesting to observe that the implicit assumption in many GAV proposals is the one of exact views. Indeed, in a setting where all the views are exact, there are no constraints in the global schema, and a first order query language is used as $\mathcal{L}_{\mathcal{M},\mathcal{S}}$, a GAV data integration system enjoys what we can call the "single database property", i.e., it is characterized by a single database, namely the global database that is obtained by associating to each element the set of tuples computed by the corresponding view over the sources. This motivates the widely shared intuition that query processing in GAV is easier than in LAV. However, it should be pointed out that the single database property only holds in such a restricted setting.

In particular, the possibility of specifying constraints in $\mathcal{G}$ greatly enhances the modeling power of GAV systems, especially in those situations where the global schema is intended to be expressed in terms of a conceptual data model, or in terms of an ontology [16]. In these cases, the language $\mathcal{L}_{\mathcal{G}}$ is in fact sufficiently powerful to allow for specifying, either implicitly or explicitly, various forms of integrity constraints on the global database.

Most of current data integration systems follow the GAV approach. Notable examples are TSIMMIS [51], Garlic [30], COIN [52], MOMIS [10], Squirrel [92], and IBIS [17]. Analogously to the case of LAV systems, these systems usually adopt simple languages for expressing both the global and the source schemas. IBIS is the only system we are aware of that takes into account integrity constraints in the global schema.

3.3 Comparison between GAV and LAV

The LAV and the GAV approaches are compared in [89] from the point of view of query processing. Generally speaking, it is well known that processing queries in the LAV approach is a difficult task. Indeed, in this approach the only knowledge we have about the data in the global schema is through the views representing the sources, and such views provide only partial information about the data. Since the mapping associates to each source a view over the global schema, it is not immediate to infer how to use the sources in order to answer queries expressed over the global schema. On the other hand, query processing looks easier in the GAV approach, where we can take advantage that the mapping directly specifies which source queries corresponds to the elements of the global schema. Indeed, in most GAV systems, query answering is based on a simple unfolding strategy.

From the point of view of modeling the data integration system, the GAV approach provides a specification mechanism that has a more procedural flavor with respect to the LAV approach. Indeed, while in LAV the designer may concentrate on declaratively specifying the content of the source in terms of the global schema, in GAV, one is forced to specify how to get the data of the global schema by means of queries over the sources. A throughout analysis of the differences/similarities of the two approaches from the point of view of modeling is still missing. A first attempt is reported in [19, 18], where the authors address the problem of checking whether a LAV system can be transformed into a GAV one, and vice-versa. They deal with transformations that are equivalent with respect to query answering, i.e., that enjoy the property that queries posed to the original system have the same answers when posed to the target system. Results on query reducibility from LAV to GAV systems may be useful, for example, to derive a procedural specification from a declarative one. Conversely, results on query reducibility from GAV to LAV may be useful to derive a declarative characterization of the content of the sources starting from a procedural specification. We briefly discuss the notions of query-preserving transformation, and of query-reducibility between classes of data integration systems.

Given two integration systems $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and $\mathcal{I}' = \langle \mathcal{G}', \mathcal{S}, \mathcal{M}' \rangle$ over the same source schema $\mathcal{S}$ and such that all elements of $\mathcal{G}$ are also elements of $\mathcal{G}'$, $\mathcal{I}'$ is said to be *query-preserving* with respect to $\mathcal{I}$, if for every query q to $\mathcal{I}$ and for every source database $\mathcal{D}$, we have that

$$q^{\mathcal{I},\mathcal{D}} = q^{\mathcal{I}',\mathcal{D}}$$

In other words, $\mathcal{I}'$ is query-preserving with respect to $\mathcal{I}$ if, for each query over the global schema of $\mathcal{I}$ and each source database, the certain answers to the query with respect to the source database that we get from the two integration

systems are identical. A class $\mathcal{C}_1$ of integration systems is *query-reducible* to a class $\mathcal{C}_2$ of integration systems if there exist a function $f : \mathcal{C}_1 \rightarrow \mathcal{C}_2$ such that, for each $\mathcal{I}_1 \in \mathcal{C}_1$ we have that $f(\mathcal{I}_1)$ is query-preserving with respect to $\mathcal{I}_1$.

With the two notions in place, the question of query reducibility between LAV and GAV is studied in [18] within a setting where views are considered sound, the global schema is expressed in the relational model, and the queries used in the integration systems (both the queries on the global schema, and the queries in the mapping) are expressed in the language of conjunctive queries. The results show that in such a setting none of the two transformations is possible. On the contrary, if one extends the framework, allowing for integrity constraints in the global schema, then reducibility holds in both directions. In particular, inclusion dependencies and a simple form of equality-generating dependencies suffice for a query-preserving transformation from a LAV system into a GAV one, whereas single head full dependencies are sufficient for the other direction. Both transformations result in a query-preserving system whose size is linearly related to the size of the original one.

Although in this paper we mainly refer to the LAV and GAV approaches to data integration, it is worth noticing that more general types of mappings have been also discussed in the literature. For example, [49] introduces the so-called GLAV approach. In GLAV, the relationships between the global schema and the sources are established by making use of both LAV and GAV assertions. More precisely, in a GLAV mapping as introduced in [49], every assertion has the form $q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{G}}$, where $q_{\mathcal{S}}$ is a conjunctive query over the source schema, and $q_{\mathcal{G}}$ is a conjunctive query over the global schema. A database $\mathcal{B}$ satisfies the assertion $q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{G}}$ with respect to a source database $\mathcal{D}$ if $q_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}}$. Thus, the GLAV approach models a situation where sources are sound. Interestingly, the technique presented in [19, 18] can be extended for transforming any GLAV system into a GAV one. The key idea is that a GLAV assertion can be transformed into a GAV assertion plus an inclusion dependency. Indeed, for each assertion

$$q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{G}}$$

in the GLAV system (where the arity of both queries is n), we introduce a new relation symbol r of arity n in the global schema of the resulting GAV system, and we associate to r the sound view $q_{\mathcal{S}}$ by means of

$$r \rightsquigarrow q_{\mathcal{S}}$$

plus the inclusion dependency

$$r \subseteq q_{\mathcal{G}}.$$

Now, it is immediate to verify that the above inclusion dependency can be treated exactly with the same technique introduced in the LAV to GAV transformation, and therefore, from the GLAV system we can obtain a query-preserving GAV system whose size is linearly related to the size of the original system.

4. QUERY PROCESSING IN LAV

In this section we discuss query processing in the LAV approach. From the definition given in Section 3, it is easy to see that answering queries in LAV systems is essentially an extended form of reasoning in the presence of incomplete information [91]. Indeed, when we answer a query over the global schema on the basis of a LAV mapping, we know only the extensions of the views associated to the sources, and this provides us with only partial information on the global database. As we already observed, in general, there are several possible global databases that are legal for the data integration system with respect to a given source database. This observation holds even for a setting where only sound views are allowed in the mapping. The problem is even more complicated when sources can be modeled as complete or exact views. In particular, dealing with exact sources essentially means applying the closed world assumption on the corresponding views [1, 85].

The following example rephrases an example given in [1]. Consider a data integration system $\mathcal{I}$ with global relational schema $\mathcal{G}$ containing (among other relations) a binary relation couple, and two constants Ann and Bill. Consider also two sources female and male, respectively with associated views

$$\mathsf{female}(f) \rightsquigarrow \{\ f, m \mid \mathsf{couple}(f, m)\ \}$$
$$\mathsf{male}(m) \rightsquigarrow \{\ f, m \mid \mathsf{couple}(f, m)\ \}$$

and consider a source database $\mathcal{D}$ with $\mathsf{female}^{\mathcal{D}} = \{\mathsf{Ann}\}$ and $\mathsf{male}^{\mathcal{D}} = \{\mathsf{Bill}\}$, and assume that there are no constraints imposed by a schema. If both sources are sound, we only know that some couple has Ann as its female component and Bill as its male component. Therefore, the query

$$Q \ = \ \{\ x, y \mid \mathsf{couple}(x, y)\ \}$$

asking for all couples would return an empty answer, i.e., $Q_c^{\mathcal{I},\mathcal{D}} = \emptyset$. However, if both sources are exact, we can conclude that all couples have Ann as their female component and Bill as their male component, and hence that $(\mathsf{Ann}, \mathsf{Bill})$ is the only couple, i.e., $Q_c^{\mathcal{I},\mathcal{D}} = \{(\mathsf{Ann}, \mathsf{Bill})\}$.

Since in LAV, sources are modeled as views over the global schema, the problem of processing a query is traditionally called *view-based query processing*. Generally speaking, the problem is to compute the answer to a query based on a set of views, rather than on the raw data in the database [89, 60].

There are two approaches to view-based query processing, called *view-based query rewriting* and *view-based query answering*, respectively. In the former approach, we are given a query q and a set of view definitions, and the goal is to reformulate the query into an expression of a fixed language $\mathcal{L}_R$ that refers only to the views and provides the answer to q. The crucial point is that the language in which we want the rewriting is fixed, and in general coincides with the language used for expressing the original query. In a LAV data integration setting, query rewriting aims at reformulating, in a way that is independent from the current source database, the original query in terms of a query to the sources. Obviously, it may happen that no rewriting in the target language $\mathcal{L}_R$ exists that is equivalent to the original query. In this case, we are interested in computing a so-called *maximally contained rewriting*, i.e., an expression that captures the original query in the best way.

Sound	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.*
Exact	CQ	CQ$^{\neq}$	PQ	Datalog	FOL
CQ	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
CQ$^{\neq}$	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
PQ	*coNP*	*coNP*	*coNP*	*coNP*	*undec.*
Datalog	*undec.*	*undec.*	*undec.*	*undec.*	*undec.*
FOL	*undec.*	*undec.*	*undec.*	*undec.*	*undec.*

Table 1: Complexity of view-based query answering

In view-based query answering, besides the query q and the view definitions, we are also given the extensions of the views. The goal is to compute the set of tuples t such that the knowledge on the view extensions logically implies that t is an answer to q, i.e., t is in the answer to q in all the databases that are consistent with the views. It is easy to see that, in a LAV data integration framework, this is exactly the problem of computing the certain answers to q with respect to a source database.

Notice the difference between the two approaches. In query rewriting, query processing is divided in two steps, where the first one re-expresses the query in terms of a given query language over the alphabet of the view names, and the second one evaluates the rewriting over the view extensions. In query answering, we do not pose any limitations on how queries are processed, and the only goal is to exploit all possible information, in particular the view extensions, to compute the answer to the query.

A large number of results have been reported for both approaches. We first focus on view-based query answering.

Query answering has been extensively investigated in the last years [1, 53, 43, 66, 4, 21]. A comprehensive framework for view-based query answering, as well as several interesting results, is presented in [53]. The framework considers various assumptions for interpreting the view extensions with respect to the corresponding definitions (closed, open, and exact view assumptions). In [1], an analysis of the complexity of the problem under the different assumptions is carried out for the case where the views and the queries are expressed in terms of various languages (conjunctive queries without and with inequalitites, positive queries, Datalog, and first-order queries). The complexity is measured with respect to the size of the view extensions (data complexity). Table 1 summarizes the results presented in [1]. Note that, for the query languages considered in that paper, the exact view assumption complicates the problem. For example, the data complexity of query answering for the case of conjunctive queries is PTIME under the sound view assumption, and coNP-complete for exact views. This can be explained by noticing that the exact view assumption introduces a form of negation, and therefore it may force to reason by cases on the objects stored in the views.

In [24], the problem is studied for a setting where the global schema models a semistructured database, i.e., a labeled directed graphs. It follows that both the user queries, and the queries used in the LAV mapping should be expressed in a query language for semistructured data. The main difficulty arising in this context is that languages for querying semistructured data enable expressing regular-path queries [2, 15, 45]. A regular-path query asks for all pairs of nodes in the database connected by a path conforming to a regular expression, and therefore may contain a restricted form of recursion. Note that, when the query contains unrestricted recursion, both view-based query rewriting and view-based query answering become undecidable, even when the views are not recursive [43].

Table 2 summarizes the results presented in [24]. Both data complexity, and expression complexity (complexity with respect to the size of the query and the view definitions) are taken into account. All upper bound results have been obtained by automata-theoretic techniques. In the analysis, a further distinction is proposed for characterizing the domain of the database (open vs. closed domain assumption). In the closed domain assumption we assume that the global database contains only objects stored in the sources. The results show that none of the cases can be solved in polynomial time (unless P = NP). This can be explained by observing that the need for considering various forms of incompleteness expressible in the query language (due to union and transitive closure), is a source of complexity for query answering. Obviously, under closed domain, our knowledge is more accurate than in the case of the open domain assumption, and this rules out the need for some combinatorial reasoning. This provides the intuition of why under closed domain the problem is "only" coNP-complete in all cases, for data, expression, and combined complexity. On the other hand, under open domain, we cannot exclude the possibility that the database contains more objects than those known to be in the views. For combined complexity, this means that we are forced to reason about the definition of the query and the views. Indeed, the problem cannot be less complex than comparing two regular path queries, and this explains the PSPACE lower bound. Interestingly, the table shows that the problem does not exceed the PSPACE complexity. Moreover, the data complexity remains in coNP, and therefore, although we are using a query language that is powerful enough to express a (limited) form of recursion, the problem is no more complex than in the case of disjunctions of conjunctive queries [1].

While regular-path queries represent the core of any query language for semistructured data, their expressive power is limited. Several authors point out that extensions are required for making them useful in real settings (see for example [14, 15, 80]). Indeed, the results in [24] have been extended to query language with the inverse operator [26], and to the class of union of conjunctive regular-path queries in [28].

Turning our attention to view-based query rewriting, several recent papers investigate the rewriting question for different classes of queries. The problem is investigated for the case of conjunctive queries (with or without arithmetic comparisons) in [66, 84], for disjunctive views in [4], for queries with aggregates in [87, 37, 56], for recursive queries and nonrecursive views in [43], for queries expressed in Description Logics in [9], for regular-path queries and their extensions

domain	views	Complexity		
		data	expression	combined
closed	all sound	*coNP*	*coNP*	*coNP*
	all exact	*coNP*	*coNP*	*coNP*
	arbitrary	*coNP*	*coNP*	*coNP*
open	all sound	*coNP*	*PSPACE*	*PSPACE*
	all exact	*coNP*	*PSPACE*	*PSPACE*
	arbitrary	*coNP*	*PSPACE*	*PSPACE*

Table 2: Complexity of view-based query answering for regular-path queries

in [23, 26, 27], and in the presence of integrity constraints in [59, 44]. Rewriting techniques for query optimization are described, for example, in [34, 3, 88], and in [46, 80, 82] for the case of path queries in semistructured data.

We already noted that view-based query rewriting and view-based query answering are different problems. Unfortunately, their similarity sometimes gives raise to a sort of confusion between the two notions. Part of the problem comes from the fact that when the query and the views are conjunctive queries, the best possible rewriting is expressible as union of conjunctive queries, which is basically the same language as the one of the original query and views. However, for other query languages this is not the case. Abstracting from the language used to express the rewriting, we can define a rewriting of a query with respect to a set of views as a function that, given the extensions of the views, returns a set of tuples that is contained in the answer set of the query in every database consistent with the views. We call the rewriting that returns precisely such set the *perfect rewriting* of the query with respect to the views. Observe that, by evaluating the perfect rewriting over given view extensions, one obtains the same set of tuples provided by view-based query answering. i.e., in data integration terminology, the set of certain answers to the query with respect to the view extension. Hence, the perfect rewriting is the best rewriting one can obtain, given the available information on both the definitions and the extensions of the views.

An immediate consequence of the relationship between perfect rewriting and query answering is that the data complexity of evaluating the perfect rewriting over the view extensions is the same as the data complexity of answering queries using views. Typically, one is interested in queries that can be evaluated in PTIME (i.e., are PTIME functions in data complexity), and hence we would like rewritings to be PTIME as well. For queries and views that are conjunctive queries (without union), the perfect rewriting is a union of conjunctive queries and hence is PTIME [1]. However, already for very simple query languages containing union the perfect rewriting is not PTIME in general. Hence, for such languages it would be interesting to characterize which instances of query rewriting admit a perfect rewriting that is PTIME. By establishing a tight connection between view-based query answering and constraint-satisfaction problems, it is argued in [27] that this is a difficult task.

5. QUERY PROCESSING IN GAV

Most GAV data integration systems do not allow integrity constraints in the global schema, and assume that views

$s_1^{\mathcal{D}}$:

12	*calvin*	*rome*	21
15	*alice*	*hong kong*	24

$s_2^{\mathcal{D}}$:

AF	*hotdog corp.*
BN	*banana ltd.*

$s_3^{\mathcal{D}}$:

12	*AF*
16	*BN*

Figure 1: Extension of sources for the example

are exact. It is easy to see that, under these assumptions, query processing can be based on a simple unfolding strategy. When we have a query q over the alphabet $\mathcal{A}_\mathcal{G}$ of the global schema, every element of $\mathcal{A}_\mathcal{G}$ is substituted with the corresponding query over the sources, and the resulting query is then evaluated at the sources. As we said before, such a strategy suffices mainly because the data integration system enjoys the single database property. Notably, the same strategy applies also in the case of sound views.

However, when the language $\mathcal{L}_\mathcal{G}$ used for expressing the global schema allows for integrity constraints, and the views are sound, then query processing in GAV systems becomes more complex. Indeed, in this case, integrity constraints can in principle be used in order to overcome incompleteness of data at the sources. The following example shows that, by taking into account foreign key constraints, one can obtain answers that would be missed by simply unfolding the user query.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, where $\mathcal{G}$ is constituted by the relations

$$\begin{array}{l} \mathsf{employee}(\mathit{Ecode}, \mathit{Ename}, \mathit{Ecity}) \\ \mathsf{company}(\mathit{Ccode}, \mathit{Cname}) \\ \mathsf{employed}(\mathit{Ecode}, \mathit{Ccode}) \end{array}$$

and the constraints

$$\begin{array}{rcl} \mathit{key}(\mathsf{employee}) & = & \{\mathit{Ecode}\} \\ \mathit{key}(\mathsf{company}) & = & \{\mathit{Ccode}\} \\ \mathsf{employed}[\mathit{Ecode}] & \subseteq & \mathsf{employee}[\mathit{Ecode}] \\ \mathsf{employed}[\mathit{Ccode}] & \subseteq & \mathsf{company}[\mathit{Ccode}] \end{array}$$

The source schema $\mathcal{S}$ consists of three sources. Source s_1, of arity 4, contains information about employees with their code, name, city, and date of birth. Source s_2, of arity 2, contains codes and names of companies. Finally, Source s_3, of arity 2, contains information about employment in companies. The mapping $\mathcal{M}$ is defined by

$$\begin{array}{rl} \mathsf{employee} \rightsquigarrow & \{\ x, y, z \mid \mathsf{s}_1(x, y, z, w)\ \} \\ \mathsf{company} \rightsquigarrow: & \{\ x, y \mid \mathsf{s}_2(x, y)\ \} \\ \mathsf{employed} \rightsquigarrow: & \{\ x, w \mid \mathsf{s}_3(x, w)\ \} \end{array}$$

Now consider the following user query q, asking for codes of employees:

$$\{\ x \mid \mathsf{employee}(x, y, z)\ \}$$

Suppose that the data stored in the source database $\mathcal{D}$ are those depicted in Figure 1: by simply unfolding q we obtain the answer $\{12\}$. However, due to the integrity constraint $\mathsf{employed}[\mathit{Ecode}] \subseteq \mathsf{employee}[\mathit{Ecode}]$, we know that 16 is the code of a person, even if it does not appear in $s_1^{\mathcal{D}}$. The correct answer to q is therefore $\{12, 16\}$. Observe that we

do not know any value for the attributes of the employee whose *Ecode* is 16.

Given a source database $\mathcal{D}$, let us call "retrieved global database" the global database that is obtained by populating each relation r in the global schema according to the mapping, i.e., by populating r with the tuples obtained by evaluating the query that the mapping associates to q. In general, integrity constraints may be violated in the retrieved global database (e.g., the retrieved global database for the above example). Regarding key constraints, let us assume that the query that the mapping associates to each global schema relation r is such that the data retrieved for r do not violate the key constraint of r. In other words, the management of key constraints is left to the designer (see next section for a discussion on this subject). On the other hand, the management of foreign key constraints cannot be left to the designer, since it is strongly related to the incompleteness of the sources. Moreover, since foreign keys are interrelation constraints, they cannot be dealt with in the GAV mapping, which, by definition, works on each global relation in isolation.

The assumption of sound views asserts that the tuples retrieved for a relation r are a subset of the tuples that the system assigns to r; therefore, we may think of completing the retrieved global database by suitably adding tuples in order to satisfy foreign key constraints, while still conforming to the mapping. When a foreign key constraint is violated, there are several ways of adding tuples to the retrieved global database to satisfy such a constraint. In other words, in the presence of foreign key constraints in the global schema, the semantics of a data integration system must be formulated in terms of a *set* of databases, instead of a single one. Since we are interested in the certain answers $q^{\mathcal{I},\mathcal{D}}$ to a query q, i.e., the tuples that satisfy q in all global databases that are legal for $\mathcal{I}$ with respect to $\mathcal{D}$, the existence of several such databases complicates the task of query answering.

In [17], a system called IBIS is presented, that takes into account key and foreign key constraints over the global relational schema. The system uses the foreign key constraints in order to retrieve data that could not be obtained in traditional data integration systems. The language for expressing both the user query and the queries in the mapping is the one of union of conjunctive queries. To process a query q, IBIS expands q by taking into account the foreign key constraints on the global relations appearing in the atoms. Such an expansion is performed by viewing each foreign key constraint $r_1[\mathbf{X}] \subseteq r_2[\mathbf{Y}]$, where $\mathbf{X}$ and $\mathbf{Y}$ are sets of h attributes and $\mathbf{Y}$ is a key for r_2, as a logic programming [77] rule

$$r_2'(\vec{X}, f_{h+1}(\vec{X}), \ldots, f_n(\vec{X})) \leftarrow r_1'(\vec{X}, X_{h+1}, \ldots, X_m)$$

where each f_i is a Skolem function, $\vec{X}$ is a vector of h variables, and we have assumed for simplicity that the attributes involved in the foreign key are the first h ones. Each r_i' is a predicate, corresponding to the global relation r_i, defined by the above rules for foreign key constraints, together with the rule

$$r_i'(X_1, \ldots, X_n) \leftarrow r_i(X_1, \ldots, X_n)$$

Once such a logic program $\Pi_{\mathcal{G}}$ has been defined, it can be used to generate the expanded query *expand_q* associated to the original query q. This is done by performing a partial evaluation [40] with respect to $\Pi_{\mathcal{G}}$ of the body of q', which is the query obtained by substituting in q each predicate r_i with r_i'. In the partial evaluation tree, a node is not expanded anymore either when no atom in the node unifies with a head of a rule, or when the node is subsumed by (i.e., is more specific than) one of its predecessors. In the latter case, the node gets an empty node as a child; intuitively this is because such a node cannot provide any answer that is not already provided by its more general predecessor. These conditions guarantee that the construction of the partial evaluation tree for a query always terminates. Then, the expansion *expand_q* of q is a union of conjunctive queries whose body is constituted by the disjunction of all nonempty leaves of the partial evaluation tree. It is possible to show that, by unfolding *expand_q* according to the mapping, and evaluating the resulting query over the sources, one obtains exactly the set of certain answers of q to $\mathcal{I}$ with respect to $\mathcal{D}$ [17].

6. INCONSISTENCIES BETWEEN SOURCES

The formalization of data integration presented in the previous sections is based on a first order logic interpretation of the assertions in the mapping, and, therefore, is not able to cope with inconsistencies between sources. Indeed, if in a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, the data retrieved from the sources do not satisfy the integrity constraints of $\mathcal{G}$, then no global database exists for $\mathcal{I}$, and query answering becomes meaningless. This is the situation occurring when data in the sources are mutually inconsistent. In practice, this situation is generally dealt with by means of suitable transformation and cleaning procedures to be applied to data retrieved by the sources (see [12, 50]). In this section, we address the problem from a more theoretical perspective.

Several recent papers aim at formally dealing with inconsistencies in databases, in particular for providing informative answers even in the case of a database that does not satisfy its integrity constraints (see, for example, [13, 6, 7, 54]). Although interesting, such results are not specifically tailored to the case of different consistent data sources that are mutually inconsistent, that is the case of interest in data integration. This case is addressed in [76], where the authors propose an operator for *merging databases* under constraints. Such operator allows one to obtain maximal amount of information from each database by means of a majority criterion used in case of conflict. However, also the approach described in [76] does not take explicitly into account the notion of mapping as introduced in our data integration setting.

In data integration, according to the definition of mapping satisfaction as given in Section 3, it may be the case that the data retrieved from the sources cannot be reconciled in the global schema in such a way that both the constraints of the global schema, and the mapping are satisfied. For example, this happens when a key constraint specified for the relation r in the global schema is violated by the tuples retrieved by the view associated to r, since the assumption of sound views does not allow us to disregard tuples from

r with duplicate keys. If we do not want to conclude in this case that no global database exists that is legal for $\mathcal{I}$ with respect to $\mathcal{D}$, we need a different characterization of the mapping. In particular, we need a characterization that allows us support query processing even when the data at the sources are incoherent with respect to the integrity constraints on the global schema.

A possible solution is to characterize the data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ (with $\mathcal{M} = \{r_1 \rightsquigarrow V_1, \ \ldots \ , r_n \rightsquigarrow V_n\}$), in terms of those global databases that

1. satisfy the integrity constraints of $\mathcal{G}$, and
2. approximate at best the satisfaction of the assertions in the mapping $\mathcal{M}$, i.e., that are *as sound as possible*.

In other, the integrity constraints of $\mathcal{G}$ are considered strong, whereas the mapping is considered soft. Given a source database $\mathcal{D}$ for $\mathcal{I}$, we can now define an ordering between the global databases for $\mathcal{I}$ as follows. If $\mathcal{B}_1$ and $\mathcal{B}_2$ are two databases that are legal with respect to $\mathcal{G}$, we say that $\mathcal{B}_1$ is *better* than $\mathcal{B}_2$ with respect to $\mathcal{D}$, denoted as $\mathcal{B}_1 \gg_{\mathcal{D}} \mathcal{B}_2$, if there exists an assertion $r_i \rightsquigarrow V_i$ in $\mathcal{M}$ such that

- $(r_i^{\mathcal{B}_1} \cap V_i^{\mathcal{D}}) \supset (r_i^{\mathcal{B}_2} \cap V_i^{\mathcal{D}})$, and
- $(r_j^{\mathcal{B}_1} \cap V_j^{\mathcal{D}}) \supseteq (r_j^{\mathcal{B}_2} \cap V_j^{\mathcal{D}})$, for all $r_j \rightsquigarrow V_j$ in $\mathcal{M}$ with $j \neq i$;

Intuitively, this means that there is at least one assertion for which $\mathcal{B}_1$ satisfies the sound mapping better than $\mathcal{B}_2$, while for no other assertion $\mathcal{B}_2$ is better than $\mathcal{B}_1$. In other words, $\mathcal{B}_1$ approximates the sound mapping better than $\mathcal{B}_2$.

It is easy to verify that the relation $\gg_{\mathcal{D}}$ is a partial order. With this notion in place, we can now define the notion of $\mathcal{B}$ satisfying the mapping $\mathcal{M}$ with respect to $\mathcal{D}$ in our setting: a database $\mathcal{B}$ that is legal with respect to $\mathcal{G}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{D}$ if $\mathcal{B}$ is maximal with respect to $\gg_{\mathcal{D}}$, i.e., for no other global database $\mathcal{B}'$ that is legal with respect to $\mathcal{G}$, we have that $\mathcal{B}' \gg_{\mathcal{D}} \mathcal{B}$.

The notion of legal database for $\mathcal{I}$ with respect to $\mathcal{D}$, and the notion of certain answer remain the same, given the new definition of satisfaction of mapping. It is immediate to verify that, if there exists a legal database for $\mathcal{I}$ with respect to $\mathcal{D}$ under the first order logic interpretation of the mapping, then the new semantics and the old one coincide, in the sense that, for each query q, the set $q^{\mathcal{I},\mathcal{D}}$ of certain answers computed under the first order semantics coincides with the set of certain answers computed under the new semantics presented here.

The problem of inconsistent sources in data integration is addressed in [64], in particular for the case where:

- the global schema is a relational schema with key and foreign key constraints,
- the mapping is of type GAV,
- the query language $\mathcal{L}_{\mathcal{M},\mathcal{S}}$ is the language of union of conjunctive queries,
- the views in the mapping are intended to be sound.

In such a setting, an algorithm is proposed for computing the certain answers of a query in the new semantical framework presented above. The algorithm checks whether a given tuple t is a certain answer to a query q with respect to a given source database $\mathcal{D}$ in coNP data complexity (i.e., with respect to the size of $\mathcal{D}$). Based on this result, the problem of computing the certain answers in the presented framework can be shown to be coNP-complete in data complexity.

7. REASONING ON QUERIES

Recent work addresses the problem of reasoning on queries in data integration systems. The basic form of reasoning on queries is checking containment, i.e., verifying whether one query returns a subset of the result computed by the other query in all databases. Most of the results on query containment concern conjunctive queries and their extensions. In [33], NP-completeness has been established for conjunctive queries, in [63, 90], Π_2^p-completeness of containment of conjunctive queries with inequalities is proved, and in [86] the case of queries with the union and difference operators is studied. For various classes of Datalog queries with inequalities, decidability and undecidability results are presented in [35] and [90], respectively. Other papers consider the case of query containment in the presence of various types of constraints [5, 39, 32, 69, 71, 70, 20], and for regular-path queries and their extensions [47, 25, 28, 41].

Besides the usual notion of containment, several other notions have been introduced related to the idea of comparing queries in a data integration setting, especially in the context of the LAV approach.

In [79], a query is said to be contained in another query *relative to a set of sources* modeled as views, if, for each extension of the views, the certain answers to the former query are a subset of the certain answers to the latter. Note that this reasoning problem is different from the usual containment checking: here we are comparing the two queries with respect to the certain answers computable on the basis of the views available. The difference becomes evident if one considers a counterexample to relative containment: Q_1 is not contained in Q_2 relative to views $\mathcal{V}$ if there is a tuple t and an extension $\mathcal{E}$ of $\mathcal{V}$, such that for each database $\mathcal{DB}$ consistent with $\mathcal{E}$ (i.e., a database $\mathcal{DB}$ such that, the result $\mathcal{V}^{\mathcal{DB}}$ of evaluating the views over $\mathcal{DB}$ is exactly $\mathcal{E}$), t is an answer of Q_1 to $\mathcal{DB}$, but there is a database $\mathcal{DB}'$ consistent with $\mathcal{E}$ such that t is not an answer of Q_2 to $\mathcal{DB}'$. In other words, Q_1 is not contained in Q_2 relative to views $\mathcal{V}$ if there are two databases $\mathcal{DB}$ and $\mathcal{DB}'$ such that $\mathcal{V}^{\mathcal{DB}} = \mathcal{V}^{\mathcal{DB}'}$ and $Q_1^{\mathcal{DB}} = Q_2^{\mathcal{DB}'}$.

In [79], it is shown that the problem of checking relative containment is Π_2^P complete in the case of conjunctive queries and views. In [74], such results are extended to the case where views have limited access patterns.

In [72], the authors introduce the notion of "p-containment"

(where "p" stands for power): a view set $\mathcal{V}$ is said to be p-contained in another view set $\mathcal{W}$, i.e., $\mathcal{W}$ has at least the answering power of $\mathcal{V}$, if $\mathcal{W}$ can answer all queries that can be answered using $\mathcal{V}$.

The notion of "information content" of materialized views is studied in [57] for a restricted class of aggregate queries, with the goal of devising techniques for checking whether a set of views is sufficient for completely answering a given query based on the views.

One of the ideas underlying the above mentioned papers is the one of losslessness: a set of views is *lossless* with respect to a query, if, no matter what the database is, we can answer the query by solely relying on the content of the views. This question is relevant for example in mobile computing, where we may be interested in checking whether a set of cached data allows us to derive the requested information without accessing the network, or in data warehouse design, in particular for the view selection problem [36], where we have to measure the quality of the choice of the views to materialize in the data warehouse. In data integration, losslessness may help in the design of the data integration system, in particular, by selecting a minimal subset of sources to access without losing query-answering power.

The definition of losslessness relies on that of certain answers: a set of views is *lossless* with respect to a query, if for every database, we can answer the query over that database by computing the certain answers based on the view extensions. It follows that there are at least two versions of losslessness, namely, losslessness under the sound view assumption, and losslessness under the exact view assumption.

The first version is obviously weaker than the second one. If views $\mathcal{V}$ are lossless with respect to a query Q under the sound view assumption, then we know that, from the intensional point of views, $\mathcal{V}$ contain enough information to completely answer Q, even though the possible incompleteness of the view extensions may prevent us form obtaining all the answers that Q would get from the database. On the other hand, if $\mathcal{V}$ are lossless with respect to a query Q under the exact view assumption, then we know that they contain enough information to completely answer Q, both from the intensional and from the extensional point of view.

In [29], the problem of losslessness is addressed in a context where both the query and the views are expressed as regular path queries. It is shown that, in the case of the sound view assumption, the problem is solvable by a technique that is based on searching for a counterexample to losslessness, i.e., two databases that are both coherent with the view extensions, and that differ in the answers to the query. Different from traditional query containment, the search for a counterexample is complicated by the presence of a quantification over all possible view extensions. The key observation in [29] is that, under the sound view assumption, we can restrict our attention to counterexamples that are linear databases, and this allows devising a method that uses, via automata-theoretic techniques, the known connection between view-based query answering and constraint satisfaction [27]. As far as the computational complexity is concerned, the problem is PSPACE-complete with respect to the view definitions, and EXPSPACE-complete with respect to the query.

It is interesting to observe that, for the case of exact views, the search for a counterexample cannot be restricted to linear databases. Actually, the question of losslessness under the exact view assumption is largely unexplored. To the best of our knowledge, the problem is open even for a setting where both the query and the views are conjunctive queries.

8. CONCLUSIONS

The aim of this tutorial was to provide an overview of some of the theoretical issues underlying data integration. Several interesting problems remain open in each of the topics that we have discussed. For example, more investigation is needed for a deep understanding of the relationship between the LAV and the GAV approaches. Open problems remain on algorithms and complexity for view-based query processing, in particular for the case of rich languages for semistructured data, for the case of exact views, and for the case of integrity constraints in the global schema. Query processing in GAV with constraints has been investigated only recently, and interesting classes of constraints have not been considered yet. The treatment of mutually inconsistent sources, and the issue of reasoning on queries present many open research questions.

Moreover, data integration is such a rich field that several important related aspects not addressed here can be identified, including the following.

- How to build an appropriate global schema, and how to discover inter-schema [31] and mapping assertions (LAV or GAV) in the design of a data integration system (see, for instance, [83]).
- How to (automatically) synthesize wrappers that present the data at the sources in a form [] that is suitable for their use in the mapping.
- How to deal with possible limitations in accessing the sources, both in LAV [84, 67, 68] and in GAV [75, 48, 73, 74].
- How to incorporate the notions of quality (data quality, quality of answers, etc.) [81], and data cleaning [12] into a formal framework for data integration.
- How to learn rules that allow for automatically mapping data items in different sources (for example, for inferring that two key values in different sources actually refer to the same real-world object [38]).
- How to go beyond the architecture based on a global schema, so as, for instance, to model data exchange, transformation, and cooperation rather than data integration (see, e.g., [55]), or to devise information integration facilities for the Semantic Web.
- How to optimize the evaluation of queries posed to a data integration system [3].

We believe that each of the above issues is characterized by interesting research problems still to investigate.

9. ACKNOWLEDGMENTS

I warmly thank Diego Calvanese, Giuseppe De Giacomo and Moshe Y. Vardi, with whom I carried out most of my research work on data integration during the last years. Also, I thank all colleagues that I have been working with in several data integration projects, in particular Andrea Calì, Domenico Lembo, Daniele Nardi, Riccardo Rosati, and all participants to the ESPRIT LTR project "DWQ (Data Warehouse Quality)", and the MIUR (Italian Ministry of University and Research) project "D2I (From Data To Information)".

Finally, this is the first paper I can dedicate to my son Domenico, and for that I want to thank his mother.

10. REFERENCES

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

[3] S. Adali, K. S. Candan, Y. Papakonstantinou, and V. S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 137–148, 1996.

[4] F. N. Afrati, M. Gergatsoulis, and T. Kavalieros. Answering queries using materialized views with disjunction. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 435–452. Springer, 1999.

[5] A. V. Aho, Y. Sagiv, and J. D. Ullman. Equivalence among relational expressions. *SIAM J. on Computing*, 8:218–246, 1979.

[6] M. Arenas, L. E. Bertossi, and J. Chomicki. Consistent query answers in inconsistent databases. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 68–79, 1999.

[7] M. Arenas, L. E. Bertossi, and J. Chomicki. Specifying and querying database repairs using logic programs with exceptions. In *Proc. of the 4th Int. Conf. on Flexible Query Answering Systems (FQAS'00)*, pages 27–41. Springer, 2000.

[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002. To appear.

[9] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 99–108, 1997.

[10] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: the MOMIS project demonstration. In *Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, 2000.

[11] A. Borgida. Description logics in data management. *IEEE Trans. on Knowledge and Data Engineering*, 7(5):671–682, 1995.

[12] M. Bouzeghoub and M. Lenzerini. Introduction to the special issue on data extraction, cleaning, and reconciliation. *Information Systems*, 26(8):535–536, 2001.

[13] F. Bry. Query answering in information systems with integrity constraints. In *IFIP WG 11.5 Working Conf. on Integrity and Control in Information System*. Chapman & Hall, 1997.

[14] P. Buneman. Semistructured data. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 117–121, 1997.

[15] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu. A query language and optimization technique for unstructured data. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 505–516, 1996.

[16] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 20th Int. Conf. on Conceptual Modeling (ER 2001)*, 2001.

[17] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, 2002. To appear.

[18] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini. On the expressive power of data integration systems. Submitted for pubblication, 2002.

[19] A. Calì, G. De Giacomo, and M. Lenzerini. Models of information integration: Turning local-as-view into global-as-view. In *Foundations of Models for Information Integration*. On line proceedings, `http://www.fmldo.org/FMII-2001`, 2001.

[20] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[21] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[22] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[23] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Rewriting of regular expressions and regular path queries. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 194–204, 1999.

[24] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Answering regular path queries using views. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 389–398, 2000.

[25] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Containment of conjunctive regular path queries with inverse. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 176–185, 2000.

[26] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*, pages 58–66, 2000.

[27] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 361–371, 2000.

[28] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query answering and query containment over semistructured data. In *Proc. of the 8th Int. Workshop on Database Programming Languages (DBPL 2001)*, 2001.

[29] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Lossless regular views. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 58–66, 2002.

[30] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proc. of the 5th Int. Workshop on Research Issues in Data Engineering – Distributed Object Management (RIDE-DOM'95)*, pages 124–131. IEEE Computer Society Press, 1995.

[31] T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *J. of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[32] E. P. F. Chan. Containment and minimization of positive conjunctive queries in oodb's. In *Proc. of the 11th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'92)*, pages 202–211, 1992.

[33] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *Proc. of the 9th ACM Symp. on Theory of Computing (STOC'77)*, pages 77–90, 1977.

[34] S. Chaudhuri, S. Krishnamurthy, S. Potarnianos, and K. Shim. Optimizing queries with materialized views. In *Proc. of the 11th IEEE Int. Conf. on Data Engineering (ICDE'95)*, Taipei (Taiwan), 1995.

[35] S. Chaudhuri and M. Y. Vardi. On the equivalence of recursive and nonrecursive Datalog programs. In *Proc. of the 11th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'92)*, pages 55–66, 1992.

[36] R. Chirkova, A. Y. Halevy, and D. Suciu. A formal perspective on the view selection problem. In *Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB 2001)*, pages 59–68, 2001.

[37] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting aggregate queries using views. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 155–166, 1999.

[38] W. W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 201–212, 1998.

[39] A. C. K. David S. Johnson. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. of Computer and System Sciences*, 28(1):167–189, 1984.

[40] G. De Giacomo. Intensional query answering by partial evaluation. *J. of Intelligent Information Systems*, 7(3):205–233, 1996.

[41] A. Deutsch and V. Tannen. Optimization properties for classes of conjunctive regular path queries. In *Proc. of the 8th Int. Workshop on Database Programming Languages (DBPL 2001)*, 2001.

[42] O. Duschka. *Query Planning and Optimization in Information Integration*. PhD thesis, Stanford University, 1997.

[43] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 109–116, 1997.

[44] O. M. Duschka and A. Y. Levy. Recursive plans for information gathering. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI'97)*, pages 778–784, 1997.

[45] M. F. Fernandez, D. Florescu, J. Kang, A. Y. Levy, and D. Suciu. Catching the boat with strudel: Experiences with a web-site management system. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 414–425, 1998.

[46] M. F. Fernandez and D. Suciu. Optimizing regular path expressions using graph schemas. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pages 14–23, 1998.

[47] D. Florescu, A. Levy, and D. Suciu. Query containment for conjunctive queries with regular expressions. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 139–148, 1998.

[48] D. Florescu, A. Y. Levy, I. Manolescu, and D. Suciu. Query optimization in the presence of limited access patterns. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 311–322, 1999.

[49] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI'99)*, pages 67–73. AAAI Press/The MIT Press, 1999.

[50] H. Galhardas, D. Florescu, D. Shasha, and E. Simon. An extensible framework for data cleaning. Technical Report 3742, INRIA, Rocquencourt, 1999.

[51] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Widom. The TSIMMIS approach to mediation: Data models and languages. *J. of Intelligent Information Systems*, 8(2):117–132, 1997.

[52] C. H. Goh, S. Bressan, S. E. Madnick, and M. D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. on Information Systems*, 17(3):270–293, 1999.

[53] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 332–347. Springer, 1999.

[54] G. Greco, S. Greco, and E. Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proc. of the 17th Int. Conf. on Logic Programming (ICLP'01)*, volume 2237 of *Lecture Notes in Artificial Intelligence*, pages 348–364. Springer, 2001.

[55] S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proc. of the Int. Workshop on the Web and Databases (WebDB'01)*, 2001.

[56] S. Grumbach, M. Rafanelli, and L. Tininini. Querying aggregate data. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 174–184, 1999.

[57] S. Grumbach and L. Tininini. On the content of materialized aggregate views. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*, pages 47–57, 2000.

[58] M. Gruninger and J. Lee. Ontology applications and design. *Communications of the ACM*, 45(2):39–41, 2002.

[59] J. Gryz. Query folding with inclusion dependencies. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pages 126–133, 1998.

[60] A. Y. Halevy. Answering queries using views: A survey. *Very Large Database J.*, 10(4):270–294, 2001.

[61] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, 1997.

[62] T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

[63] A. C. Klug. On conjunctive queries containing inequalities. *J. of the ACM*, 35(1):146–160, 1988.

[64] D. Lembo, M. Lenzerini, and R. Rosati. Source inconsistency and incompleteness in data integration. In *Proc. of the 9th Int. Workshop on Knowledge Representation meets Databases (KRDB 2002)*, 2002.

[65] A. Y. Levy. Obtaining complete answers from incomplete databases. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 402–412, 1996.

[66] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95)*, pages 95–104, 1995.

[67] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogenous information sources using source descriptions. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, 1996.

[68] A. Y. Levy, A. Rajaraman, and J. D. Ullman. Answering queries using limited external query processors. In *Proc. of the 15th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'96)*, pages 227–237, 1996.

[69] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proc. of the 12th Eur. Conf. on Artificial Intelligence (ECAI'96)*, pages 323–327, 1996.

[70] A. Y. Levy and M.-C. Rousset. Combining horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[71] A. Y. Levy and D. Suciu. Deciding containment for queries with complex objects. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, pages 20–31, 1997.

[72] C. Li, M. Bawa, and J. D. Ullman. Minimizing view sets without loosing query-answering power. In *Proc. of the 8th Int. Conf. on Database Theory (ICDT 2001)*, pages 99–103, 2001.

[73] C. Li and E. Chang. Query planning with limited source capabilities. In *Proc. of the 16th IEEE Int. Conf. on Data Engineering (ICDE 2000)*, pages 401–412, 2000.

[74] C. Li and E. Chang. On answering queries in the presence of limited access patterns. In *Proc. of the 8th Int. Conf. on Database Theory (ICDT 2001)*, pages 219–233, 2001.

[75] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. D. Ullman, and M. Valiveti. Capability based mediation in TSIMMIS. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 564–566, 1998.

[76] J. Lin and A. O. Mendelzon. Merging databases under constraints. *Int. J. of Cooperative Information Systems*, 7(1):55–76, 1998.

[77] J. W. Lloyd. *Foundations of Logic Programming (Second, Extended Edition)*. Springer, Berlin, Heidelberg, 1987.

[78] I. Manolescu, D. Florescu, and D. Kossmann. Answering XML queries on heterogeneous data sources. In *Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB 2001)*, pages 241–250, 2001.

[79] T. D. Millstein, A. Y. Levy, and M. Friedman. Query containment for data integration systems. In *Proc. of the 19th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2000)*, pages 67–75, 2000.

[80] T. Milo and D. Suciu. Index structures for path expressions. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, volume 1540 of *Lecture Notes in Computer Science*, pages 277–295. Springer, 1999.

[81] F. Naumann, U. Leser, and J. C. Freytag. Quality-driven integration of heterogenous information systems. In *Proc. of the 25th Int. Conf. on Very Large Data Bases (VLDB'99)*, pages 447–458, 1999.

[82] Y. Papakonstantinou and V. Vassalos. Query rewriting using semistructured views. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, 1999.

[83] E. Rahn and P. A. Bernstein. A survey of approaches to automatic schema matching. *Very Large Database J.*, 10(4):334–350, 2001.

[84] A. Rajaraman, Y. Sagiv, and J. D. Ullman. Answering queries using templates with binding patterns. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95)*, 1995.

[85] R. Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 119–140. Plenum Publ. Co., New York, 1978.

[86] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. of the ACM*, 27(4):633–655, 1980.

[87] D. Srivastava, S. Dar, H. V. Jagadish, and A. Levy. Answering queries with aggregation using views. In *Proc. of the 22nd Int. Conf. on Very Large Data Bases (VLDB'96)*, pages 318–329, 1996.

[88] O. G. Tsatalos, M. H. Solomon, and Y. E. Ioannidis. The GMAP: A versatile tool for phyisical data independence. *Very Large Database J.*, 5(2):101–118, 1996.

[89] J. D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.

[90] R. van der Meyden. *The Complexity of Querying Indefinite Information*. PhD thesis, Rutgers University, 1992.

[91] R. van der Meyden. Logical approaches to incomplete information. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998.

[92] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95)*, pages 4–18, 1995.