Document Number:	D1.1
Project Name:	Infomix: Boosting Information Integration
Project Number:	IST-2001-33570
Title:	**Review on models and systems for information integration**
Authors:	Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
Workpackage:	WP1
Document Type:	Deliverable
Classification:	Confidential
Distribution:	Public
Status:	Final
Document file:	
Version:	2.0
Date:	July 31, 2002
Number of pages:	45

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	July 20, 2002	First draft
v.1.1	July 27, 2002	Review from the project partners
v.1.2	July 30, 2002	Revised version
v.2.0	July 31, 2002	Final version

Abstract

A Data Integration System is constituted by three main components: a global schema, a source schema, comprising the schemas of all the sources, and a mapping between the two. There exist two main approaches for specifying the mapping: in the *local-as-view* (LAV) approach the source structures are defined as views over the global schema; on the contrary, in the *global-as-view* (GAV) approach each global element is defined in terms of a view over the source schemas. The problem of query processing is to find efficient methods for answering queries posed to the global schema on the basis of the data stored at sources. In LAV there exist two approaches to query processing: by *query rewriting*, in which one tries to compute a rewriting of the query in terms of the views and then evaluates such a rewriting, and by *query answering*, in which one aims at directly answering the query based on the view extensions. In GAV, existing systems deal with query processing by simply *unfolding* each global concept in the query with its definition in terms of the sources. In this report, we survey the most important query processing algorithms proposed in the literature for LAV, and we describe the principal GAV data integration systems and the form of query processing they adopt. Furthermore, we review recent studies showing that, in the presence of incomplete data sources and integrity constraints on the global schema, query processing in GAV is harder than simple unfolding. Finally, we analyze the main approaches to query processing in the presence of inconsistent data with respect to the global schema.

1 Introduction

Information integration is the problem of combining the data residing at different sources, and providing the user with a unified view of these data, called *global schema*. The global schema is therefore a reconciled view of the information, which can be queried by the user. It can be thought of as a set of virtual relations, in the sense that their extensions are not actually stored anywhere. A data integration system frees the user from having to locate the sources relevant to a query, interact with each source in isolation, and manually combine the data from different sources.

The interest in this kind of systems has been continuously growing in the last years. Many organizations face the problem of integrating data residing at several sources. Companies that build a Data Warehouse, a Data Mining, or an Enterprise Resource Planning system must address this problem. Also, integrating data in the World Wide Web is the subject of several investigations and projects nowadays. Finally, applications requiring accessing or re-engineering legacy systems must deal with the problem of integrating data stored in different sources.

The design of a data integration system is a very complex task, which comprises several different issues, including the following:

1. heterogeneity of the sources,
2. mapping between the global schema and the sources,
3. limitations on the mechanisms for accessing the sources,

4. materialized vs. virtual integration,
5. data cleaning and reconciliation,
6. how to process queries expressed on the global schema,
7. how to deal with integrity constraints.

Problem (1) arises because sources are typically heterogeneous, meaning that they adopt different models and systems for storing data. This poses challenging problems in both representing the sources in a common format within the integration system, and specifying the global schema. As for the first issue, data integration systems make use of suitable software components, called wrappers, that present data at the sources in the form adopted within the system, hiding the original structure of the sources and the way in which they are modeled. With regard to the specification of the global schema, the goal is to design such a schema so as to provide an appropriate abstraction of all the data residing at the sources. One aspect deserving special attention is the choice of the language used to express the global schema. Since such a view should mediate among different representations of overlapping worlds, the language should provide flexible and powerful representation mechanisms.

With regard to Problem (2), two basic approaches have been used to specify the mapping between the sources and the global schema. The first approach, called query-centric or *global-as-view* (GAV), requires that the global schema is expressed in terms of the data sources. More precisely, to every concept of the global schema, a view over the data sources is associated, so that its meaning is specified in terms of the data residing at the sources. The second approach, called source-centric or *local-as-view* (LAV), requires the global schema to be specified independently of the sources. The relationships between the global schema and the sources are established by associating each element of the sources with a view over the global schema. Thus, in the local-as-view approach, we specify the meaning of the sources in terms of the concepts in the global schema. It is clear that the latter approach favors the extensibility of the integration system, and provides a more appropriate setting for its maintenance. For example, adding a new source to the system requires only to provide the definition of the source, and does not necessarily involve changes in the global view. On the contrary, in the global-as-view approach, adding a new source may in principle require changing the definition of the concepts in the global schema. A different approach could be to specify the mapping by combining LAV and GAV views together. This approach, called *global-local-as-view* (GLAV), is quite recent and has so far drawn little attention in the literature (see, e.g., [50]), thus it will not be discussed further in this review.

Problem (3) refers to the fact that, both in the local-as-view and in the global-as-view approach, it may happen that a source presents some limitations on the types of accesses it supports. A typical example is a web source accessible through a form where one of the fields must necessarily be filled in by the user. This can be modeled by specifying the source as a relation supporting only queries with a selection on a column. Suitable notations have been proposed for such situations [91], and the consequences of these access limitations on query processing in integration systems have been investigated in several papers [91, 81, 49, 104, 103, 79, 80].

Problem (4) deals with a further criterion that one should take into account in the design of a data integration system. In particular, with respect to the data explicitly managed by the system, one can follow two different approaches, called *materialized* and *virtual*. In the materialized approach, the system computes the extensions of the concepts in the global schema by replicating the data at the sources. In the virtual approach, data residing at the sources are accessed during query processing, but they are not replicated in the integration system. Obviously, in the materialized approach, the problem of refreshing the materialized views in order to keep them up-to-date is a major issue [61]. Unless otherwise specified, in the following we only deal with the virtual approach.

Whereas the construction of the global schema concerns the intentional level of the data integration system, Problem (5) refers to a number of issues arising when considering integration at the extensional/instance level. A first issue in this context is the interpretation and merging of the data provided by the sources. Interpreting data can be regarded as the task of casting them into a common representation. Moreover, the data returned by various sources need to be converted/reconciled/combined to provide the data integration system with the requested information. The complexity of this reconciliation step is due to several problems, such as possible mismatches between data referring to the same real world object, possible errors in the data stored in the sources, possible inconsistencies between values representing the properties of real world objects in different sources [51]. The above task is known in the literature as *Data Cleaning and Reconciliation*, and the interested reader is referred to [51, 20, 11] for more details on this subject.

Problem (6) is concerned with one of the most important issues in a data integration system, i.e., the choice of the method for computing the answer to queries posed in terms of the global schema only on the basis of the data residing at the sources. The main issue is that the system should be able to re-express such queries in terms of a suitable set of queries posed to the sources, hand them to the sources, and assemble the results into the final answer.

Finally, Problem (7) arises because the language adopted to represent the integration domain should be powerful enough to cope with the issues highlighted at point (1), i.e. should be based on integrity constraints. Generally, in data integration, data at the sources are assumed to be coherent with the integrity constraints specified over the sources to which they belong, thus such constraints can be overlooked during query processing. On the other hand, data originally stored in autonomous sources may not satisfy the integrity constraints expressed in the global schema, and inconsistencies may arise in the integration system. Furthermore, integrity constraints represent fundamental knowledge about the real world and important requirements that the reconciled data have to respect, so that they cannot be neglected during query processing. Generally speaking, in the presence of some inconsistencies, traditional semantics for data integration systems consider the entire system inconsistent and are unable to support query processing even if most of the data at the sources satisfy the integrity constraints in the global schema. More recent approaches [82, 6, 7, 53] consider different semantics able to provide database instances for the global schema even in the presence of inconsistencies, and define techniques to compute answers to queries in such a scenario.

In the rest of the paper, we review different data integration systems specified by means of either the LAV or the GAV approach, the particular models they adopt to represent both the sources and the global schema and the algorithms they propose to process queries. We first provide a logical formalization of the integration context in terms of a general framework for data integration systems, comprising a global schema, a source schema, and a mapping between the two. Such a framework poses several challenges, specially related to the need of taking the semantics of the global schema into account during query processing. In particular, the semantics of a data integration system, either following a LAV or a GAV mapping, is best described in terms of a set of databases, rather than a single one, as commonly assumed in GAV data integration systems. Then we discuss the problem of query processing, both in LAV and in GAV:

- In LAV, most of the proposed solutions are based on query answering by query rewriting. The problem of *query rewriting (using views)* consists of reformulating the query into a (possibly) equivalent expression, called *rewriting*, that refers only to the source structures. Once the rewriting of the query has been computed, it can be directly evaluated over the source to obtain the answer to the query. We observe that, besides data integration, the problem of query rewriting is relevant in several fields, including data warehousing [100], query optimization [32], and supporting physical data independence [95]. A more direct approach is that of *query answering using views*, in which besides the query and the mapping definitions, we are also given the extensions of the views over the global schema. The goal is to compute the set of tuples that are the answer set of the query in all databases that are consistent with the information on the views. For the LAV approach we survey the most important query processing algorithms proposed in the literature.

- In GAV, most of the solutions proposed in the literature essentially reduce to *unfolding* the query over the global schema by substituting each global relation with its definition in terms of the sources. Actually, this means considering a particular setting in which the semantics of a data integration system is described in terms of a unique database. In most general settings, even in the global-as-view approach, query processing is intimately connected to the notion of querying *incomplete databases*, and unfolding does in general not suffice to provide answers to queries [16]. In this review, we describe some GAV data integration systems, the specific assumptions they introduce upon the general framework, and the form of query processing they adopt.

Finally, we address the problem of inconsistency in data integration and show how traditional approaches in general cannot properly handle the presence of integrity constraints expressed over the global schema. Then, we review several works carried out in the area of inconsistent databases and describe the semantics and the techniques that they propose to both *repair databases* and compute *consistent answers* from inconsistent databases.

The paper is organized as follows. Section 2 presents basic notions about the relational model, which is the one we adopt to describe the data integration framework. In Section 3, we illustrate a formal framework for data integration and describe the syntax (3.1) and the semantics (3.2) of a data integration system. Section 4 presents query processing in LAV (subsection 4.1) and GAV (subsection 4.2). In Section 5 we address the problem of data inconsistency in data integration (subsection 5.1), and survey the most relevant approaches presented in the recent literature (subsection 5.2). Finally, Section 6 concludes the paper.

2 Preliminary definitions

Since most of the works we survey in this report deal with relational databases, we illustrate here the basic notions of the relational model. Furthermore, the relational model is the one we adopt to describe in the next section a formal framework for data integration. In the relational model, predicate symbols are used to denote the relations in the database, whereas constant symbols denote the objects and the values stored in relations. We assume to have a fixed (infinite) alphabet Γ of constants, and we consider only databases over such an alphabet. We adopt the so-called *unique name assumption*, i.e., we assume that different constants denote different objects.

A *relational schema* $\mathcal{C}$ is constituted by:

- An alphabet $\mathcal{A}$ of *predicate* (or relation) symbols, each one with an associated arity denoting the number of arguments of the predicate (or attributes of the relation).
- A set $\mathcal{C}$ of *integrity constraints*, i.e., assertions on the symbols of the alphabet $\mathcal{A}$ that are intended to be satisfied in every database coherent with the schema.

A *relational database* (or simply, database, DB) $\mathcal{DB}$ over a schema $\mathcal{H}$ is simply a set of relations with constants as atomic values. We have one relation of arity n for each predicate symbol of arity n in the alphabet $\mathcal{A}$. The relation $r^{\mathcal{DB}}$ in $\mathcal{DB}$ corresponding to the predicate symbol R is constituted by a set of tuples of constants, those that satisfy the predicate R. A database $\mathcal{DB}$ over a schema $\mathcal{H}$ is said to be *coherent* with $\mathcal{H}$ if every constraint in $\mathcal{C}$ is satisfied by $\mathcal{DB}$. The notion of satisfaction depends on the type of constraints defined over the schema.

A relational query is a formula that specifies a set of data to be retrieved from a database. In the sequel we mainly refer to the class of conjunctive queries, union of conjunctive queries and Datalog queries. A *conjunctive query* (CQ) q of arity n over the schema $\mathcal{H}$ is written in the form

$$q(\vec{\mathbf{x}}) \leftarrow conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$$

where

- q belongs to a new alphabet $\mathcal{Q}$ (the alphabet of queries, that is disjoint from both Γ and $\mathcal{A}$),
- $q(\vec{\mathbf{x}})$ is the *head* of the conjunctive query,
- $conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$ is the *body* of the conjunctive query and is a conjunction of atoms involving the variables $\vec{\mathbf{x}} = X_1, \ldots, X_n$ and $\vec{\mathbf{y}} = Y_1, \ldots, Y_n$, and constants from Γ,
- the predicate symbols of the atoms are in $\mathcal{A}$,

- the number of variables of $\vec{\mathbf{x}}$ is called the *arity* of q, and is the arity of the relation denoted by the query q.

Notice that, the body of the query may also contain atoms whose predicates are arithmetic comparison predicates, i.e., built-in predicates, with the restriction that variables involved in such predicates must appear also in atoms whose predicate symbols are in $\mathcal{A}$.

Given a database $\mathcal{DB}$, the answer to q over $\mathcal{DB}$, denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $(c_1, \ldots, c_n)$, such that, when substituting each c_i for x_i, the formula

$$\exists \vec{\mathbf{y}}. conj(\vec{\mathbf{x}}, \vec{\mathbf{y}})$$

evaluates to true in $\mathcal{DB}$.

A set of conjunctive queries with the same head predicate is a *Union of Conjunctive Queries* (UCQ). More formally, a UCQ is written in the form

$$q(\vec{\mathbf{x}}) \leftarrow conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

The answer to a UCQ q over a database $\mathcal{DB}$, as usually denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $(c_1, \ldots, c_n)$, such that, when substituting each c_i for x_i, the formula

$$\exists \vec{\mathbf{y}}_1. conj_1(\vec{\mathbf{x}}, \vec{\mathbf{y}}_1) \vee \cdots \vee \exists \vec{\mathbf{y}}_m. conj_m(\vec{\mathbf{x}}, \vec{\mathbf{y}}_m)$$

evaluates to true in $\mathcal{DB}$.

Finally a Datalog query is a collection of rules, each having the same form as a conjunctive query, except that predicate symbols in the body of the rules can be in $\mathcal{Q}$ as well. In a Datalog query, each head predicate of the rules refers to an intermediate relation, and has not to contain predicates referring to database relations. The intermediate predicates are called *Intensional DataBase* (IDB) predicates, whereas predicates referring to stored relations are called *Extensional DataBase* (EDB) predicates. Given a Datalog query q and a database $\mathcal{DB}$, the answer $q^{\mathcal{DB}}$ of q over $\mathcal{DB}$ is the minimal fixpoint model of q and $\mathcal{DB}$ [2].

Given a CQ, UCQ, or Datalog query q, we also say that $q^{\mathcal{DB}}$ denotes the set of tuples that *satisfy* q over $\mathcal{DB}$.

3 Formal framework for data integration

In this section we set up a logical framework for data integration. The main components of a data integration system are the sources, the global schema and the mapping between the two. We first present the syntax, and then the semantics of a data integration system.

3.1 Syntax

We consider to have a fixed alphabet Γ of constants representing real world objects, and will take into account only databases having Γ as domain. Furthermore, we adopt the so-called *unique name assumption*, i.e., we assume that different constants denote different objects.

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

- $\mathcal{G}$ is the *global schema*, expressed in the global language $\mathcal{L}_\mathcal{G}$ over the alphabet $\mathcal{A}^\mathcal{G}$. The language $\mathcal{L}_\mathcal{G}$ determines the expressiveness allowed for specifying the global schema, i.e., the set of constraints that can be defined over it;

- $\mathcal{S}$ is the *source schema*, composed by the schemas of the various sources that are part of the data integration system. $\mathcal{S}$ is modeled in the source language $\mathcal{L}_\mathcal{S}$ over the alphabet $\mathcal{A}^\mathcal{S}$. Obviously, $\mathcal{A}^\mathcal{S}$ is disjoint from the alphabet of the global schema $\mathcal{A}^\mathcal{G}$ and, as in the previous case, $\mathcal{L}_\mathcal{S}$ determines the expressiveness allowed for specifying the source schema;

- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, i.e., the specification of the relationship between the sources and the global schema. Below, we describe the syntactic characteristics of the main existing approaches to specify the form of the mapping, while, in the next section, we investigate their semantics.

Thus, from the syntactic viewpoint, the specification of an integration system depends on the following parameters:

- The form of the global schema, i.e., the formalism used for expressing data and the global relationships between data. We assume that the global schema is expressed in the relational model, in which it is possible to define different kinds of integrity constraints. In this case, $\mathcal{A}^\mathcal{G}$ denotes the set of relational symbols of the global schema.

- The form of the source schema, i.e., the formalism used for expressing data at the sources. In particular, we assume that the sources are relational. Such an assumption is not restrictive since we can assume that suitable *wrappers* present the data at the sources in a relational form. According to this assumption, each source is modeled by a set of relations and $\mathcal{A}^\mathcal{S}$ denotes the alphabet of the relational symbols of the sources. Moreover, we assume that tuples at the sources satisfy all constraints specified on $\mathcal{S}$, thus, in the following we do not consider anymore these constraints.

- The form of the mapping. Specifically, we assume two possible forms for the mapping, called respectively *global-as-view* (GAV) and *local-as-view* (LAV). The GAV approach requires that the global schema is defined in terms of the data sources: more precisely, every relation of the global schema is expressed as a view over the sources, so that its meaning is specified in terms of the data residing at the sources. In the LAV approach, the meaning of the sources is specified in terms of the elements of the global schema: more exactly, the mapping $\mathcal{M}$ between the sources and the global schema is provided in terms of a set of views over the global schema, one for each source element. Hence, in both the approaches the mapping is provided in terms of a set of views, i.e., queries, expressed in a particular *view language* $\mathcal{L}_\mathcal{V}$, and issued either over the global schema $\mathcal{G}$ or over the source schema $\mathcal{S}$ in LAV and in GAV, respectively.

Finally, we consider *queries* posed to a data integration system and define their syntax. Each such query is a formula that is intended to provide the specification of which data to extract from the integration system, i.e., q is intended to extract a set of tuples of elements of Γ. Thus, every query has an associated arity. Each query is issued over the global schema $\mathcal{G}$, and is expressed in a specific query language, denoted by $\mathcal{L}_{\mathcal{Q}}$, over the global alphabet $\mathcal{A}^{\mathcal{G}}$.

3.2 Semantics

Intuitively, to specify the semantics of a data integration system, we have to start with a set of data at the sources, and, given such data at the sources, we have to specify which are the data that satisfy the global schema. Thus, in order to assign the 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 $\mathcal{D}$ for the source schema $\mathcal{S}$.

Based on $\mathcal{D}$, we now specify which is the information content of the global schema $\mathcal{G}$. We call any database for $\mathcal{G}$ a *global database* for $\mathcal{I}$. A global database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal* with respect to $\mathcal{D}$, if:

- $\mathcal{B}$ is coherent with $\mathcal{G}$.
- $\mathcal{B}$ satisfies the mapping with respect to $\mathcal{D}$, that is its tuples respect the relationships defined between the global and the source relations. We will explain the exact meaning of satisfying the mapping in the next subsections.

Given a source database $\mathcal{D}$ for $\mathcal{I}$, the semantics of $\mathcal{I}$ with respect to $\mathcal{D}$, denoted $sem(\mathcal{I}, \mathcal{D})$, is defined as follows:

$$sem(\mathcal{I}, \mathcal{D}) = \{ \ \mathcal{B} \mid \mathcal{B} \text{ is a legal global database for } \mathcal{I} \text{ w.r.t. } \mathcal{D} \ \}$$

Let us now turn our attention to queries. In order to define the semantics of a query q over the global schema $\mathcal{G}$, we have to take into account all the global databases legal for $\mathcal{I}$ with respect to $\mathcal{D}$. We call *certain answers* of a query q of arity n with respect to $\mathcal{I}$ and $\mathcal{D}$, the set $q^{\mathcal{I},\mathcal{D}}$ of n-tuples t such that $t \in q^{\mathcal{DB}}$ for *every* database $\mathcal{DB} \in sem(\mathcal{I}, \mathcal{D})$.

Furthermore, we call *possible answers* of a query q the set $q^{\mathcal{I},\mathcal{D}}$ of n-tuples t such that $t \in q^{\mathcal{DB}}$ for *some* database $\mathcal{DB} \in sem(\mathcal{I}, \mathcal{D})$.

In the sequel, we only refer to certain answers, simply called answers to query.

From the above definitions, it is easy to see that, in data integration, answering queries is essentially an extended form of reasoning in the presence of incomplete information [98]. Indeed, when we answer the query, we know only the extensions of the sources, and this provides us with only partial information on the global database.

It is well known that query processing in LAV is similar to query answering with incomplete information, whereas it is common opinion that query processing is much easier in GAV. In the latter approach, it is generally assumed that the semantics of a data integration system comprises a single database, and that a query can be answered by simply *unfolding* its atoms with their definitions over the sources. On the other hand, from the definition of $sem(\mathcal{I}, \mathcal{D})$ follows that, given a source database $\mathcal{D}$, in general, several global legal databases exist either in LAV or in GAV. In specific cases, no global legal database may exist, for example if data at the sources are inconsistent with respect to integrity constraints expressed in the global schema, or exactly one legal database exist. We will give some examples in Subsections 3.2.1 and 3.2.2, where we discuss the precise meaning of mapping satisfaction either in the LAV or in the GAV approach.

3.2.1 Semantics of the LAV mapping

In the LAV approach, the meaning of the sources is specified in terms of the elements of the global schema. More precisely, the mapping $\mathcal{M}$ between the sources and the global schema is provided in terms of a set of assertions of the form $\langle r, V \rangle$, where r is a relation of the source schema $\mathcal{S}$ and V is the associated view over the global schema $\mathcal{G}$.

For each such mapping assertion we have:

- V is expressed over the alphabet $\mathcal{A}^{\mathcal{G}}$;
- $V^{\mathcal{B}}$ indicates the set of tuples in the answer to V over a given global database $\mathcal{B}$, i.e. the set of tuples that satisfy V over $\mathcal{B}$;
- $r^{\mathcal{D}}$ indicates the actual set of tuples that satisfy the source relation r for a given source database $\mathcal{D}$, i.e., the extension of such source relation in $\mathcal{D}$;

Associated to each mapping assertion $\langle r, V \rangle$ we have a specification $as(V)$ of which *assumption* to adopt for the view V, i.e., given a source database $\mathcal{D}$, how to interpret $r^{\mathcal{D}}$ with respect to the set of tuples in the answer to V over a global database $\mathcal{B}$, i.e., $V^{\mathcal{B}}$.

The following three assumptions for a view V have been considered in the literature [1, 52, 69, 22] [1]:

- *Sound Views.* When a view V is *sound* (denoted with $as(V) = sound$), the extension of the associated source relation r provides a subset of the tuples satisfying V. In other words, from the fact that a tuple is in $r^{\mathcal{D}}$ one can conclude that it satisfies the corresponding view V, while from the fact that a tuple is not in $r^{\mathcal{D}}$ one cannot conclude that it does not satisfy V. Formally, a global database $\mathcal{B}$ *satisfies the sound view* V, if $r^{\mathcal{D}} \subseteq V^{\mathcal{B}}$.
- *Complete Views.* When a view V is *complete* (denoted with $as(V) = complete$), the extension of the associated source relation r provides a superset of the tuples satisfying V. In other words, from the fact that a tuple is in $r^{\mathcal{D}}$ one cannot conclude that such a tuple satisfies V. On the other hand, from the fact that a tuple is not in $r^{\mathcal{D}}$ one can conclude that such a tuple does not satisfy V. Formally, a global database $\mathcal{B}$ *satisfies the complete view* V, if $r^{\mathcal{D}} \supseteq V^{\mathcal{B}}$.
- *Exact Views.* When a view V is *exact* (denoted with $as(V_i) = exact$), the extension of the associated source relation r is exactly the set of tuples satisfying V. Formally, a global database $\mathcal{B}$ *satisfies the exact view* V, if $r^{\mathcal{D}} = V^{\mathcal{B}}$

[1] In some papers, for example [22], different assumptions on the domain of the database are also taken into account.

This notion of satisfaction of the mapping completes the definition of global legal database for the LAV approach.

According to the above definitions, in general several global legal databases may exist, e.g., in the presence of sound and/or complete views. However, in particular situations, exactly one global legal database exists, as shown in Example 3.1. In the examples of this section, we only consider mapping in which the views are all exact or all sound, since these are the most commonly addressed cases in the literature. For complete views see Example 5.1, in which some (GAV) views are considered sound and other views complete.[2]

Example 3.1 Consider a data integration system $\mathcal{I}$ comprising two sources $\mathsf{s}_1(\mathit{Name}, \mathit{City})$, which stores information about people and cities in which they reside and $\mathsf{s}_2(\mathit{City}, \mathit{Country})$, which stores information about countries where cities are located.

Suppose the global schema $\mathcal{G}$ contains only the relation $\mathsf{residence}(\mathit{Name}, \mathit{City}, \mathit{Country})$ and that the mapping $\mathcal{M}$ is as follows:

$$\begin{array}{l} \langle \mathsf{s}_1, \quad \mathsf{s}_1(N, C) \leftarrow \mathsf{residence}(N, C, Co)\rangle \\ \langle \mathsf{s}_1, \quad \mathsf{s}_2(C, Co) \leftarrow \mathsf{residence}(N, C, Co)\rangle \end{array}$$

and that we have an (infinite) alphabet Γ containing (among other symbols) three constants *Pablo*, *Rome* and *Italy*.[3]

Finally consider a source legal database $\mathcal{D}$ in which $\mathsf{s}_1^{\mathcal{D}} = \{(\mathit{Pablo}, \mathit{Rome})\}$ and $\mathsf{s}_2^{\mathcal{D}} = \{(\mathit{Rome}, \mathit{Italy})\}$, and assume that there are no constraints over the global schema.

If both views are sound, there are several global databases coherent with the global schema and the mapping, since we only know that the global relation $\mathsf{residence}$ contains some tuples that have *Pablo* as their *Name* component and *Rome* as their *City* component, and some tuples that have *Rome* as their *City* component and *Italy* as their *Country* component. Therefore, the query $q(x, y, z) \leftarrow \mathsf{residence}(x, y, z)$ asking for all residence information would return an empty answer, i.e., $q^{\mathcal{I},\mathcal{D}} = \emptyset$.

However, if both views are exact, we can conclude that all $\mathsf{residence}$ tuples have *Pablo* as their *Name* component, *Rome* as their *City* component and *Italy* as their *Country* component. Hence $(\mathit{Pablo}, \mathit{Rome}, \mathit{Italy})$ is the only tuple returned by the query q, i.e., $q^{\mathcal{I},\mathcal{D}} = \{(\mathit{Pablo}, \mathit{Rome}, \mathit{Italy})\}$. Note that in this case we have exactly one legal global database. ■

Let us now consider integrity constraints expressed over the global schema. In general, data migrating from the sources through the global schema may not satisfy these constraints, and inconsistencies may arise in the integration system. In this situation, it may happen that no legal global database exists, for example if all views are exact. However, in the presence of sound (or complete) views, there are in general several ways to add (resp. delete) tuples to (from) relations of $\mathcal{G}$ in order to satisfy the constraints.

[2] Notice that, in a data integration context, considering all views complete does not seem particularly significant.

[3] We consider only databases over a fixed alphabet of constants (see section 3).

Example 3.2 Suppose now to add to our running example a source $\mathsf{s}_3(Name, Country)$, which stores information about people citizenship, and add a global relation

$$\mathsf{citizenship}(Name, Country)$$

We complete our mapping with the assertion

$$\langle \mathsf{s}_3, \quad \mathsf{s}_3(N, Co) \leftarrow \mathsf{citizenship}(N, Co) \rangle$$

and assume to define a foreign key constraint between the component $Name$ of the relation residence and the component $Name$ in the relation citizenship. This constraint is satisfied by a database $\mathcal{DB}$ if for every tuple $t_1 \in (\mathsf{residence})^{\mathcal{DB}}$ there is a tuple $t_2 \in (\mathsf{citizenship})^{\mathcal{DB}}$ such that the $Name$ component of t_1 agrees with the $Name$ component in t_2. Consider now the constants *John* and *USA* and the same source legal database $\mathcal{D}$ of Example 3.1, and suppose that $\mathsf{s}_3^{\mathcal{D}} = \{(\mathsf{John}, USA\}$. If both views are exact, no global database exists which is coherent with the global schema and satisfies the mapping, since s_3 does not store *Pablo*'s citizenship, i.e., there is no legal global database. On the contrary, if all views are sound, then all the global databases that store *Pablo*'s citizenship are global legal databases, since they satisfy the soundness of the mapping and are coherent with the global schema. For example, consider the global databases $\mathcal{B}$ such that $\mathsf{citizenship}^{\mathcal{B}}$ is as follows:

Name	*Country*
John	*USA*
Pablo	α

where α is any constant in Γ. ■

It is worthwhile noticing that, also in the case when views are sound or complete, it might happen that no global legal database exists. Intuitively, in the former case this might happen since we can make inconsistent data satisfy the integrity constraints by suitably adding tuples in the global schema, but we are not allowed to solve inconsistencies by disregarding tuples. Conversely, in the latter case we can delete tuples but we are not allowed to add new tuples. Generally speaking, integrating data in the presence of incomplete and inconsistent data sources poses new challenging issues that cannot be addressed according to the semantics described above, and the necessity arises to resort to *nonmonotonic* semantics. We will turn back to this point in Section 5.

3.2.2 Semantics of the GAV mapping

The GAV approach requires that the global schema is defined in terms of the data sources. More exactly, every relation of the global schema is expressed as a view over the sources, so that its meaning is specified in terms of the data residing at the sources. Note that, while in the LAV approach adding a new source to the system requires only to provide the definition of the source, and does not necessarily involve changes in the global schema, in the GAV approach it may in principle require to change the definition of the concepts in the global schema.

Formally, the mapping $\mathcal{M}$ between the global schema and the sources is provided in terms of a set of assertions of the form $\langle r, V\rangle$, where r is a relation of the global schema $\mathcal{G}$ and V is the associated view over the source schema $\mathcal{S}$.

For each such mapping assertion we have:

- V is expressed over the alphabet $\mathcal{A}^{\mathcal{S}}$;
- $V^{\mathcal{D}}$ indicates the set of tuples in the answer to V over a given source database $\mathcal{D}$, i.e. the set of tuples that satisfy V over $\mathcal{D}$;
- $r^{\mathcal{B}}$ indicates the set of tuples that satisfy the global relation r for a given global database $\mathcal{B}$, i.e., the extension of such global relation in $\mathcal{B}$.

As in the LAV approach, associated to each mapping assertion $\langle r, V\rangle$ we have a specification $as(V)$ of which *assumption* to adopt for the view V, i.e., how to interpret the actual set of tuples in the answer to V over a given source database $\mathcal{D}$, i.e., $V^{\mathcal{D}}$, with respect to the set of tuples that satisfy the corresponding global relation r for a given global database $\mathcal{B}$, i.e., $r^{\mathcal{B}}$.

Also in this case, one can consider views that are *sound*, *complete,* or *exact*:

- when $as(V) = sound$, the extension of the associated global relation r provides any superset of the tuples satisfying V. In other words, from the fact that a tuple is in $V^{\mathcal{D}}$ one can conclude that it satisfies the corresponding global relation r, while from the fact that a tuple is not in $V^{\mathcal{D}}$ one cannot conclude that it does not satisfy r. Formally, a global database $\mathcal{B}$ *satisfies the sound view* V, if $V^{\mathcal{D}} \subseteq r^{\mathcal{B}}$.
- when $as(V) = complete$, the extension of the associated global relation r provides any subset of the tuples satisfying V. In other words, from the fact that a tuple is in $V^{\mathcal{D}}$ one cannot conclude that such a tuple satisfies r. On the other hand, from the fact that a tuple is not in $V^{\mathcal{D}}$ one can conclude that such a tuple does not satisfy r. Formally, a global database $\mathcal{B}$ *satisfies the complete view* V, if $V^{\mathcal{D}} \supseteq r^{\mathcal{B}}$.
- when $as(V_i) = exact$, the extension of the associated global relation r is exactly the set of tuples satisfying V. Formally, a global database $\mathcal{B}$ *satisfies the exact view* V, if $V^{\mathcal{D}} = r^{\mathcal{B}}$.

Notice that, in the GAV approach, one can easily compute a global database that satisfies the mapping by simply evaluating the views over the given source database $\mathcal{D}$. We call such a database the *retrieved global database* $ret(\mathcal{I}, \mathcal{D})$. When the views are all exact, $ret(\mathcal{I}, \mathcal{D})$ is the only database that satisfies the mapping, whereas in the presence of sound or complete views a set of global databases satisfies the mapping. Intuitively, these are the global databases obtained by adding tuples to each $r^{ret(\mathcal{I},\mathcal{D})}$ if the view associated to r is sound, or deleting tuples from it if the associated view is complete.

Obviously, in order to be legal databases, the global databases obtained as described above have to satisfy also the constraints expressed over the global schema $\mathcal{G}$. This does not hold in general, and it might happen that no legal global database exists. Even if most of the GAV data integration systems described in the literature consider a single database for the global schema, from the above definitions it should be clear that we have to account for a set of global legal databases, and, as for the LAV approach, given a source database, different situations are possible: several global legal database exist, exactly one legal database exists, no global legal database exists.

The following example illustrates the above considerations for the case where foreign key constraints are used in the global schema.[4]

Example 3.3 Consider a data integration system $\mathcal{I}$ in which the source schema $\mathcal{S}$, the alphabet of constants, the global relations, and the integrity constraints are as in Example 3.2. Suppose to have the following GAV mapping $\mathcal{M}$:

$$\begin{array}{lll}\langle \mathsf{residence}, \mathsf{residence}(N, C, Co) & \leftarrow & \mathsf{s}_1(N, C,) \wedge \mathsf{s}_2(C, Co)\rangle \\ \langle \mathsf{citizenship}, \mathsf{citizenship}(N, Co) & \leftarrow & \mathsf{s}_3(N, Co)\rangle\end{array}$$

and the source database $\mathcal{D}$ in which $\mathsf{s}_1^{\mathcal{D}} = \{(\mathit{Pablo}, \mathit{Rome})\}$, $\mathsf{s}_2^{\mathcal{D}} = \{(\mathit{Rome}, \mathit{Italy})\}$ and $\mathsf{s}_3^{\mathcal{D}} = \{(\mathit{Pablo}, \mathit{Spain})\}$ In such a case the data at the sources respect the foreign key constraint. So, if the views are all exact, exactly one global legal database exists, i.e., the retrieved global database $\mathit{ret}(\mathcal{I}, \mathcal{D})$. Therefore, given the query $q(x, y, z) \leftarrow \mathsf{residence}(x, y, z)$ we have that $q^{\mathcal{I},\mathcal{D}} = \{(\mathit{Pablo}, \mathit{Rome}, \mathit{Italy})\}$. Consider the different situation in which s_1 and s_2 have the same extensions as above, but now is $\mathsf{s}_3^{\mathcal{D}} = \{(\mathit{John}, \mathit{USA})\}$. Figure 1 shows $\mathit{ret}(\mathcal{I}, \mathcal{D})$.

$\mathsf{residence}^{\mathit{ret}(\mathcal{I},\mathcal{D})}$:

Name	*City*	*Country*
Pablo	*Rome*	*Italy*

$\mathsf{citizenship}^{\mathit{ret}(\mathcal{I},\mathcal{D})}$:

Name	*Country*
John	*USA*

Figure 1: retrieved global database

Since data at the sources do not respect the foreign key constraint, no global legal database exists in such a situation.

On the other hand, if the views are all sound we have to consider as legal all the databases that store Pablo's citizenship, i.e., all the databases obtained from $\mathit{ret}(\mathcal{I}, \mathcal{D})$ by adding tuples of the form $\{(\mathit{Pablo}, \alpha)\}$ to $\mathsf{citizenship}$, where α is any constant in Γ. Note that we still have $q^{\mathcal{I},\mathcal{D}} = \{(\mathit{Pablo}, \mathit{Rome}, \mathit{Italy})\}$.

■

As in the LAV approach, the semantics above described is not always able to deal with data inconsistency, since, for example, we might be forced to satisfy integrity constraints by deleting tuples from $\mathit{ret}(\mathcal{I}, \mathcal{D})$ (e.g., in the presence of key constraints on the global relations) but in the presence of sound or exact views we are not allowed to delete tuples. We will turn back to this problem in Section 5

[4]As in Example 3.1, we consider only cases in which the views are all exact or sound.

4 Query processing

In this section we deal with the problem of query processing, i.e., how to compute the answer to a query posed over the global schema on the basis of the data stored at the sources and exploiting the information available about the data integration system. We examine some algorithms and techniques for query processing both in LAV and in GAV integration systems. For the LAV framework we distinguish between two existing approaches: query processing by *query rewriting*, which aims at reformulating the query in terms of the source relations and then evaluating the computed rewriting over the source extensions, and *query answering using views*, which aims at directly answering the query on the basis of both the view definitions and the source extensions. For the GAV framework we describe some data integration systems in which query processing essentially amounts to unfolding.

4.1 LAV approach

Since in LAV the mapping between the sources and the global schema is described as a set of views over the global schema, query processing amounts to finding a way to answer a query posed over a database using a set of views over the same database. This problem, called *answering queries using views*, is widely studied in the literature, since it has applications in many areas. In query optimization [32], the problem is relevant because using materialized views may speed up query processing. A data warehouse can be seen as a set of materialized views, and, therefore, query processing reduces to query answering using views [58, 94, 102, 57]. In the context of database design, using views provides means for maintaining the physical perspective of the data independent from its logical perspective [95]. It is also relevant for query processing in distributed databases [63] and in federated databases [77]. Finally, since the views provide partial knowledge on the database, answering queries using views can be seen as a special case of query answering with incomplete information [60, 1].

In the following we review the main algorithms proposed in the literature to answer a query in LAV. Notice that, with respect to the formal framework described in Section 3, these algorithms assume that the views in the mapping are all sound.

4.1.1 Query rewriting

The most common approach proposed in the literature to deal with the problem of query answering using views is by means of *query rewriting*. In query rewriting, a query and a set of view definitions over the global schema are provided, and the goal is to reformulate the query into an expression, the *rewriting*, that refers only to the views, and supplies the answer to the query. A query Q is expressed in a certain query language $\mathcal{L}_Q$ over the global alphabet $\mathcal{A}^{\mathcal{G}}$, while the rewriting is expressed in a query language $\mathcal{L}'_Q$ over the source alphabet $\mathcal{A}^{\mathcal{S}}$. $\mathcal{L}_Q$ and $\mathcal{L}'_Q$ can be different and hence of different expressive power. Query answering via query rewriting is divided in two steps, where the first one consists of reformulating the query in terms of the given query language $\mathcal{L}'_Q$ over $\mathcal{A}^{\mathcal{S}}$, and the second one evaluates the rewriting over the view extensions.

We first introduce the notion of containment between Datalog queries whose EDB predicates are the relations in $\mathcal{A}^{\mathcal{G}}$. Given two Datalog queries Q_1 and Q_2, we say that Q_1 is *contained in* Q_2, denoted by $Q_1 \subseteq Q_2$, if for all databases $\mathcal{DB}$ we have that $Q_1^{\mathcal{DB}} \subseteq Q_2^{\mathcal{DB}}$. We say that Q_1 and Q_2 are *equivalent* if $Q_1 \subseteq Q_2$ and $Q_2 \subseteq Q_1$.

The problem of query containment has been studied in various settings. In [31], NP-completeness has been established for conjunctive queries, and in [35] a multi-parameter analysis has been performed for the same case, showing that the intractability is due to certain types of cycles in the queries. In [64, 97], Π_2^p-completeness of containment of conjunctive queries with inequalities was proved, and in [92] the case of queries with the union and difference operators was studied. For various classes of Datalog queries with inequalities, decidability and undecidability results were presented in [33] and [97], respectively. Query containment under constraints has also been the subject of several investigations. For example, decidability of conjunctive query containment was investigated in [5] under functional and multi-valued dependencies, in [62] under functional and inclusion dependencies, in [30, 76, 78] under constraints representing *is-a* hierarchies and complex objects, and in [37] in the case of constraints represented as Datalog programs. For queries over semistructured data, query containment for conjunctive regular path queries, in which the atoms in the body are regular expressions over binary predicates, has been studied in [18, 48], and EXPSPACE completeness has been established in [23].

To formally define the notion of rewriting, we consider queries and views defining the source relations that are unions of conjunctive queries over the global schema.

Given such a query Q and a set of views $\mathcal{V} = \{V_1, \ldots, V_m\}$ over $\mathcal{A}^{\mathcal{G}}$, the query Q_r is a *rewriting* of Q using $\mathcal{V}$ if

- $Q_r \cup \mathcal{V}$ is contained in Q, and
- Q_r does not refer to the predicates of $\mathcal{A}^{\mathcal{G}}$, i.e., the relations of the global schema appear only in the view definitions and not in the bodies of the clauses of Q_r.

We illustrate the notion of rewriting with the following example, adapted from [71], in which queries are conjunctive queries.

Example 4.1 Consider two sources, $\mathsf{r}_1(\mathit{Title}, \mathit{Year}, \mathit{Director})$, which stores information about movies filmed by european directors since 1960, and $\mathsf{r}_2(\mathit{Title}, \mathit{Critique})$, which stores information about movie reviews since 1990.

Suppose we have the following global schema

$$\begin{array}{l} \mathsf{movie}(\mathit{Title}, \mathit{Year}, \mathit{Director}) \\ \mathsf{european}(\mathit{Director}) \\ \mathsf{review}(\mathit{Title}, \mathit{Critique}) \end{array}$$

and the mapping

$$\begin{array}{rcl} \langle \mathsf{r}_1, \mathsf{r}_1(T, Y, D) & \leftarrow & \mathsf{movie}(T, Y, D) \wedge \mathsf{european}(D) \wedge Y \geq 1960 \rangle \\ \langle \mathsf{r}_2, \mathsf{r}_2(T, R) & \leftarrow & \mathsf{movie}(T, Y, D) \wedge \mathsf{review}(T, R) \wedge Y \geq 1990 \rangle \end{array}$$

Consider the query

$$\mathsf{q}(T, R) \ \leftarrow \ \mathsf{movie}(T, 1998, D) \wedge \mathsf{review}(T, R)$$

asking for reviews of movies filmed in 1998. A rewriting of such a query is

$$\mathsf{qr}(T, R) \leftarrow \mathsf{r}_1(T, 1998, D) \wedge \mathsf{r}_2(T, R)$$

Indeed, the body of qr refers only to the source relations, and hence can be directly evaluated over the source extensions. Moreover, qr together with the view definitions is equivalent to the following query over the global schema

$$\mathsf{qr}(T, R) \leftarrow \mathsf{movie}(T, 1998, D) \wedge \mathsf{european}(D) \wedge \mathsf{review}(T, R)$$

in which we deleted the atoms $1998 \geq 1960$ and $1998 \geq 1990$, which evaluate to true. It is immediate to verify that qr is contained in q. ■

In the example above, since source r_1 provides only movies filmed by European directors, the computed rewriting does not return exactly the data about reviews of movies filmed in 1998, requested by the original query. Indeed, qr is not equivalent to q. In general, the set of available sources may not store all the data needed to answer a query, and therefore the goal is to find a query expression that provides all the answers that can be obtained from the views. So, whereas in different contexts, i.e., query optimization or maintaining physical data independence [70], the focus is on finding rewritings that are logically equivalent to the original query, in data integration systems one is interested in *maximally contained* rewritings [40], formally defined as follows.

Given a query Q, a set of views $\mathcal{V} = \{V_1, \ldots, V_m\}$, and a query language $\mathcal{L}_Q$, the query Q_m is a *maximally contained rewriting* of Q using $\mathcal{V}$ w.r.t. $\mathcal{L}_Q$ if

- Q_m is a rewriting of Q using $\mathcal{V}$,
- there is no rewriting $Q' \neq Q_m$ in $\mathcal{L}_Q$ such that $Q_m \subseteq Q'$.

It is worth noticing that, since in data integration systems the relations of the global schema are virtual and we cannot extract data from them, we are interested in rewritings that use only the views, reflected by the definition above. In [72] such rewritings are called *complete*, and in the following we refer only to complete rewritings.

An important theoretical result concerning the problem of query rewriting, in the case that views and queries are conjunctive queries, is presented in [72]. In that paper the authors show that, when a query a Q is a union of conjunctive queries, if an equivalent conjunctive rewriting of Q exists, then such a rewriting has at most as many atoms as Q. Such a result leads immediately to nondeterministic polynomial-time algorithms to find either equivalent conjunctive rewritings or maximally contained rewritings that are union of conjunctive queries. In both cases it is sufficient to consider each possible conjunction of views that produces a *candidate rewriting* whose size is less or equal to the size of the original query, and then check the correctness of the rewriting. Note that the number of candidate rewritings is exponential in the size of the query.

Bucket algorithm and its extensions In order to compute all the rewritings that are contained in (and not necessarily equivalent to) the original query, the *bucket algorithm*, presented in [74], improves the technique described above since it exploits a suitable heuristic that enables one to prune the space of candidate rewritings. The algorithm was proposed in the context of the Information Manifold (IM) system [77], a project developed at AT&T. IM handles the presence of inclusion and functional dependencies over the global schema and limitations in accessing the sources, and uses conjunctive queries as the language for describing the sources and querying the system.

To compute the rewriting of a query Q, the bucket algorithm proceeds in two steps:

1. for each atom g in Q, create a bucket that contains the views from which tuples of g can be retrieved, i.e., the views whose definition contains an atom to which g can be mapped in a rewriting of the query;

2. consider as candidate rewriting each conjunctive query obtained by combining one view from each bucket, and check by means of a containment algorithm whether such a query is contained in Q. If so, the candidate rewriting is added to the answer.

If the candidate rewriting is not contained in Q, before discarding it, the algorithm checks if it can be modified by adding comparison predicates in such a way that it is contained in Q. The proof that the bucket algorithm generates the maximal contained rewriting when the query language is union of conjunctive queries, is given in [52].

Note that, on the basis of the results in [72], the algorithm considers only rewritings that have at most the same number of atoms as the original query. As shown in the same paper and in [91], the given bound on the size of the rewriting does not hold in the presence of arithmetic comparison predicates, functional dependencies over the global schema, or limitations in accessing the sources. In such cases we have to consider rewritings that are longer than the original query. More precisely, let p be the number of atoms in the query Q:

- In the presence of functional dependencies, a minimal rewriting has at most $p + d$ atoms, where d is the sum of the arities of the atoms in Q;

- In the presence of limitations in accessing the sources, a minimal rewriting has at most $p + m$ atoms, where m is the number of different variables in Q;

- In the presence of comparison predicates, the size of a minimal rewriting is at most exponential in the size of Q.

According to [73] the bucket algorithm, in practice, does not miss solutions because of the length of the rewritings it considers, but other results [40, 56, 55] demonstrate that in the presence of functional dependencies and limitations in accessing the sources, union of conjunctive queries does not suffice to obtain the maximal contained rewritings, and one needs to resort to recursive rewritings. We refer the interested reader to [40, 41, 66] for a more detailed treatment of the problem.

Two improved versions of the bucket algorithm are the *MiniCon algorithm* [89] and the *shared-variable bucket algorithm* [84]. In both the algorithms the basic idea is to examine the interaction among variables of the original query and variables in the views, in order to reduce the number of views inserted in the buckets, and hence the number of candidate rewritings to be considered. Experimental results related to the performance of MiniCon and shared-variable bucket show that these algorithms scale very well and outperform the bucket algorithm.

Inverse rules algorithm The previous algorithms search the space of possible candidate rewritings by using buckets and then checks whether each computed rewriting is contained in the original query. In the following, we describe a different algorithm, namely the *inverse rules algorithm* [40], that generates a rewriting (query plan) in time that is polynomial in the size of the query. The algorithm was developed in the context of the Infomaster system [39], an information integration tool developed at Stanford University, based on a language for representing first order logic expressions, called *Knowledge Interchange Format*. The inverse rules algorithm constructs a set of rules that invert the view definitions and provides the system with an inverse mapping which establish how to obtain the data of the global concepts from the data of the sources. The basic idea is to replace existential variables in the body of each view definition by a Skolem function.

Example 4.2 Given a view definition

$$\mathsf{v}(X) \quad \leftarrow \quad \mathsf{a}_1(X,Y) \wedge \mathsf{a}_2(X,Y)$$

the set of inverse rules obtained from it is

$$\begin{array}{rcl} \mathsf{a}_1(X,f(X)) & \leftarrow & \mathsf{v}(X) \\ \mathsf{a}_2(X,f(X)) & \leftarrow & \mathsf{v}(X) \end{array}$$

■

Given a non-recursive Datalog query Q and a set of view definitions $\mathcal{V}$, the rewriting is the Datalog program consisting of both the query and the inverse rules obtained from $\mathcal{V}$.

Note that in the skolemization phase, we just introduce function symbols in the head of the inverse rules and never introduce a symbol within another, which leads to a finite evaluation process. Since bottom-up evaluation of the rewriting can produce tuples with function symbols, these need to be discarded. A polynomial time procedure to eliminate these function symbols by adding new predicates is described in [40]. It is also shown that the inverse rules algorithm returns a maximally contained rewriting w.r.t. union of conjunctive queries, in time that is polynomial in the size of the query[5]. Even if the computational cost of constructing the query plan is polynomial, the obtained rewriting contains rules which may cause accessing views that are irrelevant for the query. [68] show that the problem of eliminating irrelevant rules has exponential time complexity. A second drawback of the inverse rules algorithm is that evaluating the inverse rules over the source extension may invert some useful computation done to produce the views [89]. However, the inverse rules algorithm can handle also recursive Datalog queries, the presence of functional dependencies over the global schema, or the presence of limitations in accessing the sources, by extending the obtained query plan with other specific rules.

[5]Note that from a non-recursive Datalog program one can easily obtain an equivalent union of conjunctive queries.

Other algorithms and results Finally, we cite the *unification-join algorithm* [90], an exponential-time query answering algorithm based on a skolemization phase and on the representation of conjunctive queries by means of hypergraphs. In [90] is also described a polynomial time version of the algorithm developed for the case in which queries are acyclic conjunctive queries. The unification join algorithm is extended in [55, 56] in order to deal with inclusion and functional dependencies expressed over the global schema.

Other studies are concerned with the problem of query rewriting using views, under the different assumptions that queries are recursive queries [4], description logics queries [8, 20], queries for semi-structured data [21, 88], queries with aggregates [54, 36], or in presence of limitations in accessing the views [91, 66, 75, 49].

4.1.2 Query answering

In general, in a data integration system, given a legal source database $\mathcal{D}$, i.e., an extension for the sources, we are interested in computing the set of certain answers for a given query Q (see Section 3). In the query rewriting approach, first Q is reformulated (not taking into account the source extensions), and only in a second step the computed rewriting is evaluated over $\mathcal{D}$. So, the capability of the rewriting to retrieve the set of certain answers depends on the given query language in which it is expressed. Such a situation can constitute a limitation on computing the set of certain answers for Q.

A more general approach to query processing, namely *view-based query answering* [1, 52, 22, 24, 19], is that to consider, besides the query and the views, also the extensions of the sources. In view-based query answering, no limits are posed to query processing, and the only goal is to compute the set of certain answers to the query by exploiting all possible information, in particular the source extensions. Notice that, since the problem of view-based query answering has been addressed in several fields different from data integration, in general authors refer to view extensions rather than source extensions. In the following we will indifferently make use of both these terms.

Unfortunately, many of the papers concerning query processing do not distinguish between query answering and query rewriting using views, and give rise 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, there are algorithms, like the bucket or the inverse rules algorithm, for computing the best possible rewriting as union of conjunctive queries. Therefore the best possible rewriting (independently of the query language) is basically expressible in the same language as the original query and views. However for other query languages this is not the case. So, in spite of the large amount of work on the subject, the relationship between query rewriting and query answering using views is not completely clarified yet.

According to [25, 26], to focus on this relationship we have to abstract from the language used to express the rewriting, thus generalizing the notion of rewriting considered in the literature. [25, 26] define a *rewriting* of a query Q with respect to a set $\mathcal{V}$ of views as a function that, given the extensions of the views, i.e., a legal source database $\mathcal{D}$, returns a set of tuples that are contained in the certain answers of the query Q w.r.t. $\mathcal{V}$ and $\mathcal{D}$, i.e., in the answer set of Q for every global database coherent with the views and $\mathcal{D}$. The rewriting that returns precisely the set of certain answers for each legal source database is called the *perfect* rewriting of the query w.r.t. the views.

Observe that, by evaluating the perfect rewriting over given view extensions, one obtains the same set of tuples provided by query answering using views. Hence, the perfect rewriting is the best rewriting that 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[6].

Typically, one is interested in queries that can be evaluated in PTIME (i.e., are PTIME functions in data complexity), and hence one 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]. In general this is not the case. The complexity of answering queries using views for different languages (both for the query and for the views) is studied in [1]. The authors deal with the two assumptions that all the views are sound (called *open world assumption* in the paper), or that all the views are exact (called *closed world assumption* in the paper). Under open world assumption they show that in the case where the views are conjunctive queries, the problem becomes coNP-hard already in the case where the query is a conjunctive query with inequality. Instead, assuming queries be conjunctive queries, the problem is already coNP-hard when the views are positive queries. As shown in [25, 26], the problem is already coNP-hard for very simple query languages containing union. It is also shown that characterizing which instances of query rewriting admit a perfect rewriting that is PTIME, is a very difficult problem due to its connection with a longstanding open problem in constraint satisfaction [65, 45]. Under the different assumption that the views are exact (closed world assumption) the problem is coNP already in the case that views and queries are conjunctive queries [1]. Note that in the above reported results about query rewriting, we implicitly considered the open world assumption. Finally, in [1] is sketched an effective way to compute the certain answers by representing the global database by means of conditional tables and querying them using the techniques for databases with incomplete information [60]. Table 1 summarizes the results presented in [1].

In [22], 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 [3, 14, 46]. 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 [40].

Table 2 summarizes the results presented in [22]. Both data complexity, and expression complexity (complexity with respect to the size of the query and the view definitions) are

[6]According to [99] the data complexity is the complexity of the problem with respect to the size of the view extensions only, i.e., data at the sources, whereas query complexity is defined with respect to the size of the views and the query.

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

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 [13, 14, 83]). Indeed, the results in [22] have been extended to query language with the inverse operator [24], and to the class of union of conjunctive regular-path queries in [27].

4.2 GAV approach

When the mapping is specified following the GAV approach, each relation of the global schema is expressed in terms of a view over the sources. Hence, it is in general assumed that, in order to answer a query posed over the global schema, it is sufficient to *unfold*

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

each atom of the original query with the corresponding view [96]. This is illustrated in the following example.

Example 4.3 Consider the same sources and the same global schema as in Example 4.1, but suppose to have the following GAV mapping

$$\begin{array}{rcl}\langle \mathsf{movie}, \mathsf{movie}(T, Y, D) & \leftarrow & \mathsf{r}_1(T, Y, D)\rangle \\ \langle \mathsf{european}, \mathsf{european}(D) & \leftarrow & \mathsf{r}_1(T, Y, D)\rangle \\ \langle \mathsf{review}, \mathsf{review}(T, R) & \leftarrow & \mathsf{r}_2(T, R)\rangle\end{array}$$

In such a situation the query

$$\mathsf{q}(T, R) \quad \leftarrow \quad \mathsf{movie}(T, 1998, D) \land \mathsf{review}(T, R)$$

asking for reviews of movies filmed in 1998, is processed by expanding the atoms according to their definitions. Such an expansion computes the same reformulation as the one in Example 4.1

$$\mathsf{q}(T, R) \quad \leftarrow \quad \mathsf{r}_1(T, 1998, D) \land \mathsf{r}_2(T, R)$$

■

It is worthwhile to stress that, even if the process of unfolding is a simple mechanism, defining the views associated to the global elements implies precisely understanding the relationships among the sources, that is in general a non-trivial task. We can say that, in the GAV approach, query processing by unfolding is realized at design time, and the main issue turns out to be the definition of *wrappers* and *mediators* [101]. As already mentioned, wrappers are software components that present data at the sources in a suitable form adopted in the integration system, and handle the access to the data taking into account source capabilities to answer queries posed to the sources [81, 103]. Mediators are modules that establish how to access and merge data residing at the sources, i.e., synthesize the view definitions associated to the global relations.

In most GAV integration systems the notion of global schema is actually missing, in the sense that the global schema is simply the collection of relations exported by the mediators and no rules or integrity constraints can actually be expressed over the global concepts.

Hence, mediators simply realize an explicit layer between the users' application and the data sources. Furthermore, even when systems provide a global schema, they consider only the assumption that the views over source schemas are exact (closed world assumption). In such a situation exactly one global legal database exists (see Section 3), i.e., the retrieved global database that is obtained by populating the global schema according to the view definitions, and evaluating the query over this unique database is equivalent to evaluating its unfolding over the sources.

In the rest of this section, we first describe some integration systems that follow the GAV approach in the simplified setting we have described above, then we turn our attention to the approaches that deal with GAV integration in more complex scenarios.

4.2.1 Methods and systems based on unfolding

We now describe the main integration systems following, the GAV approach, which are essentially based on query unfolding.

The TSIMMIS Project TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) is a joint project of the Stanford University and the Almaden IBM database research group [34]. It is based on an architecture that presents a hierarchy of wrappers and mediators, in which wrappers convert data from each source into a common data model called OEM (Object Exchange Model) and mediators combine and integrate data exported by wrappers or by other mediators. Hence, the global schema $\mathcal{G}$ is essentially constituted by the set of OEM objects exported by wrappers and mediators. Mediators are defined in terms of a logical language called MSL (Mediator Specification Language), which is essentially Datalog extended to support OEM objects. OEM is a semistructured and self-describing data model, in which each object has an associated label, a type for the value of the object and a value (or a set of values). User queries are posed in terms of objects synthesized at a mediator or directly exported by a wrapper. They are expressed in MSL or in a specific query language called LOREL (Lightweight Object REpository Language), an object-oriented extension of SQL.

Each query is processed by a module, the Mediator Specification Interpreter (MSI) [87, 103], consisting of three main components:

- The *View Expander*, which uses mediator specification to reformulate the query into a *logical plan* by expanding the objects exported by the mediator according to their definitions. The logical plan is a set of MSL rules which refer to information at the sources.
- The *Plan Generator*, also called *Cost-Based Optimizer*, which develops a *physical plan* specifying which queries will be sent to the sources, the order in which they will be processed, and how the results of the queries will be combined in order to derive the answer to the original query.
- The *Execution engine*, which executes the physical plan and produces the answer.

The problem of query processing in TSIMMIS in the presence of limitations in accessing the sources is addressed in [81] by devising a more complex Plan Generator comprising three modules:

- a *matcher*, which retrieves queries that can process part of the logical plan;
- a *sequencer*, which pieces together the selected source queries in order to construct feasible plans;
- an *optimizer*, which selects the most efficient feasible plan.

It has to be stressed that in TSIMMIS no global integration is ever performed. Each mediator performs integration independently. As a result, for example, a certain concept may be seen in completely different and even inconsistent ways by different mediators. This form of integration can be called *query-based*, since each mediator supports a certain set of queries, i.e., those related to the view it provides.

The Garlic Project The Garlic project [28], developed at IBM's Almaden Research Center, provides the user with an integrated data perspective by means of an architecture comprising a middleware layer for query processing and data access software called *Query Services & RunTime System*. The middleware layer presents an object-oriented data model based on the ODMG standard [29] that allows data from various information sources to be represented uniformly. In such a case the global schema is simply constituted by the union of local schemas, and no integrity constraints are defined over the OMDG objects. The objects are exported by the wrappers using the Garlic Data Language (GDL), which is based on the standard Object Definition Language (ODL). Each wrapper describes data at a certain source in the OMDG format and gives descriptions of source capabilities to answer queries in terms of the query plans it supports. Note that the notion of mediator, central in the TSIMMIS system, is missing in Garlic, and most of the mediator tasks, as the integration of objects from different sources, are submitted to the wrappers. Users pose queries in terms of the objects of the global schema in an object-oriented query language which is an object-extended dialect of SQL. The *Query Services & RunTime System* produces a query execution plan in which a user's query is decomposed in a set of sub-queries to the sources, by expanding each involved object with its definition provided by the relative wrapper. The query execution plan is obtained by means of some sequence of parsing, semantic checking, query reformulation and optimization, based on the information (provided by a system catalog) about where data is stored, what wrapper it is associated with, its schema, any available statistics, etc. Each involved wrapper translates the sub-queries into the source's native query language taking into account the query processing power of the source. The description of the source's query capabilities is used by the query optimizer to create a set of feasible plans and to select the best for execution.

The Squirrel Project In the approaches to data integration described so far, the data at the sources are not replicated in the integration system (virtual approach). In contrast, in the materialized approach, the system computes the extension of the concepts in the global schema and maintains it in a local repository. Maintenance of replicated data against updates to the sources is a central aspect in this context and the effectiveness of maintenance affects timeliness and availability of data.

The *Squirrel* System, developed at the University of Colorado [107, 106, 105, 59], provides a GAV framework for data integration based on the notion of *integration mediator*. Integration mediators are active modules that support data integration using a hybrid of virtual and materialized data approaches.

The initial work on Squirrel [107, 105] does not consider virtual views and studies the problems related to data materialization. A key feature of Squirrel mediators, which consist of software components implementing materialized integrated views over multiple sources, is their ability to incrementally maintain the integrated views by relying on the active capabilities of sources. More precisely, at startup the mediator sends to the source databases a specification of the incremental update information needed to maintain the views and expects sources to actively provide such information.

Integration by means of hybrid virtual or materialized views is addressed in [59] in the context of the relational model. However, the presented techniques can also be applied in the context of object-oriented database models. In the sequel we refer to classes and objects to indicate data exported by mediators. The architecture of a Squirrel mediator described in [59] consists of components that deal with the problem of refreshing the materialized portion of the supported view, namely the update queue and the Incremental Update Processor (IUP), and components related to the problem of query processing, namely the Query Processor (QP) and the Virtual Attribute Processor (VAP). The QP provides the interface of the integrated view to the users. When it receives a user's query, it tries to answer the query on the basis of the materialized portion of the view, maintained in a local store. If the QP needs virtual data to answer the query, the VAP constructs temporary relations containing the relevant data. To obtain temporary relations the VAP uses information provided by a Virtual Decomposition Plan (VDP) maintained in the local store. The notion of VDP is analogous to that of Query Decomposition Plan in query optimization. More specifically, the VDP specifies the classes that the mediator maintains (materialized, virtual, or hybrid), and provides the basic structure for retrieving data from the sources or supporting incremental maintenance[7].

Either for the materialized version of the mediator or for the hybrid one, an automatic generator of Squirrel integrators has been developed. Such a module takes as input a specification of the mediator expressed in a high-level Integration Specification Language (ISL). A specification in ISL includes a description of the relevant subschemas of the source databases, and the match criteria between objects of families of corresponding classes, in particular a list of the classes that are matched and a binary matching predicate specifying correspondences between objects of two classes. The output of this module is an implementation of the mediator in the *Heraclitus* language, a database programming language whose main feature is the ability of representing and manipulating collections of updates to the current database state (deltas). An object-oriented extension of Heraclitus, called *H20*, is used in [106].

The problem of object matching in Squirrel mediators is addressed in [107]. In partic-

[7]For a description of the update process and the relative update queue and IUP see [59].

ular, a framework is presented for supporting intricate object identifier (OID) match criteria, such as key-based matching, lookup-table-based matching, historical-based-matching, and comparison-based matching. The last criterion allows for both considering attributes other than keys in object matching, and using arbitrary boolean functions in the specification of object matching.

The MOMIS Project The MOMIS system [9, 10], jointly developed at the University of Milano and the University of Modena and Reggio Emilia, provides semi-automatic techniques for the extraction and the representation of properties holding in a single source schema (*intraschema relationships*), or between different source schemas (*interschema relationships*), and for schema clustering and integration, to identify candidates to integration and synthesize candidates into an integrated global schema. The relationships are both intensional (e.g., lexical relationships extracted according to a Wordnet supplied ontology) and extensional (e.g., lexical relationships which strengthen the corresponding intensional relationships), either defined by the designer or automatically inferred by the system. The integration process is based on a source independent object-oriented model called ODM_{I^3}, used for modeling structured and semistructured data sources in a common way. The model is described by means of the ODL_{I^3} language. In MOMIS mediators are composed of two modules:

1. the *Global Schema Builder*, which constructs the global schema by integrating the ODL_{I^3} source descriptions provided by the wrappers, and by exploiting the intraschema and the interschema relationships.

2. the *Query Manager*, which performs query processing and optimization. The Query Manager exploits extensional relationships to first identify all sources whose data are needed to answer a user's query posed over the global schema. Then it reformulates the original query into queries to the single sources, sends the obtained sub-queries to the wrappers, which execute them and report the results to the Query Manager. Finally, the Query Manager combines the single results to provide the answer to the original query.

Currently, the query processing aspect is in a preliminary stage of development, and still needs further investigation.

The DIKE Project The DIKE system [85], developed at the University of Calabria in collaboration with the University of Reggio Calabria, exploits a conceptual model called SDR-Network [86, 93] in order to uniformly handle and represent heterogeneous data sources. Each source has an associated SDR-Network, constituted by a set of nodes representing concepts and a set of arcs representing relationships between pairs of concepts. DIKE provides the user with an algorithm for data source integration which takes as input two SDR-Networks, which are either associated directly to the sources or are derived from previously constructed SDR-Networks, and computes an output *global* SDR-Network. The process is carried out by exploiting synonymies, and homonymies between arcs and nodes of the SDR-Networks, and similarity relationships between sub-nets. To each derived node (resp., arc) a mapping is associated that describes the way the node has been obtained from one or more nodes (resp., arcs) belonging to the original SDR-Networks. Finally, to each mapping between nodes a view is associated that allows to obtain the instances of the target node of the mapping, from the instances of the source nodes of the mapping. At the moment the system is under development and does not present a completely defined method to answer queries posed over the global SDR-Networks. To answer a user's query it simply uses the views associated to the global nodes involved in the query to retrieve its instances from the nodes stored at the sources.

4.2.2 Other studies and results: beyond unfolding

As we already said at the beginning of this section, all the systems described above assume an exact semantics for the views and do not allow for the specification of constraints on the global schema. With respect to the general integration framework presented in Section 3, such assumptions greatly simplify the problem of computing the answer to a query (query processing reduces to unfolding). At the same time, however, they make the GAV integration systems unable to properly take into account actual constraints imposed by the application domain. On the other hand, the importance of allowing integrity constraints on the global schema has been stressed in several works on data integration [20, 47, 46]. Furthermore, different assumptions on the views, i.e., sound or complete, can be adopted, thus bringing about a situation in which the semantics of an integration system is best described in terms of a set of databases rather than a single one. Hence, unfolding is in general not sufficient to process a query in GAV, and it is necessary to make use of reasoning mechanisms. A technique to effectively process queries is described in [16], in the case when both global schema and sources are relational, key and foreign key constraints are expressed in the global schema, views are assumed to be sound, and queries over the global schema are union of conjunctive queries. In that paper the authors show that, starting from the retrieved global database, it is possible to build a *canonical* database that has the property of faithfully representing all the databases that satisfy the global schema, in the sense that the set of certain answers to a given query can be obtained by evaluating the query over it. Furthermore, an algorithm is given that makes it possible to find the answers to a query over the given canonical database without actually building it. The algorithm constructs a suitable reformulation of the query, called *query expansion*, such that the answer to the expansion computed over the retrieved global database is equal to the answer to the original query over the canonical database. In [15], the same technique is used to deal with the mandatory participation and functional attribute constraints imposed by a conceptual model adopted to represent the global schema. However, notice that in these papers it is assumed that data at the sources are such that the global retrieved database does not violate the key constraints on the global schema, while foreign key constraints may not be satisfied.

The algorithms described above have been implemented in the IBIS system, which is described in the following.

The IBIS system The Internet-Based Information System (IBIS) [17] is a tool for the semantic integration of heterogeneous data sources, developed in the context of a collaboration between the University of Rome "La Sapienza" and CM Sistemi. IBIS adopts innovative solutions to deal with all aspects of a complex data integration environment, including source wrapping, limitations on source access, and query answering under integrity constraints. IBIS uses a relational global schema to query the data at the sources, and is able to cope with a variety of heterogeneous data sources, including data sources on the Web, relational databases, and legacy sources. Each non-relational source is wrapped to provide a relational view on it. Also, each source is considered sound. The system allows for the specification of integrity constraints on the global schema; in addition, IBIS considers the presence of some forms of constraints on the source schema, in order to perform runtime optimization during data extraction. In particular, key and foreign key constraints can be specified on the global schema, and functional dependencies and full-width inclusion dependencies, i.e., inclusions between entire relations, can be specified on the source schema.

Query processing in IBIS is separated in three phases:

1. the query is expanded to take into account the integrity constraints in the global schema;

2. the atoms in the expanded query are unfolded according to their definition in terms of the mapping, obtaining a query expressed over the sources;

3. the expanded and unfolded query is executed over the retrieved source databases (see below), to produce the answer to the original query.

Query unfolding and execution are the standard steps of query processing in GAV data integration systems, while for the expansion phase IBIS makes use of the algorithm presented in [16] that we have described above. As we already said, the expanded query has to be evaluated over the retrieved global database in order to produce the certain answers to the original query. As the construction of the retrieved global database is computationally costly, the IBIS Expander module does not construct it explicitly. Instead, it unfolds the expanded query and evaluates the unfolded query over the *retrieved source database*, whose data are extracted by the Extractor module that retrieves from the sources all the tuples that may be used to answer the original query.

Other studies deal with the problem of query processing in GAV in the presence of limitations on how sources can be accessed [80]. To answer queries over such sources one generally needs to start from a set of constants (provided e.g., by the user filling in a form, or taken from a source without access limitations) to bind attributes. Such bindings are used to access sources and there obtain new constants which in turn can be used for new accesses. Hence, query processing in GAV in the presence of access limitations in general requires the evaluation of a recursive query plan. It is worth noticing that also in this case unfolding is not sufficient to answer the query. Since source accesses are costly, an important issue is how to minimize the number of accesses to the sources while still being guaranteed to obtain all possible answers to a query. [79, 80] discuss several optimizations that can be made at compile time, during query plan generation.

5 Dealing with inconsistent data sources

In a data integration context, the data stored in autonomous and heterogeneous sources are considered as a whole, and they may result inconsistent due to the presence of integrity constraints expressed over the global schema. Generally speaking, when data are considered locally, they must satisfy only the integrity constraints specified in the source to which they belong, but they are not required to satisfy the integrity constraints expressed in the global schema. Hence, when data sources are considered as part of an integration system, and a set of global requirements is expressed in the system, the data might present inconsistencies that have to be properly taken into account during query processing.

In this section, we first illustrate the problem of data inconsistency in the data integration framework, then present a survey on the state of the art in query processing in inconsistent databases. We limit our investigation to GAV, since the works in data integration that deal with inconsistent data implicitly adopt this approach to specify the mapping.

5.1 Inconsistency in data integration

As we already said, the global schema has to provide an appropriate abstraction of the integration domain, and the language adopted to specify it has to be powerful enough to represent a unified and reconciled view of heterogeneous and overlapping sources, such as the ones based on semantic integrity constraints. Integrity constraints expressed over the global schema represent fundamental knowledge about the real world that cannot be neglected. However, the autonomy and heterogeneity of the sources bring often about a situation in which global databases are inconsistent with respect to the constraints on the global schema. On the other hand, since most of the data could satisfy such constraints, it seems unreasonable to consider the entire system inconsistent and unable to provide answers to queries.

Example 5.1 Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, in which $\mathcal{G}$ comprises the two global relations $\mathsf{author}(\mathit{Name}, \mathit{Painting})$ and $\mathsf{located}(\mathit{Work}, \mathit{Museum})$. Consider the foreign key constraint on $\mathcal{G}$ stating that every *Painting* has to be located in a museum, i.e., has to be a *Work* in the relation *located*. The mapping is GAV, i.e., it comprises two views over the sources associated to author and located, respectively. Furthermore, such views are assumed to be exact. We consider to have an underlying infinite database domain Γ, comprising, among other symbols, $\{a, b, c, d\}$. By evaluating each view in the mapping over a given source database $\mathcal{D}$, we obtain the retrieved global database $ret(\mathcal{I}, \mathcal{D}) = \{\mathsf{author}(a, b), \mathsf{author}(c, d), \mathsf{located}(b, c)\}$. It is easy to see that $ret(\mathcal{I}, \mathcal{D})$ does not satisfy the global schema, since there does not exist a tuple in located having d as first component. Consequently, since $ret(\mathcal{I}, \mathcal{D})$ is not coherent with $\mathcal{G}$ and the views are all exact, no global legal database exists according to the semantics of a data integration system. This in turn implies that query answering is meaningless, since every possible query is a logical consequence of $\mathcal{I}$ and $\mathcal{D}$: for instance, the answer to the query that asks for all works located in some museum, i.e., $q(x) \leftarrow \mathsf{located}(x, y)$, is the whole interpretation domain Γ (that is, every possible object belongs to the extension of the query).

On the other hand, $\mathsf{author}(a, b)$ and $\mathsf{located}(b, c)$ are perfectly coherent with the constraint in the global schema: therefore, the answer to the previous query with respect to the "consistent part" of $ret(\mathcal{I}, \mathcal{D})$ is $\{b\}$. ■

Assume now to work in the presence of sound and complete views. We recall that, in the GAV approach, a view is considered *sound* if it computes only a subset of the data that satisfy the corresponding global relation. For example, this assumption can refer to a situation in which the sources queried by such a view are assumed to be reliable, but not complete. i.e., they provide "nothing but the truth, but not all the truth".

Conversely, a view is considered *complete* if it provides a superset of the data that actually satisfy the global schema. For example, this assumption can be done in the case in which the sources queried by such a view store all the data that satisfy the corresponding global relation, but such data are not considered reliable, i.e., sources provide "all, but not only, the truth".

The example below shows that, when views in the mapping are complete or sound it is possible, in some cases, to overcome the inconsistencies that might arise in the global schema and satisfy the integrity constraints coherently with the mapping assumptions.

Example 5.2 Consider the database instance of Example 5.1 and assume now that the view associated to author is complete and the the view associated to $\mathsf{located}$ is exact. According to such an assumption, it is possible to make $ret(\mathcal{I}, \mathcal{D})$ satisfy the foreign key constraint by simply deleting $\mathsf{author}(c, d)\}$ from it.

On the contrary, if the view associated to $\mathsf{located}$ is sound and the other view is exact, it is possible to add $\mathsf{located}(d, \alpha)\}$ to $ret(\mathcal{I}, \mathcal{D})$ in order to satisfy the foreign key constraint, where α is any constant of the infinite domain Γ.

It is interesting to note that the answers to the query $q(x) \leftarrow \mathsf{located}(x, y)$ are b and $\{b, d\}$ in the two cases, respectively. ■

Even though in the example above it is possible to resolve inconsistencies, other situations may happen in which one is not allowed to add or delete tuple from the data retrieved at the sources, even if the assumptions on the views are sound or complete. Consider for example the different situation in which the view associated to author is sound and the view associated to $\mathsf{located}$ is complete. As in the case in which both the views were considered exact, in this case it is not possible to satisfy the foreign key constraint on the global schema. Indeed, the foreign key constraint can be satisfied either by deleting tuples to author or by adding tuples to $\mathsf{located}$, but the soundness of the former relation, and the completeness of the latter, do not allow us to perform such "repairs".

Hence, classical assumptions on the views do not allow us to properly handle data inconsistency, and generally they bring about a situation where no global legal database exists for the integration system, if data coming from the sources are incoherent with the integrity constraints specified on it.

A more general approach would be to adopt different semantics able to provide global legal databases even in the presence of inconsistencies, and define techniques to compute answers to queries even in such a situation.

The semantic problem that arises in this context is similar to the one underlying the notion of *database repair* introduced by several works in the area of *inconsistent databases*. Such works propose techniques aiming to both repair databases, i.e, obtaining consistent database instances from inconsistent ones, and compute *consistent query answers* from inconsistent databases. Roughly, a database repair is every database obtained by suitably adding and/or deleting tuples from/to an inconsistent database $\mathcal{B}$, in order to make data satisfy the integrity constraints originally violated by $\mathcal{B}$. The repairing policy always relies on a semantics that allows for addition and deletion of tuples. Several semantics have been proposed in the literature. Intuitively, each such semantics is based on an ordering defined between the databases obtained from the original inconsistent database. The notion of consistent query answer can be formally described as follows: given an inconsistent database $\mathcal{B}$, a tuple t is a consistent answer to a query q if $t \in q^{\mathcal{B}_r}$ for every B_r repair of $\mathcal{B}$.

It is important to point out that such studies basically apply to a single database setting [44, 12, 6, 7], and that almost all the proposed techniques can be employed in a data integration setting only by assuming an "exact" interpretation of the mapping [82, 38, 53].

On the other hand, no proposal explicitly deals with the problem of query answering in a data integration system with inconsistent data sources: actually, the current integration methodologies deal with inconsistency in a previous phase, called *data reconciliation* (or *data cleaning*), in which all inconsistencies are "cleaned", by ad-hoc updating algorithms. Therefore, query answering is performed when data are consistent.

In the following, we review the main proposals in the field of inconsistent databases.

5.2 State of the art

The notion of consistent query answers was originally given in [12]. However, the approach in [12] is completely proof-theoretic, and no computational technique for obtaining consistent answers from inconsistent database is provided.

In [82] the authors describe an operator for *merging databases* under constraints which allows for obtaining a maximal amount of information from each database by means of a majority criterion used in case of conflict. Even if a large set of constraints is considered, namely the constraints that can be expressed as first-order formulas, the computational complexity of the merging procedure is not explored, and no algorithm to compute consistent query answers is provided. Furthermore, the problem of dealing with incomplete databases is not taken into account. Notice also that, different from all the other studies mentioned in the following, this approach relies on a cardinality-based ordering between databases (rather than a set-containment-based ordering).

In [44] the authors propose a framework for updating theories and logical databases (i.e., databases obtained by giving priorities to sentences in the databases) that can be extended also to the case of updating views. The semantics proposed in such a paper is based on a particular set-containment based ordering between theories that "accomplish" an update to an original theory T. A theory T_1 accomplishes an insertion of σ into T if $\sigma \in T_1$, and accomplishes a deletion of σ if σ is not a logical consequence of T_1. Then, a theory T_1 accomplishes an update u to T with a *smaller change than* T_2, and thus is preferred to T_2, if both T_1 and T_2 accomplish u, but T_1 has fewer deletions than T_2, or they have the same deletions but T_1 has fewer insertions than T_2. The basic idea is to maintain as many as possible of the facts that are known to be true. In a data integration context, this means to satisfy constraints (that can be conceived as updates of the global schema) by preferably adding tuples to the retrieved database, but at the same time by adding as few tuples as possible.

In [6] the authors define an algorithm for consistent query answers in inconsistent databases based on the notion of *residues*, originally defined in the context of semantic query optimization. The method is proved to be sound and complete only for the class of universally quantified binary constraints, i.e., constraints that involve two database relations. In [7] the same authors propose a new method that can handle arbitrary universally quantified constraints by specifying the database repairs into *logic rules with exceptions* (LPe). The semantics underlying the notion of consistent query answers both in [6] and in [7] is defined on a set-containment ordering between databases. The basic idea is to consider as repairs all those consistent databases whose symmetric difference with the original inconsistent database is minimal with respect to containment. Intuitively, in a data integration context this means to satisfy integrity constraints by both adding and/or deleting tuples from the retrieved global database and consider as legal all the obtained databases that are "as close as possible" to the retrieved database.

Finally, [53] proposes a technique to deal with inconsistencies that is based on the reformulation of integrity constraints into a disjunctive Datalog program with two different forms of negation: negation as failure and classical negation. Such a program can be used both to repair databases, i.e., modify the data in the databases in order to satisfy integrity constraints, and to compute consistent query answers. The technique is proved to be sound and complete for universally quantified constraints. The semantics adopted to support this method is similar to the one in [6].

It is worth noticing that none of the above works provides solutions for the case of existentially quantified constraints. Moreover, the studies concerning data inconsistencies basically apply to a single database setting. A first attempt to overcome such limitations is proposed in [67], which presents both a general framework for data integration under inconsistent and incomplete data sources when views are considered sound, and an algorithm for consistent query answering in the presence of both key constraints and foreign key constraints in the global schema.

With regard to the works that deal with inconsistencies in database integration, the activity of integrating data is commonly conceived as a two-phase process, where the first one aims at populating the global schema, and the second one is intended to repair possible inconsistencies among data. Hence, traditional approaches implicitly interpret the mapping as an exact one. In this sense, a more general approach is needed in data integration, based on the semantics of the mapping.

Finally, we point out that the problem of reasoning with inconsistent databases is also closely related to the studies in *belief revision and update*. This area of Artificial Intelligence studies the problem of integrating new information with previous knowledge. In general, the problem is studied in a logical framework, in which the new information is a logical formula f and the previous knowledge is a logical theory (also called knowledge base) T. Of course, f may in general be inconsistent with T. The *revised* (or *updated*) knowledge base is denoted as $T \circ f$, and several semantics have been proposed for the operator $\circ$.

The semantics for belief revision can be divided into:

- *revision* semantics, when the new information f is interpreted as a modification of the knowledge about the world;
- *update* semantics, when f reflects a change in the world.

Moreover, the different semantics proposed can be either *syntax-based* or *model-based* semantics.

The problem of reasoning with inconsistent databases can be actually seen as a problem of belief revision. In fact, with respect to the above illustrated knowledge base revision framework, if we consider the global retrieved database $ret(\mathcal{I}, \mathcal{D})$ (where $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ and $\mathcal{D}$ is the source database) as the initial knowledge base T, and the global schema $\mathcal{G}$ as the new information f, then the problem of computing the answer to a query q with respect to the integration system $\mathcal{I}$ and the source database $\mathcal{D}$ corresponds to the belief revision problem $ret(\mathcal{I}, \mathcal{D}) \circ \mathcal{G} \models q$.

Based on such a correspondence, the studies in belief revision appear very relevant for the field of inconsistent databases, although the kind of knowledge considered in belief revision is generally different from the one involved in database applications. For instance, [42, 43] present several complexity results for reasoning with revised/updated knowledge bases, where the knowledge is specified in terms of propositional formulas of classical logic. However, a true survey of the state of the art in belief revision is outside the scope of the present document.

Finally, with regard the above sketched classification of semantics for belief revision, it is important to point out that all the approaches to inconsistent databases that we have considered in this section adopt a revision semantics, rather than an update semantics.

6 Conclusions

In this document we have described a general framework for data integration systems, which provide access to a set of heterogeneous sources by means of a global schema in terms of which user queries are posed. The source schemas are related to the global schema by means of a mapping. The problem of query processing in such a setting amounts to finding a way to answer a query posed over the global schema using only the information stored at the sources. We have discussed query processing in the context of the two relevant cases where the mapping between the global and the local schemas is specified following either the local-as-view (LAV) or the global-as-view (GAV) approach. While in the LAV approach each source relation is expressed in terms of a view over the global schema, in the GAV approach each global relation is associated with a view over the source relations. Since the mapping differs in the two frameworks, the problem assumes different aspects in LAV and GAV.

In the former approach query processing has been addressed by means of either *query rewriting*, which tries to reformulate the original query in terms of the sources and then evaluates the rewritten query over the data at the sources, or *query answering*, which aims at directly answering the query based on the view extensions. We have presented the algorithms proposed in the literature that deal with query rewriting and query answering in the LAV approach.

Integration in the GAV approach is treated in the literature only in the simplified setting in which no constraint is defined in the global schema. In such a setting query processing in the GAV approach reduces to simply *unfolding* the global relations in the query with their definitions. We have described some GAV integration systems that deal with the data integration problem in such a setting. We showed also that in the presence of constraints over the global schema unfolding is in general not sufficient.

Finally, we have surveyed the main approaches that deal with the problem of query answering in the presence of data inconsistent with respect to the constraints in the global schema. In this respect, it is worth noticing that none of the approach to query answering in the presence of inconsistent data provides solutions for the case of existentially quantified constraints, which seems a crucial aspect in a data integration setting. Moreover, the studies concerning data inconsistencies basically apply to a single database setting, while data integration applications handle multiple, heterogeneous data sources.

With regard to the INFOMIX project, we recall that the main purpose of INFOMIX is to provide an advanced system for information integration, which is innovative in terms of reasoning capabilities and improves the state-of-the-art ability to handling incomplete and inconsistent data sources.

The analysis of systems and models for data integration presented in this document indicates that all the aspects of a data integration system are relevant to this goal. More specifically, the following aspects play a crucial role:

- the formalism for expressing source schemas
- the formalism for expressing the global schema and the constraints in the global schema
- the form of the mapping between the sources and the global schema (LAV, GAV, GLAV)
- the characterization of the mapping in terms of soundness, completeness, exactness of the views
- the view language, i.e., the formalism for expressing the mapping (conjunctive queries, UCQ, Datalog queries, ...)
- the semantics for dealing with data inconsistent with respect to the constraints in the global schema
- the query language, i.e., the formalism for expressing the user query

The above mentioned characteristics are particularly relevant with respect to one of the main project subgoal: defining a formalism which allows both for expressing, in a declarative way, the knowledge about the information sources to be integrated, and for specifying the global information schema, while balancing the trade-off between such expressiveness requirements and the issue of providing efficient information integration algorithms.

References

[1] Serge Abiteboul and Oliver 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] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[3] Serge Abiteboul, Dallan Quass, Jason McHugh, Jennifer Widom, and Janet L. Wiener. The Lorel query language for semistructured data. *Int. J. on Digital Libraries*, 1(1):68–88, 1997.

[4] Foto N. Afrati, Manolis Gergatsoulis, and Theodoros 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] Marcelo Arenas, Leopoldo E. Bertossi, and Jan 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] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Specifying and querying database repairs using logic programs with exceptions. In *Proc. of the 4th Int. Conf. on Flexible Query Answering Systems (FQAS 2000)*, pages 27–41. Springer, 2000.

[8] Catriel Beeri, Alon Y. Levy, and Marie-Christine 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.

[9] 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.

[10] Sonia Bergamaschi, Silvana Castano, Maurizio Vincini, and Domenico Beneventano. Semantic integration of heterogeneous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.

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

[12] François 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.

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

[14] Peter Buneman, Susan Davidson, Gerd Hillebrand, and Dan 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.

[15] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of the 20th Int. Conf. on Conceptual Modeling (ER 2001)*, pages 270–284, 2001.

[16] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, volume 2348 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2002.

[17] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Paolo Naggar, and Fabio Vernacotola. IBIS: Data integration at work. In *Proc. of the 10th Ital. Conf. on Database Systems (SEBD 2002)*, pages 291–298, 2002.

[18] Diego Calvanese, Giuseppe De Giacomo, and Maurizio 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.

[19] Diego Calvanese, Giuseppe De Giacomo, and Maurizio 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.

[20] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10(3):237–271, 2001.

[21] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe 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.

[22] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe 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.

[23] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe 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.

[24] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe 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.

[25] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe 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.

[26] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe Y. Vardi. What is query rewriting? In *Proc. of the 7th Int. Workshop on Knowledge Representation meets Databases (KRDB 2000)*, pages 17–27. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-29/`, 2000.

[27] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Moshe 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.

[28] 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.

[29] Roderick G. G. Cattell and Douglas K. Barry, editors. *The Object Database Standard: ODMG 2.0.* Morgan Kaufmann, Los Altos, 1997.

[30] Edward 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.

[31] Ashok K. Chandra and Philip 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.

[32] 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.

[33] Surajit Chaudhuri and Moshe 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.

[34] Sudarshan S. Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of the 10th Meeting of the Information Processing Society of Japan (IPSJ'94)*, pages 7–18, 1994.

[35] Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, pages 56–70, 1997.

[36] Sara Cohen, Werner Nutt, and Alexander 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.

[37] Guozhu Dong and Jianwen Su. Conjunctive query containment with respect to views and constraints. *Information Processing Lett.*, 57(2):95–102, 1996.

[38] Phan Minh Dung. Integrating data from possibly inconsistent databases. In *Proc. of the 4th Int. Conf. on Cooperative Information Systems (CoopIS'96)*, pages 58–65, 1996.

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

[40] Oliver M. Duschka and Michael 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.

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

[42] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.

[43] T. Eiter and G. Gottlob. The complexity of nested counterfactuals and iterated knowledge base revisions. *J. of Computer and System Sciences*, 53(3):497–512, 1996.

[44] Ronald Fagin, Jeffrey D. Ullman, and Moshe Y. Vardi. On the semantics of updates in databases. In *Proc. of the 2nd ACM SIGACT SIGMOD Symp. on Principles of Database Systems (PODS'83)*, pages 352–365, 1983.

[45] Tomás Feder and Moshe Y. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction. *SIAM J. on Computing*, 28:57–104, 1999.

[46] Mary F. Fernandez, Daniela Florescu, Jaewoo Kang, Alon Y. Levy, and Dan 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.

[47] Mary F. Fernandez, Daniela Florescu, Alon Levy, and Dan Suciu. Verifying integrity constraints on web-sites. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 614–619, 1999.

[48] Daniela Florescu, Alon Levy, and Dan 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.

[49] Daniela Florescu, Alon Y. Levy, Ioana Manolescu, and Dan 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.

[50] Marc Friedman, Alon Levy, and Todd 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.

[51] Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. An extensible framework for data cleaning. Technical Report 3742, INRIA, Rocquencourt, 1999.

[52] Gösta Grahne and Alberto 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.

[53] Gianluigi Greco, Sergio Greco, and Ester 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.

[54] Stéphane Grumbach, Maurizio Rafanelli, and Leonardo 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.

[55] Jarek Gryz. An algorithm for query folding with functional dependencies. In *Proc. of the 7th Int. Symp. on Intelligent Information Systems*, pages 7–16, 1998.

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

[57] Himanshu Gupta, Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Index selection for OLAP. In *Proc. of the 13th IEEE Int. Conf. on Data Engineering (ICDE'97)*, pages 208–219, 1997.

[58] Venky Harinarayan, Anand Rajaraman, and Jeffrey D. Ullman. Implemeting data cubes efficiently. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 205–216, 1996.

[59] Richard Hull and Gang Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 481–492, 1996.

[60] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *J. of the ACM*, 31(4):761–791, 1984.

[61] Matthias Jarke, Maurizio Lenzerini, Yannis Vassiliou, and Panos Vassiliadis, editors. *Fundamentals of Data Warehouses.* Springer, 1999.

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

[63] Arthur M. Keller and Julie Basu. A predicate-based caching scheme for client-server database architectures. *Very Large Database J.*, 5(1):35–47, 1996.

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

[65] Phokion G. Kolaitis and Moshe Y. Vardi. Conjunctive-query containment and constraint satisfaction. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 205–213, 1998.

[66] Chung T. Kwok and Daniel Weld. Planning to gather information. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 32–39, 1996.

[67] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In *Proc. of the 9th Int. Workshop on Knowledge Representation meets Databases (KRDB 2002).* CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-54/`, 2002.

[68] Alon Levy, Richard E. Fikes, and Shuky Sagiv. Speeding up inference using relevance reasoning: A formalism and algorithms. *Artificial Intelligence*, 97(1–2), 1997.

[69] Alon 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.

[70] Alon Y. Levy. Answering queries using views: A survey. Technical report, University of Washinghton, 1999.

[71] Alon Y. Levy. Logic-based techniques in data integration. In Jack Minker, editor, *Logic Based Artificial Intelligence.* Kluwer Academic Publisher, 2000.

[72] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh 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.

[73] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Query answering algorithms for information agents. In *Proc. of the 13th Nat. Conf. on Artificial Intelligence (AAAI'96)*, pages 40–47, 1996.

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

[75] Alon Y. Levy, Anand Rajaraman, and Jeffrey 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.

[76] Alon Y. Levy and Marie-Christine 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.

[77] Alon Y. Levy, Divesh Srivastava, and Thomas Kirk. Data model and query evaluation in global information systems. *J. of Intelligent Information Systems*, 5:121–143, 1995.

[78] Alon Y. Levy and Dan 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.

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

[80] Chen Li and Edward 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.

[81] Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey D. Ullman, and Murty Valiveti. Capability based mediation in TSIMMIS. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 564–566, 1998.

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

[83] Tova Milo and Dan 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.

[84] Prasenjit Mitra. An algorithm for answering queries efficiently using views. Technical report, University of Southern California, Information Science Institute, Stanford (CA, USA), 1999. Available at `http://dbpubs.stanford.edu/pub/1999-46`.

[85] Luigi Palopoli, Giorgio Terracina, and Domenico Ursino. The system DIKE: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *Proc. of Symposium on Advances in Databases and Information Systems (ADBIS-DASFAA 2000)*, pages 108–117, 2000.

[86] Luigi Palopoli, Giorgio Terracina, and Domenico Ursino. A graph-based approach for extracting terminological properties of elements of XML documents. In *Proc. of the 17th IEEE Int. Conf. on Data Engineering (ICDE 2001)*, pages 330–340. IEEE Computer Society Press, 2001.

[87] Yannis Papakonstantinou, Hector Garcia-Molina, and Jeffrey D. Ullman. MedMaker: A mediation system based on declarative specifications. In Stanley Y. W. Su, editor, *Proc. of the 12th IEEE Int. Conf. on Data Engineering (ICDE'96)*, pages 132–141, 1996.

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

[89] Rachel Pottinger and Alon Y. Levy. A scalable algorithm for answering queries using views. In *Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, pages 484–495, 2000.

[90] Xiaolei Qian. Query folding. In *Proc. of the 12th IEEE Int. Conf. on Data Engineering (ICDE'96)*, pages 48–55, 1996.

[91] Anand Rajaraman, Yehoshua Sagiv, and Jeffrey 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.

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

[93] Giorgio Terracina and Domenico Ursino. Deriving synonymies and homonymies of object classes in semi-structured information sources. In *Proc. of International Conference on Management of Data (COMAD 2000)*, pages 21–32. McGraw-Hill, New York, 2000.

[94] Dimitri Theodoratos and Timos Sellis. Data warehouse design. In *Proc. of the 23rd Int. Conf. on Very Large Data Bases (VLDB'97)*, pages 126–135, 1997.

[95] 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.

[96] Jeffrey 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.

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

[98] Ron van der Meyden. Logical approaches to incomplete information. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998.

[99] Moshe Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM SIGACT Symp. on Theory of Computing (STOC'82)*, pages 137–146, 1982.

[100] Jennifer Widom (ed.). Special issue on materialized views and data warehousing. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2), 1995.

[101] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[102] Jian Yang, Kamalakar Karlapalem, and Qing Li. Algorithms for materialized view design in data warehousing environment. In *Proc. of the 23rd Int. Conf. on Very Large Data Bases (VLDB'97)*, pages 136–145, 1997.

[103] Ramana Yerneni, Chen Li, Hector Garcia-Molina, and Jeffrey D. Ullman. Computing capabilities of mediators. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 443–454, 1999.

[104] Ramana Yerneni, Chen Li, Jeffrey D. Ullman, and Hector Garcia-Molina. Optimizing large join queries in mediation systems. In *Proc. of the 7th Int. Conf. on Database Theory (ICDT'99)*, pages 348–364, 1999.

[105] Gang Zhou, Richard Hull, and Roger King. Generating data integration mediators that use materializations. *J. of Intelligent Information Systems*, 6:199–221, 1996.

[106] Gang Zhou, Richard Hull, Roger King, and Jean-Claude Franchitti. Data integration and warehousing using H20. *Bull. of the IEEE Computer Society Technical Committee on Data Engineering*, 18(2):29–40, 1995.

[107] Gang Zhou, Richard Hull, Roger King, and Jean-Claude 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.