Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

Information Systems 29 (2004) 147–163

www.elsevier.com/locate/infosys

Data integration under integrity constraints

Andrea Calì*, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy

Abstract

Data integration systems provide access to a set of heterogeneous, autonomous data sources through a so-called global schema. There are basically two approaches for designing a data integration system. In the global-as-view approach, one defines the elements of the global schema as views over the sources, whereas in the local-as-view approach, one characterizes the sources as views over the global schema. It is well known that processing queries in the latter approach is similar to query answering with incomplete information, and, therefore, is a complex task. On the other hand, it is a common opinion that query processing is much easier in the former approach. In this paper we show the surprising result that, when the global schema is expressed in the relational model with integrity constraints, even of simple types, the problem of incomplete information implicitly arises, making query processing difficult in the global-as-view approach as well. We then focus on global schemas with key and foreign key constraints, which represents a situation which is very common in practice, and we illustrate techniques for effectively answering queries posed to the data integration system in this case.

Keywords: Data integration; Integrity constraints; Global-as-view approach; Query answering

1. Introduction

Integrating heterogeneous data sources is a fundamental problem in databases, which has been studied extensively in the last two decades both from a formal and from a practical point of view [1–6]. Recently, mostly driven by the need to integrate data sources on the Web, much of the research on integration has focussed on so-called *data integration* [6–8]. Data integration is the problem of combining the data residing at different sources, and providing the user with a unified view of these data. Such a unified view is structured according to a so-called global schema, which represents the intensional level of the integrated and reconciled data, and provides the elements for expressing the queries over the data integration system. It follows that, in formulating the queries, the user is freed from the knowledge on where data are, how data are structured at the sources, and how data are to be merged and reconciled to fit into the global schema.

The interest in this kind of systems has been continuously growing in the last years. Many organizations face the problem of integrating data residing in several sources. Companies that build a

*Corresponding author. Tel.: +39-06-4991-8482; fax: +39-06-8530-0849.

E-mail addresses: cali@dis.uniroma1.it (A. Calì), calvanese@dis.uniroma1.it (D. Calvanese), degiacomo@dis.uniroma1.it (G. De Giacomo), lenzerini@dis.uniroma1.it (M. Lenzerini).

URLs: http://www.dis.uniroma1.it/~cali/, http://www.dis.uniroma1.it/~calvanese/, http://www.dis.uniroma1.it/~degiacomo/, http://www.dis.uniroma1.it/~lenzerini/.

0306-4379/03/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0306-4379(03)00050-4

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 somehow with data integration.

The design of a data integration system is a very complex task, which requires addressing several different issues. Here, we concentrate on two basic issues:

(1) specifying the mapping between the global schema and the sources,
(2) processing queries expressed on the global schema.

With regard to issue (1), two basic approaches have been used to specify the mapping between the sources and the global schema [7,9]. The first approach, called *global-as-view* [10–12], requires that the global schema is expressed in terms of the data sources. More precisely, to every element 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. In general, the views associated to the elements of the global schema are considered *sound*, i.e., all the data provided by a view satisfies the corresponding element of the global schema, but there may be additional data satisfying the element not provided by the view. The second approach, called *local-as-view* [13–15], requires the global schema to be specified independently from the sources. In turn, the sources are defined as views over the global schema. Comparisons of the two approaches are reported in [8,16]. In this paper, we study global-as-view data integration systems, and, according to the usual approach, we assume that the views associated to the elements of the global schema are sound.

Issue (2) is concerned with one of the most important problems in the design of a data integration system, namely, the choice of the method for computing the answer to queries posed in terms of the global schema. Since the global schema acts as the interface to the user for query formulation, it should mediate among different representations of overlapping worlds, and therefore the schema definition language should incorporate flexible and powerful representation mechanisms, such as the ones based on semantic integrity constraints. For the purpose of query answering, the system should be able to reformulate the query in terms of a suitable set of queries posed to the sources. These queries are then shipped to the sources, and the results are assembled into the final answer. It is well known that processing queries in the local-as-view approach is a difficult task [8,14,17–19]. Indeed, in this approach the only knowledge we have about the data in the global schema is through the views representing the sources, and such views provide only partial information about the data. Therefore, extracting information from the data integration system is similar to query answering with incomplete information, which is a complex task [20]. On the other hand, query processing is considered much easier in the global-as-view approach, where in general it is assumed that answering a query basically means unfolding its atoms according to their definitions in terms of the sources [7]. The reason why unfolding does the job is that the global-as-view mapping essentially specifies a single database satisfying the global schema, and evaluating the query over this unique database is equivalent to evaluating its unfolding over the sources.

While it is a common opinion in the literature that the simple technique based on unfolding is sufficient in the global-as-view approach, we show in this paper that the presence of integrity constraints in the global schema poses new challenges, specially related to the need of taking the semantics of constraints into account during query processing.

The importance of allowing integrity constraints in the global schema has been stressed in several works on data integration [15,21–25]. In [23] the problem of providing consistent answers in databases that violate integrity constraints is addressed; however, in this work only universally quantified constraints are considered. The work in [24] deals with integrity constraints in the context of *data exchange*, where the aim is to materialize a so-called *target schema* (which corresponds to the global schema in our setting). The contributions of this work are incomparable to ours: on the one hand, [24] deals with *tuple-generating dependencies* (*tgds*) and *equality-generating dependencies* (*egds*), which together are a more general class of constraints than the one we consider; on the other hand, the results of [24] are restricted to a class of tgds (the *weakly-acyclic* tgds) that ensures that the models of the target schema are always finite (otherwise the target schema could not be materialized). So the problem of dealing with infinite models is not addressed. Integrity constraints have also been dealt with in the context of deductive databases [26], however with a different aim, namely to check whether a set of integrity constraints holds in a database. Moreover, these works are not concerned with decidability or computational complexity of the problem. Integrity constraints are exploited in [27] for semantic query optimization in object databases.

The first contribution in this paper is to show that, when the global schema contains integrity constraints, even of simple forms, the semantics of the data integration system is best described in terms of a set of databases, rather than a single one, and this implies that, even in the global-as-view approach, query processing is intimately connected to the notion of querying *incomplete databases*. The fact that the problem of incomplete information is overlooked in current approaches can be explained by observing that traditional data integration systems follow one of the following strategies: they either express the global schema as a set of plain relations without integrity constraints, or they consider the sources as exact (see, e.g., [28,29]), as opposed to sound. On the contrary, the goal of our work is to study the more general setting where the global schema contains integrity constraints, and sources are considered sound (but not necessarily complete). The above result demonstrates that, in this case, we have to account for multiple global databases.

The second contribution of the paper is to study the case of global schemas expressed in the relational model with key and foreign key constraints, which represents a situation very common in practice. Although the problem of multiple global databases arises in this case, we have devised techniques for effectively answering queries posed to the data integration system. The resulting algorithm runs in polynomial time with respect to data complexity, i.e., with respect to the size of data at the sources.

The paper is organized as follows. In Section 2 we describe a formal framework for data integration. In Section 3 we show that the presence of integrity constraints in the global schema complicates the task of query processing. In Sections 4 and 5 we present our query processing algorithm for the case of global relational schema with key and foreign key constraints. Section 7 concludes the paper.

2. Framework for data integration

In this section we illustrate our formalization of a data integration system, which is based on the relational model with integrity constraints.

In the relational model, predicate symbols are used to denote the relations in the database, whereas constant symbols denote the values stored in relations. We assume to have a fixed (infinite) alphabet Γ of constants, and, if not specified otherwise, we will consider only databases over such an alphabet. In such a setting, the *unique name assumption* (that is, to assume that different constants denote different objects) is implicit. A *relational schema* (or simply *schema*) is constituted by:

- An *alphabet* $\mathcal{A}$ of predicate (or relation) symbols, each one with the associated arity, i.e., the number of arguments of the predicate (or, attributes of the relation). In the technical development, we do not use names for referring to attributes, rather, we simply use the numbers corresponding to their positions.
- A set of *integrity constraints*, i.e., assertions on the symbols of the alphabet $\mathcal{A}$ that express conditions that are intended to be satisfied in every database coherent with the schema.

A *relational database* (or simply, *database*) $\mathcal{DB}$ for a schema $\mathcal{C}$ is a set of relations with constants as atomic values, and with one relation $r^{\mathcal{DB}}$ of arity n for each predicate symbol r of arity n in the alphabet $\mathcal{A}$. It is well known that a database can be seen as a first-order interpretation for the relation symbols in the schema: the relation $r^{\mathcal{DB}}$ is the interpretation in $\mathcal{DB}$ of the predicate symbol r, in the sense that it contains the set of tuples that satisfy the predicate r in $\mathcal{DB}$. A database $\mathcal{DB}$ for a schema $\mathcal{C}$ is said to be *legal* if every constraint of $\mathcal{C}$ is satisfied by $\mathcal{DB}$. The notion of satisfaction depends on the type of constraints.

In our framework we consider the relational model with two kinds of constraints:

- *Key constraints*: given a relation r in the schema, a key constraint over r is expressed in the form $key(r) = \mathbf{A}$, where $\mathbf{A}$ is a set of attributes of r. Such a constraint is satisfied in a database $\mathcal{DB}$ if for each $t_1, t_2 \in r^{\mathcal{DB}}$, with $t_1 \neq t_2$, we have $t_1[\mathbf{A}] \neq t_2[\mathbf{A}]$, where $t[\mathbf{A}]$ is the projection of the tuple t over $\mathbf{A}$. In the following, we assume, without loss of generality, that the attributes constituting the key of a relation r are the first h attributes, i.e., $key(r) = \{1, \dots, h\}$, for an $h \leqslant arity(r)$.
- *Foreign key constraints*: a foreign key constraint is a statement of the form $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, where r_1, r_2 are relations, $\mathbf{A}$ is a sequence of distinct attributes of r_1, and $\mathbf{B}$ is $key(r_2)$, i.e., the sequence $[1, \dots, h]$ constituting the key of r_2. Such a constraint is satisfied in a database $\mathcal{DB}$ if for each tuple t_1 in $r_1^{\mathcal{DB}}$ there exists a tuple t_2 in $r_2^{\mathcal{DB}}$ such that $t_1[\mathbf{A}] = t_2[\mathbf{B}]$.

A *relational query* is a formula that specifies a set of tuples to be retrieved from a database. In this work, we consider the class of conjunctive queries. Formally, a *conjunctive query* q of *arity* n is written in the rule-based form

$$q(x_1, \dots, x_n) \leftarrow conj(x_1, \dots, x_n, y_1, \dots, y_m),$$

where

- q belongs to a new alphabet $\mathcal{Q}$ (the alphabet of queries, which is disjoint from both Γ and $\mathcal{A}$),
- $conj(x_1, \dots, x_n, y_1, \dots, y_m)$ is a conjunction of atoms involving the variables $x_1, \dots, x_n, y_1, \dots, y_m$, and a set of constants from Γ, and the predicate symbols of the atoms are in $\mathcal{C}$.

Note that, since all variables $x_1, \dots, x_n$ in the head appear also in the body, we are dealing with *safe* conjunctive queries.

The answer to a query q of arity n over a database $\mathcal{DB}$ for $\mathcal{G}$, denoted $q^{\mathcal{DB}}$, is the set of n-tuples of constants $(c_1, \dots, c_n)$, such that, when substituting each c_i for x_i, the formula $\exists(y_1, \dots, y_n) \cdot conj(x_1, \dots, x_n, y_1, \dots, y_m)$ evaluates to true in $\mathcal{DB}$. Note that the answer to q over $\mathcal{DB}$ is a relation whose arity is equal to the arity of the query q.

We now turn our attention to the notion of data integration system.

Definition 1. A *data integration system* $\mathcal{I}$ is a triple $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where $\mathcal{G}$ is the *global schema*, $\mathcal{S}$ is the *source schema*, and $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$.

The characteristics of the components of a data integration system in our approach are as follows:

- The *global schema* is expressed in the relational model with both key and foreign key constraints. We assume that in the global schema there is exactly one key constraint for each relation.
- The *source schema* is expressed in the relational model without integrity constraints. In other words, we conceive each source as a relation, and we consider the set of all relations as a unique schema, called source schema.
- The *mapping* $\mathcal{M}$ is defined following the global-as-view approach: to each relation r of the global schema $\mathcal{G}$ we associate a query $\rho(r)$ over the source schema $\mathcal{S}$. No special limitation is posed on the language used to express queries in the mapping $\mathcal{M}$, which means that, in principle, for any relation r in $\mathcal{G}$, $\rho(r)$ can be a computable query over the source schema. However, for the sake of practicality, it is reasonable to assume that such a query belongs to a class of queries with polynomial time data complexity, which implies that the evaluation of $\rho(r)$ over any source database $\mathcal{D}$ takes polynomial time with respect to the size of $\mathcal{D}$. This is exactly the assumption we make in the rest of the paper.
- Queries over the global schema are *conjunctive queries*. Indeed, in what follows, when we talk about queries posed to a data integration system, we always mean conjunctive queries.

Example 2. An example of data integration system is $\mathcal{I}^1 = \langle \mathcal{G}^1, \mathcal{S}^1, \mathcal{M}^1 \rangle$ where $\mathcal{G}^1$ is constituted by the relation symbols

$$\mathsf{person}(Pcode, Pname, Age, CityOfBirth)$$
$$\mathsf{student}(Scode, University)$$
$$\mathsf{city}(Cname, Major)$$

with the following integrity constraints:

$$key(\mathsf{person}) = \{Pcode\}$$
$$key(\mathsf{student}) = \{Scode\}$$
$$key(\mathsf{city}) = \{Cname\}$$

$$\mathsf{person}[CityOfBirth] \subseteq \mathsf{city}[Cname]$$
$$\mathsf{city}[Major] \subseteq \mathsf{person}[Pcode]$$
$$\mathsf{student}[Scode] \subseteq \mathsf{person}[Pcode]$$

$\mathcal{S}^1$ consists of three sources. Source s_1, of arity 4, contains information about persons with their code, name, age, and city of birth. Source s_2, of arity 2, contains information about enrollment of students in universities. Finally, Source s_3, of arity 2, contains information about cities with their name and their major. The mapping $\mathcal{M}^1$ is defined as follows:

$$\rho(\mathsf{person}) = \mathsf{per}(X, Y, Z, W) \leftarrow s_1(X, Y, Z, W)$$
$$\rho(\mathsf{student}) = \mathsf{stu}(X, Y) \leftarrow s_2(X, Y)$$
$$\rho(\mathsf{city}) = \mathsf{cit}(X, W) \leftarrow s_3(X, W)$$

In order to define the semantics of a data integration system, we start from the data at the sources, and specify which are the data that satisfy the global schema. A *source database* $\mathcal{D}$ for $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is constituted by one relation $r^{\mathcal{D}}$ for each source r in $\mathcal{S}$. We call *global database* for $\mathcal{I}$, or simply *database* for $\mathcal{I}$, any database for $\mathcal{G}$. A database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal* with respect to $\mathcal{D}$ if:

- $\mathcal{B}$ satisfies the integrity constraints of $\mathcal{G}$;
- $\mathcal{B}$ satisfies $\mathcal{M}$ with respect to $\mathcal{D}$, i.e., for each relation r in $\mathcal{G}$, the set of tuples $r^{\mathcal{B}}$ that $\mathcal{B}$ assigns to r is a subset of the set of tuples $\rho(r)^{\mathcal{D}}$ computed by the associated query $\rho(r)$ over $\mathcal{D}$, i.e., $\rho(r)^{\mathcal{D}} \subseteq r^{\mathcal{B}}$.

Note that the above definition amounts to consider any view $\rho(r)$ as *sound*, which means that the data provided by the sources are only a subset (possibly proper) of the data that would satisfy the relations of the global schema. Other assumptions on views are possible (see [14,18]). In particular, views may be *complete*, i.e., for each r in $\mathcal{G}$, we have $\rho(r)^{\mathcal{D}} \supseteq r^{\mathcal{B}}$, or *exact*, i.e., for each r in $\mathcal{G}$, we have $\rho(r)^{\mathcal{D}} = r^{\mathcal{B}}$. In this paper, we restrict our attention to sound views only, which are typically considered the most natural ones in a data integration setting [7,30].

At this point, we are able to give the semantics of a data integration system.

Definition 3. Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system and let $\mathcal{D}$ be a source database for $\mathcal{I}$. The *semantics* of $\mathcal{I}$ w.r.t. $\mathcal{D}$, denoted $sem^{\mathcal{D}}(\mathcal{I})$, is the set of databases for $\mathcal{I}$ that are legal w.r.t. $\mathcal{D}$, i.e., that satisfy both the constraints of $\mathcal{G}$, and the mapping $\mathcal{M}$ with respect to $\mathcal{D}$. If $sem^{\mathcal{D}}(\mathcal{I}) \neq \emptyset$, then $\mathcal{I}$ is said to be *consistent* w.r.t. $\mathcal{D}$.

By the above definition, it is clear that the semantics of a data integration system is formulated in terms of a *set* of databases, rather than a single one. Indeed, as we will show in the sequel, the cardinality of $sem^{\mathcal{D}}(\mathcal{I})$ is in general greater than one. The impact of this property on query answering will be studied in the next section.

3. Query answering in the presence of constraints

The ultimate goal of a data integration system is to answer queries posed by the user in terms of the global schema. Answering a query posed to a system representing a set of databases, is a complex task, as shown by the following example.

Example 4. Referring to Example 2, suppose we have the following source database $\mathcal{D}^1$:

$s_1^{\mathcal{D}^1}$:

101	*anne*	31	*florence*
107	*bill*	34	*oslo*

$s_2^{\mathcal{D}^1}$:

101	*bocconi*
120	*ucla*

$s_3^{\mathcal{D}^1}$:

oslo	101
florence	107

Now, due to the integrity constraints in $\mathcal{G}_1$, 120 is the code of some person. Observe, however, that nothing is said by $\mathcal{D}^1$ about the name, age and the city of birth of such a person. Therefore, we must accept as legal all databases that differ in such attributes of the person with code 120. Note that this is a consequence of the assumption of having sound views. If we had exact or complete views, this situation would have lead to an inconsistency of the data integration system. Instead, when dealing with sound views, we can think of extending the data contained in the sources in order to satisfy the integrity constraints over the global schema. The fact that, in general, there are several possible ways to carry out such an extension implies that there are several legal databases for the data integration system.

Let us now turn the attention to the notion of answer to a query posed to the data integration system. In our setting, a query q to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ is a conjunctive query, whose atoms have symbols in $\mathcal{G}$ as predicates. Our goal is to specify which are the tuples that form the answer to a certain query posed to $\mathcal{I}$. The task is complicated by the existence of several global databases which are legal for $\mathcal{I}$ w.r.t. a source database $\mathcal{D}$. In order to address this problem, we adopt the following approach: a tuple of constants is considered an answer to the query only if it is a *certain* answer, i.e., it satisfies the query in *every* database that belongs to the semantics of the data integration system.

Definition 5. Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, let $\mathcal{D}$ be a source database for $\mathcal{I}$, and let q be a query to $\mathcal{I}$. The *set of certain answers* $q^{\mathcal{I},\mathcal{D}}$ to q w.r.t. $\mathcal{I}$ and $\mathcal{D}$ is the set of tuples t of constants of the same arity as q, and $t \in q^{\mathcal{B}}$, for each $\mathcal{B} \in sem^{\mathcal{D}}(\mathcal{I})$.

As mentioned, it is generally assumed that query answering is an easy task in the global-as-view approach. Indeed, the most common technique for query answering in this approach is based on *unfolding*, i.e., substituting to each relation symbol r in the query its definition $\rho(r)$ in terms of the sources. We now show that a simple unfolding strategy is in general not sufficient for providing all correct answers in the presence of integrity constraints.

Example 6. Referring to Example 4, consider the query

$$q(X) \leftarrow \mathsf{person}(X, Y, Z, W) \wedge \mathsf{student}(X, Y)$$

The correct answer to the query is $\{101, 120\}$, because, due to the integrity constraints in $\mathcal{G}^1$, we know that 120 appears in the first attribute of person in all the databases for $\mathcal{I}^1$ that are legal w.r.t. $\mathcal{D}^1$. However, we do not get this information from $s_1^{\mathcal{D}^1}$, and, therefore, a simple unfolding strategy retrieves only the answer $\{101\}$ from $\mathcal{D}^1$, thus proving insufficient for query answering in this framework. Notice that, if the query asked for the person name instead of the person code (i.e., the head is $q(Y)$ instead of $q(X)$), then one could *not* make use of the dependencies to infer additional answers, and the correct answer would be $\{101\}$. Observe also that the same would happen if the head was $q(X, Y)$ or $q(X, Z)$.

The above example shows that, in the presence of integrity constraints, even in the global-as-view approach we have to deal with incomplete information during query processing.

4. Canonical database

We show that computing certain answers to conjunctive queries corresponds to evaluating the query over a special database, called canonical, which represents all possible global databases legal for the data integration system and which may be infinite in general.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system. In this paper we assume that, for each relation r of the global schema, the query $\rho(r)$ over the source schema $\mathcal{S}$ that the mapping $\mathcal{M}$ associates to r preserves the key constraint of r. This may require that $\rho(r)$ implements a suitable duplicate record elimination strategy that ensures that, for every source database $\mathcal{D}$, no pairs of tuples of $\mathcal{D}$ with the same value for the key of r satisfy $\rho(r)$. The problem of duplicate record elimination, and, more generally, of data cleaning, is a critical issue in data integration systems, however it is orthogonal to the problem addressed here. We refer to [31,32] for more details.

Let q be a conjunctive query posed to $\mathcal{I}$ and let $\mathcal{D}$ be a source database for $\mathcal{I}$. We illustrate a naive method for computing the answer $q^{\mathcal{I},\mathcal{D}}$ to q w.r.t. $\mathcal{I}$ and $\mathcal{D}$. The naive computation of $q^{\mathcal{I},\mathcal{D}}$ proceeds as follows:

(1) For each relation r of the global schema, we compute the relation $r^{\mathcal{D}}$ by evaluating the query $\rho(r)$ over the source database $\mathcal{D}$. The various relations so obtained form what we call the *retrieved global database* $ret(\mathcal{I},\mathcal{D})$. Note that, since we assume that $\rho(r)$ has been designed so as to resolve all key conflicts regarding r, the retrieved global database satisfies all key constraints in $\mathcal{G}$.

(2) If, additionally, the retrieved global database satisfies all foreign key constraints in $\mathcal{G}$, then we are basically done: we simply evaluate q over $ret(\mathcal{I},\mathcal{D})$, and we obtain the answer to the query.

Otherwise, based on the retrieved global database, we can build a database for $\mathcal{I}$ still satisfying the key constraints by suitably adding tuples to the relations of the global schema in such a way that also the foreign key constraints are satisfied.[1] Obviously, there are several possible ways to add tuples to the global relations.

We may try to infer all the legal databases for $\mathcal{I}$ that are coherent with the retrieved global database, and compute the tuples that satisfy the query q in all such legal databases. However, the difficulty here is that, in general, there is an infinite number of legal databases that are coherent with the retrieved global database. Fortunately, we show in the following that, starting from the retrieved global database, we can define another database, which we call *canonical*, that has the interesting property of faithfully representing all legal databases that are coherent with the retrieved global database. Although the canonical database is in general infinite, and therefore cannot be directly used for answering queries, we will show that it plays an important role in the method for query answering that we will describe in the next section.

Let us start by showing how to build the canonical database. First of all, we define the domain of such a database, which we denote $HD(\mathcal{D})$, as follows. Based on the global schema $\mathcal{G}$ of $\mathcal{I}$, we introduce the following set of function symbols:

$$HT(\mathcal{G}) = \{f_{r,i} \mid r \in \mathcal{G} \text{ and } i \leqslant arity(r) \text{ and } i \notin key(r)\}.$$

Each $f_{r,i}$ is a function symbol with the same arity as the number of attributes of $key(r)$, i.e., $arity(f_{r,i}) = arity(key(r))$. Intuitively, the role of the term $f_{r,i}(\alpha_1,\dots,\alpha_k)$ is to denote the value in the i-th column of the tuple of r having $\alpha_1,\dots,\alpha_k$ in the key columns.

From $\mathcal{D}$, we define the domain $HD(\mathcal{D})$ as the smallest set satisfying the following conditions:

- $\Gamma \subseteq HD(\mathcal{D})$,
- if $\alpha_1,\dots,\alpha_k \in HD(\mathcal{D})$, and $f_{r,i} \in HT(\mathcal{G})$, with $arity\ (f_{r,i}) = k$, then $f_{r,i}(\alpha_1,\dots,\alpha_k) \in HD(\mathcal{D})$.

Now, given the retrieved global database $ret(\mathcal{I},\mathcal{D})$, we construct inductively the canonical database $can(\mathcal{I},\mathcal{D})$ over the domain $HD(\mathcal{D})$ by starting from $ret(\iota,\mathcal{D})$ and repeatedly applying the following rule:

if $(x_1,\dots,x_h) \in r^{can(\mathcal{I},\mathcal{D})}[\mathbf{A}], (x_1,\dots,x_h) \notin r_2^{can(\mathcal{I},\mathcal{D})}[\mathbf{B}]$, and the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is in $\mathcal{G}$, then insert in $r_2^{can(\mathcal{I},\mathcal{D})}$ the tuple t such that

- $t[\mathbf{B}] = (x_1,\dots,x_h)$, and
- for each i such that $1 \leqslant i \leqslant arity(r_2)$, and i not in $\mathbf{B}$, $t[i] = f_{r_2,i}(x_1,\dots,x_h)$.

Note that the above rule does enforce the satisfaction of the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, by adding a suitable tuple in r_2: the key of the new tuple is determined by the values in $r_1[\mathbf{A}]$, and the values of the non-key attributes are formed by means of the function symbols $f_{r_2,i}$.

Example 7. Suppose that we have two relations r and s in $\mathcal{G}$, both of arity 2 and having as key the first attribute, and that the following dependencies

[1] Note that, since views are sound, i.e., they return a *subset* of the tuples in a global relation, we cannot conclude that the data integration system violates the foreign key constraints of $\mathcal{G}$. Indeed, it may be the case that the tuples needed to satisfy such constraints are not part of the retrieved subsets.

hold on $\mathcal{G}$:

$r[2] \subseteq s[1]$

$s[1] \subseteq r[1]$

Suppose that the retrieved global database stores a single tuple (a, b) in r. Then, by applying the above rule, we insert the tuple $(b, f_{s,2}(b))$ in s; successively we add $(b, f_{r,2}(b))$ in r, then $(f_{r,2}(b), f_{s,2}(f_{r,2}(b)))$ in s and so on. Observe that the two dependencies are cyclic, and in this case the construction of the canonical database requires an infinite sequence of applications of the rules.

Observe that $can(\mathcal{I}, \mathcal{D})$ is indeed a database over the domain $HD(\mathcal{D})$, and that $can(\mathcal{I}, \mathcal{D})$ may be infinite, in particular this happens when the foreign key constraints of $\mathcal{G}$ are cyclic, as in Example 7. However, it enjoys important properties, as shown in the following theorems.

Theorem 8. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system and $\mathcal{D}$ is a source database for $\mathcal{I}$. Then $can(\mathcal{I}, \mathcal{D})$ is a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$ if and only if $ret(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$.*

Proof. It is immediate to see that if $ret(\mathcal{I}, \mathcal{D})$ violates some key constraint in $\mathcal{G}$, then no legal database exists for $\mathcal{I}$ w.r.t. $\mathcal{D}$.

It remains to show that, if $ret(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$, then $can(\mathcal{I}, \mathcal{D})$ satisfies all the constraints in $\mathcal{G}$, which implies that $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$. To show that $can(\mathcal{I}, \mathcal{D})$ is indeed a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$, we consider key and foreign key constraints separately. As for key constraints, it is easy to see that the tuples inserted during the process of computing $can(\mathcal{I}, \mathcal{D})$ cannot violate any key constraints of $\mathcal{G}$. Indeed, in computing $can(\mathcal{I}, \mathcal{D})$, we insert a tuple into a relation r only when the key component of that tuple is not already present in r. Since $ret(\mathcal{I}, \mathcal{D})$ does not violate any key constraint in $\mathcal{G}$, it follows that no key constraint of $\mathcal{G}$ is violated in $can(\mathcal{I}, \mathcal{D})$. As for foreign key constraints, suppose by contradiction that the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ is violated in $can(\mathcal{I}, \mathcal{D})$. This implies that there is a tuple t in r_1 such that for no tuple t' in r_2 we have that $t'[\mathbf{B}] = t[\mathbf{A}]$. But this would imply that we can apply the rule used for constructing $can(\mathcal{I}, \mathcal{D})$, and insert a new tuple t'' in r_2 such that $t''[\mathbf{B}] = t[\mathbf{A}]$, and for each $i \leqslant arity(r_2)$ not in $\mathbf{B}$, $t'[i] = f_{r_2,i}(t[\mathbf{A}])$. Since this is impossible because of the way $can(\mathcal{I}, \mathcal{D})$ has been built, we conclude that no foreign key constraint is violated by $can(\mathcal{I}, \mathcal{D})$. □

We observe that the construction of the canonical database is similar to the construction of the *restricted chase* of a database described in [33]. The difference lies in the fact that, in the construction of the chase, fresh values are used instead of terms involving Skolem functions. In the canonical database, such terms keep track of the application of the rules, and this is necessary for our query answering technique, as we will show in Section 5.

Another interesting property of the canonical database $can(\mathcal{I}, \mathcal{D})$ is that it faithfully represents all legal databases that are coherent with the retrieved global database $ret(\mathcal{I}, \mathcal{D})$. In particular, the following theorem shows that, for every database $\mathcal{B}$ that is legal for $\mathcal{I}$ w.r.t. $\mathcal{D}$, there is a homomorphism from $can(\mathcal{I}, \mathcal{D})$ to $\mathcal{B}$ that induces a well-defined mapping from the tuples in the relations of $can(\mathcal{I}, \mathcal{D})$ to the tuples in the relations of $\mathcal{B}$.

Theorem 9. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, $\mathcal{D}$ a source database for $\mathcal{I}$, and $\mathcal{B}$ a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$. Then there is a function ψ from $HD(\mathcal{D})$ to Γ such that, for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted of elements in $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r^{can(\mathcal{I}, \mathcal{D})}$, then $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$.*

Proof. We define the function ψ from $HD(\mathcal{D})$ to Γ inductively, and we simultaneously show that for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted of elements in $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r^{can(\mathcal{I}, \mathcal{D})}$, then $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$.

We proceed by induction on the application of the rule used during the construction of $can(\mathcal{I}, \mathcal{D})$. As a base step, the function ψ maps each constant in $ret(\mathcal{I}, \mathcal{D})$ to itself. Since $\mathcal{B}$ is legal for $\mathcal{I}$ w.r.t. $\mathcal{D}$, for each relation r we have that $r^{ret(\mathcal{I}, \mathcal{D})} \subseteq r^{\mathcal{B}}$. Hence, for each r, if $c_1, \ldots, c_n$ are constants, and $(c_1, \ldots, c_n) \in r^{ret(\mathcal{I}, \mathcal{D})}$, we have that $(\psi(c_1), \ldots, \psi(c_n)) = (c_1, \ldots, c_n) \in r^{\mathcal{B}}$.

Inductive step. Suppose that in the application of the rule, we are inserting the tuple $(\alpha_1, \ldots, \alpha_h, f_{r,h+1}(\alpha_1, \ldots, \alpha_h), \ldots, f_{r,n}(\alpha_1, \ldots, \alpha_h))$ in $r^{can(\mathcal{I},\mathcal{D})}$ where r has arity n, $key(r) = \{1, \ldots, h\}$, and the tuple is inserted in $r^{can(\mathcal{I},\mathcal{D})}$ because of the foreign key constraint $w[j_1, \ldots, j_h] \subseteq r[1, \ldots, h]$. This means that there is a tuple t in $w^{can(\mathcal{I},\mathcal{D})}$ such that $t[j_1, \ldots, j_h] = (\alpha_1, \ldots, \alpha_h)$, and $(\alpha_1, \ldots, \alpha_h) \notin r^{can(\mathcal{I},\mathcal{D})}[1, \ldots, h]$. By induction hypothesis, there are $\beta_1, \ldots, \beta_h$ in Γ such that $\psi(\alpha_1) = \beta_1, \ldots, \psi(\alpha_h) = \beta_h$, and there is a tuple $t' \in w^{\mathcal{B}}$ such that for each i, $t'[i] = \psi(t[i])$, and $t'[j_1, \ldots, j_h] = (\psi(\alpha_1), \ldots, \psi(\alpha_h)) = (\beta_1, \ldots, \beta_h)$. Because of the constraint $w[j_1, \ldots, j_h] \subseteq r[1, \ldots, h]$, and because $\mathcal{B}$ is legal and $\{1, \ldots, h\}$ is the key of r, there is one and only one tuple $(\beta_1, \ldots, \beta_h, \beta_{h+1}, \ldots, \beta_n)$ in $r^{\mathcal{B}}$ having $(\beta_1, \ldots, \beta_h)$ as values for the key attributes, which implies that ψ has not assigned any value yet to $f_{r,h+1}(\alpha_1, \ldots, \alpha_h), \ldots, f_{r,n}(\alpha_1, \ldots, \alpha_h)$. Then, we set $\psi(f_{r,h+1}(\alpha_1, \ldots, \alpha_h)) = \beta_{h+1}, \ldots, \psi(f_{r,n}(\alpha_1, \ldots, \alpha_h)) = \beta_n$, and we can conclude that $(\psi(\alpha_1), \ldots, \psi(\alpha_h), \psi(f_{r,h+1}(\alpha_1, \ldots, \alpha_h)), \ldots, \psi(f_{r,n}(\alpha_1, \ldots, \alpha_h))) \in r^{\mathcal{B}}$. □

Finally, we show that, if $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, then a tuple t of constants is in $q^{\mathcal{I},\mathcal{D}}$ if and only if t is in the answer to q over the database $can(\mathcal{I},\mathcal{D})$. Note that terms involving Skolem functions are not relevant to the user, so they are never part of the certain answers. Therefore, even if the canonical database is infinite, the set of certain answers is always finite.

Theorem 10. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, q a conjunctive query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$ such that $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, and t a tuple of constants of the same arity as q. Then $t \in q^{\mathcal{I},\mathcal{D}}$ if and only if $t \in q^{can(\mathcal{I},\mathcal{D})}$.*

Proof. For the "if" direction, we show that if t is in the answer to q over $can(\mathcal{I},\mathcal{D})$, then $t \in q^{\mathcal{I},\mathcal{D}}$. Indeed, consider any $\mathcal{B}$ that is a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$. By Theorem 9, there is a function ψ from $HD(\mathcal{D})$ to Γ such that, for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted of elements in $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r^{can(\mathcal{I},\mathcal{D})}$, then $(\psi(c_1), \ldots, \psi(c_n)) \in r^{\mathcal{B}}$. The fact that t is in the answer to q over $can(\mathcal{I},\mathcal{D})$ means that there is an assignment α from the variables of q to objects in $HD(\mathcal{D})$ such that all atoms of q are true with respect to the assignment. It is easy to see that the assignment $\alpha \cdot \psi$ can be used to show that t is in the answer to q over $\mathcal{B}$.

As for the "only-if" direction, first note that, by hypothesis, $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, and, therefore, by Theorem 8, $ret(\mathcal{I},\mathcal{D})$ does not violate any key constraint in $\mathcal{G}$, which implies that $can(\mathcal{I},\mathcal{D})$ is a legal database for $\mathcal{I}$ w.r.t. $\mathcal{D}$. Therefore, from the fact that t is not in the answer to q over $can(\mathcal{I},\mathcal{D})$, we can conclude that $t \notin q^{\mathcal{I},\mathcal{D}}$. □

Example 11. Considering Example 6, the construction of the canonical database $can(\mathcal{I}^1, \mathcal{D}^1)$ proceeds as follows. First of all, to repair the violation of the constraint $\mathsf{student}[Scode] \subseteq \mathsf{person}[Pcode]$ we add $(120, f_{\mathsf{person},2}(120), f_{\mathsf{person},3}(120), f_{\mathsf{person},4}(120))$ to person. Now, from the constraint $\mathsf{person}[CityOfBirth] \subseteq \mathsf{city}[Cname]$, we deduce that $f_{\mathsf{person},4}(120)$ is the name of a city, so we add $(f_{\mathsf{person},4}(120), f_{\mathsf{city},2}(f_{\mathsf{person},4}(120)))$ to city. From $\mathsf{city}[Major] \subseteq \mathsf{person}[Pcode]$, we infer that $f_{\mathsf{city},2}(f_{\mathsf{person},4}(120)))$ is the code of some person, and we proceed adding a new tuple in person, and so on. Note that the canonical database is infinite. Recalling Example 6, we see that the evaluation of q over $can(\mathcal{I}^1, \mathcal{D}^1)$ produces $\{101, 120\}$ (disregarding terms involving Skolem functions).

Based on the above results, we can conclude that $can(\mathcal{I},\mathcal{D})$ is the right abstraction for answering queries posed to the data integration system. However, as we said before, $can(\mathcal{I},\mathcal{D})$ is in general infinite. In the next section we show that, in processing a query q posed to the data integration system, we can find the answers to q over $can(\mathcal{I},\mathcal{D})$ without actually building $can(\mathcal{I},\mathcal{D})$.

5. The Algorithm for query answering

The naive computation described in the previous section is impractical, because it requires to build the canonical database, which is in general infinite. In order to overcome the problem, we present an algorithm consisting of two separate phases. First, as we said in the previous section, we assume that, for each relation r of the global schema, the query $\rho(r)$ over the source schema that the mapping $\mathcal{M}$ associates to r preserves the key constraint of r.

(1) Instead of referring explicitly to the canonical database for query answering, we transform the original query q into a new query $exp_{\mathcal{G}}(q)$ over the global schema, called the *expansion of q w.r.t.* $\mathcal{G}$, such that the answer to $exp_{\mathcal{G}}(q)$ over the retrieved global database is equal to the answer to q over the canonical database.

(2) In order to avoid building the retrieved global database, we do not evaluate $exp_{\mathcal{G}}(q)$ on the retrieved global database. Instead, we unfold $exp_{\mathcal{G}}(q)$ to a new query, called $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$, over the source relations on the basis of $\mathcal{M}$, and we use the unfolded query $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ to access the sources.

Fig. 1 shows the basic idea of our approach. In order to obtain the certain answers $q^{\mathcal{I},\mathcal{D}}$, the user query q could in principle be evaluated (dashed arrow) over the (possibly infinite) canonical database $can(\mathcal{I},\mathcal{D})$, which is generated from the retrieved global database $ret(\mathcal{I},\mathcal{D})$ by applying rules described in Section 4. In turn, $ret(\mathcal{I},\mathcal{D})$ can be obtained from the source database $\mathcal{D}$ by evaluating the queries of the mapping. Our query answering process instead expands the query according to the constraints in $\mathcal{G}$, unfolds it according to $\mathcal{M}$, and then evaluates it on the source database.

We first deal with the query reformulation step constituting Phase (1). At the end of the section we briefly discuss Phase (2), which is the standard unfolding process in data integration [7,30].

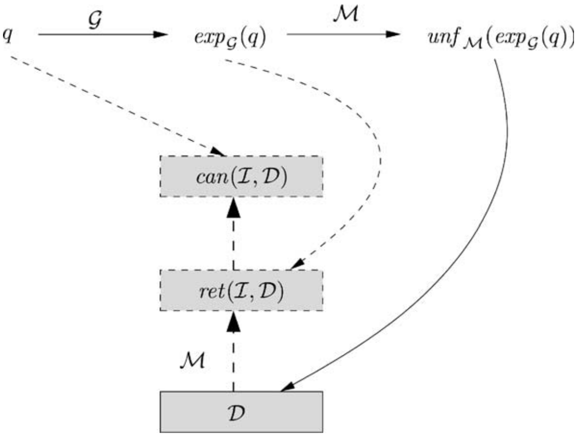
q
$\mathcal{G}$
$exp_{\mathcal{G}}(q)$
$\mathcal{M}$
$unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$
$can(\mathcal{I},\mathcal{D})$
$ret(\mathcal{I},\mathcal{D})$
$\mathcal{M}$
$\mathcal{D}$

Fig. 1. Query answering process.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, let $\mathcal{D}$ be a source database, and let q be a query over the global schema $\mathcal{G}$. We show how to reformulate the original query q into a new query $exp_{\mathcal{G}}(q)$ over the global schema, such that the answer to $exp_{\mathcal{G}}(q)$ over the (virtual) retrieved global database is equal to the answer to q over the canonical database.

The basic idea to do so is that the constraints in $\mathcal{G}$ can be captured by a suitable (*definite*) *logic program* $\mathcal{P}_{\mathcal{G}}$ [34]. To build $\mathcal{P}_{\mathcal{G}}$, we introduce for each relation symbol p in $\mathcal{G}$ a new relation symbol p' called *primed relation*. We call *primed atom* every atom whose relation symbol is primed. Then, taking into account the semantics of $\mathcal{G}$, we define $\mathcal{P}_{\mathcal{G}}$ to be constituted by the following rules (expressed in Logic Programming notation [34]):

- for each relation r of arity n, a rule:

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

- for each foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$ in $\mathcal{G}$, where $\mathbf{A} = \{i_1, \ldots, i_h\}$, $key(r_2) = \mathbf{B} = \{1, \ldots, h\}$, $arity(r_1) = n$, and $arity(r_2) = m$, a rule:

  $r'_2(X_{i_1}, \ldots, X_{i_h}, f_{r_2,h+1}(X_{i_1}, \ldots, X_{i_h}), \ldots, f_{r_2,n}$
  $(X_{i_1}, \ldots, X_{i_h})) \leftarrow r'_1(X_1, \ldots, X_m)$

Observe that, if the foreign key constraints of the global schema $\mathcal{G}$ are cyclic, then the logic program $\mathcal{P}_{\mathcal{G}}$ is recursive. In what follows, we denote by $head(\sigma)$ the head of a rule σ, and by $body(\sigma)$ the body of σ.

Example 12. We refer to Example 11. The logic program $\mathcal{P}_{\mathcal{G}}$ makes use of the primed relations $\mathsf{person}'/3$, $\mathsf{student}'/1$, $\mathsf{city}'/2$ and constitutes of the following (recursive) rules:

$\mathsf{person}'(X, Y, Z) \leftarrow \mathsf{person}(X, Y, Z)$
$\mathsf{student}'(X, Y) \leftarrow \mathsf{student}(X, Y)$
$\mathsf{city}'(X, Y) \leftarrow \mathsf{city}(X, Y)$
$\mathsf{city}'(X, f_{\mathsf{city},2}(X)) \leftarrow \mathsf{person}'(Y, Z, X)$
$\mathsf{person}'(Y, f_{\mathsf{person},2}(Y), f_{\mathsf{person},3}(Y)) \leftarrow \mathsf{city}'(X, Y)$
$\mathsf{person}'(X, f_{\mathsf{person},2}(X), f_{\mathsf{person},3}(X)) \leftarrow \mathsf{student}'(X, Y)$

If we consider $ret(\mathcal{I},\mathcal{D})$ as a set of logic program facts, it is immediate to verify that $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$ is again a definite logic program, and that the *Herbrand universe* of $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$ coincides with $HD(\mathcal{D})$. We remind the reader that every definite logic program $\mathcal{P}$ has a minimal Herbrand model, denoted $min(\mathcal{P})$ [34]. The model $min(\mathcal{P})$ has the Herbrand universe of $\mathcal{P}$ as domain, and plays a crucial role in characterizing which are the goals (conjunctions of atoms) that logically follow from the program $\mathcal{P}$.

The relationship between the logic program $\mathcal{P}_\mathcal{G}$ and the canonical database of the data integration system $\mathcal{I}$ is characterized by the following theorem. We denote by q' the query obtained by rewriting q by substituting each relation symbol r in the body $body(q)$ of q with the primed symbol r'.

Theorem 13. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, q a conjunctive query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$, and t a tuple of constants of the same arity as q. Then $t \in q^{can(\mathcal{I},\mathcal{D})}$ if and only if $t \in q'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$.*

Proof. One direction is immediate: since, up to the renaming of each relation symbol r by the corresponding primed symbol r', $can(\mathcal{I},\mathcal{D})$ is a subset of $min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))$, and q is monotone, it is obvious that $t \in q^{can(\mathcal{I},\mathcal{D})}$ implies $t \in q'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$.

For the other direction, we inductively define a function ϕ from $HD(\mathcal{D})$ to $HD(\mathcal{D})$, and we simultaneously show that for each relation r of arity n in $\mathcal{G}$, and each tuple $(c_1, \ldots, c_n)$ constituted of elements of $HD(\mathcal{D})$, if $(c_1, \ldots, c_n) \in r'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$, then $(\phi(c_1), \ldots, \phi(c_n)) \in r^{can(\mathcal{I},\mathcal{D})}$. We proceed by induction on the application of the logic program rules used during the construction of $min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))$.

Base step. The function ϕ maps each constant in $ret(\mathcal{I},\mathcal{D})$ into itself. It follows that, for each r, if $c_1, \ldots, c_n$ are constants, and $(c_1, \ldots, c_n) \in r^{ret(\mathcal{I},\mathcal{D})}$, then both $(c_1, \ldots, c_n) \in r'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$, and $(\phi(c_1), \ldots, \phi(c_n)) \in r^{can(\mathcal{I},\mathcal{D})}$.

Inductive step. Suppose that in the application of the logic program rules, we are inserting the tuple $(\alpha_1, \ldots, \alpha_h, f_{r,h+1}(\alpha_1, \ldots, \alpha_h), \ldots, f_{r,n}(\alpha_1, \ldots, \alpha_h))$ in $r'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$ where r has arity n, $key(r) = \{1, \ldots, h\}$, and the tuple is inserted in $r'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$ because of the rule

$$r'(X_{j_1}, \ldots, X_{j_h}, f_{w,h+1}(X_{j_1}, \ldots, X_{j_h}), \ldots, f_{w,n}(X_{j_1}, \ldots, X_{j_h})) \leftarrow w'(X_1, \ldots, X_m)$$

This means that there is a tuple t in $w'^{min(\mathcal{P}_G \cup ret(\mathcal{I},\mathcal{D}))}$ such that $t[j_1, \ldots, j_h] = (\alpha_1, \ldots, \alpha_h)$. For the induction hypothesis, there are $\beta_1, \ldots, \beta_h$ in $HD(\mathcal{D})$ such that $\phi(\alpha_1) = \beta_1, \ldots, \phi(\alpha_h) = \beta_h$, and there is a tuple $t' \in w^{can(\mathcal{I},\mathcal{D})}$ such that for each i, $t'[i] = \phi(t[i])$, and $t'[j_1, \ldots, j_h] = (\phi(\alpha_1), \ldots, \phi(\alpha_h)) = (\beta_1, \ldots, \beta_h)$. Because of the constraint $w[j_1, \ldots, j_h] \subseteq r[1, \ldots, h]$, and because $can(\mathcal{I},\mathcal{D})$ is legal and $\{1, \ldots, h\}$ is the key of r, there is one and only one tuple $(\beta_1, \ldots, \beta_h, \beta_{h+1}, \ldots, \beta_n)$ in $r^{can(\mathcal{I},\mathcal{D})}$ having $(\beta_1, \ldots, \beta_h)$ as values for the key attributes. Then, we set $\phi(f_{r,h+1}(\alpha_1, \ldots, \alpha_h)) = \beta_{h+1}, \ldots, \phi(f_{r,n}(\alpha_1, \ldots, \alpha_h)) = \beta_n$, and we can conclude that $(\phi(\alpha_1), \ldots, \phi(\alpha_h), \phi(f_{r,h+1}(\alpha_1, \ldots, \alpha_h)), \ldots, \phi(f_{r,n}(\alpha_1, \ldots, \alpha_h))) \in r^{can(\mathcal{I},\mathcal{D})}$. □

The logic program $\mathcal{P}_\mathcal{G}$ is used to construct the query $exp_\mathcal{G}(q)$ associated to the original query q. This construction builds upon the standard notion of SLD-tree [34] for the query q in the program $\mathcal{P}_\mathcal{G}$. The SLD-tree for a program containing recursive rules is in general infinite. In our approach, we build only a finite portion of the SLD-tree, by stopping the expansion of a branch when certain conditions are met. Building such a partial SLD-tree is known as *partial evaluation* of logic programs [35].

To generate the query $exp_\mathcal{G}(q)$, we adopt the following algorithm, called below *expansion algorithm* for q in $P_\mathcal{G}$: we build a (finite) *expansion tree* for q' in $P_\mathcal{G}$ (see below), and we return as result the query $exp_\mathcal{G}(q)$ formed as the union of all non-empty queries in the leaves of the expansion tree.

The *expansion tree* for q' in $P_\mathcal{G}$ is defined as follows:

- The root is labeled by q', and has one (primed) atom (for example the first in left-to-right order) marked as selected. We say that the root has *depth* 0.
- Except if one of the stopping conditions below is satisfied, a node d of depth k, labeled by a query g having a "selected" atom α, has one child for each rule σ in $\mathcal{P}_\mathcal{G}$ such that there exists

a most general unifier[2] $mgu(\alpha, head(\sigma))$ between the atom α and the head $head(\sigma)$ of the rule σ, in which the distinguished variables are not assigned to terms involving function symbols. Each of such children has the following properties:
  - it has depth $k+1$;
  - it is labeled by the query obtained from g by replacing the atom α with $body(\sigma)$ and by substituting the variables according to $mgu(\alpha, head(\sigma))$;
  - it has as marked "selected" one of the primed atoms (for example the first in left-to-right order).
- If one of the stopping conditions below is satisfied for a node d, then d has a single child, which is labeled by the so-called *empty query*, i.e., a query that has no atoms, and is denoted by the special symbol "□".

The stopping conditions are as follows:

(1) The node d has depth $k > n_q \cdot (n_q \cdot n_c \cdot (w+1)^w + 1)$, where n_q is the number of atoms in the query q, n_c is the number of foreign key constraints in $\mathcal{G}$, and w is the maximum number of attributes of the key of any relation in $\mathcal{G}$.

(2) The node d is labeled by a query g, and there exists a predecessor $\bar{d}$ of d labeled by a query $\bar{g}$ and a substitution θ of the variables of $\bar{g}$ that makes $\bar{g}$ equal to g.

We now show termination, soundness, and completeness of the expansion algorithm. We observe that the second stopping condition is in fact an optimization step, that does not affect soundness and completeness, but allows for further limiting the expansion of the tree.

Theorem 14 (Termination of the expansion). *Let $\mathcal{G}$ be a global schema and q a conjunctive query. Then the expansion algorithm for q w.r.t. $P_{\mathcal{G}}$ terminates, and the query $exp_{\mathcal{G}}(q)$ is a finite union of conjunctive queries.*

Proof. This is a trivial consequence of the bound on depth of the nodes of the tree that can be further expanded. □

Next we prove soundness of the expansion algorithm, i.e., that the answers to the expanded query $exp_{\mathcal{G}}(q)$ are also answers to q.

Theorem 15 (Soundness of the expansion). *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, q a conjunctive query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$, and t a tuple of constants of the same arity as q. Then $t \in q^{can(\mathcal{I},\mathcal{D})}$ if $t \in exp_{\mathcal{G}}(q)^{ret(\mathcal{I},\mathcal{D})}$.*

Proof. Since $exp_{\mathcal{G}}(q)$ is a finite union of conjunctive queries, $t \in exp_{\mathcal{G}}(q)^{ret(\mathcal{I},\mathcal{D})}$ if and only if there is a conjunctive query q_ℓ which is a disjunct in $exp_{\mathcal{G}}(q)$ such that $t \in q_\ell^{ret(\mathcal{I},\mathcal{D})}$. By the completeness of SLD-resolution [34], there exists an SLD-refutation δ_ℓ of q_ℓ w.r.t. $\mathcal{P}_{\mathcal{G}}$ returning as answer a substitution θ_t corresponding to t. Now consider the path from the query q' to q_ℓ in the expansion tree. Such a path corresponds to an SLD-derivation that can be continued with the SLD-refutation δ_ℓ. The resulting SLD-derivation is an SLD-refutation for q' w.r.t. $\mathcal{P}_{\mathcal{G}}$ with answer θ_t (note that the distinguished variables of the query q and hence q' are never instantiated during the SLD-derivations corresponding to the paths of the expansion tree). By the soundness of SLD-resolution [34], it follows that $t \in q'^{min(\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D}))}$, and hence, by Theorem 13, we have $t \in q^{can(\mathcal{I},\mathcal{D})}$. □

Next we turn our attention to completeness. We call (ground) *facts of a database $\mathcal{DB}$*, assertions of the form $r(t)$, where r is a relation symbol and t is a tuple in $r^{\mathcal{DB}}$. We introduce the notion of *level* of a fact f in the canonical database $can(\mathcal{I},\mathcal{D})$, denoted $level(f)$, inductively defined as follows:

- if $r(t)$ is a fact of $ret(\mathcal{I},\mathcal{D})$), then $level(r(t)) = 0$;
- if $r_1(t_1)$ is a fact of $can(\mathcal{I},\mathcal{D})$ with $level(r_1(t_1)) = k$, and t_2 is inserted in r_2 during the construction of $can(\mathcal{I},\mathcal{D})$ because of the foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$, then $level(r_2(t_2)) = k+1$.

Below, let n_q be the number of atoms in the query q, n_c the number of foreign key constraints

[2] We recall that, given two atoms α and β, the most general unifier $mgu(\alpha, \beta)$ is a most general substitution for the variables in α and β that makes α and β equal [34].

in $\mathcal{G}$, and w the maximum number of attributes of the key of any relation in $\mathcal{G}$.

Lemma 16. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, q a conjunctive query, and t a tuple of constants of the same arity as q, such that $t \in q^{can(\mathcal{I},\mathcal{D})}$. Then there exists a substitution of the distinguished and non-distinguished variables of q that assigns t to the distinguished variables and elements of $HD(\mathcal{D})$ to the non-distinguished variables, so that all atoms of the body of q become facts $f_1, \ldots, f_{n_q}$ of $can(\mathcal{I},\mathcal{D})$ having $level(f_i) \leqslant n_q \cdot n_c \cdot (w+1)^w$.*

Proof. The proof follows directly from Theorem 2 in [33], by observing that the canonical database $can(\mathcal{I},\mathcal{D})$ is constructed exactly as the *restricted chase* in [33], and that the level of a fact of $can(\mathcal{I},\mathcal{D})$ corresponds the level of a conjunct in the chase graph. From the results in [33], we have that there is a substitution of the distinguished and non-distinguished variables of q assigning t to the distinguished variables, such that the atoms in q, once the substitution is applied, become conjuncts of the chase having level less or equal to $n_q \cdot n_c \cdot (w+1)^w$. □

Lemma 17. *Let $\{f_1, \ldots, f_n\}$ be a set of ground facts of $can(\mathcal{I},\mathcal{D})$ each of which has $level(f_i) \leqslant k$. Then there exists an SLD-refutation for the goal $f'_1, \ldots, f'_n$ in $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$,* where *$f'_i$ is the primed atom corresponding to f_i, having length $\ell \leqslant n \cdot (k+1)$.*

Proof. Consider one of the facts f_i. We show that, if $level(f_i) = k$, then there exists an SLD-refutation of f'_i of length $k+1$. We proceed by induction on k.

If $k = 0$ then f_i is a fact $r(t)$ of $ret(\mathcal{I},\mathcal{D})$. Hence by resolving $r'(t)$ with the rule in $\mathcal{P}_\mathcal{G}$ of the form $r'(x_1, \ldots, x_m) \leftarrow r(x_1, \ldots, x_m)$, where m is the arity of r, we get an SLD-refutation of length 1.[3]

If $k > 0$ then $f_i = r_2(t_2)$ has been obtained from a fact $f = r_1(t_1)$ with $level(f) = k-1$ by applying a foreign key constraint $r_1[\mathbf{A}] \subseteq r_2[\mathbf{B}]$. By induction hypothesis, there is an SLD-refutation for the goal $f' = r'_1(t_1)$ of length k. Hence, by applying the rule corresponding to the foreign key, we get a SLD-refutation for $r'_2(t_2)$ of length $k+1$ from the one for $r_1(t_1)$.

By applying such a result for each fact f_i we get the claim. □

Theorem 18 (Completeness of the expansion). *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, q a conjunctive query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$, and t a tuple of constants of the same arity as q. Then $t \in exp_\mathcal{G}(q)^{ret(\mathcal{I},\mathcal{D})}$ if $t \in q^{can(\mathcal{I},\mathcal{D})}$.*

Proof. Let us first assume that stopping condition (2) never applies. By Lemma 16, we know that if $t \in q^{can(\mathcal{I},\mathcal{D})}$ then we can find a substitution for the distinguished and non-distinguished variables in q assigning t to the distinguished variables and such that all atoms of the body of $q(t)$ are facts $f_1, \ldots, f_{n_q}$ having $level(f_i) \leqslant n_q \cdot n_c \cdot (w+1)^w$.

On the other hand, by Lemma 17, we know that there is an SLD-refutation δ_{ground} for $f'_1, \ldots, f'_{n_q}$ in $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$ of length $n_q \cdot (n_q \cdot n_c \cdot (w+1)^w + 1)$.

By the Lifting Lemma [34], there exists a SLD-refutation δ_{lifted} of q' in $\mathcal{P}_\mathcal{G} \cup ret(\mathcal{I},\mathcal{D})$ of the same length as δ that returns as answer a substitution θ_t of the (distinguished and non-distinguished) variables making the body of the query equal to $f'_1, \ldots, f'_{n_q}$. Note that, by applying θ_t to the distinguished variables, we get the tuple t.

Consider the partial SLD-tree obtained from the expansion tree. The SLD-derivations that are stopped before arriving to all non-primed atoms in the expansion tree are all longer than the length of δ_{ground}. Hence, by the independence of the computation rule [34], a variant of the SLD-refutation (with the same length and the same computed answer) is present in the expansion tree, once we resolve the facts in $ret(\mathcal{I},\mathcal{D})$. Let us call such a refutation δ_{expan}.

Take the query q_ℓ that is in $\exp_G(q)$ labeling the leaf of the path in the expansion tree used in δ_{expan}. Such a leaf will be n_q steps before the empty goal in the refutation, and hence q_ℓ has as refutation the last n_q steps of δ_{expan}. Considering that the distinguished variables are only bounded by objects in $ret(\mathcal{I},\mathcal{D})$, and hence such variables in

[3] Differently from [34], in computing the length of a refutation, we do not count the final resolution step leading to the empty goal.

δ_{expan} get a substitution in the last n_q steps, we have that $t \in q_\ell^{ret(\mathcal{I},\mathcal{D})}$.

Now, consider the case in which stopping condition (2) may also apply. Completeness follows from the result just proved, once we make the following observation. Suppose that the shortest (possibly the only one) SLD-refutation for q' in $\mathcal{P}_{\mathcal{G}} \cup ret(\mathcal{I},\mathcal{D})$ assigning t to the distinguished variables goes through a node d, satisfying stopping condition (2). Suppose that d is labeled by goal g_t. Let us say that the length of the SLD-refutation is h, and that the node the labeled by g_t is the kth node along the SLD-refutation. Since the stopping condition applies, there is another goal $\bar{g}_t$ at a predecessor node in the SLD-refutation such that $\bar{g}_t\theta = g_t$, for some substitution θ. Obviously, such a SLD-refutation is also an SLD-refutation for $\bar{g}_t\theta$. But then, by the Lifting Lemma [34], there is an SLD-refutation of $\bar{g}_t$ of the same length k. Hence, there exists an SLD-refutation for q', assigning t to the distinguished variables, of length strictly shorter than h. This leads to a contradiction. This implies that for each SLD-refutation for q' that assigns t to the distinguished variables, if it goes through a node satisfying the stopping condition above, then there is also another refutation which is shorter that does not go through that node. Hence we may drop from the expansion tree for $exp_{\mathcal{G}}(q)$ all the conjuncts involving such nodes, without losing any answer to the original query. □

The following example illustrates the application of the expansion algorithm in a simple case.

Example 19. Consider Example 12. Suppose the user query is $q(X) \leftarrow \mathsf{person}(X, Y, Z)$. The expansion tree of q is shown in Fig. 2. Note that in the rightmost branch, stopping condition (2) is verified, since the goal $\mathsf{person}'(X, W_2, W_3)$ is subsumed by the goal $\mathsf{person}'(X, Y, Z)$, that labels the root. Therefore the evaluation stops, producing the empty clause □. The non-empty leaves, shaded in the figure, provide the following expansion $q' = exp_{\mathcal{G}}(q)$ of the query q:

$q'(X) \leftarrow \mathsf{person}(X, Y, Z)$

$q'(X) \leftarrow \mathsf{student}(X, W_1)$

$q'(X) \leftarrow \mathsf{city}(W_2, X)$

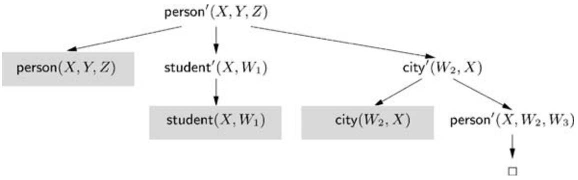

Fig. 2. Partial evaluation tree for the query of Example 12.

Intuitively, we see that the expanded query searches for codes of persons not only in the relation person, but also in student and city, where, due to the integrity constraints, it is known that codes of persons are stored.

Next we consider the unfolding step, which substitutes each atom of the (expanded) query with the body of the corresponding query in the mapping $\mathcal{M}$. As mentioned, this is a standard step in data integration [7,30] (which trivially terminates) and it is immediate to prove that it preserves soundness and completeness.

Theorem 20 (Soundness and completeness of unfolding). *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be an integration system, q a conjunctive query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$ such that $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, and t a tuple of constants of the same arity as q. Then $t \in q^{ret(\mathcal{I},\mathcal{D})}$ if and only if $t \in [unf_{\mathcal{M}}(q)]^{\mathcal{D}}$.*

Proof. Let $q = \alpha_1, \ldots, \alpha_{n_q}$. By definition of the mapping, $\alpha_i^{ret(\mathcal{I},\mathcal{D})} = [unf_{\mathcal{M}}(\alpha_i)]^{\mathcal{D}}$ for all i such that $1 \leqslant i \leqslant n_q$. Since for any tuple t we have $t \in \alpha_i^{ret(\mathcal{I},\mathcal{D})}$ if and only if $t \in [unf_{\mathcal{M}}(\alpha_i)]^{\mathcal{D}}$, the claim follows immediately. □

With soundness and completeness of both unfolding and expansion steps in place, we can state the main result of this section.

Theorem 21 (Soundness and Completeness). *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ be a data integration system, q a conjunctive query posed to $\mathcal{I}$, $\mathcal{D}$ a source database for $\mathcal{I}$ such that $\mathcal{I}$ is consistent w.r.t. $\mathcal{D}$, and t a tuple of constants of the same arity as q. Then $t \in q^{\mathcal{I},\mathcal{D}}$ if and only if $t \in [unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))]^{\mathcal{D}}$.*

Proof. By Theorem 20, $t \in [unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))]^{\mathcal{D}}$ if and only if $t \in exp_{\mathcal{G}}(q)^{ret(\mathcal{I},\mathcal{D})}$. Applying Theorems 15 and 18, we have that $t \in exp_{\mathcal{G}}(q)^{ret(\mathcal{I},\mathcal{D})}$ if and only if

$t \in q^{can(\mathcal{I},\mathcal{D})}$. By Theorem 10, $t \in q^{can(\mathcal{I},\mathcal{D})}$ if and only if $t \in q^{\mathcal{I},\mathcal{D}}$. This proves the claim. □

6. Discussion and related work

We start this section by discussing the computational complexity of our algorithm. We distinguish between combined and data complexity. We remind the reader that we use n_q to denote the number of atoms in the query q, n_c the number of foreign key constraints in $\mathcal{G}$, and w the maximum number of attributes of the key of any relation in $\mathcal{G}$.

The combined complexity is the complexity of the algorithm with respect to the size of all parameters characterizing the problem, namely, the size of the query q, the size of the global schema $\mathcal{G}$, and the size of the source database $\mathcal{D}$. It is easy to see that, under this measure, the cost of the algorithm is given by the sum of the cost of computing $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$, and the cost of evaluating $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ over the source database $\mathcal{D}$. The cost of computing $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ is directly related to the number of nodes of the SLD-tree computed by our expansion algorithm, i.e., $O(k^{n_q\cdot(n_q\cdot n_c\cdot(w+1)^w+1)})$, where k is one plus the maximum number of foreign key constraints for a relation in $\mathcal{G}$. It follows that the worst-case time complexity of computing $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ is $O(k^{n_q\cdot(n_q\cdot n_c\cdot(w+1)^w+1)})$. The cost of evaluating $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ over the source database $\mathcal{D}$ depends on both the kind of queries associated by $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ to the elements of $\mathcal{G}$, and the form of $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$. With regard to the first aspect, as mentioned in Section 2, we assume that they can be evaluated in polynomial time data complexity. With regard to the second aspect, starting from the observation that $exp_{\mathcal{G}}(q)$ is a finite union of conjunctive queries, and that the evaluation of a union of conjunctive queries can be done in time polynomial with respect to the size of the database, it is easy to see that $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ can be evaluated over the source database $\mathcal{D}$ in time polynomial in the size of $\mathcal{D}$.

Data complexity refers to the complexity of the algorithm with respect to the size of the source database $\mathcal{D}$ only. Observe that, since the size of $\mathcal{D}$ largely dominates the size of the other parameters of the problem, data complexity is the most important criterion for measuring the complexity of our method. As we said before, the cost of computing $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ is independent of the size of $\mathcal{D}$. Thus, the data complexity of our algorithm is solely determined by the complexity of evaluating $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ over the source database $\mathcal{D}$. It follows that our query answering algorithm is polynomial with respect to data complexity. Note also that, although the number of queries in $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$ to be evaluated over $\mathcal{D}$ is exponential with respect to the size of the original query q, the size of each of such queries is at most quadratic with respect to the size of q and $\mathcal{M}_{\mathcal{G},\mathcal{S}}$. In fact, each conjunctive query in $\exp_{\mathcal{G}}(q)$ has the same number of conjuncts as q, and each such conjunct is unfolded according to $\mathcal{M}_{\mathcal{G},\mathcal{S}}$.

The results presented in this paper are related to the work by Johnson and Klug on containment of conjunctive queries under functional and inclusion dependencies [33]. Indeed, we exploited one of their results in order to devise a bound on the depth of the expansion tree. The goal of [33] was to show that testing query containment under functional and inclusion dependencies is PSPACE-complete in several interesting cases. The upper bound result is obtained by exhibiting a nondeterministic PSPACE algorithm performing the test, and relying on the observation that NPSPACE is equal to PSPACE. The first thing to note is that the class of constraints considered in this paper is incomparable with the one studied in [33]. In particular, our foreign key constraints do not fall in the class considered in [33]. Despite this fact, the results by Johnson and Klug can in principle be used to show that query answering in our setting is also in PSPACE. However, the application of the method illustrated in [33] to our context would result in an algorithm that would perform an exhaustive search in the relevant finite portion of the canonical database. While such an exhaustive search is not too problematic in the context of query containment (the size of queries being probably small), it becomes too expensive in our setting, because it corresponds to a large number of accesses to the source database $\mathcal{D}$, one for each guess of the nondeterministic algorithm. On the contrary, our method is based on the idea of letting the query q and the corresponding unification with the foreign key rules guide such search, even at the price of an exponential (with respect to the query) worst-case blow-up in the space used during the construction of $unf_{\mathcal{M}}(exp_{\mathcal{G}}(q))$.

7. Conclusions

While it is a common opinion that query processing is an easy task in the global-centric approach to data integration, we have shown the surprising result that, when the global schema contains integrity constraints, even of simple forms, query processing becomes more difficult. The difficulties basically arise because of the need of dealing with incomplete information, similarly to the case of the source-centric approach to data integration. We have studied the case of global schemas expressed in the relational model with key and foreign key constraints, and we have presented techniques for effectively answering queries posed to the data integration system in this case. The method described in this paper can be extended in several ways. First, it is immediate to verify that the technique can be easily adapted to deal with the case of unions of conjunctive queries. Second, the method is still valid if we consider a more expressive version of foreign key constraints. Finally, we are working on several optimization strategies in the construction of the expansion tree.

References

[1] C. Batini, M. Lenzerini, S.B. Navathe, A comparative analysis of methodologies for database schema integration, ACM Comput. Surveys 18 (4) (1986) 323–364.

[2] A.P. Sheth, J.A. Larson, Federated database systems for managing distributed, heterogeneous, and autonomous databases, ACM Comput. Surveys 22 (3) (1990) 183–236.

[3] G. Thomas, G.R. Thompson, C.-W. Chung, E. Barkmeyer, F. Carter, M. Templeton, S. Fox, B. Hartman, Heterogeneous distributed database systems for production use, ACM Comput. Surveys 22 (3) (1990) 237–266.

[4] W. Litwin, L. Mark, N. Roussopoulos, Interoperability of multiple autonomous databases, ACM Comput. Surveys 22 (3) (1990) 267–293.

[5] T. Catarci, M. Lenzerini, Representing and using interschema knowledge in cooperative information systems, J. Intell. Cooperative Inform. Systems 2 (4) (1993) 375–398.

[6] R. Hull, Managing semantic heterogeneity in databases: A theoretical perspective, in: Proceedings of the 16th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'97), Tucson, AZ, USA, 1997, pp. 51–61.

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

[8] J.D. Ullman, Information integration using logical views, in: Proceedings of the 6th International Conference on Database Theory (ICDT'97), Delphi, Greece, Lecture Notes in Computer Science, Vol. 1186, Springer, Berlin, 1997, pp. 19–40.

[9] C. Li, E. Chang, Query planning with limited source capabilities, in: Proceedings of the 16th IEEE International Conference on Data Engineering (ICDE 2000), San Diego, CA, USA, 2000, pp. 401–412.

[10] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullman, V. Vassalos, J. Widom, The TSIMMIS approach to mediation: data models and languages, J. Intell. Inform. Systems 8 (2) (1997) 117–132.

[11] A. Tomasic, L. Raschid, P. Valduriez, Scaling access to heterogeneous data sources with DISCO, IEEE Trans. Knowledge Data Eng. 10 (5) (1998) 808–823.

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

[13] T. Kirk, A.Y. Levy, Y. Sagiv, D. Srivastava, The information manifold, in: Proceedings of the AAAI 1995 Spring Symposium on Information Gathering from Heterogeneous, Distributed Enviroments, Stanford, CA, USA, 1995, pp. 85–91.

[14] S. Abiteboul, O. Duschka, Complexity of answering queries using materialized views, in: Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98), Seattle, WA, USA, 1998, pp. 254–265.

[15] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Data integration in data warehousing, Int. J. Cooperative Inform. Systems 10 (3) (2001) 237–271.

[16] A. Calì, G. De Giacomo, M. Lenzerini, Models for information integration: turning local-as-view into global-as-view, in: Proceedings of International Workshop on Foundations of Models for Information Integration (10th Workshop in the series Foundations of Models and Languages for Data and Objects), Viterbo, Italy, 2001.

[17] J. Gryz, Query folding with inclusion dependencies, in: Proceedings of the 14th IEEE International Conference on Data Engineering (ICDE'98), Orlando, FL, USA, 1998, pp. 126–133.

[18] G. Grahne, A.O. Mendelzon, Tableau techniques for querying information sources through global schemas,

in: Proceedings of the 7th International Conference on Database Theory (ICDT'99), Jerusalem, Israel, Lecture Notes in Computer Science, Vol. 1540, Springer, Berlin, 1999, pp. 332–347.

[19] D. Calvanese, G. De Giacomo, M. Lenzerini, M.Y. Vardi, Query processing using views for regular path queries with inverse, in: Proceedings of the 19th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2000), Dallas, TX, USA, 2000, pp. 58–66.

[20] R. van der Meyden, Logical approaches to incomplete information, in: J. Chomicki, G. Saake (Eds.), Logics for Databases and Information Systems, Kluwer Academic Publisher, Dordrecht, 1998, pp. 307–356.

[21] M.F. Fernandez, D. Florescu, A. Levy, D. Suciu, Verifying integrity constraints on web-sites, in: Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99), Stockholm Sweden, 1999, pp. 614–619.

[22] M.F. Fernandez, D. Florescu, J. Kang, A.Y. Levy, D. Suciu, Catching the boat with Strudel: experiences with a web-site management system, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, WA, USA, 1998, pp. 414–425.

[23] M. Arenas, L.E. Bertossi, J. Chomicki, Consistent query answers in inconsistent databases, in: Proceedings of the 18th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'99), Philadelphia, PA, USA, 1999, pp. 68–79.

[24] R. Fagin, P.G. Kolaitis, R.J. Miller, L. Popa, Data exchange: semantics and query answering, in: Proceedings of the 9th International Conference on Database Theory (ICDT 2003), Siena, Italy, 2003, pp. 207–224.

[25] L. Popa, Y. Velegrakis, R.J. Miller, M.A. Hernández, R. Fagin, Translating Web data, in: Proceedings of the 28th International Conference on Very Large Data Bases (VLDB 2002), Hong Kong, China, 2002, pp. 598–609.

[26] R. Kowalski, F. Sadri, P. Soper, Integrity checking in deductive databases, in: Proceedings of the 13th International Conference on Very Large Data Bases (VLDB'87), Brighton, UK, 1987, pp. 61–69.

[27] J. Grant, J. Gryz, J. Minker, L. Raschid, Semantic query optimization for object databases, in: Proceedings of the 13th IEEE International Conference on Data Engineering (ICDE'97), Birmingham, UK, 1997, pp. 444–453.

[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, E.L. Wimmers, Towards heterogeneous multimedia information systems: the Garlic approach, in: Proceedings of the Fifth International Workshop on Research Issues in Data Engineering—Distributed Object Management (RIDE-DOM'95), Taipei, Taiwan, IEEE Computer Society Press, Silverspring, MD, 1995, pp. 124–131.

[29] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J.D. Ullman, M. Valiveti, Capability based mediation in TSIMMIS, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, Seattle, WA, USA, 1998, pp. 564–566.

[30] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of the 21st ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002), Madison, WI, USA, 2002, pp. 233–246.

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

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

[33] D.S. Johnson, A.C. Klug, Testing containment of conjunctive queries under functional and inclusion dependencies, J. Comput. System Sci. 28 (1) (1984) 167–189.

[34] J.W. Lloyd, Foundations of Logic Programming, 2nd extended Edition, Springer, Berlin, Heidelberg, 1987.

[35] J.W. Lloyd, J.C. Shepherdson, Partial evaluation in logic programming, J. Logic Programming 11 (1991) 217–242.