Accessing Data Integration Systems through Conceptual Schemas

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

Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Via Salaria 113, I-00198 Roma, Italy
lastname@dis.uniroma1.it,
http://www.dis.uniroma1.it/∼*lastname*

Abstract. Data integration systems provide access to a set of heterogeneous, autonomous data sources through a so-called global, or mediated view. There is a general consensus that the best way to describe the global view is through a conceptual data model, and that there are basically two approaches for designing a data integration system. In the global-as-view approach, one defines the concepts in 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 terms of a conceptual data model, even a very simple one, query processing becomes difficult in the global-as-view approach also. We demonstrate that the problem of incomplete information arises in this case too, and we illustrate some basic techniques for effectively answering queries posed to the global schema of the data integration system.

1 Introduction

Data 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 (or, mediated) schema [15, 16]. The global schema is therefore a reconciled view of the information, which can be queried by the user. It is the task of the data integration system to free the user 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 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. Here, we concentrate on the following issues: *(i)* dealing with heterogeneity of the sources, *(ii)* specifying the mapping between the global schema and the sources, *(iii)* processing queries expressed on the global schema.

Issue *(i)* refers to the fact that typically sources adopt different ontologies, models, and systems for storing data. This poses challenging problems in specifying 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 schema should mediate among different representations of overlapping worlds, the language should provide flexible and powerful representation mechanisms. This is the reason why many authors advocate the use of a conceptual data model for expressing the global schema [5, 21, 22, 3]. In this paper we follow this idea, and investigate the problem of query answering in data integration systems where the global schema is expressed in terms of an extended Entity-Relationship Model.

With regard to issue *(ii)*, two basic approaches have been used to specify the mapping between the sources and the global schema [15, 17, 18]. The first approach, called *global-as-view* (also global-schema centric, or simply global-centric), 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 *local-as-view* (or source-centric), requires the global schema to be specified independently from the sources. In turn, the sources are defined as views over the global schema. A comparison of the approaches is reported in [24]. In this paper, we concentrate on the latter approach, which is generally considered sufficiently simple and effective for practical purposes.

Finally, issue *(iii)* 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. For this purpose, the system should be able to re-express the query in terms of a suitable set of queries posed to the sources. In this reformulation process, the crucial step is deciding how to decompose the query on the global schema into a set of subqueries on the sources, based on the meaning of the mapping. The computed subqueries 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 [23, 24, 14, 1, 13, 7, 8]. 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 [25]. On the other hand, query processing looks 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 [15].

While this is a common opinion in the literature, we show that our framework poses new challenges, specially related to the need of taking the semantics of the conceptual global schema into account during query processing. Indeed, the first contribution in this paper is to show that the idea of adopting a conceptual data model for expressing the global schema, makes query processing more involved than in the simplified framework usually considered in the literature. In particular, we present the surprising result that the semantics of a 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 second contribution of the paper is the formalization of the notion of correct answer in a data integration system with a conceptual global schema, and the presentation of a query processing strategy that is able to provide all correct answers to a query posed to the system.

The paper is organized as follows. In Section 2 we describe the conceptual data model we use in our approach. Section 3 illustrates a formal framework for data integration, by describing the main components of a data integration system, namely, the global schema, the sources, and the mapping between the two, and by specifying the precise semantics of the system. In Section 4 we present our query processing algorithm. By reasoning on both the query and the conceptual global schema, the algorithm is able to compute all correct answers to a query posed to the global schema. Section 5 concludes the paper.

2 The Conceptual Data Model

We present the conceptual model which is at the basis of the integration framework introduced in the next section. The model incorporates the basic features of the *Entity-Relationship* (ER) model [10], extended with subset (or is-a) constraints on both entities and relationships. Other characteristics that are not considered in this paper for the sake of simplicity (e.g., domain of attributes, identification constraints, etc.), can also be added without affecting the results in the next sections.

An *ER schema* is a collection of entity, relationship, and attribute definitions over an *alphabet $\mathcal{A}$ of symbols*. The alphabet $\mathcal{A}$ is partitioned into a set of entity symbols (denoted by E), a set of relationship symbols (denoted by R), and a set of attribute symbols (denoted by A).

An *entity definition* has the form

define entity E
  isa: $E_1, \ldots, E_h$
  participates in: $R_1 : c_1, \ldots, R_\ell : c_\ell$
end.

where E is the entity to be defined, the **isa** clause specifies a set of entities to which E is related via is-a (i.e., the set of entities that are supersets of E), and the **participates in** clause specifies those relationships, with respective components, to which an instance of E must necessarily participate. A *relationship definition* has the form

> **define relationship** R **among** $E_1, \ldots, E_n$
> **isa**: $R_1, \ldots, R_h$
> **end**.

where R is the relationship to be defined, the entities listed in the **among** clause are those among which the relationship is defined (i.e., component i of R is an instance of entity E_i), and the **isa** clause specifies a set of relationships to which R is related via is-a. The number of entities in the **among** clause is the *arity* of R. An *attribute definition* has the form

> **define attribute** A **for** X
> *qualification*
> **end**.

where A is the attribute to be defined, X is the entity or relationship to which the attribute is associated, and *qualification* consists of none, one, or both of the keywords **functional** and **mandatory**, specifying respectively that each instance of X has a unique value for attribute A, and that each instance of X must have a value for attribute A. If the **functional** keyword is missing, the attribute is multivalued, and if the **mandatory** keyword is missing, the attribute is optional.

In the definition of an entity or a relationship, the **isa** clause may be missing. Similarly, in an entity definition, the **participates in** clause may be missing. On the contrary, the **among** clause in a relationship definition must be present.

Notice that in our model each attribute is associated to a unique entity or relationship, i.e., different entities and relationships have disjoint sets of attributes. Also, for the sake of simplicity, we do not consider the specification of the domains of attributes in our model, and we simply assume that attributes have atomic values.

The semantics of an ER schema is defined by specifying when a database satisfies all constraints imposed by the schema. Formally, a database $\mathcal{B}$ is defined over a fixed (infinite) alphabet Γ of symbols, each one denoting a semantic value. $\mathcal{B}$ assigns to each entity a subset of Γ, to each attribute A of an entity a binary relation over Γ, to each relationship R of arity n, a set of n-tuples of elements of Γ, and to each attribute A of a relationship of arity n an $(n+1)$-ary relation over Γ. The set of objects assigned by $\mathcal{B}$ to an entity, attribute, or relationship is called the set of its *instances* in $\mathcal{B}$. We say that $\mathcal{B}$ is *legal* with respect to an ER schema $\mathcal{G}$ if the following conditions are satisfied:

- For each entity definition as the one above, the set of instances of E is a subset of the sets of instances of $E_1, \ldots, E_h$, and for each pair $R_i : c_i$ in the **participates in** clause of the definition, we have that each instance of E appears as c_i-th component is some instance of R_i.

- For each attribute specification as the one above, where X is an entity, we have that for each instance of A the first component is an instance of X. Moreover, if the *qualification* contains the keyword **mandatory**, then each instance of X must appear as the first component in some instance of A, and if it contains the keyword **functional**, then there may be no two instances of A coinciding on the first component (and differing on the second component). Similar conditions hold for the case where X is a relationship.
- For each relationship specification as the one above, for each instance $(o_1, \ldots, o_n)$ of R we have that o_i is an instance of E_i. Moreover, the set of instances of R is a subset of the sets of instances of $R_1, \ldots, R_h$.

The language we use to express queries over a global schema expressed in our conceptual model is that of conjunctive queries. Formally, a *conjunctive query* (CQ) Q of arity n is written in the form

$$Q(x_1, \ldots, x_n) \leftarrow conj(x_1, \ldots, x_n, y_1, \ldots, y_m)$$

where $conj(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a conjunction of atoms involving constants of Γ and variables $x_1, \ldots, x_n, y_1, \ldots, y_m$ from an alphabet of variables. The predicates in the atoms are the so-called *concepts* of the conceptual schema, i.e., its entities, relationships and attributes:

- Each entity E in $\mathcal{G}$ has an associated predicate E of arity 1. Intuitively, $E(c)$ asserts that c is an instance of entity E.
- Each attribute A for an entity E has an associated predicate A of arity 2. Intuitively, $A(c, d)$ asserts that c is an instance of entity E and d is the value of attribute A associated to c.
- Each relationship R among the entities $E_1, \ldots, E_n$ has an associated predicate R of arity n.
- Each attribute A for a relationship R among the entities $E_1, \ldots, E_n$ has an associated predicate A of arity $n+1$. Intuitively, $A(c_1, \ldots, c_n, d)$ asserts that $(c_1, \ldots, c_n)$ is an instance of relationship R and d is the value of attribute A associated to $(c_1, \ldots, c_n)$.

From a semantic point of view, the extension of a predicate P in a database $\mathcal{B}$ coincides with the set of instances that the concept associated to P has in $\mathcal{B}$. With this consideration, we can turn our attention to the semantics of queries, and simply observe that the semantics of conjunctive queries is the usual one, where the variables in the body are existentially quantified [2]. Thus, the *answer set* $Q^{\mathcal{B}}$ *of* Q *over a database* $\mathcal{B}$ is the set of tuples $(c_1, \ldots, c_n)$ of $\mathcal{B}$ for which there are $d_1, \ldots, d_m$ in $\mathcal{B}$, such that for each atom $e(b_1, \ldots, b_k)$ in $conj(c_1, \ldots, c_n, d_1, \ldots, d_m)$, where each b_i is one of $c_1, \ldots, c_n, d_1, \ldots, d_m$, we have that $(b_1, \ldots, b_k) \in e^{\mathcal{B}}$.

Example 1. Consider the ER schema shown in Figure 1, depicted in the usual graphical notation for the ER model. The elements of such a schema are Person/1, Employee/1, City/1, Lives_In/2, pname/2, salary/2, cname/2, since/2.

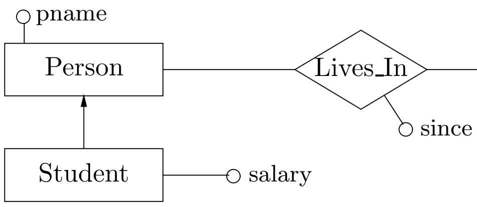

Fig. 1. ER schema of Example 1

Suppose we want to know the names of the employees who live in Rome since 1997. The corresponding CQ is

$$\begin{aligned}\mathsf{q}(y) \leftarrow\ & \mathsf{Employee}(x), \mathsf{pname}(x, y), \mathsf{Lives\_In}(x, z), \mathsf{since}(x, z, 1997),\\ & \mathsf{cname}(z, \text{“}\mathit{Rome}\text{”})\end{aligned}$$

The answer of q over the database shown in Figure 2, is *Ann*.

3 The Formal Framework for Data Integration

In this section we set up a formal framework for data integration. In particular, we describe the main components of a data integration system, namely, the global schema, the sources, and the mapping between the two. Finally, we provide the semantics both of the system, and of query answering.

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

We describe the characteristics of the various components of a data integration system in our approach:

– The *global schema* $\mathcal{G}$ is expressed in the conceptual data model described in the previous section.
– The *source schema* $\mathcal{S}$ is constituted by the schemas of the source relations. Note that we assume that the sources are expressed as relational data bases.

Person

p1
p2

Employee

p1

City

c1

Lives_In

p1	*c1*
p2	*c2*

pname

p1	*Jeremy*
p2	*Ann*

salary

p1	80,000

cname

c1	*Caracas*
c2	*Rome*
c3	*Yokohama*

since

p1	*c1*	1982
p2	*c2*	1997

Fig. 2. Extension of relations of Example 1

This is not a strong limitation, since, in case of sources of different type, we can assume that suitable wrappers present the data at the source in relational form. Moreover, we observe that all considerations presented in this paper still hold in the case of other types of sources (e.g., semistructured sources).
– The *mapping* $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ between $\mathcal{G}$ and $\mathcal{S}$ is given by associating to each concept C (either entity, relationship, or attribute) in the global schema a query $\mathcal{V}_C$ over the sources. We do not pose any constraint on the language used to express the queries in the mapping. Since sources are relational databases, we simply assume that the language is able to express computations over relational databases. Note that the *elements* that are assigned a query over the sources by the mapping coincide with the concepts of the global schema.

More precisely, the mapping associates queries to the elements of $\mathcal{G}$ as follows:

– The mapping associates a query of arity 1 to each entity of $\mathcal{G}$.
– The mapping associates a query of arity 2 to each attribute A defined for an entity in $\mathcal{G}$. Intuitively, if the query retrieves (c, d) from the sources, this means that d is a value of the attribute A of the entity instance c.
– The mapping associates a query of arity n to each relationship R of arity n in $\mathcal{G}$. Intuitively, if the query retrieves the tuple $(c_1, \ldots, c_n)$ from the sources, this means that $(c_1, \ldots, c_n)$ is an instance of R.
– The mapping associates a query of arity $n+1$ to each attribute A defined for a relationship R of arity n in $\mathcal{G}$. Intuitively, if the query retrieves $(c_1, \ldots, c_n, d)$ from the sources, this means that d is a value of the attribute A of the relationship instance $(c_1, \ldots, c_n)$.

As specified above, the intended meaning of the query $\mathcal{V}_C$ associated to the concept C is that it specifies how to retrieve the data corresponding to C in the global schema starting from the data at the sources. This confirms that we are following the global-as-views approach: the concepts in the global schema are defined as views over the source data.

Notice that all considerations reported in this paper still hold if we choose a different set of elements for specifying the mapping. For example, we could associate a single query of arity $m+1$ to each entity with m attributes (similarly for the relationships).

In order to specify the semantics of a data integration system, we have to characterize, given the set of tuples satisfying the various source relations, which are the data satisfying the global schema. In principle, given a set of data at the sources, one would like to have a corresponding single database for the global schema. Indeed, this is the case for most of the data integration systems described in the literature. However, we will show in the following the surprising result that, due to the presence of the semantic conditions that are implicit in the conceptual schema $\mathcal{G}$, in general, we will have to account for a set of databases.

We remind the reader that we assume that the databases involved in our framework (both global databases, and source databases) are defined over a fixed (infinite) alphabet Γ of symbols. In order to assign semantics to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$, we start by considering a *source database*

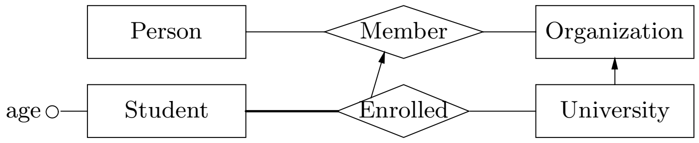

Fig. 3. Global schema of Example 2

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 *global database* for $\mathcal{I}$ any database for $\mathcal{G}$. A global database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal* with respect to $\mathcal{D}$, or, simply, *legal for $\mathcal{I}$ with respect to $\mathcal{D}$*, if:

– $\mathcal{B}$ is legal with respect to $\mathcal{G}$,
– for each element e of $\mathcal{G}$, the set of tuples $e^{\mathcal{B}}$ that $\mathcal{B}$ assigns to e is coherent with set of tuples computed by the associated query $\mathcal{V}_e$ over $\mathcal{D}$, i.e., $\mathcal{V}_e^{\mathcal{D}} \subseteq e^{\mathcal{B}}$.

The above definition implies that sources are considered *sound*: the data they provide to the integration system satisfy the global schema, but are not necessarily complete [13]. Another possibility would be to consider them *exact*. When sources are exact, the mapping between the global schema and the sources is defined in such a way that, for every source database $\mathcal{D}$, and every element e of $\mathcal{G}$, it holds that $\mathcal{V}_e^{\mathcal{D}} = e^{\mathcal{B}}$.

Example 2. Figure 3 shows the global schema $\mathcal{G}_1$ of a data integration system $\mathcal{I}_1 = \langle \mathcal{G}_1, \mathcal{S}_1, M_1 \rangle$, where *age* is a functional attribute, Student has a mandatory participation in the relationship Enrolled, Enrolled is-a Member, and University is-a Organization. The schema models persons who can be members of one or more organizations, and students who are enrolled in universities. Suppose that $\mathcal{S}_1$ is constituted by $\mathsf{s}_1, \mathsf{s}_2, \mathsf{s}_3, \mathsf{s}_4, \mathsf{s}_5, \mathsf{s}_6, \mathsf{s}_7, \mathsf{s}_8$, and that the mapping $\mathcal{M}_1$ is as follows:

$$\begin{array}{rlrl}
\mathsf{Person}(x) & \leftarrow \mathsf{s}_1(x) & \mathsf{Student}(x) & \leftarrow \mathsf{s}_3(x,y) \vee \mathsf{s}_4(x,z) \\
\mathsf{Organization}(x) & \leftarrow \mathsf{s}_2(x) & \mathsf{University}(x) & \leftarrow \mathsf{s}_5(x) \\
\mathsf{Member}(x,y) & \leftarrow \mathsf{s}_7(x,z), \mathsf{s}_8(z,y) & \mathsf{Enrolled}(x,y) & \leftarrow \mathsf{s}_4(x,y) \\
 & & \mathsf{age}(x,y) & \leftarrow \mathsf{s}_3(x,y) \vee \mathsf{s}_6(x,y,z)
\end{array}$$

We are now ready to provide the definition of the semantics of a data integration system in our formalization.

Definition 2. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a data integration system, and let $\mathcal{D}$ be a source database for $\mathcal{I}$. The* semantics *of $\mathcal{I}$ with respect to $\mathcal{D}$, denoted $sem(\mathcal{I}, \mathcal{D})$, is the set of global databases that are legal for $\mathcal{I}$ with respect to $\mathcal{D}$.*

From the above definition it is easy to see that, in our framework, given a source database $\mathcal{D}$, different situations are possible:

1. No legal global database exists. This happens, in particular, when the data at the sources retrieved by the queries associated to the elements of the global schema do not satisfy the functional attribute constraints.

Example 3. Referring to Example 2, consider a source database $\mathcal{D}_1$, where s_3 stores the tuple (t_1, a_1), and s_6 stores the tuple (t_1, a_2, v_1). The query associated to age by the mapping $\mathcal{M}_1$ specifies that, in every legal database of $\mathcal{I}_1$ both tuples should belong to the extension of age. However, age is a functional attribute in $\mathcal{G}_1$, and therefore no legal database exists for the data integration system $\mathcal{I}_1$.

2. Several legal global databases exist. This happens, for example, when the data at the sources retrieved by the queries associated to the global relations do not satisfy the is-a relationships of the global schema. In this case, it may happen that several ways exist to add suitable objects to the elements of $\mathcal{G}$ in order to satisfy the constraints. Each such ways yields a legal global database.

Example 4. Referring again to Example 2, consider a source database $\mathcal{D}_2$, where s_1 stores p_1 and p_2, s_2 stores o_1, s_5 stores u_1, and s_4 stores t_1, and the pairs (p_1, o_1) and (p_2, u_1) are in the join between s_7 and s_8. By the mapping $\mathcal{M}_1$, it follows that in every legal database of $\mathcal{I}_1$, $p_1, p_2 \in \mathsf{Person}$, $(p_1, o_1), (p_2, u_1) \in \mathsf{Member}$, $o_1 \in \mathsf{Organization}$, $t_1 \in \mathsf{Student}$, $u_1 \in \mathsf{University}$. Moreover, since $\mathcal{G}_1$ specifies that $\mathsf{Student}$ has a mandatory participation in the relationship $\mathsf{Enrolled}$, in every legal database for $\mathcal{I}_1$, t_1 *must* be enrolled in a certain university. The key point is that nothing is said in $\mathcal{D}_2$ about *which* university, and therefore we have to accept as legal all databases for $\mathcal{I}_1$ that differ in the university in which t_1 is enrolled.

In our framework, we assume that the first problem is solved by the queries extracting data at the sources. In other words, we assume that, for any functional attribute A, the corresponding query implements a suitable data cleaning strategy that ensures that, for every source database $\mathcal{D}$ and every x, at most one tuple (x, y) belongs to $\mathcal{V}_A^{\mathcal{D}}$ (similar condition holds for functional attributes of relationships). The interested reader is referred to [12] for more details of data cleaning techniques.

The second problem shows that the issue of query answering with incomplete information arises even in the global-as-view approach to data integration. Indeed, the existence of multiple global databases for the data integration system implies that query processing cannot simply reduce to evaluating the query over a single database. Rather, we should in principles take *all* possible legal global databases into account when answering a query.

It is interesting to observe that there are at least two different strategies to simplify the setting, and overcome this problem:

1. Data integration systems usually adopt a simpler data model (often, a plain relational data model) for expressing the global schema. In this case, the data retrieved from the sources trivially fits into the schema, and can be directly considered as the unique database to be processed during query answering.
2. The queries associated to the elements of the global schema are often considered as exact. In this case, analogously to the previous one, it is easy

to see that the only global database to be considered is the one formed by the data retrieved by the source. However, when data at the sources do not obey all semantic conditions that are implicit in the conceptual global schema, this single database is not coherent with the global schema, and the data integration system is inconsistent. This implies that query answering is meaningless. We argue that, in the usual case of autonomous, heterogeneous sources, it is very unlikely that data fit in the global schema, and therefore, this approach is too restrictive, in the sense that the data integration system would be often inconsistent.

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 above mentioned simplifying strategies: they either express the global schema as a set of plain relations, or consider the sources as exact (see, for instance, [9, 19, 4]). On the contrary, the goal of our work is to study the more general setting where the global schema is expressed in terms of a conceptual model, 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, and the results described in Section 4 show how to process queries in this setting.

We conclude the section by defining the notion of query posed to the data integration system. A query Q to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ is a conjunctive query, whose atoms have symbols in $\mathcal{G}$ as predicates, as illustrated in Section 2. Our goal here is to specify which are the tuples that form the answer to a query posed to a data integration system $\mathcal{I}$. The fact that, given a source database $\mathcal{D}$, several global databases may exist that are legal for $\mathcal{I}$ with respect to $\mathcal{D}$ complicates this task. In order to address this problem, we follow a first-order logic approach: a tuple $(c_1, \ldots, c_n)$ 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 3. *Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be a data integration system, let $\mathcal{D}$ be a source database for $\mathcal{I}$, and let Q be a query of arity n to $\mathcal{I}$. The* set of certain answers *$Q^{\mathcal{I},\mathcal{D}}$ to Q with respect to $\mathcal{I}$ and $\mathcal{D}$ is the set of tuples $(c_1, \ldots, c_n)$ such that $(c_1, \ldots, c_n) \in Q^{\mathcal{B}}$, for each $\mathcal{B} \in sem(\mathcal{I}, \mathcal{D})$.*

Example 5. Referring to Example 4, consider the query Q_1 to $\mathcal{I}_1$:

$$\mathsf{Q}_1(x) \leftarrow \mathsf{Member}(x, y), \mathsf{University}(y)$$

It is easy to see that $\{p_2, t_1\}$ is the set of certain answers to Q_1 with respect to $\mathcal{I}_1$ and $\mathcal{D}_2$. Thus, although $\mathcal{D}_2$ does not indicate which university t_1 is enrolled in, the semantics of $\mathcal{I}_1$ specifies that t_1 is enrolled in *a* university in all legal database for $\mathcal{I}_1$. Since Member is a generalization of Enrolled, this implies that t_1 is in the set of certain answers to Q_1 with respect to $\mathcal{I}_1$ and $\mathcal{D}_2$.

4 Answering Queries over the Global Schema

In this section we present an algorithm for computing the set of certain answers to queries posed to a data integration system. The key feature of the algorithm is to reason about both the query and the conceptual global schema in order to infer which tuples satisfy the query in all legal databases of the data integration system. Thus, the algorithm does not simply unfold the query on the basis of the mapping, as usually done in data integration systems based on the global-as-view approach. Indeed, we now show that a simple unfolding strategy does not work in our setting.

Example 6. Consider again Example 5, and suppose we simply unfold the query Q_1 in the standard way, by substituting each atom with the query that $\mathcal{M}_1$ associates to the element in the atom. Then we get the query

$$\mathsf{q}(x) \leftarrow \mathsf{s}_7(x,z), \mathsf{s}_8(z,y), \mathsf{s}_5(y)$$

If we evaluate this query over $\mathcal{D}_2$, we get $\{p_2\}$ as result, thus missing the certain answer t_1.

Next we illustrate our algorithm for computing all certain answers. The algorithm is able to add more answers to those directly extracted from the sources, by exploiting the semantic conditions expressed in the conceptual global schema.

Let $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M}_{\mathcal{G},\mathcal{S}} \rangle$ be an integration system, let $\mathcal{D}$ be a source database, and let Q be a query over the global schema $\mathcal{G}$. The algorithm is constituted by three major steps.

1. From the query Q, obtain a new query $\mathit{exp}_{\mathcal{G}}(Q)$ over the elements of the global schema G in which the knowledge in $\mathcal{G}$ that is relevant for Q has been compiled in.
2. From $\mathit{exp}_{\mathcal{G}}(Q)$, compute the query $\mathit{unf}_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(\mathit{exp}_{\mathcal{G}}(Q))$, by unfolding $\mathit{exp}_{\mathcal{G}}(Q)$ on the basis of the mapping $\mathcal{M}_{\mathcal{G},\mathcal{S}}$. The unfolding simply substitutes each atom of $\mathit{exp}_{\mathcal{G}}(Q)$ with the query associated by $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ to the element in the atom. The resulting $\mathit{unf}_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(\mathit{exp}_{\mathcal{G}}(Q))$ is a query over the source relations.
3. Evaluate the query $\mathit{unf}_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(\mathit{exp}_{\mathcal{G}}(Q))$ over the source database $\mathcal{D}$.

The last two steps are quite obvious. Instead, the first one requires to find a way to compile into the query the semantic relations holding among the concepts of the global schema $\mathcal{G}$. Such semantic relations can indeed be crucial for inferring the complete set of certain answers.

The basic idea to do so is that the relations among the elements in $\mathcal{G}$ can be captured by a suitable *rule base* $\mathcal{R}_{\mathcal{G}}$. To build $\mathcal{R}_{\mathcal{G}}$, we introduce a new predicate P' (called primed predicate) for each predicate P associated to an element P of $\mathcal{G}$. Then, from the semantics of the ER schema we devise the following rules (expressed in Logic Programming notation [20]):

– for each entity E, attribute A and relationship R in $\mathcal{G}$, we have:

$$\begin{array}{rcl} E'(x) & \leftarrow & E(x) \\ A'(x,y) & \leftarrow & A(x,y) \\ R'(x_1,\ldots,x_n) & \leftarrow & R(x_1,\ldots,x_n) \end{array}$$

– for each is-a relation between E and E_i, or between R and R_i in an entity or relationship definition of $\mathcal{G}$, we have:

$$\begin{array}{l} E'_i(x) \leftarrow E'(x) \\ R'_i(x_1,\ldots,x_n) \leftarrow R'(x_1,\ldots,x_n) \end{array}$$

– for each attribute A for an entity E or a relationship R in an attribute definition in $\mathcal{G}$, we have:

$$\begin{array}{l} E'(x) \leftarrow A'(x,y) \\ R'(x_1,\ldots,x_n) \leftarrow A'(x_1,\ldots,x_n,y) \end{array}$$

– for each relationship R involving an entity E_i as i-th component according to the corresponding relationship definition in $\mathcal{G}$, we have:

$$E'_i(x_i) \leftarrow R'(x_1,\ldots,x_i,\ldots,x_n)$$

– for each mandatory participation of an entity E in a relationship R_j in an entity definition of $\mathcal{G}$, we have:

$$R'_j(f_1(x),\ldots,x,\ldots,f_n(x)) \leftarrow E'(x)$$

where f_i are fresh Skolem functions [20].

– for each mandatory attribute A for an entity E or a relationship R in an attribute definition of $\mathcal{G}$, we have:

$$\begin{array}{l} A'(x,f(x)) \leftarrow E'(x) \\ A'(x_1,\ldots,x_n,f(x)) \leftarrow R'(x_1,\ldots,x_n) \end{array}$$

where f is a fresh Skolem function.

Once we have defined such a rule base $\mathcal{R}_\mathcal{G}$, we can use it to generate the query $exp_\mathcal{G}(Q)$ associated to the original query Q. This is done as follows:

1. First, we rewrite Q by substituting each predicate P in the body $body(Q)$ of Q with P'. We denote by Q' the resulting query. In the following we call "primed atom" every atom whose predicate is primed.
2. Then we build a *partial resolution tree* for Q', i.e., a tree having each node labeled by a conjunctive query q, with one of the atoms in $body(q)$ marked as "*selected*", obtained as follows.
   (a) The root is labeled by Q', and has marked as selected any (primed) atom in $body(Q')$ (for example the first in left-to-right order).
   (b) Except if condition (2c) below is satisfied, a node, labeled by a query q having a "selected" atom α, has one child for each rule r in $\mathcal{R}_\mathcal{G}$ such that there exists a most general unifier[1] $mgu(\alpha, head(r))$ between the atom α and the head $head(r)$ of the rule r. Each of such children has the following properties:

[1] 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 [20].

  - it is labeled by the query obtained from q by replacing the atom α with $body(r)$ and by substituting the variables with $mgu(\alpha, head(r))$;
  - it has as marked "selected" one of the primed atoms (for example the first in left-to-right order).

 (c) If a node d that is labeled by a query q and there exists a predecessor d' of d labeled by a query q' and a substitution θ of the variables of q' that makes q' equal to q, then d has a single child, which is labeled by the empty query (a query whose body is false).

3. Finally we return as result the query $exp_{\mathcal{G}}(Q)$ formed as the union of all non-empty queries in the leaves of the partial resolution tree.

The following three observations are crucial for characterizing both the termination and the correctness of our algorithm:

- The termination of the construction of the tree, and thus of the entire algorithm, is guaranteed by the condition (2c) and by the observation that all the rules in $\mathcal{R}_{\mathcal{G}}$ have a single atom in the body.
- By exploiting results on partial evaluation of logic programs (see [11]), it can be shown that $exp_{\mathcal{G}}(Q)$ is equivalent to the original query Q with respect to the global schema $\mathcal{G}$, that is, for each database $\mathcal{B}$ that is legal for $\mathcal{G}$, the evaluation of Q yields the same result as $exp_{\mathcal{G}}(Q)$, i.e., $Q^{\mathcal{B}} = (exp_{\mathcal{G}}(Q))^{\mathcal{B}}$.
- The query $exp_{\mathcal{G}}(Q)$ returned by the algorithm is a union of conjunctive queries. Each disjunct of $exp_{\mathcal{G}}(Q)$ is a conjunctive query over the predicates of the global schema, i.e., the elements that have an associated query over the sources by virtue of the mapping.

The above observations imply that, if we evaluate $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(Q))$ over the source database $\mathcal{D}$, we get exactly the set of certain answers $q^{\mathcal{I},\mathcal{D}}$ of Q with respect to $\mathcal{I}$ and $\mathcal{D}$.

With regard to the characterization of the computational complexity of the algorithm, we observe that the number of disjuncts in $exp_{\mathcal{G}}(Q)$ can be exponential in the number of rules in the rule base $\mathcal{R}_{\mathcal{G}}$ (and therefore in the size of the global schema $\mathcal{G}$), and in the number of variables in the original query Q. Note, however, that this bound is independent of the size of $\mathcal{D}$, i.e., the size of data at the sources. We remind the reader that the evaluation of a union of conjunctive queries can be done in time polynomial with respect to the size of the data. Since $exp_{\mathcal{G}}(Q)$ is a union of conjunctive queries, we can conclude that, if the queries associated by $\mathcal{M}_{\mathcal{G},\mathcal{S}}$ to the elements of $\mathcal{G}$ can be evaluated in polynomial time in the size of the data at the sources, then evaluating $unf_{\mathcal{M}_{\mathcal{G},\mathcal{S}}}(exp_{\mathcal{G}}(Q))$ over $\mathcal{D}$ is also polynomial in the size of the data at the sources. It follows that our query answering algorithm is polynomial with respect to data complexity.

Example 7. Referring again to Example 5, it is possible to see that, by evaluating the unfolding of the query returned by the algorithm, the whole set of certain answers to Q_1 with respect to $\mathcal{I}_1$ and $\mathcal{D}_2$ is obtained. In particular, t_1 is obtained by processing the rule $\mathsf{Member}'(x, y) \leftarrow \mathsf{Enrolled}'(x, y)$, which takes into account that Member is a generalization of Enrolled and the rule $\mathsf{Enrolled}'(x, f(x)) \leftarrow \mathsf{Student}'(x)$, which expresses the mandatory participation of Student in Enrolled.

5 Conclusions

While it is a common opinion that query processing is an easy task in the global-as-view approach to data integration, we have shown the surprising result that, when the global schema is expressed in terms of a conceptual data model, even a very simple one, query processing becomes difficult. The difficulties basically arise because of the need of dealing with incomplete information, similarly to the case of the local-as-view approach to data integration.

After a logic-based characterization of the data integration system, we have presented a novel query processing algorithm that is able to compute all correct answers to a query posed to the global schema, by reasoning on both the query and the conceptual global schema. We have also shown that query processing, although exponential with respect to the size of the query and the global schema, remains of polynomial data complexity.

We have implemented a first prototype of data integration system based on the presented algorithm. In addition to specifying the global schema in terms of a conceptual data model, the system allows several types of constraints to be expressed on the sources. Although we did not address this issue here, these constraints are used for carrying out several optimizations in accessing the sources. Overall, the first experiments about the performance of the system are extremely encouraging.

In this paper, we used a simple conceptual data model for expressing the global schema, and we used the language of conjunctive queries for expressing queries over the global schema. We observe, however, that all the results presented in the paper can be straightforwardly extended to the class of unions of conjunctive queries. As future work, we aim at enriching the conceptual model with more advanced features, such as disjointness assertions, and cardinality constraints on attributes and relationships. With these features, query answering becomes even more complex, due to the need of performing more sophisticated forms of data cleaning and reconciliation [6, 12]. Our goal is to modify the algorithm described in this paper so as to adapt to the new class of semantic conditions represented in the global schema.

References

[1] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized views. In *Proc. of PODS'98*, pages 254–265, 1998.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison Wesley Publ. Co., Reading, Massachussetts, 1995.

[3] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Intelligent techniques for the extraction and integration of heterogeneous information. In *Proc. of the IJCAI'99 Workshop on Intelligent Information Integration*, 1999.

[4] M. Bouzeghoub and M. Lenzerini. Special issue on data extraction, cleaning, and reconciliation. *Information Systems*, 2001. To appear.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of CoopIS'98*, pages 280–291, 1998.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 2001. To appear.

[7] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. Answering regular path queries using views. In *Proc. of ICDE 2000*, pages 389–398, 2000.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, and M. Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of LICS 2000*, pages 361–371, 2000.

[9] 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 *RIDE-DOM*, pages 124–131, 1995.

[10] P. P. Chen. The Entity-Relationship model: Toward a unified view of data. *ACM Trans. on Database Systems*, 1(1):9–36, Mar. 1976.

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

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

[13] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. of ICDT'99*, volume 1540 of *LNCS*, pages 332–347. Springer-Verlag, 1999.

[14] J. Gryz. Query folding with inclusion dependencies. In *Proc. of ICDE'98*, pages 126–133, 1998.

[15] A. Y. Halevy. Theory of answering queries using views. *SIGMOD Record*, 29(4):40–47, 2000.

[16] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of PODS'97*, 1997.

[17] A. Y. Levy. Logic-based techniques in data integration. In J. Minker, editor, *Logic Based Artificial Intelligence*. Kluwer Publishers, 2000.

[18] C. Li and E. Chang. Query planning with limited source capabilities. In *Proc. of ICDE 2000*, pages 401–412, 2000.

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

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

[21] B. Ludascher, A. Gupta, and M. E. Martone. Model-based mediation with domain maps. In *Proc. of ICDE 2001*, pages 81–90, 2001.

[22] L. Palopoli, L. Pontieri, G. Terracina, and D. Ursino. Intensional and extensional integration and abstraction of heterogeneous databases. *Data and Knowledge Engineering*, 35(3):201–237, 2000.

[23] X. Qian. Query folding. In *Proc. of ICDE'96*, pages 48–55, 1996.

[24] J. D. Ullman. Information integration using logical views. In *Proc. of ICDT'97*, volume 1186 of *LNCS*, pages 19–40. Springer-Verlag, 1997.

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