Source Integration in Data Warehousing

Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati
Dipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”
Via Salaria 113, 00198 Roma, Italy
{calvanese,degiacomo,lenzerini,nardi,rosati}@dis.uniroma1.it

Abstract

Source Integration is one of the core problems in Data Warehousing. Two critical factors for the design and maintenance of applications requiring Source Integration, and in particular Data Warehouse applications, are conceptual modeling of the domain, and reasoning support over the conceptual representation. We present a novel approach to conceptual modeling for Source Integration, which allows for suitably modeling the global concepts of the application, the individual information sources, and the constraints among different sources. Our methodological framework relies on the reasoning services associated with the modeling formalism to support an incremental Source Integration phase within the Data Warehouse design process.

1. Introduction

According to [18], integration is the most important aspect of a data warehouse. When data passes from the application-oriented operational environment to the data warehouse, possible inconsistencies and redundancies should be resolved, so that the warehouse is able to provide an integrated and reconciled view of data.

There are two basic approaches to the data integration problem, called *procedural* and *declarative*. In the procedural approach, data are integrated in an ad-hoc manner with respect to a set of predefined information needs, without resorting to an explicit notion of integrated data schema [17, 15]. In the declarative approach, the goal is to model the data at the sources by means of a suitable language, and to construct a unified representation to be used when querying the global information system [9, 1, 20].

In this paper we adopt a declarative approach to integration, and argue that two critical factors for the design and maintenance of applications requiring Information Integration, and in particular integration in Data Warehousing, are the *conceptual modeling of the domain*, and the possibility of *reasoning over the conceptual representation*. The approach we propose is based on building a conceptual representation of both the information sources and the Data Warehouse. An important aspect of the conceptual representation is the explicit specification of the set of interdependencies between objects in the sources and objects in the Data Warehouse. Thus, integration is seen as the process of understanding and representing the relationships between data in the sources and in the Data Warehouse, possibly with some reconciling actions, rather than producing a unified data schema. Moreover, suitable reasoning techniques associated with the conceptual formalism are used to support the designer during the resulting specification process. Specifically, our work provides the following main contributions:

1. We use special class-based logical formalisms, called *Description Logics* [14, 2], for the *conceptual modeling* of both the global domain and the various sources. Since the development of successful Information Integration solutions requires specific modeling features, we propose a new Description Logic, which treats n-ary relations as first-class citizens. Note that the usual characteristic of many Description Logics to model only unary predicates (concepts) and binary predicates (roles) would represent an intolerable limit in our case.

2. We provide suitable mechanisms for expressing what we call the *intermodel assertions*, i.e. interrelationships between concepts in different sources. Thus, integration is seen as the incremental process of understanding and representing the relationships between data in the sources, rather than simply producing a unified data schema. The fact that our approach is incremental is also important in amortizing the cost of integration.

3. For an accurate description of the information sources, we incorporate in our logic the possibility of describing the data at the sources in terms of a set of *relational structures*. Each relational structure is defined as a view over the conceptual representation, thus pro-

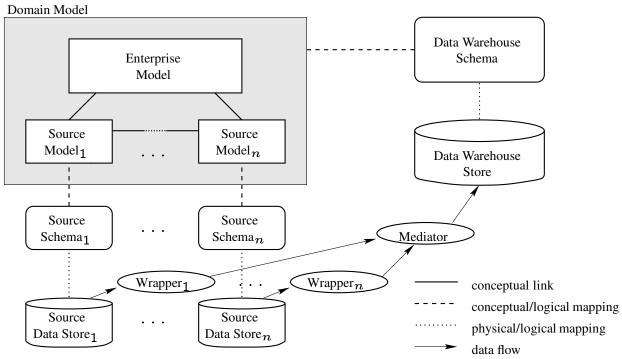

Figure 1. Architecture for Data Integration

viding a formal mapping between the description of data and the conceptual representation of the domain.

4. We provide *inference procedures* for the fundamental reasoning services, namely concept and relation subsumption, and query containment. Indeed, we make use of the first decidability result on query containment for a Description Logic with n-ary relations [4]. Based on these reasoning methods, we present a methodological framework for Information Integration, which can be applied both in the virtual and in the materialized approach.

The paper is organized as follows. In Section 2 we describe in more detail our framework for Information Integration based on Description Logics. In Section 3 we present the particular Description Logic we use in the framework. In Section 4 we illustrate how the reasoning techniques associated with our logic are used to improve the design and maintenance of the Information Integration system. Section 5 concludes the paper.

2. The Framework

In our approach to Source Integration, we refer to the architecture depicted in Figure 1, in which three layers can be identified:

- a *conceptual layer*, constituted by the Domain Model, including an Enterprise Model and one Source Model for each data source, which provides a conceptual representation of both the information sources and the Data Warehouse;
- a *logical layer* constituted by the *Source Schemas* and the *Materialized View Schema*, which describes the logical content of source data stores and of the materialized view store, respectively;
- a *physical layer*, which consists of the data stores containing the actual data of the sources and the integrated materialized views.

The methodology for Source Integration described in Section 4, and the reasoning techniques developed in Section 3, support the incremental building of the conceptual and the logical representations. The designer is provided with information on various aspects, including the global concepts relevant for new information requirements, the sources from which a new view can be defined, the correspondences between sources and/or views, and a trace of the integration steps.

We describe now the structure of the conceptual and logical layers, which constitute the core of the proposed integration framework. The actual formalism we adopt and the associated reasoning techniques are described in the next section.

Conceptual Layer. The *Enterprise Model* is a conceptual representation of the global concepts and relationships that are of interest to the application. It corresponds roughly to the notion of integrated conceptual schema in the traditional approaches to schema integration. However, since we propose an incremental approach to integration, the Enterprise Model is not necessarily a complete representation of all the data of the sources but it provides a consolidated and reconciled description of the concepts and the relationships that are important to the enterprise, and have already been analyzed. Such a description is subject to changes and additions as the analysis of the information sources proceeds. The *Source Model* of an information source is a conceptual representation of the data residing in it, or at least of the portion of data currently taken into account. Again, our approach does not require a source to be fully analyzed and conceptualized.

Both the Enterprise Model and the Source Models are expressed by means of a logic-based formalism (see Section 3) which is general and powerful enough to express the usual database models, such as the Entity-Relationship Model, the Relational Model, or the Object-Oriented Data Model (for the static part). The inference techniques associated with the formalism allow for carrying out several reasoning services on the representation.

Besides the Enterprise Model and the various Source Models, the *Domain Model* contains the specification of the interdependencies between elements of different Source Models and between Source Models and the Enterprise Model. The notion of interdependency is a central one in our approach. Since the sources are of interest in the overall architecture, integration does not simply mean producing the Enterprise Model, but rather to be able to establish the correct relationships both between the Source Models and the Enterprise Model, and between the various Source Models. We formalize the notion of interdependency by means of so called *intermodel assertions* [8], which provide a simple and effective declarative mechanism to express the dependencies that hold between entities (i.e. classes and relationships) in different models [16]. We use again a logic-based formalism to express intermodel assertions, and the associated inference techniques provide a means to reason about interdependencies among models.

Logical Layer. Our approach requires that each source, besides being conceptualized, is also described in the *Source Schema* in terms of a logical data model (in our case the Relational Model) which allows for representing the structure of the stored data. Such a structure is specified in terms of a set of relation definitions, each one expressed by means of a view (i.e. a query) over the conceptual representation of the source (i.e. the Source Model). Suitable software components, called *wrappers*, implement the mapping of physical structures to logical structures (see Figure 1).

The *Data Warehouse Schema* provides a description of the logical content of the materialized views constituting the *Data Warehouse*. Similarly to the case of the sources, each portion of the Data Warehouse Schema is described in terms of a set of definitions of relations, each one expressed in terms of a query over the Domain Model. A view is actually materialized starting from the data in the sources by means of suitable software components, called *mediators* (see Figure 1).

3. Representation and Reasoning

We describe now the formalism used both at the conceptual and the logical level, and the associated reasoning techniques.

Representation at the Conceptual Level. We use for the conceptual level a specific logic based formalism called $\mathcal{DLR}$, whose basic components are *concepts* (i.e. classes) and *n-ary relations*[1]. $\mathcal{DLR}$ is inspired by the knowledge representation languages introduced in [3, 11, 10, 8], and can be ragarded as an extension of *Description Logics* [14, 7, 2] towards n-ary relations.

We assume to deal with a finite set of *atomic relations* and *concepts*, denoted by $\mathbf{P}$ and A respectively. We use $\mathbf{R}$ to denote arbitrary *relations* (of given arity between 2 and n_{max}), and C to denote arbitrary *concepts*, respectively built according to the following syntax (i and j denote components of relations, i.e. integers between 1 and n_{max}, n denotes the arity of a relation, i.e. an integer between 2 and n_{max}, and k denotes a nonnegative integer)[2]:

$$\mathbf{R} \ ::= \ \top_n \mid \mathbf{P} \mid (\$i/n\colon C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2$$

$$C \ ::= \ \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists[\$i]\mathbf{R} \mid (\leq k\,[\$i]\mathbf{R})$$

$$\begin{array}{rcll}
\top_n^{\mathcal{I}} & \subseteq & (\Delta^{\mathcal{I}})^n & \\
\mathbf{P}^{\mathcal{I}} & \subseteq & \top_n^{\mathcal{I}} & \\
(\neg\mathbf{R})^{\mathcal{I}} & = & \top_n^{\mathcal{I}} \setminus \mathbf{R}^{\mathcal{I}} & \\
(\mathbf{R}_1 \sqcap \mathbf{R}_2)^{\mathcal{I}} & = & \mathbf{R}_1^{\mathcal{I}} \cap \mathbf{R}_2^{\mathcal{I}} & \\
(\$i/n\colon C)^{\mathcal{I}} & = & \{(d_1,\ldots,d_n) \in \top_n^{\mathcal{I}} \mid d_i \in C^{\mathcal{I}}\} & \\
\top_1^{\mathcal{I}} & = & \Delta^{\mathcal{I}} & (\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
A^{\mathcal{I}} & \subseteq & \Delta^{\mathcal{I}} & (C_1 \sqcap C_2)^{\mathcal{I}} = C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\
(\exists[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid \exists(d_1,\ldots,d_n) \in \mathbf{R}^{\mathcal{I}}. d_i = d\} & \\
(\leq k\,[\$i]\mathbf{R})^{\mathcal{I}} & = & \{d \in \Delta^{\mathcal{I}} \mid & \\
 & & \quad \sharp\{(d_1,\ldots,d_n) \in \mathbf{R}_1^{\mathcal{I}} \mid d_i = d\} \leq k\} &
\end{array}$$

Figure 2. Semantics of $\mathcal{DLR}$

The semantics of the $\mathcal{DLR}$ constructs is specified through the usual notion of interpretation. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is constituted by an *interpretation domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$ that assigns to each concept C a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, and to each relation $\mathbf{R}$ of arity n a subset $\mathbf{R}^{\mathcal{I}}$ of $(\Delta^{\mathcal{I}})^n$, such that the conditions in Figure 2 are satisfied (where $\mathbf{P}$, $\mathbf{R}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ have arity n). We observe that $\top_1$ denotes the interpretation domain, while $\top_n$, for $n > 1$, does *not* denote the n-cartesian product of the domain, but only a subset of it, that covers all relations of arity n. As a consequence, the "$\neg$" construct on relations is used to express difference of relations, rather than complement.

A $\mathcal{DLR}$ *conceptual model* $\mathcal{M}$ (i.e., either the Enterprise Model or one of the Source Models) is constituted by a finite set of *intramodel assertions*, which express knowledge on the relations and concepts in $\mathcal{M}$, and have the form

$$L \sqsubseteq L' \qquad L \not\sqsubseteq L' \qquad L \equiv L' \qquad L \not\equiv L'$$

with L, L' either two relations of the same arity or two concepts.

An interpretation $\mathcal{I}$ *satisfies* an intramodel assertion $L \sqsubseteq L'$ (resp. $L \equiv L'$) if $L^{\mathcal{I}} \subseteq L'^{\mathcal{I}}$ (resp. $L^{\mathcal{I}} = L'^{\mathcal{I}}$), and it satisfies $L \not\sqsubseteq L'$ (resp. $L \not\equiv L'$) if $\mathcal{I}$ does not satisfy $L \sqsubseteq L'$ (resp. $L \equiv L'$). An interpretation *satisfies* $\mathcal{M}$, if it satisfies all assertions in $\mathcal{M}$.

To specify knowledge on the conceptual interrelationships among the sources and/or the enterprise, we use *intermodel assertions* [8], which have essentially the form of intramodel assertions, although the two relations (concepts) L and L' belong to two different conceptual models $\mathcal{M}_i$, $\mathcal{M}_j$. Intermodel assertions can be either *extensional*, which express relationships between the extensions of the relations (concepts) involved, or *intensional*, which express conceptual relationships that are not necessarily reflected at the instance level. Formally, the interpretation of

[1] Domains, i.e. sets of values such as integer, string, etc., can be easily included in $\mathcal{DLR}$.

[2] Concepts and relations must be *well-typed*, which means that (i) only relations of the same arity n can be combined to form expressions of type $\mathbf{R}_1 \sqcap \mathbf{R}_2$ (which inherit the arity n), and (ii) $i \leq n$ whenever i denotes a component of a relation of arity n.

extensional intermodel assertions is analogous to the one of intramodel assertions. Instead, for the interpretation of intensional intermodel assertions only the intersection of the relations (concepts) L, L' with both $\top_{ni}$ and $\top_{nj}$ ($\top_{1i}$ and $\top_{1j}$) is considered. For example, an interpretation $\mathcal{I}$ satisfies the intermodel assertion $\mathbf{R}_i \sqsubseteq_{int} \mathbf{R}'_j$ if $\top_{ni}^{\mathcal{I}} \cap \top_{nj}^{\mathcal{I}} \cap \mathbf{R}_i^{\mathcal{I}} \subseteq \top_{ni}^{\mathcal{I}} \cap \top_{nj}^{\mathcal{I}} \cap \mathbf{R}_j'^{\mathcal{I}}$.

A *Domain Model (DM)* $\mathcal{W}$ is an $(m+2)$-tuple $\langle \mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_m, \mathcal{G} \rangle$ such that: (i) $\mathcal{M}_0$ is the Enterprise Model; (ii) each $\mathcal{M}_i$, for $i \in \{1, \ldots, m\}$, is a Source Model; (iii) $\mathcal{G}$ (for "glue") is a finite set of intermodel assertions. We assume that $\mathcal{G}$ always includes for each $i \in \{1, \ldots, m\}$ the following assertions: $\top_{1i} \sqsubseteq_{ext} \top_{10}$, and $\top_{ni} \sqsubseteq_{ext} \top_{n0}$ for each n such that a relation $\mathbf{R}$ of arity n appears in $\mathcal{M}_i$. An interpretation $\mathcal{I}$ *satisfies* $\mathcal{W}$ if it satisfies all the intramodel and intermodel assertions in $\mathcal{W}$.

Representation at the Logical Level. We express the logical level in terms of a set of relation schemas, each describing either a relation of a Source Schema, or a relation of the Data Warehouse Schema. Such relations are related to the DM by characterizing each relation schema in terms of a non-recursive Datalog query over the elements of the DM, i.e. a query of the form:

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

where each $\mathit{body}_i(\vec{\mathbf{x}}, \vec{\mathbf{y}}_i)$ is a conjunction of *atoms*, either $\mathbf{R}(\vec{\mathbf{t}})$ or $C(t)$ (where $\vec{\mathbf{t}}$ and t are variables in $\vec{\mathbf{x}}, \vec{\mathbf{y}}_i$)[3], with $\mathbf{R}$, C relations and concepts over the DM. The *arity* of q is equal to the number of variables of $\vec{\mathbf{x}}$.

We observe that, by means of assertions on both relations and concepts expressed in the DM, additional constraints than those directly present in the query can be imposed. This distinguishes our approach with respect to [13, 19], where n-ary relations appearing in queries are not part of the conceptual model.

Given an interpretation $\mathcal{I}$ of a DM $\mathcal{W}$, a query q for $\mathcal{W}$ of arity n is interpreted as the set $q^{\mathcal{I}}$ of n-tuples $(o_1, \ldots, o_n)$, with each $o_i \in \Delta^{\mathcal{I}}$, such that, when substituting $(o_1, \ldots, o_n)$ for $(x_1, \ldots, x_n)$, the formula

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

evaluates to true in $\mathcal{I}$. If q and q' are two queries (of the same arity) for $\mathcal{W}$, we say that q is *contained in* q' wrt $\mathcal{W}$, if $q^{\mathcal{I}} \subseteq q'^{\mathcal{I}}$ for every $\mathcal{I}$ satisfying $\mathcal{W}$.

Reasoning. The typical kinds of reasoning services needed at the conceptual level in order to support the designer in applying the integration methodology presented in Section 4 (e.g., checking whether the DM is consistent, checking whether a relation or a concept is satisfiable in the DM, checking subsumption between relations or concepts in the DM) can be reduced to checking satisfiability of the DM. The reasoning tasks can in particular be exploited for computing and incrementally maintaining the concept and relation lattice of the DM, or more generally the lattice of all concept and relation expressions.

[3] Our approach is applicable also when constants are used in the queries.

$$
\begin{array}{rcl}
\mathtt{CONTRACT}_0 & \sqsubseteq & (\$1\!:\mathtt{Client}_0) \sqcap (\$2\!:\mathtt{Dept}_0) \sqcap \\
 & & (\$3\!:\mathtt{Service}_0) \\
\mathtt{REG\text{-}AT}_0 & \sqsubseteq & (\$1\!:\mathtt{Client}_0) \sqcap (\$2\!:\mathtt{Dept}_0) \\
\mathtt{PrDept}_0 & \sqsubseteq & \mathtt{Dept}_0 \\
\mathtt{REG\text{-}AT}_1 & \sqsubseteq & (\$1\!:\mathtt{Client}_1) \sqcap (\$2\!:\mathtt{Dept}_1) \\
\mathtt{PROMOTION}_1 & \sqsubseteq & \mathtt{REG\text{-}AT}_1 \\
\mathtt{LOCATION}_1 & \sqsubseteq & (\$1\!:\mathtt{Dept}_1) \sqcap (\$2\!:\mathtt{String}) \\
\mathtt{Dept}_1 & \sqsubseteq & \exists^{\leq 1}\mathtt{LOCATION}_1[\$1].\top_2 \\
\mathtt{CONTRACT}_2 & \sqsubseteq & (\$1\!:\mathtt{Client}_2) \sqcap (\$2\!:\mathtt{Dept}_2) \sqcap \\
 & & (\$3\!:\mathtt{Service}_2) \\
\mathtt{Dept}_1 & \equiv_{ext} & \mathtt{PrDept}_0 \\
\mathtt{REG\text{-}AT}_1 & \sqsubseteq_{ext} & \mathtt{REG\text{-}AT}_0 \\
\mathtt{Client}_1 & \equiv_{ext} & \mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{REG\text{-}AT}_0[\$1].\mathtt{PrDept}_0 \\
\mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{CONTRACT}_0[\$1].\top_2 & & \\
 & \sqsubseteq_{ext} & \exists^{\geq 1}\mathtt{PROMOTION}_1[\$1].\top_2 \\
\mathtt{Client}_2 & \sqsubseteq_{ext} & \mathtt{Client}_0 \sqcap \exists^{\geq 1}\mathtt{CONTRACT}_0[\$1].\top_2 \\
\mathtt{Dept}_2 & \sqsubseteq_{ext} & \mathtt{Dept}_0 \\
\mathtt{Service}_2 & \equiv_{ext} & \mathtt{Service}_0 \\
\mathtt{Client}_1 & \equiv_{int} & \mathtt{Client}_2 \\
\mathtt{Dept}_1 & \equiv_{int} & \mathtt{Dept}_2
\end{array}
$$

Figure 3. Domain Model of the example

The expressiveness of $\mathcal{DLR}$, required for capturing meaningful properties in the DM, makes reasoning a complex task. We have devised a sound and complete procedure to decide the satisfiability of a DM which works in worst-case deterministic exponential time in the size of the DM. Indeed, this worst-case complexity is inherent to the problem, therefore reasoning with respect to a DM is EXPTIME complete. The inference method works in two steps: first, reasoning on the DM is reduced to reasoning on a knowledge base expressed in the Description Logic $\mathcal{CIQ}$ [12]; then reasoning procedures for $\mathcal{CIQ}$, based on the correspondence with Propositional Dynamic Logics, are exploited.

For reasoning at the logical level, we provide suitable techniques for query containment. In particular, we have developed an algorithm for deciding query containment with respect to a DM, which exploits a reduction to unsatisfiability in $\mathcal{CIQ}$, and which extends the one in [4, 5] to deal with both intramodel and intermodel assertions.

Example. Figure 3 shows a DM, $\mathcal{W} = (\mathcal{M}_0, \mathcal{M}_1, \mathcal{M}_2, \mathcal{G})$, that represents an enterprise and two sources containing information about contracts between clients and departments for services, and about registration of clients at departments (in the figure

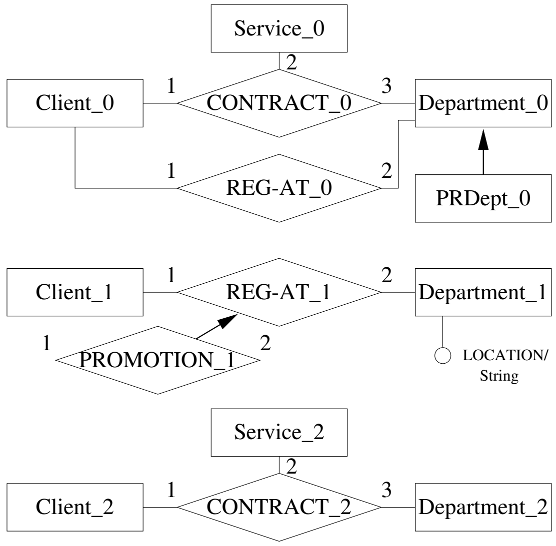

Figure 4. Enterprise and source models as Entity-Relationship diagrams

($\$i/n\colon C$) is abbreviated by ($\$i\colon C$)). Symbols subscripted by i refer to model $\mathcal{M}_i$. The intramodel assertions in $\mathcal{M}_0$, $\mathcal{M}_1$, $\mathcal{M}_2$ are visualized in Figure 4, using Entity-Relationship diagrams, which are fully compatible with $\mathcal{DLR}$. Source 1 contains information about clients registered at public-relations departments. Source 2 contains information about contracts and complete information about services. The Enterprise Model provides a reconciled conceptual description of the two sources. Note that, in this example, such reconciled description is not complete yet: e.g., the relation PROMOTION is not modeled in $\mathcal{M}_0$ (recall that our approach to integration is incremental). The various interdependencies among relations and concepts in the Enterprise Model and the two Sources Models are represented by the intermodel assertions on the right-hand side of Figure 3.

As for the logical level representation, suppose, for example, that the actual data in Source 1 are described by a relational table $\mathtt{Table}_1$ having three columns, one for the client, one for the department which the client is registered at, and one for the location of the department. Such a table is specified in terms of the DM by means of the query:

$$\mathtt{Table}_1(x,y,z) \leftarrow \mathtt{REG\text{-}AT}_1(x,y) \wedge \mathtt{LOCATION}_1(y,z)$$

Using the reasoning services associated with $\mathcal{DLR}$, we can automatically derive logical consequences of the DM. For instance, we can prove that the assertion $\mathtt{PROMOTION}_1 \sqsubseteq_{ext} \mathtt{REG\text{-}AT}_0 \sqcap (\$2\colon \mathtt{PrDept}_0)$ is a logical consequence of $\mathcal{W}$. Observe that, although $\mathcal{M}_0$ does not contain a relation PROMOTION, the above assertion relates $\mathtt{PROMOTION}_1$ to $\mathcal{M}_0$ in a precise way.

Next, consider, for instance, the following queries posed to $\mathcal{M}_0$:

$$\begin{array}{lcl} q_1(x,y) & \leftarrow & \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \\ q_2(x,y) & \leftarrow & \mathtt{Client}_0(x) \wedge \mathtt{CONTRACT}_0(x,y,z) \wedge \\ & & \mathtt{REG\text{-}AT}_0(x,w) \wedge \mathtt{PrDept}_0(w) \end{array}$$

q_2 is obviously contained in q_1. However, taking into account the assertions in $\mathcal{W}$, we can also derive that q_1 is contained in q_2 wrt $\mathcal{W}$.

4. The Methodology

We outline a methodology for Source Integration in Data Warehousing, based on the techniques previously described. The methodology deals with two scenarios, called *source-driven* and *client-driven*.

4.1. Source-Driven Integration

Source-driven integration is triggered when a new source or a new portion of a source is taken into account for integration. The steps to be accomplished in this case are:

1. *Source Model construction.* The Source Model capturing the concepts and the relationships of the new source that are critical for the enterprise is produced.

2. *Source Model integration.* The Source Model is *integrated into the Domain Model*. This can lead to changes both to the Source Models, and to the Enterprise Model. The specification of intermodel assertions and the derivation of implicit relationships by exploiting the reasoning techniques, represent the novel part of the methodology. Notably, not only assertions relating elements in one Source Model with elements in the Enterprise Model, but also assertions relating elements in different Source Models are of importance. For example, inferring that the set of instances of a relation in source S_i is always a subset of those in source S_j can be important in order to infer that accessing source S_j for retrieving instances of the relation is useless.

3. *Quality analysis.* The Quality Factors of the resulting Domain Model are evaluated and a restructuring is accomplished to match the required criteria. This step requires the use of the reasoning techniques associated with our formalisms to check for quality factors such as consistency, redundancy, readability, accessibility, believability [6].

4. *Source Schema construction.* The Source Schema, i.e. the logical view of the new source or a new portion of the source (expressed as a collection of queries over the corresponding Source Model) is produced. The source schemas are used in order to determine the sources relevant for computing answers to queries, by exploiting the ability to reason about queries.

5. *Data Warehouse Schema restructuring.* On the basis of the new source, an analysis is carried out on

whether the Data Warehouse Schema should be restructured and/or modified in order to better meet quality requirements. Again, the schema is constituted by a set of queries over the Domain Model, and for its restructuring the use of reasoning techniques is crucial. A restructuring of the Data Warehouse Schema may require the design of new mediators.

4.2. Client-Driven Integration

The client-driven design strategy refers to the case when a new query (or a set of queries) posed by a client is considered. The reasoning facilities are exploited to analyze and systematically decompose the query and check whether its components are subsumed by the views defined in the various schemas. The analysis is carried out as follows:

1. By exploiting query containment checking, we verify if and how the answer can be computed from the materialized views stored in the Data Warehouse.

2. In the case where the materialized information is not sufficient, we verify if the answer can be obtained by materializing new concepts represented in the Domain Model. In this case, query containment helps to identify the set of subqueries to be issued on the sources and to extend and/or restructure the Data Warehouse Schema. Different choices can be identified, based on various preference criteria (e.g. minimization of the number of sources [20]) which take into account the above mentioned quality factors.

3. In the case where neither the materialized data nor the concepts in the Domain Model are sufficient, the necessary data should be searched for in new sources, or in new portions of already analyzed sources. The new (portions of the) sources are then added to the Domain Model using the source-driven approach, and the process of analyzing the query is iterated.

5. Conclusions

We have presented the fundamental features of a declarative approach to Source Integration in Data Warehousing based on an expressive conceptual modeling formalism equipped with reasoning techniques. We are currently applying the presented framework to the problem of Data Warehouse design within the ESPRIT Project DWQ (Foundations of Data Warehouse Quality).

References

[1] Y. Arens, C. A. Knoblock, and W. Shen. Query reformulation for dynamic information integration. *J. of Intelligent Information Systems*, 6:99–130, 1996.

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

[3] D. Calvanese, G. De Giacomo, and M. Lenzerini. Structured objects: Modeling and reasoning. In *Proc. of DOOD-95*, number 1013 in LNCS, pages 229–246. Springer-Verlag, 1995.

[4] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of PODS-98*, 1998.

[5] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Database integration for datawarehousing. Technical Report DWQ-UNIROMA-001, DWQ Consortium, Mar. 1997.

[6] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. Technical Report DWQ-UNIROMA-002, DWQ Consortium, Oct. 1997.

[7] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In *Proc. of KR-94*, pages 109–120, 1994.

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

[9] C. Collet, M. N. Huhns, and W.-M. Shen. Resource integration using a large knowledge base in Carnot. *IEEE Computer*, 24(12):55–62, 1991.

[10] G. De Giacomo and M. Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proc. of JELIA-94*, volume 838 of *LNAI*, pages 332–346. Springer-Verlag, 1994.

[11] G. De Giacomo and M. Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proc. of IJCAI-95*, pages 801–807, 1995.

[12] G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of KR-96*, pages 316–327, 1996.

[13] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. A hybrid system integrating Datalog and concept languages. In *Proc. of AI*IA-91*, number 549 in LNAI. Springer-Verlag, 1991.

[14] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, pages 193–238. CSLI Publications, 1996.

[15] J. Hammer, H. Garcia-Molina, J. Widom, W. Labio, and Y. Zhuge. The Stanford data warehousing project. *IEEE Bull. on Data Engineering*, 18(2):41–48, 1995.

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

[17] R. Hull and G. Zhou. A framework for supporting data integration using the materialized and virtual approaches. In *Proc. of ACM SIGMOD*, pages 481–492, 1996.

[18] W. H. Inmon. *Building the Data Warehouse*. John Wiley & Sons, second edition, 1996.

[19] A. Y. Levy and M.-C. Rousset. CARIN: A representation language combining Horn rules and description logics. In *Proc. of ECAI-96*, pages 323–327, 1996.

[20] A. Y. Levy, D. Srivastava, and T. Kirk. Data model and query evaluation in global information systems. *J. of Intelligent Information Systems*, 5:121–143, 1995.