Information Integration: Conceptual Modeling and Reasoning Support

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

Information Integration is one of the core problems in cooperative information systems. We argue that two critical factors for the design and maintenance of applications requiring Information Integration are conceptual modeling of the domain, and reasoning support over the conceptual representation. In particular, we present a general architecture for Information Integration that explicitly includes a conceptual representation of the application. We illustrate how the architecture can express several integration settings and existing systems. We provide various arguments in favor of the conceptual level in the architecture and of automated reasoning over the conceptual representation. Finally, we present a specific proposal of an integration system which realizes the general architecture and is equipped with decidable reasoning procedures.

1. Introduction

Information Integration has the goal of providing an integrated and coherent view of data stored in multiple, possibly inhomogeneous information sources. It is one of the core problems in distributed databases, cooperative information systems, and data warehousing, which are key areas in the software development industry [36, 28, 34, 22].

Early work on integration was carried out in the context of database design, and focused on the so-called *schema integration* problem, i.e. designing a global, unified schema for a database application starting from several subschemas, each one produced independently from the others [3]. More recent efforts have been devoted to *information integration*, which generalizes schema integration by taking into account actual data in the integration process. Here the input is a collection of source data sets (each one constituted by a schema and actual data), and the goal is to provide an integrated and reconciled view of the data residing at the sources.

The integration system may in principle be used both to access the data and to update the stored information. However, performing updates on the integrated data requires changing the data in the sources. Hence a tight coordination between the sources and the integration system and among the different sources is needed. Such form of integration is typically of interest to federated databases [32, 18]. Recently a looser approach to integration has emerged, where the autonomy of the sources is a basic requirement, and the integration system is seen as a "client" of the sources which cannot interfere with their operation. Hence, performing updates on the integrated data is not of concern and the reconciled view is used only for answering queries. For this reason this form of integration is often called *read-only integration*. In this approach the organization responsible for the integration system is typically different and independent from the organizations managing the single sources [33]. In this paper we concentrate on read-only integration.

Information integration can be either *virtual* or *materialized*. In the first case, the integration system acts as an interface between the user and the sources [32, 24], and is typical of multi-databases, distributed databases, and more generally open systems. In virtual integration query answering is generally costly, because it requires accessing the sources. In the second case, the system maintains a replicated view of the data at the sources [19, 25], and is typical, for example, for both information systems re-engineering and data warehousing. In materialized information integration, query answering is generally more efficient, because it does not require accessing the sources, whereas maintaining the materialized views is costly, especially when the views must be up-to-date with respect to the updates at the sources (*view refreshment*). In this paper, we do not deal with the problem of view refreshment.

There are two basic approaches to the information 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. In this case, the basic issue is to design suitable software modules that access the sources in order to fulfill the predefined information requirements. Several information integration (both virtual and materialized) projects, such as TSIMMIS [12, 33], *Squirrel* [38, 23], and WHIPS [20, 37] follow this idea. They do not require an explicit notion of integrated data schema, and rely on two kinds of software components: *wrappers* that encapsulate sources, converting the underlying data objects to a common data model, and *mediators* [35] that obtain information from one or more wrappers or other mediators, refine this information by integrating and resolving conflicts among the pieces of information from the different sources, and provide the resulting information either to the user or to other mediators. The basic idea is to have one mediator for every query pattern required by the user, and generally there is no constraint on the consistency of the results of different mediators.

In the declarative approach, the goal is to model the data at the sources by means of a suitable language, to construct a unified representation, to refer to such a representation when querying the global information system, and to derive query answers by means of suitable mechanisms accessing the sources and/or the materialized views. This is the idea underlying systems such as *Carnot* [13, 21], SIMS [1, 2] and *Information Manifold* [31, 27, 29].

In this paper we propose a novel architecture for declarative, read-only Information Integration, both virtual and materialized. The architecture allows one to explicitly model data and information needs – i.e. a specification of the data that the system provides to the user – at various levels:

- The *conceptual level* contains a conceptual representation of the sources and of the reconciled integrated data, together with an explicit declarative account of the relationships among their components. Additionally, it provides a declarative representation of the information needs served by the Integration System.

- The *logical level* contains a representation in terms of a logical data model of the sources and of the data materialized by the integration system. In addition it contains the logical schemas of the information needs.

- The *physical level* contains a store for the materialized data, wrappers for the sources and mediators for the information needs and the materialized data store.

- The *meta level* is a repository of the meta information concerning the Integration System.

The relationship between the conceptual and the logical, and between the logical and the physical level is represented explicitly by specifying mappings between corresponding objects of the different levels.

The main contributions of the paper are:

- The presentation of the general architecture as outlined above. The architecture can express several different integration settings and existing systems. (Section 2)

- A discussion in favor of the conceptual level in the architecture and the ability of automated reasoning over the conceptual representation. Reasoning can be exploited in several tasks both in the design phase of the Integration System, and in its maintenance phase. (Section 3)

- A specific proposal of an integration system which realizes the general architecture and which supports modeling both at the conceptual and at the logical level, and is equipped with decidable reasoning procedures. (Section 4)

2. Architecture of integration systems

In this section we describe the architecture of an integration system resulting from the introduction of a conceptual layer. In particular, we illustrate both the various components that are maintained and used by the system, and the tasks that the system has to carry out for performing its job. The proposed architecture serves as a general setting where different approaches to integration can be evaluated and compared. Indeed, we illustrate how existing integration systems can be obtained as specializations of this general architecture. In Section 4 we present the CDLNR approach to Information Integration, viewing it as a specific instantiation of the architecture.

2.1. Components

The data structures managed by an integration system are shown in Figure 1, where four levels are sin-

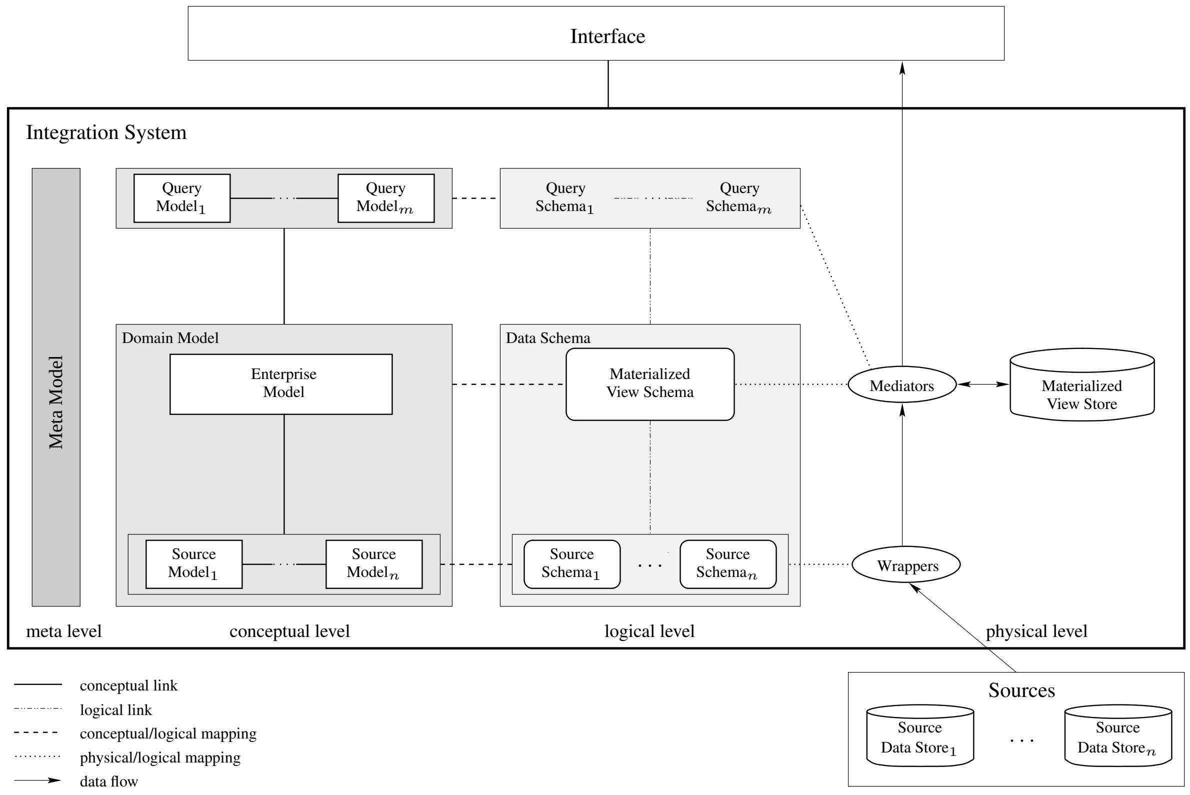

Figure 1. Architecture for Data Integration

gled out: *conceptual*, *logical*[1], *physical*, and *meta*. Furthermore, Figure 1 includes the following elements, which are outside the boundary of the integration system:

- The *Interface*, which is the module that allows the communication with both the user (i.e. anyone interested in retrieving information) and the designer (i.e. the one in charge of the building and the functioning of the system).
- The *External Sources*, which represent the independent systems managing the actual data that the system is supposed to integrate.

The conceptual level

The conceptual level contains a formal description of the concepts, the relationships between concepts, and the information requirements that the integration application has to deal with. The key feature of this level is that such a description is independent from any system consideration, and is oriented towards the goal of expressing the semantics of the application. In particular, we distinguish among the following elements in the conceptual level:

- The *Enterprise Model*[2] 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.
- For an information source S, the *Source Model* of S is a conceptual representation of the data residing in S.
- The term *Domain Model* is used to denote the union of both the Enterprise Model and the various Source Models, plus possible intermodel relationships, i.e. relationships holding between concepts belonging to different models (i.e. between one concept in source S and one concept in the Enterprise Model, or between one concept in one source and one concept in another source).

[1]Here the term "logical" is used according to the database terminology, where it denotes a description of data in terms of structures managed by DBMSs (e.g., relational tables), which are at a more abstract level with respect to the physical organization of data.

[2]Here the term "model" is used to denote a formal description in a given representation language. Note the difference with the usual meaning in databases, where it denotes the formalism itself.

- The term *Query Model* is used to denote a conceptual representation of any information need. An example of Query Model in a Data Warehouse application is a conceptual specification of a multidimensional table requiring aggregations over elementary data.

We point out that the Domain Model contains *intermodel relationships*, i.e. 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 architecture. Since the sources are of interest in the system, 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.

Any integration system including a conceptual level must adopt suitable languages for expressing the above mentioned elements of the architecture. We use the term *conceptual description language* for denoting the formalism in which the Domain Model is expressed, and the term of *conceptual query language* for denoting the language in which Query Models are expressed. An example of conceptual description language is the Entity-Relationship Model. An example of conceptual query language is any formalism for expressing queries over a semantic data model.

The logical level

The *logical level* contains the description of the data and the queries of interest to the system, expressed in terms of typical logical structures managed by DBMSs. In particular, the *Source Schema* of a source S describes the logical content of S and the *Materialized View Schema* describes the logical content of the materialized views maintained by the system. Collectively, the Source Schemas and the Materialized View Schema form what we call the *Data Schema*. Obviously, the Materialized View Schema is meaningful only in the case where the integrated data (or portions thereof) are materialized, whereas is meaningless in the case of fully virtual integration. Finally, the *Query Schemas* express the information needs at the logical level.

Suitable languages are used for expressing the above mentioned elements. We use the term *logical data model* for denoting the formalism in which the Source Schemas and the Materialized View Schemas are expressed. Similarly, the term *logical query language* refers to the language for expressing the Query Schemas. Obvious examples of logical data model and logical query language are the relational model, and SQL, respectively.

The physical level

In our architecture, the physical level refers to the actual data managed by the system. Therefore, the physical level is the one where the extensional information of the system is taken into account. In particular, the *Materialized View Store* contains the data that the system maintains materialized. Figure 1 shows also wrappers and mediators at this level. A *wrapper* is a software module that is able to access a source and retrieve the data therein in a form that is coherent with the logical specification of the source.

A *mediator* is a software module that takes as input a set of data produced by either a wrapper or another mediator, and produces as output another set of data, namely the one corresponding to the result of a given query. In other words, a mediator is always associated to a particular query at the logical level. The result of a mediator can be either materialized, or transferred to the interface.

The meta level

The meta level comprises the *Meta Model*, which is the repository with all meta infomation about the various system components, and is used by both the user and the designer. A more detailed discussion of the meta level is outside the scope of this paper, and can be found, for example, in [26].

Mappings

Figure 1 also shows the mappings between the conceptual and the logical level, and between the logical and the physical level.

Regarding the first aspect, the mapping between Source Models and Source Schemas represents the fact that the correspondence between the logical representation of data in the sources and concepts in the Source Models should be explicit. The same holds for information needs expressed at the conceptual level and queries expressed at the logical level. Finally, the correspondence between elements of the Domain Model and the Materialized View Schema represents the information on which are the concepts and relationships that are materialized in the views maintained by the system. We assume that the integration system makes it available a *conceptual/logical mapping specification language*, by which the above correspondences can be formally defined.

Regarding the second aspect, the mapping between mediators and Query Schemas and/or the Materialized Views Schema makes it explicit the fact each mediator is supposed to compute the extension a logical object, which can be either materialized or not. A wrapper is always associated to an element of a Source Schema, namely, the one whose data are extrated and retrieved by the wrapper. The mapping with Source Schemas represents exactly the correpondence between a wrapper w and the logical element whose extensional data are extracted form the source by w. Similarly to the case of conceptual/logical mapping, we postulate the existence of a *logical/phyisical mapping specification language*, by which the above correspondences can be formally defined.

2.2. Tasks

In this section we briefly discuss the tasks that should be carried out during the use of an integration system conforming to our architecture.

The first class of tasks comprises all the activities regarding the definition of the different elements of the architecture. Such activities mainly pertain to the design of the integration system. For example, the specification of the various Conceptual Models and the intermodel links belongs to this phase. We note that the architecture does not prescribe to build the conceptual level in one shot, but rather supports an incremental definition of both the Domain and the Query Models. Indeed, such models are subject to changes and additions as the analysis of the information sources proceeds.

Observe that in the (partially) materialized approach to integration, one of the most critical task is the decision of what and how to materialize. Moreover, in both the materialized and the virtual approach, the task of wrapper and mediator design is extremely important. Designing a wrapper means to decide how to access the source in order to retrieve data, and designing a mediator means to decide how to access the sources in order to answer a particular query or to materialize a particular view. Note that the design of a mediator comprises the resolution of conflicts and/or heterogeinity of data residing in different sources.

The second kind of tasks include all the design activities to be performed when a new information need arises. In this case, the new query has to be compared with those computed by the available mediators. The most important problem here is the one of *query rewriting*, i.e. checking if and how the new query can be reformulated in terms of those computed by the existing mediators. In virtual integration, this may lead the new mediator to simply call for the existing mediators. In materialized integration, reformulating the query in terms of the materialized views means avoiding to access the sources. Conversely, if the query (or part thereof) cannot be answered by simply relying on the existing materialized views, a new view (or new views) should be materialized, and the problem of query rewriting presents itself in a different form: the new view to materialize is seen as a query that has to be formulated in terms of the Source Schemas.

Finally, the third class of tasks concern the activities that are routinely carried out during the operational phase of the systems, namely data extraction, query computation, and view materialization.

2.3. Comparison with existing systems

We now show how the architecture outlined above can be instantiated to different information integration settings.

Schema integration [3] In the schema integration setting, integration starts by providing a conceptual representation of the sources (Source Models), and proceed by generating the global database schema (Enterprise Model). Such a schema is then used for the design of the implemented database (Materialized View Schema, Materialized View Store). Once such database has been created, the sources are discarded and the conceptual level is not used anymore.

Multidatabases [32, 24] The setting of multidatabases deals with different sources, which are considered as internal components of the Integration System. Based on a logical representation of the sources, mediators are designed in order to satisfy information needs also expressed at the logical level (Query Schemas). Such mediators do not materialize data in the system. Typically, the conceptual level is not taken into account.

Global information systems [33] In this setting the goal is to provide tools for the integrated access to multiple and diverse autonomous information sources and repositories, such as databases, HTML documents, unstructured files. Among the systems proposed in this framework, Information Manifold [31, 27, 29] uses a representation at the conceptual level of a reconciled view (called World View) of the information sources and no data is materialized. Also TSIMMIS [12, 33] deals with a virtual scenario, but does not provide a conceptual representation of data. One difference between the above two systems is that in the former, data at the sources are described as views over the World View, whereas in the latter, each mediator computes a view over the sources. Both these strategies have disadvantages: in the first case inter-source relationships are not expressible, and in the second case general concepts cannot be characterized independently from the sources. Notably, the CDLNR approach described in the next section does not impose any predefined direction for expressing links between the sources and the Enterprise Model.

Data warehouses [25] In this setting views are materialized, as e.g. in the WHIPS system [20, 37], in which information is not represented at the conceptual level. The lack of a conceptual level is shared by the SQUIRREL system [40, 39, 38, 23]. However, within SQUIRREL it is also possible to take into account the case of virtual views.

3. Advantages of conceptual modeling and reasoning

The most distinguished features of the architecture described in the previous section are related with the conceptual level. The role and importance of the conceptual level in traditional architectures for information management systems is well understood and tools for conceptual modeling are commonly used to drive system design.

Conversely, conceptual design has not often been addressed in the framework of Information Integration, where the focus of attention has been mostly on the logical and physical levels. We believe that conceptual modeling can play a fundamental role in Information Integration, as long as the proper modeling issues and the proper modeling tools are clearly identified. In particular, in Information Integration the conceptual model should be able to represent the relationships between data in different sources. In the next section we provide a language for representing inter-model relationships and formally characterize them, while below we discuss the advantages that derive from the ability to provide a conceptual representation and reasoning support for Information Integration. The discussion takes it for granted that once the modeling issues are well understood, tools to support the conceptual design can actually be developed.

The following is a list of advantages of conceptual modeling that are relevant both in the design and in the operation of an Integration System.

Declarative and system independent approach In general terms, one can say that a conceptual level in the architecture for Information Integration is the obvious mean for pursuing a declarative approach to Information Integration. As a consequence, all the advantages deriving from making various aspects of the system explicit are obtained. The conceptual level provides a system independent specification of the relationships between sources, and between sources and the enterprise model.

High level of abstraction in user interface One of the most tangible effects of conceptual modeling has been to break the barrier between user and system by providing a higher level of abstraction in the design. Moreover conceptual models are naturally expressed in graphical form, and graphical tools that adequately present the overall information scenario are key factors in user interfaces.

Incremental approach One criticism that is often raised to the declarative approaches to Information Integration is that it requires a reconciled view of the data, which can be very costly to obtain. As we already mentioned, having a conceptual level does not impose to fully develop it at once. Rather, one can incrementally add new sources or new elements therein, when they become available, or when needed, thus amortizing the cost of integration. Therefore, the overall design can be regarded as the incremental process of understanding and representing the relationships between data in the sources.

Documentation While in the procedural approach the information about the interrelationships between sources is hard-wired in the mediators, in a declarative approach it can be made explicit. The importance of this clearly emerges when looking at large organizations where the information about data is widespread into separate pieces of documentation that are often difficult to access and non necessarily conforming to common standards. Conceptual modeling for Information Integration can thus provide a common ground for the documentation of the enterprise data stores and can be seen as a formal specification for mediator design. By making the representation explicit we gain re-usability of the acquired knowledge, which is not achieved within a procedural approach to Information Integration.

Maintenance Classical advantages of conceptual level in the design phase are advantages also in the maintenance phase of the Information Integration System (sources change, hence "design never ends").

All the advantages outlined above can be obtained simply by having the appropriate linguistic (graphical) tools for expressing the conceptual model, as well as the mappings to the other components of the architecture.

Another set of features that one can obtain from the introduction of the conceptual level are related to the ability to *reason over the conceptual representation*. Such a possibility, which is fully supported by our approach described in the next section, can be used in the accomplishment of several activities concerning both the design and the operation of the system. For example, one can use reasoning to check the conceptual representation for inconsistencies and redundancies; to maintain the system in response to changes in the information needs; to improve the overall performance of the system. In particular, we are pursuing the goal of characterizing the quality of data and to use such a characterization to improve quality of services by relying on reasoning support on the conceptual representation [7].

4. The CDLNR approach

In this section we instantiate the general integration architecture presented in Section 2 to a specific proposal which we call the CDLNRsystem (Conceptual Data Language for N Repositories). The distinguishing features of CDLNR are: the use of the conceptual level in the representation of information and the ability of handling both materialized and virtual representation of views. Moreover, CDLNR is equipped with automated reasoning support both at the conceptual and at the logical level. A detailed description of the physical level is outside the scope of the paper.

4.1. Representation at the conceptual level

In CDLNR both the Enterprise Model and the Source Models are expressed by means of a logic-based conceptual description language, called $\mathcal{DLR}$ [6, 8], 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). To specify knowledge on the conceptual interrelationships among the sources and/or the enterprise, we use *intermodel assertions* [11] expressed also in $\mathcal{DLR}$. Intermodel assertions provide a simple and effective declarative mechanism to express the dependencies that hold between entities (i.e. classes and relationships) in different models [22].

$\mathcal{DLR}$ is a *Description Logic* [17, 9, 4] based on the formalisms presented in [5, 14], which includes *concepts* (i.e. unary relations) and *n-ary relations*. *Relations* $\mathbf{R}$ (of given arity between 2 and n_{max}) and *concepts* C are built starting from a set of *atomic relations* $\mathbf{P}$ and *atomic concepts* A 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)[3]:

$$\begin{array}{lcl}\mathbf{R} & ::= & \top_n \mid \mathbf{P} \mid (\$i/n\colon C) \mid \neg\mathbf{R} \mid \mathbf{R}_1 \sqcap \mathbf{R}_2 \\ C & ::= & \top_1 \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \\ & & \exists[\$i]\mathbf{R} \mid (\leq k\,[\$i]\mathbf{R})\end{array}$$

The semantics of the $\mathcal{DLR}$ constructs is specified through the 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.

The Enterprise Model and each Source Model 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 $\mathcal{M}$.

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

The conceptual query language for the Query Models is an SQL-like language over the alphabet of the Domain Model [6], suitably extended with aggregation constructs, in the line of those presented in [10, 15].

[3]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.

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

Figure 2. Semantic rules for $\mathcal{DLR}$ ($\mathbf{P}$, $\mathbf{R}$, $\mathbf{R}_1$, and $\mathbf{R}_2$ have arity n)

The expressiveness of $\mathcal{DLR}$, required for capturing meaningful properties at the conceptual level, makes reasoning a complex task. We have devised a sound and complete algorithm to decide the satisfiability of the conceptual representation [8].

4.2. Representation at the logical level

The CDLNR logical data model is the relational model: we express data at 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 Materialized View Schema. The mapping between the Data Schemas and the Domain Model is obtained by characterizing each relation schema in terms of a non-recursive Datalog query over the elements of the Domain Model, i.e. a query of the form:

$$q(\vec{\mathbf{x}}) \;\leftarrow\; body_1(\vec{\mathbf{x}},\vec{\mathbf{y}}_1) \vee \cdots \vee body_m(\vec{\mathbf{x}},\vec{\mathbf{y}}_m)$$

where each $body_i(\vec{\mathbf{x}},\vec{\mathbf{y}}_i)$ is a conjunction of *atoms*, either $\mathbf{R}(\vec{\mathbf{t}})$ or $C(t)$ (where $\vec{\mathbf{t}}$ and t are variables in $\vec{\mathbf{x}}$,$\vec{\mathbf{y}}_i$)with $\mathbf{R}$, C relations and concepts over the Domain Model. The *arity* of q is equal to the number of variables of $\vec{\mathbf{x}}$. By means of assertions on both relations and concepts expressed in the Domain Model, additional constraints than those directly present in the query can be imposed[4].

The logical query language is an embedded SQL, which enables to express a Query Schema in terms of the Data Schema, possibly making use of other Query Schemas.

Automated reasoning at the logical level is based on techniques for query containment developed in [6], which is exploited for performing query rewriting [33].

4.3. Example

Figure 3 shows a Domain Model $\mathcal{W}$ that represents in $\mathcal{M}_0$ an enterprise and in $\mathcal{M}_1$ and $\mathcal{M}_2$ two sources containing information about contracts between clients and departments for services, and about registration of clients at departments. 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 typical of conceptual modeling in Databases and 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, which are also part of $\mathcal{W}$.

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 mapped on $\mathcal{W}$ 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 $\mathcal{W}$. 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 $\mathcal{M}_0$:

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

[4]This distinguishes our approach with respect to [16, 30], where n-ary relations appearing in queries are not part of the conceptual model.

$$
\begin{array}{rcl}
\texttt{CONTRACT}_0 & \sqsubseteq & (\$1\colon \texttt{Client}_0) \sqcap (\$2\colon \texttt{Dept}_0) \sqcap \\
 & & (\$3\colon \texttt{Service}_0) \\
\texttt{REG-AT}_0 & \sqsubseteq & (\$1\colon \texttt{Client}_0) \sqcap (\$2\colon \texttt{Dept}_0) \\
\texttt{PrDept}_0 & \sqsubseteq & \texttt{Dept}_0 \\
 & & \\
\texttt{REG-AT}_1 & \sqsubseteq & (\$1\colon \texttt{Client}_1) \sqcap (\$2\colon \texttt{Dept}_1) \\
\texttt{PROMOTION}_1 & \sqsubseteq & \texttt{REG-AT}_1 \\
\texttt{LOCATION}_1 & \sqsubseteq & (\$1\colon \texttt{Dept}_1) \sqcap (\$2\colon \texttt{String}) \\
\texttt{Dept}_1 & \sqsubseteq & \exists^{\leq 1}\texttt{LOCATION}_1[\$1].\top_2 \\
 & & \\
\texttt{CONTRACT}_2 & \sqsubseteq & (\$1\colon \texttt{Client}_2) \sqcap (\$2\colon \texttt{Dept}_2) \sqcap \\
 & & (\$3\colon \texttt{Service}_2)
\end{array}
$$

$$
\begin{array}{rcl}
\texttt{Dept}_1 & \equiv_{ext} & \texttt{PrDept}_0 \\
\texttt{REG-AT}_1 & \sqsubseteq_{ext} & \texttt{REG-AT}_0 \\
\texttt{Client}_1 & \equiv_{ext} & \texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{REG-AT}_0[\$1].\texttt{PrDept}_0 \\
\texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{CONTRACT}_0[\$1].\top_2 & & \\
 & \sqsubseteq_{ext} & \exists^{\geq 1}\texttt{PROMOTION}_1[\$1].\top_2 \\
 & & \\
\texttt{Client}_2 & \sqsubseteq_{ext} & \texttt{Client}_0 \sqcap \exists^{\geq 1}\texttt{CONTRACT}_0[\$1].\top_2 \\
\texttt{Dept}_2 & \sqsubseteq_{ext} & \texttt{Dept}_0 \\
\texttt{Service}_2 & \equiv_{ext} & \texttt{Service}_0 \\
 & & \\
\texttt{Client}_1 & \equiv_{int} & \texttt{Client}_2 \\
\texttt{Dept}_1 & \equiv_{int} & \texttt{Dept}_2
\end{array}
$$

Figure 3. Domain model $((\$i/n\colon C)$ is abbreviated by $(\$i\colon C))$

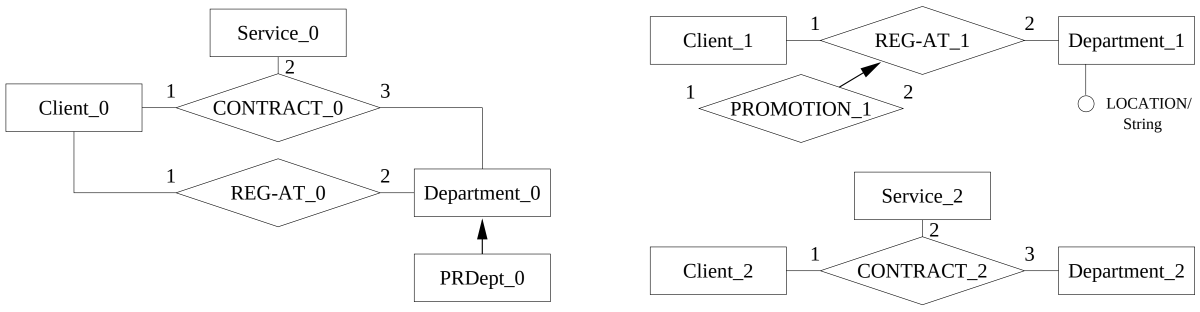

Figure 4. Enterprise and source models in Entity-Relationship diagrams

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 $\mathcal{W}$.

5. Conclusions

The main contribution of this work is a novel architecture for Information Integration which generalizes several proposals in the literature. The distinguishing feature of the proposed architecture is the emphasis on the conceptual modeling of the data, which allows for automated reasoning support for the Integration System design and maintenance tasks.

We have also proposed CDLNR, a specific Integration System based on Description Logics, that conforms to the general architecture and that allows for decidable reasoning over the conceptual model. CDLNR is currently being used in the context of data warehouse design within the ESPRIT Project DWQ (Foundations of Data Warehouse Quality) [7, 26]. A tool implementing the CDLNR system is currently under construction for such a project.

References

[1] Y. Arens, C. Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *J. of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.

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

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

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

[5] 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.

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

[7] 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.

[8] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of KR-98*, 1998.

[9] D. Calvanese, M. Lenzerini, and D. Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proc. of KR-94*, pages 109–120, Bonn, 1994. Morgan Kaufmann, Los Altos.

[10] T. Catarci, G. D'Angiolini, and M. Lenzerini. Conceptual language for statistical data modeling. *Data and Knowledge Engineering*, 17(2):93–125, 1995.

[11] 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.

[12] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Proc. of IPSI-94 Conference*, Tokyo (Japan), 1994.

[13] 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.

[14] 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.

[15] G. De Giacomo and P. Naggar. Conceptual data model with structured objects for statistical databases. In *Proceedings of the 8th International Conference on Scientific and Statistical Data Base Management*, pages 168–175, Stockholm, Sweden, June 1996.

[16] 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. An extended version appeared also in the Working Notes of the AAAI Fall Symposium "Principles of Hybrid Reasoning".

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

[18] M. García-Solaco, F. Saltor, and M. Castellanos. A semantic-discriminated approach to integration in federated databases. In *Proc. of CoopIS-95*, pages 19–31, 1995.

[19] A. Gupta and I. S. Mumick. Maintenance of materialized views: Problems, techniques, and applications. *IEEE Bull. on Data Engineering*, 18(2):3–18, 1995.

[20] 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.

[21] M. N. Huhns, N. Jacobs, T. Ksiezyk, W.-M. S. an Munindar P. Singh, and P. E. Cannata. Integrating enterprise information models in Carnot. In *Proc. of CoopIS-93*, pages 32–42, 1993.

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

[23] 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.

[24] A. Hurson, M. Bright, and S. Pakzad, editors. *Multidatabase Systems: An Advanced Solution for Global Information Sharing*. IEEE Computer Society Press, 1994.

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

[26] M. Jarke, M. A. Jeusfeld, C. Quix, and P. Vassiliadis. Architecture and quality in data warehouses. In *Proc. of CAiSE'98*, 1998.

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

[28] C. Knoblock and A. Levy, editors. *AAAI Symposium on Information Gathering from Heterogeneous, Distributed Environments*, number SS-95-08 in AAAI Spring Symposium Series. AAAI Press/The MIT Press, 1995.

[29] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query answering algorithms for information agents. In *Proc. of AAAI-96*, pages 40–47, 1996.

[30] 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.

[31] 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.

[32] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3), 1991.

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

[34] J. Widom. Special issue on materialized views and data warehousing. *IEEE Bulletin on Data Engineering*, 18(2), 1995.

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

[36] G. Wiederhold. Special issue: Intelligent integration of information. *J. of Intelligent Information Systems*, 6(2/3), 1996.

[37] J. L. Wiener, H. Gupta, W. J. Labio, Y. Zhuge, H. Garcia-Molina, and J. Widom. A system prototype for warehouse view maintenance. Technical report, Stanford University, 1996. Available at http://www-db-stanford.edu/warehousing/warehouse.html.

[38] G. Zhou, R. Hull, and R. King. Generating data integration mediators that use materializations. *J. of Intelligent Information Systems*, 6:199–221, 1996.

[39] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Data integration and warehousing using H20. *IEEE Bull. on Data Engineering*, 18(2):29–40, 1995.

[40] G. Zhou, R. Hull, R. King, and J.-C. Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of CoopIS-95*, pages 4–18, 1995.