Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information Systems 31 (2006) 1–32
www.elsevier.com/locate/infosys

Enterprise modeling and Data Warehousing in TELECOM ITALIA

Diego Calvanese[a,*], Luigi Dragone[b], Daniele Nardi[b], Riccardo Rosati[b], Stefano M. Trisolini[c,1]

[a]*Faculty of Computer Science, Free University of Bolzano/Bozen, Piazza Domenicani 3, I-39100 Bolzano-Bozen BZ, Italy*
[b]*Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Rome, Italy*
[c]*TELECOM ITALIA, Italy*

Received 8 July 2003; received in revised form 1 July 2004; accepted 23 July 2004

Abstract

We present a methodology for Data Warehouse design and its application within the TELECOM ITALIA information system. The methodology is based on a conceptual representation of the Enterprise, which is exploited both in the integration phase of the Warehouse information sources and during the knowledge discovery activity on the information stored in the Warehouse. The application of the methodology in the TELECOM ITALIA framework has been supported by prototype software tools both for conceptual modeling and for data integration and reconciliation.

Keywords: Data warehousing; Data integration; Conceptual modeling; Automated reasoning

1. Introduction

Information integration [1] is one of the main problems to be addressed when designing a Data Warehouse [2]. Possible inconsistencies and redundancies between data residing at the operational data sources and migrating to the Data Warehouse need to be resolved, so that the Warehouse is able to provide an integrated and reconciled view of data within the organization. The basic components of a data integration system are wrappers and mediators [3,4]. A wrapper is a software module that accesses a data source, extracts the relevant data, and presents such data in a specified format, typically as a set of relational tables. A mediator collects, cleans, and combines data produced by wrappers and/or other mediators, according to a specific information need of

*Corresponding author. Tel.: +39-0471-016160; fax: +39-0471-016009.
E-mail addresses: calvanese@inf.unibz.it (D. Calvanese), dragone@dis.uniroma1.it (L. Dragone), nardi@dis.uniroma1.it (D. Nardi), rosati@dis.uniroma1.it (R. Rosati), stefano.trisolini@tin.it (S.M. Trisolini).
URL: http://www.inf.unibz.it/~calvanese/, http://www.dis.uniroma1.it/~dragone/, http://www.dis.uniroma1.it/~nardi/, http://www.dis.uniroma1.it/~rosati/.
[1]Data Warehouse and DMDWM Consulting S.A.S., Italy.

0306-4379/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.is.2004.07.002

the integration system. The specification and the realization of mediators is the core problem in the design of an integration system.

In Data Warehouse applications, the data sources are mostly internal to the organization. Moreover, large organizations typically provide informational needs in terms of an integrated conceptual representation of the corporate data that abstracts from the physical and logical structure of data in the sources. The data stored in the Data Warehouse should reflect such an informational need, and hence should be defined in terms of the corporate data.

TELECOM ITALIA is the main Italian provider of national and international telecommunication services, and is among the largest companies worldwide. In large companies the need to access company data for business intelligence is both an organizational and a technical challenge, requiring a considerable amount of financial and human resources. Given the development of information technology in the nineties, in TELECOM ITALIA data warehousing [5] has been a natural evolution of enterprise-wide data management and data integration. A *Data Warehouse* can be defined as a set of materialized views over the operational information sources of an organization, designed to provide support for data analysis and management's decisions.

In the last years, TELECOM ITALIA has carried out a large integration initiative of enterprise information systems, called IBDA,[2] resulting in the construction of an enterprise-wide database integrated at the conceptual level. Due to the limitations of the available technologies and the costs of replacing and re-engineering legacy applications, such an activity has lead to a solution based on federated databases and legacy systems wrapping, according to the main guidelines of virtual enterprise system integration.

Meanwhile, the information systems of TELECOM ITALIA have quickly evolved, in particular new applications have been developed, and existing ones have been upgraded or replaced. Such a rapid evolution has been due to both internal and external factors: the birth and growth of new markets (such as mobile telephone and Internet services) and new competitors, the privatization of the company and the subsequent buyouts. Given also the various and disparate information requirements of business intelligence and decision making activities at different levels (e.g., tactical and strategical marketing), the integrated information system started showing inadequate to suit the company's new informational needs. Consequently, in order to provide an adequate timely deployment, the development of Online Analytical Processing (OLAP) and Decision Support Systems (DSS) applications has been carried out in an unstructured way, resulting in a low modularization and non-effective usage of the Data Warehouse infrastructure.

These issues have pointed out the necessity of adopting an incremental Data Warehousing methodology, in particular, the *local-as-view* (LAV) approach to Data Warehousing proposed in the context of the European project DWQ[3] [5,6]. In such an approach, each table both in a source and in the Data Warehouse is defined in terms of a view over the global model of the corporate data. This extends the traditional LAV approach to integration, where the information content of each data source is defined in terms of a query over (possibly materialized) global relations constituting the corporate view of data [1,7–10]. The LAV approach is in contrast to the *global-as-view* (GAV) approach for data integration [11–17], typically proposed in Data Warehousing [2,18]. Such an approach requires, for each information need, to specify the corresponding query in terms of the data at the sources. Notably, the LAV approach enables decoupling between information availability and information requirements. Therefore, the introduction of a new information source or the replacement of an existing one does not have any impact on the definition of the Data Warehouse expressed over the global model of corporate data.

Nonetheless, there are several important questions that are not addressed by the work on

[2]IBDA stands for "Integrazione di Basi di Dati Aziendali", i.e., integration of company databases.

[3]ESPRIT Basic Research Action Project EP 22469 "Foundations of Data Warehouse Quality (DWQ)", http://www.dbnet.ece.ntua.gr/~dwq/.

integration. More specifically, integration anticipates semantic problems with data, but does not address efficiency issues, which are critical for data warehousing. Thus, to guarantee a proper performance of the Data Warehouse, a major reorganization of the data store may be required, with additional costs. This motivates a layered architecture of the Data Warehouse [5], where a primary Data Warehouse feeds the data to several layers of aggregation (called secondary Data Warehouses or Data Marts) before they become available to the final user. Moreover, typically there is the need to efficiently take into account legacy systems that are not integrated, and external or extemporaneous data sources that can provide relevant information to the Data Warehouse, possibly for a limited time window.

In this paper we report on the experience of TELECOM ITALIA in the development of its Enterprise Data Warehouse. Such a Data Warehouse adopts a layered architecture, including various Primary Data Warehouses concerning phone traffic of different types and customer information, and several Secondary Data Warehouses, which are at the basis of the Knowledge Discovery and Data Mining activity carried out in TELECOM ITALIA. For the development of its Data Warehouse TELECOM ITALIA has followed the methodology proposed in the DWQ project [5,6], which is based on the LAV approach. Indeed, one of the distinguishing features of the approach is a rich *modeling language* for the conceptual level that extends the Entity-Relationship data model, and thus is fully compatible with the conceptual modeling tools adopted by TELECOM ITALIA. Moreover, the modeling formalism is equipped with *automated reasoning tools*, which can support the designer during Data Warehouse construction, maintenance and evolution. At the logical level, the methodology allows the designer to declaratively specify several types of *Reconciliation Correspondences* between data in different sources and in the Data Warehouse, which allow her to take care of differences in the representation at the logical level of the same entities at the conceptual level. Such correspondences are then used to automatically derive the specification of the correct mediators for the loading of the materialized views of the Data Warehouse. This is done by relying on a *query rewriting algorithm*, whose role is to reformulate the query that defines the view to materialize in terms of both the source relations and the Reconciliation Correspondences. The characteristic feature of the algorithm is that it takes into account the constraints imposed by the Conceptual Model, and uses the Reconciliation Correspondences for cleaning, integrating, and reconciling data coming from different sources.

The paper is organized as follows. In Section 2 we describe the enterprise information system in TELECOM ITALIA. In Section 3 we introduce the enterprise modeling framework at the basis of the Data Warehouse design methodology that is discussed in Section 4. In Section 5 we present the development process of a portion of the Data Warehouse of TELECOM ITALIA, concentrating on the Primary Data Warehouse design activity. In Section 7 we briefly discuss the use of the Secondary Data Warehouse for Data Mining and Decision Support applications. Finally, in Section 8 we draw some conclusions.

2. The enterprise information system in TELECOM ITALIA

In this section we sketch the main methodological and technological issues that arose in the last years in the development of the enterprise integrated database of TELECOM ITALIA and, subsequently, in the design and implementation of a Data Warehouse for TELECOM ITALIA. The two efforts, although driven by different needs and requirements can be regarded as a continuous development of an integrated view of the enterprise data. Although we analyze the design and the development of the Data Warehouses in TELECOM ITALIA, we deal with many issues that are common in a large enterprise; so our conclusions are easily generalizable to other scenarios.

2.1. Data base integration

In 1993, TELECOM ITALIA has launched a strategic project, called IBDA, with the following

main goals:

- the definition of an Enterprise Data Model and the migration/evolution of existing data;
- the design and implementation of databases covering the main sections of the Enterprise Data Model (customers, suppliers, network, administration, etc.);
- the design and implementation of a client/server architecture and of the communication middleware;
- the design and implementation of data access services.

The driving motivations of the IBDA strategic project are typical of a common scenario in many worldwide large enterprises, where there is a proliferation of legacy databases with a large overhead in the design and maintenance of software for interfacing applications and providing access to the data. IBDA was based on a staged implementation of services that form a layer separating data from application processes. More specifically, the IBDA service for a database is the exclusive agent that provides access to the data. A database integrated in the IBDA architecture is denoted as BDA and is identified by a unique number. The access is actually accomplished through contracts that enforce the semantic and referential policies for the database.

In this case, we must cope with the numerous and extremely differentiated data sources of the Enterprise Information System of TELECOM ITALIA. In particular, one can find different applications, based on different data management technologies (from hierarchical to object-relational DBMSs), that share information, or, in other words, that manage common concepts.

In 1996, the project was formally completed with the integration of 48 operational databases, while in the subsequent years new databases have been continuously added to IBDA. At present, several other databases are included in IBDA as BDAs and there are ongoing projects for adding more. In the following years it has been realized that the process of database inclusion in IBDA is basically incremental.

2.2. Data warehousing

In TELECOM ITALIA, data warehousing has been a natural evolution of data integration. Starting from a large integrated enterprise database, and given the size of the data to be stored in the Data Warehouse, the architecture of the TELECOM ITALIA Enterprise Data Warehouse includes a group of Primary Data Warehouses, which are devolved to collect, integrate and consolidate the data extracted from the Operational Data Stores. The Primary Data Warehouses feed the data to several systems on which the user applications (e.g., Decision Support System) rely. These systems are also included in the Enterprise Data Warehouse as Secondary Data Warehouses, also known as Data Marts.

The main difference between the two kinds of Data Warehouses is that the Primary Data Warehouses contain only "atomic" level data, while the Secondary Data Warehouses typically contain information that has been aggregated at different levels of detail. The Enterprise Data Warehouse architecture is basically stratified; therefore, the Secondary Data Warehouses are loaded only with data extracted from the Primary Data Warehouses.

Presently, the TELECOM ITALIA Enterprise Data Warehouse includes the following Primary Data Warehouses:

- *IPDW—Interconnection Traffic Primary Data Warehouse*, containing call detail records (CDRs), whose main purpose is to analyze network usage patterns between TELECOM ITALIA Network Nodes and other service providers. The daily loading is 40 millions CDRs and the store contains 6 months of history data.
- *CPDW—Customer Primary Data Warehouse*, containing information on TELECOM ITALIA products and services by customer, to analyze customer data. The daily loading is about 150 GB and the average size is 1.5 TB.
- *TPDW—Voice Traffic Primary Data Warehouse*, containing additional information on CDRs records from PSTN and ISDN switches, to support various analysis tasks for Marketing, Administration, and Customer Management.

The daily loading is 130 millions CDRs and the store contains 12 months of history data.

An integrated data store constitutes a solid basis for the design of Data Warehouse applications: many of the problems that arise for data warehousing are anticipated (data extraction, cleaning, and reconciliation), thus providing good quality data for analytical processing. Moreover, Data Warehouse applications can be developed in a more coherent way, because of the existence of the integrated data store.

Nonetheless, there are several important questions that are not addressed by the conventional approach to integration. More specifically, integration does anticipate semantic problems with data, but does not address efficiency issues, which are critical for data warehousing. Thus, to guarantee a proper performance of the Data Warehouse, a major re-organization of the data store may be required, with additional costs.

As a consequence of the layered approach to data warehousing taken in TELECOM ITALIA, a strong emphasis is given to methodologies. In the design of these methodologies, and their supporting tools, TELECOM ITALIA has benefited from the experience on quality assurance in data warehousing gained at the Dipartimento di Informatica e Sistemistica of the University of Rome "La Sapienza" within the European project DWQ [5]. TELECOM ITALIA has devised proprietary methodologies and tools for incremental schema and data integration based upon research results achieved within DWQ. These methodologies provide a sound framework for data acquisition in data warehousing and for the development of Data Warehouse applications, which are based on the cooperation with the top management, iterative refinement, and rapid prototyping. Such methodologies need to take into account both the complexity and the internal organization of roles and responsibilities.

3. The enterprise modeling framework

In this section we describe the general framework, developed within DWQ, at the basis of our approach to schema and data integration [5,6,19–21]. Specifically, we first illustrate the architecture of a Data Warehouse in our framework. Then, we present the conceptual level of the architecture, in particular both the formalism used for expressing the conceptual level, and the methodology for building a conceptual representation of the Data Warehouse. Finally, we describe the logical level, and the process of data integration within our architecture.

3.1. Architecture

A Data Warehouse can be seen as a database which maintains an integrated, reconciled, and materialized view of information residing in several data sources. In our approach, we explicitly model the data in the sources and in the Data Warehouse at different levels of abstraction [6,19,20]:

- The *conceptual level*, which contains a conceptual representation of the corporate data.
- The *logical level*, which contains a representation in terms of a logical data model of the sources and of the data materialized in the Data Warehouse.
- The *physical level*, which contains a specification of the stored data, the wrappers for the sources, and the mediators for loading the data store.

The relationships between the conceptual and the logical, and between the logical and the physical level, are represented explicitly by specifying mappings between corresponding objects of the different levels. In the rest of this section, we focus on the conceptual and logical levels, referring to the abstract architecture depicted in Fig. 1. For a description of the physical level we refer to [5].

3.2. Conceptual level

In the overall Data Warehouse architecture, we explicitly conceive a *conceptual level*, which provides a conceptual representation of the data managed by the enterprise, including a conceptual representation of the data residing in sources, and of the global concepts and relationships that are of

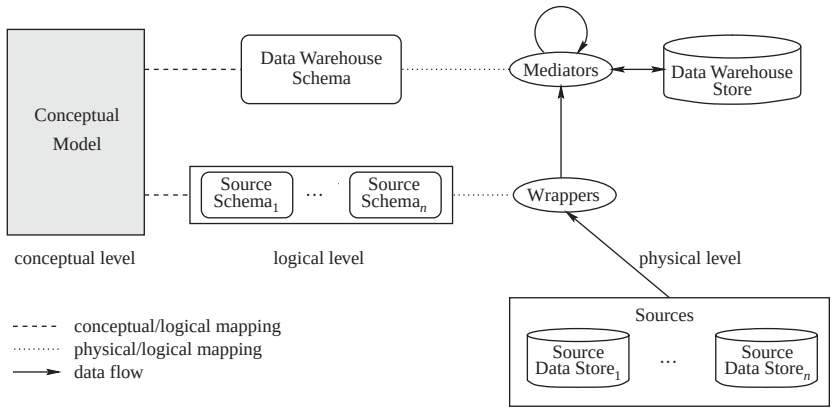

Fig. 1. Architecture for data integration.

interest to the Data Warehouse application. Such a description, for which we use the term *Conceptual Model*, is independent from any system consideration, and is oriented towards the goal of expressing the semantics of the application. The Conceptual Model corresponds roughly to the notion of integrated conceptual schema in the traditional approaches to schema integration, thus providing a consolidated view of the concepts and the relationships that are important to the enterprise and have been currently analyzed. Such a view includes a conceptual representation of the portion of data, residing in the sources, currently taken into account. Hence, our approach is not committed to the existence of a fully specified Conceptual Model, but rather supports an incremental definition of such a model. Indeed, the Conceptual Model is subject to changes and additions as the analysis of the information sources proceeds.

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. In this respect, data integration can be regarded as the process of understanding and representing the relationships between data residing in the information sources and the information contained in the Data Warehouse. Both schema integration and data integration are driven by the Conceptual Model. Furthermore, data reconciliation is also addressed at the conceptual level. In fact, the specification of interdependencies can be profitably exploited by automated reasoning tools, which are able to derive and verify several kinds of properties concerning the conceptual specification of information.

The formalization of information in the Conceptual Model is based on the distinction between *conceptual objects* and *values*. Reflecting such a distinction, the Conceptual Model consists of two components:

(1) an enriched *Entity-Relationship model*, which formalizes the properties of conceptual objects;
(2) a set of *domain assertions*, which model the properties of values (see Appendix A).

The enriched Entity-Relationship model is formalized in terms of a logic-based formalism, called $\mathcal{DLR}$ [20]. Such a formalism allows us to characterize the Entity-Relationship (ER) model augmented with several forms of constraints that cannot be expressed in the standard ER model. Moreover, it provides sophisticated automated reasoning capabilities, which can be exploited in verifying different properties of the Conceptual Model.

A detailed description of the features of $\mathcal{DLR}$ is presented in Appendix A. Here, to exemplify the use of $\mathcal{DLR}$, we show how to formalize into $\mathcal{DLR}$ a simple ER schema. A full-fledged application of the proposed methodology is shown in a case study from the telecommunication domain [22,23].

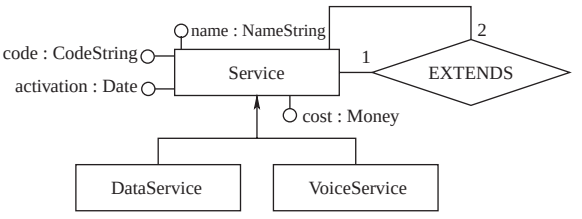

Fig. 2. Entity-Relationship schema for Example 1.

Example 1. The schema shown in Fig. 2 represents telecommunication services, partitioned into data and voice services, and a relationship modeling the fact that a service may extend another service. The following set of $\mathcal{DLR}$ assertions exactly captures the ER schema in the figure.

$$\begin{aligned}
\mathsf{Service} \quad & \sqsubseteq_{ext} (=1\ \mathsf{name}) \sqcap \forall \mathsf{name}.\mathsf{NameString} \sqcap \\
& \quad (=1\ \mathsf{code}) \sqcap \forall \mathsf{code}.\mathsf{CodeString} \sqcap \\
& \quad (=1\ \mathsf{activiation}) \sqcap \forall \mathsf{activation}.\mathsf{Data} \sqcap \\
& \quad (=1\ \mathsf{cost}) \sqcap \forall \mathsf{cost}.\mathsf{Money} \\
\mathsf{Service} \quad & \equiv \mathsf{DataService} \sqcup \mathsf{VoiceService} \\
\mathsf{DataService} \quad & \sqsubseteq_{ext} \neg \mathsf{VoiceService} \\
\mathsf{EXTENDS} \quad & \sqsubseteq_{ext} (\$1\colon \mathsf{Service}) \sqcap (\$2\colon \mathsf{Service})
\end{aligned}$$

The first assertion specifies the existence and uniqueness of the attributes of Service and their domains. The next two assertions specify that services are partitioned into data and voice services. The last assertion specifies the typing of the EXTENDS relationship.

3.3. Logical level

The logical level provides a description of the logical content of each source, called the *Source Schema*, and the logical content of the materialized views constituting the Data Warehouse, called the *Data Warehouse Schema* (see Fig. 1). Such schemas are intended to provide a structural description of the content of both the sources and the materialized views in the Data Warehouse.

A Source Schema is provided in terms of a set of relations using the relational model. The link between the logical representation and the conceptual representation of the source is formally defined by associating to each relation a query over the Conceptual Model that describes its content. In other words, the logical content of a source relation is described in terms of a view over the virtual database represented by the Conceptual Model, adopting the *local-as-view approach* [4].

To map physical structures to logical structures, we make use of suitable wrappers, which encapsulate the sources. Each wrapper presents the source as a set of relations, hiding how the source actually stores its data, the data model it adopts, etc. In particular, we assume that all attributes in the relations are of interest to the Data Warehouse application (attributes that are not of interest are hidden by the wrapper). Relation attributes are thus modeled as either entity attributes or relationship attributes in the Conceptual Model.

The Data Warehouse Schema, which expresses the logical content of the materialized views constituting the Data Warehouse, is provided in terms of a set of relations. Similarly to the case of the sources, each relation of the Data Warehouse Schema is described in terms of a query over the Conceptual Model.

3.3.1. Queries over the Conceptual Model

From a technical point of view, queries over the Conceptual Model are unions of conjunctive queries. More precisely, a query q over the Conceptual Model has the form

$$T(\tilde{\mathbf{x}}) \leftarrow q(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}),$$

where the *head* $T(\tilde{\mathbf{x}})$ defines the schema of the relation in terms of a name T, and its *arity*, i.e., the number of columns (number of components of $\tilde{\mathbf{x}}$ —we use $\tilde{\mathbf{x}}$ to denote a tuple of variables $x_1, \ldots, x_n$, for some n), and the *body* $q(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$ describes the content of the relation in terms of the Conceptual Model. The body has the form

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

where each $conj_i(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_i)$ is a conjunction of *atoms*, and $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_i$ are all the variables appearing in the conjunction. Each atom is of the form $E(t)$, $R(\tilde{\mathbf{t}})$, or $A(t, t')$, where $\tilde{\mathbf{t}}$, t, and t' are variables in $\tilde{\mathbf{x}}$, $\tilde{\mathbf{y}}_i$ or constants, and E, R, and A are respectively entities, relationships, and attributes appearing in the Conceptual Model. In the following, we will also consider queries whose body may contain special predicates that do not appear in the conceptual model.

The semantics of queries is as follows. Given a database that satisfies the Conceptual Model, a query

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

of arity n is interpreted as the set of n-tuples $(d_1, \ldots, d_n)$, where each d_i is an object of the database, such that, when substituting each d_i for x_i, the formula

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

evaluates to true.

Suitable inference techniques allow for carrying out the following reasoning services on queries by taking into account the constraints imposed by the Conceptual Model [24]:

- *Query containment*. Given two queries q_1 and q_2 (of the same arity n) over the Conceptual Model, we say that q_1 is *contained in* q_2, if the set of tuples denoted by q_1 is contained in the set of tuples denoted by q_2, in every database satisfying the Conceptual Model.
- *Query consistency*. A query q over the Conceptual Model is *consistent*, if there exists a database satisfying the Conceptual Model in which the set of tuples denoted by q is not empty.
- *Query disjointness*. Two queries q_1 and q_2 (of the same arity) over the Conceptual Model are *disjoint*, if the intersection of the set of tuples denoted by q_1 and the set of tuples denoted by q_2 is empty, in every database satisfying the Conceptual Model.

The notion of query over the Conceptual Model is a powerful tool for modeling the logical level of the Sources and the Data Warehouse. As mentioned above, we express the relational tables constituting both the Data Warehouse Schema and Source Schemas in terms of queries over the Conceptual Model, with the following characteristics:

- Relational tables are composed of tuples of values, which are the only kind of objects at the logical level. Therefore, each variable in the head of the query represents a value (not a conceptual object).
- Each variable appearing in the body of the query either denotes a conceptual object or a value, depending on the atoms in which it appears. Since, in each database that satisfies the Conceptual Model, conceptual objects and values are disjoint sets, no query can contain a variable which can be instantiated by both a conceptual object and a value.
- Each conceptual object is represented by a tuple of values at the logical level. Thus, a mechanism is needed to express this kind of correspondence between a tuple of values and the conceptual object it represents. This is taken into account by the notion of *adornment* introduced below.

3.3.2. Source and Data Warehouse Logical Schema description

We remind the reader that we assume that each source is encapsulated by a suitable wrapper, and this wrapper provides a logical view of the data stored in the source in terms of the relational model, i.e., in the form of a set of relations.

As we said before, the query associated with a source relation provides the glue between the conceptual and the logical representation. However, to capture in a precise way the data in the sources, more information is needed in order to describe the actual structure of the data in the relation. This is done by associating to the relation an *adornment*, whose role is to declare the domains of the columns of the table, and which are the attributes of the table that are used to identify the objects of the Conceptual Model. In other words, the adornment is used to make it explicit how the objects of the conceptual representation are coded into values of the logical representation.

An adorned query is an expression of the form

$$T(\tilde{\mathbf{x}}) \leftarrow q(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}) \mid \alpha_1, \ldots, \alpha_n,$$

where $\alpha_1, \ldots, \alpha_n$ constitutes the *adornment* in which each α_i is an *annotation* on variables appearing in $\tilde{\mathbf{x}}$. In particular:

- For each $X \in \tilde{\mathbf{x}}$, we have an annotation of the form

  $X :: V$,

  where V is a domain expression. Such an annotation is used to specify how values bound

to X are represented in the table at the logical level.

- For each tuple of variables $\tilde{\mathbf{z}} \subseteq \tilde{\mathbf{x}}$ that is used for identifying in T a conceptual object $Y \in \tilde{\mathbf{y}}$ mentioned in $q(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, we have an annotation of the form

  Identify($[\tilde{\mathbf{z}}]$, Y),

  where we have grouped the variables $\tilde{\mathbf{z}}$ into a single argument $[\tilde{\mathbf{z}}]$. Such an annotation makes it explicit that the tuple of values $\tilde{\mathbf{z}}$ is a representation of the conceptual object Y.

Similarly to the case of source relations, the relations to be materialized in the Data Warehouse are described as adorned queries over the Conceptual model. Note that the adorned query associated to a table in a source is the result of a reverse engineering analysis of the source, whereas in this case the adorned query is a high-level specification of what we want to materialize in the table of the Data Warehouse, and thus of the mediator for loading such a materialized view. Since we express the semantics of the Data Warehouse tables in terms of the Conceptual Model, also the relations in the Data Warehouse are seen as views of such a Conceptual Model.

3.3.3. Reconciliation Correspondences

Assume that the decision of which data to materialize has been taken, and has resulted in the specification of a new Data Warehouse relation T expressed in terms of an adorned query. One crucial task is to design the mediator for T, i.e., the program that accesses the sources and loads the correct data into the relation T. Designing the mediator for T requires first of all to reformulate the query associated with T in terms of the Source relations. However, such a reformulation is not sufficient. The task of mediator design is complicated by the possible heterogeneity between the data in the sources. We have already mentioned the most important ones, namely, mismatches between data referring to the same real world object, errors in the data stored in the sources, inconsistencies between values representing the properties of the real world objects in different sources.

To cope with this problem we follow the approach based on the notion of Reconciliation Correspondence, proposed in [19]. In order to anticipate possible errors, mismatches, and inconsistencies between data in the sources, our approach allows the designer to declaratively specify the correspondences between data in different schemas (either source schemas or Data Warehouse schema). Such specification is done through special assertions, called *Reconciliation Correspondences*.

Reconciliation Correspondences are defined in terms of relations, similarly to the case of the relations describing the sources and the Data Warehouse at the logical level. The difference with source and Data Warehouse relations is that we conceive Reconciliation Correspondences as non-materialized relations, in the sense that their extension is computed by an associated program whenever it is needed. In particular, each Reconciliation Correspondence is specified as an adorned query with an associated *program* that is called to compute the extension of the virtual relation. Note that we do not consider the actual code of the program but just its input and output parameters.

We distinguish among three types of correspondences, namely Conversion, Matching, and Merging Correspondences. A detailed description of these types of correspondences is reported in [19].

4. Methodology for Data Warehouse design

In this section we outline the proposed methodology for Data Warehouse design, which is based on the framework presented in the previous section. For a more detailed description of the methodology, we refer to [6,19,25]. The methodology identifies the following situations that need to be dealt with in a systematic manner in order to build and maintain the Conceptual Model, the Source Schemas and the Data Warehouse Schema:

- A new source (or, a new portion of a source) is analyzed and added to the whole architecture; the data integration design scenario corresponding to this situation will be called *source-centric*.

- A new query is formulated by a client; the design scenario corresponding to this situation will be called *query-centric*.

Note that, although the two scenarios refer to a dynamic situation where a new piece of information or a new request is considered, one can in practice iterate the process to design the Data Warehouse from scratch.

In the following, we describe the most important methodological aspects that are to be considered in the source-centric scenario, which is the approach adopted in the TELECOM ITALIA integration project.

4.1. Conceptual modeling

Source-centric Conceptual Model integration is triggered when a new source or a new portion of a source is taken into account for integration into the Data Warehouse. The steps to be accomplished in this case are the following:

(1) *Source Model construction*. The goal of this step is to produce the *Source Model*, i.e., the conceptual model of the source under analysis. Since in a typical Data Warehouse setting, sources already exist, this task may be accomplished through a reverse engineering activity [26].
(2) *Source Model integration*. The goal of this step is to integrate the constructed Source Model with the Conceptual Model. We remind the reader that our approach to integration is mainly declarative: the most important efforts during source integration are indeed devoted to single out and to specify the assertions relating the Conceptual Model and the Source Models, thus producing a unified conceptual representation. More precisely, the step of Source Model integration is characterized by the following activities:
   (a) *Conflict resolution*. The goal of this activity is to detect and solve structural and semantic conflicts involving the Source Model under analysis, to conform and align it to the Conceptual Model and make it compatible for integration. In general, conflict resolution cannot be fully automatized: human intervention by the designer is requested by the system when conflicts have to be solved.
   (b) *Defining inter-model assertions*. The resulting Source Model is added to the Conceptual Model, together with suitable inter-model assertions relating elements of the Source Model to elements of the Conceptual Model (in particular, these may be elements of previously integrated Source Models).
(3) *Quality analysis*. The resulting Conceptual Model is evaluated and its accordance with quality factors that are relevant at the conceptual level is evaluated [5]. The reasoning techniques associated to the formalism at the conceptual level provide essential support during this activity [6]. If necessary, a restructuring of the Conceptual Model is accomplished to match the required criteria.

4.2. Logical level and data integration

The methodology for the definition of the logical level of the Data Warehouse is based on the following steps:

(1) specification of Source and Data Warehouse Schemas;
(2) specification of Reconciliation Correspondences;
(3) specification of mediators.

The reasoning services provided at the logical level make it possible to use the specifications of Source and Data Warehouse Schema and of Reconciliation Correspondences to automatically generate the correct mediators for the loading of the Data Warehouse. This is realized by means of a query rewriting technique which uses query containment as its basic reasoning service.

4.2.1. Specifying Source and Data Warehouse Schema descriptions

The first two steps of the proposed methodology correspond to defining both the Data Warehouse Schema and each Source Schema in terms of adorned queries. While the design of Source Schemas does not arise particular problems, the

design of the Data Warehouse Schema can raise many issues due to restructuring and denormalization of the data. The choice of the data to materialize, and how to organize them in relations, is an important step in the construction of the Data Warehouse. Several aspects have to be taken into account in making these choices, for example the required storage amount, the cost of loading, refreshment, and query processing, etc. A methodology for identifying the relations to materialize that is coherent with our framework has been developed [27].

We point out that the mechanism used in our framework for specifying adorned queries is able to cope with *schematic differences* [28]. Indeed, as illustrated in [19], our framework allows for handling various forms of schematic differences, both among the sources, and between the sources and the Conceptual Model.

4.2.2. Specifying Reconciliation Correspondences

The task of specifying suitable Reconciliation Correspondences is a responsibility of the designer. Once such Reconciliation Correspondences are specified, they are profitably exploited to automatically generate mediators. In the task of specifying Reconciliation Correspondences the system can assist the designer in various ways.

First of all, since each Reconciliation Correspondence is declaratively specified as an adorned query, all reasoning tasks available for such queries can be exploited to check desirable properties of the correspondences. In particular, the system can check the consistency of queries, rejecting inconsistent ones and giving the designer useful feedback. Also, the system can automatically detect whether the adorned queries associated with different correspondences are contained in each other (or are equivalent). This is an indication of redundancy in the specification. However, to determine whether a correspondence is actually redundant, one has to consider also the types of the correspondences and the programs associated with them. For example, a less general query, thus specifying stricter conditions for applicability, may still be useful in the case where the associated program takes advantage of the specialization and operates more efficiently.

In practice, the system automatically asserts several correspondences *by default*, thus simplifying the task of the designer. We refer the reader to [19] for a detailed description of the correspondences automatically asserted by the system.

4.2.3. Specifying mediators

As mentioned, our goal is to provide support for the design of the mediator for a Data Warehouse relation T, i.e., the program that accesses the sources and loads the correct data into T. In general, the design of mediators requires a sophisticated analysis of the data, which aims at specifying, for every relation in the Data Warehouse Schema, how the tuples of the relation should be constructed from a suitable set of tuples extracted from the sources. Mediator design is typically performed by hand and is a costly step in the overall Data Warehouse design process. The framework presented here is also based on a detailed analysis of the data and of the information needs. However, the knowledge acquired by such an analysis is explicitly expressed in the description of source and Data Warehouse relations, and in the Reconciliation Correspondences. Hence, such a knowledge can be profitably exploited to support the design of the mediators associated to the Data Warehouse relations.

Suppose, we have decided to materialize in the Data Warehouse a new relation T, and let q be the adorned query specifying the data to materialize in T. Our technique requires to proceed as follows:

(1) We determine how the data in T can be obtained from the data stored in already defined Data Warehouse relations, the data stored in source relations, and the data returned by the programs that perform conversion, matching, and merging associated to the Reconciliation Correspondences. This is done by computing a *rewriting* of q in terms of the available adorned queries, i.e., a new query q' contained in q whose atoms refer to (i) the already available Data Warehouse relations, (ii) the source relations, and (iii) the available conversion, matching, and merging predicates.

(2) We specify how to deal with tuples computed by the rewriting and possibly representing the same information. Typically, we have to combine tuples coming from different sources to obtain the tuples to store in the Data Warehouse relations.
(3) We refine the rewriting returned by the algorithm. For example, certain conjuncts can be eliminated from the union of conjunctive queries according to suitable criteria for populating Data Warehouse relations. In particular, such criteria may be determined by factors that affect the quality of the data in the source relations and in the Data Warehouse, such as completeness, accuracy, confidence, freshness, etc.

The resulting query, which will be a disjunction of conjunctive queries, is the specification for the design of the mediator associated to T. The above steps are discussed in more detail in [19].

5. Enterprise modeling in Telecom Italia

In this section, we describe the development process of the Data Warehouses of Telecom Italia. Specifically, we deal with a portion of the Customer Primary Data Warehouse, and we specifically focus on the modeling of the dimension *Contract*. In the development we have carried out the following steps:

(1) building of the Conceptual Model by integrating the conceptual data models of the sources (cf. Section 4.1);
(2) definition of the contents of the Data Warehouse by specifying adorned queries expressed in terms of the Conceptual Model built in the previous step (cf. Section 4.2).

With regard to the first step of the methodology, as outlined in Section 2, the integration projects that have been undertaken in Telecom Italia have produced significant advantages for the development of the Enterprise Data Warehouse, among them, the availability of an integrated view of the enterprise data. While this constitutes a basis for the construction of the Conceptual Model, we have identified several conflicts in the modeling of the sources, which required restructuring operations in addition to the definition of inter-model assertions. These conflicts are due to different representations of the concepts in the various sources.

Our methodology presupposes the use of software tools to simplify conceptual data modeling activities. Specifically, we realized a software tool that can lay out and manage data models of the data sources and the Conceptual Model and provide reasoning services to check several properties of the models. We have used a standard design platform[4] for the design of the conceptual schema and have connected it to a reasoner module implemented on top of the Description Logics reasoning system FaCT [29]. The architecture is loosely coupled: each time a reasoning service is required by the designer, she explicitly invokes the corresponding functionality of FaCT.

In the rest of the section we discuss in detail the first step of our methodology, while the application of the second step is discussed in Section 6. Specifically, we describe now the process of integrating the data sources that have to feed the data to the Data Warehouse. As mentioned, this task can be split into various distinct phases. We notice that, since source data models are available, we do not need to construct them and we can focus our attention on the Source Model integration phase. Given the availability of an integrated view of the Enterprise we can build the Conceptual Model in a single step, because we can directly use the data models of all sources that we have planned to integrate. We remind that the output of this step is the integrated Conceptual Model, which expresses the information managed by the Enterprise. This model is the input to the data integration step.

We first describe the main data sources, then concentrate on the three main activities in source integration, namely (i) the design of the initial Conceptual Model, (ii) conflict resolution, and (iii) the definition of inter-model assertions. Finally, we

[4]To design and manage the data models we employed Cayenne GroundWorks, which is the enterprise tool adopted in Telecom Italia.

present the tool that was used to support these activities.

5.1. Data sources involved in the Customer Primary Data Warehouse

The Enterprise Information System of TELECOM ITALIA (cf. Section 2) must cope with numerous and extremely differentiated data sources. In particular, we have different applications based on different data management technologies (from hierarchical to object-relational DBMSs) that share information, or, in other words, that manage common concepts.

Due to the key role played by the customer in the TELECOM ITALIA business processes, the Customer Primary Data Warehouse involves a large number of Operational Data Stores. These Operational Data Stores can be partitioned into two main categories:

- data sources that have already been integrated (during the IBDA project); these sources, named BDAs, have an integrated Conceptual Model and are targeted to a particular subject; in most cases they are shared among different applications;
- data sources that are currently not integrated, called *legacy* data sources, which are managed by a single application and often lack a conceptual data model.

This separation has important consequences on data modeling and integration in the Data Warehouse building process.

The most important BDA data sources are the following:

DB09—Customers, which manages basic information about customers (e.g., name, address, type, main business activity if it applies, etc.);

DB19—Business-Units, which manages information about TELECOM ITALIA's business organization (e.g., business units, departments, etc.);

DB10—Contracts-and-Policies, which manages information about legal contracts and policies signed between TELECOM ITALIA and its customers and suppliers;

DB28—Service-Catalogue and *DB44—Product-Catalogue*, which describe services offered and products sold/rent by TELECOM ITALIA to customers with their fees/prices, special offers, marketing campaigns, etc.;

DB29—Service-Subscriptions, which manages information about the number and type of services subscribed and products rented by customers and their options.

The most important legacy data sources are the following:

ECIS—Enterprise-Customers, which is the application managing information about large customers (e.g., enterprises, government organizations, etc.);

RUM—Remote-Unit-Management, which is the main telecommunications network management application that interfaces the network devices to the information system;

ACM—Advanced-Customer-Management, which is the primary Customer Care front-end application that manages information about customer contacts and interactions with TELECOM ITALIA;

BCM—Breakdown-and-Complaint-Management, which is the application that manages information about service breakdowns and customer complaints;

NBP—New-Billing-Procedure and *IBS—Integrated-Billing-System*, which are the billing systems for the domestic and business customers, respectively.

There is a degree of overlapping of the information managed by the different systems. We have detected inconsistencies between information present in more than one system, and a number of source conflicts. We describe now how we have dealt with such conflicts by adopting the design methodology described in the previous section.

In the following examples, for the sake of brevity, we focus our attention only on a subset of the mentioned sources.

5.2. Conceptual Model

From the analysis of project requirements, we have come to the conclusion that concepts relevant

for the Enterprise and, consequently, for the Conceptual Model, can be successfully modeled by the Source Models of BDAs. Indeed, almost all concepts expressed in the legacy applications are subsumed by corresponding BDA concepts. However, there are a few concepts that are present in legacy applications, but are not modeled in BDAs: such concepts, which are currently irrelevant to the Data Warehouse, have been ignored but they can be integrated in future. Consequently, the Conceptual Model has been essentially built by unifying Source Models of BDAs.

5.3. Source conflict resolution

Throughout the process of Source Model integration, we have identified a large number of *conflicts*. This is due to the fact that the different Source Models have been developed independently, without employing homogeneous conventions on how to represent certain types of information. In fact, in TELECOM ITALIA, as in many large enterprises, each internal unit has great autonomy in the design of its own information sub-systems, which later have to be integrated in the Enterprise Data Warehouse. Moreover enterprise-wide design rules have changed over the years, thus systems developed at different time periods may not follow the same conventions. Since in TELECOM ITALIA we have two main types of data sources (BDAs and legacy applications), we have different kinds of conflicts between sources.

As an example, we discuss some of the integration problems that occurred when integrating a legacy data source, called *ECIS—Enterprise-Customers*, and some BDA data sources used for the design of the Customer Primary Data Warehouse. Specifically, we describe the conflict between the entities Fiscal_Code and VAT_Code.[5] The source *ECIS—Enterprise-Customers*, which manages information on large business customers, must cope with the case in which a single customer may own more than one code (fiscal or VAT).

Therefore, in this source, a code is represented separately from a customer, by using a different entity. Since the two types of codes serve the same purpose as far as the system *ECIS—Enterprise-Customers* is concerned, they have been modeled both by a unique entity. On the other hand, in *DB09—Customers* the two codes are represented by two different attributes of the entity Customer. Nevertheless, at the logical/physical level, the information about the codes is stored in two different tables; in fact, the system must maintain the history of different codes assigned to a customer.

To solve this conflict we have promoted in *DB09—Customers* the attributes of the entity Customer in two new independent entities, named VAT_Code_History and Fiscal_Code_History (Fig. 3). Instead, we did not consider it necessary to split the entity of *ECIS—Enterprise-Customers* representing the two codes into two separate entities.

5.4. Defining inter-model assertions

In the Conceptual Model we can distinguish between two kinds of inter-model assertions:

- assertions between BDA data sources and the Conceptual Model;
- assertions that involve legacy application data models.

Assertions of the former type are often straightforward to specify, whereas we need to analyze in more detail the latter.

5.4.1. Assertions between BDA Data Sources and the Conceptual Model

These assertions involve single concepts. In general, inter-model assertions link a so-called *boundary entity* to the corresponding *main entity* by means of an inclusion relation. The assertions between source models and the Conceptual Model link identical concepts, since the Conceptual Model is built from the BDA model; therefore,

[5]The fiscal code ('codice fiscale') is an identification code assigned by the Italian Ministry of Finance to every Italian citizen or enterprise, similar to an SSN. The VAT code ('partita IVA') is an identification code assigned to every Italian professional, enterprise, or organization. Despite the two codes serve both a similar purpose, namely identification, a mapping between them does not exist.

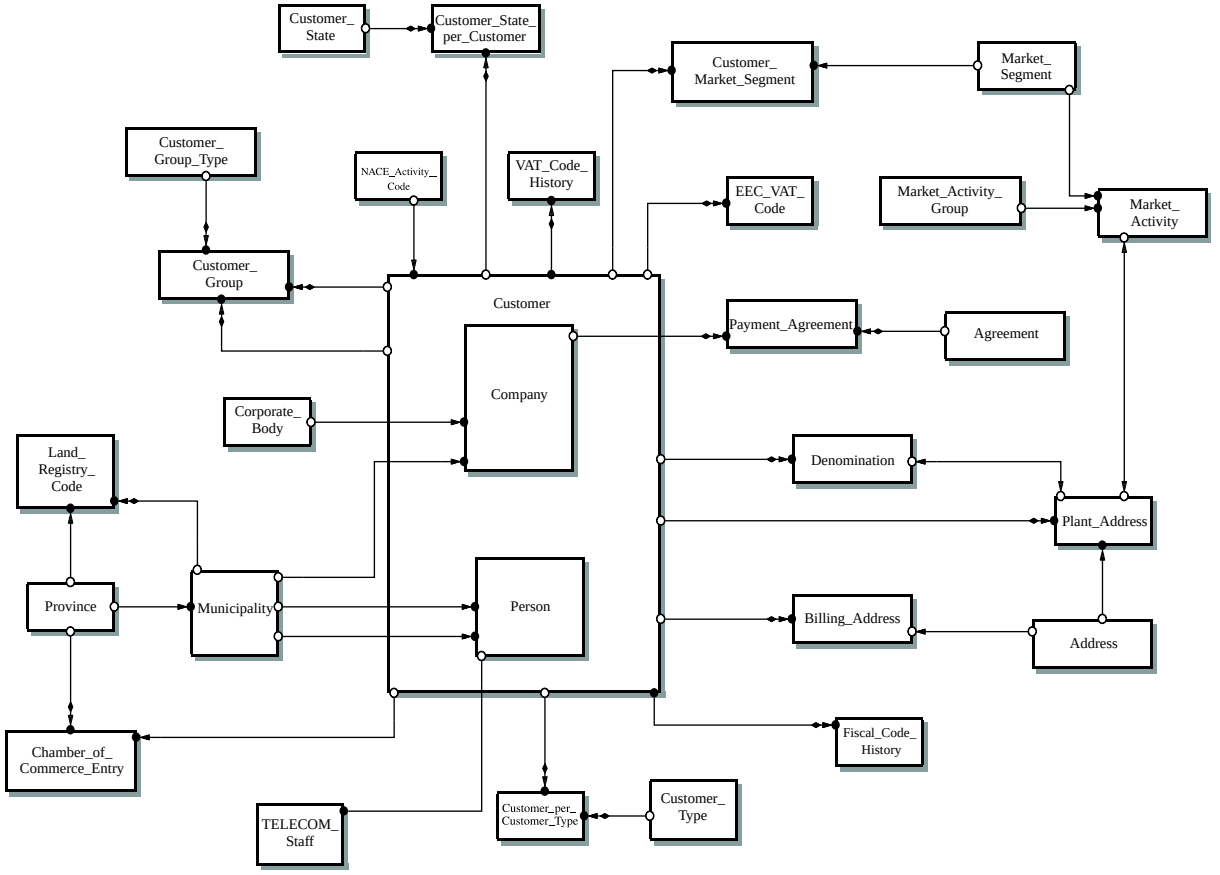

Fig. 3. Restructured conceptual data model of *DB09—Customers*.

we have only relations stating equivalence of two concepts.

5.4.2. *Assertions that involve data models of legacy applications*

Concepts of legacy applications have to be put in the proper correspondence with the related concepts in the BDA sources. This is often a difficult task. In fact, since concepts of legacy applications may represent aspects that are modeled in the BDA sources through more than one concept, we may need to express complex constraints holding between different source models.

We illustrate these difficulties on some examples. Fig. 4 shows the conceptual data model of the legacy application *ECIS—Enterprise-Customers*, while the inter-model assertions that involve *ECIS—Enterprise-Customers* are given in Table 1. In order to give an indication of the complexity of such a task, we note that the definition of the inter-model assertions of this legacy application has required the analysis of various hundreds candidate inter-model assertions between a dozen of BDA data source models and the remaining legacy applications.

- Assertion 1 is relative to the entity Appendix, whose instance maintains additional contract information. Every contract has a mandatory first appendix, defined as Contract_Appendix, that contains the amount and the payment method. Other types of appendices can be inserted as a consequence of a contract change (e.g., subscription to a new service) or of ad hoc clauses (e.g., technical support, maintenance, insurance, service level agreement). The same

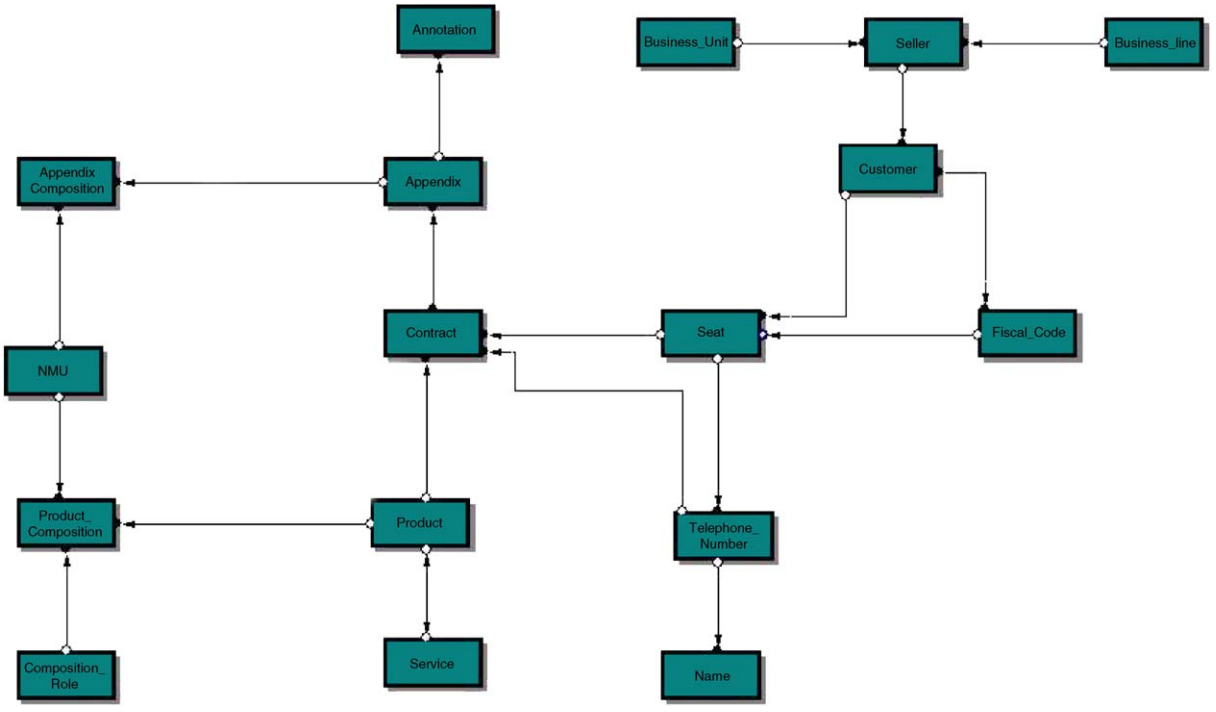

Fig. 4. Conceptual data model of *ECIS—Enterprise-Customers*.

Table 1
Inter-model assertions for the entities of the data source *ECIS—Enterprise-Customers*

$\mathsf{Appendix}_{\mathsf{ECIS}}$	$\equiv_{int}$	$(\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}} \sqcap \forall \mathsf{CHANGE}_{\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}}}.\mathsf{Standard\_Contract}_{\mathsf{DB10}}) \sqcup$ $(\mathsf{Standard\_Contract}_{\mathsf{DB10}} \sqcap \forall \mathsf{REFERENCE\_TO}_{\mathsf{Standard\_Contract}_{\mathsf{DB10}}}.\mathsf{Standard\_Contract}_{\mathsf{DB10}}) \sqcup$ $(\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}} \sqcap \forall \mathsf{MODIFY}_{\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}}}.\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}})$	(1)
$\mathsf{Fiscal\_Code}_{\mathsf{ECIS}}$	$\equiv_{ext}$	$\mathsf{Fiscal\_Code\_History}_{\mathsf{DB09}} \sqcup \mathsf{VAT\_Code\_History}_{\mathsf{DB09}}$	(2)
$\mathsf{Contract}_{\mathsf{ECIS}}$	$\equiv_{int}$	$\mathsf{Contract}_{\mathsf{DB10}}$	(3)
$\mathsf{Name}_{\mathsf{ECIS}}$	$\equiv_{ext}$	$\mathsf{Denomination}_{\mathsf{DB09}}$	(4)
$\mathsf{Seat}_{\mathsf{ECIS}}$	$\equiv_{ext}$	$\mathsf{Plant\_Seat}_{\mathsf{DB09}}$	(5)
$\mathsf{Business\_Unit}_{\mathsf{ECIS}}$	$\equiv_{ext}$	$\mathsf{Business\_Unit}_{\mathsf{DB19}}$	(6)
$\mathsf{Customer}_{\mathsf{ECIS}}$	$\sqsubseteq_{ext}$	$\mathsf{Customer}_{\mathsf{DB09}}$	(7)

concept is modeled in *DB10—Contracts-and-Policies* in a totally different way. First of all, we distinguish among three types of contracts: standard, management, and ad hoc contracts. Then, we separate main contracts, which state the subject of the agreement between TELECOM ITALIA and the customer, from secondary contracts, which specify changes to the main contract. From this specification we can deduce that every instance of $\mathsf{Appendix}_{\mathsf{ECIS}}$ is a secondary contract (standard or ad hoc) that specifies a change to a main contract (standard or ad hoc, too), previously signed.

- Assertion 2 is a consequence of the considerations exposed in Section 5.3. In *ECIS—Enterprise-Customers* the concepts related to the fiscal

and the VAT code are modeled as a single entity, while two different entities exist in the model of *DB09—Customers*. Consequently, the entity $\mathsf{Fiscal\_Code}_{\mathsf{ECIS}}$ is equivalent to the union of the entities $\mathsf{Fiscal\_Code\_History}_{\mathsf{DB09}}$ and $\mathsf{VAT\_Code\_History}_{\mathsf{DB09}}$.

- The entity $\mathsf{Contract}_{\mathsf{ECIS}}$ represents only contracts managed by the application *ECIS—Enterprise-Customers*. However, as specified by Assertion 3, the entities $\mathsf{Contract}_{\mathsf{ECIS}}$ and $\mathsf{Contract}_{\mathsf{DB10}}$ are conceptually equivalent.
- Assertion 4 is due to the fact that in *ECIS—Enterprise-Customers* the name inserted in the Yellow Pages corresponds to the concept Denomination of *DB09—Customers*.
- Assertions 5–7 are straightforward and do not need any comment.

5.5. Support tool

We present the tool that was used to support the conceptual modeling activity and some examples that show how it can effectively help the developing process. Our tool relies for the reasoning task on the FaCT reasoning system [29–31], which is a prototypical knowledge representation system based on Description Logics that uses an optimized subsumption algorithm based on tableaux. In particular, FaCT supports the typical reasoning tasks that can be performed on conceptual schemas expressed in $\mathcal{DLR}$. Notice that FaCT has been designed to support data modeling tasks, and so it provides concept-oriented functions. However, in its current version, it cannot be employed to make inference on instances of concepts.

We have integrated the FaCT reasoner as an add-on in the enterprise data modeling CASE tool Cayenne GroundWorks. Using the CASE tool, the designer can visually establish assertions between concepts of different data sources in the same way as he can establish a relation between two entities or a foreign key between two tables. We use a color based notation to assign each concept to the corresponding model and we represent assertions as links between concepts, as shown in Fig. 5. If the assertion is more complex than a simple inclusion or equivalence, the designer can annotate it with the corresponding constraint formula. A more detailed description of our support tool can be found in [32].

We note that the inherent complexity of reasoning on conceptual models expressed in $\mathcal{DLR}$ [33] has a noticeable impact on the performance of the

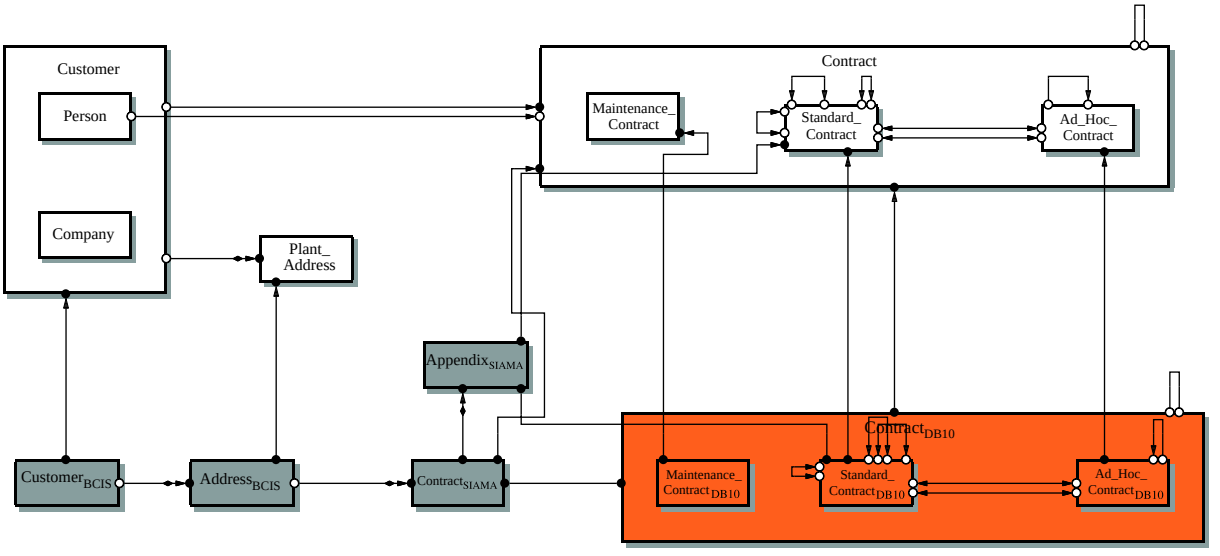

Fig. 5. Part of the Conceptual Model involved in the example with inter-model assertions.

modeling tool and the kind of support it can offer to the designer with respect to the problem of dealing with large scale conceptual models. Specifically, while FaCT, similar to other state-of-the-art Description Logic reasoners, is able to effectively reason in significant fragments of the Conceptual Model [34,35], it still cannot deal with very large models, such as the whole Conceptual Model as used in the Telecom Italia Data Warehousing applications. The support tool we have devised inherits from the reasoner the same limitations with respect to its ability of dealing with large-scale applications.

We show an example of the usage of the reasoner as a support tool in the conceptual design of the Data Warehouse. Fig. 5 shows a part of the Conceptual Model with some entities that belong to two data sources: *DB10—Contracts-and-Policies*, which is a BDA data source, and *ECIS—Enterprise-Customers*, which is a legacy application data source.

As we have noticed in Section 5.4, we must cope with a large number of inter-model assertions of various kinds. We report part of these assertions in Table 2. Through these assertions we can define the relations between the Conceptual Model and their sources, as discussed below:

- Assertions 8, 9, 11, 12, 13, 14 and 15 are simple inclusion relations between Conceptual Model entities and the corresponding source counterparts.
- Assertion 10 specifies that an instance of the entity $\mathsf{Appendix}_{\mathsf{ECIS}}$ can be:
  - o a Standard_Contract that modifies a Standard_Contract,
  - o an Ad_Hoc_Contract that modifies an Ad_Hoc_Contract, or
  - o an Ad_Hoc_Contract that modifies a Standard-Contract.
- Assertion 16, stating the conceptual equivalence between two entities in two sources, was already discussed.

We can make very interesting observations on this part of the Conceptual Model. Specifically, we have noticed the following property:

$$\mathsf{Appendix}_{\mathsf{ECIS}} \sqsubseteq_{ext} \mathsf{Standard\_Contract} \sqcup \mathsf{Ad\_Hoc\_Contract}$$

In other words, the set of instances of the entity $\mathsf{Appendix}_{\mathsf{ECIS}}$ is a subset of the union of the sets of instances of the entities Standard_Contract and Ad_Hoc_Contract. Moreover, being Standard_Contract and Ad_Hoc_Contract mutually exclusive, we cannot decide whether an instance of $\mathsf{Appendix}_{\mathsf{ECIS}}$ belongs to Standard_Contract or Ad_Hoc_Contract. We must cope with this issue if we need to load into the Data Warehouse information on the entities Standard_Contract and Ad_Hoc_Contract.

Table 2
Inter-model assertions involved in the example

$\mathsf{Customer}_{\mathsf{ECIS}}$	$\sqsubseteq_{ext}$	Customer	(8)
$\mathsf{Seat}_{\mathsf{ECIS}}$	$\sqsubseteq_{ext}$	Plant_Seat	(9)
$\mathsf{Appendix}_{\mathsf{ECIS}}$	$\sqsubseteq_{ext}$	$(\mathsf{Ad\_Hoc\_Contract} \sqcap \forall \mathsf{CHANGE}_{\mathsf{Ad\_Hoc\_Contract}}.\mathsf{Standard\_Contract}) \sqcup$ $(\mathsf{Standard\_Contract} \sqcap \forall \mathsf{REFERENCE\_TO}_{\mathsf{Standard\_Contract}}.\mathsf{Standard\_Contract}) \sqcup$ $(\mathsf{Ad\_Hoc\_Contract} \sqcap \forall \mathsf{MODIFY}_{\mathsf{Ad\_Hoc\_Contract}}.\mathsf{Ad\_Hoc\_Contract})$	(10)
$\mathsf{Contract}_{\mathsf{ECIS}}$	$\sqsubseteq_{ext}$	Contract	(11)
$\mathsf{Contract}_{\mathsf{DB10}}$	$\sqsubseteq_{ext}$	Contract	(12)
$\mathsf{Standard\_Contract}_{\mathsf{DB10}}$	$\sqsubseteq_{ext}$	Standard_Contract	(13)
$\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}}$	$\sqsubseteq_{ext}$	Ad_Hoc_Contract	(14)
$\mathsf{Maintenance\_Contract}_{\mathsf{DB10}}$	$\sqsubseteq_{ext}$	Maintenance_Contract	(15)
$\mathsf{Contract}_{\mathsf{ECIS}}$	$\equiv_{int}$	$\mathsf{Contract}_{\mathsf{DB10}}$	(16)

We have considered two solutions to this problem:

(1) restructuring the Conceptual Model by merging the entities Standard_Contract and Ad_Hoc_Contract;
(2) restructuring the data source, by introducing a new attribute in the entity $\mathsf{Appendix}_{\mathsf{ECIS}}$ to distinguish the instances that belong to Standard_Contract from those that belong to Ad_Hoc_Contract.

Obviously, the second solution is in general not acceptable, since it impacts on the structure of the Data Warehouse sources.

If we replace Assertion 10 with the following one

$$\mathsf{Appendix}_{\mathsf{ECIS}} \sqsubseteq_{ext} \mathsf{Standard\_Contract}$$

stating that the instances of entity $\mathsf{Appendix}_{\mathsf{ECIS}}$ are a subset of those of the entity Standard_Contract, the reasoner *cannot* derive anymore the following property:

$$\mathsf{Appendix}_{\mathsf{ECIS}} \sqsubseteq_{ext} \mathsf{Standard\_Contract}_{\mathsf{DB10}} \sqcup \mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}}$$

In other words, the set of instances of the entity $\mathsf{Appendix}_{\mathsf{ECIS}}$ is not necessarily a subset of the union of the sets of instances of the entities $\mathsf{Standard\_Contract}_{\mathsf{DB10}}$ and $\mathsf{Ad\_Hoc\_Contract}_{\mathsf{DB10}}$. This fact, together with the fact that $\mathsf{Appendix}_{\mathsf{ECIS}}$ is a subset of Standard_Contract, implies that we need the source *ECIS—Enterprise-Customers* to populate the Conceptual Model.

6. Data Warehousing in Telecom Italia

In this section, we describe the second step of the development process of a portion of the Customer Primary Data Warehouse of Telecom Italia. With regard to this step of the methodology, the design of the Data Warehouse has been carried out by employing a CASE tool for the logical level. In particular, we employed Platinum ER-Win,[6] supported by the prototype tool DaRe, developed within the DWQ project, which keeps track of conceptual links between data sources and helps to build mediator specifications by means of a query rewriting algorithm.

After designing the logical model of the Data Warehouse, according to our methodology, we have described, by the means of adorned queries, the contents of the data sources and of the Data Warehouse and we have introduced the required Reconciliation Correspondences. In the last phase, we have derived the specifications of the mediators that load the Data Warehouse.

In the following, we describe in more detail how the various phases of the Data Warehouse design methodology have been applied to the design of the Customer Primary Data Warehouse. We note that the design methodology has been applied also for the design of the Secondary Data Warehouses used for the Knowledge Discovery and Data Mining activities of Telecom Italia (cf. Section 7), although this activity is not reported here.

We illustrate the methodology for the design of the mediator that populates a Data Warehouse table on the example of the dimension *Contract* of the Customer Primary Data Warehouse. According to such a methodology, we follow the local-as-view paradigm and use the Conceptual Model, built in the previous step, to express by means of an adorned query the data that each data store contains. We then use the query rewriting algorithm, implemented in our tool, to design mediators that can populate a data store with information gathered from other data stores. In the case discussed below, we build the mediator that characterizes the loading of a portion of the Data Warehouse with the information extracted from the operational data sources. We want to emphasize again the key role that the Conceptual Model plays in our framework, since it allows to decouple the internal organization of each data source from the other ones.

We first describe the structure of a portion of the Customer Primary Data Warehouse, concentrating on the dimension relative to contracts.

6.1. The Customer Primary Data Warehouse

As mentioned in Section 2, the Customer Primary Data Warehouse is one of the Primary

[6] Platinum ER-Win is the standard tool adopted in Telecom Italia for such kind of tasks.

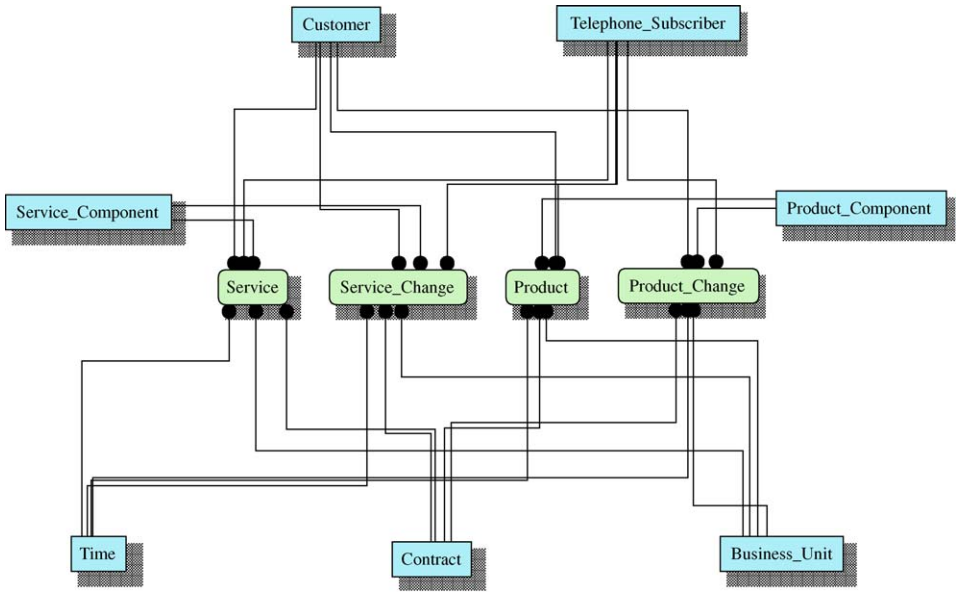

Fig. 6. Logical model of the Customer Primary Data Warehouse—*View CPDW—Customers*.

Data Warehouses, and it is divided into four *views*, concerning the representation of different aspects of the relation between the customer and the Enterprise:

Customers, which maintains information about customers and services/products that they have subscribed or bought.

Invoices, in which information about invoices and payments are stored.

Contacts, which contains data about contacts of customers with customer-care services and resellers network.

Complaints, which keeps information about customers complaints.

All the views have some common properties:

- they are designed according to the *star schema* data model with highly denormalized dimensions;
- they manage atomic information without aggregation or summarization.[7]

[7]We recall that in the Data Warehouse architecture of TELECOM ITALIA, aggregation and summarization tasks are delegated to Secondary Data Warehouses.

We recall that the dimension *Contract* belongs to the view *CPDW—Customers* and its logical data model is depicted in Fig. 6. The contents of this dimension is built on information about different versions of service activation contracts subscribed by customers of TELECOM ITALIA. So, contracts that are kept inside the Data Warehouse are relative to the supply of services of standard type (entity Standard_Contract).

For each version of a contract the data warehouse maintains the following data:

- the type of the contract (entity Acq_Contract_Type);
- the state, e.g., active, canceled, suspended (entity Contract_State);
- the business unit delegated to manage the contract (entity Business_Unit);
- if the contract is suspended, the type of cessation (entity Cessation_Type);
- the set of customer services managed by the contract (entity Customer_Service).

The start date for the validity of an instance (attribute *date_beg_val*) is the date when this instance has been inserted in the system, while

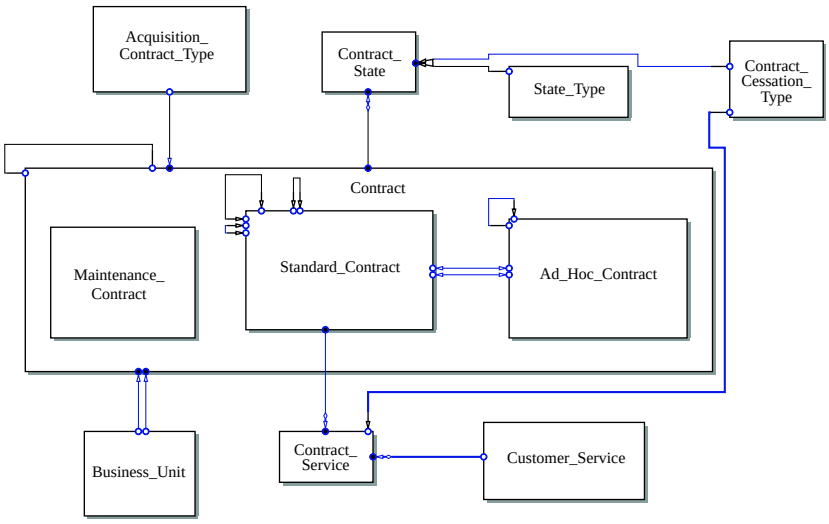

Fig. 7. Part of the Conceptual Model involved in the dimension *Contract*.

the end date for the validity (attribute *date_end_val*) is the start date for the validity of the subsequent version of the same contract, if it exists, or a conventional date in the future (constant MAXDATE).

The entity Contract_Service is the concept that characterizes this dimension. Consequently, it is identified by the Data Warehouse key *code_contr_dw*, and it connects the contract versions with the corresponding customer service versions.

6.2. Adorned query and specification of the mediator

Fig. 7 shows the portion of the Conceptual Model involved in the formulation of the adorned query that specifies the dimension *Contract* of the Customer Primary Data Warehouse. Such an adorned query, expressed in terms of the concepts of the Conceptual Model, is reported in Table 3. For conciseness, we have omitted some attributes that do not impact on the query structure significantly.

We shortly explain the structure of the adorned query. The head of the query corresponds to the relation schema that represents the dimension *Contract*, consequently it has an attribute for each field. The body of the query characterizes the content of these fields in terms of a query on the Conceptual Model. Since some attributes are significant only under certain conditions (e.g., the cessation type is defined only if the contract has been ceased), we make use of disjunction to express such a condition. In the adornment of the query we declare the abstract domain of each attribute and the identifier of each concept involved in the query.

For technical reasons, to increase performance, in the logical model of the Data Warehouse we introduce an alternative key for each dimension table, called a *data warehouse key*. In this case it is the field *code_contr_dw*. We build on this field the primary unique index of the dimension and we use this index to speed-up joins with the fact table defining the foreign key on this field rather that on the natural key of the dimension. In fact, we obtain that the record of the fact table is shorter because it contains only the values of the data warehouse key and the index used in the join is smaller.[8]

Since the values of the data warehouse key are generated by an ad hoc software procedure, we can express it by a conversion correspondence [19] that

[8]We also define a unique index on the natural key to ensure the integrity constraints, but it is not exploited by the query planner.

Table 3
Adorned query for the dimension *Contract*

$$
\begin{array}{l}
\mathsf{CONTRACT}_{\mathsf{DW}}(\\
\qquad\qquad code\_contr\_dw, date\_beg\_val, date\_end\_val, code\_contr, \\
\qquad\qquad code\_contr\_ver, date\_year, code\_acq\_contr\_type, desc\_acq\_contr\_type, \\
\qquad\qquad code\_contr\_state, code\_state\_type, desc\_state\_type, code\_cess\_type, \\
\qquad\qquad desc\_cess\_type, code\_bus\_unit, code\_customer\_ser, \\
\qquad\qquad code\_customer\_ser\_ver \\
\qquad\qquad) \leftarrow \\
\quad \mathsf{Standard\_Contract}(x), \\
\quad \mathsf{date\_yy}(x, date\_year), \ \mathsf{code\_contr\_ver}(x, code\_contr\_ver), \\
\quad \mathsf{code\_contr}(x, code\_contr), \ \mathsf{date\_ins}(x, date\_beg\_val), \\
\quad \mathsf{Business\_Unit}(y), \ \mathsf{ISSUED\_BY}(x, y), \ \mathsf{code\_bus\_unit}(y, code\_bus\_unit), \\
\quad \mathsf{Acq\_Contract\_Type}(z), \ \mathsf{CHARACTERIZED\_BY}(x, z), \\
\quad \mathsf{code\_acq\_contr\_type}(z, code\_acq\_contr\_type), \\
\quad \mathsf{desc\_acq\_contr\_type}(z, desc\_acq\_contr\_type), \\
\quad \mathsf{Contract\_State}(w), \ \mathsf{CHARACTERIZE}(w, x), \\
\quad \mathsf{code\_contr\_state}(w, code\_contr\_state), \\
\quad ((code\_cess\_type = \mathsf{NULL}, \ desc\_cess\_type = \mathsf{NULL}) \vee \\
\quad\ (\mathsf{Cessation\_Type}(v), \ \mathsf{IDENTIFIED\_BY}(w, v), \\
\quad\quad \mathsf{code\_cess\_type}(v, code\_cess\_type), \ \mathsf{desc\_cess\_type}(v, desc\_cess\_type))), \\
\quad \mathsf{State\_Type}(u), \ \mathsf{IDENTIFIED\_BY}(w, u), \\
\quad \mathsf{code\_state\_type}(u, code\_state\_type), \ \mathsf{desc\_state\_type}(u, desc\_state\_type), \\
\quad \mathsf{Contract\_Service}(r), \ \mathsf{MANAGED\_BY}(r, x), \\
\quad \mathsf{Customer\_Service}(s), \ \mathsf{REFERENCE\_TO}(r, s), \\
\quad \mathsf{code\_customer\_ser}(s, code\_customer\_ser), \\
\quad \mathsf{code\_customer\_ser\_ver}(s, code\_costumer\_ser\_ver), \\
\quad ((\mathsf{Standard\_Contract}(t), \ \mathsf{PRECEDE}(x, t), \ \mathsf{date\_ins}(t, date\_end\_val)) \vee \\
\quad\ date\_end\_val = \mathsf{MAXDATE}) \\
\quad | \ code\_contr\_dw :: \mathsf{CODE}_{\mathsf{DW}}, date\_beg\_val, date\_end\_val :: \mathsf{DATE}, \\
\qquad date\_year :: \mathsf{NOT\_STANDARD\_DATE}, \\
\qquad code\_contr\_ver, code\_contr, code\_bus\_unit, code\_acq\_contr\_type, code\_contr\_state, \\
\qquad code\_state\_type, code\_cess\_type, code\_customer\_ser, \\
\qquad code\_customer\_ser\_ver :: \mathsf{CODE}, \\
\qquad desc\_acq\_contr\_type, desc\_cess\_type, desc\_state\_type :: \mathsf{DESCRIPTION}, \\
\qquad Identify([code\_contr\_dw], r), Identify([code\_contr, code\_contr\_ver, date\_year], x), \\
\qquad Identify([code\_acq\_contr\_type], z), Identify([code\_contr\_state], w), \\
\qquad Identify([code\_state\_type], u), Identify([code\_cess\_type], v), \\
\qquad Identify([code\_bus\_unit], y), \\
\qquad Identify([code\_customer\_ser, code\_customer\_ser\_ver], s).
\end{array}
$$

links the value of the *natural key*, i.e., the key of the Conceptual Model, to the value of the data warehouse key through a program. The correspondence for the dimension *Contract* is shown in Table 4. We notice that the body of the conversion correspondence is a query that describes under which conditions the correspondence can be employed.

We want to emphasize that the Data Warehouse stores also some information, called *technical*, that

Table 4
Conversion correspondence for dimension *Contract*

$convert_{contract}($
$[code\_contr, code\_contr\_ver, date\_year, code\_customer\_ser,$
$code\_customer\_ser\_ver, date\_beg\_val], [code\_contr\_dw]$
$) \leftarrow$
Standard_Contract(x),
date_yy($x, date\_year$), code_contr_ver($x, code\_contr\_ver$),
code_contr($x, code\_contr$), date_ins($x, date\_beg\_val$),
Contract_Service(y), MANAGED_BY(y, x),
Customer_Service(z), REFERENCE_TO(y, z),
code_customer_ser($z, code\_customer\_ser$),
code_customer_ser_ver($z, code\_costumer\_ser\_ver$),
| $code\_contr, code\_contr\_ver, code\_customer\_ser, code\_customer\_ser\_ver$ :: CODE,
$date\_year$ :: NOT_STANDARD_DATE, $date\_beg\_val$ :: DATE,
$code\_contr\_dw$ :: $\mathsf{CODE_{DW}}$
through $CustomerServiceContractToDW([code\_contr, code\_contr\_ver,$
$date\_year, code\_customer\_ser, code\_customer\_ser\_ver, date\_beg\_val],$
$[code\_contr\_dw])$.

concerns the processes of loading and refreshing of the Data Warehouse and that we cannot express in terms of the Conceptual Model. In particular, we have the following technical information:

- the field *flag_val*, which has a boolean value and marks the instances that are currently active, in other words all the instances such that the current date is between the beginning and end date of their validity.[9]
- the field *date_ins*, which contains the date in which the instance has been loaded for the first time in the Data Warehouse; consequently, the value of such a field depends on when the loading job had been scheduled;
- the field *date_upd*, which keeps the date of the last update of the instance in the Data Warehouse, due to a transaction on the source system. In other words, it contains the date of execution of the loading job that has noticed a misalignment between information in the Data Warehouse and the corresponding data sources.

[9]The Conceptual Model lacks the concept of current date.

At the logical/physical level we must cope with updates of consolidated data in the Data Warehouse:

- when some information has been inserted late in the source system;[10]
- when a new version of a consolidated element is inserted, the previous version must be marked as invalid.

We have assigned to these fields the value NULL.

We cannot deal successfully with this kind of data because the Conceptual Model does not have the right concepts to model update data operations. In fact, in order to express the contents of the Data Warehouse completely and correctly we need a conceptual data model that is able to capture every transaction on the data sources. In other words, every insert, delete, or update operation at the logical/physical level must correspond to an appropriate update operation at the conceptual level. Such kind of information can be

[10]We have noticed several business processes that do not plan the immediate insertion of data in the Enterprise Information System.

more adequately managed within the specification of the ETL process, in order to separate data modeling issues from updating policies.

Since we have collected information on the data sources and on the Data Warehouse, and we have expressed it in terms of data models, adorned queries, and schema correspondences, we need to employ this knowledge to obtain the specification of the mediators that allow us to populate the Data Warehouse. The mediator specification is carried out by our support tool, described in Section 6.3, which implements the rewriting algorithm presented in [19].

We obtain the specification of the mediator that populates the relation $\mathsf{CONTRACT}_{\mathsf{DW}}$ with data extracted from some DBA data sources. The mediator specification is shown in Table 5 and in Fig. 8 we have shown, for clarity, the same specification expressed in SQL. The head of the mediator specification is the relation that the mediator is going to populate, and obviously coincides with the head of the adorned query that describes this relation. Instead, the body of the query is defined only in terms of relations (tables or views) of data sources that populate the Data Warehouse and Reconciliation Correspondences, while there is no reference to conceptual level elements. Notice that the rewriting algorithm has correctly introduced the program specified by the Conversion Correspondence to compute the values of the data warehouse key. We observe that we do not need to take special care when combining data coming from different sources, since the answers to different disjuncts of the query do not contain tuples that represent the same real world entity or the same value.

Table 5
Mediator specification for the dimensional *Contract*

$$
\begin{aligned}
&\mathsf{CONTRACT}_{\mathsf{DW}}(\\
&\qquad \mathit{code\_contr\_dw}, \mathit{date\_beg\_val}, \mathit{date\_end\_val}, \mathit{code\_contr},\\
&\qquad \mathit{code\_contr\_ver}, \mathit{date\_year}, \mathit{code\_acq\_contr\_type}, \mathit{desc\_acq\_contr\_type},\\
&\qquad \mathit{code\_contr\_state}, \mathit{code\_state\_type}, \mathit{desc\_state\_type}, \mathit{code\_cess\_type},\\
&\qquad \mathit{desc\_cess\_type}, \mathit{code\_bus\_unit}, \mathit{code\_customer\_ser},\\
&\qquad \mathit{code\_customer\_ser\_ver}\\
&\qquad) \leftarrow\\
&\quad \mathsf{STANDARD\_CONTRACT}_{\mathsf{DB10}}(\mathit{code\_contr\_dba}, \mathit{code\_contr\_ver},\\
&\qquad \mathit{date\_year}, \mathit{date\_begin\_val}, \mathit{code\_bus\_unit}, \mathit{code\_acq\_contr\_type},\\
&\qquad \mathit{code\_contr\_state}, \mathit{code\_cess\_type}, \mathit{code\_state\_type}, \ldots,\\
&\qquad \mathit{code\_contr\_dba\_next}, \mathit{code\_contr\_ver\_next}, \mathit{date\_year\_next}),\\
&\quad \mathsf{ACQ\_CONTRACT\_TYPE}_{\mathsf{DB10}}(\mathit{code\_acq\_contr\_type}, \mathit{desc\_acq\_contr\_type}, \ldots),\\
&\quad ((\mathit{code\_cess\_type} = \mathsf{NULL},\ \mathit{desc\_cess\_type} = \mathsf{NULL}) \vee\\
&\quad \mathsf{CESSATION\_TYPE}_{\mathsf{DB10}}(\mathit{code\_cess\_type}, \mathit{desc\_cess\_type}, \ldots)),\\
&\quad \mathsf{STATE\_TYPE}_{\mathsf{DB10}}(\mathit{code\_state\_type}, \mathit{desc\_state\_type}, \ldots),\\
&\quad ((\mathit{code\_contr\_dba\_next} = \mathsf{NULL},\ \mathit{code\_contr\_ver\_next} = \mathsf{NULL},\\
&\qquad \mathit{date\_year\_next} = \mathsf{NULL},\ \mathit{date\_end\_val} = \mathsf{MAXDATE}) \vee\\
&\quad \mathsf{STANDARD\_CONTRACT}_{\mathsf{DB10}}(\mathit{code\_contr\_dba\_next}, \mathit{code\_contr\_ver\_next},\\
&\qquad \mathit{date\_year\_next}, \mathit{date\_end\_val}, \ldots)),\\
&\quad \mathsf{CONTRACT\_SERVICE}_{\mathsf{DB29}}(\mathit{code\_contr\_dba}, \mathit{code\_contr\_ver}, \mathit{date\_year},\\
&\qquad \mathit{code\_customer\_ser}, \mathit{code\_customer\_ser\_ver}),\\
&\quad \mathit{CustomerServiceContractToDW}([\mathit{code\_contr}, \mathit{code\_contr\_ver},\\
&\qquad \mathit{date\_year}, \mathit{code\_customer\_ser}, \mathit{code\_customer\_ser\_ver}, \mathit{date\_beg\_val}],\\
&\qquad [\mathit{code\_contr\_dw}]).
\end{aligned}
$$

```
SELECT
        CustomerServiceContractToDW(X.code_contr, X.code_contr_ver,
        X.date_year, X.code_customer_ser, X.code_customer_ser_ver,
        X.date_beg_val) AS code_contr_dw,
        X.date_beg_val,
        COALESCE(W.date_beg_val, MAXDATE()) AS date_end_val,
        X.code_contr,
        X.code_contr_ver,
        X.date_year,
        X.code_acq_contr_type,
        Y.desc_acq_contr_type,
        X.code_contr_state,
        X.code_state_type,
        U.desc_state_type,
        X.code_cess_type,
        Z.desc_cess_type,
        X.code_bus_unit,
        V.code_customer_ser,
        V.code_customer_ser_ver
FROM
        DB10.STANDARD_CONTRACT AS X
        JOIN DB10.ACQ_CONTRACT_TYPE AS Y
          ON X.code_acq_contr_type = Y.code_acq_contr_type
        LEFT OUTER JOIN DB10.CESSATION_TYPE AS Z
          ON X.code_cess_type = Z.code_cess_type
        JOIN DB10.STATE_TYPE AS U
          ON X.code_state_type = U.code_state_type
        LEFT OUTER JOIN DB10.STANDARD_CONTRACT AS V
          ON U.code_contr_dba_next = V.code_contr_dba
          AND U.code_contr_ver_next = V.code_contr
          AND U.date_year_next = V.date_year
        JOIN DB29.CONTRACT_SERVICE AS W
          ON X.code_contr_dba = W.code_contr_dba
          AND X.code_contr_ver = W.code_contr
          AND X.date_year = W.date_year
```

Fig. 8. SQL version of the mediator specification for the dimension *Contract*.

6.3. Support tool

Within the DWQ project a prototype tool, named DaRe, has been developed that supports the data integration methodology described in this paper. We refer to [36] for a detailed description of the tool. Here, we briefly mention its main characteristics and report on its usage for the Data Warehouse design activities in Telecom Italia.

The tool is composed of two main components:

(1) a metadata repository that allows us to manage the conceptual and logical models of the data sources and of the Data Warehouse;
(2) a front-end application, interfaced with the repository, that helps in the construction of the mediators by exploiting query rewriting.

The repository is currently stored in a relational database (Oracle), nevertheless, the system is completely parametric with respect to the storage system, since its internal architecture is highly stratified. The front-end element has been developed in Java 2 and uses the Swing library to build the user interface and the JDBC library to implement its data access objects.

The tool provides the following functionalities:

- definition of the logical models of data sources;

- definition of the logical model of the Data Warehouse;
- definition of the mapping between the different levels by means of adorned queries;
- definition of Reconciliation Correspondences;
- automatic generation of the mediators.

For the construction of the Conceptual Model the tool relies on an external conceptual modeling tool (i.e., PLATINUM ER-WIN) that can interface with the metadata repository to access and store the Conceptual Model.

We describe how this tool supports the key activities of our data integration methodology. The working environment is subdivided in four components, one for each key step of the methodology:

Source Schema Specification, to manage the source data models and specify each Source Schema in terms of the Conceptual Model;

Data Warehouse Schema Specification, to manage the Data Warehouse Schema and the corresponding adorned queries;

Reconciliation Correspondences Specification, to establish and manage the Reconciliation Correspondences;

Derive Mediator, to derive the mediator specifications exploiting the query rewriting algorithm.

The adorned query design form is shown in Fig. 9.

The algorithm, implemented in the tool, that performs the automatic synthesis of the mediator through query rewriting, relies heavily on the ability to check query containment under the constraints imposed by the Conceptual Model. Such a task requires in general that the Description Logics reasoner is able to reason not only on concepts but also on individual objects [24]. Since FACT does currently not support this task, we have actually implemented an incomplete variant of the query containment test, in which objects are treated as concepts.

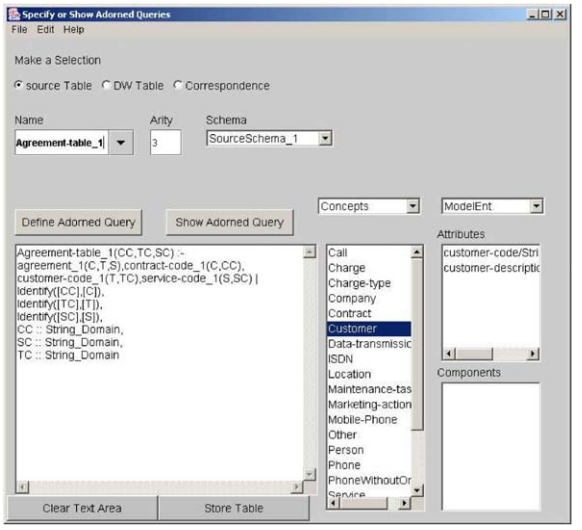

Fig. 9. DARE tool: adorned query definition form.

7. Impact on Data Mining and Decision Support applications

We now briefly describe the Decision Support System of TELECOM ITALIA and the Data Mining applications that we have developed on top of the Data Warehousing infrastructure. We focus our attention on the impact that our design methodology has on the whole business process that involves decision making activities.

The applications in the Decision Support System can be distinguished by their usage scenarios:

- applications that support tactical decisions (e.g., customer service, direct marketing, cross selling, up selling);
- applications that support strategical decisions (e.g., planning, strategical marketing, corporate government).

Both types of applications, which are developed on Secondary Data Warehouses according to the architecture described in Section 2, require different data analysis activities (e.g., OLAP, summarization, multi-dimensional analysis, enterprise reporting). We have implemented an OLAP customer analysis application using different data management techniques (relational, multi-dimensional, hybrid). Our results are presented in [37].

We have mainly concentrated on the development of analysis applications based on Data Mining techniques. In particular, we have

studied different business problems, like customer segmentation, customer behavioral analysis, and customer survival analysis. The techniques adopted for Knowledge Discovery and Data Mining (KDD) and some results of their application are reported in [38,39]. We explicitly mention an investigation about the relationship between traditional econometric tools and KDD marketing applications and a combined approach for telecommunications strategic marketing [40]. In the present work, we are not concerned with such applications, but with the impact of the above described methodology in the implementation of the Secondary Data Warehouses and of the data analysis applications.

We emphasize that the availability of an integrated conceptual data model, designed during previous tasks, helps to substantially reduce development costs for data analysis applications. In fact, one of the most critical aspects in the design of such applications is the definition and localization of information required to build the model. In particular, the use of the query rewriting algorithm in the design of mediators allows us to express the information needs of a data analysis application with respect to the Conceptual Model and to automatically obtain the queries to submit to the data sources (e.g., Primary Data Warehouses, Operational Data Stores) to retrieve the necessary information. Consequently, we can employ the methodology and the tools used to construct the Primary Data Warehouses in the design of Secondary Data Warehouses.

However, since data analysis often needs summarized information, or *measures*, we add a preliminary step to extend the Conceptual Model by introducing concepts that model these aggregations. This operation is, in some ways, similar to the design of multidimensional cubes in OLAP applications. In particular, a cube is modeled as a n-ary relationship between entities that correspond to the dimensions, while the measures are modeled as attributes of the relation.[11] Therefore, an integrated conceptual data model is not only essential to the definition of the Enterprise Information System, but it is also useful to cope with the knowledge that can be extracted from it.

A key factor in a Decision Support System is the ability to represent the knowledge acquired through the analysis of data. In fact, communicability of knowledge is essential in order to use it effectively. Our approach fully supports such a capability. Specifically, in the case of Data Mining applications we have that:

(1) the datasets (training, validation, and scoring) used while building models in the Knowledge Discovery process are expressed as views on the Conceptual Model;
(2) the learning process (e.g., decision trees and ILP) often produces sets of symbolic rules expressed on the schema of the training data.

As a result, one can employ rewriting techniques to express the knowledge extracted from the data directly with respect to the Conceptual Model. In this way, we can reduce the effort needed to build and maintain the knowledge about these rules. Additionally, since the whole Enterprise shares an integrated data model, we can easily disseminate the data analysis results.

8. Conclusions

In this paper we have described the approach and methodologies applied in TELECOM ITALIA for the development of Data Warehouse applications. The methodologies have been developed within the framework of the Esprit Research Project DWQ and tailored to the specific needs of TELECOM ITALIA. The experience gained can be summarized as follows.

A *data integration based approach to Data Warehousing is a critical success factor* to build a quality Enterprise Data Warehouse, both in terms of quality of the product (applications and handled data) and in terms of quality of the process.

Integration and Data Warehouse construction are incremental. Integration is an incremental process both because, from the organizational point of

[11] Although this information, being aggregated data, may not be directly available in the sources, we can model these aspects using Reconciliation Correspondences.

view, it is unlikely to be able to merge all the data sources in one step, and because the Data Administrator needs to incorporate new data sources that continuously become available. The construction of a Data Warehouse is an incremental process as well, especially in a scenario where a Data Warehouse structure including several levels of organization is adopted, as it is the case in enterprises of the size of TELECOM ITALIA. The incremental nature of the approach to both Data Integration and Data Warehousing deeply influences the design of methodologies and tools supporting these tasks. In particular, an incremental process is well supported by the local-as-view approach to Data Integration.

Data Warehousing application development is greatly enhanced by the availability of an integrated Enterprise Conceptual Model, which gives to the enterprise information assets a meaningful interpretation that is shared among application developers and business specialists. The Enterprise Conceptual Model plays also a key role in the Knowledge Management process, since it enables the building of a coherent Enterprise Knowledge Base, thus promoting the discovery, representation and dissemination of knowledge at the enterprise level.

The *investments required for the development of methodology custom support tools* and their integration with market CASE tools and DBMSs are significant. On the other hand, the mapping translation and code generation facilities based on automated reasoning enable in most cases the *rapid prototyping* of Data Warehouse applications. In fact, the Data Warehouse designer is able to easily obtain a prototype that can be employed to test and validate the design, even involving final users of Decision Support Systems, since it is possible to quickly build reports, decision dash boards, and pivot tables for testing purposes. This is a noticeable advantage in the design of such kinds of applications, where it is very difficult to validate user requirements before a late stage of development.

A relevant issue of our framework is represented by the *need to build and maintain the Enterprise Conceptual Model and related specifications*, or, in other words, to accurately manage the lifecycle of metadata. In a large enterprise like TELECOM ITALIA, it could become a serious problem, since a large number of information systems are coexisting and evolving independently, and metadata quality is strongly dependent on its freshness. Consequently, there is the need to build and disseminate a metadata management methodology, supported by adequate tools. In our framework, the declarative specification of constraints and mappings may alleviate, in part, this problem. Nevertheless, since the modeling and integration tasks are performed mainly at the conceptual level, we have experimented a higher degree of stability of the model with respect to technology and implementation changes, which usually have an impact on the logical/physical level. Not only, the Enterprise Conceptual Model is more stable than application models, since it depends mainly on the domain of interest. Thus, it can be adopted as a guide to change management and migration processes (i.e., migration from a legacy application to an ERP package), providing a sound foundation of the data item definitions that is substantially independent from their implementation.

An *approach centered around the Enterprise Conceptual Model* gives a number of advantages in terms of building cooperative solutions during the whole application lifecycle. In fact, most software development projects are mainly concerned with the integration of existing solutions (i.e., exporting business logic in terms of web services). So a rich common data definition shared between different project teams, which in a large corporation like TELECOM ITALIA probably belong to different departments or suppliers, helps to reduce interfacing errors and misunderstandings, improving quality and time-to-market of applications.

We have also experimented a *positive influence on data quality assessment tasks*, which are usually performed during the Data Warehouse building process. A formal and unique data definition on all Enterprise Information System levels greatly helps to identify data quality problems and their causes, and to develop cleaning strategies. Not only the ability of keeping track of inter-schema dependencies allows us to maintain

control of data redundancy in the Operational Data Store.

Appendix A. The Description Logic $\mathcal{DLR}$

$\mathcal{DLR}$ belongs to the family of *Description Logics*, introduced and studied in the field of Knowledge Representation [33,41,42]. Generally speaking, Description Logics are class-based representation formalisms that allow one to express several kinds of relationships and constraints (e.g., subclass constraints) holding among classes. Specifically, $\mathcal{DLR}$ includes:

- *concepts*, which are used to represent entity types (or simply *entities*), i.e., sets of *conceptual objects* having common properties. Besides atomic concepts, $\mathcal{DLR}$ allows for constructing complex concepts by means of specific operators: *conjunction* (denoted $\sqcap$), *disjunction* (denoted $\sqcup$), *negation* (denoted $\neg$), forms of *quantification*, and *numeric restrictions*;
- *n-ary relationships*, which are used to represent relationship types (or simply *relationships*), i.e., sets of tuples, each of which represents an association between conceptual objects belonging to different entities. The participation of conceptual objects in relationships models properties that correspond to relations with other conceptual objects. As for concepts, $\mathcal{DLR}$ allows for constructing complex relationships by means of *conjunction*, *disjunction*, *difference* (written using conjunction and negation), and *selection on one component* (denoted $(\$i : C)$, where i is a component and C is a concept);
- *attributes*, which are used to associate conceptual objects (or tuples of conceptual objects) with properties expressed by values belonging to one of several *domains*.

The Conceptual Model for a given application is specified by means of a set of *inclusion assertions*, written in the form

$$C_1 \sqsubseteq_{ext} C_2, \quad R_1 \sqsubseteq_{ext} R_2,$$

where C_1 and C_2 are two concepts and R_1 and R_2 are two relationships (of the same arity). The inclusion assertion $C_1 \sqsubseteq C_2$ states that each object that is an instance of C_1 is also an instance of C_2 (similarly for tuples in relationships). In the following, we make also use of *equality assertions* $C_1 \equiv C_2$, which can be considered as an abbreviation for a pair of inclusion assertions $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$ (similarly for relationships).

Inclusion assertions, used in combination with the concept and relationship constructs of $\mathcal{DLR}$, provide a powerful mechanism for expressing various forms of inter-relationships between concepts, relationships, and attributes. In particular, $\mathcal{DLR}$ allows for expressing:

- inclusion (i.e., ISA) and equivalence relationships between entities and relationships;
- disjointness and covering constraints, and, more generally, boolean combinations between entities and relationships;
- universal and existential qualification of concepts and relationship components;
- participation and functionality constraints, and more complex forms of cardinality constraints;
- definitions (expressing necessary and sufficient conditions) of entities and relationships in terms of other entities and relationships;

These features make $\mathcal{DLR}$ powerful enough to express not only the ER model, but also other conceptual, semantic, and object-oriented data models. Consequently, the proposed approach could be pursued also by adopting a conceptual modeling framework not based on the Entity-Relationship formalism. Moreover, $\mathcal{DLR}$ assertions provide a simple and effective declarative mechanism to express, by means of so-called *inter-model assertions* [20,43], the dependencies that hold between entities and relationships in different sources [44]. The use of inter-model assertions allows for an incremental approach to the integration of the conceptual models of the sources and of the enterprise [6,20]. We distinguish between two kinds of inter-model assertions: extensional and intensional ones (cf. [20,43]).

- Intuitively, an *extensional inter-model assertion*, denoted with $\sqsubseteq_{ext}$ (resp., $\equiv_{ext}$), is significant at

the instance level. Formally, such an assertion is interpreted as set-inclusion (resp., equality), exactly as an assertion in a single conceptual model: a concept is extensionally included in another concept if and only each instance of the first concept is also an instance of the second one.

- An *intensional inter-model assertion*, denoted with $\sqsubseteq_{int}$ (resp., $\equiv_{int}$), is used when two entities or relations in different sources represent the same concepts, but do not necessarily have coinciding sets of instances. Formally, a concept of a certain source is intentionally equivalent to another concept of a different source, if the extensions of the two concepts coincide on the sets of objects that are common to both sources (similarly for an intensional inclusion assertion).

One distinguishing feature of $\mathcal{DLR}$ is that sound and complete algorithms for reasoning over a set of logical assertions are available [24]. By exploiting such algorithms, one gains the possibility of reasoning about the Conceptual Model. In particular, one can automatically verify the following properties:

- *consistency of the Conceptual Model*, i.e., whether there exists a database satisfying all constraints expressed in the Conceptual Model;
- *entity* (*respectively*, *relationship*) *satisfiability*, i.e., whether there exists a database that satisfies the Conceptual Model in which a certain entity (respectively, relationship) has a non-empty extension;
- *entity* (*respectively*, *relationship*) *subsumption*, i.e., whether the extension of an entity (respectively, relationship) is a subset of the extension of another entity (respectively, relationship) in every database satisfying the constraints imposed by the Conceptual Model;
- *constraint inference*, i.e., whether a certain constraint holds for all databases satisfying the Conceptual Model.

Such reasoning services support the designer in the construction process of the Conceptual Model: they can be used, for instance, for inferring inclusion between entities and relationships, and detecting inconsistencies and redundancies.

References

[1] M. Lenzerini, Data integration: a theoretical perspective, in: Proceedings of the 21st ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002), 2002, pp. 233–246.
[2] W.H. Inmon, Building the Data Warehouse, 2nd ed., Wiley, New York, 1996.
[3] G. Wiederhold, Mediators in the architecture of future information systems, IEEE Computer 25 (3) (1992) 38–49.
[4] J.D. Ullman, Information integration using logical views, in: Proceedings of the Sixth International Conference on Database Theory (ICDT'97), vol. 1186, Lecture Notes in Computer Science, Springer, Berlin, 1997, pp. 19–40.
[5] M. Jarke, M. Lenzerini, Y. Vassiliou, P. Vassiliadis (Eds.), Fundamentals of Data Warehouses, Springer, Berlin, 1999.
[6] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Information integration: conceptual modeling and reasoning support, in: Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS'98), 1998, pp. 280–291.
[7] A.Y. Halevy, Answering queries using views: a survey, Very Large Database J. 10 (4) (2001) 270–294.
[8] O.M. Duschka, M.R. Genesereth, A.Y. Levy, Recursive query plans for data integration, J. Logic Programming 43 (1) (2000) 49–73.
[9] F.N. Afrati, M. Gergatsoulis, T. Kavalieros, Answering queries using materialized views with disjunction, in: Proceedings of the Seventh International Conference on Database Theory (ICDT'99), vol. 1540, Lecture Notes in Computer Science, Springer, Berlin, 1999, pp. 435–452.
[10] R. Pottinger, A.Y. Levy, A scalable algorithm for answering queries using views, in: Proceedings of the 26th International Conference on Very Large Data Bases (VLDB 2000), 2000, pp. 484–495.
[11] G. Zhou, R. Hull, R. King, Generating data integration mediators that use materializations, J. Intell. Inform. Systems 6 (1996) 199–221.
[12] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J.D. Ullnan, V. Vassalos, J. Widom, The TSIMMIS approach to mediation: data models and languages, J. Intell. Inform. Systems 8 (2) (1997) 117–132.
[13] R. Yereni, C. Li, H. Garcia-Molina, J.D. Ullman, Computing capabilities of mediators, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1999, pp. 443–454.
[14] C.H. Goh, S. Bressan, S.E. Madnick, M.D. Siegel, Context interchange: new features and formalisms for the intelligent integration of information, ACM Trans. Inform. Systems 17 (3) (1999) 270–293.
[15] S. Bergamaschi, S. Castano, M. Vincini, D. Beneventano, Semantic integration of heterogeneous information sources, Data Knowledge Eng. 36 (3) (2001) 215–249.

[16] D. Ursino, Extraction and Exploitation of Intensional Knowledge from Heterogeneous Information Sources, vol. 2282, Lecture Notes in Computer Science, Springer, Berlin, 2002.
[17] A. Calí, D. Calvanese, G. De Giacomo, M. Lenzerini, P. Naggar, F. Vernacotola, IBIS: Data integration at work, in: Proceedings of the 10th Italian Conference on Database Systems (SEBD 2002), 2002, pp. 291–298.
[18] J. Widom, Research problems in data warehousing, in: Proceedings of the Fourth International Conference on Information and Knowledge Management (CIKM'95), 1995.
[19] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Data integration in data warehousing, Int. J. Cooperative Inform. Systems 10 (3) (2001) 237–271.
[20] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Description logic framework for information integration, in: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), 1998, pp. 2–13.
[21] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Source integration in data warehousing, in: Proceedings of the Ninth International Workshop on Database and Expert Systems Applications (DEXA'98), IEEE Computer Society Press, Silver Spring, MD, 1998, pp. 192–197.
[22] S.M. Trisolini, M. Lenzerini, D. Nardi, Data integration and warehousing in Telecom Italia, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1999, pp. 538–539.
[23] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Use of the reconciliation tool at Telecom Italia, Technical Report DWQ-UNIROMA-007, DWQ Consortium (October 1999).
[24] D. Calvanese, G. De Giacomo, M. Lenzerini, On the decidability of query containment under constraints, in: Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98), 1998, pp. 149–158.
[25] D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, R. Rosati, Schema and data integration methodology for DWQ, Technical Report DWQ-UNIROMA-004, DWQ Consortium (September 1998).
[26] C. Batini, S. Ceri, S.B. Navathe, Conceptual Database Design, an Entity-Relationship Approach, Benjamin and Cummings Publ. Co., Menlo Park, CA, 1992.
[27] D. Theodoratos, S. Ligoudistianos, T. Sellis, Designing the global Data Warehouse with SPJ views, in: Proceedings of the 11th International Conference on Advanced Information Systems Engineering (CAiSE'99), 1999.
[28] A. Sheth, V. Kashyap, So far (schematically) yet so near (semantically), in: Proceedings of the IFIP DS-5 Conference on Semantics of Interoperable Database Systems, Elsevier Science Publishers (North-Holland), Amsterdam, 1992.
[29] I. Horrocks, Using an expressive description logic: FaCT or fiction? in: Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98), 1998, pp. 636–647.
[30] I. Horrocks, The FaCT system, in: H. de Swart (Ed.), Proceedings of the Second International Conference on Analytic Tableaux and Related Methods (TABLEAUX'98), vol. 1397, Lecture Notes in Artificial Intelligence, Springer, Berlin, 1998, pp. 307–312.
[31] I. Horrocks, FaCT and iFaCT, in: Proceedings of the 1999 Description Logic Workshop (DL'99), CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999, pp. 133–135.
[32] D. Calvanese, R. Capitini, M. Lenzerini, D. Nardi, R. Rosati, V. Stancati, Strumenti di ausilio alla modellazione dei dati nella progettazione di Data Warehouse, Technical Report 2-98, Dipartimento di Informatica e Sistemistica Università di Roma "La Sapienza"—TELECOM Italia, 1998 (in Italian).
[33] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, P.F. Patel-Schneider (Eds.), The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, Cambridge, 2003.
[34] D. Berardi, D. Calvanese, G. De Giacomo, Reasoning on UML class diagrams using description logic based systems, in: Proceedings of the KI'2001 Workshop on Applications of Description Logics, CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-44/, 2001.
[35] D. Berardi, Using description logics to reason on UML class diagrams, in: Proceedings of the KI'2002 Workshop on Applications of Description Logics, CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-63/, 2002.
[36] D. Calvanese, D. Lembo, M. Lenzerini, D. Nardi, Strumento di integrazione e riconciliazione dei dati nei Data Warehouse, Technical Report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"—TELECOM Italia, 2000 (in Italian).
[37] D. Caffio, D. Nardi, R. Rosati, Modellazione multidimensionale dei dati, Technical Report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"—TELECOM Italia, 1999 (in Italian).
[38] L. Dragone, D. Nardi, L. Tosco, Metodi e tecniche di Data Mining, Technical Report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"—TELECOM Italia, 1999 (in Italian).
[39] L. Dragone, D. Nardi, Sviluppo di applicazioni di Data Mining, Technical Report, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza"—TELECOM Italia, 2000 (in Italian).
[40] S. Cazzella, L. Dragone, S.M. Trisolini, Telecommunications strategic marketing—KDD and economic modeling, in: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), IEEE Computer Society Press, Silver Spring, MD, 2002.
[41] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, Reasoning in description logics, in: G. Brewka (Ed.), Principles of Knowledge Representation, Studies in Logic, Language and Information, CSLI Publications, 1996, pp. 193–238.

[42] A. Borgida, Description logics in data management, IEEE Trans. Knowledge Data Eng. 7 (5) (1995) 671–682.
[43] T. Catarci, M. Lenzerini, Representing and using interschema knowledge in cooperative information systems, J. Intell. Cooperative Inform. Systems 2 (4) (1993) 375–398.
[44] R. Hull, Managing semantic heterogeneity in databases: a theoretical perspective, in: Proceedings of the 16th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'97), 1997, pp. 51–61.