Document Number:	DWQ-UNIROMA-003
Project Name:	Foundations of Data Warehouse Quality (DWQ)
Project Number:	EP 22469
Title:	**Experimentation with the incremental view integration**
Authors:	Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati
Workpackage:	WP3
Document Type:	Deliverable (D.3.3)
Classification:	Confidential
Distribution:	DWQ Consortium
Status:	Final
Document file:	wp3-3.tex
Version:	1.0
Date:	September 20, 1998
Number of pages:	20

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	September 20, 1998	Final version

1 Introduction

Data Warehouse applications are current practice for many organizations, because the analysis of data can improve the strategic and management decision process, which are key tasks within any enterprise.

While there is a great demand for Data Warehouse applications, techniques and methods cannot adequately satisfy the needs. One major factor contributing to this situation is the difficulty of capitalizing the efforts made in the design of one application for subsequent applications, possibly sharing portions of the data. In fact, even when there is the interest to build and maintain an integrated enterprise data repository, the dynamics of the enterprise make it infeasible to provide all the data for Data Warehouse applications from a stable enterprise repository.

The development of Data Warehouse applications in Telecom Italia, the major italian telephone company, is one example of this scenario. A very large and successful project on integration has lead to an integrated enterprise database providing access to the company data. However, since in such a big enterprise there is still a large degree of development that strategically takes place in the various internal divisions, a stable situation has been achieved where some of the applications are legacy ones, even though the most part is integrated. Therefore, while the enterprise integration effort provides a large integrated basis, integration is continuosly catching up with new data, coming both from internal applications and often from external sources.

As for Data Warehouse applications in Telecom Italia, they are built with very strong time constraints, by many project teams and with requirements for data often changing. This practical situation gives rise to a significant duplication of efforts, between various Data Warehouse applications and between Data Warehouse development and enterprise integration. Such considerations have motivated a specific interest in the methodologies and tools developed within the DWQ project and, in particular, in the framework for conceptual Data Warehouse modeling developed at the University of Rome "La Sapienza".

In order to provide an experimental testbed for the methodology for conceptual Data Warehouse modeling developed in the previous stage of the project [CDGL$^+$97], we have identified, among the current Data Warehouse projects in Telecom Italia, the application called 'Customer Care', for collecting the information concerning assistance for customers, to be used both for analyzing the services offered by the company and to assist the operator answering customer calls. Several reasons, in addition to internal considerations, justify such a choice. In particular, such an application combines the requirements of data analysis together with more traditional development functions. Such a situation, which is very common in practice, is especially interesting for an integration approach to the design of Data Warehouse applications. Moreover, given that the application was at an intermediate development stage, we could refer both to the source driven and to the client driven scenarios outlined in [CDGL$^+$97].

The goal of this report is to describe the tools that we have developed to support our methodology for conceptual Data Warehouse modeling, and the application of the proposed methodology within a practical setting constituted by the 'Customer Care' project in Telecom Italia.

Obviously, the study reported in this paper, although referring to a real applicative project, is suitably adapted to respect the privacy about the company internal data and policies.

The document is organized as follows. We first describe the tool that we have developed to support the stage of conceptual Data Warehouse model construction. Then, we discuss the relevant features of the conceptual Data Warehouse model defined for the 'Customer Care' application and, finally, we present the subsystem providing the basic reasoning services and discuss some examples where such reasoning capabilities can be effectively used to improve the design of the conceptual Data Warehouse model.

2 A prototype for conceptual modeling

The methodology described in [LNRS98] requires the usage of software tools which allow for supporting and simplifying the conceptual modeling phase. For this reason, our aim was to use a system that allows for handling the Source Model, the Enterprise Model, and the Conceptual Data Warehouse Model, and that can be easily interfaced with the subsystem which analyzes schema quality. Moreover, an inference engine is necessary in order to perform the reasoning tasks needed for verifying quality criteria and for elaborating the Data Warehouse schema.

As for the first point, we had to satisfy both the need for an efficient acquisition of the data coming from the applications in Telecom Italia, and to use a powerful graphical environment. Based on such considerations, we have worked with GroundWorks©, a tool by Cayenne Software. The reasons for the choice of GroundWorks are better explained by the following remarks.

- Within the DWQ project, UNIROMA1 has recently developed a prototype for conceptual modeling based on the use of ConceptBase. However, at the beginning of the experimentation with Telecom Italia, such a prototype had not been already implemented.
- Designers in Telecom Italia use GroundWorks for the conceptual data modeling phase, therefore they desired to use a tool with the same GroundWorks interface and with the same input formats.
- However, the integration methodology we have experimented in Telecom Italia does not depend in any way on this prototype, which means that it can be applied in the same way using a different tool for the conceptual modeling phase (e.g. ConceptBase).

In this section we focus on the basic functionalities that we have added to GroundWorks in order to build the Conceptual Data Warehouse Model through the graphical interface. In Section 4 we shall discuss the implementation of reasoning services.

2.1 Graphical construction of Conceptual Data Warehouse Model

Even though GroundWorks does not provide the ability of changing its interface, it is however possible to define a convention in the usage of the constructs provided by the system, in order to allow for both modeling multiple source models and intermodel assertions, and visualizing the Conceptual Data Warehouse Model in a satisfactory way. Since the graphical notation used by GroundWorks does not have any built-in capabilities for conceptual Data Warehouse modeling, we have introduced a specialized notation which makes use of expressions in the language of Description Logics.

For the visualization of the Conceptual Data Warehouse Model, we use colouring of the entities, in order to distinguish the models of different sources and the enterprise model. In this way, it is also possible to easily see the intermodel assertions, since such relationships connect entities from different models, hence such entities have different colours. Moreover, we adopt the convention to use a fixed colour (white) for the enterprise model, which allows for identifying entities belonging to such a model during the translation phase, where these entities are treated in a special way.

Furthermore, we have implemented a method for renaming entities and relationships from the various models that have to be inserted into the Conceptual Data Warehouse Model. This is needed for the following reasons:

- GroundWorks does not allow the presence of more than one entity with the same name. Since the Conceptual Data Warehouse Model is obtained by iteratively merging several conceptual models, if two entities belonging to different models have the same name, an error arises when merging such models. Therefore it is necessary to provide different names for entities;
- GroundWorks allows for giving the same name to relationships between different entities. This causes an ambiguity which is problematic (and unacceptable) both when writing intermodel assertions, and when reasoning about the conceptual representation of the Data Warehouse.

We rename entities and relationships by labelling them with the source they belong to, in the following way:

- each source model is associated with a suffix, and such a suffix is added to the name of each entity in the model. For instance, the suffix “SourceA” is added to all entities in the Source Model “SourceA”;
- the same kind of renaming is applied to relationships. In this case the string which is added as suffix is formed by concatenating the names of the entities (modified according to the above point) involved in the relationship.

We now describe the translation of intermodel assertions. Since in GroundWorks it is only possible to connect entities by means of usual (binary) relationships of the ER model, we decided to exploit the use of the information about the cardinality of the relationships in order to specify the kind of assertion (inclusion, equivalence, non-inclusion, and non-equivalence); moreover, we use a convention on the role names of relationships, in order to distinguish beween intensional and extensional assertions. Table 1 reports such conventions.

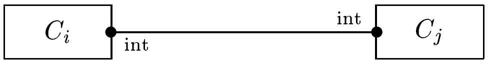
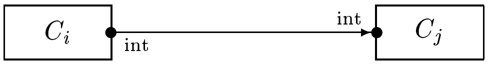
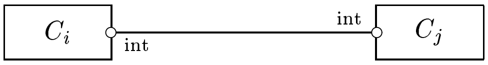
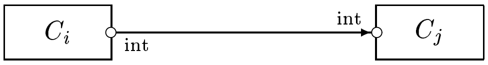
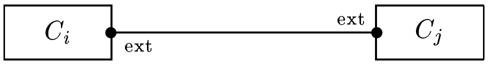
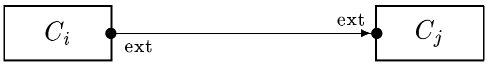
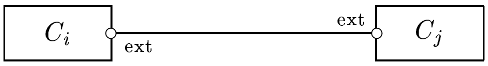
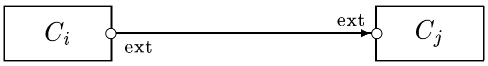
$C_i \equiv_{int} C_j$	
$C_i \sqsubseteq_{int} C_j$	
$C_i \not\equiv_{int} C_j$	
$C_i \not\sqsubseteq_{int} C_j$	
$C_i \equiv_{ext} C_j$	
$C_i \sqsubseteq_{ext} C_j$	
$C_i \not\equiv_{ext} C_j$	
$C_i \not\sqsubseteq_{ext} C_j$	

Table 1: Conventions used for representing intermodel assertions

Notice that the distinction between intensional and extensional assertions is realized by using special role names for relationships: specifically, the name **int** is used for both role names in a relationship representing an intensional intermodel assertion, while the name **ext** is used for both role names in a relationship representing an extensional intermodel assertion.

Since in this way it is only possible to express assertions between entities without imposing additional constraints, we provide the possibility of attaching constraints to relationships, by expressing such constraints in the language of Description Logics. In particular, we adopt the syntax of the system FaCT [Hor98], which conforms to the general standard of Description Logic based Knowledge representation systems [PS93]. Such a syntax allows for a textual specification of Description Logics assertions; therefore, it is possible to attach such assertions to relationships in GroundWorks by specifying them in one of the fields of the form which describes the relationship properties. In particular, the field *description* is used, since it can be displayed very easily (by simply double-clicking on the dot associated with the role). The symbol # is added to the role name: this feature is used both to display the presence of additional constraints to the user, and to facilitate the automatic translation of the schema in terms of a FaCT knowledge base (see Section 4).

3 The Conceptual Data Warehouse Model of the DW 'Customer Care'

In this section we describe the Conceptual Data Warehouse Model of the application 'Customer Care'. We first comment on the generation of the Enterprise Model, describing the kinds of conflicts arising when source models are integrated, then we present in detail the intermodel assertions we have defined in the Conceptual Data Warehouse Model.

3.1 Solving conflicts between sources

During the model integration phase a significant number of conflicts arises: this is due to the fact that different models are designed without considering information overlapping, often using different design conventions.

In the case of the application under consideration, the starting point is a set of applications which have already been integrated (they are called *BDA sources*); this set of sources are merged with two proprietary databases (called *legacy* systems). Therefore, conflicts may arise only between these two kinds of models.

Starting from the analysis reported in [LNRS98], two conflicts have been identified, which are due to the different modeling of the same concepts in two different sources, the BDA source DB09 and the legacy system LEGACY1.

The restructured source models of the BDA source and the legacy system are reported in Figures 1 and 2 respectively.

3.2 Enterprise Model

Starting from the project specifications for the Data Warehouse 'Customer Care', we have analyzed the concepts relevant for the Data Warehouse application, and we have realized that they can be modeled through the conceptual models of the BDA sources. In other words, the concepts expressed in the legacy systems are entirely included in the concepts expressed in BDA source models. Notice that this is not always the case. In fact, the same source contains a few entities that are not modeled in BDA sources, and which should be added to the integrated Enterprise Model. However, such concepts are not needed for the 'Customer Care' Data Warehouse.

Therefore, we have created the Enterprise Model by merging the BDA Source Models.

3.3 Intermodel assertions

The intermodel assertions which can be expressed in the logical framework that we have adopted for representing the Conceptual Data Warehouse Model [CDGL$^+$98b, CDGL$^+$98c] are distinguished in two different classes:

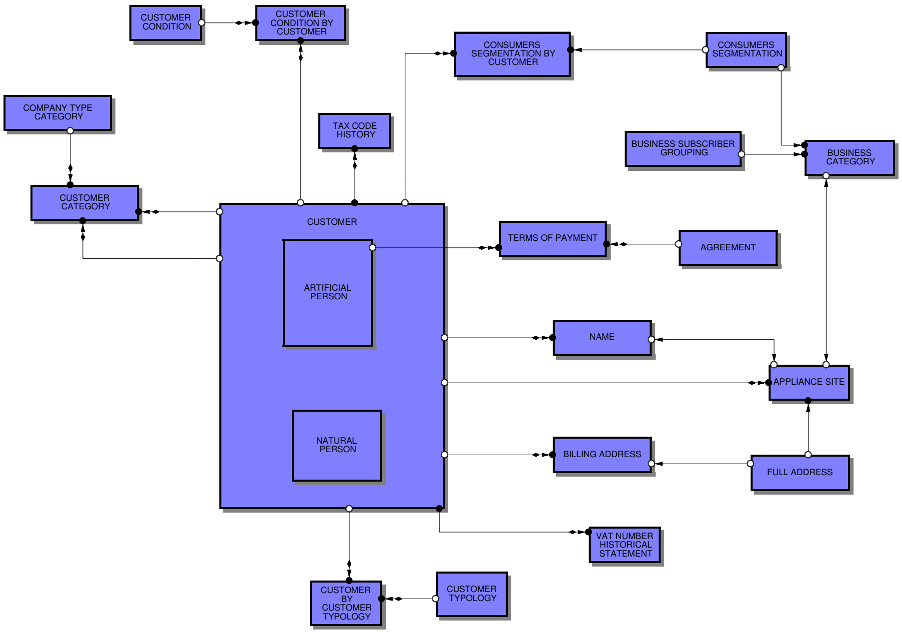


Figure 1: Restructured conceptual model of DB09

- assertions between a BDA source and the Enterprise model, and between two BDA Source Models;
- assertions involving models of legacy systems.

The distinction is important, since the first kind of assertions are conceptually simple, while assertions involving models of legacy systems can be very complex.

Let us briefly recall the meaning of intermodel assertions. *Extensional* assertions impose constraints not only at the conceptual level, but also at the concept instance level. Hence, an extensional equivalence assertion implies that the concepts connected by the assertion have the same informational content. Therefore, when extracting data from the sources, there is no difference in choosing one of the two sources. On the contrary, an assertion of the form $C_i \sqsubseteq_{ext} C_j$ states that the concept C_i has less informational content than the concept C_j; hence the source whose model contains the concept C_j will be chosen.

Instead, intensional assertions are used to indicate relationships which hold at the conceptual level, but do not necessarily hold at the instance level. This implies that, in order to have complete information, it is anyway necessary to take into account both souces, even when the concept in one source is less general than the concept expressed in the other source.

3.4 Assertions between BDA sources and between Enterprise Model and BDA sources

Intermodel assertions between BDA sources and between Enterprise Model and BDA sources are assertions involving single concepts. In the case of assertions between BDA sources, the concepts involved are called *proprietary entity* and *border entity*, and the kind of assertion is typically an extensional inclusion assertion, stating that the border entity is included in the proprietary entity.

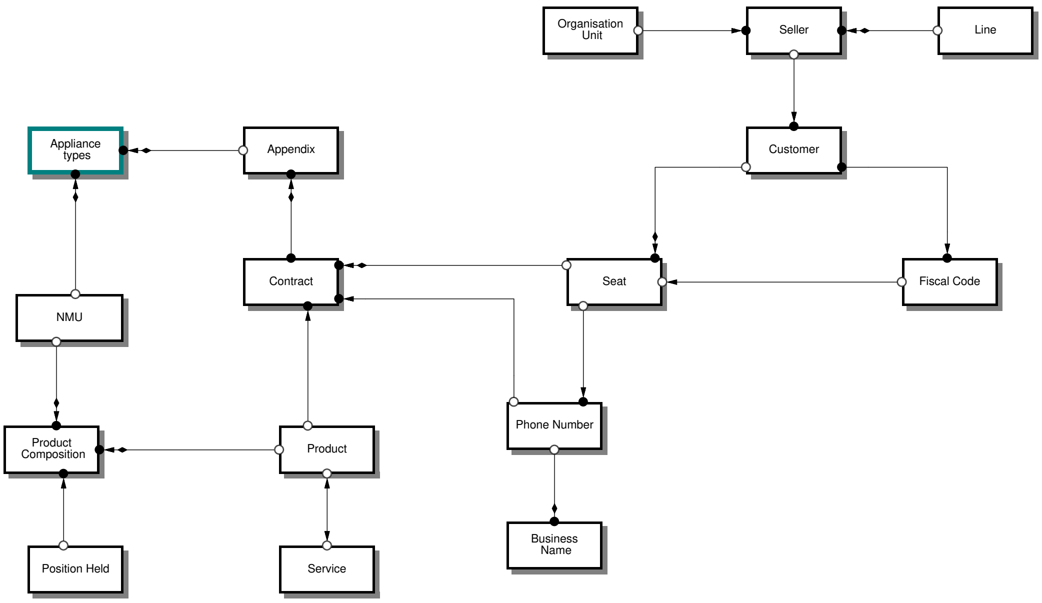


Figure 2: Restructured conceptual model of LEGACY1

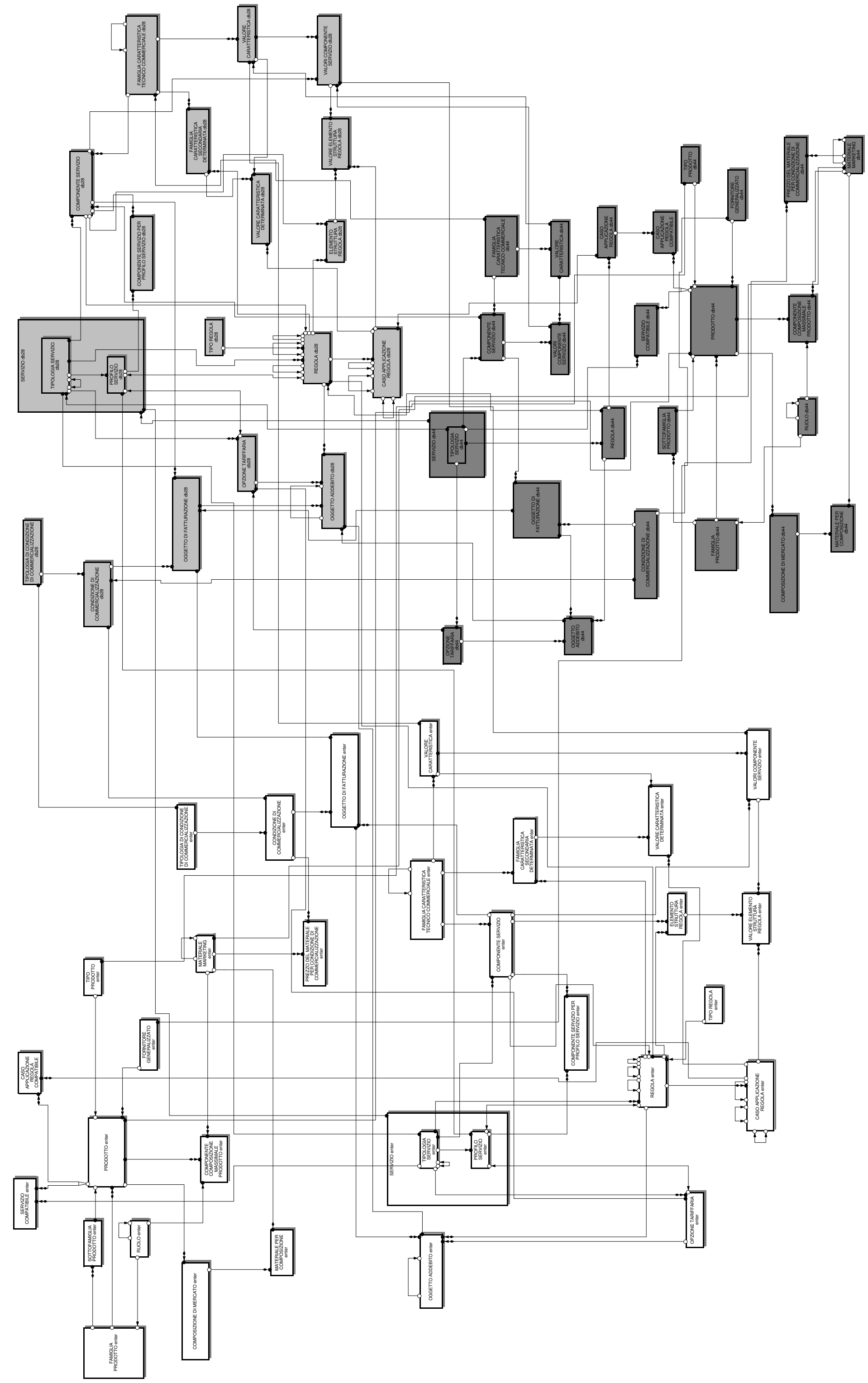


Figure 3: Portion of the CDWM involving the BDA sources DB28 and DB44

Instead, an assertion between a BDA source and the Enterprise Model establishes an (extensional) equivalence between two concepts, since the Enterprise Model is created starting from the BDA source models. Therefore an assertion between a BDA source and the Enterprise Model is an extensional equivalence assertion.

Below we report some examples of these two kinds of intermodel assertions. They represent some of the intermodel assertions that are graphically expressed in Figure 3, which describes the part of the Conceptual Data Warehouse Model of the Data Warehouse 'Customer Care' concerning the BDA sources DB28 and DB44 (coloured entities), and the corresponding part of the Enterprise Model (white entities). Source concepts are subscribed with the name of the source they belong, while the concepts belonging to the Enterprise Model are not subscribed.

$$\begin{array}{rcl} ServiceType_{DB44} & \sqsubseteq_{ext} & ServiceType_{DB28} \\ ServiceType_{DB28} & \equiv_{ext} & ServiceType \end{array}$$

$$\begin{array}{rcl} Service_{DB44} & \sqsubseteq_{ext} & Service_{DB28} \\ Service_{DB28} & \equiv_{ext} & Service \end{array}$$

3.5 Assertions involving models of legacy systems

The concepts expressed in the models of legacy systems are generally related in a straightforward way to the concepts in BDA source models. However, sometimes such a connection is very difficult to establish, and it is necessary to express complex intermodel assertions: roughly, this is due to the fact that some concepts expressed in the models of legacy systems are modeled in BDA sources through several concepts.

All intermodel assertions of this kind are grouped in three tables: Table 2 reports the assertions concerning the LEGACY1 legacy system, Table 3 reports the assertions concerning the LEGACY2 legacy system, and Table 4 contains the assertions between concepts from the LEGACY2 model and concepts from the LEGACY1 model.

Figure 4 shows the part of the Conceptual Data Warehouse Model interested by such assertions.

Assertions concerning LEGACY1

Assertion 1 concerns the entity *Appendix*. Roughly speaking, an appendix contains some additional information about a contract, and each contract is associated with a set of appendices. The first appendix of a contract, called *contract appendix*, is mandatory: it indicates the payment procedure and the contract amount. If the contract changes, other kinds of appendix may be present.

Notably, in the BDA source DB10 the same concept is modeled in a completely different way. First, there are three possible kinds of contract (*standard* contract, *ad-hoc* contract, and *managing* contract). Moreover, contracts are further distinguished between *primary* and *secondary* contracts. Secondary contracts are used to express changes to a primary contract. Hence, it is possible to deduce that an instance of $Appendix_{legacy1}$ is a secondary contract (which in turn may be either an ad-hoc or a standard contract) which modifies a primary (either ad-hoc or standard) contract.

The concept belonging to the LEGACY1 model which is involved in assertion 3 represents the catalog of all materials used in the LEGACY1 application: the marketing material represents the set of materials known by the marketing division. Therefore, the entity $NMU_{legacy1}$ describes a set of objects greater than the set of objects corresponding to the entity $MarketingMaterial_{DB44}$, which is the reason why 3 is an inclusion assertion.

The entities $Product_{legacy1}$ and $Contract_{legacy1}$, which appear in assertions 4 and 6, respectively represent the set of products and contracts which are handled by the LEGACY1 application, hence they are just a subset of the set of products and contracts managed by the enterprise. On the contrary, the services managed by the LEGACY1 application (and modeled by the entity $Service_{legacy1}$) are all the services which can be associated with a product, hence they correspond to all the services provided by the company (see assertion 5).

As for assertion 7, in the LEGACY1 model the concept *Legal Name* corresponds to the information associated with the phone number in the phone directory. Thus, this concept corresponds to the entity *Name* in the model of the BDA source DB09.

Assertions 8, 9, and 10 are rather trivial.

Finally, the concept *Appliance Types* in the LEGACY1 model is measured by the products bought by the customer, hence it corresponds to the concept expressed by the entity $Product\,By\,Customer_{DB29}$.

$$
\begin{array}{rcll}
Appendix_{legacy1} & \equiv_{int} & (Ad\,Hoc\,Contract_{DB10} \sqcap & (1)\\
 & & \forall varies_{Ad\,Hoc\,Contract_{DB10}}.Standard\,Contract_{DB10}) \sqcup & \\
 & & (Standard\,Contract_{DB10} \sqcap & \\
 & & \forall refers\,to_{Standard\,Contract_{DB10}}.Standard\,Contract_{DB10}) \sqcup & \\
 & & (Ad\,Hoc\,Contract_{DB10} \sqcap & \\
 & & \forall modifies_{Ad\,Hoc\,Contract_{DB10}}.Ad\,Hoc\,Contract_{DB10}) & \\
Fiscal\,Code_{legacy1} & \equiv_{ext} & Tax\,Code\,History_{DB09} \sqcup VAT\ Nr\ History_{DB09} & (2)\\
Marketing\,Material_{DB44} & \sqsubseteq_{ext} & NMU_{legacy1} & (3)\\
Product_{legacy1} & \sqsubseteq_{ext} & Product_{DB44} & (4)\\
Service_{legacy1} & \equiv_{ext} & Service_{DB28} & (5)\\
Contract_{legacy1} & \equiv_{int} & Contract_{DB10} & (6)\\
Legal\,Name_{legacy1} & \equiv_{ext} & Name_{DB09} & (7)\\
Site_{legacy1} & \equiv_{ext} & Appliance\ Site_{DB09} & (8)\\
Organization\,Unit_{legacy1} & \equiv_{ext} & Organization\,Unit & (9)\\
Customer_{legacy1} & \sqsubseteq_{ext} & Customer_{DB09} & (10)\\
Position\,Held_{legacy1} & \equiv_{ext} & Role_{DB44} & (11)\\
Product\,Composition_{legacy1} & \equiv_{int} & Maximal\,Composition\,Product_{DB09} & (12)\\
Appliance\,Types_{legacy1} & \equiv_{ext} & Product\,By\,Customer_{DB09} & (13)
\end{array}
$$

Table 2: Assertions involving concepts in the LEGACY1 model

Assertions concerning LEGACY2

The assertions between concepts in the legacy system LEGACY2 and the Enterprise Model and BDA source models are rather simple, and we do not further comment on them. Such assertions are reported in Table 3. Instead, the assertions reported in Table 4 specify relationships between entities in the models of LEGACY1 and LEGACY2.

As for assertions 25 and 26, they show the fact that sometimes it is necessary to express assertions between concepts in models of two different legacy systems. In this case, there are homonym entities (e.g., *Product*) which actually represent different concepts in the two models.

Finally, the concept *Appliance Types* is equivalent for both systems, however it is not defined over the same domain. Therefore it is necessary to introduce assertion 27.

4 Reasoning subsystem

In this section we present the subsystem we have developed for reasoning about the conceptual representation of the Data Warehouse. Such a subsystem is constituted by a module which translates the Conceptual Data Warehouse Model from the format used in GroundWorks, and described in Section 2, to the language of Description Logics, and by the knowledge representation system FaCT [Hor97, Hor98], a prototype system for expressive Description Logics developed at the University of Manchester. The

$$
\begin{array}{rcll}
Customer_{legacy2} & \sqsubseteq_{ext} & Customer_{DB09} & (14) \\
Natural\ Person_{legacy2} & \sqsubseteq_{ext} & Natural\ Person_{DB09} & (15) \\
Seat_{legacy2} & \equiv_{ext} & Appliance\ Site_{DB09} & (16) \\
Organization\ Unit_{legacy2} & \equiv_{ext} & Organization\ Unit & (17) \\
Business\ Category_{legacy2} & \equiv_{ext} & Business\ Category_{DB09} & (18) \\
Consumer\ Segmentation_{legacy2} & \equiv_{ext} & Consumer\ Segmentation_{DB09} & (19) \\
Billing\ Address_{legacy2} & \equiv_{ext} & Billing\ Address_{DB09} & (20) \\
Work\ Order_{legacy2} & \equiv_{ext} & Work\ Order_{DB29} & (21) \\
Product_{legacy2} & \sqsubseteq_{ext} & Product\ By\ Customer_{DB29} & (22) \\
Service_{legacy2} & \sqsubseteq_{ext} & Service\ By\ Customer_{DB29} & (23) \\
Appliance\ Types_{legacy2} & \sqsubseteq_{ext} & Appliance\ Types_{DB29} & (24)
\end{array}
$$

Table 3: Assertions involving concepts in the LEGACY2 model

$$
\begin{array}{rcll}
Product_{legacy2} & \not\sqsubseteq_{ext} & Product_{legacy1} & (25) \\
Service_{legacy2} & \not\sqsubseteq_{ext} & Service_{legacy1} & (26) \\
Appliance\ Types_{legacy2} & \sqsubseteq_{int} & Appliance\ Types_{legacy1} & (27)
\end{array}
$$

Table 4: Intermodel assertions between the LEGACY1 model and the LEGACY2 model

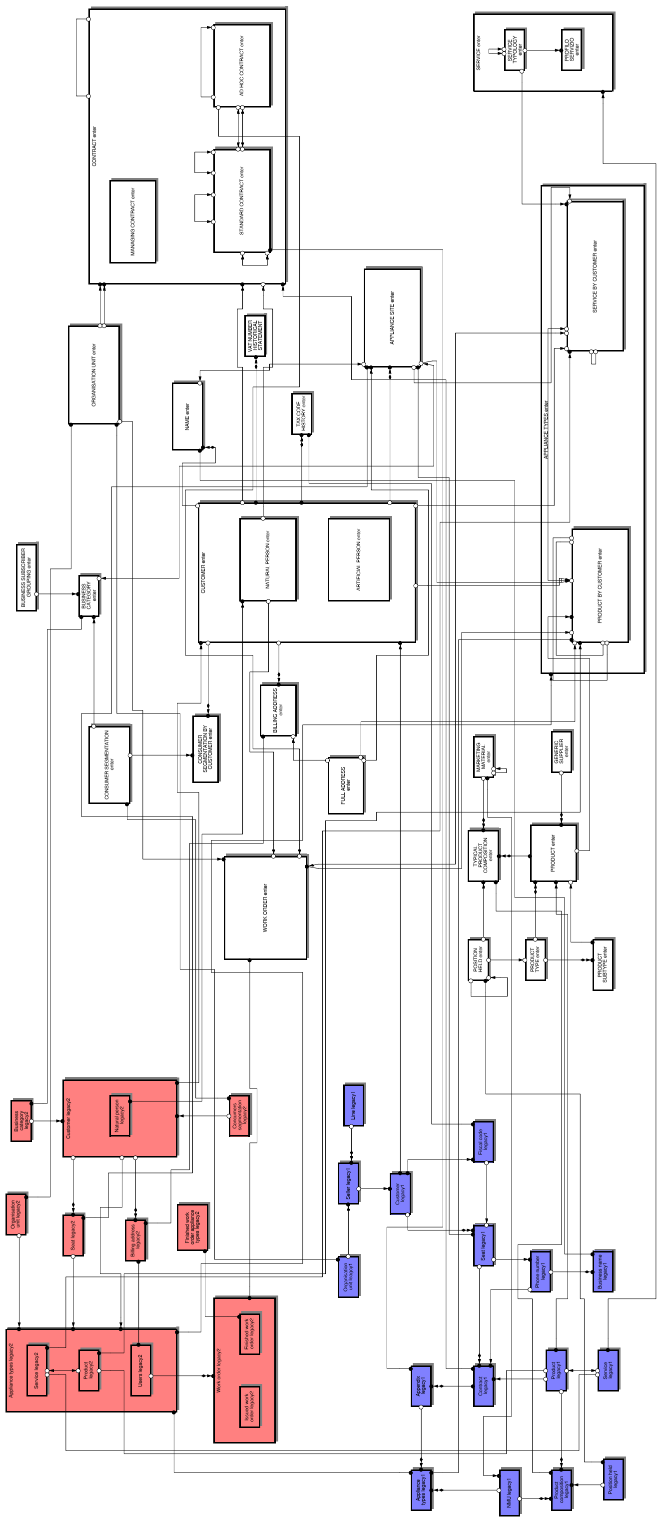


Figure 4: Portion of the CDWM concerning the legacy systems

idea is to use the reasoning abilities provided by FaCT in order to discover properties of the conceptual representation of the Data Warehouse.

In the rest of the section we first present the system FaCT, then we provide a description of the translator GroundWorks–FaCT, and finally we present some examples showing the use of the reasoning abilities of the system in the design of the Conceptual Data Warehouse Model.

4.1 The inference engine: FaCT

FaCT is a prototype Description Logic knowledge representation system which uses an optimised tableaux subsumption algorithm. In particular, FaCT supports transitively closed roles, a role/description hierarchy, and general concept inclusion axioms (implications of the form $C \sqsubseteq D$ where C and D are arbitrary concept descriptions).

FaCT is mainly intended as a tool for conceptual model design and thus provides only concept based reasoning services. FaCT provides `*TOP*` ($\top$) and `*BOTTOM*` ($\bot$) as the only predefined concepts. Moreover, it provides the following features:

- language constructs which allow for concept and role expressions of the $\mathcal{ALC}$ Description Logic [DLNS96]: in particular, FaCT allows for atomic attribute and role definitions, concept disjunction, and cyclic concept definitions;
- commands to make inferences;
- commands to perform queries about the knowledge base.

However, FaCT capabilities are not sufficient to provide a direct implementation of our approach. In fact, in order to define intermodel assertions of the form $\not\equiv_{int}$, $\not\sqsubseteq_{int}$, $\not\equiv_{ext}$, $\not\sqsubseteq_{ext}$, it is necessary to use expressions involving concept instances, thus making it impossible to find a suitable translation using FaCT syntax. Hence, our reasoning subsystem is not able to handle intermodel assertions of the above form: this is not a real limitation, since (as shown in Section 3) such assertions are hardly used in the specification of the Conceptual Data Warehouse Model.

4.2 GroundWorks-FaCT translator

In order to use FaCT for reasoning about conceptual models, we implemented a program for the automatic translation of Entity-Relationship models (expressed in the language of GroundWorks) to a set of FaCT definitions.

Specifically, such a program receives as input a set of GroundWorks exportation files (which have the extension `.var`) and generates as output a file containing the specification of a FaCT knowledge base which is equivalent to the initial Entity-Relationship models.

4.3 Using the reasoning subsystem: an example

We will now give an example of usage of the reasoning subsystem to support Data Warehouse conceptual design. We refer to the model shown in Figure 5, which is a portion of the Conceptual Data Warehouse Model of the 'Customer Care' application: it is constitued by:

- the Enterprise Model;
- some entities belonging to the model of a legacy system;
- some entities belonging to the model of an integrated BDA source.

Due to readibility problems, the Enterprise Model is only partially shown.

Let us now comment on the intermodel assertions involving the entities belonging to the Source Models and the Enterprise Model, which are reported in Table 5.

1. Assertions 30, 31, 32, 33, 34, 35, 36 are simple inclusion relations between entites belonging to Source Models and entities belonging to the Enterprise Model.

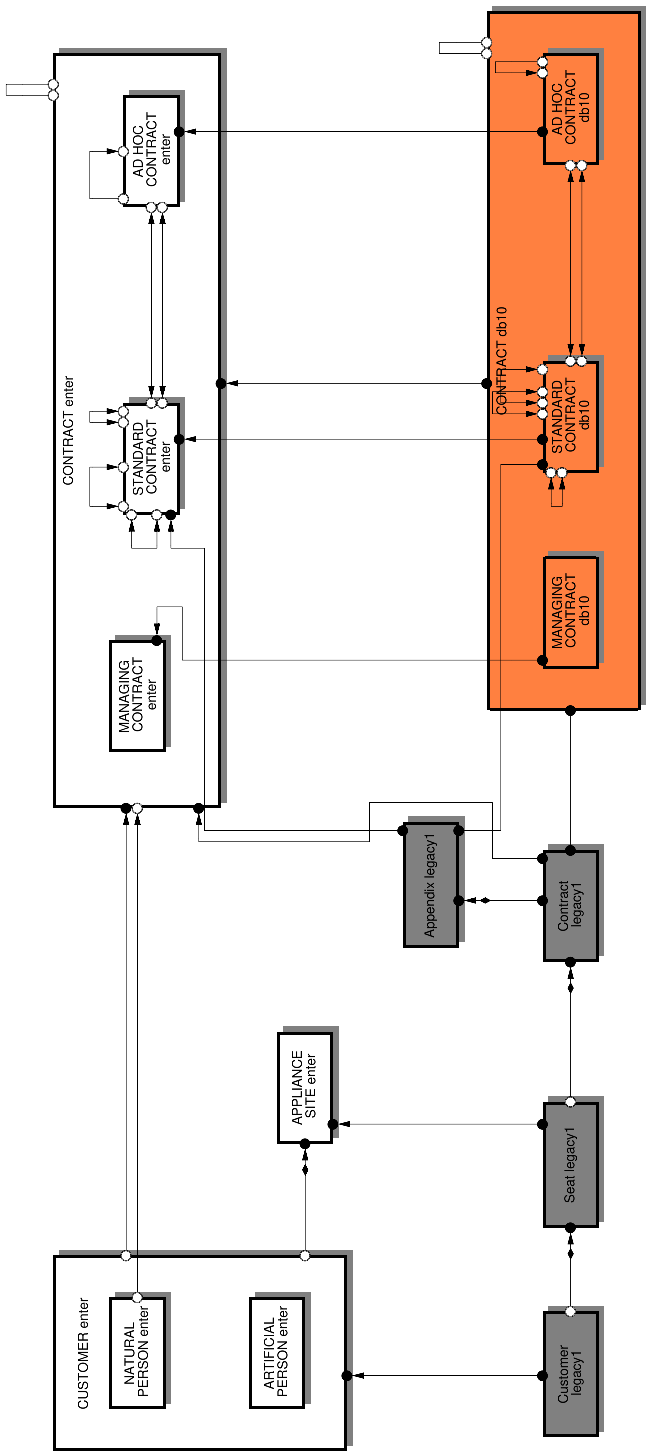


Figure 5: Portion of the CDWM of the 'Customer Care' application

$$\begin{aligned}
&Appendix_{legacy1} \sqsubseteq_{ext} \\
&\quad (Ad\,Hoc\,Contract_{enter} \sqcap \\
&\quad \forall varies_{Ad\,Hoc\,Contract_{enter}}.Standard\,Contract_{enter}) \sqcup \\
&\quad (Standard\,Contract_{enter} \sqcap \\
&\quad \forall refers_to_{Standard\,Contract_{enter}}.Standard\,Contract_{enter}) \sqcup \\
&\quad (Ad\,Hoc\,Contract_{enter} \sqcap \\
&\quad \forall modifies_{Ad\,Hoc\,Contract_{enter}}.Ad\,Hoc\,Contract_{enter})
\end{aligned} \tag{28}$$

$$\begin{aligned}
&Appendix_{legacy1} \equiv_{int} \\
&\quad (Ad\,Hoc\,Contract_{db10} \sqcap \\
&\quad \forall varies_{Ad\,Hoc\,Contract_{db10}}.Standard\,Contract_{db10}) \sqcup \\
&\quad (Standard\,Contract_{db10} \sqcap \\
&\quad \forall refers_to_{Standard\,Contract_{db10}}.Standard\,Contract_{db10}) \sqcup \\
&\quad (Ad\,Hoc\,Contract_{db10} \sqcap \\
&\quad \forall modifies_{Ad\,Hoc\,Contract_{db10}}.Ad\,Hoc\,Contract_{db10})
\end{aligned} \tag{29}$$

$$Customer_{legacy1} \sqsubseteq_{ext} Customer_{enter} \tag{30}$$
$$Seat_{legacy1} \sqsubseteq_{ext} Appliance\,Site_{enter} \tag{31}$$
$$Contract_{legacy1} \sqsubseteq_{ext} Contract_{enter} \tag{32}$$
$$Contract_{db10} \sqsubseteq_{ext} Contract_{enter} \tag{33}$$
$$Standard\,Contract_{db10} \sqsubseteq_{ext} Standard\,Contract_{enter} \tag{34}$$
$$Ad\,Hoc\,Contract_{db10} \sqsubseteq_{ext} Ad\,Hoc\,Contract_{enter} \tag{35}$$
$$Managing\,Contract_{db10} \sqsubseteq_{ext} Managing\,Contract_{enter} \tag{36}$$
$$Contract_{legacy1} \equiv_{int} Contract_{db10} \tag{37}$$

Table 5: Intermodel assertions between source models and Enterprise Model

2. Assertion 28 specifies that the entity $Appendix_{legacy1}$ can be:
   - a $Standard\,Contract_{enter}$ modifying another $Standard\,Contract_{enter}$;
   - a $Ad\,Hoc\,Contract_{enter}$ modifying another $Ad\,Hoc\,Contract_{enter}$;
   - a $Ad\,Hoc\,Contract_{enter}$ modifying a $Standard\,Contract_{enter}$.
3. Assertion 29 specifies that the entity $Appendix_{legacy1}$ is conceptually equivalent to:
   - a $Standard\,Contract_{db10}$ which modifies another $Standard\,Contract_{db10}$;
   - a $Ad\,Hoc\,Contract_{db10}$ which modifies another $Ad\,Hoc\,Contract_{db10}$;
   - a $Ad\,Hoc\,Contract_{db10}$ which modifies a $Standard\,Contract_{db10}$.

   However, these entities are defined over domains that are not coincident, but only have a common intersection.
4. Assertion 37 specifies an equivalence relation between the entities $Contract_{legacy1}$ and $Contract_{db10}$, but only at the conceptual level (in a way similar to what explained in the previous point).

Reasoning above this portion of Conceptual Data Warehouse Model makes it possible to point out some interesting properties.

In the first scenario we considered, a property that has been derived by the reasoning subsystem is the following:

$$Appendix_{legacy1} \quad \sqsubseteq_{ext} \quad StandardContract_{enter} \sqcup AdHocContract_{enter}$$

Informally, this property states that the instances belonging to the entity $Appendix_{legacy1}$ are a subset of the set formed by the instances belonging to the entities $StandardContract_{enter}$ and $AdHocContract_{enter}$. As an immediate consequence, it follows that is not possible to populate correctly the latter entities, because it is not decidable whether an instance from $Appendix_{legacy1}$ belongs to either $StandardContract_{enter}$ or $AdHocContract_{enter}$. This is necessary when we want to populate the Data Warehouse with information coming from the entity $StandardContract_{enter}$ and/or the entity $AdHocContract_{enter}$.

There are two possible solutions to this problem:

1. restructuring the Enterprise Model, which corresponds to merging the entities $StandardContract_{enter}$ and $AdHocContract_{enter}$, so that the inconsistency can be eliminated;

2. modifying the Source Model, e.g. introducing an attribute in the entity $Appendix_{legacy1}$, so that we can distinguish between the instances which belong to $StandardContract_{enter}$ and the ones belonging to $AdHocContract_{enter}$. Obviously, this is not an acceptable solution from the designer's viewpoint.

We will now consider a different scenario, in which we suppose that the model has been modified by eliminating assertion 28 and inserting the following expression:

$$Appendix_{legacy1} \sqsubseteq_{ext} StandardContract_{enter}$$

In this case, the instances belonging to the entity $Appendix_{legacy1}$ are a subset of the instances belonging to the entity $StandardContract_{enter}$.

Another property which can be discovered by making use of the reasoning subsystem is consistency of the following concept expression:

$$Appendix_{legacy1} \sqcap \neg(StandardContract_{db10} \sqcup AdHocContract_{db10})$$

which implies that the instances belonging to the entity $Appendix_{legacy1}$ are not a subset of the set formed by the instances belonging to the entities $StandardContract_{db10}$ and $AdHocContract_{db10}$ This assertion, together with the new one previously introduced, leads us to deduce that the legacy source is necessary in order to populate the Enterprise Model.

5 Conclusions

In this report we have described the tools developed to support the methodology for conceptual Data Warehouse modeling presented in [CDGL+97] and its application to a project of practical industrial relevance undertaken at Telecom Italia. The outcomes of this practical experience have been manyfold. In particular the following considerations are in order.

- The size and the nature of the example are rather significant and the design of the conceptual Data Warehouse model has been accomplished very successfully providing both the desired abstractions for the conceptual level of Data Warehouse design and a practical basis for the incremental development of Data Warehouse applications and traditional integration process.

- The tools are embedded within already exiting solutions and thus the burden for the designer to get acquainted with the new features of the system is limited to the new design notions arising in conceptual Data Warehouse modeling.

- The application of reasoning techniques provides very valuable insights for the designer; especially, if the new functionalities can be obtained within standard tools and methodologies currently in use.

- Because of the limitation of the current technology for reasoning in Description Logic systems the prototype for reasoning cannot deal with very large models (only with respect to some reasoning tasks).

The overall evaluation of the potential for industrial application of our methodology for conceptual DW modeling is therefore extremely positive. We intend to continue the work in this direction by providing a graphical interface to support queries on the conceptual model, in order to provide information about the sources and the data contained therein. Such information can be of great interest to drive the subsequent stages of the design, namely the logical design of the Data Warehouse materialized Data Store and the overall data extraction process [CDGL+98d].

Acknowledgements

We are sincerely grateful to Telecom Italia, and, in particular, Stefano Trisolini, Massimo Biondi, Valeria Stancati, and Daniela Segreto, for providing all the support needed for the experimentation on the 'Customer Care' project. In addition, we would like to thank G. Pinzi of Cayenne Software for providing us all the necessary support to use GroudWorks. Finally, we thank Riccardo Capitini for his work on the implementation of the tool and on the construction of the Conceptual Data Warehouse Model for the 'Customer Care' Data Warehouse.

References

[CDGL95] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Structured objects: Modeling and reasoning. In *Proceedings of the Fourth International Conference on Deductive and Object-Oriented Databases (DOOD-95)*, number 1013 in Lecture Notes in Computer Science, pages 229–246. Springer-Verlag, 1995.

[CDGL+97] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Source integration in data warehousing. Technical Report DWQ-UNIROMA-002, DWQ Consortium, October 1997.

[CDGL98a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS-98)*, pages 149–158, 1998.

[CDGL+98b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13, 1998.

[CDGL+98c] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proceedings of the Sixth International Conference on Cooperative Information Systems (CoopIS-98)*, pages 280–291, 1998.

[CDGL+98d] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Source and data integration methodology for dwq. Technical Report DWQ-UNIROMA-004, DWQ Consortium, September 1998.

[CL93] Tiziana Catarci and Maurizio Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.

[CLN94] Diego Calvanese, Maurizio Lenzerini, and Daniele Nardi. A unified framework for class based representation formalisms. In J. Doyle, E. Sandewall, and P. Torasso, editors, *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR-94)*, pages 109–120, Bonn, 1994. Morgan Kaufmann, Los Altos.

[DGL94] Giuseppe De Giacomo and Maurizio Lenzerini. Description logics with inverse roles, functional restrictions, and n-ary relations. In *Proceedings of the Fourth European Workshop on Logics in Artificial Intelligence (JELIA-94)*, volume 838 of *Lecture Notes In Artificial Intelligence*, pages 332–346. Springer-Verlag, 1994.

[DGL95] Giuseppe De Giacomo and Maurizio Lenzerini. What's in an aggregate: Foundations for description logics with tuples and sets. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 801–807, 1995.

[DLNS96] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. Reasoning in description logics. In Gerhard Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[Hor97] Ian Horrocks. Optimisation techniques for expressive description logics. Technical Report UMCS-97-2-1, University of Manchester, Department of Computer Science, 1997.

[Hor98] Ian 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)*, pages 636–647, 1998.

[LNRS98] Maurizio Lenzerini, Daniele Nardi, Riccardo Rosati, and Valeria Stancati. Analisi di metodi e tecniche per la progettazione di data warehouse (analysis of techniques and methods for data warehouse design). Technical Report DWQ-UNIROMA-004, DIS–Telecom Italia, 1998. In italian.

[PS93] P. F. Patel-Schneider and Bill Swartout. Working version (draft): Description logic specification from the KRSS effort, June 1993. Unpublished Manuscript.

Appendix

We include as appendix the Conceptual Data Warehouse Model of the 'Customer Care' application, reported in Figure 6.

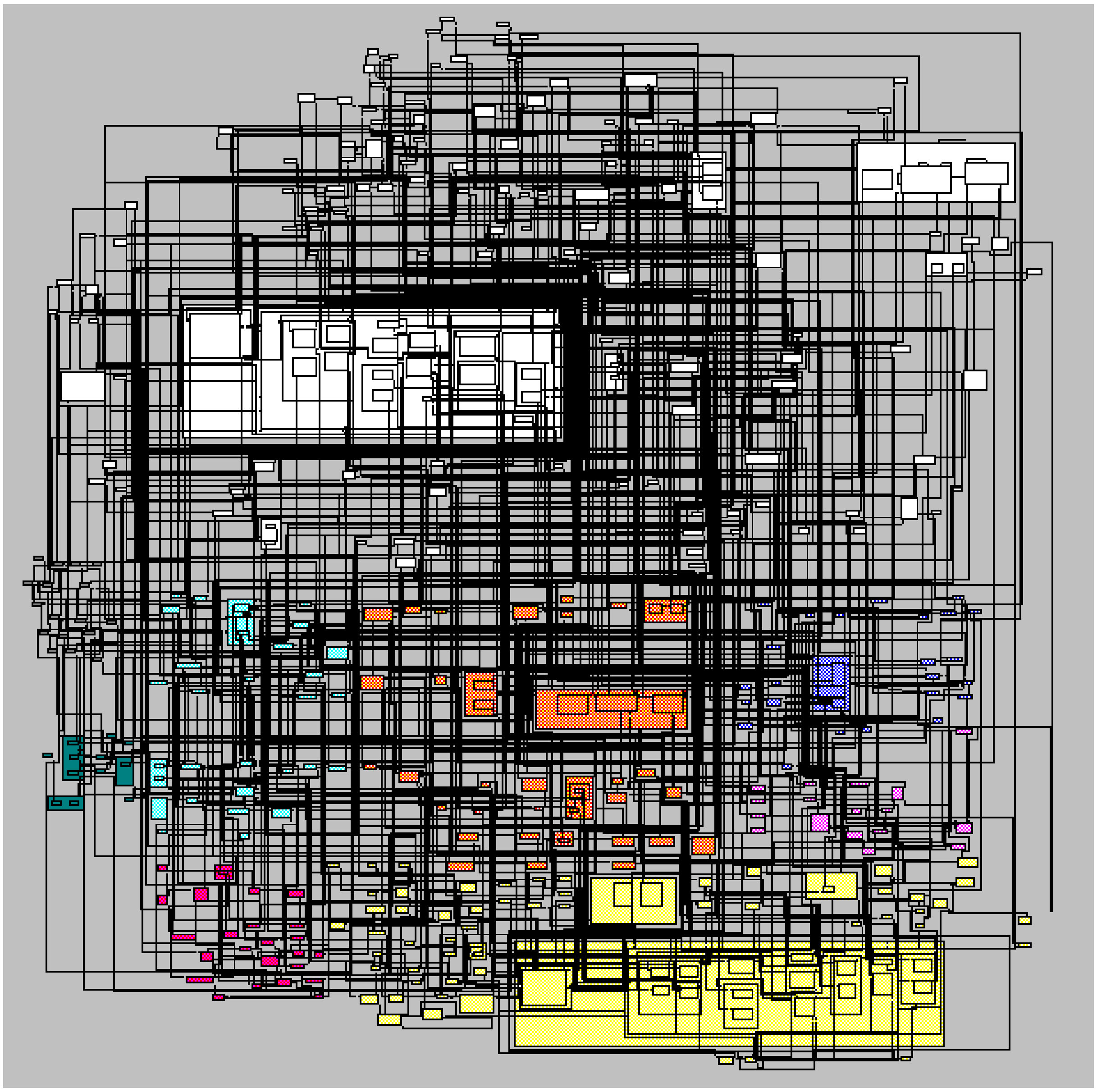


Figure 6: The Conceptual Data Warehouse Model of the 'Customer Care' application