Document Number:	D2.1
Project Name:	Infomix: Boosting Information Integration
Project Number:	IST-2001-33570
Title:	**Functional specification of the INFOMIX system**
Authors:	Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati
Workpackage:	WP2
Document Type:	Deliverable
Classification:	Confidential
Distribution:	Infomix Consortium
Status:	final
Document file:	
Version:	2.0
Date:	September 30, 2002
Number of pages:	18

Document Change Record		
Version	**Date**	**Reason for Change**
v.1.0	September 26, 2002	First draft
v.2.0	September 30, 2002	Final version

Abstract

In this document we define the functional specification of a system for data integration, which constitutes the basis for the development of the INFOMIX system. In particular, we first introduce the INFOMIX approach to information integration, by describing a formal framework and a declarative semantics for data integration systems. Such a formal specification is general enough to capture all the main current approaches to data integration, and is also able to deal with inconsistent and incomplete data sources. Then, based on this formal model of a data integration system, and on the preliminary structure of the system outlined in the INFOMIX proposal, we identify the general functionalities that the system should supply, thus providing a functional specification of the data integration system.

1 Introduction

Recent developments of information and communication technology enable accessing a large number of data sources, developed in different times, with different organizational principles and models, and supported by different hw/sw platforms. Since more and more users need to extract and analyze the information stored in highly heterogeneous sources, the necessity arises to provide them with new and more efficient methodologies enabling the achievement of their aims.

The problem of developing new techniques for heterogeneous sources integration and designing effective data integration systems has been addressed by several research projects in the last years. Most of the data integration systems described in the literature are based on a unified view of data, called *mediated or global schema*, and on two modules, called *wrapper* [26, 34] and *mediator* [33]. Wrappers are system components that present data at the sources in a suitable form adopted in the integration system, and handle the access to the data taking into account source capabilities to answer queries posed to the sources. Mediators are modules that establish how to collect, align, and merge data coming from the wrappers, according to the structure of the global schema.

A crucial aspect in the design and the realization of mediators is the specification of the mapping between the sources and the global schema. Two basic approaches have been proposed in the literature. The first approach, called *global-as-view* (or simply GAV), focuses on the elements of the global schema, and associates to each of them a view over the sources. On the contrary, in the second approach, called *local-as-view* (or simply LAV), the focus is on the sources, in the sense that a view over the global schema is associated to each of them. A comparison between the two approaches is reported in [31, 7]. Generally speaking, the LAV approach favors the extensibility of the system, whereas the GAV approach seems more natural for the designer, and is adequate when the set of sources rarely changes.

Examples of LAV systems are Information Manifold [21], Infomaster [13], and the systems presented in [28] and [9]. Information Manifold and the system described in [9] express the global schema in terms of Description Logics [5, 3], and adopt the language of conjunctive queries to specify both user queries and views in the mapping. The system described in [28] uses an XML global schema, and adopts XML-based query languages for both the mapping and the queries on the global schema. More powerful schema languages for expressing the global schema are reported in [13, 19, 10, 8]. In particular, [13, 19] discuss the case where various forms of relational integrity constraints are expressible in the global schema, including functional and inclusion dependencies, whereas [10, 8] consider a setting where the global schema is expressed in terms of Description Logics, which allow for the specification of various types of constraints.

Examples of GAV systems are TSIMMIS [16], Garlic [11], COIN [17], MOMIS [4], Squirrel [35], and IBIS [7]. These systems usually adopt simple languages for expressing both the global and the source schemas. IBIS is the only system we are aware of that takes into account integrity constraints in the global schema. We refer to [22] for a more detailed description of both LAV and GAV systems.

In this document we introduce the INFOMIX approach to information integration, describe a formal framework for data integration systems, discuss the system semantics, and provide a general specification of its main functionalities.

The principal goal of the INFOMIX project is to provide advanced techniques and innovative methodologies for information integration systems. In a nutshell, the project aims at developing a theory, comprising a comprehensive information model and information integration algorithms, and a prototype implementation of a knowledge-based system for advanced information integration, by using computational logic and integrating research results on data acquisition and transformation. Special attention will be devoted to the definition of declarative user-interaction mechanisms, and techniques for handling semi-structured data, and incomplete and inconsistent data sources.

The formal framework for data integration is defined in terms of a global schema, which provides the unified and centralized view of the data, a source schema, comprising the schemas of all sources involved in the integration application, and the mapping that specifies the relationship between the two. Generally speaking, the framework allows for the specification of constraints of general form on both the global schema and the sources, and the definition of complex forms of mappings between the global schema and the source schema. More specifically, the mapping is given in terms of a set of assertions where each assertion associates a view over the global schema to a view over the sources, as proposed in [24]. Such an approach captures both LAV and GAV mappings, and allows also for the specification of more complex dependencies between elements of the global schema and elements of the sources.

With regard to the semantics, we deeply concentrate on the semantics of the mapping, and analyze several assumptions that can be adopted on mapping assertions, in order to specify how to interpret data that can be retrieved at the sources with respect to data that satisfy the corresponding portion of the global schema. With this respect, we first consider classical *exact*, *sound*, or *complete* assertions, that correspond to the different situations in which data in the answer to a view over the global schema are exactly the data in the answer to the corresponding view over the sources, or are a superset or a subset of such data. Then, we address the more general case in which data retrieved at the sources do not respect integrity constraints expressed over the global schema, and cannot be reconciled in such a way that both integrity constraints and mapping assertions are satisfied. Classical assumptions on mapping assertions do not allow us to properly handle such inconsistencies, and generally they bring about a situation where no database instance exists for the global schema. In this respect, we propose a more general approach in which the classical assumptions on mapping assertions can be suitably relaxed.

It is worth noticing that the semantic problem that arises in this context is similar to the one underlying the notion of *database repair* introduced by several works in the area of *inconsistent databases*. However, many of this studies basically apply to a single database setting [6, 1, 2], and the proposed techniques can be employed in a data integration setting only by assuming an "exact" interpretation of mapping assertions [27, 12, 18]. We refer to [22] for a more detailed analysis of these studies.

Finally, based on our formal model of a data integration system, and on the preliminary structure of the system outlined in the INFOMIX proposal, we have identified the general functionalities that the system should provide, thus providing a functional specification of the data integration system. In particular, we have divided the system features into four levels of capabilities:

- **Final User Level**, which comprises functionalities that allow users both to pose their queries to the system and to suitably access the results computed by the system.
- **Information Service Level**, which comprises functionalities that allow for the modeling of the global schema, the source schema, and the mapping, and for the reformulation of queries expressed over the global schema in terms of queries on the sources.
- **Internal Integration Level**, which comprises functionalities to optimize and execute the query plan computed by the query reformulation process.
- **Data Acquisition and Transformation Level**, which comprises functionalities to properly access the sources, retrieve data from them, and suitably transform the acquired data into the internal homogeneous data format adopted in the system.

It is worthwhile to stress that the formal framework for data integration, the system semantics and the classification of system functionalities provided in this document, are of general purpose, and do not have to be considered specific for the INFOMIX system: however, the aim of INFOMIX is to provide an implementation of a significant subset of such functionalities.

The present document is organized as follows. In Section 2, we illustrate a formal framework for data integration and describe the syntax (2.1) and the semantics (2.2) of a data integration system. Section 3 describes the preliminary structure of the INFOMIX system. Based on such a structure, in Section 4 we present the general functionalities of a data integration system. Finally, Section 5 concludes the paper.

2 Formal framework for data integration

In this section we define a logical framework for data integration. The main components of a data integration system are the sources, the global schema and the mapping between the two. We first present the syntax, and then the semantics of a data integration system.

2.1 Syntax

We consider to have a fixed infinite alphabet Γ of constants representing real world objects, and assume that the structures constituting the databases involved in our framework are defined over the fixed interpretation domain Γ, i.e., Γ represents all the possible elements in a database instance.

Formally, a data integration system $\mathcal{I}$ is a triple $\langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, where:

- $\mathcal{G}$ is the *global schema*, expressed in the global language $\mathcal{L}_{\mathcal{G}}$ over the alphabet $\mathcal{A}^{\mathcal{G}}$. The alphabet comprises a symbol for each element of $\mathcal{G}$ (i.e., relation if $\mathcal{G}$ is relational, class if $\mathcal{G}$ is object-oriented, etc.). The language $\mathcal{L}_{\mathcal{G}}$ determines the expressiveness allowed for specifying the global schema, i.e., the set of constraints that can be defined over it;

- $\mathcal{S}$ is the *source schema*, composed by the schemas of the various sources that are part of the data integration system. $\mathcal{S}$ is modeled in the source language $\mathcal{L}_{\mathcal{S}}$ over the alphabet $\mathcal{A}^{\mathcal{S}}$. We request that, $\mathcal{A}^{\mathcal{S}}$ is disjoint from the alphabet of the global schema $\mathcal{A}^{\mathcal{G}}$. As in the previous case, the alphabet comprises a symbol for each element of the sources, and $\mathcal{L}_{\mathcal{S}}$ determines the expressiveness allowed for specifying the source schema;

- $\mathcal{M}$ is the *mapping* between $\mathcal{G}$ and $\mathcal{S}$, i.e., the specification of the relationship between the sources and the global schema. It is constituted by a set of *assertions* of the form

$$\begin{array}{c} q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}, \\ q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}} \end{array}$$

  where $q_{\mathcal{S}}$ and $q_{\mathcal{G}}$ are two queries, respectively over the source schema $\mathcal{S}$, and over the global schema $\mathcal{G}$. Queries $q_{\mathcal{S}}$ are expressed in a query language $\mathcal{L}_{\mathcal{M},\mathcal{S}}$ over the alphabet $\mathcal{A}_{\mathcal{S}}$, and queries $q_{\mathcal{G}}$ are expressed in a query language $\mathcal{L}_{\mathcal{M},\mathcal{G}}$ over the alphabet $\mathcal{A}_{\mathcal{G}}$. Intuitively, an assertion $q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}$ specifies that the concept represented by the query $q_{\mathcal{S}}$ over the sources is put in correspondence with the concept in the global schema represented by the query $q_{\mathcal{G}}$ (similarly for an assertion of type $q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}}$). The exact meaning of such a correspondence will be described in the next subsection.

Thus, from the syntactic viewpoint, the specification of an integration system depends on the following parameters:

- The form of the global schema, i.e., the formalism used for expressing data and the global relationships between data.

- The form of the source schema, i.e., the formalism used for expressing data at the sources. Moreover, we assume that the data at the sources satisfy all constraints specified on $\mathcal{S}$, thus, in the following we do not consider anymore these constraints.

- The form of the mapping. As discussed in detail in [22], in the data integration literature two possible forms for the mapping are studied, called respectively *global-as-view* (GAV) and *local-as-view* (LAV). The GAV approach requires that the global schema is defined in terms of the data sources: more precisely, every element of the global schema is expressed as a view over the sources, so that its meaning is specified in terms of the data residing at the sources. With respect to the mapping syntax above defined, the GAV approach corresponds to restricting the queries $q_{\mathcal{G}}$ to unary queries, i.e., queries containing a single element of the global schema. In the LAV approach, the meaning of the sources is specified in terms of the elements of the global schema: more exactly, the mapping $\mathcal{M}$ between the sources and the global schema is provided in terms of a set of views over the global schema, one for each source element. With respect to the mapping syntax above defined, the LAV approach corresponds to restricting the queries $q_{\mathcal{S}}$ to unary queries, i.e., queries containing a single element of the source schema. Therefore, the above definition of mapping of our framework corresponds to a generalized form that comprises LAV and GAV as special cases.

Finally, we consider *queries* posed to a data integration system and define their syntax. Each such query is a formula that is intended to provide the specification of which data to extract from the integration system, i.e., q is intended to extract a set of elements of Γ. Each query is issued over the global schema $\mathcal{G}$, and is expressed in a specific query language, denoted by $\mathcal{L}_{\mathcal{Q}}$, over the global alphabet $\mathcal{A}^{\mathcal{G}}$.

2.2 Semantics

Intuitively, to specify the semantics of a data integration system, we have to start with a set of data at the sources, and, given such data at the sources, we have to specify which are the data that satisfy the global schema. Thus, in order to assign the semantics to a data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$, we start by considering a *source database* for $\mathcal{I}$, i.e., a database $\mathcal{D}$ for the source schema $\mathcal{S}$. Given a query q over $\mathcal{S}$ and a source database $\mathcal{D}$, we denote as $q^{\mathcal{D}}$ the set of objects in the answer to q over $\mathcal{D}$, i.e. the set of objects that satisfy q over $\mathcal{D}$.

Based on $\mathcal{D}$, we now specify which is the information content of the global schema $\mathcal{G}$. We call any database $\mathcal{B}$ for $\mathcal{G}$ a *global database* for $\mathcal{I}$. Furthermore, given a query q over $\mathcal{G}$ and a global database $\mathcal{B}$, we denote as $q^{\mathcal{B}}$ the set of objects in the answer to q over $\mathcal{B}$, i.e. the set of objects that satisfy q over $\mathcal{B}$.

A global database $\mathcal{B}$ for $\mathcal{I}$ is said to be *legal* with respect to $\mathcal{D}$ if:

1. $\mathcal{B}$ is coherent with $\mathcal{G}$;

2. $\mathcal{B}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{D}$, namely the objects in $\mathcal{B}$ satisfy the relationships between the global and the source elements defined by the mapping. More precisely, we say that $\mathcal{B}$ *satisfies* $\mathcal{M}$ *with respect to* $\mathcal{D}$ if:

   (a) for each assertion in $\mathcal{M}$ of the form $q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}$, each object in $q_{\mathcal{S}}^{\mathcal{D}}$ is also an element of $q_{\mathcal{G}}^{\mathcal{B}}$, i.e., $q_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}}$;

(b) for each assertion in $\mathcal{M}$ of the form $q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}}$, each object in $q_{\mathcal{G}}^{\mathcal{B}}$ is also an element of $q_{\mathcal{S}}^{\mathcal{D}}$, i.e., $q_{\mathcal{G}}^{\mathcal{B}} \subseteq q_{\mathcal{S}}^{\mathcal{D}}$.

Notice that, from the above semantics of the mapping $\mathcal{M}$, it follows that in our framework it is possible to express all the kinds of interpretations of the mapping assertions studied in data integration, namely the sound, complete, and exact interpretation (we refer to our previous project deliverable [22] for a detailed presentation of this topic). In particular, if we want to formulate a generic mapping assertion V defining a relationship between the query over the global schema $q_{\mathcal{G}}$ and the query over the source schema $q_{\mathcal{S}}$:

- a *sound* interpretation of V corresponds in our framework to the assertion $q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}$;
- a *complete* interpretation of V corresponds to the assertion $q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}}$;
- an *exact* interpretation of V corresponds to the pair of assertions $q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}$, $q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}}$.

In an analogous way, one can express sound, complete or exact interpretations of a mapping assertion defining a relationship between $q_{\mathcal{S}}$ and $q_{\mathcal{G}}$.

Given a source database $\mathcal{D}$ for $\mathcal{I}$, the semantics of $\mathcal{I}$ with respect to $\mathcal{D}$, denoted $sem(\mathcal{I}, \mathcal{D})$, is defined as follows:

$$sem(\mathcal{I}, \mathcal{D}) = \{ \ \mathcal{B} \mid \mathcal{B} \text{ is a legal global database for } \mathcal{I} \text{ w.r.t. } \mathcal{D} \ \}$$

Let us now turn our attention to queries. In order to define the semantics of a query q over the global schema $\mathcal{G}$, we have to take into account all the global databases legal for $\mathcal{I}$ with respect to $\mathcal{D}$. We call *certain answers* (or simply *answers*) of a query q with respect to $\mathcal{I}$ and $\mathcal{D}$, the set $q^{\mathcal{I},\mathcal{D}}$ of objects t such that $t \in q^{\mathcal{DB}}$ for *every* database $\mathcal{DB} \in sem(\mathcal{I}, \mathcal{D})$.

Furthermore, we call *possible answers* of a query q the set $q^{\mathcal{I},\mathcal{D}}$ of objects t such that $t \in q^{\mathcal{DB}}$ for *some* database $\mathcal{DB} \in sem(\mathcal{I}, \mathcal{D})$.

From the above definitions, it is easy to see that, in data integration, answering queries is essentially an extended form of reasoning in the presence of incomplete information [32]. Indeed, when we answer the query, we know only the extensions of the sources, and this provides us with only partial information on the global database.

2.3 Dealing with inconsistent data sources

According to the semantics above defined, in which we adopted a first-order logic interpretation of the mapping, it may be the case that the data retrieved from the sources cannot be reconciled in the global schema in such a way that both the constraints of the global schema and the mapping are satisfied [23]. As illustrated in [22], this happens, for instance, in a relational context, when a key constraint specified for the relation r in the global schema is violated by the tuples retrieved by the view associated to r, since the assumption of sound views does not allow us to disregard tuples from r with duplicate keys. If we do not want to conclude in this case that no global database exists that is legal for $\mathcal{I}$ with respect to $\mathcal{D}$, we need a different characterization of the mapping. In particular, we need a characterization that allows us to support query processing even when the data at the sources are incoherent with respect to the integrity constraints on the global schema.

A possible solution is to characterize the data integration system $\mathcal{I} = \langle \mathcal{G}, \mathcal{S}, \mathcal{M} \rangle$ in terms of those global databases that:

1. satisfy the integrity constraints of $\mathcal{G}$, and
2. approximate at best the satisfaction of the assertions in the mapping $\mathcal{M}$, i.e., that are *as close as possible* to the mapping $\mathcal{M}$.

In other words, the integrity constraints of $\mathcal{G}$ are considered "strong", whereas the mapping is considered "soft".

We now propose a modified definition of the semantics for the integration system that reflects the above idea. Given a source database $\mathcal{D}$ for $\mathcal{I}$, we define a partial order (based on set-containment) between the global databases for $\mathcal{I}$. If $\mathcal{B}_1$ and $\mathcal{B}_2$ are two databases that are legal with respect to $\mathcal{G}$, we say that $\mathcal{B}_1$ is *better* than $\mathcal{B}_2$ with respect to $\mathcal{D}$, denoted as $\mathcal{B}_1 \gg_{\mathcal{D}} \mathcal{B}_2$, if:

1. for each assertion $q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}$ in $\mathcal{M}$, $q_{\mathcal{G}}^{\mathcal{B}_1} \cap q_{\mathcal{S}}^{\mathcal{D}} \supseteq q_{\mathcal{G}}^{\mathcal{B}_2} \cap q_{\mathcal{S}}^{\mathcal{D}}$;
2. for each assertion $q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}}$ in $\mathcal{M}$, $q_{\mathcal{G}}^{\mathcal{B}_1} - q_{\mathcal{S}}^{\mathcal{D}} \subseteq q_{\mathcal{G}}^{\mathcal{B}_2} - q_{\mathcal{S}}^{\mathcal{D}}$;
3. at least one of the following conditions holds:
   (a) there exists an assertion $q_{\mathcal{S}} \sqsubseteq q_{\mathcal{G}}$ in $\mathcal{M}$ such that $q_{\mathcal{G}}^{\mathcal{B}_1} \cap q_{\mathcal{S}}^{\mathcal{D}} \supset q_{\mathcal{G}}^{\mathcal{B}_2} \cap q_{\mathcal{S}}^{\mathcal{D}}$;
   (b) there exists an assertion $q_{\mathcal{G}} \sqsubseteq q_{\mathcal{S}}$ in $\mathcal{M}$ such that $q_{\mathcal{G}}^{\mathcal{B}_1} - q_{\mathcal{S}}^{\mathcal{D}} \subset q_{\mathcal{G}}^{\mathcal{B}_2} - q_{\mathcal{S}}^{\mathcal{D}}$.

Intuitively, this means that there is at least one assertion for which $\mathcal{B}_1$ satisfies the sound mapping better than $\mathcal{B}_2$, while for no other assertion $\mathcal{B}_2$ is better than $\mathcal{B}_1$. In other words, $\mathcal{B}_1$ approximates the mapping better than $\mathcal{B}_2$.

It is easy to verify that the relation $\gg_{\mathcal{D}}$ is a partial order. With this notion in place, we can now define the notion of $\mathcal{B}$ satisfying the mapping $\mathcal{M}$ with respect to $\mathcal{D}$ in our setting: a database $\mathcal{B}$ that is legal with respect to $\mathcal{G}$ satisfies the mapping $\mathcal{M}$ with respect to $\mathcal{D}$ if $\mathcal{B}$ is maximal with respect to $\gg_{\mathcal{D}}$, i.e., for no other global database $\mathcal{B}'$ that is legal with respect to $\mathcal{G}$, we have that $\mathcal{B}' \gg_{\mathcal{D}} \mathcal{B}$.

The notion of legal database for $\mathcal{I}$ with respect to $\mathcal{D}$, and the notions of answers remain the same, given the new definition of satisfaction of mapping. It is immediate to verify that, if there exists a legal database for $\mathcal{I}$ with respect to $\mathcal{D}$ under the first-order logic interpretation of the mapping, then the new semantics and the old one coincide, in the sense that, for each query q, the set $q^{\mathcal{I},\mathcal{D}}$ of certain answers computed under the first-order semantics coincides with the set of certain answers computed under the new semantics.

3 Preliminary Structure of the System

Generally speaking, a data integration system provides a uniform access to a set of heterogeneous data sources, in such a way that the final users are freed from having to locate the sources relevant for their queries, separately interact with each source, and manually combine the data from different sources. The system has to offer the ablity to design, query, and maintain specific data integration applications. In particular, it has to support the designer in modeling the application domain, by providing flexible and powerful representation mechanisms that can mediate among different representations of overlapping worlds provided at the sources. Furthermore, it has to support final users in formulating queries to the system, and getting answers from the sources, by providing ad-hoc mechanisms for processing queries expressed on the global schema.

Functionalities concerning internal communication between the different components of the system must be guaranteed, and suitable transformations of data migrating through the system must be supported. Finally, the system should provide facilities to suitably access data sources and present their data in a specific form adopted within the system, i.e. provide assistance to integrate, maintain, and execute wrappers.

According to such requirements, the INFOMIX project plans to pursue a layered approach, in which the system features are divided into four levels of capabilities, as outlined in the project proposal. Specifically, the levels of the integration system are:

Final User Level: Generally speaking, final users pose queries to the integration system to obtain reconciled information from the set of autonomous and heterogeneous sources integrated by the system. Hence, at the higher level, a system *Interface* towards the final user has to support her/him in specifying queries and accessing the answers computed by the system.

Information Service Level: At the second level the system provides two types of main features:

- support for modeling the information in the data integration system, which is stored in an *Information model*. In particular, through the *Model Manager*, it provides automatic support to the verification of coherency, redundancy, and adequacy of the application specification. The Model Manager solves these reasoning tasks by methods that make use of an advanced *Computational Logic System*;

- support for reformulating each query expressed on the global schema in terms of a set of queries at the sources, by means of a *Query Reformulation* component. The automatic synthesis of such queries is based on reasoning methods developed both for the formalism expressing the global schema and for the mapping specification language. An important feature of the system is the ability of synthesizing queries at the sources for any form of mapping specified, which requires the implementation of different reformulation methods and techniques. As in the case of the Model Manager, the reasoning methods make use of the Computational Logic System.

Internal Integration Level: At the third level, the system presents an *Integration Engine*, which is based on computational logic and deductive database technology [20]. In particular, recent research results on information integration [23] indicate that the reformulation process may yield a query which can be suitably expressed in terms of disjunctive Datalog. Thus, on the basis of the reformulation process, the system could devise a query plan in terms of a disjunctive Datalog program, and execute it by applying suitable optimization strategies.

Data Acquisition and Transformation Level: Generally speaking, data integration systems integrate heterogeneous sources in which data are modeled in different ways. Hence, at the lower level, the system takes care of the problem of accessing the data sources. Access to heterogeneous data is transparently possible through a set of supported services, which include source location, data acquisition, and transformation of the data into the internal homogeneous data format.

4 System Functionalities

In this section we describe the general functionalities of a data integration system. Such functionalities are divided in four levels, according to the system structure outlined in the previous section. This classification has to be seen as a general classification comprising all the features that a system should provide at different levels. Actually, the INFOMIX system is intended to implement a significant subset of such features.

Functionalities for the final user level At this level the system provides support to the final user for formulating queries on the global schema and accessing intermediate and final results. More specifically, final user functionalities comprise:

- global schema browsing:

  The global schema represents the centralized view of the data exported by the integration system, and the entry point of all final user applications. Hence, the system should allow the final user to suitably inspect the global schema, e.g. by means of graphical navigation facilities, in order to make him exactly understand the structure of the global schema, and analyze the relationships and dependencies holding among the global elements. In other words, global schema browsing functionalities make the user understand which information can be extracted from the system, and support him in formulating queries.

- query formulation:

  The system must allow users to express their queries over the global schema. Basically, it has to provide an appropriate environment (textual or graphical) for query formulation, with standard text-edit functionalities, and support users with a specific query language. Furthermore, assistance to properly use the language features and exploit its expressive power, together with mechanisms to check the syntactic correctness of the formulated queries should be supplied.

- query answer presentation:

  The functionalities for query answer presentation are concerned with both the visualization of the query plan produced by the system, and the displaying of the final results obtained after the execution of the plan. Notice that the query plan is a conceptual specification of how it is possible to construct the objects in the answer to the query starting from a suitable set of objects extracted from the sources. Hence, as for the first aspect, the system should present the query plan in order to allow the user to refine it according to his needs. With regard to the second issue, the system should visualize the final answers in an easily understandable format. Furthermore, it should provide "preview" answers to queries. Indeed, in many cases, it may be useful for users to have quick access to some sort of approximate or partial answers during a long-running query, in order to get early feedback.

Functionalities for the information service level Functionalities for the information service level support for the design, the specification, and the maintenance of the data models that represent the integration domain. Furthermore, they allows for the reformulation and optimization of user queries posed on the global schema. More specifically, information service level functionalities comprise:

- specification and manipulation of the global schema:

  The main goal in specifying the global schema is to design it so as to provide an appropriate abstraction of all data residing at the sources. Hence, such a process has to be supported by a suitable data modeling language, allowing for the specification of rich information models, and by mechanisms to check the consistency of the realized schema. Furthermore, in order to cope with the dynamic nature of the integration applications and to perform changes in the schema, the system should provide suitable update functions and facilities to iteratively construct the global schema.

- specification and manipulation of the source schema:

  These functionalities are concerned with the definition, the maintenance and the update of the source schema. Such a schema provides the representations of the sources in the system according to the output produced by the functionalities of the data acquisition and transformation level.

  As for the global schema specification, the system has to supply suitable source modeling language and appropriate support mechanisms, allowing for the representation of the source elements and the relationships holding among them, and the specification of constraints on a single source schema, e.g., possible limitations on how each source can be queried. Furthermore, source schema update functions have to be guaranteed.

- specification and manipulation of the mapping:

  Such functionalities allow for the specification and the manipulation of relationships holding between elements of different schemas. They enable one to specify the mapping between the global schema and the source schema, and to support such a process by means of a specific mapping language. In particular, the system has to support the designer in the specification of GAV, LAV or more complex mapping assertions. Furthermore, it has to be possible to specify if the mapping assertion has to be considered exact, sound, or complete, and if it can be relaxed in order to deal with possible data inconsistencies. Finally, such functionalities should also allow one to declaratively specify rules involving elements of different sources, e.g., in order to indicate reconciliation operations that have to be properly taken into account to compute answers to user queries.

- query reformulation:

  Answers to queries expressed by the user in terms of the global schema have to be computed by the system only on the basis of data stored at the sources. For this purpose, the system should provide functionalities to re-express each query in terms of a suitable set of queries posed to the sources. In this reformulation process, the crucial step is deciding how to decompose the query on the global schema into a set of sub-queries on the source schema, based on the meaning of the mapping. Notice that, for LAV mappings, integration systems provide reformulation functionalities based on query rewriting techniques [14, 25, 29], also called query folding in the literature [30, 19], whereas for GAV mappings these functionalities generally rely on the simple query unfolding mechanism, which is considered sufficient to answer queries. Recent studies [7] show that in the presence of integrity constraints expressed on the global schema, also when the mapping is GAV, more complex reformulation functionalities are needed to properly process queries. Generally speaking, the system has to provide functionalities to properly reformulate queries in the presence of any kind of mapping: LAV, GAV or more general mappings. Query reformulation functions produce a query plan specifying how it is possible to construct the objects in the answer to the query from a suitable set of objects extracted from the sources.

- information service level optimization:

  As we already said, the reformulation process may in general produce a query plan expressed in terms of disjunctive Datalog [23]. Such a program represents the input to the the functionalities of the internal level (see below), which rely on a Computational Logic System. In order to cope with data-intensive applications in a Computational Logic System, the information service level has to provide optimization functionalities aiming at localizing the computation, allowing the logic system to focus on the relevant data where it is really needed to apply its complex reasoning capabilities.

Functionalities for the internal integration level The internal integration level functionalities are generally hidden from the users and give support to the functionalities of the other levels. More specifically, they comprise:

- internal integration level optimization:

  Differently from traditional database query optimization techniques, an aspect deserving particular attention in data integration is how to minimize the number of accesses to the sources while still being guaranteed to obtain all possible answers to a query. This can be performed by taking into account several aspects, e.g., challenges and opportunities provided by the redundant and overlapping information in sources, any available statistics, information about where data are stored, what wrapper they are associated with, and constraints and dependencies specified on the source logical schema.

Furthermore, internal optimization functionalities for the Computational Logic System have to be supplied by the integration level in order to make the system more suitable to deal with large input data. Several techniques can be adopted to perform such optimizations, as the usage of magic set methods and the improvement of internal data structure of the system.

The optimization functionalities produce an optimized query plan, generally called query execution plan.

- query evaluation

  Once a query has been reformulated and optimized, it is executed in order to produce the answer to the original query. In other words, query evaluation functionalities perform the query execution plan by suitably resorting to the functionalities of the data acquisition and transformation level in order to access data at the sources. The final results are then shipped to the final user level in order to be visualized by presentation functionalities.

Functionalities for the data acquisition and transformation level The task of the data acquisition and transformation level is to provide a layer in which all data stored at the sources are presented to the other components of the system in a uniform way. Functionalities for the data acquisition and transformation level comprise:

- internal data management:

  Such functionalities allow for the transformation of the data in the system internal format. Notice that different components of the system might rely on different data models: relational, conceptual, object-oriented, logic-based, etc. Such functionalities make the components of the system communicate with each other and manage data in their own specific format.

- data cleaning:

  When data migrate from the sources into the integration system, there may arise representation mismatches between data referring to the same real world object, errors in the data stored in the sources, or inconsistencies between values representing the properties of the real world objects in different sources [15]. Functionalities of the data acquisition and transformation level should allow for suitably cleaning, filtering and merging data from different sources, so as to resolve inconsistencies among values.

- source wrapper integration:

Wrappers are software components that encapsulate data sources in order to present them in a specific format, and handle the access to the data taking into account source capabilities to answer queries posed to the sources [26, 34]. They allow for the extraction of information from the sources that is relevant for the integration application, while discarding irrelevant data, and for representing possible integrity constraints specified on the source, e.g., limitation in accessing the sources. It is worth noticing that the structure and the behavior of wrappers depends on the type of the sources accessed, which are in general of different nature, ranging from relational databases to web sources, legacy systems, XML documents, etc. Hence, source wrapper integration functionalities should allow for the integration in the system of different wrappers designed for different data sources.

- wrapper execution:

  These functionalities accept queries expressed on the source schema, i.e., queries that are part of the query execution plan, and execute the associate wrapper. More in details, the executed wrappers transform queries on the source schema into appropriate queries to the physical data sources, i.e., translate them into sources' native query language, taking into account the query processing power of the sources, perform suitable additional processing, and return results so as they could be used by internal level functionalities.

5 Conclusions

In this document we have introduced the INFOMIX approach to information integration, by describing a comprehensive framework for data integration systems, defining a formal semantics, and providing a general specification of the main system functionalities.

In particular, our integration framework allows for the definition of constraints of general form on the global schema, and for the specification of complex constraint relationships between the local data sources and the global schema, i.e., our definition of mapping corresponds to a generalized form that comprises LAV and GAV as special cases. With regard to the semantics, we have proposed a very general approach in which "classical" exact, sound, and complete assumptions on the mapping assertions can be suitably relaxed to deal with possible inconsistencies of data at the sources with respect to integrity constraints expressed on the global schema.

Finally, on the basis of our formal model for data integration, and on the preliminary structure of the INFOMIX system presented in the project proposal, we have provided a functional specification of the data integration system. In particular, we have divided the system features into four levels of capabilities, as outlined in the project proposal. Generally speaking, we have listed capacities to properly design and maintain the information model, formulate queries on the global schema and re-formulate them in terms of queries at the sources, optimize and execute the reformulated queries, access the physical sources, acquire and transform their data in the system internal homogeneous data format.

References

[1] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Consistent query answers in inconsistent databases. In *Proc. of the 18th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'99)*, pages 68–79, 1999.

[2] Marcelo Arenas, Leopoldo E. Bertossi, and Jan Chomicki. Specifying and querying database repairs using logic programs with exceptions. In *Proc. of the 4th Int. Conf. on Flexible Query Answering Systems (FQAS 2000)*, pages 27–41. Springer, 2000.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002. To appear.

[4] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: the MOMIS project demonstration. In *Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, 2000.

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

[6] François Bry. Query answering in information systems with integrity constraints. In *IFIP WG 11.5 Working Conf. on Integrity and Control in Information System*. Chapman & Hall, 1997.

[7] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. In *Proc. of the 14th Conf. on Advanced Information Systems Engineering (CAiSE 2002)*, volume 2348 of *Lecture Notes in Computer Science*, pages 262–279. Springer, 2002.

[8] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 17th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, pages 386–391, 2000.

[9] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Data integration in data warehousing. *Int. J. of Cooperative Information Systems*, 10(3):237–271, 2001.

[10] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.

[11] M. J. Carey, L. M. Haas, P. M. Schwarz, M. Arya, W. F. Cody, R. Fagin, M. Flickner, A. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J. H. Williams, and E. L. Wimmers. Towards heterogeneous multimedia information systems: The Garlic approach. In *Proc. of the 5th Int. Workshop on Research Issues in Data Engineering –*

Distributed Object Management (RIDE-DOM'95), pages 124–131. IEEE Computer Society Press, 1995.

[12] Phan Minh Dung. Integrating data from possibly inconsistent databases. In *Proc. of the 4th Int. Conf. on Cooperative Information Systems (CoopIS'96)*, pages 58–65, 1996.

[13] Oliver Duschka. *Query Planning and Optimization in Information Integration.* PhD thesis, Stanford University, 1997.

[14] Oliver M. Duschka, Michael R. Genesereth, and Alon Y. Levy. Recursive query plans for data integration. *J. of Logic Programming*, 43(1):49–73, 2000.

[15] Helena Galhardas, Daniela Florescu, Dennis Shasha, and Eric Simon. An extensible framework for data cleaning. Technical Report 3742, INRIA, Rocquencourt, 1999.

[16] Hector Garcia-Molina, Yannis Papakonstantinou, Dallan Quass, Anand Rajaraman, Yehoshua Sagiv, Jeffrey D. Ullman, Vasilis Vassalos, and Jennifer Widom. The TSIMMIS approach to mediation: Data models and languages. *J. of Intelligent Information Systems*, 8(2):117–132, 1997.

[17] Cheng Hian Goh, Stéphane Bressan, Stuart E. Madnick, and Michael D. Siegel. Context interchange: New features and formalisms for the intelligent integration of information. *ACM Trans. on Information Systems*, 17(3):270–293, 1999.

[18] Gianluigi Greco, Sergio Greco, and Ester Zumpano. A logic programming approach to the integration, repairing and querying of inconsistent databases. In *Proc. of the 17th Int. Conf. on Logic Programming (ICLP'01)*, volume 2237 of *Lecture Notes in Artificial Intelligence*, pages 348–364. Springer, 2001.

[19] Jarek Gryz. Query folding with inclusion dependencies. In *Proc. of the 14th IEEE Int. Conf. on Data Engineering (ICDE'98)*, pages 126–133, 1998.

[20] Minker Jack, editor. *Logic-based Artificial Intelligence.* Kluver Academic Publishers, 2001.

[21] Thomas Kirk, Alon Y. Levy, Yehoshua Sagiv, and Divesh Srivastava. The Information Manifold. In *Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, pages 85–91, 1995.

[22] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Review on models and systems for information integration. Technical Report D1.1, Infomix Consortium, July 2002.

[23] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In *Proc. of the 9th Int. Workshop on Knowledge Representation meets Databases (KRDB 2002).* CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-54/`, 2002.

[24] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002)*, pages 233–246, 2002.

[25] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *Proc. of the 14th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'95)*, pages 95–104, 1995.

[26] Chen Li, Ramana Yerneni, Vasilis Vassalos, Hector Garcia-Molina, Yannis Papakonstantinou, Jeffrey D. Ullman, and Murty Valiveti. Capability based mediation in TSIMMIS. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 564–566, 1998.

[27] Jinxin Lin and Alberto O. Mendelzon. Merging databases under constraints. *Int. J. of Cooperative Information Systems*, 7(1):55–76, 1998.

[28] Ioana Manolescu, Daniela Florescu, and Donald Kossmann. Answering XML queries on heterogeneous data sources. In *Proc. of the 27th Int. Conf. on Very Large Data Bases (VLDB 2001)*, pages 241–250, 2001.

[29] Rachel Pottinger and Alon Y. Levy. A scalable algorithm for answering queries using views. In *Proc. of the 26th Int. Conf. on Very Large Data Bases (VLDB 2000)*, pages 484–495, 2000.

[30] Xiaolei Qian. Query folding. In *Proc. of the 12th IEEE Int. Conf. on Data Engineering (ICDE'96)*, pages 48–55, 1996.

[31] Jeffrey D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf. on Database Theory (ICDT'97)*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.

[32] Ron van der Meyden. Logical approaches to incomplete information. In Jan Chomicki and Günter Saake, editors, *Logics for Databases and Information Systems*, pages 307–356. Kluwer Academic Publisher, 1998.

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

[34] Ramana Yerneni, Chen Li, Hector Garcia-Molina, and Jeffrey D. Ullman. Computing capabilities of mediators. In *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, pages 443–454, 1999.

[35] Gang Zhou, Richard Hull, Roger King, and Jean-Claude Franchitti. Using object matching and materialization to integrate heterogeneous databases. In *Proc. of the 3rd Int. Conf. on Cooperative Information Systems (CoopIS'95)*, pages 4–18, 1995.