MO Universita' degli Studi di Modena

INTERDATA

Programma di ricerca (cofinanziato dal MURST, esercizio 1997)

Metodologie e tecnologie per la gestione di dati e processi su reti Internet e Intranet

MOMIS: An Intelligent System for the Integration of Semistructured and Structured Data

SONIA BERGAMASCHI, DOMENICO BENEVENTANO, MAURIZIO VINCINI, SILVANA CASTANO

T3-R07 **10 novembre 1998**

Sommario

Providing an integrated access to multiple heterogeneous sources is a challenging issue in global information systems. In this paper, we propose an I^3 system, called MOMIS, to information integration which takes into account both structured and semistructured data. Our goal is to provide an *integrated access* to heterogeneous information sources, allowing a user to pose a single query and to receive a single unified answer.

Tema	Tema 3: Integrazione di informazioni estratte da siti WWW
Codice	T3-R07
Data	10 novembre 1998
Tipo di prodotto	Rapporto tecnico
Unità responsabile	MO
Unità coinvolte	MO, BS
Autori	Sonia Bergamaschi, Domenico Beneventano, Maurizio Vincini, Silvana Castano
Autore da contattare	Sonia Bergamaschi Dipartimento di Scienze dell'Ingegneria Università degli Studi di Modena Via Campi 213/B, 41100 Modena, Italia sonia@dsi.unimo.it

MOMIS: An Intelligent System for the Integration of Semistructured and Structured Data

Sonia Bergamaschi Maurizio Vincini Silvana Castano

Sommario

Providing an integrated access to multiple heterogeneous sources is a challenging issue in global information systems. In this paper, we propose an I^3 system, called MOMIS, to information integration which takes into account both structured and semistructured data. Our goal is to provide an *integrated access* to heterogeneous information sources, allowing a user to pose a single query and to receive a single unified answer. A common standard data model and language, derived from the standard ODMG-93, and extended for integration purposes is introduced in MOMIS. The integration process is based on the analysis of schema descriptions of each source, which is given in form of *object patterns* in case of semistructured data sources. *Description Logics* plus *clustering techniques* constitute the theoretical framework and are exploited for constructing an integrated view of the information in the separate sources (i.e., global schema) and for query processing and optimization.

1 Introduction

Providing an integrated access to multiple heterogeneous sources is a challenging issue in global information systems. In this context, semistructured data has emerged as an important research topic, to model information sources that cannot be constrained by a schema (such as object repositories, file systems, document retrieval systems). A general representation of semistructured data is in terms of a graph-like or tree-like structure, which describes both the information constituting the schema and the data itself. Modeling an information source in terms of these structures has several advantages, especially for query processing and optimization [20]. Moreover, viewing structured sources in terms of semistructured data facilitates the development of mediator systems allowing a flexible data exchange paradigm across heterogeneous sources [2, 38]. For example, in the TSIMMIS (The Stanford-IBM Manager of Multiple Information Sources) project [32, 38], a semistructured data model called OEM (Object Exchange Model) is used for describing the sources to be integrated. It is a self-describing model where each data item has associated a descriptive label, without a strong typing system. The semantic knowledge is effectively encoded in the MSL (Mediator Specification Language) rules enforcing source integration at the mediator level. Although the generality and conciseness of OEM and MSL make this approach a good candidate for the integration of widely heterogeneous and semistructured information sources, a major drawback in such an approach is that dynamically adding sources is an expensive task. In fact, new TSIMMIS sources not only must be wrapped, but the mediators that uses them have to be redefined and their MSL definitions recompiled. The administrator of the system must figure out weather and how to use the new sources.

The issue of adding structure (also called schema) to semistructured data has been recently investigated in the literature specially for browsing and for query optimization purposes. In particular, in [43], the notion of *dataguide* has been proposed as a "loose description of the structure of the data" actually stored in an information source. For example, a dataguide may

say that *Person* in a source can have outgoing edges labeled *name*, *address*, and *qualification*, that an *address* is either a string, but that it can be defined also in terms of *city*, *street*, and *zipcode*. The existence of a path in the dataguide can be exploited to determine information relevant in the source for answering a query.

Goal of this paper is to present an I^3 system, called MOMIS (Mediator envirOnment for Multiple Information Sources), for the integration of structured and semistructured data sources [12]. MOMIS has been conceived in the framework of the *INTERDATA* project, as a joint collaboration between University of Milano and University of Modena. MOMIS relies on two pre-existing tools, namely ARTEMIS (Analysis of Requirements: Tool Environment for Multiple Information Systems) and ODB-Tools and integrates them to provide an I^3 architecture, feasible for integration and query optimization.

A common data model (ODM_{I^3}) and language (ODL_{I^3}) are adopted in MOMIS to describe integration information regarding the sources. ODM_{I^3} and ODL_{I^3} are defined as a subset of the corresponding ones in ODMG-93 [31], following the proposal for a standard mediator language developed by the I^3/POB working group [21]. Moreover, ODL_{I^3} introduces new constructors to support the integration process. Besides ODB-Tools, to support inference and reasoning in source integration and query optimization, MOMIS relies on a Description Logics, OLCD (Object Language with Complements allowing Descriptive cycles [11, 13]), as a kernel language and ODB-Tools as the supporting tool [8, 9, 10].

To support a semi-automatic, schema-based integration process, MOMIS relies on hierarchical clustering techniques and on ARTEMIS as the supporting tool.

First, a Common Thesaurus of *terminological relationships* is constructed, based on ODL_{I^3} source descriptions, to provide the a relevant tool for the identification of the information to be integrated in the different sources. Secondly, clustering techniques are employed to identify ODL_{I^3} classes describing similar information in different sources, which are merged into an integrated global description of the analyzed sources. Original contributions of this paper w.r.t. previous proposals (and in particular [12]), are related to the following aspects:

- Modeling schema information from the sources of semistructured data, that we call *object patterns*. In this sense, we investigate the same issue of [43]. ODL_{I^3} and the corresponding description logics-based extensions are discussed, for a proper representation of object patterns in the system to support both the integration and query processes.
- Extension of the affinity-based clustering procedure for dealing with object patterns in the integration process.
- Query processing and optimization using the knowledge embedded in the obtained global schema are discussed, specially for the aspects related to object patterns and semistructured data sources. In this context, ODB-Tools support for global query decomposition and optimization is described.

The paper is organized as follows. In Section 2, we outline the approach to schema integration, give basic notations and assumptions regarding semistructured data together with a running example used throughout the paper. In Section 3 we introduce the ODL_{I^3} language, the OLCD description logics and its inference techniques. In Section 4, we describe the construction of a Common Thesaurus of terminological relationships. In Section 5, we illustrate the affinity-based technique for the analysis of ODL_{I^3} schemas and the clustering process for the formulation of groups of classes with affinity. In Section 6, we describe the unification process for building global patterns which leads to the construction of the mediator global schema. In Section 7 we show the effectiveness of the global schema with respect to query processing

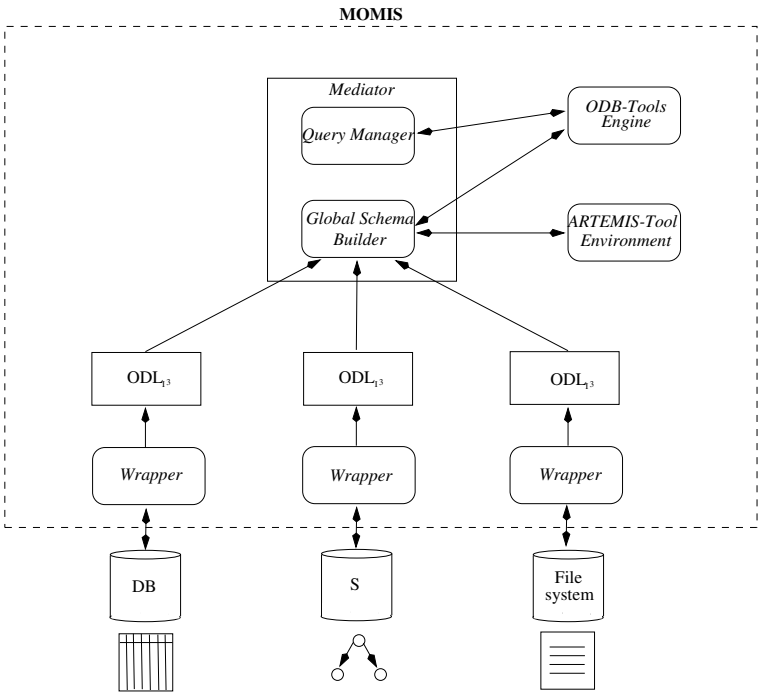

Figura 1: Architecture of MOMIS

and optimization. In Section 8, we discuss previous work on semistructured data modeling and heterogeneous information integration. Finally, in Section 9 we give our concluding remarks.

2 MOMIS architecture and basic concepts of the system

We describe the MOMIS architecture and introduce the basic concepts which will be used throughout the paper.

2.1 System architecture

The MOMIS system has been conceived to provide an integrated access to heterogeneous information stored in traditional databases (e.g., relational, object-oriented) or file systems, as well as in semistructured sources. Following the basic I^3 architecture [6], four major components are identified in MOMIS (see Fig. 1):

- One or more *Wrappers* on top of sources that are responsible for translating the schema of a data source into the common ODL$_{I^3}$ language. A wrapper performs also the translation of a query expressed in the ODL$_{I^3}$ language into a local request executable by the query processor of the corresponding source.
- The *Mediator* composed of two modules: the *Global Schema Builder* (GSB) and the *Query Manager* (QM). The GSB module combines, integrates, and refines data received from wrappers. The QM module performs query processing and optimization. In particular, it generates the OQL$_{I^3}$ queries for wrappers, starting from a single query formulated by the user on the global schema. Using Description Logics techniques, the QM component can generate in an automatic way the translation of a a given submitted query into separate sub-queries for local sources.

- The *ODB-Tools Engine*, a tool based on the OLCD Description Logics [11, 13] which performs schema validation and query optimization [8, 9, 10].
- The *ARTEMIS-Tool Environment*, a tool performing schema analysis and clustering [25].

The goal of this paper is to describe how the *integration* of structured and semistructured data sources and *query processing and optimization* are performed in MOMIS.

Integration process. The integration of structured and semistructured data sources is performed in a semi-automatic way, by exploiting schema ODL_{I^3} descriptions of the sources with a combination of Description Logics and clustering techniques. In particular, the following activities are performed:

1. *Extraction of terminological relationships*, with the ODB-Tools support. In this step, a Common Thesaurus of *terminological relationships* is constructed. Terminological relationships express inter-schema knowledge among different sources and corresponds to intensional assertions used in [30]. Terminological relationships are derived in a semi-automatic way from ODL_{I^3} schema descriptions, by analyzing structure and context of classes, by using ODB-Tools and the Description Logics techniques. Extraction of terminological relationships will be described in Section 4.
2. *Affinity-based clustering of ODL_{I^3} classes*, with the support of the ARTEMIS-Tool environment. Terminological relationships in the Thesaurus are used to evaluate the level of *affinity* between ODL_{I^3} classes in order to identify the information to be integrated at the global level. To this end, ARTEMIS computes proper coefficients that measure the level of affinity of ODL_{I^3} classes based on their names and attributes. ODL_{I^3} classes with affinity are grouped together using hierarchical clustering techniques [36].
3. *Construction of the mediator global schema*, with the support of ODB-Tools. Affinity clusters of ODL_{I^3} classes are analyzed to construct the global schema of the Mediator. An integrated global ODL_{I^3} class is defined for each cluster, which is representative of all classes belonging to the cluster, and is characterized by the union of their attributes. The set of global ODL_{I^3} classes defined constitutes the global schema of the Mediator to be used for posing queries against the sources. In this phase, OLCD and ODB-Tools are exploited for a semi-automatic generation of the global ODL_{I^3} classes.

Query processing and optimization. When a user submits a query on the global schema, MOMIS takes the query and produces a set of sub-queries to be sent to each involved information source. According to other approaches proposed in this area [4, 5], this process consists of two main activities:

1. *Semantic optimization.* The Mediator operates on the query by exploiting semantic optimization techniques supported by ODB-Tools in order to reduce the query access plan cost. Optimization is based on logical inference from knowledge on integrity constraint rules of the mediator global schema.
2. *Query plan formulation.* Once the optimized query has been obtained, a set of subqueries is generated for all involved sources. To this end, the Mediator exploits mappings associated with global ODL_{I^3} classes to express the global optimized query in terms of local source schemas. Semantic optimization and query plan formulation will be described in Section 7.

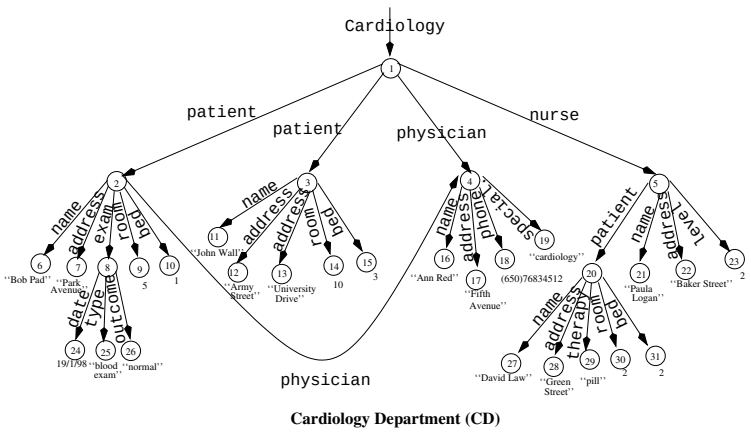

Figura 2: Cardiology department (CD)

The ODL$_{I^3}$ and the OLCD Description Logics languages employed in MOMIS will be briefly introduced in Section 3. While for conventional structured databases, the description of the schema is always available and can be translated in ODL$_{I^3}$ by a wrapper, for semistructured sources this is not. For this reason, additional schema extraction procedures are necessary to associate a schema with a semistructured source (e.g., in form of dataguides [43]). In MOMIS, we introduce the concept of *object pattern* for representing the structure of objects of a semistructured source at the intensional level, as we will discuss in the following section.

2.2 Semistructured data and object patterns

The basic characteristic of semistructured data is that they are "self-describing". This means that the information generally associated with a schema is specified within the data. Models for semistructured data have been proposed in the literature [20, 22, 52]. According to these models, semistructured sources are represented as rooted, labeled graph with the semistructured data (e.g., an image or free-form text) as nodes and labels on edges. Fig. 2 shows an example of semistructured source, containing information related to the `Cardiology Department` of a given hospital. In the paper, we suppose that a semistructured source is conform with the OEM model [2, 52], and we recall the semistructured object definition. A semistructured object (object, for short) can be viewed as a triple of the form $\langle$`id`, `label`, `value`$\rangle$, where `id` is the object identifier, `label` is a string describing what the object represents, and `value` is the value, that can be atomic or complex. The atomic value can be integer, real, string, image, while the complex value is a set of semistructured objects, that is, a set of pairs (*id,label*). In analogy with this characterization, we define an atomic semistructured object as an object with an atomic value, and complex semistructured object as an object whose value is complex. For example, $\langle$`16`, `name`, "`Ann Red`"$\rangle$ represents an atomic semistructured object with string value, while $\langle$`8`, `exam`, {`(24, date), (25, type), (26, outcome)`}$\rangle$ represents a complex semistructured object whose value is composed by/given by three atomic objects. A complex object can be thought as the parent of all the objects that form its value (children objects). A given object can have one or more parents. We denote the fact that an object so' is a child object of another object so by $so \rightarrow so'$ and use notation $label(so)$ to denote the label of so. Thus, intuitively, a semistructured source S can be viewed as a directed graph. Nodes represent semistructured objects and directed edges represent the relationship between a complex object and all the objects representing its value.

Patient-pattern = (Patient,{name,address,exam*,room,bed,therapy*,physician*})
Physician-pattern = (Physician,{name,address,phone,specialization})
Nurse-pattern = (Nurse,{name,address,level,patient})
Exam-pattern = (Exam,{date,type,outcome})

Figura 3: The object patterns for the semistructured source in Fig. 2

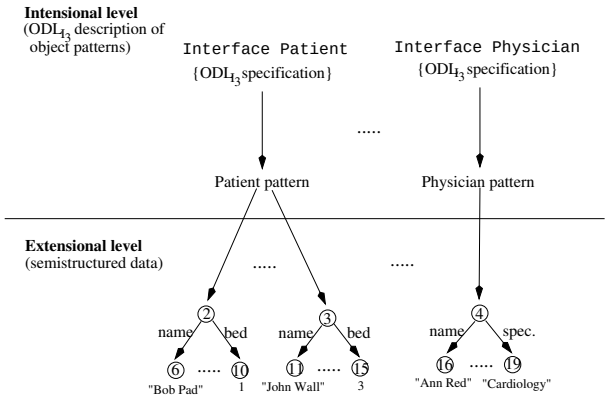

Figura 4: Abstraction levels of semistructured objects in MOMIS

In semistructured data models, labels are descriptive as much as possible and can be used to group objects by assigning the same label to related objects. For this reason, in order to discover the different kind of objects stored in a semistructured source S, we adopt an approach based on patterns, which relies on the open world semantics of the Description Logics. The approach works as follows. All the complex objects so of S are partitioned into disjoint sets, denoted set_l, such that all objects belonging to the same set have the same label l. Subsequently, we associate an *object pattern* with each set. Formally, an object pattern is defined as follows.

Definition 1 (Object pattern) *Let set_l be a set computed on a semistructured source S. The* object pattern *of set_l is a pair of the form $\langle l,A\rangle$, where l is the label of the objects belonging to set_l, and $A = \bigcup label(so')$ such that there exists at least one object $so \in set_l$ with $so \rightarrow so'$.*

From this definition, an object pattern is representative of all different objects that describe the same concept in a given semistructured source. In particular, l denotes the concept and set A the properties (or attributes) characterizing the concept in the source. Since semistructured objects can be heterogeneous, labels in the set A of an object pattern can be defined only for some of the objects in set$_l$, but not for all. We call such kind of labels "optional" and denote them with symbol "*".

Object patterns for all the objects in our semistructured source are shown in Fig. 3. Four object patterns are defined: `Patient` containing information about patients; `Physician` and `Nurse` containing information about physicians and nurses, respectively; `Exam` containing information about `date`, `type`, and `outcome`.

Object pattern description follows an *open world semantics* typical of the Description Logics approach [14, 17, 24, 57]. Objects of a pattern share a common minimal structure represented by non optional properties, but can have further additional (i.e., optional) properties. In this way, objects in a semistructured data source can evolve and add new properties, but they will be retrieved as valid instances of the corresponding object pattern when processing a query.

Intensive Care Department (ID)

```
Patient(code, first_name, last_name, address, test, doctor_id)
Medical_Staff(id, first_name, last_name, phone, address, availability, position)
Test(number, type, date, laboratory, result)
Dis_Patient(code, date, note)
```

Figura 5: Intensive Care Department (ID)

At the intensional level ODL_{I^3} description for each type of information stored in S is provided. The extensional level is the level of semistructured objects (see Fig. 4).

2.3 Running example

In order to illustrate the way our approach works, we will use the following example of integration in the medical organization domain.
Consider the `Cardiology` and `Intensive Care` departments of a given hospital, needing to share information about their patients. The `Cardiology` department (`CD`) contains semistructured objects about patients with ischemic heart diseases, hypertension, and about physicians and nurses who have access in the line of duty to information concerning patient's health. Fig. 2 illustrates a portion of the data. There is one complex root object with four complex children objects, two patients, one physician, and one nurse. (In the real applications, there are many patients, physicians, and nurses). Each `Patient` has an atomic `name`, `address`, `room`, and `bed`. `Physician` contains as its value four atomic objects (i.e., `name`, `address`, `phone`, and `specialization`), while `Nurse` contains as its value three atomic objects (i.e., `name`, `address`, and `level`) and one complex object (i.e., the `patient` of which is responsible).

In the `Intensive Care` department (`ID`) there is a relational database containing information similar to information of the `Cardiology` department: it stores data both on patients with diagnoses of trauma, myocardial infarction and on medical staff. There are four relations: `Patient`, `Medical_Staff`, `Test`, and `Dis_Patient` (see Fig. 5). Personal information related to patients is maintained into the `Patient` relation. `Dis_Patient` instance is a subset of `Patient` instance and contains information about discharged patients. Each "clinician", that is, a licensed professional such as doctor, nurse, pharmacist (`Medical_Staff` relation) has a `first_name`, `last_name`, `phone`, `address`, `availability`, and `position`. The relation `Test` is used to maintain information regarding all patient tests.

3 The ODL_{I^3} language and OLCD

In this section, we describe the ODL_{I^3} language and informally describe the OLCD description logics. ODL_{I^3} descriptions are translated into OLCD descriptions in order to perform inference tasks typical of description logics. Two example of translation particularly useful for semistructured data types are given.

3.1 The ODL_{I^3} language

In order to easily communicate source descriptions between wrappers and the mediator engine, we introduce a high level data description language, called ODL_{I^3}. According to recommendations of [21], and to the diffusion of the object data model (and its standard ODMG-93), ODL_{I^3} is very close to the ODL language, supporting requirements of our intelligent information integration systems. ODL_{I^3} is a source independent language used by the mediator to manage the

system in a common way (we suppose to deal with different source types, such as structured and semistructured data).

The main extensions w.r.t. ODL (see Appendix A) is the capability of expressing: two kinds of rules, the *union* (`union`) operator, the *optional* (*) constructor and *terminological relationships* intra and inter sources.

The rules are *if then* rules, expressing in a declarative way integrity constraints intra and inter sources, and *mapping* rules between sources. The union operator permits the expression of alternative data structures in a class definition, thus capturing the flavour of semistructured data. An example of its use and its OLCD translation will be shown in the following. The symbol * after an attribute name specifies that this attribute is optional. Terminological relationship permit to represent synonymy (SYN), hypernymy (BT), hyponymy (NT) and positive associations (RT) between two types.

Translation of any particular source schema into a ODL_{I^3} corresponding schema is performed by a wrapper. Moreover, the wrapper is also responsible for adding information useful to the mediator, such as the source name and type [1].

Object patterns and source schemas are translated into ODL_{I^3} descriptions. The translation into ODL_{I^3} is straightforward, on the basis of the ODL_{I^3} syntax and of the object pattern definition. In particular, given a pattern $\langle l, A \rangle$ or a relation of a relational source we have that: i) an ODL_{I^3} class name corresponds to l or to the relation name, respectively, and ii) for each label $l' \in A$ or relation attribute, an attribute is defined in the corresponding ODL_{I^3} class. As an example, `CD.Patient` and `ID.Medical_Staff` will be described as follows:

```
interface Patient

 ( source semistructured
      Cardiology_Department )
{  attribute string    name;
   attribute string    address;
   attribute set<Exam> exam*;
   attribute integer   room;
   attribute integer   bed;
   attribute string    therapy*;
   attribute set<Physician> physician*;
   };
```

```
interface Medical_Staff

 ( source relational Intensive_Care
   extent Medical_Staffers)
   key id )
{  attribute string  id;
   attribute string  first_name;
   attribute string  last_name;
   attribute integer phone;
   attribute string  address;
   attribute string  availability;
   attribute string  position; };
```

A special case occurs when "highly" heterogeneous objects are defined in semistructured sources. With reference to our semistructured source CD of Fig. 2, consider the semistructured object:
$so = \langle 2, \texttt{patient}, \{(6, \texttt{name}), (7, \texttt{address}), (8, \texttt{exam}), (9, \texttt{room}), (10, \texttt{bed})\}\rangle$.

Suppose that the patient object `3` is present with a non-atomic `address` object, that is, `address` contains as its value three atomic objects (`city, street, zipcode`) (see Fig. 6). In this case, pattern extraction should proceed in a slightly different way. In particular, a new pattern for address is extracted, $\texttt{address-pattern} = (\texttt{address}, \{\texttt{city}, \texttt{street}, \texttt{zipcode}\})$. To take into account this kind of heterogeneity, in ODL_{I^3} we define the special constructor **union** for a class. The specification in ODL_{I^3} of `Patient` and `address` patterns is shown in Figure 7.

The semantics of the union constructor and of optional attributes in ODL_{I^3} will be discussed in the next section, using the OLCD Description Logics.

[1] In general, the wrapper will provide the description of a portion of a source schema, namely the source subschema the source exports for integration and thus made accessible to the mediator.

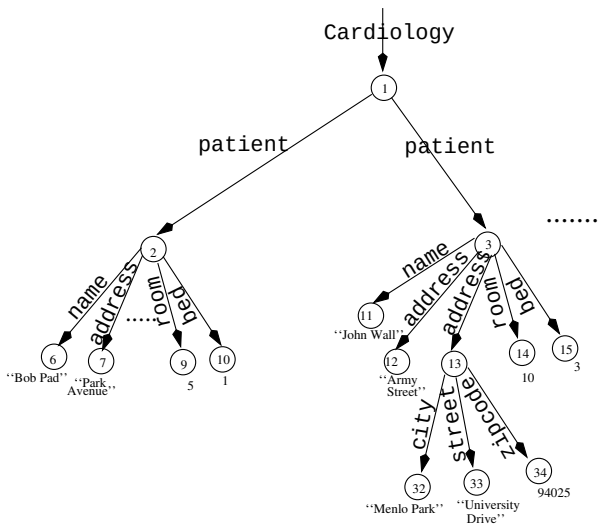

Figura 6: An example of highly heterogeneous objects

```
interface Address
 ( source semistructured
      Cardiology_Department )
{  attribute string city;
   attribute string street;
   attribute string zipcode; };
 union
{  string; };
```

```
interface Patient
   ( source semistructured
        Cardiology_Department )
{  attribute string   name;
   attribute Address  address;
   ............................. };
```

Figura 7: An example of union type

3.2 OLCD: a formalism for complex objects including integrity constraints

In this section, we give an intuitive description of OLCD. Readers interested in a formal account can refer to [11]. OLCD is an extension of the *object description language* ODL (not to be confused with ODL-ODMG93), introduced in [13] and is in the tradition of complex object data models [1, 46]. OLCD, as its ancestor ODL, provides a *system of base types*: string, boolean, integer, real; the type constructors *tuple*, *set* and *class* allow the construction of complex value types and class types. Class types (also briefly called *classes*) denote sets of *objects with an identity and a value*, while *value types* denote sets of *complex, finitely nested values without object identity*. In addition, an intersection operator can be used to create intersections of previously introduced types allowing simple and multiple inheritance specialization and an union operator ($\sqcup$) can be used to create unions of previously introduced types allowing to express generalization.

Finally, types can be given names. Named types come in two flavours: a named type may be *primitive* that means the user has to specify an *element*'s membership in the interpretation of the name or *virtual* and in such a case its interpretation is computed.

The extensions to ODL introduced in OLCD are: *quantified path types*, *integrity constraint rules* and *union* ($\sqcup$) constructor. The first extension has been introduced to deal easily and powerfully with nested structures. Paths, which are essentially sequences of attributes, represent the central ingredient of object-oriented query languages to navigate through the aggregation

hierarchies of classes and types of a schema [15].

In particular, following [15], we provide *quantified* paths to navigate through set types. The allowed quantifications are existential and universal and they can appear more than once in the same path. A path type is a type associating with a path to a type of the formalism. Therefore, by means of path types, we can express a class of integrity constraints.

The second extension allows the declarative expression of integrity constraints represented as *if then rule* universally quantified over the elements of the domain with an antecedent and a consequent which are types of the formalism.

The *union* ($\sqcup$) operator can be used to represents the semantics of the `union` operator of the ODL$_{I^3}$. It has been formalized in [11], with the meaning of the union of all possible union attribute instances.

For example, the `Address` pattern of Figure 7 is translated in OLCD as follows:

σ_V(`Address`) = [`city` : `String`, `street` : `String`, `zipcode` : `String`] $\sqcup$ `String`

The *union* ($\sqcup$) operator is also useful to define in our formalism optional attributes. In fact an optional attribute specify that a data may exist or not for a given instance. In OLCD we describe this as union betweeen the attribute domain and the *bottom* type ($\bot$).

For our example, the `Patient` pattern can be represented as follows[2]:

σ_P(`Patient`) = $\triangle$ [`name` : `String`, `address` : `String`, `exam` : {`Exam`} $\sqcup$ $\bot$,
`room` : `Integer`, `bed` : `Integer`, `therapy` : `String` $\sqcup$ $\bot$,
`physician` : {`Physician`} $\sqcup$ $\bot$]

Description Logics, and thus OLCD, permits, by exploiting virtual types semantics, and, given a *type as set* semantics to type descriptions, to provide relevant reasoning techniques: computing *subsumption* relations between types (i.e. "isa" relationships implied by types descriptions), deciding *equivalence* between types and detecting *incoherent* (i.e., always empty) types.

As a subsumption example in the context of optional attributes, let us suppose to consider the value types `A` and `B`,

σ_V(`A`) = [`att1` : `String`, `att2` : `String`]
σ_V(`B`) = [`att1` : `String`, `att2` : `String` $\sqcup$ $\bot$]

By computing subsumption between types `A` and `B` we obtain that `B` subsumes `A` (A isa B) even if it has not been explicitely declared.

We developed a prototype system, called ODB-Tools engine [8, 9, 10], based on OLCD and implementing the above reasoning techniques, available via Internet at [10].

4 Extraction of terminological relationships

The goal of this activity is the construction of a Thesaurus of terminological relationships describing common knowledge about ODL$_{I^3}$ classes and attributes of source schemas. For this reason, it is called *Common Thesaurus*.

The following kinds of terminological relationships are represented in the Common Thesaurus:

- SYN (Synonym-of), defined between two terms t_i and t_j, with $t_i \neq t_j$, that are considered synonyms, that is, that can be interchangeably used in every considered source, without changes in meaning. An example of SYN relationship in our running example is ⟨`ID.Test` SYN `CD.Exam`⟩.[3] SYN is symmetric, that is, t_i SYN $t_j \Rightarrow t_j$ SYN t_i.

[2] σ_P and σ_V introduces primitive types and virtual types respectively

[3] We use dot notation for specifying the source where a given term is used.

- BT (Broader Terms), or hypernymy, defined between two terms t_i and t_j such as t_i has a broader, more general meaning than t_j. An example of BT relationship in our example is ⟨`ID.Medical_Staff` BT `CD.Nurse`⟩. BT relationship is not symmetric. The opposite of BT is NT (Narrower Terms), that is, t_i BT $t_j \Rightarrow t_j$ NT t_i.

- RT (Related Terms), or positive association, defined between two terms t_i and t_j that are generally used together in the same context. For example, we can have the following relationship ⟨`CD.Patient` RT `CD.Exam`⟩. RT is symmetric, that is, t_i RT $t_j \Rightarrow t_j$ RT t_i.

Discovering terminological relationships from source schemas is a semi-automatic activity in MOMIS. The designer is assisted by ODB-Tools and the activity proceeds in the following steps:

1. **Automated extraction of relationships from source schemas.**

   By exploiting ODB-Tools capabilities and semantically rich schema descriptions, an initial set of BT, NT, and RT can be automatically extracted. In particular, by translating ODL_{I^3} into OLCD descriptions, ODB-Tools infers BT/NT relationships among classes directly from generalization hierarchies, and RT relationships from aggregation hierarchies, respectively. Other RT relationships are extracted from the expression of foreign keys in relational source schemas. A special case is represented by a foreign key that is the primary key both in the original and in the referenced tables: in this case a BT/NT relationship can be extracted. (This case holds for `ID.Dis_Patient` and `ID.Patient` (see Appendix B))

   In case of semistructured sources, RT relationships are extracted, due to the nature of relationships in the semistructured data model.

   Example 1 Consider the `Cardiology` and `Intensive Care` departments. The set of terminological relationships automatically extracted by ODB-Tools are the following:

   ⟨`ID.Patient` RT `ID.Medical_Staff`⟩
   ⟨`ID.Patient` RT `ID.Test`⟩
   ⟨`CD.Patient` RT `CD.Physician`⟩
   ⟨`CD.Patient` RT `CD.Exam`⟩
   ⟨`ID.Patient` BT `ID.Dis_Patient`⟩
   ⟨`CD.Nurse` RT `CD.Patient`⟩

2. **Revision/integration of relationships.** Additional terminological relationships can be added to the initial automatically extracted set. The designer can interact with the tool and supply new terminological relationships. Terminological relationships that can be interactively supplied are those regarding synonyms and, in general, those holding according to specific domain knowledge. For instance, since in our example generalization hierarchies cannot be expressed in the source data models (semistructured and relational), BT/NT relationships are manually supplied by the designer.
   An aid in this stage can be provided to the designer by exploiting an underlying lexical system (see for example WordNet [40, 50]). In this case, synonyms, hypernyms, hyponyms, related terms may be automatically proposed to the designer, by selecting them according to relationships predefined in the lexical system.

   Explicit terminological relationships, in general, may correlate ODL_{I^3} classes whose types presents conflicts with respect to the semantics of subsumption/generalization and equivalence relationships, which force to supertype/subtype and the same type relations to hold between ODL_{I^3} classes, respectively.

For example, suppose that a SYN relationship is defined between two classes having different structures, say `ID.Test` and `CD.Exam` (see Appendix B). In order to translate this terminological relationship into a *valid equivalence relationship* for ODB-Tools it is necessary to "uniform" the types of both classes, i.e., to give the same structure to both classes. The resulting modified ODL_{I^3} classes are:

```
interface Test                          interface Exam
( ... )                                 ( ... )
{ attribute string  type;               { attribute integer date;
  attribute integer date;                 attribute string  type;
  attribute string  laboratory;           attribute string  outcome;
  attribute string  result;               attribute string  laboratory;
  attribute string  outcome; };           attribute string  result; };
```

The same problem arises for the previously inferred BT relationship between `ID.Patient` and `ID.Dis_Patient`. Translating this terminological relationship into a *valid subsumption relationship* for ODB-Tools implies, in order to avoid strict inheritance violation, to modify both the original classes (shown in Appendix B) as follows:

```
interface Patient                       interface Dis_Patient
( ...  )                                ( ...  )
{ attribute string  code;               { attribute string  code;
  attribute string  first_name;           attribute integer date;
  attribute string  last_name;            attribute string  note;
  attribute string  address;              attribute string  position;
  attribute string  test;                 attribute string  first_name;
  attribute string  doctor_id; };         attribute string  last_name;
                                          attribute string  address;
                                          attribute string  test;
                                          attribute string  doctor_id; };
```

At last, when an RT terminological relationship is asserted, for example between `CD.Patient` and `ID.Test`, a new aggregation attribute is defined as follows:

```
interface Patient                                interface Test
( ... )                                          ( ... )
{  attribute string          name;               {  attribute string  type;
   attribute string          address;               attribute integer date;
   attribute set<Exam>       exam*;                 attribute string  laboratory;
   attribute integer         room;                  attribute string  result;};
   attribute integer         bed;
   attribute string          therapy*;
   attribute set<Physician>  physician*;
   attribute Test            newRT1};
```

These tranformations lead to the automatic generation of a "virtual schema" containing a modified description of each local schema and are used by ODB-Tools to perform further inferences in order to enrich the Thesaurus.

Example 2 In our example, suppose that the designer supplies the following terminological relationships for classes and attributes:

⟨ID.Medical_Staff BT CD.Physician⟩
⟨ID.Medical_Staff BT CD.Nurse⟩
⟨ID.Test SYN CD.Exam⟩
⟨ID.Patient SYN CD.Patient⟩
⟨CD.Patient.physician BT ID.Patient.doctor_id⟩
⟨CD.Patient.name BT ID.Patient.first_name⟩
⟨CD.Patient.name BT ID.Patient.last_name⟩
⟨CD.Nurse.level SYN ID.Medical_Staff.position⟩
⟨CD.Exam.outcome SYN ID.Test.result⟩

3. **Validation of relationships.** In this step, ODB-Tools is employed to validate terminological relationships defined for attributes in the Thesaurus. Validation is based on the compatibility of domains associated with attributes. This way, *valid* and *invalid* terminological relationships are distinguished. In particular, let $a_t = \langle n_t, d_t \rangle$ and $a_q = \langle n_q, d_q \rangle$ be two attributes.[4]

   The following checks are executed on attribute's name relationships using ODB-Tools:

   - $\langle n_t$ SYN $n_q \rangle$: the relationship is marked as valid if d_t and d_q are equivalent, or if one is more specialized than the other;
   - $\langle n_t$ BT $n_q \rangle$: the relationship is marked as valid if d_t contains or is equivalent to d_q;
   - $\langle n_t$ NT $n_q \rangle$: the relationship is marked as valid if d_t is contained in, or is equivalent to d_q.

   When the attribute domain contains the union connector, a *valid relationship* is obtained when at least one domain is compatible. Such validation is performed by ODB-Tools exploiting subsumption and equivalence computation.

   Example 3 Referring to the Thesaurus defined in example 2, we report the output of the validation phase: for each relationship, control flag [1] denotes a valid relationship while control flag [0] an invalid one.

   ⟨CD.Patient.physician BT ID.Patient.doctor_id⟩ [0]
   ⟨CD.Patient.name BT ID.Patient.first_name⟩ [1]
   ⟨CD.Patient.name BT ID.Patient.last_name⟩ [1]
   ⟨CD.Nurse.level SYN ID.Medical_Staff.position⟩ [0]
   ⟨CD.Exam.outcome SYN ID.Test.result⟩ [1]

4. **Inferring new relationships.** In this step, inference capabilities of ODB-Tools are exploited. A new set of terminological relationships is inferred by ODB-Tools, by using the "virtual schema" defined in the revision/integration step and by calculating new subsumption and aggregation relationships. These new semantic relationships are then translated into terminological relationships enriching thus the Thesaurus containing both explicit and inferred relationships. The result is the so-called Common Thesaurus.

   Example 4 With reference to our example, terminological relationships inferred from explicit ones are:

[4] We assume that an attribute a is defined as a pair $a = \langle n, d \rangle$ where n is the name and d is the domain, respectively.

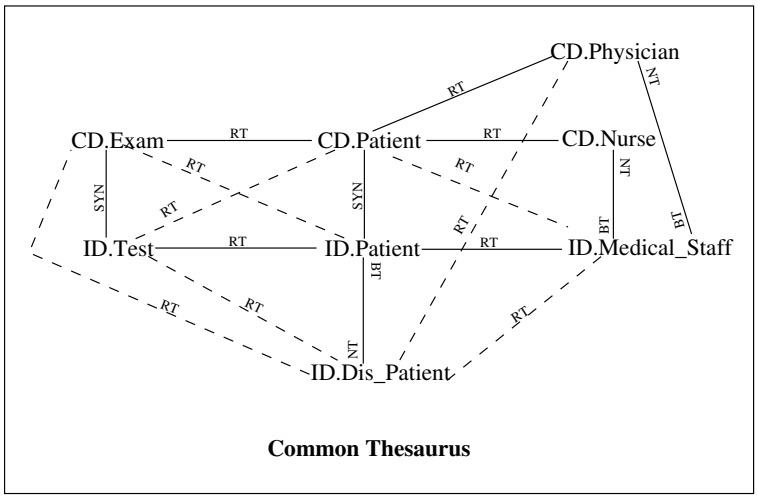

Figura 8: Common Thesaurus for `Cardiology` and `Intensive Care` department

⟨`ID.Patient` RT `CD.Physician`⟩
⟨`ID.Patient` RT `CD.Exam`⟩
⟨`CD.Patient` RT `ID.Medical_Staff`⟩
⟨`CD.Patient` RT `ID.Test`⟩
⟨`ID.Dis_Patient` RT `ID.Medical_Staff`⟩
⟨`ID.Dis_Patient` RT `CD.Exam`⟩
⟨`ID.Dis_Patient` RT `ID.Test`⟩
⟨`ID.Dis_Patient` RT `ID.Medical_Staff`⟩
⟨`ID.Dis_Patient` RT `CD.Physician`⟩

A graphical representation of the Common Thesaurus for `Cardiology` and `Intensive Care` departments is reported in Fig. 8, where solid lines represent explicit/extracted relationships, dashed lines represent inferred relationships, and superscripts indicate the kind of terminological relationship.[5]

5 Affinity-based clustering of ODL$_{I^3}$ classes

To integrate the ODL$_{I^3}$ classes of the different sources into a global ODL$_{I^3}$ class, we employ hierarchical clustering techniques based on the concept of *affinity*. This way, we identify ODL$_{I^3}$ classes that describe the same or semantically related information in different source schemas.

This activity is performed with the support of the ARTEMIS tool environment. ARTEMIS has been conceived for a semi-automatic integration of heterogeneous structured databases [28]. In the context of MOMIS, the ARTEMIS affinity framework has been extended to be applied to the analysis of ODL$_{I^3}$ schema descriptions, including object patterns. In the following, we describe the extensions to the affinity-based clustering introduced in MOMIS to cope with object pattern and semistructured data integration.

ODL$_{I^3}$ classes are analyzed and compared by means of *affinity coefficients* which allow us to determine the level of similarity between classes in different source schemas. In particular,

[5]For the sake of simplicity, only relationships between class names are reported.

ARTEMIS evaluates a *Global Affinity* coefficient as the linear combination of a *Name Affinity* coefficient and a *Structural Affinity* coefficient, respectively.

Affinity coefficients for ODL_{I^3} classes are evaluated by exploiting terminological relationships in the Common Thesaurus. To this end, a strength $\sigma_{\Re}$ is assigned to each type of terminological relationship $\Re$ in the Common Thesaurus, with $\sigma_{\text{SYN}} \geq \sigma_{\text{BT/NT}} \geq \sigma_{\text{RT}}$. In the following, when necessary, we use notation $\sigma_{ij_{\Re}}$ to denote the strength of the terminological relationship $\Re$ for terms t_i and t_j in the Thesaurus; furthermore, we use $\sigma_{\text{SYN}} = 1$, $\sigma_{\text{BT}} = \sigma_{\text{NT}} = 0.8$ and $\sigma_{\text{RT}} = 0.5$.

Two terms have affinity if they are connected through a path in the Common Thesaurus. Their level of affinity depends on the length of the path, on the type of relationships involved in this path, and on their strengths. The affinity of two terms in the Thesaurus is 0 if a path does not exist between them in the Common Thesaurus and it is 1 if they coincide or are synonyms. In remaining cases, the longer the path defined between two terms, the lower their affinity. For a given path length, the higher the strength of the involved relationships, the greater the affinity of the involved terms.

Let c and c' be two classes belonging to sources S and S' respectively. Let us now define how the affinity coefficients are computed.

5.1 Name Affinity coefficient

The Name Affinity coefficient of two classes c and c', denoted $NA(c, c')$, is the measure of the affinity of their names n_c and $n_{c'}$, represented as terms in the Common Thesaurus (see Table 1).

Coefficient	Value	Condition
$NA(c, c')$	1	if $n_c = n_{c'}$
	$0 < \sigma_{12_{\Re}} \cdot \sigma_{23_{\Re}} \cdot \ldots \cdot \sigma_{m(m+1)_{\Re}} \leq 1$	if $n_c \rightarrow^m n_{c'}$ AND $NA(c, c') \geq \alpha$
	0	in all other cases

Legend:
$n_c \rightarrow^m n_{c'}$ denotes the path of length m, with $m \geq 1$, between n_c and $n_{c'}$ in the Common Thesaurus for which $NA(c, c')$ is the highest value.
α is a threshold used to select name having high affinity.

Tabella 1: Name Affinity coefficient

For any pairs of classes, $NA(c, c') \in [0, 1]$. We introduce a threshold α to select classes having high values of Name Affinity (e.g., $\alpha \in [0.4, 0.6]$).

Example 5 Consider the Common Thesaurus illustrated in Figure 8. A path exists between `CD.Physician` and `ID.Patient` in the Thesaurus.

$$\texttt{CD.Physician} \xrightarrow{\text{RT}} \texttt{CD.Patient} \xrightarrow{\text{SYN}} \texttt{ID.Patient}$$

Therefore, we have that $NA(\texttt{CD.Physician}, \texttt{ID.Patient}) = 0.5 \cdot 1 = 0.5$.

In the remainder of the paper symbol $\sim$ will be used to denote affinity between names.

5.2 Structural Affinity coefficient

The Structural Affinity coefficient of two classes c and c', denoted $SA(c, c')$, is the measure of the affinity of their attributes (see Table 2). The Structural Affinity coefficient is evaluated using Dice's function [29] (refined by a control factor F_c), which returns an affinity value in the range

Coefficient	Value	Condition
$SA(c, c')$	$\frac{2 \cdot \lvert\{(a_t, a_q) \mid a_t \in A(c), a_q \in A(c'), n_t \sim n_q\}\rvert}{\lvert A(c)\rvert + \lvert A(c')\rvert} \cdot F_c$	if $\lvert C\rvert \neq 0$
	0	if $\lvert C\rvert = 0$

Legend:
$C = \{(a_t, a_q) \mid a_t \in A(c), a_q \in A(c'), n_t \sim n_q\}$, where $A(c)$ and $A(c')$ are the sets of attributes in c and c', respectively
$F_c = \frac{\lvert\{x \in C \mid\ flag(x)=1\ \}\rvert}{\lvert C\rvert}$, where $flag(x) = 1$ stands for a positive result

Tabella 2: Structural Affinity coefficient

$[0, 1]$ proportional to the number of attributes whose names have affinity in the Thesaurus. The value 0 indicates the absence of attributes with affinity in the considered classes, while the value 1 indicates that all attributes defined in the two classes have affinity. The greater the number of attributes with affinity in the considered classes, and the greater the number of positive control results, the higher the $SA()$ value for the classes.
In general, given two classes, an attribute of one class may have affinity with more than one attribute of the other class. In the evaluation of the $SA()$ coefficient, we consider these multiple affinities as a single affinity between one attribute and a set of attributes.
In the $SA()$ formula, the control factor F_c evaluates the percentage of valid relationships between attributes obtained in the step Validation of relationships (step 3) illustrated in Section 4.

For the evaluation of Structural Affinity, *optional* attributes of ODL$_{I^3}$ classes representing object patterns must be considered properly. Depending on which attributes are taken into account, the following options are possible for the computation of the $SA()$ coefficient:

1. *All attribute-based.* With this option, *optional* attributes are treated as the other ones and are always taken into account when evaluating the affinity of ODL$_{I^3}$ classes describing object patterns.

2. *Common attribute-based.* With this option, *optional* attributes are not taken into account when evaluating the affinity.

3. *Threshold-based.* With this option, *optional* attributes are taken into account for affinity evaluation only if they are common to at least a certain number of objects of the considered object pattern.

The third option is difficult to apply, since it requires setting the value of a threshold, which can be dependent on the specific object pattern or on the source. As for the other two options, they have different implications on the affinity values produced. Given a pattern to be compared, the second option gives affinity values higher than the first one, in presence of the same number of attribute pairs with affinity. In fact, less attributes are considered at the denominator of the $SA()$ formula choosing option 2). On the other side, if most attributes of an object pattern are optional, then the option 1) is better for Structural Affinity evaluation. The choice between the first two options depends on the specific application under analysis. In our example, we applied both options and we discuss obtained values in the following, when presenting results of clustering.

Example 6 Consider classes `CD.Physician` and `ID.Patient`. By applying the all attribute-based option, we have that

$$SA(\texttt{CD.Physician}, \texttt{ID.Patient}) = \frac{2 \cdot 2}{4 + 6} = 0.4$$

due to the following affinities: `CD.name`$\sim$\{`ID.first_name`,`ID.last_name`\} and `CD.address`$\sim$`ID.address`.

5.3 Global Affinity coefficient

The Global Affinity coefficient of two classes c and c', denoted $GA(c, c')$, is the measure of their affinity computed as the weighted sum of the Name and Structural Affinity coefficients (see Table 3).

Coefficient	Value	Condition
$GA(c, c')$	$w_{NA} \cdot NA(c, c') + w_{SA} \cdot SA(c, c')$	in all cases

Legend:
w_{NA} and w_{SA}, with $w_{NA}, w_{SA} \in [0, 1]$ and $w_{NA} + w_{SA} = 1$, are introduced to assess the relevance of each coefficient in computing the global affinity value.

Tabella 3: Global Affinity coefficient

Weights in $GA(c, c')$ allow the analyst to differently stress the impact of each coefficient in the evaluation of the global affinity value.

Example 7 Let us return to the Name and Structural Affinity coefficients of Example 5, and Example 6, respectively. The Global Affinity coefficient of `C.Physician` and `ID.Patient` is computed as follows:

$$GA(\texttt{CD.Physician}, \texttt{ID.Patient}) = 0.5 \cdot 0.5 + 0.5 \cdot 0.4 = 0.45$$

using $w_{NA} = w_{SA} = 0.5$, since we consider both affinity coefficients equally relevant.

5.4 Considerations on the affinity evaluation process

The goodness of affinity evaluation relies on both the linguistic correctness of the terminological relationships and on the parameters (i.e., strengths, weights, thresholds) intervening in the calculation of the coefficients and their relative values. As it is intuitive, affinity has an element of subjectivity, due to the fact that the knowledge and experience of the designer can be required to integrate terminological relationships typical of the domain under analysis and to properly set parameter values. To make affinity even more objective, interactive functionalities are provided in MOMIS. In particular, the construction and validation of the Common Thesaurus in ODB-Tools is interactive, and the designer can supply additional terminological relationships typical of application domain under analysis. Affinity evaluation process is also interactive and weight-based in the ARTEMIS tool environment, to enable the designer to properly set the involved parameters and to validate the choices performed by the tool in all steps of the process. The ARTEMIS affinity approach has been experimented on different sets of conceptual database schemas to select a set of default values (i.e., the ones working satisfactorily in most cases) for setting the tool. The values of terminological relationship strengths in the Thesaurus, and of affinity weights and thresholds used in the examples of this paper correspond to these selected default values. Default values can however be dynamically varied by the designer when necessary, to tailor the the affinity calculation to the specific application. For a deeper discussion of the experimentation of the use of a dictionary of strengthened terminological relationships for affinity analysis of data schemas in the Italian Public Administration domain, the reader can refer to [26, 28]. More general considerations on usage and settings of semi-automatic, weight-based techniques for large-scale conceptual schema analysis and comparison can be found in [27].

5.5 Clustering of ODL$_{I^3}$ classes

To identify all the ODL$_{I^3}$ classes having high affinity in the considered source schemas, we employ a hierarchical clustering technique, which classifies classes into groups at different levels of affinity, forming a tree [36].

The hierarchical clustering procedure uses a matrix M of rank K where k the total number of ODL$_{I^3}$ classes to be analyzed. An entry $M[h,k]$ of the matrix represents the affinity coefficient $GA(c_h, c_k)$ between classes c_h and c_k.

Clustering is iterative and starts by placing each class in a cluster by itself. Then, at each iteration, the two clusters having the greatest $GA()$ value in M are merged. M is updated at each merging operation by deleting the rows and the columns corresponding to the merged clusters, and by inserting a new row and a new column for the newly defined cluster c_{hk}. The $GA()$ value between c_{hk} and each remaining cluster $\bar{c}$ in M is computed. The new value between c_{hk} and a remaining cluster is set to the maximum $GA()$ value between the $GA()$ values c_h and c_k had with $\bar{c}$ in the matrix. The procedure terminates when only one cluster is left and produces as the output a tree of ODL$_{I^3}$ classes, where intermediate nodes have an associated $GA()$ value and leaves are ODL$_{I^3}$ classes.

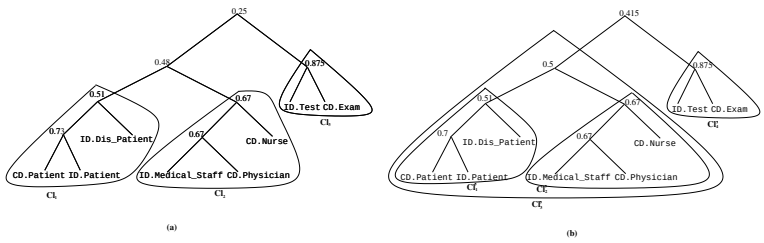

Figura 9: Affinity trees of `Cardiology` and `Intensive Care` departments

Figure 9 shows the affinity trees resulting from clustering our set of ODL$_{I^3}$ classes. In particular, Figure 9(a) illustrates the affinity tree obtained by applying affinity evaluation option 1. (i.e., all attribute-based). Figure 9(b) illustrates the affinity tree obtained by applying option 2. (i.e., common attribute-based).

Once the affinity tree has been constructed, the question is the selection of clusters to be integrated for the definition of the global schema. Cluster selection is interactive, based on the numerical affinity values in the affinity tree. In particular, ARTEMIS provides a threshold-based mechanism for cluster selection. The designer specifies a value for a threshold T and clusters characterized by a $GA()$ value greater than or equal to T are selected and proposed. High values of T return small, highly homogeneous clusters. By decreasing T's value, clusters with more classes can be selected. In the tool, the default value of T is set to 0.5. This default value can be refined dynamically, on the basis of the specific application under analysis.

With reference to our examples of Fig. 9, clusters have been selected using the default threshold value of 0.5. As the affinity trees show, the number of attributes considered during the Structural Affinity evaluation affects the number of clusters that are selected. In the first case three clusters are selected: $\mathbf{Cl_1}$, representing information about patients; $\mathbf{Cl_2}$ representing information about medical staff; and $\mathbf{Cl_3}$ representing information about medical exams. In the second case, clusters $\mathbf{Cl'_1}$ and $\mathbf{Cl'_2}$ are still selected, representing information about patients and medical staff, respectively. Moreover, an additional cluster has been formed and selected in the affinity tree, namely $\mathbf{Cl'_3}$, which merges both $\mathbf{Cl'_1}$ and $\mathbf{Cl'_2}$. This occurs because the `Patient` pattern includes some optional attributes. For this reason, by applying option 2., the Structural Affinity between the ODL$_{I_3}$ class representing the `Patient` pattern and the other ODL$_{I_3}$ classes

is higher than the Structural Affinity values obtained using option 1. (being the number of attribute pairs with affinity equal in both cases).

We believe that both these options are valid and different application domains can be found where the first one is preferable over the second one and vice versa. To keep ARTEMIS flexible and general-purpose, ODL$_{I^3}$ classes can be clustered according to both options (1. and 2.), and the affinity tree to be used for cluster selection is decided by the designer.

Cluster selection from an affinity tree is also interactive, because it is dependent on the kind of query that need to be supported at the global level, known by the designer. For example, if it is known that only queries regarding people in general will be posed at the global level, affinity tree in Fig. 9(b) is more suitable, and cluster $\mathbf{Cl'_3}$ is selected for the construction of the global schema. If more detailed queries are of interest for the considered domain making a distinction between patients and medical staff, this information must be kept separate at the global level. In this case, both affinity trees are suitable, since two distinct clusters are identified for patients and medical staff. In the following, we describe the construction of the global schema starting from clusters $\mathbf{Cl_1}$, $\mathbf{Cl_2}$, and $\mathbf{Cl_3}$ in the affinity tree of Fig. 9(a).

6 Construction of the mediator global schema

In this section, the process leading to the definition of the global schema of the mediator is described. The global schema is the mediator view of data stored in local sources. A global ODL$_{I^3}$ class, that is, *global_class*$_i$, is derived from a selected cluster of the affinity tree, to provide an integrated representation of all the classes contained in the cluster.

The generation of the *global_class*$_i$ is interactive with the designer and is performed with the Global Schema Builder (GSB) module support. Let $\texttt{Cl}_\texttt{i}$ be a selected cluster in the affinity tree. First, the GSB module of MOMIS associates with the *global_class*$_i$a set A of *global attributes*, corresponding to the union of the attributes of the classes belonging to $\texttt{Cl}_\texttt{i}$. Attributes with affinity are unified into a unique global attribute in *global_class*$_i$. The attribute unification process is performed automatically for what concerns the names of attributes with affinity, according to the following rules:

- for attributes that have name affinity due to SYN relationships, one term is selected (with the help of the Common Thesaurus) and becomes the name to the corresponding global attribute in *global_class*$_i$;
- for attributes that have name affinity due to BT and NT relationships, a broader term for all of them is selected from the Common Thesaurus and becomes the name to the corresponding global attribute in *global_class*$_i$.

For example, the following set of global attributes is obtained as the output of the attribute unification process for cluster $\texttt{Cl}_1$ of Fig. 9:

A = (`name, code, address, exam*, room, bed, therapy*, date, note, physician*`)

Additional information has to be provided for completing the global ODL$_{I^3}$ classes definition. In particular, MOMIS asks the designer to interactively supply information regarding:

1. the *global_class*$_i$ name
   the tool proposes a set of candidate names for the global class, by exploiting the terminological relationships defined in the Common Thesaurus for the class names in $\texttt{Cl}_\texttt{i}$. Based on these candidates, the designer can decide the most suitable name to be assigned to

global_class$_i$. A different name can however be assigned, if retrieved candidates are not suitable. In our example, the designer decides to assign name `Hospital_Patient` to the global class defined in correspondence of cluster $\texttt{Cl}_1$.

Therefore, with respect to $\texttt{Cl}_1$ we obtain the global class:

Hospital_Patient = (`Hospital_Patient`,{`name`,`code`,`address`,`exam*`,`room`, `bed`,`therapy*`,`date`,`note`,`physician*`})

2. *mappings* between the global attributes of *global_class*$_i$ and the corresponding attributes of the classes in $\texttt{Cl}_\texttt{i}$

   In particular, if a global attribute is obtained from more than one attribute of the same class in the $\texttt{Cl}_\texttt{i}$, the designer is asked to specify the type of *correspondence* to be set for the global attribute. In MOMIS, we provide the following options:

   - *and* correspondence.
     This specifies that a global attribute corresponds to the concatenation of the attributes of a class $c_h \in \texttt{Cl}_\texttt{i}$.
     For example, the global attribute `name` of `Hospital_Patient` corresponds to both `first_name` and `last_name` attributes of `ID.Patient` in $\texttt{Cl}_1$. By specifying the *and* correspondence for `name`, the designer states that the values of both `first_name` and `last_name` attributes have to be considered when `ID.Patient` is involved.
   - *union* correspondence.
     This specifies that a global attribute corresponds to at most one of the specified attributes of a class $c_h \in \texttt{Cl}_\texttt{i}$;

3. *Default values* to be assigned to global attributes in correspondence of a class $c_h \in \texttt{Cl}_\texttt{i}$. A default value is always verified for the global attribute when evaluated on class c_h.

4. *New attributes* for the global class.

   For example, a new attribute `dept` can be added to the global class `Hospital_Patient`. This attribute is used to maintain information related to the department where a patient is hospitalized.

An example of ODL$_{I^3}$ specification for the global class `Hospital_Patient` is shown in Fig. 10. As we can see from this figure, for each attribute of `Hospital_Patient`, mapping rules are defined specifying: i) information on how to map the attribute on the corresponding attributes of the ODL$_{I^3}$ classes in the associated cluster, and ii) possible default/null values defined for the attribute in the cluster classes. For example, for the global attribute `name`, the mapping rule specifies the attributes that have to be considered in each class of the cluster $\texttt{Cl}_1$. In this case, an *and* correspondence is defined for `name` for the class `ID.Patient`. A mapping rule is defined for the global attribute `dept` to specify the value to be associated with `dept` for the instances of `CD.Patient`, `ID.Patient`, and `ID.Dis_Patient`.

As stated above, an *union* correspondance is useful when a global attribute is suitable for two or more local attributes of a source, depending on the value of another local attribute, say *tag attribute*. For example, let us suppose to have a cluster describing an `automobile` class and a local source storing the prices of cars both in Italian Lire and US Dollars, with `country` as the tag attribute.

The designer may define an *union* correspondence between the attributes `Italian_price` and `US_price` by introducing a rule specifying the local mapping based on the value of the `country` attribute. By using the mapping rule sintax, the declaration is as follows:

```
interface Hospital_Patient
{ attribute name
     mapping_rule (ID.Patient.first_name and
                   ID.Patient.last_name),
                   CD.Patient.name;
     ...
  attribute physician
     mapping_rule CD.Patient.physician,
                   Id.Patient.doctor_id
  attribute dept
     mapping_rule CD.Patient = 'Cardiology',
                   ID.Patient = 'Intensive Care',
                   ID.Dis_Patient = 'Intensive Care'
}
```

Figura 10: Example of global class specification in ODL_{I^3}

```
...
 attribute integer price
    mapping rule (S.car.Italian_price union
                   S.car.US_price on Rule1),
                  ...
...
 rule Rule1 { case of S.car.country:
              ''Italy'' : S.car.Italian_price;
              ''US''    : S.car.US_price; }
```

As we can see, no specification is given for the domain of a global attribute. At present, in fact, query issued at MOMIS returns a set of instances of different formats depending on the origin sources.
Our aim for future work, as suggested in [39, 35], is to add other predefined mapping function coming to the development of an open library containing the main functionalities that the designer can utilize, customize, and enrich with new operations.

The specification in ODL_{I^3} of a global class $global_class_i$ is thus translated into a persistent mapping table. As an example, the mapping table for the `Hospital_Patient` class is shown in Fig. 11.

Hospital_Patient	**code**	**name**	**exam**	**...**	**physician**	**dept**
CD.Patient	null	name	exam	...	physician	'Cardiology'
ID.Patient	code	first_name *and* last_name	test	...	doctor_id	'Intensive Care'
ID.Dis_Patient	code	null	null	...	ID.Patient. doctor_id	'Intensive Care'

Figura 11: `Hospital_Patient` mapping table

7 Query Processing and Optimization

In this section, we briefly describe how the information on the global schema are exploited for global query processing. When the user submits a query to the MOMIS system, the MOMIS Query Manager (QM) produces a set of subqueries that will be sent to each involved information source.

According to other semantic approaches [4, 5], this process consists of two main phases:

- semantic optimization
- query plan formulation

7.1 Semantic Optimization

In this phase, QM processes the given query by exploiting the semantic optimization techniques supported by ODB-Tools [7, 8, 9, 10], in order to reduce the query access plan cost. The query is replaced by a new one that incorporates any possible restriction which is not present in the original query but is logically implied by the query on the global schema. The transformation is based on logical inferences from content knowledge (in particular integrity constraints rules) of the mediator global schema, shared among the classes belonging to the cluster.

Let us consider, as an example, the query: "Retrieve the names of the patients with an exam result of 'Heart risk' ".

```
Q1:
select name
from   Hospital_Patient
where  exam.result = 'Heart risk'
```

Let us suppose that in our `Hospital` domain there exist a relationship between the result of the exam and the department of the patient, for example, that all the patients with an exam result of 'Heart risk' are kept at 'Cardiology'. Thus the designer can define the following integrity constraint rule:

```
rule R1 forall X in Hospital_Patient:  (X.exam.result = 'Heart risk') then X.dept
= ''Cardiology';
```

The QM, using the query optimizer of ODB-Tools, executes the semantic expansion of the query by applying rule `R1`. The resulting query is the following:

```
Q1:
select name
from   Hospital_Patient
where  exam.result = 'Heart risk'
and    dept = 'Cardiology'
```

Semantic expansion is performed in order to add boolean factors in the "where clause": this process makes query plan formulation more expensive (because a heavier query has to be translated for each interesting source) but single sources' query processing overhead can be lighter in case secondary indexes on added predicates exist in the involved sources. Furthermore, the introduction of a boolean factor can be useful for query plan formulation as it is in our 'Heart risk' example.

Check algorithm
/* **Input:** Cluster Cl_j with its k classes */

begin procedure:
for each class $c_h \in Cl_j$ **do**
for each boolean factor **do**
if (global attribute $\in$ boolean factor) corresponds to:
1. null value in c_h: no local query is generated on c_h;
2. default value in c_h: **if** the value specified in the boolean factor differs from the default one **then** no local query is generated on c_h;

end procedure.

Figura 12: Check algorithm

7.2 Query Plan Formulation

Once the Query Manager has produced the optimized query, a set of subqueries for the local information sources will be generated. For each information source, by using the mapping table associated with each global class, the Query Manager has to express the query in terms of local schemas. In order to obtain each local query, the QM checks and translates every boolean factor in the where clause. In particular, a *Check* algorithm (shown in Fig. 12) is used to determine the local queries to be generated (starting from the cluster associated with the global class queried by the user).
Referring to our example, step 1 of the *Check* algorithm will exclude the `Dis_Patient` class and step 2 the `Patient` class of the `Intensive_Care_Department` source, so that we derive only the following query for the `Cardiology_Department` source:

```
select R.name
from   Patient R
where  exists X in R.exam : X.outcome = 'Heart risk'
```

In such a way, an effective optimization is performed because only one local source (and not two) have to be accessed.
If integrity constraints are provided also for a local source, it is possible to perform semantic query optimization with ODB-Tools on the corresponding local query too, before sending the query to the source. For example, if the following rule is provided for the `Cardiology_Department` source:

```
rule R2 forall X in Patient:  (exists X.exam.outcome = 'Heart risk') then X.room
> 20 ;
```

`R2` can be used to optimize the local query, giving raise to the following query:

```
select R.name
from   Patient R
where  exists X in R.exam : X.outcome = 'Heart risk'
and    R.room > 20
```

This transformation could be useful if an index is available for the `room` attribute.

As another example of query formulation, consider the case of heterogeneous address representation and suppose we want to find the names of all the patients with a given address. We issue the `Q2` query:

```
Q2: select name
from Hospital_Patient
where address.street = 'Army Street'
```

Supposing that no integrity constraint rule is given, the processing of query `Q2` would result in the generation of the following three queries: two queries for the `Cardiology` department (as we have to take into account the presence of the union constructor on address) and one query for the `Intensive Care` department.

```
CD:                                    CD:
select R.name                          select R.name
from   Patient R                       from   Patient R
where  R.address = 'Army Street'       where  R.address.street = 'Army Street'

ID:
select P.first_name, P.last_name
from   Patient P
where  P.address = 'Army Street'
```

8 Related work and discussion

Works related to the issues discussed in this paper are in the area of semistructured data and of heterogeneous information integration.

Semistructured data. The issue of modeling semistructured data has been investigated in the literature. In particular, a survey of problems concerning semistructured data modeling and querying is presented in [20]. Two similar models for semistructured data have been proposed [22, 52], based on rooted, labeled graph with the objects as nodes and labels on edges. According to the model presented in [22], information resides at labels only, while according to the "Object Exchange Model" (OEM) proposed by Papakonstantinou et. al. in [52], information also resides at nodes. In the first one, edges are labeled both with the usual base types of data such as integer, real, string, and with names (called *symbols*) that correspond to the attribute names in a relation or class name. In the second one, we have that *i)* leaf nodes are labeled with data; *ii)* internal nodes are not labeled with meaningful data; and *iii)* edges are labeled only with symbols. It is possible to define mappings between the two models: by introducing new edges an OEM graph can be converted into the model proposed in [22].

The issue of adding structure to semistructured data, which is more directly concerned with our concept of object pattern, has also been investigated. In particular, in [43], the notion of *dataguide* has been proposed as a "loose description of the structure of the data" actually stored in an information source. A proposal to infer structure in semistructured data has been presented in [51], where the authors use a graph-based data model derived from [22, 52]. The data structure, i.e. a type hierarchy for semistructured data model, is obtained by an algorithm which measures the *jump* between types, based on the relative importance of some attributes w.r.t. the others in a large set of data. In this way, a "reasonably small approximation" of the typing of large and irregular data collection is built, by distinguish the "core" attributes from the more circumstantial ones. In [23], a new notion of a graph schema appropriate for rooted, labeled graph databases has been proposed. The main usages of the structure extracted from a semistructured source have been presented for query optimization. In fact, the existence of a path in the structure simplifies query evaluation by limiting the query only to data that are relevant. In this paper, we are more concerned with usage of the structure in form of object patterns to support the integration of semistructured sources with structured databases.

Heterogeneous information integration. In this area, many projects based on a mediator

architecture have been developed [4, 5, 18, 32, 34, 41, 42].
MOMIS is based on a mediator architecture and follows the 'semantic approach'.

Following the classification of integration system proposed by Hull [44], MOMIS is in the line of the "virtual approach". Virtual approach was first proposed in multidatabase models in the early 80s [49, 55]. More recently, systems have been developed based on the use of description logics [3, 47] such as CLASSIC [17]. All of the virtual approaches are based on a model of query decomposition, sending subqueries to source databases, and merging the answers that come back. Recent systems based on description logics are focused primarily on conjunctive queries (i.e., expressible using select, project and join), and have more the flavor of the *Open World Assumption* - the answer provided through an integrated view will hold a subset of the complete answer that is implied by the underlying databases. For the schema, a "top-down" approach is used: in essence a global schema encompassing all relevant information is created, and data held in the source databases is expressed as views over this global schema [56].

With references to the same classification proposed by Hull, MOMIS is in the category of "read-only views", i.e. systems whose task is to support an integrated, read-only, view of data that resides in multiple databases. The most similar projects are: GARLIC, SIMS, Information Manifold and Infomaster.

The GARLIC project [41, 42] builds up on a complex wrapper architecture to describe the local sources with an OO language (GDL), and on the definition of Garlic Complex Objects to manually unify the local sources to define a global schema.
The SIMS project [3, 5] proposes to create a global schema definition exploiting the use of description logics (i.e. the LOOM language) for describing information sources. The use of a global schema allows both GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them.

Information Manifold Systems [45, 47], as the MOMIS project, provides a source independent, query independent mediator. The input schema of an Information Manifold System is a set of descriptions of the sources; so, given a query, the system will create a plan for answering the query using the source. The algorithms to decide the useful information sources and to generate the query plan are provided [47, 48]. With respect to the input schema generation, it is completely modeled by the user, while in our approach it is system-guided.

Infomaster System [39, 35] provides integrated access to multiple distributed heterogenuos information sources giving the illusion of a centralized, homogeneous information system. It is based on a global schema, completely modeled by the user, and a core system that dynamically determines an efficient plan to answer the user's queries by using translation rules that harmonizing heterogeneous sources.

On the other hand, many projects are based on a 'structural' approach [18, 32, 34]. The TSIMMIS project [32, 38] follows a 'structural' approach and uses a self-describing model (OEM) to represent the data objects and pattern matching techniques to perform a predefined set of queries based on a query template. The semantic knowledge is effectively encoded in the MSL (Mediator Specification Language) rules enforcing source integration at the mediator level. Although the generality and conciseness of OEM and MSL make this approach a good candidate for the integration of widely heterogeneous and semistructured information sources, a major drawback in such an approach is that dynamically adding sources is an expensive task. In fact, new TSIMMIS sources not only must be wrapped, but the mediators that uses them have to be redefined and their MSL definitions recompiled. The administrator of the system must figure out weather and how to use the new sources [56].

Another research area is focused on the identification and resolution of the semantic inconsistencies between schemas by exploiting the definition of correspondence between schema elements. Some approaches rely on terminological analysis of schema elements. In [19], a

Summary Schema Model (SSM) has been proposed to provide support to identify semantically similar entities. According to an existing taxonomy, the SSM is used as an abstract and concise description of all data available in the sources. Two entities are semantically related if they are mapped to the same concept of the taxonomy.

In [37], the main purpose consists in defining techniques based on fuzzy logic to derive terminological relationships between schema elements.

Other approaches consider also attributes and domains of schema elements [16, 33]. In [16], relations, attributes, and integrity constraints in source schema are properly expressed using logics-based languages to unify knowledge of all the elements in the sources.

Finally, some researchers have proposed to consider federated architectures for resolution of semantic heterogeneity, where notion of federated schema is given [53, 54].

9 Conclusions and future work

In this paper, we have presented an intelligent approach to schema integration for heterogeneous information sources. It is a semantic approach based on a Description Logics component (ODB-Tools engine) and a cluster generator module, ARTEMIS, together with a minimal ODL_{I^3} interface module. Generation of the global schema for the mediator is a semi-automated process. An ODL_{I^3} language is available and basic functionalities related to Common Thesaurus construction and clustering have been implemented. We are developing on unification functionalities in the framework of the architecture presented in Section 2 and on the design and implementation of visual interfaces supporting designer interaction with the system. Future research work will be devoted to a deeper study of query processing and optimization processes, and to the development of corresponding Query Manager functionalities.

Riferimenti bibliografici

[1] S. Abiteboul and P. Kanellakis, "Object Identity as a Query Language Primitive", *SIGMOD*, 159–173, ACM Press , 1989.

[2] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener, "The Lorel Query Language for Semistructured Data", *Journal of Digital Libraries*, 1(1), November 1996.

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

[4] Y. Arens, C.Y. Chee, C. Hsu, and C. A. Knoblock, "Retrieving and Integrating Data from Multiple Information Sources," in *International Journal of Intelligent and Cooperative Information Systems. Vol. 2, No. 2. Pp. 127-158*, 1993.

[5] Y. Arens, C. A. Knoblock and C. Hsu, "Query Processing in the SIMS Information Mediator," in *Advanced Planning Technology, editor, Austin Tate, AAAI Press, Menlo Park, CA*, 1996.

[6] , ARPA, "ARPA I^3 Reference Architecture", Available at http://www.isse.gmu.edu/I3_Arch/index.html

[7] D. Beneventano, S. Bergamaschi and C. Sartori "Semantic Query Optimization by Subsumption in OODB," in *Proc. of the Int. Workshop on Flexible Query-Answering Systems (FQAS96)*, Roskilde, Denmark, may 1996, pp. 167-185.

[8] D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini, "ODB-Tools: A Description Logics Based Tool for Schema Validation and Semantic Query Optimization in Object Oriented Databases", in *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, April 1997.

[9] D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini, "ODB-QOptimizer: a Tool for Semantic Query Optimization in OODB", in *Proc. of Fifth Conference of the Italian Association for Artificial Intelligence (AI*IA97)*, Rome 1997.

[10] http://sparc20.dsi.unimo.it/index.html

[11] D. Beneventano, S. Bergamaschi, S. Lodi and C. Sartori, "Consistency Checking in Complex Object Database Schemata with Integrity Constraints", in *IEEE Transactions on Knowledge and Data Engineering*, Vol. 10, pag. 576-598, 1998.

[12] S. Bergamaschi, S. Castano, S. De Capitani di Vimercati, S. Montanari, M. Vincini, "An Intelligent Approach to Information Integration," in *International Conference on Formal Ontology in Information Systems (FOIS'98)*, Trento, Italy, June 1998.

[13] S. Bergamaschi and B. Nebel, "Acquisition and Validation of Complex Object Database Schemata Supporting Multiple Inheritance," *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks and Complex Problem Solving Technologies*, 4:185–203, 1994.

[14] S. Bergamaschi and C. Sartori, "On Taxonomic Reasoning in Conceptual Design," in *ACM Transactions on database Systems*, 17(3):385–422, September 1992.

[15] E. Bertino, M. Negri, G. Pelagatti and L. Sbattella, "Object-Oriented Query Languages: The Notion and the Issues", in *IEEE Trans. Knowl. and Data Engineering*, vol. 4, num. 3, pag. 223-236, 1992.

[16] J.M. Blanco, A. Illarramendi, A. Goni, "Building a Federated Relational Database System: An Approach Using a Knowledge-Based System," *Int. Journal of Intelligent and Cooperative Information Systems*, Vol.3, No.4, 1994, pp.415-455.

[17] A. Borgida, R.J. Brachman, D.L. McGuinness and L.A. Resnik, "CLASSIC: A Structural Data Model for Objects," in *SIGMOD*, pages 58-67, Portland, Oregon, 1989.

[18] Y.J. Breibart et al. "Database Integration in a Distributed Heterogeneous Database System," in *Proc. 2nd Intl IEEE Conf.on Data Engineering, Los Angeles, CA*, 1986.

[19] M.W. Bright, A.R. Hurson, S. Pakzad, "Automated Resolution of Semantic Heterogeneity in Multidatabases", *ACM Transactions on Database Systems,* Vol.19, No.2, June 1994, pp.212-253.

[20] P. Buneman, "Semistructured Data," in *Proc. of 1997 Symposium on Principles of Database Systems (PODS97)*, Tucson, Arizona, May 1997, pp. 117-121.

[21] P. Buneman, L. Raschid, J. Ullman, "Mediator Languages - a Proposal for a Standard", Report of an I^3/POB working group held at the University of Maryland, April 1996. *ftp://ftp.umiacs.umd.edu/pub/ONRrept/medmodel96.ps.*

[22] P. Buneman, S. Davidson, G. Hillebrand, and D. Suciu "A Query Language and Optimization Techniques for Unstructured Data", in *Proc. of the ACM SIGMOD International Conference*, Montreal, Canada, June 1996, pp. 505-516.

[23] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu "Adding Structure to Unstructured Data", in *Proc. of ICDT*, Delphi, Greece, January 1997, pp. 336-350.

[24] D. Calvanese, G. De Giacomo and M. Lenzerini, "Structured Objects: Modeling and Reasoning", *in Proc. of Int. Conference on Deductive and Object-Oriented Databases*, 1995.

[25] S. Castano, V. De Antonellis, "Deriving Global Conceptual Views from Multiple Information Sources," invited paper, in *preProc. of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspectives and Future Directions*, Los Angeles, November 1997.

[26] S. Castano, V. De Antonellis, "Semantic Dictionary Design for Database Interoperability", in *Proc. of Int. Conf. on Data Engineering, ICDE'97*, Birmingham, UK, April 1997.

[27] S. Castano, V. De Antonellis, M.G. Fugini, B. Pernici, "Conceptual Schema Analysis: Techniques and Applications", accepted for publication on *ACM Transactions on Database Systems*, 1997.

[28] S. Castano, V. De Antonellis, S. De Capitani Di Vimercati, "Global Viewing of Heterogeneous Data Sources", *Internal Report N. 98-08*, Dip. Elettronica e Informazione, Politecnico di Milano, 1998, (submitted for publication).

[29] S. Castano, V. De Antonellis, "Engineering a Library of Reusable Conceptual Components", *Information and Software Technology*, 39, 1997, pp.65-76.

[30] T. Catarci and M. Lenzerini, "Representing and using interschema knowledge in cooperative information systems", *Journal of Intelligent and Cooperative Information Systems*, 2(4), 375-398, 1993.

[31] R. Cattel (ed.), *The Object Database Standard: ODMG-93*, Morgan Kaufmann, 1996.

[32] S. Chawathe, H. Garcia Molina, J. Hammer, K. Ireland, Y. Papakostantinou, J.Ullman, and J.Widom, "The TSIMMIS project: Integration of Heterogeneous Information Sources," *in IPSJ Conference, Tokyo, Japan*, 1994. ftp://db.stanford.edu/pub/chawathe/1994/tsimmis-overview.ps.

[33] C. Clifton, E. Housman, A. Rosenthal, "Experience with a Combined Approach to Attribute-Matching Across Heterogeneous Databases," in *IFIP DS-7 Data Semantics Conf.*, Switzerland, 1997

[34] U. Dayal and H. Hwuang, "View Definition and Generalization for Database Integration in a Multidatabase System," In *Proc. IEEE Workshop on Object-Oriented DBMS - Asilomar, CA*, 1986.

[35] O. M. Duschka and M. R. Genesereth, "Infomaster - An Information Integration Tool", in *proceedings of the International Workshop "Intelligent Information Integration" during the 21st German Annual Conference on Artificial Intelligence, KI-97.* Freiburg, Germany, September 1997.

[36] B. Everitt, *Cluster Analysis*, Heinemann Educational Books Ltd, Social Science Research Council, 1974.

[37] P. Fankhauser, M. Kracker, E.J. Neuhold, "Semantic vs. Structural Resemblance of Classes," *SIGMOD RECORD*, Vol.20, No.4, December 1991, pp.59-63.

[38] H. Garcia-Molina et al., "The TSIMMIS Approach to Mediation: Data Models and Languages", in *NGITS workshop*, 1995.
ftp://db.stanford.edu/pub/garcia/1995/tsimmis-models- languages.ps.

[39] M. R. Genesereth, A. M. Keller and O. Duschka, "Infomaster: An Information Integration System", in *proceedings of 1997 ACM SIGMOD Conference*, May 1997.

[40] J. Gilarranz, J. Gonzalo, F. Verdejo, "Using the EuroWordNet Multilingual Semantic Database", in *Proc. of AAAI-96 Spring Symposium Cross-Language Text and Speech Retrieval*, 1996.

[41] M.J.Carey, L.M. Haas, P.M. Schwarz, M. Arya, W.F. Cody, R. Fagin, M. Flickner, A.W. Luniewski, W. Niblack, D. Petkovic, J. Thomas, J.H. Williams and E.L. Wimmers, "Towards Multimedia Information System: The Garlic Approach", *IBM Almaden Research Center*, San Jose, 1994.

[42] M.T. Roth, P. Scharz, "Don't Scrap It, Wrap it! A Wrapper Architecture for Legacy Data Sources", in *Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, 1997.

[43] R. Goldman and J. Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Data", in *Proc. of the Twenty-Third International Conference on very Large Data Bases*, 1997.

[44] R. Hull. Managing Semantic Heterogeneity in Databases: A Theoretical Perspective. *ACM Symp. on Principles of Database Systems*, pages 51-61, 1997.

[45] T. Kirk, A. Y. Levy, Y. Sagiv and D. Srivastava, "The Information Manifold", *In Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous*, Distributed Environments, 1995.

[46] C. Lecluse and P. Richard, "Modelling Complex Structures in Object-Oriented Databases", in *Proc. of Symp. on Principles of Database Systems*, p.362-369, 1989.

[47] A. Y. Levy, A. Rajaraman and J. J. Ordille, "Querying Heterogeneous Information Sources Using Source Descriptions", *Proc. of VLDB 1996*, 251-262.

[48] A. Y. Levy, A. Rajaraman and J. J. Ordille, "Query-Answering Algorithms for Information Agents", in *AAAI/IAAI*, Vol. 1 1996: 40-47.

[49] W. Litwin, L. Mark, and N. Roussopoulos. Interoperability of multiple autonomous databases. *ACM Computing Surveys*, 22:267-293, 1990.

[50] A.G. Miller, "WordNet: A Lexical Database for English", *Communications of the ACM*, Vol. 38, No.11, November 1995, pp. 39 - 41.

[51] S. Nestorov, S. Abiteboul and R. Motwani, "Inferring Structure in Semistructured Data", SIGMOD Record 26(4): 39-43 (1997)

[52] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, "Object Exchange Across Heterogeneous Information Sources", in *Proc. of ICDE*, Taipei, Taiwan, March 1995, pp. 251-260.

[53] M.P. Reddy, B.E. Prasad, P.G. Reddy, A. Gupta, "A Methodology for Integration of Heterogeneous Databases," *IEEE Trans. on Knowledge and Data Engineering,* Vol.6, No.6, December 1994, pp.920-933.

[54] A.P. Sheth and J.P. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys,* Vol. 22, No. 3, September 1990.

[55] Thomas et al. Heterogeneous distributed database systems for production use. *ACM Computing Surveys*, 22(3):237-266, September 1990.

[56] J. D. Ullman. Information integration using logical views. *Intl. Conf on Database Theory*, pages 19-40, 1997.

[57] W.A. Woods and J.G. Schmolze, "The kl-one family", in *F.W. Lehman, editor*, published as a Special issue of *Computers & Mathematics with Applications*, Volume 23, Number 2-9.

A The ODL$_{I^3}$ description language

The following is a BNF description for the ODL$_{I^3}$ description language.
We included the syntax fragments which differ from the original ODL grammar, referring to this one for the remainder.

⟨interface_dcl⟩ ::= ⟨interface_header⟩ **{**[⟨interface_body⟩]
[**union** ⟨interface_body⟩]**};**
⟨interface_header⟩ ::= **interface** ⟨identifier⟩
[⟨inheritance_spec⟩]
[⟨type_property_list⟩]
⟨inheritance_spec⟩ ::= : ⟨scoped_name⟩ [,⟨inheritance_spec⟩]

Local schema pattern definition: the wrapper must indicate the kind and the name of the source of each pattern.

⟨type_property_list⟩ ::= **(** [⟨source_spec⟩] [⟨extent_spec⟩] [⟨key_spec⟩] [⟨f_key_spec⟩] **)**
⟨source_spec⟩ ::= **source** ⟨source_type⟩ ⟨source_name⟩
⟨source_type⟩ ::= **relational** | **nfrelational** | **object** | **file** | **semistructured**
⟨source_name⟩ ::= ⟨identifier⟩
⟨extent_spec⟩ ::= **extent** ⟨extent_list⟩
⟨extent_list⟩ ::= ⟨string⟩ | ⟨string⟩,⟨extent_list⟩
⟨key_spec⟩ ::= **key**[s] ⟨key_list⟩
⟨f_key_spec⟩ ::= **foreign_key** (⟨f_key_list⟩) references ⟨identifier⟩[,⟨f_key_spec⟩]
...

Global pattern definition rule, used to map the attributes between the global definition and the corrisponding ones in the local sources.

⟨attr_dcl⟩ ::= **[readonly] attribute**
[⟨domain_type⟩] ⟨attribute_name⟩ **[*]**
[⟨fixed_array_size⟩] [⟨mapping_rule_dcl⟩]
⟨mapping_rule_dcl⟩ ::= **mapping_rule** ⟨rule_list⟩
⟨rule_list⟩ ::= ⟨rule⟩ | ⟨rule⟩,⟨rule_list⟩
⟨rule⟩ ::= ⟨local_attr_name⟩ | '⟨identifier⟩'
⟨and_expression⟩ | ⟨union_expression⟩
⟨and_expression⟩ ::= **(** ⟨local_attr_name⟩ **and** ⟨and_list⟩ **)**
⟨and_list⟩ ::= ⟨local_attr_name⟩ | ⟨local_attr_name⟩ **and** ⟨and_list⟩
⟨union_expression⟩ ::= **(** ⟨local_attr_name⟩ **union** ⟨union_list⟩ **on** ⟨identifier⟩ **)**
⟨union_list⟩ ::= ⟨local_attr_name⟩ | ⟨local_attr_name⟩ **union** ⟨union_list⟩
⟨local_attr_name⟩ ::= ⟨source_name⟩.⟨class_name⟩.⟨attribute_name⟩
...

Terminological relationships used to define the Common Thesaurus.

⟨relationships_list⟩ ::= ⟨relationship_dcl⟩**;** | ⟨relationship_dcl⟩**;** ⟨relationships_list⟩
⟨relationships_dcl⟩ ::= ⟨local_attr_name⟩ ⟨relationship_type⟩ ⟨local_attr_name⟩
⟨relationship_type⟩ ::= **SYN** | **BT** | **NT** | **RT**
...

OLCD integrity constraint definition: declaration of rule (using *if then* definition) valid for each instance of the data; mapping rule specification (*or* and *union* specification rule).

⟨rule_list⟩ ::= ⟨rule_dcl⟩**;** | ⟨rule_dcl⟩**;** ⟨rule_list⟩
⟨rule_dcl⟩ ::= **rule** ⟨identifier⟩ ⟨rule_spec⟩
⟨rule_spec⟩ ::= ⟨rule_pre⟩ **then** ⟨rule_post⟩ | **{** ⟨case_dcl⟩ **}**
⟨rule_pre⟩ ::= ⟨forall⟩ ⟨identifier⟩ **in** ⟨identifier⟩ **:** ⟨rule_body_list⟩
⟨rule_post⟩ ::= ⟨rule_body_list⟩
⟨case_dcl⟩ ::= **case of** ⟨identifier⟩ **:** ⟨case_list⟩
⟨case_list⟩ ::= ⟨case_spec⟩ | ⟨case_spec⟩ ⟨case_list⟩
⟨case_spec⟩ ::= ⟨identifier⟩ **:** ⟨identifier⟩ **;**
⟨rule_body_list⟩ ::= **(** ⟨rule_body_list⟩ **)** | ⟨rule_body⟩ |
⟨rule_body_list⟩ **and** ⟨rule_body⟩ |
⟨rule_body_list⟩ **and (** ⟨rule_body_list⟩ **)**
⟨rule_body⟩ ::= ⟨dotted_name⟩ ⟨rule_const_op⟩ ⟨literal_value⟩ |
⟨dotted_name⟩ ⟨rule_const_op⟩ ⟨rule_cast⟩ ⟨literal_value⟩ |
⟨dotted_name⟩ **in** ⟨dotted_name⟩ |
⟨forall⟩ ⟨identifier⟩ **in** ⟨dotted_name⟩ **:** ⟨rule_body_list⟩ |
exists ⟨identifier⟩ **in** ⟨dotted_name⟩ **:** ⟨rule_body_list⟩
⟨rule_const_op⟩ ::= = | ≥ | ≤ | > | <
⟨rule_cast⟩ ::= **(**⟨simple_type_spec⟩**)**
⟨dotted_name⟩ ::= ⟨identifier⟩ | ⟨identifier⟩.⟨dotted_name⟩
⟨forall⟩ ::= **for all** | **forall**

B ODL$_{I^3}$ sources descriptions

```
Cardiology_Department (CD):

interface Patient                              interface Physician
 ( source semistructured                        ( source semistructured
      Cardiology_Department )                        Cardiology_Department )
{  attribute string         name;              {  attribute string  name;
   attribute string         address;              attribute string  address;
   attribute set<Exam>      exam*;                attribute integer phone;
   attribute integer        room;                 attribute string specialization;};
   attribute integer        bed;
   attribute string         therapy*;
   attribute set<Physician> physician*; };


interface Nurse                                 interface Exam
 ( source semistructured                        ( source semistructured
      Cardiology_Department )                        Cardiology_Department )
{  attribute string       name;                 {  attribute integer date;
```

```
   attribute string       address;            attribute string  type;
   attribute integer      level;              attribute string  outcome; };
   attribute set<Patient> patient; };
```

Intensive_Care_Department (ID):

```
interface Patient
( source relational Intensive_Care
  extent Patients
  key  code
  foreign_key(test) references Test,
  foreign_key(doctor_id) references
  Medical_Staff )
{ attribute string  code;
  attribute string  first_name;
  attribute string  last_name;
  attribute string  address;
  attribute string  test;
  attribute string  doctor_id; };
```

```
interface Medical_Staff
( source relational Intensive_Care
  extent Medical_Staffers
  key id )
{ attribute string  id;
  attribute string  first_name;
  attribute string  last_name;
  attribute integer phone;
  attribute string  address;
  attribute string  availability;
  attribute string  position; };
```

```
interface Test
( source relational Intensive_Care
  extent Tests
  key number )
{ attribute integer number;
  attribute string  type;
  attribute integer date;
  attribute string  laboratory;
  attribute string  result; };
```

```
interface Dis_Patient
( source relational Intensive_Care
  extent Dis_Patients
  key code
  foreign_key(code) references Patient)
{ attribute string  code;
  attribute integer date;
  attribute string  note; };
```