SI-Designer: a tool for intelligent integration of information

D. Beneventano[1,2], *S. Bergamaschi*[1,2], *I. Benetti*[1], *A. Corni*[1,2], *F. Guerra*[1] *and G. Malvezzi*[1]

(1) Università di Modena e Reggio Emilia, DSI - Via Campi 213/B, 41100 Modena
(2) CSITE-CNR Bologna V.le Risorgimento 2, 40136 Bologna
e-mail : {domenico.beneventano, sonia.bergamaschi, ilario.benetti, corni.alberto}@unimo.it

Abstract

SI-Designer (Source Integrator Designer) is a designer support tool for semi−automatic integration of heterogeneous sources schemata (relational, object and semi−structured sources); it has been implemented within the MOMIS project and it carries out integration following a semantic approach which uses intelligent *Description Logics*-based techniques, clustering techniques and an extended ODMG-ODL language, $\mathrm{ODL_{I^3}}$, to represent schemata, extracted, integrated information. Starting from the sources' $\mathrm{ODL_{I^3}}$ descriptions (local schemata) SI-Designer supports the designer in the creation of an integrated view of all the sources (global schema) which is expressed in the same $\mathrm{ODL_{I^3}}$ language. We propose SI-Designer as a tool to build *virtual catalogs* in the E-Commerce environment.

1. Introduction

In the last years the need to access distributed information and the problem of the integration of data coming from heterogeneous sources have become more and more important. Companies have equipped themselves with data storing systems building up informative systems containing data which are related one another, but which are often redundant, heterogeneous and not always substantial. On the other hand, the web explosion, both at internet and intranet level, has enlarged the need for the sharing and retrieving of information located in different sources thus obtaining an integrated view so as to eliminate any contradiction or redundancy. The problems that have to be faced in this field are mainly due to both structural and application heterogeneity, as well as to the lack of a common ontology, causing semantic differences between information sources. Moreover these semantic differences can cause different kinds of conflicts, ranging from simple contradictions in names' use (when different names are used by different source to indicate the same concept), to structural conflicts (when different models/primitives are used to represent the same information).

The integration problem is relevant also in the E- Commerce environment. Electronic catalogs are a key component of E-Commerce and they can be organized as individual company catalogs or they can participate in a multi-catalog framework. In the second case, from a user point of view, it is very important to have a uniform interface to search products, that is a uniform view of data coming from different companies catalogs and a unique query language. On the other hand, from a company point of view it is important to guarantee both the uniqueness of their catalogs and the participation in a multi-catalog framework. Virtual Catalogs, as proposed in [14], synthetize this approach as they are conceived as instruments to dynamically retrieve information from multiple catalogs and present product data in a unified manner, without directly storing product data from catalogs. Customers, instead of having to interact with multiple heterogeneous catalogs, can interact in a uniform way with a virtual catalog.

In this work we propose a designer support tool for information integration of both structured and semi-structured sources developed within the MOMIS system. This tool is a suitable instrument also to build *virtual catalogs* in the E-Commerce environment.

The MOMIS project (Mediator environment for Multiple Information Sources) [3, 4, 5] aims to integrate data from structured and semi - structured data sources;
see *http://sparc20.dsi.unimo.it/*. SI-Designer is a designer support tool for semi - automatic integration of heterogeneous sources schema (relational, object and semi-structured sources); it has been implemented within the MOMIS project and it carries out integration following a semantic approach which uses intelligent OLCD Description logics- based techniques, clustering techniques and an ODL-ODMG extended language to represent extracted and integrated information, $\mathrm{ODL_{I^3}}$. Starting from the $\mathrm{ODL_{I^3}}$ descriptions (local schema) of the source, SI-Designer supports the designer in the creation of an integrated view of all the sources (global schema) which is expressed in the same $\mathrm{ODL_{I^3}}$ language.

The global schema is obtained using different stages, creat-

This research has been partially funded by the italian MURST ex-40% INTERDATA project - Metodologie e Tecnologie per la Gestione di Dati e Processi su Reti Internet e Intranet.

ing a *Common Thesaurus* of intra and inter-schema relationships. The sources to be integrated are described through the $\mathrm{ODL_{I^3}}$ language and, by using OLCD inference techniques, intra-schema relationships are extracted and shared in the *Common Thesaurus*.
After this initial phase the *Common Thesaurus* is enriched with inter-schema relationships, obtained in the following way: (1) using the lexical WordNet [15] system, which identifies the affinities between inter-schema concepts on the basis of their denominations' lexicon/meaning; (2) using the ARTEMIS [8] system, which evaluates structural affinities among inter-schema concepts.
Starting from obtained *Common Thesaurus* and using OLCD inference and ARTEMIS clustering techniques, a global schema containing a general view of the integrated sources is obtained. This work describes the SI- Designer tool, supporting the designer in the construction of the *Common Thesaurus* and the global schema. In comparison with proceeding papers on MOMIS, we introduce a new sequence of the global schema construction phases and the full description of the interaction with WordNet. The work is structured as follows. In section 2 the MOMIS and SI-Designer architectures are introduced together with a reference example; in section 3 and 4 the *Common Thesaurus* and global schema construction are respectively described. Some conclusive remarks are reported in section 5.

2. MOMIS Architecture

MOMIS was designed to supply an integrated access to heterogeneous information stored in traditional database (i.e. relational, object oriented), file system, and semi-structured data sources. It follows the I^3 architecture [12] (Figure 1):

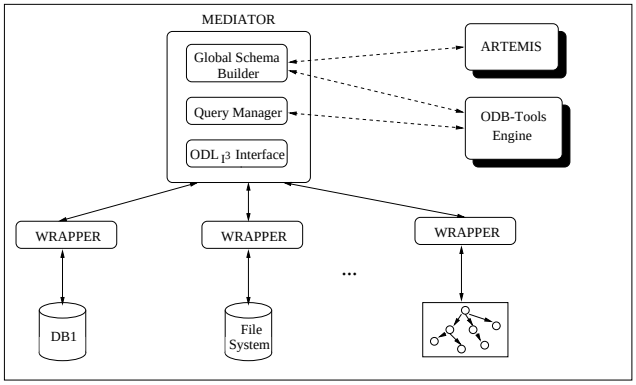

Figure 1: MOMIS Architecture

1. *Wrappers*, placed over each sources, represent the interface modules between the mediator and the local data sources. Their function is double: in the integration phase, they translate the description of the information held in the source. This description is supplied through the $\mathrm{ODL_{I^3}}$ language;
   in the query processing phase, they translate the query which has been received by the mediator (expressed in the common query $\mathrm{OQL_{I^3}}$ language, derived by the OQL language) in a query expressed in the source query language. The wrappers must also export query result data, providing them to the mediator through the data common model used by the system.
2. *The Mediator*: is the core module and it is composed by two separate modules: *Global Schema Builder (GSB)*: it is the module which generates the global schema to be provided to the user, starting from the source descriptions expressed in $\mathrm{ODL_{I^3}}$.
   Query Manager (QM): it is the query management module. It generates $\mathrm{OQL_{I^3}}$ language queries to be sent to wrappers starting from each query posed by the user on the global schema. QM automatically generates the translation of the query into a corresponding set of sub-queries for the sources.
3. SI-Designer, is the framework with a graphical interface which supports the designer in the overall integration process.
4. *ODB-Tools Engine*, is the OLCD *Description Logics* [1, 6] based tool performing schema validation and query optimization [2].
5. *ARTEMIS-Tool Environment*, is an *affinity-based* clustering tool performing $\mathrm{ODL_{I^3}}$ class analysis and clustering [8].

2.1. SI-Designer architecture

Sources integration is based on the individuation of an ontology shared by each source; the ontology is represented as a set of terminological relationships called *Common Thesaurus*.
As shown in Fig 2, *GSB* is composed by two modules:

- SIM (*Source Integrator Module*): extracts intra-schema relationships starting from a relational, object and semi-structured source. Moreover this module performs the "semantic validation" of relationships and infers new relationships by exploiting ODB-Tools capabilities.
- SLIM (*Sources Lexical Integrator Module*) extracts inter-schema relationships between names and attributes of $\mathrm{ODL_{I^3}}$ classes of different sources, exploiting the WordNet lexical system.

SI-Designer (Fig 2 and Fig 3) provides the designer with a graphical interface to interact with SIM, SLIM and ARTEMIS modules showing the extracted relationships and helping him in the *Common Thesaurus* construction.
Once the *Common Thesaurus*, has been built, SI-Designer uses the ARTEMIS module to evaluate a disjoint set of struc-

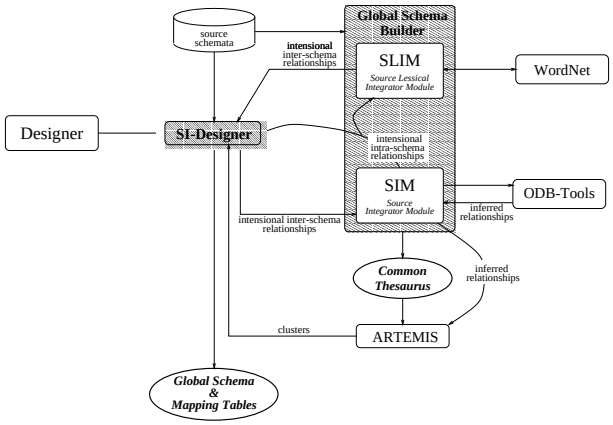

Figure 2: Global Schema Builder Architecture.

tural similar classes (*clusters*). Each cluster has a corresponding *global class* (a view over all similar classes belonging to the cluster) characterized by a set of global attributes and a *mapping-table*. SI-Designer automatically generates a set of global attributes for each global class and a *mapping table* which maps each global attribute into the local attributes of the classes in the cluster.

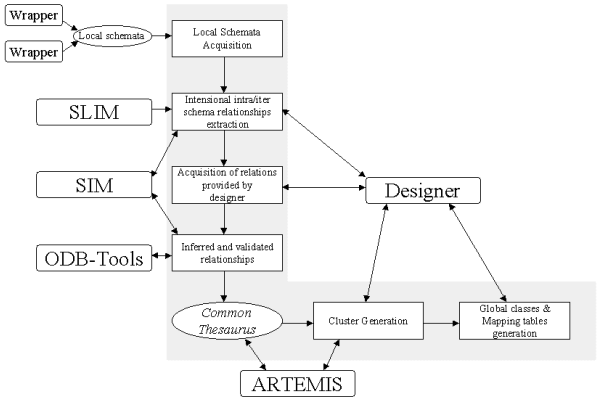

Figure 3: SI-Designer Architecture.

Then, a semi-automatic interaction with the designer starts: the designer may revise the set of global attributes and the mapping table, to assign a name to each global class, so as to achieve a global schema. The integration process can be divided in two phases: (**1**) Generation of the *Common Thesaurus*, (**2**) Generation of the *Global Schema*.

2.2. Running example

In order to illustrate the way our approach works, we will use the following example of integration in the Restaurant Guide domain. Consider two different datasources that store information about restaurants. The `Eating Source` guide-

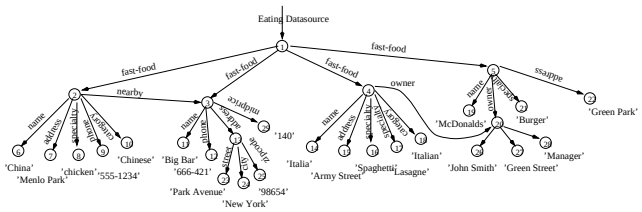

Figure 4: Eating Source (ED)

book (`ED`) contains semistructured objects about restaurants of the west coast and their menu, quality, ... (Figure 4 illustrates a portion of the data).

According to the models proposed in literature for semistructured information sources [7, 16], a semistructured source is represented as a rooted, labeled graph where nodes contain data (e.g., an image or free-form text) and labeled edges describe the concept represented by data in the corresponding node. In Figure 4 there is one root object with four complex children objects that represent restaurants. Each `Restaurant` has a `name`, `category` and `specialty`. Furthermore, some `Restaurant` have an atomic `address` and some other a complex `address`, a `phone`, a complex object `nearby`, that specifies the nearest restaurant, and `owner`, that indicates the `name`, the `address` and the `job` of the restaurant's owner.

In our approach, we derive and explicitly represent also the schema of semi-structured information sources, by introducing the notion of *object pattern* [4], and, in particular, we developed an XML/ODL$_{I^3}$ wrapper able to extract from an XML file an ODL$_{I^3}$ description (see the following section).

The `Food Guide` Database (`FD`) is a relational database containing information about USA restaurants from a wide variety of publications (e.g., newspaper reviews, regional guidebooks). The schema of this source is composed of four relations (see Figure 5), namely, `Restaurant`, `Bistro`, `Person`, and `Brasserie`. Information related to restaurants is maintained into the `Restaurant` relation. `Bistro` instances are a subset of `Restaurant` instances and give information about the small informal restaurants that serve wine. Each `Restaurant` and `Bistro` is managed by a `Person`. Information about places where drinks and snacks are served on are stored in the `Brasserie` relation.

Food Guide Database (FD)

```
Restaurant(r_code, name, street, zip_code, pers_id,
       special_dish, category, tourist_menu_price)
Bistro(r_code, type, pers_id)
Person(pers_id, first_name, last_name,
       qualification)
Brasserie(b_code, name, address)
```

Figure 5: Food Guide Database (FD)

2.3. The ODL$_{I^3}$ language

For a semantically rich representation of source schemas and object patterns associated with information sources, we introduce an object-oriented language, called ODL$_{I^3}$. According to recommendations of ODMG and to the diffusion of I^3/POB [12, 10], the object data model ODL$_{I^3}$ is very close to the ODL language. ODL$_{I^3}$ is a source independent language used for information extraction to describe heterogeneous schemas of structured and semistructured data sources in a common way. ODL$_{I^3}$ introduces the following main extensions with respect to ODL:

Union constructor: denoted by `union`, is introduced to express alternative data structures in the definition of a class, thus capturing requirements of semistructured data.

Optional constructor: is introduced to specify that a class attribute is optional. This constructor too has been introduced to capture requirements of semistructured data.

Integrity constraint rules: are introduced in ODL$_{I^3}$ in order to express, in a declarative way, *if then* integrity constraint rules at both intra- and inter-source level.

Intensional relationships: are *terminological relationships* expressing inter-schema knowledge for the source schemas. Intensional relationships are defined between classes and attributes names, called terms. The following relationships can be specified in ODL$_{I^3}$:

• SYN (Synonym-of), defined between two terms t_i and t_j, with $t_i \neq t_j$, that are considered synonyms in every considered source (i.e., t_i and t_j can be indifferently used in every source to denote a certain concept).

• BT (Broader Terms), or hypernyms, defined between two terms t_i and t_j such as t_i has a broader, more general meaning than t_j. BT relationship is not symmetric. The opposite of BT is NT (Narrower Terms), or hyponyms [1].

• RT (Related Terms), or positive association, defined between two terms t_i and t_j that are generally used together in the same context in the considered sources.

An intensional relationships is only a terminological relationship, with no implication on the extension/compatibility of the structure (domain) of the two involved classes (attributes).

Extensional relationships : Intensional relationships SYN, BT and NT between two classes C_1 and C_2 may be "strengthened" by establishing that they are also *extensional* relationships [4, 9]. Consequently, the following extensional relationships can be defined in ODL$_{I^3}$:

C_1 SYN$_{ext}$ C_2: this means that the instances of C_1 are the same of C_2.

C_1 BT$_{ext}$ C_2: this means that the instances of C_1 are a superset of the instances of C_2.

C_1 NT$_{ext}$ C_2: this means that the instances of C_1 are a subset of the instances of C_2.

[1] We include in the model both the BT and NT relationships for semplicity.

Moreover, extensional relationships "constrain" the structure of the two classes C_1 and C_2 , that is C_1 NT$_{ext}$ C_2 is semantically equivalent to an "isa" relationship.

As to summarize:

C_1 NT$_{ext}$ C_2 is equivalent to C_1 ISA C_2 plus C_1 NT C_2;

C_1 BT$_{ext}$ C_2 is equivalent to C_2 ISA C_1 plus C_1 BT C_2;

C_1 SYN$_{ext}$ C_2 is equivalent to C_1 ISA C_2 and C_2 ISA C_1 plus C_1 SYN C_2.

Mapping Rules : are introduced in ODL$_{I^3}$ in order to express relationships holding between the integrated ODL$_{I^3}$ schema description of the information sources and the ODL$_{I^3}$ schema description of the original sources.

The extraction process for translating object patterns and source schemas into ODL$_{I^3}$ descriptions is shown in [4]; this translation is performed by a wrapper. As an example, below we report the ODL$_{I^3}$ representation of the `ED.Fast-Food` object pattern and of the `FD.Restaurant` relation; other classes of the ODL$_{I^3}$ schema representation of the `ED` and `FD` sources is reported in Appendix A.

```
interface Fast-Food
  ( source semistructured Eating_Source )
{  attribute string      name;
   attribute Address     address;
   attribute integer     phone*;
   attribute set<string> specialty;
   attribute string      category;
   attribute Fast-Food   nearby*;
   attribute integer     midprice*;
   attribute Owner       owner*;     };

interface Restaurant
( source relational Food_Guide )
key r_code
foreign_key(pers_id) references Person )
{  attribute string         r_code;
   attribute string         name;
   attribute string         street;
   attribute string         zip_code;
   attribute integer        pers_id;
   attribute string         special_dish;
   attribute integer        category;
   attribute integer tourist_menu_price; };
```

3. Generation of a *Common Thesaurus*

The *Common Thesaurus* is a set of terminological intensional and extensional relationships, describing inter-schema knowledge about classes and attributes of sources schemas; it provides a reference on which to base the identification of classes candidate to integration and subsequent derivation of their global representation. In the Common Thesaurus, we express inter-schema knowledge in form of terminological relationships (SYN, BT, NT, and RT) and extensional relationships (SYN$_{ext}$, BT$_{ext}$, and NT$_{ext}$ between

classes and/or attribute names. The Common Thesaurus is constructed through an incremental process during which relationships are added in the following order: **1.** *schema-derived relationships*: Terminological and extensional relationships holding at intra-schema level. These relationships are extracted by the SIM module by analyzing each ODL$_{I^3}$ schema separately. In particular, intra-schema RT relationships are extracted from the specification of foreign keys in relational source schemas. When a foreign key is also a primary key both in the original and in the referenced relation, a BT/NT relationship is extracted As an example, from the `Bistro` and `Restaurant` classes in the ODL$_{I^3}$ descriptions reported in Appendix A, it follows that ⟨`FD.Restaurant` BT `FD.Bistro`⟩

2. *lexical-derived relationships*: Terminological relationships holding at inter- schema level are extracted by the SLIM module by analyzing different sources ODL$_{I^3}$ schemas together. In the next section we will examine these relationships, as their extraction is one of the main contributions of this paper w.r.t. previous papers on MOMIS [3, 4, 5].

3. *designer-supplied relationships*: Terminological and extensional relationships supplied directly by the designer, to capture specific domain knowledge about the source schemas. This is a crucial operation, because the new relationships are forced to belong to the Common Thesaurus and thus used to generate the global integrated schema. This means that, if a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema. Our system help the designer in detecting wrong relationships by performing a *Relationships validation* step with ODB-Tools. The validation of intensional relationships between attribute names is based on the compatibility of the domains associated with the attributes. **4.** *inferred relationships*: Terminological and extensional new relationships inferred by exploiting inference capabilities of ODB-Tools.

All these relationships are added to the Common Thesaurus and thus considered in the subsequent phase of construction of Global Schema. For a more detailed description of the above described process see [4].

Terminological relationships defined in each step hold at the intensional level by definition. Furthermore, in each of the above step the designer may "strengthen" a terminological relationships SYN, BT and NT between two classes C_1 and C_2 by establishing that they hold also at the extensional level, thus defining also an extensional relationship. The specification of an extensional relationship, on one hand, implies the insertion of a corresponding intensional relationship in the Common Thesaurus and, on the other hand, enable subsumption computation (i.e., inferred relationships) and consistency checking between two classes the C_1 and C_2.

3.1. Lexical-derived inter-schema relationships

The extraction of these relationships is based upon the lexical relations holding between classes and attributes names, deriving from the mining of used words. This is a kind of knowledge which is not based on the rules of a data definition language but derives from the name assigned by the designer. It is a designer's task to assign descriptive/meaningful names or, at least, correctly interpretable names. An interpretation uncertainty is therefore inherent to the language ambiguity; Bates [13] writes *"the probability of two persons using the same term in describing the same thing is less than 20%"*.
Anyway, knowledge associated with schema names is an opportunity that must be exploited to extract relationships. As it is almost impossible to carry out this task manually when the number and dimensions of schema grows, it was decided to experiment the use of the WordNet [15] lexical system to extract and propose to the designer intensional inter-schema relationships.

3.1.1. The WordNet database

WordNet is a lexical database which was developed by the Princeton University [15] Cognitive science Laboratory. WordNet is inspired by current psycholinguistic human lexical memory connected theories and it is regarded as the most important researcher's available resource in the fields of computational linguistics, textual analysis and other related areas. The lexical Wordnet database, in the current 1.6 version has 64089 lemma which are organized in 99757 synonym sets (*synset*).
The starting point of lexical semantics is the constatation of the existence of a conventional association between the words form (i.e. the way in which they are pronounced or written) and the concept/meaning they express; such association is of the many-to-many kind, giving rise to the following properties:
Synonymy: property of a concept/meaning which can be expressed with two or more words. A synonyms group is named *synset*. Note that one and only *synset* exists for each concept/meaning. Later a *synset* will be indicated with s, while $\mathcal{S}$ will indicate the *synset* set.
Polysemy: property of a single word having two or more meanings.
The correspondence between the words form and their meaning is synthesized in the so called *Lexical Matrix* $\mathcal{M}$, in which the words meaning are reported in rows (hence each row represents a *synset*) and columns represent the words form (form/base lemma).
Each matrix element is a definition $e = (f, m)$, where f is the *base form* and m (*meaning*) is the meaning counter; for example (`address`, `2`) refers to the address where a person or an organization can be found; while (`address`,

1) refers to a computer address in the informatics sphere. From here on the base form and the meaning of an element $e = (f, m)$ will be respectively indicated with $e.f$ and $e.m$. An element of the $\mathcal{M}$ matrix may be *null* or *indefinite*.
As only one $\mathcal{M}$ row is associated to a *synset*, from here on we will use $s \in \mathcal{S}$ as a $\mathcal{M}$ row indicator. In other words the non null elements of the $\mathcal{M}[s]$ row, represent each and every s element. In the same way, as only one $\mathcal{M}$ column is associated to a base form, from here on we will use the base forms as $\mathcal{M}$ columns index.

3.1.2. Semantic relationships between schema terms

With the concept of *term* we associate a definition to each class or attribute name. A *term* is formed by the $t = (n, e)$ couple, where n indicates a class or attribute name, and e indicates a definition. A class or attribute name n are qualified as follows a class name is qualified by the name of the source schema to whom the class belongs (`source_name.class_name`), an attribute name is moreover qualified with the name of the class to whom it belongs (`source_name.class_name.attribute_name`). The classes and attributes names set is indicated by **N**; the set of words in **N** is indicated by $\mathbb{I}$. The relation between *synset* defined in Wordnet are the starting point to define semantic relations between words. Various relations are obtainable with the WordNet database; some of them are between single words others are between *synset*. In this context we will use the following relations between *synset*: Synonymy, Hypernymy, Hyponymy, Olonymy, Meronymy and Correlation[2] As hyponymy and meronymy are inverse relations to hypernymy and olonymy, respectively, the set of relations between *synset* is the following:
$\mathcal{W} = \{\mathbf{S}_{ynonymy}, \mathbf{H}_{ypernymy}, \mathbf{O}_{lonymy}, \mathbf{C}_{orrelation}\}$.
Given the *synset* $\mathcal{S}$ set and the $\mathcal{W}$ relations set, The function $\phi : \mathcal{S} \times \mathcal{W} \longrightarrow 2^{\mathcal{S}}$ is inserted giving for each *synset* s the set of *synset* associated through the $r \in \mathcal{W}$ relation:

$$\phi(s, r) = \{s' \mid s' \in \mathcal{S}, r \in \mathcal{W}, \langle s' r s \rangle\}$$

Given a *synset* $\mathcal{S}$ set and a $\mathbb{I}$set of words, the function$\mathcal{H} : \mathcal{S} \longrightarrow 2^{\mathbb{I}}$ is defined associating, on the basis of the lexical matrix, a set of words to a given *synset* :

$$\mathcal{H}(s) = \{t = (n, e) | n \in \mathbf{N}, \mathcal{M}[s][t.e.f] = t.e\}$$

We can hence obtain the relations between the words using the relations existing between the *synset* that contain those words. Given a set of words $\mathbb{I}$, the set of relations between words $\mathcal{R}$, $\mathcal{R} \subseteq \mathbb{I} \times \mathcal{W} \times \mathbb{I}$, is defined as follows:

$$\mathcal{R} = \{\langle t_i r t_j \rangle | r \in \mathcal{W}, t_i, t_j \in \mathbb{I}, \exists s : t_i \in \mathcal{H}(s), t_j \in \phi(s, r), t_i \neq t_j\}$$

The relations deriving from are proposed as semantic relations to be inserted in the *Common Thesaurus* according to the following correspondence:
Synonymy: corresponds to a SYN relation.
Hypernymy: corresponds to a BT relation.
Olonymy: corresponds to a RT relation.
Correlation: corresponds to a RT relation.
On the basis of these considerations, an algorithm has been developed which having as input the terms related to the schemata to be integrated, outputs the detected semantic relations. We will now consider the use of the developed tool, SLIM. Given a name n the associated words must be chosen. This choice involves two steps:

1. Base form choice. The designer is supported in such a choice by the system which gives him the word base form using the WordNet morphologic processor. For example, in Figure 6, by selecting the `address` attribute, we obtain the `address` base form from the morphologic processor. If a base form is not found, or there is an ambiguity[3], or it is not satisfactory, the designer can directly introduce it.

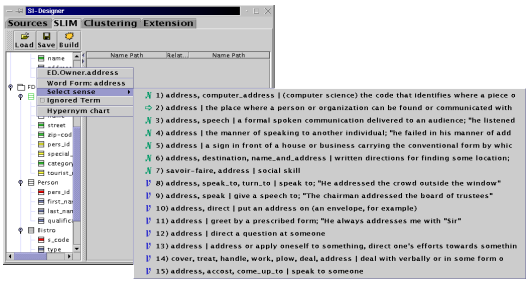

Figure 6: `address` Meanings

2. Meaning choice. The designer can relate a name to one, more than one, or no meaning. The choice of not relating a name to any meaning can be made for various reasons: (**a**) the concept is too complex and it can not be expressed with one word (e.g. `special_dish`); (**b**) it belongs to the *tops*, i.e. to the generic concepts, therefore it would be related to the whole (e.g. `relation`); (**c**) it is a substitute key, therefore it doesn't add any knowledge (e.g. `pers_id` of the table `Person`); (**d**) it is used as *foreign key*, therefore this relation has already been used during the extraction of relations from the schema structure (e.g. `pers_id` of the `Bistro` table). In such a choice, the designer is supported by the tool which gives him, for a given name, its hypernymy hierarchy. Figure 7 shows the hypernymy hierarchy of `street`; in this case, meanings 1 and 2 have to be both selected as they are similar.
The designer selects one or more meanings from those found in WordNet starting from the base form chosen at step 1. Therefore, all the words that are related to the same name, share the same base form. For example, in Figure 6 all the

[2]Correlation is a relation which links 2 *synset* sharing the same hypernym, i.e. the same "father".

[3]For example 3 `axes` base forms are found: `ax` (1 sense), `axis` (5 senses), `axe` (2 senses).

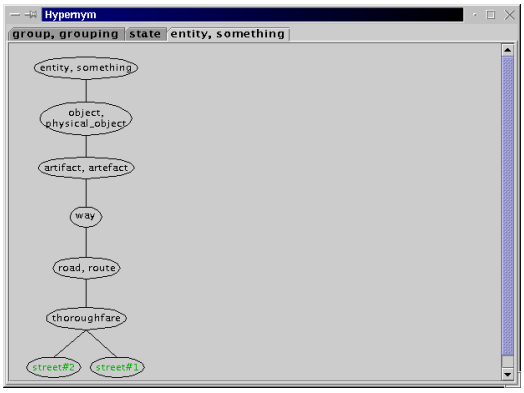

Figure 7: SLIM : hypernymy hierarchy of `street`

15 meanings that WordNet relates to the `address` base form are obtained. Selecting them all, i.e. considering 15 words for the `address` attribute, we could obtain "*wrong*" results, which are not suitable within the examined context. Some of them are shown in the following:

```
<Eating_Source.Fast-Food.address
  NT Food_Guide.Restaurant.name>
<Eating_Source.Owner.job
  NT Food_Guide.Person.Person>
<Food_Guide.Person.Person
  NT Eating_Source.Fast-Food.category>
```

Note that some of this relationships can look quite strange but they are true in some particular context. The problem, hence, is now resolving the meaning ambiguity so that a context-suitable couple (base form, meaning counter) can be supplied to WordNet for each concept of a source. To help the designer in the choice of the "right" meaning, for each couple (base form, meaning counter), a syntactic category (names - **N**, verbs - **V**, adjectives -**Aj**, Adverbs - **Av**) is indicated (see Figure 6).
This semi-automatic approach reduces the complexity of the designer task, in fact, a "*difficult*" problem (i.e. is finding the relations between all words), is divided in many "*easy*" ones, choosing each terms' meaning from a list. In practice this is an 80/20 problem, that is 80% of the words is worked out in the 20% of the time, just the time for reading the definitions, while the remaining 20% occupies the 80% of the time, because the choice is between very similar meanings. To speed up the 80% part a "cache" of the already selected couple (base form, meaning counter) is used (see Figure 6: the symbol $\Rightarrow$ denotes the meaning already chosen by the designer for the `address` concept).
Furthermore, SI-Designer can show the generalization hierarchy of the meanings in order to help the designer in the most difficult choices. For example, (see Figure 8) in the case of `city`: we see that "city#2" inherits only from "administrative_district ..." whereas "city#1" inherits also from "geographic_area", thus, as "city#2" refers only to an administrative connotation, we select "city#1". At the end of

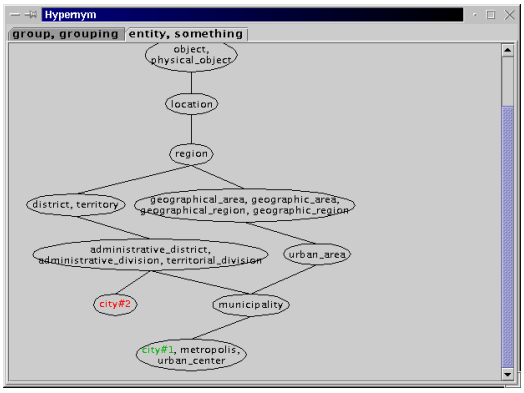

Figure 8: SLIM : hypernymy hierarchy of `city`

this phase, SI-Designer shows the relationships derived by using WordNet (see Figure 9). The designer may delete any of the shown relationships and add new ones. When the set of relationships satisfy him, he SELECTS the **Clustering** panel, thus implicitly saving the set of relationships in the Common Thesaurus.

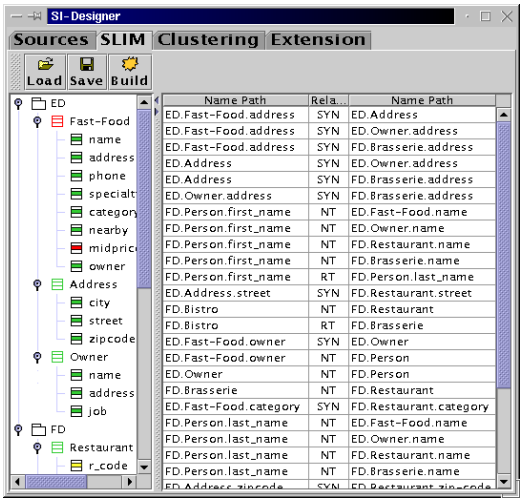

Figure 9: Inter-schema relationships extracted by SLIM

4. Generation of the global classes

Once the *Common Thesaurus* has been built, SI-Designer can generate the global class. Such activity is carried out in the following way: (1) *Affinities calculation*: (2) *Cluster generation*: (3) Generation of the global attributes and of the *mapping-table*.
In the first phase, SI-Designer works as an interface between the ARTEMIS module and the designer who can interact

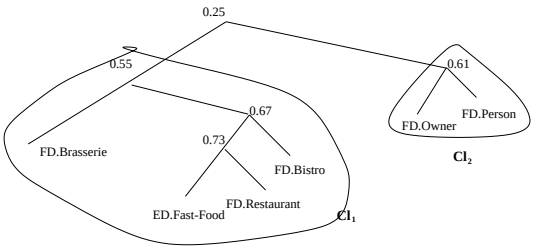

Figure 10: Affinity tree: clusters with $\sigma = 0.5$.

with ARTEMIS many times, until he is satisfied by the set of the obtained clusters. In the second phase, the tool builds, for each cluster, a global class to which a set of global attributes is associated, supplying the designer an interface to revise the global attributes set and the mapping-tables proposed by the tool. The designer can use the same interface to assign a name to each global class. For a more detailed description of this phase see [4].

4.1. Clusters generation

To identify all the ODL$_{I^3}$ classes having affinity in the considered source schemas, ARTEMIS uses a hierarchical clustering technique, which classifies classes into groups at different levels of affinity, forming a tree [11]. The leaves represent all the local classes: adjacent leaves represent classes with high affinity, while leaves far apart from each other represent classes with low affinity; each node represents a clustering level and is associated to the affinity coefficient between the sub-trees (clusters) it joins.
Within SI-Designer, the designer, at any iteration, can insert a threshold value which is used by ARTEMIS to build clusters: each cluster is made by all the classes belonging to a sub-tree having at the root node a coefficient which is higher than the threshold value. Figure 10 shows the local classes tree and the clusters for a threshold value σ=0.5.

4.2. Global attributes and mapping tables generation

For each cluster, SI-Designer creates a set of global attributes and, for each of them, it determines the correspondence with the *local attributes* (i.e. those of the classes belonging to the cluster to which the global class corresponds). In some cases, the correspondence is unique while in other cases the tool identifies different kinds of correspondences but can't solve their ambiguity: in this case the tool asks the designer to choose the right one. The tool builds the global attributes set to be associated to a cluster in two phases: (1) Union of the attributes of all the classes belonging to the cluster (2) Fusion of the "similar" attributes; In this phase SI-Designer tries to eliminate these redundancies considering the relationships of the *Common Thesaurus*. The fusion process is automatic for the attributes which are associated

Food_Place	code	name	...	zone
ED.Fast-Food	null	name	...	'Pacific Coast'
FD.Restaurant	r_code	name	...	'Atlantic Coast'
FD.Bistro	r_code	null	...	'Atlantic Coast'
FD.Brasserie	b_code	name	...	'Atlantic Coast'

Figure 11: `Food_Place` Mapping-table.

by validated relationships while it is not always automatic when their relationships are not validated. In particular, SI-Designer operates in the following way:

Attributes associated in validated relationships.
For these attributes the fusion is always automatic. To each of the attributes connected by `SYN` relationships SI-Designer will connect one only global attribute: the domain and local attributes are the same and the name can be chosen by the designer between those proposed by SI-Designer or explicitly introduced. The attributes connected by `NT` relationships are treated by SI-Designer substituting them with a global attribute having the same name and the domain of the generalization attribute. For example: the `name`, `first_name` and `last_name` attributes are connected by the `first_name NT name` and `last_name BT name` specialization relations, therefore the attributes will be represented by the global attribute `name`.
Attributes associated in non validated relationships.
Common Thesaurus relationships that do not passed validations belong to this category: SI-Designer can automatically find a global attribute only in a limited set of cases: it's up to the designer to add global attributes needed to complete the integration. The automatic individuation of a global attribute is only performed in this case, if the attributes in the relationships have the following requirements: (1) they are linked by `SYN` or `BT` relationship; (2) related classes belong to the same cluster; (3) they represents aggregation hierarchy (complex attributes or foreign key);
Once the global attribute set has been found, the designer can extend it to represent further local sources information: this case often occurs when some information is stored in a local source as a metadata.
While creating global attributes, SI-Designer builds also a *mapping-table* (see Figure 11).
It is a $MT[CL][AG]$ table where CL represents the set of the local classes which belong to the cluster referred by the mapping-table, and AG represents the global attributes set built by SI-Designer. Let C be the name of a local class, A the name of a global attribute and AL the name of a local attribute; each element $MT[C][A]$ of the table can assume the following values:
• AL, with $AL \in C$. This value is used when:
(**a**) a global attribute refers to the information stored in AL local attribute. For example, the `name` global attribute reflects the `name` attributes in `FD.Restaurant`;

(**b**) NT relationship between attributes belonging to different classes. For example, the `code` global attribute refers to `r_Code` in `FD.Restaurant` and to `b_Code` of `Brasserie`.
• AL_1 ***and ... and*** AL_n, with $AL_i \in C, i = 1, \ldots, n$.
This is used when the value of the A attribute represents the linking of the values assumed by a set of attributes belonging to the same local class C. For example, the `name` global attribute of the $\texttt{Cl}_2$ global class represents the linking of `first_name` and `last_name` of class local attributes from `Food_Guide.Person` local class. By specifying the *and* correspondence between `first_name` and `last_name` for the global attribute `name`, we state that the values of both `first_name` and `last_name` have to be considered as values of `name` when class `FD.Person` is considered.
• ***case of*** AL $cost_1$: AL_1 ... $cost_n$: AL_n
$AL, AL_i \in C, i = 1, ..., n$ and $cost_i$ are constants.
This situation occurs when the A global attribute can assume one value in a set of AL_i belonging to the same class and the value choice passes through a third attribute, from the same class, which act as a selector. • $costant$. In this case a global attribute value doesn't refer to any local attribute and a value is set by the designer. For example, `zone` global attribute of `Food_Place` gets the `'Pacific Coast'` value while accessing `ED.Fast-Food` and the `'Atlantic Coast'` value while accessing `FD.Bistro`.
• $null$. In this case A global attribute, while accessing the C local class doesn't get any value. For example, the `code` global attribute, of the `Food_Place` global class, doesn't assume any value in the `ED.Restaurant` local class.
SI-Designer provides the designer with an interface that allows a complete view of all the global classes (names and attributes), including the mapping-tables, class names setting and *mapping table* editing.
At the end, the ODL_{I^3} description of the Global Schema is obtained. For example, the global class `Food_Place` (see Figure 12) and its mapping table (see Figure 11) are obtained. Having this global class available, the user can pose queries on Restaurants with respect to this class, disregarding the original sources and their schemata.

5. Conclusions

In this paper the SI-Designer support tool for the integration of heterogeneous data sources has been introduced.
This tool works within the MOMIS project, which is under development at the University of Modena and Reggio Emilia. This work was particularly focused on the SLIM component description, implemented to perform interaction with the WordNet system, in order to automatic extract intensional inter-schema relationships.
A direct interaction with WordNet, without the SLIM support, cannot be proposed as an effective aid to the designer because of the very high number of relationship suggested by WordNet for each term, considering that only few of them are valid within the context of the considered sources. The interaction with the designer, implemented through SLIM, is a satisfactory level and represents a real aid during sources'integration activity.
SI-Designer is a good candidate as a tool to build Virtual Catalogs in the E-Commerce environment. We are applying it in the development of a virtual catalog for cars involving Fiat, Renault and Wolkswagen catalogs.

```
interface =  Food_Place
{ attribute name
    mapping_rule =  ED.Fast-Food.name,
                    FD.Restaurant.name,
                    FD.Brasserie.name;
   ...
  attribute   specialty
    mapping_rule
      ED.Fast-Food.specialty,
      FD.Restaurant.special_dish;
  attribute   address
    mapping_rule    ED.Fast-Food.address,
      (FD.Restaurant.street and
       FD.Restaurant.zip_code and
       FD.Brasserie.address);
  attribute  zone
    mapping_rule
      ED.Fast-Food = 'Pacific Coast'',
      FD.Restaurant = 'Atlantic Coast',
      FD.Bistro = 'Atlantic Coast',
      FD.Brasserie = 'Atlantic Coast';
}
```

Figure 12: `Food_Place` global class

6. References

[1] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, July/August 1998.

[2] D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in oodb. In *Int. Conference on Data Engineering - ICDE97*, 1997.

[3] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Records*, 28(1), March 1999.

[4] S. Bergamaschi, S. Castano, M. Vincini and D. Beneventano, Semantic Integration and Query of Heterogeneous Information Sources, *special issue of Data*

and Knowledge Engineering (DKE) on Intelligent Information Integration. (to appear) (A preliminary version of the paper appears in the proceedings of IJCAI-99 Workshop on Intelligent Information Integration 31 July 1999, Stockholm)

[5] D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori and M. Vincini. Information Integration: the MOMIS Project Demonstration. supporting multiple inheritance. *Proc. Int. Conf. on Very Large Data Bases VLDB-2000* (Cairo, Egypt, 2000)

[6] S. Bergamaschi and B. Nebel. Acquisition and validation of complex object database schemata supporting multiple inheritance. *Journal of Applied Intelligence*, 4:185–203, 1994.

[7] P. Buneman. Semistructured data. In *Proc. of 1997 Symposium on Principles of Database Systems (PODS97)*, pages 117–121, Tucson, Arizona, 1997.

[8] S. Castano and V. De Antonellis. Deriving global conceptual views from multiple information sources. In *preProc. of ER'97 Preconference Symposium on Conceptual Modeling, Historical Perspectives and Future Directions*, 1997.

[9] 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.

[10] R.G.G. Cattell and others., editors. *The Object Data Standard: ODMG 2.0*. Morgan Kaufmann Publishers, San Francisco, CA, 1997.

[11] B. Everitt. *Computer-Aided Database Design: the DATAID Project*. Heinemann Educational Books Ltd, Social Science Research Council, 1974.

[12] R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.

[13] Bates M. Subject access in online catalogs: A design model. *Journal of the American Society for Information Science*, 11:357–376, 1986.

[14] Arthur M. Keller, Smart Catalogs and Virtual Catalogs, *International Conference on Frontiers of Electronic Commerce*, October 95; earlier version appeared in USENIX Workshop on Electronic Commerce, July 1995. A later version is http://WWW-DB.Stanford.EDU/pub/keller

[15] A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[16] Y.Papakonstantinou, H.Garcia-Molina, and J.Widom. Object exchange across heterogeneous information sources. In *Proc. of ICDE95*, Taipei, Taiwan, 1995.

A. ODL$_{I^3}$ sources descriptions

```
Eating_Source (ED):
interface Fast-Food
    ( source semistructured Eating_Source )
{  attribute string         name;
    attribute Address        address;
    attribute integer        phone*;
    attribute set<string>    specialty;
    attribute string         category;
    attribute Restaurant     nearby*;
    attribute integer        midprice*;
    attribute Owner          owner*; };

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

interface Owner (
   source semistructured Eating_Source)
{  attribute string   name;
    attribute Address  address;
    attribute string   job; };

Food_Guide_Source (FD):
interface Restaurant ( source
   relational Food_Guide  key r_code
   foreign_key(pers_id) references Person )
{  attribute string    r_code;
   attribute string    name;
   attribute string    street;
   attribute string    zip_code;
   attribute integer   pers_id;
   attribute string    special_dish;
   attribute integer   category;
   attribute integer   tourist_menu_price;};

interface Person (
   source relational Food_Guide key pers_id)
{  attribute integer        pers_id;
    attribute string         first_name;
    attribute string         last_name;
    attribute integer        qualification;};

interface Bistro ( source relational Food_Guide
   key r_code
   foreign_key(r_code) references Restaurant,
   foreign_key(pers_id) references Person)
{  attribute string      r_code;
    attribute set<string> type;
    attribute integer     pers_id;};

interface Brasserie ( source relational Food_Guide
   key b_code )
{  attribute string       b_code;
    attribute string       name;
    attribute string       address; };
```