

THE MOMIS SYSTEM

S. Bergamaschi^{1,2}, D. Beneventano^{1,2}, A. Corni, G.Gelati¹, F.Guerra¹, D.Miselli¹, D.Montanari¹, M.Vincini¹

¹ Dipartimento di Ingegneria dell'Informazione
Universita' di Modena e Reggio Emilia

² IEIIT-CNR Istituto di Elettronica e di Ingegneria dell'Informazione e
delle Telecomunicazioni - Bologna

Index

PART 1: SYSTEM PRESENTATION

1) SYNOPSIS	2
2) THE MOMIS INTEGRATION METHODOLOGY	2
3) THE MOMIS QUERY MANAGER	5
4) THE MOMIS DEVELOPMENT WITHIN THE SEWASIE PROJECT	8

PART 2: APPLICATION TUTORIAL

1) THE MOMIS ONTOLOGY BUILDER	9
2) THE MOMIS QUERY MANAGER	35

1. SYNOPSIS

The Semantic Web exploits semantic markups to provide Web pages with machine-readable definitions. It thus requires the a priori existence of ontologies that represent the domains associated with the given information sources. This approach relies on the accuracy of the selected reference ontology and on the existence of tools allowing an efficient and effective annotation as well as the subsequent search of the annotated knowledge base. However, most ontologies in common use are generic and that the annotation phase (in which semantic annotations connect Web page parts to ontology items) causes a loss of semantics. Also, cooperation among many actors has to be achieved to build the searchable knowledge body and then use it. On the contrary, it is possible to “build” an ontology by involving the sources and annotating them (according to a lexical ontology) that more precisely represents the domain.

The MOMIS (Mediator envirOnment for Multiple Information Sources) is a framework to perform information extraction and integration from both structured and semi-structured data sources, plus query management facilities to take incoming queries and process them through the exploitation of the annotated GVV. The framework consists of a language and several semi-automatic tools.

- The ODL-I3 language is an object-oriented language, with an underlying Description Logic; it is derived from the standard ODMG.
- Information integration is performed in a semi-automatic way, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and ODL-I3 descriptions of source schemas with a combination of clustering techniques and Description Logics. This integration process gives rise to a virtual integrated view of the underlying sources (the Global Schema, GVV) for which mapping rules and integrity constraints are specified to handle heterogeneity.
- The MOMIS Query Manager is the coordinated set of functions which take an incoming query, decompose the query according to the mapping of the GVV onto the local data sources relevant for the query, send the subqueries to these data sources, collect their answers, perform any residual filtering as necessary, and finally deliver the answer to the requesting user.

The MOMIS system is based on a conventional wrapper/mediator architecture, and provides methods and open tools for data management in Internet-based information systems by using a CORBA-2 interface. The MOMIS development begun as a joint collaboration between the University of Modena and Reggio Emilia and University of Milano and Brescia, within the INTERDATA national research project. The research activities continue within the SEWASIE European project (IST-2001-34825).

The MOMIS methodology is completely described in D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini: "Synthesizing an Integrated Ontology ", IEEE Internet Computing Magazine, September-October 2003,42-51.

2. THE MOMIS INTEGRATION METHODOLOGY

In this section, we describe the information integration process for building the GVV. The process is shown in Figure 1.

The ODL_{I3} Language

As a common data model for integrating a given set of local information sources, MOMIS uses an object-oriented language called ODL_{I3}, which is an evolution of the OODBMS standard language ODL. ODL_{I3} extends ODL with the following relationships expressing intra- and inter-schema knowledge for the source schemas: SYN (synonym of), BT (broader terms), NT (narrower terms) and RT (related terms). By means of ODL_{I3}, only one language is exploited to describe both the sources (the input of the synthesis process) and the GVV (the result of the process). The translation of

ODL₃ descriptions into one of the Semantic Web standards such as RDF, DAML+OIL, OWL is a straightforward process. In fact, from a general perspective an ODL₃ concept corresponds to a *Class* of a the Semantic Web standard, and ODL₃ relationships are translated into *properties*.

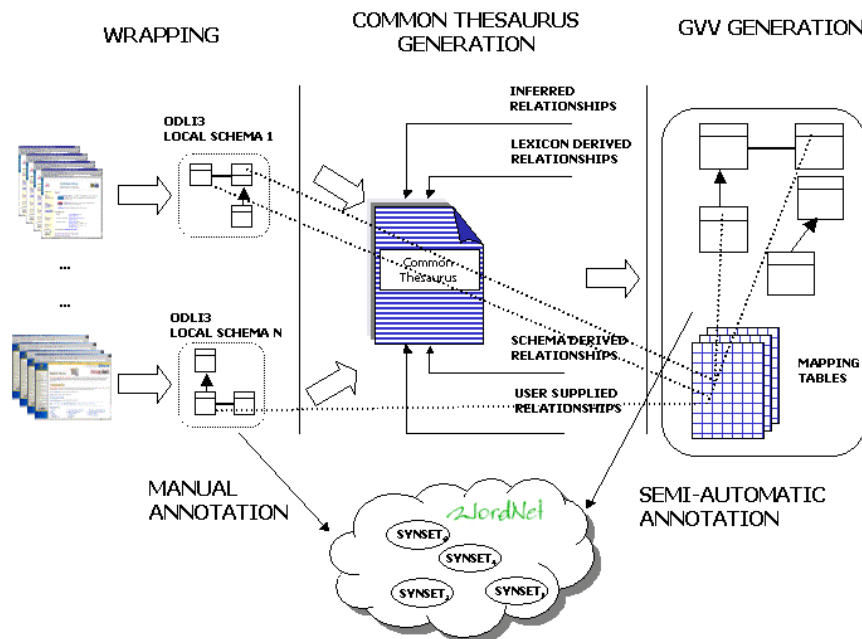


Figure 1. Overview of the ontology-generation process. The figure shows the local schemas' generation, where local schemas are annotated according to the lexical ontology WordNet, the Common Thesaurus generation, and finally the GVV global classes. In particular, these ones are connected by means of mapping tables to the local schemas and are (semi-automatically) annotated according to WordNet.

Wrapping: extracting data structure for sources

A wrapper logically converts the source data structure into the ODL₃ model. The wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources. For conventional structured information sources (e.g. relational databases), schema description is always available and can be directly translated. For semistructured information sources, a schema description is in general not directly available at the sources. A basic characteristic of semistructured data is that they are “self-describing” hence information associated with the schema is specified within data. Thus, a wrapper has to implement a methodology to extract and explicitly represent the conceptual schema of a semi-structured source. We developed a wrapper for XML/DTDs files. By using that wrapper, DTD elements are translated into semi-structured objects, according to different proposed methods [S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web - From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.].

Manual annotation of a local source with WordNet

For each element of the local schema, the integration designer has to manually choose the appropriate meaning in the WordNet [A.G. Miller. *A lexical database for English*. Communications of the ACM, 38(11):39:41,1995.] lexical system. The annotation phase is composed of two different steps: in the Word Form choice step, the WordNet morphologic processor aids the designer by suggesting a word form corresponding to the given term; in the Meaning choice step the designer can choose to map an element on zero, one or more senses. The annotation assigns a name (this name can be the original one or a word form chosen from the designer), and a set of meanings, to each local class and attribute of the local schema.

Common Thesaurus Generation

MOMIS constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of SYN, BT, NT, and RT relationships. The Common Thesaurus is constructed through an incremental process in which the following relationships are added:

schema-derived relationships: relationships holding at intra-schema level are automatically extracted by analyzing each schema separately. For example, analyzing XML data files, BT/NT relationships are generated from couples IDs/IDREFs and RT relationships from nested elements.

lexicon-derived relationship: we exploit the annotation phase in order to translate relationships holding at the lexical level into relationships to be added to the Common Thesaurus. For example, the hypernymy lexical relation is translated into a BT relationship.

designer-supplied relationships: new relationships can be supplied directly by the designer, to capture specific domain knowledge. If a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema;

inferred relationships: Description Logics (DL) techniques of ODB-Tools [D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini ODB-QOptimizer: a tool for semantic query optimization in OODB. ICDE'97, UK, April 1997.] are exploited to infer new relationships, by means of subsumption computation applied to a "virtual schema" obtained by interpreting BT/NT as subclass relationships and RT as domain attributes.

Global Virtual View (GVV) Generation

The MOMIS methodology allows us to identify similar ODL_{β} classes, that is, classes that describe the same or semantically related concept in different sources. To this end, *affinity coefficients* are evaluated for all possible pairs of ODL_{β} classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name Affinity* coefficient) and their attributes (*Structural Affinity* coefficient) and are fused into the *Global Affinity* coefficient, calculated by means of the linear combination of the two coefficients¹⁴. Global affinity coefficients are then used by a hierarchical clustering algorithm, to classify ODL_{β} classes according to their degree of affinity.

For each cluster C_i , a Global Class GC , with a set of Global Attributes GA_1, \dots, GA_N , and a Mapping Table MT , expressing mappings between local and global attributes, are defined. The Mapping Table is a table whose columns represent the local classes, which belong to the Global Class and whose rows represent the global attributes. An element $MT[GA][LC]$ is a function which represents how local attributes of LC are mapped into the global attribute GA : $MT[GA][LC] = f(LAS)$

where LAS is a subset of the local attributes of LC .

Global Virtual View (GVV) Generation

The MOMIS methodology allows us to identify similar ODL_{β} classes, that is, classes that describe the same or semantically related concepts in different sources. To this end, *affinity coefficients* are evaluated for all possible pairs of ODL_{β} classes, on the basis of the relationships included in the Common Thesaurus, properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name Affinity* coefficient) and their attributes (*Structural Affinity* coefficient) and are fused into the *Global Affinity* coefficient, calculated by means of the linear combination of the two coefficients [S. Castano, V. De Antonellis, S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. IEEE TKDE, 13(2), 2001]. Global affinity coefficients are then used by a hierarchical clustering algorithm, to classify ODL_{β} classes according to their degree of affinity.

For each cluster C , a Global Class GC , with a set of Global Attributes GA_1, \dots, GA_N , and a Mapping Table MT , expressing mappings between local and global attributes, are defined. The Mapping Table is a table whose columns represent the local classes, which belong to the Global Class, and whose

rows represent the global attributes. An element $MT[GA][L]$ is a function which represents how local attributes of L are mapped into the global attribute GA. **Global Virtual View (GVV) Annotation**

A GVV annotation consists in assigning a name and a set (eventually empty) of meanings to each global element (class or attribute).

In order to semi-automatically associate an annotation to each global class, we consider the set of all its “broadest” local classes, w.r.t. the relationships included in the Common Thesaurus. On the basis of this set, the designer will annotate the global class as follows:

name choice: the designer is responsible for the choice of the name: the system only suggests a list of possible names. The designer may select a name within the proposed list or introduce a new one.

meaning choice: the union of the meanings of the “broadest” local classes are proposed to the designer as meanings of the Global Class; the designer may change this set.

A similar approach is used for Global Attributes Annotation.

3. THE MOMIS QUERY MANAGER

The MOMIS Query Manager is the coordinated set of functions which take an incoming query, define a decomposition of the query according to the mapping of the GVV onto the local data sources relevant for the query, send the subqueries to these data sources, collect their answers, perform any residual filtering as necessary, and finally deliver the answer to the requesting user.

The query processing of queries expressed on the GVV (*global query*) consist of the following steps:

- 1) *Query rewriting* : to rewrite a global query as an equivalent set of queries expressed on the local sources (*local queries*).
- 2) *Local queries execution* : the local queries are sent and executed at local sources.
- 3) *Fusion and Reconciliation*: the local answers are fused into the global answer.

3.1 Query rewriting

The query rewriting method depends on the approach used to model the mappings between the GVV and the local schemata: Momis uses a GAV approach, then the global query is rewritten by means of unfolding, that is, by expanding each atom of the global query according to its definition in the mapping. In the following we describe our query rewriting method.

Given a global class G, related to the local class L1, L2, ..., Ln, we consider a Global Query Q over G:

$$Q = \text{select } \langle Q_select_list \rangle \text{ from } G \text{ where } \langle Q_condition \rangle$$

where $\langle Q_condition \rangle$ is a Boolean expression of positive atomic constraints: (GA1 op value) or (GA1 op GA2), where GA1 and GA2 are attributes of the global class G. Note that we do not consider negation in our framework.

The *query rewriting* process is composed of the following steps:

1) Atomic constraint mapping

In this step, each atomic constraint of Q is rewritten into one that can be supported by the local class. The Atomic constraint mapping is performed on the basis of *mapping functions* defined in the Mapping Table: for each not null element $MT[GA][L]$ we define a *mapping function*, denoted by $MT_F[GA][L]$, which represents how the local attributes of L are mapped into the global attribute GA.

The function $MT_F[GA][L]$ must be a function executable/supported by the local source of the class L. For example, for relational sources, $MT_F[GA][L]$ is an SQL-92 value expression; the following

defaults hold: if $MT[GA][L]=\{LA\}$ then $MT_F[GA][L]=LA$ and, if $MT[GA][L]$ contains more than one string attribute, then $MT_F[GA][L]$ is the string concatenation.

The constraint mapping depends on the definition of *Resolution Functions* for global attributes, which will be introduced in subsection 3.3; an example of constraint mapping will be shown in section 3.4.

2) Residual Constraints computation

Intuitively, residual constraints are the constraints of the global query that are not mapped in all local queries.

3) Local select-list computation

The select-list of a local query is a set of attributes, including the global query attributes, the join attributes, the the residual constraints attributes, translated into the correspondent set of local attributes on the basis of the mapping table.

The output of the Query Rewriting process is a set of local queries; each local query Q_L over L is in a form supported on the local source of the class L . For example, for relational sources, a local query Q_L over L will be of the form:

$$Q_L = \text{select } \langle Q_L\text{-select-list} \rangle \text{ from } L \text{ where } \langle Q_L\text{-condition} \rangle$$

3.2 Local queries execution

A local query L_Q is sent to the source including the local class L ; its answer is transformed by applying the mapping functions related to L : in this way, we perform the conversion of the local class instances into the GVV instances. The result of this conversion is materialized in a temporary table.

3.3 Fusion and Reconciliation

When all the local query answers are materialized, their fusion and reconciliation into the global answer is performed as follows:

a) Join Conditions

To identify instances of the same object (*Object Identification*) and fuse them we introduce Join Conditions among local classes. Join Conditions are defined by selecting one ore more global attributes JA_1, \dots, JA_n (called Join Attributes); a join attribute must have a not null mapping in all the local classes. The join condition between local classes L_1 and L_2 is obtained as:

$$JC(L_1, L_2) : L_1.JA_1 = L_2.JA_1 \text{ and } \dots \text{ and } L_1.JA_n = L_2.JA_n$$

Note that a join condition is an expression over global attributes: in fact, this condition is applied after the conversion of the local class instances into the GVV instances.

b) Full-Disjunction

In our GAV approach, each global class is expressed by means of a particular operator, called *full-disjunction* [Anand Rajaraman , Jeffrey D. Ullman: Integrating Information by Outerjoins and Full Disjunctions. PODS 1996, pages 238-248], introduced by Ullman in the context of Information Integration, with the following, informal, definition: “Computing the natural outerjoin of many relations in a way that preserves all possible connections among facts”. We apply this definition in our context, then, given a global class G composed of L_1, L_2, \dots, L_n , the instance of G is the full-disjunction of L_1, L_2, \dots, L_n , computed on the basis of the Join Conditions; intuitively, we use the Join Conditions instead of the natural outer join. In the case of two classes, the Full Disjunction corresponds to the full outer join.

c) Resolution functions

For each global attribute mapped into more than one source, a Resolution Function [Felix Naumann, Matthias Häußler : Declarative Data Merging with Conflict Resolution. International

Conference on Information Quality (IQ 2002). 2002, pages 212-224] is defined to solve data conflicts. Some examples of resolution functions introduced in our system are:

- *Generic resolution function* : Additional input to the resolution function can be values from other domains. For instance, when dealing with different prices, the value of a date attribute might be used to choose the most recent price.
- The *highest informational quality* value on the basis of an information quality model: the quality score can refer to a source in general, or be attribute-specific.
- *Random function*
- Resolution functions for *numerical attributes*: SUM, AVG, ..

3.4 Homogeneous Attributes

If the designer knows that there are no data conflicts for a global attribute mapped into more than one source (that is, the instances of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an *Homogeneous Attribute*. Of course, for homogeneous attributes we do not need to define a Resolution Function. A global attribute mapped into one source is a particular case of homogeneous attribute. For default, each global attribute is considered as homogeneous.

3.4.1. Atomic constraint mapping for Homogeneous Attribute

The constraint mapping depends on the definition of the Resolution Function, for example, if the numerical global attribute GA is mapped into L1 and L2, and we define as resolution function the AVG function, the constraint (GA = value) cannot be pushed at the local sources, as for data conflicts of this attribute, the constraint may be globally true but locally false. In this case, the constraint is mapped as *true* in both the local sources. On the other hand, if GA is an homogeneous attribute the constraint can be pushed at the local sources.

For homogeneous attributes the constraint mappings is defined as follows:

- An atomic constraint (GA op value) is mapped into the local class L as:

(MT _F [GA][L] op value) if	MT[GA][L] is not null and
	the op operator is supported into L
true	otherwise

- An atomic constraint (GA1 op GA2) is mapped into the local class L as:

(MT _F [GA1][L] op MT _F [GA2][L]) if	MT[GA1][L] and MT[GA1][L] are not null
	and
	the op operator is supported into L
true	otherwise

For default, each operator used in the global query is supported into a local class: the designer can define, for each local class, the operators which are not supported.

For non homogeneous attributes the constraint mappings is performed on the basis of the resolution functions. We are working/developing on this problem, in general, an atomic constraint defined on non homogeneous attributes is not rewritten in the local sources (that is, is rewritten as true). The current implementation of the system assumes that each global operator is supported.

4. THE MOMIS DEVELOPMENT WITHIN THE SEWASIE PROJECT

MOMIS supports the semiautomatic building and annotation of domain ontologies by integrating the schemas of information sources. The MOMIS framework is currently adopted in the Semantic Web Agents in Integrated Economies (SEWASIE) European research project (www.sewasie.org), coordinated by UNIMORE. SEWASIE aims at implementing an advanced search engine that enables intelligent access to heterogeneous data sources on the Web via semantic enrichment, providing the basis for structured secure Web-based communication. To achieve this goal, SEWASIE creates a virtual network based on Sewasie information nodes (SINodes), which consist of managed information sources, wrappers, and a metadata repository. SINodes metadata represent GVV of the overall information sources that each manage. To maintain the GVV of a SINode, we are investigating two distinct aspects: the system overload in maintaining the built ontologies and the effects of inserting new sources that could modify existing ontologies. Future work will address the improving of the annotation phase by allowing the designer to face multilingual environments, that is adopting a multilingual lexical database.

5. APPLICATION TUTORIAL

The MOMIS Ontology Builder

The architecture of the MOMIS system is depicted in Fig. 1. The Ontology Builder (shown as SI-Designer) was developed to support the integration methodology (<http://www.dbgroup.unimo.it/Momis>) presented in [D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini: "Synthesizing an Integrated Ontology ", IEEE Internet Computing Magazine, September-October 2003,42-51]. The Query Manager architecture and functionalities are described in [Maurizio Lenzerini, Zoran Majkić, Domenico Beneventano, Federica Mandreoli: Techniques for query reformulation, query merging, and information reconciliation, D3.2.A SEWASIE Semantic Webs and AgentS in Integrated Economies IST-2001-34825)].

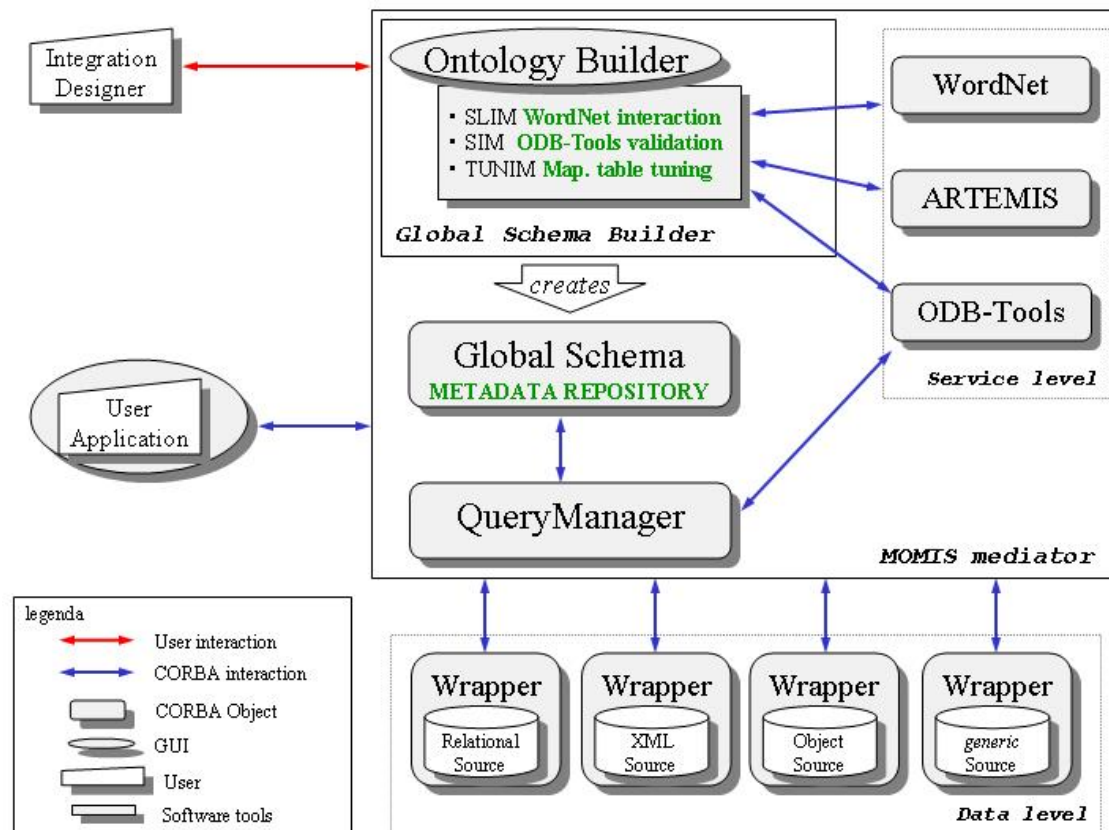


Fig. 1 - MOMIS Architecture

The MOMIS “Ontology Builder” component (hereafter, “SiDesigner”, or “SiD”, or “Ontology Builder”, or “OB”) guides the user (the so-called integration designer) during the integration process which leads to the development of an integrated view of the sources (GVV).

2. 1 Starting the Ontology Builder and the Query Manager

To start the Ontology Builder component, you must be connected with Internet (better if you have a fast connection: basic modem connections may be inadequate to manage the required downloads).

Then, type the following address in your browser's address bar:

<http://dbgroup.unimo.it/Momis/momis-iswc/>

A username and a password are necessary; if you don't have it yet, you may request one from Sonia Bergamaschi by sending an email at the address dbgroup@sparc20.ing.unimo.it.

After login, click on the link "[This starts the Momis Ontology Builder through Java Web start](#)".

The prototype needs Java Web Start to be installed on your system. You can find further information on how to get it at the address (<http://www.java.sun.com/products/javawebstart/>).

Load the Ontology Builder (please note that the first time you connect, loading may take a few seconds. This mainly depends on your kind of connection).

When the loading ends, the following window appears:



You can choose among the following options.

1. "**Create a new schema**" enables the creation of a new schema by starting the acquisition of new sources;
2. "**Open a schema**" enables the loading of a previous-saved schema, such as a schema saved during a previous work session. (see "**How to save your work.**" for further details).
3. "**Open example**" provides access to examples ready to be used. The desired example can be selected from the scrolling menu.

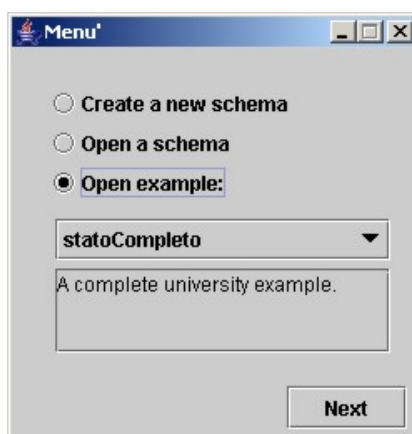


Fig. 2 – Inserting a Global Schema name

Click on "*Next*" to start the graphical interface.

When the option "**Create a new schema**" is selected, the graphical interface starts with the window in Fig. 6.: you have to insert the name of the global schema you are going to build. This name will be assigned to the whole integrated schema, the so-called "Global Virtual View" or "GVV".

The MOMIS system starts visualizing the Ontology Builder. If you want to query a previously created GVV you may directly switch to the Query Manager by selecting the Query Manager panel at the top of the interface (Fig. 6).

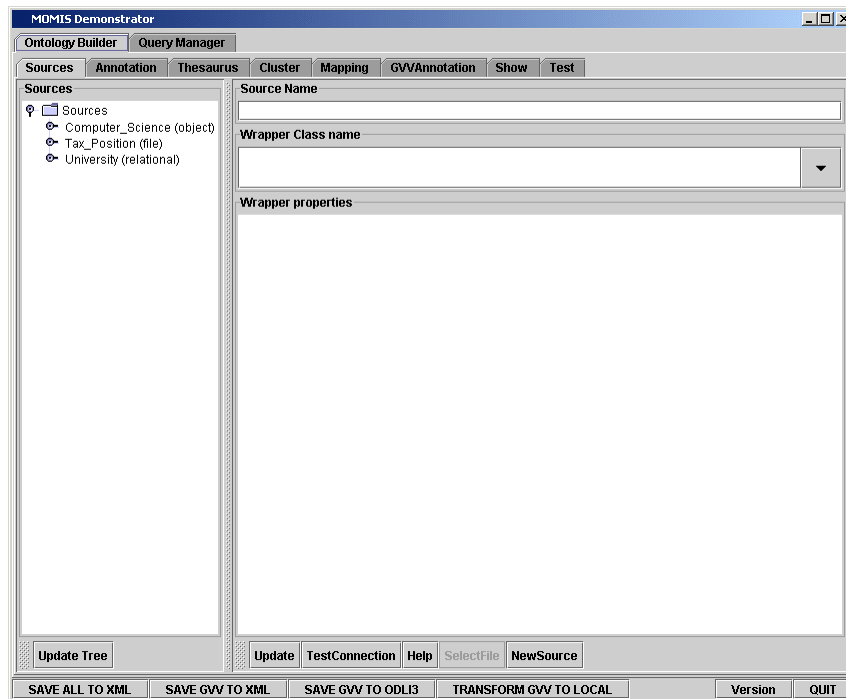


Fig. 6– The MOMIS demonstrator

How to save your work.

You can save the work done by clicking on the button at the bottom of the graphical interface at any time during the integration process (Fig. 7). The button starts a new window where you can choose both the name of the file (in which you want to save the status of your work) and its location (path) in your system.

Even if the file extension is not an essential element for saving your work; nevertheless the file will be saved in the XML format. Therefore, we suggest that you add the extension “.xml” to the name of the file to be saved.



Fig. 7 – Saving the current state of the work

2. Local Sources Schemata Extraction

Acquiring data sources (“First Phase: Sources”).

The first phase of the sources integration concerns their acquisition. Fig. 8 shows the Ontology

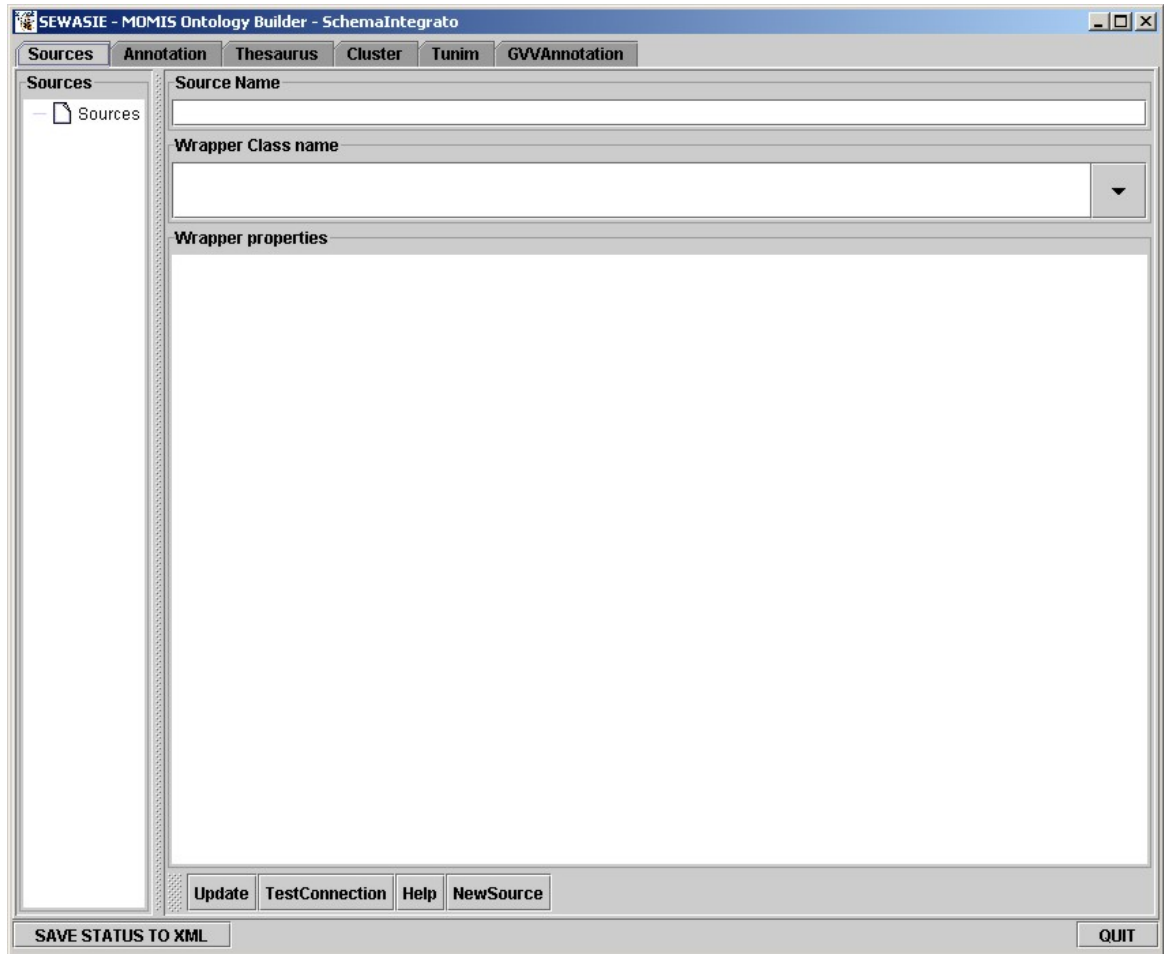


Fig. 8 – First phase “Sources”

Builder window.

Data sources are acquired by means of “wrappers”.

The Ontology Builder implements several wrappers; in this document you can find information/explanations about XML/DTD Wrappers and JDBC/ODBC Wrappers.

In the following table, we report the icons used by the graphical interface and their intended meanings:

 Local source

 Global Source

 Local Interface(Local source)

 Global Interface(Global Class)

 Local attribute

 Global attribute

2.1 XML/DTD Wrapper.

Such a wrapper enables users to acquire XML/DTD sources. In particular we are explicitly referring to files whose extension is “.dtd” (Document Type Definition), and therefore to files which contain a description of the objects and attributes being modelled.

Four steps are required to set up this kind of wrappers:

1. Insert the source name (the name you want to assign to the source) in the input field named “*Source Name*”.
2. Select the wrapper
“*it.unimo.dbgroup.momis.communication.core.xmlDtd.WrapperXMLCore*” from the menu.
3. Insert the path of the file “.dtd”. The path has to be inserted with the closing bar “/”.
4. Select “*NewSource*” from the bottom of the interface. This will start the acquisition of the new source.

Following these steps, and if the selected file corresponds to a known format, then the data source will appear on the left side of the interface. You can see the “structure” of the data source by clicking on the left side of each node, i.e., click on the element showed by the arrow in Fig 9.

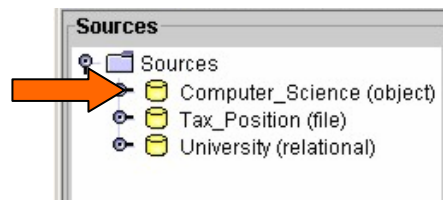


Fig. 9 – Source Navigation

2.2 JDBC/ODBC Wrapper.

The JDBC/ODBC wrapper enables the user to acquire data sources such as MS Excel files, MS Access databases, etc. In principle, this wrapper should be able to manage any source that “exports” its own data in a ODBC data source. The first step for using data services is to configure an ODBC data source.

2.2.1. Configuring an ODBC Data Source.

Creation of an “ODBC data source”:

1. Check for the required user rights; you need to have administrator rights
2. Open the “*ODBC Data Source Administrator*” interface; you can obtain it from “*Start–Settings – Control Panel – ODBC Data Source Administrator*” or from “*Start – Settings - Control Panel –Administrative Tools – ODBC Data Source Administrator*”.

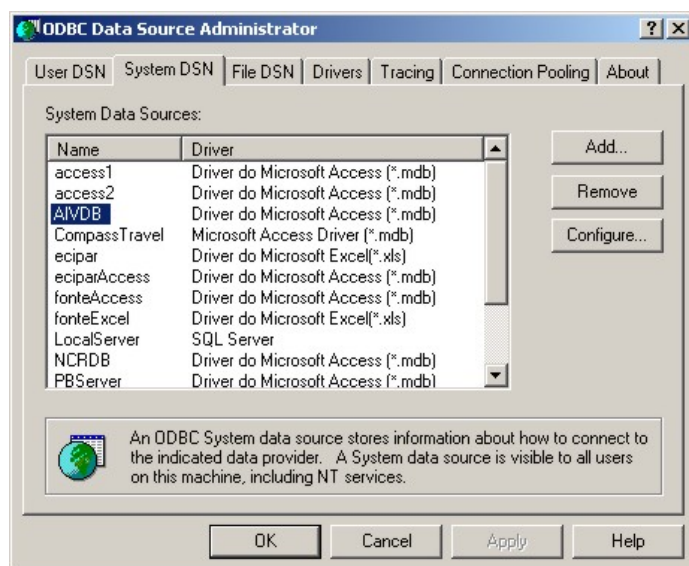


Fig. 3 – Administrating an ODBC Data Source

3. Select “*System DSN*” by clicking on the corresponding tabbed pane.
4. Click on “*Add*” to configure a new data source; select an already existing data source; click on “*Configure*” to modify the settings of the selected source (such as changing passwords and so on). To delete a link to an ODBC Data Source, you have to select it and click on “*Remove*”.
5. when clicking on “*Add*”, a new window (Fig 5) appears. Then, from the displayed list, you can select a driver for which you want to set up a data source from the list. In our example we

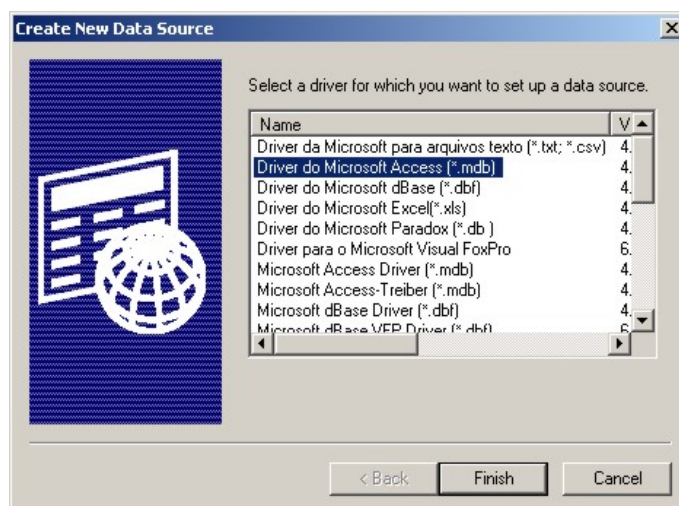


Fig. 4 – Selecting a driver for a new data source

selected “*Driver do Microsoft Access (*.mdb)*”. Click on “*End*” to confirm your choice.

6. The name of the new source has to be inserted in the field “*Data Source Name*”, while a description of the data source can be given in the field “*Data Source Description*” (Fig. 6). Descriptions are not mandatory. Click on “*Select...*” to specify the file (the path of the file within your system) that you want to set up as an ODBC data source.

7. Click on “Ok” to finish the ODBC data source set up.



Fig. 5 – Setup of a source

At this point, an ODBC data source has been configured. Now, the SIDesigner’s wrapper can be connected with it and the schema extraction started.

When an ODBC source has been created, four steps are required to set up a JDBC/ODBC wrapper:

1. Insert the source name (the name you want to assign to the source) in the input field named “Source Name”.
2. Select the wrapper “*it.unimo.dbgroup.momis.communication.core.jdbc.WrapperJdbcCore*” from the menu.
3. Insert the required parameters, namely:

- `WrapperJdbcCore.DriverClassName=sun.jdbc.odbc.JdbcOdbcDriver`

(It defines the driver to be used)

- `WrapperJdbcCore.Url=jdbc:odbc:dataSourceNameDefined`

(Replace **dataSourceNameDefined** with the name of the desired ODBC Data source. The proper set up of the ODBC Data source is a crucial condition for using JDBC/ODBC wrappers! See “

” for further details)

- `WrapperJdbcCore.User=usernameDefined`

(Replace **usernameDefined** with the username required to login to the source. Just skip this input in case no username is required.)

- `WrapperJdbcCore.Password=pwdDefined`

(Replace **pwdDefined** with the password required to login to the source. Otherwise, do not fill this parameter)

- `WrapperJdbcCore.debug=true`

(Select **true** as a debug parameter)

4. Select “NewSource” from the bottom of the interface and you will acquire the new source.

In the lower part of the window you will find a “Help” button that provides help for the user concerning the proper settings for the JDBC wrapper. If the button is pressed when no source name is

given, then a message will remind you to insert it. In the same way, a message will appear when no wrapper is selected. In other cases, the “Help” button opens a window with a list of the parameters described above. From this window you can copy and paste the cited parameters in the appropriate fields. (see Fig 10).

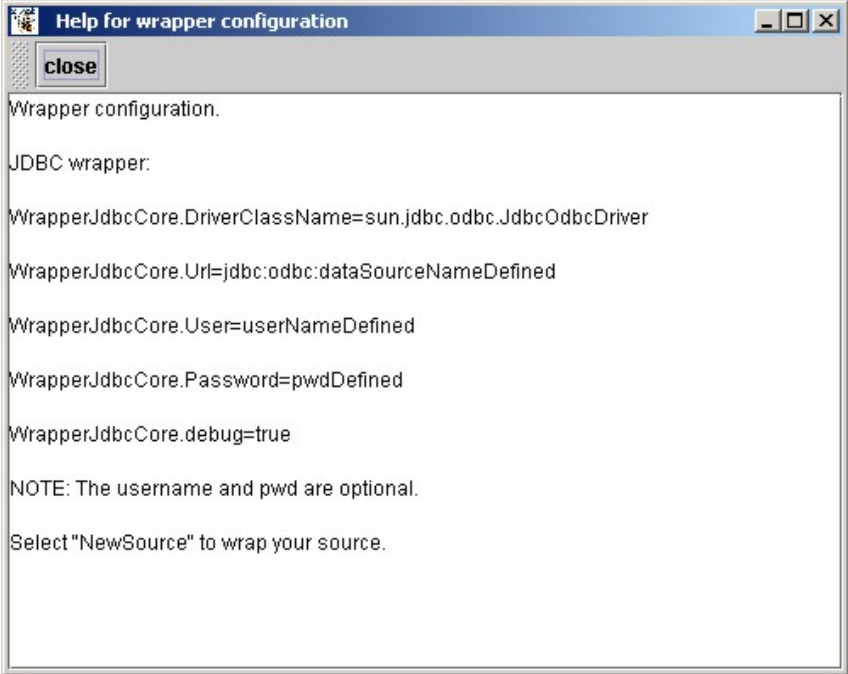


Fig. 10 – Help window

To load more than one source, repeat the four steps described above. Each loaded source will be represented on the left side of the MOMIS interface. The structure of each loaded data source can be viewed. (Fig. 9).

After the acquisition of two or more data sources, you can obtain a view similar to the one depicted in Fig. 11.

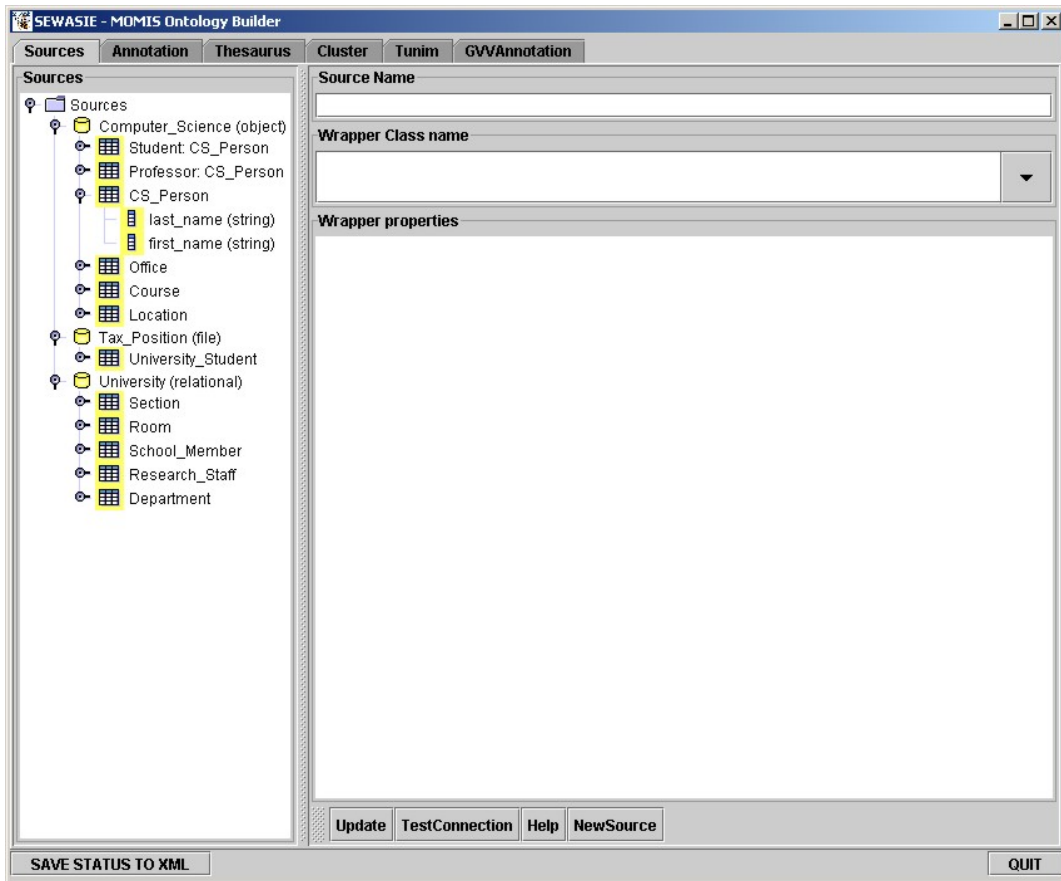


Fig. 11 – View of the acquired data sources

3. Local Sources Annotation with WordNet.

Once you have acquired the sources, you have to annotate each element of a source (classes and attributes) with respect to the lexical ontology WordNet.

You have to choose “Annotation” in the top of the graphical interface (Fig.).

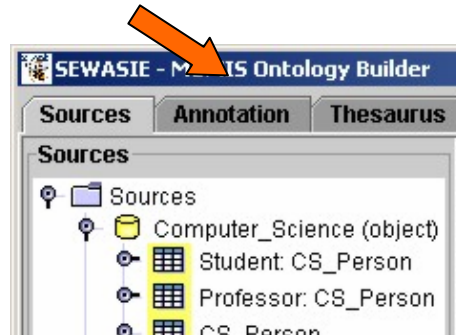


Fig. 12 – Changing the phase: selecting Annotation

The annotation phase consists of two steps repeated for each class and attribute:









1. Choice of a word form: one English term describing a concept represented by a class or an attribute has to be introduced;
2. Choice of meanings: the designer needs to select for each word form zero, one or more meanings belonging to the selected word form.

Word forms and meanings are proposed by means of the interaction with the “*WordNet*®” lexical database (<http://www.cogsci.princeton.edu/~wn/>). The graphical interface shows on the left a tree representing the involved sources. The user can navigate across the tree and choose the classes/attributes to be annotated by means of the right button of the mouse.

You do not have to annotate all the elements, but the more terms are annotated, the more lexical relationships will likely be extracted by interacting with WordNet (see next section).

The MOMIS Ontology Builder attempts to automatically propose a word form for each term. This operation has good results if source terms are expressed in English without special characters, compound terms, etc.

The interface uses some colored icons in order to help the designer to find the terms to be annotated:

-  **Local Class: selected word form, no chosen meaning.**
-  **Local Class: word form not found, no chosen meaning.**
-  **Local Class: selected word form, chosen meaning.**
-  **Local Class: ignored term.**
-  **Local Attribute: selected word form, no chosen meaning.**
-  **Local Attribute: no found word form, no chosen meaning.**
-  **Local Attribute: selected word form, chosen meaning.**
-  **Local Attribute: ignored term.**

Tab. 1 –Icons used in the Annotation phase

By using the right button of the mouse you may access to a sub-menu to annotate the terms (see Fig. 13).

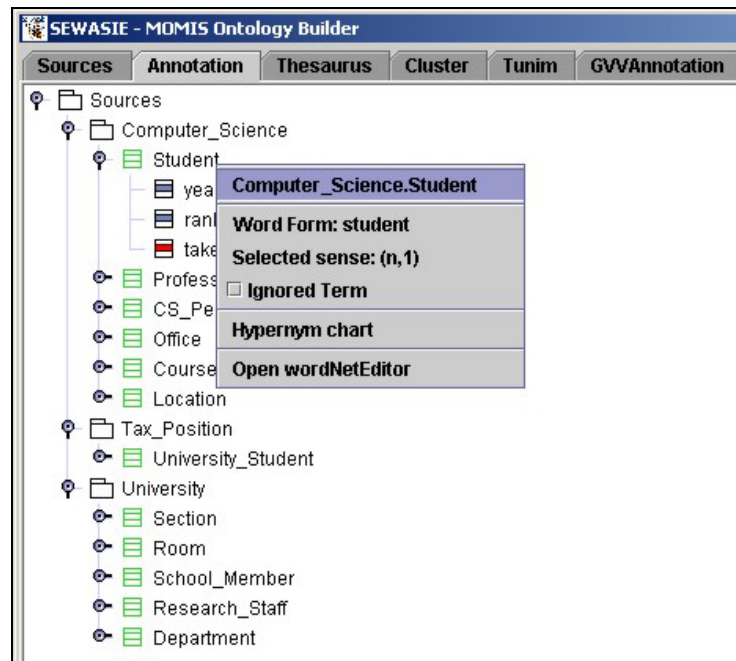




Fig. 13 – Annotation Menu

The menu shows different things in different cases:

Case 1. Word Form not found within the lexical database, icons:   (see Fig. 14)

- On the first row, the name and the path of the selected term are visualized;
- “**Word Form:**” is followed by the name of the term (class or attribute) not founded in WordNet;
- “**No match**” shows that a corresponding term in WordNet is not found;
- Other options.

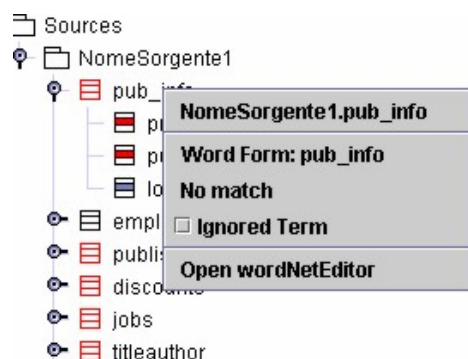


Fig. 14 – Word form not found

Now, the designer may select “*Word Form*” in order to insert a new word form. The interface shows if the inserted word form is found. The designer may insert a new word form by means of the “*Retry*” button or abort the operation coming back to the previous word form with the “*Cancel*” button (see Fig. 15).

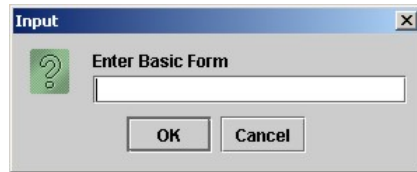


Fig. 15 – Interface for word form insertion

Case 2. Word form found inside WordNet and meaning not chosen, icons:  

- On the first row, the name and the path of the selected term are visualized;
- “**Word Form:**” is followed by the name of the term (class or attribute) not founded in WordNet;
- “**Select sense**” is the option to be selected in order to assign the more appropriate meaning to the term;
- Other options.

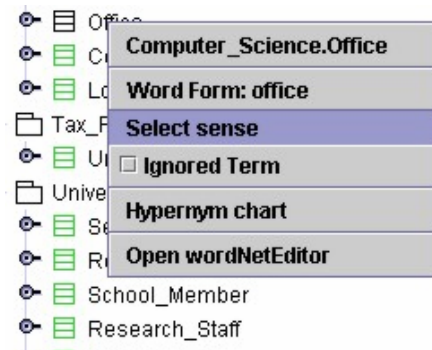


Fig. 16 – No chosen meaning

By choosing the option “*Select sense*” a new window (see Fig. 17), where it is possible to select one or more of the meanings, appears. In order to select a meaning the designer has to press the left button on the mouse pointing the sense to be chosen.

Inside the WordNet DB there are: nouns (), verbs (), adjectives (), adverbs ().

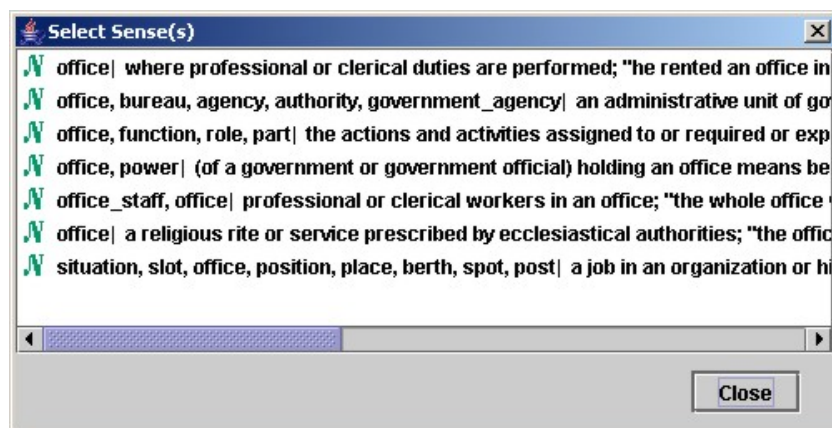




Fig. 17 –Selecting a Meaning

A selected meaning is shown by means of a flag () in the left part of the window.

The system shows with an arrow () previously selected meanings for the same word form. The designer may select another time these meanings or select new ones.

Case 3. Word Form found inside the WordNet DB and meaning chosen, icons:  

- On the first row, the name and the path of the selected term are visualized;
- “**Word Form:**” is followed by the name of the term (class or attribute) not founded in WordNet;
- “**Selected sense:**” is followed by the chosen meanings’ list “(type, meaning ID number);
- Other options.

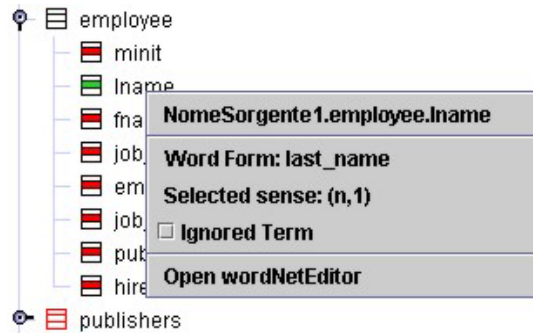


Fig. 18 –meaning selection

It is always possible to change the annotation.

In order to do it, you have to press the right button of the mouse pointing the term you want to modify and then select the “*Selected sense*” option with the left button: a new window for meaning selection appears.

5. Common Thesaurus Generation.

Relationships among terms, concepts, attributes, etc. allow for richer and more efficient management of the underlying content. In order to extract and manage the relationships, the user has to select the “*Thesaurus*” tab at the top of the interface window (see Fig. 19).

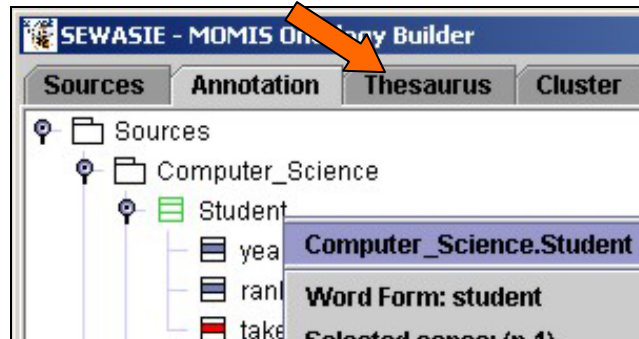


Fig. 19 - Pannello delle relazioni

The Ontology Builder is able to manage different kinds of relationships, classify-able with respect to the kind and the provider.

Kind of relationships

SYN	The terms are synonyms
NT	The meaning of the left term is narrower than the one of the right
BT	The meaning of the left term is broader than the one of the right
RT	The terms are correlated

Relationship's provider

740	Lexicon-derived relationships
780	User-provided relationships
900	Schema derived relationships
902	Inferred relationships

The method used to compute the relationships is the following:

1. Schema-derived relationships extraction. The relationships are automatically extracted by the system. Press the button “*Schema-Derived Rels*”.
2. Lexicon-derived relationships extraction. It is performed on the basis of the annotation phase. Press the button “*Lexicon-Derived Rels*”.
3. New relationships provided manually by the user. Press the button “*Add*”.
4. Inferring new relationships. The relationships are inferred by means of a Description Logics engine (ODB-Tools). Press the button “*Inferred Rels*”.

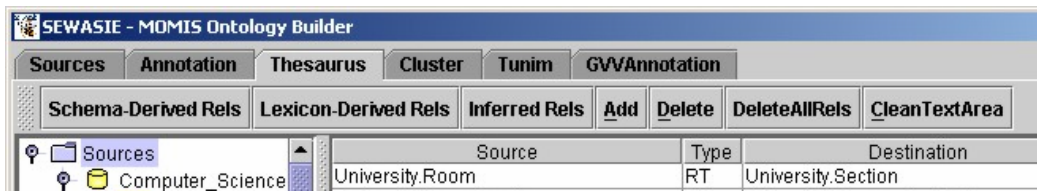


fig. 20– Common Thesaurus Generation Interface

Global Classes creation (Cluster).

In order to create the global classes the user has to press “Cluster” (see Fig.21).

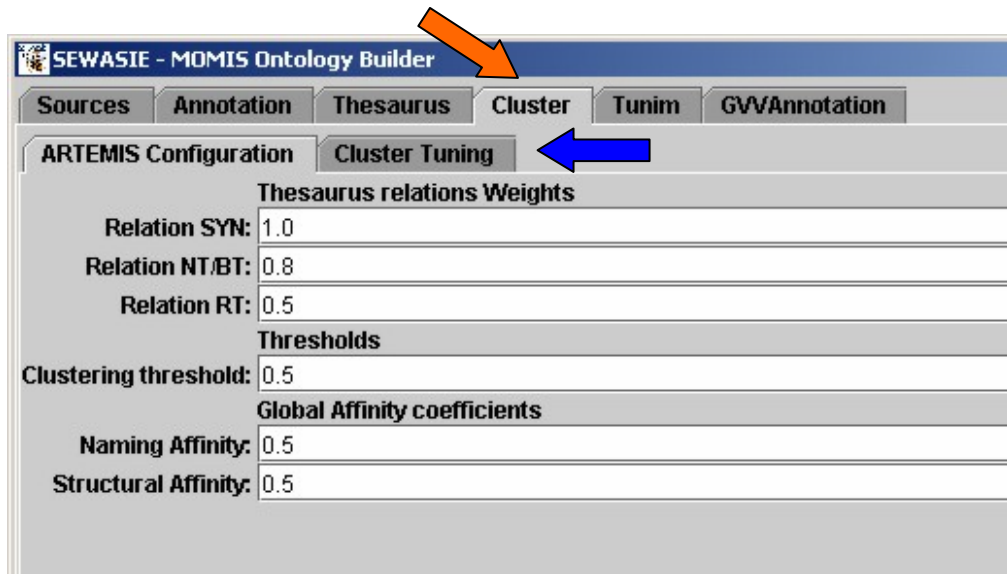


fig. 21 – Cluster definition

In fig.22 the Clustering configuration interface is shown. By means of this interface the user can change the parameters used by the Ontology Builder to compute the clusters (and the global classes). The user can change this parameter in order to obtain a global classes set that completely represent the local sources.

In order to build the global classes, you have to select “Cluster Tuning”, (blue arrow).

The interface (see fig.23) shows on the left the sources tree, and on the right the global classes tree (which is initially empty).

On the top of the interface there is the button “CREATE CLUSTERS”, that has to be pressed to build the global classes.

If the global classes have already been calculated, an alert asks for confirmation from the user to delete the global class or to preserve the old situation. The result is shown in fig. 23

The graphical interface allows the user to do further operations:

- “Delete Global Class”: to delete a selected global class.
- “Add Global Class”: to create a new global class.
- “Unmap Local Class”: to remove a local class from a specific global class.
- “Map Local Class”: to manually select a global class for a specific local class. In order to have a good result we prefer to change the parameters (see fig 22) and redo the cluster operation.
- “Delete All Clusters”: to delete all the global classes.

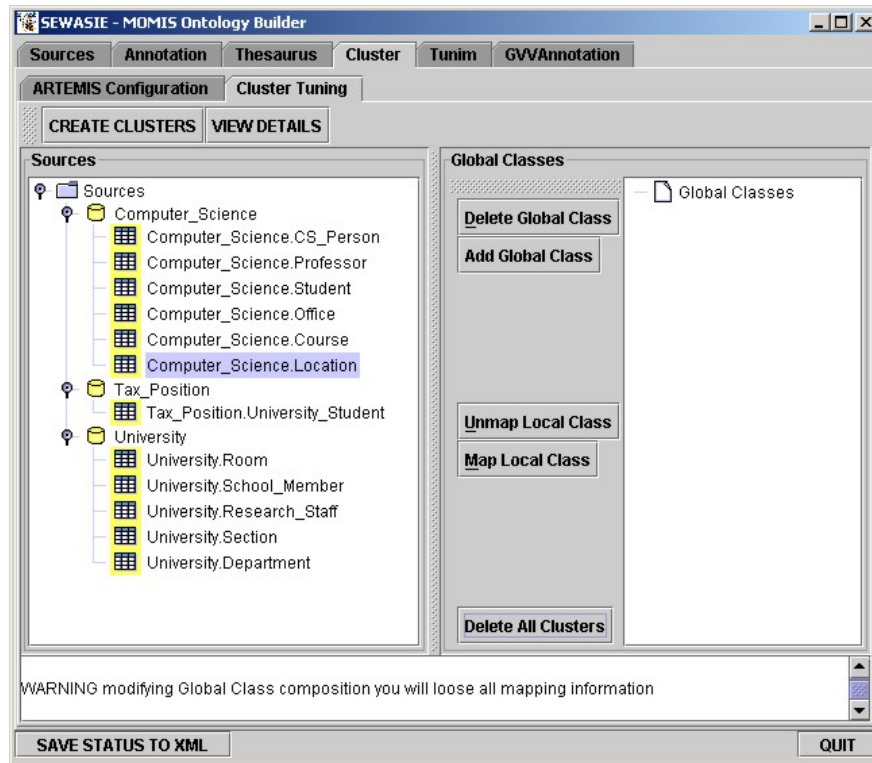


fig. 22 – Global Classes creation

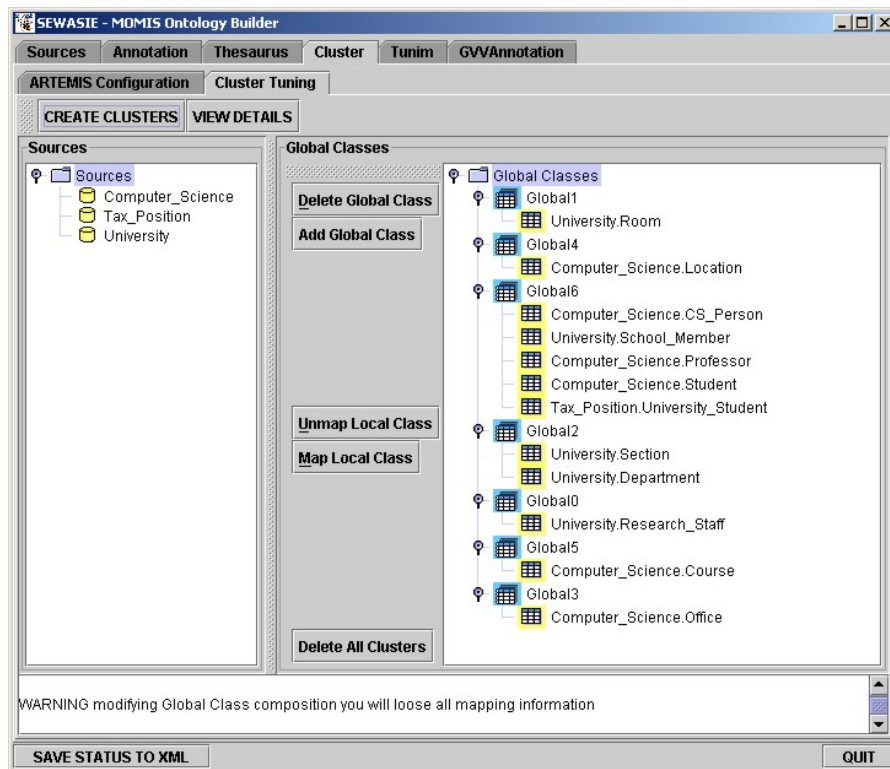




fig. 23 – Global Classes created

The Ontology Builder automatically generates a name for each global class. This name will be automatically modified in the next step “*Mapping*”.

Global Attributes and Mapping.

The “*Mapping*” phase allows the user to visualize and manage the global attributes of each global class created in the previous step “*Cluster*”.

The window (fig. 25) initially shows on the left the global classes tree or – “*Global Interfaces*” (icon ) and for each one the list of local classes belonging to it (icon )

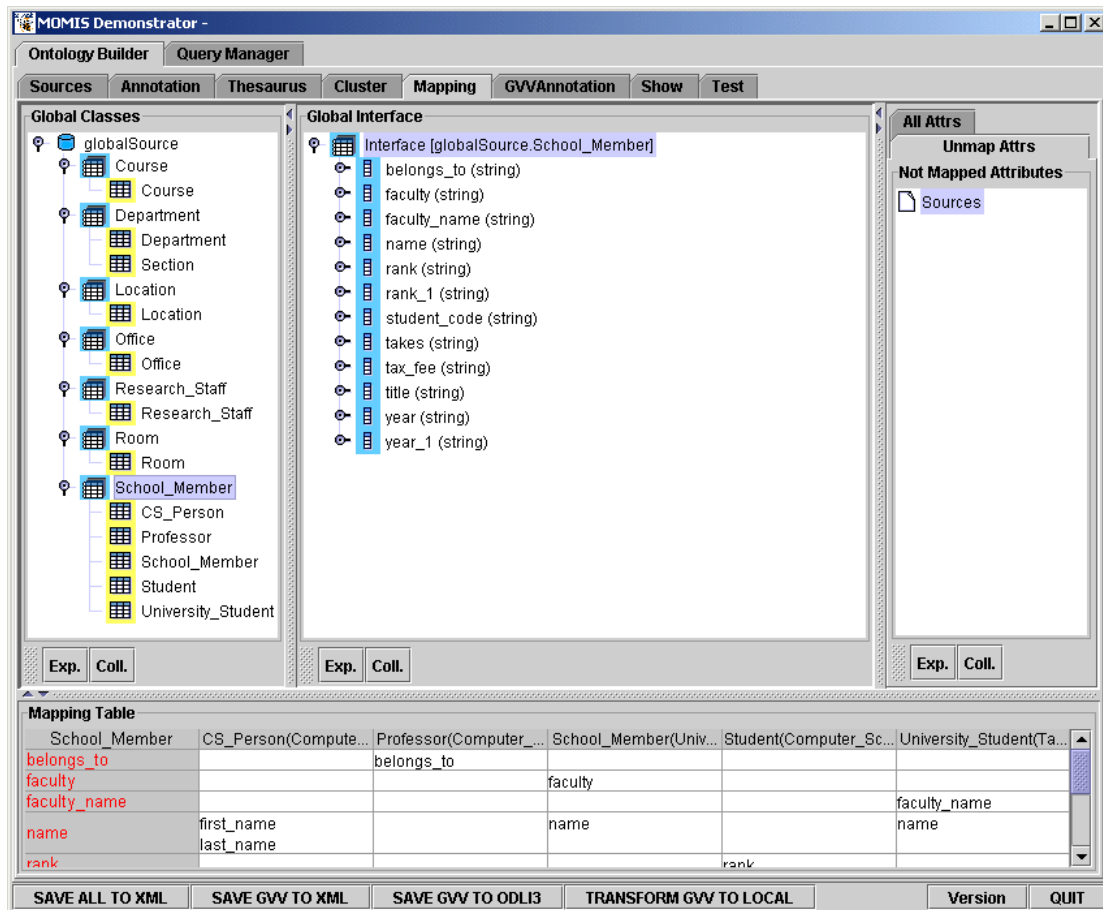




fig. 25 – Mapping interface

You can see in fig. 25 that the name of the global classes have changed since the previous step (fig. 24). The Ontology Builder provides a name for each global class calculated on the basis of the local classes (the Ontology Builder suggests the most general name).

It is possible to modify the global class name by means of the right button of the mouse (see Fig. 25). By selecting the option “*Rename Global Interface*” a pop-up menu appears where the user can insert the new name.

By selecting a node or a leaf of the global classes tree (on the left), in the central part of the interface the contents of the selected global class is shown. In particular, the user can visualize and manage the global attributes () and by opening each node, the local attributes the are mapped on it ()

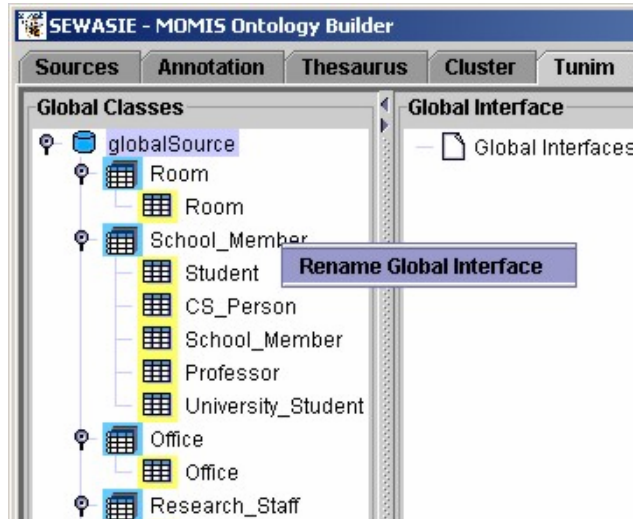


fig. 25– Global class renaming

When a global class is chosen, the corresponding “*Mapping Table*” is selected in the lower panel of the window. The Mapping Table shows how local attributes are “mapped” into a global attribute.

The leftmost column of the “*Mapping Table*” represents the list of all the global attributes (red square), the first row represents all the local classes belonging to the global class (blue square); the table elements are the local attributes (that are part of a local class) “mapped” in a specific global attribute (row). More attributes could be mapped by the same global attributes (see green circle).

Mapping Table					
School_Member	Student(Computer_Sci...	CS_Person(Computer...	School_Member(Unive...	Professor(Computer_...	University_Student(Tax...
faculty			faculty		
rank_1				rank	
year			year		
title				title	
rank	rank				
belongs_to				belongs_to	
student_code					student_code
year_1	year				
tax_fee					tax_fee
takes	takes				
faculty_name					faculty_name
name		first_name	name		name
		last_name			

fig. 26 - Mapping Table

Local Resolution function

A local resolution function is applied to each local attribute mapped by the mapping table. The simplest local resolution function is the identity (default). By clicking twice on a mapped local attribute a new window appears.

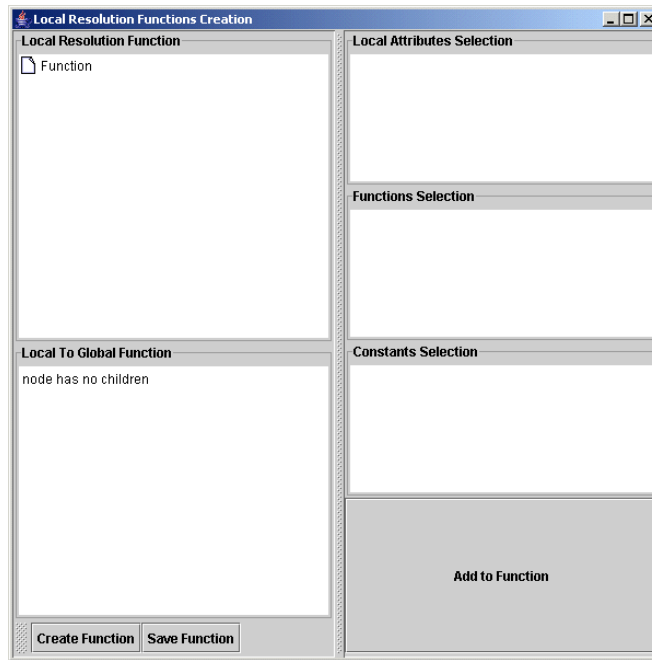


fig. 27.A – Local resolution function

This window is initially empty; the Local Resolution Function area will show the created function, Local attributes Selection area will show the involved attributes, the Function Selection area will show the functions applicable to the chosen attributes, the Constants Selection area allows the user to insert some constants.

By clicking on “*function*” in the Local Resolution Function area, the interface will be initialized.

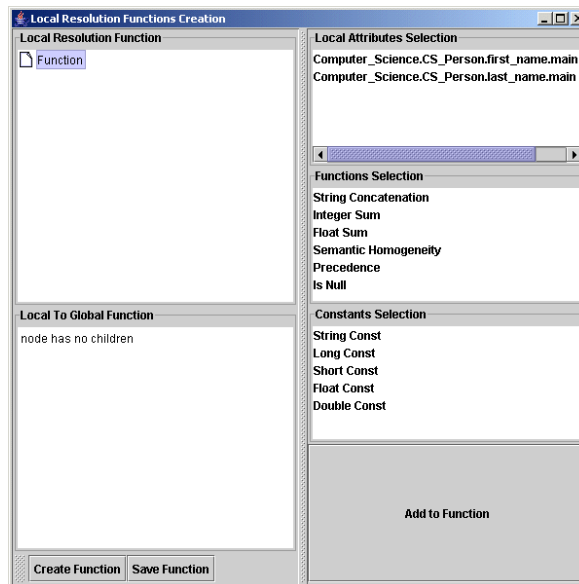


fig. 27.B – Local resolution function

The user has to select the function, press the “*Add to Function*” button and then choose the involved attributes (each selection has to be followed by a click on the “*Add to Function*” button).

Finally the user has to press the “*Create Function*” button to visualize and check the created function and then the “*Save Function*” button to save his work. An example of result is shown in the following figure where a concat (+) function is applied to “*first_name*” and “*last_name*” attribute values of the Person class in the Computer Science source to obtain the value of the global attribute “*name*”

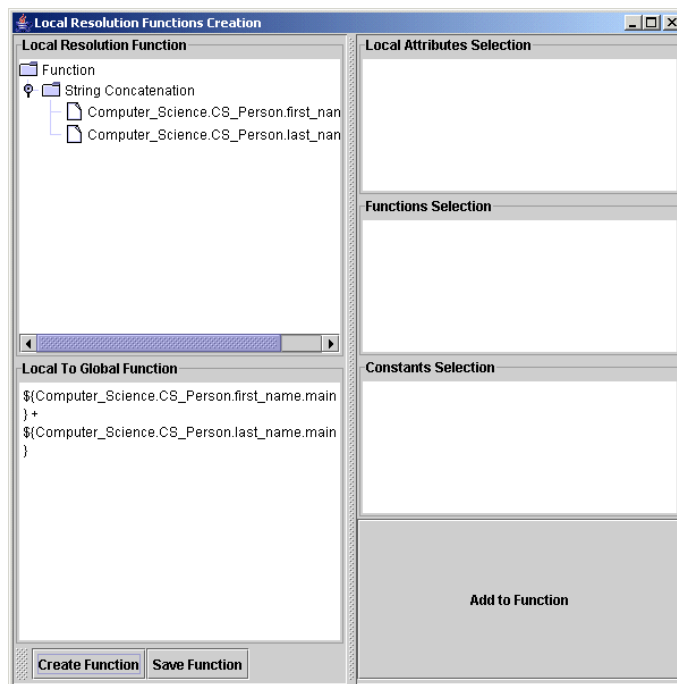



fig. 27.C – Local resolution function

Global Resolution function

The Global Attribute value is obtained by applying the Global Resolution Function to the values obtained by the local to global function. By clicking twice on a global attribute a new window to set the Global Resolution function appears. The interface working is the same of the Local Resolution Function.

The central part of the window allows the user to manage the global attributes. By selecting a global attribute (icon ) and by pressing the right button of the mouse a new menu is shown (see Fig. 28) with the following options:

- **“Rename Global Attribute”**: the user can change the name automatically provided by the OB exploiting the annotations and suggesting the most general term.
- **“Add New Global Attribute”**: a new global attribute can be added to a global class.
- **“Remove This Global Attribute”**: a global attribute can be removed. The previously local mapped attributes are moved in the “not mapped” attributes interface.

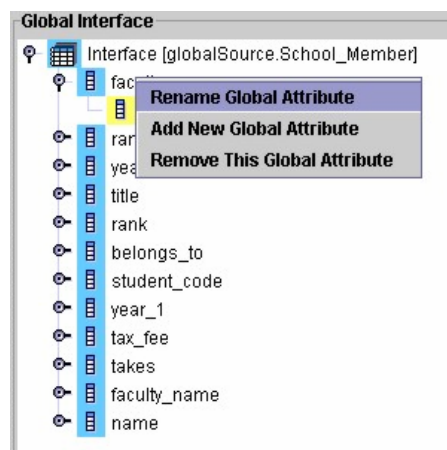


fig. 28– Global Attribute management

By means of this panel it is possible to manage the local attributes, by pressing the right button of the mouse the user can (see Fig 29):

- “**Unmap Local Attribute**” : the selected local attribute will be removed and will be inserted in the not mapped items tree on the right. The same result can be obtained by means of a drag and drop operation.
- “**Add New Global Attribute**” : this option allows the user to add a new global attribute.

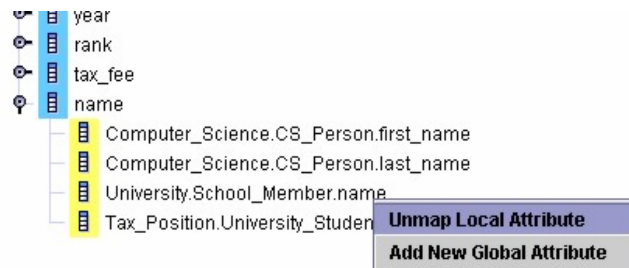


fig. 29 – Local Attribute management

If there are not mapped attributes and the user is passing to the next step, the Ontology Builder shows a warning (see Fig 32).

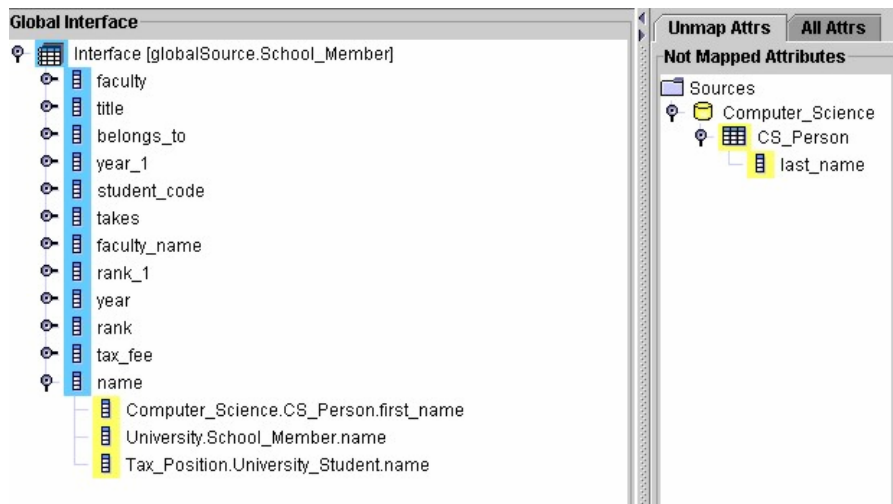


fig. 30 - Not mapped attributes

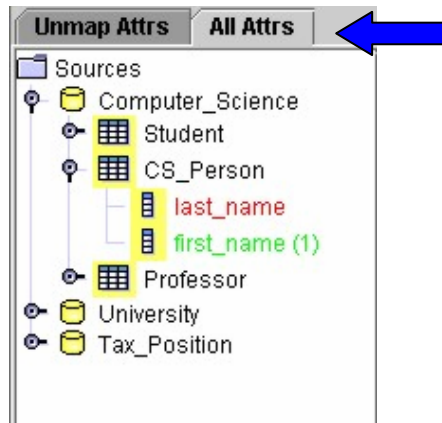


fig. 31 –Sources tree



fig. 32 – Warning of not mapped attributes

The Global Virtual View annotation (GVVAnnotation).

The last step concerns the global virtual view annotation.

The goal of this step is to improve the semantics of a GVV in order to provide a sharable domain ontology.

The Ontology Builder automatically calculates the GVV annotation by exploiting the meanings of the local classes.

The user can navigate the tree (see Fig.33) and modify the proposed annotation, see section “Sources Annotation“ for more information.

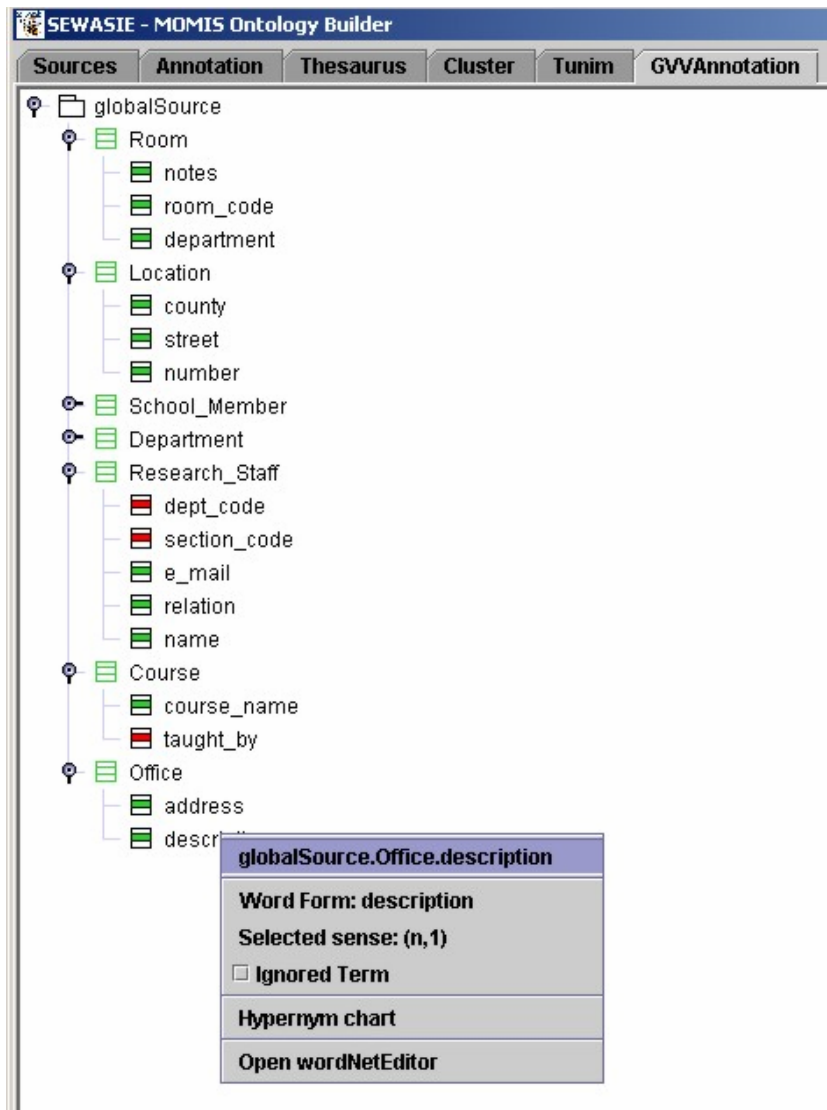


fig. 33 – GVV annotation

The “Show” panel.

It is possible to graphically visualize the local classes and the obtained GVV.

The left part of the page shows by means of a tree interface the sources. The buttons on the bottom allow to switch from the local source view to the GVV view.

The right part shows, graphically, the sources. The first button on the bottom allows the user to select a compact visualization (only the classes name are shown) or an extended visualization (classes and attributes name are shown).

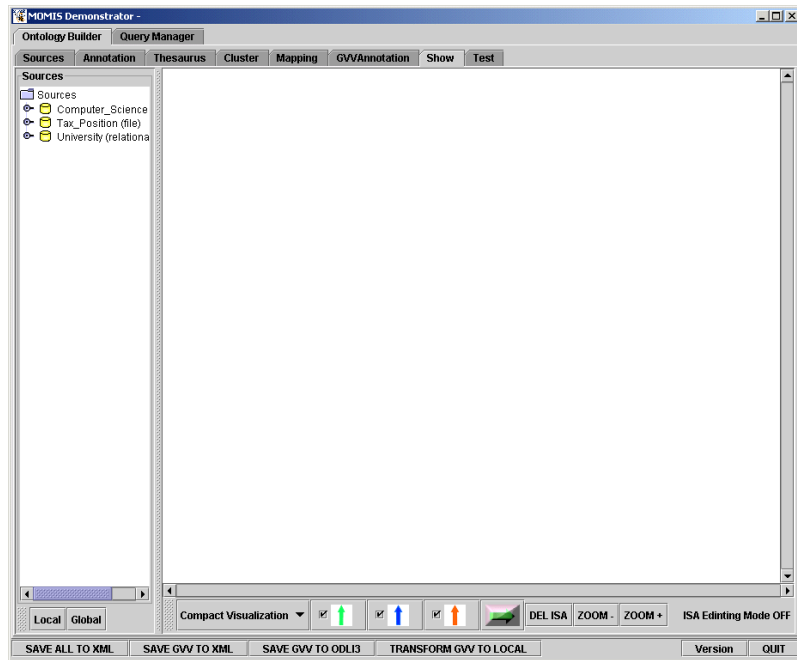


fig. 34 – The show panel

Next figure shows the visualization of a GVV.

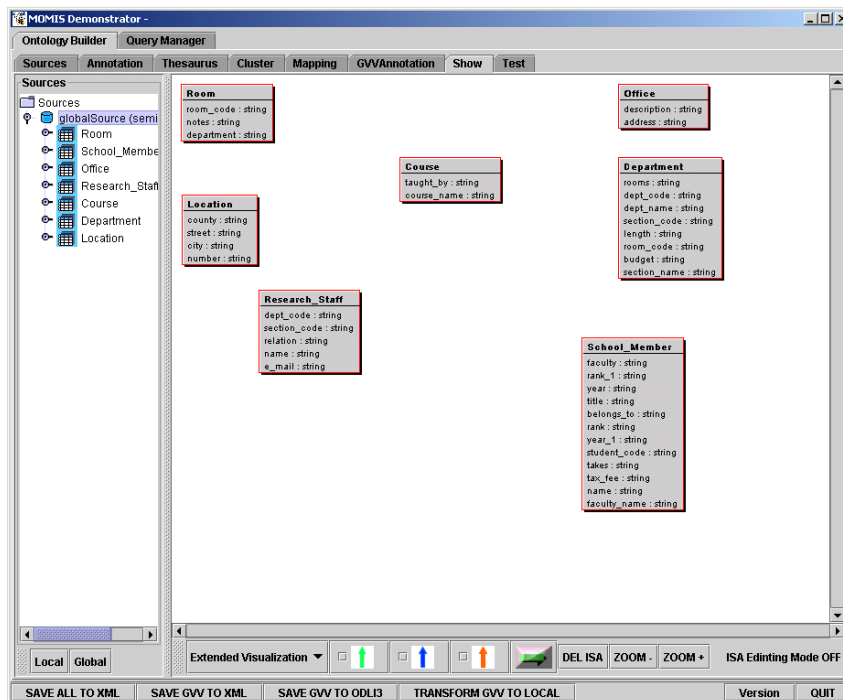


fig. 35 – An example of a GVV visualization

The Query Manager.

When you select the Query Manager module from the MOMIS demonstrator an empty page appears. The first operation is to load a GVV. Click with the right button on the empty screen. A new menu appears:



fig. 36 – The Query Manager: first operation

If you select “*New Query Manager*” and then “*Load Schema from Ontology Builder*” you can query the GVV created with the Ontology Builder, otherwise you can query a previously created GVV by loading it from a file. All the local sources have to be available in order to start the Query Manager.

The Query Manager interface allows the user to query a single global class.

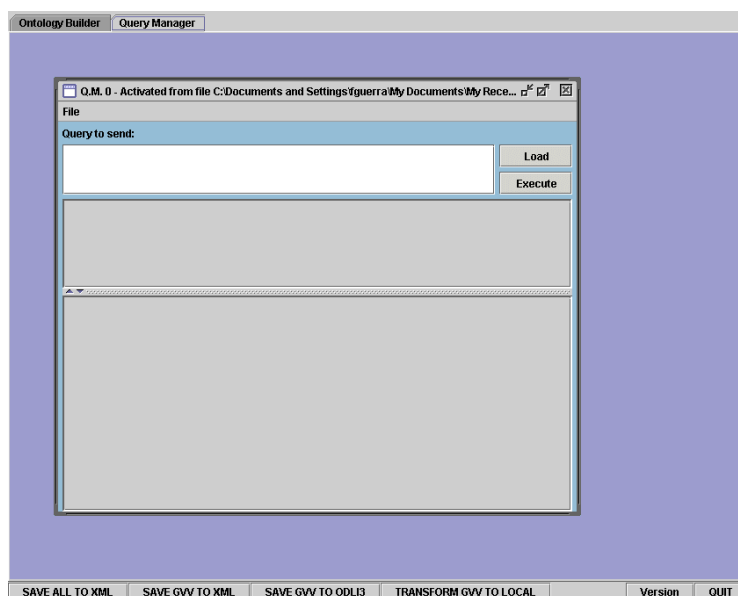


fig. 37 – The Query Manager: the interface

To show the Query Manager functionalities, we load a previously created GVV of three sources (named “usawear”, “fibre2fashion” and “tessilmoda”), containing real data related to textile enterprises (www.usawear.org, www.fibre2fashion.com, www.tessilmoda.com).

By choosing “Show Global Schema” in the File menu, we can see the GVV and the configuration parameters of the local integrated sources (see fig.)

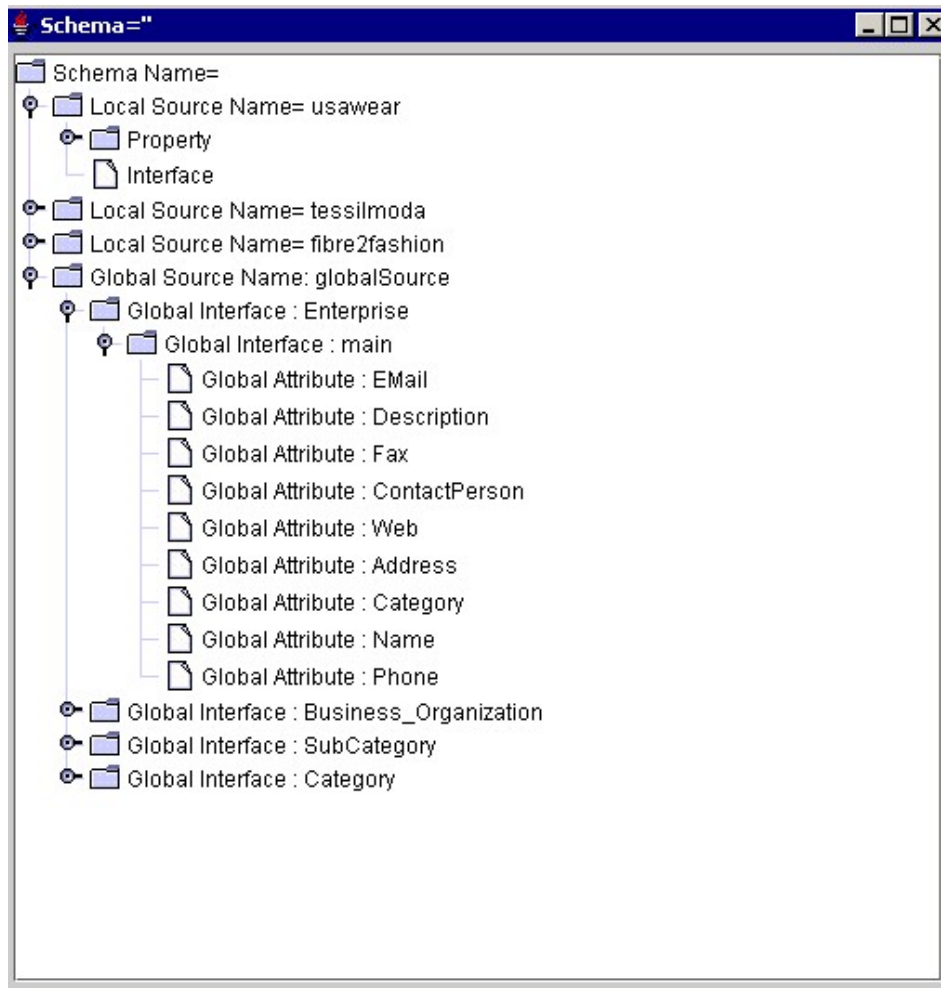
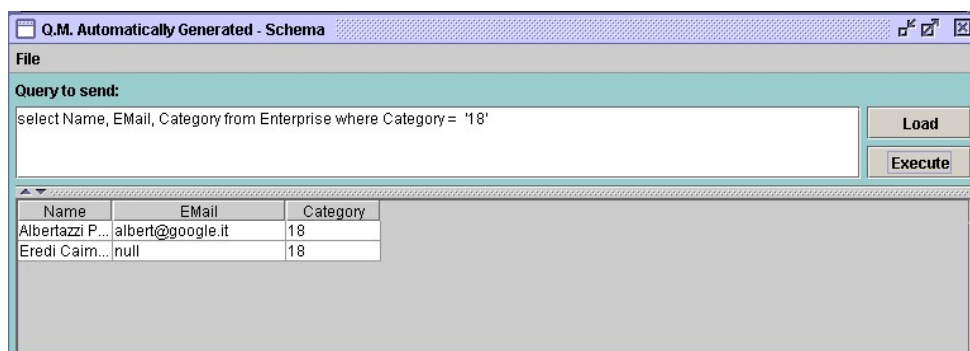


fig. 38 – The Query Manager: the schema view

In the “Query to send” window we can write our query; for example:

Q1 : select Name, EMail, Category from Enterprise where Category = '18'



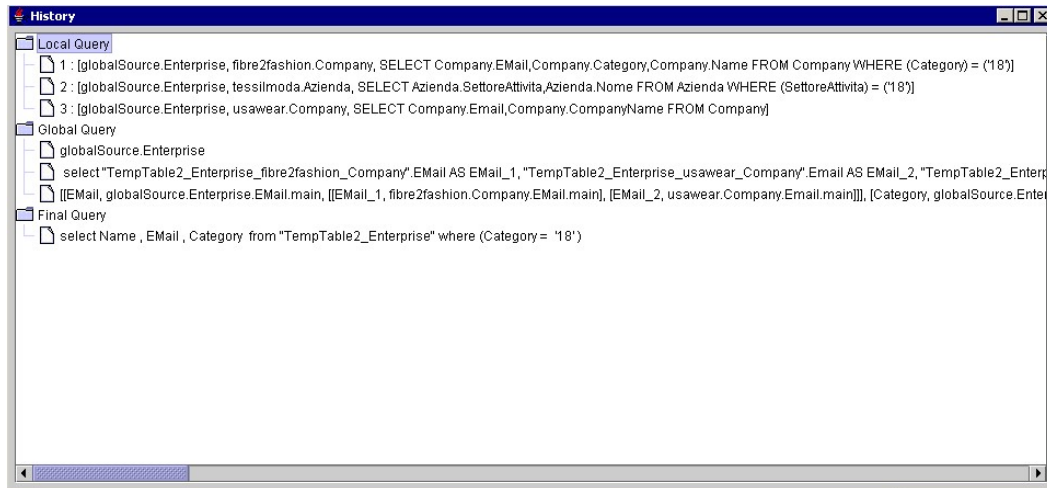
The syntax of a query is

```
Q = select <Q_select-list> from G where <Q_condition>
```

where <Q_condition> is a Boolean expression of positive atomic constraints: (GA1 op value) or (GA1 op GA2), where GA1 and GA2 are attributes of the global class G and op is a relational operator and value is a constant value (all constant values are denoted in quotation mark: '12', 'house', ..).

With “Execute” we execute this query; its result is shown in the main window, above.

By choosing “Show Last History Schema” in the File menu, we can see the main steps of the query rewriting with respect to local classes.



First of all, we can see the Local Queries:

```
1 : [globalSource.Enterprise, fibre2fashion.Company, SELECT
Company.Email,Company.Category,Company.Name FROM Company WHERE (Category) =
('18')]
```

```
2 : [globalSource.Enterprise, tessilmoda.Azienda, SELECT
Azienda.SettoreAttivita,Azienda.Nome FROM Azienda WHERE (SettoreAttivita) =
('18')]
```

```
3 : [globalSource.Enterprise, usawear.Company, SELECT
Company.Email,Company.CompanyName FROM Company]
```

In particular, we can observe, that the constraint `Category = '18'` is mapped into

- 1) `(Category) = ('18')`, for the local class `fibre2fashion.Company`
- 2) `(SettoreAttivita) = ('18')`, for the local class `tessilmoda.Azienda`
- 3) `true`, for the local class `usawear.Company` as `Category` is not mapped into this local class.

The answers of these local queries are stored in temporary tables, such as `TempTable2_Enterprise_fibre2fashion_Company`.

The second entry “Global Query” shows the query that performs the “full disjunction” operation:

```
select "TempTable2_Enterprise_fibre2fashion_Company".EMail AS EMail_1,
"TempTable2_Enterprise_usawear_Company".Email AS EMail_2,
"TempTable2_Enterprise_fibre2fashion_Company".Category AS Category_1,
"TempTable2_Enterprise_tessilmoda_Azienda".SettoreAttivita AS
Category_2, "TempTable2_Enterprise_fibre2fashion_Company".Name AS
Name_1, "TempTable2_Enterprise_tessilmoda_Azienda".Nome AS Name_2,
"TempTable2_Enterprise_usawear_Company".CompanyName AS Name_3

from "TempTable2_Enterprise_fibre2fashion_Company" full outer
join "TempTable2_Enterprise_tessilmoda_Azienda" on
```

```

((( "TempTable2_Enterprise_tessilmoda_Azienda".Nome) =
("TempTable2_Enterprise_fibre2fashion_Company".Name))) full
outer join "TempTable2_Enterprise_usawear_Company" on
((( "TempTable2_Enterprise_usawear_Company".CompanyName) =
("TempTable2_Enterprise_fibre2fashion_Company".Name)) OR
("TempTable2_Enterprise_usawear_Company".CompanyName) =
("TempTable2_Enterprise_tessilmoda_Azienda".Nome)))

```

The last entry is the “Final Query” :

```

select Name , EMail , Category from "TempTable2_Enterprise" where
(Category = '18' )

```

where:

- TempTable2_Enterprise is obtained starting from the answer of Global Query (which performs full disjunction) and by applying the resolution functions;
- (Category = '18') is the residual constraint

The execution of the “Final Query” gives the result of the user query.