S.Bergamaschi, D.Beneventano, S.Quattrini, E.Kazazi

User Guide
MOMIS Version 1.2

Contents

List of Figures

Introduction

This User Guide helps you learning how to integrate heterogeneous and distributed sources with the MOMIS Data Integration System.

The User Guide is organized as follows: Section 1 presents the MOMIS system, by describing its architecture and and by identifying the main phases of the data integration process; in Section 2, the MOMIS GUI and the Global Schema generation process are explained with images and examples; Section 3 explains how to query the obtained Global Schema and finally Section 4 explains in details how to use the MOMIS Query Manager Web Interface.

If you are looking for a quicker MOMIS guide, have a look at the MOMIS Video Tutorials, available on the DataRiver YouTube channel (http://www.youtube.com/user/DataRiverSrl) or on the Datariver Web Site (http://www.datariver.it).
You can find all MOMIS publications on the DBGroup publications page: http://www.dbgroup.unimo.it/site2012/index.php/publications

Chapter 1

The MOMIS Data Integration System

1.1 MOMIS Overview

"Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data" [6].

The MOMIS Data Integration System (Mediator envirOnment for Multiple Information Sources) is a framework able to integrate data coming from heterogeneous and distributed data sources (structured and semi-structured) in a semi-automatic way, to bring out new information from apparently unrelated existing data. An object-oriented language, with an underlying Description Logic, called ODL_{I^3}, derived from the standard ODMG is introduced for information extraction. A software component, named wrapper (Fig. 1.1) extracts the schema of a source, translating it in the ODL_{I^3}language. The discovery of relationships between the schemas of the information sources exploits the semantics of the data sources, clustering techniques and description logics inferences. The integration process gives rise to a mediated schema, also called Global Schema (GS) (Fig. 1.1) that is a reconciled, integrated, and virtual view of the underlying sources. MOMIS performs data integration following a virtual approach, no centralized copy of data is made and follows a global-as-view (GAV) approach: the obtained global schema is expressed in terms of the local source schemas. A full and detailed description of the MOMIS system from a theoretical point of view is out of our scope and can be found in [5, 2, 4].

The goal of the MOMIS system [3] is the minimization of the integration process costs. In traditional Data Integration Systems, designers have to manually build the integrated schema, defining all the mappings between

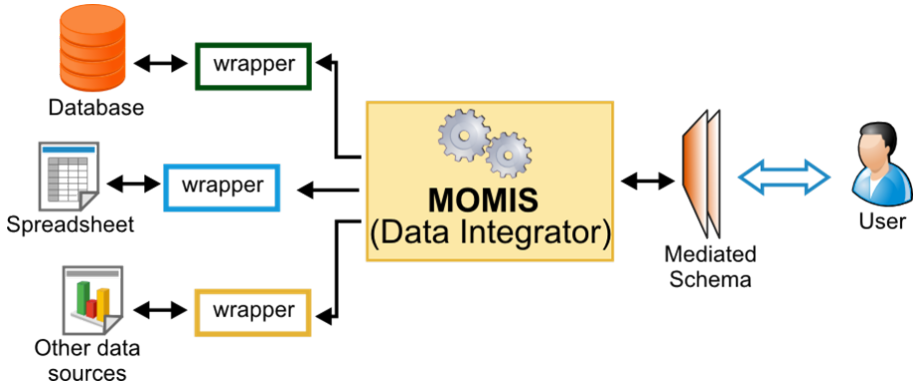


Fig. 1.1: The Data Integration Process

each global class/attribute and the corresponding local classes/attributes on the local data sources, thus the integration process requires several days or weeks depending on the size of the integration project. Another drawback is due to the fact that designers can see the global result of the integration only at the end of the overall integration process, and it is only at that time that they can refine mappings in order to improve the integrated schema. To overcome these problems, a first result of integration is semi-automatically derived by MOMIS and proposed to the designer in few minutes; she/he can then improve this integration result, through an iterative refinement process and a set of features. Some of these features are listed below:

- a GUI that facilitates the integration process
- a set of explore and preview tools that allow the designer to preview the integration result during each phase
- the possibility to create different unified views to explore the global result of the data integration process
- a suite of tools to semantically annotate data sources w.r.t. a common lexical reference; these tools allow the designer to import/export the local source annotations, and permit to extend the lexical reference itself with domain glossaries
- a preview of the query plan that allows the designer to visualize, for each executed global query, the set of queries that compose the query plan

1.2 Data Integration Process and MOMIS Architecture

MOMIS builds a unified schema, called Global Schema (GS), of several (heterogeneous) data sources (also called local sources), and allows users to formulate queries on it. As reported in the previous section it follows a Global-As-View (GAV) approach for the definition of mappings between the GS and local schemas: the GS is expressed in terms of the local schemas. The GS generation process is composed by four main phases:

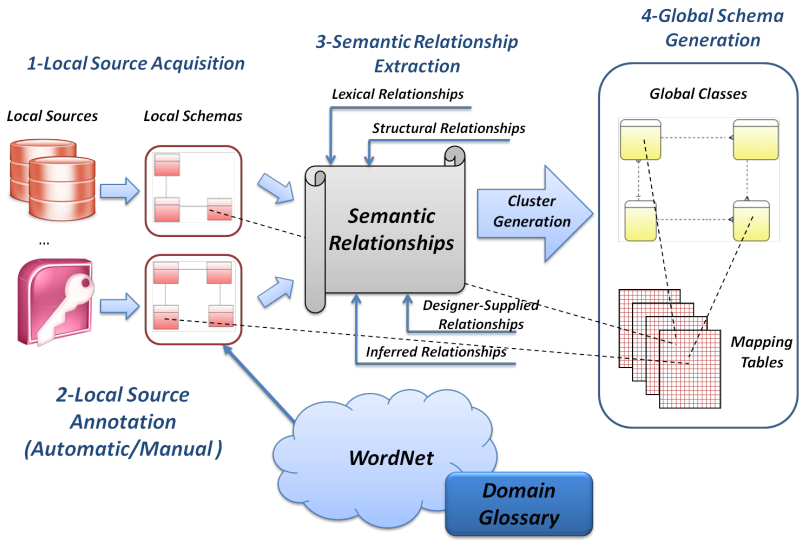


Fig. 1.2: The MOMIS Data Integration Process

1. **Local Schema Acquisition:** (Fig. 1.2-1) The extraction of the Local Source Schema is performed by *wrappers*[1], that logically extract the schema of each local source and convert it into the common language ODL_{I^3}.

2. **Local Source Annotation:** (Fig. 1.2-2) the designer is asked to *annotate* the local sources, i.e. to associate to each class and attribute name one or more meanings w.r.t. a common lexical reference, that in this case is the lexical database WordNet [7]. WordNet is a thesaurus for the English language, that groups terms (called lemmas in the WordNet terminology) into sets of synonyms called synsets, provides short definitions (called gloss), and connects the synsets through a wide network of semantic relationships.

   The designer can perform automatic annotation and/or can manually select a base form and the appropriate WordNet meaning(s) (i.e. synset(s)) for each term. Moreover, the designer can extend WordNet with Domain Glossaries. The Local Source Annotation phase is performed by the *Global Schema Designer* tool (Fig. 1.4).

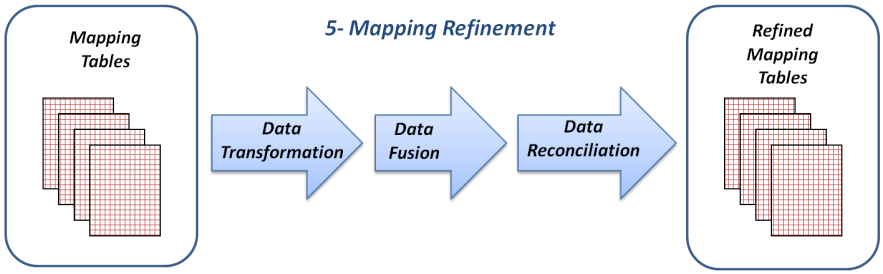


Fig. 1.3: Mapping Refinement

3. **Semantic Relationships Extraction:** (Fig. 1.2-3) starting from the annotated local schemas, MOMIS derives a set of intra and inter-schema semantic relationships in the form of: synonyms (SYN), broader terms/narrower terms (BT/NT) and related terms (RT) relationships. The set of semantic relationships is incrementally built by adding: *structural* relationships (deriving from the structure of each schema), *lexical* relationships (deriving from the element annotations, by exploiting the WordNet semantic network), *designer-supplied* relationships (representing specific domain knowledge) and *inferred* relation-

[1]The type of sources supported at present are MySQL, Microsoft SQL Server, Oracle, DB2, PostgreSQL, JDBC Sources, JDBC-ODBC Sources, Microsoft Excel File, CSV File, Web Service.

ships (deriving from Description Logics equivalence and subsumption computation). The Semantic Relationship Extraction phase is performed by the Global Schema Designer tool.

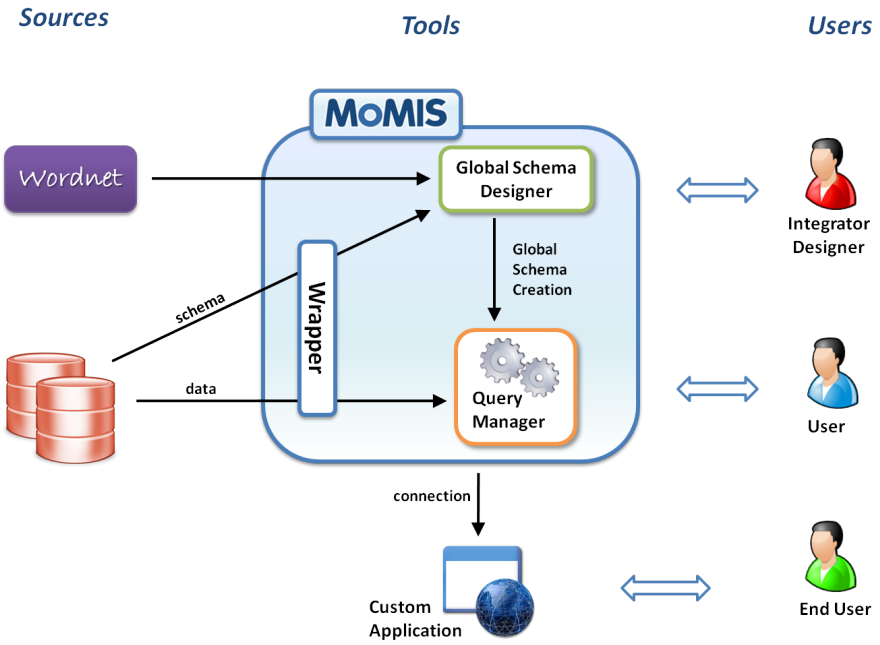


Fig. 1.4: The MOMIS architecture

4. **GS generation:** starting from the discovered semantic relationships and the local sources schemas, MOMIS generates a GS consisting of a set of global classes, plus *Mapping Tables* which contain the mappings to connect the global attributes of each global class with the local source attributes. The GS generation is a process where classes describing the same or semantically related concepts in different sources are identified and clusterized into the same global class (Fig. 1.2-4). The designer may interactively refine and complete the proposed integration result through the GUI provided by the Global Schema Designer tool. In particular, she/he can: modify the proposed global classes and mappings (Fig. 1.3); select the appropriate *Join Function* for each global class; define *Transformation Functions* in order to transform the local attribute values into the corresponding global attribute values; and solve possible data conflicts through the definition of *Resolution Functions* (applied to each global attribute to obtain, starting from the values computed by the Transformation Functions the corresponding value of the global attribute).

Finally, once obtained the desired integration result, a user can pose queries on the GS by using the *Query Manager* tool (see Fig. 1.4). As MOMIS follows a GAV approach, the query processing is performed by means of query unfolding. The query unfolding process generates for each global query (i.e. a query on the GS) a Query Plan composed by a set of queries:

- a set of local queries that have to be executed on the local sources simultaneously by means of wrappers,
- a mapping query for merging the partial results (defined by means of the join function),
- a final query to apply the resolution functions and residual clauses.

Moreover, MOMIS provides *the Query Manager Web Service* which allows to integrate MOMIS with other applications (e.g. Business Intelligence solutions), and a user-friendly *Web Application* (Fig. 1.4) to guide an end-user, without experience on data integration solutions, to easily compose and execute query on the integrated schema (see Chapter 4).

1.3 The WISDOM case study

In this user guide we used an integration project that involves a set of local sources representing Hotels and Campings, the data sources has been extracted from three popular Websites: www.venere.com/it, www.saperviaggiare.it and www.guidacampeggi.com.

These sources have been used in the WISDOM project (Web Intelligent Search based on DOMain ontologies) (www.dbgroup.unimo.it/wisdom) for the development of a Tourism Vertical Web Portal (Fig. 1.5).

Fig. 1.5: The WISDOM Project dataset

1.4 Important concepts in MOMIS

As reported in the Section 1.1, data integration is the problem of combining data residing at different Local Sources, and providing the user with a unified view of these data. These unified view is called Global Schema. Different Global Schemas may be created starting from the same set of Local Sources.
Project
A Project is a set of Local Sources and a set of Global Schemas. Projects will be described in Section 2.3. **Global Schema** A Global Schema is a set of global classes and relationships among them; on the other, in a Project, we use the term Global Schema to denote not only classes and relationships among them, but also a set of information: local sources and for each local source the local classes and local attributes involved in the integration, local sources annotation, semantic relationships, mappings between global classes and local classes etc. All these informations related to a Global Schema are stored in an xml file.

A **workspace** is the directory where you store all your project folders. You need to have one workspace directory.

You can create as many projects as you need and they are saved in the workspace.

1.5 Glossary

In this tutorial some abbreviations will be used:

- GS stands for Global Schema;
- GC stands for Global Class;
- GA stands for Global Attribute;
- LC stands for Local Class;
- LA stands for Local Attribute;
- TF stands for Transformation Function;
- RF stands for Resolution Function;
- JF stands for Join Function;
- *Label* is the name of an attribute or a class;
- *Sense* is a meaning of a label.

Chapter 2

The MOMIS System

2.1 Launching MOMIS

MOMIS, is a Java based application and as such a Java Runtime Environment (JRE) is required in order to run it. It is developed as an Eclipse Rich Client Platform[1] (RCP) application, so no installation is required. To tun MOMIS complete the following steps:

- Download[2] the right zip folder corresponding to your operating system.
- Unzip the MOMIS zip file and, in the folder, double-click the executable file.
- A *License* window displays. Read and accept the terms of the license agreement to continue[3] .

That is all, now you can start integrating sources and querying them.

[1] http://wiki.eclipse.org/Rich_Client_Platform

[2] You can dowload the zip folder from **http://www.datariver.it -> Download** menu

[3] The *License* displays only the first time you run MOMIS, you will be asked to accept the MOMIS License GPL Version 2, if you don't accept the license the application will not start and will exit

2.2 Getting Started with MOMIS

The MOMIS Main Window

MOMIS starts presenting a Welcome Page (Fig. 2.1), in which you can find some information about Data Integration and the MOMIS Application. If you click on *MOMIS Demo Project* the Welcome Page will close and the project used in this user guide will load in the workspace.

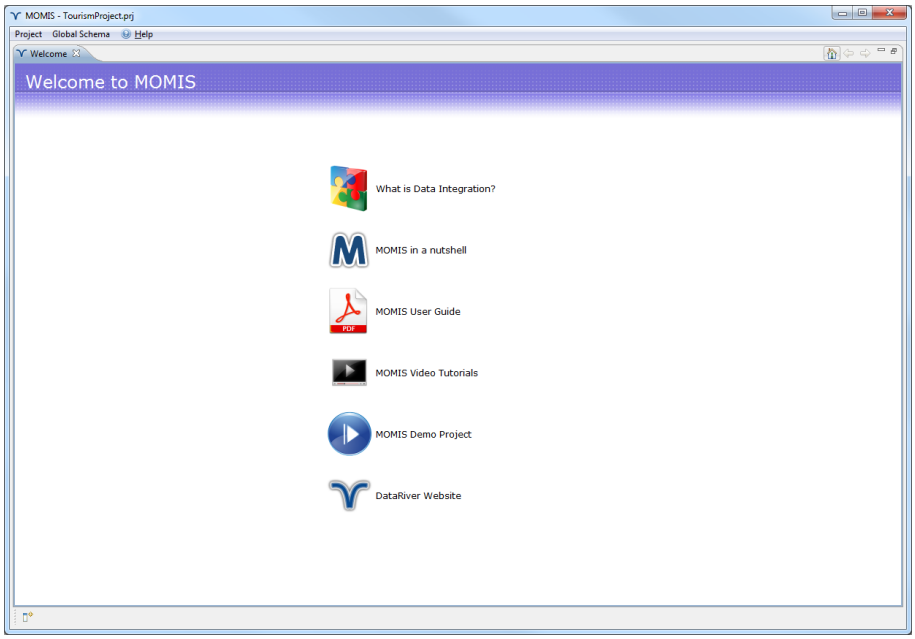


Fig. 2.1: MOMIS Welcome Page

MOMIS main window is divided into 3 views (Fig. 2.2):

1. *Source Explorer*, where are visualized the sources of the current project;
2. *Global Schema Explorer*, where are visualized the global schemas of the current project;
3. *Editor Panel*, where you will be able to edit the global schemas of the current project.

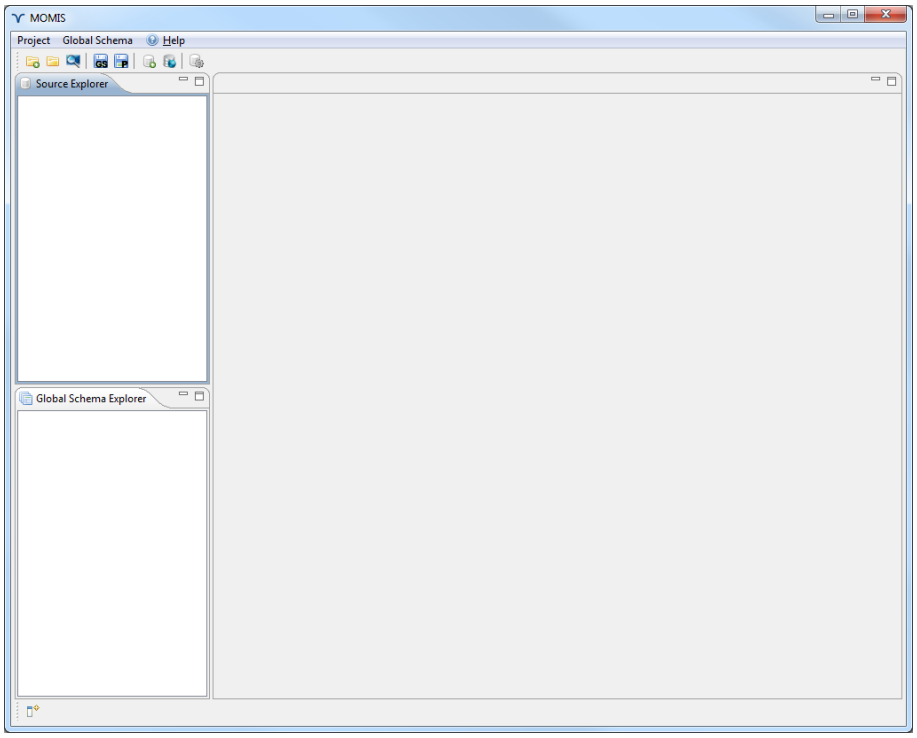


Fig. 2.2: MOMIS Main Window

Menu bar and Toolbar

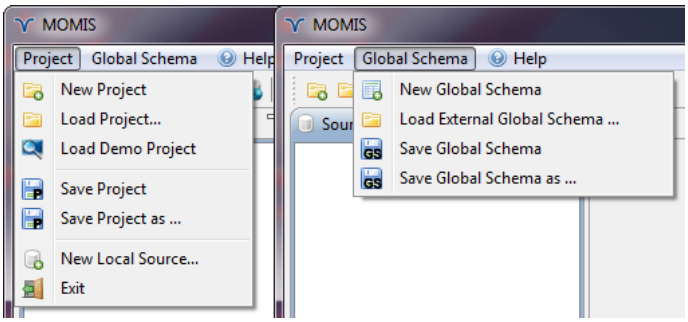


Fig. 2.3: Menu bar

The menu bar is composed of three menus (Fig. 2.3):

1. *Project*: is composed of a set of menu items that enables the acquisition of a local source schema, permits you to create a new integration project, upload an existing one, upload the demo project or save the current project.

2. *Global Schema*: is composed of a set of menu items that permits you to create a new global schema, upload an existing one or save the current global schema.

3. *Help*: is composed of a set of menu items that permits you to open the Welcome Page and the About Dialog.

In the tables below are listed the menu items and the toolbar items available to you.

Table 2.1: Menu

Icon	Menu Item	Description
	New Project	Create a new empty project
	Load Project	Load a previously created project
	Load Demo Project	Load the project used in this user guide
	Save Project	Save the current project
	Save Project as ...	Create a copy of the current project
	New Local Source ...	Upload the schema of a new source
	Exit	Exit the application

Table 2.2: Global Schema Menu

Icon	Menu Item	Description
	New Global Schema	Create a new global schema in the current project
	Save Global Schema	Save the current global schema
	Save Global Schema as ...	Create a copy of the current global schema

Table 2.3: Toolbar Items

Toolbar Icon	Description
	Create a new empty project
	Load a previously created project
	Load the demo project used in this user guide
	Save the current global schema
	Save the current project
	Upload the schema of a new source
	Upload the schema of a new source through web service
	Launch Query Manager on the current schema

Project Generation Process

A Project is a set of Local Sources and a set of Global Schemas; the main steps of the Project Generation Process are:

1. Local Source Schema Extraction (Section **??**)
2. Global Schema Generation (Section 2.5)

Once completed the integration process, it is possible to pose query on the obtained Global Schema by using:

- the Query Manager Interface
- the MOMIS QM Web Interface

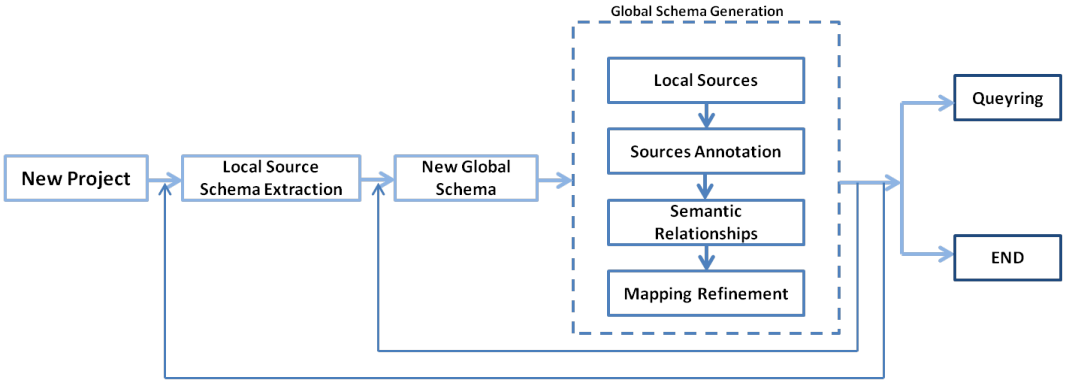


Fig. 2.4: Project Generation Process

2.3 How to Create a new Project?

Click *Project -> New Project* and in the displayed dialog box (Fig. 2.5) enter a name for your project, click *OK*. The system will open the *Local Source Schema Acquisition* wizard (see Section 2.4).

2.4 Local Source Schema Extraction

The extraction of the local source schema is performed by *wrappers*. A wrapper is a software component that logically converts the source data structure

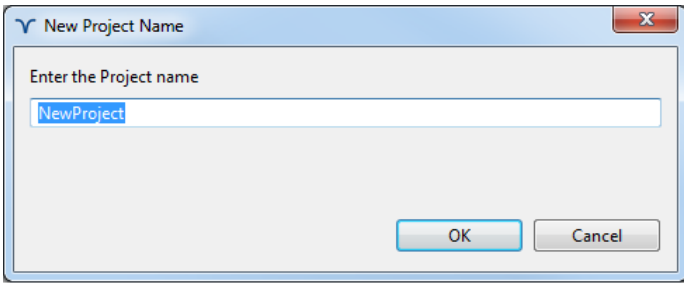


Fig. 2.5: New Project dialog

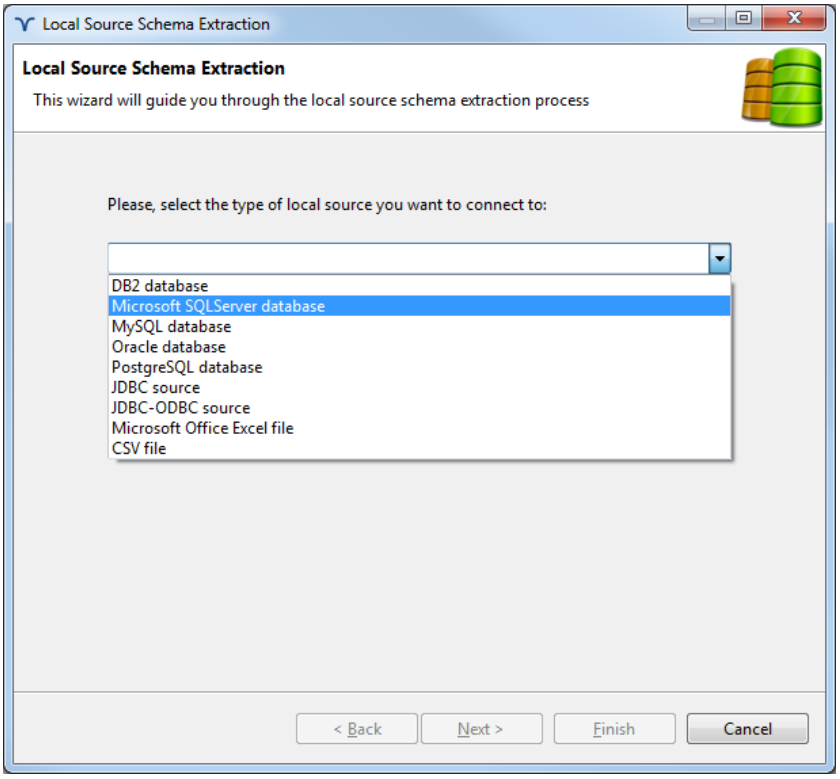


Fig. 2.6: Local Source Type Selection

into the ODL$_{I^3}$data language[4]. Let's see how to extract the local source schema. Click *Project -> New Local Source* and a wizard that will guide you, in the local source schema acquisition process, will start. Choose from the list the local source type (Fig. 2.6), and enter a name in the text field (the name entered should be unique within a project). Depending on the source type selected, the next wizard page can be different. Let us see how to extract the schema of a Microsoft SQL Server database, a Microsoft Excel file and a MySQL Database through Web Service.

2.4.1 Microsoft SQL Server database Schema Extraction

In the first wizard page select *Microsoft SQL Server database* as source type and enter a name.

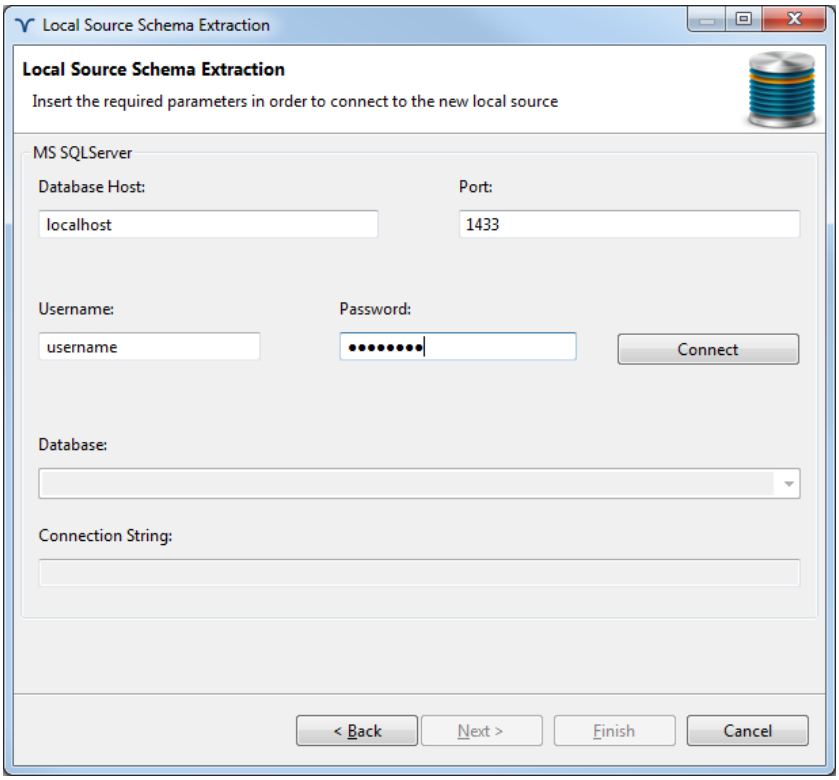


Fig. 2.7: Microsoft SQL Server Connection Parameters

[4]The type of sources supported in this version are: MySQL, Microsoft SQL Server, Oracle, DB2, PostgreSQL, JDBC Sources, JDBC-ODBC Sources, Microsoft Excel File, CSV File, Web Service. For each data source a wrapper component is implemented.

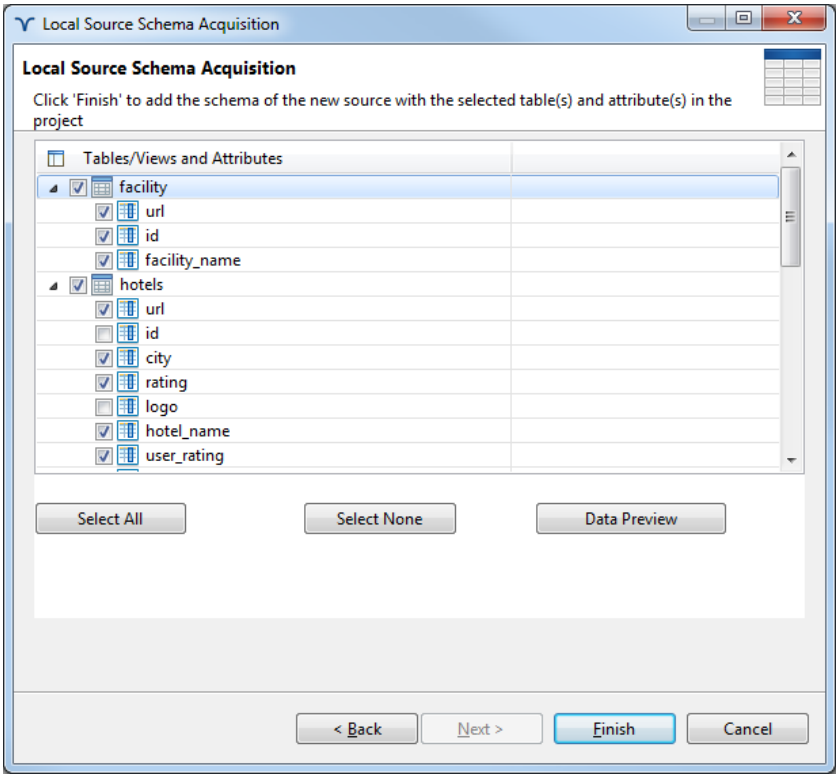


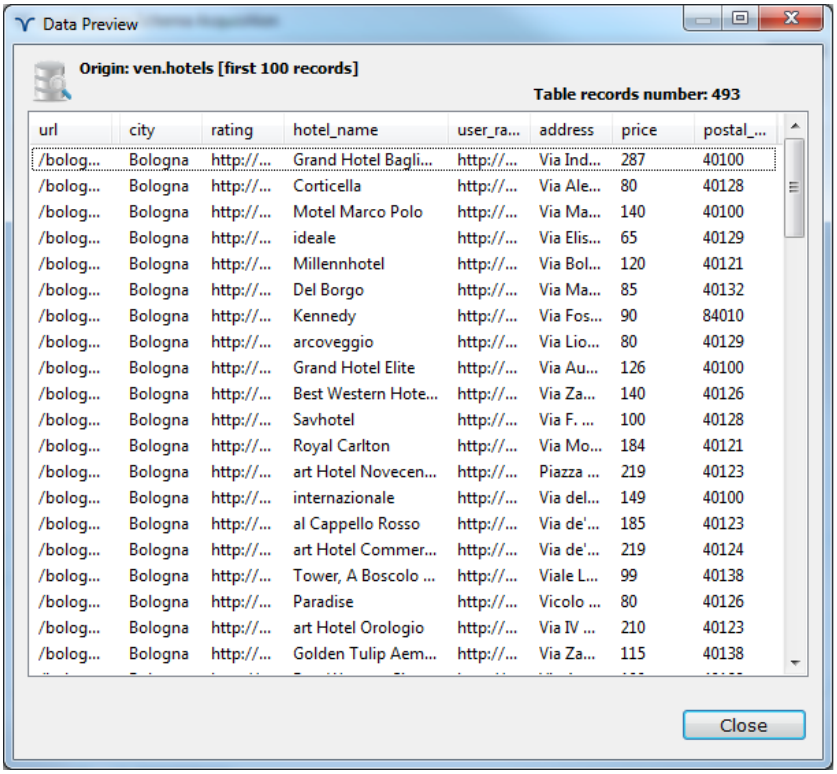
Fig. 2.8: Tables and Attributes Selection

As you can see in Fig. 2.7, you have to enter the correct parameters to establish the connection with the database server:

- Server Domain: the server hostname
- Port: the port number
- Username and Password

Then by clicking the *Connect* button, the *Database* list will be populated. Choose the database name you want to connect to, from the list.

Click the *Next* button to see the database tables and attributes. Depending on the integration needs you can choose to acquire, by selecting the tree nodes, a subset of the source tables and attributes (Fig. 2.8). A Data Preview functionality gives you access to the data stored in the different tables. By selecting a table and clicking on *Data Preview* you are allowed to explore the first one hundred records of the table or of a single attribute and see the table total records number (Fig. 2.9).



url	city	rating	hotel_name	user_ra...	address	price	postal_...
/bolog...	Bologna	http://...	Grand Hotel Bagli...	http://...	Via Ind...	287	40100
/bolog...	Bologna	http://...	Corticella	http://...	Via Ale...	80	40128
/bolog...	Bologna	http://...	Motel Marco Polo	http://...	Via Ma...	140	40100
/bolog...	Bologna	http://...	ideale	http://...	Via Elis...	65	40129
/bolog...	Bologna	http://...	Millennhotel	http://...	Via Bol...	120	40121
/bolog...	Bologna	http://...	Del Borgo	http://...	Via Ma...	85	40132
/bolog...	Bologna	http://...	Kennedy	http://...	Via Fos...	90	84010
/bolog...	Bologna	http://...	arcoveggio	http://...	Via Lio...	80	40129
/bolog...	Bologna	http://...	Grand Hotel Elite	http://...	Via Au...	126	40100
/bolog...	Bologna	http://...	Best Western Hote...	http://...	Via Za...	140	40126
/bolog...	Bologna	http://...	Savhotel	http://...	Via F. ...	100	40128
/bolog...	Bologna	http://...	Royal Carlton	http://...	Via Mo...	184	40121
/bolog...	Bologna	http://...	art Hotel Novecen...	http://...	Piazza ...	219	40123
/bolog...	Bologna	http://...	internazionale	http://...	Via del...	149	40100
/bolog...	Bologna	http://...	al Cappello Rosso	http://...	Via de'...	185	40123
/bolog...	Bologna	http://...	art Hotel Commer...	http://...	Via de'...	219	40124
/bolog...	Bologna	http://...	Tower, A Boscolo ...	http://...	Viale L...	99	40138
/bolog...	Bologna	http://...	Paradise	http://...	Vicolo ...	80	40126
/bolog...	Bologna	http://...	art Hotel Orologio	http://...	Via IV ...	210	40123
/bolog...	Bologna	http://...	Golden Tulip Aem...	http://...	Via Za...	115	40138

Fig. 2.9: Data Preview

Then click on *Finish* to complete the source schema extraction. Once the local source schema extraction process is finished the tables of the source (called in the ODL_{I^3}language local classes) will appear in the *Source Explorer* view. From there you can access at any time the local Data Preview functionality by right-clicking on a local class and choosing *Data Preview*.

2.4.2 Microsoft Excel File Schema Extraction

In the first wizard page select *Microsoft Office Excel file* as source type and enter a name[5]

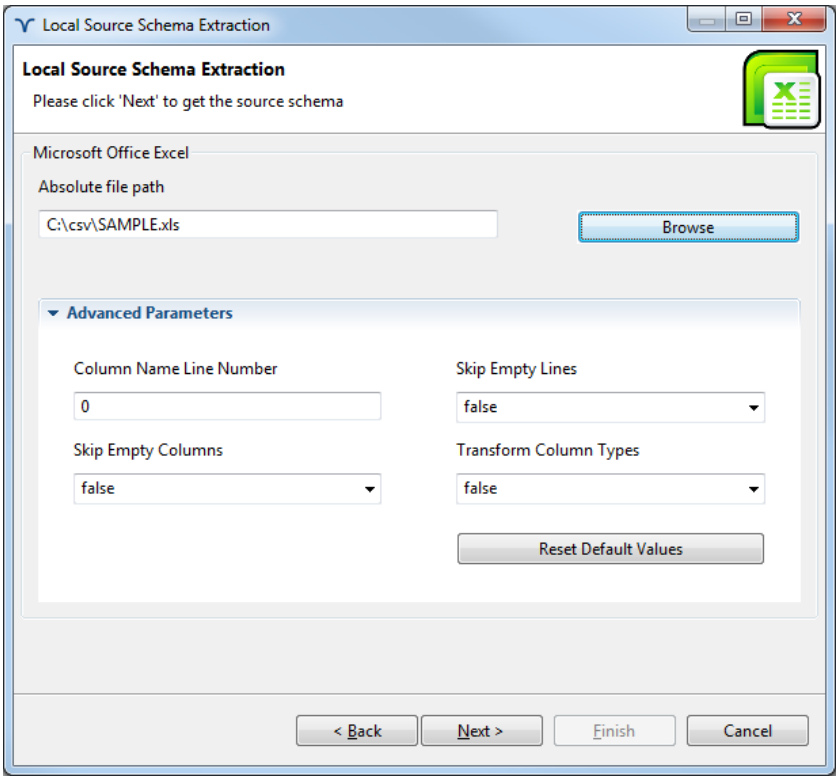


Fig. 2.10: Microsoft Excel File Configuration Parameters

As you can see in figure Fig. 2.10, you have to enter the absolute path of the Excel File and complete the configuration parameters reported below:

- Column Name Line Number: The line number from which to get the

[5]The Microsoft Excel Wrapper and CSV Wrapper are based on the Metamodel library: http://metamodel.eobjects.org/

names of the columns[6]

- Skip Empty Lines: Boolean that defines if empty lines in the Excel spreadsheet should be skipped while reading the spreadsheet
- Skip Empty Columns: Boolean that defines if empty columns in the Excel spreadsheet should be skipped while reading the spreadsheet
- Transform Column Types: Boolean that defines if the system should try to transform column data types[7]

Click the *Next* button to see the sheets of the file and the columns of each sheet. Depending on the integration needs you can choose to acquire, by selecting the tree nodes, a subset of the sheets and columns. You can use the *Data Preview* tool on a sheet (to explore the first one hundred rows of the sheet) or on a column (to explore the first one hundred values of a column).

Then click on *Finish* to complete the source schema extraction. Once the local source schema extraction process is finished the sheets of the file (called in the ODL_{I^3}language local classes) will appear in the *Source Explorer* view. From there you can access at any time the local Data Preview by right-clicking on a local class and choosing *Data Preview*.

2.4.3 Connection to a MySQL database through Web Service

If it's not possible to establish a direct JDBC connection with a particular source that is distributed over the Internet then the Web Service Wrapper can be used. This wrapper has two main components: a server and a client.

1. *Wrapper Web Service Server component*: On the server side, a daemon (*WSDLWrapper_impl*) runs: it offers a Web Service server object and give you access to a data source for which a wrapper in MOMIS exists. **Let's see how to start the Web Service Server?** Obtain:
   - datariver-communication-1.2.jar
   - datariver-core-1.2.jar
   - velocity-1.4.jar
   - commons-collections-20040616.jar;

[6]Note that this line number is affected by the Skip Empty Lines property! If Skip Empty Lines is set to true, the line numbers will begin from the first non-empty line.

[7]If set to false all columns types will be string

- log4j-1.2.14.jar
- mysql-connector-java-5.1.6.jar

packages from the source distribution.

Then, you can directly start the Web Service as follows:

java -cp list-of-the-jars-reported-above it.unimo.datariver.communication.tools.wsdl.WSDLWrapper.WSDLWrapper_impl End_Point_URL schemaName.xml source_name

- End Point URL: an enpoind url describes the port and the path where the service will response to, eg http://localhost:8080/test
- schemaName.xml: schemaName.xml it is an xml file that contains the source connection/configuration parameters.
- sourceName: the name of the source

Below is reported the xml schema of the schemaName.xml:

```
<?xml version="1.0"?>

<Schema
name="name" momisCodeVersion="momis.0.0.1-SNAPSHOT compiled.${today.date}" >

<Source
    name="source_name"
    description=""
    type="relational"
    >
<SourceCommunicationConfiguration
driverName="it.unimo.datariver.communication.core.jdbc.WrapperJdbcCore_MySql">
WrapperJdbcCore.DriverClassName=com.mysql.jdbc.Driver
WrapperJdbcCore.Password=password
WrapperJdbcCore.Url=jdbc\:mysql\://hostname\:3306/database_name
WrapperJdbcCore.User=user
WrapperJdbcCore.schemaName=source_name
</SourceCommunicationConfiguration>
  </Source>
  </Schema>
```

2. *Wrapper Web Service client component*: The Client is the wrapper component used to access a published web service. So first of all you have to publish the service as shown above.

Click on the toolbar item  to upload the schema of the source through web service. Enter the connection parameters as shown in figure Fig. 2.11 and then click *Finish*.

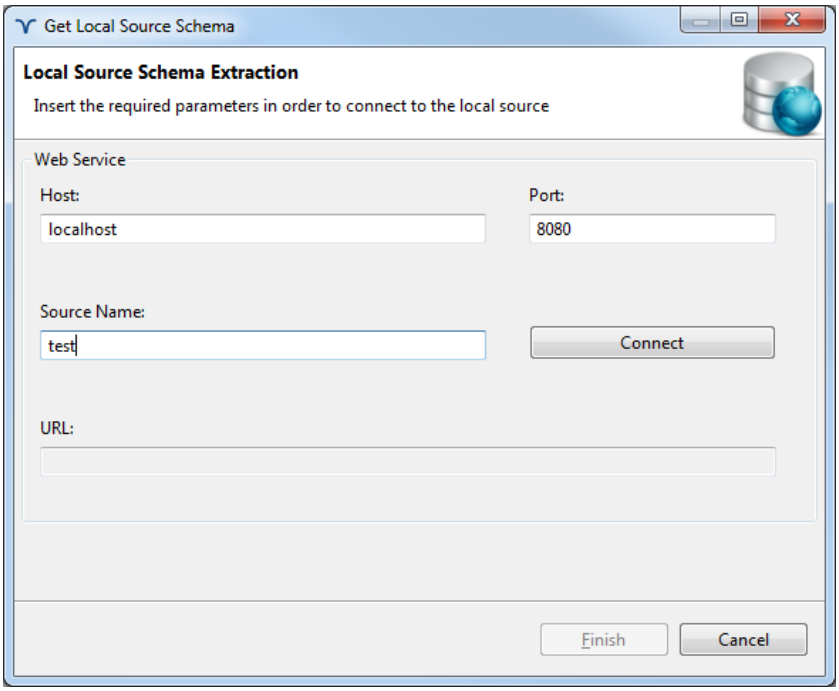


Fig. 2.11: Web Service Connection Settings

The new source displays in the *Source Explorer* view.

2.5 How to create a new Global Schema?

Click on *Global Schema -> New Global Schema*, enter a name for the new global schema and click *OK* (Fig. 2.12). The name of the global schema must be unique within a project. The new GS displays in the *Global Schema Explorer* view. The system automatically loads the just created GS in the Global Schema Designer editor (Fig. 2.13).

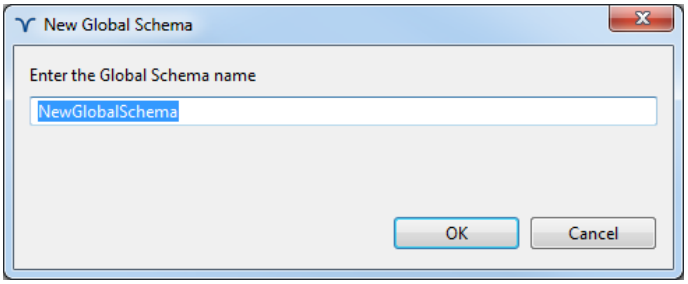


Fig. 2.12: New Global Schema dialog

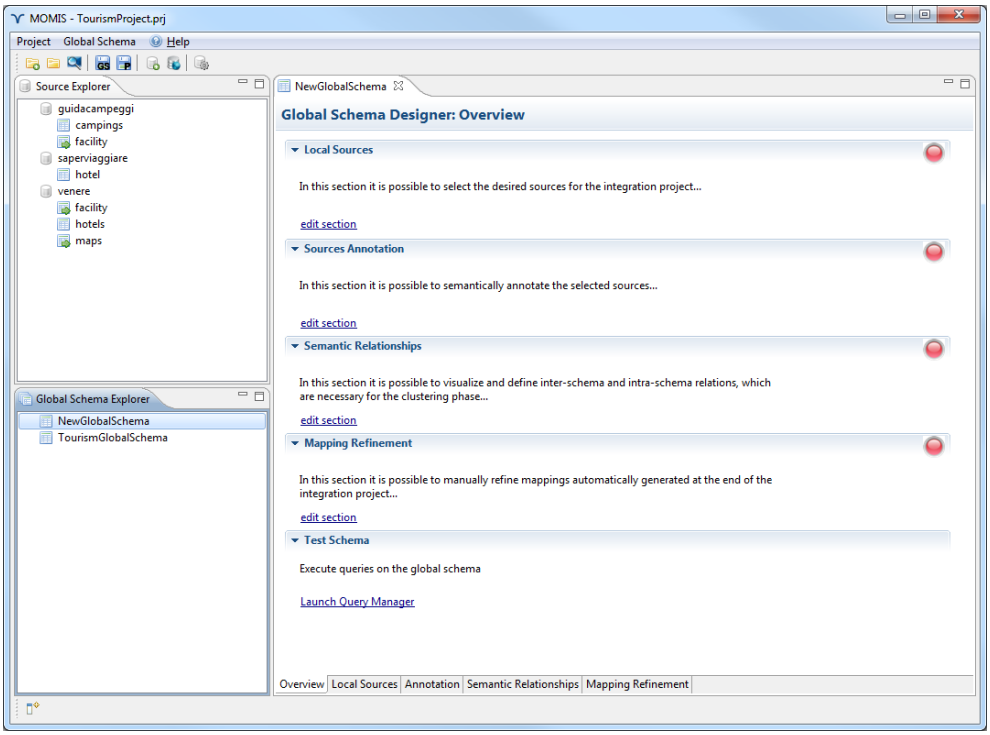


Fig. 2.13: Global Schema Designer: Overview

From here you can edit any of the following sections:

- *Local Sources*
- *Sources Annotation*
- *Semantic Relationships*
- *Mapping Refinement*

You have to complete each section, in order to obtain a correct GS.

2.6 Local Sources Selection

In this section you have to select the sources you want to include in your GS, by choosing them among the sources you acquired before. Right-click on a source from the *Source Explorer* view and click on *Add selected source to the Global Schema*; for each selected source you can see more information in the *Source Details: ODL_{I^3} Representation* section, as shown in Fig. 2.14.

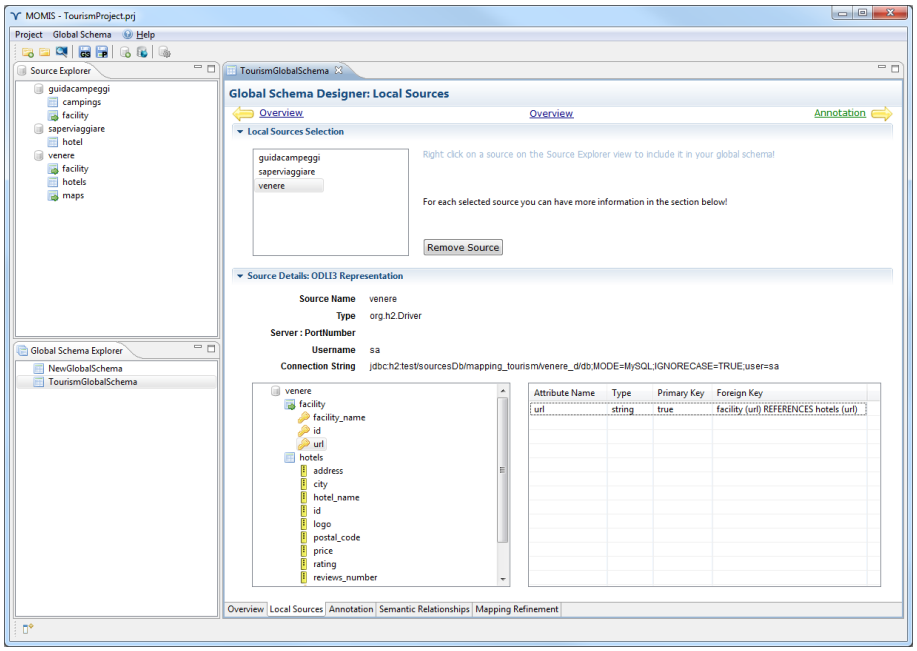


Fig. 2.14: Local Sources Details

2.7 Local Sources Annotation

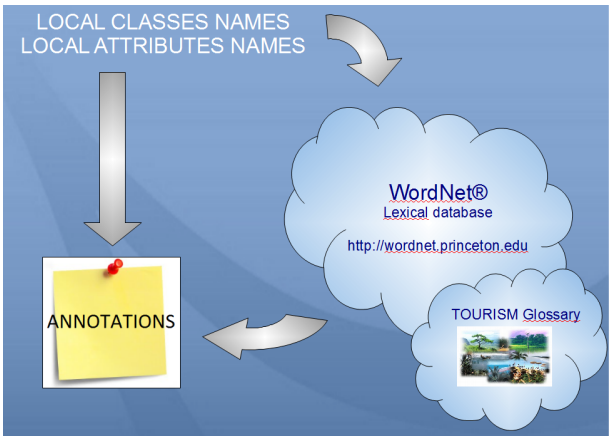


Fig. 2.15: Annotation with WordNet

In the first step we selected the local sources that we want to integrate (Fig. 2.14). In this step we want to express the meaning of classes and attributes, therefore the annotation procces must be performed. What is the annotation? It is a mapping of a given term (class or attribute name) to a well-defined set of concepts of a lexical ontology. The annotation process consists in associating to each term one or more meanings w.r.t. a common lexical reference. For example, the attribute *"name"* will be annotated with the meaning *"a language unit by which a person or thing is known"*. In the current version of MOMIS, we adopt as lexical reference the WordNet database[8]. WordNet is a thesaurus for the English language, that groups words (called lemmas in the WordNet terminology) into sets of synonyms called synsets, provides short definitions (called gloss), and connects the synsets through a wide network of semantic relationships. Usually integration projects involve large data sources, with hundreds of tables and attributes, coming from a particular domain of interest (e.g. tourism), that's why it is possible to import domain glossaries in order to annotate terms belonging to a specific domain. For example for our project we included the glossary of the Tourism domain(Fig. 2.15). Why do we need to annotate? Because, starting from the local source annotations, MOMIS can derive lexical relationships among elements of different local sources(this process is called Semantic Relationship

[8]WordNet is freely and publicly available for download see http://wordnet.princeton.edu. In the current version we used WordNet version 3.0. WordNet is distributed as-it-is and we extrapolated the WordNet internal organization, into a relational DBMS (DataBase Management System) and used it to store also the domain glossaries

Extraction phase and it is explained in details in 2.8 section). The manual annotation is a two steps process:

- **Base form choice**[9]: the system automatically proposes the base form if the term exists in WordNet or in the domain glossary, if no base form is found or the base form found is not satisfactory, we can manually insert it by choosing among all the base forms available in WordNet or in the domain glossary.
- **Meanings choice**[10]: we have to map the term on one or more meanings among all the available meanings of the inserted base form.

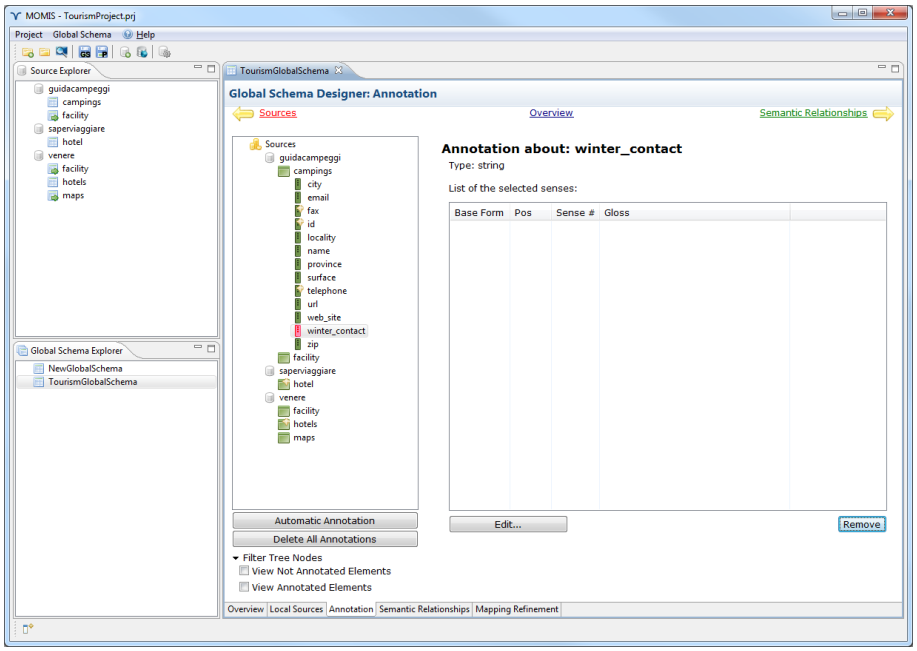


Fig. 2.16: Source Annotation Section

As you can see from Fig. 2.16, on the left, there is a tree representing the local source classes and attributes. The colored icons help you find the elements that are not annotated and the elements already annotated (see Fig. 2.17).

You can perform an automatic annotation and then manually refine the annotations proposed by the system. You are not forced to annotate all

[9]The *base form* is the english root form of a word (only the root form is stored in the database unless it has irregular inflected forms)

[10]The *meaning* is a definition that represents the sense or the significance of a word, i.e. a dictionary definition of a particular word

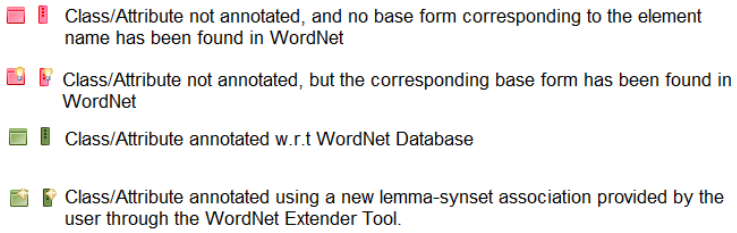


Fig. 2.17: Annotation Icons

terms, but in this case there is a loss of semantics. This means that the system will be able to discover less Lexical Relationships and this impacts on the quality of the proposed global source.

2.7.1 Manual Annotation

As we described in the previous section, the manual annotation is a two steps process:

1. Base form choice;
2. Meaning/s choice.

This process must be repeated for each term you want to annotate.

In WordNet lemmas and synsets are classified into four syntactic categories:nouns, verbs, adjectives and adverbs. Each syntactic category is identified by the symbols shown in Fig. 2.18.

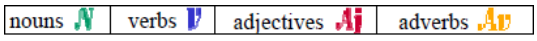


Fig. 2.18: WordNet Icons

Let us see how to annotate class and attribute names. Click on the tree node (Fig. 2.16) that represents the attribute/class that you want to annotate and click on *Add annotation* button; a new window will be opened. If the term exists in WordNet or in the domain glossary then the base form is automatically proposed by the system. If you don't find it satisfactory, you can enter another one. Then click on *Add base form* button and choose among the possible meanings of the base form(Fig. 2.19). You can select one or more meanings.

If the term does not exists in WordNet or in the domain glossary, then the system shows a message: "ATTENTION: lemma .. doesn't exist into WordNet database"; in this case you have three possible choices:

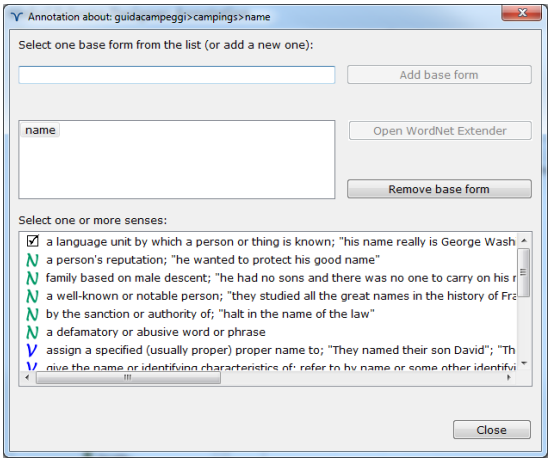


Fig. 2.19: Annotation of Class and Attribute Names

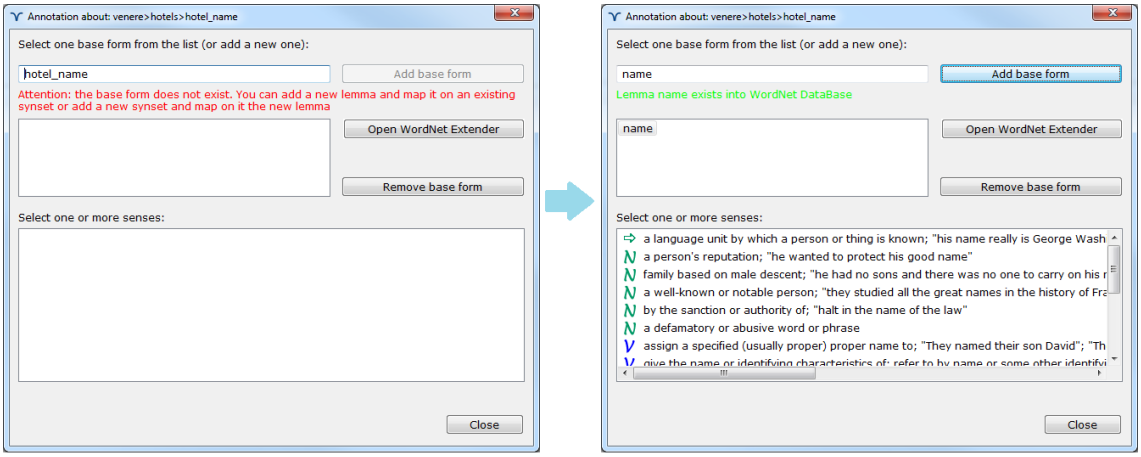


Fig. 2.20: hotel_name annotation

1. do not annotate the term

2. insert a different base form that is present in WordNet; for example in fig. 2.20 you may see that the base form "hotel_name" is not present in WordNet, so we decided to choose the base form "name"; since we used "name" for annotating another element before (Fig. 2.19), a green arrow notices us which meanings have been previously choosen for that base form.

3. insert the lemma "hotel_name" into the WordNet database using the *WordNet Extender Tool* (which will be explained in depth in the subsection 2.7.2), by clicking on the *Open WordNet Extender* button.

Since the manual annotation is time consuming, MOMIS provides two particular tools for accelerating this process:

1. The Automatic Annotation;

2. The import of Source Annotations.

These tools will be described in details in the subsections 2.7.3 and 2.7.4.

2.7.2 The WordNet Extender Tool

The WordNet Extender [1] tool enables the extension of the lexical reference with domain glossaries. In order to understand how to use the *WordNet Extender*, it is important to get the sense of some concepts related to WordNet. In WordNet, english nouns, verbs, adjectives, and adverbs are organized into sets of synonyms (called synset), each synset has a short definition called gloss. One ore more base forms (called lemmas) are associated to each synset; for example the gloss "the number is used in calling a particular telephone" is the definition of a synset associated to the lemma "phone_number".

A word that has more than one sense is said to be *polysemous*; two words that share at least one sense in common are said to be *synonyms*. So in WordNet each lemma is associated to one or more synsets as the same word can be used with different meanings in different contexts.

Each synset is associated to one or more lemmas; several words can be used to express the same concept; for example "zip" and "postal_code" are associated to the synset identified by the gloss "a code of letters and digits added to a postal address to aid in the sorting of mail".

The synsets are linked through semantic relations:

- A hypernym relationship connects two synsets where the first generalizes the second, e.g. *"animal" is a hypernym of "dog"*[11];
- A hyponym relationship connects two synsets where the first specializes the second, e.g. *"student" is a hyponym of "person"*footnoteThis means that the synset associated to "student" is a hyponym of the synset associated to "person".. The hyponym is the opposite relationship of hypernym.

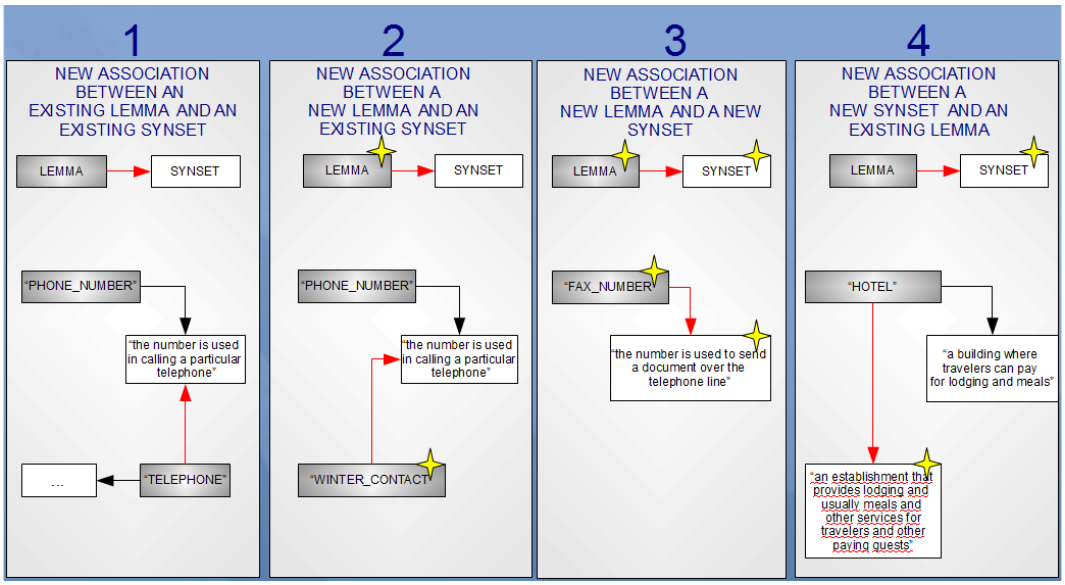


Fig. 2.21: Using WordNet Extender for exending WordNet database.

Both nouns and verbs are organized into hierarchies, defined by hypernym relationships. By using the WordNet Extender you will be able to:

1. extend WordNet by inserting a new association between an existing lemma and an existing synset(Fig. 2.21-1); for example, we could associate the lemma "telephone", which is already present in WordNet, with one of the synsets of the lemma "phone_numbe";
2. extend WordNet by inserting a new lemma and associate it with an existing synset (Fig. 2.21-2); for example, we could create the new lemma "winter_contact", which is not present in WordNet, and associate it with the synsets "the number is used in calling a particular telephone";

[11]This means that the synset associated to "animal" is a hypernym of the synset associated to "dog".

3. extend WordNet by inserting a new lemma and a new synset for it (Fig. 2.21-3); for example, we could create the lemma "fax_number", and associate it with a new synset with gloss: "the number is used to send a document over the telephone line";

4. extend WordNet by inserting a new synset and associate it with an existing lemma (Fig. 2.21-4); for example, we could create the new synset "an establishment that provides lodging and usually meals and other services for travelers and other paying guests" and associate it with the existing lemma "hotel";

5. in case you are adding a verb or a noun synset (as in the previous case number 3 and 4), it is necessary to link the new synset to the hierarchy. You should specify at least one new hypernym relationship with another synset in the graph.

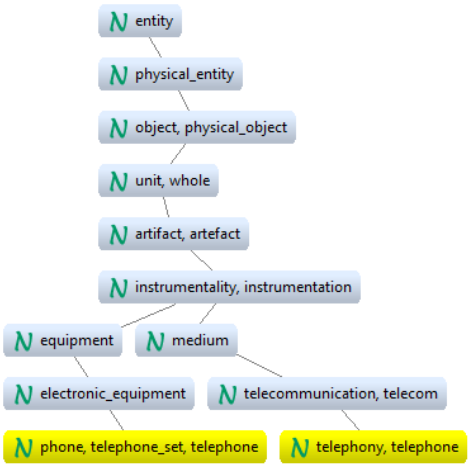


Fig. 2.22: Hypernym Graph of the lemma "telephone"

The *Hypernym Graph Viewer*, is very helpful to create sound relationships between the added synsets and the pre-existing ones during the extension of the lexical reference. From the annotation tree, right click on a class or attribute name, then select the *View Hypernym Graph* menu item. It is possible to navigate the graph by focusing on a specific synset to view only the branch of its hypernyms, or by using the keyword search. For example the direct hypernyms of the two synsets associated to "telephone"

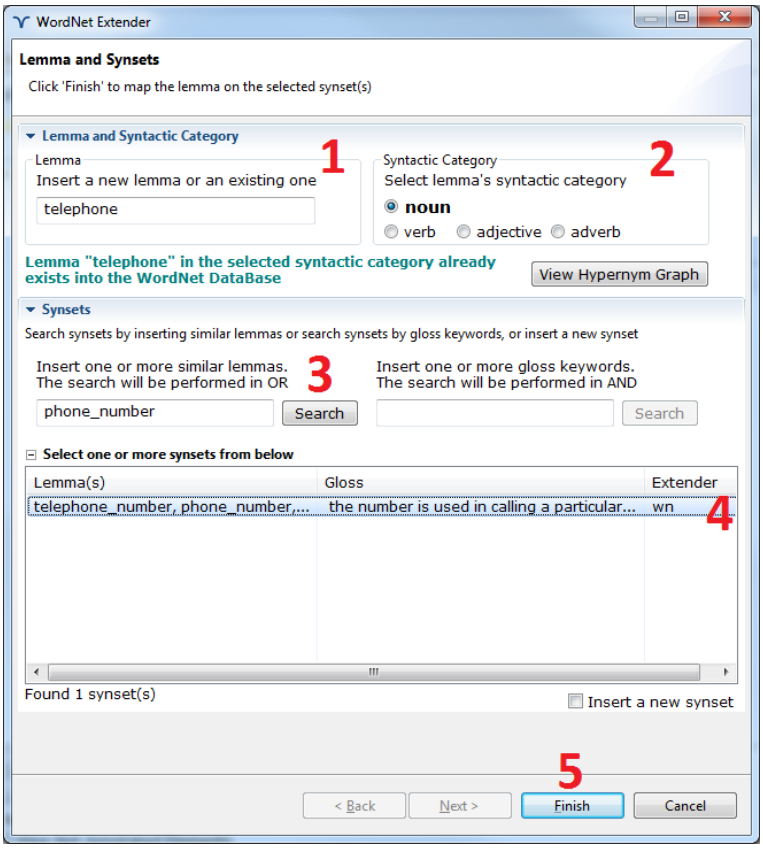


Fig. 2.23: Steps to follow for associating a new or existing lemma with an existing synset

are "telecommunication, telecom" and "electronic_equipment" (Fig. 2.22). By double-clicking on a synset you will see its whole hierarchy[12].

The wordnet Extender usage is quite simple: for associating a (new or existing) lemma with an existing synset you are asked to follow the steps shown in fig. 2.23:

1. insert the new lemma;
2. choose the lemma's syntactic category;
3. search the synset by inserting similar lemmas or gloss keywords (to make a search among synset glosses);
4. select the synset(s) you want the lemma to be associated with;

[12]The synset associated to "entity" is the top parent of the hierarchy, for synsets of the noun syntactic category

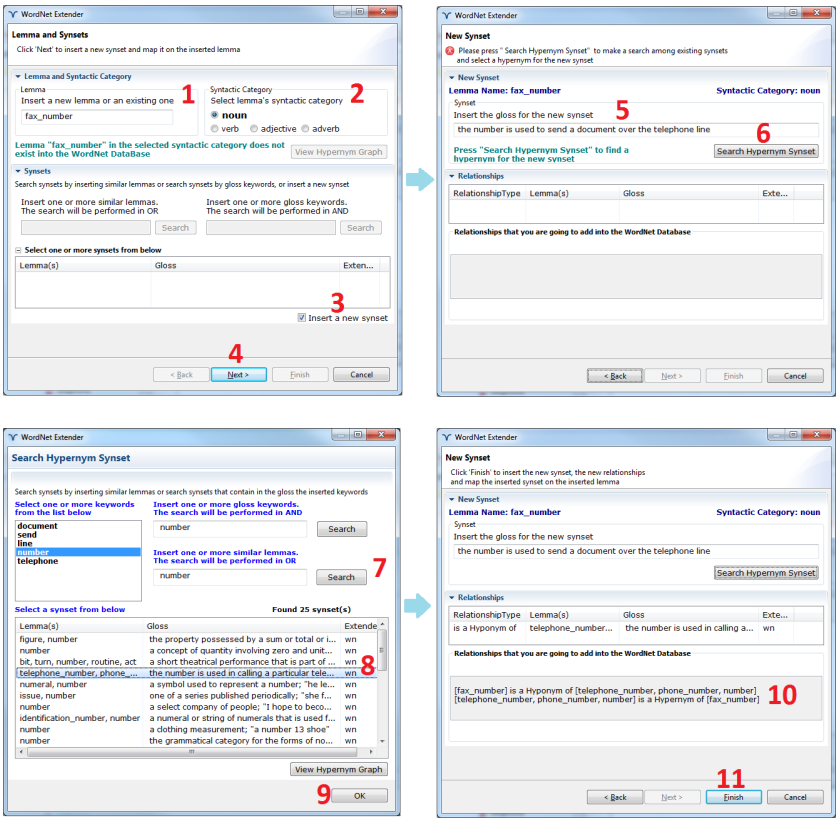


Fig. 2.24: Steps to follow for associating a new or existing lemma with a new synset

5. click on *finish*;

Then you will be able to annotate the term by using the new association lemma-synset just inserted into WordNet. If, at step 4, you do not find the proper synset for your new lemma, you can add a new synset simply by following the next steps.

For associating a (new or existing) lemma with a new synset you are asked to follow the steps shown in Fig. 2.24:

1. insert the lemma;
2. choose the lemma's syntactic category;
3. check *Insert a new synset*;
4. click on *Next*;
5. insert the gloss for the new synset;

6. click on *search hypernym synset*: a new window will appear, for helping you in finding the more appropriate hypernym synset in order to accurately place the new synset in the WordNet hierarchy;

7. choose one or more keywords among the ones proposed by the system or insert other keywords and search among glosses or lemmas, then click on search; a list of synsets will appear;

8. select the hypernym synset and click on *View Hypernym Chart* for having a look at its collocation in the WordNet hierarchy, in order to check if it is the more appropriate hypernym for the new synset;

9. click on *OK*

10. the system will notice the new relationships that will be added into the WordNet database.

11. click on *Finish* to complete the process; at this point you can annotate the term by using the new synset just added into the WordNet database.

2.7.3 Automatic Annotation

In order to optimize the annotation phase and increase the annotation accuracy, we implemented an automatic annotation algorithm which includes stemming and stop words removal functionalities. The main advantage of automatic annotation is simply speed: wholly or partially automated methods facilitate the annotation of large sets of classes. If the lexical reference has been extended, the automatic annotation algorithm associates to each data source element the more recent meaning of the domain glossary, else, it associates to the data source element the first meaning (most common used meaning) present in WordNet[13] By clicking on the *Automatic Annotation* button the system will try to annotate all the terms (Fig. 2.25). The algorithm gives good results in presence of English words but is not able to deal with several compound words. That's why we added the *Data Preview* tool, right click on a tree node and explore the values of the class/attribute. For example "winter_contact"; through the data preview tool we can notice that "winter_contact" is the number that should be used in winter so we choose the base form phone_number and select the firts meaning.

After the automatic annotation you can manually refine the annotations proposed by the system.

[13]Synsets in WordNet are ranked in the order of their utilization frequency.

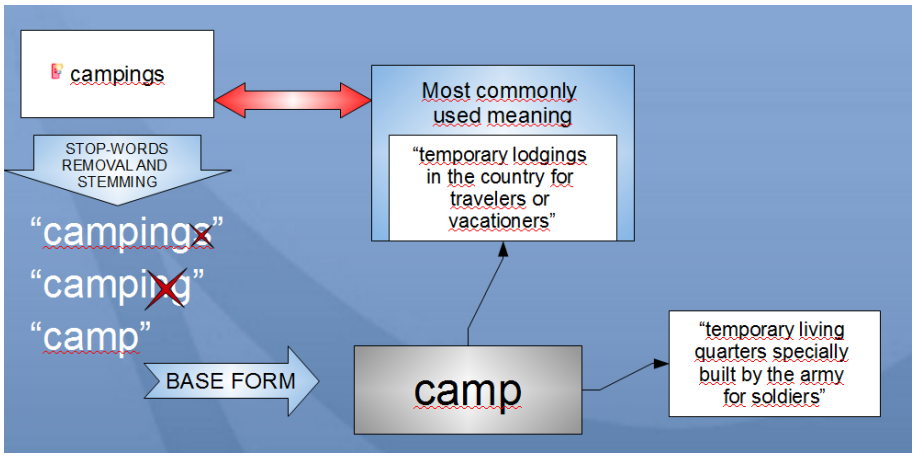


Fig. 2.25: Automatic Annotation Process

Notice that the automatic annotation will not try to annotate the elements you have preaviously manually annotated so you will not lose your work.

2.7.4 Annotation Importer

The reuse of previous annotations is an important feature. For this reason, a tool for easily importing source annotations from a GS to another GS has been developed. If one of the sources has been annotated already in another global schema you can import the source annotation by right-clicking on the source name, select *Import source annotation from another global schema* and selecting the schema from which you want to import the annotations(Fig.2.26). For example we create a new global schema, named *NewGlobalSchema* and we want to import the annotation of the local source *guidacampeggi*, so we right click on the source name and choose *Import Source Annotation from another Global Schema*, choose TourismGlobalSchema and you will see that all the annotations for that source have been imported.

An example

Let's see how all these tools are useful with an example. In the annotation phase, we found as the most adequate annotation for the term guidacampeggi.campings: **Base form**: "camp" and **Meaning**: "temporary lodgings in the country for travelers or vacationers", while a problem arise from the source venere. We didn't find satisfactory the meaning associated to hotel: "a building where travelers can pay for lodging and meals" because it refers to

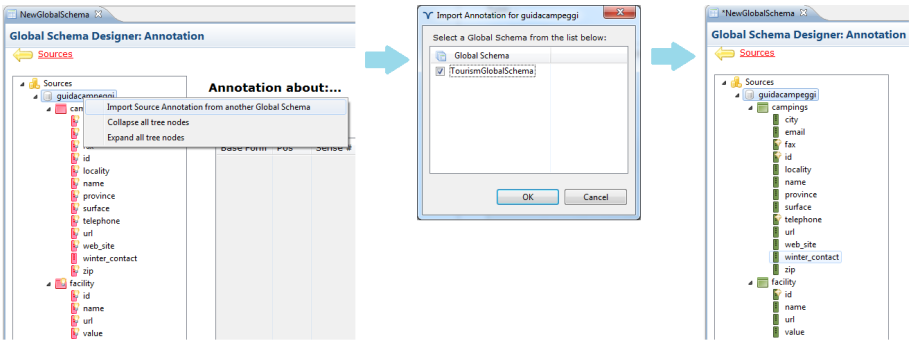


Fig. 2.26: Annotation Importing

hotel as a building and not as a service. From the Hypernym Chart we can notice that the synset associated to hotel is a hyponym of the synset "a structure that has a roof and walls and stands more or less permanently in one place" identified also by the set of lemmas building, edifice (Fig. 2.27) and camp is a hyponym of the synset "structures collectively in which people are housed", identified also by the set of lemmas living_accommodations, housing, lodging Through the *WordNet Extender* tool we created a new synset for the lemma hotel by introducing the gloss "a lodging that provides accommodation, meals and other services for travelers and other paying guests". Then we have to build a relation between the added concept and pre-existing ones. We are asked to select some keywords (the system automatically suggest a set of relevant keywords, deriving them from the inserted gloss) or insert new keywords and make a search among existing synsets. In our case, we want to relate hotel with the already existing synset "structures collectively in which people are housed", identified also by the set of lemmas living_accommodations, housing, lodging and create a new hypernym relationship (Fig. 2.28). That means into the WordNet database will be added two symmetric relationships:

- hotel is a hyponym of living_accommodations, housing, lodging
- living_accommodations, housing, lodging is a hypernym of hotel

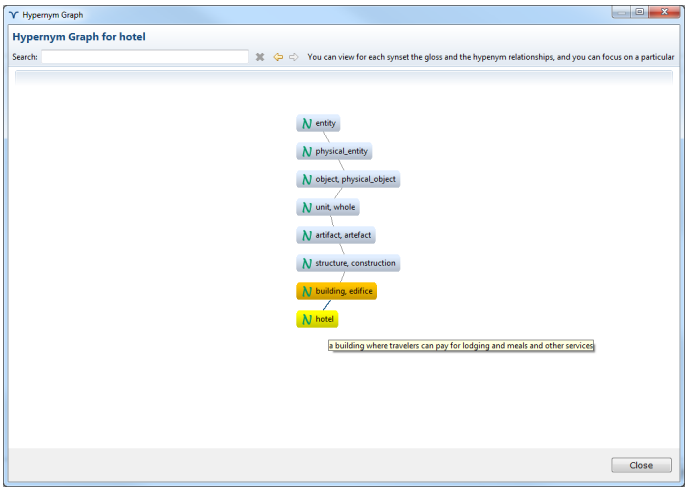


Fig. 2.27: Hypernym Graph Viewer: "hotel"

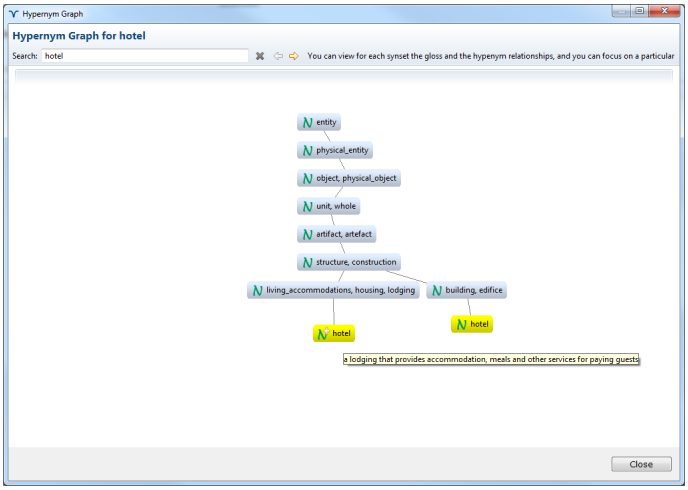


Fig. 2.28: Hypernym Graph Viewer: New Synset for "hotel"

2.8 Semantic Relationships Extraction

Starting from the annotated local schemas, MOMIS builds a Common Thesaurus that describes intra and inter-schema knowledge in the form of the following semantic relationships: synonyms (SYN), broader terms/narrower terms (BT/NT) and related terms (RT).

- SYN: A SYN B means that A and B are synonyms. SYN is a symmetric relationship.
- BT/NT: A BT B means that A is a broader than B. The opposite relationship of BT is NT (A NT B means that A is more specific than B).
- RT: A RT B means that the terms are related.

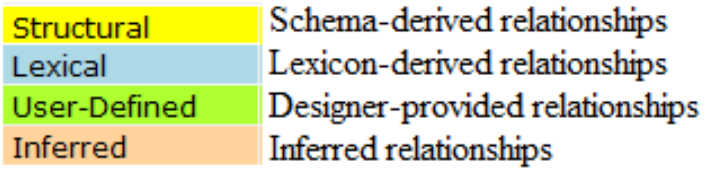


Fig. 2.29: Semantic Relationships Type

The Common Thesaurus is incrementally built by adding four types of relationships (Fig. 2.33):
Schema-derived relationships (also called Structural relationships): automatic extraction of intra schema relationships from each schema separately. For example, we extract intra schema RTrelationships from foreign keys in relational source schemas. In the relational model, a foreign key is a set of attributes in a relation used to express a reference to another relation. In *guidacempeggi* source we have a foreign key, so a Structural relationship is automatically discovered:

FK: facility(url) REFERENCES campings(url) => guidacampeggi.facility
RT guidacampeggi.campings

When a foreign key is also a primary key, in both the original and reference relations, MOMIS extracts BT and NT relationships, which are derived from inheritance relationships in object oriented schemas. In *venere* source we have a foreign key that is also a primary key in both the original and reference relations, so a structural relationship is automatically discovered.

FK: maps(url) REFERENCES hotels(url) => venere.maps NT venere.hotels

Lexicon-derived relationships (also called Lexical relationships): inter-schema lexical relationships derived by the annotated terms and WordNet interaction. As described in the previous section WordNet connects the synsets through a wide network of semantic relationships. Actually, it is possible to exploit the following semantic relationships:

- Synonymy: terms that share the same meaning, i.e. belong to the same synset
- Hypernymy/Hyponymy: Generalization relation/Specialization relation. Hypernym and Hyponym are opposite relations.
- Meronymy/Holonymy: is the partwhole or HASA relation[14]. Meronymy and Holonymy are opposite relations
- Correlation: is a relation between two terms in two synsets that shares the same father in the hypernymy graph.

The relations coming from WordNet are added to the Common Thesaurus according to the following mapping:

- Synonymy: corresponds to a SYN relationship, e.g. *guidacampeggi.campings.zip SYNvenere.hotels.postal_code*
- Hypernymy/Hyponymy: correspond respectively to a BT/NT relationship, e.g. *venere.hotels.hotel_name NTguidacampeggi.campings.name*
- Meronymy/Holonomy: correspond to a RT relationship.
- Correlation: corresponds to a RT relationship, e.g. *venere.hotels.hotel_name RTvenere.facility.facility_name*[15]

Designer-supplied relationships: specific domain knowledge capture. The designer can supply new relationships. This operation is important because if meaningless or incorrect relationshipis are inserted, the subsequent integration process can produce a wrong global schema.
Inferred relationships: MOMIS exploits a Description Logic reasoner,

[14]A concept represented by the synset X is a meronym of a concept represented by the synset Y if in English we can assert X is a part of Y

[15]From the Hypernym Graph Viewer you can notice that *hotel_name* and *facility_name* share the same father: *name*

ODB-Tools[16], to infer new relationships by applying equivalence and subsumption computation. For example in the Common Thesaurus we have:
Structural relationship: *guidacampeggi.facility RT guidacampeggi.campings*
Lexical relationship: *guidacampeggi.facility syn venere.facility*
Thanks to ODB-tools the system automatically infers a new relationship:
venere.facility rt guidacampeggi.campings

In Fig. 2.30 you can see the *Semantic Relationships* section.

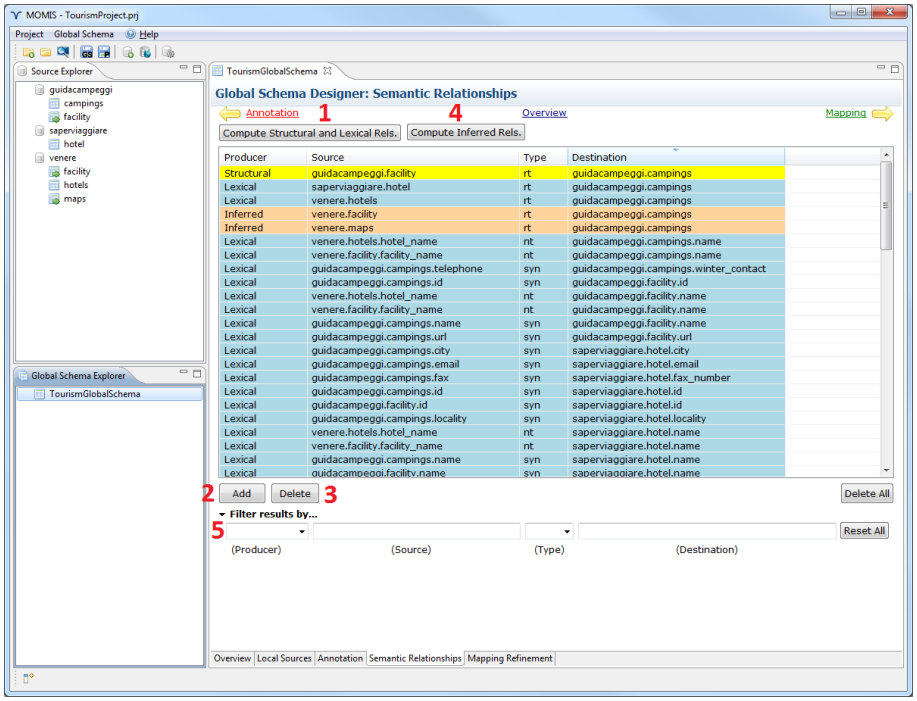


Fig. 2.30: Semantic Relationships Editor

To compute the Structural and the Lexical relationships press the *Compute Structural and Lexical Rels.* button (see 1 in Fig. 2.30). To supply new relationships press the *Add* button (see 2 in Fig. 2.30). A new dialog will be open and it allows you to add one ore more relationships among attributes/classes of different sources (Fig 2.31). If a relationship is meaningless, you can delete it from the Common Thesaurus; select the relationship and then press the *Delete* button (see 3 in Fig. 2.30). Then to infer new relationshpis starting from the ones present in the Common Thesaurus press the *Compute Inferred Rels.* button (see 4 in Fig. 2.30).

[16]The ODB-Tools is a fully modular software for schema validation and query optimization in OODB. For more information see: http://www.dbgroup.unimo.it/ODB-Tools.html

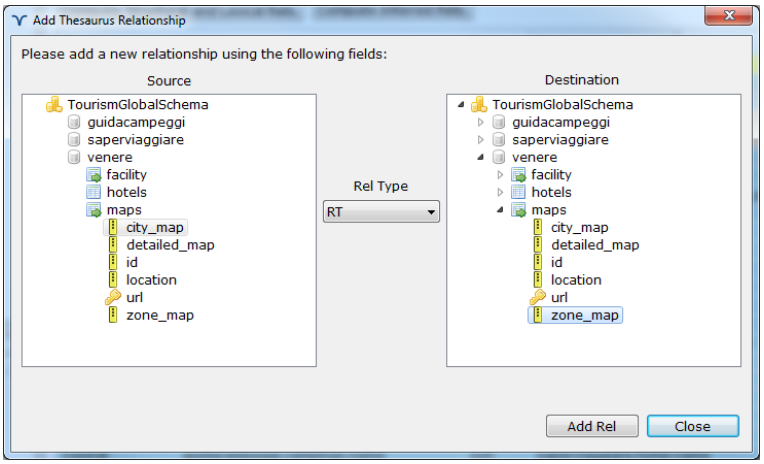


Fig. 2.31: User-provided Relationships Interface

The GUI also allows you to view filtered relationships: by Producer, Source, Type (SYN, RT, BT/NT) or Destination (see 5 in Fig. 2.30).

Note: each time you modify any annotation, all the Semantic Relationships have to be computed again (It is not necessary to erase them all before computing).

2.9 Mapping Table Creation and Refinement

The first step of the generation of a GS is the automatic creation of clusters: classes describing the same or semantically related concepts in different sources are grouped together in clusters using hierarchical clustering techniques; for each cluster a global class, with a set of global attributes and a mapping table, expressing mappings between local and global attributes, is defined.

After generating the clusters we can interactively refine and complete the proposed integration result.

By means of this interface we can set the parameters used by MOMIS to compute the clusters(Fig. 2.32). Starting from the left, the first 3 parameters set the "weight" of the different relationships, and the other 4 parameters set the threshold of affinity among local classes that have to be mapped together in the same global class, depending on the local class names and structures. You can set those parameters in order to obtain the more appropriate global schema.

If you click on the button *Generate Clusters*, the Global Classes will be created and loaded in the Global Sources tree of the *Mapping Refinement* section (Fig. 2.34). Depending on which configuration you chose, the system will build different clusters. It's your job to decide which configuration is better for your sources integration process.

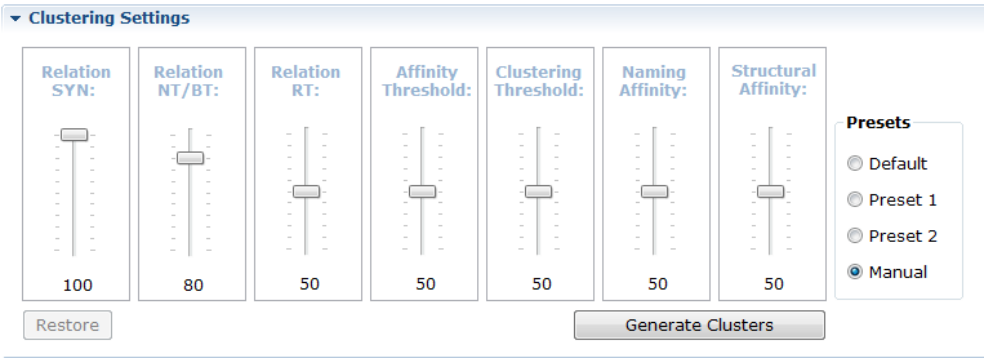


Fig. 2.32: Global Classes Generation

In Fig. 2.33 we report the icons used by the GUI and their intended meanings.

In this section you can also see the *Mapping Table* generated by the System for each Global Class, whose columns represent the local classes belonging to the GC and whose rows represent the global attributes of the GC. The element (GlobalAttribute,LocalClass) of the *Mapping Table* represents

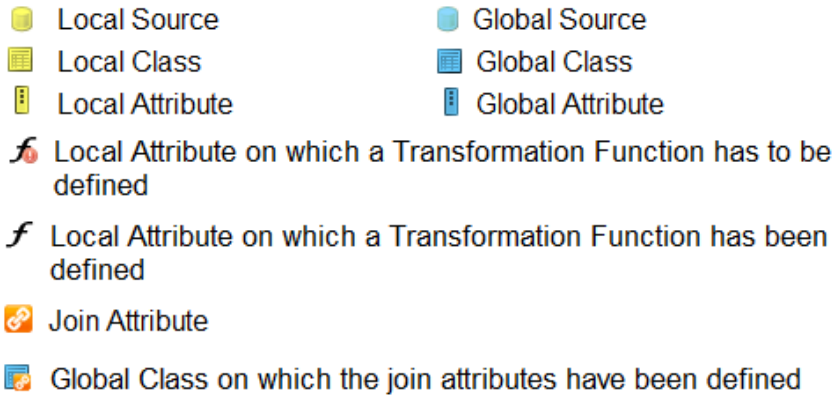


Fig. 2.33: Global Schema Editor: Icons Legend

the set of local attributes of LC which are mapped into the GlobalAttribute: in this way a GlobalAttribute may correspond to one o more local attributes of a LC; the element (GlobalAttribute,LocalClass) is empty if no LA of LC is mapped into the GlobalAttribute.

The context menu of the *Global Sources tree* (right click on global source, global class or global attribute) allows you to:

- add, remove or rename global classes/global attributes;
- set or modify the type of a GA;
- have a look at the local data using the *Data Preview* feature.
- set an attribute as join attribute(involved in a join condition);
- edit join functions of global classes.
- have a look at the global data using the *Global Data Preview* feature (available just after having selected the join attributes for the global class).
- edit the *Transformation Function* of a LA;
- edit the *Resolution Function* of a GA;

The "Mapping" phase permits the user to visualize and manage the GA of each Global Class created.

By selecting a node or a leaf of the *Global Classes Tree* the contents of the selected GC is shown. In particular, the user may visualize and manage the

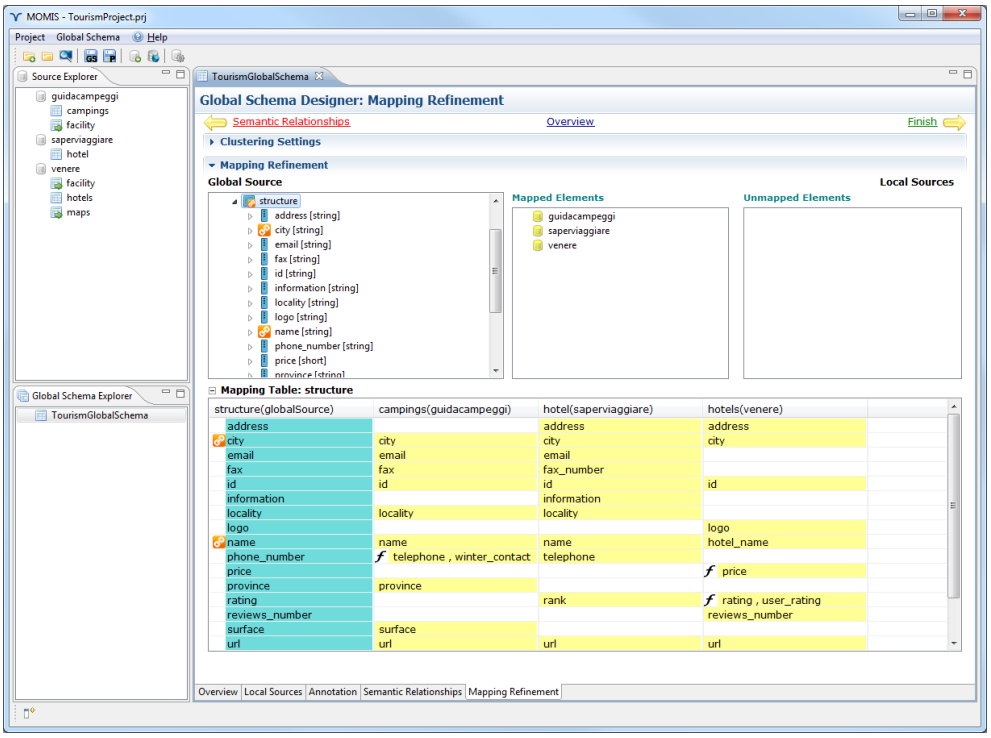


Fig. 2.34: Mapping Refinement Panel

Global Attributes (blue icon) and, by opening each node, the local attributes that are mapped on it (yellow icon).

If you right-click on a GA and choose *Remove Global Attribute* the attribute will be removed and the previously local mapped attributes are moved in the *Unmapped Elements* panel on the right. Attributes from the *Unmapped elements* panel can be mapped into a GA moving them on it, using Drag&Drop.

2.9.1 Join Functions

From the Mapping Refinement interface you can impose join conditions between local classes mapped on a GC, by right-clicking on a GC, choosing *edit Join Function*, writing the function in the Join Function Panel and save it. The system can provide a *default Join Function* as shown in Fig. 2.35.

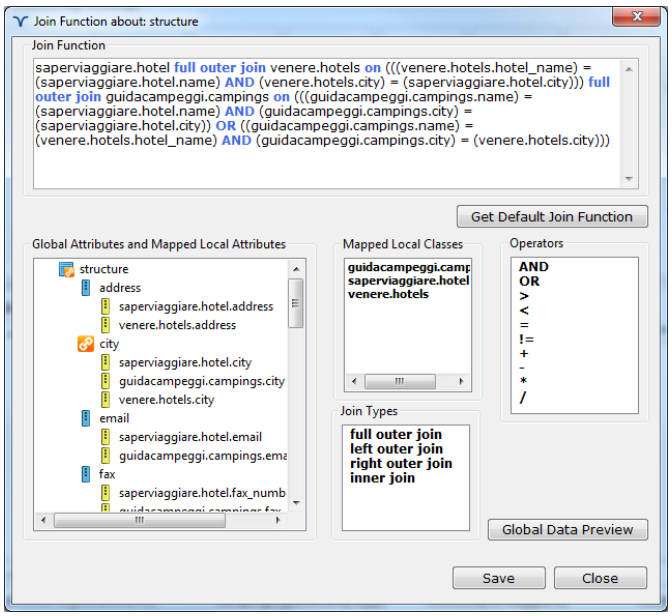


Fig. 2.35: Join Function

MOMIS follows a Global-As-View (GAV) approach for the definition of mappings between the GS and local schemata: the GS is expressed in terms of the local schemata. This means that for each Global Class C a view VC over the Local Classes of C must be defined.

One of the main innovation of the MOMIS system is that the view associated to a GC may be automatically composed by the system, i.e. this view need not be explicitly defined by the integration designer.

The automatic composition of the view is based on the following assumption: a GC performs *Data Fusion* among its local class instances, where *Data Fusion* is the process of fusing multiple records representing the same real-world object into a single and consistent representation.

The first step in a Data Fusion process is *Object Identification*, i.e. to identify instantiation of the same object in different sources (also known as record linkage, duplicate detection, reference reconciliation, and many others). In the MOMIS system, Object Identification is performed by *Join Conditions* among local classes belonging to the GC; this join condition is

specified on the transformed local attributes. More precisely, *Object Identification* requires to specify a set of global attributes, called *Join Attributes*, such that for each JoinAttribute, the element (JoinAttribute, LocalClass) is not null for each LocalClass belonging to the GC; then Join Conditions are automatically defined as follows.

Let JA1,..., JAk be the join attributes specified by the integration designer. Given the element (JAi,LC), let TF(JA,LC) be the corresponding Transformation Function.

For each pair of local classes LC1,LC2 belonging to the GC, the Join Condition between LC1 and LC2, denoted by JC(LC1,LC2) is defined as follows

TF(JA1,LC1) = TF(JA1,LC2)
AND
...
AND
TF(JAk,LC1) = TF(JAk,LC2)

As an example, for the GC *structure* and the local classes
LC1=guidacampeggi.campings
LC2=saperviaggiare.hotel
LC3=venere.hotels
in the mapping table we have respectively

- (name,LC1)=name with Identity as TF
- (name,LC2)=name with Identity as TF
- (name,LC3)=hotel_name Identity as TF
- (city,LC1)=city with Identity as TF
- (city,LC2)=city with Identity as TF
- (city,LC3)=city with Identity as TF

If the GA *n*ame is chosen as Join Attribute, i.e. JA=name, then

- JC(LC1,LC2) : LC1.name = LC2.name
- JC(LC1,LC3) : LC1.name = LC3.hotel_name)
- JC(LC2,LC3) : LC2.name = LC3.hotel_name

If *n*ame and *c*ity are chosen as Join Attributes, i.e. JA1=name and JA2=city, then

- JC(LC1,LC2) : (LC1.name = LC2.name) AND (LC1.city = LC2.city)
- JC(LC1,LC3) : (LC1.name = LC3.hotel_name) AND (LC1.city = LC3.city)
- JC(LC2,LC3) : (LC2.name = LC3.hotel_name) AND (LC2.city = LC3.city)

JC(LC1,LC2,LC3) : (LC1.name = LC2.name) AND (LC1.city = LC2.city)

On the basis of these *Join Conditions*, multiple records (coming from different local classes and representing the same real-world object) are combined in a single record by means of a *full outer join* operator.

For example, assuming the last case of join conditions, the view for the GC structure is automatically obtained as:

LC1 full outer join LC2 on ((LC1.name = LC2.name) AND (LC1.city = LC2.city)) full outer join LC3 on ((LC1.name = LC3.hotel_name) AND (LC1.city = LC3.city)) OR (LC2.name = LC3.hotel_name) AND (LC2.city = LC3.city)

Note: if you want to read more about Join Functions see **Appendix B**.

2.9.2 Transformation Functions Editor

When a GC is selected, the corresponding Mapping Table is shown in the lower panel of the window. By right-clicking on a non empty element (GlobalAttribute, LocalClass) or on a local attribute in the *Global Classes Tree* and choosing Edit Transformation Function a new window appears (Fig. 2.36). The Transformation Function transforms the type of a Local mapped Attribute into the one of the Global Attribute.

When you define a new TF you have to bear in mind the following considerations:

1. if only a LA of LC is mapped into a GA, the default Transformation Function is Identity , i.e. no transformation is needed and the type of the GA is assumed as the type of the LA;

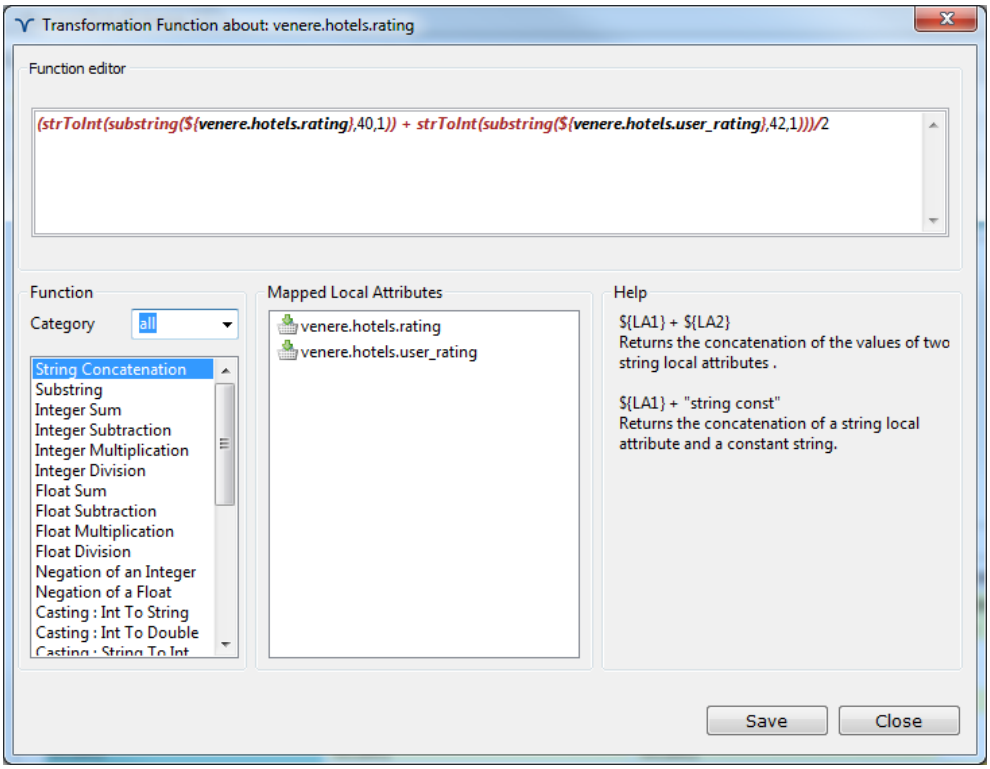


Fig. 2.36: Transformation Function Panel for the Local Attributes *rating* and *user_rating* of the local class Venere.Hotels

2. if more than one LA of LC are mapped into a GA, you have to select the type of the GA and apply the proper TF to the Local Attributes.

As shown in Fig. 2.36, the Mapped Local Attributes area will show all the attributes of the Local Class *Venere.Hotels* mapped on the GA *rating*.

The Function area will show all the functions applicable to the local attributes to transform their values.

In order to compose the TF you have to double click on a function (so it will be written in the *Function editor*), then choose the attributes you want to involve and put them into the function as parameters by clicking twice. You may manually edit the function in the *Function editor*.

In the example shown in Fig. 2.36 the TF applied calculates the average value between the local attributes *venere.hotels.rating* and *user_rating*, and is explained in fig. 2.37: the local attributes *venere.hotels.rating*, *venere.hotels.user_rating* and *saperviaggiare.hotel.rank* refer to an evaluation of the hotel quality and are mapped together on the same global attribute *rating*. Now, we need to define the type and then the possible values for this global attribute.after having a look at the local sources through the *Data Preview* tool, we discover

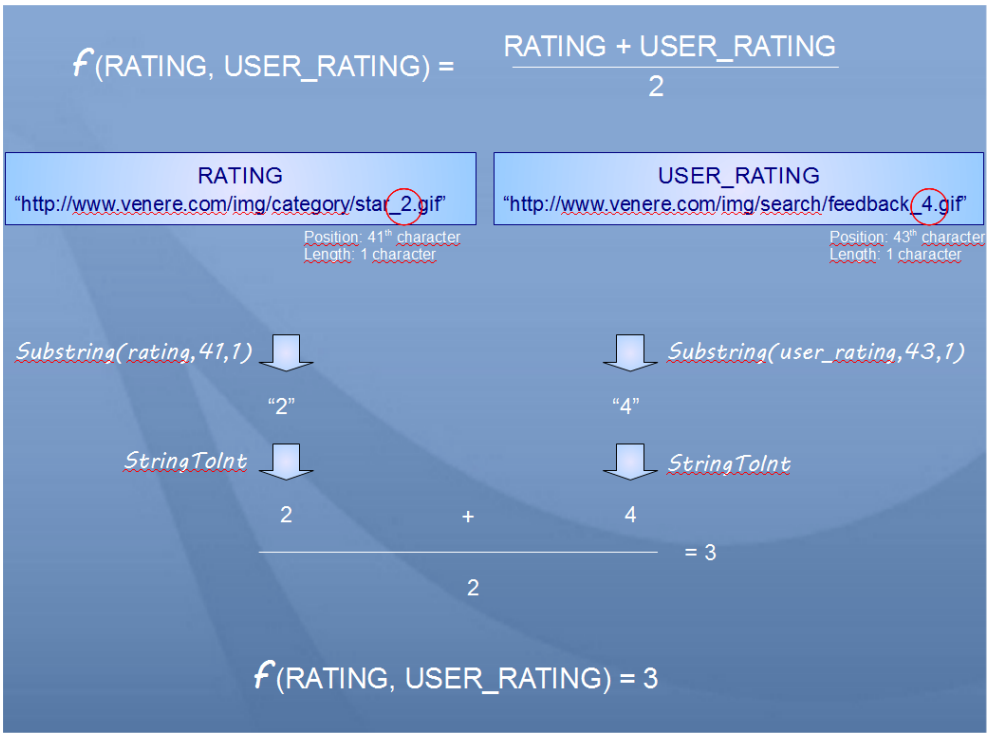


Fig. 2.37: Transformation Function applied for the Local Attributes *rating* and *user_rating* of the local class Venere.Hotels

that *rank* is a string containing a number from 1 to 5, *rating* and *user_rating* instead are urls linking to an image containing the stars; if we want *rating* and *user_rating* to look similar to *rank*, we need to apply a substring function, that extracts just the number of the stars from the whole url, the stringtoint function for casting it to an integer value, the sum and the division operator for calculating the average value.

Finally click on *Save* to save the inserted funcition.

The icon "f" for a non empty element of a Mapping Table means that a TF is defined for that element.

Note: if you want to read more about Transformation Functions see **Appendix A**.

2.9.3 Resolution Functions

Another issue is how to obtain the GA value when it is mapped onto more Local Attributes; in this case, we need to define a Resolution Function. Its application will permit to obtain, starting from the values of the Local Attributes (eventually transformed by a TF), the value for the GA.

By right-clicking on a GA and choosing *edit Resolution Function* a new window appears (Fig. 2.38). The interface is similar to the Transformation Function's one, but the available functions are different:

- If function :

```
functionif (${@condition}, ${@true}, ${@false})
```

- Coalesce function:

```
coalesce(${@function1},${@function2})
```

- String Concatenation Function:

```
@function1 + @function2
```

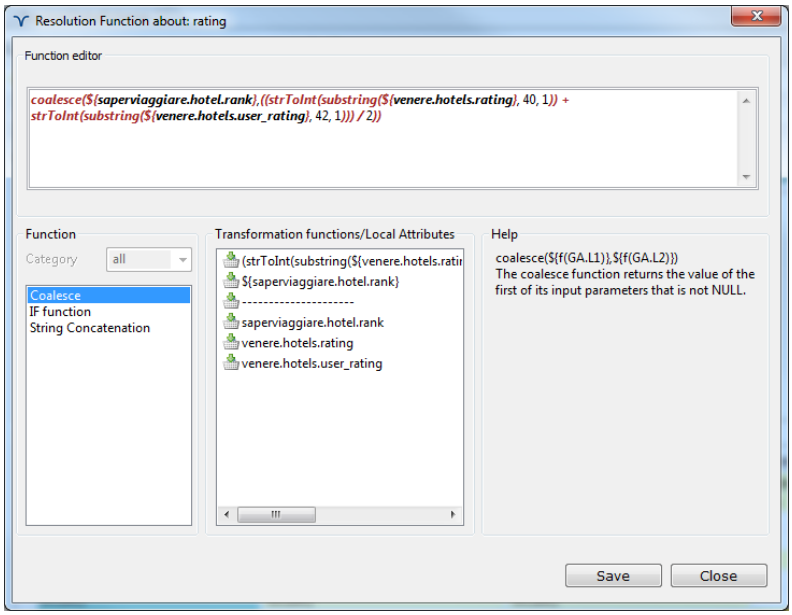


Fig. 2.38: Resolution Function Interface

The *If function* allows you to impose a condition among attributes: if the condition is true the function returns the first value, else returns the second one.

For example:

FUNCTIONIF (state <> 'Italy', LC2.name, LC1.name + LC3.hotel_name)

returns LC2.name if *state* value is different from 'Italy', otherwise returns LC.name+LC3.hotel_name.

The *Coalesce function* returns the first not null value of a given list of Local Attributes transformed by the Transformation Function.

For example:

COALESCE(LC2.name, LC1.name, LC3.hotel_name)

returns LC2.name if it's not null, if null it returns LC1.name if it's not null, otherwise returns LC3.hotel_name.

The *String concatenation function* returns concatenation of the values computed by the Transformation Functions applied to the local attributes (the String concatenation function accepta only string values as it's input).

For example:

STRINGCONCATENATION(LC2.name, LC1.name, LC3.hotel_name)

Note: Resolution Functions have to be defined only after the definition of Transformation Functions.

Chapter 3

Querying the Global Schema

3.1 Query Manager Interface

Finally, once completed the integration process, you can pose query on the obtained Global Schema by using the Query Manager Interface. To launch the Query Manager right click on the *GS* in the Global Schema Explorer view or click on *Launch Query Manager* hyperlink in the overview page (see Fig. 3.1).

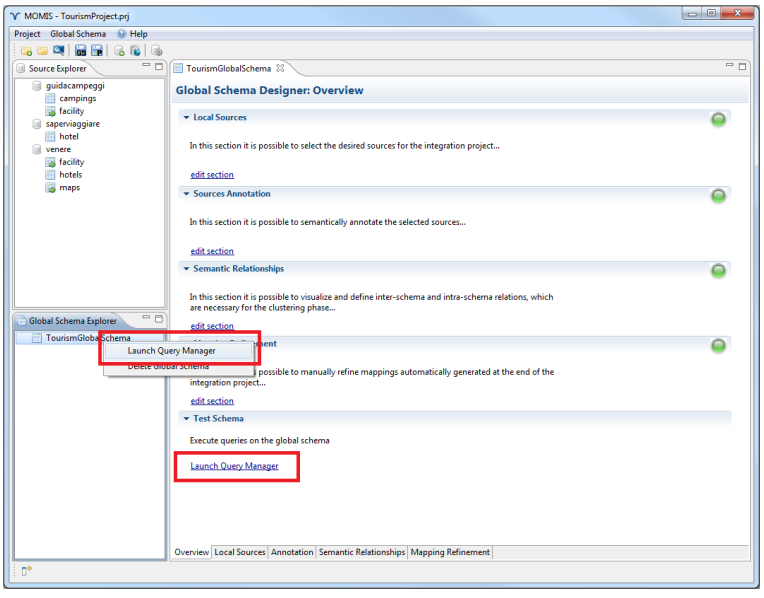


Fig. 3.1: Launch the Query Manager

The Query Manager interface lets you compose, run and save queries. It is composed by:

- The global source panel (see 1 in Fig. 3.2)
- The query text field (see 2 in Fig. 3.2)
- The result panel (see 3 in Fig. 3.2)

The global source tree helps you write the query (by clicking the global source tree nodes). To execute the inserted query click on *Run Query* button and you can see the result appear in the tabular panel (see 3 in Fig. 3.2).

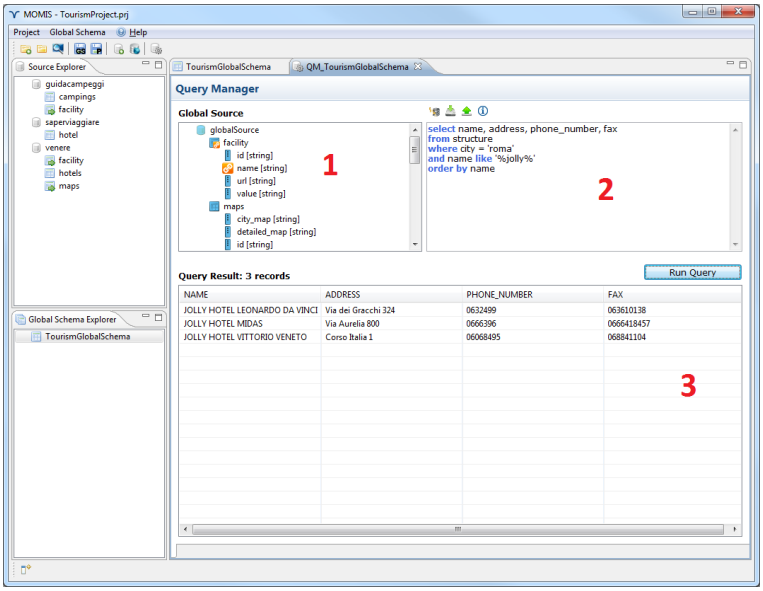


Fig. 3.2: The Query Manager Interface

In the MOMIS system, a global query (i.e. a query over the *GS*) is expressed by using the OQL_{I^3} language[1], an extension of the ODMG OQL language.

Let us execute the global query reported below (see 2 in Fig. 3.2)

```
SELECT name, address, phone_number, fax
FROM structure
WHERE city = 'roma'
AND name LIKE '%jolly%'
ORDER BY name
```

Click on *Run Query*[2], the query will be executed and the result will be shown in the tabular panel (see 3 in Fig. 3.2).

[1] You can find the full OQL_{I^3} syntax in Appendix D

[2] In order to obtain a result in the query phase all the local sources must be reachable.

3.2 Query Saving and Loading

As shown in Fig: 3.4 you can save the executed query. To save the query just click on *Save query* button (see 1 in Fig. 3.3) and insert the name you want to assign to it[3].

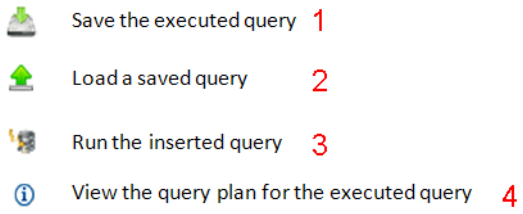


Fig. 3.3: Query Manager icon buttons

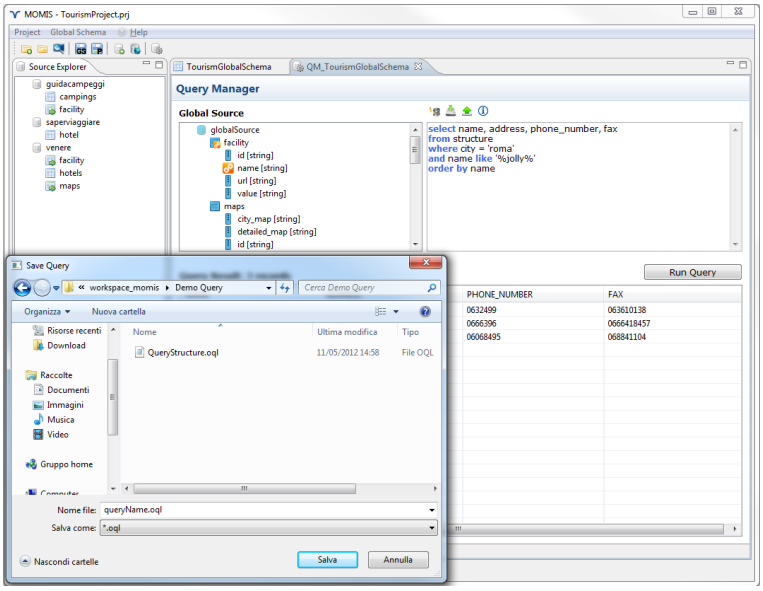


Fig. 3.4: Save the Executed Query

Also by clicking on *Load query* button (see 2 in Fig. 3.3) you can load an already saved query. Just select the file that contains the query and the query will appear in the text field (see 2 in Fig. 3.2).

[3]The file must have the .oql extension

3.3 Query Plan Viewer

Let us see how the result of a query is obtained?
MOMIS follows a Global-As-View (GAV) approach for the definition of mappings between the GS and local schemas: the GS is expressed in terms of the local schemas. The mapping is expressed by defining, for each global class GC, a mapping query q_G over the schemas of a set of local classes L belonging to GC. The query translation is performed by means of query unfolding, i.e., by expanding a global query on a global class $GC$of the GS according to the definition of the mapping query q_G for more details see Appendix C.

The query unfolding process generates for each global query a Query Plan composed by a set of queries:

- a set of local queries that have to be executed on the local sources simultaneously by means of wrappers (see Fig. 3.5),
- the mapping query that will fuse the partial results coming from the local sources, on the basis of the join function (see Fig. 3.6),
- a final query to apply the resolution functions and residual clauses (see Fig. 3.6).

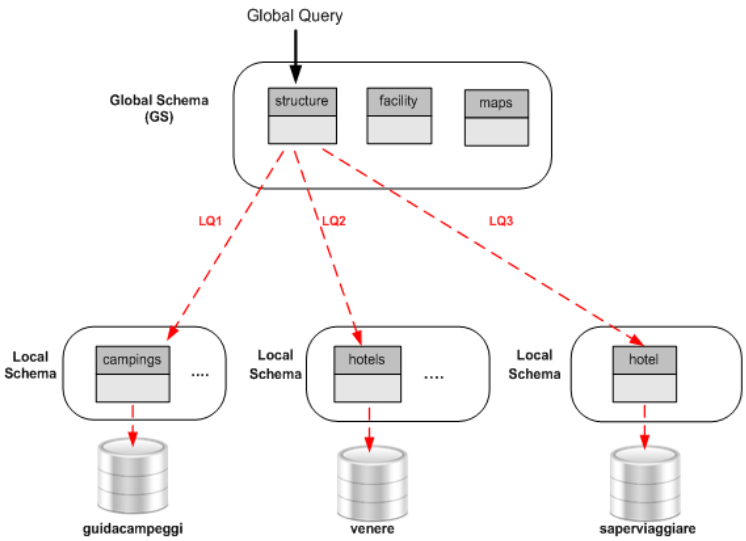


Fig. 3.5: Local Queries Execution

A relational database QMDB gives support to the Query Manager for the fusion of partial results, that are stored in temporary tables (see Fig. 3.6).

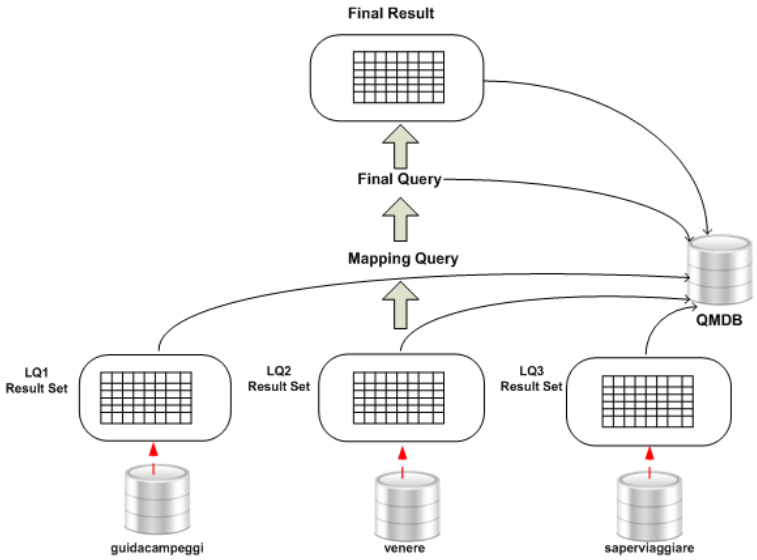


Fig. 3.6: Query Processing

For each executed query you can view the Query Plan, that means you can view the set of queries that compose the query plan and also through the data preview tool you can explore the content of the temporary tables, so you can understand how the data fusion process is performed.

In order to clarify, let us execute the query:

```
SELECT name, address, price, rating, fax
FROM structure
WHERE price > 100
AND price < 200
AND rating = '3'
AND city = 'Bologna'
ORDER BY price
```

To open the Query Plan Viewer (see Fig. 3.7)just click on the *Information* button (see 4 in Fig. 3.3). By clicking on the tree nodes (see 2 in Fig. 3.7) you can visualize in the text field (see 1 in Fig. 3.7) the selected query. Also each query can be executed on the QMDB by right clicking on a tree node and choosing *Data Preview* (see Fig. 3.8), you will get the first one hundred records and the total number of records of the table. Let's explore the query plan of the previous executed query. You can notice that three local queries have been generated, a local query is generated for each local class mapped in the global class and involved in the query, and this query translation is

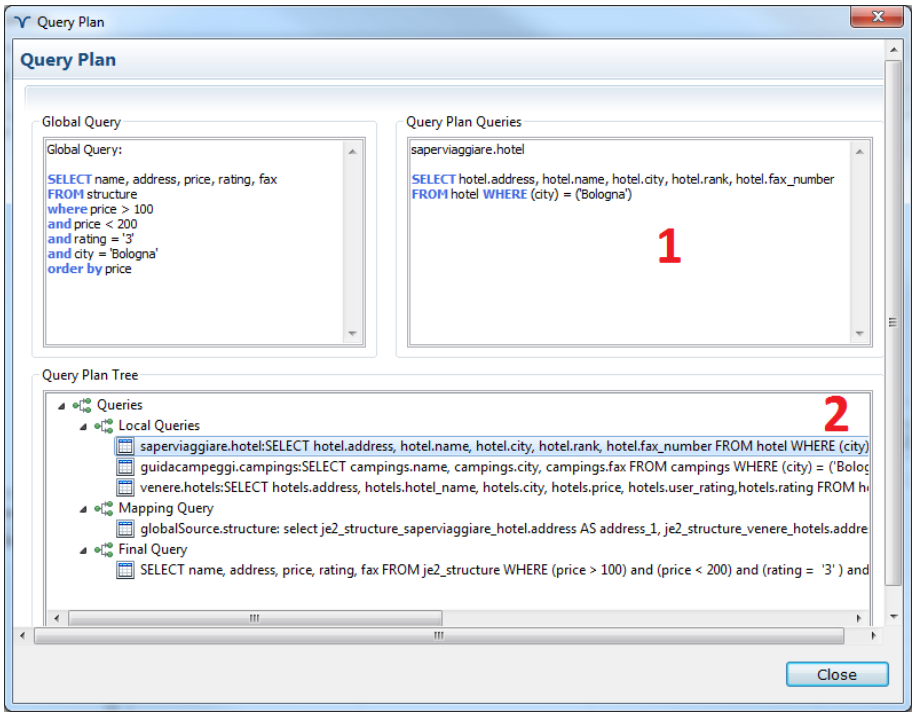


Fig. 3.7: Query Plan Viewer

performed by considering the mappings among the global class and the local schemas (see Fig. 3.9):

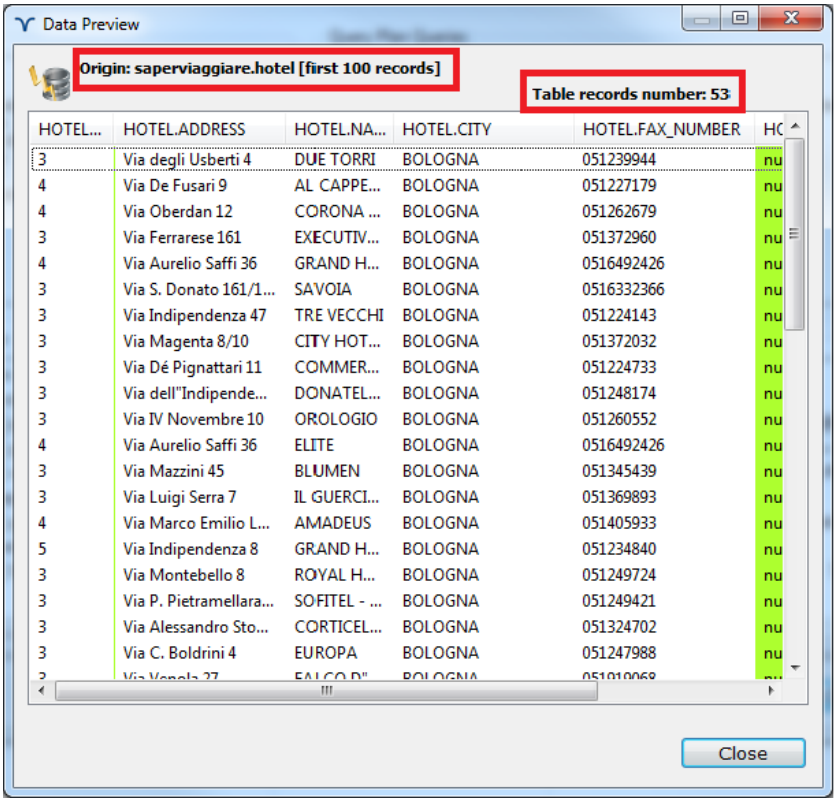


Fig. 3.8: Data Preview

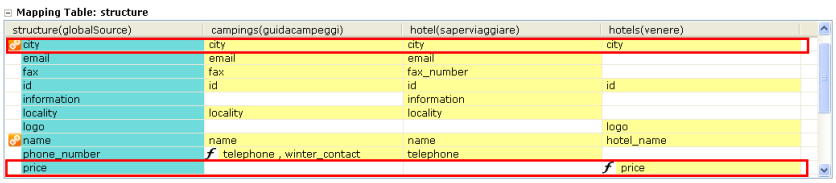


Fig. 3.9: Mapping Table of "structure"global class

```
LQ1:
guidacampeggi.campings

SELECT campings.name, campings.city, campings.fax
FROM campings WHERE (city) = ('Bologna')

LQ2:
venere.hotels

SELECT hotels.address, hotels.hotel_name, hotels.city,
hotels.price, hotels.user_rating,hotels.rating
FROM hotels
WHERE ((price) > (100)
and ((price) < (200)
and (city) = ('Bologna')))

LQ3:
saperviaggiare.hotel

SELECT hotel.address, hotel.name, hotel.city, hotel.rank, hotel.fax_number
FROM hotel
WHERE (city) = ('Bologna')
```

The local queries are executed by means of wrappers and partial results are materialized in temporary tables into the QMDB relational database. Through the *Data Preview* tool you can visualize the content of this tables. The partial results are fused together by executing the mapping query. At the end the final query that applies the resolution functions and the residual clauses is executed:

```
SELECT name, address, price, rating, fax
FROM je1_structure
WHERE (price > 100)
and (price < 200)
and (rating =  '3' )
and (city =  'Bologna')
ORDER BY price ASC
```

Chapter 4

Querying the Global Schema with the MOMIS QM Web Interface

4.1 Query Composition and Execution

To compose and execute your query you have to follow some main steps (Fig. 4.1):

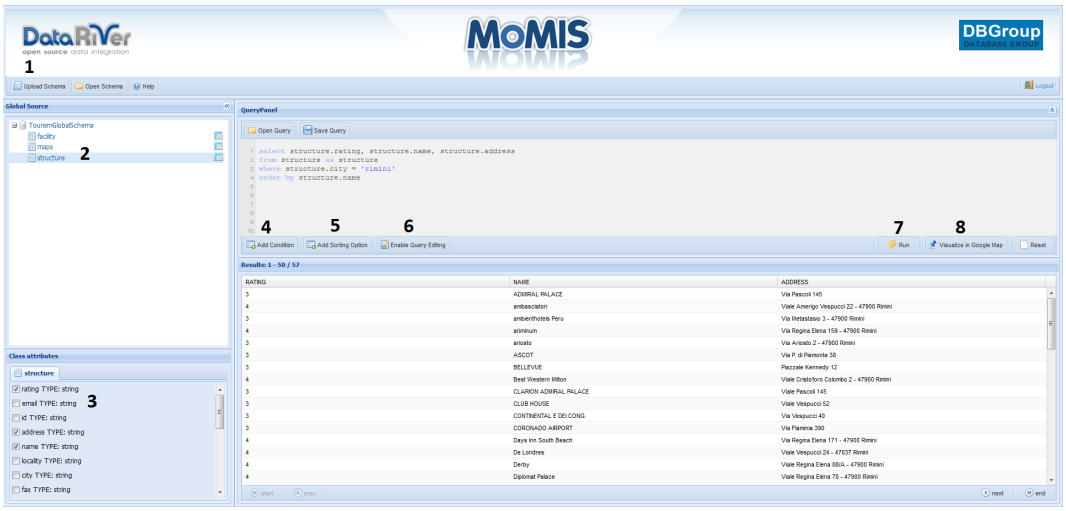


Fig. 4.1: Main steps for Query Composition and Execution

1. Upload the Global Schema you previously created with MOMIS; the left-panel of the interface will load the Global Source tree.

2. Click on a global class of the tree to see its attributes as checkbox down in the "Class Attributes" panel where you can select them.

3. The attributes you choose will be written in the *query editor*, together with the class.

   Note: If you want to perform a two-way JOIN between classes, you have to click on the first class, then click on a class referenced; an alert message will ask you if you want to join the classes; after clicking the "yes" button you will see the attributes of both classes in the "Class attributes" panel, in two different tabs, and the join condition will be automatically written in the query editor. In the *GlobalSources Tree* the node referred to the second class will be expanded, and the reference to the first class will be enlighten.

4. After choosing the attributes you can add and/or conditions by clicking the "add condition" button (Fig. 4.2). Conditions may concern any attribute of the classes involved, even the ones you didn't select.

5. If you want the query result to be ordered by a specific attribute set you may also add sorting options (Fig. 4.3), by clicking the corresponding button. Sorting options may concern only the attributes included in the query.

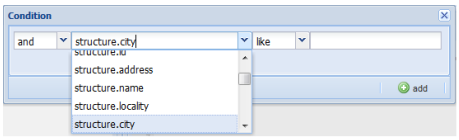


Fig. 4.2: Add Condition panel

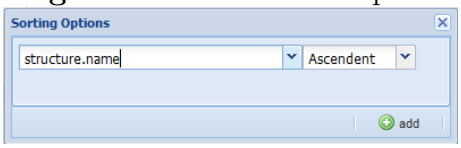


Fig. 4.3: Add Sorting Options panel

6. At any step of this sequence the query is automatically obtained by your interaction, and you may manually modify it in the *Query Editor*, but it's recommended to do it only after point 5, just before running the query.

7. Run the query and you will see the corresponding output in the "Results" grid.

The results are paginated 50 at a time, and you can look through them using the bottom bar of the grid. You may expand the grid by collapsing the top panel of the view port.

Lastly, if you click on one of the rows of the results Grid, you will see a new window (Fig. 4.4) containing all the attributes of that row, in order to look through them in a more readable way.

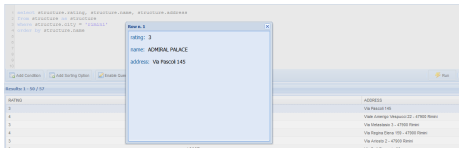


Fig. 4.4: Row Data shown in a new window

4.2 Visualize Results in a Map

Once the query has been executed you can visualize the query result on a Google Map[1]. First of all click on *Visualize in Google Map* button (see 8 in Fig. 4.1) and then select the attribute you want to geocode[2], so it's values will be used to place the markers on the Google Map (see Fig. 4.5), and click on *View Results in Map* button. Now your records are shown in the map (see Fig. 4.6)

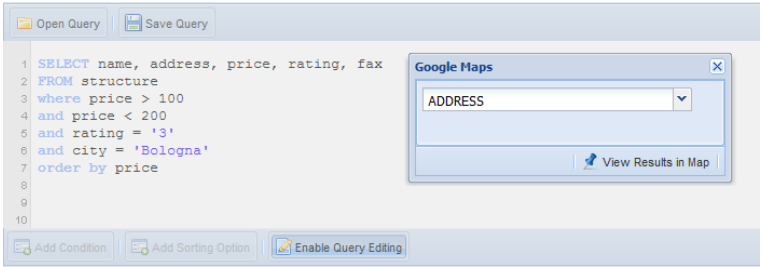


Fig. 4.5: Select Attribute

[1]https://developers.google.com/maps/

[2]Geocoding is the process of converting an address into geographic coordinates, which are used to place markers on the map.

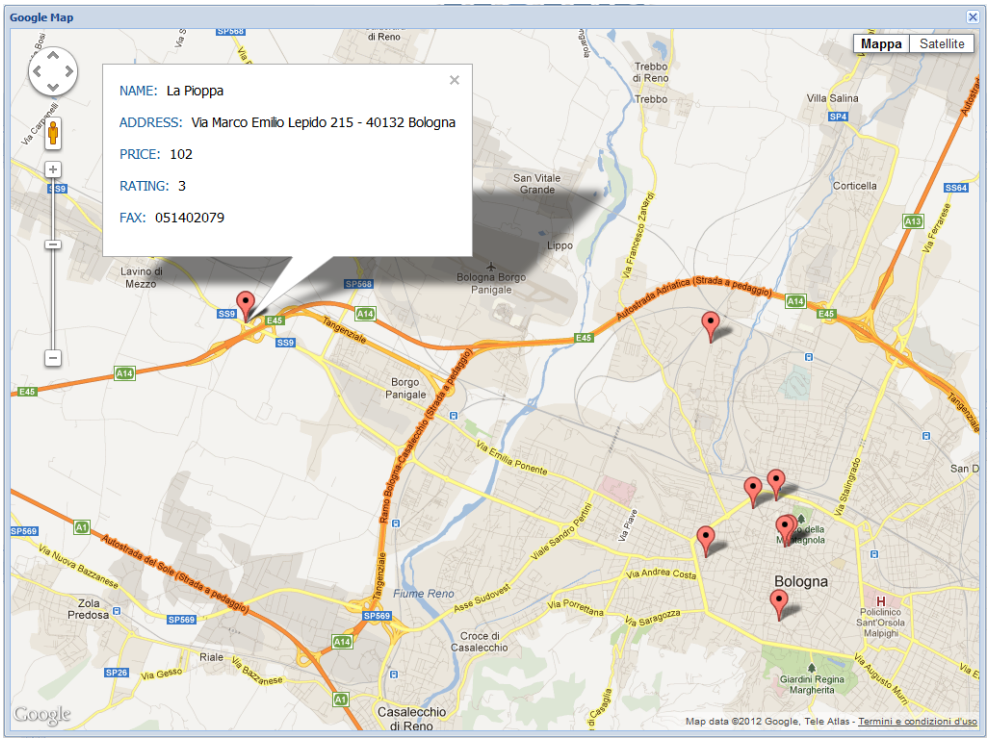


Fig. 4.6: Google Map

4.3 Query saving

At any time you can save your query, by clicking on the ”Save Query” button and inserting the name you want to assign to it (Fig. 4.7).

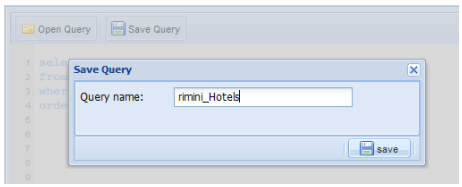


Fig. 4.7: Query Saving

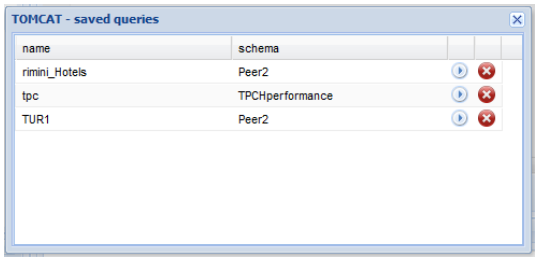


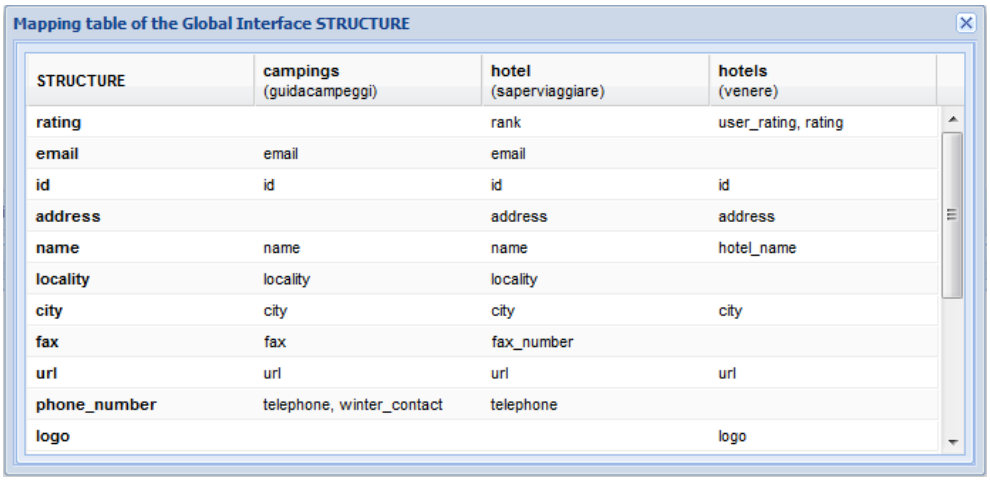
Fig. 4.8: Opening a saved Query

If you click on the ”Open Query” button a new window will appear (Fig. 4.8), containing all the queries you saved before, their names and the Global Schema on which they had been executed.

From there you can run or delete any of this queries by clicking on the corresponding button.

4.4 Mapping table

If you right-click on a *globalClass* of the *globalSource Tree*, a new window will appear, containing the Mapping Table of the Global Class (Fig. 4.9), that 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, the first row represents all the local classes belonging to the global class; the table elements are the local attributes (that are part of a local class) ”mapped” in a specific global attribute (row). More attributes may be mapped into the same global attribute.



STRUCTURE	campings (guidacampeggi)	hotel (saperviaggiare)	hotels (venere)
rating		rank	user_rating, rating
email	email	email	
id	id	id	id
address		address	address
name	name	name	hotel_name
locality	locality	locality	
city	city	city	city
fax	fax	fax_number	
url	url	url	url
phone_number	telephone, winter_contact	telephone	
logo			logo

Fig. 4.9: Hotels Mapping Table

Appendix A

Data Transformation Functions and Join Function (Theoretical Background)

Data Transformation Functions

For each not null element MT[GA][L] (an element MT[GA][L] represents the set of local attributes of L which are mapped onto the global attribute GA) we define a Data Transformation Function, denoted by MTF[GA][L], which represents how the local attributes of L are mapped into the global attribute GA. MTF[GA][L] is a function that must be executable/supported by thse local source of the local class L. For relational sources MTF[GA][L] can be also a SQL value expression.

As an example, let's take the global class *structure*, the global attribute *phone_number* and the local class *guidacampeggi.campings*. In the Mapping Table we can notice that:

MT[phone_number][guidacampeggi.campings] = telephone, winter_contact

One transformation function that can be applied is the *String Concatenation*:

telephone + "Winter Phone Number:" + winter_contact

Join Functions

MOMIS follows a Global-As-View (GAV) approach for the definition of mappings between the $GS$and local schemas: the $GS$is expressed in terms of the

local schemas. This means that for each Global Class GC a view VC over the Local Classes of GC must be defined.

One of the main innovation of the MOMIS system is that the view associated to GC is automatically defined by the system, i.e. this view don't need to be explicitly defined by the integration designer. *Data Fusion* is the process of fusing multiple records representing the same real-world object into a single and consistent representation. In the MOMIS system, *Data Fusion* is performed at the global classes level, by the mapping query q_G associated to a Global Class. The first step in the *Data Fusion* process is *Object Identification*, i.e. to identify instantiation of the same object in different sources (also known as record linkage, duplicate detection, reference reconciliation, and many others). In the MOMIS system, Object Identification is performed by *Join Function* (also called Join Condition). Join Conditions is a convenient way to perform Object Identification when it is possible to assume that error-free and shared object identifiers exist among different sources. Join Conditions are defined among pairs of local classes belonging to the same global class. More precisely, we specify a set of global attributes **JA** of GC, called **Join Attributes**, such that for each join attribute JA_i that belongs to **JA**, i= 1 .. k, and for each local class L L(G) belonging to G, the element MT[JA][L] is not null. Given **JA** = JA_1,JA_2, ... ,JA_k, for each pair of local classes L_1, L_2 belonging to L(G), the Join Condition between L_1 and L_2, denoted by JC(L_1,L_2) is defined as follows:

$$\text{MTF}[JA_1][L_1] = \text{MTF}[JA_1][L_2] \text{ and } \ldots \text{ and } \text{MTF}[JA_k][L_1] = \text{MTF}[JA_k][L_2]$$

Join conditions are specified at design time, and they are used at query time to identify tuples referring to the same real-world entity. If two tuples satisfy a join condition imposed over the corresponding relations, then the two tuples are assumed to be semantically equivalent. If they differ on corresponding attributes (attributes that are mapped to the same attribute in the global schema) then a ¨correct value¨ is obtained by applying appropriate conflict resolution functions.

As an example, let's take the global class *structure* and suppose that two local classes are mapped on it:
L_1=*venere.hotels*
L_2=*saperviaggiare.hotel*

If the global attribute *name* is chosen as Join Attribute, i.e. JA=name, then

JC(L_1,L_2): L_1.hotel_name = L_2.name

If the global attributes *name* and *city* are chosen as Join Attributes, i.e. JA_1=name and JA_2=city, then

JC(L_1,L_2): L_1.hotel_name = L_2.name AND L_1.city = L_2.city

On the basis of these *Join Conditions*, multiple records (coming from different local classes and representing the same real-world object) are combined in a single record by means of a *full outer join* operator.

Resolution Functions

The second step in a Data Fusion process is *Data Reconciliation*, i.e. to solve conflicts among instantiations of the same object in different sources. In the Data Fusion process considered in the MOMIS system, conflicts may arise for global attributes mapped onto more than one LC; *Data Reconciliation* is then performed by Resolution Functions; in fact, as explained in the previous section, for each GlobalAttribute such that there are more than one non empty elements (GlobalAttribute, LocalClass$_i$), we must define a *Resolution Function* to obtain, starting from the values computed by the Transformation Functions specified for (GlobalAttribute, LocalClass$_i$), the corresponding value for GlobalAttribute.

To resume, in the MOMIS system, Data Fusion is performed by combining the SQL operator of *full outer join* with *resolution functions*. From a theoretical point of view, this operation is called *full outer join merge operator* (Felix Naumann, Johann Christoph Freytag, and Ulf Leser. Completeness of integrated information sources. Inf. Syst., 29(7):583Ű615, 2004). This data fusion operation is automatically defined by the MOMIS system; the integration designer must only define *Transformation and Resolution Functions* and the *Join Attributes*.

On the other hand, the integration designer may change the view automatically associated to a GC by explicitly defining a specific Join Condition, by right-clicking on a GC, choosing edit Join Function, writing the function in the Join Function Panel and save it. The default Join Function provided by the system is based on Join Attributes as defined before.

The choice of Join Attributes is fundamental to perform a correct Data Fusion process. For more technical details on the Data Fusion process please see Section B. In the following we list the conditions for a correct definition of Join attributes; please note that in this version of the software if a condition

is not satisfied, no errors and/or no warnings will be displayed.

Condition 1) For a GC you need to select Join Attributes or to defined explicitly join conditions.

If no JoinAttribute is defined for a GC and no Join Condition is explicitly defined (see next section), then no join condition is defined for the full outer join operation that then corresponds to a Cartesian Product. As a consequence, no fusion is performed.

Condition 2) For each selected JoinAttribute, if for the corresponding element (JoinAttribute, LocalClass) is null, then all Join Condition related to this LC are considered true.

Condition 3) For each selected JoinAttribute, no Resolution Function needs to be defined.

Condition 4) A more subtle condition which need to be satisfied by the Join Attribute is that the corresponding local attributes must be a key in all local classes (see Section B.

Warning 1) For a GC you need to select Join Attributes or to define explicitly join conditions. If no JoinAttribute is defined for a GC and no Join Condition is explicitly defined, then no join condition is defined for the full outer join operation and thus no fusion is performed.

Warning 2) For each selected JoinAttribute, the element (JoinAttribute, LocalClass) of the mapping table must be not null for each LC belonging to the GC.

If for the selected JoinAttribute, the element (JoinAttribute, LocalClass) is null, then all Join Condition related to this element are considered true and then the fusion is not properly performed for the instances coming from this LocalClass.

As an example, let us consider that in the above GC Hotel the element (Name,LC2) is null; then, if the global attribute Name is chosen as Join Attribute, we have that JC(LC1,LC2) is true and thus no fusion is performed. If Name and City are chosen as Join Attributes we have that JC(LC1,LC2) is true AND LC2.city = LC1.city, i.e. JC(LC1,LC2) is LC2.city = LC1.city and thus fusion is performed only on the City attribute.

Appendix B

Data Fusion (Theoretical Background)

Data fusion is the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation. In MOMIS Data fusion is performed with the so-called emphFULL OUTER JOIN MERGE OPERATOR [8]

To give an intuition about this FULL OUTER JOIN MERGE OPERATOR, we introduce a simple example of GS. The relational schema of the local classes (relation tables)

LL1(K1,A), K1 primary key
LL2(K1,A), K1 primary key
LL3(K1,A), K1 primary key

are integrated in the global class C with the following Mapping Table:

	LL1	LL2	LL3
K1	K1	K1	K1
A	A	A	A

For the sake of simplicity, we consider that all local attributes are not transformed with respect to the global class, i.e. for each local attributes the identity transformation function is considered.

As said before, MOMIS follows a Global-As-View (GAV) approach for the definition of mappings between the GVV and local schemata: this means

that for the global class C a view VC over the local classes LL1, LL2 and LL3 must be defined; this view is defined with the FULL OUTER JOIN MERG OPERATOR: in the following we will show an SQL implementation of this operator.

First of all, if K1 is defined as Join Attribute , the join condition among the local classes will be performed on the basis of the corresponding K1 attributes of the local classes, as follows

```
select * from
LL1 full outer join LL2 on (LL1.K1 = LL2.K1)
full outer join  LL3 on (LL3.K1 = LL2.K1 OR LL3.K1 = LL1.K1)
```

It can be demonstrated that the order of local classes in the full outer join operation is not relevant[1], i.e., the same result will be obtained with the following expression:

```
select * from
LL3 full outer join LL2 on (LL3.K1 = LL2.K1)
full outer join  LL1 on (LL1.K1 = LL2.K1 OR LL1.K1 = LL3.K1)
```

Then in the following we will use the first expression given above.

To complete the definition of the view associated to the global class, we need to define its K1 and A values. The K1 value is one of the not null values among LL1.K1,LL2.K1 and LL3.K1 ; then it can be computed as isnull(isnull(LL1.K1,LL2.K1), LL3.K1) (the order of LL1.K1,LL2.K1 and LL3.K1 is not relevant).

The A value depends on the chosen resolution function; as an example, let us consider the coalesce resolution function:

```
A = coalesce(LL1.A,LL2.A,LL3.A)
```

Then the view associated to the global class is:

```
select isnull(isnull(LL1.K1,LL2.K1),LL3.K1) AS K1,
coalesce(LL1.A,LL2.A,LL3.A) AS A
from LL1 full outer join LL2 on (LL1.K1 = LL2.K1)
full outer join  LL3 on (LL3.K1 = LL2.K1 OR LL3.K1 = LL1.K1)
```

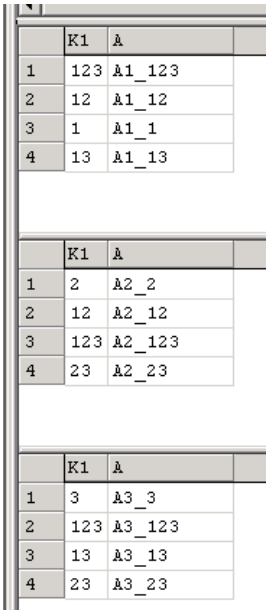
In other words, at the query

```
select * from C
```

on the global class C will correspond the above query on its local classes.

[1]This is true when the join attribute K1 is a key in each local class

To give an example of the result of this view, let us consider the following instances of local classes (LL1,LL2 and LL3 respectively):



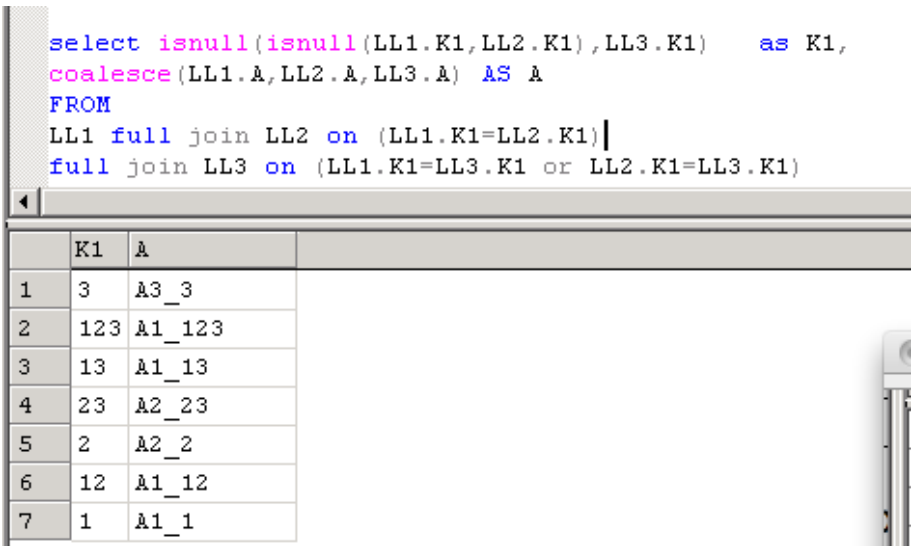
	K1	A
1	123	A1_123
2	12	A1_12
3	1	A1_1
4	13	A1_13

	K1	A
1	2	A2_2
2	12	A2_12
3	123	A2_123
4	23	A2_23

	K1	A
1	3	A3_3
2	123	A3_123
3	13	A3_13
4	23	A3_23

Please notice that the value 123 for K1 means that a record with this value for K1 is present in local classes LL1, LL2 and LL3, value 12 means that the record is present in local classes LL1 and LL2, and so on.

The result is as follows



```
select isnull(isnull(LL1.K1,LL2.K1),LL3.K1)    as K1,
coalesce(LL1.A,LL2.A,LL3.A) AS A
FROM
LL1 full join LL2 on (LL1.K1=LL2.K1)
full join LL3 on (LL1.K1=LL3.K1 or LL2.K1=LL3.K1)
```

	K1	A
1	3	A3_3
2	123	A1_123
3	13	A1_13
4	23	A2_23
5	2	A2_2
6	12	A1_12
7	1	A1_1

We can observe that, for each K1 value in the local classes, there is only

a record with this K1 value in the result, and the A value is computed on the basis of the resolution function.

To highlight the importance of choosing a key as Join Attribute, let us suppose that K1 is not a key in a local class, say LL2, i.e.

`LL2(K1,A)`

This means that inLL2 we can more than one record with the same value for K1:

	K1	A
1	2	A2_2
2	12	A2_12
3	123	A2_123
4	23	A2_23
5	23	A2_23_BIS
6	123	A2_123_BIS

Now the result of the view is as follows:

```
select isnull(isnull(LL1.K1,LL2.K1),LL3.K1)    as K1,
coalesce(LL1.A,LL2.A,LL3.A) AS A
FROM
LL1 full join LL2 on (LL1.K1=LL2.K1)
full join LL3 on (LL1.K1=LL3.K1 or LL2.K1=LL3.K1)
```

	K1	A
1	3	A3_3
2	123	A1_123
3	123	A1_123
4	13	A1_13
5	23	A2_23
6	23	A2_23_BIS
7	2	A2_2
8	12	A1_12
9	1	A1_1

We can observe that, for each repeated value in LL2.K1, we have a repeated value in K1 of the global class, and, due to the resolution function,

some of these repeated values (see second and third records in the above result) can have the same A value.

To illustrate several scenario and how to define *Mapping Table, Resolution Functions* and *Join Condition* for these scenario

Customer1 (Code,Address1), Code primary key
Customer2 (Code,Address2), Code primary key
Customer3 (Code,Address3), Code primary key

1. The Join Attribute Code is key in each local class.
   (a) A unique address chosen among addresses from local sources on the basis of some criteria: in this case Address local attributes are mapped onto a unique global attribute Address defined by an appropriate resolution function
   (b) A unique address containing all addresses from local sources: in this case Address local attributes are mapped onto a unique global attribute Address defined by the (string) concatenation resolution function
2. The Join Attribute Code is not a key in some local classes.
   (a) A unique address chosen among addresses from local sources on the basis of some criteria
   (b) A unique address containing all addresses from local sources

Appendix C

Query Unfolding (Theoretical Background)

The query unfolding process is performed for each global query Q over a global class GC of the global schema. Given the global query Q and the mapping defined on the Mapping Table MT, the query unfolding process generates the set of local queries LQs to be executed on the sources, the mapping query q_G for merging the partial results, and the final query to apply the resolution functions and residual clauses. We give an intuitive explanation of the main steps in the query unfolding process, you can find more information in [9].

C.1 Query Unfolding steps

Given a global query Q as following:

$$\begin{aligned} Q \;=\; & SELECT < Q\_SELECT - list > \\ & FROM\ G \\ & WHERE\ \ < Q\_condition > \end{aligned}$$

where $< Q\_condition >$ is a boolean expression of positive atomic constraints $(GA_1\ op\ value)$ or $(GA_1\ op\ GA_2)$, with GA_1 and GA_2 attributes of GC.

The query unfolding process is made up of the following three steps:

- Step 1: generation of the local queries LQs;
- Step 2: generation of the mapping query q_G;
- Step 3: generation of the final query.

Step 1. Generation of the local queries LQs. Each local query LQ is expressed as following:

$$\begin{aligned} LQ \;=\; & SELECT < SELECT-list > \\ & FROM\ L \\ & WHERE\ < condition > \end{aligned}$$

where L is a local class related to GC. The $< SELECT\_list >$ is computed by considering the union of:

- the global attributes in $< Q\_SELECT-list >$ with a not null mapping in L,
- the global attributes used to express the join condition for L,
- the global attributes in $< Q\_condition >$ with a not null mapping in L.

The set of global attributes is transformed in the corresponding set of local attributes on the basis of the Mapping Table MT. The $< condition >$ is computed by performing an atomic constraint mapping: each atomic constraint of $< condition >$ is rewritten into one that is supported by the local source. The atomic constraint mapping is performed on the basic of the Data Transformation Functions and Resolution Functions defined in the Mapping Table. For example, if a numerical global attribute GA is mapped onto the local classes L_1 and L_2, and an average resolution function AVG is defined for GA, the constraint $(GA = value)$ cannot be pushed at the local source, because the AVG function has to be computed at a global level. In this case, the constraint will be mapped as true in both the local sources, and the resolution function will be computed only at a global level. On the other hand, if GA is an *homogeneous attribute* (no resolution function defined), the constraint will be pushed at the local sources.

Thus, an atomic constraint $(GA_1\ op\ value)$ will be rewritten on the local class L as follows:

$$\begin{cases} (MTF[GA][L]\ op\ value) & \text{if } MT[GA][L] \textit{ is not null} \text{ and} \\ & \textit{the op operator is supported by } L \text{ and} \\ & \textit{the data transformation function } MTF \textit{ is supported by } L \\ & \textit{the resolution function } f \textit{ is supported by } L \\ true & \text{else} \end{cases}$$

Atomic constraints of the kind $(GA_1\ op\ GA_2)$ will be rewritten in a similar way.

Step 2. Generation of the mapping query q_G. The LQs partial results will be merged together by means of the *full outerjoin-merge* operator.

Step 3. Generation of the final query. The final query performs the application of *resolution functions* and *residual clauses*:

- for *Homogeneous Attributes* (no conflict on data values), the system can consider one of the values without preference;
- for *non Homogeneous Attributes*, the system have to apply the associated Resolution Function.

C.2 Multiple Class Queries

Given the global classes $GC_1, GC_2, ..., GC_n$ we consider a Global Query Q :

```
Q  =  select <Q_select-list>
      from G1,G2, ..., Gn
      where <Q_condition>
      order by <order_by_list>
```

where `<Q_condition>` is a Boolean expression of positive atomic constraints: `(Gi.GA1 op value)` or `(Gi.GA1 op Gj.GA2)`, where `GA1` and `GA2` are global attributes.

The query unfolding is performed in two steps. In the first step, with standard rewriting rules, the `<Q_condition>` is unfolded w.r.t. the global classes `Gi` of the query `Q`. In this way we obtain the following rewriting:

```
Q'  = select <Q_select-list>
      from Q1,Q2, ..., Qn
      where <join_condition>
      and <residual_predicate>
      order by <order_by_list>
```

where

- `Qi` is a Single Class Query:

```
Qi  = select <Qi_select-list>
      from Gi
      where <Qi_condition>
```

where

  - `<Qi_select-list>` is the union of the attributes in `<Q_select-list>`, in `<join_condition>` and in `<residual_predicate>`.
  - `<Qi_condition>` is a condition which can be solved w.r.t. the global class `Gi`, i.e., is a condition which uses global attributes of `Gi`.

- `<join_condition>` is a conjunction of constraints `(Qi.GA1 op Qj.GA2)`
- `<residual_predicate>` are the *residual predicates*.

In the second step, each single class query is unfolded w.r.t. local classes by taking into account the mappings $\mathcal{M}$; this step has been discussed in the previous section.

Appendix D

The OQL$_{I3}$ query language syntax

In the following we included the OQL$_{I^3}$ syntax accepted by the MOMIS Query Manager. From the **1.2 version** the aggregate functions has been added to the OQL$_{I^3}$syntax accepted by the MOMIS Query Manager.

```
<asterisk> ::= *
<comma> ::= ,
<period> ::= .
<quote> ::= ’
<underscore> ::= _
<equals operator> ::= =
<not equals operator> ::= !=
<less than operator> ::= <
<less than or equals operator> ::= <=
<greater than operator> ::= >
<greater than or equals operator> ::= >=
<left paren> ::= (
<right paren> ::= )
<query> ::=
SELECT [ <quantifier> ] <select clause>
<from clause>
[ <where clause> ]
[ <order by clause> ]
[ <group by clause> ]
[ <having clause> ]

<quantifier> ::= DISTINCT
```

```
<select clause> ::= <asterisk> | <select list> [ { <comma> <select list> }

<select list> ::=
<qualifier> <period> <asterisk> | <select sublist> [ <correlation specification> ]

<select sublist> ::= <attribute reference> | <set function specification>

<attribute reference> ::= [ <qualifier> <period> ] <global attribute name>

<qualifier> ::= <global interface name> | <correlation name>

<global interface name> ::= <identifier>

<global attribute name> ::= <identifier>

<correlation name> ::= <identifier>

<set function specification> ::=
COUNT <left paren> <asterisk> <right paren> | <general set function>

<general set function> ::=
<set function type> <left paren> [ <quantifier> ] <attribute reference> <right paren>

<set function type> ::= AVG | MAX | MIN | SUM | COUNT

<from clause> ::=
FROM <global interface reference> [ { <comma> <global interface reference> }... ]

<global interface reference> ::=
<global interface name> [ <correlation specification> ]

<correlation specification> ::= AS <correlation name>

<where clause> ::= WHERE <search condition>

<search condition> ::=
                  <boolean term>
                | <search condition> OR <boolean term>

<boolean term> ::=
               <boolean factor>
             | <boolean term> AND <boolean factor>
```

```
<boolean factor> ::= <boolean primary>

<boolean primary> ::=
                    <predicate>
                  | <left paren> <search condition> <right paren>

<predicate> ::=
              <comparison predicate>
            |  <like predicate>
            |  <null predicate>
            |  <join predicate>

<comparison predicate> ::=
                        <attribute reference> <comp op> <value expression>

<comp op> ::=
             <equals operator>
           |  <not equals operator>
           |  <less than operator>
           |  <greater than operator>
           |  <less than or equals operator>
           |  <greater than or equals operator>

<null predicate> ::= [NOT] IS NULL <attribute reference>

<like predicate> ::= <attribute reference> LIKE <pattern>

<pattern> ::= <quote> <value expression> <quote>

<join predicate> ::= <attribute reference> <comp op> <attribute reference>

<order by clause> ::= ORDER BY <sort specification list>

<sort specification list> ::=
<sort specification> [ { <comma> <sort specification> }... ]

<sort specification> ::= <sort key> [ <ordering specification> ]

<sort key> ::= <attribute reference>

<ordering specification> ::= ASC | DESC

<group by clause> ::= GROUP BY <grouping attribute reference list>
```

```
<grouping attribute reference list> ::=
<grouping attribute reference> [ { <comma> <grouping attribute reference> }... ]

<grouping attribute reference> ::= <attribute reference>

<having clause> ::= HAVING <search condition>
```

Bibliography

[1] R. Benassi, S. Bergamaschi, A. Fergnani, and D. Miselli. Extending a lexicon ontology for intelligent information integration. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 278–282. IOS Press, 2004.

[2] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an Integrated Ontology. *IEEE Internet Computing Journal*, pages 42–51, Sep-Oct 2003.

[3] S. Bergamaschi, D. Beneventano, A. Corni, E. Kazazi, M. Orsini, L. Po, and S. Sorrentino. Open Source release of the MOMIS Data Integration System. In G. Mecca and S. Greco, editors, *Proc. of the Nineteenth Italian Symposium on Advanced Database Systems, SEBD, 26-29 June 2011, Maratea, Italy*, pages 175–186, 2011.

[4] S. Bergamaschi, D. Beneventano, F. Guerra, and M. Orsini. Data integration. In D. W. Embley and B. Thalheim, editors, *Handbook of Conceptual Modeling Springer, Berlin, Germany*, pages 443–478, 2011.

[5] S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data Knowl. Eng.*, 36(3):215–249, 2001.

[6] M. Lenzerini. Data Integration: A Theoretical Perspective. In L. Popa, editor, *PODS*, pages 233–246. ACM, 2002.

[7] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.

[8] F. Naumann, J. C. Freytag, and U. Leser. Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615, 2004.

[9] M. Orsini. *Query Management in Data Integration Systems: the MOMIS approach.* PhD thesis, Doctorate School in ICT - Computer Engineering and Science , University of Modena and Reggio Emilia, 2009.