

# Getting Through the THALIA Benchmark with MOMIS\*

D. Beneventano   S. Bergamaschi   M. Vincini   M. Orsini   R. C. Nana Mbinkeu

Dipartimento di Ingegneria dell'Informazione  
Università di Modena e Reggio Emilia, Italy  
Via Vignolese 905 Modena  
bergamaschi.sonia@unimore.it

## ABSTRACT

During the last decade many data integration systems characterized by a classical wrapper/mediator architecture [1] based on a Global Virtual Schema (*Global Virtual View* - *GVV*) have been proposed. The data sources store data, while the *GVV* provides a reconciled, integrated, and virtual view of the underlying sources. Each mediator system proposed (see [2] for a survey) contribute to the state of the art advancement by focusing on one or more challenge of the data integration problem, ranging from syntactical and semantic heterogeneities. At present, solving heterogeneities is a partially manual task, requiring a great amount of customization for data reconciliation and for writing specific not-reusable programming code. The last Lowell Report [3] has provided the guidelines for the definition of a public benchmark for data integration problems, called THALIA (Test Harness for the Assessment of Legacy information Integration Approaches) [4].

In this paper we show how the MOMIS mediator system [5,6,7] can support all the twelve queries of the THALIA benchmark by simply extending and combining the declarative translation functions available in MOMIS and without any overhead of new code. This is a remarkable result, in fact, as far as we know, no system has provided a complete answer to the benchmark.

## 1. INTRODUCTION

During the last decade many data integration systems characterized by a classical wrapper/mediator architecture [1] based on a Global Virtual Schema (*Global Virtual View* - *GVV*) have been proposed. The data sources store the real data, while the *GVV* provides a reconciled, integrated, and virtual view of the data sources. Modelling the mappings among sources and the *GVV* is a crucial aspect. Two basic approaches for specifying the mappings in a Data Integration System have been proposed in the literature: *Local-As-View* (*LAV*), and *Global-As-View* (*GAV*), respectively [8,9].

The *LAV* approach is based on the assumption that a global schema representing the conceptualization of a domain exists and the contents of each local source must be described in terms of the

global schema. This assumption holds only in the case that the *GVV* is stable and well-established in the organization. This constitutes the main limitation of the *LAV* approach. Another negative issue is the complexity of query processing which needs reasoning techniques. On the other hand, as a positive aspect, the *LAV* approach facilitates the extensibility of the system: adding a new source simply means enriching the mapping with a new assertion, without other changes [10].

In the *GAV* approach the contents of the elements of the *GVV* is not predefined and is described in terms of a view of the local sources. *GAV* favours the system in carrying out query processing, because it tells the system how to use the sources to retrieve data (unfolding). However, extending the system with a new source is now more difficult: the new source may indeed have an impact on the definition of various classes of the *GVV*, whose associated views need to be redefined.

Each mediator system proposed tried to solve as much as possible the integration problem, focusing on different aspects to provide a (partial) answer to one or many challenges of the problem, ranging from system-level heterogeneities, to structural syntax level heterogeneities at the semantic level. The approaches still rely on human interventions, requiring customization for data reconciliation and writing specific not reusable programming code. The specialization of the proposed mediator system makes the comparison among the systems difficult. Therefore, the last Lowell Report [3] has provided the guidelines for the definition of a public benchmark for the information integration problem. The proposal is called THALIA (Test Harness for the Assessment of Legacy information Integration Approaches) [4], and it provides researchers with a collection of downloadable data sources representing University course catalogues, a set of twelve benchmark queries, as well as a scoring function for ranking the performance of an integration system. THALIA benchmark focuses on syntactic and semantic heterogeneities in order to pose the greatest technical challenges to the research community.

Starting from our previous experience in the system information integration area, where we developed the MOMIS, a mediator system following a *GAV* approach [5,6,7], we developed an

---

\* This work was partially supported by MIUR co-funded project NeP4B (<http://www.dbgroup.unimo.it/nep4b>).

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the VLDB copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Very Large Data Base Endowment. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires a fee and/or special permission from the publisher, ACM.

VLDB '07, September 23-28, 2007, Vienna, Austria.

Copyright 2007 VLDB Endowment, ACM 978-1-59593-649-3/07/09.

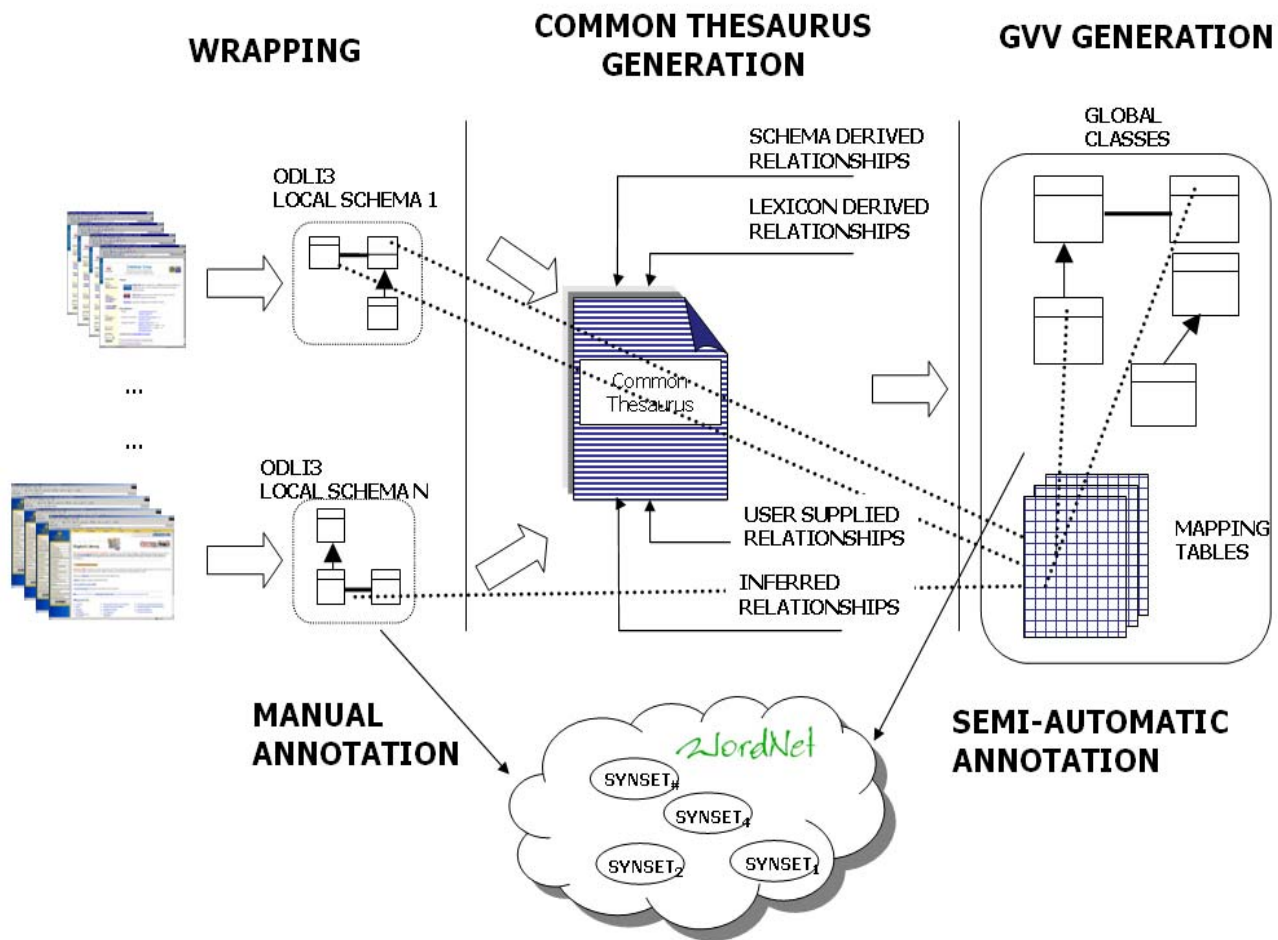


Figure 1: Ontology Generation Process

extension of the system devoted to the support of syntactic and semantic data heterogeneities by means of *declarative Mapping Data Transformation Functions (MDTFs)* that avoids the overhead due to write ad-hoc hard-coded transformation functions.

MDTFs allow MOMIS to deal with all the twelve queries of the THALIA benchmark: by using a simple combination of these functions and without any overhead of new code we are able to fully satisfy the goal. This is a remarkable result, in fact as far we know no mediator system has provided a complete answer to the benchmark.

The paper is structured as follows. Section 2 presents the THALIA benchmark and Section 3 the MOMIS integration methodology. Section 4 defines the mapping refinements that in Section 5 are used for the global query translation. Section 6 reports the experimental results with THALIA, Section 7 the related works and, finally, Section 8 concludes our work.

## 2. The THALIA Benchmark

THALIA is a public available testbed and benchmark for information integration systems [4]. It provides over 40 downloadable sources representing University course catalog from computer science around the world. The goal of the benchmark is a systematic classification of the different types of syntactic and semantic heterogeneities that are described by the twelve queries provided.

For each case, a benchmark query is formulated and it is applied (easily) to a *target schema* as well as a *challenge schema* which provides the heterogeneity solved by the integration system. The heterogeneities are divided into three different categories: Attribute Heterogeneities (Query 1, 2, 3, 4 and 5), Missing Data (Query 6, 7, 8) and Structural Heterogeneities (Query 9, 10, 11, 12).

**Attribute Heterogeneities:** inconsistencies that exist between two single attributes in different schemata.

The simplest is *attribute synonyms*, where the same information is stored in attributes with different names. In **Query 1** ‘instructor’ could be thought as a synonyms of ‘lecturer’.

A *simple mapping* heterogeneity is referred to attributes that differ by a mathematical transformation. In **Query 2** the two attributes contain a time value in 12 and 24 hours, respectively.

*Union types:* in many cases attributes in different schemas use different data types to store the same information: in **Query 3** the target schema uses a string attribute to indicate the course name while the challenge schema uses a string description including the name and the link of the course.

*Complex mapping:* it is the case where related attributes differ by a complex transformation of their values: for example **Query 4** contains a number of ‘credit hour’ in the target schema and a string description of the expected work in the challenge schema.

A typical real case involves two sources where the same information is denoted in different languages, giving rise to a *language expression* heterogeneity. **Query 5** proposes a target schema where the course name is expressed in English and a challenge schema in German.

**Missing data:** heterogeneities due to missing information (value or structure) in one of the schemas.

*Nulls treatment and Semantic incompatibility:* distinguishes the cases where an attribute does not exist in a source from the one where the attribute has a null value in a particular record. In **Query 6**, course book is not present in the challenge schema and it is present only for some records in the target schema. **Query 8** proposes a target schema with a student classification that is not present in the challenge schema.

*Virtual column:* the information could be explicit in a source and only implicitly available in the others. **Query 7** gives an example where the course prerequisites are present together with the course description in the challenge schema.

**Structural Heterogeneities:** heterogeneities due to missing information (value or structure) in one of the schemas.

*Structural heterogeneity of an attribute:* the same attribute may be located in different position in different schemas. **Query 9** propose a target schema with the room attribute in the course relation and a challenge schema where the room information is an element of section that is a part of a course.

*Handling sets:* a single attribute contains a string that describes a set of values in a schema, while the same information is split into single attributes in an other schema. **Query 10** contains a target schema with a single lecturer attribute of course and a challenge schema where a professor is defined for each section of the course.

*Attribute name does not define semantics:* a typical case where the attribute name does not refer to the semantics of the contained information. **Query 11** contains a challenge schema where the lecturer are spread in three different attributes whose names are the teaching periods.

*Attribute composition:* complex data can be represented either as a single string or a set of attributes. **Query 12** contains a challenge schema where the title, day and time of a course are contained in a single attribute rather than in three different attributes.

### 3. MOMIS Integration Methodology

The Mediator Environment for Multiple Information Sources (MOMIS) is a framework for extracting information and integrating heterogeneous, structured and semistructured data sources ([www.dbgroup.unimo.it/momis/](http://www.dbgroup.unimo.it/momis/)). Unlike other data-integration systems that follow the local-as-view (LAV) approach, MOMIS implements a semiautomatic methodology that follows the global-as-view (GAV) approach: the obtained global schema is expressed in terms of the data sources. MOMIS generates a global schema that provides an integrated GVV composed of a set of global classes that represent the information contained in the underlying sources and the mappings that establish the connections among the global attributes of the global classes and the source schemata. Since a GVV conceptualizes a

domain, it might be thought of as an ontology for the integrated sources.

MOMIS uses ODL<sub>13</sub>, which is based on the Object Definition Language (ODL) to describe both the input (the source schemata) and the result of the synthesis process (the GVV).

In the following we describe the MOMIS Ontology generation process by means of three different THALIA's queries (query 4, 7 and 12), each one referring to a different heterogeneity category.

#### 3.1 GVV Generation

The GVV Generation process can be outlined as follows (see Figure 1):

1. *Extraction of Local Source Schemata:* Wrappers acquire schemata of the involved local sources and convert them into ODL<sub>13</sub>. Schema description of structured sources (e.g. relational database and object-oriented database) can be directly translated, while the extraction of schemata from semistructured sources need suitable techniques as described in [11]. To perform information extraction from XML Schema files, like other systems [12], we developed a wrapper that automatically translate the XSD schema into relational structures and import data into a relational database. All schemata and data provided by THALIA benchmark (10 schemas and 701 records) are automatically wrapped into a relational database.
2. *Local Source Annotation:* Terms denoting schema elements in data sources are semantically annotated according to a common lexical reference in order to provide a shared meaning to each of them. We choose the WordNet ([wordnet.princeton.edu](http://wordnet.princeton.edu)) database as lexical reference. The system automatically detects, for each term in the sources, the (most commonly) used meaning present in WordNet. Algorithms for automatic annotation prepares terms by applying stop-words and stemming functionalities to enhance the accuracy result. Then the Integration Designer can manually revise the meaning(s) for each annotated term. Considering the THALIA schemata, the recall rate of the terms automatically annotated in the sources is 80%, with a precision of 82%.
3. *Common Thesaurus Generation:* MOMIS builds a Common Thesaurus that describes intra and inter-schema knowledge in the form of: synonyms (SYN), broader terms/narrower terms (BT/NT), meronymy/holonymy (RT) relationships. The Common Thesaurus is incrementally built by starting from schema-derived relationships, i.e. automatic extraction of intra-schema relationships from each schema separately. Then, the relationships existing in the WordNet database between the annotated meanings are exploited for generating relationships (lexicon-derived relationships) between the respective elements. The Integration Designer may add new relationships to capture specific domain knowledge. Finally, by means of a Description Logics reasoner, ODB-Tools [13], which performs equivalence and subsumption computation), new relationships of the Common Thesaurus are inferred.
4. *GVV generation:* Starting from the Common Thesaurus and the local sources schemata, MOMIS generates a GVV consisting of a set of global classes, plus mappings to connect the global attributes of each global class and the local sources' attributes. Going into details, the GVV generation is a process where ODL<sub>13</sub> classes describing the

same or semantically related concepts in different sources are identified and clustered in the same global class by means of the ARTEMIS tool [14]. ARTEMIS determines the degree of matching of two classes, based on their names and their structure, and produces an affinity tree. Clusters for integration are interactively selected from the affinity tree using a non-predefined threshold based mechanism. The Integration Designer may interactively refine and complete the proposed integration results; in particular, the mappings which have been automatically created by the system can be fine-tuned by means of the MDTFs, as will be discussed in next section. For each GVV involving a THALIA's query, MOMIS automatically detect the right basic relations and attributes mapping. For example, the GVV referred to Query 1 contains the mapping between the *Instructor* and *Lecturer* attributes, i.e. the THALIA foreseen challenge. For Query 12, the system recognizes the mapping between *CourseTitle* and *Title* of the two schemata, while the challenge, i.e. information contained in a unique attribute rather than separate attributes, is obtained by specifying a Data Transformation Function.

5. *GVV annotation*: The GVV is automatically annotated, i.e. each of its elements is associated to the broadest meanings extracted from the annotated sources. The annotation of a GVV is a significant result, since these metadata may be exploited for external users and applications interoperability. As an example, we report a fragment of annotated GVV (Query 12):

Global class / attribute	Local Classes	Meaning (from WordNet)
Course	Cmu.Course Brown.Course	Course#1
Instructor	Cmu.Lecturer Brown.Instructor	Instructor#1
Title	Cmu.CourseTitle Brown.Title	Title#1

Figure 2: GVV annotation

## 4. Mapping Refinement

During the GVV generation process, the system automatically generates a Mapping Table (MT) for each global class C of the GVV, whose columns represent the local classes L belonging to C and whose rows represent the global attributes of C. An element MT [GA][L] represents the set of local attributes of L which are mapped onto the global attribute GA.

After this automatic step, the Integration Designer may refine the MT by adding:

- Mapping Data Transformation Functions applied to local attributes
- Join Conditions between pairs of local classes belonging to C
- Resolution Functions for global attributes to solve data conflicts of local attribute values [15].

By exploiting the enriched MT the system automatically generates the *mapping query* associated to the global class C.

### 4.1 Mapping Data Transformation Functions

The Integration Designer may define, or refine, for each element MT[GA][L], a *Mapping Data Transformation Function*, denoted by MDTF[GA][L], which represents the mapping of local attributes of L into the global attribute GA. MDTF[GA][L] is a function that has to be executable/supported at the local source by the local source wrapper. In fact, we want to push as much as possible function execution at the sources, as suggested in [16] for constraint mappings.

MDTFs are obtained by combining SQL-92 functions, classic string manipulation functions available in MOMIS.

The following SQL-92 like functions are accepted:

CHAR\_LENGTH: returns the length of a string

POSITION: searches a pattern in a string

SUBSTRING: returns a part of a string

CAST: converts a value from a type to another

CASE ... WHEN ... THEN: transforms a record on the basis of a specific data value

In addition, the classic RIGHT and LEFT string functions, i.e. the first (or the last) n characters of a string, are available in the system.

The above functions are executed at the wrapper level by the right translation of the particular SQL-dialect for relational database systems and are built-in for other wrapper (like XML).

We added a specific function for datetime type conversion:

TIME12-24: transforms a string to a time value expressed in 12 or 24 hours format.

**Example Query 4:** a complex transformation function is required to convert the string that describes the expected scope of the course in *ethz* into a credit unit. At the ETH's Computer Science site is present the conversion formula:

$$\#KE = \#V + \#U + 1$$

that the designer inserts in the mapping table by specifying the following MDTF:

```
MDTF[Unit][ethz.Unterricht] =
CAST(SUBSTRING(Umfang, POSITION('V' IN Umfang) - 1, 1)
AS int)
+ CAST(SUBSTRING(Umfang, POSITION('U' IN Umfang) - 1,
1) AS int) + 1
```

**Example Query 7:** the following MDTF “infer s” that the course has a prerequisite course by extracting the text following the ‘Prerequisite’ term.

```
MDTF[prerequisite][asu.Course] =
CASE POSITION('%Prerequisite%' IN Description)
WHEN 0 THEN 'None'
ELSE RIGHT(Description,
CHAR_LENGTH(Description) -
POSITION('%Prerequisite%' IN Description) + 1)
END
```

**Example Query 12:** the challenge is to extract the correct title, day and time values from the title column in the catalog of the

Brown University that contains all the above information in a string. By using a combination of SUBSTRING and POSITION functions it is possible to obtain what requested.

```
MDTF[Title][brown.Course] =
SUBSTRING(Title FROM POSITION('/' IN Title) + 3 FOR
    POSITION('hr.' IN SUBSTRING(Title FROM
        POSITION('/' IN Title) + 3 FOR 100)) - 1)
MDTF[Day][brown.Course] =
SUBSTRING(Title FROM POSITION('hr.' IN Title) + 4 FOR
    POSITION(' ' IN SUBSTRING(Title
        FROM POSITION('hr.' IN Title) + 4
            FOR 10)))
MDTF[Time][brown.Course] =
SUBSTRING(Title IN POSITION(' ' FROM SUBSTRING(Title
    FROM POSITION('hr.' IN Title) + 4 FOR 10)) +
    POSITION('hr.' IN Title) + 4 FOR 15)
```

The transformation of a local class L obtained by applying all the Mapping Data Transformation Functions MDTF[GA][L] is denoted with T(L).

## 4.2 Join Conditions

Merging data from different sources requires different instantiations of the same real world object to be identified; this process is called object identification [17]. The topic of object identification is currently a very active research area with significant contributions both from the artificial intelligence [18] and database communities [19,20]. We assume, for the sake of simplicity, that the integration designer be able to define join conditions among local classes.

The Join Conditions among pairs of local classes belonging to the same global class are introduced in order to identify instances of the same object and fuse them. Given two local classes L1 and L2 belonging to C, a Join Condition between L1 and L2, denoted with JC(L1,L2), is a Boolean expression of atomic constraints (L1.Ai Op L2.Aj) where Ai (Aj) are global attributes with a not null mapping in L1 (L2) and Op is a relational operator.

As an example, for Query 5, the designer should define the following join condition:

$$JC(L1, L2) : L1.CourseName = L2. Titel$$

where L1= cmu.Course and L2= ethz.Unterricht.

## 4.3 Resolution Functions

The fusion of data coming from different sources and taking into account the problem of inconsistent information among sources is a hot research topic [17,21,22,23,24]. In MOMIS the approach proposed in [17] has been adopted: a *Resolution Function* for solving data conflicts may be defined for each global attribute mapping onto local attributes coming from more than one local source.

A global attribute with no data conflicts (i.e. the instances of the same real object in different local classes having the same value for this common attribute), is called Homogeneous Attribute. Of course, for homogeneous attributes, resolution functions are not necessary (a global attribute mapped onto only one source is a particular case of an homogeneous attribute).

As an example, in Query 1, we may define a *precedence function* for Course Title:

gatech.Course.Title has a higher precedence than cmu.Course.CourseTitle.

## 4.4 The Mapping Query

MOMIS follows a GAV approach, thus for each global class C, a *mapping query* QC over the schemas of a set of local classes L must be defined. MOMIS automatically generates the mapping query QC associated to C, by extending the Full Disjunction (FD) operator [25,26]. In our context: given a global class C composed of L1,L2, ..., Ln, we consider

FD(T(L1),T (L2),...,T (Ln)), computed on the basis of the Join Conditions.

With two classes, FD corresponds to the full (outer) join:

$$FD(T (L1),T (L2)) = T (L1) \text{ full join } T (L2) \text{ on } (JC(L1,L2)).$$

For the complete definition and computation of FD see [23].

Finally, QC is obtained by applying Resolution Functions to the attributes resulting from FD: for a global attribute GA we apply the related Resolution Function to T (L1).GA, T (L2).GA, . . . , T (Lk).GA [15].

## 5. Query Rewriting with MDTFs

To answer a query expressed on the GVV (*global query*) the query must be rewritten as an equivalent set of queries expressed on the local schemata (*local queries*); this query translation is performed by considering the mappings among the GVV and the local schemata. In a GAV approach query translation is performed by means of *query unfolding*, i.e., by expanding a global query on a global class C of the GVV according to the definition of the *mapping query* QC as defined in section 4.4. In this section we present the MOMIS query unfolding method including MDTFs.

### 5.1 Query Unfolding

The query unfolding process is performed for a *Global Query Q* over a global class C of the GVV :

$$Q = SELECT \langle Q\_SELECT-list \rangle \text{ from } C \text{ where } \langle Q\_condition \rangle$$

where  $\langle Q\_condition \rangle$  is a Boolean expression of positive atomic constraints: (GA1 op value) or (GA1 op GA2), with GA1 and GA2 attributes of C.

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

**Step 1) Generation of Local Queries:**

$$LQ = SELECT \langle SELECT-list \rangle \\ FROM L \\ WHERE \langle condition \rangle$$

where L is a local class related to C.

The  $\langle SELECT-list \rangle$  is computed by considering the union of:

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

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 basis of the *MDTFs* and *Resolution Functions* defined in the Mapping Table. For example, if the numerical global attribute *GA* is mapped onto *L1* and *L2*, and we define *AVG* as resolution function, the constraint (*GA = value*) cannot be pushed at the local sources, because *AVG* has to be calculated at a global level. In this case, the constraint is mapped as true in both the local sources. On the other hand, if *GA* is an homogeneous attribute the constraint can be pushed at the local sources. For example, an atomic constraint (*GA op value*) is mapped onto the local class *L* as follows:

(*MDTF [GA][L] op value*) if *MT [GA][L]* is not null and  
the *op* operator is supported into *L*

true                      otherwise

An atomic constraint (*GA1 op GA2*) is mapped in a similar way.  
**Step 2)** Generation of *FD(LQ1, LQ2, ... , LQn)* which computes the Full Disjunction of the *LQs*

**Step 3)** Generation of the final query (application of *Resolution Functions*):

- for Homogeneous Attributes we can take one of the value;
- for non Homogeneous Attributes (e.g. *Address*) we apply the associated Resolution Function (in this case the precedence function).

## 5.2 Multilingual Query Condition

In an information integration scenario, data are frequently expressed in different languages, for example a source contains english data and another german or italian data.

MOMIS supports multilingual query conditions expressed by means of a *multilingual constraint* in the form

*GA op TRANSLATE(term, Language)*

where *GA* is a global attribute and *term* is a word of a given *Language*. The condition of a Global Query is then a Boolean expression of atomic and multilingual constraints.

The rewriting of a multilingual constraint is performed by a *TRANSLATION* function that translate from the language specified in *TRANSLATE(term, Language)* into the different languages of the local sources. The translation is obtained by exploiting the open dictionary of the Gutenberg Project ([www.gutenberg.org](http://www.gutenberg.org)) and determining for each term, the translation in the language of the local source, that is a metadata associated to each source during the integration phase.

More precisely, the rewriting of a multilingual constraint works as follows. Given two languages, *Language1* and *Language2*, and a term of *Language1*, we consider a function *TRANSLATION(term, Language1, Language2)* whose result is a set of terms of *Language2*: {*term\_1, ..., term\_n* }, with  $n \geq 1$ . To perform the constraint mapping (see Step 1 above) of a multilingual constraint

*AG LIKE TRANSLATE(term, Language)*

w.r.t. a local class *L* we consider a preliminary step where the multilingual constraint is transformed into a disjunction of atomic constraints:

*GA LIKE term\_1 OR .... OR GA LIKE term\_n*

where *term\_i*,  $1 \leq i \leq n$ , is a term obtained by *TRANSLATION(term, Language, Language2)*, and *Language2* is

the language of the local class *L*. Then the disjunction of atomic constraints is mapped into the local class *L* attributes as discussed before.

**Example Query 5:** the challenge is related to the expression of the attribute values in different languages. For example, in the target schema the course name is expressed in english and in the challenge schema in german.

The global query submitted to the MOMIS Query Manager is the following:

```
SELECT Name
FROM Course
WHERE Name LIKE TRANSLATE('%Database%',
'en')
```

For the target schema (umd), which is in English, no translation is required, so the local query is:

```
SELECT CourseName
FROM Course
WHERE CourseName LIKE '%Database%'
```

For the challenge schema (ethz) where the language is german, the global query is translated in the following local query:

```
SELECT Titel
FROM Unterricht
WHERE Titel LIKE '%Datenbank%'
OR Titel like '%Datei%'
OR Titel like '%Datenbasis%'
```

A complete example of Query rewriting will be shown in section 6.2.

## 6. THALIA Benchmark: Experimental Results

This section describes the results of MOMIS applied to the THALIA benchmark. MOMIS is fully written in Java and is available on-line via Java Web Start (<http://www.dbgrou.unimo.it/momisjws>).

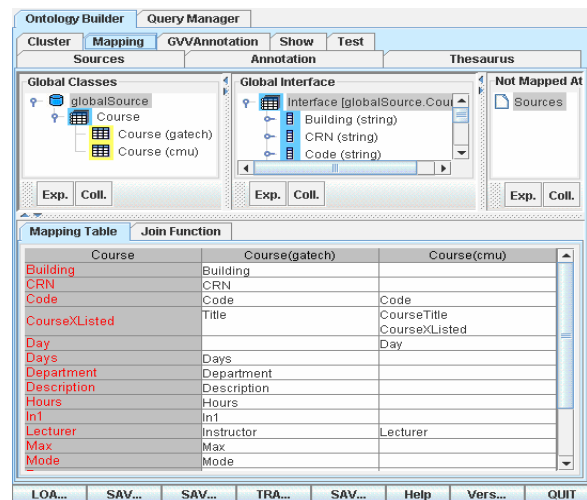


Figure 3 – MOMIS Schema Mapping example

The THALIA benchmark provides the twelve queries in XML format, while MOMIS provides an SQL-like syntax as a query language; we transformed the queries in *SPJ query* by a straightforward operation, with no effect on the benchmark result. In addition, our XML-Schema wrapper generates a relational view of a schema and automatically load the xml data file into a relation database, thus each data set provided by THALIA is loaded by the wrapper in a specific database.

## 6.1 Integration Phase

During this phase, for each pair of target and challenge schema related to a single query, the designer semi-automatically creates a specific GVV, i.e. twelve GVV's has been deployed for the benchmark.

Since the reference schemas are very simple, the GVV's creation was simple and completely automatic, while the designer had to carefully work on mapping refinements. As an example, we describe the GVV creation related to Query 1.

The reference schemas are of Georgia Tech University and of Carnegie Mellon University.

Georgia Tech University schema contains only Course class with many attributes such as Title, Section, Instructor, Room, Description, ...

Also Carnegie Mellon University contains only Course class with attributes like CourseTitle, Room, Lecturer, Time, Unit, ...

The automatic Global Class obtained is shown in Figure 2, where it is possible to note that the Lecturer and Instructor attributes of the sources are mapped into a single global attribute.

During the mapping refinement phase, for each query, a specific composition of mapping functions has been inserted to overcome the challenge. Appendix A reports the mapping refinement for each query, while the following table summarize the MDTF functions used for each query challenge:

Attribute Heterogeneities	
Query 1	Only Attributes mapping
Query 2	TIME12-24, SUBSTRING
Query 3	SUBSTRING, POSITION
Query 4	CAST, SUBSTRING, POSITION
Query 5	TRANSLATE
Missing data	
Query 6	Attributes mapping, NULL treatment
Query 7	CASE WHEN ... THEN, CHAR_LENGTH, RIGHT, POSITION
Query 8	Attributes mapping, NULL treatment
Structural Heterogeneities	
Query 9	SUBSTRING, POSITION
Query 10	SUBSTRING, POSITION
Query 11	CASE WHEN ... THEN, CHAR_LENGTH
Query 12	SUBSTRING, POSITION

## 6.2 Query Phase

During this phase, the writes the queries over the GVV's (see appendix A). MOMIS provides a command line interface for queries and a grid interface for the data answer.

To show a complete example of query processing we consider the following query:

**Example Query 4:** the benchmark query 'List all database courses that carry more than 10 credit hours' is formulated as follows:

```
Select Title, Units
from Course
where Title LIKE TRANSLATE('%Database%', 'en')
and Units > 10
```

The (portion of) the Mapping Table of the class Course involved in the query is the following:

Course	ethz.Unterricht	Cmu.Course
Title	Titel	CourseTitle
Units	MDTF[Unit][ethz.Unterricht]	Units

This global query is automatically rewritten for the target schema (cmu) and for the challenge schema (ethz) as:

Local Query 1 - Source "cmu" (LQ1)  
 SELECT Course.CourseTitle, Course.Units  
 FROM Course  
 WHERE (CourseTitle) like ('%Database%')  
 AND Units > 10

Local Query 2 - Source "ethz" (LQ2)  
 SELECT Unterricht.Titel,  
 (CAST(SUBSTRING(Umfang, CHARINDEX('V', Umfang) - 1, 1) AS int) + CAST(SUBSTRING(Umfang, CHARINDEX('U', Umfang) - 1, 1) AS int) + 1) AS Umfang  
 FROM Unterricht  
 WHERE ((Titel) like ('%Datenbank%') or (Titel) like ('%Datei%') or (Titel) like ('%Datenbasis%')) AND  
 (CAST(SUBSTRING(Umfang, CHARINDEX('V', Umfang) - 1, 1) AS int) + CAST(SUBSTRING(Umfang, CHARINDEX('U', Umfang) - 1, 1) AS int) + 1) > 10

The above local queries are executed at the local sources and then the Full Disjunction of their results is computed as follows (see [23] for details):

Full disjunction computation (FD)  
 Select LQ1.CourseTitle AS Title\_1, LQ2.Titel AS Title\_2,  
 LQ1.Units as Units\_1, LQ2.Umfang AS Units\_2  
 from LQ1 full outer join LQ2 on LQ1.CourseTitle = LQ2.Titel

The result of the global query is obtained by applying Resolution Functions to the above FD query:

```
Select resolution(Title_1, Title_2) AS Title,
resolution(Units_1, Units_2) as Units
from FD
```

The records obtained after the query execution are shown in a grid, where, for each attribute of a single record, the user can visualize the local source that has provided the data.

### 6.3 Experimental Comparison

Three different integration systems have reported the THALIA benchmark results: Cohera, Integration Wizard (IWIZ) [4] and a ‘Keyword Join’ system [27].

In the following table we compare the results, with the specification of extra effort for query answer.

Query	Cohera	Integration Wizard	‘Keyword join’
Query 1	Yes	Yes, SMALL	Yes
Query 2	Yes, SMALL	Yes, SMALL	NO
Query 3	Yes, MODERATE	Yes, MODERATE	Yes
Query 4	NO	NO	Yes, difficult
Query 5	NO	NO	Yes, difficult
Query 6	Yes	Yes, MODERATE	Yes
Query 7	Yes, MODERATE	Yes, MODERATE	NO
Query 8	NO	NO	NO
Query 9	Yes	Yes, SMALL	Yes, need semantic metadata
Query 10	Yes	Yes, SMALL	Yes, need semantic metadata
Query 11	Yes, MODERATE	Yes, MODERATE	Yes
Query 12	Yes, MODERATE	Yes, MODERATE	Yes

SMALL: small amount of code

MODERATE: moderate amount of code

Summarizing, Cohera and IWIZ can solve 9 queries, some of these by adding a significant amount of code, while the system presented in [27] could deal with 5 queries easily, and another 2 queries with a small amount of metadata, without any custom code.

In MOMIS, by means of declarative functions, it is very easy to deal with all the 12 queries: we estimate that an integration designer might define the requested customization (reported in Appendix A) within 4-6 hours.

## 7. Related Work

In the area of heterogeneous information integration, many projects based on mediator architectures have been developed. The mediator-based TSIMMIS project [28] follows a “structural” approach and uses a self-describing model (OEM) to represent heterogeneous data sources and the MSL (Mediator Specification Language) rule to enforce source integration. In TSIMMIS, by means of MSL, arbitrary views (in particular, recursive views) can be defined at the mediator layer. The MOMIS system made a different choice: starting from the semi-automatic generated mappings between global and local attributes stored in the

mapping tables, views (global classes) are defined by means of a predefined operator, i.e. the full disjunction, that has been recognized as providing a natural semantics for data merging queries. In particular, in the view definition resolution functions are defined to take into account data conflicts.

SIMS [29] proposes the creation of a global schema by exploiting the use of Description Logics (i.e., the LOOM language) for the description of information sources. The use of a global schema allows both GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them.

The Information Manifold system [30] provides a source independent and query independent mediator. The input schema of Information Manifold is a set of descriptions of the sources and the integrated schema is mainly defined manually by the designer, while in our approach it is tool-supported.

The goal of Clío [31] is to develop a tool for semi-automatically creating mappings between two data representations (i.e., with user input). First of all, in the Clío framework the focus is on the schema mapping problem in which a source is mapped onto a different, but fixed, “target” schema, while the focus of our proposal is the semi-automatic generation of a “target” schema, i.e. the Global Virtual View, starting from the sources. Moreover, the semi-automatic tool for creating schema mappings, developed in Clío, employs a mapping-by-example paradigm that relies on the use of value mappings describing how a value of a target attribute can be created from a set of values of source attributes. Our proposal for creating schema mappings can be considered orthogonal with respect to this paradigm. In fact, the main techniques of mapping construction rely on the meanings of the class and attribute names selected by the designer in the annotation phase and by considering the semantic relationships between meanings coming from the common lexical ontology. On the other hand, MOMIS and CLIO share a common mapping semantics among a (target) global schema and a set of source schemata expressed by the full-disjunction operator.

Infomaster [32] provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized, homogeneous information system. The main difference of this project w.r.t. our approach is the lack of a tool aid-support for the designer in the integration process.

## 8. Conclusion

In this paper we presented MOMIS, extended with MDTFs and proved it satisfies all the challenges provided by the THALIA benchmark. Future work will be devoted to enhance the TRANSLATION function by exploiting altavista babelfish translator (<http://babelfish.altavista.com>), by means of the API provided by Jonathan Feinberg (<http://babel.mrfeinberg.com>).

## 9. REFERENCES

- [1] Wiederhold, G. Intelligent integration of information. In ACM SIGMOD Conference (1993) (pp. 434–437).
- [2] Doan A., Halevy A. Y. Semantic Integration Research in the Database Community: A Brief Survey. *AI Magazine* 26(1): 83-94 (2005).
- [3] Abiteboul S., Agrawal R., Bernstein P. A., Carey M. J., Ceri S., Croft W. B., DeWitt D. J., Franklin M. J., Garcia-Molina H., Gawlick D., Gray J., Haas L. M., Halevy A. Y.,

- Hellerstein J. M., Ioannidis Y. E., Kersten M. L., Pazzani M. J., Lesk M., Maier D., Naughton J. F., Schek H., Sellis T. K., Silberschatz A., Stonebraker M., Snodgrass R. T., Ullman J. D., Weikum G., Widom J., Zdonik S. B. The Lowell database research self-assessment. *Commun. ACM* 48(5): 111-118 (2005).
- [4] Hammer J., Stonebraker M., Topsakal O. THALIA: Test Harness for the Assessment of Legacy Information Integration Approaches. *ICDE 2005*: 485-486.
- [5] Beneventano D., Bergamaschi S., Guerra F., Vincini M. Synthesizing an Integrated Ontology. *IEEE Internet Computing* 7(5): 42-51 (2003).
- [6] Beneventano D., Bergamaschi S., Castano S., Vincini M. Semantic Integration of Heterogeneous Information Sources. Special Issue on Intelligent Information Integration, *Data & Knowledge Engineering*, Vol. 36, Num. 1, Pages 215-249, Elsevier Science B.V. 2001.
- [7] Bergamaschi S., Castano S., Vincini M. Semantic Integration of Semistructured and Structured Data Sources. *SIGMOD Record Special Issue on Semantic Interoperability in Global Information*, Vol. 28, No. 1, March 1999.
- [8] Halevy, A. Y. Answering queries using views: A survey. *VLDB Journal*, 10(4), 270-294, 2001.
- [9] Ullman, J. D. Information integration using logical views. In *ICDT Conference*, (pp. 19-40), 1997.
- [10] Maurizio Lenzerini: Data Integration: A Theoretical Perspective. *PODS 2002*: 233-246.
- [11] Abiteboul S., Buneman P., Suciu D. Data on the Web: From relations to semistructured data and XML. *Data Management Systems*. Morgan Kaufmann (2000).
- [12] Du F., Amer-Yahia S., Freire J. ShreX: Managing XML Documents in Relational Databases. *VLDB 2004*: 1297-1300, 2004.
- [13] Beneventano, D., Bergamaschi, S., Lodi, S., Sartori, C. Consistency Checking in Complex Object Database Schemata with Integrity Constraints. *IEEE Trans. Knowledge and Data Eng.*, 10(4), (pp. 576-598) 1998.
- [14] Castano S., De Antonellis V., De Capitani di Vimercati S. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2) 2001.
- [15] Beneventano D., Bergamaschi S. Semantic search engines based on data integration systems. In *Semantic Web Services: Theory, Tools and Applications*. Idea Group Publishing, 2006.
- [16] Chang K., Garcia-Molina H. Mind Your Vocabulary: Query Mapping Across Heterogeneous Information Sources. *SIGMOD Conference 1999*: 335-346.
- [17] Naumann, F., Haussler, M. Declarative data merging with conflict resolution. In *MIT-IQ Conference* (pp. 212-224) 2002.
- [18] Tejada, S., Knoblock, C. A., Minton, S. Learning object identification rules for information integration. *Inf. Syst.*, 26 (8), 607-633, 2001.
- [19] Ananthakrishna, R., Chaudhuri, S., Ganti, V. Eliminating fuzzy duplicates in data warehouses. In *VLDB Conference*, (pp. 586-597) 2002.
- [20] Chaudhuri, S., Ganjam, K., Ganti, V., Motwani, R. Robust and efficient fuzzy match for online data cleaning. In *ACM SIGMOD Conference* (pp. 313-324) 2003.
- [21] De Giacomo, G. D., Lembo, D., Lenzerini, M., Rosati, R. Tackling inconsistencies in data integration through source preferences. In *ACM IQIS Workshop* (pp. 27-34) 2004.
- [22] Bertossi, L. E., Chomicki, J. Query answering in inconsistent databases. In J. Chomicki, R. van der Meyden, & G. Saake (Eds.), *Logics for Emerging Applications of Databases* (pp. 43-83) 2003. Springer.
- [23] Greco, G., Greco, S., Zumpano, E. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng.*, 15 (6), 1389-1408 2003.
- [24] Lin, J., Mendelzon, A. O. Merging databases under constraints. *Int. J. Cooperative Inf. Syst.*, 7 (1), 55-76, 1998.
- [25] Galindo-Legaria, C. A. Outerjoins as disjunctions. In *ACM-SIGMOD Conference* (pp. 348-358), 1994.
- [26] Rajaraman, A., Ullman, J. D. Integrating information by outerjoins and full disjunctions. In *ACM-PODS Conference* (pp. 238-248) 1996.
- [27] Yu B., Liu L., Ooi B. C., Tan K. L. Keyword Join: Realizing Keyword Search for Information Integration, in MIT press, *Computer Science (CS)*, 2006, <http://hdl.handle.net/1721.1/30263>
- [28] Li C., Yerneni R., Vassalos V., Garcia-Molina H., Papanikolaou Y., Ullman J., Valiveti M. Capability Based Mediation in TSIMMIS. *SIGMOD 98*, Seattle, June 1998.
- [29] Knoblock C. A. Ambite J.L. Flexible and scalable cost-based query planning in mediators: A transformational approach. *Artificial Intelligence*, 118(1-2):115-161, 2000.
- [30] Levy A., The Information Manifold Approach to Data Integration, *IEEE Intelligent Systems*, 1312-16, 1998.
- [31] Miller R. J., Hernandez M. A., Haas L. M., Yan L., Ho C. T. H., Popa L., Fagin R., The Clio project: managing heterogeneity, *ACM SIGMOD Record* 30, 1 (March 2001), pp. 78-83.
- [32] Genesereth M. R., Keller A. M., Duschka O., Infomaster: An Information Integration System, in proceedings of 1997 *ACM SIGMOD Conference*, May 1997.

## 10. APPENDIX A Mapping Refinement

### Query 1

Mapping between Instructor attribute of Georgia Tech University and Lecturer attribute of Carnegie Mellon University.  
No mapping refinement.

### Query 2

Mapping between Time attribute of Carnegie Mellon University and Times attribute of University of Massachusetts.

Mapping refinement:

$MDTF[Time][umb.Course] = TIME12-24(Times, 1, 12) +$

SUBSTRING(Times, 6, 1) +  
TIME12-24(Times, 7, 12)

### Query 3

Mapping between CourseName attribute of University of Maryland and Title attribute of Brown University.

Mapping refinement:

```
MDTF[Title][brown.Course] = SUBSTRING(Title FROM
    POSITION('/' IN Title) + 3 FOR POSITION ('hr.' IN
    SUBSTRING (Title FROM POSITION ('/' IN Title)
    + 3 FOR 100)) - 1)
```

### Query 4

Mapping between Units attribute of Carnegie Mellon University and Umfang attribute of ETH Zurich.

Mapping refinement:

```
MDTF[Unit][ethz.Unterricht] =
CAST(SUBSTRING(Umfang, POSITION('V' IN Umfang) - 1, 1)
    AS int)
+ CAST(SUBSTRING(Umfang, POSITION('U' IN Umfang) - 1,
    1) AS int) + 1
```

### Query 5

Mapping between CourseName attribute of University of Maryland and Title attribute of ETH Zurich.

No mapping refinement.

### Query 6

Mapping between title attribute of University of Toronto and no attribute of Carnegie Mellon University.

### Query 7

Mapping between prerequisite attribute of University of Michigan and description attribute in Arizona State University.

Mapping refinement:

```
MDTF[prerequisite][asu.Course] =
CASE POSITION('%Prerequisite%' IN Description)
WHEN 0 THEN 'None'
    ELSE RIGHT(Description,
    CHAR_LENGTH(Description) -
    POSITION('%Prerequisite%' IN Description) + 1)
END
```

### Query 8

Mapping between 'Course restricted' attribute of Georgia Tech University and no attribute of ETH Zurich.

No mapping refinement.

### Query 9

Mapping between room attribute of Brown University and time attribute of University of Maryland.

Mapping refinement:

MDTF[Room][umd.section] = SUBSTRING(Time FROM  
POSITION('%%' IN Time) FOR 30)

### Query 10

Mapping between lecturer attribute of Carnegie Mellon University and title attribute of University of Maryland.

Mapping refinement:

```
MDTF[Title][umd.section] = SUBSTRING(Title FROM
    POSITION('%%' IN Title) FOR POSITION('%%' IN
    Title) + 2) FOR POSITION('%%' IN Title) + 1)
```

### Query 11

Mapping between lecturer attribute of Carnegie Mellon University and attributes named Fall2003, Winter2004 and Spring2004 of University of California, San Diego.

Mapping refinement:

```
MDTF[Lecturer][ucsd.Course] =
CASE WHEN (CHAR_LENGTH (Fall2003) >
CHAR_LENGTH (Winter2004) AND
CHAR_LENGTH (Fall2003) > CHAR_LENGTH (Spring2004))
    THEN Fall2003
WHEN (CHAR_LENGTH (Winter2004) >
CHAR_LENGTH (Fall2003) AND CHAR_LENGTH
(Winter2004) > CHAR_LENGTH (Spring2004))
    THEN Winter2004
WHEN (CHAR_LENGTH (Spring2004) >
CHAR_LENGTH (Fall2003) AND
CHAR_LENGTH (Spring2004) >
CHAR_LENGTH (Winter2004)) THEN Spring2004
END
```

### Query 12

Mapping between CourseTitle, Day, Time attribute of Carnegie Mellon University and Title attribute of Brown University.

Mapping refinement:

```
MDTF[Title][brown.Course] =
SUBSTRING(Title FROM POSITION('/' IN Title) + 3 FOR
    POSITION ('hr.' IN SUBSTRING(Title FROM
    POSITION ('/' IN Title) + 3 FOR 100)) - 1)
MDTF[Day][brown.Course] =
SUBSTRING(Title FROM POSITION('hr.' IN Title) + 4 FOR
    POSITION(' ' IN SUBSTRING(Title
    FROM POSITION('hr.' IN Title) + 4
    FOR10)))
MDTF[Time][brown.Course] =
SUBSTRING(Title IN POSITION(' ' FROM SUBSTRING(Title
    FROM POSITION('hr.' IN Title) + 4 FOR 10)) +
    POSITION('hr.' IN Title) + 4 FOR 15)
```