Semantic Integration and Query Optimization of Heterogeneous Data Sources*
(Invited Paper)

D. Beneventano[1], S. Bergamaschi[1], S. Castano[2], V. De Antonellis[3], A. Ferrara[2]
F. Guerra[1], F. Mandreoli[1], G. Ornetti[2], and M. Vincini[1]

[1] Università di Modena e Reggio Emilia
{beneventano.domenico,bergamaschi.sonia,guerra.francesco,
mandreoli.federica, vincini.maurizio}@unimo.it
[2] Università di Milano
{castano,ferrara,ornetti}@dsi.unimi.it
[3] Università di Brescia
deantone@ing.unibs.it

Abstract. In modern Internet/Intranet-based architectures, an increasing number of applications requires an integrated and uniform access to a multitude of heterogeneous and distributed data sources. In this paper, we describe the ARTEMIS/MOMIS system for the semantic integration and query optimization of heterogeneous structured and semistructured data sources.

1 Introduction

In modern Internet/Intranet-based architectures, an increasing number of applications requires an integrated and uniform access to a multitude of heterogeneous and distributed data sources. A methodological framework for the integration of highly heterogeneous data sources requires guidance and support tools for dealing with different types of heterogeneity that can occur among the sources, and should provide intelligent techniques and tools for automating as much as possible various activities involved in the semantic integration process, such as for example the acquisition of interschema properties, the schema matching and reconciliation activities, or the generation of semantic mappings between the global schema and the local schemas of the underlying data sources for query purposes. A first type of heterogeneity to be considered when integrating heterogeneous sources derives from the level of structuring of data to be integrated. Several data models and languages can be adopted for data representation and storage, ranging from non structured data, typical of file systems, to highly structured data, typical of database systems, to semistructured data, typical of Web data sources [8]. Another type of heterogeneity to be considered derives from the terminology, structure, and context characterizing data stored at each source,

* This paper has been partially funded by MIUR COFIN2000 D2I project.

which requires to take into account data semantics and to define semantic mappings between data. These problems have been first addressed in the context of database integration [9], and more recently in the context of Web and XML data integration [10, 5, 6].

In this paper, we describe the ARTEMIS/MOMIS system for the semantic integration and query optimization of heterogeneous structured and semistructured data sources [2, 3]. ARTEMIS/MOMIS supports a virtual approach to integration, in that data residing at the sources are accessed during query processing by exploiting defined mappings, instead of being replicated at the global level. The global schema, associated mapping descriptions and interschema knowledge obtained from the integration process provide support for expressing queries in terms of the global schema over underlying data sources, with mechanisms for the reformulation and answering of such queries in terms of the data stored in the sources. Such a problem is known in the literature as view-based query processing, and has been studied very actively in the recent years[7, 11, 12].

The paper is organized as follows. Section 2 describes the interschema knowledge extraction and representation process. Section 3 is devoted to the source schema matching process. Section 4 describes the global schema and mapping generation process. Section 5 illustrates the semantic query otimization process. Finally, in Section 6, we give our concluding remarks.

2 Interschema knowledge extraction and representation

The semantic integration of strongly heterogeneous data sources is performed by constructing a semantically rich representation of the data sources to be integrated, by means of a common data model based on the ODL_{I^3} language. In ARTEMIS/MOMIS, this task is performed by wrapper tools developed for both structured and semistructured data sources [2]. The subsequent phase of the integration process is the extraction and representation of the interschema properties featuring the sources to be integrated. In ARTEMIS/MOMIS, interschema knowledge is expressed through *intensional properties* and *extensional properties*. Intensional properties are *terminological relationships* expressing inter-schema knowledge at the intensional level for the source schemas. Intensional relationships are defined between classes and attributes, and are specified by considering class/attribute names, called terms. The following relationships can be specified in ODL_{I^3}: SYN (Synonym-of), BT/NT (Broader/Narrower Terms), or hypernymy/hyponymy, and RT (Related Terms), or positive association [2, 3].

An intensional relationships is only a terminological relationship, with no implications on the extension/compatibility of the structure (domain) of the two involved classes (attributes). For capturing this kind of knowledge, extensional properties are used. Extensional properties express interschema knowledge at the extensional level. The intensional relationships SYN, BT, NT and RT between two classes C_1 and C_2 may be "strengthened" by establishing that they are also *extensional* relationships [4]. Consequently, the following extensional relationships can be defined in ODL_{I^3}:

UNIVERSITY source (UNI)

```
Research_Staff(name,e_mail,dept_code,s_code)
School_Member(name,school,year,e_mail)
Department(dept_name,dept_code,budget)
Section(section_name,s_code,length,room_code)
Room(room_code,seats_number,notes)
```

COMPUTER_SCIENCE source (CS)

```
CS_Person(first_name,last_name)
Professor:CS_Person(belongs_to:Division,rank)
Student:CS_Person(year,takes:set⟨Course⟩,rank,e_mail)
Division(description,address:Location)
Location(city,street,number,country)
Course(course_name,taught_by:Professor)
```

TAX_POSITION_XML source (TP)

```
<!ELEMENT ListOfStudent (Student*)>
<!ELEMENT Student (name, s_code, school_name, e_mail, tax_fee)>
<!ELEMENT name (#PCDATA)>
...
```

Fig. 1. Three heterogeneous University sources

- C_1 SYN_{ext} C_2: the instances of C_1 and C_2 are the same.
- C_1 BT_{ext} C_2: the instances of C_1 are a superset of the instances of C_2.
- C_1 NT_{ext} C_2: the instances of C_1 are a subset of the instances of C_2.
- C_1 RT_{ext} C_2: the instances of C_1 overlap the instances of C_2.
- C_1 DISJ_{ext} C_2: the instances of C_1 are disjoint from the instances of C_2.

Intensional and extensional properties can be partially automatically extracted and partially explicitly declared by the integration designer. A *Common Thesaurus* of terminological and extensional relationships is constructed, describing interschema knowledge about ODL_{I^3} classes and attributes of source schemas.

The Common Thesaurus is built through an incremental process during which relationships are added in the following order: *i) schema-derived relationships*: intensional and extensional relationships holding at intraschema level are extracted by analyzing each ODL_{I^3} schema separately; *ii) lexical-derived relationships*: intensional relationships holding at interschema level are extracted by analyzing different sources ODL_{I^3} schemas together according to the Wordnet supplied ontology; *iii) designer-supplied relationships*: additional intensional and extensional relationships are supplied directly by the designer, to capture domain knowledge about the source schemas. Supplied relationships are validated with ODB-Tools [1]; *iv) inferred relationships*: a new set of relationships (i.e. new generalization and aggregation properties) is inferred by ODB-Tools by reasoning over the union of the local schemas enriched with currently available interscheme properties. As a running example, we consider three heterogeneous

sources (see Figure 1): the first source `University` (UNI) is a relational database; the second source `Computer_Science` (CS) is an object–oriented database; the third source, `Tax_Position` (TP), is an XML file. Possible extensional relationships for the sources of the running example are the following:

1. `UNI.School_Member` SYN_{Ext} `TP.Student`
2. `CS.Student` NT_{Ext}`UNI.School_Member`
3. `CS.Professor` NT_{Ext} `UNI.Research_Staff`
4. `CS.Professor` DISJ_{Ext} `UNI.School_Member`
5. `UNI.Research_Staff` DISJ_{Ext} `TP.Student`
6. `UNI.Research_Staff` DISJ_{Ext} `CS.Student`
7. `CS.Student` NT_{Ext} `CS.CS_Person`
8. `CS.Professor` NT_{Ext} `CS.CS_Person`
9. `UNI.Section` RT_{ext}`CS.Division`
10. `CS.Student` NT_{ext}`TP.Student`

Relationships from 1 to 4 and 9 are supplied by the designer; relationships 5 and 6 are intra–schema extensional relationships automatically extracted by the system from the is–a hierarchies of the `COMPUTER_SCIENCE` source; relationship 10 is inferred from 1 and 2.

3 Source schema matching

Goal of this phase of the integration process in ARTEMIS/MOMIS is to identify ODL_{I^3} classes candidate to integration, that is, classes that describe the same or semantically related information in different source schemas. To this end, *affinity coefficients* are evaluated for all possible pairs of ODL_{I^3} classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two ODL_{I^3} classes c and c' based on their names (*Name Affinity* coefficient) and their attributes (*Structural Affinity* coefficient). A *Global Intensional Affinity* (GIA) coefficient is then determined as the linear combination of the Name and Structural Affinity coefficients. To evaluate such coefficients, an affinity function $A()$ is defined on top of the Common Thesaurus to evaluate the affinity of two terms. The affinity $A(t, t')$ of two terms t and t' is equal to the highest-strength path of terminological relationships (i.e., intensional) between them, if at least one path exist, and is zero otherwise. Given two terms t and t' and a path of terminological relationships between them, the strength of this path is computed by multiplying the strengths of all terminological relationships involved in it. $A(t, t')$ coincides with the strength of the highest-strength path between t and t', denoted by $\rightarrow^m$, that is, $A(t, t') = \sigma_{12_{\Re}} \cdot \sigma_{23_{\Re}} \cdot \ldots \cdot \sigma_{(m-1)m}$. To assess the level of affinity of two ODL_{I3} classes in a comprehensive way, a Global Affinity coefficient taking into account also the knowledge provided by extensional properties is introduced. We denote by $\Re_{ext}$, with $\Re_{ext} \in \{SYN_{ext}, BT_{ext}, NT_{ext}, RT_{ext}, DJS_{ext}\}$ an inter-schema extensional relationship. For affinity evaluation, each extensional relationship $\Re_{ext}$ is associated with a strength $\sigma_{\Re_{ext}} \in [0, 1]$, expressing its implication for affinity evaluation. Following what has been done in assessing the strengths of the

Table 1. Affinity coefficients for schema matching

Coefficient	Definition	Description
$NA(c, c')$	$A(n_c, n_{c'})$	$NA(c, c')$ is the value of the affinity between the names of two classes, if this value exceeds a specified threshold.
$SA(c, c')$	$\frac{2 \cdot \lvert\{(a_t, a_q) \lvert a_t \in A(c), a_q \in A(c'), n_t \sim n_q\}\rvert}{\lvert A(c)\rvert + \lvert A(c')\rvert}$	$SA(c, c')$ is the measure of the level of structural matching of the classes, proportional to the number of their attributes whose names have affinity and whose domains are compatible.
$GIA(c, c')$	$w_{NA} \cdot NA(c, c') + w_{SA} \cdot SA(c, c')$	$GIA(c, c')$ is a global measure of the affinity of two classes based on intensional properties only, computed as the linear combination of their name and structural affinity, respectively.
$GA(c, c')$	$GIA(c, c')$+(1-$GIA(c, c')$) $\cdot\ \sigma_{\Re_{ext}}(c, c')$	$GA(c, c')$ is a comprehensive measure of the affinity of two classes, which takes into account also the extensional properties holding between the two classes.

Legend: $A(c)$ and $A(c')$ are the sets of attributes of c and c', respectively; $\lvert X \rvert$ denotes the cardinality of set X; $\sim$ denotes name affinity

terminological relationships, we consider equivalence (SYN_{ext}) as the strongest indicator for affinity; moreover we consider inclusion (BT_{ext}/NT_{ext}) stronger than overlapping (RT_{ext}). Table 1 summarizes the affinity coefficients.

Global affinity coefficients are then used by a hierarchical clustering algorithm, to classify ODL_{I^3} classes. The output of the clustering procedure is an affinity tree, where ODL_{I^3} classes are the leaves and intermediate nodes have an associated affinity value, holding for the classes in the corresponding cluster. Clusters for integration (candidate clusters) are interactively selected from the affinity tree using a threshold based mechanism.

Candidate clusters are characterized with respect to *size* and *level of homogeneity*. The size of a candidate cluster is the number of ODL_{I3} classes contained. The level of homogeneity of a candidate cluster is given by the threshold value, and consequently by the affinity value associated with the cluster in the affinity tree (i.e., a high value of affinity corresponds to a high level of homogeneity). The number, size, and level of homogeneity of candidate clusters is determined by the designer who can set the most suitable value for the threshold γ.

In Figure 2, we show the affinity tree computed by considering both intensional and extensional relationships, using a threshold value $\gamma = 0.5$.

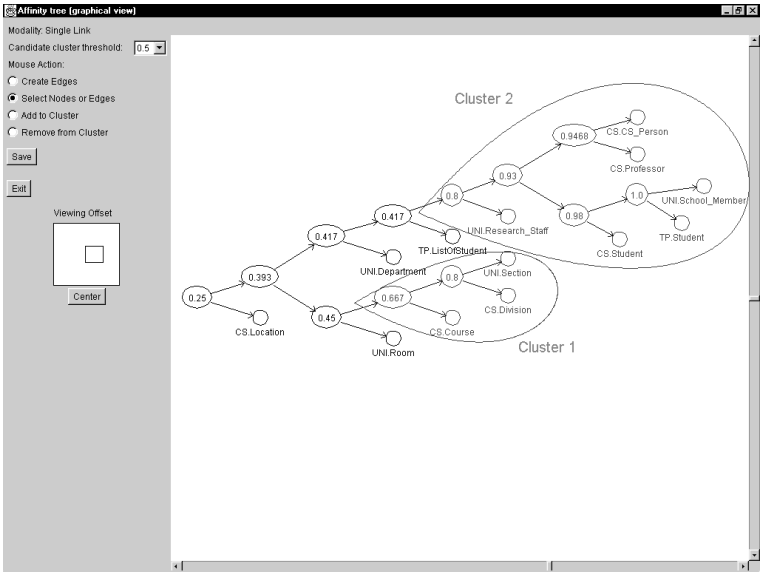

Fig. 2. Candidate clusters with extensional properties

4 Global schema and mapping generation

For each selected cluster in the tree, a global ODL_{I^3} class providing the unified view of all the classes of the cluster is defined in ARTEMIS/MOMIS. The generation of global classes is interactive with the designer. Let Cl_i be a selected cluster in the affinity tree and *global _class*$_i$ the global ODL_{I^3} class to be defined for Cl_i. First, we associate with *global _class*$_i$ a set of global attributes, corresponding to the union of the attributes of the classes belonging to Cl_i. The attribute unification process is performed automatically for what concerns names according to the following rules: i)for attributes that have a SYN relationship, only one term is selected as the name for the corresponding global attribute in *global _class*$_i$; ii)for attributes that have a BT/NT relationship, a name which is a broader term for all of them is selected and assigned to the corresponding global attribute in *global _class*$_i$.

To complete global class definition, information on attribute mappings and default values is provided by the designer in the form of *mapping rules*. For each global ODL_{I^3} class a persistent *mapping-table* storing all the mappings is generated; it is a table whose columns represent the set of the local classes belonging to the cluster and whose rows represent the global attributes. An element $MT[L][ag]$ represents the set of attributes of the local class L which are mapped into the global attribute ag: the value of the ag attribute is a function of the values assumed by the set of attributes $MT[L][ag]$. Some simple and frequent

University_Person	name	dept	e_mail	section	school	
UNI.Research_Staff	name	dept_code	e_mail	s_code	null	...
UNI.School_Member	name	null	e_mail	null	school	...
CS.CS_Person	first_name and last_name	null	null	null	"cs"	...
CS.Student	first_name and last_name	null	e_mail	null	"cs"	...
CS.Professor	first_name and last_name	null	null	null	"cs"	...
TP.Student	name	null	e_mail	null	school_name	...

	year	belong_to	takes	rank	s_code	tax_fee
...	null	null	null	"professor"	null	null
...	year	null	null	"student"	null	null
...	null	null	null	null	null	null
...	year	null	takes	rank	null	null
...	null	"belong_to"	null	rank	null	null
...	null	null	null	"student"	s_code	tax_fee

Fig. 3. Mapping Table for the global class `University_Person`

cases of such function are the following (see Figure 3 as an example):

- *identity* : the (ag) value *is equal to* the la value; we denote this case as $MT[L][ag] = la$.
- *concatenation* : the ag value is obtained as a concatenation of the values assumed by a set of local attributes la_i of the local class L; we denote this case as $MT[L][ag] = la_1$ `and` $\ldots$ `and` la_n (see MT[`CS.Student`][`name`]).

When the ag has no correspondence with any attribute of the local class L, the possible choices are:

- *constant* : ag assumes into the local class L a costant value set by the designer; we denote this case by $MT[L][ag] =$ `const` (see the `rank` attribute).
- *undefined* : ag is set undefined into the local class L; we denote this case by $MT[L][ag] =$ `null`.

5 Semantic query optimization and reformulation

The optimized query reformulation is obtained on the basis of the computed Base Extension set and by using the semantic query optimization techniques previously developed by the authors [1].

5.1 Base Extensions

Intuitively, given a global class, a Base Extension gathers up all objects of some local classes such that the set of Base Extensions satisfies all the extensional relationships defined over the set of local classes and allows a partitioning of the set of the sources objects.

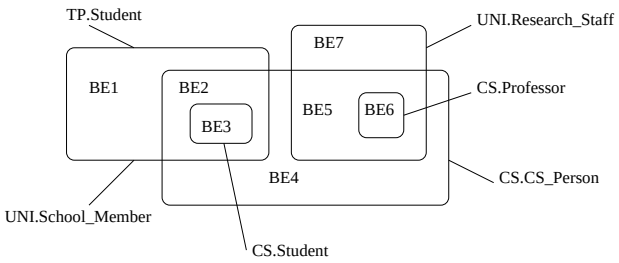

Fig. 4. Base extension Set for the global class University_Person

CL \ BE	BE1	BE2	BE3	BE4	BE5	BE6	BE7
UNI.School_Member	1	1	1	0	0	0	0
UNI.Research_Staff	0	0	0	0	1	1	1
CS.CS_Person	0	1	1	1	1	1	0
CS.Student	0	0	1	0	0	0	0
CS.Professor	0	0	0	0	0	1	0
TP.Student	1	1	1	0	0	0	0

Fig. 5. Tabular representation of the Base Extension Set of University_Person

Definition 1 (Base Extension set). *Let $G = (\boldsymbol{L}_G, \boldsymbol{GA}, MT)$ be a global class and $\mathcal{I}$ be an instance of the inter-sources schema σ: $\boldsymbol{L} \rightarrow \mathbf{S}(\boldsymbol{LA}, \boldsymbol{L})$. A* set of base extensions *of G on $\mathcal{I}$ is a pair $(\boldsymbol{B}, F)$, where $\boldsymbol{B}$ is a set of* base extension names *(denoted by $B_1, B_2, \ldots$), $\mathcal{I}(B_i) = \bigcap_{L \in F(B)} \mathcal{I}(L)$, and F is a total function F : $\boldsymbol{B} \rightarrow 2^{\boldsymbol{L}_G}$ such that: $\bigcup_{B \in \boldsymbol{B}} F(B) = \boldsymbol{L}_G$, and the set $\left\{ \mathcal{I}(B) - \bigcup_{L \in (\boldsymbol{L}_G - F(B))} \mathcal{I}(L) \mid B \in \boldsymbol{B} \right\}$ is a partition of $\bigcup_{L \in \boldsymbol{L}_G} \mathcal{I}(L)$.*

At present, we adopt the algorithm of [12] to determine a base extension set[4].

Figure 4 shows the Base Extension Set for the Global Class `University_Person`. A Base extension set of a Global Class G is represented by a table. Table rows represent the local classes of the global class and table columns represent the base extensions. The presence of a `1` in the table cell (L, B) means $L \in F(B)$ (e.g. see Figure 5).

The attributes of a Base Extension B are the global attributes which are mapped, by a not null mapping, into a local class of B. Formally:

Definition 2 (Base Extension Attributes). *Let $G = (\boldsymbol{L}_G, \boldsymbol{GA}, MT)$ be a global class and $(\boldsymbol{B}, F)$ be the set of base extensions of G, then the attributes of a base extension $B \in \boldsymbol{B}$ are defined as:*

$$A(B) = \{ag \in \boldsymbol{GA} \mid \exists L \in F(B), MT[L][ag] \neq \texttt{null}\}.$$

[4] More than one base extension set can be obtained on the basis of the above definition; the discussion about the quality of the selected set is out of the scope of this paper.

$A(BE1) = \{\texttt{name}, \texttt{year}, \texttt{school}, \underline{\texttt{rank}}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax\_fee}\}$

$A(BE2) = \{\texttt{name}, \texttt{year}, \texttt{school}, \underline{\texttt{rank}}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax\_fee}\}$

$A(BE3) = \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax\_fee}, \texttt{takes}\}$

$A(BE4) = \{\texttt{name}, \underline{\texttt{school}}\}$

$A(BE5) = \{\texttt{name}, \underline{\texttt{rank}}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}, \underline{\texttt{school}}\}$

$A(BE6) = \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}, \texttt{belong\_to}, \underline{\texttt{school}}\}$

$A(BE7) = \{\texttt{name}, \underline{\texttt{rank}}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}\}$

Fig. 6. Base Extension Attributes

The Base Extension Attributes of our example are shown in Figure 6, where underlined attributes correspond to constant mapping.

Definition 3 (Domination). *Given a global class* $G = (\boldsymbol{L}_G, \boldsymbol{GA}, MT)$*, the set of base extensions* $(\boldsymbol{B}, F)$ *of* G*, and* $B_1, B_2 \in \boldsymbol{B}$ *we say that* B_1 dominates (dom) B_2 *w.r.t. the set of global attributes* $X \subseteq \boldsymbol{GA}$ *iff* $X \subseteq A(B_1) \cap A(B_2) \wedge F(B_1) \subset F(B_2)$.

5.2 Query plan and execution

We will show the effectiveness of our optimization method by means of the following (mediated) query:

```
Q: select e_mail
    from University_Person
    where school = 'eng'
    and (s_code = 'a1x' or year = '2001' or belong_to = 'eng')
```

Processing the above query, without considering extensional knowledge, would individuate all the local classes for which at least one of the mediated query attributes has a not null mapping with it. The candidate local classes are thus all the local classes of `University_Person` but `UNI.Research_Staff` and `CS.CS_Person`. The query has thus to be reformulated on the basis of the above classes.

The approach is performed in the following steps.

1. Determination of the Local Class Set We consider the *Disjunctive Normal Form - DNF* of the query condition: $DNF =$ `F1 or F2 or F3` where:

`F1` = `(school='eng') and (s_code='a1x')`

`F2` = `(school='eng') and (belong_to='eng')`

`F3` = `(school='eng') and (year='2001')`

For each factor F of DNF we define the set: $BE(F) = \{B \mid \forall\ ag \text{ of } F, ag \in A(B)\}$, i.e., $B \in BE(F)$ iff $A(B)$ contains all the global attributes of the factor

F. Then we define the set $BE_{min}(F) = \{B \mid B \in BE(F) \wedge \nexists\ B^{'} \in BE(F) \mid B^{'} dom\ B\}$.

In our example $BE(\texttt{F1}) = BE(\texttt{F3}) = \{BE1, BE2, BE3\}$ and $BE(\texttt{F2}) = \emptyset$, since the involved attributes, i.e. `school` and `belong_to`, are contained in A(BE6), but the `school` is mapped to the constant value `'cs'` and thus `(school='eng')` is false.

Intuitively, a factor F of DNF such that $BE(F) = \emptyset$ can be eliminated as the value of F is always false. In our example, we obtain a simplified $DNF =$ `F1 or F3`. On the basis of this simplification, we obtain the following result: we do not have to access the local class `CS.Professor` as the only conjunctive predicate related to their attributes `(school='eng') and (belong_to='eng')` has been eliminated. Local classes are determined by considering the union of all the local classes included in $BE_{min}(F_i)$. With reference to our example, we have $BE_{min}(\texttt{F1}) = BE_{min}(\texttt{F3}) = \{BE1\}$, as $BE1$ dominates both $BE2$ and $BE3$; as a consequence the identified local classes are: {`TP.Student`, `UNI.School_Member`}. Notice that the classes `CS.Student` and `CS.CS_Person` are excluded from query execution too: they are useless as their objects are also instances of other local classes (as, for instance, stated by `CS.Student` NT_{Ext} `UNI.School_Member`) and their contributions to the query, that is attributes `school` and `e_mail`, are already provided by the identified local classes.

2. Query Reformulation For each pair of local classes belonging to the same base extension the related join attributes are considered. In our example, we have only a base extension, $BE1$, then the local classes are `TP.Student` and `UNI.School_Member` and the join attribute is `name` for both the classes.

Then, we consider the simplified DNF obtained in the previous phase and, for each factor F, we build a *local query* for each local class of $BE(F)$. The query `<condition>` is the is the conjunction of all predicates of the factor F which can be *solved* in the local class `L` (at least one predicate since `L`$\in BE(F)$) and the `<select-list>` is obtained by adding to the query select all the join attributes.

In our example, we have four local queries (a local query for each factor):

```
QF1L1: select e_mail, name      QF1L2: select e_mail, name
       from TP.Student                 from UNI.School_Member
       where (school_name='eng')       where (school = 'eng')
       and (s_code='a1x')

QF3L1: select e_mail, name      QF3L2: select e_mail, name
       from TP.Student                 from UNI.School_Member
       where (school_name='eng')       where (school = 'eng')
                                       and (year = 2001)
```

3. Local Query Execution For each source, by using ODB-Tools, we calculate the query inheritance hierarchy w.r.t. subsumption relationship and we send to the wrapper only the most generalized queries, enriched in the `<select-list>`

with the local attributes contained into the other local queries, to be translated and executed by the local sources. This approach reduces the data access cost by the avoidance of redudant multiple accesses to the local sources. In the example we execute only QF1L2 and QF3L1, rewrited with other local attributes:

```
QF3L1: select e_mail,name,s_code   QF1L2: select e_mail,name,year
       from TP.Student                     from UNI.School_Member
       where (school_name='eng')           where (school = 'eng')
```

QF1L1 and QF3L2 will be obtained at the mediator level on the basis of local queries results:

```
QF1L1: select e_mail, name      QF3L2: select e_mail, name
       from QF1L2                      from QF3L1
       where (s_code='a1x')            where (year = 2001)
```

4. Mediated Query Execution The first step of the Mediated Query Execution is the object fusion of the local query results belonging to the same base extension; this is implemented by a query, called *object fusion query*, which is performed for each factor and each base extension previously individuated.

In our example, for factor F1 we have only the base extension $BE1$ and the object fusion of QF1L1 and QF1L2 is based on the direct–join on the join attribute name, thus we obtain the following object fusion query (the analogously for F1):

```
QF1BE1: select e_mail                  QF3BE1: select e_mail
        from QF1L1,QF1L2                       from QF3L1,QF3L2
        where QF1L1.name=QF1L2.name            where QF3L1.name=QF3L2.name
```

of course, all the object fusion queries have the same <select-list>.

The second and last step of the Mediated Query Execution is the full outer join of the object fusion queries: in our example, the full outer join between QF1BE1 and QF3BE1.

6 Concluding remarks

In this paper, we have described the semantic integration and query optimization process for heterogeneous data sources with reference to the ARTEMIS/MOMIS system. Main features of ARTEMIS/MOMIS can be summarized as follows: i) use of formalization and reasoning capabilities of Description Logics for a semantically rich representation of heterogeneous data sources for both semantic integration and query reformulation and optimization: ii) semiautomatic extraction of interschema properties relating schema concepts of data sources at the conceptual level; iii) derivation of an integrated and unified representation of the involved data sources with explicit representation of the schema mappings and interschema knowledge exploited for query processing and semantic optimization.

References

1. D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in OODB. In *Int. Conf. on Data Engineering - ICDE97*, 1997. http://sparc20.dsi.unimo.it.
2. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogenous information sources. *Data and Knowledge Engineering*, 36(3):215–249, 2001.
3. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Data and Knowledge Engineering*, 13(2), 2001.
4. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
5. S. Cluet, P. Veltri, and D. Vodislav. Views in a large scale XML repository. In *Proc. of the VLDB 2001*, Roma, Italy, 2001.
6. A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. of ACM SIGMOD*, Santa Barbara, California, USA, 2001.
7. O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proc. of the Sixteenth ACM SIGMOD Symposium on Principles of Database Systems*, 1997.
8. R. Goldman, J. McHugh, and J. Widom. From semistructured data to XML: Migrating the lore data model and query languages. In *Proc. of International Workshop on the Web and Databases (WebDB'99)*, pages 25–30, Philadelphia, Pennsylvania, USA, 1999.
9. R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'97)*, 1997.
10. J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with Cupid. In *Proc. of VLDB 2001*, Roma, Italy, 2001.
11. R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *Proc. of VLDB 2000*, pages 484–495, Cairo, Egypt, 2000.
12. I. Schmitt and C. Türker. An Incremental Approach to Schema Integration by Refining Extensional Relationships. In *Proc. of the ACM CIKM 98*, pages 322–330, New York, 1998. ACM Press.