# Exploiting extensional knowledge for a mediator based Query Manager

D. Beneventano[1,2], S. Bergamaschi[1,2], F. Guerra[1], and M. Vincini[1]

[1] Dipartimento di Scienze dell'Ingegneria - Università di Modena e Reggio Emilia
DSI - Via Vignolese 905, 41100 Modena
{domenico.beneventano,sonia.bergamaschi, guerra.francesco, maurizio.vincini}@unimo.it
[2] CSITE-CNR Bologna V.le Risorgimento 2, 40136 Bologna

**Abstract.** Query processing in global information systems integrating multiple heterogeneous sources is a challenging issue in relation to the effective extraction of *information* available on-line. In this paper we propose intelligent, tool-supported techniques for querying global information systems integrating both structured and semistructured data sources. The techniques have been developed in the environment of a data integration, wrapper/mediator based system, MOMIS, and try to achieve two main goals: *optimized query reformulation* w.r.t local sources and *object fusion*, i.e. grouping together information (from the same or different sources) about the same real-world entity.
The developed techniques rely on the availability of *integration knowledge*, i.e. local source schemata, a virtual mediated schema and its *mapping descriptions*, that is semantic mappings w.r.t. the underlying sources both at the *intensional* and *extensional* level. Mapping descriptions, obtained as a result of the semi-automatic integration process of multiple heterogeneous sources developed for the MOMIS system, include, unlike previous data integration proposals, *extensional intra/interschema knowledge*. Extensional knowledge is exploited to detect extensionally overlapping classes and to discover implicit join criteria among classes, which enables the goals of *optimized query reformulation* and *object fusion* to be achieved. The techniques have been implemented in the MOMIS system but can be applied, in general, to data integration systems including extensional intra/interschema knowledge in *mapping descriptions*.

## 1 Introduction

The purpose of data integration is to provide a uniform interface to multiple heterogeneous sources. Applications range from *searching information on the net* to *providing an uniform consistent view* of data associated with the various legacy systems of an enterprise. Query processing in such global information systems environment is a challenging issue which has been faced in many previous works in both the AI and Database Community [2, 3, 18, 15, 22, 7, 6]. A data integration system, based on conventional wrapper/mediator architectures, usually allows the user to pose a query and receive a unified answer without the need of: *locating* the sources relevant to the query, *interacting* with each source in isolation and *combining* the data coming from the different sources. Data integration systems usually follow this architecture: each data source provides a schema and a mediated (global) *virtual* schema of all the sources is obtained manually or semi-automatically, for a particular integration application. The mediated schema has a set of *mapping descriptions* (called source descriptions in [16]) that specify the semantic mapping between the mediated schema and the sources schema. The data integration system uses these mapping descriptions to reformulate a user query into queries over the source schemata.

Unlike previous mentioned approaches, *mapping descriptions* obtained as a result of the semi-automatic integration process developed for the MOMIS system [5, 4], include *extensional in-*

*tra/interschema knowledge* which is an important step in the pre-integration phase [25]. For instance, if there are two classes Person in two different sources, then these classes may contain instances corresponding to the same real-world object or may refer to disjoint sets of real-world objects. Handling *extensional relationships* among object classes of different schemata is a fundamental task performed by the integration designer for a correct and complete schema integration.

As stated in [21], one of the main tasks of mediators is to fuse information from heterogeneous information sources. This may involve, for example, removing redundancies, and resolving inconsistencies in favor of the most reliable source. Mediators play the central role in information integration, and one of their most important task is to perform *object fusion*. This involves grouping together information (from the same or different sources) about the same real-world entity. In doing this fusion, the mediator may also "refine" the information by removing redundancies, resolving inconsistencies between sources in favor of the most reliable source, and so on. A mediator may also have to avoid accessing to a particular source if, on the basis of extensional interschema knowledge, another involved source includes the information of such a source, or will provide an empty answer or if another source provides similar information at a lower cost (either financial or computational).

From a theoretical point of view, solving a user (mediated) query, i.e. giving a single unified answer w.r.t. multiple sources, implies to face two main problems: *query reformulation/optimization* and *object fusion*. Much research effort has concentrated on Query reformulation/optimization [10, 20, 11, 12, 23, 17, 19], whereas, the object fusion problem has received little attention [21, 26]. We believe that it is a relevant topic now and will become more important in the future as integration systems cope with more and more information that has not been nicely structured and partitioned in advance. To solve *Object Fusion* it is necessary to devise a theoretical framework that exploits all the available integration knowledge, and, in particular, extensional inter/intra schema knowledge.

The theoretical framework we propose provides intelligent, tool-supported techniques to query global information systems integrating both structured and semistructured data sources.

The techniques have been developed in the environment of a data integration, wrapper/mediator based system, MOMIS, and rely on the availability of integration knowledge, i.e. local sources schemata, a virtual mediated schema and its *mapping descriptions*, i.e. semantic mappings w.r.t. the underlying sources both at the intensional and *extensional* level. Mapping descriptions include, unlike previous data integration proposals, *extensional intra/interschema knowledge*.

Extensional knowledge is exploited to detect extensionally overlapping classes and to discover implicit join criteria among classes, thus allowing to achieve the *optimized query reformulation* and *object fusion* goals. In particular, starting from the method developed in [25], we exploit the "base extension" approach in order to face the reformulation/optimization problem of a mediated query and, on the basis of mapping descriptions, we develop a semi–automatic method to discover implicit join rules among classes in order to face the object fusion problem.

The techniques have been implemented in the MOMIS system but can be applied, in general, to data integration systems including extensional intra/interschema knowledge in the *mapping descriptions*.

### 1.1 Overview of the MOMIS system

Like other integration projects [1, 24], MOMIS follows a "semantic approach" to information integration based on the conceptual schema, or metadata, of the information sources, and on the $I^3$ architecture [14] (see Figure 1); for a detailed description of the MOMIS system see [6, 4] available at http://www.dbgroup.unimo.it/Momis. The system is composed by the following functional elements that communicates using the CORBA [13] standard:

1. a common data model, $\text{ODM}_{I^3}$, which is defined according to the $\text{ODL}_{I^3}$ language, to describe source schemas for integration purposes. $\text{ODM}_{I^3}$ and $\text{ODL}_{I^3}$ have been defined as subset of the corresponding ones in ODMG, following the proposal for a standard mediator language developed by the $I^3$/POB working group [8]. In addition, $\text{ODL}_{I^3}$ introduces new constructors (intensional/extensional relationships) to support the semantic integration process;
2. *Wrappers*, placed over each sources, translate metadata descriptions of the sources into the common $\text{ODL}_{I^3}$ representation, translate (reformulate) a global query expressed in the $\text{OQL}_{I^3}$[1] query language into queries expressed in the sources languages and export query result data set;
3. a *Mediator*, which is composed of two modules: the *Global Schema Builder* (GSB) and the *Query Manager* (QM). The GSB module processes and integrates $\text{ODL}_{I^3}$ descriptions received from wrappers to derive the mediated schema. The QM module performs query processing and optimization. The QM generates $\text{OQL}_{I^3}$ queries to be sent to wrappers starting from each query posed by the user on the mediated schema. QM automatically generates the translation of the query into a corresponding set of sub-queries for the sources and synthesize a unified global answer for the user.

The original contribution of MOMIS is related to the availability of a set of techniques for the designer to face common problems that arise when integrating pre-existing information sources. MOMIS provides the capability of explicitly introducing many kinds of knowledge for integration, such as integrity constraints, intra- and inter-source intensional and extensional relationships, and designer supplied domain knowledge. A *Common Thesaurus*, which has the role of a shared ontology of the source is built in a semi-automatic way. The *Common Thesaurus* is a set of intra and inter-schema intensional and extensional relationships, describing inter-schema knowledge about classes and attributes of sources schemas; it provides a reference on which to base the identification of classes candidate to integration and subsequent derivation of their global representation.

MOMIS supports information integration in the creation of a mediated schema of all sources (Global Schema) combining reasoning capabilities of Description Logics (ODB-Tools in Figure 1) with affinity-based clustering techniques (Artemis in Figure 1), by exploiting a common ontology for the sources constructed using lexical knowledge from WordNet and validated integration knowledge.

The user application interacts with MOMIS to query the Global Schema by using the $\text{OQL}_{I^3}$ language. This phase is performed by the QM that generates the $\text{OQL}_{I^3}$ queries for wrappers. Using Mapping Descriptions and DLs techniques, the QM generates in an automatic way the reformulation/optimization of the generic $\text{OQL}_{I^3}$ query into different sub-queries, one for each involved local source.

To achieve the mediated query result, the QM has to assemble each local sub-query result into a unified data set. This process involves the solution of redundancy and reconciliation problems, due to the incomplete and overlapping information available on the local sources, i.e. *Object Fusion*.

[1] $\text{OQL}_{I^3}$ is a subset of OQL-ODMG.

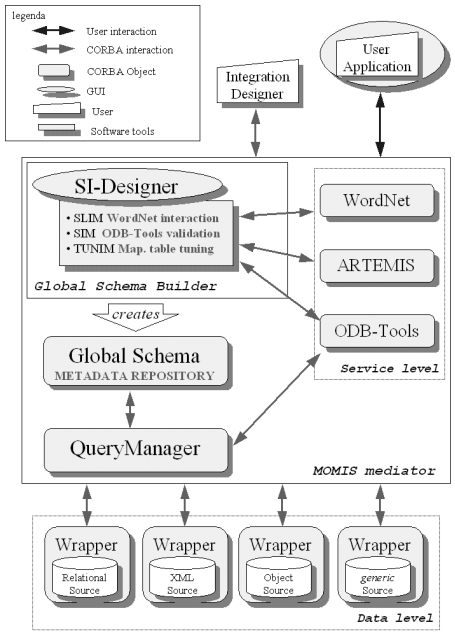


**Fig. 1.** the MOMIS system architecture

As a mediator is not the *owner* of the data stored in the local classes but it only provides a virtual view, this means that the mediator has to *recognize* instances of the sources to be fused in an object. This recognition is a difficult task as: each source may have its own techniques to identify objects, like keys for relational or OIDs for object sources, and, usually instances referring to the same real word object are identified with different keys or OIDs, depending on the source the object is stored. The idea of our approach is to find *semantically homogeneous attributes* for each instance of each local class, on the basis of the available integration knowledge.

The outline of the paper is the following. Section 2 presents preliminaries and the formalization of Mapping Descriptions. In particular, Subsection 2.1 outlines the MOMIS approach to data integration and $\text{ODL}_{I^3}$ relationships together with a running example which will be used in the remainder of the paper. Subsection 2.2 provides the the formalization of Mapping Descriptions. Section 3 introduces our solution to the Object Fusion problem. Finally, Section 4 presents the MOMIS Query Manager implementing the proposed theoretical framework.

The authors are aware that the presented work is quite preliminary, but think that this new approach may merit attention.

## 2 Mapping Descriptions

### 2.1 Preliminaries

The MOMIS approach to intelligent schema integration is supported by a tool, SI-Designer [4] and is articulated in the following phases:

1. *Generation of a Common Thesaurus*
2. *Affinity analysis of* $\text{ODL}_{I^3}$ *classes* Relationships in the *Common Thesaurus* are used to evaluate the level of *affinity* between classes intra and inter sources. The concept of affinity is introduced to formalize the kind of relationships that can occur between classes from the integration point

of view. The affinity of two classes is established by means of affinity coefficients based on class names, class structures and relationships in *Common Thesaurus*

3. *Clustering $ODL_{I^3}$ classes* Classes with affinity in different sources are grouped together in clusters using hierarchical clustering techniques. The goal is to identify the classes that have to be integrated since describing the same or semantically related information
4. *Generation of the mediated schema* A *global class* is defined for each cluster, which is representative of all cluster's classes and is characterized by the union of their attributes. The global schema for the analyzed sources is composed of all the global classes derived from clusters, and is the basis for posing queries against the sources.

For a detailed description of the method see [6, 4]. In the following we briefly introduce the $ODL_{I^3}$ primitives related to intensional/extensional relationships and the running example that will be used in the remainder of this paper (Figure 2).

### $ODL_{I^3}$ relationships

In order to permit a semantically rich representation of source schemas relationships, $ODL_{I^3}$ introduces the following primitives:

**Intensional relationships.** These are *terminological relationships* expressing intra and inter-schema knowledge 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), defined between two terms $t_i$ and $t_j$, with $t_i \neq t_j$, that are considered synonyms in every considered source (i.e., $t_i$ and $t_j$ can be indifferently used in every source to denote a certain concept).
- BT (Broader Terms), or hypernymy, defined between two terms $t_i$ and $t_j$ such as $t_i$ has a broader, more general meaning than $t_j$. BT relationship is not symmetric. The opposite of BT is NT (Narrower Terms), or hyponymy.
- RT (Related Terms), or positive association, defined between two terms $t_i$ and $t_j$ that are generally used together in the same context in the considered sources.

An intensional relationships has no implications for the extension/compatibility of the structure (domain) of the two involved classes (attributes). Consequently, our notion of intensional relationships is different from the one proposed by Catarci and Lenzerini [9], where an intensional relationships has some extensional import.

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

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

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

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

$C_1$ $\text{DISJ}_{ext}$ $C_2$: this means that the instances of $C_1$ are disjoint from the instances of $C_2$.

In contrast with [25] we do not introduce an *overlap relationship* as we assume a default overlap relationships among two classes if no extensional relationship is specified. Moreover, extensional relationships "constrain" the structure of the two classes $C_1$ and $C_2$ , that is $C_1$ $\text{NT}_{ext}$ $C_2$ is semantically equivalent to an "isa" relationship.

**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. 2.** Three heterogeneous University Sources

**Running example** We consider three sources with different data model. The first source is a relational database, `University` ($S_1$), containing data about the staff and the students of a given university. The relations are: `Research_Staff`, `School_Member`, `Department`, `Section` and `Room`. For a given professor (in `Research_Staff`) his department (`dept_code`) and his section (`s_code`) are stored. In the relation `School_Member` the information `name`, `year` and `school` about students enrolled at the university are stored.

The second source `Computer_Science` ($S_2$) is an object–oriented database containing information about people belonging to the computer science department of the same university, and is an object-oriented database. There are six classes: `CS_Person`, `Professor`, `Student`, `Division`, `Location` and `Course`. Information is quite similar to the first source: it stores data on professors and students, also giving the possibility to retrieve the division of a given professor. This division may be part of another department, being a logical specialization of `Department`. The class `Location` maintains the division address. With respect to students, we may know the courses they take and their year.
A third source is also available, `Tax_Position` ($S_3$), derived from the Registry Office. It consists of an XML file, storing information about student's tax_fees.

### 2.2 Global Class and Mapping Tables

Starting from the output of the cluster generation (*Clustering $ODL_{I^3}$ classes* , see Figure 3), we define, for each cluster, a *Global Class* that represents the mediated view of all the classes of the cluster. For each global class a set of *global attributes* and, for each of them, the mappings with the

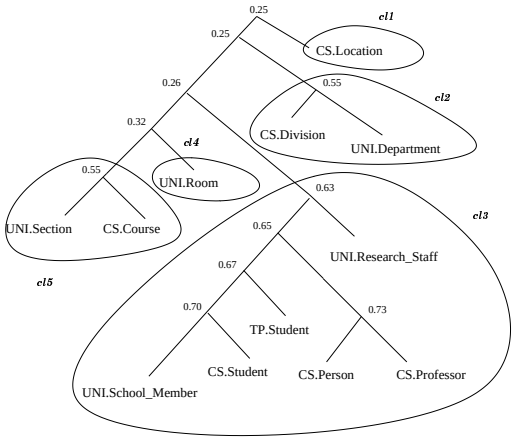


**Fig. 3.** Example of affinity tree and selected Clusters

| 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. 4.** Mapping Table of the Global Class University_Person

*local attributes* (i.e. the attributes of the local classes belonging to the cluster) are given [2]. Briefly, we can say that the global attributes are obtained in two steps: (1) Union of the attributes of all the classes belonging to the cluster; (2) Fusion of the "similar" attributes; in this step redundancies are eliminated in a semi–automatic way taking into account the relationships stored in the *Common Thesaurus*. For each global class a persistent *mapping-table* storing all the mappings is generated; it is a table whose rows represent the set of the local classes which belong to the cluster and whose columns represent the global attributes. An element $MT[L][ga]$ represents the set of attributes of the local class $L$ which are mapped into the global attribute $ga$: the value of the $ga$ attribute is a function of the values assumed by the set of attributes $MT[L][ga]$. Some simple and frequent cases of such function are the following (see Figure 4 as an example):

- *identity* : the $ga$ value *is equal to* the $la$ value; we denote this case as $MT[L][ga] = la$.
- *concatenation* : the $ga$ 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 $MT[L][ga] = la_1$ `and` $\ldots$ `and` $la_n$ (see $MT$[`CS.Student`][`name`] in Figure 4).

When the global attribute $ga$ has no correspondence with any attribute of the local class $L$, the designer can choose between the following two solutions:

[2] For a detailed description of the mappings selection and of the tool SI-Designer which assist the designer in this integration phase see [4].

- *constant* : the global attribute $ga$ assumes into the local class $L$ a costant value set by the designer; we denote this case by $MT[L][ga] = \texttt{const}$ (see the Rank attribute).
- *undefined* : the global attribute $ga$ is set undefined into the local class $L$; we denote this case by $MT[L][ga] = \texttt{null}$.

Formally, let $\mathbf{L}$ be a set of *local class names* (denoted by $L_1, L_2, \ldots$) and let $\mathbf{LA}$ be a set of *local attributes* (denoted by $la_1, la_2, \ldots$); $L_A$ is a total function $L_A \colon \mathbf{L} \rightarrow 2^{\mathbf{LA}}$ which associates local class names with attributes. Let $\mathbf{GA}$ be a set of *global attributes* (denoted by $ga_1, ga_2, \ldots$).

**Definition 1 (Global Class).** *Given a set $\boldsymbol{L}$ of local class names and a set $\boldsymbol{GA}$ of global attributes a* global class *$G$ is a tuple $G = (\boldsymbol{L}, \boldsymbol{GA}, MT)$ where $MT$, called* mapping table, *is a total function*

$$MT : \boldsymbol{L} \times \boldsymbol{GA} \rightarrow 2^{\boldsymbol{L}} \cup \{const\} \cup \{null\}$$

### 2.3 Extensional Relationships and Base Extensions

**Definition 2 (Extensional Relationships Set).** *Given a set $\boldsymbol{L}$ of local class names, by ERS we denote an* extensional relationships set *among classes of $\boldsymbol{L}$:*

$$ERS \subseteq \left\{ L_1 \Theta L_2 \mid L_1, L_2 \in \boldsymbol{L} \text{ and } \Theta \in \{\mathrm{SYN}_{ext}, \mathrm{BT}_{ext}, \mathrm{NT}_{ext}, \mathrm{DISJ}_{ext}\} \right\}$$

As an example, let us consider the following extensional relationships among the classes of `University_Person` expressing the integration designer's knowledge about the sources:

1. `UNI.School_Member` $\mathrm{SYN}_{Ext}$ `TP.Student`
2. `CS.Student` $\mathrm{NT}_{Ext}$ `UNI.School_Member`
3. `CS.Professor` $\mathrm{NT}_{Ext}$ `UNI.Research_Staff`
4. `CS.Professor` $\mathrm{DISJ}_{Ext}$ `UNI.School_Member`
5. `UNI.Research_Staff` $\mathrm{DISJ}_{Ext}$ `TP.Student`
6. `UNI.Research_Staff` $\mathrm{DISJ}_{Ext}$ `CS.Student`
7. `CS.Student` $\mathrm{NT}_{Ext}$ `CS.CS_Person`
8. `CS.Professor` $\mathrm{NT}_{Ext}$ `CS.CS_Person`

Relationships from 1 to 6 are designer-supplied inter-schema relationships; 7 and 8 are intra–schema extensional relationships automatically extracted by the system from the isa relationships of the `COMPUTER_SCIENCE` source.

Intuitively, a Base Extension is a partitioning of the set of the sources objects. The starting point are the instances of the local classes: given a local class $L$, $\mathcal{I}(L)$ denotes the set of its real-world objects. An instance of a global class G is an instance satisfying all the extensional relationships defined over the set of local classes.

**Definition 3 (Global Class Instance).** *Given a global class $G = (\boldsymbol{L}, \boldsymbol{GA}, MT)$ let $\mathcal{I} \colon \boldsymbol{L} \rightarrow 2^{\boldsymbol{D}}$ be a function which associates local classes to their instances, where $\boldsymbol{D}$ is the set of real-world objects. Let ERS be a set of extensional relationships among classes of $\boldsymbol{L}$. We say that $\mathcal{I}$ is an instance of $G$ w.r.t. ERS if*

1. $\mathcal{I}(L_1) \equiv \mathcal{I}(L_2), \ \forall L_1 \mathrm{SYN}_{ext} L_2 \in ERS$
2. $\mathcal{I}(L_1) \subseteq \mathcal{I}(L_2), \ \forall L_1 \mathrm{NT}_{ext} L_2 \in ERS$
3. $\mathcal{I}(L_1) \supseteq \mathcal{I}(L_2), \ \forall L_1 \mathrm{BT}_{ext} L_2 \in ERS$
4. $\mathcal{I}(L_1) \cap \mathcal{I}(L_2) = \emptyset, \ \forall \ L_1 \mathrm{DISJ}_{ext} L_2 \in ERS$

**Definition 4 (Base Extension).** *Given a global class $G = (\boldsymbol{L}, \boldsymbol{GA}, MT)$ and a set of extensional relationships ERS, let $\mathcal{I}$ an instance of $G$ w.r.t. ERS. 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 $B_1, B_2, \ldots$) and $F$ is a total function $F : \boldsymbol{B} \rightarrow 2^{\boldsymbol{L}}$ such that: $\bigcup_{B \in \boldsymbol{B}} F(B) = \boldsymbol{L}$ and the set $\left\{ \bigcap_{L \in F(B)} \mathcal{I}(L) - \bigcup_{L \in (\boldsymbol{L} - F(B))} \mathcal{I}(L) \mid B \in \boldsymbol{L} \right\}$ is a partition of $\bigcup_{L \in \boldsymbol{L}} \mathcal{I}(L)$.*

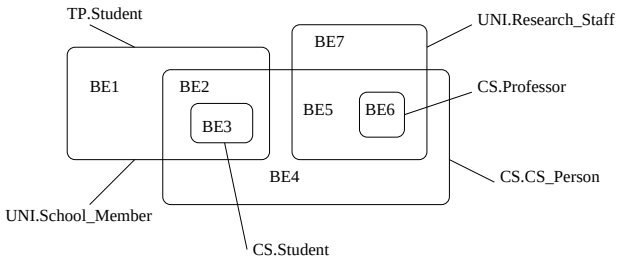


**Fig. 5.** Base extension Set for the global class University_Person

| 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. 6.** Tabular representation of the Base Extension Set of University_Person

We adopt the algorithm of [25] to determine a base extension set[3]. Figure 5 shows the Base Extension Set for 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 $L \in F(B)$ (see Figure 6).

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

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

The Base Extensions Attributes of our example are showed in Figure 7.

**Definition 6 (Domination).** *Given a global class $G = (\boldsymbol{L}, \boldsymbol{GA}, MT)$ and a set of base extensions $(\boldsymbol{B}, F)$ of $G$, for all $B_1, B_2 \in \boldsymbol{B}$ we say that $B_1$* dominates *$B_2$ w.r.t. the set of global attributes $X \subseteq \boldsymbol{GA}$ iff $X \subseteq A(B_1) \cap A(B_2) \ \wedge \ F(B_1) \subset F(B_2)$.*

## 3 The Object fusion problem

The base extension set allows the detection of all the overlapping classes in a global class. For each pair of overlapping classes we have to face the *object fusion problem*, i.e. to identify instances of the same object and fuse them.

The goal is thus to find semantic identifiers for each instance of each local class *virtually* integrated at mediator level. In fact, MOMIS (as all mediator based systems) is not the owner of

[3] 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.

$A(B_1) = \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax}\}$

$A(B_2) = \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax}\}$

$A(B_3) = \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax}, \texttt{takes}\}$

$A(B_4) = \{\texttt{name}, \texttt{school}\}$

$A(B_5) = \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}, \texttt{school}\}$

$A(B_6) = \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}, \texttt{belong\_to}, \texttt{school}\}$

$A(B_7) = \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}\}$

**Fig. 7.** Base Extensions Attributes

the objects that extracts from the sources, but only a way to access them. So, two objects coming from two different sources may share the same *oid*, even if semantically not related; on the other hand two objects may have two different identifiers even if semantically related.

We can find a semantic identifier, for the objects extracted from the sources, in local interfaces keys; in this case criteria that allow the object fusion process to be semi-automatic can be found. The classes which must be considered for object fusion *belong to the same base extension*, furthermore, intensional and extensional relationships between classes, stored in the Common Thesaurus, can be exploited to detect semantically homogeneous attributes and join attributes.

**Definition 7 (Semantically Homogeneous Attributes).** *Given a global class* $G = (\boldsymbol{L}, \boldsymbol{GA}, MT)$, *let* $L_1, L_2 \in \boldsymbol{L}$ *be two local classes and* $X_1 \subseteq L_A(L_1), X_2 \subseteq L_A(L_2)$ *two set of local attributes. We say that* $X_1$ *and* $X_2$ *are* semantically homogeneous *iff there exists a global attribute* $ga \in \boldsymbol{GA}$ *such that the set of attributes occurring in* $MT[L_1][ga]$ *(*$MT[L_2][ga]$*) is equal to* $X_1$ *(*$X_2$*).*

**Definition 8 (Join Attributes).** *Given a global class* $G = (\boldsymbol{L}, \boldsymbol{GA}, MT)$, *let* $L_1, L_2 \in \boldsymbol{L}$ *be two local classes. The* Join Attributes *between* $(L_1)$ *and* $(L_2)$ *are two sequences of set of attributes* $X_1^i \subseteq L_A(L_1)$ *and* $X_2^i \subseteq L_A(L_2)$, *such that*

- direct–join *:* $X_1^i$ *and* $X_2^i$ *are semantically homogeneous, for all* $1 \leq i \leq n$ ***or***
- indirect–join *: an explicit* mapping *between* $X_1^i$ *and* $X_2^i$ *is defined.*

Thus, for each pair of local classes $(L_1, L_2)$ belonging to the same base extension, the following types of join may occur:

- *implicit direct–join* : Using information about keys and extensional relationships between classes, it is possible to automatically identify the set of *direct* join attributes between two local classes. The most evident case is when two local key attributes are semantically homogeneous: these sets are automatically detected as join attributes (the *QM* automatically generates an equijoin predicate among the involved local classes). For example: `CS.Student` $\mathrm{NT}_{Ext}$ `UNI.School_Member` and the set $X_1 = \{\texttt{CS.Student.first\_name}, \texttt{CS.Student.last\_name}\}$ is a key for the local class `CS.Student` and it is semantically homogeneous to the set $X_2 = \{\texttt{UNI.School\_Member.name}\}$: in fact, $X_1$ and $X_2$ are mapped into the same global attribute `name` (see figure 4).

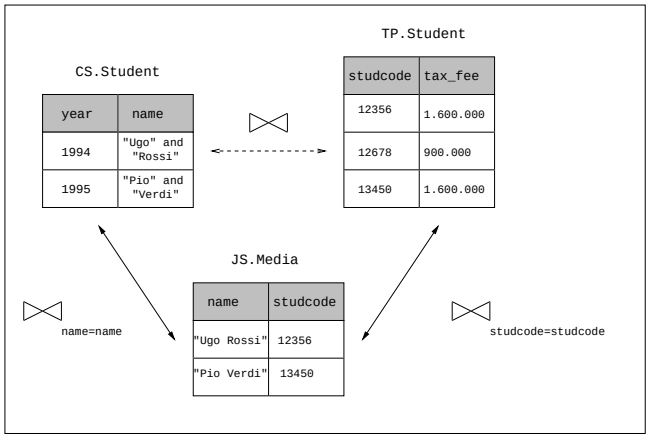


**Fig. 8.** Indirect–join by a Join Table

- *direct–join* : the designer explicitely introduces the two sets of join attributes, $X_1$ and $X_2$ and an equijoin predicate. In this case, the QM verify that the join attributes, $X_1$ and $X_2$, be *semantically homogeneous*, i.e. they are mapped into the same set of global attribute in the Mapping Table.
- *indirect–join* : the join is *indirectly* performed between the two sets of join attributes $X_1$ and $X_2$ through a designer–defined *mapping*. In this case, it is not necessary that $X_1$ and $X_2$ be semantically homogeneous. For example, to define a join criteria between `CS.Student` and `TP.Student` the designer defines a *Join Table* where the local attributes `name` and `s_code` are explicitly related (see figure 8).

## 4 The MOMIS Query Manager

We show the optimized query reformulation obtained with our method on the following (mediated) query:

```
Q: select email
   from University_Person
   where school = 'cs'
   and (s_code = 'a1x' or
        section = 'info1' or year = '2001')
```

Processing the above query, without considering extensional relationships, would individuate all the local classes for which at least one of the mediated query attributes has a not null mapping with it: `UNI.Research_Staff`, `UNI.School_Member`, `CS.Student` and `TP.Student`. The query has thus to be reformulated on the basis of the above classes. By considering extensional relationships it is possible to avoid the access to some local class, obtaining an effective query optimization, independent from any specific cost model. In our method, extensional relationships are considered by using the computed base extensions. In Figure 9 we show, and outline in the following, query processing phases on the basis of our method.

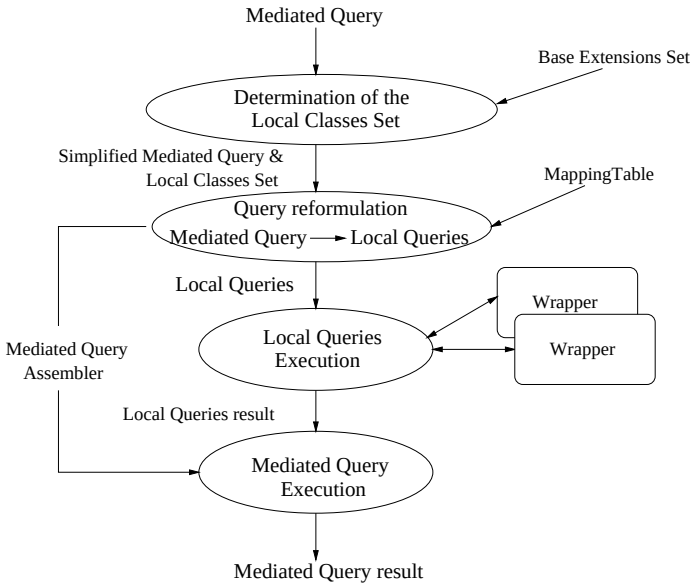


**Fig. 9.** Query processing phases

**Determination of the Local Classes set**

1. the subset of Base Extensions including the global attributes of the query is computed
2. the local classes included into the Base Extension subset computed at point 1. are determined

**Query Reformulation** w.r.t local sources.

1. query reformulation into local queries on the basis of the classes identified in the previous phase

**Local query execution**

1. local queries are sent to the wrappers to be translated and executed by the local sources

**Mediated query execution**

1. for each Base Extension object fusion of the local queries results is performed
2. the union of the results of the queries identified in the previous phase is performed
3. residual predicates (i.e., the predicates which are not included in any local query) are solved.

**Determination of the Local Classes Set** We consider the *Disjunctive Normal Form - DNF* of the query condition; for the query `Q` we have: $DNF = F_1$ `or`$F_2$ `or`$F_3$ where:

$F_1$ = `(school='cs') and (s_code='a1x')`

$F_2$ = `(school='cs') and (section='info1')`

$F_3$ = `(school='cs') and (year='2001')`

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

- $BE(F_1) = BE(F_3) = \{BE1, BE2, BE3\}$
- $BE(F_2) = \emptyset$

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 $DNF = F_1$ `or`$F_3$.

On the basis of this simplification, we obtain the following result: we do not have to access the local class `UNI.Research_Staff` as the only predicate related to its attributes `(section =`

'info1') has been eliminated. This optimization result is due to the extensional relationships: UNI.Research_Staff $DISJ_{Ext}$ TP.Student and UNI.Research_Staff $DISJ_{Ext}$ CS.Student.

In the presence of more complex queries and a large set of extensional relationships the optimization results that can be obtained by using base extensions can be effective. Furthermore, in some cases, as:

```
Q: select email from University_Person
   where school = 'cs'    and section = 'info1'
```

we have $BE(F) = \emptyset$, that is an empty answer (no access at the sources).

Local classes are determined by considering the union of all the local classes included in the previously evaluated base extension subset and subsequently eliminating *dominated* ones. With reference to our example, we have $BE(F_1) = BE(F_3) = \{BE1\}$, as $BE1$ dominates both $BE2$ and $BE3$; as a consequence the identified local classes are: { TP.Student, UNI.School_Member }. Therefore, also the class CS.Student is excluded from query execution: it is useless as its instances are included in the ones of another local class (as stated by CS.Student $NT_{Ext}$ UNI.School_Member) and its contribution to the query, the attribute school_name, is already present in UNI.School_Member).

As to summarize, the result of this phase are: a simplified Mediated Query, a set of base extensions and, then, a Local Classes Set.

**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, $BE_1$, then the local classes are TP.Student and UNI.School_Member and the join attribute is name for both the classes. We consider the *Conjunctive Normal Form - CNF* of the simplified DNF obtained in the previous phase: CNF = (school = 'cs') and ((s_code = 'a1x') or (year = '2001')) and we build a *local query* for each local class individuated in the previous phase

```
QL: select  <select-list> from L
    where <condition>
```

in two steps:

1. <condition> is the conjunction of all factors of *CNF* which can be *solved* in the local class L; these factors are rewritten w.r.t. the attributes of local class L on the basis of the mapping table. In our example, for the class TP.Student we have <condition>=(school_name='cs') and for the class UNI.School_Member we have <condition>=(school='cs'). Note that, the same predicate may be mapped into more than one class, on semantic homogeneous attributes. From a theoretical point of view, this multiple mapping does not introduce any problem as semantic homogeneous attributes have been individuated in the integration activity; on the other hand, from an optimized execution point of view classes supporting such a predicate could be chosen among according to cost (either financial or computational) criteria.
   After considering all the local classes, the predicates which are not included in any local query are considered as *residual predicates*. In our example: ((s_code='a1x') or (year='2001')).
2. <select-list> is obtained by adding to the query attributes all the join attributes and the attributes included into the residual predicates.

In our example, we have two local queries:

```
QL1: select email, name, s_code
     from TP.Student
     where (school_name = 'cs')
```

```
QL2: select email, name, year
       from UNI.School_Member
       where (school = 'cs')
```

**Mediated Query Execution** The first step of the Mediated Query Execution is the object fusion of the local queries results of the local classes belonging to the same base extension: this is performed by a query, called *object fusion query*, for each base extension previously individuated.

In our example, we have only the base extension $BE_1$ and the object fusion of `QL1` and `QL2` is based on the direct–join on the join attribute `name`, thus we obtain the object fusion query:

```
QBE1:  select email, s_code, year
       from QL1,QL2
       where QL1.name=Ql2.name
```

where the `<select-list>` is obtained by adding to the attributes list the set of attributes of the residual predicates; of course, all the object fusion queries have the same `<select-list>`.

The second step of the Mediated Query Execution is the union of the object fusion queries. In our example, it is not performed as there is a single base extension. The third and last step of the Mediated Query Execution is the execution of the residual predicates:

```
QR: select email from QBE1
    where (s_code = 'a1x') or
          (year = '2001')
```

## 5 Conclusions

In this paper we proposed tool-supported techniques to query global information systems. The techniques have been developed in the environment of a data integration, wrapper/mediator based system, MOMIS, and try to achieve two main goals: *query reformulation* w.r.t local sources and *object fusion*, i.e. grouping together information (from the same or different sources) about the same real-world entity.

The developed techniques rely on the availability of *integration knowledge*, i.e. local sources schemata, a virtual mediated schema and its *mapping descriptions*, that is semantic mappings w.r.t. the underlying sources both at the intensional and *extensional* level. Mapping descriptions include, unlike previous data integration proposals, *extensional intra/interschema knowledge*. Extensional knowledge is exploited to detect extensionally overlapping classes and to define join criteria among classes, thus allowing the *optimized query reformulation* and *object fusion* goals to be achieved.

We are aware that the presented techniques need further investigations and extensions, but we believe that this research line is new and promising for data integration systems.

## References


1. Y. Arens, C.Y. Chee, C. Hsu, and C. A. Knoblock. Retrieving and integrating data from multiple information sources. *International Journal of Intelligent and Cooperative Information Systems*, 2(2):127–158, 1993.
2. Y. Arens, C. A. Knoblock, and C. Hsu. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.
3. Y. Arens, C. A. Knoblock, and W. Shem. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems (JIIS)*, 6(1):99–130, 1996.
4. I. Benetti, D. Beneventano, S. Bergamaschi, A. Corni, F. Guerra, and G. Malvezzi. Si-designer: a tool for intelligent integration of information. *Int. Conference on System Sciences (HICSS2001)*, 2001.

5. D. Beneventano, S. Bergamaschi, S. Castano, A. Corni, R. Guidetti, G. Malvezzi, M. Melchiori, and M. Vincini. Information integration: The momis project demonstration. In *VLDB 2000, Proc. of 26th International Conference on Very Large Data Bases, 2000, Egypt*, 2000.
6. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogenous information sources. *Journal of Data and Knowledge Engineering*, 36(3):215–249, 2001.
7. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Records*, 28(1), March 1999.
8. P. Buneman, L. Raschid, and J. Ullman. Mediator languages - a proposal for a standard, April 1996. Available at ftp://ftp.umiacs.umd.edu/pub/ONRrept/medmodel96.ps.
9. T. Catarci and M. Lenzerini. Representing and using interschema knowledge in cooperative information systems. *Journal of Intelligent and Cooperative Information Systems*, 2(4):375–398, 1993.
10. O. Duschka. Query optimization using local completeness. In *Proc. of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference*, 1997.
11. 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.
12. S. Gnanaprakasam E. Lambrecht, S. Kambhampati. Optimizing recursive information-gathering plans. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI 99*, 1999.
13. Object Management Group. Object management group. http://www.omg.org/.
14. R. Hull and R. King et al. Arpa $i^3$ reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.
15. Z. Ives, A. Levy, D. Weld, D. Florescu, and M. Friedman. Adaptive query processing for internet applications. *Data Engineering Bulletin*, 23(2):19–26, 2000.
16. Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy, and D. S. Weld. An adaptive query execution system for data integration. In *Proc. ACM SIGMOD Int. Conf. on Management of Data, USA*, 1999.
17. C.A. Knoblock J.L. Ambite. Flexible and scalable query planning in distributed and heterogeneous environments. In *Proc. of the 4th Int. Conf. on Artificial Intelligence Planning Systems. AAAI*, 1998.
18. T. Kirk, A. Y. Levy, Y. Sagiv, and D. Srivastava. The information manifold. In *In Working Notes of the AAAI Spring Symposium on Information Gathering from Heterogeneous*, 1995.
19. C.A. Knoblock. Planning, executing, sensing, and replanning for information gathering. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, IJCAI 95*, 1995.
20. A. Y. Levy, A. Rajaraman, and J. J. Ordille. Query-answering algorithms for information agents. In *AAAI/IAAI*, volume 1, pages 40–47, 1996.
21. Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Object fusion in mediator systems. In *VLDB Int. Conf.*, Bombay, India, September 1996.
22. Y. Papakonstantinou, A. Gupta, and L. M. Haas. Capabilities-based query rewriting in mediator systems. *Distributed and Parallel Databases*, 6(1):73–110, 1998.
23. R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt*, pages 484–495. Morgan Kaufmann, 2000.
24. M.T. Roth and P. Scharz. Don't scrap it, wrap it! a wrapper architecture for legacy data sources. In *Proc. of the 23rd Int. Conf. on Very Large Databases*, Athens, Greece, 1997.
25. I. Schmitt and C. Türker. An Incremental Approach to Schema Integration by Refining Extensional Relationships. In G. Gardarin, J. French, N. Pissinou, K. Makki, and L. Bougamin, editors, *Proc. of the 7th ACM CIKM Int. Conf. on Information and Knowledge Management, November 3–7, 1998, Bethesda, Maryland, USA*, pages 322–330, New York, 1998. ACM Press.
26. R. Yerneni, Y. Papakonstantinou, S. Abiteboul, and H. Garcia-Molina. Fusion queries over internet databases. In *Advances in Database Technology - EDBT'98, 6th International Conference on Extending Database Technology*, volume 1377 of *Lecture Notes in Computer Science*, 1998.