D2I

Integrazione, Warehousing e Mining di sorgenti eterogenee

Programma di ricerca (cofinanziato dal MURST, esercizio 2000)

Query Manager functional specifications

D. Beneventano, S. Bergamaschi, F. Mandreoli, M. Marchica

D1.R8 **20 Dicembre 2001**

Sommario

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 report we delineate the functional specifications of a query manager in the context of a data integration, wrapper/mediator based system, the MOMIS system.

The developed techniques rely on the availability of *integration knowledge* whose semantics is expressed in terms of description logics. Integration knowledge includes 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 perform semantic query optimization in a mediator based system as it allows to devise an *optimized query reformulation* method.

Tema	Tema 1: Integrazione di dati provenienti da sorgenti eterogenee
Codice	D1.R8
Data	20 Dicembre 2001
Tipo di prodotto	Rapporto tecnico
Numero di pagine	16
Unità responsabile	MO
Unità coinvolte	MO
Autore da contattare	Federica Mandreoli D.S.I. Università degli studi di Modena e Reggio Emilia Via Vignolese, 905 41100 Modena, Italia mandreoli.federica@unimo.it

Query Manager functional specifications

D. Beneventano, S. Bergamaschi, F. Mandreoli, M. Marchica

20 Dicembre 2001

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 report we delineate the functional specifications of a query manager in the context of a data integration, wrapper/mediator based system, the MOMIS system.

The developed techniques rely on the availability of *integration knowledge* whose semantics is expressed in terms of description logics. Integration knowledge includes 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 perform semantic query optimization in a mediator based system as it allows to devise an *optimized query reformulation* method.

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, 7, 11, 13]. A data integration system, based on conventional wrapper/mediator architectures, 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 [11]) that specify the semantic mapping between the mediated schema and the source schemas. Unlike previous mentioned approaches, in the MOMIS system [3, 4], *mapping descriptions* obtained as a result of the semi-automatic integration process include *extensional intra/interschema knowledge* which represents fundamental knowledge for a correct and complete schema integration [15]. The mediator uses mapping descriptions to reformulate a user query into queries over the source schemata.

In this report we describe the query processing approach adopted in the MOMIS system pointing out the Query Manager modules performing its phases. Our query processing approach exploits the available integration knowledge that is expressed in terms of: local source schemata, a virtual mediated schema and its *mapping descriptions*, *extensional intra/interschema knowledge*.

Extensional knowledge is exploited to perform semantic query optimization in a mediator based system as it allows to devise an *optimized query reformulation* method. In particular, starting from the method developed in [15], we exploit the "base extension" approach and the description logics inference techniques in order to face the reformulation/optimization problem of a mediated query.

The outline of the report is the following. Section 2 provides and overview of the MOMIS system. Section 3 introduces the running example. Section 4 explains the integration knowledge exploited for query processing. Finally, Section 5 shows the Query Manager functional specifications.

2 Overview of the MOMIS system

Like other integration projects [1, 14], 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 [10]; for a detailed description of the MOMIS system see [7, 3] available at http://www.dbgroup.unimo.it/Momis.

2.1 The ODL$_{I^3}$ language

The system supports an object-oriented language, called ODL$_{I^3}$, which is very close to the ODL language [10, 9] and allows a semantically rich representation of source schemas and object patterns associated with information sources to be integrated. ODL$_{I^3}$ is a source independent language used for information extraction to describe heterogeneous schemas of structured and semistructured data sources in a common way. ODL$_{I^3}$ introduces three main extensions with respect to ODL [10]: intentional relationships, that are *terminological relationships* expressing intra and inter-schema knowledge for the source schemas, extensional relationships and integrity constraint rules. The intensional relationships of synonymity (SYN), hypernymy (BT) and positive association (NT) between two classes C_1 and C_2 may be "strengthened" by establishing that they are also *extensional* relationships [8]. Consequently, the following extensional relationships can be defined in ODL$_{I^3}$:

C_1 SYN$_{ext}$ C_2: the instances of C_1 are the same of C_2.

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 DISJ$_{ext}$ C_2: the instances of C_1 are disjoint from the instances of C_2.

Unlike [15], we do not introduce the *overlap relationship* as we assume default overlap relationships among classes if no extensional relationship is specified. Moreover, extensional relationships "constrain" the structure of the two classes C_1 and C_2. If an extensional relationship C_1 NT$_{ext}$ C_2 is issued, we have that:

- strict inheritance between C_1 and C_2 is enforced for the common attributes;
- both C_1 and C_2 may have further attributes as we adopt usual description logics semantics (i.e. open world semantics).

Extensional relationships can be partially automatically extracted and partially explicitly declared by the integration designer.

The integrity constraint rules are introduced in ODL$_{I^3}$ in order to express, in a declarative way, *if then* integrity constraint rules at both intra- and inter-sources level. The antecedent and the consequent of an integrity constraint rule are expressions formulated by the *intersection* operator (**and**), the *union* operator (**or**) and the *negation* operator (**not**).

ODL$_{I^3}$ descriptions are translated into OLCD (*Object Language with Complements allowing Descriptive cycles*) descriptions [6] in order to perform Description Logics inferences that will be

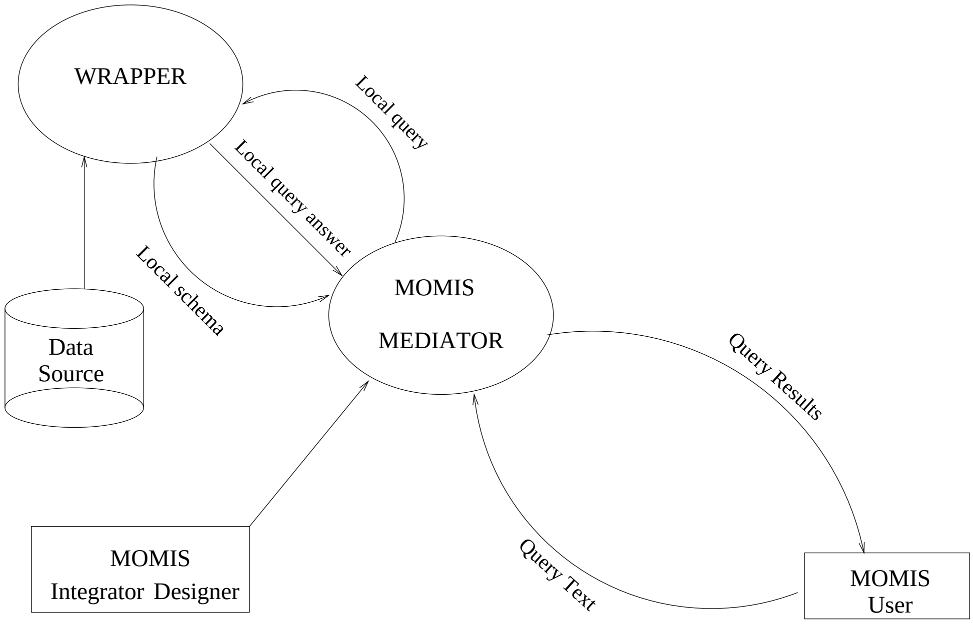

Figure 1: the MOMIS functional elements

useful for semantic integration. Indeed, OLCD supports some interesting reasoning techniques: *subsumption* and *equivalence* detection between types (i.e. "is-a" relationships implied by type descriptions), and *inconsistency* type detection.

In particular, MOMIS translates into OLCD ODL$_{I^3}$ class descriptions, integrity constraint rules and extensional relationships, giving rise to a schema called *inter-sources schema*, formally defined as follows. 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 $L_A \colon \mathbf{L} \to 2^{\mathbf{LA}}$ which associates local class names with attributes. The *inter-sources schema* $\sigma \colon \mathbf{L} \to \mathbf{S}(\mathbf{LA}, \mathbf{L})$ associates local class names to their descriptions; an instance $\mathcal{I}$ of σ defines the instances of the local classes: given a local class L, $\mathcal{I}(L)$ denotes the set of its real-world.

2.2 The functional elements

The functional elements of the MOMIS system are shown in Fig. 1. *Wrappers*, placed over each sources, translate metadata descriptions of the sources into the common ODL$_{I^3}$ representation, translate (reformulate) a global query expressed in the OQL$_{I^3}$[1] query language into queries expressed in the sources languages and export query result data set. The *Mediator* provides an integrated access to the information stored in the sources by interacting with the MOMIS designer in order to derive a Global Schema and with the MOMIS users who formulate queries on the Global Schema.

At a more detailed level, the Mediator is composed of two modules (see Fig. 2): the *Global Schema Builder* (GSB) and the *Query Manager* (QM). The GSB module processes and integrates ODL$_{I^3}$ descriptions received from wrappers to derive the mediated schema by combining reasoning capabilities of Description Logics (ODB-Tools in Figure 2) with affinity-based clustering techniques (Artemis in Figure 2) and by exploiting lexical knowledge (WordNet Artemis in Figure 2 [12]) and validated integration knowledge. A *Common Thesaurus*, which has the role of a shared ontology of the source, is iteratively built by the GSB in a semi-automatic way. The *Common Thesaurus* is a set of intra and inter-schema intensional and extensional relationships,

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

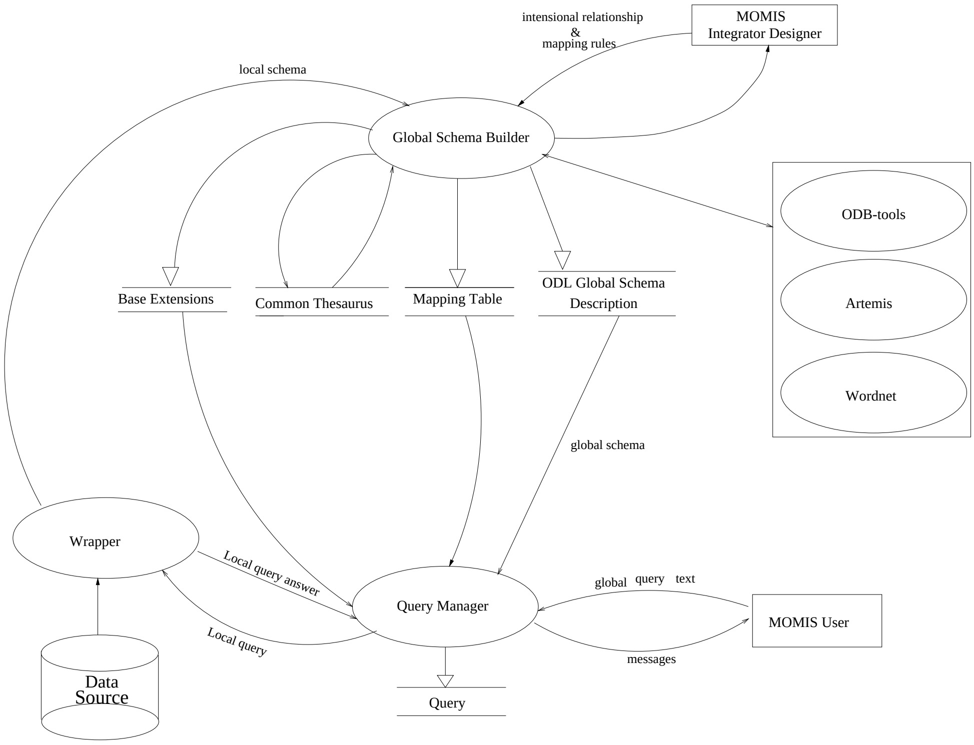

Figure 2: the MOMIS mediator

describing inter-schema knowledge about classes and attributes of sources schemas; it provides a reference for GSB on which to base the identification of classes candidate to integration and subsequent derivation of their global representation. Besides the Global Schema, the GSB modules stores the overall knowledge about the mapping between the Global Schema and the local sources in the Mapping Table and in the Base Extensions (see Sec. 4).

The user application interacts with MOMIS to query the Global Schema by using the OQL_{I^3} language. The query processing phase is performed by the QM which, exploiting the knowledge held by Mapping Table and Base Extensions, 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. More details about the query manager functional specifications, which is the main topic addressed in this report, can be found in Sec. 5.

To achieve the global 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*. 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 (see [5]).

3 Running example

We consider three sources with different data model (see Figure 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)>
...
```

Figure 3: Three heterogeneous University Sources

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`, `e_mail`, 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 enrollment 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.

Examples of extensional relationships in the Common Thesaurus may be 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`

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

An example of integrity constraint rule at intra-source level is the following:

```
rule Rule1 forall X in CS.Professor:
  X.rank = 'full'  then X.belongs_to.description='department';
```

At inter-sources level, both rules between local classes and global/local classes can be expressed.

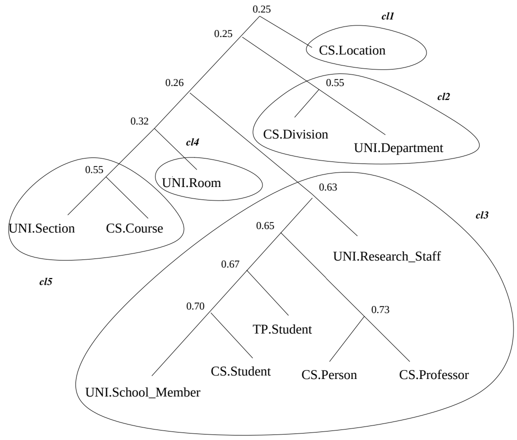

Figure 4: Example of affinity tree and selected Clusters

Examples of inter-sources integrity constraint rules are the following:

```
rule Rule2 forall X in UNI.School_Member:
  X.school = 'cs' then X in TP.Student;

rule Rule3 forall X in CS.Student:
  X in CS.Student  then (X in TP.Student)
  and not (X in UNI.Research_Staff);
```

We suppose that the first three phases of the integration have been performed on our University integration example[2], thus obtaining the mediated global schema with five clusters (cl_1 to cl_5 of Figure 4) and a global class for each cluster.

4 Mapping Descriptions and Base Extensions

In this section, we describe the knowledge exploited by the Query Manager module

4.1 Global Class and Mapping Tables

For each global class outcome of the first three phases of the integration process, a persistent *mapping-table* storing all the mappings is generated; it is a table whose rows represent the set of the local classes belonging 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 5 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`]).

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

[2]For a detailed description of the mappings selection and of the tool SI-Designer which assist the designer in this integration phase see [3].

University_Person	name	dept	e_mail	section	school	year	belong_to	takes	rank	s_code	tax_fee
UNI.Research_Staff	name	dept_code	e_mail	s_code	null	null	null	null	"professor"	null	null
UNI.School_Member	name	null	e_mail	null	school	year	null	null	"student"	null	null
CS.CS_Person	first_name and last_name	null	null	null	"cs"	null	null	null	null	null	null
CS.Student	first_name and last_name	null	e_mail	null	"cs"	year	null	takes	rank	null	null
CS.Professor	first_name and last_name	null	null	null	"cs"	null	"belong_to"	null	rank	null	null
TP.Student	name	null	e_mail	null	school_name	null	null	null	"student"	s_code	tax_fee

Figure 5: Mapping Table of the Global Class University_Person

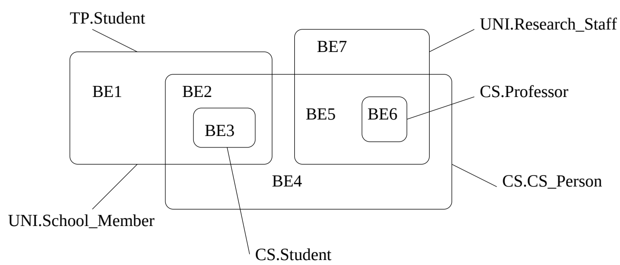

Figure 6: Base extension Set for the global class University_Person

- *constant* : *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* : *ga* is set undefined into the local class L; we denote this case by $MT[L][ga] = \texttt{null}$.

Formally, let σ be an inter-sources schema, $\sigma\colon \mathbf{L} \rightarrow \mathbf{S}(\mathbf{LA}, \mathbf{L})$. In order to define a global class G, we have to consider a subset of $\mathbf{L}$, which are the local classes of the cluster related to G; this subset is denoted with $\mathbf{L}_G$ and with $\mathbf{LA}_G$ we denote the attributes of $\mathbf{L}_G$.

Definition 4.1 (Global Class) *Given a set $\boldsymbol{L}$ of local class names and a set $\boldsymbol{GA}$ of global attributes (denoted by ga_1, ga_2, $\ldots$), a* global class *G is a tuple $G = (\boldsymbol{L}_G, \boldsymbol{GA}, MT)$ where MT, called* mapping table, *is a total function $MT : \boldsymbol{L}_G \times \boldsymbol{GA} \rightarrow 2^{\boldsymbol{LA}_G} \cup \{\texttt{const}\} \cup \{\texttt{null}\}$.*

4.2 Base Extensions

Intuitively, given a global class, a Base Extension collects 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 source objects.

Definition 4.2 (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 $\sigma\colon \boldsymbol{L} \rightarrow \mathbf{S}(\boldsymbol{LA}, \boldsymbol{L})$. 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 $\bigcup_{L \in \boldsymbol{L}_G} \mathcal{I}(L)$.*

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

Figure 7: Tabular representation of the Base Extension Set of University_Person

$$
\begin{aligned}
A(BE1) &= \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax\_fee}\}\\
A(BE2) &= \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax\_fee}\}\\
A(BE3) &= \{\texttt{name}, \texttt{year}, \texttt{school}, \texttt{rank}, \texttt{e\_mail}, \texttt{s\_code}, \texttt{tax\_fee},\texttt{takes}\}\\
A(BE4) &= \{\texttt{name}, \texttt{school}\}\\
A(BE5) &= \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}, \texttt{school}\}\\
A(BE6) &= \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}, \texttt{belong\_to}, \texttt{school}\}\\
A(BE7) &= \{\texttt{name}, \texttt{rank}, \texttt{dept}, \texttt{e\_mail}, \texttt{section}\}
\end{aligned}
$$

Figure 8: Base Extension Attributes

At present, we adopt the algorithm of [15] to determine a base extension set[3]. Figure 6 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 $L \in F(B)$ (e.g. see Figure 7).

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 4.3 (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:*

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

The Base Extension Attributes of our example are shown in Figure 8.

Definition 4.4 (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 *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 The Query Manager

In this section, we describe the approach adopted for query processing. According to other semantic approaches [2], this process consists of two main functionalities (Fig. 9):

Query Plan Definition The objective is to specify an execution plan for the query formulated on the Global Schema. The execution plan contains a basic query for each global class

[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.

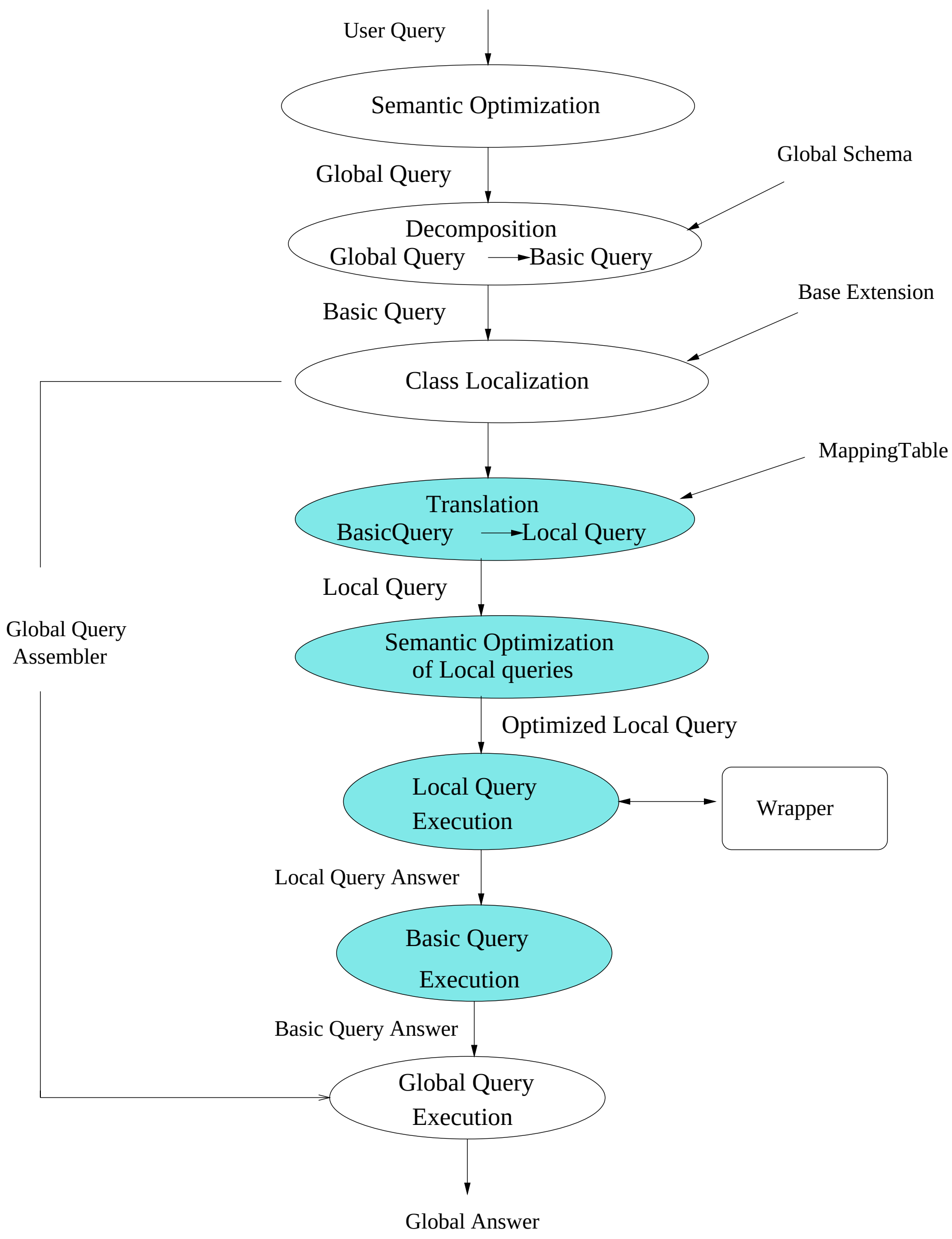

Figure 9: Query processing phases

involved and the operations to be applied to the basic query answers in order to get a global answer. For each basic query, an access plan specifying how to get and to integrate the local source information is generated. The involved query processing phases are:

1. **Semantic optimization of the global query** The global query is replaced by a new one that incorporates any possible restriction which is not present in the original query but is logically implied by the integrity rules at both intra- and inter-sources level.
2. **Decomposition of the global query into basic queries** Usually a global query involves different global classes. Starting from the global query, the decomposition phase consists in generating a basic query for each global class involved. It is performed by extracting from the global query all the predicates addressed to a single global class.
3. **Determination of the Local Class set** For each basic query, the subset of Base Extensions containing the global attributes of the query is computed. The local classes are determined by selecting those included into the above Base Extension subset. In this phase, for each involved base extension, the object fusion rules are defined. It implies to establish a semantic key allowing to join objects that are instances of two different local classes.
4. **Query Reformulation** The query reformulation phase exploits the mapping table information in order to translate each basic query into local queries on the basis of the classes identified in the previous phase.
5. **Semantic optimization of the local queries** The local queries are transformed by considering the integrity constraint rules at both intra- and inter-sources level.

Query Execution By applying the execution plan, the query manager evaluates the global query. The execution plan is applied backwards, that is wrappers execute local queries in order to get pertinent information from the sources, such information are then combined on the basis of the object fusion rules contained in the access plan in order to get a query answer for each global class involved. Then, the Query Manager performs the operations contained in the execution plan in order to get the global answer starting from the answers of basic queries. The involved query processing phases are:

6. **Local query execution** Local queries are sent to the wrappers to be translated and executed by the local sources. Once received the local query answers, the QM translates local values into global values by applying the mapping rules from local sources to global classes contained in the mapping table.
7. **Basic query execution** The objective of this phase is to get an answer for each basic query of the execution plan by means of two steps:
   - for each Base Extension, the object fusion rules are applied to the local query results,
   - the union of the results of the queries identified in the previous phase is performed.
8. **Global query execution** The global answer is obtained by applying the operations specified in the execution plan to the basic query answers.

When a user submits a query to the MOMIS system, the global answer is computed by the QM functional elements shown in Fig. 10 and 11 that perform the above described processing phases: the query plan definition and the query execution, respectively.

We will show the effectiveness of our query processing approach by means of the following (mediated) query:

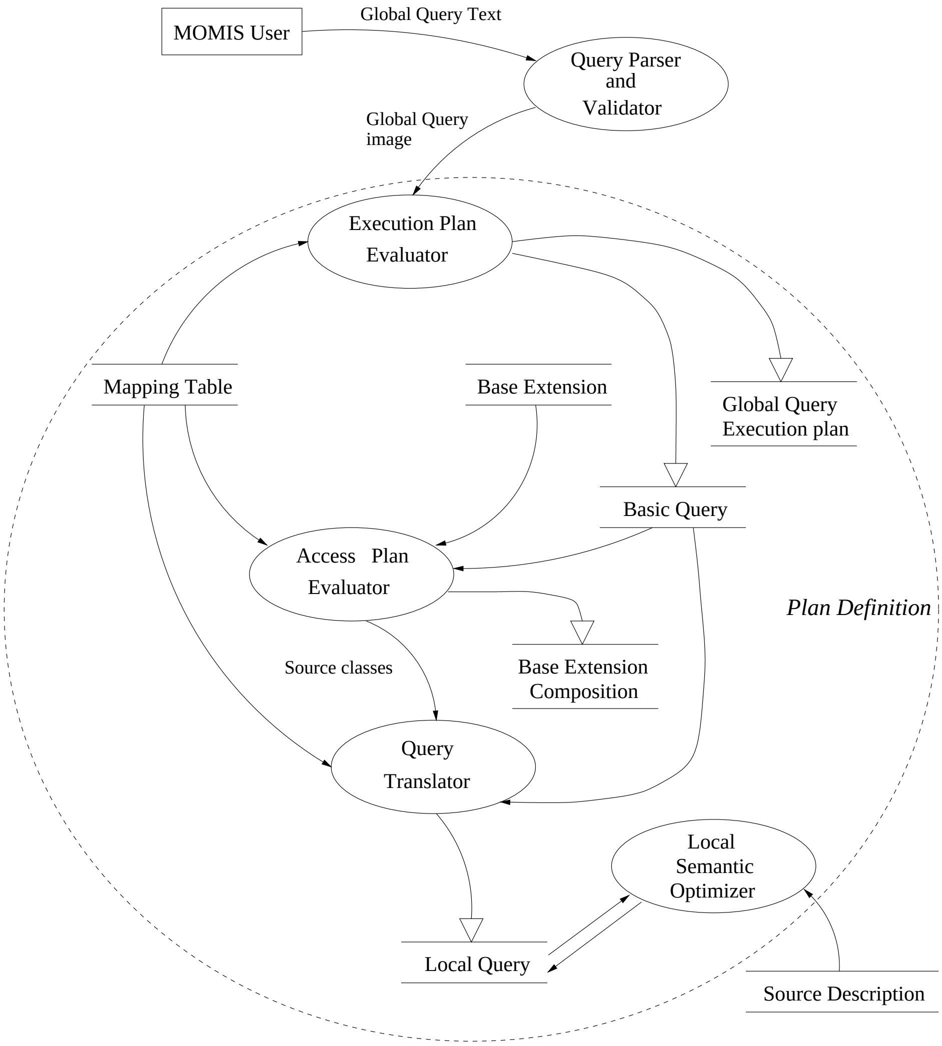

Figure 10: the query plan definition

```
Q: select e_mail
    from University_Person
    where school = 'cs'
    and (s_code = 'a1x' or year = '2001' or tax_fee < 200)
```

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

By considering extensional knowledge, i.e. integrity constraint rules and extensional relationships, and its exploitation by means of base extensions, description logics, types and mapping table, the access to some local class can be avoided, obtaining an effective query optimization, independent from any specific cost model. Let us follow the processing phases of Figure 9 in order to show this result.

1. Semantic optimization of the global query

This optimization can be performed if some integrity constraint rule at inter-sources level are available. In this case, the query can be transformed by considering the integrity rules. For

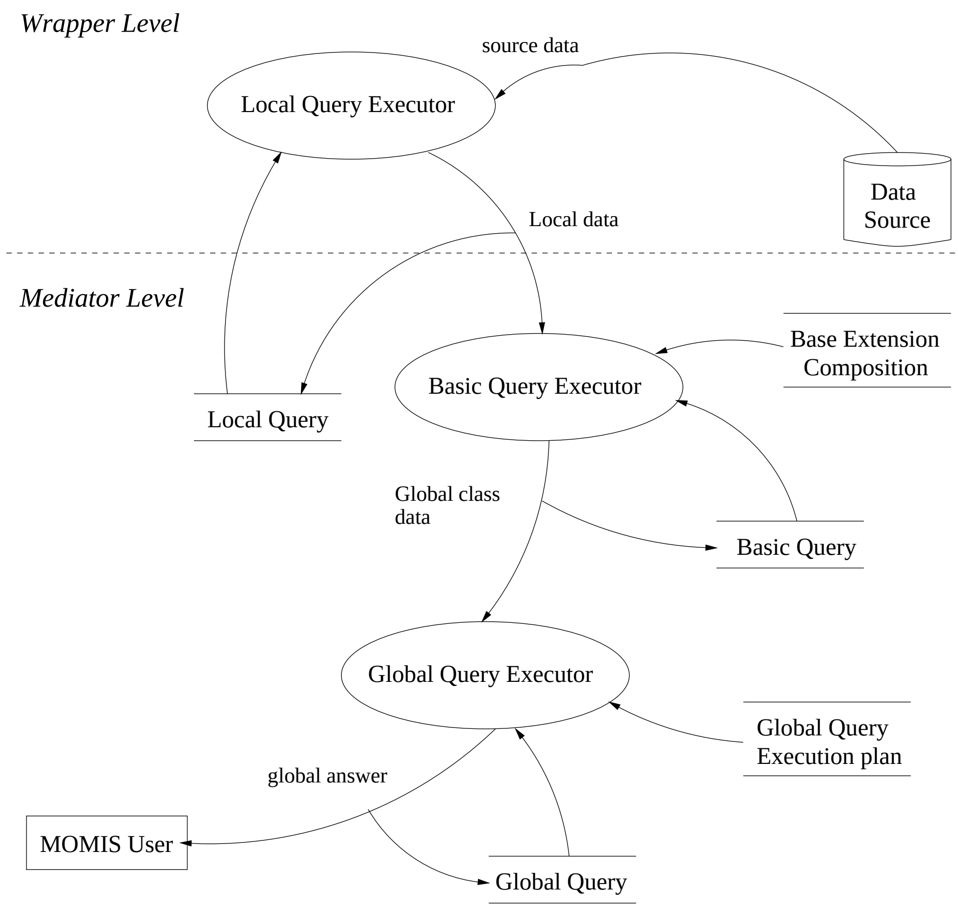

Figure 11: the query execution

example, let us consider the following query:

```
Q1: select name
    from University_Person X
    where X.rank = 'graduate'
    and X.tax_fee < 200
```

If the following integrity rules is available:

```
rule Rule4 forall X in University_Person:
  X.rank = 'graduate' then (X.tax_fee <100)
```

then the factor `(X.tax_fee < 200)` can be eliminated from the query `Q1`. Notice that to perform semantic optimization of the global query we need to use the mapping table. In fact, the global query refers to a global class (global attributes) and intra-sources integrity constraint rules refers to local classes (local attributes).

2. Decomposition of the global query into basic queries

In this case the query involves only one global class, therefore no decomposition is required.

3. Determination of the Local Class Set

In order to determine the local classes for query reformulation we first add to the original query a non-null condition on the select clause attributes; for the query `Q` we have:

```
Q: select e_mail
    from University_Person
    where school = 'cs'
    and (s_code = 'a1x' or year = '2001' or tax_fee < 200)
    and e_mail is not null
```

then we consider the *Disjunctive Normal Form - DNF* of the query condition: $DNF =$ `F1 or F2 or F3` where:

`F1` = `(school='cs') and (s_code='a1x') and (e_mail is not null)`
`F2` = `(school='cs') and (tax_fee<200)and (e_mail is not null)`
`F3` = `(school='cs') and (year='2001')and (e_mail is not null)`

For each factor F of DNF we define the set: $BE(F) = \{B \mid \forall\ ga$ 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:

- $BE(\texttt{F1}) = BE(\texttt{F3}) = \{BE1, BE2, BE3\}$
- $BE(\texttt{F2}) = \emptyset$

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 $DNF =$ `F1 or F3`.

On the basis of this simplification, we obtain the following result: we do not have to access the local classes `UNI.Research_Staff` and `CS.Professor` as the only predicate related to their attributes `(tax_fee<200)` has been eliminated.

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 to the sources).

Local classes are determined by considering the union of all the local classes included in a subset of the previously evaluated base extensions obtained by eliminating the *dominated* ones. Indeed, the concept of domination as introduced in Def. 4.4 can be applied w.r.t. the global attributes of each factor F for which we compute $BE(F)$. With reference to our example, we have $BE(\texttt{F1}) = BE(\texttt{F3}) = \{BE1\}$, 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.

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

4. Query Reformulation

Query reformulation is based on the base extensions selected in the previous phase. First, 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 $BE(F)$.

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

in two steps:

1. `<condition>` 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`$\in BE(F)$); these predicates are rewritten w.r.t. the attributes of local class `L` on the basis of the mapping table. In our example, for factor `F1` and the class `TP.Student` we have

   `<condition> = (school_name='cs')and(s_code='a1x')and(e_mail is not null)`

   whereas for the class `UNI.School_Member` we have `<condition> = (school='cs') and (e_mail is not null)`. Note that, the same predicate may be mapped into semantic homogeneous attributes of more than one class. From a theoretical point of view, such a multiple mapping introduces no problem as semantic homogeneous attributes have been individuated in the integration activity; on the other hand, from an optimized execution point of view only some classes supporting such a predicate could be selected according to cost (either financial or computational) criteria.

2. `<select-list>` is obtained by adding to the query select clause all the join attributes.

In our example, we have two local queries:

```
QF1L1: select e_mail, name
       from TP.Student
       where (school_name='cs')
       and (s_code='a1x')
       and (e_mail is not null)
```

```
QF1L2: select e_mail, name
       from UNI.School_Member
       where (school = 'cs')
       and (e_mail is not null)
```

5. Semantic optimization of the local queries

The local queries are transformed, first, by considering the inter-sources integrity constraint rules and, then by considering the intra-sources ones.

By considering the integrity constraint rules at inter-sources level, the subsumption relationships between local queries are computed: if a local query QL_1 is subsumed by a local query QL_2 and the *select* attributes of QL_1 are a subset of the *select* attributes of the query QL_2 thus it is not necessary to execute the local query QL_1. For example, by applying rule `Rule2`, we obtain that the local query `QF1L2` is subsumed by the local query `QF1L1`. In this case we avoid the `QF1L2` execution as its select clause attributes coincide with those of `QF1L1`.

For each source a local semantic query optimization is performed by considering, for each local query, the intra source integrity constraint rules. Note that this kind of optimization is independent from the physical organization of data of a source as our techniques are independent from any specific cost model.

6. Local query execution

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

7. Basic query execution

The first step of the basic 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:

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

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

The second and last step of the basic query execution is the union of the object fusion queries. In our example, the outcome is the union between `QF1BE1` and `QF3BE1`, that is the analogue of `QF1BE1` for factor `F3` and $BE1$.

8. Global query execution

Since the example query involves only one global class, we skip this step.

6 Conclusions

In this report we described the functional specifications of the Query Manager that is under development in the environment of a data integration, wrapper/mediator based system, MOMIS.

The techniques rely on the availability of *integration knowledge*, whose semantics is given in terms of description logics. Integration knowledge includes: 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, *extensional intra/inter-schema knowledge*. Extensional knowledge is exploited to perform semantic query optimization in a mediator based system as it allows to devise an optimized query reformulation method. The method is based on *base extensions* and description logics inference techniques. In particular, base extensions permit to transform a query on the basis of the extensional relationships between classes, whereas, semantic query optimization techniques are used to transform the query on the basis of the intra- and inter-sources integrity constraint rules.

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 W. Shem. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems (JIIS)*, 6(1):99–130, 1996.

[3] 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.

[4] 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.

[5] D. Beneventano, S. Bergamaschi, F. Guerra, M. Vincini, and M. Vincini. Exploiting extensional knowledge for a mediator based query manager. In *Convegno Nazionale su Sistemi Evoluti per Basi di Dati - SEBD2001, Venezia*, 2001.

[6] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, July/August 1998.

[7] 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.

[8] 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.

[9] R. G. G. Cattell, editor. *The Object Database Standard: ODMG93*. Morgan Kaufmann Publishers, San Mateo, CA, 1994.

[10] R. Hull and R. King et al. Arpa i^3 reference architecture, 1995. Available at http://www.isse.gmu.edu/I3_Arch/index.html.

[11] 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.

[12] A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[13] Y. Papakonstantinou, A. Gupta, and L. M. Haas. Capabilities-based query rewriting in mediator systems. *Distributed and Parallel Databases*, 6(1):73–110, 1998.

[14] 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.

[15] 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.