Dottorato di Ricerca in
Computer Engineering and Science
Scuola di Dottorato in
Information and Communication Technologies
XXI Ciclo

Università degli Studi di Modena e Reggio Emilia
Dipartimento di Ingegneria dell'Informazione

---

tesi per il conseguimento del titolo di dottore di ricerca

Query Management in Data Integration Systems: the MOMIS approach

Tesi di:
Dott. Ing. Mirko Orsini

Relatore:
Prof. Domenico Beneventano

Co-Relatore:
Prof. Isabel F. Cruz

Il Direttore:
Prof. Sonia Bergamaschi

Keywords:
Data Integration
Query Processing
Mediator Systems
Metadata
Access Control

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data Integration (or Information Integration) is the problem of combining related data residing at heterogeneous sources, and providing the user with a unified view of these data [68, 74, 111, 86].

In current real world applications, the problem of designing effective data integration systems is important to enable the collaboration across different domains and the cooperation between different enterprises. The problem of developing effective information integration techniques is becoming even more important with the latest trends in collaborative environments, such as Web 2.0, Web application Mashups, and Cooperative projects on grids, where more and more data have to be shared by different groups and organizations.

Large enterprises spend a great deal of time and money on combining information from different sources into a unified format. Information integration is frequently cited as the biggest and most expensive challenge of the information-technology . As stated in a recent survey [23], data integration is thought to consume about 40% of the enterprises budget [24, 67, 69]. Market-intelligence firm IDC estimates that the market for data integration and access software (which includes the key enabling technology for information integration) was about $2.5 billion in 2007 and is expected to grow to $3.8 billion in 2012, for an average annual growth rate of 8.7% [75].

During the last three decades many systems and applications have been developed to accomplish the goal of an effective information integration. Such applications take data that is stored in several disparated information sources and build a unified integrated view, possibly virtual, containing information from all the sources. The sources can be conventional databases or other types of information, such as collections of Web pages. The three most common approaches for data integration are:

- *Federated databases*. The sources are independent, but one source can call on others to supply information.

- *Warehousing.* Copies of data from several sources are stored in a single database, called a (data) warehouse. Possibly, the data stored at the warehouse is pre-processed in some way before storage; e.g., data may be filtered, and relations may be joined or aggregated. The warehouse is updated periodically, and data may need to be transformed conforming to the schema at the warehouse.

- *Mediator systems.* A mediator is a software component that supports a virtual database, which the user may query as if it were materialized. The mediator stores no data of its own. It translates the user's query into a set of queries to be executed to the sources, fuses together the partial results coming from the sources, and returns the unified result to the user.

This thesis focuses on the Mediator systems approach, and, in particular, on Semantic Data Integration Systems, to perform data integration. In the recent years, the explosive growth of information online has given rise to even more application classes that require semantic integration. Semantic Data Integration Systems (e.g., [16, 89, 88, 76, 84, 82]) exploit the semantic relations between the sources to provide users with a uniform query interface (called *mediated schema* or *global schema*) to a set of data sources, thus freeing them from manually querying each individual source. Unlike other approaches, semantic information integration, try to synthesize a global view of the underlying data sources, as much as possible, automatically.

Integration Systems are usually characterized by a classical wrapper/mediator architecture [114] based on a Global Virtual Schema (Global Virtual View - GVV) and a set of data sources. The data sources store the real data, while the GVV provides a reconciled, integrated, and virtual view of the underlying sources. Modeling the mappings among sources and the GVV is a crucial aspect. Two basic approaches for specifying the mappings in a Data Integration System have been proposed in the literature: Local-As-View (LAV), and Global-As-View (GAV), respectively [68, 110].

In this thesis, we describe the MOMIS (Mediator EnvirOnment for Multiple Information Sources) Data Integration System [9, 16, 15], which performs information extraction and integration from both structured and semi-structured data sources. An object-oriented language, with an underlying Description Logic, called ODL_{I^3} [16] (see appendix A), is introduced for information extraction. Information integration is then performed in a *semi-automatic* way, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and ODL_{I^3} descriptions of source schemas with a combination of clustering techniques and Description Logics. This integration process gives rise to a virtual integrated view (the Global Virtual View - GVV)

of the underlying sources for which mapping rules and integrity constraints are specified to handle heterogeneity. MOMIS is based on a conventional wrapper/mediator architecture, and provides methods and open tools for data management in Internet-based information systems. MOMIS follows a global-as-view (GAV) approach: the obtained global schema is expressed in terms of the data sources.

The main service provided by a Data Integration System is to answer queries posed in terms of the global schema, irrespectively of the method used for the specification of the mapping between the global schema and the sources. To answer a query over the global schema (global query), the query has to be reformulated into an equivalent set of queries expressed on the local schemata (local queries). This query reformulation is performed by considering the mapping between the global schema and the local schemata. In [86], the reformulation problem is analyzed for both the case of local-as-view, and the case of global-as-view mappings.

In this thesis, we refer to the MOMIS System to describe general query processing techniques for Data Integration systems.

There is a strong relationship between query processing in data integration and the problem of query answering with incomplete information. Since data sources are in general autonomous, in real-world applications the problem arises of mutually inconsistent data sources. Since data sources are heterogeneous and autonomous, the main problem to face in Data Integration systems is that of incomplete and inconsistent information. The main advantage of an information integration system is providing users with a *complete* and *concise* view of the underlying data without needing to access the data sources separately. Complete because no object is forgotten in the result; concise because no object is represented twice and the data is presented to the user without conflicts (uncertain or conflicting data values). In [94, 28, 30], the problem of fusing multiple records representing the same real-world object into a single, consistent, and clean representation (Data fusion), is investigated.

Information integration is a very challenging problem, which must be solved at various levels of abstraction [6, 100]. In data integration, different levels of heterogeneities have to be solved: *system-level heterogeneities* (i.e., differences at the hardware level), *structural-level heterogeneities* (i.e., differences at the schema level), *syntactic-level heterogeneities* (i.e., differences in the syntax used to define similar data), *semantic-level heterogeneities* (i.e., differences in the meaning and usage of similar data). In the area of heterogeneous information integration, many projects based on mediator architectures have been developed, trying to solve the heterogeneity problem at different levels. Each mediator system proposed tried to solve as much as

possible the integration problem, focusing on different aspects to provide a (partial) answer to one or many challenges of the problem. The approaches still rely on human interventions, requiring customization for data reconciliation and writing specific not reusable programming code. The specialization of the proposed mediator system makes the comparison among the systems difficult. Therefore, the last Lowell Report [1] has provided the guidelines for the definition of a public benchmark for the information integration problem. The proposal is called THALIA (Test Harness for the Assessment of Legacy information Integration Approaches) [71], and it provides researchers with a collection of downloadable data sources representing University course catalogues, a set of twelve benchmark queries, as well as a scoring function for ranking the performance of an integration system. THALIA benchmark focuses on syntactic and semantic heterogeneities in order to pose the greatest technical challenges to the research community.

In [11], an extension of the MOMIS system is proposed to support syntactic and semantic data heterogeneities by means of declarative Mapping Data Transformation Functions, avoiding the overhead due to write ad-hoc hard-coded transformation functions. Mapping Data Transformation Functions allow MOMIS to deal with all the twelve queries of the THALIA benchmark, without any overhead of new code. The MOMIS system is able to fully satisfy the goal proposed by the benchmark. This is a remarkable result, in fact as far we know no mediator system has provided a complete answer to the benchmark.

Research on data integration has provided languages and systems able to guarantee an integrated representation of a given set of data sources. A significant limitation common to most proposals is that only intensional knowledge is considered, with little or no consideration for extensional knowledge. Ignoring the values assumed by a global attribute may generate meaningless, too selective or empty queries. On the other hand, knowing all the data collected from a global class is infeasible for a user: databases contain large amount of data which a user cannot deal with. A metadata structure derived from an analysis of the attribute extension could be of great help in overcoming such limitation. In [9, 17, 8], a technique for providing metadata related to attribute values is described. Such metadata represent a synthesized and meaningful information emerging from the data. These metadata are called "relevant values", as they provide the users with a synthetic description of the values of the attribute which refer to by representing with a reduced number of values its domain. Such metadata are useful for querying an integrated database, since integration puts together in the same global class a number of local *semantically similar* classes coming from different sources and a set of global attributes which generalize the local classes. Consequently, the

name/description of a global class/global attribute is often generic and this fact could significantly limit the effectiveness of querying.

Another problem that have to be faced in data integration is that of securing the access to the integrated resources. The access to the shared resources needs to be controlled and enforced by global security policies, while local organizations need to maintain the control of their local security policies. Access control levels are heterogeneous across the different organizations, but they have to be merged into a global security model to guarantee the enforcement of security policies at the global level. The following requirements have to satisfied by the security framework: 1)*Autonomy*: the local security policies must not be affected by the security policy of the global level; 2)*Confidentiality*: given a security clearance, if a schema element is not accessible locally before the integration, then it must not be accessible after integration; 3)*Availability*: given a security clearance, if a local schema element is accessible before integration, then it must continue to be accessible after integration. In [45], a privacy-preserving method for classifying the integrated information that depends on the local security policies and preserves the autonomy of the local sources on their security policies is presented.

In collaborative environments, data and resources are shared by different groups and organizations in order to support common tasks. Depending on several factors such as the task, the participants, and data sensitivity, access to these shared resources needs to be controlled and enforced by security policies. A security framework for collaborative environments have to be highly dynamic and flexible to satisfy the following requirements: multiple organizations share resources and tasks; users may not be previously identified and the number of users may be large and unknown in advance; users properties or attributes (e.g., age, status), such as environmental variables (e.g., time, position) and contextual variables (e.g., task, team membership) can change during users sessions; user's roles are not static (e.g., due to change of location); resource availability may change. The role-based access control (RBAC) model is particularly suited to dynamic task-oriented environments due to its flexibility and policy-neutrality [98], which enables it to express a large range of policies. In [43], a security framework for collaborative applications that relies on the role-based access control (RBAC) model, is presented. In the security framework, roles are pre-defined and organized in a hierarchy (partial order). However, users are not previously identified, therefore the actions that they can perform are dynamically determined based on their own attribute values and on the attribute values associated with the resources. By exploring the capabilities of semantic web technologies, and in particular of OWL 1.1, a prototype [44] has been implemented to model both the security framework and the domain of interest and to perform several types of

reasoning.

In this thesis the issue of Query Management in Data Integration Systems is investigated, taking into account several problems that have to be faced during the query processing phase.

The thesis is organized as follows. Chapter 2 introduces the problem of data Integration and the Query Processing techniques for Data Integration Systems referring to the MOMIS Data Integration System. The MOMIS Query Manager prototype is described. In Chapter 3, the THALIA testbed for Data Integration Systems is presented. Experimental results show how the MOMIS Query Manager can deal with all the queries of the benchmark. In Chapter 4, a new kind of metadata that offers a synthesized view of an attributes values, the relevant values, is defined. The effectiveness of such metadata for creating or refining a search query in a knowledge base is demonstrated by means of experimental results. Chapter 5 introduces the security issues that have to be faced in Data integration/interoperation systems. A method to preserve data confidentiality and availability when querying integrated data is presented. A security framework for collaborative applications in which the actions that users can perform are dynamically determined is presented, and the effectiveness of the framework is demonstrated by an implemented prototype. The conclusions are discussed in Chapter 6.

Chapter 2

Query Processing in Data Integration Systems

2.1 Data Integration Systems

Data integration is the problem of combining data residing at different autonomous sources, and providing the user with a unified view of these data. The problem of designing Data Integration Systems is important in current real world applications, and is characterized by a number of issues that are interesting from a theoretical point of view [86]. Integration Systems are usually characterized by a classical wrapper/mediator architecture [114] based on a Global Virtual Schema (Global Virtual View - GVV) and a set of data sources. The data sources store the real data, while the GVV provides a reconciled, integrated, and virtual view of the underlying sources. Modeling the mappings among sources and the GVV is a crucial aspect. Two basic approaches for specifying the mappings in a Data Integration System have been proposed in the literature: Local-As-View (LAV), and Global-As-View (GAV), respectively [68, 110].

The LAV approach is based on the assumption that a global schema representing the conceptualization of a domain exists, and the contents of each local source must be described in terms of the global schema. This assumption holds only in the case that the GVV is stable and well-established in the organization. This constitutes the main limitation of the LAV approach. Another negative issue is the complexity of query processing, which needs reasoning techniques. On the other hand, as a positive aspect, the LAV approach facilitates the extensibility of the system: adding a new source simply means enriching the mapping with a new assertion, without other changes [86].

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

2.2 The MOMIS system for Semantic Data Integration

The Mediator Environment for Multiple Information Sources (MOMIS) is a Data Integration System which performs information extraction and integration from both structured and semi-structured data sources. An object-oriented language, with an underlying Description Logic, called ODL_{I^3} [16] (see appendix A), is introduced for information extraction. Information integration is then performed in a semi-automatic way, by exploiting the knowledge in a Common Thesaurus (defined by the framework) and ODL_{I^3} descriptions of source schemas with a combination of clustering techniques and Description Logics.

This integration process gives rise to a virtual integrated view (the Global Virtual View - GVV) of the underlying sources for which mapping rules and integrity constraints are specified to handle heterogeneity. Given a set of data sources related to a domain it is thus possible to synthesize a GVV that conceptualizes a domain: it might be thought of a basic domain ontology for the integrated sources. MOMIS is based on a conventional wrapper/mediator architecture, and provides methods and open tools for data management in Internet-based information systems. MOMIS follows a global-as-view (GAV) approach: the obtained global schema is expressed in terms of the data sources.

The integrated global schema (GVV) generated by the MOMIS system is composed of a set of global classes that represent the information contained in the underlying sources and the mappings that establish the connections among global classes attributes and the source schemata. The mappings among the local sources and the GVV are defined in a semi-automatic way. We faced the problem of extending the GVV after the insertion of a new source in a semi-automatic way. In [9], we proposed a method to extend a GVV, avoiding starting from scratch the integration process.

2.2.1 Definition of Integration System

Without lack of generality, we refer to the MOMIS System which performs information extraction and integration from both structured and semi-structured data sources. In the MOMIS System, local and global sources are described by an object-oriented language, with an underlying Description Logic, called ODL_{I^3} [16]. ODL_{I^3} is an extended version of the *Object Definition Language*[1].

The main components of a data integration system are the global schema, the local sources, and the mappings.

Definition 1 (MOMIS Integration System) *A MOMIS Integration System* $IS = \langle GVV, \mathcal{N}, \mathcal{M} \rangle$ *is constituted by:*

- *A Global Virtual View (GVV), which is the* global schema *expressed in* ODL_{I^3}
- *A set* $\mathcal{N}$ *of* local sources*; each local source has a* schema *also expressed in* ODL_{I^3}
- *A set* $\mathcal{M}$ *of GAV mapping assertions between* GVV *and* $\mathcal{N}$*, where each assertion associates to an element* G *in* GVV *a query* $q_{\mathcal{N}}$ *over the schemas of a set of local sources in* $\mathcal{N}$*.*

  More precisely, for each global class $G \in GVV$ *we define:*

  1. *a (possibly empty) set of local classes, denoted by* $\mathcal{L}G$*, belonging to the local sources in* $\mathcal{N}$ *.*
  2. *a* mapping query q_G *over the schemas of a set of local classes* $L(G)$ *belonging to* G*.*

Intuitively, the GVV is the intensional representation of the information provided by the Integration System, whereas the mapping specifies how such an intensional representation relates to the local sources managed by the Integration System. The MOMIS GVV, thanks to the ODL_{I^3} language, can have *is-a* relationships and both *key* and *foreign key* constraints. In Appendix A, the syntax of the ODL_{I^3} language is presented.

The semantics of an Integration System is defined in [33, 13].

2.2.2 GVV generation

The GVV generation process generates in a semi-automatic way the set of global classes, and the mappings that establish the connections among global classes attributes and the local source schemata. The GVV generation process is shown in Figure 2.1 and can be outlined as follows:

[1] www.service-architecture.com/database/articles/odmg_3_0.html

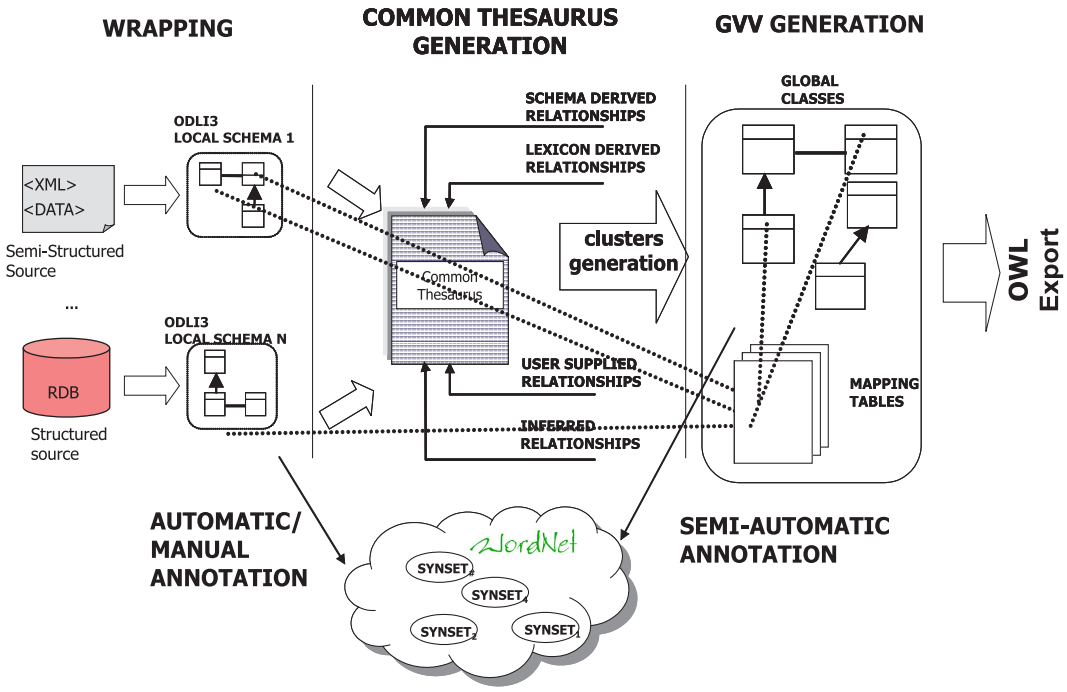


Figure 2.1: Global Virtual View generation process

Extraction of Local Source Schemata

Wrappers acquire schemata of the involved local sources and translated them into the ODL_{I^3} common language. Schema description of structured sources (e.g. relational database and object-oriented database) can be directly translated, while the extraction of schemata from semistructured sources need suitable techniques as described in [2]. To perform information extraction from XML Schema files, like other systems [57], we developed a wrapper that automatically translate the XSD schema into relational structures and import data into a relational database.

Local Source Annotation

Terms denoting schema elements in data sources are semantically annotated according to a common lexical reference, in order to provide a shared meaning to each of them. We choose the WordNet (wordnet.princeton.edu) database as lexical reference. By means of a set of Word Sense Disambiguation algorithms, the lexical annotation is automatically performed [20]. The designer is supported during the annotation phase to select the correct meaning(s) for each term: terms are prepared by applying stop-words and stemming

functionalities to enhance the accuracy result, then algorithms for automatic annotation are applied. The automatic annotation can be performed by combining five different Word Sense Disambiguation algorithms: SD (Structural Disambiguation) [21], WND (WordNet Domains Disambiguation) [21], WordNet first sense heuristic, Gloss Similarity [12] and Iterative Gloss Similarity [12]. Then, the Integration Designer can manually revise the meaning(s) for each annotated term.

Common Thesaurus Generation

Starting from the annotated local schemas, MOMIS extracts relationships describing inter- and intra-schema knowledge about classes and attributes of the source schemata that are inserted in the Common Thesaurus. The Common Thesaurus describes intra and inter-schema knowledge in the form of: synonyms (SYN), broader terms/narrower terms (BT/NT), meronymy/holonymy (RT) relationships. The Common Thesaurus is incrementally built starting from schema-derived relationships, automatically extracted intra-schema relationships from each schema separately. Then, the relationships existing in the WordNet database between the annotated meanings are exploited to generate relationships between the respective elements (classes, attributes), called lexicon-derived relationships. The Integration Designer may add new relationships to capture specific domain knowledge, and finally, by means of a Description Logics reasoner, ODB-Tools [10](which performs equivalence and subsumption computation), infers new relationships and computes the transitive closure of Common Thesaurus relationships.

GVV generation

Exploiting the Common Thesaurus relationships and the local sources schemata, MOMIS generates a GVV consisting of a set of global classes, plus mappings to connect the global attributes of each global class and the local sources' attributes. Going into details, the GVV generation is a process where ODL_{I^3} classes describing the same or semantically related concepts in different sources are identified and clusterized in the same global class by means of the ARTEMIS tool [35]. ARTEMIS determines the degree of matching of two classes, based on their names and their structure, and produces an affinity tree. Clusters for integration are interactively selected from the affinity tree using a non-predefined threshold based mechanism.

In many papers (see [16, 9]) the MOMIS approach for the semi-automatic building of the GVV has been described. The MOMIS approach starts from a set of local sources and giving rise to a Mapping Table (MT) for each

global class G of GVV , whose columns represent the local classes $\mathcal{L}G$ belonging to G and whose rows represent the global attributes of G. An element $MT[GA][L]$ represents the set of local attributes of L which are mapped onto the global attribute GA.

In Table 2.1, an example of Mapping Table is presented. The global class *Course* is mapped to the local class *Cmu.Course* of the *Cmu* source and to the local class *Brown.Course* of the *Brown* source. The *Title* and *Instructor* global attributes are mapped to both the sources, the *Time* global attribute is mapped only to the *Cmu* source, and the *Credits* global attribute is mapped only to the *Brown* source.

Global attributes *Course*	Local attributes *Cmu.Course*	Local attributes *Brown.Course*
Title	CourseTitle	Title
Instructor	Lecturer	Instructor
Time	Time	-
Credits	-	Credits

Table 2.1: Mapping Table example

The Integration Designer may interactively refine and complete the proposed integration results; in particular, the mappings which have been automatically created by the system can be fine-tuned by means of mapping refinements, as will be discussed in Section 2.2.3.

GVV annotation

Exploiting the annotated local schemata and the mappings between local and global schemata, the MOMIS system automatically assigns name and meaning to each element of the global schema. The GVV is automatically annotated, i.e. each of its elements is associated to the broadest meanings extracted from the annotated sources. The annotation of a GVV is a significant result, since these metadata may be exploited for external users and applications interoperability. In Table 2.2, a fragment of annotated GVV is shown as example.

2.2.3 Mapping Refinement: mapping query definition

After the GVV generation process, each global class G is associated to a Mapping Table. Starting from the Mapping Table of G, the integration designer, supported by the Ontology Builder graphical interface, can implicitly define the mapping query q_G associated to the global class G by:

Global attribute	Local attributes	Meaning (from WordNet)
Course	Cmu.Course Brown.Course	Course#1:*education imparted in a series of lessons or meetings*
Instructor	Cmu.Lecturer Brown.Instructor	Instructor#1:*a person whose occupation is teaching*
Title	Cmu.CourseTitle Brown.Title	Title#3:*a general or descriptive heading for a section of written work*

Table 2.2: GVV annotation

1. using and extending the Mapping Table with
   - *Data Conversion Functions* from local to global attributes
   - *Join Condition and Join Tables* among pairs of local classes belonging to G
   - *Resolution Functions* for global attributes to solve data conflicts of local attribute values [7]
2. using and extending the *full outerjoin-merge* operator, proposed in [95] to solve data conflicts of common local attribute values and merge common attributes into one (see section 2.2.4)

By exploiting the enriched MT, refined by the integration designer, the system automatically generates the mapping query q_G associated to the global class G. The mapping query q_G generation and execution process, is fully described in Section 2.3.1.

Data Conversion Functions

The designer can define how local attributes are mapped onto the global attribute GA by means of *Data Conversion Functions*: for each not null element $MT[GA][L]$ we define a *Data Conversion Function*, denoted by $MTF[GA][L]$, which represents how the local attributes of L are mapped into the global attribute GA. $MTF[GA][L]$ is a function that must be *executable/supported* by the local source of the class L. For example, for relational sources, $MTF[GA][L]$ is an SQL value expression; the following defaults hold: if $MT[GA][L] = LA$ then $MTF[GA][L] = LA$ and, if $MT[GA][L]$ contains more than one string attribute, then $MTF[GA][L]$ is the string concatenation.
$\bar{L}$ denotes L transformed by the Data Conversion Function; the schema of $\bar{L}$ is composed of the global attributes GA such that $MT[GA][L]$ is not null.

In the following we show some Data Conversion Functions expressed as SQL-92 like functions. These functions are used in [11] to compare MOMIS in the context of the THALIA Benchmark. In Chapter 3, the details about how the MOMIS system can deal with the THALIA benchmark are presented. The following Data Conversion Functions are used to compare MOMIS in the context of the THALIA benchmark:

- CHAR_LENGTH: return the length of a string
- POSITION: searches a pattern in a string
- SUBSTRING: return a part of a string
- CAST: converts a value from a type to another
- CASE ... WHEN ... THEN: transforms a record on he basis of a specific data value
- RIGHT and LEFT string functions: the first(or the last) n characters of a string

In our system, these function are executed at wrapper level by the right translation of the particular SQL-dialect for the relation DBMSs and are build-in for other wrapper(like XML). Finally, we defined a specific function for datetime type conversion:

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

Let us consider the following example, related to query 7 of the THALIA Benchmark: a global attribute *prerequisite* is mapped into the local attribute *description* of the local class Course, the transformation could infer the value of prerequisite from the information that is attached to the description, in the form of text that follows the "prerequisite" term:

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

Join Conditions

In this section we introduce a simple method (called *Join Conditions*) to identify instances of the same object, we discuss the limitations of this method and we present some examples for extending the *Join Conditions*.

Join Conditions are defined among pairs of local classes belonging to the same global class. Given two local classes $L1$ and $L2$ belonging to G, a Join Condition between $L1$ and $L2$, denoted with $JC(\bar{L}1, \bar{L}2)$, is an expression over $\bar{L}1.A_i$ and $\bar{L}2.A_j$ where A_i (A_j) are global attributes with a not null mapping in $\bar{L}1$ ($\bar{L}2$).
We consider a restricted expression for join conditions. We fix a set of global attributes JA, called Join Attributes, such that: $\forall A_i \in JA, 1 \leq i \leq n$ has a null mapping in $L_1(L_2)$, the join condition is:

$$L_1.A_1 = L_2.A_1$$
$$AND$$
$$\ldots$$
$$L_1.A_n = L_2.A_n$$

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

The details on how Join conditions are exploited at query time to join the relations are presented in Section 2.2.4.

Join conditions are a convenient way to perform object identification when it is possible to assume that error-free and shared object identifiers exist among different sources. Considering the definition of join condition given above, Join conditions may fail to capture many correspondences. The limitation just described is related to the fact that the join condition requires an exact identity among the compared fields.

In order to overcome such limitation, join conditions can be extended adopting less restrictive join operators, as described in [14], or adopting record matching techniques [12].

In [14] a new operator, called containment join condition (CJC), is defined by exploiting the LIKE operator in order to check the containment of the values of two string attributes. For example, by selecting A_1,A_2 as *Join*

Attributes the operator $CJC(L_1, L_2)$ is defined as follows:

$$('\%' + L_1.A_1 +' \%' \ LIKE \ '\%' + L_2.A_1 +' \%' \ AND \ L_1.A_2 = L_2.A_2)$$
$$OR$$
$$(L_1.A_1 = L_2.A_1 \ AND \ '\%' + L_1.A_2 +' \%' \ LIKE \ '\%' + L_2.A_2 +' \%')$$

Otherwise, the designer can define ad hoc functions with more restrictive conditions. For example, $CJC(L_1, L_2)$:

$$('\%' + L_1.A_1 +' \%' \ LIKE \ '\%' + L_2.A_1 +' \%' \ AND \ L_1.A_2 = L_2.A_2)$$

Another more general solution could be to adopt approximate join conditions, and to exploit all the information contained in the tuples. This solution, however, while applicable, is not feasible. Indeed, approximate joins [64] are extremely expensive operations, as approximate comparisons of fields (e.g. strings) are expensive themselves. Tuple comparisons become even more expensive if all (comparable) fields of the tuples to compare have to be taken into account. Furthermore, performing an approximate join over two tables requires a quadratic number of such comparisons.

In [12], a preliminary approach to perform periodical approximate joins activities at sources, using sophisticated record matching techniques, is proposed. The proposed approach builds *Join Tables* that summarize the results of such operations. The join tables are then used at query time to obtain information about semantically equivalent objects.

Approximate join conditions are out of the scope of this thesis and we assume that a shared object identifiers exists among different sources. This assumption is justified as follows. As stated in [94, 28], a data integration process is composed of three steps: 1. Schema matching & mapping. 2. Duplicate detection. 3. Data fusion. The result of the duplicate detection step is the assignment of an object-ID to each representation. Two representations with the same object-ID indicate duplicates. Note that more than two representations can share the same object-ID, thus forming duplicate clusters. In the case presented above, instances of different local classes are identified by the same object-ID. This is the most common case, in which the problem of duplicate detection is considered solved. The focus of this section is on the data fusion problem. The goal of data fusion is to fuse the duplicate representations into a single one, while inconsistencies in the data are resolved.

Resolution Functions

The fusion of data coming from different sources taking into account the problem of inconsistent information among sources is a hot research topic [63, 27,

65, 96, 91]. In the context of MOMIS, the *Resolution Function* proposed in [96] are adopted. A Resolution Function for solving data conflicts may be defined for each global attribute mapping onto local attributes coming from more than one local source. A global attribute with no data conflicts (i.e. the instances of the same real object in different local classes having the same value for this common attribute), is called Homogeneous Attribute. Of course, for homogeneous attributes, resolution functions are not necessary (a global attribute mapped onto only one source is a particular case of an homogeneous attribute).

Definition 2 (Resolution function) *Let D be an attribute domain and $D^+ := D \cup \perp$ where $\perp$ represents the null value. A resolution function f is an associative function $f : D^+ \times D^+ \rightarrow D^+$ with*

$$f(x,y) \quad := \quad \begin{cases} \perp & \text{if } x = \perp \text{and } y = \perp \\ x & \text{if } y = \perp \text{and } x \neq \perp \\ y & \text{if } x = \perp \text{and } y \neq \perp \\ g(x,y) & \text{else} \end{cases}$$

where $g : D \times D \rightarrow D$. Function g is an internal associative resolution function.

We use conflict resolution strategies, based on *Resolution Functions*, as introduced in [96, 29]: for global attributes, mapped in more than one local attributes, the designer defines, in the Mapping Table, *Resolution Functions* to solve data conflicts of local attribute values. For example, if $\bar{L}_1.B$ and $\bar{L}_2.B$ are numerical attribute, we can define $G.B = \texttt{avg}(\bar{L}_1.B, \bar{L}_2.B)$.

If the designer knows that there are no data conflicts for a global attribute mapped onto more than one source (that is, the instances of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an *Homogeneous Attribute*; of course, for homogeneous attributes resolution functions are not necessary. A global attribute mapped into only one local class is a particular case of an homogeneous attribute.

Homogeneous Attributes If the designer knows that there are no data conflicts for a global attribute mapped onto more than one source (that is, the instances of the same real object in different local classes have the same value for this common attribute), he can define this attribute as an *Homogeneous Attribute*; this is the default in our system. Of course, for homogeneous attributes resolution functions are not necessary. A global attribute mapped onto only one source is a particular case of an homogeneous attribute.

2.2.4 Merge operators with Conflict Resolution

Full Outerjoin-merge Operator

In this section we introduce the definition of the *full outerjoin-merge* operator (as given in [95]) and use it to define the mapping query for a global class. The join-merge operator returns tuples for G that are joined from tuples in L_i and L_j, using the join condition between L_i and L_j. Without loss of generality we consider a single join attribute denoted by ID, and then the join condition $JC(L_i, L_j)$ is:

$$L_i.ID = L_j.ID$$

For all attributes exclusively provided by L_i, the values of L_i are used, for all attributes exclusively provided by L_j, the values of L_j are used. For common attributes, the join-merge operator applies the resolution function f defined in Section 2.2.3 to determine the final value. The values of all other attributes of G are padded with null values.

Definition 3 (Join-merge operator) *Let G be a global class with schema $S(G)$ and let L_i and L_j two local classes of G, with schema $S(L_i)$ and $S(L_j)$.*

$$\begin{aligned} L_i \sqcap L_j \;=\; & \{\textit{tuple } t[A] \mid \exists r \in L_i, \exists s \in L_j \textit{ with} \\ & t[ID] = r[ID] = s[ID], \\ & t[A] = s[A], \forall A \in S(L_i) \setminus S(L_j), \\ & t[A] = r[A], \forall A \in S(L_j) \setminus S(L_i), \\ & t[A] = f(r[A], s[A]), \forall A \in S(L_i) \cap S(L_j), A \neq ID, \\ & t[A] = \perp, \forall A \in S(G) \setminus (S(L_i) \cup S(L_j))\} \end{aligned}$$

where f is a resolution function as defined before.

The definition of the left outerjoin-merge operator is based on the outer-union operator $\biguplus$, which performs a union over relations with differing attribute sets [41]. The attribute set of the result is the union of the attribute sets of the two relations. In our case this is the entire attribute set $S(G)$ because the result of a join-merge operation has $S(G)$ as attribute set.

Definition 4 (Left outerjoin-merge operator) *Let G be a global class with schema $S(G)$ and let L_i and L_j two local classes of G, with schema $S(L_i)$ and $S(L_j)$.*

$$L_i \sqcupset L_j \;=\; L_i \sqcap L_j \biguplus \Big(L_i \setminus \Pi_{S(L_i)}(L_i \sqcap L_j) \Big)$$

The left outerjoin-merge corresponds to the classical left outerjoin applying the same restrictions as for the join-merge operator. The left outerjoin-merge $L_i \sqsupset L_j$ guarantees that all tuples from L_i appear in the result. Wherever possible, they are joined with tuples from the other source. If not possible, the missing values are padded with null. Since the right outerjoin-merge operator is basically the same as the left outerjoin-merge, we continue the discussion only with the latter.

Definition 5 (Full outerjoin-merge operator) *Let G be a global class with schema $S(G)$ and let L_i and L_j two local classes of G, with schema $S(L_i)$ and $S(L_j)$.*

$$L_i \sqcup L_j \quad = \quad L_i \sqcap L_j \biguplus \Big(L_i \setminus \Pi_{S(L_i)}(L_i \sqcap L_j) \Big) \biguplus \Big(L_j \setminus \Pi_{S(L_j)}(L_i \sqcap L_j) \Big)$$

The full outerjoin-merge operator guarantees that every tuple from both sources enters the result. Missing values in attributes of tuples that do not have a matching tuple in the other source are padded with null values.

Definition of the mapping query q_G Given a global class G with a set $\mathcal{L}$ of associated local classes, the mapping query q_G associated to G is defined on the basis of the *full outerjoin-merge* operator, using ID as join attribute.

In order to define the mapping query q_G on the basis of the *full outerjoin-merge* operator, we have to consider the general case of three local classes L_1, L_2 and L_3 belonging to G. Given the three local classes L_1, L_2 and L_3, we have to define a Join Condition for each couple of the set: $JC(L_1, L_2)$, $JC(L_2, L_3)$, and $JC(L_1, L_3)$

The set of Join Conditions is *consistent* if, given $JC(L_1, L_2) = \text{true}$, $JC(L_2, L_3) = \text{true}$, it implies $JC(L_1, L_3) = \text{true}$ for each instance of L_1, L_2 and L_3.

We consider the mapping table shown in Table 2.3, in which the mappings between the global class G and the three local classes L_1, L_2 and L_3 belonging to G are presented, to investigate the general case of more than two local classes.

G	L_1	L_2	L_3
ID	**ID**	**ID**	**ID**
A	A	A	A
B	NULL	B	NULL
C	NULL	NULL	C

Table 2.3: Mapping Table example

Setting the global attribute ID as *Join Attribute*, it implies to consider the following set of *join conditions*:

$$JC(L_1, L_2) : L_1.ID = L_2.ID$$
$$JC(L_2, L_3) : L_2.ID = L_3.ID$$
$$JC(L_1, L_3) : L_1.ID = L_3.ID$$

It is simple to verify that the above *join conditions* are *consistent*.

Definition 6 (Associativity of $\sqcup$) *Given a set $\mathcal{L}$ of local classes belonging to G, a mapping query q_G associated to G, defined on the basis of the* full outerjoin-merge *operator $\sqcup$, a set of join conditions defined on the same join attribute ID. We claim that, if:*

1. join conditions *are* consistent

2. resolution functions *are* associative

then the $\sqcup$ operator is associative.

Thus, for a global class G with three local classes $\{L_1, L_2, L_3\}$, the mapping query q_G is one of the following equivalent expressions:

$$q_G = (L_1 \sqcup L_2) \sqcup L_3 \equiv (L_2 \sqcup L_3) \sqcup L_1 \equiv (L_1 \sqcup L_3) \sqcup L_2 \quad (2.1)$$

and will be denoted by $q_G = \bigsqcup\{L_1, L_2, L_3\}$.

From a general point of view, the implementation and computation of q_G requires:

1. the implementation of the *full outerjoin-merge* operator $\sqcup$, which includes resolution functions;

2. the definition of an access plan, i.e. to choose which one of the equivalent expressions in equation 2.1 to compute, that is to choose the join execution order.

In [95, 28], an implementation of the Full Outerjoin-merge Operator is proposed; the underlying engine of the entire process is the *XXL* framework, an extensible library for building database management systems [56].

An important consequence of the associativity of $\sqcup$, is that is possible to compute the *full outerjoin-merge* operator $\sqcup$ by an SQL implementation, as described in the following.

Full Outerjoin-merge Operator : An SQL implementation

In the following, we describe a possible SQL implementation to compute the *full outerjoin-merge* operator $\sqcup$. If the conditions of Definition 6 (*join conditions* are *consistent*, and *resolution functions* are *associative*) are satisfied, then the $\sqcup$ operator is *associative* and it is possible to compute the mapping query q_G by the following two steps:

1. *join sequence* computation
2. *resolution functions* computation

Considering the mapping table shown in Table 2.3, in which the three local classes L_1, L_2 and L_3 belong to G, a *join sequence* FOJ is one of the following equivalent SQL expression:

$$
\begin{aligned}
FOJ_{123} \;=\; & SELECT\ ISNULL(L_1.ID,\ ISNULL(L_2.ID, L_3.ID))\ AS\ ID, \\
& L_1.A,\ L_2.A,\ L_3.A,\ B,\ C \\
& FROM\ (L_1\ FULL\ JOIN\ L_2\ ON\ JC(L_1, L_2)) \\
& FULL\ JOIN\ L_3\ ON\ (JC(L_1, L_3)\ OR\ JC(L_2, L_3))
\end{aligned}
$$

$$
\begin{aligned}
FOJ_{132} \;=\; & SELECT\ ISNULL(L_1.ID,\ ISNULL(L_2.ID, L_3.ID))\ AS\ ID, \\
& L_1.A,\ L_2.A,\ L_3.A,\ B,\ C \\
& FROM\ (L_1\ FULL\ JOIN\ L_3\ ON\ JC(L_1, L_3)) \\
& FULL\ JOIN\ L_2\ ON\ (JC(L_1, L_2)\ OR\ JC(L_2, L_3))
\end{aligned}
$$

$$
\begin{aligned}
FOJ_{231} \;=\; & SELECT\ ISNULL(L_1.ID,\ ISNULL(L_2.ID, L_3.ID))\ AS\ ID, \\
& L_1.A,\ L_2.A,\ L_3.A,\ B,\ C \\
& FROM\ (L_2\ FULL\ JOIN\ L_3\ ON\ JC(L_2, L_3)) \\
& FULL\ JOIN\ L_1\ ON\ (JC(L_1, L_2)\ OR\ JC(L_1, L_3))
\end{aligned}
$$

The hypothesis of *consistent* join conditions the three *join sequences* FOJ_{123}, FOJ_{132}, and FOJ_{231} are equivalent. That is, the result of the *join sequence* computation is independent respect to the order in which the join conditions are computed. Note, the operator $ISNULL()$ is used to return the join attribute ID only once. The MOMIS Query Manager implements the *full outerjoin-merge* computation as shown above, considering *consistent* join conditions.

So far, we considered the case of a unique join attribute ID shared by all the local classes belonging to the global class; join conditions based on

a such global attribute are *consistent*. In the following, we investigate the general case where there is not a unique join attribute ID shared by all the local classes, but each pair of local classes has a different join attribute: in this case join conditions are *not consistent*.

We consider the mapping table shown in Table 2.4, in which each *Join Attribute* is shared only by a couple of local classes.

G	L_1	L_2	L_3
ID1	**ID1**	**ID1**	NULL
ID2	NULL	**ID2**	**ID2**
ID3	**ID3**	NULL	**ID3**
C	C	NULL	NULL

Table 2.4: Mapping Table example

Since *resolution functions* are applied after the *join sequence*, they do not affect the *join sequence* computation between the local classes. Thus, we can consider a mapping in which only *Join Attributes* are shared between more than one local class, as the mapping table example in Table 2.4. The mapping presented in Table 2.4, implies the following set of *join conditions*:

$$JC(L_1, L_2) : L_1.ID1 = L_2.ID1$$
$$JC(L_2, L_3) : L_2.ID2 = L_3.ID2$$
$$JC(L_1, L_3) : L_1.ID3 = L_3.ID3$$

It is simple to verify that the above *join conditions* are *not consistent*. For example, given the L_1,L_2, and L_3 instances presented in Table 2.5, it is simple to verify that $JC(L_1, L_2) = true$, $JC(L_2, L_3) = true$, but $JC(L_1, L_3) = false$.

L_1		
ID1	ID3	C
10	60	a

L_2	
ID1	ID2
10	20

L_3	
ID2	ID3
20	30

Table 2.5: Local classes instances example

Thus, if we consider the *full outerjoin-merge* computation, the result depends on the order in which the join conditions are computed, i.e., the three possible *join sequences* are not equivalent.

The three possible *join sequences* resulting from the mapping table shown in Table 2.4 are the following SQL expression:

$$
\begin{aligned}
FOJ_{123} \;=\; & SELECT\ ISNULL(L_1.ID1, L_2.ID1)\ AS\ ID1, \\
& ISNULL(L_2.ID2, L_3.ID2)\ AS\ ID2, \\
& ISNULL(L_1.ID3, L_3.ID3)\ AS\ ID3,\ C \\
& FROM\ (L_1\ FULL\ JOIN\ L_2\ ON\ (L_1.ID1 = L_2.ID1)) \\
& FULL\ JOIN\ L_3\ ON\ ((L_1.ID3 = L_3.ID3) \\
& OR\ (L_2.ID2 = L_3.ID2))
\end{aligned}
$$

$$
\begin{aligned}
FOJ_{132} \;=\; & SELECT\ ISNULL(L_1.ID1, L_2.ID1)\ AS\ ID1, \\
& ISNULL(L_2.ID2, L_3.ID2)\ AS\ ID2, \\
& ISNULL(L_1.ID3, L_3.ID3)\ AS\ ID3,\ C \\
& FROM\ (L_1\ FULL\ JOIN\ L_3\ ON\ (L_1.ID3 = L_3.ID3)) \\
& FULL\ JOIN\ L_2\ ON\ ((L_1.ID1 = L_2.ID1) \\
& OR\ (L_2.ID2 = L_3.ID2))
\end{aligned}
$$

$$
\begin{aligned}
FOJ_{231} \;=\; & SELECT\ ISNULL(L_1.ID1, L_2.ID1)\ AS\ ID1, \\
& ISNULL(L_2.ID2, L_3.ID2)\ AS\ ID2, \\
& ISNULL(L_1.ID3, L_3.ID3)\ AS\ ID3,\ C \\
& FROM\ (L_2\ FULL\ JOIN\ L_3\ ON\ (L_2.ID2 = L_3.ID2)) \\
& FULL\ JOIN\ L_1\ ON\ ((L_1.ID1 = L_2.ID1) \\
& OR\ (L_1.ID3 = L_3.ID3))
\end{aligned}
$$

Note, the operator $ISNULL()$ is used to return, for each couple of local classes, the join attribute only once: the join is computed on $ID1$ for L_1, L_2; on $ID2$ for L_2, L_3; on $ID3$ for L_1, L_3. Given the instances of Table 2.5, the *join sequences* computation gives the results shown in Table 2.6

FOJ_{123}			
ID1	ID2	ID3	C
10	20	60	a

FOJ_{132}			
ID1	ID2	ID3	C
10	20	60	a
10	20	30	NULL

FOJ_{231}			
ID1	ID2	ID3	C
10	20	60	a

Table 2.6: Join sequences example

The *join sequences* FOJ_{123} and FOJ_{231} are *incomplete*, since they do not return the tuple from L_3. The *join sequence* FOJ_{132} returns the tuple

from L_3, fused to the only possible tuple, the tuple from L_2; since there are not correspondences with tuples from L_1, a $NULL$ value is returned for the attribute C. The *join sequence* FOJ_{132} is *complete*, since all the tuples from the local classes L_1,L_2, and L_3 are returned. The *completeness* of a data fusion operator is defined in [30], even if it is defined considering only two local classes.

The *join sequence* FOJ_{132} represents an operator widely investigated in literature: the *Full Disjunction* operator. The *Full Disjunction* (FD)is informally defined as *"Computing the natural outerjoin of many relations in a way that preserves all possible connection among facts"* [102]. The FD operator is the most suitable operator in the general case of more than two local classes with generic *join conditions*, but the definition and the computation of the *Full Disjunction* are not trivial.

Regarding the definition of the FD operator, two equivalent definitions are given in literature: in [61, 102] the FD operator is defined on the basis of subsumption between tuples; in [42] a definition of FD, based on the concept of tuple set, is given. Concerning the *Full Disjunction* computation, two approaches can be considered: in [102] the FD computation through *join sequences* is investigated: the result is that it is not always possible to compute the FD through a *join sequence*; in [42] an incremental algorithm to compute the ranked FD is proposed.

To present an example of *Full Disjunction*, that is not possible to compute through *join sequences*, we consider the L_1,L_2, and L_3 instances shown in Table 2.7.

L_1		
ID1	ID3	C
a	z	6
b	w	7

L_2	
ID1	ID2
a	4

L_3	
ID2	ID3
4	w

Table 2.7: Local classes instances example

Considering one of the defintion given in [61, 102, 42], the *Full Disjunction* is presented in Table 2.8.

Note, the second tuple of the FD is given by the fusion of the tuple from L_2 together with the tuple from L_3, but no tuples from L_1. Thus, the attribute C has a $NULL$ value.

Given the instances of Table 2.7, the three possible *join sequences* give the results shown in Table 2.9

All the three *join sequences* return the same result, but the second tuple of the FD is not returned by any *join sequence*. This is an example of *Full Disjunction*, that is not possible to compute through *join sequences*.

FD			
ID1	ID3	C	ID2
a	z	6	4
a	w	NULL	4
b	w	7	4

Table 2.8: Local classes instances example

FOJ_{123}			
ID1	ID2	ID3	C
a	4	z	6
b	4	w	7

FOJ_{132}			
ID1	ID2	ID3	C
a	4	z	6
b	4	w	7

FOJ_{231}			
ID1	ID2	ID3	C
a	4	z	6
b	4	w	7

Table 2.9: Join sequences example

It is possible to demonstrate that, in case of *consistent* join conditions, the FD can be computed by any of the possible *join sequence* FOJ. Given the mapping table of Table 2.3 and setting the global attribute ID as *Join Attribute*, it implies to consider the following set of *join conditions*:

$$JC(L_1, L_2) : L_1.ID = L_2.ID$$
$$JC(L_2, L_3) : L_2.ID = L_3.ID$$
$$JC(L_1, L_3) : L_1.ID = L_3.ID$$

In this case, all the three *join sequences* return the same result, that is the *Full Disjunction* result.

2.3 Query Processing in Data Integration Systems

In this section, we refer to the MOMIS System to describe general query processing techniques for Data Integration systems. In section 2.4, a detailed description of the MOMIS Query Manager module is reported.

To answer a query on the GVV (global query) the query must be rewritten as an equivalent set of queries expressed on the local schemata (local queries); the query translation is performed by considering the mapping between the GVV and the local schemata.

Query processing in data integration requires a reformulation step: a query over the global schema has to be reformulated in terms of a set of queries over the local schemata. This reformulation step is strictly dependent on the method used for the specification of the mapping [86]. Query

processing in the GAV approach can be based on an unfolding strategy: given a query q over the global schema, every element of the global schema is substituted with the corresponding query over the local schemata, and the resulting query is then evaluated on data stored by local sources. Query processing in GAV is reduced to unfolding, if there are no integrity constraints on the global schema. If integrity constraints are present, a further reformulation need to be performed [33]. This scenario was considered for MOMIS in [7], where a query posed in terms of the GVV is expanded to take into account the explicit and implicit constraints: all constraints in the GVV are compiled in the expansion, so that the expanded query can be processed by ignoring constraints. The atoms (i.e. sub-queries referring to a single global class) are extracted from the expanded query and then can be processed with an unfolding process strategy. In this thesis, we only consider global schemata without integrity constraints.

The LAV approach generally requires more sophisticated query processing techniques; since in the LAV approach sources are modeled as views over the global schema, the problem of processing a query is called view-based query processing. Briefly, in the case of an LAV mapping, concepts in the local source schemas are defined in terms of the global schema and a query on the global schema is processed by means of an inference mechanism aiming to re-express the atoms of the global schema in terms of atoms at the sources.

MOMIS follows a GAV approach, thus the mapping is expressed by defining, for each global class G, a mapping query q_G over the schemata of a set of local classes $L(G)$ belonging to G. The query translation is performed by means of query unfolding, i.e., by expanding a global query on a global class G of the GVV according to the definition of the mapping query q_G. In the MOMIS System, a global query can be expressed by the OQL_{I^3} language, an extension of the ODMG OQL language.

In Appendix B the OQL_{I^3} syntax is reported. We first describe the query unfolding process performed for a global query expressed over a single global class of the GVV. In section 2.3.2, the query unfolding process for multiple class queries is described. In section 3.4.1, an approach for dealing with queries expressed over multilingual data sources is proposed.

2.3.1 Querying Unfolding

The query unfolding process is performed for each global query Q over a global class G of the GVV. Given the global query Q and the mapping defined on the Mapping Table MT, the query unfolding process generates the set of local queries LQs to be executed on the sources, the mapping query q_G for merging the partial results, and the final query to apply the resolu-

tion functions and residual clauses. We first give an intuitive explanation of the main steps in the query unfolding process. Then, we give the formal description of the query unfolding process.

Querying Unfolding steps

Given a global query Q as following:

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

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

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

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

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

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

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

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

The set of global attributes is transformed in the corresponding set of local attributes on the basis of the Mapping Table MT. The $< condition >$ is computed by performing an atomic constraint mapping: each atomic constraint of $< condition >$ is rewritten into one that is supported by the local

source. The atomic constraint mapping is performed on the basic of the Data conversion Functions and Resolution Functions defined in the Mapping Table. For example, if a numerical global attribute GA is mapped onto the local classes L_1 and L_2, and an average resolution function AVG is defined for GA, the constraint $(GA = value)$ cannot be pushed at the local source, because the AVG function has to be computed at a global level. In this case, the constraint will be mapped as true in both the local sources, and the resolution function will be computed only at a global level. On the other hand, if GA is an *homogeneous attribute* (no resolution function defined), the constraint will be pushed at the local sources.

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

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

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

Step 2. Generation of the mapping query q_G. The LQs partial results will be merged together by means of the *full outerjoin-merge* operator as defined in Section 2.2.4.

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

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

Querying Unfolding (formal)

Let G be a *global class* with schema $S(G) = (GA_1, \ldots, GA_n)$, where GA is a *global attribute*; let $\mathcal{L}$ be the set of local classes belonging to G and let $L \in \mathcal{L}$ be a *local class* with schema $S(L) = (a_1, \ldots, a_p)$, where a is a *local* attribute.

MT is the mapping among the global schema and the local schemata: $MT[GA, L] \subseteq S(L)$ denotes the mapping of information $GA \in S(G)$ into

the local class L, with value equal to *null* if GA is not mapped in L. We introduce no limitations about the global class mapping mechanism. We only require that it is always possible to state if $MT[GA, L]$ is *null* or not.

The mapping satisfies the GAV approach: for each $GA \in S(G)$ it exists at least one $L \in \mathcal{L}$ such that $MT[GA, L] \neq null$. Given the local class $L \in \mathcal{L}$, we define the schema of L w.r.t. G, denoted by $S^G(L)$, as $S^G(L) = \{GA \in S(G) \mid MT[GA, L]$ is not *null*$\}$.

We denote by $\mathcal{D}(A)$ the domain of a local or global attribute and we assume that each a such domain contains the *null value* $\perp$.

We investigate our techniques by adopting as reference data model for schema and query specification the relational model. Thus, each local class L is a relation name with schema $S(L)$ and ℓ denotes a local relation.

For each local class L, we consider an unary operator $[r]_L^G$, that, given a relation ℓ with schema $S(L)$ produces a relation $\ell^G = [r]_L^G$ with schema $S^G(L)$ obtained by

1. (*Schema Translation*) renaming the attributes of r into attributes of G by means of the mapping MT (for example, `firstn` and `lastn` to `Name`);

2. (*Data conversion*) converting the tuples of r into tuple of r^G by suitable functions such as *string concatenation* (for example, `'Rita' + 'Verde'` to `'Rita Verde'`).

A tuple of ℓ^G will be denote with t_L^G. Then, given the local class set $\mathcal{L} = \{L_1, \ldots, L_n\}$, we introduce the notion of *sources database* as $db = \langle \ell_1^G, \ldots, \ell_n^G \rangle$.

As an example, let us consider two relation L_1, L_2 with schema

$S(L_1)$ = (firstn, lastn, year, e_mail)
$S(L_2)$ = (name, e_mail, dept_code, s_code)

and the global class G with the following mapping table:

	Name	**E_mail**	**Section**	**Year**	**Dept**
L_1	firstn $\underline{+}$ lastn	e_mail	null	year	null
L_2	name	e_mail	s_code	null	dept_code

$S(G)$ = (Name, E_mail, Section, Year, Dept)
$S^G(L_1)$ = (Name, E_mail, Year)
$S^G(L_2)$ = (Name, E_mail, Dept, Section)

We assume a global SQL-like query expression over the global schema G having the form

$$\texttt{select } AL \texttt{ from } G \texttt{ where } Q$$

where $AL \subseteq S(G)$ is an attribute list and Q is query condition specified as a boolean expression of positive *atomic predicate* having the form (A *op value*) or (A *op* A'), with $A, A' \in S(G)$.

For example, by considering the following query:

```
select Name from G
where Name like 'P*' and (Year='1' or Dept='Dept1)
```

we have $AL = \{\texttt{Name}\}$ and
$Q =$ `Name like 'P*' and (Year='1' or Dept='Dept1)`.

We consider the mapping query q_G as defined in Section 2.2.4 for *consistent* join conditions and we introduce the following algebraic notation for FOJ: $FOJ(\ell_1^G, \ldots, \ell_n^G)$.
In this way, we have:

$$G = FOJ(\ell_1^G, \ldots, \ell_n^G)$$

Then, considering the algebraic notation for the global query Q:

$$Q = \pi_{AL}(\sigma_Q(g))$$

we can write:

$$\pi_{AL}(\sigma_Q(FOJ(\ell_1^G, \ldots, \ell_n^G))$$

In our context, the query rewriting problem consists into rewrite this expression into an equivalent form:

$$\pi_{AL}(\sigma_{Q_r}(FOJ([lq_1]_{L_1}^G, \ldots, [lq_n]_{L_n}^G)$$

where

- $lq_i = \pi_{AL_i}(\sigma_{LQ_i}(\ell_i))$ is the answer to the local query for the local class L_i
- Q_r is the residual condition.

Since we are not the owner of the data, the local queries are sent and execute on local sources; then, in order to reduce the size of the the local query answer lq_i, it is important: (1) to maximize the selectivity of the local query condition LQ_i and (2) to minimize the cardinality of the local query select-list AL_i. For the query rewriting method shown in the following, both these properties hold.

The steps to compute LQ_i, $1 \leq i \leq n$, and Q_r are the following:

1. **Query normalization**

   The first step is to convert Q into a Disjunctive Normal Form (DNF) query Q^d where the predicates are atomic. The DNF query will be of the form $Q^d = C_1 \vee C_2 \vee \ldots \vee C_m$, where each conjunction term C_i has the form $P_1 \wedge P_2 \wedge \ldots \wedge P_n$, i.e., conjunction of atomic predicates.

   In the example, we have:
   $Q^d =$ `(Name like 'P*' and Year='1') or (Name like 'P*' and Dept='Dept1')`

2. **Local Query condition** LQ_i

   In this step, each atomic predicate P in Q^d is rewritten into one that can be supported by the local class L.

   An atomic predicate P is *searchable* in the local class L if the global attributes used in P are present in the local class L, i.e., the global attributes are mapped into L and the mapping $MT[GA, L] \neq null$.

   We make the hypothesis that each atomic predicate P searchable in L is fully expressible/supported in L, i.e., it exists a query condition Q_L^P expressed w.r.t. the local class schema $S(L)$ such that, for each ℓ, if ℓ' is the relation obtained as answer to the evaluation of Q_L^P on ℓ, then

   $$[\ell']_L^G = P(\ell^G)$$

   An atomic predicate P in Q^d is rewritten into P_L w.r.t. L as follows:

   - if P is *searchable* in L: $P_L = Q_L^P$,
   - if P is not *searchable* in L: $P_L = true$

   In the example, we consider the following predicate rewriting:

	`Name like 'P*'`	`Year='1'`	`Dept='Dept1'`
L1	`lastn like 'P*'`	`year='1'`	`true`
L2	`name like 'P*'`	`true`	`dept_code='Dept1'`

   The computation of the local query condition LQ_i is obtained by rewriting each atomic predicate P in Q^d into P_{L_i} w.r.t. L_i.

   In the example, we have:
   LQ_1 = `(lastn like 'P*' and year='1')` LQ_2 = `(name like 'P*' and dept_code= 'Dept1')`

3. **Residual Condition** Q_r

   The computation of Q_r is performed by the following three steps:

   (a) Transform Q into a Conjunctive Normal Form (CNF) query Q^c, the logical AND of clauses C which are the logical OR of atomic predicate P.

   In the example, we have:
   Q^c = `Name like 'P*' and (Year='1' or Dept='Dept1')`

   (b) Any clause C of Q^c containing not searchable (in one or more local classes) atomic predicates is a *residual clause*;

   In the example, we have:

   - `Dept='Dept1'` is not searchable in L_1 then `(Year='1' or Dept='Dept1')` is a residual condition for L_1
   - `Year='1'` is not searchable in L_2 then `(Year='1' or Dept='Dept1')` is a residual condition for L_2

   (c) Residual Condition Q_r is equal to the logical AND of residual clauses.

   In the example, we have:
   Q_r = `(Year='1' or Dept='Dept1')`

4. **Select list of a local query** LQ

   The select list of the query LQ for the local class L, denoted with LAL, is obtained by considering the union of the following sets of attributes:

   (a) attributes of the global select list, AL

   (b) *Join Attribute Set* for the local class L, $JA(L)$

   (c) attributes in the Residual Condition, A_r

   and by transforming these attributes on the basis of the Mapping Table:

$$LAL = \{A \in S(L) | \exists GA \in (AL \cup JA(L) \cup A_r), A \in MT[GA, L]\}$$

   In the example, we have:
   A_r = { `Year,Dept` }
   AL = { `Name` }
   $JA(L_1)$ = { `Name`}
   $JA(L_2)$ = { `Name`}
   then:
   LAL_1 = {`lastn,lastn,year`}
   LAL_2 = {`name,dept_code`}

In conclusion, we have the following local queries:

```
select lastn,lastn,year
from L1
where (firstn like 'P*' and year='1')

select name,dept_code
from L2
where (name like 'P*' and dept_code= 'Dept1')
```

The local query LQ_i is sent to the source related to the local class L_i; its answer, q_i, is transformed by the operator $[.]_G^{L_i}$ and the result is stored in a temporary table T_i.

Now, we first compute FOJ of the temporary tables T_i, and then, we obtain the global query answer by applying the resolution functions.

2.3.2 Multiple Class Queries

Given the global classes $G_1, G_2, ..., G_n$ we consider a Global Query Q :

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

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

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

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

where

- `Qi` is a Single Class Query:

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

where

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

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

In the second step, each single class query is unfolded w.r.t. local classes by taking into account the mappings $\mathcal{M}$; this step has been discussed in Section 2.3.1.

2.4 The MOMIS Query Manager

The MOMIS *Query Manager* is the software module that is in charge of all the query processing phase. It is composed of several modules to enable a user to compose a query over the Global Virtual View (GVV), automatically generate a query plan for the execution of the global query, execute the set of local queries on the sources, perform the fusion of the local answers into a *consistent* and *concise* unified answer, and present the global answer to the user. The user can be a human user that needs a Graphical User Interface (GUI) to compose a query and to view the results, or a software application that can interact with the *Query Manager* sending queries and receiving answers.

2.4.1 The Query Manager Architecture

The *Query Manager* architecture is organized into several functional blocks that coincide with the implemented software modules. The Query Manager is fully implemented in Java. The software is divided into 10 packages, each implementing a different feature, and composed by more than 100 Java classes (about 32000 code lines). For code re-usage some feature are implemented with cross-package code.

Figure 2.2 shows the *Query Manager* architecture:

- **Graphical User Interface (Query Composer)**: the Query Composer module helps users to compose a query over the global classes of the GVV through a graphical interface presenting the GVV in a tree representation. The composed query can then be executed and the final result is shown in a grid.

- **Unfolder**: receive the global query Q and by means of the mapping generates the Query Plan QP. The Query Plan is composed by the set of local queries LQs to be executed at the sources, the mapping query q_G to merge the partial results by means of the *full outerjoin-merge* operator, and the final query to apply *resolution functions* and *residual clauses*.

- **Join Engine**: receive as input the Query Plan QP and execute the set of queries to obtain the final result. The Join Engine, first execute the set of local queries on the sources simultaneously, then it performs the fusion executing the mapping query on the set of partial results, and, finally, it obtains the global result by applying the final query. Each local query is sent to the respective wrapper for the translation into the specific source query language, so that the query can be executed at the data source. A relational database $QMDB$ gives support to the *Query Manager* for the fusion of partial results, that are stored in a temporary table.

The architecture shown in Figure 2.2 consider the execution of a global query expressed on a single global class. In case the global query that have to be executed by the *Query Manager* is expressed over a set of global classes, i.e. a multiple class query (see Section 2.3.2), the architecture is shown in Figure 2.3:

- **Graphical User Interface (Query Composer)**: multiple class global queries MQ can be composed by users helped by the Query Composer module. More than one global class can be joined just choosing one of the "Referenced Classes" of the currently selected class with no need to specify any join condition between the among classes as it is automatically inserted.

- **Multiple class query Unfolder**: receive the multiple class global queries MQ and considering the join clause expressed in the MQ generates a Query Plan QP. The Query Plan is composed by the set of global queries Qs, each one involving only a single global class, and the join query.

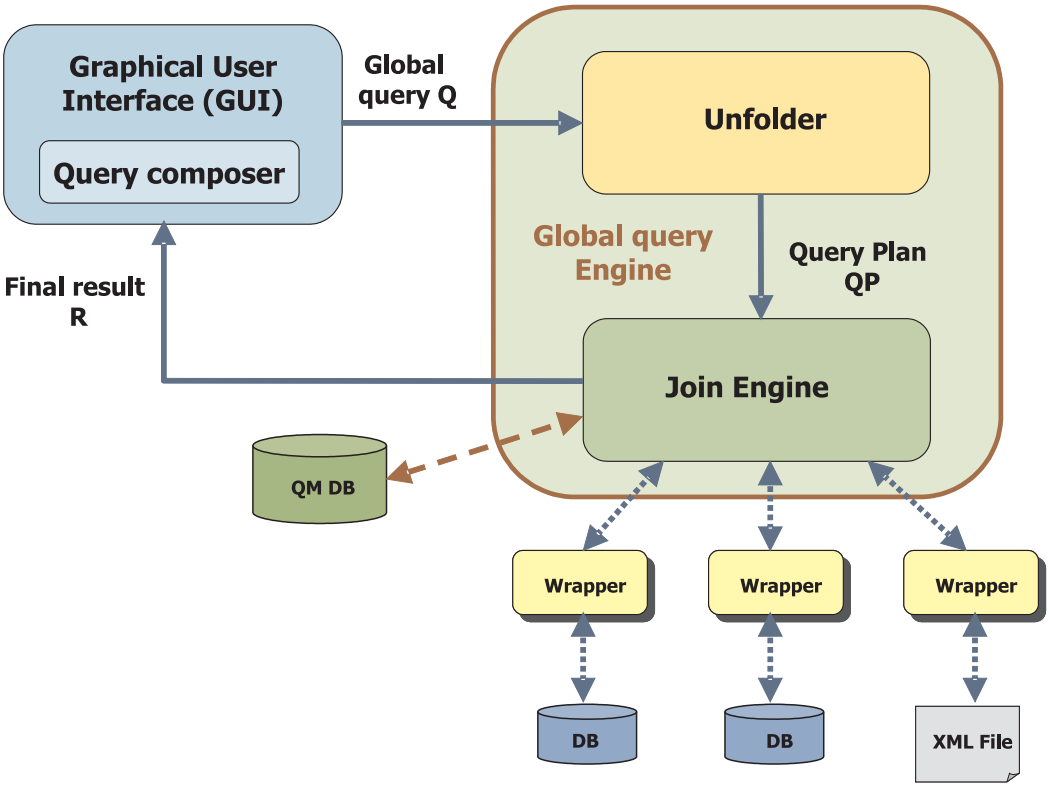


Figure 2.2: The *Query Manager* architecture

- **Multiple class query Join Engine**: receive as input the Query Plan QP and execute the set of global queries to obtain the final result. The Join Engine, first execute the set of global queries simultaneously, then it joins the results by executing the join query to obtain the final result. Each global query is sent to a Global Query Engine to execute the global query over the data sources. As shown in Figure 2.2, each Global Query Engine is composed by an Unfolder module and a Join Engine. A relational database $QMDB$ gives support to the *Query Manager* for the fusion of partial results, that are stored in a temporary table.

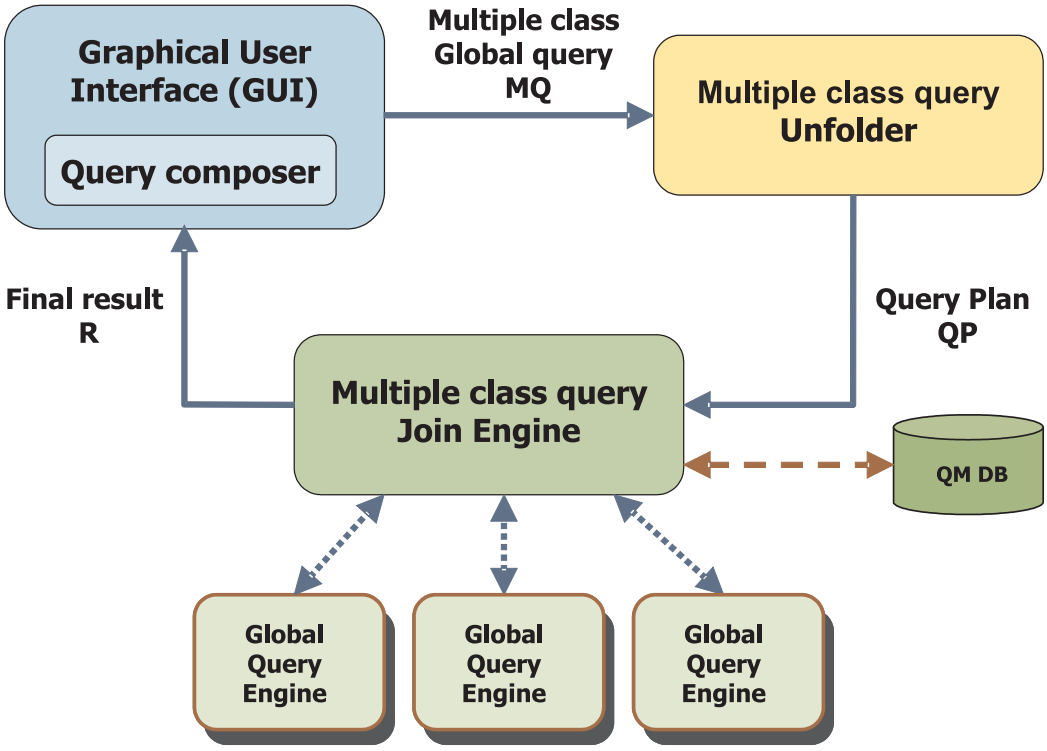


Figure 2.3: The *Query Manager* architecture for Multiple class queries

2.4.2 Query Composer

In order to assure full usability of the system even to users who do not know the OQL language, a graphical user interface has been developed to compose queries over the GVV. This interface, shown in Figure 2.4, presents in a

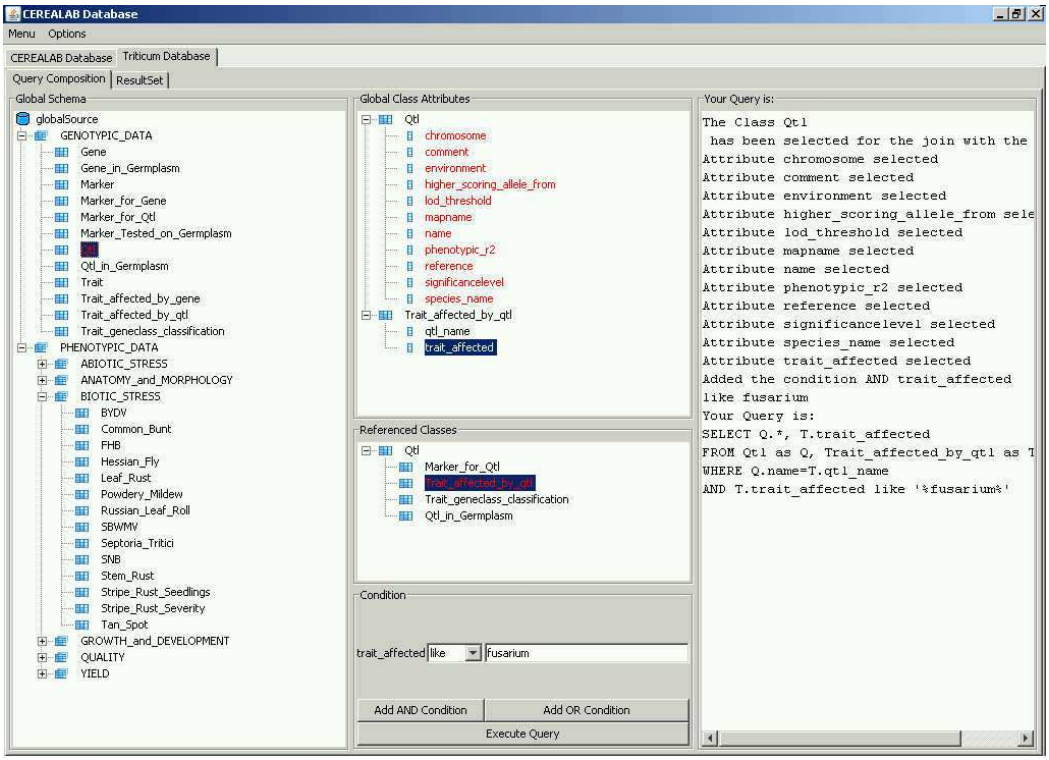


Figure 2.4: The Graphical User Interface for querying the Global Virtual View

tree representation the Global Virtual View. The user can select the global classes to be queried and their attributes are shown in the "Global Class Attributes" panel with a simple click. Then the attributes of interest can be selected, specifying, if necessary, a condition in the "Condition" panel with the usual SQL and logic operators. More than one global class can be joined just choosing one of the "Referenced Classes" of the currently selected class with no need to specify any join condition between the among classes as it is automatically inserted. Selections and conditions specified by the user are then automatically translated into an SQL query and sent to the MOMIS *Query Manager* .

We successfully tested the Graphical User Interface in the context of the CEREALAB project[2], that developed an ontology of molecular and phenotypic cereals data. Figure 2.4 shows an example of the formulation of the query "retrieve all the QTLs that affect the resistance of a plant to the fungus "Fusarium"".

[2]`http://www.cerealab.org`

To do this, the user selects the class `QTL` from the tree on the left side representing the GVV. All the attributes of `QTL` are shown in the tree in the middle panel. Then, the user adds to the selection the Referenced Class `Trait_affected_by_qtl`. All the attributes of this class are then automatically added to the "Global Class Attributes" panel, and the user may select attributes from this global class. To restrict the query only to the Fusarium-related QTLs, it is just needed to add in the "Condition" panel the condition `Trait_affected like fusarium`. Then, clicking the button "Execute Query", the following query is composed, shown in the right side panel and sent to the MOMIS Query Manager:

```
SELECT Q.*, T.trait_affected
FROM Trait_affected_by_qtl as T, Qtl as Q
WHERE T.qtl_name=Q.name AND T.trait_affected like '%fusarium%'
```

The result presented to the user is shown in Figure 2.5

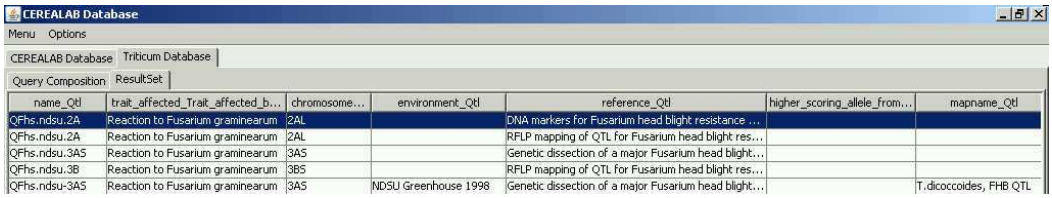


Figure 2.5: The Result Set obtained querying the Global Virtual View

2.4.3 Query Unfolder

The Query Unfolder module receive a global query Q, and automatically generates a Query Plan that is composed by the set of local queries LQs, the mapping query q_G, and the final query.

Query Validation and Normalization

Before the Query Plan generation phase, the global query Q have to be validated to verify both the syntactic and semantic soundness. A parsing module verify the syntactic soundness of the query Q against the OQL syntax (see Appendix B), and store an image of the query in the main memory. Then, the semantic soundness is verified by considering the real membership of the involved attributes to the selected classes, and checking if the query constraints can be expressed on that kind of attributes.

Query Plan Generation

The Query Plan is composed by the set of local queries LQs to be executed at the sources, the mapping query q_G to merge the partial results by means of the *full outerjoin-merge* operator, and the final query to apply *resolution functions* and *residual clauses*. The query translation is performed by means of query unfolding process, described in Section 2.3.1. Local queries are rewritten considering the mapping between the GVV and the local schemata and exploiting the Data Conversion Functions (see Section 2.2.3), which are defined by the user. The mapping query $q_G$is automatically generated, by means of the *full outerjoin-merge* operator (see Section 2.2.4). The final query is automatically generated to apply the *residual clauses* and reconciliation techniques by means of *resolution functions* (see Section 2.2.3, 2.2.4).

In case a multiple class query MQ is received by the Multiple class query Unfolder, the Query Plan is composed by a set of global queries Qs and the join query. The query translation is simpler. The multiple class query MQ is decomposed into a set of global queries Qs, each one involving only a single global class, and a join query is generated for merging the global queries results.

The current implementation of the Query Manager does not take into account the problem of FOJ optimization. In [92], an optimization technique for FOJ queries is proposed. The method applies an optimization substituting the full outer join expression with a left/right outer join or an inner join.

2.4.4 Join Engine

The Join Engine receive as input a Query Plan QP and is in charge of the all query execution process to obtain the final result. A global query execution process is performed by the Join Engine in three different steps:

- **Step 1. Local queries execution:** the local queries are executed on the sources simultaneously. Each local query is expressed in the OQL language, and it is translated into the specific source query language by means of a wrapper. The local queries are executed at the sources and the local answers are materialized in temporary tables into the $QMDB$ relational database.

- **Step 2. Fusion of the local answers:** the local answers stored into the $QMDB$ relational database are fused together executing the mapping query q_G. The fusion is performed by means of the *full outerjoin-merge* operator as described in Section 2.2.4. The FOJ query is ex-

ecuted on the $QMDB$ relational database to fuse together the local answers stored in the temporary tables. The answer of the FOJ query is obtained by storing a view in the $QMDB$ relational database.

- **Step 3. Conflict Resolution and residual clauses:** the conflicting values on the local answers are reconciliated by applying the resolution functions (see Section 2.2.3) choosen by the user. The resolution functions are implemented in the Java code and are applied only in the case of conflicting values stored in the temporary tables. The residual clauses are applied by executing the final query on the fused answer that is obtained by storing a view in the $QMDB$ relational database. The final result R, is then sent to the GUI to be shown to the user.

In case the query that have to be executed is a Muliple class query MQ, the query execution process involves a Multiple class query Join Engine and a set of Global Query Engines. A Muliple class query execution process is performed by the Multiple class query Join Engine in two steps:

- **Step 1. Global queries execution:** the global queries are executed on the Global Query Engines simultaneously. Each global query is sent to a Global Query Engine, composed by an Unfolder module and a Join Engine, that executes the global query performing the steps described above. The answers of the global queries are materialized in temporary tables into the $QMDB$ relational database.

- **Step 2. Join of the global answers:** the global answers sotred in the temporary tables are joined by executing the join query on the $QMDB$ relational database. The answer of the join query is obtained by storing a view in the $QMDB$ relational database.

In Figure 2.6, an example of query execution process for a Multiple class query is presented.

The multiple class query q_0 is decomposed into the two global class queries scq_{G1} and scq_{G2}, and the jon query is $scq_{G1}\ join\ scq_{G2}$. The two single class queries are sent two the Global Query Engines that rewrites them into a set of local queries to be executed at the sources. The global query scq_{G1} is rewritten into the three local queries $L1scq_{G1}$ (source 1), and $L2scq_{G1}$, $L3scq_{G1}$ (source 2). The local answers coming from the queries generated by scq_{G1} are fused together by exectuing the FOJ query $L1scq_{G1}\ fulljoin\ L2scq_{G1}\ fulljoin\ L3scq_{G1}$. The global query scq_{G2} is rewritten into two local queries $L1scq_{G2}$ (source 2), and $L2scq_{G2}$ (source 3). The local answers coming from the queries generated by scq_{G2} are fused together by exectuing the FOJ query $L1scq_{G2}\ fulljoin\ L2scq_{G2}$.

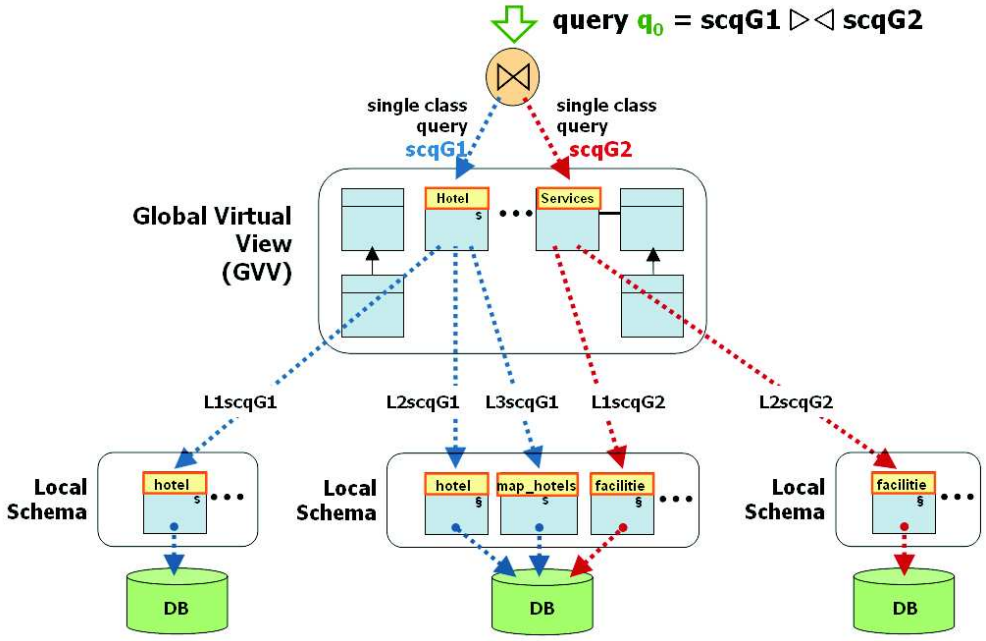


Figure 2.6: The query execution process for a Multiple class query

An example of query fusion process for a Multiple class query is shown in Figure 2.7.

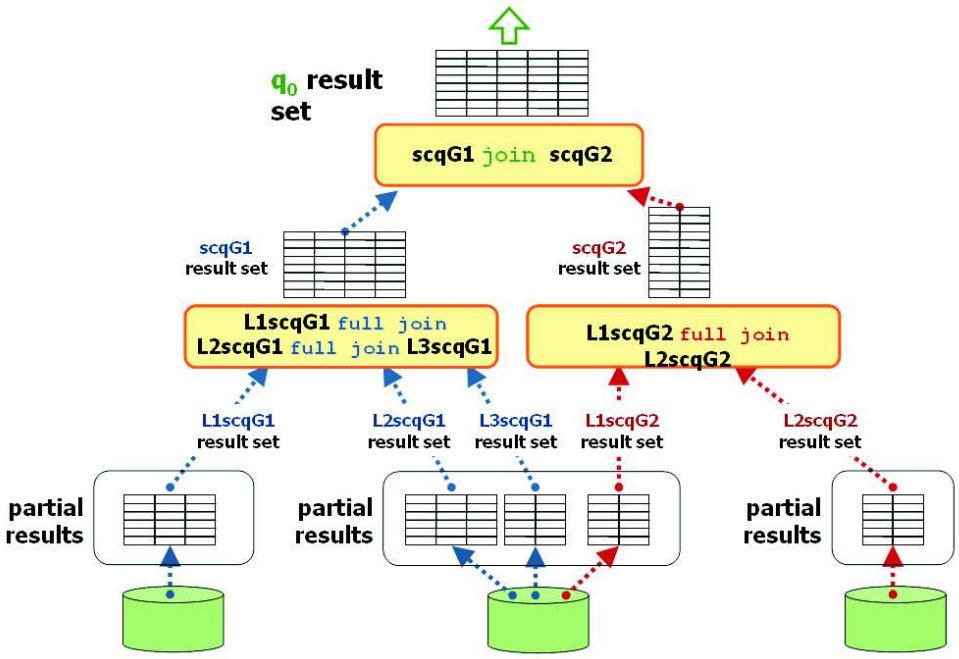


Figure 2.7: The fusion process for a Multiple class query

The examples shown in Figure 2.6 and 2.7 are referred to the following set of queries:

q_0:

```
SELECT H.name, H.address, H.city, H.price, S.facility,
       S.structure_name, S.structure_city
FROM hotels as H, services as S
WHERE H.city = S.structure_city and H.name = S.structure_name
and H.city = 'rimini and H.price > 50 and H.price < 80
and S.facility = 'air conditioning'
order by H.price
```

scq_{G1}:

```
SELECT H.name , H.address , H.city , H.price  FROM hotels as H
WHERE (H.city =  'rimini' ) and (H.price > 50) and (H.price < 80)
```

scq_{G2}:

```
SELECT S.facility , S.structure_name , S.structure_city
FROM services as S
WHERE (S.facility = 'air conditioning')
```

$L1scq_{G1}$:

```
SELECT hotels.name, hotels.address, hotels.city
FROM hotels
WHERE (city) = ('rimini')
```

$L2scq_{G1}$:

```
SELECT maps_hotels.hotels_name2, maps_hotels.hotels_city
FROM maps_hotels
WHERE (hotels_city) = ('rimini')
```

$L3scq_{G1}$:

```
SELECT hotels.name2, hotels.address, hotels.price, hotels.city
FROM hotels
WHERE ((city) = ('rimini') and ((price) > (50) and (price) < (80)))
```

$L1scq_{G2}$:

```
SELECT facilities_hotels.hotel_name2, facilities_hotels.hotels_city,
       facilities_hotels.facility
FROM facilities_hotels
WHERE (facility) = ('air conditioning')
```

$L2scq_{G2}$:

```
SELECT facilities_campings.campings_name,
       facilities_campings.campings_city, facilities_campings.name
FROM facilities_campings
WHERE (name) = ('air conditioning')
```

$L1scq_{G1}$ $fulljoin$ $L2scq_{G1}$ $fulljoin$ $L3scq_{G1}$:

```
saperviaggiare.hotels full outer join venereEn.hotels on (
((venereEn.hotels.name2) = (saperviaggiare.hotels.name)
AND (venereEn.hotels.city) = (saperviaggiare.hotels.city)))
full outer join venereEn.maps_hotels on (
((venereEn.maps_hotels.hotels_name2) = (saperviaggiare.hotels.name)
AND (venereEn.maps_hotels.hotels_city) = (saperviaggiare.hotels.city))
OR ((venereEn.maps_hotels.hotels_name2) = (venereEn.hotels.name2)
AND (venereEn.maps_hotels.hotels_city) = (venereEn.hotels.city)))
```

$L1scq_{G2}$ $fulljoin$ $L2scq_{G2}$**:**

```
guidacampeggi.facilities full outer join venere.facilities on (
(venere.facilities.facility) = (guidacampeggi.facilities.name)
AND (venere.facilities.hotels_city) =
    (guidacampeggi.facilities.campings_city)
AND (venere.facilities.hotel_name2) =
    (guidacampeggi.facilities.campings_name))
```

scq_{G1} $join$ scq_{G2}**:**

```
guidacampeggi.facilities full outer join venere.facilities on (
(venere.facilities.facility) = (guidacampeggi.facilities.name)
AND (venere.facilities.hotels_city) =
    (guidacampeggi.facilities.campings_city)
AND (venere.facilities.hotel_name2) =
    (guidacampeggi.facilities.campings_name))
```

Chapter 3

Getting Through the THALIA Benchmark with MOMIS

During the last decade many data integration systems characterized by a classical wrapper/mediator architecture [114] based on a Global Virtual Schema (Global Virtual View - GVV) have been proposed. The data sources store the real data, while the GVV provides a reconciled, integrated, and virtual view of the data sources. Modelling the mappings among sources and the GVV is a crucial aspect. Two basic approaches for specifying the mappings in a Data Integration System have been proposed in the literature: Local-As-View (LAV), and Global-As-View (GAV), respectively [68, 110]. In Section 2.1 the problem of Data Integration is introduced and the LAV and GAV approaches are discussued. The MOMIS system follows a GAV approach, the MOMIS integration methodology is discussed in Section 2.2.

In the area of heterogeneous information integration, many projects based on mediator architectures have been developed. The mediator-based TSIMMIS project [89] follows a "structural" approach and uses a self-describing model (OEM) to represent heterogeneous data sources and the MSL (Mediator Specification Language) rule to enforce source integration. In TSIMMIS, by means of MSL, arbitrary views (in particular, recursive views) can be defined at the mediator layer. The MOMIS system made a different choice: starting from the semi-automatic generated mappings between global and local attributes stored in the mapping tables, views (global classes) are defined by means of a predefined operator, i.e. the full disjunction, that has been recognized as providing a natural semantics for data merging queries. In particular, in the view definition resolution functions are defined to take into account data conflicts. SIMS [5] proposes the creation of a global schema by exploiting the use of Description Logics (i.e., the LOOM language) for the description of information sources. The use of a global schema allows both

GARLIC and SIMS projects to support every possible user queries on the schema instead of a predefined subset of them. The Information Manifold system [87] provides a source independent and query independent mediator. The input schema of Information Manifold is a set of descriptions of the sources and the integrated schema is mainly defined manually by the designer, while in our approach it is tool-supported. The goal of CLIO [93] is to develop a tool for semi-automatically creating mappings between two data representations (i.e., with user input). First of all, in the CLIO framework the focus is on the schema mapping problem in which a source is mapped onto a different, but fixed, "target" schema, while the focus of our proposal is the semi-automatic generation of a "target" schema, i.e. the Global Virtual View, starting from the sources. Moreover, the semi-automatic tool for creating schema mappings, developed in CLIO, employs a mapping-by-example paradigm that relies on the use of value mappings describing how a value of a target attribute can be created from a set of values of source attributes. Our proposal for creating schema mappings can be considered orthogonal with respect to this paradigm. Indeed, the main techniques of mapping construction rely on the meanings of the class and attribute names selected by the designer in the annotation phase and by considering the semantic relationships between meanings coming from the common lexical ontology. On the other hand, MOMIS and CLIO share a common mapping semantics among a (target) global schema and a set of source schemata expressed by the full-disjunction operator. Infomaster [62] provides integrated access to multiple distributed heterogeneous information sources giving the illusion of a centralized, homogeneous information system. The main difference of this project w.r.t. our approach is the lack of a tool aid-support for the designer in the integration process.

Each mediator system proposed tried to solve as much as possible the integration problem, focusing on different aspects to provide a (partial) answer to one or many challenges of the problem, ranging from system-level heterogeneities, to structural syntax level heterogeneities at the semantic level. The approaches still rely on human interventions, requiring customization for data reconciliation and writing specific not reusable programming code. The specialization of the proposed mediator system makes the comparison among the systems difficult. Therefore, the last Lowell Report [1] has provided the guidelines for the definition of a public benchmark for the information integration problem. The proposal is called THALIA (Test Harness for the Assessment of Legacy information Integration Approaches) [71], and it provides researchers with a collection of downloadable data sources representing University course catalogues, a set of twelve benchmark queries, as well as a scoring function for ranking the performance of an integration sys-

tem. THALIA benchmark focuses on syntactic and semantic heterogeneities in order to pose the greatest technical challenges to the research community.

Starting from our previous experience in the system information integration area, where we developed the MOMIS, a mediator system following a GAV approach [9, 16, 15], we developed an extension of the system [11] devoted to the support of syntactic and semantic data heterogeneities by means of declarative Mapping Data Transformation Functions (MDTFs) that avoids the overhead due to write ad-hoc hard-coded transformation functions.

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

3.1 The THALIA Benchmark

THALIA is a public available testbed and benchmark for information integration systems [71]. It provides over 40 downloadable sources representing University course catalog from computer science around the world. The goal of the benchmark is a systematic classification of the different types of syntactic and semantic heterogeneities that are described by the twelve queries provided.

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

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

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

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

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

- *Complex mapping.* It is the case where related attributes differ by a complex transformation of their values. For example Query 4 contains a number of 'credit hour' in the target schema and a string description of the expected work in the challenge schema.

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

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

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

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

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

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

- *Handling sets.* A single attribute contains a string that describes a set of values in a schema, while the same information is split into single

attributes in an other schema. Query 10 contains a target schema with a single lecturer attribute of course and a challenge schema where a professor is defined for each section of the course.

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

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

3.2 MOMIS Integration Methodology

The MOMIS system implements a semiautomatic methodology that follows the global-as-view (GAV) approach. The MOMIS integration methodology is fully described in Section 2.2, and it consists of the following phases:

- Extraction of Local Source Schemata
- Local Source Annotation
- Common Thesaurus Generation
- GVV generation
- GVV annotation

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

3.2.1 GVV Generation

Extraction of the Local Source Schemata is performed by wrappers, that acquire schemata of the involved local sources and convert them into ODL_{I^3}. Schema description of structured sources (e.g. relational database and object-oriented database) can be directly translated, while the extraction of schemata from semistructured sources need suitable techniques as described in [2].

To perform information extraction from XML Schema files, like other systems [57], we developed a wrapper that automatically translate the XSD schema into relational structures and import data into a relational database. All schemata and data provided by THALIA benchmark (10 schemas and 701 records) are automatically wrapped into a relational database.

The techniques implemented to perform the local sources annotation respect to the WordNet (wordnet.princeton.edu) database lexical reference, allow the MOMIS system to automatically annotate most of the terms in the THALIA's sources. Considering the THALIA schemata, the recall rate of the terms automatically annotated in the sources is 80%, with a precision of 82%.

During the GVV generation phase, the Integration Designer may interactively refine and complete the proposed integration results; in particular, the mappings which have been automatically created by the system can be fine-tuned by means of the MDTFs, as will be discussed in section 3.3. For each GVV involving a THALIA's query, MOMIS automatically detect the right basic relations and attributes mapping. For example, the GVV referred to Query 1 contains the mapping between the *Instructor* and *Lecturer* attributes, i.e. the THALIA foreseen challenge. For Query 12, the system recognizes the mapping between *CourseTitle* and *Title* of the two schemata, while the challenge, i.e. information contained in a unique attribute rather than separate attributes, is obtained by specifying a Data Transformation Function.

The obtained GVV is automatically annotated, i.e. each of its elements is associated to the broadest meanings extracted from the annotated sources. The annotation of a GVV is a significant result, since these metadata may be exploited for external users and applications interoperability. As an example, we report a fragment of annotated GVV (Query 12) in table 3.1:

Global attribute	Local attributes	Meaning (from WordNet)
Course	Cmu.Course Brown.Course	Course#1:*education imparted in a series of lessons or meetings*
Instructor	Cmu.Lecturer Brown.Instructor	Instructor#1:*a person whose occupation is teaching*
Title	Cmu.CourseTitle Brown.Title	Title#3:*a general or descriptive heading for a section of written work*

Table 3.1: GVV annotation

3.3 Mapping Refinement

During the S generation process, the system automatically generates a Mapping Table (MT) for each global class G of the globalschema, whose columns represent the local classes $\mathcal{L}$ belonging to G and whose rows represent the global attributes of G. An element $MT[GA, L]$ represents the set of local attributes of L which are mapped onto the global attribute GA. After this automatic step, the Integration Designer may refine the MT by adding:

- Mapping Data Transformation Functions applied to local attributes
- Join Conditions between pairs of local classes belonging to G
- Resolution Functions for global attributes to solve data conflicts of local attribute values [7].

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

3.3.1 Mapping Data Transformation Functions

The Integration Designer may define, or refine, for each element $MT[GA, L]$, a Mapping Data Transformation Function, denoted by $MDTF[GA, L]$, which represents the mapping of local attributes of L into the global attribute GA. $MDTF[GA, L]$ is a function that has to be executable/supported at the local source by the local source wrapper. In fact, we want to push as much as possible function execution at the sources, as suggested in [36] for constraint mappings.

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

The following SQL-92 like functions are accepted:

- CHAR_LENGTH: return the length of a string
- POSITION: searches a pattern in a string
- SUBSTRING: return a part of a string
- CAST: converts a value from a type to another
- CASE ... WHEN ... THEN: transforms a record on he basis of a specific data value
- RIGHT and LEFT string functions: the first(or the last) n characters of a string

The above functions are executed at wrapper level by the right translation of the particular SQL-dialect for the relation DBMSs and are build-in for other wrapper(like XML). Finally, we added a specific function for datetime type conversion:

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

In the following, we report the mapping refinements applied for Query 4, 7, 12.

Example Query 4: a complex transformation function is required to convert the string that describes the expected scope of the course in ethz into a credit unit. At the ETH's Computer Science site is present the conversion formula $\#KE = \#V + \#U + 1$, that the designer inserts in the mapping table by specifying the following $MDTF$:

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

Example Query 7: the following $MDTF$ infers that the course has a prerequisite course by extracting the text following the 'Prerequisite' term.

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

Example Query 12: the challenge is to extract the correct title, day and time values from the title column in the catalog of the Brown University that contains all the above information in a sring. By using a combination of SUBSTRING and POSITION functions it is possible to obtain what requested.

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

```
SUBSTRING(Title FROM POSITION('hr.' IN Title) + 4 FOR
    POSITION(' ' IN SUBSTRING(Title FROM
        POSITION('hr.' IN Title) + 4 FOR 10)))
MDTF[Time][brown.Course] =
SUBSTRING(Title IN POSITION(' ' FROM SUBSTRING(Title FROM
    POSITION('hr.' IN Title) + 4 FOR 10)) +
        POSITION('hr.' IN Title) + 4 FOR 15)
```

3.3.2 The Mapping Query

MOMIS follows a GAV approach, thus for each global class G, a mapping query q_G over the schemas of a set of local classes $\mathcal{L}$ must be defined. MOMIS automatically generates the mapping query q_G associated to G, by extending the Full Disjunction (FD) operator [25,26]. The mapping query generation process and merge operators are described in Section 2.2.4.

We assume, for the sake of simplicity, that the integration designer be able to define join conditions among local classes. Join Conditions (see section 2.2.3) are defined for each global class. As an example, for Query 5, the designer should define the following join condition:

```
JC(L1, L2) : L1.CourseName = L2. Titel
where L1= cmu.Course and L2= ethz.Unterricht.
```

Resolution Function (see Section 2.2.3) for solving data conflicts may be defined for each global attribute mapping onto local attributes coming from more than one local source. As an example, in Query 1, we defined a precedence function for the global attribute $CourseTitle$:

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

3.4 Query Rewriting with declarative Mapping Data Transformation Functions

To answer a query expressed on the GVV (global query) the query must be rewritten as an equivalent set of queries expressed on the local schemata (local queries); this query translation is performed by considering the mappings among the GVV and the local schemata. In a GAV approach query translation is performed by means of query unfolding, i.e., by expanding a global query on a global class of the GVV according to the definition of the mapping query as defined in Section 2.3.

The query unfolding process is fully described in Section 2.3.1. In this section we present the MOMIS query unfolding method in order to consider Multilingual query conditions and Mapping Data Tranformation functions. A complete example of query rewriting is given in Section 3.5.

3.4.1 Query Unfolding for Multilingual query conditions

In this section, the query unfolding process described in Section 2.3.1 is described in order to consider Multilingual query conditions.

In an information integration scenario, frequently the data are expressed in different language, for example a source contains english data and another in german o italian language.
The MOMIS system provide the possibility to propose a query condition in a specific language, for example in english, and the query rewriting process tries to translate each local source query in a suitable condition.
This operation is performed by a TRANSLATION function, that translate the words from a specific language into the different languages of the local sources. The translation is obtained by exploiting the open dictionary of the Gutenberg Project (www.gutenberg.org) and determine, for each word, the translation in the language of the local source, that is a metadata associated to each source during the integration phase.
More precisely, given a global attribute, a multilingual constraint is:

$$GA\ op\ TRANSLATE(term, Language)$$

The condition of a global query is then a Boolean expression of either atomic and multilingual constraints.
Given two languages, Language1 and Language2, and a term of Language1, we consider a function TRANSLATE(term,Language1,Language2) whose result is a set of terms of Language2:

$$\{term_1, ..., term_n\},\ with\ n\ \geq\ 1.$$

To perform the constraint mapping (see step 1 above) of a multilingual constraint:

$$AG\ LIKE\ term\_1\, OR....OR\, GA\, LIKE\, term\_n$$

where $term_i$, $1 \leq i \geq n$, is a term obtained by TRANSLATION(term,Language1,Language2), and Language2 is the language of the local source L. Then the disjunction of atomic constraints is mapped into the local class L as discussed before.

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

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

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

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

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

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

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

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

3.5 Experimental Results

This section describes the results of MOMIS applied to the THALIA benchmark.

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

Integration Phase

During this phase, for each pair of target and challenge schema related to a single query, the designer semi-automatically creates a specific GVV, i.e. twelve GVVs has been deployed for the benchmark. Since the reference schemas are very simple, the GVVs creation was simple and completely automatic, while the designer had to carefully work on mapping refinements. As an example, we describe the GVV creation related to Query 1. The reference schemas are of Georgia Tech University and of Carnegie Mellon University. Georgia Tech University schema contains only *Course* class with many attributes such as *Title*, *Section*, *Instructor*, *Room*, *Description*, Also Carnegie Mellon University contains only *Course* class with attributes like *CourseTitle*, *Room*, *Lecturer*, *Time*, *Unit*, The Global Class obtained automatically is shown in Figure 3.1, where it is possible to note that the *Lecturer* and *Instructor* attributes of the sources are mapped into a single global attribute.

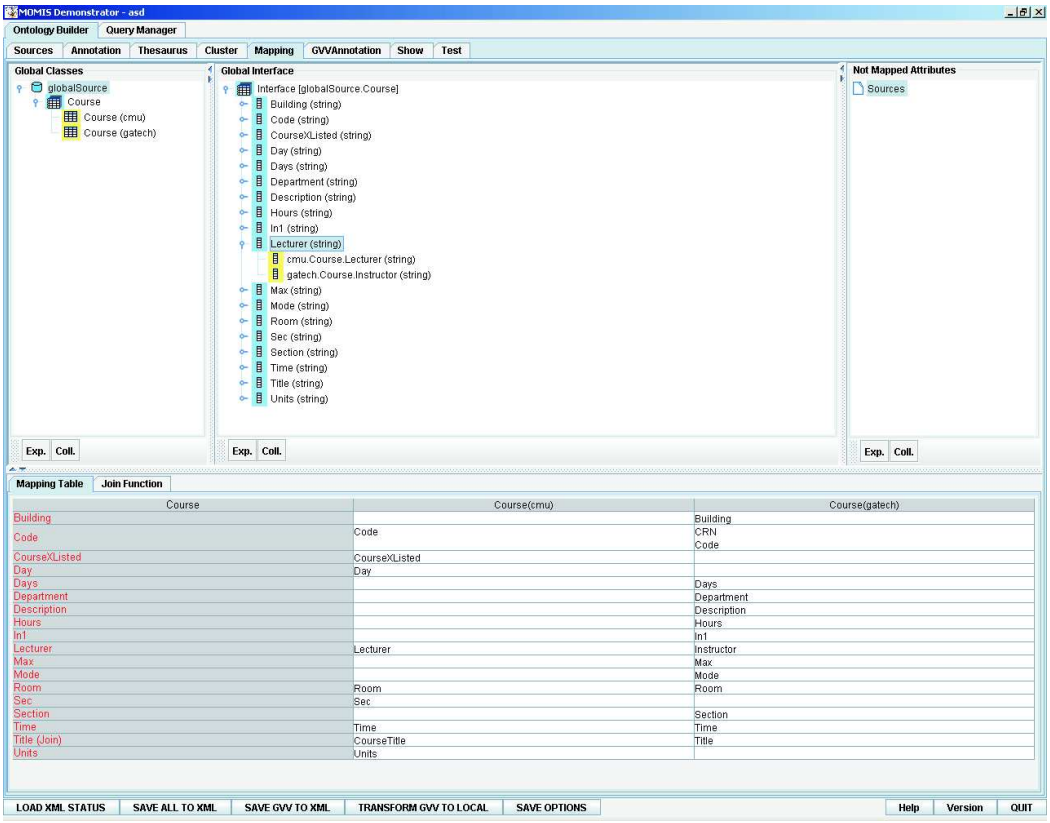


Figure 3.1: MOMIS Schema Mapping example

Mapping Refinement

During the mapping refinement phase, for each query, a specific composition of mapping functions has been inserted to overcome the challenge. Table 3.2 summarize the MDTF functions used for each query challenge.

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

Table 3.2: Thalia Mapping Data Transformation Functions

In the following, the mapping refinements specified for each benchmark query are reported.

Query 1. Mapping between Instructor attribute of Georgia Tech University and Lecturer attribute of Carnegie Mellon University.

```
No mapping refinement required.
```

Query 2. Mapping between Time attribute of Carnegie Mellon University and Times attribute of University of Massachusetts. Mapping refinement:

```
MDTF[Time][umb.Course] = TIME12-24(Times, 1, 12) +
                         SUBSTRING(Times, 6, 1) +
                         TIME12-24(Times, 7, 12)
```

Query 3. Mapping between CourseName attribute of University of Maryland and Title attribute of Brown University. Mapping refinement:

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

Query 4. Mapping between Units attribute of Carnegie Mellon University and Umfang attribute of ETH Zurich. Mapping refinement:

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

Query 5. Mapping between CourseName attribute of University of Maryland and Title attribute of ETH Zurich.

```
No mapping refinement required.
```

Query 6. Mapping between title attribute of University of Toronto and no attribute of Carnegie Mellon University.

Query 7. Mapping between prerequisite attribute of University of Michigan and description attribute in Arizona State University. Mapping refinement:

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

Query 8. Mapping between 'Course restricted' attribute of Georgia Tech University and no attribute of ETH Zurich.

```
No mapping refinement required.
```

Query 9. Mapping between room attribute of Brown University and time attribute of University of Maryland. Mapping refinement:

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

Query 10. Mapping between lecturer attribute of Carnegie Mellon University and title attribute of University of Maryland. Mapping refinement:

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

Query 11. Mapping between lecturer attribute of Carnegie Mellon University and attributes named Fall2003, Winter2004 and Spring2004 of University of California, San Diego. Mapping refinement:

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

Query 12. Mapping between CourseTitle, Day, Time attribute of Carnegie Mellon University and Title attribute of Brown University. Mapping refinement:

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

MDTF[Day][brown.Course] =
SUBSTRING(Title FROM POSITION('hr.' IN Title) + 4 FOR
    POSITION(' ' IN SUBSTRING(Title FROM
    POSITION('hr.' IN Title) + 4 FOR 10)))
```

```
MDTF[Time][brown.Course] =
SUBSTRING(Title IN POSITION(' ' FROM SUBSTRING(Title FROM
    POSITION('hr.' IN Title) + 4 FOR 10)) +
    POSITION('hr.' IN Title) + 4 FOR 15)
```

Query Execution

MOMIS provides a command line interface for querying a GVV and a grid interface to show the query answer.

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

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

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

The portion of the Mapping Table of the class *Course* involved in the query is shown in Table 3.3.

Global attributes *Course*	Local attributes *ethz.Unterricht*	Local attributes *Cmu.Course*
Title	Titel	CourseTitle
Units	MDTF[Unit][ethz.Unterricht]	Units

Table 3.3: Mapping Table example

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

Local Query 1 (LQ1) - Source *cmu*:

```
SELECT Course.CourseTitle, Course.Units
FROM Course
WHERE (CourseTitle) like ('%Database%')
AND Units > 10
```

Local Query 2 (LQ2) - Source *ethz*:

```
SELECT Unterricht.Titel,
  (CAST(SUBSTRING(Umfang, CHARINDEX('V', Umfang)-1,1) AS int) +
  CAST(SUBSTRING(Umfang, CHARINDEX('U', Umfang)-1,1) AS int)+ 1)
  AS Umfang
FROM Unterricht
WHERE ((Titel) like ('%Datenbank%') or
      (Titel) like ('%Datei%') or
      (Titel) like ('%Datenbasis%')) AND
      (CAST(SUBSTRING(Umfang, CHARINDEX('V', Umfang)-1,1) AS int) +
      CAST(SUBSTRING(Umfang, CHARINDEX('U', Umfang)-1,1) AS int)
      + 1) > 10
```

The above local queries are executed at the local sources and then the Full Disjunction of their results is computed as follows:

Full disjunction computation (FD):

```
SELECT LQ1.CourseTitle AS Title_1, LQ2.Titel AS Title_2,
       LQ1.Units AS Units_1, LQ2.Umfang AS Units_2
FROM LQ1 FULL OUTER JOIN LQ2 ON (LQ1.CourseTitle = LQ2.Titel)
```

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

```
SELECT resolution(Title_1, Title_2) AS Title,
         resolution(Units_1, Units_2) AS Units
FROM FD
```

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

3.5.1 Data Integration Systems Comparison

Three different integration systems have reported the THALIA benchmark results: Cohera, Integration Wizard (IWIZ) [71] and a 'Keyword Join' system [118].

In Table 3.4 a comparison of that systems and the MOMIS system is presented, with the specification of extra effort for query answer. The comparison specifies when a system is able to solve a query or not, and what is the extra effort to solve the query: SMALL/MODERATE stands for small/moderate amount of code needed to solve the query.

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

Table 3.4: Data Integration Systems comparison

The Cohera and IWIZ systems can solve 9 queries, some of these by adding a significant amount of code, while the system presented in [118] can deal with 5 queries easily, and other 2 queries adding some metadata, without any custom code.

In the MOMIS system, by means of declarative functions, it is very easy to deal with all the 12 queries. This is a remarkable result, in fact, as far as we know, no system has provided a complete answer to the benchmark. The MOMIS system can solve 4 queries automatically and the other queries refining the mapping by means of the Mapping Data Transformation Functions. We estimate that an integration designer might define the mapping refinements needed to solve all the benchmark queries within 4-6 hours.

Chapter 4

Relevant Values: a new type of metadata for querying Data Integration Systems

Integration of data from multiple sources is one of the main issues facing the database and artificial intelligence research communities. A common approach for integrating information sources is to build a mediated schema as a synthesis of them. By managing all the collected data in a common way, a mediated schema allows the user to pose a query according to a global perception of the handled information. A query over the mediated schema is translated into a set of sub-queries for the involved sources by means of automatic unfolding-rewriting operations taking into account the mediated and the sources schemata. Results from sub-queries are finally unified by data reconciliation techniques (see [86, 7], Section 2.2.4 for an overview).

Research on data integration has provided languages and systems able to guarantee an integrated representation of a given set of data sources. A significant limitation common to most proposals is that only intensional knowledge is considered, with little or no consideration for extensional knowledge.

In this Chapter, we describe a technique for providing metadata related to attribute values. Such metadata represent a synthesized and meaningful information emerging from the data. We call these metadata "relevant values" as they provide the users with a synthetic description of the values of the attribute which refer to by representing with a reduced number of values its domain. We claim that such metadata are useful for querying an integrated database, since integration puts together in the same global class a number of local *semantically similar* classes coming from different sources and a set of global attributes which generalize the local classes. Consequently, the name/description of a global class/global attribute is often generic and this

fact could significantly limit the effectiveness of querying. Let us suppose, for instance, that the user has a good knowledge of a single source, say "S", and that she/he is interested in items whose global attribute "A" contains the word "x", as of terminology of source "S". The user could completely miss the fact that in source "T" the word "y" refers to a very similar concept, and therefore a query with target "x" would return only a partial result, w.r.t. the contents of the global class. Moreover, ignoring the values assumed by a global attribute may generate meaningless, too selective or empty queries. On the other hand, knowing all the data collected from a global class is infeasible for a user: databases contain large amount of data which a user cannot deal with. A metadata structure derived from an analysis of the attribute extension could be of great help in overcoming such limitation.

This work is done in the context of the MOMIS system [16]. The MOMIS integration methodology is presented in Section 2.2. The MOMIS integration process gives rise to a Global Virtual View (GVV) in the form of Global Classes and global attributes of the a set of data sources. In [9], we proposed a partial solution to the semantic enrichment of a GVV by providing a semantic annotation of all the Global Classes of the GVV with respect to the WordNet lexical database[1], and thus providing each term with a well-understood meaning. Relevant Values will semantically enrich a GVV, since they provide semantic information about the data sources the GVV refers to. Moreover, in [8] a first heuristic for calculating relevant values was described.

In [19, 17], we improved the approach proposed in [22], by providing a flexible parametric technique to deal with string attributes. It is not a severe limitation, as: (1) data coming from web-site wrappers are generally represented as strings; (2) several techniques have been developed in literature for clustering numeric values where it is easy to define element orderings (see [78] for a survey). The method was implemented in a prototype called *RELEVANT* (RELEvant VAlue geNeraTor) described in section 4.3. In [18], $RELEVANT^{News}$, a web feed reader with advanced features that couples the capabilities of *RELEVANT* and of a feed reader, is described.

The Chapter is organized as follows. Section 4.1 describes a real reference scenario, then, in section 4.2, we describe our technique to elicit relevant values for a selected attribute. Section 4.3 provides the description of the *RELEVANT* prototype together with a more detailed description of our technique. Section 4.4 gives some experimental results by comparing the relevant values obtained for a selected domain w.r.t. reference clustering provided by a domain expert. A running example, extracted from the domain described in section 4.4, is used through the paper to give an intuition of the proposed

[1] http://wordnet.princeton.edu/

technique. Section 4.5 describes how relevant values may be exploited for querying data sources. In Section 4.6 the *RELEVANT*$^{\text{News}}$ system is described. *RELEVANT*$^{\text{News}}$ is a web feed reader with advanced features, since it couples the capabilities of *RELEVANT* and of a feed reader. Section 4.7 discusses some related work and, finally, Section 4.8 sketches out some conclusions and future works.

4.1 A real reference scenario

Throughout Europe, much of the industrial structure is made of small and medium- sized enterprises (SMEs) in fields such as agriculture, manufacturing, commerce and services. For social and historical reasons, these tend to aggregate into sectoral clusters in various parts of respective countries.

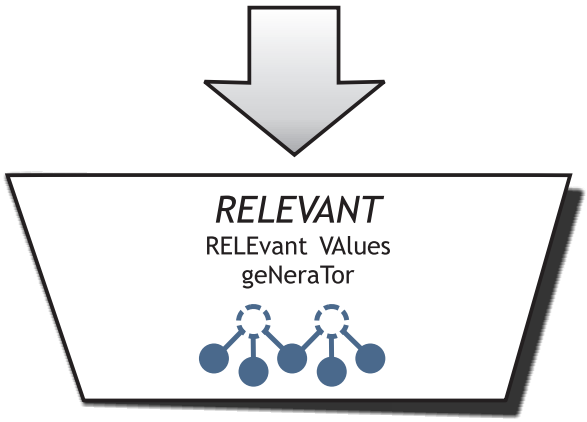
One of the keys to sustainability and success is being able to get information such as a cheaper supplier, an innovative working method, a new market, potential clients, partners, sponsors, and so on. The knowledge of competitors' data may provide remarkable advantages. Enterprises usually publish in Internet several data about their activities by means of web information systems. Current Internet search tools are inadequate as search results are often of little use with their huge number of page hits.

Suppose an SME needs to find out about a topic - a product, a supplier, a fashion trend, a standard, etc. For example, a search is made for 'deep moulding processes'. A query to Google for 'deep moulding processes' listed more than 1 million hits at the time of writing, reporting not only deep moulding processes, but also the mould technology, enterprises and so on. Eventually a useful contact may be found, and the search can continue through links concerning the mould processes. It is manifest that looking for information in this way is a time-consuming task. A search engine trying to overcome current system was developed within the SEWASIE project[2] coordinated by our research group. Within SEWASIE, we exploited and improved the MOMIS system and, in particular, we built a knowledge base for the enterprise mechanical sector by extracting data from four specialized web-sites:

- `www.subforn.net`: provides access to a database containing more than 6.000 subcontractors. Companies are classified on the basis of their production. Mechanical and mould sectors are divided into 53 different categories. For each category, several specific kinds of production (almost 1000) are defined.

[2]http://www.sewasie.org

Adhesive agents	Finished and semifinished product tests
Adhesive labels	Finished and semifinished product weight tests
Adhesive tapes	Injection
Adhesives	Injection blow moulding
Aluminum and magnesium casting	Injection moulded parts
Assembling operations	Injection moulding machines, general purpose
Assembly	Intrusion moulding
Assembly of printed circuits	Mould engineering
Assembly/modules	Moulded parts
Automatic assembly machines	Moulded parts from semifinished products
Blake assembly	Moulding
Blow moulding	Moulds for Ceramics
Blow moulding machines	Moulds for plastic and non-metal parts
Blowing	Moulds manufacturing
Blowing and injection moulding or forming	Moulds, Dies
Cast iron casting	Normal moulding
Cast moulded parts	Other finished and semifinished product tests
Casting	Physical tests
Casting Dies	Plastic castings
Casting machines for open moulds	Plastic technical products moulding
Casting with other moulding methods	Plastic Molding
Design	Precision moulds for thermoplastic resins
Design/development	Rotation moulding
Designing	Sandwich moulding
Dip moulding	Test modules
Family mould injection	Testing
Finished and semifinished product dimensional tests	Testing of modules and equipment



Design	Design; Design/development
Adhesives	Adhesive agents; Adhesive labels; Adhesive tapes; Adhesives
Assembly	Assembling operations; Assembly; Assembly of printed circuits; Assembly/modules; Automatic assembly machines; Blake assembly
Testing	Finished and semifinished product dimensional tests; Finished and semifinished product tests; Finished and semifinished product weight tests; Physical tests; Test modules; Testing; Testing of modules and equipment
Plastics Molding	Plastics Molding
Moulding	Blow moulding; Blow moulding machines; Cast moulded parts; Casting with other moulding methods; Family mould injection; Injection; Injection blow moulding; Injection moulded parts; Injection moulding machines, general purpose; Intrusion moulding; Mould engineering; Moulded parts; Moulded parts from semifinished products; Moulding; Moulds for Ceramics; Moulds manufacturing; Moulds, Dies; Normal moulding; Precision moulds for thermoplastic resins; Rotation moulding; Sandwich moulding
Casting	Aluminum and magnesium casting; Cast iron casting; Cast moulded parts; Casting; Casting Dies; Casting machines for open moulds; Casting with other moulding methods; Plastic castings
Injection	Blowing and injection moulding or forming; Family mould injection; Injection; Injection blow moulding; Injection moulded parts
Moulding; Blowing	Blow moulding; Blow moulding machines; Blowing; Cast moulded parts; Casting with other moulding methods; Dip moulding; Family mould injection; Injection blow moulding; Injection moulded parts; Intrusion moulding; Mould engineering; Moulded parts; Moulded parts from semifinished products; Moulding; Moulds for Ceramics; Moulds manufacturing; Moulds for plastic and non-metal parts; Moulds, Dies; Normal moulding; Plastic technical products moulding; Precision moulds for thermoplastic resins; Rotation moulding; Sandwich moulding

Figure 4.1: Example of relevant values

- `www.plasticaitalia.net`: contains more than 12.000 italian companies classified on the basis of a three level hierarchy specialized in more than 300 items.

- `www.tuttostampi.com`: contains 4000 companies categorized in 58 different kinds of services.

- `www.deformazione.it`: more than 2000 companies are catalogued on the basis of 39 different sectors.

By means of MOMIS, a knowledge base composed of two main global classes is built: one class stores data about companies, the second one contains all the production categories.

According to this representation, it was very difficult for a user to select companies working on specific sectors: the user does not know the more than 1500 possible categories (the union of all the different categories used in the four web-sites). If she/he is expert of a specific site he might focus on a specific category but similar categories may be denoted with different names in other sites, therefore the answer will be incomplete.

A great help for the user would be the knowledge of a small amount of values (say, less than 100) which "represent" the complete domain, taking into account also the different terminologies, synonyms, source specificities, etc. Such "relevant values" could provide an overview of data and greatly improve the effectiveness, completeness and usefulness of queries (see figure 4.1).

4.2 Eliciting Relevant Values from Data

There are several models for representing knowledge bases. Without loss of generality, let us refer to the concepts of MOMIS. The Global Virtual View built with MOMIS is composed of Global Classes, with Global Attributes (GA). Our goal is to extract the relevant values of a GA. Each relevant value is described by a relevant value name and a set of values of the attribute domain.

The idea is that analyzing an attribute domain, we may find values which may be clustered because *strongly related*. Providing a name to these clusters, we may refer to a relevant value name which encompasses a set of values. More formally, given a class C and one of its attributes At, a **relevant value** for it, rv^{At} is a pair $rv^{At} = \langle rvn^{At}, values^{At} \rangle$. rvn^{At} is the name of the relevant value set, while $values^{At}$ is the set of values referring to it. For instance, in figure 4.1 the values "Adhesive agents, Adhesive labels, Adhesive

tapes, Adhesives" are clustered and the relevant name "Adhesives" is elicited from data.

Now we should answer two questions: how can we cluster the values of the domain in order to put together in a relevant value a set of values which are strongly related? How can we choose the relevant value names? The first question will be answered by means of clustering techniques, adapted to the problem on hand; the second will require the intervention of the designer, but we will provide, in Section 4.3.4, an effective *assistant*.

Like most cluster tasks with non-numeric attributes, the problems are related to find an effective representation of the points (i.e. the attribute values) in a space, and to devise a suitable similarity function to be exploited by the clustering algorithm. The technique we propose builds a binary representation of the attribute values, and exploits three different kinds of measure to build some structure upon the flat set binary representation: 1) the *syntactic* similarity, mapping all the words of the attribute values in an abstract space, and defining a syntactic similarity function in such space; 2) the *domination* measure, expressed by the root elements described later on; and 3) the *lexical* similarity, which tries to identify semantically related values expressed with a different terminology.

Such measures are automatically extracted: the manual annotation of each attribute value (e.g. with reference to a given ontology) would be a time-consuming and error-prone operation also discouraged by the high number of values and the update frequency.

The similarity measures we propose are then used by a clustering algorithm (in *RELEVANT* the user may generate both partitions and overlapped clusters). The clusters of values produced are so far called *relevant values sets*. The user may balance the weight of the three different similarity measures.

4.2.1 The syntactic similarity

Terms related to the same object may have the same etymology and then share a common root: several similarity measures are based on this idea (e.g. the Levenshtein distance and the other metrics derived from it). In the same way, we may assume that related attribute values share terms. By means of this measure, we group different attribute values sharing common terms.

It is trivial to show that a term may be polysemous, i.e. it may be used in different attribute values with different meanings, especially in multi-word values. In our experience, the syntactic similarity alone could not be sufficient, but in conjunction with the similarities described below, it provides satisfactory results.

4.2.2 Domination: the root elements

Nomina sunt consequentia rerum
Giustiniano, Institutiones, Liber II, 7, 3 [3]

A similarity measure may be extracted from the *Domination* relationships between the attribute values. Considering two attribute values a_1 and a_2, we say that a_1 dominates a_2 if a_1 is more "general" than a_2. Any partial order on attribute values could be used to define domination.

On the basis of an analysis of several databases, we observed that it is frequent to have string domains with values composed of many words, also with abbreviations. We observed also that the same word, or group of words, may be further qualified (i.e. specialized) with multiple words in many ways. For example, the attribute describing a kind of production for a mechanical enterprise may contain the value "Mould" and the values "Mould ejectors, Mould engineering, ...". Thus, we approximate the domination between attribute values, a semantic property, with the *Contains* function, a syntactic property. *Contains* is a function based on string containment: $Contains(X, Y) = true$ iff $stem(X) \supseteq stem(Y)$, where X and Y are sets of words and *stem* is a *stemming operator* for words[4]. Then we say that Y dominates X if it is contained in X. The domination is a partial order and can be represented by an oriented graph. Let us say, for instance, that an edge goes from the dominating value (more general) to the dominated one (more specific). The integration designer should verify how much the graph represents the general notion of a value being "more general" than another, but in our experience the graph is usually sound.

Our idea is to exploit the domination for building clusters of values around *root elements*. A root element is an attribute value with only outgoing edges in the domination graph, and can be taken as a representative of the cluster composed by the nodes recursively touched by its outgoing edges.

Running Example 1 Let us consider an attribute *At* representing production categories of enterprises of the mechanical sector. For simplicity we assume that *At* has the following values: $W = \{$Blowing, Blow moulding, Moulding, Casting, Injection moulding, Rubber processing, Rubber processing machines, Steel, Steel casting, Assembling operations, Assembly$\}$. Six root elements are computed for the *At* values (see Table 4.1), i.e. only six values are contained in other values without containing any other value. The same value may be linked to different root elements (see Steel casting and Blow moulding).

[3] Names are consequences, or, one might say, the expressions, of things.
[4] a standard operator in natural language processing.

Root element	Values
Rubber processing	Rubber processing, Rubber processing machines
Steel	Steel, Steel casting
Casting	Casting, Steel casting
Moulding	Moulding, Blow moulding, Injection moulding
Blowing	Blowing, Blow moulding
Assembly	Assembly, Assembling operations

Table 4.1: Root Elements for the Production Categories attribute domain

4.2.3 The lexical similarity: using WordNet

WordNet is a large lexical database grouping English words into sets of cognitive synonyms (synsets), each one expressing a distinct concept. Synsets are described with a definition (a *gloss*) and are interlinked by means of conceptual-semantic and lexical relations. Since a term may be associated to different synsets due to the polysemy, a manual operation is generally required to the user for selecting the appropriate synset for each term. On the other hand, by exploiting the lexical similarity it is possible to group different values which refer to semantically related synsets.

Two different values, sharing one or more synsets are potentially similar. We can thus compute similarity on the basis of the shared synsets.

Finally, we observe that a compound value is in general composed of nouns linked with an (implicit) relationship. Consequently, we may consider them as "multi-word" values that is sets of single-words values.

Running Example 2 Let us cluster the values W of example 1 using the lexical similarity only. The result is a set of four groups of values as table 4.2 shows:

Clusters
Assembling operations
Assembly
Rubber processing, Rubber processing machines
Blowing, Blow moulding, Moulding, Casting, Injection moulding, Steel, Steel casting

Table 4.2: Clusters calculated with lexical similarity

The clustering algorithm puts into the same cluster MOULDING and CASTING since the lemmas Cast and Mould refer to the same synset. On the other hand, ASSEMBLY and ASSEMBLING do not refer to the same Wordnet synset, and they are split into different clusters. Handling multi-words attribute values as sets of single-words attribute values produces chain of partially related values: e.g. steel casting is put in the same relevant value of casting and steel.

4.3 The *RELEVANT* prototype

RELEVANT is a software tool for calculating relevant values. Giving as input a list of attribute values, *RELEVANT* generates a set of relevant values according to the designer selections. Figure 4.2 shows the *RELEVANT* functional architecture, which is organized into five blocks:

1. **Data pre-processing**: three binary representations of the values of an attribute are obtained with three matrices representing the different kinds of similarity measure.
2. **Similarity Computation**: the designer selects how to measure the similarity (metrics selection) and which kinds of similarity are used (the syntactic similarity, the domination measure, the lexical similarity or a combination of the three).
3. **Clustering technique selection**: this module implements some clustering algorithms to compute the set of relevant values on the basis of the choices made at step 2.
4. **Name selection**: for each group of values defined in step 3, a name representative of all the values has to be identified.
5. **Validation**: we implemented some standard techniques to evaluate cluster quality. At present, additional work and experiments are necessary to go beyond the simple evaluation, so as to provide effective assistance to the designer in the critical task of parameter configuration.

4.3.1 Step 1: Binary Representation of attribute values

The *RELEVANT* starting point is the creation of three binary matrices, according to the different measures introduced in section 4.2: MTV, MTR,

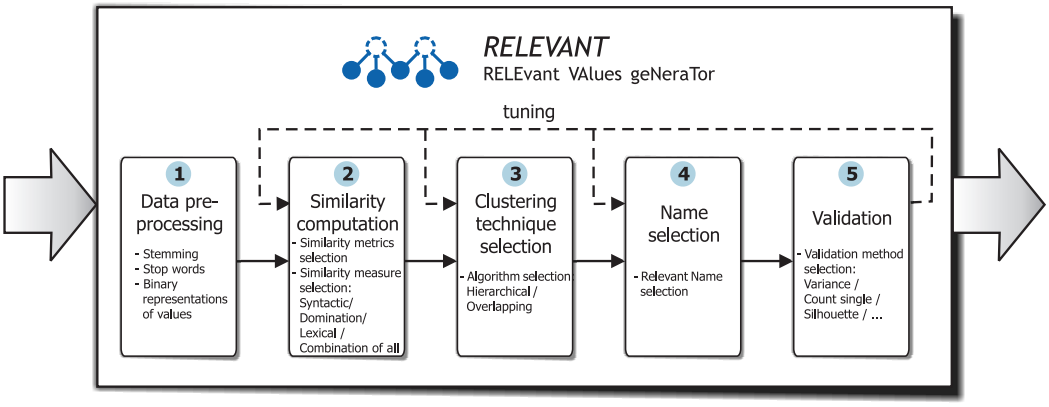


Figure 4.2: The *RELEVANT* functional architecture

MTL.

The *Syntactic Matching Table* (*MTV*) is a binary representation of all the values of an attribute *At* w.r.t. the universe of words considered (i.e. is the union of the words included in the extension of *At*). Notice that multi-word attributes contribute to the universe of words with multiple words.

MTV is typically sparse: for each row there is a number of elements different from zero that is equal to the number of words contained in the associated attribute, except for the stop–words.

The *Root Elements Matching Table* (*MTR*) shows the root elements associated to the attribute values: each column of the matrix is a root element and the rows are the attribute values.

The *Lexical Matching Table MTL* shows the synsets associated to the attribute values: each column of the matrix is a synset and the rows are the attribute values.

Running Example 3 For simplicity, we show only the *MTV* matrix creation (see table 4.3). The application of the stem operator to the data set introduced in Running example 1 generates the following sets of strings:

$stem(W)$= a^0[BLOW], a^1[MOULD, BLOW], a^2[MOULD], a^3[MOULD, INJECT], a^4[CAST], a^5[RUBBER, PROCESS], a^6[RUBBER, PROCESS, MACHIN], a^7[STEEL], a^8[STEEL, CAST], a^9[OPER, ASSEMBL], a^{10}[ASSEMBL].

The universe of words U obtained is the following:

U = { u^0[OPER], u^1[STEEL], u^2[RUBBER], u^3[MOULD], u^4[BLOW], u^5[PROCESS], u^6[INJECT], u^7[MACHIN], u^8[CAST], u^9[ASSEMBL]]}.

	u^0	u^1	u^2	u^3	u^4	u^5	u^6	u^7	u^8	u^9
a^0	0	0	0	0	1	0	0	0	0	0
a^1	0	0	0	1	1	0	0	0	0	0
a^2	0	0	0	1	0	0	0	0	0	0
a^3	0	0	0	0	0	0	1	0	0	0
a^4	0	0	0	0	0	0	0	1	0	0
a^5	0	0	1	0	0	1	0	0	0	0
a^6	0	0	1	0	0	1	0	0	0	0
a^7	0	1	0	0	0	0	0	0	0	0
a^8	0	1	0	0	0	0	0	1	0	0
a^9	1	0	0	0	0	0	0	0	0	1
a^{10}	0	0	0	0	0	0	0	0	0	1

Table 4.3: MTV obtained for the considered set of values

4.3.2 Step 2: Similarity computation

Two tasks are executed in this step: the selection of the metrics for computing the similarity on the matrices created in the previous step and the computation of the affinity matrices AMV, AMR and AML derived from the matching tables MTV, MTR and MTL respectively.

Concerning the first task, the tool implements some of the metrics commonly adopted in information retrieval (Simple Matching, Russel & Rao measure, Tanamoto Coefficient, Sorensen measure, Jaccard's Similarity [113]). Due to the sparseness of the binary matrix, the Jaccard similarity metric, which only considers the positive values in both the attribute value representations[5], is used in this paper and set as default.

Concerning the second task, three new matrices express the three different affinity measures calculated by applying the selected similarity metrics on MTV, MTR and MTL. The matrices are built as follows. Given a matrix $AMV(AMR, AML)$ a generic element $e_{i,j}$ is obtained computing the similarity between the e_i and e_j rows of the matrix $MTV(MTR, MTL)$, on the basis of the selected metrics.

Finally AMV, AMR and AML are linearly combined into the Global Affinity Matrix $GAM = \|gam_{hk}\|$. An element $gam_{hk} = lc_v \times amv_{hk} + lc_r \times amr_{hk} + lc_l \times aml_{hk}$, where the values of lc_v, lc_r and lc_l are chosen by the designer such that $lc_v, lc_r, lc_l \in [0, 1]$ and $lc_v + lc_r + lc_l = 1$.

Running Example 4

[5]Let us define B_{11} as the total number of times a bit is ON in both bit strings, B_{00} as the total number of times a bit is OFF in both bit strings, and L as the length of the bit string, the Jaccard metric is defined as $B_{11}/(L - B_{00})$

Table 4.4 shows the affinity matrix AMV computed with the Jaccard metric related to MTV in table 4.3.

	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}
a^0	1	0.5	0	0	0	0	0	0	0	0	0
a^1	0.5	1	0.5	0.33	0	0	0	0	0	0	0
a^2	0	0.5	1	0.5	0	0	0	0	0	0	0
a^3	0	0.33	0.5	1	0	0	0	0	0	0	0
a^4	0	0	0	0	1	0.67	0	0	0.5	0	0
a^5	0	0	0	0	0	1	0.67	0	0	0	0
a^6	0	0	0	0	0	0	1	0	0	0	0
a^7	0	0	0	0	0	0	0	1	0.5	0	0
a^8	0	0	0	0	0.5	0	0	0.5	1	0	0
a^9	0	0	0	0	0	0	0	0	0	1	0.5
a^{10}	0	0	0	0	0	0	0	0	0	0.5	1

Table 4.4: The Affinity Matrix AMV

4.3.3 Step 3: Clustering technique selection

The prototype implements two different clustering algorithms: a classical agglomerative hierarchical clustering algorithm performs a partition of the values set, a second algorithm generates overlapping clusters (a variation of the algorithm in [40] is implemented).

The hierarchical clustering algorithm. A hierarchical clustering algorithm classifies elements into groups at different levels of affinity, forming a tree [58]. The hierarchical clustering procedure is applied to the matrix GAM. Once the affinity tree has been built, clusters are interactively computed on the basis of the numerical affinity values in the affinity tree and a threshold-based mechanism for cluster selection specified by the designer. High values of threshold return small, highly fragmented clusters. By decreasing the threshold value, bigger clusters are generated.

The overlapping clustering algorithm. The algorithm is based on the technique described in [40] and it is based on the idea of extending some sets of values given as input with other data set elements. In particular, the algorithm starts from a set of *poles* $\mathcal{P} = \{P_1, ..., P_l\}$ where P_i is a subset of the considered values set and $P_i \cap P_j = \{\}\ \forall i \neq j$. Then, a membership degree is calculated for each elements of the values set with respect to each pole.

Finally, by means of a specific similarity measure evaluating the membership degrees, each element is assigned to one or more poles similar to it.

It is trivial to show that the results are highly dependent on the heuristic used for calculating the initial set of poles. Using the similarities available in our specific model, we implemented two techniques for calculating poles: the first one considers the results of the hierarchical clustering as poles, the second one considers the root elements as poles. The results are different: in the first case the similarity measures assume a key role; in the second case, no similarity measure is computed since the algorithm exploits only the domination.

Running Example 5 Table 4.5 shows the relevant values calculated using the hierarchical clustering algorithm with the linear combination technique and the overlapping clustering algorithm with root elements as poles. Names are selected as described in Step 4.

- Hierarchical clustering: Poles are obtained with the linear combination ($lc_l = 0.5 lc_v = 0.4, lc_r = 0.1$) and $T = 0.08$ (see Table 5(a) where the name and the associated values are shown for each relevant value)

- Root elements as poles (see Table 5(b))

(a) Partition obtained with the linear combination

Name	Values
Assembly	Assembling operations; Assembly
Rubber processing	Rubber processing; Rubber processing machines
Casting; Steel; Moulding; Blowing	Blowing; Blow moulding; Casting; Steel; Steel casting; Moulding; Injection moulding

(b) Overlapped clusters obtained with root elements as poles

Name	Values
Assembly	Assembling operations; Assembly
Rubber processing	Rubber processing; Rubber processing machines
Moulding	Moulding; Injection moulding; Blow moulding
Blowing	Blowing; Blow moulding
Steel	Steel; Steel casting

Table 4.5: Examples of relevant values

4.3.4 Step 4: Name selection

A relevant value name is typically the most general value among the *values*, i.e. given a generic $rv_i = \langle rvn_i, values_i \rangle$, rvn_i is the most general value of $values_i$. The simplest way to detect a list of rvn_i candidates is to use the *Contains* function. The designer may select the most appropriate name among them.

Running Example 6 Let us consider a relevant value with *values*: Assembling operations and Assembly. Assembly is selected as name, since it contains the other element. On the other hand, if we consider the following *values*: Blow moulding, Extrusion / blow moulding lines, Blowing, Injection blow moulding, Moulding, Injection moulding, there are two candidates (i.e. Moulding and Blowing) satisfying the containment conditions above.

4.3.5 Step 5: Validation

The results of clustering algorithms must be assessed with quality measures. We implemented a set of standard quality measures to support the designer in the tuning activity. As stated in [70], two main cluster validation methods could be defined: external criteria, which are based on a comparison with a pre-specified cluster structure, provided by external knowledge (in our case human domain experts), and internal criteria, which are based on quantities that involve the vectors of the data set themselves. To provide a complete validation framework we compute a set of quantitative measures, some external cluster validity indexes (Rand, Jaccard, Folkes and Mallows indexes), and an internal cluster validity index (Silhouette index). Every experiment produced a number of outliers, i.e. clusters containing a single value. Only a few experiments produced a number of outliers quite high. In our setting, outliers are values for which no effective, synthetic metadata are provided: the number of outliers must be compared with that of the reference set. The measures do not consider the outliers. The meaning of the measures is the following:

- **countRV**: number of relevant values obtained for the configuration. This value depends on the threshold set by the integration designer;

- **average, max_elements, variance**: the descriptive statistics over the number of elements; in particular, average expresses the average number of values belonging to a relevant value, max_elements indicates the dimension of the largest cluster and the variance shows the variance

degree among the dimensions of the clusters; for values sets equally distributed on the domain max_elements is close to the average value and variance is low; the average value also gives an idea of the effectiveness of metadata representativity and "compression", all the elements included in non-outlier cluster are represented by the associated relevant value

- **percentage of outliers**: best results are when it is close to that of reference set;

- **Rand Statistic index, Jaccard index, Folkes and Mallows index [70]**: compute the closeness of two sets of clusters evaluating couples of values that belong to the same cluster in both the sets;

- **silhouette [103]** (only if a hierarchical clustering algorithm is used): calculates for each object a silhouette value, which ranges from -1 (badly clustered) to 1 (well clustered); then, for each cluster calculates the average index; the global index in the table is the weighted average over all the clusters, excluding outliers;

- **overlapping degree** (only if an overlapping clustering algorithm is used): indicates the percentage of elements which belong to more than one relevant value.

Notice that the Rand, Jaccard, Folkes and Mallows indexes compare two different sets of clusters. We use these indexes both to compare the *RELEVANT* results w.r.t. a reference set, and to compare the differences between two different parameter settings. The reference set is provided by a domain expert or is a "gold standard".

4.4 Experimental results

We considered a subset of the 1500 category names composed of 300 elements: such limitation allows a domain expert to manually build a set of relevant attribute values as reference set and to easily compare the quality of the prototype results with the reference set. Since *RELEVANT* is highly parametrized, we tested the prototype in some significant configurations. Chosen a similarity metric (in this case the Jaccard metric), we set up the clustering threshold in order to produce a number of relevant values close to that of the reference set. Table 4.6 summarizes the characteristics of the nine different configurations we considered.

RS	the reference set provided by the domain expert
SY-HI	Syntactic similarity only ($lc_v = 1, lc_r = 0, lc_l = 0$) and hierarchical clustering
SY-OVHI	Syntactic similarity only and overlapping clustering where poles are the clusters obtained with the SY-HI configuration
DO-HI	Domination only ($lc_v = 0, lc_r = 1, lc_l = 0$) and hierarchical clustering
DO-OVHI	Domination only and overlapping clustering where poles are the clusters obtained with the DO-HI configuration
LX-HI	Lexical similarity only ($lc_v = 0, lc_r = 0, lc_l = 1$) and hierarchical clustering
LX-OVHI	Lexical similarity only and overlapping clustering where poles are the clusters obtained with the LX-HI configuration
LC-HI	linear combination of the similarities ($lc_v = 0.4, lc_r = 0.1, lc_l = 0.5$) and hierarchical clustering
LC-OVHI	linear combination of the similarities ($lc_v = 0.4, lc_r = 0.1, lc_l = 0.5$) and overlapping clustering where poles are the clusters obtained with the LC-HI configuration
OVRE	overlapping clustering where poles are the root elements.

Table 4.6: The configurations evaluated

External analysis. We compared the relevant values generated by *RELEVANT* (with several parameter settings) with a reference set provided by a domain expert. Tables 4.7 and 4.8show some descriptive statistics (the first six rows) and three external evaluation indexes (rows 7-9).

Internal analysis with the Silhouette Index. We compared the relevant values generated by RELEVANT with different configurations based on hierarchical clustering algorithms using the Silhouette Index (see Tables 4.7 and 4.8, rows 10-12). The Silhouette validation method, as well as other standard internal validation methods, is not applicable for overlapping approaches. We considered three silhouette measures: the average, minimum and maximum of the Silhouette index for all the clusters. Of course, the largest index indicates the best clustering, and therefore the best set of relevant values.

		RS	SY-HI	SY-OVHI	DO-HI	DO-OVHI	LX-HI
1	count rv	45	18	91	11	11	6
2	average	6.36	11.61	3.10	23.09	23.09	37.50
3	max elements	76	94	58	196	173	197
4	outlier fraction	4.67	30.33	6.00	15.33	15.33	25.00
5	coefficient of variation	0.51	0.40	1.26	0.32	0.32	0.23
6	overlapping degree	23	-	37	-	0.12	-
7	Rand Statistic	-	0.85	0.89	0.62	0.68	0.59
8	Jaccard	-	0.20	0.32	0.20	0.31	0.13
9	Folkes and Mallows	-	0.34	0.49	0.44	0.41	0.30
10	Silhouette Avg	-	0.35	-	0.84	-	0.73
11	Silhouette Min	-	-0.01	-	0.15	-	0.13
12	Silhouette Max	-	0.68	-	1.00	-	1.00

Table 4.7: External and Internal evaluation

		RS	LX-OVHI	LC-HI	LC-OVHI	OVRE
1	count rv	45	81	4	35	13
2	average	6.36	3.70	57.75	7.49	4.95
3	max elements	76	162	209	111	39
4	outlier fraction	4.67	0.00	23.00	12.67	4.33
5	coefficient of variation	0.51	1.14	0.16	0.57	0.54
6	overlapping degree	23	10	-	16	30
7	Rand Statistic	-	0.70	0.55	0.81	0.95
8	Jaccard	-	0.15	0.12	0.15	0.52
9	Folkes and Mallows	-	0.29	0.29	0.27	0.70
10	Silhouette Avg	-	-	0.51	-	-
11	Silhouette Min	-	-	0.10	-	-
12	Silhouette Max	-	-	0.75	-	-

Table 4.8: External and Internal evaluation (2)

Experimental remarks. By analyzing the measures summarized in Tables 4.7 and 4.8, we observe that:

- The evaluation is done for a given similarity metric (in this case, the Jaccard Similarity). Experiments with other similarity metrics confirm the trend.
- With the syntactic similarity alone (columns 3 and 4), it was impossible to set the clustering threshold as to obtain a number of relevant values close to that obtained by the domain expert. The result is either a low

number of relevant values (17, not shown in table) or a too high number (109, with 30% of outliers). The syntactic similarity relies only on the similarity of the values. In our domain, there are several values made up of multiple words. The algorithm erroneously uses these values for creating large relevant values composed of terms not related to each other, unless a high threshold is used.

- The technique for calculating the lexical similarity produces low values in the lexical affinity matrix. Consequently, the coefficient lc_l must be greater that the other ones in order to contribute to the result. Moreover, we tested the algorithm with a standard WordNet release; better results may be obtained by extending WordNet with new synsets representing the attribute values which are peculiar for this environment.

- The Silhouette index shows that three of the four test average over 0.5, with a satisfactory 0.85 for the DO-HI experiment. The minimum is also above 0.1 for the three best cases, meaning that even the worse cluster gives some contribution (0 means indifference).

- The best result without overlapping is given by the DO-HI experiment, meaning that without clustering the dominance seem to be the most important property.

- The overlapping clustering generally produces better results, i.e. some reasonably large clusters, a small number of outliers, and low variance. In particular the settings OVRE and LC-OVHI generate the best results: with OVRE the clusters are not too big (maximum 39 elements) and with LC-OVHI the number of clusters (35) is not far from that of the reference set; In both cases the variance coefficient is quite low and the rand statistics is slightly below one.

4.5 Querying with Relevant Values

Thanks to the knowledge provided by relevant values, the user has two new ways of formulating queries, according to two scenarios.

1. The user has only a general idea of what she/he is searching for and composes a query predicate for instance by selecting a value x among the relevant value names. Note that instead of using the classical equality or LIKE operator, we should consider a new one, say RELATED TO, taking into account the mapping between relevant value names and values. It is beyond the scope of this paper to discuss such operator,

but a naive implementation could be to substitute At RELATED TO x where x is a relevant value name, with At IN (SELECT *values* FROM METADATA.At WHERE rvn='x')

To give a flavor of the novelty of the approach, we should observe that: (a) The user seldom has a deep knowledge of all the integrated data, so the list of the relevant value names, elicited from data, is of great help in providing insight on the value domain, and in assisting query formulation; (b) w.r.t. the base SQL predicate At LIKE $'\%x\%'$ we propose a rewriting of the query which is guided by the semantics of clustering and string containment, and uses also, as base tools, the information retrieval techniques of stemming and stop words.

2. The user knows that the result must include tuples satisfying the predicate $At = v$, but she/he is aware that, due to the integration process, tuples with values v' *similar to* v might also be relevant. In this case the query could be transformed in a query of type 1 above by substituting $At = v$ with At RELATED TO rvn, where $v \in values(rvn)$, or possibly with a disjunction of predicates like that, if overlapping clustering is used.

4.6 ***RELEVANT***$^{\text{News}}$: a semantic news feed aggregator

Many newspapers publish their news in Internet. A recent research from the Italian Institute of statistics[6] shows that there is an increasing trend of mastheads publishing their contents on the Net often joining to the paper edition an Internet edition with special and more complete information [7]. Internet newspapers may update their contents frequently: thus there is not a daily issue but the news are continuously updated and published. As a consequence, hundreds of thousand of partially overlapping news are daily published.

The amount of information daily published is so wide that is unimaginable for a user. On the other hand, the availability of news generates new updated information needs for people. The RSS technology supports Internet users in staying updated: news are published in the form of RSS feeds that are periodically downloaded by specific applications called feed readers.

[6] http://www.istat.it

[7] Istat report about the Italian online newspapers (years 2005-2006), available at http://culturaincifre.istat.it/

In order to improve the users' selection of the interesting feeds from different newspapers, publishers group feeds in categories.

The RSS technology and the news classification in categories does not solve all the "news overload" issues. First, the categories are not fixed, and then the same topic may be called in different sites in different ways. Consequently, a user that wants to be updated about a specific topic has to manually browse the categories of potentially all the newspapers looking for interesting news. Then, the amount of news feeds daily published is so wide that automatic tools are required. If we consider the feeds published only by the five main Italian newspapers in one day, more than one thousand of news are available in their websites [8]. Such news are partially overlapping, since different newspapers publish the same information in different news. RSS feeds from different newspapers may carry the same information in different places, and therefore can confuse the reader. A great improvement might be to show groups, and leave to the reader the optional task of drilling down the group, if necessary, to compare the different flavors of the same information given by the different sources.

This work relies on the *RELEVANT* prototype, presented in Section 4.3. The tool has been conceived for improving the user's knowledge of the attributes of database tables: by means of clustering techniques, *RELEVANT* provides to the user a synthetic representation of the values of the attribute taking into account syntactic, dominance and lexical relationships for defining similarity measures among the attribute values.

In this Section we describe $RELEVANT^{\text{News}}$, a web feed reader with advanced features, since it couples the capabilities of *RELEVANT* and of a feed reader. By applying *RELEVANT* to the titles of the feeds, we can group related news published by different newspapers in different times in semantically related clusters. In particular, each cluster contains news related under the following dimensions: 1) Spatial perspective: the news with the similar titles published in different newspapers; 2) Temporal perspective: the news with the similar titles published in different times.

Several feed readers have been proposed in the literature (see section 4.7 for related works), but at the best of our knowledge $RELEVANT^{\text{News}}$ is the only lexical knowledge based feed aggregator.

Section 4.6.1 shows the $RELEVANT^{\text{News}}$ architecture, and in section 4.6.2 we discuss some experimental results. Section 4.7 shows some related works and, finally, Section 4.8 introduces future work.

[8] We considered the feeds on average available in the newspapers "Il Corriere della Sera", "La Repubblica", "La Gazzetta dello Sport", "Il sole 24 ore", "La Stampa" in a week of analysis.

4.6.1 *RELEVANT*$^{\text{News}}$ architecture

RELEVANT$^{\text{News}}$ is a web application including three components:

- A **feed aggregator** is in charge of collecting the feeds selected by the user;
- A **RSS repository**: *RELEVANT*$^{\text{News}}$ requires a database for sharing feeds published in different days by different newspapers;
- ***RELEVANT*** computes and groups similar news.

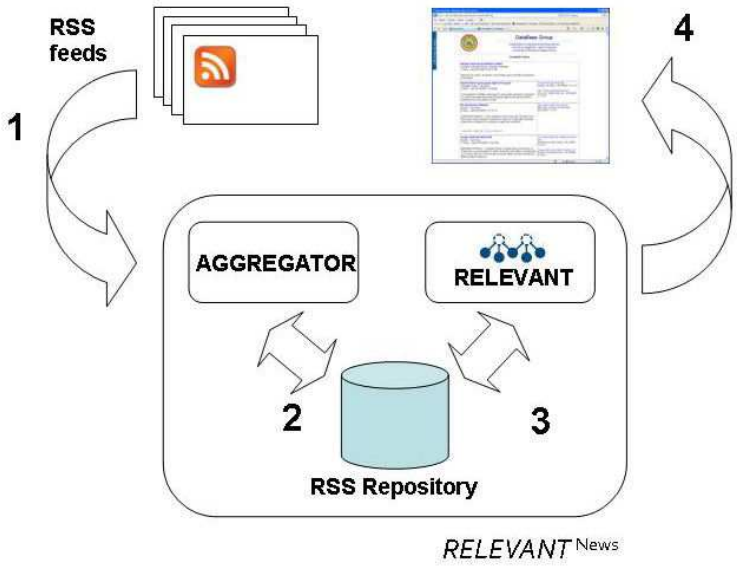


Figure 4.3: The *RELEVANT*$^{\text{News}}$ functional architecture

The *RELEVANT*$^{\text{News}}$ functional architecture is composed of four steps (see figure 4.3):

1. **selection of the news feeds**: a simple graphical user interface allows the user to select the interesting news feeds (by means of their URL) and to setup the updating policy, i.e. how frequently the feed has to be checked for new items;
2. **repository population**: a database supports the collection of the feeds. Thus it is possible to provide to the user news that are related to a topic, but are no longer published. The user may select a deadline for the maintenance of the news;

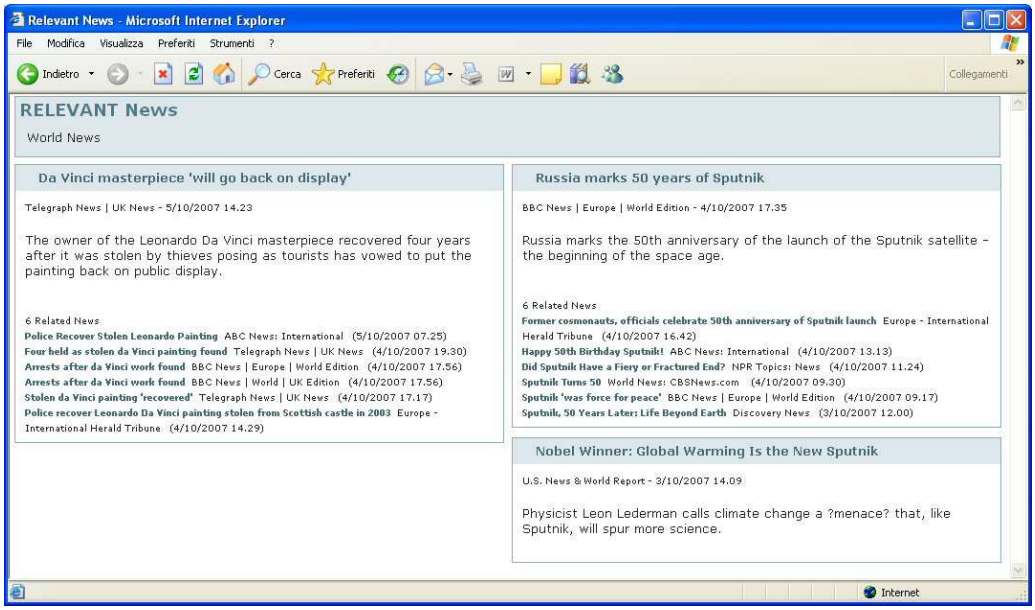


Figure 4.4: *RELEVANT*$^{\text{News}}$ screen-shot

3. **news clustering**: by means of *RELEVANT* similar news are grouped, and for each cluster, a news, representative of the cluster, is selected. Concerning the clustering process, a simple graphical interface may allow the user to parametrize the algorithm settings, establishing the dimension of the clusters (big clusters with loosely related news, or small clusters containing strictly related information), and tuning the weight of the different similarities (lexical, dominance and syntax). Concerning the selection of the news representative of the cluster (the *relevant news*), the user may choose: a) the name extracted by *RELEVANT* ; or b) the last published news;

4. **Relevant news publication**: a web interface shows the news in terms of title, source, date and content. In case of clustered news, the relevant news is visualized together to the list of cluster related news.

In figure 4.4, a screen-shot of the *RELEVANT*$^{\text{News}}$ interface is shown. Each box contains a different news. In case of similarities, the relevant news text is shown in the box and the cluster related ones can be reached through a link in the bottom box.

4.6.2 Experimental results

We tested *RELEVANT*$^{\text{News}}$ analyzing 730 news from 30 feed providers, published from the 1st to 4th of October 2007. The limited number of feeds allows us to evaluate the results by means of quality indexes provided by the tool, and of some qualitative, user-supplied evaluations. Since a gold standard for news does not exist, and different clusters for the same set of feeds may be provided by a domain expert due to the different grouping criteria may be adopted, in the following, we will analyze several settings producing different clusters of news. Later on, after a brief explanation of the dataset, we will discuss some results and some numerical evaluations of 12 configurations.

Dataset analysis

The case study is defined by choosing 16 different news publishers with similar RSS feeds topic, i.e. world news, U.S. news and Europe news, analyzing altogether 30 RSS feeds. In particular, we considered 9 newspapers (6 from U.S., Chicago Tribune, New York Times, Wall Street Journal, Time, USA Today and U.S. News, and 3 European, Daily Telegraph, The Guardian and International Herald Tribune), 5 TV Network (4 from U.S., ABC News, CNN, CBS and Discovery Channel, and BBC from U.K.) and 2 International Press Agencies (Reuters and NPR).

During the period of time in analysis (4 days), each publisher provided on average 45 news, with a peak of Daily Telegraph, 141 news and ABC, 89 news, while Chicago Tribune only 5 news. Since the news topics are partially overlapping, the news are also partially overlapping: the same information may be published in different feeds at the same moment.

Evaluation of the results

Since it is not possible to compare the results produced by *RELEVANT*$^{\text{News}}$ with a gold standard, we will discuss and compare the results computed with different settings. In particular, we considered three different thresholds for the clustering algorithm (since the optimal number of clusters is not known , we considered thresholds producing respectively 300, 450 and 600 clusters) and two different tunings of the similarity parameters. Concerning these parameters, we evaluated a "lexical configuration", where the lexical similarity among the news titles assumes a main role, and a "syntactic configuration", where the syntactic similarity is the main similarity measure. We did not take into account a "dominance configuration" as its application in the analyzed dataset is not significant, i.e. only few title are "contained" in other

	SYN_300	LEX_300	SYN_450	LEX_450	SYN_600	LEX_600
# single	241	217	388	348	527	511
Max elem	268	78	30	20	9	7
Avg elem	2.49	2.40	1.55	1.61	1.22	1.24
Variance	16.17	7.31	2.13	1.96	0.81	0.81
Silhouette	0.31	0.34	0.35	0.35	0.45	0.49

Table 4.9: Qualitative results

titles.

The results computed by the *RELEVANT* feedback module are summarized in table 1. The Silhouette values highlight a good clustering process in all the settings (ranges from -1, worst to +1, best). Another interesting information is provided by the "count single" value, that in all the settings is closed to the number of obtained cluster (almost 80% of the computed clusters contains only one element in all the settings). These values, which may be symptom of weakness of the tool are due to news for which no significant similarity has been found. The analysis of the dataset confirms that the observed news are related to general/generic topics from the world, and in the period of time in observation no event with a worldwide importance happened. Thus, we may suppose that clusters similar to the ones computed by $RELEVANT^{\text{News}}$ may be produced by a human reader.

In Table 4.10, the clusters obtained considering two different configurations are analyzed. Qualitative analysis shows that lexical similarities improve the results. Table 4.10(a), where the clusters are computed with the syntactic configuration, shows that the news related to the recover of a stolen Leonardo da Vinci painting are grouped in two clusters. On the other hand, in Table 4.10(b) the news are grouped in the same cluster, due to the lexical similarities among the news titles. Similar considerations may be done for news titles represented in Tables 4.10(c) and 4.10(d). In this case, it is interesting to observe that the syntactic configuration produces three clusters, but the first news is correctly not included in a overall cluster (see Table 4.10(d)) in the lexical configuration, since it refer to a different topic.

4.7 Related Work

Metadata Extraction There is a rich literature about metadata extraction both in the area of Semantic Web, where metadata support automatic applications to understand web-site contents and in the area of Information Retrieval, where they allow document classification.

(a) News related to the "Da Vinci" stolen grouped with the syntactic configuration

#1	Arrests after da Vinci work found
	Da Vinci masterpiece "will go back on display"
	Four held as stolen da Vinci painting found
	Stolen da Vinci painting "recovered"
#2	Police recover Leonardo painting stolen from Scottish castle in 2003
	Police Recover Stolen Leonardo Painting

(b) News related to the "Da Vinci" stolen grouped with the lexical configuration

#1	Arrests after da Vinci work found
	Da Vinci masterpiece "will go back on display"
	Four held as stolen da Vinci painting found
	Stolen da Vinci painting "recovered"
	Police recover Leonardo painting stolen from Scottish castle in 2003
	Police Recover Stolen Leonardo Painting

(c) News related to the "Sputnik" grouped with the syntactic configuration

#1	Nobel Winner: Global Warming Is the New Sputnik
#2	Did Sputnik Have a Fiery or Fractured End?
	Former cosmonauts, officials celebrate 50th anniversary of Sputnik launch
	Happy 50th Birthday Sputnik!
	Sputnik "was force for peace"
#3	Russia marks 50 years of Sputnik
	Sputnik Turns 50
	Sputnik, 50 Years Later: Life Beyond Earth

(d) News related to the "Sputnik" grouped with the lexical configuration

#1	Nobel Winner: Global Warming Is the New Sputnik
#2	Did Sputnik Have a Fiery or Fractured End?
	Former cosmonauts, officials celebrate 50th anniversary of Sputnik launch
	Happy 50th Birthday Sputnik!
	Sputnik "was force for peace"
	Russia marks 50 years of Sputnik
	Sputnik Turns 50
	Sputnik, 50 Years Later: Life Beyond Earth

Table 4.10: A clustering example

Concerning the Semantic Web, [112] provides a complete state of the art evaluating 27 annotation tools. Among the analyzed tools, one of the most promising is the KIM platform [81], which addresses the complete cycle of metadata creation, storage and semantic-based search. KIM exploits an upper-level ontology which is used as reference for the created metadata. The DDC/RDF-Editor [117] is another interesting recently developed tool which automatically extracts values of metadata according to the Dublin Core Metadata Standard from web sources. Unlike these systems, RELEVANT has been conceived as a component of the MOMIS system which provides an integrated view of heterogeneous data sources where each class/attribute is associated to a well-understood lexical meaning (by means of a WordNet link). Our technique adds to the lexical meaning another kind of knowledge that is a "synthesis" of the values an attribute may assume. An interesting issue is addressed in [85] where a metadata acquisition infrastructure, called ASDI, which adapts lexicons and patterns to a domain hierarchy and user preferences, is presented. In particular, the authors introduce the concept of "high quality semantic metadata", i.e. metadata that accurately capture the meaning of the data they describe. To improve quality, a verification layer, composed of an automatic verification engine and a manual user evaluation tool is introduced. In RELEVANT, a set of standard quality measures (described in section 4.5) ensures the quality of the relevant values.

IR techniques have been widely used for classifying documents. In [104], dynamic taxonomies are proposed for supporting users in searches. By means of such multidimensional classification system, an element may be classified under several topics. The approach we propose in this paper builds a flat structure and is less efficient for supporting user searches. However, dynamic taxonomies require heavy manual work for their construction: a domain expert has to build a reference taxonomy and to associate the instances to the taxonomy items while our method is mostly unsupervised. In [108], the authors propose some techniques to map documents in an abstract space and to cluster them. The general idea is similar to the clustering component of our method and the work provides useful hints in the choice of the clustering algorithm and quality measure. With respect to this work, we extend the similarity measure with domination and lexical measures.

Finally, some works have analyzed the relationships amongst IR techniques, DB integration and metadata extraction [37]. In particular, in [72] a model, based on the standard OMGs MOF, is introduced for expressing metadata of different data sources. The metadata are then exploited for accessing, integrating and synthesizing information, as required by the user. According to this perspective, work may be considered in some way similar to the ROLL–UP/DRILL–DOWN operator of OLAP, implying a hierarchic

organization of attributes. However, in OLAP, the designer has to manually specify such attributes within the dimension table; in our approach the values aggregations are automatically discovered.

Advanced News Aggregators Several aggregators have been developed and implemented. Most of them are available as commercial products and their internal mechanisms are not known [9]. It is possible to group them in three different categories:

1. **Simple readers** provide only a graphical interface for visualizing and collecting RSS feeds from different newspapers. Simple functions supporting the user in reading are provided (e.g. search engine, different ordering, association of news to a map, ...);

2. **News classifiers** show the news classified on the basis of criteria sometimes decided by the user. Simple classifications may exploit the categories and/or the keywords provided by the web sites;

3. **Advanced aggregators** provide additional features for supporting the user in reading, clustering, classifying and storing news.

There are several interesting proposals of advanced aggregators in literature. In [66], Velthune, a news search engine is proposed. The tool is based on a naive classifier that classifies the news in few categories. Unlike this approach, $RELEVANT^{\text{News}}$ computes clusters of similar news on the basis of their title. Classifying thousands of news in few categories produces large sets of news belonging to the same category that are not easily readable by a user. In [90] the authors propose an aggregator, called RCS (RSS Clusgator System), implementing a technique for temporal updating the contents of the clusters. NewsInEssence [101] is an advanced aggregator that computes similar news on the basis of a TF*IDF clustering algorithm, and provides to the reader a synthesis of them. Although $RELEVANT^{\text{News}}$ does not provide any synthesis, it implements a parametrized clustering algorithm based on syntactic/lexical/dominance relationships, that may be properly tuned for improving the creation of the clusters. Finally, the idea of $RELEVANT^{\text{News}}$ may be compared with Google News [10] where each news is associated with a list of related information. Differently from us, Google News does not allow the user to select the newspapers. All the newspapers are analyzed and the news are provided to the user on the basis of a collaborative filtering [53].

[9] See http://www.dmoz.org/Computers/Software/Internet/Clients/WWW/Feed_Readers/ for a non complete set of aggregators.

[10] http://news.google.com/

4.8 Discussions and future work

We defined a new type of metadata, the relevant values of an attribute domain. These values are provided to the user in order to ease his sources understanding. As usual in data analysis, the startup phase requires the setting of several critical parameters. Nevertheless, for a given parameter setting, the technique calculates the relevant value set without any human intervention. Moreover the parameters and similarity metrics selection determine the quality of the relevant value set. Therefore, the designer has to carefully evaluate the results and possibly change some parameters in order to improve the result quality.

The experimental results evaluated by means of *RELEVANT* show that our technique produces results close to the relevant values provided by a domain expert. The best results are obtained by applying the overlapping clustering algorithms. A first outcome of the availability of the relevant values is a more effective way of querying data. The knowledge elicited by RELEVANT can be exploited by means of a new operator, RELATED TO, taking into account the mapping between relevant value names and values.

A second outcome of the techniques developed in *RELEVANT* is given by the $RELEVANT^{\text{News}}$ prototype. The $RELEVANT^{\text{News}}$ system is a news feed reader able to group similar news by means of data mining and clustering techniques applied to the feed titles. $RELEVANT^{\text{News}}$ is created by coupling the *RELEVANT* prototype to a feed aggregator and an RSS repository.

Future work will be addressed on calculating relevant values for groups of attributes, i.e. by considering the values of multiple attributes. Moreover, we are developing some techniques for extracting and using more detailed lexical similarities. In particular, hyponym and hypernym relationships between values may be exploited for generating more appropriate relevant values sets.

Future work related to the $RELEVANT^{\text{News}}$ prototype will be addressed on developing new techniques suitable for the feed domain. For example, we are studying a similarity based on term frequency-inverse document frequency (TD*IDF), which takes also into account the word-spread. The idea is that unusual words and specific terms may be related to the same news.

Chapter 5

Access Control in Data Integration Systems

Data integration/interoperation systems integrate information from different local sources to enable communication and exchange of data between them. A common model for these systems involves a global representation of the local data, which acts as a mediator for translating queries and conveying data to and from the distributed sources using the global-as-view (GAV) approach [86] (see Section 2.1).

The access to these shared resources needs to be controlled and enforced by global security policies, while local organizations need to mantain the control of their local security policies. Access control levels are heterogeneous across the different organizations, but they have to be merged into a global security model to guarantee the enforcement of security policies at the global level. The following requirements have to satisfied by the security framework: 1)*Autonomy*: the local security policies must not be affected by the security policy of the global level; 2)*Confidentiality*: given a security clearance, if a schema element is not accessible locally before the integration, then it must not be accessible after integration; 3)*Availability*: given a security clearance, if a local schema element is accessible before integration, then it must continue to be accessible after integration.

In section 5.1, a privacy-preserving method for classifying the integrated information that depends on the local security policies and preserves the autonomy of the local sources on their security policies is presented [45]. The proposed method transforms the security levels defined on the XML schema elements of each local source into security levels on the triples of the local RDF schemas, which form a lattice. The local security lattices are then merged together to generate a global partially ordered security graph. We show how the merged data in the global RDF schema can be classified in

different security classes belonging to the global partially ordered security graph.

In the recent years there has been increased usage of collaborative models of work and environment, such as Web 2.0 applications, cooperative projects on grids, Mashups applications. In collaborative environments, data and resources are shared by different groups and organizations in order to support common tasks. Depending on several factors such as the task, the participants, and data sensitivity, access to these shared resources needs to be controlled and enforced by security policies. A secuirty framework for collaborative environments have to be highly dynamic and flexible to satisfy the following requirements: multiple organizations share resources and tasks; users may not be previously identified and the number of users may be large and unknown in advance; users properties or attributes (e.g., age, status), such as environmental variables (e.g., time, position) and contextual variables (e.g., task, team membership) can change during users sessions; user's roles are not static (e.g., due to change of location); resource availability may change. The role-based access control (RBAC) model is particularly suited to dynamic task-oriented environments due to its flexibility and policy-neutrality [98], which enables it to express a large range of policies.

In section 5.2, a security framework for collaborative applications that relies on the role-based access control (RBAC) model, is presented. In our framework [43], roles are pre-defined and organized in a hierarchy (partial order). However, we assume that users are not previously identified, therefore the actions that they can perform are dynamically determined based on their own attribute values and on the attribute values associated with the resources. Those values can vary over time (e.g., the users location or whether the resource is open for visiting) thus enabling or disabling a users ability to perform an action on a particular resource. In our framework, constraint values form partial orders and determine the association of actions with the resources and of users with roles.

A prototype [44] has been implemented by exploring the capabilities of semantic web technologies, and in particular of OWL 1.1, to model both our framework and the domain of interest and to perform several types of reasoning. In addition, we have implemented a user interface whose purpose is twofold: (1) to offer a visual explanation of the underlying reasoning by displaying roles and their associations with users (e.g., as the users locations vary); and (2) to enable monitoring of users that are involved in a collaborative application. Our interface uses the Google Maps API and is particularly suited to collaborative applications where the users geospatial locations are of interest.

5.1 A Secure Mediator for Integrating Multiple Level Access Control Policies

Data integration/interoperation systems integrate information from different local sources to enable communication and exchange of data between them. A common model for these systems involves a global representation of the local data, which acts as a mediator for translating queries and conveying data to and from these sources using the global-as-view (GAV) approach [86] (see Section 2.1).

Semantic Web languages such as RDF Schema (or RDFS) [32] and OWL [107] are particularly suited to represent the global information and to abstract from the particular data formats (relational, XML, etc.) or from the different schemas within the same format, thus addressing respectively problems of syntactic heterogeneity [47] and of structural (or schematic) heterogeneity [116]. In Chapter 3 the THALIA benchmark for information integration systems (that focuses on syntactic and semantic heterogeneities) is introduced, and the MOMIS integration approach to deal with the benchmark is presented.

In this Section, we present a method for mapping security levels among the components of a distributed system where data in the local sources are represented in XML. Distributed data is integrated using a semantic-based approach that maps each XML schema into an RDF schema and subsequently integrates those schemas into a global RDF schema using a global as view (GAV) approach.

We present a privacy-preserving method for classifying the integrated information that depends on the local security policies and preserves the autonomy of the local sources on their security policies. The proposed method transforms the security levels defined on the XML schema elements of each local source into security levels on the triples of the local RDF schemas, which form a lattice. The local security lattices are then merged together to generate a global partially ordered security graph. We show how the merged data in the global RDF schema can be classified in different security classes belonging to the global partially ordered security graph.

The chapter is organized as follows. Section 5.1.1 defines the security problem of an integration/interoperation system and introduces an integration scenario. Section 5.1.2 presents our security framework, including the *autonomy*, *confidentiality*, and *availability* requirements, the local security lattices and the process in which they are merged to form a global security lattice; we introduce definitions and a theorem that states that the security mappings that need to be established between two local schemas and be-

tween a local and a global schema satisfy the requirements. Section 5.1.3 gives a brief overview of related work, and Section 5.1.4 describes our main contributions, and point to future work.

5.1.1 Problem Definition

The problem that we address is the security of an integration/interoperation model. In particular, if the local schemas are integrated in the global schema, how can the security policy of the global schema be specified taking into account the local security policies?

We adopt a model in which each local organization enforces a multiple level access control model on its schemas [31]. In this model, data are categorized into security levels and users are assigned security clearances.

We define a *partial order* or *lattice* $\preceq$ on the set of security levels as follows: given two security levels s_i and s_j, data classified at level s_i can be accessed by anyone with security clearance s_j, such that $s_i \preceq s_j$. The partial order can be represented by a directed acyclic graph. A chain in the graph represents a total order among the security levels along the chain.

For better investigating the problem, we introduce a possible integration scenario in which we consider two healthcare organizations, for instance a health insurance company and a hospital that want to integrate some of their patient data. The data are stored in XML. Figures 5.1.1 and 5.1.2 show respectively portions of the XML schemas of the hospital and of the health insurance company. Although data pertain to the same domain, the XML schemas display structural heterogeneities—the element *patient* is contained (nested) in the element *hospital* in one schema, while in the other schema the element *hospital* is contained (nested) in the element *customer*. In reality, the relationship between patients (or customers) and hospitals is "many-to-many" but due to the hierarchical nature of XML such relationships need to be represented using containment.

In addition to structural heterogeneity, our example also illustrates a case of semantic heterogeneity in that two elements that refer to the same concept have different names: *patient* and *customer*. In order to overcome syntactic, structural, and semantic heterogeneities, schemas can be integrated at a semantic level. For example, the problem of structural heterogeneities has been addressed in a previous approach [116], where a two-step integration framework is proposed. In the first step, the XML schemas are transformed into RDF schemas. RDF is a language built on top of XML, which can be used to describe relationships between entities. These relationships can be expressed in terms of *triples* of the form (s, p, o). The first element, s, is the *subject* of the triple, the second element, p, is the *predicate* or *property*, and

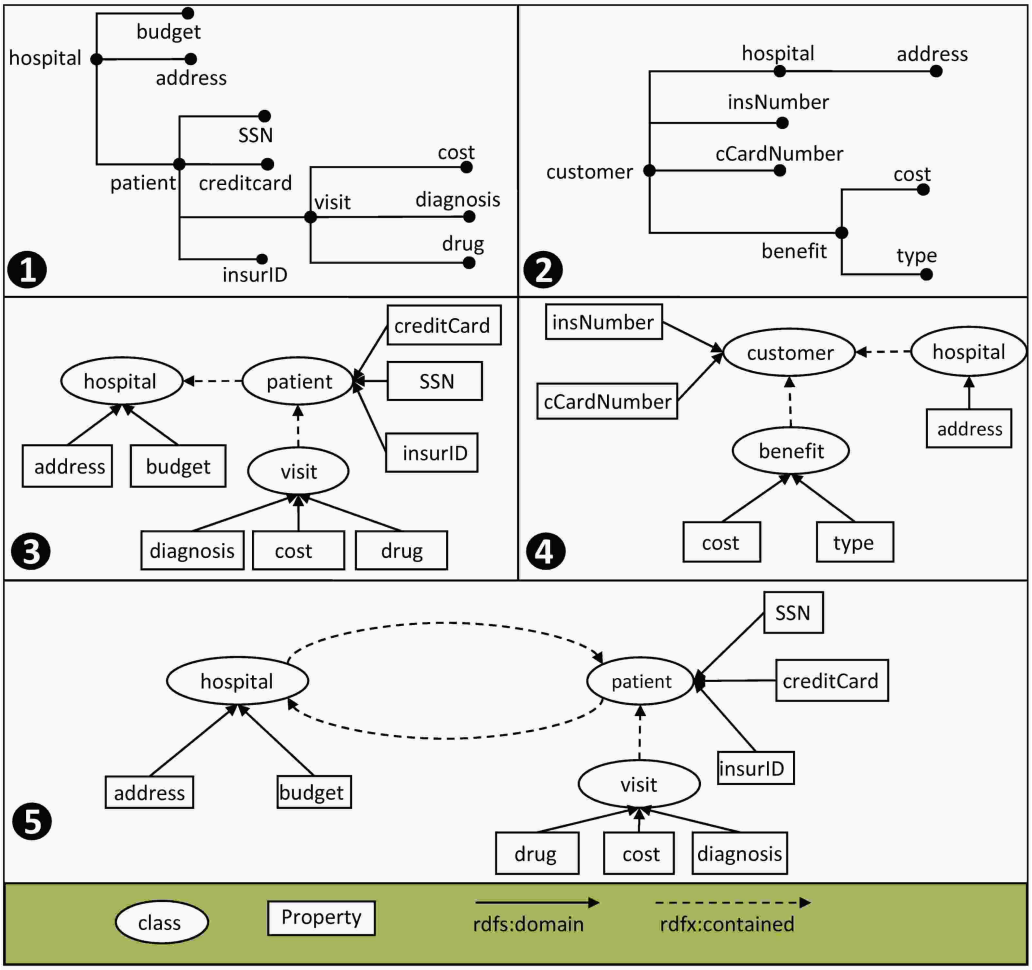


Figure 5.1: Local and Global schemas: 1. Hospital XML schema 2. Insurance XML schema 3. Hospital RDF schema 4. Insurance RDF schema 5. Global RDF schema.

the third element, o, is the *object* or *value* of the property. The subject of the triple is also called *domain* of the property and the object is called *range* of the property. We define a mapping function μ next.

Definition 7 *The mapping function μ maps an XML schema element to an RDF schema element. If v is a complex XML schema element, then $\mu(v)$ belongs to the set of RDFS classes. If v is a simple XML schema element or an attribute, then $\mu(v)$ belongs to the set of RDF properties.*

As shown in Figures 5.1.1 and 5.1.3, the complex XML schema element *patient* is mapped to the RDFS class *patient*, whereas the simple XML schema

element *creditcard* is mapped to the RDF property *creditCard*. As can be seen in Figures 5.1.3 and 5.1.4., the two structurally heterogeneous elements are now mapped to two different classes. A property called *rdfx:contained* is used to record the parent-child relationship between complex XML elements. The second step is that of merging the local RDF schemas into a global schema and it consists of: (1) merging of equivalent RDFS classes and RDF properties from the local sources into a single class or property on the global schema; (2) copying a class or property into the global RDF schema if an equivalent class or property does not exist. A possible global RDF schema is shown in Figure 5.1.5. Here the local classes *patient* and *customer* have been mapped to the global class *patient*.

5.1.2 Security Framework

In this section we discuss the process of mapping security levels associated with the elements of the local XML schemas to the global RDF schema triples. The local security policies are represented as local security lattices associated with both the XML and the RDF schema levels. Local security lattices are merged into a global security lattice representing the global security levels associated with the global RDF schema. We assume that the only action that is permitted on the local sources is the *read* action. The results can be extended also to the *write* action, but we assume that users can only write and change the values of the local sources they are associated with. The security of the interoperation systems must satisfy the following requirements:

- *Autonomy.* The local security policies must not be affected by the security policy of the global level.
- *Confidentiality.* Given a security clearance, if a schema element is not accessible locally before the integration, then it must not be accessible after integration.
- *Availability.* Given a security clearance, if a local schema element is accessible before integration, then it must continue to be accessible after integration.

We also make the following assumptions and observations on the local XML and RDF schemas: very sensitive portions of the local XML schemas might not be shared at all; the global level contains the RDF schema, but not the instances (which reside locally). The security levels on the local XML schema elements are used to restrict access to the corresponding XML instance elements.

Local Security Lattices

Definition 8 *A* security specification *on the XML schema tree is a pair* $[v, s]$ *where* v *is a node of the local XML schema and* s *is the security level associated with* v*. We denote the* set of security specifications *by* S_X*.*

We modify a previously proposed model to specify the security levels globally, which assigns security levels to RDFS triples based on RDF patterns [77]. Instead, we assign security levels to RDFS triples based on XML schema elements.

Definition 9 *A* security object *is a pair* $[t, s]$*, where* t *is an RDFS triple and* s *is the security level associated with* t*. We denote the* set of security objects of a local RDF schema *by* S_L*.*

We consider two kinds of RDF schema triples: *subject triples* and *subject-object triples*.

Definition 10 *A* subject triple *is an RDFS triple (s, p, o) where the subject s is a mapping* $\mu(v)$ *of an XML schema element v, and the predicate p and object o belong to the RDFS vocabulary. A* subject-object triple *is an RDFS triple (s, p, o) where the subject s and object o are two mappings* $\mu(u)$ *and* $\mu(v)$ *of two XML schema elements u and v which are in a parent-child or containment relationship, and the predicate p is either* rdfs:domain *or* rdfx:contained*.*

For example, *(hospital, rdf:type, rdfs:Class)* is a *subject triple* where only the subject *hospital* is mapped from an XML schema element. The triple *(creditCard, rdfs:domain, patient)* is a *subject-object triple* where the subject *creditCard* and the object *patient* are mapped from XML schema elements. Security levels assigned to *subject triples* will restrict access to information on single entities of the original XML schemas whereas in *subject-object triples* the two elements of the local XML schema may have different security levels. Accordingly, we define two security mappings that associate security specifications on the local XML schemas to security objects on the local RDF schemas.

Definition 11 *A* subject security mapping σ *maps a security specification in* S_X *of the form* $[v, s]$ *to a set of security objects in* S_L*, of the form* $[t, s]$*, such that (1)* t *is a subject triple; (2)* s *is the same security level for all security objects. There are, therefore, as many security objects as there are triples* t *that correspond to XML schema element* v*. A triple* t *can either correspond directly to an element* v *or can be classified by inference using RDFS entailment [77].*

For instance, consider the security specification [*SSN, adm*] in Figure 5.2.1. The subject security mapping σ maps that security specification to security object [*(SSN, rdf:type, rdfs:Class), adm*] in Figure 5.2.3 and to security object [*(SSN, rdf:type, rdfs:Resource), adm*] containing the entailed triple (because due to inheritance every *class* is also a *resource* in the RDFS model).

Definition 12 *A* subject-object security mapping κ *maps a pair of security specifications* $[v_1, s_1]$ *and* $[v_2, s_2]$ *in* S_X *to a security object* $[t, s]$ *in* S_L*, where* t *is a subject-object triple and the security level* s *is the least upper bound (LUB) of the security specifications levels* s_1 *and* s_2*.*

Every *subject-object triple* is assigned to the least upper bound (LUB) of the security levels of the corresponding XML schema elements. Instead, the *subject triples* are assigned to the security level of the corresponding XML schema element. For instance, consider the security specifications [*hospital, pub*] and [*budget, adm*] in Figure 5.2.1. where *hospital* and *budget* are in a parent-child relationship. The subject-object security mapping κ maps them to the security object [*(budget, rdfs:domain, hospital), adm*], if $LUB(pub, adm) = adm$. Figure 5.2 shows the mappings of the security specifications on the XML schemas to the security objects on the local RDFS triples.

Global Security Lattice generation

Next, we discuss the process of merging the local security lattices into a global security lattice representing the global security levels associated with the global RDF schema, and the classification of the global RDFS triples. The merging process can be carried out by an agreement among the security administrators of the local sources. Some local security levels from different sources may be merged in the global security lattice, while others may be just copied into it. Constraints on the orderings among security levels at the different local sources are used to define the global order. One requirement of the merging is that there are no cycles in the resulting partial order [54, 97]. The partial order $\preceq$ in the local sources must also be preserved in the global security lattice. Therefore, one or more local security levels can be merged into a global security level.

Definition 13 *The mapping function* θ *maps a local security level to a global security level. The mapping function* Θ *maps a set of local security levels,* L_i*, to a set of global security levels* $\Theta(L_i) = \{\theta(l) | \, l \in L_i\}$*.*

We show an example in Figure 5.3 in which the dotted lines represent the mappings defined by θ. The local levels *s-adm (secure administration)* and

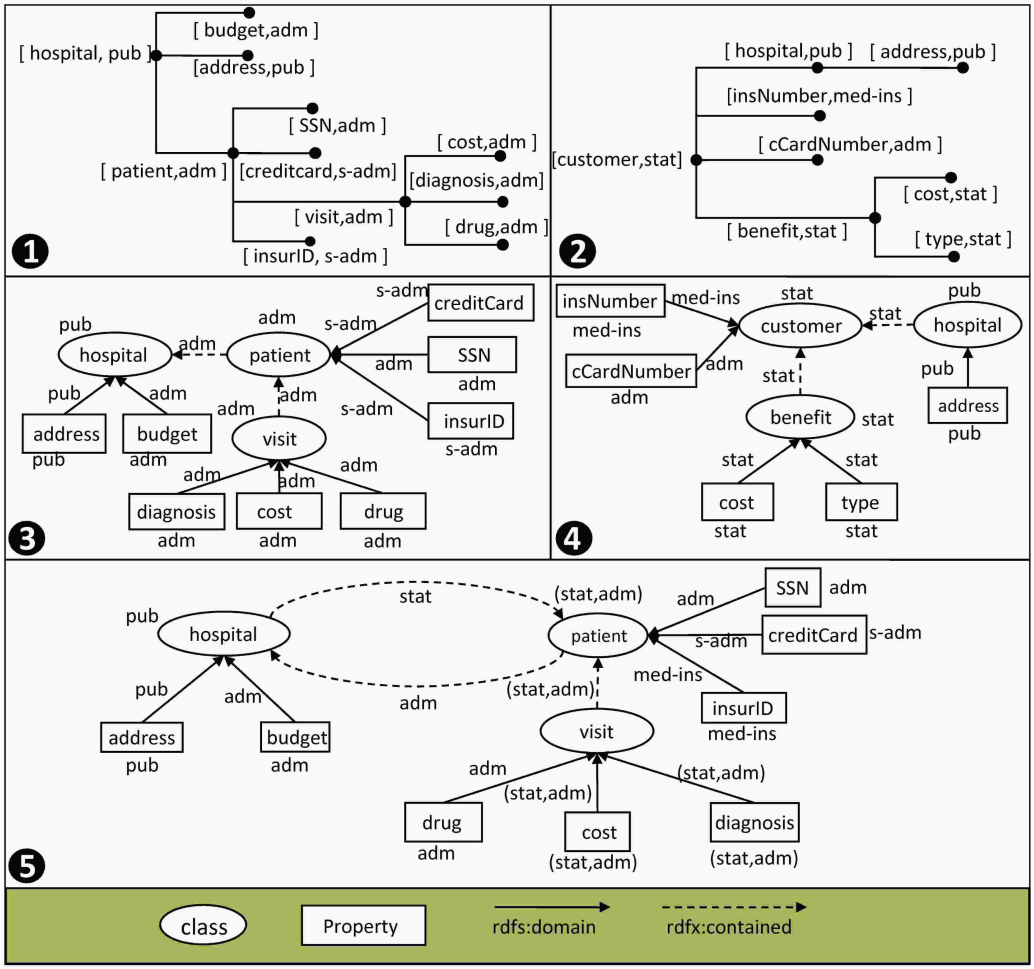


Figure 5.2: Security levels and mappings: 1. Hospital XML schema 2. Insurance XML schema 3. Hospital RDF schema 4. Insurance RDF schema 5. Global RDF schema.

adm (administration) are merged into the global level *s-adm* and $\preceq$ is preserved globally. The classification of the global triples is performed by exploiting the mappings between the triples of the local and of the global RDF schemas and the mappings between the local security levels and the global ones after the merging. A global triple will be assigned a security level by taking into account the security levels of the corresponding triples in the local sources. In the most general case, the local triples mapped to the same global triple will have local security levels mapped to different global security levels. Therefore, there can be more than one candidate security level for a global triple.

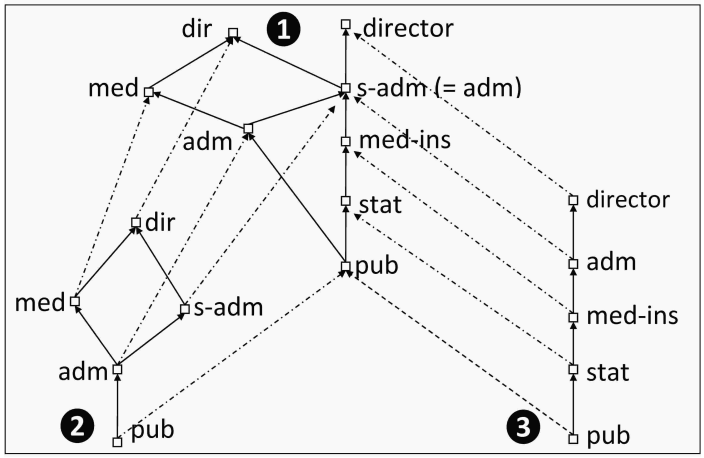


Figure 5.3: Security lattices: 1. Global security lattice 2. Hospital security lattice 3. Insurance security lattice.

Definition 14 *Let S be a subset of the global security levels. The source of S,* source(S)*, is the subset of S such that for each element s_i in source(S) there is no element s_j in S such that $s_j \preceq s_i$ in the graph induced by S. Each element s_i is called* minimal*.*

In Figure 5.3.1, $source(\{dir, med, med\text{-}ins\})$ is the set $\{med, med\text{-}ins\}$.

Definition 15 *Let S_G be the set of security objects of the global RDF schema and S_{Li} be a set of local security objects in S_L, where the triples in each security object in S_{Li} are mapped to the same global triple t_{g_i}. Let L_i be the set of local security levels of S_{Li}. The* global security mapping γ *maps each S_{Li} to a subset S_{Gi} of S_G, whose elements share that same triple but have as security level one of the security levels in source(S), where $S = \Theta(L_i)$. The cardinality of the set S_{Gi} is the same as the cardinality of source(S).*

For instance, consider two local triples *(cost, rdfs:domain, visit)* in Figure 5.2.3 and *(cost, rdfs:domain, benefit)* in Figure 5.2.4. that are mapped to the same global triple t_{g1} = *(cost, rdfs:domain, visit)* in Figure 5.2.5. The global security mapping γ maps the set S_1 formed by the two local security objects [*(cost, rdfs:domain, visit), adm*] and [*(cost, rdfs:domain, benefit), stat*] to the set S_{G1} formed by the global security objects [*(cost, rdfs:domain, visit), stat*] and [*(cost, rdfs:domain, visit), adm*], because $source(S_1) = \{adm, stat\}$.

Theorem *Assuming security autonomy after source integration, the local security mappings σ and κ and the global security mapping γ preserve data confidentiality and availability.*

Proof Sketch By means of the local security mappings σ and κ, the local security levels are mapped either to themselves (in the case of a subject triple) or to their least upper bound (in the case of a subject-object triple). Given two local security levels l_1 and l_2, we have $l_1, l_2 \preceq LUB(l_1, l_2)$. The global security mapping γ maps a set of local security objects to a set of global security objects where the global security levels are minimal. It may be that a global triple is associated with a global security level $g \preceq LUB(l_1, l_2)$, but due to the security autonomy of the local sources the local triple will remain classified at level $LUB(l_1, l_2)$. Therefore, if an XML schema element cannot be accessed before integration, it will continue to be inaccessible afterward, thus guaranteeing the confidentiality of the data.

Through the subject security mapping σ, the local security level remains the same, therefore the XML schema element remains available. Subject-object security mapping κ maps two security specifications to a security object, therefore the security level obtained may be more restrictive. This type of mapping deals with the security of the relationship between the subject and the object elements. Even if the relationship is restricted, they can always be accessed individually at the corresponding single triples' security levels. The global security levels obtained by the global security mapping γ are minimal because some local triples are classified at those minimal security levels. Therefore, the minimal global security levels guarantee the availability of the data.

5.1.3 Related Work

XML Access Control Models XML access control models have been the focus of recent research, including approaches in which the access control model is expressed in terms of tuples that specify who can access which schema element, what type of access is allowed, and how the access rights propagate on the XML tree [25, 48, 49].

RDF/S Access Control Models A method for transforming RDF graphs into trees so as to hide subtrees of a given node has been proposed [80]. Related work includes the work by Farkas and Jain [77] that has been mentioned in Section 5.1.2.

Secure Interoperation Models The approach by Pan *et al.* uses a mediator among database systems in an RBAC access control model and mappings between roles in different local sources [99]. In another approach, Candan *et al.* propose a secure interoperation model where a global mediator can

enforce global access control rules, or just be a conveyor of the information exchanged between the local sources [34]. Bonatti *et al.* propose the merging of sets of ordered security levels using a logic programming approach [31]. In other work, Dawson *et al.* propose a framework for secure interoperation between local applications mediated by a global application [54]. The work of Farkas *et al.* is the closest to ours [59]. However, they use a "top-down" approach in which they start from the RDF global schema, whereas we start from the XML sources. Another difference is that they use discretionary access rights, whereas we use multiple level security lattices.

5.1.4 Conclusions and Future Work

We have proposed a translation model for security levels from local XML schema sources to a global RDF schema. We follow a bottom-up approach and respect the principle of local autonomy in that local security policies continue to be valid. In the future, we will consider the implications of having specifications of security levels not only on the XML schema elements, but also on their instances. We will expand our approach to full XML schemas, including for example IDREF tags. We will also investigate how this approach can be generalized to other data representation models. Furthermore, we plan to incorporate our model into the MOMIS system [11].

5.2 Dynamic Role Assignment in Collaborative Environments: a Constraint and Attribute Based Security Framework

With the latest trends in collaborative environments, such as Web 2.0 and cooperative projects on grids, more and more resources are being shared by different groups and organizations in order to support common tasks. Depending on several factors such as the task, the participants, and data sensitivity, access to these shared resources needs to be controlled and enforced by security policies. The role-based access control (RBAC) model defines roles that have specific privileges on resources and decouples the identity of the users from the resources [105]. In the RBAC model and its variations, constraints can be placed for example on the associations of users with roles or of roles with permissions. When the number of users is high in comparison with the number of roles [3, 4], an automated way to grant permissions is desirable in order to eliminate the burden of manually assigning roles to users. The RBAC model is particularly suited to dynamic task-oriented environments due to its flexibility and policy-neutrality [98], which enables it to express a large range of policies.

We investigate a security framework for collaborative applications that relies on the RBAC model. Roles are pre-defined and organized in a hierarchy (partial order). However, we assume that users are not previously identified. Thus, the actions that they can perform are dynamically determined based on their own attribute values and on the values of the attributes associated with the resources. The user's attribute values can vary over time during a session (e.g., the user's location), thus enabling or disabling the user's roles.

We will focus on a scenario associated with the Olympic Games, where not only the venues directly associated with the Olympic Games (e.g., stadiums, gymnasiums) but also tourist attractions in the area (e.g., museums, parks) are resources of interest in our framework. Access to venues and specific places inside the venues depend on the users' types. For example, some spectators can only take part in the opening ceremony, whereas others can access all swimming events or all track and field events, depending on the tickets they have purchased. In addition to visitors, there are many organizations collaborating with one another and sharing information and services (including police forces, hosting companies, media, and sport organizations) who ultimately serve a large range of visitors as well as the competing athletes and their support teams.

Privileges granted to users depend not only on each particular organization but can also differ among members of the same organization. For

example, some members of the escort service for teams and athletes may be restricted to escort out of a specific venue but not out of other venues (a situation similar to taxi drivers in some cities, where a taxi that transports passengers from the city to the airport cannot subsequently pick up passengers at the airport).

Different people will have different privileges depending on their status. For example, members of the Olympic Committee, who have VIP status, will have reserved seating in all competitions, while top officials of the local organizing committee, who also enjoy VIP status, may have non-assigned seating. Police officers will be able to enter any area, but without seating privileges. Children or students under a certain age may be able to join tours of the Olympic Stadium for free, while other people will have to pay a fee. For security reasons access to the Olympic Village is restricted to few people besides the athletes and their immediate support teams: for example, employees and volunteers specifically assigned to work in that particular area.

In our approach, the roles of each different collaborating organization are structured in a dominance hierarchy where "higher" roles have all the privileges of "lower" roles. The roles associated with all the organizations can be represented as the union of the hierarchies of roles of the single organizations. Some of the roles have fixed and previously known sets of users, such as police, members of the local organizing committee, or the athletes. Other roles have a large number of possible users that cannot be known a priori, for instance journalists, volunteers, and visitors. In this case, constraints on user attribute values can be used to assign the correct role to each user, based on the values of different attributes (e.g., status, credentials, location, organization). Roles are assigned to users depending on the actual values of their attributes (e.g., VIP, journalist, main stadium, NBC). Constraint values in our framework form partial orders and determine the association of actions with the resources and of users with roles. Therefore, users' actions are dynamically determined based on their own attribute values and on the values of the attributes associated with the resources.

We have designed and implemented a prototype of our access control framework using semantic web technologies. The roles and other entities defined in the RBAC model are represented using the OWL 1.1 language [73], which is a standard language based on Description Logic. Based on previous work, we use two ontologies: the first ontology describes the domain and the second describes the RBAC entities and is partly derived from the first [38]. Reasoning is performed using the Pellet reasoner [39] and is used to implement several functions, such as user to role assignment, separation of duty constraints, symmetry, and class equivalence.

Our model shares some similarities with other approaches including RB-

RBAC [3, 4], GEO-RBAC [51, 26], and ROWLBAC [60]. A notable difference is that it has been fully implemented, while the other approaches have not. Therefore, we have leveraged the expressiveness of an actual reasoning mechanism. However, all the other approaches also propose some sort of reasoning. In particular, RB-RBAC uses rules to determine hierarchical roles starting from a partial order of constraints, while GEO-RBAC uses propagation of constraints along the role hierarchy. We extend RB-BAC by starting from individual partial orders of attribute constraints and then unifying them. In comparison with GEO-RBAC, our framework is more general in that it targets all types of constraints, not only spatial constraints. We also consider resource attribute constraints, whose satisfaction enables or disables the privileges defined on the resources. ROWLBAC, even if not implemented, proposes reasoning as performed by OWL. The most similar approach to our current approach is our former approach, which was also fully implemented using semantic web technologies [38]. However, in that approach, we used a simpler constraint framework and did not explicitly consider spatial constraints.

The chapter is organized as follows. In Section 5.2.1, we present the security model and in particular, the attribute constraints arranged in partially ordered sets and their correspondence with the roles. In Section 5.2.2, we describe the different types of entailment that our model supports and give examples of some rules of Description Logic that can be used to express security policies. We also show the process by which users are assigned the correct roles by taking into account constraints. In Section 5.2.3 we describe the implementation of the access control model including the design choices we have made. Related work is mentioned in Section 5.2.4 and conclusions and future work are discussed in Section 5.2.5.

5.2.1 Security Model

In this section, we describe the different components that make up our framework. We start by extending our scenario and then we describe the different components that are present in our model. Those components, modeled as classes and as constraints, extend the usual RBAC components.

Scenario

In our scenario, which is a much simplified version of the kind of considerations needed for the Olympic Games, there are four collaborating organizations: *Media*, *Sports*, *HostingCity*, and *Visitors*. The organizations share the same resources and each of them can be modeled separately. The first

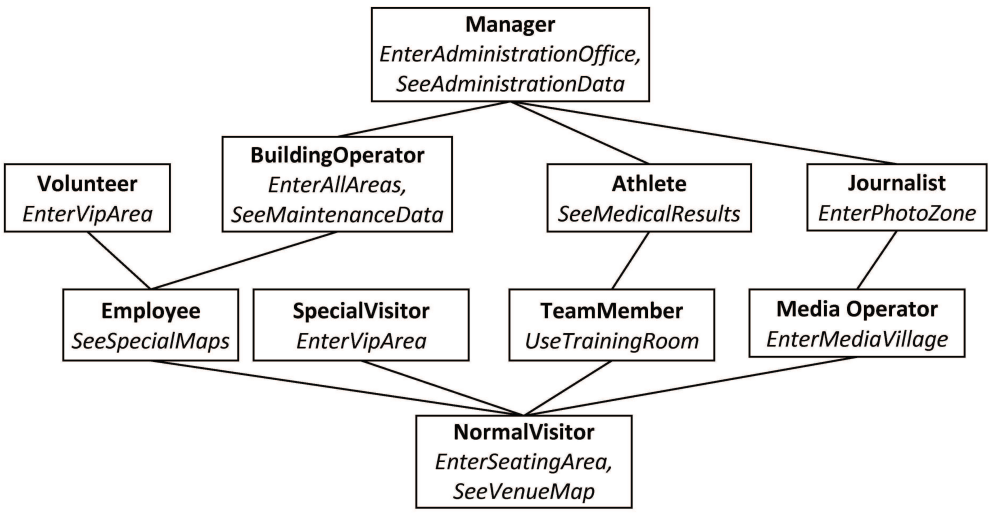


Figure 5.4: Roles and privileges for the Olympic Games organizations.

organization, *Media*, comprises *MediaOperator* and *Journalist*, where *MediaOperator* has privilege *EnterMediaVillage*, to enter a resource that is reserved for media operators and *Journalist* inherits the privileges of *MediaOperator*. *Journalist* has one additional privilege, *EnterPhotoZone*, to enter a special area that is particularly suitable for taking close up pictures of the athletes.

The second organization, *Sports*, comprises *TeamMember* and *Athlete*. The third organization, *HostingCity*, comprises people who take care of all local organizational tasks. The fourth organization, *Visitors*, comprises all the different people who attend the Olympic Games. We model them as an organization so that we can deal with them similarly to the other groups of people. Visitors can have different degrees of importance, spanning from "VIP" (e.g., members of the Olympic Committee) to "normal" (e.g., common spectators). These different degrees of importance correspond to different privileges. Privileges and the overall role hierarchy of our collaboration scenario is shown in Figure 5.4. The roles that carry more privileges are shown higher in the hierarchy: for example, the *Manager* role contains all the privileges of the roles that are its descendants in addition to its own, while the role *Volunteer*, which is not a descendant of *Manager*, comprises all the roles of *Employee* and of *NormalVisitor* in addition to its own.

We consider that each organization determines how the roles are assigned to their users depending on their attribute values. For instance, in our scenario, visitors have attributes *Importance, Age*, and *Location*. The *Visitor* organization assigns the role *SpecialVisitor* depending on the values of these attributes, for instance if somebody's *Importance* attribute is equal to *VIP*,

Age is greater than 21, and *Location* is inside *VIPArea*. In our model, *Location* is both an attribute of users, which is used to associate roles with users, and of resources.

The final role hierarchy shown in Figure 5.4 is derived using simple inference on a description of the organizations and resources using ontologies. The security administrator checks and validates the inference results. Further description of this step will be given in Section 5.2.2.

Framework components

In this section, we explain the conceptual components of our system.

Resource class. This class represents the entities on which different actions are or not allowed (e.g., *SeatingArea*). Resources have associated attributes (e.g., *Capacity* of the Olympic Stadium).

Action class. This class represents the actions that can be performed by users on the resources (e.g., *Enter*).

Privilege class. Objects of this class are pairs $\langle$*Action, Resource*$\rangle$. For example, the privilege $\langle$*Enter, SeatingArea*$\rangle$ allows some users to enter the seating area.

Privilege attribute constraints. These constraints are pairs $\langle p,a \rangle$, where p is a privilege (e.g., $\langle$*Enter, SeatingArea*$\rangle$) and a is a pair $\langle$*attribute,attributeconstraint*$\rangle$ (e.g., $\langle$*isOpen,* $= true\rangle$) associated with the resource that is part of the privilege (in this case, *SeatingArea*). Attribute constraints are recursively defined as follows:

```
attributeconstraint ::= (attributeconstraint)
                        | RELATIONALOPERATOR constant
                        | NEGATION (attributeconstraint)
                        | attributeconstraint BINARYBOOLEANOPERATOR
                          attributeconstraint
```

where a constant can be of different types (e.g., string, number, Boolean, area) and therefore the relational operator (e.g., $=$, $\leq$)) is polymorphic in that it is able to compare different types (for example, $\leq$, when used for areas will be equivalent to set containment, $\subseteq$). Examples of attribute constraints include: $\geq 10 \wedge \leq 18$, and $\neg(\geq 10 \wedge \leq 18)$ and $\leq$ *SeatedArea*. The definition of attribute constraint can be further extended.

Role class. This class is a placeholder for all the roles that are defined. Conceptually, a role is a set of privileges. Roles are assigned to users via sessions.

Role attribute constraints. These constraints are pairs $\langle r,a \rangle$, where r is a role (e.g., *SpecialVisitor*), and a is an attribute pair $\langle$*attribute, attributeconstraint*$\rangle$

(e.g., $\langle Importance, = VIP \rangle$), where *attributeconstraint* is defined as previously. There is a many-to-many relationship between roles and attribute pairs. The role *SpecialVisitor* is assigned to a user if the attribute *Importance* has value $= VIP$. When a role attribute constraint refers to spatial attributes, for example, $\langle Journalist, \langle Location, \leq MediaVillage \rangle \rangle$ the role $Journalist$ is activated when the user is in the $MediaVillage$ (provided that other attribute pairs, if any, are also satisfied).

Session. A user is assigned a session upon entering the system (e.g., $John\_680481$). A session is owned by a single user and has a set of roles associated with it. We assume that attribute values associated with resources are not allowed to change during a session. However, attribute values associated with users can change. For example, the location of a user can change during a session, therefore the corresponding attribute *Location* value changes.

Attribute constraints

As presented in Section 5.2.1, role attribute constraints denote a many-to-many relationship between roles and attribute pairs. For a role to be assigned to a user, the user's attribute values must satisfy the attribute constraints. As previously described, the constraints can be expressed in different ways. For instance, a constraint on *Age* can be expressed as a range, for example, ≥ 21, or a constraint on *Importance* can be expressed as a single value, for example $= VIP$. The former constraint would have to be satisfied for someone to have the privilege to enter a bar, whereas the second one would have to be satisfied for someone to access a VIP area.

It is possible to establish a partial order among attribute constraints in the case where an attribute constraint dominates another one. For example, for attribute *age*, ≥ 21 dominates ≥ 18 as someone who is older than 21 is also older than 18. Likewise, for attribute *importance*, $= VIP$ should dominate $= normal$. In our approach, we interpret the dominance relationship between attribute constraints as a satisfiability relationship. Thus, to say that a constraint a dominates a constraint b, written $b \preceq a$ is tantamount to saying that when a is satisfied, b is also satisfied.

Examples of partial orders are shown in Figure 5.5. Figure 5.5.1 shows the constraint for user attribute *Age*. The constraint B_3 is dominated by the constraint B_2, and the constraint B_2 is dominated by the constraint B_1 $((\geq 5) \preceq (\geq 18) \preceq (\geq 21))$. Therefore, if the constraint B_1 is satisfied, then the constraints B_2 and B_3 are also satisfied. Figure 5.5.2 shows the constraint for user attribute *Location*, that is, if the coordinates of a user fall inside one of the regions, then the user is located inside the region. In this case, the dominance relationship represents the spatial containment between

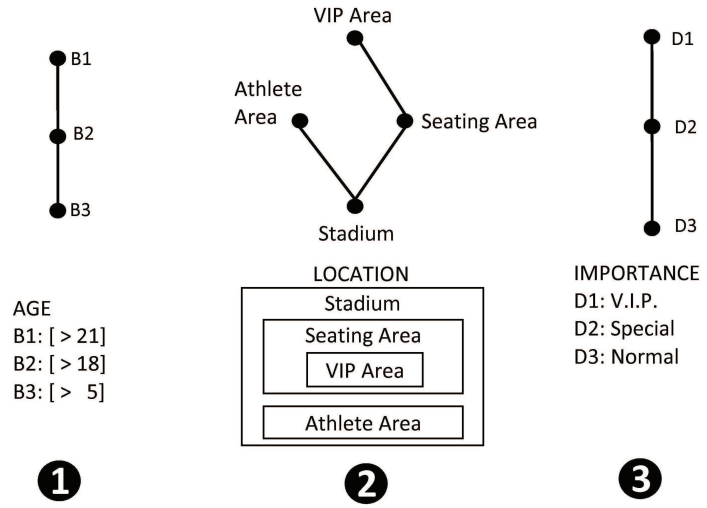


Figure 5.5: Partial orders: 1. *Age* partial order 2. *Location* partial order 3. *Importance* partial order.

the polygons. If the constraint $\leq VIPArea$ is satisfied, meaning if the user is inside location *VIPArea*, then the constraints $\leq SeatingArea$ and $\leq Stadium$ are also satisfied ($(\leq Stadium) \preceq (\leq SeatingArea) \preceq (\leq VIPArea)$). If the constraint $\leq AthleteArea$ is satisfied, only the constraint $\leq Stadium$ is also satisfied ($(\leq Stadium) \preceq (\leq AthleteArea)$). Figure 5.5.3 shows the constraint for attribute $Importance$. The constraint D_3 is dominated by D_2, which is in turn dominated by D_1 ($D_3 \preceq D_2 \preceq D_1$). Therefore, if the constraint D_1 is satisfied by the user $Importance$ value, then the constraints D_2 and D_3 are also satisfied.

We argue that in a scenario with different collaborating organizations, each having a different role hierarchy, the definition of partial orders of constraints can play an important role. Each organization will have its attributes and respective constraints. However, some of the attributes may be the same, but with different constraints on them. For instance, with respect to Figure 5.5.1 it is not difficult to imagine different constraints on the *Age* attribute. If these organizations share their role hierarchies, then they would also share their role attribute constraints. In the next subsection, we discuss the integration of different partially ordered sets of role attribute constraints into one partially ordered set.

Age	Location	Importance	Role	Privilege
>=21	<= VIP Area	= VIP	Journalist	Enter, reserved best seat
>=21	<= VIP Area	= Special	MediaOperator	Enter, reserved seat
>=18	<= Seating Area	= Normal	NormalVisitor	Enter, seat anywhere

Figure 5.6: Attribute constraints and roles.

Role-constraints partial order

A role r can be associated with a tuple A of user attribute constraints over distinct attributes. The pair $\langle r, A\rangle$ represents the constraints that must be satisfied to activate the role. Roles are assigned to users, based on the constraints that the user's attribute values satisfy. In Figure 5.6 we show three roles and their associate attribute constraints, whose partial orders are shown in Figure 5.5. The role *Journalist* is the dominant role represented in the table. Also, each attribute constraint of *Journalist* dominates the corresponding attribute constraint of the other roles, that is, the sets of attribute constraints represented in each row are in *componentwise order* [52]. This type of order can be defined on the tuples of the Cartesian product of partially ordered sets. A tuple of the Cartesian product (e.g., $\langle \geq 21, \leq VIPArea, = VIP\rangle$) dominates another tuple (e.g., $\langle \geq 21, \leq VIPArea, = Special\rangle$) if each element of the first tuple dominates the corresponding element of the second tuple (that is, $\geq 21 \preceq \geq 21$, $\leq VIPArea \preceq \leq VIPArea$, and $= Special \preceq = VIP$).

The cardinality of the Cartesian product of the partially ordered sets of constraints can be much higher than the cardinality of the set of roles. For instance, in the example of Figure 5.5, there are $3 * 4 * 3 = 36$ possible combinations of the different attribute constraints, but likely fewer roles. Therefore, a user may satisfy a set of attribute constraints that does not correspond to any role. For instance, in Figure 5.6, a user may satisfy the constraints $Age \geq 18$, $Location \leq SeatingArea$, and $Importance = VIP$, which does not correspond to any role. Nonetheless, the user should be assigned the most dominant role possible, that is, *NormalVisitor* [4].

We will discuss later in the implementation part how this feature has been implemented in our framework.

Transformation functions

A transformation function can be defined on an attribute to associate the attribute values defined in a certain domain with values on a different domain.

For example, given the integer attribute *Age*, the transformation function *child* : *Age* → *Boolean* associates values greater than 5 to the Boolean *false* and values up to 5 to the Boolean *true*. Transformation functions are total functions. The domain of a transformation function can be the Cartesian product of several attribute domains, associating a set of attributes values with a single attribute value. As in the GEO-RBAC model [26], an example of a transformation function is a location transformation that associates the geographic coordinates of a user with a logical location (e.g., *OlympicStadiumArea*).

With transformation functions applied to a set of user attributes, the constraints can be defined on the target of the transformation function. Applying transformation functions to the user attributes can help in simplifying the computation of the constraints and in preserving privacy [50]. Indeed, if a transformation function is applied on an attribute, only the transformed values (logical values) will be computed over the constraints. The real values will be in a certain sense masked. Moreover, through transformation functions, it is possible to map a set of constraints defined on several attributes into simpler constraints, for example into constraints on Boolean values. In our framework we have implemented only the location transformation function.

5.2.2 Reasoning

In the last few years there has been a good amount of research in modeling security models for dynamic environments with the use of Description Logic [109, 38, 60]. Toninelli *et al.* [109] use the OWL language and Logic Programming to model the security policies of a pervasive computing environment. Finin *et al.* consider two approaches for modeling the RBAC model with OWL [60]. Cirio *et al.*, whose work we continue, leverage semantic web technologies to help the security administrator define security policies [38].

The expressiveness of OWL allows for a rich representation of rules and relationships between domain entities and for expressing policies. In particular, it is possible to express:

- Equivalence or disjointness between classes of objects. For instance, it is possible to say that two classes are equivalent and therefore they inherit the properties of each other, or disjoint, therefore an object cannot be an instance of both classes. We use the disjointness feature to implement separation of duty constraints. For instance, it is possible to say that an object belonging to the *TaxiDriver* class cannot belong to the *Police* class.

- Subclass hierarchies, with multiple inheritance. The subclass inherits the properties of the superclass. We use this feature in two ways: 1) to implement the role constraint hierarchy; 2) to create sets of classes in order to specify a common policy for all of them. The classes of a set are placed under a superclass, to which privileges are attached. Through inheritance, the set of the subclasses inherits the privileges attached to the superclass.

- Properties can be of two types: datatype properties and object properties. We use datatype properties to model the constraints and object properties to assign the privileges to the roles. Object properties, in turn, can also be divided into symmetric, anti-symmetric, transitive, anti-transitive, functional and inverse functional properties.

- New classes can be combined from existing classes using intersection, union, and negation.

- Axioms can be written to express policies. For example, to express the fact that some members of the escort service for teams and athletes may be restricted to escort out of a specific venue but not out of other venues.

We use two types of ontologies in our model: the domain ontology and the RBAC ontology.

The *domain ontology* represents the relationships that hold between the entities of the domain. It can be an existing ontology that describes a particular organization. The domain ontology can contain any of the OWL constructs described above. We give an example of a portion of our domain ontology in Figure 5.7. In the figure, we show the ontology classes *Manager, BuildingOperator, Employee*, that are in a class/subclass relationship. Some of the relationships between the different classes are represented by object properties, such as *works*. The *RBAC ontology* (see Figure 5.8) has four main classes that represent the main concepts of the RBAC model: *Roles, Privileges, Actions,* and *Resources* [38]. These main classes are related to one another by object properties. For example the class *Role* has a relationship named *grants* with the class *Privilege*. These properties are useful during reasoning, because they guide the reasoner in classifying each of the ontology concepts under the appropriate class of the RBAC ontology. Two classification tasks are performed: of the user session and of the classes of the domain ontology into classes of the RBAC ontology. A user session is represented as an instance of the class *Thing* and its attribute values are used by the reasoner to classify the user session in the correct role. The classification

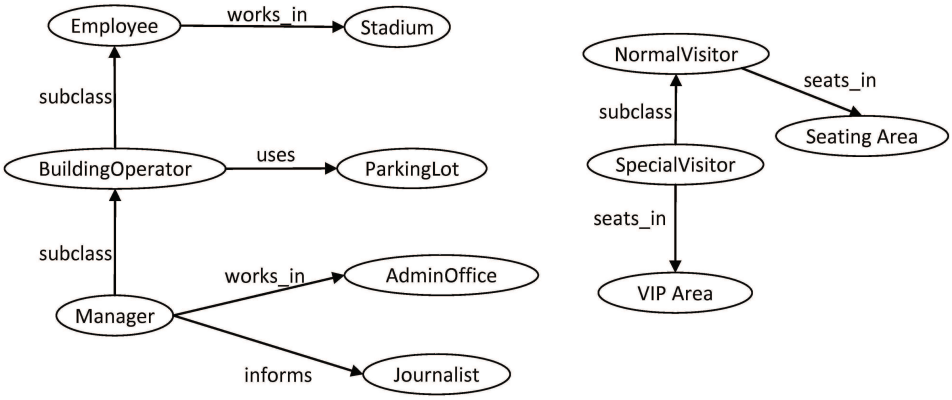


Figure 5.7: Domain ontology (portion).

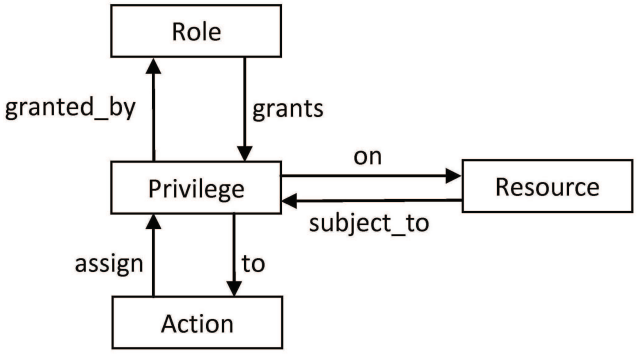


Figure 5.8: RBAC ontology.

of the domain ontology can be performed either by the security administrator or by the the DL reasoner. The latter will classify the different classes of the domain ontology under the classes of the RBAC ontology, following predefined axioms. The axioms are specifications of relationships that must hold between resources [38]. For example, the following rule classifies entities of the domain ontology as subclasses of the *Action* class:

$$\exists assign.Privilege \sqcap \neg\{Resource, Privilege, Role\}$$

where *assign* is a property in the domain ontology, therefore given the assertion *assign(Enter, FreeEnter)*, the reasoner classifies *Enter* as a subclass of *Action*.

5.2.3 Prototype

We implemented the security model described in Section 5.2.1 relying on semantic web technologies. In particular, the access control model and the features of the application domain are modeled using OWL-DL ontologies. The inference capabilities supported by the OWL-DL language enable the association of the ontology with the Pellet reasoner to perform the classification and reasoning tasks described in Section 5.2.2. We used Protege 4.0 to write the ontologies, and the Jena API, as an interface to the ontologies. We used the OWL 1.1 language for complex user-defined data types by means of the new *DataRange* constructors. The Pellet reasoner 1.5.2. supports reasoning on the new constructors. In what follows, we describe the classes that we used in the domain ontologies and in the RBAC ontology.

Domain ontology

In the domain ontology, the entities of the domain are described with OWL classes, data type, and object properties. A figure of a portion of our domain ontology was shown in the previous section.

As mentioned in Section 5.2.2, the security administrator defines privileges in the domain ontology. Conceptually, the privileges are pairs of *actions* and *resources*. From a practical point of view, this means augmenting the domain ontology by adding new classes to represent the privileges and actions unless they are already in the domain ontology. The security administrator also creates relationships between the classes of the domain ontology and the new added classes. In Figure 5.9, we show a portion of this process. The figure has three parts. The RBAC ontology is shown at the top. In the beginning, this is a very simple ontology. The domain ontology is shown on the left and on the right is the ontology that specifies the privileges and actions.

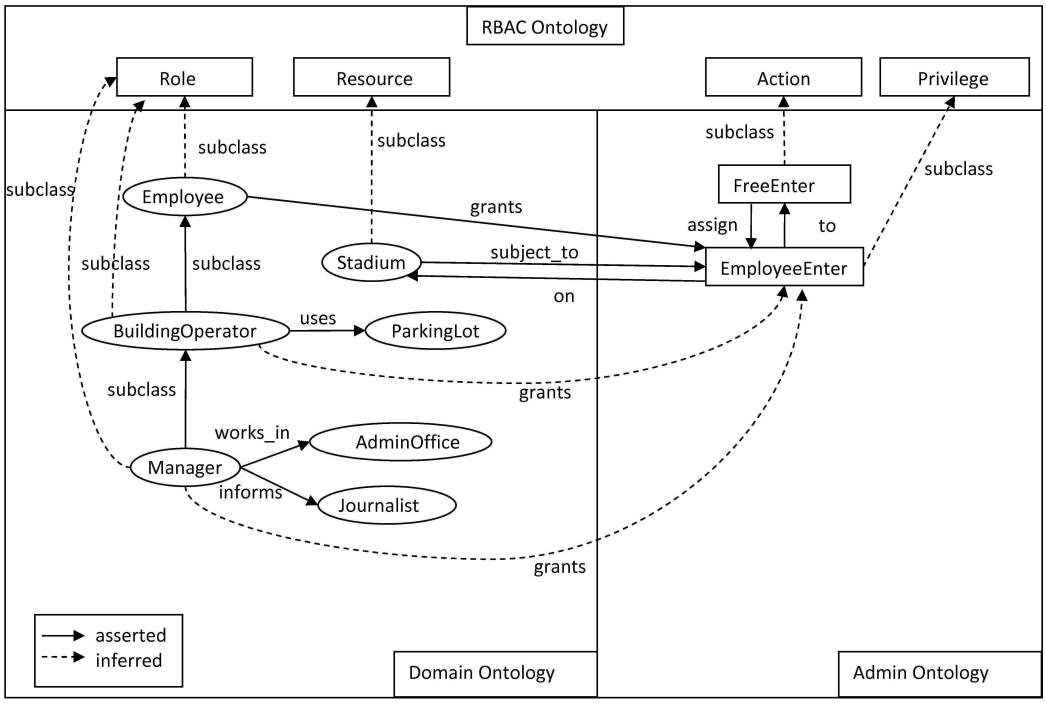


Figure 5.9: Domain and RBAC ontology.

The latter can be created by the security administrator or be an existing specification of privileges and actions.

In our example, the security administrator creates two OWL classes: *EmployeeEnter*, to represent a privilege, and *FreeEnter*, to represent the associated action. Object property *to* connects the privilege with its action. Privilege *EmployeeEnter* is associated through the object property *on* to the class *Stadium* in the domain ontology. The last object property that is added is the *grants* property that connects a class of the domain ontology, *Employee*, to the privilege *EmployeeEnter* class. As mentioned in Section 5.2.2, the reasoner uses the object properties *grants, on, to* to classify each domain ontology class under the correct RBAC ontology class. For instance, *Employee* is classified as a subclass of *Role* and the reasoner places under *Role* the subclasses of *Employee* as well. In this way, the RBAC ontology is extended with all the classes of the domain ontology. The different roles are associated to their privileges through the *grants* object property.

RBAC ontology

We describe now the additions to the RBAC ontology after the reasoning process. The privileges and actions remain the same as before the reasoning process. Next, we show how we model the attribute constraints on the user and resource attributes and how the ontology is used to assign a session to its roles.

- ***Constraints***. As mentioned in Section 5.2.1 we have attribute, attribute constraint pairs such as (*Importance, =VIP*) or (*Age,* $\geq$ 21). Since the constraints are always used in connection with resources or roles (that is, they cannot exist by themselves) there are two steps in modeling them:

  (1) Declaration of the attribute as an OWL data type property and definition of its domain. The domain is the union of the role classes to which the constraint is associated. The range is the XML data type to which the constraint value belongs. We have considered only string and integer data types for now. For instance, to model the constraints on the *Importance* attribute, we first declare a data type property named *Importance*, whose domain is the union of all the roles that have *Importance* as a constraint, e.g., the set {*SpecialVisitor, NormalVisitor*}. We declare the range of *Importance* to be the string data type.

  (2) Restriction of the values that the attribute can have inside the classes that represent roles or resources. For example, in the class for role *SpecialVisitor*, we restrict property *Importance* to assume only value *special*.

- ***RoleConstraint class***. As was mentioned in Section 5.2.1, the RoleConstraint represents a role and its constraints. We model every *RoleConstraint* as an OWL class. The name of the *RoleConstraint* class is the same as the name of the role, for instance, *SpecialVisitor*. The value of the attribute *Importance* is restricted to assume only the value *special* for the class *SpecialVisitor*. In other words, we are saying that the class *SpecialVisitor* is the class of all objects, whose *Importance* attribute has value *special*. The OWL code for the SpecialVisitor class is shown in Figure 5.10.

- ***Session***. At runtime, we add sessions as instances of the OWL class *Thing*, which is the superclass of all the classes of the domain. These instances are augmented with the attributes and values available from the user. The attributes and values of the instance guide the reasoner

```
<owl:Class rdf:about="#SpecialVisitor">
    <owl:equivalentClass>
      <owl:Class>
        <owl:intersectionOf rdf:parseType="Collection">
          <owl:Restriction>
            <owl:onProperty rdf:resource="#hasAge"/>
            <owl:hasValue>&gt;40</owl:hasValue>
          </owl:Restriction>
          <owl:Restriction>
            <owl:onProperty rdf:resource="#Importance"/>
            <owl:hasValue>VIP</owl:hasValue>
          </owl:Restriction>
        </owl:intersectionOf>
      </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

Figure 5.10: SpecialVisitor role constraint.

in the classification. For instance, in Figure 5.11 we show an instance of the user session with two attributes *Importance, Corporation* and values *special, HostingCity*, that is classified by the reasoner under the *RoleConstraint* class *Volunteer*.

- ***Resource constraints***. With constraints on resource attributes, we have to be able to deal with individual instances, and not with classes of objects anymore. Since each subclass of the class *Resource* can have different instances with different attribute values, we have to identify at instance level the resources that satisfy the constraints. If such resources exist then we can associate them with the instance of the user session. This association happens after the instance of the user session has been classified under a role constraint. OWL-DL does not allow for the specification of conditions about actual instances to identify the resources whose attributes satisfy the constraints. Therefore, we use SPARQL queries to verify that such resources exist [38].

Transformation functions

We have implemented the transformation function for the location attribute using the Google Maps API, which allows to define named areas on the map and symbols to represent people. The symbols can be moved around on the map to simulate the movement of people. If a symbol is inside one of the areas, the API returns the area name, which is used as the *Location* attribute of the user. The transformation functions serve also another purpose in

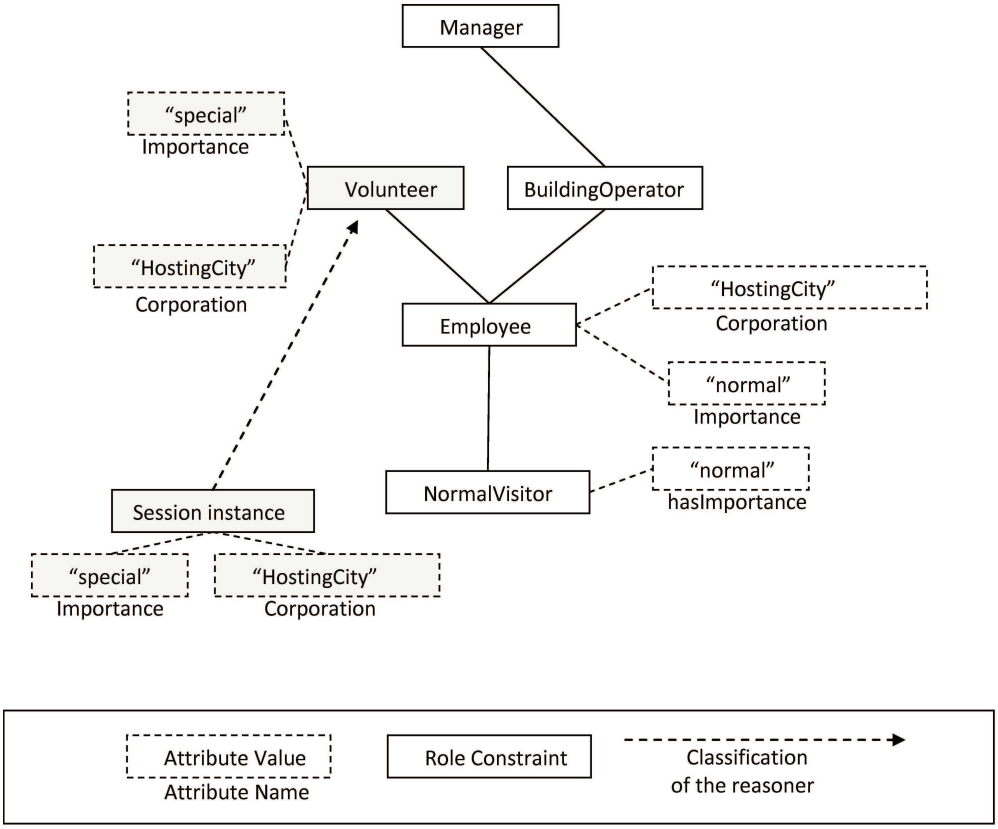


Figure 5.11: User session classification.

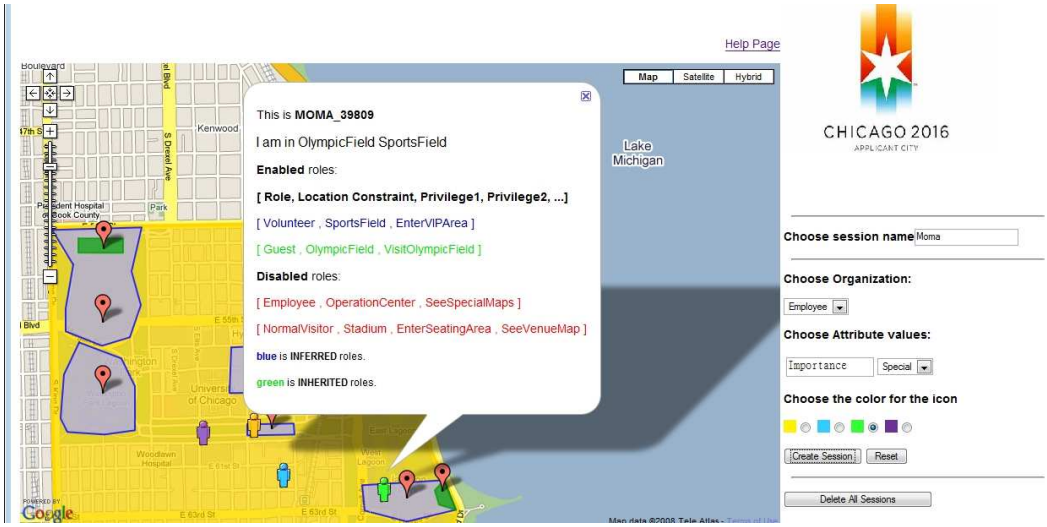


Figure 5.12: User interface.

masking from the real attribute values of the user. The access decision is performed on the transformed values and not on the real ones, increasing in this way the privacy of the user. For instance, the location attribute values on which the access decision is made do not show the real coordinates of a user, but a larger area. The location privacy of the user is thus increased [26]. Other transformation functions will be implemented in the future.

Graphical user interface

For the Olympic Games scenario, we implemented a user interface to illustrate our framework, as shown in Figure 5.12. It is composed of two parts, the map on the left and a form to retrieve attribute values when new sessions are created on the right. We have defined eight different areas in the map, associated with different values for the *Location* attribute. The form is used to enter attributes and their values. First, the attribute *Organization* is entered and next a pull down menu allows to choose another attribute for which a value will be entered.

Each session is represented by an icon displaying a person. When a session is created, a unique identifier is appended to the session name. The icon can be dragged and dropped in the map, thus changing the location attribute. The other values of the attributes can also be changed and the session attributes updated. Depending on the values of the attributes, the roles in the session may change. Each time the icon is dropped, the dialog window, which can be seen in the figure, is used to show the enabled and

disabled roles associated with that person and the privileges associated with that role.

Client server architecture

The framework has been implemented in a four-tier client-server architecture:

- **Tier 1: Application web page**. It has been developed with *JavaScript* technology and integrates with the Google Maps API, which also runs JavaScript.
- **Tier 2: Java™Applet Program**. It is downloaded from the server side and runs on the user's browser. It is responsible for handling the network traffic with the server.
- **Tier 3: Server Side Java™Program**. It is essentially a network server program, and it is responsible for network server functions, loading the ontology files, and interpreting and processing user requests.
- **Tier 4: Ontologies**. Ontologies are stored in this tier and modeled and maintained independently of the rest of the application.

5.2.4 Related Work

Geo-RBAC proposes a model for associating roles with logical location [26, 51]. Logical locations are regions of space defined by real world coordinates and a user can only assume roles that are associated with the location the user is in. In our model, location can be expressed as an attribute of the user along with other attributes, whose values determine the possible roles.

The Proteus system is intended for pervasive computing environments [109]. In Proteus, contexts are defined as intermediaries between entities and operations that they can perform on resources. Contexts are created by data sensed from the environment and reasoning is used to activate permissions on specific resources. Contexts can also inherit constraints from each other. However, Proteus is not role-based.

Kulkarni and Tripathi [83] devise a context-aware access control model. Constraints are defined on different entities of the model, for instance, resources and user attributes. Users can activate personalized permissions in addition to their roles, thus having a somewhat dynamic Role-Permission assignment. Role revocation is also supported, when values of the user attributes no longer satisfy the constraints. Attribute constraints are not arranged in lattices.

ROWLBAC proposes modeling RBAC with OWL [60]. Two different approaches for modeling roles are shown, one where roles are represented as classes and another one where roles are represented as instances. Attribute constraints on role assignments are not modeled, however, and there is no associated system.

RB-RBAC (Rule-Based RBAC) shares some similarity with our approach in that a hierarchy of constraints is mapped to a hierarchy of roles [3, 4]. The rules that associate attributes to roles are arranged in a hierarchy of seniority. When a senior rule is satisfied, the junior rules are automatically satisfied and all the roles produced by the senior rule and the junior ones are assigned to the user. Several other aspects are also considered, including the concept of role hierarchies that are induced by rules. However, they consider just one hierarchy of constraints.

5.2.5 Conclusions

The main contributions are summarized as follows:

- We decouple the constraints on the attributes of users from the roles and investigate the relations between hierarchies of attribute constraints and of the roles. Likewise, we decouple the constraints on the resources from their privileges. This simplifies the process of reasoning about users, resources, roles, and privileges.

- We consider dynamic attributes for users, whose values can vary during the same user session. An example includes location, though we offer a unified approach to any attribute type.

- Our model is expressive enough to capture hierarchies both of constraints and of roles and the associated inheritance reasoning as well as reasoning to combine constraints and to infer roles and user sessions.

- We have implemented our framework by exploring the capabilities of semantic web technologies and namely of OWL 1.1 [73] to model our framework and the domain, and to perform reasoning using the Pellet reasoner [39].

- We have adopted a client-server architecture and implemented a user interface whose purpose it twofold: (1) to offer a visual explanation of the underlying reasoning by displaying roles and their associations with users (e.g., as the user's locations vary); (2) to enable monitoring of the users that are involved in a collaborative application. Our interface,

which uses the Google Maps API, is particularly suited to collaborative applications where the users' geospatial location is of interest.

Future work includes:

- Adding expressiveness to our framework by allowing other types of constraints, namely temporal [79] or more complex constraints. In addition, further exploration of the consequences of component wise order (or lack thereof) and of the implementation of transformation functions for attributes other than location can be undertaken.
- Investigating reasoning, conflict resolution, and other aspects of merging ontologies of constraints and roles.
- Considering other privacy aspects, in particular when revealing to other organizations the structure of one's own. Work in privacy-preserving ontology matching [106, 46] needs to be investigated in our particular context.
- Designing a framework for the evaluation of dynamic constraint approaches that will take into account security metrics and the complexity of the evaluation [26] as well as the efficiency of the implementation using semantic web languages and reasoning [60].

Chapter 6

Conclusions

6.1 Key Contributions

This thesis investigated the issue of Query Management in Data Integration Systems, taking into account several problems that have to be faced during the query processing phase. The achieved goals of the thesis have been the study, analysis and proposal of techniques for effectively querying Data Integration Systems. Several software prototypes have been implemented to demonstrate the effectiveness of the proposed techniques. The MOMIS Query Manager prototype (see Chapter 2) has been developed to enable users to query an integrated schema, and to provide users a consistent and concise unified answer. The effectiveness of the MOMIS Query Manager prototype has been demonstrated by means of the THALIA testbed for Data Integration Systems described in Chapter 3. The MOMIS Query Manager can deal with all the queries of the benchmark. This is a remarkable result since no mediator system, excluding MOMIS, has provided a complete answer to the benchmark. Moreover, the MOMIS Query Manager has been experimented in the context of the WISDOM, NeP4B, STIL, and CEREALAB research projects.

A new kind of metadata that offers a synthesized view of an attributes values, the relevant values, has been defined (see Chapter 4). A method and a prototype to compute such metadata, based on data mining and clustering techniques, have been developed. The effectiveness of such metadata for creating or refining a search query in a knowledge base is demonstrated by means of experimental results.

The security issues in Data integration/interoperation systems have been investigated (Chapter 5), and an innovative method to preserve data confidentiality and availability when querying integrated data has been proposed.

A security framework for collaborative applications, in which the actions that users can perform are dynamically determined on the basis of their attribute values, has been presented, and the effectiveness of the framework has been demonstrated by an implemented prototype.

6.2 Publications

The research activities described in this thesis have produced the following publications:

- On the issue of Query Processing in Data Integration Systems, one publication on international conference [11].
- On the topic of metadata generation and exploitation for querying Data integration Systems, one article in international journal [22], two papers in international conferences [19, 8], and two publications in national conferences [18, 17].
- On the security issues in Data integration/interoperation systems, three publications in international conferences [45, 43, 44].

Appendix A

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

The following is a BNF description for the ODL$_{I^3}$ description language. The object-oriented language, with an underlying Description Logic, is introduced for information extraction. The ODL$_{I^3}$ language is presented in [16], in the following we included the syntax fragments which differ from the original ODL grammar, referring to this one for the remainder.

⟨interface_dcl⟩ ::= ⟨interface_header⟩
{[⟨ interface_body⟩]**}**;
[**union** ⟨identifier⟩ **{** ⟨interface_body⟩ **};**]

⟨interface_header⟩ ::= **interface** ⟨identifier⟩
[⟨inheritance_spec⟩]
[⟨type_property_list⟩]

⟨inheritance_spec⟩ ::= : ⟨scoped_name⟩
[,⟨inheritance_spec⟩]

Local schema pattern definition: the wrapper must indicate the kind and the name of the source of each pattern.

⟨type_property_list⟩ ::= **(** [⟨source_spec⟩]
[⟨extent_spec⟩]
[⟨key_spec⟩] [⟨f_key_spec⟩] [⟨c_key_spec⟩] **)**
⟨source_spec⟩ ::= **source** ⟨source_type⟩
⟨source_name⟩
⟨source_type⟩ ::= **relational** | **nfrelational**
| **object** | **file**
| **semistructured**
⟨source_name⟩ ::= ⟨identifier⟩
⟨extent_spec⟩ ::= **extent** ⟨extent_list⟩
⟨extent_list⟩ ::= ⟨string⟩ | ⟨string⟩,⟨extent_list⟩
⟨key_spec⟩ ::= **key**[**s**] ⟨key_list⟩
⟨f_key_spec⟩ ::= **foreign_key** (⟨f_key_list⟩)
references ⟨key_list
⟩ [⟨f_key_spec⟩]
⟨c_key_spec⟩ ::= **candidate_key** ⟨identifier⟩
(⟨key_list⟩**)**

Global pattern definition rule, used to map the attributes between the global definition and the corrisponding ones in the local sources.

⟨attr_dcl⟩ ::= **[readonly] attribute**
[⟨domain_type⟩]
⟨attribute_name⟩ **[*]**
[⟨fixed_array_size⟩]
[⟨mapping_rule_dcl⟩]
⟨mapping_rule_dcl⟩ ::= **mapping_rule** ⟨rule_list⟩
⟨rule_list⟩ ::= ⟨rule⟩ | ⟨rule⟩**,**⟨rule_list⟩
⟨rule⟩ ::= ⟨local_attr_name⟩ |
'⟨identifier⟩'
⟨and_expression⟩ |
⟨union_expression⟩
⟨and_expression⟩ ::= **(** ⟨local_attr_name⟩ **and**
⟨and_list⟩ **)**
⟨and_list⟩ ::= ⟨local_attr_name⟩
| ⟨local_attr_name⟩ **and**
⟨and_list⟩
⟨union_expression⟩ ::= **(** ⟨local_attr_name⟩ **union**
⟨union_list⟩ **on** ⟨identifier⟩ **)**
⟨union_list⟩ ::= ⟨local_attr_name⟩
| ⟨local_attr_name⟩ **union**
⟨union_list⟩
⟨local_attr_name⟩ ::= ⟨source_name⟩**.**⟨class_name⟩**.**
⟨attribute_name⟩

...

Terminological relationships used to define the Common Thesaurus.

⟨relationships_list⟩ ::= ⟨relationship_dcl⟩**;** |
⟨relationship_dcl⟩**;**
⟨relationships_list⟩
⟨relationships_dcl⟩ ::= ⟨local_name⟩
⟨relationship_type⟩
⟨local_name⟩
⟨local_name⟩ ::= ⟨source_name⟩**.**
⟨local_class_name⟩
[**.**⟨local_attr_name⟩]
⟨relationship_type⟩ ::= **SYN** | **BT** | **NT** | **RT**

...

OLCD integrity constraint definition: declaration of rule (using *if then* definition) valid for each instance of the data; mapping rule specification (*or* and *union* specification rule).

⟨rule_list⟩ ::= ⟨rule_dcl⟩**;** | ⟨rule_dcl⟩**;** ⟨rule_list⟩
⟨rule_dcl⟩ ::= **rule** ⟨identifier⟩ ⟨rule_spec⟩
⟨rule_spec⟩ ::= ⟨rule_pre⟩ **then** ⟨rule_post⟩ |
{ ⟨case_dcl⟩ **}**
⟨rule_pre⟩ ::= ⟨forall⟩ ⟨identifier⟩ **in** ⟨identifier⟩ **:**
⟨rule_body_list⟩
⟨rule_post⟩ ::= ⟨rule_body_list⟩
⟨case_dcl⟩ ::= **case of** ⟨identifier⟩ **:** ⟨case_list⟩
⟨case_list⟩ ::= ⟨case_spec⟩ | ⟨case_spec⟩ ⟨case_list⟩
⟨case_spec⟩ ::= ⟨identifier⟩ **:** ⟨identifier⟩ **;**
⟨rule_body_list⟩ ::= **(** ⟨rule_body_list⟩ **)** |
⟨rule_body⟩ |
⟨rule_body_list⟩ **and**
⟨rule_body⟩ |
⟨rule_body_list⟩ **and**
(⟨rule_body_list⟩ **)**
⟨rule_body⟩ ::= ⟨dotted_name⟩
⟨rule_const_op⟩
⟨literal_value⟩ |
⟨dotted_name⟩
⟨rule_const_op⟩
⟨rule_cast⟩ ⟨literal_value⟩ |
⟨dotted_name⟩ **in**
⟨dotted_name⟩ |
⟨forall⟩ ⟨identifier⟩ **in**
⟨dotted_name⟩ **:**
⟨rule_body_list⟩ |
exists ⟨identifier⟩ **in**
⟨dotted_name⟩ **:**
⟨rule_body_list⟩
⟨rule_const_op⟩ ::= = | ≥ | ≤ | > | <
⟨rule_cast⟩ ::= **(**⟨simple_type_spec⟩**)**
⟨dotted_name⟩ ::= ⟨identifier⟩ | ⟨identifier⟩**.**
⟨dotted_name⟩
⟨forall⟩ ::= **for all** | **forall**

Appendix B

The OQL$_{I^3}$ query language syntax

The following is a BNF description for the OQL$_{I^3}$ query language. The OQL$_{I^3}$ query language is similar to the SQL query language. In the following we included the OQL$_{I^3}$ syntax accepted by the MOMIS Query Manager.

A generic OQL$_{I^3}$ query is expressed as following:

SELECT	select_list
FROM	from_list
WHERE	where_clause
ORDER BY	order_list

where:

select_list	::=	*
		[DISTINCT]
		GlobalClass-1.Attribute-1, GlobalClass-n.Attribute-q, ...
from_list	::=	GlobalClass [AS GlobalClass-Alias]
		GlobalClass [AS GlobalClass-Alias], from_list
simple_condition	::=	GlobalClass-m.Attribute-p rel-op GlobalClass-n.Attribute-q
		GlobalClass.Attribute rel-op Const
where_clause	::=	simple_condition
		where_clause
		[NOT] where_clause
		[NOT] where_clause
		simple_condition AND where_clause
		simple_condition OR where_clause
order_list	::=	GlobalClass.Attribute DESC
		GlobalClass.Attribute ASC
		GlobalClass.Attribute DESC , order_list
		GlobalClass.Attribute ASC, order_list
rel-op	::=	LIKE
		=
		!=
		$\rangle$
		$\rangle =$
		$\langle$
		$\langle =$
Const	::=	oql-datatype value

Bibliography

[1] S. Abiteboul, R. Agrawal, P. Bernstein, M. Carey, S. Ceri, B. Croft, D. DeWitt, M. Franklin, H. G. Molina, D. Gawlick, J. Gray, L. Haas, A. Halevy, J. Hellerstein, Y. Ioannidis, M. Kersten, M. Pazzani, M. Lesk, D. Maier, J. Naughton, H. Schek, T. Sellis, A. Silberschatz, M. Stonebraker, R. Snodgrass, J. Ullman, G. Weikum, J. Widom, and S. Zdonik. The lowell database research self-assessment. *Commun. ACM*, 48(5):111–118, 2005.

[2] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: from relations to semistructured data and XML*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

[3] M. A. Al-Kahtani and R. Sandhu. A Model for Attribute-Based User-Role Assignment. In *Annual Computer Security Applications Conference (ACSAC)*, pages 353–364. IEEE Computer Society, 2002.

[4] M. A. Al-Kahtani and R. Sandhu. Induced role hierarchies with attribute-based RBAC. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 142–148, 2003.

[5] J. L. Ambite and C. A. Knoblock. Flexible and scalable cost-based query planning in mediators: A transformational approach. *Artificial Intelligence Journal*, 118:1–2, 2000.

[6] C. Batini, M. Lenzerini, and S. B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[7] D. Beneventano and S. Bergamaschi. Semantic Search Engines based on Data Integration Systems. In *Semantic Web: Theory, Tools and Applications (Ed. Jorge Cardoso)*. Idea Group Publishing, 2006.

[8] D. Beneventano, S. Bergamaschi, S. Bruschi, F. Guerra, M. Orsini, and M. Vincini. Instances navigation for querying integrated data from web-sites. In *In Int. Conf. on Web Information Systems and Technologies*, Setubal, Portugal, April 2006.

[9] D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing*, 7(5):42–51, 2003.

[10] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Trans. Knowl. Data Eng.*, 10(4):576–598, 1998.

[11] D. Beneventano, S. Bergamaschi, M. Vincini, M. Orsini, and R. C. N. Mbinkeu. Getting through the THALIA benchmark with MOMIS. In *International Workshop on Database Interoperability (InterDB) co-located with VLDB*, 2007.

[12] D. Beneventano, F. Guerra, M. Orsini, L. Po, A. Sala, M. D. Gioia, M. Comerio, F. de Paoli, A. Maurino, M. Palmonari, C. Gennaro, F. Sebastiani, A. Turati, D. Cerizza, I. Celino, and F. Corcoglioniti. Detailed design for building semantic peer. *Networked Peers for Business, Deliverable D.2.1, Final Version*, apr 2008. http://www.dbgroup.unimo.it/publication/d2_1.pdf.

[13] D. Beneventano and M. Lenzerini. Final release of the system prototype for query management. *Sewasie, Deliverable D.3.5, Final Version*, Apr. 2005. http://www.dbgroup.unimo.it/prototipo/paper/D3.5_final.pdf.

[14] S. Bergamaschi, D. Beneventano, F. Guerra, and M. Vincini. Building a tourism information provider with the MOMIS system. *Information Technology & Tourism*, 7(3-4):221–238, 2005.

[15] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Rec.*, 28(1):54–59, 1999.

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

[17] S. Bergamaschi, F. Guerra, M. Orsini, and C. Sartori. A new type of metadata for querying data integration systems. In M. Ceci, D. Malerba, and L. Tanca, editors, *SEBD*, pages 266–273, 2007.

[18] S. Bergamaschi, F. Guerra, M. Orsini, C. Sartori, and M. Vincini. Relevantnews: a semantic news feed aggregator. In G. Semeraro, E. D. Sciascio, C. Morbidoni, and H. Stoermer, editors, *SWAP*, volume 314 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2007.

[19] S. Bergamaschi, M. Orsini, F. Guerra, and C. Sartori. Relevant values: New metadata to provide insight on attribute values at schema level. In J. Cardoso, J. Cordeiro, and J. Filipe, editors, *ICEIS (1)*, pages 274–279, 2007.

[20] S. Bergamaschi, L. Po, and S. Sorrentino. Automatic annotation in data integration systems. In R. Meersman, Z. Tari, and P. Herrero, editors, *OTM Workshops (1)*, volume 4805 of *Lecture Notes in Computer Science*, pages 27–28. Springer, 2007.

[21] S. Bergamaschi, L. Po, and S. Sorrentino. Automatic annotation for mapping discovery in data integration systems. In S. Gaglio, I. Infantino, and D. Saccà, editors, *SEBD*, pages 334–341, 2008.

[22] S. Bergamaschi, C. Sartori, F. Guerra, and M. Orsini. Extracting relevant attribute values for improved search. *IEEE Internet Computing*, 11(5):26–35, 2007.

[23] P. A. Bernstein and L. M. Haas. A guide to the tools and core technologies for merging information from disparate sources. *Communications of the ACM*, 51(9):72–79, sep 2008.

[24] P. A. Bernstein and S. Melnik. Model management 2.0: manipulating richer mappings. In C. Y. Chan, B. C. Ooi, and A. Zhou, editors, *SIGMOD Conference*, pages 1–12. ACM, 2007.

[25] E. Bertino, S. Castano, E. Ferrari, and M. Mesiti. Protection and administration of XML data sources. *Data and Knowledge Engineering*, 43(3):237–260, 2002.

[26] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 29–37, 2005.

[27] L. E. Bertossi and J. Chomicki. Query answering in inconsistent databases. In J. Chomicki, R. van der Meyden, and G. Saake, editors, *Logics for Emerging Applications of Databases*, pages 43–83. Springer, 2003.

[28] J. Bleiholder, K. Draba, and F. Naumann. Fusem - exploring different semantics of data fusion. In *VLDB*, pages 1350–1353, 2007.

[29] J. Bleiholder and F. Naumann. Declarative data fusion - syntax, semantics, and implementation. In J. Eder, H.-M. Haav, A. Kalja, and J. Penjam, editors, *ADBIS*, volume 3631 of *Lecture Notes in Computer Science*, pages 58–73. Springer, 2005.

[30] J. Bleiholder and F. Naumann. Data fusion. *ACM Computing Surveys*, 2008.

[31] P. A. Bonatti, M. L. Sapino, and V. S. Subrahmanian. Merging heterogeneous security orderings. *Journal of Computer Security*, 5(1):3–29, 1997.

[32] D. Brickley and R. Guha. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema, W3C Working Draft, February 2004.

[33] A. Calì, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Data integration under integrity constraints. *Inf. Syst.*, 29(2):147–163, 2004.

[34] K. S. Candan, S. Jajodia, and V. S. Subrahmanian. Secure mediated databases. In *IEEE International Conference on Data Engineering (ICDE)*, pages 28–37, 1996.

[35] S. Castano, V. D. Antonellis, and S. D. C. di Vimercati. Global viewing of heterogeneous data sources. *IEEE Trans. Knowl. Data Eng.*, 13(2):277–297, 2001.

[36] K. C.-C. Chang and H. Garcia-Molina. Mind your vocabulary: Query mapping across heterogeneous information sources. In Delis et al. [55], pages 335–346.

[37] S. Chaudhuri, R. Ramakrishnan, and G. Weikum. Integrating db and ir technologies: What is the sound of one hand clapping? In *Proc. of the 2nd Conf. on Innovative Data Systems Research, Asilomar, CA, USA*, pages 1–12, 2005.

[38] L. Cirio, I. F. Cruz, and R. Tamassia. A Role and Attribute Based Access Control System Using Semantic Web Technologies. In *International IFIP Workshop on Semantic Web and Web Semantics*, volume 4806 of *Lecture Notes in Computer Science*, pages 1256–1266. Springer, 2007.

[39] Clark & Parsia, LLC. Pellet. http://pellet.owldl.com.

[40] G. Cleuziou, L. Martin, and C. Vrain. PoBOC: An overlapping clustering algorithm, application to rule-based classification and textual data. In *Proceedings of the 16th ECAI conference*, pages 440–444, 2004.

[41] E. F. Codd. Extending the database relational model to capture more meaning. *ACM Trans. Database Syst.*, 4(4):397–434, 1979.

[42] S. Cohen and Y. Sagiv. An incremental algorithm for computing ranked full disjunctions. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 98–107, New York, NY, USA, 2005. ACM.

[43] I. F. Cruz, R. Gjomemo, B. Lin, and M. Orsini. A constraint and attribute based security framework for dynamic role assignment in collaborative environments. In *CollaborateCom 2008 - 4th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2008.

[44] I. F. Cruz, R. Gjomemo, B. Lin, and M. Orsini. A location aware role and attribute based access control system. In *ACM GIS 2008 - 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2008.

[45] I. F. Cruz, R. Gjomemo, and M. Orsini. A secure mediator for integrating multiple level access control policies. In I. Lovrek, R. J. Howlett, and L. C. Jain, editors, *KES (2)*, volume 5178 of *Lecture Notes in Computer Science*, pages 354–362. Springer, 2008.

[46] I. F. Cruz, R. Tamassia, and D. Yao. Privacy-Preserving Schema Matching Using Mutual Information. In *IFIP Conference on Data and Applications Security (DBSec)*, volume 4602 of *Lecture Notes in Computer Science*, pages 93–94. Springer, 2007.

[47] I. F. Cruz and H. Xiao. Using a Layered Approach for Interoperability on the Semantic Web. In *Int. Conf. Web Information Systems Engineering (WISE)*, pages 221–232, 2003.

[48] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for XML documents. *ACM Trans. on Information and System Security*, 5(2):169–202, 2002.

[49] E. Damiani, P. Samarati, S. De Capitani di Vimercati, and S. Paraboschi. Controlling access to XML documents. *IEEE Internet Computing*, 5(6):18–28, 2001.

[50] M. L. Damiani and E. Bertino. Access Control and Privacy in Location-Aware Services for Mobile Organizations. In *International Conference on Mobile Data Management (MDM)*, pages 11–20, 2006.

[51] M. L. Damiani, E. Bertino, B. Catania, and P. Perlasca. GEO-RBAC: A Spatially Aware RBAC. *ACM Transactions on Information and System Security (TISSEC)*, 10(1):2, 2007.

[52] M. R. Darnel. *Theory of Lattice-Ordered Groups*. CRC Press, New York, New York, 10016, 1995.

[53] A. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In Williamson et al. [115], pages 271–280.

[54] S. Dawson, S. Qian, and P. Samarati. Providing security and interoperation of heterogeneous systems. *Distributed and Parallel Databases*, 8(1):119–145, 2000.

[55] A. Delis, C. Faloutsos, and S. Ghandeharizadeh, editors. *SIGMOD 1999, Proceedings ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadelphia, Pennsylvania, USA*. ACM Press, 1999.

[56] J. V. den Bercken, B. Blohsfeld, J.-P. Dittrich, J. Krämer, T. Schäfer, M. Schneider, and B. Seeger. Xxl - a library approach to supporting efficient implementations of advanced database queries. In *VLDB*, pages 39–48, 2001.

[57] F. Du, S. Amer-Yahia, and J. Freire. Shrex: Managing xml documents in relational databases. In M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, editors, *VLDB*, pages 1297–1300. Morgan Kaufmann, 2004.

[58] B. S. Everitt. *Cluster Analysis.* Edward Arnold and Halsted Press, 1993.

[59] C. Farkas, A. Jain, D. Wijesekera, A. Singhal, and B. Thuraisingham. Semantic-aware data protection in web services. In *IEEE Workshop on Web Service Security*, 2006.

[60] T. W. Finin, A. Joshi, L. Kagal, J. Niu, R. S. Sandhu, W. H. Winsborough, and B. M. Thuraisingham. ROWLBAC: Representing Role Based Access Control in OWL. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 73–82, 2008.

[61] C. A. Galindo-Legaria. Outerjoins as disjunctions. In R. T. Snodgrass and M. Winslett, editors, *SIGMOD Conference*, pages 348–358. ACM Press, 1994.

[62] M. R. Genesereth, A. M. Keller, and O. M. Duschka. Infomaster: An information integration system. In J. Peckham, editor, *SIGMOD Conference*, pages 539–542. ACM Press, 1997.

[63] G. D. Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tackling inconsistencies in data integration through source preferences. In F. Naumann and M. Scannapieco, editors, *IQIS*, pages 27–34. ACM, 2004.

[64] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In P. M. G. Apers, P. Atzeni, S. Ceri, S. Paraboschi, K. Ramamohanarao, and R. T. Snodgrass, editors, *VLDB*, pages 491–500. Morgan Kaufmann, 2001.

[65] G. Greco, S. Greco, and E. Zumpano. A logical framework for querying and repairing inconsistent databases. *IEEE Trans. Knowl. Data Eng.*, 15(6):1389–1408, 2003.

[66] A. Gulli. The anatomy of a news search engine. In A. Ellis and T. Hagino, editors, *WWW (Special interest tracks and posters)*, pages 880–881. ACM, 2005.

[67] L. M. Haas. Beauty and the beast: The theory and practice of information integration. In T. Schwentick and D. Suciu, editors, *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 28–43. Springer, 2007.

[68] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[69] A. Y. Halevy, N. Ashish, D. Bitton, M. Carey, D. Draper, J. Pollock, A. Rosenthal, and V. Sikka. Enterprise information integration: successes, challenges and controversies. In *SIGMOD '05: Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 778–787, New York, NY, USA, 2005. ACM.

[70] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001.

[71] J. Hammer, M. Stonebraker, and O. Topsakal. Thalia: Test harness for the assessment of legacy information integration approaches. In *Proceedings of the International Conference on Data Engineering (ICDE*, pages 485–486, 2005.

[72] R. Hauch, A. Miller, and R. Cardwell. Information intelligence: metadata for information discovery, access, and integration. In F. Özcan, editor, *SIGMOD Conference*, pages 793–798. ACM, 2005.

[73] I. Horrocks, P. F. Patel-Schneider, and B. Motik. OWL 1.1 Web Ontology Language Structural Specification and Functional-Style Syntax, 2007.

[74] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *PODS*, pages 51–61. ACM Press, 1997.

[75] IDC. Worldwide data integration and access software 20082012 forecast. *Worldwide Software 2008-2012 Forecast Summary*, Apr. 2008.

[76] Z. G. Ives, D. Florescu, M. Friedman, A. Y. Levy, and D. S. Weld. An adaptive query execution system for data integration. In Delis et al. [55], pages 299–310.

[77] A. Jain and C. Farkas. Secure resource description framework: an access control model. In *ACM Symp. on Access Control Models and Technologies (SACMAT)*, pages 121–129, 2006.

[78] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[79] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A Generalized Temporal Role-Based Access Control Model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005.

[80] S. Kaushik, D. Wijesekera, and P. Ammann. Policy-based dissemination of partial web-ontologies. In *Workshop on Secure Web Services (SWS)*, pages 43–52, 2005.

[81] A. Kiryakov, B. Popov, I. Terziev, D. Manov, and D. Ognyanoff. Semantic annotation, indexing, and retrieval. *J. Web Sem.*, 2(1):49–79, 2004.

[82] C. A. Knoblock, S. Minton, J. L. Ambite, N. Ashish, P. J. Modi, I. Muslea, A. Philpot, and S. Tejada. Modeling web sources for information integration. In *AAAI/IAAI*, pages 211–218, 1998.

[83] D. Kulkarni and A. Tripathi. Context-aware Role-based Access Control in Pervasive Computing Systems. In *ACM Symposium on Access Control Models and Technologies (SACMAT)*, pages 113–122, 2008.

[84] E. Lambrecht, S. Kambhampati, and S. Gnanaprakasam. Optimizing recursive information-gathering plans. In T. Dean, editor, *IJCAI*, pages 1204–1211. Morgan Kaufmann, 1999.

[85] Y. Lei, M. Sabou, V. Lopez, J. Zhu, V. S. Uren, and E. Motta. An infrastructure for acquiring high quality semantic metadata. In Y. Sure and J. Domingue, editors, *ESWC*, volume 4011 of *Lecture Notes in Computer Science*, pages 230–244. Springer, 2006.

[86] M. Lenzerini. Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM.

[87] A. Y. Levy. The information manifold approach to data integration. *IEEE Intelligent Systems*, 13:12–16, 1998.

[88] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *VLDB*, pages 251–262. Morgan Kaufmann, 1996.

[89] C. Li, R. Yerneni, V. Vassalos, H. Garcia-Molina, Y. Papakonstantinou, J. D. Ullman, and M. Valiveti. Capability based mediation in tsimmis. In L. M. Haas and A. Tiwary, editors, *SIGMOD Conference*, pages 564–566. ACM Press, 1998.

[90] X. Li, J. Yan, Z. Deng, L. Ji, W. Fan, B. Zhang, and Z. Chen. A novel clustering-based rss aggregator. In Williamson et al. [115], pages 1309–1310.

[91] J. Lin and A. O. Mendelzon. Merging databases under constraints. *Int. J. Cooperative Inf. Syst.*, 7(1):55–76, 1998.

[92] R. C. N. Mbinkeu. Full outer join optimization techniques in integration information systems. In *Sixime Manifestation des Jeunes Chercheurs en Sciences et Technologies de lInformation et de la Communication (MAJESTICS)*, 2008.

[93] R. J. Miller, M. A. Hernández, L. M. Haas, L.-L. Yan, C. T. H. Ho, R. Fagin, and L. Popa. The clio project: Managing heterogeneity. *SIGMOD Record*, 30(1):78–83, 2001.

[94] F. Naumann, A. Bilke, J. Bleiholder, and M. Weis. Data fusion in three steps: Resolving schema, tuple, and value inconsistencies. *IEEE Data Eng. Bull.*, 29(2):21–31, 2006.

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

[96] F. Naumann and M. Häussler. Declarative data merging with conflict resolution. In C. Fisher and B. N. Davidson, editors, *IQ*, pages 212–224. MIT, 2002.

[97] M. Oliva and F. Saltor. Integrating security policies in federated database systems. In *Annual Working Conf. on Database Security (DBSec)*, pages 135–148, 2000.

[98] S. L. Osborn, R. S. Sandhu, and Q. Munawer. Configuring Role-based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security (TISSEC)*, 3(2):85–106, 2000.

[99] C.-C. Pan, P. Mitra, and P. Liu. Semantic access control for information interoperation. In *ACM Symp. on Access Control Models and Technologies (SACMAT)*, pages 237–246, 2006.

[100] R. Pottinger and P. A. Bernstein. Merging models based on given correspondences. In *VLDB*, pages 826–873, 2003.

[101] D. R. Radev, J. Otterbacher, A. Winkel, and S. Blair-Goldensohn. Newsinessence: summarizing online news topics. *Commun. ACM*, 48(10):95–98, 2005.

[102] A. Rajaraman and J. D. Ullman. Integrating information by outerjoins and full disjunctions. In *PODS*, pages 238–248. ACM Press, 1996.

[103] P. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65, 1987.

[104] G. M. Sacco. Dynamic taxonomies and guided searches. *JASIST*, 57(6):792–796, 2006.

[105] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-Based Access Control Models. *Computer*, 29(2):38–47, 1996.

[106] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid. Privacy Preserving Schema and Data Matching. In *ACM SIGMOD International Conference on Management of Data*, pages 653–664, 2007.

[107] M. K. Smith, C. Welty, and D. L. McGuinness. OWL web ontology language guide. http://www.w3.org/TR/owl-guide/, February 2004.

[108] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Proc. of KDD Workshop on Text Mining*, 2000.

[109] A. Toninelli, R. Montanari, L. Kagal, and O. Lassila. Proteus: A Semantic Context-Aware Adaptive Policy Model. In *IEEE International Workshop on Policies for Distributed Systems and Networks*, pages 129–140, 2007.

[110] J. D. Ullman. Information integration using logical views. In *ICDT '97: Proceedings of the 6th International Conference on Database Theory*, pages 19–40, London, UK, 1997. Springer-Verlag.

[111] J. D. Ullman. Information integration using logical views. *Theor. Comput. Sci.*, 239(2):189–210, 2000.

[112] V. S. Uren, P. Cimiano, J. Iria, S. Handschuh, M. Vargas-Vera, E. Motta, and F. Ciravegna. Semantic annotation for knowledge management: Requirements and a survey of the state of the art. *J. Web Sem.*, 4(1):14–28, 2006.

[113] C. J. van Rijsbergen. *Information Retrieval.* Butterworth, 1979.

[114] G. Wiederhold. Intelligent integration of information. In P. Buneman and S. Jajodia, editors, *SIGMOD Conference*, pages 434–437. ACM Press, 1993.

[115] C. L. Williamson, M. E. Zurko, P. F. Patel-Schneider, and P. J. Shenoy, editors. *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007.* ACM, 2007.

[116] H. Xiao and I. F. Cruz. Integrating and exchanging XML data using ontologies. In *Journal on Data Semantics VI*, volume 4090 of *Lecture Notes in Computer Science*, pages 67–89. Springer, 2006.

[117] N. A. Yahaya and R. Buang. Automated metadata extraction from web sources. *wi-iatw*, 0:176–179, 2006.

[118] B. Yu, L. Liu, B. C. Ooi, and K. L. Tan. Keyword join: Realizing keyword search for information integration. 2006.