DEGREE OF DOCTOR OF PHILOSOPHY IN
COMPUTER ENGINEERING AND SCIENCE

DOCTORATE SCHOOL IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXII Cycle

UNIVERSITY OF MODENA AND REGGIO EMILIA

INFORMATION ENGINEERING DEPARTMENT

Ph.D. DISSERTATION

Data and Service Integration: Architectures and Applications to Real Domains

Candidate:
Antonio Sala
Advisor:
Prof.Sonia Bergamaschi
Co-Advisor:
Dr.Howard Ho
The Director of the School:
Prof.Sonia Bergamaschi

DOTTORATO DI RICERCA IN
COMPUTER ENGINEERING AND SCIENCE

SCUOLA DI DOTTORATO IN
INFORMATION AND COMMUNICATION TECHNOLOGIES

XXII Ciclo

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

TESI PER IL CONSEGUIMENTO DEL TITOLO DI DOTTORE DI RICERCA

Data and Services Integration: Architectures and Applications to Real Domains

Tesi di:
Antonio Sala
Relatore:
Prof.Sonia Bergamaschi
Co-Relatore:
Dr.Howard Ho
Il Direttore:
Prof.Sonia Bergamaschi

Keywords:

Semantic Data Integration
Data and Service Integration
Service as Data
Multimedia Data Integration
Distributed Architectures for Data Integration

Abstract

This thesis focuses on Semantic Data Integration Systems, with particular attention to mediator system approaches, to perform data and service integration. My research activity is based on the MOMIS (Mediator Environment for Multiple Information Sources) system, developed by the DBGroup of the University of Modena and Reggio Emilia.

One of the topics of this thesis is the application of MOMIS to the bioinformatics domain to integrate different public databases to create an ontology of molecular and phenotypic cereals data.

However, the main contribution of this thesis is a semantic approach to perform aggregated search of data and services. In particular, I describe a technique that, on the basis of an ontological representation of data and services related to a domain, supports the translation of a data query into a service discovery process, that has also been implemented as a MOMIS extension. This approach can be described as a *Service as Data* approach, as opposed to *Data as a Service* approaches. In the Service as Data approach, informative services are considered as a kind of source to be integrated with other data sources, to enhance the domain knowledge provided by a Global Schema of data.

Finally, new technologies and approaches for data integration have been investigated, in particular distributed architecture, with the objective to provide a scalable architecture for data integration. An integration framework in a distributed environment is presented that allows realizing a data integration process on the cloud.

Sommario

L'argomento principale di questa tesi sono i Sistemi di Integrazione Dati Semantici, con particolare attenzione agli approcci basati su sistemi a mediatore, per l'integrazione di dati e servizi. La mia attivitá di ricerca si basa sul sistema MOMIS (Mediator Environment for Multiple Information Sources), sviluppato dal DBGroup dell'Universitá di Modena e Reggio Emilia.

Il primo argomento di questa tesi è appunto l'applicazione di MOMIS nel dominio bioinformatico con l'obiettivo di integrare diversi database pubblici per creare un'ontologia di dati molecolari e fenotipici di piante di cereali.

Il contributo principale di questa tesi è peró un approccio semantico per eseguire ricerche aggregate di dati e servizi. In particulare, viene descritta una tecnica che, sulla base di una rappresentazione ontologica di dati e servizi relativi a un particolare dominio, supporta la traduzione di una interrogazione formulata sui dati in un processo di scoperta di servizi. Questa tecnica è stata anche sviluppata come estensione di MOMIS. Questo approccio puó essere definito come un approccio *Service as Data*, in opposizione agli approcci *Data as a Service*. Nell'approccio Service as Data, i servizi informativi vengono considerati come un particolare tipo di sorgente da integrare con altre sorgenti dati per aumentare la conoscenza di dominio fornita da uno schema globale di dati.

Infine, sono stati studiati nuovi approcci e nuove tecnologie, in particolare le architetture distribuite, con l'obiettivo di realizzare un'architettura scalabile per l'integrazione dati. Viene presentato un framework per l'integrazione in un ambiente distribuito che consente di sviluppare un processo di integrazione dati "on the cloud".

Contents

List of Figures

Chapter 1

Data and Service Integration Systems

1.1 Introduction and Motivation

Data integration is the problem of combining data residing at different sources, and providing the user with a unified view of these data [Lenzerini, 2002]. Data integration is crucial in numerous real world applications in different domains: in large enterprises that own a multitude of data sources, or when two similar companies need to merge their databases; in scientific projects, where data sets are being produced independently by multiple researchers and it is needed to combine research results from different repositories; for better cooperation among government agencies, each with their own data sources [Halevy et al., 2006]. In addition, information integration has also played an important role in internet search. For example, specialized search engines that integrate data from multiple deep web sources in specific domains (e.g., travel, jobs) can make use of information integration techniques. Finally, information integration has also raised importance in the life sciences, where diverse data is being produced at increasing rates, and progress depends on researchers' ability to synthesize data from multiple sources [Halevy et al., 2006].

The problem of designing and developing effective information integration techniques became even more important with the advent of the Web that eased producing and sharing data in collaborative environments. One of the main motivations for data integration is that data often reside in different, heterogeneous and distributed data sources which are typically processed by legacy systems.

Data integration is a research field that has been deeply investigated in

the last twenty years, especially by the database research community, for its important impact and possible applications in different fields, and the research community has been developing numerous and different techniques for integrating heterogeneous data sources. Data Integration Systems supply a transparent access to a collection of data stored in multiple, autonomous, and heterogeneous data sources. Sources can range from database systems and legacy systems to forms on the Web, web services and flat files.

Different approaches have been proposed and developed during the last three decades:

- *A federated database system (FDBS)* is a collection of cooperating database systems that are autonomous and possibly heterogeneous, integrated to various degree [Heimbigner and McLeod, 1985, Sheth and Larson, 1990]. A global schemas has to be defined to integrate data originated in the participating databases. The design of the integrated schema needs to deal with data integration issues on how to combine the same or similar data represented differently in different participating databases. The schemas of the participating databases refers to legacy systems, i.e. systems existing before the integrated database schema is designed. Thus, the design process becomes bottom-up, whereas homogeneous databases are usually designed top-down [DBL, 2009].

- In the *data warehouse* approach [Inmon, 2002], copies of data from several databases and data sources are stored in a single database, called a (data) warehouse. The data stored at the warehouse are pre-processed in a complex way before storage; it is usually referred to as extract, transform and load (ETL) activities on the data to create a data warehouse. The warehouse has to be updated periodically, but since data must be extracted, transformed and loaded into the warehouse, there is an element of latency in the warehouse data. Data warehouses are mainly designed to facilitate reporting and analysis.

- *Read-only data integration systems* are data integration systems based on mediator systems. A mediator [Wiederhold, 1992] is a software component that supports a virtual database, which the user may query as if it were materialized. The mediated schema is not meant to store any data, but is purely a logical schema that provides a middle layer making the applications independent of the data resources. It is based on building a unified *global schema* as a synthesis of the source schemas to be integrated. 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 rewriting operations taking into account the mediated and the source schemas. Results from sub-queries are finally unified by data reconciliation techniques. One of the most important aspects in the design of a mediator based system is the specification of the correspondence between the data at the sources and those in the mediated schema. The global schema is built for the information integration application and contains only the aspects of the domain that are relevant to the application. Therefore, it does not necessarily contain all the attributes present in the sources, but only a subset of them. The global schema represents a conceptualization of the sources involved, i.e. a domain ontology. Section 1.3.1 describes some mediator based data integration systems.

- *Data exchange* [Fagin et al., 2005, Kolaitis, 2005]: refers to an architecture that includes source and target databases, and the semantic mappings that specify how to populate the target from data in the source. It is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible. Data exchange is used in many tasks that require data to be transferred between existing, independently created applications. In data exchange scenarios, the target schema is often independently created and comes with its own constraints.

- In *Peer-to-peer (P2P)* data integration systems [Calvanese et al., 2004, Halevy et al., 2003] the integration is done among several peers. Peer-to-peer systems enable highly distributed file access where users search for data stored in peers. In a peer-to-peer database queries are propagated between the participating peer nodes. The advantage of peer-data management systems is that it is no longer necessary to create a single global schema in cases where such a schema is hard to build or agree upon.

Another goal of data integration is to enable rapid development of new applications requiring data from multiple sources. This simple goal hides many challenges, from identifying the best data sources to use, to creating an appropriate schema for the integrated data [Haas, 2007].

The Semantic Web vision of the World Wide Web Consortium[1] is to extend the current Web, so that "information is given well-defined

[1] `http://www.w3.org/`

meaning, better enabling computers and people to work in cooperation." [Berners-Lee et al., 2001]. The data representations defined by Semantic Web initiatives can be seen as the next step in the evolution of data management. The central idea of the Semantic Web initiative is to enrich Web content by machine-processable semantics and is based on the use of semantic annotations to describe the meaning of information. Ontologies [Guarino, 1995] describe the knowledge needed to understand collections of Web information, and the semantic annotations are linked to such ontologies. Logic-based techniques can the be exploited to process and query collections of meta-data and ontologies.

The main limitations of traditional data integration systems have been represented by the fact that the definition of the global schema (or the domain ontology) and of the mappings between the data schemas of the different sources is usually a manual or semi-automatic process.

The semantic integration research area is a joint effort between the data integration research community, and the knowledge management and ontology research. Ontologies can be the solution to some of the challenges of data integration but also the source of new problems, like ontology alignment. Ontologies, instead of schemas, are a richer method to represent information domains and can enable collaboration within and across information domains. [Noy, 2004] enumerates the three main dimensions of semantic-integration research:

- Mapping discovery: how to find similarities between two ontologies.
- Declarative formal representations of mappings: how to represent the mappings between two ontologies to enable reasoning with mappings.
- Reasoning with mappings: what to do with the mappings, what type of reasoning they can be used for.

The advent of Service Oriented Architecture (SOA) is also playing an important role in Information Technology. The key benefits of SOA includes a new and richer alignment of business and I/T processes from modeling and service identification, significant agility by exploiting loose coupling of functionality and services, and the opportunity to achieve significant reuse. Although application processes have been the focus of SOA, the benefits of SOA are not limited to business logic and application integration. Equally applicable and important benefits from SOA to the enterprise can be obtained by extending SOA approaches to include data and information sources [Dan et al., 2007]. Inspired by the advantage of Web services technologies proposing the software as a service (SaaS) model

[Bennett et al., 2000, Buyya, 2009], recently, various research effort have concentrated on the development of the concept of data as a service (DaaS) [Dan et al., 2007].

Semantics play an important role in the complete lifecycle of Web services. During development, the service provider can explicate the intended semantics by annotating the appropriate parts of the Web service with concepts from a richer semantic model. Since semantic models (i.e., ontologies) provide agreement on the meaning and intended use of terms, and may provide formal and informal definitions of the entities, there will be less ambiguity in the intended semantics of the provider. During discovery, the service requester can describe the service requirements using terms from the semantic model. Reasoning techniques can be used to find the semantic similarity between the service description and the request. During composition, the functional aspect of the annotations can be used to aggregate the functionality of multiple services to create useful service compositions. More importantly, semantics can make it possible to specify mappings between exchanged data, which would be difficult to do with syntactic representation offered by the current standards. During invocation, mappings can be used for data transformations. Therefore, once represented, semantics can be leveraged by tools to automate service discovery, mediation, composition and monitoring. In the recent years, several research efforts focused on this topic, as for example WSMO [Roman et al., 2004], OWL-S [Akkiraju et al., 2005], WSDL-S [Akkiraju et al., 2005]. As the set of available Web services expands, it becomes increasingly important to have automated tools to help identify services that match the requirements of a service requester. Finding suitable Web services depends on the facilities available for service providers to describe the capabilities of their services and for service requesters to describe their requirements in an unambiguous and ideally, machine-interpretable form. Adding semantics to represent the requirements and capabilities of Web services is essential for achieving this unambiguity and machine-interpretability.

From a user perspective, data and services provide a complementary vision of an information source: data provide detailed information about specific needs, while services execute processes involving data and returning an informative result too. For this reason, users need to perform aggregated searches able to identify not only relevant data, but also services able to operate on them. At the current state of the art such aggregated search can be performed manually only by expert users, that first identify relevant data, and then identify existing relevant services. Data search or service invocation can be individually realized, and both these operations provide value to users in the context of complex interactions.

The research on data integration and service discovering has involved from the beginning different (not always overlapping) communities. As a consequence, data and services are described with different models, and different techniques to retrieve data and services have been developed. Nevertheless, from a user perspective, the border between data and services is often not so definite, since data and services provide a complementary view of the available resources: data provide detailed information about specific needs, while services execute processes involving data and returning an informative result.

Users need new techniques to manage data and services in a unified manner: both the richness of the information available on the Web and the difficulties the user faces in gathering such information (as a service result, as a query on a form, as a query on a data source containing information extracted from web sites) make a tool for searching at the same time data and related services, with the same language, really necessary. The need to perform aggregated search is a young research field and very few approaches have been proposed. Search computing [Ceri, 2009] is a novel discipline whose goal is to answer to complex, multi-domain queries. The goal of such research activity is supporting a new generation of pure data oriented user queries such as "where can I attend a database scientific conference in a city within a hot region served by luxury hotels and reachable with cheap flights?", while a more ambitious goal is to enrich the above mentioned query by offering not only the list of scientific database conferences with locations and the name of the hotel and flights, but also the list of services able to book hotels and flights. In [Murdock and Lalmas, 2008], the authors consider as aggregated search the task of searching and assembling information from a variety of sources, placing it in a single interface.

However, the increasing amount of digital information also requires to think to new architectures able to ideally process peta-bytes of data. New distributed architectures are now being defined, to guarantee incremental scalability. Cloud Architectures [Weiss, 2007, Buyya, 2009] address key difficulties surrounding large-scale data processing. In traditional data processing it is difficult to get as many machines as an application needs. It is also difficult to get the machines when one needs them. Cloud architectures allows distributing and coordinating a large-scale job on different machines, running processes on them, and provisioning another machine to recover if one machine fails. Furthermore, "Clouds" can easily scale up and down based on dynamic workloads, guaranteeing handling large processes in a cost effective way. Cloud Architectures take advantage of simple Application Programming Interfaces (API) of Internet-accessible services that scale on-demand, that are industrial-strength, where the complex reliability

and scalability logic of the underlying services remains implemented and hidden inside the cloud. With the increasing amount of data available in different domains, sometimes up to, and even not limited to, peta-bytes of data, it seems obvious that data integration can take great advantage of cloud architectures to realize distributed implementation of data integration processes. These advantages can be represented either by providing data integration applications that run on the cloud, to improve performances and guarantee scalability and reliability, or to distribute the results of data integration applications from the cloud studying advanced APIs for Data as a Service approaches .

1.2 Contributions and Structure of the Thesis

In this section the main contributions of this thesis are presented together with references to the publications resulted from the research activity. This thesis focuses on Semantic Data Integration Systems, with particular attention to Mediator system approaches, to perform data and service integration. The work is based on the MOMIS[2] (Mediator Environment for Multiple Information Sources) system, developed by the DBGroup of the University of Modena and Reggio Emilia, which performs information extraction and integration from both structured and semi-structured data sources in a semi-automatic way, exploiting the knowledge in a Common Thesaurus (defined by the framework) and the descriptions of source schemas with a combination of clustering and Description Logics techniques. The result of the integration process is a Global virtual Schema (GS) of the underlying data 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 distributed and heterogeneous information systems. The MOMIS system is described in details in Chapter 2.

One of the topics of my research activity was the analysis of the possibility to exploit the MOMIS system in the bioinformatics domain, where large amount of data is available and data integration is a crucial task. In particular, MOMIS was extended to be applied in this domain for the development of an ontology of molecular and phenotypic cereals data, obtained integrating existing public web databases. The GS obtained is the first italian knowledge base that combines molecular and

[2]See `http://www.dbgroup.unimore.it` for references about the MOMIS project.

phenotypic data about cereals. The GS can be queried transparently with regards to integrated data sources using an easy to use graphical interface regardless of the specific languages of the source databases. This work, described in Chapter 3, was conducted in the context of the Cerealab[3] Project in collaboration with the "Department of Agrarian and Food Sciences" of the University of Modena and Reggio Emilia and has been published in [Bergamaschi and Sala, 2006, Bergamaschi and Sala, 2007] and [Sala and Bergamaschi, 2009].

Another topic of my research activity was the study and the development of extensions for the MOMIS system in the context of the NeP4B (Networked Peers for Business)[4] project. The first extension was proposed to enrich its action sphere for representing and querying multimedia data sources other than "traditional" (e.g. relational) data sources. This work, described in Chapter 4, was conducted in collaboration with the "Institute of Information Science and Technologies" (ISTI) of the Italian National Research Council (CNR) of Pisa, and has been published in [Beneventano et al., 2008a].

Other activity was devoted to design and develop a semantic approach to perform aggregated search of data and services. This approach extends MOMIS to include the knowledge offered by the service available in a domain when building a GS. The approach presented in this thesis can be described as a *Service as Data* approach, as opposed to *Data as a Service* approaches. In this approach, informative services are considered as a kind of data source that can enhance the domain knowledge provided by a Global Schema. In particular, I participated in the development of a technique that, on the basis of an ontological representation of data and services related to a domain, supports the translation of a data query into a keyword-based service discovery process. I implemented this approach as a MOMIS extension, as described in Chapter 5. This work was conducted in collaboration with the "Department of Informatics, Systems and Communication" (DISCO) of the University of Milano Bicocca and the Cefriel Research Center of Milan, and has been published in [Palmonari et al., 2007, Guerra et al., 2009, Beneventano et al., 2009].

Finally, new possible technologies and approaches for data integration have been investigated, in particular distributed architectures. The objective was to provide a scalable architecture to collect and integrate publicly available datasets. An integration framework has been developed in a distributed environment to realize a data integration process on the cloud. The work has been conducted at the IBM Almaden Research Center of San Jose,

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

[4] `http://www.dbgroup.unimo.it/nep4b`

California, and was tested on U.S government spending data. The government domain represents an interesting case study, since an increasing amount of government data, especially about spendings, is being disclosed lately by different governments. This work is described in Chapter 6 and has been published in [Sala et al., 2010].

1.3 Context and Related Work

In this section the state of the art in the field of Semantic Data Integration is presented, together with a brief description of the main works and concepts of the other fields related to the work of this thesis, namely Web Service Retrieval and Cloud Architectures.

1.3.1 Data Integration Systems

The research community has been investigating data integration for nearly thirty years: different research communities (database, artificial intelligence, semantic web) have been developing and addressing issues related to data integration in different perspectives. Many different approaches have been proposed and a large number of projects have been produced, making it difficult to provide a classification of the previous work based on comprehensive and shared criteria.

Mediator [Wiederhold, 1992] represents one of the most studied architecture for data integration, and is based on building a mediated schema (the Global Schema) as a synthesis of the source schemas to be integrated. 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 subqueries for the involved sources by means of automatic unfolding-rewriting operations taking into account the mediated and the source schemas. Results from sub-queries are finally unified by data reconciliation techniques. One of the most important aspects in the design of a mediator based system is the specification of the correspondences between the data at the sources and those in the mediated schema [Lenzerini, 2002]. Two basic approaches for specifying mappings in a data integration system have been proposed in the literature, called global-as-view (GAV), where the contents of the mediated schema is expressed in terms of queries over the sources, and local-as-view (LAV), based on the idea that the data sources can be described as views over the mediated schema. The two approaches have been fused in a unique model, called global-local-as-view (GLAV) which synthesizes the character-

istics of the two approaches [Friedman et al., 1999]. Roughly speaking, the main issues of the GAV approach are related to the update of the mediated schema. If sources change, the mediated schema could change with side effects on the applications which refer to it. On the other hands, several issues related to the query rewriting have to be faced for building a query manager for LAV systems.

Several systems have been developed following these approaches (see [Halevy et al., 2006] for a survey). I now briefly describe the most famous forerunner projects that traced the main trends in data integration systems. Among them, the TSIMMIS [Li et al., 1998] system is one of the first data integration system which follows a "structural" approach for building a GAV virtual view of structured and semi-structured data sources. The approach is based on the development of wrappers, that translate the data model of the sources into a self-describing model, OEM (Object Exchange Model), and on the MSL (Mediator Specification Language) integration rules, that are manually specified by the integration designer to enforce source integration. In TSIMMIS, by means of MSL, arbitrary views (in particular, recursive views) can be defined at the mediator layer.

The Information Manifold system [Levy, 1998] 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.

Garlic [Carey et al., 1995] builds a wrapper-based architecture to describe the local source data using an object oriented language. The authors propose an approach where data and schema transformations are handled in a uniform way. The main component of Garlic is a query processor which optimizes and executes queries over diverse data sources posed in an object-extended SQL. The system interacts with wrappers which take care of the data transformations from source data to the middleware model either directly or by constructing views over the middleware schema. Garlic wrappers use an abstraction, the so-called Garlic objects, and the Garlic Definition Language to describe data from diverse resources into a uniform format, the wrappers format. This is then used to produce the global schema. These objects are also used to build views when multiple data sources are encapsulated. Although the authors acknowledge the problem of schematic heterogeneity (data under one schema are represented as metadata in another), they dont tackle it directly. Instead, they have built a semi-automatic tool, Clio, which helps to integrate data and schema transformation processes.

The Clio project [Fagin et al., 2009] pioneered the use of schema mappings, developing a tool for semi-automatically creating mappings between two data representations. In the Clio framework a source schema is mapped

into a different, but fixed, "target" schema. 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.

Other architectures have been developed for managing in a unified manner several data sources. Among them, *matching* is an operation which takes two schemas as input and produces a mapping between elements of the two schemas that correspond semantically to each other [Rahm and Bernstein, 2001]. Thus, matching differs from mediation since it does not build a global schema. On the other hand, mediator systems exploit matching techniques for executing their tasks. Matching techniques exploit different strategies to obtain the correspondances [Rahm and Bernstein, 2001].

Several issues are still open in data integration systems: most of them are related to the conflicts generated when input models are differently represented. In [Pottinger and Bernstein, 2003] conflicts are categorized in three level (representation, meta-model and fundamental conflicts) and a technique for solving them is presented. Another important issue, called entity resolution, is related to identifying the same object in different databases [Bleiholder and Naumann, 2008, Menestrina et al., 2006].

Another problem relevant to data integration, and in particular when large ontologies need to be integrated, is related to the size of the ontology that can be handled. Several approaches have been proposed to extract modules from a given ontology, that can be mainly classified in two different classes of approaches [Palmisano et al., 2009]: approaches that exploit traversal-based extraction techniques considering graph-based representations of ontologies [Noy and Musen, 2004, Doran et al., 2007, d'Aquin et al., 2007, Seidenberg and Rector, 2006], and logic-based approaches grounded on Description Logics semantics [Grau et al., 2007, Jiménez-Ruiz et al., 2008, Wang et al., 2008].

Out in the marketplace, tools for integrating information have proliferated lately. All the biggest player in the IT market offer tools for information integration. As an example, IBM offers the Infosphere[5] suite for Information Integration, Data Warehouse and Master Data Management. It is a complete suite with a large set of functionalities spread in numerous different applications. The complexty of solutions like Infosphere demonstrates how difficult is the data integration problem in the enterprise market, making it impossible to be tackled with a single "out of the box" application. IBM

[5] `http://www-01.ibm.com/software/data/integration/`

Content Integrator is a tool able to integrate enterprise applications with various types of content such as documents, images, audio, video, structured and semi-structured data stored in many heterogeneous environments. It works with any IBM or non-IBM data source, business intelligence tool or operating system. The integration process produces a virtual database that communicates with the data sources through forms, one for each type of data sources. It does not provide any specific mechanisms for resolving data conflicts though.

Other examples of softwares offered by the big IT companies for data integration are Oracle Data Integrator[6] and Microsoft SQL Server Integration Serices[7], but these tools only provide ETL capabilities for data warehousing.

Different open source projects also released tools for data integration. Talend[8] was one of the first ones. Developed by a French start-up company with the same name, it is an Eclipse[9] based environment mainly for ETL purposes. It generates code (Java or Perl plus SQL) implementing the ETL steps defined with a graphical user interface. It provides data cleansing and data profiling capabilities, and is characterised by a great number of connectors (more than 400) including application-specific connectors.

MOMIS will also be realeased as an open-source tool for data integration. The development of this version is carried out by a Spin-Off of the University of Modena and Reggio Emilia called DataRiver[10]. A comparison of the MOMIS capabilities with some commercial data integration applications can be found in [Bergamaschi and Maurino, 2009, Romano, 2007].

1.3.2 Aggregated search

The focus of this thesis is not limited to data integration, but a new approach to data and service integration and retrieval is proposed. Data and service aggregated search aims at identifying not only relevant data, but also services able to operate on them. At the current state of the art such aggregated search can be performed manually only by expert users, that first identify relevant data, and then identify existing relevant services. Aggregated search is a very young research field and very few approaches have been proposed, but all of them are mainly focused on data aggregation according to different criteria. Search computing [Ceri, 2009] is a novel discipline whose goal is to answer to complex, multi domain queries. In this research activity the

[6] http://www.oracle.com/technology/products/oracle-data-integrator/
[7] http://www.microsoft.com/sqlserver/2008/en/us/integration.aspx
[8] http://www.talend.com
[9] http://www.eclipse.org
[10] http://www.datariver.it

need is to aggregate multi-domain query provided by domain specific search engine. In [Brambilla and Ceri, 2009] authors propose an architecture for search computing and they introduce a simple classification of services. Exact services have a "relational" behavior and return a set of unranked answers; search services return a list of answer ranked according to some measure of relevance.

In [Murdock and Lalmas, 2008], authors consider as aggregated search the task of searching and assembling information from a variety of sources, placing it in a single interface.

In this thesis I describe an approach that, on the basis of an ontological representation of data and services related to a domain, supports the translation of a data query into a keyword-based service discovery process. In the next section, some Web Service discovery and retrieval approaches are presented, to present the context where the aggregated search of data and service is proposed.

1.3.3 Web Service Retrieval

The discovery of (semantic) Web services is an important open issue, that aims at the automatic detection of services offered on a network. Current discovery solutions were developed in the context of automatic service composition. Seamless composition of Web services has enormous potential in streamlining business-to-business or enterprise application integration [Srivastava and Koehler, 2003, Dustdar and Schreiner, 2005, Rao and Su, 2005]. Thus, the "client" of the discovery procedure is an automated computer program with little, if any, tolerance to inexact results. Semantics can play an important role in such a process. Semantic Web Services aim at providing a new level of functionality on top of the current Web and current services, by enabling automatic discovery, composition, invocation and interoperation of Web Services [Holger et al., 2003, McIlraith et al., 2001]. Recent standards are based on ontologies for modeling the capabilities of web services [Bergamaschi and Maurino, 2009]: examples are represented by the Web Service Modeling Ontology, WSMO [Roman et al., 2004] and OWL-S [Burstein et al., 2004]. In these standards, descriptions of services refer to external domain ontologies providing the semantic of concepts exploited in service descriptions. Service discovery is the first crucial step to compose services for executing a process. Several approaches to service retrieval applicable to OWL-S service descriptions have been proposed. The different approaches have been compared against benchmarks of the OWLS-TC family (OWLS-TC1, OWLS-TC2.2, OWLS-TC3); a summary of the result of the S3 contest is discussed in [rep, 2009].

In general, service matchmaking aims at discovering services satisfying a given set of requirements. Such requirements are usually specified in terms of types of the information exchanged by the service; in other words, matchmakers are usually targeted to find services whose Inputs (I), Outputs (O), Preconditions (P) and Effects (E) match with a set of desired I/O/P/E; the original goal is to guarantee a correct exchange of messages between the selected service and the services it is supposed to interoperate with. As a result, most of the approaches are originally based on logical I/O matching strategies, which result in different logical classes of matching, namely exact, plug-in, subsume (and subsumed-by), and logical fail. However, the need to rank services belonging to the same matching class against a given I/O specification, and the need to overcome the rigidity of logical methods lead to complement logical matching techniques with non logical, approximate ranking techniques, resulting in hybrid matchmakers. As a result, hybrid matchmakers provide more generic service retrieval capabilities, that is, provide an ordered set of services as results.

In the following the matchmakers for OWL-S that implement non pure logic based approaches, namely, OWLS-MX2 [Klusch et al., 2009], OWLS-MX3 [Klusch and Kapahnke, 2009], iMatcher2 [Kiefer and Bernstein, 2008], and Opossum [Toch et al., 2007], are briefly described.

The OWLS-MX2 and OWLS-MX3 matchmakers perform service selection based on logic-based I/O match and on non logic-based text match; text match techniques are based on syntactic similarity metrics such as the Cosine and Extended Jaccard [Kiefer and Bernstein, 2008]; the recent OWLS-MX3 matchmaker exploits also a concept semantic similarity metric. The authors also show that this metric allows to correct some false positives and some false negatives resulting from the other matching strategies. Moreover the OWLS-MX3 matchmaker has adaptive capabilities that make it learn to optimally aggregate the results of different matching filters (off-line).

iMatcher2 performs hybrid approximate matchmaking by evaluating a vector-based similarity of the unfolded service signatures (the path from the root concept and the I/O concepts), and text-based similarities of the service unfolded signatures, the service names, and the service descriptions. Moreover, the iMatcher2 creates composed matching strategies by machine learning algorithms. This approach is based on the definition of queries describing the desired features of the services (name, description, I/O).

The Opossum Matchmaker is based on methods for semantic indexing and approximate retrieval of Web services. It relies on graph-based indexing in which connected services can be approximately composed, while graph distance (shortest path distance, concept depth/average ontology depth) represents service relevance. A query interface translates a user's query, expressed

as a set of keywords, into a virtual semantic Web service, which in turn is matched against indexed services; therefore, although this is the only approach exploiting keyword-based queries, the retrieval process is based on the separate evaluation of I/O.

1.3.4 Data and Service Integration Approaches

The problem to consider together data and services is usually defined in *Data as a Service* approaches. Data are considered as a specific type of services [Truong and Dustdar, 2009] able to provide information exposing some WSDL or RESTfull [Richardson and Ruby, 2007] interface. In this approach, data services are used as part of complex and value added processes realized by Service Oriented Architectures (SOA) [Hansen et al., 2003, Zhu et al., 2004]. Another famous example of Data as a Service is the Amazon Simple Storage Service (S3) [S3, 2009] that is an online storage web service offered by Amazon Web Services. Amazon S3 provides unlimited storage through a simple service interface.

The approach presented in this thesis can be described as a *Service as Data* approach. In the Service as Data approach, informative services are considered as a kind of data source to be integrated with other data sources, to provide a global view about the information managed in a networked organizations. To the best of our knowledge this approach is completely new in the literature.

1.3.5 Data Integration and Cloud Computing

Cloud Architectures [Weiss, 2007] allows building applications that use the underlying computing infrastructure only when it is needed (for example to process a user request), draw the necessary resources on-demand (like compute servers or storage), perform a specific job, then relinquish the unneeded resources and often dispose themselves after the job is done. While in operation, the application scales up or down elastically based on resource needs.

Hadoop [Had, 2009] is a framework for running applications on large clusters of commodity hardware. Hadoop implements a computational paradigm named map/reduce, where the application is divided into many small fragments of work, each of which may be executed or reexecuted on any node in the cluster. In addition, it provides a distributed file system (HDFS) that stores data on the compute nodes, providing very high aggregate bandwidth across the cluster. Both map/reduce and the distributed file system are designed so that node failures are automatically handled by the framework.

Amazon Simple Storage Service (S3) [S3, 2009, Varia, 2009] was one of the first commercial cloud applications. Amazon S3 provides a simple web service interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the web. It is an example of how the cloud architectures are spreading because of the great advantages they offer in terms of scalability and cost effectiveness. While cloud storage applications are already quite popular, the next step will be represented by entire applications that runs on the cloud in a distributed manner. Commercial companies like Google and Windows are already using cloud architectures and offering solutions "on the cloud". Google's Bigtable [Chang et al., 2006] is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in Bigtable, including web indexing, Google Earth, and Google Finance. Another example is Windows Azure [Chappell, 2009], a platform developed by Microsoft for running Windows applications and storing data in the cloud.

Considering the amount of data that is usually involved in data integration applications, cloud architectures seem of great interest for this purpose. Very recent research projects exist aiming at developing scalable infrastructure for the integration of public data and making different data sets publicly available. Examples of projects based on Hadoop include [Dalvi et al., 2009, AST, 2009, fla, 2009]. The ASTERIX project [AST, 2009] is developing new technologies for storing, managing, indexing, querying, analyzing, and subscribing to vast quantities of semi-structured information on large clusters. The project is combining ideas from three distinct areas (semi-structured data, parallel databases, and data-intensive computing) to create a next-generation, open source software platform that scales by running on large, shared-nothing computing clusters. The Flamingo Project [fla, 2009] focuses on data cleaning aspects, developing algorithms in order to make query answering and information retrieval efficient in the presence of inconsistencies in information systems. Both projects are funded by the American National Science Foundation (NSF).

Chapter 2

The Problem of Data Integration

2.1 Data Integration Systems

Integration Systems are usually characterized by a classical wrapper-mediator architecture [Wiederhold, 1992] used to build a mediated schema of a set of data sources. The data sources store the real data, while the mediated schema provides a reconciled, integrated and virtual view of the underlying sources. Many research efforts have been devoted to modeling the mappings between the sources and the mediated schema. Two different approaches have been proposed for specifying these mappings: the Local-as-View approach (LAV) describes the data sources as views over the mediated schema; in contrast, the Global-as-View (GAV) approach describes the mediated schema as a set of view definitions over the source relations [Lenzerini, 2002, Halevy, 2001, Ullman, 1997]. The two approaches have been fused in a unique model, called Global-Local-as-View (GLAV), which synthesizes the characteristics of the two approaches [Friedman et al., 1999]. A general description is included in the following.

In this thesis I will focus on the MOMIS[1] system (Mediator EnvirOment for Multiple Information Sources), developed by the DBGroup of the University of Modena and Reggio Emilia. MOMIS is an Intelligent Data Integration framework designed for the integration of heterogeneous data sources that adopts a GAV approach. A general description of MOMIS is provided in this chapter. An open-source version of MOMIS will be soon released by the Spin-Off of the University of Modena and Reggio Emilia DataRiver[2].

[1] See `http://www.dbgroup.unimore.it` for references about the MOMIS project.

[2] `http://www.datariver.it`

2.2 The MOMIS Data Integration System

Definition of the MOMIS Integration System

Definition 1 *Given a set N of data sources to be integrated, we can define a MOMIS Integration System IS = (GS, N, M) as constituted by:*

- *A Global Schema (GS), which is a schema expressed in the ODL_{I^3} language*
- *A set N of data sources; each source has a schema also expressed in ODL_{I^3}*
- *A set M of GAV mapping assertions between the GS and N, where each assertion associates to an element G in GS a query q_N over the schemas of the set N of local sources. More precisely, for each global class $G \in GS$ we define:*
  1. *a (possibly empty) set of classes, denoted by L(G), belonging to the local sources in N*
  2. *a conjunctive query q_G over the L(G) classes*

Intuitively, the GS is the intensional representation of the information provided by the Integration System, whereas the mapping assertions specify how such an intensional representation relates to the local sources managed by the Integration System. The semantics of an Integration System is defined in [Cali et al., 2002, Beneventano and Lenzerini, 2005].

MOMIS 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}, described in Section 2.3 (see Appendix A for its syntax), 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 Schema GS) 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 possible to semi-automatically synthesize a GS that conceptualizes a domain and thus might be thought as a basic domain ontology for the integrated sources.

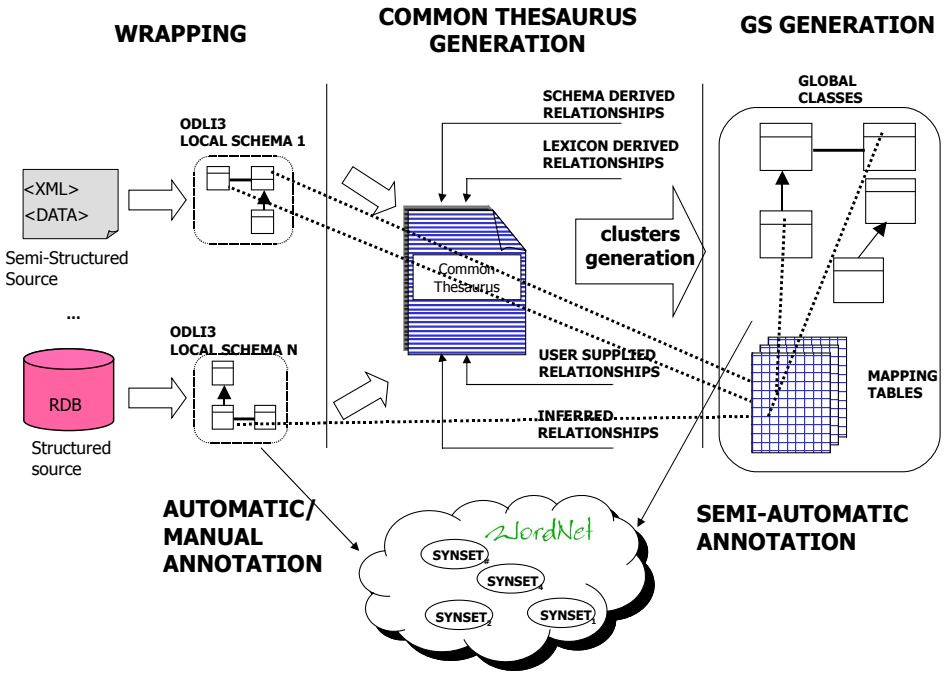


Figure 2.1: Global Schema generation process with the MOMIS system

2.3 The ODL$_{I^3}$Language

The ODL$_{I^3}$ language used in the MOMIS system to represent data is an extension of the *Object Definition Language* (ODL), an object-oriented language developed by ODMG[34]. ODL$_{I^3}$ is transparently translated into a Description Logic [Beneventano et al., 2003, Bergamaschi et al., 2001]. ODL$_{I^3}$ allows different kinds of data sources and the view resulting from the integration process to be represented in a common data model. In ODL$_{I^3}$ all the the data sources are represented as a set of classes and attributes. Moreover, some constructors and rules are present in the language to handle the heterogeneity:

Union constructor. The union constructor is introduced to express alternative data structures in the definition of an ODL$_{I^3}$ class, thus capturing requirements of semistructured data.

[3] `http://www.odmg.org/`

[4] `http://www.service-architecture.com/database/articles/odmg_3_0.html`

Optional constructor. The optional constructor is introduced for class attributes to specify that an attribute is optional for an instance (i.e., it could be not specified in the instance).

Integrity constraint rules. This kind of rule is introduced in ODL_{I^3} in order to express, in a declarative way, *if then* integrity constraint rules at both intra- and inter-source level.

Intensional relationships. They are *terminological relationships* expressing intra- and inter-schema knowledge for the source schemas. Intensional relationships are defined between classes and attributes, and are specified by considering class/attribute names, called terms. The following relationships can be specified in ODL_{I^3}:

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

An intensional relationship is only a terminological relationship, with no implications on the extension or compatibility of the structure (domain) of the two involved classes (attributes).

Extensional relationships. Intensional relationships SYN, BT and NT between two classes C_1 and C_2 may be "strengthened" by establishing that they are also *extensional* relationships.

Mapping Rules. This kind of rule is introduced in ODL_{I^3} in order to express relationships holding between the integrated ODL_{I^3} schema description of the information sources and the ODL_{I^3} schema description of the original sources.

Using ODL_{I^3} for representing sources and ontologies is not a limitation: the interoperability of the ODL_{I^3} descriptions is guaranteed by a software module able to translate the descriptions into the Web Ontology Language OWL [Orsini, 2004].

2.4 Global Schema Generation with MOMIS

The integrated global schema 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 class attributes and the source schemata.

The process of creating the GS and defining the mappings, shown in figure 2.1, can be summarized in the following steps:

Local source schemas extraction The first phase of the integration process is the choice of the data sources and their translation into the ODL_{I^3} format. The translation process is performed by the MOMIS wrappers, which logically converts the source data structure into the ODL_{I^3} model. The wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources.

Lexical annotation of local sources The goal of the annotation phase is to semantically annotate terms denoting schema elements in data sources according to a common lexical reference, which means to assign a shared meaning to each of them. The WordNet lexical database is usually referred to as lexical reference [Fellbaum, 1998], but other lexical references might be used to cope with specific domains. The annotation step of the schema elements, pre-processed by the system applying stop-words and stemming techniques, can be performed manually by the integration designer, or automatically by the MOMIS system. The manual annotation is composed of two different steps: in the Word Form choice step, the WordNet morphological processor suggests a word form corresponding to the given term; in the Meaning choice step, the designer can choose to map an element to zero, one or more senses. In MOMIS the annotation step can also be performed automatically combining a set of Word Sense Disambiguation algorithms [Bergamaschi et al., 2007]: SD (Structural Disambiguation) [Bergamaschi et al., 2008], WND (WordNet Domains Disambiguation) [Bergamaschi et al., 2008], WordNet first sense heuristic, Gloss Similarity [Beneventano et al., 2008b] and Iterative Gloss Similarity [Beneventano et al., 2008b]. For further details about the annotation techniques in MOMIS see [Po, 2009].

Common thesaurus generation Starting from the annotated local schemata, MOMIS constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of SYN(synonyms), BT/NT(broader

terms/narrower terms), and RT(meronymy/holonymy) relationships. The Common Thesaurus is constructed through an incremental process in which the following relationships are added:

- schema-derived relationships: relationships holding at intra-schema level are automatically extracted by analyzing each schema separately. Some heuristic can be defined for specific kind of sources. For example, MOMIS extracts intra-schema RT relationships from foreign keys in relational source schemas. When a foreign key is also a primary key, in both the original and referenced relation, MOMIS extracts BT and NT relationships, which are derived from inheritance relationships in object-oriented schemas.
- lexicon-derived relationships: the annotation phase is exploited to translate relationships holding at the lexical level into relationships to be added to the Common Thesaurus. These relationships may be inferred from lexical knowledge (e.g. by querying WordNet for relationships between senses).
- designer-supplied relationships: new relationships can be supplied directly by the designer, to capture specific domain knowledge.
- inferred relationships: Description Logics (DL) techniques of ODB-Tools [Bergamaschi et al., 1997], are exploited to infer new relationships.

Global Schema Generation The Global virtual Schema (GS) consists of a set of classes (called Global Classes), plus mappings to connect the global attributes of each Global Class and the local source attributes. The MOMIS methodology allows identifying similar ODL_{I^3} classes (i.e. classes that describe the same or semantically related concept in different sources) and mappings to connect the global attributes of each global class to the local source attributes. To this end, affinity coefficients are evaluated for all possible pairs of ODL_{I^3} classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name Affinity coefficient*) and their attributes (*Structural Affinity coefficient*) and are fused into the *Global Affinity coefficient*, calculated by means of the linear combination of the two coefficients [Castano et al., 2001]. Global affinity coefficients are then used by a hierarchical clustering algorithm, to cluster ODL_{I^3} classes according to their degree of affinity. For each cluster C, a Global Class G, with a set of Global Attributes $GA_1, \ldots GA_N$, and a Mapping Table MT,

expressing mappings between local and global attributes, are defined. The Mapping Table is a table whose columns represent the local classes which belong to the Global Class and whose rows represent the global attributes. An element $MT[GA][LC]$ is a function which represents how local attributes of the Local Class LC are mapped into the global attribute GA:

$$MT[GA][LC] = f(LAS)$$

where LAS is a subset of the local attributes of LC.

GS lexical annotation To annotate a GS means to assign a name and a set (eventually empty) of meanings to each global element (class or attribute). MOMIS automatically annotates each global element proposing the broadest meaning extracted from the annotations of the local sources, based on the relationships included in the Common Thesaurus. Names and meanings have then to be confirmed by the ontology designer. This annotation step is a significant result, since these metadata may be exploited by external users and applications.

2.4.1 Mapping Refinement

After the GS generation process, each global class GC is associated to a Mapping Table. Starting from the Mapping Table of GC, the integration designer can implicitly define the mapping query q_G associated to the global class G by:

1. extending the Mapping Table with
   - Data Conversion Functions from local to global attributes
   - Join Conditions among pairs of local classes belonging to G
   - Resolution Functions for global attributes to solve data conflicts of local attribute values.
2. using and extending the *full outerjoin-merge* operator, proposed in [Naumann et al., 2004], to solve data conflicts of common local attribute values and merge common attributes into one.

Data Conversion Functions

The Ontology Designer can define, for each not null element $MT[GA][LC]$, a Data Conversion Function, denoted by $MTF[GA][LC]$, which represents the mapping of local attributes of LC into the global attribute GA.

$MTF[GA][LC]$ is a function that must be executable/supported by the class LC local source. For example, for relational sources, $MTF[GA][LC]$ is an SQL value expression. $T(LC)$ denotes L transformed by the Data Conversion Functions. Further details about MOMIS data conversion functions can be found in [Beneventano et al., 2007].

Join Conditions

Merging data from different sources requires different instantiations of the same real world object to be identified; this process is called object identification [Bleiholder and Naumann, 2008]. To identify instances of the same object and fuse them we introduce Join Conditions among pairs of local classes belonging to the same global class. Given two local classes L_1 and L_2 belonging to G, a Join Condition between L_1 and L_2, denoted with $JC(L_1, L_2)$, is an expression over $L_1.A_i$ and $L_2.A_j$ where A_i (A_j) are local attributes with a not null mapping in L_1 (L_2).

Resolution Functions

In MOMIS, the approach proposed in [Naumann and Häussler, 2002] has been adopted: a Resolution Function for solving data conflicts may be defined for each global attribute mapping onto local attributes coming from more than one local source; in this way we can define what value should appear in the result. 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). In MOMIS we use conflict resolution strategies based on Resolution Functions as introduced in [Naumann and Häussler, 2002, Bleiholder and Naumann, 2005]: 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 $L_1.B$ and $L_2.B$ are numerical attribute, we can define $G.B = avg(L_1.BandL_2.B)$. The MOMIS system provides some standard kinds of resolution functions (Random, Aggregation, Coalescence and others)[Beneventano et al., 2007, Orsini, 2009].

Full Outer join-merge operator

The *full outer join-merge* operator, proposed in [Naumann et al., 2004], is used to define the mapping query q_G. For every Global Class G, mapped on

a set L of local classes $L_i \dots L_j$, the *full outer join-merge* operator guarantees that every tuple from all the local sources enters the result. As an example, referring to a Global Class G, mapped on two local classes L_1 and L_2, the join-merge operator returns tuples for G that are joined from tuples in L_1 and L_2 using the join condition specified by the integration designer. Missing values in attributes of tuples that do not have a matching tuple in the other source are padded with null values. For all attributes exclusively provided by L_1, the values of L_1 are used, for all attributes exclusively provided by L_2 , the values of L_2 are used. For common attributes, the join-merge operator applies the resolution functions defined before to determine the final value. The values of all other attributes of G are padded with null values. Further details about the use of the *full outer join-merge* operator can be found in [Orsini, 2009].

2.5 Query Execution

The MOMIS Query Manager allows the user to pose a query expressed in OQL [Beneventano et al., 2003, Orsini, 2009] over the ontology and to obtain a unified answer from all the data sources integrated in the GS. When the MOMIS Query Manager receives a query, it rewrites the global query as an equivalent set of queries expressed on the local schemas (local queries); this query translation is carried out by considering the mapping between the GS and the local schemata. The query translation is thus performed by means of query unfolding, i.e. by expanding a global query on a global class G of the GS according to the definition of the mapping query q_G. The local queries are then executed on the sources, and MOMIS performs the fusion of the local answers into a consistent and concise unified answer, and present the global answer to the user. In order to assure full usability of the system even to users with low information technology skills, that are often the target of many information integration application, a graphical user interface to compose queries over the MOMIS GS was developed. This interface was initially developed as part of a MOMIS application presented in Chapter 3 and is thus presented in details in Section 3.6. This interface was then made available as a MOMIS component and can thus be automatically used with any MOMIS schema.

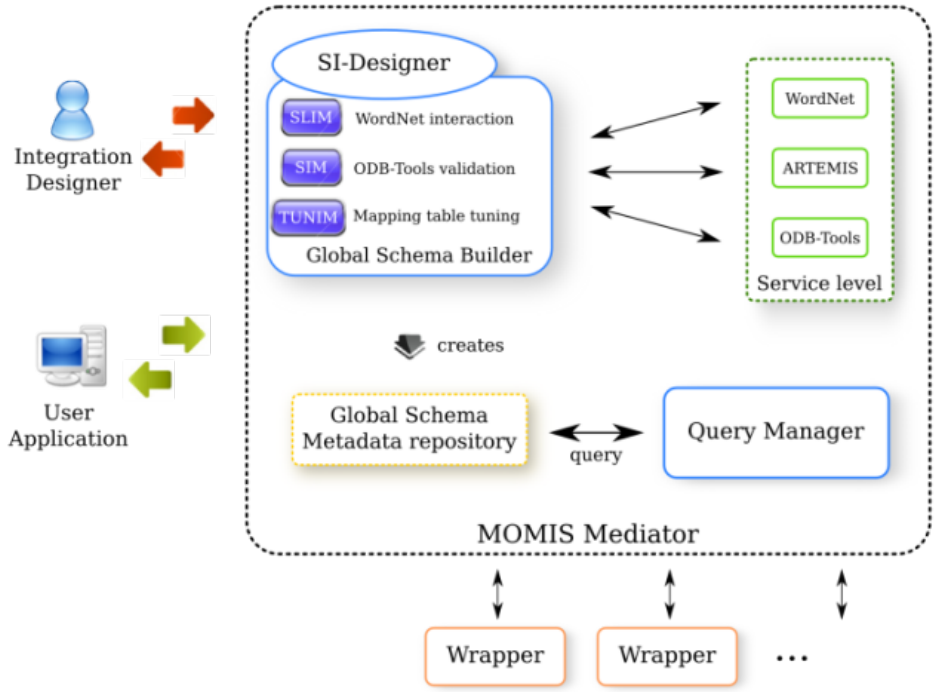


Figure 2.2: MOMIS Architecture

2.6 MOMIS Architecture

In this section we briefly describe the architecture of the MOMIS system, shown in Figure 2.2. Further details can be found in [Orsini, 2009].

The main components of MOMIS are:

- **Wrappers:** Wrappers are software modules with the role of managing the interactions with the data sources. The MOMIS wrappers translate the source data structures into ODL_{I^3}. Their role is to deal with the diversity of the data sources thus allowing MOMIS to pay no attention to the language details of the different data sources. Wrappers are available for different kind of data sources, ranging from different database management systems to semi-structured data like XML, RDF, OWL formats. Wrappers logically guarantee two main operations:
  - *getschema()* translates the schema from the original format into ODL_{I^3}, dealing with the necessary data type conversions
  - *runquery()* executes a query on the local source. The MOMIS Query Manager translates a query on the GS (a global query) into a set of local queries to be locally executed by means of wrappers.

- **Global Schema Builder:** The Global Schema Builder is the main module that interacts with the wrappers and the Service tools to generate the Global Schema. It is composed of different components that implements the different steps of the integration process described in Section 2.4.

- **Service tools:** the Service tools are important modules exploited by the Global Schema Builder and the Query Manager. These include the software module in charge of managing the WordNet lexical database used for the annotation phase, the Artemis tool implementing the clustering algorithm, and the ODB-Tools reasoner used to calculated inferred relationships.

- **Query manager:** the Query Manager is the component in charge of solving a user query: it generates the local queries queries for wrappers, starting from a global query formulated by the user on the global schema. The Query Manager provides a graphical Query Composer to graphically formulate queries on a Global Schema described in details in Section 3.6.

Chapter 3

An Integrated Ontology for Molecular and Phenotypic Cereal Data

In this chapter the application of the MOMIS system in the bioinformatics domain is presented. This work was realized in the context of the Cerealab project[1], conducted by the "Department of Agrarian and Food Sciences" of the University of Modena and Reggio Emilia in collaboration with the Database Group of the University of Modena and Reggio Emilia and funded by the Regional Government of Emilia Romagna in Italy. The aim of the Cerealab project is to increase competitiveness of Regional seed companies through the use of modern selection technologies making available to the cereal breeders the knowledge learnt in the research activity by the Universities. In particular, this work aimed at providing breeders with a tool to perform genotypic selection of cereal cultivars from phenotypic traits through the Marker Assisted Selection (MAS)[Kelly and Miklas, 1999].

The objective of this work was the development of the Cerealab database, an ontology of molecular and phenotypic cereals data, wheat, barley and rice in particular, realized by integrating existing public web databases with the database developed by the research group of the Cerealab project. The integration process was performed with the MOMIS system and resulted in a Global virtual Schema of the underlying data sources that researchers and breeders can query transparently with regards to integrated data sources using an easy to use graphical interface regardless of the specific languages of the source databases. The work presented in this chapter was published in [Bergamaschi and Sala, 2006, Bergamaschi and Sala, 2007] and [Sala and Bergamaschi, 2009].

3.1 Motivations and Related Works

The most important objective of cereal breeding programs all over the world is to combine high yield with high quality and disease resistance. In classical plant breeding the phenotype of a plant is assessed by different measuring (yield, resistance, etc.) and superior phenotypes are assumed to be the result of superior genotypes. Selection for traits that show a continuous range of values, such as yield, plant height or days to flowering is tedious, because the relation between observed trait values (phenotype) and the underlying genotype is not straightforward as quantitative traits are typically controlled by many genes and each gene contributes only a small part to the observed variation. Although classical breeding practices have been very successful in producing a range of improved varieties, the developments in the field of biotechnology and molecular biology can be employed to enhance plant breeding efforts and to speed up the creation of cultivars. Molecular markers

[1] http://www.cerealab.org

have become an important tool in plant breeding and several areas of application such as Marker Assisted Selection (MAS) or genetic fingerprinting for plant variety protection or agri-food traceability programs. The demands for the storage of genotyping data is thus increasing due to the speed at which numerous markers are being developed.

Public Biological Data Sources In the last few years numerous public data sources have been realized and are now available for researchers in the field of molecular biology. For maize, the MaizeGDB database [Lawrence et al., 2004], is a repository for sequences, information on stock, phenotype, genotypic variation and mapping data, Panzea [Zhao et al., 2006] is a database and resource for molecular and functional diversity in the maize genome. The GrainGenes [Matthews et al., 2003] database focuses on grass and cereals storing both genotypic and phenotypic data. Gramene [Jaiswal et al., 2006] is a comparative genome mapping database for grasses, using the rice genome as an anchor. Other databases have been developed for managing molecular and phenotypic information in other plants such as the Arabidopsis Hormone Database [Peng et al., 2008]. However most of these databases manage genomic and phenotypic data separately. Recently some databases designed to store and manage both phenotypic and genotyping data have been reported: AppleBreed [Antofie et al., 2007] for perennial crops, PlantDB [Exner et al., 2008], Germinate [Lee et al., 2005]. PlantDB and Germinate are very versatile databases designed to manage plant genetic resource collections and to integrate genotypic and phenotypic information and can be applied to many species, although in these resources the data available are restricted to those provided by the developers.

Manual Integration of Public Biological Data Sources Even if many information sources are available for the research community regarding molecular data as well as phenotypic data, the problem is that marker-assisted selection requires having access to different kind of data, which usually reside on many different databases. Accessing, understanding and composing these data usually require long time and deep domain knowledge. In fact, the user has to query different databases and to combine manually the data he or she obtains. Especially for some small breeders that lack a thorough preparation as far as molecular biology is concerned, coping with such a great quantity of information may be difficult. Furthermore, the data obtained from different research projects should have the same general structure to favor the exchange of information. These data are repeatedly stored in different locations and a considerable time is needed to convert data stored

in a certain format into another format that can be input in larger databases and combining data from different experiments for a joint analysis is difficult. The general goal of the scientific community should be to integrate all the data available to help less experienced users to exploit the great amount of data already available, and to provide a unique access to all the information needed by cereal breeders to perform marker-assisted selection.

Automatic Integration of Molecular and Phenotypic Data Sources The objective of the Cerealab database is to reuse the already available data and combine them with those obtained by genotyping activity of the Cerealab research group in order to create a tool for plant breeders and geneticists. The database can help them in unraveling the genetics of economically important phenotypic traits; in identifying and choosing molecular markers associated to key traits; in identifying alleles of such markers associated to positive variants of traits; and in choosing the desired parentals for breeding programs.

The Cerealab Database represents a Global virtual Schema, i.e. a unique ontology that integrates existing public data sources providing both molecular and phenotypic data about wheat, barley and rice. Moreover, the ontology has to easily integrate new molecular data coming from the research activity of the Cerealab project.

As far as we know, no resource is available containing both these two kinds of data for the purpose of this project. For this reason, we developed a Global virtual Schema (GS) which is the integration of existing molecular and phenotypic data sources with data provided by the Cerealab project. The GS can be seen as an ontology of the underlying sources.

Other ontologies about these domain exist, but none of these correlates phenotypic data with molecular data. For example the Trait Ontology (TO)[2] is a controlled vocabulary that describes each trait as a distinguishable feature, characteristic, quality or phenotypic feature of a developing or mature individual. The TO partially covers our domain of interest, and thus has been used as a reference. Our ontology overcomes the TO as it integrates the trait ontology with molecular data related to phenotypic data.

Related Works on Biological Data Integration Systems In the last few years the problem of data integration for biology has become really important both due to continuous increases in data volumes and the growing diversity in types of data that need to be managed. For example the Transparent Access to Multiple Bioinformatics Information Sources project, known

[2] `http://www.gramene.org/plant_ontology/`

as TAMBIS [Stevens et al., 2000], is a mediator-based integration system in which a domain ontology for molecular biology and bioinformatics is used in a retrieval-based information integration system for biologist. TAMBIS uses the global ontology to formulate queries through a graphical interface where a user needs to browse through concepts defined in a global schema and select the ones that are of interest for the particular query. TAMBIS can seem similar to our approach but, in this system, the global schema is constructed manually, while in MOMIS clusters of similar classes and mappings of global schema classes with local schemas are automatically generated once the sources have been imported and semi-automatically annotated. The process of generation of the Global Schema is thus semi-automatic.

BioKleisli [Davidson et al., 1997] is primarily a loosely-coupled federated database system. The mediator on top of the underlying integration system relies mainly on a high-level query language (the Collection Programming Language, or CPL) more expressive than SQL that provides the ability to query across several sources. The BioKleisli project is mainly aimed at performing a horizontal integration. In fact, a query attribute is usually bound to an attribute in a single predetermined source; there is essentially no integration of sources with content overlap. Furthermore, no optimization based on source characteristics or source content is performed. K2 [Davidson et al., 2001] is the newer version of the BioKleisli system. K2 abandons CPL and replaces it by OQL, a more widely used query language. This change does not modify the overall flow of the system. Queries are still decomposed into subqueries and sent to the underlying sources using data drivers, while the query optimizer remains a rule-based optimizer.

DiscoveryLink [Haas et al., 2001] is a mediator-based and wrapper-oriented middleware integration system. It serves as an intermediary for applications that need to access data from several biological sources. Applications typically connect to DiscoveryLink and submit a query in SQL on the global schema, not necessarily aware of the underlying sources. These two systems offer format and location transparency but do not hide the sources and do not offer schema or data reconciliation.

A survey of these and some other well-known systems that are currently available can be found in [Hernandez and Kambhampati, 2004].

As it can be seen, the data integration problem for biology has been addressed in numerous way, but as far as we know the approach presented in this thesis is the first one that combines molecular and phenotypic data in an integrated ontology (all the other systems integrates only molecular data sources). Moreover, except TAMBIS, the existing systems usually uses the SQL language to formulate queries, while in our system I developed a graphical interface for query formulation which is considered a necessity as

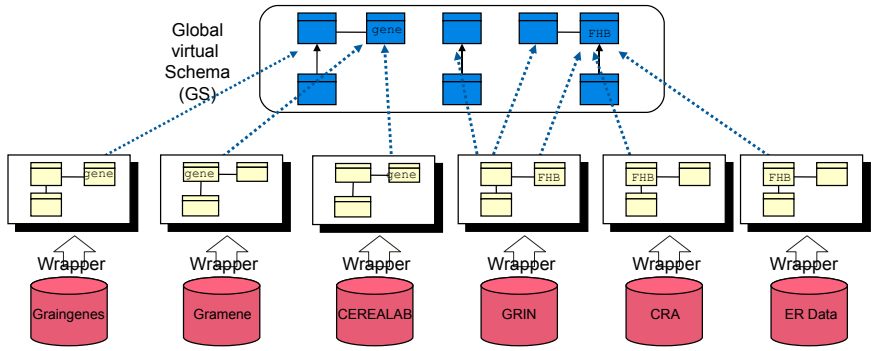


Figure 3.1: Creating the Global (virtual) Schema with the MOMIS System

the users of this kind of systems have low IT expertise and thus need a user-friendly system (see Section 3.6).

The rest of this chapter is organized as follows: Section 3.2 describes the terminology of Cerealab project domain, while in Section 3.3 the data sources involved in the integration process are presented. Section 3.4 describes the application of the MOMIS data integration process to the data sources involved. Section 3.5 describes the resulting integrated ontology while Section 3.6 sketches out the querying process with the MOMIS Query Manager and presents the graphical Query Composer developed to graphically formulate queries over the integrated ontology.

3.2 Description of the Cerealab Domain

To facilitate the comprehension of the terms involved in our project and used in the rest of the chapter, I provide in this section a brief description of the domain of the Cerealab project and of the data sources to be integrated. The main entities about molecular data are three:

- **Gene:** it is the unit of heredity in living organisms consisting of a sequence of DNA that occupies a specific location on a chromosome and determines a particular characteristic in an organism.

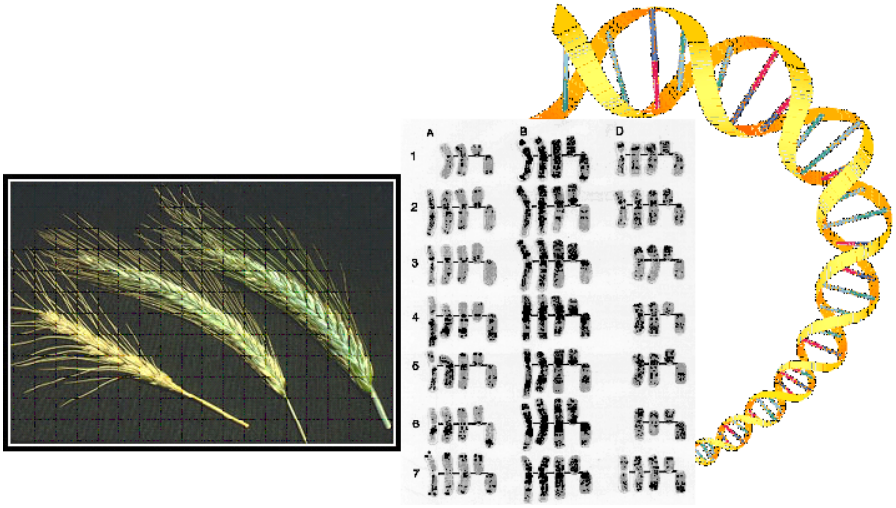
- **QTL:** a quantitative trait locus, it is a region of DNA that putatively contains one or more genes associated with a trait. Though not necessarily genes themselves, QTLs are stretches of DNA that are closely linked to the genes that underlie the trait in question. A Trait is considered as any single feature or quantifiable measurement of an organism.

The Trait is an inherited feature of a plant, and is thus influenced by Genes and QTLs.

- **Marker:** it is a known DNA sequence (e.g. a gene or part of gene) that can be identified by a simple assay, associated with a certain phenotype. A genetic marker may be a short DNA sequence, such as a sequence surrounding a single base-pair change, or long one, like microsatellites. They are used to 'flag' the position of a particular gene or the inheritance of a particular characteristic.

All these entities have their own specific attributes, such as its chromosome, which is a physically organized piece of DNA that contains Genes or QTLs; or its Allele, which is any one of a number of viable DNA codings that occupies a given locus (position) on a chromosome.

The term Germplasm identifies an assemblage of plants that has been selected for a particular attribute or combination of attributes and is clearly distinct, uniform and stable in its characteristics.



3.3 Description of the Data sources

The choice to integrate existing data was driven by the fact that part of the necessary data is already available to the research community. The data sources for the Cerealab database have been chosen among those particularly relevant for MAS of wheat (Triticum aestivum and Triticum turgidum

ssp. durum), barley (Hordeum vulgare) and rice (Oryza sativa). As far as molecular data are concerned, the **Gramene** database[3] was used as a data source for rice and **GrainGenes**[4] for wheat and barley. GrainGenes is the international database for wheat, barley, rye and oat. For these species it constitutes the main and primary repository for information about genetic maps, genes, QTLs, markers, primers and alleles. GrainGens was created by the US Department of Agriculture as a tool for breeders, pathologists, geneticists and molecular biologists, aimed at providing them with an access to all available information and help them to create improved cultivars of Triticeae (wheat, barley, rye, and triticale) and oat crops. A major focus of interest for all these groups of researchers is genetic mapping of molecular markers, genes and QTLs. The overall goal of this database is to represent a central point for both retrieving and submitting information regarding cereal genetics and biology [Matthews et al., 2003]. Gramene is another, independent, but anyway closely related database that is important for many of GrainGenes users. It is a comparative genome mapping database for grasses, using the rice genome as an anchor, taking advantage of the known genetic colinearity between rice and major crop plant genomes to provide researches about other species with the benefits of an annotated genome. It combines and interrelates information on the structure and organization of genomes and genes, functions of proteins, various maps (genetic, physical and sequence), mapped markers, QTLs and literature citations. Gramene is available on-line and allows downloading the dump of part of or the complete relational database. The MOMIS system allows integrating distributed data sources that can be accessed remotely by wrappers, since the only requirement is having read access to the data. In fact, its virtual approach assures the independence of the data sources thus not affecting them for the integration purpose. However, I decided to create a local copy of Gramene to be integrated in Cerealab, avoiding possible bottlenecks caused by having a remote data source, but it is maintained completely independent and no modification was made on it. GrainGenes, like Gramene, is available on-line and can be queried by means of web forms, but it allows also posing SQL queries instead of using predefined forms. This option leaves great freedom to users with knowledge of the SQL language, but can be difficult for those with only basic computer science background. Differently from Gramene, GrainGenes cannot be downloaded entirely, but we were given this possibility contacting the curators. Therefore, I restored GrainGenes locally on our server as well, still maintaining it independent. These databases were integrated with

[3] `http://www.gramene.org`
[4] `http://www.graingenes.org`

another molecular data source represented by a relational database created with data obtained from a systematic genotyping work performed by the research group of the Cerealab laboratory.

As far as phenotypic evaluations are regarded, the **Germplasm Resources Information Network (GRIN)**[5] was used for wheat and barley descriptors, available on-line as CSV files; public data of the **Agricultural Research Council (CRA)** national field trials for wheat, barley and rice, public data of the **Emilia-Romagna variety field trials** for wheat and barley and public data of the **Ente Risi**[6] field trials for rice were collected by the Cerealab researchers to create a local repository of italian pheotypic data.

All the above mentioned data sources, if considered separately, present incomplete and sometimes overlapping information for the purpose of the Cerealab project.

3.4 The MOMIS process for the Cerealab Database

The MOMIS process described in Chapter 2 was applied to the above mentioned data sources. The steps of the process for building the Cerealab Global Schema are here presented and were published in [Bergamaschi and Sala, 2006, Bergamaschi and Sala, 2007]. The Cerealab database is available at `www.cerealab.org`.

Local source schemas extraction In our case wrapping the source schemas was a straighforward process as the sources are all relational databases and MOMIS provides wrappers for most of the Relational Database Management Systems (RDBMS). In this case the MOMIS SqlServer and the MySQL wrappers were used to manage all the sources involved.

Lexical annotation of local sources The annotation step is crucial to leverage the lexical knowledge of the schemas in the GS generation. Thus, in this phase, we had to assign a name and a set of meanings belonging to the WordNet lexical system to each local class and attribute of the local schemas. Considering the very specific domain of this application, we had to widely extend WordNet with a large number of new very specific terms. Guided

[5] `http://www.ars-grin.gov/`
[6] `http://www.enterisi.it`

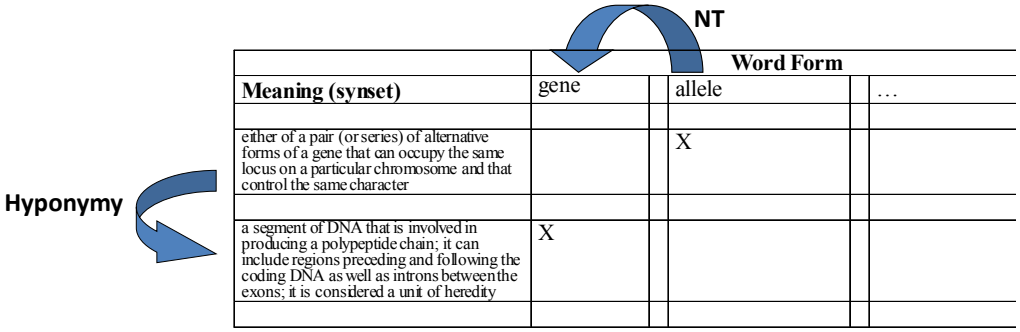


Figure 3.2: The Annotation step in MOMIS

by the domain experts of the Cerealab research group, I used the WordNet Editor [Benassi et al., 2004] provided by the MOMIS system to face such a specific domain.

As an example, WordNet provides for the term "Gene" the meaning: *"a segment of DNA that is involved in producing a polypeptide chain; it can include regions preceding and following the coding DNA as well as introns between the exons; it is considered a unit of heredity"*, which is correct. But the term "Marker" is present in WordNet as *"some conspicuous object used to distinguish or mark something"* which is obviuosly not the right meaning. For this term we added the correct meaning for this domain that is *"A segment of DNA with an identifiable physical location on a chromosome for any feature that has been genetically mapped"*.

This extension step has to be performed just the first time a domain is handled, since the capability of the WordNet Editor to cache the annoted terms and synsets allows the integration designer to reuse the new terms and meanings added to WordNet to annotate other source schemas. The annotation phase was thus quite a time consuming task only for the first data source schema since many of the terms recurred in the other source schemas.

Common thesaurus generation Figure 3.2 shows lexical annotation of the two terms *Gene* and *Allele* according to WordNet. A hyponymy relationship exists between the associated WordNet meanings of these two terms, thus leading to the generation of a NT relationship between the two terms in the Common Thesaurus. Table 3.1 shows some relationships automatically extracted by MOMIS for the *Cerealab.Gene* class. In our case, the relationships holding at lexical level are widely exploited to discover semantic rela-

tionships between local classes. Being so specific the domain we were facing, the annotation phase gave rise to a large number of BT/NT relationships in the Common Thesaurus.

Source	Type	Destination
cerealab.gene	SYN	graingenes.gene
cerealab.gene	BT	graingenes.gene.allele
cerealab.gene	BT	graingenes.chromosome
cerealab.gene	RT	graingenes.qtl
cerealab.gene	BT	graingenes.marker
cerealab.gene	RT	graingenes.marker.probe
...	...	...

Table 3.1: Some Relationships in the Common Thesaurus for the class *Cerealab.Gene*

Global Schema Generation The large number of BT/NT relationships forced us to give less weight than usual to these kind of relationships when calculating the Name Affinity and the Structural Affinity coefficients defined in Section 2.4. In particular, we tuned the weights in order to consider SYN relationships as the most relevant. In this way the clustering algorithm gave rise to a set of significant Global Classes that reflected the entities defined in Section 3.2. For each global class MOMIS automatically created a Mapping Table MT. As an example, the Mapping Table of the Global Class *Gene* is presented in Table 3.2.

No data conversion functions are needed as all the data types appearing in the different sources are homogeneous. The identification of Join Conditions is relatively easy as in this domain each of the main entities is usually identified by a specific name or symbol, and this attribute is always present in the sources. As an example, the join condition for the *Gene* Global Class, mapped on the three sources Gramene, Graingenes and Cerealab, is defined as follow:

```
((graingenes.gene.name) = (cerealab.gene.name)) AND
(((gramene.gene.name) = (cerealab.gene.name))
OR ((gramene.gene.name) = (graingenes.gene.name)))
```

Concerning Resolution Functions, the following functions had to be applied:

Gene (Global)	**Gene (Cerealab)**	**Gene (Gramene)**	**Gene (Graingenes)**
allele	allele	allele	allele
chromosome	chromosome	chromosome	chromosome
comment	comment		comment
geneclass	geneclass	geneclass	
germplasm	germplasm	germplasm	germplasm
locus	locus	locus	locus
name	gene_name	name	name
reference	reference	title	reference_title
sequence		sequence	sequence
species		species	species

Table 3.2: Mapping Table of the Gene Global Class

1. *Precedence function*: experimental results obtained by the Cerealab research group regard mainly italian cultivars. These data could be different from data from other sources, especially referring to phenotypic information since the Gramene and Graingenes are american databases. For this reason a precedence function was used, to give priority to the Cerealab data as they are related to italian cultivars.

2. *All Values*: considering the integration viewpoint, the aim is to preserve all the information coming from the sources. For example, a "reference" attribute is often present for many entities to provide bibliographic references. Sometimes each source can cite different relevant references for the instance in exam. To let the user get all the data provided by the local sources as a result, we used an *All Values* function provided by MOMIS to return all the references present in the different sources.

GS lexical annotation The Cerealab Global Schema was automatically annotated by MOMIS to provide a shared meaning to its elements. In this way the GS can be seen as an ontology of the underlying sources. This ontology is the most relevant result of this work as it is the first of its kind, as described in details in the next section. An OWL version can also be obtained as MOMIS provides OWL import and output functionalities.

3.5 The Cerealab Ontology

The Cerealab GS obtained from the integration process can be seen as ontology of the underlying data sources automatically obtained exploiting the semantics of the annotation of the data sources integrated. The valuable element of this ontology is the combination of existing phenotypic and genotypic data sources with data coming from the Cerealab project. A different schema was realized for each species, i.e. wheat, barley and rice. The three schemas are nearly the same, but each one contains data related to its specific species. Part of the Triticum ontology can be seen in Figure 3.3. The schema is divided in two parts or sub-ontologies: *Genotypic data* and *Phenotypic data*. The sub-ontology Genotypic data contains the main classes related to molecular data, namely *Trait*, *Gene*, *QTL* and *Marker*. The attributes of the classes *Trait*, *Gene*, *QTL* and *Marker* store facts about these entities. For example the information that can be retrieved for the class *Gene* includes its *name*, *gene class* in which it was classified, *germplasm* in which this gene was identified, *chromosome* on which it was mapped, its *alleles*, *sequence* where available and the scientific *references*. The markers can be *marker_for* instances of the classes *Gene* or *QTL*. Each *trait* can be affected by one or more Genes or QTLs.

Which gene or QTL underlies a certain trait can be found in the classes *Trait_affected_by_gene* and *Trait_affected_by_qtl*. For each gene and QTL it is possible to retrieve the germplasm in which it was identified in the classes *Gene_in_Germplasm* and *QTL_in_Germplasm* respectively. Those classes are reifications of the relationships expressing the identification of a gene or QTL in a Germplasm. The relationships among markers and genes and QTLs are reified in the classes *Marker_for_gene* and *Marker_for_QTL* respectively. Moreover, information about which germplasm the markers have been tested on (resulting from the genotyping activity conducted within the Cerealab project) are available in the class *Marker_tested_on_Germplasm*. The *Phenotypic data* sub-ontology contains field evaluations data for each trait. The data are divided into six super-classes chosen among those of major interest for cereal breeders: *Abiotic Stress*, *Biotic Stress*, *Anatomy and Morphology Related*, *Growth and Development*, *Quality*, and *Yield*. Each super-class is divided into several sub-classes, e.g. *Biotic Stress* has been divided into 14 sub-classes, such as *(resistance to) Stem Rust*, *Powdery Mildew*, *Fusarium Head Blight (FHB)* and others. Most classes are further divided according to different distinct phenotypic data sources (GRIN for international, Emilia-Romagna, CRA and Enterisi evaluation data for Italian germplasm collections). In Fig.3.3 only the *Biotic Stress* super-class is reported for the sake of readability.

This ontology allows to correlate the molecular data of Gramene, Graingenes and the Cerealab project with the phenotypic data of the GRIN database and those collected by the Cerealab project. In this way, it is possible to find the specific molecular markers that can identify genes or QTLs that express a particular phenotypic trait. In this way genotypic selection of cereals cultivars can be performed starting from phenotypic data.

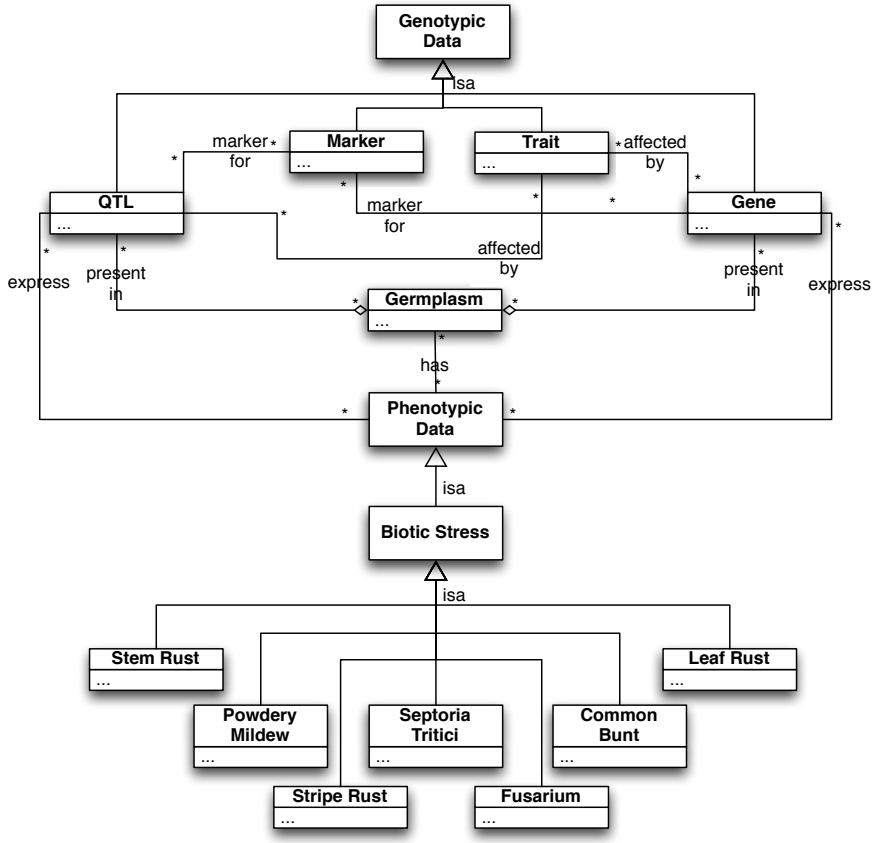


Figure 3.3: An excerpt of the Cerealab Ontology

3.6 Querying the Cerealab Ontology

An important requirement we addressed in this work is usability: as this ontology is a working tool for users with high domain knowledge but very low IT expertise, it follows that the usage of the system has to be as much user-friendly as possible, and it is necessary to provide the users with a graphical

interface to query this ontology. The MOMIS Query Manager allows the user to pose a query over the ontology and to obtain a unified answer from all the data sources integrated in the GS [Beneventano and Bergamaschi, 2007]. When the MOMIS Query Manager receives a query, it rewrites the global query as an equivalent set of queries expressed on the local schemas (local queries); this query translation is carried out by considering the mapping between the GS and the local schemas as described in Section 2.5. Considering the users the result of this work was targeted to, and to assure full usability of the system even to users who do not know anything of query languages, a graphical Query Composed has been developed to formulate queries over the GS.

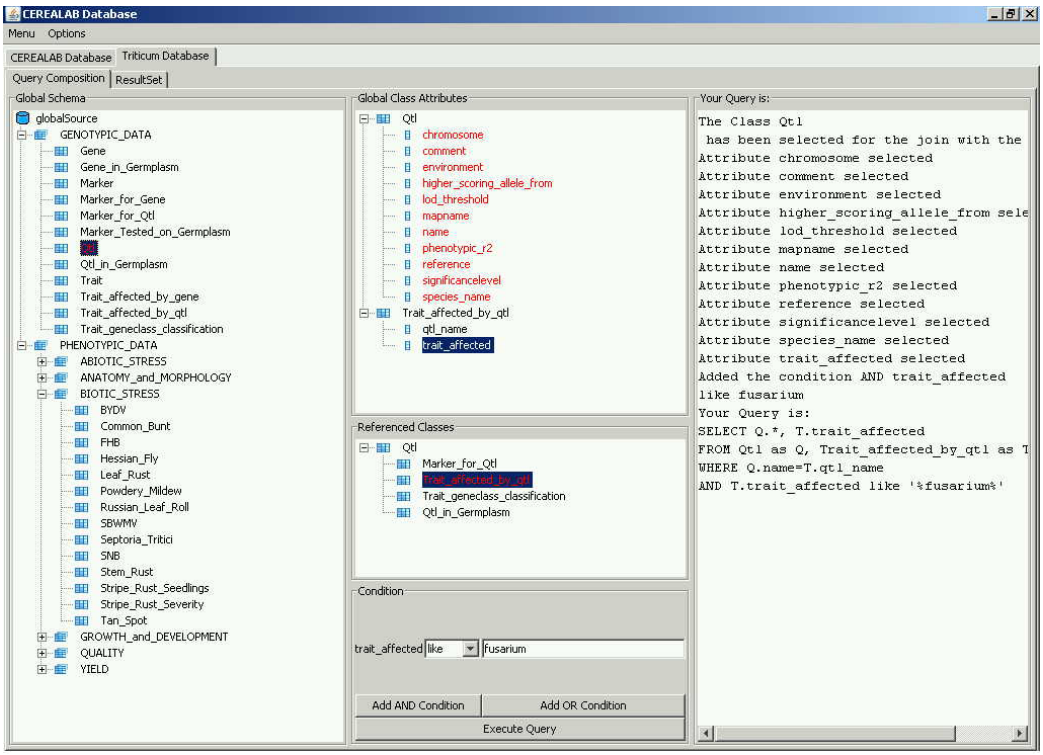


Figure 3.4: The Query Composer for the MOMIS Query Manger

This interface, shown in Figure 3.4, presents the ontology in a tree representation. The user can select the global classes to be queried; attributes are shown in the "Global Class Attributes" panel and may be selected with a simple click. Then the attributes of interest can be selected, specifying, if necessary, a condition in the "Condition" panel with the usual logic expressions. 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 classes (as it is automatically inserted). Selections and conditions specified by the user are then automatically translated into a query to be sent to the MOMIS Query Manager which rewrites it as an equivalent set of queries to be executed on local data sources, and merges their results in a unified answer to be presented to the user.

Figure 3.4 shows an example of the formulation of the query Q: *"retrieve all the QTLs that affect the resistance of a plant to the fungus "Fusarium""*. This query lets the user find which QTLs, i.e. which pieces of DNA, influence the resistance of a plant to a particular fungus, "Fusarium" in this case, that can affect a plant with a disease and eventually cause its death. The result of the query, i.e. the QTLs that can express a high resistance to this fungus, allows the breeder to find a molecular marker that can help him to identify the presence of the QTL in the plant genome, and thus to decide whether to choose or not that germplasm for breeding.

To do this, the user selects the class *QTL* from the tree representing the GS on the left side. 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 Fig.3.5

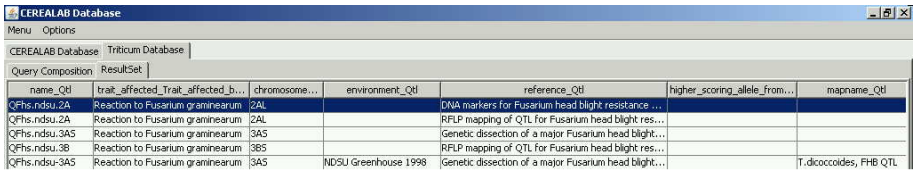


Figure 3.5: The Result Set obtained after query Q submission

As a practical example, a breeder interested in yield related traits can find in the Triticum schema the varieties characterized by a 1000 kernel weight of his interest. Once he obtains a list of varieties, the user can check which

of them have been genotyped for instance with molecular markers associated with QTLs/genes that determinate resistance to certain pathogens, in order to choose those that harbor the high scoring allele. To do this, the user has to browse the ontology *Phenotypic Data*, then in the *Yields* related trait subclass, select the class *a1000_Kernel_Weight* from the tree representing the GS. All the attributes of *a1000_Kernel_Weight* are shown in the Global Class Attributes tree. If the users adds to the selection the Referenced Class *Marker_Tested_on_germplasm* 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. The user can then choose to retrieve the phenotypic evaluation data from all or only one selected data source (GRIN and/or CRA in this case). Then if he is interested only in varieties with 1000 Kernel Weight values higher than 20 grams he just has to add in the Condition panel the condition "> 20". Clicking the button Execute Query, all the phenotypic data available for cultivars that meet the criteria are retrieved, and the molecular data retrieved comprise: QTL or gene, associated marker and genotyping data from Cerealab project. In this way the user can choose from the varieties with satisfying 1000 kernel weight those that have been already genotyped for molecular markers associated with resistance and harbor the higher scoring allele . Otherwise if the breeder is interested only in a given variety (for instance "Eureka" or "Bolero") he can restrict the query only to those two cultivars by adding in the Condition panel the condition: Germplasm like 'Eureka% and "add and condition" 'Bolero%'.

3.7 Discussion

In this chapter a real application of the MOMIS system was described. We created the Cerealab ontology providing both molecular and phenotypic data about wheat, barley and rice, integrating existing molecular and phenotypic data sources and data provided by the Cerealab project. This application perfectly describe the advantages provided by the MOMIS approach especially in the automatic discovery of the mappings between the GS and the local sources. The Cerealab ontology schema allows users to retrieve data coming from numerous data sources by providing a single easy interface instead of navigating through numerous web databases. The Query Composer available in MOMIS also represents a valuable tool for users of this kind of applications as they usually have low IT expertise and thus need a user-friendly interface.

The Cerealab ontology is also a very important contribution for breeders as it can improve the breeding process allowing cereal breeders to find the

right molecular markers to be used to intentionally breed certain traits, or combinations of traits, over others. To do this, access both to molecular data and phenotypic evaluation of traits is required. No resource was available so far that combined both these two kind of data and thus many data sources had to be accessed and the information obtained had to be combined manually. With our system both molecular and phenotypic data are available through a single graphical interface.

The Cerealab ontology will be improved within the SITEIA (Safety, Technologies and Innovation for the Agro-food sector) laboratory[7], funded by the Emilia Romagna region to continue the work started within the Cerealab project.

[7] http://www.siteia.it

Chapter 4

Representing and Querying Data and Multimedia Sources with MOMIS

Managing data and multimedia sources with a unique tool is a challenging issue. In this chapter, we present a methodology and a tool for building and querying an integrated virtual schema of data and multimedia sources. The tool is designed by coupling MOMIS with the MILOS multimedia content management system, developed by the Institute of Information Science and Technologies (ISTI) of the Italian National Research Council (CNR) of Pisa, Italy.

The work presented in this and the next chapter was conducted in the context of the NeP4B (Networked Peers for Business)[1] project, where we aim to contribute innovative Information and Communication Technology (ICT) solutions for Small and Medium Enterprises (SME), by developing an advanced technological infrastructure to enable companies of any nature, size and geographic location to search for partners, exchange data, negotiate and collaborate without limitations and constraints. The idea at the basis of the NeP4B project is to create a network of semantic peers, related to each other by mappings. Each semantic peer exposes a Semantic Peer Data Ontology (SPDO), that is a Global Schema representing all the knowledge available in the peer, in terms of data sources, multimedia sources, and service related to the same domain.

This chapter focuses on the representation of multimedia data sources in MOMIS and describes both the MOMIS query processing (a detailed description of the traditional query execution in the MOMIS Query Manager can be found in [Beneventano and Bergamaschi, 2007, Orsini, 2009]) and its extension for multimedia query processing. The work presented in this chapter has been published in [Beneventano et al., 2008a].

4.1 Motivations

Similarity search for content-based retrieval (where content can be any combination of text, image, audio/video, etc.) has gained importance in recent years, also because of the advantage of ranking the retrieved results according to their proximity to a query. The systems usually exploit data types such as sets, strings, vectors, or complex structures that can be exemplified by XML documents. Intuitively, the problem is to find similar objects with respect to a query object according to a domain specific distance measure. However, the growing need to deal with large, possibly distributed, archives requires specialized indexing support to speedup retrieval. The common assumption is that the costs to build and to maintain an index structure are lower compared to the ones that are needed to execute a query.

[1] `http://www.dbgroup.unimo.it/nep4b`

In this chapter an approach to extend the action sphere of traditional mediator systems allowing them to manage "traditional" and "multimedia" data sources at the same time is presented. The approach is implemented in a tool for integrating traditional and multimedia data sources in a virtual global schema. My contributions in this work were in the analysis of the characteristics of multimedia sources and the definition of their representation in MOMISİ also participated in the development of the methods for coupling MOMISwith MILOS.

We believe this approach is an interesting achievement for several reasons. First, the application domain: there are several use cases where joining traditional and multimedia data is relevant (see Section 4.2 for a concrete scenario). Second, multimedia and traditional data sources are usually represented with different models. While there is a rich literature for transforming the differently modeled traditional data sources into a common model and it is possible to represent different multimedia sources with a uniform standard model such as MPEG-7, a standard for representing traditional and multimedia data does not exist. Finally, different languages and different interfaces for querying "traditional" and "multimedia" data sources have been developed. The former relies on expressive languages allowing expressing selection clauses, the latter typically implements similarity search techniques for retrieving multimedia documents similar to the one provided by the user.

In this chapter I will describe the methodology based on the MOMIS system for building and querying a Semantic Peer Data Ontology (SPDO), i.e. a Global Schema including traditional and multimedia data sources related to a domain. The developed system couples MOMIS with the MILOS system [Amato et al., 2004], a Multimedia Content Management System tailored to support design and effective implementation of digital library applications. MILOS supports the storage and content based retrieval of any multimedia documents whose descriptions are provided by using arbitrary metadata models represented in XML.

This work extends the MOMIS integration capabilities in different directions: a language extension to capture the semantics of multimedia data types and a multimedia wrapper to automatically extract multimedia data source schemata have been designed and implemented. Moreover, problems and solutions to extend a mediation query processing to deal with multimedia data are discussed.

The chapter is organized as follows: Section 4.2 describes an applicative scenario and the example that will be referred to through the rest of the chapter. Section 4.3 presents the ODL_{I^3} representation of multimedia sources, while Section 4.4 proposes an overview of the system, and finally Section 4.5 introduces the problem of querying structured and multimedia

data in a mediator system.

4.2 An applicative scenario

Let us consider the tourist domain where we can find many portals and websites. The information about a tourist service, location or event is frequently widespread in different specific websites, due to the specialization of the information publisher. Thus, if a user wants to have a complete knowledge about a location, he or she has to navigate through several websites. This issue generates multiple queries with search engines, retrieving incomplete and overlapping information.

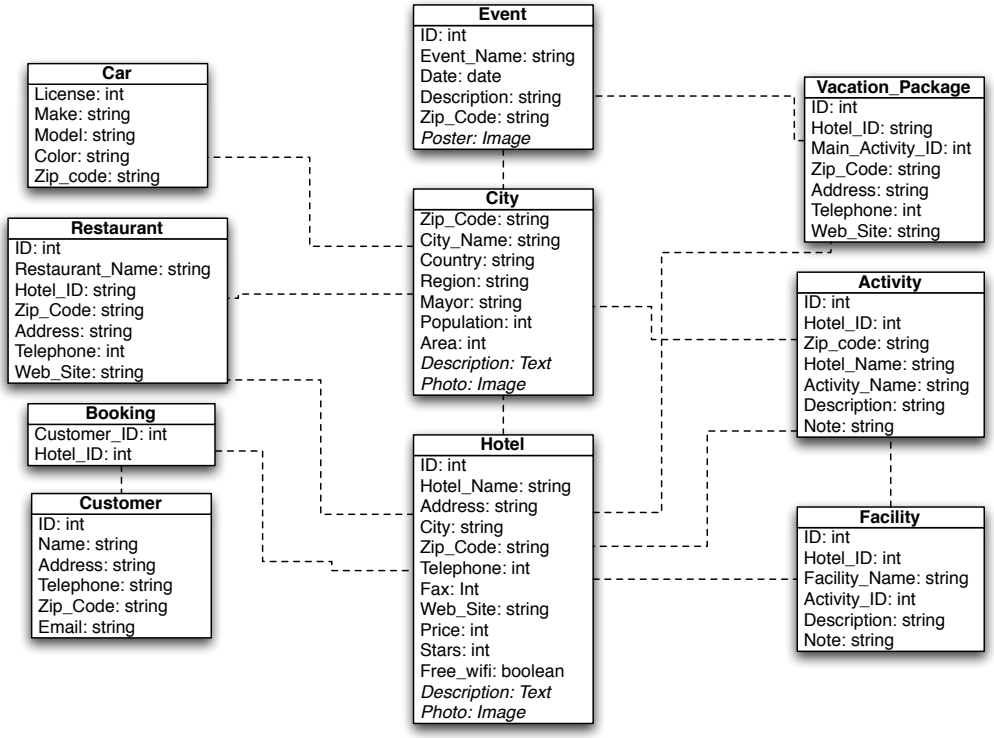


Figure 4.1: The Tourist Global Schema (Multimedia Attributes are shown in *italics*)

An application which provides a unified view of the data provided by different web sources, may guarantee the tourist promoters and travelers a more complete and easy to find information. Moreover, since our approach provides only a unified representation of data which still reside on specialized

websites, the possibility of having out-of-date information is the same as from navigating the single specialized websites one at a time.

Let us introduce as an example three information systems providing information about Italian locations which we want to integrate to create a large information source available for tourist purposes:

- `BookAtMe` provides information about more than 30.000 hotels in more than 8.000 destinations. For each hotel, information is provided about facilities, prices, policies, *multimedia contents* ...
- `Touring` provides information about Italian hotels, restaurants and cities. By means of three different forms, it is possible to find the available hotels, restaurants and main monuments for each city.
- `TicketOne` provides information about Italian cultural events. For each event, a description including place, price and details (e.g. including *multimedia contents*) is provided. The information system offers services to check the ticket availability of a particular event and, also, to buy tickets.

In Figure 4.1 we provide a graphical representation of the schema obtained from the integration of the three data sources. To give an idea of the global view obtained, *hotels* and *restaurants* are located in a certain *city*. A restaurant can be either the restaurant of a certain hotel or not. Every hotel has certain *facilities*, and some kind of *activities* can be performed as the facilities of that hotel. *Events* take place in a city, and the hotels offer *vacation packages* for specific events. Information about *customers* of the hotels and about *bookings* made by customers for certain hotels are stored. Moreover, *cars* are available for rental in certain cities. The special attribute type `Image` is introduced to represent Multimedia Objects, namely *Photo*, and *Poster* of the City, Hotel and Event classes. These Multimedia Objects are annotated with the MPEG-7 standard as described in the following.

Each data source contains traditional data types (e.g. city name in the Event class) together with multimedia data type (e.g. poster in the same Event class). Current search engines for multimedia content perform retrieval of similar objects using advanced similarity search techniques. The NeP4B project aims at combining the exact search on traditional data with a similarity search on multimedia data, exploiting the SPDO obtained by integrating the different data sources (traditional as well as multimedia). As an example, consider a user who wants to attend an event which involves fireworks. Using advanced search techniques over multimedia documents, in this case "poster", it will be possible to search for posters that contain "festival" and related images of fireworks.

The SPDO is a Global Schema is built applying the MOMIS methodology described in Section 2.4 to traditional data sources and multimedia sources, managed by MILOS.

Let us consider for example the ODL_{I^3} description of the global class *Hotel* of Figure 4.1. This class will be used as a reference in the rest of this chapter. The class is the result of the integration of the two local classes *resort* and *hotel*, having description:

```
interface resort() {
    //standards attributes
    attribute   string      Name;
    attribute   string      Telephone;
    attribute   string      Fax;
    attribute   string      Address;
    attribute   string      Zip;
    attribute   string      Web-site;
    attribute   integer     Room_num;
    attribute   integer     Price_avg;
    attribute   string      City;
    attribute   integer     Stars;

    //multimedia attributes
    attribute   Image       Photo;
    attribute   Text        Description;
}

interface hotel() {
//standard attributes
    attribute   string      denomination;
    attribute   string      tel;
    attribute   string      fax;
    attribute   string      address;
    attribute   string     zipcode;
    attribute   string      www;
    attribute   integer     rooms;
    attribute   integer     mean_price;
    attribute   string      location;
    attribute   boolean     free_wifi;

    //multimedia attributes
    attribute   Image       img;
    attribute   Text        commentary;
}
```

The mapping table of such Global Class takes into account the multimedia attributes `Commentary, Description` (Text) and `Photo, img` (Image) introduced by multimedia sources. In general, only "homogenous" mappings between multimedia attributes are permitted, e.g, `Text`-to-`Text`, `Image`-to-`Image`, etc.

Hotel	resort	hotel
Hotel_Name (join)	Name	denomination
Address	Address	address
City	City	location
Zip_Code	Zip	zipcode
Telephone	Telephone	tel
Fax	Fax	fax
Web_Site	Web-site	www
Price (RF)	Price_avg	mean_price
Stars	Stars	–
Free_wifi	–	free_wifi
Description	Description	commentary
Photo	Photo	img

According to the mappings, we specify the *name* as join attribute intending that instances of the classes *resort* and *hotel* having the same *Name* (*denomination*) represent the same real object. Moreover, a resolution function may be defined for the global attribute price, i.e. the value of the global attribute price is the maximum of the values assumed by both the local sources.

4.3 Representing multimedia data within the SPDO

Multimedia sources (MMS) are managed and queried by means of MILOS.

Each MMS is described through an ODL_{I^3} schema (although it is internally implemented in XML) called Local Multimedia Schema (LMS) that encapsulates specific predefined data-types for the management of multimedia data. In particular, in LMS there is a set n of attributes $a_1, \ldots, a_n$, declared using standard predefined ODL_{I^3} types (such as string, double, integer, etc.). These attributes are referred to as *standard attributes* and support set oriented operators typical of structured and semi-structured data, such as $=$, $<$, $>$, ... Along with the standard attributes, LMS includes a set of m attributes $s_1, \ldots, s_m$, declared by means of special predefined classes, which

support the similarity operator $\sim$. The query on these attributes returns a ranked list sorted by descending relevance. We refer to these attributes as *multimedia attributes*. For instance, we can have a `Text` class that encapsulates natural language descriptions searchable through the full-text search. An `Image` class that encapsulates visual descriptors encoded in MPEG-7 extracted from data objects (photos, videos, etc). The following example shows the declaration of the city class, which comprises five standard attributes and two multimedia attributes.

```
interface City() {
    // standard attributes
    attribute   string      City_Name;
    attribute   string      Zip_Code;
    attribute   string      Country;
    attribute   string      Region;
    attribute   string      Mayor;
    attribute   integer     Population;
attribute   integer     Area;

    // multimedia attributes
    attribute   Image       Photo;
    attribute   Text        Description;
}
```

Once a MMS is described with the ODL_{I^3} language, the MOMIS approach can be applied without further distinction between a "traditional" data source and a MMS to build the SPDO.

4.4 The system at a glance

Our approach exploits and extends the MOMIS and MILOS systems, where MOMIS is exploited for building and querying the SPDO, and MILOS for managing the interaction with the multimedia sources.

As depicted in Figure 4.2, MOMIS relies on wrappers for interacting with the sources. A wrapper has to accomplish two tasks:

1. translating the descriptions of the sources into a common language (ODL_{I^3}) representation. The translation from the usual modelling languages (relational, object, XML, OWL) and ODL_{I^3} is quite straightforward. In the case of multimedia sources, special functionalities, provided by MILOS, are required for representing the multimedia data types.

2. executing query at the local source level. The MOMIS Query Manager is able to translate a query on the SPDO (a *global query*) into a set of local queries to be locally executed by means of wrappers. As shown in Section 4.5.1, MILOS will manage the execution of queries on the multimedia sources.

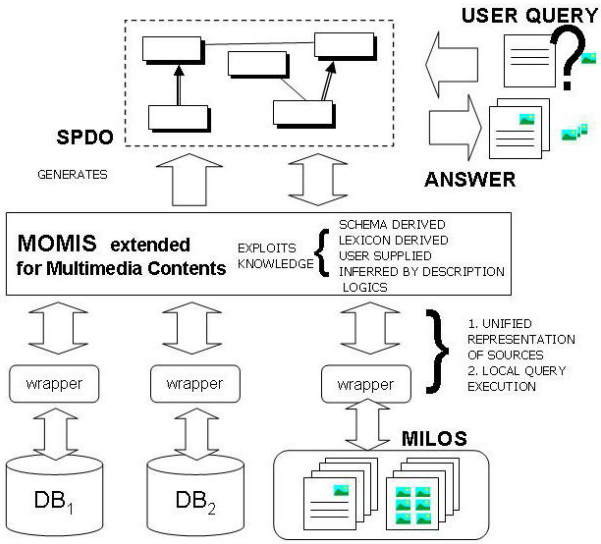


Figure 4.2: The functional architecture

The integration of the MOMIS and the MILOS systems is achieved by means of the Web Services technology. The Remote Method Invocation (RMI) of MILOS allows MOMIS to search for multimedia contents by invoking the method:

```
search(String LMSClassName, String[] values,
       String[] fields, String[] operators, String returnField)
```

where the `fields` parameter is an array of (application known) names of multimedia attributes of the LMS (Local Multimedia Sources ClassName) to search for. The `values` parameter specifies the values that the fields must match (the different fields are searched by using the connective *AND*). The `operators` parameter specifies the matching operator to be used (i.e.: the standard XQuery operators "=", ">", "<", ">=", "<=", "! =" and the similarity operator "∼" for similarity search on multimedia attributes as shown in next section). Finally, `returnField` specifies the field of the retrieved records that the application wants to know. When a user submits a query on the global schema, MOMIS takes the query and produces a set of translated sub-queries to be sent to each involved data source. If the local source is a

multimedia one, MOMIS translates the local query into an invocation of the *search* method following the technique described in Section 4.5.2.

4.5 Querying structured and multimedia data

According to the functional architecture depicted in figure 4.2, the proposed approach relies on the engines of the local sources for the execution of the queries. Thus, the query processing has to manage the heterogeneity that such architecture entails, in particular the different operators supported by the different local sources. We consider a scenario where:

1. *Traditional data sources* support queries with set oriented operators typical of structured and semi-structured data, such as $=$, $<$, $>$, ... More precisely the condition of such kinds of query, denoted by <data_cond>, is a conjunction of positive atomic constraints (*attribute op value*) or ($attribute_1$ *op* $attribute_2$) with *op* a relational operator. Our proposal exploits generic DBMSs for managing data sources.

2. *Multimedia data sources* support, besides a <data_cond> on "standard" attribute, a <multimedia_cond> which express full-text search on textual attributes (declared as `Text` type), similarity search on MPEG-7 attributes (declared as `Image` type). We consider multimedia sources managed by the MILOS system, as explored in the next Section.

4.5.1 Processing Queries on Multimedia Sources

Let us consider the multimedia class *city* described in section 4.2, and suppose that a user would like to find all the cities that have zip code equal to 41100, image similar to the one given as example (by providing its URL), and that also best match the given textual description. The following query expresses, in an SQL-like query language, the user request:

```
select City_Name from City
where Zip_Code = '41100' and Photo ~ "http://turismo.comune.modena.it/38.jpg"
      and Description ~ "cathedral"
```

The MILOS system is responsible for the query execution. A particular attention has been paid to the implementation of the $\sim$ operator, in case of

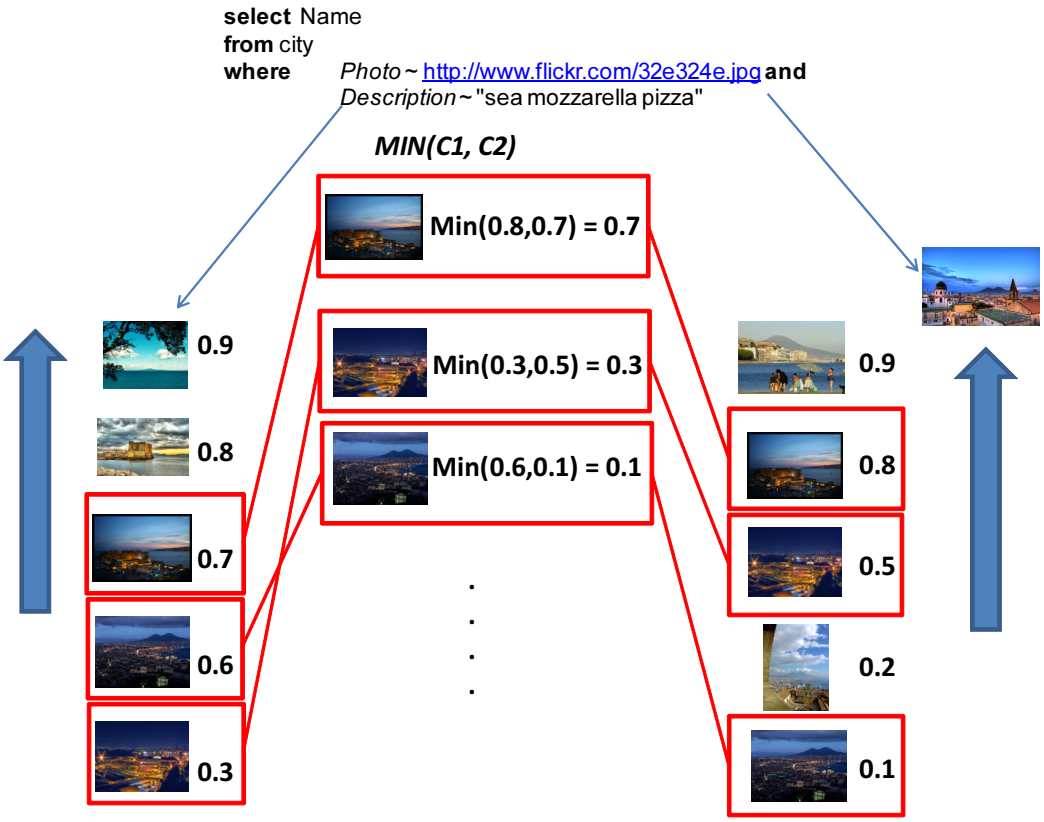


Figure 4.3: Example of multimedia query processing.

nearest neighbors queries (as it is in our case). This operator, in fact, introduces the problem of complex query search, when more similarity conditions are combined using the AND operator. To determine the top k cities, that is, k objects with the highest overall scores, the naive algorithm must access every object in the database, to find its score under each attribute.

In Figure 4.3 we show an example of query on multimedia sources, where the *MIN* operator is used as fuzzy AND to combine the scores (ranging from 0 to 1) of two streams of result sets retrieved by the full-text part of the query and the visual similarity part. In [Fagin, 1998], Fagin addressed this problem for a user query involving several multimedia attributes. Notice that MILOS generates a unique combined score even if more than one similarity condition is expressed.

4.5.2 Querying the SPDO

Given a global class G with either multimedia attributes, denoted by <multimedia_attr> (such as `photo` and `description` in the class `Hotel`) and "standard" data type attributes, denoted by <data_attr> (such as `name` and `city` in the class `Hotel`), a query on G (*global query*) is a conjunctive query, expressed in a simple abstract SQL-like syntax as:

$$\begin{aligned} Q \;=\; & SELECT < data\_attr >, < multimedia\_attr > \\ & FROM\ G \\ & WHERE\ \ < data\_cond > \\ & AND\ \ \ < multimedia\_cond > \end{aligned}$$

where <data_cond> is on standard data type attributes of G and <multimedia_cond> is expressed, with the similarity operator $\sim$, on multimedia attributes of G.

To answer a global query on G, the query must be rewritten as an equivalent set of queries (*local queries*) expressed on the local classes $L(G)$ belonging to G. This query rewriting is performed by considering the mapping between the SPDO and the local schemata; in a GAV approach the query rewriting is performed by means of query unfolding, i.e., by expanding the global query on G according to the definition of its mapping query q_G.

In the case of "standard" local data sources, the local execution of a query produces a list of objects matching <data_cond>. In the case of local multimedia data repositories, the $\sim$ operator is used to ideally re-order all the objects on the basis of their similarity expressed by the <multimedia_cond>. The final result will be a ranked list of the objects of each multimedia source. At the global level, the system needs to combine all this local results. The results are fused by MOMIS using a Full Outerjoin-merge (defined in Section 2.5) operation between the lists. Multimedia result lists are merged by means of both ranks and full join conditions in order to obtain a unique multimedia ranked list of results. The join conditions are defined by means of a join attribute, which is a "standard" attribute present in all local classes. This list has to be further fused with the results coming from "standard" sources.

In this section we give an intuitive description of this process, describing the query unfolding process for <data_cond> and showing how to extend this process to <multimedia_cond>.

Query unfolding for <data_cond>

Let us consider the following global query:

```
select Hotel_Name, Web_Site, Price from Hotel
where Price < 100 and Stars = 3 and Free_wifi= TRUE
```

The process for executing the global query consists of the following steps:

1. **Computation of Local Query conditions:** constraints on the global query are rewritten into corresponding constraints supported by the local classes. In particular, an atomic constraint (GA *op* `value`) is translated into a constraint on the local class L, considering the mappings defined in the Mapping Table, as follows:

   ($MTF[GA][L]$ *op* `value`)
   if GA is an homogeneous attribute **and**
   $MTF[GA][L]$ is not null **and**
   the *op* operator is supported into L
   true otherwise

   where $MTF[GA][L]$ is the data conversion function defined for the local attribute L with respect to the global attribute GA. For example, the constraint `stars = 3` is translated into a constraint `Stars = 3` considering the local class `resort` and is not translated into any constraint considering the local class `hotel`.

2. **Computation of Residual Conditions:** Conditions on not homogeneous attributes cannot be translated into local conditions: they are considered as *residual* and have to be solved at the global level. As an example, let us suppose that a numerical global attribute (as for instance `price` in our example) GA is mapped onto $L1$ and $L2$, and an AVG (average) function is defined as resolution function, the constraint (GA = value) cannot be pushed at the local sources, since the MAX function has to be calculated at a global level and the constraint may be globally true but locally false.

3. **Generation and execution of local queries:** for each local source involved in the global query, a local query is generated. The select list is obtained with the union of the attributes of the global select list, of the Join Condition and of the Residual Condition; these attributes are transformed into the corresponding ones at the local level on the basis of the Mapping Table.

   ```
   LQ_resort = select Name, Price_avg, Web-site from resort
               where Stars = 3
   ```

```
LQ_hotel  = select denomination, mean_price, www from hotel
            where free_wifi= TRUE
```

These queries are executed on the local sources.

4. **Fusion of local answers** : The local answers are fused into the global answer on the basis of the mapping query q_G defined for G, i.e. by using a Full Outerjoin-merge operation. Intuitively, this operation is substantially performed in two steps:

   (a) Computation of the **full join** of local answers (`FOJ`):

   ```
   LQ_resort full join LQ_hotel on
    (LQ_resort.Name=LQ_hotel.Denomination)
   ```

   (b) **Application of the Resolution Functions :** for each attribute GA of the global query the related Resolution Function is applied to `FOJ` thus obtaining a relation R_FOJ ; in our example the result is the relation `R_FOJ(Hotel_Name, Web_Site, Price)`. Notice that if the `Web_Site` attribute does not have any Resolution Function applied, i.e. we hold all the values, all the `Web_Site` values for a certain hotel X are retained: we will have only one record for X in `R_FOJ`, where the value of the `Web_Site` attribute is the concatenation of the `Web_Site` values (obviously adequately separated, e.g. by a comma).

5. **Application of the Residual Condition:** the result of the global query is obtained by applying the residual condition to the `R_FOJ`:

   ```
   select Hotel_Name, Web_Site, Price from R_FOJ where Price < 100
   ```

Query unfolding for <multimedia_cond>

Let us start our discussion about <*multimedia_cond*> with the simplest case where multimedia attributes are mapped on a single local source: for the global class `Hotel` we consider $M[resort][Photo] = NULL$ and $M[hotel][description] = NULL$. Let us consider a query with a <*multimedia_cond*>:

```
select Hotel_Name, Photo from Hotel
where Stars = 3
and Free_wifi = TRUE and Photo ~ "http://www.hotels.com/imgs/x123.jpg"
and Description ~ "private beach"
```

By applying the unfolding process described before, there is no residual condition since all the involved attributes are homogeneous; the local queries are:

```
LQ_resort  = select Name, Description from resort
             where Stars= 3 and
             and Description ~ "private beach"

LQ_hotel  = select denomination, img from hotel
            where free_wifi= TRUE
            img ~ "http://www.hotels.com/imgs/x123.jpg"
```

Each local query produces a stream of tuples in order of decreasing similarity w.r.t. the given query. In the fusion step, tuples of *LQ_resort* and *LQ_hotel* are joined on the condition `(LQ_resort.Name=LQ_hotel.Denomination)` and the resulting `FOJ` provides the global query answer since it is necessary to apply neither resolution functions nor residual conditions.

Obviously, in general, there is no need to compute the whole combined result list. Rather, the query evaluation typically retrieve a ranked result set, where an aggregated score (from *LQ_resort* and *LQ_hotel*) is attached to each tuple returned. In fact, only a few top-ranked results are normally of interest to the user. This is exactly the same complex query problem that arises in processing queries on local multimedia sources discussed in Section 4.5.1. However, although the same solution applies also in this case, we adopt a more general approach, applicable to any situation, as shown in the following.

Let us consider now the general case where a multimedia attribute is mapped on more than one local source. As discussed before, (1) we need to define a resolution function for such attribute and (2) constraints on such attribute cannot be fully solved on local sources and need to be solved at global level.

To describe this problem, let us suppose that the global attribute `photo` is mapped on both the local classes `resort` and `hotel` and let us consider a query with a <multimedia_cond> on such a global attribute `photo`:

```
select Hotel_Name, Web_Site, Photo, Description from Hotel
where Price < 100 and Stars = 3 and Free_wifi= TRUE and
      Photo ~ "http://www.hotels.com/imgs/x123.jpg"
```

If a resolution function is defined for the attribute `photo`, according to the query unfolding process described in the previous section, the constraint

for the attribute `photo` needs to be solved at the global level and then it will not be rewritten at the local level.

In the case of multimedia attributes, specific resolution functions may be considered. For this reason, we defined a new resolution function, called *MOST_SIMILAR*, which returns the multimedia objects most similar to the one expressed in the <multimedia_cond>.

In the following we discuss the query unfolding process in the presence of this resolution function; note that only the query unfolding steps concerning the constraint on `photo` are described.

In step 3 the constraint on `photo` is rewritten at the local level as:

```
LQ_resort = select Name, Web-site, Photo, Description from resort
     where Stars = 3 and Photo ~ "http://www.hotels.com/imgs/x123.jpg"

LQ_hotel = select denomination, www, img, commentary from hotel
  where free_wifi= TRUE and img ~ "http://www.hotels.com/imgs/x123.jpg"
```

Intuitively, each local query produces a result list where each "local record" has a score and is ordered according to the similarity of the local `Image` attributes with the given sample image.

In step 4, after the `FOJ` computation, the *MOST_SIMILAR* function must be "applied": for records with a not null "multimedia object" both for `img` and `Photo`, we need to identify the "multimedia object" most similar to the one expressed in the <multimedia_cond>. Since the local query results are *congruent*, i.e. are related to the same <multimedia_cond>, the *MOST_SIMILAR* function exploits the associated scores for selecting the multimedia results.

In the case of *not congruent* results, as in the following example, another technique has to be applied. Suppose that *hotel* contains only the multimedia attribute `img`, and *resort* contains both the multimedia attribute `Photo` and `Description`. Consider now a query with a <multimedia_cond> on both such global attributes:

```
select Hote_Name, Web_Site, Photo, Description from Hotel
where Price < 100 and Stars = 3 and Free_wifi= TRUE and
      Photo ~ "http://www.hotels.com/imgs/x123.jpg"
      and Description ~ "private beach"
```

In step 3 multimedia constraints are rewritten at the local level as:

```
LQ_resort =  select Name, Web-site, Photo, Description from resort
             where stars= 3 and Photo ~ "http://www.hotels.com/imgs/x123.jpg"
             and Description ~ "private beach"
```

```
LQ_hotel  =  select denomination, www , img from hotel
             where free_wifi= TRUE and
             img ~ "http://www.hotels.com/imgs/x123.jpg"
```

The local execution of both queries generates a list of results, each one with a score. However, the scores are "not congruent", since the one coming from `LQ_resort` is the product of a combination of two scores (with the fuzzy **AND** for instance), while the list coming from `LQ_hotel` has scores that correspond only to the similarity on the `img` attributes.

Taking the objects with greatest score will favor the *hotel* records.

In this scenario neither the scores are useful to define the concept of similarity, nor the position of each object in the local result sets, i.e. its rank in each answer to the local queries. Ranks could be used to define a new "score" of similarity. Thus, a way to define a *MOST_SIMILAR* aggregation function is to use the ordinal ranks of each object in the returned lists.

As suggested in [Fagin et al., 2004], a simple yet effective aggregation function for ordinal ranks is the *median* function. This function takes as the aggregated score of an object its median position in all the returned lists. Thus, given n multimedia sources that answer to a query by returning n different ranked lists $r_1, \ldots, r_n$, the aggregated score of an object o will be $median(r_1(o), \ldots, r_n(o))$. The aggregated list will reflect the scores computed using the *median* function. The *median* function is demonstrated [Fagin et al., 2004] to be near-optimal, even for *top-k* or partial lists. Then, it is possible to produce an aggregated list even if the different sources return only $top - k$ lists as their local answers to the query. For all these reasons, we take the median as the *MOST_SIMILAR* aggregation function for multimedia objects at the global level.

4.6 Discussion

In this chapter I presented the coupling of the MOMIS integration system with the MILOS multimedia content management system. The goal was to implement a framework that allows a user to create and query an integrated view of traditional data and multimedia data sources. While building the GS only requires representing multimedia objects in the source schemas, query processing presents more interesting problems: I discussed some cases where multimedia constraints can be expressed in a global query without requiring multimedia processing capabilities at the global level. This is an interesting result, since it upsets the usual paradigm on which mediator systems are based, stating that the *query processing power* of a mediator is greater than the one of the integrated data sources [Chang and Garcia-Molina, 1999].

Chapter 5

Aggregated search of data and services

This chapter presents a semantic approach to perform aggregated search of data and services, that was carried on in the context of the NeP4B project in collaboration with the "Department of Informatics, Systems and Communication" (DISCO) of the University of Milano Bicocca and the Cefriel Research Center of Milan. We designed an extension for the MOMIS methodology to build a Global Schema that is the representation of all the knowledge available in a peer, both in the form of data and services. In particular, we developed a technique that, on the basis of an ontological representation of data and services related to the same domain, supports the translation of a data query into a service discovery process.

For evaluating our approach, we developed a prototype that couples the MOMIS system with a new information retrieval-based web service engine, the eXtended Information Retrieval Engine (XIRE). The results of our experiments shows the effectiveness of our proposal with respect to a collection of queries from the OWL-S Service Retrieval Test Collection 2.2[1].

Specifically, I participated in the definition of the methodology and implemented the MOMIS extensions for building the Global Schema and translating a data query into a keyword based query for the interaction with XIRE. I also run the tests on the MOMIS side to evaluate our approach. The work presented in this chapter was published in [Palmonari et al., 2007, Guerra et al., 2009, Beneventano et al., 2009].

5.1 Motivations

In the knowledge society, users need to access data and invoke services. Data search or service invocation can be individually realized, and both these operations provide value to users in the context of complex interactions. Data integration has been deeply investigated in the last decades, while the research on services is definitely more recent and was boosted by the development of Internet. As the set of available Web services grows, it becomes increasingly important to have automated tools to help identify services that match the requirements of a service requester. Service Oriented Architectures (SOA) are now widespread and nowadays application are made up of different services, each accomplishing a defined sub-task, and their composition satisfies the user requirement. SOA thus guarantees for modularity and software reuse. Large enterprises have now thousands of different services available in-house, representing an important asset that need to be valued. Gathering a thorough knowledge of the existing services allows exploiting

[1] `http://semwebcentral.org/projects/owls-tc/`

these services instead of re-writing something that is maybe already available but not known. Obviously such a knowledge eases the service outsource too.

Finding suitable services thus depends on the facilities available for service providers to describe the capabilities of their services and for service requesters to describe their requirements in an unambiguous and ideally machine-interpretable form. Adding semantics for service requirements and capabilities representation is essential for achieving this unambiguity and machine-interpretability. The process of abstracting goals from user desires is called goal discovery [Keller et al., 2004], i.e., the user or the discovery engine has to find a goal that describes (with different levels of accuracy) his or her requirements and desires. In the current literature on service and Web service discovery this step is mostly neglected.

The research on data integration and semantic service discovering has involved from the beginning different (not always overlapping) communities. As a consequence, data and services are described with different models, and different techniques to retrieve data and services have been developed. Nevertheless, from a user perspective, the border between data and services is often not so definite, since data and services provide a complementary view of the available resources: data provide detailed information about specific needs, while services execute processes involving data and returning an informative result.

Users need new techniques to manage data and services in a unified manner: both the richness of the information available on the Web and the difficulties the user faces in gathering such information (as a service result, as a query on a form, as a query on a data source containing information extracted from web sites) make a tool for search at the same time data and related services, with the same language, really necessary. As described in Section 1.3.2, the need to perform aggregated search is a young research field and very few approaches have been proposed.

Service as Data In this thesis I propose a new interpretation of aggregated search, i.e. the integration of distinct search modalities, on distinct sources and by distinct search spaces, obtained by a single query and producing aggregated results. The approach presented in this chapter can be described as a *Service as Data* approach, as opposed to *Data as a Service* approaches. In the Service as Data approach, informative services are considered as a kind of data source to be integrated with other data sources, to enhance the knowledge available in the Global Schema. To the best of our knowledge this approach is completely new in the literature. In particular we designed

and developed a system architecture able to enhance a SPDO representing the common knowledge extracted from heterogeneous sources, presented in Chapter 4, with mappings to the services available in the peer that are possibly related to the terms of the SPDO, and an information retrieval-based web service engine, XIRE, able to provide a list of ranked services according to a set of weighted keywords. Starting from an SQL-like query expressed in the terminology of the SPDO, and thanks to the extraction algorithm proposed, our systems is able to retrieve all data and services that are relevant to the query.

Reference example Referring to the same running example described in Section 4.2, let us consider a user who wants to find the name of hotels available in Modena. A query over the SPDO presented in the last chapter, would be:

```
select Hotel.*
from   Hotel
where  Hotel.City = 'Modena'
```

While the problem of finding relevant data for such query is well defined, an important problem is to retrieve, among the many services available, the ones that are possibly related to this query, according to the semantics of the terms involved in the query.

Such aggregated search is achieved by: (i) building the SPDO, (ii) building a Global Light Service Ontology (GLSO) consisting of the lightweight version of the ontologies used in the service semantic descriptions, (iii) defining a set of mappings between the SPDO and the GLSO, (iii) exploiting, at query time, term rewriting techniques based on these mappings to build a keyword-based query for service retrieval expressed in the GLSO terminology starting from a SQL query on the data sources.

Preliminary evaluations based on state-of-the-art benchmarks for semantic web services discovery show that our information retrieval-based approach provide promising results.

The outline of the chapter is the following: Section 5.2 introduces our approach for building a unified view of data and services. In paticular, Section 5.3 analyzes the models used for representing services and their translation into the ODL_{I^3} language to be managed by MOMIS. Section 5.4 and Section 5.5 describe how services are represented in our approach and mapped to a Global Schema of data sources, while Section 5.6 provides the description of the technique for retrieving data and related services. The system architecture is presented in Section 5.7 and finally experimental results of our approach are evaluated in Section 5.8.

5.2 Building the Global Data and Service View at Set-up Time

In the NeP4B approach, data sources and services are grouped into semantic peers. Each semantic peer generates a Peer Virtual View (PVV), i.e. a unified representation of the data and the services held by the sources belonging to the peer. A PVV is made up of the following components, shown in Figure 5.2:

- a **Semantic Peer Data Ontology (SPDO)** of the data, i.e. a common representation of all the data sources belonging to the peer; the SPDO is built by means of MOMIS as described in Section 2.4.
- a **Global Light Service Ontology (GLSO)** that provides, by means of a set of concepts and attributes, a global view of all the concepts and attributes used for the descriptions of the Web services available in the peer;
- a set of **mappings** which connect GLSO elements to SPDO elements.

5.3 Describing Services

As we said in Section 1.3.3, different ontology based standards for modeling the capabilities of web services have been recently suggested. The two main models proposed are the Web Service Modeling Ontology [Roman et al., 2004] and the OWL-S Ontology [Burstein et al., 2004]. Both these approaches use ontologies for modelling service descriptions. Our approach exploits these ontologies to enrich a SPDO with mappings to services relevant to the concepts of the SPDO itself. The approach is based on the MOMIS system, thus making it necessary to translate the ontologies describing the services into the ODL_{I^3} language. OWL-S semantic descriptions are ontologies expressed with the Web Ontology Language OWL[2] and refer to OWL domain service ontologies (SOs). MOMIS already provides an OWL wrapper able to translate an OWL ontology into the ODL_{I^3} language and vice versa [Orsini, 2004]. WSMO, instead, required to be analyzed to design its translation into ODL_{I^3}. This analysys is presented in this section after a brief description of WSMO.

[2] `http://www.w3.org/TR/owl-ref/`

WSMO WSMO is made up of four top level elements, namely Ontologies, Web services, Goals and Mediators. WSMO Ontologies form the data model upon which the other elements are built and describe all the concepts that the modeller of a given semantic description will use, along with the relationships between these concepts.

A Web service description in WSMO consists of five sub-components: *non-functional properties*, *imported ontologies*, *used mediators*, a *capability* and *interfaces*. The function of the service is described in terms of a capability, which states: what must be true about the input to the service (*precondition*); the state of the world (*assumption*) in order for the service to be invoked; what will be true about the output of service (*postcondition*) and the state of the world (*effect*) after the service has been executed. An interface of the WSMO Web Service contains a choreography description to describe how to invoke the service and an orchestration description to state which other services are used by this service in order to provide its functionality.

A *WSMO Goal* describes the end users perspective in terms of the function that the user desires and the way in which they wish to invoke the service. A goal in WSMO is described by *non-functional properties*, *imported ontologies*, *used mediators*, requested *capability* and requested *interface*.

The final top entity of WSMO is WSMO Mediators. A WSMO Mediator provides mechanisms to resolve interoperability issues between the other top entities of WSMO. An ontology to ontology mediator (ooMediator) can be used to resolve mismatches between two different ontologies that describe the same domain. A Web Service to Web Service mediator (wwMediator) can be used to state that a given WSMO Web Service has some relationship with another Web Service, i.e. it provides a subset of its functionalities or their equivalents. Web Service to Goal Mediators (wgMediator) allow the user to state that a given Goal can be fulfilled by the specified Web Service, while Goal to Goal Mediators (ggMediator) can be used to state the relationship between two WSMO Goals, i.e. equivalence or that one goal is a sub goal of the other.

The WSMO conceptual model can be formalized in terms of the Web Service Modeling Language (WSML) [de Bruijn et al., 2005], which provides a family of logical languages for expressing WSMO Ontologies, Web services, Goals and Mediators with different levels of expressivity and reasoning complexity. In particular, WSML consists of a number of variants based on different logical formalisms. The WSML variants more used for SWS are the Logic Programming based languages WSML-Core, WSML-Flight and WSML-Rule. In particular WSML-Flight is popular because it offers a good balance between expressiveness and a nice computational behaviour. WSML-

Flight (and its extension WSML-Rule) is a language based on F-Logic.

F-logic provides a logical foundation for frame-based and object-oriented languages for data and knowledge representation [Kifer et al., 1995]. The F-logic syntax is based on Logic Programming for rule definition and reasoning, but atoms of rules are defined by object constructors that explicitly provide an ontological approach for the representation of objects. The main F-logic ontological elements are instances, concepts, attributes and relations. Membership to classes and is-a relations are represented with special built in predicate symbols.

WSML-Flight axioms have the general form:

$$head :- body$$

where *head* is an atomic formula and body is a conjunction of negated (NAF) atomic formulas. Atomic formulas can be molecules, identity or relation expressions. Molecules combine atoms to provide shortcut specifications about a same object. As an example, the atomic formula below is a molecule that specifies that Miramare is a three-stars Hotel that offers wi-fi connection and cable TV as facilities:

$$\texttt{miramare} : \texttt{hotel}\,[\texttt{category->}3, \texttt{facilities->>}\,\{\texttt{wiFi}, \texttt{cableTv}\}]$$

The following is an example of rule stating that hotel of category equal or higher than 3 accept credit card as payment method:

```
x[acceptPaymentMethod ->> creditCard]:-
                    x[category -> y], y >= 3.
```

From WSMO to ODL$_{I^3}$ Being F-logic the logical basis for WSMO service semantic representation and ODL$_{I^3}$ the language based on Description Logics exploited for data annotation and integration, the first step for our aim is the translation between the two languages. Since both languages are based on a object-oriented perspective, the two languages are quite similar with respect to the ontological description approach. However, there are very strong differences since ODL$_{I^3}$ does not allow to express rules in the sense of LP rules; moreover F-logic is possibly non monotonic, allowing Negation As Failure (NAF) in the rule bodies.

Much of the semantic of a F-logic ontology is encoded in the rules, and therefore the impossibility of translating such axioms (we will refer as axioms to rules, according the WSMO terminology) into ODL$_{I^3}$ is very restrictive with respect to the desirable semantic translation from the ontologies exploited for service descriptions and the data ontologies. However, it is

possible to selectively extract from a F-logic ontology at least a piece of information that can be encoded into ODL$_{I^3}$. In particular it is possible to extract the following ontological aspects which can be encoded into ODL$_{I^3}$:

- concepts (mapped into ODL$_{I^3}$ "interface")
- attributes (mapped into ODL$_{I^3}$ "attribute")
- instances (mapped into ODL$_{I^3}$ "instance")
- relation (mapped into ODL$_{I^3}$ "relationship" or into ODL$_{I^3}$ interface via reification)
- the is-A hierarchy

An example of translation is given in Figure 5.1. Given a F-Logic ontology O is therefore possible to extract a light-weight ODL$_{I^3}$ ontology O' which holds some of the core aspects of the semantic content included in O.

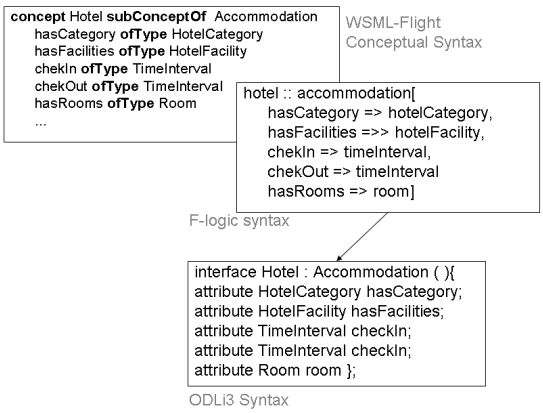


Figure 5.1: Example of ontological translation from an F-logic representation to a ODL$_{I^3}$ representation; the top-right rectangle represents the WSML-Flight conceptual syntax

5.4 GLSO Construction and Semantic Similarity Matrix

The analysis presented in the previous section proves that both WSMO and OWL-S models can be used in our approach. We thus decided to refer to OWL-S as the reference semantic description language, to have the possibility

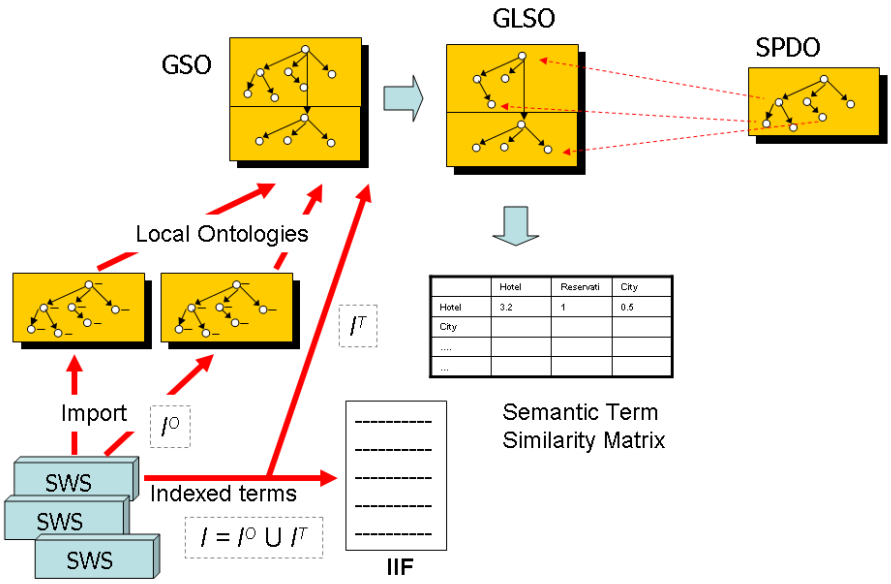


Figure 5.2: A sketch of the overall process for building the GLSO

to exploit the large set of service descriptions given in the OWL-S Service Retrieval Test Collection 2.2. (OWLS-TC)[3] to evaluate our approach. For this reason, in the rest of this chapter we will always refer to OWL-S for semantic web services.

The global light service ontology is built by means of a process consisting of three main steps: (i) service indexing, (ii) Global Service Ontology (GSO) construction, (iii) Global Light Service Ontology (GLSO) construction and Semantic Similarity Matrix (SSM) definition. A sketch of the overall process is given in Figure 5.2. Each of these steps is now briefly described, for a detailed description see [Beneventano et al., 2009, Guerra et al., 2009] and [Palmonari et al., 2010].

Service Indexing

In order to inquiry services, an Information Retrieval approach is applied to the semantic descriptions of Web services. Referring to the OWLS-TC, we stress that: 1) several services may be collected from the Web or from available repositories (OWLS-TC provides more than 1000 for the OWL-S 1.1 language), 2) several ontologies are referred to in the service descriptions (OWLS-TC provides 43 ontologies), and 3) these ontologies may concern dif-

[3] http://projects.semwebcentral.org/projects/owls-tc/

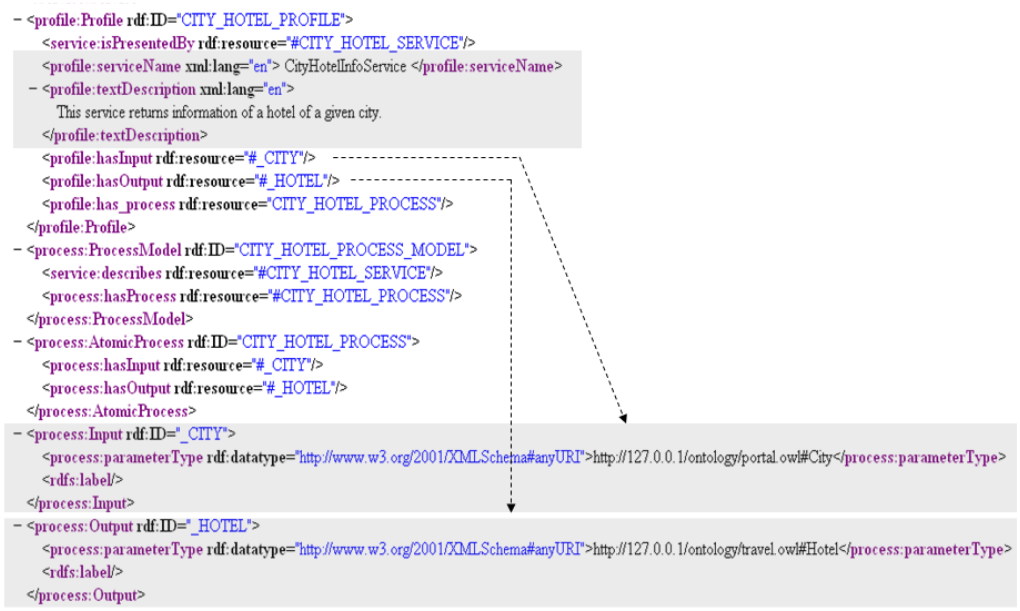


Figure 5.3: An example of service profile

ferent domains (OWLS-TC provides services from seven different domains).

The IR approach, aimed at locating relevant services related to a data query, requires a formal representation of the service descriptions stored in the repository, and it is based on full text indexing which extracts terms from six specific sections of a service description: *service name, service description, input, output, pre-condition and post-condition*[4].

As an example, the service "City_Hotel_Service" from the OLWS-TC collection imports two domain ontologies (Travel and Portal) and is described by: the name "CityHotelInfoService", the text descriptions "This service returns information of a hotel of a given city", and the ontology concepts *City* (from the Portal ontology) and *Hotel* (from the Travel ontology) as input and output respectively. While the service name and description consist of short text sections, input and output refer to domain SOs, namely, a portal and a travel ontology (pre-condition and post-condition are represented analogously but miss in the example). According to this process a set of index terms I that will be part of the dictionary is extracted from a Web Service descriptions. We call I^O, where $I^O \subseteq I$, the set of index terms taken from ontology resources (e.g. *City* and *Hotel* in the example); we call I^T, where

[4]More precisely, ontological references for inputs and outputs are defined by *process:Input* and *process:Output*; moreover, URIs are not completely specified for sake of clarity

$I^T \subseteq I$, the set of index terms extracted from textual descriptions (e.g. "information", "hotel", "city"). When the sets I^O and I^T are not disjoint, i.e. $I^O \cap I^T \neq \emptyset$, each term belonging to the intersection is reported just once in the Dictionary (in the example the terms "hotel" and "city" extracted from both the ontology and the service descriptions appears just once in the set $I = \{"information", Hotel, City\}$.

The indexing structure is based on a "structured document" approach, where the structure of the document consists of the six aforementioned sections. The inverted file structure consists of (i) a *dictionary* file based on I, and (ii) a *posting file*, with a list of references (for each term in the dictionary) to the services' sections where the considered term occurs. The posting file is organized to store, for each index term $i \in I$, a list of blocks containing (i) the service identifier, and (ii) the identifiers of the service's section in which the term appears.

Each block has a variable length, as a term appears in at least one section of a service in the posting list, but it may appear in more than one section. In the usual IR approach to text indexing an index term weight is computed, which quantifies the informative role of the term in the considered document. We propose an approach to sections weighting, in order to enhance the informative role of term occurrences in the distinct sections of the service. These importance weights will be used in the query evaluation phase in order to rank the services retrieved as a response to a user's query as explained in Section 5.6.

GSO construction

The Global Service Ontology (GSO) is built by (i) loosely merging each service ontology O such that $i \in O$ for $i \in I^O$ (e.g. the portal and travel ontologies in the reference example),taking into account ontology imports recursively, and (ii) associating a concept C_i with each $i \in I^T$, by introducing a class $Terms$ subclass of $Thing$ in the GSO and stating that for every $i \in I^T$, C_i is subclass of $Terms$ (e.g. the term "reservation" occurring in the *service name* or *service description* sections). With "loosely merging" we mean that SOs are merged asserting that their top concepts are all subclasses of *Thing* without attempting to integrate similar concepts across the different integrated ontologies. Therefore, if the source SOs are consistent, the GSO can be assumed to be consistent because no axioms establishing relationships among the concepts of the source SOs are introduced (e.g. axioms referring to the same intended concept would actually refer to two distinct concepts with two different URI). Loose merging is clearly not the optimal choice with respect to ontology integration, but since the existent techniques do not allow

to integrate ontologies in a completely automatic way [Noy, 2004], this is the only technique that guarantees consistency, without requiring a further user intervention. Moreover, since the XIRE retrieval component is based on IR techniques, approximate solutions to the ontology integration problem can be considered acceptable; instead, the whole GSO building process need to be fully automatized.

Construction of the GLSO and the Semantic Similarity Matrix

The GSO may result extremely large in size, which makes the semi-automatic process of mapping the GSO to the SPDO more expensive; moreover, only a subset of the terms of the ontologies used and recursively imported by the SWS descriptions are actually relevant for describing the SWS. To solve this problem a technique to reduce the ontology size is exploited and a GLSO (Global Light Service Ontology) is obtained. XIRE provides an ontology module extraction algorithm that exploits a traversal-based approach [Palmisano et al., 2009]. The algorithm, named SSiMap K-ex (Semantic Similarity and Mapping-driven K ontology module Extraction), is based on the extraction of (i) a subtree of the tree representing the unfolded concept hierarchy of the GSO, and (ii) a set of properties whose domains include the concepts in such a subtree. The subtree is built by traversing upwards the subclass relation until the root starting from the index terms I, and by traversing downwards the subclass relation through a path of an arbitrary length k. The algorithm is described in details in [Palmonari et al., 2010].

Another output created along with the GLSO is the Semantic Similarity Matrix (SSM), which is exploited later for building the query to be submitted to the IR engine. Let Sig^O be the signature of an ontology O, i.e. the set of all concepts, property and instance names occurring in O; $SSM = Sig^{GLSO} \times Sig^{GLSO}$ is a matrix whose values represent the semantic similarity between two objects in the GLSO; the function $sim : Sig^{GLSO} \times Sig^{GLSO} \rightarrow [0, 1]$, where $sim(x, x) = 1$, is a similarity function based on the structure of the ontology; the function may be defined according to for different similarity metrics (e.g. Conceptual Similarity, Lin, Resnik, ScaledShortestPath and ShortestPath)[Bernstein et al., 2005] and may take into account subclass paths, domain and range restrictions on properties, membership of instances, and so on. In particular, by analyzing the similarity results obtained with different metrics with the GLSO created for experiments (see next Section 5.8) we chose the Conceptual Similarity (see [Palmonari et al., 2010] for details), a similarity measure based on the reduction of the ontology to a taxonomy, then to a graph, in order to provide a positional analysis of concepts within the graph, assuming that

each arc has the same weight. When the reference ontology is composed of subontologies as the GLSO, this similarity metric tends to prefer nodes belonging to the same ontology, which is one of the reason why it has been chosen in our approach.

5.5 Mapping of Data and Service Ontologies

The process to discover the mappings between the SPDO and the GLSO exploits the MOMIS system. The two schemas represented in ODL_{I^3} are annotated according to WordNet as described in Section 2.4. These annotations are then exploited to generate a Common Thesaurus of relationships between the elements of the SPDO and the GLSO. Mappings between the elements from the SPDO and the GLSO are generated by exploiting the MOMIS clustering algorithm. In particular, the clustering algorithm relies on an input matrix where the columns (the rows) represent the source schema elements and the value in a cell weights the relatedness of the schema elements that are in the corresponding row and column. To compute the mappings between the SPDO and the GLSO, the clustering algorithm requires an input matrix with the SPDO schema names and the names of the GLSO concepts. The weights are obtained by taking into account measures based on syntactic, lexical and structural similarity that are computed exploiting the knowledge about the sources in the MOMIS common thesaurus. In particular, the **syntactic similarity** measures the similarity between the names used for describing the elements in the SPDO and in the GLSO. Several string similarity metrics have already been proposed [Cohen et al., 2003], e.g., Jaccard, Hamming, Levenshtein, etc. As our approach is independent from the similarity metrics selected we leave this choice to the application.

String similarity, unfortunately, may fail in highly heterogeneous environments that lack a common vocabulary. In these cases, it is critically important to be able to capture the meaning of a word and not only its syntax. For this reason, we employ a **lexical similarity** measure based on WordNet that evaluates the terms on the basis of their synonyms, hypernyms and hyponyms. In particular, our approach builds a weighted network of relationships exploiting the terms in Wordnet and the relationships in the MOMIS Common Thesaurus. The weight of the path connecting two terms having minimum weight measures the lexical similarity of these terms.

Finally, a **structural similarity** measure compares the concept structures for evaluating their similarities. Concepts with similar structures have a huge structural similarity value.

The algorithm for generating the input matrix is highly customizable,

thus making possible for the user to select the importance of each measure according to the source structures and contents. Moreover, the user may select the algorithm threshold in order to obtain big clusters (and consequently less selective associations between the SPDO and GLSO elements) or clusters where only strictly related elements are grouped. Mappings are then automatically generated exploiting the clustering result. The following cases are possible:

- **A cluster contains only SPDO classes**: it is not exploited for the mapping generation; this cluster is caused by the selection of a clustering threshold less selective than the one chosen in the SPDO creation process.

- **A cluster contains only GLSO classes**: it is not exploited for the mapping generation; it means that there are descriptions of Web Services which are strongly related.

- **A cluster contains classes belonging both to the SPDO and the GLSO**: this cluster produces for each SPDO class a mapping to each GLSO class. Mappings between the attributes of the classes are generated on the basis of the relationships held in the MOMIS Common Thesaurus.

As an example, consider the SPDO described in Section 4.2, and a piece of the *GLSO* concerning the class *Accomodation* and the attributes this class is domain of; using a dotted notation in the form of *"concept.property"* this piece of ontology is represented as follows:

```
Accomodation
    Accomodation.Denomination
    Accomodation.Location
    Accomodation.Country
```

The following mappings are generated with the application of our approach:

```
Hotel  -->  Accomodation

Hotel.Name -->  Accomodation.Denomination
Hotel.City -->  Accomodation.Location
Hotel.Country -->  Accomodation.Country
```

Observe that clusters can be exploited to correct false dissimilarities in the SSM. However, clusters provide also another solution to the false dissimilarities problem: when two terms t_1 and t_2 of the GLSO are clustered together, they are mapped to the same term s of the SPDO; when a query formulated in the SPDO terminology contains the term s, both t_1 and t_2 will be extracted as keywords and used to retrieve relevant services.

5.6 Data and Service Retrieval - Execution Time

Let us introduce a query expressed in the SQL language:

```
select <select_attribute_list>
from   <from_class_list>
where  <condition>
```

The answer to this query is a data set from the data sources together with a set of ranked services which are potentially useful, since they are related to the concepts appearing in the query and then to the retrieved data.

The query processing is thus divided into two steps:

- a data set from the data sources is obtained with a SQL query processing on the integrated SPDO view (as described in Section 4.5)
- a set of services related to the query is obtained by exploiting the mappings between the SPDO and the GLSO and invoking the execution of the IR engine.

Data results are obtained by exploiting the MOMIS Query Manager which rewrites the global query as an equivalent set of queries expressed on the local schemata (local queries); this query translation is carried out by considering the mapping between the SPDO and the local schemata. Results from the local sources are then merged by exploiting the reconciliation techniques.

Services are retrieved by the XIRE (eXtended Information Retrieval Engine) component, which is a service search engine based on the vector space model [Christopher D. Manning and Schütze, 2008], and implemented with the open source libraries Lucene [Hatcher and Gospodnetic, 2004]; as explained in the following, the query is a set of weighted terms. The process for the creation of the set of weighted terms is represented in Figure 5.4.

The process is started by MOMIS which extracts the terms in the SQL query that appear in the SPDO. Then, it checks which elements of the GLSO

are mapped on these terms of the SPDO and send these GLSO terms to XIRE; XIRE expands these keywords on the basis of the SSM, and creates a set of weighted term (the query vector) to be used to retrieve the related services by means of Information Retrieval techniques as described in next Section.

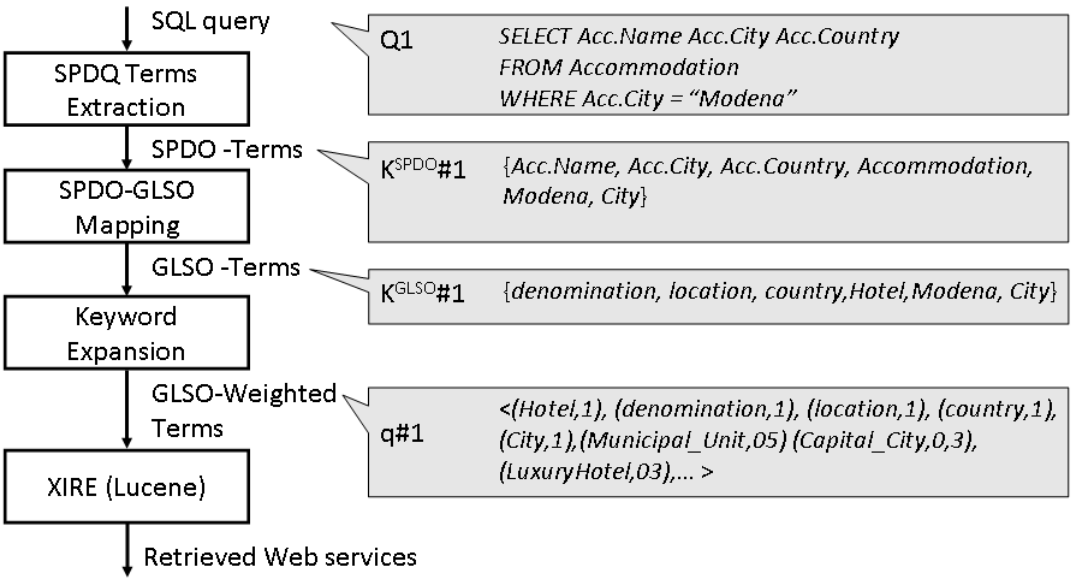


Figure 5.4: The query processing steps and their application to the reference example

5.6.1 Service Retrieval

Terms extraction. Given a SQL query expressed in the SPDO terminology, the set of K^{SPDO} of terms extracted from an SQL query consists of: all the classes given in the "FROM" clause, all the attributes and the values used in the "SELECT" and "WHERE" clauses, and all their ranges defined by ontology classes. An example is represented in Figure 5.4 where the set of terms extracted from the query Q1 consists of the set $K^{SPDO}\#1$.

Terms rewriting The set of terms K^{SPDO} extracted from the user query are rewritten in a set of keywords K^{GLSO} exploiting the mappings between the SPDO and the GLSO. Let us define a data to service ontology mapping function $\mu : Sig^{SPDO} \rightarrow \mathcal{P}\left(Sig^{GLSO}\right)$. The function, given a term $s \in SPDO$ returns a set of keywords $T \subseteq Sig^{GLSO}$ iff every $t \in T$ is in the same cluster of s. Given a set of terms $K^{SPDO} = \{k_0, ..., k_m\}$, each term k_i with $0 \leq i \leq m$ is replaced by the set of keywords returned by $\mu(k_0)$. By

using the mappings described in Section 5.2, and assuming $\mu(Modena) = Modena$, and $\mu(City) = City$, the set of keywords obtained in the reference example is the set $K^{GLSO}\#1$[5] represented in Figure 5.4.

Keywords expansion Semantic similarity between GLSO terms defined in the SSM is exploited to expand the K^{GLSO} set with additional terms into a vector $q =< (k_1, w_1), ..., (k_n, w_n) >$, where for $1 \leq i \leq n$, $k_i \in Sig^{GLSO}$, and w_i are weights that represent the importance of each keyword in describing the interesting services. The vector q is obtained by associating each keyword with a weight equal to 1, and adding a set of terms that in the SSM are similar to the given keywords up to a given threshold weight according to their similarity w.r.t. the given keywords.

More formally, let $simset_s(t) \subseteq Sig^{GLSO}$ be the set of terms of the GLSO such that their similarity w.r.t. t is greater than a given threshold th. Given a set of keywords $K^{GLSO} = \{k_0, ..., k_m\}$, the vector q is obtained as follows: all the keywords $k_i \in K^{GLSO}$, with $1 \leq i \leq m$, are inserted in q and are associated with a weight $w_i = 1$; for every $k_i \in K^{GLSO}$ the set $simset_s(k_i)$ is inserted in q, and each element $e \in simset_s(k_i)$ is associated with a weight $w_e = sim(k_i, e)$; duplicate terms in q are discarded, keeping the terms associated with the greatest weight.

For example, let us consider the set of keywords $K^{GLSO}\#1$ given in the reference example. Assume to set the normalized similarity threshold $th = 0,95$; $simset_{0,95}(City) \subseteq \{Municipal\_Unit, Capital\_City\}$,
$sim(City, Municipal\_Unit) = 0,999$ ($City$ is subclass of $Municipal\_Unit$ in the Travel ontology) and $sim(City, Capital\_City) = 0,995$ ($Capital\_City$ is subclass of $City$); a piece of the resulting weighted term query vector $q\#1$, including $Municipal\_Unit$,$Capital\_City$ and $LuxuryHotel$ (added in an analogous way based on the ontology including $Hotel$), is represented in Figure 5.4.

Services retrieval. Query evaluation is based on the vector space model [Christopher D. Manning and Schütze, 2008]; by this model both documents (represented by Web Service descriptions) and queries (extracted queries) are represented as vectors in a n-dimensional space (where n is the total number of index terms extracted from the document collections). Each vector represents a document, where indexed keywords have weights different from zero. The value of such weight is computed according to the weights of the six sections of the service description in which the keyword appears. We assume the implicit constraint query term (a single keyword)must appear in *at least one* section of a service description in order to retrieve that service.

[5]Remind that all the GLSO terms are URIs, although parts of the URI specification are omitted for sake of clarity

Based on the above assumptions the weight which at query evaluation time is associated with a keyword and a service description is equal to the maximum of the weights of the service sections in which the keyword appears.

5.7 System Architecture

To evaluate our approach, we extended MOMIS and coupled it with XIRE (eXtended Information Retrieval Engine), a semantic Web service retriever developed by the "DISCO" of the University of Milano Bicocca. The global architecture of the prototype developed is shown in Figure 5.5. The MOMIS

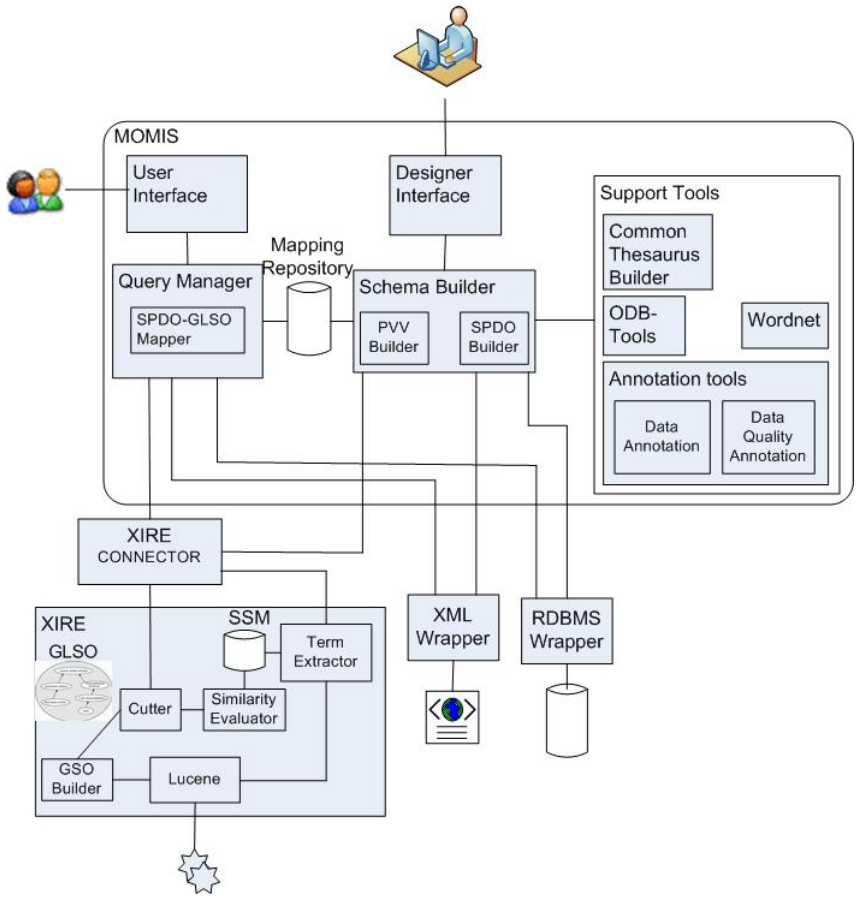


Figure 5.5: The Data and Service Aggregated Search Prototype

architecture has already been described in Section 2.6, therefore in this section we will focus on the description of the extensions I developed to couple MOMIS with the XIRE search engine.

5.7.1 MOMIS

MOMIS has been extended with new modules to create and query a PVV. These new components are:

- *PVV builder* which is in charge of generating the PVV that represents both the data sources and the services available in a peer (including the discovered mappings between the SPDO and the GLSO).
- *SPDO-GLSO Mappera*, which is in charge of extracting the keywords to be sent to the XIRE engine.
- *Xire Connector* that connects MOMIS and XIRE.

The PVV Builder

The PVV builder is a component of the MOMIS Schema Builder that provides the mappings between the SPDO and the GLSO (i.e. the ontology representing the web services in a peer) elements. The PVV builder implements a clustering algorithm that works on the basis of the descriptions of the GLSO elements (extracted by a specific wrapper), the SPDO description and set of semantic relationships between them computed by MOMIS. The mappings resulting from this process express one to one matches of SPDO classes and attributes into GLSO concepts and properties.

The SPDO-GLSO Mapper

The SPDO-GLSO Mapper extract the relevant keywords from a query on the SPDO. The relevant keywords are those identifying schema elements and searched values. This component then evaluates the mappings computed by the PVV builder, to extract, for each keyword, the correspondent terms in the GLSO, if they exist. Such terms are then sent to XIRE by means of the XIRE wrapper.

The XIRE Conector

The XIRE Connector is the component in charge of managing the interactions between MOMIS and the XIRE system. The connector mainly provides two tasks:

- *getGLSOschema()* translates the schema of the GLSO (expressed in OWL) into ODL_{I^3}.

- *serviceDiscovery()* enables the discovery of the services on the basis of a query on the SPDO. It is in charge of the invocation of the XIRE engine with the set of keywords extracted by the MOMIS Query Manager from a query on the SPDO (a global query) and collects the list of web services relevant to the global query returned by XIRE.

5.7.2 XIRE

The main modules of XIRE are the following:

- *Lucene* is the core of the module and is a information retrieval tool developed by Apache Software Foundation[Hatcher and Gospodnetic, 2004], available at http://lucene.apache.org/.
- *GSO Builder* is in charge of building the GSO based on the indexes built by XIRE.
- *Cutter* is the module realizing the module extraction algorithm described in Section 5.4.
- *Similarity Evaluator* is the components devoted to the definition of the semantic similarity matrix.
- *Term Extractor* is in charge, at query time,of extracting the set of weighted keywords provided to Lucene from the terms of the GLSO.

All these modules except for Lucene (which is an existent component) are described in the next subsections.

GSO Builder

The GSO Builder is in charge of building the global service ontology starting from the list of terms extracted by the semantic web service descriptions provided by the Lucene component. For each term, that is also a reference to an ontological concept, the GSO Builder imports the whole ontology associated to it. In such a way all related concepts are added to the GSO. GSO Builder makes use of the Jena framework, in particular the OWL API.

Cutter

Cutter is the module in charge of lightening, at set-up time, the GSO to improve the performance by implementing the SSiMap K-ex algorithm; the

configuration used set $K = 2$. For example in the experiment described in the next Section the Cutter module produced a GLSO by reducing concepts to a tenth of the original number: from 4300 concepts to 485.

Similarity Evaluator

At set-up time Similarity Evaluator builds a similarity matrix (SSM - see Figure 5.5) based on GLSO concepts: the matrix contains the similarity value for each couple of GLSO concepts. The Java library SimPack [Bernstein et al., 2005][6] has been used for building the Similarity Matrix. SimPack provides algorithms to calculate many syntactic and semantic similarity metrics (e.g. Conceptual Similarity, Lin, Resnik, ScaledShortestPath and ShortestPath[Bernstein et al., 2005]), which can be used, and eventually combined, in the XIRE component by configuring the Similarity Evaluator. To provide the normalized similarity value we exploited the Common Distance Conversion available in the SimPack library.

Term Extractor

Term extractor exploits the similarity matrix (SSM - see Figure 5.5) finding, for each keyword provided by the XIRE wrapper, the most similar concepts. This is realized by using a normalized similarity threshold (in a range from 0 to 1), that in our experiments was set to 0.95, in order to restrict as much as possible the terms proliferation.

5.8 Experiments

In this section an evaluation of the prototype described in section 5.7 is presented. Before describing the experiments done with the XIRE component, a preliminary consideration has to be introduced. As the IR based approach to Web service retrieval is quite a new approach to the location of relevant Web services, there exists no standard collection which allows to fully evaluate the effectiveness of such an approach. We recall that the effectiveness of an IR based approach is measured by means of two main measures, the Recall and the Precision, which have to be assessed with respect to a specific query. The Precision is the proportion of relevant documents over the documents retrieved in response to a query; the Recall is the proportion of relevant documents retrieved by the system over the documents truly relevant to the considered query. The main components of an IR experiment are

[6] `http://www.ifi.uzh.ch/ddis/simpack.html`

then a collection of documents, a collection of queries, and for each query the knowledge of the documents which are truly relevant to it. However, a similar collections have been defined to evaluate semantic Web service matchmaking (e.g. the OWLS-TC benchmark [Klusch et al., 2009] for OWL-S descriptions), which is an activity related to service retrieval (from an IR approach) although based on rather different assumptions. Therefore, due to the unavailability of a collection for text-based Web services retrieval, and aim at having as well an idea of the effectiveness of the XIRE component, we applied a preliminary evaluation of XIRE by using the OWLS-TC benchmark. It is the only collection frequently used and thus also regularly cited in literature. OWLS-TC is also the base of two of the other collections (SAWSDLTC and the Koblenz dataset [7] [8]). It can clearly be viewed as the current de facto standard of SWS test collections for OWL-S WS descriptions. Its popularity is due to its size and to the fact that, unlike all other collections (except for SAWSDL-TC), it does not only contain advertisements, but also sample requests and a complete set of binary relevance judgments of the advertisements with respect to the requests. A version of the benchmark introducing also graded relevance sets, namely the OWLS-TC 3.0, has been introduced only very recently.

For a discussion about the different assumptions behind IR-based service retrieval and matchmaking, we refer to Section 1.3.3. By this simple and preliminary experiment we want to show that based on the analysis of the textual descriptions and of the concepts specified in a Web Service definition, the XIRE approach is able to retrieve more relevant material than the one retrieved by a usual match-making approach to web service retrieval.

5.8.1 Results evaluation

We conduct our experiments on a computer equipped with a Core2 T5600 1.83GHz, 2Gb of RAM and Windows XP Professional as operating system. We selected from the benchmark the only 4 queries related to the tourist domain and applicable to the data cointained in the SPDO described in Section 4.2 (queries 22, 23, 25 and 26). We considered only the set of semantic web services related to travel (165 services). These queries, expressed in natural language, are the following:

- *Query 22* The client aims to know about the destination for surfing.

[7] `http://projects.semwebcentral.org/projects/sawsdl-tc/`

[8] `https://www.uni-koblenz.de/FB4/Institutes/IFI/AGStaab/Projects/xmedia/dl-tree.htm`

- *Query 23* The client aims to know about the destination where facilities for sports hiking and surfing available
- *Query 25* The client aims to know about the destination of the organization for a certain type of surfing
- *Query 26* The client aims to know about hotel in a city of a country

Starting from the description in natural language of these queries, we built 4 SQL queries, shown in Figure 5.6.

Query 22
```
SELECT *
FROM ACTIVITY, CITY
WHERE ACTIVITY.ZIP_CODE = CITY.ZIP_CODE
AND ACTIVITY.NAME = "surfing"
```

Query 23
```
SELECT HOTEL.ZIP_CODE
FROM ACTIVITY, FACILITY, HOTEL
WHERE FACILITY.HOTEL_ID = HOTEL.ID
AND FACILITY.ACTIVITY_ID = ACTIVITY.ID
AND
( ACTIVITY.NAME = "surfing" OR
ACTIVITY.NAME = "hiking" )
```

Query 25
```
SELECT CITY_NAME
FROM CITY, EVENT
WHERE CITY.ZIP_CODE = EVENT.ZIP_CODE
AND
( EVENT.NAME LIKE '%surfing%'
OR
EVENT.DESCRIPTION LIKE '%surfing%' )
```

Query 26
```
SELECT HOTEL.*
FROM HOTEL, CITY
WHERE CITY.ZIP_CODE = HOTEL.ZIP_CODE
AND CITY.COUNTRY = 'Italy'
```

Figure 5.6: The SQL queries used in the experiments

At set-up time the indexing of the set of SWS took 640 milliseconds, and the the GSO starting from the terms extracted by Lucene is composed by 4300 concepts. The similarity matrix for ConceptualSimilarity took 14 seconds. The SSiMap K-ex algorithm produces a GLSO with 485 concepts setting $K = 2$.

The run time evaluation is based on 4 different software configurations.

- **Pure IR Engine**. In this configuration we extract from queries (expressed in natural language) the relevant keywords and we use them as input for Lucene.
- **IR Engine with expansion** In the second configuration we consider the keyword previously defined and we use the semantic similarity matrix to improve the set of keywords considered.
- **IR Engine with MOMIS** In this configuration the keywords are extracted by MOMIS starting from the query expressed in the SQL syntax and we do not use the semantic similarity matrix

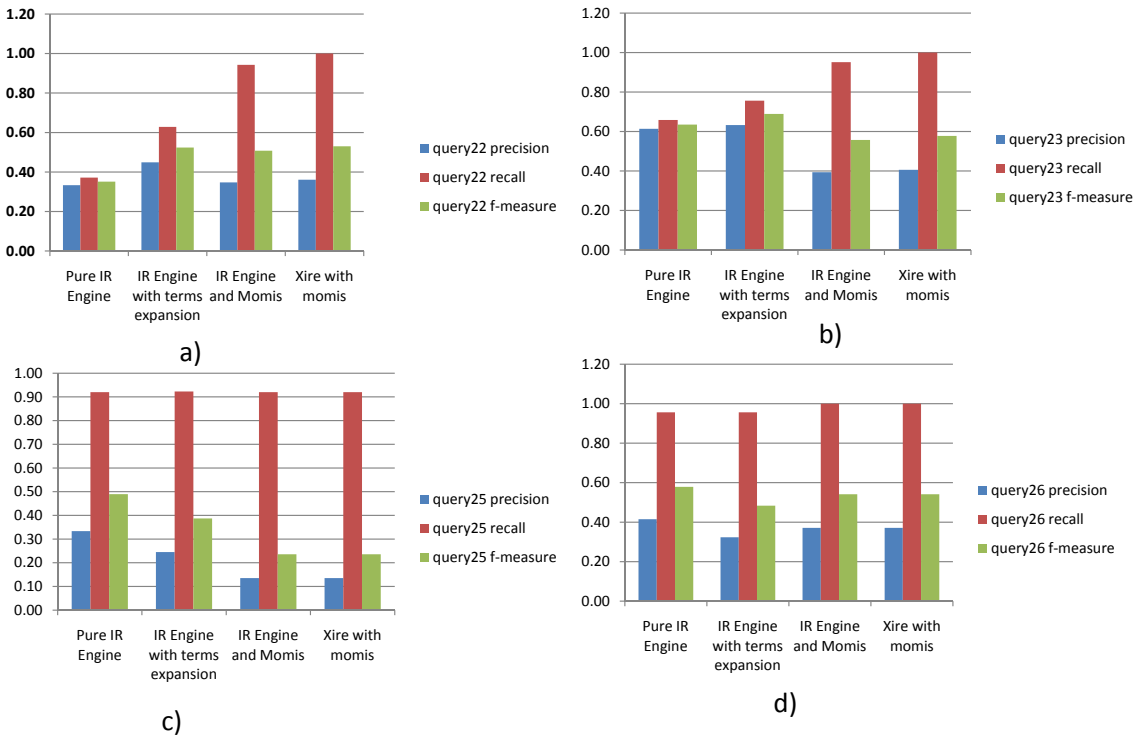


Figure 5.7: Precision and Recall of queries 22,23,25,26

- **XIRE with MOMIS**. In the last configuration we use all software components of XIRE and MOMIS representing the whole prototype

Results of the experiments are described in figure 5.7, and figure 5.8, where for each query we report the precision, recall and f-measure (the weighted harmonic mean of precision and recall) in all 4 different configurations.

Analyzing the results, we can see that when we consider the whole prototype the recall is extremely high, always around 1. On the contrary, precision is moderate. The relatively low precision (which we expected) is due to the main fact that the benchmark we used consider as relevant much less documents than those related to a keyword based search. Precision will certainly be higher when a suitable evaluation experiment (in the IR style) will be set up.

For example, concerning query 26, keywords are CITY, COUNTRY, HOTEL. We assume that all SWS descriptions containing at least 2 of such keywords or semantically close to the keyword are relevant. Notice that in our approach it is non influential if the keyword is an input or an output, while in the case of matchmaker systems this is quite relevant (see Section

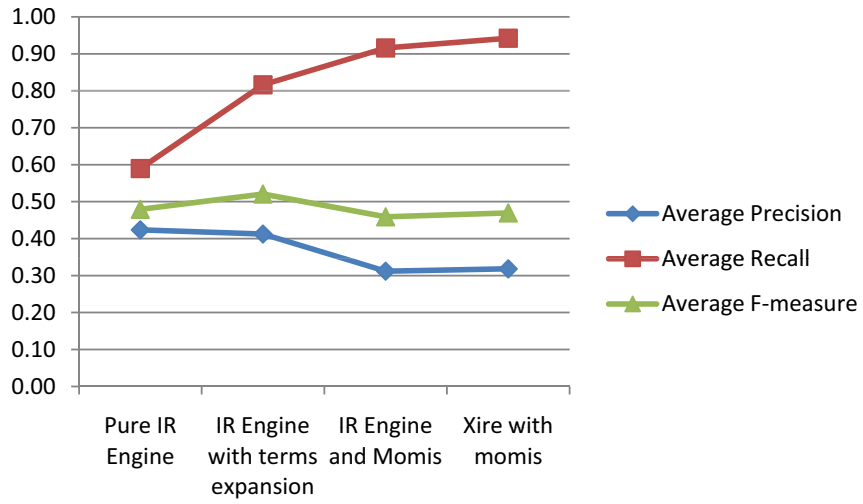


Figure 5.8: Average Precision and Recall over 4 queries

Query Number	SPDO terms	GLSO terms
Query 22	5	38
Query 23	8	57
Query 25	7	16
Query 26	10	35

Table 5.1: keyword identified for each query

5.9 for further discussion). If we recalculate the precision with this new set the value will be equal to 0,65. Moreover precision can be also increased by fully exploiting the structure of Web Services: to this aim we intend to apply techniques for structured documents retrieval, so as to be able to specify keywords related to the distinct sections. To this aim in XIRE we have already defined a data structure which considers input, output and description part of each SWS description that can be further investigated. Another observation is that with respect to a pure IR-based approach the use of SPDO (provided by MOMIS) increases significantly the result in terms of recall and reduce the precision (even if we have to consider the relevance problem above mentioned). This can be explained by considering table 5.1 where for each query we report the number of keyword found in the SPDO and in the GLSO.

In table 5.1 we notice that the number of identified keywords in the SPDO is significantly lower than the number of keywords identified in the GLSO. Consequently the number of relevant retrieved SWS description increases thus explaining the increase of recall.

5.9 Discussion

In complex and added value processes, users need both to search data and to discover interesting related services, by means of a new generation of aggregated search technology. In this chapter I presented an original contribution in this field by proposing an integrated approach for data and services discovery. The approach presented in this chapter can be described as a *Service as Data* approach, as opposed to *Data as a Service* approaches. In the Service as Data approach, informative services are considered as a kind of data source to be integrated with other data sources, to enhance the domain knowledge provided by a Global Schema in a networked organizations. To the best of our knowledge this approach is completely new in the literature. I described an original approach to connect data descriptions and semantic web services descriptions at design time starting from terms extracted by an information retrieval engine. The proposed approach is based on an innovative algorithm that reduces the service ontology to be easily managed. Starting from a data query expressed over the Semantic Peer Data Ontology, our system first extracts the useful terms from the data queries. As a second step it identifies in the Global Lite Service Ontology the terms related to those extracted from the data query; these terms will be used as keywords in the query to be submitted to the IR engine. Finally this set of keywords is augmented by a set of similar keywords determined by analyzing the concept similarity matrix. Experimental results based on OWLS-TC, the most widely accepted benchmark in the field of semantic web services show that our approach ensure a high level of recall (around 100%) and an acceptable level of precision. Such results show also that our approach is recall-oriented: this is due to the fact that a keyword based search enhanced by a semantic query term expansion is able to locate much more relevant services than the ones located by usual matchmaking techniques. On the contrary, the relatively low precision is due to the main fact that the benchmark used to make these evaluations considers as relevant much less documents than those related to a keyword based search. A fundamental difference between most of the traditional service matchmakers and our approach is in terms of objectives: we are interested in finding a set of services related to a set of terms extracted from a data query, and we do not distinguish between inputs and outputs

as traditional service matchmakers do. In short, the hybrid matchmakers mentioned in Section 1.3.3 perform approximate retrieval of OWL-S descriptions, while our approach is explicitly based on a "relevance"-based retrieval against a set of keywords. The retrieved services might be relevant to the set of keywords w.r.t. their inputs, outputs or descriptions. Building the retrieval approach on this relevance principle provides also the motivation for choosing a semantic concept similarity metric to perform query expansion.

Chapter 6

Integration of Government Spending Data on Hadoop

In this chapter I describe an experience in developing a Hadoop based integration flow to collect and integrate publicly available government datasets related to government spending. This work was conducted at the IBM Almaden Research Center of San José, California, with the Information Integration Group as part of the Midas project. The objective of Midas is to design an integration framework in a distributed environment to guarantee for scalability and flexibility. The integration designed in this work was tested with publicly available datasets in the government domain, integrating these data to allow users, U.S. taxpayers in this case, to easily access the data their government discloses on the web in different websites. This work also provides users with easy-to-use tools and an API to query and explore this data to gather information from the integrated data that allows evaluating how tax money is spent. In particular, during the six months I spent at the IBM Almaden Research Center, I participated in the definition of the integration flow and I was responsible of all the implementation of the integration flow in the government domain.

6.1 Motivation

IBM reports[1] that 15 petabytes (10^{15} bytes) of data are created every day, eight times more than the information in all the libraries in the United States. Data integration will necessarily need to consider distributed architectures to cope with the increasing amount of data that are produced, especially in some domains. Governments are making a great effort lately to exploit the new Internet technologies to disclose their information, especially those concerning how tax money is spent. The government domain is thus a perfect use case for data integration frameworks in distributed environments, since the amount of data governments produce is huge and data typically need to be collected from different government agencies. The objective of the project described in this chapter is to extract, cleanse, and integrate datasets publicly available at government web sites and provide a deep insight into U.S. government spending. The publicly available datasets we used contain structured information about members of the United State Congress, congressional districts, federal agencies, government contractors, and the spending itself. For scalability and flexibility reasons, JSON was chosen as our data model and Jaql as data manipulation and transformation language running on Hadoop (see Section 6.2 for details). I built a scalable system architecture with an end-to-end application addressing data integration issues and mixed keyword-based search and SQL query interface to achieve deep insight

[1] `http://www.ibm.com/smarterplanet/us/en/business_analytics/ideas/`

through BI-like aggregate queries. Specifically, we solved an ETL-style integration problem through stages like data extraction, data cleansing and normalization and entity resolution.

The chapter is organized as follows. Section 6.2 overviews the software used for the integration flow and describes its steps, while Section 6.3 gives a more detailed description of the domain. Sections 6.4 through 6.7 describe the overall Midas integration flow through different stages. These stages are broken into: data extraction stage in Section 6.4, cleansing and normalization stage in Section 6.5, join stage in Section 6.6, and finally search and query interface in Section 6.7.

6.2 Overview of the System

The integration flow is composed of a few key components that are nicely connected on a system architecture mixing Hadoop and DB2 as our underlying infrastructure. After describing the software used to realize the integration flow, I briefly present its key components.

6.2.1 Software used to realize the integration flow

The integration flow is realized exploiting open-source tools for distributed architectures and data manipulation that are presented in this section.

Javascript Object Notation (JSON)

JSON[2] is a lightweight data-interchange format, easy for humans to read and write and easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers: JSON is built on two structures:

- A collection of name/value pairs
- An ordered list of values

Hadoop

Apache Hadoop[3][4] is a Java software framework that supports data-intensive distributed applications under a free license.[1] It enables applications to

[2] `http://www.json.org`
[3] `http://hadoop.apache.org`
[4] `http://wiki.apache.org/hadoop/ProjectDescription`

work with thousands of nodes and petabytes of data. Hadoop implements a computational paradigm named map/reduce [Dean and Ghemawat, 2010], where the application is divided into many small fragments of work, each of which may be executed or re-executed on any node in the cluster. In addition, it provides a distributed file system called **HDFS** that stores data on the compute nodes, running the work near the data and providing very high aggregate bandwidth across the cluster. Both map/reduce and the distributed file system are designed so that node failures are automatically handled by the framework.

Jaql

Jaql[5] is an open-source query language developed at the IBM Almaden Research Center, designed for JSON. Jaql is primarily used to analyze large-scale semi-structured data. Its core objectives are the parallelism, delegated to the Hadoop framework Jaql runs on, and extensibility, guaranteed by the possibility to easily extend Jaql with User Defined Functions (UDF).

6.2.2 The Midas Integration Flow

An overview of the Midas process is presented in Figure 6.1. The figure shows the process starting from the relational versions of the data sources (DB2). First, relational data is converted into JSON format. Then, Jaql, extended with User Defined Java Functions (UDF), is used for the cleaning and integration process. The output of the process is an integrated "clean" dataset in JSON format which can be indexed for keyword search or converted back to the relational model to allow posing more interesting aggregate queries. Note that DB2 is chosen for the initial stage of the flow so that we can leverage existing DB2 utilities to profile the data, but the process can ideally start converting the original data directly into JSON. For the final stage, we also convert the data from JSON format to relational model and then import them to DB2. This is because currently, for real time queries, SQL aggregate queries perform better than Jaql aggregate queries. The key steps of the integration flow, realized with the software presented in the previous section, are now briefly described.

Exploration/Extraction

Manual exploration is required to understand related public datasets available at different ".gov" websites. Extraction refers to the process of down-

[5] `http://www.jaql.org`

loading, and sometimes crawling, data and reformatting it for our purpose. After all the datasets are brought in-house, additional data exploration and profiling are needed to derive the data characteristics and understand its semantics. Data characteristics are needed to define the cleansing/normalization step, while understanding of its semantics helps us determine the join-key in the entity resolution step.

Cleansing/Normalization

To present the final data in some consistent format and, more importantly, to join data coming from different sources, data cleansing and normalization is needed. For instance, Congress members' names need to be normalized (standardized) so that entity resolution at the next component can easily be applied.

Entity Resolution

Entity resolution [Bleiholder and Naumann, 2008] is needed in deduplicating tuples and in joining tuples from related tables. The former is needed, for example, to integrate different records concerning the same contract from USAspending.gov (see Section 6.4 for details). The latter type of entity resolution is needed, for example, to create the Congress members dataset, as it is created with data coming from different web pages, or to combine together data about earmarks with the information about their sponsors.

Search API/UI

Each integrated JSON data set has been indexed using the Lucene[6] library. Thus, the required fields of each record can be searched, and records matching the specified value for a certain attribute are immediately retrieved even for datasets with millions of records. All the Lucene indices can be queried by means of a keyword-based search User Interface (UI) based on PHP programming.

JSON-to-RDB import

In addition to the search interface over our nested objects in the JSON format, we also wanted to support a SQL API to these datasets so that traditional BI applications can be built on top over our relational tables or views.

[6] `http://lucene.apache.org`

I have implemented a JSON-to-RDB export in two phases: (1) Jaql is used to flatten and normalize our JSON objects, then (2) a Java module imports flattened JSON objects to DB2 tables with the same structures (or schemas).

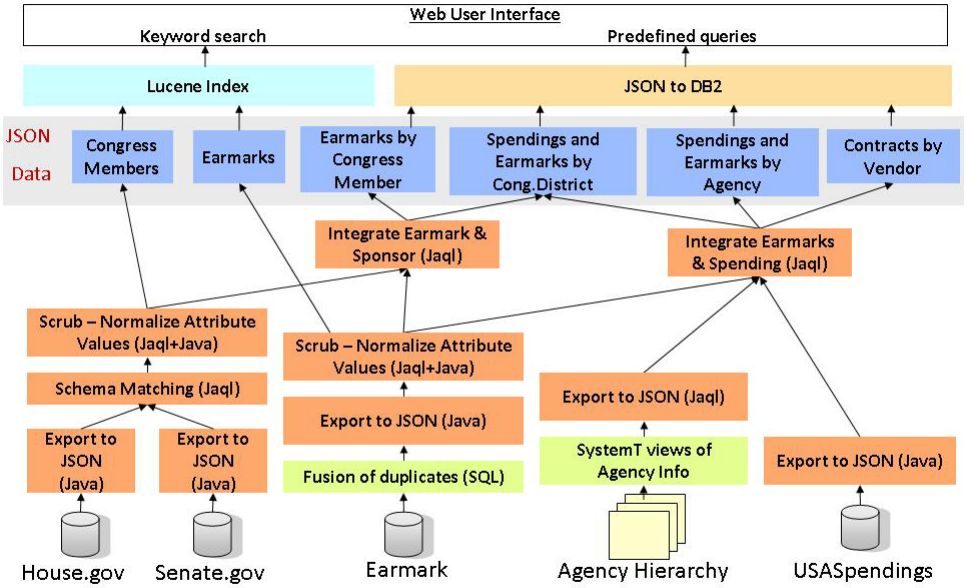


Figure 6.1: Midas process for the government domain

Pre-canned Aggregate Query API/UI

Once the datasets have been flattened and imported into DB2 tables (with explicitly declared and normalized join keys), we can support aggregate SQL queries at the API. For the UI, I built some sample pre-canned aggregate queries, for example:

- Return a list of Congress members ordered by number of earmarks sponsored in 2008.

- Return a list of 2008 earmarks ordered by amount.

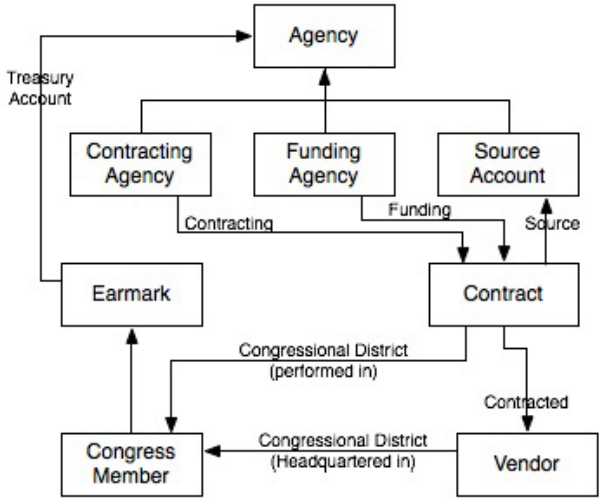


Figure 6.2: The government domain conceptual schema

6.3 Description of the domain

Figure 6.2 illustrates how the different government entities we considered are related to each another, allowing for deeper analysis. Both spending entities (contracts and earmarks) contain information related to Congress. Each earmark is sponsored by members of Congress while contracts authorize work to be done in specific states and congressional districts which can then be tied to their respective Senators and House representatives. The district of the vendor that is hired for the contract is also present in the data, thus making it possible to relate the vendor with a Congress Member. Furthermore, both spending entities are related to government agencies. Earmarks provide information on the agency that received the funds from Congress. Contracts contain data on all agencies that were involved. By bringing together detailed information on each of these entities (Congress, spending, federal agencies) from independent sources, we can take advantage of their relationships and provide much deeper insight than a single source can.

6.4 Data Source Description & Data Extraction

In this section I describe in detail each data source involved and the process to collect all the data to be used in the Midas integration flow.

6.4.1 The Congress Member Dataset

Senate.gov is the official website of the U.S. Senate. It provides the complete list of Senators with details about their party, state, website, and contact information. The information, in addition to the HTML web pages, is also available in XML format. Midas automatically download and parse the XML file of the Senate directory using a Java program, and then create and execute an `INSERT` statement into DB2 through JDBC.

House.gov is the official website of the U.S. House of Representatives. The House directory is an Excel file with attributes including each Representative's first name, middle name, last name, suffix, complete address and title. Five additional attributes (party, state, congressional district, phone, and office room) are available in other HTML tables on the website. The Excel file and the HTML pages are imported into DB2 as CSV files. The final House data source is obtained by joining the different information using the House members' full name (i.e., the concatenation of first name, middle name, and last name) as the join key.

6.4.2 The Spending Dataset

Earmarks.omb.gov is a website curated by the Office of Management and Budget (OMB) that provides data on earmarks approved by the U.S. Congress. The semantics of earmarks are precisely defined by the OMB as *"funds provided by the Congress for projects, programs, or grants where the purported congressional direction circumvents otherwise applicable merit-based or competitive allocation processes, or specifies the location or recipient, or otherwise curtails the ability of the executive branch to manage its statutory and constitutional responsibilities pertaining to the funds allocation process."*

For every earmark, there are several different attributes that describe its details: the amount of money approved, a description of the earmark, the program the earmark is part of, a citation of the bill the earmark appears in, the Department of the Treasury code for the account where the earmark has been specified, and the certifying official. Furthermore, information about the Congress members (i.e., name, state, congressional district) that sponsored the earmark and information about the recipient of the earmark are reported.

The earmark data is provided as a CSV file that was loaded into DB2. Some data representation issues had to be faced as each "logical" earmark record is often horizontally partitioned into two tuples, with each tuple having

a disjoint set of attributes that contains data. The only exception to this rule is that the earmark id, a join key, is present in both tuples. Specifically, the first set of attributes is related to sponsors and the second set of attributes is related to recipients. There are few exceptions to the statement above. Only one tuple per logical earmark is present if the earmark has no designated sponsor (e.g. Presidential earmarks) or no recipient information (there is only one such case, which is likely due to missing data).

Another exception occurs when there is more than one co-sponsor per earmark. Some very "popular" earmarks have as many as 58 co-sponsors. As a result, that logical earmark record is partitioned into 59 tuples in the CSV file (with 58 co-sponsor tuples and one recipient tuple). The last type of exception arises when a single earmark (identified by the same id) with the same sponsor is further divided into multiple entries. For example, a $500,000 USD earmark may be divided into 10 different tuples each with $50,000 USD. All 10 such tuples share the same information and earmark id. The situation becomes more complicated when there are multiple entries sharing the same id and, at the same time, multiple co-sponsors. In such a case, the same amount of earmark money is counted multiple times as a Cartesian-product across each co-sponsor. Thus, to derive the total amount of earmark money, we have to divide the sum of money by the number of co-sponsors. I cleaned all these exceptions using views in DB2.

USAspending.gov is a website developed by the Office of Management and Budget in response to the Federal Funding Accountability and Transparency Act of 2006 which mandated a publicly searchable website that contained comprehensive information on all U.S. federal spending. Spending is broken down into several categories: contracts, purchase cards, grants, loans, subcontracts, and subgrants. Data is available for each year starting from the fiscal year 2000. As of now, we only consider the contracts data from the fiscal year 2008.

Data is available in multiple formats. It can be downloaded in XML format using the provided API, or searched using a web interface that allows for the data to be downloaded in multiple formats (e.g., XML, tab or comma-delimited ASCII format). After obtaining a complete set of data, we imported it into DB2.

For the fiscal year 2008, there are approximately 4.4 million distinct contract entries which span across 25 "major agencies," as labeled by the website. The data itself is quite extensive, with each entry possessing 152 different attributes. The information within the data can be categorized into three related concepts (or entities): details of the contract (such as dollar amount,

timeline, location of the work), information on the federal agencies involved, and information on the contractor hired.

During our data exploration phase, we faced three significant issues: (1) dealing with missing values and the meaning of missing values, (2) identification of a global key for contracts, and (3) dealing with data quality issues by looking for reference tables for the cleansing and data value standardization.

In DB2, I created three views on the spending data: contract information, agency information, and contractor information.

6.5 Cleansing and Normalizing Data

6.5.1 Merging House and Senate Datasets

At this point, all our data sources are available as DB2 tables or views. The Midas workflow starts converting data from House.gov and Senate.gov into JSON using a Java converter. The process is straightforward: every instance value of an attribute from a table is converted to a JSON object *attribute: instance*. Two different (flat) JSON files are created and stored in the the Hadoop Distributed File System[7] (HDFS), one for the Senate and one for the House. Then, using a Jaql query, the two JSON files are fused together (union), and the format of some attributes is normalized both for the purpose of joining and presentation: for example, the attribute "full name" is created as ⟨last-name, first-name middle-name suffix⟩, or the congressional district is normalized in the form of two-letter state code followed by two digits (e.g. CA-01). The output is then the Congress JSON file in the HDFS.

6.5.2 Cleaning Earmarks

The earmarks data is translated into JSON using the same Java converter. Simple data cleansing is performed in Jaql: the attribute "sponsor full name" is created in the same way as in the Congress source, the sponsor congressional district is normalized as before and the resulting JSON version of the Earmarks.omb.gov data source is stored in the HDFS.

Before joining the earmark JSON file with the Congress JSON file, the sponsor names (Congress member names) in the earmarks are further normalized according to the names appearing in the Congress source. This process is accomplished by a Jaql query running on Hadoop. This step is necessary since there may be some discrepancies between the names appearing in one source with respect to the names appearing in another source. We

[7] `http://hadoop.apache.org/hdfs/`

observed the following discrepancies: (1) the first name is swapped with the middle name, (2) the first initial is swapped with the middle initial, and (3) a double last name "Bono Mack" in the House data appears just as "Bono" in the earmark data set.

6.5.3 Cleaning USAspending data

Contracts data coming from USAspending.gov are converted into JSON by the usual Java converter. Then Jaql is used as manipulation language to perform some entity resolution steps. Different records are available for every contract, usually if the money obligated for that contract changes or the actual contract value has been modified while the contract is being executed. In this case, instead of changing the record associated with that contract, a new record is added to the dataset with the new information. Each record is considered in the USAspending.gov website as a single, independent contract. Thus, the total value of an actual contract (information which is not available on the original website) is the sum of the values of each record reported for that contract. Negative values are also allowed for contract values, and it is not clear if in this case the actual value of the contract has decreased, or money has been de-allocated by the agency for that contract, or some other meaning. Jaql manipulations (group by) are then performed to create a single record for each contract, identified by its procurement instrument id, containing all the different amounts of money in the different records concerning that contract. The USAspending data set is then ready, containing a single record for each contract with the actual total value of the contract as the sum of the different record values.

The same data set can be processed to create a different aggregated data set, with an entry for each major contractor. In this way we obtain information about the total number of contracts awarded to a contractor. This aggregation is performed considering the "parent company name" attribute: different divisions of the same company can be contracted by a federal agency, thus reporting a different name for the contractor in the contract record. We aggregate the contracts by parent company to identify the total amount of federal contracts awarded to a single corporation.

6.6 Joining Different Data Sources

Once the data cleansing phase has been performed, the data sets can be easily joined: the sponsor full name can be used as a join key to link earmarks data to Congress member information, allowing aggregating earmarks

by sponsor name or by congressional district. In this case, each JSON object corresponds to a Congress member, with information about all the earmarks he/she sponsored, or to a congressional district, with all the information about the earmarks that were sponsored by the Congress member of that congressional district. In the second case, the contracts from the USAspending.gov data that were performed in that congressional district are available for integration.

Correlating earmarks with contracts data coming from USAspending.gov is less straightforward. Earmarks are funds given to a government agency, while contracts are money spent by a government agency. Thus, it is meaningful to aggregate the different data by agency, for example to check what percentage of an agency budget comes from earmarks.

The problem arises from the different methods used in the different government sources to identify different agencies. Earmarks.omb.gov and USAspending.gov use a combination of names and codes to identify and associate government organizations with individual records. In an attempt to establish a link across these datasets one must extract the US government organizational hierarchy from which those names and codes are obtained. In our work we discovered that there are four classes of hierarchies, codes, and names within the referenced documents that encompass the US governments organizational hierarchy.

In the data sources we considered, we can distinguish hierarchical relationships and budgetary relationships. As an example, agency X that reports to agency Y, from an organizational point of view, can manage a treasury account for some of its activities that is part of the budget of the Agency Z. The problem is the poor documentation available about these codes, causing even government employees to sometimes have problems understanding them. Once we figured the meaning of these codes out, referring to some official U.S. government documents (namely NIST Special Publication 800-87 Revision 1-2008 and FIPS 95-2), we created a JSON data set representing the hierarchy of the different agencies. This data set is then used in the integration flow as a reference for the relationships among the different agencies that appear in the data sources. In this way it became possible to trace earmarks that enter an agency budget, and the contracts that are paid by that agency.

6.7 Interface to the processed data

The output of the steps described above is a clean dataset of resolved entities with information coming from different data sources. The process normal-

izes data among the different sources, thus allowing joining related datasets. Combining the different available datasets enhances the information provided by a single data source. As an example, if we consider the earmarks information provided by the Office of Management and Budget, we can integrate this data with the data from the Congress websites, namely House.gov and Senate.gov, to give a more complete picture of the earmark and of the Congress members that sponsored that earmark.

As an interface to the results of our integration flow, we provide different methods for accessing the clean data sets we obtained, depending on the level of granularity and on the querying interface required by the different categories of users our work is intended for. The resulting JSON dataset stored in HDFS was indexed with a Lucene-based search engine for keyword-based search. The keyword-based search is available either by means of an API or from a Web User Interface to ease the human consumption. An example of the results obtained by means of the Web User Interface to the keyword-based search is shown in Figure 6.3.

On the other hand, the JSON to DB2 step creates a normalized clean database where specific queries can be posed. An API has been developed to provide programmatic access to the database, and a Web User Interface is also available for certain pre-selected aggregate queries with user-specified predicates. As an example for the earmark data set, the list of Congress members ordered by the number of earmarks they sponsored can be easily obtained. The results can then be browsed since some of the attributes shown by the Web User Interface link to other queries that might provide further information about the data we are looking at. For example, an earmark id will lead the user to all the information about that earmark and also to a list of all its sponsors. Following the link of a Congress member name, we get the list of all the earmarks sponsored by that Congress member.

As another example, earmarks can be ordered by the total amount of money approved for that earmark, which can be used to find the highest funded earmarks in a given fiscal year.

Other examples of the query that are made possible by the Midas integration flow are:

- Earmarks ordered by their total amount.
- Congress members ordered by the number of earmarks they sponsored.
- Congress members ordered by the total amount of the earmarks they sponsored.
- Congressional districts ordered by the total amount of contracts awarded in their area.

Earmark Short Description	: ABC Unified School District, Cerritos, CA for an after-school program at Melbourne Elementary School
Earmark ID	: 253021
Earmark Code	: FIE1
Sponsor	: Linda Sánchez Representative CA-39
Treasury Account	: 910204
Certifying Official	: Budget Service Director
Enacted in year	: 2008
Budget Enforcement Act (BEA) Category	: Discretionary
Citation Reference	: Joint Explanatory Statement to accompany H.R. 2764, Division G (Labor/HHS/Education)
Citation Source	: Appropriations Statute - By Reference
Citation Excerpt	: Consolidated Appropriations Act, 2008 (P.L. 110-161), Division G: Provided further, That $103,293,000 of the funds for subpart 1, part D of title V of the ESEA shall be available for the projects and in the amounts specified in the explanatory statement described in section 4 (in the matter preceding division A of this consolidated Act); Joint Explanatory Statement: The amended bill includes funds within FIE to support the following projects in the following amounts: ABC Unified School District, Cerritos, CA for an after-school program at Melbourne Elementary School, 195,000
Citation Comment	: First-time Earmark: When determining whether the earmark was a first-time earmark, the Department of Education examined only fiscal years 2004 through 2007. The earmark was identified as a prior earmark if the recipient received a Department of Education earmark in any of those years, whether or not the award was for the same project activities. All of the earmarks specified in the College Cost Reduction and Access Act (P.L. 110-084, Title I, Sec. 103) were defined as first-time earmarks for the purpose of this exercise.
Amount	: $ 192,000
Program	: Fund for the Improvement of Education: Programs of National Significance
Recipient Phone	: (202) 205-4352

Figure 6.3: Earmark data obtained by means of the Web User Interface to the keyword-based search

The above list is just an example of the queries that are available from the Web User Interface to the DB2 version of the integrated data.

6.8 Discussion

In this chapter I described my experience in realizing an integration flow of JSON data exploiting the Jaql query language on the Hadoop framework. The integration concerned government data in order to enable exploring U.S. government spending data. Jaql was used as it allows expressive data manipulation and transformation, and it also eases data cleansing and normalization steps through Java UDF extensions. Hadoop was used as it provides a reliable and scalable framework for distributed parallel computing. In fact, scalability is a major concern for data integration and especially in integrating government data as more and more data about government spending is being disclosed lately. I also provided different tools to access and explore the output dataset, namely, keyword-based search and BI-like aggregate queries.

The work presented in this chapter is an interesting example of an integration flow in a distributed environment that shows two interesting results from the perspective of data integration application on distributed architectures: I designed and developed an integration flow that can be seen as an appli-

cation directly implemented on the cloud, and the results of this integration flow were made available from the could through an API in a Data as a Service approach. The Hadoop framework is an open source project providing an environment with a distributed file system that easily allows realizing a cloud for data storage and the execution of distributed applications on commodity hardware. Jaql is a powerful yet simple data manipulation language that transparently runs on Hadoop and easily allows numerous operations on data in JSON format. The integration flow presented in this chapter could thus be generalized to provide an architecture for data integration on the cloud: Jaql can be used as query language, and the UDFs developed in this work can potentially be made available as a data cleansing and normalization library to be used with Jaql.

Chapter 7

Conclusions

This thesis focuses on Semantic Data Integration systems, and in particular on applications and extensions of the MOMIS system.

I described a real application of the MOMIS system within the context of the Cerealab Project where I developed a unique ontology providing both molecular and phenotypic data about wheat, barley and rice, integrating existing molecular and phenotypic data sources and data provided by the Cerealab project. This application perfectly describe the advantages provided by the MOMIS approach especially in the automatic discovery and definition of mappings between the global schema and the data sources. The integrated schema allows users to retrieve data coming from numerous data sources by querying a single interface instead of navigating through numerous web databases, querying them and then manually fusing the information obtained. The Query Composer available in MOMIS also represents a valuable tool for users of this kind of applications as they usually have low IT expertise and thus need a user-friendly interface. This integrated ontology is also a very important contribution as it can improve the breeding process allowing cereal breeders to find the right molecular markers to be used to intentionally breed certain traits, or combinations of traits, over others. To do this, access both to molecular data and phenotypic evaluation of traits is required. No resource was available so far that combined both these two kind of data and thus many data sources had to be accessed and the information obtained had to be combined manually. With the system presented in this thesis both molecular and phenotypic data are available through a single graphical interface. Thanks also to the quality of this work, the Cerealab Project have been re-funded by the Emilia Romagna government and the Cerealab database is now being improved within the SITEIA laboratory.

In Chapter 4 I introduced an extension of the MOMIS integration system to implement a tool that allows a user to create and query an integrated view of data and multimedia sources. This work required both the extension of the ODL_{I^3} language to represent multimedia objects in MOMIS and introduced interesting problems in the query processing. I described some cases where multimedia constraints can be expressed in a global query without requiring multimedia processing capabilities at the global level. This is an interesting result, since it upsets the usual paradigm on which mediator systems are based, stating that the query processing power of a mediator is greater than the one of the integrated data sources

The main contribution presented in this thesis is though an integrated approach for data and service discovery. In Chapter 5 I introduced a *Service as Data* approach, as opposed to *Data as a Service* approaches. In the Service as Data approach, informative services are considered as a kind of data source to be integrated with other data sources, to enhance the domain knowledge

provided by a Global Schema. To the best of our knowledge this approach is completely new in the literature. I described an original approach to connect data descriptions and semantic web service descriptions. Starting from a data query expressed over a Global Schema, the system presented first extracts the useful terms from the data queries, then identifies the terms describing the services that are related to those extracted from the query to be used in an information retrieval service discovery approach. Experimental results show that this approach ensure a high level of recall (around 100%) and an acceptable level of precision. A fundamental paradigm shift in the service discovery process is determined by this approach, mostly in terms of objectives: we are interested in finding a set of services "related" to a set of terms extracted from a query formulated over a data schema. The retrieved services might be relevant to the set of keywords w.r.t. their inputs, outputs or descriptions.

Finally, in Chapter 6 I described my experience in realizing an integration flow of JSON data exploiting the Jaql query language on the Hadoop framework. The integration concerned government data in order to enable exploring U.S. government spending data. This work was conducted at the IBM Almaden Research Center of San José, California, during my six months period stay as a research intern. This work is an example of an integration flow in a distributed environment that shows two interesting results from the perspective of data integration application on distributed architectures: I designed and developed an integration flow that can be seen as an application directly implemented on the cloud, and the results of this integration flow were made available from the could by means of an API in a Data as a Service approach. The Hadoop framework is an open source project providing an environment with a distributed file system that easily allows realizing a cloud for data storage and the execution of distributed applications on commodity hardware. Jaql is a powerful yet simple data manipulation language that transparently runs on Hadoop and easily allows numerous operations on data in JSON format. The integration flow presented could thus be generalized to provide an architecture for data integration on the cloud: Jaql can be used as query language, and the UDFs developed in this work can potentially be made available as a data cleansing and normalization library to be used with Jaql.

Appendix A

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

The following is the BNF description of the ODL$_{I^3}$ description language. This object-oriented language, with an underlying Description Logic, is introduced for information extraction. The ODL$_{I^3}$ language is presented in [Bergamaschi et al., 2001], in the following I include the syntax fragments which differ from the original ODL grammar, referring to it 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** | **multimedia**
| **glso**
⟨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**
...

Extended base type definition for multimedia objects.

⟨BaseTypeSpec⟩ ::= ⟨FloatingPtType⟩ |
⟨IntegerType⟩ |
⟨CharType⟩ |
⟨BooleanType⟩ |
⟨OctetType⟩ |
⟨RangeType⟩ |
⟨AnyType⟩ |
⟨MultiMediaType⟩

⟨MultiMediaType⟩ ::= **Text** |
Image

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**

Bibliography

[S3, 2009] (2009). Amazon simple storage service. `http://aws.amazon.com/s3/`. Last visited 17 January 2010.

[rep, 2009] (2009). Annual international contest s3 on semantic service selection retrieval performance evaluation of matchmakers for semantic web services. `http://www-ags.dfki.uni-sb.de/~klusch/s3/html/2009.html`. Last visited 17 January 2010.

[Had, 2009] (2009). Apache hadoop project. http://hadoop.apache.org/.

[AST, 2009] (2009). Asterix: A highly scalable parallel platform for semistructured data management and analysis. `http://asterix.ics.uci.edu/about.xml`.

[DBL, 2009] (2009). Data integration. In Liu, L. and Özsu, M. T., editors, *Encyclopedia of Database Systems*, page 585. Springer US.

[fla, 2009] (2009). The flamingo project on data cleaning. `http://flamingo.ics.uci.edu/`.

[Akkiraju et al., 2005] Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., and Verma, K. (2005). Web service semantics - wsdl-s. W3C Member Submission.

[Amato et al., 2004] Amato, G., Gennaro, C., Savino, P., and Rabitti, F. (2004). Milos: a Multimedia Content Management System for Digital Library Applications. In *Proceedings of the 8th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2004)*, volume 3232 of *Lecture Notes in Computer Science*, pages 14–25. Springer.

[Antofie et al., 2007] Antofie, A., Lateur, M., Oger, R., Patocchi, A., Durel, C. E., and Van de Weg, W. E. (2007). A new versatile database created for geneticists and breeders to link molecular and phenotypic data in perennial crops. *Bioinformatics*, 23(7):882–891.

[Benassi et al., 2004] Benassi, R., Bergamaschi, S., Fergnani, A., and Miselli, D. (2004). Extending a lexicon ontology for intelligent information integration. In de Mántaras, R. L. and Saitta, L., editors, *ECAI*, pages 278–282. IOS Press.

[Beneventano and Bergamaschi, 2007] Beneventano, D. and Bergamaschi, S. (2007). *Semantic Web Services: Theory, Tools and Applications*, chapter Semantic Search Engines based on Data Integration Systems. Idea Group Publishing.

[Beneventano et al., 2008a] Beneventano, D., Bergamaschi, S., Gennaro, C., Guerra, F., Mordacchini, M., and Sala, A. (2008a). A mediator system for data and multimedia sources. In *Data Integration through Semantic Technology, Workshop at the 3rd Asian Semantic Web Conference, Bangkok, Thailand*.

[Beneventano et al., 2003] Beneventano, D., Bergamaschi, S., and Sartori, C. (2003). Description logics for semantic query optimization in object-oriented database systems. *ACM Trans. Database Syst.*, 28:1–50.

[Beneventano et al., 2007] Beneventano, D., Bergamaschi, S., Vincini, M., Orsini, M., and Nana Mbinkeu, R. C. (2007). Getting through the thalia benchmark with momis. In *Proceedings of the Third International Workshop on Database Interoperability (InterDB 2007) held in conjunction with the 33rd International Conference on Very Large Data Bases, VLDB 2007, Vienna, Austria, September 24*.

[Beneventano et al., 2009] Beneventano, D., Guerra, F., Maurino, A., Palmonari, M., Pasi, G., and Sala, A. (2009). Unified Semantic Search of Data and Services. In F. Sartori, M. Á. Sicilia, & N. Manouselis, editor, *Metadata and Semantic Research, Communications in Computer and Information Science, Volume 46. ISBN 978-3-642-04589-9. Springer Berlin Heidelberg, 2009, p. 95*, pages 95–+.

[Beneventano et al., 2008b] Beneventano, D., Guerra, F., Orsini, M., Po, L., Sala, A., Gioia, M. D., Comerio, M., de Paoli, F., Maurino, A., Palmonari, M., Gennaro, C., Sebastiani, F., Turati, A., Cerizza, D., Celino, I., and Corcoglioniti, F. (2008b). Detailed design for building semantic peer. *Networked Peers for Business, Deliverable D.2.1, Final Version, available at http : //www.dbgroup.unimo.it/publication/d2_1.pdf*, pages 52–57.

[Beneventano and Lenzerini, 2005] Beneventano, D. and Lenzerini, M. (2005). Final release of the system prototype for

query management. sewasie, deliverable d.3.5, final version. http://www.dbgroup.unimo.it/prototipo/paper/D3.5Final.pdf.

[Bennett et al., 2000] Bennett, K., Layzell, P., Budgen, D., Brereton, P., Macaulay, L., and Munro, M. (2000). Service-based software: the future for flexible software. *Asia-Pacific Software Engineering Conference*, 0:214.

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

[Bergamaschi and Maurino, 2009] Bergamaschi, S. and Maurino, A. (2009). Toward a unified view of data and services. In Vossen, G., Long, D. D. E., and Yu, J. X., editors, *WISE*, volume 5802 of *Lecture Notes in Computer Science*, pages 11–12. Springer.

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

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

[Bergamaschi and Sala, 2006] Bergamaschi, S. and Sala, A. (2006). Virtual integration of existing web databases for the genotypic selection of cereal cultivars. In Meersman, R. and Tari, Z., editors, *OTM Conferences (1)*, volume 4275 of *Lecture Notes in Computer Science*, pages 909–926. Springer.

[Bergamaschi and Sala, 2007] Bergamaschi, S. and Sala, A. (2007). Creating and querying an integrated ontology for molecular and phenotypic cereals data. In Sicilia, M.-Á. and Lytras, M. D., editors, *MTSR*, pages 445–454. Springer.

[Bergamaschi et al., 1997] Bergamaschi, S., Sartori, C., Beneventano, D., and Vincini, M. (1997). Odb-tools: A description logics based tool for schema validation and semantic query optimization in object oriented databases. In Lenzerini, M., editor, *AI*IA*, volume 1321 of *Lecture Notes in Computer Science*, pages 435–438. Springer.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific American*.

[Bernstein et al., 2005] Bernstein, A., Kaufmann, E., Buerki, C., and Klein, M. (2005). How similar is it? towards personalized similarity measures in ontologies. In *Wirtschaftsinformatik*, pages 1347–1366.

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

[Bleiholder and Naumann, 2008] Bleiholder, J. and Naumann, F. (2008). Data fusion. *ACM Comput. Surv.*, 41(1).

[Brambilla and Ceri, 2009] Brambilla, M. and Ceri, S. (2009). Engineering search computing applications: vision and challenges. In *ESEC/SIGSOFT FSE*, pages 365–372.

[Burstein et al., 2004] Burstein, M., Hobbs, J., Lassila, O., Mcdermott, D., Mcilraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., and Sycara, K. (2004). OWL-S: Semantic Markup for Web Services. Website.

[Buyya, 2009] Buyya, R. (2009). Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 1, Washington, DC, USA. IEEE Computer Society.

[Cali et al., 2002] Cali, A., Calvanese, D., Giacomo, G. D., and Lenzerini, M. (2002). Data integration under integrity constraints. In *Information Systems*, pages 262–279. Springer.

[Calvanese et al., 2004] Calvanese, D., Giacomo, G. D., Lenzerini, M., and Rosati, R. (2004). Logical foundations of peer-to-peer data integration. In Deutsch, A., editor, *PODS*, pages 241–251. ACM.

[Carey et al., 1995] Carey, M. J., Haas, L. M., Schwarz, P. M., Arya, M., Cody, W. F., Fagin, R., Thomas, J., H, J., and Wimmers, E. L. (1995). Towards heterogeneous multimedia information systems: The garlic approach. In *In RIDE-DOM*, pages 124–131.

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

[Ceri, 2009] Ceri, S. (2009). Search computing. In *ICDE*, pages 1–3. IEEE.

[Chang et al., 2006] Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2006). Bigtable: a distributed storage system for structured data. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 205–218, Berkeley, CA, USA. USENIX Association.

[Chang and Garcia-Molina, 1999] Chang, K. C.-C. and Garcia-Molina, H. (1999). Mind your vocabulary: Query mapping across heterogeneous information sources. In *SIGMOD Conference*, pages 335–346.

[Chappell, 2009] Chappell, D. (2009). Introducing windows azure, white paper. http://go.microsoft.com/?linkid=9682907.

[Christopher D. Manning and Schütze, 2008] Christopher D. Manning, P. R. and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P. D., and Fienberg, S. E. (2003). A comparison of string distance metrics for name-matching tasks. In Kambhampati, S. and Knoblock, C. A., editors, *IIWeb*, pages 73–78.

[Dalvi et al., 2009] Dalvi, N. N., Kumar, R., Pang, B., Ramakrishnan, R., Tomkins, A., Bohannon, P., Keerthi, S., and Merugu, S. (2009). A web of concepts. In *PODS*, pages 1–12.

[Dan et al., 2007] Dan, A., Johnson, R., and Arsanjani, A. (2007). Information as a service: Modeling and realization. In *SDSOA '07: Proceedings of the International Workshop on Systems Development in SOA Environments*, page 2, Washington, DC, USA. IEEE Computer Society.

[d'Aquin et al., 2007] d'Aquin, M., Doran, P., Motta, E., and Tamma, V. A. M. (2007). Towards a parametric ontology modularization framework based on graph transformation. In *WoMO*.

[Davidson et al., 2001] Davidson, S. B., Crabtree, J., Brunk, B. P., Schug, J., Tannen, V., Overton, G. C., and Jr., C. J. S. (2001). K2/kleisli and gus: Experiments in integrated access to genomic data sources. *IBM Systems Journal*, 40(2):512–531.

[Davidson et al., 1997] Davidson, S. B., Overton, G. C., Tannen, V., and Wong, L. (1997). Biokleisli: A digital library for biomedical researchers. *Int. J. on Digital Libraries*, 1(1):36–53.

[de Bruijn et al., 2005] de Bruijn, J., Lausen, H., Krummenacher, R., Polleres, A., Predoiu, L., Kifer, M., and Fensel, D. (October 2005). D16.1v0.21 The Web Service Modeling Language WSML. *WSMO Working Group*.

[Dean and Ghemawat, 2010] Dean, J. and Ghemawat, S. (2010). System and method for efficient large-scale data processing. US Patent 7,650,331.

[Doran et al., 2007] Doran, P., Tamma, V., and Iannone, L. (2007). Ontology module extraction for ontology reuse: an ontology engineering perspective. In *CIKM '07: Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 61–70, New York, NY, USA. ACM.

[Dustdar and Schreiner, 2005] Dustdar, S. and Schreiner, W. (2005). A survey on web services composition. *Int. J. Web Grid Serv.*, 1(1):1–30.

[Exner et al., 2008] Exner, V., Hirsch-Hoffmann, M., Gruissem, and Wilhelm; Hennig, L. (2008). Plantdb - a versatile database for managing plant research. *Plant Meth.*

[Fagin, 1998] Fagin, R. (1998). Fuzzy queries in multimedia database systems. In *PODS '98: Proceedings of the seventeenth symposium on Principles of database systems*, pages 1–10, New York, NY, USA. ACM.

[Fagin et al., 2009] Fagin, R., Haas, L. M., Hernández, M. A., Miller, R. J., Popa, L., and Velegrakis, Y. (2009). Clio: Schema mapping creation and data exchange. In *Conceptual Modeling: Foundations and Applications*, pages 198–236.

[Fagin et al., 2005] Fagin, R., Kolaitis, P. G., Miller, R. J., and Popa, L. (2005). Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124.

[Fagin et al., 2004] Fagin, R., Kumar, R., Mahdian, M., Sivakumar, D., and Vee, E. (2004). Comparing and aggregating rankings with ties. In *PODS '04: Proceedings of the twenty-third symposium on Principles of database systems*, pages 47–58, New York, NY, USA. ACM.

[Fellbaum, 1998] Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

[Friedman et al., 1999] Friedman, M., Levy, A., and Millstein, T. (1999). Navigational plans for data integration. In *In Proceedings of the National Conference on Artificial Intelligence (AAAI*, pages 67–73. AAAI Press/The MIT Press.

[Grau et al., 2007] Grau, B. C., Horrocks, I., Kazakov, Y., and Sattler, U. (2007). Just the right amount: Extracting modules from ontologies. In *Proceedings of WWW-2007: the 16th International World Wide Web Conference, Banff, Alberta, Canada, May 8–12, 2007.*

[Guarino, 1995] Guarino, N. (1995). Formal ontology, conceptual analysis and knowledge representation. *Int. J. Hum.-Comput. Stud.*, 43(5-6):625–640.

[Guerra et al., 2009] Guerra, F., Maurino, A., Palmonari, M., Pasi, G., and Sala, A. (2009). Searching for data and services. In *1st International Workshop on Interoperability through Semantic Data and Service Integration (ISDSI 2009), Camogli (Genova), Italy.*

[Haas, 2007] Haas, L. M. (2007). Beauty and the beast: The theory and practice of information integration. In Schwentick, T. and Suciu, D., editors, *ICDT*, volume 4353 of *Lecture Notes in Computer Science*, pages 28–43. Springer.

[Haas et al., 2001] Haas, L. M., Schwarz, P. M., Kodali, P., Kotlar, E., Rice, J. E., and Swope, W. C. (2001). Discoverylink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(2):489–511.

[Halevy, 2001] Halevy, A. Y. (2001). Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294.

[Halevy et al., 2003] Halevy, A. Y., Ives, Z. G., Suciu, D., and Tatarinov, I. (2003). Schema mediation in peer data management systems. In Dayal, U., Ramamritham, K., and Vijayaraman, T. M., editors, *ICDE*, pages 505–. IEEE Computer Society.

[Halevy et al., 2006] Halevy, A. Y., Rajaraman, A., and Ordille, J. J. (2006). Data integration: The teenage years. In Dayal, U., Whang, K.-Y., Lomet, D. B., Alonso, G., Lohman, G. M., Kersten, M. L., Cha, S. K., and Kim, Y.-K., editors, *VLDB*, pages 9–16. ACM.

[Hansen et al., 2003] Hansen, M., Madnick, S. E., and Siegel, M. (2003). Data integration using web services. In *Proceedings of the VLDB 2002 Workshop EEXTT and CAiSE 2002 Workshop DTWeb on Efficiency and Effectiveness of XML Tools and Techniques and Data Integration over the Web-Revised Papers*, pages 165–182, London, UK. Springer-Verlag.

[Hatcher and Gospodnetic, 2004] Hatcher, E. and Gospodnetic, O. (2004). *Lucene in Action*. Manning Publications.

[Heimbigner and McLeod, 1985] Heimbigner, D. and McLeod, D. (1985). A federated architecture for information management. *ACM Trans. Inf. Syst.*, 3(3):253–278.

[Hernandez and Kambhampati, 2004] Hernandez, T. and Kambhampati, S. (2004). Integration of biological sources: Current systems and challenges ahead. *SIGMOD Record*, 33(3):51–60.

[Holger et al., 2003] Holger, R. L., Lausen, H., Arroyo, S., Bruijn, J., Fensel, D., and Innsbruck, U. (2003). Semantic web services: description requirements and current technologies.

[Inmon, 2002] Inmon, W. H. (2002). *Building the Data Warehouse,3rd Edition*. John Wiley & Sons, Inc., New York, NY, USA.

[Jaiswal et al., 2006] Jaiswal, P., Ni, J., Yap, I., Ware, D., Spooner, W., Youens-Clark, K., Ren, L., Liang, C., Zhao, W., Ratnapu, K., Faga, B., Canaran, P., Fogleman, M., Hebbard, C., Avraham, S., Schmidt, S., Casstevens, T. M., Buckler, E. S., Stein, L., and McCouch, S. (2006). Gramene: a bird's eye view of cereal genomes. *Nucleic Acids Research*, 34(Database-Issue):717–723.

[Jiménez-Ruiz et al., 2008] Jiménez-Ruiz, E., Grau, B. C., Sattler, U., Schneider, T., and Llavori, R. B. (2008). Safe and economic re-use of ontologies: A logic-based methodology and tool support. In *ESWC*, pages 185–199.

[Keller et al., 2004] Keller, U., Lara, R., Polleres, A., Toma, I., Kifer, M., and Fensel, D. (2004). D5.1 v0.1 WSMO Web Service Discovery.

[Kelly and Miklas, 1999] Kelly, J. D. and Miklas, P. N. (1999). *Common Bean Improvement in the Twenty-first Century (Developments in Plant Breeding)*, chapter Marker Assisted Selection. Springer.

[Kiefer and Bernstein, 2008] Kiefer, C. and Bernstein, A. (2008). The creation and evaluation of isparql strategies for matchmaking. In *ESWC*, pages 463–477.

[Kifer et al., 1995] Kifer, M., Lausen, G., and Wu, J. (1995). Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843.

[Klusch et al., 2009] Klusch, M., Fries, B., and Sycara, K. P. (2009). Owls-mx: A hybrid semantic web service matchmaker for owl-s services. *J. Web Sem.*, 7(2):121–133.

[Klusch and Kapahnke, 2009] Klusch, M. and Kapahnke, P. (2009). Owls-mx3: An adaptive hybrid semantic service matchmaker for owl-s. In *Proceedings of the Third International Workshop on Service Matchmaking and Resource Retrieval in the Semantic Web, at the 8th International Semantic Web Conference Washington D.C., USA*.

[Kolaitis, 2005] Kolaitis, P. G. (2005). Schema mappings, data exchange, and metadata management. In *PODS '05: Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 61–75, New York, NY, USA. ACM.

[Lawrence et al., 2004] Lawrence, C. J., Dong, Q., Polacco, M. L., Seigfried, T. E., and Brendel, V. (2004). Maizegdb, the community database for maize genetics and genomics. *Nucleic Acids Research*, 32(Database-Issue):393–397.

[Lee et al., 2005] Lee, J. M., Davenport, G. F., Marshall, D., Ellis, T. N., Ambrose, M. J., Dicks, J., van Hintum, T. J., and Flavell, A. J. (2005). GERMINATE. A Generic Database for Integrating Genotypic and Phenotypic Information for Plant Genetic Resource Collections. *Plant Physiol.*, 139(2):619–631.

[Lenzerini, 2002] Lenzerini, M. (2002). Data integration: A theoretical perspective. In Popa, L., editor, *PODS*, pages 233–246. ACM.

[Levy, 1998] Levy, A. Y. (1998). The information manifold approach to data integration. *IEEE Intelligent Systems*, 13:12–16.

[Li et al., 1998] Li, C., Yerneni, R., Vassalos, V., Garcia-Molina, H., Papakonstantinou, Y., Ullman, J. D., and Valiveti, M. (1998). Capability based mediation in tsimmis. In *SIGMOD Conference*, pages 564–566.

[Matthews et al., 2003] Matthews, D. E., Carollo, V. L., Lazo, G. R., and Anderson, O. D. (2003). Graingenes, the genome database for small-grain crops. *Nucleic Acids Research*, 31(1):183–186.

[McIlraith et al., 2001] McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. *IEEE Intelligent Systems*, 16:46–53.

[Menestrina et al., 2006] Menestrina, D., Benjelloun, O., and Garcia-Molina, H. (2006). Generic entity resolution with data confidences. In *CleanDB*.

[Murdock and Lalmas, 2008] Murdock, V. and Lalmas, M. (2008). Workshop on aggregated search. *SIGIR Forum*, 42(2):80–83.

[Naumann et al., 2004] Naumann, F., Freytag, J. C., and Leser, U. (2004). Completeness of integrated information sources. *Inf. Syst.*, 29(7):583–615.

[Naumann and Häussler, 2002] Naumann, F. and Häussler, M. (2002). Declarative data merging with conflict resolution. In Fisher, C. and Davidson, B. N., editors, *IQ*, pages 212–224. MIT.

[Noy, 2004] Noy, N. F. (2004). Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70.

[Noy and Musen, 2004] Noy, N. F. and Musen, M. A. (2004). Specifying ontology views by traversal. In McIlraith, S. A., Plexousakis, D., and van Harmelen, F., editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 713–725. Springer.

[Orsini, 2004] Orsini, M. (2003/2004). Interoperabilitá tra ontologie eterogenee: i traduttori OWL - ODLI3. Master's thesis, University of Modena and Reggio Emilia.

[Orsini, 2009] Orsini, M. (2009). *Query Management in Data Integration Systems: the MOMIS approach.* PhD thesis, International Doctorate School in Information and Communication Technologies of the University of Modena and Reggio Emilia.

[Palmisano et al., 2009] Palmisano, I., Tamma, V., Payne, T., and Doran, P. (2009). Task oriented evaluation of module extraction techniques. In *8th International Semantic Web Conference (ISWC2009)*, volume 5823 of *LNCS*, pages 130–145. Springer.

[Palmonari et al., 2007] Palmonari, M., Guerra, F., Turati, A., Maurino, A., Beneventano, D., Valle, E. D., Sala, A., and Cerizza, D. (2007). Toward

a unified view of data and services. In *Proceedings of the 1st International International Workshop on Semantic Data and Service Integration, Vienna, Austria*.

[Palmonari et al., 2010] Palmonari, M., Sala, A., Maurino, A., Guerra, F., Pasi, G., and Frisoni, G. (2010). Aggregated search of data and services. *Inf. Syst.* Submitted.

[Peng et al., 2008] Peng, Z.-Y., Zhou, X., Li, L., Yu, X., Li, H., Jiang, Z., Cao, G., Bai, M., Wang, X., Jiang, C., Lu, H., Hou, X., Qu, L., Wang, Z., Zuo, J., Fu, X., Su, Z., Li, S., and Guo, H. (2008). Arabidopsis hormone database: a comprehensive genetic and phenotypic information database for plant hormone research in arabidopsis. *Nucl. Acids Res.*, pages gkn873+.

[Po, 2009] Po, L. (2009). *Automatic Lexical Annotation: an effective technique for dynamic data integration*. PhD thesis, International Doctorate School in Information and Communication Technologies of the University of Modena and Reggio Emilia.

[Pottinger and Bernstein, 2003] Pottinger, R. and Bernstein, P. A. (2003). Merging models based on given correspondences. In *VLDB*, pages 826–873.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350.

[Rao and Su, 2005] Rao, J. and Su, X. (2005). A survey of automated web service composition methods. pages 43–54.

[Richardson and Ruby, 2007] Richardson, L. and Ruby, S. (2007). *Restful web services*. O'Reilly.

[Roman et al., 2004] Roman, D., Lausen, H., , and Keller, U. (2004). Web Service Modeling Ontology (WSMO).

[Romano, 2007] Romano, F. (2007). Analisi e valutazione comparativa dei principali sistemi di integrazione dati commerciali rispetto al sistema momis attraverso il benchmark thalia.

[Sala and Bergamaschi, 2009] Sala, A. and Bergamaschi, S. (2009). A mediator-based approach to ontology generation and querying of molecular and phenotypic cereals data. *IJMSO*, 4(1/2):85–92.

[Sala et al., 2010] Sala, A., Lin, C., and Ho, H. (2010). Midas for Government: Integration of Government Spending Data on Hadoop. In *IEEE ICDE International Workshop on New Trends in Information Integration (NTII'10)*, Long Beach, CA.

[Seidenberg and Rector, 2006] Seidenberg, J. and Rector, A. L. (2006). Web ontology segmentation: analysis, classification and use. In *WWW*, pages 13–22.

[Sheth and Larson, 1990] Sheth, A. P. and Larson, J. A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Comput. Surv.*, 22(3):183–236.

[Srivastava and Koehler, 2003] Srivastava, B. and Koehler, J. (2003). Web service composition - current solutions and open problems. *Proceedings of ICAPS 2003*.

[Stevens et al., 2000] Stevens, R., Baker, P. G., Bechhofer, S., Ng, G., Jacoby, A., Paton, N. W., Goble, C. A., and Brass, A. (2000). Tambis: Transparent access to multiple bioinformatics information sources. *Bioinformatics*, 16(2):184–186.

[Toch et al., 2007] Toch, E., Gal, A., Reinhartz-Berger, I., and Dori, D. (2007). A semantic approach to approximate service retrieval. *ACM Trans. Internet Techn.*, 8(1).

[Truong and Dustdar, 2009] Truong, H.-L. and Dustdar, S. (2009). On analyzing and specifying concerns for data as a service. In *In Proc of IEEE Asian-Pacific Service Computing Conference*.

[Ullman, 1997] Ullman, J. D. (1997). Information integration using logical views. In Afrati, F. N. and Kolaitis, P. G., editors, *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer.

[Varia, 2009] Varia, J. (2009). Cloud architectures white paper, amazon. http://jineshvaria.s3.amazonaws.com/public/cloudarchitectures-varia.pdf.

[Wang et al., 2008] Wang, Z., Wang, K., Topor, R. W., and Pan, J. Z. (2008). Forgetting concepts in dl-lite. In *ESWC*, pages 245–257.

[Weiss, 2007] Weiss, A. (2007). Computing in the clouds. *netWorker*, 11(4):16–25.

[Wiederhold, 1992] Wiederhold, G. (1992). Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49.

[Zhao et al., 2006] Zhao, W., Canaran, P., Jurkuta, R., Fulton, T., Glaubitz, J., Buckler, E. S., Doebley, J., Gaut, B., Goodman, M., Holland, J., Kresovich, S., McMullen, M. D., Stein, L., and Ware, D. (2006). Panzea: a database and resource for molecular and functional diversity in the maize genome. *Nucleic Acids Research*, 34(Database-Issue):752–757.

[Zhu et al., 2004] Zhu, F., Turner, M., Kotsiopoulos, I., Bennett, K., Russell, M., Budgen, D., Brereton, P., Keane, J., Layzell, P., Rigby, M., and Xu, J. (2004). Dynamic data integration using web services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 262, Washington, DC, USA. IEEE Computer Society.