A Semantic Framework for Web-based Accommodation Information Integration

Kai Yang

University of Technology, Sydney

A thesis submitted for the degree of

Doctor of Philosophy

2012

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Student

____

Abstract

With the tremendous growth of the Web, a broad spectrum of accommodation information is to be found on the Internet. In order to adequately support information users in collecting and sharing information online, it is important to create an effective information integration solution, and to provide integrated access to the vast numbers of online information sources. In addition to the problem of distributed information sources, information users also need to cope with the heterogeneous nature of the online information sources, where individual information sources are stored and presented following their own structures and formats. In this thesis, we explore some of the challenges in the field of information integration, and propose solutions to some of the arising challenges. We focus on the utilization of ontology for integrating heterogeneous, structured and semi-structured information sources, where instance level data are stored and presented according to meta-data level schemas. In particular, we looked at XML-based data that is stored according to XML schemas.

In a first step towards a large-scale information integration solution, we propose a semantic integration framework. The proposed framework solves the problem of information integration on three levels: the data level, process level and architecture level. On the data level, we leverage the benefit of ontology, and use ontology as a mediator for enabling semantic interoperability among heterogeneous data sources. On the process level, we alter the process of information integration, and propose a three step integration process named as the publish-combine-use mechanism. The primary goal is to distribute the efforts of collecting and integrating information sources to various types of end users. In the proposed approach, information providers have more control over their own data sources, as data sources are able to join and leave the information sharing network according to their own preferences. On the architecture level, we combine the flexibility offered by the emerging distributed P2P approach with the query processing capability provided by the centralized approach. The joint architecture is similar to the structure of the online accommodation industry.

This thesis also demonstrates the practical applicability of the proposed semantic integration framework by implementing a prototype system. The prototype system named the “accommodation hub” is specifically developed for integrating online accommodation information in the large, distributed, heterogeneous online environment. The proposed semantic integration solution and the implemented prototype system are evaluated to provide a measure of the system performance and usage. Results show that the proposed solution delivers better performance with respect to some of the evaluation criteria than some related approaches in information integration.

Acknowledgements

This thesis is fully supported by the ARC Linkage Project, a joint effort between the University of Technology, Sydney and the Lido Group. Here, I would like to acknowledge that this research would not have been possible without the support from both the university and the industry partner.

First of all, I want to thank my supervisor, Prof. Robert Steele, who has brought me to this exciting research project, and guided me throughout my candidature. I would like to express my gratitude to him for providing me with the inspirational ideas and comments for this research.

I also wish to thank all my colleagues from the university of technology, Sydney for the invaluable inputs to this research and the enjoyable teamwork. In particular, I would like to express my gratitude to Amanda Lo for the productive collaboration and the delightful company throughout this research. Thanks to John Murphy and Rita Shen for the valuable discussions had within this research. I am also grateful to our industry partner, more specifically Matthew Tyler and John Taranto for all their advices and feedbacks on the prototype system.

Finally, my gratitude to my family and friends who made this work possible with their support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Ever since the advent of the Internet, the adoption of Web technologies gives businesses the ability to make their own information available online. Through this way, businesses are opened to a bigger market and are able to reach customers globally regardless of their geographical or physical location. According to AC Nielsen's online consumer press release, by September 2006 there are over 6 million online customers in Australia alone, and such number is growing at a constant pace (ACNielsen, 2006). On the other hand, the World Wide Web has revolutionised the way people access and share information, and it provides a simple and effective means for users to search, browse and retrieve information available online. While the Internet has brought numerous opportunities to businesses and offers easy solutions for personal information access, it has also created a number of technical challenges such as information retrieval and integration. Due to the vast amount of information available on the Web, how to identify and retrieve the right piece of information becomes a real challenge. In addition, since each information source is modelled and described using its own structure and vocabularies, how to solve data heterogeneity and enable integrated information access becomes another major challenge.

For the online accommodation industry, the situation is similar yet more challenging. From a technical point of view, Web-based technologies offer businesses the opportunities to improve its operation efficiency while reducing its operation cost. Many operational tasks such as communication, collaboration, distribution and customer service have been moved online (Werthner & Klein, 1999). A number of independently operated electronic accommodation solutions have been developed, which include Web sites, Web-based applications and booking portals. Since each system often uses its own data model and structure for data representation and storage, data heterogeneity becomes a main issue when exchanging data amongst different information systems. From a business point of view, the online accommodation industry involves a wide range of players including the suppliers (e.g. small and large hotels), the retailers (e.g. hotel Websites, online booking portals), the wholesalers (e.g. travel agents, distribution channels, corporate booking tools), and the consumers (e.g. corporate customers such as the government). In some cases, players with different roles need to collaborate with each other to achieve a common business goal. For instance, in order to fulfil a corporate booking order, travel agents need to collaborate with multiple service providers for the arrangement of travel facilities, and this can not be easily achieved without the exchange of information. In those cases, information (e.g. inventory and order information) needs to be shared and exchanged among different business parties. Again, how to enable seamless information sharing becomes a key success factor.

While enabling information sharing among online service providers is important for facilitating business collaborations, it is also important for online consumers to assist their decision making. Information acquisition and processing are crucial steps of decision making (Ariely, 2000). For the consumers, the scattered availability of information and the heterogeneous nature of data on the Web let decision making a difficult task. In order to select the best service available according to their own preferences, online consumers often need to consult a broad range of information scattered among heterogeneous online sources. The process of information gathering is often tedious and time consuming. For example, if a traveller wishes to take a holiday in Sydney, and he plans to stay in a hotel with a fabulous harbour view and good entertainment facility. Then he needs to consider a number of factors including the location of the hotel, its entertainment facilities and the price. If the traveller does not want to spend too much time on travel, he may also need information such as bus or train time tables, the location of famous tourism spots, or even the weather report to ensure a pleasant trip. The decision making process is supported by a number of repeated information acquisition tasks. Therefore, to better serve the information needs of online consumers, a more effective and efficient solution needs to be developed to provide integrated access to the heterogeneous online information sources.

In addition, the development of Web 2.0 applications such as the trip advisors and hotel review sites empower consumers to share reviews and personal experiences on the Web. A large number of online communities are formed with specialities in different travel domains. Since each community group provides information for a particular domain, the integration of scattered information becomes another challenge, and the integration of such information can increase the service and price transparency and improve service quality.

Due to the above-mentioned reasons, a significant amount of effort has been devoted to create techniques and tools for automating the integration of online data sources. However, the majority of the developed solutions are still manual, requiring complex programmatic set-up, and have limited code reusability. Hence, they do not scale well to cope with large-scale information integration. On the other hand, the online accommodation industry has a number of unique characteristics, and requires an efficient and flexible solution for enabling information sharing across hundreds of heterogeneous data sources. In this thesis, we specifically focus on the online accommodation domain, and our research is based on four characteristics of the online accommodation industry, which are its large, heterogeneous, dynamic and highly distributed nature.

1.1 Nature of the online accommodation industry

Large

Tourism is an information intensive industry (Inkpen, 1998), and the nature of the Internet makes it an ideal platform for the tourism industry to broadcast service information to consumers all around the world. Indeed the fact that among all industries surging online, the travel industry is by far leading other service industries in its share of e-commerce (Dinlersoz & Hernändez-Murillo, 2005), meaning that the industry recognises the potentials and benefits that the Internet brings. Its dynamic nature and consumer's demand for the ability to search and acquire information on travel arrangements makes it very suitable for e-commerce. Some researchers have claimed that the travel industry has the potential to become the biggest industry with the majority of its sales online (Dinlersoz & Hernändez-Murillo, 2005). Another figure published by ABS yearbook 2007 (ABS, 2007) reveals that in the year 2004-05, 77% of australian accommodation providers have some sort of a computerised system that takes care of all or part of their daily business processes, while 31% of the providers have a Web presence, and these numbers are growing at a constant pace.

In addition, the merging of accommodation and Internet developments lead to the emergence of a range of new players, which provide new tools, information and services for both consumers and industry. Typical examples include global distribution systems (e.g. Galileo System), online booking portals (e.g. wotif.com[1]), trip advisors (e.g. tripadvisor.com[2]), hotel review sites (e.g. virtual-reviews.com.au[3]), travel forums, and online travel communities. The number of emerging players is also increasing rapidly. All these figures show that the online accommodation industry is a large information repository with a great number of data sources.

Heterogeneous

Another important characteristic of the online accommodation domain is data heterogeneity, and this is inherited from the nature of the World Wide Web. The recent initiatives to create a Semantic Web (Antoniou & Harmelen, 2004) also emphasized the issues of data heterogeneity and the lack of semantic interoperability. The World Wide Web is currently dominated by a vast amount of structured and semi-structured content, and the amount of structured data on the Web and the diversity of the structures in which these data are stored are rising. The same situation appears in the online accommodation domain where data are annotated and stored in diverse structures and syntax. Despite the efforts in developing schema standards for faciliating seamless data exchange, some of the developed standards are still hard to impose, or failed to reach wide acceptance. As a result, the issue of data heterogeneity remains as a major problem in the online accommodation domain.

The problem of data heterogeneity can be classified into three categories: syntactic conflict, semantic conflict and schematic conflict. Syntactic conflict occurs when the same data is written using different languages and representations. Al-

[1] See http://www.wotif.com
[2] See http://www.tripadvisor.com
[3] See http://www.virtualreviews.com.au

though XML is currently the most widely used and well adopted standard for data exchange on the Web (Abiteboul *et al.*, 1999), there are still many other widely used mark-up languages and formats such as SGML (Bryan, 1988). Even with the same mark-up language, syntactic problems occur if the same data or concept is modelled using different structure or logic. On the semantic level, one concept may be defined using different semantic terms or vocabularies. For example, the concept of hotel can be defined as “Hotel” in one data model, but is defined as “Accommodation” in another data model. In some cases, the same term is used to carry different concepts. For instance, the term “Name” refers to guest name in one data model, but is used to carry hotel name in another data model. Although both names use the term “Name”, they refer to totally different things. Other forms of semantic heterogeneity also exist such as the granularity difference. Schematic heterogeneity appears when different schemas are defined to model a same domain. We refer you to Chapter 4 for complete explaination on various types of data heterogeneity.

Dynamic

Due to the rapid development of the Internet, a large number of new players have emerged, which provide new products and services to both consumers and industry, and the amount of hotel information available on the Web is also on the rise (Paraskevas, 2005). On one hand, the Internet has reduced the cost for new players to enter the online accommodation market. Both small and medium sized businesses now have equal opportunities to compete with large corporate hotel chains. On the other hand, the Internet has increased consumer’s bargaining and switching power by providing better and richer information, and the online consumer group nowadays is better informed and more price aware. Although this is a great advantage for the consumers, it is however a major challenge for most online businesses as new businesses can easily enter the market, and the competition within the market is rapidly increasing (Gratzer & Winiwarter, 2003). As a result, information sources come and go quickly as businesses enter and leave the market, and the availability of the online information sources becomes more dynamic and hard to predict. Hence, an information integration solution is required.

Distributed

The distributed nature of the online accommodation domain can be reflected in two aspects: organizational and geographical. Figure 1.1 shows the organizational structure of the online accommodation industry, and the organizational hierarchy consists of four layers including the providers, the franchisers, the wholesalers and the intermediaries. Each layer is linked to multiple entities from its sub-layer. For example, a hotel franchiser often collaborates with multiple hotel providers, and a wholesaler is often linked to multiple franchisers and hoteliers. Therefore, information integration occurs when data travels from the lower layer to the upper layer. For example, a hotel intermediary often provides integrated access to information sources that are maintained by its sub-layers, including the wholesalers, the franchisers, and the providers. In this thesis, we refer this nature as organizationally distributed, and each organizational entity itself is an independent information source.

In addition, the online accommodation industry is filled with vast numbers of geographically distributed information systems, and it is opened to businesses all around the world regardless of their geographical or physical locations. Businesses can easily enter the market with simple Internet access and basic Web-based technologies, thus leading to a large number of geographically distributed information repositories. To cope with the distributed nature of the online accommodation

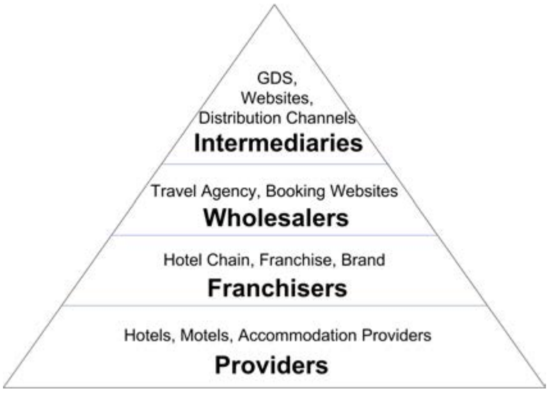

GDS,
Websites,
Distribution Channels
Intermediaries
Travel Agency, Booking Websites
Wholesalers
Hotel Chain, Franchise, Brand
Franchisers
Hotels, Motels, Accommodation Providers
Providers

Figure 1.1: The accommodation industry hierarchy

industry, and to provide integrated access to the scattered online information, an effecient and scalable information integration solution is crucial.

1.2 Research Scope

In order to enable seamless information sharing in the large, heterogeneous, dynamic and highly distributed on-line environment, an efficient and flexible information integration solution is required. Therefore, we define the mission of this thesis as:

> *The development of a semantic information integration solution to enable integrated information access in the large, heterogeneous, dynamic and highly distributed online environment of the accommodation industry, and to assist information sharing aross organisational boundaries.*

In this thesis, we mainly focus on the integration of structured and semi-structured data sources, where instance level data are stored and presented according to meta-data level schemas. In particular, we look at XML (Bray *et al.*,

1998) data sources that are stored according to XML schemas (Fallside, 2000). Furthermore, we do not consider unstructured information sources such as plain text, keywords, natural language sentences or other unstructured information formats.

In addition, we solve the problem of information integration at various levels of abstraction, including data level, process level and architecture level. On the data level, we use ontology to solve the issue of data heterogeneity, and to enable semantic inter-operability across large numbers of data sources. On the process level, we alter the process of information integration to facilitate the sharing of information. On the architecture level, we combine the flexibility offered by the emerging distributed P2P approach with the query processing capability provided by the centralized integration approach. Through the combination of various integration methods and techniques, we aim to develop an efficient, scalable and flexible solution for information integration.

1.2.1 Ontology and Information Integration

Ontology has long been used in information integration to facilitate the interoperation of heterogeneous information sources (Heflin & Hendler, 2000). In the context of information integration, ontology can be considered as a shared understanding of a specific domain, which contains collections of consensual concepts, relations, axioms and instances (Fallside, 2000). Using ontology, different systems or users can have a shared understanding of the same domain.

In this thesis, we use ontology to solve the issue of data heterogeneity, and to enable semantic inter-operation across a large number of heterogeneous data sources. We mainly focus on three aspects of ontology usage, including domain modeling, schema mapping and data mediation.

- Domain Modeling – First, we study the use of ontology in domain modeling, and use ontology to construct abstract data models of some domains. The constructed data models can capture basic conceptual elements defined in those domains, thus providing a semantically rich model over the integrated data sources.

- Schema Mapping – Then, we look at ways of automating the process of schema mapping, and use ontology to document the alignment defined between a pair of XML schema and ontology. Through the schema mapping process, a semantic network is formed, connecting a large group of heterogeneous data sources.

- Data Mediation – At last, we test the use of ontology for data translation, and use ontology to transform data between assorted formats. The transformation is performed using the conceptual mappings and alignments defined during design-time.

1.2.2 Information Integration Process

On the process level, we explore new ways of information sharing on the Web. The goal is to maximize the flexibility of the integration solution as well as to increase the capability of dealing with large-scale information integration. Inspired by the publish-discover-invoke paradigm invented by the Web service community, we break the process of information integration into a three-step collaboration process. The modified integration process provides a sufficient amount of freedom to information providers to allow the dynamic joining and leaving of data sources. It also distributes the tasks of information acquisition, schema mapping and mapping maintenance to the vast Internet users. Conventional integration processes often use a so-called combine-provide approach. In this approach, an integration solution is developed to collect, integrate and provide information over the scattered data sources (e.g. Chawathe *et al.*, 1994; Kirk *et al.*, 1995; Mena *et al.*, 1996). The tasks of data collection, integration and provision are often performed by one or a small group of people. The amount of work grows exponentially as the total number of integrated sources increases, thus it is not scalable enough to cope with large-scale information sharing. In this thesis, we look at possible ways of reduce the effort of data collection and integration, and examine the possibility of distributing the tasks of acquisition, integration and provision to various types of end users.

1.2.3 Information Integration Architecture

At last, we study the advantages and disadvantages of various types of information integration approaches, and examine the feasibility of each type of approach in online accommodation information integration. Based on the architecture, existing information integration approaches can be classified into two categories: data warehousing approaches (e.g. Dai & Zhang, 2006; Zhou *et al.*, 1995b) and wrapper-mediator approaches (e.g. Chawathe *et al.*, 1994; Kirk *et al.*, 1995; Mena *et al.*, 1996). The wrapper-mediator approach can be further divided into two sub-groups: centralized integration approach (e.g. Chawathe *et al.*, 1994; Kirk *et al.*, 1995; Levy *et al.*, 1996) and distributed peer-to-peer approach (e.g. Broekstra *et al.*, 2003; Lu, 2003; Ng *et al.*, 2004). We focus on four basic requirements of information integration, including accuracy, efficiency, flexibility and scalability.

- Accuracy – accuracy refers to the ability of providing correct results according to a user's request, and the ability to adjust to the constantly changing environment. Accuracy is a crucial requirement for this research, since some of the data sources in the online accommodation domain come and go in an unpredictable way.

- Efficiency – efficiency is another important requirement for information integration, it is a key measurement for evaluating the performance of the developed solution.

- Flexibility – flexibility refers to the ability of dealing with the changes occurred at the integrated data sources. As data sources come and go in an unpredictable way, the joining or leaving of an data source should not affect the overall performance of the integration process.

- Scalability – scalability refers to the ability of handling an increasing number of data sources.

1.3 Research Contribution

The main contribution of this thesis is a novel way of integrating heterogeneous data sources. From a technological perspective, this thesis provides a novel information integration solution for the large, heterogeneous, dynamic and highly distributed online environment. The proposed architecture contains a number of abstraction layers that allow the easy development of query-driven ontology integration solutions on the online e-commerce domain. The foundamental principle behind the proposed integration solution is an ontology mediated approach for solving data heterogenity issues. From a business perspective, the developed solution creates a virtual marketplace that brings together buyers and sellers to assist the information sharing and collaboration among all players in the online accommodation market. In addition, the solution enables inter-organizational information sharing among all players in the online accommodation domain, ranging from corporate consumers to international hotel chains to domestic travel agencies.

On a broad scope, this thesis contributes to the Semantic Web community in several aspects as listed below.

- The abstraction of information integration layers to allow the easy development of query-driven ontology integration solutions on the online e-commerce domain.

- The proposal of a semantic integration framework formed by three novel concepts, which are ontology mediation, the publish-combine-use integration process, and a hybrid integration architecture. The proposed solution solves the problem of information integration on three abstract levels.

- The proposal of an ontology mediated information integration approach for large scale integration of accommodation industry information on the Web. In the proposed approach, ontology is used as a mediator for solving the problem of data heterogeneity, and to enable semantic interoperability. A semantic network is established to provide seamless information sharing among heterogeneous data sources.

- In addition, this thesis explores new ways of information sharing on the Web. We change the conventional integration process into a large, three step collaboration task, referred to as the "Publish-Combine-Use". The proposed process distributes the tasks of information acquisition, integration and provision to various types of end users. It also gives some freedom to information providers to allow the control of their own data sources. Data sources can join and leave the information sharing network according to their own preferences. In the proposed process, information providers publish and maintain their information sources; hub operators combine and provide the integrated source; and public users use the integrated information sources to serve their own information needs.

- This thesis also introduces a hybrid information integration architecture that combines the traditional centralized information integration architecture with the emerging distributed peer-to-peer integration architecture. The proposed architecture is suitable for large scale information integration on the large, heterogeneous, dynamic and highly distributed Web environment.

This thesis also brings a number of contributions to the semantic Web community in terms of ontology utilization, data transformation, and information integration.

- We developed a novel approach for defining semantic mappings between XML schema and ontology. The proposed mapping approach uses ontology for representing concept mappings defined between a XML schema and an ontology, and the primary goal is to capture the unique information defined in both the XML schema and the ontology (Yang *et al.*, 2007).

- In addition, we proposed a compensational approach for instance level data transformation, mainly between XML and ontology data formats, and the transformation process is performed base on the predefined concept mappings (Yang *et al.*, 2008).

- To enable the efficient definition of semantic mapping among homogeneous Ontologies, we created a many-to-many ontology mapping algorithm. The proposed algorithm is developed based on the logic of the quick-sort algorithm and the idea of concept classification (Yang & Steele, 2009).

In general, the main contribution of this research is the proposal of a semantic integration framework that aims to solve the problem of information integration on three abstract levels. This research is beneficial to some of the information intensive industries including the tourism and travel industry, the online e-commerce industry, and other industries that require sophisticated information sharing and integration. In addition, this research also brings a certain amount of benefit to some research communities including the Semantic Web, Web information integration, and database communities.

1.4 Research Methodology

This thesis uses the design science research methodology adopted by some of the information systems researches (Hevner *et al.*, 2004). We choose to use the design science research methodology as the main focus of our research is the creation and evaluation of a novel information integration solution that can be directly applied into real world domains. This type of research fails well into the design science research field.

1.4.1 Design Science Research in Information Systems

Design science research involves the design of novel or innovative artifacts and the analysis of the use and/or performance of such artifacts to improve and understand the behavior of aspects of Information Systems(Hevner *et al.*, 2004). The core part of design science research is the design of an artifact. In the field of computer science, this includes but not limited to algorithms, approaches, methods, techniques, system architectures. In our case, the artifact is the proposed semantic information integration framework, and its containing elements as well as methods and techniques. Another important stage of design science research is the study of the created artifact. This includes the analysis of the performance or the use of the invented artifact. In this thesis, we focus on evaluating the performance of the proposed framework in large-scale information integration, where the integrated data sources are highly scattered and distributed. The proposed framework are evaluated against a number of evaluation criterion, including accuracy, efficiency, flexibility, and scalability.

1.4.2 Apply the Design Science Research Methodology

According to the design science research organization, a typical design science research consists of the following stages.

- Awareness of Problem: the awareness of an interesting problem that is either an industrial challenge or a widely studied research problem from a particular reference discipline.

- Suggestion: the suggestion phase of the research is to propose a solution for the identified problem using existing or newly invented methods or techniques. Suggestion is an essentially creative step wherein new functionality is envisioned based on a novel configuration of either existing or new and existing elements.

- Development: The implementation of the proposed solution. The implementation will vary depending on the artifact to be constructed. In most computer science related design science researches, the implementation will lead to the production of a software program or system.

- Evaluation: Once constructed, the artifact is evaluated according to criteria that are always implicit and frequently made explicit in the awareness of problem phase. Hypotheses are made about the performance or behavior of the artifact. Evidences are collected to support the approval of the identified hypotheses. In information system researches, this phase normally involves the preparation of a well-defined testing plan, collection of testing data, and the analysis of the collected data.

- Conclusion: This is the final phase of a design science research. Typically, it is the result of solving the identified research problem. Even in some cases, there are still deviations in the behavior of the artifact from the hypothetical predictions. It is important to provide adequate contributions towards solving the identified research problem.

Our research follows the steps defined in the design science research methodology. The following paragraphs document the way how design science research methodology is applied in this research.

- Awareness of Problem: Our research focuses on the integration of a large set of distributed and heterogeneous information sources. This is not only a challenging industrial problem, especially for the e-commerce domain as well as the online accommodation industry. It is also a widely studied research problem.

- Suggestion: Again, our research focus is to solve the integration problem using an ontology mediated framework. In this phase, we study existing information integration approaches, and compare their strengths and weaknesses. Based on the finding, we design the research artefact, including a set of information integration theories and methods.The proposed solution can be abstracted into three conceptual layers: ontology, process, and query. Ontology mediation is the fundamental principle behind the proposed framework.

- Development: The proposed research solution lead to the development of a number of artifacts (algorithms, methods), including:

  - A novel approach for defining semantic mappings between XML schema and Ontology.
  - A compensational approach for instance level data transformation, mainly between XML and Ontology data formats.
  - A many-to-many ontology mapping algorithm.
  - A bottom-up approach for resolving queries issued to the group of heterogeneous information sources.
  - We also implement the developed theories and methods in the form of a prototype system called "accommodation hub", so that we can conduct evaluation against the developed prototype system. The developed system demonstrates all major functionalities offered by the proposed integration framework.

- Evaluation: Each developed artifact is evaluated individually, including the proposed algorithms and methods. The overall framework is evaluated by measuring the performance of the developed prototype system named accommodation hub.

- Conclusion: At the final stage of our research, we documented all the findings discovered from this research, and submitted a number of research papers into relevant conferences and journals. Several research papers have been accepted and published. In addition, some findings discovered from the evaluation will lead to future research works, and the modification of the developed artefact.

1.5 Thesis Outline

This thesis is organized into nine chapters. We went through four research steps following the design science research methodology, including background study, method, prototype and evaluation. The first two steps fall into the process of *build*, and the last two steps fall into the process of *evaluate*.

Chapters 1-2 give a background study on the information integration domain. Chapter 1 starts with the discussion of the current nature of the online accommodation industry, and then looks at the research scope and its contributions. The research methodology used in this thesis is also discussed in Chapter 1. Chapter 2 gives a comprehensive study on previous works in information integration. We compare the advantages and disadvantages of the conventional information integration approaches with the focus of four integration requirements including accuracy, efficiency, scalability and flexibility. We also examine the capabilities of the conventional integration approaches in solving large scale information integration.

Chapters 3-6 introduce the developed techniques and methods. Chapter 3 gives an overview of the proposed semantic integration framework, and we present the proposed framework on three levels: data level, process level and architecture level. Various methods developed in the framework are also discussed, such as the ontology mediation approach, the “publish-combine-use” process, and the hybrid integration architecture. Chapter 4 expands on the data level solution, and gives a detailed explanation on ontology and semantic interoperability. In Chapter 5, we present the integration process, and tools developed to assist the integration process. In Chapter 6, we discuss the hybrid system architecture and its elements.

Chapters 7-8 evalute the proposed solution by implementing a real world prototype system. Chapter 7 shows the developed prototype system. The prototype system, named the “accommodation hub”, is implemented using the theories and methods proposed in earlier chapters. The evaluation of the proposed methods and techniques, as well as the evaluation of the developed prototype system are discussed in Chapter 8. Chapter 9 concludes this thesis with a summary of its achievements and future research directions.

Chapter 2

Literature Review

In this chapter, we look at some current information integration solutions, and we classify those solutions into two main categories: data warehousing based solutions (e.g. Dai & Zhang, 2006) and wrapper-mediator based solutions (e.g. Liang *et al.*, 2008).

In the data warehousing approach, heterogeneous data from distributed data sources are gathered together and placed into a central repository. A global schema is defined to provide unified query access to the integrated data. When a query is issued against the global schema, it is directly executed in the centralized data repository rather than being distributed to the individual data sources. The data warehousing approach offers a tightly coupled integration structure, as all data are collected, transformed, and loaded into a central location. Therefore, to provide users with the most up-to-date information, and to ensure the system coherence between the central repository and the local sources, data stored in the central repository needs to be periodically updated. While such an approach works effectively in small scale integration environments, it is, however, not feasible to periodically load and materialize data in a large-scale, especially when the integrated data sources are dynamic and autonomous.

In the wrapper-mediator approach, data from individual sources are neither materialized nor stored in a central location. Instead, wrappers are built on top of each individual source to provide managed data access, and a mediator is implemented to provide unified access to the connecting wrappers. In response to the query posed against the global schema, results are fetched directly from each integrated data source, and the fetched data are combined together to form the final result. Hence, the wrapper-mediator approach is more flexible, and provides up-to-date query results. However, in the wrapper-mediator approaches, the query posed against the global schema needs to be translated or reformulated into individual formats at run-time, and the results gathered from individual sources also need to be combined at run-time.

In order to further examine the capability of various integration solutions in large-scale information integration, in this chapter, we review some of the current solutions developed for online information integration. We select typical examples from both the data warehousing and the wrapper-mediator category, and compare the advantages and disadvantages of each integration approach. Since information integration is a fairly big research field, we do not attempt to give a comprehensive study of the entire domain; rather we focus on some of the typical solutions developed for the online information integration, and compare techniques and solutions developed for structured and semi-structured data sources. The remaining sections of this chapter are organized as follows: the next section discusses the data warehousing approach, and we look at three types of data sources, including database, Web information, and XML data source. Section 2.2 compares some integration solutions developed following the wrapper-mediator approach. Base on their architecture design, we categorize the discussed solutions into two groups: centralized approach and distributed approach. Finally, we summarize the research findings made from the literature review, and discuss the advantages and disadvantages of each type of integration solution.

2.1 Data Warehousing Approach

Early forms of data warehousing solutions are mainly used by the database community for integrating data from heterogeneous databases. Examples include the H2O project (Zhou *et al.*, 1995b). With the increasing popularity of the Internet, some researchers start to look at the problem of information integration on the Web, and several approaches were proposed including the multi-agent approach (Dai & Zhang, 2006) that aims to build a multi-agent system for integrating Web data, and the database approach (Prasad & Rajaraman, 1998) which aims to convert existing Web sites and applications into databases so that they can be queried using SQL statements. Efforts have also been devoted to the integration of XML data sources. Examples include the Xyleme (March, 2001) project that aims to build a vast XML data warehouse for integrating XML data on the Web.

2.1.1 Database

The H2O project (Zhou *et al.*, 1995b) is one of the early projects that focus on the data integration issues of data warehouse. It uses a hybrid virtual and materialized data integration approach, in which parts of the data is materialized in a persistent store while other parts of data are kept as virtual views. The benefit of this approach is that it allows the incremental updates of data from the operational databases to the data warehouse. This approach also optimizes the integration process by only materializing data that is critical to response time, and leaving other data as virtual views managed by mediators. A taxonomy is also introduced to assist the integration of data.

One major contribution of the H2O project is the development of a tool for generating integration mediators, called Squirrel (Zhou *et al.*, 1995a). Squirrel can be used to construct integration mediators for specific data integration applications. The mediator generation process relies on a high level language called Integration Specification Language (ISL), which is developed for modeling the integration problem on an abstract level. Based on the problem domain defined using ISL, data integration mediators are created. Mediators generated from Squirrel are used to manage virtual data, and to provide query pre-processing as well as query shipping functions. Another major contribution of the H2O project is the combination of both the data warehousing approach and the wrapper-mediator approach for data integration. However, since their main focus is on database integration, the developed solution can not be effectively used for information integration on the Web, thus not suitable for large-scale information integration.

2.1.2 Web Information

Dai and Zhang proposed a multi-agent based data integration framework for integrating Web information (Dai & Zhang, 2006). The proposed framework consists of two parts: a multi-agent run-time environment that enables the communication and collaboration among autonomous agents, and a Data Integration (DI) server that connects the agents with a central data warehouse. In their solution, agents travel on the Web to collect data available across the Internet. Upon the identification of a new data source, data is extracted, translated by the agent, which is then sent to the DI server. The DI server loads the received data into the central data warehouse.

In their research, agent technologies are mainly used to cope with two aspects of data integration, which are data collection and data distribution. Although the benefits offered by the multi-agent technologies can be well adapted to assist the data collection and distribution processes, issues exist in other aspects of data integration such as data transformation and integration remain unsolved.

In contrast to the Dai and Zhang's approach, the solution developed by Li et al. (Prasad & Rajaraman, 1998) more focuses on the processes of data transformationn and integration. In Li's approach, they developed a virtual database management system (VDBMS) to transfer existing Web sites and applications into relational databases. The VDB technology creates a relational view over existing Web information sources, and provides SQL query capabilities to external applications via ODBC and JDBC interfaces. Based on the developed VDB techniques, they created a web-based data warehouse to enable the sharing and mining of Web information. Although their solution provides an automatic way of generating data interfaces, the integration process still involves large amounts of programmatic work, and the generated code is not reusable. Hence, both Dai and Zhang's approach and Li's approach are not suitable for large-scale information integration.

2.1.3 XML Data

With the increasing popularity of XML and the growing amount of XML data sources available online, some researchers start to look at the integration of XML data sources. One popular project is the Xyleme project (March, 2001), which aims to build a data warehouse to integrate XML data on the Web. The Xyleme system uses a four layers structure, where each layer provides an unique set of functionalities. The four functional modules include a physical layer, a logical layer, an application layer and an interface layer.

The physical layer includes a data repository and an index manager, and is responsible for the storage of XML documents. In the data repository, XML data is stored in DOM tree formats (Marini, 2002) in conjunction with byte streams to optimize the processing speed, and the index manager handles the indexing of XML documents. The logic layer is responsible for data acquisition and query processing, and it contains a crawler and acquisition module that collects XML documents on the Web. The logic layer also contains a loader that loads the acquired XML documents into the central data repository, and the layer also has a semantic module and query processor which together provides integrated query capability for end-users. The application layer provides change management and monitoring of the identified XML data source. Finally, the interface layer provides interfaces to the end-users as well as to other Xyleme client applications.

Xyleme uses tree based structure and tree types for modeling data sources. An abstract tree type is defined to provide mediation between the user's query and the database. The role of the abstract tree type can be considered as a global data schema that provides uniform access to a set of heterogeneous XML sources. Users can issue a query by querying on a sub-tree structure from the abstracted tree type, and this allows users to issue queries without having to be aware of the actual schema describing the XML sources. During the query processing phase, the issued query will be translated into concrete query formats that comply to the XML data schema stored in the repository, and each concrete query will be processed by the database. Although Xyleme materializes data in a central location to enable data integration, the way queries are handled in Xyleme is similar to conventional wrapper-mediator approaches, where semantic mappings are defined between local and global schemas to enable dynamic query translation and processing at run-time. Mapping information is stored into map translation tables, and is used for pre-evaluating queries at compile time.

Benefits offered by Xyleme include the ability for storing and querying on a large set of heterogeneous XML sources, and the ability of querying XML sources using abstract tree structures. However, since all the XML data sources are materialized in a central location, the Xyleme solution can not be used in cases where the integrated data sources are constantly changing.

2.1.4 ATDW

The ATDW (Australian Tourism Data Warehouse) is an cooperative initiative to standardize Australian tourism product information. One major output of the project is the development of a data storage system that is used to store various types of tourism information, including accommodation, attraction, event, tour, hire and transport products information. The idea of the ATDW initiative is similar to the data warehousing approach, where a centralized data storage system is used to store the integrated information. However, the data storage system mainly acts as a large business registry, and it does not provide real-time information such as up-to-date price information and service availability information. In contrast, our research aims to provide real-time access to the integrated online information.

2.1.5 Conclusion

The data warehousing approach typically materializes data into a central repository for query processing. Although the materialization of data allows effective query processing regardless of the availability of actual data source, this approach also has a number of limitations:

- Data Inconsistency – data materialized in the central repository is not consistent with the actual data stored at the local data source. For example, changes occurred at the local data source are not immediately reflected by the central data repository.

- Low Scalability – to ensure the materialized data is up-to-date and consistent, data stored at the central repository needs to be updated periodically. As a result, scalability issues appear when the number of integrated data sources is large and increasing.

- Low Flexibility – query processing heavily relies on the availability of the central data repository, failure of the central repository will lead to the failure of the entire system.

Therefore, the data warehousing approach is mainly used in domains where the data coherence is not essential, and the number of integrated data sources is comparatively small. Hence the approach is not widely used for large-scale information integration.

2.2 Wrapper-Mediator Approach

Comparing to the data warehousing approach, the wrapper-mediator approach does not materialize data into a persistent location. Instead, data are directly fetched from individual data sources during execution time. In contrast to the data warehousing approach, the wrapper-mediator approach provides more up-to-date information and is more capable for integrating information sources in highly dynamic environment such as the Web. The wrapper-mediator approach often consists of two parts: a wrapper that provides data access to the individual data source, and a mediator that provides data translation among different formats. In most cases, a mediator is connected to a large number of wrappers to provide centralized and unified access to the connecting wrappers (Liang *et al.*, 2008). However, in some cases, a wrapper also acts as a mediator and offers mediation between heterogeneous data formats (Ng *et al.*, 2004). Based on the way wrappers are associated with mediators, we classify the existing wrapper-mediator approaches into two categories: centralized approach and distributed approach.

The centralized data integration approaches often relies on the definition of a global schema to provide a reconciled and integrated view of all the underlying local data sources. In other words, a global schema is defined to provide semantic mediation between heterogeneous local schemas. Examples of the centralized integration approach include TSIMMIS (Chawathe *et al.*, 1994), (Glimm *et al.*, 2007), (Liang *et al.*, 2008), and (Shi *et al.*, 2010), just to mention a few. Based on the way that local data sources are related to the global schema, the centralized integration approach can be further classified into two sub-categories: Global-as-View (GaV) approach (Glimm *et al.*, 2007) and Local-as-View (LaV) approach (Fundulaki *et al.*, 2002; Manolescu *et al.*, 2000). In the GaV approach, the mediated global schema is expressed using terms defined from the local data sources, whereas in the LaV approach, local data sources are defined as views of the global schema. The following sub-sections introduces some GaV and LaV integration solutions.

In contrast to the centralized approach, the distributed integration approach uses groups of distributed mediators to tackle the problem of semantic interoperability. Each individual data source itself is a wrapper and also a mediator. All data sources are connected in a distributed Peer-to-Peer (P2P) fashion (Ng *et al.*, 2004), and each pair of relating sources are linked via the semantic mapping defined on the schema level. When a query is posed against one local data source, it is then propagated to its immediate neighborhoods, and such process repeats until the issued query is fully resolved. Current works in the field of distributed data integration will be covered in Section 2.2.2.

2.2.1 Centralized Integration Approach

Early forms of the centralized integration approach can be traced back to the early 1990s, where the concept of federated databases emerged (Sheth & Larson, 1990). The initial idea of the federated database system was to integrate multiple autonomous database systems into one single system to provide a uniform query interface to end-users. The data integration approach used in most federated database systems can be considered as early forms of the wrapper-mediator approach, where each constituent database is managed by a wrapper and the federated database acts as the mediator. In the late 1990s and early 2000s, the wrapper-mediator approach was gradually adopted by the Web community for integrating information on the Web. Early efforts (Chawathe *et al.*, 1994) used a global schema for querying heterogeneous data sources, and the global schema is described as an unified view to local data sources. This approach later became known as the Global-as-View (GaV) approach (e.g. Liang *et al.*, 2008; Shi *et al.*, 2010). The approach introduced by the information manifold project (Kirk *et al.*, 1995) uses a predefined global schema for modeling local data sources, and this approach is later known as the Local-as-View (LaV) approach (e.g. Fundulaki *et al.*, 2002; Manolescu *et al.*, 2000). Since both the GaV and the LaV approach have their own strengths and weaknesses, some researchers suggest to combine the two approaches. As a result, solutions such as the GLaV (Friedman *et al.*, 1999) approach and the BGLaV approach (Xu & Embley, 2010) emerge. This section discusses various types of centralized integration solutions.

2.2.1.1 Global-as-View (GaV)

TSIMMIS

The TSIMMIS project (Chawathe *et al.*, 1994) is a joint effort between Stanford University and the IBM Almaden Research Center. The project aims to develop a set of tools to facilitate the integration of heterogeneous data sources. In the project, a wrapper-mediator architecture is introduced for integrating both structured and unstructured data. Each data source is connected with a wrapper that is responsible for translating user queries written in a common model to local executable requests as well as translating the returning results back to the common format. On top of the wrappers is a mediator that ships user queries to their corresponding information sources. In addition, the wrapper is also responsible for combining the set of results received from individual wrappers and return it back to user. Duplicated information are removed from the merged results. The data integration architecture used in the TSIMMIS project is a typical example of the wrapper-mediator integration approach.

One major part of the TSIMMIS project is the development of a common data model called the Object Exchange Model (OEM) and a query language named OEM-QL. OEM is an object-based information exchange model that is used to provide reconciliation over heterogeneous data models, and the OEM-QL uses a form of object-logic. However, since the structure of OEM is quite primitive, non-comprehensive and does not support modeling of sophisticated object relationships, it was later replaced by other types of data models such as XML schema (Fallside, 2000) and ontology (Kalinichenko *et al.*, 2003). Due to the fact that the global data model is constructed as views over the underlying local sources, updates occurred at local level will lead to the recreation of the global model, and this is also a common drawback for other GaV integration approaches. The TSIMMIS project has brought significant influences to later data integration researches.

Garlic

Similar to the TSIMMIS project, the Garlic project also uses a common object-oriented data model for integrating heterogeneous information sources. In addition, it provides a query language base on object-logic to enable the querying of multiple individual data sources. However, the Garlic project (Carey *et al.*, 1995) is more focused on the integration of heterogeneous multimedia information sources, where data is stored in various repositories including text files, databases, image servers, video servers and so on. Unlike the TSIMMIS project, the Garlic project does not provide any tools for fast generation of wrappers and mediators. Rather, it provides a data management solution to transform multimedia sources via semantic data connections. Although the solution looks like a conventional database management system with an object-oriented schema, in fact, it is a collection of scattered and highly distributed data sources. The Garlic project also uses a typical Global-as-View integration architecture, where a global data model is defined as views over the local data sources. Similar to the TSIMMIS project, the data model used in Garlic is also quite primitive, non-comprehensive and does not support modeling of sophisticated object relationships, thus a lot of programming efforts is needed during the integration process.

COIN

The COIN project (Goh, 1997) introduces the idea of using logic-based object-oriented formalism for reconciliating semantic heterogeneity and to achieve semantic interoperability among heterogeneous data sources. They argue that the same concept may differ from source to source when taking consideration of their contexts. For instance, the concept of money amount refers to the same thing for two data sources S_1 and S_2, however, if S_1 and S_2 exist within different contexts (e.g. S_1 is in Japan and S_2 is in the U.S.), they will have different semantic meanings. When taking the factor of currency exchange rate into consideration, the concept "money amount" from S_1 is different from the concept "money amount" in S_2, and the value of S_1 is far less than the value in S_2. Hence to solve the problem of semantic heterogeneity caused by various context factors, they proposed a logic-based framework, called the context interchange framework, to support the automatic reasoning and translation of queries into proper formats, and to eliminate their context heterogeneity.

The proposed context interchange framework is comprised of three components: a domain model with a collection of primitive types and semantic types, a set of elevation axioms, and a set of context axioms. The domain model acts as a global schema, and provides a unified view of a specific domain. To a certain extent, the domain model can be considered as a domain ontology. The set of elevation axioms stores the semantic correspondence defined between concepts from the domain model and the actual data source. And the set of context axioms is used to associate the data sources with their corresponding context, and to support the correct transformation of data. Compared to previous projects such as the TSIMMIS and Garlic, the domain model proposed in COIN is more comprehensive and allows the inclusion of various context information. Further, the proposed model also allows simple reasoning based on description logic, thus is more effective in solving semantic heterogeneity caused by context factors.

MIX

The MIX (Mediation of Information using XML) project (Baru *et al.*, 1999) uses XML DTD as a semi-structured data model for data integration. The choice of using XML DTD allows more flexible and structured modeling of data sources. In their approach, queries are represented using a declarative query language named XMAS. XMAS supports object fusion and pattern matching on input XML documents, and it also provides an effective grouping and ordering mechanism for fast XML result generation. A graphical user interface is also implemented to assist the querying and browsing of XML data. However, the MIX project mainly focuses on semi-structured data sources, and each data source must conform to the associated DTD schema. Furthermore, the use of XML schema as a global data model does not solve the issue of semantic heterogeneity, as XML schema emphasizes on the structure of the integrated data source rather than its semantic meaning. Hence, the solution developed from the MIX project can not cope with large-scale information integration.

Ontology-based Approaches

Due to the expressive power offered by ontology and the ability of handling simple reasoning tasks, ontology become widely used by the information integration community (Fallside, 2000). One early example is the SIMS project (Arens *et al.*, 1993). SIMS uses ontology to model the concepts and logic defined for a shared domain. All information sources are modeled using concepts from a global ontology, and objects from each data source are associated to concepts of the ontology. However, the ontology model used in SIMS is not formally defined using ontology languages, thus it does not provide reasoning capabilities.

The work done by Cruz et al. from the University of Illinois at Chicago (Cruz *et al.*, 2004) focuses on the integration of XML data sources. They used an ontology mediated integration approach where local XML schemas are lifted into RDF ontology, and each local RDF ontology provides basic structure and semantic modeling of its underlying XML data source. All local RDF ontology are merged together to form an integrated global RDF ontology. Their approach can be considered as a typical example of the GaV integration approach, where the global schema is defined as an unified view over its underlying data source schemas.

In Cruz's approach, a query posed against the global ontology is decomposed into sub-queries, where each sub-query is answered by one data source. Queries posed against an individual data source can also be answered by other local peers. Two query rewriting algorithms are proposed to assist the query execution process. In the first case, a query posed against the global ontology is rewritten into a union of sub-queries, and each sub-query is executed over one XML data source. Answers received from individual data sources are integrated to produce the final answer. In the second case, a query posed against a local XML source is executed, at the same time, it is also rewritten into a union of sub-queries via the global ontology, and the sub-queries are executed to retrieve answers from other parts of the system. Received results are combined. The query rewriting process is based on the conjunctive formula (Glimm *et al.*, 2007). Comparing to some earlier solutions, the Cruz's approach offers higher flexibility for large-scale information integration, as they propose to use ontology mediation for transforming data across different formats. However, since Cruz's approach is base on the GaV design principle, significant programming effort is involved when dealing with changes occurring at local data sources.

The GaV approach also has been applied to some industrial domains such as the medical information integration (Shi *et al.*, 2010), deep web information integration (Liang *et al.*, 2008). Yunmei et. al. (Shi *et al.*, 2010) propose to use semantic web-based technologies to integrate heterogeneous data from medical information systems. In their approach, an ontology is usde as the global data model, and the global ontology is generated by merging the set of local data models provided by each individual data source. They also used the OGSA-DAI accessing middleware for querying each local data source. Ping Liang et. al. (Liang *et al.*, 2008) uses the GaV approach for integrating hetereogenous web information sources. In their approach, the local database schema is first lifted into a local ontology, which is then mapped to a global ontology. They also proposed a semantic query rewriting algorithm to reslove the queries issued at the global level.

Conclusion

The GaV solutions have evolved from relational data integration solutions (e.g. Chawathe *et al.*, 1994) to heterogeneous semi-structured and structured data integration solutions (e.g. Baru *et al.*, 1999; Liang *et al.*, 2008; Shi *et al.*, 2010). The data model used by the GaV solutions also changed from a simple object-oriented model (e.g. Carey *et al.*, 1995) to a complex and meaningful ontological model (e.g. Cruz *et al.*, 2004; Liang *et al.*, 2008; Shi *et al.*, 2010). In the GaV approach, a global schema is defined as a joint view over the local data sources. Benefits offered by the GaV approach include:

- Simple Query Rewriting – query rewriting in the GaV approach is simple, as the global schema is defined as a collection of views on the local data sources. Queries posed on the global schema can be directly replaced using terms and expressions defined from the local data source schema.

- Efficient – query rewriting in the GaV approach does not involve complicated query reformulation steps, thus query processing is faster in comparing to the LaV approach.

However, one major drawback of the GaV approach is its scalability. Since the global schema is defined as a joint view of local data sources, the joining or leaving of local data sources will lead to the recreation of the global schema. Hence the GaV approach is not scalable for large scale data integration, and not suitable for data integration in dynamic environments such as the Web.

2.2.1.2 Local-as-View (LaV)

Information Manifold

The LaV approach is initially proposed by the IM (Information Manifold) project (Kirk *et al.*, 1995). Unlike the GaV approaches, IM uses a well formed knowledge base to describe local data sources. The content of a local data source is represented using a combination of both the CLASSIC description logic and the horn rules (Brachman *et al.*, 1991).

The core of IM is the definition of the knowledge base, which includes a domain model and properties of external data sources. The domain model is defined as a collection of constraint formula, where each formula contains conjunctions and disjunctions of atoms. For instance, given two primitive concept terminologies "business" describing businesses, and "business_type" referring to the role of a business. The concept "travel_agent" can be modeled as

$$\text{(business} \cap \text{(fills business\_type "Travel"))}$$

where both $\cap$ and "fills" are binary relations describing the relationships between objects, and the value "Travel" is a constant.

The contents of external data sources are also represented in the knowledge base. For sources without internal structures, basic properties such as locations, protocols, topics and ownership are stored. For structured data sources, semantic mappings are defined to model the relations between inter-connecting data sources. For example, given two external data sources provided by Qantas Airline: $q\_flights(F, Dep, Arr)$, refers to the information of flight F departure from Dep to destination Arr, and the other one $q\_quote(A, F, P, D)$ representing the quote given by agent A on flight F with price P and date D. Then the relationship between these two information sources can be represented using existing domain relation $quote(A, Airline, Dep, Arr, P, D)$, where the relation quote is a predefined relation in the knowledge base representing the domain of a flight quote. The relationship between the given information sources is defined as follow:

$$\text{q\_flights(F, Dep, Arr)} \wedge \text{q\_quote(A, F, P, D)} \Rightarrow \text{quote(A, 'Qantas', Dep, Arr, P, D)}$$

During query answering, a set of relations is selected from the domain model, the system then determines the group of external data sources needed to compute the selected relation. The posed query is then divided into sub-queries, and executed by the selected information sources. Answers received from each local source are collected and combined, and then translated into the format comply to the global domain model. This query answering process later become known as query rewriting (Papakonstantinou & Vassalos, 2006). One major benefit offered by the IM solution is the ability of performing efficient inference based on the predefined knowledge base, and the ability of resolving integrated queries.

Agora

Manolescu et al. (Manolescu *et al.*, 2000) propose to use XML schema as a global data model for integrating relational and XML data sources. Unlike the MIX project, Agora follows the LaV integration approach. In their approach, individual data sources are defined as views over the global XML schema, and the local views are represented in the form of relational schema. Query execution in Agora is divided into two parts: In the first part, the global XQuery is normalized and a virtual generic relational schema is created to assist the query translation process. The normalized query is then translated into a SQL query. The normalization and translation of XQuery into the SQL format is completely independent from the relation defined between the local source and the global schema. In the second part of query processing, the generated SQL query is rewritten into a SQL query that can be executed by the actual data source, and this translation is based on the relation defined between the global schema and the local source. Drawbacks of the XML based integration approach include the inability of resolving semantic heterogeneity, low flexibity and scalablity.

STYX

The solution developed from the C-Web project (Fundulaki *et al.*, 2002), called STYX, uses ontology as the global data model for data integration. In this approach, XML sources are described as views of the global ontology, and they use a path-to-path based mapping rule to encapsulate concept mapping defined between local data sources and the global ontology. The concepts from local XML sources are represented as XPaths, and their corresponding concepts from the global ontology are expressed as schema paths. Concepts from interconnecting sources are linked by keys to allow the joining of results during query processing, and the idea of key is similar to the idea of key in relational databases. The query language used in the STYX system follows the OQL syntax, and the query process consists of query translation and query rewriting.

Conclusion

Similar to the GaV solutions, some of the LaV integration solutions also experienced the transition from relational data integration (e.g Kirk *et al.*, 1995) to structured and semi-structured data integration (e.g Fundulaki *et al.*, 2002). In addition, the data model used by LaV integration solutions also changed from a simple description logic based model to a complex data model defined using ontology. Since the LaV approach use a predefined global schema for data integration, and local data sources are modeled as views of the global schema, thus the joining or leaving of local data source will not affect the global schema or mappings to the global schema. Hence, the LaV approach offers some benefits, which include:

- Scalability – changes that have occurred at local data sources will not affect the global schema, and the joining or leaving of local data sources will only lead to the recreation of mappings to the global schema and does not require the re-definition of the global schema. Hence the LaV approach is more scalable for large, dynamic data integration.

However, the utilization of a predefined global schema for data integration has a number of drawbacks:

- Complex Query Rewriting – query rewriting in LaV approach involves a number of complex reformulation steps. The process of translating a query posed over the global schema into a set of queries on local data sources is known as query reformulation. Although a lot of research effort has been spent to automate the process of query reformulation, some of the developed techniques are still manual, requiring complex programmatic setup, and have limited code reusability (Levy *et al.*, 1995).
- Complex Global Schema – since the global schema is defined independently from local data sources, the definition of the global schema requires in-depth knowledge across all local data sources, which makes the schema definition process more complicated, especially when the number of integrated data sources is large and heterogeneous.

Although the LaV approach has been successfully applied to the field of Web information integration, the limitation exposed by the LaV approach becomes a bottle-neck for data integration in the large, heterogeneous, dynamic, and distributed domain. Significant efforts are required to create effective techniques for query rewriting, as well as for the definition of a global schema to cover all concepts across different data source domains. Such efforts grow significantly as the integration domain become bigger.

2.2.1.3 Hybrid Approach

Since both the GaV approach and the LaV approach have several limitations, some researchers propose to combine the two approaches to create a more optimized information integration solution. One solution proposed by Friedman *et al.* (1999) combines the expressive powers of both LaV and GaV. The approach, known as GLaV (Global-Local-as-View), uses recursive queries to define views over local data sources, and data can be derived from local views defined over the global schema. In addition, the GLaV approach allows the conjunction of global relations, which is beyond the expressive power of the GaV approach. In comparison with both the GaV approach and the LaV approach, the GLaV approach is more flexible for dynamic information integration where the availability of local data sources are hard to predict.

The solution proposed by Cali et. al. (Cali *et al.*, 2002) uses a translation algorithm to translate the LaV approach into a GaV approach for fast query processing. The logic program produced from the translation algorithm is used for answering queries using query unfolding techniques. Hence, the Cali et. al. approach combines the scalability offered by LaV with the fast query processing power offered by GaV. However, the evaluation of the generated logic program is comparatively slow and requires longer query processing time, which neutralized the query processing simplicity offered by GaV.

Xu and Embley (Xu & Embley, 2010) proposed a BGLaV approach, and they use source-to-target mappings based on a predefined and independent ontology schema to encapsulate views defined for local data sources. Views and query unfolding information are semi-automatically generated at mapping time, and stored by the defined mapping elements. Therefore, they claim that their approach provides better query performance than the Cali et. al. approach.

Conclusion

The hybrid integration approach combines the benefits of both the LaV approach and the GaV approach to provide better integration solutions. Although some solutions are able to reduce the complexity of query reformulation while maintaining high scalability offered by LaV, the complexity of mapping definition becomes a trade-off. The effort spent for defining mappings from the local data sources to the global schema has increased. Hence, an important step towards the creation of an effective hybrid integration solution is to reduce the complexity of schema mapping definition, and this is one of the primary goals of our research.

2.2.2 Distributed P2P Approach

Content-based search in P2P networks (Lu, 2003) has received a large amount of interest in recent years, and has led to the emergence of semantic P2P networks. Several initiatives are carried out including SWAP (Broekstra *et al.*, 2003), PeerDB (Ng *et al.*, 2004), which aim to tackle the problem of semantic interoperability in a large, global scale by using the P2P architecture. In contrast to the traditional centralized data integration approach, the P2P approach aims to achieve semantic interoperability using local agreements established among neighbourhood peers. The local agreement is reached in the form of local semantic mapping between related data source peers. Based on the defined mappings, data management tasks such as query routing, query reformation and query processing can be performed in the semantic P2P network. Centralized integration solutions often rely on the definition of a global schema to provide reconciled and integrated views of the underlying local data sources, and the definition as well as the enforcement of the global schema are often hard to achieve. In contrast to the centralized integration approaches, the distributed approach leverages the power of local consensus and self-organization, and the global level semantic agreement is achieved using local peer-to-peer based semantic mappings. This section discusses some of the works done in distributed information integration.

PeerDB

PeerDB is a joint effort done by National University of Singapore and Fudan University (Ng *et al.*, 2004). The PeerDB system uses a P2P-based architecture, where each peer is an independently operated data management system, and is linked to a group of neighborhood peers. The connections established across all local peers form an integrated data sharing network. Relying on the formed data connection network, PeerDB enables seamless data sharing without any global schemas. Each peer within the PeerDB network is a full fledged object management system with content-based searching capabilities. Query reformulation in PeerDB is based on keyword matching as oppose to schema mapping used by many other PDMS systems (e.g. Halevy *et al.*, 2003), thus it is faster than schema mapping based query reformulation processes. However, the reformulated queries are less meaningful, and the execution of the reformulated query need to be managed by human users.

Conclusion

In contrast to the centralized integration solutions, the distributed P2P approach has a number of advantages:

- Scalable – it is more scalable in the sense that no global schema is required for data sharing, and global consensus can be reached in the form of interconnected local agreements. Changes occurred at local data sources will not affect the overall data sharing network.
- Flexible – the sharing of data does not rely on any centralized modules, and the failure of local peers will not affect the overall data sharing network. In

addition, data sources in the centralized integration approach are added or removed in a controlled manner; whereas in the distributed P2P approach, local data sources can join or leave the information sharing network according to their own interests.

Limitations exposed by the distributed P2P integration approach include:

- Low Efficiency – query processing are performed in an iterative P2P fashion, which involves repetitive number of query translation and reformulation steps.
- Low Accuracy – query accuracy becomes a crucial issue when inconsistent mappings are defined in the data sharing network. This problem was initially introduced by Aberer *et al.* (2002), and they proposed a semantic routing algorithm base on probability to tackle this issue.

The distributed P2P approach provides a more flexible way of data sharing, where individual data sources are linked in a self-organized fashion. The flexibility offered by the distributed P2P approach can be well applied to large-scale data integration. Inspired by the distributed approach, our solution also uses the self-managed information sharing approach, where information owners can join and leave the information sharing network depend on their own choices.

2.3 Discussion

In this section, we summarize the research findings made from the previous sections. We compare the advantages and disadvantages of current data integration approaches, including the data warehousing approach, the centralized data integration approach, and the distributed P2P approach. Goh et al (Goh *et al.*,

1994) revealed a number of challenges for large-scale data integration in the dynamic and distributed environment. Their findings show that an effective data integration solution should be able to handle constant data source changes while providing sufficient system efficiency and accuracy. Hence, we compare the current solutions based on four integration characteristics, which are scalability, flexibility, efficiency and accuracy. Scalability and flexibility refer to the ability of handling a growing number of data sources and the ability of coping with data source changes, such as the joining or leaving of data sources. Efficiency and accuracy refer to the ability of effectively and correctly handling user requests and performing data integration tasks.

2.3.1 Scalability and Flexibility

In the data warehousing approach, data collected from local sources are materialized, transformed and loaded into a central repository. To provide users with the most up-to-date information and to ensure the data consistency between the central repository and the local data sources, data stored in the central repository needs to be periodically updated. The update process requires the reloading and materialization of data from all local sources, and the complexity of the integration task grows significantly as the number of sources becomes larger. In addition, changes occurred at local sources can not be immediately reflected by the central repository. Therefore, the data warehousing approach offers limited scalability and flexibility for large-scale data integration.

In contrast, the wrapper-mediator approach offers better flexibility and scalability, as data gathered from local sources are not materialized, thus it is able to provide the most up-to-date information to end-users. Among various wrapper-mediator solutions, the centralized integration solutions often require the definition of a global schema. In the LaV approach, the global schema definition requires in-depth knowledge across all local domains, and the task of global schema definition becomes more complex as the number of data sources increases. In addition, applying the defined global schema is also hard, as it needs the support of all the involving parties. In the GaV approach, the global schema is generated as views of local source schemas. As a result, the joining or leaving of local data sources will lead to the recreation of the global schema. Therefore, both the LaV approach and the GaV approach experience scalability and flexibility limitations for large-scale information integration, and both approaches require the definition and maintenance of a global schema.

On the other hand, distributed integration solutions often rely on the local semantic relationships built between neighourhood data sources. In some cases, the distributed approach can be considered as a bottom-up agreement-building process, where a global level agreement is achieved by building local, pair-wised agreements between neighbourhood data sources known as peers. In order to become part of the global network, each peer needs to define some local mappings to a small set of related peers, and the global network is gradually formed by the joining of vast numbers of peers. This approach offers higher flexibility and scalability, as the joining or leaving of peers will not affect the overall performance of the data sharing network. The distributed approach considers global semantic interoperability as collections of local data connections, and the formation of the global level agreement does not require the enforcement of a global schema. Therefore, it offers higher scalability and flexibility for large-scale information integration.

2.3.2 Efficiency

The efficiency of current information integration solution is dependent on a number of factors, including the integration architecture, the integration process, and the tools and techniques developed for the integration process. From an architecture point of view, the centralized integration approach offers higher efficiency as data is directly transformed from the source format to the target format. In contrast, data transformation in the distributed integration approach often involves a number of mediation steps. Data fetched from a local peer is transformed into several mediation formats before it can be completely transformed into the target format.

In the centralized integration scenario, a query request issued to the system is directly resolved by its query rewriting techniques. The issued query is dispatched to a number of local data sources, and results fetched from those sources are combined together to generate the final answer. However, in a distributed integration scenario, a query issued against one local peer is passed to some other local peers connected in the network, and each peer acts as a mediator as well as a contributor to the integration process. Hence, query processing in the distributed approach requires an extract step called "query routing". Query routing can be considered as the process of identifying the right data source for answering a particular query. Most centralized solutions do not require the routing step, as the centralized system unit often maintains a global index over the connecting data sources. Thus the source selection process is fast and simple. However, since the distributed integration approach does not have any centralized components or index, the identification and selection of data sources can only be performed using broadcasting techniques. Aberer et al (Aberer *et al.*, 2002) proposed a semantic gossiping algorithm for passing queries in a semantic P2P network, their solution is similar to the flooding techniques used in the unstructured P2P networks. Tempich et al (Tempich *et al.*, 2004) proposed a query routing approach based on social metaphors. In their approach, peers are able to memorize the queries successfully answered by their neighbors to reduce the amount of peer selection effort required for query answering. Although the techniques developed from previous research efforts are able to improve the efficiency of query routing in the semantic P2P network, in general, query routing in the distributed integration approach still expose some efficiency issues, especially when the number of data sources is large.

Many techniques and tools are developed to assist the integration process. Some researchers focus on the improvement of the schema mapping process (Doan *et al.*, 2002; Mitra *et al.*, 2000), and some aim to automate the task of data transformation (Rodrigues & Cardoso, 2006). However, limited efforts are spent for creating large-scale integration techniques. For instance, among the current mapping techniques, most of them mainly focus on the mapping of a pair of ontologies. While those techniques are effective in creating one-to-one ontology mappings, they are less efficient when dealing with the many-to-many ontology mapping scenarios. In addition, the majority of the developed integration solutions and techniques are still manual or offer limited usability, and some even require complex programmatic setup and configuration.

2.3.3 Accuracy

Schema mapping and query processing are important steps in information integration. Query processing in the GaV approach is simple, as the global schema is defined as a collection of views on the local data sources. Queries posed on the global schema can be directly replaced using terms and expressions defined from the local data source schema, and this process is also known as query unfolding (Qian, 1996). In contrast, query processing in the LaV approach is more complex, and requires the rewriting of query using different views. Several view based query rewriting algorithms are developed including the Bucket algorithm (Grahne & Mendelzon, 1999; Levy *et al.*, 1996), the inverse-rules algorithm (Qian, 1996) and the MiniCon algorithm (Pottinger & Levy, 2000). We refer you to (Halevy, 2000) for a comprehensive study on view based query rewriting. On the other hand, query processing in the distributed approach uses mixture of both query unfolding and view based query rewriting techniques (Tatarinov & Halevy, 2004). The query reformulation process is performed in an iterative and collaborative fashion. A query posed against one local peer is reformulated and propagated to its immediate neighbors, and its neighbors reformulate the received query and pass it onto their neighbors. Such process continues until either the query has been fully resolved or been propagated to the entire network.

Since the query reformulation process is heavily reliant on schema mappings, ensuring the accuracy of the defined schema mapping is a necessity for correct information integration. Due to the fact that most online data sources are highly autonomous and heterogeneous, semantic conflicts occur on a frequent basis. The centralized approach uses a global schema to eliminate irriconcilable semantic conflicts. However, the problem is hard to prevent in the distributed approach, as each peer has limited knowledge of the semantic mappings defined on the rest of the network, mapping inconsistency become a major issue. Cudre-Mauroux et al (Cudre-Mauroux *et al.*, 2006) proposed a probabilistic message passing algorithm for reducing the query errors caused by mapping inconsistency. Although the proposed algorithm provide run time solution for query accuracy improvement, it retains certain efficiency limitations, as each query message routing step requires a number of consistency checking calculations. Such calculation can be eliminated if semantic mappings defined in the network are consistent and correct. Hence, schema mapping is an important step in information integration.

We summarize our findings in the following table, each integration approach is rated based on four integration characteristics.

Table 2.1: Integration approaches comparison

Integration Techniques/Characteristics	**Global-as-View**	**Local-as-View**	**Distributed Integration Approach**
Semantic Interoperability	Top-Down Approach via Global to Local Mapping	Top-Down Approach via Global to Local Mapping	Bottom-Up Approach via Local Mappings
Change of Data Source	Global Schema Regeneration	Expressing Local Data Source using Global Schema	Regeneration of Local Neighborhood Mapping
Query Processing	Single Query Unfolding	Single View based Query Rewriting	Iterative Query Unfolding & View based Query Rewriting
Query Routing	Direct Data Source Selection	Direct Data Source Selection	Iterative Broadcasting Techniques
Query Accuracy Enhancement	Schema Mapping Validation	Schema Mapping Validation	Runtime Inconsistency Detection
Schema Mapping	Single Direction Mapping	Single Direction Mapping	Bi-directional Mapping
Scalability	Low	Medium	High
Flexibility	Low	Medium	High
Efficiency	High	Medium	Low
Accuracy	High	High	Low

2.3.4 Conclusion

The study of current solutions in Web information integration has lead to a number of findings:

- Scalability – in contrast to the data warehousing approach, the wrapper-mediator approach offers higher scalability and is more capable of handling large-scale information integration. Among various wrapper-mediator solutions, the GaV approach offers limited scalablility, as changes occurring at local sources will lead to the re-generation of the global schema. In the LaV approach, the global schema becomes a major obstacle, as the global schema needs to cover concepts appearing in all local source domains, which is often hard to achieve in real-world cases. In contrast, the distributed integration approach does not require any forms of global agreements or schemas. Instead, they encourage people to build local, pair-wised agreements among a small set of neighbourhood peers. The local alignments are used to form a global data sharing network. Therefore, the distributed approach offers high scalability for large-scale information integration.

- Flexibility – in most centralized solutions, local data sources are added to the integration system in a controlled manner, thus offering limited flexibility for local data source owners. In the distributed approach, data sources come and leave in a self-organized and self-interested fashion, and the joining or leaving of local data sources do not affect the overall performance of the data sharing network. Hence, the distributed approach offers higher flexibility for local data source owners, as they have more control over their own data sources.

- Efficiency – despite the large number of effort towards automating the process of information integration, the majority of the developed solutions are still complex and have limited code reusability. Some solutions even require complex programmatic setup and configuration. To improve the efficiency of current integration solutions, tasks such as schema mapping and data transformation need to be automated and simplified. Current solutions in the centralized integration field mainly focus on the improvement of the query reformulation techniques, whereas the distributed solutions more emphasize on the tasks of schema mapping and data transformation. In addition, query execution in the centralized integration approach is simple, as only one query reformulation step is involved for each service request. But the distributed approach requires an iterative number of query translation steps. Hence, the centralized approach is more efficient in handling query requests.

- Accuracy – accuracy of both the centralized approach and the distributed approach are reliant on the schema mapping. In the distributed integration approach, mapping inconsistency issues occur when there is a semantic conflict between two mapping alignments. This can happen as there is no global agreement defined to enforce semantic consistency, thus an extra task is required to detect mapping inconsistencies (Cudre-Mauroux *et al.*, 2006). In addition, schema mappings defined in the centralized approach only support single direction data translation, whereas in the distributed P2P approach, the defined mapping must support bi-directional data transformation. Hence, schema mapping in the distributed approach is more restricted than the centralized approach.

In the next chapter, we introduce a proposed information integration solution. The proposed solution is motivated by both the centralized and the distributed integration approach. On one hand, we leverage the benefits offered by the centralized integration approach, and use ontology as a mediator for enabling large-scale information integration. On the other hand, we use the self-organized information sharing principle introduced by the distributed approach, and we allow information owners to join and leave the information sharing network in a self-controlled fashion.

2.4 Related Work on Semantic Web

In previous literature review sections, we compared some of the typical solutions proposed for large scale information integration, and we uncovered the strengths and limitations of each compared integration approach. Inspired by the finding, we come up with a new approach for solving the information integration problem, that is, to combine the distributed P2P network architecture with the centralized wrapper-mediator based query processing approach. In this section, we compare our research work with some of the Semantic Web query initiatives. Since the Semantic Web is a vast research domain, thus we narrow our focus down to the following research fields: e-Tourism ontology efforts, Semantic Web query researches, and community based searching researches.

2.4.1 E-Tourism Ontology

In the past few years, a number of ontologies have been developed for the e-tourism industry, including Harmonise (Fodor & Werther, 2005), QALL-ME Ontology (Ou *et al.*, 2008), Hi-Touch (Mondeca, 2004), DERI OnTour (DERI, 2009), EON Traveling Ontology (EON, 2009). Some of those works are led by industry, and others are academia initiatives. Some uses agent-based technologies, and some use web application technologies. Regardless of the technological differences used by those projects, they have one thing in common, that is to enable seamless data exchange in the tourism industry. Another commonality among those initiatives is that they all rely on the current Semantic Web technology.

Harmonise Ontology

The Harmonise ontology (Fodor & Werther, 2005) is developed by the harmonise project. The main objective of the Harmonise project is to create an international network to enable the seamless data exchange of travel related information in the tourism industry. A framework is proposed called the Harmonise Netowork for the Exchange of Travel and Tourism Information (HarmoNET). Members of the network can share data by mapping their specific data model to the Harmonise ontology. RDFS is used as the standard ontology language for modelling and documenting the Harmonise ontology.

QALL-ME Ontology

Unlike the Harmonise ontology, the QALL-ME initiative (Ou *et al.*, 2008) offers user a set of tools for translating queries from natural languages into a standard ontology query language. The developed tools act as the seamless gateway for translating queries between their own local formats and the global ontology format. The QALL-ME ontology is encoded using OWL-DL, which also supports simple reasoning on description logic.

Hi-Touch Ontology

The Hi-Touch ontology (Mondeca, 2004) was developed to formalize knowledge on travelers expectations and to propose customized tourism products. The ontology is encoded in OWL, and it provides classification on the tourist objects, which as linked together by semantic relationships. In the Hi-Touch ontology, each tourism object falls into one of three classes, documents, objects, and publications. The terminologies defined in the Hi-Touch ontology ensures the consistency between the distributed local databases, thus enable data sharing between those databases.

Other Works

The DERI e-Tourism ontology (DERI, 2009) was developed by STI Innsbruck. The ontology focuses on the description of accommodations and infrastructure and enables a user, who queries a tourism portal to find a package of relevant accommodations and infrastructure. The EON Travelling Ontology (EON, 2009) was developed by the Institute National de lAudiovisuel in France. It describes tourism concepts that are divided into temporal entities (e.g., reservations) and spatial entities, which further comprise dynamic artefacts (e.g., means of transportation) as well as static artefacts which comprise town sights or lodging facilities.

Discussion

All the above mentioned research efforts aim to create and convey a standard ontology schema to enable data sharing in the tourism industry. Whilst each ontology effort provides a comprehensive set of terminologies and vocabularies, they often contain a diversified mixture of concepts, thus making ontology mediation a difficult task. In addition, enforcing a global ontology schema among a vast number of data sources is always a challenging and time consuming task. In comparison to our research work, we focus on reaching global consensus by establishing local agreement between peers and using peer-to-peer architecture. Global consensus is established via a network of local consensus between neighbourhood schema mappings. We believe that our approach is more scalable and flexible for establish large scale schema consensus, thus easy and simply for ontology mediation.

2.4.2 Semantic Web Query

Several recent research studies also look at the problem of distributed data integration using Semantic Web technologies. Two well-known projects in this domain are the DARQ project (Quilitz & Leser, 2008), and the SemWIQ (Langegger *et al.*, 2008) project. Other researches such as (Calvanese *et al.*, 2008) provide algorithms for processing distributed queries using a centralized global schema. In this section, we provide an in-depth review on some of the recent research works, and discuss the differences between our research and their research works.

DARQ

The DARQ project (Quilitz & Leser, 2008) solves the query formulation problem faced by many distributed RDF data sources. It uses the SPARQL query language, which is a W3C recommendation for query RDF semantic data sources. The DARQ solution provides transparent query access to multiple SPARQL services, where distributed RDF graphs do not need to be loaded into the same machine for query processing. The global query is decomposed into a number of sub-queries, where each sub-query can be answer by an individual SPARQL service. DARQ also offers query rewriting as well as cost-based query optimization to speed-up query execution. One limitation with the DARQ solution is that, it only supports queries with bound predicates, unions and left-outer-joins operations are not supported.

SemWIQ

Similar to the DARQ solution, the SemWIQ system (Langegger *et al.*, 2008) provides a federated query platform for querying distributed RDF data sources. Data sources can be registered and abandoned freely, and all RDF schema or OWL vocabularies can be used to describe their data. In SemWIQ, a query does not directly refer to actual endpoints, instead it contains graph patterns to virtual data set, thus providing a transparent on-the-fly view to the end-user.

Other Works

Calvanese et al. (Calvanese *et al.*, 2008) present an algorithm for answering queries submitted over a data integration system. It follows the GAV approach and assumes that the views associated to the elements of the mediated schema are sound. However, in this approach, the query processing is more complex that in traditional GAV systems, as the presence of integrity constraints in the mediated schema, implies in the need of taking the semantics of such constraints into account during query execution. This algorithm uses integrity constraints and mappings for, respectively, inferring additional information in the query (query expansion) and rewriting the query over the sources. This way, extracting information in this approach is similar to query answering with incomplete information, which is a difficult task.

Discussion

Whilst the DARQ and SemWIQ projects provide solution for querying distributed semantic data sources, they mainly deal with RDF data sources. In our research, we emphasis on XML data sources, and provide OWL/RDF wrappers on the registered XML data sources. In addition, the DARQ project, SemWIQ project and Calvaneses work use centralized integration architecture, where one global schema is used for query rewriting and decomposition. Our research, on the other hand, uses distributed integration structure. We use the concept of data source community, where data sharing within each community follows the centralized structure, and data sharing between different data communities follows the mediated approach.

2.4.3 Community based Query

Community based query resolving, sometimes also known as collaborative query resolving using Peer-to-Peer network, is a comparatively new research field. Several attempts were made to achieve distributed data sharing. Comito et.al (Comito *et al.*, 2010) proposed a new search technique known as Adaptive Path Selection (APS) to optimize query resolving on distributed XML data sources. In their solution, a multi-path XML query is resolved by querying either the most selective path or the whole path set based on the selectivity of the paths in that query. In contrast to the conventional approaches, the network traffic generated by their strategy is significantly less. However, their solution relies heavily on the selectivity of each query path, and this value is dynamic in some cases, thus causing unpredictable performance issues.

SWAPSTER

SWAPSTER is a prototype system developed by the SWAP project (Broekstra *et al.*, 2003). The SWAP project is a Semantic Web initiative that aims to combine the Semantic Web languages for knowledge representation with P2P knowledge sharing for distributed knowledge management. SWAPSTER allows the sharing of knowledge between distributed users in a P2P fashion. It uses a routing algorithm called REMINDIN' to efficiently route complex queries in dynamic P2P networks. The routing algorithm is built on a number of social metaphors, which are translated into a number of semantic overlay networks for efficient information discovery. The focus of SWAPSTER is different from the PDMS approaches such as PeerDB and Piazza, as it mainly focuses on the correct and efficient routing of query messages, and the discovery of knowledge information.

Chapter 3

A Semantic Integration Framework

In the previous chapter, we have reviewed some current solutions for Web information integration, and we found that both the centralized integration approach and the distributed integration approach have their own strengths and limitations. Inspired by previous works, we introduce an ontology-based semantic integration framework. By combining the strengths of various integration solutions, we aim to provide an effective, flexible and scalable solution for large-scale information integration. Our research mainly focuses on the large, dynamic and distributed information integration domains (such as the online accommodation domain). The major parts of this chapter are organized as follows. First, we introduce the basic ideas and design principles used to create the framework. The overall structure as well as the basic elements defined in the framework are presented in Section 3.2. The proposed framework aims to solve the problem of information integration on three levels of abstraction, including the data level, the process level and the architecture level. Section 3.3 shows the techniques and methods developed on the data level, and we look at the role of ontology for solving the data heterogeneity problem. Section 3.4 shows the process level solutions, and we decompose the information integration process to three independent sub-processes. Section 3.5 shows a hybrid architecture for large-scale information integration, followed by the conclusion in Section 3.6.

3.1 Combining the existing integration approaches

In Chapter 1, we have introduced some of the challenges of information integration in the online accommodation domain. From previous research studies, we learnt that the online accommodation domain is a highly distributed place, where a large number of distributed and heterogeneous data sources exist, and some of those data sources are constantly changing. Hence, to cope with those challenges, an effective information integration solution must satisfy the following list of requirements:

- Integrated – to cope with the distributed nature of the online accommodation domain, the developed solution must provide integrated functionality with high data coherence and consistency.
- Good Performance – the performance of the developed solution is reflected in two dimensions: accuracy and efficiency, and both properties are crucial for the online accommodation information integration. An effective integration solution should provide both high efficiency and high accuracy.
- Flexible– the developed solution should offer a sufficient amount of flexibility to cope with any changes that may happen in the information integration process. More specifically, the solution should allow the joining and leaving of individual data sources according to their own preferences. The joining or leaving of any data source should not affect the overall performance of the system.

- Scalable – the developed solution should be able to integrate an increasing number of data sources, and it should be expandable to cope with information integration in a large scale.

To ensure the integrity of the developed integration solution, we base our solution on the wrapper-mediator approach, and use ontology as a mediator to enable integrated information access. Performance of the developed solution is improved by simplifying and automating the integration process. We first break the integration process into several small tasks, and then build techniques and tools to assist each task. The modification of the integration process also gives a larger amount of freedom to the data source owners, thus providing higher flexiblity. In addition, we combine the centralized architecture with the distributed architecture to form a hybrid information integration architecture; the hybrid integration structure is beneficial for dealing with large-scale information integration.

3.1.1 Ontology and Data Heterogeneity

In Chapter 2, we learned that the wrapper-mediator approach provides higher coherence and data consistency for information integration, as data is directly fetched from local data sources at system run-time. In addition, ontology has proven its ability in respect to solving the data heterogeneity problem (Antoniou & Harmelen, 2004). Inspired by those works, we combine the wrapper-mediator approach with the utilization of ontology, and propose an ontology mediated approach for solving the problem of data heterogeneity. Like the wrapper-mediator approach, the proposed approach uses ontology as a mediator to provide integrated access to a group of heterogeneous data sources, and individual data sources are connected to the mediator via schema mappings. The proposed approach is further discussed in Section 3.3.

3.1.2 Integration Process

On the process level, we change the process of information integration into three sequential tasks. Since data sources in the online accommodation domain come and go in a dynamic and unpredictable way, it is often difficult to manage data sources in a controlled manner. Thus, we decide to allow individual data sources to join or leave the information sharing network in a self-organized and interest-driven fashion, meaning that each individual source can join or leave the network at anytime based on its own choice. This feature not only reduces the effort of data collection and information acquisition, it also offers more freedom to the information providers. Data source owners now have more control over their own data sources. To give an illustrative example, we use a hotel rate sharing scenario from the online accommodation domain. A hotel rate sharing system is often connected with hundreds of hotel rate sources provided by different hoteliers. In traditional cases, the system is managed by a middle agent known as the hotel intermediary. The role of the intermediary is to collect hotel information from local data sources scattered on the Web, and this involves a number of tasks such as the collection of the data sources and the maintenance of the established connections. This approach has several limitations: 1.) it is effortless to collect a large number of useful data sources on the Web, and the maintenance as well as the monitoring of the collected data sources also require significant efforts. Hence it is difficult to handle such a task by one single party, especially when the number of integrated data sources is large and highly dynamic. 2.) the joining or leaving of local data sources are heavily controlled by the intermediary, and individual hoteliers have less control over their own data sources, thus changes occurring at the local level can not be immediately reflected to the integration system. For those reasons, we decide to allow individual sources to join or leave the information sharing network depending on their own interests, and the management of local to global relationships are also performed by local source owners. As a result, the information integration process is divided into three sequential sub-processes, called publish, combine, and use. Each process involves a particular type of user, and the process distributes the efforts of information integration to various type of users.

3.1.3 Integration Architecture

Since the online accommodation domain is so large, it is impractical and difficult to enforce one global schema over a large number of data sources. Therefore, reaching a global agreement across all information providers is hard. To solve this problem, we adopt the bottom-up semantic building technique introduced in the distributed integration approach, and we allocate the task of designing and enforcing a global schema to small community groups. Each community group is formed by a small group of data sources, and each formed group uses its own global schema for information sharing. The bottom-up agreement building approach simplifies the task of global schema definition, as each schema now only needs to cover a smaller set of terms, concepts and relations, thus is easier to define. In addition, the efforts required for maintaining and enforcing the defined schema are also reduced, as each schema is shared by a smaller group of users, it is therefore easier to reach an agreement between those users.

In order to build a global agreement among all community groups, schemas created by the local groups are connected following the distributed approach. Each community group is linked to one or other community group to achieve larger scale information sharing, and such structure is beneficial for dealing with information integration on an increasing scale. Community groups are interconnected via schema mappings built between their global schemas, and such structure follows the distributed integration architecture.

On the other hand, information sharing in each community group is achieved via the centralized integration approach. Local data sources are connected to the global schema via local to global schema mappings, and individual data sources can join the community group by relating its schema to the community group's global schema. We describe the combined architecture as a hybrid integration approach.

All the above mentioned concepts provide the basic ideas and design principles used by our semantic integration framework. Since information integration in the proposed framework is achieved by establishing a semantically interoperable network, we name the proposed framework as the semantic integration framework. The following section gives an overview of the proposed framework.

3.2 Overview of the semantic integration framework

We start by introducing the basic elements and concepts defined in the proposed framework. Figure 3.1 gives an overview of the semantic integration framework. The framework consists of several key elements and concepts including *Peer*, *Hub*, *Community Group*, *Neighbour*, *Local Schema*, *Global Schema*, *Schema Mapping*, *Provider*, *Operator*, and *User*.

- Peer – a peer is a data/information source that contains data stored in structured and semi-structured formats. It is the fundamental element that forms the integration framework. In this thesis, we use the words *Peer*, *Data Source* and *Information Source* interchangeably to refer to the individual data sources in the system of accommodation information integration.

- Hub – a hub is a local, centralized mediator that provides integrated, unified information access to a group of peers. Each hub defines and maintains its own global schema, and is connected to a number of peers via schema mappings. Hubs are also inter-connected via schema mappings to form a large, integrated information sharing network.

- Community Group – the entity formed by a hub and its connecting peers is called a community group. A community group can be considered as a collection of peers with common interest, where a hub acts as an access interface to its connecting peers.

- Neighbours – two inter-connected community groups are called neighbours. Neighbours are connected via concept mappings defined at a schema level, and the establishment of neighbours is based on semantic similarities and relationships.

- Local Schema – the data model used by a peer is called a local schema. In this research, the local schema can be considered as the XML schema in general. The local schema is stored at the peer side.

- Global Schema – the data model used by a community group is called a global schema. Since ontology is used as the data model for information sharing in a community group, global schema mainly refers to the ontology used by a particular community group. The global schema is stored at the hub side.

- Schema Mapping – the set of associations defined between a pair of data models is called a schema mapping. In the proposed framework, schema mappings are defined between peers and hubs, as well as between hubs

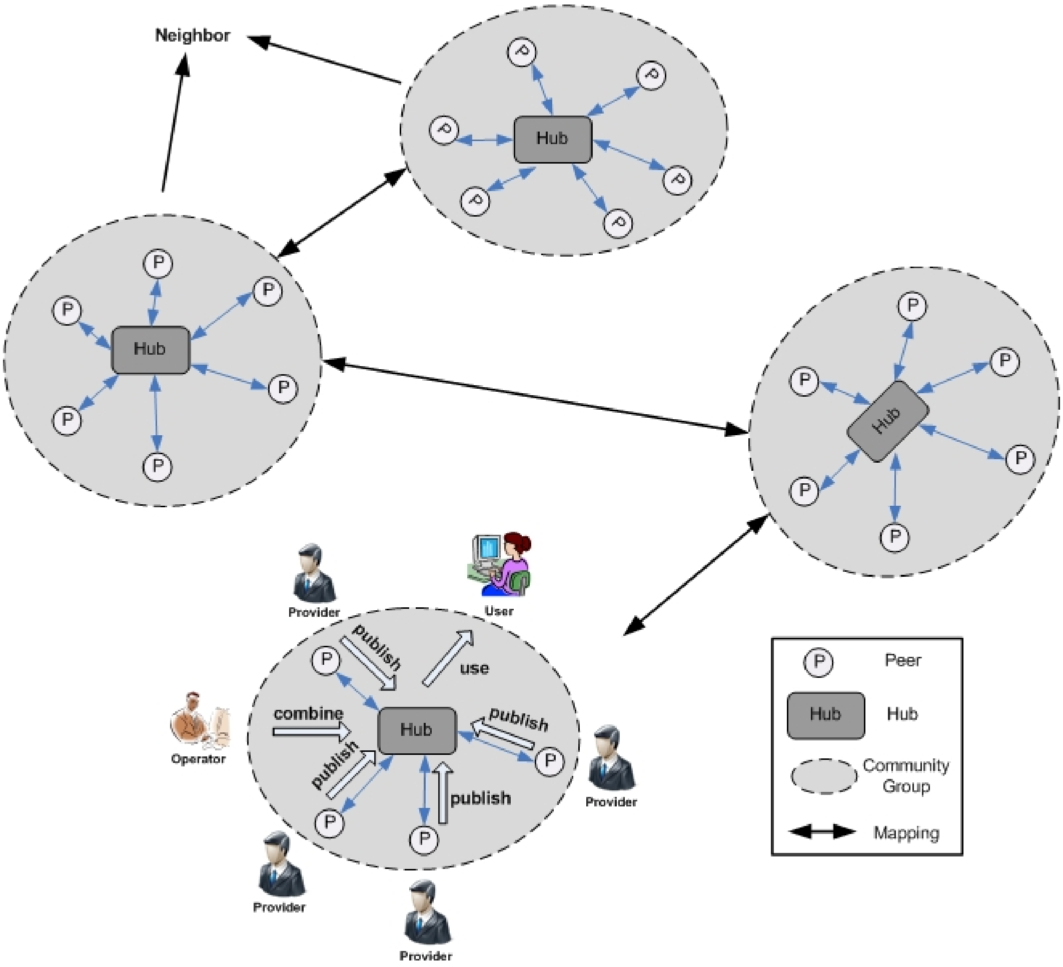


Figure 3.1: The semantic integration framework

and hubs. The goal is to create a semantically interoperable network for information sharing.

- Provider – the owner of a peer is called a provider. Providers can select the community group they wish to join, and the joining of a peer to the selected community group is achieved via schema mapping.
- Operator – an operator handles the basic operational tasks required for running a hub, and he/she is also responsible for the definition and maintenance of the global ontology.
- User – in the proposed framework, people who make use of the integrated information source are called users. Users can query over the integrated information source, and in the online accommodation industry, users can be generally considered as online consumers.

Having defined the basic concepts and elements, we now introduce the proposed framework, and we present the framework in three dimensions: semantic interoperability, integration process, and hybrid integration architecture.

3.2.1 Semantic Interoperability

Semantic interoperability in the proposed framework is achieved by establishing a semantically interoperable network that connects together the group of heterogeneous data sources. In this thesis, we name the estalished semantic network as the *semantic layer* of the proposed framework. The semantic layer of the proposed framework provides an integrated conceptual view over the entire information sharing network. On a semantic level, members of the information sharing network are connected via the conceptual mappings defined on the schema level. In each community group, the schemas defined from individual data sources are mapped to the global ontology shared within the community group. At the same time, the global ontologies shared within each community group are also interconnected to enable information sharing among them. In the proposed framework, ontology is mainly used in three ways: domain modeling, schema mapping, and data translation. First, ontology is used as a shared conceptual model over a group of heterogeneous data sources, in this case, accommodation providers. Second, it is used as a modeling technique for representing concept mappings defined between a pair of schemas. Last, the defined mapping is used for data translation among heterogeneous formats. Section 3.3 further explains the role of ontology in achieving semantic interoperability of the proposed framework.

3.2.2 Publish-Combine-Use Integration Process

The proposed framework provides a three step information integration process, including *publish*, *combine*, and *use*. Each step contains some tasks defined for the online information sharing domain. During the *publish* phase, information providers select on the information hub they wish to join, and then publish their data sources to the selected information hub. The publishing of a data source to the selected hub is achieved by mapping the source schema to the global ontology shared in the hub. The publication process formally makes the data source available to public users. During the *combine* phase, hub operators define a number of query interfaces to provide users with the facilities for querying over the published information sources. Each defined query interface provides integrated access to a number of published data sources. Finally, during the *use* phase, information users query the published data sources via the pre-defined query interfaces. The publish-combine-use process provides a flexible integration solution to allow the involvement of all information parties in the information integration process.

3.2.3 Hybrid Integration Architecture

The architecture of the semantically interoperable network is formed by combining the centralized integration architecture with the distributed architecture. We name the formed integration architecture as a hybrid integration architecture. Each community group uses a centralized integration architecture, and neighbourhood community groups are connected in a distributed P2P fashion. The hybrid architecture also enables the collaboration between neighbourhood community groups for achieving large-scale information sharing. The hybrid information architecture also allows collaborative query processing in the proposed framework. The process of query processing involves tasks such as query generation, query execution, and query translation. Query processing within each community group is performed following a bottom-up approach, where information from local data formats are firstly transformed into the generic ontology format, and then integrated together to solve the issued query. Query processing among neighbourhood community groups is based on the translation of query messages among different formats. All query level tasks are dependent on the conceptual mappings defined on the semantic level.

3.3 Ontology and Semantic Interoperability

A semantic network is formed to enable the semantic interoperability of the proposed integration framework. The semantic network is established by mapping corresponding concepts defined from heterogeneous source schemas. In this thesis, we name the established semantic network as the *semantic layer* of the proposed framework. The *semantic layer* provides an integrated conceptual view over the entire information sharing network, and it is the foundation for performing various integration tasks in the integration framework.

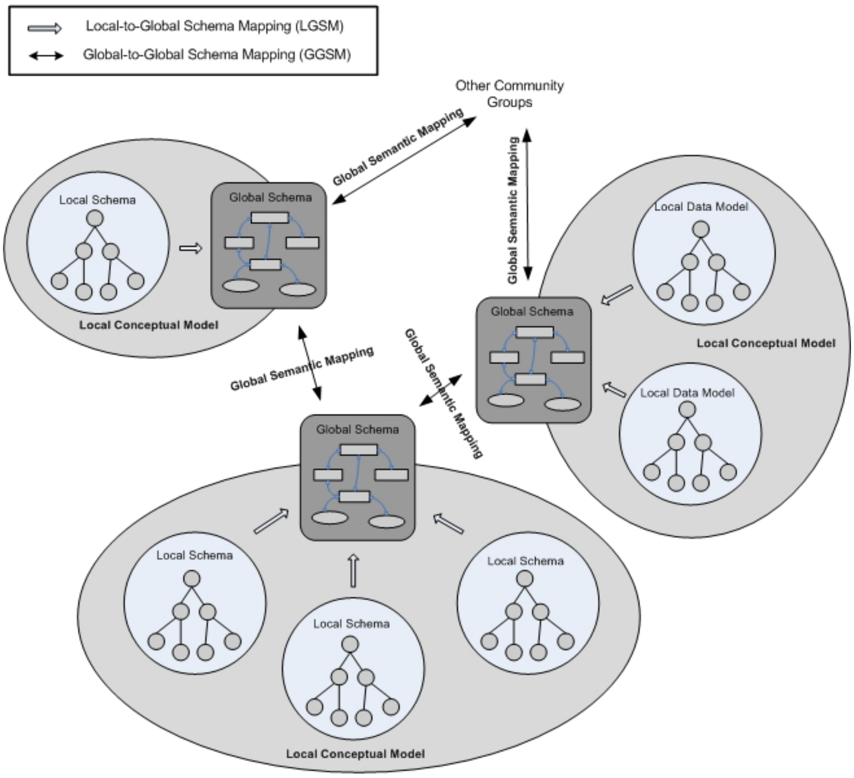


Figure 3.2: Schema mapping and semantic interoperability

3.3.1 Semantic Layer

The semantic layer consists of two parts: *Local Conceptual Model*, and *Global Semantic Mapping*. Figure 3.2 gives an overview on the semantic layer of the proposed framework. As shown in the diagram, each *Local Conceptual Model* is comprised of three parts: local source schema, global schema and schema mapping.

- Local Source Schema – local source schema refers to the data model used by each local data source. Since our research mainly focus on the integration of XML-based data sources, where each data source is compliant to a predefined XML schema, thus the local source schema in our case mainly refers to the pre-defined XML schema.

- Global Schema – global schema refers to the shared data model used by a community group. Our research uses ontology as the global data model to provide a shared understanding for a particular domain. Hence, the global data model in this research mainly refers to the global ontologies.

- Schema Mapping – concepts from each local source schema are mapped to the concepts from the global schema. The defined schema mapping follows the centralized integration approach, where local schemas are defined as views over the global schema. In this research, the process of relating concepts defined from the local schema to the global schema shared by the community is called *LGSM (Local-to-Global Schema Mapping)*. In the proposed framework, LGSMs are performed by the data source owners instead of the hub operators, and the goal is to reduce the efforts required for creating and maintaining the large numbers of mappings. The definition of LGSM allows the integrated and unified information access to all peers within a community group.

At the same time, local conceptual models are inter-connected via *Global Semantic Mappings*, and the goal is to enable information sharing across different community groups. As described earlier, each community hub defines and maintains its own global schema, the defined schema is associated to a number of similar schemas maintained by its neighbourhood hubs. The process of mapping concepts among neighbourhood schemas is referred to as *GGSM (Global-to-Global Schema Mapping)*. Different from the LGSM, the GGSM is performed by hub operators, and is defined following the distributed architecture.

Both LGSM and GGSM are the backbones of the information sharing network, and the accuracy of the defined semantic mappings directly affects the performance of the formed information sharing network. Hence, to assist the semantic mapping processes, a number of techniques and methods are developed. The developed techniques and methods will be introduced in Chapter 4, other information on the semantic layer is also covered in Chapter 4.

3.3.2 Ontology

Here, we will discuss the utilization of ontology in the proposed framework. In particular we look at the modeling and reasoning powers of ontology and how they are used to achieve online information integration.

In recent years, ontology has been widely used by various research communities and industry groups to facilitate information sharing and knowledge representation (Kim, 2000). In the context of information integration, ontology can be considered as a shared understanding of a specific domain, which is comprised of collections of agreed concepts, relations, axioms, and instances. In some cases, ontology is used as a shared concept hierarchy to facilitate information integration (Wache *et al.*, 2001). In other cases, it is used as a knowledge base to assist decision making or other types of knowledge-based reasoning (Davies *et al.*, 2002).

In this research, we leverage both the modeling and reasoning powers of ontology for online information integration. On one hand, ontology is used as a data model providing concepts and relations defined for the online accommodation domain. On the other hand, it is used as an application model assisting the mapping between the local XML schema and the global ontology.

3.3.2.1 Data Model

As mentioned in earlier sections, each community group adopts the centralized integration structure, where a global data model is defined to provide mediated information access to a number of heterogeneous data sources. To implement this structure, a data modeling technique must be selected to create the global data model and to provide a shared understanding on the same domain. In order to facilitate large-scale online information sharing, the selected modeling technique must satisfy the following requirements:

- Simple and Easy – the selected modeling technique should be simple and easy to use, and the creation of a global data model should be performed easily by users who have limited data modeling knowledge. This criteria is assessed by examining the current data modeling techniques and tools.

- Object-Oriented – the object-oriented paradigm has long been used for data modeling in large, complex environments such as the distributed data integration (Worboys & Maguire, 1990). Previous studies (e.g. Lecluse *et al.*, 1988) have also demonstrated the ability of the object-oriented paradigm in complex data modeling. Hence, our data modeling technique also follows the object-oriented paradigm, where collections of concepts, properties and relationships are defined.

- Accessible and Exchangeable – since the constructed data model is not only shared within the same community group, but also shared across different community groups, the data model must be accessible and exchangeable in the entire information sharing network. This feature is also known as the online accessibility of the developed data model.
- Reasoning – finally, the developed data model should provide logic to allow simple reasoning, and to assist the query answering process.

In order to select an optimal data modeling technique, we compare some existing data modeling techniques to examine their abilities in fulfilling the above mentioned requirements. The compared techniques include OEM (Chawathe *et al.*, 1994), XML schema (Manolescu *et al.*, 2000), UML (Halpin & Halpin, 1999) and ontology (Cruz *et al.*, 2004). The following table shows the results obtained from our comparison process.

Table 3.1: Data model comparison

	OEM	**XML Schema**	**UML**	**Ontology**
Simple and Easy	No	Tools provided	Tools provided	Tools provided
Object-Oriented	Yes	No	Yes	Yes
Accessible and Exchangeable	No Web-based access	Can be easily accessed and exchanged	No Web-based access	Can be easily accessed and exchanged
Reasoning	First Order Logic	No	No	Description Logic (Baader *et al.*, 2001) / First Order Logic (Qg, 1997)

From the results, it is clear that ontology outperforms other techniques in domain modeling. Hence, we choose to use ontology as the data model for the developed semantic integration framework. In addition, we choose to use OWL

(McGuinness & van Harmelen, 2003) as the language for representing our ontology.

The selection of OWL as the ontology language is based on two reasons: 1.) In comparing to some other languages such as OIL+DAML (Wroe *et al.*, 2003), OWL is written in XML format, so it inherits all the benefits provided by XML. It allows information to be exchanged easily across different platforms. On top of that, it also allows the exchange to be taken place between different applications. OWL also captures information relating to classes, properties, attributes as well as the constraints displayed by UML and OCL together. 2.) In contrast to RDF (Manola & Miller, 2004), OWL allows more complex relationship structure such as inheritance of properties or transition, thus providing more reasoning capability. Figure 3.3 shows a partial hotel ontology written in OWL.

For example, by just considering all the classes that come within the Hotel domain, we would have the following defined OWL classes:

```
<owl:Class rdf:about =“#Hotel” />

<owl:Class rdf:ID =“Address” />

<owl:Class rdf:ID =“Room” />

<owl:Class rdf:ID =“Price” />
```

A relationship between two related classes is defined as ObjectProperty, for example, the relationship “locateAt” describes the location of a hotel and we have expressed this in OWL as:

```
<owl:ObjectProperty rdf:about =“#locateAt” >

    <rdfs:range rdf:resource=“#Address” />

    <rdfs:domain rdf:resource=“#Hotel” />

</owl:ObjectProperty>
```

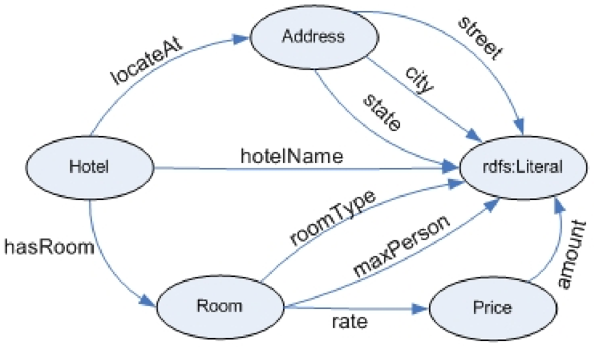
A relationship between a class and its property is defined in OWL through DatatypeProperty. For example, “hotelName” is a property of class “Hotel” with data type “Literal” and it can be expressed as:

```
<owl:DatatypeProperty rdf:about =“#hotelName” >
    <rdfs:range rdf:resource=“rdfs#Literal” />
    <rdfs:domain rdf:resource=“#Hotel” />
</owl:DatatypeProperty>
```

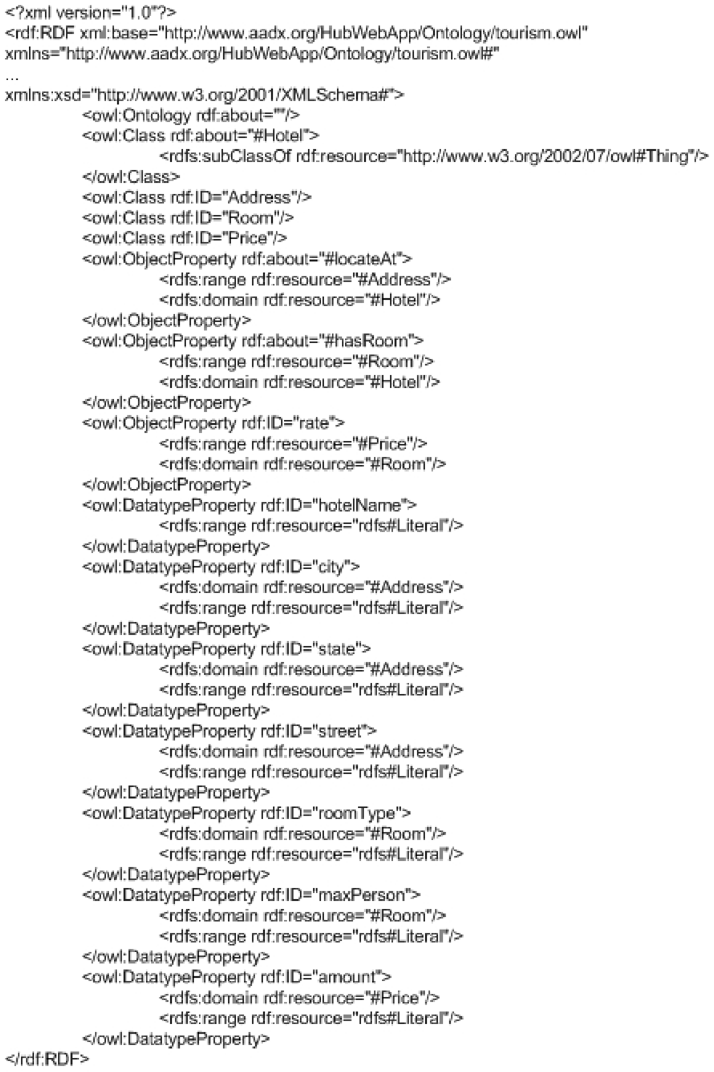
Ontology provides the fundamental principle in describing concepts as well as their relationships. It provides a shared understanding of a specific domain to solve the issue of semantic heterogeneity. It also acts as a mediator for reconciling different data sources. In Chapter 4, we introduce an ontology-based transformation approach that is developed to assist the exchange of heterogeneous information across a large numbers of data sources.

3.3.2.2 Schema Mapping

In the proposed framework, ontology is also used to facilitate schema mappings. In particular, it is used to model the conceptual alignments defined between a local XML schema and a global ontology. Referring back to the previous sections, semantic relationships are defined between the local data models and the global data model to enable semantic interoperability. In the case of online information integration, the local data model refers to the local XML schema, whereas the global data model refers to the global ontology. Hence, to define a local to global schema mapping, concepts defined from the local XML schema are aligned to their corresponding concepts defined from the global ontology. This process is known as the LGSM.



A.) Ontology Graph



```
<?xml version="1.0"?>
<rdf:RDF xml:base="http://www.aadx.org/HubWebApp/Ontology/tourism.owl"
xmlns="http://www.aadx.org/HubWebApp/Ontology/tourism.owl#"
...
xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
        <owl:Ontology rdf:about=""/>
        <owl:Class rdf:about="#Hotel">
                <rdfs:subClassOf rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
        </owl:Class>
        <owl:Class rdf:ID="Address"/>
        <owl:Class rdf:ID="Room"/>
        <owl:Class rdf:ID="Price"/>
        <owl:ObjectProperty rdf:about="#locateAt">
                <rdfs:range rdf:resource="#Address"/>
                <rdfs:domain rdf:resource="#Hotel"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:about="#hasRoom">
                <rdfs:range rdf:resource="#Room"/>
                <rdfs:domain rdf:resource="#Hotel"/>
        </owl:ObjectProperty>
        <owl:ObjectProperty rdf:ID="rate">
                <rdfs:range rdf:resource="#Price"/>
                <rdfs:domain rdf:resource="#Room"/>
        </owl:ObjectProperty>
        <owl:DatatypeProperty rdf:ID="hotelName">
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="city">
                <rdfs:domain rdf:resource="#Address"/>
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="state">
                <rdfs:domain rdf:resource="#Address"/>
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="street">
                <rdfs:domain rdf:resource="#Address"/>
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="roomType">
                <rdfs:domain rdf:resource="#Room"/>
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="maxPerson">
                <rdfs:domain rdf:resource="#Room"/>
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
        <owl:DatatypeProperty rdf:ID="amount">
                <rdfs:domain rdf:resource="#Price"/>
                <rdfs:range rdf:resource="rdfs#Literal"/>
        </owl:DatatypeProperty>
</rdf:RDF>
```

B.) Ontology written in OWL

Figure 3.3: Hotel domain ontology

However, we found that some of the structures defined in the XML schema such as the nesting structure and element order can not be fully represented using ontology. This is due to the fundamental differences between XML schema and ontology, thus structural information will be lost during the data translation phase in which data from source XML format is translated to its corresponding local ontology format. To overcome this issue, and to capture the unique information defined by both the XML schema and the ontology, we propose a mapping ontology to represent the concept mappings defined between a XML schema and an existing ontology. The primary goal is to use the captured concept mapping information for compensating information loss that has occurred during single trip data translation. The mapping ontology and the ontology-based data transformation is introduced in Chapter 4. For now, we know that ontology is also used as an application model for representing mappings defined between XML schema and ontology. The use of ontology for representing concept mappings was initially proposed by the Hamo-ten project (Fodor *et al.*, 2005), where a semantic bridge is developed for storing mappings defined between a pair of ontologies. Different from their solution, our mapping ontology focuses on the mapping defined between an XML schema and an ontology.

3.3.2.3 Data Mediation

Finally, ontology is used as a mediator for enabling seamless data transformation among heterogeneous data formats. As our research mainly focuses on the integration of XML-based data sources, ontology is therefore used as a mediator for translating XML-based data. Figure 3.4 shows the general structure of the ontology mediated XML transformation. The entire transformation process can be generally divided into two phases: *Concept Mapping* and *Data Translation*. *Concept Mapping* focuses on the definition of the concepts and initiates the map-

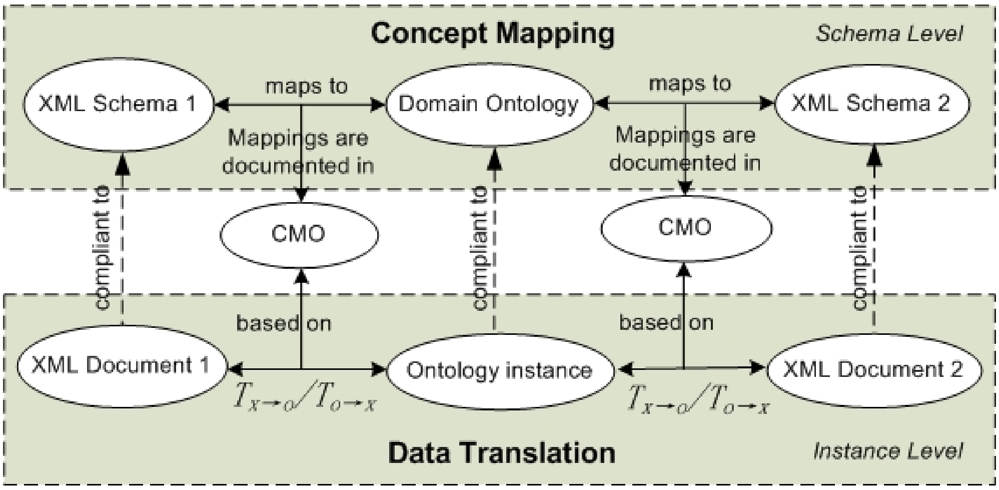


Figure 3.4: ontology mediated XML transformation

ping between the local XML schemas and the global domain ontology. During *Concept Mapping*, local XML schemas are mapped to the global ontology. Data Translation focuses on the translation between different XML formats. During *Data Translation*, the input XML document is firstly transformed into the global ontology format, which will then be transformed into the targeted XML format. The proposed transformation method is further explained in Chapter 4.

3.4 Publish-Combine-Use Integration Process

In the proposed framework, we divide the process of information integration into three steps: publish, combine and use. The first two steps are mainly concerned with the establishment of the semantic information sharing network, whereas the last step deals with the utilization of the established semantic information sharing network. We name the modified integration process as the *process layer* of the proposed framework. Figure 3.5 shows the activities involved in the information integration process.

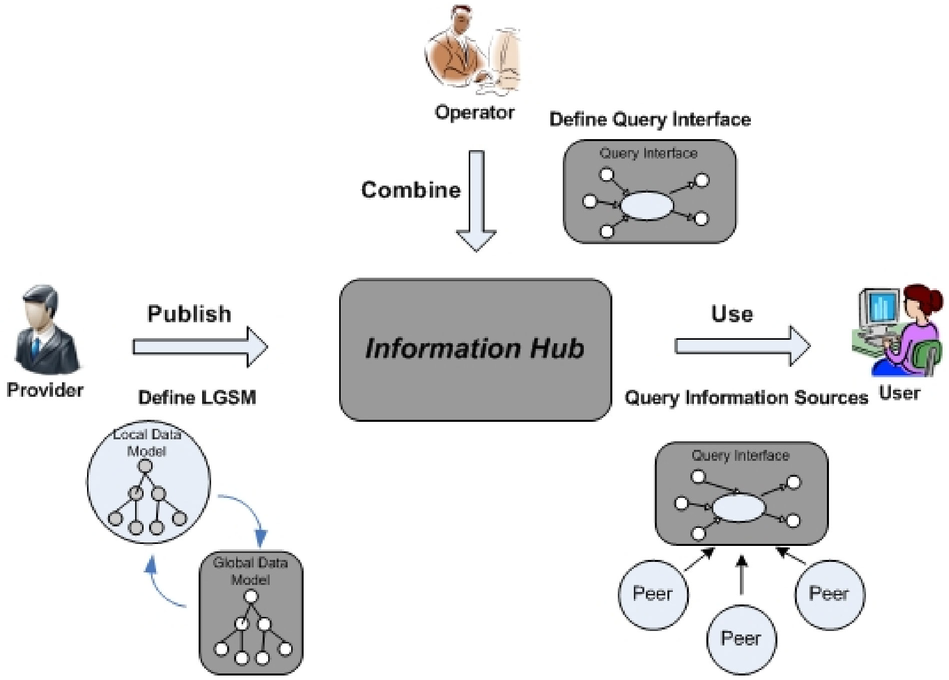


Figure 3.5: Information integration process

3.4.1 Publish

The first step of the modified information integration process is the publication of local data/information sources, so that the published sources can be easily located and used by the public users. In order to publish a data/information source on a chosen information hub, the data source owners need to relate their data source schema to the global schema defined at the hub side, and this task is known as the LGSM process.

Data source acquisition is the foundational task in most conventional information integration processes. In traditional cases, the acquisition process is often performed by the information operator who has total control of the entire integration process, thus the operator needs to have in-depth knowledge of all the acquired information sources, and the efforts required to collect those information sources are large. To solve those issues, we use a self-organized approach motivated by both the distributed approach (Broekstra *et al.*, 2003) and the publish-find-invoke mechanism adopted by the Web service architecture (Alonso & Casati, 2005). Instead of collecting the information sources by one or a small number of information operators, we encourage data source owners to publish their own data sources. The publishing of information sources by individual information providers offers higher flexibility and scalability for large-scale information integration. The publication process is supported by the community hub, where information providers can gain access to the hub, and submit new information sources to the hub. In addition, they can also perform basic management tasks to update or remove the published information sources. Information of the published sources is loaded into the hub for later query processing purposes. The involvement of information providers for publishing information sources drastically reduces the time and effort required for information source acquisition.

The key task involved in the publication process is the definition of local to global schema mappings. Again this is performed by individual information providers instead of the information operator, and the goal is to distribute the efforts required for creating a large number of schema mappings to data source owners. In addition, providers often have better knowledge of their own data sources, thus they are more capable of defining the correct schema mappings over their own data sources. The local to global schema mapping process is performed at the hub side, and the defined mappings are also stored by the community hub. To allow better query performance, we use the centralized integration approach for creating local to global schema mapping, where a predefined global data model is used. Concepts from local data models are associated with the concepts defined from the global data model. To assist the schema mapping process, a schema mapping tool is developed, and the developed tool is introduced in Chapter 6.

3.4.2 Combine

The primary goal of the combining process is to present the published information sources to the public users in well organised forms. To fulfil this goal, several tasks need to be performed: first, the set of published information sources need to be categorized into meaningful groups; second, a number of graphical user interfaces need to be defined to assist the query of the published information sources. The first task can be achieved by organizing information sources into information groups. An *information group* is a collection of information sources that are organized together based on their common properties. To allow users with no query knowledge to perform basic query operations over the integrated information sources, a query interface is defined. A query interface is a graphical user interface with predefined sets of parameters. As shown in figure 3.5, the definition of query interfaces is performed by hub operators. Each query interface is designed to perform a specific query task, and the designed query interfaces provide users with easy-to-use interfaces for querying over the set of integrated information sources. Hence, the combining process is further divided into two sub-process: 1.) the definition of information groups. 2.) the definition of query interfaces.

Since all peers can join the information sharing network in a self-organized fashion, it is unavoidable that the information provided by those sources would vary in some aspects. For example, one peer may provide hotel rate information in Sydney, whereas another peer may provide hotel rate information all around Australia. Although both peers provide similar information, the functions offered by those peers vary boardly. To search hotel information in Australia, it is obvious that the first peer can not be used for such a purpose. Therefore, to enhance the information quality of query interfaces, we group similar information sources into information groups, and organize the published information sources into proper categories. Similar information sources are grouped together. Continue on the previous example, to create a information group that provides information all around Australia, the first peer should be filtered out from the group, and the second peer should be selected. To facilitate the group of published information sources, each source is associated with a source description, describing its functionality and information.

In the query interface definition, hub operators decide on the set of input and output parameters used for a particular query. The initial goal is to provide a user friendly interface to users who have no or limited query language knowledge. The secondary goal is to enforce a set of input and output constraints to facilitate query processing. Take the online accommodation domain as an example, for a query interface that is designed to provide an unified set of hotel prices based on a given location, the query interface often needs to take the parameter "location" as input, and return the list of hotel prices containing "hotel name", "room type" and "price". Both the input and output parameters are selected from the global data model defined by the hub operator, and a semi-automatic tool is created to assist the query definition process. The developed query definition tool is also introduced in Chapter 6.

3.4.3 Use

The ultimate goal of query interface definition is to provide simple and easy-to-use query interfaces to users, so that users from non-IT background can perform complex query processing tasks. Using the pre-defined query interfaces, users can easily issue a query message without any query language knowledge, and the process is as simple as filling out a Web form. Continuing with the hotel rate example, to issue a query message that selects all hotel prices in Sydney, the user only needs to type in the value "Sydney" into the hotel location input field. A query message is then automatically issued to the hub. The issued query message is then processed following the steps explained in Section 3.5. Finally, results generated from the query are presented to the user in a predefined format. All the tasks involved in the information integration are performed at the hub side. Hence, the information hub provides a basic platform for enabling the collaboration among all palyers involved in the information integration process.

3.5 Hybrid Integration Architecture and Query Processing

The proposed semantic integration framework uses a hybrid integration architecture. On one hand, information integration in each community group is implemented following the centralized approach. Individual data/information sources are connected to the information hub using local to global schema alignments. As shown in figure 3.2, a schema mapping is defined between each pair of XML schema and the global ontology at that hub. On the other hand, information integration across neighbourhood community groups are implemented following the distributed approach. As previously mentioned, there are several benefits for connecting hubs following the distributed integration approach.

- It simplifies the task of ontology definition, as it does not require the enforcement of a global ontology across all the community groups. Each community group now uses a much smaller global ontology, which is easier to define and maintain.
- The distributed connection of information hubs offers higher flexiblity in coping with large-scale information integration. The distributed P2P structure can be easily expanded to cope with data source increase, it is also

flexible in handling data sources changes and system failure, as the failure of one information hub will not lead to the failure of the entire integration network.

Query processing in the proposed framework is collaboratively performed by a group of information peers and community groups. We name the set of methods and principles used for query processing as the *query layer*. Functions provided by the query layer include query message generation, query translation, data translation and query execution. All query related tasks are performed based on the information captured by the semantic layer. Query processing in the proposed semantic integration framework uses a bottom-up approach. The goal is to simplify the query rewriting process when dealing with large, complex query messages. Given a query Q, then the query resolution process can be defined as two iterative steps: 1.) The execution of query Q in a community group G_n. 2.) The propagation of query Q to G_n's neighbourhood groups G_{nb} when Q can not be resolved by G_n.

In the first step, when a query is received by a community group, the community's hub tries to answer such a query using its connecting peers, and the query answering process follows a bottom-up approach, where data from individual sources are translated into a common global format, and then merged into a single data set. The issued query is then executed against the merged data set to generate the final result. The benefit of the bottom-up query processing approach is that it simplifies the query rewriting process by translating data from heterogeneous local formats to a common global format.

If the user is not satisfied with the query results produced from the community group G_n, he can expand the query process to a number of community groups. In this case, the issued query is propagated to the neighbourhood community groups. When a query message Q travels from one community group to another, it is also translated from the source ontology format to the destination ontology format. This process is called query translation. Since query translation is based on the semantic mapping defined between a pair of global data models, thus the defined mapping must capture a sufficient amount of mapping information to support bi-directional query translation. For example, given two neighbourhood community groups C_1 and C_2, and a query message Q traveling between C_1 and C_2. To propagate the query Q from C_1 to C_2, Q needs to be translated from C_1 format to C_2 format. Likewise, to pass the query Q from C_2 to C_1, Q needs to be translated from C_2 format to C_1 format. Both translation processes are based on the semantic mapping defined between C_1 and C_2. Hence, we say that the mapping defined between C_1 and C_2 must support bi-directional query translation.

The processes are performed iteratively until a predefined query execution limit is reached. Chapter 6 gives an in-depth explanation of the hybrid integration architecture, and the query processing function of the proposed framework.

3.6 Conclusion

In this chapter, we have introduced a semantic integration framework for the online accommodation domain. The goal is to tackle the major challenges of large-scale online information integration, where the integrated information sources are highly dynamic, distributed and heterogeneous. We combined the strengths offered by conventional integration approaches, and unique features provide by the proposed framework can be summarized as follows:

- Ontology Mediation – semantic interoperability is achieved via ontology mediation. A semantic network is established to connect together heterogeneous information sources via schema mappings. Global level consensus is not achieved by enforcing one single global data model. Instead, it is built by a group of inter-connected local agreements.

- Publish-Combine-Use Integration Process – we simplify the conventional information integration process and change it to a more systematic integration process. The modified process is organized into three collaborative steps, where each step is carried out by a specific type of user. Information providers now have more control over their own information sources. Individual information sources can join or leave the information sharing network in a self-managed fashion. Efforts required for data source acquisition and schema mappings are reduced.

- Hybrid Architecture – query processing in the local community groups is performed following the centralized approach. Query answering among a set of community groups is performed in a collaborative manner following the distributed approach.

On one hand, we leverage the flexibility and scalability offered by the distributed approach. On the other hand, we leverage the benefits inherited from the mediator-wrapper approach and the centralized integration approach. We also discussed various aspects of the proposed framework, including framework architecture, integration process and the utilization of ontology. This chapter gives an overview of the proposed framework. Detail on various parts of the proposed framework are covered in Chapter 4, 5 and 6. Chapter 4 focuses on the semantic layer of the proposed framework, and the developed techniques and methods are discussed. Detailed information of the integration process is discussed in chapter 5. Chapter 6 discusses the techniques used in query processing.

Chapter 4

Ontology and Semantic Interoperability

In Chapter 3, we have introduced the semantic integration framework. As mentioned in Chapter 3, the framework can be divided into three layers: the *semantic layer*, the *process layer*, and the *query layer*. The *semantic layer* captures the semantic information defined over the group of heterogeneous information sources of accommodation information. It includes various techniques and methods developed for enabling the semantic interoperability of the information sharing network. The *process layer* defines activities involved in the information integration process, and provides tools to assist the integration of heterogeneous information sources. The *query layer* contains methods and techniques developed for querying the integrated information sources. All three layers together forms the semantic integration framework, and provides an effective and flexible solution for large-scale information integration. In this chapter, we give a further discussion on the *semantic layer* of the proposed integration framework, and introduce the techniques and methods developed for this layer. To assist the local to global schema mapping process, we propose a mapping ontology for representing concept mappings defined between a pair of a XML schema and ontology. To facilitate the global to global schema mapping, we developed a many-to-many ontology mapping method to quickly identify the group of similar ontologies for a given set of input ontologies. The remaining sections of this chapter are organized as follows: Section 1 gives a formal definition of the semantic layer. The proposed mapping ontology is introduced in Section 2. Section 3 introduces the many-to-many ontology mapping method, followed by the conclusion in Section 4.

4.1 Semantic Layer

The semantic layer can be considered as a collection of inter-connected local conceptual models, where each local conceptual model provides an integrated conceptual view over its community group and the containing information sources. The structure of the semantic layer is presented as follows:

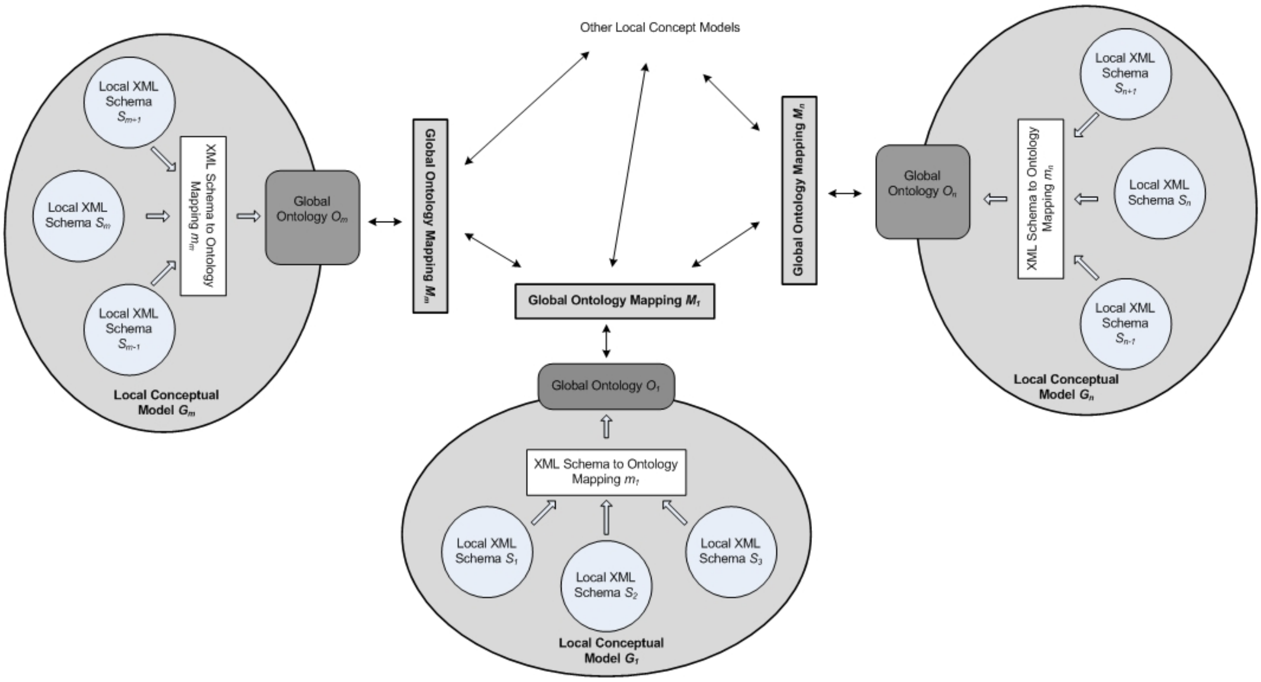

Other Local Concept Models
Local XML Schema S_{m+1}
Local XML Schema S_m
Local XML Schema S_{m-1}
XML Schema to Ontology Mapping m_m
Global Ontology O_m
Global Ontology Mapping M_m
Local Conceptual Model G_m
Global Ontology Mapping M_1
Global Ontology O_1
XML Schema to Ontology Mapping m_1
Local XML Schema S_1
Local XML Schema S_2
Local XML Schema S_3
Local Conceptual Model G_1
Global Ontology Mapping M_n
Global Ontology O_n
XML Schema to Ontology Mapping m_n
Local XML Schema S_{n+1}
Local XML Schema S_n
Local XML Schema S_{n-1}
Local Conceptual Model G_n

Figure 4.1: Semantic layer

As shown in Figure 4.1, the semantic layer, denoted as SL, can be formally defined as a turple $(\overline{G}, \overline{M})$ where

- $\overline{G}$ is a set of local conceptual models defined for each community group. Each local concept model G_i is a turple $(O_i, \overline{S}, \overline{m})$ where O_i is the global ontology shared by members of the community group, $\overline{S}$ is the set of local XML schemas, and $\overline{m}$ is the set of local mappings defined between the global schema G_i and the set of XML schema $\overline{S}$. Each local mapping m is a mapping instance defined using the concept mapping ontology, which will be introduced in Section 4.2.2.
- $\overline{M}$ is a set of global mappings defined among the community groups $\overline{G}$. Each global mapping M_i is defined from an ontology O_i in $\overline{O}$ to a set of ontologies $\overline{O_k} \subset \overline{O}$, where $O_i \notin \overline{O_k}$.

The semantic layer can be considered as a large group of data models (including both local schemas and global schemas), which are inter-connected via schema mappings (including both LGSMs and GGSMs). To enable the semantic interoperability of individual community groups, and to facilitate information integration among heterogeneous XML-based information sources, a local-to-global semantic relationship is created. A schema defined from each XML data source is mapped to the global ontology for that community group. The concept mapping process can be viewed as the mapping of concepts defined from the local XML schema to their corresponding concepts defined from the global ontology. On the other hand, in order to achieve seamless information sharing across different community groups, schema mappings are defined between adjacent global ontologies. Since the formation of the information sharing network heavily relies on the defined schema mappings, incorrect mappings will directly affect the performance of the formed information sharing network. Therefore, to assist the schema mapping processes, we developed two mapping techniques: 1.) a local-to-global mapping techniques; 2.) and a global-to-global mapping method. Section 4.2 introduces the technique developed for the local-to-global schema mapping process, and Section 4.3 introduces the method developed for the global-to-global schema mapping process.

4.2 Local-to-Global Schema Mapping

In the proposed integration framework, the local-to-global schema mapping is manually performed by data source owners known as the information providers. Although a lot of research efforts were devoted for automating the schema mapping process, most of the developed techniques and tools are still inaccurate, require complex training processes or require a large amount of user refinement. In addition, some research studies show that the schema mapping defined by human users outperforms most of the automated mapping techniques in terms of accuracy and efficiency. Hence, in this research, we use the manual schema mapping approach for creating the local-to-global schema mapping. To assist the mapping process, we developed a Web-based tool to allow easy definition of concept alignments between a XML schema and an ontology. The developed mapping tool is shown in Chapter 6. The concept mapping process is achieved by dragging and dropping a pair of similar concepts. Since the concept mapping process is manually performed by human users, our research focus is not the mapping process itself, but rather the representation of the defined mappings.

In Section 3.3.2, we have mentioned that each community group adopts a bottom-up query processing approach, where data is first translated from their local XML formats to the common ontology format, and then aggregated together to form a single ontology-compliant data set format. The data transformation is dependent on the schema mapping defined between the XML schema and the global ontology. However, due to the fundamental differences between XML schema and ontology, data structural information defined in XML schema cannot be precisely described using ontology (Klein *et al.*, 2000). Likewise, concepts and relations defined in ontology cannot be accurately modeled using XML schema. These differences lead to possible loss of information during the transformation from XML document to ontology instance or vice versa, thus they become a major obstacle for XML to ontology transformation.

To overcome this problem, and to enable consistent data transformation between a XML format and ontology format, we propose a mapping representation ontology for describing concept mappings defined between an XML schema and an ontology. Our primary goal is to encapsulate a sufficient amount of information in order to compensate the loss of information occurred during the single-trip data translation process.

4.2.1 XML Schema and Ontology Comparison

We start by comparing XML schema and ontology. Previous studies already revealed the differences between XML schema and ontology (Decker *et al.*, 2000; Gil & Ratnakar, 2002; Klein *et al.*, 2000), while XML schema provides rich syntax and structure definitions for data modeling, ontology, on the other hand, allows for a more sophisticated semantic modeling of a particular domain. In this work, we analyze the possible problems that may occur during the translation between XML and ontology instance.

In this section, we generally divide the differences between XML schema and ontology into three groups: data type, structure and relation. Table 4.1 shows the major differences between XML schema and ontology, their similarities are not covered.

	XML Schema	**Ontology**
Data type	XML schema supports large number of buildin data types including string, boolean, decimal, float, date, etc. A complete list of datatypes is documented in **?**.	Some ontology languages such as RDF only supports limited number of data types. Others such as DAML and OWL allows use of XML Schema datatypes by referring to the datatype URI.
Structure	XML schema uses nested data structure, where each element can be mixed with other simple, complex or mixed elements. The topmost element is considered as the root element of the concept hierarchy. Upper elements are seen as the parent of its content lower elements.	Ontology supports element composition through properties. Each class can have various datatype properties and object properties. However, ontology is an abstract knowledge base where relationships among the terminologies defined in the knowledge base can form a circular concept graph. Therefore, every class exist in the ontology can be seen as the root element.
	XML schema allows the definition of structural constraints, concepts such as sequence is used to describe the order between content items.	Ontology does not support order between properties.
Relation	XML schema only supports inheritance through type derivation (extension or restriction). It does not support multiple-inheritance.	Ontology supports multiple-inheritance, one class can inherit properties from multiple parent classes.
	XML schema does not provide grammars for relation constraints definition.	Ontology supports inheritance on properties, it also provides simple logics on relations such as *transitive* and *symmetric* for reasoning on class.

Table 4.1: XML schema and ontology difference

Due to the above mentioned differences, we argue that it is not feasible to translate an XML document completely into an ontology instance or vice versa without the loss of any structural or semantic information. We tested our argument with some conventional XML to ontology translation approaches, including (Bohring & Auer, 2005; Ferdinand *et al.*, 2004; Lehti & Fankhauser, 2004; Rodrigues & Cardoso, 2006; Zhang & Gu, 2005).

In our test, the reversibility of a given translation process is defined as: For an input ontology instance O_i, $T_{o\rightarrow x}$ is reversible if and only if $O_i = T_{o\rightarrow x}(T_{x\rightarrow o}(O_i))$; For an input XML document X_i, $T_{x\rightarrow o}$ is reversible if and only if $X_i = T_{x\rightarrow o}(T_{o\rightarrow x}(X_i))$. In both cases:

- $T_{x\rightarrow o}$ represents the process of XML to ontology translation.

- $T_{o \rightarrow x}$ represents the process of ontology to XML translation.
- ontology instance O_1 = ontology instance O_2 when both O_1 and O_2 comply with the same ontology schema and they both carry the same amount of information. Likewise, XML document X_1 = XML document X_2 when both X_1 and X_2 comply with the same XML schema and they both carry the same amount of information.

Following results are generated from the experiment:

Approach	$T_{x \rightarrow o}$	$T_{o \rightarrow x}$	**Preservation of Data type Information**	**Preservation of Structural Information**	**Preservation of Relational Information**	**Reversible**
Bohring and Auer, 2005	Yes	No	Yes	Yes	NA	NA
Lehti and Fankhauser, 2005	Yes	No	Yes	No	NA	NA
Rodrigues et al., 2006	Yes	No	Yes	Yes	NA	NA
Ferdinand et al., 2004	Yes	No	Yes	Yes	NA	NA
Zhang & Gu 2005	No	Yes	Yes	NA	NA	NA

Table 4.2: Conventional translation approaches

Bohring (Bohring & Auer, 2005), Lehti (Lehti & Fankhauser, 2004), and Ferdinand (Ferdinand *et al.*, 2004) all proposed methods in automatically generating OWL ontologies from either an XML instance or schema, with Bohring and Auers proposal being the only proposal where the availability of an XML schema is not mandatory. Due to the fundamental differences between XML schema and ontology (Yang *et al.*, 2007), all three proposals had to derive methodologies to overcome issues such as the preservation of data type, structural and relational information to ensure that the integrity of the information was preserved during the translation process. Both Bohring and Ferdinand argued the fact relational information can be detected from XML documents; hence an XML instance is required for the mapping to take place.

Bohring (Bohring & Auer, 2005) proposed a method that offers the generation of an XML schema from an instance when it is not available and attempts to preserve the schema's structure via relations discovered from the instance. While data type information is generally preserved, our investigation on XML2OWL, the prototype based on Bohring and Auer's approach, has shown some drawbacks that this approach entails. Unless an XML schema is provided, data type information can be misinterpreted during the derivation of the XML schema from an instance as data type information is generally not explicitly declared within an XML instance. Moreover, schemas generated would not be as detailed as a manually created XML schema, not to mention the fact that it is unable to detect schematic elements such as: SimpleTypes, patterns substitionGroups and others. It should also be noted that relational information within the OWL model is interpreted rather than preserved.

Ferdinand (Ferdinand *et al.*, 2004) proposed an automated lifting of XML schema to OWL by mapping XML to RDF and XML schema to OWL. Similar to Bohring and Auer's work, it attempts to interpret the semantics that implicitly exist within the XML documents, hence preserving its structure via the discovered relationships. The main drawback with this proposal is that the mappings are done independently; hence the RDF instances generated may not comply with the OWL model created from the XML schema. The explicitly generated OWL model shows which element is the root element mapped from the XML schema and as implicit relations between elements are interpreted from the XML instance, it can be considered that structural information are loosely defined.

Lehti (Lehti & Fankhauser, 2004), proposed a method for integrating XML data sources via an OWL based system. Unlike Bohring and Auer's proposal, their method requires an XML schema in order to initiate mapping from XML to OWL. Although data type information is preserved, structural information does not seem to be conserved during the mapping process. In general, complex and simple type definitions are respectively mapped to OWL classes and datatypes, while elements and attribute declarations are mapped to OWL properties. Relations between elements are not interpreted. For example, the defined OWL model does not have any indication as to which class is the root element. As OWL does not define order between properties, order is ignored; however this will create inconsistencies if we wish to reverse the mapping and create XML schema and instances from the OWL ontology due to the fact that structural information is lost during XML to OWL transformation.

Different to the above mentioned mapping methodologies, Rodrigues (Rodrigues & Cardoso, 2006) proposed a manual solution to map XML schema documents into existing OWL ontologies which will then generate the rules required for the automatic transformation of XML instances into OWL instances. Unlike the above mentioned works, JXML2OWL allows XML instances to be mapped into an existing OWL ontology instance. As JXML2OWL allows the ontology model to be predefined independently, instances generated is semantically richer than ontology instances generated from an XML model.

In contrast to the above works, which mainly focused on facilitating translation from XML to ontology, Zhang (Zhang & Gu, 2005) provided a method for the translation from ontology to XML. They proposed an algorithm, called the OTX, which takes ontology as input and creates an XML schema output that contains certain parts of the converted ontology. The created XML schema will act as the mediated semantic schema which will then be mapped onto the source schema. A semantic knowledge base is used to govern the mapping between the mediated semantic and the source schema. As XML schema supports a wide range of data types, this proposal is capable to preserve data type information during ontology to XML translation. However as multiple inheritance is not considered in their work, it is difficult to justify whether structural or relational information has been preserved during the translation. Zhang and Gus work does not restrict the ontology language used, as long as it is capable of representing the hierarchy and relationships between concepts.

It should be noted that none of the above mentioned research provided or demonstrated a potential for allowing a bidirectional data translation between XML and ontology. Our work aims to provide a solution that facilitates the bidirectional translation between XML to ontology by ensuring that

1. Information on element order is captured during $T_{x \to o}$;

2. Preservation of data type information during $T_{x \to o}$, regardless of the ontology language used;

3. Most importantly, sufficient information will be captured during $T_{x \to o}$ to ensure $T_{o \to x}$ can be undertaken without any loss of information.

4.2.2 Use of Ontology for Concept Mapping

One way to achieve consistent bidirectional translation between XML and ontology is to use a compensation approach. We defined the problem domain as follows: given an XML schema X, an ontology schema O and a set of concept mappings defined between X and O is denoted as M_{xo}, the logical constraints defined by the schema X is defined as set $\sum_x$ covering all syntactical and structural definitions, while logical constraints defined by the ontology O is defined as $\sum_o$ covering all concepts, properties and relations. In addition, the logical information encapsulated in M_{xo} is defined as $\sum_M$. The problem of information loss occurring during the translation between XML and ontology is denoted as

$\sum_{loss} = \neg(\sum_x \cap \sum_o)$, and to compensate the loss of information the condition $\sum_{loss} \subseteq \sum_M$ has to be met so that $\sum_x \subseteq (\sum_o \cup \sum_M)$ and $\sum_o \subseteq (\sum_x \cup \sum_M)$. Therefore, we developed an ontology capable in capturing concept mapping information defined between XML schema and ontology.

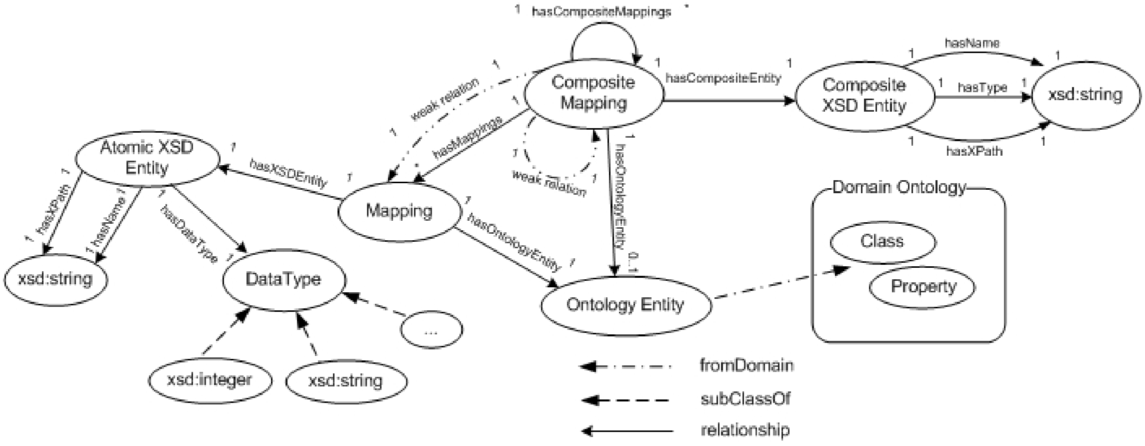


Figure 4.2: Partial description of concept mapping ontology

Concept Mapping Ontology (CMO)

Figure 4.2 shows the CMO (Concept Mapping Ontology). The class Mapping is defined to represent atomic mappings between an atomic XSD concept and an ontology entity. Another class, Composite Mapping, is used to represent a node structure, where its sub-nodes are represented by other Composite Mapping instances, and its leaves are represented through Mapping instances. In addition, the Mapping class is associated with an Atomic XSD Entity and an Ontology Entity, where the Atomic XSD Entity class retains all information required for the construction of an atomic XML element or attribute. A set of formal definitions is given below.

1. A *Composite Mapping* (CM) is defined as a 3-tuple $CM = ((M_s|CM_s|EM_s)^*, CE?, OE?)$

where M_s is the set of sub mappings; CM_s is the set of sub composite mappings; EM_s is the set of sub-empty mapping; CE is a *Composite XSD Entity* used for capturing structural information of a complex XML schema element; OE is an *Ontology Entity* used for representing class, data type property or object property from ontology. An *Empty Mapping* is a *Composite Mapping* with either no CE or OE.

2. A *Mapping* (M) is defined as a 2-tuple $M = (AE, OE)$ where AE is an *Atomic XSD Entity* used for capturing concept information of the atomic XML schema element selected for the mapping.

3. An *Atomic XSD Entity* (AE) is defined as a 3-tuple $AE = (P, N, DT)$ where P is the XPath of AE; N is the name of AE; DT is the data type of AE, which can be any XML schema simple type.

4. An *Ontology Entity* (OE) is defined as an entity with reference to other *Classes*, *Data Type Properties* or *Object Properties* from the domain ontology.

5. A *Weak Relation* is a relationship that only applies to the *Atomic XSD Entities* between either two *Composite Mappings* or between a *Composite Mapping* and a *Mapping*.

Note that a conventional mapping representation approach is considered as ontology-oriented, meaning that for each ontology class, data type or object property, a corresponding concept mapping representation is created so that all mappings are linked together through ontology relations (Rodrigues & Cardoso, 2006). Such an approach is beneficial for retaining semantic information defined in ontology, thus is mainly used for XML to ontology translation. In contrast, CMO focuses on the structure defined by the XML schema. Starting from the root element, for each composite element a composite mapping is created to model its position. Element sequence is modeled using left-right relation, where the element appears on the left most mapping has a higher order ranking than its right hand side correspondences. All concept mappings are linked together as a recursive node-leaf structure. The goal is to retain all information, including the data type, structural and relational information defined in both the XML schema and ontology.

Creation of a CMO Instance

A CMO instance is a document that is specifically created to represent concept mappings defined between an XML schema and an ontology. The creation of a CMO instance follows the following rules:

1. A *Mapping* instance is created when a mapping is defined between either an xsd:attribute or xsd:element containing no sub-elements or xsd:simpleType and an ontology Data Type Property.

2. A *Composite Mapping* instance is created when a mapping is defined between either an xsd:complexType or xsd:element containing sub-elements and an ontology *Class*. Same set of rules are applied to the sub-elements of the composite element.

3. Given two mappings (either common *Mappings*, *Composite Mapping* or *Empty Mappings*) M_1 and M_2, if a relation R exists between OE_1 in M_1 and OE_2 in M_2, then an *Empty Mapping* instance EM_i is created to model R between M_1 and M_2 , where M_1 is the parent of EM_i and M_2 is the child of EM_i.

4. An *Empty Mapping* instance is created when a structural difference occurs between a parent mapping and its child mapping. *Empty Mapping* with

ontology class or relation is created to model missing ontology structure or concept. *Empty Mapping* with XML element is used to model missing XML structure or concept.

5. Given a *Composite Mapping* M_1 and a mapping (either *Composite Mapping* or *Mapping*) M_2, if the CE in M_1 is the parent of the CE/AE in M_2 and there already exist a path between M_1 and M_2 that represents their ontological concepts and relations, then a *Weak Relation* will be created for representing the parent-child relation between M_1 and M_2.

These rules provide the basic guidelines for the creation of a CMO instance. Rule 1 and 2 are defined to model corresponding concept mappings between the XML schema and ontology; Rule 3 is defined to model the relational information between concept mappings; Rule 4 and 5 are defined to capture structural differences between XML schema and ontology.

4.2.2.1 Ontology Mediated XML Transformation

The ultimate goal of CMO is to capture a sufficient amount of concept mapping information to achieve bidirectional data transformation. In previous sections, we have explained the use of CMO for capturing concept mapping information. In this section, we will introduce the bidirectional data translation process and the ontology mediated XML transformation.

In earlier parts of this chapter, we have briefly introduced the ontology mediated XML transformation method. As shown in figure 3.4, the transformation process contains two phases: *Concept Mapping* and *Data Translation*. *Concept Mapping* focuses on the definition of the concepts and initiates the mapping between the local XML schemas and the global domain ontology. During *Concept*

Mapping, local XML schemas are mapped to the global ontology where concept mapping information is captured using CMO. Data Translation focuses on the exchange of data between different XML formats. During *Data Translation*, the input XML document is firstly transformed into the global ontology format, which will then be transformed into the targeted XML format. The transformation process is performed based on the information captured by CMO, as shown in figure 3.4. Instance level data translation consists of two steps: XML to ontology Translation and ontology to XML Translation. The process of translating an XML document into an ontology instance is called XML to ontology Translation. Similarly, the process of translating an ontology instance into an XML document is called ontology to XML Translation. The two translation algorithms are introduced below.

XML to Ontology Translation

The process of XML to ontology translation denoted as $T_{x \to o}$ is defined as: Given an input XML document X_i and the mapping ontology instance M_i, an ontology graph G with values from corresponding elements in X_i will be created. Each node in G is denoted as $N = (n, t, v)$ where n represents the name of node N, t represents the type of N (class, data property or object property), and v represents the value of N. The flow of $T_{x \to o}$ is shown in algorithm 1.

Algorithm 1 XML to Ontology translation algorithm

```
Translate $X_i$ into $X_i = [E_1, E_2, E_3 \dots E_n]$
for all $E \in X_i$ do
  $M \leftarrow M_i(E)$
  if $M$ is a Composite Mapping then
    $G \leftarrow$ Create_Graph($M$)
    $N_s \leftarrow f_p(G_r, M)$
    if $N_s$ is empty set then
      Set $G$ as the root graph $G_r$
    else
      for all $N_p \in N_s$ do
        Set $G$ as the child graph of $N_p$
      end for
    end if
  else
    if $M$ is a common Mapping then
      $M = (AE, OE)$
      $N_s \leftarrow f_p(G_r, M)$
      $N_c \leftarrow f_c(N_s, M)$, $N_c = (n_c, t_c, v_c)$
      if $v_c$ is empty then
        $v_c = E$'s value
      else
        create node $N'_c = (n'_c, t'_c, v'_c)$
        set $v'_c = E$'s value, $n'_c = OE$'s name, $t'_c$= $OE$'s type
        set $N'_c$ child node of $N_p$
      end if
    else {$M$ is an Empty Mapping}
      Ignore $E$ and Continue
    end if
  end if
end for
```

Algorithm 2 Function Create_Graph(M)

```
$M = (M_s, CM_s, EM_s, AE?, OE)$
create root node $N_r = (n_r, t_r, v_r)$
set $n_r = OE$'s name, $t_r = OE$'s type
for all $m \in M_s$ do
    $m = (AE_m, OE_m)$
    create node $N_m = (n_m, t_m, v_m)$
    set $n_m = OE_m$'s name, $t_m = OE_m$'s type
    set $N_m$ the child node of $N_r$
end for
for all $em \in EM_s$ do
    if $O(em)$ then
        $G_{em} \leftarrow Create\_Graph(em)$
        set $G_{em}$ sub-graph of $N_r$
    end if
end for
```

Notation	Definition
AE	*Atomic Entity* as defined in section 4.2.2
$f_c(N_s, M)$	Given a node N_s and a mapping M, this function returns the node N_c such that $N_c = AE$ in M, and there exist N_p in N_s that N_p is parent node of N_c
$f_p(G_r, M)$	Given a graph G_r and a mapping M, this function returns the set of node N_s in G_r that for all $N \in N_s$, N is parent node of OE in M
$f_r(G)$	Find the root node N_r, given a graph G
M_s, CM_s, EM_s	Set of mappings, composite mappings and empty mappings
$M_i(E)$	Given an XML element E, find the corresponding mapping M defined in the mapping representation instance M_i
N_s	Set of nodes
OE	Ontology Entity as defined in Section 4.2.2
$O(em)$	Given an empty mapping em, returns true if em represents an Ontology entity

The Create_Graph(M) function creates a graph structure based on a given mapping definition M. For all the mappings and empty mappings defined in M and the descendants of M a corresponding node is created. Algorithm 2 shows the Create_Graph function. Notations used in previous workflow are listed in the table.

Ontology to XML Translation

The translation from ontology graph to XML (denoted as $T_{o \to x}$) is defined as: Given an ontology graph G and the mapping ontology instance M_i, create an XML document X_o with values from corresponding nodes in G. The flow of $T_{o \to x}$ is shown in algorithm 3 .

Algorithm 3 Ontology to XML translation

```
$M_r \leftarrow f_r(M_i)$
**if** $M_r$ is *Composite Mapping* **then**
  $M_r = (M_s, CM_s, EM_s, AE, OE)$
  $N_s \leftarrow f_n(G, M)$
  **for all** $n \in N_s$ **do**
    start element $N$
    **for all** $m \in M_s \cap CM_s \cap EM_s$ **do**
      repeat translation process on $m$
    **end for**
    end element $N$
  **end for**
**else** {$M_r$ is common Mapping}
  $M = (AE, OE)$
  $AE = (P, N, DT)$
  $N_s \leftarrow f_n(G, M)$
  **for all** $n \in N_s$ **do**
    create element $E$ with $value$ = $n$'s value and $name$ = $AE$'s name
  **end for**
**else** {$M_r$ is *Empty Mapping* and $A(M)$}
  $M = (M_s, CM_s, EM_s, AE, OE)$
  $AE = (P, N, DT)$
  create element $E$ with name = $AE$'s name
**end if**
```

Notation	**Definition**
AE	*Atomic Entity* as defined in Section 4.2.2
$A(M)$	Given an empty mapping M, returns true if M represents an XML entity.
DT	*Data type* as defined in Section 4.2.2
$f_n(G, M)$	Given a graph G and a mapping M, this function returns the set of node N_s that all $N \in N_s$, N is equivalent to AE in M
$f_r(M_i)$	Find the root mapping M_r, given a mapping Ontology M_i
M_s, CM_s, EM_s	Set of mappings, composite mappings and empty mappings
N	Name of a given atomic XSD entity as defined in Section 4.2.2
N_s	Set of nodes
OE	*Ontology Entity* as defined in Section 4.2.2
P	Path of a given atomic XSD entity as defined in Section 4.2.2

The proposed transformation method is used in the query processing phase to facilitate the translation of XML data to ontology format. Detailed information about the translation process with a worked example is covered in chapter 5.

4.3 Global-to-Global Schema Mapping

To enable information sharing across different community groups, similar ontologies from neighborhood community groups are also mapped together. In our research, this concept mapping process is called the Global-to-Global Schema Mapping (GGSM). Unlike the LGSM which creates a one-to-one mapping relationship between a pair of schemas, the GGSM creates a many-to-many mapping relationship among a group of schemas, where the number of involved ontologies is large. The goal is to enable information sharing across different community groups, and to establish a semantic overlay network following the distributed P2P architecture.

The main challenge faced by the GGSM is the identification of a set of similar ontologies. Since the online accommodation domain is vast and contains a large number of community groups, the total number of involved ontologies is also large. In order to find the set of similar ontologies, information operators often need to compare on a large set of existing ontologies, and the comparison is often tedious and error-prone. Hence, to assist the GGSM, we propose a concept classification based mapping algorithm to enable the automatic detection of a group of similar ontologies. In contrast to conventional ontology mapping solutions, the proposed solution is able to create a many-to-many mapping relationship on a set of heterogeneous ontologies.

4.3.1 Many-to-Many Ontology Mapping

We start by demonstrating the computational complexity of pair-wise mapping methods under many-to-many mapping scenarios. Conventional mapping approaches compare concepts in a pair-wise and iterative fashion. In order to completely map concepts defined in two homogeneous ontologies, all potential concept pairs need to be iteratively compared. We find that for two ontologies, which respectively have n and m number of concepts, at least $\sum_{k=0}^{n-1}(m-k), (m \geq n)$ number of comparisons are needed in order to compare one group of concepts against another. The computational complexity of such a process is $O(n^2)$ when $m = n$. It should be noted that the complexity increases dramatically as the number of concepts (n) increases. This is concurrent with Marc and Steffen's (Ehrig & Staab, 2004a) findings in which they revealed the computational complexity of some popular ontology mapping approaches including Prompt (Noy & Musen, 2003), NOM (Ehrig & Staab, 2004b) and GLUE (Doan *et al.*, 2003). Their results also show that the computational complexity of the above mentioned approaches is $O(n^2)$ or $O(n^2 \times \log n)$. In addition, to establish mappings among a group of ontologies (more than two ontologies), the number of ontologies involved in the mapping process will also dramatically increase the complexity of the entire mapping process. For example, to compare all concepts defined in k number of ontologies where each ontology has n number of concepts, the computational complexity of using pair-wise mapping solutions is $O(n^{(k-1)})$.

To reduce the number of comparisons required in establishing many-to-many mappings between ontologies, we propose a Concept Classification based Mapping (CCM) approach. Instead of comparing all concept pairs in an iterative and repeating fashion, a classification tree is constructed and used to classify a group of input ontology concepts into conceptual groups. By leveraging the power of classification trees (Breiman *et al.*, 1984), we aim to minimize the number of comparison steps required in a many-to-many ontology mapping scenario. Figure 4.3 shows the overall mapping process. The process contains three major steps: ontology parsing, tree construction and concept classification.

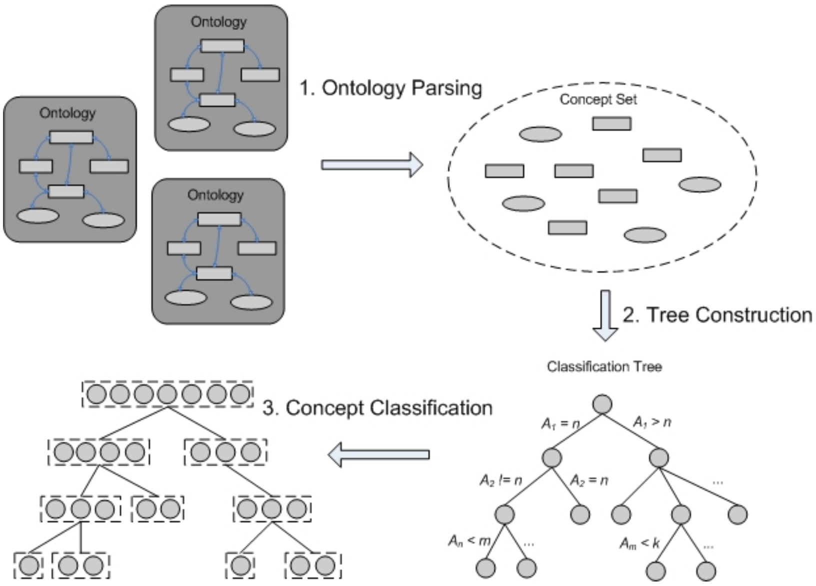


Figure 4.3: Concept classification based mapping process

4.3.1.1 Ontology Parsing

Ontology parsing is the phase where ontologies are arranged into an inter-connected set of concepts. The primary goal is to eliminate both the syntactic and semantic heterogeneity carried by different ontology modeling languages. Given a number of ontologies written in different ontology formats such as DAML+OIL (Wroe *et al.*, 2003) or OWL (McGuinness & van Harmelen, 2003), each ontology is converted into a node-edge concept set denoted as $S = (\overline{N}, \overline{E})$ where $\overline{N}$ is a set of nodes, and $\overline{E} = \{(N_i, N_j)|N_i \in \overline{N}, N_j \in \overline{N}\}$ is a set of edges. The ontology parsing process can therefore be formalized as: given an ontology $O = (\overline{C}, \overline{P})$ where $\overline{C}$ is a set of ontology classes, and $\overline{P}$ is a set of ontology properties. Convert O into concept set $S = (\overline{N}, \overline{E})$, where all classes and properties are translated into nodes $\overline{N}$, and the ownerships between classes and properties are translated into edges $\overline{E}$. Since ontology parsing is only a preparation step towards the actual mapping process, it is therefore not considered as part of the overall complexity of the mapping process and is not further discussed in this paper.

4.3.1.2 Classification Tree Construction

A classification tree is created to classify concepts into multiple conceptual groups. In order to construct a classification tree, a set of split attributes need to be defined. A split attribute is a selection criteria that is used to split one set of data into two or multiple data sets. In conventional tree construction approaches, the set of split attributes and the values associated with them are obtained from a series of learning processes (Dobra, 2003). These tree construction approaches often consist of two phases (Breiman *et al.*, 1984): the first phase is called the growth phase, which aims to build a large classification tree from a set of training data. The second phase is called the pruning phase, which focuses on the cost-complexity measurement and generalization error estimation. Instead of using machine learning to determine the set of split attributes, we provide a predefined set of split attributes and we consider the process of concept classification as a special type of data classification with pre-known split attribute and unknown attribute value. We construct the classification tree based on two major split attributes: concept granularity and semantic similarity, and inspired by quick-sort algorithm (Hoare, 1962) we use randomly picked concept as the attribute value. In addition, we use the edge-based approach (Sussna, 1993) for measuring granularity differences between two given concepts, and use Jiang and Conrath's approach (Jiang & Conrath, 1997) for measuring the semantic similarity. The granularity calculation is used to determine the conceptual depths of two given concepts, and the semantic similarity calculation is used to determine the degree of similarity between two given concepts. The constructed classification tree is introduced in section 3.

4.3.1.3 Concept Classification

In the last step, the group of input concepts obtained from the parsing process are fed into the classification tree constructed in the previous step. The concept classification process can be seen as a recursive number of partition steps, and in each partition step one group of concepts is spliced into multiple sub-groups based on the split criteria defined by the node of the classification tree. Such a process continues until it reaches the point that each concept group only contains an unique set of concepts with no ontology redundancy. We eliminate the possibility of mapping concepts from the same ontology in one concept group by restricting the number of concepts from each ontology to one. In other words, all concept groups are partitioned/spliced into sub-groups until all concepts from the same concept group are from different ontologies. Detailed information about the concept classification process is covered in section 4.

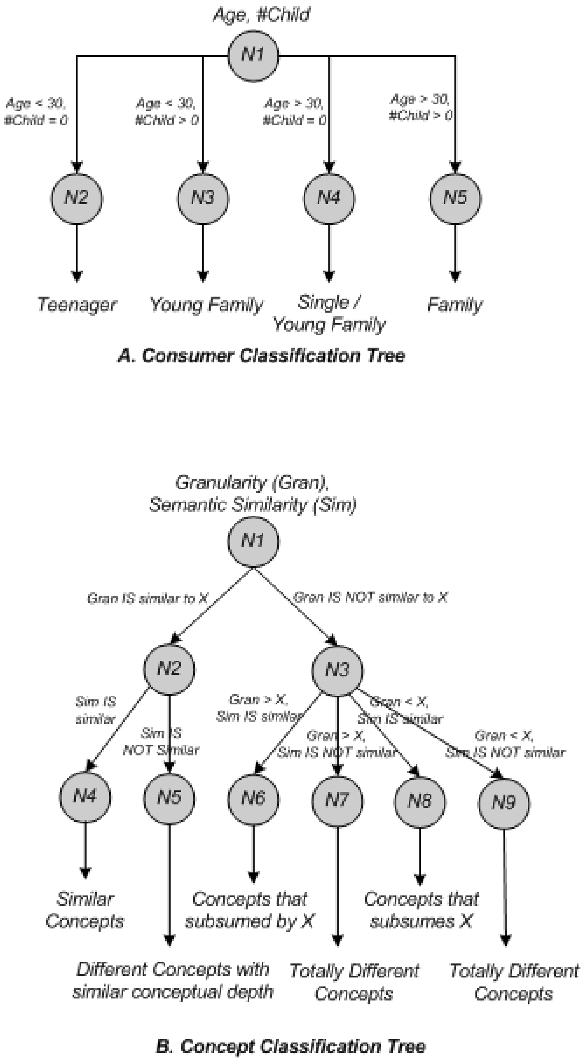
4.3.2 Classification Tree

A classification tree can be considered as a hierarchy of classifier nodes, where each classifier node is used to split one group of data into multiple sub-groups. Let $T = \overline{N}$ represent the classification tree with nodes $\overline{N}$. Each node N_i contains a set of split attributes $\overline{A_i}$. A split attribute A_i is used for spliting data into sub-groups. Let (N_i, N_j) represent the edge from node N_i to one of its children node N_j, then each edge (N_i, N_j) has a function $\mu_{N_i \rightarrow N_j}$, where the function $\mu_{N_i \rightarrow N_j} = \{(A_k, \lambda, V_k) | A_k \in \overline{A_i}, V_k \in \overline{V_j}\}$ has a set of split criterions. Each split criteria (A_k, λ, V_k) has a split attribute $A_k \in \overline{A_i}$ inherited from the parent node N_i, a logical predicate λ and an attribute value V_k. Figure 4.4-A shows a sample classification tree for classifying people into different consumer groups. As illustrated in the diagram, the root node contains two split attributes: Age and Number of Children. The classification function $\mu_{N_1 \rightarrow N_2}$ has two split criterion: $(Age, <, 30)$, $(NumberofChildren, =, 0)$, which can be used to determine the teenager consumer group.

As mentioned previously, the concept classification process can be considered as a special type of data classification with a pre-known set of split attributes and unknown attribute value. We also found that a group of ontology concepts can be considered as similar if they have similar granularity and high semantic similarity. Therefore, we constructed the concept classification tree as a total number of 9 classifier nodes with two split attributes. As shown in Figure 4.4-B, two split attributes are defined in the classification tree, which are *granularity* and *semantic similarity*. *Granularity* is used to measure the conceptual depth of given concepts, and it is obtained by locating the common parent of two given concepts and the distances from the concepts to their common parent. The *semantic similarity* is used to determine the semantic meaning of a given concept, and it is calculated based on Jiang and Conrath's approach (Jiang & Conrath, 1997). Six split criterion are defined in the bottom of the tree to classify concepts into similar conceptual groups, the variable X represents the concept that is randomly selected from the group of input ontology concepts. The selection of a random concept is inspired by the quick-sort algorithm, and this is also known as the pivot selection process. In order to accurately measure the granularity difference and the semantic similarity between two given concepts, we extended our solution on existing research, and detailed information is covered in the following two sub-sections.

4.3.2.1 Granularity Calculation

We base our granularity calculation on an edge based approach (Sussna, 1993), and we suggest to calculate the granularity of two given concepts based on the shortest edge distance from the concepts to their closest common parent. The calculation is done using the online dictionary WordNet(Miller, 1995). The closest common parent is defined as the first class upward in the concept hierarchy that subsumes both concepts, and the edge distance between the two concepts is defined as the shortest geometric distance between the nodes that represent the concepts. The concept with a longer edge distance to the common parent has higher granularity value than the one with shorter distance. Since some factors such as node density, node depth may affect the value calculated by the edge distance theory (Sussna, 1993), thus we consider three factors in the granularity calculation process, which are distance to common parent, node density and distance to leaf node. A leaf node is a concept node that with no child nodes. The node density of a concept c is obtained from the following formula:



A. Consumer Classification Tree

B. Concept Classification Tree

Figure 4.4: Classification tree

$$Dens(c) = \frac{E(c)}{\overline{E}} \tag{4.1}$$

$\overline{E}$ represents the average density of the hierarchy, and is obtained by dividing the total number of edges with the total number of parent nodes. $E(c)$ represents the number of siblings of the given concept node. The overall granularity calculation formula is shown as:

$$Gran(c_1, c_2) = \frac{Dens(c_1) \times path(c_1, p) \times leaf(c_2)}{Dens(c_2) \times path(c_2, p) \times leaf(c_1)} \tag{4.2}$$

p is the common parent between c_1 and c_2. $path(c, p)$ represents the shortest path from concept c to parent p, and $leaf(c)$ represents the shortest distance from concept c to its containing leaf concept node. The value obtained from the calculation shows the granularity difference between two given concepts. We say that concept c_1 has higher granularity than concept c_2 if the granularity value $Gran(c_1, c_2)$ is greater than 1. Or two concepts have same granularity when $Gran(c_1, c_2) = 1$.

4.3.2.2 Semantic Similarity Calculation

We choose to use Jay and David's approach for semantic similarity calculation. In their approach, the similarity comparison is derived from the edge-based approach by adding the node-based approach (Resnik, 1992) as a decision factor. Their approach also considered the factors of node depth and node density, and they used two variables $\alpha(\alpha \geq 0)$ and $\beta(0 \leq \beta \leq 1)$ to control the degree of node depth and density factors contributing to the edge weighting computation. Since our approach has separated the granularity calculation from semantic calculation, we do not consider these factors in the semantic similarity calculation. Hence the similarity comparison formula derived from Jay and David's approach is simplified as:

$$Sim(w_1, w_2) = 2 \times \log(P(LSuper(c_1, c_2))) - (\log P(c_1) + \log P(c_2)) \tag{4.3}$$

c_1, c_2 are the senses of word w_1 and w_2 in the WordNet taxonomy denoted as $c_1 = sen(w_1)$, $c_2 = sen(w_2)$ respectively, $LSuper(c_1, c_2)$ represents the lowest super-ordinate of c_1 and c_2. $P(c)$ is the probability of encountering concept c in the concept hierarchy. We refer you to (Jiang & Conrath, 1997) for detailed explanation of this formula.

4.3.3 Concept Classification

The concept classification process is built based on the logic of the quicksort algorithm (Hoare, 1962). We choose to use the quicksort algorithm for two reasons. First, the process of ontology mapping is an iterative number of comparison steps, where one concept needs to be compared with all other concepts defined from different ontologies. This scenario is similar to the quicksort algorithm, where every member from the input list is compared with each other in an iterative fashion. Second, the quicksort algorithm is able to reduce the number of comparisons to $O(n \log n)$ in the average case, thus it is able to minimize the number of unnecessary comparison steps required by conventional ontology mapping solutions.

Let $\overline{C}_{in} \subseteq \Omega$ be the set of input ontology concepts, c_r represents the random concept/pivot that is selected from Ω. Then the concept classification process can be formalized as shown in Algorithm 1. The concept classification process returns a set of mapping groups. For each mapping group there is a maximum one concept from each input ontology. The conceptual groups produced from the concept classification process are used as mappings to establish many-to-many concept mappings among a large number of heterogeneous input ontologies. An

Algorithm 4 $Classify(\overline{C}_{in})$

```
$Var : \overline{C}_{c1}, \overline{C}_{c2}, \overline{C}_{c3}, \overline{C}_{c4}, \overline{C}_{c5}, \overline{C}_{c6}$
if Every $c \in \overline{C}_{in}$ is from an unique ontology then
    return $\overline{C}_{in}$ as a Mapping Group
else
    select a random concept $c_r$ from $\overline{C}_{in}$
    for all $c \in \overline{C}_{in}$ do
        if $1 - \beta \leq Gran(c, c_r) \leq 1 + \beta$ then
            Add $c$ to $\overline{C}_{c1}$ if $\alpha \leq Sim(c, c_r) \leq 1$
            Otherwise, add $c$ to $\overline{C}_{c2}$
        else {Different Granularity}
            if $(Gran(c, c_r) > 1) and (Sim(c, c_r) < \alpha)$ then
                Add $c$ to $\overline{C}_{c3}$
            else if $(Gran(c, c_r) < 1) and (Sim(c, c_r) < \alpha)$ then
                Add $c$ to $\overline{C}_{c4}$
            else if $(Gran(c, c_r) > 1) and (Sim(c, c_r) > \alpha)$ then
                Add $c$ to $\overline{C}_{c5}$
            else if $(Gran(c, c_r) > 1) and (Sim(c, c_r) < \alpha)$ then
                Add $c$ to $\overline{C}_{c6}$
            end if
        end if
    end for
    if Every $c \in \overline{C}_{c1}$ is from an unique ontology then
        return $\overline{C}_{c1}$ as a Mapping Group +... + $Classify(\overline{C}_{c6})$
    else
        return $Classify(\overline{C}_{c1}) + ... + Classify(\overline{C}_{c6})$
    end if
end if
```

analysis of the features and performance of the approach is covered in more detail in Chapter 8.

4.3.4 Related Work

Previous research efforts have led to a large number of ontology mapping solutions. While the majority of the research have focused on solving the one-to-one ontology mapping problem, problems regarding many-to-many ontology mapping among a large group of heterogeneous ontologies have been neglected. Ming Mao, et. al. (Mao *et al.*, 2008) propose a non-instance learning-based approach for ontology mapping. Similar to our approach they treat ontology mapping as a binary-classification problem. In their approach, a SVM model is built and trained to classify testing data into different groups. Concept mappings are extracted from the testing data using a naive descendant extraction algorithm. However, their solution focuses on the one-to-one mapping of a pair of ontologies, thus it is rather complex when applied to a many-to-many mapping scenario.

Some researches also focused on improving the efficiency of the mapping process. Marc and Steffen (Ehrig & Staab, 2004a) proposed a quick ontology mapping (QOM) algorithm that reduces the rum-time complexity of conventional mapping approaches. The run-time complexity of their approach is $O(n \times \log n)$ as opposed to $O(n^2)$ for conventional mapping approaches. However, they improved the efficiency of the one-to-one mapping process by reducing the complexity of the similarity calculation method, whereas our approach aims to improve the efficiency of many-to-many ontology mapping by reducing the number of repeating concept comparisons.

4.4 Conclusion

In this chapter we have discussed the semantic layer of the proposed integration framework. In particular, we looked at how local conceptual models can be related together to form an integrated and unified global conceptual model. The resultant semantic layer represents the way semantic relationships are established among a large set of heterogeneous data sources. It also shows how local data sources are related to the global schema within each community group.

In section 4.2, we have explained the method for creating local to global schema mappings. The local conceptual model is created by mapping concepts defined from the local XML schemas to the corresponding concepts defined from the global ontology. Section 4.3 proposes a new method for creating global to global ontology mapping on concept similarity.

In summary, to enable semantic interoperability of the proposed semantic integration framework, we have developed two techniques. 1.) A mapping ontology for representing concept mappings defined between XML schema and ontology; 2.) And an automatic ontology mapping algorithm for the efficient detection of a group of similar ontologies. The developed methods are compared with some existing methods, and it is shown that our approaches have unique characteristics against conventional approaches. The semantic layer acts as a foundation for the query processing within the proposed framework, and chapter 6 will provide a detailed explaination of the proposed query layer.

Chapter 5

Information Integration Process

In Chapter 4, we have discussed the semantic layer of the proposed integration framework. The ultimate goal of the semantic layer is to establish a semantically interoperable network by defining schema mappings among heterogeneous information sources. In this chapter, we further discuss the process of creating the information sharing network, and look at the roles of human users involved in the information integration process. In addition, we introduce some new concepts and develop a few tools to assist the process of information integration. To facilitate the management of the published information sources, we introduce the concept of an *information group.* An information group is a collection of published information sources that is grouped together following a common characteristic. In order to allow users with no or limited query language knowledge to issue complex query messages, we introduce the concept of *query interface.* A query interface is a predefined query message with no query constraints but a set of input and output parameters. Each query interface is responsible for the dynamic and automatic generation of one type of query message. The remaining sections of this chapter are organized as follows. Section 5.1 introduces the proposed information integration process. The concept of an information group is discussed in Section 5.2. Section 5.3 discuss the concept of query interface, followed by the conclusion in section 5.5.

5.1 Publish-Combine-Use

As introduced in Chapter 3, we break the process of information integration into three sequential steps: publish, combine, and use. Figure 3.5 illustrates the proposed integration process. Information providers publish information sources to the information hub; hub operators combine and manage the published information sources; and information users query the published information sources. The integration process is formed by three fundamental operations, which are Publish, Combine and Use. Each operation is comprised of a number of tasks as described below.

- **Publish** – publishing information sources to the information hub is a relatively straightforward process. The first step is to determine some basic information about how to describe the submitted information source. Once the information is determined, the next step is to perform the registration, which is done through a Web-based user interface. During the registration, information providers need to relate their data source schema with the hub's global ontology, and this process is known as the schema mapping process.

  - *Register source*: The registration of an information source to the information hub is performed by information providers through a Web-based user interface. The registration process is as simple as filling out a Web-based registration form. The only challenging task is to determine the correct information to be used for describing the information source.

  - *Define schema mapping*: The second step in the publish process is the definition of schema mappings. This is also performed by the information providers with the assistance of a graphical tool. The mapping of the local source schema to the global ontology schema is also known as the LGSM. As explained in Chapter 4, the task of LGSM is to create conceptual alignments between corresponding concepts defined from the local schema to the global ontology schema. The schema mapping task is explained in Chapter 4.

- **Combine** – managing the published information sources involves some organizing and tidying up activities. First, hub operators need to classify the registered sources into meaningful groups, and each group is formed to serve a particular information need. Then some query interfaces are defined to provide user-friendly interfaces for accessing the published information sources.

  - *Define information group*: The classification of the published information is based on two decision factors. First, hub operators need to have some basic knowledge about each published information source. Then, they need to determine how to organize the published sources to better serve different information needs.
  - *Define query interface*: The goal of query interface definition is to provide graphical interfaces to the information users so that they can easily access the published information source. During the query interface definition, hub operators need to decide on the set of input and output parameters. The process is supported by a graphical tool that displays the global ontology schema as a concept hierarchy.

- **Use** – tasks included in the use phase are comparatively simple, as users only need to fill out the pre-defined query interfaces in order to gain access to the published information source. No query language knowledge is required throughout the process.
    - *Use query interface*: The defined query interfaces are displayed as graphical Web-based forms. Users only need to fill out the forms to query over the published information sources.

Since the tasks included in the publish process and the use process are relatively straighforward, or have already been covered in previous chapters, in this chapter, we mainly look at the tasks involved in the combine process. In particular, we discuss the tasks of information group definition and query interface definition. The following sections further explain each task in detail.

5.1.1 Integration Example - Hotel Rate Sharing

To help revisit the proposed integration process, we use a hotel rate information sharing example. Assume that a hotelier from Sydney wants to share its inventory information online. Following the proposed integration process, the hotelier first needs to choose the information hub they wish to join. In this case, a hotel information hub is selected. In order to join the hub, the hotelier needs to associate its source schema with the hub's global data model, and this is achieved via schema mapping. After mapping the corresponding concepts defined from the local schema to the global schema at the hub side, the inventory source is then connected with the information hub, and can be accessed by the public users. The publishing of other information sources to the selected information hub also follows the same registration scenario. The information network grows larger as new sources join the hub.

To manage the published information sources, and to better serve the information needs of general users, hub operators need to perform a number of source management tasks. One of the tasks is to classify the published information sources into meaningful groups. To do this, we introduce the concept of information group. An *information group* is a collection of information sources with common characteristics. For example, the set of information sources that provide hotel rate information in Sydney can be organized together to form an information group. The formed groups provide information for specific application domains. Another management task is to present the published information to users in well-organized and easy-to-use forms. To do this hub operators need to define a set of query interfaces and each interface is created for the generation of a particular type of query message. The defined query interface is associated with a number of input and output parameters, and is displayed to information users as a graphical form. Continuing on the previous example, a hotel rate search query interface is defined to provide hotel rate information base on location. The defined query interface takes location as input, and returns a list of hotel rate information including hotel name, address, room type and room price.

The defined query interfaces automate the integration process, as users with no or limited IT knowledge can issue complex query messages, and the process is as simple as filling out a Web form. To issue a query message that selects all hotel prices in Sydney, the user only needs to type in the value "Sydney" into the hotel location input field. A query message is then automatically issued to the hub. The issued query message is then processed following a bottom-up query processing approach. Finally, results generated from the query are presented to the user in a pre-defined format. All the above mentioned integration tasks are performed at the hub side. Hence, the information hub provides a collaboration platform for all users involved in the information integration process.

5.1.2 Benefits

In contrast to the conventional integration process, the proposed integration process is more agile and flexible. The task of information integration is no longer performed by a single type of user, rather it is distributed to various types of users including the data source owners, the hub operators and the public information users. All users now share the responsibility of integrating and maintaining the large numbers of online information sources, and the amount of effort left for the information intermediatories has been drastically reduced. The proposed integration process also provides a number of benefits to each type of users.

- Provider – the proposed process offers more freedom to the information providers, and individual information sources can join or leave the information sharing network in a self-managed fashion. In addition, individual providers now have the ability to manage and maintain their own source schemas, and changes occurred at the local schema can be directly managed by information providers.
- Operator – the tasks of collecting and managing a vast group of information sources is distributed to individual information providers. Hub operators can concentrate on the tasks of organizing and presenting the published information.
- User – through the definition of query interfaces, users with no or limited IT knowledge is able to query the published information sources.

5.2 Information Group

To allow the better management of the published information sources, we introduce the concept of information group. An information group is simply a set of published information sources that have common properties, for example, a group of hotel rate sources that provide rate information for hotels in Sydney can be grouped together to form a Sydney hotel rate information group. The primary goal of the information group concept is to organize the published sources into meaningful categories to better serve various information needs.

The formation of an information group is based on two sets of information. The first set is the description of the published information source, and this information is provided by the information providers during registration. Continuing on the previous example, the Sydney hotel price information source can be described as “an information source that provides hotel rate information of hotels located in Sydney”. Although the description information gives a general understanding on the published information source, this information is often vague or incomplete. Hence, considering the description information alone is not sufficient for creating an information group. Information providers also need to consult some other information generated from the schema mapping process. During schema mapping, the semantic relation established between the data source and the global ontology not only provides a physical connection to the published information source, it also creates a semantic structure of the source. During the schema mapping, each source schema is mapped to a fractional part of the global ontology, and the mapped ontology parts can be used to assist the understanding of the published information source structure.

5.3 Query Interface

Information users in the online accommodation domain often have limited computer knowledge, and are from diverse educational backgrounds. Hence it is unreasonable to expect all users to have enough knowledge for defining query messages. Therefore, to assist the issue of query messages, and to ensure that users with no or limited query language knowledge can perform complex query tasks, we introduce the concept of query interface.

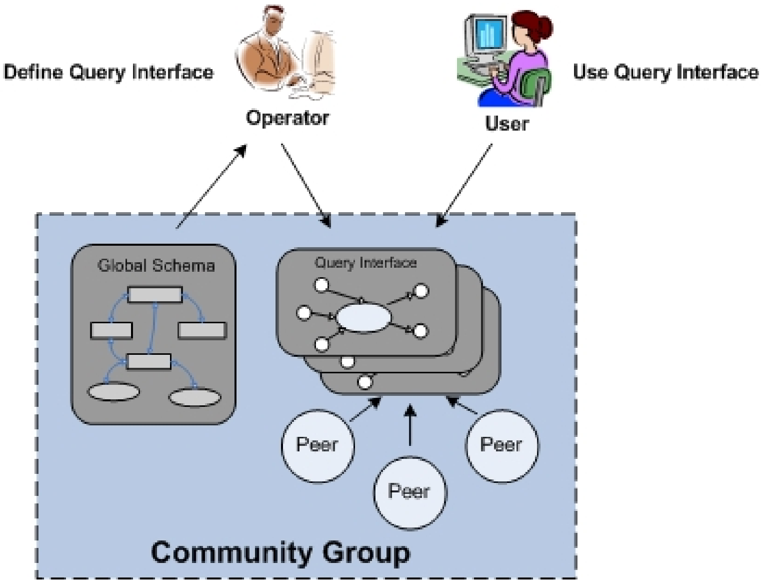


Figure 5.1: The use of query interface

Figure 5.1 illustrates the concept of the query interface. A query interface is a predefined query message with no query constraints but a set of input and output parameters. Each query interface is defined using the concepts selected from a concept hierarchy, and the concept hierarchy is generated from the global ontology maintained at the hub side. The selected concept becomes a parameter attached to the query interface. During query time, the defined query interfaces are displayed to the users as graphical Web-based forms. Users can enter values to the Web forms by following the instruction provided by the form. Values entered by the user are directed back to the query interfaces to create machine understandable query messages. Each query message is defined in the form of SPARQL (Angles & Gutierrez, 2008). Section 6.3 further explains the steps involved in query message generation, in this chapter, we focus on the definition as well as the storage of the defined query interface.

A *Query Interface* (QT) can be formally defined as a 5-tuple
$QT = (N, F, \{P_{in}\}, \{P_{out}\}, \{T\}?)$, where

- N is the *Query Name*;
- F is the functionality provided by the query interface, known as *Functionality*;
- $\{P_{in}\}$ is a list of *Input Parameter* used to initialize a query;
- $\{P_{out}\}$ is a list of *Output Parameter* returned by a query;
- $\{T\}$ is a list of *Triples* used for connecting all the concepts specified in the input and output parameter lists. This attribute is optional when the group of concepts defined from the input and output parameters are already interconnected.

Each input parameter P_{in} is defined as a 4-tuple $P_{in} = (N, V, C, DT)$, where N is the name of the parameter; V is the variable identifier assigned to the parameter, and it is used to uniquely identify a particular parameter; C is the ontological concept associated to the parameter; and DT is the data type associated with the parameter, the data type information is inherited from the defined ontology and it can be any predefined XSD data types (Biron & Malhotra, 2004).

Likewise, an output parameter P_{out} is defined as a 3-tuple $P_{out} = (N, V, C)$, where N is the name of the parameter; V is the variable identifier assigned to the output parameter, and it is also used to uniquely identify a particular parameter; C is the ontological concept associated to the parameter. Unlike input parameters, an output parameter does not contain any data type information, since the data type of all the output parameters are considered as “Literal”.

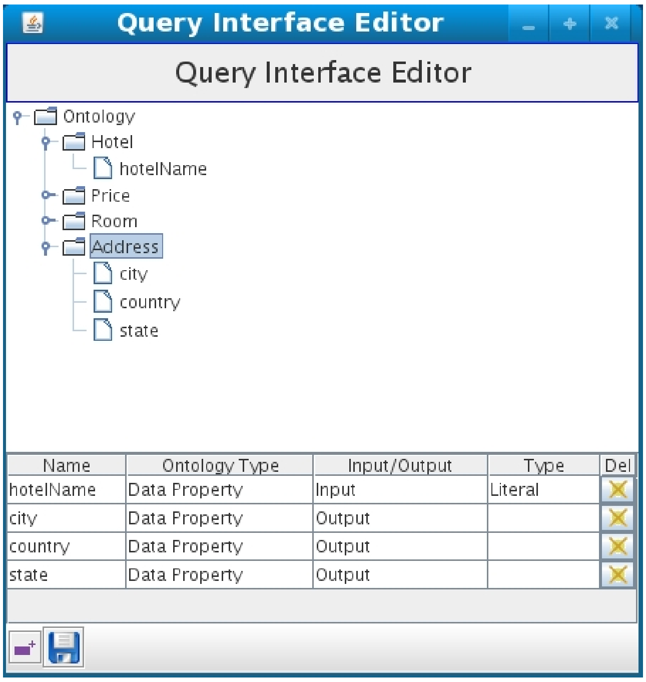
Each triple T is defined as a 3-tuple $T = (D, P, R)$, where D represents the domain of the triple; P represents the property connecting the concepts D and R; and R represents the range variable of the triple. Each triple can be considered as a link that connects two neighborhood conceptual nodes, and the list of triples together forms an integrated ontology graph.

The definition of a query interface is achieved by two sequential steps: 1.) Parameter Selection; 2.) Graph Formation. The specification of query parameters as well as other query related information is performed by human users via graphical tools, whereas the formation of an ontology graph for the query interface is automatically performed by the developed tool.

5.3.1 Parameter Selection

To assist the definition of query interfaces, we developed a graphical tool called QIE (Query Interface Editor). As shown in figure 5.2, QIE is consist of two components: an ontology module that displays the global ontology in the form of a concept hierarchy, and a query interface module that shows relevant query interface information, including query name, query functionality and the set of query parameters.

During parameter selection, an information operator specifies the name and the functionality of the query interface, and he/she also decides on the set of parameters used for the query. For example, a hotel address query interface has the name “hotel address search”, and provides the functionality “find the address for a given hotel”. The query interface should at least have the input parameter “hotelName” and the output parameter “Address”. Both input and output parameters are selected from the concept hierarchy displayed in the ontology module.



Name	Ontology Type	Input/Output	Type	Del
hotelName	Data Property	Input	Literal	☒
city	Data Property	Output		☒
country	Data Property	Output		☒
state	Data Property	Output		☒

Figure 5.2: Query interface editor

However, since the ultimate goal of query processing is to retrieve relevant content from given information sources, and ontology entities such as classes or object properties do not carry data content but rather the structure and conceptual information defined between relevant ontological concepts. Hence, we do not allow the definition of query parameters using object properties. In addition, parameters defined using ontology classes are normalized down to a set of datatype properties, and the reason is because datatype properties are the major conceptual components used for carrying content information.

For example, the ontology class “Address” illustrated in figure 6.2 is normalized down to a set of three datatype properties including “street”, “city”, and “state”. Each datatype property carries partial information for the class “Address”, and the three properties together forms the class “Address”. In other words, given a parameter P_a = *(’Hotel Address’, ’?out1’, ’Address’)* defined using the ontology class "Address", the parameter is normalized down to three parameters written as P_1 = *('Hotel Street', '?out1', 'street')*, P_2 = *('Hotel City', '?out2', 'city')* and P_3 = *('Hotel State', '?out3', 'state')* respectively. In this thesis, the process of normalizing a parameter defined using an ontological class down to a set of parameters defined using data type properties is called Parameter Normalization. Parameter normalization is automatically performed by QIE.

After the selection of query parameters, a graph is formed to link together the set of selected parameters. The goal is to enable automatic query message generation for the defined query interface. The graph formation process is also automatically performed by QIE, and the process itself is further discussed in Section 6.3.

5.3.2 XML Query Interface

Each query interface is stored as an XML document. We choose to use XML for the following reasons: 1.) XML is a widely adopted standard for storing and exchanging data on the Web, and it is also suitable for modeling the nesting structure of a query interface and its containing elements; 2.) More importantly, the availability of XML related standards such as XSLT (Braga *et al.*, 2005) allows the fast transformation of XML document into various data formats such as HTML, including web forms that are written in HTML. Hence, the use of XML allows the quick transformation and manipulation of query interfaces.

The XML Schema defined for query interfaces is shown below. In the defined schema, each query interface contains an element "*template*", formed by "*name*", "*functionality*", and "*triples*". The element "*triples*" is a list of triple with type "*LocalTriple*". The schema is defined following the formal definition provided in early parts of this section.

```
<?xml version=”1.0”?>
<xsd:schema xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
targetNamespace=”aadx:template”
xmlns:tpl=”aadx:template”>
<xsd:element name=”template” type=”tpl:QueryTemplate”/>
<xsd:complexType name=”QueryTemplate”>
<xsd:sequence>
<xsd:element name=”name” type=”xsd:string” minOccurs=”1” maxOccurs=”1”/>
<xsd:element name=”functionality” type=”xsd:string” minOccurs=”1” maxOccurs=”1”/>
<xsd:element name=”triples” type=”tpl:LocalTriple” minOccurs=”0” maxOccurs=”1”/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name=”LocalTriple”>
<xsd:sequence>
<xsd:element name=”triple” type=”tpl:Triple” minOccurs=”1”
maxOccurs=”unbounded”/>
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name=”Triple”>
<xsd:sequence>
<xsd:element name=”domain” type=”xsd:string” minOccurs=”1” maxOccurs=”1” />
<xsd:element name=”property” type=”xsd:string” minOccurs=”1” maxOccurs=”1” />
<xsd:element name=”range” type=”xsd:string” minOccurs=”1” maxOccurs=”1” />
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Using the defined schema, each query interface is stored in the form of an XML document, and the “*hotel address search*” query interface is written as follow.

```
 <?xml version=”1.0”?>
<x:template xmlns:x=”aadx:template”>
<name>hotel address search</name>
<functionality>find the hotel address for a given hotel</functionality>
<triples>
<triple>
<domain>?tmp1</domain>
<property>aadx:hotelName</property>
<range>?in1</range>
</triple>
```

```
<triple>
<domain>?tmp2</domain>
<property>aadx:state</property>
<range>?out2</range>
</triple>
<triple>
<domain>?tmp2</domain>
<property>aadx:country</property>
<range>?out3</range>
</triple>
</triples>
</x:template>
```

The first part of the document captures the information specified by the information operator, which includes query name, query functionality. The document stores the list of identified local triples, and each local triple is documented in the form of Domain, Property and Range, and each local triple represents a connection between two adjacent ontology concepts. All local triples together form an integrated ontology graph.

5.3.3 Display Query Interface

The defined query interfaces are transformed into graphical web forms written in HTML. To enable the fast transformation of XML into HTML, we choose to use XSLT for transforming query interfaces into web forms. Each query interface is transformed into a HTML form, and the name of the template is transformed into heading. In addition, the functionality of the template is transformed into paragraph, and its inputs are transformed into input fields. The transformation process is shown below.

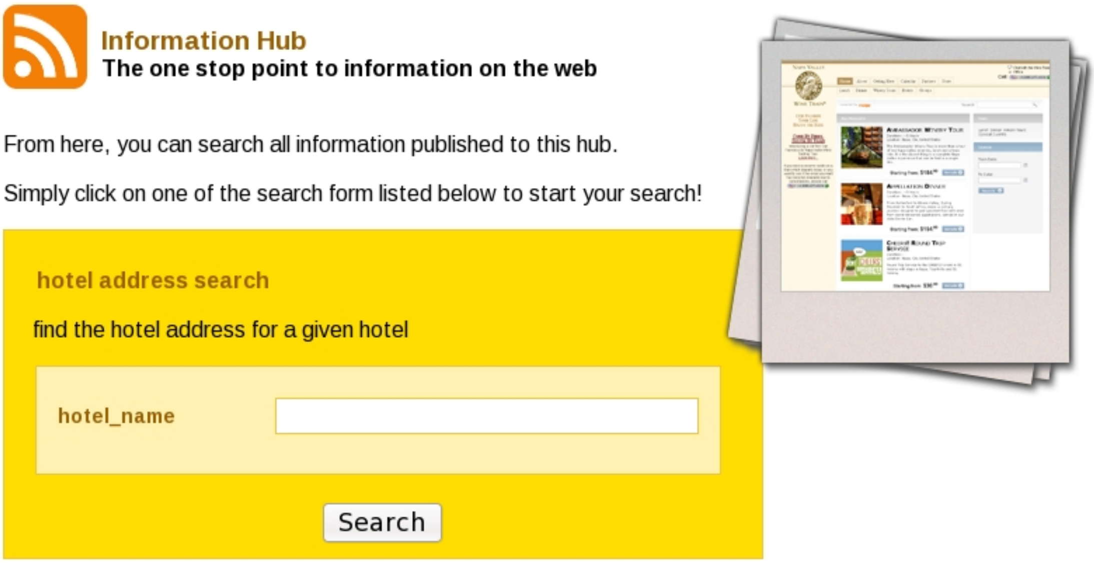
```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/template">
```

```
<form>
<div>
<h3><xsl:value-of select="name"/></h3>
<p><xsl:value-of select="functionality"/></p>
<xsl:for-each select="inputs/parameter">
<fieldset>
<label for="<xsl:value-of select='variable'/>">
<xsl:value-of select='name'/>
</label>
<input type="text" name="<xsl:value-of select='variable'/>"/>
</fieldset>
</xsl:for-each>
<input type="submit" value="Search" name="submit" />
</div>
</form>
</xsl:template>
</xsl:stylesheet>
```

Figure 5.3 shows the graphical web form generated for the hotel address search example. From the web form, users can issue complex query messages by entering values into corresponding input fields. To prevent users from entering invalid query values, the entered values are tested against their associated data types. For example, an integer can only contain the letters from 0 to 9. The input validation process is performed using regular expressions (Owens *et al.*, 2009). Since the focus of our research is the generation of query messages, we do not discuss the validation of user inputs in further details, and we assume that all the values entered by the user are correct and compliant to the data type associated with each parameter.

5.4 Conclusion

This chapter further explains the proposed integration process. The process is formed by a publish-combine-use cycle, where each part of the cycle is performed



Information Hub

The one stop point to information on the web

From here, you can search all information published to this hub.

Simply click on one of the search form listed below to start your search!

hotel address search

find the hotel address for a given hotel

hotel_name

Figure 5.3: Graphical web form

by a unique type of users. The proposed process provides a more flexible and more systematic way of sharing information. In order to seamlessly connect the activities involved in the integration process, and to assist various integration tasks, we introduced two concepts. The concept of information group serves two purposes. First, it allows the classification of the published information sources into meaningful groups, so that the published source can be managed in well-organized forms. Second, information sources can be grouped together according to different information needs. In addition, the concept of query interface is used to connect the combine and the use processes. On one hand, hub operators define query interfaces to create unified views over the published information sources. On the other hand, information users use the defined query interface for accessing the published information. In the next chapter, we will look at the query layer of the proposed integration framework.

Chapter 6

Query Processing

In Chapter 4, we have explained about how the semantic layer is formed via schema mappings. The ultimate goal of the semantic layer is to establish a semantically interoperable network by defining schema mappings among heterogeneous information sources. In Chapter 5, we have introduced the concept of query interface. To a certain extent, a query interface can be considered as a programmatic interface that connects the information users with the semantic network. In this chapter, we discuss some of the techniques developed to automate the task of query processing. The developed techniques are based on the information provided by both the semantic layer and the process layer. Query processing on the query layer is performed following the centralized integration approach. To avoid sophisticated query rewriting, we propose a bottom-up query execution approach. In our approach, data from heterogeneous XML formats are transformed into the global ontology format, and then aggregated together before query execution. The remaining sections of this chapter is organized as follows. Section 6.1 gives an overview of the query layer. The selected query language is introduced in section 6.2. Section 6.3 discusses the query message generation process. All query processing related tasks are explained in section 6.4, followed by conclusion in section 6.5.

6.1 Query Layer Overview

As introduced in Chapter 3, the query layer provides basic query functionalities, which include the issue of query messages, query translation, data translation, and query execution. The ultimate goal of the query layer is to provide query capabilities over the integrated information sources, and to enable the collaboration among all parties involved in the online accommodation domain. To achieve this goal, we need to resolve a number of research issues. First, we need to automate the process of query message generation. Second, we need to develop an effective query processing method.

How to generate a query message?

As previously mentioned, query interfaces are defined to assist the issue of complex query messages. Each query interface is displayed as a graphical form, where information users can enter values in the form by following the instruction provided by the interface. Therefore, the automatic translation of user-entered value into a well-formed query message becomes the first challenge.

How to resolve the generated query?

Another challenge is the processing of the generated query message. In the proposed framework, query processing is performed in a hybrid fashion. In most cases, the query message is resolved by peers from one community group. However, information users can choose to use a number of neighbourhood community groups for query processing.

Figure 6.1 shows the major tasks involved in query processing. When a query message Q is issued to the community group G_n, G_n tries to resolve the received

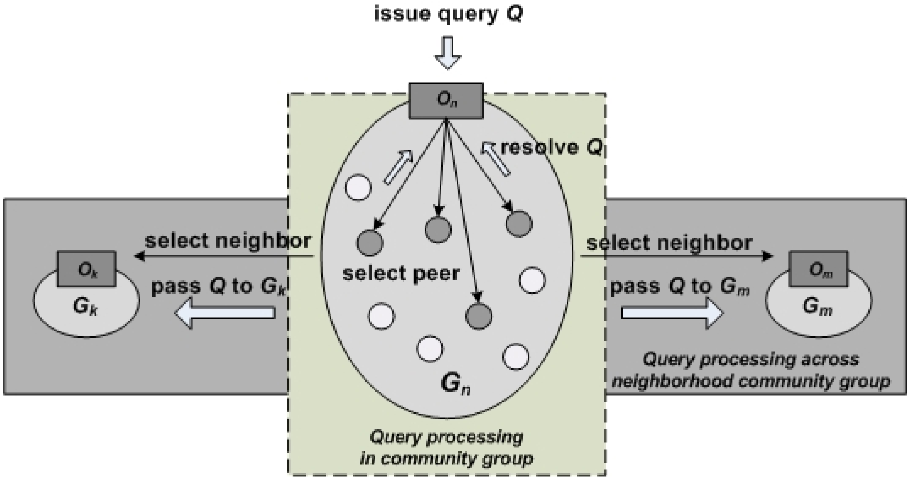


Figure 6.1: Query processing

query using its connecting peers. As shown in the dashed box in figure 6.1, G_n firstly tests the query Q against its local mappings, and selects on the set of peers that are able to resolve the received query Q. The selection process is called peer selection. Once the set of peers are selected, G_n then translates the selected information peers into a common format that conforms to the global ontology format O_n , and combines them into one ontology dataset. Finally, the combined dataset is used to resolve the received query message Q. We call this process as the bottom-up query processing approach.

In some cases, the query results produced from the community group G_n do not meet the users' requirements, i.e. no result is generated from G_n. Users need to initiate another process to resolve the issued query in a much boarder scope. In those cases, the issued query message Q is passed onto the neighbouring community groups. To do this, G_n needs to identify the groups of neighbourhood community groups that are able to answer the received query, and the neighbourhood identification process is called neighbourhood selection. Neighborhood selection is achieved by comparing concepts from the received query Q with the set of global mappings defined between a community group G_n and its neighbours $\{G_k, \ldots, G_m\}$. Once the set of neighbourhood is determined, it (G_n) then translates the query from the source format (conform to ontology O_n) to its target format (conform to ontology O_m), and passes the query to G_m. We name the process of translating query message from one ontology format to another ontology format as query translation. When G_m receives the query, it resolves the query Q using its own connecting peers. If no answer is produced from G_m , the query Q is then further passed onto other community groups, and this process repeats iteratively until some results are generated from the query or a pre-configured iteration limit is reached.

Tasks involved in neighbourhood query processing can be sum up as four iterative steps, including *peer selection*, *query resolving*, *neighbourhood selection* and *query translation*:

- Peer Selection – the initial step in query processing is to select the set of information sources that are capable to resolve the issued query. Given a query message Q and a community group $G_i - (O_i, S_s, m_i)$, then the process of peer selection can be considered as the process of comparing the concepts defined from the query Q against the concepts defined in the local mapping m_i.

- Query Resolving – query resolving refers to the process of resolving a given query Q using the selected set of information peers. The process follows a bottom-up approach, where data from each selected local sources are translated into a generic global format, which then aggregated together and executed against the query Q.

- Neighborhood Selection – if one community group can not answer the issued query, then the query message is passed onto other neighbourhood community groups for further processing. In order to pass the query Q, Q needs to be compared against the global schema mappings M_s that are defined between neighbourhood community groups. This process is known as the neighbourhood selection. Neighborhood query test identifies the set of community groups where a given query should be passed onto.

- Query Translation – upon the detection of a potential neighbourhood community group, the query is then translated from the source format to the destination format. This process is called query translation.

6.2 Query Language

Query messages are formulated using portions of SPARQL (SPARQL Protocol And RDF Query Language) (Prud'hommeaux & Seaborne, 2008). We choose to use SPARQL for the following reasons. 1.) In each community group, query messages are formulated using the terms and concepts defined from the global ontology model. Since SPARQL is specifically designed for querying ontology sources (Prud'hommeaux & Seaborne, 2008), it allows the simple formation of complex query messages using the concepts defined from the global ontology. 2.) In addition, compared to other ontology query languages such as RQL (Karvounarakis *et al.*, 2002) or OWL-QL (Fikes *et al.*, 2003), SPARQL more emphasizes the selection of information as opposed to the manipulation of information, thus it is suitable for the integration of heterogeneous information where data manipulations such as deletion or update are not crucial. 3.) More importantly, a SPARQL query can be executed against a set of different ontology graphs known as the RDF Dataset (Manola & Miller, 2004), thus allowing the fast gathering of information from multiple ontology sources. In our case, this feature is used to integrate on a large number of similar ontology instances. 4.) Finally, each constraints defined in SPARQL is represented in triple form, and this is beneficial for efficient query execution, as SPARQL is implemented by various research and commercial efforts such as JENA (Carroll *et al.*, 2003). Due to the above mentioned reasons, we choose to use SPARQL as the primary language for query processing.

6.2.1 SPARQL

SPARQL queries are formulated following the Prefix-Select-From-Where format which is similar to the traditional Select-From-Where format. In contrast to the conventional query languages, SPARQL more emphasizes on the data selection power, and it does not provide data manipulation features such as deletion or update. In addition, SPARQL supports four query forms including SELECT, CONSTRUCT, ASK and DESCRIBE. The SELECT and ASK query forms are mostly used for pattern matching, whereas the CONSTRUCT and DESCRIBE query forms can be used to form RDF graphs over the result sets. On the other hand, both the SELECT and the CONSTRUCT forms directly query the content of the resources, whereas the ASK form is used to test whether or not a query pattern has a solution. In addition, the DESCRIBE query form is used to query the metadata information about the resources. Since our research focuses on the integration of heterogeneous resource content, and we do not consider the query of metadata level information such as the description of resources, therefore the DESCRIBE form is eliminated from the query language.

All these processes together forms the query processing iteration cycle, and each community group performs this cycle on a query driven basis to provide query processing power for the information integration framework, and each cycle is triggered by the reception of a query message. Section 6.4 further explains the above mentioned processes.

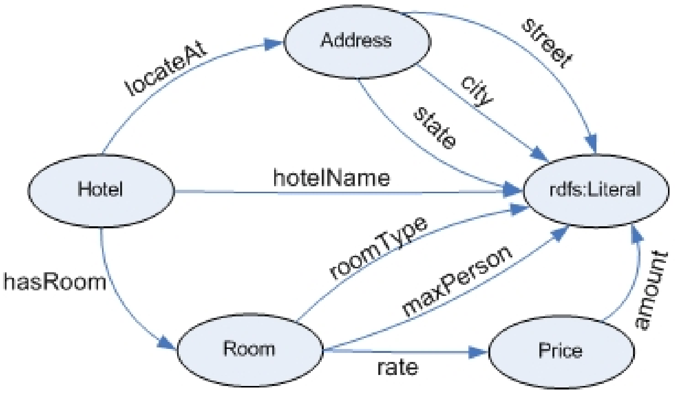


Figure 6.2: Hotel ontology graph

The modified SPARQL query syntax consists of four parts, a prefix clause that specifies the data source context, a *select/construct/ask* clause that specifies the resulting objects, a *from clause* that specifies the graphs used to form the result, and a *where clause* that defines the conditions a query need to satisfy. The *from clause* is optional in the case that only one ontology graph is used for query processing, and the *where clause* consists of a conjunctive number of triple patterns. The following example shows the query "select all the hotel name and address for hotels in Sydney" in SPARQL format, the query is formulated using concepts defined from the ontology shown in figure 6.2.

```
Q: Prefix    aadx: <http://www.aadx.org/.../tourism.owl#>
   Select    ?var1 ?var2
   Where     { ?tmp2 aadx:hotelName ?var1 .
               ?tmp1 aadx:street ?var2 .
               ?tmp1 aadx:city 'sydney' .
               ?tmp2 aadx:locateAt ?tmp1 . }
```

In the above example, the prefix clause specifies the namespace bindings. It contains the statement "*Prefix*", the binding variable "*aadx:*", and the namespace "*<http://www...tourism.owl#>*". Once the prefix binding is declared, all the namespaces used inside the query message can be replaced by the declared binding variables. For example, the relationship "*http://www...tourism.owl#hotelName*" can be written as "*aadx:hotelName*". The select clause "*Select ?var1 ?var2*" identifies the resulting objects returned by the query Q. It contains the statement "*Select*" and a set of global query variables "*?var1*", "*?var2*". Since only one ontology graph is used inside the query, the from clause can be neglected. However, in the case where more than one ontology graphs are used for query processing, the from clause must specify all the involved ontology graphs. At last, the where clause defines the conditions that a query must satisfy. The where clause consists of a "*Where*" statement, a set of triple patterns and a filter clause. Each triple pattern is defined in the form of Subject, Predicate, and Object, where Predicate represents the relation defined from the domain Subject to the range Object. The filter clause defines further constraints that a query must satisfy, and these constraints mainly include the atomic predicates (isURI, isLITERAL) and the relational predicates ($=$, $! =$, $>$, $<$, $\leq$, $\geq$).

6.2.2 Triple Pattern

Each SPARQL query contains a set of triple patterns known as the basic graph pattern, and the defined graph patterns are used to compare against the input ontology graph, searching for the set of relevant sub-graphs that matches the defined triple pattern. Hence, the triple pattern directly affects the result generated from the query. As previously mentioned, a triple pattern consists of three parts: Subject, Predicate, and Object. A Subject is either a query variable defined in the select clause (such as "*?var1*" or "*?var2*") or a temporary query variable that is declared to link two unconnected objects (e.g. "*?tmp2*"). Each declared query variable represents a concept defined in the ontology, for instance, the variable "*?tmp2*" represents the concept "*Hotel*". A Predicate is the relationship defined between the Subject and the Object, and it is either an object property or a data type property defined from the ontology. An Object represents the value attached to the subject, and it can be either a query variable or a constant. In this thesis, we classify the types of triple patterns into two groups: Global Triple and Local Triple.

6.2.3 Global Triple

Global triple pattern represents the query conditions specified by the user. In the previous example, the user issues a query to "*Select all hotel names and addresses in Sydney*", this leads to the generation of three triple patterns: "*?tmp2 aadx:hotelName ?var1*", "*?tmp1 aadx:street ?var2*", "*?tmp1 aadx:city 'sydney'* ". The first two patterns specifies relevant parts of the result generated from the query, and the third pattern defines the constraints that a query must satisfy. Each defined triple pattern represents a condition that a query must satisfy.

6.2.4 Local Triple

However, in most cases, concepts or relations appeared in the defined triple patterns are not linked. In the previous example, the concept "*city*" and the concept "*hotelName*" are not directly linked in the defined ontology, as shown in figure 6.3. Leaving these triple patterns unconnected will lead to a loosely defined set of query conditions, and may produce incorrect or imprecise query answers. Hence, to ensure that an accurate set of query results can be produced from a given set of triple patterns, the defined patterns must be linked together to form a connected ontology sub-graph, and each defined pattern is applied on top of another to produce a single conjunctive query condition. Continuing on the previous ex-

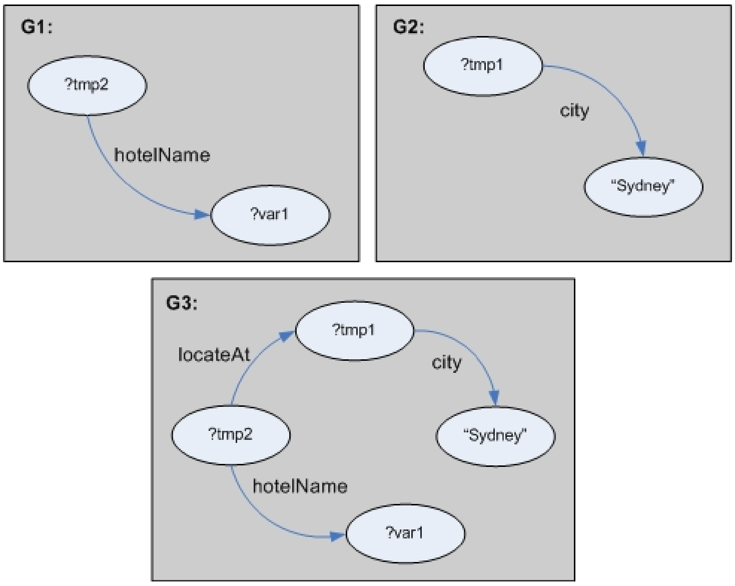


Figure 6.3: Ontology sub-graphs

ample, the triple patterns "*?tmp2 aadx:hotelName ?var1*" and "*?tmp1 aadx:city 'sydney'*"' represent the ontology graphs G_1 and G_2 respectively, as shown in figure 6.3. However, given the ontology defined in figure 6.3, it is clear that G_1 and G_2 are not connected, thus they are not treated as one single conjunctive query condition. Instead, the two triple patterns are used as two separate query conditions, which produces a set of less precise and loosely formed query answers.

To connect the graph G_1 with G_2, we need to identify the set of concept nodes between G_1 and G_2. In the ontology graph, the concept "*locateAt*" is defined to connect the concept "*Hotel*" with the concept "*Address*". It also can be used to connect the graph G_1 with G_2. As a result, a local triple pattern is defined "*?tmp2 aadx:locateAt ?tmp1*", linking together the two unconnected ontology graphs. The next section further explains how the values entered by the user are automatically translated into the SPARQL form.

6.3 Query Message Generation

In earlier part of this chapter we have mentioned that the initial task of the query layer is to turn the issued query request into well-formed query messages, and this is achieved through the process of query message generation. The query message generation process is consisting of two sub-processes: 1.) graph formation; 2.) message generation. The first process is to define a well-formed query interface. As described in Section 5.3, each query interface is defined by selecting concepts from the community group's global ontology, and the defined interface is used for generating a particular type of query message. However, the selected concepts are often not connected, thus unable to produce a well formed query message. To solve this issue, we need to find the set of local triples that connects together the selected parameters. The identified local triples are used to form an integrated ontology subgraph for message generation. In the second process, values entered by the user are fed to the query interface to generate the final query message. Query messages are formulated using the SPARQL language.

6.3.1 Graph Formation

As mentioned in section 5.3.1, the parameter selection process specifies the basic information relating to a predefined query template, including query name, query functionality, and input/output parameters. However, this information all together is not sufficient for forming a well formulated query message. In the previous section, we mentioned that the parameter information specified by hub operators can be used to form a set of global triple patterns, each represents a sub-component of the ontology graph. However, these scattered ontology graphs are often unconnected, and can not be directly used to form an integrated query constraint, thus leading to imprecise formulation of query messages. Therefore,

Algorithm 5 FindTriple4Class($O, C_n, C_m, R = \emptyset$)

```
Input: a global ontology $O = (\{C\}, \{OP\}, \{DP\})$;
Input: two ontology classes $C_n$, $C_m$.
Output: set of local triples: $\{T\}$.
if $C_n == C_m$ then
  return $R$;
end if
for all $OP_i \in \{OP\}$ do
  $OP_i = (D_i, N_i, R_i)$;
  if $D_i == C_n$ or $R_i == C_n$ then
    if $OP_i \in R$ then
      continue;
    else
      if $D_i == C_n$ then
        $R$ = FindTriple4Class($O$, $D_i$, $C_m$, $R$) $\cup \{OP_i\}$ ;
      else
        $R$ = FindTriple4Class($O$, $R_i$, $C_m$, $R$) $\cup \{OP_i\}$ ;
      end if
    end if
  end if
end for
return $R$;
```

in order to integrate the set of unconnected sub-graph, and to form an integrated ontology graph that can be directly used for query formulation, we need to identify the set of local triples.

Algorithm 6 FindTriples(O, $\{P\}$)

```
Input: a global ontology $O = (\{C\}, \{OP\}, \{DP\})$;
Input: a set of parameters $\{P|P = (N, V, C, DT?)\}$.
Output: a set of local triples: $\{T\}$.
$R = \emptyset$; // Set of local triples
$C = \emptyset$; // Set of classes
for all $DP_i \in \{DP\}$ // Find the set of classes do
  $DP_i = (D_i, N_i, R_i)$;
  if $N_i \in \{P|P = (N, V, C, DT?)\}$ then
    $C = C \cup \{D_i\}$;
    $R = R \cup \{DP_i\}$;
  end if
end for
for all $C_i, C_{i+1} \in C$ do
  $\{T\}^i_{i+1} = FindTriple4Class(O, C_i, C_{i+1})$;
  $R = R \cup \{T\}^i_{i+1}$;
end for
return $|R|$;
```

Each local triple $T = (D, P, R)$ represents the property that connects two adjacent ontology concepts, denoted as D and R respectively. The connection between two unrelated ontology concepts C_1 and C_3 can be denoted as as set of local triples $\{T_{1 \to 2}, T_{2 \to 3}\}$, where $T_{1 \to 2} = (C_1, P, C_2)$ is the local triple connecting concepts C_1 and C_2, and $T_{2 \to 3} = (C_2, P, C_3)$ is the local triple that connects the concepts C_2 and C_3. Likewise, the set of local triples used to connect two scattered concepts C_n and C_m can be represented as $\{T_{n \to n+1}, T_{n+1 \to n+2}, \ldots, T_{m-1 \to m}\}$. We use the notation $\{T\}^n_m$ to denote the set of local triples connecting two concepts C_n and C_m. The problem of finding a set of triples connecting C_n and C_m is similar to the path finding problem (Yang & Szpakowicz, 1994). Unlike the path finding problem, we do not need to find the shortest distance between two given nodes, thus making the problem less complicated and easier to solve. Given an ontology O and two ontology classes C_n and C_m, the identification of the set of local triples connecting C_n and C_m is shown in algorithm 5 .

The method identifies the set of local triples used to navigate from the source class C_n to the destination class C_m. On the other hand, the identification of local triples for a group of parameters requires further steps, as all the parameters are associated with data type properties but not ontology classes. To connect two parameters, we need to find the group of ontology classes that are associated with the data type properties before applying the triple identification method. As shown in algorithm 6, the identification method first identifies the set of classes that are associated with the data type properties from each individual parameter. It then applies the triple identification method shown in algorithm 5 on to each pair of classes. Finally, redundant local triples are abandoned from the result, and the method returns a unique set of local triples with no redundancy, denoted as $|R|$. All the identified local triples together form an integrated ontology graph, and is used for later query message generation.

6.3.2 Message Generation

Second part of the query generation process is to turn the predefined query interface into a query message written in SPARQL format. In the query interface, each defined output parameter is translated into a part of the select clause. The values collected from end users are translated into triple patterns, written in the form of Domain, Property and Range, where Domain is the variable assigned to the parameter, Property is the associated ontology property, and Range is the value entered by the end user. For example, one triple pattern generated from the previous hotel address search example is written as "*?tmp1 aadx:hotelName 'Novotel Sydney'*". In addition, all the identified local triples are also translated into parts of the query message to form a conjunctive set of query condition. As a result, the following query is generated from the query interface defined in Section 5.3.

```
Select/Ask/Construct          ?out1 ?out2 ?out3
Where           { ?tmp1 aadx:hotelName 'Novotel Sydney' .
                    ?tmp1 aadx:locateAt ?tmp2 .
                    ?tmp2 aadx:city ?out1 .
                    ?tmp2 aadx:state ?out2 .
                    ?tmp2 aadx:country ?out3 . }
```

The issued query message is used to query over the integrated set of information sources. In the next section, we will discuss the processing of the issued query message.

6.4 Query Processing

In Section 6.1, we have briefly mentioned that query processing in the proposed framework involves four iterative steps, including *Peer Selection*, *Query Resolving*, *Neighborhood Selection* and *Query Translation*. In this section, we explain each task in further detail.

6.4.1 Peer Selection

The initial step in query processing is to test the received query message against the defined local mappings. The primary goal is to determine the set of information peers that are capable of resolving the received query. As defined in Chapter 4, a community group G is a 3-tuple $G = (O, S_s, m_s)$, where O is the global ontology shared within the community group, S_s represents the group of peer schema, and m_s is the set of semantic mapping defined between S_s and O. Each semantic mapping $m \in m_s$ is a Ontology Mapping Instance as defined in Section 4.2.2 that models the conceptual relationship between the peer schema S and the global ontology O. From m each peer schema S is associated to a sub-graph of the global ontology O denoted as O_m. In addition, each ontology O can be considered as a set $\sum o$ covering all concepts, properties and relations. Hence, the sub-graph O_m can be considered as a set $\sum m$ such that $\sum m \subseteq \sum o$.

On the other hand, the query constraints defined in the given query Q can be also considered as an ontology graph, and we treat the set of query constraints as a sub-ontology extracted from the global ontology O, denoted as $\sum Q \subseteq \sum o$. Hence, we say that Q is resolvable via mapping m, if $\sum Q \subseteq \sum m$.

As a result, each local mapping m is tested against the query Q and the condition $\sum Q \subseteq \sum m$. If the condition can be satisfied by the mapping m, then the local peer P that is associated with the mapping m is considered as a potential candidate for resolving the received query Q. Results generated from the peer selection are passed onto the next stage of query processing, which is query execution.

6.4.2 Query Resolving

As mentioned in Chapter 2, conventional LaV approaches use local views for resolving a given query defined in the global format (e.g. Kirk *et al.*, 1995). In those approaches, the issued query message is decomposed into a set of sub-queries that are resolvable by individual sources. The results gathered from individual information sources are combined together to form an integrated result. One major drawback of this approach is the high complexity of the query rewriting process, known as the view-based query rewriting. While view-based query rewriting has been a well studied research topic in the database domain, it remains as a challenging issue in the semantic web community. The problem of how to resolve an ontology query using views defined in XML schema remains unsolved. Hence, to avoid complex query rewriting, we use an alternative approach for resolving ontology queries using XML data sources.

Figure 6.4 illustrates the proposed approach. In our approach, we firstly decide on the set of information peers that are capable of answering the issued query. This is achieved by comparing the concepts defined from the query message against the concepts defined from local schema. Then the selected information peers are transformed from their local formats to the global ontology format. The goal is to eliminate data heterogeneity exist among individual information sources. Finally, the transformed sources are used as a single data set for resolving the issued query. In the proposed approach, instead of decomposing a large ontology query into sub-queries, we combine the local peers to an integrated data set, and then resolve the global query using the integrated data set. Since our approach transforms data from local formats to the generic global format as oppose to the conventional transformation of a global query into local queries, we call our approach as the bottom-up approach.

The backbone of the proposed query solution is the transformation of data from local XML formats to the global ontology format, and this is performed based on the semantic mapping defined on the schema level. In chapter 4, we have introduced the schema mapping and the data transformation process. Here, we use a real world example to further demonstrate the proposed transformation process. Assuming that two local XML schemas (S_1 and S_2) are selected for a given query Q and each selected schema possess a different data structure as shown in figure 6.5. A domain ontology O_s is defined to model the key concepts exist in the hotel domain.

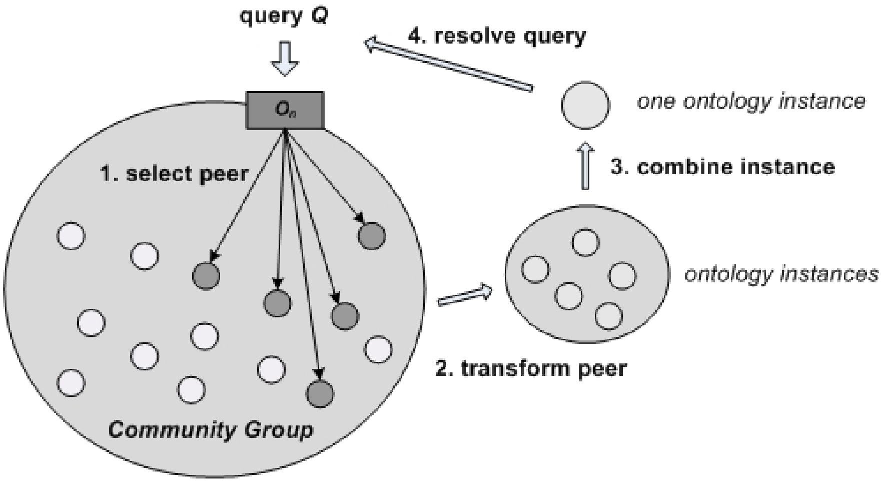


Figure 6.4: Bottom-up query resolving

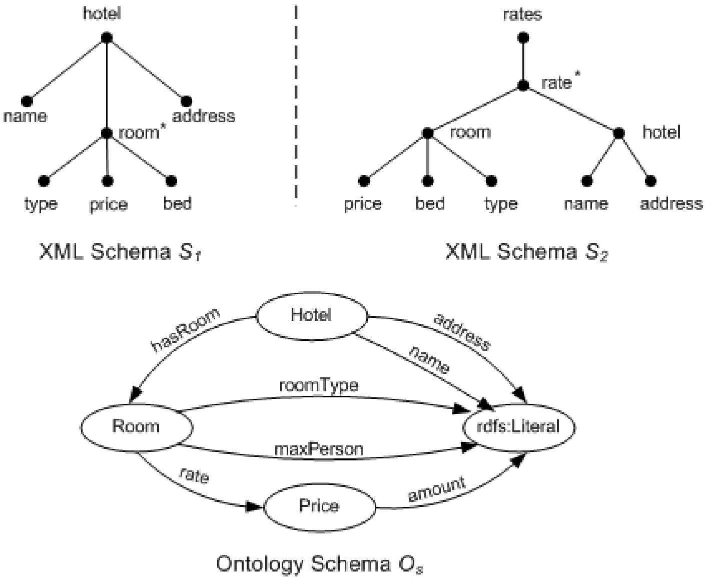
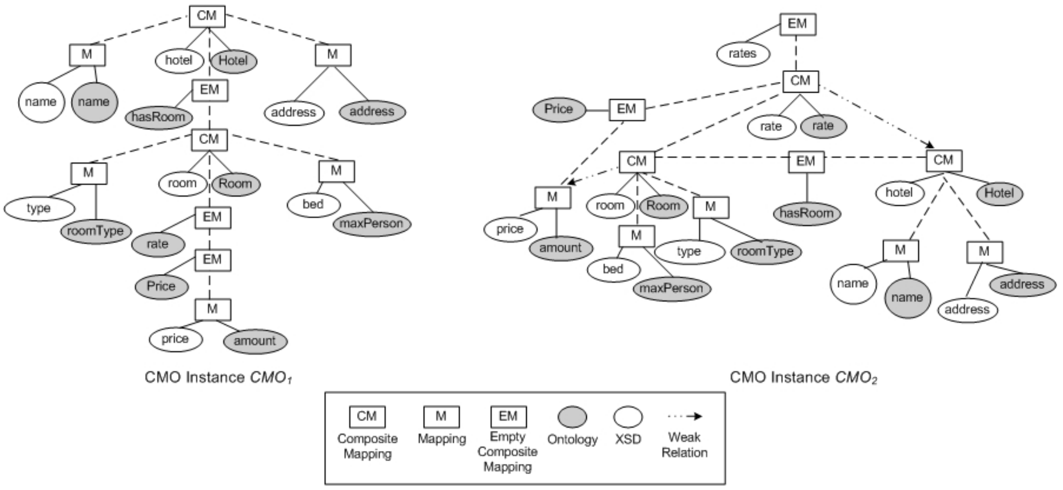


Figure 6.5: Hotel rate schema example



CMO Instance *CMO₁*

CMO Instance *CMO₂*

Figure 6.6: CMO instances

During Concept Mapping, both S_1 and S_2 are mapped to O_s and the concept mappings are documented using CMO instances as shown in figure 6.6. CMO_1 represents the concept mapping defined between S_1 and O_s; CMO_2 documents the concept mapping defined between S_2 and O_s. The creation of CMO_1 and CMO_2 follows the rules defined in section 4.2.2.

During Data Translation, the following input XML document X_i is received by $T_{x \rightarrow o}$:

```
 <?xml version='1.0' encoding='utf-8'?>
<hotel>
<name>Novotel Sydney</name>
<room>
<type>Standard</type>
<price>319.00</price>
<bed>1</bed>
</room>
<room>
<type>Deluxe</type>
<price>339.00</price>
<bed>1</bed>
</room>
...
```

<address>100 Murray Street, Sydney, NSW</address>
</hotel>

X_i is compliant to the XML schema S_1 and will be translated into the targeted format that is compliant to S_2. Following the workflow defined in section 4.2.3, X_i is firstly translated into a list structure denoted as X_i = [hotel, name, room, type, price, bed, room, type, price, bed ... address]; For each $E \in X_i$ a corresponding mapping is retrieved from M_i. Starting from the first element "hotel", a mapping is found from COM_1 (figure 6.6) denoted as $CM_{hotel} = (M_{name}, EM_{hasRoom}, M_{address}, AE, OE)$; Since CM_{hotel} is a composite mapping, an ontology graph G_1 is created for CM_{hotel}, as illustrated in figure 6.7-1; G_1 is set as the root graph since a root graph does not exist yet. For the second element "name", the mapping $M_{name} = (AE_{name}, OE_{name})$ is retrieved from COM_1, with OE_{name} = (name: Hotel, type: Class); The $f_p(G_r, M)$ function will then find the set of parent nodes in the root graph for M_{name}, and the $f_c(N_s, M)$ function locates the node N_{name} for M_{name}; Since N_{name} does not have a value yet, sets the value for N_{name}. For the third element "room", the mapping $CM_{Room} = (M_{roomType}, EM_{rate}, M_{maxPerson}, AE, OE)$ is retrieved. A corresponding ontology graph G_2 is created as shown in figure 6.7-3; the generated graph is attached to the pre-existing nodes within the root graph. Repeat the same steps on the rest of the elements in X_i, finally an ontology graph is created as illustrated in figure 6.7-4.

Once the selected data sources are transformed into the global ontology format, we then combine the transformed data sources into one single data set, represented in the form of an ontology instance. As each transformed data source can be considered as an instance of the defined global ontology, all the generated ontology instances can be aggregated together to form an integrated ontology instance. Finally, the integrated ontology instance is used for resolving the issued

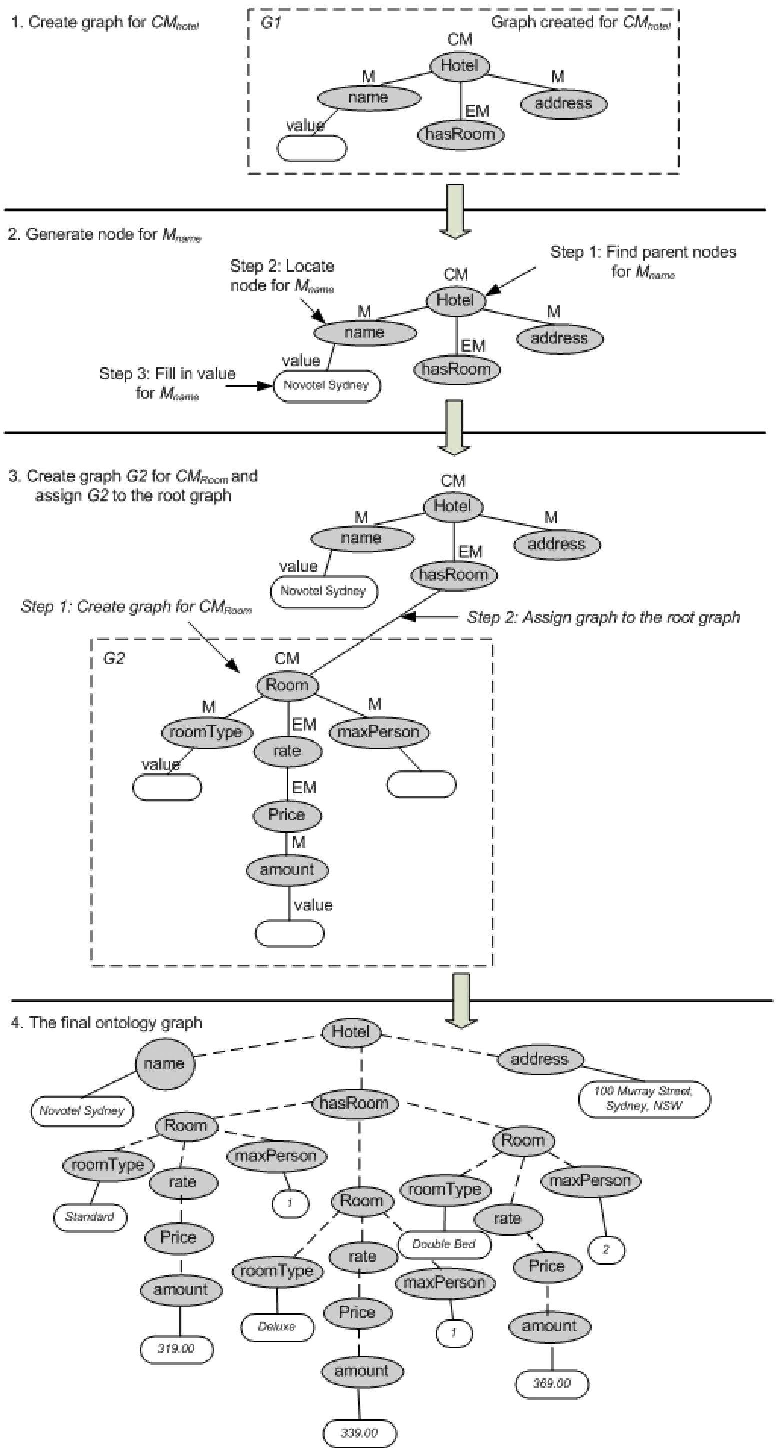

1. Create graph for CM_{hotel}
G1
Graph created for CM_{hotel}
CM
Hotel
M
name
M
address
EM
hasRoom
value
2. Generate node for M_{name}
Step 2: Locate node for M_{name}
Step 1: Find parent nodes for M_{name}
Step 3: Fill in value for M_{name}
Novotel Sydney
3. Create graph G2 for CM_{Room} and assign G2 to the root graph
Step 1: Create graph for CM_{Room}
Step 2: Assign graph to the root graph
G2
Room
roomType
rate
maxPerson
Price
amount
4. The final ontology graph
100 Murray Street, Sydney, NSW
Standard
Deluxe
Double Bed
319.00
339.00
369.00
1
1
2

Figure 6.7: Sample flow for $T_{x \to o}$

query message, and we use Jena[1] for implementing the query engine.

6.4.3 Neighborhood Selection

In some cases, a community group may not be able to provide any answers for the received query. For example, assume that a community group is formed by a group of hotel providers in Sydney, and a query message is issued to collect hotel information in Melbourne. Since no members from the community group are able to provide any information for the issued query, the community group produces empty result for the issued query message. As a result, users need to execute the query in a much boarder scope, where a group of inter-connected community groups are used. The issued query message will be passed onto the neighbourhood community groups for further processing. To do so, the query process needs to find out the group of neighbourhood community groups that are able to resolve the issued query, and this is achieved via neighbourhood selection.

Similar to the process of peer selection, neighbourhood selection also relies on the semantic information captured by the semantic layer, more specifically, the schema mapping defined between a pair of global ontologies. Given a pair of neighbourhood ontology O_i and O_j, the mapping defined between them denoted as M, then the concepts from O_j that are covered by the mapping M can be represented as $\sum_M^j$. To determine whether a given query Q can be resolved by the community group G_j, we need to examine the concept set $\sum Q$ against the concept set $\sum_M^j$. If $\sum Q \subseteq \sum_M^j$, then we say that the query is resolvable by the neighorhood G_j. Both the peer selection and the neighbourhood selection are based on the basic idea of concept comparison (Weinstein & Birmingham, 1999).

[1] See http://jena.sourceforge.net/

6.4.4 Query Translation

To propagate a query from one community group to another, the query needs to be translated from the source ontology format to the destination ontology format. The query translation process can be considered as a function $T_{o_i \rightarrow o_j}(Q_i, M)$, where Q_i is the query message posed against source ontology O_i, and M is the concept mapping defined between source ontology O_i and target ontology O_j. The query translation will lead to the result query Q_j that is compliant to the target ontology O_j. The basic principle used in the translation process is the replacement of source concepts to their corresponding target concepts defined in the concept mapping M.

```
Q: Select    ?out1 ?out2
     Where    { ?tmp1 O1:hotelName 'Novotel Sydney' .
               ?tmp1 O1:locateAt ?tmp2 .
               ?tmp2 O1:city ?out1 .
               ?tmp2 O1:street ?out2 . }
```

To give an illustrative example, we use the query Q from earlier example, and translate it into the format that conforms to the ontology O_2 as shown in figure .

Assume that the following concept mappings are defined between O_1 and O_2. We use the both as view (BAV) approach (Boyd et al., 2004) to define concept mappings between two given ontology schemas. In the single direction mapping such as the $O_1 \rightarrow O_2$ mapping, for each concept from the source ontology (O_1) we define a concept mapping that associates the concept from the source ontology (O_1) to its corresponding concept in the target ontology (O_2). If there is no corresponding concept exists in the target ontology (O_2), we simply assign a null value to the concept, e.g. "$O_1 : locateAt \rightarrow null$".

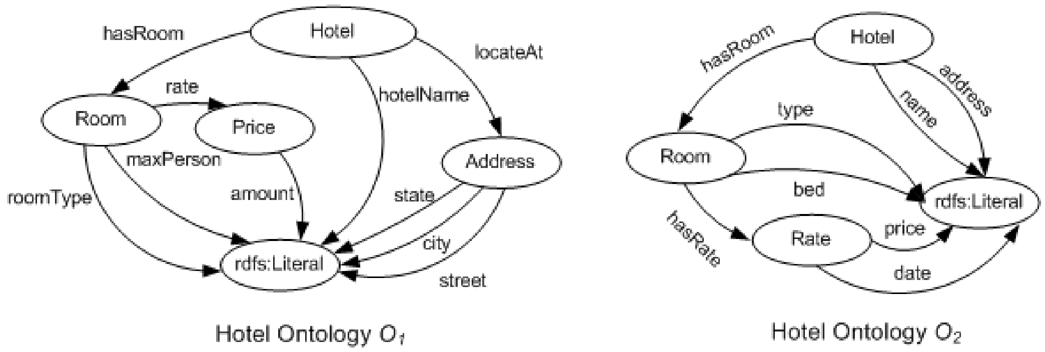


Figure 6.8: Sample hotel ontology

$O_1 \rightarrow O_2$ **Mapping**	$O_2 \rightarrow O_1$ **Mapping**
O_1:Hotel → O_2:Hotel	O_2:Hotel → O_1:Hotel
O_1:Room → O_2:Room	O_2:Room → O_1:Room
O_1:Price → O_2:Rate	O_2:Rate → O_1:Price
O_1:Address → null	O_2:hasCity → O_1:city
O_1:hotelName → O_2:name	O_2:hasStreet → O_1:street
O_1:locateAt → null	O_2:name → O_1:hotelName
O_1:maxPerson → O_2:bed	O_2:bed → O_1:maxPerson
O_1:roomType → O_2:type	O_2:type → O_1:roomType
O_1:amount → O_2:price	O_2:date → null
O_1:city → O_2:hasCity	O_2:price → O_1:amount
O_1:street → O_2:hasStreet	
O_1:state → null	

Table 6.1: Concept mapping between homogeneous ontologies

In the first step, concepts from the source ontology are replaced with their corresponding concepts from the target ontology. This process simply replaces each concept from the input query Q with its corresponding concept defined in the mapping M. For example, the concept "O_1:hotelName" from query Q is replaced with the concept "O_2:name" in query Q'. At the same time, concepts that are mapped to null values are eliminated from the query. For example the triple "?tmp1 O_1:locateAt ?tmp2 ." is eliminated in query Q'. The following query Q' is produced from the initial translation.

Q': *Select* *?out1 ?out2*

Where { *?tmp1* O_2*:name 'Novotel Sydney' .*

?tmp2 O_2*:hasCity ?out1 .*

?tmp2 O_2*:hasStreet ?out2 .* }

In the second step, the translated triples are connected together to produce an integrated ontology sub-graph, and the graph formation process is the same as the process used for query message generation. The basic principle is to relate two unconnected ontology sub-graphs by finding a conceptual path between them. For example, the conceptual path between the graph "?tmp1 O_2:name 'Novotel Sydney"' and the graph "?tmp2 O_2:hasCity ?out1" is the concept "O_2:Hotel", so the query is then translated into Q'' as shown below.

Q'': *Select* *?out1 ?out2*

Where { *?tmp1* O_2*:name 'Novotel Sydney' .*

?tmp1 O_2*:hasCity ?out1 .*

?tmp1 O_2*:hasStreet ?out2 .* }

The generated query is compliant to the ontology model O_2, and another query processing iteration is triggered by query Q'', which follows the steps of peer selection, query resolving, neighbourhood selection and query translation.

6.5 Conclusion

In this chapter we have discussed the query layer of the proposed integration framework. To assist the task of query processing we have developed two query processing techniques, which are the semi-automatic generation of query messages and the bottom-up query processing method. The generation of query messages is dependent on the principle of graph formation, where scattered ontology graphs are connected together to form an integrated ontology graph. On the other hand, query processing on the query layer is collaboratively performed by connecting the set of community groups. Users can decide to use one or a number of community groups for query processing. In the first case, a query is executed using a bottom-up query processing approach, in which data from heterogeneous XML formats are firstly transformed into the global ontology format, and then aggregated together to produce an integrated query data set. The goal of the bottom-up query answering approach is to avoid sophisticated query rewriting processes. In the second case, a query is executed in an iterative fashion, where the issued query message is executed by neighourhood community groups. In the next chapter, we introduce a prototype system called the "Accommodation Hub". The prototype system is implemented following the methods and techniques introduced in the semantic integration framework, and it is used to demonstrate the practical applicability and feasibility of the proposed framework.

Chapter 7

Accommodation Hub - An Accommodation Information Integration Prototype System

In previous chapters, we have focused on the discussion and explanation of the proposed semantic integration framework. The main goal of the proposed framework is to enable information integration in the large, dynamic, heterogeneous and distributed accommodation environment. In this chapter, we present a system developed following the principles and methods introduced in previous chapters. The prototype system named "Accommodation Hub" is specifically designed for the online accommodation domain, and it is based on the semantic integration framework proposed in previous chapters. According to the design science research methodology (Hevner *et al.*, 2004), the developed system is used to "demonstrate the practical applicability of the proposed artifact by applying the artifact into real world domains". The remaining sections of this chapter are organized as follows. Section 7.1 discusses information integration in the online accommodation domain. Section 7.2 introduces the accommodation hub, the prototype system is comprised of three modules: mapping module, query module and interface module. Each module is further discussed in its sub-sections. Several real world scenarios are presented in Section 7.3, and those examples are used to demonstrate the applicability of the accommodation hub system in the online accommodation domain. Section 7.4 gives a conclusion of this chapter.

7.1 Introduction

The online accommodation market is a large market where information technologies have been applied for some time, leading to a plethora of different information sources, each with its own format, structure and provider (Hitz, 2006). As a result, consumers are burdened with finding and visiting various web sites and applications in order to gather all their desired information and products. The situation gets even worse when an increasing number of information sources join and leave the market in unpredictable ways. Besides, information provided by independent information providers is often stored and presented in different formats, which leads to the problem of data heterogeneity. What is therefore required is an information integration solution to provide users with all the information sources available in the online accommodation domain, regardless of their formats and physical location. For example, an end user should be able to collect the hotel price information, weather condition, tourist feedback information and other hotel related information in one place without the need of visiting multiple websites.

On the other hand, information providers involved in the online accommodation domain can be considered as a large network of interlinked information nodes. The structure of the accommodation information network is similar to the structure of the proposed hybrid information structure. Players such as individual hotel providers, online service providers, or individual websites and blogs can be seen as bottom level information peers. Those lower level information peers are connected to players such as large corporate hotel groups, hotel intermediaries, and online distribution channels to form different community groups. Information peers (e.g. individual hotel providers) are connected to their community groups (e.g. corporate hotel group) following a centralized integration structure, whereas neighborhood community groups can be connected in a distributed P2P fashion to provide shared information access. For example, a hotel group in New South Wales can be linked to a hotel group in South Australia to provide shared information access. Therefore, to enable effective information sharing and integration among all involved information players in the online accommodation domain, a solution is required to maintain the existing information structure of the online accommodation, and to establish close working relationships among all the involved information players.

In addition, the online accommodation domain is large and dynamic, and information sources come and go in unpredictable ways. It is therefore not reasonable to expect one single party to take control of all the tasks required for collecting, integrating and managing the large number of heterogeneous information sources. A more effective solution would be to distribute the total effort and cost of information integration to individual players, and mobilize all the parties involved in the online accommodation domain. For example, individual information providers such as hoteliers can take the role of publishing their information sources online, as it will help them to boost their public awareness while allowing more control of their own information sources. On the other hand, hotel intermediaries are more effective in integrating and managing the connected information sources, and this allows them to improve their service quality by providing comprehensive information to their customers. Hence, the proposed publish-combine-use mechanism can be well adopted into the online accommodation domain. The proposed mechanism not only reduces the effort and cost of large-scale information integration, it also offers more flexibility and scalability as information sources can join and leave the network according to their own interests.

Due to the above mentioned reasons, the proposed semantic integration framework can be considered as a well-matched solution for information integration in the online accommodation domain. As explained in preceding chapters, ideas proposed in this thesis can be summarized as three concepts: ontology mediation, publish-combine-use integration process, and hybrid integration. The ontology mediation theory can be used for solving issues such as data heterogeneity, and to provide one stop access over heterogeneous information sources. The publish-combine-use mechanism provides a flexible solution that involves all players in the online accommodation domain in achieving large-scale information integration. Finally, the concept of hybrid integration introduces a hybrid information integration architecture that fits neatly with the information structure of the online accommodation domain.

7.2 Accommodation Hub

In this section, we introduce the web-based prototype system developed for the online accommodation domain. The prototype system is implemented following the theories and methods proposed in previous chapters. In previous chapters, we have introduced the principles and methods developed for the semantic integration framework, which include ontology mediation, hybrid integration, and publish-combine-use mechanism. Ontology mediation acts as the fundamental principle for integrating online information sources, where heterogeneous information sources are integrated through the mediation of a global ontology. The principle of hybrid integration defines the behavior of the proposed framework, where collaborations among individual information sources are performed in both the centralized and distributed fashion. Finally, the publish-combine-use mechanism shows the interaction among heterogeneous information parties. We implement all three concepts together in one prototype system, called the "Accommodation Hub". We choose to use the word "hub" for naming our prototype system, because the behavior of the system is similar to the functionality provided by a physical hub device, and the system can be consider as an one-stop access point to a number of connecting information sources.

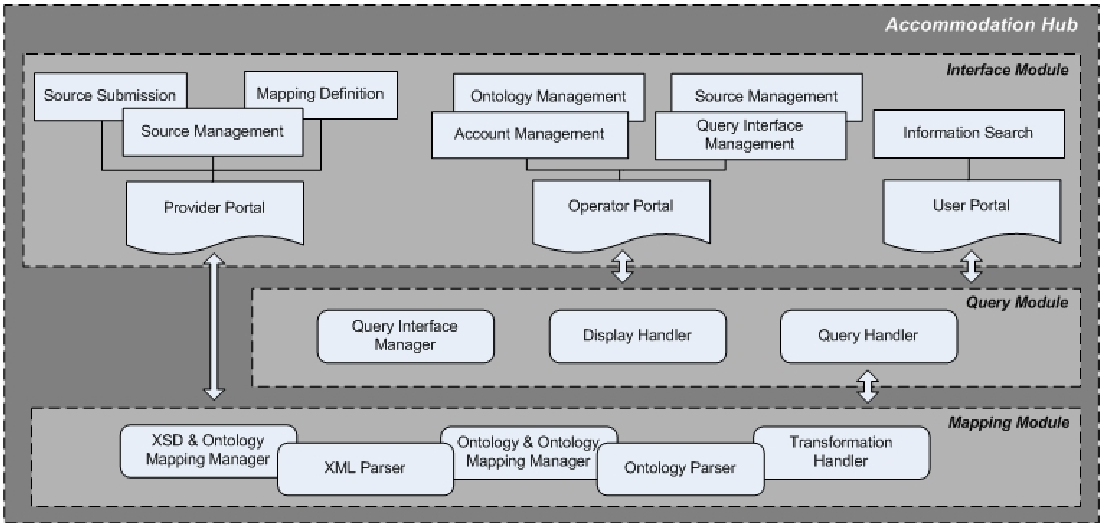


Figure 7.1: Accommodation hub

Figure 7.1 gives an overview of the accommodation hub. Starting from the bottom of the architecture, the system is comprised of three modules: mapping module, query module and interface module.

- The mapping module implements all features concerning with the ontology mediation theory, which includes the definition of schema level mappings, and the transformation of instance level data. Since the ontology mediation theory provides fundamental principles for enabling information integration, the mapping module also acts as the foundation for the prototype system.

- On top of the mapping module is the query module. The query module provides various query processing functionalities, including query interface definition, query generation, query execution, peer selection, neighborhood selection and query translation. Each function is evolved from the hybrid integration theory, and some of them rely on the methods implemented by the mapping module. For examples, the query execution function uses the data transformation method implemented in the mapping module.

- The interface module implements a set of functionalities following the publish-combine-use mechanism, each integration process is implemented as a separate web portal. The provider portal is implemented to allow the publication of information sources. The operator portal provides integration functionalities, and the user portal provides integrated information access. Each web portal provides a set of graphical user interfaces to assist the information integration process.

All three modules together form the accommodation hub. The hub prototype itself is a web based application system implemented using the J2EE[1] technology, and each system module is implemented using Java. However, parts of the mapping module are implemented in Perl using the WordNet perl API[2]. In the following sections, we discuss the implementation details of each system module. We start from the bottom of the system, as the upper modules are developed based on the functions provided by the lower modules.

[1] Java 2 Platform, Enterprise Edition (J2EE)

[2] See http://wn-similarity.sourceforge.net/

7.2.1 Mapping Module

As illustrated in figure 7.2, the mapping module contains five major system components: XML parser, ontology parser, XSD and ontology mapping manager, ontology and ontology mapping manager, and transformation handler. The XML parser is implemented using the standard Java parser, where a XML document is parsed into a DOM tree. The ontology parser is built using the Java OWL-API[1] library, the main reason is because OWL-API can be integrated into online application such as Java Applet. Through the ontology parser, an ontology document written in OWL is parsed into a virtual OWLModel stored in memory. The XSD and ontology mapping manager handles the tasks of concept mapping definition and the generation of mapping ontology instances. A mapping ontology instance is unmarshalled into a group of inter-connected Java classes, and each class is used as a temporary data container during structure reconciliation.

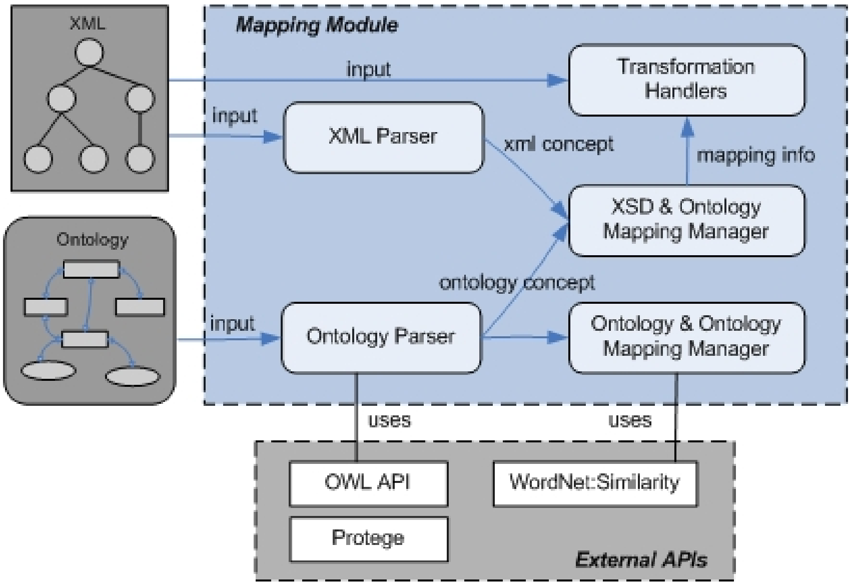


Figure 7.2: Mapping module

[1] See http://owlapi.sourceforge.net/

The ontology mapping manager is implemented using both Java and the WordNet Perl package. It provides many-to-many concept mapping recommendations. Finally, the transformation handler is implemented to perform instance level data transformation. An input XML document is transformed into an ontology instance based on the methods introduced in the ontology mediation theory. The mapping module is implemented as an independent Java package to be included by other Java applications.

7.2.2 Query Module

The query module provides query processing capabilities, and some of the functions implemented in the query module heavily rely on the methods implemented in the mapping module. Three system components are implemented as shown in figure 7.3, including query interface manager, display handler, and query handler. Query interface manager provides functions that defines, generates and stores a query interface. A query interface is an XML document providing basic query structure, more information on query interface can be found in chapter 5.

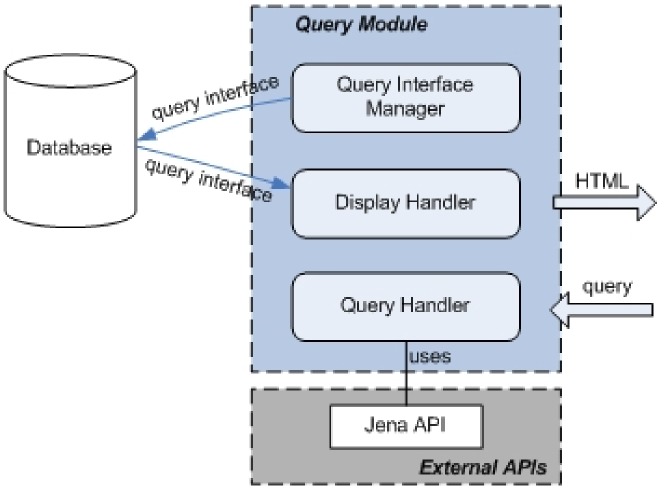


Figure 7.3: Query module

The display handler is implemented using the Java XSLT transformation engine, and the transformation is performed using the XSLT document defined in chapter 5. Finally, the query handler implements functions such as query passing, query translation and query execution. The parsing of a query is implemented by comparing the concept sets defined in the schema mapping against concepts defined from the ontology. The query translation is performed by replacing concepts defined in the query message with their corresponding concepts defined from the schema mapping. Query execution is implemented using the data transformation method provided by the mapping module, and the query message is processed using the Jena[1] query engine, an Java implementation of the SPARQL query message. In our first implementation, the query module is integrated into the web application. However, it can be separated into its own package to allow further use by other applications.

7.2.3 Interface Module

The interface module implements tools, user interfaces and top-level business logic that enables the interactions between the end-users and the accommodation hub system. Based on the publish-combine-use mechanism, the interface module is split into three sub-applications, each is developed for a specific user group. For information providers, a provider web portal is developed to allow the publishing of information sources. Features provided by the provider portal include source submission, mapping definition and source management. Source submission allows the publishing of information sources owned by the information provider. A core part of the publishing process is the definition of schema level mappings, and this is done via a graphical mapping tool.

Figure 7.5 shows the developed mapping tool. The concepts defined from

[1] See http://jena.sourceforge.net/

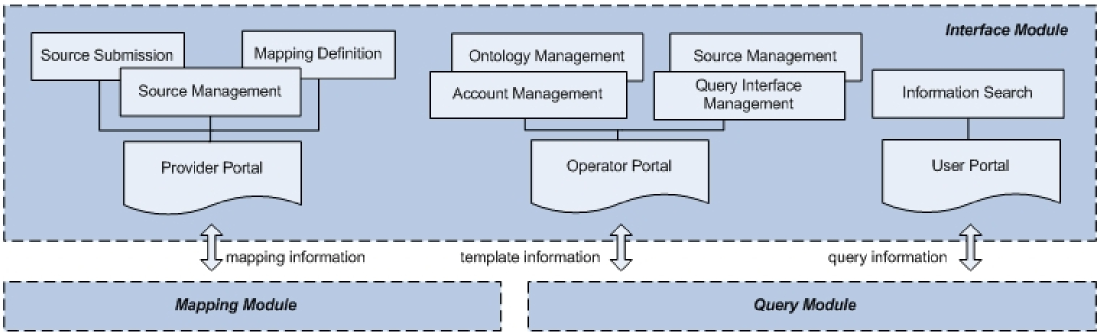
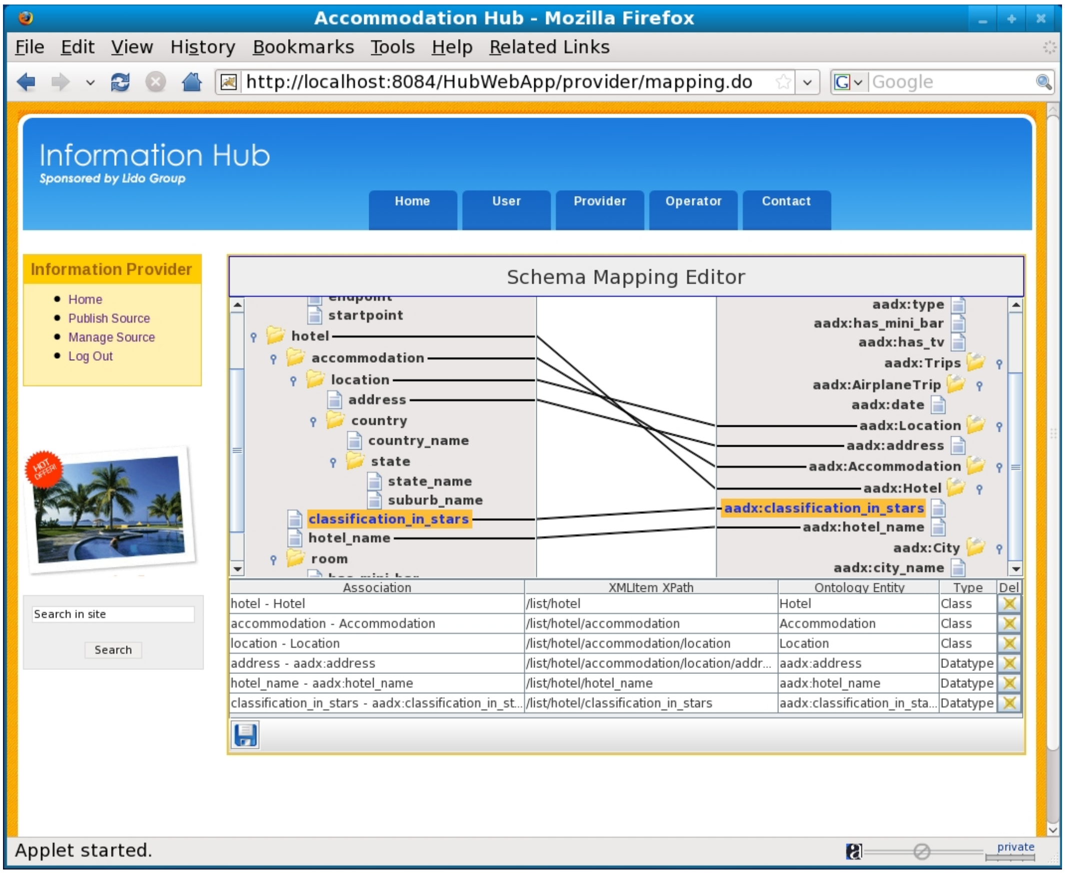


Figure 7.4: Interface module

the XML schema is listed on the left hand panel, and the ontology concepts are displayed on the right. Both schemas are presented as concept hierarchies. To define a concept mapping between a pair of similar concepts, users need to drag the concept from the XML schema tree and drop it to the corresponding node listed in the ontology tree. A black line is drawn between the pair of concepts to indicate the semantic mapping defined between them. For example, the concept "hotel" defined from the XML schema is mapped to the ontology concept "aadx:Hotel". The defined concept mappings are also displayed in the mapping table listed at the bottom of the tool. The mapping tool is implemented using methods provided by the mapping module.

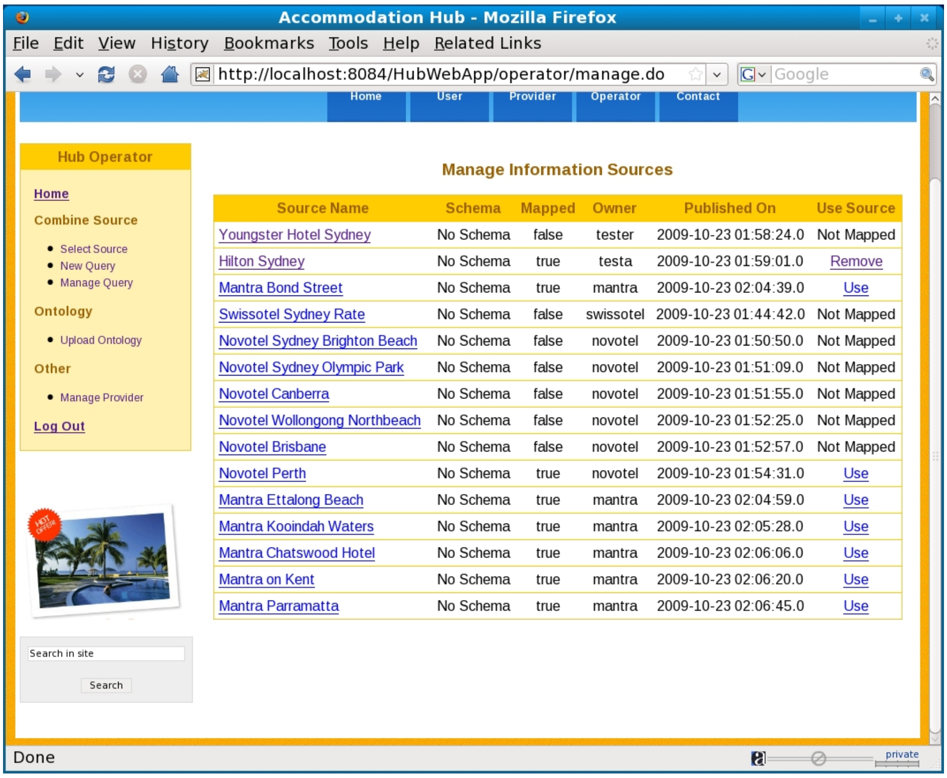
In the operator portal, an information operator manages the published information sources and defines a set of query interfaces to provide unified information access for information users. Each query interface is defined via the graphical tool called the Query Interface Editor (QIE). During query interface definition, QIE parses the global ontology into a concept hierarchy, and displays the hierarchy to the information operator. The information operator specifies the name and the functionality of the query interface, and he/she also decides on the set of parameters used for the query by selecting corresponding ontology concepts from



Association	XMLItem XPath	Ontology Entity	Type	Del
hotel - Hotel	/list/hotel	Hotel	Class	☒
accommodation - Accommodation	/list/hotel/accommodation	Accommodation	Class	☒
location - Location	/list/hotel/accommodation/location	Location	Class	☒
address - aadx:address	/list/hotel/accommodation/location/addr...	aadx:address	Datatype	☒
hotel_name - aadx:hotel_name	/list/hotel/hotel_name	aadx:hotel_name	Datatype	☒
classification_in_stars - aadx:classification_in_st...	/list/hotel/classification_in_stars	aadx:classification_in_sta...	Datatype	☒

Figure 7.5: Schema mapping editor

the concept hierarchy. For example, to define a query interface that provides address look up for hotels. The query interface should have an input parameter “hotel_name” and an output parameter “address”. Both input and output parameters are selected from the displayed concept hierarchy, as shown in figure 7.5. Other functionalities that support the daily operation of the accommodation hub are also implemented in the operator portal, which include global ontology management and account management.



Manage Information Sources

Source Name	Schema	Mapped	Owner	Published On	Use Source
Youngster Hotel Sydney	No Schema	false	tester	2009-10-23 01:58:24.0	Not Mapped
Hilton Sydney	No Schema	true	testa	2009-10-23 01:59:01.0	Remove
Mantra Bond Street	No Schema	true	mantra	2009-10-23 02:04:39.0	Use
Swissotel Sydney Rate	No Schema	false	swissotel	2009-10-23 01:44:42.0	Not Mapped
Novotel Sydney Brighton Beach	No Schema	false	novotel	2009-10-23 01:50:50.0	Not Mapped
Novotel Sydney Olympic Park	No Schema	false	novotel	2009-10-23 01:51:09.0	Not Mapped
Novotel Canberra	No Schema	false	novotel	2009-10-23 01:51:55.0	Not Mapped
Novotel Wollongong Northbeach	No Schema	false	novotel	2009-10-23 01:52:25.0	Not Mapped
Novotel Brisbane	No Schema	false	novotel	2009-10-23 01:52:57.0	Not Mapped
Novotel Perth	No Schema	true	novotel	2009-10-23 01:54:31.0	Use
Mantra Ettalong Beach	No Schema	true	mantra	2009-10-23 02:04:59.0	Use
Mantra Kooindah Waters	No Schema	true	mantra	2009-10-23 02:05:28.0	Use
Mantra Chatswood Hotel	No Schema	true	mantra	2009-10-23 02:06:06.0	Use
Mantra on Kent	No Schema	true	mantra	2009-10-23 02:06:20.0	Use
Mantra Parramatta	No Schema	true	mantra	2009-10-23 02:06:45.0	Use

Figure 7.6: Query interfaces in user portal

Finally, the user portal displays the defined query interfaces as web forms, and users can conduct information search via the displayed web form. Figure 7.6 shows the hotel address search query interface displayed in the user portal. Query message issued via the web form is passed onto the query module for further processing, and query processing in the prototype system is implemented following the hybrid integration approach.

The focus of this chapter is to demonstrate the applicability of the proposed integration framework rather than giving a comprehensive description on the developed prototype system. Therefore, in this chapter, we do not cover all the implementation details of the developed prototype system. However, a comprehensive description on the accommodation hub system including its functionalities and interfaces are provided in the appendix section of this thesis.

7.3 Case Study

To help understand the developed prototype system and its practical use, in this section we present two real world cases from the online accommodation domain. The first case covers scenarios in aggregating a set of similar information sources, each information source is structured in its own format. The second case shows the integration of a set of complementary information sources, where each information source gives a partial view on the integrated information. The presented cases are used to demonstrate the application of the developed prototype system.

7.3.1 Hotel Rate

The Lido Group[1] is a leading service provider in online accommodation brokering, and it specializes in delivering accommodation related services to large corporate consumers. Part of Lido's daily operation is to collect up-to-date hotel rate information from hundreds of different hotel suppliers, and then provide those information to its customers in a generic and aggregated format. To fulfill this requirement it is not feasible to store all hotel rate information into one system,

[1] See http://www.lido.com.au/

due to storage costs, maintenance effort and data coherency. In addition, to allow the immediate reflection of changes occurring at individual information sources, the hotel rate information should not be materialized into a central repository but rather be collected from individual sources at system run time. For those reasons, the developed accommodation hub system is used as an information integration platform, to provide integrated access to heterogeneous hotel rate sources. Each hotel supplier is an information provider that provides hotel rate information. The Lido Group is the operator of the accommodation hub, performing integration tasks such as the definition of query interfaces, maintenance of the global ontology. The corporate customers are the information users.

In the accommodation hub, an hotel supplier can join the hub by registering a provider account. After gaining access to the provider portal, hotel suppliers can publish their hotel rate sources to the hub via schema mapping. The defined schema mapping establishes a virtual connection between the published information source and the accommodation hub. To allow the query of the connecting information sources, the hub operator defines a query interface, in this case, the hotel rate search query interface. The defined query interface is displayed to the user as a web form. Assume that a customer wants to find a hotel in "Sydney" with a price limit of "less than $100". A query is issued using the defined criteria, and the issued query is executed by the hub to produce the final result. During query execution, information collected from individual sources are transformed, combined and processed via ontology mediation.

In addition, a hotel supplier can leave the accommodation hub at any time by removing its information source from the hub system. The removed information source becomes invisible to the information users, but does not affect the operation of other information sources. Nevertheless, the hub operator can organize the query interface in various ways to improve the visuality of its result. For example, to allow the easy comparison of hotel location, the location can be defined as the first output field for the hotel rate search query interface. The accommodation hub system provides a flexible solution for hotel rate aggregation over the large number of heterogeneous information sources. It also provides more freedom to the hotel suppliers, allowing them to join or leave the system depending on their own interests. Hotel suppliers now have more control over their own inventory sources. In addition, without materializing the information collected from individual sources, results produced from the hub system are more up-to-date with high data coherency.

7.3.2 Hotel Review

Trip advisors[1] and hotel review sites[2] are web applications that allow consumers to share personal experiences online. The basic principle in those applications is the joining of personal travel experiences with their related travel information. For example, the information “Novotel darling harbour resort is the best accommodation I ever experienced!” should be associated with the hotel “Novotel Darling Harbour Resort”. The gathering of hotel review information can be treated as an information integration problem that involves a large number of information providers. In the online hotel review environment, review information provided by online travelers need to be combined with the information provided by hoteliers, and each information source represents a scattered information piece that when combined together can serve a particular information need. This scenario is different from the previous hotel rate example where individual information sources are aggregated together rather than being combined.

Through the accommodation hub system, both travelers and hotel service

[1] See http://www.tripadvisor.com/

[2] See http://www.virtualreviews.com.au/

providers can become part of the information provider team. Each information provider publishes its information source into the hub, and defines the schema mapping between them. For example, a tourist group can publish their travel experiences on hotels in Australia, whereas a hotel provider publishes its hotel information to the hub. The hub operator creates a query interface that provides hotel price information together with travel experience for a given suburb. Using the defined query interface, a tourist can specify the information he wants to request. Assume that the tourist wants to select the hotel information in "Darling Harbour". The hub extracts the information from heterogeneous information sources. First, the hotel price information together with hotel name is extracted from the sources provided by hotel suppliers. On the other hand, the travel experience information together with hotel names are extracted from sources provided by the tourist group. Both sets of information are joined together using the common concept, in this case, the hotel name. The result is displayed to the tourist as one single unified information source.

In the hotel review case, information providers could be individual travelers, online community groups, hotel providers or even hotel intermediaries, thus those information sources are quite dynamic, heterogeneous and changing in unpredictable ways. The flexibility offered by the accommodation hub system can be well leveraged to cope with the integration of hotel review information. Individual information providers can join or leave the hub system according to their own interests, and the published information source can be organized into different forms to serve the needs of information users.

7.4 Conclusion

In this chapter, we presented the prototype system created using the theories and methods proposed in preceding chapters. The developed system named “accommodation hub” can be used for enabling information integration in the online accommodation domain. To demonstrate its applicability, we presented two real world examples. In the first example, the accommodation hub is used to aggregate hotel rate information collected from heterogeneous information sources. In the second case, the system is used for integrating personal travel experience information with hotel related information such as hotel price. Each case represents an unique information integration scenario. In the first case, information sources are aggregated in a parallel fashion, whereas in the second case, information sources are combined in a complementary fashion. The evaluation of the developed prototype system is covered in the next chapter.

Chapter 8

Evaluation

In this chapter, we evaluate the proposed semantic integration framework. The evaluation is conducted using the prototype system that is developed in the preceding chapter and the appendix. We organize the evaluation into two parts: first, we evaluate the techniques and methods used in the semantic integration framework, which include the many-to-many ontology mapping method, the XML to ontology mapping and transformation method, and the bottom-up query resolving method; second, we focus on the evaluation of the semantic integration framework as a whole, and examine the behavior as well as the performance of the developed prototype system for information integration. This chapter is presented as follows. Section 8.1 shows the evaluation settings, including the evaluation criteria, data sets and test cases. Section 8.2 shows the evaluation results and research findings, followed by discussion and conclusion in Section 8.3.

8.1 Evaluation Design

Throughout this thesis we have developed a number of methods to facilitate information integration, which include the many-to-many ontology mapping method, the XML to ontology mapping and transformation method, and the bottom-up query resolving method. In the first part of the evaluation, we set up sets of experiments to evaluate each developed information integration method. This part of the evaluation serves two purposes: first, it reveals the behaviour of the developed methods in information integration. Second, it shows that the developed methods deliver better results with respect to some of the evaluation criteria than some related information integration approaches. Each method is measured against a pre-selected set of evaluation criteria.

The second part of the evaluation is designed to measure the developed system as a whole, and it is used to study the behavior of the information integration solution. Evaluating the developed solution as a whole is comparatively difficult for several reasons: first, to the best of our knowledge, the accommodation hub is the first application that leverages the benefit of vast user involvement for semantic mapping definition, and it is difficult to simulate the information integration process with a sufficiently large population of users. Second, there are few agreed upon standards with respect to the evaluation of information integration solutions. Existing evaluation proposals such as the Thomas et. al.'s work (Gannon *et al.*, 2009) only provide partial guidance for measuring certain aspects of information integration, some information integration properties are ignored. Although it is an interesting research topic to define a comprehensive standard for measuring information integration solutions, however, this task is out of the scope of our research. Therefore, in this evaluation, we do not compare our solution with existing integration solutions or benchmarks. Instead, we try to study the behavior and performance of the developed solution in information integration. Furthermore, we do not make any scientific claims on the evaluation results, but to provide audiences with an intuitive perception on the system performance and usage.

The evaluation settings are presented in three dimensions: evaluation measurement, data set, and test case. An evaluation measurement is a method that helps to determine the satisfaction rate of a particular requirement. For instance, the measurements of precision and recall are normally used to determine the level of compliance with accuracy (Noy & Musen, 2002). Results generated from the evaluation measurements are used to study the behavior of the developed system and methods. A data set provides a collection of data samples where each sample represents a particular data instance selected from the information integration domain. A test case often represents a particular information integration scenario, and the defined test cases are used to evaluate different information integration scenarios.

8.1.1 Evaluation Measurements

Previous research works have proposed some methods for measuring information integration methods. A typical example would be the measurement of ontology mapping algorithms, and it is often performed by calculating the values of precision and recall (Noy & Musen, 2002). The measurement of precision and recall is widely used to determine the accuracy of an ontology mapping method. In this section, we leverage some of the existing measurements to evaluate our information integration solution. In addition, we also introduce a few new measurement techniques to assist the evaluation process. The following subsections introduce some of the evaluation measurements.

Precision and Recall

Like most conventional ontology mapping evaluations, our evaluation also uses recall and precision to determine the level of accuracy of the developed ontology mapping method. However, unlike conventional ontology mapping methods, our method focuses on the mapping of a group of homogeneous ontologies. Hence, the accuracy calculation varies slightly from conventional measurements. In conventional cases, precision measures the correctness of the information integration process, whether an issued query can lead to the correct combination of individual information sources. It is calculated using the total number of correct mappings ($M_{correct}$) divided by the total number of identified mappings ($M_{identified}$). In a many-to-many mapping relationship, the value produced from conventional precision formula does not fully reflect the accuracy of the produced concept mappings. For example, in the mapping scenario illustrated in figure 8.1, there is a total number of 8 identified concept mappings among the ontologies O_1, O_2 and O_3, and a total number of 4 correct mappings. Using conventional calculation, the precision of the mapping method is 0.5. However, the concept mappings generated between O_1, O_3, and between O_2, O_3 are extremely poor, and the precision value of 0.5 does not fully reflect the true accuracy of the mapping method.

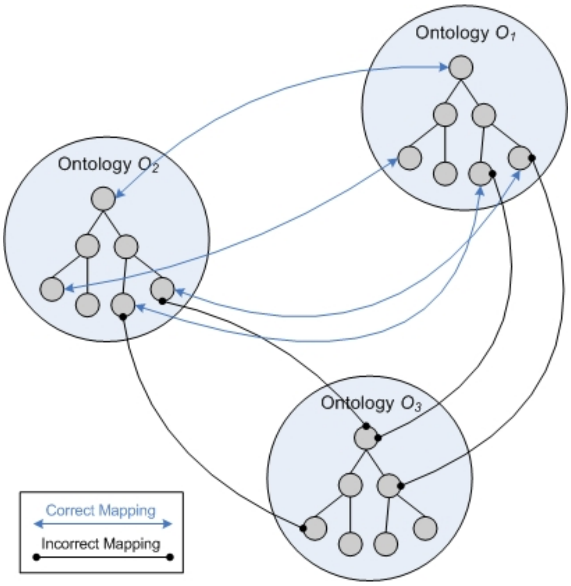


Figure 8.1: Incorrect precision calculation

To improve the accuracy of the precision calculation in measuring many-to-many ontology mapping solutions, we use an average precision calculation method to measure the precision of our ontology mapping algorithm. In our case, precision describes the average proportion between the total number of correct concept mappings, and the total number of identified concept mappings. The average precision ($P_{average}$) is calculated as the sum of all precision values ($\sum_{k=1}^{n} P_k$) divided by the total number of ontology mappings ($N_{total\_mappings}$), as formulated in the equation below. Each precision (P_k) is calculated using conventional precision calculation method. We say that a high precision corresponds to a high accuracy.

$$P_{average} = \frac{\sum\limits_{k=1}^{n} P_k}{N_{total\_mappings}} \tag{8.1}$$

Likewise, the average recall value ($R_{average}$) is also calculated as the sum of all recall values divided by the total number of ontology mappings. Each recall value (R_k) is also obtained from conventional calculation. The calculation of recall is documented in the following equation. A high recall also implies a high accuracy of the framework. It measures the completeness of the identified information sources, whether all the correct information sources are found.

$$R_{average} = \frac{\sum\limits_{k=1}^{n} R_k}{N_{total\_mappings}} \tag{8.2}$$

In our case, we treat precision as same weighted as recall, and we do not conduct the f-measure which combines precision with recall.

Total Comparison Cycle

To measure the performance of the developed ontology mapping method, we introduce the concept of comparison cycle. A comparison cycle is a calculation cycle used to determine the similarity between a pair of ontology concepts. Likewise, total comparison cycles (C_{total}) represents the total number of comparison cycle executed to compare all concepts defined from the input ontologies. We say that the less comparison cycles the mapping process requires the more efficient it is. In our experiment, we use this measurement to compare the performances of the related ontology mapping methods.

Mapping Generation Time

Mapping generation time ($T_{mapping}$) represents the total amount of time consumed for generating a particular mapping instance. It is used to measure the performance of the XML and ontology mapping method by calculating the total amount of time consumed for generating a particular ontology mapping instance. However, we do not consider the amount of time involved for defining the mapping, in other words, the amount of human efforts involved in creating the mapping is not included.

Data Transformation Time

Data transformation time ($T_{transform}$) is also used to measure the efficiency of the XML to ontology mapping method. It represents the total amount of time consumed for transforming a XML document into an ontology instance.

System Response Time

We use system response time ($T_{response}$) to measure the performance of the query resolving method. System response time refers to the time used for identifying, transforming, and integrating individual information sources, and such process is implemented by the prototype system as introduced in the previous chapter. Since the prototype system is implemented as a web application, we use the total time used to obtain a resulting query page from the prototype system to represent the system response time.

In addition, the measurement of system response time is also used to assist the study of the prototype system. First, it shows the trade-off between the system performance and the increase of information sources. The increment of information source is simulated by increasing the total number of information sources, as well as information file size. Second, it reveals the nature of the system in handling diversified information sources. This is achieved by calculating the average amount of time consumed on 1KB of information source, and we name such figure as the average time consumption. The average time consumption (t) is calculated as the total amount of system response time ($T_{response}$) divided by the total size (in terms of KB) of the integrated information sources (S_{total}). The following equation shows the calculation of average time consumption. The increase of t indicates the decrease in system performance.

$$t = \frac{T_{response}}{S_{total}} \tag{8.3}$$

Conversion Number

All above mentioned measurements mainly focus on the measurement of system performance, other characteristics of information integration such as flexibility and scalability are not evaluated. Thomas et. al. (Gannon *et al.*, 2009) propose to use conversion number for analysing the adaptability, extensibility and scalability of semantic information integration solutions. In their approach, they define those information integration properties including adaptability, extensibility and scalability as the number of conversions required for handling semantic changes occurred during information integration. W. Chiu (Chiu, 2001) describes system flexibility as the ability to adapt readily to intense changes of usage or demand to meet predefined objectives. In our case, flexibility is defined as the ability of coping with changes occurred at the integrated information sources, more specifically, the joining and leaving of information sources. Kotsis and Taferner defines scalability as the ability to handle an increase in the load of a system. In our case, scalability can be referred to as the ability of handling an increase number of information sources in the information integration process. Base on their research, we also use the measurement of conversion number to evaluate the flexibility and scalability of the developed integration solution. In our evaluation, the conversion number represents the total number of semantic mapping definitions required for handling changes occurred during information integration, such as the joining or leaving of information sources. The less conversions required for handling a particular change the more flexible and scalable the solution is.

8.1.2 Data Sets

One ontology data set is created for the evaluation of the developed ontology mapping method. We select all the tourism ontologies with a total number of around four thousand classes from the online daml ontology library[1], and then tailor each ontology into simple and smaller ontologies with smaller number of classes and properties. The set of ontology concepts used to form the testing data can be seen as a representative of the online accommodation domain, and each small ontology is tailored for a particular business need so that the designed tests can better match with real world information sharing scenarios. For example, the ontology defined in Appendix A is used for accomodation inventory information sharing, whereas other ontologies are defined to serve different business needs. Using the selected set of ontology concepts, we form several new ontologies for testing purposes, and increase their semantic complexity by adding more classes and properties. This is mainly due to the fact that in real world cases, both simple

[1] See http://www.daml.org/ontologies/

and complex ontology models are used for information sharing and integration, thus we need to study the behavior of the developed solution in coping with increased semantic complexity.

Two XML data sets are created for simulating the integration process. The first XML data set is created using XML schema gathered online. We select a total number of 50 XML schema from hotel websites, online booking portals, and travel organizations, and we collect the schemas with high caution to cover real world scenarios on online accommodation information sharing. The collected schema are restructured to form 70 sample XML schema that provide hotel rate information. For each restructured sample schema, we create one XML document that conforms to the schema, and the XML document is filled with sample hotel rate information. A total number of 70 XML documents are created with an average file size of 300Kb. The created XML documents are used as independent information sources for the evaluation. The first XML data sets are generated to simulate real world hotel inventory information sharing scenarios.

The second XML data set contains hotel review and promotion information. Again we use the data types defined from the selected XML schema to create a total number of 80 sample schema. Half of the created sample schema are used as hotel review schema, and the other 40 samples are used for carrying hotel promotion information. For each created sample schema, we generate an XML document corresponds to the schema, and the XML document is filled with sample hotel information. Each XML document represents a real world online information source, and the created XML documents are stored onto the same web server as the prototype system for testing purpose. We assume that the formed sample information sources are sufficient for representing real world information source examples.

A total number of twenty testing queries are created, and the defined queries are divided into two sets. The first query set contains simple queries for selecting hotel rate information. The second query set contains much complex queries that select hotel related information such as hotel review, hotel promotion. The defined queries start with simple hotel rate selection, and expand to large, complex information selection. We gradually increase the query complexity to test its effect on information integration. The predefined sample queries are used as inputs throughout the evaluation process. Appendix B shows some sample documents used in the evaluation.

8.1.3 Test Cases

We test the ontology mapping method in two sets of test cases. In the first set of test cases, we test our solution against an increasing number of ontologies. This experiment aims to identify the effect of ontology increase on the proposed mapping solution. We used twenty tourism ontologies with a total number of one thousand classes selected from the ontology data set. We start with a pair of ontologies and increase the number of ontology by two in each test case. In the second set of test cases, we test our solution against the increasing number of involved ontology concepts. The goal is to reveal the performance of the solution in handling increased ontology complexity. We use a total number of 12 ontologies from previous test cases, and this time we start with 12 partial ontologies with total number of 120 classes, and in each test iteration we increase the number of ontology concepts involved in each ontology by ten until a total number of one thousand and two hundred classes are compared in the test case. In both test cases, we measure the value of precision and recall, and the total number of comparison cycle at each test iteration.

To evaluate the XML to ontology mapping and transformation method, we design two sets of mapping and transformation test cases. The first set evaluates the performance of the developed method in generating concept mappings. In each test iteration, we create a set of concept mappings between an input XML schema and an ontology. The input schemas are created using concepts and elements selected from the ontology and XML data sets. We gradually increase the complexity of the input schemas by adding elements and concepts, and measures the corresponding value of $T_{mapping}$ for generating a mapping instance. The second test case set uses the mapping instances generated from the first set, and performs instance level data transformation over a group of input XML documents. We increase the size of the input XML document in each test iteration, and measures the corresponding $T_{transform}$ value consumed for each data transformation. The goal is to study the trend of the developed solution in handling increasing amount of data transformation.

Two sets of test cases are created to simulate the processes of information integration. In the first set of test cases, we test the prototype system with an increasing number of information sources. We start with 10 sample information sources, and each information source with an average file size of 100Kb. We feed the system with the predefined set of queries, then collect the data required for calculating the evaluation criteria. We increase the total number of information sources by 5 in each test iteration, and repeat the process until all information sources from the hotel rate data set are added to the integration process. We repeat the same scenario on a fixed number of information sources, and increase the size of each information source by average of 50Kb in each test iteration. We measure the system response time in each test case using Apache JMeter[1]. Apache JMeter is a testing application specifically designed for testing web applications, and it provides functions that measure system performance.

In the second set of test cases, we measure the behavior of the system in

[1] See http://jakarta.apache.org/jmeter/

handling a diversified number of information sources. We start with 50 sample information sources, each with an average file size of 100Kb, and instead of changing the nature of the integrated information source, we change the complexity of the issued query message. In each test case, the issued query message is resolved by an increasing number of information sources. The average time consumption t is recorded to assist our study.

Finally, to evaluate the flexibility and scalability of the developed prototype system, we use a seperate test case set. At the first stage of the designed test case, we randomly selected 50 information sources from both the hotel rate and hotel review data sets, and connect the selected sources to the prototype system. We calculate the total number of semantic mapping required to integrate all involving information sources. We then randomly add and remove information sources to and from the prototype system, and record the corresponding change of conversion number. Data gathered from the experiment is compared with conventional integration approaches.

All test cases are implemented using Java, and some are implemented as a J2EE web application. The first test case set is implemented using both Java and Perl, and part of the ontology mapping algorithm is implemented using Perl. We choose to use Apache Tomcat[1] as the web server for simulating the information integration process. All test cases are executed on a computer with 2GB RAM and 2.16GHz CPU. Performance data is collected using Apache JMeter and Java programs.

[1] See http://tomcat.apache.org/

8.2 Evaluation Results

Ontology Mapping

Here, we present the test results generated from the above mentioned experiments. The first two test sets focus on the evaluation of the developed ontology mapping method. Table 8.1 and 8.2 shows the results obtained from the first two test sets. During each test, we record the total number of ontologies, concepts, and the corresponding concept comparison cycle used in each test iteration. We also calculate the average precision ($P_{average}$) and recall ($R_{average}$) of the produced mapping results.

Total Ontologies	**Total Concepts**	$P_{average}$	$R_{average}$	C_{total}
2	300	0.67	0.85	1053
3	300	0.63	0.76	926
4	300	0.58	0.77	903
5	300	0.65	0.71	867
6	300	0.53	0.75	831
8	300	0.69	0.65	798
10	300	0.62	0.73	760
12	300	0.71	0.67	715

Table 8.1: Partial test results for ontology mapping evaluation

As shown in Table 8.1, the total number of concept comparison cycle (C_{total}) decreases as the total number of ontology increases. With a fixed number of ontology concepts, the semantic complexity of each input ontology also decreases in each test iteration. This is another factor that affects the overall performance of the developed many-to-many ontology mapping method. The mapping method shows better performance when dealing with a larger number of input ontologies. In table 8.2, the total number of comparison cycle increases as the total number of compared concept increases. We present the generated results in two illustrative diagrams, as shown in Figure 8.2.

Total Ontologies	Total Concepts	$P_{average}$	$R_{average}$	C_{total}
12	120	0.67	0.83	213
12	240	0.73	0.77	528
12	360	0.58	0.79	835
12	480	0.65	0.71	1223
12	600	0.53	0.75	1580
12	840	0.69	0.85	2468
12	960	0.62	0.83	2971
12	1080	0.71	0.68	3376
12	1200	0.71	0.68	3802

Table 8.2: Partial test results for ontology mapping evaluation

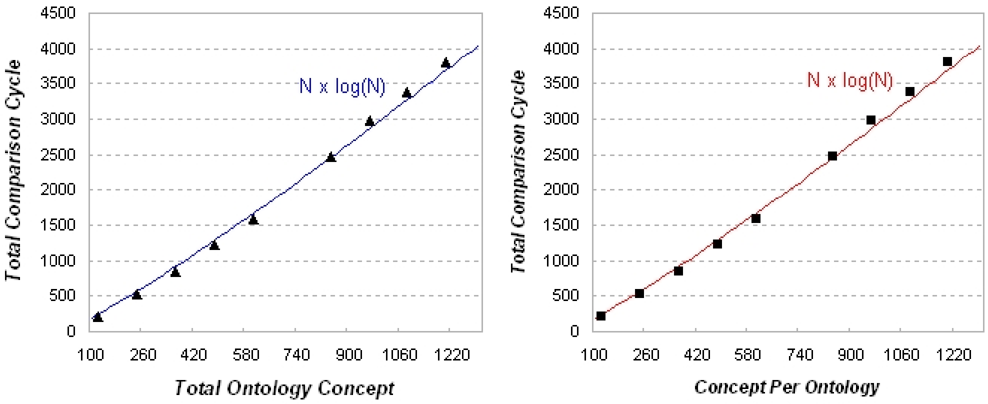


Figure 8.2: Total concept comparison cycle

Figure 8.2 compares the increase of concept comparison cycle (C_{total}) with the increase of concept quantity as well as the increase of ontology complexity. We use the ratio of average concept per ontology to represent the semantic complexity of the input ontologies, and it is calculated as the total number of concept divided by the total number of ontology. Results obtained from the second experiment show that the proportion between C_{total} and the total number of concept is similar to the function of $n \times \log n$, where n is the total number of ontology concept.

Mapping and Transformation

Table 8.3 shows the mapping results obtained from the third test. The complexity of the mapping scenario is gradually increased by adding more concepts and elements to the input ontology and XML schema. As the table shows, the total mapping generation time increases as the total number of mapping associations defined between the ontology and the XML schema increases. However, the increase of mapping granularity, such as the increase of datatype associations as well as the increase of object associations does not directly affect the overall mapping generation performance. In addition, the mapping associations defined between different ontology elements show similar weights to the mapping generation process.

ID	**Class**	**OP**	**DP**	**Total Associations**	$T_{mapping}$
$mp1$	6	4	11	21	7ms
$mp2$	12	8	27	47	33ms
$mp3$	20	11	47	78	93ms
$mp4$	27	19	54	100	87ms
$mp5$	33	21	37	91	102ms
$mp6$	41	32	80	153	325ms
$mp7$	54	39	62	155	298ms

Table 8.3: Partial results for mapping generation

The mapping instances from the previous test are used to evaluate the transformation method, and the test is conducted over an increasing amount of XML data. Figure 8.3 plots the the total transformation time $T_{transform}$ against the input XML file size. The data transformation time increases as the size of the input XML document increases. In order to study the behaviour of the transformation method in handling complex transformation scenarios, we compare the average time consumed over four different transformation scenarios, and each transformation is based a mapping instance selected from the previous test. We choose to use four mapping instances, including $mp1$, $mp2$, $mp4$, and $mp7$, each with different level of mapping complexity as shown in table 8.3. Results obtained from the test show that data transformation under complex mapping associations require more processing time than the transformation based on simple mappings, especially when the input XML file size is large.

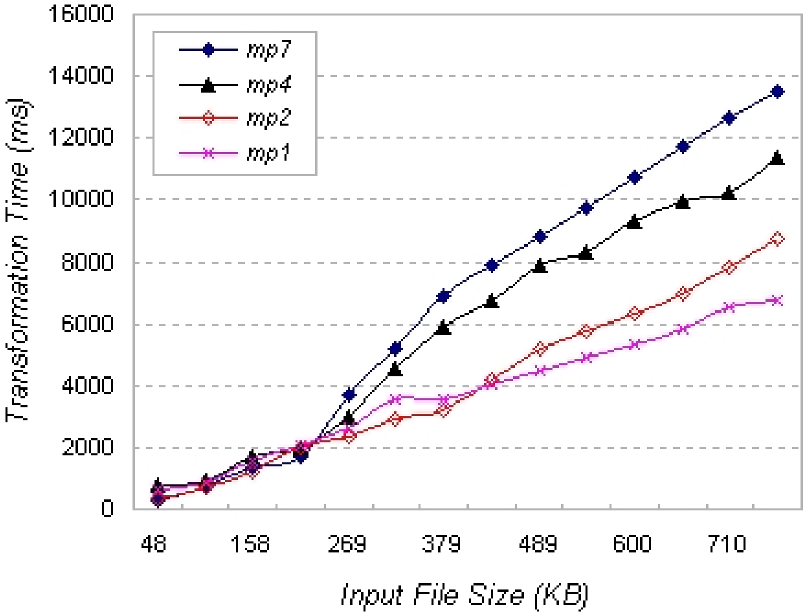
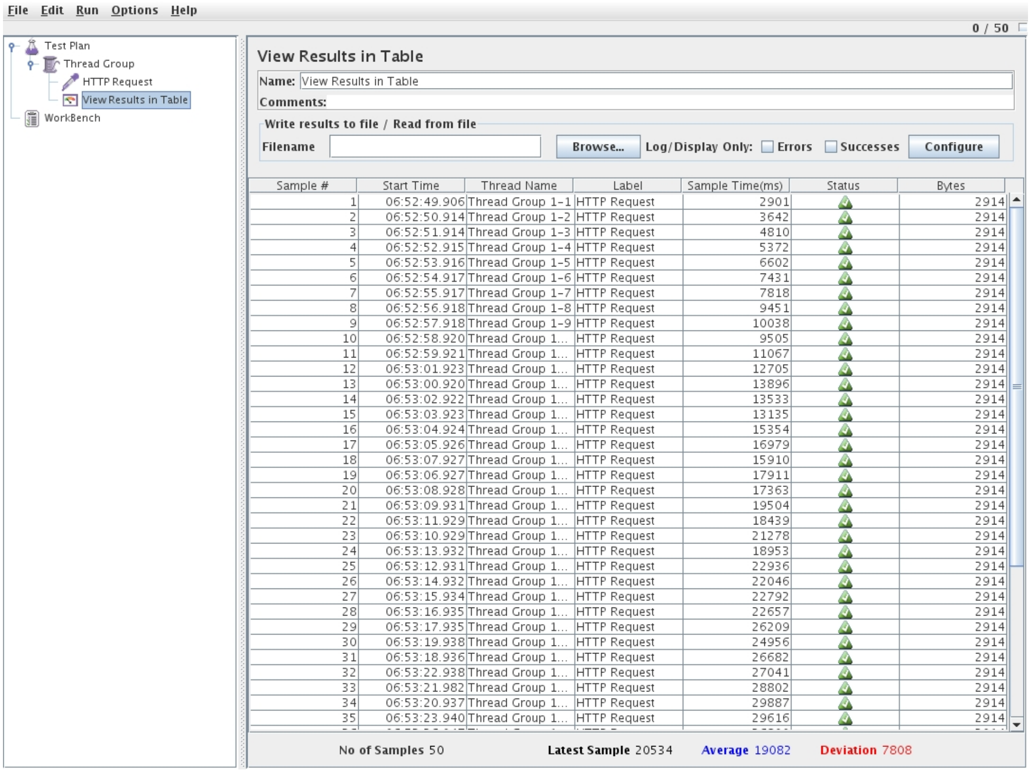


Figure 8.3: XML to Ontology Transformation Time ($T_{transform}$)

Information Integration

Three test cases are executed to simulate the process of information integration, and the corresponding system performance data is collected using Apache JMeter. Figure 8.4 shows the test results generated from a hotel search query test. The test was conducted on a total number of 50 information sources with an average file size of 50Kb (Kilobyte), where each XML file represents a real world information source. 50 test threads are used to simulate the test, each thread is responsible for the execution of a HTTP query request. We define a 1 second delay between each issued thread, and the goal is to provide sufficient amount of execution time for each issued query request. All the system response time is recorded and displayed in table formats. As illustrated in the figure, the average response time for the executed query requests is 19082 ms (milliseconds) with a standard deviation of 7808 ms. We use the average response time to plot the final test result.



View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename

Log/Display Only: □ Errors □ Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes
1	06:52:49.906	Thread Group 1-1	HTTP Request	2901	✓	2914
2	06:52:50.914	Thread Group 1-2	HTTP Request	3642	✓	2914
3	06:52:51.914	Thread Group 1-3	HTTP Request	4810	✓	2914
4	06:52:52.915	Thread Group 1-4	HTTP Request	5372	✓	2914
5	06:52:53.916	Thread Group 1-5	HTTP Request	6602	✓	2914
6	06:52:54.917	Thread Group 1-6	HTTP Request	7431	✓	2914
7	06:52:55.917	Thread Group 1-7	HTTP Request	7818	✓	2914
8	06:52:56.918	Thread Group 1-8	HTTP Request	9451	✓	2914
9	06:52:57.918	Thread Group 1-9	HTTP Request	10038	✓	2914
10	06:52:58.920	Thread Group 1...	HTTP Request	9505	✓	2914
11	06:52:59.921	Thread Group 1...	HTTP Request	11067	✓	2914
12	06:53:01.923	Thread Group 1...	HTTP Request	12705	✓	2914
13	06:53:00.920	Thread Group 1...	HTTP Request	13896	✓	2914
14	06:53:02.922	Thread Group 1...	HTTP Request	13533	✓	2914
15	06:53:03.923	Thread Group 1...	HTTP Request	13135	✓	2914
16	06:53:04.924	Thread Group 1...	HTTP Request	15354	✓	2914
17	06:53:05.926	Thread Group 1...	HTTP Request	16979	✓	2914
18	06:53:07.927	Thread Group 1...	HTTP Request	15910	✓	2914
19	06:53:06.927	Thread Group 1...	HTTP Request	17911	✓	2914
20	06:53:08.928	Thread Group 1...	HTTP Request	17363	✓	2914
21	06:53:09.931	Thread Group 1...	HTTP Request	19504	✓	2914
22	06:53:11.929	Thread Group 1...	HTTP Request	18439	✓	2914
23	06:53:10.929	Thread Group 1...	HTTP Request	21278	✓	2914
24	06:53:13.932	Thread Group 1...	HTTP Request	18953	✓	2914
25	06:53:12.931	Thread Group 1...	HTTP Request	22936	✓	2914
26	06:53:14.932	Thread Group 1...	HTTP Request	22046	✓	2914
27	06:53:15.934	Thread Group 1...	HTTP Request	22792	✓	2914
28	06:53:16.935	Thread Group 1...	HTTP Request	22657	✓	2914
29	06:53:17.935	Thread Group 1...	HTTP Request	26209	✓	2914
30	06:53:19.938	Thread Group 1...	HTTP Request	24956	✓	2914
31	06:53:18.936	Thread Group 1...	HTTP Request	26682	✓	2914
32	06:53:22.938	Thread Group 1...	HTTP Request	27041	✓	2914
33	06:53:21.982	Thread Group 1...	HTTP Request	28802	✓	2914
34	06:53:20.937	Thread Group 1...	HTTP Request	29887	✓	2914
35	06:53:23.940	Thread Group 1...	HTTP Request	29616	✓	2914

No of Samples 50 Latest Sample 20534 Average 19082 Deviation 7808

Figure 8.4: Sample test results

The following results are produced from the evaluation tests. In each test case, we measured the system response time against an increasing number of information sources. The goal is to evaluate the performance of the developed system in handling an increasing amount of information sources. We use the average time consumption (t) to represent the system efficiency, and the calculation of t is discussed in section 8.1. Results generated from the test cases are presented in figure 8.5. Based on the data gathered from the performed test cases, we found that the system efficiency decreases (the value of t increases) as the number of information source increases. However, the system efficiency starts to stablize when the total number of information source increase to a certain level, and not much efficiency difference occur when the total number of information source is large. Hence, we conclude that the developed system is scalable when dealing with information intgeration over a large number of information sources.

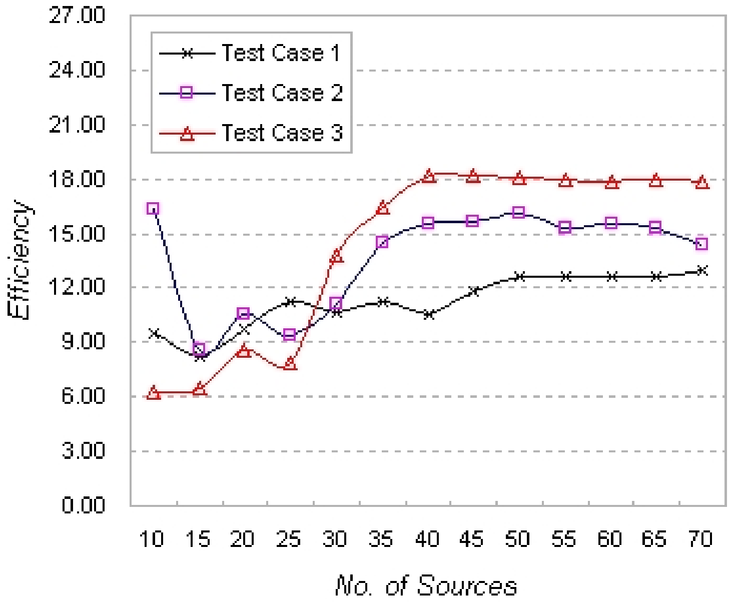


Figure 8.5: Test results on increasing number of information sources

In addition, we compare the three test cases with their input query complexities. In each test case, we feed the system with an unique set of query messages, and the complexity of the input query message increases from test case 1 to test case 3. As shown in figure 8.5, test case 1 shows better performance than other two test cases, and this is particularly true when the total number of integrated information sources is large. The system efficiency decreases when the average query complexity increases throughout the three test cases. Hence, we conclude that query complexity is a major factor that affect the performance of the developed information integration system.

Flexibility and Scalability

The last experiment tests the flexibility and scalability of the developed system. We measure the conversion number required as information sources join and leave the information sharing network. Results show that in the hybrid information integration architecture, the joining or leaving of n number of information sources only require the definition or deletion of n number of semantic mappings. The joining and leaving of individual information sources do not affect the overall performance of the integration system, and it neither affects the semantic mappings defined to other information sources.

8.3 Discussion and Conclusion

The evaluation of the many-to-many ontology mapping method demonstrates the applicability of the developed method in solving the many-to-many ontology mapping problem. From the experiment, we found that the many-to-many mapping method requires a total number of $n \times \log n$ comparisons to establish mappings among k number of ontologies, where each ontology has m number of concepts ($n = m \times k$). This is better than the number of comparisons (m^{k-1}) required by most conventional approaches when both m and k are large. The developed mapping method outperforms conventional ontology mapping approaches in terms of concept comparison cycle. Furthermore, the increase of ontology complexity does not directly affect the overall performance of the mapping method. However, as the developed method uses semantic similarity calculation combined with concept partition for concept classification, one major drawdrack of this approach is the accuracy of the establish mapping associations. The accuracy of the developed mapping method is comparatively lower than the pair-wised mapping approaches. Hence, we suggest to use the developed mapping method for establishing primitive mapping results, and the established mapping associations can be further refined by human users. Nevertheless, the accuracy of the developed mapping method can be improved by introducing extra concept measurements to the similarity calculation process.

Another finding of the evaluation study is the performance of the developed mapping generation and transformation method. The developed method shows good performance in handling complex mapping generation and transformation scenarios. Although the transformation requires more processing time for transforming data under complex mapping rules, the increase of processing time is trivial as compared to the increase of mapping complexity. The plot of both $T_{mapping}$ and $T_{transform}$ show that the developed method is scalable in handling a large amount of information sources. In addition, the evaluation of the system also proves that the bottom-up query resolving approach is effective for comparatively large-scale information integration. The efficiency of the developed system remains steady when the total number of involved information source is large, since the bottom-up approach requires the transformation of XML data into ontology format, and a large proportion of time is spent on the transformation process rather than the query process itself. Hence, the developed system is capable for handling large scale information integration.

In summary, this chapter has given an evaluation on the prototype system by conducting a set of experiments. The primary goal of the evaluation is to study the performance as well as behavior of the developed system in information integration. Results produced from the experiments show that the developed system provide good accuracy and flexibility in distributed information integration. Further evaluation is required to test the developed system against a sufficiently large population of data, and to simulate process of information integration in the large, dynamic, distributed and heterogeneous environment with the involvement of a vast amount of users.

Chapter 9

Conclusions

Nowadays, the online accommodation industry is increasingly dominated by information (Werthner & Klein, 1999). With the tremendous growth of the Web and the vast amount of information available online, the seamless integration of accommodation related information becomes a real challenge. In order to adequately support information users in collecting and sharing information online it is inevitable to create an effective information integration solution, to provide integrated access to the large number of online information sources. However, due to the nature of the online accommodation industry, a number of requirements arise, which include:

- the ability to provide correct information as according to user's request,
- the ability to provide efficient integration on a large number of heterogeneous information sources,
- the ability of coping with changes occurring at information sources,
- and the ability of integrating on an increasing number of information sources.

In this thesis, we summarize those requirements as the accuracy, efficiency, flexibility and scalability. To meet those requirements, we proposed a semantic integration framework for large-scale information integration in the online accommodation domain, and we also developed a number of techniques and methods to assist the integration process. Our solution specifically focuses on the structured and semi-structured information sources, where instance level data are stored and presented according to meta-data level schemas. In particular, we looked at XML-based data sources that are stored according to XML schemas.

The proposed framework provides solutions to the problem of information integration on three levels: data level, process level and architecture level. On the data level, we leveraged the benefit of ontology, and proposed an ontology mediated information integration approach for solving the problem of data heterogeneity. In our approach, ontology is used as a mediator for the reconciliation and harmonization of heterogeneous information sources. We also proposed a number of methods for defining schema mappings, as well as for the data transformation between XML and ontology formats. On the process level, we proposed a publish-combine-use approach for large-scale information integration. The primary goal is to reduce the cost and effort of collecting and integrating heterogeneous information sources. In the proposed approach, information providers have more control over their own information sources, as they are able to join and leave the information sharing network depend on their own interests. In addition, the proposed approach also enables the collaboration among all parties involved in the information integration process. Finally, on the architecture level, we combined the centralized integration architecture with the emerging distributed P2P architecture. The proposed architecture combine the flexibility offered by the distributed P2P approach with the query processing capability provided by the centralized approach. The proposed hybrid structure is similar to the structure of the online accommodation industry, where individual players are connected together following a distributed and hybrid fashion.

In order to demonstrate the practical applicability of the proposed semantic integration framework, we implemented a prototype system named the accommodation hub. The accommodation hub system is developed to enable information sharing and integration in the online accommodation domain. Two real world cases are discussed to demonstrate the use of the accommodation hub system. The developed prototype system is evaluated against the identified integration requirements, and results produced from the evaluation show that the developed system meets the basic requirements of information integration in the large, dynamic, distributed and heterogeneous online accommodation environment.

9.1 Future Research Directions

The semantic integration framework provides basic theories and methods for solving the challenges that exist in online accommodation information integration. To allow better application of the proposed framework into real world domains, further experiments and developments are required. One potential research direction is to test the developed prototype system in a commercial environment, and make corresponding modifications to improve its commercial viability. This requires the involvement of a large number of real world users and providers. One step towards this goal is to make the developed prototype system publicly accessible, and collect feedback from both users and providers.

On the theory side, the framework provides an information integration solution via ontology mediation. Our research mainly focused on the integration process, and we did not consider factors relating to the creation and maintenance of the ontology. In real world cases, the global ontology may evolve and change as according to real world domains. The replacement of the global ontology may lead to potential modification of the schema mappings. To avoid the recreation of schema mappings, the framework needs to take ontology change into consideration. In addition, issues such as security and trust are not considered in the research. For example, whether the provided information is correct and trustworthy. Those issues are appropriate for potential new research directions.

Appendix A

Data Samples

This section shows some of the data samples used in this research. The used data samples include XML schema, XML document, and ontology schema. Some of those samples are used for the prototype implementation, and some are used for the evaluation process. Due to the page limitation, we only include some typical examples from the data sample, and we do not provide comprehensive set of samples.

A.1 Sample Schemas

Sample Hotel XML Schema

```
<?xml version=”1.0” ?>
<xs:schema xmlns:xs=”http://www.w3.org/2001/XMLSchema”
attributeFormDefault=”qualified” elementFormDefault=”qualified”>
   <xs:element name=”list”>
      <xs:complexType>
         <xs:choice maxOccurs=”unbounded”>
            <xs:element name=”hotel”>
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name=”classification_in_stars” type=”xs:string” minOccurs=”0”
/>
```

```
        <xs:element name=”hotel_name” type=”xs:string” minOccurs=”0” />
        <xs:element name=”accommodation” minOccurs=”0” maxOccurs=”unbounded”>
        <xs:complexType>
        <xs:sequence>
        <xs:element name=”location” minOccurs=”0” maxOccurs=”unbounded”>
        <xs:complexType>
        <xs:sequence>
        <xs:element name=”address” type=”xs:string” minOccurs=”0” />
        <xs:element name=”country” minOccurs=”0” maxOccurs=”unbounded”>
        <xs:complexType>
        <xs:sequence>
        <xs:element name=”country_name” type=”xs:string” minOccurs=”0” />
        <xs:element name=”city” minOccurs=”0” maxOccurs=”unbounded”>
          <xs:complexType>
          <xs:sequence>
          <xs:element name=”city_name” type=”xs:string” minOccurs=”0” />
          </xs:sequence>
          </xs:complexType>
          </xs:element>
          </xs:sequence>
          </xs:complexType>
          </xs:element>
          </xs:sequence>
          </xs:complexType>
          </xs:element>
          </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name=”room” minOccurs=”0” maxOccurs=”unbounded”>
          <xs:complexType>
          <xs:sequence>
          <xs:element name=”has_tv” type=”xs:string” minOccurs=”0” />
          <xs:element name=”type” type=”xs:string” minOccurs=”0” />
          <xs:element name=”has_mini_bar” type=”xs:string” minOccurs=”0” />
          </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:choice>
```

```
        </xs:complexType>
    </xs:element>
</xs:schema>
```

Sample Hotel XML Schema

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xs:schema targetNamespace=”http://www.aadx.org/schema/hotel2.xsd”
            xmlns:xs=”http://www.w3.org/2001/XMLSchema”
            xmlns:ct=”http://www.aadx.org/schema/hotel2.xsd” >
<xs:element name=”hotels”>
    <xs:element name=”hotel” maxOccurs=”unbounded” >
        <xs:complexType>
<xs:element ref=”hotelname” />
        <xs:element ref=”website” />
        <xs:element ref=”phone” />
        <xs:element ref=”address” />
        <xs:element name=”room” type=”ct:roomType” minOccurs=”0” maxOccurs=”unbounded” />
        </xs:complexType>
    </xs:element>
    <xs:element name=”address” type=”xs:string” />
    <xs:element name=”hotelname” type=”xs:string” />
    <xs:element name=”phone” type=”xs:string” />
    <xs:complexType name=”equipmentType”>
        <xs:choice maxOccurs=”unbounded”>
            <xs:element name=”aircondition” type=”xs:boolean” />
            <xs:element name=”tv” type=”xs:boolean” />
            <xs:element name=”minibar” type=”xs:boolean” />
            <xs:element name=”shower” type=”xs:boolean” />
            <xs:element name=”livingroom” type=”xs:boolean” />
        </xs:choice>
    </xs:complexType>
    <xs:complexType name=”roomType”>
        <xs:sequence>
<xs:element name=”typeName” type=”xs:string”/>
<xs:element name=”roomRate” type=”xs:float”/>
<xs:element name=”guestNumber” type=”xs:integer”/>
<xs:element name=”equips” type=”ct:equipmentType”/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

Sample Hotel Review XML Schema

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xs:schema targetNamespace=”http://www.aadx.org/schema/hotelreview1.xsd”
            xmlns:xs=”http://www.w3.org/2001/XMLSchema”
            xmlns:rv=”http://www.aadx.org/schema/hotelreview1.xsd” >
    <xs:element name=”reviews” maxOccurs=”unbounded” >
        <xs:complexType>
            <xs:element name=”review”>
                <xs:complexType>
                    <xs:element ref=”hotelname” />
                    <xs:element ref=”address” />
                    <xs:element ref=”rating” />
                    <xs:element ref=”comment” />
                </xs:complexType>
            </xs:element>
        </xs:complexType>
    </xs:element>
    <xs:element name=”address” type=”xs:string” />
    <xs:element name=”hotelname” type=”xs:string” />
    <xs:element name=”rating” type=”xs:integer” />
    <xs:element name=”comment” type=”xs:string” />
</xs:schema>
```

Sample Hotel Review XML Schema

```
<?xml version=”1.0” encoding=”UTF-8”?>
<xs:schema targetNamespace=”http://www.aadx.org/schema/hotelreview2.xsd”
            xmlns:xs=”http://www.w3.org/2001/XMLSchema”
            xmlns:r=”http://www.aadx.org/schema/hotelreview2.xsd” >
    <xs:element name=”reviews” maxOccurs=”unbounded” >
        <xs:complexType>
        <xs:element name=”hotel”>
                <xs:complexType>
                    <xs:element name=”hotel_name” type=”xs:string”>
                    <xs:element name=”location” type=”r:addressType”>
                    <xs:element name=”review” type=”xs:string” />
                </xs:complexType>
        </xs:element>
        </xs:complexType>
    </xs:element>
    <xs:complexType name=”addressType”>
        <xs:choice maxOccurs=”unbounded”>
        <xs:element name=”address” type=”xs:string” />
        <xs:element name=”suburb” type=”xs:string” />
        <xs:element name=”state” type=”xs:string” />
        <xs:element name=”country” type=”xs:string” />
      </xs:choice>
    </xs:complexType>
</xs:schema>
```

A.2 Sample XML Documents

Sample Hotel XML Document

```
<?xml version=”1.0” encoding=”utf-8”?>
<list>
  <hotel>
    <accommodation>
        <location>
          <country>
            <country_name>australia</country_name>
            <state>
                <suburb_name>sydney</suburb_name>
                <state_name>nsw</state_name>
            </state>
          </country>
          <address>265 broadway</address>
        </location>
    </accommodation>
    <classification_in_stars>4</classification_in_stars>
    <hotel_name>four seasons sydney resort</hotel_name>
    <room>
        <has_tv>true</has_tv>
        <type>single</type>
        <has_mini_bar>true</has_mini_bar>
    </room>
    <room>
        <has_tv>true</has_tv>
        <type>double</type>
        <has_mini_bar>false</has_mini_bar>
    </room>
    <room>
        <has_tv>true</has_tv>
        <type>suite</type>
        <has_mini_bar>true</has_mini_bar>
    </room>
    <room>
        <has_tv>false</has_tv>
        <type>single</type>
        <has_mini_bar>false</has_mini_bar>
    </room>
  </hotel>
  <hotel>...</hotel>
  ...
</list>
```

Sample Hotel XML Document

```
<?xml version=”1.0” encoding=”utf-8”?>
<hotels>
  <hotel>
  <hotelname>novotel sydney resort</hotelname>
  <website>www.novotel.com.au</website>
  <phone>91234567</phone>
  <address>265 broadway street, Sydney, Australia</address>
  <room>
    <typeName>Standard Double</typeName>
    <roomRate>124.00</roomRate>
    <guestNumber>2</guestNumber>
    <equips>
    <aircondition>true</aircondition>
    <tv>true</tv>
    <shower>true</shower>
    </equips>
  </room>
  <room>
    <typeName>Delux Double</typeName>
    <roomRate>224.00</roomRate>
    <guestNumber>2</guestNumber>
    <equips>
      <aircondition>true</aircondition>
      <tv>true</tv>
```

```
            <shower>true</shower>
            <livingroom>true</livingroom>
        </equips>
    </room>
    <room>
        <typeName>Standard Single</typeName>
        <roomRate>89.00</roomRate>
        <guestNumber>1</guestNumber>
        <equips>
        <aircondition>true</aircondition>
        <tv>true</tv>
        <shower>true</shower>
        </equips>
    </room>
    <room>
    <typeName>High Deluxe</typeName>
    <roomRate>149.00</roomRate>
    <guestNumber>2</guestNumber>
    <equips>
        <aircondition>true</aircondition>
        <tv>true</tv>
        <shower>true</shower>
    </equips>
    </room>
    </hotel>
    <hotel>...</hotel>
</hotels>
```

Sample Hotel Review XML Document

```
<?xml version="1.0" encoding="UTF-8"?>
<reviews>
    <review>
        <hotelname>Southern Cross Hotel Sydney</hotelname>
        <address>99 King George St, Sydney, Australia</address>
        <rating>4</rating>
        <comment>Very clean, and good breakfast.</comment>
    </review>
    <review>
        <hotelname>Allis Hotel Melbourne</hotelname>
        <address>23 Williams St, Melbourne, Australia</address>
        <rating>4</rating>
        <comment>Good service, and room is large and bright!</comment>
    </review>
    <review>
        <hotelname>Wilston Hotel Darling Harbour</hotelname>
        <address>123 Darling St, Darling Harbour, Australia</address>
        <rating>5</rating>
        <comment>Nice view, good location and high quality food!</comment>
    </review>
    <review>
        ...      </review>
</reviews>
```

Sample Hotel Review XML Document

```
<?xml version=”1.0” encoding=”UTF-8”?>
<reviews>
    <hotel>
        <hotel_name>Southern Cross Hotel Sydney</hotel_name>
        <location>
            <address>99 King George St</address>
            <suburb>Sydney</suburb>
            <state>NSW</state>
            <country>Australia</country>
        </location>
        <review>Very clean, and good breakfast.</review>
    </hotel>
    <hotel>
        <hotel_name>Allis Hotel Melbourne</hotel_name>
        <location>
            <address>23 Williams St</address>
            <suburb>Melbourne</suburb>
            <state>VIC</state>
            <country>Australia</country>
        </location>
        <review>Good service, and room is large and bright!</review>
    </hotel>
    <hotel>      ...      </hotel>
</reviews>
```

Sample Accommodation Ontology

```
<?xml version=”1.0”?>
<rdf:RDF xml:base=”http://localhost:8084/HubWebApp/Ontology/hotel.owl”
                          xmlns=”http://localhost:8084/HubWebApp/Ontology/hotel.owl#”
                          xmlns:owl=”http://www.w3.org/2002/07/owl#”
                          xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
                          xmlns:rdfs=”http://www.w3.org/2000/01/rdf-schema#”
                          xmlns:xsd=”http://www.w3.org/2001/XMLSchema#”>
<owl:Ontology rdf:about=””/>
<owl:Class rdf:about=”#Hotel”>
<rdfs:subClassOf rdf:resource=”http://www.w3.org/2002/07/owl#Thing”/>
</owl:Class>
<owl:Class rdf:about=”#HotelReview”>
<rdfs:subClassOf rdf:resource=”http://www.w3.org/2002/07/owl#Thing”/>
</owl:Class>
<owl:Class rdf:ID=”Address”/>
<owl:Class rdf:ID=”Room”/>
<owl:Class rdf:ID=”Price”/>
<owl:Class rdf:ID=”Folio”/>
<owl:Class rdf:ID=”RoomService”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”MiniBar”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”Bar”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”GiftShop”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”Laundry”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”Meal”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”Movie”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”Phone”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
```

```
</owl:Class>
<owl:Class rdf:ID=”Internet”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:Class rdf:ID=”OtherService”>
<rdfs:subClassOf rdf:resource=”#Folio”/>
</owl:Class>
<owl:ObjectProperty rdf:about=”#locateAt”>
<rdfs:range rdf:resource=”#Address”/>
<rdfs:domain rdf:resource=”#Hotel”/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:about=”#hasRoom”>
<rdfs:range rdf:resource=”#Room”/>
<rdfs:domain rdf:resource=”#Hotel”/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=”rate”>
<rdfs:range rdf:resource=”#Price”/>
<rdfs:domain rdf:resource=”#Room”/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=”additionalService”>
<rdfs:range rdf:resource=”#Hotel”/>
<rdfs:domain rdf:resource=”#Folio”/>
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=”feedback”>
<rdfs:range rdf:resource=”#Hotel”/>
<rdfs:domain rdf:resource=”#HotelReview”/>
</owl:ObjectProperty>
<owl:DatatypeProperty rdf:ID=”hotelName”>
<rdfs:range rdf:resource=”rdfs#Literal”/>
<rdfs:domain rdf:resource=”#Hotel”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”city”>
<rdfs:domain rdf:resource=”#Address”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”state”>
<rdfs:domain rdf:resource=”#Address”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”country”>
<rdfs:domain rdf:resource=”#Address”/>
```

```
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”roomType”>
<rdfs:domain rdf:resource=”#Room”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”maxPerson”>
<rdfs:domain rdf:resource=”#Room”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”amount”>
<rdfs:domain rdf:resource=”#Price”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”comment”>
<rdfs:domain rdf:resource=”#HotelReview”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”rating”>
<rdfs:domain rdf:resource=”#HotelReview”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
<owl:DatatypeProperty rdf:ID=”review”>
<rdfs:domain rdf:resource=”#HotelReview”/>
<rdfs:range rdf:resource=”rdfs#Literal”/>
</owl:DatatypeProperty>
</rdf:RDF>
```

Appendix B

Accommodation Hub

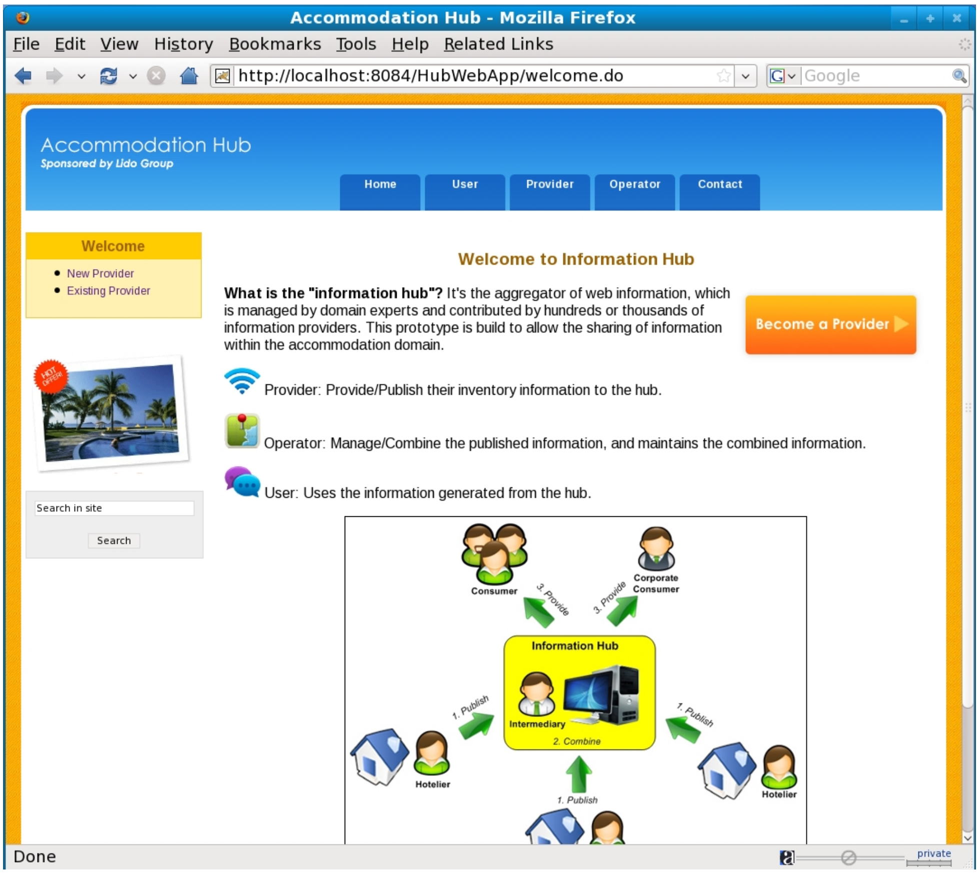
In chapter 7 we have introduced the accommodation hub prototype system, however, we did not provide comprehensive instructions on the functionality offered by the system. This section explains the accommodation hub system in further detail, and we provide a comprehensive description on each implemented system feature. The accommodation hub system is implemented as a web application, and is formed by three web portals: provider portal, operator portal, and user portal. Figure B.1 shows the accommodation hub system.

B.1 Provider Portal

The provider portal is designed to allow information providers for publishing their information sources. Features implemented in the provider portal include account registration, source publishing, and source management.

B.1.1 Account Registration

In order to gain access to the provider portal, providers need to register an account on the accommodation hub. Account registration is done by filling out an account



Welcome to Information Hub

What is the "information hub"? It's the aggregator of web information, which is managed by domain experts and contributed by hundreds or thousands of information providers. This prototype is build to allow the sharing of information within the accommodation domain.

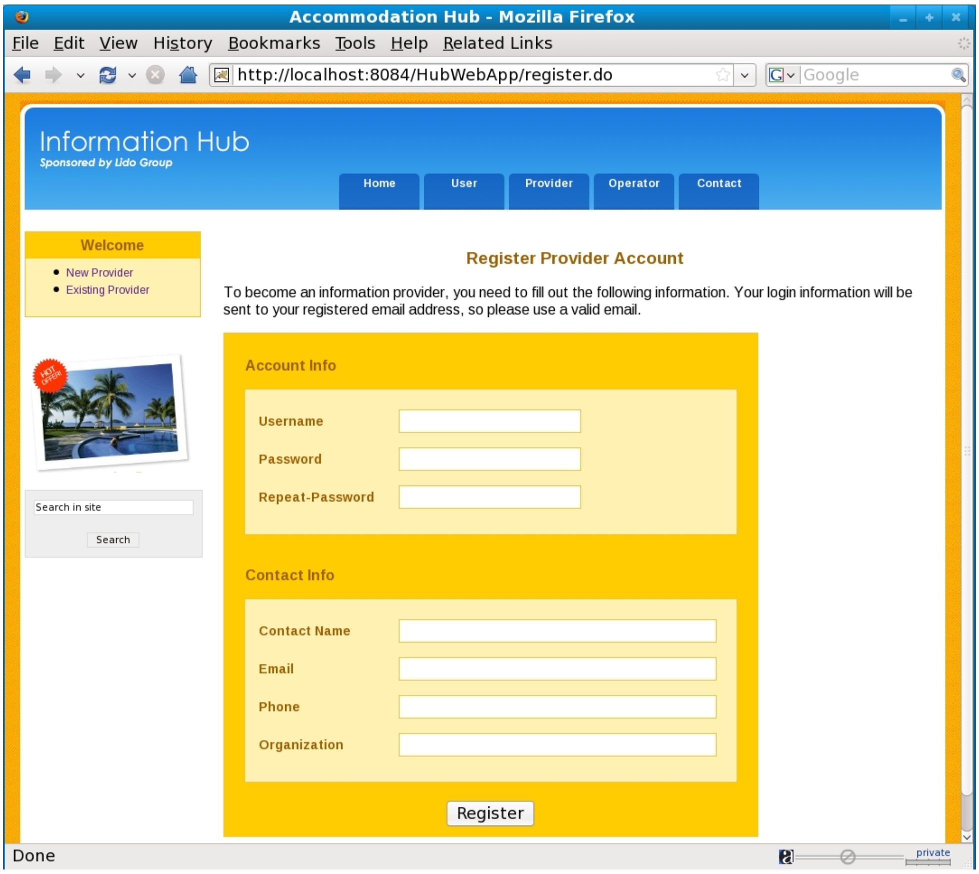
Provider: Provide/Publish their inventory information to the hub.

Operator: Manage/Combine the published information, and maintains the combined information.

User: Uses the information generated from the hub.

Figure B.1: Accommodation hub

registration form, as shown in figure B.8. The registered account information will be automatically sent to the provider by email.



Register Provider Account

To become an information provider, you need to fill out the following information. Your login information will be sent to your registered email address, so please use a valid email.

Account Info

Username

Password

Repeat-Password

Contact Info

Contact Name

Email

Phone

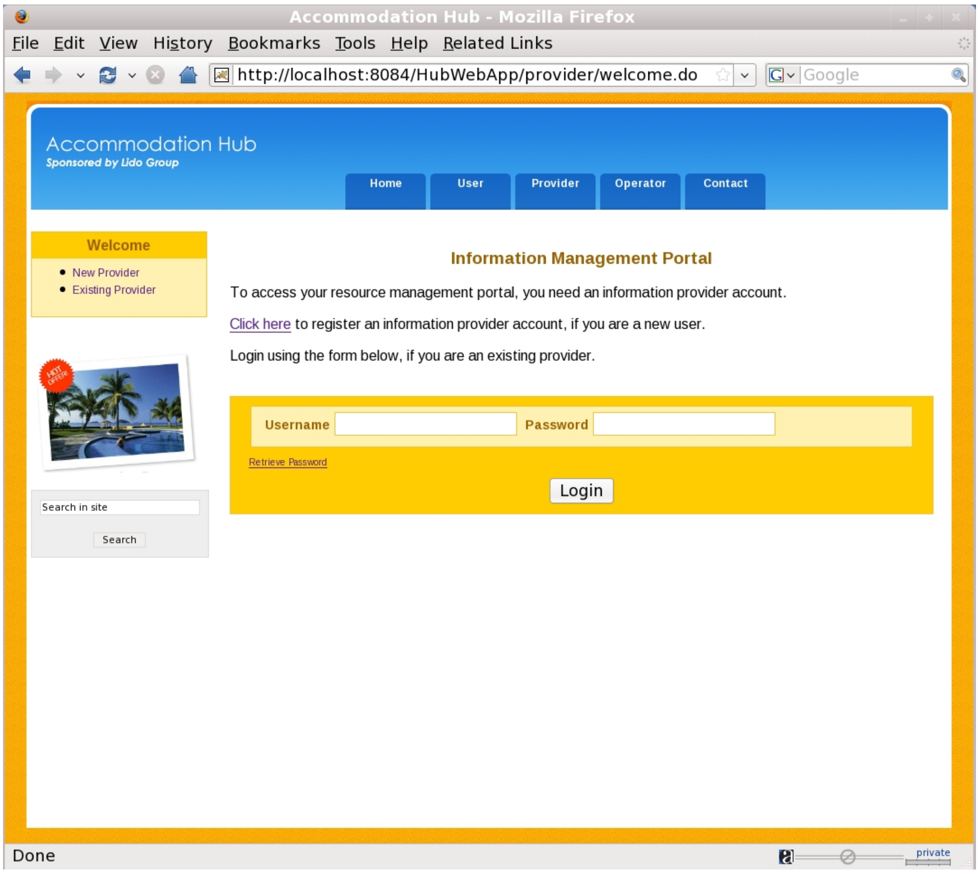
Organization

Register

Figure B.2: Account registration

B.1.2 Log In

After account registration, information providers can log onto the accommodation hub system via the log in form. The log in form also provide security access for information operator.



Information Management Portal

To access your resource management portal, you need an information provider account.

Click here to register an information provider account, if you are a new user.

Login using the form below, if you are an existing provider.

Username
Password

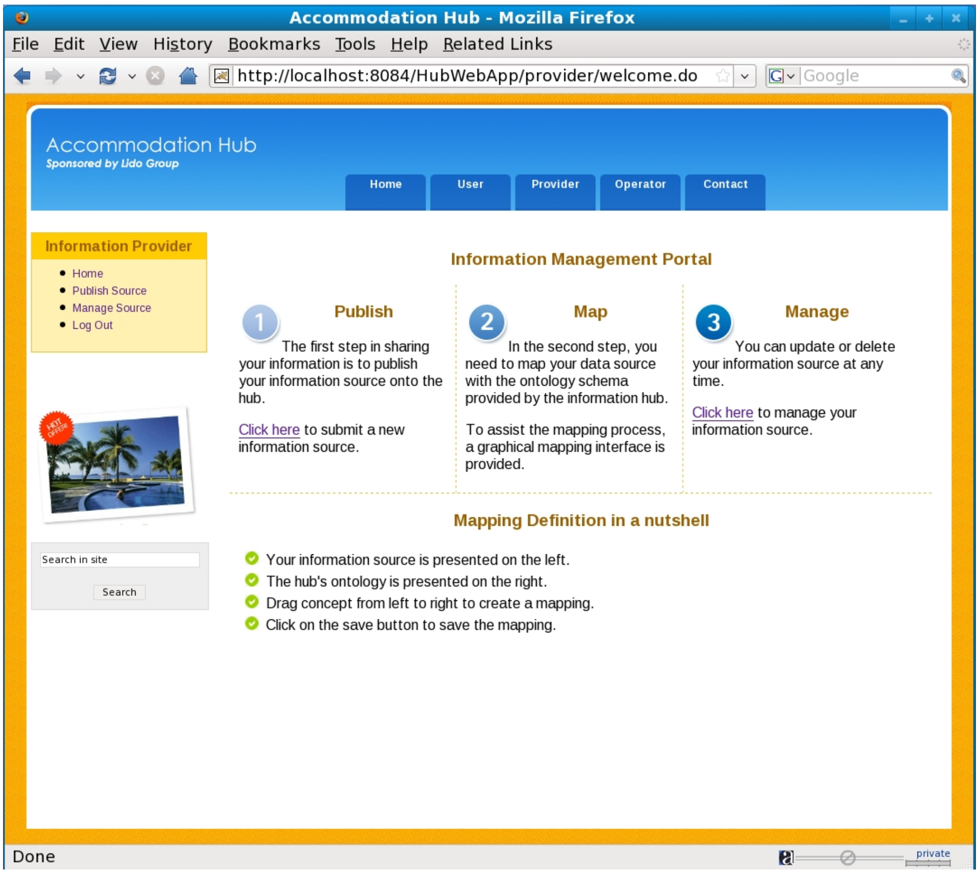
Retrieve Password

Login

Figure B.3: Login form

B.1.3 Provider Portal

The provider portal provides a list of features for the publication and management of information sources owned by information providers. The left hand side menu provides navigation to the list of implemented features. Providers can also access the implemented feature via the links provided on the main page.



Information Management Portal

1 Publish

The first step in sharing your information is to publish your information source onto the hub.

Click here to submit a new information source.

2 Map

In the second step, you need to map your data source with the ontology schema provided by the information hub.

To assist the mapping process, a graphical mapping interface is provided.

3 Manage

You can update or delete your information source at any time.

Click here to manage your information source.

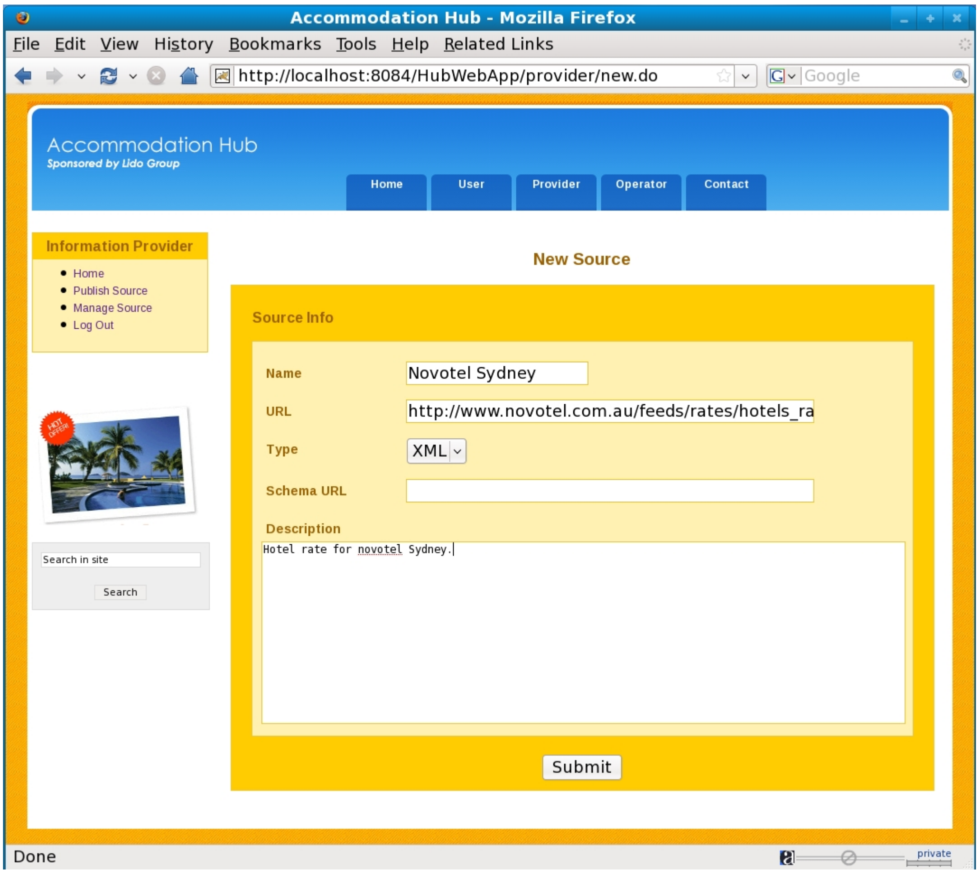
Mapping Definition in a nutshell

- Your information source is presented on the left.
- The hub's ontology is presented on the right.
- Drag concept from left to right to create a mapping.
- Click on the save button to save the mapping.

Figure B.4: Provider portal

B.1.4 Publish Information Source

The publishing of an information source is performed in two steps. In the first step, an information provider submits meta-data information relating to the published information source, including its source name, the link to the information source, file type, schema link, and some text to describe the published source. All those information will be stored into the accommodation hub.



New Source

Source Info

Name: Novotel Sydney

URL: http://www.novotel.com.au/feeds/rates/hotels_ra

Type: XML

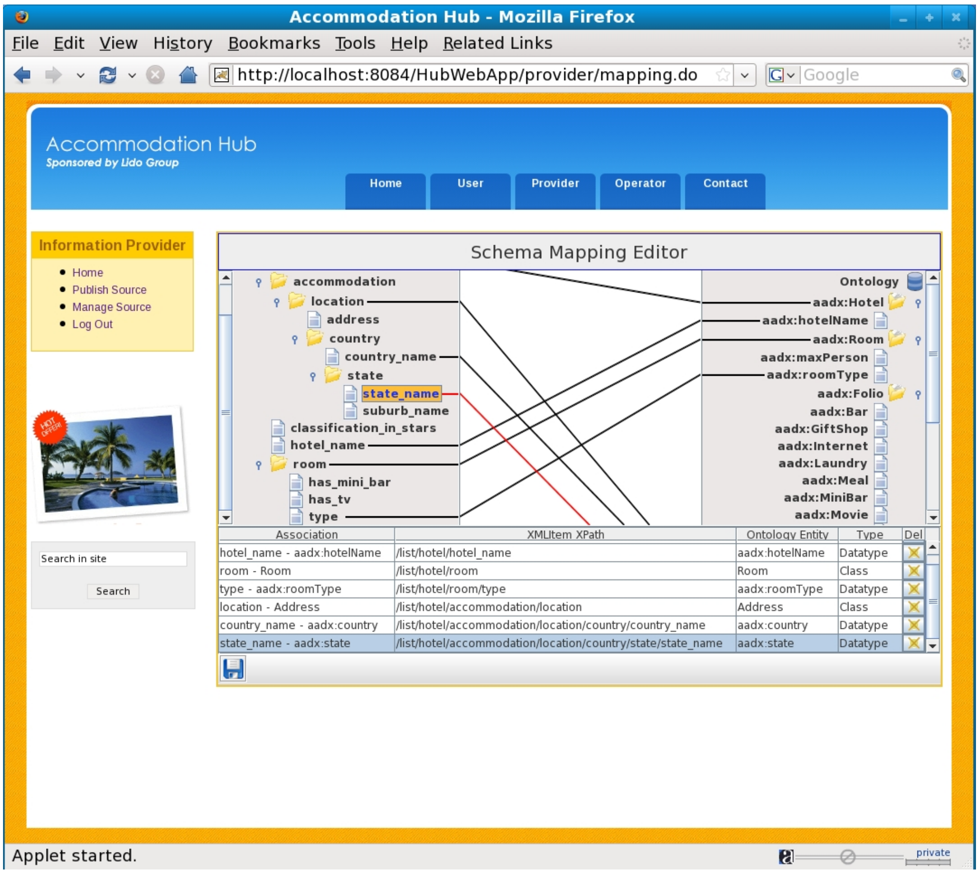
Schema URL:

Description: Hotel rate for novotel Sydney.

Submit

Figure B.5: Publish source

In the second step, the schema from the submitted information source will be automatically loaded into the mapping tool, as shown in figure B.6. At the same time, the global ontology shared within the accommodation hub will also be loaded into the mapping tool, as shown on the right hand side of figure B.6. Concept mappings between corresponding concepts defined from the XML schema and the global ontology are defined by the information provider. For example, the concept "hotel_name" is mapped to the concept "aadx:hotelName" defined from the ontology. The defined schema mappings are stored into the hub for query purposes. The implementation of the mapping tool follows the theories explained in chapter 4.

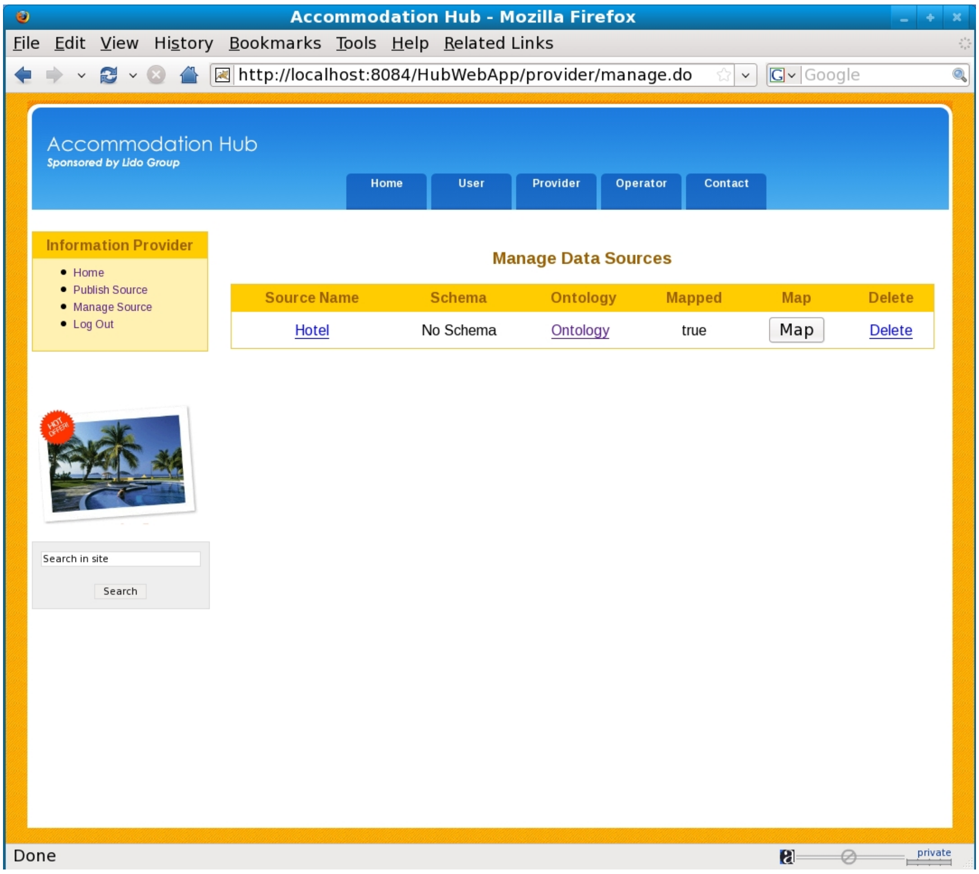


Association	XMLItem XPath	Ontology Entity	Type	Del
hotel_name - aadx:hotelName	/list/hotel/hotel_name	aadx:hotelName	Datatype	☒
room - Room	/list/hotel/room	Room	Class	☒
type - aadx:roomType	/list/hotel/room/type	aadx:roomType	Datatype	☒
location - Address	/list/hotel/accommodation/location	Address	Class	☒
country_name - aadx:country	/list/hotel/accommodation/location/country/country_name	aadx:country	Datatype	☒
state_name - aadx:state	/list/hotel/accommodation/location/country/state/state_name	aadx:state	Datatype	☒

Figure B.6: Mapping definition

B.1.5 Manage Source

The provider portal also provides functionality for the management of published information sources. As shown in figure B.7, information providers are able to remove and update the published information source from the source management page.



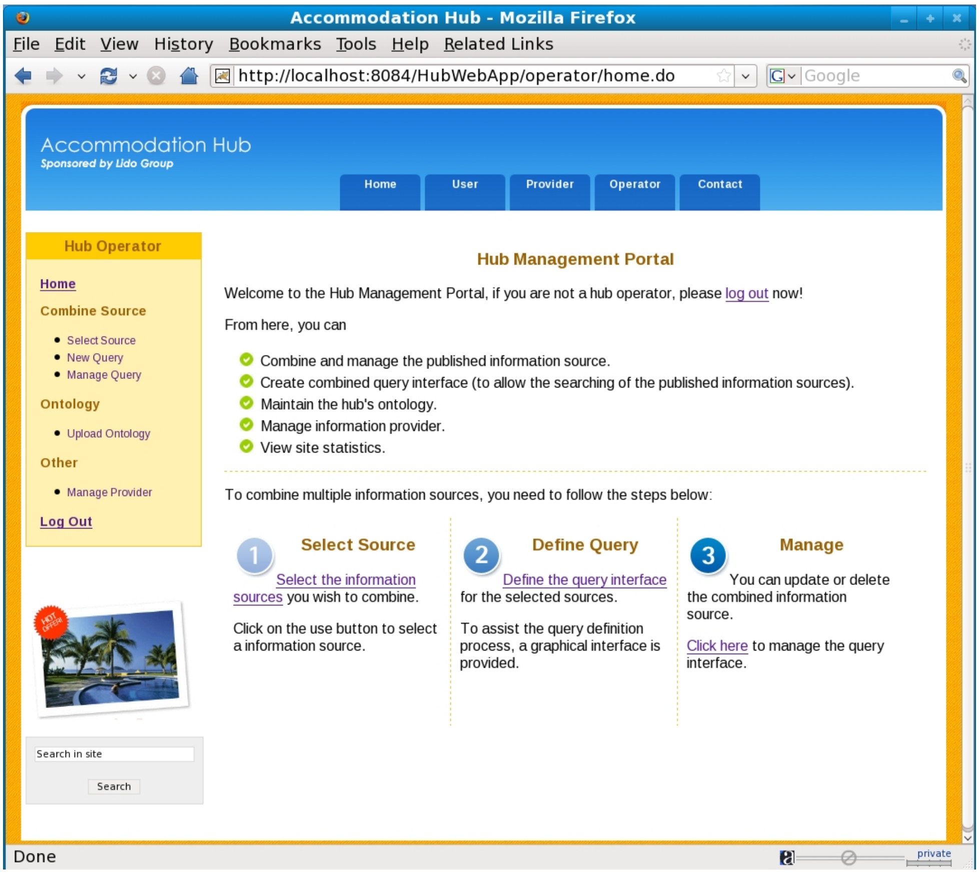
Manage Data Sources

Source Name	Schema	Ontology	Mapped	Map	Delete
Hotel	No Schema	Ontology	true	Map	Delete

Figure B.7: Manage source

B.2 Operator Portal

The operator portal (hub management portal) provides management functions for information operators. Features implemented in the operator portal include source management, query management, and ontology management. One major task in the information integration process is the integration of the published information sources, and this task is performed by hub operator via the operator portal.



Hub Management Portal

Welcome to the Hub Management Portal, if you are not a hub operator, please log out now!

From here, you can

- Combine and manage the published information source.
- Create combined query interface (to allow the searching of the published information sources).
- Maintain the hub's ontology.
- Manage information provider.
- View site statistics.

To combine multiple information sources, you need to follow the steps below:

1 Select Source

Select the information sources you wish to combine.

Click on the use button to select a information source.

2 Define Query

Define the query interface for the selected sources.

To assist the query definition process, a graphical interface is provided.

3 Manage

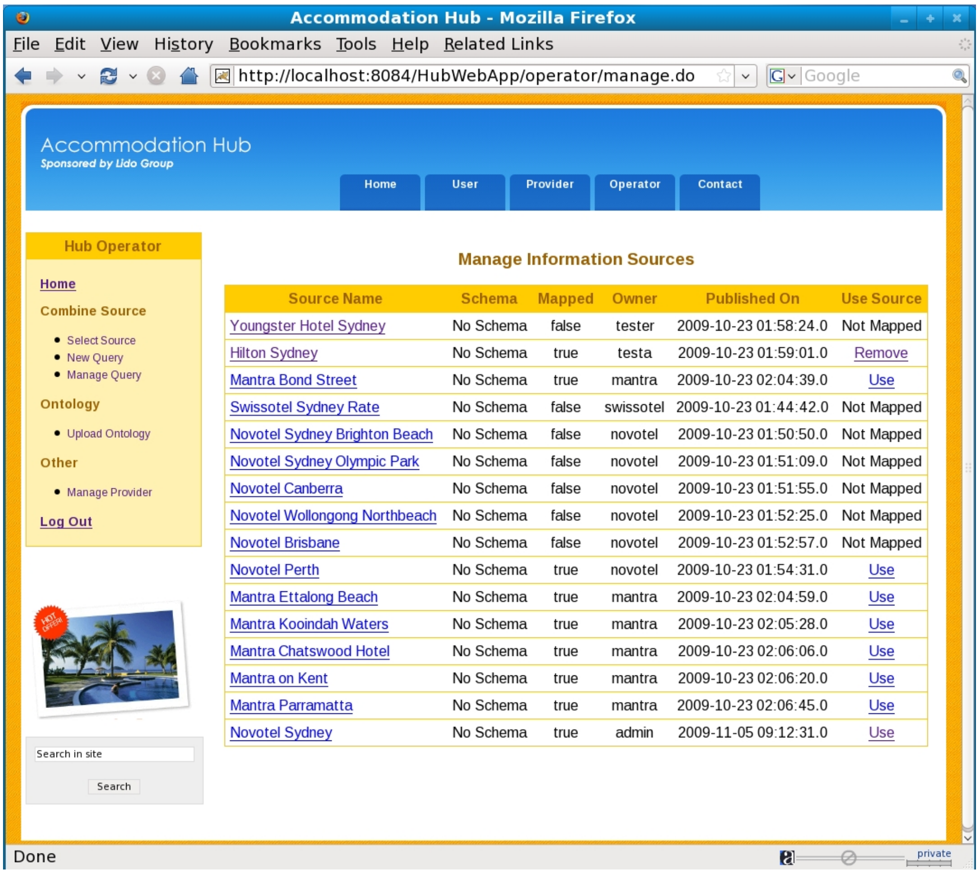
You can update or delete the combined information source.

Click here to manage the query interface.

Figure B.8: Operator portal

B.2.1 Source Management

First step towards the combination of the published sources is source selection. This is done by using the source management function. As shown in figure B.9, a list of published sources are displayed to hub operators. From the portal, operators can select the sources they wish to use for the integration process. All other source relevant information is also displayed to the operator, such as the owner of the source, the publishing date, mapping status, etc.



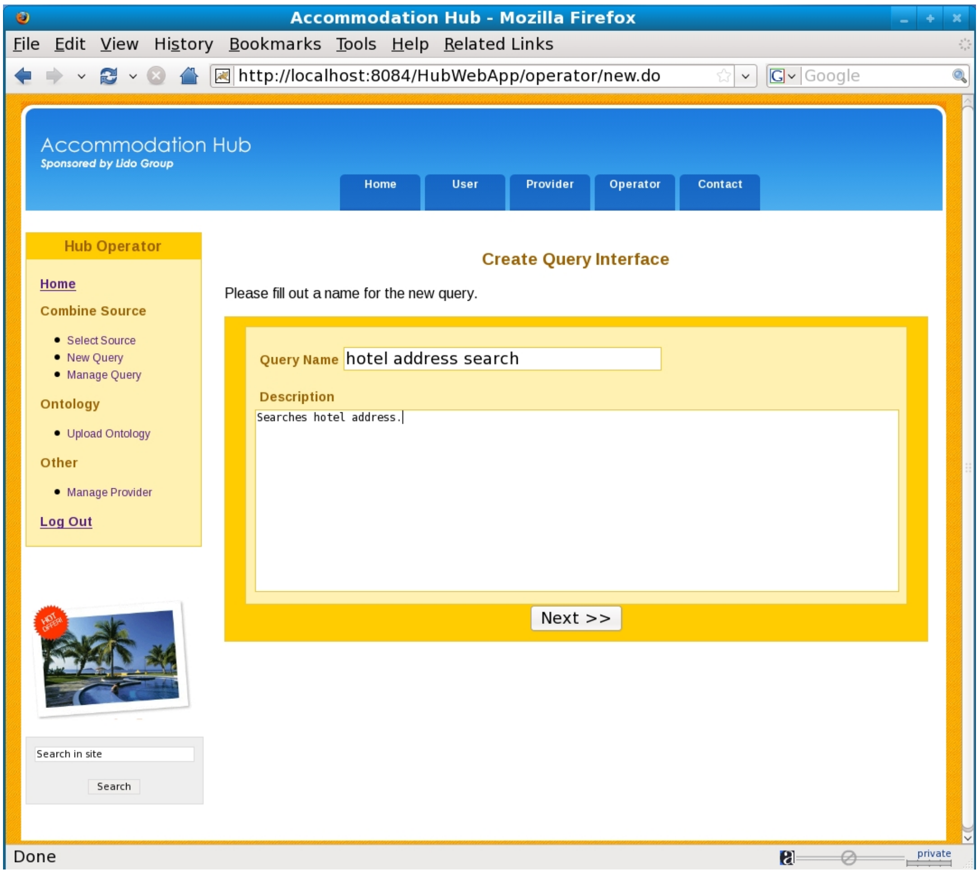
Manage Information Sources

Source Name	Schema	Mapped	Owner	Published On	Use Source
Youngster Hotel Sydney	No Schema	false	tester	2009-10-23 01:58:24.0	Not Mapped
Hilton Sydney	No Schema	true	testa	2009-10-23 01:59:01.0	Remove
Mantra Bond Street	No Schema	true	mantra	2009-10-23 02:04:39.0	Use
Swissotel Sydney Rate	No Schema	false	swissotel	2009-10-23 01:44:42.0	Not Mapped
Novotel Sydney Brighton Beach	No Schema	false	novotel	2009-10-23 01:50:50.0	Not Mapped
Novotel Sydney Olympic Park	No Schema	false	novotel	2009-10-23 01:51:09.0	Not Mapped
Novotel Canberra	No Schema	false	novotel	2009-10-23 01:51:55.0	Not Mapped
Novotel Wollongong Northbeach	No Schema	false	novotel	2009-10-23 01:52:25.0	Not Mapped
Novotel Brisbane	No Schema	false	novotel	2009-10-23 01:52:57.0	Not Mapped
Novotel Perth	No Schema	true	novotel	2009-10-23 01:54:31.0	Use
Mantra Ettalong Beach	No Schema	true	mantra	2009-10-23 02:04:59.0	Use
Mantra Kooindah Waters	No Schema	true	mantra	2009-10-23 02:05:28.0	Use
Mantra Chatswood Hotel	No Schema	true	mantra	2009-10-23 02:06:06.0	Use
Mantra on Kent	No Schema	true	mantra	2009-10-23 02:06:20.0	Use
Mantra Parramatta	No Schema	true	mantra	2009-10-23 02:06:45.0	Use
Novotel Sydney	No Schema	true	admin	2009-11-05 09:12:31.0	Use

Figure B.9: Source management

B.2.2 Query Definition

As mentioned in earlier chapters, to provide users with easy-to-use query interfaces, hub operators needs to define a number of popular query templates. The definition of a query template is achieved in two sequential steps. In step one, a hub operator specifies the basic information used for the query template, such as query name and description.



Create Query Interface

Please fill out a name for the new query.

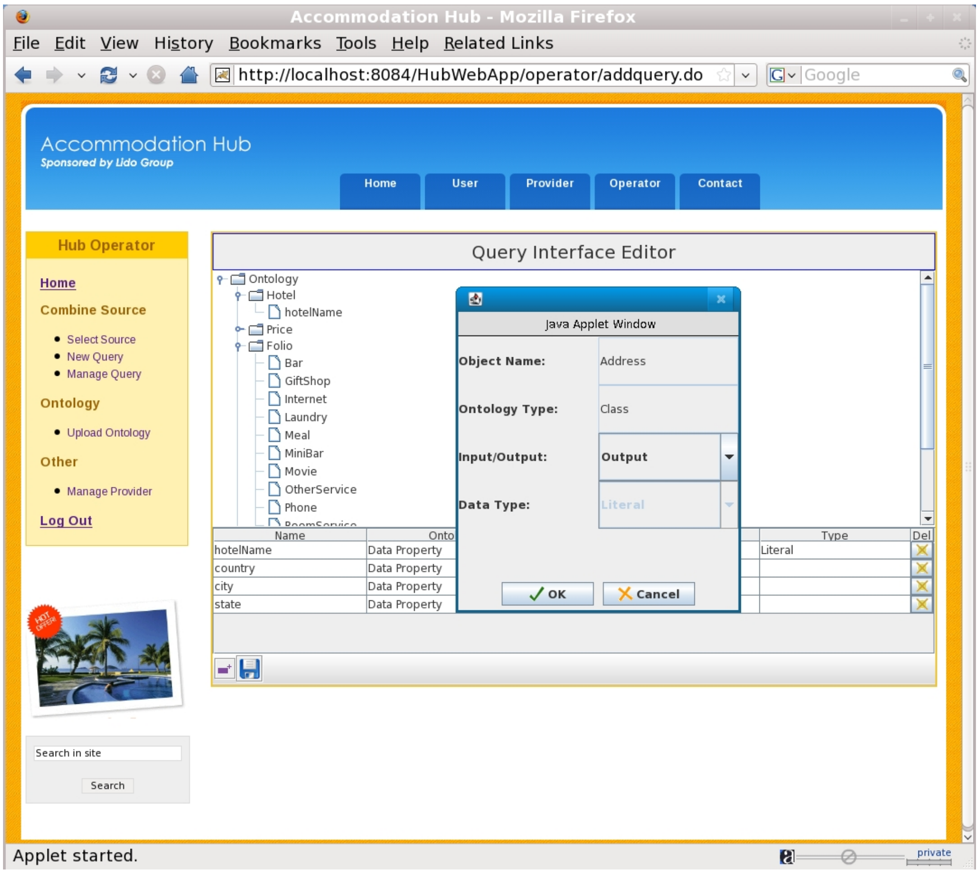
Query Name: hotel address search

Description: Searches hotel address.

Next >>

Figure B.10: Query definition step 1

In step two, the hub operator defines the set of query parameters. As previously mentioned, the definition of a query parameter is performed by selecting the corresponding concept from the ontology tree. A pop up window is displayed to define the parameter related properties.



Object Name: Address

Ontology Type: Class

Input/Output: Output

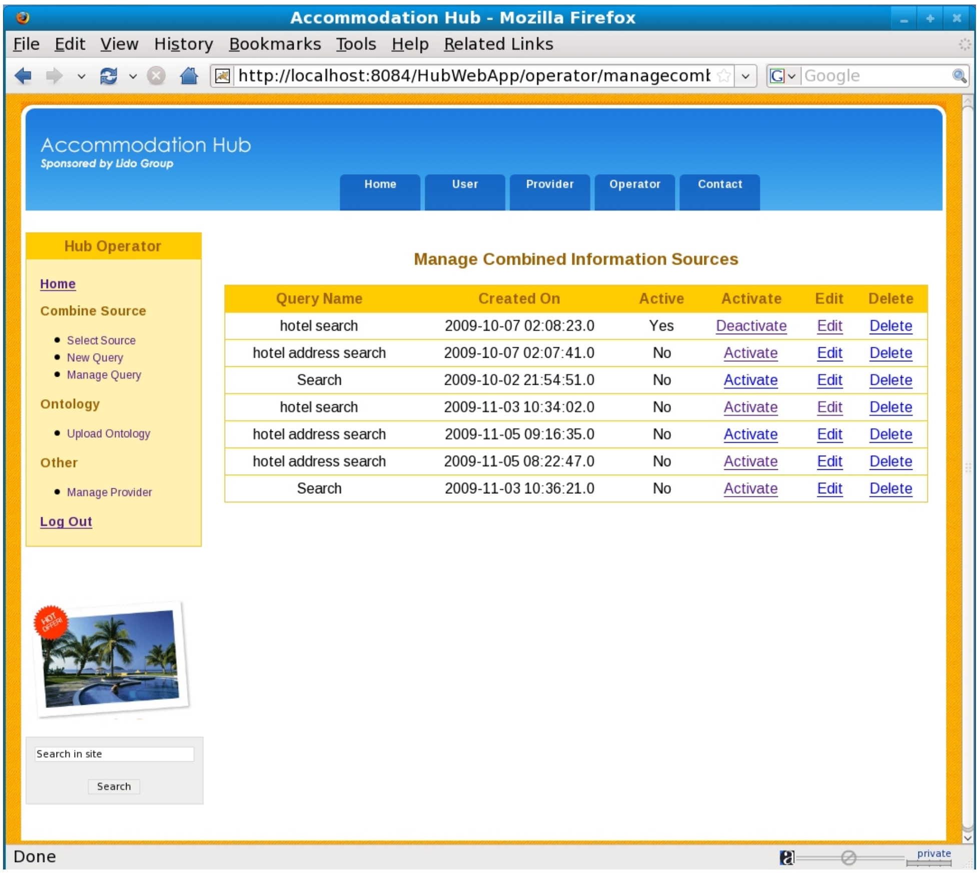
Data Type: Literal

Name	Onto	Type	Del
hotelName	Data Property	Literal	☒
country	Data Property		☒
city	Data Property		☒
state	Data Property		☒

Figure B.11: Query definition step 2

B.2.3 Query Management

The defined queries are managed by hub operators via the query management portal. As shown in figure B.12, the activate link allows the activation of a defined query. Once the query is activated, it will be displayed to the public information users. The edit link allows the update of a defined query interface, and the delete link enables the removal of a defined query interface.



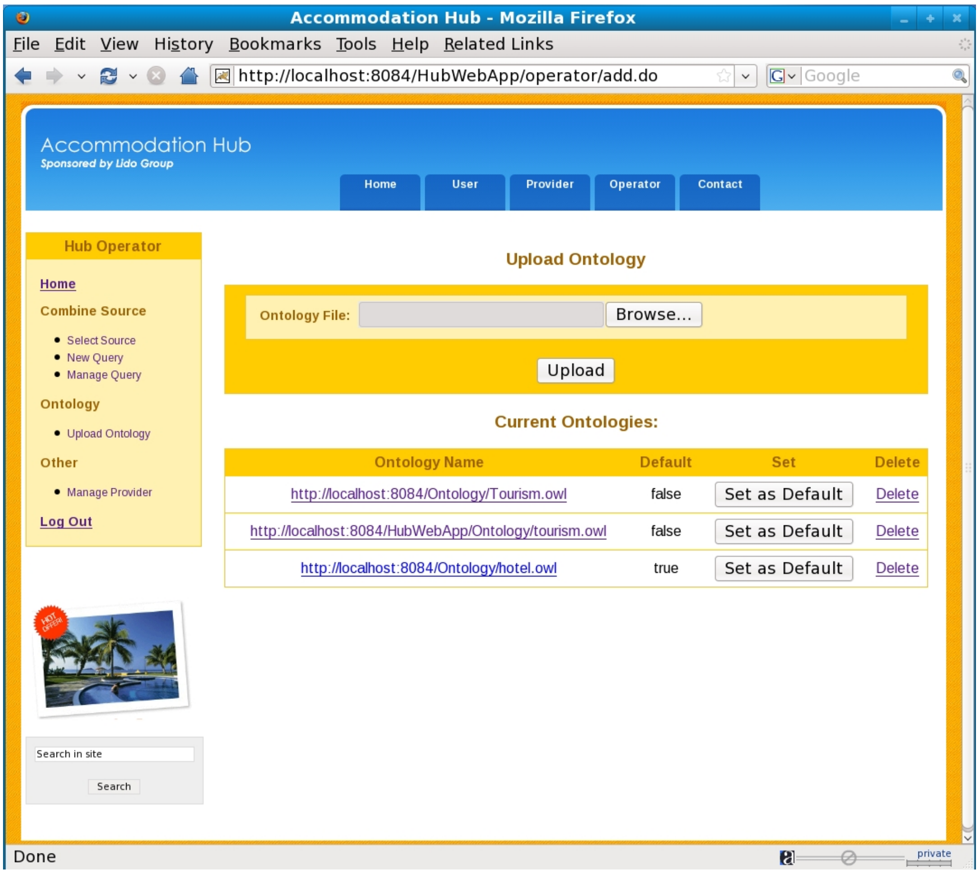
Manage Combined Information Sources

Query Name	Created On	Active	Activate	Edit	Delete
hotel search	2009-10-07 02:08:23.0	Yes	Deactivate	Edit	Delete
hotel address search	2009-10-07 02:07:41.0	No	Activate	Edit	Delete
Search	2009-10-02 21:54:51.0	No	Activate	Edit	Delete
hotel search	2009-11-03 10:34:02.0	No	Activate	Edit	Delete
hotel address search	2009-11-05 09:16:35.0	No	Activate	Edit	Delete
hotel address search	2009-11-05 08:22:47.0	No	Activate	Edit	Delete
Search	2009-11-03 10:36:21.0	No	Activate	Edit	Delete

Figure B.12: Query Management

B.2.4 Ontology Management

Another key feature provided by the operator portal is the management of shared ontology models. The shared ontology acts as the heart to the accommodation hub system, it is used for reconciliating the structure differences encountered in heterogeneous information sources. In the accommodation hub system, an ontology is defined by hub operators using third party tools, and the defined ontology is uploaded to the hub system from the ontology management page. As shown in figure B.13, the ontology management page also provides functions for the update and deletion of the uploaded ontology schemas.



Upload Ontology

Ontology File: Browse...

Upload

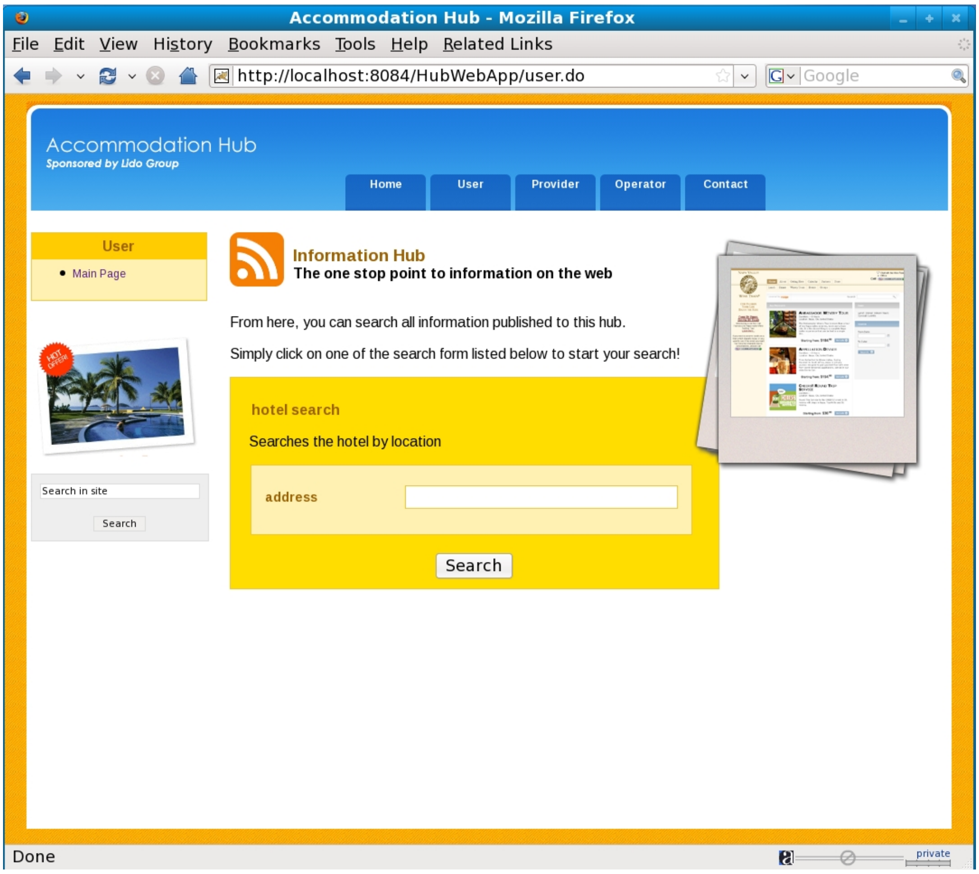
Current Ontologies:

Ontology Name	Default	Set	Delete
http://localhost:8084/Ontology/Tourism.owl	false	Set as Default	Delete
http://localhost:8084/HubWebApp/Ontology/tourism.owl	false	Set as Default	Delete
http://localhost:8084/Ontology/hotel.owl	true	Set as Default	Delete

Figure B.13: Ontology management

B.3 User Portal

Finally, the user portal provides integrated access to the set of published information sources in an organized manner. All the defined query templates are displayed to users as web forms. Figure B-7 shows the hotel search query displayed in web form.



Information Hub
The one stop point to information on the web

From here, you can search all information published to this hub.

Simply click on one of the search form listed below to start your search!

hotel search

Searches the hotel by location

address

Figure B.14: User portal

References

Aberer, K., Cudré-Mauroux, P. & Hauswirth, M. (2002). A framework for semantic gossiping. *SIGMOD Record*, **31**, 2002. 44, 47

Abiteboul, S., Buneman, P. & Suciu, D. (1999). *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann. 6

ABS (2007). *Australian Bureau of Statistics Yearbook 2007*. ABS Australian Bureau of Statistics, canberra, australia. 4

ACNielsen (2006). Global consumer confidence report. 1–4. 1

Alonso, G. & Casati, F. (2005). Web services and service-oriented architectures. In *In ICDE*, 1147. 81

Angles, R. & Gutierrez, C. (2008). The expressive power of sparql. In *International Semantic Web Conference*, 114–129. 126

Antoniou, G. & Harmelen, F.v. (2004). *A Semantic Web Primer*. MIT Press, Cambridge. 5, 61

Arens, Y., Chee, C.Y., Hsu, C.N. & Knoblock, C.A. (1993). Retrieving and integrating datafrom multiple information sources. 34

Ariely, D. (2000). Controlling the information flow: Effects on consumers decision making and preferences. *Journal of Consumer Research*, **27**, 233–248. 2

Baader, F., Horrocks, I. & Sattler, U. (2001). Description logics for the semantic web. *IEEE Data Engineering Bulletin*, **16**, 4–9. 74

Baru, C.K., Gupta, A., Ludäscher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P. & Chu, V. (1999). Xml-based information mediation with mix. In *SIGMOD Conference*, 597–599. 33, 36

Biron, P.V. & Malhotra, A., eds. (2004). *XML Schema Part 2: Datatypes*. W3C Recommendation, W3C, 2nd edn. 127

Bohring, H. & Auer, S. (2005). Mapping xml to owl ontologies. In *Leipziger Informatik-Tage, volume 72 of LNI*, 147–156, GI. 93, 94, 95

Brachman, R.J., McGuinness, D.L., Patel-Schneider, P.F., Resnick, L.A., Resnick, L.A. & Borgida, A. (1991). Living with classic: When and how to use a kl-one-like language. In *Principles of Semantic Networks*, 401–456, Morgan Kaufmann. 37

Braga, D., Campi, A., Cappa, R. & Salvi, D. (2005). Xslt by example. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, 1158–1159, ACM, New York, NY, USA. 130

Bray, T., Paoli, J. & Sperberg-McQueen, C.M. (1998). Extensible markup language (xml) 1.0. *W3C Recommendation*. 8

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984). Classification and regression trees. *Wadsworth International Group*. 108, 109

Broekstra, J., Ehrig, M., Haase, P., Harmelen, F.V., Kampman, A., Sabou, M., Siebes, R., Staab, S., Stuckenschmidt, H. & Tempich, C. (2003). A metadata model for semantics-based peer-to-peer systems. In *In: Proceedings of the WWW03 Workshop on Semantics in Peer-to-Peer and Grid Computing*. 11, 42, 57, 81

Bryan, M. (1988). *SGML : an author's guide to the standard generalized markup language*. Addison-Wesley. 6

Cali, A., Calvanese, D., Giacomo, G.D. & Lenzerini, M. (2002). On the expressive power of data integration systems. In *In: Proceedings of ER*, 338–350. 41

Calvanese, D., Giacomo, G., Lembo, D. & Lenzerini, M. (2008). Data integration through dl-litea ontologies. *3rd International Workshop on Semantics in Data and Knowledge Bases*. 55, 56

Carey, M.J., Haas, L.M., Schwarz, P.M., Arya, M., Cody, W.F., Fagin, R., Thomas, J., H, J. & Wimmers, E.L. (1995). Towards heterogeneous multimedia information systems: The garlic approach. In *In RIDE-DOM*, 124–131. 31, 36

Carroll, J.J., Dickinson, I., Dollin, C., Seaborne, A., Wilkinson, K., Reynolds, D. & Reynolds, D. (2003). Jena: Implementing the semantic web recommendations. 74–83. 140

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. & Widom, J. (1994). The tsimmis project: Integration of heterogeneous information sources. In *In Proceedings of IPSJ Conference*, 7–18. 11, 29, 30, 36, 74

Chiu, W. (2001). Design scalability-an update, high volume web sites, software group. 184

Comito, C., Talia, D. & Trunfio, P. (2010). Selectivity-based xml query processing in structured peer-to-peer networks. *Proceedings of the 14th International Database Engineering and Applications Symposium*. 57

Cruz, I.F., Xiao, H. & Hsu, F. (2004). Peer-to-peer semantic integration of xml and rdf data sources. In *In Third International Workshop on Agents and Peer-to-Peer Computing (AP2PC) 2004*. 34, 36, 74

Cudre-Mauroux, P., Aberer, K. & Feher, A. (2006). Probabilistic message passing in peer data management systems. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, 41, IEEE Computer Society, Washington, DC, USA. 49, 52

Dai, Y. & Zhang, S. (2006). Multi-agent based data integration in real-world. *Proceedings of the 6th IEEE International Conference on Computer and Information Technology*. 11, 21, 23, 24

Davies, J., Duke, A. & Stonkus, A. (2002). Ontoshare: Using ontologies for knowledge sharing. In *Semantic Web Workshop*. 72

Decker, S., Melnik, S., Harmelen, F.V., Fensel, D., Klein, M., Broekstra, J., Erdmann, M. & Horrocks, I. (2000). The semantic web: The roles of xml and rdf. *IEEE Internet Computing*, **4**, 63–74. 92

DERI (2009). Ontour ontology. *http://e-tourism.deri.at/ont/index.html*. 53, 54

Dinlersoz, E. & Hernändez-Murillo, R. (2005). The diffusion of electronic business in the united states. *Federal Reserve Bank of St. Louis Review*, 11–34. 4

Doan, A., Madhavan, J., Domingos, P. & Halevy, A. (2002). Learning to map between ontologies on the semantic web. *Proceedings of 11th International Conference on World Wide Web*, 662–673. 48

Doan, A., Domingos, P. & Halevy, A. (2003). Learning to match the schemas of data sources: A multistrategy approach. *VLDB Journal*, 279–301. 107

Dobra, A.V. (2003). *Scalable Classification and Regression Tree Construction*. Ph.D. thesis, Cornell University. 109

Ehrig, M. & Staab, S. (2004a). Efficiency of ontology mapping approaches. *International Workshop on Semantic Intelligent Middleware for the Web and the Grid (ECAI04)*, 47–61. 107, 117

Ehrig, M. & Staab, S. (2004b). Quick ontology mapping with qom. *Technical report, University of Karlsruhe, Institute*. 107

EON (2009). Eon traveling ontology. *http://homepage.cwi.nl/EON-TravellingOntology-v0.2.daml*. 53, 54

Fallside, D.C. (2000). Xml schema part 0: Primer. *W3C Working Draft*. 9, 31, 34

Ferdinand, M., Zirpins, C. & Trastour, D. (2004). Lifting xml schema to owl. In N. Koch, P. Fraternali & M. Wirsing, eds., *Web Engineering - 4th International Conference, ICWE 2004, Munich, Germany, July 26-30, 2004, Proceedings*, 354–358, Springer Heidelberg. 93, 94, 95

Fikes, R., Hayes, P. & Horrocks, I.R. (2003). Owl-ql - a language for deductive query answering on the semantic web. 139

FODOR, O. & WERTHER, H. (2005). Harmonise - a step towards an interoperable e-tourism marketplace. *International Journal of Electronice Commerce*. 53

FODOR, O., HÖPKEN, W. & WERTHNER, H. (2005). Exploiting semantic web technologies for harmonising e-markets. In *Tourism (JITT), Special Issue on Semantic Web Technologies and Applications*. 78

FRIEDMAN, M., LEVY, A. & MILLSTEIN, T. (1999). Navigational plans for data integration. In *In Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 67–73, AAAI Press/The MIT Press. 30, 41

FUNDULAKI, I., AMANN, B., BEERI, C., SCHOLL, M. & VERCOUSTRE, A.M. (2002). Styx: Connecting the xml web to the world of semantics. In *EDBT '02: Proceedings of the 8th International Conference on Extending Database Technology*, 759–761, Springer-Verlag, London, UK. 29, 30, 39

GANNON, T., MADNICK, S.E., MOULTON, A., SIEGEL, M., SABBOUH, M. & ZHU, H. (2009). Framework for the analysis of the adaptability, extensibility, and scalability of semantic information integration and the context mediation approach. In *HICSS*, 1–11. 179, 184

GIL, A. & RATNAKAR, V. (2002). A comparison of (semantic) markup languages. In *In Proceedings of the 15th International FLAIRS Conference*, 413–418. 92

GLIMM, B., HORROCKS, I., LUTZ, C. & SATTLER, U. (2007). Conjunctive query answering for the description logic shiq. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*. 29, 35

GOH, C.H. (1997). Context interchange: New features and formalisms for the intelligent integration of information. *ACM Transactions on Information Systems*, **17**, 270–293. 32

GOH, C.H., MADNICK, S.E. & SIEGEL, M.D. (1994). Context interchange: overcoming the challenges of large-scale interoperable database systems in a dynamic environment. In *CIKM '94: Proceedings of the third international conference on Information and knowledge management*, 337–346, ACM, New York, NY, USA. 44

GRAHNE, G. & MENDELZON, A.O. (1999). Tableau techniques for querying information sources through global schemas. In *In Proc. of the 7th Int. Conf. on Database Theory (ICDT'99), volume 1540 of Lecture Notes in Computer Science*, 332–347, Springer. 48

GRATZER, M. & WINIWARTER, W. (2003). Challenges of the internet for the sme hotel sector in austria. *The Fifth International Conference on Information Integration and Web-based Applications Services*. 7

HALEVY, A., IVES, Z., MORK, P. & TATARINOV, I. (2003). Piazza: Data management infrastructure for semantic web applications. In *Proceedings of the 12th international conference on World Wide Web, 2003*, 556–567. 43

HALEVY, A.Y. (2000). Answering queries using views: A survey. **10**, 270–294. 49

HALPIN, T. & HALPIN, D.T. (1999). Data modeling in uml and orm revisited. In *Proc. EMMSAD99: 4th IFIP WG8.1 Int. Workshop on Evaluation of Modeling Methods in Systems Analysis and Design*. 74

HEFLIN, J. & HENDLER, J. (2000). Semantic interoperability on the web. *Extreme Markup Languages*. 9

Hevner, A.R., March, S.T., Park, J. & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, **28**. 15, 161

Hitz, S.M.M.J., M. (2006). Information and communication technologies in tourism 2006. 162

Hoare, C.A.R. (1962). Quicksort. *The Computer Journal*, 10–15. 110, 115

Inkpen, G. (1998). *Information Technology for Travel and Tourism*. Addison Wesley, Edinburgh. 4

Jiang, J.J. & Conrath, D.W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *Proceedings of International Conference Research on Computational Linguistics*. 110, 112, 115

Kalinichenko, L., Missikoff, M. & Skvortsov, N. (2003). Ontological modeling. In *Proceeding of the 5 th Russian Conference on Digital Libraries, St.-Petersburg*. 31

Karvounarakis, G., Alexaki, S., Christophides, V., Plexousakis, D. & Scholl, M. (2002). Rql: a declarative query language for rdf. In *WWW*, 592–603. 139

Kim, H.M. (2000). Developing ontologies to enable knowledge management: Integrating business process and data driven approaches. In *AAAI Workshop on Bringing Knowledge to Business Processes*. 72

Kirk, T., Levy, A.Y., Sagiv, Y. & Srivastava, D. (1995). The information manifold. In *In Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments*, 85–91. 11, 30, 36, 39, 150

KLEIN, M., FENSEL, D., HARMELEN, F.V. & HORROCKS, I. (2000). The relation between ontologies and xml schemas. In *XML Schemas, Link oping Electronic Articles in Computer and Information Science*. 92

LANGEGGER, A., WOB, W. & BLOCHL, M. (2008). A semantic web middleware for virtual data integration on the web. *Proceedings of the 5th European Semantic Web Conference (ESWC)*. 55, 56

LECLUSE, C., RICHARD, P. & VELEZ, F. (1988). O2, an object-oriented data model. In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 424–433, ACM, New York, NY, USA. 73

LEHTI, P. & FANKHAUSER, P. (2004). Xml data integration with owl: Experiences and challenges. *Applications and the Internet, IEEE/IPSJ International Symposium on*, **0**, 160. 93, 94, 95

LEVY, A., RAJARAMAN, A. & ORDILLE, J. (1996). Querying heterogeneous information sources using source descriptions. 251–262. 11, 48

LEVY, A.Y., MENDELZON, A.O., SAGIV, Y. & SRIVASTAVA, D. (1995). Answering queries using views. 40

LIANG, P., LIU, Y., LIANG, S. & CHEN, H. (2008). Semantic query for integrated heterogeneous database systems. In *First International Conference on Intelligent Networks and Intelligent Systems*, 741–744. 21, 28, 29, 30, 35, 36

LU, J. (2003). Content-based retrieval in hybrid peer-to-peer networks. In *In Proceedings of the ACM Conference on Information and Knowledge Management*, 199–206, ACM Press. 11, 42

MANOLA, F. & MILLER, E., eds. (2004). *RDF Primer*. W3C Recommendation, World Wide Web Consortium. 75, 139

MANOLESCU, I., KOSSMANN, D., FLORESCU, D., KOSSMANN, D., XHUMARI, F. & OLTEANU, D. (2000). Agora: Living with xml and relational. In *In Proceedings of International Conference on Very Large Databases (VLDB*, 623–626, Morgan Kaufmann. 29, 30, 38, 74

MAO, M., PENG, Y. & SPRING, M. (2008). Ontology mapping: As a binary classification problem. *Proceedings of the 4th International Conference on Semantics, Knowledge and Grid*. 117

MARCH, L.X. (2001). A dynamic warehouse for xml data of the web. *IEEE Data Engineering Bulletin*, **24**, 40–47. 23, 25

MARINI, J. (2002). *Document Object Model*. McGraw-Hill, Inc., New York, NY, USA. 25

MCGUINNESS, D.L. & VAN HARMELEN, F. (2003). Owl web ontology language overview. 75, 109

MENA, E., KASHYAP, V., SHETH, A. & ILLARRAMENDI, A. (1996). Observer: An approach for query processing in global information systems based on interoperation across pre-existing ontologies. 14–25, IEEE Computer Society Press. 11

MILLER, G.A. (1995). Wordnet: a lexical database for english. *ACM*. 112

MITRA, P., WIEDERHOLD, G. & KERSTEN, M. (2000). A graph-oriented model for articulation of ontology interdependencies. *Proceedings of Conference on Extending Database Technology*, 86–100. 48

MONDECA (2004). Semantic web methodologies and tools for intra-european sustainable tourism. *White Paper*. 53, 54

Ng, W.S., Ooi, B.C., Tan, K.L. & Zhou, A. (2004). Peerdb: a p2p-based system for distributed data sharing. In *Data Engineering, 2003. Proceedings. 19th International Conference on*, 633–644. 11, 28, 29, 42, 43

Noy, N. & Musen, M. (2003). The prompt suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies*, 983–1024. 107

Noy, N.F. & Musen, M.A. (2002). Evaluating ontology-mapping tools: Requirements and experience. In *In Proceedings of OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management*, 1–14. 180

Ou, S., Pekar, V., Orasan, C., Spurk, C. & Negri, M. (2008). Development and alignment of a domain specific ontology for question answering, in european language resources association (elra). *Proceedings of the Sixth International Language Resources and Evaluation*. 53, 54

Owens, S., Reppy, J. & Turon, A. (2009). Regular-expression derivatives re-examined. *Journal of Functional Programming*, **19**, 173–190. 133

Papakonstantinou, Y. & Vassalos, V. (2006). Query rewriting for semistructured data. 38

Paraskevas, A. (2005). The impact of technological innovation in managing global value chains in the tourism industry. *OECD Korea Conference on Global Tourism Growth: A Challenge for SMEs*. 6

Pottinger, R. & Levy, A. (2000). A scalable algorithm for answering queries using views. In *In Proceedings of VLDB*, 484–495. 49

Prasad, S.T.S. & Rajaraman, A. (1998). Virtual database technology, xml, and the evolution of the web. *IEEE Data Engineering Bull.*, **21**, 48–52. 23, 25

Prud'hommeaux, E. & Seaborne, A. (2008). Sparql query language for rdf. 139

Qg, C.C. (1997). First order logic in practice. In *In Proc. FTP*. 74

Qian, X. (1996). Query folding. In *In Proceedings of the International Conference on Data Engineering*, 48–55. 48

Quilitz, B. & Leser, U. (2008). Querying distributed rdf data sources with sparql. *Proceedings of the 5th European Semantic Web Conference (ESWC)*. 55

Resnik, P. (1992). Wordnet and distributional analysis: A class-based approach to lexical discovery. *Proceedings of the AAAI Symposium on Probabilistic Approaches to Natural Language*. 114

Rodrigues, R.P., T. & Cardoso, J. (2006). Mapping xml to existing owl ontologies. 48, 93, 96, 99

Sheth, A.P. & Larson, J.A. (1990). Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, **22**, 183–236. 29

Shi, Y., Liu, X., Xu, Y. & Ji, Z. (2010). Semantic-based data integration model applied to heterogeneous medical information system. In *The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 624–628. 29, 30, 35, 36

SUSSNA, M. (1993). Word sense disambiguation for free-text indexing using a massive semantic network. *Proceedings of the Second International Conference on Information and Knowledge Management*. 110, 112

TATARINOV, I. & HALEVY, A. (2004). Efficient query reformulation in peer data management systems. 49

TEMPICH, C., STAAB, S. & WRANIK, A. (2004). Remindin': Semantic query routing in peer-to-peer networks based on social metaphors. In *the 13th international conference on World Wide Web*, New York, NY, USA. 47

WACHE, H., VÖGELE, T., VISSER, U., STUCKENSCHMIDT, H., SCHUSTER, G., NEUMANN, H. & HÜBNER, S. (2001). Ontology-based integration of information - a survey of existing approaches. 72

WEINSTEIN, P.C. & BIRMINGHAM, W.P. (1999). Comparing concepts in differentiated ontologies. In *Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW'99*. 156

WERTHNER, H. & KLEIN, S. (1999). Information technology and tourism - a challenging relationship. 2, 199

WORBOYS, H.H., M & MAGUIRE, D. (1990). Object-oriented data modelling for spatial databases. *International Journal of Geographic Information Systems*, **4**. 73

WROE, C., BECHHOFER, S., LORD, P., RECTOR, A., GOBLE, C., STEVENS, R., STEVENS, R. & OF, D. (2003). Building ontologies in daml+ oil. 75, 109

XU, L. & EMBLEY, D.W. (2010). Combining the best of global-as-view and local-as-view for data integration. In *In Information Systems Technologies and Its Applications - ISTA*, 123–136. 30, 41

Yang, K. & Steele, R. (2009). Ontology mapping based on concept classification. *In proceedings of the DEST conference 2009*. 14

Yang, K., Steele, R. & Lo, A. (2007). An ontology for xml schema to ontology mapping representation. In *iiWAS*, 101–111. 14, 94

Yang, K., Lo, A. & Steele, R. (2008). Ontology mediated xml data translation. *IJWIS*, **4**, 181–197. 14

Yang, L. & Szpakowicz, S. (1994). Path-finding in networks. In *Proc SEAR CC '90 South East Asia Regional Computer Confederation Conf*. 147

Zhang, L. & Gu, J.G. (2005). Ontology based semantic mapping architecture. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, vol. 4, 2200–2205. 93, 96

Zhou, G., Hull, R. & King, R. (1995a). Squirrel phase 1: Generating data integration mediators that use materialization. 23

Zhou, G., Hull, R., King, R. & Franchitti, J. (1995b). Supporting data integration and warehousing using h2o. *Data Engineering*, 29. 11, 23