NeOn: Lifecycle Support for Networked Ontologies

Integrated Project (IST-2005-027595)

Priority: IST-2004-2.4.7 — "Semantic-based knowledge and content systems"

D1.4.2 Metadata Management and Reasoning with Networked Ontologies in Distributed Environment

Deliverable Co-ordinator: **Yimin Wang**

Deliverable Co-ordinating Institution: **Universität Karlsruhe – TH (UKARL)**

Other Authors: **Raúl Palma (Universidad Polit écnica di Madrid – UPM)**

This deliverable provides an introduction to development of tool support on database integration and semantic query answering over autonomous, heterogenous data sources in NeOn project. We present novel applications called NeOnDBMap and NeOnQA to enable distributed database integration and query answering by creating the mappings between database schemata and ontologies. We also evaluate our approach against case study data from NeOn project and find it promising in real life scenario.

Document Identifier:	NEON/2008/D1.4.2/v1.0	Date due:	February 28, 2008
Class Deliverable:	NEON EU-IST-2005-027595	Submission date:	February 28, 2008
Project start date	March 1, 2006	Version:	v1.0
Project duration:	4 years	State:	Final
		Distribution:	Public

NeOn Consortium

This document is part of the NeOn research project funded by the IST Programme of the Commission of the European Communities by the grant number IST-2005-027595. The following partners are involved in the project:

Open University (OU) – Coordinator Knowledge Media Institute – KMi Berrill Building, Walton Hall Milton Keynes, MK7 6AA United Kingdom Contact person: Martin Dzbor, Enrico Motta E-mail address: {m.dzbor, e.motta}@open.ac.uk	**Universität Karlsruhe – TH (UKARL)** Institut für Angewandte Informatik und Formale Beschreibungsverfahren – AIFB Englerstrasse 11 D-76128 Karlsruhe, Germany Contact person: Peter Haase E-mail address: pha@aifb.uni-karlsruhe.de
Universidad Politécnica de Madrid (UPM) Campus de Montegancedo 28660 Boadilla del Monte Spain Contact person: Asunción Gómez Pérez E-mail address: asun@fi.ump.es	**Software AG (SAG)** Uhlandstrasse 12 64297 Darmstadt Germany Contact person: Walter Waterfeld E-mail address: walter.waterfeld@softwareag.com
Intelligent Software Components S.A. (ISOCO) Calle de Pedro de Valdivia 10 28006 Madrid Spain Contact person: Jesús Contreras E-mail address: jcontreras@isoco.com	**Institut 'Jožef Stefan' (JSI)** Jamova 39 SL–1000 Ljubljana Slovenia Contact person: Marko Grobelnik E-mail address: marko.grobelnik@ijs.si
Institut National de Recherche en Informatique et en Automatique (INRIA) ZIRST – 665 avenue de l'Europe Montbonnot Saint Martin 38334 Saint-Ismier, France Contact person: Jérôme Euzenat E-mail address: jerome.euzenat@inrialpes.fr	**University of Sheffield (USFD)** Dept. of Computer Science Regent Court 211 Portobello street S14DP Sheffield, United Kingdom Contact person: Hamish Cunningham E-mail address: hamish@dcs.shef.ac.uk
Universität Kolenz-Landau (UKO-LD) Universitätsstrasse 1 56070 Koblenz Germany Contact person: Steffen Staab E-mail address: staab@uni-koblenz.de	**Consiglio Nazionale delle Ricerche (CNR)** Institute of cognitive sciences and technologies Via S. Marino della Battaglia 44 – 00185 Roma-Lazio Italy Contact person: Aldo Gangemi E-mail address: aldo.gangemi@istc.cnr.it
Ontoprise GmbH. (ONTO) Amalienbadstr. 36 (Raumfabrik 29) 76227 Karlsruhe Germany Contact person: Jürgen Angele E-mail address: angele@ontoprise.de	**Food and Agriculture Organization of the United Nations (FAO)** Viale delle Terme di Caracalla 00100 Rome Italy Contact person: Marta Iglesias E-mail address: marta.iglesias@fao.org
Atos Origin S.A. (ATOS) Calle de Albarracín, 25 28037 Madrid Spain Contact person: Tomás Pariente Lobo E-mail address: tomas.parientelobo@atosorigin.com	**Laboratorios KIN, S.A. (KIN)** C/Ciudad de Granada, 123 08018 Barcelona Spain Contact person: Antonio López E-mail address: alopez@kin.es

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed writing parts of this document:

- Universität Karlsruhe – TH (UKARL)
- Universidad Politécnica di Madrid (UPM)

Change Log

Version	Date	Amended by	Changes
0.1	20-10-2007	Yimin Wang	Creation
0.2	07-12-2007	Yimin Wang	Most contents
0.3	18-12-2007	Yimin Wang	Evaluation and Conclusion
0.4	18-01-2008	Yimin Wang	Mapping
0.5	26-01-2008	Raul Palma	Content Checking
0.6	31-01-2008	Yimin Wang	Revising
0.7	04-02-2008	Yimin Wang	Ready for review
1.0	28-02-2008	Yimin Wang	Final Version after QA

Executive Summary

This deliverable is part of the work in Workpackage 1 “Dynamics of Networked Ontologies” of the NeOn project. Following the work in Task 1.4, we extend our work in D1.4.1 to develop an integrated approach for the managing semantic data, networked ontologies and related metadata in the distributed scenario. For this individual part of Workpackage 1, we develop new prototypes that handles complex relationships between distributed semantic data, including ontologies and relational databases.

In NeOn project, an obvious example is FAO, which often have many departments, maintaining distributed data that are reasonably interconnected. In order to take advantage of semantic technologies, FAO people have been trying to populate ontologies from databases but they find it is quite inefficient to directly query large ontologies that represent as data from database.

As we focus on convenient maintenance and efficient processing of complicated interactions between databases across physical and organizational boundary, it’s possible to use networked ontologies to integrate distributed databases, realizing query answering over distributed and interconnected databases.

In this deliverable, we take advantage of recent development of networked ontologies model, introducing a novel approach to integrate distributed databases using networked ontologies. Compare with our previous work in D1.4.1, we have several advantageous:

- We extend our mappings from ontology-ontology mapping to ontology-database schema mapping;
- the new developed NeOnQA is now able to query database using semantic queries;
- we implement a tool to link ontology with database by creating ontology-database mapping, called NeOnDBMap.

NeOnQA relies on the decentralized network infrastructure, which exploits diverse connectivity between participants in a network and the cumulative bandwidth of network participants. To evaluate our approach, we compare our system with previous decentralized ontology query answering system – KOANp2p to see the potential advances.

Contents

List of Figures

Chapter 1

Introduction

In this chapter, we discuss the scope of this deliverable, the motivation of our work, the state-of-the-art research topics that are related to this deliverable, and how this deliverable is organized.

1.1 Motivation

Nowadays, ontologies are increasingly applied as backbones for the next generation information systems. Many efforts have been made in both theoretical approaches [CGLR04, WVV$^+$01] and real-life applications [ABM05, DL06, CWW$^+$06] in integrating databases using ontologies. People also have defined particular semantics to process distributed ontologies that have mappings between each other [SBT05] and investigated query answering over distributed data based on simple ontologies [GR03]. However, we find it difficult to directly implement these approaches to our work in NeOn project. The major challenges to direct apply these existing approaches can be typically concluded as following:

- Less work considers a decentralized and distributed scenario.
- When people consider interconnections between data sources, they normally don't consider the dynamics of these data sources. For example, if the schema of a data source changes, the interconnection might collapse as the other data sources may not be able to recognize the changed schema.
- The developed tools normally either don't support the above mentioned two points or have not been widely deployed within real-life scenario.

Big organizations, such as Food and Agricultural Organization of United Nations (FAO) often have many departments, maintaining distributed data that are reasonably interconnected. For example, KCEW, FIES and GILW (different FAO departments) are now collaboratively developing semantic applications in the fishery domain within NeOn project and they are sharing fishery data across departments which may locate around the world. In order to take advantage of semantic technologies, FAO people have been trying to populate ontologies from databases but they find it is quite inefficient to directly query large ontologies. Therefore, they call for an approach to execute semantic query over the interconnected distributed databases in an integrated manner.

1.2 Solution and Contribution

As the next generation ontology model, the networked ontology model [HRW$^+$06] aims to provide enhanced functionalities to handle dynamic and heterogeneous ontologies that are interconnected within a networking scenario. We realize that managing heterogeneous, interconnected database has similar requirements and scope for handling data in a distributed scenario. Also, as we will discuss in the next subsection, there have been many efforts in integrating real life databases using ontologies[RGP06, ABM05, DL06]. Therefore,

it's possible to use networked ontologies to integrate distributed databases, realizing query answering over distributed and interconnected databases. The potential benefit here is mainly contributed by convenient maintenance and efficient processing of complicated interactions between databases across physical and organizational boundary.

In this deliverable, we take advantage of recent development of networked ontologies model, introducing a novel approach to integrated distributed databases using networked ontologies.

As our central contribution, we implement this approach by developing real life applications called NeOnDBMap and NeOnQA to help integrating and querying distributed databases, respectively.

1. NeOnDBMap establishes the mapping between ontology and database schema and also supports creating mappings between ontologies. The ontologies and databases can be distributed. NeOnDBMap creates ontology-databse schema mapping by lifting database schema into an ontology that can be processed by reasoners, such as KAON2.
2. NeOnQA integrates the ontologies and mappings that are distributed and interconnected. NeOnQA uses Oyster ere as metadata registry to identify the remote ontology resource and propagate local resource.

We also evaluate our approach against FAO data. The evaluation results of our approach show that the performance and usability of our actual system are satisfactory in the real life scenario:

1. The query answering results show the completeness of our approach of bridging database and ontology that are distributed.
2. This approach is able to scale in a decentralized network.
3. The performance of NeOnQA system is comparatively better compare to previous systems, such as KAONp2p, in many circumstances.

1.3 State-of-the-art

There are two major aspects of the relate work: one falls into the semantic web context, and the other is related to classical data integration.

1.3.1 Semantic Web

Integrating relational databases and structured data has been always a hot topic in the Semantic Web community. There have been many efforts in developing applications to support querying semantic data by using semantic web ontologies.

D2R Server[1] uses the D2RQ mapping language to capture mappings between relational database schemata and OWL/RDFS ontologies. The central object in D2RQ is the *ClassMap* which represents a mapping from a set of entities described within the database, to a class or a group of similar classes of resources. Each *ClassMap* has a set of property bridges, which specify the mapping from relational table column to class property. D2R Server allows applications to query RDB using the SPARQL via a query rewriting approach. Similar mapping mechanism and rewriting approach are employed in the SquirrelRDF project[2], RDF Gateway[3]. Virtuoso[4] recently has released a declarative meta schema language for mapping SQL data to RDF ontologies. R_2O [RGP06] is a Relational-to-OWL mapping language that provides an extensible set of primitives with well-defined semantics.

[1] http://sites.wiwiss.fu-berlin.de/suhl/bizer/d2r-server/
[2] http://jena.sourceforge.net/SquirrelRDF
[3] http://www.intellidimension.com
[4] http://virtuoso.openlinksw.com

An and colleagues [ABM05] present a tool which could automatically infer the Local-as-View (LaV) mapping formulas from simple predicate correspondences between relational schema and formal ontologies. Complete automatic approach to define semantic mapping is difficult, however, one enhancement could be candidate mappings suggested automatically to assist and facilitate users in creating mappings between schema and ontologies.

Dejing Dou and colleagues [DL06] propose an ontology-based framework called OntoGrate for relational database integration. The mappings are defined by using bridge-axioms, and the queries are described by a language called WEB-PDDL which will be rewritten into SQL queries for data retrieval. Piazza [HIM$^+$04] is a P2P-based data integration system with consideration of semantic web vision. The current system is implemented with the XML data model for its mapping language and query answering. Francois [GR03] considers theoretic aspects of answering query using views for semantic web, mainly focusing on description logic formalism.[DL06] proposes a view based approach to query traditional Chinese medical data by integrating relational databases using ontologies. A series of tools are provided to create and manage RDF ontology – database schemata mappings, to support search and query functionalities.

In NeOn project, the semantic query over relational database is supported in F-logic semantics. However, it is not feasible to integrate F-logic ontologies with OWL ontologies to establish an integrated query scheme. As we adopt “dual language approach” in NeOn project, the OWL ontology schema support is still missing for querying relational databases.

Our approach proposes distributed database integration by using networked ontologies, that are not considered in previous approaches. We take the advantage of networked ontology model that are potentially advancing in handling dynamic and interconnect semantic data. We believe this networked ontology model can also be well applied in managing distributed databases.

1.3.2 Classical Database Integration

Closely relevant areas from classical AI and database communities are commonly referred as *logic-based data integration* [CG05] and *ontology-based data integration* [WVV$^+$01] [CGL$^+$04]. A popular approach is to take the advantageous of description logics formalism to define the global ontology to mediate a set of heterogeneous data sources [CGL$^+$04]. Calvanese and colleagues [CG05] propose a specific DL language called $\mathcal{ALCQI}$ for ontology-based database integration. The conjunctive query answering in $\mathcal{ALCQI}$-mediated integration system is decidable. They also propose the DL-Lite, a specifically tailored restriction of $\mathcal{ALCQI}$ that not only ensures tractability of query answering, but also keeping enough expressive power to capture the fundamental aspects of conceptual data models.

Within the traditional database community, a substantial number of works have been done in data integration over past couple of decades [HRO06]. Among them, one typical approach to query mediation is referred as *answering or rewriting queries using views*[Hal01]. Most previous works has been focused on the relational databases and XML data. For example, several general query rewriting algorithms, the *bucket algorithm* [LRO96], the *inver-rule algorithm* [Qia96] and a more scalable rewriting algorithm called MiniCon [PH01], have been proposed for rewriting queries using views.

Our work focus on applying the above mentioned theoretical advances, borrowing the idea of ontology integration for database integration, enhancing our system with up-to-date theoretical approaches, that are typically not implemented in a real life application for actual users.

1.4 Overview of the Deliverable

In the following, we first discuss the overall updated integrated architecture for T1.4 in Chapter 2. Then in Chapter 3, we present foundations of our approach for query answering, discovering and integrating distributed databases based on networked ontologies. Afterwards, we describe the overview of implementation and corresponding applications for NeOnDBMap and NeOnQA in Chapter 4 and 5, respectively. We also

evaluate of our approach against real life data to see the applicability and scalability of our system. Finally, we conclude the major contributions of this deliverable and discuss possible future extensions in Chapter 6.

Chapter 2

Overall Architecture

2.1 A Decentralized Structure in Distributed Environment

Traditional Communication topologies include the classic Client-Server mode, which is consisted of a server with high computational capability, providing arbitrary services; and many clients with comparatively low computational capability maintain the connection to the server and consume its services for the end-user. Although such a paradigm holds the great advantages: (1) The distinguished and unambiguous responsibilities between client and server; (2) and centralized data storage with high security level for administration, it has still problems in: (1) The network traffic congestion, once the number of simultaneous client requests to a given server increases. and (2) lack of the system robustness, since a critical server fail will cause all the clients' requests not to be fulfilled.

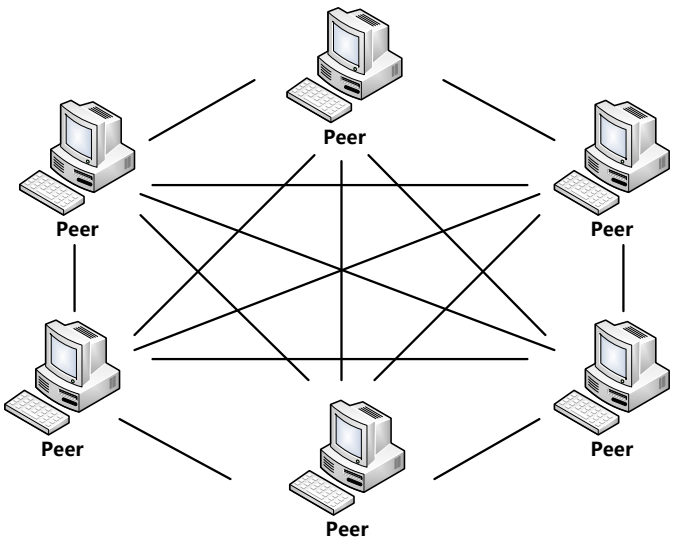

Figure 2.1: The decentralized network infrastructure of NeOnQA

Thus, NeOnQA relies on the decentralized network infrastructure (Figure 2.1), which exploits diverse connectivity between participants in a network and the cumulative bandwidth of network participants. In detail, each peer in the P2P network infrastructure holds a substantial number of ontologies as the local resource. On the one hand, the peer will publish its local resource so that other can discover and access at the moment of it starts. On the other hand, each peer can also get the information about the resources residing at other peers, for either building the integration system or performing query tasks over the integrated system. When

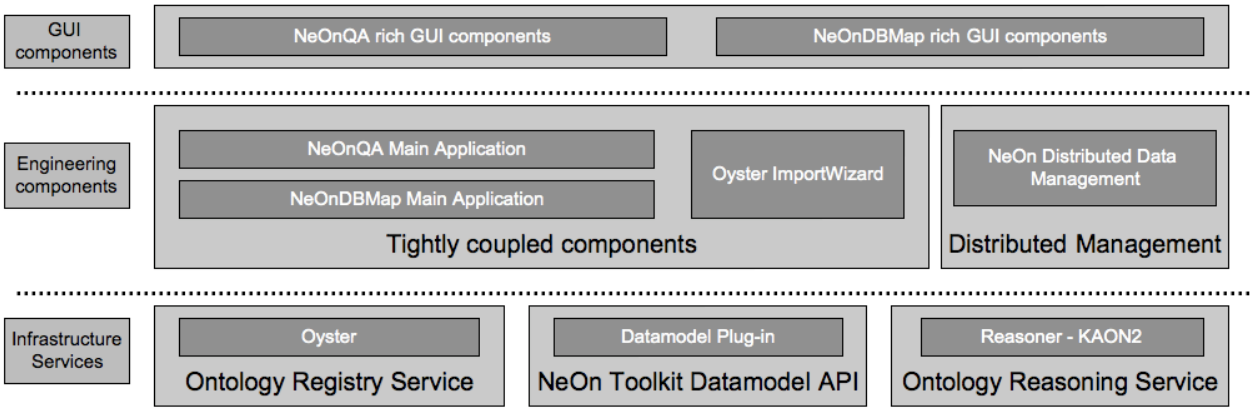

Figure 2.2: The updated integrated architecture for metadata management and query answering over distributed semantic data.

some peer exits, the robustness of the infrastructure makes the rest still work correctly.

In addition, some prerequisites for designing NeOnQA are necessary to be clarified and coincided: (1) As several languages exist for ontology modeling, NeOnQA supports only the OWL-DL ontology, since it employs KAON2 as its underlying infrastructure for ontology manipulation. (2) The URI of the ontology can be used for its identification and we identify uniquely an ontology by its URI and the peer's IP, in which this ontology locates. In other words, it is allowed that *homogenous ontologies* (the ones that has the same URI) can exist on different peers, but not on the same peer. (3) The metadata of a certain ontology in NeOnQA refers only to the *related ontologies* that have mappings to the this certain ontology.

Please note: In this deliverable, peer also refers to distributed nodes.

2.2 Architecture of Components

Here we introduce the overall architecture for representing the positions of plug-ins that are developed along with this deliverable (Figure 2.2).

- There are two rich GUI components for NeOnQA and NeOnDBMap, respectively, while they are have their main applications running at the background.
- We use Oyster as both web service provider and ontology registry to manage local ontology repository and propagate ontologies via internet.
- We apply Datamodel plug-in in Core NeOn Toolkit to provide API for processing ontologies and KAON2 as central engine for query answering over semantic data.
- All applications are centrally controlled by a distributed data management component in background. This component controls the threads in data communication between multiple applications via Java Sockets, web service and RMI connections.

Chapter 3

Foundations and Approaches

In this chapter, we introduce some foundations for networked ontology management. As we assume readers of this deliverable are familiar with OWL DL syntax and semantics, we introduce basis of conjunctive query answering over OWL DL ontologies and mapping systems that cover database-ontology mapping. We also discuss the distributed metadata that are used in this deliverable for discovering distributed resources.

3.1 Conjunctive Query Answering

In this deliverable, we assume readers are familiar with OWL DL syntax and semantics [BCM+03]. Let *KB* be a OWL knowledge base, N_P be a set of names such that all concepts and roles are in N_P. An *atom* $P(s_1, ..., s_n)$ has the form $P(s_1, \ldots, s_n)$, denoted as $P(\mathbf{s})$, where $P \in N_P$, and s_i are either variables or individuals from *KB*. An atom is called a *DL-atom* if P is a OWL-concept or role; it is called *non-DL-atom* otherwise.

Definition 1 (Conjunctive Queries) *Let* $x_1, \ldots, x_n$ *and* $y_1, \ldots, y_m$ *be sets of* distinguished *and* non-distinguished *variables, denoted as* $\mathbf{x}$ *and* $\mathbf{y}$*, respectively. A* conjunctive query $Q(\mathbf{x}, \mathbf{y})$ *over a KB is a conjunction of atoms* $\bigwedge P_i(\mathbf{s_i})$*, where the variables in* $\mathbf{s_i}$ *are contained in either* $\mathbf{x}$ *or* $\mathbf{y}$*. We denote operator* π *[MSS04] to translate* $Q(\mathbf{x}, \mathbf{y})$ *into a first-order formula with free variables* $\mathbf{x}$*:* $\pi(Q(\mathbf{x}, \mathbf{y}))=\exists \mathbf{y}: \bigwedge(P_i(\mathbf{s_i}))$*.*

For $Q_1(\mathbf{x}, \mathbf{y_1})$ and $Q_2(\mathbf{x}, \mathbf{y_2})$ conjunctive queries, a *query containment* axiom $Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1})$ has the following semantics:

$$\pi(Q_2(\mathbf{x}, \mathbf{y_2}) \sqsubseteq Q_1(\mathbf{x}, \mathbf{y_1})) = \forall \mathbf{x} : \pi(Q_1(\mathbf{x}, \mathbf{y_1})) \leftarrow \pi(Q_2(\mathbf{x}, \mathbf{y_2}))$$

Definition 2 (Conjunctive Query Answering) *An* answer *of a conjunctive query* $Q(\mathbf{x}, \mathbf{y})$ *w.r.t. KB is an assignment* θ *of individuals to distinguished variables, using* $\mathsf{Ans}(Q, KB)$ *as a function, such that* $\pi(KB) \models \pi(Q(\mathbf{x}\theta, \mathbf{y}))$*.*

We refer readers to [MSS04, HT00] for further issues in conjunctive query answering for ontologies.

We follow the general framework of [Len02] to formalize the notion of a mapping system for DL ontologies, where mappings are expressed as correspondences between conjunctive queries[1] over ontologies.

3.2 Mapping Systems for Integration

Since we focus on querying heterogeneous databases in an integrated manner, after selecting the target database, there are two major steps: (1) Identifying how this target database relates to other databases on different distributed nodes; (2) integrating the related databases with target database.

[1] We denote a conjunctive query as $q(\mathbf{x}, \mathbf{y})$, with $\mathbf{x}$ and $\mathbf{y}$ sets of *distinguished* and *non-distinguished* variables, respectively.

Normally, mappings are used to represent the relationships between database schema. In our approach, we do not directly process the database schema mappings. Instead, we lift database schema into ontology and represent the database schema mappings using ontology mappings. Managing ontology mapping and monitor the dynamics of ontologies connected by mappings are central concerns of networked ontology research [HRW+06].

In NeOnQA, on the one hand, we query and integrate heterogeneous ontologies that are distributed with mappings similar to our previous work in [WHP07, HW07]. On the other hand, we use the approach introduced by Motik and colleagues [MHS07] to process the integration of local OWL DL ontologies and relational databases. Based on these works, we develop a distributed database integration system (Figure 3.1) to support querying distributed databases using ontologies.

Let's first look at the definition of an OWL DL ontology mapping system [HW07]: We follow the framework of [Len02] to formalize the notion of an OWL DL ontology mapping system, where mappings are represented as correspondences between conjunctive queries over ontologies.[2]

Definition 3 (Mapping System) *An mapping system $\mathcal{MS}$ is a triple $(\mathcal{S}, \mathcal{T}, \mathcal{M})$, where*

- *$\mathcal{S}$ is the* source ontology*, $\mathcal{T}$ is the* target ontology*,*
- *$\mathcal{M}$ is the mapping between $\mathcal{S}$ and $\mathcal{T}$, i.e. a set of assertions $q_S \rightsquigarrow q_T$, where q_S and q_T are conjunctive queries over $\mathcal{S}$ and $\mathcal{T}$, respectively, with the same set of distinguished variables $\mathbf{x}$, and $\rightsquigarrow \in \{\sqsubseteq, \sqsupseteq, \equiv\}$.*

An assertion $q_S \sqsubseteq q_T$ is called a sound *mapping, requiring that q_S is contained by q_T w.r.t. $S \cup \mathcal{T}$; an assertion $q_S \sqsupseteq q_T$ is called a* complete *mapping, requiring that q_T is contained by q_S w.r.t. $\mathcal{S} \cup \mathcal{T}$; and an assertion $q_S \equiv q_T$ is called an* exact *mapping, requiring it to be sound and complete.*

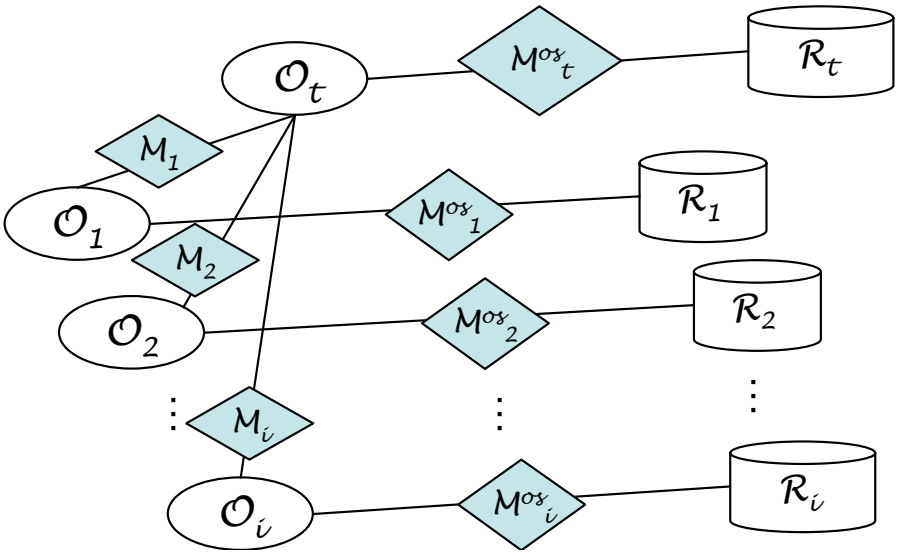

Figure 3.1: Distributed databases integration system.

In [HM] the semantics of the mapping system has been defined by translation into first-order logic. The intuitions behind the semantics of the main inference task for $\mathcal{MS}$, i.e. computing answers for a conjunctive query $Q(\mathbf{x}, \mathbf{y})$ w.r.t. $\mathcal{MS}$ have been discussed in [HW07]. We can see from Figure 3.1, ontologies that are mapped to database schemata forms an OWL DL ontology mapping system which enables query answering over distributed ontologies.

We follow the work in [MHS07] to define the mappings between OWL DL ontologies and database schema.

[2]We denote a conjunctive query as $q(\mathbf{x}, \mathbf{y})$, with $\mathbf{x}$ and $\mathbf{y}$ sets of *distinguished* and *non-distinguished* variables, respectively.

Definition 4 (Ontology-Schema Mapping) *An ontology-schema mapping system $\mathcal{MS}^{os}$ is a triple $(\mathcal{O}, DB, \mathcal{M}^{os})$, where*

- *$\mathcal{O}$ is the* source ontology*, $\mathcal{R}$ is the* target schema of *of a relational database,*
- *$\mathcal{M}^{os}$ is the mapping between $\mathcal{O}$ and $\mathcal{R}$, i.e. a set of assertions $q_S \rightsquigarrow q_T$, where q_S and q_T are conjunctive queries over $\mathcal{O}$ and $\mathcal{R}$, respectively, with the same set of distinguished variables $\mathbf{x}$, and $\rightsquigarrow \in \{\sqsubseteq, \sqsupseteq, \equiv\}$.*

In Figure 3.1, on each distributed node, the semantic query are interpreted to retrieve the answers from databases. We refer readers to [MHS07] for the technical details of this interpretation.

3.3 Distributed Database Integration System

Then we define the distributed database integration system. In the following, we denote $I = \{1, \ldots, n\}, n \in \mathbb{N}$ and $i \neq j; i, j \in I$.

Definition 5 (Distributed Database Integration System) *A distributed database integration system is a triple $(\{\mathcal{MS}_i\}, \{\mathcal{MS}_i^{os}\}, \{DB_i\})$, where*

1. *$\{\mathcal{MS}_i\}$ is a set of OWL DL ontology mapping systems which involves a set of ontologies $\{\mathcal{O}_i\}$ with a single target ontology $\mathcal{O}_t$;*
2. *$\{\mathcal{MS}_i^{os}\}$ is set of ontology-schema mapping systems which involves a set of ontologies $\{\mathcal{O}_i\}$ and database schemata $\{\mathcal{R}_i\}$;*
3. *$\{DB_i\}$ is set of databases with schemata $\{\mathcal{R}_i\}$ and target database DB_t.*

Let $Q(\mathbf{x}, \mathbf{y})$ be a conjunctive query over database DB_t. The query answering for DB_t is to compute answers of conjunctive query $Q(\mathbf{x}, \mathbf{y})$ over $\mathcal{O}_t$.

Based on the foundations discussed above, we can implement NeOnQA system. The developed applications will be introduced in Section 3.2 and evaluated in Section 5.3

3.4 Metadata for Distributed Ontologies

In NeOnQA, we follow the successful approach of expertise-based node selection [HSvH04], which has already been applied in the node-to-node systems Bibster [HBE+04] and Oyster [PH05]. In this approach, nodes advertise their resource descriptions according to the metadata ontology in the network to form acquaintances, whereby the nodes are fully autonomous in choosing their acquaintances. Moreover, we assume there is no global control in the form of a global registry to manage acquaintances. Acquaintances are managed in a decentralized manner, i.e. by the individual node using its metadata registry. Here we briefly introduce the metadata to describe distributed nodes and mappings between ontologies.

For the description of ontology metadata we rely on OMV, the Ontology Metadata Vocabulary [HSH+05]. The extensions required to model metadata of nodes are realized as an extension to the OMV ontology, called P-OMV. Figure 3.2 shows an overview of the P-OMV ontology. [3]

Each NeOnQA node carries a unique ID (`UID`) to be identified. We simply use IP addresses in NeOnQA. In addition to the unique identifier, each node carries a `name` for identification, which is primarily used for human interpretation. The `expertise` is an abstract description of the node in terms of some topic ontology. The property `acquaintedWith` describes the acquaintances of a node with other nodes. The node-to-node

[3] `http://omv.ontoware.org/`

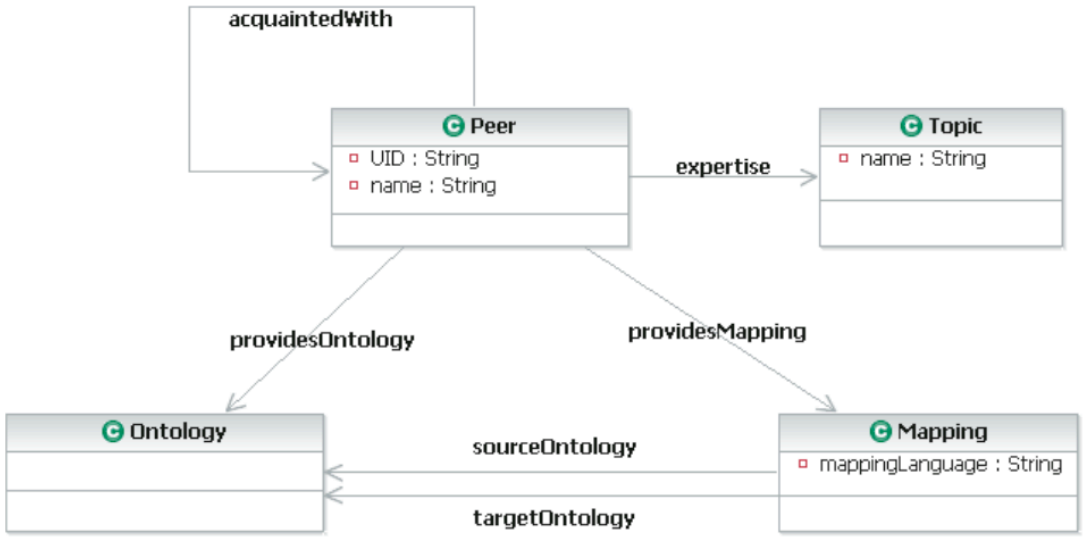

Figure 3.2: Overview of the P-OMV Ontology

network then consists of local nodes, each with a set of acquaintances, which define the node-to-node network topology. The property `providesOntology` describes the relationship between the node and the ontologies provided by the node. It is essential for locating relevant information resources in the network. The property `providesMapping` is used to describe which mappings between ontologies a node provides. Mappings are used to describe the correspondences between different ontologies provided by the nodes. The properties `sourceOntology` and `targetOntology` specify the ontologies that are being mapped. In general, mappings need not be symmetric, a distinction between mapping source and target is therefore required. The property `mappingLanguage` is used to indicate the language that is used to express the mapping.

Chapter 4

NeOnDBMap – Creating Mappings between Ontologies and Database Schemata

4.1 Overview

The distributed query answering system, NeOnQA, is supported by a graphical mapping component – NeOnDBMap, which enables the creation of two kinds of mappings: the ontology mapping and the database schema mapping. By using this component, users can establish an ontology representing the schema of a remote or local MySQL database. Furthermore, user can create mappings between the ontology that lifted from database schema and other ontologies by means of ontology mappings. All the created mappings are automatically persisted together with other ontologies that can be integrated for query answering. The following Figure 4.1 gives a general view of NeOnDBMap.

4.1.1 Create a Database Schema Mapping

Following the foundations and approaches introduced in Chapter 3, here we introduce the work flow of creating a database schema mapping. NeOnDBMap currently only supports MySQL database, other databases like Oracle and DB2 will be taken into account in near future. As shown on the left side of Figure 4.2, a connection to the user defined MySQL database should be established first so that its schema information can be extracted and organized as a tree-like structure for. Users can arbitrarily create an OWL class based on the database schema by specifying a certain table column in the database schema. Users can also create a property instance (such as ObjectProperty, DataProperty and AnnotationProperty, etc.) by specifying the relationship between two columns in the table. The mapping is established manually according to the semantics implied by the database schema. At the end, the mapping will be represented as an ontology document and stored as an ontology in the local ontology repository. As we have implement NeOnDBMap as a NeOn Toolkit plug-in, the stored ontology can be directly exported to ontology projects in NeOn Toolkit.

4.1.2 Create an OWL Ontology Mapping

Here we introduce the workflow of creating an OWL ontology mapping (the right side of Figure 4.2). It is started by specifying two local ontologies (which can also be database schema mappings), extracting and forming their TBoxes and RBoxes to tree-like structures. Users can then manually define mappings which bridges the entities from the both sides with same types, such as OWLClass-mapping, ObjectProperty-mapping, and so on. All successfully created mapping entries will be persisted into an XML-based ontology document and stored into the ontology projects in NeOn Toolkit for further actions.

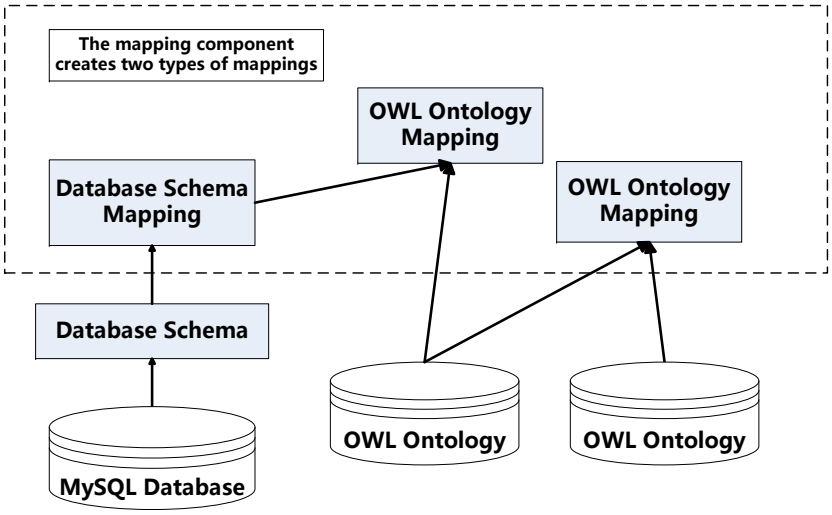

Figure 4.1: Architecture of NeOnDBMap.

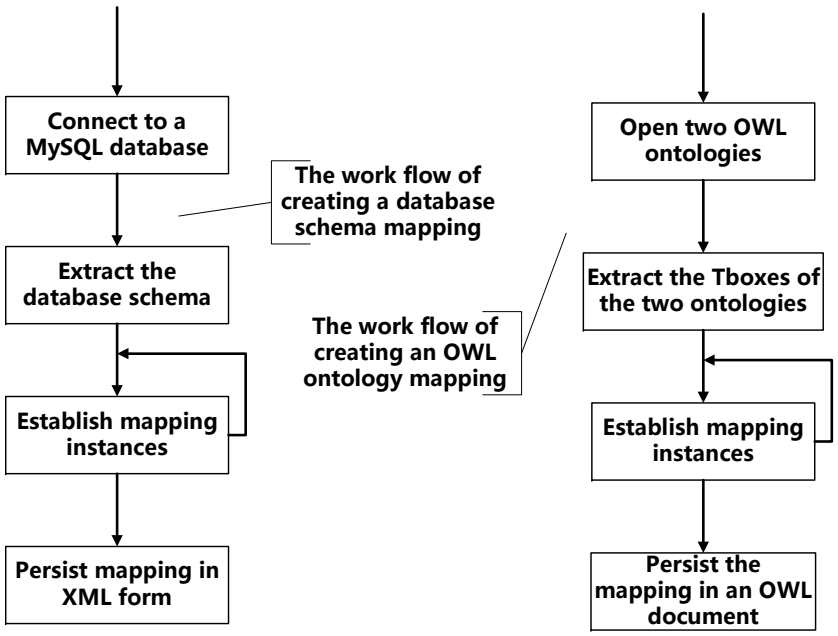

Figure 4.2: The work flow of two parts in NeOnDBMap.

4.2 Use Cases

Now we present two use cases to get started with NeOnDBMap. In the first scenario, we intend to create a database schema mapping for the database FIGIS[1] residing locally(Please note: Remote databases can be also accessed by specifying its IP address with the database name). The sample procedure can be as follows:

- *Step 1:* We should first connect to the database by specifying the database's address, its username and password. If the connection successfully established, the database schema would be automatically extracted as a tree-like structure as shown in Figure 4.3.
- *Step 2:* Now we can manually select a certain column to create an OWLClass-mapping, or two columns in the same table to create an ObjecProperty, DataProperty or AnnotationProperty. The creation of mapping entries is usually according to the semantics involving in the database schema. (Figure 4.3)
- *Step 3:* Do not forget to click the "save" button to persist this database schema mapping after all the necessary mapping entries has been created. (Figure 4.5)

In the second scenario, we intend to create an OWL ontology mapping which bridge two local ontologies with the URIs: `http://www.loa-cnr.it/Files/DLPOnts/DOLCE` and `http://swrc.ontoware.org/ontology`.

- *Step 1:* We should first open two OWL ontologies from the file system. As shown in Figure 4.6, the TBoxes and RBoxes of these two ontologies are extracted once after they are opened.
- *Step 2:* Then we can manually select the OWL entities on the both sides (OWLClass, ObjectProperty or DataProperty) with the same type to create mapping entries. Mapping relation can be either "sub-relation" (indicating "subClassOf" if two OWLClass-es selected), "super-relation" (indicating "superClassOf" if two OWLClass-es selected), or "equivalent-relation" (indicating "equivalentClassOf" if two OWLClass-es selected). (Figure 4.7 and Figure 4.8)
- *Step 2:* Similarly to the creation of database schema mapping, it is also necessary to click the "save" button to persist the mapping file at last. (Figure 4.9)

[1] `http://www.fao.org/fi/figis/index.jsp`

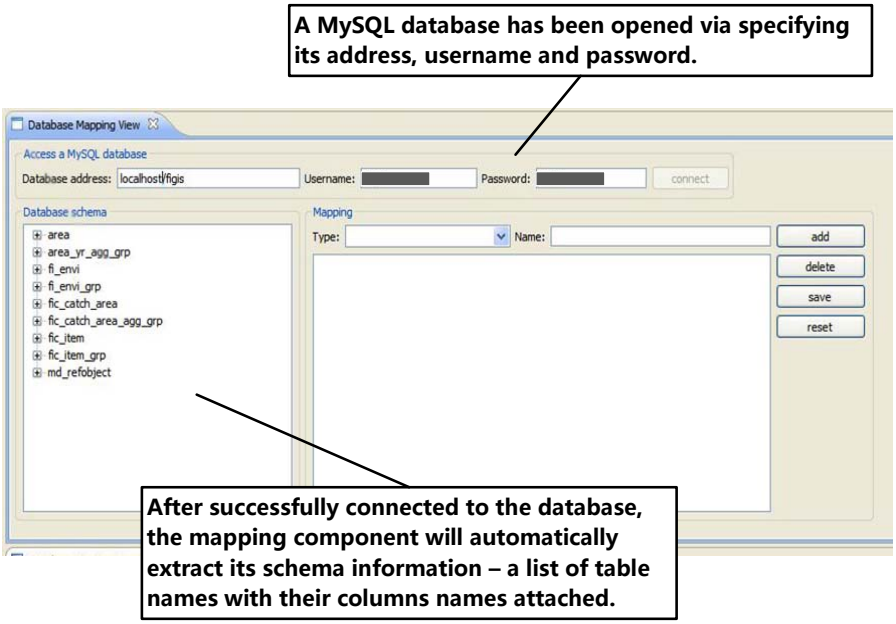

Figure 4.3: Connecting to a MySQL database.

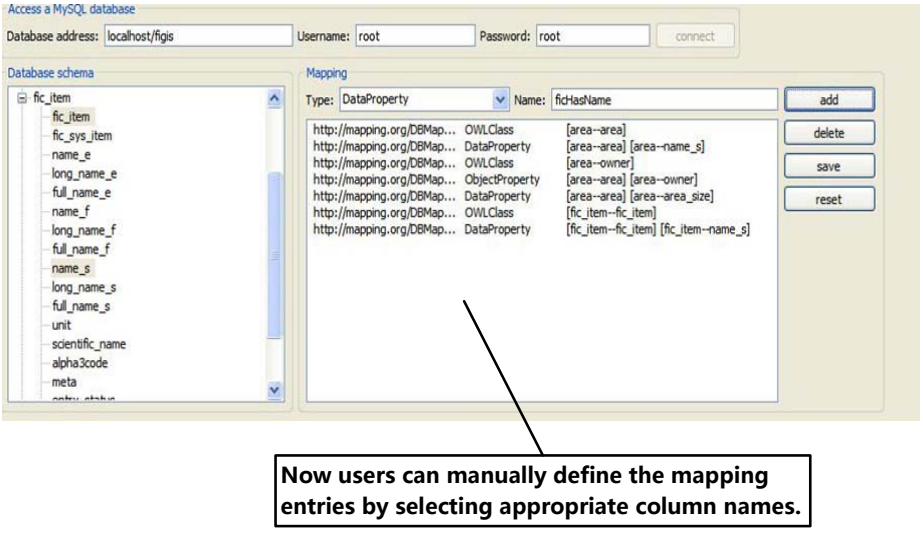

Figure 4.4: Editing mapping entries.

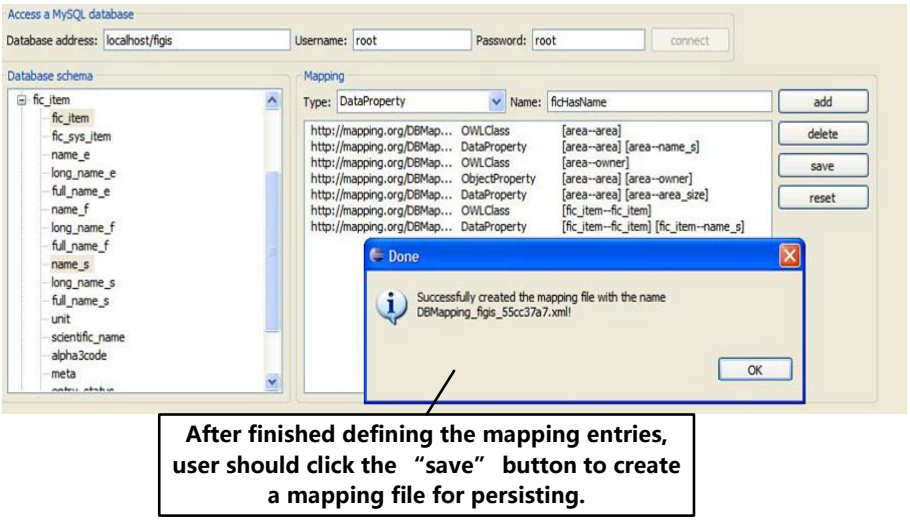

Figure 4.5: Saving a database schema mapping to the file system.

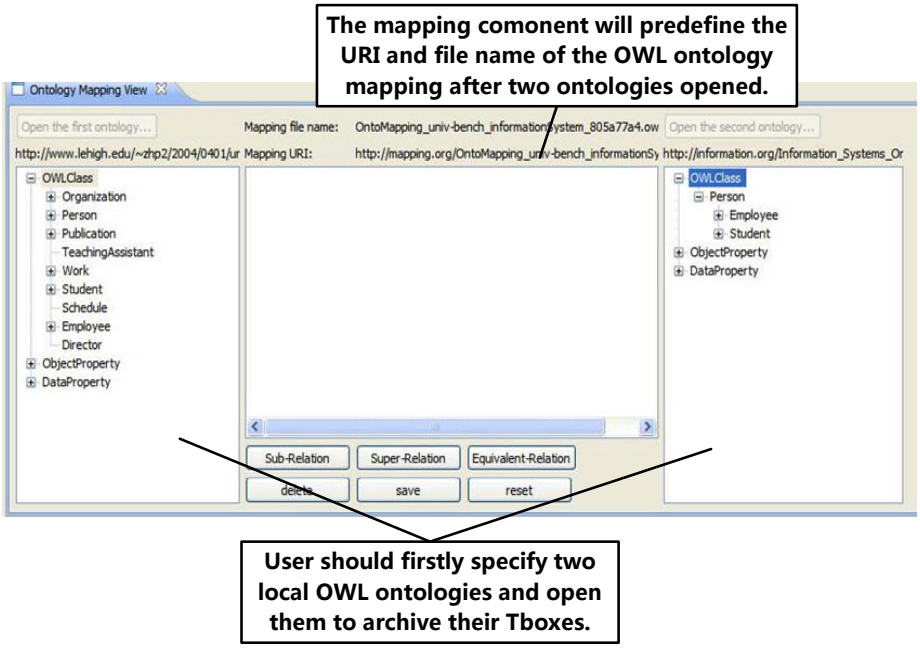

Figure 4.6: Open two local OWL ontologies.

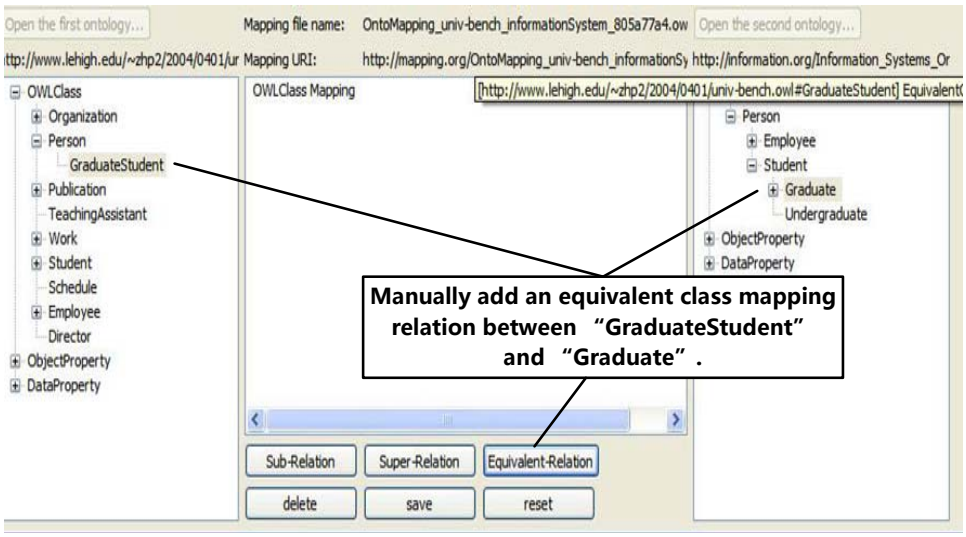

Figure 4.7: Manually editing mapping entries.

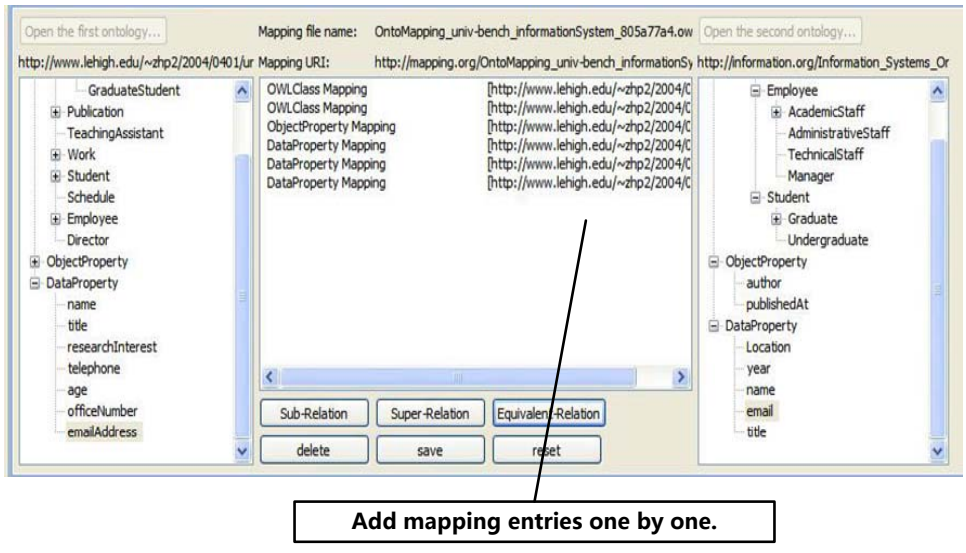

Figure 4.8: Manually editing mapping entries.

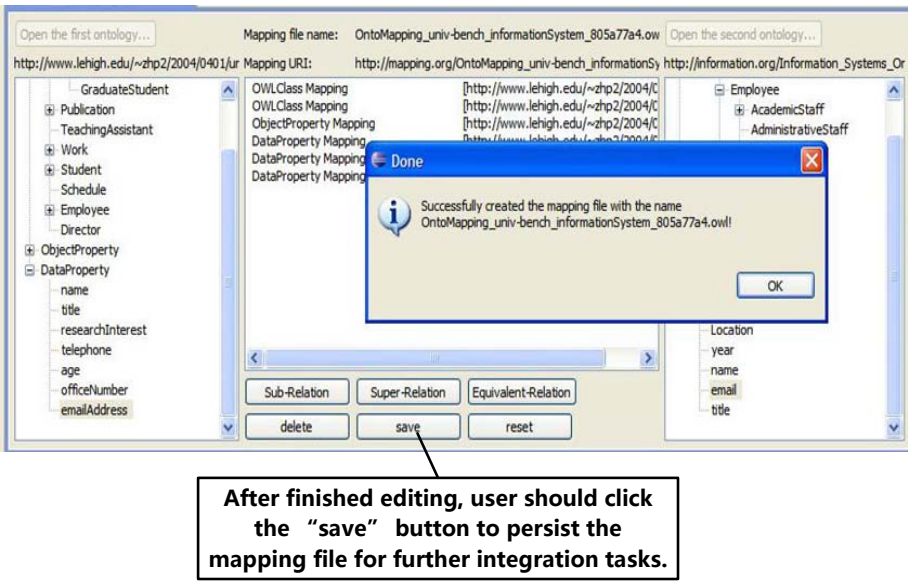

Figure 4.9: Saving an OWL ontology mapping to the file system.

Chapter 5

NeOnQA – An Infrastructure for Distributed Query Answering over Semantic Data

5.1 Overview of Structure

After creating ontologies that represent relational database schemata and mappings between ontologies by using NeOnDBMap, we need to integrate the ontologies and mappings and perform distributed query answering tasks. We have developed an application, called NeOnQA, to support semantic query answering over integrated distributed semantic data, including ontologies and relational databases. In this section, we first provide a brief overview of NeOnQA. Figure 5.1 depicts an architecture of one NeOnQA node that interacts with other NeOnQA nodes in the distributed network. Next, we introduce the components in this architecture in detail.

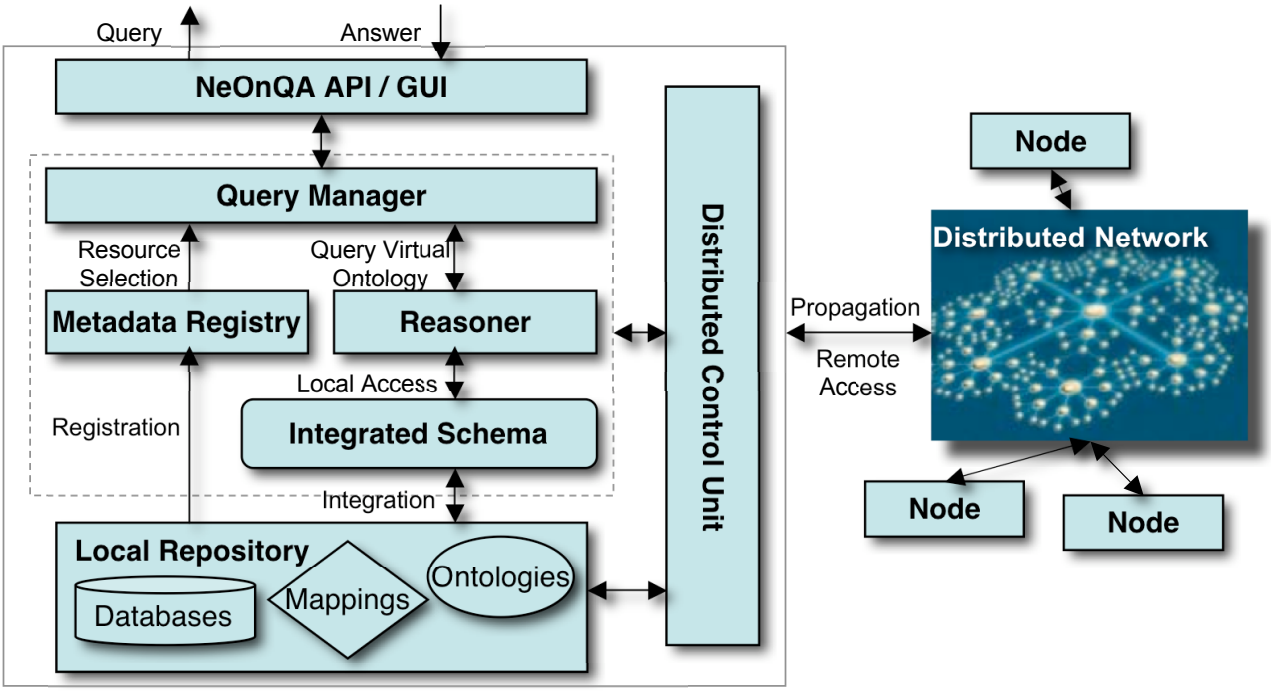

Figure 5.1: Overview of NeOnQA Architecture

The *local repository* consists of databases, mappings and ontologies. In this deliverable, we consider ontologies as OWL DL ontologies and mappings as database-ontology mappings, or ontology-ontology mappings.

The mappings are responsible for linking heterogeneous resources. The local repository can be shared in the distributed network.

The *integrated schema* provides an integrated view towards the local repository. We apply logic integration to the database schema and ontology TBox, formulating an integrated schema using mappings. It therefore doesn't matter whether the data are stored in database or ontology ABox as we only need to query against the integrated schema. We can also access and integrate remote databases and ontologies by using *distributed control unit* introduced later. Here we mainly focus on discussing the approaches for integrating databases as our previous work have addressed the problem of distributed ontology ABox integration and query answering [HW07].

The *query manager* is the component responsible for answering queries over extracted database schemata in the distributed network. Here we consider queries as conjunctive queries over OWL DL ontologies that are mapped with database schemata. The query process can be divided into two steps: Resource selection and query answering.

1. The purpose of the resource selection is to search resources in the distributed network that are relevant to answer a particular query using the *metadata registry*. The Metadata Registry maintains metadata about resources available (i.e. nodes, ontologies, and mappings), which may be accessible either locally or remotely in the distributed network. We extend the selection approach introduced in our previous work [HW07] by enabling the identification of mappings in the metadata of distributed nodes. We introduce a concept of "integrated schema" that logically integrates relevant databases and mappings in the distributed network, represented using the mapping formalism described in Section 3.2.

2. In the step of query answering, the query is evaluated against the integrated schema by the *reasoner*. In NeOnQA, we rely on KAON2 as the underlying reasoning engine, as KAON2 does not retrieve the remote data but only accesses the remote schema [HW07].

The *distributed control unit* is an important component that is mostly different from the P2P network sub-layer introduced in [HW07]. On the on hand, we use web service to propagate the resource in local repository. On the other hand, the communications with remote nodes for retrieving schema or sending results are established by direct JAVA socket connection due to overheads occurred in our real life experiments in using web service for data transfer.

Users can issue queries by using *NeOnQA GUI and API*, get results of queries, identify local and remote resource, and manage the local system. We use SPARQL. as conjunctive query language in NeOnQA.

5.2 Application

After a theoretical introduction and overview of implementation of NeOnQA provided in previous sections, this section is concerning about an actual application. We expect user to have a brief understanding to NeOnQA through comprehensive depiction with screenshots about how to get started using NeOnQA. This section includes three aspects: (1) Server and configuration, (2) resource selection and metadata.

5.2.1 The Server and Configuration

In the scenario of distributed network, each node should be a client that can access the local or remote ontologies, which are representing schemata of database, and meanwhile, each node also publishes local resources. Different from other networked ontology systems like KAONp2p [HW07], we have three different data communication mechanisms between nodes simultaneously:

- Classic TCP connections between distributed nodes for transferring query answering requests and results;

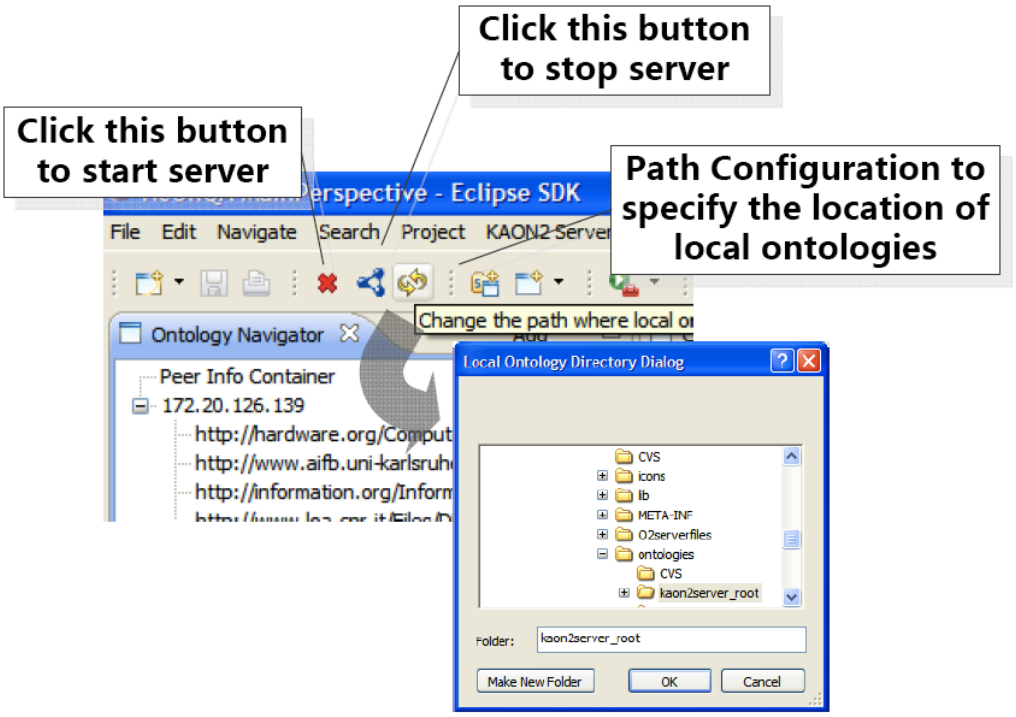

Figure 5.2: The server configuration component.

- Java Remote Method Invocation (RMI) for connection with KAON2 server and databases;
- Oyster service for ontology metadata propagation and discovery.

Based on these communication mechanisms, we implement an application for server configuration (c.f. Figure 5.2). This part is consisted of three buttons embed into the toolbar on the top of the eclipse framework. They are illustrated from right to left as follows:

- The `Folder Configuration` button specifies the location of local ontologies. Configuration will be persisted when the application exits and restored when restarts. However, please note the folder can not be changed during the execution of KAON2 server.
- The `Start Server` button enables a facility to start the KAON2 and Oyster2, loads local ontologies, and also initialize internal services to allow interaction with remote nodes, which are all supposed to be in a global network. Starting the server is the first step to run this application.
- The `Stop Server` button is used to terminate the KAON2 server and Oyster2, empty the data model instances and stop internal services. Please note: Clicking this button is a mandatory step before exit the application for correct future operations.

5.2.2 Resource Selection and Metadata Management

The ontology navigator keeps a group of nodes under control. Each node is presented by its IP address, with a list of its ontologies (indicated by the URIs) appended. All the entries are gathered together to build up a tree-like structure so that users can intuitively identify which ontology belongs to which node. Once the Start Server button is clicked, the local node with its ontology information are added into this tree. To add a remote node, user can simply click the `Add` button, either put selection from node information retrieved by Oyster2 or manually type in the IP address of desired node. Then the information about ontologies and their

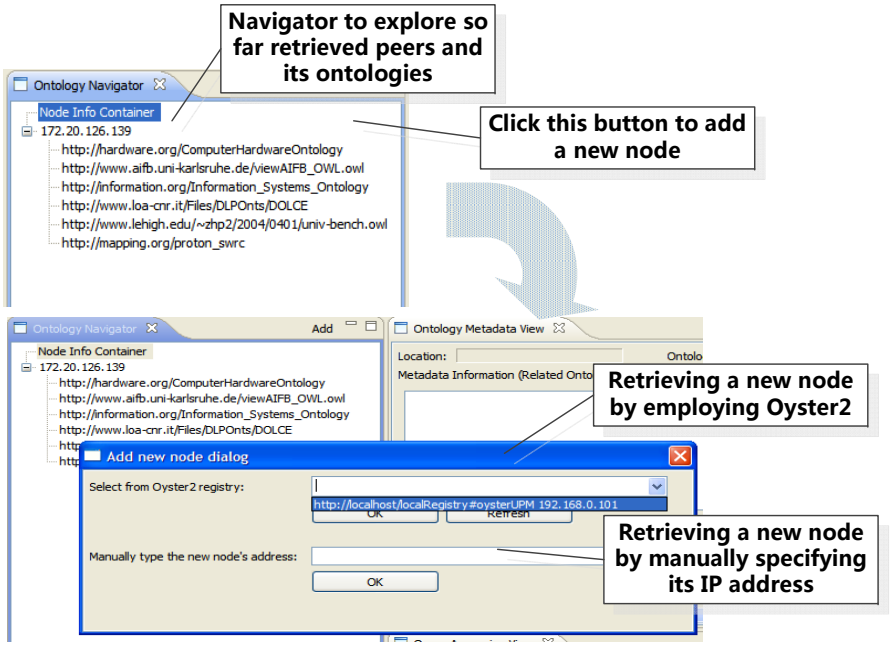

Figure 5.3: Resource selection component.

mapping metadata from this node will be also obtained by accessing appropriate remote services. After that, a data model for this new node will be built up and appended as a node to the navigator tree. The screenshot related to the ontology navigator part is shown in Figure 5.3

When a certain ontology is selected in the navigator, the correspond ontology metadata loading from Oyster will be shown in the metadata management part. As shown in Figure 5.4, existing related ontologies are outlined in a list viewer, which could be either located in local or in remote. Related ontologies are identified by its URI, together with its node's IP address. New related ontologies can be selected from the currently accessible ontologies, by means of a set of widgets below. Of course entries in the list can also be removed so that certain related ontologies are excluded from the integration scope. The button `Set This Ontology for Query` is a critical task for the preparation of query answering tasks, since it will analyze implicitly the related ontologies with their locations, calculate an optimized solution for a suit of distributed query so that the query task can be performed under a highest level of concurrency.

5.2.3 Query Answering

The precondition of a query task is that a target ontology has to be chosen for querying by clicking the `Set This Ontology for Query` button in the metadata management part. If satisfied, users can then put in the query by using the SPARQL language and query over this ontology, which has already been integrated with its related ontologies. After a query is executed, the time cost is presented. All the answers, either originated from each ontology itself or generated from the deductive reasoning progresses after integrating these ontologies, are collected in a list to the result viewer. The query answering part is shown in Figure 5.5. Please note the ontology here is either an OWL DL ontology or an ontology representing a database schema.

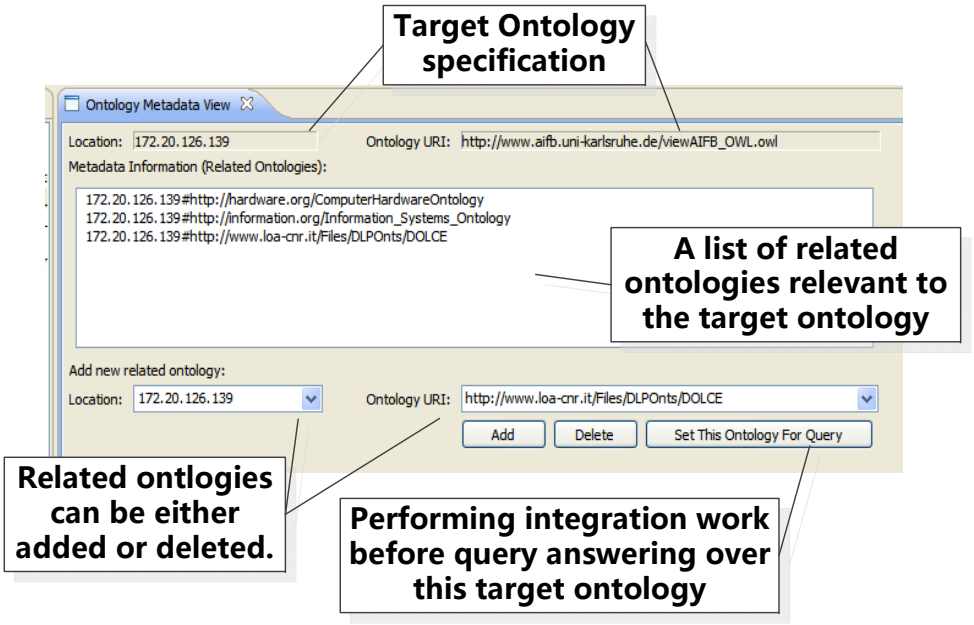

Figure 5.4: Metadata management component.

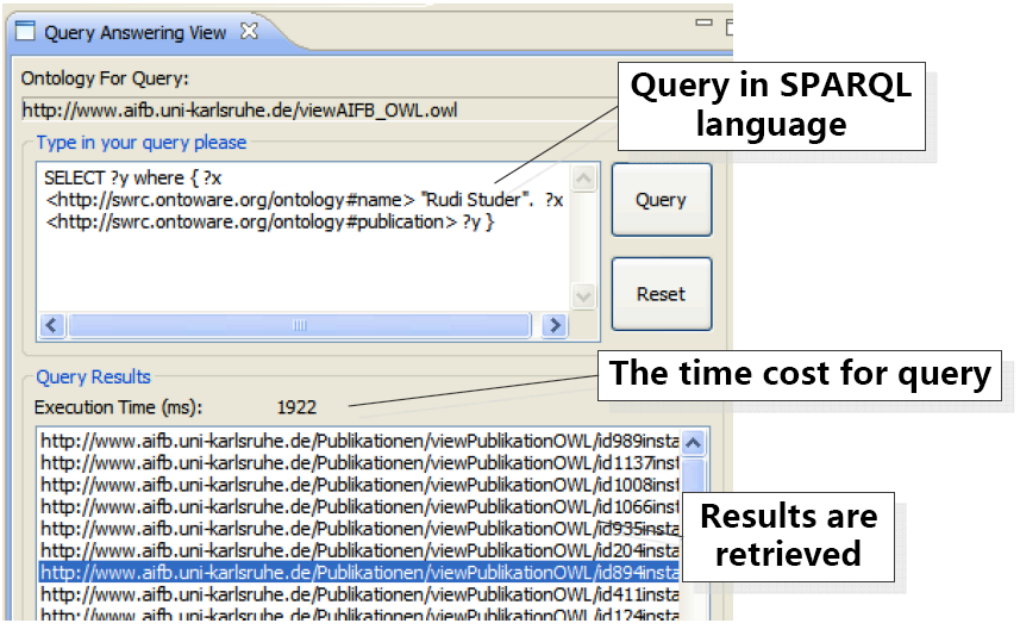

Figure 5.5: The query answering component.

5.3 Experimental Evaluation

After the concrete description about the NeOnQA, this section outlines two specific use cases to evaluate the functionalities, scalability and performance of our approach. The scenario and results for each use case are depicted as follows. We intend to show that NeOnQA is fully functioned for integrating heterogenous data sources including ontologies and relational databases, and more efficient on distributed query answering over semantic data.

5.3.1 Heterogenous Data Integration

Before performing the evaluation illustrated as follows, we prepared a set of ontologies and two MySQL databases for integration and query answering purposes, which were containing real-life data from FAO. The ontologies could be divided into two categories with different schema: one followed the FIGIS ontology [1] and the other followed the ASFA ontology[2]. Each node held a data set with a size of approximately 8-10 MByte. The two MySQL databases were established with heterogenous schemas at different locations, each of which contained also a data set with a volume of ca. 10 MByte for ASFA and FIGIS databases. Further, we predefined two ontology-schema mappings to connect to these databases respectively and an ontology mappings according to our mapping formalism to relate the FIGIS ontology with the ASFA ontology in both directions.

We evaluated the functionality and scalability of NeOnQA to integrate heterogenous data sources by presenting four experimental deployments – (1) a two-node established network infrastructure, each of which held an ontology (one FIGIS and one ASFA); (2)a four-node established network infrastructure, each of which held an ontology (two FIGISs and two ASFAs); (3) a four-node established network infrastructure, three of which held ontologies (one FIGIS and two ASFAs) and the rest held an ontology-schema mapping; (4) an eight-node established network infrastructure, six of which held ontologies (three FIGISs and three ASFAs) and the rest two held the ontology-schema mappings.

For each experimental deployment, we manually put the ontology mapping on an arbitrary node and choose it to performed integration work. Three SPARQL queries from the FIGIS benchmark of different complexity were selected and computed over each of the integrated information systems[3]:

- `SELECT ?x WHERE { ?x rdf:type ub:Area }`
- `SELECT ?x ?y WHERE { ?x rdf:type Area . ?y rdf:type ub:hasName . ?y ub:hasProduct ?x}`
- `SELECT ?x ?y ?z WHERE { ?x rdf:type ub:Area . ?y rdf:type ub:LongName . ?z rdf:type ub:Product . ?x ub:hasProduct ?z . ?z ub:hasName ?y . ?x ub:hasName ?y }`

In order to reflect the effectiveness in the integration work of NeOnQA, we disabled the distributed computation capability when answering these queries. This was realized by explicitly defining each ontology (either a FIGIS ontology, an ASFA ontology or an ontology-schema mapping) holding all other ontologies as its "related ontologies". The results of computing these queries under each deployment are depicted in Figure 5.6.

The results show that, the time cost for query answering are proportionally increased with the size of data in the integrated information system (from experiment 1 of ca. 20 MByte, experiment 2 of ca. 40 MByte and experiment 3 of ca. 80 MByte), but slightly affected by the form of data sources (from experiment 2 with four ontologies and experiment 3 with 3 ontologies and one relational database).

[1] `http://www.fao.org/fishery/figis/`
[2] `http://www.fao.org/fishery/asfa/`
[3] The prefix `ub` stands for FIGIS ontology

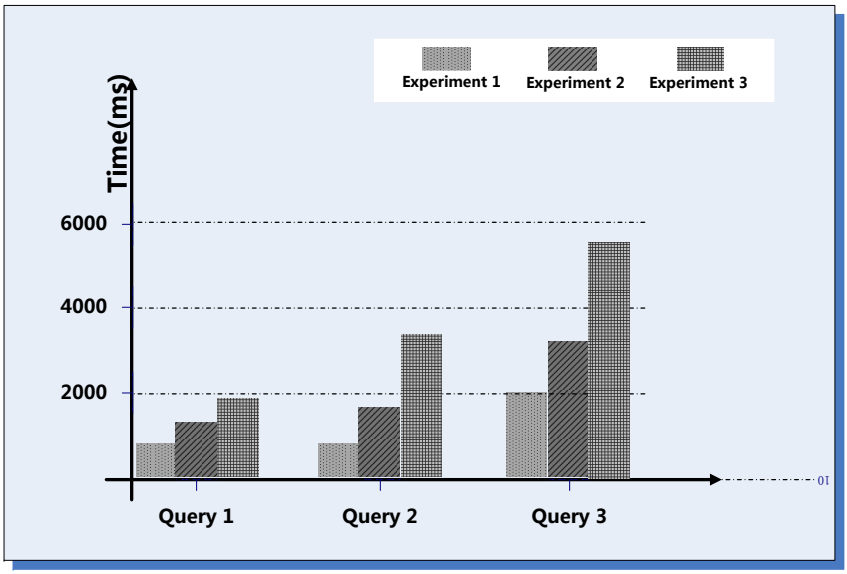

Figure 5.6: The time consumption of three queries for different size of integrated data.

5.3.2 Query Answering over Integrated System

In this section, we present experimental results for evaluating the cost for query answering by NeOnQA. We focus on the performance comparisons with another approach called KAONp2p – the one also developed for computing queries over ontology integration systems in a decentralized setting. In Section 1.3 we've already given a brief introduction for this approach, which employs a similar expressive mapping formalism for mediating heterogeneous ontologies as NeOnQA, and gathers needed parts of remote ontologies into local to compute queries. Since the performance of query answering in KAONp2p is essentially dominated by the size of the data [HW07], we stand for the point of affections by the degree of ontology distribution and heterogeneity, to show how NeOnQA has its improvement in distributed computation.

We again employed the FIGIS and ASFA ontologies the same queries used in the previous section to evaluate the performance differences between NeOnQA and KAONp2p. Three experiments designed in terms of the degree of distribution were implemented under two scenarios: (1) the network infrastructure built by NeOnQA and (2) the network infrastructure built by KAONp2p. We deployed a four-node network infrastructure. Each node held one ontology that represents as database schema ($\mathcal{O}_1$ on node 1, $\mathcal{O}_2$ on node 2, $\mathcal{O}_3$ on node 3, $\mathcal{O}_4$ on node 4). We assumed that $\mathcal{O}_1$ had been predefined to have relations to other three ontologies.

Unlike ontologies, the interconnections between database schemata are usually controlled by users, therefore we can define whether two ontologies that represent database schemata have mapping between each other.

Within each experiment, all the three queries were executed on node 1 under different situations of integrity. The results were:

- In the first experiment, we defined additionally that all the other three ontologies had relations between each other by specifying the metadata of each ontology in NeOnQA. Queries were performed over such an integrated information system both in NeOnQA and KAONp2p. (The time consumption for both cases is shown in Figure 5.7.)
- In the second experiment, we defined additionally that $\mathcal{O}_2$ and $\mathcal{O}_3$ were related with mapping, but not together with $\mathcal{O}_4$. Queries were performed over such an integrated information system both in NeOnQA and KAONp2p. (The time consumption for both cases is shown in Figure5.8.)

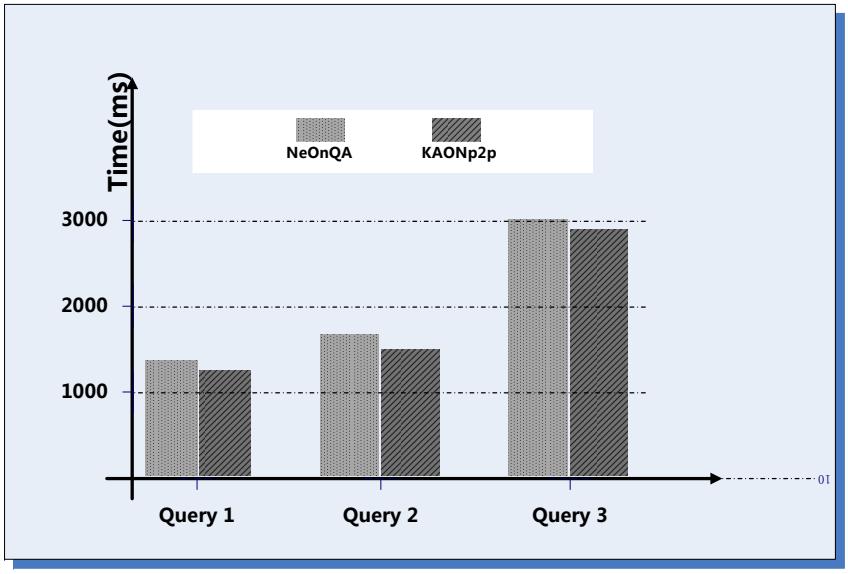

Figure 5.7: The Time consumption in experiment 1.

- In the third experiment, we defined additionally that there's no relations between $\mathcal{O}_2$, $\mathcal{O}_3$ and $\mathcal{O}_4$. Queries were also performed over such an integrated information system both in NeOnQA and KAONp2p. (The time consumption for both cases is shown in Figure 5.9.

In the above three experiments, we consistently had the same number of answers for each queries, whereas there were 10026 answers for query 1, 5764 answers for query 2 and 9 answers for query 3. The results were obvious: (1) NeOnQA consumed slightly higher time to perform distributed query answering in the first experiment due to network overheads caused by extra data communications (including transmitting virtual ontologies, queries and also answers); (2) it showed large improvements in the next two ones since NeOnQA had additional analysis of the ontology metadata and computed the queries with maximum possible distribution and concurrency, that is, queries were performed at the same time on two nodes in experiments 2 and on three nodes in experiment 3.

5.3.3 Summary

Through the two evaluations we've successfully integrated heterogenous data sources in different scales and performed queries over the integrated systems . On the one hand, arbitrary number of ontologies and relational databases can be integrated in a decentralized way to form a single view of integration system through appropriate mappings. On the other hand, queries over integrated information system can be computed without losing any answer, whereas its performance greatly affected by the relations between autonomous data sources. These two aspects implicitly reflects the performance and effectiveness of NeOnQA for information integration and query answering. According to the evaluations, we argue that although the size of the integrated data dominates essentially to the performance of query answering, the mechanism of distributed computation is indeed an effective solution for real life data.

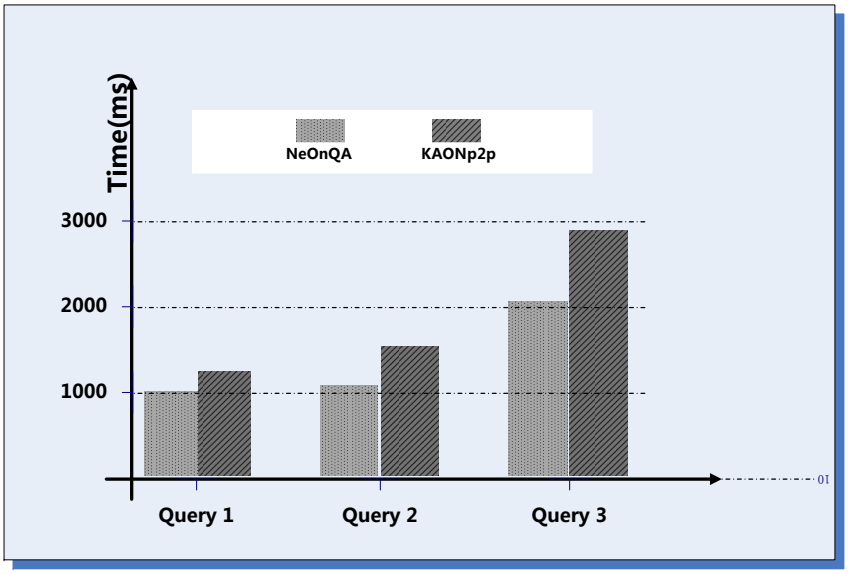

Figure 5.8: The Time consumption in experiment 2.

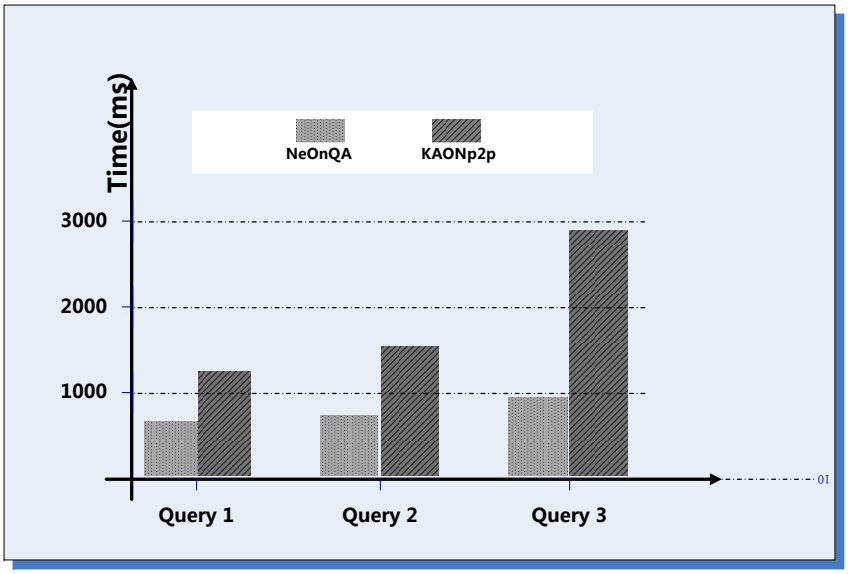

Figure 5.9: The Time consumption in experiment 3.

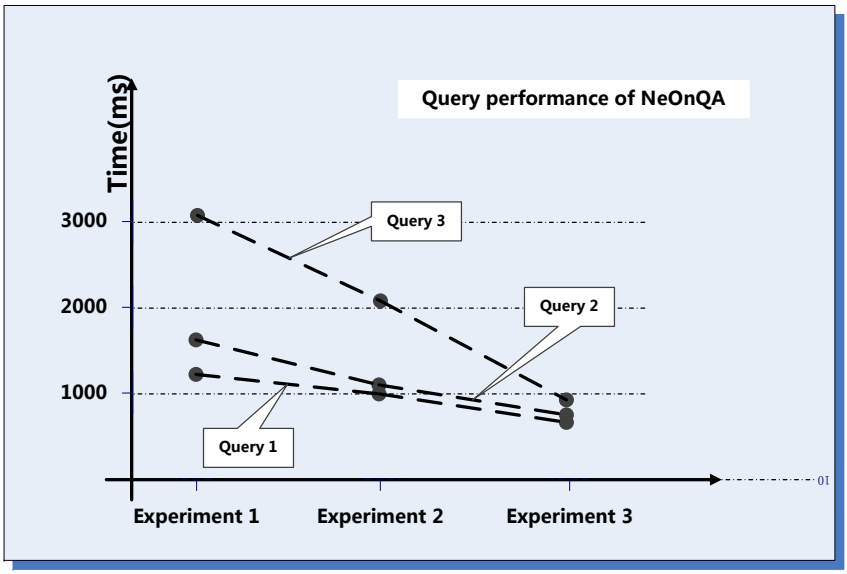

Figure 5.10: The query performance in three experiments.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This deliverable provided an introduction to current development of tool-support of database integration and query answering over autonomous, heterogenous data sources and presented two novel applications that support the integration and query answering tasks: NeOnDBMap and NeOnQA.

- A visualized tool, NeOnDBMap, was introduced to create ontology- database mappings for relational databases. NeOnDBMap enables users to involve traditional databases into a semantic information integration system, and retrieve answers from them by performing a deductive query by using our optimized distributed query answering tool – NeOnQA.
- NeOnQA was established based on an integrated distributed network infrastructure, in which nodes were interacting with each other while ontologies were still residing on distributed nodes. Under such an environment, users on arbitrary nodes were able to discover and manage heterogenous ontologies either on local or at remote node and perform query answering tasks with a simple and intuitive manner.

Besides, the fundamental facilities for ontology management were employed from underlying tools (KAON2, Oyster).
NeOnDBMap and NeOnQA provided an effective way to build semantic integration systems, and optimized the performance of query answering over these integration systems. The contribution of this deliverable falls into two aspects in detail:

- For issues related to *data integration*, it employed the progresses from several research efforts to establish a model for heterogenous data integration. Data sources could either be traditional relational databases or ontologies containing actual data and local schemas. A group of global ontologies built up a decentralized and networked structure to define the mappings between these autonomous data sources. Users on the top had an overview of the integration system and perform query answering tasks over it.
- For issues related to *query answering*, it took especially the distributed query answering into account and proposed an mechanism to perform distributed computation capability for query answering. As queries were concurrently executed on several nodes, it balances largely on the computation load and performance optimization.

Our approach was illustrated both in theoretical and practical points of views throughout this deliverable. We first discussed its theoretical basis on how to integrate heterogenous data sources (OWL DL ontologies and relational databases) and how to compute queries over an integrated information system. We gave a brief depiction to its paradigm, its architecture and its work flows with a comprehensive explanation. We provided a suit of intuitive and concrete guidelines for its usage. Through a series of use cases elaborated, our work in this deliverable had been substantiated to fully satisfied the motivation described in Chapter 1.

6.2 Future Work

There are several research directions which stem from the work presented in this deliverable:

- *Data integration* is a major problem in the Semantic Web that many researches concentrate in. Integrating heterogenous data sources, especially in dealing with other data formats, such as XML data, should be further considered. Further, automatic database schema import and mapping to ontology is another direct advance that we going to pursue.
- *Query answering*. Concerning query answering, there are several future directions. An obvious next step will be deploying advanced peer-to-peer query answering approaches from traditional database community, which has been proved to be efficient, mature and robust [HIM+04]. Another future work could be applying preference based approach for consistent semantic query answering over inconsistent databases that are believed to be common on today's Web.

The work will continue and devote to provide a finely encapsuled platform within distributed network infrastructure for better data integration and query answering in the NeOn project.

Bibliography

[ABM05] Yuan An, Alexander Borgida, and John Mylopoulos. Constructing complex semantic mappings between xml data and ontologies. In *International Semantic Web Conference*, pages 6–20, 2005.

[BCM+03] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, New York, NY, USA, 2003.

[CG05] Diego Calvanese and Giuseppe De Giacomo. Data integration: A logic-based perspective. *AI Magazine*, 26(1):59–70, 2005.

[CGL+04] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. What to ask to a peer: Ontolgoy-based query reformulation. In *KR*, pages 469–478, 2004.

[CGLR04] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of PODS'04*, pages 241–251, 2004.

[CWW+06] Huajun Chen, Yimin Wang, Heng Wang, Yuxin Mao, Jinmin Tang, Cunyin Zhou, Ainin Yin, and Zhaohui Wu. Towards a semantic web of relational databases: A practical semantic toolkit and an in-use case from traditional chinese medicine. In *International Semantic Web Conference*, pages 750–763, 2006.

[DL06] Dejing Dou and Paea LePendu. Ontology-based integration for relational databases. In *SAC*, pages 461–466, 2006.

[GR03] Francois Goasdoue and Marie-Christine Rousset. Querying distributed data through distributed ontologies: A simple but scalable approach. *IEEE Intelligent Systems*, 18(5):60–65, 2003.

[Hal01] Alon Y. Halevy. Answering queries using views: A survey. *VLDB J.*, 10(4):270–294, 2001.

[HBE+04] P. Haase, J. Broekstra, M. Ehrig, M. Menken, P. Mika, M. Plechawski, P. Pyszlak, B. Schnizler, R. Siebes, S. Staab, and C. Tempich. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the Third International Semantic Web Conference, Hiroshima, Japan, 2004*, NOV 2004.

[HIM+04] Alon Y. Halevy, Zachary G. Ives, Jayant Madhavan, Peter Mork, Dan Suciu, and Igor Tatarinov. The piazza peer data management system. *IEEE Trans. Knowl. Data Eng. (TKDE)*, 16(7):787–798, 2004.

[HM] Peter Haase and Boris Motik. |a mapping system for the integration of owl-dl ontologies.

[HRO06] Alon Y. Halevy, Anand Rajaraman, and Joann J. Ordille. Data integration: The teenage years. In *VLDB*, pages 9–16, 2006.

[HRW+06] Peter Haase, Sebastian Rudolph, Yimin Wang, Saartje Brockmans, Raul Palma, Jéróme Euzenat, and Mathieu d'Aquin. D1.1.1 networked ontology model. Technical Report D1.1.1, Universität Karlsruhe, NOV 2006.

[HSH+05] Jens Hartmann, York Sure, Peter Haase, Raul Palma, and Mari del Carmen Suárez-Figueroa. Omv – ontology metadata vocabulary. In Chris Welty, editor, *ISWC 2005 - In Ontology Patterns for the Semantic Web*, NOV 2005.

[HSvH04] P. Haase, R. Siebes, and F. van Harmelen. Peer selection in peer-to-peer networks with semantic topologies. In *Proceedings of the First International IFIP Conference on Semantics of a Networked World: ICSNW 2004, Paris, France, June 17-19, 2004.*, pages 108–125, 2004.

[HT00] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 399–404. AAAI Press / The MIT Press, 2000.

[HW07] Peter Haase and Yimin Wang. A decentralize infrastructure for query answering over distributed ontologies. In *The 22nd Annual ACM Symposium on Applied Computing (SAC'07)*, Seoul, Korea, 2007. To appear.

[Len02] M. Lenzerini. Data integration: a theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM Press, 2002.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB*, pages 251–262, 1996.

[MHS07] Boris Motik, Ian Horrocks, and Ulrike Sattler. Bridging the gap between owl and relational databases. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 807–816. ACM, 2007.

[MSS04] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with Rules. In *International Semantic Web Conference*, pages 549–563, 2004.

[PH01] Rachel Pottinger and Alon Y. Halevy. Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198, 2001.

[PH05] R. Palma and P. Haase. Oyster - sharing and re-using ontologies in a peer-to-peer community. In *International Semantic Web Conference*, pages 1059–1062, 2005.

[Qia96] Xiaolei Qian. Query folding. In Stanley Y. Su, editor, *12th Int. Conference on Data Engineering*, pages 48–55, New Orleans, Louisiana, 1996.

[RGP06] Jesús Barrasa Rodriguez and Asunción Gómez-Pérez. Upgrading relational legacy data to the semantic web. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 1069–1070, New York, NY, USA, 2006. ACM Press.

[SBT05] Luciano Serafini, Alexander Borgida, and Andrei Tamilin. Aspects of distributed and modular ontology reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence - IJCAI05*, pages 570–575, 2005.

[WHP07] Yimin Wang, Peter Haase, and Raúl Palma. D1.4.1 prototypes for managing networked ontologies. Technical Report D1.4.1, Universität Karlsruhe, FEB 2007.

[WVV+01] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information — a survey of existing approaches. In H. Stuckenschmidt, editor, *IJCAI–01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.