OWSCIS: Ontology and Web Service based Cooperation of Information Sources

Raji Ghawi, Thibault Poulain, Guillermo Gomez and Nadine Cullot
Laboratoire Electronique Informatique et Image
UMR CNRS 5158, Université de Bourgogne
21000 Dijon, France
{raji.ghawi, thibault.poulain, nadine.cullot}@u-bourgogne.fr, gomezcarpio@yahoo.com

Abstract

The growing amount of distributed data over the internet leads to increasing needs for interoperability. Being able to take into account the meaning of information is a real challenge for suitable data sharing. The semantic web and the ontologies are relevant technologies to provide semantic cooperation of heterogeneous sources. We propose a complete architecture OWSCIS (Ontology and Web Service based Cooperation of Information Sources) which allows to query a cooperation of information sources. The semantic of the local data is expressed using local ontologies which are mapped to a reference ontology. This reference ontology can be queried by an end user to transparently access the cooperation. The different components of the architecture are described: the data providers, the knowledge base, the inter-ontology mapping process, the visualization service and the querying service. A special focus is done on the latter service.

1. Introduction and Motivation

In the last few years, huge amounts of information are becoming available over the web and over corporate and governmental networks. The exponential growth of the Internet as well as current advances in telecommunication technologies led to an unparalleled increase of the number of online information sources. However, the access to information remains limited as long as it is stored in separate sources.

This situation, along with an increasing specialization of works, a large variety of data representation paradigms, and rising acquisition costs of multimedia data, exacerbates the need for easy methods to combine data from different sources. To achieve this, information has to be shared, combined and/or exchanged between multiple actors (individuals, companies and governments) and/or from multiple information sources, and accessed transparently by their final receivers. This problem is known as *information integration* or *interoperability problem*.

Interoperability is hard to achieve due to: 1) the *distribution* of information over multiple sources - a query could not be answered by the data available from a single information source, 2) the *heterogeneity* of the different information sources, and 3) the *instability* - new information sources appear every day, while others disappear, while existing information sources change the format of their data, or change their content.

Most of research efforts to reach interoperability tend to only address interoperability issues at the platform and syntactic level. However, such efforts fail to focus on the semantic, which contains an important part of information. Discrepancies in the way information sources are specified hinder information sharing between software applications. Conversely, being able to explicitly model the meaning of information promises to move information integration technology to new levels of flexibility and automation [1].

The Semantic Web is a key approach to the semantic interoperability. It is, as originally proposed by Tim Berners Lee [5], an extension of the current web, in which the web content can be expressed in a way that, in addition to be human readable, can be understood by software agents. The vision of the semantic web energized the development of ontologies. An ontology explicitly describes the various concepts of a given knowledge domain and their semantic relations in a formal way that is agreed by several parties. Hence, it can be taken as a unifying formalism for giving information a common representation and semantics.

In addition to ontologies, web services are increasingly used to support the interoperability between different applications and clients over the web using recently developed internet-oriented data models, standards and protocols such as SOAP, WSDL, and XML. Web services guarantee the

independence of an application from any particular platform or implementation.

We propose a cooperation architecture, called OWSCIS[1] , that uses ontologies and web services technologies to provide an interoperable solution for integrating distributed and heterogonous information sources. Most of the architecture components are encapsulated in web services to perform specific tasks, like the mapping, querying and visualization web services.

In our architecture, information sources may contain different types of data structures. However, all of these sources must be mapped to a local ontology which will express the semantic of information sources. In OWSCIS, information sources are encapsulated in what we call "Data Providers". A data provider is a module that allows to wrap an information source to a local ontology using our tool DB2OWL [9]. Local ontologies are mapped also to a reference ontology.

This paper is organized as follows. In section 2, we give a background and review some related works. In section 3, we give an overview of the general cooperation architecture and its various components. In section 4, we introduce the inter-ontology mapping process. Section 5 presents the querying process in details. In section 6, we discuss our system features and its implemented parts and we conclude with some future work remarks.

2. Background and Related Works

There are many systems that propose to solve the problem of interoperability between distributed heterogeneous information systems using ontologies. It is very difficult to exhaustively cover the state of the art of this domain for several reasons including the large number of proposed systems, and the variety of the contexts within which those approaches are exploited. However, we refer to the most known systems in this area such as BUSTER [16], SIMS [2], KRAFT [12], COIN [11], Carnot [19], Infosleuth [4], and OBSERVER [14]. We refer also to some interesting surveys such as [18], [17], and [6].

There are several criteria that are usually used to compare ontology-based integration systems, such as: the types of information sources involved in the integration, the architecture type, the use of ontologies, the ontology representation language and the query processing. In this section, we present these features as well as some related works, and we compare our system OWSCIS versus these features in section 6.

The *information sources* involved in a cooperation system are the sources used by the system to retrieve the information to answer users' queries. Buccella et al. in [6] divide the information sources feature into two categories: the state of information sources (SIS) and the type of information sources (TIS). The state of information sources may be static or dynamic, in the later case the integration system should provide some means to know which sources are available at a given moment. Examples of dynamic systems are SIMS, OBSERVER, KRAFT, InfoSleuth and COIN. The main types of information supported by the system include: databases, XML documents, files and HTML pages. Most of existing systems support databases but only some of them support semi-structured data (XML or HTML pages) such as OBSERVER, Infosleuth and COIN.

Two types of *architecture* are generally identified in cooperation systems: agent-based and wrapper/mediator-based systems. Agent architectures are designed to allow software processes to communicate knowledge across networks, in high-level communication protocols. They are highly dynamic and open, allowing agents to locate other agents at runtime, discover the capabilities of other agents, and form cooperative alliances. However, if the number of agents is large, the communication among them may become expensive to implement and their interaction become difficult to understand.

Mediator-based architectures are based on two main components: mediator and wrapper. The mediator is usually used to create an integrated view of data over multiple sources, and the wrapper is used to map local information sources to a common data model. In this type of architecture, all the information needed to achieve the integration is stored in the mediator. However, the mediator can make itself appear complex and difficult to manipulate. Also, performance aspects must be taken into account [6]. KRAFT, Carnot, and Infosleuth are examples of agent-based systems, while SIMS and COIN are examples of mediator-based systems.

In [18], Wache et al. distinguish three types of approaches according to the way of *exploiting ontologies* in information cooperation: single, multiple and hybrid ontology approaches. Single ontology approaches use one global ontology to which all information sources are linked by relations expressed via mappings that identify the correspondence between each information source and the ontology. In multiple ontologies approaches, each information source is described by its own ontology and inter-ontology mappings are used to express the relationships between the ontologies. The hybrid approaches combine the two previous approaches. Each information source has its own ontology and the semantic of the domain of interest as a whole is described by a global reference ontology. In these approaches there are two types of

[1] OWSCIS stands for: Ontology and Web Service based Cooperation of Information Sources.

mappings: mappings between an information source and its local ontology and mappings between local ontologies and the global ontology. SIMS is an example of the first approach, OBSERVER of the second and BUSTER of the third.

Literature describes many *ontology specification languages*. We can distinguish between traditional ontology languages (such as Ontolingua, OKBC, OCML, F-Logic, LOOM) and web-based languages (XML, RDF(S), XOL, SHOE, OIL, DAML, DAML+OIL, OWL). These languages are based on different knowledge representation paradigms such as Description Logics (CLASSIC, OIL, LOOM, CARIN, AL-log, DLR, and OWL-DL), and Frame-based systems (F-Logic, OKBC, Ontolingua). We refer to [8] and [18] for a comparison of ontology languages.

The *query process* is a set of steps needed to achieve a query defined by the users. The general approach used to process global queries get through three steps: first, a global query is decomposed into a number of sub-queries such that the data needed by each sub-query are available from one information source. After the decomposition, each sub-query is translated to a query or some queries of the corresponding local information system and sent there for execution. Finally, the results returned from local sources are combined into the answer. Most of cooperation systems follow this general strategy. However, each system has its own particular method to process queries which is different from other systems (see [6] to review different methods of different systems).

3. General Architecture

OWSCIS is a cooperation system between several information sources and aims at answering user queries on these sources in an integrated centralized way. Users can query heterogeneous and distributed information sources simultaneously and combine the obtained results in order to transparently get information that may not be available directly. For each incorporated information source a local ontology is used to describe it. This local ontology has to be linked to actual information; therefore, mappings between the source and the local ontology have to be provided.

Beside local source ontologies, a global ontology is used to describe the semantics of the whole domain of interest. This global ontology is a reference for all incorporated local ontologies and it is supposed to cover their domains. In order to interconnect local ontologies, they are mapped to the global ontology which plays the role of mediator.

Our architecture deals with all steps needed to interconnect various data storing systems, from the creation of a local ontology to the visualization of the results obtained by queries. It consists of several modules and web services, each of them aims at performing a specific task as shown in figure 1. A data provider module is a wrapper that associates an information source with its local ontology. A knowledge base module is a unique mediator used to encapsulate the global ontology with a toolbox and a directory for participating data providers. A mapping web service is used to establish mappings between the local ontologies and the global one. User's queries are submitted only on the reference ontology via a query web service that analyses the queries, decompose them into sub-queries which are redelivered to the relevant data providers. Finally, a visualization web service performs the tasks of suitably presenting the obtained results to the end user.

3.1. Knowledge Base

This is the main component of the architecture; it centralizes all information needed to exploit the whole cooperation system. It is composed of a reference ontology, a mappings directory, and a toolbox. The reference ontology describes an agreement over a specific knowledge domain. All the specified concepts of the local ontologies are assumed to have similar concepts in a reference ontology. Each local ontology refers to specific parts of a domain which is globally covered by a reference ontology. Several reference ontologies can be designed if different knowledge domains are considered but do not yet deal with cases where a data provider describes information overlapping two or more reference ontologies. A reference ontology is generated as an agreement between the data providers.

The reference ontology is described in OWL-DL language2, a W3C recommendation for publishing and sharing ontologies on the web. OWL-DL is based on Description Logics [3], a family of knowledge representation languages, which is characterized by its expressiveness and reasoning power.

The mapping directory is a simple table listing the concepts from the reference ontology and associating each concept with a list of the data providers that have an equivalent concept. It is created during the mapping process and used to quickly find which data provider is relevant to a given query. It does not contain the exact mappings, which are stored in the various data providers.

The toolbox contains all information needed to perform the mappings, including a set of tools and methods that are used by the mapping web service to estimate the similarity between ontologies components. It describes the way the mapping estimation will happen. It basically list which

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

methods should be used for the similarity estimation and their relative importance. If several mapping methods are available, it also defines which one, or which combination of them has to be used.

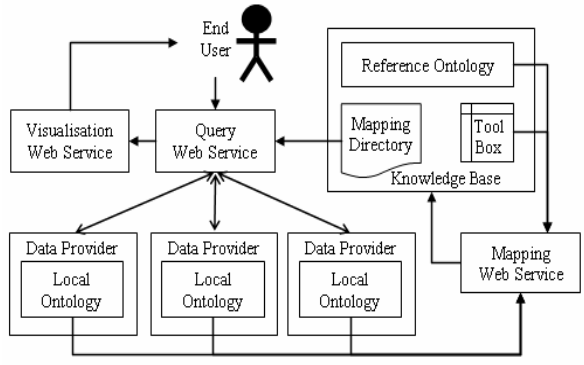

Figure 1. Global architecture of OWSCIS system.

3.2. Data Provider

A data provider encapsulates an information source incorporated in the cooperation system. We consider that an information source may contain different types of data structures. A data provider can contain several of them as long as they are all associated to a common ontology called the Local Ontology which describes the whole semantic of this information source.

The data provider also holds two types of mappings: information source to local ontology mapping, and local ontology to reference ontology mapping, as described in figure 2. In this paper, we only deal with information source based on relational databases.

The local ontology of the data provider does not necessarily exist before the connection to the platform; therefore it has to be created from the information source. We suppose in this paper that the information source is a relational database and that the other types of sources have to be treated in another suitable way. We have developed a tool called DB2OWL that automatically generates a local ontology from a relational database. This tool also generates a description of the mapping between the database and the resulting local ontology. This mapping document is used in the query process as we will see in section 4.2. Our objective is to keep the instances apart from the structure of their ontology. Therefore, the generated ontology only contains the concepts and properties but not the instances, which stay in the database and are retrieved and translated as needed in response to user queries. A data provider service also plays the role of wrapper that translates queries over its local ontology into SQL queries over its data source and reformulates the results according to the local ontology.

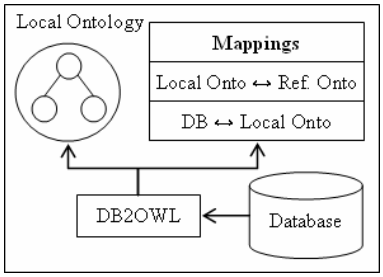

Figure 2. The architecture of the data provider.

3.3. Mapping Web Service

This service is used to map the local ontologies to the reference domain ontology. Any ontology can be mapped to another within two sets of mappings. By centralizing the mappings we ensure a limited number of mappings as well as a reduced loss of information. Any query can be solved over a view on the reference ontology. This allows the simplification of the queries.

To add a new data provider to the architecture, its local ontology must be mapped to the reference ontology following some general guidelines complemented with specific parameters of the knowledge base module.

The mapping web service compares two ontologies using the methods defined inside the knowledge base module toolbox, and produces inter-ontology mappings which will be stored into the appropriate data provider, as well as an up-to-date version of the mappings directory in the knowledge base module. The similarity estimating process is discussed later in section 4.

3.4. Querying Web Service

Once the various data providers are connected to the knowledge base module, users can query them using the querying web service. When a query (expressed in SPARQL language [15]) is submitted to the system, it is analyzed by this service and decomposed into a set of modular queries. Then, using the mapping directory in the knowledge base, the query web service redirects the single queries to the suitable data providers.

When a SPARQL query is received by a data provider service, it is translated into an SQL query using the mappings between the database and the local ontology. The SQL query is executed in the database and its result is encapsulated as a SPARQL answer and returned to the query web service. The query web service collects the responses returned from data provider services and recomposes them in one coherent response which is sent to the visualization web service. The full querying process is described in section 5. The final answer is redirected to the visualization web service which

displays it in an easily-understandable way for the user.

3.5. Visualization Web Service

The visualization service proposes different functionalities including the visualization of the reference ontology or the visualization of the queries and their results. It offers a graphical interface allowing the navigation through the reference ontology and it is possible to visualize the concept hierarchy, the properties with their domain and range.

The visualization service can also be used to show the results of a query in a dynamic and convivial way. The main idea is to use the semantic information described in the reference ontology to enrich the results of the query allowing a more intelligent visualization of them.

The query is analyzed to identify all the relevant concepts and properties participating to the query. Then these components are identified in the reference ontology to extract a coherent sub-part of it, containing these concepts and properties.

An adequate visualization of the results of the query using this ontology sub-part is then chosen including 2D representation (such are graphs, sets or other metaphoric representation) or 3D visualization.

The user can see the results and dynamically queries the proposed representation. The query results are viewed as instances of the concepts and properties specified in the query and satisfying its restrictions.

The visualization process aims at proposing a real dynamic and convivial presentation of the query results helping the user to analyze them quickly.

4. Mapping Process

As mentioned in section 2.3, a mapping web service is used to establish mappings between local ontologies and the reference ontology in the knowledge base module. The mapping process is composed of four main steps: preprocessing, similarity estimation, refining and exploitation.

After a preprocessing step which cleans up the data, we compute a first similarity estimation. The similarity estimation gives a numerical similarity estimation value to all pairs of concepts (C_1, C_2), C_1 being a concept of a local ontology and C_2 being a concept of the reference ontology. It uses several similarity estimation methods described in the toolbox. We use both semantic and structural methods. On one hand, we automatically extract known words from the concepts names and perform a semantic similarity estimation over them. On another hand, we estimate structural similarity by comparing concept names as a string.

Combining the two methods allows us to benefit from both similarity estimation methods. Semantic similarity gives results closer to human inferred similarity than string similarity over known words. String similarity allows us to obtain a similarity measure over the parts of the concepts names that we were not able to recognize as keywords, like dubious abbreviations or acronyms. The various results are normalized and combined using a weighted mean. These weights are defined in the toolbox as well. The resulting table of values is then refined.

The refining step allows to solve cases where the similarity value between two concepts is neither high enough nor low enough to determine whether there is an equivalence or not. This process relies on the iterative application of a set of rules on the structure of both ontologies. It disambiguates the similarity results obtained during the previous step by taking into account the neighborhood of the concepts. Specific neighborhood situations are described by so-called rules provided by the toolbox. The toolbox also describes how the similarity of the pair of concepts or properties is modified when such situations are encountered.

For example, if both fathers concepts of a checked pair of concepts are similar, then this checked pair has more chance to be similar. Therefore, if the condition of this rule is respected then the similarity between the two checked concepts of the pair is increased. This iterative process propagates relevant similarities over the structures of the ontologies.

Once similarities are estimated, they must be translated from their numerical values into mappings. This translation can be done automatically, producing the overall mapping between the two ontologies, or iteratively. In that latter case, the program suggests what appears to be the best mapping, and let the expert validate or not the choice. Once a mapping is validated by the expert, modifications can be made into the similarity table to respect the consistency between the following mapping suggestions and the existing ones.

The resulting mappings are stored in the data provider, to allow the translation of the queries and answers from and to this specific data provider.

5. Querying Process

Users submit their queries (expressed in SPARQL language [15]) to the querying web service in terms of the reference ontology. When the querying web service receives a query, it will decompose it into a set of sub-queries using the mapping directory in the knowledge base. The mapping directory contains a list of reference ontology's components and shows for each concept (respectively, property) which local ontologies have an equivalent concept (respectively, property). Each sub-query will be directed to the suitable data provider where it will be rewritten in

terms of the local ontology and translated to an equivalent SQL query.

The SQL query is evaluated by the local DBMS and its results will be formulated as SPARQL Query Results in XML Format[3]. These results are returned to the querying web service. The various results collected from the different data providers will be recomposed by the querying web service yielding the final result of the user's query. The whole query processing is illustrated in figure 3.

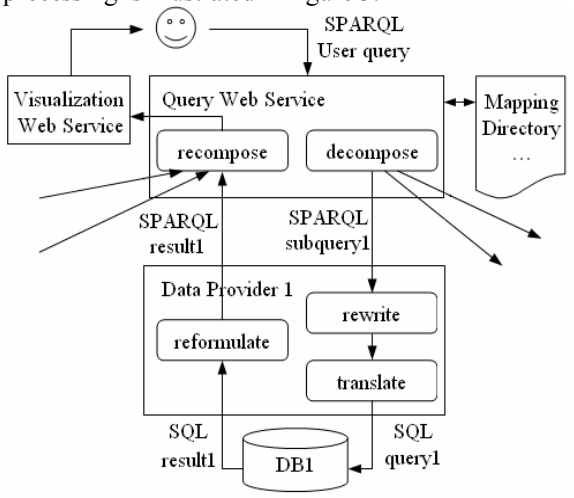

Figure 3. Query processing in OWSCIS.

5.1. Query decomposition

A query is composed of a set of triple patterns; each of them corresponds to a concept or a property in the reference ontology. Using the mapping directory, the query is decomposed into a set of modular queries which will be sent to relevant data providers. For each component (concept or property) in the reference ontology, the mapping directory contains a list of which local ontologies has an equivalent component. The decomposition process depends on the ontological components mentioned in the query triple patterns. For a given local ontology, the sub-query will be a copy of the original query, but contains only the triple patterns that correspond to some components in the given local ontology.

If a variable in the global query is shared between several sub-queries, it will be set in the SELECT clause of each of these sub-queries, and it will be used later for re-composing their results. For example, let us consider the reference ontology in figure (4a) and the two local ontologies in figure (4b) and (4c). Let us consider the query in figure 5 which retrieves the titles of books written by a teacher of Database module. This query is decomposed into two sub-queries shown in figure (6a, 6b) over the local ontologies. Note that the variables `?fn` and `?ln` are shared between both queries, so they are used in the SELECT clause of both of them.

[3] http://www.w3.org/TR/rdf-sparql-XMLres/.

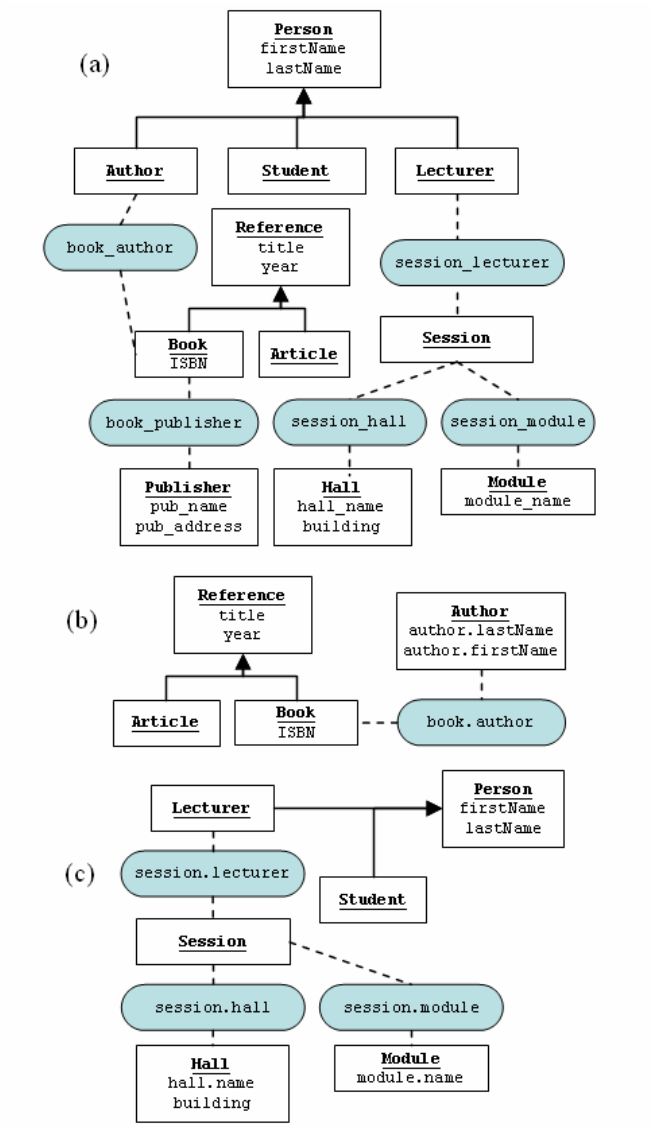

Figure 4. (a) An excerpt of reference ontology, (b)(c) excerpts of two local ontologies.

```
SELECT ?t
WHERE {
  ?b    ro:book_author    ?a .
  ?b    ro:title    ?t .
  ?a    ro:lastName     ?ln .
  ?a    ro:firstName    ?fn .
  ?s    ro:session_lecturer    ?l .
  ?l    ro:lastName     ?ln .
  ?l    ro:firstName    ?fn .
  ?s    ro:session_module    ?m .
  ?m    ro:module_name     "Database" .
}
```

Figure 5. Global query

After the decomposition phase (performed by querying web service), each sub-query is redelivered to its appropriate data provider. When a sub-query is received by a data provider, it is firstly rewritten in terms of the local ontology (by replacing each reference ontology component by its equivalent component in the local ontology). For the example, the rewritten sub-queries are shown in figure (6c,

6d). The new query is translated into SQL query over the local database.

```
SELECT ?ln ?fn ?t
WHERE {
  ?b  ro:title  ?t .
  ?b  ro:book_author  ?a .
  ?a  ro:lastName  ?ln .
  ?a  ro:firstName  ?fn .
}
```

(a) Sub-query1

```
SELECT ?ln ?fn
WHERE {
  ?s  ro:session_lecturer  ?l .
  ?s  ro:session_module  ?m .
  ?m  ro:module_name  "Database" .
  ?l  ro:lastName  ?ln .
  ?l  ro:firstName  ?fn .
}
```

(b) Sub-query2

```
SELECT ?ln ?fn ?t
WHERE {
  ?b  lo1:title  ?t .
  ?b  lo1:book.author  ?a .
  ?a  lo1:author.lastName  ?ln .
  ?a  lo1:author.firstName  ?fn .
}
```

(c) Rewritten sub-query1

```
SELECT ?ln ?fn
WHERE {
  ?s  lo2:session.lecturer  ?l .
  ?s  lo2:session.module  ?m .
  ?m  lo2:module.name  "Database".
  ?l  lo2:lastName  ?ln .
  ?l  lo2:firstName  ?fn .
}
```

(d) Rewritten sub-query2

Figure 6. Two sub-queries over local ontologies.

5.2. SPARQL to SQL translation

In order to perform this kind of translation, a suitable SQL statement is needed for every concept and property in the local ontology. These statements are used as bracketed (nested) SELECT statements to form the FROM clause of the final SQL query.

In the case where the local ontology is created by DB2OWL tool, the necessary SQL statements are automatically created by the query engine (using the generated mapping document which associates each ontology component with its equivalent database component, see section 2.2). In other cases, SQL statements have to be manually provided.

A statement that corresponds to a property has two selected columns representing the domain and the range of the corresponding property. These columns are given two aliases (C0 and C1 respectively). Such a statement typically retrieves the values of the pair <domain, range> of the property from the database. For example, the SQL statements for the terms mentioned in sub-quey2 are listed in figure 7.

When the translator receives a SPARQL query, it establishes a basic graph pattern (BGP)[4] of the SPARQL query as defined in [7]. For example, the BGP representing the sub-query2 is shown in figure 8. Then, each edge in this graph is associated with the suitable SQL statement (from those mentioned above) and a unique alias is generated for this statement. The start node of the edge is associated to the first selected column in the statement (C0) and the end node is associated to the second one (C1). The set of statements representing all the edges in the graph form the FROM clause of the final SQL query.

lo2:firstName

```
SELECT person.personId AS C0,
       person.firstName AS C1
FROM person
```

lo2:lastName

```
SELECT person.personId AS C0,
       person.lastName AS C1
FROM person
```

lo2:session.lecturer

```
SELECT session.sessionId AS c0,
       lecturer.lecturerId AS c1
FROM session, lecturer
WHERE session.lecturerId = lecturer.lecturerId
```

lo2:session.module

```
SELECT session.sessionId AS c0,
       module.moduleId AS c1
FROM session, module
WHERE session.moduleId = module.moduleId
```

lo2:module.name

```
SELECT module.moduleId AS c0,
       module.moduleName AS c1
FROM module
```

Figure 7. SQL statements for sub-query2 components.

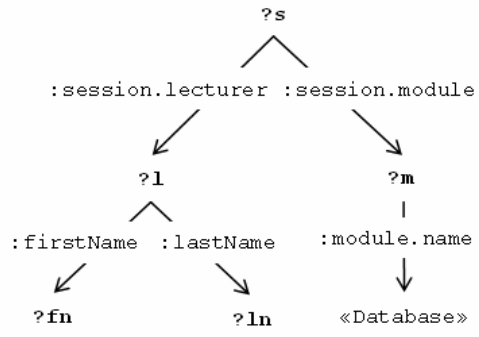

Figure 8. BGP for sub-query2.

When two (or more) edges share a variable node, then there equivalent statements will be joined depending on the columns representing the shared node and the direction of the edge (incoming or outgoing). If an edge has a literal end node, then the equivalent statement will be restricted using a logical condition in which the column equivalent to the node equals the literal value.

[4] A basic graph pattern is a directed graph BGP = (N, E), where N is a set of nodes representing subjects and objects, and E is a set of edges representing predicates.

The SELECT clause in SQL query will be the columns equivalent to variables in SELECT clause in SPARQL query. FILTER clauses are translated as conjunctive WHERE clauses. Logical and comparing operators in a FILTER clause are translated to equivalent operators in SQL and each variable mentioned in the FILTER is replaced by anyone of its equivalent columns from used statement. The final SQL query is shown in figure 9.

When the final SQL query is obtained, it is evaluated over the local database. The results are then formulated as SPARQL query results in XML format. These results are returned to the querying web service.

```
SELECT V0.C1 AS ln, V1.C1 AS fn
FROM
(SELECT person.personId AS C0,
       person.firstName AS C1
FROM person ) AS V0,
(SELECT person.personId AS C0,
       person.lastName AS C1
FROM person ) AS V1,
(SELECT session.sessionId AS C0,
       lecturer.lecturerId AS C1
FROM session, lecturer
WHERE
(session.lecturerId = lecturer.lecturerId)
) AS V2,
(SELECT session.sessionId AS C0,
       module.moduleId AS C1
FROM session, module
WHERE (session.moduleId = module.moduleId)
) AS V3,
(SELECT module.moduleId AS C0,
       module.moduleName AS C1
FROM module ) AS V4
WHERE (V0.C0 = V1.C0) AND (V2.C1 = V0.C0)
AND (V2.C0 = V3.C0) AND (V3.C1 = V4.C0)
AND (V4.C1 = 'Database')
```

Figure 9. Final SQL query equivalent to SPARQL sub-query2.

5.3. Results re-composition

Distributed sub-queries are now solved at the data provider level, and the results (formatted as XML documents) are returned to the query web service. These results are recomposed depending on shared variables. If two queries share the same variable, their results will be joined on the shared variable. The joined results will be projected on the variables selected be the user in the original query.

For example, sub-query2 has returned values for variables `?fn` and `?ln`, and sub-query1 has returned values for variables `?fn`, `?ln` and `?t`, therefore the results of two sub-queries are joined on variables `?fn` and `?ln`. In other words, sub-query1 results are restricted to those in which `?fn` and `?ln` has values in sub-query2 results, then they are projected on `?t` since the global query were asking for values of `?t`.

6. Discussion and Conclusion

We introduced OWSCIS (Ontology and Web Service based Cooperation of Information Sources), an architecture for the cooperation of information sources using ontologies and web services. In this architecture, information sources are wrapped to local ontologies to express their semantic. These local ontologies are mapped to a reference one agreed by the cooperation. Different web services are proposed to 1) interconnect the local sources to the local ontology, 2) map the local ontologies to the reference one, 3) query the cooperation through the reference ontology, and 4) suitably visualize the results of the query. In order to compare OWSCIS with existing systems we will use the features presented in section 2.

Currently, the only type of information sources for which OWSCIS provides tools is *relational databases*. In the future, OWSCIS will provide tools for non-relational databases and XML documents and schemas. The information sources are *dynamic* in OWSCIS, they may connect or disconnect from the system at any time. The mapping directory provide up-to-date information about which sources are connected at a given moment.

OWSCIS *architecture* is *mediator-based* where data providers play the role of (but not restricted to) wrappers , and the role of mediator is played by both mapping and querying web services. In order to enhance the system performance, we choose to store the local-to-reference ontology mappings locally at the data provider level, whereas a summary of these mappings is stored centrally at the mapping directory. This directory is updated when a data provider connect or disconnect to the system or when it changes its contents. In another performance aspect, we choose to centralize the query resolution in one web service and to synchronize the resolution of sub-queries.

Concerning the use of ontologies, our system belongs to the *hybrid ontology approach*. We use a local ontology for each information source, as well as a unique global ontology as a reference for the local ontologies. The advantage of wrapping each information source to a local ontology is to allow the development of source ontology independently of other sources or ontologies. Hence, the integration task can be simplified and easily support the addition and removal of sources.

In OWSCIS, we use *OWL language* for both the local ontologies and the reference ontology. OWL is a rich and expressive ontology language that provides additional vocabulary over RDF and RDF schema, as well as a formal semantics. OWL is currently a W3C recommendation.

We use *SPARQL* as *query language* in OWSCIS. We adopt the general methodology of query processing in integration systems (see section 2) and adapt it for decomposing SPARQL queries over the global ontology into several sub-queries over the

local ontologies. However, our approach involves a novel technique for translating queries from SPARQL to SQL.

SPARQL is an emerging W3C query language for RDF data. It may be used to query OWL ontologies. There are several projects [13] that propose to use Relational DBMSs to store and query RDF data using SQL and SPARQL. One of the most challenging problems in such projects is the translation of SPARQL queries into SQL. Cyganiak in [10] defines a relational algebra for SPARQL and outlines a set of rules to establish the equivalence between this algebra and SQL. Chebotko et al. [7] propose a basic graph pattern translation algorithm, BGPtoSQL, that translates a basic graph pattern to its SQL equivalent. Based on BGPtoSQL, they propose a semantics preserving SPARQL-to-SQL query translation algorithm for SPARQL queries that contain arbitrary complex optional graph patterns.

However, all proposed approaches address the translation of SPARQL queries into SQL queries over an RDF store database (in which there is only one table 'triples' with three columns: subject, predicate and object). The approach used in OWSCIS aims at translating a SPARQL query into an equivalent SQL query over any arbitrary relational database (whenever a suitable mapping exists).

OWSCIS architecture is partially implemented in Java using Jena API, JDBC, WordNet API and other available APIs. Implemented parts are: DB2OWL, SPARQL-to-SQL translator and inter-ontology mapping module. DB2OWL [9] is a tool which automatically generates a local ontology from a local relational database with the associated mappings. The inter-ontology mapping module is currently to be tested on significant ontologies and the SPARQL-to-SQL translator is still under development.

10. References

[1] Alexiev, V.; Breu, M. and de Bruijn, J. “Information Integration with Ontologies: Experiences from an Industrial Showcase,” John Wiley & Sons, 2005.

[2] Arens, Y.; Hsu, C.; Knoblock, C. A. Huhns, M. N. and Singh, M. P. (ed.) “Query Processing in the SIMS Information Mediator Readings in Agents,” Morgan Kaufmann, 1997, 82-90.

[3] Baader, F.; Horrocks, I. and Sattler, U. “Description Logics as Ontology Languages for the Semantic Web,” Mechanizing Mathematical Reasoning, 2005, 228-248.

[4] Bayardo, R.J. et al. “InfoSleuth: Agent-Based Semantic Integration of Information in Open and Dynamic Environments,” In ACM SIGMOD Record Vol. 26, No. 2 (June 1997), SIGMOD '97. Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA, 1997, 195-206.

[5] Berners-Lee T.; Hendler J. and Lassila O. “The Semantic Web,” Scientific American, May 2001.

[6] Buccella, A.; Cechich, S. A. and Brisaboa, N. “Ontology-Based Data Integration Methods: A Framework for Comparison,” Colombian Journal of Computation. , v.6, n.1, 2005.

[7] Chebotko, A.; Lu, S.; Jamil, H. M. and Fotouhi, F. “Semantics Preserving SPARQL-to-SQL Query Translation for Optional Graph Patterns,” 2006

[8] Corcho, O. and Gomez-Perez, A. “Evaluating knowledge representation and reasoning capabilities of ontology specification languages,” In Proceedings of the ECAI 2000 Workshop on Applications of Ontologies and Problem-Solving Methods, Berlin, 2000.

[9] Cullot, N.; Ghawi, R. and Yétongnon, K. “DB2OWL : A Tool for Automatic Database-to-Ontology Mapping,” SEBD, 2007, 491-494

[10] Cyganiak, R. “A relational algebra for SPARQL,” Technical Report HPL-2005-170. 2005.

[11] Goh, C.H., Bressan, S., Siegel, M. and Madnick, S. E. “Context Interchange: New Features and Formalisms for the Intelligent Integration of Information,” ACM Transactions on Information Systems, Vol. 17(3), 1999, 270–293.

[12] Gray, P.D.M. et al. “KRAFT: Knowledge Fusion from Distributed Databases and Knowledge Bases,” In Proceedings of the DEXA 1997, Toulouse, France, 1997, 682-691.

[13] “Mapping Semantic Web Data with RDBMSes,” http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/

[14] Mena, E.; Kashyap, V.; Sheth, A. P. and Illarramendi, A. “OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies,” Conference on Cooperative Information Systems, 1996, 14-25.

[15] Prud'hommeaux, E. and Seaborne, A. “SPARQL Query Language for RDF,” (Working Draft) W3C, 2007

[16] Stuckenschmidt, H.; Wache, H.; Vogele, T. and Visser, U. “Enabling technologies for interoperability,” In Visser, U., and Pundt, H., eds., Workshop on the 14th International Symposium of Computer Science for Environmental Protection, Bonn, Germany: TZI, University of Bremen, 2000, 35-46.

[17] Tamma, V. and Visser, P. “Integration of Heterogeneous Sources: Towards a Framework for comparing Techniques,” Proceedings of the AI*IA '98 Workshop on Techniques for Organisation and Intelligent Access of Heterogeneous Information, 1998

[18] Wache, H. et al., (ed.) “Ontology-based integration of information - a survey of existing approaches,” IJCAI--01 Workshop: Ontologies and Information Sharing, 2001, 108-117.

[19] Woelk, D. et al. “Using Carnot for Enterprise Information Integration,” Second International Conf. Parallel and Distributed Information Systems. January, 1993, 133-136.