

5-12-2012

## Semantic Integration of Coastal Buoys Data using SPARQL

Rakesh Kumar Gourineni

Follow this and additional works at: <https://scholarsjunction.msstate.edu/td>

---

### Recommended Citation

Gourineni, Rakesh Kumar, "Semantic Integration of Coastal Buoys Data using SPARQL" (2012). *Theses and Dissertations*. 3906.

<https://scholarsjunction.msstate.edu/td/3906>

This Graduate Thesis - Open Access is brought to you for free and open access by the Theses and Dissertations at Scholars Junction. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of Scholars Junction. For more information, please contact [scholcomm@msstate.libanswers.com](mailto:scholcomm@msstate.libanswers.com).

SEMANTIC INTEGRATION OF COASTAL BUOYS DATA USING SPARQL

By

Rakesh Kumar Gourineni

A Thesis  
Submitted to the Faculty of  
Mississippi State University  
in Partial Fulfillment of the Requirements  
for the Degree of Master of Science  
in Electrical Engineering  
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

May 2012

Copyright 2012

By

Rakesh Kumar Gourineni

SEMANTIC INTEGRATION OF COASTAL BUOYS DATA USING SPARQL

By

Rakesh Kumar Gourineni

Approved:

---

Nicolas H. Younan  
Department Head and James Worth  
Bagley Chair Department of Electrical  
and Computer Engineering  
(Major Advisor)

---

Roger L. King  
Giles Distinguished Professor  
Director of CAVS Department of  
Electrical and Computer Engineering  
(Co-Major Advisor)

---

James V. Aanstoos  
Associate Research Professor  
Geosystems Research Institute  
(Committee Member)

---

James E. Fowler  
Professor and Graduate Program  
Director of Department of Electrical  
and Computer Engineering

---

Sarah A. Rajala  
Dean of the Bagley College of Engineering

Name: Rakesh Kumar Gourineni

Date of Degree: May 11, 2012

Institution: Mississippi State University

Major Field: Electrical and Computer Engineering

Major Professor: Nicholas H. Younan

Title of Study: SEMANTIC INTEGRATION OF COASTAL BUOYS DATA USING SPARQL

Pages in Study: 47

Candidate for Degree of Master of Science

Currently, the data provided by the heterogeneous buoy sensors/networks (e.g. National Data Buoy center (NDBC), Gulf Of Maine Ocean Observing System (GoMoos) etc. is not amenable to the development of integrated systems due to conflicts in the data representation at syntactic and structural levels. With the rapid increase in the amount of information, the integration of heterogeneous resources is an important issue and requires integrative technologies such as semantic web. In distributed data dissemination system, normally querying on single database will not provide relevant information and requires querying across interrelated data sources to retrieve holistic information. In this thesis we develop system for integrating two different Resource Description Framework (RDF) data sources through intelligent querying using Simple Protocol and RDF Query Language (SPARQL). We use Semantic Web application framework from AllegroGraph that provides functionality for developing triple store for the ontological representations, forming federated stores and querying it through SPARQL.

## DEDICATION

I would like to dedicate this research to my parents, family members, and friends.

## ACKNOWLEDGEMENTS

I would like to express my heartfelt and sincere thanks to my major advisor Dr. Nicolas H. Younan for his continued support and guidance through my research and for his time to review the report. I would like to thank my co-major advisor Dr. Roger L. King for his help, guidance, and patience throughout and also for his valuable suggestions that helped me to complete this thesis successfully. I'm grateful to my committee member, Dr. James V. Aanstoos, for his support, encouragement, and valuable suggestions.

I would like to thank the Northern Gulf Institute (NGI), a NOAA cooperative institute for funding this project. I would also like to thank all the graduate students for their support and advices.

I shall always be thankful to my family and friends for their unconditional love and support throughout my Master's program

## TABLE OF CONTENTS

	Page
DEDICATION .....	ii
ACKNOWLEDGEMENTS .....	iii
LIST OF TABLES .....	vi
LIST OF FIGURES .....	vii
CHAPTER	
I.    INTRODUCTION .....	1
1.1    Background .....	1
1.2    Ontologies .....	3
1.3    Semantic Integration of Heterogeneous Data Sources.....	5
1.4    Triple Store .....	5
1.5    Federated Database .....	6
1.6    Motivation and Objectives .....	7
II.   LITERATURE REVIEW .....	9
2.1    RDF stores .....	9
2.1.1    Sesame .....	10
2.1.2    3-store .....	10
2.1.3    Jena .....	11
2.1.4    Parka .....	12
2.1.5    RDFLib/Redland.....	12
2.1.6    TAP .....	13
2.2    Querying Federated Databases .....	13
III.  METHODOLOGY .....	15
3.1    Semantic Web .....	15
3.2    Federated Database .....	16
3.2.1    Requirements of federated database management system.....	18
3.3    Overview of semantic integration approach .....	18
3.3.1    Semantic integration considerations .....	18



3.3.2	AllegroGraph .....	19
3.3.3	Sesame Java API.....	20
3.3.4	AllegroGraph Web View Framework.....	21
3.4	Architecture of a semantic integration approach .....	22
3.4.1	Creating a repository and Storing triples .....	24
3.4.2	Creating Federated Repository .....	25
3.4.3	Querying through SPARQL.....	25
3.5	protégé-OWL .....	26
3.6	Apache Tomcat .....	28
IV.	RESULTS .....	29
4.1	Find out the all the atmospheric properties measured by barometer in NDBC and GoMOOS ? .....	34
V.	CONCLUSION AND FUTURE WORK .....	43
5.1	Conclusion .....	43
5.2	Future Work .....	44
	REFERENCES .....	45

## LIST OF TABLES

TABLE	Page
1.1 Representation of data in a triple store [11] .....	6
3.1 Syntactical representation of NDBC and GoMOOS ontologies .....	17

## LIST OF FIGURES

FIGURE	Page
1.1 Representation of a federated database .....	6
1.2 Overview of the research.....	8
2.1 Example of RDQL query .....	11
3.1 Allegrograph Database overview .....	20
3.2 Snippet of Allegrograph Web View.....	21
3.3 Overview of the implementation.....	23
3.4 Snippet of a SPARQL query .....	26
3.5 Ontology representation of a domain in protégé-OWL editor .....	28
4.1 Creating NDBC triple store.....	30
4.2 NDBC triple store with triples imported.....	31
4.3 Creating GoMOOS triple store .....	32
4.4 GoMOOS triple store with triples imported.....	33
4.5 Representation of the Federated store formed from NDBC and GoMOOS .....	34
4.6 Results of SPARQL query to get the atmospheric properties measured by barometer.....	35
4.7 Results of SPARQL query to get the atmospheric properties measured by thermistor .....	36
4.8 Results of SPARQL query to get the atmospheric properties measured by seismometer.....	37
4.9 Results of SPARQL query to get the devices used in the coastal buoys .....	38

4.10	Results of SPARQL query to obtain the station ids from NDBC and GoMOOS .....	39
4.11	SPARQL query results depicts the parameters measured by NDBC and GoMOOS .....	40
4.12	Graphical View of the SPARQL CONSTRUCT query .....	41
4.13	Graphical View of the SPARQL CONSTRUCT query .....	42

# CHAPTER I

## INTRODUCTION

### **1.1 Background**

In World Wide Web, the term Web 2.0 deals with web applications which allow users to interrelate with each other [1]. All the blogs, wikis, and social networks come under Web 2.0. In addition to retrieving the information, Web 2.0 offers many useful characteristics such as rich user experience, participation of the user and dynamic content. For instance, users can get control over the data which is provided to them.

There are three parts in Web 2.0. The first one is Rich Internet Application which deals with how a particular application is transformed from a desktop application to an internet application with the help of Asynchronous JavaScript and XML (AJAX). The second is Service Oriented Architecture which explains how smaller applications integrate to become much richer applications [1]. Finally, the third one is the social web, in which end user is an imperative part which deals with the interaction of Web 2.0 with the end user. All the Web 2.0 applications are built with AJAX support so that it works in any browser. Besides this, a language, this is iterative and with good web services, must be used to build the applications so that they can be updated very easily. On the whole, the primary idea of Web 2.0 applications is sharing and integrating the data which results in valuable results when queried.

The World Wide Web (WWW) is a system of interlinked hyper text documents accessed through the internet. With the help of the WWW, humans can retrieve texts,

images, and other multimedia from the web pages [2]. It is the main rationale for making the Internet as a household application. But only humans can interact with the WWW which forms a great drawback besides its massive conquest. As computers do not have the ability to understand the meaning of the web pages, they cannot make conclusions from the information and compare different information sources. In addition to this we cannot use the information on the web on a large scale because the information can be processed by anyone with the absence of a global system [3]. Semantic Web provides the solution for this problem.

In order to make the computers do more useful work, a stack support namely the “Web of Data” was proposed by W3C[4]. This “Web of Data” is referred to as the Semantic Web which will help people in maintaining the databases such as building the data stores on the web, forming relations among the data, and querying across the data stores. The Semantic Web data refers to date, time stamps, chemical properties, titles etc. must be in standard format so that it can be easily reachable and manageable by the Semantic Web tools [5]. In addition to this, the relations among the data should also be present. The Semantic Web enables “intelligent” reasoning capabilities for web based systems by providing a meaningful structure to the web data.

Standardized XML-type data formats such as Resource Description Framework(RDF) and Ontology Web Language(OWL) are used to solve the heterogeneity problems and to achieve semantic interoperability on the Semantic Web. RDF is an XML based general purpose language used to describe information about the web sources in standard format [6]. OWL is a W3C standard XML based language which is used to provide additional vocabulary to the data along with the formal semantics [7]. When compared with RDF, OWL is a more expressive language because it adds more

vocabulary for representing the properties of the data, relations between the classes and more machine interoperability. So in order to provide semantic integration of the datasets on the Semantic Web, we define the datasets in ontologies.

## 1.2 Ontologies

Ontologies are metadata schemas which are described with vocabulary of concepts, machine-readable semantics, instances, properties etc. With the help of ontologies people and machine can communicate easily [8]. They provide specific vocabulary that is required by a particular application. In recent years, ontologies are used in various fields such as semantic Web services, semantic integration, knowledge management, electronic commerce, social networks, etc. For instance, the wind direction parameter, which is measured by sensors can be represented as a defined concept in an ontology. This is shown in Example 1.

```
<owl:Class rdf:about="#WindDirection">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:someValuesFrom rdf:resource="#StationID"/>
        <owl:onProperty>
          <owl:ObjectProperty rdf:about="#hasStationID"/>
        </owl:onProperty>
      </owl:Restriction>
    <owl:Restriction>
```

```

    <owl:hasValue
rdf:resource="http://sweet.jpl.nasa.gov/ontology/earthrealm.owl#OceanRegion"/>
    <owl:onProperty>
        <owl:ObjectProperty rdf:about="#hasMeasurement"/>
    </owl:onProperty>
</owl:Restriction>
<owl:Restriction>
    <owl:hasValue
rdf:resource="http://sweet.jpl.nasa.gov/ontology/space.owl#Direction"/>
    <owl:onProperty
rdf:resource="http://sweet.jpl.nasa.gov/ontology/space.owl#hasDirection"/>
    </owl:Restriction>
<owl:Restriction>
    <owl:someValuesFrom rdf:resource="#Sensors"/>
    <owl:onProperty>
        <owl:ObjectProperty rdf:about="#measuredBy"/>
    </owl:onProperty>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
</owl:equivalentClass>
<rdfs:subClassOf rdf:resource="#Wind"/>
</owl:Class>

```

Example 1: Snippet of Ontological Representation for Wind Direction Class



However, with the rapid increase in the amount of information, different enterprises that use different database systems to store and search the data and querying across single datasets will not provide relevant and useful information. Thus, there is a need for semantic integration of the heterogeneous data sources to resolve the heterogeneity problems.

### **1.3 Semantic Integration of Heterogeneous Data Sources**

In recent years, semantic integration is widely used in a variety of processing applications which are actively used in the web, database, and data mining communities. Semantic integration deals with solving the heterogeneity problems, which are initiated from the semantically heterogeneous data [9]. The heterogeneity problems include the modeling of complex relations in different sources, matching of data definitions in ontologies, and the reconciliation of inconsistencies. So, there is a need for semantic integration of heterogeneous data sources.

In order to store the ontology files of different organizations, we need a flexible and efficient database which can handle very large datasets and can perform efficient queries to retrieve the information about the devices and also certain properties [10]. The ontology data is represented as RDF triples that forms a RDF graph which can be stored in RDF triple stores.

### **1.4 Triple Store**

A Triple store is a purpose built database which is used to store the RDF metadata in the form of triples [11]. A triple store can consist of large number of triples which is like a relational database and can be queried through RDF query languages such as Simple Protocol and RDF Query Language (SPARQL), RDF Data Query Language

(RDQL) etc. More than one billion triples can be stored into one single triple store. The data in the triple stores is stored in the form of subject-predicate-object like “Sam is 45”. The triple store representation is represented in Table 1.1. The triple store implementations are done on the databases namely 3store, 4 store, Allegrograph etc. [12].

Table 1.1 Representation of data in a triple store [11]

Subject(Resource URI)	Predicate(Property URI)	Object(Entity value)
(...)	(...)	(...)

### 1.5 Federated Database

The term database system refers to the software database management system which manages one or more databases. In the same way, a federated database system is a collection of databases which are syntactically different. A federated store is structured by the amalgamation of heterogeneous database systems. As the federated store consists of diverse information sources, heterogeneities arise. Naming conflicts, data conflicts, metadata conflicts, and domain conflicts are some of the heterogeneities [13]. A federated database is represented in Figure 1.1.

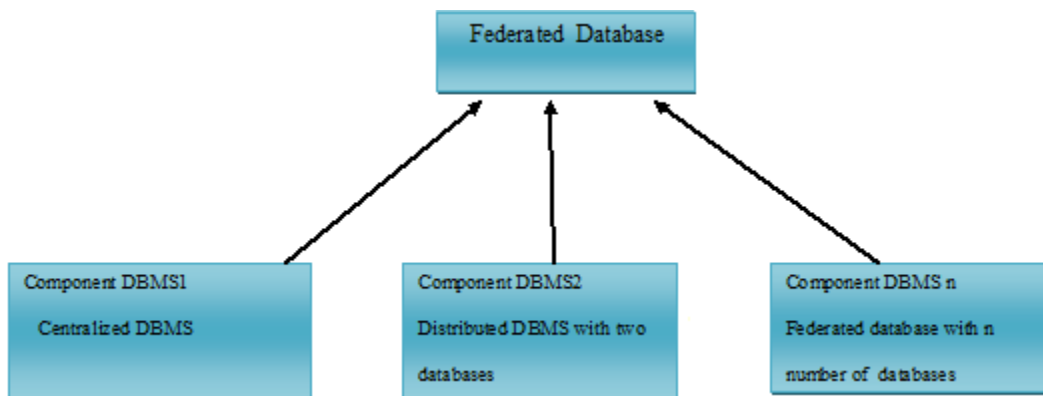


Figure 1.1 Representation of a federated database

## 1.6 Motivation and Objectives

Ocean sensor networks such as GoMOOS (Gulf of Maine Ocean Observing Systems) [14] and NDBC (National Data Buoy Center) [15] provide real time or near real time sensor data. Due to the heterogeneous and non semantic nature of the sensor data it prevents the semantic interoperability in ocean sensor networks. So the two ontologies cannot communicate due to the heterogeneities in their properties. NDBC and GoMOOS both maintain the same Marine/meteorological data and consist of the same sensors but the vocabulary representations are different. For instance the wind direction in NDBC is represented as “Wind Direction” but in GoMOOS it is represented as “Wind\_Direction”. Though NDBC and GoMOOS refer to the same parameter semantically they are represented in a different manner.

In this research, to overcome the heterogeneities of the data sources, intelligent querying across different knowledge bases is required. Federated database aids users to query multiple datasets at the same time and these multiple responses will be standardized to one result set. The National Data Buoy Center (NDBC) and the Gulf of Maine Ocean Observing System (GoMOOS) Ontology Web Language files (OWL files) are stored as RDF triple stores in Allegrograph database [16]. A federated data-store is formed in Allegrograph by the amalgamation of the NDBC and GoMOOS OWL files. The querying of these knowledge bases is achieved via SPARQL which is a RDF query language. The main objective of the query languages is to make the machine to understand a particular application. So if we want the information regarding the wind direction parameter, then, the result must contain the values from NDBC and GoMOOS. The preliminary implementation of the proposed intelligent querying using SPARQL is shown in Figure 1.2.

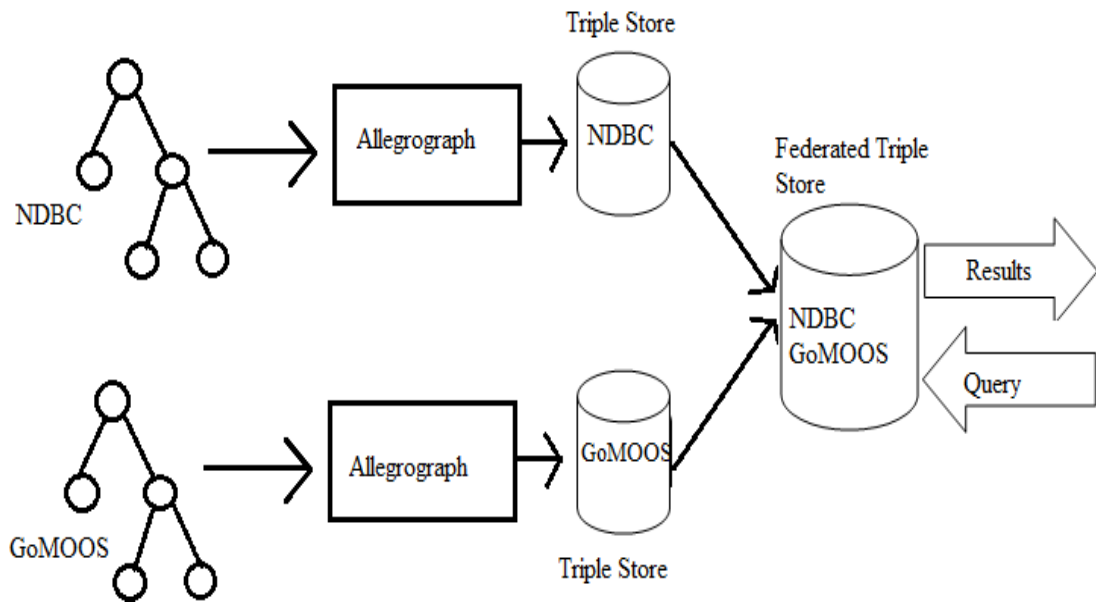


Figure 1.2 Overview of the research.

## CHAPTER II

### LITERATURE REVIEW

In the last few years, wide research is going on in semantic integration to facilitate interoperability between different systems. As querying on single datasets does not retrieve efficient results, the research in semantic integration include techniques for matching database schemas and answering queries using multiple sources of data. Moreover the research in semantic integration became prominent as it is actively used in several fields such as databases, integration of the information and ontologies. In this section we will discuss about the research work in storing the RDF metadata as triple stores and querying across the federated databases using SPARQL, which is an RDF query language.

#### **2.1 RDF stores**

For the query processing, storage engine is the core component, as it comprehends the organization of the systems and the implementation of SPARQL. Large amount of (subject, predicate, object) triples are stored by the triple stores [17]. Examples of triple store implementations are Sesame [18], 3-store [19], Jena [20], Allegrograph [16] and OpenLink Virtuoso [21]. All these triple store implementations use the relational model. In addition to these we have other RDF storage implementations such as Parka [22], RDFLib/Redland [23] and TAP [24].

### **2.1.1 Sesame**

In [18], architecture Sesame was developed for the efficient storage and querying of large quantities of metadata in RDF and RDF Schema. The design and implementation of Sesame is independent of other specific storage devices, so that it can be stored on top of variety of storage devices such as triple stores or relational databases or object oriented databases. A support for concurrency control, independent export of RDF and RDFS information and RQL(RDF Query Language), a query language of RDF is provided by Sesame.

From [18] we can infer that the most important feature of Sesame is its abstraction, which helps Sesame to port to different repositories such as relational databases, RDF triple stores and remote services on the web. Moreover Sesame is a server based application and it is used as a remote service for managing the RDF data on the semantic web.

### **2.1.2 3-store**

In paper [19] 3 store and RDF engine is described which efficiently supports RDF and RDFS over relatively large RDF knowledge bases using a relational database backend to perform queries. The main aim of the work is to design and implement a system for scalable storage of RDF data. The 3-layer optimization model of Sophia[25] is used for the multi-level optimization and also enables efficient RDBMS storage. Both classes and instances are stored using unified storage mechanism. The 3 store's layers are characterized as RDF syntax, RDF representation and relational databases system.

In the database structure of 3-store, using hash of the resource URIs and literal values as a foreign key the schema is normalized. Resources and literals are stored in separate tables with a hash of their values used as the primary key. Both resources and

literals use hashing function so that the triple table contains a flag to indicate whether the object of the triple is a literal or a resource. The 3-store RDQL query transforms an RDQL query into an SQL query over the underlying RDBMS representation of the RDF data. RDQL is the more common RDF query language. There are other graph matching query languages with comparable syntaxes and capabilities. The example of query expressions is given in Figure 2.1

```
SELECT ?a
WHERE (<r:1>, <p:a>, ?a),
      (?a, <p:b>, "foo"),
      (?a, <p:b>, "bar")
```

Figure 2.1 Example of RDQL query

### 2.1.3 Jena

Jena [20] is a leading semantic web tool kit for manipulating RDF models which has been developed by Hewlett-Packard Labs. In [20] the architecture of Jena is given. The heart of Jena architecture is RDF graph, which is a set of nodes. There are three layers in this architecture. The first one is the graph layer, which stores the triples as the universal data structure. This layer is based on the RDF Abstract Syntax [26] and it implements triple stores, both in memory and backed by persistent storage, read-only views of non-triple data as triples and virtual triples corresponding to the results of inference processes over some further set of triples as premises. The second is the model layer, which gives a richer set of methods for operating on both graph and nodes within the graph. The third is the enhgraph layer, which provides the multiple simultaneous views. This layer provides an extension point for providing views of graphs and views of

nodes within a graph. The needs of the Model and ontology API are generalized and thus making the design decision stateless. To provide polymorphic objects within the enhgraph, Java single inheritance model is used, thus allowing multiple inheritances.

In Jena the queries are executed against the graphs which have multiple statement tables. There is a handler for each statement table to convert between the graph view of Jena and the tuple view of SQL. To evaluate the triple pattern, the query processor passes the pattern, in turn, to each table handler for evaluation. But the drawback with Jena is that it is not suitable for storing large volume of data that we require.

#### **2.1.4 Parka**

Parka [22] is an inferencing database written over a custom relational database back-end; the Parka version has been modified to support RDF. Query execution is handled by the relational engine, as in 3-store, though the table layout is significantly different, so comparing the translation engine algorithms is beyond the scope of this document. According to its documentation, Parka has an upper limit on its knowledge base size of around 2.5 million triples, which appears to be due to the structure of the relational indexes, and which makes it inappropriate for the scale of data with which we are working.

#### **2.1.5 RDFLib/Redland**

The Redland [23] suite appears to be capable of storage of large RDF graphs, but currently has no graph matching query facility, so is unsuitable for our purposes. Adding a query mechanism to Redland was considered, but because it does not use a DBMS back-end it would have been considerably more effort to implement than to port the existing 3-store code to another environment. However RDF parser from Redland



provides a C API for extracting the triples from RDF/XML and RDF/Ntriples documents. It abstracts the complexities of the RDF syntax, and removes the implementation burden from our core goal of building a scalable, persistent RDF knowledge base.

### **2.1.6 TAP**

TAP [24] is an RDBMS backed RDF knowledge base which has appeared since the start of the development of 3-store version 2. It provides an Apache module, TApache, which allows remote access to the stored RDF, much like 3-store. However, TApache provides no graph matching query interface such as RDQL, and provides only the direct triple matching method, GetData, much like the low level librdfsql access methods used internally by all the RDF stores.

## **2.2 Querying Federated Databases**

In [26] the architecture of an end-to-end semantic search engine that uses a graph data model to enable interactive query answering over structured and interlinked data collected from many disparate sources on the web. The architecture of Semantic Web Search Engine requires the components such as

- (i) **Crawler.** Used to store the web documents and this architecture also uses multicrawler, which is a pipelined crawling architecture, syntactically transforms the data from different sources into single federated store for easy integration into a semantic web system.
- (ii) **Object Consolidator.** Within RDF, URIs(Uniform Resource Identifier) are used to uniquely identify the entries. But on the web due to the lack of URIs cause conflicts in the entities. We can eradicate this by merging equivalent entities representing a particular person having the same values for an email property.

- (iii) Query Processor. The query processor creates and optimizes the logical plan for answering the queries for all the databases stored in a federated store.
- (iv) User interface. To provide user-friendly search, query and browsing over the data indexed, the architecture provides a user interface which is the human access point to the Semantic Web Search Engine.

In [27] an approach known as the federated database approach, allows the applications to access data across several heterogeneous databases as if they are accessing a single database without changing the state of the individual databases. They took the example of the queries related to airlines and aircraft classes. The steps followed are formulation of integration policy, schema transformation, conflict identification, conflict resolution, schema merging, and querying on the federated database.

In [28] a new schema was proposed to store, index and query RDF data. Graph feature of RDF data is taken into consideration which might help to reduce the join costs on the vertical database structure. The contributions of the paper are graph portioning of RDF triples, signature indexing and SQL rewriting.

In our approach we created a federated database by using Allegrograph and queried it through SPARQL.

## CHAPTER III

### METHODOLOGY

#### **3.1 Semantic Web**

For the management and exploitation of the web data semantic web symbolizes a vision for a new era[29]. In the present World, the World Wide Web (WWW) consists of a large number of web data. The problem with the rapid growth of web data is that it is not useful to use it on a large scale as there is no global system for publishing the data. For instance, if we look at the information regarding weather data, sports, television guides, etc., there is a large number of HTML (Hyper Text Markup Language) files, but it is not easy to use in the way that one might use [30]. This web of data is referred to as the semantic web which will help people in maintaining the databases such as building the data stores on the web and forming relations among the data and querying across the data stores. Therefore, it is very tricky to find appropriate answers for specific questions from users. In order to make computers do more useful work, a stack support namely “Web of Data” was proposed by W3C. In recent years the innovations in the semantic web made it an efficient way of representing the WWW data. The semantic web enhances the web data for better understanding and responding to the user queries. The semantic web data refers to date, time stamps, chemical properties, titles, etc., must be in a standard format so that it can be easily reachable and manageable by the semantic web tools [30]. In addition to this, the relations among the data should also be present. The

semantic web enables “intelligent” reasoning capabilities for the web based systems by providing a meaningful structure to the web data.

The semantic web is based on a set of XML (Extensible Markup Language) languages and RDF that can be used to markup the content of web pages. The effectiveness of the semantic web depends on ontologies. Ontology provides specific vocabulary that is required by a particular application. With the help of ontologies people and machine can communicate easily[31]. The main vision of the semantic web is the transformation of the web into an Internet wide knowledge representation system, in which web pages provide information and ontologies provide the conceptual framework needed to interpret that information [32]. The semantic web depends on ontologies to give the meaning of the data. The semantic web has the potential to provide the web services infrastructure with the semantic information that it needs. It also provides formal languages and ontologies to reason about service descriptions, message content, business rules, and relations between these ontologies. The semantic web transforms the web into a repository of computer readable data, and the web services provide the tools for the automatic use of that data.

### **3.2 Federated Database**

In recent years the conventional file processing systems used by different organizations have been replaced by Database Management Systems (DBMS)[33]. Today every organization has several different DBMS and databases and their applications. But these DBMS are not decision support, overall control, and distributed. So, even though DBMS eradicate the limitations of the conventional file processing systems, there is a need for the organizations to look towards the heterogeneous distributed database

scenario. This scenario deals with bounding large number of DBMS within a network [33]. A heterogeneous distributed database management system can be better explained with an example. Consider two different ontological representations, NDBC and GoMOOS which maintain the same sensor data, and differ in their syntactical representations. For instance wind speed, air temperature, water temperature, wind direction, wind gust and wave height are represented in differently as shown in Table 3.1.

Table 3.1 Syntactical representation of NDBC and GoMOOS ontologies

X	GoMOOS
Wind speed	Wind_speed
Air Temperature	Air_Temperature
Water temperature	Sea_Surface_Temperature
Wind direction	Wind_direction
Wind gust	Wind_gust
Wave Height	Significant_wave_height
Dominant wave period	Dominant_wave_period

A federated database approach proposed by Hammer and Mcleod[33] augments the accessibility of the heterogeneous database systems and allows applications to access global data. Each local database in the federated database is considered as a logical component which is lashed with one or more federated schemas.

### **3.2.1 Requirements of federated database management system**

In this section, we will look at the main requirements of a federated database management system [33].

1. A federated database system must consist of a large number of heterogeneous databases so that the querying can be done across a single federated database.
2. Each and every single database in the federated database must be accessible using any of the query languages.
3. There should not be any changes in the existing data.
4. The federated database system must be feasible enough to add new databases into it.
5. The most important aspect of a federated database system is its performance. Its performance must be almost equal to that of single database.

A federated database system must meet all the above requirements. In our research we built a federated store which is an amalgamation of the NDBC and GoMOOS systems. To query across this federated database, we used SPARQL query language.

### **3.3 Overview of semantic integration approach**

In this section the methodology and the system development tools used in the semantic integration approach is discussed.

#### **3.3.1 Semantic integration considerations**

For integrating the existing databases the major challenge lies in constructing a global database which satisfies the requirements mentioned in section 3.4.1. When we integrate heterogeneous databases there is a need to discover the hidden relationships

which do not appear in individual databases. Many conflicts arise while integrating the heterogeneous databases [33]. These are

1. Name Conflicts - , refers to representing the same data with their synonyms. For instance the concept of atmospheric pressure in the NDBC ontology is equivalent to barometric pressure in GoMOOS ontology.
2. Different representation conflicts - , arise when syntactically the same data is represented with the same constructs such as denoting an entity s, an entity in one database, and as an attribute in other databases.
3. Conflicts in application semantics - , arise due to different insights of different users.

Therefore it is of utmost important to eradicate these conflicts so that we can achieve semantic interoperability between the heterogeneous databases.

### **3.3.2 AllegroGraph**

AllegroGraph is a semantic web application framework database that can store ontology files in the form of triples. By querying these triples through various query APIs like SPARQL and Prolog, we can obtain the desired results, which are intended by the user. Figure 3.1 represents the block diagram of an AllegroGraph database. It consists of assertions (triples) that have five fields namely subject, predicate, object, graph, and id [16]. The string dictionary manages these strings of arbitrary sizes (subject, predicate, object, and graph) and UPIs and prevents duplication. AllegroGraph uses indices that contain the assertions and additional information for boosting the query process. It can perform freetext searching in the assertions using its free text indices, and also records the deleted triples.

In our research we used Java client distribution provided by Allegrograph. The java client uses java sesame API(Application Programming Interface). The communication with the java client is done through HTTP port.

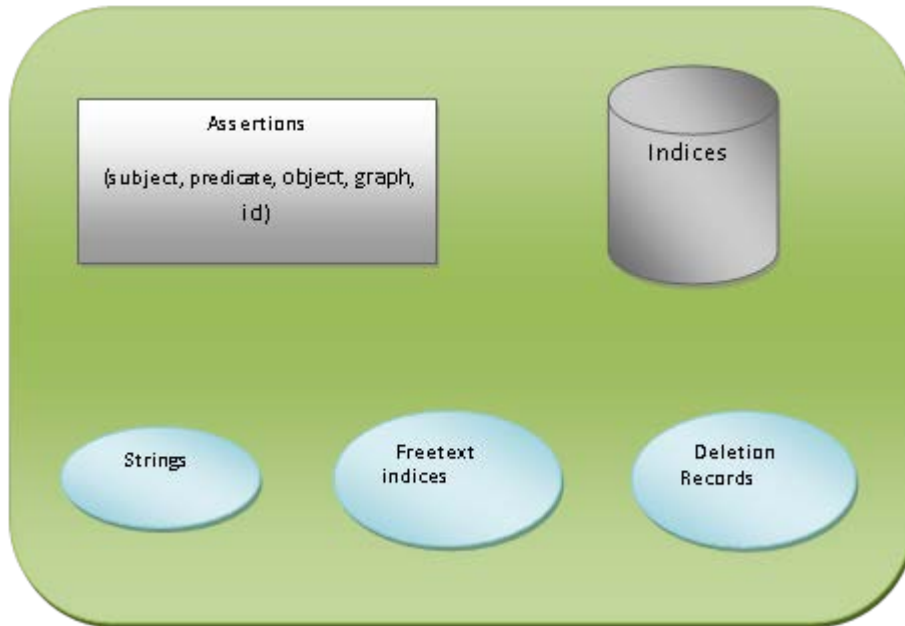


Figure 3.1 Allegrograph Database overview

### 3.3.3 Sesame Java API

Sesame is an open source Java framework which is used for querying, analyzing and managing the RDF data [34]. It can be used as a database for RDF data or as a Java library for applications that need to work with RDF internally. Sesame provides the necessary tools to parse, interpret, query and store all the information in a separate database or on a remote server [35].

Sesame supports two query languages namely SeRQL(Sesame RDF query language) and SPARQL. Albaba is an API of Sesame which allows mapping Java classes onto the ontologies so that the Java source files can be generated. Sesame's API offers a



stackable interface which helps in abstracting the storage engine from the query interface. Through sesame API many other triple stores can be used including Allegrograph, Mulgara and Virtuoso Universal server. In this study we used sesame java API with Allegrograph database and querying is done in SPARQL.

### 3.3.4 AllegroGraph Web View Framework

The AllegroGraph Web View[36] is a graphical interface for managing and querying the AllegroGraph triple stores. The HTTP (Hypertext Transfer Protocol) interface of AllegroGraph is used by the AllegroGraph Web View. A Snippet for the AllegroGraph Web View is shown in Figure 3.2.

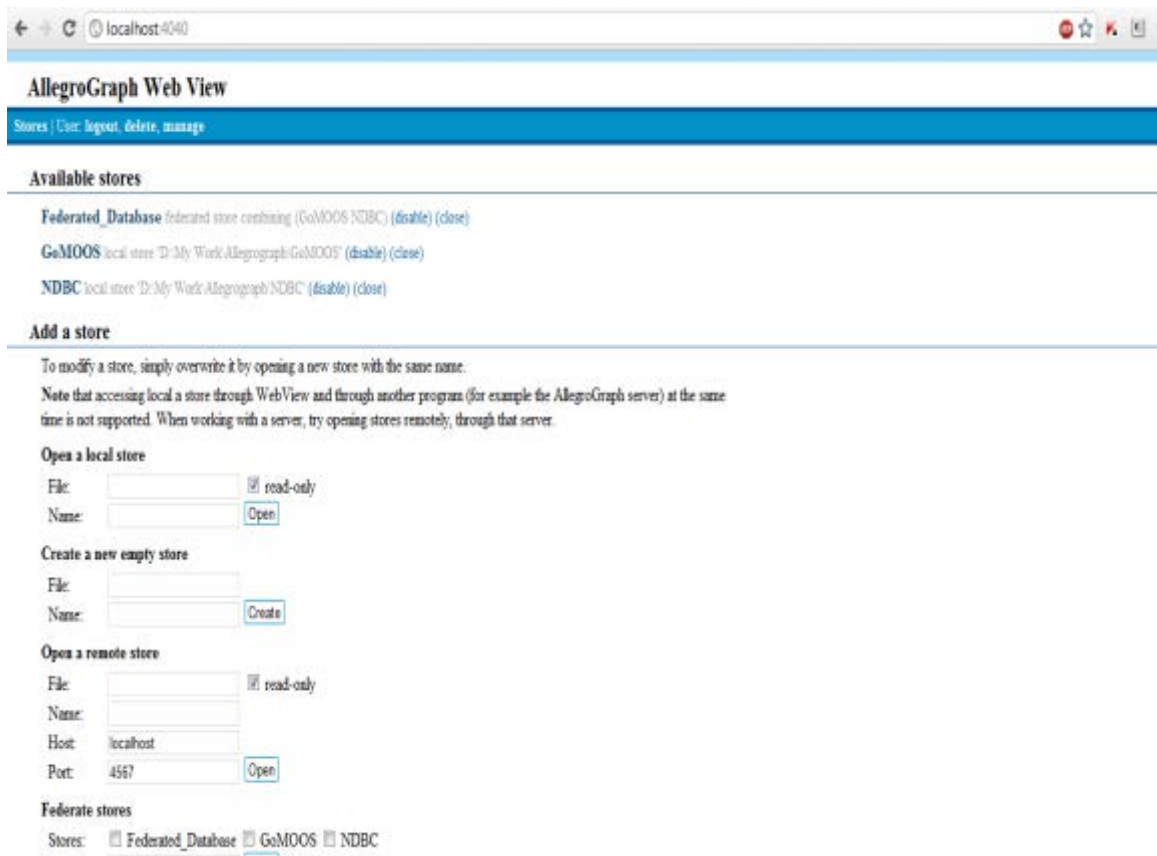


Figure 3.2 Snippet of Allegrograph Web View.

Using the AllegroGraph Web View, we can create individual triple stores by loading RDF data into a repository and browse the available catalogs and repositories[36]. We can create federated stores and issue SPARQL and Prolog queries on the federated stores. The AllegroGraph Web View helps us to view and add namespaces. A SPARQL-style notation is used for RDF resources throughout the interface.

### **3.4 Architecture of a semantic integration approach**

The basic idea behind the semantic integration approach is described as follows: Suppose there are two information sources, NDBC and GoMOOS, with their respective ontologies, describing the same domain but differ in naming convention as mentioned in Table 1. For instance, the concept of atmospheric pressure in the NDBC ontology is equivalent to barometric pressure in GoMOOS ontology. If we want information regarding the number of devices present at the coastal buoys then the result must show the devices from both NDBC and GoMOOS. The architecture of the semantic integration approach used in our research is illustrated in Figure 3.3.

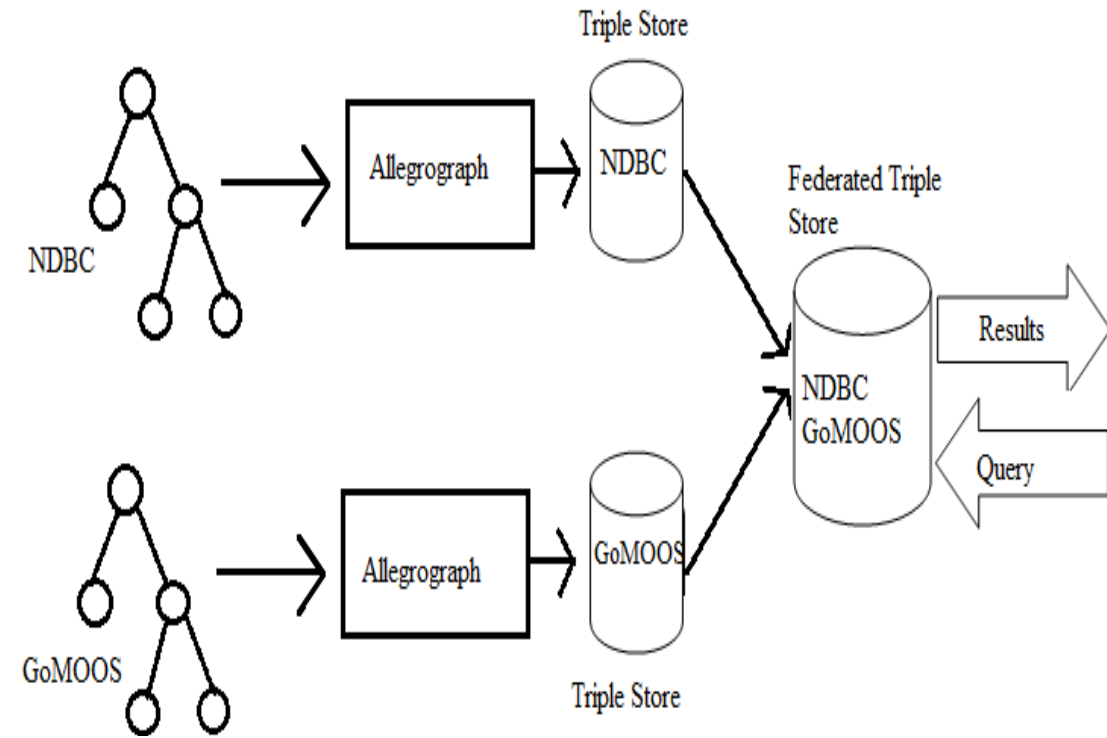


Figure 3.3 Overview of the implementation.

AllegroGraph[16] uses AllegroGraph Webview Framework [36], which is a graphical interface through which we can manage and query the AllegroGraph triples.

The steps involved in the methodology can be described in three steps as follows:

1. In the first step, the two ontology files, NDBC and GoMOOS which consist of RDF metadata are loaded into the Allegrograph database as different individual repositories(triple stores) , thus forming individual triple stores of NDBC and GoMOOS.
2. In the second step, these individual triple stores are combined to form a federated database. As querying on a single dataset is very easy and it will not lead to useful information retrieval, we need a federated database.
3. In the third step, we use SPARQL query language, which is an RDF query language to query on the federated database. This yields knowledgeable results.

Before querying the federated database, we need to discover the relations between these two datasources.

### 3.4.1 Creating a repository and Storing triples

Firstly start the Allegrograph server which is done by the username and password assigned to the server. Once we initialize the server a client-side server object is created, which can access the available repositories in the Allegrograph server by calling `listCatalog()` method. We use `CreateRepository()` method in order to create a client-side repository object, which is used to open our desired repository. We create a repository named NDBC. This repository is initialized using `initialize()` method. We will get a warning method if we initialize the same repository twice. Once the repository is initialized we need to check the type of indices to store. To quickly identify the block of triples, Allegrograph uses a set of stored indices which matches a specific query pattern. The default set of indices are `spogi`, `gspoi`, `posgi` and `I` where

- S represents Subject URI
- P represents Predicate URI
- O represents Object URI
- G represents Graph URI
- I represents triple identifier

The information of the index is denoted by the order of the letters. For instance `posgi` index represents the triples stored first by predicate, then by object, then by subject and finally by graph. The fifth column in the index denotes the triple number. Thus the repository will consist of only the indices which we need. Now, the NDBC repository must be loaded with the triples. The Java sesame API client loads the triples in N-triples format or RDF/XML format. In our implementation we stored our triples in RDF/XML format. We use `addFile()` method to store the RDF data from the ontology files of

ndbc.owl, loading the NDBC repository with the triples. In the same way GoMOOS repository is created and it is loaded with the RDF data from gomoosont.owl.

### **3.4.2 Creating Federated Repository**

Allegrograph aids us to combine the repositories and search them in parallel. This can be done by querying a single federated repository which will distribute the queries to the secondary repositories and combines the result. The federated repository can be created as follows.

- Open NDBC repository which is loaded with the RDF data from ndbc.owl.
- Open GoMOOS repository which is loaded with the RDF data from gomoosont.owl
- Call federate( ) method to create a federated repository.

### **3.4.3 Querying through SPARQL**

RDF is a directed, labeled graph data format for representing the information in the web [37]. The information in the social networks, personal information, and metadata are represented in RDF. Many query languages were proposed to retrieve the information from the RDF files [38]. But most of the query languages such as RQL (Resource Description Framework Query Language) and RDQL (Resource Description Framework Query language) are restricted to a single value, format, and type of information and do not enable for data sharing and merging. To overcome these limitations and to meet the user cases and requirements, a SPARQL query language was proposed. SPARQL was developed as a query language for RDF by W3C and querying is done by triple patterns, conjunctions, and disjunctions [39].

A SPARQL query language consists of semantic annotations and descriptions when compared with the existing standard sensor web languages, which allows the sensor

data to be understood and processed in a meaningful way by a variety of applications with different purposes. The four different query forms of SPARQL are the SELECT query, which pulls out the values into a table from a SPARQL endpoint, the CONSTRUCT query, which extracts the information and converts them into valid RDF, the ASK query, which provides True/False results for a query, and the DESCRIBE query, which helps to pull an RDF graph [39]. In our research, we used SPARQL query language to query across the federated database of NDBC and GoMOOS and to retrieve the results from the federated database formed by combining NDBC and GoMOOS. A sample SPARQL query used is shown in Figure 3.4.

```
SELECT DISTINCT ?StationID ?Stations
WHERE
{
base:wind_gust1 base:hasStationID ?StationID.
base:air_temperature_1 base:hasStationID ?StationID.
base:WindGust_1 base:StationID ?Stations.
base:AirTemperature_1 base:StationID ?Stations.
}
```

Figure 3.4 Snippet of a SPARQL query

### 3.5 protégé-OWL

The protégé-OWL editor is a software platform that allows users to create and manipulate OWL ontologies. A set of actions which are implemented are supports loading, saving, editing, and visualizing OWL ontologies. The users can also import

existing ontologies into the project by means of protégé plugins, which appear as tabs. It also supports the SPARQL (SPARQL Protocol and RDF Query Language) query language that allows users to retrieve the desired data by running queries on the ontology knowledge base.

In this application, ontologies representing knowledge of the coastal domain for different information sources are built in an Ontology Web Language using a protégé-OWL editor. Once a new ontology project is created, tabs, such as OWLClasses, properties and Individual tabs, are used to add classes, individuals, and properties to the ontology as per the application needs. Figure 3.5 shows an OWL ontology representing the domain knowledge in terms of classes, properties, and individuals with a protégé editor.

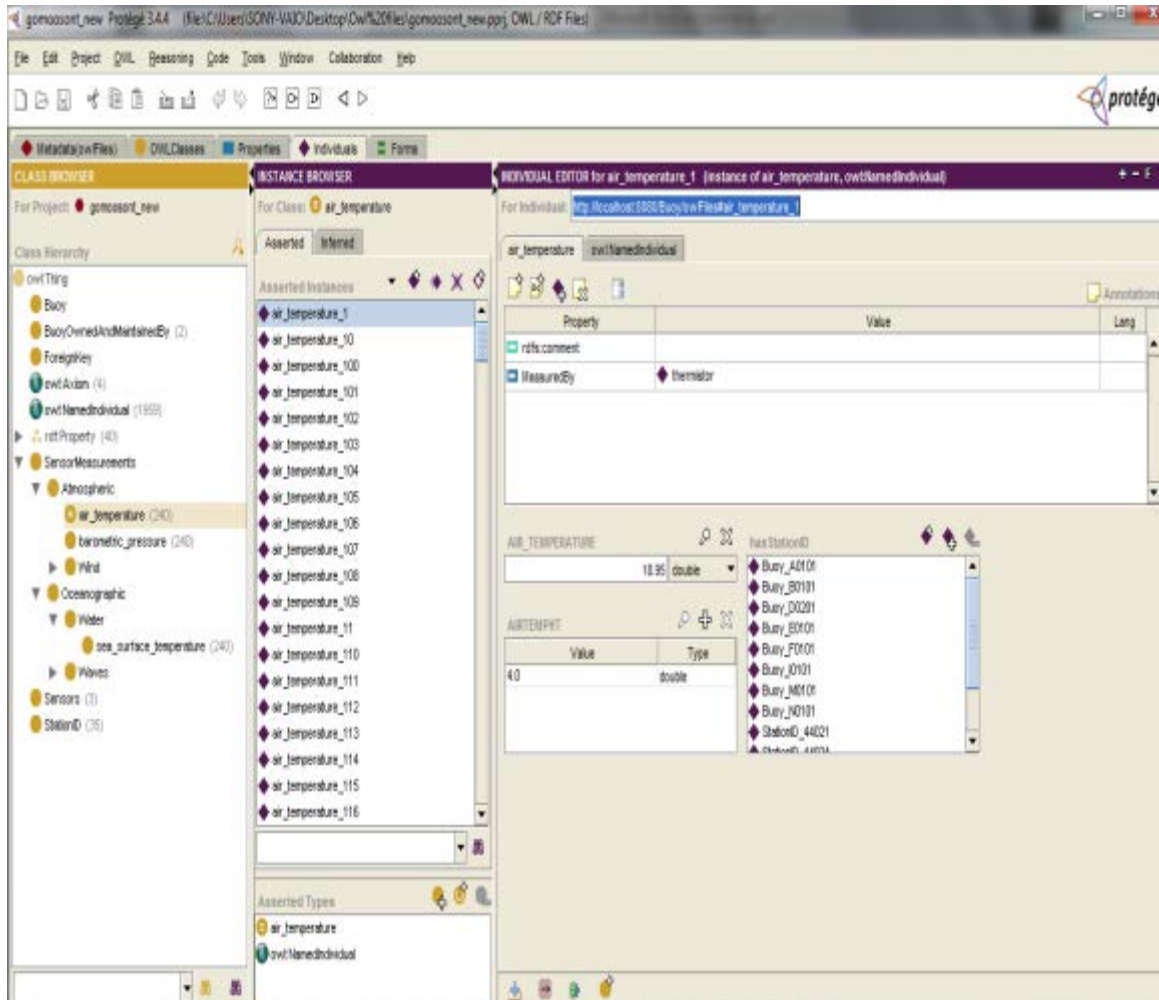


Figure 3.5 Ontology representation of a domain in protégé-OWL editor

### 3.6 Apache Tomcat

An Apache Tomcat is a servlet container developed by the Apache Software Foundation (ASF) [40]. The Tomcat implements the Java Servlet and the JavaServer pages (JSP) technologies. It provides a pure Java HTTP web server environment for the Java code to run. The Tomcat is used as the container for almost all the blocks of the architecture. The ontology files are deployed in Tomcat as a web archive (WAR) file. The Tomcat is responsible for serving the request/response from the client.



## CHAPTER IV

### RESULTS

In this chapter, the results for the implementation of the semantic integration framework on coastal buoys data are presented. The graphical user interface for the semantic integration approach is developed using the Allegrograph Web View. The querying in the AllegroGraph Web View can be done in SPARQL or Prolog. In this study we have used SPARQL. In the AllegroGraph Web View, we first create individual triple stores of NDBC and GoMOOS. Then, we form a federated database, which is formed by the amalgamation of the two individual triple stores. Figures 4.1-4.13 show the procedure for the implementation. The results will be retrieved by querying the federated database formed by combining NDBC and GoMOOS.



Figure 4.1 Creating NDBC triple store



Figure 4.2 NDBC triple store with triples imported

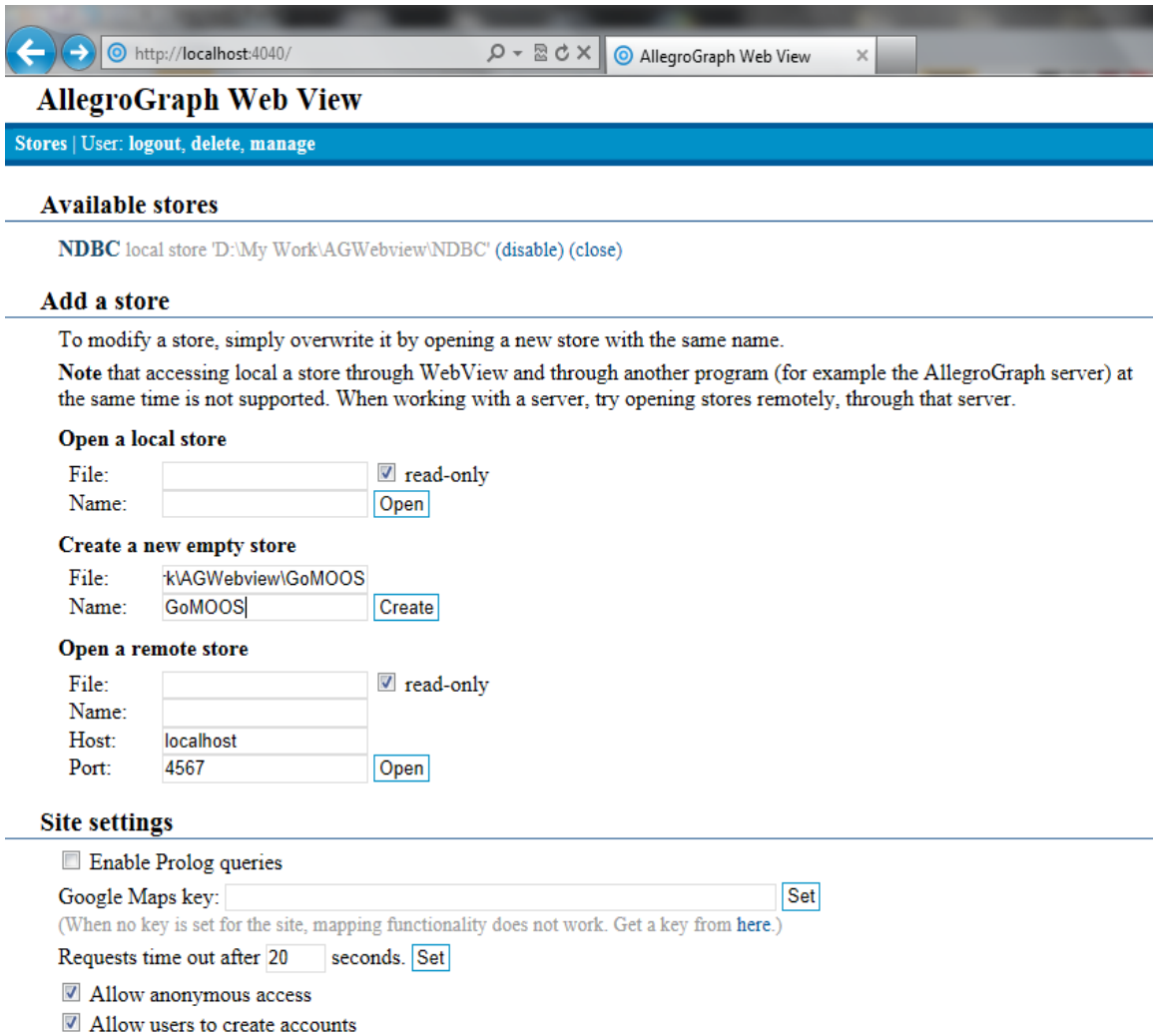


Figure 4.3 Creating GoMOOS triple store

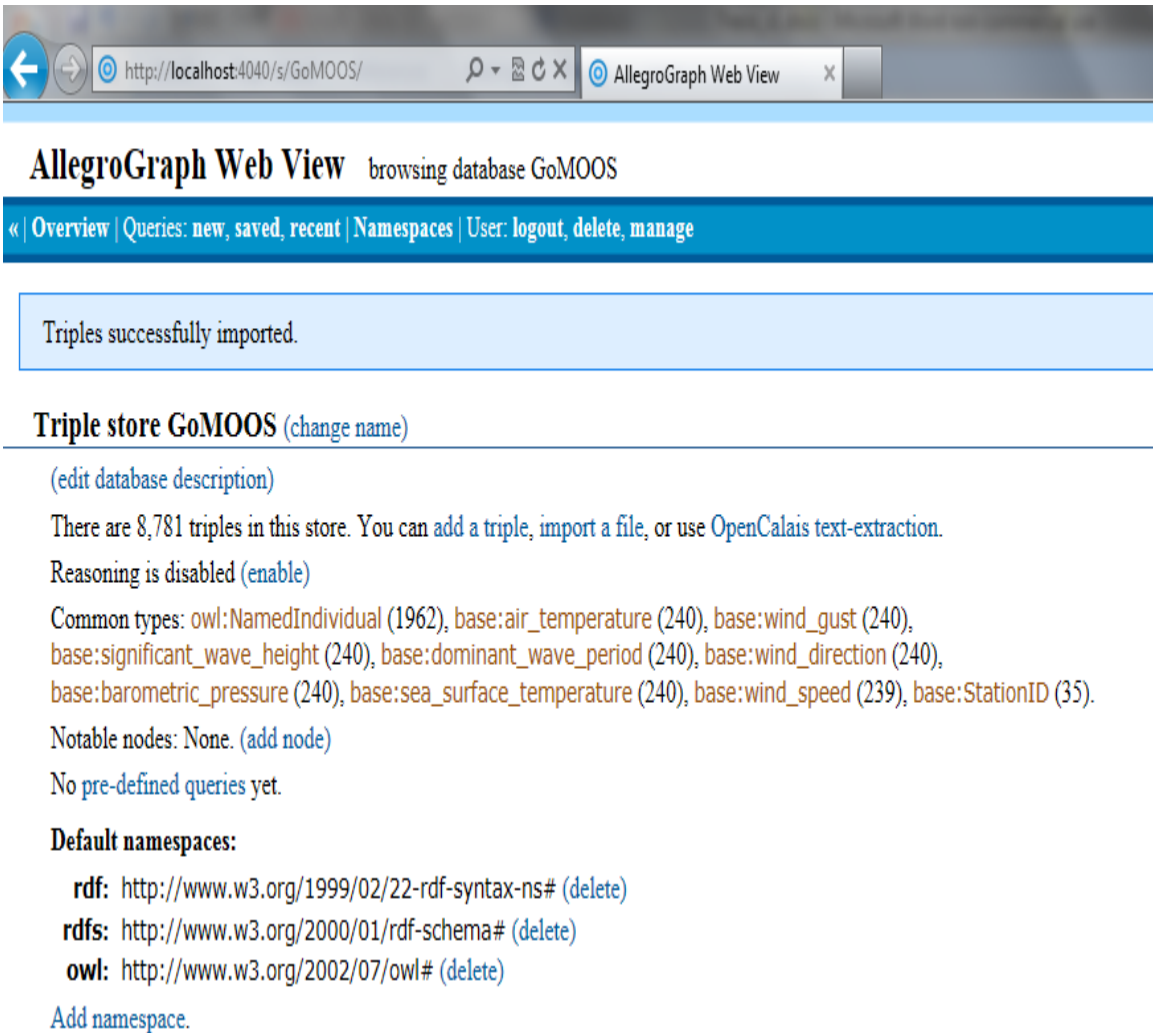


Figure 4.4 GoMOOS triple store with triples imported

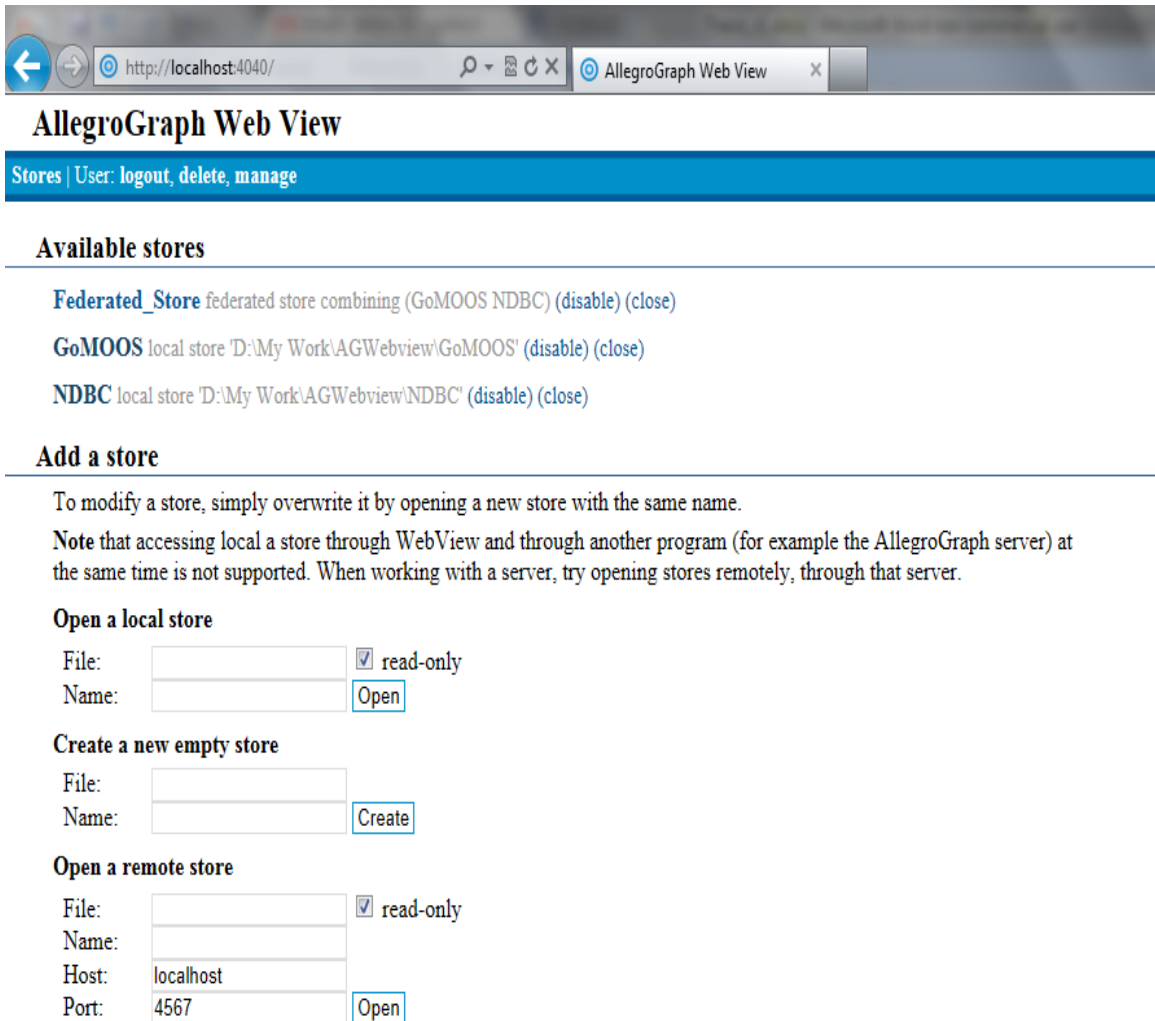


Figure 4.5 Representation of the Federated store formed from NDBC and GoMOOS

#### 4.1 Find out the all the atmospheric properties measured by barometer in NDBC and GoMOOS ?

From the query we can get the results as in NDBC the atmospheric property is 'AtmosphericPressure' and in GoMOOS the atmospheric property is 'barometric\_pressure' shown in Figure 4.6. The result of the query is obtained by querying on the federated database formed by combining both NDBC and GoMOOS. So, we can infer that the same atmospheric property is represented differently in both NDBC and GoMOOS.

The screenshot shows the AllegroGraph Web View interface. The browser address bar displays 'localhost:4040/s/Federated\_Database/#query/3'. The page title is 'AllegroGraph Web View' with a subtitle 'browsing database Federated\_Database'. A navigation bar includes links for 'Overview', 'Queries: new, saved, recent', 'Namespaces', and 'User: logout, delete, manage'. The main section is titled 'Edit query' and contains a SPARQL query:
 

```
show namespaces, add a namespace
SELECT DISTINCT ?NDBC ?GoMOOS
FROM<http://localhost:8080/Buoy/owlFiles/ndbc_new.owl>
FROM<http://localhost:8080/Buoy/owlFiles/gomooosont_new.owl> |
WHERE
{
  ?NDBC base:ismeasuredby base:Barometer .
  ?GoMOOS base:MeasuredBy base:barometer.
}
```

 Below the query are buttons for 'Execute', 'Save as' (with an empty text input), '(optional)', and a 'Shared' checkbox. The 'Result' section shows a table with two columns: '?NDBC' and '?GoMOOS'. The first row of data contains 'base:AtmosphericPressure' under '?NDBC' and 'base:barometric\_pressure' under '?GoMOOS'.

Figure 4.6 Results of SPARQL query to get the atmospheric properties measured by barometer.

Similarly Figure 4.7 gives the information regarding the atmospheric parameters measured by thermistor in both NDBC and GoMOOS. The result show that Dominant Wave Period is measured which is represented as ‘Air Temperature’ in NDBC and air\_temperature in GoMOOS. Thus querying the federated database retrieves useful results.

The screenshot shows the AllegroGraph Web View interface. The browser address bar displays 'localhost:4040/s/Federated\_Database/#query/1'. The page title is 'AllegroGraph Web View browsing database Federated\_Database'. A navigation bar includes links for 'Overview', 'Queries: new, saved, recent', 'Namespaces', and 'User: logout, delete, manage'. The main section is titled 'Edit query' and contains a SPARQL query:

```
show namespaces, add a namespace

SELECT DISTINCT ?GoMOOS ?NDBC
FROM<http://localhost:8080/Buoy/owlFiles/ndbc_new.owl>
FROM<http://localhost:8080/Buoy/owlFiles/gomooosont_new.owl>
WHERE
{
  |?NDBC base:ismeasuredby base:Thermistor.

  ?GoMOOS base:MeasuredBy base:thermistor.
}
```

Below the query, there are buttons for 'Execute' and 'Save', followed by an 'as' field, '(optional)', and a 'Shared' checkbox.

The 'Result' section displays the following table:

<u>?GoMOOS</u>	<u>?NDBC</u>
<u>base:air_temperature</u>	<u>base:AirTemperature</u>

Figure 4.7 Results of SPARQL query to get the atmospheric properties measured by thermistor

Similarly Figure 4.8 gives the information regarding the atmospheric parameters measured by seismometers in both NDBC and GoMOOS. The result show that Dominant Wave Period is measured which is represented as ‘DominantWavePeriod’ in NDBC and dominant\_wave\_period in GoMOOS. Thus querying the federated database retrieves useful results from both datasets.



The screenshot shows the AllegroGraph Web View interface. The browser address bar indicates the URL is localhost:4040/s/Federated\_Database/#query/4. The page title is "AllegroGraph Web View" and the subtitle is "browsing database Federated\_Database". A navigation bar contains links for "Overview", "Queries: new, saved, recent", "Namespaces", and "User: logout, delete, manage".

The main section is titled "Edit query" and contains a link "show namespaces, add a namespace". Below this is a SPARQL query:

```
SELECT DISTINCT ?NDBC ?GoMOOS
FROM<http://localhost:8080/Buoy/owlFiles/ndbc_new.owl>
FROM<http://localhost:8080/Buoy/owlFiles/gomoosont_new.owl>
WHERE
{
  ?NDBC base:ismeasuredby base:Seismometers .

  ?GoMOOS base:MeasuredBy base:seismometers.
}
```

Below the query is an "Execute" button, a "Save" button, an "as" input field, an "(optional)" checkbox, and a "Shared" checkbox.

The "Result" section shows a table with two columns: "?NDBC" and "?GoMOOS". The table contains one row of results:

?NDBC	?GoMOOS
base:DominantWavePeriod	base:dominant_wave_period

Figure 4.8 Results of SPARQL query to get the atmospheric properties measured by seismometer.

The SPARQL query on the federated database to obtain the information regarding the devices present in both NDBC and GoMOOS datasets is represented in Figure 4.9. From the results shown, we can infer that the same devices in GoMOOS and NDBC are represented differently.

localhost:4040/s/Federated\_Store/#query/0

AllegroGraph WebVi... www.franz.com/agr... SPARQL Tutorial | AI... SPARQL Protocol an... SPARQL Demo SPARQL Query Lang...

## AllegroGraph Web View

browsing database Federated\_Store

« | Overview | Queries: new, saved, recent | Namespaces | User: logout, delete, manage

### Edit query

show namespaces, add a namespace

```
SELECT DISTINCT ?gomoos1 ?gomoos2 ?gomoos3 ?ndbc1 ?ndbc2 ?ndbc3
WHERE
{
  base:air_temperature_1 base:IsMeasuredBy ?gomoos1.
  base:barometric_pressure_1 base:IsMeasuredBy ?gomoos2.
  base:wind_gust1 base:IsMeasuredBy ?gomoos3.
  base:AirTemperature_1 base:MeasuredBy ?ndbc1.
  base:AtmosphericPressure_1 base:MeasuredBy ?ndbc2.
  base:WindGust_1 base:MeasuredBy ?ndbc3.
}
```

Execute Save as  (optional)  Shared

### Result

?gomoos1	?gomoos2	?gomoos3	?ndbc1	?ndbc2	?ndbc3
"Thermistor"	"Barometer"	"Anemometer"	"thermistor"	"barometer"	"anemometer"

Figure 4.9 Results of SPARQL query to get the devices used in the coastal buoys

The SPARQL query using CONSTRUCT -to obtain some of the buoy station ids maintained by NDBC and GoMOOS is represented in Figure 4.10. When we query on the federated store we get to know that the representation of the maintained attribute is represented differently. In NDBC, it is denoted as Owned and in GoMOOS it is given by ownedAndMaintainedBy.

AllegroGraph Web View browsing database Federated\_Store

Overview | Queries: new, saved, recent | Namespaces | User: logout, delete, manage | Long parts | Graph names

### Edit query

show namespaces, add a namespace

```

CONSTRUCT {
  base:Buoy_A0101 base:ownedAndMaintainedBy base:GoMOOS.
  base:Buoy_B0101 base:ownedAndMaintainedBy base:GoMOOS.
  base:StationID_44037 base:ownedAndMaintainedBy base:GoMOOS.
  base:StationID_44038 base:ownedAndMaintainedBy base:GoMOOS.
  base:StationID_21413 base:Ownedby base:NDBC.
  base:StationID_32301 base:Ownedby base:NDBC.
  base:StationID_41002 base:Ownedby base:NDBC.
  base:StationID_41010 base:Ownedby base:NDBC.
}WHERE
{
  base:GoMOOS rdf:type owl:NamedIndividual.
  base:NDBC rdf:type owl:NamedIndividual.
}

```

Execute Save as (optional) Shared

### Result

View this result as a graph.

Subject	Predicate	Object
base:StationID_41010	base:Ownedby	base:NDBC
base:StationID_41002	base:Ownedby	base:NDBC
base:StationID_32301	base:Ownedby	base:NDBC
base:StationID_21413	base:Ownedby	base:NDBC
base:StationID_44038	base:ownedAndMaintainedBy	base:GoMOOS
base:StationID_44037	base:ownedAndMaintainedBy	base:GoMOOS
base:Buoy_B0101	base:ownedAndMaintainedBy	base:GoMOOS
base:Buoy_A0101	base:ownedAndMaintainedBy	base:GoMOOS

Figure 4.10 Results of SPARQL query to obtain the station ids from NDBC and GoMOOS

Figure 4.11 provides the information regarding the parameters measured by NDBC and GoMOOS. We can infer that the same air temperature is represented as air\_temperature in GoMOOS and AirTemperature in NDBC.

localhost:4040/s/Federated\_Database/#query/4

**AllegroGraph Web View** browsing database Federated\_Database

« Overview | Queries: new, saved, recent | Namespaces | User: logout, delete, manage Long parts Graph names

**Edit query**

[show namespaces, add a namespace](#)

```

CONSTRUCT
{
  base:air_temperature_1 base:IsMeasuredBy base:GoMOOS.
  base:sea_surface_temperature_1 base:IsMeasuredBy base:GoMOOS.
  base:barometric_pressure_1 base:IsMeasuredBy base:GoMOOS.
  base:wind_direction_1 base:IsMeasuredBy base:GoMOOS.
  base:AirTemperature_1 base:MeasuredBy base:NDBC.
  base:WaterTemperature1 base:MeasuredBy base:NDBC.
  base:AtmosphericPressure1 base:MeasuredBy base:NDBC.
  base:WindDirection1 base:MeasuredBy base:NDBC.
}

```

Execute Save as  (optional)  Shared

**Result**

[View this result as a graph.](#)

Subject	Predicate	Object
base:WindDirection1	base:MeasuredBy	base:NDBC
base:AtmosphericPressure1	base:MeasuredBy	base:NDBC
base:WaterTemperature1	base:MeasuredBy	base:NDBC
base:AirTemperature_1	base:MeasuredBy	base:NDBC
base:wind_direction_1	base:IsMeasuredBy	base:GoMOOS
base:barometric_pressure_1	base:IsMeasuredBy	base:GoMOOS
base:sea_surface_temperature_1	base:IsMeasuredBy	base:GoMOOS
base:air_temperature_1	base:IsMeasuredBy	base:GoMOOS

Figure 4.11 SPARQL query results depicts the parameters measured by NDBC and GoMOOS

CONSTRUCT in SPARQL provides not only to extract data from the triple stores but also helps to create new useful data. CONSTRUCT returns a graph which is a set of triples, as shown in Figure 4.12 and Figure 4.13.

localhost:4040/s/Federated\_Database/#query/1

show namespaces, add a namespace

```
CONSTRUCT
{
  base:Bus_40101 base:ownedAndMaintainedBy base:GMDCS.
  base:Bus_50101 base:ownedAndMaintainedBy base:GMDCS.
  base:StationID_44037 base:ownedAndMaintainedBy base:GMDCS.
  base:StationID_44038 base:ownedAndMaintainedBy base:GMDCS.
  base:StationID_21409 base:OwnedBy base:NRBC.
  base:StationID_32301 base:OwnedBy base:NRBC.
  base:StationID_41002 base:OwnedBy base:NRBC.
  base:StationID_41010 base:OwnedBy base:NRBC.
}
```

Execute Save as (optional) Shared

**Result**

[Back to table view.](#)  
Drag nodes to rearrange them. Showing 8 of 8 triples.  
base:StationID\_44038

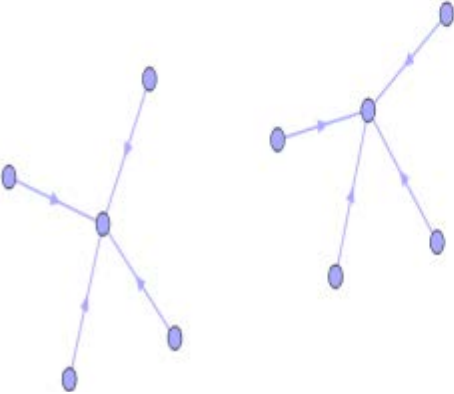


Figure 4.12 Graphical View of the SPARQL CONSTRUCT query

show namespaces, add a namespace

```

CONSTRUCT
{
  base:air_temperature_1 base:IsMeasuredBy base:G0M00S.
  base:sea_surface_temperature_1 base:IsMeasuredBy base:G0M00S.
  base:barometric_pressure_1 base:IsMeasuredBy base:G0M00S.
  base:wind_direction_1 base:IsMeasuredBy base:G0M00S.
  base:air_temperature_1 base:MeasuredBy base:WDC.
  base:WaterTemperature1 base:MeasuredBy base:WDC.
  base:AtmosphericPressure1 base:MeasuredBy base:WDC.
  base:WindDirection1 base:MeasuredBy base:WDC.
}

```

Execute Save as (optional) Shared

---

**Result**

◀ Back to table view.  
 Drag nodes to rearrange them. Showing 8 of 8 triples.  
 base:WindDirection1

Figure 4.13 Graphical View of the SPARQL CONSTRUCT query

The federated store is built and we queried intelligently across the knowledge bases through SPARQL, which allowed expressing queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware.

## CHAPTER V

### CONCLUSION AND FUTURE WORK

#### 5.1 Conclusion

In recent years the innovations in the semantic web made it as an efficient way of representing the data on the World Wide Web. Semantic web enables “intelligent” reasoning capabilities for the web based systems by providing a meaningful structure to the web data. It also enhances the web data for better understanding and responding the user queries. However, with the rapid increase in the amount of information on the semantic web, the integration of heterogeneous web-resources has become a very grave issue. Thus, there is a need for ontology mapping and intelligent querying across different knowledge bases. Ontology mapping is a process in which for each entity in an ontology, a corresponding entity is found in a different but same domain ontology which shares the same meaning [2]. In general, the large number of existing mapping algorithms causes uncertainty due to limited accurate mappings. Even, querying across single datasets will not provide relevant and useful information. Addressing the above constraints, we propose an ontology mapping of two different Resource Description Framework (RDF) datasets and intelligently querying across the federated data-store through SPARQL. In this study, the federated data-store is formed by the amalgamation of National Data Buoy Center (NDBC) and Gulf Of Maine Ocean Observing System (GOMOOS) ontology web Language files (owl files). SPARQL is a RDF query language that can be used for querying the ontology files which are stored in AllegroGraph. The main objective of the

query languages is to make the machine to understand a particular application. We use AGWebview, which is a graphical interface for managing and querying the AllegroGraph triple stores provided by Allegrograph. In this paper the ontologies(GoMOOS and NDBC) are stored as individual triple stores. These individual stores are combined to form a federated store and are queried through SPARQL which is an RDF query language. Thus the heterogeneous data sources are combined and queried intelligently.

## **5.2 Future Work**

Currently the federated database is formed by combining two ontologies. In future the federated store must be formed by more ontologies so that the retrieved results will be more useful. Moreover we can use Gruff, which is a graph based triple store browser for Allegrograph instead of AGWebview, through which we can display visual graphs of subsets of a store's resources and their links.



## REFERENCES

- [1] Guha, R.; Al-Dabass, D.; , "Impact of Web 2.0 and Cloud Computing Platform on Software Engineering," Electronic System Design (ISED), 2010 International Symposium on , vol., no., pp.213-218, 20-22 Dec. 2010 doi: 10.1109/ISED.2010.48
- [2] Hirsch, P.M.; , "Exercise the power of the World Wide Web," Computer Applications in Power, IEEE , vol.8, no.3, pp.25-29, Jul 1995 doi:10.1109/67.392022
- [3] <http://www.w3.org/standards/semanticweb/>
- [4] <http://www.w3.org/standards/semanticweb/data.html>
- [5] Klyne, G., and Carroll, J.J. (eds.) Resource Description Framework (RDF): Concepts and abstract Syntax. W3C Recommendation 10 February 2004. Latest version available at <http://www.w3.org/TR/rdf-concepts/>for rdf.
- [6] Patel-Schneider, P.F., Hayes, P., and Horrocks, I. (eds.). OWL Web Ontology Language Semantic and Abstract Syntax. W3C Recommendation 10 February 2004. Latest version available at <http://www.w3.org/TR/owl-semantic/>for owl.
- [7] Alexander Maedche and Steffen Staab. Ontology Learning for the Semantic Web.
- [8] Biffl, S.; Sunindyo, W.D.; Moser, T.; , "Semantic Integration of Heterogeneous Data Sources for Monitoring Frequent-Release Software Projects," Complex, Intelligent and Software Intensive Systems (CISIS), 2010 International Conference on ,vol.,no.,pp.360-367, 15-18 Feb. 2010 doi:10.1109/CISIS.2010.58
- [9] Dibowski, H.; Kabitzsch, K.; , "Ontology-based Device Descriptions and triple store based device repository for automation devices," Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on , vol., no., pp.1-9, 13-16Sept.2010doi:10.1109/ETFA.2010.5641257

- [10] Dibowski, H.; Kabitzsch, K.; , "Ontology-based Device Descriptions and triple store based device repository for automation devices," *Emerging Technologies and Factory Automation (ETFA)*, 2010 IEEE Conference on , vol., no., pp.1-9, 13-16 Sept. 2010
- [11] <http://www.franz.com/agraph/support/documentation/current/agraph-introduction.html#header3-103>
- [12] Soutou, C.; , "Towards a methodology for developing a federated database system ," *Computing and Information*, 1993. *Proceedings ICCI '93.*, Fifth International Conference on , vol., no., pp.560-564, 27-29 May 1993.
- [13] <http://www.gomoos.org/>.
- [14] <http://www.ndbc.noaa.gov/>
- [15] <http://www.franz.com/agraph/allegrograph/>
- [16] Kamel, M.N.; Kamel, N.N.; , "The federated database management system: an architecture of distributed systems for the 90's," *Distributed Computing Systems, 1990. Proceedings., Second IEEE Workshop on Future Trends of* , vol., no., pp.346-352, 30 Sep-2 Oct 1990 doi: 10.1109/FTDCS.1990.138344
- [17] J. Broekstra, A. Kampan, and F. van Harmelen. Sesame: Ageneric architecture for storing and querying RDF and RDF Schema. In *Int'l Semantic Web Conference '02*, pages 54–68, 2002
- [18] S. Harris and N. Gibbins. 3store: Efficient bulk RDF storage. In *In Proc. of PSSS'03*, pages 1–15, 2003.
- [19] The Jena Semantic Web Toolkit, Web page, Hewlett-Packard Labs, 2003, <http://www.hpl.hp.com/semweb/jena.html>.
- [20] <http://virtuoso.openlinksw.com/>
- [21] Kilian Stoel, Merwyn Taylor, and James Hendler, Efficient management of verylarge ontologies, *Proceedings of American Association for Artificial Intelligencen Conference (AAAI-97)* (1997).
- [22] Raptor RDF Parser Toolkit, <http://www.redland.opensource.ac.uk/raptor/>, 2003
- [23] TAP Project, GetData: Querying the Data Web, <http://tap.stanford.edu/tap/getdata.html>, 2003
- [24] S. Harris and N. Gibbins. 3store: Efficient bulk RDF storage. In *In Proc. of PSSS'03*, pages 1–15, 2003.

- [25] A. Harth, J. Umbrich, A. Hogan, and S. Decker. Yars2: A federated repository for querying graph structured data from the web. In ISWC/ASWC, 2007.
- [26] M.N. Kamel, N.N. Kamel Federated database management system: requirements, issues and solutions *Computer Communications*, 15 (4) (1992), pp. 270–278
- [27] Yan, Y., Wang, C., Zhou, A., Qian, W., Ma, L., and Pan, Y. Efficiently querying RDF data in triple stores. In *Proceedings of WWW*. 2008, 1053-1054.
- [28] Rauber, A.; Witvoet, O.; Aschenbrenner, A.; Bruckner, R.; , "Putting the World Wide Web into a data warehouse: a DWH-based approach to Web analysis," *Database and Expert Systems Applications, 2002. Proceedings. 13th International Workshop on* , vol., no., pp. 822- 826, 2-6 Sept. 2002 doi: 10.1109/DEXA.2002.1045999
- [29] <http://www.w3.org/2005/Incubator/usdl/XGR-usdl-20111027/>
- [30] Maedche, A., & Staab, S. (2001). Ontology Learning for the Semantic Web. *IEEE Intelligent Systems*16(2), 72-79.
- [31] Wang Yong-gui; Jia Zhen; , "Research on semantic Web mining," *Computer Design and Applications (ICCD A), 2010 International Conference on* , vol.1, no., pp.V1-67-V1-70, 25-27 June 2010doi: 10.1109/ICCD A.2010.5541057
- [32] Maedche, A.; Staab, S.; , "Ontology learning for the Semantic Web," *Intelligent Systems, IEEE* , vol.16, no.2, pp. 72- 79, Mar-Apr 2001 doi: 10.1109/5254.920602URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=920602&isnumber=19905>
- [33] <http://www.michelepasin.org/techblog/2011/02/24/survey-of-pythonic-tools-for-rdf-and-linked-data-programming/>
- [34] <http://www.openrdf.org/doc/sesame/users/ch01.html>
- [35] <http://www.franz.com/agraph/support/documentation/current/agwebview.html>
- [36] <http://www.w3.org/RDF/>
- [37] Teswanich, W., Chittayasothorn, S.: A Transformation of RDF Documents and Schemas to Relational Databases. In: IEEE
- [38] Gupta, R.; Malik, S.K.; , "SPARQL Semantics and Execution Analysis in Semantic Web Using Various Tools," *Communication Systems and Network Technologies (CSNT), 2011 International Conference on* , vol., no., pp.278-282, 3-5 June 2011doi: 10.1109/CSNT.2011.67
- [39] <http://tomcat.apache.org>