Master Thesis in Geographical Information Science nr 99

COMPARISON OF GEOSPATIAL SUPPORT IN RDF STORES: EVALUATION FOR ICOS CARBON PORTAL METADATA

Syed Muhammad Amir Raza

2019
Department of
Physical Geography and Ecosystem Science
Centre for Geographical Information Systems
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Syed Muhammad Amir Raza (2019). Comparison of geospatial support in RDF stores: Evaluation for ICOS Carbon Portal metadata.
Master degree thesis, 30/ credits in Master in Geographical Information Science
Department of Physical Geography and Ecosystem Science, Lund University

COMPARISON OF GEOSPATIAL SUPPORT IN RDF STORES: EVALUATION FOR ICOS CARBON PORTAL METADATA

Syed Muhammad Amir Raza

Masters thesis, 30 credits in Geographical Information Sciences

Supervisors

Lars Harrie
Dept of Physical Geography and Ecosystem Science
Faculty of Science, Lund University

Weiming Huang
Dept of Physical Geography and Ecosystem Science
Faculty of Science, Lund University

Abstract

The evolution of World Wide Web (WWW) into semantic web is happening with the aid of standards like Resource Description Framework (RDF), SPARQL and a few others from World Wide Web Consortium (W3C). Over the years, semantic data management technologies have been introduced as software platforms commonly known as RDF stores. Lately these RDF stores have been tested for processing and maintenance of large data sets complying with Linked Data principles. In order to standardize geographic capabilities in these RDF stores, Open Geospatial Consortium (OGC) adopted GeoSPARQL as an extension to SPARQL query. Our study aims to discuss the geospatial capabilities, and the conformance to GeoSPARQL standard, of the five RDF stores: Eclipse RDF4J 2.4.0, Apache Jena 3.9.0, Openlink Virtuoso 7.2.4, Stardog 6.0.1 and GraphDB 8.8.0. Along with the investigation of features, the performance evaluation of these RDF stores has also been conducted by measuring the execution times of a set of GeoSPARQL queries. The evaluation query set consists of non- topological, spatial selection as well as spatial join queries adopted from a spatial benchmark, Geographica.

The geospatial component of Integrated Carbon Observation System (ICOS) Carbon Portal (CP) metadata has been used for performance evaluation in order to establish the suitability of the RDF stores for ICOS-CP requirements. Java Programs have been developed in order to interact with all the RDF stores for upload of data and execution of benchmark queries. Some result set disparities amongst the RDF stores as well as variation in performance metrics on different hardware platforms have also been highlighted in our research.

Keywords: Geography, Geographical Information Systems (GIS), GeoSPARQL, Geospatial query language, RDF stores, Java Programming, RDF4J, Jena, Virtuoso, Stardog, GraphDB

Table of Contents

List of Figures

List of Tables

List of Abbreviations

4-IM	-	Four Intersection Model
API	-	Application Programming Interface
ATC	-	Atmospheric Thematic Centre
BBC	-	British Broadcasting Corporation
CP	-	ICOS-CP
DBMS	-	Database Management System
DE-9IM	-	Dimensionally Extended Nine Intersection Model
ERIC	-	European Research Infrastructure Consortium
ESE	-	Empirical Software Engineering
ETC	-	Ecosystem Thematic Centre
FG	-	Facebook Graph
GIS	-	Geographic Information System
GML	-	Geography Markup Language

HTML	-	Hyper Text Markup Language
HTTP	-	Hyper Text Transfer Protocol
ICOS	-	Integrated Carbon Observation System
ICOS-CP	-	Integrated Carbon Observation System Carbon Portal
ICOS-OTC	-	ICOS Ocean Thematic Center
ICOS-RI	-	ICOS Research Infrastructure
INSPIRE	-	Infrastructure for Spatial Information in Europe
IRI	-	International Resource Identifier
JSON-LD	-	JavaScript Object Notation for Linked Data
JVM	-	Java Virtual Machine
KD-tree	-	K Dimensional Tree
LD	-	Linked Data
LOD	-	Linked Open Data
LUBM	-	Leigh University Benchmark
MBR	-	Minimum Bounding Rectangle
OGC	-	Open Geospatial Consortium
OTC	-	ICOS-OTC
OWL	-	Web Ontology Language
RCC	-	Regional Connection Calculus
RDBMS	-	Relational Database Management System
RDF	-	Resource Description Framework
RDFa	-	Resource Description Framework in Attributes
RDFS	-	Resource Description Framework Schema
RDF/XML	-	Resource Description Framework Extensible Markup Language
RIF	-	Rule Interchange Format
SOCAT	-	Surface Ocean Carbon Dioxide Atlas
SQL	-	Structured Query Language
URI	-	Uniform Resource Identifier
USGS	-	United States Geological Survey
VOS	-	Voluntary Observing Ships
W3C	-	World Wide Web Consortium
WCS	-	Web Coverage Service
WFS	-	Web Feature Service
WGS84	-	World Geodetic System 1984
WKT	-	Well Known Text
WMS	-	Web Map Service
WPS	-	Web Processing Service
WWW	-	World Wide Web
XML	-	Extensible Markup Language

List of Products

AllegroGraph. https://allegrograph.com/
Franz Inc. 2201 Broadway, Suite 715 Oakland, CA 94612, USA. http://franz.com/

GraphDB. http://graphdb.ontotext.com.
Ontotext USA, Inc.One Evertrust Plaza, Suite 1103 Jersey City, NJ 07302, USA. https://www.ontotext.com/

Intel Dual Core/Core i5. Intel Corporation. 2200 Mission College Blvd. Santa Clara, CA 95054-1549 USA. https://www.intel.com/

Jena. https://jena.apache.org/.
The Apache Software Foundation. 401 Edgewater Place, Suite 600 Wakefield, MA 01880 USA. https://www.apache.org/

Oracle Spatial and Graph. Oracle Corporation 500 Parkway, Redwood Shores, CA 94065. USA https://www.oracle.com/

OWLIM. Obsolete and succeeded by Ontotext GraphDB.

Parliament. http://parliament.semwebcentral.org/.
Raytheon BBN Technologies Raytheon Company 870 Winter Street Waltham, MA 02451-1449 USA https://www.raytheon.com/ourcompany/bbn

PostGIS. PostGIS Project Steering Committee . https://postgis.net/

PostgreSQLPostGIS. The PostgreSQL Global Development Group. https://www.postgresql.org/

RDF4J. http://rdf4j.org/.
Eclipse Foundation, Inc. 102 Centrepointe Drive Ottawa, Ontario, Canada, K2G 6B1. https://www.eclipse.org/org/

Sesame. OpenRDF Sesame is obsolete and succeeded by Eclipse RDF4J.

Stardog. Stardog Union 2101 Wilson Boulevard, Suite 800 Arlington, VA 22201. USA.
https://www.stardog.com/

Strabon. Department of Informatics and Telecommunications, National and Kapodistrian University of Athens. Greece. http://www.strabon.di.uoa.gr/

uSeekM. Open Hub uSeekM. Black Duck Inc. 800 District Ave Burlington, MA 01803. USA https://www.openhub.net/p/useekm

Virtuoso. https://virtuoso.openlinksw.com/. OpenLink Software, Inc. 20 Burlington Mall Road, Suite 322, Burlington, MA 01803 U.S.A. https://www.openlinksw.com/

Windows 7. Microsoft Corporation, One Microsoft Way Redmond, WA 98052-6399 USA

1. INTRODUCTION

1.1. Background

The World Wide Web in its basic form is a mesh (web) of inter-connected (hyperlinked) documents which facilitate information sharing to a human audience. The pledge of semantic web over the last two decades has been to transform the web of documents to a web of data (W3C 2001); from a people centric stage to a data centric platform where machines have an equal chance to digest the web contents (Berners-Lee et al. 2001). The concept of Linked Data classifies the interconnectedness of data in the semantic web. In the classical web, the knowledge interpretation from available information sources centered on human beings. The semantic web is a crossover from data/information model to a knowledge model for machines and software modules. Berners-Lee et al. (2001) proposes that semantic web can be realized by incorporation of extensions to the web in the form of standards. HTML is the standard language in document oriented web; RDF is the standard model in the data oriented web. RDF defines a common framework for data interchange and linking on the web in a graph model. Within the RDF framework are serialization standards (data formats i.e. RDF/XML, Turtle, JSON-LD etc.) and basic vocabularies (RDFS and OWL ontologies). The ontologies are represented in RDF model itself and hence the provision of writing new vocabularies is inherent in the RDF framework. SPARQL is the World Wide Web Consortium (W3C) standard query language for the semantic web (W3C 2013b). SPARQL is a tool for RDF data, in nearly the same manner as SQL is for relational data (that is based on the concept of tables, rows and columns).

Geographic data requires additional capabilities for storage and query. Over the years these have been catered by: spatial extensions to RDBMS software, markup extensions for geographic data (GML, WKT etc.) and geo support in the Big Data engines like Geo Spark and Spatial Hadoop (Lenka et al. 2016). The progress in web and mobile GIS over the years has empowered the distribution and visualization of geographic data beyond the mapping and geo-informatics professionals. An average mobile or a computer user is planning his vacations, booking his hotels, organizing air and road travel on a map with live weather and traffic congestion visibility on his device. Geographic data exchange on the web is one key enabler of this upheaval and there are a number of tools and technologies that are responsible for geographic data on the web. The semantic web also needs some spatial extensions for geospatial semantic data in the RDF model for similar requirements.

Battle and Kolas (2012) discuss the efforts undertaken over the years to complement RDF and SPARQL standards with spatial extensions for processing and integration of geospatial linked data. GeoSPARQL has been adapted by the OGC for representation and query of geospatial linked data in RDF model (OGC 2012). GeoSPARQL provides ontology for representation of geographic data as well as topological relationships and SPARQL extension for geospatial aware RDF queries and reasoning.

Many software products capable of storage, query and reasoning on RDF graphs have been offered in the market over the years. Usually referred as RDF stores (also called triple or quad stores), some of these products have incorporated the spatial extensions like GeoSPARQL to handle the geospatial data.

1.2. Problem Statement

Athanasiou et al. (2013) have identified two challenges to the geospatial semantic web as: (i) development of standards and (ii) development of technological artifacts (products) conforming to these standards. A key argument of Athanasiou et al. (2013) is that there is significant development on the first challenge, but the progress on the second challenge (i.e RDF stores with GeoSPARQL conformance and support) is overdue. Certain software products have included geospatial support extensions in their recent releases; however the level of conformance to the standard varies from one product to the other and is not consistent across the market.

Athanasiou et al. (2013) also presents a market research on the conformance of different RDF stores to GeoSPARQL standard and performance measurements evaluated in 2013 highlighting the lack of conformance to any geospatial standard, particularly GeoSPARQL. To our knowledge, since 2013, a thorough study has not been conducted to evaluate the GeoSPARQL support across different RDF stores. As the technology has evolved overtime, a fresh assessment of these objectives is required to establish: the state of the art, the interoperability amongst different RDF platforms, and for the wider benefit of geospatial community at large.

1.3. Aim

The general aim of this research is to study the geospatial capabilities and performance of RDF stores on spatial queries. The suitability of an RDF store for maintenance and distribution of geospatial component of ICOS-CP metadata is also part of the objectives of this study. In particular the thesis aims to study the geospatial capabilities of five RDF stores : Eclipse RDF4J, Apache Jena, Openlink Virtuoso, Stardog, and Ontotext GraphDB. Details of these RDF stores and ICOS-CP are discussed in chapter 4. The specific aim of the study is to answer the following questions:

1. What are the geospatial support features of the RDF stores?
2. What is the level of conformance to the GeoSPARQL standard provided by the RDF stores?
3. What is the performance of the RDF stores for the geospatial SPARQL queries?
4. What are the spatial indexing techniques (if any)?
5. What is the suitability of the RDF store for management of ICOS-CP metadata geospatial component?

1.4. Study Design

This research is an empirical study in software engineering discipline. Empirical Software Engineering is a body of knowledge of applied software engineering research with a strong

empirical component (ESE n.d.). By the start of this century, it was realized that Software Engineering is a big science and empiricism is a necessary ingredient of this science. Empirical studies in software engineering, study the software-related artifacts in order to characterize, understand, evaluate, predict, control, manage and improve them via qualitative and quantitative analysis (Zhang et al. 2018). The most commonly used empirical studies in SE are controlled experiments, case studies and survey (Garcia et al. 2007). Zhang et al. (2018) also found that the interest on empirical methods in software engineering has grown over the years.

This study is undertaken to evaluate the software artifacts and ascertain the geospatial qualitative and quantitative characteristic of the selected RDF stores. The qualitative portion of the research studies the architecture, design, and feature support of the RDF stores from the relevant documentation provided by the vendors of the product. The quantitative perspective of the research is conducted under controlled experiment methods by execution of same set of benchmark queries in similar computing resources on the same geospatial data.

ICOS is a Pan-European research framework for collection of carbon flux and greenhouse gas concentration. The ICOS-CP is the central point for distribution of ICOS research data. The metadata at ICOS-CP is used by the users to explore and search the required datasets for download, and it is maintained as LOD. This study utilizes the geospatial subset of ICOS-CP metadata for evaluation of potential RDF stores with geospatial extension. The ICOS-CP and the datasets used for the study are discussed in detail in chapter 4.

In the qualitative perspective of this study, the product documentation is consulted to assess what all software artifacts and configurations are required to use spatial data in an RDF store. This reveals the methodology of geospatial support as well as identifies the limitations in a particular system and therefore enables the investigation of our first research question. The software documentation is also studied to evaluate the conformance to GeoSPARQL standard specifications, which is required for cross platform compatibility of spatial data. The spatial subset of ICOS-CP metadata is uploaded in the RDF store to test the documented features and this methodology helps us investigate the second research question.

The quantitative research starts with execution of GeoSPARQL compliant benchmark query set on the spatial subset of ICOS-CP metadata. The performance of the query set is evaluated, and analyzed to draw a cross comparison between different RDF stores to study the third research question. The benchmark query set is executed in standard as well as optimized environment to observe the difference of performances in indexed versus un-indexed configurations in each RDF store (if possible), to investigate the fourth research question.

The final research question relates to ICOS-CP considerations, and this is studied with the focus on ICOS-CP geospatial metadata requirements. Results obtained in first four research questions; when analyzed against the ICOS-CP requirements help us establish answer to this last research question.

1.5. Disposition

The report is compiled in seven chapters. A discussion of the technological concepts is conducted in the start of chapter 2 before introducing the details of the GeoSPARQL standard. Chapter 3 deals with an overview of the previous related studies. Chapter 4 comprises the research methodologies and data used for this research. The overview of Geospatial RDF benchmark selected for the study and the benchmark queries established to measure the performance of the software artifacts is also included in chapter 4. In chapter 5, the qualitative and quantities results of the research are presented followed by a discussion on these findings in chapter 6. The conclusion is drawn at the end in chapter 7. For the interested users, the SPARQL queries used for the evaluation of each RDF store and custom code developed to interact from programming environment through the APIs is available at the url https://github.com/Raza-Amir-Syed/TestGeoRDFStores as well as https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8974835&fileOId=8974841.

1.6. Limitations

This study has been conducted with some known limitations listed below:

- The research is undertaken as an off campus study with a single dedicated research student. Therefore the available time, human as well as computing resources were limited. It was not practically possible to test all the available RDF stores and hence a subset was shortlisted in consultation with ICOS-CP team. The available computing resources were limited to a personal computer and hence performance tests on high end machines as well as processing large datasets were also not practical.
- The spatial dataset used for this study is a subset of ICOS-CP metadata. The size of this dataset is modest because data from all the sensors of ICOS is not yet available. However it was preferred to use the actual CP metadata instead of a simulated dataset.
- The Semantic Web has a number of associated tools and technologies; discussion on all of them was not possible. Hence only the toolset considered relevant for the readers of this research are discussed.
- The inference and reasoning framework in semantic web require a detailed discussion of complex technologies. Therefore this topic could not be included in the scope of this study; hence the features of GeoSPARQL dealing with inference and knowledge reasoning through the ontology are also not discussed in detail.

2. TECHNICAL BACKGROUND

2.1. Background of the Semantic Web

In the classical version, the atomic unit of the web is an HTML document. HTML focuses on the presentation of data; hence data would have its meaning only, when it is surrounded by its associated HTML document. The same data, presented in different documents, can have totally different connotations. Consider the word "Java" for example. A web document where programming technologies are listed might use this word in the context of a programming language "Java". Another web document relating to tourist destinations might list this word as the "Java" Island in Indonesia. Unless the context of the document is recognized by rendering the HTML content to a human viewer, this difference in the meaning of Java as a technology term or a geographic entity cannot be appreciated.

As discussed above, rendering of HTML document is targeted on human spectators and data itself does not carry the meaning. Machines and software components cannot appreciate the meaning (semantics) of the data and therefore processing of the web contents by these agents from semantic perspective is not possible. As the machines and software are incapable to make out the meaning of the contents of web documents, the searches over the classical web are mostly related to words or phrases rather than meaningful questions. In other words, the HTML oriented web is unable to answer questions where the software components need to extract information from more than one sources, link them semantically, perform some reasoning and then generate an answer. For example the answer to the question "What was the population of USA when Michael Jordan was born?" is not possible until the whole phrase is found in a web document (Sakr et al. 2018a). Even if it is found, then what happens if the name in the question is changed to Mohammad Ali?

The semantic Web is an extension of the classic web where structure and meaning are provided to the data (Berners-Lee et al. 2001). The atomic unit of this web is a meaningful (semantic) structured data item. In the Semantic Web, the example of "Java" discussed above might have following implications:

- There are two different data items for Java as an island and Java as a programming language. Something like: *places:Java and tech_terms:Java*.
- The semantics are included within the data item itself, hence we can look up the details of *places:Java* and *tech_terms:Java* i.e. they can be de-referenced.
- There is some mechanism that leads the audience (man or machine) to conclude (infer) that the *places:Java* lies in a country *places:Indonesia*.
- The semantic qualifiers are associated with unique locators. For example *"places:"* might stand for *www.places.net.* Similarly the other qualifier "*tech_terms"* is also associated to unique locators and identifiers.

- The term *places:java* itself has a globally unique identity preferably known as International resource Identifier. Similarly *places:Indonesia* and *tech_terms:java* have unique IRIs.
- With the last implication, it can be further implied that: ideally there is only one item on the web when we are referring to anyone of the "*Java"* words discussed above.

Berners-Lee et al. (2001) defines the semantic web as an effort to enable the machines and software agents to: find data, establish relationships amongst data items and process information automatically. The semantic web is not a replacement of the classic web; rather it complements the web. In terms of technology, the semantic web is a set of standards (extensions) to the classic web and it builds upon the existing toolset as shown in the semantic web stack (Figure 2-1). The middle layer (RDF, RDFS, OWL, RIF and SPARQL) are part of the semantic web enabling technology and utilizes the lower layers (XML, URI and Unicode) which are already available from the existing web. The cryptography and trust services depicted in Figure 2-1 are other technologies (not limited to web) for secure and reliable communication between source and destination. The user interface shown in the same figure is the topmost layer to provide convenient access to the users of the semantic web. Some of the web search engines over the years have adapted to the semantic technology. Therefore the search example discussed earlier "What was the population of USA when Michael Jordan was born?" fetches some meaningful results on semantic web search engines. Even if the name is changed to Muhammad Ali, the search engine is able to bring meaningful results. If the same search is performed over a search engine that does not yet utilize the semantic technology, then these queries are still unanswered. A brief introduction of the semantic web technology is discussed in the next few sections.

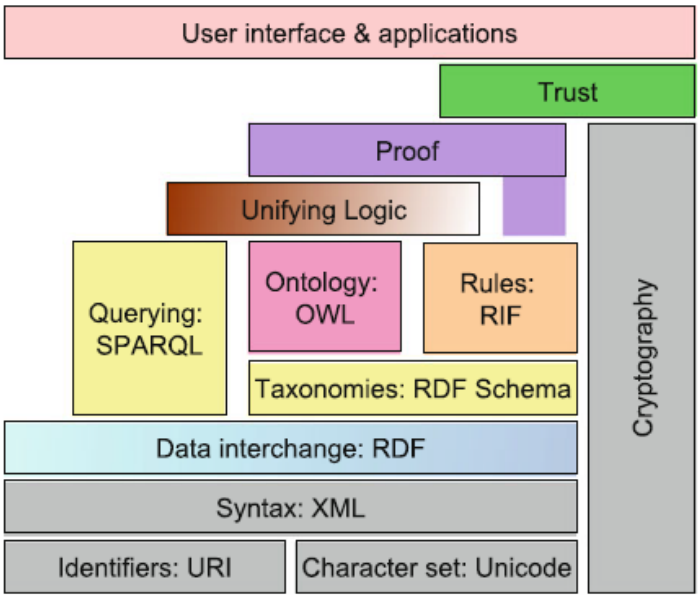


Figure 2-1 Sematic Web Stack (W3C Semantic Web - XML2000, 2008)

2.2. Semantic Web Technology

The semantic web enabling technology is a set of tools (standards, extension and artifacts) which enrich the web with semantic context. The semantic web is based on the logic of a graph. This modeling of web content to a graph is managed by the W3C specifications for RDF. The grammar and basic terminologies of the semantic web is offered by the RDF and RDFS vocabularies while the ontologies driven by Web ontology language manages the knowledge model on the semantic web. SPARQL is the query language for the RDF model. There are some more tools associated with the semantic web and the RDF; however only the basic technologies listed above are considered necessary for our reader and are discussed in the next few sections.

2.2.1. RDF Model

RDF is a framework for representation of information on the web (W3C 2014a). The model adopts that any information can be represented by set of simple sentences; each sentence composed of three words (a subject, a predicate and an object), known as a triple. Referring back to our example from section 2.1, the information can be expressed in RDF, as in listing 2-1.

```
places:Java is Island.
techTerms:Java is progLanguge.
```

Listing 2-1 RDF Representation of 'Java'

These sentences (triples) are RDF statements which are visualized as simple graphs with two nodes (subject and object) connected by an arc (the predicate). The graph visualization for the above two sentence is depicted in Figure 2-2.

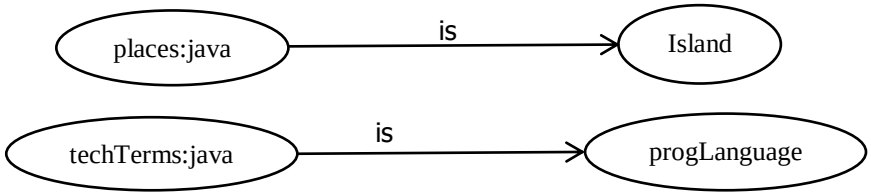


Figure 2-2 RDF triples in a graph model

The subject and object graph nodes in RDF model are resources which can be represented as an IRI or a literal. The object can sometimes be a blank node also. The whole information model is built by adding more triples i.e. more sub graphs to the model as shown in the listing 2-2.

```
places:Java   is   locationType:Island.
places:java   inside   places:Indonesia
places:Indonesia   is   locationType:Country
```

Listing 2-2 Enriched RDF Representation of places:Java

When more and more triples added, the graph grows and the model becomes more informative. With meaningful relationships (predicates), the model can be used to draw more information from the existing information model, by concluding more facts from the existing knowledge graph. It is important to note that RDF provides a conceptual model, but not the syntax (file format) to express such a model/graph. The syntax used above is just an arbitrary format,

deemed easier to read as it appears closer to English language construct. From the semantic web stack (Figure 2-1) it can be deducted that RDF is a layer above XML. There are number of RDF serialization formats: Turtle, N-Triples, RDF/XML, JSON-LD, RDFa and a few others.

2.2.2. RDF Vocabulary

Semantic web contents are required to be structured and organized i.e. the graph nodes need some form of grouping and relationship amongst themselves. The semantic web needs to express meaningful relationships which require meaningful terminology. More importantly the semantics of this terminology should be understood in the same meaning across the user domain. This organization and semantic consistency is expressed by shared vocabularies. The basic vocabulary in this regards is a set of terms known as RDF vocabulary. It is important to note that RDF vocabulary should not be confused with RDF model. The latter is a concept while the former is an actual set of semantic web terminologies. The most important frequently used term, defined in RDF vocabulary is "type" represented as *rdf:type* where the prefix "rdf:" refers to *https://www.w3.org/1999/02/22-rdf-syntax-ns#*. The term *rdf:type* is used as a predicate to associate an instance to its class. In listing 2-3, it is expressed that *places:Indonesia* is an instance of a class *Country*.

```
places:Java          rdf:type      Island.
places:Indonesia     rdf:type      Country.
```

Listing 2-3 Relationship of instances to their classes with RDF Vocabulary

2.2.3. RDF Schema (RDFS) Vocabulary

RDFS is another standard vocabulary that defines terminologies for defining class structure. W3C defines RDFS as a semantic extension of RDF; it provides mechanisms for describing groups of related resources and the relationships between these resources (W3C 2014b). RDFS provides the set of limited but basic classes, properties and utility properties to express the relationship amongst different groups and resources. RDFS supplies the fundamental elements and along with OWL it helps create more complex ontologies and vocabularies which in turn enable the knowledge inference from the semantic web content. Some important constructs of RDFS are: *rdfs:Resource, rdfs:Class, rdfs:subClassOf, rdfs:domain, rdfs:range* and a few more. The prefix "rdfs:" refers to *http://www.w3.org/2000/01/rdf-schema#*. some RDF statements using RDFS constructs are given in listing 2-4.

```
Country   rdfs:subClassOf  PoliticalBounday.
Country   rdfs:domain      places:GeoLocation
```

Listing 2-4 RDFS Constructs for class hierarchy

Another statement that can be added in the listing 2-4 to enrich the RDF model with more information is "*places:java inside places:Indonesia*". However, RDF and RDFS do not provide any construct for "inside". This is because RDF & RDFS are the basic vocabularies for RDF statements and they only define the most general and basic constructs that are needed by all the other data stores or vocabularies built on top of these vocabularies. "inside" is not such a

general construct, however a vocabulary specifically built for geospatial domain might consider "inside" as a general construct.

2.2.4. Web Ontology Language (OWL)

W3C defines OWL as: a semantic web language designed to represent rich and complex knowledge about things, groups of things, and relations between things (W3C 2011). OWL enables the expression of knowledge ontologies which define the basis for reasoning. These ontologies can comprise of domain specific vocabularies and model for specific area to organize and structure the data as well as infer (conclude) new knowledge. An ontology model for the geospatial domain in a vocabulary called "geo" can be created, which defines the constructs needed to represent the geographic properties as given in listing 2-5.

```
places:Java          geo:hasCoordinates   geo:Polygon(X)
places:Indonesia     geo:hasCoordinates   geo:Polygon(Y)
places:Java          geo:inside           places:Indonesia
```

Listing 2-5 RDF Representation with a custom vocabulary

The reasoning capability can lead to inferences from other facts of knowledge already added to the system. Therefore if a new location is added in the above RDF model and it is specified that the new location lies inside *places:Java*, then the reasoning capability enables the system to infer that the new added location also lies inside *places:Indonesia*. This can be achieved by adding a logic rule to the ontology that states if there is a statement *"places:y geo:inside places:x"* and the model also has a statement *"places:z geo:inside places:y"* then it can be inferred that "*places:z geo:inside places:x"*. Therefore this third statement is automatically added to the model by the inference engine.

Reasoning can be based on simple rules with a rule engine or it can be based on ontology (classification based). There could be forward chaining or backward chaining reasoning and there are a number of rule definition languages (Rattanasawad et al. 2018). W3C has recommended Rule Interchange Format specification as a standard to exchange rules between the different rule systems in particular among the web rule engines (W3C 2013). The reasoning framework in semantic web is a vast and complex topic requiring a detailed discussion which is considered beyond the scope of this study.

2.2.5. SPARQL

SPARQL is the W3C standard Query language for RDF data (W3C 2008). SPARQL is a tool to query an RDF graph by specifying graph patterns, as a triple matching criteria to shortlist the sub-graphs from the data set. SPARQL is the de-facto standard for RDF data in the same sense that SQL is for the relational data. While SQL returns the result set in tabular (relation) format only, SPARQL can return the result as: a graph itself, in tabular form, or as a true/false value. SPARQL 1.1 also extends the functionality to update existing data in the graph. A typical SPARQL query consists of the following clauses:

- A *SELECT* or *CONSTRUCT* or *DESCRIB* or *ASK* clause expresses the notion of how the result is to be returned from the query. A *SELECT* returns a tabular result set, a CONSTRUCT or *DESCRIBE* returns a graph while an *ASK* returns a true/false value.
- An optional *FROM* clause specifies the named graph which is to be queried from.
- A *WHERE* clause contains the graph pattern to filter the sub graphs to be included in the result. Variables, expressions and algebra operators are included here. A variable in the query is identified by a prefix "?" or "$".
- At the start of the query, there can be some PREFIX clauses for aliases to some namespaces representing different vocabularies.

A sample query to retrieve all locations in the country Indonesia could be written as something given in listing 2-6. Note that the absence of FROM clause implies that the data is to be fetched from the default graph in the dataset.

```
PREFIX places: <http://places.com/names#>
PREFIX geo: <http://www.gis.geo/ont/geo#>

SELECT ?vPlaces
WHERE {
? vPlaces  geo:inside  places:Indonesia
}
```

Listing 2-6 Sample SPARQL query for places inside Indonesia

The query in listing 2-6 can successfully return those places inside Indonesia, which have been explicitly specified as inside Indonesia with a predicate "inside". However if it is intended to find places that are inside Indonesia from spatial algebra perspective, and not necessarily specified by an RDF statement, then it is required that the framework offers geospatial support. In geospatial enabled search and query frameworks, a set of topological relationship extension functions are made available. For example with such a support, the above query can be re-written (transformed) to find the actual places spatially inside a polygon as listing 2-7.

```
PREFIX places: <http://places.com/names#>
PREFIX geo: <http://www.gis.geo/ont/geo#>

SELECT ?vPlaces
WHERE {
places:Indonesia   geo:hasCoordinates   ?polygonIndonesia.
?vPlaces            geo:hasCoordinates  ?polygonPlaces.
FILTER(geo:isWithin(?polygonPlaces,?polygonIndonesia)).
}
```

Listing 2-7 Revised SPARQL query for places inside Indonesia

2.3. Linked Data

Berners-Lee (2006) highlighted that the success of the semantic web not only depends on the tools and technologies, but the technology must be supplemented with the availability of interlinked data on the web. Therefore the standards devised for the semantic web is just one

piece of the puzzle, while the other lies in the availability of data sets which are semantically interoperable. Linked Data allows meaningful links to be created between pieces of data on the web while promoting a decentralized structure. HTML hyperlinks can link documents; however these links do not express any meaning to the underlying hyperlink. LD focuses on the semantic linking of data on the web. It aims to transform the web from linking documents, into a universal space where pieces of data from different domains are semantically linked and integrated to create a global web of data (Heath and Bizer 2011). Bizer et al. (2009) highlighted four principles of LD which were outlined by (Berners-Lee 2006) as follows:

- Use of Uniform Resource Identifier (URI) as name for things.
- Use of HTTP URIs so that the names can be looked up.
- A look up on the URI should provide meaningful information using the RDF and SPARQL standards.
- Use HTTP URIs for names of all other things so that it can be looked up and interlinked.

2.3.1. Linked Data Sets and Repositories

In order to have interlinked datasets, the concept of Linked Open Data has been introduced. In simple terms, LOD is the linked data that is open for use from licensing perspective. Although it may never be possible to have all linked data as open or all open data as linked, the LOD movement has certainly received attention. The linked data web has grown rapidly in last few years and it was estimated that by 2014 the number of interlinked RDF datasets crossed the 10,000 figure with an estimated RDF statements numbering up to 150 billion (Sakr et al. 2018b).

DBpeida (DBpedia – Wikki n.d.) is an example of LD data set; a large-scale multi-language knowledge base extracted from Wikipedia which represents information in RDF model (Lehmann et al. 2015). There are estimated 3 billion triples in DBPeida and the dataset describes around 4.58 million entities with 50 million links to other RDF datasets (DBpedia – About n.d.). Amongst the other industries, television and broadcasting industry has also embraced the LD concept. BBC is amongst the largest broadcasting corporations in the world. BBC Programmes was launched in 2007 to provide machine readable feeds (RDF/XML, JSON-LD & XML) for every program that BBC broadcasts. BBC has developed its own ontologies (BBC – Ontologies n.d.) to organize and structure its broadcast concepts used in the BBC stores.

GeoNames (GeoNames – Database n.d.) is a spatial LOD dataset containing 25 million geographic names; about 11 million unique features categorized into different classes and subclasses like location names, postal codes addresses etc. GeoNames maintains its ontology (GeoNames – Ontology n.d.) to structure and organize the semantic data. Linked data has also empowered the search engines as well as the social networking industry. Google Knowledge Graph is another example of linked data set that started assisting the search engine since 2012. It is estimated that by 2016 the graph held over 70 billion facts (Enterprise Scale Knowledge Graph-ISWC 2018). It is now possible to ask Google some meaningful questions as discussed in section 2.1.

Another familiar example is Facebook Graph and the supporting API, which have been assisting the development and social networking community since 2013. Facebook encourages developers of the social applications to use the API framework for generating RDF triples in order to capture important user actions. The data from FG is vital for targeted campaigns, including advertisement or even political opinion making (Fruchter et al. 2018).

Linked Open Data Cloud (LOD Cloud n.d.) is a project that maintains the diagram of datasets in the cloud of linked open data (Figure 2-3).

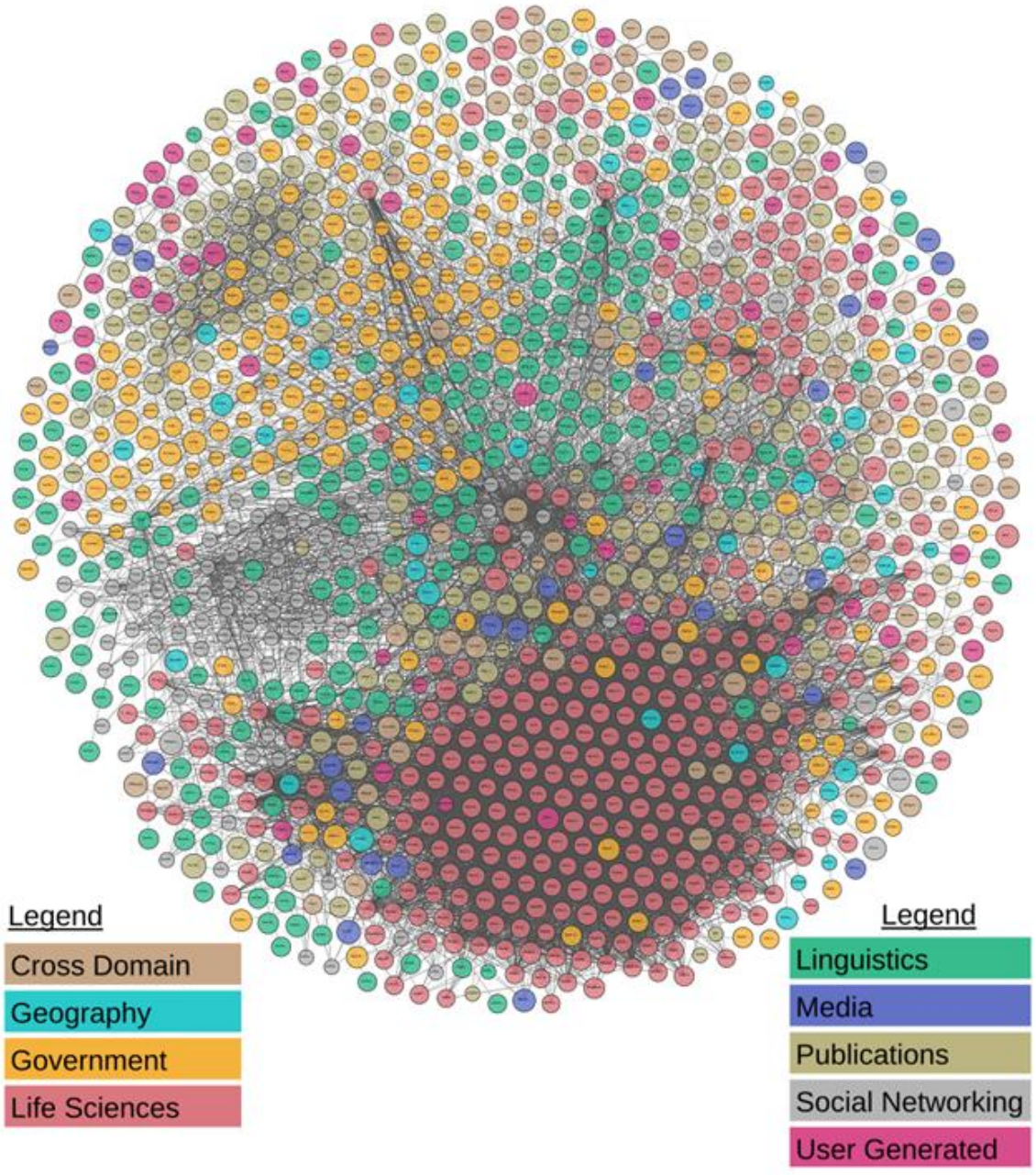


Figure 2-3 LOD Cloud diagram lod-clod.net

As of June 2018, the cloud contains more than 1200 datasets that have been published in linked data format. The LOD cloud diagram depicts the scale, size and heterogeneity of the data. Each circle in the diagram shows a linked dataset where the color of the circle depicts the domain of the data as shown in the diagram legend. The lines between datasets reflect the RDF links within individual datasets i.e. where one datasets refers to another through the IRIs. The whole diagram appears as a cloud of interlinked circles and each dataset conforms to linked open data principles. The project requires that a dataset depicted in the cloud as a circle has at least one thousand RDF statements, has RDF links to at least 50 other datasets within the diagram, and is accessible via a SPARQL endpoint, or by RDF crawling, or through an RDF dump.

As the size of the LD datasets has been constantly growing, the capacity of RDF stores to handle large amount of data has been put to test over the recent years.Leigh University Benchmark is a method for benchmarking and evaluation of semantic web datasets (Guo et al. 2005). A result set known as LUBM 4400K is known for upload, inference and query of 1.08 trillion triples about universities and their departments on Oracle Spatial and Graph platform using LUBM in 2014 (W3C 2018). The first report of a trillion RDF statements upload was made by Franz (Franz Inc) in 2011 on an AllegroGraph platform. Multibilion RDF statement uploads have been reported in the state of the art RDF stores like Stardog, Openlink Virtuoso and others (Boncz et al. 2014).

2.4. Basic Geospatial Concepts

Geographic Information Systems deal with storage, analysis and presentation of geographic information (Nalepa and Furmanska 2009). In order to achieve this, GIS needs to address several specific problems, including: efficient and optimal storage, optimized analysis as well as effective visualization. With the advent of the Web GIS, the map applications like Google Maps have gained a wide acceptance and popularity and a new generation of clearinghouse networks have been developed. A spatial data clearinghouse network is a distributed network that links geospatial data producers, managers, and users electronically (Mansourian et al. 2010). Spatial aware software agents on a smartphone have taken the GIS usage into daily life. Some important tools, technologies and associated terminologies in this context are briefly described in the next sections.

2.4.1. Spatial Representation Standards

Standards are an efficient way to address the issues of interoperability across all domains. Likewise the geospatial domain has also resorted to standards for data interchange and exchange. With the progress of web GIS technology over the years, simple and efficient geographic representation has received special attention due to their less overhead on the communication channels. The OGC Abstract Specifications, models the world in terms of Features (OGC 2003). A feature represents the abstract model of a real world phenomenon and it can represent a physical entity. Features can have spatial as well as non-spatial attributes. The features having spatial attributes are associated to a geometry object which in turn represents a real world object

along with its spatial specifications. Point, line, polygon and other constructs represent different types of geometry.

A textual representation of geometric objects and spatial reference system is provided by the OGC standard Well Known Text. WKT can represent the geometry objects: *Geometry, Point, MutiPoint, LineString, MultiLineString, Polygon, Multipolygon, Triangle, CircularString, Curve, MultiCurve, CompoundCurve, CurvePolygon, Surface, MultiSurface, PolyhedralSurface, TIN (Triangulated irregular network) and GeometryCollection*. WKT is a widely used format and it can represent coordinates in 2D, 3D and 4D space. A few simple examples of WKT representation of geometries in 2D are given in listing 2-8.

```
POINT(5  7)
LINESTRING(5  7,10  12,  11 15)
POLYGON((3  10,4  40, 2  40, 3  10))
```

Listing 2-8 Sample WKT representation of geometries

Geography Markup Language is an XML grammar and OGC standard for representation and exchange of geographic information including the spatial and non-spatial properties. GML is based on OGC Abstract Specifications; hence it models the world in terms of features and geometries. The geometry primitives that make up the GML geometry model (Zhang et al. 2015) are given in Figure 2-4.

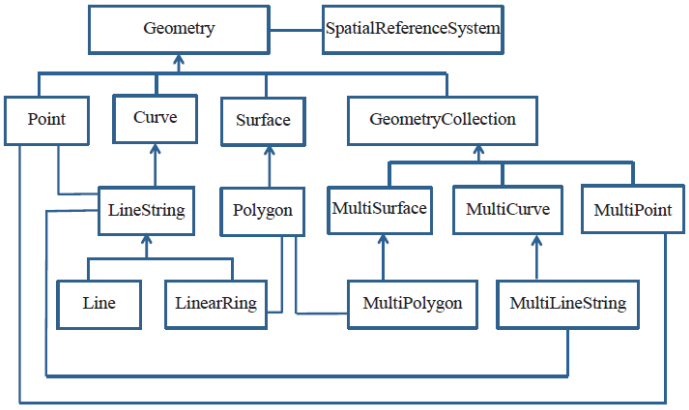


Figure 2-4 GML 3.2 Geometry Primitives

2.4.2. Spatial and Topological Relationships

Spatial entities are related to each other in some form of relationship within the reference space. An island is inside a country, a road crosses an urban area, and a highway intersects another highway. These real-world relations can be modeled as relationships between geometry objects. How far a school lies from a particular road is a spatial relationship but not a topological one. A school lies inside a specific urban unit is a spatial relationships as well as topological relationship. Topological relationships are a subset of spatial relationship with the characteristic that the relationship holds if the size or shape of the geometry changes. Spatial relationships have received distinctive attention in GIS, with special focus on topological relationships.

There are a few widely used topological relationship models. The 4-IM (Egenhofer et al. 1993) and DE-9IM are based on point-set topology while the Regional Connection Calculus based RCC-5 and RCC-8 are models of another category (Baode. and Dong-Qi 2016). Both 4-IM and RCC-8 can represent 8 types of topological relations and they can be translated to each other. The mathematics behind these models is beyond the scope of this thesis.

The OGC simple feature access common architecture builds on the DE-9IM and offers the relationships: *Equals, Disjoint, Intersects, Touches, Crosses, Within, Contains, Overlaps and Relate*. The associated functions/predicates for these relationships take two geometry objects as input parameters and return a boolean value to indicate if the relationship between the geometries holds or not. Relate takes the third argument as the pattern to test and returns a boolean value.

In addition to the above described relationships, some other spatial non-topological utility functions are included in the state of the art GIS software like: *Buffer, ConvexHull, Boundary, Envelope, Intersection, Union, Difference and Symmetric difference*. These functions accept geometry objects as input and return a geometry object after performing the desired operation. Another utility function *Distance* could take two geometries as input and return the distance between them as a double value. In some implementations, the *Distance* function could take a third parameter which is the unit of measurement for the returned value (i.e. meter, kilometer, mile etc.). The *Distance* function can be used to check if an object lies within certain vicinity (within buffer) of another geometric object, or nearby another object.

2.4.3. Geographic Web Services in Connection to Linked Data

OGC has also recommended specifications for interoperability on the web, like Web Map Service, Web Feature Service, Web Coverage Service, Web Processing Service and few more. WFS is an implementation specification (OGC 2005), which allows a client to retrieve, query, and manipulate feature-level geospatial data encoded in GML from multiple sources (Zhang at al. 2015). WMS (OGC 2006) is capable of creating and displaying maps in standard image format that come simultaneously from multiple heterogeneous sources (Zhang et al. 2015).

These standards have been utilized for syntactic interoperability of different geographic datasets and have been vital in the advancement of web GIS services. These technologies enable the interoperability of heterogeneous geographic datasets as well as heterogeneous frameworks. Although, these standards are not directly used for semantic interoperability of geospatial data; there has been some progress in utilization of these services along with the semantic web stack technology. Tschirner et al. (2011) introduced a SPARQL web service with a WFS services as the back end data source, in place of a RDF store. The SPARQL queries received at SPARQL web service are converted to WFS queries for processing. The results received from WFS are transformed back to RDF using a mapping between GML and OWL ontology. Jones et al. (2014) used a reverse implementation by providing WFS to access the geographic linked dataset (RDF store). The queries received by the WFS are translated to SPARQL queries which are processed by an RDF engine. The results generated from this engine are transformed back to WFS XML documents which are returned by the WFS to the caller client.

Earlier implementation of linked data services on existing heterogeneous databases needed a transformation and replication of the complete dataset to the RDF graph model. The data needed to be constantly replicated as the new updates were committed to the source database. By using WFS as the source of linked data services it is not needed to transform the whole data upfront. Hietanen et al. (2016) demonstrated this with a prototype framework that is capable to transform GML to RDF on the fly and offer linked data services on top of a WFS source. Only the object/group URIs are required to be generated upfront, while the remaining data is acquired with query and results transformation at run time. The client interacts with a linked data service while the transformed query is redirected to WFS and updated results are seamlessly fetched from the WFS resource. These results are then transformed back to desired RDF serialization format to be presented to the client.

2.4.4. Spatial Indexing

Unless the complete data in a database is to be processed, most other operations require locating and fetching an entity or a set of entities and then process them. Indexing is a frequently used technique to optimize search and lookup of the specific entities in software systems. The objective of indexing is to reduce the time needed to find and fetch a particular entity. The essence of an index is generally the sort order. Spatial indexing is tricky as there is no sort order in a 2-D space (Samet 2015). For example, a sorting order established on distances computed from point x to all other points, can be void if the reference point changes from x to y. Therefore systems in spatial domain require more sophisticated techniques for search and query optimization. It is obvious that the search and query performance gets a boost from indexing, but this also leverages considerable overhead cost for other operations like data load, update and delete as indices also need to be updated with these operations.

Spatial trees, is a methodology to partition the data or space by structuring the data or space in a sorted hierarchical tree structure (Samet 2015). This enables to discard a specific group of partitions (sub-tree) if it does not match the criteria at search/query time. An R-tree splits the spatial data (point or rectangular representation of geometry) in N rectangular boxes with equal points (Simon 2018). Then each rectangle is recursively further split into more rectangles until there are only N points left in the final boxes. The rectangles at different levels are the tree nodes and this is the most common type of sorted spatial data structure. The main concept behind R-tree is the MBR. KD-tree is similar to R-tree, used for point data; however instead of generating 'N' boxes, data is sorted into two rectangles around the median point which generates disjoint decomposition of space. R+ tree is a compromise between R-tree and a KD-tree. Quad trees are also used for 2D space where each node has exactly four children. While trees are a data driven structure, a space driven spatial indexing can also be conducted in Grid-based spatial index.

2.5. Geospatial Semantic Web and Metadata

Data interoperability amongst different tools and technologies has been an issue since the advent of the web. The schematic and syntactic interoperability is managed by different web standards like markup languages such as GML or XML and mapping service standards like WMS, WCS

etc. (Zhang et al. 2015). The semantic interoperability cannot be managed with these standards because the semantic data is generally lying in the meta tags which do not provide a well-organized structure. The interlinking amongst different geospatial datasets can be well managed if the metadata can be handled better at the semantic level. RDF is the cornerstone technology of semantic web that enables us to represent information about resources. A resource can be a document, an entity or more widely, it can be anything that is identifiable on the web. Hence RDF represents metadata about the web resources. The semantic web can therefore harness the knowledge linking, inference and reasoning capabilities in the geospatial metadata domain, if suitable geospatial extensions are attached to the semantic web toolset. The geospatial datasets are primarily used for these functions:

- Storage of geographic data in a consolidated and consistent manner. The data might include topology information and constraints.
- Search optimization in order to find relevant and required data efficiently. The search might be based on spatial ranges as well as other complex geometric calculation to find the appropriate data from the repository.
- Download the data for desired analysis.

In order to annotate underlying data for the purpose of search and analysis, metadata is needed and it plays important role in building practical applications of GIS (Nalepa and Furmanska 2009). Semantic web technologies provide methods to create and represent structured data that can server as metadata and answer the semantic interchange and exchange queries. The linked data enables the unique and linked references to the desired entities; improving upon the consistency of the datasets. The semantic characterization of metadata through RDF and linked data principles are increasingly adopted in recent years and studies have shown that the the linked data approach for geospatial is also on the rise in recent years.

Goodwin et al. (2009) produced an RDF dataset for administrative geography of Great Britain from the data of National Mapping Agency (Ordnance Survey). A custom ontology was devised to represent regions at different levels and their spatial topological association with each other. Another effort to create spatial linked data sets is LinkedGeoData, an RDF representation of OpenStreetMap data (Stadler et al. 2012). It represents all kinds of spatial features, such as roads or boundaries and is interlinked with DBpedia and GeoNames. Brink et al. (2014) produced transformations of structured spatial data from GML to RDF statements. In addition to geometry coding, the study also presented a transformation of information model from underlying UML to web ontology.

Gore (1999) envisaged the Digital Earth as a multi-resolution, three-dimensional representation of the planet. To address the mechanism of semantic integration of geospatial information from heterogeneous sources for Digital Earth, Vilches-Blázquez et al. (2014) presented a technique to publish linked data for Spanish Nation data-sets related to INSPIRE themes. These datasets were drawn from four diverse domains i.e. administrative units, Hydrology, statistical units and

meteorology; the study also recommended to use geospatial extension for RDF data i.e. GeoSPARQL. Cheatham et al. (2018) created a unified knowledge graph "GeoLink" to seamlessly query and reason over the metadata of prominent geoscience repositories of USA, using the linked data principles. Huang et al. (2018) presented a technique based on relative positioning using linked data to resolve a spatial visualization problem of unsynchronized geometries between thematic and base map objects in the map mashups.

2.6. The GeoSPARQL standard

GeoSPARQL defines a spatial extension to SPARQL query language for geographic information (OGC 2012). There were some other earlier initiatives to incorporate geographic extensions to SPARQL but those were limited to a particular project or an organization. OGC has adopted GeoSPARQL in order to standardize the use of geographic data in the realm of the semantic web and linked data. GeoSPARQL provides the following features:

- a core component (RDFS/OWL vocabulary)
- a set of SPARQL extension functions for spatial computations
- a set of query rewrite rules

2.6.1. Core Component

The RDFS/OWL vocabulary is used for representation of spatial data in a consistent simple feature model. Core RDFS/OWL classes and RDF properties for representation and assertion models are defined here. GeoSPARQL defines a limited vocabulary and expects more domain specific vocabularies to be built upon this base. Both RDFS and OWL have been used in order to enable the systems reasoning capability to benefit from the GeoSPARQL. The basic classes are ogc:*SpatialObject* and ogc:*Feature* which may have a geometry. The geometry component is represented by the class *geo*:*Geometry* which represents the spatial properties of any feature (Figure 2-5). A feature can have several geometry objects, each associated with the property *geo:hasGeometry*. The *geo:hasDefaultGeomtry* is used to link the feature to the default geometry amongst the number of different associated geometries.

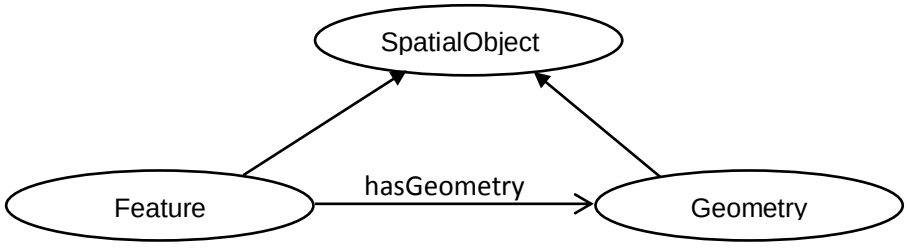


Figure 2-5 GeoSPARQL Fundamental class structure

GeoSPARQL supports two literal formats (serialization) for the spatial representation of geometry: WKT and GML. The geometry object is associated to the literal by the respective predicates, *geo:asWKT* or *geo:asGML*. The *sf:WktLiteral* and *sf:gmlLiteral* are defined in order to represent the data type of the literals. An example of a WKT representation in RDF is given in

listing 2-9. The URI for the coordinate reference system can be included within the literal; if not provided then the default is assumed as WGS84.

```
ex:anXYZSquare  rdf:type   geo:Feature.
ex:anXYZSquare  geo:hasGeometry  ex:geom1 .
ex:geom1  rdf:type  sf:Point.
ex:geom1  geo:asWKT   "POINT(-77.03524 38.889468)"^^geo:wktLiteral.
```

Listing 2-9 Sample WKT representation of geometries

2.6.2. SPARQL Extension Functions

The following spatial methods are included in the GeoSPARQL specifications:

- **Non-topological spatial functions:** These functions return a geometry object after performing the relevant spatial operation: *geof:distance, geof:buffer, geof:convexHull, geof:intersection, geof:union, geof:difference, geof:symDifference, geof:envelope, geof:boundary, and geof:getsrid*.
- **Geometry topological relationship functions**: There are four categories of functions and they all return a boolean value as follows:
  - The functions consistent with DE-9IM simple features specifications are: *geof:sfEquals, geof:sfDisjoint,geof:sfIntersects, geof:sfTouches, geof:sfCrosses, geof:sfWithin, geof:sfContains, geof:sfOverlaps*.
  - A function *geof:relate* is a common query function to check a topological relation between two geometries.
  - The functions consistent with RCC-8 specifications are*: geof:rcc8eq, geof:rcc8dc, geof:rcc8ec, geof:rcc8po, geof:rcc8tppi, geof:rcc8tpp, geof:rcc8ntpp, geof:rcc8ntppi.*
  - The functions consistent with Egenhofer model are: *geof:ehEquals, geof:ehDisjoint, geof:ehMeet, geof:ehOverlap, geof:ehCovers, geof:ehCoveredBy, geof:ehInside, geof:ehContains.*

2.6.3. Query Re-write Rules

Query re-writes rules are included in the GeoSPARQL specifications. This facilitates the usage of spatial predicates in the SPARQL query pattern. The spatial relation can be used like a predicate in the query *where* clause, but seamlessly, a spatial function is used when the query executes. For example, one way to check if a geometry (geom1), lies inside another geometry (geom2), is by using the GeoSPARQL extension function in SPARQL query as follows:

```
FILTER(geof:sfWithin(?geom1,?geom2))
```

Listing 2-10 Sample WKT representation of geometries

If the query re-write is configured than the same clause can be written in predicate form as:

```
?geom1  geof:inside  ?geom2
```

Listing 2-11 Sample WKT representation of geometries

This is supported because the query re-write, automatically transforms (re-writes) the query clause of listing 2-11 to that of listing 2-10. However this transformation is internal for the RDF

store and it is seamless to the users. Another feature of the query re-write is that parameters to the spatial and topological functions are not necessarily required to be concrete WKT or GML literals; because these re-rewrite rules enable the parameters to be geometry or feature objects. The re-write enables the query processor to find the underlying geometry literals from the feature or geometry objects and re-write the function accordingly during the transformation stage.

3. PREVIOUS WORK IN EVALUATION OF RDF STORES

As the semantic web starts evolving into mainstream web supplemented by the growth of online linked data repositories, the RDF stores have been scrutinized for their capabilities to manage the growing industry requirements. Liu and Hu (2005) evaluated seven RDF stores from load and query performance perspective with LUBM datasets. Rohloff et al. (2007) evaluated the triple dataset with LUBM datasets in hybrid RDF stores. Bizer and Schultz (2008) investigated RDF Stores with Berlin SPARQL Benchmark against the load and query times. Morsey et al. (2011) adopted a different approach to evaluate four RDF stores using the DBPedia SPARQL Benchmark and generating interesting QpS(Queries per second) and QMpH (Query mixes per hour) metrics. Cheng et al. (2012) have contested to evaluate three RDF stores (Jena, Sesame and RDF-3x) by investigating each of the query parsing, planning and execution phases. However none of these studies have considered the geospatial standards as part of the evaluation criteria.

Battle and Kolas (2012) conducted a research on enabling of geospatial semantic web with Parliament RDF store and GeoSPARQL. The study builds upon the evolution of GeoSPARQL as the OGC standard and highlights how the conformance to GeoSPARQL across the linked data domain could enable a standard geospatial semantic web. The spatial RDMS have performed reasonably well in geospatial calculations and indexing; however the relational model is unable to handle scenarios where inferences, cross entity joins and variable properties are involved. Parliament is an implementation of RDF store based on GeoSPARQL standard. It builds spatial indexes based on R-trees. The goal in spatial indexing is to split the query in multiple parts for query optimization. A model of linked data from GeoNames and USGS in Parliament was tested for GeoSPARQL conformance and implementation by Battle and Kolas (212).

Garbis et al. (2013) present a benchmark, "Geographica" for evaluation of geospatial RDF stores with two spatial extensions: GeoSPARQL and stSPARQL. Geographica utilizes both synthetically generated as well as real world workloads. The Mirco benchmark within Geographica aims the evaluation of spatial functions like spatial selection, joins, topological relationships and aggregate functions with 29 geospatial SPARQL queries. In the Macro benchmark, Geographica aims at testing 11 application scenarios like reverse geocoding, map search and browsing etc. Evaluation on synthetic data as well as real world data is conducted on RDF stores: Strabon, uSeekM and Parliament in that study.

The most thorough evaluation of RDF stores with reference to GeoSPARQL conformance has been conducted in a GeoKnow project (Athanasiou et al. 2013). RDF stores: Virtuoso, Parliament, OWLIM, uSeekM and Strabon were evaluated along with spatial DBMS: Oracle Spatial and PostgreSQL/PostGIS. The study also included AllegroGraph 4.10, however due to cumbersome conversion of geometric data to custom format required by the platform; it was excluded in the quantitative evaluation of the study.

Athanasiou et al. (2013) evaluation found that Virtuoso 7.0 only supported two dimensional point data from the *pos:* namespace prefix with properties *pos:lat* and *pos:long*. There was no

support for other geometries like line string or polygons and the support for geometry literals conforming WKT or GML was also not available. The topological relationships available in Virtuoso 7.0 were extended through three functions from the *bif:* namespace prefix with relations *bif:ST_intersects, bif:ST_contains* and *bif:ST_within* in addition to some spatial utility/analysis functions. The same study also evaluated OWLIM standards edition which is predecessor of GraphDB. OWLIM support was also restricted to two dimensional point geometries only, from *pos:* namespace prefix. Geospatial support required geospatial indexing and the topological relationships were available from the namespace prefix *omgeo:* as *omgeo:nearby* and omgeo:*within.*

According to Athanasiou et al. (2013), virtuoso 7.0 and OWLIM-SE 5.3 were categorized of as having severely limited geospatial support, while uSeekM 1.2 and Parliament 2.7 were rated as better at spatial support as well as GeoSPARQL compliance. Amongst the RDF stores examined in that study, uSeekM 1.2, Parliament 2.7, Strabon 3.2 and Oracle RDF 11gR2 supported WKT geometry encoding while Virtuso 7.0, OWLIM-SE 5.3 and Allegrograph 4.10 did not support any of WKT or GML standards. A critical observation recorded in that study was that amongst the tested, no two RDF stores had identical geometry representations. Even if some of these supported the WKT, the namespace prefixes were not the same for related vocabularies. Geometry transformation was consistently required during the data upload and query testing. Regarding spatial indexes, R-tree was most popular technique amongst the examined products.

Athanasiou et al. (2013) could not completely evaluate Virtuoso and OWLIM-SE because these RDF stores only supported point geometries. In comparison, the other platforms had to load and index complex geometries; hence a fair assessment was not possible. An overall assessment reveals that in terms of spatial operations, the geospatial DBMS based systems quite easily outperform their RDF competitors. On the other hand, the support for interlinking spatial features is provided by RDF stores which appear totally out of scope for any sort of DBMS based systems. It was concluded that conformance to GeoSPARQL lags consistently and none of the RDF stores offered complete conformance. Amongst the evaluated platforms, Parliament provides comparatively better coverage of the GeoSPARQL standards.

4. MATERIALS AND METHODS

4.1. Integrated Carbon Observation System and the Carbon Portal

The tests in this study are performed on ICOS-CP metadata. ICOS is a research infrastructure established for long term research of greenhouse gasses. The ICOS datasets are freely distributed through the CP and the metadata at the carbon portal plays a vital role in identification of exact datasets required to the users. One of the aims of this study is to identify the suitable RDF stores for efficient management of spatial component of this metadata. Brief introduction to ICOS, the ICOS-CP, ICOS data and metadata is discussed in next few sections.

4.1.1. Integrated Carbon Observation System

ICOS is a Pan-European Research Infrastructure of 12 member countries founded in 2008 (ICOS – About n.d.). ICOS-RI is coordinated and integrated by the ICOS European Research Infrastructure Consortium which was established in 2015. It consists of a network of European observation systems operated at member state level. The ICOS-RI provides high-precision scientific data on carbon cycle and greenhouse gas concentrations. ICOS is coordinating and taking part in several of the European Union's Horizon 2020 research and innovation projects.

ICOS-RI was created to establish a sustained greenhouse gas observation system and enable high quality climate change research and increase usability of the research data. The mission of ICOS-RI is to enable research to: track carbon fluxes in Europe and adjacent regions, to provide long-term observations, and to monitor and assess the effectiveness of reduction in greenhouse gases emission on global atmospheric composition levels (ICOS – Mission n.d.). ICOS has over 130 greenhouse gases measuring stations, three thematic centers (ocean, atmosphere and ecosystem), a head office, a carbon portal facility and a central analytical laboratory.

4.1.2. The ICOS Carbon Portal

ICOS-CP is part of ICOS ERIC and is hosted by the Department of Physical Geography and Ecosystem Science at Lund University, Sweden, with contribution from Wageningen University Netherlands (ICOS-CP – Introduction n.d.). ICOS data is openly available at the carbon portal, a one-stop shop for all ICOS data products. The carbon portal provides free and open access to the high quality ICOS data. It is the gateway to all observational data, derived services and products from ICOS-RI to inform and assist its users. The portal enables access to raw, near real time and final quality-controlled data, supplemented with elaborated (model) data and analyses. It is expected that by the end of 2019 all stations are in full operation and deliver ICOS data through this portal. The services offered at the portal (ICOS-CP – Introduction n.d.) are:

- discovery, preview and download of quality-controlled observational data
- advanced visualizations such as animated flux maps
- popular-scientific products for policy makers, authorities, teachers and students.

The carbon portal is expected to be always changing, in order to constantly develop services and fulfill needs of the users (ICOS-CP – About n.d.). ICOS-CP is involved in several international projects, one of which is ENVRI-FAIR, i.e. Environmental Research Infrastructure and Findable, Accessible, interoperable and Re-usable. FAIR is a set of principles to put specific emphasis on enhancing the ability of machines to automatically find and use the data, in addition to supporting its reuse by individuals (Wilkinson et al. 2016). FAIR is the data approach of the carbon portal for interoperability of data (ICOS-CP – FAIR n.d.). The basic design principle of the portal for metadata is to use linked open data, semantic web ontology, scalable and containerized services, all based on open source software and sharing.

4.1.3 ICOS Data

ICOS-RI builds on three domains: ecosystem, atmosphere and ocean; each domain consisting of its own network of stations i.e. research sites or platforms (ICOS-CP – About n.d.). Each domain has an associated thematic center: Ecosystem Thematic Centre, Atmospheric Thematic Centre and Oceanic Thematic Centre. The raw data from the field stations is uploaded to a central data center for safe custody within 24 hours of data collection. The same data is also sent to relevant thematic center for processing and quality control. The processed data from the thematic centers is delivered to the CP where the data is organized into structured datasets with identifiers for tracking and archiving. These datasets are committed in the central data center and users can access this data from the CP.

The ICOS-RI produces around 25 to 30 TB of sensor data annually, with around 1 GB of processed data products and around 5 to 20 TB elaborated data. Each station in ICOS infrastructure can consist of several sensors. The ocean domain of ICOS consists of a network of marine and coastal stations including Fixed Ocean Stations, Marine Flux Towers and Voluntary Observing Ships. VOS measures the CO_2 on the ocean surface as well as temperature and pressure. The VOSs are usually commercial ships as well as cargo and research vessels operating regularly repeated routes. ICOS marine segment focuses on the North Atlantic and adjacent seas. The linear coverage of ship tracks is integrated with satellite based observations. Interpolation between ship passages as well as extrapolation is used to model the spatial coverage of relevant data objects (OTC – Strategy n.d.).

4.1.4 Other Data at ICOS CP

As the carbon portal started evolving into an efficient data distribution platform, few other related data producers have shown keen interest in delivery of their data through the ICOS-CP. Therefore, in addition to official ICOS data, the ICOS-CP also harvests data from other sources including the Surface Ocean CO_2 Atlas.

SOCAT is a synthesis activity for quality-controlled, surface ocean fCO (fugacity of carbon dioxide) observations by the international marine carbon research community with more than one hundred contributors (SOCAT – Info n.d.). SOCAT data is publicly available, discoverable and citable and the SOCAT community exists since 2007. SOCAT datasets are released

biannually starting from 2011, and an annual public releases is also issued. In version 6 of SOCAT released in June 2018, 23.4 million observations from 1957 to 2017 for the global oceans and coastal seas covering 10 countries has been published. SOCAT welcomes new data submission for inclusion in the next releases. The research team at ICOS-OTC heavily contributes the creation of the SOCAT. ICOS-CP also maintains SOCAT datasets for distribution.

4.1.5 ICOS Metadata

The ICOS data is generated at many different levels and variety of sources including but not limited to: sensors, semi processed data, intermediate data, and structure datasets for end users. The datasets generated from ICOS artifacts are expected to grow manifolds in next few years to come. In order to enable the users to efficiently find the desired data as well as keeping the data download to focused area, the correct identification of the desired data is important. It is required to have the capability to track and archive data artifacts at all levels. This is managed by multi-level metadata generated at different stages of data collection and processing. The ICOS-CP users need access to a certain amount of this metadata for their required search and download. Therefore the metadata requirements of ICOS are devised as follows:

Metadata store at ICOS-CP should be fast and efficient, mostly open to the public. It should be scalable and serviceable 24/7. The metadata services follow FAIR data principle. Any portal should be able to link to ICOS metadata and vice versa i.e. data should be discoverable. This means that there should be an access point for human and machine access to the metadata (SPARQL endpoint). Hence data should be in linked open data form. This requires an RDF database. All data artifacts should have unique IDs preferably IRIs which could be de-referenced with a landing page. Hence metadata should be ontology driven and accessible through http(s), via SPARQL .The ICOS research data and the collection platforms have geographic features and characteristics, hence the metadata services should be spatial aware.

ICOS-CP maintains the ICOS metadata as linked open data and users (human and software agents) can query from this dataset through the SPARQL endpoint at the carbon portal (ICOS-CP – SPARQL End Point n.d.). The ICOS CP ontology can be accessed at (ICOS-CP – Ontology n.d.). The basic name spaces represented by ICOS CP metadata is http://meta.icos-cp.eu/ontologies/cpmeta/ defined as "*cpmeta:*". All stations in the metadata have geographic characteristics as point data, however the data objects and stations relating to VOS platforms have spatial coverage as well, which is represented as GeoJSON string of line and polygon data. This data is used for our study as discussed in the next section.

4.2. Research Data

The data set used for this research was extracted from the subset of ICOS-CP metadata, relating to SOCAT datasets. The data was downloaded as an RDF/XML file from https://meta.icos-cp.eu/resources/socat/ on October 30, 2018. The downloaded metadata covers all the geographic

data available in ICOS-CP metadata on that date. The geospatial data in the downloaded dataset relates to the following facts:

Polygon and Line String Data. The polygon and line string data represents the spatial coverage of the associated data object. The spatial coverage of the data object is the trajectory of the underlying VOS platforms. The simple ship trajectories are coded as line strings while the complex trajectories have been simplified as polygons. For each data object having a spatial coverage, there is an RDF statement that connects it to the spatial coverage object with the predicate *cpmeta:hasSpatialCoverage*. The spatial coverage object is then associated to its geographic literal with the predicate *cpmeta:asGeoJSON*. The geometry in the source data is coded as a GeoJSON string literals. There are 88 polygons and 853 line strings as spatial coverage objects in GeoJSON format in the downloaded data. Listing 4-1 depicts the RDF statements (in Turtle format) for a sample data from the downloaded data set.

```
PREFIX  obj: <http://meta.icos-cp.eu/ objects />
PREFIX  res: http://meta.icos-cp.eu/resources/
PREFIX  cpmeta1: <http://meta.icos-cp.eu/ontologies/cpmeta/>

obj:obj1    rdf:type    cpmeta1:DataObject
obj:obj1    cpmeta1:hasSpatialCoverage  res:spcov_obj1
res:spcov_obj1  rdf:type  cpmeta1:SpatialCoverage
res:spcov_obj1  cpmeta1:asGeoJSON  {"type": "Polygon",    "coordinates":
                    [[[11.195, 54.047522], [14.45336, 55.079674], [18.387972],
                    [11.195, 54.406729], [11.195, 54.047522]]]}
```

Listing 4-1 Sample ICOS CP SOCAT metatdata statements for polygon data

Point Data. The point data in our downloaded metadata are the longitude and latitude of the ICOS Stations. The RDF statements for this data has the Station Id as subject and *cpmeta:hasLatitude* or *cpmeta:hasLongitude* as predicates. The objects in these statements are the double literal values. There are 127 stations with these two predicates in the downloaded dataset. A sample data extracted from the data is given in listing 4-2.

```
PREFIX cpmeta1: <http://meta.icos-cp.eu/ontologies/cpmeta/>

stations:OS_11SS   rdf:type  cpmeta1:OS
stations:OS_11SS   cpmeta1:hasLatitude    51.226774
stations:OS_11SS   cpmeta1:hasLongitude   2.934924
```

Listing 4-2 Sample ICOS-CP metatdata statements for point data

4.2.1. Preparation of Research Data

The total number of spatial objects in the dataset is 1068 (88 polygons + 853 line strings +127 points). As mentioned earlier in section 2.5.1, GeoSPARQL supports two literal formats (serialization) for the spatial representation of geometry: WKT and GML. GeoSPARQL however has no support for GeoJSON literals or the *cpmeta:hasLatitude* and *cpmeta:hasLongitude* predicates. Therefore in order to use the spatial data from the downloaded data set, a

transformation to the GeoSPARQL compliant format is required. This transformation was achieved with SPARQL *Construct* statements as depicted in listings 4-3 to listing 4-5.

```
PREFIX cpmeta1: <http://meta.icos-cp.eu/ontologies/cpmeta/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX sf: <http://www.opengis.net/ont/sf#>

CONSTRUCT {?obj a geo:Feature;
               geo:hasGeometry [
                   a sf:Polygon;
                   geo:asWKT ?wkt                ] }
  WHERE{
 ?obj cpmeta1:asGeoJSON ?geoJSON
 BIND(REPLACE(REPLACE(REPLACE(REPLACE(?geoJSON," \"","")," \t","")," \n","")," ","") AS
?v1).
 BIND(REPLACE(REPLACE(REPLACE(?v1,"type:",""),"coordinates:",""),"],\\["," @") AS ?v2).
 BIND(REPLACE(REPLACE(?v2,","," "),"@",",") AS ?v3).
 BIND(REPLACE(REPLACE(REPLACE(?v3," \\[\\[\\[","((")," \\[\\[","(")," \\[","") AS ?v4).
 BIND(REPLACE(REPLACE(REPLACE(?v4,"]]]","))")),"]]",")"),"]","") AS ?v5).
 BIND(UCASE(REPLACE(REPLACE(?v5," \\{","")," }","")) AS ?v6).
 BIND(CONCAT(?v6,"^^geo:wktLiteral") AS ?wkt).
 FILTER(CONTAINS(UCASE(?wkt),"POLYGON")).
}
```

Listing 4-3 SPARQL Query to construct GeoSPARQL compliant RDF for polygon data

```
PREFIX cpmeta1: <http://meta.icos-cp.eu/ontologies/cpmeta/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX sf: <http://www.opengis.net/ont/sf#>

CONSTRUCT {?obj a geo:Feature;
               geo:hasGeometry [
                   a sf:LineString;
                   geo:asWKT ?wkt
               ] }
  WHERE{
 ?obj cpmeta1:asGeoJSON ?geoJSON
 BIND(REPLACE(REPLACE(REPLACE(REPLACE(?geoJSON,"\"","")," \t","")," \n","")," ","") AS
?v1).
 BIND(REPLACE(REPLACE(REPLACE(?v1,"type:",""),"coordinates:",""),"],\\["," @") AS ?v2).
 BIND(REPLACE(REPLACE(?v2,","," "),"@",",") AS ?v3).
 BIND(REPLACE(REPLACE(REPLACE(?v3," \\[\\[\\[","((")," \\[\\[","(")," \\[","") AS ?v4).
 BIND(REPLACE(REPLACE(REPLACE(?v4,"]]]","))")),"]]",")"),"]","") AS ?v5).
 BIND(UCASE(REPLACE(REPLACE(?v5,"\\{",""),"}","")) AS ?v6).
 BIND(CONCAT(?v6,"^^geo:wktLiteral") AS ?wkt).
 FILTER(CONTAINS(UCASE(?wkt),"LINESTRING")).
}
```

Listing 4-4 SPARQL Query to construct GeoSPARQL compliant RDF for line string data

```
PREFIX cpmeta1: <http://meta.icos-cp.eu/ontologies/cpmeta/>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
```

```
PREFIX sf: <http://www.opengis.net/ont/sf#>

CONSTRUCT {?obj a geo:Feature;
               geo:hasGeometry [
                   a sf:Point;
                   geo:asWKT ?wkt
               ] }
  WHERE{
 ?obj cpmeta1:hasLatitude ?Lat;
        cpmeta1:hasLongitude ?Lon.
  BIND(CONCAT("Point(",STR(?Lon)," ",STR(?Lat),")","^^geo:wktLiteral") as ?WKT).
}
```

Listing 4-5 SPARQL Query to construct GeoSPARQL compliant RDF for point data

These queries construct sub-graphs against each spatial object in the downloaded dataset. The total spatial objects in our data are 1068 and the SPARQL constructs for data transformation generate four RDF statements for each object. Therefore our generated geospatial dataset for the research has 4272 RDF statements. A sample of RDF statements relating to a point, a line string and a polygon are shown in listing 4-6. The geometry object identifiers have been replaced (_:genid-123, _:genid-456 and _:genid-789) in listing 4-6 to avoid complexity, because the actual identifiers are 60 characters long. This spatial dataset of 4272 statements is used in the evaluation of SPARQL queries for all the selected RDF stores as discussed in the next sections.

```
PREFIX station: <http://meta.icos-cp.eu/resources/stations/>
PREFIX sf: <http://www.opengis.net/ont/sf#>
PREFIX geo: <http://www.opengis.net/ont/geosparql#>
PREFIX res: <http://meta.icos-cp.eu/resources/>

station: OS_11BE   rdf:type  geo:Feature
station: OS_11BE   geo:hasGeometry    _:genid-123
_:genid-123    rdf:type  sf:Point
_:genid-123    geo:asWKT  "POINT(3.113968 51.360925)"^^geo:wktLiteral

res: spcov_-8m8Q4cXf0ONCgUZB-7vjLFE   rdf:type    geo:Feature
res: spcov_-8m8Q4cXf0ONCgUZB-7vjLFE   geo:hasGeometry  _:genid-456
_:genid-456  rdf:type   sf:Polygon
_:genid-456  geo:asWKT  "POLYGON ((18.591 54.945998,18.808718 54.280256,21.759
                         59.062,24.959511 59.814679,25.196764 60.108495,21.731027
                         59.344977,17.209978 57.398751,13.004 54.845,13.032
                         54.843,13.032019 54.843287,13.858343 54.79279,18.632254
                         56.861101,18.30532 55.076225,14.6811 54.661957,13.198002
                         54.830021,13.034 54.843,13.022 54.841,12.810975
                         54.861532,11.342376 54.198734,11.932917 54.079201,18.591
                         54.945998))"^^geo:wktLiteral

res: spcov_8ilDZu8ZSuBc4WEJ8n41mLoF   rdf:type    geo:Feature
res: spcov_8ilDZu8ZSuBc4WEJ8n41mLoF   geo:hasGeometry  _:genid-789
_:genid-789  rdf:type   sf:LineString
```

```
_:genid-789  geo:asWKT  "LINESTRING (-0.60175 49.685,-1.9754 50.068,-2.9287 49.978,-
                         5.0234 49.581,-7.8892 48.464,-12.148 46.701,-16.986 44.59,-
                         22.019 42.325,-28.109 39.467,-29.475 38.704,-31.989 37.033,-
                         36.201 34.146,-40.738 30.868,-47.031 26.176,-49.015 24.663,-
                         51.22 20.732,-52.722 18.371,-55.709 16.155,-60.59 14.459,-
                         60.591 14.459)"^^geo:wktLiteral
```

Listing 4-6 RDF Statements for point, line string and polygon geometry from study dataset

4.3. Evaluation Technique

Software engineering has experienced a turn towards empiricism as it shifted from design oriented discipline to an insight-driven and theory-centric discipline over the years (Fernandez and Passoth 2018). This study is planned as an empirical software engineering study to evaluate the software artifacts of selected RDF stores. The endeavor of this study during the cross comparison of the selected five RDF stores is to conduct the evaluation of each store in a controlled experiment method. For this purpose, during the quantitative portion of the study (where the statistics of software performance are gathered), same set of environment (i.e. hardware resources and operating environments) is maintained. Each RDF store is tested on two different machines, both having different set of hardware resources but similar operating environments. Henceforth we define two machines as follows:

- Machine A is a computing machine with Intel Dual core processor (2.3 G Hz) and 2 GB of main memory.
- Machine B is a dedicated virtual machine with an Intel Corei5 (3.4 G Hz) processor with 04 GB memory.

The operating environment on both machines is uniform which includes Microsoft Windows 7 ultimate 64 bit with Eclipse IDE for Java Developers version 4.9.0 for source code development and compilation. Java version 1.8.201 is used for all the tests on these machines. At any time only one of the RDF stores are running on the machine.

4.3.1. Selection of RDF Stores

The management and maintenance of ICOS-CP metadata requires an efficient data store which supports the linked open data and has a suitable spatial component as well. Additionally the ICOS data is growing at a rapid rate and that would implicate a growing size as well. The RDF stores to be evaluated for suitability of this data are required to be capable to handle large datasets and also offer the geospatial support conforming to an established standard (GeoSPARQL).

There are quite many RDF stores that could have been potential candidate for this study. However, the study has limited timelines and other resources, as well as ICOS-CP preferences. Therefore in consultation with the architecture and design team at ICOS-CP, following criterion has been established for shortlisting the RDF stores to be studied:

- The RDF store should be actively supported.
- The RDF store should support W3C standards like SPARQL 1.1, and also semantic reasoning, including ontological and rule-based reasoning.
- The RDF store should have advanced geospatial capacity.
- The RDF store should preferably support GeoSPARQL.
- The RDF store should preferably be open-source.

Based upon the criteria listed above, following RDF stores were selected for this evaluation:

- Eclipse RDF4J
- Apache Jena
- OpenLink Virtuoso
- Ontotext GraphDB
- Stardog

RDF4J (formerly known as Sesame) and Jena are selected because these open source products are commonly used as libraries or underlying framework component by a wide range of other RDF stores. A number of RDF databases and stores support Java APIs conforming to RDF4J and Jena interfaces. Openlink Virtuoso also offers an open source edition and it also extends reasonable geospatial support and indexing. GraphDB offers a free edition and it provides a strong GeoSPARQL support. Stardog also extends considerable geospatial support and it offers a few months trial for testing purposes.

4.3.2. Qualitative Study

The software components and the background building blocks incorporated in each of the RDF stores to manage the geospatial linked data were investigated from the official documentation of each of the RDF stores. The documentation is consulted primarily for the study of following qualitative features:

- software components, architecture, deployment and licensing
- built in applications and interface utilities for data upload, query and management
- utilization of other software components (from open source community)
- available support for access to the RDF store from programming environment (APIs)
- geospatial support and capabilities
- conformance to GeoSPARQL
- extended spatial operations and topological relationships
- geospatial query optimization and spatial indexing techniques.

4.3.3. Preparation for Quantitative Study

After these features are studied, the software is downloaded for installation and testing. All the selected RDF stores provide some sort of command line or web interface which has been used to upload the research data and run a few basic SPARQL queries to validate the software is

configured and executed smoothly. Geospatial support/indexing was also enabled for validating the documented features. The basic execution of software artifacts is tested here to confirm the smooth (error free) operations, but no quantitative measurements were recorded at this stage.

4.3.4. Quantitative Study

The quantitative portion of the study focuses on performance metrics against a set of geospatial queries. The GeoSPARQL compliant query set was derived from the geospatial SPARQL benchmark, *Geographica* (Garbis et al. 2013). The set of queries derived from the micro benchmark of Geographica against real world datasets are considered in this study.

Geographica was developed to evaluate the GeoSPARQL as well as stSPARQL compliance and there are 29 queries in the micro benchmark divided in three categories: non-topological, topological and spatial join queries. Queries Q6, Q28 and Q29 are evaded for not being GeoSPARQL compliant. Q14 required a function call from within another function and this was not supported by most of the under study RDF stores, hence it was also eliminated from the benchmark queries. Geographica uses three different datasets for evaluation, each set having different types of geometries. However in this study, all the three types of geometries (point, lines and polygon) reside in a single dataset; hence the queries are slightly modified to use self joins without violating the original spatial intent of the queries as shown in Table 4-1.

Table 4-1 GeoSPARQL compliant Geographica queries for evaluation of ICOS CP Metadata

S#	Ref	Operation	Geographica Benchmark Query (on 03 spatial datasets)	Query for this study (on ICOS metadata spatial data set)
Non topological functions				
1	Q1	Boundary	Construct Boundary of all polygons of one dataset	Construct Boundary of all polygons in the dataset
2	Q2	Envelope	Construct Envelope of all polygons of one dataset	Construct Envelope of all polygons in the dataset
3	Q3	Convex Hull	Construct Convex Hull of all polygons of one dataset	Construct Convex Hull of all polygons in the dataset
4	Q4	Buffer	Construct Buffer of all lines of one dataset	Construct Buffer of all line strings of in the dataset
5	Q5	Buffer	Construct Buffer of all Polygons of one dataset	Construct Buffer of all polygons in the dataset
	~~*Q6*~~	~~*Area*~~	~~*Construct Area of all Polygons*~~	*Non GeoSPARQL compliant function.*
Spatial Selection				
6	Q7	Equals	Find all lines of one dataset that are spatially equal to a given line.	Find all line strings that are spatially equal to a given line string.
7	Q8	Equals	Find all polygons of one dataset that are spatially equal to a given polygon	Find all polygons that are spatially equal to a given polygon.
8	Q9	Intersect	Find all lines of one dataset that intersect a given polygon	Find all line strings that intersect a given Polygon.
9	Q10	Intersect	Find all polygons of one dataset that intersect a given line	Find all polygons that intersect a given line string.

10	Q11	Overlaps	Find all polygons of one dataset that spatially Overlaps a given polygon	Find all polygons that spatially Overlaps a given polygon
11	Q12	Crosses	Find all lines of one dataset that spatially cross a given line	Find all line strings that spatially cross a given line string
12	Q13	Within Polygon	Find all points of one dataset that are contained in a given polygon	Find all points of ICOS metadata that are spatially within a given polygon
	~~Q14~~	Within Buffer	Find all points of one dataset that are within the buffer of a given point	Find all points that are within the buffer of a given point
13	Q15	Near a Point	Find all points of one dataset that are within fixed distance of a given point	Find all points that are within a fixed distance to a given point.
14	Q16	Disjoint	Find all points of one dataset that are spatially disjoint of a given polygon	Find all points that are disjoint to a given polygon.
15	Q17	Disjoint	Find all lines of one dataset that are spatially disjoint of a given polygon	Find all line strings that are spatially disjoint with a given polygon.
Spatial Joins				
16	Q18	Equals	Find all points of a dataset which are equal to a point in another dataset	Find point to point equality of all ICOS metadata points.
17	Q19	Intersects	Find all points of one dataset that intersect a line of another dataset	Find all points that intersect any line string in the dataset.
18	Q20	Intersects	Find all points of one dataset that intersect a polygon of another dataset	Find all points that intersect any polygon in the dataset.
19	Q21	Intersects	Find all lines of one dataset that intersect a polygon of another dataset	Find all line strings that intersect any polygon in the dataset
20	Q22	Within	Find all points of one dataset that are within a polygon of another dataset	Find all point and polygons where the point lies inside the polygon.
21	Q23	Within	Find all lines of one dataset that are within a polygon of another dataset	Find all line strings that lie within any polygon in the dataset
22	Q24	Within	Find all polygons of one dataset that are within a polygon of another set.	Find all polygons which are completely within any other polygon in the dataset.
23	Q25	Crosses	Find all lines of one dataset that cross a polygon of another dataset	Find all line strings that cross a polygon in the dataset.
24	Q26	Touches	Find all polygons of a dataset that touch other polygons	Find all polygons that touch any other polygon in the dataset
25	Q27	Overlaps	Find all polygons of one dataset that overlap polygons of another dataset	Find all polygons that overlap any other polygon in the dataset.
Aggregate Functions				
	~~*Q28*~~	~~*Extension*~~	~~*Construct the Extension of all polygons of a dataset*~~	*Non GeoSPARQL compliant*
	~~*Q29*~~	~~*Union*~~	~~*Construct Union of All Polygons of a dataset*~~	

Amongst the five selected RDF stores, some have limited or no GeoSPARQL compliance. In such cases, only those queries from Table 4-1 are evaluated which are available in that RDF store. The only leverage allowed in such tests, is to change the query from an object to a box query (if applicable). Hence, if an RDF store does not support a *within* or *intersect* topological relationship between two arbitrary objects, but rather offers the functionality to test the same operations between one object and a rectangle (box), then this change is accommodated for processing the query. These exceptions are highlighted in the result sections.

For this study each RDF store has been tested through the available programming APIs and custom java code has been developed to test each of the stores. The built in applications or command line tools are therefore not used in the quantitative evaluation. When the evaluation code for an individual RDF store is executed on the designated machine, the necessary initializations are conducted before proceeding to execute the benchmark queries. The initialization code creates storage structures on the machine (repositories or datasets or database) along with necessary spatial parameters (if required). The research data is then loaded in the local disk based storage structures and the whole query set is executed in a loop for 100 times. A complete query set (set of benchmark queries) is executed in each iteration, followed by next iteration and so on for one hundred times. This means that the query set executes 100 times in one configuration for each RDF store separately on both machine A and machine B. Therefore 100 different query times are recorded for each query. The first 20 iterations are not used for calculation of performance measurements. The statistics are computed against each query for the average of last 80 iterations (iteration number 21 to 100) in order to balance out any spikes within individual iterations. More detailed discussion on this topic is conducted in chapter 6.

If the RDF store supports geospatial capabilities in both indexed and un-indexed configurations, then separate performance of each query in both modes is recorded and statistics are computed accordingly. For reference and larger benefit of the geospatial community, the developed source code as well as the compiled program have been uploaded to an online source code repository at url https://github.com/Raza-Amir-Syed/TestGeoRDFStores as well as as well as https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8974835&fileOId=8974841.

4.3.5. ICOS-CP Considerations

In consultation with ICOS-CP team it has been established that the main capability for any potential RDF store to be utilized at ICOS-CP, is that the support of sophisticated spatial queries is a compulsion. This should be augmented with a support of spatial index to accommodate the future data growth perspective. In terms of spatial relationships, the overlap and within functionality to identify data elements that have overlapping geometric boundaries or completely covered by other geometry are considered to be of prime interest. The crossing of ship trajectories and equality amongst spatial objects could also be topological relationships of interest in the near future. This entails that from the established set of benchmark spatial queries: Q11, Q13, Q22, Q23, Q24, and Q27 are of prime interest from ICOS perspective. Q7, Q8, Q12, and Q25 also hold an element of interest for any potential RDF store to be considered fulfilling the ICOS-CP requirements in near future.

4.4. Introduction to Programming with the RDF Stores

A brief introduction to the technical details of the selected RDF stores and usage of these from programming environment is given in the next few sections.

4.4.1. Eclipse RDF4J

Eclipse RDF4J is an open source Java framework for storage, parsing, inference and query of RDF data. Formerly known as OpenRDF Sesame, it was created by the Dutch software solutions company Aduna before moving to Eclipse foundation in 2016. Eclipse RDF4J can either be used as a Java library to process RDF data internally or as a standalone RDF database. RDF4J offers Java API to connect to other RDF stores as well as SPARQL endpoints offering transparent access to remote RDF repositories and thereby enabling developers to build powerful linked data and semantic web applications. The framework supports all of the mainstream RDF serialization formats including RDF/XML, Turtle, N-Triples, N-Quads, JSON-LD and few others.

Primarily, RDF4J offers two types of transactional RDF databases: an in-memory store and a native store. The RDF4J Memory store is an RDF store residing completely in memory with an optional synchronization to the disk. This is a high performance RDF store for small datasets scaling to the amount of main memory available. The RDF4J Native store uses direct disk for persistence. The native store has a smaller memory footprint; is more scalable solution with better consistency and durability; has default indexing on different subject, predicate, and object combination; and is considered suitable for medium sized datasets around 100 million triples.

RDF4J supports a stacked architecture enabling software components to be stacked on top of the others to extend the functionalities like: RDFS inference, rule based reasoning, full text search as well as geospatial indexing. The abstraction provided by the RDF4J architecture and its vendor neutral APIs have received a considerable attention in the RDF databases community, and many RDF stores use RDF4J framework APIs.

During this study, the RDF4J 2.4.0 was used. The downloaded work package (Eclipse RDF4J – Downloads n.d.) contains: an RDF4J server (the RDF store) to manage RDF data as RDF4J repositories; a workbench web application to connect, query and interface the RDF4J repositories through the RDF4J server; a console application to directly parse, process and query an RDF file; and a set of java libraries. It is also possible to download only the java libraries and use these APIs (JAR files) for internally using RDF4J framework from a java program. For our study, this technique was used and a java project was created as a driver to use the RDF4J java libraries. Version 2.4.0 of RDF4J requires Java 8 host JVM. The wide acceptance of RDF4J framework in the RDF database community is driven from the RDF4J core APIs. Key components of the relevant APIs are discussed in the following sections.

Storage and Interface Layer (SAIL) is an interface for RDF to store statements and evaluate queries over them. Statements can be grouped in named contexts or in the null context. The RDF4J SAIL API (*org.eclipse.rdf4j.repository.sail*) is a collection of interfaces designed for low level transaction access to RDF data. SAIL API enables the decoupling between the database implementation and functional modules like parsers, query engines, end-user API access etc. At the low level, the SAIL operates on query algebra which is an object representation of a SPARQL query. The SAIL provides the *StackableSail* interface, which allows SAIL

implementations to be stacked on top of each other. This provides a chain of responsibility where each underlying SAIL object in the stack implements a specific feature like reasoning, access control, data filtering, query expansion, spatial indexing, persistence etc. At the bottom of the stack, the last implementation in the stack is a sail which cannot be stacked on top of any other. Programmatically, this has been achieved with the interface *StackableSail*. The bottom sail does not implement this interface therefore it cannot appear on top of others, and has to be the bottom layer in the stack. This bottom SAIL object is responsible for the persistence of data. One of the RDF4J stores (native or memory) is this SAIL, while the others which can be stacked above, include: ForwardChainingRDFSInferencer (inferenceing layer), LuceneSAIL (full-text indexing layer), and a few more.

The central point of access for RDF4J compatible RDF stores as well as SPARQL endpoints is the Repository API *(org.eclipse.rdf4j.repository).* The repository framework provides a transparent access to the underlying RDF database with consistent interfaces for storage, query and processing. Eclipse RDF4J itself provides three implementations of these interfaces. *SailRepository* operates on top of a stack of SAIL objects and is used when creating an Eclipse RDF4J local repository. The constructor for *SailRepository* class requires a SAIL object as parameter. *HTTPRepository* acts as a proxy to an RDF4J server repository accessible through HTTP. *SPARQLRepository* is a proxy to a SPARQL endpoint (which is not necessarily implemented with RDF4J). Other than these three, the third party implementation can be provided by anyone interested to extend their database as an RDF4J repository. The RDF4J compatible RDF stores (Openlink Virtuoso, Stardog, Ontotext GraphDB and others) have provided their custom implementation of repository API to expose their platforms in consistence with RDF4J framework.

The core of RDF4J framework is the Model API *(org.eclipse.rdf4j.model)* which defines the building blocks of RDF processing. Some of the important interfaces in this API include: *Statement*, *Resource*, *Literal*, *Value*, *IRI*, *BNode* and more. The Model API provides pre-defined IRIs for well-known vocabularies like RDF, RDFS, OWL, Dublin Core (DC), and Friend of a Friend (FOAF) in the package *org.eclipse.rdf4j.model.vocabulary*. The RDF model is a logical collection of RDF statements. RDF4J *Model* interface is implemented as an extension of Java collection class *java.util.Set<statement>* enabling the use of *Model* as any other java collection. The parsing toolkit in RDF4J is Rio consisting of many modules for each of the specific syntax. The java code developed in this research for RDF4J tests is available online at https://github.com/Raza-Amir-Syed/TestGeoRDFStores/tree/master/RDF4JDriver.

4.4.2. Apache Jena

Apache Jena is an open source java framework for building semantic web applications. It provides extensive Java libraries to facilitate developers to handle RDF, RDFS, RDFa, OWL and SPARQL as well as rule based inference and reasoning along with a variety of storage strategies for RDF stores. Jena was originally developed in HP labs UK in 2000 before moving to Apache

software foundation in 2010. The Jena work package for version 3.9.0 used for this study includes following artifacts:

- TDB is a high performance persistence store supporting full range of Jena APIs. A TDB is a Jena RDF store that can be directly accessed from a single machine. TDB supports transaction and protected against corruption. There are command line scripts as well as Jena APIs for management of TDB.
- Fuseki is a SPARQL server component that can use TDB as underlying persistent storage and also enable the access from multiple applications. Fuseki can be launched as a standalone server, a web application or an operating system service and it exposes management interface for server monitoring and administration.
- ARQ is a SPARQL query processing engine for Jena. It supports SPARQL 1.1 as well as SPARQL graph store protocol.
- RDF API consists of core packages and interfaces used for RDF data storage and processing in Jena. For our study, this API was used for development of a driver program to conduct SPARQL benchmark tests for query performance.
- The work package also contains Ontology API as well as Inference API to add custom semantics as well as inference and reasoning on the RDF data.

In Jena APIs, the Model class denotes an RDF graph and it contains the collection of RDF triples. It is an abstraction over different ways to store the graph like memory structures, disk-based persistent stores and inference engines etc. At lower levels, Jena uses another interface Graph for simpler abstraction and lower level interaction. The required methods and interfaces to manage the RDF data can be acquired from the Model object for processing an RDF graph.

An RDF Dataset has one or more graphs with one designated as a default graph. In Jena, the *Dataset* class represents an RDF Dataset which contains *Models*, one of which is the default *Model*. Each Dataset has an associated file location (a folder) where the data is stored. TDB datasets can be created from the static methods in *TDBFactory* class. The java code developed in this research for Jena tests is available online at https://github.com/Raza-Amir-Syed/TestGeoRDFStores/tree/master/JenaRDFDriver.

4.4.3. Openlink Virtuoso

Openlink Virtuoso is a cross platform web server, a file server, and a database server in a single multithreaded server process. Therefore it is more suitably defined as a universal data access middleware. Virtuoso offers a high performance virtual database engine on an underlying distributed architecture. On their homepage, the Openlink community defines virtuoso as a “Data Junction Box that drives enterprise and individual agility by deriving a Semantic Web of Linked Data from existing data silos”. In contemporary information era, data processing requires traversal over heterogeneous data sources spread over many different platforms. Virtuoso offers a cost-effective platform for projection of data from many different sources. Virtuoso offers both

commercial and open source editions. For this study, the open source edition of Openlink Virtuoso version 7.2.4 is used.

The Virtuoso quad store (RDF store) is built on top of an RDBMS. From version 5.0.7 Virtuoso can be used as an RDF store. The RDF processing middleware manages the RDF data in RDBMS tables and different database types are used (*IRI_ID, RDF_BOX* etc.) for management of RDF statements on an RDBMS platform. The main tables for RDF statements include *RDF_QUAD, RDF_PREFIX, RDF_OBJ* and a few more. Virtuoso also supports R2RML expression language for mapping of relational databases to RDF datasets.

The RDF capability includes built-in support for SPARQL and SPARUL. Starting from Virtuoso version 4.5, a SPARQL query can be used in place of any SQL query by appending a key word "SPARQL" followed by a space in front of the query text. Such queries are sent to SPARQL query processor. A linked data middleware Sponger is also integrated into the SPARQL processor for URI de-referencing within SPARQL query patterns. Openlink Virtuoso does not support Unicode in SPARQL and comments inside SPARQL are also not supported, however some other extension are made available.

Virtuoso offers a number of APIs and data access connection methods on different platforms like Java, .Net and others. To establish a client connection with Virtuoso server, the options include ODBC, JDBC as well as OLEDB. In this study the JDBC connection was used. For JDBC client connections, the *Java.sql* package can be used in consistence with any other JDBC technology. A JDBC connection to the virtuoso server is obtained through the *DriverManager* class by providing the host, port, user and password parameters. This connection can then be used for creating statement objects and executing query on these statements. The queries can be SQL or SPARQL, differentiated by the first key word "SPARQL" in front of the query.

For direct RDF store processing, Openlink Virtuoso has provided Data Access Providers. Virtuoso 7.2.4 supports three drivers in this regards: Virtuosos Jena Provide, Virtuoso Sesame/RDF4J Provider as well as Virtuoso Redlands provider. For this study Virtuoso Sesame provider is used as shown in figure 5-1 (Virtuoso – Sesame Provider n.d.).

The Virtuoso Sesame provider leverages the Sesame/RDF4J framework to process the Virtuoso RDF store using Java language. Therefore, while the underlying data management and processing is being conducted by the Virtuoso server, the RDF4J java framework can be used in programming environment for RDF4J friendly java code. For this purpose Virtuoso has provided a java library which exposes a *VirtuosoRepository* class which is consistent with the RDF4J Repository API. It is however important to note that the Virtuoso Sesame/RDF4J provider also uses underlying JDBC for client connection to the virtuoso server. The java code developed in this research for Virtuoso tests is available online at https://github.com/Raza-Amir-Syed/TestGeoRDFStores/tree/master/VirtuosoDriver.

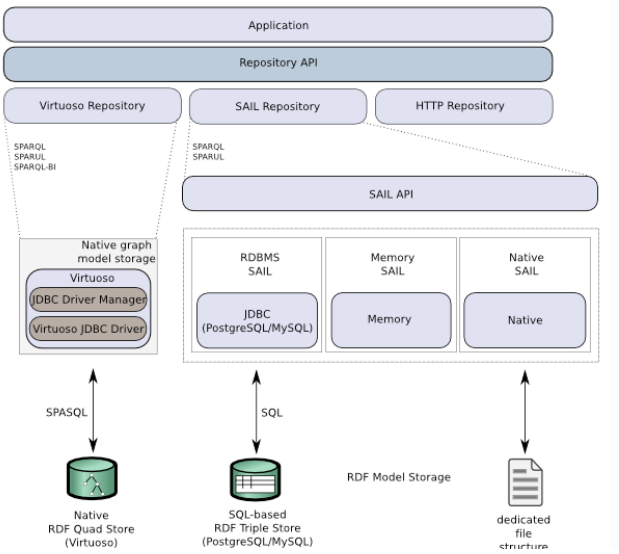


Figure 4-1 Virtuoso Sesame Stack

4.4.4. Stardog

Stardog defines itself as an Enterprise Knowledge Graph platform capable to manage the enterprise data in full generality, which is scale-able and connected from many diverse, distributed and heterogeneous data sources as unification platform. Stardog claims to make the transformation of enterprise data into knowledge at a faster pace. Stardog prefers to call itself a Knowled Graph (a graph database with knowledge toolkit). Stardog whitepapers also suggest that the proper way of managing the data silos in an enterprise is a knowledge graph and that is why graph is the model for next 20 years.

Stardog data model is an RDF graph and it supports SPARQL query as well as other basic graph data support like inference and reasoning etc. While graph is the data model for Stardog, it also supports property graph model and Gremlin graph traversal language. Stardog also supports Virtual graphs by re-writing SPARQL queries to SQL and transforming the tabular results back to RDF. A number of RDBMS are supported in this manner. Stardog version 6.0.1 Enterprise Edition (60 days trial) has been used for this study. Stardog server runs in a java container. The *stardog-admin.bat* script can be used to start or stop the server as well as other functions like creating a database, running a query etc. from the command window. The Stardog management studio (a web application) can also be accessed at http://localhost:5820.

Stardog supports programming interfaces from Java, over HTTP, Javascript, Clojure, Groovy, Spring and .Net. Stardog recommends SNARL API as native and preferred method of programming. Other than SNARL, Stardog also supports interface with Sesame/RDF4J and Jena frameworks through APIs. For network connections, the SPARQL HTTP protocol from Stardog is default for client connections. Stardog also supported another network protocol, ‘SNARL’ in the earlier version but it has been depreciated since Stardog 4.2. It is important to mention here that SNARL network protocol and SNARL Java API are two different artifacts. The SNARL

network protocol is depreciated, while SNARL Java API is still the preffered way of programming from Java. For this study both the SNARL API and the Sesame/RDF4J API have been used separately to test the SPARQL benchmark queries.

When programming from Java, one way to connect the Stardog server is from HTTP. The server can reside on the same machine or it can be a remote Stardog server instance available over HTTP. The other method is to run a Stardog server instance within the same JVM as the Java program. This is known as an embedded server and it helps avoid some of the HTTP overhead when there is a local server. The SNARL API exposes methods to obtain connection to the embedded server and use the connection for subsequent database creation; data upload as well as run SPARL queries through these local connections. Stardog Sesame/RDF4J API exposes a *StardogRepository* class which can connect to an embedded server or to a remote Stardog server over HTTP. This class is in compliance with RDF4J *Repository* class and therefore the RDF4J/Sesame Java framework can be used with this repository object for subsequent RDF operations. The java code developed in this research for Stardog tests is available online at https://github.com/Raza-Amir-Syed/TestGeoRDFStores/tree/master/StardogDriver.

4.4.5. Ontotext GraphDB

GraphDB is a robust and scalable graph/RDF database capable to balance the use of linked data cloud datasets as well as local resources. GraphDB implements RDF4J interfaces and supports W3C SPARQL 1.1 protocol specifications as well as RDF serialization formats. GraphDB can perform semantic inference at scale allowing users to derive new semantic facts from existing facts. GraphDB offers three editions: Free, Standard and Enterprise. Our evaluation uses the free edition of version 8.8.0. It is important to highlight that Free version is free for use but not licensed as open source. The GraphDB server runs in a Java container and it also includes a workbench web application for managing the database and other administration tasks. The workbench also offers tools to explore data as well as class relationships and properties from the vocabularies and ontologies used in the data.

GraphDB is packaged as a Storage and Inference Layer (SAIL) for RDF4J and makes extensive use of the RDF4J framework features. GraphDB implements the SAIL API interface so that it can be integrated with the rest of the RDF4J framework. A user application can be designed to use GraphDB directly through the RDF4J SAIL API or via the higher-level functional interfaces. When a GraphDB repository is exposed using the RDF4J HTTP Server, users can manage the repository through the GraphDB embedded Workbench, or the RDF4J Workbench, or other tools integrated with RDF4J. The java code developed in this research for GraphDB tests is available online at https://github.com/Raza-Amir-Syed/TestGeoRDFStores/tree/master/GraphDBDriver.

5. RESULT

5.1. Eclipse RDF4J

5.1.1. Geospatial Support

The GeoSPARQL spatial support is enabled in RDF4J by inclusion of *rdf4j-queryalgebra-geosparql* library in the server class path. The WKT serialization format is supported for representation of geographic data and the spatial support is provided in all types of stores. However, for optimization of spatial queries, the spatial indexing is only available in repositories created with *LuceneSail* or its derivatives (*SolrSail* and *ElasticSearchSail*). Although Lucene is basically a full text indexing framework, however the *lucene-spatial-extras* module handles the spatial indexing of geometries. By default, LuceneSail spatially indexes only the fields represented by the predicate *http://www.opengis.net/ont/geosparql#asWKT* using R-tree. Additional fields for indexing can be specified through the *LuceneSail.WKT_FIELDS* parameter. Spatial4J library is used for conversion to and from WKT and shape objects while the spatial algebra is handled by the JTS library. RDF4J supports a rich set of topological and non-topological GeoSPARQL compliant spatial functions which are available both in indexed as well as non-indexed configuration as listed below:

- Non-toplogical functions include *geof:distance, geof:boundary, geof:buffer, geof:convexHull, geof:difference, geof:envelope, geof:intersection, geof:getSRID, geof:symDifference, geof:union,* and *geof:relate.*
- Simple feature topology functions include *geof:sfEquals, geof:sfDisjoint, geof:sfIntersects, geof:sfTouches, geof:sfCrosses, geof:sfWithin, geof:sfContains,* and *geof:sfOverlaps.*
- Eigenhofer topology functions include *geof:ehEquals, geof:ehDisjoint, geof:ehMeet, geof:ehOverlap, geof:ehCovers, geof:ehCoveredBy , geof:ehInside,* and *geof:ehContains.*
- RCC8 topology functions include *geof:rcc8eq, geof:rcc8dc, geof:rcc8ec, geof:rcc8po, geof:rcc8tppi, geof:rcc8tpp, geof:rcc8ntpp,* and *geof:rcc8ntppi.*

5.1.2. Benchmark Query Performance

Eclipse RDF4J supports spatial search and query in both indexed as well as non-indexed spatial configurations. However during the tests, it was found that LucenSail is not responding properly for spatial indexing and instead of improved performance, the spatial queries became nearly un-responsive in LuceneSail configuration. The matter was reported to RDF4J development team, and an issue was created at Github (https://github.com/eclipse/rdf4j/issues/1160). At the time of this writing, the issue is not yet resolved; hence the tests are only performed on an RDF4J Native store without spatial indexing. Another error in RDF4J 2.4.0 was found in the buffer function and the issue was already reported (https://github.com/eclipse/rdf4j/issues/1128). Later on, the buffer function issue has been rectified in 2.4.1, however Q4 and Q5 in our tests were not

executed because the latest software artifacts downloaded when this study was initiated pertains to version 2.4.0. The results for benchmark queries are given in Table 5-1. Some queries perform a little better on machine A, while others have performed considerably better on machine B. The overall performance gain on machine B compared to machine A is nearly 65 times faster.

Table 5-1 Average time in Milli Seconds for benchmark queries in Eclipse RDF4J

Benchmark Query		**Machine A Intel Dual Core 2 GB memory**	**Machine B Intel Core i5 4 GB memory**
Non topological functions			
Q1	Boundary	1.01	1.42
Q2	Envelope	0.98	2.44
Q3	Convex	0.86	0.62
Q4	Buffer	Bug in the buffer function	
Q5	Buffer		
Spatial Selection			
Q7	Equals	48.11	3.57
Q8	Equals	3.49	3.22
Q9	Intersect	8.65	5.53
Q10	Intersect	1.25	3.50
Q11	Overlaps	1.13	4.55
Q12	Crosses	1.24	2.59
Q13	Within	16.95	5.07
Q15	Near a Point	1.59	2.62
Q16	Disjoint	1.05	5.21
Q17	Disjoint	1.03	2.60
Spatial Joins			
Q18	Equals	11.86	11.22
Q19	Intersects	7110.93	8.57
Q20	Intersects	17.75	9.38
Q21	Intersects	8.38	9.60
Q22	Within	10.58	7.97
Q23	Within	434.45	7.45
Q24	Within	41.44	8.39
Q25	Crosses	7.86	8.79
Q26	Touches	1620.35	9.92
Q27	Overlaps	1.59	12.68

5.2. Apache Jena

5.2.1. Geospatial Support

Spatial query in Jena is supported by the spatial extension since version 2.11.0. It has lately been notified on the Apache Jena website that Jena spatial query is planned to retire (Jena – Spatial

Query n.d.), in favor of geosparql-jena. For our study however, the Jena spatial query extension was used, because the study artifacts were downloaded in October 2018, few months before the notification was stated.

In Jena spatial query extension, the spatial search and query is enabled with the creation of a spatial index. There can be two types of indices: Apache Lucene for same machine or Apache Solr for large scale enterprise level search. The advantage of using Lucene Spatial is that Jena already uses Lucene for text indexing, so the same system is used for spatial indexing as well.

The *SpatialDatasetFactory* class contains static methods to create datasets with Lucene spatial index. A spatial enabled dataset can be created from these methods on top of a TDB as the base dataset. When a spatially indexed dataset is created, than any changes to the dataset triggers the spatial indexing, if the relevant predicates are found in the updated data. By default Jena supports two predicates for geometry literals. The first one is a pair of latitude and longitude: *http://www.w3.org/2003/01/geo/wgs84_pos#lat, http://www.w3.org/2003/01/geo/wgs84_pos#lon*. The second one is *http://www.opengis.net/ont/geosparql#asWKT* . Custom geo predicates can also be added in Jena spatial, however geometry support is for WKT literals only. In terms of spatial and topological relationships, Jena spatial query has a limited support for GeoSPARQL. The spatial relationships available in the Jena spatial query extension are:

- *spatial:nearby(lat, lon, radius)*
- *spatial:withinCircle(lat, lon, radius)*
- *spatial:withinBox (lat_min, lon_min,lat_max, lon_max)*
- *spatial:intersectBox(lat_min, lon_min,lat_max, lon_max)*
- *spatial:north (lat, lon)*
- *spatial:south (lat, lon)*
- *spatial:west (lat, lon)*
- *spatial:east (lat, lon)*

5.2.2. Benchmark Query Performance

Amongst the tested RDF stores, Jena covers the minimum number of benchmark queries. The benchmark queries Q1, Q2, Q3, Q4, Q5 and Q6 were not executed because no related utility spatial functions for *Boundary*, *Envelope*, *ConvexHull* and *Buffer* of geometries are available in Jena. The spatial selection queries Q7, Q8, Q11, Q12, Q16 and Q17 were also not executed because Jena spatial extension has no equivalent spatial relations for: E*quals*, *Overlap*, *Cross* and *Buffer*. Although Q9 and Q10 queries were evaluated, but the second parameter to the related intersects function is a box represented by two latitude and longitude pairs. In the other evaluated RDF stores, the second parameter of corresponding relations can be a geometry literal or a geometry variable as well. The spatial join queries Q18 to Q27 are not available in Jena because a join requires the second parameter of topological functions to be geometry variables. As the spatial search in Jena is not available without the spatial index, the comparison between indexed

and un-indexed configurations could not be drawn. Furthermore, all queries performance (Table 5-2) deteriorated on machine B with an overall performance decrease of 5 times.

Table 5-2 Average time in Milli Seconds for benchmark queries in Apache Jena

Benchmark Query		**Machine A Intel Dual Core 2 GB memory**	**Machine B Intel Core i5 4 GB memory**
Non topological functions			
Q1, Q2,Q3, Q4, Q5, Q6		Not Supported	
Spatial Selection			
Q7, Q8		Not Supported	
Q9	Intersect	3.63	5.98
Q10	Intersect	3.74	5.74
Q11, Q12		Not Supported	
Q13	Within	3.78	9.14
Q15	Near a Point	7.73	69.76
Q16, Q17		Not Supported	
Spatial Joins			
Q18, Q19, Q20, Q21, Q22, Q23, Q24, Q25, Q26, Q27		Not Supported	

5.3. Openlink Virtuoso (Universal Server)

5.3.1. Geospatial Support

Openlink Virtuoso version 6.01.3126 onward has supported geospatial capabilities which are focused specifically for spatial data in RDF but can also be used in SQL. Virtuoso 7.1 onwards support the WKT literal representation for geometries with a few exceptions for some shapes. Another supported geometry literal is BOX with two pairs of longitude and latitude. Virtuoso assigns a special internal type *virtrdf:Geometry*, for managing geometry. All WKT literals are converted to this type and such literals are automatically indexed in a two dimensional R-tree containing all distinct geometries occurring in any quad of any graph under any predicate.

The default reference system is WGS-84 with coordinates in degrees longitude and latitude. The *ST_Transform* function is provided for coordinate transformation but it requires the v7proj4 plugin, while *ST_SetSRID* is used for altering SRID without altering coordinates. There are some other utility functions as well. The spatial and topological relationship functions are:

- *bif:st_intersects* is used for checking if two shapes intersect.
- *bif:st_within* is used for checking if a shape lies completely inside another shape.
- *bif:st_contains* is used for checking if a shape completely bounds another shape.
- *bif:st_distance* is used to find distance between two shapes.

5.3.2. Benchmark Query Performance

The tests for Openlink virtuoso have been conducted from two different environments: one with a simple JDBC connection and the other with RDF4J provider. Both of these configurations have been used on two different hardware machines, and the results are presented in Table 5-3. The benchmark query Q1 has been evaluated with the function *bif: ST_ExteriorRing*, while Q2 has been executed with the *bif:st_get_bounding_box* function. There are no functions for *ConvexHull, Buffer* and *Equal*, therefore Q4, Q5, Q7 and Q8 are not covered. Similarly Q11, Q12, Q16, Q17 and Q18 are not covered because of no equivalent relationships for *Overlap, Cross, Disjoint* and *Equal.* Spatial join queries Q25, Q26 and Q27 were also not executed because of missing support of equivalent relationship.

Virtuoso does not extend any *Nearby* relationship extension function, however Q15 has been executed using the *bif:st_distance* function. All queries have better performance results on Machine B in both environments. The overall performance Machine B is around two time faster than Machine A in both configurations.

Table 5-3 Average time in Milli Seconds for benchmark queries in Openlink Virtuoso

Benchmark Query		**Machine A Intel Dual Core 2 GB memory**		**Machine B Intel Core i5 4 GB memory**	
		JDBC	**RDF4J Provider**	**JDBC**	**RDF4J Provider**
Non topological functions					
Q1	Boundary	5.18	5.59	4.26	2.82
Q2	Envelope	2.40	2.46	1.67	1.34
Q3, Q4, Q5		Not Supported			
Spatial Selection					
Q7, Q8					
Q9	Intersect	2.93	3.18	1.32	1.39
Q10	Intersect	5.29	5.66	2.06	2.16
Q11, Q12		Not Supported			
Q13	Within	2.01	1.41	0.74	0.75
Q15	Near a Point	2.76	2.71	0.92	1.08
Q16, Q17		Not Supported			
Spatial Joins					
Q18		Not Supported			
Q19	Intersects	12.19	13.10	5.62	5.57
Q20	Intersects	23.99	27.13	12.18	11.80
Q21	Intersects	28.96	30.61	15.00	14.81
Q22	Within	25.70	30.53	13.03	12.57
Q23	Within	31.14	35.15	15.66	15.27
Q24	Within	29.79	38.11	15.55	14.72
Q25, Q26, Q27		Not Supported			

5.4. Stardog (knowledge Graph)

5.4.1. Geospatial Support

In order to enable geospatial support and spatial query, spatial indexing needs to be enabled. The related option is to be specified at the time of creation of the database (listing 5-1). Stardog supports geometry data in WKT format, however to use all the shapes in WKT standard, the JTS library has to be included in the Java class path (listing 5-1).

```
AdminConnection.newDatabase(dbName).set(GeospatialOptions.SPATIAL_ENABLED, true).create();
Stardog.builder().set(GeospatialOptions.USE_JTS,true).create();
```

Listing 5-1 Enable Stardog geospatial and JTS support

When the spatial support is enabled, the data commit triggers Stardog to index all features from relevant vocabularies. The point data can be encoded with the lat, lon predicates of WGS84 vocabulary or as a WKT point literal. Stardog offers five spatial and topological operators, *geof:within, geof: nearby, geof:distance, geof:area and geof:relate* as follows :

- *geof:within* relationship can be used to check if a geometry is inside another geometry.
- *geof:nearby* can be used to check if some object is within a certain distance to a point.
- *geof:area* is a non-spatial utility function for area computation.
- *geof:relate* can be used to test 05 relationships amongst two geometries. The 05 relationships are *geo:contains, geo:within, geo:intersects, geo:equals,* and *geo:disjoint*. The *geof:relate* can be used in two forms. The first form is to make a boolean check to test a relationship i.e. *FILTER(geof:relate(?geom1,?geom2,relationship))*. The second form is to find out which one of the 05 relationship holds i.e. *?rel geof:relate (?geom1 ?geom2)*

5.4.2. Benchmark Query Performance

Query performance in Stardog has been conducted with SNARL API as well as with the RDF4J API. The benchmark queries Q1, Q2, Q3, Q4 and Q5 were not executed because of no equivalent extension functions for *Boundary, Envelope, ConvexHull* and *Buffer*. Q11, Q12, Q25, Q26 and Q27 were also not executed because no relationships for *Overlap, Cross* and *Touch* are provided by Stardog.

It is interesting to note that within relationship in Stardog can be tested with two methods. One uses the syntax "*?geom1 geof:within ?geom2*" and the other method is from inside the relate functions as "*geof:relate(?goem1, ?geom2, geo:within)*". For this evaluation the second method i.e. *goef:relate* is used. The benchmark query set performance is given in Table 5-4. All queries have performed better on the higher specification Machine B, around 3 times faster than Machine A in both configurations.

Table 5-4 Average time in Milli Seconds for benchmark queries in Stardog

Benchmark Query		Machine A Intel Dual Core 2 GB memory		Machine B Intel Core i5 4 GB memory	
		SNARL	**RDF4J**	**SNARL**	**RDF4J**
Non topological functions					
Q1, Q2, Q3, Q4, Q5		Not Supported			
Spatial Selection					
Q7	Equals	109.58	79.91	26.12	26.18
Q8	Equals	15.41	7.63	2.43	2.33
Q9	Intersect	14.13	9.28	2.80	2.67
Q10	Intersect	10.76	7.68	1.81	1.42
Q11, Q12		Not Supported			
Q13	Within	16.60	11.73	3.31	2.75
Q15	Near a Point	19.06	6.28	1.31	1.10
Q16	Disjoint	17.79	58.44	1.76	1.43
Q17	Disjoint	27.23	9.34	1.79	1.53
Spatial Joins					
Q18	Equals	141.80	84.86	18.38	18.49
Q19	Intersects	10329.48	10323.93	3768.82	3772.82
Q20	Intersects	864.51	850.89	254.17	256.48
Q21	Intersects	17.63	13.63	4.98	3.37
Q22	Within	22.81	28.18	4.29	4.16
Q23	Within	307.29	335.19	103.07	101.53
Q24	Within	223.24	242.99	68.51	68.27
Q25, Q26, Q27		Not Supported			

5.5. Ontotext GraphDB

5.5.1. Geospatial Support

GraphDB has the framework of plugins; one of these is GeoSPARQL. It is however important to note that geospatial data, search and query is supported even when the GeoSPARQL plugin is not enabled. When the GeoSPARQL plugin is enabled, the spatial data is indexed and it offers optimized query. GraphDB supports both WKT as well as GML geometry literals in conformance with the GeoSPARQL specification. Each set of spatial functions is provided in two formats: one for the non-indexed query and other for query under spatial indexing. The functions conforming to non-indexed query are always available, while the indexed version of functions is only available when the geospatial index has been enabled. The GeoSPARQL plugin supports two indexes: quad prefix tree and geohash prefix tree.

GraphDB offers each spatial relationship with either a *geof:* namespace or a *geo:* namespace. The relationships in *geof:* namespace are available all the time and they perform search and query without using spatial indexing. The other set of similar relationships with the *geo:*

namespace are only available when spatial indexing is available. Both examples are given in listing 5-2 and 5-3.

```
geof:sfOverlaps(?geoLiteral1,?geoLiteral2)
geof:sfWithin(?geoLiteral1,?geoLiteral2)
```

Listing 5-2 Example of non-indexed functions

```
?geom1 geo:sfOverlaps ?geom2
?geom1 geo:sfwithin ?geom2
```

Listing 5-3 Example of indexed functions

Graph DB also offers geospatial extension for specialized indexing on 2-D geospatial data conforming to WGS84 Geo positioning RDF vocabulary. The extension enables efficient query against this data.

5.5.2. Benchmark Query Performance

GraphDB is the only RDF store in our study which has allowed the evaluation of all the benchmark queries. It also supports the queries in both indexed as well as spatially un-indexed configurations as shown in Table 5-5.

Table 5-5 Average time in Milli Seconds for benchmark queries in GraphDB

Benchmark Query		Machine A		Machine B	
		Non	Indexed	Non	Indexed
Non topological functions					
Q1	Boundary	29.74	5.86	1.08	3.37
Q2	Envelope	18.43	4.78	1.30	4.04
Q3	Convex	6.34	2.99	1.04	1.10
Q4	Buffer	17.51	2.74	1.79	0.60
Q5	Buffer	10.69	3.30	4.70	2.93
Spatial Selection					
Q7	Equals	484.35	6.29	274.97	6.57
Q8	Equals	31.45	6.91	7.33	10.16
Q9	Intersect	62.85	6.48	20.60	14.98
Q10	Intersect	7.94	17.34	2.44	8.16
Q11	Overlaps	9.00	11.49	2.83	15.08
Q12	Crosses	34.68	15.71	3.76	6.81
Q13	Within	79.96	11.56	31.21	15.72
Q15	Near a Point	59.88	8.39	5.39	5.73
Q16	Disjoint	15.55	9.18	3.36	11.12
Q17	Disjoint	8.71	11.95	3.46	6.44
Spatial Joins					
Q18	Equals	31169.61	24.46	3445.23	25.33
Q19	Intersects	54119.74	16.56	33033.93	21.01
Q20	Intersects	96.28	23.24	38.94	16.45
Q21	Intersects	43.95	25.24	19.14	19.02
Q22	Within	82.44	22.11	39.14	20.83
Q23	Within	1796.98	17.69	1008.27	23.68
Q24	Within	313.13	22.81	141.65	24.23
Q25	Crosses	58.20	15.81	28.82	18.17
Q26	Touches	7725.28	24.46	4517.61	20.01
Q27	Overlaps	36.81	17.86	11.10	21.16

Most of the queries in both indexed as well as un-indexed configurations have performed better on higher specifications Machine B. The overall performance of indexed query is around 300 times and 150 times faster than un-indexed configuration on both machines A and B respectively.

5.6 Cross Comparison

Feature wise comparison of RDF stores, drawn during the qualitative study is summarized in the Table 5-6. The results collected in the quantitative portion relates to tests conducted on two different machines: Machine A and Machine B. The cross comparison of these results are depicted in Tables 5-7 and 5-8 on respective machines.

Table 5-6 GeoSPARQL Features compliance for five RDF stores

GeoSPARQL Feature Support	RDF4J 2.4.0	Jena 3.9.0	Virtuoso 7.2.4	Stardog 6.0.1	GraphDB 8.8.0
Storage	Custom	Custom	RDBMS	Custom	Custom
Geometry literals support	WKT	WKT	WKT	WKT	WKT,GML
Spatial analysis GeoSPARQL support	Strong	No	Limited	Limited	Strong
Topological relationships GeoSPARQL support	Strong	Very limited	Limited	Limited	Strong
Spatial relationships syntax conforming to GeoSPARQL	Yes	No	No	Partially	Yes
Spatial query without index	Yes	No	No	No	Yes
Spatial Index Technique	Lucene Spatial	Lucene Spatial, Solr	R-tree	Lucene Spatial	Quad-prefix tree, Geohash

Table 5-7 Cross Comparison of Benchmark Query Performance on Machine A (Intel Dual Core, 2 GB memory)

Benchmark Query		Eclipse RDF4J	Virtuoso		Apache Jena	Stardog		GraphDB	
			RDF4J	JDBC		SNARL	RDF4J	No	Index
Non-topological queries									
Q1	Boundary	1.01	5.59	5.18				29.74	5.86
Q2	Envelope	0.98	2.46	2.40				18.43	4.78
Q3	Convex	0.86						6.34	2.99
Q4	Buffer							17.51	2.74
Q5	Buffer							10.69	3.30
Spatial Selection queries									
Q7	Equals	48.11				109.58	79.91	484.35	6.29
Q8	Equals	3.49				15.41	7.63	31.45	6.91
Q9	Intersect	8.65	3.18	2.93	3.63	14.13	9.28	62.85	6.48
Q10	Intersect	1.25	5.66	5.29	3.74	10.76	7.68	7.94	17.34
Q11	Overlaps	1.13						9.00	11.49
Q12	Crosses	1.24						34.68	15.71
Q13	Within	16.95	1.41	2.01	3.78	16.60	11.73	79.96	11.56
Q15	Near a Point	1.59	2.71	2.76	7.73	19.06	6.28	59.88	8.39
Q16	Disjoint	1.05				17.79	58.44	15.55	9.18

Q17	Disjoint	1.03				27.23	9.34	8.71	11.95
Spatial Join queries									
Q18	Equals	11.86				141.80	84.86	31169.61	24.46
Q19	Intersects	7110.93	13.10	12.19		10329.48	10323.93	54119.74	16.56
Q20	Intersects	17.75	27.13	23.99		864.51	850.89	96.28	23.24
Q21	Intersects	8.38	30.61	28.96		17.63	13.63	43.95	25.24
Q22	Within	10.58	30.53	25.70		22.81	28.18	82.44	22.11
Q23	Within	434.45	35.15	31.14		307.29	335.19	1796.98	17.69
Q24	Within	41.44	38.11	29.79		223.24	242.99	313.13	22.81
Q25	Crosses	7.86						58.20	15.81
Q26	Touches	1620.35						7725.28	24.46
Q27	Overlaps	1.5875						36.81	17.86

Table 5-8 Cross Comparison of Benchmark Query Performance on Machine B (Intel Core i5, 4 GB memory)

Benchmark Query		**Eclipse RDF4J**	**Virtuoso**		**Apache Jena**	**Stardog**		**GraphDB**	
			RDF4J	**JDBC**		**SNARL**	**RDF4J**	**No Index**	**Index**
Non-topological queries									
Q1	Boundary	1.42	2.82	4.26				1.08	3.37
Q2	Envelope	2.44	1.34	1.67				1.30	4.04
Q3	Convex	0.62						1.04	1.10
Q4	Buffer							1.79	0.60
Q5	Buffer							4.70	2.93
Spatial Selection queries									
Q7	Equals	3.57				26.12	26.18	274.97	6.57
Q8	Equals	3.22				2.43	2.33	7.33	10.16
Q9	Intersect	5.53	1.39	1.32	5.98	2.80	2.67	20.60	14.98
Q10	Intersect	3.50	2.16	2.06	5.74	1.81	1.42	2.44	8.16
Q11	Overlaps	4.55						2.83	15.08
Q12	Crosses	2.59						3.76	6.81
Q13	Within	5.07	0.75	0.74	9.14	3.31	2.75	31.21	15.72
Q15	Near a Point	2.62	1.08	0.92	69.76	1.31	1.10	5.39	5.73
Q16	Disjoint	5.21				1.76	1.43	3.36	11.12
Q17	Disjoint	2.60				1.79	1.53	3.46	6.44
Spatial Join queries									
Q18	Equals	11.22				18.38	18.49	3445.23	25.33
Q19	Intersects	8.57	5.57	5.62		3768.82	3772.82	33033.93	21.01
Q20	Intersects	9.38	11.80	12.18		254.17	256.48	38.94	16.45
Q21	Intersects	9.60	14.81	15.00		4.98	3.37	19.14	19.02
Q22	Within	7.97	12.57	13.03		4.29	4.16	39.14	20.83
Q23	Within	7.45	15.27	15.66		103.07	101.53	1008.27	23.68
Q24	Within	8.39	14.72	15.55		68.51	68.27	141.65	24.23
Q25	Crosses	8.79						28.82	18.17
Q26	Touches	9.92						4517.61	20.01
Q27	Overlaps	12.68						11.10	21.16

Table 5-9 and 5-10 highlight the cross comparison with the focus on benchmark queries of interest to ICOS-CP metadata management as given in section 4.3.4. The cells highlighted in background color in table 5-9 and 5-10, represent the best performing platform for each individual query (represented in each row).

Table 5-9 Query Results with ICOS Focus on Machine A

Query			Virtuoso			Stardog		GraphDB	
No.	Operation	RDF4J	RDF4J		Jena	SNARL	RDF4J	No Index	Index
Prime Focus									
Q11	Overlaps	1.13						9.00	11.49
Q13	Within	16.95	1.41	2.01	3.78	16.60	11.73	79.96	11.56
Q22	Within	10.58	30.53	25.70		22.81	28.18	82.44	22.11
Q23	Within	434.45	35.15	31.14		307.29	335.19	1796.98	17.69
Q24	Within	41.44	38.11	29.79		223.24	242.99	313.13	22.81
Q27	Overlaps	1.5875						36.81	17.86
Potential Future requirements									
Q7	Equals	48.11				109.58	79.91	484.35	6.29
Q8	Equals	3.49				15.41	7.63	31.45	6.91
Q12	Crosses	1.24						34.68	15.71
Q25	Crosses	7.86						58.20	15.81

Table 5-10 Query Results with ICOS Focus on Machine B

Query			Virtuoso			Stardog		GraphDB	
No.	Operation	RDF4J	RDF4J	JDBC	Jena	SNARL	RDF4J	No Index	Index
Prime Focus									
Q11	Overlaps	4.55						2.83	15.08
Q13	Within	5.07	0.75	0.74	9.14	3.31	2.75	31.21	15.72
Q22	Within	7.97	12.57	13.03		4.29	4.16	39.14	20.83
Q23	Within	7.45	15.27	15.66		103.07	101.53	1008.27	23.68
Q24	Within	8.39	14.72	15.55		68.51	68.27	141.65	24.23
Q27	Overlaps	12.68						11.10	21.16
Potential Future requirements									
Q7	Equals	3.57				26.12	26.18	274.97	6.57
Q8	Equals	3.22				2.43	2.33	7.33	10.16
Q12	Crosses	2.59						3.76	6.81
Q25	Crosses	8.79						28.82	18.17

5.7. Variation in Result Sets

Performance of queries in terms of time consumed during query processing is an aim of this study; however the analysis of result sets returned from each query is not a focus of this study. None the less few broad observations are recorded about the result sets returned from different queries and these are briefly highlighted in the next few paragraphs.

The data sets returned from spatial selection and join queries for Eclipse RDF4J and GraphDB are consistent with each other. Apache Jena only supported four queries from our set of benchmark queries, but data sets returned as query output for these queries are also found to be consistent with Eclipse RDF4J and Graph DB. An important observation is this manner is that for the benchmark queries Q9, Q10 and Q13, the second parameter in RDF4J and GraphDB is polygon geometry, while in case of Jena it is a box. But the data sets returned from these queries in Jena are also consistent with RDF4J. On the contrary the data sets returned from quite a few queries in Virtuoso and Stardog contain considerable mismatches amongst themselves as well as with the other three RDF stores. Also it is found that the result set of queries Q21 and Q25 for GraphDB without indexing are slightly different than the result set of the same queries in indexed configuration. The underlying reasons for such mismatches in result sets could include different logic in execution of the relevant topological operations or the precision of calculation, but a thorough investigation on this subject is required to establish the exact causes.

One more important reflection in this regards is when the result set of spatial join queries were observed. In Eclipse RDF4J and GraphDB the result set of a *within* operation is found as a subset of the result set of *intersect* operation. However, when the same operations were observed for Stardog, it was noted that the result sets returned from *within* operation are not included in the result set of *intersect* operations. Therefore, for Stardog the results of *intersect* and *within* operations are disjoint. The investigation of exact reasons for these differences in query results returned in different RDF stores for similar operations, require a more focused study on this subject, which was beyond the aims of our study.

6. DISCUSSION

All the five tested RDF stores provide geospatial support. Other than Jena, the rest of the four platforms have a reasonable set of functionality in their respective spatial extensions. With regards to GeoSPARQL however, GraphDB and RDF4J offer the strongest compliance. Jena is least compliant in the tested products, while Virtuoso is also assessed as weak in terms of GeoSPARQL compliance. Stardog support to GeoSPARQL is better than Virtuoso, however it is also not be categorized in the strong GeoSPARQL compliant category. Furthermore, while this study was near the conclusion stages, GeoSPARQL extensions for newer version of both Jena and Virtuoso have been announced. This requires a re-investigation of Jena as well as Virtuoso with these new extensions.

With respect to query performances, logically it is assumed that performance of all RDF stores on machine B (higher computing power) should be better in comparison to performance on machine A. The query performance results for RDF4J, Virtuoso, Stardog and GraphDB (both indexed and un-indexed) conform to this assumption and a reasonable gain in performance for all or most queries on machine B is observed in these four platforms. In case of Jena however, the performance comparison is a surprise, as the performance of all four queries was inferior on machine B by 5 times (500%) compared to machine A. The reasons for Jena performance deteriorating on a machine with higher specifications could not be established in this study. With regards to optimization, the only RDF store where indexed versus un-indexed contrast could be drawn was GraphDB, and it reflected a 300 times faster query in indexed configuration on machine A while the performance gain with indexing on machine B is around 150 times.

In terms of stating the RDF store that could be rated as best with regards to overall query performance, the conclusion is tricky. For the 25 established benchmark queries, only GraphDB and RDF4J supported all the queries while rest of the RDF stores only supported a subset of these as depicted in table 6-1. It is evident from the table that the actual comparison on the query performances is practical for RDF stores which support all queries i.e. GraphDB and RDF4J only. Insight into this performance comparison is depicted in Tables 5-7 and 5-8 for machine A and machine B respectively. On machine A, most of the queries individually perform better on the RDF4J platform, however for few spatial join queries like Q23 and Q26 along with one spatial selection query Q19, GraphDB with indexing has a massive performance difference. Due to this huge difference in these three queries, the overall performance indicator on machine A is in the favor of GraphDB with indexing. On machine B however, RDF4J clearly performs best. This behavior depicts that with the change in hardware resources, the performance indicators for the RDF stores change.

Table 6 -1 Benchmark Query Support on Selected Platforms

Benchmark query	Coverage
Q9	All Five RDF Stores
Q10	
Q13	
Q15	
Q7	Four RDF Stores RDF4J, Virtuoso, Stardog, GraphDB
Q8	
Q19	
Q20	
Q21	
Q22	
Q23	
Q24	
Q1	Three RDF Stores RDF4J, Virtuoso, GraphDB
Q2	
Q16	Three RDF Stores RDF4J, Stardog, GraphDB
Q17	
Q18	
Q3	Two RDF Stores RDF4J, GraphDB
Q11	
Q12	
Q25	
Q26	
Q27	
Q4	Supported by two Stores (RDF4J & Stardog) Executed on one Store (GraphDB)
Q5	

The variation in RDF store performance can also be associated with the size of the data set under processing. The issues relating to variations in performance on different hardware could be related to a few factors like: in memory processing, caching or other implementation details. However official documentation of the tested RDF stores does not provide much insight about such behavior.

In order to explain the methodology of query performance measurement in this study, Table 6-2 and 6-3 consisting of two query performance charts are presented.

Table 6 -2 Q13 repeated performance charts - No. of iterations versus nanoseconds

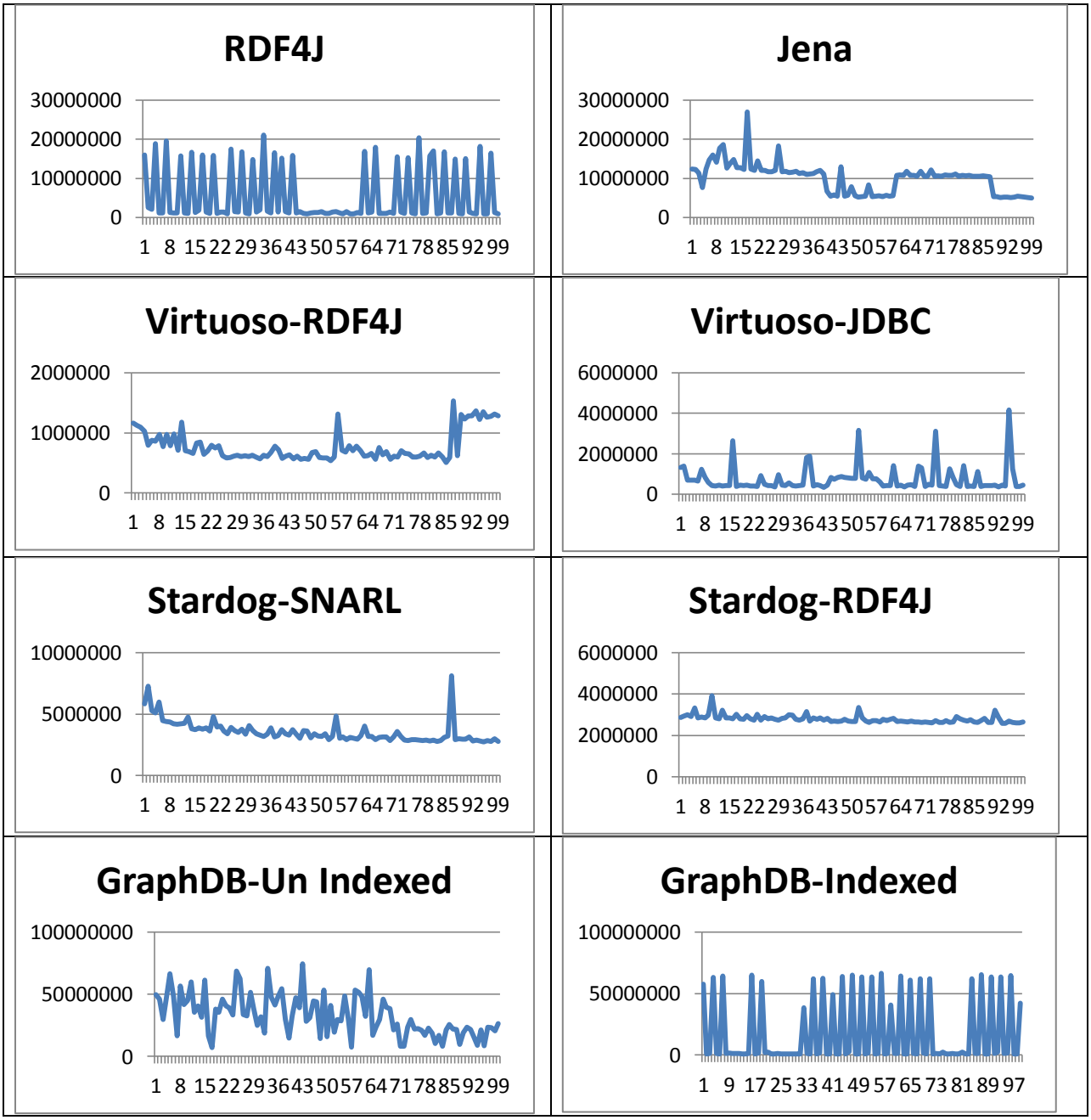
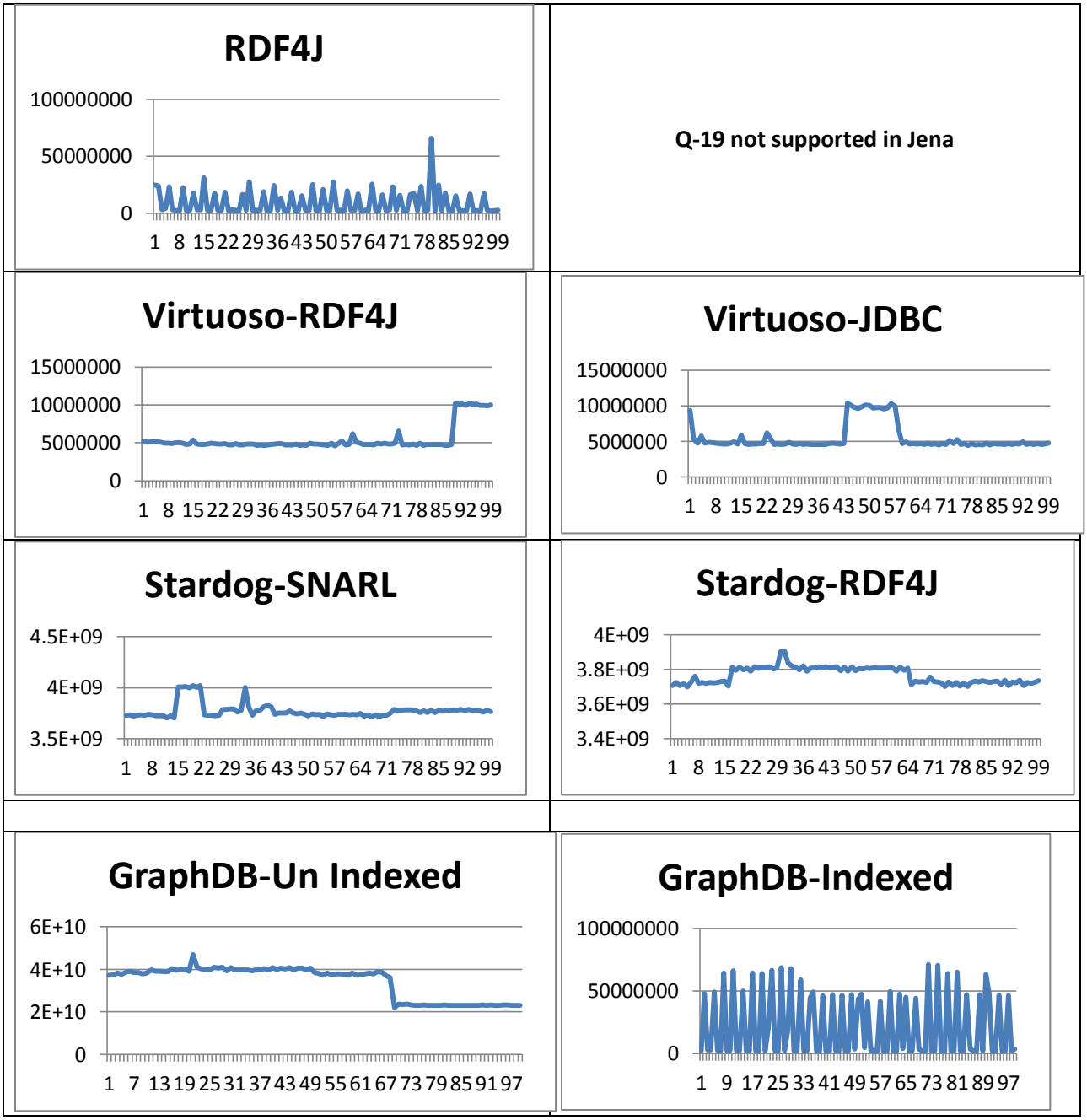


Table 6 -3 Q19 repeated performance charts - No. of iterations versus nanoseconds



During the study, when the query set was executed for more than one times, it was observed that each iteration yielded a different query time for the same query. Many times these query times differed considerably. Hence, the decision of which iteration to be taken as the standard query performance is important and can implicate our performance analysis seriously. Table 6-2 presents the graphical depiction of benchmark query Q13 performance on each platform when executed repeatedly. The x-axis represents the iteration number and y-axis shows the query time in Nano seconds, for each iteration. Table 6-3 presents the same statistics for Q19. The graphs

for only two queries are shown here, one query from spatial selection group and one from spatial join group. However the patterns discussed for these two, is generally applicable to all other queries as well.

In different query processors, it is normal that the first time the query is submitted, the query processor negotiates an execution plan; hence the first time a specific query is executed, it is expected to take more time compared to the next execution of the same query, as the next iteration could find an execution plan already established. However in our tests it was observed that the first iteration was not necessarily the one with the highest query time as evident in many of the charts given in Table 6-2 and 6-3. Furthermore it can also be observed from these charts that query times are not stable in each iteration and at random iterations the query times could be high, and at others times these can be low, with considerable differences. The documentation available from the developers of tested RDF stores did not offer details of such behavior or any associated implication like caching, query plans etc. It was therefore established that instead of taking any single run as the value for a specific query performance, an average over a repeated execution may be considered as the standard unbiased execution time and it could balance out the random spikes as visible in the graphs in Table 6-2 and 6-3. In addition to average over a number of iterations, it was also considered that the JVM requires necessary memory allocations as well as other initializations actions. Some queries also depict more random behavior in the few initial iterations. In light of these, following methodology was devised for query performance statistics:

- The complete benchmark query set is executed for one hundred times on each RDF store.
- Therefore for each query, one hundred different query times are recorded.
- From these one hundred, the starting twenty iterations are considered as initialization, warm up and stabilizing the query processor activity.
- The next eighty iterations are averaged, and this average is considered the standard query times for an individual query for the sake of result statistics.

The product manuals and related documentation from the product vendors did not cover the mechanics behind the query processors that could be responsible for different query performance over repeated executions. A more focused study targeting the insight into the query processing could be conducted to establish these factors. For our study the average over a number of runs was used only to balance out the bias in different runs.

In chapter 3, the state of affairs with respect to previous studies was highlighted and it was noted from (Athanasiou et al. 2013) that the data to be uploaded for test queries, required to be transformed to each of the RDF stores native type before. For our study however, the same datasets has been used in all RDF stores without any transformations. This has been possible because all the five RDF stores tested in our study are found to be supporting the WKT geometry literals with the same namespace prefix and other syntax. While the study in 2013 evaluated that Virtuoso and GraphDB (OWLIM) only supported point geometries at that time, the support for

different geometry shapes is covered in all the five RDF stores that we have tested in this study. Therefore all five RDF stores were able to process the point, line string and polygon data. The GML literals however are only supported by GraphDB at this time.

GraphDB has considerably advanced in terms of GeoSPARQL support as compared to its predecessor OWLIM since 2013. While it supported only point geometry with handful of topological and spatial relationship not conforming to GeoSPARQL in 2013, the platform is now providing the broadest coverage of GeoSPARQL standards amongst the five tested in this study. For GraphDB and RDF4J our devised benchmark SPARQL queries did not require any transformation, as the topological relationship functions extended by both of these platforms are consistent with GeoSPARQL syntax. For rest of the three RDF stores (Virtuoso, Stardog and Jena), all queries needed to be amended for each platform because the GeoSPARQL conformance in this regards is not present in these platforms.

An important utilization of this study is to evaluate the suitability of the selected RDF stores for the geometric part of the ICOS-CP metadata to be exposed as linked open data. Table 5-9 and 5-10 in the results chapter provide an insight in this regards. The ICOS-CP search and query requirements appear to be fulfilled completely by GraphDB and RDF4J. Stardog and Virtuoso can also be utilized as they offer all *Within* and *Equals* queries; however *Overlap* and *Crosses* operations are not supported in these platforms. In terms of query performance, Jena appears to be not fulfilling any level of requirements and therefore could not be a potential choice from any practical implementation. If we study the best query performance for each query in table 5-9, and 5-10, (colored background cells), RDF4J and GraphDB columns collectively dominate the tables.

7. CONCLUSION

In this thesis geospatial capabilities of five RDF stores have been evaluated with a special focus on GeoSPARQL compliance as well as utilization for metadata management at ICOS-CP. It is concluded that all the five RDF stores offer varying levels of geospatial support features. RDF4J and GraphDB offer strongest compliance to GeoSPARQL and they also appear to be most suitable platforms for ICOS-CP requirements in our evaluation. Jena has the weakest GeoSPARQL compliance and is also not suitable for ICOS-CP while Virtuoso and Stardog have partial compliance to GeoSPARQL as well as partial suitability for ICOS-CP requirements.

In terms of query performances, the quantitative indicators also favor the GraphDB and RDF4J. All RDF stores offer spatial indexing and Lucene Spatial appears to be the most popular indexing technique in the evaluated platforms. Considerable optimization is observed in platforms where indexed performance was comparable versus un-indexed spatial query. With regards to state of the art on the progress relating to GeoSPARQL compliance in RDF stores, the progress appears to be still on the lower side. The dissimilarity in the result sets returned by similar operations in different RDF stores has also been highlighted which could be thoroughly investigated in future studies.

The testing code developed during this research (Java programs) can be executed on different platforms and have been made available online. The performance metrics appear to change considerably when tested on larger data sets as well as on better computing resources. Both the dataset size as well as host machines resources available for this study were modest, and therefore it is recommended that these tests may be conducted on machines with better computing resources as well as on larger geospatial datasets to evaluate more realistic suitability for ICOS-CP requirements.

Appendix A – Java Source Code

1. The complete source code developed during this study along with listing of java libraries (jars) used in the java projects is available in LUP as zip file at:
https://lup.lub.lu.se/luur/download?func=downloadFile&recordOId=8974835&fileOId=8974841.

2. The source code along with library files (jars) used in the study also available at:
https://github.com/Raza-Amir-Syed/TestGeoRDFStores.

References

Athanasiou, S., L. Bezati, G. Giannopoulos, K. Patroumpas, and D. Skoutaset. 2013. Market and Research Overview: Report on Geospatial RDF Stores – Where do we stand. *EU Geknow Project*. Accessed 01 January 2019, <http://svn.aksw.org/projects/GeoKnow/Public/D2.1.1_Market_and_Research_Overview.pdf>

Baode, J., and W. Dong-Qi. 2016. An Intersection Model of RCC-5 for Spatial Relationships and its Application – Science Alert Responsive Version. In *Journal of Software Engineering Volume 11 (1): 102-108.* DOI: 10.3923/jse.2017.102.108

Battle, R., and D. Kolas. 2012. Enabling the geospatial Semantic Web with Parliament and GeoSPARQL. In *Semantic Web Journal 3(4):355-370*. DOI: 10.3233/SW-2012-0065. University of California: IOS press.

BBC – Ontologies. n.d. Accessed 20 March 2019, <https://www.bbc.co.uk/ontologies>

Berners-Lee, T., J. Hendler, and O. Lassila. 2001. The Semantic Web. In *Scientific American, 284(5):34-43.*

Berners-Lee, T. 2006. Linked Data - Design Issues. Retrieved 01 January 2019, from https://www.w3.org/DesignIssues/LinkedData.html

Bizer, C., and A. Schultz. 2008. Benchmarking the Performance of Storage Systems that expose SPARQL Endpoint. In *4th International Workshop on Scalable Semantic Web knowledge Base Systems.*

Bizer, C., T. Heath, T. Berners-Lee. 2009 Linked Data - The story so far. In *International Journal on Semantic Web and Information Systems*, 5 (3), 1-22. DOI:10.4018/jswis.2009081901.

Brink , L. V. D., P. Janssen , W. Quak, and J. Stoter. 2014. Linking spatial data: semi-automated conversion of geoinformation models and GML data to RDF∗. In *International Journal of Spatial Data Infrastructures Research 9(2014), pp.59-85.*

Boncz, P., O. Erling, and M. Pham. (2014). Advances in Large-Scale RDF Data Management. In *Linked Open Data – Creating Knowledge Out of Interlinked Data, Results of the LOD2 Project*. S. Auer, V. Bryl, S. Tramp (Eds.). pp 21-44. LNCS 8861. Springer Book. DOI: 10.1007/978-3-319-09846-3.

Cheng, L., S. Kotoulas, T. Ward, and G. Theodoropoulos. 2012. Runtime Characterisation of Triple Stores: An Initial Investigation. In *15th IEEE International Conference on Computational Science and Engineering*. University of Ireland 2012. DOI: 10.1109/ICCSE.2012.19.

Cheatham, M., A. Krisnadhi, R. Amini, P. Hitzler, K. Janowicz, A. Shepherd, T. Narock, M. Jones, et al. 2018. The GeoLink knowledge graph. In *Big Earth Data, 2:2, 131-143*. DOI: 10.1080/20964471.2018.1469291

DBPedia – Wikki. n.d. Accessed 20 March 2019, <https://wiki.dbpedia.org/>.

DBPedia – About. n.d. Accessed 20 March 2019, <https://wiki.dbpedia.org/about>.

Eclipse RDF4J – Downloads. n.d. Accessed 20 March 2019, <http://rdf4j.org/download/>

Egenhofer, M. J., J. Sharma, and D. M. Mark. 1993. A critical comparison of the 4-intersection and 9-intersection models for spatial relations: Formal Analysis. In *R. McMaster and M. Armstrong (Eds.) Autocarta 11, Minneapolis, MN, pp.1-11*.

Enterprise Scale Knowledge Graph-ISWC. 2018. In *International Semantic Web Conference, October 2018*. Monterey Canada. Acessed 01 Januray 2019, <http://iswc2018.semanticweb.org/wp-content/uploads/2018/10/Panel-all.pdf/>

ESE. n.d. *Journal of Empirical Software Engineering*. 1996-2018. Accessed 01 Januray 2019, <https://www.springer.com/computer/swe/journal/10664>

Franz Inc. 2201 Broadway, Suite 715 Oakland, California 94612. Accessed 20 March 2019 <http://www.franz.com/>

Fernandez, D. M., and J. Passoth. 2018. Empirical Software Engineering: From Discipline to Interdiscipline. *Cornell University*. DOI: 10.1016/j.jss.2018.11.019

Fruchter, N., M. Specter, and B. Yuan. 2018. *Facebook/Cambridge Analytica: Privacy lessons and a way forward*. Internet Policy Research Initiative Massachusetts Institute of Technology. Accessed 13 February 2019, <https://internetpolicy.mit.edu/blog-2018-fb-cambridgeanalytica/>

*Garbis, G., K. Kyzirakos, and M. Koubarakis. 2013. Geographica: A Benchmark for Geospatial RDF Stores *. In Alani H. et al. (eds) The Semantic Web – ISWC 2013.* Lecture Notes in Computer Science, vol 8219. Springer, Berlin, Heidelberg. *DOI: 10.1007/978-3-642-41338-4_22*

Garcia, F., M. Serrano, J. A. Cruz-Lemus, M. Generao, C. Calero, and M. Piattini, 2007. Empirical Studties in Software Engineering Courses: Some Pedagogical Experiences*. In *International Journal of Engineering Education 24(4).*

GeoNames – Database. n.d. Accessed 20 March 2019, <http://www.geonames.org>

GeoNames – Ontology. n.d. Accessed 20 March 2019, <http://www.geonames.org/ontology/>

Goodwin, J., C. Dolbear, and G. Hart. 2009. Geographical linked data: The administrative geography of Great Britain on the semantic web. In *Transactions in GIS, 12, pp.19-30.* DOI: 10.1111/j.1467-9671.2008.01133.x

Gore, A. 1999. The Digital Earth: Understanding Our Planet in the 21st Century. In *Photogrammetric Engineering and Remote Sensing 65 (5): 528*. In *Australian Surveyor 43(2)* . DOI: 10.1080/00050348.1998.10558728.

Guo, Y., Z. X. Pan, and J. Heflin. 2005. LUBM: A benchmark for OWL knowledge base systems. In Journal of Web Semantics, vol. 3, pp. 158- 182. DOI: 10.1016/j.websem.2005.06.005

Heath, T., and C. Bizer. 2011. Linked Data: Evolving the Web into a Global Data Space. In Synthesis Lectures on the Semantic Web: Theory and Technology, 1st edn. Morgan and Claypool, San Rafael. DOI: 10.2200/S00334ED1V01Y201102WBE001.

Hietanen, E., L. Lehto, and P. Latvala. 2016. Providing Geographic Datasets as Linked Data in SDI. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLI-B2, 2016 XXIII ISPRS Congress, 12–19 July 2016, Prague, Czech Republic*

Huang, W., A. Mansourian, E. Abdolmajidi, H. Xu, and L. Harrie. 2018. Synchronising geometric representations for map mashups using relative positioning and Linked Data. In *International Journal of Geographical Information Science, 32:6, 1117-1137.* DOI: 10.1080/13658816.2018.1441416

ICOS – About. n.d. Accessed 20 March 2019, <https://www.icos-ri.eu/about-us>.

ICOS – Mission. n.d. Accessed 20 March 2019, <https://www.icos-ri.eu/our-mission>.

ICOS-CP – Introduction. n.d. Accessed 20 March 2019, https://www.icos-cp.eu/.

ICOS-CP – About. n.d. Accessed 20 March 2019, <https://www.icos-cp.eu/about-carbon-portal>

ICOS-CP – FAIR. n.d. Accessed 20 March 2019, <https://www.icos-cp.eu/fair_use>

ICOS-CP – SPARQL Endpoint. n.d. Accessed 20 March 2019, <https://meta.icos-cp.eu/sparqlclient/>

ICOS-CP – Ontology. n.d. Accessed 20 March 2019, <https://github.com/ICOS-Carbon-Portal/meta/tree/master/src/main/resources/owl>

Jena – Spatial Query. n.d. Accessed 20 March 2019, <https://jena.apache.org/documentation/query/spatial-query.html>

Jones, J., W. Kuhn, C. Keßler, and S. Scheider. Making the web of data available via web feature services. 2014. In *Connecting a Digital Europe through Location and Place. pp. 341–361*. Springer International Publishing. Accessed 15 March 2019, <http://carsten.io/jones-kuhn-kessler-scheider-agile2014.pdf>.

Lehmann, J., R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. P. Morsey, et al. 2015. DBpedia A large-scale, multilingual knowledge base extracted from wikipedia. In *Semantic Web 1 (2012) 1–5 1 IOS Press*. DOI: 10.3233/SW-140134.

Lenka, R. K., R. K. Barik, N. Gupta, and S. M. Ali. 2016. Comparative analysis of Spatial Hadoop and GeoSpark for Geospatial Big Data Analytics. 2016. *In 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, 2016, pp. 484-488*. DOI: 10.1109/IC3I.2016.7918013.

LOD Cloud. n.d. Accessed 20 March 2019 <https://lod-cloud.net/>

Liu, B., and B. Hu. 2005. An Evaluation of RDF Storage Systems for Large Data Applications. In *First International Conference on Semantics, Knowledge and Grid*, Beijing, 2005, pp. 59-59. DOI: 10.1109/SKG.2005.37.

Mansourian, A., E. Omidi, A. Toomanian, L. Harrie. 2010. Expert system to enhance the functionality of clearinghouse services. In *ELSEVIER. Computers, Environment and Urban Systems Volume 35, Issue 2, March 2011, Pages 159-172.* DOI: 10.1016/j.compenvurbsys.2010.06.003

Morsey, M., J. Lehmann, S. Auer, and N. N. Axel-Cyrile. 2011. DBpedia SPARQL Benchmark – Performance Assessment with Real Queries on Real Data. In *International Semantic Web conference ISWC-2011* pp 454-469. DOI: 10.1007/978-3-642-25073-6_29.

Nalepa, G. J., and W. T. Furmanska. 2009. Review of semantic web technologies for GIS. Department of Automatics, AGH University of Science and Technology, Krakow, Poland. EU ICT Project INDECT.

OGC. 2003. Abstract Specifications. *Open Geospatial Consortium.* Accessed 01 January 2019, <https://www.opengeospatial.org/docs/as>

OGC. 2005. Web Feature Service (WFS) Implementation Specification. *Ref No OGC 04-094*. Accessed 01 January 2019, <http://portal.opengeospatial.org/files/?artifact_id=8339>

OGC. 2006. Web Map Service (WMS) Implementation Specification. *Ref No OGC 06-042*. Accessed 01 January 2019, <http://portal.opengeospatial.org/files/?artifact_id=14416>

OGC. 2012. GeoSPARQL - A Geographic Query Language for RDF Data. Accessed 01 January 2019, <http://www.opengis.net/doc/IS/geosparql/1.0>

OTC – Strategy. n.d. Accessed 01 January 2019, <https://otc.icos-cp.eu/strategy-and-coverage>

Rattanasawad, T., M. Buranarach, K. R. Saikaew, and T. Sunpnithi. 2018. A Comparative Study of Rule-Based Inference Engines for the Semantic Web. In *IIEICE Transactions on Information and Systems 2018 Volume E101.D Issue 1 Pages 82-89*. DOI: 10.1587/transinf.2017SWP0004.

Rohloff, K., M. Dean, I. Emmons, D. Ryder, and J. Sumner. 2007. An Evaluation of Triple-Store Technologies for Large Data Stores. In *Meersman R., Tari Z., Herrero P. (eds) On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*. Lecture Notes in Computer Science, vol 4806. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-76890-6_38.

Sakr, S., M. Wylot, R. Mutharaju, D. L. Phuoc, and I. Fundulaki. 2018a. Introduction to Linked Data. In *Linked Data Storing, Querying and Reasoning. Springer Book. Springer Internation Publishing 2018 pp 1-8.* DOI: 10.1007/978-3-319-73515-3

Sakr, S., M. Wylot, R. Mutharaju, D. L. Phuoc, and I. Fundulaki. 2018b. Distributed Reasoning of RDF Data. In *Linked Data Storing, Querying and Reasoning. Springer Book. Springer Internation Publishing 2018 pp 109.* DOI: 10.1007/978-3-319-73515-3

Samet, H. 2015. Sorting Spatial Data. In *International Encyclopedia of Geography, D. Richardson, ed., Wiley and Sons, Oxford, UK, 2016*.

Schmachtenberg, M., C. Bizer, and H. Paulheim. 2014. Adoption of the linked data best practices in different topical domains. In *The Semantic Web. In: Mika P. et al. (eds) The Semantic Web – ISWC 2014*. Lecture Notes in Computer Science, vol 8796. Springer, Cham. DOI: 10.1007/978-3-319-11964-9_16

Simon, G. 2018. An Introduction to Geo Indexes and their performance characteristics: Part I. Accessed 25 December 2018, <https://www.arangodb.com/2018/01/introduction-geo-indexes-performance-characteristics-part-1/>

SOCAT – Info. n.d. Accessed 01 January 2019, <www.socat.info>

Stadler, C., J. Lehmann, K. Höffner, and Sören Auer. 2012. LinkedGeoData: A Core for a Web of Spatial Open Data. In *Semantic Web 3 (4), 333-354*. DOI: 10.3233/SW-2011-0052

Tschirner, S., A. Scherp, and S. Staab. 2011. Semantic access to INSPIRE How to publish and query advanced GML data. In *Proceedings of the Terra Cognita Workshop on Foundations, Technologies and Applications of the Geospatial Web 798. pp. 75–87.*

Vilches-Blázquez, L. M., B. Villazón-Terrazas, O. Corcho, and A. Gómez-Pérez, A. 2014. Integrating geographical information in the Linked Digital Earth. In *International Journal of Digital Earth, 7(7), pp.554-575*. DOI: 10.1080/17538947.2013.783127

Virtuoso – Sesame Provider. n.d. Accessed 01 January 2019, <http://docs.openlinksw.com/virtuoso/rdfnativestorageproviderssesame/>

W3C. 2001. Semantic Web Standards. *World Wide Web Consortium Recommendation* Accessed 01 January 2019, <https://www.w3.org/2001/sw/wiki/Main_Page>

W3C. 2008. SPARQL Query Language for RDF. *World Wide Web Consortium Recommendation*. Accessed 01 January 2019, <https://www.w3.org/TR/rdf-sparql-query/>

W3C. 2011. Web Ontology Language. *World Wide Web Consortium Recommendation.* Accessed 01 January 2019, <https://www.w3.org/OWL/>

W3C. 2013a. RIF Overview (2nd Edition). *World Wide Web Consortium Recommendation*. Accessed 01 January 2019, <https://www.w3.org/standards/techs/rif>

W3C. 2013b. SPARQL 1.1 Query Language for RDF. *World Wide Web Consortium Recommendation*. Accessed 01 January 2019, <https://www.w3.org/TR/sparql11-query/>

W3C. 2014a. RDF 1.1 Concepts and Abstract Syntax. *World Wide Web Consortium Recommendation.* Accessed 01 January 2019, <https://www.w3.org/TR/rdf11-concepts/>

W3C. 2014b. RDFS Schema 1.1. *World Wide Web Consortium Recommendation*. Accessed 01 January 2019, <https://www.w3.org/TR/rdf-schema/>

W3C. 2018. Large Triple Stores. *World Wide Web Consortium*. Accessed 01 January 2019, <https://www.w3.org/wiki/LargeTripleStores>

Wilkinson, M. D., M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J. Boiten, et al. 2016. The FAIR Guiding Principles for scientific data management and stewardship. In *Scientific Data* ***volume3****, Article number: 160018 (2016).* DOI: 10.1038/sdata.2016.18.

Zhang, C., T. Zhao, and W. Li. 2015. Geospatial Semantic Web. Springer Books. Springer Cham Heidelberg New York. DOI 10.1007/978-3-319-17801-1

Zhang, L., T. Jia-Hao, J. Jiang, Y Liu, M. Pu, and T. Yue. 2018. Empirical Research in Software Engineering - A Literature Survey. In Journal of Computer Science and Technology Volume 33, Issue 5, pp 876–899. DOI: 10.1007/s11390-018-1864-x.

Series from Lund University

Department of Physical Geography and Ecosystem Science

Master Thesis in Geographical Information Science

1. *Anthony Lawther:* The application of GIS-based binary logistic regression for slope failure susceptibility mapping in the Western Grampian Mountains, Scotland (2008).
2. *Rickard Hansen:* Daily mobility in Grenoble Metropolitan Region, France. Applied GIS methods in time geographical research (2008).
3. *Emil Bayramov:* Environmental monitoring of bio-restoration activities using GIS and Remote Sensing (2009).
4. *Rafael Villarreal Pacheco:* Applications of Geographic Information Systems as an analytical and visualization tool for mass real estate valuation: a case study of Fontibon District, Bogota, Columbia (2009).
5. *Siri Oestreich Waage:* a case study of route solving for oversized transport: The use of GIS functionalities in transport of transformers, as part of maintaining a reliable power infrastructure (2010).
6. *Edgar Pimiento:* Shallow landslide susceptibility – Modelling and validation (2010).
7. *Martina Schäfer:* Near real-time mapping of floodwater mosquito breeding sites using aerial photographs (2010).
8. *August Pieter van Waarden-Nagel:* Land use evaluation to assess the outcome of the programme of rehabilitation measures for the river Rhine in the Netherlands (2010).
9. *Samira Muhammad:* Development and implementation of air quality data mart for Ontario, Canada: A case study of air quality in Ontario using OLAP tool. (2010).
10. *Fredros Oketch Okumu*: Using remotely sensed data to explore spatial and temporal relationships between photosynthetic productivity of vegetation and malaria transmission intensities in selected parts of Africa (2011).
11. *Svajunas Plunge:* Advanced decision support methods for solving diffuse water pollution problems (2011).
12. *Jonathan Higgins:* Monitoring urban growth in greater Lagos: A case study using GIS to monitor the urban growth of Lagos 1990 - 2008 and produce future growth prospects for the city (2011).
13. *Mårten Karlberg:* Mobile Map Client API: Design and Implementation for Android (2011).
14. *Jeanette McBride:* Mapping Chicago area urban tree canopy using color infrared imagery (2011).
15. *Andrew Farina:* Exploring the relationship between land surface temperature and vegetation abundance for urban heat island mitigation in Seville, Spain (2011).
16. *David Kanyari*: Nairobi City Journey Planner: An online and a Mobile Application (2011).

17. *Laura V. Drews:* Multi-criteria GIS analysis for siting of small wind power plants - A case study from Berlin (2012).
18. *Qaisar Nadeem:* Best living neighborhood in the city - A GIS based multi criteria evaluation of ArRiyadh City (2012).
19. *Ahmed Mohamed El Saeid Mustafa:* Development of a photo voltaic building rooftop integration analysis tool for GIS for Dokki District, Cairo, Egypt (2012).
20. *Daniel Patrick Taylor*: Eastern Oyster Aquaculture: Estuarine Remediation via Site Suitability and Spatially Explicit Carrying Capacity Modeling in Virginia's Chesapeake Bay (2013).
21. *Angeleta Oveta Wilson:* A Participatory GIS approach to *unearthing* Manchester's Cultural Heritage '*gold mine'* (2013).
22. *Ola Svensson:* Visibility and Tholos Tombs in the Messenian Landscape: A Comparative Case Study of the Pylian Hinterlands and the Soulima Valley (2013).
23. *Monika Ogden:* Land use impact on water quality in two river systems in South Africa (2013).
24. *Stefan Rova:* A GIS based approach assessing phosphorus load impact on Lake Flaten in Salem, Sweden (2013).
25. *Yann Buhot:* Analysis of the history of landscape changes over a period of 200 years. How can we predict past landscape pattern scenario and the impact on habitat diversity? (2013).
26. *Christina Fotiou:* Evaluating habitat suitability and spectral heterogeneity models to predict weed species presence (2014).
27. *Inese Linuza:* Accuracy Assessment in Glacier Change Analysis (2014).
28. *Agnieszka Griffin:* Domestic energy consumption and social living standards: a GIS analysis within the Greater London Authority area (2014).
29. *Brynja Guðmundsdóttir:* Detection of potential arable land with remote sensing and GIS - A Case Study for Kjósarhreppur (2014).
30. *Oleksandr Nekrasov:* Processing of MODIS Vegetation Indices for analysis of agricultural droughts in the southern Ukraine between the years 2000-2012 (2014).
31. *Sarah Tressel:* Recommendations for a polar Earth science portal in the context of Arctic Spatial Data Infrastructure (2014).
32. *Caroline Gevaert:* Combining Hyperspectral UAV and Multispectral Formosat-2 Imagery for Precision Agriculture Applications (2014).
33. *Salem Jamal-Uddeen:* Using GeoTools to implement the multi-criteria evaluation analysis - weighted linear combination model (2014).
34. *Samanah Seyedi-Shandiz:* Schematic representation of geographical railway network at the Swedish Transport Administration (2014).
35. *Kazi Masel Ullah:* Urban Land-use planning using Geographical Information System and analytical hierarchy process: case study Dhaka City (2014).
36. *Alexia Chang-Wailing Spitteler:* Development of a web application based on MCDA and GIS for the decision support of river and floodplain rehabilitation projects (2014).

37. *Alessandro De Martino:* Geographic accessibility analysis and evaluation of potential changes to the public transportation system in the City of Milan (2014).
38. *Alireza Mollasalehi:* GIS Based Modelling for Fuel Reduction Using Controlled Burn in Australia. Case Study: Logan City, QLD (2015).
39. *Negin A. Sanati:* Chronic Kidney Disease Mortality in Costa Rica; Geographical Distribution, Spatial Analysis and Non-traditional Risk Factors (2015).
40. *Karen McIntyre:* Benthic mapping of the Bluefields Bay fish sanctuary, Jamaica (2015).
41. *Kees van Duijvendijk:* Feasibility of a low-cost weather sensor network for agricultural purposes: A preliminary assessment (2015).
42. *Sebastian Andersson Hylander:* Evaluation of cultural ecosystem services using GIS (2015).
43. *Deborah Bowyer:* Measuring Urban Growth, Urban Form and Accessibility as Indicators of Urban Sprawl in Hamilton, New Zealand (2015).
44. *Stefan Arvidsson:* Relationship between tree species composition and phenology extracted from satellite data in Swedish forests (2015).
45. *Damián Giménez Cruz*: GIS-based optimal localisation of beekeeping in rural Kenya (2016).
46. *Alejandra Narváez Vallejo:* Can the introduction of the topographic indices in LPJ-GUESS improve the spatial representation of environmental variables? (2016).
47. *Anna Lundgren:* Development of a method for mapping the highest coastline in Sweden using breaklines extracted from high resolution digital elevation models (2016).
48. *Oluwatomi Esther Adejoro:* Does location also matter? A spatial analysis of social achievements of young South Australians (2016).
49. *Hristo Dobrev Tomov:* Automated temporal NDVI analysis over the Middle East for the period 1982 - 2010 (2016).
50. *Vincent Muller:* Impact of Security Context on Mobile Clinic Activities A GIS Multi Criteria Evaluation based on an MSF Humanitarian Mission in Cameroon (2016).
51. *Gezahagn Negash Seboka:* Spatial Assessment of NDVI as an Indicator of Desertification in Ethiopia using Remote Sensing and GIS (2016).
52. *Holly Buhler:* Evaluation of Interfacility Medical Transport Journey Times in Southeastern British Columbia. (2016).
53. *Lars Ole Grottenberg*: Assessing the ability to share spatial data between emergency management organisations in the High North (2016).
54. *Sean Grant:* The Right Tree in the Right Place: Using GIS to Maximize the Net Benefits from Urban Forests (2016).
55. *Irshad Jamal:* Multi-Criteria GIS Analysis for School Site Selection in Gorno-Badakhshan Autonomous Oblast, Tajikistan (2016).
56. *Fulgencio Sanmartín:* Wisdom-volkano: A novel tool based on open GIS and time-series visualization to analyse and share volcanic data (2016).

57. *Nezha Acil:* Remote sensing-based monitoring of snow cover dynamics and its influence on vegetation growth in the Middle Atlas Mountains (2016).
58. *Julia Hjalmarsson:* A Weighty Issue: Estimation of Fire Size with Geographically Weighted Logistic Regression (2016).
59. *Mathewos Tamiru Amato:* Using multi-criteria evaluation and GIS for chronic food and nutrition insecurity indicators analysis in Ethiopia (2016).
60. *Karim Alaa El Din Mohamed Soliman El Attar:* Bicycling Suitability in Downtown, Cairo, Egypt (2016).
61. *Gilbert Akol Echelai:* Asset Management: Integrating GIS as a Decision Support Tool in Meter Management in National Water and Sewerage Corporation (2016).
62. *Terje Slinning:* Analytic comparison of multibeam echo soundings (2016).
63. *Gréta Hlín Sveinsdóttir:* GIS-based MCDA for decision support: A framework for wind farm siting in Iceland (2017).
64. *Jonas Sjögren:* Consequences of a flood in Kristianstad, Sweden: A GIS-based analysis of impacts on important societal functions (2017).
65. *Nadine Raska:* 3D geologic subsurface modelling within the Mackenzie Plain, Northwest Territories, Canada (2017).
66. *Panagiotis Symeonidis*: Study of spatial and temporal variation of atmospheric optical parameters and their relation with PM 2.5 concentration over Europe using GIS technologies (2017).
67. *Michaela Bobeck:* A GIS-based Multi-Criteria Decision Analysis of Wind Farm Site Suitability in New South Wales, Australia, from a Sustainable Development Perspective (2017).
68. *Raghdaa Eissa*: Developing a GIS Model for the Assessment of Outdoor Recreational Facilities in New Cities Case Study: Tenth of Ramadan City, Egypt (2017).
69. *Zahra Khais Shahid*: Biofuel plantations and isoprene emissions in Svea and Götaland (2017).
70. *Mirza Amir Liaquat Baig*: Using geographical information systems in epidemiology: Mapping and analyzing occurrence of diarrhea in urban - residential area of Islamabad, Pakistan (2017).
71. *Joakim Jörwall*: Quantitative model of Present and Future well-being in the EU-28: A spatial Multi-Criteria Evaluation of socioeconomic and climatic comfort factors (2017).
72. *Elin Haettner*: Energy Poverty in the Dublin Region: Modelling Geographies of Risk (2017).
73. *Harry Eriksson*: Geochemistry of stream plants and its statistical relations to soil- and bedrock geology, slope directions and till geochemistry. A GIS-analysis of small catchments in northern Sweden (2017).
74. *Daniel Gardevärn:* PPGIS and Public meetings – An evaluation of public participation methods for urban planning (2017).
75. *Kim Friberg:* Sensitivity Analysis and Calibration of Multi Energy Balance Land Surface Model Parameters (2017).
76. *Viktor Svanerud:* Taking the bus to the park? A study of accessibility to green areas in Gothenburg through different modes of transport (2017).

77. *Lisa-Gaye Greene*: Deadly Designs: The Impact of Road Design on Road Crash Patterns along Jamaica's North Coast Highway (2017).
78. *Katarina Jemec Parker*: Spatial and temporal analysis of fecal indicator bacteria concentrations in beach water in San Diego, California (2017).
79. *Angela Kabiru*: An Exploratory Study of Middle Stone Age and Later Stone Age Site Locations in Kenya's Central Rift Valley Using Landscape Analysis: A GIS Approach (2017).
80. *Kristean Björkmann*: Subjective Well-Being and Environment: A GIS-Based Analysis (2018).
81. *Williams Erhunmonmen Ojo*: Measuring spatial accessibility to healthcare for people living with HIV-AIDS in southern Nigeria (2018).
82. *Daniel Assefa*: Developing Data Extraction and Dynamic Data Visualization (Styling) Modules for Web GIS Risk Assessment System (WGRAS). (2018).
83. *Adela Nistora*: Inundation scenarios in a changing climate: assessing potential impacts of sea-level rise on the coast of South-East England (2018).
84. *Marc Seliger*: Thirsty landscapes - Investigating growing irrigation water consumption and potential conservation measures within Utah's largest master-planned community: Daybreak (2018).
85. *Luka Jovičić*: Spatial Data Harmonisation in Regional Context in Accordance with INSPIRE Implementing Rules (2018).
86. *Christina Kourdounouli*: Analysis of Urban Ecosystem Condition Indicators for the Large Urban Zones and City Cores in EU (2018).
87. *Jeremy Azzopardi*: Effect of distance measures and feature representations on distance-based accessibility measures (2018).
88. *Patrick Kabatha*: An open source web GIS tool for analysis and visualization of elephant GPS telemetry data, alongside environmental and anthropogenic variables (2018).
89. *Richard Alphonce Giliba*: Effects of Climate Change on Potential Geographical Distribution of Prunus africana (African cherry) in the Eastern Arc Mountain Forests of Tanzania (2018).
90. *Eiður Kristinn Eiðsson*: Transformation and linking of authoritative multi-scale geodata for the Semantic Web: A case study of Swedish national building data sets (2018).
91. *Niamh Harty*: HOP!: a PGIS and citizen science approach to monitoring the condition of upland paths (2018).
92. *José Estuardo Jara Alvear*: Solar photovoltaic potential to complement hydropower in Ecuador: A GIS-based framework of analysis (2018).
93. *Brendan O'Neill*: Multicriteria Site Suitability for Algal Biofuel Production Facilities (2018).
94. *Roman Spataru*: Spatial-temporal GIS analysis in public health – a case study of polio disease (2018).
95. *Alicja Miodońska*: Assessing evolution of ice caps in Suðurland, Iceland, in years 1986 - 2014, using multispectral satellite imagery (2019).
96. *Dennis Lindell Schettini*: A Spatial Analysis of Homicide Crime's Distribution and Association with Deprivation in Stockholm Between 2010-2017 (2019).

97. *Damiano Vesentini*: The Po Delta Biosphere Reserve: Management challenges and priorities deriving from anthropogenic pressure and sea level rise (2019).
98. *Emilie Arnesten*: Impacts of future sea level rise and high water on roads, railways and environmental objects: a GIS analysis of the potential effects of increasing sea levels and highest projected high water in Scania, Sweden (2019).
99. *Syed Muhammad Amir Raza*: Comparison of geospatial support in RDF stores: Evaluation for ICOS Carbon Portal metadata (2019).