

Challenges Over Two Semantic Repositories - OWLIM and Allegro Graph

Paria Tajabor¹, Tara Raafat²

¹Department of Chemical and Process Engineering, Surrey University, Guildford, United Kingdom

²Mphasis, University of Surrey, Avco Systems Ltd University of Surrey, Guildford, United Kingdom

*Corresponding author, e-mail: p.tajabor@gmail.com

Abstract

The purpose of this research study is exploring two kind of semantic repositories with regards to various factors to find the best approaches that an artificial manager can use to produce ontology in a system based on their interaction, association and research. To this end, as the best way to evaluate each system and comparing with others is analysis, several benchmarking over these two repositories were examined. These two semantic repositories: OWLIM and AllegroGraph will be the main core of this study. The general objective of this study is to be able to create an efficient and cost-effective manner reports which is required to support decision making in any large enterprise.

Keywords: OWLIM, AllegroGraph, RDF, reasoning, semantic repository, semantic-Web, SPARQL, Ontology, Query

Copyright © 2016 Institute of Advanced Engineering and Science. All rights reserved.

1. Introduction

It is necessary to point out that in huge projects, government processes; companies, IT systems and so forth there are different information and databases. These have various types of formats, sets of forms, documents and repositories, with not very good management and coordination. It is obvious that rampant redundancy will occur when the procedure of developing or redeveloping in the system, using the same information but with different formats, architectures and models, is being processed. Basically, views and attitudes towards a wide range of information as well as integration of accurate and complete understanding is difficult and in some cases almost impossible and unachievable. On the other hand developing this information is really expensive and time consuming; hence the problem is exacerbated more when this process would take place in lack of interoperability, inconsistent designs and redundant systems. For this reason, different kinds of repositories are provided to replace this loss of information and data fragmentation with an architected approach to integrating, developing and managing information throughout the system.

This work defines two different semantic repositories and comparing them in terms of different parameters and factors. To this aim, beside the backgrounds and exploring various theories of researches, analysis through different aspect of assessment is done. In fact the research objectives can be summarized in the fully assessment of a selected semantic repositories.

2. Background and Motivation

Over the last decade, as semantic web has rapidly developed, the effort of system developers has also significantly increased, especially in fields where nowadays the importance of semantic repositories is equal to HTTP servers. This has led to successful numbers of ontology standards and robust metadata and the role of these standards is akin to the role of SQL to developing or spreading DBMS relations [1].

The characteristics of a semantic repository are generally similar to the data base management systems (DBMS) [2]. They handle data functions, storing information, querying and managing organized data. In fact, as Dimitrova in his 2010 conference pointed out, a semantic repository combines features of inference engines and DBMS. However, the major

difference between semantic repository and DBMS is ontology, which is just used in semantic repositories as semantic schema [3].

By developing a system many connections among languages may be lost, which could lead to error prone, lack of integration, complexity and finally system failures. This issue would be catastrophic when there are combinations of different systems, multiple contractors and vendors with several architectural methods. "The fragmentation of information is profound" [4].

As previously mentioned, this project has been carried out to find two different semantic repositories. As Kiryakov argued, OWLIM as an inference layer and storage for Sesame with reliable persistence strategy and high performance reasoning has the power to combine OWL DLP, RDFS and OWL Horst using TRREE engine [5]. The following section will outline OWLIM in more detail afterward next parts which will be about the other repository that is called allograph.

2.1. OWLIM: A Pragmatic Semantic Repository for OWL

OWLIM with the performance of OWL DLP reasoning is based on forward chaining of entitlement rules. This means that the goal is reachable among starting from available facts or data using inference rules for extracting more data until that goal can be achieved [6]. The significance of OWLIM is its scalability over millions of statements. In addition it can process a knowledge base of more than 10 million explicit statements while, by using forward chaining reasoning, it has the potential to extend handling of statements to around 19 million. The important point is that according to the size of the semantic repository, the speed of uploading and storing is varied from 3000 to 18000 statements per second for a small to maximal size repository. On the other hand, as noted, OWLIM acts as an inference strategy, hence deletion processes are not cheap and it takes a few minutes. Another fact to be noted is that the amount of diverse queries will be assessed in milliseconds [7].

Although OWLIM comes from the term of OWL In-Memory, according to Kiryakov, Ognyanov and Manov "OWLIM is the short name of the OWLMemSchemaRepository SAIL (Storage and Inference Layer) for Sesame, which supports partial reasoning over OWL DLP". As discussed by named researchers all content of this repository is preserved and loaded from the main memory, hence it is a kind of in-memory reasoning which is capable of well-organized query answering and recovering, subsequently it has strong strategies for backup and persistency.

The key features of the current release of OWLIM can be summarized as follows:

- The most scalable semantic repository in the World, both in terms of the volume of RDF data it can store and the speed with which it can load and inference.
- Pure Java implementation, ensuring ease of deployment and portability.
- Compatible with Sesame 2, which brings interoperability benefits and support for all major RDF syntaxes and query languages [8].
- Special geometrical query constructions and SPARQL extensions functions [1].
- High performance retraction of statements and their inferences-so inference materialization speeds up retrieval, but without delete performance degradation.
- Powerful and expressive consistency/integrity constraint checking mechanisms.
- RDF Priming, based upon activation spreading, allows efficient data selection and context-aware query answering for handling huge datasets [9]

The limitations of OWLIM are related to its reasoning strategy. In general, the expressivity of the language supported cannot be extended in the Description Logic direction, because the semantics must be able to be captured in (Horn) rules. The total materialization strategy has drawbacks when changes to the explicitly asserted statements occur frequently. For expressive semantics and certain ontologies, the number of implicit statements can grow quickly with the expected degradation in performance [10].

2.2. Allegro Graph

Allegro Graph is a kind of database used as a framework for making semantic web applications. Data and meta-data can be saved in it in the triples form and this triples search (do query) is possible among different types of query APIs such as Prolog and SPARQL, along with the application of RDFS ++ reasoning with its built-in reasoner. Allegro Graph includes support for federation, social network analysis and gruff [11]. Inter alia, Allegro Graph is characterized by being modern and as a database with a powerful, stable graphic frame 20 and high

efficiency. Compared with a rational database, a graphic database can have any number of relationships for any saved sample. These relationships appear in the form of links which take the form of a network or graph in combination with each other. Combined with disk based storage, Allegro Graph efficiently makes use of the memory. This causes a better performance and maintenance, while at the same time; billions of quads can be measured. Among a very high number of client application programs, AllegroGraph supports SPARQL, RDFS ++ and prolog reasoning [12].

Expressive and powerful querying and reasoning is one of the positive points of this approach. In fact, Allegro Graph enables the most expansive set of arrays for query and access to information in the RDF datastore. Description logics or OWL-DL reasoners are more capable of managing complex ontologies. They try to be complete and successful in responding to all the queries, yet, on the other hand, when increasing the number of triples to millions; they act completely different and are unpredictable in terms of execution time. Franze (2010) believes that Allegro Graph provides a very high speed reasoner and practical RDFS++.

Some Features of Allegro Graph are pointed below:

- Social Networking Analysis [11]
- Native data types and Efficient range queries [13]
- Federation [11]
- Gruff [14]

Regarding the figure 1, the structure of Allegro graph can be better understood. According to the inference of Franz, it should be mentioned that “AllegroGraph provides a REST protocol architecture which is essentially a superset of the sesame HTTP client”.

Franz staff directly covers adapters for different languages like sesame java, sesame Jena, Python using the sesame signature and Lisp.

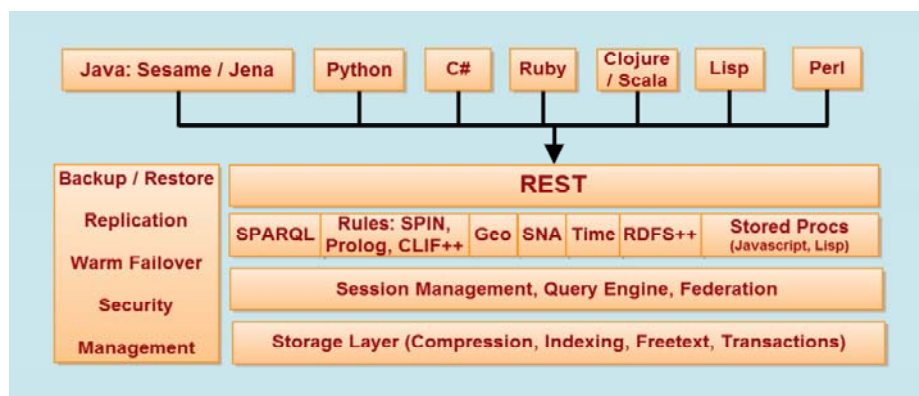


Figure 1. Architecture of Allegro graph [11]

3. Methodology

In this part the ways that has been used in this project to benchmark repositories and achieve some reasonable results, are outlined as follows:

3.1. Information Sourcing

First of all to beginning the analysis, I needed huge amount of ontologies, and the reason was to put these ontologies to the selected repositories for querying and accordingly benchmarking. To this end loads of ontologies were taken from the main web site of ontology which is (Watson.kmi.open). In addition, a class of ontology from my point of view has been made and imported it into one of ontologies, this effort was done to evaluate faster, easier and for more accuracy and completeness of queries.

3.2. Benchmarking and Analysis

The aim of this part is benchmarking the selected semantic repositories for comparison between their different components and finally a conclusion which will be based on the obtained

data. These repositories are considerably different in the amount of their levels and that might have influence on the final systems on deployment decision. For example, load and query response times, scalability, supported query languages, semantic expressivity or reasoning capability. Therefore, to achieve this goal some queries were provided to use for searching among triples and finding the relevant results. Queries are designed from simple till complex for better assessing in different situations and they were tested separately in each repository regarding to their different sizes from small to medium size ontologies. It will be testing two semantic repositories in terms of their dataset Load time, query results and query execution speed. By adding various ontologies in various sizes into selected repositories, a total of four different dataset sizes were created to test repositories in different situations which will be clarified more in the practical part of project.

4. Finding and Analysis

4.1. Benchmarking Factors

Some factors upon both repositories will be briefly presented in the Table 1.

Table 1. Features of repositories

| Factors | Allegrograph | BigOWLIM |
|-------------------------|----------------------------------|--------------------------------|
| Storage form | Native-based | Memory and native-based |
| Query Language Support | SPARQL, TWINQL, SeRQL and Prolog | SPARQL and SeRQL |
| Reasoner Integration | Built-in, Jena, Racerpro, Sesame | Built-in, Sesame |
| RDF Update | API | API |
| Reasoning tactic | Backward chaining | Forward chaining |
| Client part | PL/SQL, Java and C | java |
| Platform maintained | Unix, Windows, Solaris and Mac | Unix, Windows, Solaris and Mac |
| RDF view support | Not definitive | No |
| Format of Serialization | N-Triples, RDF-XML and N3 | N-Triples, RDF-XML and N3 |

4.2. Test Description

The next level of doing this benchmark is testing these two kind of semantic repositories in terms of their dataset Load time, query results and query execution speed. The baseline test is run on small and medium size ontologies. By adding various ontologies in various sizes into selected repositories, a total of four different dataset sizes were created to test repositories in different situations. In other words, this testing is performed with the following variation in ontologies-size conditions (Table 2).

Table 2. Ontology sizes

| conditions | 1 | 2 | 3 | 4 |
|-----------------------------|-------|-------|-------|--------|
| Ontologies or triples sizes | 10000 | 50000 | 70000 | 100000 |

Furthermore to better assess these repositories in different situations, 5 different queries were created from simple to complex, and then they were tested separately in each repository regarding to their different sizes (Table 3).

Table 3. Queries

| Queries | Q1 | Q2 | Q3 | Q4 | Q5 |
|---------|----|----|----|----|----|
|---------|----|----|----|----|----|

Hence given these 4 ontology sizes and 5 queries, a total of 20 test conditions have been performed on 2 different repositories which lead to 40 results that will be illustrated in following parts.

4.3. Dataset Load Timings

First of all loading a particular ontology from all four dataset sizes, based on two selected repositories has been done with the help of SPARQL statements which is „load transportation”, can easily calculate load time (Figure 2).

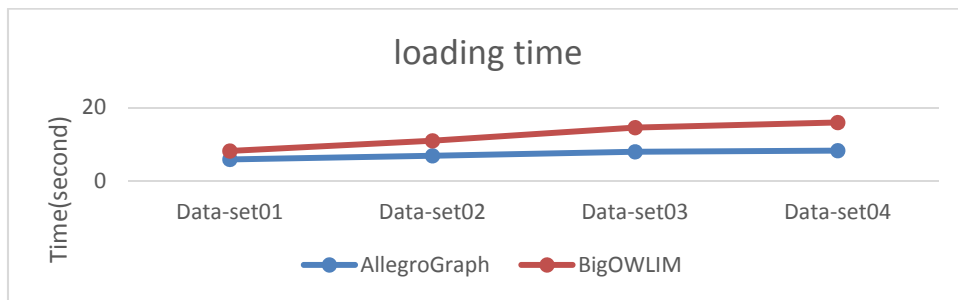


Figure 2. Transportation ontology loading time

It can be claimed that Allograph had the best performance in terms of loading time and they are consistent and scalable against increasing dataset size and BigOWLIM is the lower repository because it performs forward-chaining of facts and stores them explicitly [15].

4.4. Query Results and Execution Speed Analysis

By performing provided queries, operation of searching and finding correct data will be activated and returned back to the browser.

As mentioned before to evaluate better, I made a class from my point of view and imported it into the ontology of transportation, hence concept of all five queries are based on this class.

Query 1:

This is a relatively simple query which was designed to find a different kind of CityCar as well as those classes which are disjointed with it.

```
SELECT ?y ?x WHERE
{ ?x rdfs:subClassOf "CityCar" .
  ?y owl:disjointWith "CityCar" . }
```

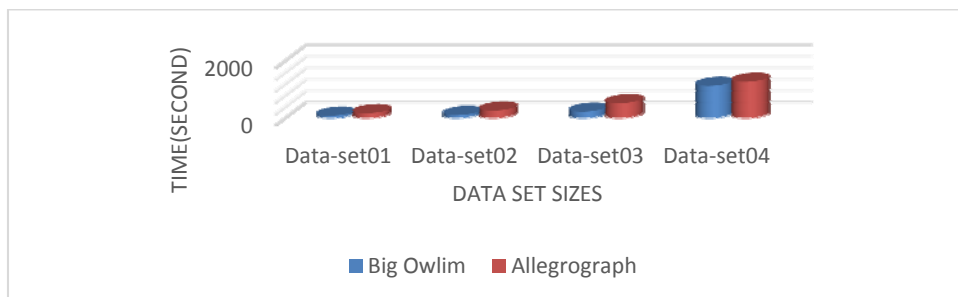


Figure 3. Query 1 outcomes

From the Figure 3 can be claimed that this query generally returns relatively quickly, regardless of ontology size. Both stores were competitive up to dataset 04, with sub-second response times.

Table 4. Results of Query 1 in detail

| Repository | Data-set size | Answer | Correct answer | Min time (msec) | Triple counting |
|--------------|---------------|--------|----------------|-----------------|-----------------|
| Allegrograph | 10000 | yes | yes | 160 | 137 |
| Big Owlilim | 10000 | yes | yes | 73 | 107 |
| Allegrograph | 50000 | yes | yes | 253 | 67 |
| Big Owlilim | 50000 | yes | yes | 97 | 85 |
| Allegrograph | 70000 | yes | yes | 512 | 124 |
| Big Owlilim | 70000 | yes | yes | 214 | 97 |
| Allegrograph | 100000 | yes | yes | 1257 | 74 |
| Big Owlilim | 100000 | yes | yes | 116 | 114 |

From the table 4 can be seen that all both stores yielded usable data for this query.

Query 2:

This query was designed to find any relevant information about specific customer, such as "paria".

```
SELECT ?x ?y ?z ?w ?p ?R ?N WHERE
{ "Paria" rdfs:subClassOf ?x
  ?x rdfs:subClassOf ?y
  ?y rdfs:subClassOf ?z
  ?z rdfs:subClassOf ?w
  ?R rdfs:subClassOf ?N
  ?p rdfs:subClassOf ?R
  ?w rdfs:subClassOf ?p . }
```

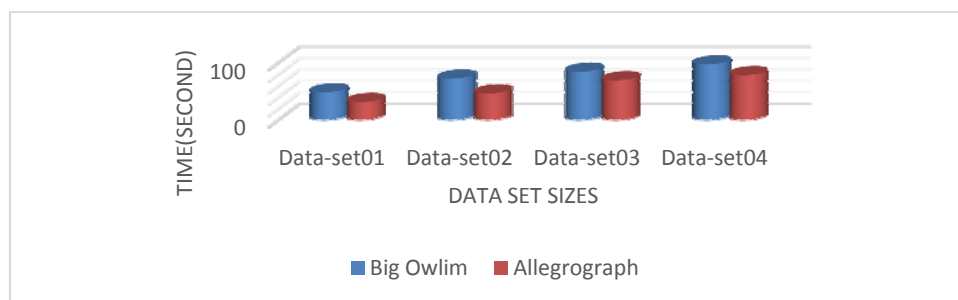


Figure 4. Query 2 outcomes

As it is shown in the Figure 4, this query generally returns relatively quickly, regardless of Ontology size. It seems that AllegroGraph did better on this query, with responses below 100 (m/sec) across the board. OWLIM was close behind.

Table 5. Results of Query 2 in detail

| Repository | Data-set size | Answer | Correct answer | Min time (msec) | Triple counting |
|--------------|---------------|--------|----------------|-----------------|-----------------|
| Allegrograph | 10000 | yes | yes | 31 | 88 |
| Big Owlilim | 10000 | yes | yes | 48 | 86 |
| Allegrograph | 50000 | yes | yes | 46 | 107 |
| Big Owlilim | 50000 | yes | yes | 72 | 115 |
| Allegrograph | 70000 | yes | yes | 69 | 115 |
| Big Owlilim | 70000 | yes | yes | 84 | 79 |
| Allegrograph | 100000 | yes | yes | 78 | 132 |
| Big Owlilim | 100000 | yes | yes | 97 | 112 |

From the Table 5, it can be seen that both stores yielded usable data for this query.

Query 3:

This query which is relatively complex attempts to find different kinds of Sport Car plus SUV car. After that, the query defines those cars which are City Car that use Fuel and returns back their types.

```
SELECT ?x ?y ?z ?w WHERE
{ ?x rdfs:subClassOf "SUVCar"
  ?z rdfs:subClassOf "CityCar"
  "FuelType" rdfs:subClassOf ?z
  ?y rdfs:subClassOf "SportsCar"
  ?w rdfs:subClassOf "FuelType" . }
```

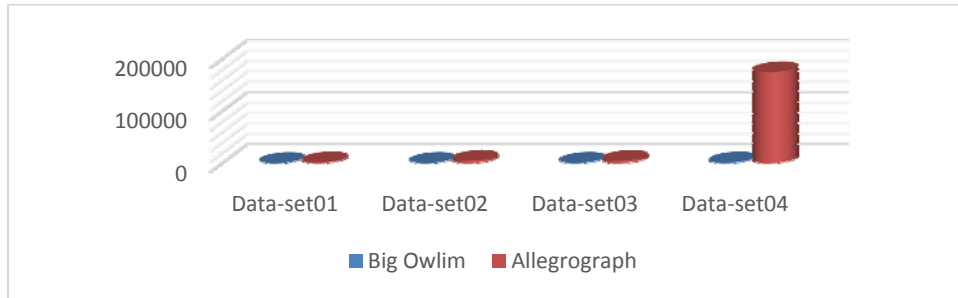


Figure 5. Query 3 outcomes

Allegrograph had the lowest response times up to dataset04, with OWLIM a close second. However, their response at dataset04 was 5 times faster than Allegrograph (Figure 5).

Table 6. Results of Query 3 in details

| Repository | Data-set size | Answer | Correct answer | Min time (msec) | Triple counting |
|--------------|---------------|--------|----------------|-----------------|-----------------|
| Allegrograph | 10000 | yes | yes | 1495 | 90 |
| Big Owl | 10000 | yes | yes | 48 | 112 |
| Allegrograph | 50000 | yes | yes | 4532 | 107 |
| Big Owl | 50000 | yes | yes | 72 | 85 |
| Allegrograph | 70000 | yes | yes | 4376 | 115 |
| Big Owl | 70000 | yes | yes | 84 | 112 |
| Allegrograph | 100000 | yes | yes | 174321 | 105 |
| Big Owl | 100000 | yes | yes | 97 | 100 |

Query 4:

This query has been provided to show details of several customers, it deals with large strings. The result set grows linearly with ontology size. This query generally returns fairly quickly at small ontology sizes, but slowly at larger sizes.

```
SELECT ?y ?X ?z ?A ?B ?C ?m ?n ?o ?T ?U ?V WHERE
{ "Tara" rdfs:subClassOf ?X
  "Zohre" rdfs:subClassOf ?z
  "Paria" rdfs:subClassOf ?y
  ?z rdfs:subClassOf ?C
  ?X rdfs:subClassOf ?B
  ?y rdfs:subClassOf ?A
  ?C rdfs:subClassOf ?o
  ?A rdfs:subClassOf ?m
  ?B rdfs:subClassOf ?n
  ?m rdfs:subClassOf ?T
  ?n rdfs:subClassOf ?U
  ?o rdfs:subClassOf ?V . }
```

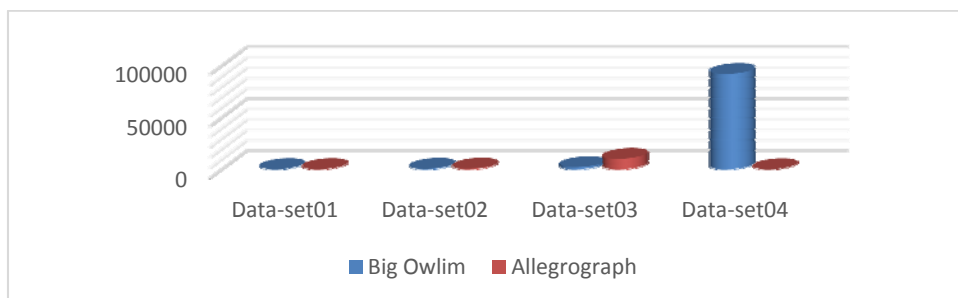


Figure 6. Query 4 outcomes

It can be seen that AllegroGraph had a failure at dataset04 triples. And OwlIm yielded usable data at all ontology sizes (Figure 6).

Table 7. Results of Query 4 in detail

| Repository | Data-set size | Answer | Correct answer | Min time (msec) | Triple counting |
|--------------|---------------|--------|----------------|-----------------|-----------------|
| Allegrograph | 10000 | yes | Yes | 113 | 113 |
| Big OwlIm | 10000 | yes | Yes | 116 | 110 |
| Allegrograph | 50000 | yes | Yes | 583 | 95 |
| Big OwlIm | 50000 | yes | Yes | 451 | 116 |
| Allegrograph | 70000 | yes | Yes | 10001.5 | 330 |
| Big OwlIm | 70000 | yes | Yes | 1948 | 115 |
| Allegrograph | 100000 | No | - | - | - |
| Big OwlIm | 100000 | yes | yes | 91841 | 113 |

OWLIM performed the best on this query. AllegroGraph running the slowest at the higher ontology sizes. AllegroGraph had some of the faster times at Dataset01, 02 and 03; however, it did not yield results at dataset04 triples (Table 7).

Query 5:

This query is designed to test disjoints. Actually this query tries to find all classes which are disjoint with Vehicle, as well as that disjoint class which has Benz in its subclasses and also those classes which have City Car in their subclasses and are disjoint with BMW3. This is the most complex query in this project.

```
SELECT ?x ?y ?z ?v ?w WHERE
{ ?y owl:disjointWith "Vehicle"
  ?z rdfs:subClassOf "Vehicle"
  ?x owl:disjointWith "Vehicle"
  "CityCar" rdfs:subClassOf ?z
```

```
?v rdfs:subClassOf ?x
"Benz" rdfs:subClassOf ?v
?w rdfs:subClassOf "CityCar"
"BMW3" owl:disjointWith ?w . }
```

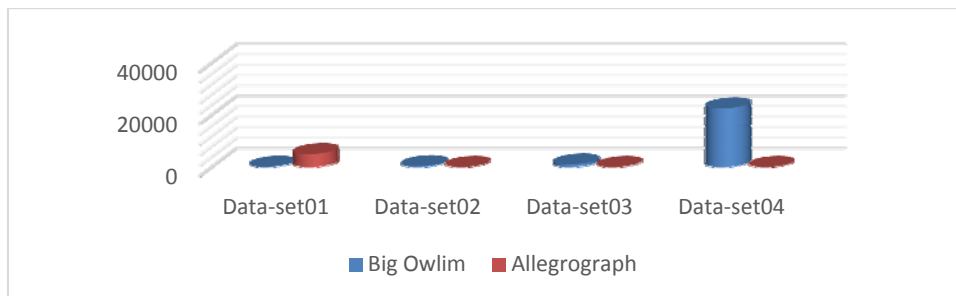


Figure 7. Query 5 outcomes

This query as is shown in the figure 7 generally returns fairly quickly at small ontology sizes, but slowly at larger sizes. And also AllegroGraph did not yield usable data above dataset01 sizes.

Table 8. Results of Query 5 in detail

| Repository | Data-set size | Answer | Correct answer | Min time (msec) | Triple counting |
|--------------|---------------|--------|----------------|-----------------|-----------------|
| Allegrograph | 10000 | yes | yes | 4931 | 127 |
| Big OwlIm | 10000 | yes | yes | 98 | 120 |
| Allegrograph | 50000 | No | - | - | - |
| Big OwlIm | 50000 | yes | yes | 313 | 83 |
| Allegrograph | 70000 | No | - | - | - |
| Big OwlIm | 70000 | yes | yes | 983 | 62 |
| Allegrograph | 100000 | No | - | - | - |
| Big OwlIm | 100000 | yes | yes | 22509 | 113 |

Allegro Graph only ran at dataset01, but had response times that were much higher than the sub-second responses of OWLIM (Table 8).



4.5. Summary

As our benchmarking over these 2 repositories have finished, I would like to highlight some weaknesses and strengths of each triple store regarding these indicated results as a summary of this part of project.

1. AllegroGraph was generally slower than OWLIM, especially for larger ontology sizes. It could not execute query 5 for larger ontologies. This was probably the hardest query for both systems. In addition it performed better than OWLIM on query 2.
2. OWLIM did the best on query 3. It had a close second to AllegroGraph on query 2. A part of these presented point are briefly provided in Table 9 as a short summary.

Table 9. Summary of Query Analysis

| Query | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 |
|--------------|----|----|----|----|-----------|--------------|
| Allegrograph | | | | | Dataset04 | Dataset02-04 |
| Big Owlrim | | | | | | |

| | |
|-------------------------|---|
| Has correct data- done | |
| Did not run or time out |  |
| Incorrect data |  |

5. Conclusions and Future Work

Based on the evaluation effort, analysts have understood that “the current state of RDF persistent stores, “triple stores”, is not at a sufficiently mature level to justify recommending their use in a production system being used on a daily basis” [9], [16].

5.1. Evaluation Summary

If we want to summarize the evaluation briefly, we should say that it was felt that OWLIM performed the best. OWLIM was often much better at the lower ontology sizes, up to dataset02. However, at the larger ontology sizes, dataset03 and dataset04, OWLIM’s throughput dropped off dramatically in many cases. This assessment, which was based primarily on performance and stability considerations, has been evaluated in three main levels. These levels were; General, Functional and Performance.

5.2. General Assessment

In the following Table 10, we captured some important factors that we faced with them during this project and play a significant role in assessing each system.

Table 10. General assessment

| Parameters | Features | Allegrograph | Big Owlrim |
|------------|-------------------|--------------|------------|
| Usability | Easy to install | 100% | 25% |
| | Easy to develop | 75% | 75% |
| | Overall usability | 75% | 75% |
| Support | Detecting defects | 15% | 15% |
| | Documentation | 100% | 100% |
| Licensing | Overall Support | 50% | 80% |
| | how works | 50% | 50% |

Generally, OWLIM got high marks for usability and support. And licensing of all approaches is in the middle level.

AllegroGraph got good marks for usability, and overall support due to the willingness of the Franz team to engage and work on issues [11]. However, they do not have a defect tracking system.

5.3. Functional Assessment

An overall qualitative assessment of perceptions is captured in Table 11.

Table 11. Functional assessment

| Parameters | Features | Allegrograph | Big Owlilm |
|------------------|-------------------------------|--------------|------------|
| System | Architecture and overall view | 60% | 95% |
| | batch loading | 40% | 90% |
| Data import | data formats supported | 40% | 90% |
| | overall view | 40% | 90% |
| API | SPARQL | 90% | 90% |
| | overall view | 50% | 85% |
| Querying | overall view | 50% | 90% |
| Infrencyng | overall view | - | 90% |
| Interoperability | overall view | 50% | 50% |
| Operational | overall view | - | 40% |

5.4. Performance Assessment

As 2 selected semantic repositories were analyzed, it can be claimed that, overall, based on performance and usability, OWLIM was deemed the best repositories (based on Table 11).

5.5. Future Work

“Utilizing semantic web technologies in commercial applications requires confidence by the decision makers that the underlying semantic repositories can deliver the required quality of service while managing the overhead of processing the metadata of potentially huge amount of information organized in complex taxonomies” [17]. This study explores the analyzing of two semantic repositories and my idea to benchmarking better and more accurate for future and further work is expanding factors of assessment. This including adding more triples to billion ones, performance impact of simultaneous users and transaction-related processes, modification test such as insertion, update and deletion operations. Furthermore, some additional supports can be provided to improve the query response times. For example, to ensure the freshness of data, some efficient updates on documents can be organized. Also query optimization techniques can help to improve the query response time.

Based on these mentioned points I can summarize my recommendation for future work in as follow:

- deployment of more repositories
- development of more ontology
- development of SPARQL syntax in more complex queries
- more complex operations in repositories to find out more accurate about response time such as update, deletion, modification and so on

And finally, in future I would like to examine other semantic repository architectures to explore more about each differentiation aspects.

References

- [1] Seaborne, Andy. SPARQL Query Language for RDF. [Online] january 28, 2008. [Cited: July 28, 2011.] <http://www.w3.org/TR/rdf-sparql-query/#grammar>.
- [2] *Benchmarking over a Semantic Repository*. Yadav, Pranjul and Samala, Vinith. 2010, IEEE, ICoAC, pp. 51-59.
- [3] Wiley, John and Sons. *Ontology Evolution. Semantic Web Technologies*. online : Wiley, 2006, pp. 51-70.
- [4] *Designing a Semantic Repository*. Casanave, Cory. 2007, Model Driven Solutions, pp. 1-5.
- [5] Tauberer, Joshua. What is RDF and what is it good for? *rdfabout*. [Online] January 2008. [Cited: August 05, 2011.] <http://www.rdfabout.com/intro/>.

- [6] Freeman, James. Methods of Rule-Based Systems. *ai-depot*. [Online] 2009. [Cited: July 13, 2011.] <http://ai-depot.com/Tutorial/RuleBased-Methods.html>.
- [7] Kiryakov, Atanas, Ognyanov, Damyan and Manov, Dimitar. OWLIM-a Pragmatic Semantic Repository for OWL. *ontotext*. [Online] 2004. [Cited: 07 12, 2011.] https://dx.doi.org/10.1007/11581116_19
- [8] *Semantic Repository for RDF(S) and OWL*. Group, Sirma. 2009, OWLIM Primer, pp. 6-24.
- [9] *What Is RDF*. Tauberer, Joshua. 2006, XML, pp. 1-3.
- [10] *OWL Web Ontology Language*. Harmelen, Frank van and McGuinness, Deborah L. 2004, W3C, pp. 2023-2050.
- [11] Franz. AllegroGraph. *franz Inc*. [Online] february 22, 2010. [Cited: july 27, 2011.] <http://www.franz.com/agraph/allegrograph/>.
- [12] unknown. Comparison of Triple Stores. *bioontology*. [Online] Google Docs, october 13, 2005. [Cited: August 04, 2011.] http://www.bioontology.org/wiki/images/6/6a/Triple_Stores.pdf.
- [13] Franz. SPARQL API reference. *Franz*. [Online] 2005-2011. [Cited: August 06, 2011.] <http://www.franz.com/agraph/support/documentation/v4/sparql-reference.html>.
- [14] *AllegroGraph RDFStore*. Roychowdhury, Ruku and Kariwala, Shagun. 2011, Database and the Web, pp. 1-11.
- [15] Stoilov, Dobri. Primer Background Knowledge. *Ontotext*. [Online] May 24, 2011. [Cited: July 23, 2011.] <http://owlim.ontotext.com/display/OWLIMv41/Primer+Background+Knowledge>.
- [16] Revelytix. *Triple Store Evaluation*. s.l. : Revelytix, 2010.
- [17] *A Pragmatic Approach to Semantic Repositories*. Thakker, Dhavalkumar, et al. Nottingham : Springer-Verlag Berlin Heidelberg, 2010, Vol. 1, pp. 379-393.