A Pragmatic Approach to Semantic Repositories Benchmarking

Dhavalkumar Thakker[1,2], Taha Osman[1] , Shakti Gohil[1], Phil Lakin[2]

[1]Nottingham Trent University, Clifton Lane, Nottingham NG11 8NS, UK
[2]Press Association, 16 Castle Boulevard, Pavilion House, Nottingham, UK
Email: {Dhaval.Thakker, Phil.Lakin}@pressassociation.com,
{Taha.osman, N0239676}@ntu.ac.uk

Abstract. The aim of this paper is to benchmark various semantic repositories in order to evaluate their deployment in a commercial image retrieval and browsing application. We adopt a two-phase approach for evaluating the target semantic repositories: analytical parameters such as query language and reasoning support are used to select the pool of the target repositories, and practical parameters such as load and query response times are used to select the best match to application requirements. In addition to utilising a widely accepted benchmark for OWL repositories (UOBM), we also use a real-life dataset from the target application, which provides us with the opportunity of consolidating our findings. A distinctive advantage of this benchmarking study is that the essential requirements for the target system such as the semantic expressivity and data scalability are clearly defined, which allows us to claim contribution to the benchmarking methodology for this class of applications.

1 Introduction

Based on the concept of autonomous interpretation of machine-understandable metadata, semantic web technologies can deliver intelligent management of user-transparent access to an increasingly complex mesh of interrelated information, which makes these technologies especially appealing to organizations with complex information taxonomy and rich data sets such as the BBC [1], Reuters [2] and Yahoo [3]. However, to promote the adoption of the semantic web technologies beyond organisations that are resourceful in technology-related innovation, clear benchmarks are required that indicate that the tools facilitating the deployment of the semantic technologies are capable of cost-effectively handling potentially enormous amounts of data and increasingly complex information structures. There are many aspects for the organisations to consider: the expertise required for semantically enabling the organization's information infrastructure, the costs involved in superimposing the extra layer of meta-data and the overheads related with processing it, the technical challenges in synchronising with existing data stores, etc. In this study, we focus on evaluating the computing engine of the semantic web technologies, semantic repositories (SR).

Kiryakov et al. define a semantic repository as "a tool, which combines the functionality of an RDF-based DBMS and an inference engine and can store data and evaluate queries, regarding the semantics of ontologies and metadata schemata." [4]. As semantic technologies become more lucrative, an increasing number of commercial and freeware semantic repositories are being offered. These repositories vary significantly at a number of levels that might affect their deployment decision in the target systems, to mention few: supported query languages, semantic expressivity (reasoning capability), load and query response times, and scalability. It might also be necessary to analyze the combined effect of one or two parameters in the target systems, for instance the capacity of the semantic repositories in handling increasing dataset sizes has to be considered in tandem with the supported semantic expressivity and the retrieval throughput.

A distinctive feature of this study is that it is motivated by the practical deployment requirements for semantically-enabling an existing application for a digital images retailer's retrieval and browsing engine. This allows us to inform the benchmarking exercise about the precise essential and desirable requirements of the semantic repository, which we also claim presents a roadmap for benchmarking this rich class of applications.

We uniquely classify the benchmarking parameters into non-functional (analytical) and functional (practical). The analytical parameters, such as expected level of reasoning and query language support aid in narrowing down the pool of benchmarked semantic repositories, while the practical parameters such as the query response time helps to select the optimum repository for the target system.

In order to consolidate our results, we use a public benchmark that satisfies the requirements of our target system (the University Ontology benchmark - UOBM [5]), as well as devising a dataset from the applications knowledge base. This allows us to consolidate our results and vet them against published work on semantic repositories benchmarking.

The rest of the paper is structured as follows: section 2 surveys the current semantic repositories benchmarking approaches. Section 3 discusses the details the commercial deployment case study for benchmark. Section 4 studies the benchmarking methodology, while section 5 analyzes the experimental results. The paper's conclusion and plans for further work is detailed in section 6.

2 Benchmarking Semantic Web Technologies

Benchmarking semantic repositories is significantly more challenging than that of RDMS, primarily because of the complexity of evaluating the additional reasoning layer. For semantic repositories, unlike relational databases there exists no standard benchmark similar to TPC [6].

Benchmarking approaches can be classified into studies of the reasoning engines and studies of the semantic repositories (the RDF stores and the inferencing engine). The first approach [7] [8] mainly targets the description logic community or developers interested in optimising the reasoning engines and integrating them into their semantic datastores. This benchmarking exercise is motivated by the requirement of deploying semantic web technology in a commercial search and browsing engine,

and hence is chiefly interested in benchmarking approaches evaluating ready-to-deploy semantic repositories. Below we discuss some of the published work on semantic repositories benchmarking.

The Lehigh University Benchmark LUBM [9] was the first standard platform to benchmark OWL systems, but it gradually fell behind with the increasing expressivity of OWL reasoning and could not support a modest reasoning logic such as OWL Lite [10]. The University Ontology Benchmark (UOBM) benchmark [5] was devised to improve the reasoning coverage of LUBM by adding TBox axioms that make use of all OWL Lite and OWL DL constructs. Both benchmarks predate the advent of the SPARQL RDF query language, and hence do not evaluate advanced query features such as OPTIONAL filters and UNION operations [11].

[12] introduces the Berlin SPARQL benchmark (BSBM) for comparing the performance of systems that expose SPARQL endpoints. The benchmark is built around an e-commerce use case, which extends its benefits to similar class of applications desiring to embrace semantic technologies. BSBM focuses provides comprehensive evaluation for SPARQL query features. However, the benchmark does not evaluate update operation on the RDF stores and has no information on precision/recall and primarily targets the throughput results with the assumption that the systems are precise and complete. The list of benchmarked systems by BSBM is not exhaustive.

All the works discussed above represent valuable contributions to the methodology of semantic technologies benchmarking and can also offer reusable datasets and query results at the practical level, which allow us to compare our results with other published benchmarking studies. However, we believe that for the decision to adopt a specific semantic repository for the deployment of our commercial application can only be based on a benchmarking study that mirrors the demands of our semantic retrieval and browsing engine within an enterprise setup. This entails using a similar dataset, evaluating the required level of expressivity, and considering the evaluation of all established semantic repositories including freeware systems such Jena TDB [13] and Sesame [14], as well as commercial offerings such as Allegrograph [15], Virtuoso [16] and BigOWLIM [17].

3 Commercial Deployment Case Study

This study has been conducted with a commercial deployment case study at the heart of its objectives. This section gives more details on the motivations of the exercise with the nature of the proposed application.

3.1. Motivation

Press Association (PA) is the UK's leading multimedia news and information provider and supplier of business-to-business media services. The photography arm of the PA, Press Association Images is looking into the utilization of semantic web technologies to improve the image browsing experience for their customers. Therefore this study focuses on the particular concerns of this implementation, such as the sheer volume of

data and other fundamental performance measures such as load time, query response time and level of inference.

Along with gauging potential benefits of semantic technologies, our motivation to perform this benchmarking is to evaluate the scalability of current semantic technologies in handling potentially large datasets while maintaining reasoning and retrieval throughput. Our concern about the scalability stems from the fact that unplanned use of the OWL properties can result into impractical reasoning complexity. For the benefit of the reader, it is useful to highlight the complexity of the ontology we utilize in our implementation. The PA Images ontology in its current form has total 147 classes, 60 object properties and 30 object properties. The OWL species of the ontology is OWL-DL and the DL expressivity is ALCHOIN (D).

Apart from the standard classification hierarchy and object and data type properties, we utilize what we see as the "smart" properties of OWL. One example of these properties is the inverse property *"owl:inverseOf",* which implicitly allows defining relationship in both directions [18]. For example, an application based on PA Images ontology has a relationship category where father-son, parent-child, husband-wife bi-directional relationship are heavily utilized. The other property which we find very useful is the value constraint in OWL-DL *"owl:hasValue"* that links a restriction class to a value. For example, "Actor is a person who has value for the property *profession* equal to *acting*". This is very useful property as it allows for the automatic classification of individuals into categories depending on the value of some of their properties. This is a desirable functionality as instead of relying on the annotator to remember category of an entity while entering data it could be automatically inferred based on the properties of entities.

These properties make reasoning challenging and require a level of language expressivity in the domains of OWL-LITE and OWL-DL. For example, when inverse properties are used in some of the reasoning engines, it prohibits the use of highly efficient optimization techniques [10]. The aim of this benchmarking study is to investigate how various repositories will handle such reasoning requirements while maintaining acceptable query response time.

As discussed, the PA Images ontology is light-weight DL ontology. For increasing the confidence of our benchmarking, we researched the availability of published benchmarks with datasets with characteristics similar to ours, i.e. datasets that support OWL-DL level of reasoning and contain few million triples. We selected UOBM for this purpose as the prime focus of the UOBM dataset has been inferencing and reasoning which meets our requirements. UOBM also supplies dataset of variable length ranging from 0.2 million triples in UOBM-1 to around 6.6 million triples in UOBM-30. We discarded using the BSBM [12] dataset as it is primarily designed to test Repositories in terms of RDF and SPARQL support instead of higher order of reasoning capabilities. The DBpedia [19] dataset was also not considered relevant to the task due to the lack of formal ontology structure in its datasets as it is governed by combination of external ontologies SKOS, UMBEL and WordNet in addition to its own custom ontology, making judgment on precision and recall challenging task.

3.2 PA Dataset

In this section, we provide useful information on the PA Dataset which contains three components: PA Images ontology, Knowledge base and image captions.

1. PA Images ontology

The first component of the dataset is layered owl-dl ontologies: one of these ontologies defines the entities in our domains primarily consisting of sports, news and entertainment images. This ontology contains entities such as footballers, sport teams, politicians, stadiums, tournaments, actors, award events. The set of ontologies contains another ontology – a media ontology defining image metadata attributes.

2. PA Knowledge base(KB)

PA KB is the data operating on the PA ontology. Manual generation of such data as part of a knowledge base is a colossal and quite cumbersome task. However, we alleviated the burden of manual compilation of creating such KB by leveraging the rich amount of structured knowledge publically available in DBpedia [19]. We see DBpedia being at the centre of the linked data cloud (LoD) efforts [20] mainly due to its knowledge coverage across multiple domains. LoD is a medium for domain experts to come together and share the knowledge about the domains they are expert in. We have successfully used SPARQL CONSTRUCT [11] queries to achieve ontology mapping between PA Images and DBpedia ontologies to extract the instances from DBpedia KB and generate a clean, contexualised PA KB.

3. Image captions

Image captions triples were generated randomly using an instance generator that links an image with list of entities from the KB. The images represent an adequate mixture of indices of People (player, actor, politician) at Events (Tournaments, Signing, Awards etc), or people seen with other people. Apart from the ontology, the dataset is expressed in N-Triples serialization fromat. Table 1 gives more information on the dynamics of PA dataset components.

Dataset	No of Triples	Entities/Images	Disk space
KB	6.6 Millions	1.2 Millions	1.23 GB
Image captions	8 Millions	5 Millions	1.57 GB
Schema			136 KB

Table 1. PA Dataset dynamics

4 Benchmarking Methodology

4.1 Semantic Repository Selection and Benchmarking Environment

We have selected the Semantic Repositories for this benchmarking based on the following selection criteria.

1. Minimum level of inference required is RDFS reasoning
2. Support for SPARQL or SPARQL-like RDF query language
3. As per the definition of the Semantic Repository, any tool that is combination of reasoner and storage backend. This criterion ruled out the selection of Pellet, Racer,

and KAON2 as these tools need to be used in conjunction with the databases. The repositories that satisfy the aforementioned criteria hence selected for benchmarking are: Virtuoso [16], Allegrogaph [15], Sesame [14], Jena TDB [13], Oracle [23] and BigOWLIM [17].

Hardware Setup. The experiment was conducted on a DELL workstation (processor: Intel Core 2 Quad Q9400 2.66GHz; memory: 8GB DDR2 667; hard disks: 160GB (10,000 rpm) SATA2, 750GB (7,200 rpm) SATA2) running Windows XP professional x64 edition, 2003, Service pack 2 as operating system using Java version 1.6.0_16.

4.2 Benchmarking parameters

We uniquely classify the benchmarking parameters into non-functional (A=analytical) and functional (P=practical). This section gives more information on these parameters.

1. Identification of the Semantic Repositories storage technology in Native, Memory-based or Database-based storage systems (A). Both native and database based techniques store data persistently while memory-based stores utilize main memory to store RDF graphs. The database technique uses RDBMS to store data while native store use a flat file structure. Understanding the behaviour of the semantic repositories in these classifications helps predicting the store's behaviour under various conditions, for example scalability of the memory based repositories will be limited to the amount of memory space available.
2. Identification of Semantic Repositories in forward, backward or hybrid chaining reasoning strategies (A). The forward-chaining repositories support materialisation where they compute and store the possible inferencing of facts at load time. The backward-chaining repositories perform the inferencing at the query time.
3. Load time (P) is a standard benchmarking parameter that measures the performance of repositories in terms of the time it takes loading datasets. We believe that for the class of application similar to ours, update time is more relevant as the load time is generally one-off and could be performed offline. We cover update time in 6 below.
4. Using query response time (P), we measure the time for issuing a query and obtaining the results. We have created a query-mix that exploits OWL-DL and OWL-Lite constructs from PA images ontology and we use the queries provided by the UOBM to exploits different construct of UOBM.
5. We use query results analysis (P) to measure completeness and correctness of the query results. The results of this analysis will allow us to judge a repository as sound, complete or both. With the query results analysis, where possible we also want to analyze the results to verify the OWL properties supported by a Semantic Repository under the dataset load in this experiment. The repositories advertise type of inferencing supported by them however as observed by [7] [10], for larger datasets most of the tools seem to fail simplest of OWL reasoning queries.
6. Most triple-stores use SPARQL for querying RDF; however there is no standardization for modification to RDF data. SPARQL/Update is an effort to standardize the update language for RDF graphs for updating graphs with modification operations. The alternative is to use a programming language and custom

APIs. With RDF store update tests (A&P) parameter, we test and analyze repositories by schemata and data update queries and indentify the repositories that use either SPARQL/Update or custom APIs for doing so.

7. The identification of repository support for RDF serialization formats (A) allows us to study different serialisation (RDF/XML, N-Triples, N3, Turtle) offered by the repositories.
8. We also want to analyze the scalability (P) of the repositories, i.e. loading and querying time of semantic data is linear with the dataset sizes.
9. Reasoner Integration (A) is a parameter designed to identify the reasoners integration supported by a repository.
10. We will also identify query languages supported (A) by a repository.
11. Inferencing and reasoning is computationally challenging task and clustering support (A) is helpful for practical implementations. We want to identify semantic stores that supports clustering configuration in their standard setup.
12. From the application development perspective, we want to analyze the client API supported (A) in various programming languages.
13. Identification of different platform supported (A) by a semantic repository. This could be a crucial factor for many organisations.
14. The trend to move relational data to RDF graphs can be encouraged by repositories that have in-built support for converting relational data into RDF data (A). We will identify the stores that have in-built support for such functionality.

Table 2 shows our observation for the selected repositories.

Parameters	Jena TDB	Virtuoso	Allegrograph	BigOWLIM	Sesame	Oracle
Storage Type	Native	Native, RDBMS-based	Native	Memory, Native	Memory, Native, RDBMS	Native
Reasoning strategy	Backward chaining	Backward Chaining	Backward chaining	Forward chaining	Forward chaining	Forward chaining
Serialization format	rdf/xml, n3,ntriples	rdf/xml, n3	rdf/xml, n3, ntriples	rdf/xml, n3, triples	rdf/xml, n3, triples	rdf/xml, ntriples
Reasoner Integration	Built-in	Built-in, Jena, Sesame, Redland	Built-in, Jena, Racerpro, Sesame	Built-in, Sesame	Built-in	Built-in, Jena
RDFview support	No	Yes	Not conclusive	No	No	Not conclusive
RDF Update	sparql/ udpate	sparql/ update	api	api	api	api
Query Language Support	tql, sparql	sparql, spasql	sparql, twinql, serql, prolog	sparql, serql	sparql, serql, rql	bespoke, sparql
Clusteri	No	Yes	Yes	Not	No	Not

ng				conclusive		conclusive
Client side	Java	PL/SQL Java, C	Java, Python, Ruby, Lisp,	Java	Java	PL/SQL, Java
Platform supported	Windows, Unix, Mac, Solaris	Windows Unix,Mac, Solaris	Mac, Windows, Unix, Solaris	Windows, Unix, Mac, Solaris	Windows, Unix, Mac, Solaris	Unix, Solaris, Windows, Mac
Licensing	Free	Free, Commercial	Free, Commercial	Free, Commercial	Free	Commercial

Table 2. Analytical Parameter observation

5 Benchmarking results

5.1 UOBM Dataset results and Analysis

UOBM Dataset Load timings. Although UOBM-30 is the super set of other datasets of UOBM, as an opportunity to gauge the load time scalability, we decided to load all the four datasets. The datasets were loaded in four different graphs as these datasets contain overlapping data and if loaded in the same graph it will generate redundancy and unexpected results when queried. The aim here was to evaluate the load time as the dataset size increases.

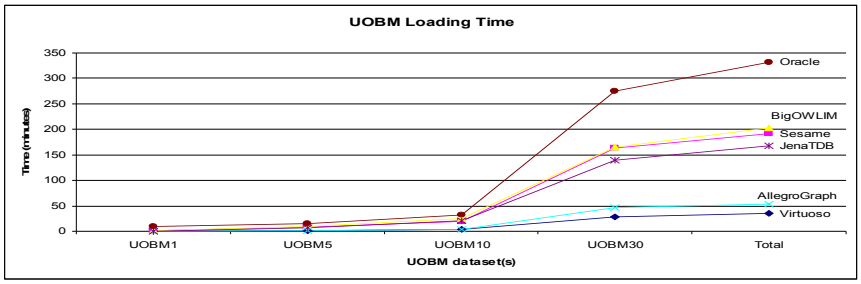

Figure 1. UOBM Loading Time

From the graph in Figure 1, we can clearly identify that virtuoso performs best among these tools for loading A-Box by taking approx 27.5 minutes to load UOBM data-set with 30 university and allegro-graph is the closest competitor of virtuoso in loading data into a store. BigOWLIM performs second slowest among the tools as it performs the "forward-chaining" of facts and stores them explicitly. Oracle is the slowest in loading all four datasets as it take more time than BigOWLIM in forward chaining process. Another interesting observation can be made about the performance of these repositories in terms of how well they scale for the increase in dataset sizes. Virtuoso and Allegrograph are quite consistent and scalable in terms of dataset sizes and takes almost same amount of time (linear) as the load increases.

Query Result and Execution Speed Analysis. UOBM supplied 15 queries with different levels of complexity where 12 queries fall under OWL-Lite and the remaining 3 queries are of OWL-DL expressivity. To our knowledge, the UOBM benchmark has not published a query result set. Therefore we had to generate the answer keys in order to enable checking the correctness and completeness of the

returned results. We generated answer keys by modifying queries to remove complex inference and firing them against the benchmarked repositories. Our precision and recall analysis is based on this and for the scrutiny we publish the result sets [24].

Next, we analyze query response times taking into account the context of the precision and recall. In the Table 3, "N" against a query indicates empty result set when at least some results were expected. (P) next to a timing indicates that the repository took that much amount of time but returned partial results.

From Table 3, we can conclude that BigOWLIM answers 12 out of 15 queries completely while answering query no. 9 partially and performs the execution faster in most of the cases with the average time 0.038 seconds. Sesame answers 4 queries completely while answering 2 queries partially. Average time to answer these queries is 0.09 seconds. Allegrograph answers 7 queries completely, while answering 2 queries partially. However Allegrograph is the slowest and takes on average 219 seconds to answer queries. Virtuoso has the worst recall, as virtuoso answers 1 query partially and the other completely at the average speed of 3.388 seconds. We would also like to draw attention to virtuoso's different behaviour in answering a query from a SPARQL end-point and from the Jena Adapter API. From the API, Virtuoso repository is able to answer only 2 query while from a SPARQL end-point it answer 3 queries. Moreover, for the UOBM query 1, virtuoso's SPARQL end-point returns 21 triple which is correct as well as complete but when we fire the same query from the API it returns 105 triples. 21 triples out of these 105 triples are correct. We believe that this can be attributed to a bug in the API implementation rather than problem with the soundness of the repository.

	Execution Timings (seconds)					
No.	Virtuoso	Allegrograph	Oracle	Sesame	Jena TDB	BigOWLIM
Q1	6.766 (P)	21.921	0.141	0.203	0.031	0.047
Q2	N	8.906(P)	N	0.001(P)	0.001(P)	0.062
Q3	N	651.237	N	0.109	0.016	0.062
Q4	N	N(infinite)	N	0.14	120	0.063
Q5	N	1.281	N	N	N	0.047
Q6	N	1153.025	N	N	N	0.047
Q7	N	300.12	N	N	N	0.001
Q8	N	6.843(P)	N	N	N	0.031
Q9	N	N	N	N	N	0.031(P)
Q10	0	0.25	0.001(P)	0.001	0.001	0.016
Q11	N	N(infinite)	0.001(P)	0.094(P)	N(infinite)	0.062
Q12	N	476.507	N	N	N	0.016
Q13	N	N	N	N	N	N
Q14	N	N(infinite)	N	N	N	0.016
Q15	N	N	N	N	N	N

Table 3. UOBM Query execution speed and result analysis

Jena TDB answers 4 queries completely while one partially. Average speed is 24 seconds which is skewed by the time it takes to answer Q4. Oracle answers 3 queries, among them one completely and two partially at the average speed of 0.048 seconds.

Closer examinations of queries show that queries Q5 and Q7 are not answered by all the repositories except Allegrograph and BigOWLIM. Queries Q5 and Q7 involves transitive (*owl:TransitiveProperty*) property based inference. As this is the case for both of the queries it is possible to conclude that this property is not supported by Virtuoso, Sesame, Jena TDB and Oracle. Q6 relies on semantic repositories to support (*owl:inverseOf)* and all except Allegrograph and BigOWLIM answers this query. However, to consolidate the conclusion that other tools do not yet support inverse property, we can rely on the PA Dataset results as the dataset includes some queries of the same complexity. Q10 requires symmetric property support and is correctly answered by all the SRs. Q13 requires support for OWL lite-level of cardinality and not answered by any of these tools. OWL Lite cardinality restrictions allow statements concerning cardinalities of value 0 or 1 i.e. min 1, max 1. Q15 requires support for dl-level of cardinality that is not answered by any of these tools.

At the time of compiling this paper, correspondence with the semantic technologies team at Oracle established that they introduced improved OWL reasoning capability in the new release of Oracle (11g Release 2). Unfortunately the release is currently not available for our benchmarking platform, Windows OS. Hence we decided to omit Oracle from experimentation with the PA Data set (below).

5.2 PA Dataset Experiment Analysis

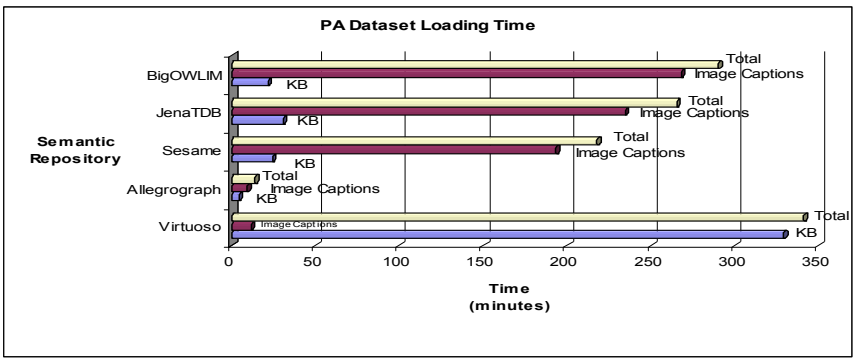

Figure 2. PA Dataset Loading Timings

Loading time for KB and Images. Allegrograph was able to load whole of the dataset in under 15 minutes. This result is inline with the UOBM results as it was comparatively (second place to Virtuoso) faster. Similar to the UOBM benchmark results, BigOWLIM performs slower in loading PA datasets. This pattern repeats for Jena TDB and Sesame, which are again in the list of slower performers as is with UOBM results. However, the major difference in PA Dataset results from UOBM results is the performance of Virtuoso which took the least amount of time in loading all four of the UOBM datasets, for PA Dataset takes the maximum amount of the time

among the repositories. There is an operational distinction between UOBM and PA Dataset, in which UOBM dataset is in RDF/XML serialization where PA Dataset is expressed in N-Triples. Virtuoso's Jena Adapter API lacks the functionality to load N-Triples and we had to load the PA Dataset using TTLP_MT function from the command line, which can explain the store's relatively lengthy loading time.

Query execution speed and results analysis. The list of PA Dataset queries is available here [24]. The results are outlined in Table 4. Carrying forward the observations from the UOBM query results, it is possible to conclude that inverse property of OWL expressivity as required to answer queries 6, 12, 15 is not supported by Virtuoso and Sesame.

In our tests, BigOWLIM was able to answer all the queries. Allegrograph answered all the queries except two. Sesame answered six queries completely while two queries partially. On a one-to-one comparison between Allegrograph and BigOWLIM , two repositories that answered maximum number of queries and between Sesame and BigOWLIM, two fastest repositories, it is clear to see that execution speed-wise BigOWLIM outperforms Allegrograph and sesame for almost all of the dataset queries.

Query No.	Virtuoso	Allegrograph	Sesame	Jena TDB	BigOWLIM
Q1	2.234 (P)	26.422	0.469(P)	0.047	0.219
Q2	N	N	N	N	0.063
Q4	N	N	N	N	0.047
Q5	0.172	1.719	0.141	N	0.078
Q6	N	3.765	N	0.001	0.45
Q7	84.469	28.688	0.203	N	0.093
Q8	0.047	3.39	0.11	0.001	0.062
Q9	0.156	1.782	0.171	N	0.016
Q10	0.001	1.734	0.047	N	0
Q11	N	1.734	0.11	0.001	0.062
Q12	N	16.14	N	N	0.079
Q13	5.563(P)	1.812	0.016(P)	0.001	0.641
Q15	N	1.688	N	N	0.031

Table 4. PA Images Query Execution speed results

Modifications Tests. Modifications to the data and ontology is an important task performed against a SR [12]. Although the complexity of modifications can change considerably across applications and usage patterns, the execution speed and correctness of modifications is vitally important for any commercial application. This area of benchmarking has been ignored so far. As SPARQL specification in its current state provides no implementation of update or delete parameters these experiments also provide an insight to how each SR handles them in absence of standardization.

T-Box/ontology modifications. One of the main advantages of using a semantic ontology is the possibility of loose couplings of schemata from the data. In Virtuoso, the schema is loaded separately from the data and the repository requires any query to inform the SR which schema it shall use for the purpose of inferencing. This is done using *"define input:inference 'schema name'"* prefix as part of the query. We believe that this approach allows maximum loose coupling of data from schemata as the same dataset can be reasoned using different schemata. BigOWLIM protects and places restriction on deleting components of the base schema. In BigOWLIM, the schemata is stored as "imports" parameter of the repository configuration and are treated as "read-only", thus these schemata are protected from delete operations. Jena TDB, Oracle, Allegrograph expects an ontology to be present at its absolute or relative URL; hence modifications could be made to the schemata outside the scope of these repositories.

A-box Insertion operations. We believe that in most of the applications, loading of datasets of size of UOBM dataset is done once, while most of the loadings are insertion in small sizes. Here we test these tools on how they fare under small insertions and utilize the same methods we used for loading the whole of the dataset to perform insertions. The results in the Figure 3 illustrate this with small KB addition and small number of image additions to the PA Dataset. The results provide reassurance that all the repositories (virtuoso and Allegrograph when warmed up), can handle small amount of loading (insertions) relatively fast. It is also important to highlight here that all the repositories have approximately 12 million triples already stored when this loading call occurs.

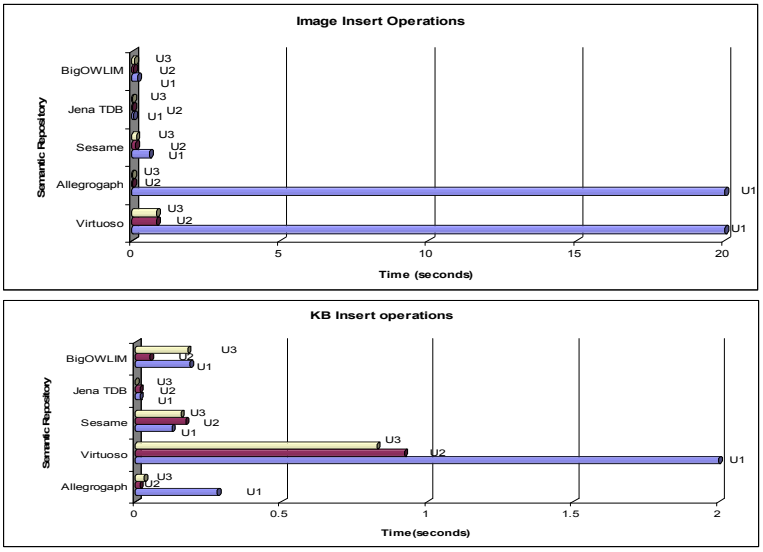

Figure 3. Insertion operations

Update and deletion operations.

U1= updating two actors relationship from “partner” to “spouse”, U2= updating an image caption to identify previously incorrectly identified person, U3= updating ontology to make “Person” and “Group” classes to be disjoint classes., D1= Deleting relationship between two British Royalty., D2= Deleting a player’s playing position, D3= Deleting a band’s genre

Table 5. Modification queries

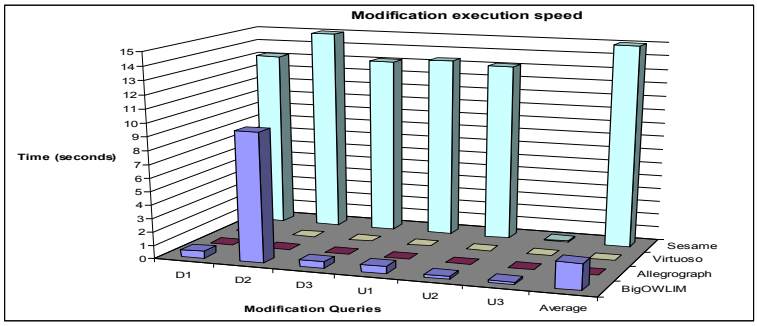

Figure 4. Modification operation results

Again, similar to the small data insertions, it is important to test these SRs on their performance on small routine deletion or updates () that happens in small amounts but in higher frequency. The aim here is to analyze the execution speed and also determine how they deal with modifications in the absence of a standardized SPARQL protocol. We treat the update and delete queries in the same frame and display and compare in the same graph (Figure 4). This is because for most of these tools the update operation is two step operation: a. delete a fact and b. insert a new fact instead of the deleted fact.

In BigOWLIM, deletion of the fact is performed from the API, as the repository does not implement a customized extension of SPARQL. There is also an area of concern for this class of repository that utilizes “forward chaining” as by nature the delete operation is slow, i.e. any fact deletion shall also delete any other facts that are inferred based on them. We found that the latest version of BigOWLIM (3.2.3) provided to us with a major improvement in delete operation, which means that upon delete, BigOWLIM invalidates only the inferred facts which are no longer inferable as opposed to dropping all inferred facts and inferring everything from scratch. However in these experiments, whenever this process (invalidating only the inferred facts) was involved such as in the query D2 the performance of system is slower than other simpler delete operations.

Allegrograph deals with the deletion of triples from the store using the base API and the execution speed is quite fast. Similar to Jena, Virtuoso provides an extension of SPARQL for the update and deletes queries. Using the SPARQL/UPDATE queries, Virtuoso runs very fast. We were not able to perform similar operations with Jena TDB as it runs out of the memory for each of these operations.

6 Conclusions and Future Work

Utilising 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. This paper investigates the benchmarking of the major freeware and commercial semantic repositories for a commercial image retrieval and browsing application. Our benchmarking methodology translates the precise essential and desirable requirements of our application into a set of functional (practical) and non-functional (analytical) parameters for benchmarking the target semantic repositories, and we claim that this methodology will prove useful for benchmarking applications with similar characteristics. In order to consolidate our benchmarking results, we use UOBM, a public benchmark that satisfies the requirements of our target system, as well as devise a dataset from the application's knowledge base.

Our analysis of the benchmarking results established that all the evaluated repositories were sound for both the dataset queries as the query results returned by the repositories were correct for corresponding queries. However none of the benchmarked repositories were able to answer all the queries in the UOBM dataset, and hence we conclude that the evaluated repositories currently cannot handle the OWL reasoning level required to answer the UOBM queries.

In our tests, BigOWLIM provides the best average query response time and answers maximum number of queries for both the datasets. Sesame, Jena, Virtuoso and Oracle offered sub-second query response time for the majority of queries they answer. Allegrograph answers more queries than the former four repositories hence offers better coverage of OWL properties. However, we found that the average query response time for Allegrograph was the highest for both the dataset and believe that this repository requires further optimisation to handle complex OWL capabilities. The modifications operations testing confirmed that the forward chaining repositories offer slower response times compared to the backward chaining repositories. This is especially more noticeable for delete operation where sesame was consistently and BigOWLIM was variably slower in deleting triples.

Our plans for further work involve expanding this benchmark exercise to billion triples of extended PA Dataset and adding extra benchmarking parameters such as the performance impact of concurrent users and transaction-related operations. We would also like to test the new capabilities of the Oracle's semantic repository.

7 References

[1] Kobilarov, G., Scott, T., Raimond, Y., Oliver, S., Sizemore, C., Smethurst, M., Bizer, C., Lee, R.: Media Meets Semantic Web – How the BBC Uses DBpedia and Linked Data to Make Connections. In: European Semantic Web Conference (ESWC2009), pp. 723 - 737, Springer-Verlag Berlin, Heidelberg (2009).

[2] http://www.opencalais.com/

[3] Mika, P.: Microsearch: An Interface for Semantic Search. In: Proceedings of the Workshop on Semantic Search (SemSearch 2008) at the 5th European Semantic Web Conference (ESWC 2008) , June 2, 2008, Tenerife, Spain , Vol. 334CEUR-WS.org (2008)
[4] Kiryakov, A.:Measurable Targets for Scalable Reasoning., Ontotext Technology White Paper, Nov 2007.
[5] Ma, L., Yang, Y., Qiu, Z., Xie, G., Pan, Y., Liu, S.: Towards a Complete OWL Ontology Benchmark. In Sure, Y., Domingue, J., eds.: Proc. of the 3rd European Semantic Web Conference (ESWC'06). pp. 125 - 139 Volume 4011 of LNCS., Budva, Montenegro, Springer (2006)
[6] TPC Database Benchmark, http://www.tpc.org/information/benchmarks.asp
[7] Bock, J., Haase, P., Ji, Q., Volz, R.: Benchmarking OWL Reasoners. In: Proc. Of the ARea2008 Workshop, Tenerife, Spain (June 2008)
[8] T. Gardiner, I. Horrocks, D. Tsarkov, Automatic benchmarking of description logic reasoners, in: Proceedings of the 2006 International Workshop on Description Logics (DL06), Windermere, UK, 2006.
[9] Guo, Y., Pan, Z., Heflin, J.: An Evaluation of Knowledge Base Systems for Large OWL Datasets. In: Proc. of Third International Semantic Web Conference, pp. 274-288, Springer-Verlag Berlin, Heidelberg (2009).
[10] T. Weithöner, T. Liebig, M. Luther, and S. Böhm, What's Wrong with OWL Benchmarks?,Proceedings of the Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006),Athens, GA, USA November 2006
[11] http://www.w3.org/TR/rdf-sparql-query/
[12] Bizer C., Schultz, A.: Benchmarking the Performance of Storage Systems that expose SPARQL Endpoints. In: Proceedings of the ISWC Workshop on Scalable Semantic Web Knowledge-base systems (SSWS), Karlsruhe, Germany (2008)
[13] http://openjena.org/TDB/
[14] Broekstra, J., Kampman, A., Harmelen, A.v.: Sesame: A generic architecture for storing and querying RDF and RDF Schema. In Ian Horrocks and James Hendler, editors, Proceedings of the first Int'l Semantic Web Conference (ISWC 2002.Springer Verlag (2002)
[15] http://www.franz.com/agraph/allegrograph/
[16] Erling, O., Mikhailov, I.: RDF Support in the Virtuoso DBMS.In: Proceedings of the 1st Conference on Social Semantic Web (CSSW), pp. (7-24), Springer Berlin / Heidelberg (2009)
[17] Kiryakov, A.: OWLIM: balancing between scalable repository and light-weight reasoner. In:Developer's Track, WWW2006 (2006).
[18] http://www.w3.org/TR/owl-guide/
[19] Auer, S.,Bizer, C., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In: 2nd Asian Semantic Web Conference. Springer Berlin / Heidelberg (2007)
[20] Hausenblas, M.:Exploiting Linked Data to Build Web Applications. IEEE Internet Computing, vol. 13, no. 4, 68-73 (2009)
[21] Ding, Z., Peng, Y., Pan, R., Yu, Y.: A Bayesian Methodology towards Automatic Ontology Mapping. In:Proceedings of the AAAI-05 C&O Workshop on Contexts and Ontologies: Theory, Practice and Applications, AAAI Press (2005)
[22] Mongiello, M., Totaro, R.:Automatic Ontology Mapping for Agent Communication in an e-Commerce Environment, In: Proceedings of EC-Web, Copenhagen, Denmark pp.21-30, Springer Berlin, Heidelberg (2005)
[23] http://www.oracle.com/technology/tech/semantic_technologies/index.html
[24] http://realizingsemanticweb.blogspot.com/2010/03/benchmarking.html