An Evaluation of Triple-Store Technologies for Large Data Stores

Kurt Rohloff, Mike Dean, Ian Emmons, Dorene Ryder, and John Sumner

BBN Technologies
10 Moulton St.,
Cambridge, MA 02138, USA
{krohloff,mdean,iemmons,dryder,jsumner}@bbn.com

Abstract. This paper presents a comparison of performance of various triple-store technologies currently in either production release or beta test. Our comparison of triple-store technologies is biased toward a deployment scenario where the triple-store needs to load data and respond to queries over a very large knowledge base (on the order of hundreds of millions of triples.) The comparisons in this paper are based on the Lehigh University Benchmark (LUBM) software tools. We used the LUBM university ontology, datasets, and standard queries to perform our comparisons. We find that over our test regimen, the triple-stores based on the DAML DB and BigOWLIM technologies exhibit the best performance among the triple-stores tested.

1 Introduction

There has been an explosion recently in the development of new technologies for various areas of the Semantic Web. Some of these manifold new developments include query processing, protocols and triple-store technologies among many others. In this paper we evaluate the most promising new Semantic Web triple-store technologies.

This evaluation was motivated by a client with an existing Semantic Web application who wanted to evaluate other storage alternatives. Therefore this study focused on the particular concerns of the client's application, such as the sheer volume of data, fundamental performance measures (load time and query speed), and very basic inference.

This study leverages the Lehigh University Benchmark (LUBM) [1,2], a widely accepted methodology for evaluating triple-stores that is relevant to the context of this evaluation. The metrics in the LUBM evaluation methodology include load time, repository size, and the response time of 14 different queries. The 14 queries are also designed to assess the soundness and completeness of the reasoning performed.

To date there have only been a handful of attempts at formal evaluations, studies or benchmarking efforts that characterize key aspects of triple-stores. While these studies provide valuable insight into many of the challenges associated with benchmarking triple-stores, most of these studies address specific ranges of performance metrics measured under targeted test conditions. We focus on evaluating what we have identified as the most promising currently available and beta-test triple-store technologies that can handle very large data-sets (on the order of 1 billion triples.)

The most relevant studies to this evaluation include the MIT Scalability Report, the RDF Scalable Storage report, and the 2003 Semantic Web Tools Assessment. The MIT report [3] was primarily focused on the storage and query response capabilities of triple-stores, and queries with large expected results in particular. The RDF Scalable Storage Report [4] provides an overview of a number of open source/free triple-store implementations that are available and does not include any benchmarks or performance measurements. The Semantic Web Tools Assessment[1], a part of the DARPA Agent Markup Language (DAML) program, documented a qualitative survey of available OWL tools.

The triple-stores we evaluated in this study (coupled with their appropriate query framework) are:

- Sesame 1.2.6 + MySQL 5.0
- Sesame 1.2.6 + DAML DB 2.2.1.2
- Sesame 1.2.6 + SwiftOWLIM 2.8.3
- Sesame 1.2.6 + BigOWLIM 0.9.2
- Jena 2.5.2 + MySQL 5.0
- Jena 2.5.2 + DAML DB 2.2.1.2
- AllegroGraph 1 (Free Java Edition)
- AllegroGraph 2.0.1 (Free Java Edition)

The remainder of the paper is organized as follows: Section 2 presents the methodology and metrics we use to evaluate the triple-store technologies. Section 3 presents the study results, and Section 4 analyzes the results and discusses areas for future work.

2 Evaluation Methodology and Metrics

2.1 Framework

We performed our evaluation in a Java environment using a modified version of the LUBM test harness [1,2]. The modified test harness incrementally loads test data in ten blocks and a full set of queries are run after each incremental load. This allowed us to collect data on how the data load times and query response times vary with the number of triples loaded into the triple-store. The triple stores tested here supported different query languages, and so the LUBM queries were ported into their appropriate query languages (SeRQL and SPARQL.)

Data for our evaluation was generated by the UBA (Univ-Bench Artificial data generator) tool developed by the LUBM authors. The LUBM test data are extensible data created over the Univ-Bench ontology. LUBM datasets represent multiple universities, each with multiple professors, students, graduate students, courses, and departments. The courses, students, graduate students, and professors belong to departments, and students, graduate students, and professors can be assigned to courses.

Most of the triple-stores (MySQL[2], DAML DB[3], SwiftOWLIM, and BigOWLIM[4]) require a query framework. We used Sesame[5] and Jena[6] as query frameworks due to

[1] http://www.daml.org/2003/10/tool-assessment/Repository.html
[2] http://www.mysql.com/
[3] http://www.daml.org/2001/09/damldb/
[4] http://www.ontotext.com/owlim/
[5] http://openrdf.org/
[6] http://www.hpl.hp.com/semweb/

their wide acceptance and maturity. Both versions of AllegroGraph[7] came with their own query framework. A brief overview of the features and the capabilities of the paired triple-store technologies and query frameworks is presented in Table 1. Note that an enhanced version of DAML DB called "Parliament", has been released as a part of BBN's Asio tool suite for the Semantic Web[8].

Table 1. Triple-Store Feature Matrix

	Sesame + MySQL	Sesame + DAML DB	Sesame + SwiftOWLIM	Jena + MySQL	Jena + DAML DB	AllegroGraph 1	Sesame + BigOWLIM	AllegroGraph 2.0.1
Query language	SeRQL	SeRQL	SeRQL	SPARQL	SPARQL	SPARQL	SeRQL	SPARQL
Rule support	RDFS	Limited RDFS	RDFS + most of OWL Lite	RDFS + most of OWL Lite	Limited RDFS	RDFS + a bit of OWL Lite	RDFS + most of OWL Lite	RDFS + a bit of OWL Lite
owl:sameAs	No	Using Layered SAIL	Yes	Yes	No	Yes	Yes	Yes
owl:Inverse Functional Property	No	Yes	Yes	Yes	Yes	No	Yes	No
API Exten-sibility	Yes, SAIL	Yes, SAIL	Yes, SAIL	Yes, Jena	Yes, Jena	No	Yes, SAIL	No
Maturity	Mature	Mature	Mature	Mature	Mature	Mature	Beta	Beta
Price	Free	Free	Free	Free	Free	Not free	Unknown	Not free

We initially generated a data set consisting of 1500 universities written to independent OWL data files, which corresponds to roughly 200 million triples. We found that a number of the triple-store technologies (Sesame + BigOWLIM, Sesame + DAML DB and Jena + DAML DB) performed very well over this range of data, so we generated a second, larger data set to further test the performance of these triple-stores. The data-set used in this second round of testing consisted of 8000 universities written to independent OWL data files, which corresponds to roughly 1 billion triples. Due to time constraints, we were not able to evaluate the triple-stores over the entire larger data-set, but we were able to evaluate the selected triple-stores over most of this data. The developers of AllegroGraph 2.0.1 were unable to provide us a beta-test version of their technology during the first round of testing, but we included this triple-store in our second round of testing with the 8000-university dataset.

[7] http://www.franz.com/products/allegrograph/
[8] http://asio.bbn.com/parliament.html

2.2 Test Environment and Hardware

Our evaluation focused on centralized data-stores with a single user making queries. For this reason, we assumed that the test dataset, triple-store, and client software all reside on the same host and that queries are processed serially.

The test hardware used in the study was a Dell PowerEdge 2950 with two (2) Quad Core Intel Xeon E5345 Processors running at 2.33Ghz, a 1333MHz FSB, 16GB of RAM running at 533MHz and 6TB of storage disk storage. The test harness was written in Java and the tests were run using the Sun Java virtual machine, version 1.6.

We found that all of the triple-stores except Sesame + DAML DB required additional heap space to operate on datasets with more than a million triples. We tested multiple settings of the Java VM -Xmx and -Xms options to allocate additional heap space. Setting the virtual machine options of "-Xms 13000m -Xmx13000m", which allocates 13GB of heap to the virtual machine, provided all around good performance for the triple-stores. This large allocation of heap space could potentially be a hindrance to DAML DB which accesses RAM outside of the Java heap.

During initial testing we observed the Java virtual machine halting test execution due to the garbage collection operation exceeding its overhead limit. Through both testing rounds we turned off the garbage collection overhead limit using the "-XX:-UseGCOverheadLimit" JVM option. All triple-stores were then able to run without halting due to memory errors except for Sesame + SwiftOWLIM and AllegroGraphv1, which were able to run at least as long before halting due to an out-of-memory error.

2.3 Metrics

The individual metrics initially used by the LUBM were used as a starting point for the data collection in this evaluation study. Specifically, we collected data on:

- Cumulative Load Time: the time, measured in hours, to load the OWL files describing university departments into the triple-store for a given number of triples. This includes any time spent processing the ontology and source files.
- Query Response Time: query response time is calculated as the mean of the execution time for each of four identical queries.
- Query Completeness and Soundness: a triple-store is complete if it returns all of the correct responses to a query, while a triple-store is sound if it only returns correct responses to a query.
- Disk-Space Requirements: the amount of disk-space required to load the evaluation data. This data was collected only during the second round of testing for the triple-stores that were able to load and perform query operations on the large dataset.

The LUBM queries we used to measure the query response time were a low-volume, low-complexity query (LUBM Query 1), a high-volume, low-complexity query (LUBM Query 2), and a high complexity query (LUBM Query 9). Our descriptions of volume and complexity with respect to query types is taken from the LUBM documentation. A low-volume query is one where the number of query results is very small relative to the number of triples in the triple-store. Conversely, a high-volume query is one that returns a large portion of the stored triples in response to a query. A low-complexity query is one that requires very little processing power to complete, while a high-complexity query is one that requires substantial computing power to complete.

LUBM Query 1 asks for the number of GraduateStudents at a particular university that take a particular course. The correct response for this query over our test data is always 4. LUBM Query 14 requests the number of UndergraduateStudents in the knowledge base. A correct response for this query is a large fraction of the number of triples stored in the triple-store (on the order of 15%). LUBM Query 9 requests all students who take a course taught by their adviser. This query contains 5 atomic clauses that need to be addressed by the triple-stores' query engines. The correct response to this query is on the order of several thousand. Performance issues associated with low- and high-volume queries did not affect Query 9.

During initial testing, we noticed that the ordering of query clauses has a potentially large impact on the performance of the triple-stores. By ordering the queries such that fewer items in a data-set satisfy the initial clauses the query response time of queries can be improved upon otherwise identical queries with query clause reordered so that more elements of a data-set satisfy the reordered initial query clauses.

We attempted to optimize our query orderings as much as possible to minimize query response times during the test process while using the same query orderings for all triple-store evaluations. This was not necessary when using DAML DB because it reorders queries automatically, but for the sake of consistency we used the manually reordered queries for all triple stores.

Queries commonly contain explicit restrictions that can be inferred from the ontology. In such cases, a clever query optimizer may be able to eliminate the redundant statement from the query, thereby reducing the level of processing required to answer it. We suspect that none of the triple-stores in this study perform this query optimization. We did not attempt to remove these statements manually, but this might be an avenue for future study.

3 Study Results

During our operations of the triple-stores, we found that not all of the triple-stores we tested were complete. AllegroGraph 2.0.1 was able to correctly respond to all queries. Sesame + SwiftOWLIM, Sesame + BigOWLIM and all combinations of MySQL and DAML DB with Sesame and Jena were able to respond correctly to all queries except for Queries 10-13. The triple-stores were unable to infer the consequences of inverseOf and sameAs relationships. Two of the triple-stores (AllegroGraph 1 and Sesame + BigOWLIM) returned errors with Queries 14 and 13, respectively. Rather than demonstrating incompleteness, this most likely demonstrates that these triple-stores have bugs in their implementations that should be addressed.

During the first round of testing, Sesame + BigOWLIM appeared to be unsound for a large portion of the LUBM queries. During the second round of testing, we found that we could correct these errors by exclusively using SELECT DISTINCT clauses in the queries and not using WHERE clauses in the queries. Interestingly, SwiftOWLIM did not have these difficulties, so they were most likely due to BigOWLIM's beta-test status. These alterations in the queries used for Sesame + BigOWLIM were the

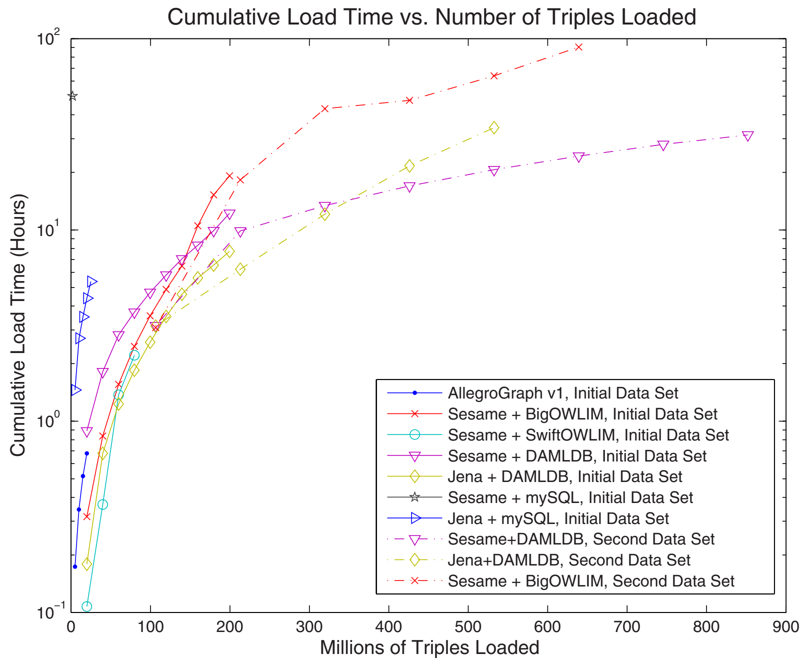

Fig. 1. Cumulative Load Time for Various Triple-Stores for Two Data Sets

only differences between queries in the triple-stores evaluations (except for translations between the SeRQL and SPARQL languages.)

During the first round of testing, Jena + DAML DB was incorrectly configured so that the full inferencing power of DAML DB was unused. This error was corrected during the second round of testing.

A graph of the cumulative load times vs. the number of triples loaded for the various triple-stores is shown in Figure 1. In the graph in Figure 1 and all other graphs in later figures, the results from operations with the initial 1500 university dataset during the first test round is plotted using solid lines and results from operations with the larger 8000 university dataset during the second test round is plotted using dashed lines.

Ideally, the cumulative load time of a triple-sore for this performance metric is linear in the number of triples loaded. (Note that a linear relationship appears to be sub-linear on a semi-log plot, which is what we used in all figures.) All of the triple-stores we tested had cumulative load times that grew linearly with the number of triples loaded.

As can be seen from the graph Jena + DAML DB and Sesame + DAML DB had the best cumulative load time performance from the triple-stores that we evaluated using the larger dataset. Over the initial dataset, Jena + MySQL and Sesame + MySQL exhibited substantially worse performance than the other triple-stores. Due to time constraints associated with their load times, we terminated the test runs of the Jena + MySQL and Sesame + MySQL triple-stores before completion. When provided 13 GB of heap space with 64-bit addressing, AllegroGraph 1 and Sesame + SwiftOWLIM were unable to load all of the triples used in the test run due to insufficient memory errors. AllegroGraph 1

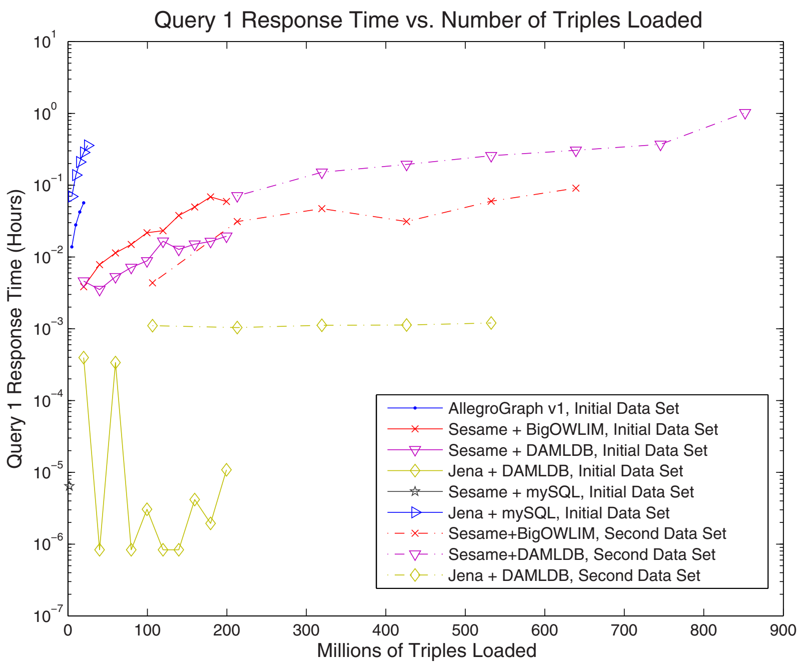

Fig. 2. Query 1 Response Times for Various Triple-Stores for Two Data Sets

was able to load 25 million triples, while Sesame + SwiftOWLIM was able to load just over 80 million triples. In order to achieve reasonable query performance in AllegroGraph 2.0.1, an indexAll operation needs to be performed after large sets of data are loaded but before query operations are issued. We found that after loading a large dataset, the indexAll operation was unable to complete (most likely due to a bug), so we were unable to obtain any meaningful performance results for AllegroGraph 2.0.1.

We also measured the amount of disk space the various triple stores required during the second testing round (see Table 2). In our evaluation, Sesame + BigOWLIM made more efficient use of disk space than Sesame + DAML DB and Jena + DAML DB.

Table 2. Disk Space Required per Million Triples Loaded

Triple-Store	Disk Space Required per Million Triples Loaded
Sesame + DAML DB	175 MB
Jena + DAML DB	167 MB
Sesame + BigOWLIM	53 MB

Figure 2 shows the Query 1 response performance of the triple-stores. Jena + DAML DB had very good performance and was able to return responses in a nearly trivial amount of time (in less than a second.) The accuracy of our measurements of query

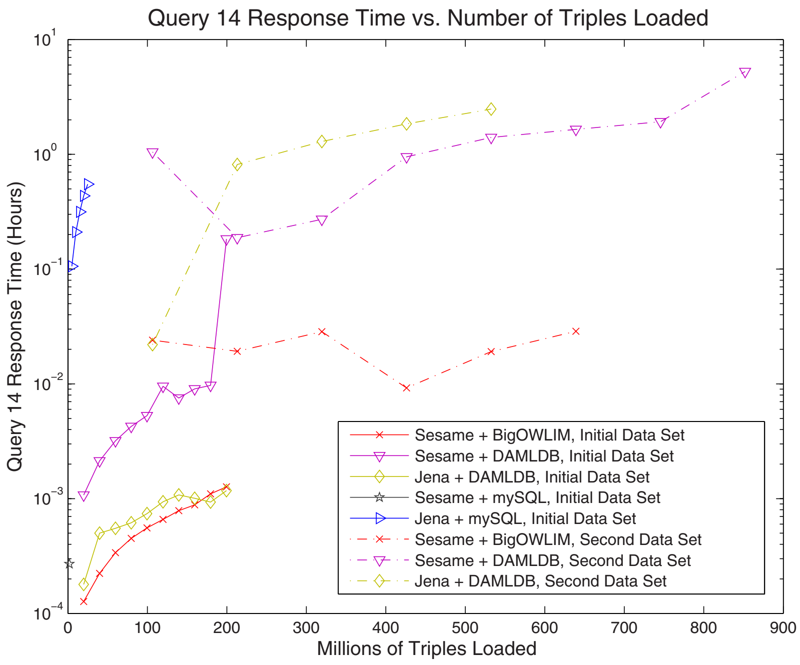

Fig. 3. Query 14 Response Times for Various Triple-Stores for Two Data Sets

response times was limited to 15ms due to the limitations of the system clock. This caused the query response times of Jena + DAML DB to appear to fluctuate wildly on the semi-log plot for the data collected during from the initial dataset. Out of the other triple-stores, Sesame + DAML DB and Sesame + BigOWLIM exhibited performance worse than Jena + DAML DB but comparable to one another.

Ideally, a triple-store should have a response time that does not increase, or increases very little as the number of triples loaded increases. We found that only Jena + DAML DB, Sesame + DAML DB and Sesame + BigOWLIM exhibit this behavior. The jump in Query 1 response time for Jena + DAML DB between rounds is most likely due to the increase in inferencing power for Jena + DAML DB between rounds.

Figure 3 shows the Query 14 response performance of the triple-stores. Sesame + BigOWLIM exhibited the best performers for this metric over both rounds of testing. The performance degradation of Jena + DAML DB between rounds is most likely due to the increase in inferencing power for Jena + DAML DB between rounds. Increased query response times for Jena + DAML DB and Sesame + DAML DB is most likely due to the limits of caching in DAML DB.

Figure 4 shows the Query 9 response performance of the triple-stores. Sesame + BigOWLIM had the best performance, and Sesame + DAML DB and Jena + DAML DB performed roughly an order of magnitude worse than Sesame + BigOWLIM. All of the triple-stores in this study had performance that degraded super-linearly with respect to the data repository size.

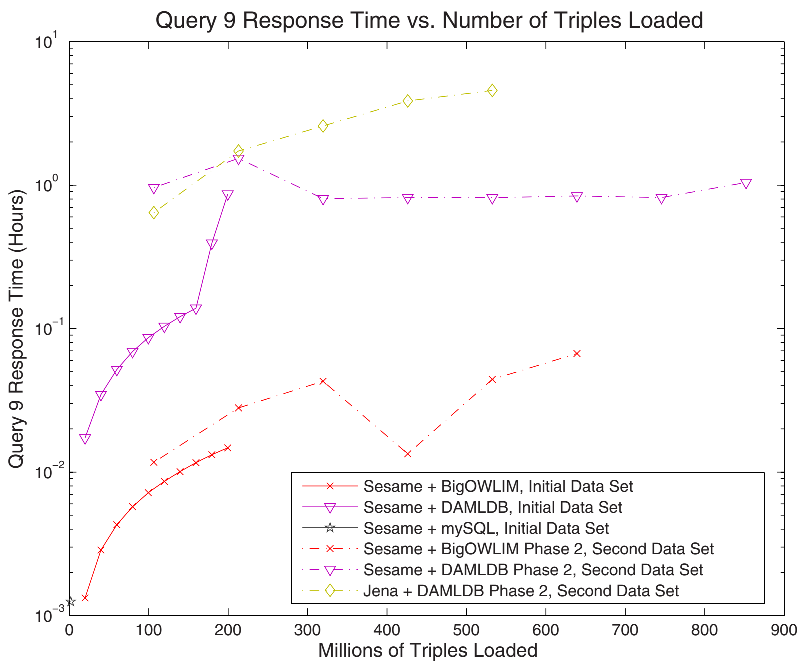

Fig. 4. Query 9 Response Times for Various Triple-Stores for Two Data Sets

4 Analysis and Areas for Future Work

Due to the observed limitations in the ability of Sesame + SwiftOWLIM to load very large datasets, we feel that this triple-store would not be a good performer as compared to the other triple-store technologies when evaluated over large datasets. Admittedly, this is a design limitation of SwiftOWLIM as it was designed to operate over smaller datasets. Due to the observed very long load times, we feel that AllegroGraph 1, Sesame + MySQL, and Jena + MySQL would not be appropriate for use with large datasets, even if load time were a secondary consideration for evaluation. If the indexAll operation in AllegroGraph 2.0.1 could be fixed, this technology could exhibit very good performance, but we were unable to adequately evaluate the query response times of this triple-store due to the non-termination of the indexAll operation.

We feel that the triple-stores that offered the best all-around performance for operations with a large dataset were Sesame + DAML DB, Jena + DAML DB, and Sesame + BigOWLIM. Each of these triple-stores has their own relative merits. Most importantly, all three of them provide adequate query response time performance for various queries, but no one triple-store is clearly better than the other triple-stores in all cases under the conditions evaluated in this study. For instance, Sesame + BigOWLIM provides better response time than the other triple-stores when responding to complex queries. Additionally, Jena + DAML DB provides the best performance for low-complexity queries, at least on the low-complexity queries that we tested. Sesame + DAML DB performs reasonably well, but not demonstrably the best in response to all queries. Additionally, these three triple-stores load data adequately fast.

One could expand this study by examining more advanced features such as reification and the effects of concurrent users on performance. Several of the triple-stores in this study support reification, but we did not make any measures of reified query performance. For similar reasons, it would be of interest to investigate how the actions of multiple concurrent users affect the performance of the triple-stores in a distributed deployment scenario. Finally, after this study concluded, the authors became aware of the University Ontology Benchmark (UOBM) [5], which studies triple-store inference capabilities in greater depth. This would be another useful direction for future work.

References

1. Guo, Y., Pan, Z., Heflin, J.: An evaluation of knowledge base systems for large OWL datasets. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 274–288. Springer, Heidelberg (2004)
2. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2), 158–182 (2005)
3. Lee, R.: Scalability report on triple store applications. Technical report, Massachusetts Institute of Technology (July 2004), `http://simile.mit.edu/reports/stores/`
4. Beckett, D.: SWAD-Europe deliverable 10.1: Scalability and storage: Survey of free software / open source RDF storage systems. Technical Report IST-2001-34732, EU (July 2002), `http://www.w3.org/2001/sw/Europe/reports/rdf_scalable_storage_report`
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.) ESWC 2006. LNCS, vol. 4011, pp. 125–139. Springer, Heidelberg (2006)