Evaluation of Current RDF Database Solutions

Florian Stegmaier[1], Udo Gröbner[1], Mario Döller[1], Harald Kosch[1] and Gero Baese[2]

[1] Chair of Distributed Information Systems
University of Passau
Passau, Germany
forename.surname@uni-passau.de
[2] Corporate Technology
Siemens AG
Munich, Germany
gero.baese@siemens.com

Abstract. Unstructured data (e.g., digital still images) is generated, distributed and stored worldwide at an ever increasing rate. In order to provide efficient annotation, storage and search capabilities among this data and XML based description formats, data stores and query languages have been introduced. As XML lacks on expressing semantic meanings and coherences, it has been enhanced by the Resource Description Format (RDF) and the associated query language SPARQL.
In this context, the paper evaluates currently existing RDF databases that support the SPARQL query language by the following means: general features such as details about software producer and license information, architectural comparison and efficiency comparison of the interpretation of SPARQL queries on a scalable test data set.

1 Introduction

The production of unstructured data especially in the multimedia domain is overwhelming. For instance, recent studies[3] report that 60% of today's mobile multimedia devices equipped with an image sensor, audio support and video playback have basic multimedia functionalities, almost nine out of ten in the year 2011. In this context, the annotation of unstructured data has become a necessity in order to increase retrieval efficiency during search. In the last couple of years, the Extensible Markup Language (XML) [16], due to its interoperability features, has become a de-facto standard as a basis for the use of description formats in various domains. In the case of multimedia, there are for instance the well known MPEG-7 [13] and Dublin Core [12] standards or in the domain of cultural heritage the Museumdat[4] and the Categories for the Description of Works of Art (CDWA) Lite[5] description formats. All these formats provide a

[3] http://www.multimediaintelligence.com
[4] http://museum.zib.de/museumdat/museumdat-v1.0.pdf
[5] http://www.getty.edu/research/conducting_research/standards/cdwa/cdwalite.html

XML Schema for annotation purposes. Related to this, several XML databases (e.g., Xindice[6]) and query languages (e.g., XPath 2.0 [2], XQuery [20]) have been introduced in order to improve storage and retrieval capabilities of XML instance documents.

The description based on XML Schema has its advantages in expressing structural and descriptive information. However, it lacks in expressing semantic coherences and semantic meaning within content descriptions. In order to close this gap, techniques emerging from the Semantic Web[7] have been introduced. The main contribution is RDF [19] and its quasi standard query language SPARQL [11]. Both, are recommendations of W3C[8], just as XML.

In this context, the paper provides an evaluation of currently existing RDF databases that support the SPARQL query language. The evaluation concentrates on general features such as details about software producer and license information as well as an architectural comparison and efficiency comparison of the interpretation of SPARQL queries on a scalable test data set.

The remainder of this paper is organized as follows: Section 2 covers some basic informations about accessing and evaluating RDF data. The definition of evaluation criteria is done in section 4. Section 5 provides an architectural overview of the triple stores in scope. Details about the test environment and the results of the performance tests are part of section 6. The paper is concluded in section 7.

2 Related work

This chapter covers basic information about related paradigms and technologies/standards required to perform the evaluation.

2.1 RDF data representation and storage approaches

Recent work already investigated several approaches concerning the storage of RDF data. In general, RDF data can be represented in different formats:

- *Notation 3 (N3)* [3] is a very complex language in order to store RDF-Triples, which was issued in 1998.
- *N-Triples* [17] was a recommendation of W3C, published in the year 2004. It is a subset of N3 in order to reduce its complexity.
- *Terse RDF Triple Language (Turtle)* [1] was invented in order to enlarge the expressiveness of N-Triples. The Turtle syntax is also used to define graph patterns in the query language SPARQL [8].
- *RDF/XML* [18] defines an XML syntax for representing RDF-Triples.

Three fundamental different storage approaches can be identified at present:

[6] `http://xml.apache.org/xindice/`
[7] `http://www.w3.org/2001/sw/`
[8] `http://www.w3.org`

– *in-memory storage* allocates a certain amount of the available main memory to store the given RDF data. Obviously this approach is intended to be used for few RDF data.
– *native storage* is a way to save RDF data permanently on the file system. These implementations may fall back on (in this terms) well investigated index structures, such as B-Tree.
– *relational database storage* makes use of relational database systems (e.g., PostgreSQL) to store RDF data permanently. Like the *native storage*, this approach relies on research results in the database domain (e.g., indices or efficient processing). Two different mapping strategies have been considered: The first is an universal table, which contains all RDF triples. The second solution is to create a mapping of the ontology into a table structure. Apparently, this leads to a (potentially) large number of tables.

2.2 RDF databases

An overview of frameworks and applications with the ability to store and to query RDF data is provided in Table 1. To retrieve the stored data, (quasi–) standards can be used, in names RDF Query Language (RQL) [10], RDF Data Query Language (RDQL) [15] and finally the W3C Recommendation SPARQL Protocol and RDF Query Language (SPARQL) [21]. A comparison of RDF query languages of the year 2004 can be found in [14].

2.3 RDF performance benchmarks

In addition to the huge efforts necessary to provide RDF database systems and defining query languages, appropriate evaluation methodologies[9] for triple stores have been introduced recently.

This section gives an overview of three promising performance benchmarks:

Berlin SPARQL Benchmark (BSBM)[10] [5] provides an benchmark using SPARQL. This benchmark includes a data generator and a test suite. The data generator is able to build a scalable amount of test data in RDF/XML format, which is based on an e-commerce use case. For example, a search for products from different suppliers can be performed or comments on the product can be provided. The mode of operation of the test suite is based on a use–case taken from real life. An automtic execution of miscellaneous queries is imitating the behavior of human operators.

Lehigh University Benchmark (LUBM)[11] [9] specifies the test data by an ontology named *Univ-Bench*. It represents an university with professors, students, courses and so on. The test data set can be constructed with the associated data generator [6]. The benchmark contains 14 test queries written in a KIF[12]–like

[9] `http://esw.w3.org/topic/RdfStoreBenchmarking`
[10] `http://www4.wiwiss.fu-berlin.de/bizer/BerlinSPARQLBenchmark/`
[11] `http://swat.cse.lehigh.edu/projects/lubm/`
[12] `http://www.csee.umbc.edu/kse/kif/`

Table 1. Overview of available RDF Triple Stores (abbreviations: o. = ongoing, disc. = discontinued, e.d.s. = early developing stage, u. = unknown)

Name	State	Programming language	Supported query language	Supported storage	Part of eval.	License
3Store	o.	C	SPARQL, RDQL	MySQL, Berkley DB	no	GPL
AllegroGraph	o.	Lisp	SPARQL	– (native disk storage)	yes	commercial
ARC	o.	PHP	SPARQL	MySQL	no	open source
BigOWLIM	o.	Java	SPARQL	– (plug-in of Sesame)	no	commercial
Bigdata	o.	Java	SPARQL	distributed databases	no	GPL
Boca	disc.	Java	SPARQL	relational databases	no	Eclipse Public License
Inkling	disc.	Java	SquishQL	relational databases	no	GPL
Jena	o.	Java	SPARQL, RDQL	in–memory, native disk storage, relational backends	yes	open source
Heart	e.d.s.	u.	u.	u.	no	u.
Kowari metastore	disc.	Java	SPARQL, RDQL, iTQL	native disk storage	no	Mozilla Public License
Mulgara	o.	Java	SPARQL, TQL & Jena bindings	integrated database	no	Open Software License v3.0
Open Anzo	o.	Java	SPARQL	relational database	yes	Eclipse Public License
Oracle's Semantic Technologies	o.	Java	SPARQL	relational database	yes	BSD-style license
RAP	o.	PHP	SPARQL, RDQL	in–memory, relational database	no	LGPL
rdfDB	o.	Perl	SQLish query language	Sleepycat Berkeley DB	no	open source
RDFStore	o.	Perl	SPARQL, RDQL	relational database	no	open source
Redland	o.	C	SPARQL, RDQL	relational databases	no	LGPL 2.1, GPL 2 or Apache 2
Semantics.Server 1.0	o.	.NET	SPARQL	MySQL	no	commercial
SemWeb – DotNet	o.	.NET	SPARQL	in–memory, relational database	no	GPL
Sesame	o.	Java	SPARQL, SeRQL	in–memory, native disk storage, relational database	yes	BSD-style license
Virtuoso	o.	Java	SPARQL	relational database	no	open source & commercial & open source
YARS	o.	Java	subset of N3	Berkeley DB	no	BSD-style license

language and a test suite called *UBT*, which manages the loading of data and the query execution automatically.

SP²B SPARQL Performance Benchmark (SP²B)[13] [7] benchmark consists of two major components. The first component is a (command line driven) data generator, which can automatically create the evaluation data. The amount of triples in this data set is scalable and based on the DBLP Computer Science Library[14]. In this case the data generator uses several well known ontologies, such as Friend of a Friend (FOAF)[15]. The second component consists of SPARQL queries, which are specifically designed for the DBLP use case.

3 Preselection of technologies in scope

This section provides the reasoning for the chosen databases and evaluation benchmark.

All technologies, which are discontinued or in a too early state of development, are excluded. As the development of Boca, Inkling, Kowari and RDFStore is discontinued and the Heart project is not yet implemented, a closer examination is not possible.

Furthermore, all databases shall have the ability to interpret SPARQL queries. As the overview in section 2.2 shows, rdfDB and YARS do not support SPARQL, these databases will not be part of the further evaluation.

Based on the evaluation in [7] the achieved evaluation of ARC, Redland and Virtuoso are insufficient, thus a further examination of these databases is not part of this paper. Our paper extends this previous work by highlighting architectural facets and general information of the tested databases (see section 4 for details). Furthermore, we collected yet available databases in table 1, which takes the current technologies and implementation efforts (e.g., Oracle's Semantic Technologies) into account. Schmidt et al. investigated in [7] the execution times for in–memory and native storage. In contrast to that, our evaluation is based on the relational storage approach.

The evaluation is based on SP²B, because it is most up–to–date and SPARQL specific. In order to use LUBM, a translation of the queries into SPARQL must be conducted, which is not satisfactory. Comparing the test data structure of BSBM to the data of SP²B, the SP²B data uses already well known ontologies, which is an additional advantage.

4 Evaluation criteria

The evaluation of RDF databases is based on three categories. The first category focuses on general information about the technologies:

[13] `http://dbis.informatik.uni-freiburg.de/index.php?project=SP2B`
[14] `http://www.informatik.uni-trier.de/~ley/db/`
[15] `http://www.foaf-project.org`

Software producer provides details about the company implementing the framework.

Associated licenses shed light on the usage of the frameworks, whether it can be used in business applications or not.

Project documentation should be rather complete. Furthermore, tutorials should be available supporting the work with these systems especially in the period of vocational adjustment.

Support is the last basic criteria. Support should be covered for example by an active forum or a newsgroup.

The aspects of the second category examine architectural facets of the considered frameworks, such as:

Extensibility is a very important criteria for the integration of new features, e.g., to optimize the existing working process. One of these features could be the implementation of new indices, which accelerate the performance and advance the efficiency of the entire system.

Architectural overview provides an insight into the structure of the framework and the used programming language.

OWL should be supported by the databases, because it enlarges the semantic expressiveness of RDF especially as far as reasoning is concerned.

Available query languages is another point of interest, is there support for other RDF addressing query languages in addition to SPARQL.

Interpretable RDF data formats are not part of central focus. The most important formats (as mentioned in section 2.1) should be covered by the frameworks from the point of completeness.

The evaluation of these two categories can be found in Chapter 5.

The third category is based on the expressiveness of SPARQL queries and the performance of the frameworks / applications. SPARQL consists of four different query forms: *SELECT*, *ASK*, *CONSTRUCT* and *DESCRIBE*. This evaluation is restricted to the *SELECT* query type. It is discussed in Chapter 6. Further details about the test environment are provided there, too.

5 Evaluation of considered databases

This section covers the evaluation of AllegroGraph, Jena, Open Anzo, Oracle's Semantic Technologies and Sesame following the reasoning in section 3.

5.1 AllegroGraph

The *software producer* of AllegroGraph RDF Store[16] is Franz Inc.[17]. The company has been founded in 1984 and is well known for its Lisp programming

[16] `http://www.franz.com/agraph/allegrograph/`

[17] `http://franz.com/`

language expertise. Recently, they also started developing semantic tools, like AllegroGraph.

The *associated licenses* of AllegroGraph come in two different flavors. The version evaluated in this paper is the free edition, which is limited to 50 million triples maximum. In contrast to that, the enterprise version has no limits regarding to the number of stored triples but underlies a commercial license.

The *product documentation* delivered by Franz Inc. is rather complete. Several useful example Java classes can be found on the companies website alongside the Javadoc of the Java binding.

Support for AllegroGraph is offered by Franz Inc. in a commercial way. In detail, they offer training for the software, seminars and consulting services, which also includes application-specific coding if needed.

AllegroGraph is not *extensible*. It is closed source and stores data as well as the database indices inside its particular storage stack.

Because of its closed source, an *architectural overview* is not possible. Therefore, figure 1 shows a client server architecture of AllegroGraph. The software is developed especially for 64 Bit systems and runs out of the box, as it doesn't need any other databases or software. Storage, indexing and query processing is performed inside AllegroGraph. The software can be accessed using Java, C#, Python or Lisp. There are bindings for Sesame or Jena integration available and also an option to access AllegroGraph via HTTP.

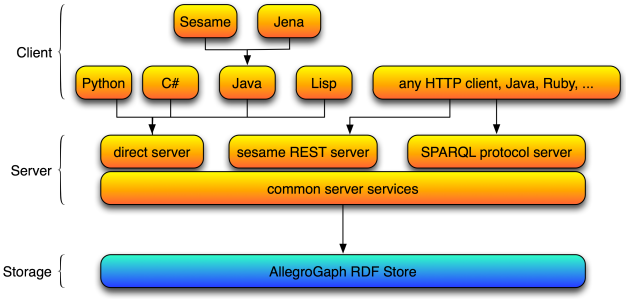

Fig. 1. AllegroGraph client server architecture

Franz Inc. suggests using TopBraid Composer[18] by TopQuadrant Inc. for *OWL support*.

The *available query language* of the software is SPARQL, but it also supports low level API calls for direct access to triples by subject, predicate and object. With those API calls, it is possible to retrieve all datasets matching a certain triple. The API calls provide functionality, which can be compared to SQL SELECT statements.

[18] `http://www.topquadrant.com/topbraid/composer/index.html`

The *interpretable RDF data formats* of AllegroGraph are RDF/XML and N-Triples. Other formats are planned to be supported in future versions.

5.2 Jena

The *software producers* of Jena[19] are the HP Labs[20], which are a part of the Hewlett-Packard Development Company. This department was founded in 1966 by Bill Hewlett and Dave Packard. Jena was developed in the terms of the HP Labs Semantic Web Research.

The *associated license* of the Jena project is completely open source. This implies that redistribution and use in source and binary forms with or without modification are permitted[21].

The Jena *product documentation* can be found on the project page and is widely complete. The documentation covers the central parts of Jena providing basic information about the framework, Javadocs and several tutorials respectively HowTos. The downloadable version of Jena also includes code examples, which underline the basic steps in the working process of Jena.

The *support* focuses on a newsgroup[22], which is founded in the Yahoo! Groups[23]. It may be considered unsatisfactory that support is primarily limited to a newsgroup. But due to the fact that there is a large amount of registered members[24] the activity of the newsgroup and therefore the delivered support is excellent.

The Jena download package includes the source files of the entire Jena project implemented in Java. This provides a basis for implementations *extending* the framework, for instance with new indices.

Figure 2 illustrates an *architectural overview* of Jena. The framework offers methods to load RDF data into a memory based triple store, a native storage or into a persistent triple store. In order to build a persistent triple store a variety of relational databases, for example MySQL, PostgreSQL or Oracle, can be used. The stored data may be retrieved through SPARQL queries. A standard implementation of the SPARQL query language is encapsulated in the ARQ package of Jena. SPARQL queries can be executed using Java applications or by the use of the graphical frontend Joseki[25]. The Ontology API provides methods to work on ontologies of different formats, like OWL or RDFS. Jena's Core RDF Model API offers methods to create, manipulate, navigate, read, write or query RDF data. The remaining major components are on the one hand the Inference API, which allows the integration of inference engines or reasoners into the system. On the other hand the Reification API is a proposal to optimize the representation of reification.

[19] `http://jena.sourceforge.net/`
[20] `http://www.hpl.hp.com/`
[21] `http://jena.sourceforge.net/license.html`
[22] `http://tech.groups.yahoo.com/group/jena-dev/`
[23] `http://groups.yahoo.com/`
[24] Members of the Jena newsgroup (at time of writing): 2752
[25] `http://www.joseki.org/`

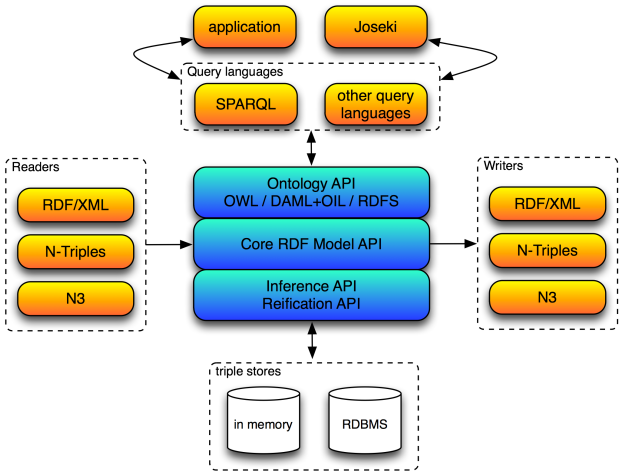

Fig. 2. Architectural overview of Jena

OWL support is given in form of the Ontology API. The inference subsystem[26] enables the use of inference engines or reasoners in Jena.

Besides SPARQL, RDQL is a *supported query language*. In a tutorial about RDQL it is recommended that new users of Jena should use SPARQL instead.

Jena uses readers and writers for RDF/XML, N-Triples and N3, which are commonly known *RDF data formats*.

5.3 Open Anzo

Open Anzo[27] is the prosecution of Boca[28] and other components produced by the IBM Semantic Layered Research Platform[29].

The Open Anzo project offers a good *product documentation*. The key topics are architectural facets of the current version, programmer guides and design documents. There are also documents available describing key features of an upcoming version of Open Anzo.

The *support* is based on several tutorials and a Google group[30] with about 63 members at time of writing.

As already mentioned, Open Anzo is complete open source, underlying the Eclipse Public License. So it is possible to *extend* the given framework by needed functionalities.

[26] `http://jena.sourceforge.net/inference/`
[27] `http://www.openanzo.org/`
[28] `http://ibm-slrp.sourceforge.net/`
[29] `http://ibm-slrp.sourceforge.net/`
[30] `http://groups.google.com/group/openanzo`

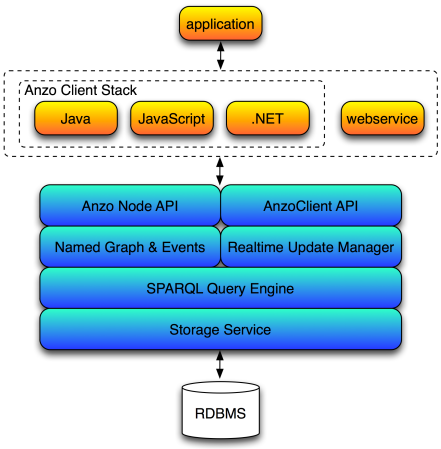

Fig. 3. Architectural overview of Open Anzo

Figure 3 highlights the main components of the Open Anzo *architecture*. Open Anzo can be used with three modes of operation. It is possible to embed it in an application, run it as a remote server or use it locally. The entry points to the framework are the Anzo Client Stack (offers API implementations in Java, Javascript and .NET) or a webservice. The Anzo Node API is the basis to describe the structure of RDF data. The named graph component enables user to access the RDF data. Beside that, the AnzoClient API encapsulates transaction preconditions and connectivity events to the database. The purpose of the Realtime Update Manager is to deliver messages about certain processing states. In order to execute SPARQL queries in Open Anzo, the SPARQL Query API is needed. The Storage Service is used to save and retrieve RDF data using a relational database (like DB2 or Oracle). This is the center of any mode of operation in an Open Anzo system.

There are OWL related classes in the project, but further information is missing in the documentation regarding the coverage of *OWL* functionalities. The producers claim on the product page that other semantic web technologies (3^{rd} party components) could easily be plugged into the system.

Open Anzo supports SPARQL queries and typed full-text search capabilities, which also use an index system in order to improve the retrieval process.

N3, N-Triples, RDF/XML and TriX[31] are the supported *RDF data formats*.

[31] `http://www.w3.org/2004/03/trix/`

5.4 Oracle's Semantic Technologies

Software producer Oracle[32] is one of the major players in database business. The company comprises relational database knowledge of 30 years and has added support for semantic technologies to its products lately. The evaluated Semantic add-on is the Jena Adapter 2.0 for Oracle Databases. It implements the Jena Graph and model APIs as described earlier. The add-on requires Oracle Database 11g Release 11.1.0.6 (or higher) or Oracle Database 10g Release 10.20.0.1 or 10.2.0.3.

Licensing options can be found at the Oracle page[33]. The Jena Adapter is provided from Oracle for free as closed source.

Product documentation can be found at Oracle Semantic Technologies Center[34] and offers code samples, usage scenarios, training material and documentation for administrators as well as developers. The documentation provides a good overview, but the structure of the website could be improved for usability reasons.

Support is available via the Oracle forum[35] for free, with excellent answer times. Paid support is also available from several partners[36] and from Oracle itself.

An overview of the semantic capabilities of Oracle's add–ons is illustrated in figure 4.

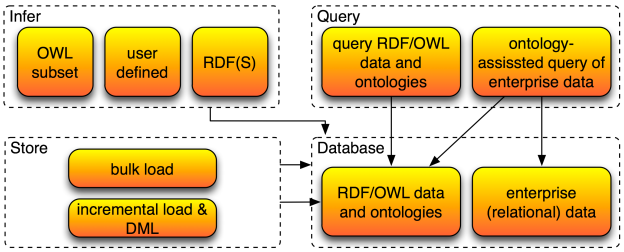

Fig. 4. Oracle's Semantic Technologies capabilities

Oracle supports large graphs of billions of triples, which can be queried by SPARQL-like syntax and/or SQL. Complete SPARQL support is at the time of this writing only available via the Jena adapter but native support for SPARQL is planned. The RDF data model includes capabilities for inference via RDFS, its

[32] http://www.oracle.com
[33] http://www.oracle.com/us/corporate/pricing/index.htm
[34] http://www.oracle.com/technology/tech/semantic_technologies/index.html
[35] http://forums.oracle.com/forums/forum.jspa?forumID=269
[36] http://www.oracle.com/technology/tech/semantic_technologies/htdocs/semtech_partners.html

subset RDFS++, OWL, its subsets OWLSIF and OWLPrime, and user–defined rules.

RDF data formats are RDF/XML, N-Triples and N3 because Jena is being utilized. Semantic data can also be compressed by using the advanced compression option to reduce needed disk space.

5.5 Sesame

The *software producer* of Sesame[37] is Aduna[38]. This company sets the focus of their work in revealing the meaning of information. Sesame was started as a prototype of the EU project On-To-Knowledge[39] and is now developed by Aduna in a cooperation with NLNet Foundation[40].

Like Jena, Sesames *associated license* is open source underlying the BSD-style license.

The *product documentation* of Sesame is well organized. There is a large user guide available for every version of Sesame. Users can also refer to Javadocs and tutorials completed with example code.

Aduna provides *support* in form of an active forum accessible on the project page and a mailing list based on SourceForge[41]. Commercial consulting services are also provided.

Sesame's download package is shipped with the Java source files. Therefore, a basis for *extending* the framework is provided similar to Jena.

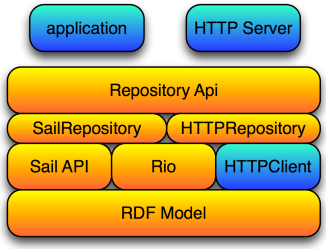

Fig. 5. Architectural overview of Sesame

Figure 5 shows an *architectural overview* of Sesame. In order to use Sesame, Apache Tomcat is recommended. The Sesame package also contains two web applications, the Sesame server which stores the RDF data and the OpenRDF

[37] `http://www.openrdf.org/`
[38] `http://www.aduna-software.com/`
[39] `http://www.ontoknowledge.org/`
[40] `http://www.nlnet.nl/`
[41] `http://www.sourceforge.net`

Workbench as a graphical frontend for the server. This workbench can manage repositories, load RDF data and execute queries. Sesame is able to handle all three in section 2.1 discussed approaches to store RDF data. The RDF Model implements basic concepts about RDF data. The component RDF I/O (Rio) consists of a set of parser and writer for the handling of RDF data. This is for instance used by the Storage And Inference Layer (Sail) API for initializing, querying, modifying and the shut down of RDF stores. On the topmost layer constitutes the Repository API the main entrance to address repositories. Compared to Sail, which is rather a low level API, the Repository API is the associated high level API with a larger amount of methods for managing RDF data. The HTTPRepository is an implementation that acts like a proxy in order to connect to a remote Sesame server via the HTTP protocol.

In order to achieve *OWL support* a Plug-In is available called BigOWLIM[42]. It is implemented as a high performance semantic repository for Sesame and packaged as a Sail.

Alternatively to SPARQL Sesame is able to interpret the Sesame RDF Query Language (SeRQL) [4] integrated for enhancing the functionality of RQL and RDQL.

Sesame offers parsers for various well known *RDF formats* N3, N-Triples, RDF/XML, Turtle and two new formats TriG[43] and TriX.

6 Performance tests

The performance tests of AllegroGraph 3.3.1, Jena (SDB 1.1), Open Anzo 3.1.0, Oracle's Semantic Technologies (Jena Adapter v.2.0)and Sesame 2.2.4 are conducted in the following test environment. It consists of a client and a server connected over a 1 Gb LAN network. The main parts of the server are two Intel Xeon 3,8GHz Single-Core CPUs, 6 GB RAM and two 136GB Ultra320-SCSI HDDs in a Hardware-RAID-1 with a Ubuntu 8.04.1 operating system running on top. The client is a MacBook Pro with a 2,4 GHz Intel Core 2 Duo CPU, 2 GB Ram and a 150 GB Fujitsu HDD and the Mac OS 10.5.7 operating system. In order to create persistent triple stores in Jena and Sesame, PostgreSQL is used. All performance tests are conducted with the standard configurations of the frameworks and database backends.

The queries of the SP^{2}B benchmark can be classified into two groups according to the expected complexity. On the one hand *FILTER*, *OPTIONAL* and *UNION* are very similar to well known SQL paradigms (*SELECT*, left outer joins, relational *UNION*). Only minor influence on the performance of query execution is assumed, because efficient implementations can be used [7]. On the other hand keywords like *DISTINCT*, *LIMIT* or *OFFSET* will seriously affect the query execution [7] (*pipeline breaker*). The queries will indicate the correctness of this assumptions, as they insist on at least one of the keywords or a combination of them. The graph structure, which will be build by the queries can

[42] `http://ontotext.com/owlim/big/`
[43] `http://www4.wiwiss.fu-berlin.de/bizer/TriG/`

be distinguished into long path chains[44], bushy patterns[45] or the combination of these two structure types.

The evaluation data was created in the N3 data format with the SP^{2}B data generator. A data set with about 100.000 triples (10.3 MB) another with 1.000.000 triples (107 MB) and a last one with 5.000.000 triples (538 MB) have been created. In order to import the N3 data into AllegroGraph, CWM[46] has been used to parse the N3 data into RDF/XML, which AllegroGraph is able to process. The parser was not able to parse the dataset with 5.000.000 triples. Therefore, this data set could not be tested with AllegroGraph.

The following part shows the results of the evaluation focusing on the query execution time. This time only includes the query execution and the transfer of the result set from the server to the client (opening and closing of the connection to the repository not included). The time unit given in the figure 6 are milliseconds. A value of 1.000.000 milliseconds indicates a timeout of the query.

The execution times clearly show a great difference in the query execution between Jena, Open Anzo, Oracle's Semantic Technologies, Sesame and AllegroGraph and are similar to the execution times achieved in [7] for in–memory and native storage. For instance the execution of query 4 regarding the 100.000 triple test set lasts 28 milliseconds in Jena and 18 milliseconds in Oracle. In contrast, this query on the same test set took 14478 milliseconds in Sesame, 141155 milliseconds in Open Anzo and 176496 milliseconds in AllegroGraph. There are also queries, where Sesame's execution times are smaller than Jena's or Oracles, for example Query 1 and 2 (also in the two bigger data sets). A reason for this behavior comparing Jena, Oracle and Sesame is the diverse import strategy of these two frameworks. The import of data in Sesame leads to the creation of 69 tables for the 100.000 triples data set, 79 tables for the 1.000.000 triples data set and 87 tables for the 5.000.000 triples data set. Jena creates constantly 4 tables (universal table approach as discussed in section 2.1). Oracle's Semantic Technologies is using the Jena framework, the storage approach is the identical. Sesame performs a mapping of the different entities in the N3 data sets directly into tables of the database while building several other tables to save the RDF triples data. Jena doesn't use a mapping like this. Obviously, queries consisting of a great amount of *dots*[47] increase the execution time on a database with about 70 tables compared to a database with only 4 tables. The other way round Sesame is able to minimize the number of cross products during query execution because it is able to address the elements of a special entity saved in a particular table. AllegroGraph is saving the triples directly on the hard disk. It creates one data file containing the RDF data and several other files, which purpose is not deducible. Although AllegroGraph uses some kind of indices on the repository the execution lasts much longer than in the other frameworks.

[44] Similar to joins over a few tables in a relational database.
[45] For example queries on a Star Schema
[46] `http://www.w3.org/2000/10/swap/doc/cwm.html`
[47] *dots* are similar to joins.

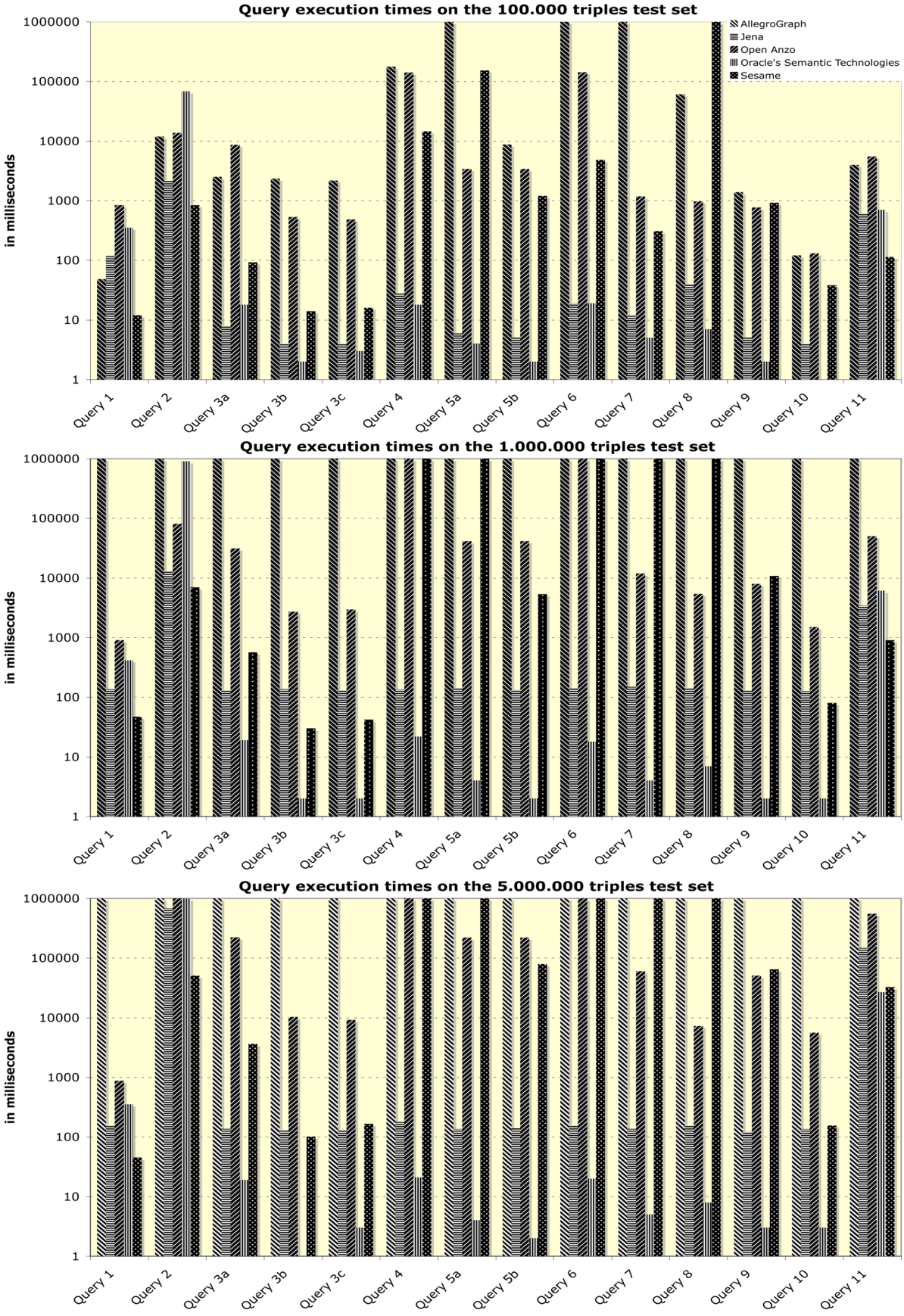

Fig. 6. Query execution times on the three different test sets

Figure 6 also shows the results of the evaluation for the 1.000.000 triples data set and for the 5.000.000 triples data set. The execution time of AllegroGraph was exceeding the time limit (terminated after 30 minutes per query) for the 1.000.000 triples data set. There is also an ascent of the execution times and timeouts observable for the other triple stores.

7 Conclusion & Outlook

The architectural overview of chapter 5 and the performance tests of chapter 6 shows that AllegroGraph is not fulfilling the criteria defined in chapter 4. It is neither extensible nor are the execution times satisfying. Jena and Sesame are both API extensible but Jena obtained continuous evaluation times at the moment. Oracle's Semantic Technologies is using the Jena framework but it comes with database procedures, which have an impact on the performance. In contrast to that, Open Anzo serves well for small data but is not very good in handling big amounts of RDF data. Jena and Oracle Semantics Technologies are fulfilling the chosen criteria best. However, a decision to use one or the other framework must be based on the domain to be addressed by such a system and on the query structure expected. A deeper analysis of these two factors helps finding the answer, what kind of storage approach would be appropriate.

This paper, especially section 2.2 shows that huge efforts were done in the field of accessing RDF data. This trend is still ongoing as the development of new RDF triple stores (e.g., HEART) is indicating. Up to now, only relational databases or XML databases are in scope of these technologies. Only one database, namely Bigdata, is able to operate on a distributed database. Enlarging the set of accessible backends may improve the performance issues of certain query paradigms in a good way. Future work could focus on the mapping of SPARQL to SQL. Here, already well known database techniques could seriously enhance the processing of queries.

8 Acknowledgments

This work has been supported in part by the THESEUS Program, which is funded by the German Federal Ministry of Economics and Technology.

References

1. David Beckett. Turtle - terse rdf triple language. http://www.dajobe.org/2004/01/turtle/, November 2007.
2. Anders Berglund, Scott Boag, Don Chamberlin, Mary F. Fernandez, Michael Kay, Jonathan Robie, and Jerome Simeon. XML Path Language (XPath) 2.0. *W3C Recommendation, http://www.w3.org/TR/xpath20/*, 2007.
3. Tim Berners-Lee. Notation 3. http://www.w3.org/DesignIssues/Notation3, March 2006.

4. Jeen Broekstra and Arjohn Kampman. SeRQL: A Second Generation RDF Query Language. http://www.w3.org/2001/sw/Europe/events/20031113-storage/positions/aduna.pdf, November 2003.
5. Christian Bizer et al. Benchmarking the performance of storage systems that expose sparql endpoints. In *Proceedings of the 4th International Workshop on Scalable Semantic Web knowledge Base Systems (SSWS2008)*, 2008.
6. Kurt Rohloff et al. An evaluation of triple-store technologies for large data stores. In Robert Meersman et al., editor, *OTM Workshops (2)*, volume 4806 of *Lecture Notes in Computer Science*, pages 1105–1114. Springer, 2007.
7. Michael Schmidt et al. SP2Bench: A SPARQL Performance Benchmark. *CoRR*, abs/0806.4627, 2008.
8. Pascal Hitzler et al. *Semantic Web*. Springer, 2008.
9. Yuanbo Guo et al. Lubm: A benchmark for owl knowledge base systems. *J. Web Sem.*, 3(2–3):158–182, 2005.
10. Gregory Karvounarakis, Sofia Alexaki, Vassilis Christophides, Dimitris Plexousakis and Michel Scholl. RQL: a declarative query language for RDF. In *WWW*, pages 592–603, 2002.
11. Eric Prud hommeaux and Andy Seaborne. SPARQL Query Language for RDF. *W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/*, 2008.
12. Dublin Core Metadata Initiative. Dublin core metadata element set - version 1.1: Reference description. *http://dublincore.org/documents/dces/*, 2008.
13. J. M. Martinez, R. Koenen, and F. Pereira. MPEG-7. *IEEE Multimedia*, 9(2):78–87, April-June 2002.
14. Peter Haase, Jeen Broekstra, Andreas Eberhart and Raphael Volz. A Comparison of RDF Query Languages. In *International Semantic Web Conference*, volume 3298, pages 502–517, 2004.
15. Andy Seaborne. RDQL - A Query Language for RDF. http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/, January 2004.
16. W3C. Extensible Markup Language (XML) 1.1, W3C Recommendation. *http://www.w3.org/XML/*, February 2004.
17. W3C. Rdf test cases. http://www.w3.org/TR/rdf-testcases/, February 2004.
18. W3C. RDF/XML Syntax Specification (Revised). http://www.w3.org/TR/rdf-syntax-grammar/, February 2004.
19. W3C. Resource Description Framework (RDF). *http://www.w3.org/RDF/*, 2004.
20. W3C. XQuery 1.0: An XML Query Language. *W3C, http://www.w3.org/TR/2007/REC-xquery-20070123/*, 2007.
21. W3C. SPARQL Query Language for RDF. http://www.w3.org/TR/rdf-sparql-query/, January 2008.