Dominik Tomaszuk
Institute of Computer Science
University of Bialystok

DOCUMENT-ORIENTED TRIPLESTORE BASED ON RDF/JSON

Abstract: This paper defines dokument-oriented database as RDF triplestore. It proposes an alternative mean of serializing RDF triples using JavaScript Object Notation (JSON), a lightweight representation format which emphasizes legibility and brevity. This paper proposes declarative SPARQL mapping method to object-oriented imperative query language. This query language uses RDF/JSON syntax. It means that the dokument-oriented database, such as MongoDB, could be a triplestore.

Keywords: Semantic Web, Resource Description Framework (RDF), knowledge representation, triplestores, document-orientet databases, object-oriented programming (OOP).

1. Introduction

A simplified view of the Semantic Web is a collection of web retrievable RDF documents, each containing a Resource Description Framework (RDF) graphs. RDF has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources. In other words, RDF is a general-purpose language for representing information on the Web. RDF is designed to model and store information. It is also developed to provide interoperability between applications that exchange machine-understandable information on the Web.

RDF Recommendations [1, 2, 3] explain the meaning of subject, predicate and object. These expressions are known as triples in RDF terminology. The subject denotes the resource. The predicate means traits or aspects of the resource and expresses a relationship between the subject and the object. A collection of RDF statements intrinsically represents the labeled, directed multi-graph (figure 1). The nodes of an RDF graph are its subjects and objects.

More formally, let U to be the set of all URI references, B an infinite set of blank nodes, L the set RDF plain literals, and D the set of all RDF

ISBN 978–83–7431–273–8 ISSN 0860-150X

Fig. 1. RDF graph data model

typed literals. All the four sets are defined in [1]. U, B and L are pairwise disjoint. Let $O = U \cup B \cup L \cup D$ and $S = U \cup B$, then $T \subset S \times U \times O$ is set of all RDF triples[1].

A data model is an abstract model that describes how data is represented and accessed. Data models formally define data elements and relationships among data elements for a domain of interest. A data model explicitly determines the structure of data or structured data. Typical applications of data models include database models. A database model or database schema is the structure or format of a database, described in a formal language supported by the database management system. In other words, a database model is the application of a data model when used in conjunction with a database management system.

This paper studies the RDF model from a database perspective. From this point of view it is compared with other database models, in particular with graph database models, which are very close to RDF approach. A graph database uses nodes, edges and properties to represent and store information. This model is an alternative to relational databases or other structured storage systems based on columns.

In this paper a new document-oriented triplestore based on RDF/JSON is presented. Section 2 provides a related work on triplestores. In the next section document-oriented, semi-structured, JSON and key-value databases are presented. In section 4 three mapping SPARQL to Mongo Query Language algorithms are proposed. In the next section results of the experiments are presented. Conclusions end the paper.

2. Related work

A triplestore is a purpose-built database for the storage and retrieval of RDF data. A triplestore is optimized for the storage and retrieval of many triples. Much like a relational database, one stores information in a triple-

[1] The RDF core Working Group noted that it is aware of no reason why literals should not be subjects and a future WG with a less restrictive charter may extend the syntaxes to allow literals as the subjects of statements.

store and retrieves it via a query language. A difficulty with implementing triplestores over SQL is that triples may be stored and implemented efficiently by querying of a graph-based RDF model. One of the query languages for the RDF model is SPARQL [4]. Mapping from SPARQL onto SQL queries is difficult. There are a lot of triplestores that use relational database to store RDF triples.

Jena [5] is a Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract model. A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL. Jena supports serialisation of RDF graphs to a relational database, RDF/XML [6], Turtle [7] and Notation3 [8]. Joseki is a part of Jena. It is an RDF server. The goal of Joseki is to provide an HTTP interface to RDF data. It fully supports SPARQL for querying. Joseki can also be run a stand-alone server.

Sesame [9] is a framework for querying and analyzing RDF data. It contains a triplestore. One of the primary components of Sesame is the RDF query language SeRQL. Sesane can be sourced with data from memory, native store that stores and retrieves its data directly to/from the disk and relational database.

Another framework is RDF API for PHP (RAP) [10]. RAP includes in-memory and database model storage. The MemModel implementation stores statements in an array in the system memory. The DbModel implementation stores statements in an relational database. RAP implements the SPARQL query.

4Store [11] is an example of a triplestore that is a storage and query engine. It holds RDF data. It does not provide many features over and above RDF storage and SPARQL queries.

Mulgara [12] is another triplestore. It is massively scalable, and transaction-safe. Mulgara instances can be queried via the iTQL query language and the SPARQL query language. Mulgara is not based on a relational database due to the large numbers of table joins encountered by relational systems when dealing with metadata. Instead, Mulgara is a completely new database optimized for metadata management. Mulgara models hold metadata in the RDF triples. Metadata may be imported into or exported from Mulgara in RDF or Notation 3 form [8].

JRDF [13] is an open source RDF framework for Java. It provides an object oriented model of RDF graphs. JRDF API allow RDF to be created and written using various formats such as RDF/XML [6] and Notation3 [8]. It has an implementation of SPARQL [4]. By default JRDF uses in memory

store. It also can use a disk based store through various persistence layers. JRDF also implement a triple store across Hadoop using a technique called RDF molecules [14].

AllegroGraph [15] is another example of a triplestore. In contrast with a relational database, a AllegroGraph considers each stored item to have any number of relationships. It is designed to store RDF tuples. It implements the SPARQL query. AllegroGraph does not use relational database. It loads triples through RDF/XML [6], N-Quads[2] and N-Triples [16].

3. Document-oriented databases

As opposed to relational databases, document-based databases do not store data in tables with uniform sized fields for each record. Instead, each record is stored as a document that has certain characteristics. Any number of fields of any length can be added to a document. Fields can also contain multiple pieces of data.

Unlike a relational database where each record would have the same set of fields and unused fields might be kept empty, in document-oriented database there are no empty fields in either record. Document-oriented database allows information to be added any time without wasting storage space for empty fields as in relational databases. For these cases, JavaScript Object Notation (JSON) is often used.

JSON is a lightweight data-interchange format. It is effortless for humans to read and write. It is easy for machines to parse and generate. JSON is based on a subset of the [17]. JSON is a text format that is completely independent from programming languages. These properties make JSON a perfect data-interchange language.

JSON is built on two structures: a collection of name/value pairs and an ordered list of values. In most languages, collection of name-value pairs is realized as an object, struct, record, hash table, associative array, dictionary or keyed list. In various languages, ordered list of values is realized as an array, list, vector or sequence. JSON's basic types are numbers, strings, booleans, arrays, object and null. Number in JSON is integer, real, or floating point[3]. The string is double-quoted Unicode with backslash escaping. The boolean contains true or false value. The array is an ordered sequence

[2] N-Triples with context.

[3] The octal and hexadecimal formats are not used.

of values, comma-separated and enclosed in square brackets. The object is collection of key:value pairs, comma-separated and enclosed in curly brackets. Null contains empty value [18]. Main advantage of JSON is that it translates directly into universal data structures.

In JSON each record can have a non-standard amount of information. Such information is properly called semi-structured data. Semi-structured data is a form of data model that does not conform with the formal structure of tables and data models associated with databases but contains nonetheless tags or other markers to separate semantic elements and hierarchies of records and fields within the data. In this model, there is no separation between the data and the schema, and the amount of structure used depends on the purpose. The advantage of this model is that it provides a flexible format for data exchange between different types of systems and the data transfer format may be portable [19].

Most of document-oriented databases are also key-value databases. The key-value database characterized by the fact that each key is associated with one value or set of values. The relationship between a key and its value is sometimes called a mapping or binding. It is closely related to the mathematical concept of a function with a finite domain. JSON helps store key-value pairs in a formatted way. A document can have any number of key-value pairs. Instead of using a schema, documents of the same time all have a similar set of key-value pairs.

4. MongoDB as a triplestore

Document-oriented key-value stores such as MongoDB [20] are different than triple stores. The key-value store consists of two terms: keys and values, the triple stores consists of three terms: subjects, predicates and objects. It is difficult to map a key-value pairs to RDF triples.

It is a proposed method for mapping an RDF graph to the structure of JSON. Such syntax, being the equivalent to RDF model, would be more legible and brief. It describes JSON structure for RDF graph that expresses the whole RDF model that does not lose information.

4.1. MongoDB overview

MongoDB [20] is a document-oriented database written in the C++ programming language. The goal of MongoDB is to bridge the gap between key-value stores and traditional relational databases. MongoDB manages collections of JSON-like documents. This allows many applications to model

data in a more natural way, as data can be nested in complex hierarchies and still query-able.

In MongoDB all strings are UTF-8. Non-UTF-8 data can be saved, queried, and retrieved with a special binary data type – BSON [21]. This is nominally a superset of JSON types. BSON types include string, integer, double, date, byte array[4], boolean, null and BSON object. It is a binary format in which zero or more key/value pairs are stored as a single document.

BSON documents consist of a well ordered list of elements. Each element consists of a field name, a type and a value. Field names are strings. Compared to JSON, BSON is efficient both in storage space and scan-speed. Large elements in a BSON document are prefixed with a length field to facilitate scanning.

MongoDB has object query language. Main methods are:

- find([query], [fields]) – returns a result set of records. It can also contain statement should only affect records that meet specified criteria,
- limit() and skip() – restrict the range of results,
- distinct() – retrieves only unique data entries,
- count() and size() – returns the number of records with and without skip() and limit(),
- insert() and update() – add and modify data,
- explain() and stats() – show various statistics.

Listing 2 presents an example of MongoDB methods.

```
db.mydb.find({name:"Jan Kowalski"}).limit(3).forEach(printjson);
```

Listing 1. Simple MongoDB query in JavaScript

MongoDB supports cursors that comprises a control structure for the successive traversal of records in a result set. Main cursor methods are: forEach(func), print(), map(), hasNext() and next().

Listing 2 presents an example of MongoDB cursor.

```
var cursor = db.mydb.find();
while (cursor.hasNext()) printjson(cursor.next());
```

Listing 2. Simple MongoDB cursor with iterator (JavaScript)

[4] Binary data

4.2. Data structure schema

It is proposed to use a modified version of the RDF/JSON serialization [22] to store triples in the MongoDB. Three keys symbolize subject, predicate and object. All three keys should be JSON string. The key named 'subject' should contain RDF subject that should be blank node or URI reference type[5]. The key named 'predicate' should contain RDF predicate that should be URI reference type. The key named 'object' should contain RDF object that should be blank node, URI reference, plain literal or typed literal type. Table 1 presents types of RDF terms.

Table 1

Types of RDF terms in MongoDB

Types	Subject	Predicate	Object	Example
URI	allowed	allowed	allowed	`<http://example.org/path/>`
Blank node	allowed	forbidden	allowed	`_:me`
Plain literal	forbidden but possible	forbidden	allowed	`"chat"@en`
Typed literal	forbidden but possible	forbidden	allowed	`"f"^^<http://example.org/char>`

When type of value is a plain literal it optionally can be defined with language suffix. Languages are determined by appending the simple literal with 'at' sign and the language tag, that is defined in [23] and if supplied, it must not be empty. When the type of value is typed literal it should contain URI similarly append double caret signs followed by any legal URI full form. Literals are written inside double quotation marks.

When the type of value is URI, value should be full URI, not just short QName [24]. The URIs are in enclosed in inequality signs. Blank nodes are represented by the underscore sign, colon sign and node identifier.

Listing 3 presents a schema of a single object (record). Listing 4 presents a simple RDF statement based on JSON Schema [25] and used in MongoDB.

4.3. Mapping SPARQL to Mongo Query Language

It is proposed to introduce a triplestore based on document-oriented MongoDB. RDF has different query languages. The official and most widely used is the SPARQL [4][6].

[5] There is no problem to literals will be supported in the future. SPARQL allows such solution.

[6] It is a recursive acronym that stands for SPARQL Protocol and RDF Query Language.

```
{"description":"RDF/JSON Schema for MongoDB",
 "type":"object",
 "properties":{
 "subject":{
 "type":"string",
 "description":"Subject, possible types of values: bnode and uri
                (in future literal and typed-literal)",
},
"predicate":{
 "type":"string",
 "description":"Predicate, possible types of values: uri",
},
"object":{
 "type":"string",
 "description":"Object, possible types of values: bnode, uri, literal
                and typed-literal",
}
}
}
```

Listing 3. Schema of a single object in JSON Schema

```
{subject: "_:a",
 predicate: "<http://xmlns.com/foaf/0.1/name>",
 object: "\"Dominik Tomaszuk\""}
```

Listing 4. RDF/JSON in MongoDB

SPARQL is adapted to the specific structure of RDF graphs and is based on triples that constitute them. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along with their conjunctions and disjunctions. It also supports extensible value testing and constraining queries by source RDF graph. The results of SPARQL queries can be results sets or RDF graphs.

SPARQL has a similar structure as SQL [26]. There are a lot of keywords that are similar to SQL, such as: SELECT, DISTINCT, WHERE, ORDER BY etc. The SELECT form of results returns all or a subset of, the variables bound in a query pattern match. The WHERE clause provides the basic graph pattern to match against the data graph. The WHERE clause may have a FILTER keyword. In FILTERs restrict solutions to those for which the filter expression evaluates to true. The ORDER BY clause establishes the order of a solution sequence. Following the ORDER BY clause is a sequence of order comparators, composed of an expression and an optional order modifier either ASC() or DESC(). The OFFSET causes the solutions generated to start after the specified number of solutions. The LIMIT clause puts an upper bound on the number of solutions returned.

To make queries concise, SPARQL allows the definition of prefixes and base URIs in a fashion similar to Terse RDF Triple Language (Turtle) [7].

SPARQL support variables. A query variable is a member of the set V where V is infinite and disjoint from O (RDF term, see section 1). Variables are indicated by a "?" or "$" prefix. Most forms of SPARQL query contain a set of triple patterns called a basic graph pattern. Triple patterns are like RDF triples except that each of the subject, predicate and object may be a variable. More formally, a triple pattern is member of the set: (O ∪V) × (U ∪V) × (O ∪ V). A basic graph pattern matches a subgraph of the RDF data when RDF terms from that subgraph may be substituted for the variables and the result is RDF graph equivalent to the subgraph. Triple Patterns are written as a whitespace-separated list of a subject, predicate and object. Triple patterns with a common subject can be written so that the subject is only written once and is used for more than one triple pattern by employing the semicolon notation. If triple patterns share both subject and predicate, the objects may be separated by comma sign. Listing 5 presents a simple SPARQL query.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?mbox
WHERE
{?x foaf:name ?name .
 ?x foaf:mbox ?mbox . }
```

Listing 5. Simple SPARQL query

MongoDB uses own query syntax and do not support SPARQL. Mongo Query Language is an object-oriented imperative language. SPARQL is a domain-specific declarative language. Therefore, it is difficult to replace the SPARQL to MongoDB Query Language.

It is proposed an algorithm that maps SPARQL to Mongo Query Language. This proposal allows MongoDB change in the tripletore and SPARQL endpoint. The proposals return results in Turtle. Therefore, results can be processed again. The first algorithm matching:

- single pattern,
- multiple patterns,
- literals with language tags,
- literals with arbitrary datatypes,
- optional patterns.

The algorithm supports restrictions the values of strings with regex function with and without flags. Regex function is equivalent to XPath [31] fn:matches and supports regular expressions [30]. The algorithm supports

also restricting numeric values using >, <, <= and >= operators, that are equivalent to XPath op:numeric-greater-than and op:numeric-less-than with or without logical disjunction. The second algorithm supports distinct matching. The third algorithm matching alternatives. Algorithms 1, 2 and 3 present an idea of mapping SPARQL to an MongoDB object-oriented language.

```
Input: RDF/JSON
Output: Turtle
TP←triple patterns
LTP←singly-linked list of TP
V←query variables
RT←RDF term (literal ∪ typed-literal ∪ URI ∪ black node), RU←RDF URI,
RS←RDF URI ∪ black node
T←RDF triples, T = RT × RU × RO
TQV←three elements array of singly-linked list of query variables or RDF terms
TQV[0]←singly-linked list of V ∪ RS, TQV[1]←V ∪ RU, TQV[2]←V ∪ RT
JO←JSON object where key ∈ "subject" ∪ "predicate" ∪ "object"
C←cursor
LC←singly-linked list of cursors
 parse SELECT clause, WHERE clause
 add V ‖ RT to TQV, TP to LTP
 Foreach LTP do
  create JO, C
  add C to LC
  Foreach TQL do
   If V.previous == V then
    add to JO in C condition with value of previous RT in find method
   end
   If TQL value != V then
    add to JO in C condition in find method
   end
  end
  If FILTER ∈ WHERE than
   If regex function ∈ FILTER then
    add to JO in C condition with regular expression in find method
   Else
    add to JO in C condition with numeric values operator in find method
   end
  end
  If OPTIONAL ∈ WHERE then
   mark C as optional
  end
  If ORDER BY clause ∉ ∅ then
   parse ORDER BY
   create JO
   If DESC() ∈ ORDER BY then
    add to JO with value 1 to sort method
   Else
    add to JO with value -1 to sort method
   end
   add to C
   move C to top of LC
  end
  If LIMIT clause ∉ ∅ then
   parse LIMIT
   add to C limit method
  end
```

```
  If OFFSET clause ∉ ∅ then
   parse OFFSET
   add to C skip method
  end
 end
 Foreach LC do
  create loop
  print results in loop
  If optional == true then
   print results in previous loop that is not optional
  end
 end
```

Algorithm 1. Mapping SPARQL to Mongo Query Language main algorithm

```
Input: RDF/JSON
Output: Turtle
TP←triple patterns
LTP←singly-linked list of TP
V←query variables
RT←RDF term (literal ∪ typed-literal ∪ URI ∪ black node),
RU←RDF URI, RS←RDF URI ∪ black node
T←RDF triples, T = RT × RU × RO
TQV←three elements array of singly-linked list of query variables or RDF terms
TQV[0]←singly-linked list of V ∪ RS, TQV[1]←V ∪ RU,
TQV[2]←V ∪ RT
JO←JSON object where key ∈ "subject" ∪ "predicate" ∪ "object"
C←cursor
LC←singly-linked list of cursors
 parse SELECT clause, WHERE clause
 add V ‖ RT to TQV, TP to LTP
 Foreach LTP do
  create JO
  create C
  add C to TDC
  Foreach TQL do
   If V.previous == V then
    add to JO in C condition with value of previous RT in distinct method
   end
   If TQL value != V then
    add to JO in C condition in distinct method
   end
  end
  If FILTER ∈ WHERE then
   If regex function ∈ FILTER then
    add to JO in C condition with regular expression in distinct method
   Else
    add to JO in C condition with numeric values operator in distinct method
   end
  end
  add to C limit method with 1 in parameter
 end
 create C condition with value of RT in find method
 print results
```

Algorithm 2. Mapping SPARQL DISTINCT to Mongo Query Language algorithm

```
Input: RDF/JSON
Output: Turtle
TP←triple patterns
LTP←singly-linked list of TP
V←query variables
RT←RDF term (literal ∪ typed-literal ∪ URI ∪ black node), RU←RDF URI,
RS←RDF URI ∪ black node
T←RDF triples, T = RT × RU × RO
TQV←three elements array of singly-linked list of query variables or RDF terms
TQV[0]←singly-linked list of V ∪ RS, TQV[1]←V ∪ RU, TQV[2]←V ∪ RT
JO←JSON object where key ∈ "subject" ∪ "predicate" ∪ "object"
C←cursor
 parse SELECT clause, WHERE clause
 create JO, C
 parse left union group, right union group
 add to JO in C condition with values of previous RT in find method using $or key
 If ORDER BY clause ∉ ∅ then
  parse ORDER BY
  create JO
  If DESC() ∈ O then
   add to JO with value 1 to sort method
  Else
   add to JO with value -1 to sort method
  end
  add to C
  move C to top of LC
 end
 If LIMIT clause ∉ ∅ then
  parse LIMIT
  add to C limit method
 end
 If OFFSET clause ∉ ∅ then
  parse OFFSET
  add to C skip method
 end
 Foreach C do
  print results
 end
```

Algorithm 3. Mapping SPARQL UNION to Mongo Query Language algorithm

5. Experimental results

Speed criterion, which is adopted to measure the document-oriented triplestore, is time of load and select triples. Loading is the process of adding one or more records, objects or statements to the database. In SQL and SPARQL 1.1 it is INSERT clause. In addition, triples could be also added through various APIs. In Mongo Query Language loading uses insert() or save() methods. Selection is the process of return of a result set of records, objects or statements from the database. These results may include order and specified criteria. In SQL and SPARQL it is SELECT clause. In Mongo Query Language selection uses find() or distinct() methods.

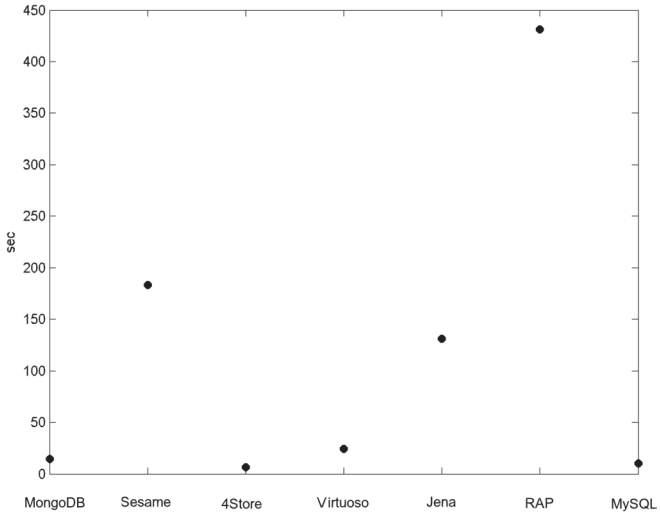

Fig. 2. Times of loading triples

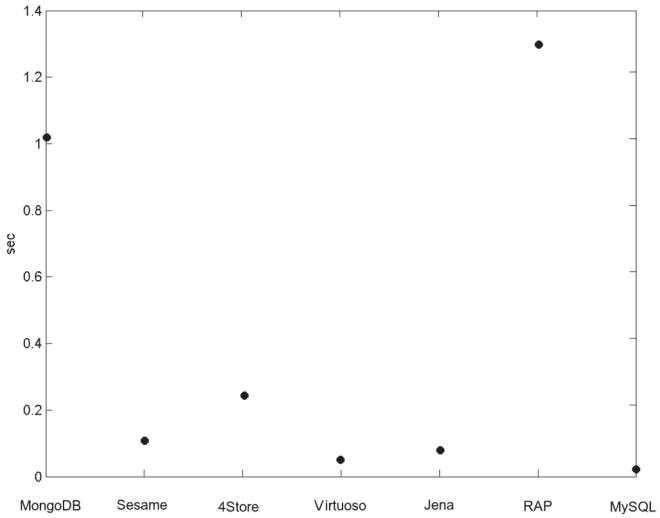

Fig. 3. Times of selecting triples

Main tests are executed on typical desktop computer with two Intel Core 2 Quad 2666MHz CPUs and 4GB RAM. Testing computer running with Ubuntu 9.10 in Ext3 filesystem. Tests are based on the Berlin SPARQL Benchmark [27]. Tests are performed on 1 000 313 triples. Selection is based on the Query 6. Tests concern MongoDB [20] compared to triplestore such as Sesame [9], 4Store [11], Virtuoso [28], Jena SDB with MySQL [5], RAP [10] and MySQL with InnoDB engine [29].

Figure 2 presents times of loading triples. The chart shows that the best times have 4Store, MySQL and MongoDB. It is important that MySQL is a relational database, which is not triplestore and data in this database are not triples, so the best times belongs to 4Storne and MongoDB with triples in RDF/JSON.

Figure 3 shows times of matching multiple patterns. The chart shows that the worst times have RAP and MongoDB. Mongo Query Language do not select multiple patterns well but match simple patterns, distinct patterns (algorithm 2) and alternatives (algorithm 3); it has time close to others triplestores and relational database.

6. Conclusions

The problem of how to store RDF triples has produced many proposals and models. Some of them are relational, object-oriented or use a graph. I have produced a thought-out and simple proposal. I believe my document-oriented triplestore based on RDF/JSON is an interesting approach.

MongoDB is a document-oriented database that uses JSON. I propose algorithms that transform SPARQL queries to Mongo Query Language. This allows MongoDB to be a triplestore. I also present RDF/JSON that can be used in this triplestore.

The experiments show that MongoDB and document-oriented key-value with semi-structured JSON database could be a good triplestore. The advantage of MongoDB as triplestore is that it is very scalable. It means that MongoDB provides more efficient work with increasing number of objects in the database.

I realized that some further work on this issue is still necessary to extend proposed algorithms to support CONSTRUCT, ASK, DESCRIBE queries and source graphs.

Acknowledgements

The author would like to thank Ivan Herman from the World Wide Web Consortium. Professor Henryk Rybinski's comments and support were invaluable.

REFERENCES

[1] G. Klyne and J. J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. World Wide Web Consortium, 2004.

[2] P. Hayes. RDF Semantics. World Wide Web Consortium, 2004.

[3] F. Manola E. and Miller. RDF Primer. World Wide Web Consortium, 2004.

[4] E. Prud'hommeaux and A Seaborne. SPARQL Query Language for RDF. World Wide Web Consortium, 2008.

[5] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne and K. Wilkinson. Jena: implementing the semantic web recommendations. International World Wide Web Conference, Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters, 2004.

[6] D. Beckett, RDF/XML Syntax Specification (Revised). World Wide Web Consortium, 2004.

[7] D. Beckett, T. Berners-Lee, Turtle – Terse RDF Triple Language. World Wide Web Consortium, 2008.

[8] T. Berners-Lee, Notation3. World Wide Web Consortium, 2006.

[9] J. Broekstra, A. Kampman and F. van Harmelen. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema, The Semantic Web – ISWC 2002, Springer, 2002.

[10] R. Oldakowski, C. Bizer, D. Westphal. RAP: RDF API for PHP. In Proceedings International Workshop on Interpreted Languages, MIT Press, 2004.

[11] S. Harris, N. Lamb and N. Shadbolt. 4store: The Design and Implementation of a Clustered RDF Store, The 5th International Workshop on Scalable Semantic Web Knowledge Base Systems, 2009.

[12] A. Muys. Building an EnterpriseScale Database for RDF Data.

[13] A Fnewman. Getting Started with JRDF. 2010.

[14] A. Newman, J. Hunter, Y. Li, C. Bouton and M Davis. A Scale-Out RDF Molecule Store for Distributed Processing of Biomedical Data. Semantic Web for Health Care and Life Sciences Workshop, 2008.

[15] J. Aasman. Allegro Graph: RDF Triple Database. Technical Report 1, Franz Incorporated, 2006.

[16] J. Grant and D. Beckett. RDF Test Cases. World Wide Web Consortium, 2004.

[17] M. Cowlishaw. ECMAScript language specification. International Organization for Standardization, 1998.

[18] D. Crockford. The application/json Media Type for JavaScript Object Notation (JSON). Internet Engineering Task Force, 2006.

[19] S. Abiteboul. Querying semi-structured data. Database Theory-ICDT'97, Springer, 1997.

[20] K. Chodorow. MongoDB manual. 10gen. 2009.

[21] M. Dirolf. BSON specyfication. 2009.

[22] D. Tomaszuk. Serializing RDF graphs in JSON. III Krajowa Konferencja Naukowa Technologie Przetwarzania Danych, 2010.

[23] H. Alvestrand, Tags for the Identification of Languages. Internet Engineering Task Force, 2001.

[24] T. Bray, D. Hollander, A. Layman and R. Tobin. Namespaces in XML (Second Edition). World Wide Web Consortium, 2006.

[25] K. Zyp. A JSON Media Type for Describing the Structure and Meaning of JSON Documents. Internet Engineering Task Force, 2010.

[26] D. D. Chamberlin and R. F. Boyce. ISO/IEC 9075:1992 – Database Language SQL. International Organization for Standardization, 1992.

[27] C. Bizer and A. Schultz. Berlin SPARQL Benchmark. International Journal On Semantic Web and Information Systems, 2009.

[28] O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. Networked Knowledge – Networked Media. Springer, 2009.

[29] M. Widenius and D. Axmark. MySQL reference manual: documentation from the source. O'Reilly Media, 2002.

[30] M. Davis and A. Heninger. Unicode Regular Expressions, Unicode Consortium, 2008.

[31] A. Malhotra, J. Melton and N. Walsh. XQuery 1.0 and XPath 2.0 Functions and Operators, World Wide Web Consortium, 2007.

Dominik Tomaszuk
Institute of Computer Science
University of Bialystok
dtomaszuk@ii.uwb.edu.pl