ENGINEERING JOURNAL

Article

Effective Partitioning and Multiple RDF Indexing for Database Triple Store

Sunitha Abburu[a,*] **and Suresh Babu Golla**[b]

Department of Computer Applications, Adhiyamaan College of Engineering, Hosur, Tamilnadu, India
Email: [a]drsunithaabburu@yahoo.com (Corresponding author), [b]suresh_babu_golla@yahoo.com

Abstract. The capability of semantic technology leads to adaption of semantic technology to multiple applications of various domains. Due to vast number of applications, the size of RDF triple store is increasing. Effective semantic query execution has become a challenge due to the structure of RDF triple store. Effective indexing and partitioning leads to good sematic query performance against RDF triple store. The current research work has focused on various indexing techniques and proposed a predicate centric partitioning and multiple RDF indexing method for database triple store. A detailed analysis process is been executed to measure and compare the query performance. The current method is evaluated using standard benchmark and real datasets with various indexing techniques. Later the methodology is applied to R&D project management dataset. A set of twenty seven queries has been derived by considering various user requirements that cover most of the SPARQL constructs. The method is implemented and a detailed evaluation has been successfully carried out. The query time is evaluated on R&D project management dataset. The test results indicate that the proposed method provides considerable improvement in overall query performance.

Keywords: Ontology, RDF, partition, indexing, SPARQL query.

ENGINEERING JOURNAL Volume 19 Issue 5
Received 24 October 2014
Accepted 23 February 2015
Published 31 October 2015
Online at http://www.engj.org/
DOI:10.4186/ej.2015.19.5.139

DOI:10.4186/ej.2015.19.5.139

1. Introduction

Semantic technology provides various tools and techniques that supports:

- Automation, integration
- Common framework and understanding
- Reusability and sharability across application, enterprise and community boundaries and
- Machine processable formats

Ontology is a formal, explicit specification of a shared conceptualization [1]. Ontology plays a vital role in Semantic Web (SW). SW is the vision of Tim Berners-Lee. The objective is to represent the knowledge in machine understandable format [2]. Ontology provides domain vocabulary, domain knowledge, common understanding, data sharability, information interoperability, reusability and supports semantic information retrieval. The major elements of ontology are concepts, properties that relates concepts and the instances of the concepts [3].

Ontology language is a formal language that encodes domain knowledge and supports reasoning. Ontology languages are classified by structure or syntax. The most common and popularly famous language is markup ontology language, which uses a markup scheme to encode knowledge. Among all markup ontology languages Resource Description Framework (RDF) [4], Resource Description Framework and Schema (RDFS) [5] and Web Ontology Language (OWL) [6] are most popular [7] and recommended by World Wide Web Consortium (W3C). RDF has wide acceptance due to its flexibility. RDF represents data in homogeneous and machine understandable format. This enable application interoperability and semantic retrieval.

RDFS facilitates description of ontology elements with the help of knowledge representation data models in the form of classes/concepts and properties known as TBox. RDF is a standard data model that interchange and merge instance data, known as ABox. TBox is associated with classes/concepts and ABox is associated with instances of the classes. RDF data model represents resources in the form of subject-predicate-object expressions known as RDF triple. A triple describes a resource in the form of (subject, predicate, and object) or (subject, property, value). Subject-predicate-object databases are known as triple stores. SPARQL is a formal RDF query language [8]. It is used retrieve and manipulate triples from RDF triple store.

SPARQL query language is used to execute semantic queries on RDF triple store. SPARQL to SQL rewriters are used to execute semantic queries on classical Relational Database Management Systems (RDBMS). Semantic queries on relational data bases can be executed by mapping relational data base constructs with the ontology elements and SPARQL to SQL rewriters such as D2R server [9] and Vituoso RDF view [10]. Christian et al. [11] evaluated and compared the performance of RDF stores and SPARQL to SQL rewriters. The evaluation proves that the fastest SPARQL to SQL rewriters outperforms fastest native RDF stores with increasing dataset size. However, there are no exact counter parts for several SPARQL query constructs in SQL.

RDF triple store can be stored in file formats, native triple store (eg. AllegroGraph [12], Jena TDB [13], HStar [14]) or in relational data bases in triple format (eg. Jena SDB [15], Oracle semantic store [16], virtuoso RDF store [17]). RDF store uses file system to store triples and SPARQL is a formal query language to retrieve and manipulate RDF triples. Database RDF stores use relational or object relational databases as the backend store [18] to store RDF triples and SPARQL as a formal query language to retrieve and manipulate triples with the support of SPARQL query engine. Native triple store is straight forward than triples stored on relational data bases. However database management systems have many significant features such as performance, robustness, reliability and availability. Thus relational database is a very good solution for storing and querying RDF triple [19]. Thus database triple stores are the effective method to store RDF triples. Due to the structure and flexibility of RDF and adoption of semantic technologies by various domains, the RDF triple store size is increasing from million to billion to trillion. The performance of the system depends upon RDF storage structure, indexing techniques, query and reasoning capabilities. Efficient query processing is required for any effective system [20]. The current research work focuses on query performance aspects of database triple stores.

The current research focuses on two aspects: partitioning and indexing RDF triples to improve the performance of the system in terms of the query processing time. Major challenges with the current RDF indexing methods are: data redundancy, index storage cost, time complexity etc. The current research work mainly focusses on query processing time for large and real time RDF triples stored in relational data bases.

The paper presents an effective predicate based partitioning and multiple RDF indexing method. The performance of the method is evaluated and compared with popular RDF indexing methods. The evaluation is done using different, widely accepted benchmark and real datasets for various sizes.

The rest of the paper is organized as follows. Section 2 discusses RDF indexing methods, section 3 illustrates predicate based partitioning and multiple RDF indexing method, Section 4 provides evaluation results and Section 5 summarizes the conclusions.

2. Discussion

There are several approaches to store and query RDF triples in relational databases. Among all the relational database triple stores, the straightest forward and popular approach is vertical table approach [21]. Vertical table approach [22] stores the description of classes, properties and their instances in a single universal table. The schema of this table has three columns: subject, predicate, object and represents instance identifier, property of an instance and value of an instance, respectively. The oblivious advantages of the approach are: simple structure, fixed number of columns, most straight forward relational method and both data and metadata can be processed in the same way. Its major limitations are: that every query has to search the whole database, expensive joins are required and provides less query efficiency. The query processing time can be reduced by partitioning and indexing of RDF triples. The following are the various RDF indexing methods proposed by various researchers.

Kolas et al., [23] enhanced the idea of indexing on each column separately by using linked list as data structure. This approach maps each RDF resource 'r' to a set of three pointers {p1, p2, p3} where a resource can be either IRI or Literal value. Pointer p1 points to the first record in which 'r' occurs as subject, pointer p2 points to the first record in which 'r' occurs as predicate and pointer p3 points to the first record in which 'r' occurs as object. Further each record in the triple table consists three points that points to the next triples in the same table containing the same subject, predicate and object respectively. This approach consumes less storage space, but slightly high I/O cost.

G.H.L Fletcher et al., [24] proposed an approach which is similar to Kolas's approach called TripleT. In this approach each RDF resource 'r' contains three buckets, one for all pairs of (p, o) where r appears as subject, one for all pairs of (s, o) where r appears as predicate and one for (s, p) where r appears as object. Each bucket can be represented as a linked list. For example in the case (r, p, o) r contains a pointer that points to the first triple in the table where r is presented as subject. Further each triple has a pointer that points to the next triple in which r appears as subject. Oblivious advantages of TripleT are: simple to design and maintain. All information of a resource can be maintained in a list. However, care is needed while building lists for resources. Multiple lists are to be processed while query contains multiple variables.

L. Ma et al., [25] proposed a solution for addressing the issues of vertical table approach. The approach describes four different kinds of B-Tree indexing techniques on the triple table. The first kind of indexing technique builds a separate index on each column. That is an index on subject column alone, an index on predicate column alone and an index on object column alone. The second kind builds an index on the combination of subject and property columns and an index on object column alone. The third kind builds an index on combination of property and object columns. The fourth kind of indexing builds an index on combination of all the three columns. It is observed that the first kind of indexes give better query performance than the remaining three. The same has been considered for performance evaluation in the current research work.

C. Weiss et al., introduced HexaStore [26] that is an efficient and scalable RDF data engine. HexaStore has six Btree indexes, one for each possible order of triples. The triple patterns are (S, P, O), (S, O, P), (P, S, O), (P, O, S), (O, S, P) and (O, P, S). This approach is a multiple indexing approach. It allows fast merge joins for any pair of triple patterns and reduces query processing time. But RDF elements are redundant, for example (S, O, P) and (O, S, P) indexes have redundant RDF elements.

T. Neumann et al., [27] describes an extensive multiple indexing approach. This enhances the idea of six tuple indexing with nine additional indexes. The nine additional indexes are: six indexes on all six possible binary permutations (SP, SO, PS, PO, OS, OP) and three indexes on each column of SPO triple. The additional nine indexes are also known as projection indexes. The projection indexes map search keys to all the triples that satisfy the search key. The projection indexes optimize query performance and reduce intermediate joins.

HexaStore, Neumann's RDF indexing method and L. Ma's indexing approaches build index on permutations of triple patterns. These have redundant RDF elements that increasing storage cost. Kolas's

and Fletcher's indexing methods use linked list approach to build indexing over RDF triples. Handling pointers for massive RDF stores is complex. Major limitations with the RDF indexing approaches in the state of the art are:

- Data redundancy
- Indexing storage cost
- Placing all possible instances for an RDF query with in small scope
- Query processing time
- Improve relevance of query results

3. Approach

The current research work is a part of a Research and Development (R&D) project, financially sponsored by DRDO (Defense Research and Development Organization, Govt. of India) the project deals with the semantic management of R&D project management. The proposed approach is applied on R&D project management dataset that represents R&D project knowledge base using multiple sub ontologies.

The current research work discusses a generic two step approach.

- Partitioning on predicate
- Multiple RDF Indexing (predicate centric indexing, i.e. P, PS, PO, PSO, POS)

The RDF partitioning and indexing method is applied to reduce redundancy, query processing time and placing all possible instances for an RDF query with in small scope that supports fast retrieval.

Few advantages of using partition techniques are [28]:

- Easy and Improved manageability of large tables
- Fast query processing
- Flexible indexing
- Better optimization of storage costs
- Larger table capacity

Popular partitioning approaches are range, list and hash partitioning. The current research adopted hash partitioning. List and range partitions are not appropriate for RDF triple store. Since no RDF element gives any specific list or range of values and RDF elements are interchangeable in different context. That is, a subject in an RDF triple can be an object in another RDF triple. Hash is a partition technique that distributes triples equally among specified number of partitions based on partition column. Hash partition is more suitable for RDF data structure. Hash partition on RDF can be done on any RDF element: subject, predicate or object. For R&D project management, most of the user query patterns are with known predicate. Further various benchmark and real time SPARQL queries and analysis concludes that most of the user query patterns contain known predicate. To identify the partitioning element, user query pattern is observed and found that, partition on predicate column gives more effective results in terms of query processing time.

The effectiveness of the indexing techniques is evaluated and measured based on the coverage of RDF elements and the user query patterns. SPARQL has four different forms SELECT, CONSTRUCT, DESCRIBE and ASK [29]. All the four forms of SPARQL queries return results based on a set triple patterns. The triple pattern set contains patterns to be matched against an RDF dataset. Various possible SPARQL query pattern are shown in Table. 1.

Table 1. SPARQL query patterns.

Pattern	Subject	Predicate	Object
1	:s	:p	:o
2	:s	:p	?
3	?	:p	:o
4	?	:p	?
5	:s	?	:o
6	:s	?	?
7	?	?	:o
8	?	?	?

A sample SPARQL query where clause pattern, looks like: "<http://example.org/book/book1> <http://purl.org/dc/elements/1.1/title> ?title". Elements preceded by ? are known as variables. Variables represents unknown values or data to be found. SPARQL query processor searches for the data bound to the variables so that the query pattern matches for RDF data in triple store. Index on known elements or combination of known elements of a pattern assist query processor to find data bound to variables and improve query performance.

To improve the query performance, the current approach applies partitioning on predicate and multiple indexing on all combinations of RDF elements that are predicate centric (P, PS, PO, PSO and POS), see Fig. 1. The method uses five indexes P, PS, PO, PSO and POS which reduces index storage cost and covers most of the user query patterns.

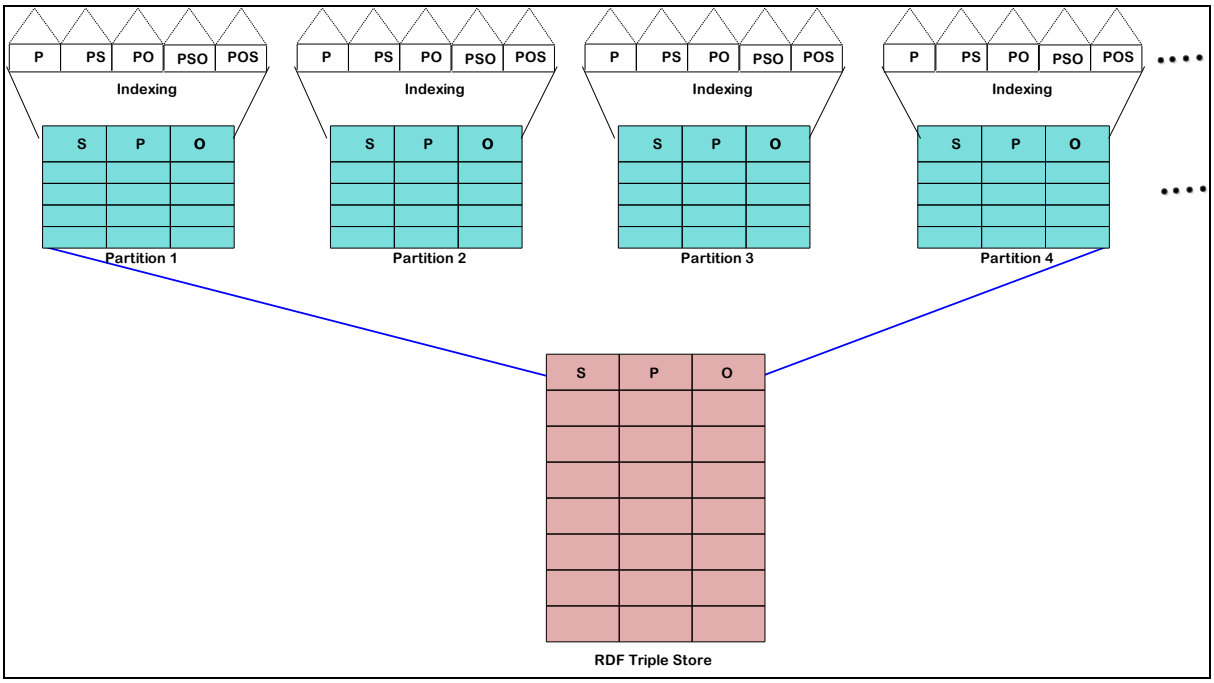


Fig. 1. Partition based multiple RDF indexing method.

Steps involved in the current partition based multiple RDF indexing are:

Step 1. Create table space
Step 2. Create table by applying hash partition on predicate element
Step 3. Load RDF data
Step 4. Create multiple indexes: P, PS, PO, PSO and POS

4. Evaluation

There are several real and benchmark datasets and respective SPARQL queries available in the literature to evaluate RDF storage and retrieval performances. Various real and benchmark data sets along with the benchmark generator algorithm are described in [30]. DBpedia [31] explains a real data set, BSBM and SP2 [32] which are popular and widely accepted as benchmark data sets. The current work has taken DBpedia, BSBM and SP2 data sets to evaluate the predicate centric partitioning and multiple indexing approach.

Various performance metrics benchmarks are defined in the state of art. BSBM performance metrics are Query Mixes per Hour (QMpH), Queries per Second (QpS) and Load Time (LT). SP2 performance metrics are Arithmetic Mean (AM) and Geometric Mean (GM) of elapsed time of SP2 benchmark queries. DBpedia measures Query Processing Time (QPT) of individual queries.

The current methodology is been implemented for R&D project data base. The effectiveness of the methodology is evaluated in two ways 1. Using real and benchmark datasets. 2. R&D project management dataset.

To evaluate the performance of the current method, various indexing techniques (L.Ma's First index, Hexa Store and Neumann's RDF index) including the current approach have been applied on BSBM, SP2

DOI:10.4186/ej.2015.19.5.139

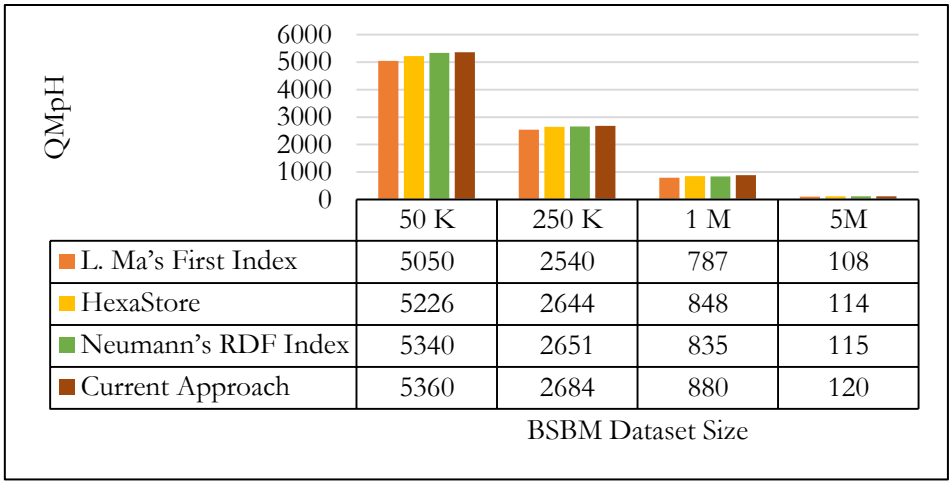
and DBpedia datasets. The RDF indexing methods are implemented for different sizes of BSBM, SP2 benchmark datasets (50K, 250K, 1M, 5M) and DBpedia datasets (2L,4.5L, 15M), where L: lakhs K: thousand and M :million triples.

Testing is done in a windows8 desktop system of memory 12GB and processor 3.4 GHz. Oracle semantic store is used as a triple store. The evaluation report is presented below.

4.1. Observations from Performance Evaluation Test Results on Benchmark and Real Dataset

a) BSBM benchmark dataset

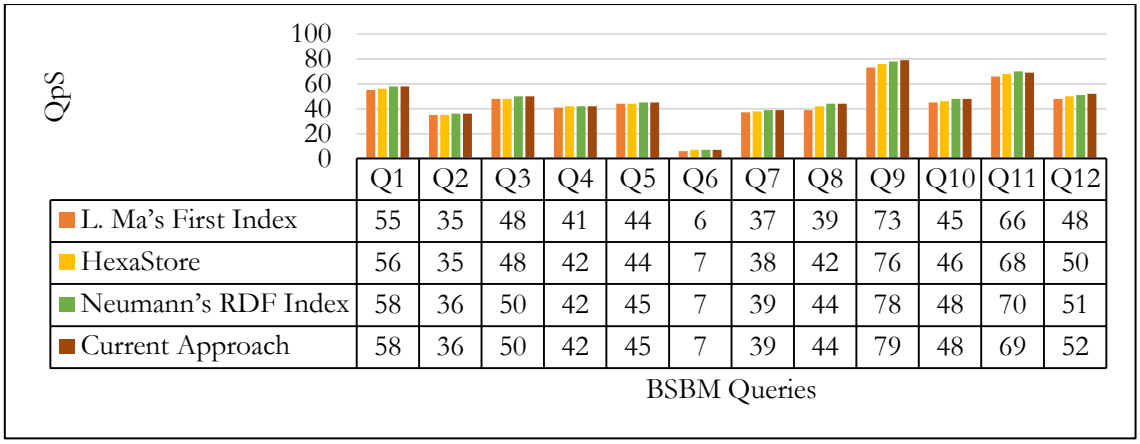
BSBM performance metrics are QMpH and QpS. BSBM test drive is executed with 25 warm ups and 100 runs on various RDF indexing methods. It is observed that, predicate based partitioning and indexing method improves the overall QMpH as seen in Fig. 2.



	50 K	250 K	1 M	5M
L. Ma's First Index	5050	2540	787	108
HexaStore	5226	2644	848	114
Neumann's RDF Index	5340	2651	835	115
Current Approach	5360	2684	880	120

Fig. 2. QMpH analysis of RDF indexing methods on various sizes of BSBM benchmark dataset.

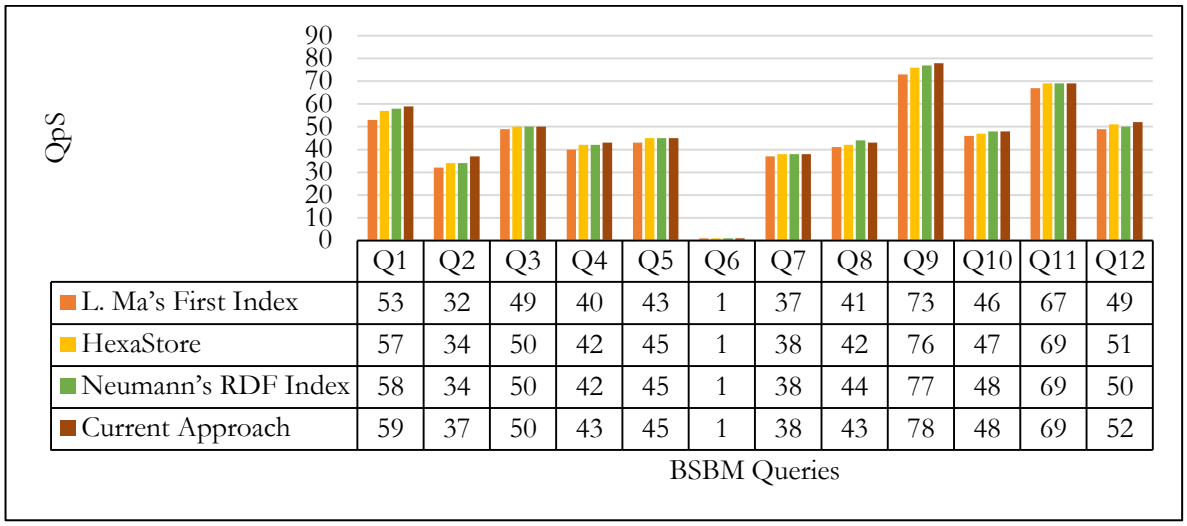
Q5 and Q8 are the two queries for which the performance is almost same as other indexing approaches. Q5 and Q6 are queries with multiple filters. This indicates that the predicate based partitioning and indexing technique performance is same as other approaches for the queries with multiple filters. For ten queries out of twelve, the QpS is increasing as the dataset size is increasing. The results are shown in Figs. 3–6.



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
L. Ma's First Index	55	35	48	41	44	6	37	39	73	45	66	48
HexaStore	56	35	48	42	44	7	38	42	76	46	68	50
Neumann's RDF Index	58	36	50	42	45	7	39	44	78	48	70	51
Current Approach	58	36	50	42	45	7	39	44	79	48	69	52

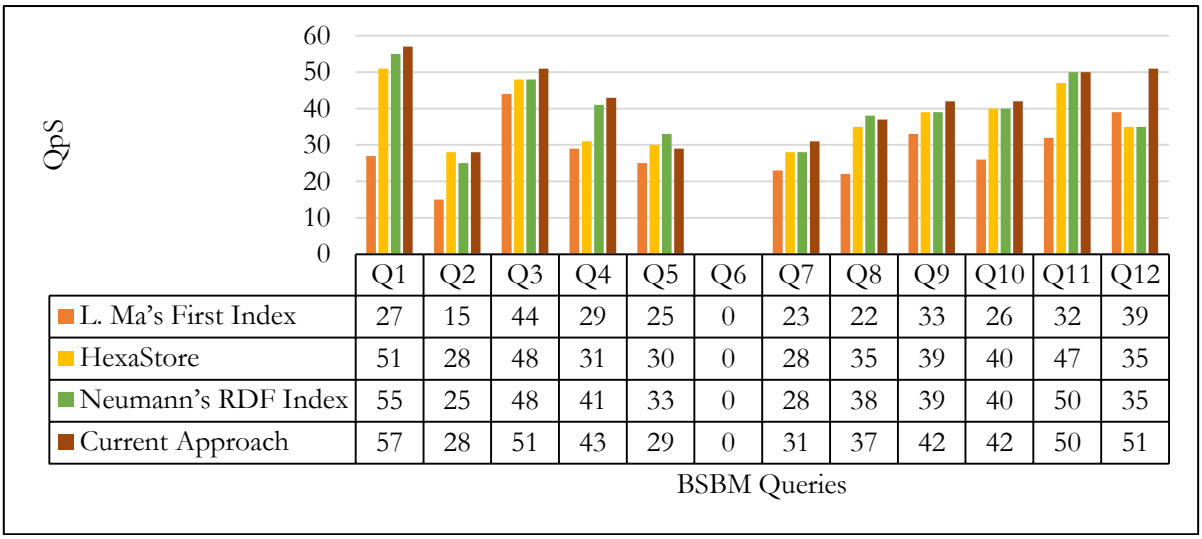
Fig. 3. QpS analysis of RDF indexing methods on BSBM benchmark dataset of size 50K.

DOI:10.4186/ej.2015.19.5.139



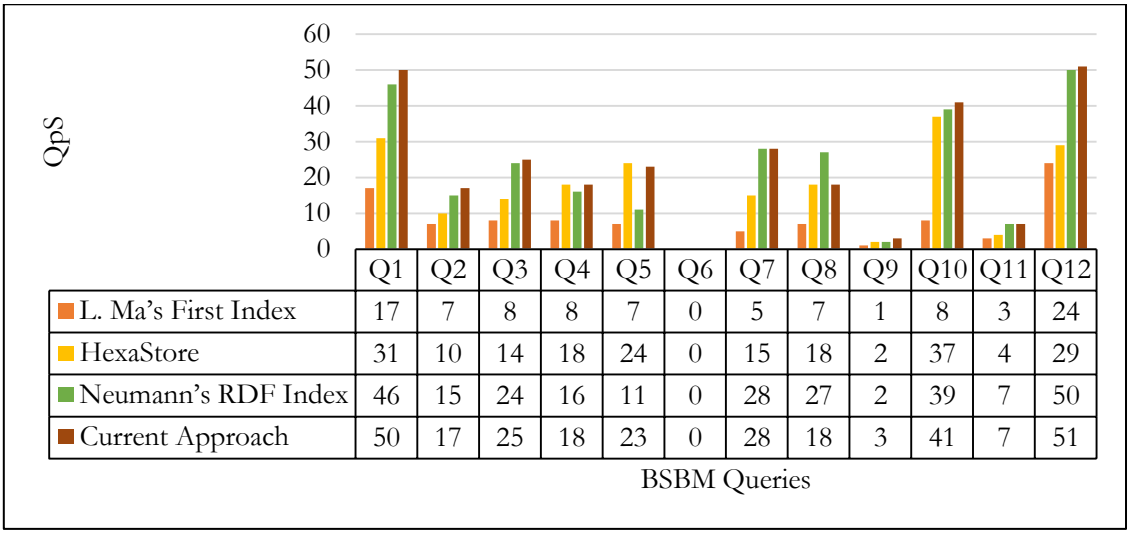
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
L. Ma's First Index	53	32	49	40	43	1	37	41	73	46	67	49
HexaStore	57	34	50	42	45	1	38	42	76	47	69	51
Neumann's RDF Index	58	34	50	42	45	1	38	44	77	48	69	50
Current Approach	59	37	50	43	45	1	38	43	78	48	69	52

Fig. 4. QpS analysis of RDF indexing methods on BSBM benchmark dataset of size 250K.



	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
L. Ma's First Index	27	15	44	29	25	0	23	22	33	26	32	39
HexaStore	51	28	48	31	30	0	28	35	39	40	47	35
Neumann's RDF Index	55	25	48	41	33	0	28	38	39	40	50	35
Current Approach	57	28	51	43	29	0	31	37	42	42	50	51

Fig. 5. QpS analysis of RDF indexing methods on BSBM benchmark dataset of size 1M.



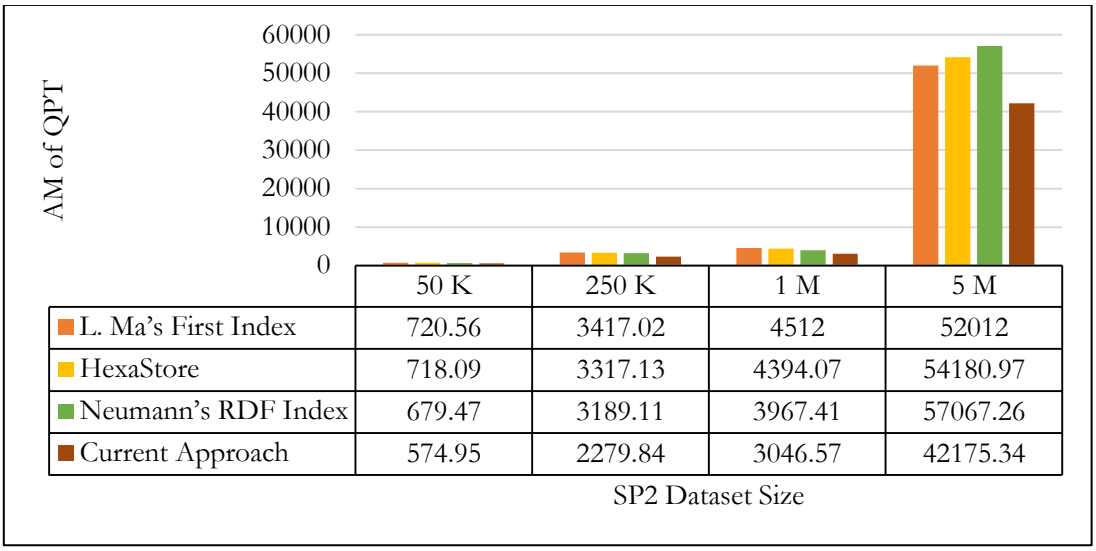
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
L. Ma's First Index	17	7	8	8	7	0	5	7	1	8	3	24
HexaStore	31	10	14	18	24	0	15	18	2	37	4	29
Neumann's RDF Index	46	15	24	16	11	0	28	27	2	39	7	50
Current Approach	50	17	25	18	23	0	28	18	3	41	7	51

Fig. 6. QpS analysis of RDF indexing methods on BSBM benchmark dataset of size 5M.

DOI:10.4186/ej.2015.19.5.139

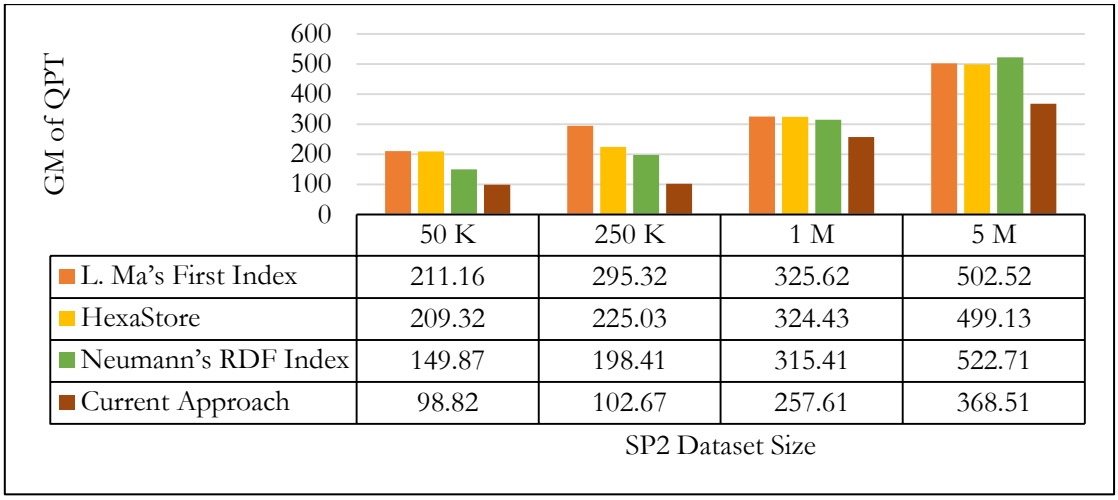
b) SP2 benchmark dataset

The next set of tests have been carried out on various sizes of SP2 benchmark datasets to compare the performance of the current approach with various RDF indexing methods. SP2 performance metrics are AM of QPT, GM of QPT. In the current evaluation process, the AM and GM are measured in milliseconds. The test results in Figs. 7 and 8 indicate that the predicate based partitioning and indexing approach improves query performance.



	50 K	250 K	1 M	5 M
L. Ma's First Index	720.56	3417.02	4512	52012
HexaStore	718.09	3317.13	4394.07	54180.97
Neumann's RDF Index	679.47	3189.11	3967.41	57067.26
Current Approach	574.95	2279.84	3046.57	42175.34

Fig. 7. Test results using SP2 datasets: AM of QPT.

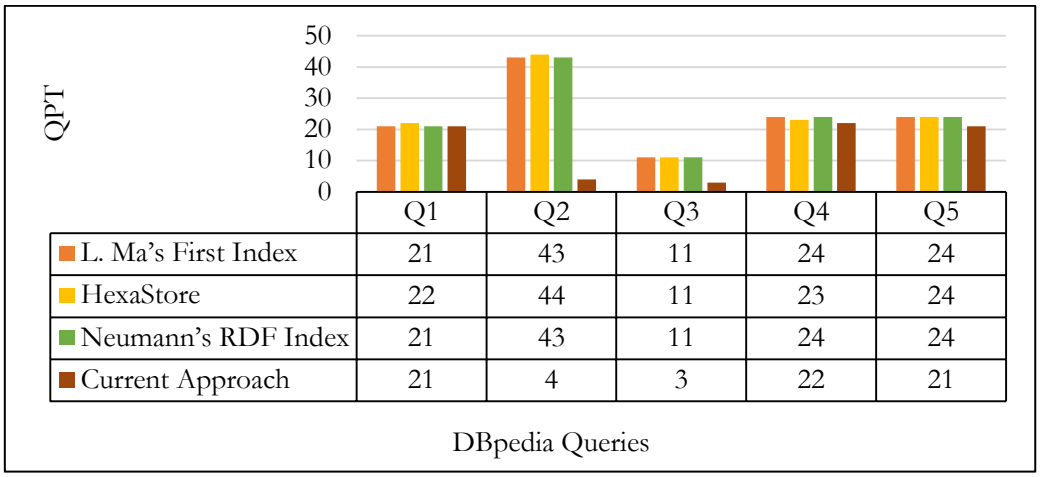


	50 K	250 K	1 M	5 M
L. Ma's First Index	211.16	295.32	325.62	502.52
HexaStore	209.32	225.03	324.43	499.13
Neumann's RDF Index	149.87	198.41	315.41	522.71
Current Approach	98.82	102.67	257.61	368.51

Fig. 8. Test results using SP2 datasets : GM of QPT.

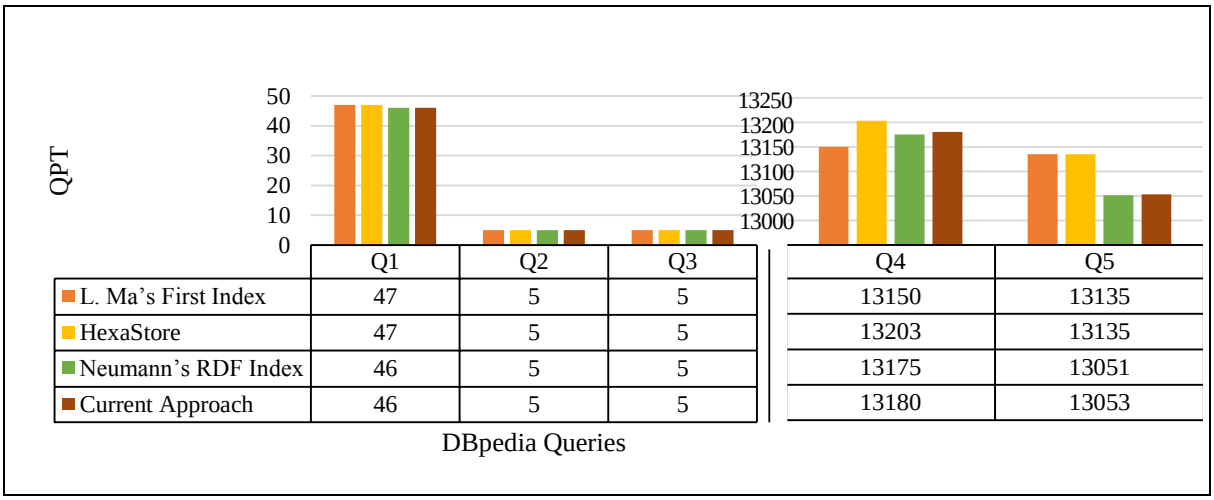
c) DBpedia real dataset

The current methodology performance is compared with other indexing methods by using DBpedia dataset. DBpedia performance metric is QPT in milliseconds of individual queries. DBpedia has three kinds of data sets of different sizes. They are homepages dataset of 2L size, geo-coordinates dataset of size 4.5L and infoboxes dataset of 15M size. All the three types of DBpedia are considered in the evaluation process. It is observed that the current methodology is showing improved results with DBpedia infoboxes and homepages datasets. For geo-coordinates DBpedia dataset, the performance is almost same as other indexing approaches. Figures 9–11 show evaluation results. The results are presented using two different scales depending on the execution time.



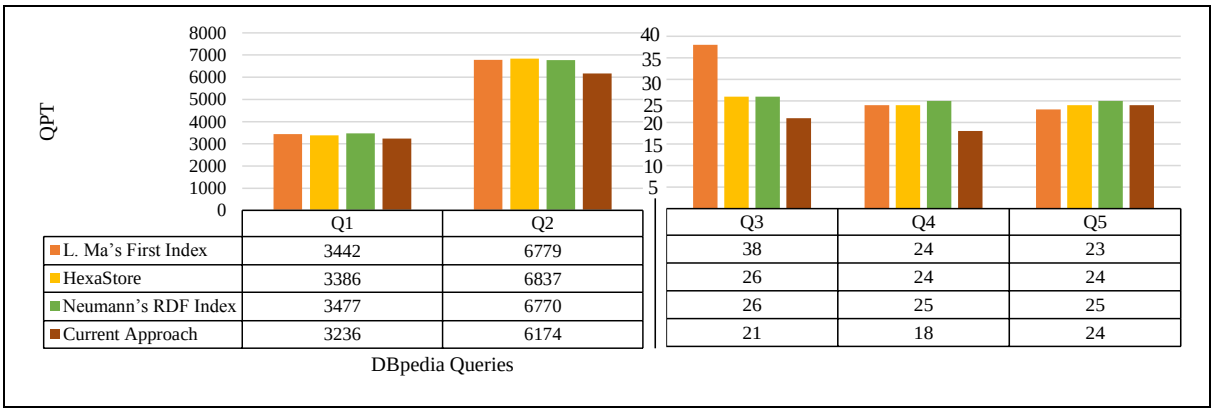
	Q1	Q2	Q3	Q4	Q5
L. Ma's First Index	21	43	11	24	24
HexaStore	22	44	11	23	24
Neumann's RDF Index	21	43	11	24	24
Current Approach	21	4	3	22	21

Fig. 9. Comparison of RDF indexing methods on DBpedia homepages dataset of size 2L.



	Q1	Q2	Q3	Q4	Q5
L. Ma's First Index	47	5	5	13150	13135
HexaStore	47	5	5	13203	13135
Neumann's RDF Index	46	5	5	13175	13051
Current Approach	46	5	5	13180	13053

Fig. 10. Comparison of RDF indexing methods on DBpedia geo-coordinates dataset of size 4.5L.



	Q1	Q2	Q3	Q4	Q5
L. Ma's First Index	3442	6779	38	24	23
HexaStore	3386	6837	26	24	24
Neumann's RDF Index	3477	6770	26	25	25
Current Approach	3236	6174	21	18	24

Fig. 11. Comparison of RDF indexing methods on DBpedia infoboxes dataset of size 15M.

4.2. R&D Project Management Dataset

Several subject areas relevant to the R&D project management domain are identified as follows:

- Project
- Person
- Document
- Event
- Equipment

- Organization
- Publications
- Results
- Cost
- Research areas

To evaluate performance of the current approach against R&D project management dataset, a query set of twenty seven SPARQL queries have been listed out considering various user requirements that covers various SPARQL query constructs. Table 2 gives user queries and their SPARQL representation. Table 3 shows the mapping of various SPARQL queries constructs to user queries.

Table 2. User queries and their SPARQL representation.

Q1. List R&D projects order by value of grant-in-aid
PREFIX proj: <http://www.drdo.org/Project#> SELECT ?p ?cost WHERE { ?p a proj:Project; proj:Cost ?cost} order by DESC(?cost)
Q2. List institutions order by funds received in the current year
PREFIX proj: <http://www.drdo.org/Project#> PREFIX org: <http://www.drdo.org/Organization#> SELECT ?org (SUM(?cost) AS ?amt) WHERE { ?p proj:Cost ?cost; proj:sponsored_to ?org} GROUP BY ?org Order by DESC(?amt)
Q3. List top ten institutions based on total funds received in the last three years
PREFIX proj: <http://www.drdo.org/Project#> PREFIX org: <http://www.drdo.org/Organization#> SELECT ?o (SUM(?c) as ?tot_fund) WHERE {?p proj:sponsored_to ?o; proj:Cost ?c.} GROUP BY ?o order by DESC(?tot_fund) LIMIT 10
Q4. List total number of projects sanctioned group by subject
PREFIX proj: <http://www.drdo.org/Project#> SELECT ?sub (COUNT(?p) as ?no_of_proj) WHERE { ?p a proj:Project; proj:Area ?sub} GROUP BY ?sub
Q5. List number of projects sanctioned group by state
PREFIX proj: <http://www.drdo.org/Project#> PREFIX org: <http://www.drdo.org/Organization#> SELECT ?st (COUNT(?p) as ?no_of_proj) WHERE {?p proj:sponsored_to ?o. ?o org:State ?st} GROUP BY ?st
Q6. Find number of projects granted and total funds released to a specific PI
PREFIX per: <http://www.drdo.org/Person#> PREFIX proj: <http://www.drdo.org/Project#> select (COUNT(?pro) as ?no_of_proj) (SUM(?c) as ?tot_fund) where {per:R_Thaokar per:is_responsible_of ?pro. ?pro proj:Cost ?c}
Q7. Funds released to South India / North India
PREFIX proj: <http://www.drdo.org/Project#>

PREFIX org: <http://www.drdo.org/Organization#> PREFIX reg: <http://www.drdo.org/Region#> SELECT ?st (SUM(?c) as ?tot_fund) WHERE {?p proj:sponsored_to ?o; proj:Cost ?c. ?o org:State ?st FILTER EXISTS{?r a reg:South; reg:name ?st}} GROUP BY ?st
Q8. Find total amount granted under particular scheme in a particular year
PREFIX org: <http://www.drdo.org/Organization#> PREFIX proj: <http://www.drdo.org/Project#> Select (SUM(?c) as ?tot_fund) where {?p proj:Scheme_Short_Name ?s FILTER(?s="BRNS"). ?p proj:Cost ?c}
Q9. Total funds released to a specific state
PREFIX org: <http://www.drdo.org/Organization#> PREFIX proj: <http://www.drdo.org/Project#> Select (SUM(?c) as ?tot_fund) where {?o org:holds ?p. ?p proj:Cost ?c. FILTER EXISTS {?o org:State "Kerala"}}
Q10. List out projects grants above one crore.
PREFIX proj: <http://www.drdo.org/Project#> Select ?p where {?p proj:Cost ?c FILTER (?c>10000000.00)}
Q11. List projects granted to either Universities or National_Laboratories
PREFIX proj: <http://www.drdo.org/Project#> PREFIX org: <http://www.drdo.org/Organization#> Select ?p where {{?o a org:National_Laboratory; org:holds ?p} UNION {?o a org:University; org:holds ?p}}
Q12. List projects granted to both Govt. and Autonomous institutions
PREFIX proj: <http://www.drdo.org/Project#> PREFIX org: <http://www.drdo.org/Organization#> Select ?p where {?o a org:Autonomous; a org:University; org:holds ?p}
Q13. List out projects grants between 20 to 50 lakhs
PREFIX proj: <http://www.drdo.org/Project#> Select ?p where {?p proj:Cost ?c FILTER (?c>=2000000.00&&?c<=5000000.00)}
Q14. List out projects granted between 2010 –2011 years
PREFIX proj: <http://www.drdo.org/Project#> Select ?p where {?p proj:Year "2010-11"}
Q15. Number of projects granted to female PI's
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> Select (COUNT(?p) as ?no_of_proj) where {?pi per:is_responsible_of ?p; per:Gender "F"}
Q16. Find total amount granted in particular year
PREFIX proj: <http://www.drdo.org/Project#> Select (SUM(?c) as ?tot_amt) where {?p proj:Cost ?c; proj:Year "2010-11"}
Q17. Find weather a specific state received any grants or not
PREFIX org: <http://www.drdo.org/Organization#> ASK {?o org:holds ?p; org:State "Kerala"}
Q18. Generate a graph of total funds received by various states
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#> PREFIX org: <http://www.drdo.org/Organization#> PREFIX proj: <http://www.drdo.org/Project#> CONSTRUCT { ?st proj:tot_amt ?tot_fund } where { {SELECT (SUM(?c) as ?tot_fund) WHERE {?o org:holds ?p; org:State ?st. ?p proj:Cost ?c} GROUP BY ?st}}
Q19. Export projects of grant more than one crore
PREFIX proj: <http://www.drdo.org/Project#> CONSTRUCT {?p proj:Cost1 ?c}

where {?p proj:Cost ?c FILTER (?c>10000000.00)}
Q20. List out PI and patents derived out of funded projects
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> select ?pi ?pub {?pi per:is_responsible_of ?p. OPTIONAL {?p proj:publications ?pub}}
Q21. List out all PI and publications greater than two
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> select ?pi ?pub {?pi per:is_responsible_of ?p. OPTIONAL {?p proj:no_of_publications ?pub. FILTER(?pub>2.0)}}
Q22. List out all publications derived out of funded projects
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> select ?p ?pub {?p proj:publications ?pub}
Q23. List out projects which have no publication
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> select ?p {?p proj:publications ?pub FILTER(!bound(?pub))}
Q24. List out projects which has got publications and patents from funded projects
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> select ?p {?p proj:publications ?pub; proj:patents ?pet.}
Q25. List projects where PI is JRF itself
PREFIX proj: <http://www.drdo.org/Project#> PREFIX per: <http://www.drdo.org/Person#> select ?pi {?pe per:Full_Name ?jn; per:Full_Name ?pi FILTER(?jn=?pi)}
Q26. List out all projects order by number of publications
PREFIX proj: <http://www.drdo.org/Project#> select ?p {?p proj:no_of_publications ?pub} order by DESC(?pub)
Q27. List out all PI patents and their publications
PREFIX proj: <http://www.drdo.org/Project#> select ?pi ?pub {?p proj:has_PI ?pi. OPTIONAL{?p proj:publications ?pub}. OPTIONAL{?p proj: patents ?pet}}

For R&D project management dataset, the evaluation of the current methodology is done by measuring overall and individual query performance. The individual query performance is measured by QPT in milliseconds. The overall query performance is measured by AM and GM of QPT of all queries from the query set. It is observed that the predicate based partitioning and indexing method improves the overall query performance. AM metric are shown in Fig. 12. GM metric are shown in Fig. 13.

Further series of tests are done to evaluate query performance. The queries with multiple pattern and filters are taking slightly more time (in milliseconds). 24 query patterns out of twenty seven are showing better performance using the current methodology with R&D project management dataset. Results are presented in the graph depending on the execution time. Fig. 14a shows individual query performance of queries of QPT less than thousand milliseconds. Fig. 14b shows individual query performance of queries of QPT more than thousand milliseconds.

DOI:10.4186/ej.2015.19.5.139

Table 3. Mapping of various SPARQL queries constructs to user queries.

	Order By	Aggregate Functions	LIMIT	Group By	FILTER	Self-Join	Union	Multiple pattern	Negation	Un bounded	Between	Optional	Construct	Ask
Q1	*							*						
Q2		*		*				*						
Q3	*	*	*	*				*						
Q4		*		*				*						
Q5		*		*				*						
Q6		*						*						
Q7		*		*	*			*						
Q8		*			*			*						
Q9		*			*			*						
Q10					*									
Q11							*	*						
Q12								*						
Q13					*						*			
Q14											*			
Q15		*						*						
Q16		*						*						
Q17								*						*
Q18		*		*				*					*	
Q19					*								*	
Q20												*		
Q21					*							*		
Q22										*				
Q23					*				*					
Q24								*						
Q25					*	*		*						
Q26	*													
Q27					*							*		

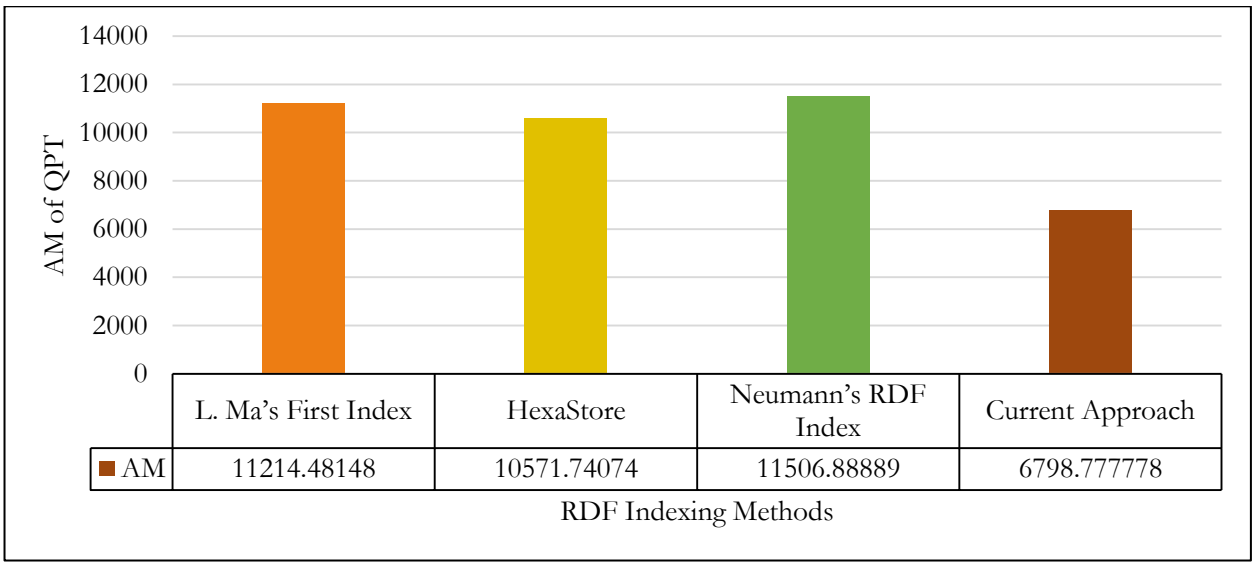


Fig. 12. Arithmetic mean of QPT with respect to various RDF indexing methods on R&D project management dataset.

DOI:10.4186/ej.2015.19.5.139

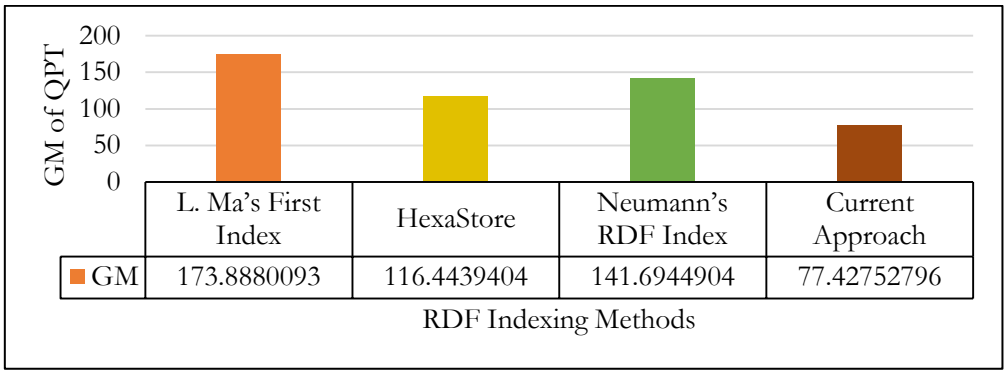
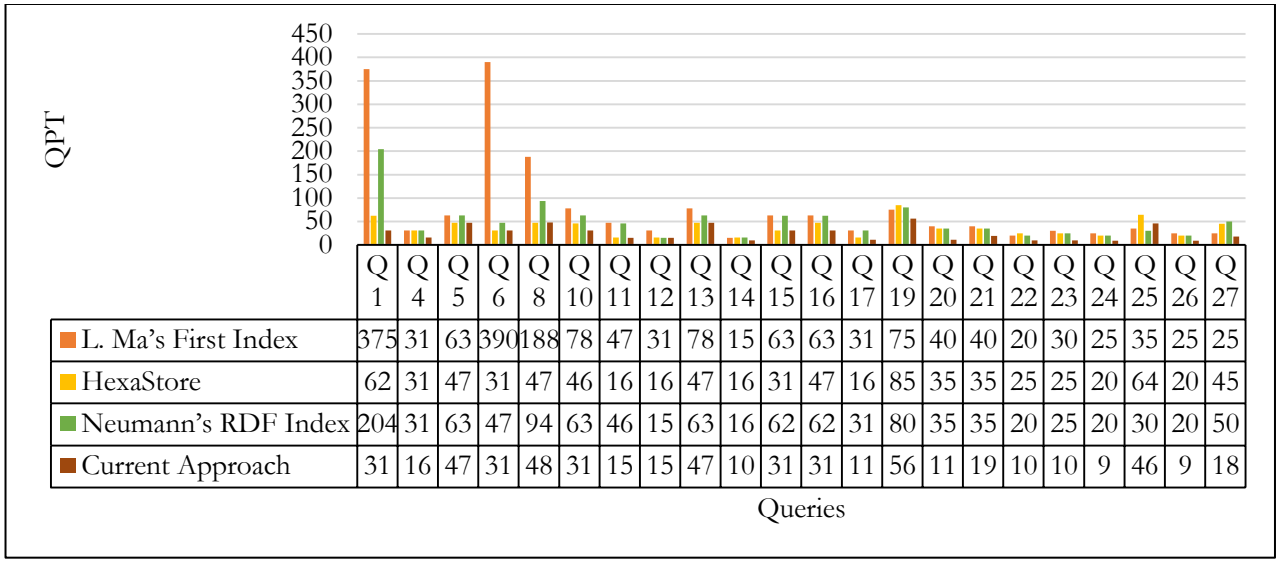
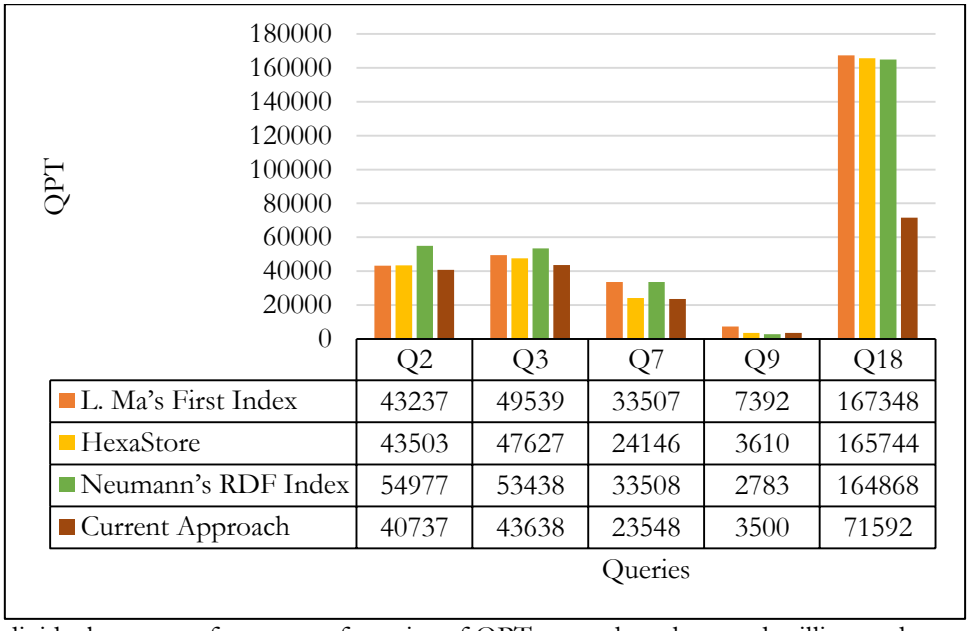


Fig. 13. Geometric mean of QPT with respect to various RDF indexing methods on R&D project management dataset.



	Q1	Q4	Q5	Q6	Q8	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q19	Q20	Q21	Q22	Q23	Q24	Q25	Q26	Q27
L. Ma's First Index	375	31	63	390	188	78	47	31	78	15	63	63	31	75	40	40	20	30	25	35	25	25
HexaStore	62	31	47	31	47	46	16	16	47	16	31	47	16	85	35	35	25	25	20	64	20	45
Neumann's RDF Index	204	31	63	47	94	63	46	15	63	16	62	62	31	80	35	35	20	25	20	30	20	50
Current Approach	31	16	47	31	48	31	15	15	47	10	31	31	11	56	11	19	10	10	9	46	9	18

Fig. 14a. Individual query performance of queries of QPT less than thousand milliseconds.



	Q2	Q3	Q7	Q9	Q18
L. Ma's First Index	43237	49539	33507	7392	167348
HexaStore	43503	47627	24146	3610	165744
Neumann's RDF Index	54977	53438	33508	2783	164868
Current Approach	40737	43638	23548	3500	71592

Fig. 14b. Individual query performance of queries of QPT more than thousand milliseconds.

5. Conclusion and future work

The current research work has proposed predicate centric partitioning and multiple RDF indexing for database triple store. The evaluation of the method is done by using various query performance metrics. The query performance is evaluated by implementing the current methodology on various datasets and compared the results with various RDF indexing methods. The current work has considered BSBM, SP2 benchmark datasets and DBPedia real dataset as a test bed. Further the partitioning and indexing method is applied to R&D project management dataset. The results indicate that, the overall query performance is improved when compared to other indexing techniques and shows very effective results for R&D project management dataset. The effectiveness of the partitioning and multiple RDF indexing method is compared with L. Ma' first indexing, HexaStore and Neumann's RDF indexing approaches. The test results indicate, the proposed method improves overall query performance. It is observed that 24 queries out of query set of 27 queries, has improved query performance. The present research is hoped to form a reference baseline for further research work on the effectiveness of the queries with filter, multiple pattern and unknown predicate.

References

[1] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, pp. 199−220, 1993.

[2] J. Martinez-Gil and J. F. Aldana-Montes, "KnoE: A web mining tool to validate previously discovered semantic correspondences," *Journal of Computer Science and Technology*, vol. 27, no. 6, pp. 1222–1232, Nov. 2012.

[3] M. Horridge, "What are owl ontologies?," in *A Practical Guide To Building OWL Ontologies Using Protégé 4 and CO-ODE Tools*, Edition 1.3. The University Of Manchester, Mar. 2001, pp. 10–12.

[4] M. K. Smith, C. Welty, and D. L. McGuinness. (2004). OWL Web Ontology Language Guide. [Online]. Available: http://www.w3.org/TR/owl-guide/

[5] D. Brickley and R.V. Guha. (2004). RDF Vocabulary Description Language 1.0: RDF Schema. [Online]. Available: http://www.w3.org/TR/rdf-schema/

[6] M. Frank and M. Eric. (2004). RDF Primer. [Online]. Available: http://www.w3.org/TR/rdf-primer/

[7] A. Sunitha and G. Suresh Babu, "Survey on ontology languages," *CiiT International Journal of Artificial Intelligent Systems and Machine Learning*, vol. 5, 2013, pp. 440–445.

[8] B. DuCharme, "The semantic web, RDF, and linked data (and SPARQL)," in *Learning SPARQL*, 2nd ed. USA: O'Reilly Media, 2013, pp. 19–45.

[9] D2R Server: Accessing Databases with SPARQL and as Linked Data. [Online]. Available: http://d2rq.org/d2r-server

[10] Mapping SQL Data to Linked Data Views. [Online]. Available: http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/VOSSQL2RDF

[11] B. Christian and S. Andreas, "The Berlin SPARQL benchmark," *International Journal on Semantic Web & Information Systems*, vol. 5, no. 2, pp. 1–24, 2009.

[12] AllegroGraph. [Online]. Available: http://www.franz.com/products/allegrograph/index.lhtml

[13] Jena TDB. [Online]. Available: http://jena.apache.org/documentation/tdb/

[14] Y. Chen, J. Ou, Y. Jiang, and X. Meng, "HStar-a semantic repository for large scale owl documents," in *Proc. 1st Asian Semantic Web Conference*, *Lecture Notes in Computer Science, Springer*, 2006, vol. 4185, pp. 415–428.

[15] Jena SDB. [Online]. Available: http://jena.apache.org/documentation/sdb/

[16] M. Chuck, *Oracle Database Semantic Technologies Developer's Guide, 11g Release 2 (11.2)*. Jun. 2013.

[17] Virtuoso. [online] Available http://virtuoso.openlinksw.com/dataspace/doc/dav/wiki/Main/

[18] S. Heymans, L. Ma, D. Anicic, Z. Ma, N. Steinmetz, Y. Pan, J. Mei, A. Fokoue, A. Kalyanpur, A. Kershenbaum, E. Schonberg, K. Srinivas, C. Feier, G. Hench, B. Wetzstein, and U. Keller, "Ontology reasoning with large data repositories," *Ontology Management: Semantic Web, Semantic Web Services and Business*. Springer, 2008, ch. 4.

[19] L. Al-Jadir, C. Parent, and S. Spaccapietra, "Reasoning with large ontologies stored in relational databases: the ontomind approach," *Journal of Data & Knowledge Engineering*, vol. 69, no. 11, pp. 1158–1180, Nov. 2010.

[20] C. Supasate and I. Chalermek, "An analysis of deductive-query processing approaches for logic macroprograms in wireless sensor networks," *Engineering Journal*, vol. 16, no. 4, pp. 47–61, Jul. 2012.
[21] X. Wang, W. Shuyi, D. Pufeng, and F. Zhiyong, "Storing and indexing RDF data in a column-oriented DBMS," in *Proc. 2nd International Workshop on Database Technology and Applications (DBTA), IEEE*, 2000, pp. 1–4.
[22] D. Beckett and J. Grant. Mapping Semantic Web Data with RDBMSes, SWAD-Europe Deliverable 10.2, 2003. [Online]. Available: http://www.w3.org/2001/sw/Europe/reports/scalable_rdbms_mapping_report/
[23] D. Kolas, I. Emmons, and M. Dean, "Efficient linked-list RDF indexing in parliament," in *Proc. The Fifth International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2009)*, pp. 17–32.
[24] G. H. L. Fletcher and W. B. Peter, "Scalable indexing of RDF graphs for efficient join processing," in *Proc. The 18th ACM Conference on Information and Knowledge Management*, Nov. 2–6, 2009, pp. 1513–1516.
[25] L. Ma, Z. Su, Y. Pan, L. Zhang, and T. Liu, "RStar: An RDF storage and query system for enterprise resource management," in *Proc. the Thirteenth ACM International Conference on Information and Knowledge Management*, ACM, 2004, pp.484–491.
[26] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: Sextuple indexing for semantic web data management," in *Proc. the VLDB Endowment*, Aug. 2008, vol. 1, no. 1, pp. 1008–1019.
[27] T. Neumann and G. Weikum, "xRDF3X:fast querying, high update rates, and consistency for RDF databases," in *Proc. the VLDB Endowment*, Sept. 2010, vol. 3, no. 1–2, pp. 256–263.
[28] Advantages of using table partitioning. [Online]. Available http://www.dbatodba.com/db2/db2-udb-v9/advantages-of-using-table-partitioning/
[29] S. Harris and A. Seaborne, (2013). SPARQL 1.1 Query Language. [Online]. Available: http://www.w3.org/TR/sparql11-query/
[30] D. Songyun, K. Anastasios, S. Kavitha, and U. Octavian, "Apples and oranges: A Comparison of RDF benchmarks and real RDF datasets," in *Proc. Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, New York, NY, USA, 2011, pp. 145–156.
[31] RDF Store Benchmarks with DBpedia. [Online]. Available: http://wifo5-03.informatik.uni-mannheim.de/benchmarks-200801/
[32] S. Michael, H. Thomas, L. Georg, and P. Christoph, (2008). SP2Bench: A SPARQL Performance Benchmark. [Online]. Available: http://arxiv.org/abs/0806.4627