ISSN:1991-8178

Australian Journal of Basic and Applied Sciences

Journal home page: www.ajbasweb.com

A Novel Method For Dynamic SPARQL Endpoint Generation In NoSQL Databases

[1]Kiran V K and [2]Dr G Sudha Sadasivam

[1]Assistant Professor Dept. of Computer Science and Engineering PSG College of Technology Coimbatore, India
[2]Professor Dept.Computer Science & Engineering PSG College of Technology Coimbatore, India

ARTICLE INFO

Article history:
Received 12 November 2014
Received in revised form 26 December 2014
Accepted 29 January 2015
Available online 10 February 2015

Keywords:
SPARQL, RDF, NoSQL, HBase, Semantic Web

ABSTRACT

SPARQL endpoint exposes the data that it holds as RDF such that SPARQL queries can be run on them. This is an important requirement for any knowledge base if it has to expose its data in a machine understandable format. Nowadays NoSQL databases are increasingly used to store data along with RDBMS. However, NoSQL databases are not machine friendly as majority of them doesn't expose a SPARQL endpoint and hence are not semantic web ready. This paper proposes a novel approach for exposing data in a column oriented store like HBase in RDF.

© 2015 AENSI Publisher All rights reserved.

To Cite This Article: Kiran V K and Dr G Sudha Sadasivam., A Novel Method For Dynamic SPARQL Endpoint Generation In NoSQL Databases. ***Aust. J. Basic & Appl. Sci.,*** *9(6): 65-71, 2015*

INTRODUCTION

World Wide Web (WWW) is basically a collection of hyperlinked documents. The links between documents are basically followed by a human and the process is known as browsing. However, if a human being tries to search through the entire set of web documents for a particular entity by following these links he may end up by spending his entire lifetime for it without finding the relevant contents!

Here comes the importance of making the machines (information processing systems) understand what each web document talks about.RDF (Resource Description Framework) attempts to solve this problem by providing a lightweight web technology friendly method of metadata representation.RDF talks about "things" and "relationship between things"(John Domingue, et al., 2011; http://jena.sourceforge.net/tutorial/RDF_API/). RDF facilitates naming of relationship between the things and to check whether any two things are related to each other by following the named links. The things and relationships are resources which can be uniquely identified across the web using URI (Uniform Resource Identifier).Each document on the web can have its corresponding RDF document that gives a machine understandable representation of it.

For any data store that is liable to change at a drastic rate there are two schemas (predicate representing each data store) (Kiran, V.K., Dr.G. Sudha Sadasivam, 2013).One is the static schema that gives the schema of a warehoused data store, the other is dynamic schema that represents a predicate with varying number of variables that gets added and deleted as and when new data enters the data store. This paper proposes a SPARQL endpoint generation method for both the classes of data stores addressed previously.

RDF:

Resource Description Framework (RDF) is used to give a digital representation to any real world entity (John Domingue, et al., 2011; http://jena.sourceforge.net/tutorial/RDF_API/). These real world entities are known as ***resources*** in RDF. Every real world entity will have certain attributes whose values will distinguish it from other entities that belong to the same group. These attributes are known as ***properties*** and value taken by the property is known as ***property value***. The properties and property values are again treated as things.RDF uses features provided by XML to represent resources, properties and relationship. The "***thing***" its "***property***" and the "***property value***" together is known as ***RDF triple***.

For instance suppose that we need to show that a person named Ram is 30 years old using RDF.A possible representation can be like the one shown in Fig.1:

Corresponding Author: Kiran V K, Assistant Professor Dept. of Computer Science and Engineering PSG College of Technology Coimbatore, India
E-mail: kiranvk2011@gmail.com

```
<rdf:RDF
  xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
  xmlns:details='http://example/info#' >
  <rdf:Description rdf:about='http://example/persons/Ram'>
    <details:name>Ram</details:name>
    <details:age>30</details:age>
  </rdf:Description>
</rdf:RDF>
```

Fig. 1: Sample RDF representation.

Here the properties name, age are represented as ***local names*** of ***namespace prefix*** details. This is done to make the entries more readable.

Web Ontology Language (OWL) is a vocabulary extension to RDF (John Domingue et al., 2011; http://www.w3.org/TR/owl-features/). The triples in RDF can be given extended meaning using vocabulary provided by OWL. One such feature provided by OWL used in this paper is the notion of ***classes*** provided by OWL. Classes group resources having similar characteristics together.

Hbase:

HBase (http://hbase.apache.org/book/) is a scalable, distributed, column-oriented dynamic-schema database for structured data.. HBase is a subproject of the Apache Software Foundation's Hadoop project. In HBase there is no concept of database. HBase data is modeled as a multidimensional map in which values (the table *cells*) are indexed by four keys:

value = Map(TableName, RowKey, ColumnKey, Timestamp).

The row key is the primary key of the table and is typically a string. Rows are sorted by row key in lexicographic order.. Column Key is a combination of column family name and column name. Each column family can have arbitrary no: of columns but the column families are fixed during table creation. Hence applications can dynamically create new columns on the fly.

Table I: Person Table With Two Column Families (For Illustration Only)

RowKey	TimeStamp	Column Family	
		name	age
001	T1	name : first Ram	
001	T2	name : last Shankar	
001	T3		age : num 24
002	T4	name : first Shyam	
002	T5	name : last Mohan	

Table I shows a sample HBase table's conceptual view .Here it is assumed that for row identified by key "001" there are three different versions created at different points in time whose timestamps corresponds to T1, T2, T3, similar for the case of row with key "002".As shown, certain cells can be empty in HBase, which means all columns of a column family need not carry data.

Section IV describes the proposed methodology for exposing the data in HBase as RDF.

Proposed Methodology:

The mapping steps applied for creating RDF representation are (Kavitha, C., et al., 2011; Oliver Cure, et al., 2012; Madhav Krishna, 2006):

- An OWL class corresponding to each table in data store is constructed.
- Each row in HBase table is converted to a resource.
- Each column family, column pair is converted as a property of the resource with property name of the format "***column family_ column***".
- The value of a particular column will become the property value of the column family, column pair.

Hence the RDF representation of HBase table "Person" given in table I is as given in Fig.2.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:base="http://sample.com#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
  <rdf:Description rdf:about="http://sample.com#001">
    <base:age_num>24</base:age_num>
    <base:name_last>Shankar</base:name_last>
    <base:name_first>Ram</base:name_first>
    <rdf:type rdf:resource="http://sample.com#person"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://sample.com#002">
    <base:name_last>Mohan</base:name_last>
    <base:name_first>Shyam</base:name_first>
    <rdf:type rdf:resource="http://sample.com#person"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://sample.com#person">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
  </rdf:Description>
</rdf:RDF>
```

Fig. 2: RDF Representation of Table I.

The table name "Person" is converted to an OWL class with URI "http://sample.com#person". Here the namespace "http://sample.com "is denoted by the namespace prefix "base". Two resources of type "Person" is created for the two rows "001" and "002".It has to be noted that the empty ***column family_ column*** properties does not include with each resource. Therefore the proposed method takes into consideration the data model supported by HBase.

The above defined method is first of its kind for exposing data in NoSQL data store as RDF. If implemented over the stores it will give a SPARQL endpoint (John Domingue, et al., 2011; http://www.w3.org/TR/rdf-sparql-query/) over which queries can be issued. For implementing the mapping steps Apache Jena was used which would take HBase fields and convert to its equivalent RDF.

Proposed Architecture:

Architecture is defined by the recommended practice as the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution (ANSI/IEEE Std 1471-2000). The proposed methodology can be implemented using ***pipe and filters*** software architecture.

A. Pipes and Filters:

Pipes and Filters architecture focuses on developing a solution that is composed of multiple data processing elements called ***filters*** which works together to produce the required transformation on the input data for the system. The entities that allow flow of unprocessed/processed data among the filters are known as ***pipes***.Fig.3 shows the pipes and filters architecture for the proposed methodology. The input source is HBase table which consists of millions of rows maybe increasing or decreasing in number dynamically. So the data in HBase is taken as a stream input by the proposed system. This is first passed through ***Filter1*** which converts the data in HBase to RDF as per the proposed methodology outlined in section IV. ***Filter 2*** is invoked by ***Filter 1*** when a RDF model with sufficient number of entities is created by ***Filter1.***

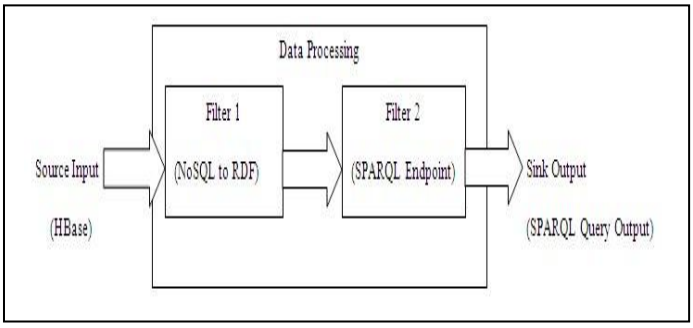

Fig. 3: Architecture of Proposed Solution

Filter 2 then allows issuing SPARQL queries against the model and gives the result of SPARQL query as output.

Applications:

Converting unstructured data to RDF in a column-oriented store like HBase would be helpful when we try to semantically integrate multiple HBase tables with the help of OWL ontology (Agustina Buccella et al., 2003). Ontology defines the terms used to describe and represent an area of knowledge. Ontology is used by people, databases, and applications that need to share domain information (a domain is just a specific subject area or area of knowledge, like medicine, tool manufacturing, real estate, automobile repair, financial management, etc.). Ontology includes computer-usable definitions of basic concepts in the domain and the relationships among them. They encode knowledge in a domain and also knowledge that spans across domains. In this way, they make that knowledge reusable.

The Web Ontology Language (OWL 2) (John Domingue et al., 2011; http://www.w3.org/TR/owl-features/) is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontology provides classes, properties, individuals, and data values and is stored as Semantic Web documents. OWL 2 ontology can be used along with information written in RDF, and OWL 2 ontology themselves are primarily exchanged as RDF documents.

So if we have a global OWL ontology that relates between each RDF resources from different HBase tables, then globally across the all HBase tables we can issue a query ie a SPARQL query that will fetch relevant results from all the tables.

Fig.4 shows a possible application of the proposed solution applied over multiple HBase tables. A ***scanner*** (Kiran, V.K., G. Dr. Sudha Sadasivam, 2013) is created for each HBase table which goes through all HBase entries. So, if there are 'n' HBase tables, we have to initialize 'n' separate scanner threads. Each scanner converts the unstructured HBase data to RDF "on-the-fly" (http://d2rq.org/d2r-server) by applying the transformation of ***filter1*** of fig.3.Here "on-the-fly" refers to the fact that the RDF form is created dynamically whenever HBase entries are found and then buffered and kept in memory for SPARQL query. Hence the RDF representation generated is called virtual RDF as it is a subgraph of a large hypothetical RDF graph. The threads scan a predefined number of entities, convert then to RDF and gives to RDF merger module, which creates a dynamic RDF model that can be used for SPARQL query.

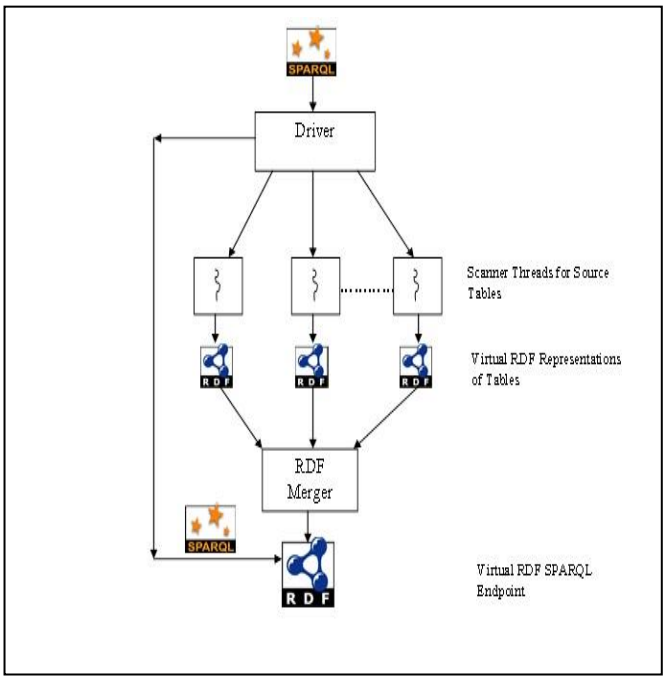

Fig. 4: Querying Across Multiple Tables

Here the RDF merger module acts as ***filter 2*** shown in fig.3.Therefore, the architecture proposed in fig.3, is slightly modified so that there are multiple agents performing the operation of ***filter1*** and there is a single agent performing operation of ***filter2***.Agent here refers to autonomous entities that can perform an operation.

The primary duty of the driver module shown in Fig.4 is to instantiate and co-ordinate scanner threads .Once the scanner threads have dumped the RDF

representation after a run and waits for RDF merger module to merge the dump into a consistent RDF model, the driver module has to record the current scan states of the scanners. This will allow the scanner threads to proceed further in converting HBase data to RDF. Also, the driver module accepts the SPARQL query from the client and repeatedly applies it over the virtual RDF model chunks created by RDF merger as a particular HBase table may require multiple stages to be entirely converted to RDF.

Results:

The SPARQL endpoint generator was run over a HBase cluster of eight 4GB RAM, 1.86GHZ Intel Xeon multiprocessor machines. Random records similar to those shown in Table I was inserted into HBase. The data size in the HBase cluster amounted to 4.33 TB. Apart from random data, tweet data from twitter database was also used to populate HBase. This approach was taken to achieve a dataset of considerable size within a small period of time.A node outside the HBase cluster was taken as client node. The client node is a modest PC with 2GB RAM and 1GHZ Intel Pentium processor. The client node would issue ***scanner*** for reading HBase table contents and would use Apache Jena to convert HBase contents to RDF.Fig.5 shows the topology of nodes used for analysis of proposed approach.

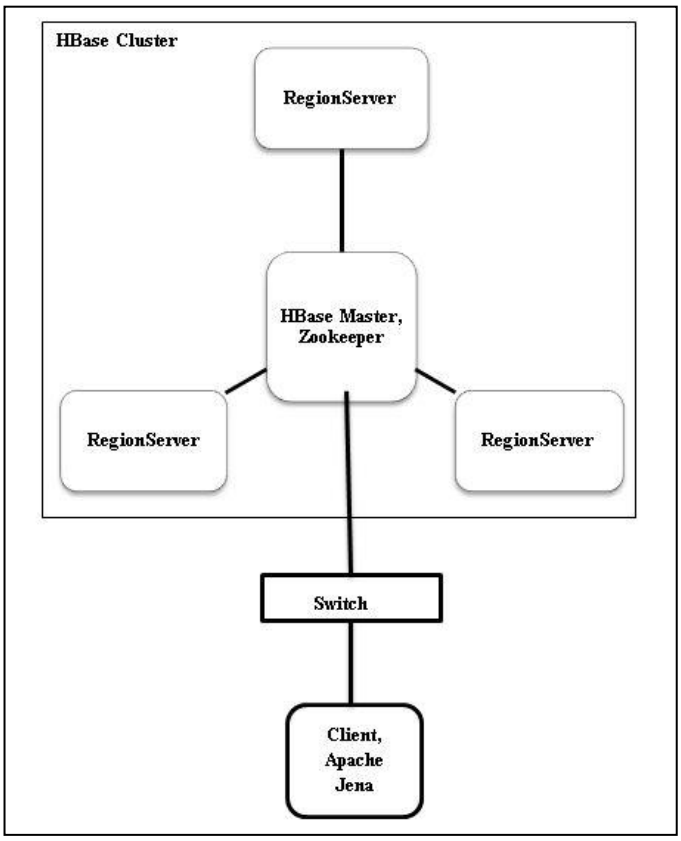

Fig. 5: Topology of Nodes

It was found that the time taken to convert HBase entries into RDF varies almost linearly with respect to the number of entries in HBase table as shown in Fig.6.From Table II it is evident that when number of records to be processed in HBase goes over a few million the conversion time gets to the order of hours.

Conclusions and future works:

The proposed method for exposing SPARQL endpoint for HBase is first of its kind method of making NoSQL data stores semantic web friendly. The proposed method greatly helps to fill the gap of ontology based integration tools in the domain of NoSQL databases. An elaborate integration framework over multiple NoSQL data stores from various vendors can be a future development over this proposed methodology. Methods can also be proposed to extract ontology from NoSQL stores quickly.

Table II: Hbase To Rdf Conversion Time

Individuals (x 10^3)	Time Taken in s
1	0.49
5	2.34
10	3.88
50	16.61
100	26.77
200	23.37
500	74.08
700	182.64
1000	265.93
5000	1319.22
10000	2642.87

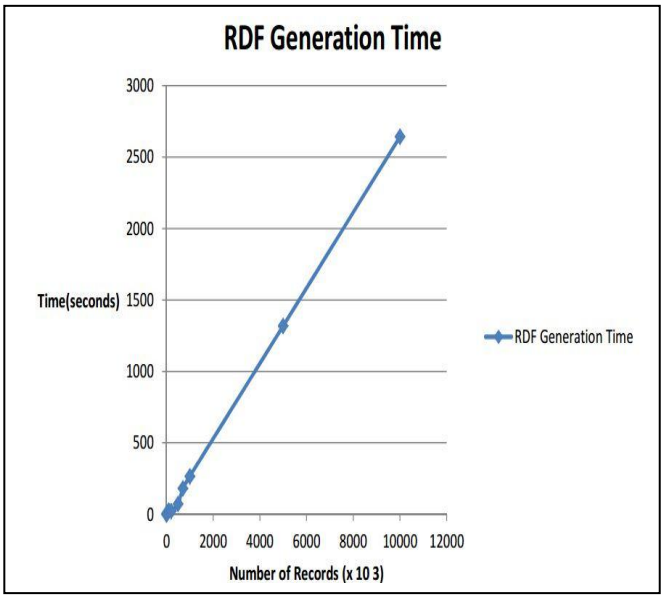

Fig. 5: RDF Generation Time versus Number of Individuals

ACKNOWLEDGEMENT

We would like to sincerely thank Mr.Chidambaram Kollengode, Director, Cloud Computing (Big Data Hadoop Platform and Analytics), Nokia for infrastructure and guidance given for the work. We would also like to thank Dr.R.Rudramoorthy, principal, PSG College of Technology for providing us with necessary facilities for the work.

REFERENCES

John Domingue, Dieter Fensel, James A. Hendler, 2011. "Handbook of Semantic Web Technologies", Springer.

http://jena.sourceforge.net/tutorial/RDF_API/, gives a good introduction to RDF and Apache Jena.

Kiran, V.K., Dr.G. Sudha Sadasivam, 2013. "A Schema Generation Approach for Column Oriented NoSQL Data Stores", Proceedings of PSG-ACM National Conference on Intelligent Computing (NCIC-2013), 26.-27.

http://www.w3.org/TR/owl-features/, gives an overview of OWL specification.

http://hbase.apache.org/book/architecture.html#arch.overview, talks about HBase storage architecture.

Kavitha, C., G. Sudha Sadasivam, Sangeetha N. Shenoy, 2011. "Ontology Based Semantic Integration of Heterogeneous Databases", European Journal of Scientific Research, 64(1): 115-122.

Oliver Cure, Myriam Lamolle, Chan Le Duc, Fadhela Kerdjoudj, 2012. "On The Potential Integration of an Ontology-Based Data Access Approach in NoSQL Stores", Third International Conference on Emerging Intelligent Data and Web Technologies, pp: 166-173.

Madhav Krishna, 2006. "Retaining Semantics in Relational Databases by Mapping them to RDF", Proceedings of IEEE/WIC/ACM WI-IAT.

http://www.w3.org/TR/rdf-sparql-query/, gives complete specification of SPARQL standard.

ANSI/IEEE Std 1471-2000. Recommended Practice for Architectural Description of Software-Intensive Systems.

Agustina Buccella, Alejandra Cechich, Nieves R. Brisaboa, 2003. "An Ontology Approach to Data Integration", JCS&T 3(2): 62-68.

Kiran, V.K., G. Dr. Sudha Sadasivam, 2013. "Generalized Framework for Using Multiple NoSQL

Databases" ,ICCBDA '13, Proceedings of the International Conference on Cloud and Big Data Analytics, pp: 4.

http://d2rq.org/d2r-server, is the D2RQ tool website which provides SPARQL endpoint for RDBMS.

ANSI/IEEE Std 1471-2000, Recommended Practice for Architectural Description of Software-Intensive Systems.

Agustina Buccella, Alejandra Cechich, Nieves R. Brisaboa, 2003. "An Ontology Approach to Data Integration", JCS&T 3(2): 62-68.

Kiran, V.K., Dr. G. Sudha Sadasivam, 2013."Generalized Framework for Using Multiple NoSQL Databases" ,ICCBDA '13, Proceedings of the International Conference on Cloud and Big Data Analytics, pp: 4.

http://d2rq.org/d2r-server, is the D2RQ tool website which provides SPARQL endpoint for RDBMS.