A SURVEY ON MANAGING BIG RDF GRAPH BY USING MULTIPLE INDEXING APPROACHES IN BIG DATA

[1]Foram M. Gohel,[2]Ashutosh A. Abhangi
[1]ME Scholar,[2]Assistant Professor
[1 2]Computer Engineering Department,
[1 2]Noble Group of Institution, Junagadh, Gujarat.

Abstract : Management of large scale RDF graph is very challenging task. It is not easy to access and manage large-scale, million-node (big) RDF graphs. A possible solution to this problem is require Map-Reduced based algorithms and techniques, in using semantic web. So RDF data management done more efficiently. From the relational database perspective, efficiency and scalability of RDF data model are derived from triplet model easily. /so, in this survey we describe the different types of approaches by using vertical-partitioning with triple nature of RDF. By using these different approaches we analyze that using vertical-partitioning in RDF triple nature, we can reduce time as well as storage also.

IndexTerms - RDF graphs, Map-Reduce, Semantic web, Vertical-Partitioning, Hexastore, etc…..

I. INTRODUCTION

1.1 Big data:

Big data is a covering all term for the non-traditional strategies and technologies needed to collect, organize and process from large datasets. While the problem of working with data that extends the computing power or storage of a single computer is not new, the ubiquity, scale, and value of this type of computing has greatly become larger in modernistic years. The Word Big Data Defines as Extremely huge data sets, which may be analyzed computationally to reveal patterns, trends, and associations, especially relation to human behavior and interactions. The term big data applies to information that cannot be processed or analyzed using traditional processes or tools[1]. Increasingly, organizations today we are facing more and more big data challenges. Big data challenges are indexing, shorting, search, manage, data creation, sharing, transfer updating and information privacy.

A huge repository of terabytes of data is generated each day from modern informationsystems and digital technologies such as Internet of Things and cloud computing[2]. Analysis of these massive data requires a lot of efforts at multiple levels to extract knowledge fordecision making. Therefore, big data analysis is a current area of research and development. Hence, for managing these terabytes of data the concept of RDF is arrived.

1.2 RDF Graph:

The Resource Description Framework (RDF)[5] is a general framework for how to describe any Internet resource such as a Web site and its content. An RDF description (such descriptions are often referred to as metadata, or ”data about data”) can include the authors of there source, date of creation or updating, the organization of the pages on a site (the sitemap),information that describes content in terms of audience or content rating[8], key words for search engine data collection, subject categories, and so forth. The Resource Description Framework will make it possible for everyone to share Website and other descriptions more easily and for software developers to build products that can use the metadata to provide better search engines and directories, to act as intelligent agents, and to give Web users more control of what they’re viewing.

The above example is called RDF graph or sometimes called an RDF triple. Of the two, triple is the most helpful term as it describes the breaking of the statement into its three constituent parts: the subject, predicate, and object of the statement.[4]

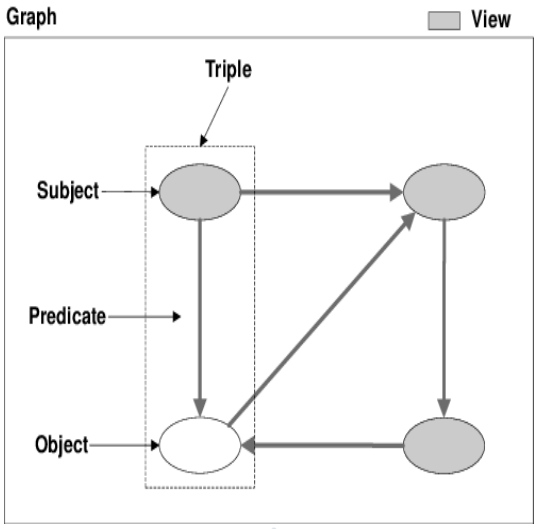

Fig 1: RDF Graph Model[9]

1.3 Map-Reduce Algorithm:

Map-Reduce are a parallel programming model in displayed by Google. The thought is unique from utilitarian programming dialect in big data. Map-Reduce parts the issue them handling into two phases (outline and lessen arrange). The guide organize expends are (critical, esteem) of sets and gatherings of yield in (key, esteem) matches too. The organize forms the yield of guide arrange with keys and yields the last outcome. Map-Reduce structure are simply requires to the software engineer giving guide and reduce (join) strategy. However, just if the undertaking can be preoccupied as tasks over (key, esteem) Map-Reduce is reasonable.[12]

The Map-Reduce algorithm the important tasks, namely Map and Reduce.

1. The Map task takes a set of data and converts it into another set of data, where individual elements are broken down into tuples key-value pairs.[24]

2. The Reduce task takes the output from the Map as an input and combines those data tuples key-value pairs into a smaller set of tuples.

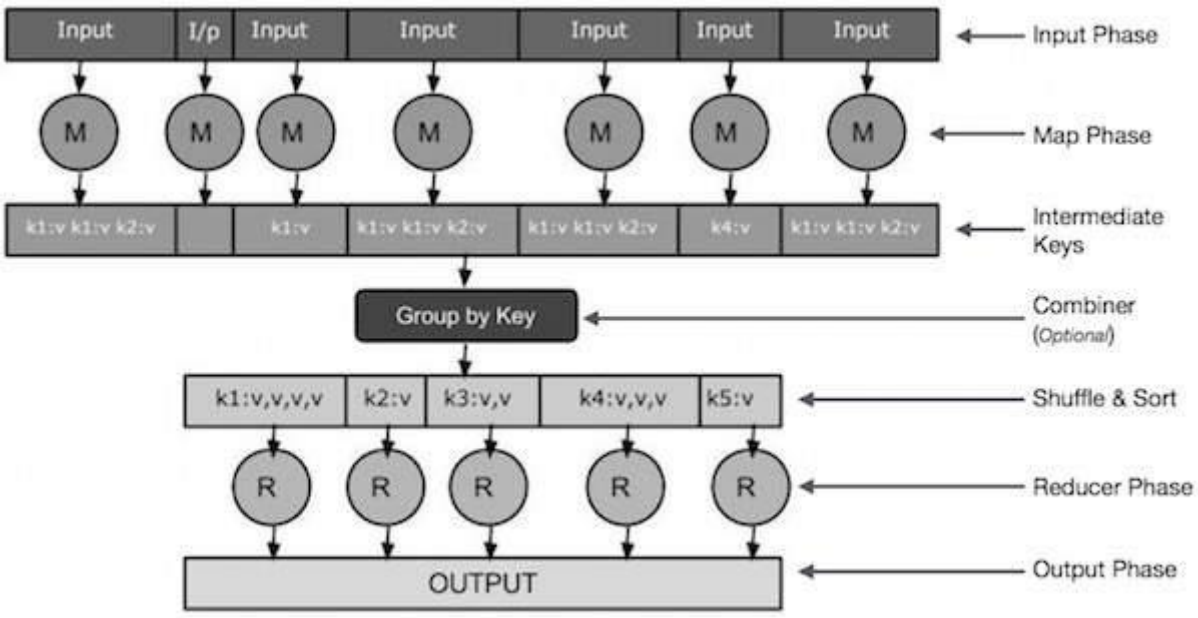

Fig 2: Map-Reduce Algorithm Phases[15]

1.4 Column-Oriented Partitioning:

There are four general types (most common categories) of NoSQL databases. Each of these categories has its own specific attributes and limitations. There is not a single solutions which is better than all the others, however there are some databases that are better to solve specific problems[19]. To clarify the NoSQL databases, let's discuss the most common categories:

- Key-value stores
- Column-oriented
- Graph
- Document oriented

II. Related Work:

2.1 Vertical-Partitioning:

In this scheme, a triples table is rewritten into n two-column tables, one table per property, where n is the number of unique properties in the data. This vertical-partitioning model is oriented towards answering queries in which the property resource is bound, or, otherwise, the search is limited to only a few properties. In fact, while Abadi et al. argue convincingly against the property-table solutions of the property-based two-column-table approach they introduce shares most of the disadvantages of those property-table solutions itself. In fact, the two-column tables used by are themselves a special variation of property tables too. Specifically, these two-column tables are akin to the multi-valued property tables introduced in; namely[26], the latter also store single properties with subject and object columns. In this respect, the most significant novelty of has been to integrate such two-column property tables into a column-oriented DBMS. Unfortunately, such an assumption is hard to be realized in a real-world setting. Thus, there is a need for scalable semantic web data management that will not depend on assumptions about the number of properties in the data or the (property-bound) nature of the executed queries.

2.2 Hexastore:

We took the vertical partitioning idea further, to its full logical conclusion. The result does not discriminate against any RDF elements; it treats subjects, properties and objects equally. Thus, each RDF element type deserves to have special index structure built around it. Moreover, every possible order of the importance or precedence of that three elements in an indexing scheme is materialized. The result amounts into a sextuple indexing scheme. We call a store that maintains six such indices a Hexastore. Each indexing structure in a Hexastore centres around one RDF element and it defines a prioritization between other two elements. Thus, the Hexastore equivalent of two-column property table can be either indexed by subject and allow for a list of multiple object entries per subject, or vice versa.

Hexastore does not take any prioritization of that three triple attributes for granted. RDF triples are not assumed to exist in a property-based universe. Hence, a Hexastore[26] creates not only property-headed divisions, but also subject-headed and object-headed ones.

In the former case, a given subject header s is associated to the property vector p(s) and to the object vector o(s); a list of associated objects $o_{p(s)}$ is appended to each entry in the property vector. Same as lists of associated properties $p_{o(s)}$ are appended to entries in the object vectors. Again, a list of properties $p_{sy(ox)}$ for object ox and subject sy in this object-headed indexing is identical to the property list $p_{ox(sy)}$ of the subject-headed indexing. Same as a list of subjects $s_{px(oy)}$ for property px and object oy in the object-headed indexing is identical to the subject list soy(px) featured in the property-headed indexing.

Putting it all together, the information for each triple (s, p, and o) in the data is represented in six ways, one for each possible prioritization of the three elements. We name these 3! = 6 prioritization ways by acronyms made up from the initials of the three RDF elements in the order of each prioritization. For example, the indexing that groups the data into subject-headed divisions with property vectors and lists of objects per vector is the spo indexing. Likewise, the osp indexing groups data into object-headed divisions of subject vectors with property lists per subject. In this framework, the column-oriented vertical partitioning scheme of, in which two-column property tables are sorted by subject, which can be seen as a special, simplified variant of our pso indexing. The six indexing schemes are then called spo, sop, pso, pos, osp, and ops.

Fig 3 represents a general example of spo indexing in a Hexastore where subject key s_i is associated to a sorted vector of n_i property keys, $\{p^i1, p^i2. \ . \ . \ p^ini\}$. Each property key p^i_j is, in its turn, linked to an associated sorted list of ki j object keys. These objects lists are accordingly shared with the pso indices. The same spo pattern will be repeated for every subjects in the Hexastore. Moreover, analogous patterns are materialized in other five indexing schemes.

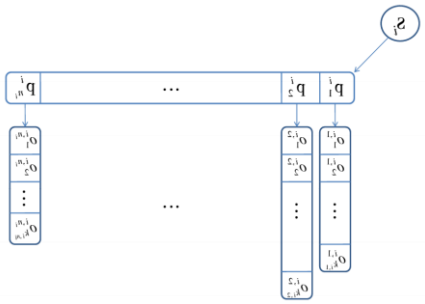

Fig 3 : spo indexing in a Hexastore[17]

2.3 COVP1 & COVP2:

The COVP method through our pso indexing. This indexing provides an enhancement compared to the purely vertical-partitioning approach of; namely, the pso indexing groups together multiple objects$\{o_1,o_2,...,o_n\}$related to the same subjects by a unique property p; on the other hand, in the vertical partitioning scheme, a separate $\langle s,o_i \rangle$ entry is made for each such object o_i in the two-column property table for property p. Moreover, we heed the suggestion in [5] that a second copy of each two-column property table can be created, sorted on the object column.

In fact, this suggestion was not followed in [5];instead, only unclusterd B+ tree indices were built on the object columns with the vertically-partitioned architecture implemented in Postgres. However, such tree indices were not built when the same vertically-partitioned architecture was implemented in a column-oriented DBMS, which in fact provides the top performance in [5]. Besides, the object column is not sorted in any of the approaches examined in [5]. Still, the suggestion of having a second copy of each two-column property table, sorted on object, is tantamount to having both a pso and a pos index in our scheme. Thus, for the sake of completeness, we also conduct experiments on such a two-index property-oriented store. In order to distinguish between the two, we call the single-index (i.e., pso) property-oriented store COVP1, and the two-index (i.e., pso and pos) store COVP2. The latter illustrates both the benefits of using a second index in comparison to the single-index COVP1[26], as well as its limitations in comparison to the six-index Hexastore.

III. RESULTS AND DISCUSSION

Table 1: Experimental Results of Comparison of the techniques

	COVP1	Hexastore	COVP2
0	500	200	200
1.00E+06	1200	450	500
2.00E+06	2500	852	1000
3.00E+06	3700	1007	1478
4.00E+06	4800	1348	1753
5.00E+06	5500	1563	2256
6.00E+06	7000	2145	2900

Here, We have some Experimental results for all three techniques for vertical partitioning method which is Hexastore, COVP1 and COVP2. From these results We found the graph which is given bellow that tells that the Hexastore technique is better in time searching and COVP1 technique is better for store the RDF graph in Big data world.

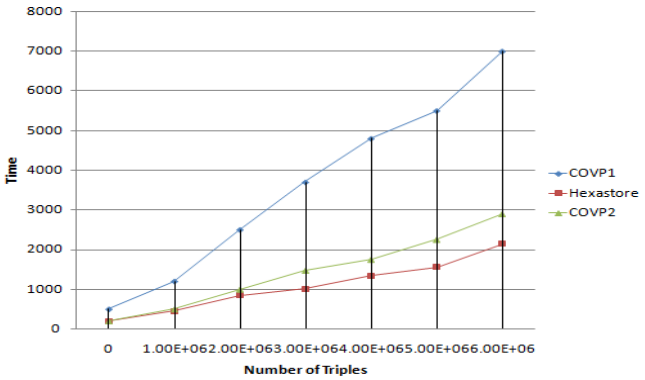

Fig 4 : Comparison Graph Of three different techniques

IV. FUTURE DIRECTION:

1. Uncertain Big RDF Graph Management

In real world scenario management of uncertain RDF graph is very challenging task. So as we survey in this work that if we using map-reduced based algorithms with NoSQL column-oriented partitioning distributed database then it's easily manage the uncertain RDF graph and required less storage.[14]

2. Indexing Big RDF Graphs

A possible solution to such problem is represented by indexing data structures Map-Reduce[14] model based algorithm which improve a query processing on big RDF graphs and tried to exploit the computation power and such complexity above. So using the indexing big RDF we can get better result.

V. CONCLUSION:

After this survey we can conclude that in managing RDF graph the column-oriented vertical partitioning methods and Hexastore method is very effective. For, better storage there is COVP1 (column-oriented vertical-partitioning) method is best and for better time the Hexastore technique is best.

VI. REFERENCES:

[1] https://en.wikipedia.org/wiki/Big data

[2] https://www.extrahop.com/company/blog/2016/it-operations-analytics-itoa-big- data/3

[3] https://searchmicroservices.techtarget.com/definition/Resource-Description-Framework-RDF

[4] http://www.linkeddatatools.com/introducing-rdf

[5] Data Science and BIG Data are not the same , "https://tijiwrotesomething.blogspot.com/2015/08/data-science-and-big-data- are-not-same.html"

[6] D. P. Acharjya, Kauser Ahmed P. "A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools.",(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, No. 2, 2016.

[7] Min Chen, Shiwen Mao, Yunhao Liu. "Big Data: A Survey.", Springer Sci- ence+Business Media New York 2014.

[8] https://www.tutorialspoint.com/map reduce/map reduce introduction.htm

[9] Sachin Arun Thanekar, K. Subrahmanyam, A. B. Bagwan ”Big Data and MapRe- duce Challenges, Opportunities and Trends.”, International Journal of Electrical and Computer Engineering (IJECE) Vol. 6, No. 6, December 2016, pp. 2911 2919 ISSN: 2088-8708, DOI: 10.11591/ijece.v6i6.10555

[10] Oguntimilehin A., Ademola E.O. ”A Review of Big Data Management, Benefits and Challenges.”, Vol. 5, No. 6 June 2014,ISSN 2079-8407 Journal of Emerging Trends in Computing and Information Sciences

[11] Fernando L. F. Almeida. ”Benefits, Challenges and Tools of Big Data Management.”, JOURNAL OF SYSTEMS INTEGRATION 2017.

[12] Hiba Alsghaier, Mohammed Akour, Issa Shehabat, Samah Aldiabat ”The Importance of Big Data Analytics in Business: A Case Study .”, American Journal of Software Engineering and Applications 2017; 6(4): 111-115 .

[13] Roberto De Virgilio. ”Smart RDF Data storage in Graph Databases.”, 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.

[14] Alfredo Cuzzocrea, Rajkumar Buyya, Vincenzo Passanisi, Giovanni Pilato. ”MapReduce-based Algorithms for Managing Big RDF Graphs: State-of-the-Art Analysis, Paradigms, and Future Directions.”, IEEE 2017

[15] Maxime Lefranois, Antoine Zimmermann, Noorani Bakerally. ”A SPARQL extension for generating RDF from heterogeneous formats.”, Springer 2017

[16] Federal University of Santa Catarina (UFSC) , ”A middleware for storing massive RDF graphs into NoSQL”, Researchgate-2017

[17] Brad Bebee, Daniel Choi, Ankit Gupta, Andi Gutmans, Ankesh Khandelwal, Yigit Kiran, Sainath Mallidi, Bruce McGaughy, Mike Personick, Karthik Rajan, Simone Rondelli, Alexander Ryazanov, Michael Schmidt, Kunal Sengupta, Bryan Thompson, Divij Vaidya, and Shawn Wang, ”Amazon Neptune: Graph Data Management in the Cloud”

[18] Jiewen Huang, Daniel J. Abadi, Kun Ren, ”Scalable SPARQL Quering of Large RDF Graphs”, 2011-VLDB Endowment

[19] Hyunsik Choi, Jihoon Son, YongHyun Cho Min Kyoung Sung, Yon Dohn Chung, ”SPIDER : A System for Scalable, Parallel / Distributed Evaluation of large scale RDF Data” ,VLDB 09, August 24 28, 2009, Lyon, France Copyright 2009 VLDB Endowment, ACM 0000000000000/00/00.

[20] Mohammad Farhan Husain, Pankil Doshi, Latifur Khan, and Bhavani Thuraising- ham, ”Storage and Retrieval of Large RDF Graph Using Hadoop and MapReduce”, Springer-Verlag Berlin Heidelberg 2009

[21] Cathrin Weiss, Panagiotis Karras, Abraham Bernstein, ”Hexastore: Sextuple Indexing for Semantic Web Data Management ”, PVLDB ’08, August 23-28, 2008, Auckland,

New Zealand Copyright 2008 VLDB Endowment, ACM 978-1-60558-305-1/08/08

[22] ”https://onlinecourses.nptel.ac.in/ ”

[23] ”https://slideplayer.com/slide/7237541/ ”

[24] David Dietrich, Barry Heller, Beibei Yang ”Data Science Big Data Analytics”, 2015

[25] David C. Faye, Olivier Cur, Guillaume Blin ”A survey of RDF storage approaches”, ARIMA Journal, vol. 15 (2012)

[26] Cathrin Weiss, Panagiotis Karras, Abraham Bernstein ” Hexastore: Sextuple Indexing for Semantic Web Data Management”, PVLDB '08, August 23-28, 2008, Auckland, New Zealand Copyright 2008 VLDB Endowment, ACM 978-1-60558-305-1/08/08