

EFFICIENT MANAGEMENT OF RDF DATA

V Dhivya¹, S. Sarah Suba Rani¹, D. Deepa² and R. K. Kapilavani³.

¹ Student, Department of CSE, Prince Sri Venkateshwara
Padmavathy Engineering College, Tamilnadu, India.

² Assistant Professor, Department of CSE, Prince Sri Venkateshwara
Padmavathy Engineering College, Tamilnadu, India.

³ Assistant Professor, Department of CSE, Prince Dr.K. Vasudevan College of
Engineering and Technology, Tamilnadu, India.

ABSTRACT

RDF is the Resource Description Framework which acts as a meta content for the schematic & information level data by encoding them into sophisticated graphs. Following older methods such as graph partitioning, sharding or Min-Cut algorithms makes the process complex due to presence of high number of joins. Here, we propose RDF Management with increased scalability and efficiency for the cloud. Before partitioning the data, we analyze the data at instance-level and schema-level. We propose new algorithm for partitioning the data's in such a way that systems performance is made twice as faster than the existing systems performance.etc.

KEY WORDS -RDF, Triples, SPARQL, BGP

1. INTRODUCTION

Processing huge amount of data in parallel in cloud is a very easy task .But some process requires sequential processing as they are difficult to parallelize due to the generation of inter-process traffic. Parallelism of process for distributed data's involves large number of joins. This may lead to a heavy weighted process.RDF data management is the best way to reduce the number of joins. GridVine is told to be the one which introduces large scale dissolution RDF management technique. Several RDF systems follow hash partitioning(i.e., triples)as it can manage workloads efficiently but often results in complexity in processing

Here we propose efficient RDF management for distributed environment. This concept relies on mining semantically data's at schema and instance level

This article concentrates on :

- a composite model which links up data's semantically by partitioning them into graphs
- a well built architecture to handle the partitioned data's
- A well derived algorithm to handle queries based on the partitioned data's

This approach supports query processing in every aspect to keep track of each and every step of the process.

There are many approaches which have been proposed to enhance RDF storing strategies and SPARQL query evaluation. The most common concept used is triples which operates as a set i.e., subject-predicate-object . RDF-3X also follows a similar technique by implementing SPARQL queries using RISC architecture most of the classical approaches make use of clustering of molecules concept where similar data's are structured like a B+ tree. Each molecule represents each instance and are attached to its common subject . In Owens, gibbons approach the query from user is stored in basic graph patterns and are allowed to find a match in their RDF.

The Hadoop Distributed RDF Store(HDRS) process distributed RDF data using MapReduce by dividing independent data's into chunks as in RAPID+ which uses(object, key)pair to store the integer ID and the value allocated to the independent data. Jena HBase is application program interface(API) which is used to implement triples by query processing.

Older approaches such as Virtuoso uses Quad elements such as graphs, subject, predicate, object. These quads occupy the same table and are divided on the basis of the subject. Algorithm such as Standard Graph Partitioning algorithm partition data's in

such a way that the number of inter-node joins are reduced. WARP(Workload Aware partial Replication and Partitioning for RDF) is such an approach that partitions data based on workloads and execute them in parallel

2. RELATED WORKS

There has been many works proposed to enhance the RDF in its query exhaustion process. Triples is used to be one of the best method which operates on subject, predicate and object. It stores not only the data's but figures out the relationship between the data's. Data's are partitioned in a parameterized format using PIG(parametric index graph){18}.PIG creates interdependency between data's for dynamically accessible systems. It intersects all sub graphs which has similar subject produce a single metadata for all the sub graphs which are joined together as a single graph[45].To exchange RDF in a distributed environment we use 3 logical components which are the header ,Dictionary and triples header contains the physical structure and size information of the meta content and is the entry point for the information. Dictionary us integer ID to map the contents defect using locate and extract functions it fetches the data. Triples acts as a object here to map data's in a hierarchical fashion. For the query processing[9],the user input data's are split into sub queries and each data provided with a integer for uniquely identification. [59]LUBM is the common benchmarking approach used it benchmark more than one million benchmark triples. It takes very less time to deliver the results.[40] Bourglana benchmark differs with LUBM in term of the time efficiency and query processing .[26] Virtuoso is an open source once which supports ODBC connections to evaluate queries.[41] DBpedia can extract one billion queries and is more loud than other approaches. OLTP Benchmarking [42] is the best suited approach for the RDF as it can operate on all DMS's platform. It makes the process Light weighted and supports many to many relationship to balance the workload for faster accessing we use WRAP[Work Load Aware Replication and Partitioning of API [30] Approach to support parallel processing, we replicate necessary data's for sub queries. There are two types, first one passes the query and evaluate parallel .The second approach is Multiple Queries Processing which Subdivides the queries an process them in parallel and finally joins them to get the desired result. Gridvine[2] is the interface which builds an bridge between the information's those are extracted and the meta content. We use RISC supported Engine RDF 3x to build compressed indexes to save CPU consumption and time we use RDF 3x 0.35 here to keep track of the quantify of data and to update dynamically growing data's.

3. STORAGE MODEL

We follow composite structure which clusters the relational data's on the basis of the subject common to them. Each node .i.e., molecule is provided with an integer ID which is used to uniquely identify the particular data. A triple table containing the (key, value) pair is maintained where object represents the literal and the value is the integer ID allocated for it.

Each and every distinct data holds a place in the memory and data's under common subjects holds similar value. Horizontally and vertically semantically information is stored through molecules and templates. For example, college ID may b the template for the course cluster and molecules for it are the name of the different courses

All the queries distributed over processors are grouped together and the result is provided as URIs.

3.1 KEY VALUE

Each and every data is operated using the key value. For Example,(course,0011) will automatically point to the course with ID as 0011.It is majorly used to encode all URIs with unique ID. The relational Graph Partitioning Algorithm uses structuring and groups all the datasets.

With each leaf node having its unique ID it follows 2 paths one leading to extract the literal corresponding to the key value and the other one provides the output.

3.2 MOLECULES NAD TEMPLATES

Templates have unique ID through which its whole branch is operated, A template contains its clusters and molecules grouped together. Triples store separate table for this template list which is accessed by the RDF which in turn points to the clusters and molecules. Here is where the process becomes light weighted. The triples for this template is stored in the main memory and in the disk so that any loss of this table can be backed up and it is arranged in column oriented fashion.

Molecules represent each and every instance. It is grouped based on the cluster and the templates. It is arranged in a fashion where it can easily be fetched and join to obtain the expected result. It is retrieved based on the (key, value) pair indicating the literal and its integer ID representing the molecule

4. SYSTEM ARCHITECTURE

4.1. Semantic db RDF generation

The Resource Description Framework (RDF) is constructed for semantic data on a Relational Database containing Structured as well as Unstructured data. A Schema is identified for the relational database and a RDF representing the schema of the database is constructed through model provided by the jena api. The Model contains all the information about the data linkages in the schema. In this process the schema can also be altered based on admin requirement so that the search process can be effective.

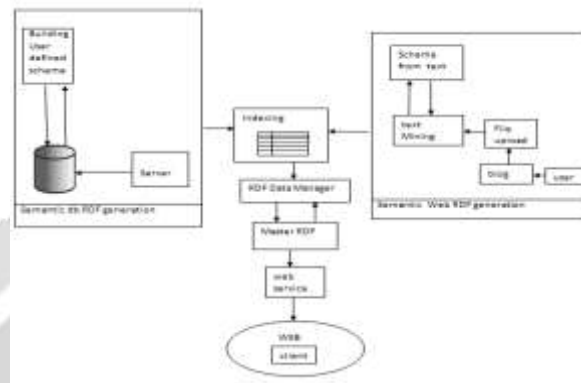


Fig.1: Master RDF management architecture

4.2. SEMANTIC WEB RDF GENERATION

The semantic Web RDF generation is an generate and RDF data for use entered data. The user Entered data is converted to RDF file is constructed for the jena api. The converted RDF file is created an web service and when that the RDF file is required to send an web service response.

4.3. DATA PARTITIONING AND INDEXING

The RDF is also generated by mining the text contents uploaded by users in blogs and contents of the file are analyzed and the meta contents are manipulated. The meta contents are the key for search process so that the file can be rendered on demand. The ext mining process analyses the text word by word and also picks up the literal meaning behind the group of words that constitute the sentence. The words are analyzed in WordNet api so that the related terms can be found for use in meta content in generation of RDF. Generally RDF runs in the Web services of servers all over the world to provide the schematic data that the server holds in Data_Base in distribution in the web to access it. Hence this process is shown in real time and the text also analyzed in web service provided by a different environment so the user uploaded content will also be analyzed in real time servers in their own natural language processing strategies and the results are obtained in a RDF format so that it can be understood by other servers .

4.4. QUERY PROCESSING OVER INDEXED DATA

Similar data are grouped together that are relate to the same resource .The data level process are subjected to structure level processing by indexing the semantic data elements. Multiple RDFs are grouped and structured together to form a master RDF data that holds all the semantic information of a server that support reasoning in any format in query processing the different resources are interlinked with high degree of relational factors by the predicates in the triples. The query processing is handled directly in the RDF file by the iterating in the triples forming a discreet relation with the service query and the URI representing the location of the resource is returned and this process is handled in a web service in real time servers. Hence the structure oriented approach to RDF data management where the data partitioning and query processing make use of the structure patterns generated by the RDF .

5.STRUCTURING DATA

Traditional data partitioning approaches such as property table or hash partitioning leads to height network latencies. It has faced difficulties in partitioning and scaling the resources.

In our composite approach we analyse data from instance and logical level by iterating the process until proper schema is obtained.

5.1 GRAPH PARTITIONING AND INDEXING

We use adaptive partitioning method to handle queries and workload it uses sliding window to track the amount of workload assigned to the processors. After mining the contents, we build schema by analysing the datasets word by word wordnet API. So that meta content would become easier

After grouping the data's on the basis of their subjects we provide a unique ID to each and every template which acts as a primary key to manipulate the data and index them in certain order.

6. OPERATIONS

1. LOAD BALANCING

The process is split into multi sub-tasks and is executed in parallel for faster processing. Semantic web contents are dynamically uploaded. Hence, may have more number of workloads. Therefore, we go for parallel processing to handle bulk amount of data's,

2. UPDATES

The growth of data's and information get increased day by day. So to keep track of them, we use complex update technique which modifies molecule properties whenever necessary. It uses the index value as reference and rewrites the molecule and saves them column-wise.

7. QUERY EVALUATION

Evaluating the query is the most important part of query and is divided into sub-queries by extracting only the semantically available information from them and is set for comparison with the RDF and then from the schema, the data is retrieved and URI is generated as the output.

ALGORITHM 1-

QUERY EVALUATION ALGORITHM

- Semantic information retrieval
- Dividing into sub queries
- Processing sub-queries in parallel.
- Joining process to obtain an output.

8. HANDLING QUERIES

- **BGP**- It is used to retrieve triples grouped same subject. Since, ID is given to each molecule retrieval becomes easier and no any joint of results are required as these acts implicitly.

DISTRIBUTED JOINS- Joining results of sub-queries is done with the help of triples. The triple table gathers 3 elements subject-predicate-object where each of them are relationally linked to one another using template ID. Therefore, joining of these molecules becomes easier

QUERY EVALUATION ALGORITHM FOR DISTRIBUTED JOINS

```

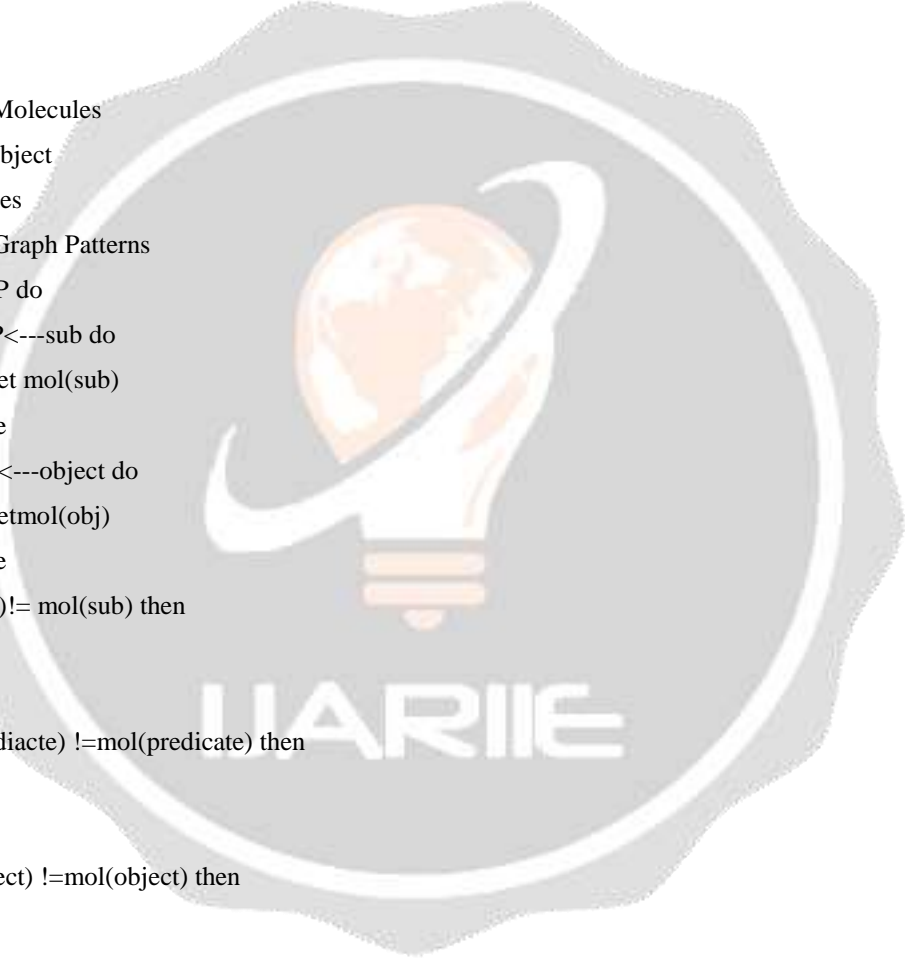
Procedure QueryHandler(P,Q)
  If the query has BGP then
    BGP ← Query
  If the BGP has subject && If the BGP has object
    molecules ← Get(Subject, object)
  end if
  If all the molecules has Triples then
  If all BGP has triples then
    Check  if( Triples.Subject == molecule.Subject)&&
    (Triples.Predicate == molecule.Predicate)&&
    (Triples.Object == molecule.Object)
  If yes
    Stop
  else

```

```

Get NextMolecule
end if
End if
End if
Result ← Get(Molecule,BGP)
End if
result ← Result
End if
Senduser(results)
End Procedure

```



```

Mol< ---Molecules
sub<---subject
T<---triples
GP<----Graph Patterns
For all GP do
While GP<---sub do
Mol<---get mol(sub)
End while
while GP<---object do
Mol<---getmol(obj)
End while
if TP(sub)!= mol(sub) then
Nextmol
end if
if Tp(prediacte) !=mol(predicate) then
Nextmol
end if
if Tp(object) !=mol(object) then
Nextmol
end if
results<--- retrieve (mol,GP)
end if
result<---results
end for
end procedure

```

9. BENCHMARKING

We use DBpedia and LUBM benchmarking approach which allows RDF to be compared with 4 synthetic workloads which is ranked to be the top 4 RDFs. It also allows 1.2 million triples to be compared and displays the ranking of those in sequential order .

10. SYSTEMS USED

Jena- we use jena 2.6.4 to measure the performance and time taken execute every query

RDF-3X:RDF 3X 0.3.5 is used to provide generic solution after query processing

virtuoso-open source virtuoso 6.1.3 is used to access database through ODBC connection

11.PERFORMANCE EVALUATION

To evaluate performance we compare the existing RDF with and the proposed RDF master RDF taking various criteria's.The main advantage of the proposed system is its time saving capabilities and reduced number of joins. It provides faster and efficient accessibilities which is faster than classical approach. Master RDF retrieval is very fast when compared to the sub RDF retrieval. These are the important aspects which decides the efficiency of the proposed system.

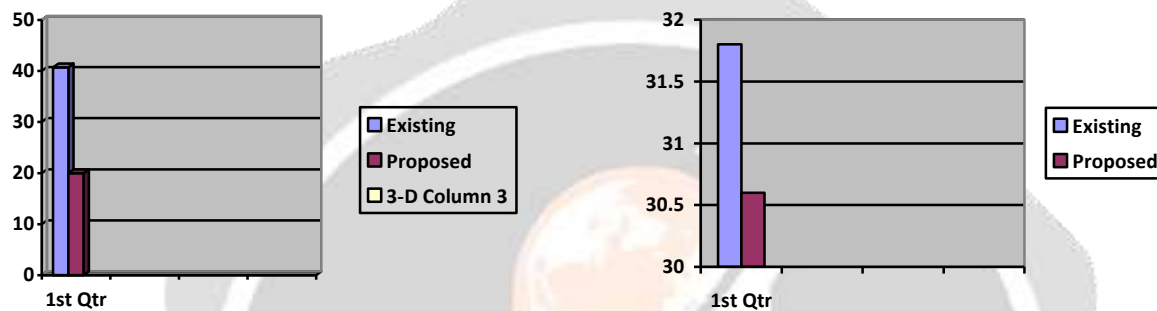


Chart1: COST EFFICIENCY

11. CONCLUSIONS

Master RDF generation is efficient and scalable because of its light weighted processing property as analysis of data is done from instance and schema level. Therefore generation RDF becomes easy and simple

12.REFERENCES

- [1] P. Cudre-Mauroux, S. Agarwal, and K. Aberer, "GridVine: An infrastructure for peer information management," *IEEE Internet Comput.*, vol. 11, no. 5, pp. 36–44, Sep./Oct. 2007.
- [2] B. Liu and B. Hu, "An Evaluation of RDF Storage Systems for Large Data Applications," in *Proc. 1st Int. Conf. Semantics, Knowl. Grid*, Nov. 2005, p. 59.
- [3]- Query execution time on Amazon EC2 for 1,600 Universities from LUBM dataset. Fig.
- [4] O. Erling and I. Mikhailov, "RDF Support in the virtuoso DBMS," in *Networked Knowledge-Networked Media*. New York, NY, USA: Springer, 2009, pp. 7–24.
- [5] K. Hose and R. Schenkel, "WARP: Workload-aware replication and partitioning for RDF," in *Proc. 29th Int. Conf. IEEE Data Eng. Workshops (ICDEW)*, 2013, pp. 1–6. .
- [6] G. Demartini, I. Enchev, M. Wylot, J. Gapany, and P. Cudre-Mauroux, "BowlognaBenchBenchmarking RDF Analytics," in *Data-Driven Process Discovery and Analysis*. New York, NY, USA: Springer, 2012, pp. 82–102.
- [7] C. Becker. (2008). RDF store benchmarks with DBpedia [Online]. Available: <http://wifo5-03.informatik.uni-mannheim.de/benchmarks-200801/>
- [8] D. E. Difallah, A. Pavlo, C. Curino, and P. Cudre-Mauroux, "OLTP-Bench: An extensible testbed for benchmarking relational databases," *Proc. VLDB Endowment*, vol. 7, no. 4, pp. 277–288, 2013.