A RDF data compress model based on octree structure

Kaidong Wang[1,2],Haidong Fu[1,2],Shen Peng[1,2],Yu Gong[1,2],Jinguang Gu[1,2*]
[1]College of Computer Science and Technology, Wuhan University of Science and Technology, China
[2]Hubei Province Key Laboratory of Intelligent Information Processing and Real-time Industrial System, Wuhan, Hubei, 430065, China
shzwkd@foxmail.com, drfu@163.com, sean_ps@163.com, wustgy@126.com, Simon@wust.edu.cn

***Abstract*—With the advent of the big data, large size of RDF datasets was flooded the Web. These data supported and managed through SPARQL endpoints, and it plays a role as a semantic net nodes based on the Linked Data Projects methods. The size of these RDF datasets was so huge that when systems need to process these datasets, their indexes cannot be fully loaded in main memory. For the reason given above, when system manages these huge datasets, the SPARQL queries can only perform through the slow disk accesses, which causes serious query performance problems.**

This paper proposed a method to solve this problem, offered a compact indexed RDF structure called oct-triples, which based on the octree structure, to store the indexes saved as three-dimensional matrix, which mapped by the dictionary and RDF triples. And this method has clearly an advantage on the compressibility compared with traditional compression solutions.

Keywords*—*RDF dataset compression; octree; three-dimensional matrix;

I. INTRODUCTION

RDF (Resource Description Framework) [1] is a technical standard proposed by the W3C which used for processing metadata, its purpose is to define a method for describing resources and relations. Obviously , this method is the extend from the document-centric perspective of the Web, rather than using to serve web pages for human readers, people tried to make it possible to share information in a way that can be processed automatically by computers, which allowed data form different sources to be connected and queried, while RDF plays an important role in the semantic queries and linked data publishing.

Linked Data [2] publishing structured and machine-readable data on the Web, and linked these data from different datasets, it built on the resources and relations between them. The basic idea of Linked Data is using RDF to publish structured data on Web and use links between RDF to interlink data from different datasets. RDF is based on the idea of making statements of resources expressions, it offered a triple structure data model to describe the structured linked data in the real world, which follows a subject-predicate-object structure. The subject is the description of the resource, the predicate is the traits of the resource, and expresses the relationship between the subject and object, the object denotes the value of the resource's property. To query the RDF data, W3C had standardized a query language called SPARQL [3], it is based on triple patterns, and treat RDF triples that each subject, predicate and object as a variable, while its queries solve the conjunctions of triple patterns by graph matching.

The semantic web now is composed of very large RDF datasets from different fields, so when we manage and query the data, the performance of the query and scalability of data may be the core problems. Although we can store the data of huge size by using enough numbers of disks, however the large size not only cause low efficiency issue of the query but also makes the performance of the RDF publication and exchange more worse. With the widely use of the SPARQL query, publishing and exchange of RDF in the query also become more and more popular. So it is significantly to reduce the size of the dataset, this need to establish an advisable logical structure to store the linked data.

In this paper, we would introduce the current compress solutions of linked data; then we would describe how to map the logical model from the RDF triples and describe the octree structure as the basis to store the model; we will evaluate the advantages of this model against the current compress methods and pay attention on the compression ratios for real-world datasets. Finally, we give a conclusion and devise future work on current achievements.

II. CURRENT SOLUTIONS OF RDF DATA COMPRESS

Based on theoretical analysis on the redundant type of the linked data, [4] divided the redundancy types of the linked data to two types: semantic redundancy and inter-structure syntactic redundancy. Semantic redundancy means that a RDF dataset exists some triples that can be deleted without leading to any changes in its meaning, and we can reduce these triples by rule reasoning to realise data compress storage. Inner-structure syntactic redundancy denotes that the redundancy problem caused by the storage structure of the triples in the RDF datasets, and it can be serialized by change the structure that the data stored in the linked data, in this way, we can express the information of the linked data by more compact data structure to compress the dataset. In actual compression results, iner-structure syntactic compression method worked better efficiency than semantic method, so, most datasets compress solutions worked from the inter-structure syntactic aspect.

Paper[5] was the first to propose several base cases to compress the RDF datasets, utilize some normal compress solutions for file, for example, [6] and [7] had changed the file structure of the linked data, which could clearly reduce the size of the dataset. And [8] proposed the method called BitMat, which used a compact bit matrix data structure to store the RDF datasets, while it also supports the directly query operation upon the compressed datasets, but this method has low expansibility and the compress method it used was simple. [9-10] introduced a method for linked data compression, which called HDT(Header-Dictionary-Triples), and it divide the information of the linked data into three parts to describe the the RDF dataset: the first part called Header, and it stores the logical and physical metadata which contains the information that describes the dataset, such as the provenance of the dataset and some statistical information of the dataset; The second part called Dictionary, the major function of this part is to map the data concentration resources into a unique mark; The third part called Triples, this part first used the Dictionary part to deal the data concentration triples, then divides this triples into groups on the basis of the subject information, at last it will compress these triples. HDT method has saved the underlying structure of the RDF datasets, at the same time it also avoids the problem that data concentration is too long and the repeated resources description.

In the rest of this paper, we will propose a RDF dataset compress method called Oct-Triple which based on the idea of HDT that use the dictionary to divide and map the huge RDF datasets to some ID triples, and by treat the ID triples as points of the three-dimensional matrix, then store and compress these points by octree data structure and serialization, this method enhances the efficiency of the data compress and also made the logical structure of these triples stores and operate in an intuitively way.

III. Oct-triple compress model

A. Analysis of RDF data redundancy

In inner-structure syntactic compression methods, methods that based on relationship three-dimensional matrix clearly describe the exist relationship of subjects, predicates and objects in data structure. But for to build the data structure of matrix, there also need to store some points that did not denote any triples, especially when the linked data increased to a huge size, the three-dimensional matrix of the linked data will become a super sparse matrix data structure to store it, and will generate many redundant data. Obviously, SPARQL query can support this data structure well, but the compression rate of it did not work enough well.

According to the principle of the three-dimensional matrix data storage, it can be seen that the redundant information stored in the matrix is mainly for the construction of the full three-dimensional matrix, but it also distributes the storage space for the relationship that did not exist. If the redundant data that did not have a relationship can be removed from the matrix, the compression rate. So the question turned to how to store and compress a three-dimensional matrix.

The octree [11] is a tree data structure which used to describe the information in the three-dimensional matrix. The octree data structure is compressed from three-dimensional grid data matrix. It is a generalization of a 2D grid matrix of quad-tree in 3D space. The data structure divides the space it represented by X, Y, Z three directions from the middle segment, and divides the three-dimensional space into eight cube sizes. Each node corresponds to a sub cube octree and the cube, and the root node correspond to the total cube, as long as a child node is not completely blank or completely occupied by only one element, it will continue to eight equal to the corresponding node has eight child nodes. By defining the flag and assigning the node to define whether there is a point in the subcube, when the identity of 1 is a child node, when the flag is 0, means no child nodes. In the end, we can get an unbalanced tree, and then we can exclude the existence of some large redundant space.

Base on the above ideas, this paper proposes a compress model based on the ID triples achieved from map process to the RDF triples, by using the octree structure to store three-dimensional matrix, then we will traverse the whole tree to obtain a linear sequence value, finally we can remove large amount of redundant information in the matrix, hereinafter referred to as the compress model for Oct-Triple.

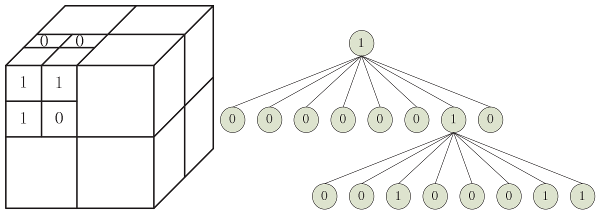
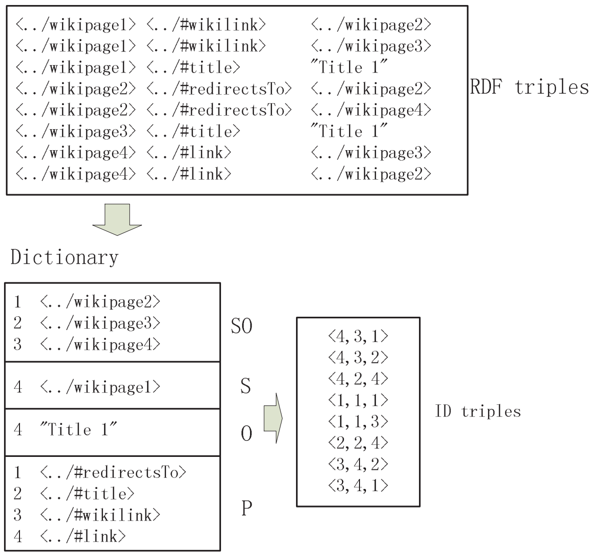


Fig. 1. Example of a octree data structure

B. Oct-Triple Model

Based on the idea of Header-Dictionary-Triples that extract information, build the dictionary and mapped the triples to ID triples, we can get some ID triples that consist of three integers, which can be treated as the points of three-dimensional matrix, if directly stored these points as three-dimensional matrix, there will be a lot of redundant storage spaces because of the exist of the blank cubes of the matrix, from the previous analysis for sparse three-dimensional matrix we can know that if we can remove the child matrix which lead redundant storage space, the storage space can be reduced greatly. In the previous section, we know that we can achieve this goal by using the octree data structure, while considered that the nodes we set before have used the mark 1 and 0 to identify the existence of the nodes, which denotes the status of the nodes just use two values, so we can store these values of nodes by bits. We can traverse the nodes of the octree to obtain linear sequence value that composed by marks 1 and 0, and this value can use to denote the elements of dataset, which stored as points in the matrix, this way convert the octree data structure into a linear list, and each element correspond to a node of the octree. By this way, the octree data structure can be stored in the main memory by an ultra compact way, which can use no pointer or just one pointer for the whole tree. So this paper focus on the idea that first extract the ID triples which mapped from the RDF triples, then treated them as points of a three-dimensional matrix and traverse each node to read its value so that we can get a linear sequence for storage, in this way, the triples that stored in the main memory can change to value 0 or 1as a bit in the linear sequence, which made the size of the ID triples reduced. According to the above process, we can construct the Oct-Triple model which describes the nodes with linear sequence values used 1 or 0, compared with the original method that Header-Dictionary-Triples used to store and compress the ID triples, Oct-Triples store these triples by matrix and mark value 0 or 1 in bit level, a leaf node in the main memory just occupy only 1 bit of space, reduce the size of storage space caused by the large number of ID triples. In the rest we will introduce how to get a linear sequence value, by the process that extract the header information, construct the dictionary and map the triples into ID triples, finally store them into the octree data structure, then serialize it.

Oct-Triple reference some steps and idea of Header-Dictionary-Triples, utilize the structure that composed the triples, to divide them into subject parts, predicate parts and object parts, then combine them to compare the value of them and mark different marks to construct dictionary, by use the dictionary construct before, we can map these RDF triples into corresponding ID triples composed by three parts of integers. The process is as follows fig.2.



```
<../wikipage1> <../#wikilink>      <../wikipage2>
<../wikipage1> <../#wikilink>      <../wikipage3>
<../wikipage1> <../#title>         "Title 1"
<../wikipage2> <../#redirectsTo>   <../wikipage2>
<../wikipage2> <../#redirectsTo>   <../wikipage4>
<../wikipage3> <../#title>         "Title 1"
<../wikipage4> <../#link>          <../wikipage3>
<../wikipage4> <../#link>          <../wikipage2>
```

RDF triples

Dictionary

```
1  <../wikipage2>              SO
2  <../wikipage3>
3  <../wikipage4>
4  <../wikipage1>              S
4  "Title 1"                   O
1  <../#redirectsTo>           P
2  <../#title>
3  <../#wikilink>
4  <../#link>
```

```
<4,3,1>
<4,3,2>
<4,2,4>
<1,1,1>
<1,1,3>
<2,2,4>
<3,4,2>
<3,4,1>
```

ID triples

Fig. 2. Construct ID triples

The idea of storing ID triples in the Oct-Triple model is treated them as a set of points stored in a three-dimensional matrix, processes are as follows:

- First compared all these ID values of the sets to get the max one which denoted by symbol N_max, based on the value of N_max, count the value denoted by symbol L_max to define the range of the three-dimensional matrix that correspond to the octree data structure, the formula is as follow:

$$L\_max = \lceil \log_2(N\_max) \rceil \tag{1}$$

- While stored a ID triples correspond to the point (x,y,z) in the 3D coordinate axis, first compared the value of x, y and z with each value of middle points of the coordinate axis, to count which is its place in the eight child matrices, then marked the value of this node with integer 1, represents the presence of one or more nodes in the three-dimensional matrix range of this child node, and marked other child node without triples stored in with integer 0, that denotes that there exists no points in the scope of the child three-dimensional matrix.

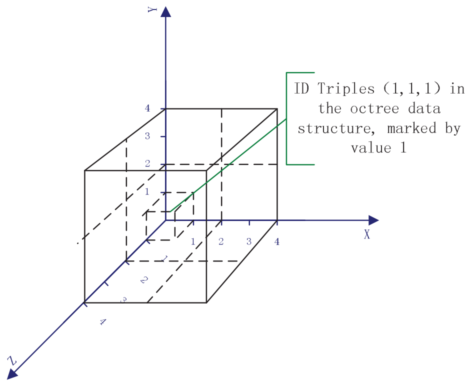
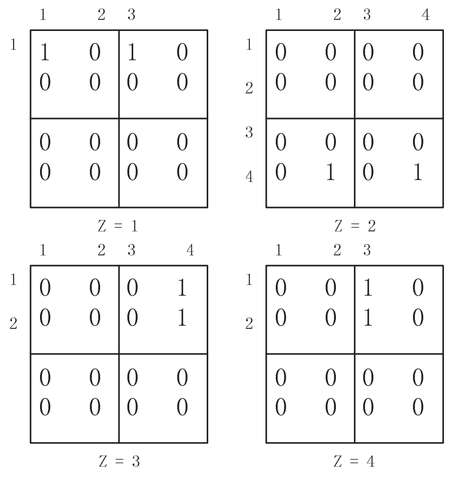

Y
X
Z
1
2
3
4
1
2
3
4
1
2
3
4
ID Triples (1,1,1) in the octree data structure, marked by value 1

Fig. 3. ID triple in the range of three-dimensional matrix

- Then continued to count the location of the point in the eight child three-dimensional matrices of the corresponding node, until the range of the node can just store only one value. After these processes, the whole points of the three-dimensional matrix can be described by each node of the octree data structure, the coordinates of the nodes is the content of the ID triples.



Z = 1

	1	2	3	
1	1	0	1	0
	0	0	0	0
	0	0	0	0
	0	0	0	0

Z = 2

	1	2	3	4
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	1	0	1

Z = 3

	1	2	3	4
1	0	0	0	1
2	0	0	0	1
	0	0	0	0
	0	0	0	0

Z = 4

	1	2	3	
1	0	0	1	0
2	0	0	1	0
	0	0	0	0
	0	0	0	0

Fig. 4. ID triples stored in the octree structure

- Based on the predefined order to traverse all nodes of the tree structure, and collect these mark values, finally we can get a linear sequence list composed by the mark value of the leaf nodes and there parent nodes, the list composed by number 1 and number 0, by stored the sequence data with bit by bit, the data set can be compressed.

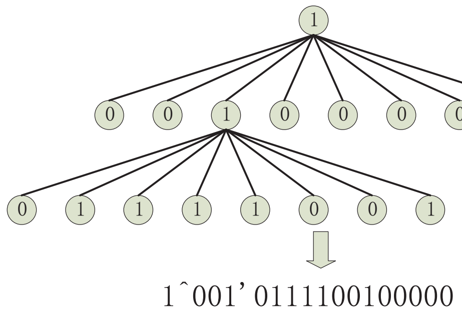


Fig. 5. Linear list of the ID triples

C. *Algorithm of construct Oct-Triple*

The algorithm of construct Oct-Triple is as follows, this algorithm programs by tha scala language, and run the program in spark:

Import: RDF dataset DATA with format N-Triple

Export: linear sequence list of ID triples

Initial preparation: construct the class data structure Triple for RDF triples, which composed by subject(s), predicate(p) and object(o) three parts; construct the class data structure TreeNode for the nodes of octree, which included the properties of range values(x_min:Double,x_max:Double, y_min:Double,y_max:Double,z_min:Double,z_max:Double), marked value(data:Integer) and eight child nodes(childnodes:Array[TreeNode]=new Array[TreeNode](8))

- Load the file and deal with the triples of DATA, filter the lines which had been annotated or the blank lines, then read each line of data and divide to save it as objects of Triple. Finally can get a RDD dataset composed by objects of class Triple. Then flattening the data in the dataset and remove the repeated data, construct a dictionary dic with the map relationship with ID and the metadata.
- Combine the Triple dataset and the dictionary dic, and mapped the real RDF triples into ID triples based on the rule of dic.
- Combine the ID triples to calculate the maximum ID value N_max, utilize the rule of three-dimensional matrix and octree data structure can count the maximum level n of the octree and the maximum range L_max of the matrix.
- Traverse the ID triples, extract three ID values from each triples as its value in the coordinate axis of the three-dimensional matrix, calculate the range it located in the child nodes recursively, marked the leaf node and its parent nodes with value 1 which the ID triple located in, finally can get an unbalanced octree data structure.

```
While (n != 0)
    IF （x,y,z） ∈ TreeNode.childNode(i)
      Create TreeNode.childNode(i)
```

- Traverse all the nodes of octree constructed before, obtain a linear sequence list, and converse the list by 8 bits to a Byte to compress and store it.

IV. Analysis of experiment

A. *Datasets for experiment*

To identify the availability of the Oct-Triple model, we did the compression experiment of the datasets in table 1, all datasets in this table ordered by its size.

TABLE I. DATASETS

dataset	*Size of the file*	*Triples*	*predicates*
CN2012	18.2M	137441	11
Archive Hub	73.5M	431088	141
World Net	98.9M	515940	515934
geospecies	182.8M	689750	376715

B. *Compressed result*

When store the triples as points in three-dimensional matrix with Oct-Triple, it also stored the nodes that contained the leaf nodes which correspond to these points, these will lead to some extra spend for storage of dataset, but with the recursively divide structure that octree data structure have, its cost of parent nodes is diminishing by a $\log_2(n)$ level by each level of the tree, so compared with the convenience the structure tooked and the compress ratio it can get, these cost on storage can be ignored.

The table below is compared with HDT triples on the size of file and compression ratio (in this table we just compared the situation of triples’ storage, will not calculate with the size of Header and Dictionary file), table is as follow:

TABLE II. SIZE OF TRIPLES AFTER COMPRESSED

Dataset	*Triples*	*HDT*	*Oct-Triple*	
		Triple set Size(MB)	*Triple set Size(MB)*	*Ratio*
CN2012	137441	0.70	0.25	35.7%
Archive Hub	431088	2.50	1.62	64.8%
World Net	515940	3.80	0.50	13.2%
geospecies	689750	5.20	1.51	29.0%

C. *Analysis to the result*

The analysis focuses on the space that the file occupies to represent the triples of the RDF dataset.

To analysis the results of different we can find that in some datasets, the compress ratio of the Oct-Triple model clearly shows the better result than other datasets, while we analysis the structure of the datasets, we can find that the ratio of dataset depends on the number of different subjects, different predicates and different objects, the dataset World Net has the best compress ratio 13.2% on these results, and its numbers between different subjects, different predicates and different objects has the minimum minuses, and for the dataset Archive Hub which has the worst compress ratio, its predicates and subjects have huge minus result. That is because of the store rule of the octree data structure. It will hold less number of redundant tree nodes that have no triples contain in it when the points in it have more uniform distribution.

To compared with the method k^2-triple [12-13] that divides the three-dimensional matrix into several two-dimensional matrices ordered by the different predicates, in the compression of dataset which its ID triples have a huge number of different predicates, the Oct-Triple have advantage on it. Because of the k^2-tree data structure, each different predicate ID means that a new k^2-tree and a corresponding mark of predicate ID will be constructed. And at the same time, decompression and query efficiency will be depressed when the predicates' number increased, while the Oct-Triple stored in just one linear sequence list, on this aspect it will not be affected as much as the k^2-triple.

V. CONCLUTION AND FUTURE WORK

This paper from the aspect of the inner-structure syntactic data compress method, concluded the advantages and the disadvantages of it, then proposed the Oct-Triple model: which based on the three-dimensional matrix constructed by the map result of subjects, predicates and objects of RDF triples, and store it with linear sequence to compress it, this model has a greater advantage in compress ratio compared with the ID triples, utilize the linear sequence to store the matrix and decreased the size of the ID triples generated from the RDF dataset.

The research shows that the compress method we proposed that have a good compress ratio, and the advantage of octree data structure will make the decompress and query operations more easier, but it will store some redundant information in it when the data structure has too less predicates. So the future work is updated the structure of Oct-Triple model by divides it into more compact type of child matrix model, and this divide of data will make it possible to construct a distributed query on the base of Oct-Triple, that is one of the future works of this paper.

REFERENCES

[1] Manola, Frank, Eric Miller, and Brian McBride, "RDF 1.1 Primer," W3C Working Group Note, 2014.

[2] Christian B et al., "Linked Data ? The Emerging Cloud of Data ? :1. Linked Data -The Story So Far," Ipsj Magazine, vol. 52, pp. 284-292, 2011.

[3] Harris, Steve, Andy Seaborne, and Eric Prud'hommeaux, "SPARQL query language for RDF," The World Wide Web Consortium, 2016.

[4] Wu et al., "How redundant is it?-an empirical analysis on linked datasets," Proceedings of the 5th International Conference on Consuming Linked Data-Volume 1264. CEUR-WS. org, pp. 97-108, 2014.

[5] Navarro, Gonzalo, and Veli Mäkinen, "Compressed full-text indexes," ACM Computing Surveys (CSUR), vol. 39, no. 1, pp. 2, 2007.

[6] Fernández, Javier D., Claudio Gutierrez, and Miguel A. Martínez-Prieto, "RDF compression: basic approaches," Proceedings of the 19th international conference on World wide web. ACM, 2010.

[7] Pavlov, Igor, "Lzma sdk (software development kit)," 2015.

[8] Seward J, "bzip2 and libbzip2," 2010.

[9] Álvarez-García et al., "Compressed vertical partitioning for efficient RDF management," Knowledge and Information Systems, vol. 44, no. 2, pp. 439-474, 2015.

[10] Fernández, Javier D., Miguel A. Martínez-Prieto, and Claudio Gutierrez, "Compact representation of large RDF data sets for publishing and exchange," International Semantic Web Conference. Springer Berlin Heidelberg, pp. 193-208, 2010.

[11] Meagher, Donald JR., "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer," Electrical and Systems Engineering Department Rensseiaer Polytechnic Institute Image Processing Laboratory, 1980.

[12] Atre, Medha et al. "Matrix Bit loaded: a scalable lightweight join query processor for RDF data," Proceedings of the 19th international conference on World wide web. ACM, pp. 41-50, 2010.

[13] Álvarez-García et al., "Compressed k2-triples for full-in-memory RDF engines," arXiv preprint arXiv:1105.4004, 2011.