Send Orders for Reprints to reprints@benthamscience.ae

1472 *The Open Cybernetics & Systemics Journal,* 2015, *9,* 1472-1477

Open Access

A New Encoding Scheme of Supporting Data Update Efficiently

Houliang Xie[1,*] and Liang Lei[2]

[1]*Information Engineering Department, Zhangjiajie Institute of Aeronautical Engineering, Zhangjiajie 427000;* [2]*School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000*

Abstract: At present, more and more data are expressed in the form of XML format, how to manage these data efficiently becomes an important issue. To solve the problems of large encoded data storage space, data update and low query efficiency, we propose a new encoding scheme named SDUE (Supporting Data Update Efficiently). We tried to decompose the encoding node location information, avoiding recording redundant information, thus effectively saves storage space; moreover, SDUE also effectively support data updates. In the area of query, since record-taking point SDUE path information encoded in the path of the query, which avoided the structural join operations, query efficiency has been improved efficiently. Experimental results show that compared with other coding, SDUE coded has obvious advantages such as storage space utilization, efficiency query and nodes update speed.

Keywords: Efficiency query, nodes update speed, storage space, XML data encoding.

1. INTRODUCTION

At present, more and more data in XML format suggests that XML data is becoming a mainstream form of data and the standard exchange and processing of data on the network. The way of managing XML data more effectively is obtaining more and more attention. In recent years, for storing and querying XML data, researchers have proposed a number of new technologies that enable efficient storage and improved querying XML data. But there are some disadvantages, such as the lower utilization, dynamic data update problem, the path length affecting query time. In this regard, this article, considering from the perspective of XML data storage and query, proposes a new data coding scheme SDUE. SDUE coding can support any number of nodes update, which completely avoids re-encoding, the encoding also has a high storage efficiency and query efficiency.

2. RELATED RESEARCH

In order to improve the efficiency of storage and query XML data, XML data needs to be converted into structured data. As for the query structured data, dealing with conversion and check for data have been made by lots of current researchers coding scheme [1-6] and query technology [7-11].

In the study of the coding scheme, most of them are about the study of interval coding and prefix code. Region Coding design ideas is an XML document tree that each node is assigned a range of numbers <start, end>, through the front junction preorder and postorder traversal order determining the relationship between the nodes. The nodes of the former order and after preorder sequence has been determined at the time of the initial coding. However coding interval can not effectively support data updates. The researchers based on the interval and after the preamble sequence have proposed to improve the encoding vector-based [1] based on the dynamic range coding DLS [5]. The improved coding has smaller the range of storage space, but it can not be completely support data update. And query efficiency is not high. Prefix code [6]. The idea of the parent node coding node is about judging by the relationship between nodes. Structural relationship of prefix code is clear, which can effectively support data updates, but recorded a lot of duplication of information and low storage space utilization.

Research data queries are more concerns about the main structure, which can be summarized as optimization techniques, slicing technology and technology-based decomposition path. Structural optimization technology is mainly used to connect indexing techniques [7] and reduce the node set traversal to improve the efficiency. In slicing technique [8,9], a regional division of the method sets unnecessary nodes connected into different areas to reduce unnecessary structural join operations, but also essentially optimize operations on the structure of the connection slicing technique. Destination path decomposition technique [11] is to eliminate repetitive queries to improve query efficiency, in the path decomposition, which turns the complex query into simpler sub-queries, and intermediate results by subquery, generating the final query results.

3. SDUE CODING

3.1. SDUE Coding

Fig. (**1**) is a document tree of XML, the solid circle represents the tree of the junction nodes, the dashed rectangle boxes represents the node values, The attribute name and attribute values are represent in solid rectangular box. SDUE

1874-110X/15

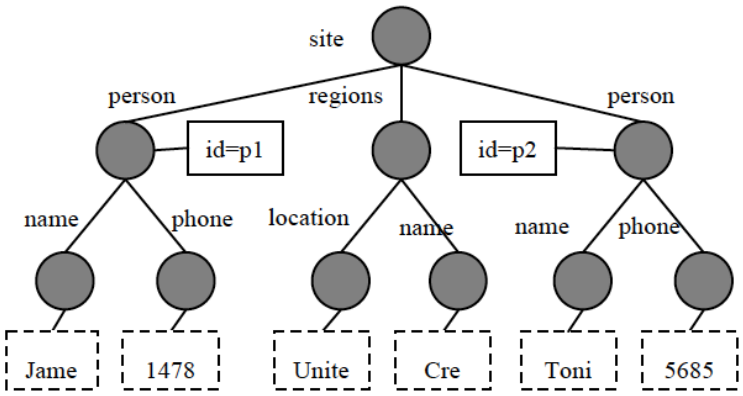


Fig. (1). The model of XML document tree.

code used four data table to record the data of XML document:

(1) The table of the node name. Node name table records node name and node name number, for different nodes, as long as the node name is same, the corresponding node number is also same, as shown in Table **1**.

Table 1. Node name table.

n_NameId	n_Name
1	site
2	person
3	regions
4	name
5	phone
6	location

Table 2. The table of node property.

n_Id	Attribute Name	Attribute Value
2	id	p1
4	id	p2

Table 3. Node value table.

n_Id	Value
5	Jame
6	1478
7	Unite
8	Cre
9	Toni
10	5685

(2) The table of node property. Nodes attribute table records the numbers, node attributes and attribute values, nodes number is different from the name of the Numbers, as shown in Table **2**.

(3) Node value table. The table records nodes number and value, as shown in Table **3**.

(4) The table of Node location. It records the nodes position and their relationships. The table can be shown as five elements group <n_Id , n_NameId , n_Postion , pathNameId , pathFreId>, as shown in Table **4**.

The n_Id represents the junction code number, n_NameId represents the number of node, N_Postion values of nodes is the parent record which a child node, the root node of the n_Postion value is set null. Node n_Postion value is represent with Numbers and letters, coding for the first time is only digital representation, nodes of the first child (left 1) n_Postion value is 1, the second child node n_Postion value is 2, and so on.

The path name pathNameId represents the number of nodes. PathNameId records the path information of itself to the root, such as Table **4**, the n_Id for 10 nodes pathNameId value for 1/2 "/", through the node name list can be found, "site/person/" is the name of 1/2 "/" path.

The pathFreId represents the frequency of node path. There are many different paths with the same name in the XML document path, so we need to distinguish with pathFreId values. Such as n_Id in Table **4** to 5 and 9 nodes corresponding path name number are "1/2 /", but they are a different path, and it need to be distinguish by pathFreId value.

Definition 1 If the u node are the v's ancestor, node v is the ancestor of n node, and it exists a path L from u to v and it a path S from u to n, the L is the parent paths of S, S is the sub path of S.

Theorem 1 If the occurrence frequency of the path named L is n, then W is any sub path of L, the number of occurrences: W < = n.

Proof: the XML document tree traversal is top-down access order, then it must visit the parent node first and then access the child node, so the parent node is accessed is always more than or equal to the number of child nodes access number, the number of occurrences of the parent path is al-

Table 4. The table of node location.

n_Id	n_NameId	n_Postion	Path NameId	Path FreId
1	1	null	null	null
2	2	1	1/	1
3	3	2	1/	1
4	2	3	1/	2
5	4	1	1/2/	1
6	5	2	1/2/	1
7	6	1	1/3/	1
8	4	2	1/3/	1
9	4	1	1/2/	2
10	5	2	1/2/	2

ways more than or equal to the path is the number of occurrences of child nodes.

3.1.1. Node Updates

Definition 2: N_Postion value consists of Numbers and letters, in the form of n_Postion value it is continuous presence of letters (or digital) known as a letter (or number) fragments, n_Postion LNPV value length is the sum of number of segments of letters and Numbers.

N_Postion value can be added and subtracted 1, which only produces an effect on the end of the segments of letters or Numbers when it calculated. Digital comparison rules is applied in n_Postion, as for the letter part, the comparison corresponds ASCII letters, the comparison of letters and Numbers distinguished positive and negative, all subtractive digits (or letters) are less than those without a minus sign (or letters). For example 1<2 , -1>-2 , a<b , -a>-b. The n_Postion value comparison results with location of the number (or letters) pieces, it is only when the current digital segment (or letters) is equal to the next that the Compare continue. For example: 101ab and 101cd, their first segment are same, then compare their second ones, because ab is less than cd, 101ab < 101cd.

There are four situations in Node updates

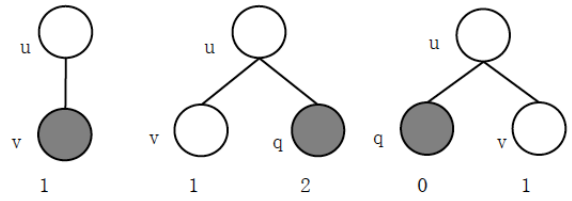
1. New added nodes have no brothers, then the new node n_Postion value is 1. As shown in Fig. (**2a**).

2. The new added nodes have and only have left brothers, the new nodes n_Postion value is left brothers n_Postion value plus 1, as shown in Fig. (**2b**).

3. New added nodes have and only have the right brother, then the new nodes n_Postion value is right brother value minus 1, as shown in Fig. (**2c**).

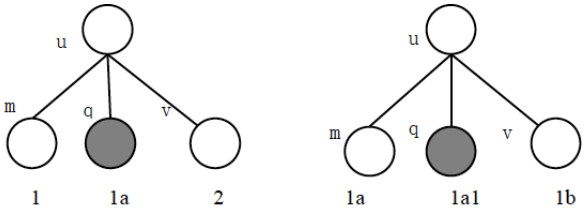
4. The new added nodes have left brothers and the right brothers at the same time, discussing in the following three conditions:

(1) Left nodes LNPV value equal to the right brother LNPV values. If left brother nodes N_Postion values ends with digits, the new node n_Postion value is the value of itself plus a, as shown in Fig. (**3a**); If left brother nodes n_Postion values closing in on the letter, the new nodes n_Postion value is left brothers n_Postion value behind the Numbers 1, as shown in Fig. (**3b**).



(**a**) Has no brother (**b**) Has left brother (**c**) Has right brother

Fig. (2). Add new nodes.

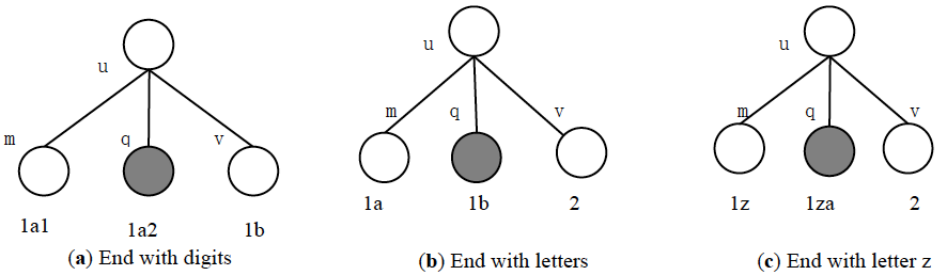


(**a**) End with digits (**b**) End with letters

Fig. (3). Left nodes LNPV value equal to the right brother LNPV values.

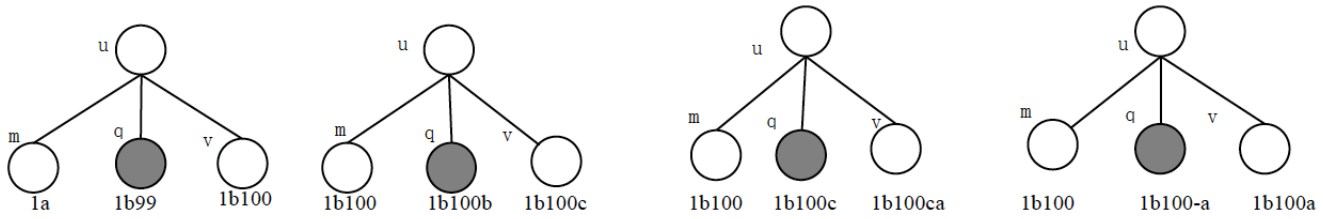
(2) The Left brother's LNPV values is more than the right one. If the left brother nodes n_Postion values closing in digits, then plus one directly, as shown in Fig. (**4a**). If it ends with letters, the last letter should first converted to ASCII, ASCII value plus one, and then converted to the corresponding letters, as shown in Fig. (**4b**). If left his brother's last letter z node n_Postion values, the new nodes n_Postion value is left brothers behind n_Postion value directly with letters a, as shown in Fig. (**4c**).

(3) The left brother's LNPV value is less than the right brothers. If the right brother's n_Postion values closing in digit, the new nodes n_Postion value is the right one minus



(**a**) End with digits (**b**) End with letters (**c**) End with letter z

Fig. (4). The left brother's LNPV value is more than the right brothers.



(**a**) End with digits for left brother (**b**) End with letters for right brother (**c**) More than one letter (**d**) Only one letter a

Fig. (5). The left brother's LNPV value is less than the right brothers.

1, as shown in Fig. (**5a**); If the right brother n_Postion values ends with letter, the last letter should be converted to ASCII first, then make ASCII value minus 1, and then converted to the corresponding letters, as shown in Fig. (**5b**); If the right brother n_Postion value is the last letter a fragment and letters to more than one letter, n_Postion value is right brother nodes removed at the end of the letter a is a new node n_Postion value, as shown in Fig. (**5c**); If the right brother nodes n_Postion value the last letter is a fragment is only one letter a, and letters are letters a transform into n_Postion value after the "-a" n_Postion value with the new node, as shown in Fig. (**5d**).

SDUE code has the following properties:

Property 1: Nodes a and b, if a is the prefix of path-FreId=b.pathFreId and a.pathNameId+ '/'+a.n_NameId is b.pathNameId, a is the ancestor nodes of b, b is the offspring of a.

Property 2: Nodes a and b, if a.pathFreId=b.pathFreId and a.pathNameId+ '/'+a. n_NameId=b.pathNameId,then a is the parent node of b, b is a's child node.

Property 3: Nodes a and b, if a.pathFreId= b.pathFreId and a.pathNameId=b.pathNameId,then a is the brother node of b, if a.n_Postion<b.n_Postion, then b is a's child node.

3.1.2. Data Query

In SDUE coding, first of all transform the XML data into structured data stored in a relational database, the data of queries in relational database, query processing steps follows:

(1) Decompose the query path into Q1 and Q2, Q2 represents the last query path's node name and number, the name number information query path Q1 said other nodes;

(2) To determine whether a query path contains the predicate condition or not, if it doesn't include the predicate condition, shift (3);If containing the predicate condition, shift (4);

(3) Access to relational tables node position, with nodes position in the table name by Q2 number comparison, Q1 and node location path name number comparison of relational tables, return and to satisfy the two conditions of vertex set R, query over;

(4) Extract the information of query path predicates, through nodes table and attribute table to find out the satisfaction value conditions set V of nodes and attribute node set A condition;

(5) Through Q1, Q2 and V, respectively, and the node location path name in the table number, name of node number and node number comparison, it is concluded that satisfy the three conditions of node collection R1;

(6) Extract nodal set R which contains A path in R1, R is result sets, query over..

4. THE EXPERIMENT AND ANALYSIS

The experiment happens on a PC(2.7 GHz processor,2GB memory), Windows XP operating system, the MySQL database, using the Java programming language. Experimental data are XMark generated XML test data.

4.1. Data Encoding

In the experiment, set the XMark coefficient is 0.087, 0.174, 0.260, 0.346, 0.432, and generate the size of 10126 KB, 20215 KB, 30267 KB, 40146 KB, 50066 KB documents, use interval coding XISS, prefix encoding LSDX and SDUE code to encode XML documents, respectively. The situation that coding takes up relational database storage shown as Fig. (**6**).

The experimental results show that encoding takes the relational database storage space increasing with the the XML document, for the three kinds of coding mode above, the same size of the XML document which takes up the largest relational database storage space LSDX coding, because the code node information of LSDX contains all the ancestor's

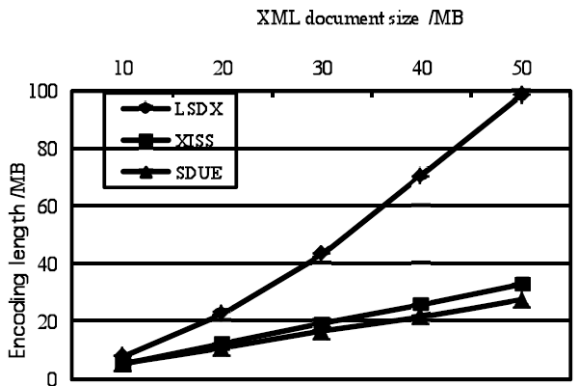


Fig. (6). Encoding length.

nodes path information, a lot of storage space could be used to represent the position relationship between nodes, with the increasing of the depth of the nodes, it will make the data length coding way is nonlinear growth. Interval coding and SDUE XISS coding length have no direct relationship with the depth of the node and the relational database occupied not much storage space. In SDUE coding, coding for the first time the location of the node's relationship is only represent in numerals, A fewer number can represent a large number of node position, The complexity of space is O (log10n), the number of node represent the path, the depth of the node has less influence on the code's length, XISS and SDUE coding length increased with the node number and depth linearly.

4.2. Data Update

The experiment generates the size to 1 MB, 5 MB, 10 MB of XML documents respectively. By adding LSDX and SDUE code 10, 100, 1000 respectively, the results as shown in Table **5**.

That can be found by experiment, with the increase of the number of new nodes and the document size, it has more and more advantages in the speed of SDUE code updating. Due to record LSDX coding node is prefix code, before inserting new node, it needs to transform the path of the new node information into prefix code, however, it would take much time to finish it so the efficiency of data updating is not high. Because SDUE coding has recorded the node path number, it can be very quickly to determine the insert position for just transform the node name and its number, then most of time can be used on the insert.

4.3. Data Query

In this experiment, the size of 100 MB of XML documents, document contains 1442042 nodes, set up five different query condition in query use cases, as shown in Table **6**.With SDUE code query algorithm, XISS algorithm and B + tree query algorithm are compared, the query results are shown in Fig. (**7**).

Table 6. Query cases.

Number	Query Path
Q1	//regions//item
Q2	/site/catagories//description/parlist/listitem/text
Q3	/site/regions/Europe/item[@id=number1@]
Q4	/site/person/name[Tomes]
Q5	//itemref[@item=item1@]/price[113.87]

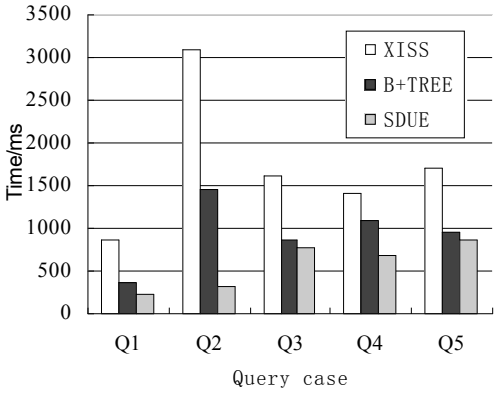


Fig. (7). The comparison of query time.

Table 5. Node update time.

Document Size	Node Number	Add Node Number	Update Time/ms	
			SDUE	LSDX
1MB	14974	10	211	463
		100	561	1871
		1000	1589	7526
5MB	72156	10	213	963
		100	564	3563
		1000	1596	13522
10MB	145864	10	216	1659
		100	568	6015
		1000	1604	24560

Through experiments, the query performance SDUE coding algorithm is superior to the other two search algorithm. XISS query uses a structured connection, which needs to traverse the nodes, and has low query efficiency. B + tree algorithm uses the pointer to find the next node connected to participate, and do not need to traverse the entire node set, which reduces the time complexity of the structure of the connection. But the intermediate result set size and query path length still affect query efficiency. In SDUE coding query algorithm, by converting the path name to the path name ID query, the query path length does not affect query efficiency. The time complexity is O (n). When the query path has predicate conditions, it takes extra time to deal with the predicate conditions. But the overall performance is still higher than the other two algorithms.

CONCLUSION

Proposed SDUE coding by decomposing node route information avoids repetition of recording information and has a high space utilization. It uses path information with the node name of the node number to indicate and distinguish path name in same different paths by setting the frequency of the path name, which can effectively support node ancestor, indicating offspring parent-child relationship. By numbers and letters which represent fraternal relations between nodes together, flexible combination of numbers and letters can effectively support data update. In terms of path query, since the structure can avoid join operations, SDUE coding also has good query performance.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Houliang Xie (1974), male, vice professor, His research interests include knowledge integration, XML encoding.

REFERENCES

[1] Z. Can-Wei, F. Shao-Rong, and L. Zi-Yu, "Dynamic containment labeling scheme for XML", *Journal of Software*, vol. 23, no. 3, pp. 582-593, 2012.

[2] Y. Li, M. Zong-Min, and L. Jian, "XML modeling of fuzzy data with relational databases", *Chinese Journal of Computers*, vol. 34, no. 2, pp. 291-303, 2011.

[3] F. Xue, and M. Zong-Min, "Construction of DICOM SR relational database based on foggy diagnostic model and representation with XML", *Journal of Northeastern University*, vol. 34, no. 4, pp. 486-489, 2013.

[4] S. Y. Noh, S. K. Gadia, and H. Jang, "Comparisons of three data storage models in parametric temporal databases", *Journal of Central South University*, vol. 20, no. 7, pp. 1919-1927, 2013.

[5] J. Ren, X. Yin, and X. Guo, "A dynamic labeling scheme for XML document", *Journal of Communication and Computer*, vol. 3, no. 5, pp. 61-65, 2006.

[6] D. C. An, J. Y. Kim, and S. Park, "Access control and labeling scheme for dynamic XML data", In: *The 3rd International Conference on Grid and Pervasive Computing Workshops*, 2008, pp. 329-334.

[7] W. Hongqiang, L. Jianzhong, and W. Hongzhi, "Processing XPath over F&B-Index", *Journal of Computer Research and Development*, vol. 47, no. 5, pp. 866-877, 2010.

[8] L. De-Xi, W. Chang-Xuan, and L. Xi-Ping, "A snippet retrieval srategy based on element weighting model", *Chinese Journal of Computers*, vol. 36, no. 8, pp. 1729-1744, 2013.

[9] H. Jing, L. Jiaheng, and M. Xiaofeng, "Effieient XML keyword query refinement with meaningful results generation", *Journal of Computer Research and Development*, vol. 47, no. 5, pp. 841-848, 2010.

[10] Z. Jun-Feng, and M. Xiao-Feng, "Keyword search on XML Data: a survey", *Chinese Journal of Computers*, vol. 35, no. 12, pp. 2459-2478, 2012.

[11] F. Linlin, L. Husheng, and G. Hongyu, "A pipelinning solution of XML Holistic Twig query", *Journal of Computer Research and Development*, vol. 48, no. z2, pp. 105-113, 2011.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015