Effective Decomposing Approach for Historical XML Documents

Ming Shien Cheng [1], PingYu Hsu[2] and MinTzu Wang[3]

[1, 2,3]*Department of Business Administration, National Central University*
[1]*984401019@cc.ncu.edu.tw*, [2]*pyhsu@mgt.ncu.edu.tw*, [3]*93441024@cc.ncu.edu.tw*
[1]*Department of Industrial Engineering and Management,*
Ming Chi University of Technology
[1]*mscheng@mail.mcut.edu.tw*
[3]*Department of Information Management,*
Technology and Science Institute of Northern Taiwan
[3]*mtwang@tsint.edu.tw*

Abstract

Recently, XML is widely used as the de facto standard for data representation and exchanging in Internet. In 2006, office application groups such as OpenOffice.org and Microsoft office both adopted XML as the main data storage format. Historical XML documents often have tiny differences between versions, but are stored individual independent space, so the abilities for efficient storing historical office documents are become a growing issue. This paper introduces an efficient way to decompose multi-version XML documents and store effectively for advanced retrieving. Not only effective storage space but also keeping the integral of original documents is the characteristic of our research. It minimizes the change of data content and structures when transmute historical XML documents. For enterprises, the approaches of our research can manage electronic documents in proper way and all messages in document were preserved to reuse.

Keywords: *OpenOffice.org, XML, Historical document, Storage structure*

1. Introduction

Recently the preservation of the electronic document has become an important issue for government organizations and enterprises; especially it needs to consider the difference between different office software of having various operating systems (OS), brands or versions, as well as the compatibility among them that shall be solved in advance. In 2006, two world-leading players of the office software, OpenOffice.org and Microsoft Office, have coincidently launched the data storage method with adopted the Extensible Markup Language (XML)[11] as the main body, and then XML has gradually become a standard format. XML is featured in with extensibility, structuralization and verifiability, which cannot be restricted by computer platforms and programming languages; therefore, it has become one of those formats that recommended by World Wide Web Consortium (W3C)[16] Based on the features of XML, the office software, such as OpenOffice.org and Microsoft Office, have developed to make enterprises' document to have the uniform format in order to achieve the goal of properly managing, delivering, preserving and processing these electronic documents.

Impress is a presentation tool of the OpenOffice.org, even it is belonged to a part of the office software, it is not supported enterprises' daily business operations, but includes the meeting report and product presentation. Most presentation documents have showed high similarity among individual versions with only making slight adjustment in contents or the order of layout; therefore, the utilization of storage space is very inefficient, and the management of files has become complicated in the future.

Thus, the purpose of this study is to make use of the open storage framework for XML documents to develop the algorithm that can process the multi-versions of presentation documents and seek for the storage method with high efficiency. In addition to save the storage space of document, it shall also be able to maintain its integrity.

In this XML study, the researcher has introduced the concept of threshold value for the first time. For the previous researches on processing the text or structure storage, the scope of data is restricted frequently. Thus, when there are more document formats that carried out the processing are based on XML, the processing standards will be definitely different in compiling and editing different electronic documents. Therefore, the application of threshold value can greatly increase the flexibility of document storage with responding to different requirements.

2. Literature Review

This Section will explore and discuss relevant literature on the research objective of this study. Firstly, it will introduce Impress presentation software of OpenOffice.org, and then it will explore the storage structure of XML document from many relevant literatures, including the version control.

2.1. XML Open Storage Structure

XML had been promoted by Unicode Consortium [15] and used to support the comprehensive natural languages. At the same time, XML is also one of the international standards that passed ISO certification [12, 18]; hence, it has become a optimal medium for the data exchange. Usually, XML will be compared with HTML. In the current network environment, there are more and more web pages that wrote by XML, thus, as compared with HTML, there are some dissimilar characteristics of [7]:

XML is extensible without restricting to the fixed tags.

XML is emphasized on the meaning of data rather than the presentation of data.

The syntax definition of XML document shall be correct format (well-formed) and legal (Formally validated); thus, it shall conform to the definitions of XML Standards [18], including a copy of XML document can only have a root element, and the initiated tag of the nested structure shall have the respond termination tags.

2.2. Relevant Exploration of XML Document Storage

The design purpose of XML is to transmit and store data; thus, when most of XML document are processing, including the document modification, increase and delete and random access of nodes, and we called these actions were “Parsing”. There are two methods of Parsing: one is the Document Object Model (DOM) [17] and the other is Sequential Access XML Parser API (Simple API for XML, SAX) [13]. The framework of DOM is a standard that established by W3C, it has featured with the independence from languages and platform. When parsing the XML document, transform the elements, attributes and texts of the document into a tree structure and store in the memory. Each node shall be regarded as an individual object and included the embedded value which can be operated programmers; in addition, a clear structure and easily understand will be its advantage. For SAX, it is a set of techniques that regarded XML document as the streaming interface, when downloading XML files with the sequential processing methods, and it will access the document with using the commands that set by programmers. However, it cannot be modified or accessed at will, as compared with DOM for using the tree structure to store in memory and occupied several folds of storage space for the original document, SAX is able to access any XML document regardless of its size, the designer's self-build model showed that when it only needs part of XML document, SAX will save even more storage space.

[10] has also proposed the version control of XML, it will take the source XML data with based on the edit-based method to dismantle; however, this method needs to divided each element of this document into an individual object; however, the recover cost is too huge, thus S. Y. Chien et al. have proposed the usefulness-based clustering, which duplicated the storage in other pages. Even this process will be consumed some storage space, it still can make the recovering speed to become more efficient. However, this method is unable to solve the basic issue about controlling the traditional versions. And, to achieve the simplified storage space, it needs to dismantle the document of the source data into a minimum unit of each element, even it has increased the storage space with the clustering method to accelerate the recovering speed; however, the simplification of the primary space that still needs to be consumed more processing cost, thus, it showed an inefficient flexibility of processing document.

As for the latest literature about processing the historical version, they have emphasized on the combination with the data mining [4, 5, 6], within the ever-changing environment, XML data will be different by following different versions, data structures and texts will also be changed dynamically. Then, made use of the algorithm that similar to Apriori [8] and FPG [3] to mining the commonly changing part of the substructure of XML document [6]. From the dynamic XML version document to extract the slight change in time, and by means of these information to conduct the future forecast and application; and [5] thought that by following with the changes in versions, it will be existed a similar model among those substructures of XML document, as known as FCSPs, and it can sort out the longest FCSPs, from the resemble FPG algorithm; [4] has defined the changed substructure by following the changed version as the FRACTURE, then adopted the Level-wise and Divided-and-Conquer methods to mine the longest FRACTUREs; in addition, the mined result can be used to the future applications, such as the index or clustering, etc., of XML.

The recombination of this document is a area that received less researches [9], or only proposed those methods that shall be conducted the recombination but without any basis. Till the thought that made by [2], when querying the document, it shall show the necessary result of such user at last; therefore, it has to indicate any node and regard such node as the root node to conduct the document recombination on the stored data that adopted the relational database with using the sequential coding pattern, as well as increasing the querying efficiency.

3. Algorithm

The basic assumptions of processing documents in our study: texts simplification; unnecessary parsing with aiming at the grammar issues; different versions being pretty identical to each other; only saving the different part of each version to preserve integrity.

3.1. Data Structure

The data structure is mainly composed of the coding techniques and cooperated with the threshold value to divide such document, and then preserve it in the form of matrix; among which, the access of original data is adopted SAX [13] to conduct the sequential access, and the coding has maintained the parent-son relationship for the tree structure, and conducted the processing in accordance with this basis [1].

3.2. Documents Processing Algorithm

Algorithm is mainly divided into 3 parts: f_list, XSS and Data_recovery algorithm.

3.2.1. f_list Algorithm

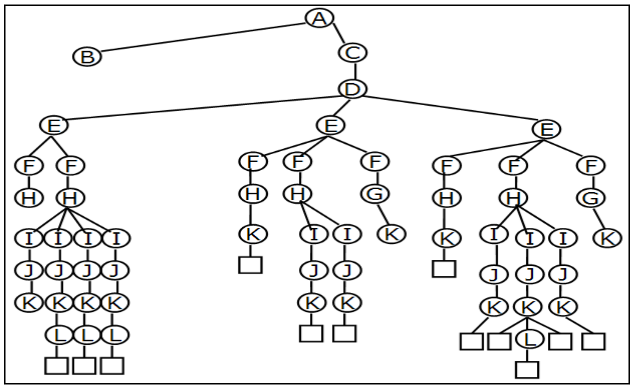


Figure 1. Example of Impress Presentation XML Document Structure

As showed in Figure 1, the introduction of the threshold value concept, firstly, calculate the splitting value for document tags; among which indicated the displayed texts, and English alphabet indicated the name of tags, if the names of tags are identical to each other, but their added attributes are different, then they shall still be regarded as the same tags. Then, such document has 12 tags, assumed A tags $=t_1$, B tags $=t_2$,…,L tags $=t_{12}$.Splitting value indicated that the level of tags for splitting such document. Therefore, as for the splitting value of the i_{th} tag, its formula will be $f_i = \frac{C(t_i)}{C(t_i)+S(t_i)}$, C (t_i) is the total sum of i_{th} tag, and S (t_i) is the total amount of sons and grandsons for t_i . Among which, the text will be regarded as a tag. For example, E tags' total amount is 3, and the total sum of its sons and grandsons tags will be 63; thus, the splitting value of E tags will be f_5=3/(3+63)=1/66=0.0455. If taking E tags to be the splitting point of this document, then such article can be divided into 4 parts, the sub-tree is composed of tags A, B, C and D, and 3 sub-trees that used E tags as the root elements. As for the splitting value of tags is greater than 0.35 的 tags, it has a power splitting capability, but it is easily to split articles into very scrappy; for example, the tag K's f_{11}=13/29=0.4482, tag L's f_{12}=4/8=0.5. If taking tag K or tags L as the splitting point, thus the document processing will cause a burden on such system, thus, we chose to enter the threshold value into the system, which needs to be entered into $\sigma_{\min}$ and $\sigma_{\max}$ respectively; among which, $0 \le \sigma_{\min} \le \sigma_{\max} \le 1$, for different types of document, it needs to decide the threshold value while the first version entering into the system. Among which, $\sigma_{\min}$ is the lower limit of the threshold value, that can remove those tags without splitting capability. $\sigma_{\max}$ is the upper limit of the threshold value; however, establish the suitable upper limit that can avoid articles from spilt too scrappy. Therefore, if it assumed to set the upper and lower limits for the threshold value $\sigma_{\max} = 0.25$ and $\sigma_{\min} = 0.025$, and then the calculated splitting value is shown as follows:

Table1. Tags Splitting Value

Tag	Splitting Value	Tag	Splitting Value
A (t_1)	$f_1 = 1/(1+69) = 1/70 = 0.0143$	G (t_7)	$f_7 = 0.5$
B (t_2)	$f_2 = 1/(1+0) = 1/1 = 1$	H (t_8)	$f_8 = 0.1176$
C (t_3)	$f_3 = 1/(1+67) = 1/68 = 0.0147$	I (t_9)	$f_9 = 0.2195$
D (t_4)	$f_4 = 1/(1+66) = 1/67 = 0.0149$	J (t_{10})	$f_{10} = 0.2813$
E (t_5)	$f_5 = 3/(3+63) = 3/66 = 0.0455$	K (t_{12})	$f_{11} = 0.4482$
F (t_6)	$f_6 = 8/(8+55) = 8/63 = 0.1270$	L (t_{13})	$f_{12} = 0.5$

Tag's splitting point is conformed to the threshold value, and tags are E, F, H and I, and the original XML document can be split into the sub-tree as follows:

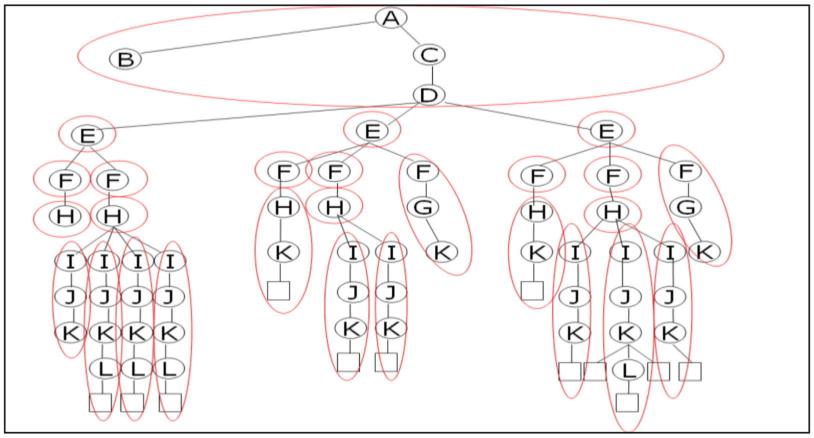


Figure 2. XML Document Sectional Split

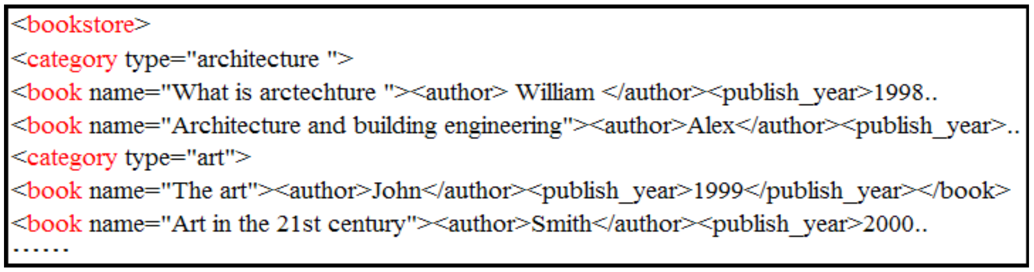
Thus, the detailed f_list algorithm is shown as follows:

f_list **algorithm** (Find Splitting Tags)

```
Input: (1) $V_1$ (Version one of XML document)
            (2) The threshold of splitting( $\sigma_{\min}, \sigma_{\max}$ ( $0 \le \sigma_{\min} \le \sigma_{\max} \le 1$)
Output: Splitting tags of XML document
Scan $V_1$ and $T=\{t_1, ,\ldots,t_n\}$ // T are the set of $V_1$'s all tags //
For each tag $t_i$ in $T$
  Count $C(t_i)$ and $S(t_i)$
    $f_i = \frac{C(t_i)}{C(t_i)+S(t_i)}$
    IF $\sigma_{\min} \le f_i \le \sigma_{\max}$
          THEN add $t_i$ to $F$ // $t_i$ is the splitting tag, and F is the set of all $t_i$ which splitting
      value match the threshold //
    End if
Next
Sort $F$ by splitting value ascending order into f_list
```

3.2.2. XSS Algorithm

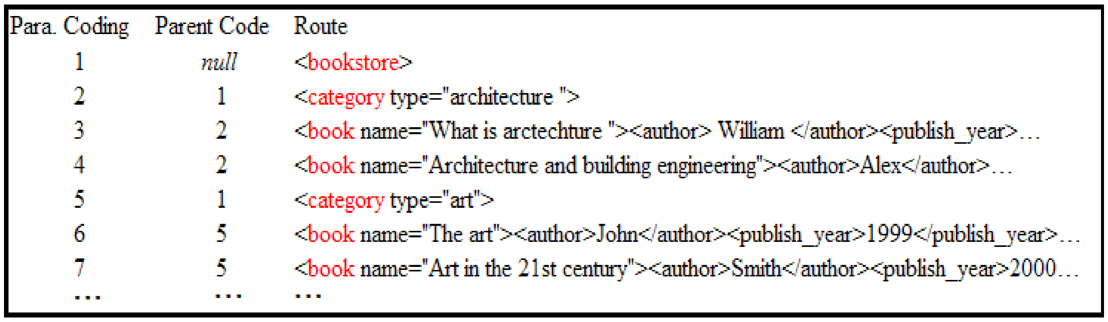
Makes use of the f-list algorithm to compute the splitting point for such type of XML document, and it is a pre-operating for the XML Storage Structure (XSS). XSS algorithm is adopted the splitting point for the f_list of XML document, and continuously carry out the splitting process on XML historical version document, XML document as shown in Figure 3, if assumed the splitting point is tags <category>and<book>, then such part of XML can be split as follows:



```
<bookstore>
<category type="architecture ">
<book name="What is arctechture "><author> William </author><publish_year>1998..
<book name="Architecture and building engineering"><author>Alex</author><publish_year>..
<category type="art">
<book name="The art"><author>John</author><publish_year>1999</publish_year></book>
<book name="Art in the 21st century"><author>Smith</author><publish_year>2000..
……
```

Figure 3. Take <category> and <book> to be Splitting Points

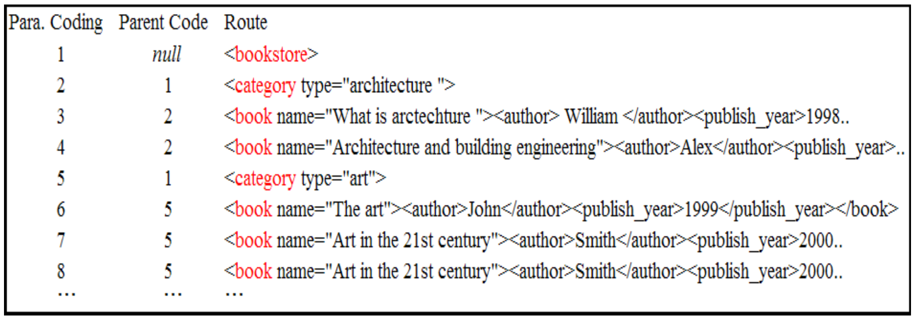
The selection of splitting point and threshold value will be the confirmation for the degree of easy modification of document; thus, when the upper limit of the threshold value is bigger, then the splitting degree of word paragraph will become more detailed. After processing such document, it will conduct the unique Paragraph coding for those word paragraphs that split out of each document, and then record the parent relationship with such word paragraph, and the relationship chart of this section, as shown in Figure 4:



Para. Coding	Parent Code	Route
1	*null*	<bookstore>
2	1	<category type="architecture ">
3	2	<book name="What is arctechture "><author> William </author><publish_year>…
4	2	<book name="Architecture and building engineering"><author>Alex</author>…
5	1	<category type="art">
6	5	<book name="The art"><author>John</author><publish_year>1999</publish_year>…
7	5	<book name="Art in the 21st century"><author>Smith</author><publish_year>2000…
…	…	…

Figure 4. Joining Word Paragraph Coding

Tag <bookstore> is the root element of such XML document; therefore, its texts coding is 1, and since it is a root element, thus the field of ParentCode will be null, likewise, paragraph<category type="architecture” >’s texts coding is 2, and since such paragraph is the nested structure for the <bookstore> tags; thus, its parent paragraph will be coding 1<bookstore>, and so forth to complete coding. This relationship table of paragraph will be the import value of XSS algorithm. Before starting the algorithm processing, it has firstly conducted the transformation, to transform Parent Code into the matrix of subparagraph coding, and the reasons are as follows:



Para. Coding	Parent Code	Route
1	*null*	<bookstore>
2	1	<category type="architecture ">
3	2	<book name="What is arctechture "><author> William </author><publish_year>1998..
4	2	<book name="Architecture and building engineering"><author>Alex</author><publish_year>..
5	1	<category type="art">
6	5	<book name="The art"><author>John</author><publish_year>1999</publish_year></book>
7	5	<book name="Art in the 21st century"><author>Smith</author><publish_year>2000..
8	5	<book name="Art in the 21st century"><author>Smith</author><publish_year>2000..
…	…	…

Figure 5. Another Possible Cutting Situation of XML Document

Assumed the relationship table that dissolved from the original XML document as shown in the Figure 5, then, coding 7 and 8 have the same text contents, in order to achieve the high-efficient simplified XML document, the coding 8 will be removed; however, when conducting the document recover, the information of such word paragraph will be no longer appeared. Thus, after the paragraph relationship table imported XSS algorithm, it needs to transform the subparagraph coding matrix firstly. Assumed r_y indicated every word paragraph, thus rc_y and rf_{cy} are the corresponding coding and parent paragraph coding for r_y; for example, r_2 is <category type="architecture" >, and rc_2=2 and rfc_2=1 are the corresponding codes. After transformed the subparagraph coding table into a matrix, as shown in Table 2, in such matrix, $rcc_{x,y}$ indicated the subparagraph coding, x, y are indicated the horizontal coordinates and vertical coordinates respectively, values of y shall be corresponding to the word paragraph of the paragraph relationship table; for example, r_5 is <category type="art">, thus $rcc_{1,5}$ =6, and its first word subparagraph is r_6, likewise, $rcc_{2,5}$ =7, $rcc_{3,5}$ =8. If such word paragraph is leaves node, which meant, except parent paragraph, that it never have the subparagraph, as shown in Table2.

Table 2. Transformation of Subparagraph Texts Coding

Para. Coding	Parent Code	Route	Child Code Matrix			
1	*null*	<bookstore>	2	5		
2	1	<category type="architecture ">	3	4		
3	2	<book name="What is arctechture ">…				
4	2	<book name="Architecture and …				
5	1	<category type="art">	6	7	8	
6	5	<book name="The art"><author>John…				
7	5	<book name="Art in the 21st century">…				
8	5	<book name="Art in the 21st century">…				
…	…	…	…			

```
Sub Convert ( $v_i$ )
    For each $r_v$ in $v_i$ paragraph relation table where it hasn't converted
        Using $rc_v$ and $rc_v$ to generate Child Code Matrix
        If any $r_v$ doesn't have child then
            $rcc_{1,v}$ is empty
        End if
    Next
End Sub
```

After transformed the subparagraph coding into a matrix, $r_7 = r_8$=<book name="Art in the 21st century">.., by means of the XSS algorithm processing, r_8 word paragraph and its corresponding coding will be cleared out, but the document's information has not lost yet, and it only needs to change $rcc_{3,5}$=8 into $rcc_{3,5}$=7. Thus, during the period of document recovery, it still can express the information of three word paragraphs of r_5.

Before carrying out the XSS algorithm, firstly define R is an assembly set that formed by r_1~ r_{y-1}. Set R is the text comparing basis of comparing with r_y. Where $RCC_y=\{ rcc_{1,y}, rcc_{2,y}, ...rcc_{x,y}\}$, and $RCC_y \subset RCC$; therefore, XSS algorithm is shown as follows:

***XSS* algorithm** (XML Storage Structure)

```
Input:  $v_1, v_2, v_3 .... v_n$ (paragraph relation table of history version of XML document from
        f_list splitting
Output:         XML Storage Structure (XSS)
Call Sub Convert ($v_i$ )
        For each $rcc_{1,v}$ which is empty
                Compare $r_v$ with R set
                If $r_v$ matches any $r_{v-n}$ |n:1<=n<y then
                Find $rcc_{x.k} = rc_v$
                Modify $rcc_{x.k}$ to $rcc_{v-n}$
                  End if
        Next
        For each $rcc_{1.v}$ which is not empty
                For each $col_x$ in Child Code Matrix
                        If $r_v = r_{v-n}$ | n:1<=n<y and
   { $rcc_{m.v}$ , $rcc_{m.v-n}$ } |m=1~x, $rcc_{m.v}$ , $rcc_{m.v-n}$ then
                                Find $rcc_{x.k} = rc_v$
                     Modify $rcc_{x.k}$ to $rcc_{v-n}$
                        End if
                Next
        Next
End if
Clear every $r_v$, , $rc_v$, , and $rfc_v$ and $rcc_{m.v}$ | m=1~x which $rc_v$ is not root number and $rc_v$ doesn't
exist in RCC
```

For example, as shown in Table 3, since the parent paragraph coding of these two word paragraphs have *null* value, thus it indicated that there are two documents; among which, at the time when XML document of v_1 has initially entered into the system, the word paragraph r_7 of coding 7 will be deleted by adopting the algorithm processing. After document of v_2 entered the system, firstly, it shall split the XML document of v_2 by using splitting point *f_list*, and store the split document, together with its texts coding paragraph relationship table into the system, as shown in the left bottom of Table 3. Among which, difference existed in 3 points for these 2 versions when comparing Version 1 and Version 2. Changes similar to r_{22} that can be called as the structural variation. Changes of r_{23} are the revisions of texts, and r_{25} is indicated the movement of word paragraph, which moved from the first nested structure of the original <category type="art"> to the third, but there is no such correction or modification made for the text contents.

Table 3. When Version 2 Document Entering the System

Para. Coding	Parent Code	Route	Child Code Matrix			
1	*null*	<bookstore>	2	5		
2	1	<category type="architecture ">	3	4		
3	2	<book name="What is arctechture ">…				
4	2	<book name="Architecture and …				
5	1	<category type="art">	6	7	7	
6	5	<book name="The art"><author>John…				
7	5	<book name="Art in the 21st century">…				
…	…	…	…			
18	*null*	<bookstore>	19	22		
19	18	<category type="architecture ">	20	21		
20	19	<book name="What is arctechture ">…				
21	19	<book name="Architecture and …				
22	18	<category type="arts">	23	24	25	
23	22	<book name="Art in the 20st century">…				
24	22	<book name="Art in the 21st century">…				
25	22	<book name="The art"><author>John…				
…	…	…	…			

By means of the data explanation, it begins to enter the algorithm of XSS. After document entered the system, it will immediately call up the subprogram, Transform (v_2), the newly added subparagraph coding matrix as shown in Table 3. When starting the simplification process to the word paragraph of v_2, it shall be split into 2 parts, the first part is that when the subparagraph coding $rcc_{1,v}$ of the word paragraph r_y is null value, as shown in the above figure, r_{20}, r_{21}, r_{23}, r_{24} and r_{25}, conducted the comparison with these word paragraphs and the set of $R_y = \{ r_1 , r_2 \ldots r_{y-1} \}$, r_{20} will compared with r_1 , $r_2 \ldots r_{19}$ for texts, if those compared results are different to each other, then it can regard such word paragraph as the newly added part of the historical version, or the part that has been modified or revised, such as r_{23}; if the compared results are matched, such as r_{20} is identical to r_3, then search the texts coding "20" of r_{20} under the subparagraph texts coding, and changed such subparagraph texts coding into the texts coding "3" of r_3, namely $rcc_{1,19}$=3; at the same time, as for the moving part of texts, such as r_{25}, since its texts have not yet changed or modified, thus, the compared results are identical to each other; therefore, according to the same method, $rcc_{3,22}$=3 will be changed into "6", and v_2 is changed as shown in Table 4.

Table 4. Step 1: When $rcc_{1,y}$ is a Null Value

18	*null*	<bookstore>	19	22		
19	18	<category type="architecture ">	3	4		
20	19	<book name="What is arctechture ">…				
21	19	<book name="Architecture and …				
22	18	<category type="arts">	23	7	6	
23	22	<book name="Art in the 20st century">…				
24	22	<book name="Art in the 21st century">…				
25	22	<book name="The art"><author>John…				
…	…	…	…			

After comparison, when $rcc_{1,y}$ equals to null value, it can enter to the second part, $rcc_{1,y}$ not equals to the null value, as r_{18}, r_{19} and r_{22}in the above figure, the texts comparing methods are also as abovementioned; however, except the same methods for texts, the subparagraph texts coding shall be totally identical, such as r_{19} and r_2, their texts are identical to each other, as well as $rcc_{1,19} = rcc_{1,2}$ and $rcc_{2,19} = rcc_{2,2}$; therefore, it can be showed that there is no change in structure or text, and it can look for the subparagraph coding "19" and change into the coding " 2" of r_2.

Through Step 1 and Step 2, it can immediately found the repeated part out of v_1 and v_2 in XML document, if it needs to achieve the structural simplification, it only needs to make each word paragraph coding rc_y not become the root word paragraph, or when the subparagraph texts coding is not appeared, delete the data for entire row, including r_y, rc_v , $rfc_{y,}$ and $rcc_{m,y}|$m=1~x, , and only retain the newly added part and changed/ modified part. The completely simplified XXS is shown in Table5.

Table 5. Simplified XSS Structure

Para. Coding	Parent Code	Route	Child Code Matrix			
1	*null*	<bookstore>	2	5		
2	1	<category type="architecture ">	3	4		
3	2	<book name="What is arctechture ">…				
4	2	<book name="Architecture and …				
5	1	<category type="art">	6	7	7	
6	5	<book name="The art"><author>John…				
7	5	<book name="Art in the 21st century">…				
…	…	…	…			
18	*null*	<bookstore>	2	22		
22	18	<category type="arts">	23	7	6	
23	22	<book name="Art in the 20st century">…				
…	…	…	…			

3.2.3. Data_recovery Algorithm

The main purpose of the texts structure for XSS is to retain and preserve the information of the original XML historical document, except for that the repeatedly appeared part can be deleted as well with simplifying the storage space for such document. When we need the information of this document, it can be then presented immediately; therefore, it needs to develop algorithm to recombine the words and paragraphs. Continuously make use of the XSS structure in Table 5 to describe the process of document recovery. If users need the Version 2 XML document, he/she shall enter the version number $j=2$ into the system, through the algorithm to determine the coding of Version 2 root word paragraph; in this example, it will be rc_{18}=18 and using the recursive method to process rc_{18}. Since the texts coding of the subparagraph is represented by matrix, thus, it will be adopted the Depth-First Search (DSF) method to recover the document.

When entering rc_{18} into the recursive algorithm, firstly to process the first subparagraph texts coding , the conditions of determination have the following 2 steps sequentially: first step, to recombine word paragraph into the document, such as when rc_{18}=18, duplicate and store <bookstore> into the XML file; in addition, transmitting the first word subparagraph coding $rc_{1,18}$=2 into its own recursive algorithm, and concurrently store the r_2 word paragraph<category type="architecture">, and so forth. The second step is to determine whether $rcc_{x,y}$ is a null value or not; if so, just exit the recursive algorithm then. Among which, the null value has indicated two meanings, one is indicated the searching that has completed to the leaves word paragraph, as $rcc_{1,23}$ in 0. Another situation has completely searched all subparagraph texts coding, such as $rcc_{3,18}$, $rcc_{4,22}$,. *Data_recovery algorithm* is shown as follows.

***Data_recovery* Algorithm** (XML document recompose)

```
Input:   XML Storage Structure (XSS)
   Selected historical XML document version number $j$
Output: Original XML document $v_i$
Main ( )
Find $v_i$ root number $rc_v$
Recursive ( $rc_v$ )
Return $v_i$
Function Recursive ( $rc_v$ )
For each $col_x$
         $paragraph\_n = cc_{x..v}$
         Copy and store $r_{.v}$ into XML document when entering function
         If $rcc_{x,y} \in \phi$ then
                  Exit for
         End if
         Recursive ( $paragraph\_n$ )
Next
```

4. Empirical Analysis

This section is mainly composed of 2 major parts: first part is to describe the data-oriented slides image file of the OpenOffice.org Impress presentation; and the second part is to analyze the results and data that came from those experiments.

4.1. Experiment Design

The data of this experiment is the slides that used to make presentation of OpenOffice 2.0 in Centrum der Büro-und Informationstechnik (CeBIT) on March 2006, the document contents included graphs, difference in font sizes, color usage, hyperlink and animation layout, and the total page number is reached to 37 pages, which is identical to the slide quantity of common presentation. Since the languages is German, thus, in order to make it to be easily understand, this experiment has transformed the original Germany version Impress document into the English version [15]; in addition to make such translation, the rest parts of this presentation have not yet changed and maintain the framework of the original document.

Besides, in order to increase the document's variability, selected 102 key words from those articles and randomly changed key words to assist in yielding different versions document.

4.2. Experiment Result and Analysis

Since the source data is only one part existed, thus it shall use the simulative methods to create different versions. Among those possible causes of yielding different versions, the presenter's preference will be different for each other, thus it cannot be simulated; and another factor is that the continuously increased page number while preparing the presentation slide file; as a result, based on such assumption to design the Experiment 1.

Experiment 1: Continuously Increasing Page Number for Each Version Document

Experiment Description: when simulating the users with compiling document, the process of gradually increasing the contents for such presentation. Since it only has a copy of the original text with a total page number is 37; therefore, it assumed that the OpenOffice Impress of v_1 only has 8 pages, and this part that may not changed ever will be completed in advance;

therefore, v_1 is contained the first 4 pages and ending 4 pages of the original document. Each version will be increased one page more than the previous one; thus, v_{30} is the original content, and the experimental result is shown as in Figure 6.

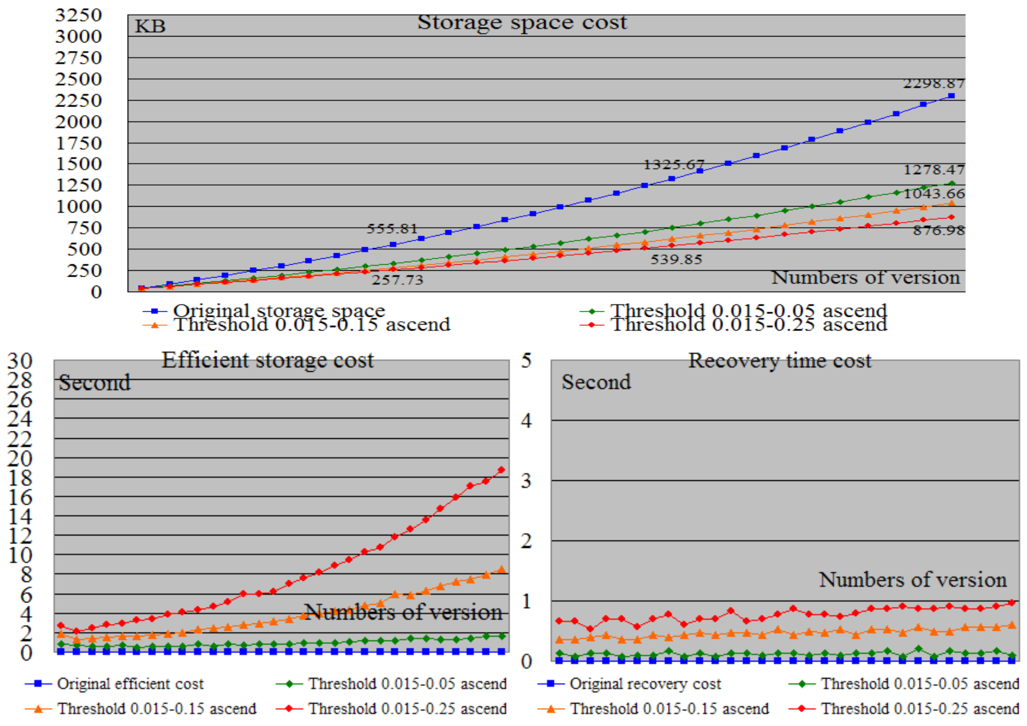


Figure 6. Document Pages Number Increasing Storage Space, Simplify and Recovery Time

Analysis on experimental result: the original storage space with accumulated 30 versions is 2298.87 KB, and if use the example of 0.015-0.25 to make explanation, in v_{10},, it may possibly save about 54% of storage space, and v_{20} will be about 59%; after completed 30 versions, it can be simplified about 62%; thus, the document will be saved more space (storage) by following the evolution of versions. Among which, the upper curved graphs showed that when the storing the original document contents, it shall have the non-preset effectiveness; such as the changes in font sizes and colors, and the increase in animation, etc., they will be made definition in XML, when increasing 1 page, except the storage of page, it still needs to contain the definition space for the special effect; in addition, when using the XSS algorithm to process, it will delete the repeated effect definition between versions, and the degree of upper curved will be slowed down, and the more versions will make more optimal rate of simplified space.

In addition to compare with the original file and the storage space after used the algorithm to make process, at the same time, the threshold value may also be changed as well, and then inspect whether there is any difference in the usability of space or not, as shown in Figure 6, the threshold values of the rhombus points are among 0.015-0.05; comparatively, the presentation document is adopted “pages” as the standard of minimum division. 0.015-0.15 are based on the “text boxes”, sphere points are around 0.015-0.25, which is indicated the text of each line, or namely the item sign, Bullet, as the minimum division standard. From the figure, there is only 0.1 for the difference in the upper limit of the threshold value, but the result will be greatly different. When changed the value of upper limit from 0.05 into 0.15, the improved efficiency will be more and the efficiency of its original simplification is not reached 50%. In this experiment, it has assumed that the version document is belonged to the increasing pattern, there is no change in document's contents, but the page number is increasing by the method of inserting page.

As for the time comparison, since the most original un-processed documents that don't need time to conduct the simplified and recovered storage space; therefore, it assumed that consumed zero second. Within the different threshold values for other lines, and the time that system consumed for processing, and this experiment increasing version document,; thus, when the upper limit of threshold value is bigger, then the number of word paragraphs of such version will be divided into a bigger number then. In addition, when carrying out the simplification for the structure, it will be consumed more computing capability of computer.

Experiment 2: Continuously Pages Number Decreasing for each Version Document

Experiment Description: In Experiment 1, it has assumed within an extreme condition, when presenter is increasing 1 page for each version to simulate the data; therefore, in Experiment 2, it will be adopted the same data, but within another extreme condition. At this moment, v_1 is the original version, and v_2 is 1 page less than v_1, and so forth, the v_{30} will only have 8 pages, and contents will be identical to v_1 in Experiment 1, as well as under different threshold values, it will effected the simplified document. Experiment result is shown in Figure 7:

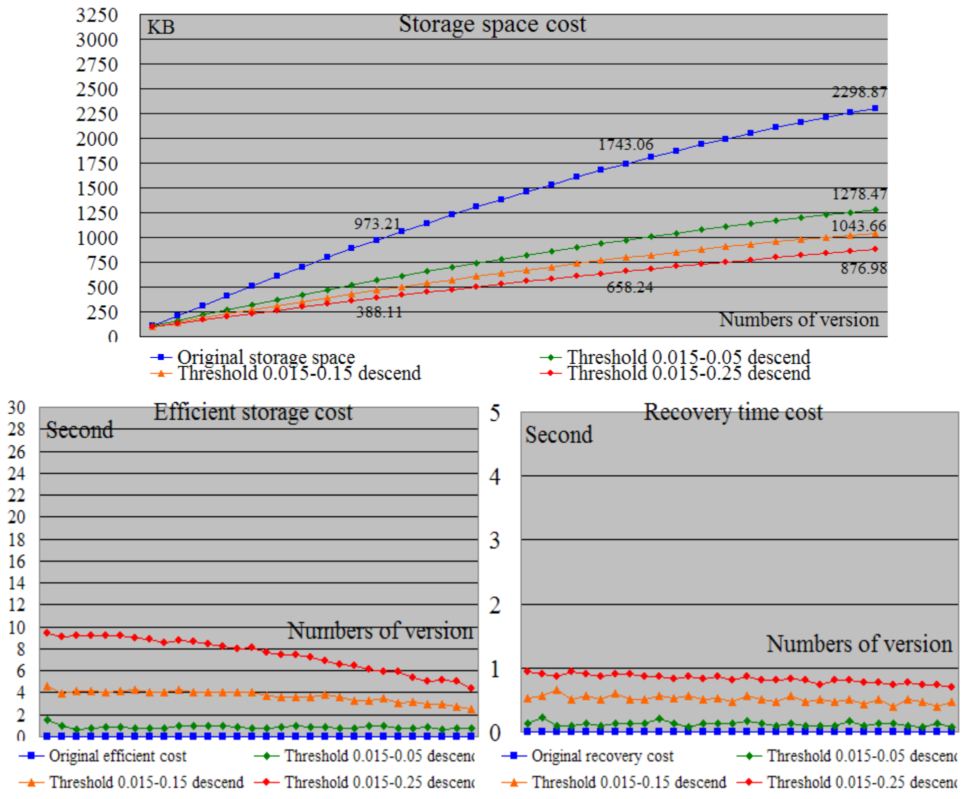


Figure 7. Document Pages Number Decreasing Storage Space, Simplify and Recovery Time

Experimental Result Analysis: as compared with Experiment 1, its graphs showed a concave down trend, the reason is that since the definition of format effectiveness has become less and less, after processed by using algorithm, the identical definition (repeated) will be deleted, thus the concave down trend will be promptly mitigated. Taking the same standards threshold values from 0.015-0.25 to compare Experiment 1 with Experiment 2, both of them will be saved about 62% of storage space after stored 30 versions; therefore, it can verify that such algorithm is applicable to the increasing or decreasing conditions among versions, and the simplification of document will be maintained.

The time of simplification will be displayed in opposite direction to the increasing version, since the number of versions is bigger, the data is less, and then the computing capability of simplified space that has consumed is efficient; moreover, the time of recovering document will be maintained within one second with slightly fluctuation, but it is still within the acceptable scope.

Experiment 3: Pages Number is Increasing and Replacing Texts with Key Words

Experiment Description: in the aforesaid experiment, it has simulated the single situation only; therefore, the Experiment 4 will simulate the situation that will be occurred more frequently everyday – by following the versions, users may have also increased the pages number, as well as modified the original text contents at the same time. Therefore, as for the data simulation, it will adopt the method of inserting 1 page into each version, that is, 8 pages for v_1, and 37 pages for v_{30}, as to the modification of the previous/ original text contents, it has also adopted the method of replacing with 10, 20 and 40 key words, respectively, to complete 30 historical versions document in order to inspect the coexistence for these two situations. Since in Experiment 1 and Experiment 3, regardless of pages number increasing or replacing key words, the threshold value 0.015-0.25 has an excellent ratio of saving the space; thus, this experiment is divided on the basis of the threshold value, as shown in Figure 8.

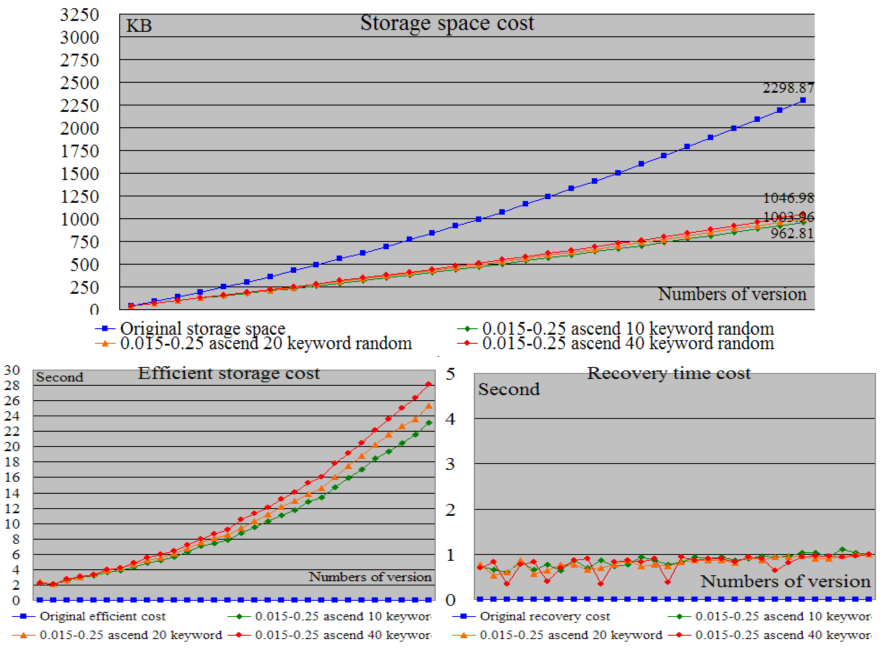
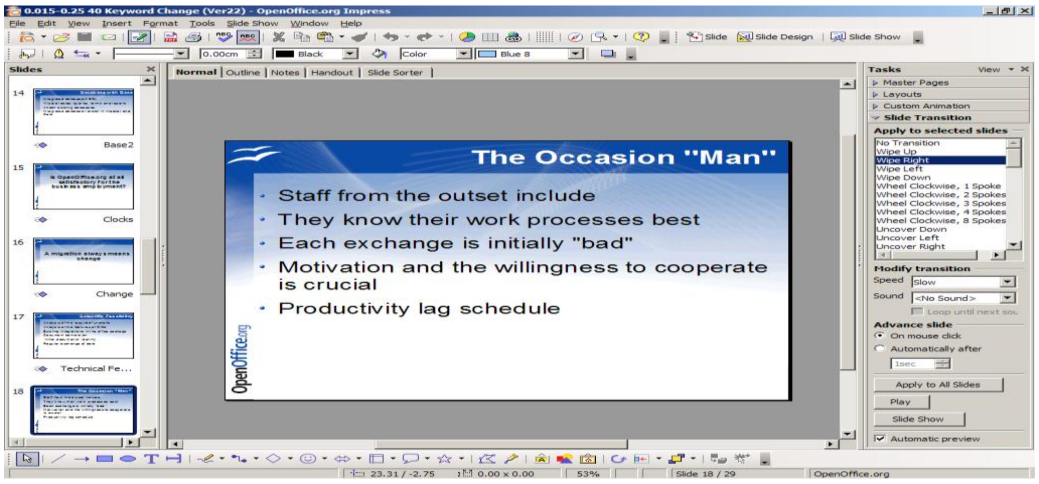


Figure 8. Pages Number is Increasing and Replacing Key Words for Storage Space, Simplify and Recovery Time

Experiment Result Analysis: as shown in aforesaid figure, it can find out an interesting phenomenon. When increasing the number of replacing key words from 10 to 20, and the storage space of file has increased a margin as same as increasing the number from 20 to 40; thus, it may changed back to the original words or phrases at the end of the process; thus, the systems needs to maintain the first time it appeared and to simplify the repeated part. Therefore, the trend of such phenomenon can be roughly seen, when replacing more key words, or there are more versions, and increasing repeated part; then the saved space is more efficient. As compared with the time that will be identical to the aforesaid experiment, the time of simplification will be depended on the number of data for such version, and its recovery time will maintain stable.

Experiment 4: Document Recovery

Experiment Description: this experiment is attempted to present that even after completed the simplification of document, the remained information is still able to make response at all time. This experiment is constructed at the threshold value 0.015-0.25 and replacing with 40 key words, which used to verify whether it would be recovered or not. The eighteenth page of v_{22} and the first page of v_{10} can be randomly selected and recovered as the screen as follows:


0.015-0.25 40 Keyword Change (Ver22) - OpenOffice.org Impress

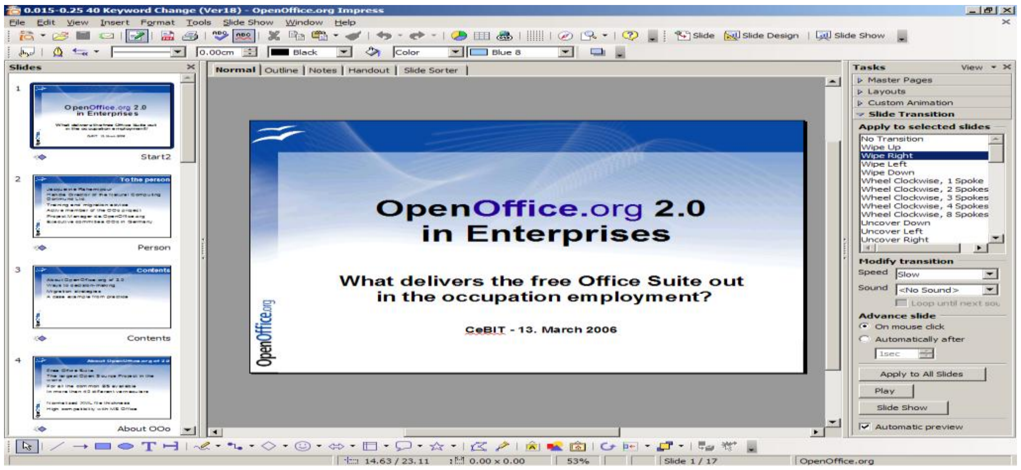
The Occasion "Man"

- Staff from the outset include
- They know their work processes best
- Each exchange is initially "bad"
- Motivation and the willingness to cooperate is crucial
- Productivity lag schedule

Speed: Slow
Sound: <No Sound>
Advance slide: On mouse click
Slide 18 / 29

Figure 9. Recover the eighteenth Page of v_{22}

Experiment result (1): through the eighteenth page of v_{22} as shown in Figure 10, after recovered, it included the ground colors, titles and the text inside the text boxes that will remained the size of the original version, as well as the setting of animation. The title of the original source presentation document [14] is The Factor “Man”, after randomly replaced 40 key words through the entire document, and the contents of this page have not been changed, but the title has replaced to The Occasion “Man”; therefore, from the screen of this experiment result (1), when recovering document, it has also maintained the information of the original document, and after compared the replaced document with the original document, there is no difference between them at all.


0.015-0.25 40 Keyword Change (Ver18) - OpenOffice.org Impress
File Edit View Insert Format Tools Slide Show Window Help
Slides
Normal Outline Notes Handout Slide Sorter
Tasks
Master Pages
Layouts
Custom Animation
Slide Transition
Start2
Person
Contents
About OOo

OpenOffice.org 2.0
in Enterprises

What delivers the free Office Suite out
in the occupation employment?

CeBIT - 13. March 2006

Apply to selected slides
Wipe Right
Modify transition
Speed Slow
Sound <No Sound>
Advance slide
On mouse click
Automatic preview
Slide 1 / 17

Figure 10. First Page of Recovered v_{10}

Experiment result (2): as compared the version document of v_{10} with the original source document [14], the title of OpenOffice.org 2.0 in Enterprises has not been replaced yet; however, the contents, carries and delivers of another text box have been replaced, and replaced business with occupation. In addition, in this experiment, the result, color, font, animation of the recovery document can be normally presented. As compared with the original v_{10}, the result will be identical to each other, loyally presented the version of replaced key words; thus, it has verified that those experiment are successful.

5. Conclusion and Future Research Suggestions

5.1. Conclusion

In this study, it has practically adopted the OpenOffice.org Impress presentation document to carry out the experiment, and utilize algorithms to simplify the contents of XML files in Impress. From those experiments to simulate the time when presenters making the slides, the situation of continuously increasing and deleting the document; however, under the condition of continuously changing in the structure of XML document, if it can reach to 30 versions, then it can save about 62% storage space. In addition, it is assumed that users have continuously changing the key words or sentences of the document, it can be clearly expressed even the scope of modifying key words is increased; however, the storage space may still be able to reach an excellent level. At last, it has conducted the Experiment 3 which has conformed to the actual situation, even its changing scope of the document is the maximum, and algorithms still can acquire identical part from files to make simplification. From those experiments we can understand that the establishment of threshold value, and make users to flexibly use the storage space, in accordance with each document with various topics, to establish different threshold values to optimize the storage space.
In the last experiment of this study, by means of integrating 30 versions of Impress presentation document into a single system, it can not only express to practically recover the historical document, but also conduct the optimal management to documents:

1. To respond to enter the era of great amount of electronic document exchange, this system can be sued to preserve the XML document that may be developed in an exponential growth in the future, and meanwhile, it can also slow down the rapid development in the future.

2. The trend of future document storage will be intended to change from strenuousness into lightness; as usual, the situation of files spreading will cause the maximum damage to the company; thus, well-processed the document storage; at the same time, documents can be integrated for the future management and application, including the authorized administrator of document who can modify contents only. For example, when a company is modifying its name, it can be made one-time correction in such system.

5.2. Example of Practical Applications

In addition to apply to simplify storage space, it also can be developed to be the future business models: homogeneity of information integration

1. Integration of Homogenous Data: even the current experiment result has showed in the Impress presentation document with a same topic, it still can simulate the practical possibility that can save about averagely 55%~70% of storage space; however, if it can focused on the presentation document with different topics in the future, since the similar structure and includes the identical definition of formats, thus it can promote to integrate those presentation document of various sequence with setting one threshold value.
2. Integration with Homeomorphous Data: In OpenOffice.org, except Impress, documents that conformed to ODF format storage, including the Writer word processing, Calc Spreadsheets calculation modules. In the future, all documents that conformed to XML format can be integrated to make document management to be more easily.
3. Recover Point Mechanism: it is the automatic storage mechanism of office document. Usually, in the production process, since the consideration of documents' storage space for systems, it may only automatically make one or two versions storage space; and, if any damage occurred, then those document will be invalid as well; therefore, it will make use of the automatic storage mechanism that proposed by this study, and set the time interval of carrying out the storage process to the document for once every 5 minutes. As a result, except for greatly decreasing the storage space, the information can be maintained for the document within every different time interval of storage.
4. Similarities Inspection: the storage system of this study can be the database for inspecting the promotion of plagiary in the future, since the behavior of document plagiary is usually reedited the contents of different pages in the document, or only changed the key words. However, these procedures will be difficult to implement manually. Therefore, it has only regarded the document as the historical version which may be involved in plagiary, if the document simplification of the Version 2 is high, then it may indicated that we can reasonably suspect the document of Version 2 which has the suspicion of plagiarizing Version 1.

5.3. Future Research Suggestion

The processing of storage space in the future that shall not be restricted to texts only, graphs may frequently appeared around us, and the structure of graphs may possibly evolved in the future; for example, the chemical molecular bonding, etc. When the graph needs to be stored, these version-evolved graphs in the space of computer memory is complicated and useless, but they will be preserved as the historic message; therefore, if this method can be promoted to graphs, and recorded the parent-son and evolving relationships for such combination that will make graph to be well-managed as well.

In the practical conditions, as for those industries that will extremely focus on their storage space, such as the retail business or manufacturing industry that needs bulk sales and purchase the components and parts, in order to be well-stored the transaction data, those factors cannot store the most detailed data into the database, but needs to regularly adopt the tape to make such storage. It will take a large quantity of time for the sequential access process when responding historic data; however, most of the historic data are similar and only the time points of occurrence are different. Thus, in the future, it only needs to use database to store the transaction data, and further it can be developed as a tool of data analysis in order to help with the business competitiveness as well.

References

[1] 林昌正，「多 XML 文件整合萃取工具之研究」，國立中央大學，碩士論文，民國 97 年。
[2] Chebotko, D. Liu, M. Atay, S. Lu and F. Fotouhi, “Reconstructing XML subtrees from relational storage of XML documents”, Proceedings of the Second IEEE International Workshop on XML Schema and Data Management (XSDM05), in conjunction with ICDE05, Tokyo, Japan, **(2005)** April.
[3] J. Han, J. Pei and Y. Yin, “Mining frequent patterns without candidate generation”, In Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD’00), Dallas, TX, **(2000)** May, pp. 1–12.
[4] L. Chen, S. S. Bhowmick and L. T. Chia, “FRACTURE-Mining: Mining Frequently and Concurrently Mutating Structures from Historical XML Documents”, Elsevier Science Journal: Data & Knowledge Engineering, vol. 59, Issue 2, **(2006)**, pp. 320-347
[5] L. Chen, S. S. Bhowmick and L. T. Chia, "Mining Maximal Frequently Changing Subtree Patterns from XML Documents", In Proceedings of the 6th International Conference on Data Warehousing and Knowledge Discovery(DaWaK), Zaragoza, Spain, **(2004)**, pp. 68-76.
[6] L. H. Rusu, W. Rahayu and D. Taniar, "Mining Changes from Versions of Dynamic XML Documents", KDXD 2006, LNCS3915, **(2006)**, pp. 3-12.
[7] M. P. Papazoglou and P. M. A. Ribbers, “e-Business: Organizational and Technical Foundations”, John Wiley & Sons, Forthcoming, **(2005)**.
[8] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules”, In Proc. 1994 Int. Conf. Very Large Data Bases (VLDB’94), Santiago, Chile, **(1994)** September, pp. 487–499.
[9] R. Krishnamurthy, V. T. Chakaravarthy, R. Kaushik and J. F. Naughton, “Recursive XML schemas, recursive XML queries, and relational storage: XML-to-SQL query translation”, in: Proc. of the ICDE Conference, **(2004)**, pp. 42–53.
[10] S. Y. Chien, V. J. Tsotras and C. Zaniolo, “Efficient schemes for managing multiversion XML documents”, VLDB J., vol. 11, no. 4, **(2002)** December, pp. 332–353.
[11] Extensible Markup Language (XML) http://www.w3.org/xml/
[12] International Organization for Standardization. Available from http://www.iso.org/iso/home.htm
[13] Megginson Technologies: Simple API for XML. Available from http://www.megginson.com/downloads/SAX/
[14] OpenOffice.org 2.0 in Enterprises. English version. Available from http://www.ba.ncu.edu.tw/dmerplab/CeBIT_OOo_En.odp
[15] Unicode in XML and other Markup Languages, “Unicode Technical Report#20”. Available from http://unicode.org/reports/tr20/tr20-6.html
[16] World Wide Web Consortium. Available from http://www.w3.org/
[17] W3C’s Document Object Model (DOM). Available from http://www.w3.org/DOM
[18] W3C’s Extensible Markup Language (XML) 1.0 (Fourth Edition). Available from http://www.w3.org/TR/REC-xml/

Authors

Ming-Shien Cheng received the MS degree from the Management Sciences Department of Tamkang University, Taiwan in 1988. He is a lecturer at the Industrial Engineering and Management Department at the Ming Chi University of Technology and a doctoral student in the Business Administration Department at National Central University in Jhongli, Taiwan. Her research interests include data mining, database management and ERP applications in business domains. His papers have been published in LNCS.

Ping-Yu Hsu is a Professor and the department head of Business Administration at the National Central University in Jhongli, Taiwan. Ping-Yu received his PhD in Computer Science from UCLA, USA. He also works as the secretary-in-chief of the Chinese ERP association. His research interest focuses on business data applications, including data modeling, data warehousing, data mining, and ERP applications in business domains. His papers have been published in IEEE Trans on Software Engineering, Information Systems, Information Sciences, Computers and Operations research and various other journals.

Min-Tzu Wang received the MS degree from the Computer & Information Sciences Department, University of Oregon, USA. She is a lecturer at the Information Management department at the Technology and Science Institute of Northern Taiwan and a doctoral student in the Business Administration department at National Central University in Jhongli, Taiwan. Her research interests include data mining, information retrieval, E-commerce, M-commerce, and ERP applications in business domains. Her papers have been published in LNCS and various other journals.