A Temporal Database Compression with Differential Method

Sushil Kumar
Dept. of C. S. E.
NIRT, Sajjan Singh Nagar, Bhopal (M.P.), India

Sarita S. Bhadauria
Dept. of E. C. E., MITS, Gwalior (M.P.), India

Roopam Gupta
Dept. of I. T., UIT, RGPV, Bhopal (M.P.), India

ABSTRACT
Now days it is very tedious job to keep files for personal as well as commercial computing. There are various type of compressing technique used, but one step ahead from them available technique is described here. Almost every application the backend used is database. That why my technique is dedicated to this type of databases. In this proposed technique we consider every type of compression, but when comes to date and time based database, not much compression technique deals with it. For text type compression there are many techniques same for images. But here we are proposing mainly for time and date type data bases. The practical use for this compression may useful for LIC policies, Stock Exchange, Railways Reservations databases etc. It may also useful for Employees working in a firm, maintaining daily database for salary purposes like Time in and Time out. For this type of database the proposed technique will give a big amount of compressions than any other type of techniques. We have techniques regarding database compression are character, memo, number, date, time compression which can work for individual fields in a database. In this paper main concentration has been given for time compressions. We suggested one example in tabular form on that our differential and time method has been applied.

Keywords
Compression, Compression Ratio, Compression factor, Fixed Length Coding (FLC), Huffman after using Fixed Length Code (HFLC), LZW(Lampel Ziv Welch), Lossy Compression, Nonlossy Compression, RLE (Run Length Encoding), Saving Percentage, Temporal Database.

I. INTRODUCTION
As Compression is known as, it is art of presenting the information in a compact form rather than of uncompressed form. There are various compressions proposed but for particularly date and time very rare one. In this type of compression a substantial amount of memory can be saved. Our methodology will compress even more than earlier available compression types. What we are proposing it can be implemented on any type of software. It does not require any special type of software. What our methodology is, it can use daily today life, as we said in abstract it very much used in different types of databases. We have taken one example of employee time in and time out, which is all most every organizations are using in today computer era. Because it is very easy to keep track of all employee working under one organization.

Many compression techniques available are Run Length Encoding, Huffman Encoding, Lempel Zev Welch, Different Dictionary Based Compression, Delta and Differential Compression, Fixed Length Code, Burrow Wheeler Transform, Move to Front Technique, Word based

Text Compression, Pattern Matching in Compression Form etc. some of them are described here [19].

1.1 RLE
If a data item d occurs n consecutive times in the input stream, replace the n occurrences with the single pair nd. The n consecutive occurrences of a data item are called a run length of n, and this approach to data compression is called run length encoding or RLE [18, 20].

1.2 Huffman Encoding
It uses the probability distribution of the alphabets of the source to develop the code word for symbols. According to the probabilities the code word has been assigned. Shorter word code words for higher probabilities and longer code words for smaller probabilities are assigned. For this type of work binary tree is created using the symbols as leaves according to their probabilities and path of those are used as code words. There are two types of Huffman Family are used: Static Huffman and Adaptive Huffman. In static coding first frequencies and then it generates a tree for both compression and decompression process. But adaptive method develops the tree while calculating the frequency and there will be two trees in both the process [22, 23].

1.3 Shannon Fano Coding
This is similar to Static Huffman Coding only difference is the creation of code word. All other process is similar one [20].

1.4 LZW
LZW is a general compression algorithm capable of working on almost any type of data. It is generally fast in both compressing and decompressing data and does not require the use of floating-point operations. Also LZW writes compressed data as bytes and not as words [20].

LZW is referred as a substitution or dictionary-based encoding algorithm. The algorithm builds a data dictionary (also called a translation table or string table) of data occurring in an uncompressed data stream. Patterns of data (substrings) are identified in the data stream and are matched to entries in the dictionary. If the substring is not present in the dictionary, a code phrase is created based on the data content of the substring, and it is stored in the dictionary. The phrase is then written to the compressed output stream.

Data Compression technique on text Files: A comparison study has been done by Haroon Altarawneh et. al., he has taken different methods of data compression English text files, LZW, Huffman,

Fixed Length Coding (FLC) and Huffman after using Fixed Length Code (HFLC). He evaluated a test on these algorithms on different text files or different file sizes and taken a comparison in terms of comparison: Size, Ratio, Time (Speed) and entropy. And they found that LZW is the best algorithm in all the compression scales.

According to them LZW is a general compression algorithm capable of working on almost any type of data. It creates a table of strings commonly accruing in the data being compressed, and replaces original data with reference into the table. LZW Compression replaces strings of characters into a single code. Compression occurs when a single code is output instead of a string of characters. It starts with a dictionary of all the single character with indexes. It starts expanding the dictionary as information gets send through. Pretty soon, redundant strings will be coded as a single bit, and compression has occurred [23].

II. MEASURING PERFORMANCES OF COMPRESSION

There are various criteria of measuring the performance of the compression depends on the nature of the application used. Mainly it is used for space the time efficiency is other factor. As we all know that the compression behavior depends on the redundancy of symbols in the source files. It is very difficult to measure performance of compression in general. The performance of the algorithms depends on structure of the input source and also category of the compression algorithms i.e. lossy or lossless. For example lossy and lossles examples are given in my previous paper [15]. The measuring of general performances is difficult and there should be different measurements to evaluate the performances of different compression techniques.

2.1 Compression Ratio

It is the ratio between the size of the compressed file and the size of the source file [7].

$$\text{Compression Ratio} = \frac{\textit{Size after compression}}{\textit{Size before compression}}$$

2.2 Compression Factor

It is just reverse of the compression ratio. That is ratio between the size of the source file and the size of the compressed file.

$$\text{Compression Factor} = \frac{\textit{Size before compression}}{\textit{Size after compression}}$$

2.3 Saving Percentage

It calculates the shrinkage of the source file as a percentage.

$$\text{Saving Percentage} = \frac{\textit{Size before compression} - \textit{size after compression}}{\textit{Size before compression}} \%$$

All the above methods evaluate the effectiveness of compression using file sizes. There are some other methods to evaluate the performances of compression algorithms like Compression time, Computational Complexity and Probability Distribution are also used to measure the effectiveness [20].

DATA Compression is the science and art of representing information in a compact form.

The data may also classified as text, audio, image and video while the real digital data format consists of 0's and 1's in a binary format

- Text data are usually represented by 8-bit extended ASCII code or EBCDIC having extension .txt, .tex, .doc.
- Binary data include data base file spreadsheet data, excitable files and program codes having extension as .bin.
- Image data are represented often by a two dimensional array of pixels in which each pixel is associated with its color code having extension as .bmp, and .psd.
- Graphics data are in the form of vectors or mathematical equations, for example data format is .png (portable network graphics).
- Sound data are represented by a wave function having extension as wav [7].

III. PROPOSED NEW METHODOLOGY

- Characters can be coded into 5-bits coding.
- Memo can be coded into 6-bits coding.
- Dates can be coded into 16-bits coding.
- Time can also be coded into 16-bits coding [15].

IV. IMPLIMENTATION OF PROPOSED METHODOLOGY

We have proposed some new techniques that can be implemented as given below:

4.1 Character

In general 8-bit ASCII code have been used for representing character, but when one declare any attribute to be of character type they often interested only in alphabet character from A-Z or a-z. But with character some more information is needed while dealing with character like space, end of line, comma, full stop, single codes, and nothing. These all will counts only 32 in number and that can be coded in 5-bit coding only [7].

4.2 Memo

The memo field often includes character other than alphabets like number underscore plus minus etc. and it is found that at most 64 symbols are used in general so in this situation 6 bits are sufficient to represent them. Out of these 32 are the same as that in the previous case and remaining 32 are used for numbers 0 to 9 and special characters like “”, !, @, #, $, ^, &, *, (,), -, _, =, +, <, >, ?, /, :, ;, |, \. So in this way 64 symbols can be accommodated in 6-bit coding only [7].

4.3 Number

An attribute having this type is often used to represent certain quantity or amount or extent that anything may have. It can be either the integer or float depending upon the nature or accuracy of the quantity.

By far the most compact and exact representation of this type of data is their own binary equivalent. But it has found that of the following method, which can take advantage of nature and format of the quantity.

Differential Method, Delta Method, BCD Code [15].

4.3.1 Differential Method

In this approach when one wants to compress this type of data we can either look out for the smallest or largest value for this attribute, and store this value together with the table structure. Now the original value could be represented as the difference between the original value and this value, which in fact will be less than the original quantity, if the values are distributed in a linear fashion. For example if it has the following values like755, 762,792,720,725,789 then if one choose 720 as the base value and store it together with the attribute definition in the table structure, then the original data will be stored as 35,42,72,0,5,69 much less than the original one. Now during the query processing these values could be used directly without any additional processing if the query is being modified to process them.

4.4 Date

Date field is often associated with temporal database, and most often used inside the data ware house, that contains historical information about any aspect of life. Most database store the date as 8-byte entry (2 for day, 2 for month & 4 for year). Here we have proposed only 2-byte format that works with traditional DOS and WINDOW environment [7].

4.5 Time

The usual ways of storing a time stamp of any event require 6-byte and implemented in that manner in most of the database systems. But here again we could save substantial space by representing a time stamp in a manner that require only 2-byte for its storage that used in DOS and WINDOW. The Bit-wise distribution of Hour, Minutes & Second is shown below.

H H H H H M M M M M M S S S S S

The time can be converted into a 2-byte value using the following formula.

Time = 2048*HOUR+32* MINUTE + SEC/2.

This scheme is quite common in traditional MS-DOS file system where second value is measured in two-second interval. If we want to be more precise we have to add one more bit to accommodate second entry because 6 bits are required to represent 60-second domain. We will take the first mentioned scheme to represent time of any event.

Let we have a time value of 16:40:24 in HH: MM: SS format, applying this to formula.

We get time = 34060 and its binary equivalent is:

1 0 0 0 0 1 0 1 0 0 0 0 1 1 0 0

Hour Minutes Second

So in this way we can represent this time stamp. Now to get each of these separately we have to perform the bit wise shift operation in the following way. In this field 1952 could be used for temporal variable and 1984 for NULL value.

Hour: Right shift the entry by 11 Bit position.

Minutes: Left shift by 5-Bit position followed by right shift of 10 times.

Second: Left shift by 11-bit position followed by right shift of 11 times and multiply it by 2.

There is one problem it counts in 2 seconds intervals, other than that it works efficiently like other environments.

V. RESULTS AND EVALUATION OF PROPOSED METHODOLOGY

We have taken some 25 employee records on one day attendance from an organization. Now apply on that the different compressions which are proposed in this paper above. In below table we have applied Differential Compression for numbers and Time Compression on both Time fields and also calculated the number of bits before compression and after compression. Below the table we have calculated the total number of bits before compression and after compression. One can see that original size i.e. ASCII size in bits is 3000. But the same file size has been compressed to 1064 bits. The same comparison is also can be seen graphically given below. In modern database systems table structure is also stored together with the database file so that any application can make use of it. When we consider this compression scheme we will store this structure without any modification, it is only the data that will be stored according to this new scheme. The file also store some additional words like field separator, end of record to mark and distinguish separate attribute and record, and these will be there in proportion to the number of records in the file.

From table1 the compression ratio, compression factor and saving percentage can be found as below:

Compression Ratio = 1064/3000 = 0.035467.

Compression Factor = 3000/1064 = 28.19548.

Saving Percentage = (3000 – 1064) / 3000 = 0.6453 = 64.53 %.

It means there is saving in memory is 64.53. Through graph shown below is also giving the clear picture of saving a drastic difference between original size and compressed size. The blue one is before compression i.e. full coverage of graph, but the red one is only compressed one. So after compression only this much area of storage is required and it is very less, only 100 – 64.53 = 35.47 % of original size.

Table1. List of employees for one day attendance

Employee ID	Time in	Time out	Original size (Bits)	Compressed size (Bits)
136	08:30:00	16:30:55	120	44
101	08:35:09	16:20:34	120	38
134	08:37:07	16:10:02	120	44
108	08:39:58	16:10:00	120	38
112	08:45:09	15:55:56	120	44
109	08:45:03	16:40:09	120	38
138	08:46:00	16:30:34	120	44
135	08:47:34	16:50:55	120	44
100	08:47:39	16:00:02	120	38
111	08:48:00	16:01:59	120	44
137	08:48:34	16:02:02	120	44
103	08:48:35	16:10:00	120	38
195	08:48:36	16:34:00	120	44
128	08:48:37	16:01:11	120	44
104	08:49:45	16:55:09	120	38
124	08:50:02	16:20:00	120	44
142	08:51:00	16:21:00	120	44
178	08:52:54	16:04:04	120	44
129	08:52:55	16:04:05	120	44
159	08:53:09	16:06:59	120	44
192	08:53:10	16:12:12	120	44
184	08:53:11	16:11:16	120	44
189	08:53:57	16:00:58	120	44
156	08:54:49	16:23:12	120	44
174	08:55:46	15:56:57	120	44
		TOTAL	**3000**	**1064**

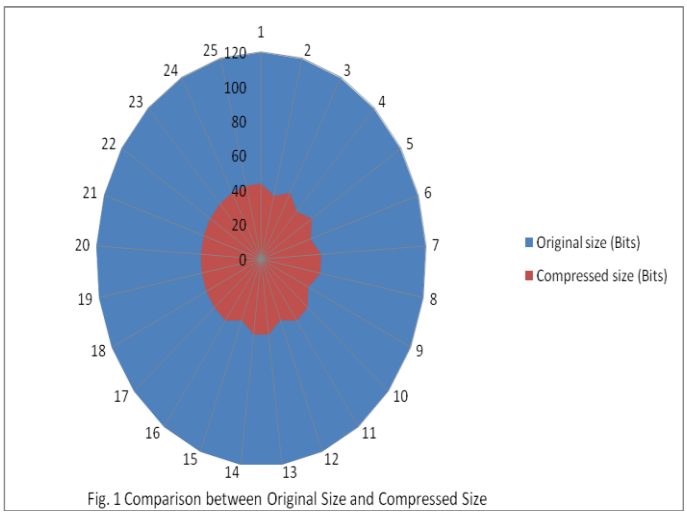

Fig. 1 Comparison between Original Size and Compressed Size

VI. CONCLUDING REMARK

In our paper we have taken original size of Employee record. On that our proposed methodology has been applied. We have presented a very new compression technique for temporal database. Our technique will give better results, so that the proposed technique has better performance. The CPU utilization will also increases due to compression. Also compression is beneficiate for Input and Output performance. The graph above fig.1 will give the clear picture of compression. The red colour area is only required after compression rest can be saved. In comparison to blue area red is much smaller, it shows better compression.

VII. REFERENCES

[1] A.S. Tanenbaum "Computer Network" (Fourth Edition Prentice-Hall of India Limited).

[2] Cormack, G. V. 1985. "Data Compression on a Database System". Commun. ACM 28 12, (Dec.), 1336-1342.

[3] Debra A. Ielwer and Daniel S. Hirschberg "Data Compression" –IEEE JUNE 2002.

[4] Navathe S.B, Elmasn R. "Fundamentals of Database System" (Pearson Education).

[5] Pujari. A. K "Data Mining Technique" (University Press).

[6] Reghbati, H.K "An Overview of Data Compression Technique" IEEE computer (1981).

[7] Saloman D. "Data Compression the Complete Reference" Springer, 3rd Edition (2004).

[8] William Stallings, "Network Security Essentials Application and Standard" (Pearson Education)

[9] Holger Kruse, Amar Mukherjee, "Data Compression Using Text Encryption" FL 32816 Page No. 1068-0314/97 Years 1997 IEEE Department of Computer Science University of Central Florida Orlando, 32816.

[10] En-Hui Yang And John C. Kieffer, "On The Performance Of Data Compression Algorithms Based Upon String Matching" Fellow Ieee, Ieee Transactions On Information Theory, Vol, 44, No. 1, January 1998 0018-9448 1998 Ieee.

[11] Ming-Bo Lin, Member and Yung-Yi Chang, "A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm" IEEE TRANSACTIONS ON VERY LARGE SCALEINTEGRATION (VLSI) SYSTEMS, VOL, 17, NO, 9, SEPTEMBER 2009 1063-8210 Years 2009 IEEE.

[12] 'N. Magotra', W. McCoy', S. Stearns' Dept. of EECE, "A Lossless Data Compression In Real Time F. Livingston." University of New Mexico, Albuquerque, NM 87131: Dept, 9311, Sandia National Laboratory, Albuquerque, NM 87185 1058-6393/95 year 1995 IEEE.

[13] Thanos Makatos, Yannis Klonatos, Manolis Marazakis, Michail D. Flouris, and Angelos Bilas, "ZBD: Using Transparent Compression at the Block Level to Increase Storage Space Efficiency", Foundation for Research and Technology – Hellas (FORTH), P.O. Box 2208, Heraklion, GR 71409, Greece, 978-07695-2/10, © 2010 IEEE.

[14] Ming-Bo Lin, Member, IEEE, and Yung-Yi Chang, "A New Architecture of a Two-Stage Lossless Data Compression and Decompression Algorithm", 1063-8210, ©2009 IEEE.

[15] Sushil Kumar, Dr. Sarita S. Bhadauria, Dr. Roopam Gupta, "A Digital Compression Scheme Using Delta and Differential Methods", IJCA (0975-8887) Volume 25 – No.7, July 2011, page No. 18 – 25.

[16] Senthil Shanmugasundaram, Robert Lourdusamy, "IIDBE: A Lossless Text Transform for Better Compression" International Journal of Wisdom Based Computing, Vol. 1 (2), August 2011, Page No. 1 – 6.

[17] Tanakorn Wichaiwong, Kitti Koonsanit, Chuleerat Jaruskulchai, "A Simple Approach to Optimized Text Compression's Performance" 4th International Conference on Web Services Practices, IEEE Computer Society, 978-0-7695-3455-8/08, Page no. 66 – 70.

[18] M. Baritha Begum, Dr. Y. Venkataramani, "An Efficient Text Compression for Massive Volume of Data" IJCA (0975 - 8887), Volume 21 – No. 5, May 2011, page No. 5 – 9.

[19] Md. Nasim Akhtar, Md. Mamunur Rashid, Md. Shafiqul Islam, Mohammod Abul Kashem, Cyrll Y. Kolybanov, "Position Index Preserving Compression for Text Data" JCS&T Vol. !! No. 1, April 2011, Page No. 9 – 14.

[20] S. R. Kodituwakku, U.S. Amarasinghe, "Comparison of Lossless Data Compression Algorithms for Text Data" IJCSE Vol. 1 No. 4 416-425, ISSN : 0976-5166, Page No. 416-4125.

[21] Rexline S. J, Robert L, "Dictionary Based Preprocessing Methods in Text Compression – A Survey" IJWBC, Vol. 1 (2), August 2011, Page No. 13-18.

[22] Umesh S. Bhadade, Prof. A. I. Trivedi, "Lossless Text Compression using Dictionaries" IJCA (0975 - 8887) Volume 13- No. 8, January 2011, Page No. 27-34.

[23] Haroon Altarawneh, Mohammad Altarawneh, "Data Compression Techniques on Text Files: A Comparison Study", IJCA (0975 - 8887) Volume 26- No.5, july2011.