International Journal of Computer Applications (0975 – 8887)
Volume 180 – No.4, December 2017

An Efficient Text Database Compression Technique using 6 Bit Character Encoding by Table Look Up

Md. Ashiq Mahmood
Department of Computer Science & Engineering
Khulna University of Engineering & technology

Tarique Latif
Department of Computer Science & Engineering
Khulna University of Engineering & Technology

Md. Riadul Islam
Department of Computer Science & Engineering
North Western University, Khulna

ABSTRACT
Character encoding determines a term which represents a repertoire of characters by some kind of encoding technique. It covers a huge area of applications such as data communication, storage of data, textual data transmission and database technology. In this paper, a new technique of compression for text data is proposed which encodes a character by 6 bits namely 6 - Bit Encoding (6BE). Actually the working method of this technique is encoding an 8 bit character by 6 bits. This technique works with the characters which are printable. For encoding a character to 6 bit, it uses a lookup table. Firstly, it divides the characters into 4 sets and then it uses the location of characters uniquely to encode by 6 bits. By this procedure 8 bit characters are converted into 6 bits by this 6BE technique. At First, this technique on simple text. It is found that, the 6BE technique can able to compress the original text by 25%. After that this 6BE technique is used in proper database technology by compressing the text data in a table of a database. The 6BE is able to compress as well as decompress the original data with the help a lookup table. The reverse technique is also detailed for decompression to get back the original table. The outcome of 6BE technique is also applied to compress again by the known algorithm Huffman and LZW. The experimental result shows promising performance. The technique is further discussed by some examples and descriptions.

General Terms
Database Technology, Data Compression, Algorithm etc.

Keywords
Encoding, Compression, Decompression, 6-bit encoding, Compression ratio.

1. INTRODUCTION
In Data science applications, database compression actually defines the compaction of data in shorter memory space including the actual representation of main database data remaining unchanged. Data compression is a quite promising issue because of requirement of storage and different network's bandwidth. In recent years, data compression becomes a vastly used concept in computer science. Reducing bandwidth and storage requirement, encoding fewer no. of bits, less time requirement for transmission, effective utilization of channel, can be achieved by the compression of data or obvious benefits of data compression. Encoding a character by less bits is very vital for having an effective compression scheme [1]. By using fewer bits, the original representation is compressed in a less storage space [1][2]. Because of the vast importance of data compression, it is essential to found a befitting encoding technique for the presentation of a source of information as accurate as possible using the less number of bits putting the meaning of data being unharmed and unchanged [1][2][3]. In this paper a compression scheme is presented by encoding a character by 6 bits instead of usual 8 bits. It is named 6BE (6 Bit Encoding). The restoration of the information to its original format for the compressed form is quite necessary for most of the cases. For the purpose of doing this, a decoding algorithm is developed, and the performance of this technique is quite promising relevant to that operation. Actually, there are two categories of data compression algorithms are available which includes lossless and lossy data compression. Lossy compression is a kind of data compression system where file size is reduced by discarding some unimportant data that won't compression are the main examples of lossy data compression on the contrary, Lossless compression transforms each bit of data to reduce the size without losing any data after decoding. Actually the importance of lossless data compression is it doesn't allow any data loss. Even if a single bit of data is lost after decoding, that determines the file is corrupted [3][4].The most attractive part of data compression is compressing data in a database system. The performance is immensely improved because smaller size of physical data are needed to be moved for any operation on the database [1][2]. By Text compression, it can be understood that it is the technique of transforming an original symbol of the source data to a small symbol which must assure that the same information should contain as the original data and shorter representation[5][6][7]. A lossless technique named 6BE algorithm was proposed. The algorithm both encoding and decoding techniques. 8 bit characters are converted to 6 bits by this 6BE technique which divides the characters into 4 sets and using them in a single table. The characters are put in the 6BE table according to their frequency of use in English language. Then it use the location of characters uniquely to encode by 6 bits. By putting together the same set code characters, it creates smaller bit code sequence. It can successfully compress the printable characters and promising compression ratio is achieved by this logical compression [9].

2. RELATED WORKS
A lossless algorithm which uses entropy [12] is the known Huffman algorithm. Here the code is found from the binary trees. It is called variable length encoding as it gives variable length encoding. But 6BE is fixed to a size of 6 bits. Here a fewer number of bits uses more frequent words which is its main principle. JPEG files uses Huffman coding. A wavelet-based compression standard [12][13] for image is JPEG 2000.Huffman Encoding has two families which are: Adaptive Huffman Algorithms & Static Huffman Algorithms given in the literature [13][14]. In Adaptive Huffman algorithms the tress are created by calculating the frequencies.

And in Static Huffman Algorithms, firstly the frequencies are calculated & a common tree for both compression & decompression is created [13][14]. So, there will be two trees [13]. Instead of statistical models Dictionary based compression algorithms uses dictionaries [16]. A dictionary lookup-based algorithm is the Lempel-Zev Welch algorithm [16][17] which creates a string αK by using string α and with a K from dictionaries. In the dictionary the string α is also pushed. While ending the character string is also again replaced. The dictionary is not static as it is built from the dynamic data. While decoding raw dictionary it is recovered from the decoded data. At the time of compression and decompression, the dictionary can be accurately built, and is discarded after compressing or decompressing process. As LZW Algorithm [16] is simple and as it has efficient compression ratios it is very famous. The 6BE that is proposed uses a static table look up and even if new data items arrives need not change. The table i.e. the dictionary is fixed and so it can be used when we need to do compression and decompression.

3. PROPOSED METHOD

Actually we proposed 2 methods by using this 6BE technique.

- Efficient Text Compression with 6 Bit Character Encoding(T_6BE)
- Efficient Text Database Compression with 6 Bit Character Encoding(TD_6BE)

For the purpose of encoding a character in memory, 8 bit memory is needed. The proposed idea is actually a scheme of 6 bit character encoding namely 6BE which is represented by a character 6 bits instead of 8 bits. This algorithm mainly works with the general printable characters that has in normal keyboards so the scope is limited to printable characters only. A table is built to look up for representing the characters by 6 bit where it is found that there can be 2^6=64 combinations possible. From the 64 combinations 28 combinations were used and rest of the combinations has been avoided because the following sequences were produced by those combinations

Four consecutive 1 bits (1111)
Or
Four consecutive 0 bits (0000)

Correspond to the nonprintable characters such as BEEP, DEL and BACK SPACE and so on. But, at the time of performing the 5 bit character encoding scheme, it was found that 2^5 =32 combinations are available. But after avoiding above these two bit sequences usable combinations are less than 20. So 6BE is performed because it gives more usable bit sequences.

The target character sets are divided into 4 sets namely Set -1, Set-2, Set-3 and Set-4. The look up table for 6BE Table is showed in Table1. Then the 6BE table is optimized. In the Optimized table,

- Capital Letters are put in Set-1
- Small letters in Set-2
- Digits are in Set-3
- Other printable characters in Set-3 & Set-4

Which are as shown in table 1. Since the scope is limited to afford only 28 characters for each Set hence the characters need to be redistributed. According to their frequency of use, the characters are distributed to sets [21]. According to the use of frequency, the less frequency characters such as Q, q, X, x, Z, z are placed in Set-4. Each of the characters of the table are represented by 6 bits as shown in Table 1

Table 1: Look up table for 6BE

Serial no.	Decimal value	Binary value	Set-1	Set-2	Set-3	Set-4
1	05	000101	A	a	!	<
2	09	001001	B	b	“	=
3	11	001011	C	c	#	>
4	17	010001	D	d	$	?
5	18	010010	E	e	%	@
6	19	010011	F	f	&	[
7	21	010101	G	g	‘	\
8	22	010110	H	h	(	]
9	23	010111	I	i	)	^
10	25	011001	J	j	*	_
11	26	011010	K	k	+	{
12	27	011011	L	l	,	|
13	30	011110	M	m	-	}
14	34	100010	N	n	/	`
15	35	100011	O	o	0	:
16	38	100110	P	p	1	;
17	39	100111	R	r	2	.
18	41	101001	S	s	3	Q
29	43	101011	T	t	4	q
20	46	101110	U	u	5	X
21	47	101111	V	v	6	x
22	49	110001	W	w	7	Z
23	50	110010	Y	y	8	z
24	51	110011		space	9	~
25	54	110110	Set-1			
26	55	110111	Set-2			
27	57	111001	Set-3			
28	59	111011	Set-4			

3.1 Efficient Text Compression with 6 Bit Character Encoding (T_6BE)

3.1.1 Compression Technique

Input: Original Text T
Step1: Representing the text T by T' by adding set change
Step2: Representing the text T' by its corresponding 6 bits value using the 6BE table.
Let the text T' contain K bits.
Step3: If $K\%8 = 0$ then jump to Step5
Step4: If $K\%8! = 0$ then
If Kth bit is 1 then sum up m bits in the form 0101….Bits such that $(K+m)\ \%8 = 0$
If Kth bit is 0 then sum up m bits in the form 1010 …. Bits such that $(K+m)\ \%\ 8 = 0$
Step5: Taking 8 bits from $(K+m)/K$ bits and representing the text T' to compressed text Tc using corresponding ASCII text.

Example 1:

Input Sentence: I am happy
Set Representation: I set2 space am space happy.
Decimal Representation: 23 55 51 05 30 51 22 05 38 38 50
6 bit representation: 010111 110111 110011 000101 …
8-Bit representation: 01011111 01111100 11000101 …….
Compressed String: _|┼{5àÜl¬

3.1.2 Decompression Technique

Input: Compressed text Tc
Step1: Representing the text Tc by its corresponding 8 bits Binary value from the ASCII character.
Tc contain K bits
Step2: Dividing the binary set Tc by 6 for determining 6BE (Corresponding 6 bit representation) as follows
If K % $6 = 0$ then jump to the step3.
If K % $6! = 0$ then,
Remove the last m bits such that $(K - m)$ % $6 = 0$
Step3: Taking 6 bits from the $(K - m)$ bits stream and representing it by the character set of 6BE table.
Step4: excluding the set number to find the original string.

Example 2:
Compressed string***:***
_|┼{5àÜl¬
_ --- 95 --- 01011111
|--- 124 --- 01111100
┼--- 197 --- 11000101 ………….
After dividing it by 6
The 6-Bit Representation: 010111 --- 23 --- I
110111 --- 55 --- set2
110011 --- 51 --- space …………..

Decompressed String:
I set2 space am space happy
Original text: I am happy
The 6-Bit Representation: 010111 --- 23 --- I
110111 --- 55 --- set2
110011 --- 51 --- space …………..
Decompressed String:
I am space happy

3.2 Efficient Text Database Compression with 6 Bit Character Encoding (TD_6BE)

3.2.1 Compression Technique

Input: Normal string S
Step1: By adding set change, representing the string S by S'
Step2: By using the 6BE table, representing the string S' by it"s corresponding 6 bits representation. Let, the representative string S contain K bits.
Step3: If $K\%8 = 0$ then jump into Step5
Step4: If $K\%8! = 0$ then
While Kth bit is 1 then sum up m bits
In the formation of 0101… bits suchlike $(K + m)$ $\%8 = 0$, Or
While Kth bit is 0 then sum up m bits in the formation of 1010… bits suchlike $(K + m)$ $\%8 = 0$
Step5: By taking 8 bits from $(K + m)/K$, the string S' is represented to compress string Sc by applying corresponding ASCII text.

3.2.2 Decompression Technique

Input: Compressed String Sc
Step1: Representing the string Sc by its corresponding 8 bits binary value from the ASCII table. Let, Sc contain K bits.
Step2: For determining 6BE (corresponding 6 bit representation), dividing the binary representation Sc as
While $K\%6 = 0$ then Jump to step 3
While $K\%6! = 0$ then,
Crop the last m bits such like $(K - m)$ % $6 = 0$
Step3: From the $(K - m)$ bits stream, taking 6 bits and representing it by the character set of the table of 6BE.
Step4: After excluding the set number, the original string can be found.

Example 3:

***T*1**

Id	Name	DepId
2015	Ragibillah khan	1
2016	Imdad hosain	2
2017	Parvage rahman	3
2018	Kamrul hasan	4
2019	Abir Hosain	5

***T*2**

DepId	Dep
1	CSE
2	EEE
5	IICT
6	ECE
7	CE

For Left join query,
Select Id, Name, Dep from $T1$ left Join $T2$ on $T1.DepId = T2.DepId$
Then got the input Table $T3$

T3

Id	Name	Dep
2015	Ragibillah khan	CSE
2016	Imdad hosain	EEE
2017	Parvage rahman	null
2018	Kamrul hasan	Null
2019	Abir Hosain	IICT

After Compressing Table $T3$ using Compression Technique

TC

Id	Name	Dep
æxæº	Ÿq|\•ÛlU³iabª	.”ª
æxæ½	_w‘Ö□‘WŠ	I$ª
æxæÅ	›qg¼_ÎqVxXª	Null
æxæÊ	kq^□æóXZEŠ	Null
æxæÍ	rWŸ=–Þ:E^*	]rëU

Selection, projection & join operations are applied on original table $(T1, T2)$.After performing the above 3 operations, table $T3$ is derived**..** Then the 6BE compression is applied on table $T3$ and the compressed table TC is derived.

4. ANALYTICAL ANALYSIS

Using 6BE technique, each of the 8 bits is represented by 6 bits and saved 2 bits. So 6BE best efficiency is $(8 - 6)/8 = 25\%$. In this section, A theory is developed to determine the efficiency in a more precise way. Table 2 shows the parameters considered for analytical evaluation. Let Input text T has length L. After adding set code T becomes T' and L become L' where $L' = L + l$

Table 2: Parameters for analytical evaluation

Parameter	Description
T	Input text
T'	Input text including set code
l	Length of the set code
L	Length of T
L'	Length of T'; $L' = L + l$
B	Total bytes for T
B'	Total bytes for T′
m	Length of extra bits need to be added
T_c	Compressed text
B_c	Total bytes for Tc including m
$B_{c'}$	Length of T_c
η	Efficiency(B_c/B)

Let Input text T has length L. After adding set code T becomes $l' = L + lT'$ and L become L' where $L' = L + l$
Total bytes for $T, B = L \times 8$
Total bytes for $T', B' = L' \times 6 = (L + l) \times 6$
Let m bits $m\,(m >= 0)$ bits needs to be added for the compressed bit.
Total bytes for T_c $is\ B_c\ where\ B_c = B' + mB_c = B' + m$
Length of T_c $is\ B_{c'} = B_c / 8 = (B' + m)/8$
Where, $1 \leq m \leq 7$
Compression ratio,
$\eta = Compressed\ size/\ Uncompressed\ size = B_{c'} / B$
$(BB_c/8) / (L * 8)$
$B_c / (L * 8)$
$(B' + m) / (L * 8)$
For large L, m can be ignored,
So then the efficiency would be $= B' / (L * 8)$
$((L' + l) * 6) / (L * 8)$
$3(L + l) / 4L$
$(3L/4L) + (3l/4L)$
$(3/4) + (3l/4L)$
For usable compression, η<= 1 then,
$(3/4)3l / 4L + (3l/4L) <= 1$
$(3l/4L) <= (1 - (3/4))$
$(3l/4L) <= 1/4$
$3l <= 4L/4$
$l <= L/3$
$l/L <= 1/3$

Actually it is the ratio between original length and set code length. It shows that the scheme is usable until the set code length is equal or less than to $1/3rd$ of the original length. A positive efficiency is acquired by applying these analysis in this algorithm. But if this ratio is greater than $1/3$ the efficiency would be negative.

5. EXPERIMENTAL RESULT

Two types of experimental result is observed for the 2 methods.

5.1 T-6BE Method

A completely different string is found by using this 6BE technique. The common known algorithm Huffman and LZW is applied to the string which is get from the 6BE technique. Hence the experimental result actually show the comparison between 5 schemes namely, 6BE, Huffman, LZW, $6BE + Huffman$ and $6BE + LZW$. 2 types of data set is taken which are detailed below:

Distinct characters: All the distinct characters (95) in various lengths are applied into these 5 techniques.

Standard Data set: Some standard (general text) data in various lengths are applied in to this 5 techniques.

5.1.1 Compression Ratio

Figure 1 & 2 actually shows the compression ratio for distinct data set & standard data set respectively.

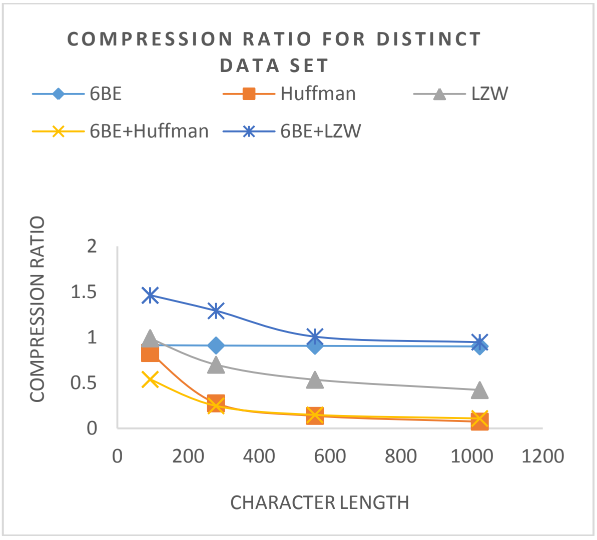

Fig 1: Compression Ratio for distinct data set

Figure 1 shows that, Huffman algorithm provides very good efficiency. The 6BE technique shows an average compression ratio but LZW shows a poor efficiency. But the important fact is that after combining these two techniques with 6BE, Compression Ratio has reduced gradually for $6BE + Huffman$ as well as $6BE + LZW$ goes to more than 1 which is negative but 6BE provides some constant efficiency.

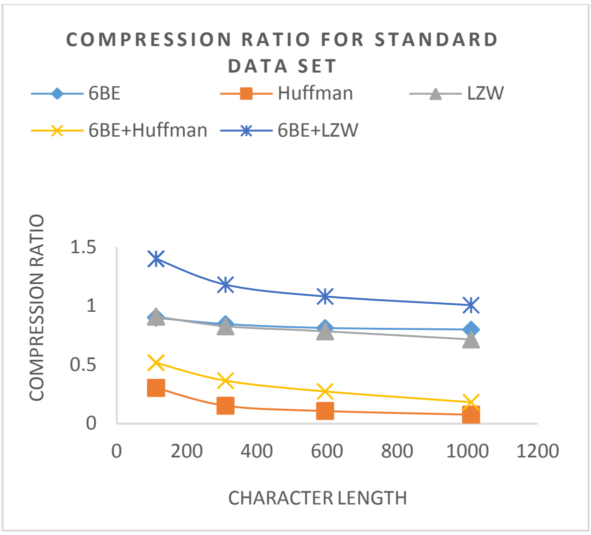

Fig 2: Compression Ratio for Standard Data Set

From figure 2, it is found that Huffman algorithm & LZW shows very promising Compression ratio. 6BE technique shows an average compression ratio. But after combining it with Huffman and 6BE it improves the performance.

5.1.2 Compression time

Figure 3 and 4 provides the compression time for distinct data set and standard data set respectively. From figure 4, the Huffman provides very bad result but 6BE gives better result than Huffman. $6BE + LZW$ shows the best performance. From figure 5, the 6BE and LZW provides promising performance.

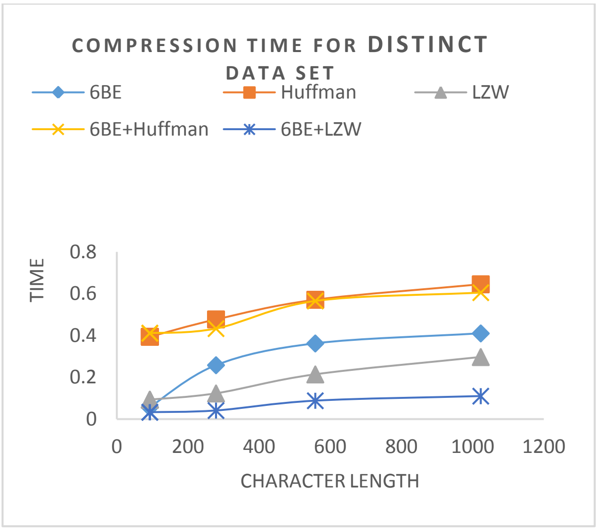

Fig 3: Compression Time for Distinct data set

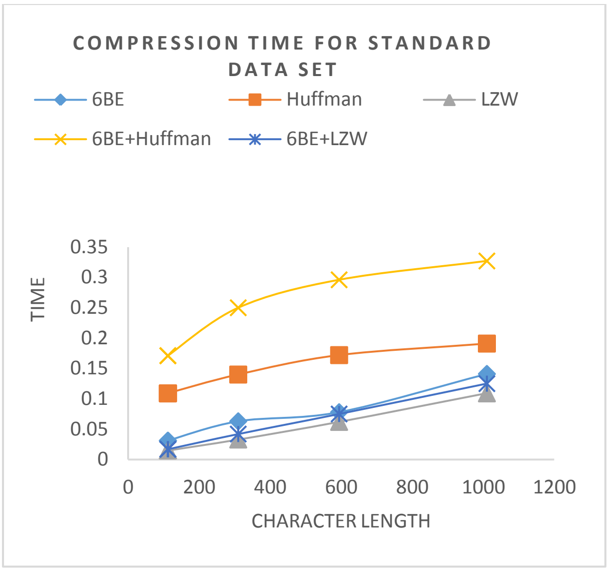

Fig 4: Compression Time for Standard Data Set

5.2 TD_6BE Method

Since 6BE produces some text in printable format. It is further compressed by applying them in Huffman and LZW compression technique. Hence the experimental result shows the comparison between 5 schemes namely, 6BE, Huffman, LZW, $6BE + Huffman$ and $6BE + LZW$. The following important database operation is applied and the generated result is analyzed in this section.

- Selection Operation: The selection operation is applied in various sizes of table data.
  $\sigma C1 = V1\ (T1)$ *Where,* $C1$ *is the column name and* $V1$ *is the value to retrieve.*

- Join Operation: The join operation of database is performed in various sizes of table data.
  $\Delta C1 = V1\ (T1 \bowtie T2)$
- *Selection with Projection (SP) Operation:* The Selection with Projection (SP) operation is performed of database in various sizes of table data.

  $\pi C1, C2 \ldots Ck\ (\pi)$

5.2.1 Compression Ratio:

Figure 5 shows the compression ratio for selection operation. The Huffman algorithm & LZW shows very promising efficiency. The 6BE technique provides an average compression ratio. But the important fact is that if these two techniques are combined with 6BE, it shows excellent efficiency far better than 6BE.This is because Huffman algorithm derives the table from the estimated probability or frequency of occurrence (weight) for each possible value of the source symbol. So more common symbols are generally represented using fewer bits than less common symbols. As well as LZW compression replaces strings of characters with single codes. It adds every new string of characters it performs to a table of strings. Compression occurs when a single code is output instead of a string of characters.

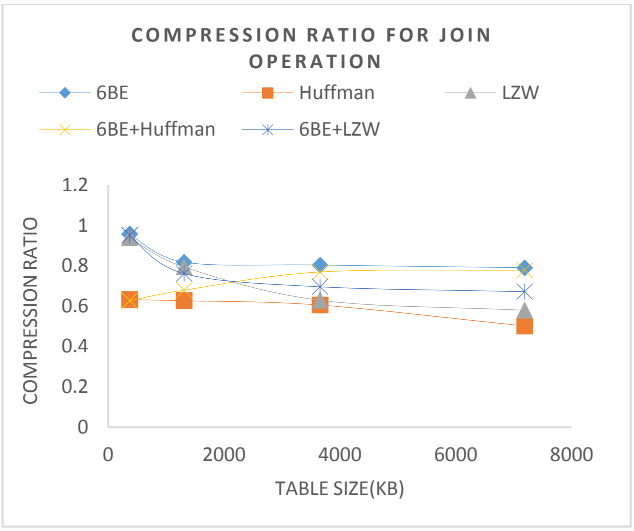

Fig 5: Compression Ratio for Selection Operation

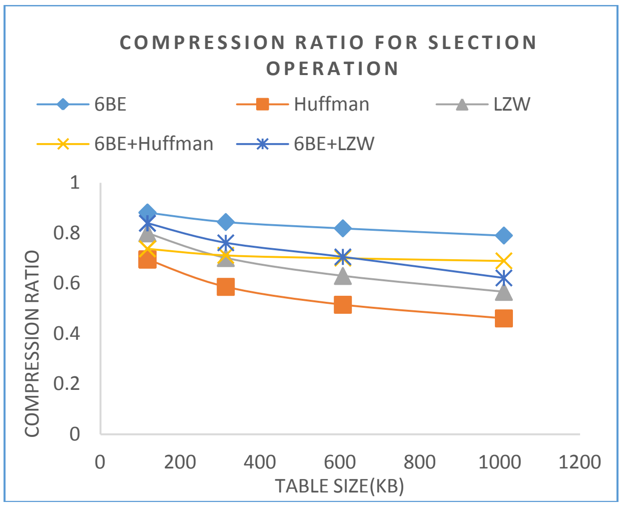

Fig 6: Compression Ratio for Join Operation

Figure 6 shows the compression ratio for join operation. By this graph it is found that Huffman algorithm & LZW provides very good Compression ratio. 6BE technique provides an average compression ratio. But when Huffman and 6BE are combined it improves the performance.

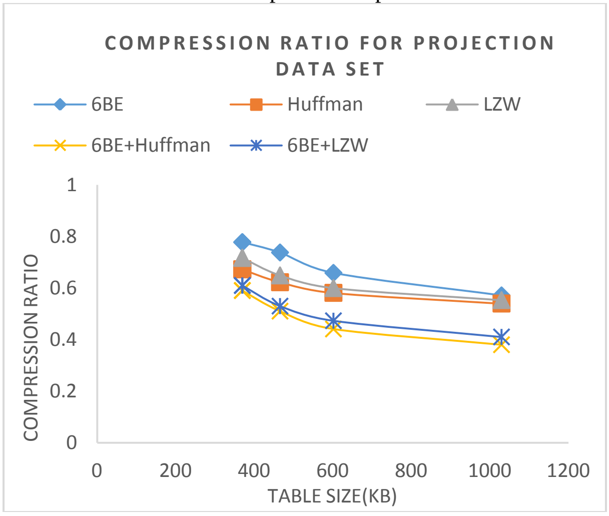

Fig 7: Compression Ratio for Selection with Projection Operation

Figure 7 shows the compression ratio for selection with projection operation. By this graph it is found that 6BE technique provides an average compression ratio. But after combining with Huffman and 6BE the performance improves.

5.2.2 *Compression time:*

Figure 8, 9 and 10 shows the time required for the compression schemes for selection operation and cross join operation respectively. From figure 8, 6BE gives better performance. $6BE + LZW$ gives the best result.

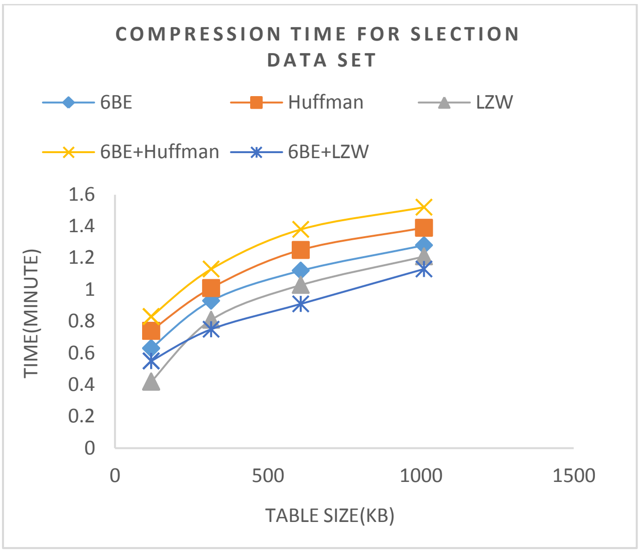

Fig 8: Compression Time for Selection Operation

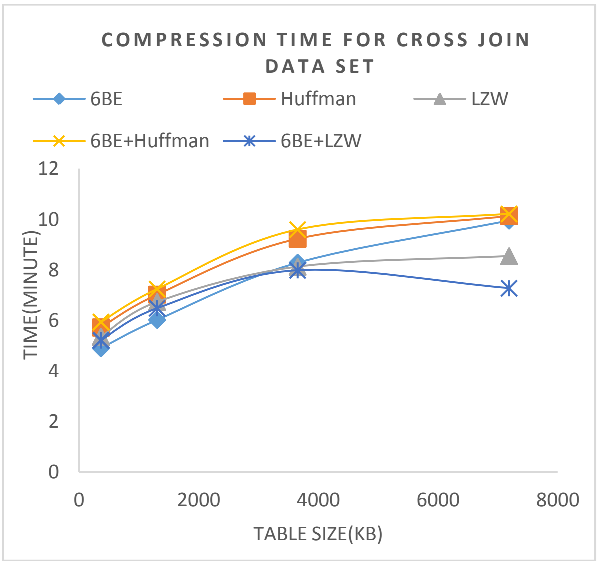

Fig 9: Compression Time for Join Operation

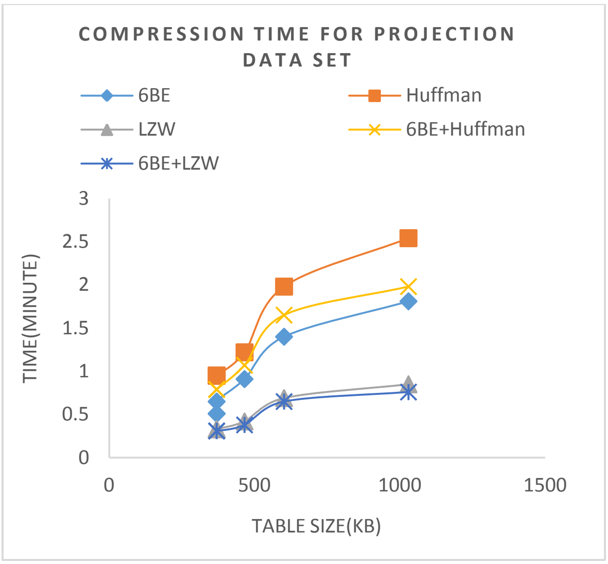

Fig 10: Compression Time for Selection with Projection Operation

5.2.3 *Retrieval performance:*

Query**:**

A range key query for retrieval performance is used,

$SELECT * FROM$

$WHERE\ DepId < V1\ AND\ Id < V2$

This query is applied to a table of size 466 kb and observe the retrieval time of this above 5 algorithms in figure 11.

It is found that the retrieval time for 6BE algorithm is quite promising. Huffman & LZW provides good result. 6BE+Huffman & 6BE+LZW provides excellent result because after compressing by 6BE the string is quite shorter than the original text .So when this compressed text is again compressed by Huffman and LZW the performance improves.

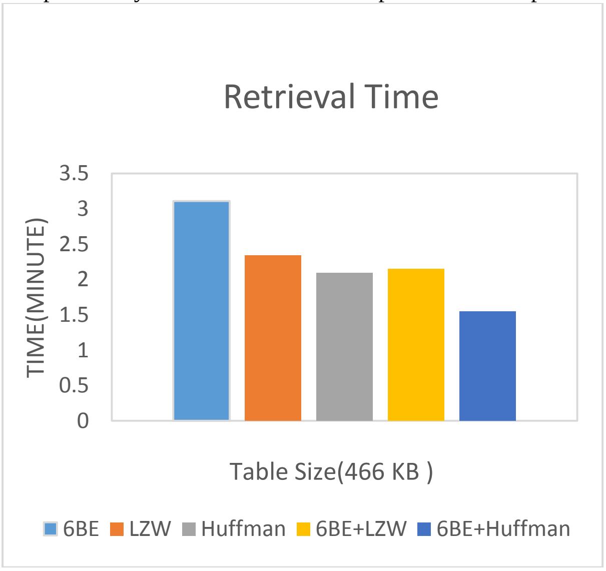

Fig 11: Retrieval Time for Selection with Projection Operation

6. CONCLUSION AND FUTURE WORK

An encoding scheme for all general characters is described here. Very little time is needed for encoding if this encoding scheme is used and when the compression ratio with performance is compared, it gives a great result. Text data of huge amount can be compressed by using this algorithm. Both forward & backward mapping can be done using this compression so it can be called it mapping complete. If the frequent set change is less, it gives a very promising performance. If parallel environment or load balancing or both is applied, better encoding time can be obtained. In future, there is a scope to improve this encoding scheme. This technique can be made more efficient by increasing the efficiency more than 25 % by using 5 bit encoding technique. The set change can also be handled in an efficient way so that it does not affect the efficiency. If this encoding scheme is used as text compression scheme the sequence of 1111 bits and 0000 bits may also be included in the input pattern.

7. REFERENCES

[1] Ashiq Mahmood , Tarique Latif and K. M. Azharul Hasan, “An Efficient 6 bit Encoding Scheme for Printable Characters by table look up”, International Conferenceon Electrical, Computer and Communication Engineering (ECCE), pp. 468-472, 2017.

[2] DwiSuarjaya. “A New Algorithm for Data Compression Optimization,”(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 3(8), 14- 24, 2012.

[3] Senthil Shanmuga sundaram, and Robert Lourdusamy, “A Comparative Study Of Text Compression Algorithms,”International Journal of Wisdom Based Computing, Vol.1 (3), pp. 68-76, 2011

[4] Hussein Al-Bahadili, and Shakir M. Hussain, “A Bit-level Text Compression Scheme Based on the ACW Algorithm,” International Journal of Automation and Computing, 7(1), pp.123-131,2010.

[5] A Carus, and A Mesut, “Fast text compression using multiple static dictionaries,” Information Technology Journal, 1013-1021, 2010.

[6] Alessio Langiu. “On parsing optimality for dictionary-based text compression,” Journal of Discrete Algorithms, 65-70, 2013.

[7] Capo-chichi, E. P., Guyennet, H. and Friedt, J. K-RLE, “a New Data Compression Algorithm for Wireless Sensor Network,” In Proceedings Third International Conference on Sensor Technologies and Applications, pp.502-507, 2009.

[8] Muthukumar Murugesan, T. Ravichandran, “Evaluate Database Compression Performance and Parallel Backup,” International Journal of Database Management Systems (IJDMS) Vol.5(4), 17-25, 2013.

[9] Amit Jain, Kamaljit I. Lakhtaria, “Comparetive Study of Dictionary Based Compression Algorithms on Text Data”, Proceedings of the Data Compression Conference, IEEE Computer Society, 2009.

[10] Ahmad Affandi, Saparudin, and Erwin, “The application of text compression to short message service using huffman table”Vol.6 No.1 Journal Generic, 19-24, 2011.

[11] Asadollah Shahbahrami, Ramin Bahrampour, Mobin

Sabbaghi Rostami, Mostafa Ayoubi Mobarhan, “Evaluation of Huffman and Arithmetic Algorithms for Multimedia Compression Standards,” International Journal of Computer Science, Engineering and Applications (IJCSEA) Volume 1, Number 4, 2011.

[12] Mamta Sharma, “Compression Using Huffman Coding”, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.5, May 2010.

[13] SR Kodituwakku, US Amarasinghe, “Comparison of lossless data compression algorithms for text data,” S.R. Kodituwakkuet. al. / Indian Journal of Computer Science and Engineering ,2012,Vol 1 No 4 416-426.

[14] Si-huiShu, Yi Shu, “A Two-Stage Data Compression Method For Real-time Database”, 3rd International Conference on System Science, Engineering Design and Manufacturing, DOI 10.1109/ICSSEM.2012.6340844, 2012.

[15] R Naqvi, RA Riaz, F Siddiqui, “Optimized RTL design and implementation of LZW algorithm for high bandwidth applications,” Electrical Review, 279-285, 2011.

[16] Zhou Yan-li, Fan Xiao-ping, Liu Shao-qiang, XiongZhe-yuan,“Improved LZW algorithm of lossless data compression for WSN”, 3rd IEEE International Conference on Computer Science and Information Technology, 2010 DOI 10.1109/ICCSIT.2010.5563620, 2010.

[17] SushilaAghav, “ Database compression techniques for performance optimization, 2nd International Conference on Computer Engineering and Technology (ICCET),10.1109/ICCET.2010.5485951, 2010”.

[18] Md. Abul Kalam Azad, Rezwana Sharmeen, Shabbir Ahmad and S. M. Kamruzzaman, “An Efficient Technique for Text Compression” The 1st International Conference onInformation Management and Business, pp. 467-473, 2005.

[19] Pujar, J.H.; Kadlaskar, L.M. "A New Lossless Method of Image Compression and Decompression Using Huffman Coding Techniques", Journal of Theoretical and Applied Information Technology. 15 (1), pp.18–23, 2010.

[20] M Garcia, P Gamallo, “Dependency-based text compression for semantic relation extraction” The 8th International Conference on Recent Advances in Natural Language Processing, 21-28, 2011.