A Digital Compression Scheme using Delta and Differential Methods

Sushil Kumar
Department of Information Technology,
Technocrats Institute of Technology, Anand Nagar,
Bhopal (M.P.),
India

Dr. Sarita S. Bhadauria
Department of Electronics & Communication Engg. MITS,
Gwalior (M.P.),
India

Dr. Roopam Gupta
Department of Information Technology,
UIT, RGPV, Bhopal (M.P.),
India

ABSTRACT
The advancement of information technology has affected all walks of our life. And when we talk the use of information technology in a business environment, we cannot ignore the presence of a huge number of data base systems as its core. Data base technology has also grown from a simple file system to data navigation system, and over a last two to three decades a majority of business institutions, organizations, industries etc. have adopted the computerization process, and as a result have been flooded with data. Temporal database (a database that require some aspect of time when organizing their information) often increases with the time like information from reservation counters (flight, railways, buses, hotels), Bank ATMs, shares price from stock market, insurance policies. So with the limited resources how to manage and store these data, the only possible solution one can have is to just compress and store it with in the available resources. The traditional approach of compression make use of entropy encoding (compress without any regard to its content), whereas we can take advantage of Differential and Delta coding compression as we do in text compression. Now days many papers using loosy compression or lossless compressions which comes under both source encoding and entropy encoding. This paper presents an attempt to apply this category of compression method for a database file with some new approaches [9]. Approaches may be different but final goal is how to compress a data to some efficient manner. The percentage of compression level will become very high with these given approaches; it may go as high as 60% to 70% of compression [18]. The approaches are so simple that can be implemented in even C or C++ also. So that programmer and user can understand so simple way. It does not require special type of software. The attempt is so simple and may be used as a new development of compression for database.

General Terms
Compression, Compression Ratio, Lossy Compression, Non lossy Compression, Data Types.

Keywords
Differential Method, Temporal Database, Delta code,

1. INTRODUCTION
Data compression is the process of converting an input data stream i.e. the source stream or the original raw data into another data stream i.e. output, or the compressed scheme that has a smaller size. A stream is either a file or a buffer space in the memory. Data compression is popular for two reasons.

1) People like to accumulate data and hate to throw anything away. No matter how big a storage device one has, some of later it is going to overflow. Data compression seems useful because it delays this inevitability.
2) People hate to wait a long time for data transfers. When sitting at a computer, waiting for a web page to come in or for a file to download, we naturally feel that anything longer than a few seconds is a long time to wait. The field of data compression is often called "Source Coding". Assign short codes to common events i.e. symbols or phrases.

There are many compression methods some suitable for text and other for graphical data (still image or movies).
There are some Basic Techniques of compressions:

1.1 Intutive Compression
This method was used even before invention of computer today these methods are mostly of historical interest.

1.1.1 Braille
It is also a part of compression in which enable the blind to read, was developed by Louis Braille in the 1980's still is common use today.

1.1.2 Irreversible Text Compression
Sometimes it is acceptable to compares text by simply throwing away some information. This is called irreversible text compression or compaction. The decompressed text may not be identical to the original, so this may be used for some special purpose. A run of consecutive blank spaces may be replaced by a single space. This may be acceptable for literary text or programs. But not for tabular for in extreme cases all text characters except letters and spaces may be thrown away and the letters may be case flattered i.e. converted to all lower or all upper case. This will remain just 27 symbols, so a symbol can be encoded in 5 instead of the usual 8 bits.

The compression ratio= 5/8=0.625

Not bad, but loss may be great.

1.1.3 Ad Hoc Text Compression
In this method some intutive ideas, where the compression must be reversible i.e. lossless.

In this case the text may be removed and their position indicated by a bit string that contains a 0 (Zero) for each text, and 1 (one) for each space.

- In this one of those rare cases where the compression factor is constant and is known in advance.
- Another old code worth mentioning is the Baudot code. This was 5-bit code per character but encodes more than 32 characters, each 5-bit code can be code of two characters, a letter and a figure. The "letter shift" and "figure shift" are used to shift between letters and figures.
- If the data include just integers, each decimal digit may be represented in 4 bits, with 2 digits packed in a byte. With dates it may be represented as the number of days since January 1, 1900 or some other convenient start date. Each date may be stored as a 16 or 24- bit number (2 or 3 bytes). If the data consists of date/time pairs, a possible compressed representation is the number of Second since a convenient start date. If stored as a 32 – bit number (4 bytes) it can be sufficient for about 136 years.
- Dictionary data are stored lexicographically can be compressed using the concept of front compression. This is based on observation that adjacent words in such a list tend to share some of their initial characters.
- The 9/19/89 syndrome: How can a date, such as 11/12/71, be represented inside a computer? One way to do this is to store the number of days since January 1, 1900 in an integer variable. If the variable is 16-bit long, including 15 magnitude bits and one sign bit, it will over flow after 215 = 32k=32,768 days which is September 19, 1989. Notice that doubling the size of such variable to 32 bits would delayed the problem until after 231=2 giga days have passed, which would occur sometime in the fall of year 5,885,416 [9].

1.2 Run Length Encoding (RLE)

In this method the idea is if a data item 'd' occurs n consecutive times in the input stream, replace the n occurrences with single pair 'nd'.

To get an idea of the compression ratio produced by RLE, among a string of N characters that needs to be compressed. Also assume that the string contains M repetitions of average length L each. Each of M repetitions is replaced by 3 characters (i.e. escape, count, data) so the size of the compressed string is:

$$N - M*L + M*3 = N - M(L - 3)$$

And the compression factor is:

$$\frac{N}{N - M(L-3)}$$

Data Compression can be viewed as a means for efficient representation of a digital source of data such as text, image, sound or any combination of all these types such as video.

Pictorially a compression scheme can be as shown below in **Fig. (1)**

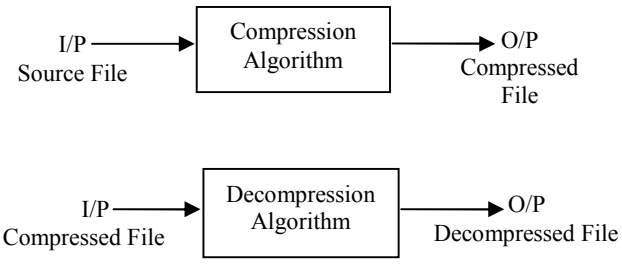

Fig. (1): Compression and decompression

At the time of transmissions the data will be transmitted through many mediums. There is possibility of going through many channels. It may be graphically understood as given in **Fig. (2)** below.

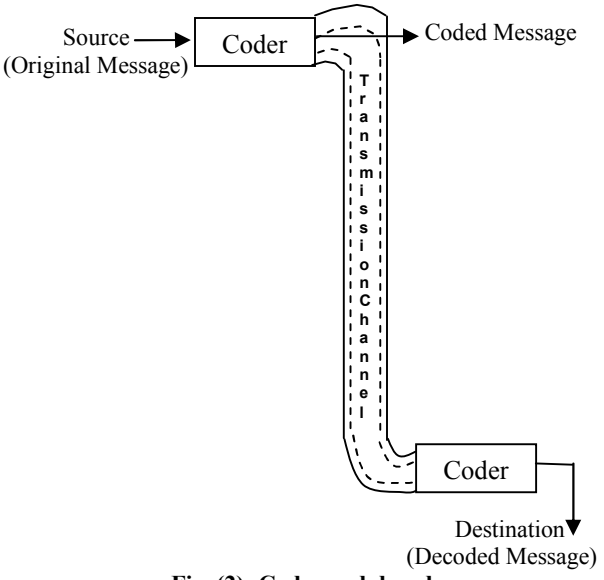

Fig. (2): Coder and decoder

All compression algorithms require two algorithms, as represented in Fig. (1), one for compressing the data at the source and another for decompressing it at the destination. These algorithms are referred to as the encoding and decoding algorithm respectively, as shown in Fig. (3). If the input and output are identical, the compression system is called lossless and when the decoded output is not identical to the original input the system is said to be lossy [16] [18]. Lossy systems are important because accepting a small amount of information loss can give a huge payoff in terms of the compression ratio possible. Compression scheme can be divided into two general categories [1]. Entropy encoding and Source encoding, Entropy encoding just manipulates bit stream without regard to what does the bit means, it is in general lossless fully reversible technique applicable to all data [19]. It includes run-length encoding, statically encoding (short code is used to represent the common symbol and long one to represent the infrequent one like Morse code, Huffman code or Color lookup table (CLUT) [4].

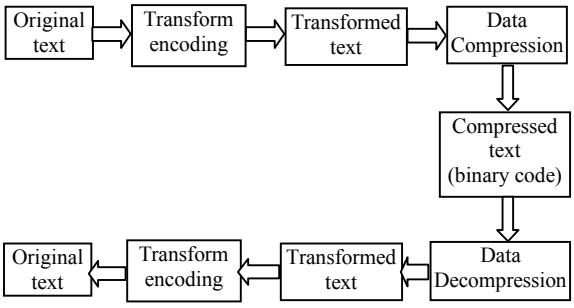

Fig. (3): Transformation Cycle of Text

Source encoding takes advantage of properties of the data under consideration to produce more compression, it includes Differential encoding, Transformation encoding, and Vector quantization.

In this paper Delta method has been used mainly in which the difference between two consecutive numbers value only are to be stored and hence the memory space occupied will be very less. And that is why this method is very easy and very much efficient and simple to implement.

Temporal databases encompass all database application that requires some aspect of time when organizing their information [6].There are many examples of applications of where some aspect of time is needed to maintain the information in a database. These include health care, where patient histories need to be maintained, insurance, where claims and accident histories are required as well as information on the times when insurance polices are in effect, reservation systems in general, where information on the date and time when reservation are in effect are required, bank data, and so on.

Together with temporal database this scheme could be used with the time series data as well, the time series data are used often in financial, sales and economics application. They involve data values that are recorded according to a specific sequence of time point. They are hence a special type of valid event data, where the event time points are predetermined according to a fixed calendar. For example the closing stock prices of a particular company on BSE [11] [12].

DATA compression is an encoding technique with which data size can be reduced according to some predefined rules pictorially shown in **Fig. (3).** There are two basic classes of data compression used in different areas. [1], [3], [5]. These are lossy data compression and lossless data compression. The lossy data compression finds its widespread use in the fields, such as image data or audio data that can accommodate the loss of some less important information within the data. On the contrary, the lossless data compression is used in the cases, such as data transmission and data storage that all information of the data needs to be kept intact, namely, the data must be able to recover completely thereafter. Some famous lossless data compression algorithms proposed is the past include Huffman codes [10], Shannon-Fanon code [2], Arithmetic coding [13], and Lempel and Zip [LZ] codes [14], [15]. Block diagrams presented in **Fig. (1) & (3), (4)** shows various predictors that can be used for lossless data compression.

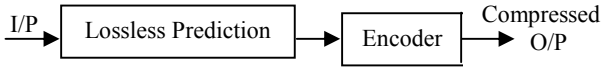

Fig. (4)

As we have seen that there is a scope of some improvement in ASCII code, the proposed solutions for compressing in better way is as given below. This method will give a very good performance of compression. Frequently used compression in computers is: Zip, gzip, winzip.

Examples of Application areas for compressions are

- Personal communication systems such as facsimile, voice mail and telephony.
- Computer Systems such as memory structures, disks and tapes.
- Mobile computing.
- Distributed computer systems.
- Computer Network especially the Internet.
- Multimedia evolution, imaging, signal Processing.
- Image archival and videoconferencing.
- Digital and satellite T.V.

2. PROPOSED METHODS

Following types of attribute value or data types are available, that can be used in a typical database management to represent the real world information.

1. Character (Printable and Non-printable)
2. Number (Integer and Float)
3. Date and time

Traditionally, these data type are represented in such a way, to make their processing in easier and efficient manner, without looking for how much space they takes. One exploits the fact that, a particular attribute has a limited domain and type when it is seen in the context of the database. In this it defines its own code that can handle all the domain values that are expected to be within an attribute.

A commercial database often supports following attribute types.

1. Character
2. Memo (alphanumeric)
3. Date
4. Numeric
5. Time

In proposed design following types of code and methods that are best suited for these attribute types are defined. This design mechanism can be applied either on a new file at the design time before storing anything in it, or can be applied on a file that have some data on it.

2.1 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. The front end of an application either data mining or database

often equipped with the reporting capability, so without bothering about upper case or lower case letter one can choose any one of them for our system and store the data according to it. Now for 26 character it requires only 5-bits[9] to represent any one of them and the remaining 6 combination of bits could be used to support the character that are required to manage the string or character database like end of line, space, full stop etc.

So it has redefined the coding in the following way.
00000-a, 00001-b
00010-c, 00011-d

00100-e and so on up to 11001-z. and the remaining 6 could be used for the following purpose.
11010-space
11011-end of line
11100-comma
11101-full stop
11110-“ “
11111-‘ ‘

So in this way a compact and complete representation of information is possible. Below **Table 1** will give the exact figure of compression, using ASCII and our above method. Comparative graph between ASCII and above method of compression is as shown in **Fig. (5).**

Table 1. Comparison between ASCII and Our method

Name	ASCII Bits	Comp-ressed Bits	Depart-ment	ASCII Bits	Comp-ressed bits
Sushil	48	30	Marketing	72	45
Manish	48	30	Finance	56	35
Avinash	56	35	Account	56	35
Ratnesh	56	35	HR	16	10
Sumit	40	25	Marketing	72	45
Vinay	40	25	Finance	56	35
Rajeev	48	30	Account	56	35
Total No. of bits	**336**	**210**		**384**	**240**

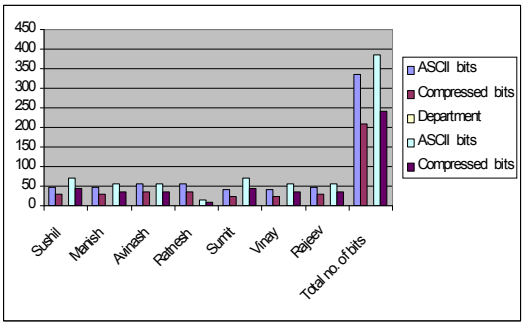

Fig. (5): Comparative Graph between ASCII and Character representation

2.2 Memo

The memo field often includes character other than alphabets like number underscore plus minus etc. and it is found that at most 61 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 one are used for numbers and special character like %, $, #,_.

These are assigned as follows.
000000-a 000001-b
000010-c, 000011-d
000100-e and so on up to 011001 for z.
And the remaining 38 could be used for the following purpose.
011010-space, 011011-end of line
011100-comma, 011101-full stop
011110-“ “ 011111-‘ ‘

100000-0, 100001-1,100010-2,100011-3 and so on up to 101001 - 9 to represent numeral. The remaining combinations are used to define the following symbols:
101010-! 101011-@ 101100- # 101101- $
101110- ^ 101111-& 110000- * 110001-(
110010-) 110011- - 110100- _ 110101- =
110110 - + 110111-< 111000-> 111001-?
111010-/ 111011- : 111100-; 111101-|
111110-\ 111111- Unused.

Memo field is often used to store information about address, telephone number etc. So if one wants to represent sushilkumar24@yahoo.com. It will take about 23 bytes but with this method it will just required 18 bytes all together. So with this coding mechanism slight space saving could be achieved [2].

Comparison between existing ASCII and above methods is as given below in tabular and graphical form **Table 2** and **Fig. (6).**

Table 2. Comparison between ASCII and Memo

Address	ASCII bits	Compre-ssed bits
# 90,Neerja Nagar, J.K. Road, Bhopal, Pin code 46202, Email: dixit_shrikrishna@yahoo.com, Mob:9713238730	816	612

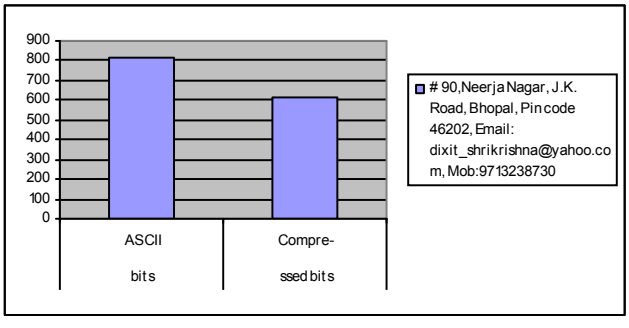

Fig. (6): Comparison between ASCII and Memo

2.3 Date

Date field is often associated with temporal database, and most often used inside the data ware house, that contain 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 it

proposed the following format **Fig. (7)** that takes only 2-byte to store any particular date as compared to 8-byte entry [9].

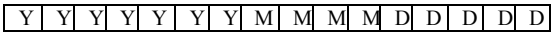

Fig. (7)

In this method any given date is converted in to a 16-bit number using the following formula.

Date = 512*(year-1980)+32*month + day

For example 09/09/2004 is any date, then it's corresponding 2-byte number will be 12585, and its corresponding binary equivalent is 0011000100101001. And putting it into our bit format **Fig. (8)** what one get is the binary equivalent of day, month and year. In this field for NULL 0 and for temporal variable UC or NOW 511 could be used.

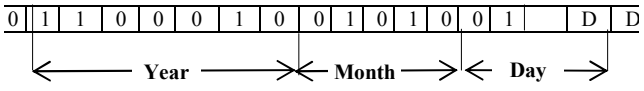

Fig. (8)

Now the conversion into actual date takes place in following manner

Year: - First right shifts the entry by 9 to get the year.

Month: - First left shifts the entry by 7 position followed by right shift by 12 positions.

Day: - First left shifts the entry by 11 positions followed by right shift of 11 positions.

2.4 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.

2.4.1 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.

2.4.2 Delta Codes

Sometimes, there may appear patterns of numbers that are practically unpredictable, but with adjacent terms close to each other, such as readings of temperature. What one could do is to record the first value, and from there record the difference to the next. For example:

{23, 27, 25, 24, 21, 19, 22, 22, 24, 27, 26}={23, +4, –2, –1, –3, –2, +3, +0, +2, +3, –}

Since the increment is from zero to three, we only need two bits to store them [6]. Note, however that the increments can be either positive or negative. This can be handled by the concept of negative binary numbers, where all integers – positive, zero, and negative can be represented as whole numbers.

2.5 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 one could save substantial space by representing a time stamp in a manner that require only 2-byte for its storage. The Bit-wise distribution of Hour, Minutes and Second is shown in following **Fig. (9).**

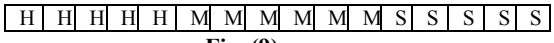

Fig. (9)

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 one wants to be more precise it has 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 a time value of 16:40:24 in HH : MM: SS format, applying this to formula one gets time = 34060 and its binary equivalent is as shown in **Fig. (10)**

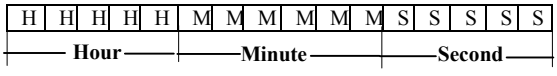

Fig. (10)

So in this way one can represent this time stamp. Now to get each of these separately it has to perform the bit wise shift operation in the following way. In this field 1952 could be used for temporal variable & 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 & multiply it by 2.

3. PERFORMANCE EVALUATION

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. Now we have following data table about various employees together with their annual salary, departments along with their joining dates in the following format of tabular form included their storage of bits required according to ASCII value as shown below **Table 3.**

In this above table take the base value for salary attribute is 115150. So the stored values will be 190, 740, 525, 830, 0, 75, and 25. That can be stored in less than 6 bytes as shown in below table. The total no. of bits required to store the above table is :

(336 + 384 + 336 + 448 + 336 + 336) = 2176.

By applying our method which has been discussed in proposed solution method in this paper, the new table which has been created after applying our differential method is as shown below **Table 4.**

After compression using differential method the total no of bits required is (210+240+102+112+112+112) =888. Now the compression is = 888/2176 = 0.408. So we have the compression ratio is 0.408, which is a good compression ratio. It means it compresses 60% of original size. Only we need 40% of original size.

In the another method, which is based on delta codes, we can arrange the contents of table according to ascending order of attribute salary and then use delta codes to store them and record the difference to the next. Before, use of delta code and contents of table are arranged in ascending order according to their Salary value. Its advantage is that, we can save one bit for each value, which is set to show sign of the difference as shown in **Table 5.**

In the above table the total no. of bits required is

(210 + 240 + 126 + 112 + 112 + 112) = 912. So the compression ratio is = 912/2176= 0.4191.

Comparing above two methods that is Delta methods and Differential compression method, the best one is differential compression method, since in Delta compression, compression ratio is 41.91% but in differential method is 40%. So the better choice is Differential method.

Finally compression between ASCII, Differential, and Delta are compared in tabular and graphical forms as shown below **Table- 6** and **Fig. (11).**

Table 6. Comparison between ASCII, Differential and Delta Code

ASCII code	Differential code	Delta code
2176	888	912

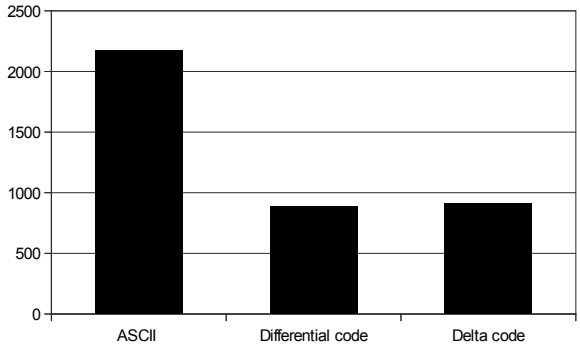

Fig. (11): Comparison between ASCII, Differential and Delta Code

4. CONCLUDING REMARK

Any compression scheme over daily operational database is not as efficient as it should be, because every time when a operation is performed (either deletion, insertion) it can only be performed on an uncompressed version, so either we first uncompress all the database then perform operation on them and then again recompress them at the end, but it is not a feasible solution at all [8]. So compression scheme work well only on those databases, which are required occasionally, so data warehouse are the best candidate for compression schemes [13]. Even when a new record has to refresh or loaded inside the warehouse one can insert the compressed format of that record at the end of the file without having to uncompress the original database [7]. So because of this, Warehouse is the best candidate for this above mentioned Delta and Differential compression scheme. Along with Data Warehouse this scheme can also be used with the hand held devices like in cell phones, palmtop, Discman, digital library etc. Now in the implementation how to handle the different group of bits in a fixed format system, this problem can be solved by a proprietary file system as used in system like Oracle, Sybase etc. or one can allocate the sufficient number of bytes to accommodate a particular field. For example one wants to store “” for that it need 6 bytes all together, but with modified 5 bit coding it can be represented in 30 bit, which means one can represent it with space of 4 bytes instead of 6 bytes. With this above-mentioned compression scheme or storage scheme query processing doesn’t requires the uncompressed format of data, rather than original query can itself be modified to handle this scheme and right now we are working on this aspect. The above compression between ASCII, Delta and Differential code, it can understand that 60% and more data have been compressed, which is one of the better results than any other method. As one can see in **Fig. (11),** the comparison between ASCII, Differential and Delta Codes the best Compression is Differential Code.

Table 3. Performance by taking ASCII value											
Name	**No. of bits**	**Depart-ment**	**No. of bits**	**Salary**	**No. of bits**	**Join bate**	**No. of bits**	**Time in**	**No. of bits**	**Time out**	**No. of bits**
Sushil	48	Marketing	72	115340	48	12/05/2003	64	10:35:40	48	16:30:00	48
Manish	48	Finance	56	115890	48	01/12/2002	64	09:40:40	48	15:50:00	48
Avnish	56	Account	56	115675	48	05/11/2004	64	11:00:00	48	16:00:00	48
Ratnesh	56	HR	16	115980	48	17/04/2002	64	10:30:30	48	16:15:00	48
Sumit	40	Marketing	72	115150	48	21/10/2001	64	09:50:50	48	16:20:20	48
Vinay	40	Finance	56	115225	48	25/05/1998	64	09:30:40	48	16:25:50	48
Rajeev	48	Account	56	115175	48	12/06/1995	64	10:00:00	48	16:30:35	48
Total No. Bits	**336**		**384**		**336**		**448**		**336**		**336**

Table 4. Performance by taking Differential Method											
Name	**No. of bits**	**Depart-ment**	**No. of bits**	**Salary**	**No. of bits**	**Join bate**	**No. of bits**	**Time in**	**No. of bits**	**Time out**	**No. of bits**
Sushil	30	Marketing	45	190	18	12/05/2003	16	10:35:40	16	16:30:00	16
Manish	30	Finance	35	740	18	01/12/2002	16	09:40:40	16	15:50:00	16
Avnish	35	Account	35	525	18	05/11/2004	16	11:00:00	16	16:00:00	16
Ratnesh	35	HR	10	830	18	17/04/2002	16	10:30:30	16	16:15:00	16
Sumit	25	Marketing	45	0	6	21/10/2001	16	09:50:50	16	16:20:20	16
Vinay	25	Finance	35	75	12	25/05/1998	16	09:30:40	16	16:25:50	16
Rajeev	30	Account	35	25	12	12/06/1995	16	10:00:00	16	16:30:35	16
Total No. Bits	**210**		**240**		**102**		**112**		**112**		**112**

Table 5. Performance by taking Delta Method											
Name	**No. of bits**	**Depart-ment**	**No. of bits**	**Salary**	**No. of bits**	**Join bate**	**No. of bits**	**Time in**	**No. of bits**	**Time out**	**No. of bits**
Sumit	25	Marketing	45	115150	36	21/10/2001	16	09:50:50	16	16:20:20	16
Rajeev	30	Account	35	25	12	12/06/1995	16	10:00:00	16	16:30:35	16
Vinay	25	Finance	35	50	12	25/05/1998	16	09:30:40	16	16:25:15	16
Sushil	30	Marketing	45	115	15	12/05/2003	16	10:35:40	16	16:30:00	16
Avinash	35	Account	35	335	18	05/11/2004	16	11:00:00	16	16:00:00	16
Manish	30	Finance	35	215	18	01/12/2002	16	09:30:40	16	15:50:00	16
Ratnesh	35	HR	10	90	12	17/04/2002	16	10:30:30	16	16:15:00	16
Total No. Bits	**210**		**240**		**126**		**112**		**112**		**112**

5. REFERENCES

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

[2] Cleary, J.G and I.H Witten "*Data Compression using Adaptive Coding and Partial String Matching"* (1984).

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

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

[5] M. Morris Mano "*Digital logic and Computer Design"* (Prentice-Hall of India Limited).

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

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

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

[9] Saloman D. *"Data Compression The Complete Reference"* Springer, 3rd Edition (2004).

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

[11] Ziv , Jacob & A. Lempel *"Compression of Individual Sequence via Variable Rate Coding"* IEEE Transaction on Information Theory, Year (1978).

[12] 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.

[13] Jianzhong Li and Hong Gao *"Efficient Algorithms for Online Analysis Processing On Compressed Data Warehouses"* Harbin Institute of Technology, China.

[14] 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.

[15] 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.

[16] '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.

[17] 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.

[18] 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.

[19] Ying Li and Khalid Sayood, *"Lossless Video Sequence Compression Using Adaptive Prediction",* 1057-7149, © 2007 IEEE.