Improving Compression Efficiency of Data Warehouse

Mohd. Fraz, Ajay Indian, Hina Saxena, Saurabh Verma

Abstract— Data compression has a paramount effect on Data warehouse for reducing data size and improving query processing. Distinct compression techniques are feasible at different levels, each of types either give good compression ratio or suitable for query processing. This paper focuses on applying lossless and lossy compression techniques on relational databases. The proposed technique is used at attribute level on Data warehouse by applying lossless compression on three types of attributes (string, integer, and float) and lossy compression on image attribute.

Index Terms— Attribute Level Compression, Data Type of Attributes, Huffman coding, JPEG coding

1 INTRODUCTION

With the steady increase in database volume, the transaction overhead is also increased. Since, it becomes almost mandatory to manage the data in an organized and an efficient way. The Data Warehouse technique enables us to store static data in an organized way, as data once committed cannot be edited or deleted, which is always fruitful for decision support and relieves us from daily transaction overhead [6]. It is very convenient to store compressed data in data warehouse to save disk storage [1]. Other reasons for storing data in compressed way are:

- Reduced query execution time as static data is stored in data warehouse.
- Reduced CPU overhead as it needs to search data in less space.
- Reduced data redundancy.

There are four levels at which compression can be performed on Data warehouse [3] - File level compression, Page level compression, Record level compression and Attribute level compression. All defined compression levels have their own merits and demerits. In terms of compression ratio, File level and Page level compression are better however, for query processing, they do not perform well, as entire file or page has to be compressed and decompressed which gives more overhead on CPU unnecessarily, hence performance degrades. On the other hand, Record level compression and Attribute level compression perform well but, does not give good compression ratio in comparison to the first two types [3].
We propose the technique for compression at attribute level. Existing techniques emphasized on fixed length coding on all attributes [16] however, variable length coding always perform well over fixed length coding [3]. Our compression technique uses variable length coding which will be applied on Relational Database that includes both lossless and lossy compression. Lossless Compression for three data types (string, integer and float) and lossy compression on image attribute is done in a relation.

- *Mohd. Fraz is currently pursuing M.Tech. in Invertis University, India, Email: mfaraz83@gmail.com*
- *Ajay Indian is working as a Head- Deptt. of Computer Applications, Invertis University, India. Email: ajay.i@invertis.org*
- *Hina Saxena is currently pursuing M.Tech. in Invertis University, India, Email: poojabhatnagar18@gmail.om*
- *Saurabh Verma is currently pursuing M.Tech. in Invertis University, India, Email: shahzada29saurabh@gmail.com*

2 CLASSIFICATION OF COMPRESSION TECHNIQUES

There are two types of compression techniques exist. Their classification can be described below in Figure 1.0

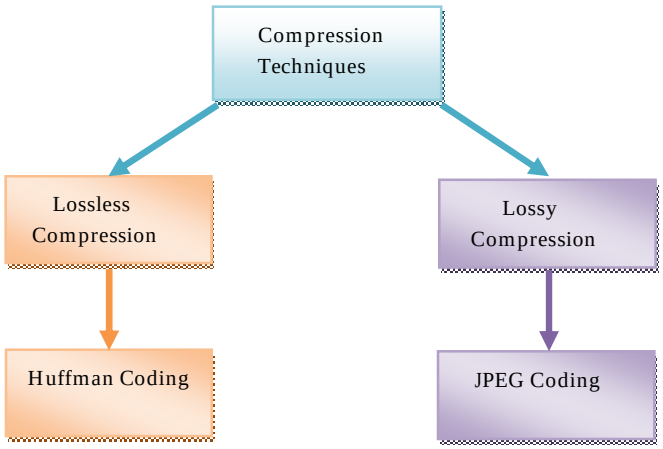

Figure 1.0

Lossless compression is one in which there is no loss of information when data is decompressed to its original form however in lossy compression, there is a loss of some information when decompressed [17].

2.1 *Huffman Coding* [1]

Huffman Encoding Algorithm is a variable length coding which uses the probability distribution of the alphabets of the

International Journal of Scientific & Engineering Research Volume 4, Issue ȝ, Ju0à2013
ISSN 2229-5518

source to develop the codeword for symbols. The frequency distribution of all the characters of the source is calculated in order to calculate the probability distribution. According to the probabilities, the codeword are assigned. Shorter codeword for higher probabilities and longer codeword for smaller probabilities are assigned. For this task, a binary tree is created using the symbols as leaves according to their probabilities and paths of those are taken as the codeword. Two families of Huffman Encoding have been proposed: Static Huffman Algorithms and Adaptive Huffman Algorithms. Static Huffman Algorithms calculate the frequencies first and then generate a common tree for both the compression and decompression processes [1]. Details of this tree should be saved or transferred with the compressed file. The Adaptive Huffman algorithms develop the tree while calculating the frequencies and there will be two trees in both the processes. In this approach, a tree is generated with the flag symbol in the beginning and is updated as the next symbol is read.

2.2 *JPEG Coding* [10, 13]
A lossy approach, JPEG coding (Figure 1.1) is a well known and popular standard for compression of still images. The source image is divided into 8×8 blocks and each block is transformed using DCT (Discrete Cosine Transform). The data compression is achieved via quantization followed by variable length coding (VLC). The quantization step size for each of the 64 DC coefficients is specified in a quantization table, which remains the same for all blocks in the image. In JPEG, the degree of compression is determined by a quantization scale factor.

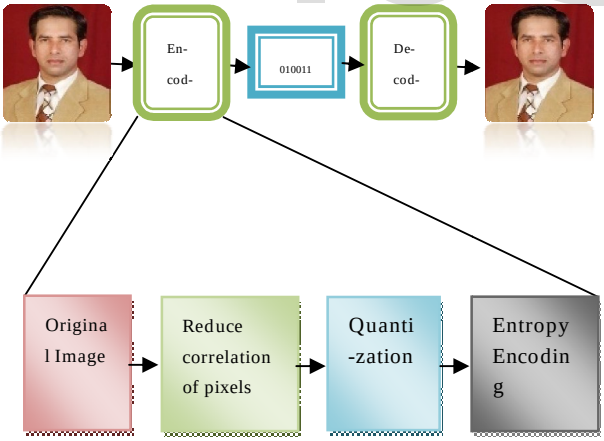

Figure 1.1

Increasing the quantization scale factor leads to coarser quantization, this gives higher compression and lower image quality. The DC coefficients of all blocks are coded separately, using a predictive scheme. JPEG is very efficient coding method but the performance of block-based DCT scheme degrades at very high compression ratio.

3 RELATED WORK

3.1 Page level compression

3.1.1 Compressing numeric attributes [8]

A page may have both numeric and non-numeric attributes. To compress numeric attributes a technique was developed in which a column containing integers were scanned to find out the lowest and highest integer value. Thus, all the values lying in between these two values are assigned the bits starting from 0. For example: - if a column on a page has 20 as minimum value and 24 as maximum then a range is defined. This range will be consisting of only 5 values and we can specify a given value in this range by using only 3 bits i.e. 20 as 000, 21 as 001, 22 as 010, 23 as 011 and 24 as 100. This minima and maxima provide a frame of reference in which all the values lie.

3.1.2 *Compressing non-numeric attributes* [8]
The same technique could be used for compression of non-numeric attributes which have low cardinality or one can say which have high redundancy in values of an attributes like gender, state or country. A high compression ratio can be achieved in such type of attributes. For example: - Gender attribute contains only two values for all the records. Therefore, only two bits are used to define its values 0 for male and 1 for female. Similarly, state and country are other attributes which have a very limited range of values.

3.2 Attribute Level Compression

The main objective of this technique is to allow the reduction of the space occupied by dimension tables with high number of rows, reducing the total space occupied by the data warehouse and leading to a consequent gains on performance[12]. It is aimed to compress two different types of database attributes:-

- Text attributes with low cardinality (known as categories).
- Attributes of free text: comments or notes (known as descriptions).

Categories [12] are those textual attributes that have a low cardinality. For example: city, country, gender, etc. Compression can be done on these types of attributes using 1 or 2 bytes. Only 1 byte is enough if value of such kind of attributes are less than 256 , and 2 bytes will be required if values are up to 65536.
A Description is a text attribute that is mainly used for visualization. For example: description attribute is the comment attribute, which is frequently found in dimension tables. This type of attributes has the particularity of having a low access frequency and it is only necessary to decode it when the final result is being constructed.

3.2.1 Categories Coding [12]
Categories coding is done through the following steps:

1. The values of the attribute are analyzed and its frequency is

International Journal of Scientific & Engineering Research Volume 4, Issue ȝ, Ju0à-2013
ISSN 2229-5518

calculated.
2. The table of codes is build based on the frequency: the most frequent values are encoded with a one byte code; the least frequent values are coded using a two bytes code. In principle, two bytes are enough, but a third byte could be used if needed.
3. The codes table and necessary metadata is written to the database.
4. The attribute is updated, replacing the original value by corresponding codes (the compressed values).

3.2.2 Descriptions Coding [12]
It is very similar to the categories coding with the major difference that in this case the value in the attribute is not regarded as a single value, but as a set of values (an ASCII string). Any text compression algorithm can be used to perform this type of compression.

4 PROPOSED WORK

In this paper, Huffman coding has been applied on three data types (string, integer and float), which is lossless by nature and JPEG coding is applied on image attribute, which is a lossy algorithm. Huffman coding is a variable length coding which gives better compression ratio in comparison to the previous techniques.

4.1 *How to compress text, integer and string attributes*
A variable length coding has been used i.e. Huffman coding. It is a lossless algorithm which goes through all the alphabets/numbers of an attribute and finds its frequency in it. Afterwards it generates the tree and assigns binary codeword for that attribute. More the frequency of an element, less the code size it gets. In the same way all the attributes are encoded.

4.2 *How to compress image attribute*
JPEG technique has been used to compress the image attribute. Once the correlation between pixels is reduced, we can take advantage of the statistical characteristics and the variable length coding theory to reduce the storage quantity.

5 RESULTS AND DISCUSSION

The database table is created and Huffman and JPEG coding techniques have been applied to evaluate the result. Several tables can be created and can be compressed using the proposed technique. We have created a table named “DbEmp”. This table contains string, integers, float and image type attributes. Other fields can also be included like email id, address, mobile number etc. which can be compressed by considering them as string type.
After applying compression techniques on the table, the below result shows the Original Attribute Size and Compressed Attribute Size with the help of graph in Figure 1.2. Compression Ratio and Gain in performance are shown with the help of Table 1.0.

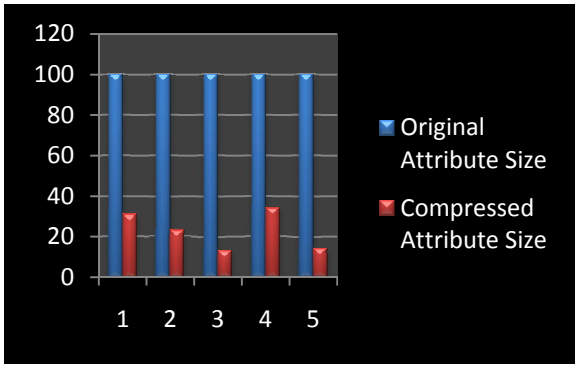

Figure 1.2

Attribute Name	Initial Size(bits)	Compressed Size(bits)	Gain (%)	Compression ratio (%)
Emp ID	72	22	69.44	30.56
Emp Name	172	42	76.74	23.26
Emp Age	16	2	87.5	12.5
Emp Sal	32	11	65.62	34.38
Emp Pic	4075	550	86.5	13.5
Average	873	125	77.16	22.84

Table 1.0

6 CONCLUSION AND FUTURE WORK

At last, it is concluded that an effective compression technique has been proposed that performs appropriate compression on Data warehouse. Compression at attribute-level is done for four types of data types. This compression technique can be applied to multi-databases. One can see the status of the table being compressed. The obtained results show that an overall Compression ratio of 22.84% is achieved, which is the best ratio ever achieved, and Gain in performance (saving in space) has increased to 77.16%. In the proposed work, we have considered only four data types but there are so many other data types like Boolean, short int, long int etc which can be compressed. Moreover, the compression of primary key and foreign key is also a main area for research as it contains all the unique values.

REFERENCES

[1] D. Huffman “A method for the construction of minimum-redundancy codes” Proc. IRE, 40(9):1098-1101, September 1952.W.
[2] J.Ziv and A. Lempel “A universal algorithm for sequential data compression” in IEEE *transactions in information theory*, vol. 3, No. 3, pp. 337-343, 1977.
[3] Debra A. Lelewer and Daniel S. Hirschberg “Data Compression”, 1987.

International Journal of Scientific & Engineering Research Volume 4, Issue ȝ, Ju0à-2013
ISSN 2229-5518

[4] G.Graefe and L.Shapiro "Data compression and database performance" In ACM/IEEE-CS Symp. On *Applied computing* pages 22 -27, April 1991.
[5] Mark A. Roth and Scott J. Van Horn "Database compression" *SIGMOD RECORD*, Vol. 22, No. 3, September 1993.
[6] G. Ray, J. R. Haritsa, and S. Seshadri. "Database compression: A performance enhancement tool" *International Conference on Management of Data*, pages 0, 1995.
[7] W. Kim "Modern Database Systems", *ACM Press*, New York, 1995.
[8] Jonathan Goldstein, Raghu Ramakrishnan and Uri Shaft "Compressing relations and indexes" in *IEEE computer society*, Washington, USA, 1998.
[9] Till Westmann, Donald Kossmann "Implementation and performance of compressed database" in *Reihe Informatik*, March 1998.
[10] Gregory K. Wallace "The JPEG still picture compression standard" in *IEEE*, December 1999.
[11] S. Chaudhuri and U. Dyal, "An overview of Data warehousing and OLAP Technology", *ACM SIGMOD* record, vol 21, No.1.
[12] Jorge Vieira, Jorge Bernardino and Henrique Madeira "Efficient compression of text attributes of data warehouse dimensions", 2005.
[13] Wei-Yi Wei "An Introduction to image compression", 2008.
[14] Amy Turske McNee "The Evolutionary Data Warehouse-An Object-Oriented Approach", 2008.
[15] Akanksha Baid and Swetha Krishnan"Binary Encoded Attribute-Pairing Technique for Database Compression" 2008.
[16] Meenakshi Sharma and Sonia Dora "Efficient Approach for Compression in Data Warehouse" *IJCA* (0975-8887) Volume 53 – No.9, September 2012.
[17] Mark Nelson and Jean-loup Gailly "The Data Compression Book" 2nd Edition.