ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Database Compression: Iterative Algorithm Technique

Deepankar Kumar Upadhyay

Abstract

In today's generation data storage and handling is one of the major issues especially when we talk about data's in big volume. And as the data keep on multiplying, storage and transmission of data requires more money. This scenario becomes more challenging when the data is in real time environment. For this reason we need to compress the data and store it. We have some orthodox methods of real time database compression and an iterative algorithm technique. In this study we will compare the various compression methods with iterative algorithm technique which provides parallel storage backup for optimization of real time database to achieve higher efficiency with better performance. The analysis and experimental results show that the iterative algorithms have better performance than the traditional algorithms.

Keywords

Compression, Backup Optimization, Database compression, Iterative algorithm, Compression ratio.

I. Introduction

A very large scale databases normally have very large size and a high degree of scarcity. That has made database compression as very important. Research in this area has considered various aspects of the problem such as developing a model for database compression, decompression and maintaining them. To protect the data against loss, users must store a copy of data in some secondary location. The database files used to store the database are sorted by disk device and a reader thread is assigned to each device, this thread reads the data from database file. A writer thread is assigned to each backup device which writes data on backup device. Parallel read operation can be increased by spreading the database files among more logical drives. Similarly, Parallel write operation can be increased by using more backup devices. The advantageous effects of data compression on I/O performance in database systems are rather obvious, i.e., its effects on disk space, bandwidth, and throughput. When we introduce database compression into real time databases. The compression algorithm must provide high compression ratio to realize high number of data storage and also the compression algorithm must be fast enough to fulfil the function of real time record and query in real time database. The compression process consists of two separate activities, modelling and coding. Modelling defines how different symbols in the input stream will be characterized. A model stores information on how frequently the symbol had occurred in the data, that is, symbol probabilities.Coding, the second part of the compression process, results in a compressed version of the data by creating a set of codes for the distinct symbols based on the probabilities provided by the model. Preferably, symbols that occur more frequently are interchanged with shorter code words and rare symbols with longer.

The homogeneity of data can affect the compression ratio of most compression algorithms, but it doesn't have any effect on compressed speed. So to achieve better compression performance the compression algorithms are specially designed for every portion of the data. Proposed algorithm provides the solution for above issue and increase the compression ratio at each scan of database system, resulting in increased application and business availability for our critical database environment

II. Background

Types of Data Compression

- Lossy data compression
- Lossless Data compression

Lossy Data Compression

This technique is most commonly used to compress multimedia data (Audio, Video & Images). Lossy Compression is irreversible compression technique which uses inexact approximations and partial data discarding to represent the content. The amount of data reduction possible using lossy data compression is often much higher than through lossless techniques. The original data contains a certain amount of information basic information theory says that there is an absolute limit in reducing the size of this data. When data is compressed, its entropy increases, and it cannot increase open-endedly. For example, a ZIP file is smaller than the original file, but repeatedly compressing the same file will not reduce the size but instead will increase the size.

Methods of Lossy Data Compression:

AUDIO DATA-A-Law, μ-law, and Algebraic code excited linear prediction (ACELP), Adaptive differential pulse code modulation (ADPCM), Fourier Transform, Differential pulse code modulation (DPCM).

VIDEO DATA– Lapped transform, discrete cosine transform (DCT), DE blocking filter, Motion compensation.

IMAGE DATA–Chain code, Discrete cosine transform (DCT), Embedded zero trees of wavelet transform

Lossless Data Compression

Unlike lossy data compression where some loss is acceptable in number of bits received , Lossless data compression system is one where data recovered after de- compression must contain exactly same number of bits as the input. If there is any loss in no of bits received after de- compression then this will corrupt the resulting information being conveyed by input stream. Methods of Lossless Data Compression

ENTROPYTYPE: Unary, Arithmetic, Huffman(Adaptive, Canonical, Modified), Shanno-fano.

DICTIONARY TYPE:Byte pair, Encoding, Lempel-Ziv (LZ77/ LZ78), LZJB,LZMA.

III. Performance Optimization Using ILC Algorithm

The planned work is efficiently designed and developed for a backup compression process for real-time database systems using ILC algorithm and can allow the compressed backups to be store in multiple storages in parallel. The planned ILC with parallel storage backup for real time database systems comprises of three phases.

Phase I: To identify and analyse the entire database, the next step is to compress the database in order to take backups and the last step is to store the compressed backups in multiple storages in parallel. The assembly of the proposed ILC based real time database compression optimization is shown in fig 1. The first phase is to analyse the database environment in which it was created. At forts, the attributes present in the database systems are analysed and identify.

Phase II: Defines the process of compression and decompression of the database using Iterative Length Compression (ILC) algorithm. The ILC algorithm is used to offer a good compression technique by allowing access to the database level and enhances the compression ratio for easy backup of database systems.

Phase III: Defines the process of storing the compressed backups at different levels of storages in parallel. The copies of compressed backups are always accessible at any system, and there is less chance of database systems to be lost and hence can easily be recovered.

IV. Experimental Evaluation

The proposed backup compression process for real-time database systems implementing ILC algorithm is based on 1GB sample database. The experiments were run on an Intel Core 2 Duo P-IV machine with 3 GB memory and 2GHz processor CPU. The planned ILC based compression model for real time environment is efficiently designed for compression and taking backup of compressed data with the database systems. Considering both compression ratio and speed, it is appropriate to use ILC algorithm or its variants to compress the data. Data volume is not only the most vital portion in the data structure, but also the least consistent portion. But we can still discover some rule in its data curve. The correlativity of data value is fragile and there is usually small movement between two neighbourhood data points. Quality code has the maximum redundancy in all three varieties of data. It seldom jumps and always keeps the same value, which is suitable to be compressed by ILC compression algorithm too. The test for simulation of data indicates that the compression ratio of ILC algorithm for quality code can achieve 85% and the time for compression and decompression can be considered as very less.

Compression Ratio: is the ratio of size of the compressed database system with the actual size of the uncompressed database systems. Also known as compression power is a computer-science term used to measure the reduction in data-representation size produced by a data compression algorithm. Compression ratio is defined as follows:

Compression Ratio = Uncompressed Size / Compressed Size

Disk Storage & Space Savings: When either type of compression is used, there is a multi-way trade-off involved between storage space (disk and buffer pool) and I/O reduction (due to better memory caching. Sometimes the space savings is given instead, which is defined as the reduction in size relative to the uncompressed size.

Space Savings = 100 * (1 - Compressed Size / Uncompressed Size).

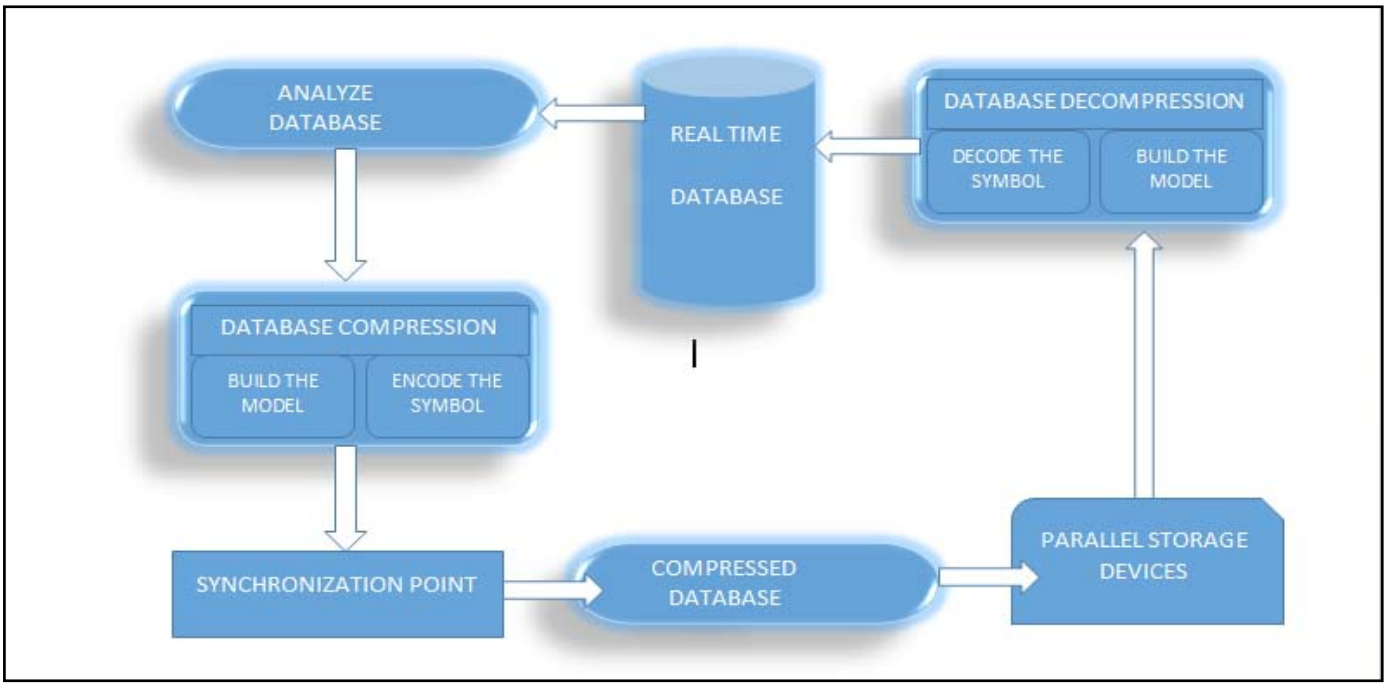

Fig. 1: Structure of Database Compression and Decompression

V. Results and Discussion

In this paper, we have seen how the database is efficiently compressed using backup compression process for real-time database systems using ILC algorithm. In order to check the efficiency of proposed ILC Algorithm for backup compression process for real-time database system we used a real time 1GB sample database for experimentation. The backup compression storage space savings for the uncompressed database is more than twice the backup compression space savings for the compressed database, which is to be predicted, given that the latter database is already compressed. The table and diagram given below defined the compression ratios and storage space savings of the proposed backup compression process for real-time database systems using ILC Algorithm.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Table1: Compression Ratio vs. Space Savings

Compression Type	Compressed File Size (MB)	Compres-sion Ratio	Space Saving
Uncompressed	1024	1.00	0%
RLE Compres-sion	580	1.77	43%
Dictionary Com-pression	525	1.95	49%
ILC Compression	370	2.76	64%

The above table (table 1.) describes the compression ratio and space savings based on size of data existing in the database. The effectiveness of compression using the proposed backup compression process for real-time database systems by ILC algorithm is compared with existing compression algorithm. For systems with auxiliary CPU cycles for performing compression, the objective is to reduce the amount of storage needed for holding the database. The following graph depicts how much compression can reduce the amount of disk storage needed.

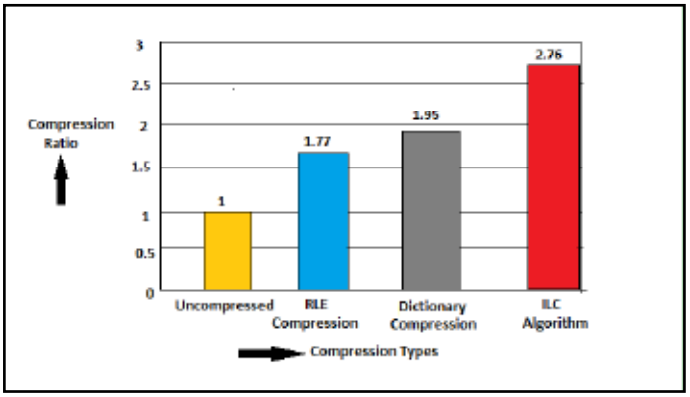

Fig. 2 : Compression Types vs. Compression Ratio

Fig.2 defines the process of compression ratio based on different types of existing compression algorithms. As size of the database increased the compression ratio of the database is decreased in the proposed compression algorithm. So, the compression ratio becomes less in the planned backup compression process for real-time database systems with parallel multi-storage process. The compression ratio is measured in terms of megabyte (M byte). Compared with existing compression algorithms, the proposed ILC algorithm achieves goodcompression rate of 50%more.

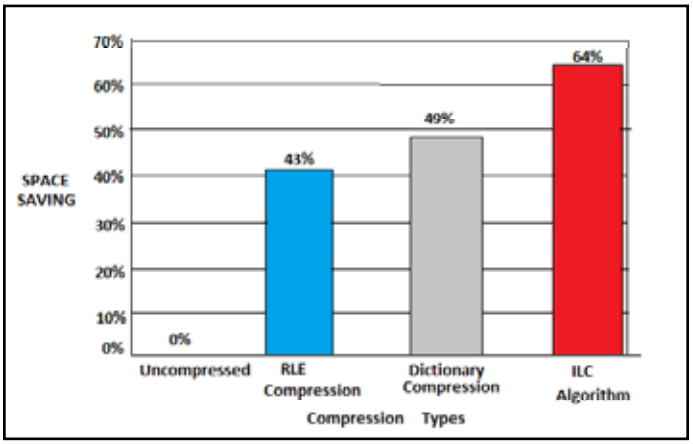

Fig.3 : Compression Types vs. Space Savings

Fig.3. defines the process of space savings based on different types of existing compression algorithms. The space savings is measured in terms of megabyte (M byte). Compared to an existing compression algorithm, theplanned ILC algorithm attains less storage space and the variance would be about 60% better.

Table 2: Performance Testing of Data Compression

Compression Type	Compressed File Size (MB)	Compres-sion Time	Decom-pression time
Uncompressed	1024	2:35	1:22
RLE Compres-sion	580	1:40	1:06
Dictionary Compression	525	1:27	0:58
ILC Compres-sion	370	1:04	0:39

The above table defines the time consumed for compression and decompression of the proposed backup compression method. The result of performance test is shown in table 2. The compression algorithm achieves good performance. In this test we compared existing algorithms and ILC algorithm. As shown in table 1 and 2, the new compression algorithm is superior in all respects in comparison to existing compression algorithm in the real-time database.

VI. Conclusion

Proposed compression approach will allow efficient recovery of the compressed data, as well as yield a valuable saving in storage costs. Experimental results have shown that the proposed backup compression method for real time database systems using ILC algorithm are effective in terms of storage space, compression ratio and backup processing compared to an existing compression algorithms. Proposed compression algorithm also improves CPU performance by allowing database operators to operate directly on compressed data and provides good compression of large databases.

References

[1]. *SushilaAghav, "Database compression techniques for performance optimization", proceedings in 2ndinternational conference onComputer Engineering and Technology (ICCET), 2010.*

[2]. *Adam Cannane , Hugh E. Williams "A Compression Scheme for Large Databases" proceedings in 11th international conference on ADC, 2010.*

[3]. *D. J. Abadi. Query execution in column-oriented database systems. MIT PhD Dissertation, 2008.PhD Thesis.*

[4]. *ChenggangZhen ,BaoqiangRen, "Design and realization of data compression in Real-time database", proceedings in 10th international conference on Computational Intelligence and Software Engineering ,2009.*

[5]. *Veluchandhar, R V. Jayakumar, M. muthuvel, K. Balasubramanian, A.Karthi,Karthikesan, G.Ramaiyan, ,A.Deepa.S.AlBertRabara, "A Backup Mechanism with Concurrency Control for Multilevel Secure Distributed Database Systems", proceedings in 3rd international conference on Digital Information Management (ICDIM), 2008.*

[6]. Hung-Yi Lin and Shih-Ying Chen, "High Indexing Compression for Spatial Databases", proceedings in IEEE 8th international conference on Computer and Information Technology Workshops, 2008.

[7]. IoannisKontoyiannis and Christos Gioran "Efficient Random Codebooks and Databases for Lossy Compression in Near-Linear Time", proceedings in IEEE Information Theory Workshops on Networking and Information Theory, 2009.Computer Society Press, Los Alamitos, California. A. Cannaneand H.Williams. General-purpose compression for efficient retrieval. Technical Report TR-99-6, Department of Computer Science.