REAL TIME DATABASE COMPRESSION OPTIMIZATION USING ITERATIVE LENGTH COMPRESSION ALGORITHM

Muthukumar Murugesan[1] and T Ravichandran[2]

[1]Research Scholar, Karpagam University,
Coimbatore, Tamilnadu-641021, India,
[2]Principal, Hindusthan Institute of Technology,
Coimbatore, Tamilnadu-641032, India,
amgmuthu@yahoo.com and dr.t.ravichandran@gmail.com

ABSTRACT

Real time databases are processing millions and billions of records on daily basis. Handling huge volume of data's in real time environment are big challenges in the world. Global domain providers will have a massive number of transactions on day-by-day. Such a domain database will be grown GBs and TBs of data during daily activities. Traditional compression methods are progressively more difficult to dealing with very large databases.

In this paper, we provide the solution to how to optimize and enhance the process for compress the real time database and achieve better performance than conventional database systems. This research will provides a solution to compress the real time databases more effectively, reduce the storage requirements, costs and increase the speed of backup. The compression of database systems for real time environment is developed with our proposed Iterative Length Compression (ILC) algorithm.

KEYWORDS

Database compression, Database decompression, Backup optimization, Real-time database

1. INTRODUCTION

Data available in a real time database are highly valuable. Commercially available real time database systems have not heavily utilized compression techniques on data stored in relational tables. A typical compression technique may offer space savings, but only at a cost of much increased query time against the data. Global domain provider used to take the backup of their database multiple times in a day also there are many providers used to take the data backup once in few hours. This kind of daily activities will consume considerable time, volume of resources and highly expensive.

Data compression algorithms have been around for nearly a century, but only today are they being put to use within mainstream information systems processing. Data stored in databases keep growing as a result of businesses requirements for more information. A big portion of the cost of keeping large amounts of data is in the cost of disk systems, and the resources utilized in managing that data. There are several places where data can be compressed, either external to the database, or internally, within the DBMS software.

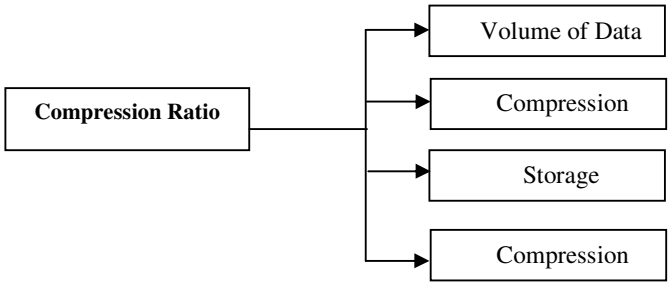
Jan Zizka (Eds) : CCSIT, SIPP, AISC, PDCTA - 2013
pp. 99–105, 2013. © CS & IT-CSCP 2013 DOI : 10.5121/csit.2013.3610

Over the last decades, improvements in CPU speed have outpaced improvements in disk access rates by orders of magnitude, motivating the use of backup compression techniques to trade reduced disk I/O against additional CPU overhead for backup compression and decompression of real time large database systems. Backup compression provides combination to reduce database and backup storage costs. Backup compression is effective in general, even with an already compressed database, and reduces both storage and elapsed times for backup and restore. Data compression affects the physical storage of columns within a row and rows within a page on disk and in memory.

Note that backup compression significantly increases CPU usage, and the additional CPU consumed by the compression process might adversely impact concurrent operations. On the plus side, backup sizes and backup/restore elapsed times can be greatly reduced. Backup compression offers the following benefits.

- Reducing Storage Requirement
- Data Transfer Rate
- Enhancing Data Security
- Backup and Recovery
- Performance Enhancement

When introducing data compression technology into real-time database, two requests must be satisfied. First, the compression algorithm must provide high compression radio to realize large numbers of data storage in real-time database. Second, the compression algorithm must fast enough to satisfy the function of real-time record and query in real-time database. In order to achieve better compression performance, the compression algorithms are specially designed for every portion of the data. The flow chart for historical data compression attributes is shown



ILC algorithm provides the solutions for the above issues and repeatedly increases the compression ratio at each scan of the database systems. The quantity of compression can be computed based on the number of iterations on the rows.

2. BACKGROUND

The compression process consists of two separate activities, modeling and coding. Modeling defines how each of the distinct symbols in the input stream will be represented. A model stores information on how often the symbol 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 constructing a set of codes for the distinct symbols based on the probabilities provided by the model. Ideally, symbols that occur more frequently are replaced with shorter code words and rare symbols with longer.

This proposed approach require two separate passes of the data: the first gathers statistics on the symbols necessary to build the model; and the second encodes the symbols according to the model. A semi-static model makes good use of specific properties of the data, while at the same time remains static during decoding. Real-time database is the combination of real-time system technology and database technology. Real-time database has the characteristic of high speed, high data throughput and so on.

3. COMPRESSION OPTIMIZATION USING ILC ALGORITHM

The proposed 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 store it in multiple storages in parallel. The proposed ILC with parallel storage backup for real time database systems comprises of three operations. The first operation is to identify and analyze the entire database, the next step is to compress the database systems to take up as backups and the last step is to store the compressed backups in multiple storages in parallel. The structure of the proposed ILC based real time database compression optimization is shown in fig 1.

The first phase is to analyze the database based on the environment in which it creates. At forts, the attributes present in the database systems are analyzed and identify the goal of the database creation and maintain a set of attributes and tables maintained in the database systems.

The second phase describes the process compression and decompression of the database using Iterative Length Compression (ILC) algorithm. The ILC algorithm is used to provide a good compression technique by allowing access to the database level and enhances the compression ratio for a ease backup of database systems.

The third phase describes the process of storing the compressed backups at different levels of storages in parallel. The copies of compressed backups are always available at any system, there is less chance of database systems to be lost and can easily be recovered.

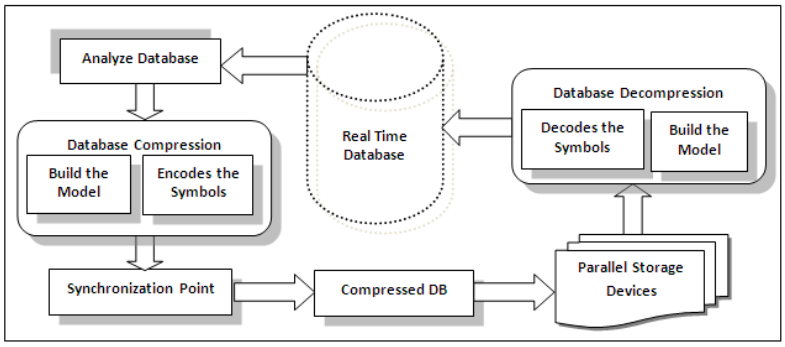


Fig.1. Structure of Database Compression and Decompression

4. EXPERIMENTAL EVALUATION

The proposed backup compression process for real-time database systems implementing ILC algorithm with 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 proposed ILC based compression model for real time environment is efficiently designed for compression and taking backup compressed data with the database systems. Considering both compression radio and speed, it is suitable to use ILC algorithm or its variations to compress historical data.

Data volume is not only the most important portion in the data structure, but also the least regular portion. But we can still find some rule in its data curve. The correlativity of data value is weak and there is usually small wave between two neighborhood data points. Quality code has the highest redundancy in three kinds 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 data indicates that the compression radio 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 original size of the uncompressed database systems. Also known as compression power, is a computer-science term used to quantify 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), 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)

5. RESULTS AND DISCUSSION

In this work, we have seen how the database is efficiently compressed using backup compression process for real-time database systems using ILC algorithm. We used a real time 1GB sample database for an experimentation to examine the efficiency of the proposed backup compression process for real-time database systems. The backup compression storage space savings for the uncompressed initial database is more than twice as much as the backup compression savings for the compressed database, which is to be expected, given that the latter database is already compressed. The below table and diagram described the compression ratios and storage space savings of the proposed backup compression process for real-time database systems using ILC algorithm.

Compression Types	Compressed File Size (MB)	Compression Ratio	Space Savings
Uncompressed	1024	1.00	0%
RLE Compression	580	1.77	43%
Dictionary Compression	525	1.95	49%
ILC Compression	370	2.76	64%

Table 1. Compression Ratio vs Space Savings

The above table (table 1.) described the compression ratio and space savings based on size of data present in the database. The efficiency of compression using the proposed backup compression process for real-time database systems using ILC algorithm is compared with an existing compression algorithms. For systems with spare CPU cycles for performing compression, the objective may be just reducing the amount of storage needed for holding the database. The following graph shows how much compression can reduce the amount of disk storage needed.

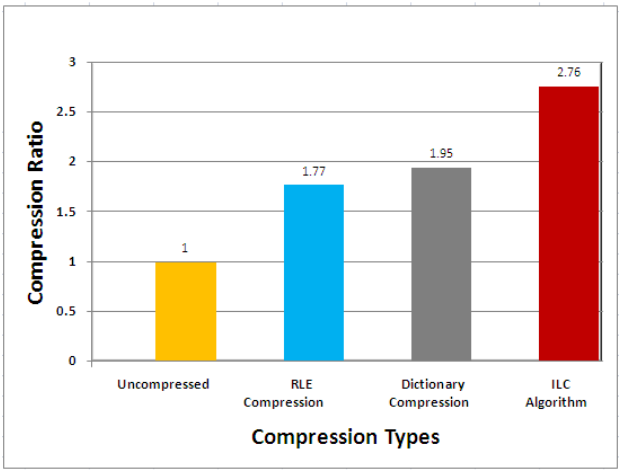


Fig. 2 Compression Types vs Compression Ratio

Fig.2 describes the process of compression ratio based on different types of existing compression algorithms. When size of the database increases the compression ratio of the database is decreased in the proposed compression algorithm. So, the compression ratio becomes less in the proposed backup compression process for real-time database systems with parallel multi-storage process. The compression ratio is measured in terms of megabyte (mb). Compared to an existing compression algorithm, the proposed ILC algorithm achieves good compression rate of 50% more.

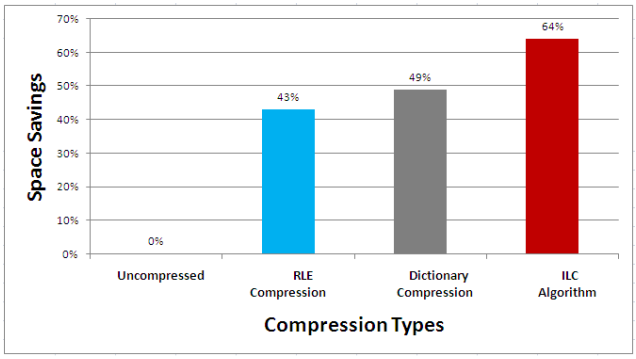


Fig.3. Compression Types vs Space Savings

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

Proposed compression significantly decreases disk storage and backup/restore times also been greatly reduced. During the proposed compression there is a multi-way trade-off involved between storage space (disk and buffer pool), I/O reduction (due to better memory caching), and performance (loss due to compression/decompression overhead, but gain due to lower I/O rate).

The most obvious advantage of database compression is that of reducing the storage requirement of information. Reducing the storage requirement of databases is equivalent to increasing the

capacity of the storage medium. Testing includes source file size, compression file size, compression time and decompression time.

Compression Types	Compressed File Size (MB)	Compression Time (mm:ss)	Decompression Time (mm:ss)
Uncompressed	1024	2.35	1.22
RLE Compression	580	1.40	1.06
Dictionary Compression	525	1.27	0.58
ILC Compression	370	1.04	0.39

Table 2. Performance Testing of Data Compression

The above table described 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. We experiment the data base size ranging from relatively simple to fairly complex. ILC compression algorithm can satisfy real-time performance demand of database. In this test we compared with existing algorithms and ILC algorithm. As shown in table 1 and 2, the new compression algorithm is superior in all respects to the general compression algorithm in the real-time database:

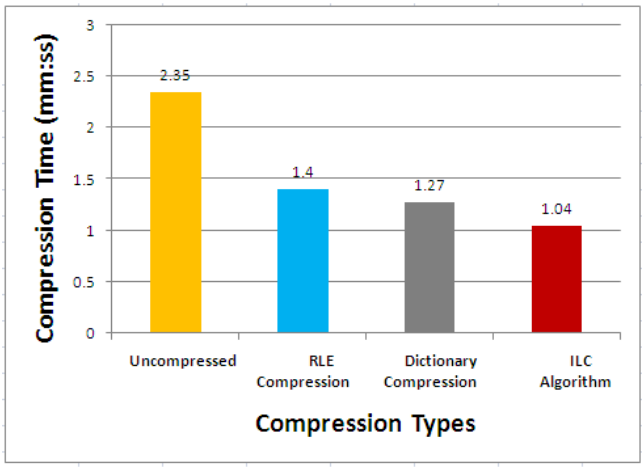


Fig.4. Compression Types vs Space Savings

1. The compression rate is nearly doubled than before. Previous data compression rate is about 20% ~ 30%, and now can reach 60% ~ 70%.
2. The compression time is greatly lower than before. The existing compression algorithm is adopted in the time code labels and the quality code of new compression algorithm, so the compression time is greatly reduced.
3. The decompression time is also cut down. So it is better to meet the requirements of real-time database.

6. CONCLUSION

Proposed compression approach will allow efficient retrieval of the compressed data, as well as produce a saving in storage costs. Experimental results have shown that the proposed backup compression process for real-time database systems using ILC algorithm are efficient in terms of

storage space, compression ratio and backup processing compared to an existing compression algorithms. This paper introduces a compression algorithm used in real-time database, which designs suitable compress method for every kind of historical data after synthetically considering compression radio, space and speed. We can see from the result of performance test that the compression algorithm has high compression performance and possesses great application value.

Proposed compression also improves CPU performance by allowing database operators to operate directly on compressed data and provides good compression of large databases. Importantly, we show that the compression effectiveness of our approach is excellent and markedly better than the existing algorithms on our test results.

REFERENCES

[1] Sushila Aghav, "Database compression techniques for performance optimization", proceedings in 2nd international conference on Computer 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] Wenjun Huang, Weimin Wang, Hui Xu, "A Lossless Data Compression Algorithm for Real-time Database", Sixth world congress, Intelligent Control and Automation, 2006.

[5] W. P. Cockshott, D. McGregor, N. Kotsis, J. Wilson "Data Compression in Database Systems", proceedings in 9th international workshop on Database and Expert Systems Applications, 1998.

[6] Chenggang Zhen , Baoqiang Ren, "Design and realization of data compression in Real-time database", proceedings in 10th international conference on Computational Intelligence and Software Engineering , 2009.

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

[8] 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.

[9] Ioannis Kontoyiannis 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.

[10] Computer Society Press, Los Alamitos, California. A. Cannane and H.Williams. General-purpose compression for efficient retrieval. Technical Report TR-99-6, Department of Computer Science, RMIT, Melbourne, Australia, 1999.