ISSN: 2321-7782 (Online)
Volume 1, Issue 4, September 2013
International Journal of Advance Research in Computer Science and Management Studies
Research Paper
Available online at: www.ijarcsms.com

Efficient Indexing Technique Using Bitmap Indices for Data Warehouses

Nikhil Dasharath Karande
Assistant Professor
Department of Computer Science and Engineering
Bharati Vidyapeeth's College of Engineering
Kolhapur – India

Abstract: The bitmap index technology is efficient for query processing in data warehousing applications. This paper focuses on efficient bitmap compression algorithm and examines the space and time complexity of the compressed bitmap index on large data sets from real applications. According to the conventional wisdom, bitmap indices are only efficient for low-cardinality attributes. However, the results show that the compressed bitmap indices are also efficient for high-cardinality attributes. Timing results demonstrate that the bitmap indices significantly outperform the projection index, which is often considered to be the most efficient access method for multi-dimensional queries. The bitmap index technology currently supported by commonly used commercial database systems and finally, discusses open issues for future research and development.

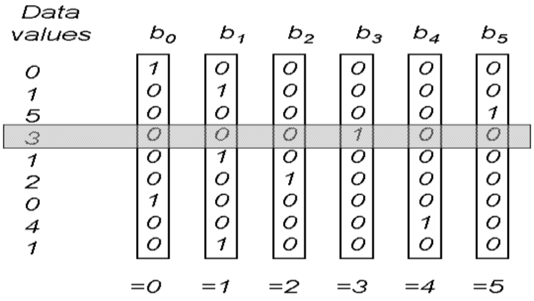
Keywords: Bitmap Index, Data Warehousing, Compression Algorithm, Projection Index, Multidimensional Queries.

I. Introduction

The most common task in data warehousing applications is to be queried large data sets to locate some selected records. To answer these queries efficiently is often difficult due to the complex nature of both the data and the queries. This paper focuses the most straightforward way of evaluating a query is to sequentially scan all data records to determine whether each record satisfies the specified conditions. A typical query condition is as follows "Count the number of cars sold by producer P in the time interval T". This search procedure could usually be accelerated by indices, such as variations of B-Trees or kd-Trees (Comer, 1979; Gaede & Guenther, 1998). Generally, as the number of attributes in a data set increases, the number of possible indexes combinations increases as well. To answer multi-dimensional queries efficiently, one faces a difficult choice. One possibility is to construct a separate index for each combination of attributes, which requires an impractical amount of space. Another possibility is to choose one of the multi-dimensional indices, which is only efficient for some of the queries. This paper focuses an indexing technology that holds a great promise in breaking the curse of dimensionality for data warehousing applications, namely the bitmap index. A very noticeable character of a bitmap index is that its primary solution to a query is a bitmap. One way to break the curse of dimensionality is to build a bitmap index for each attribute of the data set. To resolve a query involving conditions on multiple attributes, we first resolve the conditions on each attribute using the corresponding bitmap index, and obtain a solution for each condition as a bitmap. We then obtain the answer to the overall query by combining these bitmaps. Because the operations on bitmaps are well supported by computer hardware, the bitmaps can be combined easily and efficiently. Overall, result except the total query response time to scale linearly in the number of attributes involved in the query, rather than exponentially in the number of dimensions (attributes) of the data set, thus breaking the curse of dimensionality.

II. Basic Bitmap Index

The bitmap indices are one of the most efficient indexing methods available for speeding up multidimensional range queries for read-only or read mostly data (O'Neil, 1987; Rotem et al., 2005b; Wu et al., 2006). The queries are evaluated with bitwise logical operations that are well supported by computer hardware. For an attribute with c distinct values, the basic bitmap index generates c bitmaps with N bits each, where N is the number of records (rows) in the data set.



Data values	b_0	b_1	b_2	b_3	b_4	b_5
0	1	0	0	0	0	0
1	0	1	0	0	0	0
5	0	0	0	0	0	1
3	0	0	0	1	0	0
1	0	1	0	0	0	0
2	0	0	1	0	0	0
0	1	0	0	0	0	0
4	0	0	0	0	1	0
1	0	1	0	0	0	0
	=0	=1	=2	=3	=4	=5

Fig. 1 Simple bitmap index with 6 bitmaps to represent 6 distinct attribute values.

Each bit in a bitmap is set to "1" if the attribute in the record is of a specific value; otherwise the bit is set to "0". Fig. 1 shows a simple bitmap index with 6 bitmaps. Each bitmap represents a distinct attribute value. For instance, the attribute value 3 is highlighted to demonstrate the encoding. In this case, bitmap 3 is set to "1"; all other bits on the same horizontal position are set to "0".

A. *Encoding*

The basic bitmap index introduced above is also called equality-encoded bitmap index since each bitmap indicates whether or not an attribute value equals to the key. This strategy is the most efficient for equality queries such as "temperature = 100." Chan and Ioannidis (1998; 1999) developed two other encoding strategies that are called range encoding and interval encoding. These bitmap indices are optimized for one-sided and two-sided range queries, respectively. An example of a one-sided range query is "pressure < 56.7". A two-sided range query, for instance, is "35.8 < pressure < 56.7".

B. *Binning*

The strategy called binning is to reduce the number bitmaps. Since the encoding methods described before only take certain integer values as input, it may also view binning as a way to produce these integer values for the encoding Strategies. The basic idea of binning is to build a bitmap for a bin rather than each distinct attribute value. This strategy disassociates the number of bitmaps from the attribute cardinality and allows one to build a bitmap index of a prescribed size, no matter how large the attribute cardinality is. A clear advantage of this approach is that it allows one to control the index size. However, it also introduces some uncertainty in the answers if one only uses the index. To generate precise answers, one may need to examine the original data records (candidates) to verify that the user specified conditions are satisfied. The process of reading the base data to verify the query conditions is called candidate check (Stockinger et al., 2004; Rotem et al., 2005b).

C. *Compression*

Compression is the third strategy to reduce the size of bitmap indices. Since each bitmap of the bitmap index may be used separately from others, the compression is typically applied on each individual bitmap. Compression is a well-researched topic and efficient compression software packages are widely available. Even though these general-purpose compression methods are effective in reducing the size of bitmaps, query-processing operations on compressed bitmaps are often slower than on uncompressed bitmaps (Johnson, 1999). This motivated a number of researchers to improve the efficiency of compressed bitmap indices. Two of the most notable compression methods are Byte-aligned Bitmap Code (BBC) (Antoshenkov, 1994; Antoshenkov, 1996) and Word-Aligned Hybrid (WAH) code (Wu et al., 2004; Wu et al., 2006). Bitmaps compressed with BBC are slightly larger in size than those compressed with the best available general-purpose compression methods. However, operations on BBC compressed bitmaps are usually faster (Johnson, 1999). Clearly, there is a worthwhile space-time trade-off. The WAH compression takes this space-time trade-off one step further. More specifically, WAH compressed bitmaps are larger than BBC compressed ones, but operations on WAH compressed bitmaps are much faster than on BBC compressed ones. Therefore, WAH compressed bitmap indices can answer queries much faster as demonstrated in a number of different experiments (Stockinger et al. 2002; Wu et al., 2006).

III. Space Complexity – Size of Compressed Bitmap Indices

The space complexity of uncompressed bitmap indices was studied in (Chan & Ioannidis, 1998 and 1999). This paper discusses, analyze the size of compressed bitmap indices. The paper mainly focuses on the WAH compression method since BBC compression was extensively studied in (Johnson, 1999).

A. *Index Size of Real Application Data Sets*

Here now, analyze experimentally the size of compressed bitmap indices for various application data sets.

I. ***High Energy Data Set:*** Here, for an experiment we considered data set is from a high-energy physics experiment at the Stanford Linear Accelerator Center. It consists of 7.6 million records with 10 attributes. Figure 7 shows the size of the compressed bitmap indices. We notice that the size of the range-encoded bitmap index with 100 bins is about twice as large as the base data. The equality-encoded bitmap index with 1000 bins is about 30% smaller than the base data. Typically, the records of these high-energy physics experiments are not correlated with each other. Thus, it is generally hard for the run-length encoding to be effective. This is why the index sizes for range encoding are relatively large compared with the previous data sets. However, equality encoding compresses very well for this physics data set. Overall, it is observed that the actual bitmap index sizes are considerably smaller than the base data sizes and less than the sizes of typical commercial implementations of B-trees, that are often three to four times the size of the base data.

IV. Time Complexity – Query response Time

The experiment is on the two basic encoding methods, namely equality encoding and range encoding. We have chosen these two encoding methods for the following reason. Equality encoding showed to be the most space efficient method. Range encoding, on the other hand, is the most time efficient method for one-sided range queries (Chan & Ioannidis, 1998) that we use in our experiments. Analyses have shown that the worst case query response time to answer a one dimensional range query using a WAH compressed basic bitmap index (equality encoded without binning) is a linear function of the number hits (Wu et al., 2006). The analyses also indicate that the worst-case behavior is for attributes following a uniform random distribution.

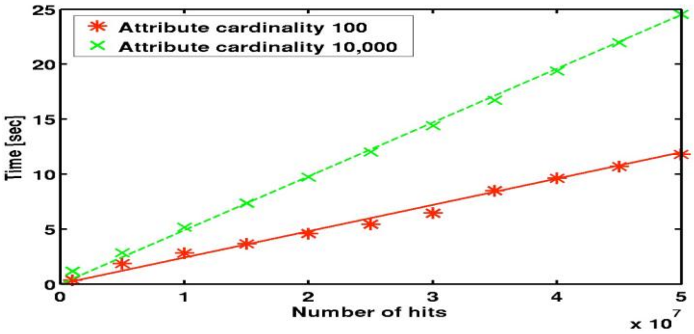


Fig. 2 Time (in seconds) to answer a one dimensional range query using WAH compressed bitmap index is a linear function of the number of hits.

The Fig. 2 plots the query response time against the number of hits for a set of queries on two attributes with different attribute cardinalities. The data values for the two attributes are randomly distributed in the range of [0; 100] and [0; 10,000] respectively. We see that in both cases the timing measurements follow straight lines, which is theoretically optimal.

V. CONCLUSION

Here, we conclude that a number of recent developments in the area of bitmap indexing technology. We organized much of the research work under the three orthogonal categories of encoding, compression and binning. We also provided a brief overview of commercial bitmap index implementations by major vendors. Most of the indexing methods reviewed were designed to efficiently answer multi-dimensional range queries. However, they are also efficient for other types of queries, such as joins on foreign keys and computations of aggregates (O'Neil & Quass, 1997). Despite the success of bitmap indices, there are a number of important questions that remain to be addressed. For example, is there an efficient bitmap index for similarity queries? How to automatically select the best combination of encoding, compression and binning techniques? How to use bitmap indices to answer more general join queries? Research work on bitmap indices so far has concentrated on answering queries efficiently, but has often neglected the issue of updating the indices. Clearly, there is a need to update the indices as new records are added. Efficient solutions to this issue could be the key to gain a wider adaptation of bitmap indices in commercial applications.

References

1. Chan, C.-Y., & Ioannidis, Y.E. (1998). Bitmap Index Design and Evaluation. SIGMOD, Seattle, Washington, USA, ACM Press.
2. Chan, C.-Y., & Ioannidis, Y.E. (1999). An Efficient Bitmap Encoding Scheme for Selection Queries, SIGMOD Conference, Philadelphia, Pennsylvania, USA, ACM Press.
3. Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record, 26 (1), 65-74.
4. Comer, D. (1979). The ubiquitous B-Tree. Computing Surveys, 11 (2), 121-137.
5. Gaede, V & Guenther, O. (1998) Multidimensional Access Methods. ACM Computing Surveys, 30 (2), 170—231.
6. Johnson, T. (1999). Performance Measurements of Compressed Bitmap Indices. International Conference on Very Large Data Bases (VLDB), Edinburgh, Scotland. Morgan Kaufmann.
7. Keim, D., & Hinneburg, A. (1999). Optimal Grid-Clustering: Towards Breaking the Curse of Dimensionality in High-Dimensional Clustering. International Conference on Very Large Data Bases (VLDB), San Francisco. Morgan Kaufmann.
8. Kiyoki, Y., & Tanaka, K., & Aiso, H., & Kamibayashi, N. (1981). Design and Evaluation of a Relational Data Base Machine Employing Advanced Data Structures and Algorithms. Symposium on Computer Architecture, Los Alamitos, CA, USA. IEEE Computer Society Press.
9. O'Neil, P., & Quass, D. (1997). Improved Query Performance with Variant Indexes. International Conference on Management of Data (SIGMOD 1997), Tucson, Arizona, USA. ACM Press.
10. Rotem, D. & Stockinger, K. & Wu, K. (2005b) Optimizing Candidate Check Costs for Bitmap Indices, Conference on Information and Knowledge Management (CIKM), Bremen, Germany, November 2005, ACM Press.
11. Stockinger, K., & Wu, K., & Shoshani, A. (2002). Strategies for Processing ad hoc Queries on Large Data Sets. International Workshop on Data Warehousing and OLAP (DOLAP), McLean, Virginia, USA.
12. Stockinger, K., & Shalf, J., & Bethel, W., & Wu, K. (2005) DEX: Increasing the Capability of Scientific Data Analysis Pipelines by Using Efficient Bitmap Indices to Accelerate Scientific Visualization, International Conference on Scientific and Statistical Database Management (SSDBM), Santa Barbara, California, USA, June 2005, IEEE Computer Society Press.
13. Wu, K., & Otoo, E.J., & Shoshani, A. (2002). Compressing Bitmap Indexes for Faster Search Operations. International Conference on Scientific and Statistical Database Management (SSDBM), Edinburgh, Scotland, UK, IEEE Computer Society Press.
14. Wu, K., & Otoo, E.J., & Shoshani, A. (2004). On the Performance of Bitmap Indices for High Cardinality Attributes. International Conference on Very Large Data Bases (VLDB), Toronto, Canada. Morgan Kaufmann.
15. Wu, K., & Otoo, E., & Shoshani, A. (2006). An Efficient Compression Scheme for Bitmap Indices. Technical Report LBNL-49626. To appear in ACM Transactions on Database Systems (TODS).
16. Wu, M.-C., & Buchmann, A.P. (1998). Encoded Bitmap Indexing for Data Warehouses. International Conference on Data Engineering (ICDE), Orlando, Florida, USA. IEEE Computer Society Press.

AUTHOR PROFILE

Mr. N. D. Karande received the B.E. Degree in Computer Science and Engineering from Bharati Vidyapeeth College of Engineering, Kolhapur, India in 2006. He received the M.Tech degree in Computer Science and Technology from Shivaji University, Kolhapur, India in 2010. From 2008 to till date, he is working as Assistant Professor at Bharati Vidyapeeth College of Engineering, Kolhapur, India. He has published various papers in the area of Database, Security and Natural Language Processing.