International Journal of Information Technology and Knowledge Management (ISSN: 0973-4414)
July-December 2012, Volume 5, No. 2, pp. 506-511

Bitmap Indexing Technique for Data warehousing and Data mining

Naveen Garg[1], Dr. H.M. Rai[2]
Ph.D. Student, NIMS University, Jaipur [1],
Professor, N.C. College of Engineering, Panipat [2]
er.gargnaveen@gmail.com, hmrai1943@gmail.com

Abstract: Sophisticated analytical tools, techniques and increase in processing power have resulted in development of new areas of database management as data warehousing and data mining. On line analytical processing (OLAP), data warehousing and data mining provide this functionality. This research paper focuses on automating aspects of the warehouse and omits the requirement of significant manual intervention such as the data acquisition, data quality management, functionality and performance optimization. Application of active database functionality into datawarehouses has been given considerable attention. Existing indexing methodologies have also been discussed and a suitable algorithm in the name of 'Bit map indexing' has been developed. The suitablity of the algorithm has been examined and analyzed on the basis of sets of observations characterstic curves and suitable inferences have been drawn.
Keywords: OLTP, OLAP, DSS.

1. Introduction

Growth of humanity is based on the growth of knowledge which itself is based on growth of knowledge bases. Knowledge bases are derived from databases using knowledge of its experts. Knowledge, such extracted helps in the development of experts through proper decisions and in turn results in humanity. Hence, for development of knowledge bases it is essential to identify, analyze and stipulate data elements and simulate process using them. Actual data/information available in real world are non linear in nature and analog in characteristics. Depending on their analogous nature and types of their characteristics they are further classified as visual, audio or text data etc. When they are kept in the actual form in storage systems they are called analog data bases. For quick retrieval and processing they must be converted into digital equivalents.

The analog data when converted to digital equivalent possess only the approximate replica as the conversion itself is based on the successive approximation methodology and the Niquist sampling rate [1] [3].

Processing of digital data becomes faster, more accurate and gives better efficiency and that too at the minimum cost resulting in most effective methodology. Thus, it becomes essential to convert acquired actual data in analog form to digital equivalents which in turn requires large storage space. So, the structured storage methodology involves handling of large as well as very large amount of data. The storage space is almost directly proportional to the volume of data.

The proper methodology must be evolved to store and extract data effectively [4]. This has resulted in the concept of data bases and data storage systems [2] [7].

2. Data Indexing

With the emergence of powerful new indexing technologies, instant and ad-hoc queries and fast data analysis are possible using existing databases. Despite the good customer service and data analysis capabilities, many customer services and data warehousing applications lack good performance. To meet this challenge in business applications

such as customer services, e-commerce etc., data warehouses must deliver data quickly through user friendly methods [5].

The main purpose of data is to access and use it. Quick access of information requires storage of data in structured form. The storage of data in structured form helps develop efficient and faster search technique to handle more complex queries and retrieve data with maximum precision.

The evolution of storage and access technique starts with the evolution of flat files. This was suitable, when files are small. As flat files require sequential scan of all the records in the file, the data access / retrieval time increases with the increase in the volume of data and thus results in more costly processing.

3. B-Tree Indexes

Like the white pages of a telephone book, which lists people names in alphabetical order, B-tree allows users to perform partial key lookups and view the data in sorted order.

B-trees offered a vast improvement over hashed keys in terms of flexibilty and were a great boon to OLTP systems because they do not require unique and arbitrary key values.

However, B-tree indexes are limited in that they are case sensitive and require a left to right match between the criteria entered and the values in the index. The first revolution in end user data access came in the 1980s with the development of 4GLs tools. Fourth Generation Languages have made it possible to develop new applications in a fraction of time required by conventional programming techniques, enabling milions of users to be brought online.

But it was still not enough. Organizations could not catch hold of the return on investment they had expected from their investments, in 4GL technology due to following reasons [6].

- While the number of users and frequency of data access continued to expand, the database kept growing larger as well.
- Although new applications allowed users access to corporate data, it was in a rigid, predefined manner that protected system resources.
- Users who were not comfortable with a character based application environment or who dit not take time to learn the cryptic commands and data layouts were still dependent on the I.T. department for their data access needs.
- To get information that went beyond the pre-established reports could take weeks by the time I.T. scheduled time to write a new report.
- Individual access was limited to one system at a time and access to multiple data sources on various hosts from the same terminal was virtually impossible.

4. Bit Mapped Indexing

Another type of advanced indexing technique is a bitmap or bitmapped indexing. Bitmap indexes provide high speed index-only query processing for instant counts, keyword searches and multicolumn combinations using multiple criteria without concatenating the columns into multipart keys.

The structure of a bitmapped index resembles a spreadsheet. The possible values go across the top, the record numbers down one side, and a flag or a ‘bit’ is set to ON or OFF in each cell, depending for a Y/N flag might resemble the table 1.

Initially, bitmaps were limited to low cardinality columns or coded data with few values such as 'Y/N' or 0 to 1 because they grew unmanageably large for high cardinality columns with many possible values especially with large amount of data. The early bitmapped indexes could not efficiently handle high cardinality data such as textual name and descriptive fields or numeric data with many values because the bit map that must be created and maintained becomes enormous.

Table 1. Bit-Map Table

Number	Yes	No
1		?
2	?	
3	?	
4		?
…… …… …… …… …… …… …… …… …… …… …… ……	…… …… …… …… …… …… …… …… …… …… …… ……	…… …… …… …… …… …… …… …… …… …… …… ……
n		

But, the present data warehousing solutions rely solely on bitmapped indexes as their indexing methodology due to its faster indexing rate. The performance impact of high cardinality data is achieved. Early concept of data being fairly static in nature and low maintainability has changed considerably.

4. Access Method Comparison

Advanced search techniques have lot of characteristics over the sequential scan and Relational (B-tree) key. A comparison is shown using some special characteristics of indexing methodology in table 4.

Table 4: Comparison of Accessing Methods

Characteristics	Sequential Scan	Relational (B-tree) Key	Inverted list Index
Key Word search	Yes	-	Yes
Partial Key Searches	Yes	-	Yes
Progressive searches (drill through)	-	-	Yes
Multiple key combination	-	Yes	Yes
Automated quantifying Count	-	-	Yes
Case Insensitive	-	-	Yes

Position insensitive	-	-	Yes
Pre-join indexes	-	-	Yes
Relational Logic	Yes	Yes	Yes
Boolean Logic	Yes	Yes	Yes
Soundex	-	-	Yes
Excluded word	-	-	Yes
Concatenated key	-	-	Yes
Composition Keys	-	-	Yes
Grouping Constants	-	-	Yes
Batch Indexing	-	-	Yes

5. Creation of Bit Map Indexing

Bit mapped indexes are useful in processing complex queries in decision support systems (DSS) and they have been implemented successfully in several commercial database systems. Major data structures used in this type of indexing is B-tree and B-tree string extension.

The indicated column of the data file is scanned to identify the unique value. These unique values are stored in the code array is used while populating the bitmaps. The index returned by the code array for a key is used to index into the bit table to locate the bitmap for the key.

A bit table is constructed to hold the bitmaps for each of the unique keys in the data file. The bitmap is a character array. This bit table is used to create the b-tree index. Each of the unique key values is picked up from the code array and the bitmap is picked from the bit table value encoding.

The simple algorithm used for creation of bitmap indexes and retrieval of data is shown in figure 1 and figure 2 in the form of flow charts.

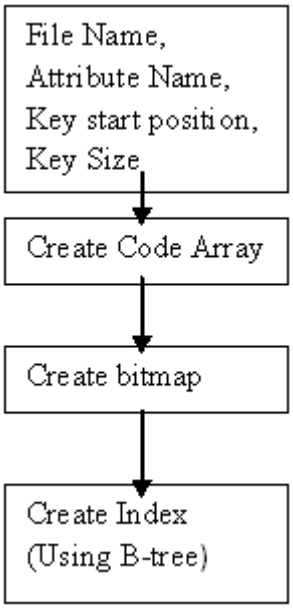

Fig 1: Flow Chart for Bitmap Index Creation

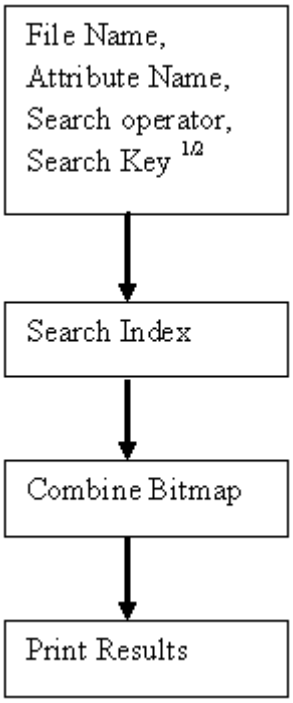

Fig. 2: Flow Chart for Retrieving Data

6. Conclusion

The bit map index algorithm developed is used and the time consumed in indexing at different cardinality is shown in the table 5.

Table 5: Observations

S. No.	Cardinality	Time in μs		
		1 col.	2 Col.	3 Col.
1	1	$2.4*10^{-5}$	$2.4*10^{-5}$	$5.8*10^{-5}$
2	2	$2.5*10^{-5}$	$2.5*10^{-5}$	$5.85*10^{-5}$
3	3	$2.5*10^{-5}$	$2.55*10^{-5}$	$5.9*10^{-5}$
4	4	$2.7*10^{-5}$	$2.56*10^{-5}$	$5.95*10^{-5}$
5	5	$2.75*10^{-5}$	$2.6*10^{-5}$	$6.05*10^{-5}$
6	6	$2.75*10^{-5}$	$2.65*10^{-5}$	$6.10*10^{-5}$
7	7	$3.1*10^{-5}$	$2.70*10^{-5}$	$6.20*10^{-5}$
8	8	$3.2*10^{-5}$	$2.75*10^{-5}$	$6.25*10^{-5}$
9	9	$3.2*10^{-5}$	$2.75*10^{-5}$	$6.28*10^{-5}$
10	10	$3.7*10^{-5}$	$2.80*10^{-5}$	$6.35*10^{-5}$
11	11	$3.75*10^{-5}$	$2.8*10^{-5}$	$6.4*10^{-5}$
12	12	$3.80*10^{-5}$	$2.85*10^{-5}$	$6.4*10^{-5}$
13	13	$4.0*10^{-5}$	$2.85*10^{-5}$	$6.51*10^{-5}$
14	14	$4.001*10^{-5}$	$2.86*10^{-5}$	$6.52*10^{-5}$

References

[1] K. Doris, E. Janssen, C. Nani and Athon Z., "A 480 mW 2.6 GS/s 10b Time-Interleaved ADC With 48.5 dB SNDR up to Nyquist in 65 nm CMOS" in IEEE Journal of Solid-state Circuits, vol. 46 No. 12, December 2011, pp. 2821-2833.

[2] You, J. Dillon, T. Liu, J., "An integration of data mining and data warehousing for hierarchical multimedia information retrieval", in International Symposium on Intelligent Multimedia, Video and Speech Processing, August 2002, pp. 373-376.

[3] Tomasic A., Garcia-Nolina, H., and Soen K., "Incremental Updates of Inverted List for Text Retrieval", proc. ACM SIGMOD cont on management of data, Minnapolis, pp 289-300.

[4] Lawyer, J.; Chowdhury, S.; Walter E., "Best practices in data warehousing to support business initiatives and needs ", 37th Annual Hawaii IEEE International Conference on System Sciences, Jan 2004.

[5] Jamil, S.; Ibrahim, R, "Performance analysis of indexing techniques in Data warehousing ", in IEEE International Conference on Emerging Technologies, Islamabad on Dec 2009.

[6] Graefe, G.; Kuno, H, "Modern B-tree techniques", in 27th IEEE International Conference on Data Engineering, Hannover, on May 2011.

[7] Chen, Q., Han, M. and Dayal. V., (2000). "A datawarehouse OLAP frame work for stable telecommunication tandem traffic analysis 2000" proceding Int. Conf. Data Engg. ICDE00, pp 201-210, San diego, CA Feb 2000.