An Efficient Approach for Data Indexing in Datawarehousing and Datamining

Naveen Garg
Ph D Scholar (Computer Science)
Sai Nath University, Jharkhand, India

Dr. H.M. Rai
Professor
N.C. College of Engineering, Panipat, India

Abstract - Today, many tools and techniques have increased in the performance of database management as data warehousing and data mining. These warehouses provide storage functionality and responsiveness to queries beyond capacity of transaction oriented database. This paper focuses on Indexing of the data warehouse and omits the requirement of significant manual intervention such as the data acquisition, data quality management, and functionality and performance optimization. Many approaches are used for data indexing such as sequential, B-tree key and some other advanced techniques. A comparison is done using some characteristics between theses indexing techniques. Based on the results of comparison, a Bitmapped indexing technique is created which increases the efficiency and reduces the time for retrieval of data.

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.

A study of actual forms of data has revealed that the data processing of actual or analog data is very difficult and complicated. This is a proven fact that the digital data processing is the only easiest, simplest, fastest, accurate most efficient type of processing. Fortunately, it is possible to convert all analog form of data into equivalent digital form, which are suitable to great extent for their development as knowledge bases.
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].

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].

1.1 Data Indexing

Once data has been cleaned properly they are to be stored in large storages such as functional data bases i.e. Data ware houses. Only to dump data in storage will again create jumbled type of data. In both cases faster search techniques need to be evolved for better query processing. To arrange data in a database in such a way that retrieval/accessing and updating becomes easier and faster, a process known as indexing comes in. An index access structure is similar to the index used in a text book which lists important terms at the end of the book in alphabetical order along with a list of address page numbers. In this case we use these addresses to locate a term in the text book by searching the specified pages. Alternatively, if no other guidance is given, the whole text book has to be searched word by word to find the term the user is interested in. This corresponds to linear search on a file. Of course, most books do have additional informatics such as chapter and section-titles that can help users to find a term without having to search through the whole book. However, the index is the only exact indication of where each term occurs in the book.

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.

II. 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. For example to find a record of ‘Chandragupta Maurya’, it is to be noticed that his name was entered as ‘Maurya Chandragupta’ including exact capitalization, punctuation and spacing. Most databases still utilize B-tree indexes, even in some RDBMs as their primary indexes.

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.

III. ADVANCE INDEXING TECHNIQUES

Although the indexing has been around since the early days of computers, there have been great advances in indexing technology over the years. Advanced indexing technology is the most effective way to reduce the disk I/O required to query, analyze, summarize and retrieve data. Following advanced indexes deliver dramatic performance improvements without major investments in hardware.

- Inverted List Indexes

- Bitmapped Indexes
- Aggregation Indexes

2.1. Inverted List Indexes

Inverted list indexes [3] provide much greater functionality and flexibility than B-tree indexes. Inverted list indexes reverse the structure with its pointers. They store the data from the database as keys, so the data content can be quickly searched on with pointers back to the database as data in the index, resulting in quick retrieval of data records.
Inverted list indexes are far superior to B-tree indexes. It can perform keyword lookups, provide an instants up-front qualifying count and support unlimited multicolumn and multidimensional queries. They enhance data access in both online (OLTP) and decision support (OLAP) environments.
Added advantage in this system is that both users and I.T. professionals benefit from the added functionality and enhanced performance gained users can intuitively search through data, finding records in a way that is obvious and logical.

2.2. Bit Mapped Indexing

Another type of advanced indexing technique is a bitmap or bitmapped indexing.

Table 1. Bit-Map Table

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

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.
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.

2.3. Aggregation Indexes

Data warehousing or Decision support applications contain millions of rows of data that users want to summarize for business intelligence. One method of summarizing information for data analysis is to perform a table scan of the famous personalities of Haryana in Sports and sort of data, but that can take enormous amount of time of CPU for each query even with parallel processing. For example some sorted source data for 'famous personalities of Haryanain Sports' for Sex, Category, Place, and year are shown in table 2.

Table 2: Aggregation Index

Sex	Category	Place	Year
F	Hockey	Rohtak	2007
F	Gymnastics	Sonipat	2004
M	Golf	Gurgaon	2010
M	Wrestling	Rohtak	2006
M	Vollyball	Karnal	2007

Based on this data, the numbers of possible lines in a report or bar chart is shown in table 3.

Table 3: Summary Indexing

Aggregate By	Number of lines in report
Sex	2
Category	5
Place	4
Year	5

Instead of sequentially reading the raw data the more common method of aggregating data is for the I.T. staff to pre build summary tables that contain the rolled up data aggregations that users want. Summary tables for predictable queries are fast at query time, but it still makes a full table scan of the large fact table to build one.
What makes it more complex is no matter how many summary tables IT builds, users always need to query in a new way given the inherent nature of data warehouse, which is to look for new information and patterns. Also there is no drill down to view the raw detailed data that makes up the summary table which is a table scan of the large related table. As an alternative, aggregation indexes can quickly summarize categories on the fly. They can dynamically calculate the number of wrestlers, in a particular period and a specific category instantly.
Aggregation indexes eliminate the needs for a summary table to match each possible user aggregation. Aggregation indexes allow the user complete flexibility in the selection criteria based on inverted list indexing then dynamically summarize the metric data at high rates of speeds.
More significant is the fact that, aggregation table gives instant access back to the detail data because they contain pointers (row ids) to the raw details data. After viewing summary one can instantly drill down and view the detail data.

IV. 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

V. 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.

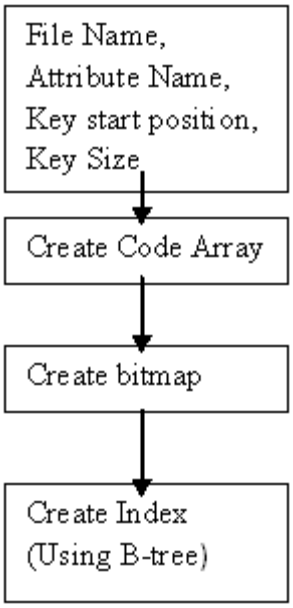

Fig 1: Flow Chart for Bitmap Index Creation

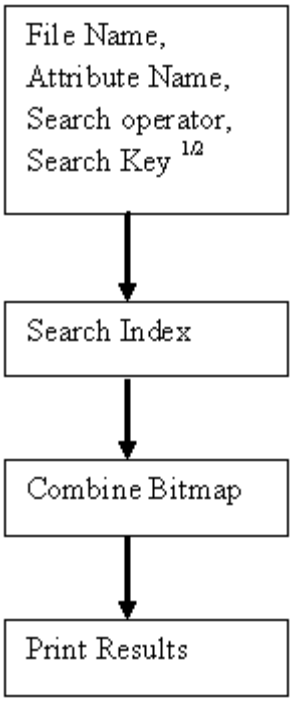

Fig. 2: Flow Chart for Retrieving Data

VI. 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.