SURVEY ON INDEXING TECHNIQUES OF DATA WAREHOUSE

[1]Nirmala Dhinwa
Asst. Professor
Dept. of MCA
Rajasthan Institute of Engineering and Technology,
Jaipur, India

[2]Bhawana Jarthalia
Asst. Professor
Dept. of MCA
Rajasthan Institute of Engineering and Technology,
Jaipur, India

Abstract :

Data warehouse system is becoming progressively important for decision-makers. Most of the queries against a large data warehouse are complex and iterative. The ability to answer these queries efficiently is a critical issue in the data warehouse environment. If the right index architecture are built on columns, the performance of queries, especially ad hoc queries will be greatly enhanced. In this paper, we provide a survey of various indexing techniques such as B-tree, Bit-mapped, Join and Projection index being studied/used in both academic research and industrial applications. With the beginning 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 task in business applications such as customer services, e-commerce etc., data warehouses must deliver data quickly through user friendly methods.

Keywords: Data warehouse, Bitmap index, B-tree index, encoded bitmap,

1. Introduction

A Data Warehouse (DW) is the base for Decision Support Systems (DSS) with a large collection of information that can be accessed through an On-line Analytical Processing (OLAP) application. This large database stores current and historical data that come from several external data sources.

When they are used in the real structure 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.

There are many ways to speed up query processing such as outline tables, indexes, parallel machines, etc. The performance when using summary tables for predetermined queries is good. However when an unpredicted query arises, the system must scan, fetch, and sort the actual data, resulting in performance degradation. Whenever the base table changes, the summary tables have to bare computed. Also building summary tables often supports only known frequent queries, and requires more time and more space than the original data. Because we cannot make all probable summary tables, choosing which ones to be built is a difficult job. Indexing is the key to achieve this objective without adding additional hardware. The objectives of this paper are to identify factors that need to be considered in order to select a proper indexing technique for data warehouse applications, and to assess indexing techniques being used in both academic research and business applications.

2. DATA WAREHOUSE PROCESS

A Data Warehouse is a integrated, time variant, nonvolatile, and subject-oriented collection of data in support of management's decision. A Data Warehouse system consists of a back-end database server, (an OLAP engine) and a front-end tool set. OLAP and Data Warehousing are usually used interchangeably. However, Data Warehousing includes jobs such as optimization, indexing, query processing and data extraction, and other services that a DBMS provides, while OLAP denotes the services, which support decision making, built on top of those services provided by a Data Warehouse. A logical structural design of a data warehouse system is depicted in Figure

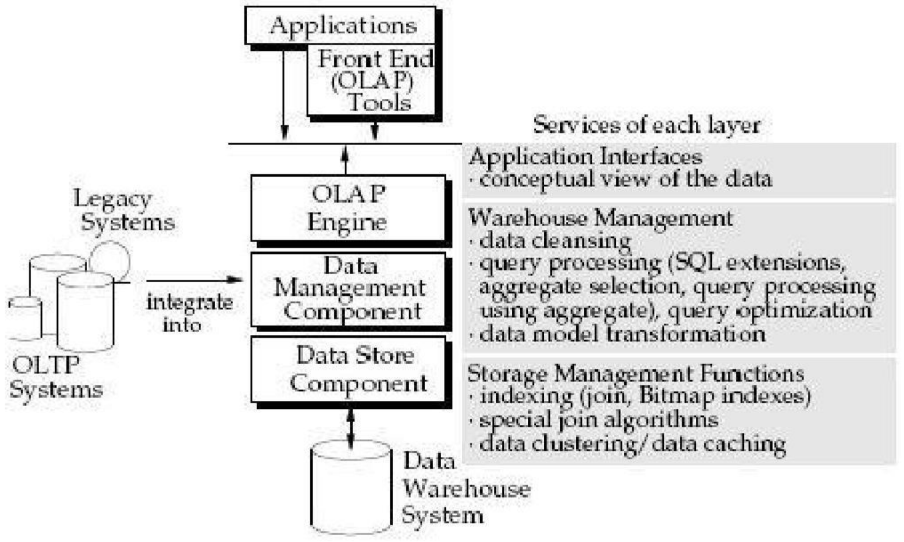

Applications
Front End (OLAP) Tools
Services of each layer
Application Interfaces
· conceptual view of the data
Legacy Systems
OLAP Engine
Warehouse Management
· data cleansing
· query processing (SQL extensions, aggregate selection, query processing using aggregate), query optimization
· data model transformation
Data Management Component
integrate into
OLTP Systems
Data Store Component
Storage Management Functions
· indexing (join, Bitmap indexes)
· special join algorithms
· data clustering/ data caching
Data Warehouse System

Fig. 1

3. Indexing Techniques in Data Warehouses

3.1 The B-Tree Index

The B-Tree Index is the default index for most relational database systems. The top mostLevel of the index is called the root. The lowest level is called the leaf node. All other levels in between are called branches. Both the root and branch contain entries that point to the next level in the index.Leaf nodes consisting of the index key and pointers pointing to the physical location (i.e., row ids) in which the matching records are stored. A B-Tree Index table is shown in Figure 2.

1) Root Node:
    It includes node pointers to its down branch nodes.
2) Branch Nodes:
    A branch node includes pointers to leaf nodes or the other branch nodes.
3) Leaf Nodes:
    It includes horizontal pointers and index items to the other leaf nodes

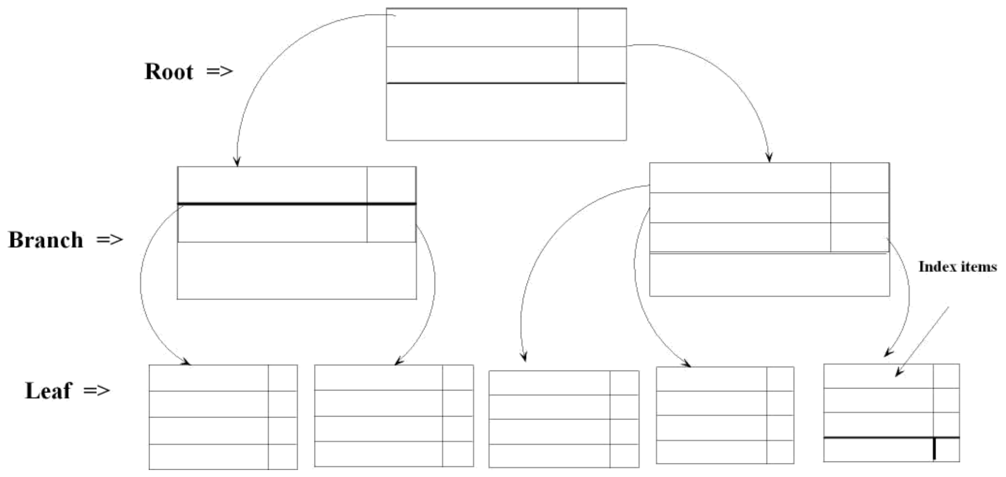


Fig. 2. B-Tree Index structure

Faculty (Table 1)

Faculty_id	City	Gender
F10	Kolkata	F
F11	Jaipur	F
F12	Jodhpur	F
F13	Kolkata	M
F14	Delhi	M
F15	Delhi	M
F16	Jaipur	F
F17	Kolkata	M
F18	Kolkata	F
F19	Jodhpur	F
F20	Jaipur	M
F21	Delhi	M
F22	Jaipur	F

Marks(Table 2)

Faculty_id	FDP Marks
F10	70
F11	50
F12	50
F13	40
F14	44
F15	30
F16	35
F17	20
F18	23
F19	24
F20	80
F21	90
F22	70

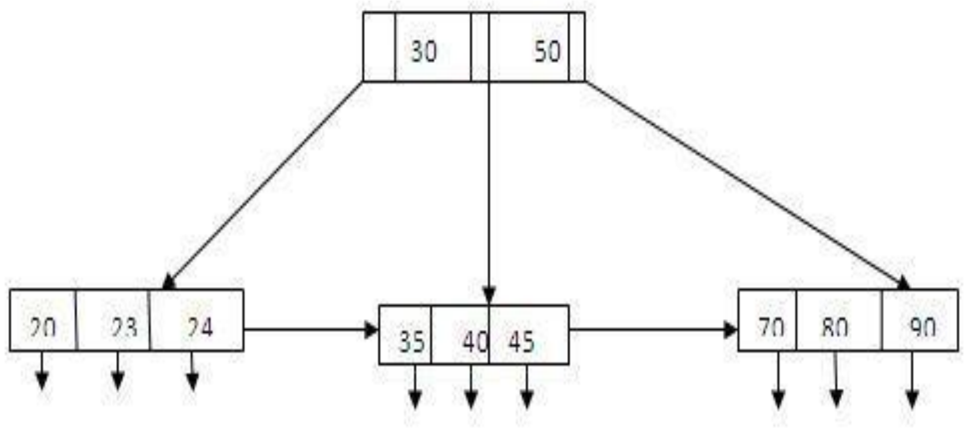


Fig. 3 The B-Tree Index on the marks column of the Faculty table

3.2 Projection Index

A Projection Index on an indexed column A in a table T stores all values of A in the same order as they come into view in T. Each row of the Projection Index saves one value of A. The row order of value x in the index is similar as the row order of value x in T. Normally, the queries against a data warehouse retrieve only a few of the table's columns, so having the Projection Index on these columns reduces tremendously the cost of querying because a single I/O operation may get more values into memory.

3.3 Bit Mapped 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 probable 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.

3.3.1 Simple bitmap index

The main motive of simple bitmap index is representing the Value of attribute with some string of bits (1 or 0). By doing so it help to indicate whether an attribute of a tuple is equal to a specific value or not. For example in an attribute GENDER there are two values, {FEMALE,MALE). Now for these values we can represent them by setting bits to each value i.e. for 'M' we set '1' when GENDER=M otherwise '0'. Similarly for "F" we can set '1' when GENDER=F otherwise '0'. Simple bitmap is easy to implement can query operation can be done easily but there lies a problem with it. When the value of attribute is large i.e. |A| = LARGE then setting bits to the corresponding value require huge space. To overcome is limitation ENCODED BITMAP is introduced.

3.3.2 Encoded Bitmap Index

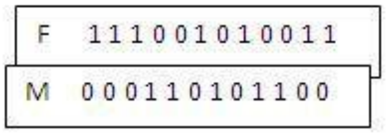
Suppose we have a table FACULTY with attribute DEPARTMENT which has 50 different values. Now if we want to build simple bitmap index on these values we have to set 50 bitmap vectors of N bits in length. So in encoded bitmap indexing instead of 50, we used [log250]=6 bitmap vectors plus a mapping table is used. For example let in table FACULTY the attribute DEPARTMENT there are three different values {CS, IT, MCA} so instead of using 3 bitmap vector we use only [log23] =2 bitmap vector.

Dept(Table 3)

Dept
CS
MCA
IT
IT
MCA
CS

3.4 Join Index

A Join Index is made by translating restrictions on the column value of a dimension table (i.e., the gender column) to limitations on a large fact table. The index is implemented using one of the two representations: row id or bitmap depending on the cardinality of the indexed column. A bitmap illustration, which is called Bitmap Join Index, is used with the low cardinality data although a row id representation is used with a high cardinality. In Data Warehouse, there are many join operations concerned; so building Join Indexes on the joining columns improves query processing time. Lets have an example, Bitmap Join Indexes on gender column in the MARKS table can be built by using the gender column in the FACULTY table and the foreign key student id in the marks table. Note that the marks table does not contain the gender column. The Bitmap Join Index for gender identical to male is created by setting a bit subsequent to a row for students id whose gender is ‘M’ to 1 in the marks Table. Otherwise, the bit is set to 0 as shown in Figure.



F	1 1 1 0 0 1 0 1 0 0 1 1
M	0 0 0 1 1 0 1 0 1 1 0 0

Refer Table2

4. CONCLUSION

The ability to extract data to respond complex, iterative, and ad hoc queries quickly is a critical issue for data warehouse applications. A good indexing technique is essential to avoid I/O intensive table scans against large data warehouse tables. The challenge is to find suitable index type that would progress the queries’ performance. B-Tree Indexes should only be used for high cardinality data and predicted queries. Bitmap Index plays a vital role in respondent queries of knowledge data warehouse as a result of they need a capability to realize operations on index level before retrieving base data. This accelerates question process hugely. Variants of ikon Indexes are introduced to scale back storage demand and speed up performance.

REFERENCES

[1] Anirban Bhattacharjee, Satyajit Ghosh, Rajdeep Chowdhury, “Comparative Study of Various Bitmap Indexing Techniques Used in Data Warehouse”, IJETTCS, Volume 1, Issue 3, September – October 2012

[2] C. DELLAQUILA and E. LEFONS and F. TANGORRA, Design and Implementation of a National Data Warehouse. Proceedings of the 5thWSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data
Bases, Madrid, Spain, February 15-17, 2006 pp. 342-347

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

[4] Jamil, S., Ibrahim, R, “Performance analysis of indexing techniques in Data warehousing ”, in IEEE International Conference on Emerging Technologies, Islamabad on_Dec 2009.

[5] Morteza Zaker, Somnuk Phon-Amnuaisuk, Su-Cheng Haw "An Adequate Design for Large Data Warehouse Systems: Bitmap index versus B-tree index" INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS , Issue 2, Volume 2, 2008

[6] Naveen Garg, Dr. H.M. Rai "Bitmap Indexing Technique for Data warehousing and Data Mining,Jaipur International journal of information technology and knowledge management (ISSN:0973-4414) July December 2012, Volume 5, No. 2, pp. 506-511.

[7] R. Kimball, L. Reeves, M. Ross, The Data Warehouse Toolkit. John Wiley Sons, NEW YORK, 2nd edition, 2002

[8]Sirirut Vanichayobon Le Gruenwald "Indexing Techniques for Data Warehouses' Queries" Norman, OK, 73019

[9] W. Inmon, Building the Data Warehouse., John Wiley Sons, fourth edition, 2005