ISSN: 2277–3061 (online)

International Journal of Computers & Technology
Volume 2 No.2 April 2012

EVALUATION OF BITMAP INDEX USING PROTOTYPE DATA WAREHOUSE

Murtadha M. Hamad
College of Computer,
University of Al-Anbar, Iraq

Muhammed Abdul-Raheem
College of Computer,
University of Al Anbar, Iraq

ABSTRACT

Bitmap indices have become popular access methods for data warehouse applications and decision support systems with large amounts of read-mostly data. This paper could arrive a number of results such as ; Bitmap Index highly improves the performance of Query Answering in Data Warehouses, It highly increases the efficiency of Complex Query processing through using bitwise operations (AND, OR). A prototype of Data Warehouse "STUDENTS DW" has been built according to the conditions of W. Inomn of Data Warehouses. This prototype is built for student's information.

Keywords

Bitmap, Indexing, Data Warehouse, Query processing.

1. INTRODUCTION

Data Warehouse is a collection of integrated, subject-oriented databases designed to support the DSS function, where each unit of data is relevant to some moment in time. The data warehouse contains atomic data and lightly summarized data [1]. DW contains historical data and it is loaded with new data on a periodic basis of time. [2]

In a study entitled "Bitmap Indices for Data Warehouses", the researchers discuss various bitmap index technologies for efficient query processing in data warehousing applications. They review the existing literature and organize the technologies into three categories; bitmap encoding, compression and binning are produced [3].

Also ,in another study entitled "An Efficient Bitmap Indexing Strategy based on Word-Aligned Hybrid for Data Warehouses ", presents an efficient bitmap indexing technique that improves the performance of the Word-Aligned Hybrid (WAH) for high-cardinality attributes in data warehousing applications. This study concludes that data reordering presents important advantages for improving the query performance in data warehouses[4]. By reviewing the above literature, we conclude the following points concerning Bitmap Index:

Bitmap index structure is used effectively in DWs due to low cost of both construction and maintenance.

Bitmap index structure increases the Query Answering efficiency by minimizing the query response time.

2. BITMAP INDEX CONCEPT

The bitmap representation is an alternate method of the row ids representation. It is simple to represent, and uses less space- and CPU become efficient than row ids when the number of distinct values of the indexed column is low.

The Bitmap Indexes improve complex query performance by applying low-cost Boolean operations such as OR, AND, and NOT in the selection predicate on multiple indexes at one time to reduce search space before going to the primary source data. Many variations of the Bitmap Index (Pure Bitmap Index, Encoded Bitmap, etc.) have been introduced, aiming to reduce space requirement and improve query performance[5].

2.1 Bitmap Index Properties:

There are many properties in Bitmap index structure, here we concentrate on the main properties of this structure.

2.1.1 Read-Only attributes

An attribute of a stable values, which are not changed or modified once they are stored in the data warehouse, and just read-operation can be done to it (in DML no deletion or updating operations are allowed). This means that values are fixed and the new values are appended to the current values. In data warehouses ,the read-only attribute concept is companion with the historical data concept. The read-only attribute is popular in a wide variety of applications, from forecasting applications to financial and scientific applications. Similarly , read-mostly attribute has such values that are not frequently changed or updated but at long time intervals.

The read-only/mostly attribute is a key factor in choosing a column to be indexed, as well as updating a value in an indexed column which means to rebuild the entire index, which is a time and space consuming operation, and causes inefficiency and degradation of the performance of the DBMS[6].

2.1.2 Low Cardinality Columns

One of the most required statistics is column cardinality. Column cardinality is the number of distinct values stored for a column or an attribute.

The column (attribute) is said to be of low cardinality if the distinct values reach 1% or more of the total number of table tuples. Gender is a classical example of low cardinality attribute since the attribute 'gender' has only two distinct values, 'male' or 'female', which means that the gender column is either 'male' or 'female' allover the table[7].

2.2 The Basics Bitmap Index

The basic idea behind bitmap indexes is to use a string of binary numbers (0 or 1) to indicate whether the indexed attribute in a table is equal to a specific value or not . That is, bitmap index is set to '1' if the specific value exists in the indexed column and '0' otherwise. For example if the indexed attribute is 'Gender'; there are only two values 'male' and 'female'; bitmap index on Gender column has only two bitmaps (bit vector), one bitmap for 'male' and another for 'female', denoted by Bm, Bf respectively. Bm is set to 1 if 'Gender' = 'male', otherwise Bm is set to 0, Similarly Bf is set to 1 if 'Gender'= female', otherwise Bf is set to 0, Fig (1) shows a simple bitmap index[8].

B^{Gender}

..	Gender	…
-	male	-
-	female	-
-	male	-
-	female	-
-	male	-
-	female	-

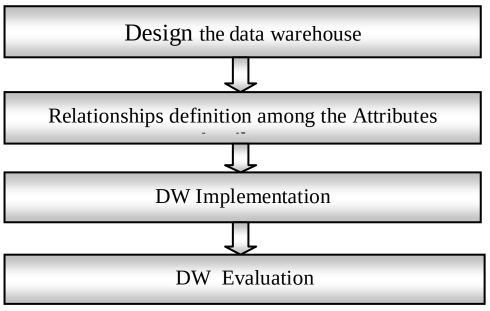
Bm	Bf
1	0
0	1
1	0
0	1
1	0
0	1

Fig (1) simple bitmap index

3. BITMAP INDEX DESIGN

3.1 General framework for Bitmap Index:

In this paper , we will evaluate bitmap index in Data Warehouses, and examine the effect of Bitmap Index on Query Processing, and what efficiency of Query Answering that produced by Bitmap Index. The flowchart that describes the overall work is as follows:


Design the data warehouse
Relationships definition among the Attributes
DW Implementation
DW Evaluation

Fig(2) Flowchart of STUDENT DW with Bitmap Index

3.2 The STUDENTS DW Design

In this paragraph, we will show two aspects ; the basic file structure and the relationships among data. After that, we apply the conditions mentioned in William Inomn 's definition of data warehouse.

3.2.1 File Structure of STUDENTS DW:

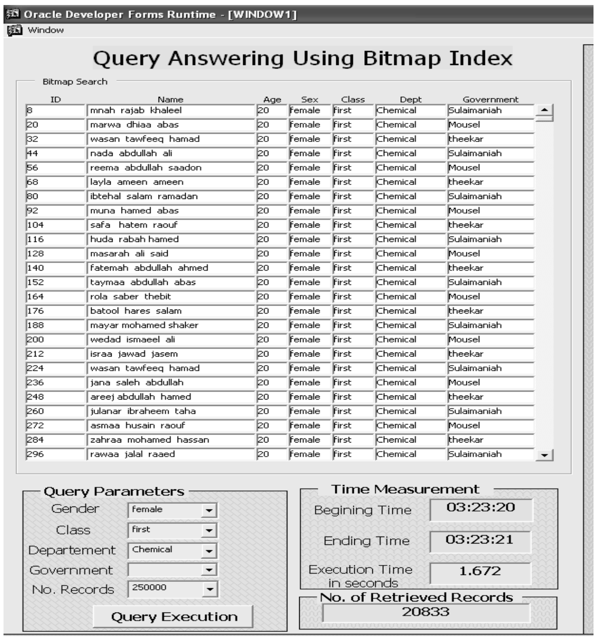
We start our DW by creating the tables according to our prototype "STUDENTS", which contains a main table named "STUD", composed of seven columns, which are (ID, NAME, AGE, SEX, CLASS, DEPT, GOVERN) , and creates bitmap index on four selected columns , which are (SEX, CLASS, DEPT, GOVERN) that have cardinality of 2, 4, 6, 18, respectively.

3.2.2 Network Architecture:

DW typically has client-server architecture, this means that the actual databases are resident at the server and all data warehousing operations are applied in server, while the client contains only access application, accepts user's queries, and sends them to DW server, receives the answers of the queries, and finally displays the results to user.

4. STUDENTS DW IMPLEMENTATION

An Oracle Environment was selected as a suitable solution to support this application with many tools such as Sql plus and Developer and others. To connect the Oracle client application to Oracle DW, Oracle has a classical long process mechanism. This is resulted in just updating a specific file at specific location in the client application with information from analogous file on Oracle server, so that developers are convened on just replacing the file in client with file in the server. When we wish to execute query, we determine the options of search, which are the complex "Where clause" conditions are equivalent in SQL, and every option is represented by combo-list; choice is done by clicking combo-list by left mouse button on a value of the combo-list. After choosing all search options, we click "Do query" button, and the result will be as in Fig (3):



Query Answering Using Bitmap Index

Bitmap Search

ID	Name	Age	Sex	Class	Dept	Government
8	mnah rajab khaleel	20	female	first	Chemical	Sulaimaniah
20	marwa dhiaa abas	20	female	first	Chemical	Mousel
32	wasan tawfeeq hamad	20	female	first	Chemical	theekar
44	nada abdullah ali	20	female	first	Chemical	Sulaimaniah
56	reema abdullah saadon	20	female	first	Chemical	Mousel
68	layla ameen ameen	20	female	first	Chemical	theekar
80	ibtehal salam ramadan	20	female	first	Chemical	Sulaimaniah
92	muna hamed abas	20	female	first	Chemical	Mousel
104	safa hatem raouf	20	female	first	Chemical	theekar
116	huda rabah hamed	20	female	first	Chemical	Sulaimaniah
128	masarah ali said	20	female	first	Chemical	Mousel
140	fatemah abdullah ahmed	20	female	first	Chemical	theekar
152	taymaa abdullah abas	20	female	first	Chemical	Sulaimaniah
164	rola saber thebit	20	female	first	Chemical	Mousel
176	batool hares salam	20	female	first	Chemical	theekar
188	mayar mohamed shaker	20	female	first	Chemical	Sulaimaniah
200	wedad ismaeel ali	20	female	first	Chemical	Mousel
212	israa jawad jasem	20	female	first	Chemical	theekar
224	wasan tawfeeq hamad	20	female	first	Chemical	Sulaimaniah
236	jana saleh abdullah	20	female	first	Chemical	Mousel
248	areej abdullah hamed	20	female	first	Chemical	theekar
260	julanar ibraheem taha	20	female	first	Chemical	Sulaimaniah
272	asmaa husain raouf	20	female	first	Chemical	Mousel
284	zahraa mohamed hassan	20	female	first	Chemical	theekar
296	rawaa jalal raaed	20	female	first	Chemical	Sulaimaniah

Query Parameters

Gender	female
Class	first
Departement	Chemical
Government	
No. Records	250000

Time Measurement

Begining Time	03:23:20
Ending Time	03:23:21
Execution Time in seconds	1.672

No. of Retrieved Records: 20833

Fig (3) Result of Query

4.1 Evaluation Steps

We perform our experiments to test the response time of bitmap index. The experiments applied on the project are of four steps as follows:

- **Step One:** searching the table using four conditions (gender, class, and department, govern), with AND bitwise operations. A search is done on four groups of records, which are 250000, 500000, 750000, 1000000, respectively.
- **Step Two:** is to search the table using three conditions (gender, class, and department), with AND bitwise operations. A search is done on four groups of records, which are 250000, 500000, 750000, 1000000, respectively.
- **Step Three:** is to search the table using two conditions (gender, class), with AND bitwise operation. A search is

done on four groups of records, which are 250000, 500000, 750000, 1000000, respectively.

- **Step Four:** searching the table using one condition (gender). Search is done on four groups of records, which are 250000, 500000, 750000, 1000000, respectively.

The cardinality of Gender is 2, while the cardinality of Class is 4, and the cardinality of Department is 6. At each step, we perform Four queries, four queries on bitmap index and four on b-tree index. Of course, both of the two sets of queries have been done as one query for each subset of records, which are 250000, 500000, 750000, 1000000 records. Response time for each query was registered and documented.

4.2 Results of Evaluation :

In this paragraph , we will demonstrate the results produced by experiments sessions of the four steps, as follows:

At step one: we applied the procedure of querying (described earlier) on this query:

```
SELECT
ID,NAME,AGE,SEX,CLASS,DEPT,GOVERNMENT
,ID_NO

FROM STUD_PU_BIT WHERE SEX = 'FEMALE' AND
CLASS = 'FIRST' AND DEPT = 'CHEMICAL' AND
GOVERN = ‘ANBAR’;
```

Three conditions were involved (sex, class, department), besides two bitwise operations (3 AND`s). Four completed queries have been done, and response time for each query was registered as shown in table 1.

Table 1 Response Time of Query Answering In Step One

No. Records	Bitmap Time in sec	No. of Retrieved Record
250000	2.718	6945
500000	6.281	13889
750000	6.531	20834
1000000	7.515	27778

At step two; we applied the procedure of querying (described earlier) on this query:

```
SELECT ID,NAME,AGE,SEX,CLASS,DEPT,ID_NO

FROM STUD_PU_BIT WHERE SEX = 'FEMALE' AND
CLASS = 'FIRST' AND DEPT = 'CHEMICAL' ;
```

Three conditions were involved (sex, class, department), beside two bitwise operations (2 AND`s). Four completed queries have been done, and response time for each query was registered as in table 2.

Table 2 Response Time of Query Answering In Step Two

No. Records	Bitmap Time in sec	No. of Retrieved Record
250000	6.719	20834
500000	7.328	41667
750000	11.11	62500
1000000	13.5	83334

At step three, we applied our procedure of querying on this query:

```
SELECT ID,NAME,AGE,SEX,CLASS,DEPT,ID_NO

FROM STUD_PU_BIT WHERE SEX = 'FEMALE' AND
CLASS = 'FIRST';
```

Two conditions were involved (sex, class), in addition to one bitwise operation (AND). Four completed queries have been done, and response time for each query was registered as in table 3.

Table 3 Response Time of Query Answering In Step One

No. Records	Bitmap Time in sec	No. of Retrieved Record
250000	6.719	20834
500000	7.328	41667
750000	11.11	62500
1000000	13.5	83334

At step four, we applied our procedure of querying on this query:

```
SELECT ID,NAME,AGE,SEX,CLASS,DEPT,ID_NO

FROM STUD_PU_BIT WHERE SEX = 'FEMALE';
```

Only one condition was considered, it was (sex), and no bitwise operations where involved . We did only six queries; we had not accomplished query on 1000000 records because of the limits of our machine. Response time for each query was registered.

4.3 Cardinality of STUDENTS DW:

In this section, the term "Cardinality" will be explained and practically implemented. As previously mentioned, cardinality is the number of distinct values in the column. We design a tool in our project that calculates the cardinality of each column.

First we create a function in oracle database and name it “Cardinality”, the duty of this function is to calculate the cardinality according to the given column name, the function is resident in the oracle database .

Algorithm to calculate the cardinality

Input: single column appears in 'where' clause.

Output: number of the distinct value in the column (i.e. cardinality).

Step1: the user enters the column name at client machine.

Step2: client sends column name to server and waits for answer.

Step3: function 'cardinality' receives the column name, opens new session with STUDENTS DW database, calculates the number of the redundant data, and returns the result to the server.

Step4: client receives the result from the server and shows it to the user,which is the cardinality of the column.

Fig (4) shows the cardinality calculating algorithm in our project interface:

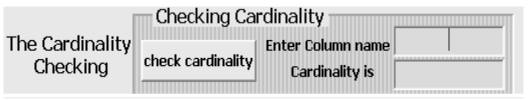
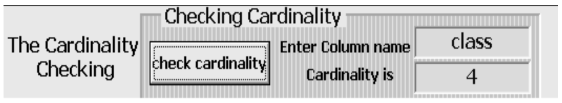


Fig (4) Cardinality Calculating Tool

Fig (5) is an example of the work of the algorithm, where it checks the cardinality of the column “class”:



Checking Cardinality

The Cardinality Checking

check cardinality

Enter Column name: class

Cardinality is: 4

Fig (5) Checking Cardinality of Column “Class”

The checking cardinality algorithm was applied to all DW table columns, and the results are listed in table 4 as follows:

Table 4 Cardinality of STUDENTS DW

No	Column name	Cardinality
1	Id	1000000
2	Name	43973
3	Sex	2
4	Class	4
5	Dept	6
6	Govern	18

5. RESULT DISCUSSION

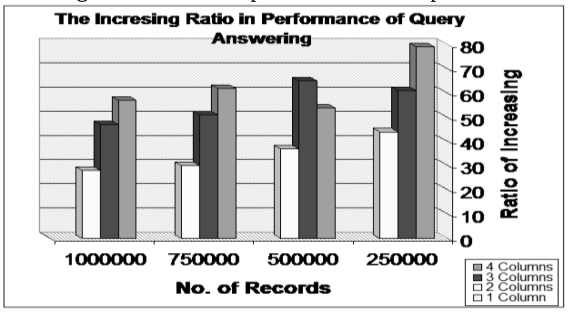
As a result, we make some notes depending on observations and induction of the results of query response times.

1.Bitmap index efficiency is increased with the increased No. of records.

2. Bitmap index is working more efficient with multiple columns and bitwise operation (AND), than in a single column.

3. Increasing the number of bitwise operations means increasing the query response time, which in turn reflects positively on the query process and improves query speed.

These conclusions were made according to the following bar-chart of improvements of response time, fig (6):



Fig(6) Increasing Ratios of Bitmap

6. QUERY OPTIMIZATION

Query optimization is the ultimate target of enhancing the performance of bitmap index. Space and time are the most important metrics to evaluate the performance of an index, and in case of Bitmap index; the focus was on time optimal index. Results obtained during the current work indicate the query optimization through the decreasing of query response time.

The optimization in query was due to the use of logical operations, efficient I/O operations, and memory and CPU efficient use in bitmap index. So it was said that bitmap index is a time optimal index.

7. CONCLUSIONS

The study concludes the following points:

1. Bitmap Index highly improves the performance of Query Answering in Data Warehouses.
2. Bitmap Index highly increases the efficiency of Complex Query processing through the use of bitwise operations (AND, OR).
3. The performance of bitmap index is increased with the increment in number of bitwise operations.
4. Bitmap Index improves I/O efficiency and CPU performance due to direct mapping of zeros and ones to the logic gates of CPU’s and I/O operations.

8. REFERENCES

[1] W. H. Inmon," Building the Data Warehouse" , 3rd edition, 2002, John Wiley& Sons, Inc,USA.

[2] Joseph M. Hellerstein, Michael Stonebraker, and James Hamilton, "Architecture of a Database System", 2007, Now Publishers Inc, USA.

[3] Kurt Stockinger, “Bitmap Indices for Data Warehouses”, University of California, 2007.

[4] Ali Hamadou , Kehua Yang, “An Efficient Bitmap Indexing Strategy based on Word-Aligned Hybrid for Data Warehouses“, Hunan University, China, 2008.

[5] Guadalupe Canahuate, Tan Apaydin,"Secondary Bitmap Indexes with Vertical and Horizontal Partitioning" , ACM. EDBT 2009, March 24–26, 2009, Saint Petersburg, Russia.

[6] Models David Marco, Michael Jennings, “Universal Meta Data”, Published by Wiley Publishing, Inc., Indianapolis, Indiana, USA, 2004.

[7] Rishi Rakesh Sinha ,Marianne Winslett, “Multi-resolution Bitmap Indexes for Scientific Data”, University of Illinois at Urbana-Champaign, 2007.

[8] Ming-Chuan Wu ,“Encoded Bitmap Indexes and Their Use for Data Warehouse Optimization”, doctorate dissertation, technical University of Darmstadt, 2001.