The bitmap index advantages on the data warehouses

El Amin Aoulad Abdelouarit,[a] Mohamed El Merouani[b], Abdellatif Medouri [c]

[a] UAE— Laboratory modeling and information theory Abdelmalek Essaadi University, Tétouan, Morocco
elamin@hotmail.es

[b] UAE— Laboratory modeling and information theory Abdelmalek Essaadi University, Tétouan, Morocco
m_merouani@yahoo.fr

[c] UAE— Laboratory modeling and information theory Abdelmalek Essaadi University, Tétouan, Morocco
amedouri@uae.ma

Abstract. The data warehouse designer should consider the effectiveness of the data query while the selection of relevant indexes and their combination with materialized views, this, because the data query is the only way to get this information from a data warehouse. The index selection is considered an NP-complete problem, because of the number of indexes is exponential in the total attributes in the data warehouse tables. So, the choose of the suitable index type is considered as the main step to start the data warehouse design. In this paper we are making a comparison study between the B-tree index as traditional index type, and the Bitmap index. this comparison are based on three factors : index size, clustering factor and compression, and we illustrate this with a real experiment.

Keywords: Business Intelligence, Data warehouse.

1 INTRODUCTION

Before the data warehouse implementation, some administration tasks that are taken by a data warehouse administrator needs to be decided, like logical and physical design, management of storage space and performance tuning.

The most important task is the physical design including data organization and access improvement. The fast access to this data needs general index to find the information wanted without reviewing all table data.

As per this need, the index selection task is considered difficult because their number is exponential in the total number of attributes in the database. So the index plays an important role in the data warehouses performance. For this reason, we focus on this data warehouse aspect, which is considered interesting for the designer while editing and query optimization selection.

The target is to minimize the query execution time, and as in a data warehouse the query uses indexes to access to the data, we will work on the problem of choosing the type of index when designing our data warehouse.

It's also important, that the index compression and size must be checked to optimize the storage usage.

There are several types of indexes supported by databases such as Bitmap, B-tree, Bitmap join, range-based bitmap index etc.. In this sense we have chosen two types of index relevant to this study, the index type: B-tree index and type Bitmap.

2 BITMAP INDEX

2.1 Definition

A bitmap index is a data structure defined in a DBMS used to optimize access to data in databases. It is a type of indexing is particularly interesting and effective in the context of selection queries. The index bitmap attribute is encoded in bits, where its low cost in terms of space occupied. All possible attribute values are considered, the value is present or not in the table. Each of these values is an array of bits, called bitmap, which contains as many bits as n-tuples present in the table. Thus, this type of index is very effective when the attributes have a low number of distinct values. Each bit represents the value of an attribute for a given tuple. For each bit, there is an encoding presence / absence (1/0), which indicates that a tuple or not the present value characterized in bitmap.

Table 1: Basic Bitmap

ROWID	C	B0	B1	B2	B3
0	2	0	0	1	0
1	1	0	1	0	0
2	3	0	0	0	1
3	0	1	0	0	0
4	3	0	0	0	1
5	1	0	1	0	0
6	0	1	0	0	0
7	0	1	0	0	0
8	2	0	0	1	0

To illustrate how a bitmap index works, we take an example EE-PP-O'Neil and O'Neil, Table 1 illustrates a basic bitmap index into a table containing 9 records, where the index is created in the C column with integers ranging from 0 to 3, we say that the cardinality of the column C is 4, by what there are 4 distinct values [0, 1, 2, 3], where the index bitmap C Contains 4 bitmaps shown as B0, B1, B2 and B3 corresponding value represents. In this example, the first line where RowID = 0, column C is worth 2, consequently, B2 column bit value "1", while the other bitmaps are set to "0". Same for the next line, where C = 1 corresponds to the bitmap B1 is set to 1 and the rest to "0". This process is repeated for the remaining lines.

2.1 Properties

Bitmap indexes have a very interesting property of responding to certain types of requests without returning the data themselves, thus optimizing the response time, disk storage. This is possible by counting operations (COUNT) and logical operators (AND, OR, etc.) that act "bit by bit" on bitmaps.

3 B-TREE INDEX

3.1 Definition

The index B-tree stores the index values and pointers to other index nodes using a recursive tree structure. The data are easily identified by traces pointers. The highest level of the index is called the root while the lowest level is called the leaf node or "leaf node". All other levels between them are called branches or internal nodes. All roots and branches contain entries that point to the next level of the index. Leaf nodes consist of the index key and pointers pointing to the physical location of records. We present details of the index B-tree structure.

The B-tree structure is used by the database server to configure the index (Figure 1)

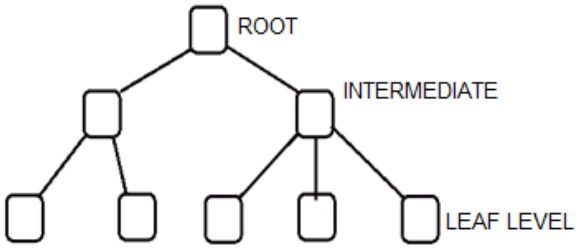

Fig 1: B-tree structure

Root or root is the highest level of the index points to the following levels of nodes branches.

Intermediate nodes or branches contain pointers to the following branches or to the leaf nodes level.

Node leaves or leaf nodes: the lowest level of the index points to other node leaves.

4 HYPOTHESIS

It's known that the bitmap index is more efficient than the B-tree one, and this by its low cardinality columns, in this paper we will focus the advantages given by bitmap index compared by B-tree one.
The factors that have been considered in this comparison are:

- The index size.
- Clustering factor.
- Compression.

4.1 Index size comparison

Step 1:

In our Data warehouse schema, we created a table named "Employees" with 100000 records and with a column named employee_id with 100000 distinct values and we added a GRADE column with 4 distinct values only.

Step 2:
We create now a standard B-Tree index on the GRADE column using this SQL :
SQL> create index employees_grade_i on employees(grade);
Then we check the index size using this query :
SQL> select index_name, index_type, distinct_keys, blevel, leaf_blocks from dba_indexes where index_name='EMPLOYEES_GRADE_I';

And we got this result:

Table 2: B-tree index size checking result in GRADE column

mINDEX_NAME	INDEX_TYPE	DISTINCT _KEYS	BLEVEL	LEAF_ BLOCKS
employees_grade_i	Normal	4	1	176

Step 3: Creating a bitmap index on the same column to compare the size (dropping the b-tree index created in first step)
SQL> create bitmap index employees_grade_bitmap_ii on employees(grade);
Step 4: In the bitmap index size checking, we use the same query:
SQL> select index_name, index_type, distinct_keys, blevel, leaf_blocks from dba_indexes where index_name='EMPLOYEES_GRADE_II';

Table 3: Bitmap index size checking result in GRADE column

INDEX_NAME	INDEX_TYPE	DISTINCT _KEYS	BLEVEL	LEAF_ BLOCKS
employees_grade_ii	Bitmap	4	1	10

Note that the index size is reduced from 176 to 10 (while going from B-tree to bitmap index)
Step 5: the bitmap index creation on employee_id column that contains 100000 distinct values:
SQL> create bitmap index employees_empid_bitmap_i on employees(employee_id).
By checking the index size using the same query, we have this table as result:

Table 4: Bitmap index size checking result in EMPLOYEE_ID column

INDEX_ NAME	INDEX _TYPE	DISTINCT _KEYS	BLEVEL	LEAF_ BLOCKS
employee s_empid_ bitmap_i	Bitmap	100000	1	348

And when trying with B-tree index we have this result:

Table 5: B-Tree index size checking result in EMPLOYEE_ID column

INDEX_NA ME	INDEX_ TYPE	DISTINCT _KEYS	BLEVEL	LEAF_ BLOCKS
employees_em pid_btree_i	B-tree	100000	1	222

4.2 Clustering factor checking

For large distinct values B-tree index occupies less size, and for minimal distinct values, the bitmap index occupies less size.
Clustering factor: considered as the sum of rows orders in a table based on the index values.

- If this amount is near the number of blocks, then the table order is well done, and the index entries in a single leaf block are pointing to rows stored in the same data blocks.
- If the value is near the number of rows, then the table is randomly ordered, so is improbable that the index entries in a single leaf block are pointing to rows stored in the same data blocks.

Table 6: Clustering factor and blocks used for B-Tree index on GRADE column

INDEX_NAME	CLUSTERING _FACTOR	BLOCKS
employees_grade _i	1148	256

Table 7: Clustering factor and blocks used for Bitmap index on GRADE column

INDEX_NAME	CLUSTERING _FACTOR	BLOCKS
employees_grade _ii	20	16

5.3 Compression advantage:

Is known that bitmap Indexes store a separate bitmap for every distinct value of a column, so it's like the storage space would become prohibitive for large tables with a lot of distinct values. As example, if a table contains 1 million rows with 10,000 distinct values, won't the Bitmap Index contain 1,000,000 x 10,000 = 10,000,000,000 bits near 1.2GB. A million-row table is considered small for a data warehouse, but 1.2GB is very big.
As Oracle uses some clever compression where long sequences of 1s or 0s in the bitmap consume hardly any space, storage space is not a significant consideration for bitmap indexes with a large number of distinct values.
With a lot of distinct values, instances of a particular value will be widely spread. This means long strings of zeros in the bitmap which become highly compressed.
The effect illustration is better shown with an example. First, create a table with:

- 1 million rows
- 2 identical columns with 10,000 distinct values
- A B-Tree index on one column
- A Bitmap Index on the other column
-

Table 8: Table comp_test creation query

```
SQL> CREATE TABLE comp_test
  2  AS
  3  SELECT mod(LEVEL, 10000) AS col1
  4  ,       mod(LEVEL, 10000) AS col2
  5  FROM   dual
  6  CONNECT BY LEVEL <= 1000000;
Table created.
SQL> CREATE INDEX COMP_TEST_col1_ix ON COMP_TEST(col1);
Index created.
SQL> CREATE BITMAP INDEX COMP_TEST_col2_bx ON
COMP_TEST(col2);
Index created.
```

Now, compare the size of each index.

Table 9: Index size comparison

```
SQL> SELECT segment_name, sum(bytes)/1024/1024
AS mb
  2  FROM   user_segments
  3  WHERE  segment_name LIKE 'COMP%'
  4  GROUP BY segment_name;

SEGMENT_NAME                          MB
------------------------------ --------
COMP_TEST                             15
COMP_TEST_COL2_BX                      4
COMP_TEST_COL1_IX                     17
```

Note that the Bitmap Index has been squished down to just 4MB – one-quarter the size of the B-Tree index! In fact, I ran this same test with 1 million distinct values (ie. One row per value) and still the bitmap index was only 50% larger than the B-Tree index.

4.3.1Results

The most difficult scenario of bitmaps compression si when it contains strings of zeros that are interrupted by one or two "ones" in each byte. It won't compress at all, but if every 8th row has the same value then there can only be 8 such uncompressible bitmaps for the column.
In other words:
There is many distinct values means each bitmap will be very compressed because they contain a lot of zeros.
The bitmaps are not very well compressed then there will not be many of them because there are only a few distinct values

5 CONCLUSION AND FUTUR WORK

A higher clustering factor has been caused by the optimizer while a full scan to a table using the B-tree index. In other side, using the bitmap index the clustering factor caused is too low compared with the first result. Here we deduct the performance by the sum of input/output required to fetch the result.
Both indexes have a similar target: the return of results as fast as possible. But the choice of which one to use should depend only on the type of application and data volume, and not on the level of cardinal.
In other side, the compression cannot be a factor that affects our choice of Bitmap index for data warehouse optimization strategy; this means that the bitmap has it supremacy on data warehouse performance
As future work, we will focus our studies on data warehouse schema comparison, especially it impact on data warehouse performance, and this to have a data warehouse physical design model for better decision system performance.

References

E. E-O'Neil and P. P-O'Neil (2000). Bitmap index design choices and their performance impli-cations *Database Engineering and Applications Symposium. IDEAS 2007. 11th International, pp. 72-84.*

R. Kimball, L. Reeves, M. Ross (2002) *The Data Warehouse Toolkit. John Wiley Sons, NEW YORK, 2nd edition, 2002*

W. Inmon (2005) *Building the Data Warehouse., John Wiley Sons, fourth edition, 2005*

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

R. Strohm, (2007) *Oracle Database Concepts 11g,Oracle*, Redwood City,CA 94065, 2007

C. Dell aquila and E. Lefons and F. Tangorra(2007) *Analytic Use of Bitmap Indices. Proceedings of the 6th WSEAS International Conference on Artificial Intelligence*, Knowledge Engineering and Data Bases, Corfu Island, Greece, February 16-19, 2007 pp. 159

K. Wu and P. Yu (1998) *Range-based bitmap Indexing for high cardinality attributes with skew*, In COMPSAC 98: Proceedings of the 22nd International Computer Software and Applications Conference. IEEE Computer Society, Washington, DC, USA, 1998, pp. 61-67.

EL AMIN A. Abdelouarit, M. El Merouani, A. Medouri (October 2013) *Data Warehouse Tuning: The Supremacy of Bitmap Index*, International Journal of Computer Applications, Volume 79 - Number 7, Published By FCS® (Foundation of Computer Science, USA),

EL AMIN A. Abdelouarit, M. El Merouani, A. Medouri (September 2013) *The impact of indexes on data warehouse performance*, International Journal of Computer Science Issues, Volume 10 – Issue 5, Number 2, Mahebourg, Republic of Mauritius.

EL AMIN A. Abdelouarit, M. El Merouani, A. Medouri (December 2012) *Optimisation des performances des entrepôts de données via les index*, Revue Ivoirienne des Sciences et Technologie, Numéro 20 décembre 2012, Abidjan (Côte d'Ivoire)

El Amin Aoulad Abdelouarit Is a Datacenter Manager and Ex Database administrator inTanger Med Port, Morocco, PhD Student doing research in Data warehouse and Data mining and it application in Port Management.

Mohamed El Merouani is professor of mathematics, with interests in Probability, Statistics, Stochastic operational research and Data mining, professor of Statistics and Computer Sciences, Poly disciplinary Faculty of Tétouan, Abdelmalek Essaâdi University, Morocco. University of Granada, Spain, Ph.D. in Mathematics, 1995.

Abdellatif Medouri is Full professor of physics, with interests in Telecommunications, Information theory and Databases; professor of Statistics and Computer Sciences, Poly disciplinary Faculty of Tétouan, Abdelmalek Essaâdi university, Morocco. University of Granada, Spain, Ph.D. in physics, 1995.