Understanding the Bit-Map

Applies to

SAP BIW 3.5, SAP NetWeaver 7.0. For more information, visit the Business Intelligence homepage.

Summary

This document is about the Bit-Map Indexing.

Author: Nilesh Ramesh Ahir

Company: IBM India

Created on: 03 September 2009

Author Bio

Nilesh Ahir has completed his masters in Software System from BITS Pilani. He has total 4 years of SAP experience. He has been working as SAP NW BI Application consultant for IBM India for last one year. Prior to this he was working with Intel India. He has experience in ABAP, BW3.5 / BI7.0 and Data mining. He has worked on other non–SAP technologies like TIBCO and web services.

Table of Contents

Introduction

There are different techniques used for improving the query performance. Some of them are partitioning and aggregation. Indexing is also one of the techniques used for improving the query performance in SAP BW/BI system. In indexing technique, we are trying to reduce number of data block access in main memory. There are different types of indexing techniques and these are as follows –

1. Primary indexing
2. Secondary indexing
3. Cluster indexing
4. Dense indexing
5. Non-dense indexing
6. B-tree indexing etc…

In this article we will be discussing an indexing technique called Bit-Map Indexing.

In SAP BW/BI system, reports are getting generated mainly from Info cubes that are storing data persistently. Every info cube has two fact tables F fact table and E fact table. For F fact table, SAP uses B tree indexing technique to improve the data loading performance. For E fact table, SAP uses Bit-Map indexing to improve the data retrieval performance during query evaluation. For line item dimensions, SAP uses B-Tree indexing.

Bit-Map Indexing

Bit-Map indexing is an innovative way of indexing where bit-maps are getting created and processor will perform the Bit wise operation to evaluate query results. This indexing technique is very efficient and used when cardinality is low

Cardinality is ratio of number of distinct values in column and total number of records in table.

Cardinality = (Number of distinct values in column) / (Total number of records in table)

Thus for a primary key of database table cardinality is 1.

Thumb Rule : Bit-Map Indexing should be used where Cardinality is less than 0.05

Building Bit-Map

The bitmap index stores the column values in bits i.e. 0 and 1. Zero represent false and 1 represent truth.

Lets take following example.

X	Y	Z
1	100	FOO
2	200	BOO
3	100	BOO
4	300	ZOO
5	200	FOO

In above data base table, X is the primary key and Y & Z are non-primary keys.

Therefore we will build Bit-Map Index for the columns Y & Z. To build the Bit-Map for Y, first identify and note down the distinct values of Y as follows

Y
100
200
300

Now ask yourself a question at every record whether the value is present in that record or not. If value is present then write the Bit value 1 i.e. true and if value is absent then 0 i.e. false.

Therefore Bit Map for Value 100 of column Y will be 10100. Similarly build the bit map for other values as well.

Bit-Map for Y

Y	Bit-Map
100	10100
200	01001
300	00010

Bit-Map for Z

Z	Bit-Map
FOO	10001
BOO	01100
ZOO	00010

Query Evaluation using Bit-Map Indexing

Thus in above section we have seen how system builds Bit-Map for various column. In this section we will see how system uses these Bit-Maps for evaluating the queries.

Query 1 : Show me all records where Y = 100 AND Z = BOO

To evaluate above query, take Bit-Map for Y=100 and Z=BOO and perform the bit wise AND operation

Bit-Map for Y = 100 is 1 0 1 0 0

Bit-Map for Z= BOO is 0 1 1 0 0

Therefore perform ANDing operation

1 0 1 0 0

AND

0 1 1 0 0

_____

Query Result 0 0 1 0 0

Thus from above Query result Bit-Map we can see that where condition is true only for 3rd record of the table where the Bit-Map value is 1.

Therefore result of above query is 3rd record of the table

X	Y	Z
3	100	BOO

Query 2: Show me all records where Y = 100 OR Z = BOO

To evaluate above query, take Bit-Map for Y=100 and Z=BOO and perform the bit wise OR operation

Bit-Map for Y = 100 is 1 0 1 0 0

Bit-Map for Z= BOO is 0 1 1 0 0

Therefore perform ORing operation

1 0 1 0 0

OR

0 1 1 0 0

_____

Query Result 1 1 1 0 0

Thus from above Query result Bit-Map we can see that where condition is true for 1st three records of the table where the Bit-Map value is 1.

Therefore result of above query is 1st three records of the table

X	Y	Z
1	100	FOO
2	200	BOO
3	100	BOO

Complex Query:

Employee data base table

Employee_number	Age	Salary (*1000)

1	25	60
2	45	60
3	50	75
4	50	100
5	50	120
6	70	110
7	85	140
8	30	260
9	25	400
10	45	350
11	50	275
12	60	260

Query: Find all employees of age in range 45 to 55 and drawing salary in ranging 100K to 200K

Bit Map for Age

Age	Bit-Map
25	100000001000
45	010000000100
50	001110000010
70	000001000000
85	000000100000
30	000000010000
60	000000000001

Bit-Map for Salary

Salary (K)	Bit Map
60	110000000000
75	001000000000
100	000100000000
120	000010000000
110	000001000000
140	000000100000
260	000000010001
400	000000001000
350	000000000100
275	000000000001

Now take the first part of condition i.e. Age is in range 45 to 55. Therefore take the bit map for 45 but there is no value 55 in table so take the bit map for values which is less than 55 and greater than 45. That value is 50 in our example.

Bit Map for 45 is 0 1 0 0 0 0 0 0 0 1 0 0

Bit Map for 50 is 0 0 1 1 0 0 0 0 0 0 1 0

Now in condition, we are talking about range therefore perform the OR operation

0 1 0 0 0 0 0 0 0 1 0 0

OR

0 0 1 1 1 0 0 0 0 0 1 0

_____

(Result1) 0 1 1 1 1 0 0 0 0 1 1 0

Now take the second part of condition i.e. Salary is in range 100K to 200K. Therefore take the bit map for 100 but there is no value 200 in table so take the bit map for all values which is less than 200 and greater than 100.

Bit Map

Salary (K)	Bit-map
100	000100000000
120	000010000000
110	000001000000
140	000000100000

Now perform the OR operation to get the result of second part of condition

0 0 0 1 0 0 0 0 0 0 0 0

OR

0 0 0 0 1 0 0 0 0 0 0 0

OR

0 0 0 0 0 1 0 0 0 0 0 0

OR

0 0 0 0 0 0 1 0 0 0 0 0

_____

(Result2) 0 0 0 1 1 1 1 0 0 0 0 0

Overall query result: age (45 to 55) AND salary (100K to 200K)

Therefore perform the AND operation for Result1 and Result2

Result1 0 1 1 1 1 0 0 0 0 1 1 0

AND

Result2 0 0 0 1 1 1 1 0 0 0 0 0

_____

Over all Query Result 0 0 0 1 1 0 0 0 0 0 0 0

Therefore from above bit map we can see that where clause holds good for only two records i.e. 4th and 5th

Thus query result will be

Employee_number	Age	Salary (*1000)
4	50	100
5	50	120

Advantages of Bit-Map Indexing

Access method specially efficient for low cardinality column of high-dimensional fact table

Takes a fraction of space as compared to B-tree

Very less index creation time

Very effective on queries with multiple-conditions in where clause

Related Content

SAP BI InfoCube Performance

Performance Tuning for SAP Business Information Warehouse

BW Performance Tuning

For more information, visit the Business Intelligence homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.