Chapter 3.19
Indexing in Data Warehouses:
Bitmaps and Beyond

Karen C. Davis
University of Cincinnati, USA

Ashima Gupta
University of Cincinnati, USA

ABSTRACT

Bitmap indexes (BIs) allow fast access to individual attribute values that are needed to answer a query by storing a bit for each distinct value and tuple. A BI is defined for a single attribute and the encodings are based solely on data values; the property map (PMap) is a multidimensional indexing technique that precomputes attribute expressions for each tuple and stores the results as bit strings. In order to determine whether the PMap is competitive with BIs, we conduct a performance study of the PMap with the range encoded bit sliced index (REBSI) using cost models to simulate storage and query processing costs for different kinds of query types. We identify parameters that have significant effect on index performance and determine situations in which either index is more suitable. These results could be useful for improving the performance of an analytical decision making system.

INTRODUCTION

A data warehouse is a repository of information collected from different sources. Querying of data warehouses for decision-making in areas such as sales and marketing planning is referred to as online analytical processing (OLAP). In the write-once-read-many environment of OLAP applications, multidimensional data analysis is now increasingly used for decision support systems (DSS). Complex DSS queries are often submitted interactively and reducing their response time is a critical issue in the data warehousing environment (Vanichayobon & Gruenwald, 1999; Jurgens & Lenz, 2001). Bitmap indexes are widely used for indexing warehouse data.

A bitmap index (BI) allows fast access to tuples based on values of attributes. Bitmap indexes consume only a fraction of the size of the indexed data and provide dramatic performance gains. Boolean operations such as AND, OR and NOT

are extremely fast for bitmap vectors, also called bitmaps or bit-vectors (O'Neil & Quass, 1997). Bitmaps indicate whether an attribute in a tuple is equal to, greater than or less than (depending upon the type of BI) a specific value or not. The length of a bit-vector is equal to the cardinality of the indexed table. The position of a bit in a bit-vector denotes the position of a tuple in the table. For example, a simple bitmap index (SBI) on an attribute status, with domain {backorder, shipped}, results in two bitmap vectors, say B_b and B_s. For B_b, the bit is set to 1 if the corresponding tuple has the value "backorder" for the attribute status, otherwise the bit is set to 0. Similarly for B_s, the bit is set to 1 if the associated tuple has the value "shipped" for the attribute status, otherwise the bit is set to 0. For another attribute, say product-category having values from 1-5, there is a bitmap vector corresponding to each of the five values, say B_1-B_5. Tuples that have product-category value as 1 have the bit corresponding to bit-vector B_1 set; the rest of the bits for that tuple are 0. Table 1 shows an SBI on status and product-category for 5 tuples with two bitmap vectors for status and five for product-category. These indexes can be interpreted as follows: tuple number 2 corresponds to a shipped order (B_s is set) with product-category 5 (B_5 is set). To illustrate query processing with an SBI, consider a simple SQL query that retrieves all tuples corresponding to shipped orders for product category 5:

SELECT * FROM Inventory WHERE status = "shipped" AND product-category = "5"

Table 1. Example of two simple bitmap indexes

	status		*product-category*				
Tuple	B_s	B_b	B_1	B_2	B_3	B_4	B_5
1	1	0	0	1	0	0	0
2	1	0	0	0	0	0	1
3	1	0	0	0	0	1	0
4	0	1	1	0	0	0	0
5	1	0	0	0	1	0	0

In order to evaluate this query using the example SBIs, a query optimizer takes the bitmaps for "status = shipped" and "product-category = 5" and performs a logical AND operation. Tuple 2 in Table 1 is the only tuple in the query answer.

We survey bitmap indexing techniques in the next section. Then we propose a novel multidimensional indexing technique that precomputes attribute expressions for data items and stores the results as bit strings. We study performance issues for this technique and a comparable bitmap index and recommend scenarios where one may be preferable to the other. We conclude with guidelines for improving query processing performance for complex range queries.

BACKGROUND

Bitmap indexes are designed for different query types including range, aggregation and join queries. Figure 1 shows tree diagrams of bitmap indexes, which we classify into three categories based on their main features. Figure 1(a) shows bitmap indexing methods that use the simple bitmap index (SBI) representation described in the previous section. The techniques that use clustering of attribute values are grouped in one category. The other category consists of techniques that are basically applications of SBI. Figure 1(b) shows encoded bitmap index (EBI) techniques that use binary encoding along with a mapping table and retrieval functions. Each attribute is encoded in such a way that the number of bitmap vectors retrieved to answer a query is reduced compared to the SBI. Bit-sliced index techniques are shown in Figure 1(c). They are based on the idea of attribute value decomposition, that is, decomposition of an attribute value in digits according to some base, either uniform or non-uniform. They can be either range or equality encoded.

BIs comparable to the novel technique introduced in the third section are discussed in further detail below. We select one technique for