Evaluation Strategies for Bitmap Indices with Binning*

Kurt Stockinger, Kesheng Wu, Arie Shoshani

Lawrence Berkeley National Laboratory
1 Cyclotron Road, Berkeley, CA 94720, USA
{KStockinger, KWu, AShoshani}@lbl.gov

Abstract. Bitmap indices are efficient data structures for querying read-only data with low attribute cardinalities. To improve the efficiency of the bitmap indices on attributes with high cardinalities, we present a new strategy to evaluate queries using bitmap indices. This work is motivated by a number of scientific data analysis applications where most attributes have cardinalities in the millions. On these attributes, binning is a common strategy to reduce the size of the bitmap index. In this article we analyze how binning affects the number of pages accessed during query processing, and propose an optimal way of using the bitmap indices to reduce the number of pages accessed. Compared with two basic strategies the new algorithm reduces the query response time by up to a factor of two. On a set of five dimensional queries on real application data, the bitmap indices are on average 10 times faster than the projection index.

1 Introduction

Large scale, high-dimensional data analysis requires specialized data structures to efficiently query the search space. Both commercial data warehouses and scientific data are typically read-only, and index data structures do not require transactional support for update operations. Under these conditions bitmap indices are suitable for complex, multi-dimensional data analyses.

The basic idea of bitmap indices is to store one slice of bits per distinct attribute value (e.g. all integers from 0 to 140). Each bit of the slice is mapped to a record or a data object. The associated bit is set if and only if the record's value fulfills the property in focus (e.g. the respective value of the record is equal to, say, 87). One of their main strengths is that complex logical selection

* The authors thank Ekow Otoo, Doron Rotem, and Heinz Stockinger for their constructive comments during the writing of this article. This work was supported by the Director, Office of Science, Office of Laboratory Policy and Infrastructure Management, of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy.

operations can be performed very quickly by means of Boolean operators such as AND, OR, or XOR.

The contributions of this article are as follows. We summarize the current state of the art of bitmap index technologies and focus in particular on queries against scientific data. Next we introduce a novel bitmap evaluation technique and compare it with currently deployed methods. We provide both analyses and experimental measurements to show that the new strategy indeed minimizes the number of records scanned. In some cases we observe a factor two improvement in query response time using the new strategy.

2 Related Work

Bitmap indices are mostly used for On-Line Analytical Processing (OLAP) and data warehouse applications [1] for complex queries in read-only or append-only environments. The most commonly used bitmap encoding strategies are *equality, range* or *interval* encoding [2,3]. Equality encoding is optimized for so-called exact match queries of the form $a = v$ where a is an attribute and v the value to be searched for. Range encoding, on the other hand, is optimized for one-sided range queries of the from a *op* v where *op* $\in \{<, \leq, >, \geq\}$. Finally, interval encoding shows the best performance characteristics for two sided-range queries of the form v_1 *op* a *op* v_2.

Traditional bitmap indices are typically used on integer and string values. However, scientific data is often based on floating point values which requires other kinds of bitmap indices based on binning [6,7]. In this case, one bitmap does not represent one attribute value but one attribute range (see Figure 1).

Assume that we want to evaluate the query $x < 63$. The bitmap that holds these values is bitmap 4 (shaded in Figure 1). This bitmap represents floating point numbers in the range of 0 to 80. In order to evaluate the query $x < 63$, two additional steps are required in order to retrieve the values that match the query condition.

Note that bitmap 4 represents values in the range of 0 to 80, which is more than what we have specified as our query condition (63). We now combine bitmap 4 and bitmap 3 with the logical operator XOR and get those values that are in the range of 60 to 80. As depicted in Figure 1, two values are left that need to be read from disk and checked against the query constraint $x < 63$. We call this additional step the *candidate check*.

There are a number of approaches to reduce the index size and increase the performance of the bitmap index for high cardinality attributes. These approaches include multicomponent encoding [2,3], binning the attribute values [6,7] and compressing the bitmaps [4,8].

3 Evaluation Strategies

The query example in Section 2 is a typical one-dimensional query since the query condition consists of only one attribute. For multi-dimensional queries that con-

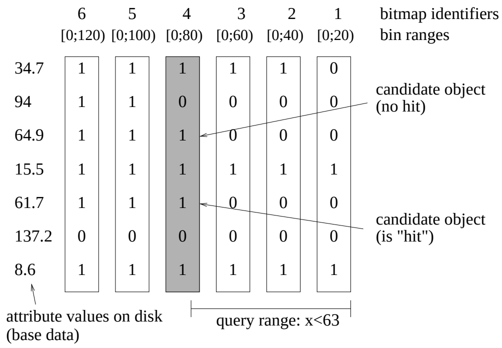

Fig. 1. One-sided range query $x < 63$ on a range encoded bitmap index.

tain several attributes, for instance $x_1 < 63$ AND $x_2 > 72$ AND $x_3 \leq 5.2$, the results of several bitmaps need to be combined. The goal is to calculate the intermediate result of each query dimension in such a way that the number of candidates and thus the number of disk scans is minimized. In this section we present three different techniques for evaluating bitmap indices with binning.

Assumptions and Definitions: The following analysis assumes all attributes have uniform distribution. This represents the worst case for the bitmap indices, which are usually more efficient on real application data as demonstrated in Section 5. Without loss of generality, we further assume the domain of each attribute is normalized to be [0, 1]. We limit all queries to be conjunctive with a one-sided range condition on each attribute $x_i < v_i$. The bin boundaries just below and above v_i are denoted by $\underline{v}_i$ and $\overline{v}_i$ respectively. For the attribute x in the domain of [0, 140], the query $x < 63$ shown in Figure 1 is normalized to be $x < 0.45$. The lower and upper ranges of the candidate bitmap are 60 and 80. After normalization, we yield $\underline{v}$=0.43 and $\overline{v}$= 0.57.

Strategy 1: Figures 2a) - g) show a graphical interpretation of the bitmap evaluation strategy on a 2-dimensional query. In the first phase, the bitmap index is scanned for both attributes. The result is an L-shape which represents the candidate records (see Figure 2) of both dimensions 1 and 2. We refer to these candidates as C_{tot}. Since the domains of the attributes are normalized, the number of records in C_{tot} is equal to the area of the L-shape times the total number of records N.

Let us assume the hit area for attribute i is denoted as H_i and the candidates for attribute i are denoted as C_i. We can calculate the candidate L-shape C_{tot} as follows: $C_{tot} = (H_1 \vee C_1) \wedge (H_2 \vee C_2) - H$, where $H = H_1 \wedge H_2$. This is equivalent to what is shown graphically in Figure 2.

In the next phase, the candidate check for attribute 1 is performed by reading the attribute values from disk and checking them against the range condition. All

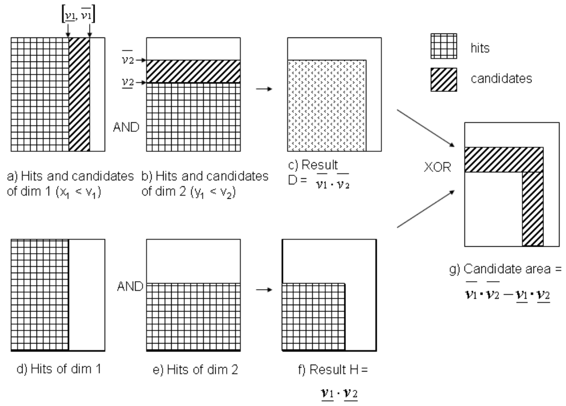

Fig. 2. Calculation of the candidate area for strategies 1 and 3.

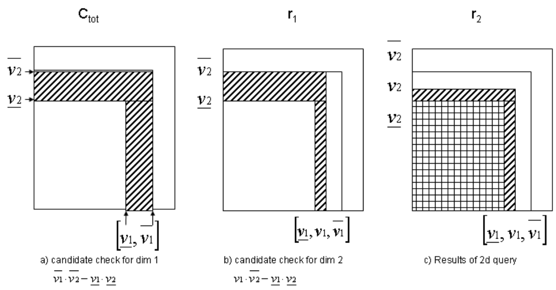

Fig. 3. Bitmap evaluation strategy 1.

records represented by C_{tot} are checked (see Figure 3a)). The records satisfying the range condition involving attribute 1 is recorded as r_1. Finally, the candidate check for dimension 2 is performed by reading all attributes represented by the area r_1 (see Figure 3b)). The results of the 2-dimensional query are shown in Figure 3c).

Let the candidate selectivity of ith dimension s_i be the fraction of records that need to be scanned, the candidate selectivity of the first dimension $s_1 = \overline{v}_1\overline{v}_2 - \underline{v}_1\underline{v}_2$, and the candidate selectivity of the second dimension $s_2 = v_1\overline{v}_2 - \underline{v}_1\underline{v}_2$. In general, the equation for the candidate selectivity is:

$$s_i = \left(\prod_{j=1}^{i-1} v_j \prod_{j=i}^{d} \overline{v}_j \right) - \prod_{j=1}^{d} \underline{v}_j \qquad (1)$$

where d refers to dimension. The total number of records scanned $S = N\sum_{i=1}^{d} s_i$.

Ideally, we would only read the candidate records during the candidate checking. Since most I/O system performs disk operations in pages, more records are actually read into memory. To more accurately evaluate the cost of candidate checking, we compute the number of page accesses for each attribute. Given the candidate selectivity s, the estimated number of pages is $(1 - e^{-\frac{sN}{p}})p$, where N is the number of records of the base data and p is the number of pages for one attribute [5]. The total number of pages for all dimensions is $\sum_{i=1}^{d}(1 - e^{-\frac{s_i N}{p}})p$. For the next two bitmap evaluation strategies, the number of pages is estimated analogically.

Strategy 2: This strategy evaluates each dimension separately. Using the same 2D example as before, the bitmap index for the first attribute is evaluated first and the candidates of this attribute are checked immediately afterward (see Figure 4a)). After these operations the condition involving the first attribute is fully resolved. In Figure 4 the result is denoted as H_1. Similarly, dimension 2 is evaluated. Since the query conditions are conjunctive, the final answer must be from H_1. Therefore, the candidates to be checked must be both in H_1 and C_2. This reduces the area from $\overline{v}_2 - \underline{v}_2$ to $v_1(\overline{v}_2 - \underline{v}_2)$.

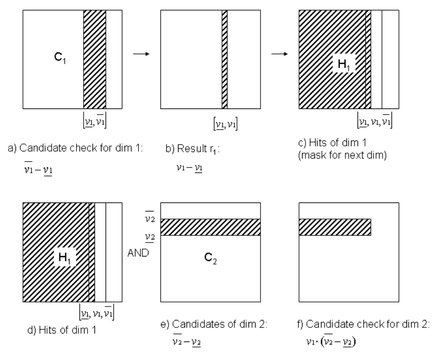

Fig. 4. Bitmap evaluation strategy 2.

It is straightforward to extend this strategy to resolve more attributes. The candidate selectivity of attribute i is as follows:

$$s_i = \left(\prod_{j=1}^{i-1} v_j\right)(\overline{v}_i - \underline{v}_i) \tag{2}$$

Strategy 3: This strategy is an optimal combination of Strategies 1 and 2. Given the values are binned, it checks the minimal number of candidates.

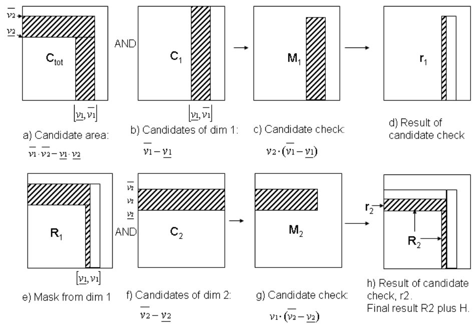

Fig. 5. Bitmap evaluation strategy 3.

The first phase of this technique is identical to Strategy 1. Once the L-shaped area C_{tot} is computed, the candidate check for dimension 1 can begin. However, rather than scanning all attributes represented by C_{tot}, this area is reduced by "AND"ing together the candidate bitmap C_1 with C_{tot} (see Figure 5c)). In this case the candidate selectivity is $s_1 = (\overline{v}_1 - \underline{v}_1)\overline{v}_2$. The result of this candidate check r_1 is combined with C_{tot} and C_1 to produce a refined candidate set $R_1 = C_{tot} \wedge \neg(M_1 \wedge \neg r_1)$, where $M_1 = C_{tot} \wedge C_1$.

To determine the minimal candidate set for attribute 2, the refined candidate set R_1 and the candidate set C_2 are "AND"ed together, which produces M_2. The area representing M_2 is $v_1(\overline{v}_2 - \underline{v}_2)$. Let r_2 denote the result of this candidate check. The final result of the two dimensional query is $H \vee R_2$, where $R_2 = R_1 \wedge \neg(M_2 \wedge \neg r_2)$.

The whole process is depicted in Figure 5. In general, the candidate selectivity for attribute i is as follows:

$$s_i = \left(\prod_{j=1}^{i-1} v_j \right) (\overline{v}_i - \underline{v}_i) \left(\prod_{j=i+1}^{d} \overline{v}_j \right) \tag{3}$$

This strategy checks the minimal number of candidates. It achieves this with some extra operations on bitmaps. The first two strategies only need to access the bitmap indices once. However, this strategy has to access the bitmap indices twice: once to determine the initial candidate set C_{tot} and once to determine the candidates for the ith attribute to compute C_i. In addition, it needs more bitwise logical operations after each candidate checking to refine the candidate sets. These extra bitmap operations clearly require time. One question we seek to address is whether the savings in reduced candidate checking is enough to offset these extra operations on bitmaps.

4 Analytical Results

In this section we evaluate the three strategies discussed in Section 3 according to the number of candidates checked and the number of pages accessed. All evaluations are carried out on a data set of 25 million records with 10 attributes. All attributes are uniform random values in the range of [0, 1]. We have chosen 25 million records since our real data used in the next section also comprises of 25 million entries. For each dimension we assume a bitmap index with 300 bins as in the performance tests in the next section. The page size is 8KB. Each attribute value takes 4 bytes to store and all pages are packed as in a typical projection index [5]. In many data warehouse applications, the projection index is observed to have the best performance in answering complex queries. We use it as a reference for measuring the performance of the various bitmap schemes. To simplify the evaluation, we set all v_i to be the same. Furthermore, we assume the query boundaries are never exactly on the bin boundaries, i.e., $v \neq \overline{v} \neq \underline{v}$. Figure 6 shows the number of candidates expected (according to Equations 1, 2 and 3) and Figure 7 shows the number of page accesses expected.

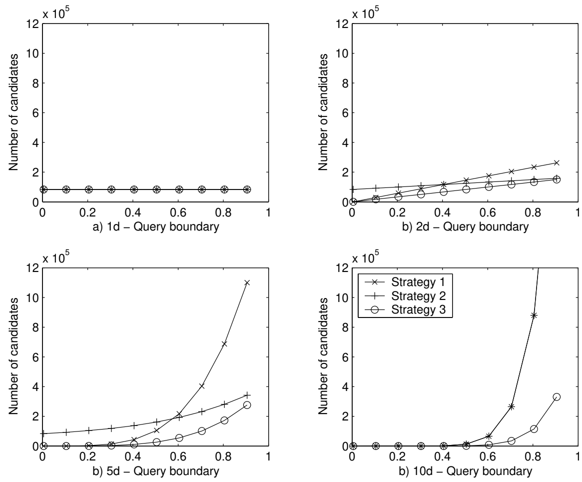

Fig. 6. Total number of candidates for multi-dimensional queries.

For one-dimensional queries there is no performance difference among the strategies (see Figure 6a)). In all cases, the candidate selectivity is 0.3% (= 1/300) which corresponds to all entries of one candidate bitmap. Note that since each page contains 2048 records, selecting one out of every 300 records means all pages are accessed.

When more than one attribute is involved, there is a significant difference among the strategies. We observe that Strategy 1 performs better than Strategy

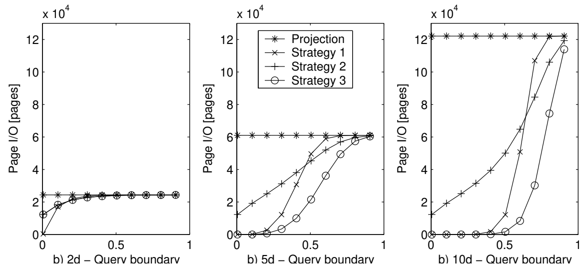

Fig. 7. Page I/O for multi-dimensional queries.

2 for queries with boundaries below 0.5. For query with boundaries above 0.5, Strategy 2 performs better than Strategy 1. However, in all cases Strategy 3 shows the best performance characteristics.

5 Bitmap Index Performance

Querying Synthetic Data: We first verify the performance model on uniform random attributes. As in the previous section, we generated a bitmap index for 25 million records and 10 attributes. The index consists of 300 range-encoded bins. The whole bitmap index is compressed with the WAH scheme [8]. The size of the base data is 1 GB. The size of the bitmap indices is about 10 times larger because range-encoded bitmap indices are hard to compress. The experiments are carried out on a 2.8 GHz Intel Pentium IV with 1 GB RAM. The I/O subsystem is a hardware RAID with two SCSI disks.

To verify the performance model, we ran the same benchmarks as reported in Section 4. The results for multi-dimensional queries with query boundaries in the range of [0, 1] are shown in Figure 8. We can see that in all cases strategy 3 results in the lowest number of candidates to be checked. The number of candidates checked is exactly as expected.

Figure 9 shows the query response time. As we expected from our analytical results, strategy 3 performs best in most cases and has a performance gain of up to a factor of two. We also see that apart from one case, the bitmap index performs always better than the projection index.

Querying Scientific Data: We also tested our new bitmap evaluation strategies on a set of real data from combustion studies. The data was computed from a direct numerical simulation of hydrogen-oxygen autoignition processes. Our timing measurements use randomly generated conditions on 10 chemical species involving hydrogen and oxygen [9]. For each attribute we built a range-encoded bitmap index with 300 bins. In this case, the total index size is only 40% larger than the base data because the distribution of the real data is not uniform. The query performance results are presented in Figure 10. We observe that all

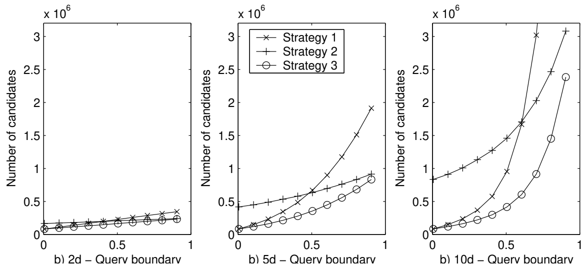

Fig. 8. Total number of candidates for multi-dimensional queries against uniformly distributed random data.

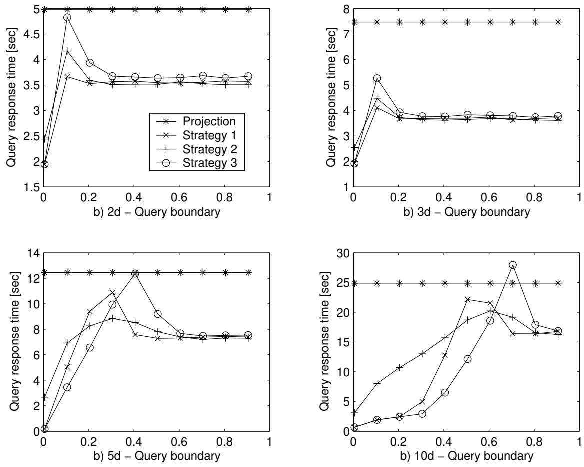

Fig. 9. Response times for multi-dimensional queries against uniformly distributed random data.

bitmap schemes work significantly better than the projection index. The relative differences among the three bitmap based evaluation strategies are small because fewer candidates are scanned than on the uniform random data. In general, Strategy 1 uses more time than others, and Strategies 2 and 3 use about the same amount of time. The average query response time using compressed bitmap indices is less than one second in all tests. For 5-dimensional queries the compressed bitmap indices are, on average, about 13 times faster than the projection index.

6 Conclusions

We introduced a novel bitmap evaluation strategy for bitmap indices with bins. It minimizes the number of records scanned during the candidate checking, but

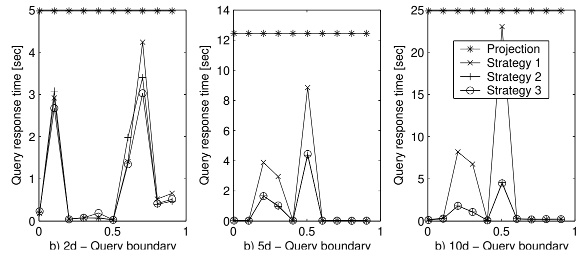

Fig. 10. Response times for multi-dimensional queries against real data.

requires more operations on bitmaps. We provided detailed analyses and experimental measurements to verify that the new scheme is indeed efficient. In many cases the new strategy scans much fewer records than previous strategies, and in some cases, it can improve the query response time by a factor of two. All strategies are shown to outperform the projection index in the majority of the cases.

Since the new bitmap index evaluation strategy uses more bitmap operations, it occasionally uses more time than others. In the future, we plan to develop a way to optimally combine the three strategies.

References

1. S. Chaudhuri and U. Dayal, An Overview of Data Warehousing and OLAP Technology, *ACM SIGMOD Record 26(1)*, March 1997.
2. C. Chan, Y.E. Ioannidis, Bitmap Index Design and Evaluation, *In SIGMOD 1998*, Seattle, Washington, USA, June 1998, ACM Press.
3. C. Chan, Y.E. Ioannidis, An Efficient Bitmap Encoding Scheme for Selection Queries. In *SIGMOD 1999*, Philadelphia, Pennsylvania, USA, June 1999, ACM Press.
4. T. Johnson, Performance Measurements of Compressed Bitmap Indices, *In VLDB 1999*, Edinburgh, Scotland, UK, September 1999, Morgan Kaufmann.
5. P. O'Neil and D. Quass. Improved Query Performance With Variant Indices. In *SIGMOD 1997*, Tucson, Arizona, USA, May 1997, ACM Press.
6. A. Shoshani, L. M. Bernardo, H. Nordberg, D. Rotem, and A. Sim. In *SSDBM 1999*, July 1999, IEEE Computer Society Press.
7. K. Stockinger. Bitmap Indices for Speeding Up High-Dimensional Data Analysis, In*DEXA 2002*, Aix-en-Provence, France, September 2002, Springer-Verlag.
8. K. Stockinger, K. Wu, and A. Shoshani. Strategies for Processing ad-hoc Queries on Large Data Warehouses. In *(DOLAP 2002)*, McLean, VA, USA, November 2002. ACM Press.
9. K. Wu, W. Koegler, J. Chen and A. Shoshani, Using Bitmap Index for Interactive Exploration of Large Datasets. In *SSDBM 2003*, July 2003, IEEE Computer Society Press.