Cost-Effective Usage of Bitmap-Indexes in DS-Systems

Andreas Lübcke
Otto-von-Guericke-University Magdeburg
Department of Computer Science
Institute for Technical and Business Information Systems
D-39016 Magdeburg, Germany
P.O. Box 4120
andreas.luebcke@ovgu.de

1 Introduction

Index structures are a widely used function of Database Management Systems (DBMS) in order to tune them for a special purpose. Finding the right index configuration is on the one hand extremely complex (NP-problem [6]) and a variation of the Knapsack Problem [9] but on the other hand manual configuration requires high administrative effort by cost-intensive experts (like DBAs).
Regarding these facts several approaches like Feedback Control Loop [14] or MAPE from IBM [7] were introduced in current research of self-tuning techniques. These proposals should reduce costs for human resources and bring us closer to more efficient autonomous DBS.
Based on these perceptions we analysed the state of the art in the field of Data Warehousing and Online Analytical Processing (OLAP) systems. In this area it is noticeable that bitmap-indexes have often a marginal use in spite of crucial advantages in the given scenarios of DSS. Many case studies are available for the Oracle DBMS which has integrated a support for bitmap-indexes [8]. Instead we could find several approaches, which were based on heuristics of experts and DBAs like Join-Indexes [12, 4]. With the aid of these approaches it is possible to evaluate design tools like Database Tuning Advisor for Microsoft SQL Server 2005 [1, 2] or DB2 Design Advisor [15] and give online advise to tune a DBS, but until now it is not possible to give an online cost estimate for bitmap-indexes. Hence, a self-tuning approach like those for B-trees [3, 11] is not available yet.

2 An Approach and Ideas on Bitmap-Index Tuning

2.1 Capability of Bitmap-Indexes

A bitmap-index is composed of a number of bit vectors where every existing value of the indexed attribute is represented by a single bit vector. The structure of a simple bitmap-index is shown in **Figure 1**. We know from experience that bitmap-indexes are particularly suitable for low cardinality attributes.
This fact leads us to the point where we must consider that only predicates from the *WHERE*-Clause are relevant to bitmap-indexes on a relation $r(R)$. Furthermore, these predicates have to be in the following form:

$$A = const.\ with\ A \in R$$

We assume that attribute A, which should be used for index candidates, has to satisfy the following condition:

$$card(dom(A))/card(r(R)) < maxSelectivity \quad (1)$$

We will name **Formula (1)** as Attribute Value Cardinality (**AVC**) in the following considerations. Some case studies show, that 1/10000 is a practicable threshold for the AVC, e. g. in Oracle 9i Date Warehousing Guide [10].
Other important facts according to bitmap-indexes are the low building cost and humble required space compared to other index structures.

TID	B	F	O
$rowid_1$	0	0	1
$rowid_2$	0	1	0
$rowid_3$	0	0	1
$rowid_4$	1	0	0
$rowid_5$	1	0	0
$rowid_6$	0	1	0
$rowid_7$	0	0	1
⋮	⋮	⋮	⋮

Figure 1: Structure of a Sample-Bitmap-Index

2.2 Underlying Self-Tuning Concepts

The following explanation for the tuning procedure is based on the Feedback Control Loop [13]. The three steps of the control cycle will be clarified through **Figure 2**. First step is the observation where the system collects statistics about the data, existing index configuration and workload. In the second step we monitor continuously our statistics and the altering conditions, e. g. the load on one special relation. Hence, we can decide, when a new index configuration could be better than the old one. After all we execute several operations[1] to tune the index configuration.

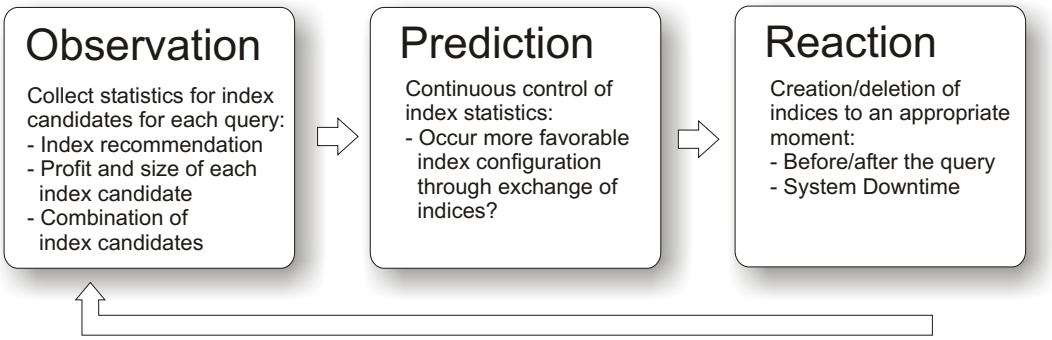

Figure 2: Feedback Control Loop according Weikum et al.

But, first of all we have to find algorithms to weigh the cost and benefit of an index candidate and also an index configuration. For the further views we take a number of queries $Q_1, ..., Q_m$ and set of index candidates $I_1, ..., I_n$ for granted. We come to the conclusion that the benefit of an index candidate I_i for a query Q_k is the maximum of difference between the execution time of query Q_k without index candidate I_i and with index candidate I_i. Hence, we will annotate in **Formula (2)**:

$$profit\,(Q_k, I_i) = max\,\{0, cost\,(Q_k) - cost\,(Q_k, I_i)\} \tag{2}$$

Furthermore, we must consider the involved administration costs to our profit calculation which we establish as $mcost\,(I_i)$ for index I_i. Assuming an assessment for one index candidate is not practicable, we adopt an index configuration $C \subseteq I_1, ..., I_j$. So we make use of **Formula (2)**

[1] Create or delete indexes

and account for maximisation of our profit in **Formula (3)**:

$$\sum_{i=1}^{m} max\left\{profit\left(Q_i, I_j : I_j \in C\right)\right\} - \sum_{I_j \in C} mcost\left(I_j\right) \tag{3}$$

In addition we have to consider that we have a restricted amount of space S in our index pool. From this we conclude to **Formula (4)**:

$$\sum_{I_j \in C} size\left(I_j\right) \leq S \tag{4}$$

Thus we can estimate after analyses of **Formula 1-4** [11], when a bitmap-index configuration C is cost-efficient. But we do not attend to the peculiarities of bitmap-indexes. Hence, we can not compare bitmap-indexes performance clearly with other index structures.

2.3 Singularity of Bitmap-Indexes according to Cost-Models

Section 2.2 presents a generally admitted cost-model which is suitable for **B-trees**. However, the cost-model does not fit for bitmap-indexes, because it does not consider the extreme high update costs $\left(O\left(n^2\right)\right)$ for a bitmap-index. Beyond we have no possibility in our established cost-model to include the advantage of logical connectability for bitmap-indexes.
To the first fact, we assume that the update-problem of bitmap-indexes is a facile problem for DSS, because updates are in these scenarios rare. Mostly the updates will be performed after the load process, before the system goes online. Furthermore, the indexes will not updated because of the huge amount of data, but rather they will be recreated. An other technique is to mark deleted tuple in an extra bit vector, to avoid the reorganisation of the bitmap-index every time. But in OLAP systems both assumptions will not afford a cost-effective index configuration.
Thus, a cost-model extension is required. Supposed that the system has all statistics about the data, indexes and workload, we can estimate the behaviour (e. g. the workload) with help of a probability density function (Ψ) of a probability. We assume our probability density function (Ψ) could be an Exponential distribution to simulate a workload. Leading to:

$$f_\lambda(x) = \lambda e^{-\lambda x} \; for \; x \geq 0 \; and \; f_\lambda(x) = 0 \; for \; x < 0$$

Hence, we describe the update behaviour in our example with the parameter $\alpha = 1 - \Psi$ which is the negation of our probability density function (Ψ) . In this case an attribute, which has not been updated for longer time, would have a high probability (theoretic) get updated in the near future. The conclusion is that the profit of a bitmap-index I_i have to be reduced, if it is updated very frequently. In other words, the adapted cost-model is defined as follows:

$$profit(Q_k, I_i) = \alpha * max\left\{0, cost(Q_k) - cost(Q_k, I_i)\right\} \tag{5}$$

The second aspect, we have to regard, is the logical connectability for bitmap-indexes. From this can be concluded that a bitmap-index can not only be used for queries in which the single indexed attribute is used but also as prefix or suffix in complex queries like B-trees too. Hence, the multi-usage of an index candidate have to be observed and should be included into the cost-model. The huge advantage of bitmap-indexes is the logical connection in any order between different bitmap-indexes to use the index candidates for complex queries. Hence, we assume that we did not have to create multi-value-bitmap-indexes, because they can build by their components (indexes). That saves space in the restricted index pool and gives us more variability for index configurations, because the attributes have not be multi-indexed in different orders or combinations. Considering these facts, we suggest a solution which follows our proposal to the update-problem. For this purpose we take any probability density function (Θ) for granted.

However, we assume that probability density functions like the exponential distribution are not adequate for our intention. Normal curve of distribution or student distribution are a more appropriate real world section in this case. In our example we choose the student distribution because this distribution has been proven that it is suitable for sample analysis.
Under the same conditions as before, the Student distribution is defined as follows:

$$f_n(x) = \frac{\Gamma(\frac{n+1}{2})}{\sqrt{n\pi}\Gamma\left(\frac{n}{2}\right)}(1+\frac{x^2}{n})^{-\frac{n+1}{2}} \quad for \ -\infty < x < +\infty$$

$$\text{with } \Gamma(x) = \int_0^{+\infty} t^{x-1}e^{-t}dt$$

Furthermore, we introduce a new parameter $\omega = 1 + \Theta$ to raise the benefit. However, we have to choose the enhancement not too severe, because the probability for re-use should not be estimated too high. But if the probability for re-use is zero, we get with aid of our parameter ω a minimum factor of one. These observations lead to a second new cost-model formula:

$$profit(Q_k, I_i) = \omega * max\left\{0, cost(Q_k) - cost(Q_k, I_i)\right\} \tag{6}$$

We have noticed the special characteristics of bitmap-indexes with **Formula 5** and **6**. Both formulas together form an improved new cost-model for bitmap-indexes:

$$profit(Q_k, I_i) = \alpha(\omega * max\left\{0, cost(Q_k) - cost(Q_k, I_i)\right\}) \tag{7}$$

The cost-model in **Formula 7** allows a comparison of bitmap-index configurations with other index structures, because we can approximate the adavantage of connectability and the update behaviour with this cost model. Additionally, we can describe the peculiarities for bitmap-indexes better than before.

2.4 Possibilities and Variants of Bitmap-Indexes

First, we will present the different variants of bitmap-indexes and why we do not observe them. One variant is the interval-coordinated-bitmap-index which is a special case and would get us away from a general approach for bitmap-indexes. Furthermore, these indexes are very complex and more difficult to predict. Hence, we should find first a solution for non-specialised bitmap-indexes. A second variant is the multi-value-bitmap-index which is more complex in his structure. This is one reason why they are such difficult to estimate. Additional complexity restricts our search domain to find a minimal and optimal[2] index configuration [5]. Furthermore, the multi-value-bitmap-indexes are not as flexible as needed to be for our targets. At last, these indexes could build together by the single indexed attributes.
Beyond we will discuss the possibility to use bitmap-indexes together with other index structures. The problem is that only one index structure can be used out of index pool. How can we conquer or avoid this problem of index pool management. One idea could be to get a set of index pools, e. g. one for every index structure. Hence, we have to change the optimizer of the DBMS and all other parts which work with the indexes. But is it possible to resolve such a complex problem with all its variants and peculiarities? An other idea is to have a set of optimizer and management instances, one for each index structure. But we have the problem how these different optimizer can communicate with each other. However, They have to collaborate with each other because we have to avoid multi-indexed attributes or relations. This solution could impose a huge overhead for optimization. These are challenging and interesting question which we want to answer.

[2] As far as possible because of the complexity of the search problem

3 Conclusion

We have shown different aspects in the field of bitmap-indexes. We pointed out the advantages and disadvantages of bitmap-indexes compared with other index structures. Furthermore, we developed an improved cost-model for bitmap-indexes according the characteristics of these indexes. The cost-model build the foundation to use index-self-tuning techniques including bitmap-indexes. There are many open problems and new approaches are needed, e. g. in the field of comparability and combined usage with other types of index structures.

References

[1] S. Agrawal, S. Chaudhuri, L. Kollár, A. P. Marathe, V. R. Narasayya, and M. Syamala. Database Tuning Advisor for Microsoft SQL Server 2005. In *VLDB '04*, pages 1110–1121, 2004.

[2] N. Bruno and S. Chaudhuri. To Tune or not to Tune? A Lightweight Physical Design Alerter. In *VLDB '06*, pages 499–510, 2006.

[3] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. In *ICDE*, pages 826–835. IEEE, 2007.

[4] D. Burleson. How to use oracle9i bitmap join indexes. http://www.dba-oracle.com/art-builder-bitmap-join-idx.htm, November 2002. Visited Novmeber 2007.

[5] C.-Y. Chan and Y. E. Ioannidis. Bitmap index design and evaluation. pages 355–366, 1998.

[6] D. Comer. The Difficulty of Optimum Index Selection. *ACM Transactions on Database Systems*, 3(4):440–445, 1978.

[7] I. Corporation. An architectural blueprint for autonomic computing. White Paper, June 2005. Third Edition.

[8] A. B. Danchenkov and D. K. Burleson. *Oracle Tuning: The Definitive Reference*. Rampant Techpress, 2006.

[9] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, Berlin, Heidelberg, 2004.

[10] P. Lane and V. Schupmann. *Oracle9i Data Warehousing Guide, Release 2 (9.2)*, pages 122–125. Oracle Corporation, 2002.

[11] M. Lühring, K.-U. Sattler, E. Schallehn, and K. Schmidt. Autonomes index tuning - dbms-integrierte verwaltung von soft indexen. In *BTW*, pages 152–171, 2007.

[12] P. Valduriez. Join indices. *ACM Trans. Database Syst.*, 12(2):218–246, 1987.

[13] G. Weikum, C. Hasse, A. Moenkeberg, and P. Zabback. The COMFORT Automatic Tuning Project, Invited Project Review. *Information Systems*, 19(5):381–432, 1994.

[14] G. Weikum, A. Mönkeberg, C. Hasse, and P. Zabback. Self-tuning Database Technology and Information Services: from Wishful Thinking to Viable Engineering. In *VLDB '02*, pages 20 – 31, 2002.

[15] D. C. Zilio, J. Rao, S. Lightstone, G. M. Lohman, A. Storm, C. Garcia-Arellano, and S. Fadden. DB2 Design Advisor: Integrated Automatic Physical Database Design. In *VLDB '04*, pages 1087–1097, 2004.