On the Selection of a Reduced Set of Indexes

MICHAEL HATZOPOULOS* AND J. (YIANNIS) KOLLIAS**

*Department of Mathematical and Ccmputer Sciences, Michigan Technological University, Houghton, MI 49931, USA

**Department of Computer Sciences, National Technical University of Athens, 9 Heroon Polytechnion Ave., Zografou, Athens (621), Greece

The secondary index selection problem (ISP) is known to be an NP-complete problem. The best possible known algorithm to find an exact solution of the ISP is the one developed by Schkolnick which utilizes the fact that the objective function of the problem satisfies the regularity condition. In a recent paper Ip et al presented a heuristic algorithm which solves the ISP with the added constraint that the creation and storage cost of the optimal selection must not exceed a certain specified cost. The current study observes that in practice this last problem can be translated to the classical ISP but requiring now that no more than k indexes may appear in the optimal set. It is shown that this new problem satisfies the regularity property. The study also presents the modifications needed to Schkolnick's algorithm for solving the problem considered.

1. INTRODUCTION

In a recent paper Ip et al[1] studied the problem of the optimal selection of secondary indexes to support a database that uses the values of secondary attributes. This problem is one of the most debated problems in the area of physical database design (see for example references 2,3 and the references cited in 1) and it is known to be NP-complete[4]. The best known (exact) algorithm, for solving the index selection problem (ISP), is the one developed by Schkolnick[5] which determines the optimal selection among n candidate secondary indexes by examining at most $O(2^{\sqrt{n}} \log n)$ combinations out of the 2^n possible combinations.

In 5 one planning period was considered and it was assumed that during that period the system receives transactions which may request the querying, update, insertion and deletion of records. The ISP was formulated as: Find the selection of indexes which minimizes the quantity $r_Q C_Q + r_U C_U + r_I C_I + r_D C_D$ (1), where r_Q, r_U, r_I and r_D are respectively the probabilities that one transaction is a query, an update, an insertion or a deletion, and, C_Q, C_U, C_I and C_D are the costs to process respectively a query, update, insertion and deletion. The algorithm devevoped substantially reduces the solution space to be searched when solving the ISP because it utilizes the fact that the cost function (1) satisfies the *regularity condition*. The condition states that, if while searching for the optimal set of indexes, a set D is reached and there is an index $j \notin D$ which increases (1), then the index j can be ignored in any subsequent search from D. For a formal definition of regularity see the Appendix.

The model in reference 1 is very similar to that in reference 5 except that it requires the minimization of (1) (Note: reference 1 asks basically for the maximization of a quantity of the form $A - r_Q C_Q - r_U C_U - r_I C_I - r_D C_D$, where A is the average cost to perform any type of transaction if no index exists in the database) subject to the additional constraint:

$$\sum_{j=1}^{n} x_j d_j \leqslant M \qquad (2)$$

where

x_j = the storage and creation cost for the index of the j-th attribute,

M = the storage and creation cost of the optimal index selection that it can be tolerated, and

d_j = 1 if the j-th attribute is indexed
0 otherwise

Therefore the problem in reference 1 is the classical ISP with the restriction that the selection must not lead to overall index creation and storage cost which exceeds a specified amount, M.

Unfortunately, the existence of (2) does not ensure the regularity condition of (1). It has been shown in reference 1 that the (1) – (2) model is equivalent to another NP-complete problem, namely the knapsack problem[6]. In the next section we argue that in practice it is reasonable to assume that the creation and storage cost, x, of *each* secondary index is a constant quite irrespective of the value of j. This observation allows us to consider the problem of minimizing (1) when no more than k indexes may appear in the optimal selection, where $k = \lfloor M/x \rfloor$. We subsequently show that this new problem maintains the regularity condition of (1). Section 3 presents the modifications needed to Schkolnick's algorithm in order to solve our problem and reports that the modifications by no means harm the initial performance of the algorithm.

2. THE NEW MODEL

Our first task is to show why it is reasonable to assume that each of the quantities x_j in (2) can be equated to the value of a constant x, i.e. in practice it is expected that secondary indexes have almost the same creation and storage costs. We examine both these costs in turn.

The creation of a secondary index is undoubtly a batch operation which takes place periodically. The problem of determining optimal intervals for creating a new version of an index is beyond the scope of the ISP as described in references 1 and 5. (However, there exist studies which may be used for attacking the optimal period selection problem[7,8]). Therefore references 1 and 5 implicitly assume that index creation occurs at the very beginning of the planning period. For the discussion that follows, we assume that a secondary index will be structured as an inverted file. The same arguments are generally applied when less popular filing schemes are used to structure indexes[9]. The process of creating an index j (j = 1, 2, ..., n) normally consists of the following

three steps: *Step* 1: The primary file is read and distributed into another sequential file having records consisting of two fields. The first field is an identification of the record (normally the primary key and rarely the address of the record in the disk) and the second is the value of the j-th attribute. If we assume that the primary file consists of N records then the output file creation is an O(N) operation. It is worth noting that if the size of the file/disk allows it, it is possible, (a) the primary file to be distributed into more than one output files, or (b) if one output file is used, then this file may contain more than one secondary attributes. Both cases although may lead to different (and cheaper) creation costs will make (1)-(2) a non-linear optimization problem. *Step* 2: The distributed file is now sorted on the values of the attribute j. If an m-way merging is used for sorting this file[6] where m is the order of merging – then again the cost is $O(N\log_m N)$ quite irrespective of the attribute j. *Step* 3: The inverted file is structured by applying some type of hierarchical structure, e.g. B-tree[10], etc. If we show that the number of nodes of the hierarchical structures are almost the same, for every index j, then the validity of our argument about the identical index creation cost is proved. This issue is immediately discussed in relation to the index storage costs.

An entry of the inverted directory for the j-th index logically consists of two parts: a distinct value of the j-th attribute and a list of identification of records (see above) which contain that particular key value. If a tree structure is employed to structure an inverted directory then the leaves of the tree have to point at exactly N records quite irrespective of the attribute j. Therefore each secondary index requires a storage of O(bN) = O(N), where b is the length in bytes to store a record identifier (i.e. a primary key value or a disk address). Naturally, a secondary attribute which is coded, say, as one character may require some more space for its index than another attribute which is coded, say, as two character string. Nevertheless, the difference in space is so small that it allows us to assume that a secondary index requires the same storage cost with any other secondary index on the system. The discussion also proves that the cost to perform the Step 3 above is the same for all indexes since each tree contains the same number of nodes. At this point we feel appropriate to comment on the fact that in many implementations the few top levels of every secondary index are placed in the main memory for purposes of fast retrievals. The case is catered for in (2) and it can be covered by our model provided of course that the variable x takes now a larger value.

From the above discussion it follows that the (1)–(2) model may be translated in practice as: 'Find the selection of indexes which minimizes the quantity $r_Q C_Q + r_U C_U + r_I C_I + r_D C_D$ when no more than k indexes can appear in the optimal solution, where $k = \lfloor M/x \rfloor$.' Before studying this new model we shall point out another possible application of the model. Secondary index selection is normally performed by the database administrator who utilizes special commands. For example, System R offers the two related commands CREATE and DROP INDEX[11]. However, database systems operating on machines with limited capacities cannot leave uncontrolled the number of secondary indexes to be selected. In fact we are aware that the maximum degree of indexing allowed by software running on the CDC S–18 and IBM S/34 computers is 4 and 15 indexes respectively. This suggests another application of the new model.

Our next step is to prove that the new ISP is regular. We first define a few more variables using the notation in reference 1. Let:

D = the set of attributes for which an index exists. In our case $|D| \leqslant k$,

$\rho_j(\gamma_j)$ = the probability that the j-th attribute is specified in a query (update),

F(j) = the time required to perform a transaction (query/update/insertion/deletion) on any index for the j-th attribute, and

N_j = the number of distinct values taken by the j-th attribute.

With the above introduced variables and following the arguments in reference 1 then (1) can also be written as

$$E(D) = \sum_{j \in D} H(j) + G(D) \tag{3}$$

where

$$H(j) = r_Q \rho_j F(j) + r_U \gamma_j F(j) + (r_I + r_D) F(j)$$

and

$$G(D) = r_Q N \prod_{j \in D} (1 - \rho_j + \rho_j / N_j)$$

It can be seen that in (3), H(j) represents the cost of maintaining the index for the j-th attribute and manipulating it during retrievals; G(D) is the cost of retrieving the records via the intersection lists of D (Note: The process of intersecting indexes is thoroughly described in reference 1).

We now show that (3) is regular. Since our proof will use some of the results in reference 5 we list, in the Appendix, these results.

Theorem: Let E(D) as defined in (3). Then E(D) is regular.

Proof. By definition

$$\begin{aligned} \Delta(D,j) &= H(j) + G(D \cup \{j\}) - G(D) \\ &= H(j) - (\rho_j - \rho_j/N_j) G(D) \\ &= H(j) - \rho_j(1 - 1/N_j) G(D). \end{aligned}$$

The fact the G(D) is a monotone nonincreasing function proves the theorem Q.E.D. (see Lemma in Appendix).

In the next section we show how the above theorem can be utilized to extend Schkolnick's algorithm to cover the ISP when considering the storage and creation cost constraint.

3. THE MODIFIED ALGORITHM

Let S be the set of all candidate indexes, |S| = n. Schkolnick's algorithm partitions S into w (disjoint)

chains. The set of candidates to be adjoined to a current partial solution D (Initially $D = \phi$) is obtained by considering the subset of independent points among the set of points which are maximal in each chain. If n_1, n_2, ..., n_w are the number of points in each chain, the maximal number of sets examined will be less than

$$(1 + n_1)(1 + n_2)...(1 + n_w) \leqslant (1 + n/w)^w. \quad (4)$$

The modification we propose reflects the fact that our problem must also satisfy $|D| \leqslant k$. This requires the following two modifications of the algorithm:

Modification 1. When searching for a local optimal the condition $|D| = k$ must be examined (this implies that the test *if* $P \neq \phi$ *and* $\forall j \notin D$ ($\Delta(D - \{j\}, j) < 0$ in reference 5 must be augmented with the condition '*and* $|D| = k$ ').

Modification 2. Before a new index enters D the condition $|D| \leqslant k$ must be checked. (This implies that the push to the stack operation in reference 5 must be preceeded by the test 'if $|D| \leqslant k$').

Theorem 1 and the above two modifications achieve the optimal selection of the required reduced set of indexes. This is managed without reducing the efficiency of the algorithm since now (4) becomes

$$(1 + \min(n_1,k))(1 + \min(n_2,k))...(1 + \min(n_w,k)) \leqslant (1 + n/w)^w.$$

APPENDIX

The purpose of the appendix is to list the definitions and results[5] which are utilized in this study.

Definition 1: Let E be a cost function defined on subsets of a set S of points. Let $\Delta(D,j) = E(D \cup \{j\}) - E(D)$. Then E is said to be *regular* if $\Delta(D',j) \leqslant \Delta(D,j)$ for any point j and sets D,D′ for which $D \subseteq D'$ and $j \notin D'$.

Definition 2: Let k be a function which maps subsets of S to any totally ordered domain with order relation given by $<$. Then k is said to be monotone nonincreasing (mni) if $D \subseteq D'$ implies $k(D') \leqslant k(D)$. **Lemma**. Let E be a cost function such that $\Delta(D,j) = E(D \cup \{j\}) - E(D)$ can be written as $\Delta(D,j) = A(j) - B(D,j)$ where, for each fixed j, B(D,j) is mni . Then E is regular.

REFERENCES

1. M.Y.L. Ip, L.V. Saxton and V.V. Ragharan, On the selection of an optimal set of indexes, *IEEE Trans. on Soft. Eng.*, vol. 9, pp. 135–143, (1983).
2. J.G. Kollias, A heuristic approach for determining the optimal degree of file inversion, *Inform. Syst.*, vol. 4, pp. 307–316, (1979).
3. M. Hatzopoulos and J.G. Kollias, Some rules for introducing indexing paths in a primary file, *Comp. J.*, vol. 23, pp. 207–211, (1980).
4. D. Comer, The difficulty of optimum index selection, *ACM TODS*, vol. 3, pp. 440–445, (1978).
5. M. Schkolnick, The optimal selection of secondary indices for files, *Inform. Syst.*, vol. 1, pp. 141–146, (1975).
6. E. Horowitz and S. Sahni, *Fundamentals of Data Structures* (Comput. Software Eng. Series) Comput. Sci. Press, (1978).
7. M. Hatzopoulos and J.G. Kollias, A dynamic model for the selection of secondary indexes, *Inform. Syst.*, vol. 8, pp. 159–164, (1983).
8. G.M. Lohman and J.A. Muckstadt, Optimum policy for batch operations: backup, checkpointing, reorganization and updating, *ACM TODS*, vol. 2, pp. 209–222, (1977).
9. T.J. Teorey and J.P. Fry, *Design of Database Structures*, Prentice Hall, (1982).
10. R. Bayer and E. McGreight, Organization and maintenance of large ordered indices, *Acta Informatica*, vol. 1, pp. 173–189, (1972).
11. C.J. Date, *An Introduction to Database Systems* (The Systems Programming Series) Addison-Wesley, 3rd Ed., p. 110, (1981).