Retrieval Efficiency Using Combined Indices

by

Michael Stonebraker*

Abstract

The problem considered here involves choosing the best set of indices for indexing a file on a secondary storage device where space may be limited. For a general class of queries and a specific index organization, approximations to the expected retrieval time for any choice of indices are developed. Subject to the simplifying assumptions the best selection of indices is obtained for several cases, both where the number of possible lists is constrained and where it is not. The examples indicate that retrieval time is quite sensitive to the choice made.

*Department of Electrical Engineering and Computer Science, University of California, Berkeley.

Introduction

We propose to treat the problem of choosing the redundant information about a file located on a secondary storage device of constrained size so that the file can be accessed in a reasonable fashion. Work of a similar nature (though not related to the current study) is reported in [1,2,3]. In order to specify a precise problem, we make certain sets of assumptions concerning the file, the queries, and the storage medium.

In particular, suppose the file, F, consists of N records $\{O_i\}$, $1 \leq i \leq N$, each containing a value for m attributes, e.g. $O_i = (a_{1i}, a_{2i}, \ldots, a_{mi})$. We shall assume that a_{ij} is real for all i,j and is bounded in the following sense. For any i and all j, $a_{ij} \subset B_i$ where B_i is a bounded interval of the real line. It is assumed that no relations are permitted between records in the file (i.e. the file is normalized [4]) and that the entire data base consists of a single file. However, the extension of the analysis presented to multifile data bases appears straightforward. We shall also assume that the N records are uniformly distributed over the subspace of Euclidian m space, $B_1 x B_2 x \ldots x B_m$. Hence, in any range R_i for attribute i, there are $(R_i/B_i)N$ records. Moreover, $(R_i/B_i)(R_j/B_j)N$ records simultaneously have attribute i in R_i and attribute j in R_j. If information to the contrary were known, another distribution could be used. However, it would complicate the analysis to follow. Also, the file could be coded

in such a fashion as to more nearly satisfy this assumption.

We make the following assumptions concerning the file activity. First, we shall ignore insertions, deletions, and record updates and will be solely concerned with retrieval. Second, queries will be made from an on line terminal and will be satisfied rapidly. (Clearly, the batching of queries would give the storage structure designer a different set of problems than might otherwise exist.) Third, queries shall conform to the following format:

$$\text{GET}[a_{ij},\ i\varepsilon V \text{ for all } j \text{ such that } a_{1j}\subset R_1, a_{2j}\subset R_2, \ldots, a_{nj}\subset R_n] \quad (1)$$

Here, R_i, $1\leq i\leq n$, are bounded intervals of the real line, $R_i\subseteq B_i$, $n\leq m$, and V is a subset of the positive integers less than m+1. Thus, the values of a subset of attributes are required for all records which have a value for attribute 1 in a range, R_1, attribute 2 in a range R_2, etc.

This query structure is chosen for several reasons. First, it contains as a subset frequently used and well understood query sets. For example, retrieval by a primary key, e.g. attribute 1, would require $R_i=B_i$ for $2\leq i\leq n$, V to be all m attributes, and R_1 could either be a single point or a range of values, if partial keys were allowed. Second, if the set of Boolean combinations of pairs of the form (attribute--range of values) is the query set allowed the user, a member of it may be processed into disjunctive normal form by an intermediate process and the storage structure presented with a number of commands of the form (1). The overall query could then be satisfied by a union of the indi-

vidual responses. Lastly, it is possible to process some queries based on a first order predicate calculus [5] into terms of the form (1). For example, suppose a file has records with three attributes, part number, supplier number and city of supplier (suitably coded). The query "find all suppliers in the same city as the supplier of part #Z" can be readily decomposed into two retrievals of the form (1). Hence, (1) may well be a useful primitive for complex query sets.

It is furthermore assumed that query activity is adequately represented by the condition that $\{R_1,\ldots,R_n\}$ are mutually independent random variables and that

$$\text{prob}[R_i=B_i]=1-P_i$$

$$\text{prob}[R_i\subset B_i]=P_i.$$

Hence, P_i represents the probability that attribute i appears non trivially in a query. Denote by S_i the length of the interval R_i, and assume

$$E[S_i \mid R_i\neq B_i]=G_i.$$

Hence, the expected size of the requested range is G_i given that attribute i appears non trivially. (Note that $\{P_i,G_i\}$, $1\leq i\leq n$, can either be computed from monitored data or estimated by a data base administrator.) It should be clearly noted that the independence assumption cannot easily be removed, and the analysis to follow is applicable only in situations where this assumption is reasonable.

Now, several assumptions must be made concerning the storage

medium. We assume it is a rotating storage device with a seek time, C_1, and a transfer time per four bytes, C_2. The particular values of C_1 and C_2 depend, of course, on the specific device used. We assume that the records of the file will be stored sequentially and that redundant indexed lists may also be constructed. An indexed list for attribute i has the form $\{(a_{ij},T_j)\}$. Here, a_{ij} is the value for attribute i of record j and T_j is a pointer to the j-th record of the sequential file. We assume that this list would be sorted into ascending order of attribute i and a bucket approach followed with all records within a certain range stored together.[1] Therefore, B_i will be divided into buckets of a size, ℓ_i, yet to be determined. Each bucket would contain a pointer and the value of attribute i for all records with attribute i in the required range. Combined indexed lists of two attributes are also allowed. These would be of the form $\{(a_{ji},a_{ki},T_i)\}$ and would be stored in two dimensional buckets of size $\ell_j x \ell_k$. Indexed lists of any number of attributes are acceptable. The two questions of interest in this study are:

1. If the number of indexed lists is unrestricted, what is the best choice of lists?

2. If only k indexed lists are allowed, what is the best choice of lists?

---

[1]There are, of course, many reasons why a multilevel structure might be chosen in practice and no apriori division of B_j into buckets declared. Analysis of such structures often involves details of a particular device and appears to be exceedingly complex. The assumption chosen represents a simple approximation that avoids both problems.

The criterion of "best" shall be to minimize the mean time necessary to access the redundant lists to obtain pointers to all potentially relevant records. This criterion is chosen for two reasons. First, in many cases the CPU time required to process the query may be small compared to retrieval time. Hence, the response time would depend on the speed of obtaining the required information from secondary storage. Secondly, any storage organization would require accessing the contents of all relevant records for a given query. Thus, it is appropriate to minimize the extra time required to search redundant lists. With these considerations in mind, we can now turn to analysis of the above two questions.

Computation of Retrieval Times

In order to proceed we have to make assumptions concerning the "edge effects". Because ℓ_j may not divide the length of B_j exactly, there will be one odd-sized bucket. In all that follows we shall ignore this irregularity. Hence, the formulas to follow are only approximately correct and valid technically only in the case that ℓ_j divides B_j exactly.

For a single indexed list for attribute j, there is a probability $1-P_j$ that attribute j appears trivially and no retrieval from secondary storage is needed. With probability P_j, the number of buckets retrieved has an expected value of $1+G_j/\ell_j$. Hence the expected time required assuming that all pointers and attribute

values require four bytes is:

$$E(\tau_j) = P_j(1+G_j/\ell_j)(C_1+2C_2N\ell_j/B_j) \qquad (2)$$

Here, ℓ_j/B_j is the expected percentage of the records falling in an individual bucket. Hence, the last factor represents the expected time to retrieve the contents of a bucket. If n simple lists are formed, it is evident (because all R_i's are independent) that the total expected time, $E(\tau)$, to retrieve all necessary buckets for a query of the form (1) is:

$$E(\tau) = \sum_{j=1}^{n} P_j(1+G_j/\ell_j)(C_1+2C_2N\ell_j/B_j) \qquad (3)$$

It is easily shown that (3) is minimized if ℓ_j is chosen as

$$\ell_j = \sqrt{G_jC_1B_j/2C_2N} \ . \qquad (4)$$

This minimum time is

$$E_{min}(\tau) = \sum_{j=1}^{n} C_1P_j(1+\sqrt{2C_2NG_j/C_1B_j})^2. \qquad (5)$$

The first question then becomes: Can a time smaller than (5) be obtained by using a more complex directory?

If a combined index is formed with attribute i and j, then the expected retrieval time with probability $P_i(1-P_j)$ is

$$(1+G_i/\ell_i)(B_j/\ell_j)(C_1+3C_2N\ell_i\ell_j/B_iB_j).$$

With probability $P_j(1-P_i)$ it is

$$(1+G_j/\ell_j)(B_i/\ell_i)(C_1+3C_2N\ell_i\ell_j/B_iB_j)$$

and with probability P_iP_j it is

$$(1+G_i/\ell_i)(1+G_j/\ell_j)(C_1+3C_2N\ell_i\ell_j/B_iB_j).$$

Hence, the expected time, $E(\tau_{ij})$, to access the list associated

with the i-th and j-th attributes is

$$E(\tau_{ij}) = (C_1+3C_2N\ell_i\ell_j/B_iB_j)(P_iP_j(1+G_i/\ell_i)(1+G_j/\ell_j)+ \quad (6)$$
$$P_i(1-P_j)(1+G_i/\ell_i)(B_j/\ell_j)+P_j(1-P_i)(1+G_j/\ell_j)(B_i/\ell_i))$$

The best values for ℓ_i and ℓ_j present analytical difficulties. However, we can solve for them in one special case. If $P_i=P_j=1$, then both attributes appear non trivially in each query and the best choice for ℓ_i and ℓ_j can easily be found as

$$\ell_i = \sqrt[3]{WG_i^2/G_j}$$
$$\ell_j = \sqrt[3]{WG_j^2/G_i}$$

where $W=B_iB_jC_1/3C_2N$ and hence

$$E_{min}(\tau_{ij}) = C_1(1+\sqrt[3]{G_iG_j/W})^3. \quad (7)$$

A combined list will be advantageous if the above is less than

$$C_1(1+\sqrt{2C_2NG_j/C_1B_j})^2 + C_1(1+\sqrt{2C_2NG_i/C_1B_i})^2.$$

Although cases can be constructed where this is not true (for example, $G_i=B_i$, $G_j=B_j$, and $G_iG_j/W=1.5$), they represent somewhat unusual device and file characteristics. All other more general cases of (6) involve analytical difficulties. Hence, we will make an additional assumption in order to proceed further.

Thus, we shall not find the ℓ_i and ℓ_j which minimize (6) but rather those which minimize the following variant (which is obtained by requiring all boxes to be searched if $R_i=B_i$ and $R_j=B_j$).

$$E^*(\tau_{ij}) = C_1P_iP_j(1+\ell_i\ell_j/W)(1+Q_i/\ell_i)(1+Q_j/\ell_j)$$

Here, $Q_i = G_i+B_i(1-P_i)/P_i$. The best choice of $\ell_i$and ℓ_j are

$$\ell_i = \sqrt[3]{WQ_i^2/Q_j}$$
$$\ell_j = \sqrt[3]{WQ_j^2/Q_i}$$

(which are safely less than B_i and B_j for all but the smallest values of P_i and P_j). Substituting the above values into (6) yields after some rearrangement

$$E_{min}(\tau_{ij}) = C_1P_iP_j(1+\sqrt[3]{Q_iQ_j/W})^3 - C_1(1-P_i)(1-P_j)(3F)(1+\sqrt[3]{W/Q_iQ_j}) \quad (8)$$

Here, $F=C_2N/C_1$. If nominal values of $N=10^6$, $C_1=75$ msec, $C_2=.02$ msec, $P_i=.5$, and $P_j=.5$ are chosen, then $\sqrt[3]{Q_iQ_j/W} > 9$ and (8) may be approximated by

$$E'_{min}(\tau_{ij}) = C_1P_iP_j(Q_iQ_j/W) - C_1(1-P_i)(1-P_j)(3F) \quad (9)$$

Let $U_i=P_iG_i/B_i$. Thus, (9) may be written as

$$E'_{min}(\tau_{ij}) = 3FC_1((U_i+1-P_i)(U_j+1-P_j)-(1-P_i)(1-P_j)) \quad (10)$$

The above approximate analysis can be performed for a combined list of any size. In this case (8) is replaced by the following more general version written for a combined list of attributes 1,...,h but easily transformed for arbitrary attributes.

$$E_{min}(\tau_{1,\ldots,h}) = C_1P_1\ldots P_h(1+\sqrt[h+1]{(h+1)FQ_1\ldots Q_h/B_1\ldots B_h})^{h+1} - C_1(1-P_1)\ldots(1-P_h)(h+1)F(1+\sqrt[h+1]{B_1\ldots B_h/(h+1)FQ_1\ldots Q_h}) \quad (11)$$

Again, if

$$\sqrt[h+1]{(h+1)FQ_1\ldots Q_h/B_1\ldots B_h} >> 1 \quad (12)$$

(11) can be simplified to

$$E'_{min}(\tau_{1,\ldots,h}) = (h+1)FC_1(\prod_1^h(U_i+(1-P_i)) - \prod_1^h(1-P_i)) \quad (13)$$

Note that (12) becomes less accurate as h increases and may be reasonable for moderate h only in situations where N is very large or C_1 is small.

We shall now examine some sample cases for insight from these computations. First, we shall assume a situation where (12) is reasonable and where n=3. A single list must include all three attributes. If three lists are permissible, then three simple indexed lists can be used. However, there are three choices if two lists are allowed; one can group any two of the attributes together.[2]

Table 1 indicates the retrieval time (computed from (13)) for each possibility in four cases. Case 1 shows one attribute with a much higher value of U than the other two. Hence, as the first three entries in the right hand portion of the table indicate, the two attributes with less query activity should be grouped together if two lists are allowed. Case 2 indicates a situation where the spread in the U's is not as large and here, it is advantageous to group the active attribute with one of the others. Case 3 is a situation where two attributes are more active than a third and they should be grouped together. The final case indicates a situation where all U's are equal. Here, the two more frequently referenced attributes should be grouped together.

However, note that in two of the four cases, one would be even better off choosing one indexed list with all three attributes, a most surprising conclusion. Also, if three lists are available, it is advantageous to use them in only one case.

---

[2]Although it is possible for an attribute to appear in two or more indices, we will not investigate that possibility.

We shall now consider a situation where (12) is not an accurate statement. In particular, if $N=10^5$, $C_1=75$ msec, and $C_2=.02$ msec, then F=26.7. Using (11) we will reconsider the four cases of Table 1. The results are indicated in Table 2.

Here, in all four cases the best choice is three indexed lists if space is available. Also, in two cases the same choice of indices should be made, if two lists are allowed, as in the situation of Table 1. In cases 1 and 4 a slight improvement is obtained by making another choice. Note that a single indexed list is never preferred.

Clearly, the examples indicate that the best choice of indexed lists varies in unexpected ways with query conditions. Moreover, retrieval times can vary markedly with the particular choice made.

Additional effort is in order on several aspects of the problem considered. First, the best choice of indices can only be made for specific examples. A search for more general results which suitably explain the above examples is needed. Also, a way to avoid the many approximations made in this paper would be desirable. Lastly, further effort should avoid some of the more unreasonable assumptions made in this simplistic analysis. Among these are the suppositions that the R's are mutually independent and that updates are ignored. However, the analytic complexity that would result might preclude obtaining any general insight except by simulation for a specific device. An example of this approach is [6].

References

1. Lowe, T.C., "The Influence of Data Base Characteristics and Usage on Direct Access File Organization", JACM 15, No. 4, October 1968, 535-548.

2. Lum, V.Y., "Multi-attribute Retrieval with Combined Indices", CACM 13, No. 11, November 1970, 660-665.

3. Mullin, J.K., "Retrieval-Update Speed Tradeoffs Using Combined Indices", CACM 14,No. 12, December 1971, 775-776.

4. Codd, E.F., "A Relational Model of Data for Large Shared Data Bases", CACM 13, No. 6, June 1970, 377-387.

5. Codd, E.F., "A Data Base Sublanguage Founded on the Relational Calculus", Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control, San Diego.

6. Senko, M.E., "Semi-Operational Evaluation of File Modeling Techniques", Information Sciences Department, IBM Research Laboratory, San Jose, February, 1971.

case	U_1	P_1	U_2	P_2	U_3	P_3	$E'(\tau_{12})$ $+E'(\tau_3)$	$E'(\tau_{13})$ $+E'(\tau_2)$	$E'(\tau_{23})$ $+E'(\tau_1)$	$E'(\tau_{123})$	$E'(\tau_1)$ $+E'(\tau_2)$ $+E'(\tau_3)$
1	1/40	1/2	1/400	1/4	1/400	1/4	34.8d	34.8d	32.7d	34.2d	32.0d
2	3/400	3/4	1/400	1/4	1/400	1/4	12.7d	12.7d	14.0d	11.6d	13.3d
3	3/400	3/4	3/400	3/4	1/400	1/4	8.8d	18.0d	18.0d	**6.4**d	18.7d
4	1/400	1/2	1/400	1/4	1/400	1/8	7.7d	8.2d	9.2d	7.9d	8.0d

$d=3FC_1/1600$

A Comparison of Retrieval Times

Table 1

case	U_1	P_1	U_2	P_2	U_3	P_3	$E(\tau_{12})$ $+E(\tau_3)$	$E(\tau_{13})$ $+E(\tau_2)$	$E(\tau_{23})$ $+E(\tau_1)$	$E(\tau_{123})$	$E(\tau_1)$ $+E(\tau_2)$ $+E(\tau_3)$
1	1/40	1/2	1/400	1/4	1/400	1/4	1.31	1.31	1.38	2.02	.95
2	3/400	3/4	1/400	1/4	1/400	1/4	1.04	1.04	1.25	1.29	.82
3	3/400	3/4	3/400	3/4	1/400	1/4	.96	1.11	1.11	1.00	.89
4	1/400	1/2	1/400	1/4	1/400	1/8	1.13	1.05	1.11	2.01	.78

(all times in seconds)

A Further Comparison of Retrieval Times

Table 2