PARTIAL-MATCH QUERIES AND FILE DESIGNS

by

WALTER A. BURKHARD

UNIVERSITY OF CALIFORNIA, SAN DIEGO
LA JOLLA, CALIFORNIA 92093

This paper is concerned with information retrieval based upon secondary keys; that is, keys which cannot in general uniquely identify a record, but can indicate certain attributes of the associated record. Partial-match retrieval deals with accessing and reading those records of a data base which match the user's query albeit the query is only partially specified. For example, suppose that the data base consists of the binary words 1010, 1110, 0011, 1101, 0010, 1111 . The response to query 1**0 where * is a don't know symbol is the set of records with keys 1010 or 1110 while the response to query 1101 is the set of records with key 1101 .

There are many practical situations in which the above file design problem is applicable. For example, an administrator of medical services might query a data base with only partial information expecting patient diagnostic information. The patent office data bank is queried under partial information during a patent search. Chemical identification is another general area which can involve searches through large data bases with only partial information. Often a retrieval task may require a best-match search so-called because the desired response for a given query is the set of records most like the query [BuKe73]. The techniques discussed here can be useful for "limited" best-match retrievals.

A record R is defined to be an ordered k-tuple $(r_1, r_2, r_3, \ldots, r_k)$ of values. Each coordinate of the k-tuple is referred to as a key and we assume that each key takes as value either 0 or 1 . Let R_k denote the set of all valid records; the cardinality of R_k is 2^k . A file F is a subset of R_k ; there are 2^{2^k} possible files.

Let Q denote the set of queries the information system is to handle. For a given file F, the subset q(F) for query $q \in Q$ denotes the desired subset of records in F . Our interest centers on partial-match queries Q_t in which t keys are specified and k-t keys are unspecified. The unspecified keys are replaced with the special place-holding symbol *. For partial-match query $q = (q_1, \ldots, q_k)$, q(F) denotes the subset of records $R = (r_1, \ldots, r_k)$ in F such that $r_i = q_i$ if q_i is either a 0 or a 1 for $1 \le i \le k$. Usually file designs only handle queries in Q_k.

There are some published partial match file designs with retrieval schemes; Knuth [Kn73] and Rivest [Ri75] review the designs.

The inverted file is a design in which a separate list is maintained of all records having a particular key value. We maintain $2 \cdot k$ lists for records in R_k . This technique, while appropriate for single-key retrieval, does not work well for partial match queries unless the number of specified keys is small. The desired set of records for a query $q \in Q_t$

This work supported in part by NSF grant GP-8557. Author's address: Computer Science Division, Department of Applied Physics and Information Science, University of California, San Diego, La Jolla, California 92093.

is the intersection of t-inverted lists. The difficulty of the task increases as t increases, while the expected number of records decreases.

More recently, Wong and Chiang [WoCh71] have proposed a file design in which every query can be responded to by taking the union of several lists. The number of list heads in a partial-match file design must be at least cardinality of R_k because the case t=k (all keys specified) must be accommodated.

A partial-match file design based on finite geometries has been proposed by Abraham, Ghosh, and Ray-Chaudhuri [AGRC68]. The file designs are applicable provided the number of keys specified is no greater that some fixed small value (for example 2).. Moreover, each record is stored in many lists.

Another partial-match file design has been proposed by Gustafson [Gu69]. Each record specifies k keys from a set of k' possible keys. The file design has two attractive properties; 1. each record is stored exactly once and 2. the expected amount of work required to answer a query decreases as the number of specified keys $t \le k$ increases.

More recently, Finkel and Bentley [FiBe74] have defined an applicable data structure referred to as the quad tree for which empirical evidence suggests that good average search times are achievable. File maintenance, especially deletion and merging, is not a computationally easy task.

Rivest [Ri75] has proposed a partial-match file scheme referred to as the associative block designs ABD . He characterizes the partial-match file designs which have minimal average search times assuming that all queries in Q_t are equally likely and then presents several file designs which are ABD's . Evidently, the ABD's are extremely difficult to determine in practice and furthermore currently exist only for even values of k .

<u>Partial Match File Schemes</u>

<u>Definition.</u> A partial match file (or PMF (k,w) for short) design is a table with k columns, $b = 2^w$ rows with entries over {0, 1, *} such that i) each row contains exactly w digits and k-w *'s , ii) given any two rows there exists at least one column in which the two rows contain differing digits.

The first condition ensures that each list has the same maximum size and the second condition guarantees that distinct lists are disjoint. We have the following PMF (3,2) as an example design.

00*	0		
01*	1	←	bucket
10*	2		addresses
11*	3		

That is row i of the table designates the keys of bucket i ; the asterisk may be either a 0 or 1 . The retrieval performance of this design is given below.

	0	1	2	3	t
average	4	8/3	5/3	1	
worse case	4	4	2	1	

A family F of PMF designs is given below which has good worst case performance. Moreover, these designs are for odd k .

<u>Definition 1.</u> For n , a natural number, let

$$F(0) = \begin{matrix} 0 \\ 1 \end{matrix} \quad n = 0 \qquad \text{and} \qquad F(n+1) = \begin{matrix} 0 \ \overrightarrow{F}(n) \ * \\ 1 \ * \ \overleftarrow{F}(n) \end{matrix} \quad n \ge 0$$

where $\overrightarrow{F}(n)$ is F(n), $\overleftarrow{F}(n)$ consists of the columns of F(n) in reverse order and 0, 1, and * each represent a column of 2^{n+1} of the specified characters.

We designate worst case behavior for these designs as $\omega(n,t)$; that is, $\omega(n,t)$ is the maximum number of lists consulted in F(n) to retrieve the desired information given that t keys are specified.

The following theorem gives the worst case performance for the F designs.

Theorem 1. For $n \geq 0$

$$\begin{aligned} \omega(n, j) &= 2^{n-j}F_{j+3} & 0 \leq j \leq n, \\ \omega(n, n+j) &= 2^{j}\, F_{n+3-2j} & 0 \leq j \leq \lceil n/2 \rceil, \\ &= 2^{n+1-j} & \lceil n/2 \rceil \leq j \leq n+1. \end{aligned}$$

where F_i is the i^{th} Fibonacci number.

Table 1 contains the performance of F(4) and a PMF (9,5) F1 similar to the PMF (3,2) presented previously. The scatter function for F1 simply extracts the first five bits of the nine within the key to use as the bucket address.

Table 1. t	0	1	2	3	4	5	6	7	8	9
Worse case F(4)	32	24	20	16	13	10	8	4	2	1
Worse case F1	32	32	32	32	32	16	8	4	2	1
Average	32	23.1	16.4	11.5	8.0	5.4	3.6	2.4	1.6	1.0

Implementation of F Designs

The F designs may be viewed as a trie; this allows an extremely straightforward search algorithm which determines which bucket contains records pertinent to a given query. The search algorithm traverses the nodes of the trie in preorder fashion. Due to the orderly structure of the F design, it is not necessary to actually store the design explicitly. Consequently, the implementation of an F design will consist of the list headers, the records themselves and the above mentioned search algorithm.

REFERENCES

[AGRC68] Abraham, C. T., S. P. Ghosh, and D. K. Ray-Chaudhuri, "File organization schemes based on finite geometries," Information and Control, 12 (February, 1968), 143-163.

[BuKe73] Burkhard, W. A. and R. M. Keller, "Some approaches to best-match file searching," CACM 16:4, (April, 1973), 230-235.

[FiBe74] Finkel, R. A. and J. L. Bentley, "Quad trees: a data structure for retrieval on composite keys," Acta Information 4, (1974), 1-9.

[Gu69] Gustafson, R. A., A Randomized Combinatorial File Structure for Storage and Retrieval Systems, Ph.D. Thesis, University of South Carolina, (1969), 92 pp.

[Kn73] Knuth, D. E., The Art of Computer Programming, Volume 3, Sorting and Searching, Addison-Wesley Publishing Company, Reading, Massachusetts, (1973).

[Ri75] Rivest, R. L., "Partial-Match Retrieval Algorithms," to appear in SIAM Computing Journal.

[WoCh71] Wong, E. and T. C. Chiang, "Canonical structure in attribute based file organization," CACM 14:9, (September, 1971), 593-59.